Formal Verification of Voting and Auction Protocols : From Privacy to Fairness and Verifiability

Jannik Dreier

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Présentée par

Jannik Dreier

Thèse dirigée par **Pr. Yassine Lakhnech**
et codirigée par **Dr. Pascal Lafourcade**

préparée au sein **du Laboratoire VERIMAG**
et de **l'Ecole Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (EDMSTII)**

Formal Verification of Voting and Auction Protocols

From Privacy to Fairness and Verifiability

Thèse soutenue publiquement le **25 novembre 2013**,
devant le jury composé de :

Dr. David Pointcheval
Ecole Normale Supérieure, CNRS, Président
Dr. Steve Kremer
INRIA Nancy – Grand Est, Rapporteur
Pr. Olivier Pereira
Université Catholique de Louvain, Rapporteur
Pr. David Basin
ETH Zürich, Examinateur
Dr. Bruno Blanchet
INRIA Paris – Rocquencourt, Examinateur
Dr. Cédric Fournet
Microsoft Research, MSR-INRIA Joint Center, Examinateur
Dr. Pascal Lafourcade
Université Joseph Fourier Grenoble, Co-Directeur de thèse

Abstract

In this document, we formally analyze security in electronic voting and electronic auctions. On-line voting is now available in several countries, for example in Estonia [Est] or parts of Switzerland [Gen13, Reg13]. Similarly, electronic auctions are increasingly used: eBay had over 112 million active users and over 350 million listings in 2012, and achieved a revenue of more than 14 billion US Dollars [Don12]. In both applications, security is a main concern, as fairness is important and money is at stake.

In the case of voting protocols, privacy is crucial to ensure free elections. We propose a hierarchy of privacy notions in the Applied π-Calculus, including different levels of coercion, special attacks such as forced-abstention attacks, and inside attackers. We also provide generalized notions for situations where votes are weighted (e.g. according to the number of shares in a company), and show that for many protocols the case with multiple coerced voters can be reduced to the case with one coerced voter. This result is made possible by a unique decomposition result we proved in the Applied π-Calculus, showing that any finite process has a unique normal form with respect to labeled bisimilarity. Moreover we provide multiple case studies illustrating how our taxonomy allows to assess the level of privacy ensured by a voting protocol.

In the case of auction protocols we also consider a hierarchy of privacy notions, and several fairness and authentication properties such as Non-Interference, Non-Cancellation and Non-Repudiation. We analyze all these properties automatically using ProVerif on three case studies, and identify several flaws. Moreover we give an abstract definition of verifiability in auctions and provide case studies in the symbolic and computational model using ProVerif and CryptoVerif respectively. Again, we identify several shortcomings, but also give a computational proof for one protocol.

Finally we explore the idea of "true bidder-verifiable auctions", i.e. auctions that can be verified by a non-expert, as the property is ensured through physical properties instead of complex cryptography. We propose two such protocols, discuss how to model the underlying physical properties, and provide a formal analysis of both protocols using ProVerif.

Keywords: Privacy, Electronic Voting, Electronic Auctions, Authentication, Fairness, Formal Verification, Symbolic Model, Computational Model, ProVerif, CryptoVerif, Applied Pi-Calculus, Unique Decomposition, Normal Form

Résumé

Dans cette thèse nous étudions formellement la sécurité des protocoles de vote et d'enchère en ligne. Le vote en ligne est utilisé en Estonie [Est] et dans certaines régions de la Suisse [Gen13, Reg13]. D'autre part, les enchères en ligne sont de plus en plus populaires : eBay comptait plus de 112 millions utilisateurs actifs et plus de 350 millions d'objets à vendre en 2012, avec un chiffre d'affaires de 14 milliards de dollars [Don12]. Dans ces deux applications, la sécurité est primordiale, à cause d'enjeux financiers et politiques.

Dans le cas des protocoles de vote, le secret du vote est crucial pour le libre choix des votants. Nous proposons une hiérarchie des notions de secret du vote par rapport à plusieurs niveaux de coercition, des attaques spécifiques (comme l'abstention forcé), et des votants corrompus. Nous généralisons ces notions pour le cas des votes pondérés (par exemple par rapport au nombre d'actions dans une société). Nous montrons aussi que sous certaines conditions le cas avec plusieurs votants sous attaque se réduit au cas avec un seul votant sous attaque. Ce résultat a été obtenu grâce à un autre résultat démontré dans le Π-calcul appliqué, montrant que tout processus fini peut se décomposer de manière unique en processus premiers. Nous illustrons notre hiérarchie sur plusieurs exemples, soulignant comment elle permet d'évaluer le niveau d'anonymat d'un protocole donné.

Dans le cas des protocoles d'enchère en ligne, nous proposons aussi une hiérarchie de notions d'anonymat, et plusieurs notions d'équité et d'authentification comme la non-interférence, la non-annulation et la non-répudiation. Nous analysons ces propriétés automatiquement à l'aide de l'outil ProVerif sur trois exemples, et découvrons plusieurs faiblesses. De plus, nous proposons une définition abstraite de la vérifiabilité, et l'appliquons sur des exemples aussi bien dans le modèle calculatoire que dans le modèle symbolique en utilisant CryptoVerif et ProVerif respectivement. Nous démontrons dans le modèle calculatoire qu'un des protocoles est vérifiable, et découvrons plusieurs faiblesses sur les autres exemples.

Finalement nous étudions le concept d'«enchères vraiment vérifiable par les enchérisseurs», c'est-à-dire des protocoles d'enchère où le bon déroulement peut être vérifié par un non-expert, car la sécurité est assurée par des moyens physiques, et non cryptographiques. Nous proposons deux tels protocoles, et une analyse formelle de ces protocoles grâce à une modélisation des propriétés physiques avec ProVerif.

Mots-clés : vie privée, vote électronique, enchère en ligne, authentification, équité, vérification formelle, modèle symbolique, modèle calculatoire, ProVerif, CryptoVerif, pi-calcul appliqué, décomposition unique, forme normale

Acknowledgments

Firstly I would like to thank all the members of my jury, in particular my reviewers Dr. Steve Kremer and Pr. Olivier Pereira, for accepting to report on this long manuscript. I would also like to express my gratitude towards the examiners Pr. David Basin, Dr. Bruno Blanchet, Dr. Cédric Fournet and Dr. David Pointcheval for their interest in my work.

I am also very grateful to my supervisors Pr. Yassine Lakhnech and Dr. Pascal Lafourcade for accepting me as a PhD student, organizing the funding, and supervising me. I have always enjoyed working with them, and without them this work would not have been possible.

Then I would to thank my collaborators, Jean-Guillaume Dumas from Laboratoire Jean Kuntzmann, for his help with Brandt's auction protocol, Cristian Ene from VERIMAG, for his help and collaboration on the π-calculus, and Hugo Jonker from Luxembourg for his collaboration on auction protocols. Jean-Guillaume was a great help in understanding the mathematics of Brandt's protocol, and he is a very enjoyable traveling companion. Cristian showed me the subtleties of process algebras, and encouraged me to continue when I was about to give up. Hugo is a great source of ideas, and a gifted organizer of productive workshops.

I also want to thank all the colleges at VERIMAG (Marie-Laure Potet, Pierre Corbineau, Nicolas Halbwachs, Susanne Graf, Claire Maiza, ...), the secretaries (Sandrine, Christine, Rosen, ...) for their help with the inevitable paperwork, the IT-Team (Jean-Noël Bouvier, Philippe Genin, Loïc Thillier) and in particular my fellow PhD students and/or office mates (Mathilde, Christian, Raphael, Ali, Eduardo, Marion, Yvan, Tommaso, ...) for making it a very nice place to work and study. I will surely remember the lunch breaks thanks to the immortal VERIFAIM (Jacques, Sophie, Marc, Valentin, Benoît, Claude, Florent, ...), and life would have been much less enjoyable without the common mountain activities: skiing and climbing (VERICLIMB).

Finally I am very grateful to my family and friends for their constant support. I want to thank my father for introducing to the enchanting worlds of mathematics, engineering and science, my mother for teaching me the fascination of languages and cultures, and my girlfriend Mathilde for simply being there.

Last but not least my thanks go to all those I forgot here...

Table of Contents

1 Introduction **5**
1.1 Introduction and Motivation 6
1.2 Related Work 8
1.3 Contributions and Organization of the Thesis 12
1.4 Publications 14

2 The Applied π-Calculus and Unique Parallel Decomposition of Processes **15**
2.1 Introduction 16
2.1.1 Outline of the Chapter 16
2.2 Syntax and Semantics 16
2.3 Observational Equivalence and Labeled Bisimilarity 21
2.4 Unique Parallel Decomposition of Processes 24
2.4.1 Related Work 24
2.4.2 Depth and Norm of Processes 25
2.4.3 Decomposition w.r.t. Strong Labeled Bisimilarity 28
2.4.4 Decomposition w.r.t. Weak Labeled Bisimilarity 36
2.5 Conclusion 45

3 eVoting **47**
3.1 Introduction 49
3.1.1 Contributions 50
3.1.2 Outline of the Chapter 51
3.2 Related Work 51
3.3 A Formal Taxonomy of Privacy in Voting 54
3.3.1 Formalizing Voting Protocols 54
3.3.2 Defining Privacy: A Modular Approach 60
3.3.3 Definitions in the Applied π-Calculus 62
3.3.4 Hierarchy 65

3.3.5 Case Studies . . . 68
3.3.5.1 Protocol by Fujioka, Okamoto and Ohta (FOO) . 68
3.3.5.2 Protocol by Okamoto . . . 77
3.3.5.3 Bingo Voting . . . 88
3.3.5.4 Protocol by Lee, Boyd, Dawson, Kim, Yang and Yoo . . . 103
3.3.5.5 Summary . . . 114
3.4 Defining Privacy for Weighted Votes . . . 115
3.4.1 Formal Definition . . . 116
3.4.2 Example: A Variant of FOO . . . 117
3.4.2.1 Adding Vote Weights . . . 117
3.4.2.2 Model and Analysis . . . 117
3.4.3 Link to Existing Definitions . . . 120
3.4.4 Including Corrupted Voters . . . 126
3.5 Multi-Voter Coercion . . . 133
3.5.1 Single-Voter Receipt-Freeness (SRF) . . . 133
3.5.2 Multi-Voter Receipt-Freeness (MRF) . . . 139
3.5.3 Single-Voter Coercion (SCR) . . . 144
3.5.4 Multi-Voter Coercion (MCR) . . . 146
3.6 Conclusion . . . 149

4 eAuctions **155**
4.1 Introduction . . . 157
4.1.1 Contributions . . . 158
4.1.2 Outline of the Chapter . . . 159
4.2 Related Work . . . 159
4.3 Fairness, Authentication and Privacy in Auctions . . . 162
4.3.1 Modeling Auction Protocols . . . 162
4.3.2 Fairness Properties . . . 166
4.3.3 Authentication Properties . . . 167
4.3.4 Privacy Properties . . . 169
4.3.4.1 Privacy . . . 170
4.3.4.2 Receipt-Freeness . . . 177
4.3.4.3 Coercion-Resistance . . . 185
4.3.5 Case Studies . . . 194
4.3.5.1 Protocol by Curtis, Pierprzyk and Seruga . . . 194
4.3.5.2 Protocol by Brandt . . . 202
4.3.5.3 Protocol by Sako . . . 207
4.3.6 Summary . . . 212
4.4 Verifiability in Auctions . . . 213
4.4.1 A Different Model of Auction Protocols . . . 213

4.4.2 Defining Verifiability . . . 214
4.4.2.1 First-Price Auctions . . . 215
4.4.2.2 Other Types of Auctions . . . 217
4.4.3 Case Studies . . . 218
4.4.3.1 Protocol by Sako . . . 218
4.4.3.2 Protocol by Curtis et al. . . . 236
4.4.4 Summary . . . 241
4.5 Towards True Bidder-Verifiable Auctions . . . 241
4.5.1 The "Cardako" Protocol . . . 242
4.5.1.1 Description . . . 242
4.5.1.2 Security Properties . . . 243
4.5.1.3 Formal Analysis . . . 244
4.5.2 The "Woodako" Protocol . . . 248
4.5.2.1 Description . . . 249
4.5.2.2 Securities Properties . . . 256
4.5.2.3 Formal Analysis . . . 257
4.5.3 Summary . . . 266
4.6 Conclusion . . . 267
4.6.1 Limitations and Future Work. . . . 268

5 Conclusion **271**
5.1 Summary . . . 272
5.2 Limitations and Directions for Future Research . . . 274

6 Résumé en Français **277**
6.1 Introduction . . . 278
6.1.1 Contributions . . . 280
6.1.2 Publications précédentes . . . 282
6.2 Le π-calcul appliqué et la décomposition unique des processus . . . 282
6.3 Les protocoles de vote . . . 283
6.3.1 Taxonomie . . . 284
6.3.2 Votes pondérés . . . 286
6.4 Les protocoles de vente aux enchères . . . 287
6.5 Conclusion . . . 289
6.5.1 Perspectives . . . 290

Bibliography **297**

Chapter 1

Introduction

In this chapter we introduce the context of the thesis and motivate our work. We also give an overview of related work, and outline the thesis and its contributions. Finally we list previous publications of preliminary results concerning the work presented in this thesis.

Contents

1.1 Introduction and Motivation 6
1.2 Related Work 8
1.3 Contributions and Organization of the Thesis 12
1.4 Publications 14

1.1 Introduction and Motivation

More and more commerce is done using the Internet, for example on-line shopping – Amazon.com achieved a revenue of more than 61 billion US Dollars in 2012 [Ama13] – or on-line auctions (eAuctions) such as eBay with a revenue of more than 14 billion US Dollars in 2012 [Don12]. Moreover administration and government rely more and more on computer system for electronic government (eGovernment) applications or electronic voting (eVoting), which is used for example in Estonia [Est] or parts of Switzerland [Gen13, Reg13].

To connect different systems or to implement such distributed applications, many protocols are developed. These protocols specify how the different participants interact, and are designed to ensure security properties (for example authentication or secrecy of the exchanged messages) as well as functional properties (for example to fulfill efficiency or real-time constraints).

However, the design of complex protocols is notoriously difficult and error-prone. One approach to tackle this problem is the use of formal methods. Formal methods rely on the use of formal models such as dedicated logics, process algebras or probabilistic arguments to analyze the security of systems or protocols. They can be used to find bugs, but also to prove that a system is secure within a given model and with respect to given security properties. The Common Criteria for Information Technology Security Evaluation [Com12a], an international standard for the certification of security critical information systems, requires formal analysis for its two highest Evaluation Assurance Levels (EALs) 6 and 7 [Com12b].

The use of formal methods has achieved many results in recent years. Since the seminal works by Dolev and Yao [DY81, DY83] and Millen [Mil84] on public key protocols, the development of the BAN-logic [BAN90], and the famous results by Lowe [Low96] on the automatic analysis of the Needham-Schroeder protocol, many weaknesses in existing and deployed protocols and standards have been identified. For example Mitchell, Shmatikov and Stern [MSS98] found anomalies in SSL (Secure Socket Layer) 3.0 using finite-state analysis. Moreover, Delaune, Kremer and Steel [DKS10] discovered several flaws in the PKCS#11 standard for cryptographic tokens using a formal model and an automated decision procedure. Similarly Smyth and Cortier [SC11] identified weaknesses concerning the voter's privacy in the Helios voting system [Adi08], again using formal analysis. More issues with Helios were discovered by Bernhard et al. [BCP+11, BPW12]. Finally the automated analysis of the YubiHSM hardware security module, designed to protect secret keys even when an intruder corrupted the rest of the machine, also revealed a flaw, leading to a redesign of the protocol [KS12].

Formal methods cannot only be used to discover attacks, but also to obtain secure and certified protocols and implementations. For example He et al. [HSD+05]

completed a modular proof of IEEE 802.11i and Transport Layer Security (TLS), one of the most widely used security protocols on the internet. More recently, a verified reference implementation of TLS was completed [BFK+13].

The main challenges however in both situations (certification and verification) remain the choice of the model, the formalization of complex security properties, and the development of automated verification tools.

In this thesis we discuss two main applications: electronic voting (eVoting) and electronic auctions (eAuction). Electronic Voting systems have been used in many countries all over the world, and even on-line voting is available in some countries such as Estonia [Est], parts of Switzerland [Gen13, Reg13] or for French expatriates [Min13]. As voting is a crucial act in modern democracies, the security requirements are high and complex, and there have been controversial discussions about the security of such systems [Par07, UK 07, Min08, Bun09].

As our second application we discuss electronic auctions. They are widely used, for example eBay had over 112 million active users and over 350 million listings in 2012 [Don12]. Since an auction includes several competing parties – multiple bidders striving for the lowest possible price, and the seller looking to achieve the highest possible price – and money is at stake, security is a major concern in such transactions as frauds are common [NTJ13].

In both applications we deal with complex systems and non-trivial properties such as Privacy, Fairness, and Verifiability. Privacy can simply mean the secrecy of the vote (or bid), but also unlinkability of the voter and his vote, or anonymity of the winning bidder. Fairness often is related to Privacy, since for example preliminary results in an election can influence the choice of the remaining voters. Yet fairness also includes robustness against cheating, for example by ensuring that a voter cannot vote twice, or that a bidder cannot modify or cancel his bid (depending on the rules of the auction). Finally Verifiability ensures that a participant can verify the correct behavior of some (un-)trusted parties, for example of the tallier in a voting protocol. In such a case a voter can check the correctness of the result after the election is over, without having to trust the authorities. This is particularly interesting if the systems are complex and difficult to understand for the participants, as the verification can be noticeably easier than the protocol execution.

As the definitions in natural language tend to contain imprecisions, one of the main challenges in formal verification is the development of formal definitions for the different properties. These definitions need to be precise, and should ideally also be suitable for automated verification, as human proofs tend to be error-prone as complexity increases. Finally the definitions should also be as complete (i.e. containing all aspects and covering all possible attacks) as possible, yet this is often difficult to achieve – how to include types of attacks which are not yet

known?

The goal of this thesis is to propose models and definitions to express and verify such security properties in both contexts. For voting we focus on Privacy properties including Receipt-Freeness and Coercion-Resistance. Receipt-Freeness means that a voter cannot construct a receipt proving to an attacker that he voted for a particular candidate, to prevent vote-buying. Coercion-Resistance allows a voter to vote for a candidate of his choice, even if the attacker tries to force him to vote for a certain candidate by interacting with him throughout the entire voting process. For auctions, we propose models and definitions for Privacy (also including Receipt-Freeness and Coercion-Resistance), Fairness and Verifiability. We also discuss several case studies to test our models on existing examples. In particular we prove one protocol secure with respect to our model and definitions, but also identify multiple flaws with other protocols. Moreover we provide a theoretical result in the Applied π-Calculus which allows us to show that some privacy notions coincide.

1.2 Related Work

Here we give only a high-level overview of the different approaches, models and tools used for formal verification of protocols. In each chapter we discuss in more detail the work related to the chapter's content.

Now we start by discussing the symbolic and computational models as well as their relationship, then we give an overview of the techniques and tools used in the symbolic model. Finally we also give a short overview over tools supporting computationally sound proofs.

In general, we distinguish two main approaches for the formal verification of protocols: the *symbolic* and *computational* models.

The Symbolic Model. In the symbolic model, cryptographic building blocks such as encryption, signatures, commitments etc. are treated as black boxes and assumed to be perfectly secure. This means that for example the decryption of an encrypted message is only possible if one knows the key, or that the only way generate a valid signature is using the secret key. Usually the intruder has full control of the network, i.e. can intercept, create, modify and delete messages, which are terms. Then there are rules specifying which operations the intruder can apply on these terms. Such a formal model was first proposed by Dolev and Yao [DY81, DY83][1], and their technique allows for automatic verification of reachability properties such as "Can the intruder access this secret value?".

[1] An attacker with similar capabilities was previously discussed informally, e.g. in [NS78].

The Computational Model. In the computational model on the contrary, messages are bitstrings, and the intruder is a (usually probabilistic polynomial-time) Turing-machine. Security properties are then typically defined as games played by the intruder. If he manages to win the game, he breaks the property. The goal is then to prove that the probability of the adversary winning the game is very low, for example close to random guessing. Usually one shows that his advantage is negligible, i.e. smaller than the inverse of any positive polynomial function of the security parameter (for example the key length). The common technique to prove such statements is a proof by reduction, i.e. one proves that if the adversary is able to win the security game with a high probability, he can break a supposedly hard mathematical problem such as the factorization of large numbers or computing the discrete logarithm. These proofs are often realized as a "sequence of games", where the first game corresponds to the initial security game, and the last game corresponds to the supposedly hard problem, which concludes the proof. In between these two games there can be many intermediate games, and each two games in the sequence differ only in a small detail. This is to decompose the proofs, as one proves that in each game the adversary has the same advantage as in the previous one, or that there is only a negligible difference. Such computational proofs tend to be more difficult to automate, in particular the sequence of games is often difficult to generate automatically.

Links between the Symbolic and Computational Model. Although both models are quite different, there are soundness results showing a link between both. In 2000 Abadi and Rogaway [AR00] showed that a protocol using encryption shown secure in the symbolic model with respect to certain properties, is also secure in the computational model with respect to the computational equivalent of these properties. Since then many more results have been obtained for various properties and combinations of different cryptographic primitives (e.g. [BP05, BDK07, CLC08, KT09b, BMU12, CLHKS12, BBU13] or [CKW11] for an overview), however also some limitations of this approach were discovered [BPW06].

Symbolic Techniques and Tools. Within the symbolic model there is a big variety of techniques. There is the initial Dolev-Yao model [DY83] based on deduction rules, there are logics such as the BAN-logic [BAN90] to model authentication, process algebras such as the Spi-Calculus [AG97] or the Applied π-Calculus [AF01], and methods based on typing such as F7 [BFG10, BBF$^+$11].

Although protocol verification problems are often undecidable in the general case [EG82, DLMS04] – in particular with unbounded message length and and an unbounded number of instances in parallel – there are various tools supporting automated protocol verification in different symbolic models. To deal with

such undecidable problems, they employ various techniques: approximations, restrictions to the case with a bounded number of instances or limited attacker capabilities, or algorithms that do not always terminate. We now discuss several tools that employ such techniques.

AVISPA [ABB+05] is a tool supporting four different back-ends for protocol verification with different techniques:

— the On-the-fly Model-Checker (OFMC) [BMV05] works by exploring the transition system of the protocol in a demand-driven way.

— the Constraint-Logic-based Attack Searcher (CL-AtSe) [Tur06] analyzes a bounded number of instances of the protocol using constraint logic.

— the SAT-based Model-Checker (SATMC) [AC04] translates the possible (finite) protocol runs into a logical formula, which is then solved by a SAT-solver.

— Finally Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP) [BHKO04] is the only back-end supporting an unbounded number of sessions by over-approximation.

AVISPA can analyze authentication as well as (weak) secrecy, modeled as reachability. We distinguish weak secrecy, i.e. "Can the secret value be computed by an adversary?", from strong secrecy, i.e. "Can the adversary distinguish two instances which only differ in the secret value?". The latter is stronger in the sense that the adversary might be incapable of computing the exact value, but still be able to test if it is equal to a specific value, for example zero. Strong secrecy can be expressed using observational equivalence. AVISPA was extended in the AVANTSSAR [AAA+12] project, but still relies on the same (although improved) back-ends.

Scyther [Cre08a, Cre08b] verifies protocol using a symbolic backwards search based on patterns. It supports bounded and unbounded number of runs, however it does not always conclude for the unbounded case. In such a situation however it still gives a verdict for the bounded case. It supports authentication and weak secrecy (i.e. reachability) properties. A variant of Scyther can also be used to generate machine-checked proofs [MCB10].

Recently the Tamarin prover [SMCB12, MSCB13] was developed, it supports verification of security protocols with an unbounded number of sessions. The security properties can be expressed with respect to a special subset of (temporal) first-order logic. It also supports Diffie-Hellman exponentiation and a user-defined subterm-convergent rewriting theory.

ProVerif is an automatic verification tool based on Horn clauses originally developed by Bruno Blanchet [Bla01]. It features a mechanical translation from the Applied π-Calculus to Horn clauses, so it can directly verify a protocol given in the Applied π-Calculus. ProVerif uses approximations and is sound but not

complete, and sometimes does not terminate. Originally designed to verify weak secrecy, it was extended to support correspondence properties between events to verify authentication properties [Bla02] and strong secrecy [Bla04a]. Another extension was made to reconstruct attack traces if possible to help identifying false attacks [AB05b]. Finally support for the verification of equivalences [BAF08] was added, which was recently improved to obtain a finer approximation [CB13b]. It also supports user-defined equational theories [AB05a].

AKISS [CCK12] is a recent tool that allows to prove trace equivalence properties for bounded processes, featuring user-defined equational theories. It is based on KISS [CDK12], a tool allowing to prove static equivalence for complex equational theories.

For a comparison of the performance of different tools (in particular Scyther, ProVerif and the different AVISPA backends) see [CLN09]. We chose ProVerif for many verification tasks throughout the thesis because of its performance, its support for authentication and equivalence properties (to analyze different notions of Privacy) and its support of user-defined equational theories (to model special cryptographic operations as well as the properties of physical objects).

Computational Tools. There are also tools working in the computational model. CryptoVerif [Bla06a, BP06, Bla07, Bla08] is a tool supporting proofs based on sequences of games. It allows for manual or (semi-)automatic proofs. CryptoVerif provides an exact upper bound on the probability of the adversary winning the game in terms of how often he calls the oracles (called *concrete security*). Recently it was extended to also directly provide implementations based on the verified protocol model [CB13a].

CertiCrypt [BGZB09] allows for certified cryptographic proofs in the computational model using the Coq proof assistant [Coq, CH88]. It also uses a sequence of games and provides strong guarantees, however the proofs are manual and tend to be cumbersome. To address this, EasyCrypt [BGHZB11] was developed. In EasyCrypt the user specifies the sequence of games, and the tool tries to automatically prove the equivalences between the games using SMT[2]-solvers. When it succeeds, it also provides a proof that can be checked using the Coq proof assistant.

A different approach is used by the Computational Indistinguishability Logic (CIL) [BDKL10], allowing to formalize computational indistinguishability proofs based on a special logic, which can also be implemented in the Coq assistant.

A recent extension to F7 also permits computationally sound proofs [FKS11].

In our computational proofs, we use CryptoVerif because of its high grade of automation, and syntactical similarity to ProVerif, which allows us to re-use big parts of the models.

[2]Satisfiability Modulo Theories

1.3 Contributions and Organization of the Thesis

In Chapter 2, we recall the Applied π-Calculus, which is used throughout the thesis. We also present two results on the unique decomposition of processes. We start by defining the subclasses of *finite* and *normed* processes, i.e. processes where all complete traces are finite, and processes where there exists at least one complete finite trace respectively. In the first result we show that any process in the subclass of normed processes can be rewritten as a composition of parallel factors in a unique (up to permutation and strong labeled bisimilarity) way, i.e. we can decompose a normed process P into factors $P_1|\ldots|P_n$, where each P_i is prime in the sense that it cannot be further decomposed without obtaining trivial (i.e. equivalent to 0) factors. In our second result, we show that similarly any finite process can be rewritten in a unique (up to permutation and weak labeled bisimilarity) way as a composition of parallel prime factors. Such results are handy as they provide a normal form, and a cancellation result in the sense that $A|B \sim C|B$ implies $A \sim C$. We use our second decomposition result in Chapter 3 in a proof showing the equivalence of two privacy notions.

In Chapter 3, we discuss privacy in electronic voting (eVoting). In a first contribution we provide a formal taxonomy of privacy in the Applied π-Calculus. This taxonomy accounts for different attacker capabilities (such as an inside or outside attacker), particular attacks (forced-abstention attacks) as well as the level of coercion possible by the attacker (simple privacy, receipt-freeness, or full coercion-resistance). We apply this taxonomy on several existing protocols (the protocol by Fujioka et al. [FOO92], the protocol by Okamoto [Oka96], the protocol by Lee et al [LBD$^+$03], and Bingo Voting [BMQR07]) to illustrate the different levels of privacy achieved by these protocols.

In our second contribution, we generalize the privacy definition to accommodate protocols with weighted votes, for example according to the percentage of shares in a company. In such a case the previous definitions based on two voters swapping their votes are unsuitable as swapping votes can lead to different outcomes and hence trivially distinguishable situations. Our solution is to abstract away from the result, and simply consider all distributions of votes giving the same result. We apply these new notions on a protocol implementing weighted votes (based on the protocol by Eliasson and Zúquete [EZ06]). We also establish precise links between these new notions and the notions from our taxonomy: we can show that the corresponding notions coincide if the votes are not weighted. In a next step, we can also show that if the protocol ensures a certain modularity condition (which is the case for most of our examples), single-voter coercion (i.e. only one voter under coercion by the attacker) and multi-voter coercion (i.e. several voters simultaneously under coercion) are equivalent. This means

that a modular protocol ensuring single-voter coercion-resistance also ensures multi-voter coercion-resistance. We can also show that for modular protocol we do not need to consider corrupted voters, a situation including corrupted voters can be reduced to a situation without corrupted voters.

In Chapter 4, we consider electronic auctions (eAuctions). In the first part, we provide formal definitions of authentication properties such as Non-Repudiation and Non-Cancellation, fairness properties such as Weak or Strong Non-Interference and "Highest Price Wins" as well as different notions of privacy and anonymity, including receipt-freeness and coercion-resistance in the Applied π-Calculus. We then consider three case studies: The protocol by Curtis et al. [CPS07], the protocol by Brandt [Bra06] and the protocol by Sako [Sak00]. We identify several problems for the first two protocols automatically using ProVerif, and provide automated proofs for all properties except receipt-freeness and coercion-resistance for the latter protocol.

In the second part of the chapter we analyze verifiability in eAuctions. We introduce a high-level model and definition, which allows for instantiations in the symbolic and computational model. Then we provide two case studies: the protocols by Sako and Curtis et al. For the protocol by Sako, we give a symbolic proof in the Applied π-Calculus with help of ProVerif (and some manual generalizations), and also a computational proof using CryptoVerif (and one manual proof). For the protocol by Curtis et al. we use ProVerif to analyze verifiability, and identify several shortcomings.

In the last part, we explore the idea of "true bidder-verifiable auctions", i.e. auction protocols that achieve verifiability without relying on complex cryptography, and can hence be verified without any specialist knowledge. To achieve such a property we propose to exploit the physical properties of certain objects, and develop two protocols inspired by Sako's protocol. The first one, called "Cardako", only uses office material, i.e. cardboard and envelopes. The second one, called "Woodako", uses a wooden box, that determines the winner in a private, secure and verifiable way. Although these protocols have their limitations with respect to scalability, they illustrate how we can realize secure auctions by exclusively relying on physical objects and their properties. We also discuss how we can apply the formal models and definitions we developed before on these protocols, and investigate a first possibility using special equational theories in ProVerif to model the physical properties. Using this model, we can automatically verify both protocols, and show that they achieve the desired security properties.

We sum up our results in Chapter 5 and discuss directions for future work.

1.4 Publications

Much of the work presented in this thesis has already been published at different conferences.

The unique decomposition results of Chapter 2 were presented at FoSSaCS 2013 [DELL13]. Preliminary results of the work presented in Chapter 3 were published at several occasions: the notion of Vote-Independence was presented at FPS 2011 [DLL11], the hierarchy of privacy notions at ICC-SFCS 2012 [DLL12b] and the work on weighted votes and multi-voter coercion at ESORICS 2012 [DLL12a]. Many of the results of Chapter 4 have also been presented before: the work on authentication, fairness and privacy at POST 2013 [DLL13], and the definition of verifiability at ASIACCS 2013 [DJL13].

Although not included in this thesis, we also provided a detailed cryptanalysis of Brandt's auction protocol [Bra06], identifying several shortcomings. This work was presented at Africacrypt 2013 [DDL13].

Chapter 2

The Applied π-Calculus and Unique Parallel Decomposition of Processes

The Applied π-Calculus is a process calculus designed for the verification of cryptographic protocols. In this chapter we recall its syntax and the semantics, as it is used throughout the rest of the thesis. We also present two results concerning the unique parallel decomposition of processes. In the first result we show that any normed process can be rewritten as a composition of (prime) parallel factors in a unique way up to strong labeled bisimilarity, i.e. we can decompose a finite process P into factors $P_1|\ldots|P_n$. In our second result we show that similarly any finite process can be decomposed uniquely up to (weak) labeled bisimilarity.

Contents

2.1 Introduction 16
2.1.1 Outline of the Chapter 16
2.2 Syntax and Semantics 16
2.3 Observational Equivalence and Labeled Bisimilarity . . . 21
2.4 Unique Parallel Decomposition of Processes 24
2.4.1 Related Work 24
2.4.2 Depth and Norm of Processes 25
2.4.3 Decomposition w.r.t. Strong Labeled Bisimilarity 28
2.4.4 Decomposition w.r.t. Weak Labeled Bisimilarity 36
2.5 Conclusion 45

2.1 Introduction

Process Algebras or Calculi are used to formally model and analyze distributed systems. Famous examples include the Calculus of Communicating Systems (CCS) due to Milner [Mil89], or Basic Parallel Processes (BPP) [Chr93]. These calculi contain basic operations such as emission and reception of messages as well as parallel composition or interleaving. In an extension to CCS, Milner, Parrow and Walker developed the π-Calculus [MPW92], which also features channel passing and scope extrusion. Abadi and Fournet [AF01] subsequently proposed the Applied π-Calculus, a variant of the π-Calculus designed for the verification of cryptographic protocols. It additionally features equational theories and active substitutions.

2.1.1 — Outline of the Chapter

In the next section we recall the syntax and semantics of the Applied π-Calculus. In Section 2.3 we present several equivalence and bisimilarity notions. We then discuss related work concerning unique decomposition of processes in Section 2.4.1, and define the depth and norm of a process in Section 2.4.2. Finally we present our unique decomposition for strong and weak bisimilarity in Section 2.4.3 and 2.4.4, respectively. Finally we conclude the chapter.

2.2 Syntax and Semantics

The Applied π-Calculus relies on a *type* or *sort* system for terms. It includes a set of *base* types such as `Integer`, `Key` or `Data`. Additionally, if τ is a type, then `Channel`$\langle\tau\rangle$ is a type (intuitively the type of a channel transmitting terms of type τ).

We suppose a *signature* Σ of functions, which consists of a finite set of *function symbols* with the associated arity and sorts. For example `enc`($message$, key), `dec`($message$, key) are of arity two with two parameters of sorts `Data` and `Key`, returning a value of type `Data`. A function with arity zero is a constant.

$M, N :=$		terms
	a, b, c, n, m, k	names
	x, y, z	variables
	$f(M_1, \ldots, M_l)$	function application

Figure 2.1 – Grammar for terms

Terms in the Applied π-Calculus are combinations of *names* (which typically correspond to data or channels), *variables* and function symbols from the *signature* Σ following the grammar depicted in Figure 2.1. These combinations have to

be correct with respect to arity and sorts of the function symbols, variables and names. Variables and names can have any type, and functions take and return only values of base types. We assume infinite sets of names and variables.

Functions typically include encryption and decryption, hashing, signing and so on. Equalities are modeled using an equational theory E which defines a relation $=_E$. A classical example, which describes the correctness of symmetric encryption, is $\mathtt{dec}(\mathtt{enc}(message, key), key) =_E message$. To simplify the notation we sometimes omit the subscript $_E$ if this is clear from the context.

Tuples can be implemented e.g. using a function $\mathtt{tuple}_n(M_1, \ldots, M_n)$ and the equations

$$\forall i \colon \mathtt{proj}_i(\mathtt{tuple}_n(M_1, \ldots, M_n)) = M_i$$

To simplify notation we also write $(M_1, \ldots, M_n)$ for $\mathtt{tuple}_n(M_1, \ldots, M_n)$, this assumes the function $\mathtt{tuple_n}$ and the destructors $\mathtt{proj}_i$ with the equations as defined above.

$P, Q :=$	plain processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction ("new")
$\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$	conditional (M, N terms)
$\mathtt{in}(u, x).P$	message input
$\mathtt{out}(u, M).P$	message output

Figure 2.2 – Grammar for Plain Processes

$A, B, P, Q :=$	active processes
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^M/_x\}$	active substitution

Figure 2.3 – Grammar for Extended Processes

There are two types of processes in the Applied π-Calculus: *plain processes* and *extended* or *active processes*. *Plain processes* are constructed using the grammar depicted in Figure 2.2. The null process 0 does nothing, the parallel composition $P|Q$ executes P and Q in parallel, and the replication $!P$ executes infinitely many copies of P in parallel. $\nu n.P$ creates a new, private name n and continues as P. $\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$ behaves as P if $N =_E M$ or as Q otherwise. Note the equality with respect to the equational theory, and that we require M and N to have the same type. The process $\mathtt{in}(u, x).P$ inputs a message on channel u,

assigns it to the variable x of type τ_x and continues as P. We assume that u is of type $\texttt{Channel}\langle\tau_x\rangle$. Finally $\texttt{out}(u, M).P$ outputs M (of type τ_M) on channel u and continues as P. Again, u has to be of of type $\texttt{Channel}\langle\tau_M\rangle$.

Active or *extended processes* are plain processes or active substitutions as shown in Figure 2.3. Note that the Applied π-Calculus does not include the "+"-operator which implements a nondeterministic choice, yet we can implement something similar using a restricted channel (see Example 7 on page 28). For more details on encoding the operator with respect to different semantics, see [NP00, PH05].

The substitution $\{^M/_x\}$ replaces the variable x with a term M. Note that we do not allow two active substitutions to define the same variable, as this might lead to situations with unclear semantics. We also require substitutions to be well-sorted and circle-free, and only allow active substitutions on variables of base sorts. We denote by $fv(A)$, $bv(A)$, $fn(A)$, $bn(A)$ the *free variables*, *bound variables*, *free names* or *bound names* respectively. A name n is bound if it is in the scope of a restriction νn, a variable x is bound if it is in the scope of a restriction νx or of an input $\texttt{in}(u, x)$. All unbound names and variables are free.

As an additional notation we write $\nu S.A$ for $\nu s_1.\nu s_2 \ldots \nu s_n.A$ where $s_1, \ldots s_n$ are the elements of a set of variables and names S. By abuse of notation we sometimes leave out ".0" at the end of a process. We also write A^k for $A|\ldots|A$ (k times), in particular $A^0 = 0$ as 0 is the neutral element of parallel composition. Inspired by ProVerif's syntax, we write $\texttt{let}\ x = M\ \texttt{in}\ P$ for $\nu x.\,(P\,\{^M/_x\})$, or even $\texttt{let}\ (x_1, \ldots, x_n) = (M_1, \ldots, M_n)\ \texttt{in}\ P$ for $\nu x_1 \ldots \nu x_n.\,(P\,\{^{M_1}/_{x_1}, \ldots, ^{M_n}/_{x_n}\})$.

The *frame* $\Phi(A)$ of an active process A is obtained by replacing all plain processes in A by 0. This frame can be seen as a representation of what is statically known to the environment about a process. The domain $dom(\Phi)$ of a frame Φ is the set of variables for which Φ defines a substitution. By abuse of notation, we also write $dom(A)$ to denote the domain of the frame $\Phi(A)$ of an active process A. Note that $dom(A) \subseteq fv(A)$, and that as we cannot have two active substitutions for the same variable, $P = Q|R$ implies $dom(P) = dom(Q) \cup dom(R)$ and $dom(Q) \cap dom(R) = \emptyset$. A frame or process is *closed* if all variables are bound or defined by an active substitution. An evaluation context $C[\_]$ denotes an active process with a hole for an active process that is not under replication, a conditional, an input or an output.

The semantics of the calculus presupposes a notion of *Structural Equivalence* ($\equiv$), which is defined as the smallest equivalence relation on extended processes that is closed under application of evaluation contexts, α-conversion on bound names and bound variables such that the rules in Figure 2.4 on the next page hold. Note the contagious nature of active substitutions: by rule SUBST they apply to any parallel process.

Example 1 *Consider the following running example, where x and y are variables,*

PAR-0	$A\|0 \equiv A$	
PAR-A	$A\|(B\|C) \equiv (A\|B)\|C$	
PAR-C	$A\|B \equiv B\|A$	
NEW-0	$\nu n.0 \equiv 0$	
NEW-C	$\nu u.\nu v.A \equiv \nu v.\nu u.A$	
NEW-PAR	$A\|\nu u.B \equiv \nu u.(A\|B)$	if $u \notin fn(A) \cup fv(A)$
REPL	$!P \equiv P\|!P$	
REWRITE	$\{^M/_x\} \equiv \{^N/_x\}$	if $M =_E N$
ALIAS	$\nu x.\,\{^M/_x\} \equiv 0$	
SUBST	$\{^M/_x\}\,\|A \equiv \{^M/_x\}\,\|A\,\{^M/_x\}$	

Figure 2.4 – Structural Equivalence

and c, d, k, l, m and n are names:

$$P_{ex} = \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\}\,|\mathtt{out}(c, enc(n,k))|\mathtt{out}(d,m)|\mathtt{in}(d,x).\mathtt{out}(c,x))$$

We have $dom(P_{ex}) = \{y\}$, $fv(P_{ex}) = \{y\}$, $bv(P_{ex}) = \{x\}$, $fn(P_{ex}) = \{n, c\}$, $bn(P_{ex}) = \{k, l, m, d\}$ *and*

$$\Phi(P_{ex}) = \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\}\,|0|0|0) \equiv \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\})$$

Internal Reduction ($\xrightarrow{\tau}$) is the smallest relation on extended processes closed by structural equivalence and application of evaluation contexts such that the rules in Figure 2.5 hold. Note that in accordance with the original notations [AF01], we sometimes omit the labels τ_c, τ_t and τ_e, and write $P \to P'$ for $P \xrightarrow{\gamma} P'$ with $\gamma \in \{\tau_c, \tau_t, \tau_e\}$. We also write $P \to^* P'$ for $P \to \ldots \to P'$.

COMM	$\mathtt{out}(a,x).P \mid \mathtt{in}(a,x).Q$	$\xrightarrow{\tau_c}$	$P \mid Q$
THEN	$\mathtt{if}\ M = M\ \mathtt{then}\ P\ \mathtt{else}\ Q$	$\xrightarrow{\tau_t}$	P
ELSE	$\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$	$\xrightarrow{\tau_e}$	Q
	for any ground terms such that $M \neq_E N$		

Figure 2.5 – Internal Reduction

Interactions of extended processes are described using labeled operational semantics ($\xrightarrow{\alpha}$, see Figure 2.6 on the following page), where α can be an input or an output of a channel name or variable of base type, e.g. $\mathtt{out}(a, u)$ where u is a variable or a name.

Labeled *external transitions* are not closed under evaluation contexts. Note that a term M (except for channel names and variables of base type) cannot be output directly. Instead, we have to assign M to a variable, which can then be output.

IN	$\mathtt{in}(a,x).P \xrightarrow{\mathtt{in}(a,M)} P\{^M/_x\}$
OUT-ATOM	$\mathtt{out}(a,u).P \xrightarrow{\mathtt{out}(a,u)} P$
OPEN-ATOM	$\dfrac{A \xrightarrow{\mathtt{out}(a,u)} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.\mathtt{out}(a,u)} A'}$
SCOPE	$\dfrac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$
PAR	$\dfrac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$
STRUCT	$\dfrac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}$

Figure 2.6 – Labeled semantics

Example 2 *Consider our running example process P_{ex}.*

$$P_{ex} = \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)) \mid \mathtt{out}(d,m) \mid \mathtt{in}(d,x).\mathtt{out}(c,x))$$

Using an internal reduction, we can execute the following transition[1]*:*

$$\begin{aligned}
P_{ex} &= \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)) \mid \mathtt{out}(d,m) \mid \mathtt{in}(d,x).\mathtt{out}(c,x)) \\
&\equiv \nu k.\nu l.\nu m.\nu d.(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)) \mid \boldsymbol{\nu x.(\{^m/_x\})} \mid \mathtt{out}(d,m) \mid \\
&\quad \mathtt{in}(d,x).\mathtt{out}(c,x)) && \textit{by PAR-0, ALIAS} \\
&\equiv \nu k.\nu l.\nu m.\nu d.(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)) \mid \nu x.\boldsymbol{(}\,\{^m/_x\} \mid \mathtt{out}(d, \boldsymbol{x}) \mid \\
&\quad \mathtt{in}(d,x).\mathtt{out}(c,x)\boldsymbol{)}) && \textit{by SUBST, NEW-PAR} \\
&\xrightarrow{\tau_c} \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)) \mid \nu x.\,(\{^m/_x\} \mid \mathtt{out}(c,x))) \\
&\equiv \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\} \mid \mathtt{out}(c, enc(n,k)) \mid \mathtt{out}(c, \boldsymbol{m})) \\
& && \textit{by SUBST, ALIAS, NEW-PAR, PAR-0}
\end{aligned}$$

Similarly, we can also execute an external transition:

$$\begin{aligned}
P_{ex} &\equiv \nu k.\nu l.\nu m.\nu d.(\{^l/_y\} \mid \nu z.\,(\{^{enc(n,k)}/_z\} \mid \mathtt{out}(c,z)) \mid \mathtt{out}(d,m) \mid \\
&\quad \mathtt{in}(d,x).\mathtt{out}(c,x)) \\
&\xrightarrow{\nu z.\mathtt{out}(c,z)} \nu k.\nu l.\nu m.\nu d.\,(\{^l/_y\} \mid \{^{enc(n,k)}/_z\} \mid \mathtt{out}(d,m) \mid \mathtt{in}(d,x).\mathtt{out}(c,x))
\end{aligned}$$

We can also see that external transitions are not closed under evaluation contexts: by rule SCOPE $\nu c.P_{ex}$ cannot execute the transition $\xrightarrow{\nu z.\mathtt{out}(c,z)}$ any more.

[1] Here and throughout the rest of the thesis we mark the differences between the steps in **bold** for better readability.

2.3 Observational Equivalence and Labeled Bisimilarity

The Applied π-Calculus has two equivalence notions for processes: *Observational Equivalence* and *Labeled Bisimilarity*. They can be used to express strong secrecy or other privacy properties. Let $A \Downarrow a$ denote that A can send a message on the channel a, i.e. when $A \rightarrow^* C[out(a, M).P]$ for some evaluation context $C[\_]$.

Definition 1 (Observational Equivalence [AF01]) Observational Equivalence *($\approx$) is the largest symmetric relation relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\ \mathcal{R}\ B$ implies:*

1. *if $A \Downarrow a$, then $B \Downarrow a$*
2. *if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A'\ \mathcal{R}\ B'$ for some B'*
3. *$C[A]\ \mathcal{R}\ C[B]$ for all closing evaluation contexts $C[\_]$*

The intuition is that two processes are observationally equivalent if each output or internal transition of one processes can be simulated by the other, and this holds for any context (corresponding for example to an attacker).

Example 3 *Consider the following processes, where f is a function of arity one:*

$$\begin{aligned} P_0 &= \nu a.\mathtt{out}(c, a) \\ P_1 &= \nu a.\nu d.(\mathtt{out}(d, a)|(\mathtt{in}(d, y).\mathtt{out}(c, y))) \\ P_2 &= \nu a.\nu d.(\mathtt{out}(d, a)|(\mathtt{in}(d, y).\mathtt{out}(c, (y, f(y))))) \end{aligned}$$

Then we have $P_0 \approx P_1$ as $P_0 \Downarrow c$ and $P_1 \Downarrow c$, $P_1 \rightarrow \nu a.\nu d.\mathtt{out}(c, a) \equiv P_0$, but neither $P_0 \approx P_2$ nor $P_1 \approx P_2$ as P_2 outputs a tuple instead of a single value, which can be tested by a context.

As Observational Equivalence can be difficult to prove due to the all-quantified context, we often use *Labeled Bisimilarity* instead. Labeled Bisimilarity is defined using the notion of *Static Equivalence*, which is based on the equivalence of two terms in a given frame.

Definition 2 (Equivalence in a Frame [AF01]) *Two terms M and N are* equal in the frame ϕ*, written $(M = N)\phi$, if and only if for any names $\tilde{n}$ and substitution σ such that $\phi \equiv \nu\tilde{n}.\sigma$ and $\{\tilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset$ we have $M\sigma =_E N\sigma$.*

Note that any frame ϕ can be written as $\nu\tilde{n}.\sigma$ modulo structural equivalence, i.e. using rule NEW-PAR.

Definition 3 (Static Equivalence ($\approx_s$) [AF01]) *Two closed frames ϕ and ψ are* statically equivalent*, written $\phi \approx_s \psi$, when $dom(\phi) = dom(\psi)$ and when for*

all terms M and N we have $(M = N)\phi$ if and only if $(M = N)\psi$. Two extended processes A and B are statically equivalent ($A \approx_s B$) if their frames are statically equivalent.

The intuition behind this definition is that two processes are statically equivalent if the messages exchanged previously with the environment cannot be distinguished with respect to the equational theory, i.e. all operations on both sides were indistinguishable. Note that this only concerns what is statically known to the environment, not the possible interactions.

Example 4 ([AF01]) *Consider the following frames, where f and g are two functions with no equations (corresponding intuitively to two independent one-way hash functions):*

$$\begin{aligned} \phi_0 &= \nu k.\, \{^{k}/_{x}\} \,|\nu s.\, \{^{s}/_{y}\} \\ \phi_1 &= \nu k.\, \{^{f(k)}/_{x}, {}^{g(k)}/_{y}\} \\ \phi_2 &= \nu k.\, \{^{k}/_{x}, {}^{f(k)}/_{y}\} \end{aligned}$$

Then $\phi_0 \approx_s \phi_1$, but $\phi_1 \not\approx_s \phi_2$ and $\phi_1 \not\approx_s \phi_2$ as $(f(x) = y)\phi_2$, but neither $(f(x) = y)\phi_0$ nor $(f(x) = y)\phi_1$. Intuitively this expresses that the attacker is unable to distinguish the output of two independent one-way hash functions from two random values. In the last frame there is a link between the two values that an attacker can check, making it distinguishable from the other two frames.

We can then define (Weak)[2] Labeled Bisimilarity.

Definition 4 ((Weak) Labeled Bisimilarity ($\approx_l$) [AF01]) (Weak) Labeled Bisimilarity *is the largest symmetric relation $\mathcal{R}$ on closed active processes, such that $A\ \mathcal{R}\ B$ implies:*

1. *$A \approx_s B$,*
2. *if $A \to A'$, then $B \to^* B'$ and $A'\ \mathcal{R}\ B'$ for some B',*
3. *if $A \xrightarrow{\alpha} A'$ and $fv(\alpha) \subseteq dom(A)$ and $bn(\alpha) \cap fn(B) = \emptyset$, then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A'\ \mathcal{R}\ B'$ for some B'.*

As hinted above, Labeled Bisimilarity is often easier to prove than Observational Equivalence since there is no quantification over all contexts. However Observational Equivalence and Labeled Bisimilarity do not coincide if active substitutions are allowed on variables of channel type [Liu11, LL12], this being the reason why we restrict active substitutions to variables of base sort. In this case, (Weak) Labeled Bisimilarity coincides with observational equivalence [Liu11], and is thus is closed under the application of evaluation contexts.

In our work on unique decomposition of processes we also consider a stronger version of labeled bisimilarity.

[2]Originally this bisimilarity notion was only called "Labeled Bisimilarity" by Abadi and Fournet [AF01], however we also call it "Weak Labeled Bisimilarity" to distinguish it from "Strong Labeled Bisimilarity".

Definition 5 (Strong Labeled Bisimilarity ($\sim_l$)) Strong Labeled Bisimilarity *is the largest symmetric relation $\mathcal{R}$ on closed active processes, such that $A\ \mathcal{R}\ B$ implies:*

1. *$A \approx_s B$,*
2. *if $A \to A'$, then $B \to B'$ and $A'\ \mathcal{R}\ B'$ for some B',*
3. *if $A \xrightarrow{\alpha} A'$ and $fv(\alpha) \subseteq dom(A)$ and $bn(\alpha) \cap fn(B) = \emptyset$, then $B \xrightarrow{\alpha} B'$ and $A'\ \mathcal{R}\ B'$ for some B'.*

This notion is stronger than Weak Labeled Bisimilarity in the sense that each step on one side has to be matched by exactly one on the other side, whereas in the case of Weak Labeled Bisimilarity a single transition could be simulated using several (internal) transitions.

Example 5 *Consider again the processes from Example 3 on page 21, where f is a function:*

$$\begin{aligned} P_0 &= \nu a.\mathtt{out}(c,a) \\ P_1 &= \nu a.\nu d.(\mathtt{out}(d,a)|(\mathtt{in}(d,y).\mathtt{out}(c,y))) \\ P_2 &= \nu a.\nu d.(\mathtt{out}(d,a)|(\mathtt{in}(d,y).\mathtt{out}(c,(y,f(y))))) \end{aligned}$$

Then we have $P_0 \approx_l P_1$ as $P_0 \xrightarrow{\nu a.\mathtt{out}(c,a)} 0$ and $P_1 \to \nu a.\nu d.\mathtt{out}(c,a) \xrightarrow{\nu a.\mathtt{out}(c,a)} 0$ and $P_1 \to \nu a.\nu d.\mathtt{out}(c,a) \approx_l P_0$. Yet neither $P_0 \approx P_2$ nor $P_1 \approx P_2$ as $P_2 \to \nu a.\nu d.\mathtt{out}(c,(a,f(a))) \equiv \nu a.\nu d.\nu z.\,\{^{a,f(a)}/_z\}\,\mathtt{out}(c,z) \xrightarrow{\nu z.\mathtt{out}(c,z)} \nu a.\nu d.\,\{^{(a,f(a))}/_z\}$ but neither P_0 nor P_1 can produce a frame that is statically equivalent. Note also that $P_0 \not\sim_l P_1$ and $P_0 \not\sim_l P_2$ as $P_0 \xrightarrow{\nu a.\mathtt{out}(c,a)} 0$ but P_1 and P_2 cannot do any external transition without a previous internal reduction. Similarly $P_1 \not\sim_l P_2$ as P_1 cannot produce a frame that is statically equivalent to $\nu a.\nu d.\,\{^{(a,f(a))}/_z\}$.

Note that restrictions can only forbid transitions, but not create new ones.

Lemma 1 *Let A be a closed extended process and $X \subseteq dom(A)$. Then $\nu X.A \xrightarrow{\mu} \nu X.A'$ implies $A \xrightarrow{\mu'} A'$ where μ can be a silent or a visible transition, and we have either*

— *$\mu' = \mu$ or*

— *for $x \in X$ and $\mu = \nu x.\mathtt{out}(\mathtt{a},\mathtt{x})$, $\mu' = \mathtt{out}(\mathtt{a},\mathtt{x})$*

Proof Any transition by $\nu X.A$ can be executed by A as the SCOPE-rule may only forbid certain transitions. Note that the STRUCT-rule allows to apply structural equivalence rules, yet none of these can enable transitions that could not be enabled without the restriction on X. The second case where μ and μ' differ is more of a syntactical corner case: a transition revealing one of the restricted values corresponds to an output of an unrestricted term in the unrestricted case, yet the underlying `out` is the same. □

2.4 Unique Parallel Decomposition of Processes

In a process algebra the question of unique process decomposition naturally arises: can we rewrite a process P as $P =^3 P_1|P_2|\ldots|P_n$, where each P_i is prime in the sense that it cannot be rewritten as the parallel composition of two non-zero processes?

Such a decomposition provides a maximally parallelized version of a given program P. Additionally, it is useful as it provides a normal form, and a cancellation result in the sense that $P|Q = P|R$ implies $Q = R$. This is convenient in proofs, for example when proving the equivalence of different security notions in electronic voting (see Chapter 3.5).

If there is an efficient procedure to transform a process into its normal form, such a decomposition can also be used to verify the equivalence of two processes [GM92]: once the processes are in normal form, one only has to verify if the factors on both sides are identical.

In the next section, we discuss related work. We then define the depth and norm of a process, and provide our first unique decomposition result with respect to strong bisimilarity. In the following section, we show the second result w.r.t. weak bisimilarity.

2.4.1 — Related Work

Unique decomposition (or factorization) has been a field of interest in process algebra for a long time. The first results for a subset of CCS were published by Moller and Milner [Mol89, MM93]. They showed that finite processes with interleaving can be uniquely decomposed with respect to strong bisimilarity. The same is true for finite processes with parallel composition, where – in contrast to interleaving – the parallel processes can synchronize. They also proved that finite processes with parallel composition can be uniquely decomposed w.r.t. weak bisimilarity.

Later on Christensen [Chr93] proved a unique decomposition result for normed processes (i.e. processes with a finite shortest complete trace) in BPP with interleaving or parallel composition w.r.t. strong bisimilarity.

Luttik and van Oostrom [LvO05] provided a generalization of the unique decomposition results for ordered monoids. They show that if the calculus satisfies certain properties, the unique decomposition result follows directly. Recently Luttik also extended this technique for weak bisimilarity [Lut12].

However, these existing results focus on "pure" calculi such as CCS or BPP or variants thereof. The Applied π-Calculus, as an "impure" variant of the π-Calculus

[3] Here = does not designate syntactical identity, but rather some behavioral equivalence or bisimilarity relation.

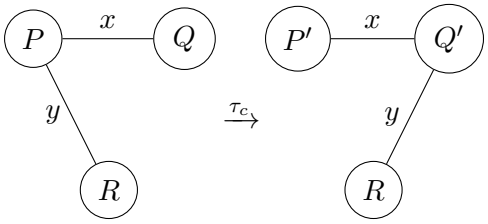

Figure 2.7 – Channel/Link Passing in the Applied π-Calculus

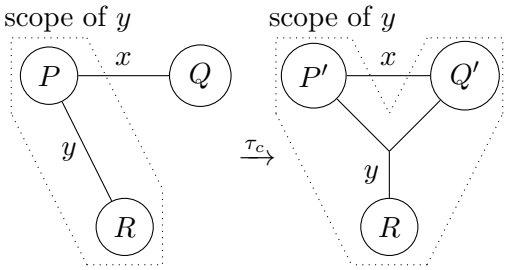

Figure 2.8 – Scope extrusion in the Applied π-Calculus

designed for the verification of cryptographic protocols, has a more complex structure and semantics. The main differences are the equational theory to model cryptographic primitives and the active substitutions. As active substitutions are minimal elements (with respect to the transition relation) different from 0, we cannot apply the general results by Luttik et al. [LvO05, Lut12].

Additionally, the Applied π-Calculus inherits the expressive power of the π-Calculus including *channel* or *link passing* (sometimes also called *mobility*) and *scope extrusion*. Consider three parallel processes P, Q and R, where P and Q synchronize using an internal reduction τ_c, i.e. $P|Q|R \xrightarrow{\tau_c} P'|Q'|R$ (see Figures 2.7 and 2.8). Channel passing allows a process P to send a channel y he shares with R to process Q (Figure 2.7). Scope extrusion arises for example when P sends a restricted channel y he shares with R to Q, since the scope after the transition includes Q' (Figure 2.8). This is of particular importance for unique decomposition since two parallel processes sharing a restricted channel might not be decomposable and hence a simple reduction might "fuse" two prime factors, which is not possible in BPP or CCS.

2.4.2 — Depth and Norm of Processes

In the following we prove unique decomposition for different subsets of processes, namely finite and normed processes. This requires to formally define the length of process traces. Let $\mathbf{Int} = \{\tau_c, \tau_t, \tau_e\}$ denote the set of labels corresponding to internal reductions or *silent* transitions, and $\mathbf{Act} = \{\mathtt{in}(a, M), \mathtt{out}(a, u), \nu u.\mathtt{out}(a, u)\}$ for any channel name a, term M and variable or name u, denote the set of labels of possible external or *visible* transitions. By construction we have $\mathbf{Act} \cap \mathbf{Int} = \emptyset$.

The *visible depth* is defined as the length of the longest trace of visible actions, i.e. labeled transitions, excluding internal reductions. Note that this may be infinite for processes including replication. We write $P \not\rightarrow$ if P cannot execute any transition, and $P \xrightarrow{\mu_1\mu_2\cdots\mu_n} P'$ for $P \xrightarrow{\mu_1} P_1 \xrightarrow{\mu_2} P_2 \xrightarrow{\mu_3} \ldots \xrightarrow{\mu_n} P'$.

Definition 6 (Visible Depth) *Let* $length_v : (\mathbf{Act} \cup \mathbf{Int})^* \mapsto \mathbb{N}$ *be a function where* $length_v(\epsilon) = 0$ *and* $length_v(\mu w) = \begin{cases} 1 + length_v(w) & \text{if } \mu \in \mathbf{Act} \\ length_v(w) & \text{otherwise} \end{cases}$
Then the visible depth $|P|_v \in (\mathbb{N} \cup \{\infty\})$ *of a closed process* P *is defined as follows:*

$$|P|_v = \sup\left\{ length_v(w) : P \xrightarrow{w} P', w \in (\mathbf{Act} \cup \mathbf{Int})^* \right\}$$

The *total depth* is defined as the length of the longest trace of actions (including internal reductions).

Definition 7 (Total Depth) *Let* $length_t : (\mathbf{Act} \cup \mathbf{Int})^* \mapsto \mathbb{N}$ *be a function where* $length_t(\epsilon) = 0$ *and* $length_t(\mu w) = 1 + length_t(w)$. *The* total depth $|P|_t \in (\mathbb{N} \cup \{\infty\})$ *of a closed process* P *is defined as follows:*

$$|P|_t = \sup\left\{ length_t(w) : P \xrightarrow{w} P', w \in (\mathbf{Act} \cup \mathbf{Int})^* \right\}$$

The *norm* of a process is defined as the length of the shortest complete trace, including internal reductions, where communications are counted as two. This is necessary to ensure that the norm of $P|Q$ is the sum of the norm of P and the norm of Q.

Definition 8 (Norm of a Process) *Let* $length_n : (\mathbf{Act} \cup \mathbf{Int})^* \mapsto \mathbb{N}$ *be a function where* $length_n(\epsilon) = 0$ *and* $length_n(\mu w) = \begin{cases} 1 + length_n(w) & \text{if } \mu \neq \tau_c \\ 2 + length_n(w) & \text{if } \mu = \tau_c \end{cases}$
The norm $\mathcal{N}(P) \in (\mathbb{N} \cup \{\infty\})$ *of a closed process* P *is defined as follows:*

$$\mathcal{N}(P) = \inf\left\{ length_n(w) : P \xrightarrow{w} P' \not\rightarrow, w \in (\mathbf{Act} \cup \mathbf{Int})^* \right\}$$

Example 6 *Consider the processes from our running example (Example 1 on page 18). We have* $|P_{ex}|_v = 2$, $|P_{ex}|_t = 3$ *and* $\mathcal{N}(P_{ex}) = 4$.

The above definitions admit some simple properties.

Lemma 2 *For any closed extended processes* P, Q *and* R *we have*

1. $|P|_v \leq |P|_t$
2. $P = Q|R$ *implies* $|P|_v = |Q|_v + |R|_v$
3. $P = Q|R$ *implies* $|P|_t = |Q|_t + |R|_t$
4. $P = Q|R$ *implies* $\mathcal{N}(P) = \mathcal{N}(Q) + \mathcal{N}(R)$

5. $P \approx_l Q$ *implies* $|P|_v = |Q|_v$
6. $P \sim_l Q$ *implies* $|P|_t = |Q|_t$
7. $P \sim_l Q$ *implies* $\mathcal{N}(P) = \mathcal{N}(Q)$

Proof Let P, Q and R be closed extended processes.

1. By definition of the length_t and length_v functions.
2. Suppose $P = Q|R$. Let w_P denote a maximal (with respect to length_v) complete (i.e. no further transitions are possible) sequence of transitions of P, i.e. $\text{length}_v(w_P) = |P|_v$. By definition of the function length_v we only count external transitions in w, which by rule PAR can originate either from Q or R, hence $|P|_v \leq |Q|_v + |R|_v$. Similarly, let w_Q and w_R denote maximal (with respect to length_v) complete sequences of transitions of Q and R respectively, i.e. $\text{length}_v(w_Q) = |Q|_v$ and $\text{length}_v(w_R) = |R|_v$. Then $w_P = w_Q w_R$ is a complete sequence of transitions of P, hence $|P|_v \geq |Q|_v + |R|_v$, thus $|P|_v = |Q|_v + |R|_v$.
3. Suppose $P = Q|R$. Let w_P denote a maximal (with respect to length_t) complete sequences of transitions of P, i.e. $\text{length}_t(w_P) = |P|_t$. As the length is maximal, there can be no synchronizations between Q and R, as otherwise we can build a longer trace by replacing this synchronization with two external reductions. Hence all transitions originate either from Q or R, hence $|P|_v \leq |Q|_t + |R|_t$. Similarly, let w_Q and w_R denote maximal (with respect to length_t) complete sequences of transitions of Q and R respectively, i.e. $\text{length}_t(w_Q) = |Q|_t$ and $\text{length}_t(w_R) = |R|_t$. Then $w_P = w_Q w_R$ is a complete sequence of transitions of P, hence $|P|_t \geq |Q|_t + |R|_t$, thus $|P|_t = |Q|_t + |R|_t$.
4. Suppose $P = Q|R$. Let w_Q and w_R denote (one of) the smallest (with respect to length_n) complete (i.e. no further transitions are possible) sequences of transitions of Q and R respectively. Then $w_P = w_Q w_R$ is a complete sequence of transitions of P, hence $\mathcal{N}(P) \leq \mathcal{N}(Q) + \mathcal{N}(R)$. Assume $\mathcal{N}(P) < \mathcal{N}(Q) + \mathcal{N}(R)$. Then there is a complete trace w'_P with $\text{length}_n(w'_P) < \text{length}_n(w_P)$. If w'_P contains no synchronizations of Q and R, each transition originates either from Q or R, hence giving shorter complete traces for Q and/or R, contradicting the minimality of w_Q and w_R. If w'_P contains a synchronization of Q and R, this can be rewritten into two external transitions of Q and R, resulting in a complete trace of the same length (by construction of length_n), leading to a contradiction. Hence $\mathcal{N}(P) = \mathcal{N}(Q) + \mathcal{N}(R)$.
5. Assume $P \approx_l Q$, but w.l.o.g. $|P|_v > |Q|_v$. Let w_s be a sequence of transitions of P with maximal number of visible transitions, i.e. $\text{length}_v(w_s) = |P|_v$. By the definition of $\approx_l$ each visible transition of P can be matched by a

visible transition by Q, giving a trace with more visible transitions than $|Q|_v$, leading to a contradiction. Hence $|P|_v = |Q|_v$.

6. Assume $P \sim_l Q$, but w.l.o.g. $|P|_t > |Q|_t$. Let w_s be a sequence of transitions of P with maximal number of transitions, i.e. $\text{length}_t(w_s) = |P|_t$. By the definition of $\sim_l$ each transition of P can be matched by a transition by Q, giving a trace with more transitions than $|Q|_t$, leading to a contradiction. Hence $|P|_t = |Q|_t$.
7. Assume $P \sim_l Q$ but w.l.o.g. $\mathcal{N}(P) < \mathcal{N}(Q)$. Let w_i be a sequence of transitions of P with minimal number of transitions, i.e. $\text{length}_n(w_i) = \mathcal{N}(P)$, and ending in a state $P' \not\rightarrow$. By the definition of $\sim_l$ each transition of P can be matched by a transition by Q, giving a trace with less transitions than $\mathcal{N}(Q)$ and ending in a state $Q' \not\rightarrow$ as $Q' \sim_l P'$ by definition, leading to a contradiction. Hence $\mathcal{N}(P) = \mathcal{N}(Q)$. □

Now we can define two important subclasses of processes: finite processes, i.e. processes with a finite longest complete trace, and normed processes, i.e. processes with a finite shortest complete trace.

Definition 9 (Finite and normed processes) *A closed process P is called* finite *if $|P|_t$ is finite (which implies $|P|_v$ is finite). A closed process P is called* normed *if $\mathcal{N}(P)$ is finite.*

It is easy to see that any finite process is normed, but not all normed processes are finite, as the following example illustrates.

Example 7 *Consider $P = \nu a.(out(a,m)|(in(a,x).(!in(b,y)))|in(a,x))$. Then we have $P \rightarrow P' \sim_l 0$, hence P is normed. However we also have $P \rightarrow P'' \sim_l !in(b,y)$, which has infinite traces. Hence P is not finite.*

It is also clear that not all processes are normed. Consider the following example.

Example 8 *Consider $P = !(\nu x.out(c,x))$. It is easy to see that for no sequence of transitions s we have $P \xrightarrow{s} P' \not\rightarrow$, i.e. P has no finite traces.*

Note however that any process without replication ("!") is finite, as no other syntactic element allows to construct infinite traces.

2.4.3 — Decomposition w.r.t. Strong Labeled Bisimilarity

We begin with the simpler case of Strong Labeled Bisimilarity. Note that $P \sim_l Q$ implies $|P|_t = |Q|_t$ and $\mathcal{N}(P) = \mathcal{N}(Q)$ for any closed processes P and Q.

We define strong parallel primeness as follows: a process is *prime* if it cannot be decomposed into non-trivial subprocesses (w.r.t. strong labeled bisimilarity).

We require the processes to be closed, which is necessary as our bisimulation relation is only defined on closed processes.

Definition 10 (Strongly Parallel Prime) *A closed process P is* strongly parallel prime *if*

— $P \not\sim_l 0$ *and*

— *for any two closed processes Q and R such that $P \sim_l Q|R$, we have $Q \sim_l 0$ or $R \sim_l 0$.*

Example 9 *Consider our running example:*

$$P_{ex} = \nu k.\nu l.\nu m.\nu d.\,(\{^{l}/_{y}\}\,|\mathtt{out}(c, enc(n,k))|\mathtt{out}(d,m)|\mathtt{in}(d,x).\mathtt{out}(c,x))$$

We can decompose P_{ex} as follows:

$$P_{ex} \sim_l (\nu l.\,\{^{l}/_{y}\})|(\nu k.\mathtt{out}(c, enc(n,k)))|(\nu d.(\nu m.\mathtt{out}(d,m)|\mathtt{in}(d,x).\mathtt{out}(c,x)))$$

The first factor $S_1 = \nu l.\,\{^{l}/_{y}\}$ is prime since it defines only one variable, and we cannot have two substitutions defining the same variable.

It is easy to see that the second factor $S_2 = \nu k.\mathtt{out}(c, enc(n,k))$ is prime, as it can only perform one external transition.

The third factor

$$S_3 = \nu d.(\nu m.\mathtt{out}(d,m)|\mathtt{in}(d,x).\mathtt{out}(c,x))$$

is prime because its two parts can synchronize using a shared restricted channel and then perform a visible external transition. Since $dom(S_3) = \emptyset$ any decomposition into two factors, i.e. such that $S_3 \sim_l S_3'|S_3''$, would mean that both factors can execute one transition each (otherwise they would be equivalent to 0 as they have an empty domain). However in that case both transitions of $S_3'|S_3''$ can be executed in any order, whereas in S_3 we have to start with the internal reduction. Hence no such decomposition exists.

Remark With respect to applications in protocol analysis, Example 9 illustrates that shared restricted names, for example private channels or shared keys, can prohibit decomposition. This is unavoidable, since a decomposition should not change the behavior of the processes (up to $\sim_l$). As our definition is solely based on the semantics and the bisimilarity notion, it allows to decompose a process as far as possible without changing the observed behavior, and thus any further decomposition will change the behavior. As a side-effect, the decomposition shows where shared restricted names (modeling for example keys) are actually used in a noticeable (w.r.t. to $\sim_l$) way, and where they can be ignored and processes can be further decomposed.

Remark Note also that within a prime factor we can recursively apply the decomposition as our bisimilarity notion is closed under the application of contexts. For example if we have a prime factor $P = \nu a.P'$, we can bring P' into normal form, i.e. $P' \sim_l P'_1|...|P'_n$, and rewrite $P = \nu a.P'$ as $P \sim_l \nu a.(P'_1|...|P'_n)$.

It is clear that not all processes can be written as a unique decomposition of parallel primes according to our definition.

Example 10 *Consider* $!P$ *for a process* $P \not\sim_l 0$*. By definition we have* $!P = P|!P$*, hence* $!P$ *is not prime. At the same time any such decomposition contains again* $!P$*, a non-prime factor, which needs to be decomposed again. Thus there is no decomposition into prime factors.*

However we can show that any closed normed process has a unique decomposition with respect to strong labeled bisimilarity. To achieve this, we need some preliminary lemmas about transitions and the domain of processes. The first lemma captures the fact that any process which cannot perform any transition and has an empty domain, is bisimilar to 0 (the empty process).

Lemma 3 *For any closed process* A *with* $dom(A) = \emptyset$ *and* $\mathcal{N}(A) = 0$*, we have* $A \sim_l 0$*.*

Proof Consider the relation $\mathcal{R} = \{(A, 0)\}$. We show that it fulfills the conditions of strong labeled bisimilarity:

1. We have $dom(A) = \emptyset = dom(0)$, hence $A \approx_s 0$.
2. Let $(A, 0) \in \mathcal{R}$. Obviously 0 cannot do any transition. Since $\mathcal{N}(A) = 0$, there exists a complete trace of length 0. Thus we have $A \not\rightarrow$, i.e. A cannot do any transition either and the remaining conditions are trivially satisfied.

As we have $(A, 0) \in \mathcal{R}$, this gives $A \sim_l 0$, which we wanted to show. □

We also need to show that if a normed process can execute a transition, it can also execute a norm-reducing transition.

Lemma 4 *Let* A *be a closed normed process with* $A \xrightarrow{\mu} A'$ *where* μ *is an internal reduction or visible transition. Then* $A \xrightarrow{\mu'} A''$ *with* $\mathcal{N}(A'') < \mathcal{N}(A)$*.*

Proof As A is normed, we have $\infty > \mathcal{N}(A)$. Moreover, $A \xrightarrow{\mu} A'$ implies $\mathcal{N}(A) > 0$, as this transition contradicts a complete trace of length 0. Hence the shortest complete trace w has $\infty > length_n(w) > 0$. Hence there is a transition μ' with $w = \mu' w'$ which reduces norm, i.e. $A \xrightarrow{\mu'} A''$ with $\mathcal{N}(A'') < \mathcal{N}(A)$. □

In a first step, we prove the existence of a decomposition.

Theorem 5 (Existence of Factorization) *Any closed normed process P can be expressed as the parallel product of strong parallel primes, i.e. $P \sim_l P_1|\ldots|P_n$ where for all $1 \leq i \leq n$, P_i is strongly parallel prime.*

Proof By induction on the norm of P, and on the size of the domain $dom(P)$.

— If $\mathcal{N}(P) = 0$:
  – If $|dom(P)| = 0$, then $P \sim_l 0$ (by Lemma 3), hence the factorization is the empty product.
  – If $|dom(P)| > 0$, then $P \not\sim_l 0$, hence P is either strongly parallel prime itself (in which case we are done), or can be written as $P \sim_l Q|R$ (by the definition of strongly parallel prime). As we have $dom(P) = dom(Q) \cup dom(R)$ with $dom(Q) \cap dom(R) = \emptyset$ and $|dom(Q)| > 0$, $|dom(R)| > 0$ (since $Q \not\sim_l 0$ and $R \not\sim_l 0$), we have $|dom(Q)| < |dom(P)|$, $|dom(R)| < |dom(P)|$, hence we can use the induction hypothesis to conclude.

— If $\mathcal{N}(P) > 0$:
  – If $|dom(P)| = 0$: P is either strongly parallel prime itself, or can be written as $P \sim_l Q|R$. Then we have $dom(P) = dom(Q) = dom(R) = \emptyset$, and $\mathcal{N}(Q) > 0$, $\mathcal{N}(R) > 0$ by Lemma 3, hence $\mathcal{N}(Q) < \mathcal{N}(P)$, $\mathcal{N}(R) < \mathcal{N}(P)$ by Lemma 2 and we can apply the induction hypothesis.
  – If $|dom(P)| > 0$, then $P \not\sim_l 0$, hence P is either strongly parallel prime itself, or can be written as $P \sim_l Q|R$. Suppose $\mathcal{N}(Q) > 0$ and $\mathcal{N}(R) > 0$, hence $\mathcal{N}(Q) < \mathcal{N}(P)$, $\mathcal{N}(R) < \mathcal{N}(P)$ and we can apply the induction hypothesis. Suppose w.l.o.g. $\mathcal{N}(Q) = 0 < \mathcal{N}(P)$, then $\mathcal{N}(R) = \mathcal{N}(P)$ by Lemma 2. Since $Q \not\sim_l 0$ this implies $|dom(Q)| > 0$ by Lemma 3, hence $|dom(R)| < |dom(P)|$, and we can use the induction hypothesis to conclude. □

We now show the uniqueness of the decomposition. As an additional notation, let $\exp(A, R)$ denote the exponent (i.e. the number of occurrences) of prime A in the unique factorization[4] of R.

Theorem 6 (Uniqueness of Factorization) *The strong parallel factorization of a closed normed process P is unique up to $\sim_l$ and permutation of the prime factors.*

Proof By induction on $\mathcal{N}(P)$, and on the size of the domain $dom(P)$.

— If $\mathcal{N}(P) = 0$:
  – If $|dom(P)| = 0$, then $P \sim_l 0$ (by Lemma 3), hence the factorization is the unique empty product.

[4]This notation only makes sense if we know that R has a unique decomposition, which however holds in the cases where we employ it during the proof and later on.

– If $|dom(P)| > 0$, $P \not\sim_l 0$. Suppose $P \sim_l Q$, but P and Q have different factorizations:

$$\begin{aligned} P &\sim_l A_1^{k_1}|A_2^{k_2}|\ldots|A_n^{k_n} \\ Q &\sim_l A_1^{l_1}|A_2^{l_2}|\ldots|A_n^{l_n} \end{aligned}$$

where the A_i's are distinct (i.e. for $i \neq j$ we have $A_i \not\sim_l A_j$) and $k_i \geq 0$, $l_i \geq 0$.

Note that since all factors A_i are prime we have $\forall i\ A_i \not\sim_l 0$, and since we also know $\mathcal{N}(P) = 0$ we have $\forall i\ \mathcal{N}(A_i) = 0$. By Lemma 3 we then have $dom(A_i) \neq \emptyset$, which implies $k_i, l_i \leq 1$ as we cannot have two substitutions defining the same variable.

Let m be such that $k_m \neq l_m$. Without loss of generality we assume $1 = k_m > l_m = 0$.

Obviously we have $dom(P) = dom(Q)$. Let $\tilde{v} = dom(P) \setminus dom(A_m)$. Then we have (by Lemma 2 and rules ALIAS and NEW-PAR):

$$\nu\tilde{v}.P \equiv A_m|\nu\tilde{v}.P' \sim_l A_m$$

where P' is P without the factor A_m, since $dom(\nu\tilde{v}.P') = \emptyset$ and $\mathcal{N}(\nu\tilde{v}.P') = 0$. Similarly

$$\nu\tilde{v}.Q \equiv |_{i\in I}\ \nu\tilde{v}_i.A_i|_{i\notin I}\ \nu\tilde{v}_i.A_i^{l_i} \sim_l |_{i\in I}\ \nu\tilde{v}_i.A_i$$

where $I = \{i | dom(A_i) \cap dom(A_m) \neq \emptyset \text{ and } l_i = 1\}$ and $\tilde{v}_i = dom(A_i) \cap \tilde{v}$. By $\nu\tilde{v}.P \sim_l \nu\tilde{v}.Q$ we have $A_m \sim_l |_{i\in I}\ \nu\tilde{v}_i.A_i$. If $|I| = 0$, we have $A_m \sim_l 0$ which contradicts the hypothesis that A_m is prime. Similarly for $|I| > 1$, we have a factorization for A_m into several processes, which also contradicts A_m prime.

For $|I| = 1$ we have the following cases: Let i denote the only index in I. If $\tilde{v}_i = \emptyset$, we have a contradiction to the distinctness hypothesis of the A_j's since $A_m \sim_l A_i$ with $m \neq i$ as $l_m = 0 \neq l_i = 1$.

If $\tilde{v}_i \neq \emptyset$ we have $dom(A_m) \subset dom(A_i)$, but $dom(A_m) \neq dom(A_i)$. Now consider $\tilde{v}' = dom(Q) \setminus dom(A_i)$. Then - as above - we have:

$$\nu\tilde{v}'.Q \equiv A_i|\nu\tilde{v}'.Q' \sim_l A_i$$

where Q' is Q without the factor A_i. Similarly

$$\nu\tilde{v}'.P \equiv |_{j\in I'}\ \nu\tilde{v}'_j.A_j|_{j\notin I'}\ \nu\tilde{v}'_j.A_j^{l_j} \sim_l |_{j\in I'}\ \nu\tilde{v}'_j.A_j$$

where $I' = \{j | dom(A_j) \cap dom(A_i) \neq \emptyset \text{ and } k_j = 1\}$ and $\tilde{v}'_j = dom(A_j) \cap \tilde{v}'$. Since $dom(A_m) \subset dom(A_i)$, $dom(A_m) \neq dom(A_i)$ and $dom(A_i) = dom(|_{j\in I'}\ \nu\tilde{v}'_j.A_j)$ we have $|I'| > 1$, hence $A_i \sim_l |_{j\in I'}\ \nu\tilde{v}'_j.A_j$ gives a

factorization of A_i, which contradicts the hypothesis that it is prime.

— If $\mathcal{N}(P) > 0$:

– If $|dom(P)| = 0$: Suppose $P \sim_l Q$, but P and Q have different factorizations:

$$\begin{aligned} P &\sim_l A_1^{k_1}|A_2^{k_2}|\ldots|A_n^{k_n} \\ Q &\sim_l A_1^{l_1}|A_2^{l_2}|\ldots|A_n^{l_n} \end{aligned}$$

where the A_i's are distinct (i.e. for $i \neq j$ we have $A_i \not\sim_l A_j$) and $k_i, l_i \geq 0$. By induction hypothesis we have that for every process R with $\mathcal{N}(R) < \mathcal{N}(P)$ the factorization is unique.

Let m be such that $k_m \neq l_m$, and that $\mathcal{N}(A_j) > \mathcal{N}(A_m)$ implies $k_j = l_j$ (i.e. A_m has the maximal norm among the factors in which P and Q differ). Without loss of generality we assume $k_m > l_m$.

In the following we use the fact that $P \sim_l Q$ and hence Q can simulate each transition of P and vice versa.

We now analyze different cases:

∗ If $P = A_m^{k_m}$, i.e. P is the power of a prime:

Note that Q cannot contain any prime factor A_r, $r \neq m$ with $\mathcal{N}(A_r) > \mathcal{N}(A_m)$: Suppose $l_r > 0$. By assumption, A_m is the maximal (w.r.t. norm) prime factor in which P and Q differ, hence $k_r = l_r > 0$. This contradicts $P = A_m^{k_m}$.

If $k_m = 1$ (i.e. P is prime), then Q is prime as well, and since $1 = k_m > l_m$ we have $Q \sim_l A_j$ for some $j \neq m$, which gives $A_m \sim_l A_j$, which contradicts the distinctness of the prime factors.

If $k_m > 1$:

Assume $l_m = 0$. Then – since $dom(A_m) = \emptyset$ – for some μ $A_m \xrightarrow{\mu} R$, $P \xrightarrow{\mu} P'$ with $\exp(A_m, P') = k_m - 1 > 0$ and $\mathcal{N}(P') < \mathcal{N}(P)$ by Lemma 4. Since $P \sim_l Q$, there exists a Q' with $Q \xrightarrow{\mu} Q'$. For any such Q' we have $\exp(A_m, Q') = 0$ since $l_m = 0$, A_m has maximal norm, and $l_i = 0$ for all A_i with $\mathcal{N}(A_i) > \mathcal{N}(A_m)$. Note that if $\mu = \tau_c$ two smaller factors in Q could fuse using scope extrusion (cf. Figure 2.8 on page 25), however no two prime factors share a secret channel (cf. the structure of P and Q). Hence the existence of such a transition τ_c fusing two factors would imply the existence of a $\mu \neq \tau_c$ that we can chose instead. Since $Q \sim_l P$ we also know that A_m can execute μ, which we can thus choose to ensure the above reasoning holds.

As P' and Q' have a unique prime factorization by induction hypothesis, we have a contradiction with $\exp(A_m, P') = k_m - 1 > 0 = \exp(A_m, Q')$. Hence assume $l_m > 0$: If $A_m \xrightarrow{\mu} R$ with $\mathcal{N}(R) < \mathcal{N}(A_m)$ for $\mu \neq \tau_c$, we have $Q \xrightarrow{\mu} Q'$ and since $P \sim_l Q$ there exists P' with $P \xrightarrow{\mu} P'$. We have $\exp(A_m, P') \geq k_m - 1 > l_m - 1 = \exp(A_m, Q')$ which contradicts

$P \sim_l Q$ using the induction hypothesis.
If no such transition μ exists, we have $A_m \xrightarrow{\tau_c} R$, hence $Q \xrightarrow{\tau_c} Q'$ and since $P \sim_l Q$ there exists P' with $P \xrightarrow{\tau_c} P'$. We know that P cannot simulate this transition using synchronization between the different copies of A_m as this would imply the existence of a visible norm-reducing transition μ (as the transition τ_c is norm-reducing as well). Hence we have again $\exp(A_m, P') \geq k_m - 1 > l_m - 1 = \exp(A_m, Q')$ which contradicts $P \sim_l Q$ using the induction hypothesis.

∗ If there exists $j \neq m$ such that $k_j > 0$:
Let μ, T be such that $P \xrightarrow{\mu} T$ and $\mathcal{N}(T) < \mathcal{N}(P)$ and for all ν such that $P \xrightarrow{\nu} P'$ with $\mathcal{N}(P') < \mathcal{N}(P)$ we have $\exp(A_m, P') \leq \exp(A_m, T)$. We now show that such μ, T exist.
Note because of Lemma 2 and $\mathcal{N}(P) < \infty$ we have $\mathcal{N}(A_i) < \infty$. This gives that if $A_i \xrightarrow{\mu} A_i'$ then $A_i \xrightarrow{\mu'} A_i''$ with $\mathcal{N}(A_i'') < \mathcal{N}(A_i)$ by Lemma 4.
Suppose no such μ, T exist. Hence for no A_i with $k_i > 0, i \neq m$ we have $A_i \xrightarrow{\mu} A_i'$, otherwise this allows a transition that would fulfill the above conditions. Hence (by Lemma 3) we have $dom(A_i) \neq \emptyset$ for any i with $k_i > 0, i \neq m$, which contradicts $|dom(P)| = 0$. Note that $\exp(A_m, T) \geq k_m$, as any transition by a factor different from A_m does not decrease the number of A_m's.
If we have a $\mu \neq \tau_c$, then - as $P \sim_l Q$ - there exists Q' with $Q \xrightarrow{\mu} Q'$ and $Q' \sim_l T$. Hence $\mathcal{N}(Q') < \mathcal{N}(Q)$ and $\exists A_t$ with $A_t \xrightarrow{\mu} R$.
If $\mathcal{N}(A_t) \leq \mathcal{N}(A_m)$ then $\exp(A_m, Q') \leq l_m < k_m \leq \exp(A_m, T)$, which gives the contradiction to the induction hypothesis,
If $\mathcal{N}(A_t) > \mathcal{N}(A_m)$ then $\exp(A_m, Q') = l_m + \exp(A_m, R)$, $t \neq m$ and $k_t = l_t > 0$ as A_m is maximal. Consider now $P \xrightarrow{\mu} P' = A_1^{k_1} | \ldots | A_t^{k_t - 1} | \ldots | A_n^{k_n} | R$ with $\exp(A_m, P') = k_m + \exp(A_m, R)$. Hence $\exp(A_m, Q') = l_m + \exp(A_m, R) < k_m + \exp(A_m, R) = \exp(A_m, P')$, which contradicts $Q' \sim_l P'$. Hence the only option for Q to match this transition would be to use a ever bigger A_s, in which case we can however apply the same argument ($k_s = l_s$). As the number of prime factors is finite, we have that $Q \not\sim_l P$ which gives the contradiction.
If no $\mu \neq \tau_c$ exists, choose a $\mu = \tau_c$. We distinguish two different cases: If the transition is matched by only one factor, we can argue as above. If the transition is matched by the synchronization of two factors ($A_r \xrightarrow{\alpha} A_r'$ and $A_s \xrightarrow{\bar{\alpha}} A_s'$), this implies that we have two visible actions on two different factors. As all transitions that do not reduce the number of A_m's are τ_c-transitions, these actions can only be matched by A_m, thus $A_m \xrightarrow{\alpha} A_m'$. Hence $Q \xrightarrow{\alpha} Q'$ with $\exp(A_m, Q') = l_m - 1$ and for any $P \xrightarrow{\alpha} P'$ we have $\exp(A_m, P') \geq k_m - 1 > l_m - 1 = \exp(A_m, Q')$,

which contradicts $P' \sim_l Q'$.

– If $|dom(P)| > 0$: This is essentially the same proof as above. In the first case (P is a power of a prime), we only have to consider the case $k_m = 1$ as $dom(A_m) \neq \emptyset$. Hence P is prime, and then Q is prime as well, and since $1 = k_m > l_m$ we have $Q \sim_l A_j$ for some $j \neq m$, which gives $A_m \sim_l A_j$, which contradicts the distinctness of the prime factors.
In the second case we have to be more careful when proving that μ and T with the desired properties exist. Once again, we suppose that they do not exist, hence for no A_i with $k_i > 0, i \neq m$ we have $A_i \xrightarrow{\mu} A_i'$, otherwise this allows a transition that would fulfill the conditions. Hence for any i with $k_i > 0$, $i \neq m$ we have $\mathcal{N}(A_i) = 0$, and by Lemma 3 we have $dom(A_i) \neq \emptyset$.
Let $\tilde{v} = dom(P) \setminus dom(A_m)$ and consider

$$\nu\tilde{v}.P \equiv A_m^{k_m} | \nu\tilde{v}.P' \sim_l A_m^{k_m}$$

where P' is P without the factor $A_m^{k_m}$. Similarly

$$\nu\tilde{v}.Q \equiv |_i \ \nu\tilde{v}_i.A_i^{l_i}$$

where $\tilde{v}_i = dom(A_i) \cap \tilde{v}$.
As $|dom(A_m^{k_m})| < |dom(P)|$ by induction hypothesis the factorization is unique. We cannot have $\tilde{v}_i = \emptyset$ for any $i \neq m$, as this contradicts the uniqueness of the factorization as $A_i \not\sim_l A_m$. Hence $dom(A_i) \neq \emptyset$ and $l_i = 1$.
As A_m is prime, we have that $A_m | R \sim_l \nu\tilde{v}_i.A_i$ for some $i \neq m$ and R. More precisely, we have $\nu\tilde{v}_i.A_i \sim_l A_m^l$ for some $l \geq 1$, as any other factorization of R would contradict the primeness of A_m. In fact, since A_m is the biggest factor in which P and Q differ and by Lemmas 1 and 3, we have $l = 1$.
We cannot have $Q \sim_l A_i$ as this would directly give a factorization of A_i. Hence there has to be another factor A_r which – by Lemma 3 – has either $dom(A_r) \neq \emptyset$ or can execute a transition (or both).
If $dom(A_r) \neq \emptyset$, consider $\tilde{v}' = dom(Q) \setminus dom(A_i)$. Then – as above – we have:

$$Q_1 = \nu\tilde{v}'.Q \equiv A_i | \nu\tilde{v}'.Q' \sim_l \nu\tilde{v}'.P = P_1$$

where Q' is Q without the factor A_i.
If $A_r \xrightarrow{\eta} A_r'$, we have

$$Q \sim_l A_i | A_r | S \xrightarrow{\eta} A_i | A_r' | S = Q_1$$

where S is Q without A_i and A_r. By $P \sim_l Q$ there exists P_1 with $P \xrightarrow{\eta} P_1 \sim_l A_i | A_r' | S$.

In both cases, we have a unique factorization by induction hypothesis. Additionally $\exp(A_i, Q_1) = 1$, and by the uniqueness of the factorization $\exp(A_i, P_1) = \exp(A_i, Q_1) = 1$. Let s be such that $dom(A_s) \cap dom(A_i) \neq \emptyset$, $k_s > 0$. Such s exists because of $dom(A_m) \subsetneq dom(A_i)$ and $dom(P) = dom(Q)$. Then by hypothesis A_s cannot do any transition, and $\exp(A_s, P_1) = \exp(A_s, P) = 1$, which contradicts $\exp(A_i, P_1) = 1$ because of the conflicting domains.

Hence μ and T with the desired properties exist, and the rest of the proof is the same as above. □

As a direct consequence, we have the following cancellation result.

Lemma 7 (Cancellation Lemma) *For any closed normed processes A, B and C, we have*

$$A|C \sim_l B|C \Rightarrow A \sim_l B$$

Proof As A, B and C are closed and normed, there exists a unique parallel factorization for each of them, i.e. $A \sim_l A_1|\ldots|A_k$, $B \sim_l B_1|\ldots|B_l$ and $C \sim_l C_1|\ldots|C_m$. Thus we have $A|C \sim_l A_1|\ldots|A_k|C_1|\ldots|C_m$ and $B|C \sim_l B_1|\ldots|B_l|C_1|\ldots|C_m$. These are prime factorizations, and by Theorem 6 they are unique. As $A|C \sim_l B|C$, they have to be identical. Hence $k + m = l + m$, thus $k = l$. We show that this implies that the factorizations of A and B have to be identical (up to $\sim_l$), which implies $A \sim_l B$. Consider the following cases:

If $k = 0$, $A \sim_l 0$. As $l = k = 0$, $B \sim_l 0$, and A and B have the same prime factorization.

If $k > 0$, we have $A \sim_l A_1|\ldots|A_k$. Suppose that there exists a prime factor A_i with $exp(A_i, A) \neq exp(A_i, B)$, then $exp(A_i, A|C) = exp(A_i, A) + exp(A_i, C) \neq exp(A_i, B) + exp(A_i, C) = exp(A_i, B|C)$, which contradicts the fact that $A|C$ and $B|C$ have the same prime factorization. □

2.4.4 — Decomposition w.r.t. Weak Labeled Bisimilarity

In this part, we discuss unique decomposition with respect to (weak) labeled bisimilarity. Note that $P \approx_l Q$ implies $|P|_v = |Q|_v$ for any closed processes P and Q (cf. Lemma 2 on page 26).

To obtain our unique decomposition result for weak labeled bisimilarity, we need to define parallel prime with respect to weak labeled bisimilarity.

Definition 11 (Weakly Parallel Prime) *A closed extended process P is* weakly parallel prime*, if*

— *$P \not\approx_l 0$ and*

— *for any two closed processes Q and R such that $P \approx_l Q|R$, we have $Q \approx_l 0$ or $R \approx_l 0$.*

This definition is analogous to strongly parallel prime. However, as the following example shows, in contrast to strong bisimilarity, not all normed processes have a unique decomposition w.r.t. to weak bisimilarity.

Example 11 *Consider* $P = \nu a.(out(a,m)|(in(a,x).(!in(b,y)))|in(a,x))$. *Then we have* $P \approx_l P|P$, *hence we have no unique decomposition. Note that this example does not contradict our previous result, as we have* $P \not\sim_l P|P$, *as* $P \to P' \sim_l 0$, *but* $P|P \to P'' \sim_l P$ *and* $P|P \not\to P'''$ *for any* $P''' \sim_l 0$. *Hence, w.r.t. strong labeled bisimilarity,* P *is prime.*

If however we consider normed processes that contain neither restriction ("ν") nor conditionals, we have that any normed process is finite (and hence has a unique decomposition, as we show below).

Lemma 8 *For any process* P *that does not contain restriction ("*ν*") or conditionals ("if then else"), we have that* P *is finite if and only if* P *is normed.*

Proof It is easy to see that any finite process is normed. To show the converse, we use induction on the structure of P.

— $P = 0$: P is obviously finite and normed.

— $P = \{^M/_x\}$: P is finite and normed.

— $P = Q|R$: If $\mathcal{N}(P) < \infty$ then $\mathcal{N}(Q) < \infty$ and $\mathcal{N}(R) < \infty$. By induction hypothesis $|Q|_t < \infty$ and $|R|_t < \infty$, hence $|P|_t < \infty$.

— $P = !Q$: If $\mathcal{N}(P) < \infty$ then $|Q|_t = 0$, hence $|P|_t < \infty$.

— $P = in(u,x).Q$ or $P = out(u,M).Q$: If $\mathcal{N}(P) < \infty$ then $\mathcal{N}(Q) < \infty$. By induction hypothesis $|Q|_t < \infty$, hence $|P|_t < \infty$. □

Similarly any process that does not contain replication is finite.

In the following we show that all finite processes have a unique decomposition w.r.t. to (weak) labeled bisimilarity. To prove this, we need some preliminary lemmas about transitions and the domain of processes.

Lemma 9 *For any closed process* A *with* $A \to^* A'$, *we have* $dom(A) = dom(A')$.

Proof The domain of a process is the set of variables for which it defines a substitution. No transition can destroy an existing active substitution. Similarly, if A executes only internal reductions, A cannot create any new active substitutions, hence $dom(A) = dom(A')$. □

Lemma 10 *For any closed process* A *for which no sequence of transitions* $A \to^* \xrightarrow{\alpha}$ A' *exists, we have* $A \approx_l A'$ *for any* A' *with* $A \to^* A'$.

Proof Consider the relation $\mathcal{R} = \{(X, Y) | A \to^* X \text{ and } A \to^* Y\}$. We will show that it fulfills the conditions of labeled bisimilarity:

1. Obviously we have $A \approx_s A$, which is closed under internal reductions (as they do not change the active substitutions). Hence for any $(C, D) \in \mathcal{R}$ we have $C \approx_s D$.
2. Let $(C, D) \in \mathcal{R}$. Hence $A \to^* C$ and $A \to^* D$. If $C \to C'$, we have $A \to^* C'$, hence $(C', D) \in \mathcal{R}$ (and symmetrically for $D \to D'$).
3. The last condition is trivially true. Suppose there exists $(C, D) \in \mathcal{R}$ such that $C \xrightarrow{\alpha} C'$, then we have $A \to^* \xrightarrow{\alpha} C'$, which contradict the hypothesis. The symmetrical case is analogous.

Obviously we have $(A, A') \in \mathcal{R}$ for any A' with $A \to^* A'$. □

The next lemma captures the fact that any process which cannot perform any visible transition and has an empty domain, is weakly bisimilar to 0 (the empty process).

Lemma 11 *If for a closed process A with $dom(A) = \emptyset$ there does not exist a sequence of transitions $A \to^* \xrightarrow{\alpha} A'$, then we have $A \approx_l 0$.*

Proof Suppose there is no sequence of transitions $A \to^* \xrightarrow{\alpha} A'$. We show that this implies $A \approx_l 0$. Consider the relation $\mathcal{R} = \{(A', 0) | A \to^* A'\}$. We show that it fulfills the conditions of labeled bisimilarity:

1. By hypothesis for any $(C, D) \in \mathcal{R}$ we have $\emptyset = dom(A) = dom(C)$ (as internal reductions do not change the active substitutions, Lemma 9) and $dom(D) = dom(0) = \emptyset$, hence $C \approx_s D$.
2. Let $(C, D) \in \mathcal{R}$. Hence $A \to^* C$ and $D = 0$. If $C \to C'$, we have $A \to^* C'$, hence $(C', 0) \in \mathcal{R}$ with $0 \to^* 0$. Note that symmetrically 0 cannot perform any transition, hence the condition is trivially true.
3. The last condition is trivially true. Suppose there exists $(C, D) \in \mathcal{R}$ such that $C \xrightarrow{\alpha} C'$, then we have $A \to^* \xrightarrow{\alpha} C'$, which contradicts the hypothesis. Symmetrically by definition 0 cannot perform any transitions at all.

As we have $(A, 0) \in \mathcal{R}$, this gives $A \approx_l 0$, which we wanted to show. □

As a direct consequence, this gives us that any non-zero process with empty domain can do a visible transition.

Corollary 12 *For every closed process A with $dom(A) = \emptyset$ and $A \not\approx_l 0$ there exists a sequence of transitions $A \to^* \xrightarrow{\alpha} A'$.*

Now we can show in a first step that a decomposition into prime factors exists.

Theorem 13 (Existence of Factorization) *Any closed finite active process P can be expressed as the parallel product of parallel primes, i.e. $P \approx_l P_1 | \ldots | P_n$ where for all $1 \leq i \leq n$ P_i is weakly parallel prime.*

Proof By induction on the visible depth of P, and on the size of the domain $dom(P)$.

— If $|P|_v = 0$:
  – If $|dom(P)| = 0$, then $P \approx_l 0$ (by Lemma 11), hence the factorization is the empty product.
  – If $|dom(P)| > 0$, then $P \not\approx_l 0$, hence P is either parallel prime itself (in which case we are done), or can be written as $P \approx_l Q|R$ with $Q \not\approx_l 0$ and $R \not\approx_l 0$ (by the definition of parallel prime). As we have $dom(P) = dom(Q) \cup dom(R)$ with $dom(Q) \cap dom(R) = \emptyset$ and $|dom(Q)| > 0$, $|dom(R)| > 0$ (by Lemma 11 since $Q \not\approx_l 0$ and $R \not\approx_l 0$), we have $|dom(Q)| < |dom(P)|$ and $|dom(R)| < |dom(P)|$, hence we can use the induction hypothesis to conclude.

— If $|P|_v > 0$:
  – If $|dom(P)| = 0$: P is either parallel prime itself, or can be written as $P \approx_l Q|R$. Then we have $dom(P) = dom(Q) = dom(R) = \emptyset$, and $|Q|_v > 0$, $|R|_v > 0$ (by Corollary 12), hence $|Q|_v < |P|_v$, $|R|_v < |P|_v$ and we can apply the induction hypothesis.
  – If $|dom(P)| > 0$: P is either parallel prime itself, or can be written as $P \approx_l Q|R$. Suppose $|Q|_v > 0$, $|R|_v > 0$, hence $|Q|_v < |P|_v$, $|R|_v < |P|_v$ and we can apply the induction hypothesis. Suppose w.l.o.g. $|Q|_v = 0 < |P|_v$, then $|R|_v = |P|_v$. Since $Q \not\approx_l 0$ by Lemma 11 this implies $|dom(Q)| > 0$, hence $|dom(R)| < |dom(P)|$, and we can use the induction hypothesis to conclude. □

To prove uniqueness we use the following relation on processes.

Definition 12 ("$\succeq$") *For two finite processes P and Q we have $P \succeq Q$ iff*
— $|P|_v > |Q|_v$ *or*
— $|P|_v = |Q|_v$ *and* $P \rightarrow^* Q$
i.e. P has either a longer visible trace than Q or P can be reduced to Q using internal reductions.

This is a partial order on finite processes modulo static equivalence. The relation is reflexive as we have $P \rightarrow^* P$, and transitive. It is also antisymmetric: Suppose $P \succeq Q$ and $Q \succeq P$, then $|P|_v = |Q|_v$, $P \rightarrow^* Q$ and $Q \rightarrow^* P$. Since P and Q are finite, we cannot have $P \rightarrow^* Q \rightarrow^* P$ for $P \not\equiv Q$ as this allows to construct an infinite trace.

Now we can show the uniqueness of the decomposition.

Theorem 14 (Uniqueness of Factorization) *The parallel factorization of a closed finite process P is unique (up to $\approx_l$).*

Proof We will prove a slightly different statement which implies the uniqueness of the factorization: Any closed finite processes P_f and Q_f with $P_f \approx_l Q_f$ have the same factorization (up to $\approx_l$).

Suppose $P_f \approx_l Q_f$, but P_f and Q_f have different factorizations:

$$\begin{aligned} P_f &= P_1|P_2|\ldots|P_{o_1} \\ Q_f &= Q_1|Q_2|\ldots|Q_{o_2} \end{aligned}$$

We can rewrite this factorization as follows:

$$\begin{aligned} P &= A_1^{k_1}|A_2^{k_2}|\ldots|A_n^{k_n} \\ Q &= A_1^{l_1}|A_2^{l_2}|\ldots|A_n^{l_n} \end{aligned}$$

where $P \approx_l P_f$ and $Q \approx_l Q_f$, the A_i's are distinct (i.e. for $i \neq j$ we have $A_i \not\approx_l A_j$) and $k_i, l_i \geq 0$.

We will show that this leads to a contradiction by induction on $a = |P|_t + |Q|_t$, and inside each case by induction on the size of the domain $b = |dom(P)| = |dom(Q)|$.

— If $a = 0$:

– If $b = 0$, then $P \approx_l 0$ (by Lemma 11), hence the factorization is the unique empty product.

– If $b > 0$, then $P \not\approx_l 0$.
Note that since $\forall i\ A_i \not\approx_l 0$ and $a = |P|_t + |Q|_t = 0$, we have $dom(A_i) \neq \emptyset$ by Lemma 11, which implies $k_i, l_i \leq 1$ as we cannot have two substitutions defining the same variable.
Let m be such that $k_m \neq l_m$. Without loss of generality we assume $1 = k_m > l_m = 0$.
Obviously we have $dom(P) = dom(Q)$. Let $\tilde{v} = dom(P) \setminus dom(A_m)$. Then we have (by Lemma 2 and rules ALIAS and NEW-PAR):

$$\nu\tilde{v}.P \equiv A_m|\nu\tilde{v}.P' \approx_l A_m$$

where P' is P without the factor A_m. Similarly

$$\nu\tilde{v}.Q \equiv |_{i\in I}\ \nu\tilde{v}_i.A_i|_{i\notin I}\ \nu\tilde{v}_i.A_i^{l_i} \approx_l |_{i\in I}\ \nu\tilde{v}_i.A_i$$

where $I = \{i | dom(A_i) \cap dom(A_m) \neq \emptyset \text{ and } l_i = 1\}$ and $\tilde{v}_i = dom(A_i) \cap \tilde{v}$. By $\nu\tilde{v}.P \approx_l \nu\tilde{v}.Q$ we have $A_m \approx_l |_{i\in I}\ \nu\tilde{v}_i.A_i$. If $|I| = 0$, we have $A_m \approx_l 0$ which contradicts the hypothesis that A_m is prime. Similarly for $|I| > 1$, we have a factorization for A_m into several processes, which also contradicts A_m prime.
For $|I| = 1$, i.e. $A_m \approx_l \nu\tilde{v}_i.A_i$ for the only index i in I, we have the

following cases: If $\tilde{v}_i = \emptyset$, we have a contradiction to the distinctness hypothesis of the A_j's since $A_m \approx_l A_i$ with $m \neq i$ as $l_m = 0 \neq l_i = 1$.
If $\tilde{v}_i \neq \emptyset$ we have $dom(A_m) \subsetneq dom(A_i)$. Now consider $\tilde{v}' = dom(Q) \setminus dom(A_i)$. Then - as above - we have:

$$\nu\tilde{v}'.Q \equiv A_i | \nu\tilde{v}'.Q' \approx_l A_i$$

where Q' is Q without the factor A_i. Similarly

$$\nu\tilde{v}'.P \equiv |_{j\in I'}\ \nu\tilde{v}'_j.A_j |_{j\notin I'}\ \nu\tilde{v}'_j.A_j^{l_j} \approx_l |_{j\in I'}\ \nu\tilde{v}'_j.A_j$$

where $I' = \{j | dom(A_j) \cap dom(A_i) \neq \emptyset \text{ and } k_j = 1\}$ and $\tilde{v}'_j = dom(A_j) \cap \tilde{v}'$. Since $dom(A_m) \subsetneq dom(A_i)$ and $dom(A_i) = dom(|_{j\in I'}\ \nu\tilde{v}'_j.A_j)$ (by $dom(P) = dom(Q)$) we have $|I'| > 1$, hence $A_i \approx_l |_{j\in I'}\ \nu\tilde{v}'_j.A_j$ gives a factorization of A_i, which contradicts the hypothesis that it is prime.

— If $a > 0$:

– If $b = 0$: If $P \approx_l 0$ then the (empty) factorization is unique. Hence suppose $0 \not\approx_l P \approx_l Q$.
Let m be such that A_m is a maximal (w.r.t. $\succeq$) A_i with $k_i \neq l_i$ (hence $k_m \neq l_m$). Without loss of generality we assume $k_m > l_m$.
In the following we will use the fact that $P \approx_l Q$ and hence Q can simulate each transition of P and vice versa. Assume $P \rightarrow^* \xrightarrow{\mu} P'$ such that $|P|_v = |P'|_v + 1$, then the labeled bisimilarity gives us $Q \rightarrow^* \xrightarrow{\mu} \rightarrow^* Q'$ with $P' \approx_l Q'$. For our proof it will be important that to simulate this transition in Q the prime factors cannot communicate. Suppose two prime factors $A_r \xrightarrow{\beta} R$ and $A_s \xrightarrow{\bar{\beta}} S$ communicated (through an internal reduction), then this has consumed at least two visible actions, hence $|Q'|_v \leq |Q|_v - 2 = |P|_v - 2 = |P'|_v - 1 < |P'|_v$. Thus P' and Q' do not have the same visible depth, which contradicts that fact that they are bisimilar.
We now analyze different cases:

∗ If $P \approx_l A_m^{k_m}$, i.e. P is the power of a prime:
Note that Q cannot contain any prime factor A_r, $r \neq m$ with $A_r \succeq A_m$:
Suppose $l_r > 0$. By assumption, A_m is a maximal (w.r.t. $\succeq$) prime factor in which P and Q differ, hence $k_r = l_r > 0$. This contradicts $P \approx_l A_m^{k_m}$.
If $k_m = 1$ (i.e. P is prime), then Q is prime as well, and since $1 = k_m > l_m$ we have $Q \approx_l A_j$ for some $j \neq m$, which gives $A_m \approx_l A_j$, which contradicts the distinctness of the prime factors.
If $k_m > 1$:
Note that this implies $dom(A_m) = \emptyset$ as otherwise we would have several

substitutions defining the same variables. Assume $l_m = 0$, then for some $\mu \in \mathbf{Act}$ $A_m \rightarrow^* \xrightarrow{\mu} R$ (by $A_m \not\approx_l 0$ and Corollary 12) with $|R|_v = |A_m|_v - 1$, so $P \rightarrow^* \xrightarrow{\mu} P'$ with $\exp(A_m, P') = k_m - 1 > 0$. Since $P \approx_l Q$, there exists a Q' with $Q \rightarrow^* \xrightarrow{\mu} \rightarrow^* Q'$. For any such Q' we have $\exp(A_m, Q') = 0$ since A_m is maximal (w.r.t. $\succeq$), $l_i = 0$ for all A_i with $|A_i|_v > |A_m|_v$ and since communication between different prime factors – which could through the exchange of secret channels lead to bigger (in the sense of visible depth) new prime factors – is not possible. As P' and Q' have a unique prime factorization by induction hypothesis, we have a contradiction with $\exp(A_m, P') = k_m - 1 > 0 = \exp(A_m, Q')$.
Hence assume $l_m > 0$:
Suppose $l_m < k_m - 1$: As $A_m \rightarrow^* \xrightarrow{\mu} R$, we have $P \rightarrow^* \xrightarrow{\mu} P'$ with $\exp(A_m, P') = k_m - 1$ and since $P \approx_l Q$ there exists Q' with $Q \rightarrow^* \xrightarrow{\mu} \rightarrow^* Q'$. Hence we have $\exp(A_m, P') = k_m - 1 > l_m \geq \exp(A_m, Q')$ which contradicts $P \approx_l Q$ using the induction hypothesis.
Hence assume $l_m = k_m - 1$:
We can write $Q = S|A_m^{l_m}$, where S is composed of prime factors. We have $S \not\approx_l A_m$ as the opposite contradicts either the distinctiveness of the prime factors or the fact that A_m is prime. Since $\emptyset = dom(A_m) = dom(P) = dom(Q) = dom(S)$ we have $S \approx_s A_m$, hence either S or A_m can do a transition the other cannot match. This transition can be a visible transition or an internal reduction.
Suppose $S \xrightarrow{\mu} S'$ with $|S'|_t < |S|_t$ such that A_m cannot match the transition. As we have $P \approx_l Q$, $S|A_m^{l_m} \xrightarrow{\mu} S'|A_m^{l_m} = Q'$ gives us that $P \rightarrow^* \xrightarrow{\mu} \rightarrow^* P'$ (w.l.o.g., when $\mu = \tau$ we have $P \rightarrow^* P'$). Since this transition reduced the total depth, we can apply the induction hypothesis and both Q' and P' have a unique prime factorization, hence $P' = R|A_m^{k_m - 1}$ where $A_m \rightarrow^* \xrightarrow{\mu} \rightarrow^* R$ (or $A_m \rightarrow^* R$ respectively). By the uniqueness of the factorization we also have $R \approx_l S'$, which contradicts the assumption that the transition cannot be simulated.
Suppose $A_m \xrightarrow{\mu} R$ with $|R|_t < |A_m|_t$ such that S cannot match the transition. As we have $P \approx_l Q$, $P = A_m^{k_m} \xrightarrow{\mu} R|A_m^{k_m - 1} = P'$ gives us that $Q \rightarrow^* \xrightarrow{\mu} \rightarrow^* Q'$ (w.l.o.g., otherwise $Q \rightarrow^* Q'$). Since this transition reduced the total depth, we can apply the induction hypothesis and both Q' and P' have the same unique prime factorization, hence $S'|A_m^{k_m - 1} = Q' \approx_l P' = R|A_m^{k_m - 1}$. Thus $R \approx_l S'$. Since A_m is the biggest factor in which P and Q differ, all other factors in S cannot be reduced to A_m, and we have $S \rightarrow^* \xrightarrow{\mu} \rightarrow^* S'$ (or $S \rightarrow^* S'$ respectively), which contradicts the assumption.

∗ If there exists $j \neq m$ such that $k_j > 0$:

Let $\mu \in \mathbf{Act}$, T be such that $P \rightarrow^* \xrightarrow{\mu} T$ and $|P|_v = |T|_v + 1$ and for all ν such that $P \rightarrow^* \xrightarrow{\nu} P'$ with $|P|_v = |P'|_v + 1$ we have $\exp(A_m, P') \leq \exp(A_m, T)$. We now show that such μ, T exist.

Suppose no such μ, T exist. Hence for no A_i with $k_i > 0, i \neq m$ we have $A_i \rightarrow^* \xrightarrow{\mu} A_i'$, otherwise this allows a transition that would fulfill the above conditions. Hence (by Lemma 11) we have $dom(A_i) \neq \emptyset$ for any i with $k_i > 0, i \neq m$, which contradicts $|dom(P)| = 0$. Note that $\exp(A_m, T) \geq k_m$, as any transition by a factor different from A_m does not decrease the number of A_m's.

As $P \approx_l Q$ there exists Q' with $Q \rightarrow^* \xrightarrow{\mu} \rightarrow^* Q'$ and $Q' \approx_l T$. Hence $|Q|_v = |Q'|_v + 1$ and $\exists A_t$ with $A_t \rightarrow^* \xrightarrow{\mu} \rightarrow^* R$ as there can be no communication between the A_i's (as shown above).

If $|A_t|_v \leq |A_m|_v$ then $\exp(A_m, Q') \leq l_m < k_m \leq \exp(A_m, T)$, which gives the contradiction to the induction hypothesis. Note that as A_m is the maximal prime factor in which P and Q differ, $A_j \rightarrow^* A_m$ implies $k_j = k_l$, hence Q' cannot contain additional A_m as a result of internal reductions - this would imply $\exp(A_j, Q') \neq \exp(A_j, P')$.

If $|A_t|_v > |A_m|_v$ then $t \neq m$, and $k_t = l_t > 0$ (as A_m is maximal). Consider $P \rightarrow^* \xrightarrow{\mu} \rightarrow^* P' = A_1^{k_1} | \ldots | A_t^{k_t - 1} | \ldots | A_n^{k_n} | R$ with $\exp(A_m, P') = k_m + \exp(A_m, R)$. Hence $\exp(A_m, Q') \leq l_m + \exp(A_m, R) < k_m + \exp(A_m, R) = \exp(A_m, P')$, which contradicts $Q' \approx_l P'$. Hence the only option for Q to match this transition would be to use a ever bigger A_s, in which case we can however apply the same argument ($k_s = l_s$). As the number of prime factors is finite, we have that $Q \not\approx_l P$ which gives the contradiction. Note that – as above – Q' cannot contain additional A_m as a result of internal reductions.

– If $b > 0$: This is essentially the same proof as above. In the first case (P is a power of a prime), we only have to consider the case $k_m = 1$ as $dom(A_m) \neq \emptyset$. Hence P is prime, and then Q is prime as well, and since $1 = k_m > l_m$ we have $Q \approx_l A_j$ for some $j \neq m$, which gives $A_m \approx_l A_j$, which contradicts the distinctness of the prime factors.

In the second case we have to be more careful when proving that μ and T with the desired properties exist. Once again, we will suppose that they do not exist, hence for no A_i with $k_i > 0, i \neq m$ we have $A_i \rightarrow^* \xrightarrow{\mu} A_i'$, otherwise this allows a transition that would fulfill the conditions. Hence (by Lemma 11) we have $dom(A_i) \neq \emptyset$ for any i with $k_i > 0, i \neq m$. Let $\tilde{v} = dom(P) \setminus dom(A_m)$ and consider

$$\nu\tilde{v}.P \equiv A_m^{k_m} | \nu\tilde{v}.P' \approx_l A_m^{k_m}$$

where P' is P without the factor $A_m^{k_m}$. Similarly

$$\nu\tilde{v}.Q \equiv |_i \ \nu\tilde{v}_i.A_i^{l_i}$$

where $\tilde{v}_i = dom(A_i) \cap \tilde{v}$.
As $|dom(A_m^{k_m})| < |dom(P)|$ by induction hypothesis the factorization is unique. We cannot have $\tilde{v}_i = \emptyset$ for any $i \neq m$, as this contradicts the uniqueness of the factorization as $A_i \not\approx_l A_m$. Hence $dom(A_i) \neq \emptyset$ and $l_i = 1$ for $i \neq m$.
As A_m is prime, we have that $A_m|R \approx_l \nu\tilde{v}_i.A_i$ for some $i \neq m$ and R. More precisely, we have $\nu\tilde{v}_i.A_i \approx_l A_m^l$ for some $l \geq 1$, as any other factorization of R would contradict the primeness of A_m. In fact, since A_m is the biggest factor in which P and Q differ and by Lemmas 11 and 1, we have $l = 1$.
We cannot have $Q \approx_l A_i$ as this would directly give a factorization of A_i. Hence there has to be another factor A_r which – by Lemma 11 – has either $dom(A_r) \neq \emptyset$ or can execute a visible transition (or both).
If $dom(A_r) \neq \emptyset$, consider $\tilde{v}' = dom(Q) \setminus dom(A_i)$. Then – as above – we have:

$$\nu\tilde{v}'.Q \equiv A_i|\nu\tilde{v}'.Q' = Q_1 \approx_l \nu\tilde{v}'.P = P_1$$

where Q' is Q without the factor A_i.
If $A_r \rightarrow^* \xrightarrow{\eta} A_r'$, we have

$$Q \approx_l A_i|A_r|S \rightarrow^* \xrightarrow{\eta} A_i|A_r'|S = Q_1$$

where S is Q without A_i and A_r. By $P \approx_l Q$ there exists P_1 with $P \rightarrow^* \xrightarrow{\eta} \rightarrow^* P_1 \approx_l A_i|A_r'|S$.
In both cases, we have a unique factorization by induction hypothesis. Additionally $\exp(A_i, Q_1) = 1$, and by the uniqueness of the factorization $\exp(A_i, P_1) = \exp(A_i, Q_1) = 1$. Let s be such that $dom(A_s) \cap dom(A_i) \neq \emptyset$, $k_s > 0$. Such s exists because of $dom(A_m) \subsetneq dom(A_i)$ and $dom(P) = dom(Q)$. Then by hypothesis A_s cannot do any visible transition, and by Lemma 10 $\exp(A_s, P_1) = \exp(A_s, P) = 1$, which contradicts $\exp(A_i, P_1) = 1$ because of the conflicting domains.
Hence μ and T with the desired properties exist, and the rest of the proof is the same as above. □

Again we have a cancellation result using the same proof as above.

Lemma 15 (Cancellation Lemma) *For any closed finite processes A, B and C, we have*

$$A|C \approx_l B|C \Rightarrow A \approx_l B$$

Proof Similar to the proof of Lemma 7. □

Type of Process	Strong Bisimilarity ($\sim_l$)	Weak Bisimilarity ($\approx_l$)
finite	Theorem 5	Theorem 13
normed	Theorem 5	(Counter-)Example 7
general	(Counter-)Example 10	(Counter-)Example 10

Table 2.1 – Summary of unique factorization results for the Applied π-Calculus

2.5 Conclusion

In this chapter we recalled the Applied π-Calculus, in particular its syntax, semantics and equivalence notions. We then presented two unique decomposition results for subsets of the Applied π-Calculus. We showed that any closed finite process can be decomposed uniquely with respect to weak labeled bisimilarity, and that any normed process can be decomposed uniquely with respect to strong labeled bisimilarity. Table 2.1 sums up our results.

As the concept of parallel prime decomposition has its inherent limitations with respect to replication ("!", see Example 10), a natural question is to find an extension to provide a normal form even in cases with infinite behavior. A first result in this direction has been obtained by Hirschkoff and Pous [HP10] for a subset of CCS with top-level replication. They define the *seed* of a process P as the process Q, Q bisimilar to P, of least size (in terms of prefixes) whose number of replicated components is maximal (among the processes of least size), and show that this representation is unique. They also provide a similar normal form result for the Restriction-Free-π-Calculus (i.e. no "ν"). It remains however open if a similar result can be obtained for the full calculus.

Another interesting question is to find an efficient algorithm that converts a process into its unique decomposition. It is unclear if such an algorithm exists and can be efficient, as simply deciding if a process is finite can be non-trivial. Such an algorithm could however allow to verify the bisimilarity of two given processes by transforming them into their normal form, and then simply checking if the normal forms are identical.

Finally we did not show formally that Strong Labeled Bisimilarity is closed under the application of contexts. Due to its close relation to Weak Labeled Bisimilarity we expect the result to hold (which is important for our decomposition to be meaningful), yet we estimate that the full formal proof will be at least similarly as complex as the proof for Weak Labeled Bisimilarity [Liu11, Liu13].

Chapter 3

eVoting

PRIVACY is a major concern in electronic voting, as it is crucial to ensure the fairness of the voting process by protecting voters from coercion.

In this chapter we propose a taxonomy of privacy for voting protocols considering different attacker capabilities (insider or outsider), different attacks (including forced abstention) and different levels of coercion (simple Privacy, Receipt-Freeness and Coercion-Resistance). As case studies we analyze several protocols to assess which levels of privacy they achieve.

The taxonomy is based on definitions where two voters swap votes, which is unsuitable for situations where votes are weighted. To address this, we propose generalized definitions and establish a precise formal link between the two approaches. We also show that in the generalized model multi-voter coercion can be (under certain conditions) reduced to single-voter coercion. This holds for Receipt-Freeness and Coercion-Resistance. Additionally we show that under the same conditions a situation with multiple corrupted voters can be reduced to a situation without corrupted voters. We also provide a case study of a protocol supporting weighted votes.

Contents

3.1 Introduction 49
3.1.1 Contributions 50
3.1.2 Outline of the Chapter 51
3.2 Related Work 51
3.3 A Formal Taxonomy of Privacy in Voting 54
3.3.1 Formalizing Voting Protocols 54
3.3.2 Defining Privacy: A Modular Approach 60
3.3.3 Definitions in the Applied π-Calculus 62
3.3.4 Hierarchy 65
3.3.5 Case Studies 68

3.3.5.1 Protocol by Fujioka, Okamoto and Ohta (FOO) . 68
3.3.5.2 Protocol by Okamoto 77
3.3.5.3 Bingo Voting 88
3.3.5.4 Protocol by Lee, Boyd, Dawson, Kim, Yang and Yoo 103
3.3.5.5 Summary 114
3.4 Defining Privacy for Weighted Votes 115
3.4.1 Formal Definition 116
3.4.2 Example: A Variant of FOO 117
3.4.2.1 Adding Vote Weights 117
3.4.2.2 Model and Analysis 117
3.4.3 Link to Existing Definitions 120
3.4.4 Including Corrupted Voters 126
3.5 Multi-Voter Coercion 133
3.5.1 Single-Voter Receipt-Freeness (SRF) 133
3.5.2 Multi-Voter Receipt-Freeness (MRF) 139
3.5.3 Single-Voter Coercion (SCR) 144
3.5.4 Multi-Voter Coercion (MCR) 146
3.6 Conclusion 149

3.1 Introduction

Electronic voting systems have been designed and employed in practice for several years, mainly in the form of direct-recording electronic (DRE) voting machines. More recently, some countries started to offer the possibility to vote over the Internet, e.g. in Estonia [Est], parts of Switzerland [Gen13, Reg13] or for French expatriates [Min13].

As voting is a central act of participation in modern democracies, the use of such systems in general elections requires high security standards and remains controversial [Par07, UK 07, Min08, Bun09]. Consequently many security requirements for electronic voting systems have been proposed:

— *Correctness*: The announced result corresponds to the sum of all submitted votes.

— *Eligibility*: Only the registered voters can vote, and nobody can submit more votes than allowed (typically only one).

— *Fairness*: The election process is fair, in particular no preliminary results that could influence other voters' decisions are available.

— *Robustness*: The protocol can tolerate some misbehaving voters.

— *Verifiability*: Voters are provided with evidence that allows them to verify the correctness of the election process.

— *Privacy*: All votes remain private.

The last requirement is crucial to ensure that the voters are free in their choice and cannot be coerced. In the literature, it is often split into several notions (e.g. [DKR09, LSB$^+$09, SC11, SB13, Jon09, DHvdG$^+$13]):

— *Vote-Privacy*: The votes are kept private with respect to an outside observer. This can also be expressed as an unlinkability between the voter and his vote.

— *Receipt-Freeness*: A voter cannot construct a receipt which allows him to prove to a third party that he voted for a certain candidate. This is to prevent vote-buying.

— *Coercion-Resistance*: Even when a voter interacts with a coercer during the entire voting process, the coercer cannot be sure whether he followed his instructions or actually voted for another candidate.

— *Security against Forced Abstention Attacks*: A coercer cannot force a voter to abstain from voting.

— *Vote-Independence*: No voter can relate his vote to any other voter's vote[1].

[1]Being able to copy votes can compromise privacy if the number of participants is small or a noticeable fraction of voters can be corrupted. Consider a case with three voters: the third voter can copy the first voter's vote and submit it as his vote. This results in (at least) two votes for

— *Everlasting Privacy*: Privacy is not only ensured at the time of the election, but also in the long term when computationally secure encryptions might turn insecure due to increased computational power.

As we argued in Chapter 1, the design of complex protocols to fulfill all these partly antipodal requirements [CMFP+06] is notoriously difficult and error-prone. Consequently many efforts have been undertaken to formally define and verify these properties (e.g. [JCJ05, MN06b, BHM08, DKR09, Jon09, KRS10, SC11]).

3.1.1 — Contributions

Generalizing the existing work, we provide the following contributions in this chapter:

1. We propose a taxonomy of privacy notions which allows to assess the level of privacy provided by a voting protocol against various attacks. The taxonomy is based on formal definitions of the classical notions (Vote-Privacy, Receipt-Freeness and Coercion-Resistance) in the Applied π-Calculus originally proposed by Delaune et al. [DKR09], but extends their work in several ways to include insider attacks as well as specialized attacks such as forced-abstention or vote-copying attacks.
2. We illustrate our model using several case studies: the protocol by Fujioka et al. [FOO92], the protocol by Okamoto [Oka96], the protocol by Lee et al. [LBD+03] and Bingo Voting [BMQR07].
3. Most existing symbolic definitions of Privacy are based on the idea of swapping votes. If the votes are private, a case where Alice votes "yes" and Bob votes "no" should be indistinguishable from a case where Alice votes "no" and Bob votes "yes". Yet this definition is unsuitable for some situations, for example in companies where votes are weighted according to the proportion of shares held by each shareholder. Consider the following example: Alice owns 50% of the stocks, and Bob and Carol each hold 25%. The cases where Alice and Bob swap votes are now easily distinguishable if Carol votes "yes" all the time, as the result of the vote is different: 75% vs. 50% vote for "yes". Note that there are still situations where privacy is ensured in the sense that different situations give the same result. The last outcome (50% yes, 50% no) could - for example - also be announced if Alice votes "yes" and Bob and Carol vote "no".

   Protocols supporting vote weights have been proposed, for example Eliasson and Zúquete [EZ06] developed a voting system supporting vote weights based on REVS [JZF03], which itself is inspired by the protocol by Fujioka et al. [FOO92]. We provide a generalized definition of privacy to accommodate

the candidate chosen by the first voter and his choice can thus be inferred from the result.

weighted votes, and show under which conditions the generalized notions coincide with the notions from the first taxonomy. We also provide a case study to illustrate our approach.

4. In the previous definitions of Receipt-Freeness and Coercion-Resistance, we only consider one voter under coercion. We provide a generalized definition for multiple coerced voters, and show that single-voter and multi-voter coercion coincide under certain assumptions using a unique decomposition result from Chapter 2.

3.1.2 — Outline of the Chapter

In the following section we review related work on formal verification of privacy in electronic voting. In Section 3.3, we present our taxonomy of privacy in voting, and discuss several case studies. To deal with weighted votes, we provide a generalized model in Section 3.4, discuss a case study, and establish a link to our taxonomy. Finally, in Section 3.5, we show that the case of multiple coerced voters can be reduced to a single coerced voter under some reasonable assumptions, and again establish a link to our taxonomy. In Section 3.6 we conclude this chapter.

3.2 Related Work

As the amount of work on electronic voting is huge, we concentrate here on related work concerning the specification and verification of privacy. We start by discussing application-independent privacy notions, followed by work on the specification and certification – including privacy – of voting protocols and systems. Then we discuss work on voting protocols in the computational model, and finally in the symbolic model.

Canetti and Gennaro [CG96] proposed computational definitions of receipt-freeness that can be applied to many different applications. Similarly Unruh and Müller-Quade defined coercion-resistance [UMQ10] in the (computational) Universal Composability (UC) framework [Can01]. Application-independent anonymity notions were also proposed by Bohli and Pashalidis [BP09]. Although these definitions are very general, the application to voting protocols often results in – for this context – unusual privacy notions (Pseudonymity etc.), compared to the classic properties such as vote-privacy that we discuss here.

Volkamer et al.[VM07] worked towards developing requirements and evaluation procedures for the certification of eVoting systems in the common criteria framework. They give a formalization of the different requirements in a high-level way, similar to a technical standard, i.e. in “formalized” natural language. Yet this specification still lacks precise semantics.

Kramer and Ryan [KR11] propose a modular specification of voting systems which includes (among many other properties) privacy and receipt-freeness. It is expressed in a multi-modal logic featuring time. Such a logic framework is very expressive and hence suitable for the specification of systems, but difficult to handle for automatic verification tools. Concerning privacy, they define coercion-resistance as the conjunction of privacy and receipt-freeness, differing from the original definition by Juels et al. [JCJ05] which includes resistance against force abstention attacks, and allows a coercer to instruct the voter to deviate from the protocol. We employ the Applied π-Calculus as the basis for our definitions as it has powerful tool support e.g. using ProVerif.

In the computational model, Juels et al. [JCJ05] were the first to give a formal definition of coercion-resistance. They showed that their protocol (which became the basis for Civitas [CCM08]) is secure with respect to their definition. However – as their protocol is based on voting "credentials" – credentials also appear in the definition. Their model is thus unsuitable for protocols that do not use credentials (e.g. Bingo Voting [BMQR07] or the protocol by Lee et al. [LBD$^+$03]), which we can accommodate in our model. A first computational definition of privacy for voting was given by Cohen/Benaloh and Fischer [CF85], a first computational definition of receipt-freeness for voting by Benaloh and Tuinstra [BT94] (although their protocol suffered from an issue with the underlying encryption scheme [FLA11]). A first computational definition within the UC-framework was given by Moran and Naor [MN06b].

De Marneffe, Pereira, and Quisquater [dMPQ07a, dMPQ07b] proposed a computational security definition based on the simulation of an ideal functionality of a fair and secure voting system by the real-world system. Although not impossible [DKP09], such an approach is rarely employed in symbolic models.

Based on similar definitions [BCP$^+$11, CPP13], Smyth and Bernhard [SB13] recently proposed computational definitions of ballot secrecy (similar to vote-privacy) and ballot independence, and showed that they coincide under some reasonable assumptions. In our symbolic model, we have a similar result (see Section 3.4.4).

A different computational game-based definition including coercion was proposed by Küsters, Truderung, and Vogt [KTV10b]. They applied it to Scantegrity II [KTV10c] as well as ThreeBallot and VAV [KTV11]. In their definition they consider the overall advantage of an attacker trying to guess a voter's vote, which is always non-negligible as in certain situations the votes are revealed, e.g. in the case of an unanimous vote. We employ a different approach by comparing different situations that lead to the same outcome: In such a case, the attacker should not have a (non-negligible) advantage.

Along similar lines to Küsters et al., Bernhard et al. [BCPW12] proposed to

measure vote-privacy using (computational) entropy. The idea is to compute how much privacy remains after an election, i.e. given a certain protocol, vote distribution and counting function. This can be computed after a concrete election (i.e. using the actually the submitted votes), or given a certain vote distribution to compute a metric giving an expectation of the level of privacy provided. They also establish a link between their information-theoretic approach and a previous computational definition [BCP+11].

In the symbolic model, Jonker [Jon09] defines privacy by analyzing a voter's choice group, i.e. the set of candidates the voter could have voted for given the attackers knowledge about the protocol execution. For maximal privacy the choice group should be as large as possible. If it contains only one option, the attacker knows the exact vote and all privacy is lost. This is somewhat similar to the work by Küsters et al. in the computational model, but Jonker does not consider probabilities: even if a choice is very unlikely, it is still considered in the choice group. In our work, we employ a different approach: we explicitly identify situations that should be inside the same choice group given only the result, and then require these situations to be observationally equivalent.

Langer et al. [LJP10] developed verifiability definitions and privacy notions based on (un-)linkability between a voter and his vote. They define (un-)linkability as a property on traces, whereas we consider bisimulations.

Küsters and Truderung [KT09a] proposed a high-level model independent definition of coercion-resistance for voting protocols. Their definition has to be instantiated with a concrete formal model. The exact security level can be defined with respect to certain chosen goals, and excluding explicit special cases. In contrast to our model, their definition is based on traces and not bisimulations, and they only define coercion-resistance (in particular no simple vote-privacy). They also explicitly consider multi-voter coercion. In their abstract model, Single-Voter Coercion and Multi-Voter Coercion turned out to be different in general. Subsequently they proposed a modified definition of Coercion-Resistance that implies both Single- and Multi-Voter Coercion-Resistance, but they did not provide a precise analysis of the link between these notions. In our model we can show that Single- and Multi-Voter Coercion are equivalent under certain conditions on the protocol. Hence we do not need to change the initial definition, and the conditions allow us to precisely characterize the difference between both notions.

Langer et al. also worked towards a high-level taxonomy of privacy [LSB+09, LSBV10] based on different levels of secrecy with respect to different attacker capabilities. This has some similarities to our taxonomy approach, however their definitions are abstract and have to be instantiated with a concrete formal process and attacker model, whereas we directly give the operational definitions in the Applied π-Calculus.

Backes et al. [BHM08] translated the definition by Juels et al. [JCJ05] to the Applied π-Calculus and provided an automated verification using ProVerif. Their approach has the same limitations as the original, computational definitions: they are difficult to apply to protocols using different techniques.

More general definitions were developed by Delaune, Kremer and Ryan [DKR09] (the so-called DKR-model). They express different levels of privacy as observational equivalence in the Applied π-Calculus. An attacker should not be able to distinguish one case in which the voter complies with the coercer's instructions and another in which he only pretends to do so and votes as he wishes. Our definitions are based on their model, but we extend it in several ways: we include forced abstention, corrupted voters and generalize it to accommodate protocols with weighted votes.

Smyth and Cortier [SC11] showed that being able to copy votes can compromise privacy if the number of participants is small or a noticeable fraction of voters can be corrupted. For example in the case of three voters, the third voter can try to copy the first voter's vote and submit it as his vote. If this succeeds, it will result in (at least) two votes for the candidate chosen by the first voter and his choice can thus be inferred from the result. They also formally analyzed ballot secrecy in Helios using an adaption of the model by Delaune, Kremer and Ryan. However we show that, in general, the DKR model is not sufficient to capture vote-independence. For example the protocol by Lee et al. [LBD$^+$03] was shown to be coercion-resistant in this model, despite its vulnerability to vote-copy attacks (see Section 3.3.5.4).

Recently Arapinis et al. [ACKR13] proposed a formal model in the Applied π-Calculus to verify everlasting privacy.

3.3 A Formal Taxonomy of Privacy in Voting

In this section we present our formal taxonomy of privacy in electronic voting. We start by giving a formalization of voting protocols, then explain our definitions informally and give the detailed definitions in the Applied π-Calculus. We present the hierarchy of notions, and discuss several case studies: the protocols by Fujioka, Okamoto and Ohta (FOO) [FOO92], by Okamoto [Oka96], by Lee et al. [LBD$^+$03] and Bingo Voting [BMQR07].

3.3.1 — Formalizing Voting Protocols

In this section we describe our model of voting protocols in the Applied π-Calculus. It is inspired by the model used by Delaune, Kremer and Ryan [DKR09].

First of all, we define the notion of a *voting protocol*. Informally, a voting protocol specifies the processes executed by voters and authorities.

Definition 13 (Voting Protocol) *A* voting protocol *is a tuple* $(V, A_1, \ldots, A_m, \tilde{c})$ *where* V *is the process that is executed by the voter, the* A_j*'s are the processes executed by the election authorities, and* $\tilde{c}$ *is a set of private channels.*

Note that the protocol only defines one process V, which is instantiated for each voter. Yet here may be several authorities, for example a registrar, a bulletin board, a mixer, a tallier, ...

In our definitions we reason about privacy using concrete instances of a voting protocol. An instance is called a *Voting Process.*

Definition 14 (Voting Process) *A* voting process *of a voting protocol* $(V, A_1, \ldots, A_m, \tilde{c})$ *is a closed plain process*

$$\nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1}|\ldots|V\sigma_{id_n}\sigma_{v_n}|A_1|\ldots|A_l)$$

where $l \leq m$*,* $\tilde{n}$ *includes (some of) the secret channel names,* $V\sigma_{id_i}\sigma_{v_i}$ *are the processes executed by the voters where:*

— σ_{id_i} *is a substitution assigning the identity to a process (this determines for example the secret keys),*

— σ_{v_i} *specifies the vote(s) and if the voter abstains,*

and $A_j s$ *are the election authorities which are required to be honest.*

As an extension to the model by Delaune et al. [DKR09], the substitution determining the vote of the voter can also specify abstention or other (correct) behaviors. In this case V includes all the possible behaviors, and σ_{v_i} determines which of them is executed. In our model an abstaining voter does not participate at all in the election, we define this formally below.

Note that each voter runs the same process V, which is instantiated with a different σ_{id_i} (his identity) and σ_{v_i} (his vote(s)). Note also that we have $l \leq m$ as not all authorities might be trusted. If an authority is not trusted, it is not modeled and left to the context, i.e. the attacker. In such a case also a private channel to this untrusted authority can be modeled as a public channel, i.e. left out in $\tilde{n}$.

Moreover, note that our definition does not specify how the result is computed, nor what information it contains. For example, the result could be only the name of the winning candidate(s) – which might be computed in a complex way, e.g. in a Single Transferable Vote (STV) system as used in Ireland [Cit13] –, or also the number of votes for each candidate. We implicitly assume the latter case for most of our examples, but our notions (and in particular the generalizations in Sections 3.4 and 3.5) can also be applied to protocols implementing more complex counting algorithms.

Example 12 *As a running example, we consider the following simple voting protocol.*

Informal description: *To construct his ballot, each voter encrypts his vote with the administrator's public key and signs it using his secret key. The resulting ballot is posted on the bulletin board. After the voting deadline is over, the administrator checks if each ballot is signed by an eligible voter. He then decrypts the correct ballots and publishes the result.*

Formal description in our model: *The protocol uses probabilistic public-key encryption and signatures, which we model using the following equational theory:*

$$\begin{aligned} \mathtt{dec}(\mathtt{enc}(m, \mathtt{pk}(sk), r), sk) &= m \\ \mathtt{checksign}(\mathtt{sign}(m, sk), \mathtt{pk}(sk)) &= m \end{aligned}$$

The first equation states that a message m encrypted using a public key can be decrypted correctly using the corresponding secret key. The second equation states that the verification with a public key succeeds if it was created using the corresponding secret key.

The protocol is then a tuple $(V, A, \emptyset)$ where

$$\begin{aligned} A &= A' \{^{pkv_1}/_{pkv}\} \mid \ldots \mid A' \{^{pkv_n}/_{pkv}\} \\ A' &= \mathtt{in}(ch, (sig, vote)). \\ &\quad \mathtt{if\ checksign}(sig, pkv) = vote \\ &\quad \mathtt{then\ sync}\ 1.\mathtt{out}(chR, \mathtt{dec}(vote, ska)) \\ &\quad \mathtt{else}\ 0 \\ V &= \nu r.\mathtt{let}\ evote = \mathtt{enc}(v, pka, r)\ \mathtt{in} \\ &\quad \mathtt{out}(ch, (\mathtt{sign}(evote, skv), evote)) \end{aligned}$$

In this example we use another important syntactical extension of the Applied π-Calculus: `sync` 1 *is a synchronization point with the following semantics: No process can continue until all other processes have reached the synchronization point. Such a behavior can directly be expressed in the standard Applied π-Calculus as follows (similarly to the idea of Delaune et al. [DKR09]): The synchronization command is replaced with an output of a message* 1 *on a restricted channel $chSync1$ (*$\mathtt{out}(chSync1, 1)$*), followed by an input on a different restricted channel $chSync2$ (*$\mathtt{in}(chSync2, m)$*). An additional synchronization process is added, receiving all messages on the private channel, and outputting messages on the second channel once all messages have been received:*

$$\begin{aligned} P_{sync} &= \mathtt{in}(chSync1, x_1) \ldots \mathtt{in}(chSync1, x_n). \\ &\quad (\mathtt{out}(chSync2, 1) \mid \ldots \mid \mathtt{out}(chSync2, 1)) \end{aligned}$$

This ensures that the processes can only continue once all n processes have reached

the synchronization point, as they have to wait for the answer by synchronization process. The context cannot interfere with the synchronization, as both channels chSync1 and chSync2 are restricted. ProSwapper [KSR10], a preprocessor for ProVerif, implements similar synchronization points, but also allows to swap the entire data of the synchronizing processes to help ProVerif proving observational equivalences.

Sometimes we need to model a synchronization where k out of n processes are sufficient to enable continuation. In that case we write $\texttt{sync}_k$ 1*, and the implementation is similar:*

$$\begin{aligned} P_{sync_k} \quad = \quad & \texttt{in}(chSync1, x_1) \ldots \texttt{in}(chSync1, x_k). \\ & (\texttt{out}(chSync2, 1) | \ldots | \texttt{out}(chSync2, 1)) \end{aligned}$$

In this example the substitution determining the identity of a voter assigns the secret key, e.g. $\sigma_{id_k} = \{^{sk_k}/_{skv}\}$*. The substitution specifying the vote as for example a vote for candidate a would be* $\sigma_{v_k} = \{^{a}/_{v}\}$*.*

In some of our definitions we also need to reason about instances where all parties, in particular all authorities, are honest.

Definition 15 (Honest Voting Process) *An* Honest Voting Process *of a voting protocol* $(V, A_1, \ldots, A_m, \tilde{c})$ *is a closed plain process*

$$\nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1} | \ldots | V\sigma_{id_n}\sigma_{v_n} | A_1 | \ldots | A_m)$$

where $\tilde{n}$ *includes the secret channel names,* $V\sigma_{id_i}\sigma_{v_i}$ *are the processes executed by the voters where:*

- σ_{id_i} *is a substitution assigning the identity to a process (this determines for example the secret keys),*
- σ_{v_i} *specifies the vote(s) and if the voter abstains,*

and $A_j s$ *are the election authorities.*

Given a voting process

$$VP = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1} | \ldots | V\sigma_{id_n}\sigma_{v_n} | A_1 | \ldots | A_l)$$

we denote by VP^H *the corresponding honest voting process, i.e.*

$$VP^H = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1} | \ldots | V\sigma_{id_n}\sigma_{v_n} | A_1 | \ldots | A_m)$$

Note that for our running example a voting process and its corresponding honest voting process coincide, as the one and only authority is required to be honest anyway.

We also often need to replace some voters inside a voting process. For this we use *Voting Contexts*.

Definition 16 ((Honest) Voting Context) *Given a voting process*

$$VP = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1}|\ldots|V\sigma_{id_n}\sigma_{v_n}|A_1|\ldots|A_l)$$

and a subset of voters I we define the Voting Context $VP_I[\_]$ *as follows:*

$$VP_I[\_] = \nu\tilde{n}.(\ |_{i\notin I} V\sigma_{id_i}\sigma_{v_i}|\_|A_1|\ldots|A_l)$$

Similarly for an honest voting process

$$VP^H = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1}|\ldots|V\sigma_{id_n}\sigma_{v_n}|A_1|\ldots|A_m)$$

and a subset of voters I we define the Honest Voting Context $VP^H_I[\_]$ *as follows:*

$$VP^H_I[\_] = \nu\tilde{n}.(\ |_{i\notin I} V\sigma_{id_i}\sigma_{v_i}|\_|A_1|\ldots|A_m)$$

Consider the following example.

Example 13 *An instance of our simple voting protocol (Example 12 on page 55) with two voters A and B looks as follows:*

$$\begin{aligned} VP = \nu sk_A.\nu sk_B.\nu sk_{Ad}.(&V\,\{^{sk_A}/_{skv}, {}^{\mathtt{pk}(sk_{Ad})}/_{pka}\}\,\{^{a}/_{v}\}\,|\\ &V\,\{^{sk_B}/_{skv}, {}^{\mathtt{pk}(sk_{Ad})}/_{pka}\}\,\{^{b}/_{v}\}\,|A'\,\{^{sk_{Ad}}/_{ska}, {}^{\mathtt{pk}(sk_A)}/_{pkv}\}\,|A'\,\{^{sk_{Ad}}/_{ska}, {}^{\mathtt{pk}(sk_B)}/_{pkv}\}) \end{aligned}$$

The corresponding voting context $VP_{\{A\}}$ then looks as follows:

$$\begin{aligned} VP_{\{A\}} = \nu sk_A.\nu sk_B.\nu sk_{Ad}.(\_|&V\,\{^{sk_B}/_{skv}, {}^{\mathtt{pk}(sk_{Ad})}/_{pka}\}\,\{^{b}/_{v}\}\,|\\ &A'\,\{^{sk_{Ad}}/_{ska}, {}^{\mathtt{pk}(sk_A)}/_{pkv}\}\,|A'\,\{^{sk_{Ad}}/_{ska}, {}^{\mathtt{pk}(sk_B)}/_{pkv}\}) \end{aligned}$$

Finally, we define abstention for a voting process. An abstaining voter does not send any message on any channel, in particular no ballot. In the real world, this would correspond to a voter that does not even go to polling station. This is different from just voting for a particular "null" candidate denoted $\perp$, which still results in sending a ballot (a *blank* vote).

However, we need to allow it to execute synchronization points, because synchronization is essential to ensure privacy, as otherwise an attacker in control of the network can simply keep track of a ballot and break privacy by linking the vote to a voter. Synchronization allows to break such links. Yet, if we allow abstention, we have to allow the non-abstaining voters to synchronize normally, otherwise they will block and no result can be announced. Hence we allow

abstaining voters to execute synchronization operations, and obtain the following definition.

Definition 17 (Abstention) *Let $A \setminus sync$ denote the process A where all* `sync` *instructions have been removed. Then a substitution σ_{v_i} makes a voter $V\sigma_{id_i}$ abstain if $(V\sigma_{id_i}\sigma_{v_i}) \setminus sync \approx_l 0$.*

In our privacy definitions we also consider (partly) corrupted voters. To formally define corrupted parties or voter revealing secrets to the attacker, we use the following two definitions. The first one turns a process P into another process P^{ch} that reveals all its inputs and secret data on the channel ch. This can be seen as trying to construct a receipt.

Definition 18 (Process P^{ch} [DKR09]) *Let P be a plain process and ch be a channel name. We define P^{ch} as follows:*

— $0^{ch} \mathrel{\hat{=}} 0$,

— $(P|Q)^{ch} \mathrel{\hat{=}} P^{ch}|Q^{ch}$,

— $(\nu n.P)^{ch} \mathrel{\hat{=}} \nu n.\mathtt{out}(ch, n).P^{ch}$ *when n is a name of base type,*

— $(\nu n.P)^{ch} \mathrel{\hat{=}} \nu n.P^{ch}$ *otherwise,*

— $(\mathtt{in}(u, x).P)^{ch} \mathrel{\hat{=}} \mathtt{in}(u, x).\mathtt{out}(ch, x).P^{ch}$ *when x is a variable of base type,*

— $(\mathtt{in}(u, x).P)^{ch} \mathrel{\hat{=}} \mathtt{in}(u, x).P^{ch}$ *otherwise,*

— $(\mathtt{out}(u, M).P)^{ch} \mathrel{\hat{=}} \mathtt{out}(u, M).P^{ch}$,

— $(!P)^{ch} \mathrel{\hat{=}} !P^{ch}$,

— $(\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q)^{ch} \mathrel{\hat{=}} \mathtt{if}\ M = N\ \mathtt{then}\ P^{ch}\ \mathtt{else}\ Q^{ch}$.

In the remainder we assume that $ch \notin fn(P) \cup bn(P)$ before applying the transformation.

The second definition does not only reveal the secret data, but also takes orders from an outsider before sending a message or branching, i.e. the process is under complete remote control.

Definition 19 (Process P^{c_1,c_2} [DKR09]) *Let P be a plain process and c_1, c_2 be channel names. We define P^{c_1,c_2} as follows:*

— $0^{c_1,c_2} \mathrel{\hat{=}} 0$,

— $(P|Q)^{c_1,c_2} \mathrel{\hat{=}} P^{c_1,c_2}|Q^{c_1,c_2}$,

— $(\nu n.P)^{c_1,c_2} \mathrel{\hat{=}} \nu n.\mathtt{out}(c_1, n).P^{c_1,c_2}$ *when n is a name of base type,*

— $(\nu n.P)^{c_1,c_2} \mathrel{\hat{=}} \nu n.P^{c_1,c_2}$ *otherwise,*

— $(\mathtt{in}(u, x).P)^{c_1,c_2} \mathrel{\hat{=}} \mathtt{in}(u, x).\mathtt{out}(c_1, x).P^{c_1,c_2}$ *when x is a variable of base type and x is a fresh variable,*

— $(\mathtt{in}(u,x).P)^{c_1,c_2} \mathrel{\hat{=}} \mathtt{in}(u,x).P^{c_1,c_2}$ *otherwise,*

— $(\mathtt{out}(u,M).P)^{c_1,c_2} \mathrel{\hat{=}} \mathtt{in}(c_2,x).\mathtt{out}(u,x).P^{c_1,c_2}$,

— $(!P)^{c_1,c_2} \mathrel{\hat{=}} !P^{c_1,c_2}$,

— $(\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q)^{c_1,c_2} \mathrel{\hat{=}} \mathtt{in}(c_2,x).\mathtt{if}\ x = \mathit{true}\ \mathtt{then}\ P^{c_1,c_2}\ \mathtt{else}\ Q^{c_1,c_2}$ *where x is a fresh variable and true is a constant.*

To hide the output of a process, we use the following definition.

Definition 20 (Process $A^{\setminus out(ch,\cdot)}$ [DKR09]) *Let A be an extended process. We define the process $A^{\setminus out(ch,\cdot)}$ as $\nu ch.(A|!in(ch,x))$.*

Given these definitions, we recall two lemmas by Delaune et al. [DKR09] used in our proofs.

Lemma 16 ([DKR09]) *Let P be a closed plain process and ch a channel name such that $ch \notin fn(P) \cup bn(P)$. We have $(P^{ch})^{\setminus out(ch,\cdot)} \approx_l P$.*

Lemma 17 ([DKR09]) *Let $C_1 = \nu\tilde{u}_1.(\_|B_1)$ and $C_2 = \nu\tilde{u}_2.(\_|B_2)$ be two evaluation contexts such that $\tilde{u}_1 \cap (fv(B_2) \cup fn(B_2)) = \emptyset$ and $\tilde{u}_2 \cap (fv(B_1) \cup fn(B_1)) = \emptyset$. Then we have $C_1[C_2[A]] \equiv C_2[C_1[A]]$ for any extended process A.*

3.3.2 — Defining Privacy: A Modular Approach

Privacy in eVoting is notoriously difficult to define. As discussed above, many different types of attacks and different attacker capabilities are considered. To systematize the existing approaches, we propose a modular definition, trying to factor out the different independent types of attacks and attacker capabilities.

In our setting, the attacker targets one voter (the *targeted voter*) and tries to extract information about the targeted voter's vote(s). If the attacker knows the votes of all other voters, he can infer the targeted voter's vote from the result. Thus we suppose that he is unsure about the vote of one other voter. This voter is called the *counterbalancing voter*, as he counterbalances different votes by the attacked voter to ensure that the result remains unchanged.

We express privacy as an observational equivalence. Intuitively, an attacker should not be able to distinguish between an execution in which the targeted voter behaves and votes as the attacker wishes, and another execution where he only pretends to do so and votes differently. To ensure that the attacker cannot tell the difference by just comparing the result, the counterbalancing voter compensates by simply swapping votes with the targeted voter.

Starting from the definitions of Coercion-Resistance, Receipt-Freeness, and Vote-Privacy in the literature we propose to factor out the three following dimensions: Communication between the attacker and the targeted voter, insider or outside attacker, and security against forced-abstention-attacks.

1. *Communication between the attacker and the targeted voter:* We define three different levels, corresponding to different attacker capabilities:
   (a) In the simplest case, the attacker only observes publicly available data and communication. We call this case Swap-Vote-Privacy[2], denoted $SwVP$.
   (b) In the second case, the targeted voter tries to convince the attacker that he voted for a certain candidate by revealing his secret data. Yet the attacker should not be able to determine if he actually sent his real data, or a fake receipt. We call this case Swap-Receipt-Freeness, denoted $SwRF$.
   (c) In the strongest case, the voter pretends to be completely under the control of the attacker, i.e. to reveal his secret data and to follow the intruder's instructions. Yet the attacker should be unable to determine if he actually complied with his instructions or if he only pretended to do so. We call this case Swap-Coercion-Resistance, denoted $SwCR$.

   Intuitively Swap-Coercion-Resistance is stronger than Swap-Receipt-Freeness, which is stronger than Swap-Vote-Privacy ($SwCR > SwRF > SwVP$).
2. *Inside or outside attacker:* The attacker may control another legitimate voter (neither the targeted nor the counterbalancing voter). In that case he could be able to compromise privacy by trying to relate the corrupted voter's vote to the targeted voter's vote (e.g. by copying it as in the attack by Smyth and Cortier [SC11]) or using the corrupted voter's secret data, such as his credentials or keys[3]. In our definitions, we distinguish two cases for the *Attacker*, corresponding again to different attacker capabilities:
   (a) *Attacker* is an Outsider (denoted O): The attacker is an external observer.
   (b) *Attacker* is an Insider (denoted I): The attacker has corrupted a legitimate voter.

   Again, Insider is intuitively the stronger setting ($I > O$).
3. *Security against forced-abstention-attacks:* A protocol can ensure that a voter can still vote as intended, although a coercer wants him to abstain. Note that in contrast to the literature [JCJ05, BHM08], we define this property independent of Coercion-Resistance, as we also want to apply it in the case of Vote-Privacy. This is because we see it as a different type of attack, that can be combined with the different attacker capabilities defined above. Our model expresses (in)security against forced-abstention attack by requiring the observational equivalence to hold:

[2]We include "*swap*" in the name to distinguish these notions based on swapping votes from the generalized notions in Section 3.4.

[3]Here we only consider a single corrupted voter, the generalization to multiple corrupted voters is discussed in Section 3.4.4.

(a) in any case, i.e. even if the voter is forced to abstain. We call this case security against Forced-Abstention-Attacks, denoted FA.

(b) if the targeted voter does not abstain from voting (i.e. always participates). We call this case Participation Only, denoted PO.

In this dimension security against Forced-Abstention-Attacks is a stronger property than Participation Only ($FA > PO$).

The strongest possible property is thus $SwCR^{I,FA}$, the weakest $SwVP^{O,PO}$. If we leave out the parameter, we take the weakest setting as a default, i.e. $SwVP$ denotes $SwVP^{O,PO}$.

3.3.3 — Definitions in the Applied π-Calculus

Our definition is parameterized using the following parameters (as explained above):

— $Privacy = \{SwCR, SwRF, SwVP\}$ ("Swap-Coercion-Resistance", "Swap-Receipt-Freeness" or "Swap-Vote-Privacy").

— $Attacker = \{I, O\}$ ("Insider" or "Outsider").

— $Abs = \{FA, PO\}$ ("Security against Forced-Abstention-Attacks"or "Participation Only").

Definition 21 (*$Privacy^{Attacker,Abs}$*) *A protocol fulfills $Privacy^{Attacker,Abs}$ if for any voting process*

$$VP = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1}|\ldots|V\sigma_{id_n}\sigma_{v_n}|A_1|\ldots|A_l)$$

there exists a process V' such that for all votes σ_{v_A} and σ_{v_B} where $V\sigma_{v_B}$ does not make a voter abstain[4]*, one of the following holds depending on the privacy setting:*

— *if Privacy is Swap-Vote-Privacy ($SwVP$):*

$$VP_I\left[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right]$$

— *if Privacy is Swap-Receipt-Freeness ($SwRF$):*

- $V'^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B}$
- $VP_I\left[(V\sigma_{id_A}\sigma_{v_A})^{chc}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[V'|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right]$

— *if Privacy is Swap-Coercion-Resistance ($SwCR$):*
For any context $C = \nu c_1.\nu c_2.(\_|P')$ with $\tilde{n} \cap fn(C) = \emptyset$ and
$VP_I\left[C\left[(V\sigma_{id_A})^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[(V\sigma_{id_A}\sigma_{v_A})^{chc}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right]$
we have

- $C\left[V'\right]^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B}$

[4] This condition is needed to ensure that in the case PO no voter can abstain.

– $VP_I\,[C\,[(V\sigma_{id_A})^{c_1,c_2}]\,|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C] \approx_l VP_I\,[C\,[V']\,|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C]$

where

— *If Attacker is:*

– *Insider(I):* $I := \{A, B, C\}$ *and* $\mathcal{V}_C := (V\sigma_{id_C})^{c_1,c_2}$

– *Outsider (O):* $I := \{A, B\}$ *and* $\mathcal{V}_C := 0$

— *If Abs is:*

– *Participation Only (PO):* $V\sigma_{id_A}$ *does not abstain*

– *Security against Forced-Abstention-Attacks (FA): he may abstain.*

As we require the conditions to hold for any voting process VP, they have to hold for any number of honest voters[5].

The following examples illustrate how the parameter values instantiate the definition and give corresponding intuitions. We start with simple privacy.

Example 14 ($SwVP^{O,PO}$) *A protocol fulfills $SwVP^{O,PO}$ if for any voting process VP such that for all votes σ_{v_A} and σ_{v_B} where σ_{v_B} and σ_{v_A} does not make a voter abstain we have:*

$$VP_{\{A,B\}}\,[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}|0] \approx_l VP_{\{A,B\}}\,[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}|0] \tag{3.1}$$

This coincides with the definition of Vote-Privacy given by Delaune et al. [DKR09]: Two situations where two voters swap votes are bisimilar. We also note that Receipt-Freeness in the DKR-model corresponds to $SwRF^{O,PO}$ in our model, and Coercion-Resistance in the DKR-model corresponds to $SwCR^{O,PO}$ in our model.

In our next example, we apply the above definition on our running example, the simple voting protocol.

Example 15 (Application) *Consider our running example of a simple voting protocol. We show that it ensures $SwVP^{O,PO}$ as defined above using ProVerif. We assume the secret keys to be private (hence $\tilde{n}$ includes the secret keys of the voters and the administrator), and the administrator to be honest. In that case, ProVerif is able to prove the bisimilarity in equation (3.1)*[6]*, which gives that the simple*

[5] In our proofs we usually only consider the case of two or three voters, respectively, as this is the base case. These proofs do however generalize in a straightforward way to any number of participants. Additionally, if the protocol is modular, i.e. if instances can be composed (cf. Definition 29 on page 128 in Section 3.4.4), then the problem of many honest voters automatically reduces to the base case of two or three voters.

[6] Note that the calculus used by ProVerif differs in some technical details from the original Applied π-Calculus. This is mainly due to the different extensions included in ProVerif, but also the semantics have been slightly simplified as the original semantics are often non-deterministic and hence difficult to reason about. Although the simplified semantics appear to be sound, there is no formal proof [Bla13], thus technically we cannot claim that the result also holds in the Applied π-Calculus. Yet we argue that ultimately our model and definitions remain as meaningful in the ProVerif-calculus as in the Applied π-Calculus.

voting protocol ensures $SwVP^{O,PO}$. The code used is available online [Dre13].

It is easy to see that this protocol does not guarantee Swap-Vote-Privacy for an inside attacker ($SwVP^{I,PO}$), as he can simply access the votes on the bulletin board and copy them[7]*. He can identify which vote was posted by which voter using the signatures.*

The protocol is not receipt-free ($SwRF^{Attacker,Abs}$) either as the randomness used for encrypting the vote can be used as a receipt. Since the bulletin board reveals which voters participated through the signatures, it is not resistant against forced-abstention attacks.

In the following example we illustrate how security against Forced-Abstention-Attacks is captured by our definition.

Example 16 ($SwVP^{O,FA}$) *A protocol fulfills $SwVP^{O,FA}$ if for any voting process VP such that for all votes σ_{v_A} and σ_{v_B} where σ_{v_B} does not make a voter abstain we have:*

$$VP_{\{A,B\}}\left[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}|0\right] \approx_l VP_{\{A,B\}}\left[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}|0\right]$$

In this case, σ_{v_A} can make a voter abstain. As σ_{v_B} may not specify abstention, we have an observational equivalence between a situation where the targeted voter abstains, and a situation where he votes and the counterbalancing voter abstains. This captures the security against forced-abstention-attacks.

Now we illustrate how we capture Receipt-Freeness, and how the inside attacker is modeled.

Example 17 ($SwRF^{I,PO}$) *A protocol fulfills $SwRF^{I,PO}$ if for any voting process VP there exists a process V' such that for all votes σ_{v_A} and σ_{v_B} where σ_{v_B} and σ_{v_A} do not make a voter abstain we have*

$$V'^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B}$$

and

$$VP_{\{A,B,C\}}\left[(V\sigma_{id_A}\sigma_{v_A})^{chc}|V\sigma_{id_B}\sigma_{v_B}|(V\sigma_{id_C})^{c_1,c_2}\right] \approx_l VP_{\{A,B,C\}}\left[V'|V\sigma_{id_B}\sigma_{v_A}|(V\sigma_{id_C})^{c_1,c_2}\right].$$

[7] Technically the attacker copies the vote from the bulletin board, and sends it to his corrupted voter using the channel c_2. The corrupted voter then submits the signed vote to the bulletin board. Since in our modeling everybody can post on the bulletin board, the attacker can also copy the vote, sign it using the key of his corrupted voter (which he received on channel c_1) and then submit it directly.

The main idea of this Receipt-Freeness definition is the following: a protocol is receipt-free if an attacker cannot distinguish between a voter revealing honestly all his secret data as a receipt, and a voter only giving away fake information and voting differently. This counter-strategy is expressed here by the process V'. The labeled bisimilarity then has to hold for a situation where voter id_A votes v_A and reveals all his secret data honestly, and a situation where he employs the counter-strategy V'. The first equation ensures that he votes v_B in the case of the counter-strategy.

To account for an inside attacker we include a corrupted voter id_C. The adversary can hence try to execute vote-copy attacks, or employ insider knowledge.

Finally we discuss Coercion-Resistance.

Example 18 (*$SwCR^{O,FA}$*) *A protocol fulfills $SwCR^{O,FA}$ if for any voting process VP there exists a process V' such that for all votes σ_{v_A} and σ_{v_B} where $V\sigma_{v_B}$ does not make a voter abstain, the following holds: For any context $C = \nu c_1.\nu c_2.(\_|P')$ with $\tilde{n} \cap fn(C) = \emptyset$ and*

$$VP_{\{A,B\}}\left[C\left[(V\sigma_{id_A})^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}|0\right] \approx_l VP_{\{A,B\}}\left[(V\sigma_{id_A}\sigma_{v_A})^{chc}|V\sigma_{id_B}\sigma_{v_B}|0\right]$$

we have

— $C\left[V'\right]^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B}$

— $VP_{\{A,B\}}\left[C\left[(V\sigma_{id_A})^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}|0\right] \approx_l VP_{\{A,B\}}\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{v_A}|0\right]$

The intuition behind this definition is the following: the context C also belongs to the attacker and tries to force the targeted voter to vote for a certain candidate or to make him abstain (depending on σ_{id_A}), whereas V' tries to vote differently and to escape coercion. The condition on the context C ensures that the voter is actually forced to vote for the candidate σ_{v_A} to ensure that both sides give the same result, and hence are not trivially distinguishable.

3.3.4 — Hierarchy

As already announced in the informal description in Section 3.3.2, we have a hierarchy of notions in each of the three dimensions.

Lemma 18 *For $Privacy \in \{SwVP, SwRF, SwCR\}$, $Attacker \in \{I, O\}$ and $Abs \in \{FA, PO\}$ we have:*

1. *Any attack that works for an outsider can also be used for an insider: If a protocol respects $Privacy^{I,Abs}$, then it also respects $Privacy^{O,Abs}$.*
2. *If a protocol is secure against Forced-Abstention attacks, it is also secure in the "Participation Only" case: If a protocol respects $Privacy^{Attacker,FA}$, it also respects $Privacy^{Attacker,PO}$.*

3. *Coercion-Resistance is stronger than Receipt-Freeness, which is stronger than Vote-Privacy:*
   — *If a protocol respects* $SwCR^{Attacker,Abs}$*, it also respects* $SwRF^{Attacker,Abs}$*.*
   — *If a protocol respects* $SwRF^{Attacker,Abs}$*, it also respects* $SwVP^{Attacker,Abs}$*.*

Proof We consider the different propositions independently.

1. We detail the case $SwVP^{I,Abs}$, the other cases are analogous. We use a proof by contradiction: suppose that a protocol does not ensure $SwVP^{O,Abs}$, but $SwVP^{I,Abs}$ holds. Then we have an instance where

$$VP_{\{A,B\}}\left[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}\right] \not\approx_l VP_{\{A,B\}}\left[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}\right]$$

   which we can rewrite as

$$VP_{\{A,B,i\}}\left[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}|V\sigma_{id_i}\sigma_{v_i}\right] \not\approx_l VP_{\{A,B,i\}}\left[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}|V\sigma_{id_i}\sigma_{v_i}\right]$$

   for some honest voter $V\sigma_{id_i}\sigma_{v_i}$. This yields the contradiction for an attacker enforcing that the corrupted voter behaves like an honest voter, i.e. a context $C = \nu c_1.\nu c_2.(P|\_)$ such that $C[V\sigma_{id_C}^{c_1,c_2}] \approx_l V\sigma_{id_i}\sigma_{v_i}$, as we have an instance violating the definition of $SwVP^{I,Abs}$. Note that technically this also works if the instance violating privacy contains only two voters, in this case the intruder forces the corrupted voter to do nothing.
2. This holds by definition: in the case FA we consider all σ_{v_A}, whereas we exclude some in the case PO. Thus, if the bisimilarity holds for FA, it also holds for PO.
3. Coercion-Resistance is stronger than Receipt-Freeness, which is stronger than Vote-Privacy:
   — The proof is similar to the proof showing that Coercion-Resistance implies Receipt-Freeness in the DKR model [DKR09]:
   Assume that we have a protocol ensuring $SwCR^{Attacker,Abs}$. Let C be an evaluation context such that $C = \nu c_1.\nu c_2.(\_|P)$ for some plain process P which fulfills

$$VP_I\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[V\sigma_{id_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \quad (3.2)$$

   Note that such a C can be constructed directly from the vote process V. By the definition of $SwCR^{Attacker,Abs}$ we know that there is a closed

plain process V' such that

$$C\left[V'\right]^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B} \tag{3.3}$$

and

$$VP_I\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right] \tag{3.4}$$

To prove that the protocol ensures $SwRF^{Attacker,Abs}$ we have to find another process V'' such that

$$V''^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B} \tag{3.5}$$

and

$$VP_I\left[V\sigma_{id_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[V''|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right] \tag{3.6}$$

Let $V'' = C[V']$. This directly fulfills the first requirement (3.5) by (3.3). By the condition (3.2) on C we have:

$$VP_I\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[V\sigma_{id_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right]$$

The second hypothesis (3.4) gives

$$VP_I\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right]$$

As labeled bisimilarity is transitive, we can conclude

$$VP_I\left[V\sigma_{id_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right]$$

which gives us the desired result for $V'' = C[V']$.

— The proof is similar to the proof of Receipt-Freeness implies Vote-Privacy in the DKR-model [DKR09]:
Assume that we have a protocol ensuring $SwRF^{Attacker,Abs}$. By hypothesis there is a closed plain process V' so that

$$V'^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B} \tag{3.7}$$

and

$$VP_I\left[V\sigma_{id_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[V'|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right].$$

We apply the context $\nu chc.(\_|!\mathtt{in}(chc,x))$ (cf. Definition 20 on page 60)

on both sides, which gives

$$VP_I\left[V\sigma_{id_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right]^{\backslash out(chc,\cdot)} \approx_l VP_I\left[V'|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right]^{\backslash out(chc,\cdot)}.$$

By using Lemma 17 on page 60 on both sides of the equivalence we obtain

$$VP_I\left[V'|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right]^{\backslash out(chc,\cdot)} \equiv VP_I\left[V'^{\backslash out(chc,\cdot)}|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right]$$

and

$$VP_I\left[V\sigma_{id_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right]^{\backslash out(chc,\cdot)} \equiv VP_I\left[\left(V\sigma_{id_A}\sigma_{v_A}^{chc}\right)^{\backslash out(chc,\cdot)}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right].$$

We can now apply Lemma 16 on page 60 and use the fact that labeled bisimilarity is closed under structural equivalence to obtain

$$VP_I\left[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[V'^{\backslash out(chc,\cdot)}|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right]$$

where we can apply (3.7) to conclude

$$VP_I\left[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l VP_I\left[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}|\mathcal{V}_C\right]$$

i.e. that the protocol ensures $SwVP^{Attacker,Abs}$. □

Taking these properties together, we arrive at the hierarchy shown in Figure 3.1 on the facing page.

3.3.5 — Case Studies

We applied our family of notions on several case studies, chosen to show that each of our dimensions corresponds to a different property of existing protocols. The results are summed up in and Table 3.1 on page 114, the position of the case studies within our hierarchy is shown in Figure 3.1 on the facing page.

§ 3.3.5.1. Protocol by Fujioka, Okamoto and Ohta (FOO). The protocol by Fujioka, Okamoto and Ohta [FOO92] is based on blind signatures and commitments. It was shown to ensure Vote-Privacy [DKR09] in the DKR-model, but is not receipt-free as the randomness of the commitment can be used as a receipt. We show that it ensures $SwVP^{I,PO}$ (i.e. Swap-Vote-Privacy against an insider, but just in the Participation Only case).

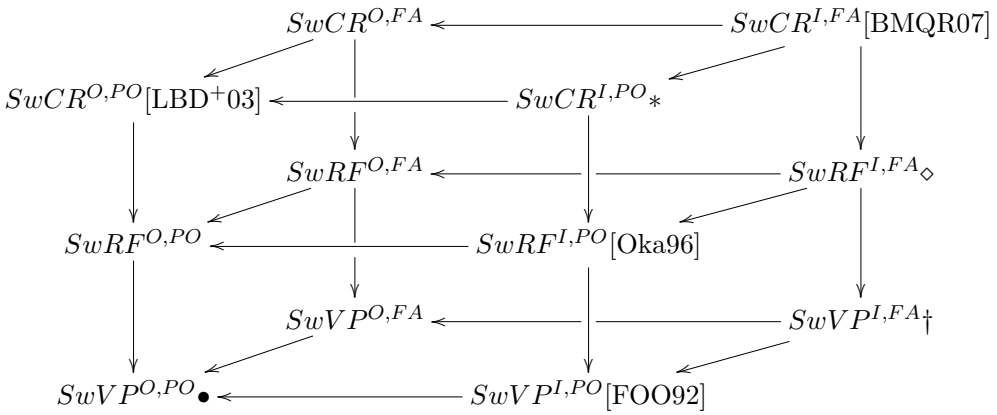

Figure 3.1 – Hierarchy of privacy notions with examples: The simple voting protocol, our running example (•); Bingo Voting [BMQR07]; Bingo Voting with voter lists (∗); Okamoto [Oka96]; Okamoto with a private channel to the honest administrator (⋄); FOO [FOO92]; FOO with a private channel to the honest administrator (†); and Lee et al. [LBD+03]. $A \rightarrow B$ means that a protocol ensuring A also ensures B.

Protocol Description. The protocol is split into three phases. In the first phase the administrator signs the voter's commitment to his vote:

— Voter V_i chooses his vote v_i and computes a commitment $x_i = \xi(v_i, r_i)$ for a random key r_i.

— He blinds the commitment using a blinding function χ, a random value b_i and obtains $e_i = \chi(x_i, b_i)$.

— He signs e_i and sends the signature $s_i = \sigma_{V_i}(e_i)$ together with e_i and his identity to the administrator.

— The administrator checks if V_i has the right to vote and has not yet voted, and if the signature s_i is correct. If all tests succeed, he signs $d_i = \sigma_A(e_i)$ and sends it back to V_i.

— V_i checks the signature, and unblinds the signature to obtain $y_i = \delta(d_i, b_i) = \sigma_A(x_i)$.

In the second phase, the voter submits his ballot:

— Voter V_i sends (x_i, y_i) to the collector C through an anonymous channel.

— The collector checks the administrator's signature and enters (x_i, y_i) as the l-th entry into a list.

When all ballots are cast or when the deadline is over, the counting phase begins:

— The collector publishes the list of correct ballots.

— V_i verifies that his commitment appears on the list and sends (l, r_i) to C using an anonymous channel.

— The collector C opens the l-th ballot using r_i and publishes the vote.

Model in the Applied π-Calculus. We use the following equational theory:

$$\begin{aligned} \texttt{open}(\texttt{commit}(m,r),r) &= m \\ \texttt{checksign}(\texttt{sign}(m,sk),\texttt{pk}(sk)) &= m \\ \texttt{unblind}(\texttt{blind}(m,r),r) &= m \\ \texttt{unblind}(\texttt{sign}(\texttt{blind}(m,r),sk),r) &= \texttt{sign}(m,sk) \end{aligned}$$

We can then express the behavior of the different parties using the following processes.

```
1  (* private keys *)
2  νska.νskva.νskvb.νskvc.
3  (* public keys *)
4  let (pka,pkva,pkvb,pkvc)
5   = (pk(ska),pk(skva),pk(skvb),pk(skvc)) in
6  (* public key disclosure *)
7  out(ch,pka).out(ch,pkva).out(ch,pkvb).out(ch,pkvc).
8  (* administrators *)
9  ((let pkv = pkva in processA) |
10  (let pkv = pkvb in processA) |
11  (let pkv = pkvc in processA) |
12  processC | processC | processC |
13 (* voters *)
14  (let skv = skva in let v = a in processV) |
15  (let skv = skvb in let v = b in processV) |
16  (let skv = skvc in let v = c in processV))
```

Listing 3.1 – The main process

The *main process* (Listing 3.1) sets up the keys and executes the participation processes (three voters, three administrators and three collectors – one for each voter –) in parallel. Note that `(* ... *)` are comments, inspired by ProVerif's syntax, and that `processA`, `processK`, `processC` and `processV` are variables for the processes of the administrator, keying process, collector and voter as defined below. Note also that although much of this syntax is inspired by ProVerif, we still rely on the original Applied π-Calculus.

The *voter*'s process (Listing 3.2 on the facing page) starts by deciding if he abstains: If yes, he does nothing except for the synchronization points. This is necessary as otherwise he blocks the other voters. If he does not abstain, he votes following the protocol described informally before.

A *voter controlled by the attacker* is modeled by the process given in Listing 3.3 on page 72. This process is obtained when computing $\texttt{processV}^{c_1,c_2}$ as defined in Definition 19 on page 59.

```
processV =
 if v = abstain then
  sync 1.
  sync 2
 else
  νb.νr.
   let commitedvote = commit(v,r) in
   let blindedvote = blind(commitedvote,b) in
   out(ch1,(pk(skv),sign(blindedvote,skv),blindedvote)).
   in(ch2,m2).
   let result = checksign(m2,pka) in
   if result = blindedvote then
    let signedvote = unblind(m2,b) in
    sync 1.
    out(ch3,(commitedvote,signedvote)).
    in(ch4,(l,commitedvote',signedvote')).
    if commitedvote' = commitedvote then
     if signedvote' = signedvote then
      sync 2.
      out(ch5,(l,r))
```

Listing 3.2 – The voting process

When the *administrator* (Listing 3.4 on the next page) receives the blinded commitment, he checks the signature, signs, and sends the result back.

The *collector* (Listing 3.5 on the following page) verifies the signature on incoming commitments using the administrator's public key. If the signature is correct, he creates a new bounded name l (the number in the list) and sends it together with the signed commitment back to the voter. The voter then reveals his randomness, which the collector uses to open the commitment.

Analysis. Delaune et al. showed that the protocol ensures Vote-Privacy in their model [DKR09] (which corresponds to $SwVP^{O,PO}$ in our model, as they do not consider inside attackers). We show that it also ensures Vote-Independence, i.e. the stronger notion of Vote-Privacy for inside attackers: $SwVP^{I,PO}$.

Theorem 19 *FOO respects* $SwVP^{I,PO}$.

Proof We show that

$$\begin{aligned}&\texttt{processV}\,\{^{skva}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{a}/_{v}\}\,|\texttt{processV}\,\{^{skvb}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{b}/_{v}\}\,|\\ &\qquad\texttt{processVc1c2}\,\{^{skvc}/_{skv},{}^{pk(ska)}/_{pka}\}\\ &\qquad\approx_l\\ &\texttt{processV}\,\{^{skva}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{b}/_{v}\}\,|\texttt{processV}\,\{^{skvb}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{a}/_{v}\}\,|\\ &\qquad\texttt{processVc1c2}\,\{^{skvc}/_{skv},{}^{pk(ska)}/_{pka}\}\end{aligned}$$

```
processVc1c2 =
 in(c2,y1).
 if y1 = true then
  sync 1.
  sync 2
 else
  νb.out(c1,b).νr.out(c1,r).
  let commitedvote = commit(v,r) in
  let blindedvote = blind(commitedvote,b) in
  in(c2,y2).out(ch1,y2).
  in(ch2,m2).out(c1,m2).
  let result = checksign(m2,pka) in
  in(c2,y3).
  if y3 = true then
   let signedvote = unblind(m2,b) in
   sync 1.
   in(c2,y4).out(ch3,y4).
   in(ch4,m3).out(c1,m3).
   let (l,commitedvoter',signedvoter') = m3 in
   in(c2,y5).
   if y5 = true then
    in(c2,y6).
    if y6 = true then
     sync 2.
     in(c2,y7).out(ch5,y7)
```

Listing 3.3 – The voting process under control of the attacker

```
processA =
 in(ch1,m1).
 let (pubkeyv,sig,blindedvote) = m1 in
 if pubkeyv = pkv && checksign(sig,pkv) = blindedvote then
  out(ch2,sign(blindedvote,ska))
```

Listing 3.4 – The administrator process

```
processC =
 in(ch3,(m3,m4)).
 if checksign(m4,pka) = m3 then
  νl.out(ch4,(l,m3,m4)).
  in(ch5,(l',rand)).
  if l = l' then
   let voteV = open(m3,rand) in
   out(res,voteV)
```

Listing 3.5 – The collector process

As labeled bisimilarity is closed under the application of contexts, it is sufficient to show that

$$\texttt{processV}\,\{^{skva}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{a}/_{v}\}\,|\texttt{processV}\,\{^{skvb}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{b}/_{v}\}$$
$$\approx_l$$
$$\texttt{processV}\,\{^{skva}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{b}/_{v}\}\,|\texttt{processV}\,\{^{skvb}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{a}/_{v}\}$$

This is different from the model and proof used in [DKR09], and more similar to the proofs by Kremer and Ryan [KR05] or Smyth [Smy11]: all keys are modeled as free names, and hence not secret. Note also that we consider only two honest voters and one voter under control of the attacker. However, as labeled bisimilarity is closed under the application of contexts, this immediately generalizes to an arbitrary number of honest or corrupted voters as we do not need any secret channel or key.

The intuition is the following: thanks to the blinding nobody can link the ballot, which is sent to the administrator, to the commitment, published later on over the anonymous channel.

We call the left hand side process P and the right hand side process Q. Note that we do not consider abstention of the honest voters here, hence the test `v=abstain` is always false. We write V_A for $\texttt{processV}\,\{^{skva}/_{skv},{}^{pk(ska)}/_{pka}\}$ and V_B for $\texttt{processV}\,\{^{skvb}/_{skv},{}^{pk(ska)}/_{pka}\}$. We now discuss the possible transitions. The honest voters construct the blinded and committed votes and send them to the administrator. We have the following transitions:

$$\begin{aligned}
P \;&\xrightarrow{\texttt{out}(ch1,x_1)} \nu b_A.\nu r_A.(P_1|\\
&\{^{(pk(skva),sign(blind(commit(a,r_A),b_A),skva),blind(commit(a,r_A),b_A))}/_{x_1}\})\\
&\xrightarrow{\texttt{out}(ch1,x_2)} \nu b_B.\nu b_A.\nu r_A.\nu r_B.(P_2|\\
&\{^{(pk(skva),sign(blind(commit(a,r_A),b_A),skva),blind(commit(a,r_A),b_A))}/_{x_1}\}\,|\\
&\{^{(pk(skvb),sign(blind(commit(b,r_B),b_B),skvb),blind(commit(b,r_B),b_B))}/_{x_2}\})
\end{aligned}$$

Similarly

$$\begin{aligned}
Q \;&\xrightarrow{\texttt{out}(ch1,x_1)} \nu b_A.\nu r_A.(Q_1|\\
&\{^{(pk(skva),sign(blind(commit(b,r_A),b_A),skva),blind(commit(b,r_A),b_A))}/_{x_1}\})\\
&\xrightarrow{\texttt{out}(ch1,x_2)} \nu b_B.\nu b_A.\nu r_A.\nu r_B.(Q_2|\\
&\{^{(pk(skva),sign(blind(commit(b,r_A),b_A),skva),blind(commit(b,r_A),b_A))}/_{x_1}\}\,|\\
&\{^{(pk(skvb),sign(blind(commit(a,r_B),b_B),skvb),blind(commit(a,r_B),b_B))}/_{x_2}\})
\end{aligned}$$

Until this point, it is easy to see that the frames are statically equivalent: both voters sent a signed commitment, but because of the blinding they are indistinguishable. Note that the messages can appear in the inverse order, however the frames remain statically equivalent. The next step depends on the context. If

the attacker returns correctly signed votes to the two voters, the processes can synchronize and go on. Otherwise at least one of them blocks and they are unable to synchronize.

If they are able to synchronize, they will output their unblinded vote to the collector. From this point on, the voters swap their roles, i.e. $V_B\{^a/_v\}$ simulates the behavior of $V_A\{^a/_v\}$. We obtain the following frames:

$$\begin{aligned}\phi_l = \nu b_B.\nu b_A.\nu r_A.\nu r_B.(&\\ \{(pk(skva),sign(blind(commit(a,r_A),b_A),skva),blind(commit(a,r_A),b_A))/_{x_1}\} \mid&\\ \{(pk(skvb),sign(blind(commit(b,r_B),b_B),skvb),blind(commit(b,r_B),b_B))/_{x_2}\} \mid&\\ \{(commit(a,r_A),sign(commit(a,r_A),ska))/_{x_3}\} \mid&\\ \{(commit(b,r_B),sign(commit(b,r_B),ska))/_{x_4}\})&\end{aligned}$$

$$\begin{aligned}\phi_r = \nu b_B.\nu b_A.\nu r_A.\nu r_B.(&\\ \{(pk(skva),sign(blind(commit(b,r_A),b_A),skva),blind(commit(b,r_A),b_A))/_{x_1}\} \mid&\\ \{(pk(skvb),sign(blind(commit(a,r_B),b_B),skvb),blind(commit(a,r_B),b_B))/_{x_2}\} \mid&\\ \{(commit(b,r_A),sign(commit(b,r_A),ska))/_{x_3}\} \mid&\\ \{(commit(a,r_B),sign(commit(a,r_B),ska))/_{x_4}\})&\end{aligned}$$

These frames are again statically equivalent, as the blinding breaks any link of the commitments to the first messages. Then it depends again on the input of the attacker. If either of the voters gets a wrong input, they cannot synchronize. If both voters receive a correct input, they reveal their random values with the corresponding l. This yields the following frames:

$$\begin{aligned}\phi'_l = \nu b_B.\nu b_A.\nu r_A.\nu r_B.\nu l_A.\nu l_B.(&\\ \{(pk(skva),sign(blind(commit(a,r_A),b_A),skva),blind(commit(a,r_A),b_A))/_{x_1}\} \mid&\\ \{(pk(skvb),sign(blind(commit(b,r_B),b_B),skvb),blind(commit(b,r_B),b_B))/_{x_2}\} \mid&\\ \{(commit(a,r_A),sign(commit(a,r_A),ska))/_{x_3}\} \mid&\\ \{(commit(b,r_B),sign(commit(b,r_B),ska))/_{x_4}\} \mid&\\ \{(l_A,r_A)/_{x_5}\} \mid \{(l_B,r_B)/_{x_6}\})&\end{aligned}$$

$$\begin{aligned}\phi'_r = \nu b_B.\nu b_A.\nu r_A.\nu r_B.\nu l_A.\nu l_B.(&\\ \{(pk(skva),sign(blind(commit(b,r_A),b_A),skva),blind(commit(b,r_A),b_A))/_{x_1}\} \mid&\\ \{(pk(skvb),sign(blind(commit(a,r_B),b_B),skvb),blind(commit(a,r_B),b_B))/_{x_2}\} \mid&\\ \{(commit(b,r_A),sign(commit(b,r_A),ska))/_{x_3}\} \mid&\\ \{(commit(a,r_B),sign(commit(a,r_B),ska))/_{x_4}\} \mid&\\ \{(l_A,r_A)/_{x_5}\} \mid \{(l_B,r_B)/_{x_6}\})&\end{aligned}$$

These frames are statically equivalent, giving us the desired result. □

This result can also be obtained automatically using ProSwapper[8] and ProVerif. The code is available on our website [Dre13].

Moreover note that using the recently developed tool AKISS, a similar proof of privacy [CCK12] for this protocol was made. The proof is fully automatic, however AKISS proves trace equivalence instead of labeled bismilarity, and uses a simpler process calculus instead of the Applied π-Calculus.

Note that FOO is not secure against Forced-Abstention-Attacks, as the voters send their identity (here modeled by their public key) to the administrator over a public channel. If we consider a modified version with a trusted administrator and a private channel between the voter and the administrator, FOO also ensures security against forced abstention attacks: $SwVP^{I,FA}$.

Theorem 20 *FOO with trusted administrator and a private channel between the voter and the administrator respects* $SwVP^{I,FA}$.

Proof As we proved above, FOO ensures Privacy for the "PO" case. Hence we only need to consider the case of abstention, i.e. show that

$$\begin{gathered}\boldsymbol{\nu ch1.\nu ch2.}(\texttt{processV}\,\{^{skva}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{abstain}/_{v}\}\mid\\ \texttt{processV}\,\{^{skvb}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{b}/_{v}\}\mid\texttt{processVc1c2}\,\{^{skvc}/_{skv},{}^{pk(ska)}/_{pka}\}\mid\\ \mathbf{processA}\,\{^{\boldsymbol{pk(skva)}}/_{\boldsymbol{pkv}}\}\mid\mathbf{processA}\,\{^{\boldsymbol{pk(skvb)}}/_{\boldsymbol{pkv}}\}\mid\mathbf{processA}\,\{^{\boldsymbol{pk(skvc)}}/_{\boldsymbol{pkv}}\})\\ \approx_l\\ \boldsymbol{\nu ch1.\nu ch2.}(\texttt{processV}\,\{^{skva}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{b}/_{v}\}\mid\\ \texttt{processV}\,\{^{skvb}/_{skv},{}^{pk(ska)}/_{pka}\}\,\{^{abstain}/_{v}\}\mid\texttt{processVc1c2}\,\{^{skvc}/_{skv},{}^{pk(ska)}/_{pka}\}\mid\\ \mathbf{processA}\,\{^{\boldsymbol{pk(skva)}}/_{\boldsymbol{pkv}}\}\mid\mathbf{processA}\,\{^{\boldsymbol{pk(skvb)}}/_{\boldsymbol{pkv}}\}\mid\mathbf{processA}\,\{^{\boldsymbol{pk(skvc)}}/_{\boldsymbol{pkv}}\})\end{gathered}$$

holds. The differences to the above proof are marked in **bold**: the channels `ch1` and `ch2` are now restricted, and we include the administrators as they are now trusted. Note that this proof is technically only valid for two honest voters and one voter under control of the attacker. Nevertheless a similar proof can be made for other numbers of voters.

The intuition is that since the administrator is honest and the registration is done over a private channel, the intruder only sees the anonymous commitments and can thus not determine which voters participated.

We call the left hand side process P and the right hand side process Q, and write V_A for $\texttt{processV}\,\{^{skva}/_{skv},{}^{pk(ska)}/_{pka}\}$, V_B for $\texttt{processV}\,\{^{skvb}/_{skv},{}^{pk(ska)}/_{pka}\}$ and $V_C^{c_1,c_2}$ for $\texttt{processVc1c2}\,\{^{skvc}/_{skv},{}^{pk(ska)}/_{pka}\}$. The beginning is similar: the non-abstaining honest voter construct the blinded and committed vote and sends them to the administrator through an internal reduction, due to the private

[8]Note that although – to the best of our knowledge – there exists no formal soundness proof for the transformations performed by ProSwapper [Smy11], in this case the result is effectively confirmed by our manual proof.

channel. The administrator will accept the ballot, sign it and send it back to the voter, again triggering an internal reduction. The voter checks the signature and waits for the synchronization. Until now, only internal reductions have taken place and the frames are empty. Then it depends on the context:

— If the corrupted voter is told to abstain, the bidder can synchronize, and publishes his ballot. We obtain the following frames:

$$\phi_l = \nu r_B.(\{^{(commit(b,r_B),sign(commit(b,r_B),ska))}/_{x_1}\})$$

and

$$\phi_r = \nu r_A.(\{^{(commit(b,r_A),sign(commit(b,r_A),ska))}/_{x_1}\})$$

Then it depends again on the input of the attacker. If the honest voter gets a wrong input, he blocks. If he receives receives a correct input he synchronizes and reveals his random values with the corresponding l. This yields the following frames:

$$\phi'_l = \nu r_B.\nu l_B.(\{^{(commit(b,r_B),sign(commit(b,r_B),ska))}/_{x_1}\} \mid \{^{(l_B,r_B)}/_{x_2}\})$$

and

$$\phi'_r = \nu r_A.\nu l_A.(\{^{(commit(b,r_A),sign(commit(b,r_A),ska))}/_{x_1}\} \mid \{^{(l_A,r_A)}/_{x_{11}}\})$$

These frames are statically equivalent as the random values are indistinguishable, which gives us the desired result.

— If the context tells the corrupted voter to vote, he forwards the given message to the administrator (over a private channel, hence an internal reduction). If the message was correct, the administrator signs and returns the ballot (again over a private channel). This message is forwarded to the context. In any other case, the process will simply block. We have the following transitions:

$$P \xrightarrow{\mathtt{in}(c2,false)} \xrightarrow{\mathtt{in}(c2,y2)} \rightarrow^* \xrightarrow{\mathtt{out}(c1,x_1)} P_1 | \{^{m2}/_{x_1}\}$$

Similarly

$$Q \xrightarrow{\mathtt{in}(c2,false)} \xrightarrow{\mathtt{in}(c2,y2)} \rightarrow^* \xrightarrow{\mathtt{out}(c1,x_1)} Q_1 | \{^{m2}/_{x_1}\}$$

These frames are identical and hence statically equivalent. If the context tells the voter to go on, the processes can synchronize, and the honest voter publishes his unblinded ballot. The corrupted bidder publishes the message he is given. This yields the following frames:

$$\phi''_l = \nu r_B.(\{^{m2}/_{x_1}\} \mid \{^{(commit(b,r_B),sign(commit(b,r_B),ska))}/_{x_2}\} \mid \{^{y4}/_{x_3}\} \mid \{^{m3}/_{x_4}\})$$

and

$$\phi''_r = \nu r_A.(\{^{m2}/_{x_1}\} \mid \{^{(commit(b,r_A),sign(commit(b,r_A),ska))}/_{x_2}\} \mid \{^{y4}/_{x_3}\} \mid \{^{m3}/_{x_4}\})$$

Then it depends again on the input of the attacker. If the honest voters gets a wrong input or the attacker decides to block $V_C^{c_1,c_2}$, the voters cannot synchronize. If the honest voters receives a correct input and if he can synchronize with $V_C^{c_1,c_2}$, he reveals his random values with the corresponding l. This yields the following frames:

$$\phi'''_l = \nu r_B.\nu l_B.(\{^{m2}/_{x_1}\} \mid \{^{(commit(b,r_B),sign(commit(b,r_B),ska))}/_{x_2}\} \mid \{^{y4}/_{x_3}\} \mid \{^{m3}/_{x_4}\} \mid \{^{(l_B,r_B)}/_{x_5}\} \mid \{^{y7}/_{x_6}\})$$

and

$$\phi'''_r = \nu r_A.\nu l_A.(\{^{m2}/_{x_1}\} \mid \{^{(commit(b,r_A),sign(commit(b,r_A),ska))}/_{x_2}\} \mid \{^{y4}/_{x_3}\} \mid \{^{m3}/_{x_4}\} \mid \{^{(l_A,r_A)}/_{x_5}\} \mid \{^{y7}/_{x_6}\})$$

These frames are statically equivalent as they only differ in the restricted random values, which gives us the desired result. □

§ 3.3.5.2. Protocol by Okamoto. The protocol by Okamoto [Oka96] is similar to the protocol by Fujioka et al. discussed above, but it uses trap-door commitments to achieve Receipt-Freeness. It is however not Coercion-Resistant as the coercer can force the voter to use a specially prepared commitment [DKR09].

Protocol Description. The main differences to FOO are the use of trap-door commitments and the existence of timeliness member to open the commitments. The first phase – during which the voter obtains a signature on his commitment – follows exactly the same protocol as FOO, except that this time ξ is a trapdoor-commitment. In the second phase the vote is submitted:

— Voter V_i sends the signed trap-door commitment to the collector C through an anonymous channel.

— The collector checks the administrators signature and enters (x_i, y_i) into a list.

— The voter sends (v_i, r_i, x_i) to the timeliness member through an untappable anonymous channel

When all ballots are cast and/or when the deadline is over, the counting phase begins:

— The collector publishes the list of correct ballots.

— V_i verifies that his commitment appears on the list.

— The timeliness member publishes a randomly shuffled list of votes v_i and a

zero-knowledge proof that he knows a permutation π for which $x_{\pi(i)} = \xi(v_i, r_i)$.

Model in the Applied π-Calculus. We use following equational theory:

$$\begin{aligned}
\texttt{checksign}(\texttt{sign}(m, sk), \texttt{pk}(sk)) &= m \\
\texttt{unblind}(\texttt{blind}(m, r), r) &= m \\
\texttt{unblind}(\texttt{sign}(\texttt{blind}(m, r), sk), r) &= \texttt{sign}(m, sk) \\
\texttt{open}(\texttt{tdcommit}(m, r, td), r) &= m \\
\texttt{tdcommit}(m_2, f(m_1, r, td, m_2), td) &= \texttt{tdcommit}(m_1, r, td)
\end{aligned}$$

The forth equation models the creation of a trap-door commitment to m using a random value r and a trap-door td, whereas the last equation permits the construction of another random value to open a commitment differently. This requires knowledge of the trap-door td and the initial random value r.

We use the following processes for our proof:

```
1  (* private keys *)
2  νska.νskva.νskvb.νskvc.
3  (* public keys *)
4  let (pka,pkva,pkvb,pkvc)
5   = (pk(ska),pk(skva),pk(skvb),pk(skvc)) in
6  (* private channels *)
7  νchT.
8  (* public key disclosure *)
9  out(ch,pka).out(ch,pkva).out(ch,pkvb).out(ch,pkvc).
10 (* administrators *)
11 (processA | processA | processA |
12  processC | processC | processC |
13  processT | processT | processT |
14 (* voters *)
15 (let skv = skva in let v = a in processV) |
16 (let skv = skvb in let v = b in processV) |
17 (let skv = skvc in let v = c in processV) )
```

Listing 3.6 – The main process

The *main process* (Listing 3.6) shows how the participation processes (three administrators, three collectors, three talliers - one for each voter - and three voters) are combined in parallel using a private channel.

The *voter* process (Listing 3.7 on the facing page) follows the nearly the same protocol as in the case of FOO, but he has to reveal the data necessary to open the commitment over a private channel to the timeliness member T. Note that in the case of abstention the voter also executes the second synchronization point, although normally it is executed by the collector. This is necessary to avoid a situation where the entire voting process blocks because of an abstaining voter.

```
processV =
 if v = abstain then
  sync 1.
  sync_n 2
 else
  νb.νr.νtd.
  let commitedvote = tdcommit(v,r,td) in
  let blindedvote = blind(commitedvote,b) in
  out(ch1,(pk(skv),sign(blindedvote,skv),blindedvote)).
  in(ch2,m2).
  let result = checksign(m2,pka) in
  if result = blindedvote then
   let signedvote = unblind(m2,b) in
   sync 1.
   out(ch3,(commitedvote,signedvote)).
   out(chT,(v,r,commitedvote))
```

Listing 3.7 – The voting process

```
processVc1c2 =
 in(c2,y1).
 if x1 = true then
  sync 1.
  sync_n 2
 else
  νb.out(c1,b).νr.out(c1,r).νtd.out(c1,td).
  let commitedvote = tdcommit(v,r,td) in
  let blindedvote = blind(commitedvote,b) in
  in(c2,y2).out(ch1,y2).
  in(ch2,m2).out(c1,m2).
  let result = checksign(m2,pka) in
  in(c2,y3).
  if y3 = true then
   let signedvote = unblind(m2,b) in
   sync 1.
   in(c2,y4).out(ch3,y4).
   in(c2,y5).out(chT,y5)
```

Listing 3.8 – The voting process under control of the attacker

```
processVchc =
 if v = abstain then
  sync 1.
  sync_n 2
 else
  νb.νr.νtd.
  out(chc,b).out(chc,r).out(chc,td).
  let commitedvote = tdcommit(a,r,td) in
  let blindedvote = blind(commitedvote,b) in
  out(ch1,(pk(skv),sign(blindedvote,skv),blindedvote)).
  out(chc,(pk(skv),sign(blindedvote,skv),blindedvote)).
  in(ch2,m2).
  let result = checksign(m2,pka) in
  if result = blindedvote then
   let signedvote = unblind(m2,b) in
   sync 1.
   out(ch3,(commitedvote,signedvote)).
   out(chc,(commitedvote,signedvote)).
   out(chT,(a,r,commitedvote)).
   out(chc,(a,r,commitedvote))
```

Listing 3.9 – The process trying to create a receipt

A *voter under control of the attacker* is described by Listing 3.8 on the previous page. This process can be obtained by calculating $\texttt{processV}^{c_1,c_2}$ as defined in Definition 19 on page 59.

A *voter trying to create a receipt* by revealing all his secret data is modeled by the process in Listing 3.9. This process can be obtained by calculating $\texttt{processV}^{chc}$ as defined in Definition 18 on page 59.

The *administrator* (Listing 3.11 on the next page) checks the signature on the blinded commitment and signs it.

The *collector* (Listing 3.12 on the facing page) verifies the signature on incoming commitments using the administrator's public key. If the signature is correct, he publishes the commitment on a public channel.

The *timeliness member* (Listing 3.13 on page 82) receives the vote, the corresponding commitment and the randomness over a private channel. He verifies the correctness of the data and then publishes the vote. Note that the synchronization point $\texttt{sync}_n$ 2 can also be enabled by an abstaining bidder (cf. Process 3.7 on the preceding page) to allow the publication of the submitted votes. As explained above, although an instance with n voters contains $2n$ instances of $\texttt{sync}_n$ 2, any n of them are sufficient to enable the continuation.

Analysis. We have the following result:

```
processV' =
 if a = abstain then
  νb.νr.νtd.
  let commitedvote = tdcommit(b,r,td) in
  let blindedvote = blind(commitedvote,b) in
  out(ch1,(pk(skv),sign(blindedvote,skv),blindedvote)).
  in(ch2,m2).
  let result = checksign(m2,pka) in
  if result = blindedvote then
   let signedvote = unblind(m2,b) in
   sync 1.
   out(ch3,(commitedvote,signedvote)).
   out(chT,(b,r,commitedvote)).
 else
  νb.νr.νtd.
  out(chc,b).out(chc,f(b,r,td,a)).out(chc,td).
  let commitedvote = tdcommit(b,r,td) in
  let blindedvote = blind(commitedvote,b) in
  out(ch1,(pk(skv),sign(blindedvote,skv),blindedvote)).
  out(chc,(pk(skv),sign(blindedvote,skv),blindedvote)).
  in(ch2,m2).
  let result = checksign(m2,pka) in
  if result = blindedvote then
   let signedvote = unblind(m2,b) in
   sync 1.
   out(ch3,(commitedvote,signedvote)).
   out(chc,(commitedvote,signedvote)).
   out(chT,(b,r,commitedvote)).
   out(chc,(a,f(b,r,td,a),commitedvote))
```

Listing 3.10 – The process V'

```
processA =
 in(ch1,m1).
 let (pubkeyv,sig,blindedvote) = m1 in
 if pubkeyv = pkv && checksign(sig,pkv) = blindedvote then
  out(ch2,sign(blindedvote,ska))
```

Listing 3.11 – The administrator process

```
processC =
 sync 1.
 in(ch3,(m3,m4)).
 if checksign(m4,pka) = m3 then
  sync_n 2.
  out(ch,(m3,m4))
```

Listing 3.12 – The collector process

```
processT =
 sync 1.
 (* receiving the commitment *)
 in(chT,(vt,rt,xt)).
 sync 2.
 if open(xt,rt) = vt then
  out(res,vt)
```

Listing 3.13 – The timeliness process

Theorem 21 *The protocol by Okamoto respects* $SwRF^{I,PO}$.

Proof We need to show that there exists a closed plain process V' such that

$$V'^{\setminus out(chc,\cdot)} \approx_l \texttt{processV}\,\{^{skva}/_{skv}, {}^{pk(ska)}/_{pka}\}\,\{^{b}/_{v}\} \tag{3.8}$$

and

$$\begin{aligned} &\nu chT.((\texttt{processV}\,\{^{skva}/_{skv}, {}^{pk(ska)}/_{pka}\}\,\{^{a}/_{v}\})^{chc}\mid \\ &\quad \texttt{processV}\,\{^{skvb}/_{skv}, {}^{pk(ska)}/_{pka}\}\,\{^{b}/_{v}\}\mid \\ &\quad \texttt{processVc1c2}\,\{^{skvc}/_{skv}, {}^{pk(ska)}/_{pka}\}\mid \\ &\quad\quad \texttt{processT}\mid\texttt{processT}\mid\texttt{processT}) \\ &\qquad\qquad \approx_l \\ &\nu chT.(V'\mid\texttt{processV}\,\{^{skvb}/_{skv}, {}^{pk(ska)}/_{pka}\}\,\{^{a}/_{v}\}\mid \\ &\quad \texttt{processVc1c2}\,\{^{skvc}/_{skv}, {}^{pk(ska)}/_{pka}\}\mid \\ &\quad\quad \texttt{processT}\mid\texttt{processT}\mid\texttt{processT}) \end{aligned} \tag{3.9}$$

As in the case of the protocol by Fujioka et al., we only consider three voters, but the proof can be generalized for more honest voters. In particular we show in Section 3.4.4 that the protocol is modular, which directly implies that privacy also holds for any number of honest voters: we can compose the above instance with another instance to obtain an instance with an arbitrary number of voters, and the bisimilarity still holds. For more details, see Section 3.4.4.

Again, we write V_A for $\texttt{processV}\,\{^{skva}/_{skv}, {}^{pk(ska)}/_{pka}\}$, V_B for $\texttt{processV}\,\{^{skvb}/_{skv}, {}^{pk(ska)}/_{pka}\}$ and $V_C^{c_1,c_2}$ for $\texttt{processVc1c2}\,\{^{skvc}/_{skv}, {}^{pk(ska)}/_{pka}\}$.

Consider the process $V' = \texttt{processV}'\,\{^{skva}/_{skv}, {}^{pk(ska)}/_{pka}\}$ that fakes a receipt. We can see that the first equivalence (3.8) holds by removing all $\texttt{out}(\texttt{chc}, \_)$ from $\texttt{processV}'$ as given in Listing 3.10 on the previous page: Independent of `a`, the resulting process is apparently equal to the original voting process $\texttt{processV}\,\{^{skva}/_{skv}\}\,\{^{b}/_{v}\}$.

The second equivalence (3.9) is more difficult to prove. The beginning is similar to the proof of privacy for FOO. We denote the left hand side process P and the right hand side process Q. The honest voters construct the blinded and committed votes and send them to the administrator. During this process, V_A

outputs his secret values and inputs, where V' fakes these values. We have the following transitions:

$$
\begin{aligned}
P \xrightarrow{\mathtt{out}(chc,x_1)} \xrightarrow{\nu x_2.\mathtt{out}(chc,x_2)} \xrightarrow{\nu x_3.\mathtt{out}(chc,x_3)} \nu b_A.\nu r_A.\nu td_A.(P_1 \mid \{b_A/x_1\} \\
\mid \{r_A/x_2\} \mid \{td_A/x_3\}) \\
\xrightarrow{\nu x_4.\mathtt{out}(ch1,x_4)} \xrightarrow{\nu x_5.\mathtt{out}(chc,x_5)} \nu b_A.\nu r_A.\nu td_A.(P_2 \mid \{b_A/x_1\} \mid \{r_A/x_2\} \\
\mid \{td_A/x_3\} \\
\mid \{(pk(skva),sign(blind(tdcommit(a,r_A,td_A),b_A),skva),blind(tdcommit(a,r_A,td_A),b_A))/x_4\} \\
\mid \{x_4/x_5\}) \\
\xrightarrow{\nu x_6.\mathtt{out}(ch1,x_6)} \nu b_A.\nu r_A.\nu td_A.\nu b_B.\nu r_B.\nu td_B.(P_3 \mid \{b_A/x_1\} \\
\mid \{r_A/x_2\} \mid \{td_A/x_3\} \\
\mid \{(pk(skva),sign(blind(tdcommit(a,r_A,td_A),b_A),skva),blind(tdcommit(a,r_A,td_A),b_A))/x_4\} \\
\mid \{x_4/x_5\} \\
\mid \{(pk(skva),sign(blind(tdcommit(b,r_B,td_B),b_B),skvb),blind(tdcommit(b,r_B,td_B),b_B))/x_6\})
\end{aligned}
$$

Similarly

$$
\begin{aligned}
Q \xrightarrow{\nu x_1.\mathtt{out}(chc,x_1)} \xrightarrow{\nu x_2.\mathtt{out}(chc,x_2)} \xrightarrow{\nu x_3.\mathtt{out}(chc,x_3)} \nu b_A.\nu r_A.\nu td_A.(Q_1 \\
\mid \{b_A/x_1\} \mid \{f(a,r_A,td_A,c)/x_2\} \mid \{td_A/x_3\}) \\
\xrightarrow{\nu x_4.\mathtt{out}(ch1,x_4)} \xrightarrow{\nu x_5.\mathtt{out}(chc,x_5)} \nu b_A.\nu r_A.\nu td_A.(Q_2 \mid \{b_A/x_1\} \\
\mid \{f(b,r_A,td_A,a)/x_2\} \mid \{td_A/x_3\} \\
\mid \{(pk(skva),sign(blind(tdcommit(b,r_A,td_A),b_A),skva),blind(tdcommit(b,r_A,td_A),b_A))/x_4\} \\
\mid \{x_4/x_5\}) \\
\xrightarrow{\nu x_6.\mathtt{out}(ch1,x_6)} \nu b_A.\nu r_A.\nu td_A.\nu b_B.\nu r_B.\nu td_B.(Q_3 \mid \{b_A/x_1\} \\
\mid \{f(a,r_A,td_A,c)/x_2\} \mid \{td_A/x_3\} \\
\mid \{(pk(skva),sign(blind(tdcommit(b,r_A,td_A),b_A),skva),blind(tdcommit(b,r_A,td_A),b_A))/x_4\} \\
\mid \{x_4/x_5\} \\
\mid \{(pk(skva),sign(blind(tdcommit(a,r_B,td_B),b_B),skvb),blind(tdcommit(a,r_B,td_B),b_B))/x_6\})
\end{aligned}
$$

We can argue that at this point the obtained frames are statically equivalent. In particular the attacker obtains in both cases

$$\mathtt{open}(\mathtt{unblind}(\mathtt{checksign}(\mathtt{proj}_2(x_4), \mathtt{pk}(skva)), x_1), x_2) = a$$

when he tries to open the commitment due to the trap-door and the faked randomness.

The next steps depend on the context:

— If the attacker tells the corrupted voter to vote, he will forward the given

ballot to the administrator. We have the following transitions:

$$P_3 \xrightarrow{\mathtt{in}(c2,false)} \xrightarrow{\mathtt{in}(c2,y2)} \xrightarrow{\nu x_7.\mathtt{out}(ch1,x_7)} \nu\tilde{x}.(P_4 |\, \{^{b_A}/_{x_1}\} \mid \{^{r_A}/_{x_2}\} \mid \{^{td_A}/_{x_3}\} \mid \{^{(pk(skva),sign(blind(tdcommit(a,r_A,td_A),b_A),skva),blind(tdcommit(a,r_A,td_A),b_A))}/_{x_4}\} \mid \{^{x_4}/_{x_5}\} \mid \{^{(pk(skva),sign(blind(tdcommit(b,r_B,td_B),b_B),skvb),blind(tdcommit(b,r_B,td_B),b_B))}/_{x_6}\} \mid \{^{y2}/_{x_7}\})$$

where $\tilde{x} = \{b_A, r_A, td_A, b_B, r_B, td_B\}$. Similarly

$$Q_3 \xrightarrow{\mathtt{in}(c2,false)} \xrightarrow{\mathtt{in}(c2,y2)} \xrightarrow{\nu x_7.\mathtt{out}(ch1,x_7)} \nu\tilde{x}.(Q_4 |\, \{^{b_A}/_{x_1}\} \mid \{^{f(a,r_A,td_A,c)}/_{x_2}\} \mid \{^{td_A}/_{x_3}\} \mid \{^{(pk(skva),sign(blind(tdcommit(b,r_A,td_A),b_A),skva),blind(tdcommit(b,r_A,td_A),b_A))}/_{x_4}\} \mid \{^{x_4}/_{x_5}\} \mid \{^{(pk(skva),sign(blind(tdcommit(a,r_B,td_B),b_B),skvb),blind(tdcommit(a,r_B,td_B),b_B))}/_{x_6}\} \mid \{^{y2}/_{x_7}\})$$

If the context then returns correctly signed votes to the two honest voters and tells $V_C^{c_1,c_2}$ to go on (i.e. sends a message containing `true`), the processes can synchronize and go on. Otherwise at least one of them blocks and they are unable to synchronize.
If they are able to synchronize, they output their votes to the collector ($V_C^{c_1,c_2}$ sends any message the attacker gives him) and send the secret values to the timeliness member over a private channel. We obtain the following frames:

$$\phi_l = \nu\tilde{x}.(\{^{b_A}/_{x_1}\} \mid \{^{r_A}/_{x_2}\} \mid \{^{td_A}/_{x_3}\} \mid \{^{(pk(skva),sign(blind(tdcommit(a,r_A,td_A),b_A),skva),blind(tdcommit(a,r_A,td_A),b_A))}/_{x_4}\} \mid \{^{x_4}/_{x_5}\} \mid \{^{(pk(skva),sign(blind(tdcommit(b,r_B,td_B),b_B),skvb),blind(tdcommit(b,r_B,td_B),b_B))}/_{x_6}\} \mid \{^{y2}/_{x_7}\} \mid \{^{(tdcommit(a,r_A,td_A),sign(tdcommit(a,r_A,td_A),ska))}/_{x_8}\} \mid \{^{(tdcommit(b,r_B,td_B),sign(tdcommit(b,r_B,td_B),ska))}/_{x_9}\} \mid \{^{(a,r_A,tdcommit(a,r_A,td_A))}/_{x_{11}}\})$$

and

$$\phi_r = \nu\tilde{x}.(\{b_A/x_1\} \mid \{f(b,r_A,td_A,a)/x_2\} \mid \{td_A/x_3\} \mid \{(pk(skva),sign(blind(tdcommit(b,r_A,td_A),b_A),skva),blind(tdcommit(b,r_A,td_A),b_A))/x_4\} \mid \{x_4/x_5\} \mid \{(pk(skva),sign(blind(tdcommit(a,r_B,td_B),b_B),skvb),blind(tdcommit(a,r_B,td_B),b_B))/x_6\} \mid \{y2/x_7\} \mid \{(tdcommit(b,r_A,td_A),sign(tdcommit(b,r_A,td_A),ska))/x_8\} \mid \{(tdcommit(a,r_B,td_B),sign(tdcommit(a,r_B,td_B),ska))/x_9\} \mid \{(a,f(b,r_A,td_A,a),tdcommit(b,r_A,td_A))/x_{10}\})$$

These frames are statically equivalent. If the attacker sent a correct input to the third timeliness process, T will put out the corresponding vote. Note that if the attacker copies V_A's vote using the possibly faked credentials, T will always output a.
More generally, the attacker cannot use T to obtain frames that are not statically equivalent as this would imply that this difference could also be made on the frames before by opening a commitment - something the attacker could do by himself. Intuitively having access to the "Timeliness" oracle does not help the attacker, as the oracle performs only operations the attacker could have done by himself.

— Similarly, if the attacker tells the corrupted voter to abstain and if the context then returns correctly signed votes to the two honest voters, the processes can synchronize and go on. Otherwise at least one of them blocks and they are unable to synchronize.
If they are able to synchronize, they output their votes to the collector and send the secret values to the timeliness member over a private channel. The timeliness member will open the commitments, synchronize and publish the votes. We obtain the following frames:

$$\phi_l = \nu\tilde{x}.(\{b_A/x_1\} \mid \{r_A/x_2\} \mid \{td_A/x_3\} \mid \{(pk(skva),sign(blind(tdcommit(a,r_A,td_A),b_A),skva),blind(tdcommit(a,r_A,td_A),b_A))/x_4\} \mid \{x_4/x_5\} \mid \{(pk(skva),sign(blind(tdcommit(b,r_B,td_B),b_B),skvb),blind(tdcommit(b,r_B,td_B),b_B))/x_6\} \mid \{(tdcommit(a,r_A,td_A),sign(tdcommit(a,r_A,td_A),ska))/x_7\} \mid \{(tdcommit(b,r_B,td_B),sign(tdcommit(b,r_B,td_B),ska))/x_8\} \mid \{(a,r_A,tdcommit(a,r_A,td_A))/x_9\} \mid \{a/x_{10}\} \mid \{b/x_{11}\})$$

and

$$\begin{aligned}\phi_r = \nu\tilde{x}.(&\{^{b_A}/_{x_1}\} \mid \{^{f(b,r_A,td_A,a)}/_{x_2}\} \mid \{^{td_A}/_{x_3}\} \mid \\ &\{^{(pk(skva),sign(blind(tdcommit(b,r_A,td_A),b_A),skva),blind(tdcommit(b,r_A,td_A),b_A))}/_{x_4}\} \mid \\ &\{^{x_4}/_{x_5}\} \mid \\ &\{^{(pk(skva),sign(blind(tdcommit(a,r_B,td_B),b_B),skvb),blind(tdcommit(a,r_B,td_B),b_B))}/_{x_6}\} \mid \\ &\{^{(tdcommit(b,r_A,td_A),sign(tdcommit(b,r_A,td_A),ska))}/_{x_7}\} \mid \\ &\{^{(tdcommit(a,r_B,td_B),sign(tdcommit(a,r_B,td_B),ska))}/_{x_8}\} \mid \\ &\{^{(a,f(b,r_A,td_A,a),tdcommit(b,r_A,td_A))}/_{x_9}\} \mid \\ &\{^{a}/_{x_{10}}\} \mid \{^{b}/_{x_{11}}\})\end{aligned}$$

Again, these frames are statically equivalent. □

Unfortunately ProVerif cannot prove this automatically as the equational theory is too complex. This is due to the trapdoor-commitment which yields a non-confluent equational theory. However, we can show that the final frames ϕ_l and ϕ_r from above are statically equivalent using KISS [CDK12]. The code is available online [Dre13].

As for FOO, we can show that the protocol also ensures Security against Forced-Abstention-Attacks if the administrator is to be trusted and there is a private channel between the administrator and the voters.

Theorem 22 *The protocol by Okamoto with a trusted administrator and a private channel to the administrator respects $SwRF^{I,FA}$.*

Proof We have to show that there exists a closed plain process V' such that

$$V'^{\setminus out(chc,\cdot)} \approx_l \texttt{processV}\,\{^{skva}/_{skv}, {}^{pk(ska)}/_{pka}\}\,\{^{b}/_{v}\}$$

and

$$\begin{aligned}&\nu chT.\boldsymbol{\nu ch1.\nu ch2.}((\texttt{processV}\,\{^{skva}/_{skv}, {}^{pk(ska)}/_{pka}\}\,\{^{a}/_{v}\})^{chc} \mid \\ &\texttt{processV}\,\{^{skvb}/_{skv}, {}^{pk(ska)}/_{pka}\}\,\{^{b}/_{v}\} \mid \texttt{processVc1c2}\,\{^{skvc}/_{skv}, {}^{pk(ska)}/_{pka}\} \mid \\ &\texttt{processT}\mid\texttt{processT}\mid\texttt{processT}\mid \\ &\textbf{processA}\,\{^{\boldsymbol{pk(skva)}}/_{\boldsymbol{pkv}}\} \mid \textbf{processA}\,\{^{\boldsymbol{pk(skvb)}}/_{\boldsymbol{pkv}}\} \mid \textbf{processA}\,\{^{\boldsymbol{pk(skvc)}}/_{\boldsymbol{pkv}}\}) \\ &\approx_l \\ &\nu chT.\boldsymbol{\nu ch1.\nu ch2.}(V' \mid \texttt{processV}\,\{^{skvb}/_{skv}, {}^{pk(ska)}/_{pka}\}\,\{^{a}/_{v}\} \mid \\ &\texttt{processVc1c2}\,\{^{skvc}/_{skv}, {}^{pk(ska)}/_{pka}\} \mid \\ &\texttt{processT}\mid\texttt{processT}\mid\texttt{processT}\mid \\ &\textbf{processA}\,\{^{\boldsymbol{pk(skva)}}/_{\boldsymbol{pkv}}\} \mid \textbf{processA}\,\{^{\boldsymbol{pk(skvb)}}/_{\boldsymbol{pkv}}\} \mid \textbf{processA}\,\{^{\boldsymbol{pk(skvc)}}/_{\boldsymbol{pkv}}\})\end{aligned}$$

As above, our proof is technically only be correct for three voters, but it can be generalized for more voters.

As above, we write V_A for $\texttt{processV}\ \{^{skva}/_{skv}, ^{pk(ska)}/_{pka}\}$, V_B for $\texttt{processV}\ \{^{skvb}/_{skv}, ^{pk(ska)}/_{pka}\}$ and $V_C^{c_1,c_2}$ for $\texttt{processVc1c2}\ \{^{skvc}/_{skv}, ^{pk(ska)}/_{pka}\}$.

Consider $V' = \texttt{processV'}\ \{^{skva}/_{skv}, ^{pk(ska)}/_{pka}\}$. The first equivalence is easy to see by removing all $\texttt{out}(\texttt{chc}, \_)$ from $\texttt{processV'}$ (Listing 3.10 on page 81). Independent of $\texttt{a}$, the resulting process is apparently equal to the original voting process $\texttt{processV}\ \{^{skva}/_{skv}\}\ \{^{b}/_{v}\}$.

The second equivalence is more difficult to prove. As for FOO, it is sufficient to analyze the case of abstention, as the other case is a consequence of the above lemma. We denote the left hand side process P and the right hand side process Q. On the left hand side the honest voter under attack V_A abstains and hence does nothing, whereas the other voter V_B constructs the blinded and committed vote and send it to the administrator. On the right hand side, V' pretends to abstain, whereas V_B abstains in reality. Note that since the channels to the administrator are now private channels, the communication only yields internal reductions. The next steps depends on the context:

— If the context tells the corrupted voter to abstain, the honest voter can synchronize and forward his ballot to the timeliness member and to the collector. The timeliness member can synchronize too, and will publish the vote. We have the following transitions:

$$\begin{aligned} P \quad &\rightarrow^* \xrightarrow{\texttt{in}(c1,false)} \xrightarrow{\nu x_1.\texttt{out}(ch3,x_1)} \nu r_B.\nu td_B.(P_1 \mid \\ &\qquad \{^{(tdcommit(b,r_B,td_B),sign(tdcommit(b,r_B,td_B),ska))}/_{x_1}\}) \\ &\rightarrow^* \xrightarrow{\nu x_2.\texttt{out}(res,x_2)} \nu r_B.\nu td_B.(P_2 \mid \\ &\{^{(tdcommit(b,r_B,td_B),sign(tdcommit(b,r_B,td_B),ska))}/_{x_1}\} \mid \{^{b}/_{x_2}\}) \end{aligned}$$

Similarly

$$\begin{aligned} Q \quad &\rightarrow^* \xrightarrow{\texttt{in}(c1,false)} \xrightarrow{\nu x_1.\texttt{out}(ch3,x_1)} \nu r_A.\nu td_A.(Q_1 \mid \\ &\qquad \{^{(tdcommit(b,r_A,td_A),sign(tdcommit(b,r_A,td_A),ska))}/_{x_1}\}) \\ &\rightarrow^* \xrightarrow{\nu x_2.\texttt{out}(res,x_2)} \nu r_A.\nu td_A.(Q_2 \mid \\ &\{^{(tdcommit(b,r_A,td_A),sign(tdcommit(b,r_A,td_A),ska))}/_{x_1}\} \mid \{^{b}/_{x_2}\}) \end{aligned}$$

It is easy to see that both frames are statically equivalent.

— If the attacker tells the corrupted voter to vote, he will forward the given ballot to the administrator. If this ballot is correctly signed, the administrator will accept it, sign it and return it. If the context then tells $V_C^{c_1,c_2}$ to go on (i.e. sends a message containing $\texttt{true}$), the processes can synchronize and go on. Otherwise at least one of them blocks and they are unable to synchronize. If they are able to synchronize, they output their votes to the collector ($V_C^{c_1,c_2}$ sends any message the attacker gives him) and send the secret values to the

timeliness member over a private channel. We obtain the following frames:

$$\phi_l = \nu r_B.\nu td_B.(\{^{(sign(checksign(proj_2(y1),pubkvc),ska))}/_{x_1}\} \mid \{^{(tdcommit(b,r_B,td_B),sign(tdcommit(b,r_B,td_B),ska))}/_{x_2}\} \mid \{^{y4}/_{x_3}\})$$

and

$$\phi_r = \nu r_A.\nu td_A.(\{^{(sign(checksign(proj_2(y1),pubkvc),ska))}/_{x_1}\} \mid \{^{(tdcommit(b,r_A,td_A),sign(tdcommit(b,r_A,td_A),ska))}/_{x_2}\} \mid \{^{y4}/_{x_3}\})$$

These frames are statically equivalent. If the attacker sent a correct input to the third timeliness process, T will put out the corresponding votes. However, since he has no access to the values used to create the commitment used by the honest voter, the only possibility for him is to create his own one. In that case the timeliness member announces the vote of the corrupted voter plus one vote b.

As above, the attacker cannot use T to obtain frames that are not statically equivalent as this would imply that this difference could also be made on the frames before by opening a commitment - something the attacker could do by himself. Intuitively having access to the “Timeliness" oracle does not help the attacker, as the oracle performs only operations the attacker could have done by himself. □

§ 3.3.5.3. Bingo Voting. Bingo Voting was developed by Bohli, Müller-Quade and Röhrich [BMQR07] to achieve coercion-resistance as well as individual and universal verifiability by using a trusted random number generator (RNG), i.e. a tamper-proof random generator providing real random numbers only to the machine and the voter (in particular not to the adversary). In our hierarchy Bingo Voting ensures $SwCR^{I,PO}$ if the voting machine is to be trusted.

Protocol Description. The protocol is split into three phases: The pre-voting phase, the voting phase and the post-voting phase. In the pre-voting phase the voting machine generates for every candidate p_j ($j \in \{1,\ldots,l\}$, l is the number of candidates) k (k is the number of voters) random values $n_{i,j}$ ($i \in \{1,\ldots,k\}$, $j \in \{1,\ldots,l\}$). It commits to the $k \cdot l$ pairs $(n_{i,j}, p_j)$ and publishes the shuffled commitments.

In the voting phase, the voter enters the voting booth and selects the candidate he wants to vote for on the voting machine. The RNG generates a random number r which is transmitted to the voting machine and displayed to the voter. The voting machine chooses for each candidate, except for the voter's choice, a dummy vote. For the chosen candidate, the random value from the RNG is used and

the receipt is created. Finally the voter checks that the number displayed on the RNG corresponds to the entry of his candidate on the receipt.

In the post-voting phase, the voting machine announces the result, publishes all receipts, and opens the commitments of all unused dummy votes. The machine also generates non-interactive zero-knowledge proofs that each unopened commitment was actually used as a dummy vote in one of the receipts.

Model in the Applied π-Calculus. As we are only interested in privacy, we ignore the zero-knowledge proofs which are necessary to achieve verifiability. This yields a very simple equational theory:

$$\texttt{open}(\texttt{commit}(m, r), r) \quad = \quad m$$

We assume the voting machine to be honest, otherwise no privacy can be guaranteed as the vote is submitted in clear by the voter. To model the voting booth, we use private channels between the voting machine and the voter, between the voter and the RNG, and between the RNG and the voting machine. To achieve better readability we use the macros `for` and `parfor` (similar to [MR10]) where e.g. `for (i = 1 to 2) out(ch, i)` corresponds to `out(ch, 1).out(ch, 2)` and `parfor (i = 1 to 2) out(ch, i)` corresponds to `(out(ch, 1) | out(ch, 2))`. Additionally we use a function `choose` where

$$\texttt{choose}(p_i, p_j, x, y) = \begin{cases} x \text{ if } i = j \\ y \text{ otherwise} \end{cases}$$

Our model depends on two parameters: k (the number of voters) and l (the number of candidates).

The *main process* (Listing 3.14 on the next page) sets up the private channels and executes the participating processes (the voting machine, three voters, and three RNGs) in parallel.

The *RNG* (Listing 3.15 on the following page) generates a random number and sends it to the voting machine and the voter over private channels.

The *voter* (Listing 3.16 on page 91) sends his vote to the voting machine, receives the random number from the RNG and the receipt.

The $V^{c1,c2}$ *process* (Listing 3.17 on page 91) simulates a coerced voter. He follows the same protocol, but votes for the candidate the coercer tells him and forwards the receipt and the random number.

The *voting machine* (Listing 3.19 on page 93) generates the dummy votes and publishes the corresponding commitments. Then k sub-processes interact with the voters, i.e. create the receipts. After all voting has been done, they publish the result, the receipts and the unused dummies in random order. Note that

```
1  (* private channels *)
2  νprivChM1.νprivChM2.ν privChM3.
3  νprivChRM1.νprivChRM2.ν privChRM3.
4  νprivChR1.νprivChR2.ν privChR3.
5  (* voting machine *)
6  (processM |
7  (* RNGs *)
8  (let privChM = privChRM1 in
9    let privChV = privChR1 in processRNG) |
10 (let privChM = privChRM2 in
11   let privChV = privChR2 in processRNG) |
12 (let privChM = privChRM3 in
13   let privChV = privChR3 in processRNG) |
14 (* voters *)
15 (let privChM = privChM1 in
16   let privChRNG = privChR1 in
17    let v = p1 in processV) |
18 (let privChM = privChM2 in
19   let privChRNG = privChR2 in
20    let v = p2 in processV) |
21 (let privChM = privChM3 in
22   let privChRNG = privChR3 in
23    let v = p3 in processV))
```

Listing 3.14 – The main process

```
1 processRNG =
2  (* generate random number *)
3  νr.
4  (* output to voting machine *)
5  out(privChM,r).
6  (* output to voter *)
7  out(privChV,r)
```

Listing 3.15 – The random number generator (RNG)

```
processV =
 if v = abstain then
  sync id_i
 else
  (* voting *)
  out(privChM,v).
  (* receipt *)
  for(i = 1 to l)
   in(privChM,receipt_i)
  (* random value, to verify receipt *)
  in(privChRNG,r)
```

Listing 3.16 – The voting process

```
processVc1c2 =
 in(c1,x1).
 if x1 = true then
  sync id_i
 else
  (* voting *)
  in(c1,x2).
  out(privChM, x2).
  (* receipt *)
  for(i = 1 to l)
   in(privChM,receipt_i).
   out(c2,receipt_i)
  (* random value, to verify receipt *)
  in(privChRNG,r).
  out(c2,r)
```

Listing 3.17 – The process $V^{c1,c2}$

```
processV' =
 in(c1,x1).
 (* voting *)
 if x1 = true then
  out(privChM,b).
  (* random value, to verify receipt *)
  in(privChRNG,r).
  (* receipt *)
  for(i = 1 to l)
   in(privChM,receipt_i).
  (* output nothing since claiming to abstain *)
 else
  in(c1,x2).
  out(privChM,b).
  (* random value, to verify receipt *)
  in(privChRNG,r).
  (* receipt *)
  for(i = 1 to l)
   in(privChM,receipt_i).
   out(c2,receipt_i)
  (* output correct random value *)
  for(i = 1 to l)
   if(x2 = p_i) then out(c2,receipt_i)
```

Listing 3.18 – The process V'

```
1  processM' =
2   in(privChV,v).in(privChR,r).
3   for(j = 1 to l)
4    let receipt_j = choose(p_j,v,r,n_{i,j}) in
5     out(privChV,receipt_j);
6   sync_k 1.
7   (* output result *)
8   out(res,v)
9   (* output receipts *)
10  for(j = 1 to l)
11   out(chRec,receipt_j)
12  (* output unused dummy votes *)
13  parfor(j = 1 to l)
14   if v_i ≠p_j then out(chDum,((n_{i,j},p_j),commit((n_{i,j},p_j),r_{i,j}),r_{i,j}))
15
16 processM'' =
17  sync id_i.
18  sync_k 1.
19  (* output unused dummy votes *)
20  parfor(j = 1 to l)
21   out(chDum,((n_{i,j},p_j),commit((n_{i,j},p_j),r_{i,j}),r_{i,j}))
22
23 processM =
24  (* prepare dummy votes *)
25  for(i = 1 to k)
26   for(j = 1 to l)
27    νn_{i,j}.νr_{i,j}.
28  parfor(i = 1 to k)
29   parfor(j = 1 to l)
30    out(ch,commit((n_{i,j},p_j),r_{i,j}))
31   (* voting *)
32   let privChV = privChM_i in
33    let privChR = privChRM_i in (processM' | processM'')
```

Listing 3.19 – The voting machine

again any k out of the $2k$ synchronization points $\mathtt{sync}_k$ 1 (as there are two voting machine processes per voter each containing a $\mathtt{sync}_k$ 1 command) are sufficient for continuation.

Analysis. As announced above, Bingo Voting respects $SwCR^{I,FA}$.

Theorem 23 *Bingo Voting respects $SwCR^{I,FA}$, i.e. Coercion-Resistance against an inside attacker and security against forced abstention attacks.*

Proof To show that Bingo Voting ensures $SwCR^{I,FA}$, we show that for any $C = \nu c_1.\nu c_2.(\_|P)$ satisfying $\tilde{n} \cap fn(C) = \emptyset$ and

$$\begin{gathered}\nu\tilde{\mathtt{ch}}.(C\,[V_A^{c_1,c_2}]\,|V_B\,\{^b/_v\}\,|V_C^{c_1,c_2}|M_{k=3,l=2}|R_A|R_B|R_C)\\ \approx_l\\ \nu\tilde{\mathtt{ch}}.(V_A\,\{^a/_v\}^{chc}\,|V_B\,\{^b/_v\}\,|V_C^{c_1,c_2}|M_{k=3,l=2}|R_A|R_B|R_C)\end{gathered}$$

we have

$$C\,[V']^{\setminus out(chc,\cdot)} \approx_l V_A\,\{^b/_v\}$$

and

$$\begin{gathered}\nu\tilde{\mathtt{ch}}.(C\,[V_A^{c_1,c_2}]\,|V_B\,\{^b/_v\}\,|V_C^{c_3,c_4}|M_{k=3,l=2}|R_A|R_B|R_C)\\ \approx_l\\ \nu\tilde{\mathtt{ch}}.(C\,[V']\,|V_B\,\{^a/_v\}\,|V_C^{c_3,c_4}|M_{k=3,l=2}|R_A|R_B|R_C)\end{gathered}$$

where

$$\begin{aligned}\nu\tilde{\mathtt{ch}} &= \nu\mathtt{privChM}_1.\nu\mathtt{privChM}_2.\nu\mathtt{privChM}_3.\\ &\quad \nu\mathtt{privChRM}_1.\nu\mathtt{privChRM}_2.\nu\mathtt{privChRM}_3.\\ &\quad \nu\mathtt{privChR}_1.\nu\mathtt{privChR}_2.\nu\mathtt{privChR}_3\\ V_A &= \mathtt{processV}\,\{^{privChM_1}/_{privChM}, {}^{privChR_1}/_{privChRNG}\}\\ V_B &= \mathtt{processV}\,\{^{privChM_2}/_{privChM}, {}^{privChR_2}/_{privChRNG}\}\\ V_C &= \mathtt{processV}\,\{^{privChM_3}/_{privChM}, {}^{privChR_3}/_{privChRNG}\}\\ V' &= \mathtt{processV'}\,\{^{privChM_1}/_{privChM}, {}^{privChR_1}/_{privChRNG}\}\\ R_A &= \mathtt{processRNG}\,\{^{privChRM_1}/_{privChM}, {}^{privChR_1}/_{privChV}\}\\ R_B &= \mathtt{processRNG}\,\{^{privChRM_2}/_{privChM}, {}^{privChR_2}/_{privChV}\}\\ R_C &= \mathtt{processRNG}\,\{^{privChRM_3}/_{privChM}, {}^{privChR_3}/_{privChV}\}\\ M_{k=k',l=l'} &= \mathtt{processM}\,\{^{k'}/_k, {}^{l'}/_l\}\end{aligned}$$

This is technically only a valid proof for two honest voters and two candidates, however a similar proof can be done for an arbitrary number of voters and candidates. In Section 3.4.4 we also show that Bingo Voting is modular, which directly generalizes this proof to an arbitrary number of honest voters.

It is easy to see that $C\,[V']^{\setminus out(chc,\cdot)} \approx_l V_A\,\{^b/_v\}$ holds. If we ignore all inputs of V' on $\mathtt{c1}$ and all outputs on $\mathtt{c2}$, we obtain V_A as both cases (abstention or not) coincide.

We now have to show that the last equivalence holds. As before, we denote the left hand side P and the right hand side Q. For better readability we concentrate on the important steps. At the beginning the voting machine publishes all commitments; then the voters enter the voting booth and vote. We consider only two candidates, thus $a = p_1$, $a = p_2$ or $a = abstain$ (similarly for b, except $b \neq abstain$). Here we concentrate on the cases $a = p_2$, $b = p_1$ and $a = abstain$, $b = p_1$, the other cases are similar.

— We start with $a = p_2$, $b = p_1$. Since all communication inside the booth takes place over internal channels which yield internal reductions (we do not detail this part) and first condition on C ensures that the targeted voter is forced to vote a, we obtain the following two frames:

$$\begin{aligned}\phi_l = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{^{commit((n_{1,1},p_1),r_{1,1})}/_{x_1}\} \mid \{^{commit((n_{2,1},p_1),r_{2,1})}/_{x_2}\} \mid \\ &\{^{commit((n_{3,1},p_1),r_{3,1})}/_{x_3}\} \mid \{^{commit((n_{1,2},p_2),r_{1,2})}/_{x_4}\} \mid \\ &\{^{commit((n_{2,2},p_2),r_{2,2})}/_{x_5}\} \mid \{^{commit((n_{3,2},p_2),r_{3,2})}/_{x_6}\} \mid \\ &\{^{r_1}/_{x_7}\} \mid \{^{n_{1,2}}/_{x_8}\} \mid \{^{r_1}/_{x_9}\})\end{aligned}$$

$$\begin{aligned}\phi_r = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{^{commit((n_{1,1},p_1),r_{1,1})}/_{x_1}\} \mid \{^{commit((n_{2,1},p_1),r_{2,1})}/_{x_2}\} \mid \\ &\{^{commit((n_{3,1},p_1),r_{3,1})}/_{x_3}\} \mid \{^{commit((n_{1,2},p_2),r_{1,2})}/_{x_4}\} \mid \\ &\{^{commit((n_{2,2},p_2),r_{2,2})}/_{x_5}\} \mid \{^{commit((n_{3,2},p_2),r_{3,2})}/_{x_6}\} \mid \\ &\{^{n_{1,1}}/_{x_7}\} \mid \{^{r_1}/_{x_8}\} \mid \{^{n_{1,1}}/_{x_9}\})\end{aligned}$$

Obviously both frames are statically equivalent. Note that at this point no information about V_B and his vote is available. Now the attacker has to vote himself or abstain, otherwise the voting machines are unable to synchronize and block.

– If he abstains, he can synchronize with processM'', and we obtain the following final frames:

$$\begin{aligned}\phi'_l = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{^{commit((n_{1,1},p_1),r_{1,1})}/_{x_1}\} \mid \{^{commit((n_{2,1},p_1),r_{2,1})}/_{x_2}\} \mid \\ &\{^{commit((n_{3,1},p_1),r_{3,1})}/_{x_3}\} \mid \{^{commit((n_{1,2},p_2),r_{1,2})}/_{x_4}\} \mid \\ &\{^{commit((n_{2,2},p_2),r_{2,2})}/_{x_5}\} \mid \{^{commit((n_{3,2},p_2),r_{3,2})}/_{x_6}\} \mid \\ &\{^{r_1}/_{x_7}\} \mid \{^{n_{1,2}}/_{x_8}\} \mid \{^{r_1}/_{x_9}\} \mid \\ &\{^{p_1}/_{x_{10}}\} \mid \{^{p_2}/_{x_{11}}\} \mid \\ &\{^{r_1}/_{x_{12}}\} \mid \{^{n_{1,2}}/_{x_{13}}\} \mid \{^{n_{2,1}}/_{x_{14}}\} \mid \{^{r_2}/_{x_{15}}\} \mid \\ &\{^{((n_{1,1},p_1),commit((n_{1,1},p_1),r_{1,1}),r_{1,1})}/_{x_{16}}\} \mid \\ &\{^{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})}/_{x_{17}}\} \mid \\ &\{^{((n_{2,2},p_2),commit((n_{2,2},p_2),r_{2,2}),r_{2,2})}/_{x_{18}}\} \mid \\ &\{^{((n_{3,2},p_2),commit((n_{3,2},p_2),r_{3,2}),r_{3,2})}/_{x_{19}}\})\end{aligned}$$

$$\phi'_r = \nu\tilde{x}.\tilde{r}.\tilde{n}.(\{commit((n_{1,1},p_1),r_{1,1})/x_1\} \mid \{commit((n_{2,1},p_1),r_{2,1})/x_2\} \mid \{commit((n_{3,1},p_1),r_{3,1})/x_3\} \mid \{commit((n_{1,2},p_2),r_{1,2})/x_4\} \mid \{commit((n_{2,2},p_2),r_{2,2})/x_5\} \mid \{commit((n_{3,2},p_2),r_{3,2})/x_6\} \mid \{n_{1,1}/x_7\} \mid \{r_1/x_8\} \mid \{n_{1,1}/x_9\} \mid \{p_1/x_{10}\} \mid \{p_2/x_{11}\} \mid \{n_{1,1}/x_{12}\} \mid \{r_1/x_{13}\} \mid \{r_2/x_{14}\} \mid \{n_{2,2}/x_{15}\} \mid \{((n_{2,1},p_1),commit((n_{2,1},p_1),r_{2,1}),r_{2,1})/x_{16}\} \mid \{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})/x_{17})\} \mid \{((n_{1,2},p_2),commit((n_{1,2},p_2),r_{1,2}),r_{1,2})/x_{18})\} \mid \{((n_{3,2},p_2),commit((n_{3,2},p_2),r_{3,2}),r_{3,2})/x_{19})\})$$

It is easy to see that these frames are statically equivalent.

– If the corrupted voter votes, he cannot relate his vote in any way to V_A's vote, as the receipts are meaningless to him and he has to submit his vote in clear. Suppose that he votes for p_1 (the other case is similar). We obtain the following final frames.

$$\phi'_l = \nu\tilde{x}.\tilde{r}.\tilde{n}.(\{commit((n_{1,1},p_1),r_{1,1})/x_1\} \mid \{commit((n_{2,1},p_1),r_{2,1})/x_2\} \mid \{commit((n_{3,1},p_1),r_{3,1})/x_3\} \mid \{commit((n_{1,2},p_2),r_{1,2})/x_4\} \mid \{commit((n_{2,2},p_2),r_{2,2})/x_5\} \mid \{commit((n_{3,2},p_2),r_{3,2})/x_6\} \mid \{r_1/x_7\} \mid \{n_{1,2}/x_8\} \mid \{r_1/x_9\} \mid \{r_3/x_{10}\} \mid \{n_{3,2}/x_{11}\} \mid \{r_3/x_{12}\} \mid \{p_1/x_{13}\} \mid \{p_1/x_{14}\} \mid \{p_2/x_{15}\} \mid \{r_1/x_{16}\} \mid \{n_{1,2}/x_{17}\} \mid \{n_{2,1}/x_{18}\} \mid \{r_2/x_{19}\} \mid \{r_3/x_{20}\} \mid \{n_{3,2}/x_{21}\} \mid \{((n_{1,1},p_1),commit((n_{1,1},p_1),r_{1,1}),r_{1,1})/x_{22}\} \mid \{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})/x_{23}\} \mid \{((n_{2,2},p_2),commit((n_{2,2},p_2),r_{2,2}),r_{2,2})/x_{24}\})$$

$$\phi'_r = \nu\tilde{x}.\tilde{r}.\tilde{n}.(\{commit((n_{1,1},p_1),r_{1,1})/x_1\} \mid \{commit((n_{2,1},p_1),r_{2,1})/x_2\} \mid \{commit((n_{3,1},p_1),r_{3,1})/x_3\} \mid \{commit((n_{1,2},p_2),r_{1,2})/x_4\} \mid \{commit((n_{2,2},p_2),r_{2,2})/x_5\} \mid \{commit((n_{3,2},p_2),r_{3,2})/x_6\} \mid \{n_{1,1}/x_7\} \mid \{r_1/x_8\} \mid \{n_{1,1}/x_9\} \mid \{r_3/x_{10}\} \mid \{n_{3,2}/x_{11}\} \mid \{r_3/x_{12}\} \mid \{p_1/x_{13}\} \mid \{p_1/x_{14}\} \mid \{p_2/x_{15}\} \mid \{n_{1,1}/x_{16}\} \mid \{r_1/x_{17}\} \mid \{r_2/x_{18}\} \mid \{n_{2,2}/x_{19}\} \mid \{r_3/x_{20}\} \mid \{n_{3,2}/x_{21}\} \mid \{((n_{2,1},p_1),commit((n_{2,1},p_1),r_{2,1}),r_{2,1})/x_{22}\} \mid \{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})/x_{23})\} \mid \{((n_{1,2},p_2),commit((n_{1,2},p_2),r_{1,2}),r_{1,2})/x_{24})\})$$

The election outcome and the data published by the voting machine do not help the attacker in distinguishing both cases. He can verify if the receipt

by the coerced voter was correct (which it is), but he still does not know if the numbers on the receipt are unrevealed commitments or fresh random numbers.

— Consider now the case $a = abstain$, $b = p_1$. Since all communication inside the booth takes place over internal channels which yield internal reductions (we again do not detail this part) and first condition on C ensures that the targeted voter is forced to abstain. Hence he does reveal any receipt (nor does V'), thus we obtain the following two frames:

$$\begin{aligned}\phi_l = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{^{commit((n_{1,1},p_1),r_{1,1})}/_{x_1}\} \mid \{^{commit((n_{2,1},p_1),r_{2,1})}/_{x_2}\} \mid \\ &\{^{commit((n_{3,1},p_1),r_{3,1})}/_{x_3}\} \mid \{^{commit((n_{1,2},p_2),r_{1,2})}/_{x_4}\} \mid \\ &\{^{commit((n_{2,2},p_2),r_{2,2})}/_{x_5}\} \mid \{^{commit((n_{3,2},p_2),r_{3,2})}/_{x_6}\})\end{aligned}$$

$$\begin{aligned}\phi_r = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{^{commit((n_{1,1},p_1),r_{1,1})}/_{x_1}\} \mid \{^{commit((n_{2,1},p_1),r_{2,1})}/_{x_2}\} \mid \\ &\{^{commit((n_{3,1},p_1),r_{3,1})}/_{x_3}\} \mid \{^{commit((n_{1,2},p_2),r_{1,2})}/_{x_4}\} \mid \\ &\{^{commit((n_{2,2},p_2),r_{2,2})}/_{x_5}\} \mid \{^{commit((n_{3,2},p_2),r_{3,2})}/_{x_6}\})\end{aligned}$$

Both frames are syntactically equivalent, hence also statically equivalent. Thus that at this point no information about V_B and his vote is available.
Now the attacker has to vote himself or abstain, otherwise the voting machines are unable to synchronize and block.

– If he abstains, he can synchronize with processM'', and we obtain the following final frames:

$$\begin{aligned}\phi'_l = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{^{commit((n_{1,1},p_1),r_{1,1})}/_{x_1}\} \mid \{^{commit((n_{2,1},p_1),r_{2,1})}/_{x_2}\} \mid \\ &\{^{commit((n_{3,1},p_1),r_{3,1})}/_{x_3}\} \mid \{^{commit((n_{1,2},p_2),r_{1,2})}/_{x_4}\} \mid \\ &\{^{commit((n_{2,2},p_2),r_{2,2})}/_{x_5}\} \mid \{^{commit((n_{3,2},p_2),r_{3,2})}/_{x_6}\} \mid \\ &\{^{p_1}/_{x_7}\} \mid \\ &\{^{r_2}/_{x_8}\} \mid \{^{n_{2,2}}/_{x_9}\} \mid \\ &\{^{((n_{1,1},p_1),commit((n_{1,1},p_1),r_{1,1}),r_{1,1})}/_{x_{10}}\} \mid \\ &\{^{((n_{2,1},p_1),commit((n_{2,1},p_1),r_{2,1}),r_{2,1})}/_{x_{11}}\} \mid \\ &\{^{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})}/_{x_{12}}\} \mid \\ &\{^{((n_{1,2},p_2),commit((n_{1,2},p_2),r_{1,2}),r_{1,2})}/_{x_{13}}\} \mid \\ &\{^{((n_{3,2},p_2),commit((n_{3,2},p_2),r_{3,2}),r_{3,2})}/_{x_{14}}\})\end{aligned}$$

$$\phi'_r = \nu\tilde{x}.\tilde{r}.\tilde{n}.(\{commit((n_{1,1},p_1),r_{1,1})/x_1\} \mid \{commit((n_{2,1},p_1),r_{2,1})/x_2\} \mid \{commit((n_{3,1},p_1),r_{3,1})/x_3\} \mid \{commit((n_{1,2},p_2),r_{1,2})/x_4\} \mid \{commit((n_{2,2},p_2),r_{2,2})/x_5\} \mid \{commit((n_{3,2},p_2),r_{3,2})/x_6\} \mid \{p_1/x_7\} \mid \{r_1/x_8\} \mid \{n_{1,2}/x_9\} \mid \{((n_{1,1},p_1),commit((n_{1,1},p_1),r_{1,1}),r_{1,1})/x_{10}\} \mid \{((n_{2,1},p_1),commit((n_{2,1},p_1),r_{2,1}),r_{2,1})/x_{11}\} \mid \{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})/x_{12})\} \mid \{((n_{2,2},p_2),commit((n_{2,2},p_2),r_{2,2}),r_{2,2})/x_{13})\} \mid \{((n_{3,2},p_2),commit((n_{3,2},p_2),r_{3,2}),r_{3,2})/x_{14})\})$$

It is easy to see that these frames are statically equivalent.

– If the corrupted voter votes, he cannot relate his vote in any way to V_A's vote, as the receipts are meaningless to him and he has to submit his vote in clear. Suppose that he votes for p_1 (the other case is similar). We obtain the following final frames.

$$\phi'_l = \nu\tilde{x}.\tilde{r}.\tilde{n}.(\{commit((n_{1,1},p_1),r_{1,1})/x_1\} \mid \{commit((n_{2,1},p_1),r_{2,1})/x_2\} \mid \{commit((n_{3,1},p_1),r_{3,1})/x_3\} \mid \{commit((n_{1,2},p_2),r_{1,2})/x_4\} \mid \{commit((n_{2,2},p_2),r_{2,2})/x_5\} \mid \{commit((n_{3,2},p_2),r_{3,2})/x_6\} \mid \{r_3/x_7\} \mid \{n_{3,2}/x_8\} \mid \{r_3/x_9\} \mid \{p_1/x_{10}\} \mid \{p_1/x_{11}\} \mid \{r_2/x_{12}\} \mid \{n_{2,2}/x_{13}\} \mid \{r_3/x_{14}\} \mid \{n_{3,2}/x_{15}\} \mid \{((n_{1,1},p_1),commit((n_{1,1},p_1),r_{1,1}),r_{1,1})/x_{16}\} \mid \{((n_{2,1},p_1),commit((n_{2,1},p_1),r_{2,1}),r_{2,1})/x_{17}\} \mid \{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})/x_{18}\} \mid \{((n_{1,2},p_2),commit((n_{1,2},p_2),r_{1,2}),r_{1,2})/x_{19}\})$$

$$\phi'_r = \nu\tilde{x}.\tilde{r}.\tilde{n}.(\{commit((n_{1,1},p_1),r_{1,1})/x_1\} \mid \{commit((n_{2,1},p_1),r_{2,1})/x_2\} \mid \{commit((n_{3,1},p_1),r_{3,1})/x_3\} \mid \{commit((n_{1,2},p_2),r_{1,2})/x_4\} \mid \{commit((n_{2,2},p_2),r_{2,2})/x_5\} \mid \{commit((n_{3,2},p_2),r_{3,2})/x_6\} \mid \{r_3/x_7\} \mid \{n_{3,2}/x_8\} \mid \{r_3/x_9\} \mid \{p_1/x_{10}\} \mid \{p_1/x_{11}\} \mid \{r_1/x_{12}\} \mid \{n_{1,2}/x_{13}\} \mid \{r_3/x_{14}\} \mid \{n_{3,2}/x_{15}\} \mid \{((n_{1,1},p_1),commit((n_{1,1},p_1),r_{1,1}),r_{1,1})/x_{16}\} \mid \{((n_{2,1},p_1),commit((n_{2,1},p_1),r_{2,1}),r_{2,1})/x_{17}\} \mid \{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})/x_{18})\} \mid \{((n_{2,2},p_2),commit((n_{2,2},p_2),r_{2,2}),r_{2,2})/x_{19})\})$$

Again, the election outcome and the data published by the voting machine do not help the attacker in distinguishing both cases. He can see that one

other voter voted for candidate one, but he cannot decide which one. □

Note that in this case we implicitly assume that the attacker does not know if a voter enters the voting booth, which is not necessarily a realistic assumption. To address this, we can modify e.g. the voters process, or the voting machine to announce the list of participants[9] (see Listing 3.20). In this case we still have $SwCR^{I,PO}$, i.e. Coercion-Resistance against an inside attacker, but no security against forced-abstention attacks any more.

Theorem 24 *Bingo Voting with voter lists respects $SwCR^{I,PO}$, i.e. Coercion-Resistance against an inside attacker in the Participation Only case.*

Proof To show that Bingo Voting with voter lists ensures $SwCR^{I,PO}$, we show that for any $C = \nu c_1.\nu c_2.(\_|P)$ satisfying $\tilde{n} \cap fn(C) = \emptyset$ and

$$\begin{gathered}\nu\tilde{\mathtt{ch}}.(C\,[V_A^{c_1,c_2}]\,|V_B\,\{^b/_v\}\,|V_C^{c_1,c_2}|\boldsymbol{Ml_{k=3,l=2}}|R_A|R_B|R_C)\\ \approx_l\\ \nu\tilde{\mathtt{ch}}.(V_A\,\{^a/_v\}^{chc}\,|V_B\,\{^b/_v\}\,|V_C^{c_1,c_2}|\boldsymbol{Ml_{k=3,l=2}}|R_A|R_B|R_C)\end{gathered}$$

we have

$$C\,[V']^{\backslash out(chc,\cdot)} \approx_l V_A\,\{^b/_v\}$$

and

$$\begin{gathered}\nu\tilde{\mathtt{ch}}.(C\,[V_A^{c_1,c_2}]\,|V_B\,\{^b/_v\}\,|V_C^{c_3,c_4}|\boldsymbol{Ml_{k=3,l=2}}|R_A|R_B|R_C)\\ \approx_l\\ \nu\tilde{\mathtt{ch}}.(C\,[V']\,|V_B\,\{^a/_v\}\,|V_C^{c_3,c_4}|\boldsymbol{Ml_{k=3,l=2}}|R_A|R_B|R_C)\end{gathered}$$

where

$$\begin{aligned}\nu\tilde{\mathtt{ch}} &= \nu\mathtt{privChM}_1.\nu\mathtt{privChM}_2.\nu\mathtt{privChM}_3.\\ &\quad\ \nu\mathtt{privChRM}_1.\nu\mathtt{privChRM}_2.\nu\mathtt{privChRM}_3.\\ &\quad\ \nu\mathtt{privChR}_1.\nu\mathtt{privChR}_2.\nu\mathtt{privChR}_3\\ V_A &= \mathtt{processV}\,\{^{privChM_1}/_{privChM}, {}^{privChR_1}/_{privChRNG}\}\\ V_B &= \mathtt{processV}\,\{^{privChM_2}/_{privChM}, {}^{privChR_2}/_{privChRNG}\}\\ V_C &= \mathtt{processV}\,\{^{privChM_3}/_{privChM}, {}^{privChR_3}/_{privChRNG}\}\\ V' &= \mathtt{processV'}\,\{^{privChM_1}/_{privChM}, {}^{privChR_1}/_{privChRNG}\}\\ R_A &= \mathtt{processRNG}\,\{^{privChRM_1}/_{privChM}, {}^{privChR_1}/_{privChV}\}\\ R_B &= \mathtt{processRNG}\,\{^{privChRM_2}/_{privChM}, {}^{privChR_2}/_{privChV}\}\\ R_C &= \mathtt{processRNG}\,\{^{privChRM_3}/_{privChM}, {}^{privChR_3}/_{privChV}\}\\ \boldsymbol{Ml_{k=k',l=l'}} &= \mathtt{processMl}\,\{^{k'}/_k, {}^{l'}/_l\}\end{aligned}$$

Again, we mark differences to the above proof in **bold**.

[9] This is for example a legal requirement for public elections in France [Fra69].

```
1  processMl' =
2   in(privChV,v).in(privChR,r).
3   for(j = 1 to l)
4    let receipt_j = choose(p_j,v,r,n_{i,j}) in
5     out(privChV,receipt_j);
6   sync_k 1.
7   (* output result *)
8   out(res,v)
9   (* output voter list *)
10  out(voters,id_i)
11  (* output receipts *)
12  for(j = 1 to l)
13   out(chRec,receipt_j)
14  (* output unused dummy votes *)
15  parfor(j = 1 to l)
16   if v_i ≠p_j then out(chDum,((n_{i,j},p_j),commit((n_{i,j},p_j),r_{i,j}),r_{i,j}))
17
18 processMl'' =
19  sync id_i.
20  sync_k 1.
21  (* output unused dummy votes *)
22  parfor(j = 1 to l)
23   out(chDum,((n_{i,j},p_j),commit((n_{i,j},p_j),r_{i,j}),r_{i,j}))
24
25 processMl =
26  (* prepare dummy votes *)
27  for(i = 1 to k)
28   for(j = 1 to l)
29    νn_{i,j}.νr_{i,j}.
30  parfor(i = 1 to k)
31   parfor(j = 1 to l)
32    out(ch,commit((n_{i,j},p_j),r_{i,j}))
33   (* voting *)
34   let privChV = privChM_i in
35    let privChR = privChRM_i in (processMl' | processMl'')
```

Listing 3.20 – The voting machine announcing a voter list

This is technically only a valid proof for two honest voters and two candidates, however a similar proof can be done for an arbitrary number of voters and candidates.

It is easy to see that $C\,[V']^{\backslash out(chc,\cdot)} \approx_l V_A\,\{^b/_v\}$ holds. If we ignore all inputs of V' on `c1` and all outputs on `c2`, we obtain V_A as both cases (abstention or not) coincide.

We now have to show that the last equivalence holds. As before, we denote the left hand side P and the right hand side Q. For better readability we concentrate on the important steps. At the beginning the voting machine publishes all commitments; then the voters enter the voting booth and vote. We consider only two candidates, thus $b = p_1$ or $b = p_2$ (similarly for a). Here we will look at the case $b = p_2$ and $a = p_1$, the other cases are similar. Since all communication inside the booth takes place over internal channels which yield internal reductions (we do not detail this part) and first condition on C ensures that the targeted voter is forced to vote a, we obtain the following two frames:

$$\begin{aligned}\phi_l = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{^{commit((n_{1,1},p_1),r_{1,1})}/_{x_1}\} \mid \{^{commit((n_{2,1},p_1),r_{2,1})}/_{x_2}\} \mid\\ &\{^{commit((n_{3,1},p_1),r_{3,1})}/_{x_3}\} \mid \{^{commit((n_{1,2},p_2),r_{1,2})}/_{x_4}\} \mid\\ &\{^{commit((n_{2,2},p_2),r_{2,2})}/_{x_5}\} \mid \{^{commit((n_{3,2},p_2),r_{3,2})}/_{x_6}\} \mid\\ &\{^{r_1}/_{x_7}\} \mid \{^{n_{1,2}}/_{x_8}\} \mid \{^{r_1}/_{x_9}\})\end{aligned}$$

$$\begin{aligned}\phi_r = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{^{commit((n_{1,1},p_1),r_{1,1})}/_{x_1}\} \mid \{^{commit((n_{2,1},p_1),r_{2,1})}/_{x_2}\} \mid\\ &\{^{commit((n_{3,1},p_1),r_{3,1})}/_{x_3}\} \mid \{^{commit((n_{1,2},p_2),r_{1,2})}/_{x_4}\} \mid\\ &\{^{commit((n_{2,2},p_2),r_{2,2})}/_{x_5}\} \mid \{^{commit((n_{3,2},p_2),r_{3,2})}/_{x_6}\} \mid\\ &\{^{n_{1,1}}/_{x_7}\} \mid \{^{r_1}/_{x_8}\} \mid \{^{n_{1,1}}/_{x_9}\})\end{aligned}$$

Obviously both frames are statically equivalent. Note that at this point no information about V_B and his vote is available. Now the attacker has to vote himself or abstain, otherwise the voting machines are unable to synchronize and block.

— If he abstains, he can synchronize with processMl'', and we obtain the following final frames:

$$\begin{aligned}\phi_l' = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{commit((n_{1,1},p_1),r_{1,1})/x_1\} \mid \{commit((n_{2,1},p_1),r_{2,1})/x_2\} \mid\\ &\{commit((n_{3,1},p_1),r_{3,1})/x_3\} \mid \{commit((n_{1,2},p_2),r_{1,2})/x_4\} \mid\\ &\{commit((n_{2,2},p_2),r_{2,2})/x_5\} \mid \{commit((n_{3,2},p_2),r_{3,2})/x_6\} \mid\\ &\{r_1/x_7\} \mid \{n_{1,2}/x_8\} \mid \{r_1/x_9\} \mid\\ &\{p_1/x_{10}\} \mid \{p_2/x_{11}\} \mid\\ &\boldsymbol{\{id_1/x_{12}\} \mid \{id_2/x_{13}\} \mid}\\ &\{r_1/x_{14}\} \mid \{n_{1,2}/x_{15}\} \mid \{n_{2,1}/x_{16}\} \mid \{r_2/x_{17}\} \mid\\ &\{((n_{1,1},p_1),commit((n_{1,1},p_1),r_{1,1}),r_{1,1})/x_{18}\} \mid\\ &\{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})/x_{19}\} \mid\\ &\{((n_{2,2},p_2),commit((n_{2,2},p_2),r_{2,2}),r_{2,2})/x_{20}\} \mid\\ &\{((n_{3,2},p_2),commit((n_{3,2},p_2),r_{3,2}),r_{3,2})/x_{21}\})\end{aligned}$$

$$\begin{aligned}\phi_r' = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{commit((n_{1,1},p_1),r_{1,1})/x_1\} \mid \{commit((n_{2,1},p_1),r_{2,1})/x_2\} \mid\\ &\{commit((n_{3,1},p_1),r_{3,1})/x_3\} \mid \{commit((n_{1,2},p_2),r_{1,2})/x_4\} \mid\\ &\{commit((n_{2,2},p_2),r_{2,2})/x_5\} \mid \{commit((n_{3,2},p_2),r_{3,2})/x_6\} \mid\\ &\{n_{1,1}/x_7\} \mid \{r_1/x_8\} \mid \{n_{1,1}/x_9\} \mid\\ &\{p_1/x_{10}\} \mid \{p_2/x_{11}\} \mid\\ &\boldsymbol{\{id_1/x_{12}\} \mid \{id_2/x_{13}\} \mid}\\ &\{n_{1,1}/x_{14}\} \mid \{r_1/x_{15}\} \mid \{r_2/x_{16}\} \mid \{n_{2,2}/x_{17}\} \mid\\ &\{((n_{2,1},p_1),commit((n_{2,1},p_1),r_{2,1}),r_{2,1})/x_{18}\} \mid\\ &\{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})/x_{19})\} \mid\\ &\{((n_{1,2},p_2),commit((n_{1,2},p_2),r_{1,2}),r_{1,2})/x_{20})\} \mid\\ &\{((n_{3,2},p_2),commit((n_{3,2},p_2),r_{3,2}),r_{3,2})/x_{21})\})\end{aligned}$$

It is easy to see that these frames are statically equivalent.

— If the corrupted voter votes, he cannot relate his vote in any way to V_A's vote, as the receipts are meaningless to him and he has to submit his vote in clear. Suppose that he votes for p_1 (the other case is similar). We obtain the following final frames.

$$\begin{aligned}\phi_l' = \nu\tilde{x}.\tilde{r}.\tilde{n}.(&\{commit((n_{1,1},p_1),r_{1,1})/x_1\} \mid \{commit((n_{2,1},p_1),r_{2,1})/x_2\} \mid\\ &\{commit((n_{3,1},p_1),r_{3,1})/x_3\} \mid \{commit((n_{1,2},p_2),r_{1,2})/x_4\} \mid\\ &\{commit((n_{2,2},p_2),r_{2,2})/x_5\} \mid \{commit((n_{3,2},p_2),r_{3,2})/x_6\} \mid\\ &\{r_1/x_7\} \mid \{n_{1,2}/x_8\} \mid \{r_1/x_9\} \mid\\ &\{r_3/x_{10}\} \mid \{n_{3,2}/x_{11}\} \mid \{r_3/x_{12}\} \mid\\ &\{p_1/x_{13}\} \mid \{p_1/x_{14}\} \mid \{p_2/x_{15}\} \mid\\ &\boldsymbol{\{id_1/x_{16}\} \mid \{id_2/x_{17}\} \mid \{id_3/x_{18}\} \mid}\\ &\{r_1/x_{19}\} \mid \{n_{1,2}/x_{20}\} \mid \{n_{2,1}/x_{21}\} \mid \{r_2/x_{22}\} \mid \{r_3/x_{23}\} \mid \{n_{3,2}/x_{24}\} \mid\\ &\{((n_{1,1},p_1),commit((n_{1,1},p_1),r_{1,1}),r_{1,1})/x_{25}\} \mid\\ &\{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})/x_{26}\} \mid\\ &\{((n_{2,2},p_2),commit((n_{2,2},p_2),r_{2,2}),r_{2,2})/x_{27}\})\end{aligned}$$

$$\phi'_r = \nu\tilde{x}.\tilde{r}.\tilde{n}.(\{commit((n_{1,1},p_1),r_{1,1})/x_1\} \mid \{commit((n_{2,1},p_1),r_{2,1})/x_2\} \mid$$
$$\{commit((n_{3,1},p_1),r_{3,1})/x_3\} \mid \{commit((n_{1,2},p_2),r_{1,2})/x_4\} \mid$$
$$\{commit((n_{2,2},p_2),r_{2,2})/x_5\} \mid \{commit((n_{3,2},p_2),r_{3,2})/x_6\} \mid$$
$$\{n_{1,1}/x_7\} \mid \{r_1/x_8\} \mid \{n_{1,1}/x_9\} \mid$$
$$\{r_3/x_{10}\} \mid \{n_{3,2}/x_{11}\} \mid \{r_3/x_{12}\} \mid$$
$$\{p_1/x_{13}\} \mid \{p_1/x_{14}\} \mid \{p_2/x_{15}\} \mid$$
$$\boldsymbol{\{id_1/x_{16}\} \mid \{id_2/x_{17}\} \mid \{id_3/x_{18}\} \mid}$$
$$\{n_{1,1}/x_{19}\} \mid \{r_1/x_{20}\} \mid \{r_2/x_{21}\} \mid \{n_{2,2}/x_{22}\} \mid \{r_3/x_{23}\} \mid \{n_{3,2}/x_{24}\} \mid$$
$$\{((n_{2,1},p_1),commit((n_{2,1},p_1),r_{2,1}),r_{2,1})/x_{25}\} \mid$$
$$\{((n_{3,1},p_1),commit((n_{3,1},p_1),r_{3,1}),r_{3,1})/x_{26})\} \mid$$
$$\{((n_{1,2},p_2),commit((n_{1,2},p_2),r_{1,2}),r_{1,2})/x_{27})\})$$

Again, the election outcome and the published data do not help the attacker in distinguishing both cases. He can verify if the receipt by the coerced voter was correct (which it is), but he still does not know if the numbers on the receipt are unrevealed commitments or fresh random numbers. □

§ 3.3.5.4. Protocol by Lee, Boyd, Dawson, Kim, Yang and Yoo. The protocol by Lee, Boyd, Dawson, Kim, Yang and Yoo [LBD+03] was shown to be Coercion-Resistant in the DKR-model [DKR09]. Yet the protocol is neither secure against an inside attacker nor against forced-abstention attacks, as we show. It is based on trusted devices that re-encrypt ballots and prove their correct behavior to the voter using designated verifier proofs (DVPs).

Protocol Description. We simplified the protocol to focus on the important parts with respect to privacy. For example, we do not consider distributed authorities, but model them as one honest authority.

— The administrator sets up the election, distributes keys and registers legitimate voters. Each voter is equipped with his personal trusted device. At the end, he publishes a list of legitimate voters and corresponding trusted devices.

— The voter encrypts his vote with the tallier's public key (using the El Gamal scheme), signs it and sends it to his trusted device over a private channel. The trusted device verifies the signature, re-encrypts and signs the vote, and returns it, together with a DVP that the re-encryption is correct, to the voter. The voter verifies the signature and the proof, double signs the ballot and publishes it on the bulletin board.

— The administrator verifies for all ballots if the voter has the right to vote and if the vote is correctly signed. He publishes the list of correct ballots, which is then shuffled by the mixer.

— The tallier decrypts the mixed votes and publishes the result.

In the following we use a simplified protocol similar to the one was proposed by Delaune et al. [DKR09]: the administrator, the mixer and the tallier are replaced by one honest collector which executes the last two steps.

Model in Applied Pi Calculus. We use a model similar to the one developed by Delaune et al. [DKR09], but we include signatures by the voters on the submitted ballots and add a third voter. This model is closer to the original protocol description. It uses the following equational theory:

$$\mathtt{decrypt}(\mathtt{penc}(m, \mathtt{pk}(sk), r), sk) = m$$
$$\mathtt{checksign}(\mathtt{sign}(m, sk), \mathtt{pk}(sk)) = m$$
$$\mathtt{rencrypt}(\mathtt{penc}(m, \mathtt{pk}(sk), r1), r2) = \mathtt{penc}(m, \mathtt{pk}(sk), f(r1, r2))$$
$$\mathtt{checkdvp}(\mathtt{dvp}(x, \mathtt{rencrypt}(x, r), r, \mathtt{pk}(sk)), x, \mathtt{rencrypt}(x, r), \mathtt{pk}(sk)) = \mathtt{ok}$$
$$\mathtt{checkdvp}(\mathtt{dvp}(x, y, z, skv), x, y, \mathtt{pk}(skv)) = \mathtt{ok}$$

These equations are standard, except for the reencryption and DVPs. The function `reencrypt` models reencryption usig a new random value. The second to last equation allows to verify a DVP of reencryption, and the last equation allows the designated verifier to forge a DVP for a different value using his secret key.

We then use the following processes:

The *main process* (Listing 3.21 on the facing page) sets up the private channels and executes the participating processes (three voters, three mixers, three talliers - one for each voter - and the keying process).

The *keying process* (Listing 3.22 on page 106) creates the private keys, distributes them over private channels and publishes the corresponding public keys. Each voter is equipped with his private key, the tallier's public key and the public key of his trusted device. Listing 3.23 on page 107 is variant of the keying process for two voters.

The voter process (Listing 3.24 on page 107) receives the necessary keys. Then he encrypts his votes, signs it and sends it over a private channel to his trusted device. When he receives the answer, he checks the DVP, double signs the ballot and sends it to the mixer (which corresponds to the publication on the bulletin board).

The *trusted device* (Listing 3.26 on page 108) receives his private and the voter's public key. When he receives an encrypted ballot from the voter, he checks the signature, re-encrypts the message, signs it and establishes a DVP to prove the correctness of his re-encryption. The result is send back to the voter.

The process in Listing 3.27 on page 109 models *a voter controlled by the attacker*. This process is obtained when calculating $\mathtt{processV}^{c_1,c_2}$ as defined in Definition 19 on page 59. Note that his trusted device is not assumed to be

```
1  (* private channels *)
2  νskcCh.νpkcCh.νvotCh.
3  νskvaCh.νskvbCh.νskvcCh.
4  νskdaCh.νskdbCh.νskdcCh.
5  νpkvaCh.νpkvbCh.νpkvcCh.
6  νpkdaCh.νpkdbCh.νpkdcCh.
7  (* administrators *)
8  (processK | processC | processC | processC |
9  (* voters *)
10 (let skvCh = skvaCh in let pkdCh = pkdaCh in
11  let v = a in processV) |
12 (let skvCh = skvbCh in let pkdCh = pkdbCh in
13  let v = b in processV) |
14 (let skvCh = skvcCh in let pkdCh = pkdcCh in
15  let v = c in processV) |
16 (* trusted devices *)
17 (let skdCh = skdaCh in
18  let pkvCh = pkvaCh in processD) |
19 (let skdCh = skdbCh in
20  let pkvCh = pkvbCh in processD) |
21 (let skdCh = skdcCh in
22  let pkvCh = pkvcCh in processD))
```

Listing 3.21 – The main process

corrupted.

The *collector* (Listing 3.29 on page 110) receives the key pair corresponding to a legitimate voter (the public keys of the voter and his trusted device) and his private key. When receives ballot, he checks the signatures and decrypts the ballots. He publishes the result on a public channel (which emulates the bulletin board).

Analysis. We note that the protocol also ensures Coercion-Resistance in our model: it ensures $SwCR^{O,PO}$. This result was already shown in [DKR09], however using a different model. We prove that the result also holds in our model, and then show that the protocol is not secure against inside attackers as vote-copying attacks are possible. The main difference to the model used in [DKR09] is that in our model the voters sign the re-encrypted ballots, as in the original protocol description. This is important with respect to vote-copying and forced abstention attacks, as the signatures allow the attacker to link the ballots to the voters.

Theorem 25 *The protocol by Lee et al. respects $SwCR^{O,PO}$ (Swap-Coercion-Resistance against an outside attacker in the Participation Only case).*

Proof We suppose that all authorities are honest and that all key distribution

```
processK =
 (* private keys *)
 νskc.νskva.νskvb.νskvc.νskda.νskdb.νskdc.
 (* public keys *)
 let (pkc,pkva,pkvb,pkvc,pkda,pkdb,pkdc)
  = (pk(skc),pk(skva),pk(skvb),pk(skvc),
     pk(skda),pk(skdb),pk(skdc)) in
 (* public key disclosure *)
 out(ch,pkc).
 out(ch,pkva).out(ch,pkvb).out(ch,pkvc).
 out(ch,pkda).out(ch,pkdb).out(ch,pkdc).
 (* distribute keys: *)
 (* voters *)
 (out(skvaCh,skva) | out(skvbCh,skvb) | out(skvcCh,skvc) |
  out(pkdaCh,pkda) | out(pkdbCh,pkdb) | out(pkdcCh,pkdc) |
  out(pkcCh,pkc)   | out(pkcCh,pkc)   | out(pkcCh,pkc)   |
 (* trusted devices *)
  out(skdaCh,skva) | out(skdbCh,skvb) | out(skdcCh,skvc) |
  out(pkvaCh,pkva) | out(pkvbCh,pkvb) | out(pkvcCh,pkvc) |
 (* collectors *)
  out(votCh,(pkva,pkda)) |
  out(votCh,(pkvb,pkdb)) |
  out(votCh,(pkvc,pkdc)) |
  out(skcCh,skc) | out(skcCh,skc) | out(skcCh,skc))
```

Listing 3.22 – The key distribution process

```
processK2 =
 (* private keys *)
 νskc.νskva.νskvb.νskda.νskdb.
 (* public keys *)
 let (pkc,pkva,pkvb,pkda,pkdb)
  = (pk(skc),pk(skva),pk(skvb),pk(skda),pk(skdb)) in
 (* public key disclosure *)
 out(ch,pkc).
 out(ch,pkva).out(ch,pkvb).out(ch,pkda).out(ch,pkdb).
 (* distribute keys: *)
 (* voters *)
 (out(skvaCh,skva) | out(skvbCh,skvb) |
  out(pkdaCh,pkda) | out(pkdbCh,pkdb) |
  out(pkcCh,pkc)   | out(pkcCh,pkc)   |
 (* trusted devices *)
  out(skdaCh,skva) | out(skdbCh,skvb) |
  out(pkvaCh,pkva) | out(pkvbCh,pkvb) |
 (* collectors *)
  out(votCh,(pkva,pkda)) |
  out(votCh,(pkvb,pkdb)) |
  out(skcCh,skc) | out(skcCh,skc))
```

Listing 3.23 – The key distribution process for two voters

```
processV =
 (* private key *)
 in(skvCh,skv).
 (* public keys of the trusted device and the collector *)
 in(pkdCh,pubkd).in(pkcCh,pubkc).
 sync 1.νr.
 let e = penc(v,pubkc,r) in
 out(chD,(pk(skv),e,sign(e,skv))).
 in(chD,m2).
 let (re,sd,dvpV) = m2 in
 if checkdvp(dvpV,e,re,pk(skv)) = ok then
  if checksign(sd,pubkd) = re then
   out(ch1,(re,sign(sd,skv)))
```

Listing 3.24 – The voting process

```
processVchc =
 (* private key *)
 in(skvCh,skv).out(chc,skv).
 (* public keys of the trusted device and the collector *)
 in(pkdCh,pubkd).out(chc,pubkd).
 in(pktCh,pubkc).out(chc,pubkc).
 sync 1.νr.out(chc,r).
 let e = penc(v,pubkc,r) in
 out(chD,(pk(skv),e,sign(e,skv))).
 in(chD,m2).out(chc,m2).
 let (re,sd,dvpV) = m2 in
 if checkdvp(dvpV,e,re,pk(skv)) = ok then
  if checksign(sd,pubkd) = re then
   out(ch1,(re,sign(sd,skv)))
```

Listing 3.25 – The voting process revealing its secret data

```
processD =
 (* private key *)
 in(skdCh,skd).
 (* public key of the voter *)
 in(pkvCh,pubkv).
 sync 1.
 in(chD,m1).
 let (pubv,enc,sig) = m1 in
 if pubv = pubkv then
  if checksign(sig,pubkv) = enc then
   νr1.
   let reenc = rencrypt(enc,r1) in
   let signD = sign(reenc,skd) in
   let dvpD = dvp(enc,reenc,r1,pubkv) in
   out(chD,(reenc,signD,dvpD))
```

Listing 3.26 – The trusted device process

```
processVc1c2 =
 (* private key *)
 in(skvCh,skv).out(c1,skv).
 (* public keys of the trusted device and the collector *)
 in(pkdCh,pubkd).out(c1,pubkd).
 in(pkcCh,pubkc).out(c1,pubkc).
 sync 1.νr.out(c1,r).
 let e = penc(v,pubkc,r) in
 in(c2,m1).out(chD,m1).
 in(chD,m2).out(c1,m2).
 let (re,sd,dvpV) = m2 in
 in(c2,m3).
 if m3 = true then
  in(c2,m4).
  if m4 = true then
   in(c2,m5).out(ch1,m5)
```

Listing 3.27 – The voting process under control of the attacker

```
processV' =
 (* private key *)
 in(skvCh,skv).out(c1,skv).
 (* public keys of the trusted device and the tallier *)
 in(pkdCh,pubkd).out(c1,pubkd).
 in(pkcCh,pubkc).out(c1,pubkc).
 sync 1.νr.out(c1,r).
 let e = penc(v,pubkc,r) in
 in(c2,m1).out(chD,(pk(skv),e,sign(e,skv))).
 let (pka,ea,sa) = m1 in
 in(chD,m2).
 let (re,sd,dvpV) = m2 in
 if checkdvp(dvpV,e,re,pk(skv)) = ok then
  if checksign(sd,pubkd) = re then
   νr'.let fk = dvp(ea,re,r',skv) in
   out(c1,(re,sd,fk)).
   in(c2,m3).
   if m3 = true then
    in(c2,m4).
    if m4 = true then
     in(c2,m5).out(ch1,m5)
```

Listing 3.28 – The voting process resisting coercion

```
processC =
 (* register legitimate voters *)
 in(votCh,(pubkv,pubkd)).
 (* collector's secret key *)
 in(skcCh,skt).
 sync 1.
 in(ch1,m1).
 let (enc,sig) = m1 in
 if checksign(checksign(sig,pubkv),pubkd) = enc then
  sync 2.
  out(res,decrypt(m,skc))
```

Listing 3.29 – The collector process

channels as well as the channel to the trusted device are private. We show that there exists a process V' such that for all votes σ_{v_A} and σ_{v_B} where neither σ_{v_A} nor σ_{v_B} makes a voter abstain, the following holds:

For any context $C = \nu c_1.\nu c_2.(\_|P')$ with $\tilde{n} \cap fn(C) = \emptyset$ and

$$S'\left[C\left[(V\sigma_{id_A})^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}\right] \approx_l S'\left[(V\sigma_{id_A}\sigma_{v_A})^{chc}|V\sigma_{id_B}\sigma_{v_B}\right] \tag{3.10}$$

we have

$$C\left[V'\right]^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B} \tag{3.11}$$

and

$$S'\left[C\left[(V\sigma_{id_A})^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}\right] \approx_l S'\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{v_A}\right] \tag{3.12}$$

where

$$\begin{aligned}
S' &= \nu\tilde{\mathtt{ch}}.(_|D_A|D_B|\mathtt{processK2}|\mathtt{processC}|\mathtt{processC}) \\
\nu\tilde{\mathtt{ch}} &= \nu\mathtt{sktCh}.\nu\mathtt{pktCh}.\nu\mathtt{votCh}.\nu\mathtt{skvcCh}.\nu\mathtt{chD}. \\
&\quad \nu\mathtt{skvaCh}.\nu\mathtt{skvbCh}.\nu\mathtt{skdaCh}.\nu\mathtt{skdbCh}. \\
&\quad \nu\mathtt{pkvaCh}.\nu\mathtt{pkvbCh}.\nu\mathtt{pkdaCh}.\nu\mathtt{pkdbCh} \\
V &= \mathtt{processV} \\
V' &= \mathtt{processV}'\left\{{}^{skvaCh}/_{skvCh}, {}^{pkdaCh}/_{pkdCh}\right\}\sigma_{v_B} \\
\sigma_{id_A} &= \left\{{}^{skvaCh}/_{skvCh}, {}^{pkdaCh}/_{pkdCh}\right\} \\
\sigma_{id_B} &= \left\{{}^{skvbCh}/_{skvCh}, {}^{pkdbCh}/_{pkdCh}\right\} \\
D_A &= \mathtt{processD}\left\{{}^{skdaCh}/_{skdCh}, {}^{pkvaCh}/_{pkvCh}\right\} \\
D_B &= \mathtt{processD}\left\{{}^{skdbCh}/_{skdCh}, {}^{pkvbCh}/_{pkvCh}\right\}
\end{aligned}$$

This proof is technically only valid for two honest voters, however a similar proof can be done for an arbitrary number of voters.

This proof is particularly challenging as we have to reason about all contexts C that fulfill the above conditions. We start by analyzing how this context has to

behave, commencing our reasoning from the result. The first condition ensures that the announced result at the end contains a vote v_A and a vote v_B. As the voter id_B always votes v_B, the context has to submit a vote for v_A. Hence the last message `m5` must be a valid vote for v_A. This also implies that messages `m4` and `m3` contain `true`, otherwise no vote is submitted.

As the trusted device's keys are secret, the only way for the context to construct a correct ballot is to submit it to the device. Otherwise he is unable to construct a valid signature that can be accepted by the talliers. This implies that he submits a ballot containing v_A as message `m1`. As $(V\sigma_{id_A}\sigma_{v_A})^{chc}$ reveals all his data on channel chc, $C\,[(V\sigma_{id_A})^{c_1,c_2}]$ has to do the same. In particular he has to publish an unsigned version of the ballot he later on submits (cf. Process 3.25 on page 108), hence he needs to output the ballot he gets back from the trusted device. The other data he needs to publish on the channel are the secret key of the voter, the public key of the device and a "random value" (like the one that is used to encrypt the vote). The keys have to be correct as they can be compared to the keys publish by the key distribution process, the "random value" can be anything as long as the frame is statically equivalent to one with a random value.

Let us now consider the equations we need to prove.

For the first equation (3.11) consider the processes V'. If we ignore all outputs on `c1` and inputs on `c2`, the process is equivalent to $V\sigma_{id_A}\sigma_{v_B}$ until the final three commands which depend on the inputs by the context. But there we can argue as above that the context $C[\_]$ will provide the correct inputs, i.e. twice `true` and finally the correctly signed ballot obtained from the trusted device.

For the second equation (3.12) we have consider the possible transitions. As before, we denote the left hand side P and the right hand side Q. Both sides start by publishing the keys, and then distributing them over restricted channels (yielding internal reductions). We do not detail this part for better readability. On both sides $C\,[(V\sigma_{id_A})^{c_1,c_2}]$ and $C\,[V']$ respectively publish the keys they receive. Then all processes synchronize, and the voters chose their random values (which the voter under attack reveals) and send their ballots to the trusted devices over the private channels. The trusted device answers with the signed and re-randomized ballot and the DVP. The voters check the DVP and then send the signed ballot to the collector. This also holds for the coerced voter as explained above. We obtain the following frame on the left hand side:

$$\begin{aligned}\phi_l = \nu\tilde{\mathtt{ch}}.\nu\tilde{r}.(&\{^{skva}/_{x_1}\} \mid \{^{pkda}/_{x_2}\} \mid \{^{pkc}/_{x_3}\} \mid \{^{r}/_{x_4}\} \mid \\ &\{^{(penc(v_A,pk(skc),f(r,r1)),sign(penc(v_A,pk(skc),f(r,r1)),skda),}\\ &\;\;dvp(penc(v_A,pk(skc),r),penc(v_A,pk(skc),f(r,r1)),r1,pk(skva)))}/_{x_5}\} \mid \\ &\{^{(penc(v_A,pk(skc),f(r,r1)),sign(sign(penc(v_A,pk(skc),f(r,r1)),skda),skva)}/_{x_6}\} \mid \\ &\{^{(penc(v_B,pk(skc),f(r_B,r1)),sign(sign(penc(v_B,pk(skc),f(r_B,r1)),skdb),skvb)}/_{x_7}\})\end{aligned}$$

where $\tilde{r}$ contains all the fresh random values created by the processes. Note that technically r is just a variable: the context can output a different value indistinguishable from the original r output by $(V\sigma_{id_A}\sigma_{v_A})^{chc}$, however this does not affect our reasoning since the frame has to remain statically equivalent to this one by (3.10). On the right hand side we have:

$$\begin{aligned}\phi_r = \nu\tilde{\mathtt{ch}}.\nu\tilde{r}.(\{^{skva}/_{x_1}\} \mid \{^{pkda}/_{x_2}\} \mid \{^{pkc}/_{x_3}\} \mid \{^{r}/_{x_4}\} \mid \\ \{^{(penc(v_B,pk(skc),f(r'',r1)),sign(penc(v_B,pk(skc),f(r'',r1)),skda),}\\ {}^{dvp(penc(v_A,pk(skc),r),penc(v_B,pk(skc),f(r'',r1)),r',skva))}/_{x_5}\} \mid \\ \{^{(penc(v_B,pk(skc),f(r'',r1)),sign(sign(penc(v_B,pk(skc),f(r'',r1)),skda),skva)}/_{x_6}\} \mid \\ \{^{(penc(v_A,pk(skc),f(r_B,r1)),sign(sign(penc(v_A,pk(skc),f(r_B,r1)),skdb),skvb)}/_{x_7}\})\end{aligned}$$

Here again r is just a variable, we denote by r'' the random value chosen by V'.

The above frames are statically equivalent as the context has no access to `skc` or `r1`, and in particular we have

$$checkdvp(third(x_5), penc(v_A, pk(skc), r), first(x_5), pk(skva) = ok$$

in both frames.

If the attacker forwards the ballots, the collectors check the signatures, synchronize and publish the result: A vote v_A and a vote v_B in both cases. If he does not forward the ballots or inputs other (and hence incorrectly signed) values, the collector(s) will block and cannot synchronize. □

Yet the protocol does not ensure simple Vote-Privacy against an inside attacker, showing that the cases *Insider* and *Outsider* are distinct. As acknowledged by the authors in their original paper [LBD+03], it is possible to copy votes. More precisely, an attacker can access the ballots on the bulletin board before the mixing takes place. He can easily verify which ballot belongs to which voter as they are signed by the voters themselves. He can remove the signature and use the ciphertext as an input to his trusted device. The trusted device will re-encrypt and sign it. This allows the attacker to construct a correct ballot which contains the same vote as the targeted honest voter. By submitting this ballot he obtains a different election outcome in both cases of the observational equivalence.

Theorem 26 *The protocol by Lee et al. does not respect $SwVP^{I,PO}$ (Swap-Vote-Privacy against an inside attacker in the Participation Only case).*

Proof In our model this can be seen as follows. We suppose that all authorities

are honest and that all key distribution channels are private. We show that

$$\nu\tilde{\mathtt{ch}}.(V_A\,\{^{a}/_{v}\}\,|V_B\,\{^{b}/_{v}\}\,|V_C^{c_1,c_2}|D_A|D_B|D_C|A) \not\approx_l \nu\tilde{\mathtt{ch}}.(V_A\,\{^{b}/_{v}\}\,|V_B\,\{^{a}/_{v}\}\,|V_C^{c_1,c_2}|D_A|D_B|D_C|A)$$

where

$$\begin{aligned}
\nu\tilde{\mathtt{ch}} &= \nu\mathtt{skcCh}.\nu\mathtt{pkcCh}.\nu\mathtt{votCh}.\\
&\quad \nu\mathtt{skvaCh}.\nu\mathtt{skvbCh}.\nu\mathtt{skvcCh}.\nu\mathtt{skdaCh}.\nu\mathtt{skdbCh}.\nu\mathtt{skdcCh}.\\
&\quad \nu\mathtt{pkvaCh}.\nu\mathtt{pkvbCh}.\nu\mathtt{pkvcCh}.\nu\mathtt{pkdaCh}.\nu\mathtt{pkdbCh}.\nu\mathtt{pkdcCh}\\
A &= (\mathtt{processK}|\mathtt{processC}|\mathtt{processC}|\mathtt{processC})\\
V_A &= \mathtt{processV}\,\{^{skvaCh}/_{skvCh}, {}^{pkdaCh}/_{pkdCh}\}\\
V_B &= \mathtt{processV}\,\{^{skvbCh}/_{skvCh}, {}^{pkdbCh}/_{pkdCh}\}\\
V_C &= \mathtt{processVc1c2}\,\{^{skvcCh}/_{skvCh}, {}^{pkdcCh}/_{pkdCh}\}\\
D_A &= \mathtt{processD}\,\{^{skdaCh}/_{skdCh}, {}^{pkvaCh}/_{pkvCh}\}\\
D_B &= \mathtt{processD}\,\{^{skdbCh}/_{skdCh}, {}^{pkvbCh}/_{pkvCh}\}\\
D_C &= \mathtt{processD}\,\{^{skdcCh}/_{skdCh}, {}^{pkvcCh}/_{pkvCh}\}
\end{aligned}$$

As before, we denote the left hand side P and the right hand side Q. For better readability we concentrate on the important steps. After key distribution, the honest voters will execute the protocol with their trusted device and eventually output the following messages on `ch1`:

$$\begin{aligned}
P \;\; &\rightarrow^* \ldots \rightarrow^* \xrightarrow{\mathtt{out}(ch1,x_1)} \xrightarrow{\mathtt{out}(ch1,x_2)}\\
&\nu r_1.\nu r_2.\nu r_3.\nu r_4.(P_1|\\
&\{^{(penc(a,pk(skt),f(r_1,r_2)),sign(sign(penc(a,pk(skt),f(r_1,r_2)),skda),skva))}/_{x_1}\}\,|\\
&\{^{(penc(b,pk(skt),f(r_3,r_4)),sign(sign(penc(b,pk(skt),f(r_3,r_4)),skdb),skvb))}/_{x_2}\})
\end{aligned}$$

$$\begin{aligned}
Q \;\; &\rightarrow^* \ldots \rightarrow^* \xrightarrow{\mathtt{out}(ch1,x_1)} \xrightarrow{\mathtt{out}(ch1,x_2)}\\
&\nu r_1.\nu r_2.\nu r_3.\nu r_4.(Q_1|\\
&\{^{(penc(b,pk(skt),f(r_1,r_2)),sign(sign(penc(b,pk(skt),f(r_1,r_2)),skda),skva))}/_{x_1}\}\,|\\
&\{^{(penc(a,pk(skt),f(r_3,r_4)),sign(sign(penc(a,pk(skt),f(r_3,r_4)),skdb),skvb))}/_{x_2}\})
\end{aligned}$$

The attacker can now target e.g. voter V_A and copy his vote. Note that he can identify which voter cast which ballot as they are signed and the keys publicly available. Through V_C he submits $penc(a, pk(skt),\ f(r_1, r_2))$ in the left hand case or $penc(b, pk(skt), f(r_1, r_2))$ in the right hand case to D_C and obtains $sign(penc(a, pk(skt), f(f(r_1, r_2), r_5)), skdc)$ and $penc(a, pk(skt), f(f(r_1, r_2), r_5)$ or $sign(penc(b, pk(skt), f(f(r_1, r_2), r_5)), skdc)$ and $penc(b, pk(skt), f(f(r_1, r_2), r_5)$ respectively, where r_5 is a fresh name (nonce). He can then sign the message and

Protocol	Security Level	Comments
Bingo Voting [BMQR07]	$SwCR^{I,FA}$	Trusted voting machine & polling booth
- with voter lists	$SwCR^{I,PO}$	Vulnerable to forced abstention attacks
Lee et al. [LBD+03]	$SwCR^{O,PO}$	Trusted randomizer, vulnerable to vote-copying
Okamoto [Oka96]	$SwRF^{I,PO}$	Uses trap-door commitments
- variant	$SwRF^{I,FA}$	Secure channel to trusted administrator
Fujioka et al. [FOO92]	$SwVP^{I,PO}$	Based on blind signatures
- variant	$SwVP^{I,FA}$	Secure channel to trusted administrator
Simple Voting Protocol, Example 12 on page 55	$SwVP^{O,PO}$	The running example, vulnerable to vote-copying

Table 3.1 – Results of the case studies

publish it on the bulletin board (i.e. send in on channel `ch1`):

$$(penc(a, pk(skt), f(f(r_1, r_2), r_5)),$$
$$sign(sign(penc(a, pk(skt), f(f(r_1, r_2), r_5)), skdc), skvc))$$

or in the right hand case

$$(penc(b, pk(skt), f(f(r_1, r_2), r_5)),$$
$$sign(sign(penc(b, pk(skt), f(f(r_1, r_2), r_5)), skdc), skvc))$$

The collector will then check the signatures - which are apparently correct - and publish the decrypted votes. On the left hand side, we will obtain two votes a and one vote b, on the right hand side one vote for a and two votes for b. Thus the frames are not statically equivalent, hence both sides are not bisimilar. □

Additionally, this protocol is not secure against forced-abstention attacks as the ballots on the bulletin board are signed by the voters. The attacker can thus easily verify if a voter voted or not.

§ 3.3.5.5. Summary. In this section we presented a taxonomy of privacy in eVoting protocols. Our taxonomy is based on three independent dimensions: Communication between the voter and the attacker, inside or outside attacker, and security against forced abstention attacks. We formalized a model for voting protocols and the different notions in the Applied π-Calculus.

To illustrate that the different dimensions of our taxonomy correspond to

different properties of existing protocols, we presented several case studies. The results of these case studies are summed up in Table 3.1 on the facing page and Figure 3.1 on page 69.

3.4 Defining Privacy for Weighted Votes

The definitions presented in the previous section are based on the idea of swapping votes. Yet these definitions are unsuitable for some situations, for example in companies where votes are weighted according to the proportion of shares held by each shareholder. Consider the following example: Alice owns 50% of the stocks, and Bob and Carol each hold 25%. The cases where Alice and Bob swap votes are now easily distinguishable if for example Carol votes "yes" all the time, as the result of the vote is different: 75% vs. 50% vote for "yes". Note that there are still situations where privacy is ensured in the sense that different situations give the same result. The last outcome (50% yes, 50% no) could - for example - also be announced if Alice votes "yes" and Bob and Carol vote "no". Protocols supporting vote weights have been proposed, for example Eliasson and Zúquete [EZ06] developed a voting system supporting vote weights based on REVS [JZF03], which itself is based on the protocol by Fujioka et al. [FOO92].

To address this issue, we now present a generalization of previous notions that takes weighted votes into account. Instead of requiring two executions where voters swap votes to be bisimilar, we require two executions to be bisimilar if they publish the same result, independent of the mapping between voters and votes. We analyze the relationship of our notion to the previous swap-based ones and give precise conditions for formally proving the equivalence between them. We use a variant of the protocol by Eliasson and Zúquete [EZ06] as a case study for our definition, and provide a partially automated proof using ProVerif.

Our privacy definition is based on the observation that - as the result of the vote is always published - some knowledge about the voter's choices can always be inferred from the outcome. The classical example is the case of a unanimous vote where the contents of all votes are revealed just by the result. Yet - as already discussed in the above - there can also be other cases where some of the votes can be inferred from the result, in particular in the case of weighted votes. If for example Alice holds 66% of the shares and Bob 34%, both votes are always revealed when announcing the result: If one option gets 66% and the other 34%, it is clear which one was chosen by Alice or Bob. However, if we have a different distribution of the shares (e.g. 50%, 25% and 25% as above), some privacy is still possible as there are several situations with the same result. Thus our main idea: If two instances of a protocol give the same result, an attacker should not be able to distinguish them. Note that this includes the previous definitions where votes

are swapped, if this gives the same result.

3.4.1 — Formal Definition

To express this formally, we need to define the result of an election. As defined above, we suppose that the result is always published on a special channel res. The following definition allows us to hide all channels except for a specified channel c, which we can use for example to reason about the result on channel res.

Definition 22 ($P|_c$) *Let $P|_c = \nu\tilde{ch}.P$ where $\tilde{ch}$ are all channels except for c, i.e. we hide all channels except for c.*

Now we can formally define our privacy notion: If two instances of a protocol give the same result, they should be bisimilar.

Definition 23 (Vote-Privacy (VP)) *A voting protocol ensures* Vote-Privacy (VP) *if for any two instances $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_l}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ and $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ we have*

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA \approx_l VPB$$

where VPA^H and VPB^H are the honest voting processes corresponding to VPA and VPB respectively.

Note that the test if the two instances give the same result is defined on the honest voting processes, but the bisimilarity ensuring privacy is then required to hold on the "normal" voting processes where some authorities can be dishonest. This distinction is necessary as otherwise the hiding of all channels but the result ($VPA^H|_{res}$ and $VPB^H|_{res}$) in the test could produce instances that cannot actually terminate as some authorities are not present, and the attacker cannot simulate them as he does not have access to the channels.

A simple interpretation of this definition is that everything apart from the result on channel res has to remain private. This obviously relies heavily on the notion of "result" and the modeling of the protocol. Typically the result will only contain the sum of all votes, which corresponds to a simple and intuitive understanding of privacy.

Some protocols may leak some additional information, for example the number of ballots on the bulletin board. For instance in the protocol by Juels et al. [JCJ05] voters can post fake ballots. In this case, the above definition of the result may lead to a too restrictive privacy notion, since two situations with the same votes but a different number of fakes are required to be bisimilar. To address this issue, we can include the number of ballots in the result if we want to accept the additional leakage. This gives very fine-grained control about the level of privacy we want to model.

Note that if the link between a voter and his vote is also published as part of the result on channel res, our definition of privacy may be true although this probably does not correspond to the intuitive understanding of privacy. This is however coherent within the model since everything apart from the result is private; simply the result itself leaks too much information.

Moreover, note that this approach to define privacy is not only limited to voting, but can also be applied e.g. on auction protocols (see Section 4.3.4), or other types of multi-party computation where several parties jointly compute a public result based on private inputs.

3.4.2 — Example: A Variant of FOO

To show that our definition is applicable in practice, we now discuss an example: Eliasson and Zúquete [EZ06] proposed an implementation of a voting system supporting vote weights based on REVS [JZF03], which itself is based on the protocol by Fujioka et al. [FOO92] discussed in Section 3.3.5.1.

§ 3.4.2.1. Adding Vote Weights. In [EZ06] Eliasson and Zúquete discuss several possibilities on how to implement weights in this protocol:

— including the weight in the vote (which requires trusting the voter for correctness or zero-knowledge proofs to verify the weight)

— using different keys when the vote is signed by the administrator, where each key corresponds to a different weight

— using multiple ballots per voter, i.e. if for example voter A holds 70% and voter B 30% of the shares, voter A sends seven and voter B three ballots.

We implemented the latter variant in the Applied π-Calculus as a case study.

Note that there is also a variant of Helios supporting vote weights [ADMPQ09], using a fourth approach: the voters vote normally, and the authorities multiply the vote with its weight before tallying, exploiting the homomorphic property of the encryption.

§ 3.4.2.2. Model and Analysis. We use the following equational theory and model:

$$\begin{aligned} \mathtt{open}(\mathtt{commit}(m,r),r) &= m \\ \mathtt{checksign}(\mathtt{sign}(m,sk),\mathtt{pk}(sk)) &= m \\ \mathtt{unblind}(\mathtt{blind}(m,r),r) &= m \\ \mathtt{unblind}(\mathtt{sign}(\mathtt{blind}(m,r),sk),r) &= \mathtt{sign}(m,sk) \end{aligned}$$

The *administrator* (Listing 3.30 on the following page) receives his private key and the public key of a legitimate voter. When receives the blinded commitment, he checks the signature, signs, and sends the result back.

```
processA =
  in(ch1,m1).
  let (pubkeyv,sig,blindedvote) = m1 in
  if pubkeyv = pkv && blindedvote = checksign(sig,pkv) then
   out(ch2,sign(blindedvote,ska))
```

Listing 3.30 – The administrator process

```
processC =
 synch 1.
 in(ch3,(m3,m4)).
 if checksign(m4,pka) = m3 then
  νl.out(ch4,(l,m3,m4)).
  in(ch5, (l',rand)).
  if l = l' then
   let voteV = open(m3,rand) in
   out(res,voteV)
```

Listing 3.31 – The collector process

```
processV =
  for (i = 1 to w)
    νb_i.νr_i.
    let commitedvote_i = commit(v,r_i) in
    let blindedvote_i = blind(commitedvote_i,b_i) in
    out(ch1,(pk(skv),sign(blindedvote_i,skv),blindedvote_i))
  for (i = 1 to w)
    in(ch2, m_i).
    let result_i = checksign(m_i,pka) in
    if result_i = blindedvote_i then
    let signedvote_i = unblind(m_i,b_i) in
  sync 1.
  for (i = 1 to w)
    out(ch3,(commitedvote_i,signedvote_i)).
    in(ch4,(l,=commitedvote_i,=signedvote_i))
  sync 2.
  for (i = 1 to w)
    out(ch5,(l_i,r_i))
```

Listing 3.32 – The voting process

The *collector* (Listing 3.31 on the preceding page) receives the administrator's public key, which he then uses to verify the signature on incoming commitments. If the signature is correct, he creates a new bounded name l (the number in the list) and sends it together with the signed commitment back to the voter. The voter then reveals his randomness, which the collector uses to open the commitment.

The *voter's process* (Listing 3.32 on the facing page) votes following the protocol by FOO, yet for several ballots according to his weight w: He first creates the blinded commitments, signs them and sends them to the administrator. He unblinds the answer and sends it to the collector, and finally reveals the randomness.

Using a manual proof we can show that

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow \sum_{i=1}^{n} v_i^A * w_i = \sum_{i=1}^{n} v_i^B * w_i. \tag{3.13}$$

Proof To show that (3.13) holds, we use a proof by contradiction. Suppose

$$\sum_{i=1}^{n} v_i^A * w_i \neq \sum_{i=1}^{n} v_i^B * w_i \tag{3.14}$$

We will show that this implies $VPA^H|_{res} \not\approx_l VPB^H|_{res}$ by showing that there exists an execution which allows the attacker to see for all i w_i times a message v_i^X on channel res. Hence he can compare messages and distinguish $VPA^H|_{res}$ and $VPB^H|_{res}$.

Consider two processes

$$\begin{aligned}
VPA^H = & \mid_{1\le i\le n} (\texttt{processV}\,\{^{v_i^A}/_{v}\}\,\{^{w_i}/_{w}\}\,\{^{pk(ska)}/_{pka}\}\,\{^{skv_i}/_{skv}\}) \\
& \quad \mid_{1\le i\le n} (\mid_{1\le k\le w_i} \texttt{processC}\,\{^{pk(ska)}/_{pka}\}) \\
& \quad \mid_{1\le i\le n} (\mid_{1\le k\le w_i} \texttt{processA}\,\{^{pk(skv_i)}/_{pkv}\}) \\
VPB^H = & \mid_{1\le i\le n} (\texttt{processV}\,\{^{v_i^B}/_{v}\}\,\{^{w_i}/_{w}\}\,\{^{pk(ska)}/_{pka}\}\,\{^{skv_i}/_{skv}\}) \\
& \quad \mid_{1\le i\le n} (\mid_{1\le k\le w_i} \texttt{processC}\,\{^{pk(ska)}/_{pka}\}) \\
& \quad \mid_{1\le i\le n} (\mid_{1\le k\le w_i} \texttt{processA}\,\{^{pk(skv_i)}/_{pkv}\})
\end{aligned}$$

In both cases, `processC` cannot synchronize and is blocked. The voter processes can output w_i times $(\texttt{pk}(\texttt{skv}_\texttt{i}), \texttt{sign}(\texttt{blindedvote}_\texttt{i}, \texttt{skv}_\texttt{i}), \texttt{blindedvote}_\texttt{i})$ (their blinded and signed vote) on channel `ch1`. The administrator replies with the correctly signed votes, which allows the voters to successfully verify and unblind the signed ballots. Then they will synchronize and output the $(\texttt{commitedvote}_\texttt{i}, \texttt{signedvote}_\texttt{i})$ on channel `ch3`. The `processC` processes can then receive the messages, verify the signatures and output $(\texttt{l}, \texttt{m3}, \texttt{m4})$, i.e.

($\mathtt{l}, \mathtt{commitedvote_i}, \mathtt{signedvote_i}$) on channel `ch4`. The voters will then verify if the messages correspond to their votes, synchronize and reveal the random values of the commitments. The collectors can open the commitments and will publish all votes on channel `res`. Hence there is an execution that reveals all votes and their weights in clear to the intruder, hence using (3.14), $VPA^H|_{res} \not\approx_l VPB^H|_{res}$. □

Using a python script available online [Dre13] that generates all cases to check given the number of voters and the discrete weight distribution, we can use ProVerif to then establish (3.15) for a given weight distribution. Although this technique allows to conclude that the protocol ensures (VP) given a certain distribution of the weights, ProVerif is unable to prove a general result for all possible distributions automatically, as we have infinitely many possible distributions and hence infinitely many cases to check.

$$\sum_{i=1}^{n} v_i^A * w_i = \sum_{i=1}^{n} v_i^B * w_i \Rightarrow VPA \approx_l VPB \tag{3.15}$$

3.4.3 — Link to Existing Definitions

To establish the relationship of our definition and the previous taxonomy in Section 3.3, we need to formally characterize their difference. Intuitively the swap-based definition assumes that swapping two votes will not change the result. This can be formalized as follows: If two instances of the protocol with the same voters give the same result, then the votes are a permutation of each other, and vice versa. This precludes weighted votes, thus the name "Equality of Votes".

Definition 24 (Equality of Votes (EQ)) *A voting protocol respects* Equality of Votes (EQ) *if for any two voting processes* $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ *and* $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *where no* $\sigma_{v_i^A}$ *or* $\sigma_{v_i^B}$ *makes a voter abstain, we have*

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Leftrightarrow \exists\pi : \forall i : \sigma_{v_i^B} = \sigma_{v^A_{\pi(i)}},$$

where π *is a permutation.*

Note that here we exclude abstention, i.e. the result can for example include a list of all abstaining voters. We also define a stronger notion that includes abstention. In that case, the result has to remain unchanged even if different voters abstain.

Definition 25 (Equality of Abstention (EQA)) *A voting protocol respects* Equality of Abstention (EQA) *if for any two voting processes* $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ *and* $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1$

$| \ldots | A_l)$, *we have*

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Leftrightarrow \exists \pi : \forall i : \sigma_{v_i^B} = \sigma_{v_{\pi(i)}^A},$$

where π *is a permutation.*

Obviously this is stronger than (EQ), i.e. any protocol respecting (EQA) also respects (EQ).

We can apply these definitions on the case studies we presented before, consider for example FOO.

Theorem 27 *FOO ensures Equality of Votes (EQ).*

Proof Suppose we have two voting processes VPA and VPB and there exists a permutation $\pi : \forall i : \sigma_{v_i^B} = \sigma_{v_{\pi(i)}^A}$. We have to show that $VPA^H|_{res} \approx_l VPB^H|_{res}$. As the protocol ensures $SwVP^{I,PO}$ (and hence also $SwVP^{O,PO}$) and we can write any permutation as a sequence of simple permutations, we have $VPA \approx_l VPB$, which gives $VPA^H|_{res} \approx_l VPB^H|_{res}$ as labeled bisimilarity is closed under the application of contexts (here the inclusion of the missing authorities and the restriction to the channel res).

Suppose we have two voting processes VPA and VPB with $VPA^H|_{res} \approx_l VPB^H|_{res}$. Assume that there exists no permutation π such that $\forall i : \sigma_{v_i^B} = \sigma_{v_{\pi(i)}^A}$, hence there exists a vote v such that the number v_i^A with $v_i^A = v$ is different of the number of v_i^B with $v_i^B = v$, i.e.

$$\sum_{1 \le i \le n, v_i^A = v} 1 \neq \sum_{1 \le i \le n, v_i^B = v} 1$$

We show that this allows an attacker to distinguish $VPA^H|_{res}$ and $VPB^H|_{res}$ which contradicts $VPA^H|_{res} \approx_l VPB^H|_{res}$. Consider the following voting processes:

$$\begin{aligned} VPA^H = & \mid_{1 \le i \le n} (\texttt{processV}\,\{^{v_i^A}/_{v}\}\,\{^{pk(ska)}/_{pka}\}\,\{^{skv_i}/_{skv}\}\mid \\ & \texttt{processC}\,\{^{pk(ska)}/_{pka}\}\mid \texttt{processA}\,\{^{pk(skv_i)}/_{pkv}\}) \\ VPB^H = & \mid_{1 \le i \le n} (\texttt{processV}\,\{^{v_i^B}/_{v}\}\,\{^{pk(ska)}/_{pka}\}\,\{^{skv_i}/_{skv}\}\mid \\ & \texttt{processC}\,\{^{pk(ska)}/_{pka}\}\mid \texttt{processA}\,\{^{pk(skv_i)}/_{pkv}\}) \end{aligned}$$

The voter processes can output $(\texttt{pk}(\texttt{skv}), \texttt{sign}(\texttt{blindedvote}, \texttt{skv}), \texttt{blindedvote})$ (their blinded and signed vote) on channel `ch1`. The administrator then replies with the signed votes, which allows the voters to successfully verify and unblind the signed ballots. Then they synchronize and output the $(\texttt{commitedvote}, \texttt{signedvote})$ on channel `ch3`. The collectors `processC` can then receive the messages, verify the signatures and output list of accepted ballots on channel `ch4`. The voters then reveal their random values `r`. The collectors can finally open and publish the

votes `voteV` on channel `res`. Hence there is an execution that reveals all votes in clear to the intruder, thus he can count the number of occurrences of each choice, which gives $VPA^H|_{res} \not\approx_l VPB^H|_{res}$. □

We can do a similar proof for the protocol by Okamoto.

Theorem 28 *The protocol by Okamoto ensures Equality of Votes (EQ).*

Proof Suppose we have two voting processes VPA and VPB and there exists a permutation $\pi : \forall i : \sigma_{v_i^B} = \sigma_{v^A_{\pi(i)}}$. We have to show that $VPA^H|_{res} \approx_l VPB^H|_{res}$. As the protocol ensures $SwVP^{I,PO}$ (and hence also $SwVP^{O,PO}$) and we can write any permutation as a sequence of simple permutations, we have $VPA \approx_l VPB$, which gives $VPA^H|_{res} \approx_l VPB^H|_{res}$ as labeled bisimilarity is closed under the application of contexts (here the inclusion of the missing authorities and the restriction to the channel res).

Suppose we have two voting processes VPA and VPB with $VPA^H|_{res} \approx_l VPB^H|_{res}$. Assume that there exists no permutation π such that $\forall i : \sigma_{v_i^B} = \sigma_{v^A_{\pi(i)}}$, hence there exists a vote v such that the number v_i^A with $v_i^A = v$ is different of the number of v_i^B with $v_i^B = v$, i.e.

$$\sum_{1 \le i \le n, v_i^A = v} 1 \neq \sum_{1 \le i \le n, v_i^B = v} 1$$

We show that this allows an attacker to distinguish $VPA^H|_{res}$ and $VPB^H|_{res}$ which contradicts $VPA^H|_{res} \approx_l VPB^H|_{res}$. Consider the following voting processes:

$$VPA^H = \nu \mathtt{chT}.\left(\left|_{1 \le i \le n} (\mathtt{processV}\left\{{}^{v_i^A}/_{v}\right\}\left\{{}^{pk(ska)}/_{pka}\right\}\left\{{}^{skv_i}/_{skv}\right\} \mid \mathtt{processT}\right)\right) \mid \mathtt{processC}\left\{{}^{pk(ska)}/_{pka}\right\}) \mid \mathtt{processA}\left\{{}^{pk(skv_i)}/_{pkv}\right\}$$

$$VPB^H = \nu \mathtt{chT}.\left(\left|_{1 \le i \le n} (\mathtt{processV}\left\{{}^{v_i^B}/_{v}\right\}\left\{{}^{pk(ska)}/_{pka}\right\}\left\{{}^{skv_i}/_{skv}\right\} \mathtt{processT}\right)\right) \mid \mathtt{processC}\left\{{}^{pk(ska)}/_{pka}\right\}) \mid \mathtt{processA}\left\{{}^{pk(skv_i)}/_{pkv}\right\}$$

In both cases, `processT` cannot synchronize and is blocked. The voter processes can output $(\mathtt{pk}(\mathtt{skv}), \mathtt{sign}(\mathtt{blindedvote}, \mathtt{skv}), \mathtt{blindedvote})$ (their blinded and signed vote) on channel `ch1`. The administrator then replies with the signed votes, which allows the voters to successfully verify and unblind the signed ballots. Then they synchronize and output the $(\mathtt{commitedvote}, \mathtt{signedvote})$ on channel `ch3`, and output $(\mathtt{v}, \mathtt{r}, \mathtt{commitedvote})$ on the private channel `chT` with the timeliness member. The `processT` processes can then receive the messages, synchronize,

verify the signatures and output the votes `vt` on channel `res`. Hence there is an execution that reveals all votes in clear to the intruder, thus he can count the number of occurrences of each choice, which gives $VPA^H|_{res} \not\approx_l VPB^H|_{res}$. □

Similarly Bingo Voting ensures Equality of Abstention as votes are not weighted and abstaining voters remain anonymous.

Theorem 29 *Bingo Voting ensures Equality of Abstention (EQA).*

Proof Suppose we have two voting processes VPA and VPB and there exists a permutation $\pi : \forall i : \sigma_{v_i^B} = \sigma_{v^A_{\pi(i)}}$. We have to show that $VPA^H|_{res} \approx_l VPB^H|_{res}$. As the protocol ensures $SwVP^{I,FA}$ and we can write any permutation as a sequence of simple permutations, we have $VPA \approx_l VPB$, which gives $VPA^H|_{res} \approx_l VPB^H|_{res}$.

Suppose we have two voting processes VPA and VPB with $VPA^H|_{res} \approx_l VPB^H|_{res}$. Assume that there exists no permutation π such that $\forall i : \sigma_{v_i^B} = \sigma_{v^A_{\pi(i)}}$, hence there exists w.l.o.g. a vote v [10] such that the number v_i^A with $v_i^A = v$ is different of the number of v_i^B with $v_i^B = v$, i.e.

$$\sum_{1\leq i\leq n, v_i^A = v} 1 \neq \sum_{1\leq i\leq n, v_i^B = v} 1 \tag{3.16}$$

We show that this allows an attacker to distinguish $VPA^H|_{res}$ and $VPB^H|_{res}$ which contradicts $VPA^H|_{res} \approx_l VPB^H|_{res}$. Consider the following voting processes:

$$\begin{aligned} VPA^H = \nu\mathrm{privChM}_1 \ldots \nu\mathrm{privChM}_n.\nu\mathrm{privChRM}_1 \ldots \nu\mathrm{privChRM}_n.\\ \nu\mathrm{privChR}_1 \ldots \nu\mathrm{privChR}_n.\Big(\mathtt{processM}\,\{^n/_k\}\\ \Big|_{1\leq i\leq n}\big(\mathtt{processV}\,\{^{v_i^A}/_v\}\,\sigma_{id_i}\,\{^{\mathrm{privChM}_i}/_{\mathrm{privChM}}\}\,\{^{\mathrm{privChR}_i}/_{\mathrm{privChRNG}}\}\big)\\ \Big|_{1\leq i\leq n}\big(\mathtt{processRNG}\,\{^{\mathrm{privChRM}_i}/_{\mathrm{privChM}}\}\,\{^{\mathrm{privChR}_i}/_{\mathrm{privChV}}\}\big)\Big) \end{aligned}$$

[10] If we have a different number of abstaining voters, this also implies that we have a candidate which has a different number of votes on both sides.

and

$$VPB^H = \nu\mathrm{privChM}_1 \ldots \nu\mathrm{privChM}_n.\nu\mathrm{privChRM}_1 \ldots \nu\mathrm{privChRM}_n.$$
$$\nu\mathrm{privChR}_1 \ldots \nu\mathrm{privChR}_n.\Big(\mathtt{processM}\,\{^n/_k\}$$
$$\Big|_{1\le i\le n}\big(\mathtt{processV}\,\{^{v_i^B}/_v\}\,\sigma_{id_i}\,\{^{\mathrm{privChM}_i}/_{\mathrm{privChM}}\}\,\{^{\mathrm{privChR}_i}/_{\mathrm{privChRNG}}\}\big)$$
$$\Big|_{1\le i\le n}\big(\mathtt{processRNG}\,\{^{\mathrm{privChRM}_i}/_{\mathrm{privChM}}\}\,\{^{\mathrm{privChR}_i}/_{\mathrm{privChV}}\}\big)\Big)$$

In both cases the voting machine process `processM` generates the commitments $\mathtt{commit}((\mathtt{n_{i,j}}, \mathtt{p_j}), \mathtt{r_{i,j}})$ and publishes them on channel `ch`. The random generators `processRNG` generate a fresh nonce and output it on the private channels to the voter and the voting machine. The voter processes outputs his vote `v` on channel $\mathtt{privChM}_i$. The voting machine receives both and outputs the receipts $\mathtt{receipt}_j$ to the voters on channel $\mathtt{PrivChM}_i$. After all votes have been received and the receipts have been send, the voting machine can synchronize and output the result, i.e. the votes `v` in clear on channel `res`. Hence there is an execution that reveals all votes in clear to the intruder, thus he can count the number of occurrences of each choice which gives $VPA^H|_{res} \not\approx_l VPB^H|_{res}$. □

Using (EQ) and (EQA) we can now establish the formal link between (VP) and the previous privacy definitions in our taxonomy.

Theorem 30 *We have:*

— *If a protocol respects (EQ), then (VP) and* $(SwVP^{O,PO})$ *are equivalent.*

— *If a protocol respects (EQA), then (VP) and* $(SwVP^{O,FA})$ *are equivalent.*

Proof We only prove the first statement, the second is analogous. We simply need to add the case where σ_{v_B} makes the voter abstain, which is however covered by (EQA), hence the resulting proof is the same.

— Suppose the protocol respects (VP). We have to show that it respects $SwVP^{O,PO}$, i.e. that for any voting process VP and for all votes σ_{v_A} and σ_{v_B} where σ_{v_B} and σ_{v_A} does not make a voter abstain we have

$$\begin{aligned} VPA := \quad & VP_{\{A,B\}}\,[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}|0] \\ & \approx_l \\ & VP_{\{A,B\}}\,[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}|0] \quad =: VPB \end{aligned}$$

Consider

$$\begin{aligned} VPA^H := \quad & VP^H_{\{A,B\}}\,[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}] \\ & \approx_l \\ & VP^H_{\{A,B\}}\,[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}] \quad := VPB^H \end{aligned}$$

Obviously the votes on the right hand side are a permutation of the votes of the left hand side and as the protocol ensures Equality of Votes we have

$$VPA^H|_{res} \approx_l VPB^H|_{res}$$

As the protocol respects Vote-Privacy, we conclude

$$VPA \approx_l VPB$$

— Suppose the protocol respects $SwVP^{O,PO}$ and we want to show that it respects (VP), i.e. for any voting process $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A}|\ldots|V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ and $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ we have

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA \approx_l VPB$$

Suppose $VPA^H|_{res} \approx_l VPB^H|_{res}$. As the protocol respects (EQ), we know that there exists a permutation π such that $\forall i : \sigma_{v_i^A} = \sigma_{v^B_{\pi(i)}}$. Using the fact that $SwVP^{O,PO}$ allows us to permute any two votes and that any permutation can be expressed as a sequence of simple permutations, we obtain $VPA \approx_l VPB$. Note that if we only have one voter, he has to vote the same on both sides and the equivalence is trivially true. □

It is easy to see that for protocols violation (EQ) the equivalence does not necessary hold: If a protocol uses weighted votes (e.g. Alice 66%, Bob 34%), it may satisfy (VP), but not ($SwVP^{O,PO}$).

Similarly, assume a simple yes/no referendum, where the proposition is accepted if at least half of the voters are in favor of the proposition. If however at the same time the ballots on the bulletin board allow to compute the number of "yes" and "no" votes, such a protocol may ensure ($SwVP^{O,PO}$) – if the ballots cannot be linked to the voters –, but not (VP) because two instances with a different outcome based on the ballots will have the same "result" on res: for example the instances where Alice votes "no" and Bob votes "yes" gives the same result as the instance where both vote "yes". Note that such a protocol would contradict (EQ) because we have instances where the votes are not a permutation of each other, but still give the same result.

3.4.4 — Including Corrupted Voters

Until now, these definitions did not include corrupted voters. This can be addressed using the following definitions.

Definition 26 (Vote-Privacy with a single corrupted voter (VP^{SC})) *A voting protocol ensures* Vote-Privacy with a single corrupted voter (VP^{SC}) *if for any two instances* $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ *and* $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$*, and a subset* $J \subseteq \{1, \ldots, n\}$*, such that* $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ *and* $|J| \leq 1$ *we have*

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA_J\left[\,|_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_J\left[\,|_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

where VPA_J *and* VPB_J *are contexts equal to* VPA *and* VPB *with a hole for the voters in* J *respectively (cf. Definition 16 on page 58).*

Again, we can establish a link to the existing definition.

Theorem 31 *We have:*

— *If a protocol respects (EQ), then (VP^{SC}) and* $SwVP^{I,PO}$ *are equivalent.*

— *If a protocol respects (EQA), then (VP^{SC}) and* $SwVP^{I,FA}$ *are equivalent.*

Proof We only consider the first case, the second case is analogous.

— Suppose the protocol respects (VP^{SC}). We have to show that it respects ($SwVP^{I,PO}$), i.e. for any voting process VP and for all votes σ_{v_A} and σ_{v_B} where σ_{v_B} and σ_{v_A} do not make a voter abstain we have

$$\begin{aligned} VPA := \; & VP_{\{A,B,C\}}\,[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}|(V\sigma_{id_C})^{c_1,c_2}] \\ & \approx_l \\ & VP_{\{A,B,C\}}\,[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}|(V\sigma_{id_C})^{c_1,c_2}] \quad =: VPB \end{aligned}$$

Consider

$$\begin{aligned} VPA^H := \; & VP^H_{\{A,B,C\}}\,[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}|V\sigma_{id_C}\sigma_{v_C}] \\ & \approx_l \\ & VP^H_{\{A,B,C\}}\,[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}|V\sigma_{id_C}\sigma_{v_C}] \quad := VPB^H \end{aligned}$$

Obviously the votes on the right hand side are a permutation of the votes of the left hand side. As the protocol ensures Equality of Votes we have

$$VPA^H|_{res} \approx_l VPB^H|_{res}$$

As the protocol respects (VP^{SC}) we conclude with $J = \{C\}$, $|J| \leq 1$, that

$$VPA \approx_l VPB$$

Note that technically the names of the channels of the corrupted voter are different, however labeled bisimilarity is closed under α-renaming.

— Suppose the protocol respects ($SwVP^{I,PO}$) and we want to show that it respects (VP^{SC}), i.e. for two instances $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ and $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$, and a subset $J \subseteq \{1, \ldots, n\}$ such that $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ and $|J| \leq 1$ we have

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA_J\left[\mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_J\left[\mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

Suppose $VPA^H|_{res} \approx_l VPB^H|_{res}$. As the protocol respects (EQ), we know that there exists a permutation π such that $\forall i : \sigma_{v_i^A} = \sigma_{v_{\pi(i)}^B}$. Additionally we know that $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$. Using the fact that ($SwVP^{I,PO}$) allows us to permute any two votes of honest voters in the presence of one corrupted voter which does not change its vote, and that any permutation (even the identity) can be expressed as a sequence of simple permutations, we obtain

$$VPA_J\left[\mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_J\left[\mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

(again after renaming).

Note that if we have no honest voter, the equivalence is trivially true (syntactical equality). Similarly if we have only one honest voter, he has to vote the same on both sides and the equivalence is again trivially true. Note also that if we do not have a corrupted voter, we can still conclude as (VP^{SC}) implies (VP). □

Naturally we can also consider multiple corrupted voters.

Definition 27 (Vote-Privacy with multiple corrupted voters (VP^{MC}))
A voting protocol ensures Vote-Privacy with multiple corrupted voters (VP^{MC}) *if for any two instances* $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ *and* $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$, *and a subset* $J \subseteq \{1, \ldots, n\}$, $J \neq \{1, \ldots, n\}$ *if* $n > 1$, *such that* $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ *we have*

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA_J\left[\mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_J\left[\mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

where VPA_J and VPB_J are contexts equal to VPA and VPB with a hole for the voters in J respectively (cf. Definition 16 on page 58).

Note that each corrupted bidder j has his own channel pair c_{1j}, c_{2j} to interact with the attacker. This avoids confusion, in particular in the Coercion-Resistance definitions below.

With these definitions we obtain a hierarchy of privacy notions.

Theorem 32 (Hierarchy of Privacy Definitions) *We have:*

— *If a protocol respects (VP^{MC}), then it also respects (VP^{SC}).*

— *If a protocol respects (VP^{SC}), then it also respects (VP).*

Proof This is a consequence of the definitions: (VP) is a special case of (VP^{SC}) with $J = \emptyset$, and (VP^{SC}) is a special case of (VP^{MC}) with $|J| \leq 1$. □

As shown in the previous section, inside and outside attackers are in general not the same as an inside attacker can for example employ vote-copying attacks. However we can show that under some reasonable assumptions on the protocol (VP), (VP^{SC}) and (VP^{MC}) coincide.

To prove this, we define the notion of a "generalized voting process" which is like a voting process, but some of the voters might be coerced or corrupted.

Definition 28 (Generalized Voting Process) *A* Generalized Voting Process *is a voting process VP with variables for the voter's processes that can either be a "normal" honest voter, a voter communicating with the intruder, or a corrupted voter, i.e. $VP = \nu\tilde{n}.(V_1|\ldots|V_n|A_1|\ldots|A_l)$ where $V_i \approx_l V\sigma_{id_i}\sigma_{v_i}$ or $V_i^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i}$ or $V_i \approx_l (V\sigma_{id_i}\sigma_{v_i})^{c_{1i},c_{2i}}$.*

The next definition captures the key property required for our proof.

Definition 29 (Modularity (Mod)) *A voting protocol is* modular (Mod) *if for any honest, coerced or corrupted voters $V_1, \ldots, V_n$ (i.e. with $V_i \approx_l V\sigma_{id_i}\sigma_{v_i}$ or $V_i^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i}$ or $V_i \approx_l (V\sigma_{id_i}\sigma_{v_i})^{c_{1i},c_{2i}}$) and the corresponding generalized voting processes $VPA = \nu\tilde{n}_A.(V_1 \mid \ldots \mid V_k \mid A_{1_A} \mid \ldots \mid A_{l_A})$, $VPB = \nu\tilde{n}_B.(V_{k+1} \mid \ldots \mid V_n \mid A_{1_B} \mid \ldots \mid A_{l_B})$ and $VP = \nu\tilde{n}.(V_1 \mid \ldots \mid V_n \mid A_1 \mid \ldots \mid A_l)$ we have*

$$VP \approx_l VPA|VPB.$$

Note that this can be applied in both directions, i.e. we can compose and decompose instances with different voters in a transparent way. In particular we can add voters by composing with an instance containing these voters, or decompose an instance into two instances and reason about both instances independently.

Consider the following example illustrating how Modularity captures that certain parts of a voting protocol are independent: imagine a protocol where

in order to escape coercion the voters can claim that a certain ballot on the bulletin board is their ballot, but this ballot was actually prepared by some honest authority to allow the voters to create a fake receipt. If we suppose that this ballot exists only once no matter how many voters are attacked, it would be enough for a single voter to fake his receipt. However we cannot compose two instances with one attacked voter each, as they would use the same fake ballot which would be noticeable for the attacker. Hence the above definition also captures the fact that faking the receipt to escape coercion can be done by each voter independently.

Similarly, a modular protocol is secure against inside attackers or corrupted voters since the different parts of the protocol are independent, as we now show.

Theorem 33 *If a protocol is modular and finite (i.e. any instance or voting process is finite), (VP) and (VP^{MC}) are equivalent.*

Proof By Theorem 32 a protocol ensuring (VP^{MC}) also ensures (VP). We now show that if a protocol is modular, finite and respects (VP), then it also respects (VP^{MC}).

We need to show that for any two instances $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ and $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$, and a subset $J \subseteq \{1,\ldots,n\}$, $J \neq \{1,\ldots,n\}$ if $n > 1$, such that $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ we have

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

As the protocol ensures (VP), we have that for any two instances $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ and $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ we have

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA \approx_l VPB$$

Assume we have two instances VPA and VPB as above, and a subset $J \subseteq \{1,\ldots,n\}$ such that $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ with $VPA^H|_{res} \approx_l VPB^H|_{res}$. Then $VPA \approx_l VPB$. Consider now

$$VPA_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]|VPA$$

By $VPA \approx_l VPB$ we have

$$VPA_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]|VPA \approx_l VPA_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]|VPB$$

As the protocol is modular, we can decompose VPB, and recompose with the corrupted voters to obtain

$$VPA_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]|VPB$$
$$\approx_l \nu\tilde{n}.\left(V\sigma_{id_1}\sigma_{v_1^A}|\ldots\mid_{j\in J}(V\sigma_{id_j}\sigma_{v_j^B})|\ldots|V\sigma_{id_n}\sigma_{v_n^A}|A_1|\ldots|A_l\right)|$$
$$VPB_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

Since $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ we have

$$\nu\tilde{n}.\left(V\sigma_{id_1}\sigma_{v_1^A}|\ldots\mid_{j\in J}(V\sigma_{id_j}\sigma_{v_j^B})|\ldots|V\sigma_{id_n}\sigma_{v_n^A}|A_1|\ldots|A_l\right)|$$
$$VPB_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$
$$\approx_l \nu\tilde{n}.\left(V\sigma_{id_1}\sigma_{v_1^A}|\ldots\mid_{j\in J}(V\sigma_{id_j}\boldsymbol{\sigma_{v_j^A}})|\ldots|V\sigma_{id_n}\sigma_{v_n^A}|A_1|\ldots|A_l\right)|$$
$$VPB_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$
$$\approx_l VPA|VPB_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

Thus

$$VPA_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]|VPA \approx_l VPA|VPB_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

As the protocol is finite, we can apply Lemma 15 on page 44 to conclude

$$VPA_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_J\left[\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

□

As in practical situations voting protocols need to announce the result after a finite amount of time, assuming that they are finite appears to be realistic. The modularity assumption used in the above theorem is also fulfilled by many protocols, as we illustrate now using three case studies. We start with our first case study from Section 3.3.5: the protocol by Fujioka, Okamoto and Ohta.

Theorem 34 *The protocol by Fujioka, Okamoto and Ohta ensures Modularity (Mod).*

Proof We analyze the structure of the voting processes. We consider the model used in the proof of $SwVP^{I,PO}$ (Theorem 19 on page 71), i.e without any secret channels or keys, and no trusted authorities. All keys are free names. In the case of n voters, we have the form

$$V_1 | \ldots | V_n$$

Hence we can rewrite this instance into n instances containing one voter each:

$$V_1 | \ldots | V_n$$

which is syntactically equivalent. □

Remark Note that the voters use synchronization points, and that even after the decomposition the voters in the different instances synchronize. This is necessary, as otherwise the decomposition becomes trivially distinguishable from the initial big instance: for example in one decomposed instance results can be published although not all votes in the other instances have been submitted, which is impossible in the big instance. Intuitively this means that modularity only works for instances sharing the same deadlines.

Remark Note also that in many proofs (e.g. of Theorem 33 on page 129, but also in the following) we replace some instances in the decomposition with bisimilar ones. However we only showed the bisimilarity in a situation where only the processes in this instance had to synchronize, yet now, in this new, bigger context, they also synchronize with other processes in other instances. Although we do not give a full formal proof here, we argue that this is sound, i.e. that the initial composed instance is bisimilar to the one after the replacement. For instances without synchronization we know that such a replacement is correct, as labeled bisimilarity is closed under the application of contexts. When replacing an instance with synchronization within a bigger instance, the only difference to a simple context application is the synchronization. However we can see that the processes in this instance can still only synchronize if all processes in the instances have reached the synchronization point, as now all processes in all instances have to reach the synchronization point. We argue similarly in the case of a k out of n synchronization (e.g. in the case of Bingo Voting below), as at the same time the values k and n are adjusted. In the case of Bingo Voting half of all processes are required for synchronization, this remains true for the replaced instance within the bigger instance as we have a different value for n.

Give the above result, by Theorems 30 on page 124 and 33 on page 129 we have that FOO ensures (VP^{MC}) as it respects (EQ) (Theorem 27 on page 121), (Mod) (Theorem 34 on the preceding page) and is finite. As a second example we analyze the protocol by Okamoto.

Theorem 35 *The protocol by Okamoto ensures Modularity (Mod).*

Proof We analyze the structure of the voting processes. In the case of n voters, we have the form

$$\nu\text{chT}.(V_1|\ldots|V_n \underset{i=1,\ldots,n}{|} \text{processT}),$$

where processT is the process executed by the timeliness member and chT is the private channel between voters and the timeliness member. For $k \in \{1,\ldots,n-1\}$, a possible decomposition would be

$$\nu\text{chT}.(V_1|\ldots|V_k \underset{i=1,\ldots,k}{|} \text{processT})|\nu\text{chT}.(V_{k+1}|\ldots|V_n \underset{i=k+1,\ldots,n}{|} \text{processT}),$$

which would be syntactically equivalent except for the private channel `chT`, which is now decomposed into two channels. This affects only one transition, namely the communication between the voter[11] and the timeliness member when the voter reveals his random value. Consider a communication between a V_i and one `processT` on the left side: This can obviously be matched by the same transition on the right, and vice versa (as the `processT` are all the same, it is not important which one of them is chosen). □

This result generalizes the proof of Theorem 21 on page 80: in the proof we only considered two honest and one corrupted voter. As (Mod) allows us to compose instances, we can add instances with an arbitrary number of honest voters. This formally extends the result to cases with an arbitrary number of voters. Our last example is Bingo Voting.

Theorem 36 *Bingo Voting ensures Modularity (Mod).*

Proof In the case of n voters, we have the following voting process

$$\begin{aligned}&\nu\text{privChM}_1\ldots\nu\text{privChM}_n.\nu\text{privChRM}_1\ldots\nu\text{privChRM}_n.\\&\nu\text{privChR}_1\ldots\nu\text{privChR}_n.(V_1|\ldots|V_n|M_{1,\ldots,n;l}|R_1|\ldots|R_n)\end{aligned} \tag{3.17}$$

where R_i are the trusted random number generators, $M_{1,\ldots,n;l}$ is the voting machine process for n voters from 1 to n and l candidates, and privChM$_i$, privChRM$_i$ and privChR$_i$ are the private channels between the voter and the voting machine, the RNG and the voting machine, and the RNG and the voter respectively. For

[11] Here we consider generalized voting processes, however due to the condition on the V_i's we know that there is exactly one message on this channel.

$k \in \{1, \ldots, n-1\}$, this can be rewritten using Lemma 17 on page 60 as

$$\begin{array}{c} \nu\text{privChM}_1 \ldots \nu\text{privChM}_k.\nu\text{privChRM}_1 \ldots \nu\text{privChRM}_k. \\ \nu\text{privChR}_1 \ldots \nu\text{privChR}_k.(V_1| \ldots |V_k|M_{1,\ldots,k;l}|R_1| \ldots |R_k)| \\ \nu\text{privChM}_{k+1} \ldots \nu\text{privChM}_n.\nu\text{privChRM}_{k+1} \ldots \nu\text{privChRM}_n. \\ \nu\text{privChR}_{k+1} \ldots \nu\text{privChR}_n.(V_{k+1}| \ldots |V_n|M_{k+1,\ldots,n;l}|R_{k+1}| \ldots |R_n) \end{array} \tag{3.18}$$

as $M_{1,\ldots,n;l} \approx_l M_{1,\ldots,k;l}|M_{k+1,\ldots,n;l}$. This can be seen from the code in the Applied π-Calculus (Listing 3.19 on page 93). Hence (3.17) and (3.18) are bisimilar. □

As for the protocol by Okamoto, this generalizes the proof of Theorem 23 on page 94 to an arbitrary number of honest voters.

3.5 Multi-Voter Coercion

In this section we define Receipt-Freeness and Coercion-Resistance for weighted votes. We first consider the case where only one voter is coerced, then we define multi-voter coercion, and we also discuss the presence of corrupted voters.

3.5.1 — Single-Voter Receipt-Freeness (SRF)

We combine the approach used in Section 3.3.2 to define Receipt-Freeness with the generalized definition of Privacy for weighted votes (Definition 23 on page 116): If two instances of a voting protocol give the same result, they should be bisimilar even if one voter reveals his secret data in one case or fakes it in the other.

Definition 30 (Single-Voter Receipt Freeness (SRF)) *A voting protocol ensures* Single-Voter Receipt Freeness (SRF) *if for any two instances* $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ *and* $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any number* $i \in \{1, \ldots, n\}$ *there exists a process* V_i' *such that we have*

$$V_i'^{\backslash out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA_{\{i\}}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VPB_{\{i\}}\left[V_i'\right].$$

This definition is equivalent to the existing one based on swapping if the protocol ensures (Mod) and (EQ) or (EQA) respectively.

Theorem 37 *We have:*

— *If a protocol respects (EQ) and (Mod),* ($SwRF^{O,PO}$) *and (SRF) are equivalent.*

— *If a protocol respects (EQA) and (Mod),* ($SwRF^{O,FA}$) *and (SRF) are equivalent.*

Proof We only prove the first statement, the second is analogous.

— Suppose a protocol ensuring (SRF). We will now show that the protocol ensures ($SwRF^{O,PO}$), i.e. for any voting process VP and for all votes σ_{v_A} and σ_{v_B} that do not make a voter abstain there exists a process V_i' such that

$$V'^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B}$$

and

$$VP_{\{A,B\}}\left[(V\sigma_{id_A}\sigma_{v_A})^{chc}|V\sigma_{id_B}\sigma_{v_B}|0\right] \approx_l VP_{\{A,B\}}\left[V'|V\sigma_{id_B}\sigma_{v_A}|0\right]$$

Consider

$$\begin{aligned} VPA := \quad & VP_{\{A,B\}}\left[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}\right] \\ & \approx_l \\ & VP_{\{A,B\}}\left[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}\right] \quad := VPB \end{aligned}$$

Obviously the votes on both sides are a permutation of each other, thus we we have $VPA^H|_{res} \approx_l VPB^H|_{res}$ (by (EQ)). We can apply (SRF) to obtain for any i the existence of a process V_i' such that

$$V_i'^{\backslash out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA_{\{i\}}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VPA_{\{i\}}\left[V_i'\right]$$

We choose $i = A$ and obtain the desired property. Note that we have to rename the channel chc_i to chc, which is possible as labeled bisimilarity is closed under α-renaming.

— Suppose a protocol ensuring ($SwRF^{O,PO}$) and we want to show (SRF), i.e. for any voting processes $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ and $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ and any number $i \in \{1, \ldots, n\}$ there exists a process V_i' such that we have

$$V_i'^{\backslash out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA_{\{i\}}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VPB_{\{i\}}\left[V_i'\right]$$

Assume $VPA^H|_{res} \approx_l VPB^H|_{res}$. Then, as we suppose (EQ), the votes on the left are a permutation of the votes on the right. ($SwRF^{O,PO}$) allows us

the swap the vote of the targeted voter i with any other voter's vote. However the other voters in the instance can also swap votes. ($SwRF^{O,PO}$) implies ($SwVP^{O,PO}$) (cf. Lemma 18 on page 65), however we cannot swap the other voter's votes directly: As one voter is under attack, we cannot apply the equation. If however the protocol is modular, we can decompose the protocol into two instances where one contains only honest voters, and where we can then swap if necessary. By composability, we can then compose both instances and obtain the desired result. □

To see why Modularity is important in the proof, consider the following example.

Example 19 *Imagine a voting protocol where each voter publishes a signed commitment to his vote, and reveals the randomness to open it to all other voters and the election authority using private channels. The election authority opens the commitments and announces the result, but does not publish the values needed to open the commitments. To obtain Receipt-Freeness, we allow a coerced voter to ask the other voters (before the voting starts, and using private channels) if somebody is voting for the candidate the coercer wants him to vote for. If somebody does, both voters swap commitments, and the coerced voter can sign and publish the other voter's commitment to claim that this is his own commitment.*

We can see that such a protocol ensures ($SwVP^{O,PO}$) as the values necessary to open the commitments remain private for an outside attacker, yet still all voters can compute and verify the outcome. The protocol also ensures ($SwRF^{O,PO}$): the coerced voter can successfully produce a false receipt by swapping commitments. At the same time his receipt includes the values to open the commitments of all voters, hence violating privacy of all other voters. The protocol also ensures (EQ) if votes are not weighted, however it does not ensure (SRF) as the voter under attack breaks privacy for all other voters. For the same reason the protocol is not modular: we cannot add a corrupted voter, since he can break privacy.

Again, we can include corrupted voter(s) as follows.

Definition 31 (SRF with multiple corrupted voters (SRF^{MC})) *A voting protocol ensures* Single-Voter Receipt Freeness with multiple corrupted voters (SRF^{MC}) *if for any two instances* $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ *and* $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any number* $i \in \{1,\ldots,n\}$ *there exists a process* V_i' *such that for any subset* $J \subseteq \{1,\ldots,i-1,i+1,\ldots n\}$, $J \neq \{1,\ldots,i-1,i+1,\ldots n\}$ *if* $n > 1$, *with* $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ *we have*

$$V_i'^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res}$$
$$\Downarrow$$
$$VPA_{J\cup\{i\}}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_{J\cup\{i\}}\left[V_i' \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

Definition 32 (SRF with a single corrupted voter (SRF^{SC})) *A voting protocol ensures* Single-Voter Receipt Freeness with a single corrupted voter (SRF^{SC}) *if it ensures Single-Voter Receipt Freeness with multiple corrupted voters (SRF^{MC}) for any $|J| \leq 1$.*

Similarly to swap-based definitions, (SRF) is stronger than (VP). The proof is analogous to the proof in the swap-based model ($SwRF^{Attacker,Abs}$ implies $SwVP^{Attacker,Abs}$ in Lemma 18 on page 65).

Theorem 38 *For any $X \in \{\epsilon^{12}, SC, MC\}$ we have: If a protocol respects (SRF^X), it also respects (VP^X).*

Proof We only show that a protocol ensuring (SRF^{MC}) also ensures (VP^{MC}), the other cases follow directly.

We need to show that for any subset $J \subseteq \{1,\ldots,i-1,i+1,\ldots n\}$, $J \neq \{1,\ldots,i-1,i+1,\ldots n\}$ if $n > 1$, such that $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ we have

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA_J\left[\mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_J\left[\mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

By hypothesis there is a closed plain process so that

$$V_i'^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B} \tag{3.19}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res}$$
$$\Downarrow$$
$$VPA_{J\cup\{i\}}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_{J\cup\{i\}}\left[V_i' \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

We suppose $VPA^H|_{res} \approx_l VPB^H|_{res}$. Then we can apply the context $\nu chc_i.(\ \_\ |$

[12] Here ϵ denotes the empty string.

$!\mathtt{in}(chc_i, x))$ on both sides, which gives

$$VPA_{J\cup\{i\}}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]^{\backslash out(chc_i,\cdot)} \approx_l VPB_{J\cup\{i\}}\left[V_i' \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]^{\backslash out(chc_i,\cdot)}$$

Using Lemma 17 on page 60 we obtain

$$VPA_{J\cup\{i\}}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]^{\backslash out(chc_i,\cdot)} \equiv VPA_{J\cup\{i\}}\left[\left(V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right)^{\backslash out(chc_i,\cdot)} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

and

$$VPB_{J\cup\{i\}}\left[V_i' \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]^{\backslash out(chc_i,\cdot)} \equiv VPB_{J\cup\{i\}}\left[V_i'^{\backslash out(chc_i,\cdot)} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

We can now apply Lemma 16 on page 60, (3.19) and use the fact that labeled bisimilarity is closed under structural equivalence and conclude

$$VPA_J\left[\mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] = VPA_{J\cup\{i\}}\left[V\sigma_{id_i}\sigma_{v_i^A} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$
$$\approx_l VPB_{J\cup\{i\}}\left[V\sigma_{id_i}\sigma_{v_i^B} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] = VPB_J\left[\mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

□

We also have the following implications and equivalences.

Theorem 39 *We have:*

— *If a protocol respects (SRF^{MC}), then it also respects (SRF^{SC}).*

— *If a protocol respects (SRF^{SC}), then it also respects (SRF).*

— *If a protocol respects (EQ) and (Mod), then (SRF^{SC}) and $SwRF^{I,PO}$ are equivalent.*

— *If a protocol respects (EQA) and (Mod), then (SRF^{SC}) and $SwRF^{I,FA}$ are equivalent.*

— *If a protocol is modular, finite (i.e. any instance is finite) and respects (SRF), then it also respects (SRF^{MC}).*

Proof The first two implications are true by definition: (SRF) is a special case of (SRF^{SC}) with $J = \emptyset$, and (SRF^{SC}) is a special case of (SRF^{MC}) with $|J| \leq 1$.

We only show that (SRF^{SC}) and $SwRF^{I,PO}$ are equivalent if (EQ) and (Mod) hold, the other case is similar.

— Suppose a protocol ensuring (SRF^{SC}). We will now show that the protocol ensures ($SwRF^{I,PO}$), i.e. for any voting process VP and for all votes σ_{v_A} and σ_{v_B} that do not make a voter abstain, there exists a process V_i' such that

$$V'^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B}$$

and

$$VP_{\{A,B,C\}}\left[(V\sigma_{id_A}\sigma_{v_A})^{chc}|V\sigma_{id_B}\sigma_{v_B}|(V\sigma_{id_C})^{c_1,c_2}\right] \approx_l VP_{\{A,B,C\}}\left[V'|V\sigma_{id_B}\sigma_{v_A}|(V\sigma_{id_C})^{c_1,c_2}\right]$$

Consider

$$\begin{aligned} VPA := \quad & VP_{\{A,B,C\}}\left[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}|V\sigma_{id_C}\sigma_{v_C}\right] \\ & \approx_l \\ & VP_{\{A,B,C\}}\left[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}|V\sigma_{id_C}\sigma_{v_C}\right] \quad := VPB \end{aligned}$$

Obviously the votes on both sides are a permutation of each other, thus we we have $VPA^H|_{res} \approx_l VPB^H|_{res}$ (by (EQ)). We can apply (SRF^{SC}) to obtain for any i the existence of a process V_i' such that

$$V_i'^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res}$$

$$\Downarrow$$

$$VPA_{\{i\}\cup J}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_{\{i\}\cup J}\left[V_i' \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right],$$

We choose $i = A$ and $J = \{C\}$ and obtain the desired property. Note that again we have to rename the channel.

— Suppose a protocol ensuring ($SwRF^{I,PO}$) and we want to show (SRF), i.e. for any voting processes $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ and $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ and any number $i \in \{1,\ldots,n\}$ there exists a process V_i' such that for any subset $J \subseteq \{1,\ldots,i-1,i+1,\ldots n\}$, $J \neq \{1,\ldots,i-1,i+1,\ldots n\}$ if $n > 1$, such that

$\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ and $|J| \leq 1$ we have

$$V_i'^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res}$$
$$\Downarrow$$
$$VPA_{\{i\}\cup J}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$
$$\approx_l$$
$$VPB_{\{i\}\cup J}\left[V_i' \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right],$$

Assume $VPA^H|_{res} \approx_l VPB^H|_{res}$. As the protocol respects (EQ), we know that there exists a permutation π such that $\forall i : \sigma_{v_i^A} = \sigma_{v_{\pi(i)}^B}$. Additionally we know that $\forall i \in I : \sigma_{v_i^A} = \sigma_{v_i^B}$. $(SwRF^{I,PO})$ allows us the swap the vote of the targeted voter i with any other voter's vote in presence of a corrupted voter. As $(SwRF^{I,PO})$ implies $(SwVP^{I,PO})$ (cf. Lemma 18 on page 65), we can also swap the votes of all other voters if necessary (as the protocol is modular), which gives the desired result.

The last claim is a consequence of Theorems 40 and 41 below: Theorem 40 on the following page states that any modular and finite protocol ensuring (SRF) also ensures (MRF) (as defined below), and Theorem 41 on page 142 states that any modular and finite protocol ensuring (MRF) also ensures (MRF^{MC}), and hence ensures (SRF^{SC}). □

3.5.2 — Multi-Voter Receipt-Freeness (MRF)

We now generalize the idea of Receipt-Freeness to the case where multiple voters are attacked. Instead of only considering one attacked voter i, we consider a set I of attacked voters. To be receipt-free, it should be possible for all attacked voters to fake the receipt. Note that we assume that there is always at least one honest voter, except for the case with only one voter.

Definition 33 (Multi-Voter Receipt Freeness (MRF)) *A voting protocol ensures* Multi-Voter Receipt Freeness (MRF) *if for any voting processes* $VPA = \nu\tilde{n}.\ (V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$, $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any subset* $I \subseteq \{1,\ldots,n\}$, $I \neq \{1,\ldots,n\}$ *if* $n > 1$, *then there exists processes* V_i' *such that we have*

$$\forall i \in I : V_i'^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA_I \left[\mid_{i \in I} (V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \right] \approx_l VPB_I \left[\mid_{i \in I} V_i' \right].$$

It is clear that (MRF) implies (SRF), and under certain conditions the converse is also true.

Theorem 40 *If a protocol is finite and modular (Mod), Single-Voter Receipt Freeness (SRF) and Multi-Voter Receipt Freeness (MRF) are equivalent.*

Proof It is easy to see that any protocol ensuring Multi-Voter Receipt Freeness also ensures Single-Voter Receipt Freeness, we simply set $I = \{i\}$.

Assume the protocol ensures Single-Voter Receipt Freeness. We want to prove that the protocol ensures Multi-Voter Receipt Freeness, i.e. that for any $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$, $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ and any subset $I \subseteq \{1, \ldots, n\}$, $I \neq \{1, \ldots, n\}$ if $n > 1$, there exists processes V_i' such that we have

$$\forall i \in I : V_i'^{\backslash out(chc_i, \cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B} \tag{3.20}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VP_I \left[\mid_{i \in I} V\sigma_{id_i}\sigma_{v_i^A}^{chc_i} \right] \approx_l VPB_I \left[\mid_{i \in I} V_i' \right] \tag{3.21}$$

We suppose $VPA^H|_{res} \approx_l VPB^H|_{res}$ and we show that there exist processes V_i' such that we have (3.20) and

$$VPA_I \left[\mid_{i \in I} V\sigma_{id_i}\sigma_{v_i^A}^{chc_i} \right] \mid_{j \in \{1,\ldots,|I|-1\}} VPA \approx_l VPB_I \left[\mid_{i \in I} V_i' \right] \mid_{j \in \{1,\ldots,|I|-1\}} VPB \tag{3.22}$$

which gives the desired result using the cancellation result from Lemma 15 and (VP). The idea is that adding other voting processes allows us to decompose, mix and compose the processes so that we can apply (SRF) on the individual instances.

We start by decomposing the left side of the bisimilarity and recompose the

processes to have $|I|$ instances with single-voter coercion:

$$\begin{aligned} & VPA_I\left[\mid_{i\in I} V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \mid_{j\in\{1,\ldots,|I|-1\}} VPA \\ \approx_l \quad & \mid_{i\in I} VPA^i_{\{i\}}\left[V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \mid_{i\notin I} VPA^i_{\{i\}}\left[V\sigma_{id_i}\sigma_{v_i^A}\right] \\ & \mid_{j\in\{1,\ldots,|I|-1\}} \left(\mid_{i\in\{1,\ldots,n\}} VPA^i_{\{i\}}\left[V\sigma_{id_i}\sigma_{v_i^A}\right]\right) \\ \approx_l \quad & \mid_{i\in I} VPA_{\{i\}}\left[V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \end{aligned} \tag{3.23}$$

In the first step we break the instances down to a single voter each, i.e. VPA^i is a voting processes only one voter i. In the next step, we assemble the voters and instances in such a way that we obtain complete instances (i.e. containing voters 1 to n) with one attacked voter per instance.

We apply the same transformations on the right side and obtain:

$$\begin{aligned} & VPB_I\left[\mid_{i\in I} V_i'\right] \mid_{j\in\{1,\ldots,|I|-1\}} VPB \\ \approx_l \quad & \mid_{i\in I} VPB^i_{\{i\}}\left[V_i'\right] \mid_{i\notin I} VPB^i_{\{i\}}\left[V\sigma_{id_i}\sigma_{v_i^B}\right] \\ & \mid_{j\in\{1,\ldots,|I|-1\}} \left(\mid_{i\in\{1,\ldots,n\}} VPB^i_{\{i\}}\left[V\sigma_{id_i}\sigma_{v_i^B}\right]\right) \\ \approx_l \quad & \mid_{i\in I} VPB_{\{i\}}\left[V_i'\right] \end{aligned} \tag{3.24}$$

As we have $VPA^H|_{res} = VPB^H|_{res}$, we can apply Single-Voter Receipt Freeness on the left side, i.e. we have for any $i \in \{1,\ldots,n\}$ there exists a process V_i' such that

$$V_i'^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B} \tag{3.25}$$

and

$$VPA_I\left[V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \approx_l VPB_I\left[V_i'\right] \tag{3.26}$$

Using this we can rewrite (3.23) in (3.24), and hence prove (3.22). □

As above, we can include corrupted voter(s) as follows.

Definition 34 (MRF with multiple corrupted voters (MRF^{MC})) *A voting protocol ensures* Multi-Voter Receipt Freeness with multiple corrupted voters (MRF^{MC}) *if for any two instances* $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ *and* $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any subset* $I \subseteq \{1,\ldots,n\}$ *there exists processes* V_i' *such that for any subset* $J \subseteq \{1,\ldots,n\}$, $J \cap I = \emptyset$, $J \cup I \neq \{1,\ldots,n\}$ *if* $n > 1$, *such that* $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ *we have*

$$V_i'^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res}$$
$$\Downarrow$$
$$VPA_{I\cup J}\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$
$$\approx_l$$
$$VPB_{I\cup J}\left[\mid_{i\in I} V_i' \mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right].$$

Definition 35 (MRF with a single corrupted voter (MRF^{SC})) *A voting protocol ensures* Multi-Voter Receipt Freeness with a single corrupted voter (MRF^{SC}) *if it ensures Multi-Voter Receipt Freeness with multiple corrupted voters (MRF^{MC}) for any $|J| \leq 1$.*

Again, we have several implications and equivalences.

Theorem 41 *We have for any $X \in \{\epsilon, SC, MC\}$:*

— *If a protocol respects (MRF^{MC}), then it also respects (MRF^{SC}).*

— *If a protocol respects (MRF^{SC}), then it also respects (MRF).*

— *If a protocol respects (MRF^{X}), then it also respects (SRF^{X}).*

— *If a protocol is modular, finite (i.e. any instance is finite) and respects (MRF), then it also respects (MRF^{MC}).*

Proof The first three implications are true by definition: (MRF) is a special case of (MRF^{SC}) with $J = \emptyset$, and (MRF^{SC}) is a special case of (MRF^{MC}) with $|J| \leq 1$. Similarly (SRF^{MC}) is a special case of (MRF^{MC}), and (SRF^{SC}) of (MRF^{SC}), and (SRF) of (MRF), all with $I = \{i\}$.

For the last claim, assume we have two instances VPA and VPB, and a subset $I \subseteq \{1, \ldots, n\}$, and a subset $J \subseteq \{1, \ldots, n\}$, $J \cap I = \emptyset$, $J \cup I \neq \{1, \ldots, n\}$ if $n > 1$, such that $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$, and $VPA^H|_{res} \approx_l VPB^H|_{res}$. Then by (MRF) there exits processes V_i' such that $V_i'^{\backslash out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$ and

$$VPA_I\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VPB_I\left[\mid_{i\in I} V_i'\right].$$

Consider now

$$VPA_{I\cup J}\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right] | VPA_I\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]$$

By the above we have

$$VPA_{I\cup J}\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]|VPA_I\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]$$
$$\approx_l VPA_{I\cup J}\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]|VPB_I\left[\mid_{i\in I}V_i'\right]$$

As the protocol is modular, we can decompose $VPB_I\left[\mid_{i\in I}V_i'\right]$, and recompose with the corrupted voters to obtain

$$VPA_{I\cup J}\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]|VPB_I\left[\mid_{i\in I}V_i'\right]$$
$$\approx_l \nu\tilde{n}.\left(V\sigma_{id_1}\sigma_{v_1^A}|\ldots\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\mid_{j\in J}(V\sigma_{id_j}\sigma_{v_j^B})|\ldots|V\sigma_{id_n}\sigma_{v_n^A}|A_1|\ldots|A_l\right)|$$
$$VPB_{I\cup J}\left[\mid_{i\in I}V_i'\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

Since $\forall i\in I:\sigma_{v_i^A}=\sigma_{v_i^B}$ we have

$$\nu\tilde{n}.\left(V\sigma_{id_1}\sigma_{v_1^A}|\ldots\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\mid_{j\in J}(V\sigma_{id_j}\boldsymbol{\sigma_{v_j^B}})|\ldots|V\sigma_{id_n}\sigma_{v_n^A}|A_1|\ldots|A_l\right)|$$
$$VPB_{I\cup J}\left[\mid_{i\in I}V_i'\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$
$$\approx_l \nu\tilde{n}.\left(V\sigma_{id_1}\sigma_{v_1^A}|\ldots\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\mid_{j\in J}(V\sigma_{id_j}\boldsymbol{\sigma_{v_j^A}})|\ldots|V\sigma_{id_n}\sigma_{v_n^A}|A_1|\ldots|A_l\right)|$$
$$VPB_{I\cup J}\left[\mid_{i\in I}V_i'\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$
$$\approx_l VPA_I\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]|VPB_{I\cup J}\left[\mid_{i\in I}V_i'\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

Thus

$$VPA_{I\cup J}\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]|VPA_I\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]$$
$$\approx_l VPA_I\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]|VPB_{I\cup J}\left[\mid_{i\in I}V_i'\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

As the protocol is finite, we can apply Lemma 15 on page 44 to conclude

$$VPA_{I\cup J}\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]\approx_l VPB_{I\cup J}\left[\mid_{i\in I}V_i'\mid_{j\in J}(V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

□

The last theorem allows us to show that the protocol by Okamoto is secure against multi-voter coercion, i.e. ensures (MRF^{MC}).

Lemma 42 *The protocol by Okamoto ensures Multi-Voter Receipt Freeness with multiple corrupted voters (MRF^{MC}).*

Proof As shown in Theorems 28 on page 122 and 35 on page 132, the protocol ensures (EQ) and is modular. Using Theorem 39 on page 137 we have that it ensures (SRF^{SC}) and (SRF). It is easy to see that all instances are finite, hence we have that it ensures (MRF) using Theorem 40 on page 140, and finally that it ensures (MRF^{MC}) using Theorem 41 on page 142. □

Note that we expect this to hold also for a variant of the protocol with weighted votes. Similarly to the variant of FOO we could implement this using multiple ballots, and expect that the resulting protocol ensures (SRF), (MRF), (MRF^{MC}) and (Mod), but neither (EQ) nor $SwRF^{I,PO}$ as the votes are weighted.

3.5.3 — Single-Voter Coercion (SCR)

After discussing Receipt-Freeness, we now define Coercion-Resistance. As before, we start with Single-Voter Coercion-Resistance.

In this case, we combine (VP) with ($SwCR$): if two instances of a voting protocol give the same result, they should be bisimilar even if one voter interacts with the attacker in one case or only pretends to do so in the other case. The coercion is modeled by the context C that interacts with the voter and tries to force him to vote for a certain candidate.

Definition 36 (Single-Voter Coercion-Resistance (SCR)) *A voting protocol ensures* Single-Voter Coercion-Resistance (SCR) *if for any two instances $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ and $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ and any number $i \in \{1,\ldots,n\}$ there exists a process V_i' such that for any context C_i with $C_i = \nu c_1.\nu c_2.(\_|P_i)$ and $\tilde{n} \cap fn(C) = \emptyset$,*

$$VPA_{\{i\}}\left[C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VPA_{\{i\}}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]$$

we have

$$C_i\left[V_i'\right]^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res} \Rightarrow VPA_{\{i\}}\left[C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VPB_{\{i\}}\left[C_i\left[V_i'\right]\right]$$

As above, we can link this definition to the swap-based definition using (Mod), (EQ) and (EQA).

Theorem 43 *We have*

— *If a protocol respects (EQ) and (Mod), (SCR) and* ($SwCR^{O,PO}$) *are equivalent.*

— *If a protocol respects (EQA) and (Mod), (SCR) and* ($SwCR^{O,FA}$) *are equivalent.*

Proof Analogous to the proof of Theorem 37 on page 133. □

Similarly, we can also consider corrupted voters.

Definition 37 (SCR with multiple corrupted voters (SCR^{MC})) *A voting protocol ensures* Single-Voter Coercion-Resistance with multiple corrupted voters (SCR^{MC}) *if for any two instances* $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ *and* $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any number* $i \in \{1,\ldots,n\}$ *there exists a process* V_i' *such that for any context* C_i *with* $C_i = \nu c_1.\nu c_2.(\_|P_i)$ *and* $\tilde{n} \cap fn(C) = \emptyset$ *and for any subset* $J \subseteq \{1,\ldots,i-1,i+1,\ldots n\}$, $J \neq \{1,\ldots,i-1,i+1,\ldots n\}$ *if* $n > 1$, *such that* $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$ *and*

$$VPA_{\{i\}\cup J}\left[C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPA_{\{i\}\cup J}\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

we have

$$C_i\left[V_i'\right]^{\backslash out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res}$$
$$\Downarrow$$
$$VPA_{\{i\}\cup J}\left[C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$
$$\approx_l$$
$$VPB_{\{i\}\cup J}\left[C_i\left[V_i'\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

Definition 38 (SCR with a single corrupted voter (SCR^{SC})) *A voting protocol ensures* Single-Voter Coercion-Resistance with a single corrupted voter (SCR^{SC}) *if it ensures Single-Voter Coercion-Resistance with multiple corrupted voters (SCR^{MC}) for any* $|J| \leq 1$.

Again, we have the same equivalencies.

Theorem 44 *We have:*

— *If a protocol respects (SCR^{MC}), then it also respects (SCR^{SC}).*

— *If a protocol respects (SCR^{SC}), then it also respects (SCR).*

— *If a protocol respects (EQ) and (Mod), then (SCR^{SC}) and ($SwCR^{I,PO}$) are equivalent.*

— *If a protocol respects (EQA) and (Mod), then (SCR^{SC}) and ($SwCR^{I,FA}$) are equivalent.*

— *If a protocol is modular, finite (i.e. any instance is finite) and respects (SCR), then it also respects (SCR^{MC}).*

Proof The first two implications are true by definition: (SCR) is a special case of (SCR^{SC}) with $J = \emptyset$, and (SCR^{SC}) is a special case of (SCR^{MC}) with $|J| \leq 1$. The other proofs are analogous to the proof of Theorem 39 on page 137. □

3.5.4 — Multi-Voter Coercion (MCR)

We now discuss Multi-Voter Coercion-Resistance. To model the case where multiple voters are attacked, we consider the set I of attacked voters.

Definition 39 (Multi-Voter Coercion-Resistance (MCR)) *A voting protocol ensures* Multi-Voter Coercion-Resistance (MCR) *if for any voting processes* $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ *and* $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any subset* $I \subseteq \{1,\ldots,n\}$, $I \neq \{1,\ldots,n\}$ *if* $n > 1$, *there exists processes* V_i' *such that for any contexts* $C_i = \nu c_1.\nu c_2.(\_|P_i), i \in I$ *and* $\tilde{n} \cap fn(C) = \emptyset$ *and*

$$VPA_I\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VPA_I\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]$$

we have

$$\forall i \in I : C_i\left[V_i'\right]^{\backslash out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res}$$
$$\Downarrow$$
$$VPA_I\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VPB_I\left[\mid_{i\in I} C_i\left[V_i'\right]\right],$$

To consider corrupted voters, we propose the following definitions.

Definition 40 (MCR with multiple corrupted voters (MCR^{MC})) *A voting protocol ensures* Multi-Voter Coercion-Resistance with multiple corrupted voters (MCR^{MC}) *if for any two instances* $VPA = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A}$

$\mid A_1 \mid \ldots \mid A_l)$ *and* $VPB = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any subset* $I \subseteq \{1,\ldots,n\}$ *there exists processes* V_i' *such that for any contexts* $C_i, i \in I$ *with* $C_i = \nu c_1.\nu c_2.(\_ \mid P_i)$ *and* $\tilde{n} \cap fn(C) = \emptyset$ *and for any subset* $J \subseteq \{1,\ldots,n\}$, $J \cap I = \emptyset$, $J \cup I \neq \{1,\ldots,n\}$ *if* $n > 1$, *such that* $\forall j \in J : \sigma_{v_j^A} = \sigma_{v_j^B}$, *and*

$$VPA_{I\cup J}\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPA_{I\cup J}\left[\mid_{i\in I} (V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

we have

$$\forall i \in I : C_i\left[V_i'\right]^{\backslash out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VPA^H|_{res} \approx_l VPB^H|_{res}$$
$$\Downarrow$$
$$VPA_{I\cup J}\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$
$$\approx_l$$
$$VPB_{I\cup J}\left[\mid_{i\in I} C_i\left[V_i'\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

Definition 41 (MCR with a single corrupted voter (MCR^{SC})) *A voting protocol ensures* Multi-Voter Coercion-Resistance with a single corrupted voter (MCR^{SC}) *if it ensures Multi-Voter Coercion-Resistance with multiple corrupted voters (MCR^{MC}) for any* $|J| \leq 1$.

Again, we have several implications and equivalences.

Theorem 45 *We have for any* $X \in \{\epsilon, SC, MC\}$*:*

— *If a protocol respects (MCR^X), then it also respects (SCR^X).*

— *If a protocol respects (MCR^{MC}), then it also respects (MCR^{SC}).*

— *If a protocol respects (MCR^{SC}), then it also respects (MCR).*

— *If a protocol respects (SCR^X), it also respects (SRF^X).*

— *If a protocol respects (MCR^X), it also respects (MRF^X).*

Proof The first three implications are direct consequences of the definitions. Then we only consider the case (MCR^{MC}) implies (MRF^{MC}), the other follow directly.

The proof is similar to the proof of $SwCR^{Attacker,Abs}$ implies $SwRF^{Attacker,Abs}$ in Lemma 18 on page 65. Assume the protocol ensures (MCR^{MC}). Let C_i be

evaluation contexts such that $C_i = \nu c_1.\nu c_2.(\_|P_i)$ for some plain processes P_i which fulfill

$$VPA_{I\cup J}\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPA_{I\cup J}\left[\mid_{i\in I} (V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \quad (3.27)$$

Note that these C_i can be constructed directly from the vote process V. By hypothesis we know that there are closed plain process V_i' such that

$$\forall i \in I : C_i\left[V_i'\right]^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$\begin{gathered} VPA^H|_{res} \approx_l VPB^H|_{res} \\ \Downarrow \\ VPA_{I\cup J}\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \\ \approx_l \\ VPB_{I\cup J}\left[\mid_{i\in I} C_i\left[V_i'\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \end{gathered} \quad (3.28)$$

We have to find other processes V_i'' such that

$$\forall i \in I : V_i'^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B} \quad (3.29)$$

and

$$\begin{gathered} VPA^H|_{res} \approx_l VPB^H|_{res} \\ \Downarrow \\ VPA_{I\cup J}\left[\mid_{i\in I} (V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \\ \approx_l \\ VPB_{I\cup J}\left[\mid_{i\in I} V_i' \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \end{gathered} \quad (3.30)$$

Let $V_i'' = C_i[V_i']$. This directly fulfills the first requirement (Equation (3.29)). For the second equation (3.30), we suppose $VPA^H|_{res} \approx_l VPB^H|_{res}$. We can then use the condition on C_i (Equation (3.27)) and obtain

$$VPA_{I\cup J}\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPA_{I\cup J}\left[\mid_{i\in I} (V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

The second hypothesis (Equation (3.28)) gives

$$VPA_{I\cup J}\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_{I\cup J}\left[\mid_{i\in I} C_i\left[V_i'\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

As labeled bisimilarity is transitive, we can conclude

$$VPA_{I\cup J}\left[\mid_{i\in I} (V\sigma_{id_i}\sigma_{v_i^A})^{chc_i} \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right] \approx_l VPB_{I\cup J}\left[\mid_{i\in I} C_i\left[V_i'\right] \mid_{j\in J} (V\sigma_{id_j})^{c_{1j},c_{2j}}\right]$$

which gives us the desired result for $V_i'' = C_i[V_i']$. □

Finally we also have equivalence between (SCR) and (MCR) as well as (MCR) and (MCR^{MC}) under the same assumptions as in the case of Receipt-Freeness using a similar proof.

Theorem 46 *We have:*

— *If a protocol is modular, finite and respects (SCR), it also ensures (MCR).*

— *If a protocol is modular, finite and respects (MCR), then it also respects (MCR^{MC}).*

Proof Analogous to the proofs of Theorem 40 on page 140 and Theorem 41 on page 142. □

The last theorem allows us to show that Bingo Voting is secure against multi-voter coercion, i.e. ensures (MCR^{MC}).

Lemma 47 *Bingo Voting ensures Multi-Voter Coercion-Resistance with multiple corrupted voters (MCR^{MC}).*

Proof As shown in Theorem 23 on page 94, the protocol ensures $SwCR^{I,FA}$. In Theorems 29 on page 123 and 36 on page 132 we showed that it ensures (EQA) and (Mod). Using Theorem 44 on page 146 we have that it ensures (SCR^{SC}) and (SCR). It is easy to see that all instances are finite, thus we have that it ensures (MCR) and (MCR^{MC}) using Theorem 46. □

3.6 Conclusion

In this chapter we proposed a taxonomy of privacy notions for eVoting protocols. We started by extending the notions originally proposed by Delaune et al. [DKR09]

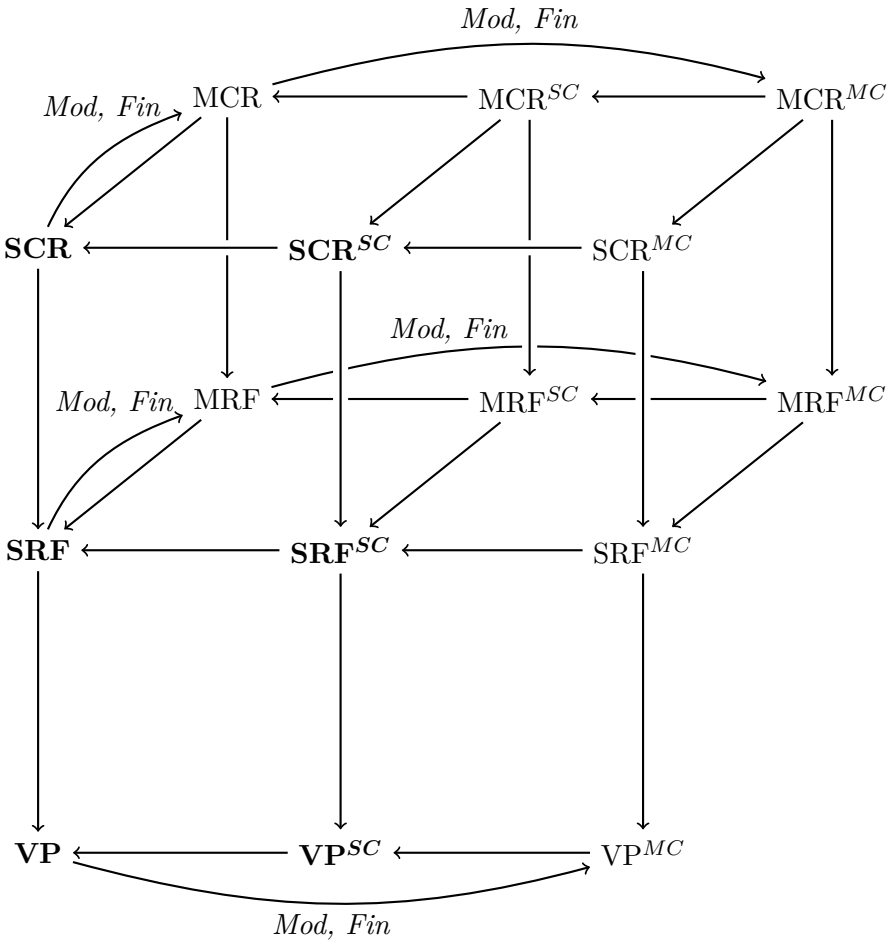

Figure 3.2 – The hierarchy of generalized privacy notions. A $\xrightarrow{C}$ B means that under the assumption C a protocol ensuring A also ensures B.

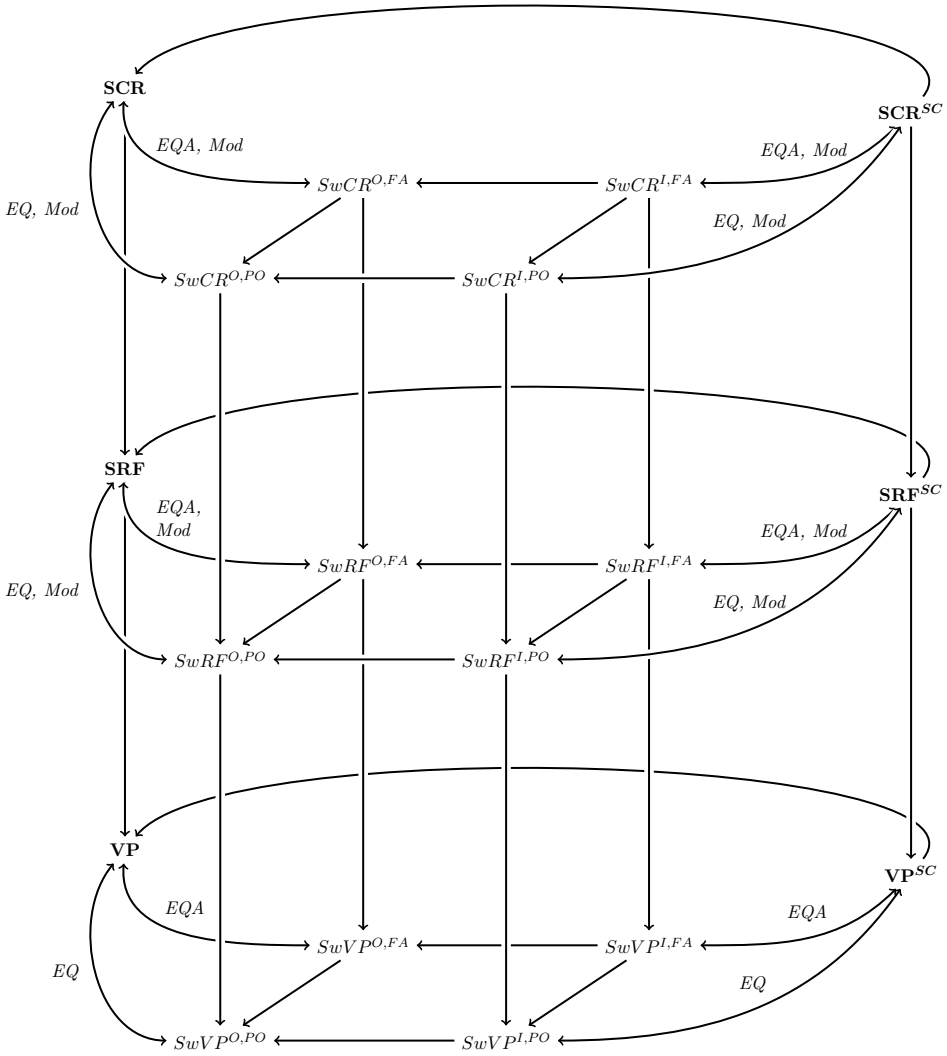

Figure 3.3 – The links between the generalized and swap-based privacy notions. A $\xrightarrow{C}$ B means that under the assumption C a protocol ensuring A also ensures B.

to cover corrupted voters and forced abstention attacks. We applied the resulting notions on several case studies: the protocols by Fujioka et al. [FOO92], the protocol by Okamoto [Oka96], the protocol by Lee et al. [LBD+03] and Bingo Voting [BMQR07]. The results are summed up in and Table 3.1 on page 114, the position of the case studies within our hierarchy is shown in Figure 3.1 on page 69.

As it turns out, these notions based on swapping votes are unsuitable for cases where votes are weighted. To address this issue we generalized the privacy notions to cover weighted votes, multiple corrupted and multiple coerced voters. Figure 3.2 on page 150 sums up the resulting notions and their relationships. We discussed a variant of the protocol by Fujioka et al. [FOO92] as a first case study. We also showed that under certain assumptions – a finite protocol and modularity –, multi-voter coercion and single-voter coercion are equivalent. This allows to prove security against multi-voter coercion by simply checking for finiteness and modularity, which are often simpler to prove than the entire property. Similarly, we were able to show that a finite protocol ensuring modularity is also secure against inside attackers: the cases with and without corrupted voters coincide. To show that this assumption is realistic, we showed that the protocol by Fujioka et al. [FOO92], the protocol by Okamoto [Oka96] and Bingo Voting [BMQR07] fulfill this property, and hence ensure Vote-Privacy with multiple corrupted voters, Multi-Voter Receipt Freeness with multiple corrupted voters and Multi-Voter Coercion-Resistance with multiple corrupted voters, respectively.

We also analyzed the link between these notions and the previous taxonomy. We were able to identify precise conditions: Equality of Votes and Equality of Abstention model the fact that votes are not weighted and that additionally abstaining voters remain anonymous, respectively. This allows to prove that some swap-based and some generalized privacy notions coincide for protocols ensuring one of the two properties. For Receipt-Freeness and Coercion-Resistance we also need Modularity as the generalized notions are slightly more powerful. The links between both taxonomies are summed up in Figure 3.3 on the previous page. The notions appearing in both Figures 3.2 on page 150 and 3.3 on the previous page are marked in **bold**, allowing to connect the two figures.

As future work, we would like to verify more properties automatically as manual proofs are often cumbersome and tend to be error-prone. One of the main problems here are the equational theories which become too complex for ProVerif to handle. There is some first work on replacing such theories with simpler, but equivalent ones [SAR13], which could help in dealing with more complex theories automatically. Moreover, KISS [CDK12] and AKISS [CCK12] can deal with more complex equational theories, which could help us in verifying more properties automatically.

We would also like give the full formal proof that we can replace a process

P synchronizing with other processes within a context $C[\_]$, with a bisimilar (modulo the synchronization) processes $P' \approx_l P$, i.e. that we have $C[P] \approx_l C[P']$.

Additionally the approach in this chapter is possibilistic: We call a protocol secure if there is a way for the targeted voter to escape coercion. As we are in a symbolic model, we do not consider probabilities. Hence the adversary may in reality still have a certain probability of detecting that the coerced voter tried to resist coercion. This is beyond the scope of this work, yet a computational translation of our definitions should be able take this into account. There already is some work in the computational model in this direction [KTV10b], yet they do not consider all different dimensions of our taxonomy.

Chapter 4

eAuctions

Auctions provide sellers and buyers with a way to exchange goods or services for a mutually acceptable price. Commonly the bidders strive for the least possible price, and the seller aims for the highest possible price. Due to the competitive nature of the process, security is important to ensure fairness.

In this chapter we propose a formal model for different security properties. We start by analyzing Authentication, Fairness and Privacy properties with the help of ProVerif. We discuss the protocols due to Curtis et al. [CPS07], Brandt [Bra06] and Sako [Sak00]. Then we propose an abstract model of Verifiability for different types of auctions, and provide case studies in the symbolic as well as in the computational model using ProVerif and CryptoVerif respectively.

Finally we discuss the idea of "true bidder-verifiable auctions", i.e. protocols that can be verified by a non-expert. This precludes relying on complex cryptography, hence we discuss two protocols achieving verifiability based only on the physical properties of the manipulated objects. Moreover, we propose a first approach to model these physical objects and their properties, and use it to provide a formal analysis of the protocols using ProVerif.

Contents

4.1 Introduction 157
4.1.1 Contributions 158
4.1.2 Outline of the Chapter 159
4.2 Related Work 159
4.3 Fairness, Authentication and Privacy in Auctions 162
4.3.1 Modeling Auction Protocols 162
4.3.2 Fairness Properties 166
4.3.3 Authentication Properties 167
4.3.4 Privacy Properties 169
4.3.4.1 Privacy 170

4.3.4.2 Receipt-Freeness 177
4.3.4.3 Coercion-Resistance 185
4.3.5 Case Studies 194
4.3.5.1 Protocol by Curtis, Pierprzyk and Seruga 194
4.3.5.2 Protocol by Brandt 202
4.3.5.3 Protocol by Sako 207
4.3.6 Summary 212
4.4 Verifiability in Auctions 213
4.4.1 A Different Model of Auction Protocols 213
4.4.2 Defining Verifiability 214
4.4.2.1 First-Price Auctions 215
4.4.2.2 Other Types of Auctions 217
4.4.3 Case Studies 218
4.4.3.1 Protocol by Sako 218
4.4.3.2 Protocol by Curtis et al. 236
4.4.4 Summary 241
4.5 Towards True Bidder-Verifiable Auctions 241
4.5.1 The “Cardako” Protocol 242
4.5.1.1 Description 242
4.5.1.2 Security Properties 243
4.5.1.3 Formal Analysis 244
4.5.2 The “Woodako” Protocol 248
4.5.2.1 Description 249
4.5.2.2 Securities Properties 256
4.5.2.3 Formal Analysis 257
4.5.3 Summary 266
4.6 Conclusion 267
4.6.1 Limitations and Future Work. 268

4.1 Introduction

Auctions are a simple method to sell goods and services. Typically a *seller* offers a good or a service, and the *bidders* make offers. Depending on the type of auction, the offers might be sent using sealed envelopes which are opened at the same time to determine the winner (the "sealed-bid" auction, e.g. [Cac99, NPS99, AS02, CLK03, LAN02, BCD+09]), or an *auctioneer* could announce prices decreasingly until one bidder is willing to pay the announced price (the "dutch auction", e.g. [RG95]). There might be several rounds, or offers might be announced publicly directly (the "English" or "shout-out" auction, e.g. [OM01]). The winner usually is the bidder submitting the highest bid, but in some cases he might only have to pay the second highest offer as a price (the "second-price"- or "Vickrey"-Auction, e.g. [HTK98, LAN02]). In general a bidder wants to win the auction at the lowest possible price, and the seller wants to sell his good at the highest possible price. For more information on different auction methods see e.g. [Kri02].

Due to the competitive nature of the auction process, security is a major concern to prohibit cheating and manipulations of the prices. As frauds are common in eAuctions [NTJ13], different security properties have been discussed in the literature. We study[1] the following security properties of auction protocols:

— **Fairness:** Firstly a fair (sealed-bid) auction protocol should not *leak* any information about the other participants and their offers until the bidding phase is over (so as to prohibit unfair tactics based on leaked information). We call this *Weak* or *Strong Noninterference*, depending on if the number of bidders is leaked or not. Thirdly a protocol should not allow anybody to win although they did not submit the highest price, i.e. ensure that the *Highest Price Wins*. Otherwise a losing bidder could try to cheat to win.

— **Authentication**: For the seller it is crucial to ensure *Non-Repudiation*, i.e. that – after the winner has been announced – the winning bidder cannot claim that he did not submit the winning bid. Additionally we might want to ensure *Non-Cancellation*, i.e. that a bidder cannot cancel a submitted offer before the winner is announced, to have binding bids.

— **Privacy:** Privacy is important in sealed-bid auctions to prevent information leakage after the auction is over, for example if an auction is organized in several rounds, or if the bids leak sensitive information e.g. to competitors or clients [BCD+09]. Again, we distinguish several different notions: *Secrecy of Bids*, *Bidding-Price Unlinkability*, *Weak Anonymity* and *Strong Anonymity* (of Bidders), *Receipt-Freeness* and *Coercion-Resistance*. Secrecy of Bids guarantees that the losing bids remain secret. In the case of Bidding-Price Unlinkability

[1] See Section 4.2 for a detailed discussion of the properties proposed in the literature.

the list of bids can be public, but must not be linkable to the losing bidders. Strong Anonymity of Bidders means that the participants, including the winner, remain anonymous, and all losing bids remain secret. In the case of Weak Anonymity the list of bids can be public, but the bids are not linkable to the bidders, including the winner, who remains anonymous. Receipt-Freeness ensures that bidders are unable to prove to an attacker (which might be another bidder trying to force them to submit a low bid so that he wins at a lower price) that they bid a certain offer, and Coercion-Resistance means that even when interacting with a coercer, the bidders can still bid a price of their choice.

— **Verifiability**: A verifiable protocol should allow the bidders to verify that the winner was correctly determined, and that no bids were manipulated or submitted by uneligible bidders. This allows to back up trust in the auctioneer or other authorities, since they often also have a financial interest in the auction. For example in the case of eBay the auction fees include a proportion of the selling price [eBa13].

Note that depending on the type of auction, some properties might be different or unachievable. For sealed-bid first-price auctions, the most common type of protocols in the literature, all of the above are suitable. However for example Secrecy of Bids is difficult to ensure in an English Auction where bids are announced publicly, but anonymity might still be achievable.

4.1.1 — Contributions

In this chapter we provide the following contributions:

— In the first part we give a formal framework in the Applied π-Calculus to model and analyze e-Auction protocols. Within this framework we define the discussed fairness, privacy and authentication properties and analyze their relationship.

— We provide three case studies: The protocol by Curtis et al. [CPS07], a protocol by Brandt [Bra06] and the protocol by Sako [Sak00]. We show how the three can be modeled in the Applied π-Calculus and verified using ProVerif [Bla01]. We discover several flaws on the first two protocols and explain how some of their shortcomings can be addressed. We also show that the protocol by Sako ensures all our properties except for receipt-freeness and coercion-resistance.

— In the second part, we give a high-level definition of Verifiability in eAuctions based on an abstract protocol level. We show how this definition can be generalized to accommodate different types of auctions.

— We apply this definition on the protocols by Sako and Curtis et al.. Again, we

rely on ProVerif to execute the Verification. For the protocol by Sako, we also provide a computational proof in CryptoVerif [Bla06a].

— Finally we provide two examples of "true bidder-verifiable" sealed-bid auction protocols. Although these protocols have their limitations with respect to scalability, they illustrate two simple ways of achieving secure and verifiable auctions by exploiting physical properties of physical objects instead of cryptography. We also discuss how we can apply the previous formal models to verify these "physical" protocols.

4.1.2 — Outline of the Chapter

In Section 4.2, we discuss related work. Then we model auction protocols in the Applied π-Calculus to formally define the security properties in Section 4.3.1. In the following Sections 4.3.2, 4.3.3 and 4.3.4 we model fairness, authentication and privacy properties respectively. In Section 4.3.5, we analyze our three case studies. In the following part (Section 4.4) we propose our abstract model of auction protocols for verifiability (Section 4.4.1) as well as the definition of verifiability (Section 4.4.2) and the case studies (Section 4.4.3). In the final part we propose two "true bidder-verifiable" protocols in Sections 4.5.1 and 4.5.2 and discuss their formal analysis. We then conclude in Section 4.6.

4.2 Related Work

In this section we discuss related work on *auction protocols*, the different *security properties*, previous work on *formal verification of auction protocols*, the *link to other applications* such as eVoting (in particular for privacy and verifiability) or contract-signing, and work on *modeling physical properties for security* and protocols.

Auction Protocols. Many electronic auction (e-Auction) protocols have been proposed in the literature (see e.g. [Bra02, Bra06, BS08, PSKT01] for an overview). They rely on a multitude of cryptographic primitives such as hash chains [SS99], signatures of knowledge and zero-knowledge proofs [OM01], coin-extractability, range proofs and proofs of knowledge [LAN02], proxy-oblivious transfers and secure evaluation functions [NPS99] and many others. There is also a protocol by Stajano and Anderson [SA99] solely based on anonymous broadcast, arguing that physical properties can be used to construct efficient and simple auction protocols. As case studies, we use the protocol by Curtis et al. [CPS07], which uses a trusted registrar and pseudonyms, the protocol by Brandt [Bra06], which is entirely distributed using secure multi-party computation, and the protocol by Sako [Sak00] which uses a distributed trusted authority.

Security Properties. The different security properties have been discussed since the early publications on eAuctions, e.g. Franklin and Reiter [FR95] discuss secrecy of bids, anonymity of bidders, fairness, non-repudiation and non-cancellation. Further publications [HTK98, KHT98, LKM01, OM01] have used and refined these notions, also adding verifiability. Abe and Suzuki [AS02] introduced and motivated Receipt-Freeness for e-Auctions. Cancellation of bids was also discussed by Stubblebine and Syverson [SS99] who proposed a protocol implementing cancellation as a feature, and another protocol ensuring non-cancellation. Still, all definitions given in these papers are informal.

Formal Verification of Auction Protocols. Although there has been much work on developing auction protocols ensuring various properties, there is considerably less work on their formal definition and analysis. Subramanian [Sub98] proposed an auction protocol and analyzed it using a BAN-style logic to show some security properties. In particular he showed the atomicity of the transaction, weak secrecy of private keys and a form of anonymity modeled as weak secrecy of the public key of the bidder. Using OFMC, Księżopolski and Lafourcade [KL07] identified an attack on authentication in an auction protocol. More recently Dong et al. [DJP11] analyzed a receipt-free auction protocol in the Applied π-Calculus. They only considered privacy, in particular secrecy of the bidding price and receipt-freeness, but only for losing bidders.

Link to other Applications. In the context of electronic voting there has been much more work on formal verification, in particular in the area of privacy as discussed in Section 3.2. Some notions are similar, yet there are also some differences to auctions. For example the information leaked by the result in both cases is usually not the same: typically in voting the published result is the sum of all votes (although there are more complex ways of computing the tally), whereas in auctions the public outcome is in most cases the winning bidder and price. Often the losing bids remain private, and in some cases even the winner stays anonymous, e.g. the well known "bidder on the phone". Although normally at least the winning price is public, there is the protocol by Brandt [Bra06] where only the winner and the seller learn the winner and the winning price, but to the best of our knowledge, this is the only protocol aiming for such a high level of privacy.

Concerning Verifiability there is a considerable amount of work in voting. The property of *individual verifiability* – a voter can verify that her vote counts correctly for the result – has been a well-established notion since the field's inception [FOO92, BT94, SK95, HS00]. Similarly the concept of *universal verifiability* – the property that all voters or even an outside an observer may verify (using only public information) the correctness of the tally – has also been discussed for a long

time [CF85, BT94, SK95, Ben96]. Kremer et al. [KRS10] formalized individual and universal verifiability in the Applied π-Calculus, and added the notion of *eligibility verifiability*: the property whereby any observer may verify, using only public information, that the set of cast votes from which the result is determined, originates only from eligible voters, and each eligible voter cast at most one vote. Smyth et al. [SRKK10] used ProVerif to automatically check Verifiability of voting protocols. They express the conditions in the definitions as reachability properties, which can be checked by ProVerif. In some of our case studies on the Verifiability of eAuction protocols we also use the Applied π-Calculus and employ a similar technique in ProVerif, but our model and definitions are more general and can also be instantiated using a computational model. Moreover, Küsters et al. [KTV10a] introduced the notion of *accountability*: when verifiability of a certain goal fails, it is possible to identify the party responsible for the failure. They also give symbolic and computational definitions of verifiability, which they identify as a weaker variant of accountability. As a case study, they apply their definitions on Bingo Voting. Although they also consider an auction protocol as a second case study, the verifiability goals used do not cover all verifications required by our definition. Finally, Guts et al [GFN09] defined *auditability*, i.e. the fact that a protocol logs sufficient evidence to convince an impartial judge that certain properties are satisfied. In our definition of verifiability the protocol only needs to convince the participants that the protocol execution was correct, but not necessary an outside judge. Moreover, Guts et al. verify auditability using static typing on the concrete protocol implementation, whereas our definition of verifiability leaves open the choice of the concrete method of verifying the definition.

While the intuition behind the notions from voting carries (to some degree) over to auctions, we note that auctions can be much more competitive as the bidders are pressed for time, for example in the case of an English auction. This can be exploited by an attacker: for example an illegal bid (e.g., by an unregistered bidder) may increase the winning price by forcing the honest bidders to increase their offer, while finally not changing the winner. Hence, a lack of verifiability (which would allow the honest bidders to identify the incorrect bid) can compromise fairness. In addition, winner and price determination can be quite complex depending on the type of auction, e.g. for second-price or multi-good auctions[2]. Thus, although providing important inspiration, verification of voting systems does not translate directly to verification of auction systems.

There has been a lot of work on Non-Repudiation in the context of contract signing and fair exchange protocols (e.g. [KR03, KV09, LV10]). We rely on the work by Klay et al. [KV09] who propose many different flavors of non-repudiation

[2]Note however that although in many cases the outcome computation in voting is relatively simple, there are more complex voting systems such as Single Transferable Vote (STV) used e.g. in Ireland [Cit13].

based on agent knowledge or authentication. We only consider "Non-Repudiation of Origin", i.e. that the bidder cannot deny that he made an offer, implemented as a form of authentication.

Modeling Physical Properties. In the last part of this chapter, we also discuss the formal analysis of protocol relying on physical objects. Several such protocols were developed for various purposes, e.g. [CK94, FNW96, NP97, Cha04]. Blaze [Bla03, Bla04b] analyzed physical locks from a cryptographic point of view, and argued that physical security should be taken into account in security modeling [Bla06b]. Other work in this area focused on modeling physical security in various ways. The Portunes framework by Dimkov et al. [DPH11] allows modeling of attacks that cross physical, digital and social domains. They focus on access control and data breaches. Moran and Naor [MN05, MN06a] proposed several protocols using physical objects, including a "human-centric" polling protocol based on physical envelopes, and also discussed their formal analysis in the UC-framework. Recently Meadows et al. [MP13] describe a way to formalize security procedures (accounting for physical objects) in logic, however again their work is more concerned about the movement of physical objects, rather than the properties of the objects itself. Basin et al. [BCSS11] proposed a formal model to verify protocols based on physical attributes such as time and location. Their model includes the constraints of these physical domains in their reasoning, however does not express properties of physical objects that can be manipulated by the participants.

4.3 Fairness, Authentication and Privacy in Auctions

In this section, we describe our model of auction protocols in the Applied π-Calculus and propose definitions for multiple properties related to fairness, authentication and privacy. We also discuss our three case studies: The protocols by Brandt [Bra06], Curtis et al. [CPS07] and Sako [Sak00].

4.3.1 — Modeling Auction Protocols

Similar to voting protocols, we model *auction protocols* in the Applied π-Calculus as follows.

Definition 42 (Auction Protocol) *An* auction protocol *is defined by a tuple* $(B, S, A_1, \ldots, A_m, \tilde{m})$ *where* B *is the process executed by the bidders,* S *is the process executed by the seller, and the* A_j*'s are the processes executed by the authorities (for example an auctioneer, a registrar etc.), and* $\tilde{m}$ *is a set of private channels. We also assume the existence of a particular public channel* res *that is only used to publish the winning bid or bidder.*

Note that we have only one process for the bidders. This means that different bidders will execute the same process, but with different variable values (e.g. the keys, the bids etc.). To reason about privacy, we talk about instances of an auction protocol, which we call *auction processes*.

Definition 43 (Auction Process) *An instance of an auction protocol* (B, S, A_1, ..., $A_m, \tilde{m}$) *is called an* auction process, *which is a closed process*

$$\nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_1}|\ldots|B\sigma_{id_k}\sigma_{b_k}|S|A_1|\ldots|A_l),$$

where $l \leq m$, $\tilde{n}$ *includes the secret channel names* $\tilde{m}$, $B\sigma_{id_i}\sigma_{b_i}$ *are the processes executed by the* k *bidders,* σ_{id_i} *is a substitution assigning the identity to the* i*-th bidder,* σ_{b_i} *specifies the* i*-th bid and* A_j*'s are the auction authorities which are required to be honest.*

The restricted channel names model private channels or secret keys. Note that we only model the honest authorities as unspecified parties are subsumed by the attacker.

By abuse of notation we write $b_l > b_o$ to express that the bidding price determined by the substitution σ_{b_l} is greater than the one assigned by σ_{b_o}, and $\max_i\{b_i\}$ denotes the maximal price assigned by any substitution σ_{b_i}. We also denote by $\arg\max_i\{b_i\}$ the set of values i for which b_i corresponds to the maximal price.

Example 20 *Consider the following simple auction protocol.*

Informal description: *Each bidder encrypts his bid using the auctioneer's public key and signs it using his secret key. The resulting bid is posted on the bulletin board. After the deadline is over, the auctioneer checks if each ballot is signed by an eligible bidder. He then decrypts the bids and determines the winner.*

Formal description in our model: *The protocol uses probabilistic public-key encryption and signatures, which we model using the following equational theory already used in Example 12 on page 55:*

$$\begin{aligned} \mathtt{dec}(\mathtt{enc}(m, \mathtt{pk}(sk), r), sk) &= m \\ \mathtt{checksign}(\mathtt{sign}(m, sk), \mathtt{pk}(sk)) &= m \end{aligned}$$

The protocol is then a tuple $(B, S, A, \emptyset)$ *where*

$$\begin{aligned}
A &= \texttt{in}(ch, (sig_1, bid_1)). \ldots \texttt{in}(ch, (sig_n, bid_n)).\\
&\quad \texttt{if checksign}(sig_1, pkb_1) = bid_1 \;\&\&\; \ldots\\
&\quad \&\& \;\texttt{checksign}(sig_n, pkb_n) = bid_n \;\texttt{then}\\
&\quad \texttt{let}\; b_1 = \texttt{dec}(bid_1, ska)\; \texttt{in}\; \ldots\\
&\quad \texttt{let}\; b_n = \texttt{dec}(bid_n, ska)\; \texttt{in}\\
&\quad \texttt{if max}(b_1, \ldots, b_n) = b_1 \texttt{then}\\
&\quad \texttt{out}(res, (sign((b_1, 1), ska), (b_1, 1)))\\
&\quad \texttt{else if}\; \ldots\\
&\quad \texttt{else out}(res, (sign((b_n, n), ska), (b_n, n)))\\
B &= \nu r.\texttt{let}\; ebid = \texttt{enc}(b, pka, r)\; \texttt{in}\\
&\quad \texttt{out}(ch, (\texttt{sign}(ebid, skb), ebid))\\
S &= \texttt{in}(res, (sig, (price, id))).\\
&\quad \texttt{if checksign}(sig, pka) = (price, id)\; \texttt{then}\; \ldots
\end{aligned}$$

In this example the substitution determining the identity of a bidder assigns the secret key, e.g. $\sigma_{id_k} = \{^{sk_k}/_{skb}\}$*. The substitution specifying the bid would be e.g.* $\sigma_{b_k} = \{^{price_k}/_{b}\}$*.*

Similarly to voting, we also define *Honest Auction Processes*, and *(Honest) Auction Contexts.*

Definition 44 (Honest Auction Process) *An* Honest Auction Process *of an auction protocol* $(B,\ S,\ A_1,\ \ldots,\ A_m, \tilde{m})$ *is a closed plain process*

$$\nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_1}| \ldots |B\sigma_{id_k}\sigma_{b_k}|S|A_1| \ldots |A_m)$$

where $\tilde{n}$ *includes the secret channel names* $\tilde{m}$*,* $B\sigma_{id_i}\sigma_{b_i}$ *are the processes executed by the* k *bidders,* σ_{id_i} *is a substitution assigning the identity to the* i*-th bidder,* σ_{b_i} *specifies the* i*-th bid and* A_j*'s are the auction authorities.*

Given an auction process

$$AP = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_1}| \ldots |B\sigma_{id_k}\sigma_{b_k}|S|A_1| \ldots |A_l)$$

we denote by AP^H *the corresponding honest auction process, i.e.*

$$AP^H = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_1}| \ldots |B\sigma_{id_k}\sigma_{b_k}|S|A_1| \ldots |A_m)$$

Definition 45 ((Honest) Auction Context) *Given an auction process*

$$AP = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_1}| \ldots |B\sigma_{id_k}\sigma_{b_k}|S|A_1| \ldots |A_l)$$

and a subset of bidders I we define the Auction Context $AP_I[\_]$ *as follows:*

$$AP_I[\_] = \nu\tilde{n}.(\ |_{i\notin I} B\sigma_{id_i}\sigma_{b_i}|\_|S|A_1|\ldots|A_l)$$

Similarly for an honest auction process

$$AP^H = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_1}|\ldots|B\sigma_{id_k}\sigma_{b_k}|S|A_1|\ldots|A_m)$$

and a subset of bidders I we define the Honest Auction Context $AP^H_I[\_]$ *as follows:*

$$AP^H_I[\_] = \nu\tilde{n}.(\ |_{i\notin I} B\sigma_{id_i}\sigma_{b_i}|\_|S|A_1|\ldots|A_m)$$

Example 21 *An instance of our simple auction protocol (Example 20 on page 163) with two bidders A and B looks as follows:*

$$AP = \nu sk_A.\nu sk_B.\nu sk_{Ad}.(S\,\{^{\mathtt{pk}(sk_{Ad})}/_{pka}\}\,|B\,\{^{sk_A}/_{skv},{}^{\mathtt{pk}(sk_{Ad})}/_{pka}\}\,\{^{price_A}/_{b}\}\,|\\ B\,\{^{sk_B}/_{skv},{}^{\mathtt{pk}(sk_{Ad})}/_{pka}\}\,\{^{price_B}/_{b}\}\,|A\,\{^{sk_{Ad}}/_{ska},{}^{\mathtt{pk}(sk_A)}/_{pkb_1},{}^{\mathtt{pk}(sk_B)}/_{pkb_2}\})$$

The corresponding auction context $AP_{\{B\}}$ then looks as follows:

$$AP_{\{B\}} = \nu sk_A.\nu sk_B.\nu sk_{Ad}.(S\,\{^{\mathtt{pk}(sk_{Ad})}/_{pka}\}\,|B\,\{^{sk_A}/_{skv},{}^{\mathtt{pk}(sk_{Ad})}/_{pka}\}\,\{^{price_A}/_{b}\}\\ |\_|A\,\{^{sk_{Ad}}/_{ska},{}^{\mathtt{pk}(sk_A)}/_{pkb_1},{}^{\mathtt{pk}(sk_B)}/_{pkb_2}\})$$

Note that as this protocol assumes an honest auctioneer, the honest auction process AP^H and the "normal" auction process AP, as well as the honest auction context $AP^H_I[\_]$ and the "normal" auction context $AP_I[\_]$ coincide.

In order to reason about reachability and authentication properties we use *events*. Events are annotations marking important steps in the protocol execution, but otherwise do not influence the behavior of processes. They allow us to verify properties such as "event `bad` is unreachable" or "on every trace event `a` is preceded by event `b`".

We use the same technique as in [ABF07, SRKK10]: Events are outputs $\mathtt{out}(e,(M_1,\ldots,M_n))$ on a special *"event" channel* $e \in E$, where E is the set of event channels, disjoint of the set of "ordinary" channels. To ensure that the adversary cannot obtain additional knowledge from the events or create events himself, we only consider traces respecting the following conditions when reasoning about processes containing events:

— Event names occur only in outputs; they are neither communicated nor used for inputs in processes and in transitions.

— Names and variables extruded in events do not appear in inputs unless they

have also been sent on other output channels.

In accordance with the syntax used by ProVerif, we also write `event` $e(M_1, \ldots, M_n)$ instead of `out`$(e, (M_1, \ldots, M_n))$, where e is the name of the event (channel), and the terms $M_1, \ldots, M_n$ are parameters.

For auction protocols we use the following events:

— `bid(p,id)`: When a bidder `id` bids the price `p` the event `bid(p,id)` is emitted.

— `recBid(p,id)`: When a bid at price `p` by bidder `id` is recorded by the auctioneer/bulletin board/etc. the event `recBid(p,id)` is called. This is used to model Non-Cancellation, i.e. from this point on a bid is considered binding.

— `won(p,id)`: When a bidder `id` wins the auction at price `p`, the event `won(p,id)` is emitted.

4.3.2 — Fairness Properties

A fair auction protocol should not leak any information about any participant until the bidding phase is over and the winning bid is announced, and hence some information is inevitably leaked. We propose the following two definitions:

Definition 46 (Strong Noninterference (SN)) *An auction protocol ensures* Strong Noninterference (SN) *if for any two auction processes APA and APB that halt at the end of the bidding phase (i.e. where we remove all code after the last* `recBid` *event) we have*

$$APA \approx_l APB.$$

This notion is very strong: Any two instances, independently of the participants and their offers, are required to be bisimilar until the end of the bidding phase. This would also require two instances with a different number of participants to be bisimilar, which will probably not hold on many protocols. A more realistic notion is the following:

Definition 47 (Weak Noninterference (WN)) *An auction protocol ensures* Weak Noninterference (WN) *if for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ that halt at the end of the bidding phase (i.e. where we remove all code after the last* `recBid` *event) we have*

$$APA \approx_l APB.$$

This only requires any two instances with the same participants $B\sigma_{id_i}$ to be bisimilar, however bids may still change. It is easy to see that (SN) implies (WN).

Theorem 48 *An auction protocol ensuring Strong Noninterference (SN) also ensures Weak Noninterference (WN).*

Proof We have to show that for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ that halt at the end of the bidding phase (i.e. where we remove all code after the last `recBid` event) we have $APA \approx_l APB$. By (SN) we have that for any two auction processes APA and APB that halt at the end of the bidding phase (i.e. where we remove all code after the last `recBid` event) we have $APA \approx_l APB$, hence we can conclude directly. □

However the above two notions only capture that no information is leaked, however we also want to ensure that the bidders cannot cheat. The following property models that there should be no way for a malicious participant to cheat and win the auction at a chosen price, independently of the other (higher) bids.

Definition 48 (Highest Price Wins (HPW)) *An electronic auction protocol ensures* Highest Price Wins (HPW) *if for any auction process AP we have for* $AP_{\{A,B\}}[\ B\sigma_{id_A}\sigma_{b_A} \mid (B\sigma_{id_B}\sigma_{b_B})^{c_1,c_2}]$ *where* b_A *is the highest bid, there is no trace containing the event* `won` *for bidder* id_B *with a lower bid.*

The idea is the following: We have an honest bidder $B\sigma_{id_A}$ who submits the highest bid. The attacker has completely corrupted another bidder $B\sigma_{id_B}$ and should be unable to win the auction on his behalf using a lower bid.

Note that these definitions can be applied independently of trust assumptions, and that different assumptions can lead to different results: For example, a protocol might ensure (HPW) if the auctioneer is trusted, but not otherwise.

4.3.3 — Authentication Properties

The first authentication property we want to define is *Non-Repudiation*, i.e. that – once the winner has been announced – a winning bidder cannot claim that the winning bid was not sent by him. As discussed in [KV09], Non-Repudiation can be expressed as form of authentication.

Definition 49 (Non-Repudiation) *An auction protocol ensures the property of* Non-Repudiation (NR) *if for every auction process AP on every possible execution trace the event* `won(p,id)` *is preceded by a corresponding event* `bid(p,id)`.

The intuition is simple: If there was a trace on which a bidder would win without submitting the winning bid, he could try to claim that he did not submit the winning bid even in a case where he rightfully won.

Note two subtleties with this definition: Firstly, since only honest parties are explicitly modeled, it is clear that only honest parties can emit events. Hence

one could think that our definition implicitly assumes some parties to be honest – however, this is not the case: If we do not trust the party that would normally emit for example the event `won`, we can simply remove this party from the model and replace it with a new party that receives the parameters on a special channel, and then emits the event using these parameters. This gives the adversary total control about the events, as it would be the case for a distrusted authority[3]. Secondly, if the protocol supports multiple auctions in parallel, we need to use auction-dependent identifiers for the bidders in our events. This is to ensure that the protocol only accepts bids that were submitted to the same auction, and that an attacker cannot submit a bid from a different auction.

The second authentication property we model is *Non-Cancellation*, i.e. that a bidder cannot cancel a submitted bid before the winner is announced.

Definition 50 (Non-Cancellation) *An auction protocol ensures the property of* Non-Cancellation (NC) *if given any auction process AP for the corresponding process* $AP_{\{i\}}\left[(B\sigma_{id_i}\sigma_{b_i})^{chc}\right]$*, i.e. containing a bidder i which*

— *reveals his secret data on channel* `chc` *(see Def. 18 on page 59), and which*

— *submits the highest bid, i.e.* $\forall j \neq i : b_i > b_j$*,*

there is no trace containing the events $\texttt{recBid}(b_i, id_i)$ *and* $\texttt{won}(b_w, id_w)$ *for another, lower bid, i.e.* $b_w < b_i$*.*

The idea is the following: The bidder id_i submits the highest bid, so he should win. If however there is the possibility that even though his bid was correctly received he did not win, this would mean that the intruder was able to cancel the bidder's bid even after reception. We require the bidder to reveal all his secret data to the intruder to capture the fact that the bidder himself might want to cancel his offer, in which case he could use his private data (keys etc.) to do so.

Note that technically we only defined Non-Cancellation for the winning bidder. This is sufficient since in a first-price auction the other bids do not influence the outcome. Additionally it can be generalized to other auction types by simply requiring that the winning price must be correct on all traces. This is to ensure that no other bids that influence the result can be canceled.

Both properties are independent: A protocol may implement the cancellation of bids as an official feature. For example after all bids have been submitted, bidders could be allowed to cancel their bids for a certain period of time, before the winner is finally announced. At the same time, such a protocol may ensure non-repudiation of the winner using e.g. signatures. Similarly a protocol may ensure Non-Cancellation but no Non-Repudiation if the submitted bids cannot be

[3]Note however that our definition also *allows* to have trusted parties. It is hence not as strong as other definitions in the literature, which require sufficient evidence to convince an impartial judge, and thus usually exclude any trust on parties participating in the protocol execution.

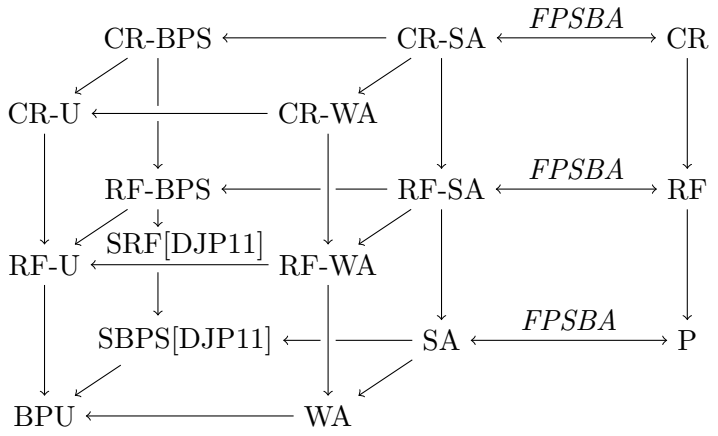

Figure 4.1 – Relations among the privacy notions. A $\xrightarrow{C}$ B means that under the assumption C a protocol ensuring A also ensures B.

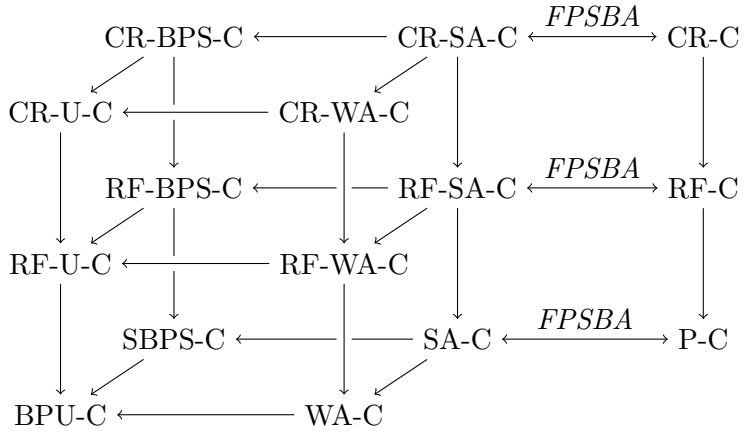

Figure 4.2 – Relations among the privacy notions including corrupted bidders. A $\xrightarrow{C}$ B means that under the assumption C a protocol ensuring A also ensures B.

canceled, but are not authenticated, so that the announced winner can successfully claim not having submitted the winning bid. Again, a protocol might ensure Non-Cancellation or Non-Repudiation for a certain trust setting, but not for another.

4.3.4 — Privacy Properties

We consider Privacy, Receipt-Freeness and Coercion-Resistance, and at each level two independent axes:

— the winner may stay anonymous (Strong Anonymity (SA, RF-SA, CR-SA) and Weak Anonymity (WA, RF-WA, CR-WA)) or not (Strong Bidding-Price Secrecy (SBPS, RF-BPS, CR-BPS) and Bidding-Price Unlinkability (BPU, RF-U, CR-U))

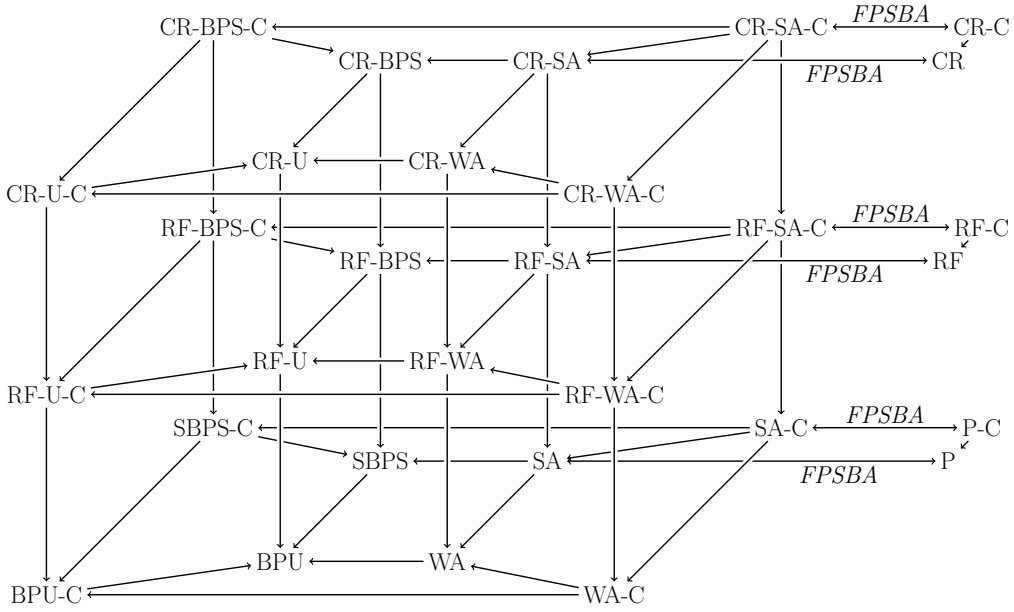

Figure 4.3 – Relations between the privacy notions with and without corrupted bidders. A $\xrightarrow{C}$ B means that under the assumption C a protocol ensuring A also ensures B.

— the losing bids may stay completely private (Strong Bidding-Price Secrecy (SBPS, RF-BPS, CR-BPS) and Strong Anonymity (SA, RF-SA, CR-SA)), or there might be list of all bids, which are however unlinkable to the bidders (Bidding-Price Unlinkability (BPU, RF-U, CR-U) and Weak Anonymity (WA, RF-WA, CR-WA)).

These definitions are expressed for protocols implementing a first-price sealed-bid auction. We also provide the generalized notions (P), (RF) and (CR), which can also be applied to other types of auctions such as second-price auctions. We show that if a protocol correctly implements a First-Price Sealed-Bid Auction (FPSBA), these notions coincide with the corresponding Strong Anonymity-notions (SA), (RF-SA) and (CR-SA). We also propose variants of the above notions including corrupted bidders. Figures 4.1 on the previous page, 4.2 on the preceding page and 4.3 provide overviews of the different notions and their relations.

§ 4.3.4.1. Privacy. The first privacy notion we consider was proposed by Dong et al. [DJP11].

Definition 51 (Strong Bidding-Price Secrecy (SBPS) [DJP11]) *An electronic auction protocol ensures* Strong Bidding-Price Secrecy (SBPS) *if for any auction process AP and any bids* $b_A < b_C, b_B < b_C$ *we have*

$$AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}\right]$$

The intuition is the following: If the losing bids are private, a losing bidder may change his bid for another losing one without this being noticeable to an attacker. This is expressed as an observational equivalence between two situations where a losing bidder changes his bid. Note that $B\sigma_{id_C}$ does not necessarily win since in AP' there might be a bidder offering a higher price, but $b_A, b_B < b_C$ guarantees that $B\sigma_{id_A}$ loses.

We propose the following, weaker notion of Bidding-Price Unlinkability, which allows the losing bids to be public, however their link to the bidders have to be secret.

Definition 52 (Bidding-Price Unlinkability (BPU)) *An electronic auction protocol ensures* Bidding-Price Unlinkability (BPU) *if for any auction process AP and any bids $b_A < b_C, b_B < b_C$ we have*

$$AP_{\{A,B,C\}}\,[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_B}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}] \approx_l AP_{\{A,B,C\}}\,[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_B}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}]$$

In this definition we require two situations in which two losing bidders swap their bids to be bisimilar. This might be the case if the bids are public, but the real identity of the bidders is hidden, e.g. through the use of pseudonyms.

Theorem 49 *If an auction protocol ensures Strong Bidding-Price Secrecy (SBPS), it also ensures Bidding-Price Unlinkability (BPU).*

Proof As we have (SBPS), suppose for any auction process AP and any bids $b_A < b_C, b_B < b_C$ we have

$$AP_{\{A,C\}}\,[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}] \approx_l AP_{\{A,C\}}\,[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}]$$

We have to show that for any auction process AP and any bids $b_A, b_B < b_C$ we have

$$AP_{\{A,B,C\}}\,[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_B}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}] \approx_l AP_{\{A,B,C\}}\,[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_B}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}]$$

Applying the hypothesis, can conclude as follows:

$$\begin{aligned} & AP_{\{A,B,C\}}\,[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_B}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}] \\ \approx_l\ & AP_{\{A,B,C\}}\,[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_B}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}] \\ \approx_l\ & AP_{\{A,B,C\}}\,[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_B}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}] \end{aligned}$$

□

Note that the previous two notions only concern the losing bids, yet we might also want to preserve the anonymity of the winning bidder.

Definition 53 (Strong Anonymity (SA)) *An electronic auction protocol ensures* Strong Anonymity (SA) *if for an auction process AP and any bids* $b_A < b_C, b_B < b_C$ *we have*

$$AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_B}\right]$$

Here we require two situations to be bisimilar where two different bidders win using the same offer, and the losing bidders may also use different bids in the two cases. This is stronger than Strong Bidding-Price Secrecy (SBPS).

Theorem 50 *If an auction protocol ensures Strong Anonymity (SA), it also ensures Strong Bidding-Price Secrecy (SBPS).*

Proof We have to show (SBPS), i.e. that for any auction process AP and any bids $b_A < b_C, b_B < b_C$ we have

$$AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}\right]$$

Suppose that for any auction process AP and any bids $b_A < b_C, b_B < b_C$ we have

$$AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_B}\right]$$

Then we can conclude as follows.

$$\begin{array}{ll} & AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \\ \approx_l & AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_B}\right] \\ \approx_l & AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}\right] \end{array}$$

□

A slightly weaker notion is Weak Anonymity, which allows the bids to be public, however their link to the bidders have to be secret, even for the winner.

Definition 54 (Weak Anonymity (WA)) *An electronic auction protocol ensures* Weak Anonymity (WA) *if for an auction process AP and any bids* $b_A < b_C$ *we have*

$$AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_A}\right]$$

Here again two different bidders win using the same bid, but the losing bidder cannot choose his bid freely as above - the two bidders swap their bids. This corresponds for example to a situation with a public list of bids in clear, but where it is private which bidder submitted which bid.

Theorem 51 *If an auction protocol ensures Strong Anonymity (SA), it also ensures Weak Anonymity (WA).*

Proof We have to show that for any auction process AP and any bids $b_A < b_C$ we have

$$AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_A}\right]$$

Suppose that for any auction process AP and any bids $b_A < b_C, b_B < b_C$ we have

$$AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_B}\right]$$

We set $b_B = b_A$ to conclude. □

Weak Anonymity (WA) is stronger than Bidding-Price Unlinkability (BPU) as even the winner remains anonymous.

Theorem 52 *If an auction protocol ensures Weak Anonymity (WA), it also ensures Bidding-Price Unlinkability (BPU).*

Proof We have to show that for any auction process AP and any bids $b_A < b_C, b_B < b_C$ we have

$$\begin{aligned} &AP_{\{A,B,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_B}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}\right] \\ &\qquad\qquad \approx_l AP_{\{A,B,C\}}\left[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_B}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \end{aligned}$$

Assume that for any auction process AP and any bids $b_A < b_C$ we have

$$AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_A}\right]$$

Then we can proceed as follows.

$$\begin{aligned} & AP_{\{A,B,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_B}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}\right] \\ \approx_l\ & AP_{\{A,B,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_B}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_B}\right] \\ \approx_l\ & AP_{\{A,B,C\}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_B}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_B}\right] \\ \approx_l\ & AP_{\{A,B,C\}}\left[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_B}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \end{aligned}$$

□

All these definitions are only meaningful for first-price auctions. To also deal with second-prices sealed-bid auctions, we can use the following generalization based on the published result (analogous to the definitions for weighted votes in Section 3.4).

Definition 55 (Privacy (P)) *An electronic auction protocol ensures* Privacy (P) *if for any two auction processes* $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and* $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *we have*

$$APA^H|_{res} \approx_l APB^H|_{res} \Rightarrow APA \approx_l APB$$

The intuition is quite simple: any two instances (consisting of the same bidders) which give the same result, for example the same winning bid, have to be bisimilar.

It turns out that for a correct first-price sealed-bid auction protocol which only publishes the winning price, this coincides with Strong Anonymity.

Definition 56 (First-Price Sealed-Bid Auction (FPSBA)) *An electronic auction protocol implements a* First-Price Sealed-Bid Auction (FPSBA) *if for any two auction processes* $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and* $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *we have*

$$APA^H|_{res} \approx_l APB^H|_{res} \Leftrightarrow \left(\max_i b_{i,A} = \max_i b_{i,B} \wedge |\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|\right)$$

This definition requires the protocol to announce the same result if and only if the maximum among the submitted bids is the same and we have the same number of tied bids, independently of which bidder submitted which bid. Without the second condition we would require two situations to give the same result even if in one there is a tie, but not in the other. This would be too strict for many protocols that identify and publish ties, but not automatically break them.

It is easy to see that both conditions hold in the case of a correct first-price sealed-bid auction protocol. This allows us to prove the equivalence of (P) and (SA).

Theorem 53 *If an electronic auction protocol implements a First-Price Sealed-Bid Auction (FPSBA), then Privacy (P) and Strong Anonymity (SA) are equivalent.*

Proof We prove both implications separately:

— Suppose an electronic auction protocol ensures (P), i.e. if for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ we have

$$APA^H|_{res} \approx_l APB^H|_{res} \Rightarrow APA \approx_l APB$$

We want to show (SA), i.e. that for any an auction process AP and any bids

$b_A < b_C, b_B < b_C$ we have

$$\begin{aligned} APA := \quad & AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A} | B\sigma_{id_C}\sigma_{b_C}\right] \\ & \approx_l \\ & AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_C} | B\sigma_{id_C}\sigma_{b_B}\right] \quad =: APB \end{aligned}$$

We can see that both sides have the number of maximal bids containing the same price, hence by (FPSBA) we obtain that $APA^H|_{res} \approx_l APB^H|_{res}$, which allows to conclude using (P).

— Suppose that we have (SA) and we want to show (P). Suppose $APA^H|_{res} \approx_l APB^H|_{res}$, then by (FPSBA) we have the same number of maximal bids containing the same price on both sides. If the winning bidder(s) did not change, only losing bids may have changed, and we can use a sequence of applications of (SA) to conclude (as in the proof of Theorem 50 on page 172). If the winning bidder(s) did change, (SA) allows us to swap the winning bid, and then – as above – we can use a sequence of applications of (SA) to conclude for the other bids. □

These definitions do not include corrupted bidders, which can be realized as follows. For better readability we mark the differences to the corresponding definitions without corrupted bidders in **bold**.

Definition 57 (SBPS with Corrupted Bidders (SBPS-C)) *An electronic auction protocol ensures* Strong Bidding-Price Secrecy with Corrupted Bidders (SBPS-C) *if for an auction process AP, any bids* $b_A < b_C, b_B < b_C$ ***and any subset of bidders*** $\boldsymbol{I}$ ***with*** $\boldsymbol{I \cap \{A, C\} = \emptyset}$ *we have*

$$\begin{aligned} & AP_{\{A,C\}\cup \boldsymbol{I}}\left[B\sigma_{id_A}\sigma_{b_A} | B\sigma_{id_C}\sigma_{b_C} \mid_{\boldsymbol{i\in I}} (\boldsymbol{B\sigma_{id_i}})^{\boldsymbol{c_{1i},c_{2i}}}\right] \\ & \qquad \approx_l AP_{\{A,C\}\cup \boldsymbol{I}}\left[B\sigma_{id_A}\sigma_{b_B} | B\sigma_{id_C}\sigma_{b_C} \mid_{\boldsymbol{i\in I}} (\boldsymbol{B\sigma_{id_i}})^{\boldsymbol{c_{1i},c_{2i}}}\right] \end{aligned}$$

This definition is identical to the definition of (SBPS), except that any subset of the other bidders in the instance can be corrupted.

We can employ the same approach for Bidding-Price Unlinkability, Weak Anonymity and Weak Anonymity.

Definition 58 (BPU with Corrupted Bidders (BPU-C)) *An electronic Auction protocol ensures* Bidding-Price Unlinkability with Corrupted Bidders (BPU-C) *if for an auction process AP, any bids* $b_A < b_C, b_B < b_C$ ***and any subset of***

bidders I with $I \cap \{A, B, C\} = \emptyset$ *we have*

$$AP_{\{A,B,C\}\cup \boldsymbol{I}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_B}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C} \mid_{\boldsymbol{i\in I}} (\boldsymbol{B\sigma_{id_i}})^{\boldsymbol{c_{1i},c_{2i}}}\right]$$
$$\approx_l AP_{\{A,B,C\}\cup \boldsymbol{I}}\left[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_B}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C} \mid_{\boldsymbol{i\in I}} (\boldsymbol{B\sigma_{id_i}})^{\boldsymbol{c_{1i},c_{2i}}}\right]$$

Definition 59 (Strong Anonymity with Corrupted Bidders (SA-C)) *An electronic auction protocol ensures* Strong Anonymity with Corrupted Bidders (SA-C) *if for an auction process AP, any bids $b_A < b_C, b_B < b_C$* ***and any subset of bidders I with $I \cap \{A, C\} = \emptyset$*** *we have*

$$AP_{\{A,C\}\cup \boldsymbol{I}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C} \mid_{\boldsymbol{i\in I}} (\boldsymbol{B\sigma_{id_i}})^{\boldsymbol{c_{1i},c_{2i}}}\right]$$
$$\approx_l AP_{\{A,C\}\cup \boldsymbol{I}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_B} \mid_{\boldsymbol{i\in I}} (\boldsymbol{B\sigma_{id_i}})^{\boldsymbol{c_{1i},c_{2i}}}\right]$$

Definition 60 (Weak Anonymity with Corrupted Bidders (WA-C)) *An electronic auction protocol ensures* Weak Anonymity with Corrupted Bidders (WA-C) *if for an auction process AP, any bids $b_A < b_C$* ***and any subset of bidders I with $I \cap \{A, C\} = \emptyset$*** *we have*

$$AP_{\{A,C\}\cup \boldsymbol{I}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C} \mid_{\boldsymbol{i\in I}} (\boldsymbol{B\sigma_{id_i}})^{\boldsymbol{c_{1i},c_{2i}}}\right]$$
$$\approx_l AP_{\{A,C\}\cup \boldsymbol{I}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_A} \mid_{\boldsymbol{i\in I}} (\boldsymbol{B\sigma_{id_i}})^{\boldsymbol{c_{1i},c_{2i}}}\right]$$

Analogous to our definitions for eVoting, we can also include corrupted bidders in the generalized definition.

Definition 61 (Privacy with Corrupted Bidders (P-C)) *An electronic auction protocol ensures* Privacy with Corrupted Bidders (P-C) *if for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$* ***and any subset of bidders $\boldsymbol{I \subseteq \{1, \ldots, n\}}$ with $\boldsymbol{\forall i \in I : \sigma_{b_{i,A}} = \sigma_{b_{i,B}}}$*** *we have*

$$APA^H|_{res} \approx_l APB^H|_{res}$$
$$\Rightarrow APA_I\left[\mid_{\boldsymbol{i\in I}} (\boldsymbol{B\sigma_{id_i}})^{\boldsymbol{c_{1i},c_{2i}}}\right] \approx_l APB_I\left[\mid_{\boldsymbol{i\in I}} (\boldsymbol{B\sigma_{id_i}})^{\boldsymbol{c_{1i},c_{2i}}}\right]$$

Then we have the following hierarchy between the different notions.

Theorem 54 *We have for any $X \in \{P, SBPS, BPU, SA, WA\}$:*

— *Any auction protocol ensuring (X-C) also ensures (X).*

— *Any auction protocol ensuring (SBPS-C) also ensures (BPU-C).*

— *Any auction protocol ensuring (SA-C) also ensures (SBPS-C).*

— *Any auction protocol ensuring (SA-C) also ensures (WA-C).*

— *Any auction protocol ensuring (WA-C) also ensures (BPU-C).*

— *If an auction protocol implements (FPSBA), then (P-C) and (SA-C) are equivalent.*

Proof The first proposition is easy to show, we simply consider $I = \emptyset$. The remaining five propositions can be shown analogously to Theorems 49, 50, 51, 52 and 53. We simply add the set of corrupted bidders, the rest of the proofs remains unchanged. □

§ 4.3.4.2. Receipt-Freeness. A first Receipt-Freeness definition for auction protocols was proposed by Dong et al. [DJP11]. It is a generalization of Strong Bidding-Price Secrecy (SBPS).

Definition 62 (Simple Receipt-Freeness (SRF) [DJP11]) *An electronic auction protocol ensures* Simple Receipt-Freeness (SRF) *if for an auction process AP and any bids $b_A < b_C, b_B < b_C$ there exists a process B' such that*

$$B'^{\backslash out(chc,\cdot)} \approx_l B\sigma_{id_A}\sigma_{b_B}$$

and

$$AP_{\{A,C\}}\left[(B\sigma_{id_A}\sigma_{b_A})^{chc} | B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B' | B\sigma_{id_C}\sigma_{b_C}\right]$$

The intuition behind this definition is a follows: If the protocol is receipt-free, an attacker cannot distinguish between a situation where a losing bidder bids b_A and reveals all his secret data on a channel *chc*, and a situation where the bidder bids b_B and only pretends to reveal his secret data (the fake strategy, modeled by process B'). Note that Simple Receipt-Freeness (SRF) implies Strong Bidding-Price Secrecy (SBPS).

Theorem 55 *An auction protocol ensuring Simple Receipt-Freeness (SRF) also ensures Strong Bidding-Price Secrecy (SBPS).*

Proof We have to show that for any auction process AP and any bids $b_A < b_C, b_B < b_C$ we have

$$AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A} | B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_B} | B\sigma_{id_C}\sigma_{b_C}\right]$$

By hypothesis we have that for any auction process AP and any bids $b_A < b_C, b_B < b_C$ there exists a process B' such that

$$B'^{\backslash out(chc,\cdot)} \approx_l B\sigma_{id_A}\sigma_{b_B}$$

and

$$AP_{\{A,C\}}\left[(B\sigma_{id_A}\sigma_{b_A})^{chc}\,|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B'|B\sigma_{id_C}\sigma_{b_C}\right]$$

We apply the context $\nu chc_{id_C}(\_|!\mathtt{in}(chc_{id_C}, x))$ on both sides, which gives

$$\begin{gathered} AP_{\{A,C\}}\left[(B\sigma_{id_A}\sigma_{b_A})^{chc}|B\sigma_{id_C}\sigma_{b_C}\right]^{\backslash out(chc,\cdot)} \\ \approx_l \\ AP_{\{A,C\}}\left[B'|B\sigma_{id_C}\sigma_{b_C}\right]^{\backslash out(chc,\cdot)} \end{gathered}$$

By using Lemma 17 on page 60 we obtain

$$\begin{gathered} AP_{\{A,C\}}\left[(B\sigma_{id_A}\sigma_{b_A})^{chc}|B\sigma_{id_C}\sigma_{b_C}\right]^{\backslash out(chc,\cdot)} \\ \equiv \\ AP_{\{A,C\}}\left[\left((B\sigma_{id_A}\sigma_{b_A})^{chc}\right)^{\backslash out(chc,\cdot)}|B\sigma_{id_C}\sigma_{b_C}\right] \end{gathered}$$

and

$$AP_{\{A,C\}}\left[B'|B\sigma_{id_C}\sigma_{b_C}\right]^{\backslash out(chc,\cdot)} \equiv AP_{\{A,C\}}\left[(B')^{\backslash out(chc,\cdot)}|B\sigma_{id_C}\sigma_{b_C}\right]$$

We can now apply Lemma 16 on page 60 and use the fact that labeled bisimilarity is closed under structural equivalence and obtain

$$AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[(B')^{\backslash out(chc,\cdot)}|B\sigma_{id_C}\sigma_{b_C}\right]$$

where we can use $(B')^{\backslash out(chc,\cdot)} \approx_l B\sigma_{id_A}\sigma_{b_B}$ to conclude. □

The definition by Dong et al. has several shortcomings: Firstly, it ensures receipt-freeness only for one losing bidder, whereas in reality several bidders might be under attack. Secondly, it does not necessary ensures the privacy of other bidders: Consider for example a protocol that allows a losing bidder to create a fake receipt for himself, e.g. using a trapdoor to generate a different decryption key, and that reveals all submitted bids to the participating bidders, e.g. to enable verifiability. Such a protocol would be secure according to above definition, but it would allow a coercer to ask a bidder to reveal the other participants bids, violating their privacy[4]. To address these issues, we propose the following notions, inspired by the definitions developed for electronic voting in the previous chapter and the above privacy notions.

[4]We observed a similar problem for voting protocols in Example 19 on page 135.

Definition 63 (RF-XXX) *An electronic auction protocol ensures* RF-XXX *if for any two auction processes* $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and* $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *such that*

- *if XXX=BPS* (Bidding-Price-Secrecy)*, for* $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ *with* $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ *and for any subset* $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$,
- *if XXX=U* (Unlinkability)*, for* $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ *with* $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ *and a permutation* Π *with* $\forall i : b_{i,B} = b_{\Pi(i),A}$*, and for any subset* $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$,
- *if XXX=SA* (Strong Anonymity)*,* $\max_i b_{i,A} = \max_i b_{i,B}$ *and* $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$*, and for any subset* $I \subseteq \{1,\ldots,k\}$,
- *if XXX=WA* (Weak Anonymity)*, there exists a permutation* Π *with* $\forall i : b_{i,B} = b_{\Pi(i),A}$*, and for any subset* $I \subseteq \{1,\ldots,k\}$,

there exist processes B'_i *such that we have*

$$\forall i \in I : B_i'^{\setminus out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA_I\left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i}\right] \approx_l APB_I\left[\mid_{i\in I} B'_i\right]$$

Consider the first case, (RF-BPS): In this definition any subset of losing bidders may create fake receipts at the same time, and the other bidders can also change their bids. It is easy to see that this definition implies Simple Receipt-Freeness (SRF).

Theorem 56 *An auction protocol ensuring Receipt-Free Bidding-Price-Secrecy (RF-BPS) also ensures Simple Receipt-Freeness (SRF).*

Proof By hypothesis we have for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ such that for $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ with $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ and for any subset $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$ there exist processes B'_i such that we have $\forall i \in I : B_i'^{\setminus out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$ and

$$APA_I\left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i}\right] \approx_l APB_I\left[\mid_{i\in I} B'_i\right]$$

We have to show that for any auction process AP and any bids $b_A, b_B < b_C$ there exists a process B' such that

$$B'^{\setminus out(chc,\cdot)} \approx_l B\sigma_{id_A}\sigma_{b_B}$$

and

$$AP_{\{A,C\}}\left[(B\sigma_{id_A}\sigma_{b_A})^{chc}\,|B\sigma_{id_C}\sigma_{b_C}\right] \approx_l AP_{\{A,C\}}\left[B'|B\sigma_{id_C}\sigma_{b_C}\right]$$

We can conclude by setting

$$\begin{aligned} APA &= AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right], \\ APB &= AP_{\{A,C\}}\left[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_B}\right] \end{aligned}$$

and $I = \{C\}$, where $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ holds. □

Similarly to our privacy definitions, we can also weaken (RF-BPS) and only consider cases where the bids are merely unlinkable to the bidders, by only considering permutations of the bids: We obtain (RF-U), which implies (BPU).

Theorem 57 *An auction protocol ensuring Receipt-Free Unlinkability (RF-U) also ensures Bidding-Price Unlinkability (BPU).*

Proof We have to show that for any auction process AP and any bids $b_A < b_C, b_B < b_C$ we have

$$\begin{aligned} &AP_{\{A,B,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_B}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}\right] \\ &\qquad\qquad \approx_l AP_{\{A,B,C\}}\left[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_B}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \end{aligned}$$

By hypothesis we have for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ such that for $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ with $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ and a permutation Π with $\forall i : b_{i,B} = b_{\Pi(i),A}$, and for any subset $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$, there exist processes B'_i such that we have

$$\forall i \in I : B_i'^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA_I\left[\,|_{i\in I}\,(B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i}\right] \approx_l APB_I\left[\,|_{i\in I}\,B'_i\right]$$

We can easily see that if we choose

$$\begin{aligned} APA &= AP_{\{A,B,C\}}\left[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_B}\sigma_{b_B}|B\sigma_{id_C}\sigma_{b_C}\right] \\ APB &= AP_{\{A,B,C\}}\left[B\sigma_{id_A}\sigma_{b_B}|B\sigma_{id_B}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}\right] \end{aligned}$$

this fits the definition of (RF-U), i.e. we have for $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ that $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ holds, and a permutation Π with $\forall i : b_{i,B} = b_{\Pi(i),A}$, hence we can choose $I = \emptyset$ and conclude. □

The third notion (RF-SA) is stronger in the sense that we also allow the winning bidder to be under attack, i.e. a winner needs to be able to create a fake receipt

that proves that he lost, and a losing bidder needs to be able to create a fake receipt that proves that he won. Note that an attacker might ask a losing bidder to prove that he bid a certain price before the auction is over. If the bidder decides to bid less and create a fake receipt, the attacker may notice that he got a fake receipt if for example the winning bid is less than the price on the receipt. This is however an inherent problem of auctions, but our definition guarantees that a losing bidder can create a fake receipt for the winning price once the auction is over and the winning price is known.

Similarly to the other cases above, we have that (RF-SA) implies (SA).

Theorem 58 *An auction protocol ensuring Receipt-Free Strong Anonymity (RF-SA) also ensures Strong Anonymity (SA).*

Proof We have to show that for any auction process AP and any bids $b_A, b_B < b_C$ we have

$$AP_{\{A,C\}}\,[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}] \approx_l AP_{\{A,C\}}\,[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_B}]$$

By hypothesis we have for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ such that $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, and for any subset $I \subseteq \{1,\ldots,k\}$ there exist processes B'_i such that we have

$$\forall i \in I : B_i'^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA_I\left[\,|_{i\in I}\,(B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i}\right] \approx_l APB_I\left[\,|_{i\in I}\,B'_i\right]$$

We can easily see that if we choose

$$\begin{aligned} APA &= AP_{\{A,C\}}\,[B\sigma_{id_A}\sigma_{b_A}|B\sigma_{id_C}\sigma_{b_C}] \\ APB &= AP_{\{A,C\}}\,[B\sigma_{id_A}\sigma_{b_C}|B\sigma_{id_C}\sigma_{b_B}] \end{aligned}$$

this fits the definition of (RF-SA), i.e. we have $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, hence we can choose $I = \emptyset$ and conclude. □

Again, we can define a weaker version where the list of prices may be public, but it has to be unlinkable to the bidders, even for the winner: (RF-WA). Similarly to (RF-SA), we have that (RF-WA) implies (WA).

Theorem 59 *An auction protocol ensuring Receipt-Free Weak Anonymity (RF-WA) also ensures Weak Anonymity (WA).*

Proof As the proof of Theorem 58, but with $b_B = b_A$. □

It is easy to see that (RF-SA) implies (RF-BPS) and (RF-WA), and that both (RF-BPS) and (RF-WA) imply (RF-U).

Theorem 60 *We have:*

— *A protocol ensuring (RF-SA) also ensures (RF-BPS) and (RF-WA).*

— *A protocol ensuring (RF-BPS) or (RF-WA) also ensures (RF-U).*

Proof The only difference between these pairs is the restriction on the bids and I:

— if XXX=BPS *(Bidding-Price-Secrecy)*, for $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ with $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ and for any subset $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$,

— if XXX=U *(Unlinkability)*, for $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ with $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ and a permutation Π with $\forall i : b_{i,B} = b_{\Pi(i),A}$, and for any subset $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$,

— if XXX=SA *(Strong Anonymity)*, $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, and for any subset $I \subseteq \{1,\ldots,k\}$,

— if XXX=WA *(Weak Anonymity)*, there exists a permutation Π with $\forall i : b_{i,B} = b_{\Pi(i),A}$, and for any subset $I \subseteq \{1,\ldots,k\}$,

It is easy to see that SA includes WA, that BPS includes U, that SA includes BPS, and finally that WA includes U. □

Finally, the following definition is a generalization of Receipt-Free Strong Anonymity (RF-SA) (analogous to (P) and (SA)): Any two instance giving the same result have to be bisimilar, even if bidders are under attack.

Definition 64 (Receipt-Freeness (RF)) *A auction protocol ensures* Receipt-Freeness (RF) *if for any two auction processes* $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and* $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and any subset* $I \subseteq \{1,\ldots,k\}$*, there exist processes* B'_i *such that we have*

$$\forall i \in I : B_i'^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA^H|_{res} \approx_l APB^H|_{res} \Rightarrow APA_I\left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i}\right] \approx_l APB_I\left[\mid_{i\in I} B'_i\right].$$

Similarly to Privacy (P), we prove that for protocols implementing a First-Price Sealed-Bid Auction (First-Price Sealed-Bid Auction), Receipt-Free Strong Anonymity (RF-SA) and Receipt-Freeness coincide.

Theorem 61 *If an electronic auction protocol implements a First-Price Sealed-Bid Auction (FPSBA), then Receipt-Freeness (RF) and Receipt-Free Strong Anonymity (RF-SA) are equivalent.*

Proof Suppose an electronic auction protocol ensures (RF), i.e. if for any two auction processes $APA = \nu\tilde{n}'.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}'.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and any subset $I \subseteq \{1, \ldots, k\}$, there exist processes B'_i such that we have

$$\forall i \in I : B_i'^{\setminus out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA^H|_{res} \approx_l APB^H|_{res} \Rightarrow APA_I\left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i}\right] \approx_l APB_I\left[\mid_{i\in I} B'_i\right].$$

We want to show (RF-SA), i.e. for any two auction processes $APA = \nu\tilde{n}'.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}'.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ such that $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, and for any subset $I \subseteq \{1, \ldots, k\}$ there exist processes B'_i such that we have

$$\forall i \in I : B_i'^{\setminus out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA_I\left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i}\right] \approx_l APB_I\left[\mid_{i\in I} B'_i\right]$$

Assume such APA and APB, then $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, hence by (FPSBA) we obtain that $APA^H|_{res} \approx_l APB^H|_{res}$, which allows to conclude using (RF).

Suppose that we have (RF-SA) and we want to show (RF). By (FPSBA) if we have $APA^H|_{res} \approx_l APB^H|_{res}$, we have $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, hence we can conclude using (RF-SA). □

Note that the above definitions implicitly assumes that all bidders not under attack are honest. If one also wants to consider corrupted bidders, we can again replace some of the honest bidders $B\sigma_{id_i}\sigma_{b_{i,X}}$ by corrupted bidders $(B\sigma_{id_i})^{c_{1i},c_{2i}}$ as follows. Again, differences to the definition without corrupted bidders are marked in **bold**.

Definition 65 (RF-XXX-C) *An electronic auction protocol ensures* RF-XXX-C *if for any two auction processes* $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and* $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *such that*

— *if XXX=BPS* (Bidding-Price-Secrecy), *for* $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ *with* $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ *and for any subset* $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$ *there exist processes* B'_i *such that* ***for any subset*** $\boldsymbol{H \subseteq \{1,\ldots,k\}}$ ***with*** $\boldsymbol{H \cap J = \emptyset}$ ***and***

— *if XXX=U* (Unlinkability), *for* $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ *with* $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ *and a permutation* Π *with* $\forall i : b_{i,B} = b_{\Pi(i),A}$, *and for any subset* $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$ *there exist processes* B'_i *such that* ***for any subset*** $\boldsymbol{H \subseteq \{1,\ldots,k\}}$ ***with*** $\boldsymbol{H \cap J = \emptyset}$ ***and***

— *if XXX=SA* (Strong Anonymity), $\max_i b_{i,A} = \max_i b_{i,B}$ *and* $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$ *and for any subset* $I \subseteq \{1,\ldots,k\}$ *there exist processes* B'_i *such that* ***for any subset*** $\boldsymbol{H \subseteq \{1,\ldots,k\}}$

— *if XXX=WA* (Weak Anonymity), *there exists a permutation* Π *with* $\forall i : b_{i,B} = b_{\Pi(i),A}$, *and for any subset* $I \subseteq \{1,\ldots,k\}$ *there exist processes* B'_i *such that* ***for any subset*** $\boldsymbol{H \subseteq \{1,\ldots,k\}}$

with $\boldsymbol{\forall h \in H : \sigma_{b_{h,A}} = \sigma_{b_{h,B}}}$ ***and*** $\boldsymbol{H \cap I = \emptyset}$, *we have*

$$\forall i \in I : B_i'^{\setminus out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA_{I\cup H}\left[\,|_{i\in I}(B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i} \;|_{\boldsymbol{h\in H}}\,(\boldsymbol{B\sigma_{id_h}})^{\boldsymbol{c_{1h},c_{2h}}}\right] \approx_l APB_{I\cup H}\left[\,|_{i\in I}\,B'_i \;|_{\boldsymbol{h\in H}}\,(\boldsymbol{B\sigma_{id_h}})^{\boldsymbol{c_{1h},c_{2h}}}\right]$$

Similarly our generalized definition looks as follows.

Definition 66 (Receipt-Freeness with Corrupted Bidders (RF-C)) *An auction protocol ensures* Receipt-Freeness with Corrupted Bidders (RF-C) *if for any two auction processes* $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and* $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and any subset* $I \subseteq \{1,\ldots,k\}$ *there exist processes* B'_i *such that* ***for any subset*** $\boldsymbol{H \subseteq \{1,\ldots,n\}}$ ***with*** $\boldsymbol{\forall h \in H : \sigma_{b_{h,A}} = \sigma_{b_{h,B}}}$ ***and*** $\boldsymbol{H \cap I = \emptyset}$, *we have*

$$\forall i \in I : B_i'^{\setminus out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA^H|_{res} \approx_l APB^H|_{res}$$
$$\Downarrow$$
$$APA_{I\cup \boldsymbol{H}}\left[\,|_{i\in I}(B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i}\;|_{\boldsymbol{h\in H}}(\boldsymbol{B\sigma_{id_h}})^{\boldsymbol{c_{1h},c_{2h}}}\right]$$
$$\approx_l$$
$$APB_{I\cup \boldsymbol{H}}\left[\,|_{i\in I}B'_i\;|_{\boldsymbol{h\in H}}(\boldsymbol{B\sigma_{id_h}})^{\boldsymbol{c_{1h},c_{2h}}}\right].$$

Then again we have the following hierarchy between the different notions.

Theorem 62 *We have for any XXX* $\in \{$*BPS, U, SA, WA*$\}$*:*

— *Any auction protocol ensuring (RF-XXX-C) also ensures (RF-XXX).*

— *Any auction protocol ensuring (RF-C) also ensures (RF).*

— *Any auction protocol ensuring (RF-BPS-C) also ensures (RF-U-C).*

— *Any auction protocol ensuring (RF-SA-C) also ensures (RF-BPS-C).*

— *Any auction protocol ensuring (RF-SA-C) also ensures (RF-WA-C).*

— *Any auction protocol ensuring (RF-WA-C) also ensures (RF-U-C).*

— *If an auction protocol implements (FPSBA), then (RF-C) and (RF-SA-C) are equivalent.*

Proof The first two propositions follow directly for $H = \emptyset$. The remaining propositions can be shown analogously to Theorems 60 and 61. We simply add the set of corrupted bidders, the rest of the proofs remains unchanged. □

§ 4.3.4.3. Coercion-Resistance. Coercion-Resistance is a stronger property than receipt-freeness: The intruder may not only ask for a receipt, but is also allowed to interact with the bidder during the bidding process and to give orders. We can generalize the previously discussed Receipt-Freeness notions to Coercion-Resistance as follows.

Definition 67 (CR-XXX) *An electronic auction protocol ensures* CR-XXX *if for any two auction processes* $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and* $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *such that*

— *if XXX=BPS* (Bidding-Price Secrecy)*, for* $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ *with* $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ *and for any subset* $I \subseteq \{1,\ldots,k\}$*,* $I \cap J = \emptyset$*,*

— *if XXX=U* (Unlinkability)*, for* $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ *with* $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ *and a permutation* Π *with* $\forall i : b_{i,B} = b_{\Pi(i),A}$*, and for any subset* $I \subseteq \{1,\ldots,k\}$*,* $I \cap J = \emptyset$*,*

— *if XXX=SA* (Strong Anonymity), $\max_i b_{i,A} = \max_i b_{i,B}$ *and* $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, *and for any subset* $I \subseteq \{1,\ldots,k\}$,

— *if XXX=WA* (Weak Anonymity), *there exists a permutation* Π *with* $\forall i : b_{i,B} = b_{\Pi(i),A}$, *and for any subset* $I \subseteq \{1,\ldots,k\}$,

there exist processes B'_i *such that for any contexts* $C_i, i \in I$ *with* $C_i = \nu c_1.\nu c_2.(\_|P_i)$, $\tilde{n} \cap fn(C) = \emptyset$ *and*

$$APA_I\left[\,|_{i\in I} C_i\left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2}\right]\right] \approx_l APA_I\left[\,|_{i\in I}(B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i}\right]$$

we have

$$\forall i \in I : C_i\left[B'_i\right]^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{v_{i,B}}$$

and

$$APA_I\left[\,|_{i\in I} C_i\left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2}\right]\right] \approx_l APB_I\left[\,|_{i\in I} C_i\left[B'_i\right]\right]$$

The difference to the previous receipt-freeness definitions is that the attacked bidders do not only reveal their data on channel c_1, but also take orders on channel c_2. The context C_i models the attacker that tries to force them to bid the price $b_{i,A}$ (this is expressed by the condition on C_i). The protocol is hence coercion-resistant if there exists a counter-strategy B' which allow the bidders to bid $b_{i,B}$ instead without the attacker noticing.

Note that the hierarchy we already observed for Privacy and Receipt-Freeness also holds on the level of Coercion-Resistance.

Theorem 63 *We have:*

— *A protocol ensuring (CR-SA) also ensures (CR-BPS) and (CR-WA).*

— *A protocol ensuring (CR-BPS) or (CR-WA) also ensures (CR-U).*

Proof Similar to Theorem 60: We only need to consider the different parameters, which are specializations of each other. □

For non sealed-bid first-price auctions, we obtain the following definition.

Definition 68 (Coercion-Resistance (CR)) *An electronic auction protocol ensures* Coercion-Resistance (CR) *if for any two auction processes* $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and* $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and any subset* $I \subseteq \{1,\ldots,k\}$, *there exists processes* B'_i *such that for any contexts* $C_i, i \in I$ *with* $C_i = \nu c_1.\nu c_2.(\_|P_i)$, $\tilde{n} \cap fn(C) = \emptyset$ *and*

$$APA_I\left[\,|_{i\in I} C_i\left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2}\right]\right] \approx_l APA_I\left[\,|_{i\in I}(B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i}\right]$$

we have

$$\forall i \in I : C_i \left[B'_i\right]^{\setminus out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{v_{i,B}}$$

and

$$APA^H|_{res} \approx_l APB^H|_{res}$$
$$\Downarrow$$
$$APA_I \left[\mid_{i\in I} C_i \left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2}\right]\right] \approx_l APB_I \left[\mid_{i\in I} C_i \left[B'_i\right]\right]$$

Again the contexts C_i model the part of the adversary interacting with the coerced bidders. The contexts are limited to force the bidders to bid their bids from the first instance to ensure that both instances with coercion result in the same election outcome, and are not trivially distinguishable. Similarly to Receipt-Freeness we can prove that for protocols implementing a First-Price Sealed-Bid Auction (FPSBA), Coercion-Resistant Strong Anonymity (CR-SA) and Coercion-Resistance (CR) coincide.

Theorem 64 *If an electronic auction protocol implements a First-Price Sealed-Bid Auction (FPSBA), then Coercion-Resistant Strong Anonymity (CR-SA) and Coercion-Resistance (CR) are equivalent.*

Proof Suppose an electronic auction protocol ensures (CR), i.e. if for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and any subset $I \subseteq \{1,\ldots,k\}$, there exists processes B'_i such that for any contexts $C_i, i \in I$ with $C_i = \nu c_1.\nu c_2.(\_|P_i)$, $\tilde{n} \cap fn(C) = \emptyset$ and

$$APA_I \left[\mid_{i\in I} C_i \left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2}\right]\right] \approx_l APA_I \left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i}\right]$$

we have

$$\forall i \in I : C_i \left[B'_i\right]^{\setminus out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{v_{i,B}}$$

and

$$APA^H|_{res} \approx_l APB^H|_{res}$$
$$\Downarrow$$
$$APA_I \left[\mid_{i\in I} C_i \left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2}\right]\right] \approx_l APB_I \left[\mid_{i\in I} C_i \left[B'_i\right]\right]$$

We want to show (CR-SA), i.e. that for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ such that $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, and for any subset $I \subseteq \{1,\ldots,k\}$ there exist processes B'_i such that for any contexts $C_i, i \in I$ with $C_i = \nu c_1.\nu c_2.(\_|P_i)$,

$\tilde{n} \cap fn(C) = \emptyset$ and

$$APA_I \left[\mid_{i \in I} C_i \left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2} \right] \right] \approx_l APA_I \left[\mid_{i \in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i} \right]$$

we have $\forall i \in I : C_i \left[B'_i \right]^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{v_{i,B}}$ and

$$APA_I \left[\mid_{i \in I} C_i \left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2} \right] \right] \approx_l APB_I \left[\mid_{i \in I} C_i \left[B'_i \right] \right]$$

Assume such APA and APB, then $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, hence by (FPSBA) we obtain that $APA^H|_{res} \approx_l APB^H|_{res}$, which allows to conclude using (CR).

Suppose that we have (CR-SA) and we want to show (CR). By (FPSBA) if we have $APA^H|_{res} \approx_l APB^H|_{res}$, we have $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, hence we can conclude using (CR-SA). □

We also note that similarly to the case of electronic voting, Coercion-Resistance is stronger than Receipt-Freeness.

Theorem 65 *An electronic auction protocol ensuring Coercion-Resistance (CR) also ensures Receipt-Freeness (RF).*

Proof Assume that for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and any subset $I \subseteq \{1, \ldots, k\}$, there exists processes B'_i such that for any contexts $C_i, i \in I$ with $C_i = \nu c_1.\nu c_2.(\_|P_i)$, $\tilde{n} \cap fn(C) = \emptyset$ and

$$APA_I \left[\mid_{i \in I} C_i \left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2} \right] \right] \approx_l APA_I \left[\mid_{i \in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i} \right]$$

we have

$$\forall i \in I : C_i \left[B'_i \right]^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{v_{i,B}}$$

and

$$APA^H|_{res} \approx_l APB^H|_{res}$$
$$\Downarrow$$
$$APA_I \left[\mid_{i \in I} C_i \left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2} \right] \right] \approx_l APB_I \left[\mid_{i \in I} C_i \left[B'_i \right] \right]$$

We have to show that for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and any subset $I \subseteq \{1, \ldots, k\}$, there exist processes B'_i such that we have

$$\forall i \in I : B'^{\backslash out(chc_i,\cdot)}_i \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA^H|_{res} \approx_l APB^H|_{res} \Rightarrow APA_I \left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i} \right] \approx_l APB_I \left[\mid_{i\in I} B'_i \right].$$

Let $C_i, i \in I$ be evaluation contexts with $C_i = \nu c_1.\nu c_2.(\_|P_i)$, $\tilde{n} \cap fn(C) = \emptyset$ and

$$APA_I \left[\mid_{i\in I} C_i \left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2} \right] \right] \approx_l APA_I \left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i} \right]$$

Note that such a C can be constructed directly from the bidders process B. By hypothesis we know that there are closed plain processes B'_i such that

$$\forall i \in I : C_i \left[B'_i \right]^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{v_{i,B}}$$

and

$$APA^H|_{res} \approx_l APB^H|_{res}$$
$$\Downarrow$$
$$APA_I \left[\mid_{i\in I} C_i \left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2} \right] \right] \approx_l APB_I \left[\mid_{i\in I} C_i \left[B'_i \right] \right]$$

We have to find other processes B''_i such that

$$\forall i \in I : B''^{\backslash out(chc_i,\cdot)}_i \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA^H|_{res} \approx_l APB^H|_{res} \Rightarrow APA_I \left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i} \right] \approx_l APB_I \left[\mid_{i\in I} B''_i \right]$$

Let $\forall i\colon B''_i = C_i[B'_i]$. This directly fulfills the first requirement. Assume $APA^H|_{res} \approx_l APB^H|_{res}$. We now use the hypotheses

$$APA_I \left[\mid_{i\in I} C_i \left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2} \right] \right] \approx_l APA_I \left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i} \right]$$

and

$$APA_I \left[\mid_{i\in I} C_i \left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2} \right] \right] \approx_l APB_I \left[\mid_{i\in I} C_i \left[B'_i \right] \right]$$

As labeled bisimilarity is transitive, we can conclude

$$APA_I \left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i} \right] \approx_l APB_I \left[\mid_{i\in I} C_i \left[B'_i \right] \right]$$

which gives us the desired result for $B''_i = C_i[B'_i]$. □

This also holds for (CR-XXX).

Theorem 66 *An electronic auction protocol ensuring (CR-XXX) also ensures (RF-XXX) for any XXX* $\in \{$*BPS, U, SA, WA*$\}$.

Proof Assume that for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ such that

— if XXX=BPS *(Bidding-Price Secrecy)*, for $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ with $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ and for any subset $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$,

— if XXX=U *(Unlinkability)*, for $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ with $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ and a permutation Π with $\forall i : b_{i,B} = b_{\Pi(i),A}$, and for any subset $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$,

— if XXX=SA *(Strong Anonymity)*, $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, and for any subset $I \subseteq \{1,\ldots,k\}$,

— if XXX=WA *(Weak Anonymity)*, there exists a permutation Π with $\forall i : b_{i,B} = b_{\Pi(i),A}$, and for any subset $I \subseteq \{1,\ldots,k\}$,

there exist processes B'_i such that for any contexts $C_i, i \in I$ with $C_i = \nu c_1.\nu c_2.(\_|P_i)$, $\tilde{n} \cap fn(C) = \emptyset$ and

$$APA_I\left[\mid_{i\in I} C_i\left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2}\right]\right] \approx_l APA_I\left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i}\right]$$

we have

$$\forall i \in I : C_i\left[B'_i\right]^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{v_{i,B}}$$

and

$$APA_I\left[\mid_{i\in I} C_i\left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2}\right]\right] \approx_l APB_I\left[\mid_{i\in I} C_i\left[B'_i\right]\right]$$

We have to show that for any two auction processes $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ and $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ such that

— if XXX=BPS *(Bidding-Price-Secrecy)*, for $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ with $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ and for any subset $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$,

— if XXX=U *(Unlinkability)*, for $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ with $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ and a permutation Π with $\forall i : b_{i,B} = b_{\Pi(i),A}$, and for any subset $I \subseteq \{1,\ldots,k\}$, $I \cap J = \emptyset$,

— if XXX=SA *(Strong Anonymity)*, $\max_i b_{i,A} = \max_i b_{i,B}$ and $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$, and for any subset $I \subseteq \{1,\ldots,k\}$,

— if XXX=WA *(Weak Anonymity)*, there exists a permutation Π with $\forall i : b_{i,B} =$

$b_{\Pi(i),A}$, and for any subset $I \subseteq \{1, \ldots, k\}$,

there exist processes B'_i such that we have

$$\forall i \in I : B_i'^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA_I \left[\mid_{i \in I} (B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i} \right] \approx_l APB_I \left[\mid_{i \in I} B'_i \right]$$

Since the conditions on the $\sigma_{b_{i,X}}$ and I depending on the parameter XXX are the same in both cases (CR and RF), we can proceed independently of XXX. Let $C_i, i \in I$ be evaluation contexts with $C_i = \nu c_1.\nu c_2.(\_|P_i)$, $\tilde{n} \cap fn(C) = \emptyset$ and

$$APA_I \left[\mid_{i \in I} C_i \left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2} \right] \right] \approx_l APA_I \left[\mid_{i \in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i} \right]$$

Note that such a C can be constructed directly from the bidders process B. By hypothesis we know that there are closed plain processes B'_i such that

$$\forall i \in I : C_i \left[B'_i \right]^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{v_{i,B}}$$

and

$$APA_I \left[\mid_{i \in I} C_i \left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2} \right] \right] \approx_l APB_I \left[\mid_{i \in I} C_i \left[B'_i \right] \right]$$

We have to find other processes B''_i such that

$$\forall i \in I : B_i''^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{b_{i,B}}$$

and

$$APA_I \left[\mid_{i \in I} (B\sigma_{id_i}\sigma_{v_{i,A}})^{chc_i} \right] \approx_l APB_I \left[\mid_{i \in I} B''_i \right]$$

Let $\forall i : B''_i = C_i[B'_i]$. This directly fulfills the first requirement. We now use the hypotheses

$$APA_I \left[\mid_{i \in I} C_i \left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2} \right] \right] \approx_l APA_I \left[\mid_{i \in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i} \right]$$

and

$$APA_I \left[\mid_{i \in I} C_i \left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2} \right] \right] \approx_l APB_I \left[\mid_{i \in I} C_i \left[B'_i \right] \right]$$

As labeled bisimilarity is transitive, we can conclude

$$APA_I \left[\mid_{i \in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i} \right] \approx_l APB_I \left[\mid_{i \in I} C_i \left[B'_i \right] \right]$$

which gives us the desired result for $B_i'' = C_i[B_i']$. □

We can also include corrupted bidders as follows. Again, differences to the definition without corrupted bidders are marked in **bold**.

Definition 69 (CR-XXX-C) *An electronic auction protocol ensures* CR-XXX-C *if for any two auction processes* $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and* $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *such that*

- *if XXX=BPS* (Bidding-Price Secrecy)*, for* $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ *with* $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ *and for any subset* $I \subseteq \{1,\ldots,k\}$ *with* $I \cap J = \emptyset$ *there exist processes* B_i' *such that* ***for any subset*** $\boldsymbol{H \subseteq \{1,\ldots,k\}}$ ***with*** $\boldsymbol{H \cap J = \emptyset}$ ***and***
- *if XXX=U* (Unlinkability)*, for* $J = \arg\max_i b_{i,A} = \arg\max_i b_{i,B}$ *with* $\forall j \in J\colon b_{j,A} = b_{j,B} = \max_i b_{i,A} = \max_i b_{i,B}$ *and a permutation* Π *with* $\forall i : b_{i,B} = b_{\Pi(i),A}$*, and for any subset* $I \subseteq \{1,\ldots,k\}$ *with* $I \cap J = \emptyset$ *there exist processes* B_i' *such that* ***for any subset*** $\boldsymbol{H \subseteq \{1,\ldots,k\}}$ ***with*** $\boldsymbol{H \cap J = \emptyset}$ ***and***
- *if XXX=SA* (Strong Anonymity)*,* $\max_i b_{i,A} = \max_i b_{i,B}$ *and* $|\arg\max_i b_{i,A}| = |\arg\max_i b_{i,B}|$ *and for any subset* $I \subseteq \{1,\ldots,k\}$ *there exist processes* B_i' *such that* ***for any subset*** $\boldsymbol{H \subseteq \{1,\ldots,k\}}$
- *if XXX=WA* (Weak Anonymity)*, there exists a permutation* Π *with* $\forall i : b_{i,B} = b_{\Pi(i),A}$*, and for any subset* $I \subseteq \{1,\ldots,k\}$ *there exist processes* B_i' *such that* ***for any subset*** $\boldsymbol{H \subseteq \{1,\ldots,k\}}$

with $\boldsymbol{\forall h \in H : \sigma_{b_{h,A}} = \sigma_{b_{h,B}}}$ ***and*** $\boldsymbol{H \cap I = \emptyset}$ *we have that for any contexts* $C_i, i \in I$ *with* $C_i = \nu c_1.\nu c_2.(\_|P_i)$*,* $\tilde{n} \cap fn(C) = \emptyset$ *and*

$$APA_{I\cup\boldsymbol{H}}\left[\mid_{i\in I} C_i\left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2}\right] \mid_{\boldsymbol{h\in H}} (\boldsymbol{B\sigma_{id_h}})^{\boldsymbol{c_{1h},c_{2h}}}\right] \approx_l APA_{I\cup\boldsymbol{H}}\left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i} \mid_{\boldsymbol{h\in H}} (\boldsymbol{B\sigma_{id_h}})^{\boldsymbol{c_{1h},c_{2h}}}\right]$$

we have

$$\forall i \in I : C_i\left[B_i'\right]^{\setminus out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{v_{i,B}}$$

and

$$APA_{I\cup\boldsymbol{H}}\left[\mid_{i\in I} C_i\left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2}\right] \mid_{\boldsymbol{h\in H}} (\boldsymbol{B\sigma_{id_h}})^{\boldsymbol{c_{1h},c_{2h}}}\right] \approx_l APB_{I\cup\boldsymbol{H}}\left[\mid_{i\in I} C_i\left[B_i'\right] \mid_{\boldsymbol{h\in H}} (\boldsymbol{B\sigma_{id_h}})^{\boldsymbol{c_{1h},c_{2h}}}\right]$$

For non sealed-bid first-price auctions, we obtain the following definition.

Definition 70 (Coercion-Resistance with Corrupted Bidders (CR-C)) *An auction protocol ensures* Coercion-Resistance with Corrupted Bidders (CR-C) *if for any two auction processes* $APA = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,A}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,A}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and* $APB = \nu\tilde{n}.(B\sigma_{id_1}\sigma_{b_{1,B}} \mid \ldots \mid B\sigma_{id_k}\sigma_{b_{k,B}} \mid S \mid A_1 \mid \ldots \mid A_l)$ *and any subset* $I \subseteq \{1,\ldots,k\}$*, there exists processes* B'_i *such that* $\boldsymbol{for\ any\ subset\ H \subseteq \{1,\ldots,n\}\ with\ \forall h \in H : \sigma_{b_{h,A}} = \sigma_{b_{h,B}}\ and\ H \cap I = \emptyset}$ *we have that for any contexts* $C_i, i \in I$ *with* $C_i = \nu c_1.\nu c_2.(\_|P_i)$*,* $\tilde{n} \cap fn(C) = \emptyset$ *and*

$$APA_{I\cup H}\left[\mid_{i\in I} C_i\left[(B\sigma_{id_i}\sigma_{b_{i,A}})^{c_1,c_2}\right] \mid_{h\in H} (B\sigma_{id_h})^{c_{1h},c_{2h}}\right] \approx_l APA_{I\cup H}\left[\mid_{i\in I} (B\sigma_{id_i}\sigma_{b_{i,A}})^{chc_i} \mid_{h\in H} (B\sigma_{id_h})^{c_{1h},c_{2h}}\right]$$

we have

$$\forall i \in I : C_i\left[B'_i\right]^{\backslash out(chc_i,\cdot)} \approx_l B\sigma_{id_i}\sigma_{v_{i,B}}$$

and

$$APA^H|_{res} \approx_l APB^H|_{res}$$

$$\Downarrow$$

$$APA_{I\cup H}\left[\mid_{i\in I} C_i\left[(B\sigma_{id_i}\sigma_{v_{i,A}})^{c_1,c_2}\right] \mid_{h\in H} (B\sigma_{id_h})^{c_{1h},c_{2h}}\right]$$

$$\approx_l$$

$$APB_{I\cup H}\left[\mid_{i\in I} C_i\left[B'_i\right] \mid_{h\in H} (B\sigma_{id_h})^{c_{1h},c_{2h}}\right]$$

Then again we have the following hierarchy between the different notions.

Theorem 67 *We have for any* $XXX \in \{BPS,\ U,\ SA,\ WA\}$*:*

- *Any auction protocol ensuring (CR-XXX-C) also ensures (CR-XXX).*
- *Any auction protocol ensuring (CR-C) also ensures (CR).*
- *Any auction protocol ensuring (CR-BPS-C) also ensures (CR-U-C).*
- *Any auction protocol ensuring (CR-SA-C) also ensures (CR-BPS-C).*
- *Any auction protocol ensuring (CR-SA-C) also ensures (CR-WA-C).*
- *Any auction protocol ensuring (CR-WA-C) also ensures (CR-U-C).*
- *If an auction protocol implements (FPSBA), then (CR-C) and (CR-SA-C) are equivalent.*

Proof The first two propositions follow directly for $H = \emptyset$. The remaining propositions can be shown analogously to Theorems 63 on page 186 and 64 on page 187. We can add the set of corrupted bidders without changing the rest of the proofs. □

Figure 4.3 on page 170 sums up the relations between all the above notions.

4.3.5 — Case Studies

We applied the previously explained definitions on three case studies using ProVerif: the protocol by Curtis et al. [CPS07], the protocol by Brandt [Bra06] and the protocol by Sako [Sak00].

§ 4.3.5.1. Protocol by Curtis, Pierprzyk and Seruga. The protocol by Curtis et al. [CPS07] was designed to support sealed-bid first- and second price auctions while guaranteeing fairness, privacy, verifiability and non-repudiation.

Description. The main idea of the protocol is the following: The bidders register with a trusted Registration Authority (RA) using a Public-Key Infrastructure (PKI), which issues pseudonyms that will then be used to submit bids to the Seller (S). It is split into three phases: Registration, Bidding, and Winner determination.

1. *Registration:* Each bidder sends his identity, a hash $h(b_i)$ of his bidding price b_i and a signature of $h(b_i)$ to the RA. The RA checks the identity and the signature using the PKI, and replies with an encrypted (using the bidder's public key) and signed message containing a newly generated pseudonym p and the hashed bid $h(b_i)$.
2. *Bidding:* The RA generates a new symmetric key k. Each bidder will send $c = Enc_{pk_S}(b_i)$, his bid b_i encrypted with the seller's public key, and a signature of c, together with his pseudonym to the RA. The RA will reply with a signature on c, and encrypts the bidders message, together with the hashed bid $h(b_i)$ from phase one, using the symmetric key k. This encrypted message is then sent to the seller.
3. *Winner determination:* After all bids have been submitted, the RA will reveal the symmetric key k to the seller. The seller can then decrypt the bids, verify the correctness of the hash and determine the winner. To identify the winner using the pseudonym he can ask the RA to reveal the true identity.

Remark Note that the protocol does not specify how exactly the winner is determined. The seller obtains all bids in clear, and can thus apply any auction function. This may be a simple first-price auction (i.e. the highest bid wins), but also second-price auctions or other auction functions are supported. However, as some of our properties rely on the event `won`, we need to specify some auction function to be able to perform the verification. Hence we assume – for simplicity – a first-price auction in the following.

Formal Model. We modeled the protocol in ProVerif using an equational theory inspired by the equations given in the original paper [CPS07], It contains equations

```
let bidder(b:int,k:skey,rk:pkey,ak:pkey,chR:channel) =
  (* first message to registrar: signed hash of bid *)
  let ha = h(intToBit(b)) in
  out(chR,(ha,sign(ha,k)));
  (* answer containing pseudonym P *)
  in(chR,m:bitstring);
  let (P:bitstring,hc:bitstring)
    = checksign(dec(m,k),rk) in
  if ha = hc then
   (* bidding *)
   let c = enc(intToBit(b),ak) in
   event bid(b,P);
   out(chR,(P,c,sign(c,k))).
```

Listing 4.1 – The bidder.

for symmetric encryption (functions `senc` and `sdec`), asymmetric encryption[5] (functions `enc`, `dec` and `pk` – which generates the public key corresponding to a secret key) and signatures (functions `sign`, `checksign` and `getmessage`):

$$\begin{aligned} \texttt{sdec}(\texttt{senc}(m, key), key) &= m \\ \texttt{dec}(\texttt{enc}(m, \texttt{pk}(sk)), sk) &= m \\ \texttt{checksign}(\texttt{sign}(m, sk), \texttt{pk}(sk)) &= m \\ \texttt{getmessage}(\texttt{sign}(m, sk)) &= m \end{aligned}$$

Listing 4.1 shows our model of the bidder, Listing 4.2 on the following page the code of the registrar. Here we employ ProVerif syntax, i.e. including types for the variables and parameters for the subprocesses, and type conversion functions (e.g. `intToBit` converting an integer to a bitstring). For a detailed explanation, see the ProVerif manual [BSC13]. Listing 4.3 on page 197 gives the code for the auctioneer, and Listing 4.4 on page 197 shows the code for an instance with two bidders. The ProVerif code used for each of the verifications below is available online [Dre13].

Analysis. We assume an honest *RA* and an honest seller.

Non-Repudiation (NR): To prove (NR), we have to show that on each possible trace the event `won(p,id)` is preceded by the event `bid(p,id)`. ProVerif can verify such properties using queries, in this case using the query

```
query p:price,id:identity;
event(won(p,id)) ==> event(bid(p,id)).
```

[5]Note that in this model the asymmetric encryption is not probabilistic (following the original equation by Curtis et al.), however the use of a probabilistic encryption scheme does not change our results.

```
let subregistrar(k:skey,bk:pkey,ak:pkey,syk:sykey,
 chS:channel,chA:channel,chR:channel) =
  (* signed hash of bid from bidder *)
  in(chR,(ha:bitstring,sa:bitstring));
  if ha = checksign(sa,bk) then
   (* create pseudonym *)
   new p:bitstring;
   out(chR,enc(sign((p,ha),k),bk));
   (* receive bid *)
   in(chR,(p1:bitstring,c:bitstring,sc:bitstring));
   if p = p1 && checksign(sc,bk) = c then
    event recBid(ha,p);
    (* confirmation to bidder *)
    out(chR,(c,sign(c,k)));
    (* encrypted bid to auctioneer *)
    out(chA,senc((p,c,ha),syk));
    (* synchronization *)
    out(chS,ack).

let registrar(k:skey,bk1:pkey,...,bkn:pkey,ak:pkey,
 chA:channel,chR:channel) =
  (* create symmetric key *)
  new syk:sykey;
  (* channel for synchronization *)
  new chS:channel;
  (* subprocesses *)
  subregistrar(k,bk1,ak,syk,chS,chA,chR) |
  ...
  subregistrar(k,bkn,ak,syk,chS,chA,chR) |
  (* reveal symmetric key after synchronization *)
  in(chS,x1:bitstring); ...; in(chS,xn:bitstring);
  out(chA,syk).
```

Listing 4.2 – The registrar.

```
let auctioneer(k:skey,pkR:pkey,chA:channel)=
 (* receive bids and sym. key *)
 in(chA,x1:bitstring);
 ...
 in(chA,xn:bitstring);
 in(chA,sk:sykey);
 (* decryption *)
 let (p1:bitstring,c1:bitstring,ha1:bitstring)
   = sdec(x1,sk) in
 let bid1:bitstring=dec(c1,k) in
 ...
 let (pn:bitstring,cn:bitstring,han:bitstring)
   = sdec(xn,sk) in
 let bidn:bitstring = dec(cn,k) in
 (* determine winner *)
 if h(bid1) = ha1 && ... && h(bidn) = han then
  if geq(bitToInt(bid1),bitToInt(bid2)) && ... &&
   geq(bitToInt(bid1),bitToInt(bidn)) then
    event won(bitToInt(bid1),p1);
    out(chW,p1)
  else
    if geq(bitToInt(bid2),bitToInt(bid1)) && ... &&
     geq(bitToInt(bid2),bitToInt(bidn)) then
      event won(bitToInt(bid2),p1);
      out(chW,p2)
    else
      ...
```

Listing 4.3 – The auctioneer.

```
process
 (* create keys *)
 new skb1:skey; new skb2:skey; new ska:skey; new skr:skey;
 (* public keys are published *)
 out(chPKI,pk(skb1)) |
 out(chPKI,pk(skb2)) |
 out(chPKI,pk(ska)) |
 out(chPKI,pk(skr)) |
 (* two bidders, the registrar and the auctioneer *)
 bidder(b1,skb1,pk(skr),pk(ska),chR) |
 bidder(b2,skb2,pk(skr),pk(ska),chR) |
 registrar(skr,pk(skb1),pk(skb2),pk(ska),chA,chR) |
 auctioneer(ska,pk(skr),chA)
```

Listing 4.4 – The main process for an instance with two bidders.

This query means that for any value `p` of type `price` and any `id` of type `identity`, if the event `won(p,id)` is recorded, it is preceded by the event `bid(p,id)`. In this case we use the pseudonyms (of type `bitstring`) to identify the bidders, hence the query looks like this:

```
query p:price,pseudo:bitstring;
event(won(p,pseudo)) ==> event(bid(p,pseudo)).
```

ProVerif finds the following attack: Since the channel between the Registration Authority and the Seller is not protected, anybody can pretend to be the RA and submit false bids, encrypted with a self-chosen symmetric key. After all false bids are submitted, the attacker reveals the symmetric key and the seller will decrypt the bogus bids. Hence the event `won(p,id)` can be emitted on a trace without any event `bid(p,id)`. We propose a solution to address this problem: If the messages from the RA to the seller are signed, the attacker cannot impersonate the RA any more and ProVerif is able to prove Non-Repudiation for the accordingly modified protocol.

Non-Cancellation (NC): Here we have to show that even if a bidder reveals his secret data to the intruder, the intruder cannot cancel a submitted bid, i.e. there is no trace with the events `recBid(p_1,id_a)` and `won(p_2,id_b)` where `p_1 > p_2`. To verify this we need to model at least two distinct prices, which can be implemented using constants, i.e. by setting `p_1 = max_price` and `p_2 = smaller_price`, where `max_price` and `smaller_price` are two constants such that `max_price > smaller_price`[6]. Then we want to test the conjunction (not the precedence as above) of two events, which is not possible directly in ProVerif. A well-known solution is to replace the underlying events with messages over a private channel to a newly added processes which will call a conjunction event `recBid_and_won` once he received all the messages. Then we could use the following query:

```
query event(recBid_and_won(p_1, id_a, p_2, id_b)).
```

where the first two parameters are from the event `recBid(p_1,id_a)` and the second from the event `won(p_2,id_b)`, here instantiated with price constants as explained above and two constants for two different bidders. However, as above, we use the pseudonyms in our events `recBid` and `won`, as for example to seller (who executes the event `won`) does not know the real identity of a bidder. Hence we cannot have `won(p_2,id_b)`, but only `won(p_2,pseudonym)`. To associate the pseudonyms with bidders and still be able to execute the above query, we added

[6]Note that most auction protocols assume a finite number of possible prices anyway, which we can model using a list of constants.

an event `getAlias(id,p)` (and an output on a private channel to the process checking for the events) executed by the registrar when he attributes a pseudonym `p` to the identity `id`. This allows us to link pseudonyms to identities, hence the process checking for the conjunction can execute the event with the real identities as described above by linking them to the pseudonyms using his input from the RA.

For the original protocol, ProVerif finds a similar attack to the one described above: An attacker can delete the messages sent by the the RA to the seller, and choose a symmetric key and send bogus messages containing prices of his choice instead. When he reveals the symmetric key, a bidder of his choice will win, hence there will be an event `won(smaller_price,id_b)` for a smaller price than the one recorded by `recBid(max_price,id_a)`. Even if we add signatures as proposed above, ProVerif still comes up with an attack: A dishonest bidder might submit a first bid triggering the event `recBid` for this bid, delete the forwarded message to the seller, and then submit a second bid at a different price[7]. A first attempt to fix this issue would be – as proposed in the original paper – by including the number of bids in the message where the RA reveals the symmetric key. This would allow the seller to verify if he received the correct number of bids. However the attack still works if two auctions take place in parallel[8]: Since the RA uses the same PKI in both cases, he will use the same keys. The malicious bidder could register in the second auction, obtain the signed bid and replace his original bid with this message. The new message will include a different pseudonym, but the seller has no means of verifying if a pseudonym corresponds to the current auction. A solution would be to use different keys for different auctions (which need to be set up in a secure way), but we were unable to verify the resulting protocol because ProVerif cannot maintain state information, which is necessary to model the counting of messages.

Noninterference: It is clear that the protocol does not ensure Strong Noninterference (SN) since an attacker can simply count the number of messages to determine the number of participants. However we can check Weak Noninterference (WN), i.e. that any two instances containing the same bidders and only differing in the bids are bisimilar up to the end of the bidding phase, using the following query in ProVerif:

```
noninterf b_1, ..., b_n.
```

This query will ask ProVerif to verify strong secrecy of the variables `b_1, ...,`

[7]Note that this only works if he is allowed to submit several bids in the same auction, which is however not forbidden in the original protocol description.

[8]Note that this attack also works is he is only allowed to submit one single bid per auction, since he is submitting to different auctions.

`b_n.`, i.e. to check that any two instances of the protocol that only differ in these variables are bisimilar. For the original protocol ProVerif finds an attack which is based on the first message, which includes the hashed bidding price. An attacker simply hashes the possible values and compares the result. If we encrypt this message using the RA's public key, ProVerif is able to prove Weak Noninterference (WN). This modification was proposed in the original paper to achieve anonymity of bidders, but turns out to be also necessary to ensure fairness.

Highest Price Wins (HPW): Here we have to show that a malicious bidder cannot win the auction at a chosen price, even if another bidder submitted a higher bid. Again, we will assume that we have a finite number of possible prices. Then we can check the property using ProVerif by modeling two bidders, the first one bidding `max_price`, and the second one is corrupted by the adversary (according to Definition 19 on page 59). To prevent the adversary from just winning using the highest possible price (which would not necessarily correspond to an attack), we declare the constant `max_price` private[9]. We also have to be sure that the protocol does not leak `max_price` before the end of the bidding phase (which would contradict the intention of declaring it private). As we already showed Weak Noninterference (WN), we can be sure that this is not the case. Hence we could check if the event `won` is reachable for the corrupted bidder `id_B` using the following query

```
query bid:price; event(won(bid, id_B)).
```

Since bidder `id_A` submitted the highest possible price and the attacker cannot access and submit this value, he should be unable to make `id_B` win the auction. However, as the seller only knows the pseudonym, we again have to modify the query slightly. Our approach is the same as above: we use the event `getAlias`. Then we can execute the following query:

```
query bid:price, alias:bitstring; event(won(bid, alias))
            ==> event(getAlias(id_A, alias)).
```

The idea is simple: we ask ProVerif to prove that if a bidder wins, it was bidder `id_A`. This implies that the corrupted bidder `id_B` cannot win, as in our first query.

For the original protocol – only corrected with added encryption of the first bid to ensure Weak Noninterference –, ProVerif finds an attack again using the fact that the messages from the RA to the Seller are not authenticated, hence an

[9] In the definition we did not require A to submit the highest possible bid, but only a higher bid than anybody else. We could model the existence of higher prices by defining additional private constants, but this would not change the verification task since they are never used by any honest participants and are not accessible to the attacker.

attacker can pretend to be RA and submit bids of his choice to win the auction at a price of his choice. If we add signatures again, ProVerif still comes up with an attack: A dishonest bidder might register twice and then replace the message from the RA to the seller containing the correct bid with his own, bogus bid obtained using the second registration. As above, this could probably be circumvented by counting the messages and using different keys for different auctions, but for the same reason as above we cannot verify this in ProVerif.

Privacy: The authors claim in the original paper that if the first message is encrypted, their protocol ensures anonymity of the bidders. Yet we can see that it does not ensure Strong Anonymity (SA) since after the symmetric key has been published, an attacker can obtain a list with hashes of all bids, which allows to distinguish $h(b_A)$, $h(b_C)$ from $h(b_B)$, $h(b_C)$. Hence we checked Weak Anonymity (WA) using the `choice[]` operator in ProVerif, which verifies if the processes obtained by instantiating a variable with two different values are bisimilar. More precisely, we can check if for two swapping bidders (the first bidder bids `b_A = choice[b_1,b_2]`, the second `b_B = choice[b_2,b_1]`) the resulting processes are bisimilar. This query leads to another possible attack: The intruder might delay the messages from the second bidder. He waits until the first bidder sent his final message and this was relayed to the seller by the RA. This allows the attacker to link this message to the first bidder and distinguish both cases based on the hash after decrypting the message using the published symmetric key. This type of attack is well-known in electronic voting [DKR09]. As a solution, we have to ensure that both messages to the seller are sent at exactly the same time using synchronization. Inspired by some techniques used in ProSwapper [KSR10], we prove that the accordingly modified protocol ensures Weak Anonymity (WA) for two honest bidders. Note that technically our definition requires to consider any number of bidders, however due to the protocol structure ProVerif cannot do this automatically: the computation of the winner in the general case includes loops, which cannot be modeled in ProVerif. A manual proof following the idea of *modularity* used for voting protocols to reduce the general case to the case of two voters is also difficult, as the protocol structure violates such a modularity condition. One of the problems is the symmetric key used between the RA and the seller – decomposing instances will lead to several distinct keys (and hence also several messages where the key is revealed).

If we consider corrupted bidders, ProVerif discovers another weakness in the protocol: the corrupted bidder can essentially test if an honest bidder submitted a certain price. This works as follows. The corrupted bidder submits the hash of the price he wants to test to the RA as if it was his bid. The RA responds with an encrypted message containing the pseudonym and the hash of the bid, signed by the RA. The malicious bidder can decrypt this message, and then encrypt

the signed values using the public key of the targeted bidder, and send him this message. The targeted bidder decrypts the message, verifies the signature and checks if the hash corresponds to his bid. If yes, he continues, if no, he aborts, which is visible to the attacker by observing the next message. A solution to this problem is to send this message from the RA to the bidder over an private channel. This was proposed in the original protocol description, however it is unclear why the message needs to be encrypted and signed if the channel is secure anyway – maybe this was intended as an implementation of a secure channel. Therefore we considered both versions here, and it turns out that the encryption and signature as proposed are insufficient. If the channel is private, ProVerif is again able to prove Weak Anonymity (WA) for two honest and one corrupted bidder.

It is also clear that the protocol is neither Receipt-Free nor Coercion-Resistant for any of the proposed notions since the hashed bidding price in the first message can be used as a receipt. Even if this message is encrypted, the data used to encrypt (keys, random values) can be used as a receipt.

Note that for all properties ProVerif responds in a few seconds on a standard PC.

§ 4.3.5.2. Protocol by Brandt. The protocol by Brandt [Bra06] was designed to ensure maximal privacy in a completely distributed way. It exploits the homomorphic properties of a distributed threshold El-Gamal Encryption scheme for a secure multi-party computation of the winner.

Informal Description. The participating bidders and the seller communicate using a broadcast channel[10], e.g. a bulletin board (an append-only memory accessible for everybody). The bids are encoded as bit-vectors where each entry corresponds to a price. The protocol then uses linear algebra operations on the bid vectors to compute a function f_i, which returns a vector containing one zero if the bidder i submitted the highest bid, and only random numbers otherwise. To be able to compute this function in a completely distributed way, and to guarantee that no coalition of malicious bidders can break privacy, these computations are performed on the encrypted bids using homomorphic properties of a distributed n out of n threshold ElGamal encryption.

In a nutshell, the protocol realizes the following steps:

1. Firstly, the distributed key is generated: each bidder chooses his part of the secret key and publishes the corresponding part of the public key on the

[10]Note that in the original paper the authors suppose a "reliable broadcast channel, i.e. the adversary has no control of communication" [Bra06]. Yet even under this very strong assumption dishonest participants can impersonate other participants by submitting messages on their behalf, as the protocol does not include any authentication of the messages. In the following we consider an attacker in control of the network, however many attacks can also be executed analogously by dishonest parties – which are considered in the original paper – in the reliable broadcast setting.

bulletin board.

2. Each bidder then computes the joint public key, encrypts his offer using this key and publishes it on the bulletin board.
3. Then the auction function f is calculated for every bidder using some operations exploiting the homomorphic property of the encryption scheme.
4. The outcome of this computation (n encrypted vectors) are published on the bulletin board, and each bidder partly decrypts each value using his secret key.
5. These partial decryptions are published and can be combined to obtain the winner[11].

Formal Model. We need to model the distributed encryption scheme and the distributed computations. The protocol assumes a finite set of possible prices, which we model as constants $p_1, \ldots, p_q$. Assuming n bidders, we define the following equational theory. The first equation models steps 3 and 4 of the protocol:

$$\begin{aligned}&\mathtt{f}(\mathtt{enc}(b_1, pkey, r_1), \ldots, \mathtt{enc}(b_n, pkey, r_n), sk_i)\\ &= \mathtt{share}((\max_i\{b_i\}, \arg\max'_i\{b_i\}), (b_1, ..., b_n), pkey, sk_i, \mathtt{g}(r_1, .., r_n))\end{aligned}$$

where the function $\arg\max'_i$ returns the smallest i for which the maximum is reached, similar to the tie-breaking used in the protocol. The equation expresses the following properties of the function `f`: If we have bids $b_1, \ldots, b_n$, encrypted using the same joint public key $pkey$, some random values $r_1, \ldots, r_n$, and a part sk_i of the secret key we obtain a share (a partial decryption) of the function outcome, i.e. the tuple (winning price, id of the winner), for the same public and secret keys and a function of the used random values. Since the share will look slightly different depending on the bids even if winning bid is the same, we also include $b_1, \ldots, b_n$ and the random values $r_1, \ldots, r_n$ in the share. This is necessary to avoid false attacks in ProVerif. The next equation corresponds to step 5 of the protocol and uses the function $\mathtt{combine}(\mathtt{pk}(k_1), \ldots, \mathtt{pk}(k_n))$ which models the computation of the joint public key based on the individual ones.

$$\begin{aligned}&\mathtt{dec}(\mathtt{share}(m, x_1, \mathtt{combine}(\mathtt{pk}(k_1), \ldots, \mathtt{pk}(k_n)), k_1, r_1), \ldots,\\ &\mathtt{share}(m, x_q, \mathtt{combine}(\mathtt{pk}(k_1), \ldots, \mathtt{pk}(k_n)), k_n, r_n)) = m\end{aligned}$$

[11] This is a simplification w.r.t. the original protocol, where the partial decryptions are not published directly, but sent to the seller. He can combine them to obtain the result and learn the winner and winning price. He then publishes part of the partial decryptions on the bulletin board so that each bidder i can compute his f_i using his partial decryptions. Yet the published values are insufficient to obtain f_j for any other bidder $j \neq i$, i.e. each bidder only learns if he won or lost, but neither the winner nor the winning price if he is not the winner himself. In our simplified version everybody learns the winner and winning price, as it would be the case with a dishonest seller revealing the winner to everybody. As it turns out, even this simplified version is flawed.

The equation models that knowing all shares of the function outcome allows to decrypt it, if

— all shares have been constructed using the same joint public key, which was computed using the function `combine` from the individual public keys, and

— the individual public keys were computed from the same secret keys that were used to compute the shares.

Since the number of different prices q and the number of participants n are finite, we can enumerate all possible equations. In particular we can list all possible parameters of the function `f`, which allows us to enumerate all instances and replace the max and arg max functions which their actual values. As our definitions require at least two bidders and at least three different prices, we use $n = 2$ and $q = 3$. This yields a convergent equational theory, which allows ProVerif to verify all the tested properties in less than one second.

The process description is straightforward: an instance with two bidders (Listing 4.5 on the facing page) and one seller (Listing 4.6 on page 206) is described in Listing 4.7 on page 206.

Note that as the bidders have to exchange their bids we have to swap sending and receiving of the bids for both processes in order to avoid deadlocks. Note also that the events `bid` and `recBid` do not use the same data types concerning the bid, this is however not a problem since they do not occur simultaneously in the definition of a property.

Analysis. We use the same ProVerif techniques we discussed in the previous section. The protocol ensures none of the defined properties except for Weak Noninterference, mainly due the lack of authentication, even if all parties are trusted.

For *Non-Repudiation* we use the query

```
query b:price, m:int; event(won(b, m)) ==> event(bid(b, m)).
```

ProVerif finds the following attack: Due to the lack of authentication, the attacker can simulate a completely different protocol execution towards the seller (i.e. setting up keys, encrypting bids of his choice, doing the calculation, and publishing the shares), which allows to construct a trace with event `won`, but without any event `bid`.

In the case of Non-Cancellation we have the problem that `won` and `recBid` use different data types. We can however address this using the conjunction event `recBid_and_won` as above and the query

```
query m : price, pkey : shpkey, r : bitstring;
event(recBid_and_won(enc(priceToBit(p1), pkey, r), one, p2, one)).
```

```
let bidder1(b:price,chK:channel,chB:channel,chR:channel) =
  (* set up joint key: create and publish one part *)
  new shkey:shskeys;
  out(chK,pk(shkey));
  (* compute joint key using second public key *)
  in(chK,pk2:shpkeys);
  let pkey = combine(pk(shkey),pk2) in
  (* encrypt bid *)
  new r:bitstring;
  let bid1 = enc(priceToBit(b),pkey,r) in
  event bid(b,one);
  out(chB,bid1);
  (* receive other bid *)
  in(chB,bid2:bitstring);
  event recBid(bid2,two);
  (* compute partial decryption *)
  let res = win(bid1,bid2,shkey) in
  out(chR,res).

let bidder2(b:price,chK:channel,chB:channel,chR:channel) =
  (* set up joint key: create and publish one part *)
  new shkey:shskeys;
  in(chK,pk2:shpkeys);
  (* compute joint key using second public key *)
  out(chK,pk(shkey));
  let pkey = combine(pk2,pk(shkey)) in
  (* encrypt bid *)
  new r:bitstring;
  let bid1 = enc(priceToBit(b),pkey,r) in
  (* receive other bid *)
  in(chB,bid2:bitstring);
  event recBid(bid2,one);
  (* publish own bid *)
  event bid(b,two);
  out(chB,bid1);
  (* compute partial decryption *)
  let res = win(bid2,bid1,shkey) in
  out(chR,res).
```

Listing 4.5 – The bidders.

```
let seller(chR:channel,chW:channel) =
  (* receive partical decryptions *)
  in(chR,x:bitstring);
  in(chR,y:bitstring);
  (* compute winner *)
  let (wbid:price,winner:int) = dec(x,y) in
  event won(wbid,winner);
  out(chW,(wbid,winner)).
```

Listing 4.6 – The seller.

```
process
 bidder1(b1,chK,chB,chR) |
 bidder2(b2,chK,chB,chR) |
 seller(chR,chW)
```

Listing 4.7 – The aution process.

The query expresses that we look for a trace where the recorded bid was an encryption of the higher price `p1`, whereas the declared winning bid was the lower price `p2`, however from the same bidder `one`. Again ProVerif finds a trace violating the property: The adversary can simulate a different instance to the seller, hence replacing the initially recorded bid with a bid of his choice.

For Highest Price Wins ProVerif identifies a similar issue using the query

```
query p:price; event(won(p, two)).
```

i.e. we look for a trace where bidder `two` wins, although bidder `one` submits the highest bid at the highest possible price. The attack is quite simple: since there is no authentication, the attacker can replace the bid of bidder `one` with a lower one to win.

Although the protocol claims to be fully private, when checking for Strong Bidding-Price Secrecy, ProVerif finds an attack that allows to completely uncover a bidder's bid: since there is no authentication, an intruder can simulate all other parties with respect to a participant. He will generate secret keys, publish the according public keys and on reception of the attacked bidder's bid, simply copy it and claim that it is his own bid. Then the joint computation and decryption will take place, and the announced winning price will be attacked bidder's offer, which is hence public. Note that this is not an attack on the security of the computation, but on the structure of the protocol.

Hence the protocol does also not ensure any notion of privacy with corrupted voter(s), and neither receipt-freeness nor coercion-resistance as all of them are vulnerable to this attack.

It is also clear that the protocol does not ensure Strong Noninterference since the number of participants is public, which allows to distinguish instances with different number of participants. However we prove Weak Noninterference using `choice[]` (the use of `noninterf` leads to false attacks).

As above, the ProVerif-code is available online [Dre13].

§ 4.3.5.3. Protocol by Sako. Kazue Sako [Sak00] proposed a protocol for sealed-bid first-price auctions which hides the bids of losing bidders and ensures verifiability. It turns out to ensure all of our properties except for receipt-freeness and coercion-resistance. In the paper Sako gives a high-level description based on primitives ensuring certain properties. Then she also proposes two instantiations using some particular cryptographic primitives: the first one uses Elgamal [EG85] encryption, and the second one employs a probabilistic version of RSA [RSA78]. Note that in this protocol dishonest authorities can break privacy, but because of verifiability a manipulation of the auction outcome can be detected.

Informal Description. Informally, the protocol works as follows:

1. The authorities select a list of allowed bids $p_1, \ldots, p_m$.
2. For each allowed bid p_i, the authorities set up encryption and decryption algorithms E_{p_i} and D_{p_i} (in both implementations simply a public and private key pair). The encryption scheme must provide an indistinguishability property. The authorities publish the encryption algorithms (or public keys in the implementation) and the list of allowed bids on a bulletin board.
3. To bid for price p_i, a bidder encrypts a public constant c using E_{p_i}, signs it and publishes the bid $C_j = E_{p_i}(c)$ together with the signature on the bulletin board.
4. After the bidding phase is over, the authorities check the signatures and start decrypting all bids with the highest possible price $t = p_m$. If $D_t(C_j) = c$, then bid j was a bid for price t. If all decryptions fail, the authorities decrease t and try again. Each time a decryption is done, they publish a proof of correct decryption to enable verifiability. This can be a zero-knowledge proof, or it might be achieved by simply publishing the secret key.
5. To verify the outcome, anybody can verify the signatures, and check the proofs of correct decryption.

In the rest of this section we consider the implementation based on public and private key pairs as a concretization of the general encryption/decryption algorithms, however we abstract away of the precise encryption scheme. Note that dishonest authorities can break privacy since they have access to all secret keys, but because of verifiability a manipulation of the auction outcome can be detected.

```
let bidder(b:pkey,k:skey,chBB:channel) =
 new s:seed;
 let offer:bitstring = penc(bidval,b,s) in
 event bid(offer,pk(k));
 out(chBB,(offer,sign(offer,k))).
```

Listing 4.8 – The bidder.

Formal Model. To verify Sako's protocol we need to model public-key encryption, signatures and proofs of correct decryption. This can be done using the following equational theory:

$$\begin{aligned}
\texttt{checksign}(\texttt{sign}(m,k),\texttt{pk}(k)) &= m\\
\texttt{getmessage}(\texttt{sign}(m,k)) &= m\\
\texttt{dec}(\texttt{penc}(m,\texttt{pk}(k),s),k) &= m\\
\texttt{checkproof}(\texttt{decProof}(\texttt{penc}(m,\texttt{pk}(k),s),m,k),&\\
\texttt{penc}(m,\texttt{pk}(k),s),m,\texttt{pk}(k)) &= \texttt{true}\\
\texttt{checkproof}(\texttt{decProof}(\texttt{penc}(m,\texttt{pk}(k1),s),&\\
\texttt{dec}(\texttt{penc}(m,\texttt{pk}(k1),s),k2),k2),&\\
\texttt{penc}(m,\texttt{pk}(k1),s),&\\
\texttt{dec}(\texttt{penc}(m,\texttt{pk}(k1),s),k2),\texttt{pk}(k2)) &= \texttt{true}
\end{aligned}$$

The first two equations model signatures: If a signature on the message m is checked using the correct public key, we obtain the message m. Similarly the third equation models public-key encryption: A message encrypted with a public key can only be opened using the corresponding private key. The last two equations model proofs of correct decryption: The verification succeeds if the proposed plaintext is the actual decryption of the ciphertext under the claimed key, even if this decryption is not meaningful as the key is not the correct one.

Consider the ProVerif code in Listing 4.8 describing the behavior of a bidder in Sako's protocol. The bidder process has three parameters: the key `b`, corresponding to the key representing his bid, his secret key `k` used for signing and the channel to the bulletin board `chBB`. He draws a fresh random seed `s`, encrypts the constant `bidval` using the price-key `b`, computes a signature on the ciphertext, executes the event `bid` on his (signed) offer and sends it to the bulletin board.

The authority (in this case for an instance with two bidders) is modeled as shown in Listing 4.9 on the facing page. Firstly the bids are received from the bulletin board. Then the signatures are checked and the bids are decrypted using the key corresponding to the highest possible price, and the decryptions are published together with a proof. Finally, if the first bidder submitted a bid for the highest price, he is declared a winner, otherwise the second bid is checked. If none of the decryptions is correct, the authority decreases the price and tries

again.

Finally, the complete model for an instance with two bidders and one possible price is shown in Listing 4.10: the private key corresponding to the price is created and the public key is published. Similarly the private keys of the bidders are created and the public keys are published. Then we have three processes in parallel: The authority and two bidders.

Analysis. Similarly to the above examples, we test *Non-Repudiation* using the following query

```
query offer:bitstring,id:pkey;
event(won(offer,id)) ==> event(bid(offer,id)).
```

To also account for dishonest bidders, we give all data of the bidder except for his secret key to the intruder, and let him compose the message which is signed and sent to the bulletin board (see Listing 4.11 for the modified bidder's process). ProVerif proves that the property still holds.

Non-Cancellation is again tested using a process that tests if the conjunction of events `recBid` and `won` for a lower bid occurs, and then executes an event `bad`. However instead of using channels to forward the event parameters to the process we "inline" it into the authority process (i.e. include it as a macro at the end of the authority process). This is possible since anyway both events are executed inside this process, and necessary since otherwise ProVerif comes up with false attacks which essentially correspond to changing a bid inside the authority process during its computation of the winner. This is however not possible as the authority is assumed to be honest. With this modification ProVerif can conclude that a "bad" situation is unreachable using the query

```
query event(bad()).
```

Similarly for *Highest Price Wins* we also use a process that executes an event `bad` if a situation violating the property (i.e. an event `won` for bidder `two`) is detected, although here no inlining is necessary. ProVerif can again conclude that such an event is unreachable.

For *Weak Non-Interference* we use the `choice` operator, as the `noninterf` command leads to false attacks. We use a small python script to generate all the cases for two bidders and two prices, which can then successfully be checked using ProVerif.

Privacy is also tested using the `choice` operator. As the protocol reveals the winner and the winning price, the highest achievable privacy notion is Strong Bidding-Price Secrecy. Hence we test two situations, where in both situations

```
let authority(p1:skey,...,pm:skey,
 k1:pkey,...,kn:pkey,
 chBB1:channel,...,chBBn:channel,
 chO11:channel,...,chOn1:channel,...,
 chO1m:channel,...,chOnm:channel,chW:channel) =
  in(chBB1,(m1:bitstring, s1:bitstring));
  ...
  in(chBBn,(mn:bitstring, sn:bitstring));
  if checksign(s1,k1) = m1 && ... &&
   checksign(sn,kn) = mn then
    let dec11 = dec(m1,p1) in
    out(chO11,(m1,dec11,decProof(m1,dec11,p1)));
    ...
    let dec1n = dec(mn,p1) in
    out(chOn1,(mn,dec1n,decProof(mn,dec1n,p1)));
    if dec11 = bidval then
     event won(m1, p1);
     out(chW,(m1,s1,one,one))
    else if ...
     ...
    else if dec1n = bidval then
     event won(mn, pn);
     out(chW,(mn,sn,n,one))
    else
     ...
     let decm1 = dec(m1,pm) in
     out(chO1m,(m1,decm1,decProof(m1,decm1,pm)));
     ...
     let decmn = dec(mn,pm) in
     out(chOnm,(mn,decmn,decProof(mn,decmn,pm)));
     if decm1 = bidval then
      event won(m1, p1);
      out(chW,(m1,s1,one,m))
     else if ...
     ...
     else if decmn = bidval then
      event won(mn, pm);
      out(chW,(mn,sn,n,m)).
```

Listing 4.9 – The authority.

```
process
 new keyone:skey;
 new k1:skey;
 new k2:skey;
 out(chPKI,pk(keyone)) |
 out(chPKI,pk(k1)) |
 out(chPKI,pk(k2)) |
 authority(keyone,pk(k1),pk(k2),
   chBB1,chBB2,chO11,chO21,chO12,chO22,chW) |
 bidder(b1,k1,chBB1)) |
 bidder(b2,k2,chBB2))
```

Listing 4.10 – An instance with two bidders.

```
let bidder(b:pkey,k:skey,chBB:channel,chAd:channel)=
  new s:seed;
  out(chAd,(s,b));
  in(chAd,offer:bitstring);
  event send(offer,pk(k));
  out(chBB,(offer,pk(k))).
```

Listing 4.11 – The dishonest bidder.

bidder one bids the same highest price `one` and wins. In situation one bidder two bids price `two`, in situation two he bids `three`. Since he loses in both situations, no information about his bid should be leaked, and both situations should be observationally equivalent. If we suppose an honest authority, ProVerif is able to prove this result.

ProVerif can prove the result even if we add a corrupted bidder, which might appear suprising as it is possible to copy bids. However, as only the losing bids are required to be private, an attack similar to the one on the protocol by Brandt does not work: the copied bid does not influence the outcome as anyway the honest bidder with the higher bid wins.

Note that the protocol is neither receipt-free nor coercion-resistant as the random values used to encrypt the constant can be used as a receipt.

Remark In the literature an attack on fairness was described [ABN10, FLPQ13]. The attack targets the implementation using the ElGamal encryption and works as follows. A dishonest bidder encrypts the constant using a key of his choice, but using zero as a random value. He obtains a ciphertext of the form $(1, M)$, which decrypts under any private key. A dishonest authority can then decrypt the bid at any chosen price, for example one euro higher than the highest other bid.

This attack violates Highest Price Wins: a corrupted bidder that did not submit the highest bid wins - yet still we did not identify it for several reasons:

— We assume an honest authority, whereas the attack requires the authority to collude with the bidder.

— As we use an abstract model of the encryption, the underlying weakness of the ElGamal scheme is not represented in model. Our equational theory permits decryption only using the secret key corresponding to the public key used for encryption. This not surprising, as we wanted to check the high-level protocol, not the ElGamal-based implementation.

If we decide to consider dishonest authorities, we obviously have even simpler attacks: the authorities can simply announce a winner and winning price of their choice. To circumvent this problem, the protocol was designed to be verifiable, i.e. the bidders can verify the announced winner as the authorities provide proofs of correct decryption. In Section 4.4.3.1 we analyze verifiability in Sako's protocol in detail, and show that this attack will be detected during the verification phase. This allows us to argue that the assumption that the authority is honest is reasonable - if it behaves dishonestly, this can be detected in the verification phase.

4.3.6 — Summary

In this section we formalized auction protocols in the Applied π-Calculus and modeled fairness, authentication and privacy properties. We verified three case studies using ProVerif: the protocol by Curtis et al. [CPS07], the protocol by Brandt [Bra06] and the protocol by Sako [Sak00].

We identified several problems with the protocol by Curtis et al.: We need to encrypt the first message to ensure Weak Noninterference, and we need to add authentication for the messages from the RA to the seller to ensure Non-Repudiation. We also need to introduce synchronization for privacy. Even with these modifications, ProVerif still finds attacks for Highest Price Wins and Non-Cancellation by mixing several instances. This could probably be addressed by using different keys for different instances and counting the number of bids, however we were unable to analyze this in ProVerif because of a technical limitation of the tool: counting the number of bids requires to maintain state information, which is not supported in ProVerif.

The protocol by Brandt ensures Weak Noninterference, however fails to achieve any other property (including privacy) due to the lack of authentication.

Finally the protocol by Sako turns out to ensure all properties except for receipt-freeness and coercion-resistance, assuming an honest authority.

4.4 Verifiability in Auctions

In the previous section we discussed Fairness, Privacy and Authentication properties of auction protocols in the Applied π-Calculus. In this section we concentrate on Verifiability, but using a more general model that can be instantiated in the symbolic model (where we use the Applied π-Calculus again) and computational model (where we use CryptoVerif).

Auction verifiability is easy to achieve in isolation, as for example in the English "shout-out" auction. However, maintaining verifiability while ensuring other properties such as privacy is far more difficult. Yet in the literature verifiability is often claimed together with other properties, but these papers typically only provide informal arguments, e.g. [Sak00, Bra06]. To address this issue, we propose a generic formal framework to analyze verifiability independent of the type of auction implemented in the protocol.

The main contribution of this section is to identify a set of scheme-independent definitions, which, taken together, cover verifiability of auctions. To this end, we focus on the bidders (distinguishing verifiability for losing bidders from the winning bidder) and the seller. We present this framework as a set of formal verifiability tests. Moreover, we investigate the auction protocols by Sako [Sak00] and by Curtis et al. [CPS07]. Sako claims verifiability without a formal proof, and we are able to give a computational and a symbolic proof for her protocol as our first case study. For the second protocol by Curtis et al. we identify several shortcomings with respect to verifiability.

The rest of this section is structured as follows. In Section 4.4.1, we describe the new abstract model of auction protocols. In Section 4.4.2, we formalize the verifiability definitions, taking into account the point of view of the seller and the bidders. In Section 4.4.3, we then apply our model to the two examples.

4.4.1 — A Different Model of Auction Protocols

As explained above, we use a very general model of auction protocols in this section. We only specify the parts necessary to define verifiability, and remain very abstract to allow instantiations in different concrete models.

We consider a set of bidders $\mathcal{B}$ and a seller S. We do not model other parties as only bidders and the seller verify the execution of the protocol.

Bids are of type *Bid* (in the simplest case just a price). When being submitted the bids might be encrypted or anonymized to ensure privacy, hence we use the type *EBid* for such bids. We assume that there is a public list $\mathbb{L}$ of length n and type $List(EBid)$ of all submitted bids, for example a bulletin board. To define the soundness of the verification tests we need a mapping between both types, i.e. a function $getPrice\colon EBid \mapsto Bid$ that gives the bid for an encrypted bid. This

function does not need to be computable for any party, as it is only used in the soundness definition.

Bidders have to register at some point, or are otherwise authenticated when bidding, in order to be able to obtain their goods once the auction has ended. This could for example be implemented using signatures, authentication tokens, MACs etc.. Therefore we require a function $isReg\colon EBid \mapsto \{true, false\}$ that returns *true* if a bid was submitted by a registered bidder, and not modified – this integrity protection is necessary to prevent manipulation of bids.

In addition, we require a public function that – given a list of bids – computes the index of the winning bid within the list of all bids: $win\colon List(Bid) \mapsto Index$. This might simply be the index of the maximal bid among all bids, but there may be more complex operations to determine this index depending on the type of auction or to deal with ties (i.e. several maximal bids).

Lastly, we assume that the variable *winBid* of type *Index* refers to the index of the announced winning bid at the end of the auction, and that each bidder has a variable *myBid* of type *Index* that refers to the index of his bid in $\mathbb{L}$.

Note that for a list l we write $l[i]$ to denote the i-th element of the list starting with 1, and $Indices(l)$ to denote the set of indices of l, i.e. $\{1, \ldots, n\}$ if l contains n elements.

Definition 71 *An auction protocol is a tuple* ($\mathcal{B}$, S, $\mathbb{L}$, *getPrice*, *isReg*, *win*, *winBid*) *where*

— $\mathcal{B}$ *is the set of bidders,*

— S *is the seller,*

— $\mathbb{L}$ *is a list of all submitted bids,*

— $getPrice\colon EBid \mapsto Bid$ *is a function mapping submitted bids to bids,*

— $isReg\colon EBid \mapsto \{true, false\}$ *is a function that returns true if a bid was submitted by a registered bidder,*

— $win\colon List(Bid) \mapsto Index$ *is a function that returns the index of the winning bid,*

— *winBid is a variable referring to the index of the winning bid at the end of the auction.*

4.4.2 — Defining Verifiability

Now we formally define verifiability for auction protocols with respect to the new generalized model. In the first part we consider only first-price auctions. Thereafter we generalize the definitions to account for second-price, multi-price, and other types of auctions.

§ 4.4.2.1. First-Price Auctions. To understand which verifications are needed, we start by discussing three different stakeholder's perspectives:

— A *losing bidder* wants to be convinced that he actually lost, i.e. that:
  – the winning bid was actually superior to his bid (as defined by the *win* function), and
  – that the winning bid was submitted by another bidder (preventing both seller and auctioneer from maliciously adding or manipulating bids to influence the final price).

— A *winning bidder* wants to check that:
  – he actually submitted the winning bid,
  – the final price is correctly computed,
  – all other bids originated from bidders, and
  – no bid was modified.

  Together, these verification checks ensure that the winning bidder is indeed the correct winner, for the correct price. Moreover, the last two checks ensure that the auction process was only influenced by legitimate bidders – neither seller nor auctioneer influenced the process.

— The *seller* wants to verify that:
  – the announced winner is correct, and
  – the winning price is correct,

  in particular if the outcome of the auction was not determined publicly (e.g. privately by the auctioneer, or using distributed computations among the bidders).

To execute these verifications, we introduce the notion of *Verification Tests*.

Definition 72 (Verification Test) *We define a* Verification Test *as an efficient terminating algorithm that takes as input the data visible to a participant of an auction protocol and returns a Boolean value.*

We deliberately do not specify more details at this point as they will depend on the underlying protocol model. Such a test could be a logical formula (whose size is polynomial in the input) in a symbolic model or a polynomial-time Turing-machine in a computational model. Obviously there can be different tests for different participants (e.g. for bidders and the seller), since they may have different views of the protocol execution.

We then define verifiability as follows.

Definition 73 (Verifiability – First-Price Auctions) *An auction protocol* ($\mathcal{B}$, S, $\mathbb{L}$, *getPrice*, *isReg*, *win*, *winBid*) *ensures* Verifiability *if we have Verification Tests* rv_s, rv_w, ov_l, ov_w, ov_s *respecting the following soundness conditions:*

1. *Registration and Integrity Verifiability (RV):*
   - *Anyone can verify that all bids on the list were submitted by registered bidders:*
     $rv_s = true \implies \forall b \in \mathbb{L}\colon isReg(b) = true$
   - *Anyone can verify that the winning bid is one of the submitted bids:*
     $rv_w = true \implies winBid \in Indices(\mathbb{L})$
2. *Outcome Verifiability (OV):*
   - *A losing bidder can verify that his bid was not the winning bid:*
     $ov_l = true \implies myBid \neq win(getPrice(\mathbb{L}))$
   - *A winning bidder can verify that his bid was the winning bid:*
     $ov_w = true \implies myBid = win(getPrice(\mathbb{L}))$
   - *The seller can verify that the winning bid is actually the highest submitted bid:*
     $ov_s = true \implies winBid = win(getPrice(\mathbb{L}))$

as well as the following completeness condition:

- *If all participants follow the protocol correctly, the above tests succeed (i.e., the implications hold in the opposite direction,* $\Longleftarrow$*, as well).*

where – with abuse of notation – we write $getPrice(\mathbb{L})$ *for* $getPrice(\mathbb{L}[1])$, …, $getPrice(\mathbb{L}[n])$.

Consider the perspective of a losing bidder: He can verify that his bid was not the winning bid (ov_l), and that the winning bid was among the ones submitted by registered bidders, which were also not modified (rv_s and rv_w). Similarly a winning bidder can check that his bid was actually the winning bid (ov_w), and that the other bids were submitted by other bidders and not modified (rv_s). Lastly, the seller can also check that the bids using for computing the winner were submitted only by registered bidders (rv_s and rv_w), and that the outcome was correct (ov_s). Hence these tests cover all the verifications discussed above.

In the case of soundness, we require the conditions to hold even in the presence of malicious participants (since the tests should check if they did their work correctly), whereas in the case of completeness we only consider honest participants. This is necessary as otherwise e.g. a dishonest auctioneer could announce the correct result, but publish incorrect evidence. Hence the verification tests fail although the outcome is correct, but this is acceptable since the auctioneer did not "work correctly" in the sense that he deviated from the protocol specification.

Definition 73 can be applied to sealed-bid auctions, where all bids are submitted in a private way, as well as English auctions, where the price increases with each

publicly announced bid. These latter are verified by applying the verification tests after each price increase.

Example 22 *Consider a simple auction system where all bidders publish their (not encrypted and not signed) bids on a bulletin board, and at the end of the bidding phase the auctioneer announces the winner. In this case there is a simple test for rv_w: anyone can simply test if the winning bid is one of the published ones. However there is no test for rv_s since bids are not authenticated. If we require bidders to sign their bids before publishing them, we also have a simple test for rv_s: verifying the signatures.*

It is clear that we have simple tests for ov_l, ov_w and ov_s since everybody can compute the winner on the public list of unencrypted bids. This however means that the protocol ensures no privacy, and no fairness since a bidder can choose his price depending on the previously submitted bids. If we add encryption for the bids to address this shortcoming, the situation becomes more complex and the auctioneer has to prove that he actually computed the winner correctly, for example using zero-knowledge proofs.

§ 4.4.2.2. Other Types of Auctions. Our definition can be extended to other auctions, including second-price auctions, more general $(M + 1)$st-price auctions, and even bulk-good auctions that have multiple winners at different prices. In a bulk-good auction, a seller offers N units of a good, and bidders can make offers such as "I want to buy X units at price Y". In that case there may be a several winners, where each of them pays a different price. More generally, we could also imagine situations where the winner has to pay the mean of the first three bids, or other more complex values. In these types of auctions the price does also depend on the other submitted bids, and not only on the winning bid.

To deal with this, we enrich our model of an auction protocol with a type *Price*. The function *win* now returns a list of winners and prices $win\colon List(Bid) \mapsto List(Index \times Price)$. We also assume that for each bidder there is a variable *myPrice* of type *Price* instantiated as the price announced to a winning bidder respectively. Similarly *winBid* is now instantiated as a list of indices of bids and prices ($List(Index \times Price)$).

For such auctions, registration verifiability does not change, but the winner(s) and the seller also want to verify the price they pay to prevent a malicious party from manipulating the price(s).

In the following definition we mark differences to Definition 73 in **bold**.

Definition 74 (Verifiability – Other Types of Auctions) *An generalized auction protocol* ($\mathcal{B}$, S, $\mathbb{L}$, *getPrice*, *isReg*, *win*, *winBid*, *winPrice*) *ensures* Verifiability *if we have Verification Tests rv_s, rv_w, ov_l, ov_w, ov_s respecting the following conditions:*

1. *Soundness:*
   (a) *Registration and Integrity Verifiability (RV):*
      — *Anyone can verify that all bids on the list were submitted by registered bidders:*
      $rv_s = true \implies \forall b \in \mathbb{L}\colon isReg(b) = true$
      — *Anyone can verify that the winning bids are among the submitted bids:*
      $rv_w = true \implies (\boldsymbol{(b,p) \in winBid \Rightarrow} b \in Indices(\mathbb{L}))$
   (b) *Outcome Verifiability (OV):*
      $\boldsymbol{Let\ winBidsPrices = win(getPrice(\mathbb{L}))}$**.**
      — *A losing bidder can verify that his bid was not the winning bids:*
      $ov_l = true \implies \forall \boldsymbol{i}\colon \boldsymbol{winBidsPrices[i] \neq (myBid, \_)}$
      — *A winning bidder can verify that his bid was among the winning bids, and that his price is correct:*
      $ov_w = true \implies \exists \boldsymbol{i}\colon \boldsymbol{winBidsPrices[i] = (myBid, myPrice)}$
      — *The seller can verify that the list of winners and the winning prices are correctly determined:*
      $ov_s = true \implies \boldsymbol{winBid = winBidsPrices}$
2. *Completeness: If all participants follow the protocol correctly, the above tests succeed (i.e., the implications hold in the opposite direction, $\Longleftarrow$, as well).*

where – with abuse of notation – we write getPrice($\mathbb{L}$) *for getPrice*($\mathbb{L}[1]$)*, …, getPrice*($\mathbb{L}[n]$)*.*

Note that e.g. in the case of a second-price auction verifying the price, for example in test ov_w, may implicitly include some more registration verification, namely checking that the second-highest bid was actually submitted by a bidder. Otherwise a malicious seller could add a higher second-highest bid or manipulate the existing one to achieve a higher selling price. This is however included in our model as the function *win* only works on the list $\mathbb{L}$, hence adding another bid later on to manipulate the bidding price violates the test, and adding or manipulating a bid in $\mathbb{L}$ violates rv_s.

4.4.3 — Case Studies

We now discuss the two case studies: the protocols by Sako and Curtis et al..

§ 4.4.3.1. Protocol by Sako. We use the protocol by Sako [Sak00] already described in Section 4.3.5.3. In our high-level model, we formalize this protocol using a set of bidders $\mathcal{B}$ and a seller S. The list of all submitted bids $\mathbb{L}$ is published on the bulletin board. The function *getPrice*(C) decrypts the bid by trying all

possible prices until the correct value is found, i.e. until $D_t(C) = c$ (as the authorities would), and then returns t. The function *isReg* returns if a bid was submitted by one of the registered bidders. The function *win* returns the index of the highest bid, and *winBid* will point to the index of the winning bid at the end of the auction as announced by the authorities on the bulletin board.

The verification tests proposed by Sako can be translated to our framework as follows: The test rv_s simply checks all the signatures. For rv_w one can check if the encrypted value appears in the list of bids on the bulletin board when the winner is announced. Finally the test for ov_l, ov_w and ov_s works as follows: Any participant can check that all decryptions corresponding to a potentially higher bid were unsuccessful (i.e. the result was different from M_t), and verify the proofs of correct decryption. To check if he won or lost, a bidder can simply compare his bid to the winning price. Similarly the seller can check if the announced winning bid is actually the winning bid.

We analyze the protocol in two different settings. We start using the Applied π-Calculus and ProVerif and give a first, symbolic analysis. Then, to show that our high-level model can also be instantiated in a computational setting, we provide a computational analysis with the help of CryptoVerif.

Verifiability in the Symbolic Setting. Based on the modeling of the protocol from Section 4.3.5.3 we also model the proposed tests in the Applied π-Calculus and formally show that they are actually sound and complete. The proofs are done by ProVerif for a simple case (e.g. only two bidders), and then extended manually to the general case. This is necessary as ProVerif is unable to handle the general case directly.

In order to verify the verification tests we model them as processes: they receive the necessary data on some channels and output whether they accepted this input or not. This also allows us to check soundness and completeness of the tests in ProVerif, since these properties can be modeled as reachability properties such as “If the input to the test was generated by honest participants according to the protocol, can the process “test” emit a message “KO”?” (which corresponds to completeness) or “Is it possible for an attacker to send messages to the process “test” such that it emits a message “OK” although the input is not correct?” (which corresponds to soundness).

The first test rv_s, described in Listing 4.12 on the next page, is actually a simple implementation of our soundness condition ($\forall b \in \mathbb{L} : \mathit{isReg}(b) = \mathit{true}$): It receives all bidders' public keys and the bids, and then verifies all signatures. We show that is sound and complete in Theorem 68.

Theorem 68 *The test rv_s (see Listing 4.12 on the next page) for the protocol by Sako [Sak00] is sound and complete.*

```
let testrvs(chRVS:channel,k1:pkey,...,kn:skey,
            chBB1:channel,...,chBBn:channel) =
 in(chBB1,(m1:bitstring,s1:bitstring));
 ...
 in(chBBn,(mn:bitstring,sn:bitstring));
 if checksign(s1,k1) = m1 && ... &&
  checksign(sn,kn) = mn then
   out(chRVS,OK)
 else
   out(chRVS,KO).
```

Listing 4.12 – The test rv_s.

```
let testrvs(chRVS:channel,k1:pkey,...,kn:skey,
            chBB1:channel,...,chBBn:channel) =
 in(chBB1,(m1:bitstring,s1:bitstring));
 ...
 in(chBBn,(mn:bitstring,sn:bitstring));
 if checksign(s1,k1) = m1 && ... &&
  checksign(sn,kn) = mn then
   out(chRVS,OK);
   event accept(m1,s1,...,mn,sn)
 else
   out(chRVS,KO);
   event reject().
```

Listing 4.13 – The test rv_s with events.

Proof We start by showing that it is sufficient to verify the case of only one bid, and that this generalizes to situations with many bids. More precisely, we show that

$$rv_s = true \Rightarrow \forall b \in \mathbb{L}\colon isReg(b) = true$$

holds for any length n of $\mathbb{L}$ (corresponding to the number of bids). Listing 4.12 on the facing page gives the test rv_s for n bids. Let $testrvs_i$ denote analogously the test for i bids, and $testrvs_i \xrightarrow{*} OK$ denote that – given a protocol execution – after some interactions and reductions it outputs "OK" on channel `chRVS`. We assume that the test is sound and complete for one bid, i.e. for soundness

$$testrvs_1 \xrightarrow{*} OK \Rightarrow isReg(\mathbb{L}) = true \tag{4.1}$$

and for completeness

$$isReg(\mathbb{L}) = true \Rightarrow testrvs_1 \xrightarrow{*} OK \tag{4.2}$$

where $\mathbb{L}$ contains only one entry.

Then we can see from the code that

$$\begin{gathered} testrvs_n \xrightarrow{*} OK \\ \Downarrow \\ \forall 1 \leq i \leq n : testrvs_1 \{^{ki}/_{k1}, {}^{chBBi}/_{chBB1}\} \xrightarrow{*} OK \end{gathered}$$

since $testrvs_n$ checks the signatures of all bids on channels `chBB1` to `chBBn` using the keys `k1` to `kn`, and $testrvs_1 \{^{ki}/_{k1}, {}^{chBBi}/_{chBB1}\}$ checks the signature of the bid i on channel `chBBi` using the key `ki`. If we suppose now that $testrvs_1$ is sound (4.1), we obtain

$$\begin{gathered} \forall 1 \leq i \leq n : testrvs_1 \{^{ki}/_{k1}, {}^{chBBi}/_{chBB1}\} \xrightarrow{*} OK \\ \Downarrow \\ \forall 1 \leq i \leq n : isReg(List(\mathbb{L}[i])) = true \\ \Downarrow \\ \forall b \in \mathbb{L}\colon isReg(b) = true \end{gathered}$$

We can make a similar argument for completeness:

$$\begin{gathered}\forall b \in \mathbb{L}\colon \mathit{isReg}(b) = \mathit{true} \\ \Downarrow \\ \forall 1 \leq i \leq n : \mathit{isReg}(\mathit{List}(\mathbb{L}[i])) = \mathit{true} \\ \Downarrow \\ \forall 1 \leq i \leq n : \mathit{testrvs}_1 \left\{ {}^{ki}/_{k1}, {}^{chBBi}/_{chBB1} \right\} \xrightarrow{*} \mathit{OK} \\ \Downarrow \\ \mathit{testrvs}_n \xrightarrow{*} \mathit{OK}\end{gathered}$$

Hence it is sufficient to check the case of only one bid, i.e. (4.1) and (4.2). As hinted above, these are reachability properties ("The test will not output "OK" if a bid was not submitted by a registered bidder", "The test will not output KO if the bid was correctly submitted by a registered bidder"). We prove (4.1) and (4.2) automatically using ProVerif.

For (4.1) we add two events `accept` (cf. Listing 4.13 on page 220) and `send` (inside a registered bidder) to the model, and ProVerif shows that any event `accept` is preceded by a corresponding `send`, i.e. the test only accepts bids sent by registered bidders:

```
query m:bitstring, s:bitstring;
event(accept(m, s)) ==> event(send(m, s)).
```

For (4.2) we add an event event `reject` in the process `testrvs` (cf. Listing 4.13 on page 220) and ProVerif shows that it is unreachable if an honest bidder sends his bid over a private channel[12]:

```
query event(reject()).
```

The code used is available online [Dre13]. □

The second test rv_w, given in Listing 4.14 on the next page, takes all published bids on channel `chBB` and the winning bid published on channel `chW` and checks if the winning bid is among the published ones. This is again the direct implementation of the soundness condition, and we show that it is sound and complete in the following theorem.

Theorem 69 *The test rv_w (see Listing 4.14 on the facing page) for the protocol by Sako [Sak00] is sound and complete.*

[12]This is necessary, since otherwise the attacker might send a bogus bid instead leading to a false "attack". Another solution would be to use the "passive" attacker in ProVerif, which cannot modify messages. As we are only interested in the reachability of the event `reject`, both approaches should be equivalent: the passive attacker might learn more since he has access to the messages, but he cannot exploit this as he is not allowed to create or modify messages.

```
let testrvw(chRVW:channel,chW:channel,
            chBB1:channel,...,chBBn:channel) =
 in(chBB1,(m1:bitstring,s1:bitstring));
 ...
 in(chBBn,(mn:bitstring,sn:bitstring));
 in(chW,(m:bitstring,s:bitstring,ind:int,price:int));
 if (m1 = m && s1 = s) || ... || (mn = m && sn = s) then
  out(chRVW,OK)
 else
  out(chRVW,KO).
```

Listing 4.14 – The test rv_w.

```
let testrvw(chRVW:channel,chW:channel,
            chBB1:channel,...,chBBn:channel) =
 in(chBB1,(m1:bitstring,s1:bitstring));
 event onBB(m1,s1);
 ...
 in(chBBn,(mn:bitstring,sn:bitstring));
 event onBB(mn,sn);
 in(chW,(m:bitstring,s:bitstring,ind:int,price:int));
 if (m1 = m && s1 = s) || ... || (mn = m && sn = s) then
  out(chRVW,OK);
  event accept(m,s);
 else
  out(chRVW,KO);
  event reject().
```

Listing 4.15 – The test rv_w with events.

Proof Again, Listing 4.14 on the previous page gives the test rv_w for n bids, and let $testrvw_i$ denote the test rv_w for i bids. $testrvw_i \xrightarrow{*} OK$ denotes that – given a protocol execution – after some interactions and reductions the test outputs "OK" on channel `chRVW`. Similarly to rv_s, we automatically prove using ProVerif that in the case of only one bid, the test only accepts the winning bid if it was published on the bulletin board (soundness):

$$testrvw_1 \xrightarrow{*} OK \Rightarrow winBid \in Indices(\mathbb{L}) \tag{4.3}$$

using the query

```
query m:bitstring, s:bitstring;
event(accept(m, s)) ==> event(onBB(m, s)).
```

and the annotated process in Listing 4.15 on the preceding page.

We also show that the test succeeds if the authority follows the protocol (completeness):

$$winBid \in Indices(\mathbb{L}) \Rightarrow testrvw_1 \xrightarrow{*} OK \tag{4.4}$$

where $\mathbb{L}$ contains only one entry, using the query

```
query event(reject()).
```

using the same annotations (cf. Listing 4.15 on the previous page), and an honest bidder and an honest authority interacting over private channels.

We now have to show that this generalizes to situations with many bids. More precisely, we need to prove that

$$rv_w = true \Rightarrow winBid \in Indices(\mathbb{L})$$

holds for any length n of $\mathbb{L}$ (corresponding to the number of bids). Again, by analyzing the code we have

$$\begin{gathered} testrvw_n \xrightarrow{*} OK \\ \Downarrow \\ \exists 1 \leq i \leq n : testrvw_1 \left\{ {}^{chBBi}/_{chBB1} \right\} \xrightarrow{*} OK \end{gathered}$$

since $testrvw_n$ checks if the winning bid received on channel `chW` equals one of the bids received on channels `chBB1` to `chBBn`, and $testrvw_1 \left\{ {}^{chBBi}/_{chBB1} \right\}$ checks if the winning bid received on channel `chW` equals the bid i received on channel

`chBBi`. Using (4.3), we obtain that

$$\begin{gathered}\exists 1 \leq i \leq n : testrvw_1 \{^{chBBi}/_{chBB1}\} \xrightarrow{*} OK \\ \Downarrow \\ \exists 1 \leq i \leq n : winBid \in Indices(List(\mathbb{L}[i])) \\ \Downarrow \\ winBid \in Indices(\mathbb{L})\end{gathered}$$

We can make a similar argument for completeness:

$$\begin{gathered}winBid \in Indices(\mathbb{L}) \\ \Downarrow \\ \exists 1 \leq i \leq n : winBid \in Indices(List(\mathbb{L}[i])) \\ \Downarrow \\ \exists 1 \leq i \leq n : testrvw_1 \{^{chBBi}/_{chBB1}\} \xrightarrow{*} OK \\ \Downarrow \\ testrvw_n \xrightarrow{*} OK\end{gathered}$$

using (4.4). □

For the outcome verification tests we employ a similar approach. Firstly note that for Sako's protocol the tests ov_l, ov_w and ov_s are all the same, described by process `testov` (Listing 4.16 on the following page). The winning bid, the winning price and the winner's index are announced on channel `chW`. Morevover the proofs of correct decryption are published on channels `chOij`. The test then takes the first set of proofs of correct decryption and verifies them (lines 5-9). Then it checks if the announced winning price and winner are correct with respect to the obtained decryptions, i.e. if one of the decryptions corresponds to the constant, and if this is the one announced as the winner (lines 10-17). If within the set no decryption corresponds to the constant, it repeats the same checks for the next set of decryptions and proofs (lines 18-35). Note that the test accepts the case when all decryptions for all prices fail and the winning price is none of the checked ones. This might be the case if the authority correctly followed the protocol, but the bidders sent wrong encryptions. We show that this test is sound and complete in Theorem 70.

Theorem 70 *The test* `testov` *(see Listing 4.16 on the next page) for the protocol by Sako [Sak00] is sound and complete.*

Proof Again, Listing 4.16 on the following page gives the test `testov` for n bids and m possible prices. Let $testov_{i,j}$ denote the test for i bids and j possible prices, and $testov_{i,j} \xrightarrow{*} OK$ denote that – given a protocol execution – after some interactions and reductions it outputs "OK" on channel `chOV`, and $k = winBid$

```
let testov(chOV:channel,chW:channel,pk1:pkey,...,pkm:pkey,
           chO11:channel,...,chOn1:channel,...,
           chO1m:channel,...,chOnm:channel) =
 in(chW,(m,s,w,p));
 in(chO11,(c1,m1,p1));
 ...
 in(chOn1,(cn,mn,pn));
 if checkproof(p1,c1,m1,pk1) && ... &&
  checkproof(pn,cn,mn,pk1) then
   if p = one && ((w = one && m1 = bidval) || ... ||
    (w = n && m1 <> bidval && ... && m(n-1) <> bidval &&
    mn = bidval))
     then
      out(chOV,OK)
   else
     if m1 <> bidval && ... && mn <> bidval
      && p <> one then
       ...
       in(chO1m,(c1,m1,p1));
       ...
       in(chOnm,(cn,mn,pn));
       if checkproof(p1,c1,m1,pkm) && ... &&
        checkproof(pn,cn,mn,pkm) then
         if p = m && ((w = one && m1 = bidval) || ... ||
          (w = n && m1 <> bidval && ... &&
          m(n-1) <> bidval && mn = bidval)) then
           out(chOV,OK)
         else
          if m1 <> bidval && ... && mn <> bidval &&
           p <> m then
            out(chOV,OK)
          else
            out(chOV,KO)
       else
         out(chOV,KO)
     ...
 else
   out(chOV,KO).
```

Listing 4.16 – The test for $ov_{\{l,w,s\}}$.

```
let testov(chOV:channel,chW:channel,pk1:skey,pk2:skey,
           chO11:channel,chO21:channel) =
  in(chW,(m:bitstring,s:bitstring,w:int,p:int));
  in(chO11,(c1:bitstring,m1:bitstring,p1:bitstring));
  in(chO21,(c2:bitstring,m2:bitstring,p2:bitstring));
  if (checkproof(p1,c1,m1,pk(pk1))) &&
   (checkproof(p2,c2,m2,pk(pk1))) then
    if p = one && w = one && m1 = bidToBit(bidval) then
      if dec(c1,pk1) <> bidToBit(bidval) then
        event bad() else 0
    else
    if p = one && w = two && m1 <> bidToBit(bidval) &&
     m2 = bidToBit(bidval) then
      if dec(c1,pk1) = bidToBit(bidval) ||
       dec(c2,pk1) <> bidToBit(bidval) then
        event bad() else 0
    else
    if m1 <> bidToBit(bidval) && m2 <> bidToBit(bidval)
     && p <> one then
      if dec(c1,pk1) = bidToBit(bidval) ||
       dec(c2,pk1) = bidToBit(bidval) then
        event bad()
      else 0
    else
      event KO()
  else
    event KO().

process
 in(chPk1,(pk1:skey));
 in(chPk2,(pk2:skey));
 testov(chOV,chW,pk1,pk2,chO11,chO21)
```

Listing 4.17 – Proving soundness of $ov_{\{l,w,s\}}$.

the index of the winning bid. Using ProVerif we show that in the case of $i = 2$ bids and $j = 1$ price, the test only accepts the announced winner and winning price if they are correct (soundness):

$$\begin{array}{c} testov_{2,1} \xrightarrow{*} OK \\ \Downarrow \\ (getPrice(\mathbb{L}[winBid]) = one \Rightarrow winBid = win(getPrice(\mathbb{L}))) \end{array} \tag{4.5}$$

To prove this using ProVerif we use the process depicted in Listing 4.17 on the previous page. The idea is the following: the adversary generates the input for the test, i.e. chooses the keys, generates the bids and proofs of decryption and announces the winner. Every time the test accepts an outcome based on the proofs, we check if the decryptions using the secret key actually give the values they are supposed to give. For example if the winner is bidder `one` at price `one`, his bid must decrypt correctly under key `pk1` to the constant. Similarly, if the winner is bidder `two` at price `one`, his bid must decrypt correctly under key `pk1` to the constant and the other bidder's bid must not decrypt correctly under this key. If such a test fails, an event `bad` is executed. Using the following query ProVerif can then prove that such an event is unreachable:

```
query event(bad()).
```

We also show that the test succeeds if the authority follows the protocol correctly (completeness):

$$\begin{array}{c} (getPrice(\mathbb{L}[winBid]) = one \Rightarrow winBid = win(getPrice(\mathbb{L}))) \\ \Downarrow \\ testov_{2,1} \xrightarrow{*} OK \end{array} \tag{4.6}$$

using – once again – a variant of the process `testov` containing events `reject` where the process outputs `KO` and the query

```
query event(reject()).
```

We now show that this generalizes to situations with many bids and prices. We start by discussing soundness, the proofs of completeness are analogous. First we prove the generalization to m prices, but still for two bids. We have to show that the following holds:

$$testov_{2,m} \xrightarrow{*} OK \Rightarrow winBid = win(getPrice(\mathbb{L}))$$

By analyzing the code we can see that

$$\begin{gathered} testov_{2,m} \xrightarrow{*} OK \\ \Downarrow \\ \forall 1 \leq i \leq m : testov_{2,1} \{^{pki}/_{pk1}, ^{chO1i}/_{chO11}, ^{chO2i}/_{chO21}, ^{i}/_{one}\} \xrightarrow{*} OK \end{gathered}$$

i.e. that if $testov_{2,m}$ holds for the winner input from channel `chW`, prices (i.e. public keys) `pk1` to `pkm`, decryptions and proofs from channels `chO11` to `chO2m`, then for any i

$$testov_{2,1} \{^{pki}/_{pk1}, ^{chO1i}/_{chO11}, ^{chO2i}/_{chO21}, ^{i}/_{one}\}$$

holds for the same winner, but only checking price i instead of one, and hence considering only the price (key) `pki` and proofs from channels `chO1i` and `chO2i`. Using (4.5), we deduce that:

$$\begin{gathered} \forall 1 \leq i \leq m : testov_{2,1} \{^{pki}/_{pk1}, ^{chO1i}/_{chO11}, ^{chO2i}/_{chO21}, ^{i}/_{one}\} \xrightarrow{*} OK \\ \Downarrow \\ \forall 1 \leq i \leq m : getPrice(\mathbb{L}[winBid]) = i \Rightarrow winBid = win(getPrice(\mathbb{L})) \\ \Downarrow \\ winBid = win(getPrice(\mathbb{L})) \end{gathered}$$

Hence we have

$$testov_{2,m} \xrightarrow{*} OK \Rightarrow winBid = win(getPrice(\mathbb{L})) \tag{4.7}$$

In the next step, we generalize the result to n bids, i.e. we want to show that

$$testov_{n,m} \xrightarrow{*} OK \Rightarrow winBid = win(getPrice(\mathbb{L}))$$

Firstly note that in this protocol the highest bid wins, and that in the case of ties the bid with the lowest index in the list wins (cf. Listing 4.9 on page 209), i.e. we have

$$\begin{gathered} winBid = win(getPrice(\mathbb{L})) \\ \Updownarrow \\ \forall 1 \leq i \leq n : (getPrice(\mathbb{L}[i]) \leq getPrice(\mathbb{L}[winBid])) \wedge \\ (getPrice(\mathbb{L}[i]) = getPrice(\mathbb{L}[winBid]) \Rightarrow i > winBid) \\ \Updownarrow \\ (\forall i < winBid : win(getPrice(\mathbb{L}[i], \mathbb{L}[winBid])) = 2) \wedge \\ (\forall i > winBid : win(getPrice(\mathbb{L}[winBid], \mathbb{L}[i])) = 1) \end{gathered} \tag{4.8}$$

Let $k = winBid$. We can see from the code that

$$\begin{gathered} testov_{n,m} \xrightarrow{*} OK \\ \Downarrow \\ (\forall i < winBid : testov_{2,m} \{^{chOi1}/_{chO11}, ^{chOk1}/_{chO21}, \ldots, \\ ^{chOim}/_{chO1m}, ^{chOkm}/_{chO2m}, ^{two}/_{w}\} \xrightarrow{*} OK) \wedge \\ (\forall i > winBid : testov_{2,m} \{^{chOk1}/_{chO11}, ^{chOi1}/_{chO21}, \ldots, \\ ^{chOkm}/_{chO1m}, ^{chOim}/_{chO2m}, ^{one}/_{w}\} \xrightarrow{*} OK) \end{gathered}$$

i.e. that if $testov_{n,m}$ holds on for the winner input from channel `chW`, prices (i.e. public keys) `pk1` to `pkm`, decryptions and proofs from channels `chO11` to `chOnm`, then for any $i < winBid$

$$testov_{2,m} \{^{chOi1}/_{chO11}, ^{chOk1}/_{chO21}, \ldots, ^{chOim}/_{chO1m}, ^{chOkm}/_{chO2m}, ^{two}/_{w}\}$$

holds for winner `two` and considering only the decryptions and proofs from channels `chOi1` and `chOk1` to `chOim` and `chOkm`, and analogously for $i > winBid$. Using the intermediate result (4.7) we have

$$\begin{gathered} (\forall i < winBid : win(getPrice(\mathbb{L}[i], \mathbb{L}[winBid])) = 2) \wedge \\ (\forall i > winBid : win(getPrice(\mathbb{L}[winBid], \mathbb{L}[i])) = 1) \end{gathered}$$

and using (4.8) we can conclude

$$winBid = win(getPrice(\mathbb{L}))$$

We now make a similar argument for completeness, starting again by proving soundness for 2 bids and m prices.

$$\begin{gathered} winBid = win(getPrice(\mathbb{L})) \\ \Downarrow \\ \forall 1 \leq i \leq m : getPrice(\mathbb{L}[winBid]) = i \Rightarrow winBid = win(getPrice(\mathbb{L})) \\ \Downarrow \\ \forall 1 \leq i \leq m : testov_{2,i} \{^{pki}/_{pk1}, ^{chO1i}/_{chO11}, ^{chO2i}/_{chO21}, ^{i}/_{one}\} \xrightarrow{*} OK \\ \Downarrow \\ testov_{2,m} \xrightarrow{*} OK \end{gathered}$$

We conclude for n bids as follows.

$$
\begin{gathered}
winBid = win(getPrice(\mathbb{L})) \\
\Downarrow \\
(\forall i < winBid : win(getPrice(\mathbb{L}[i], \mathbb{L}[winBid])) = 2) \wedge \\
(\forall i > winBid : win(getPrice(\mathbb{L}[winBid], \mathbb{L}[i])) = 1) \\
\Downarrow \\
(\forall i < winBid : testov_{2,m}\, \{^{chOi1}/_{chO11}, ^{chOk1}/_{chO21}, \ldots, \\
^{chOim}/_{chO1m}, ^{chOkm}/_{chO2m}, ^{two}/_{w}\} \xrightarrow{*} OK) \wedge \\
(\forall i > winBid : testov_{2,m}\, \{^{chOk1}/_{chO11}, ^{chOi1}/_{chO21}, \ldots, \\
^{chOkm}/_{chO1m}, ^{chOim}/_{chO2m}, ^{one}/_{w}\} \xrightarrow{*} OK) \\
\Downarrow \\
testov_{n,m} \xrightarrow{*} OK
\end{gathered}
$$

□

We thus conclude that all proposed tests are sound and complete, hence the protocol by Sako [Sak00] is *verifiable*. The code used is available on our website [Dre13].

Verifiability in the computational setting. Next, we analyze both soundness and completeness of the verifiability tests in the computational setting using CryptoVerif [Bla06a]. CryptoVerif has two input modes: the mode "*channels*" and "*oracles*". The mode "channels" takes process descriptions and queries similar to ProVerif. These are then translated into "oracles", and CryptoVerif tries to prove the queries using a sequence of games. This sequence of games can be found automatically in simple cases, or manually using an interactive front-end. Each transition can be a simplification of the current game, the application of an equation or of a cryptographic hypotheses such as *IND-CPA* (indistinguishability under chosen plaintext attacks [BDPR98]) for public key encryption or *UF-CMA* (existential unforgeability against adaptive chosen message attacks [GMR88]) for signatures. For more details on CryptoVerif, we refer to the original paper [Bla06a] or the CryptoVerif Manual [BC12].

Due to the similarities with ProVerif, we employ the mode "channels". We do not repeat the model and tests, but directly discuss the verification games. However, as we are not in a symbolic setting any more, we need to slightly adapt our model. We replace the abstract primitive "proof of correct decryption" with the implementation originally proposed by Sako: the authority simply reveals the secret key to enable verification of the decryption. Again, the entire code used is available on our website [Dre13].

For the first test, rv_s, we assume a *UF-CMA* probabilistic signature scheme. In Listing 4.18 on the next page we present our CryptoVerif code which works as follows: up to N bidders post their bids on the bulletin board (the choice of the

price is left to the adversary), and then the adversary can submit bids to the test. Note in here we model the test for N bidders by allowing the adversary to submit N messages, and that he wins if he can reach the event `accept(m)` at least once without using one of the honest bids.

CryptoVerif is able to conclude automatically that the test is sound, i.e. it will only accept registered bidders: each event `accept(m)` is preceded by an event `send(m)`. For N bidders the advantage computed by CryptoVerif is $N * adv_{sign}$, where adv_{sign} is the advantage against the signature scheme.

```
1  let bidder =
2    in(chBid,b:pkey);
3    new ks:skeyseed;
4    insert keys(spkgen(ks));
5    let k = sskgen(ks) in
6    new s:seed;
7    let offer:sblocksize = bitToSign(enc(bidval,b,s)) in
8    new ss:sseed;
9    let offers = sign(offer,k,ss) in
10   event send(offer);
11   out(chBB1,(offer,offers)).
12
13 let testrvs =
14   in(chBB2,(m:sblocksize,s:signature));
15   get keys(k) suchthat check(m,k,s) in
16     event accept(m).
17
18 process
19   ((!i<=N testrvs) |
20    (!i<=N bidder))
```

Listing 4.18 – The CryptoVerif code for checking soundness of rv_s.

Similarly, in Listing 4.19 on the facing page, we describe our CryptoVerif code for completeness of rv_s: A honest bidder creates his keys, registers himself and submits a bid to the test. To avoid false attacks, we use "inlining" again, i.e. we merge the bidder's process with the test process. CryptoVerif concludes automatically that the event `reject` is unreachable, i.e. that such bids are always accepted. Here we need no cryptographic hypothesis, only the correctness of the signature scheme, i.e. that

$$check(m, pkgen(r), sign(m, skgen(r), r2)) = true$$

holds for any message m and any seeds r and $r2$. The computed advantage is 0, but this only means that – since the signature scheme is correct – the bid will always be accepted.

```
let testcomp =
  in(chAd,b:pkey);
  new k:skeyseed;
  let spk = spkgen(k) in
  let ssk = sskgen(k) in
  insert keys(spk);
  new s:seed;
  new s1:sseed;
  let offer:sblocksize = bitToSign(enc(bidval,b,s)) in
  if check(offer,spk,sign(offer,ssk,s1)) <> true then
      event reject().

process
  !i<=N testcomp
```

Listing 4.19 – The CryptoVerif code for checking completeness of rv_s.

CryptoVerif immediately finishes the soundness and completeness proofs for test rv_w (Listings 4.20 and 4.21 on the following page, respectively). For soundness we prove that each event `accept(m,s)` is preceded by an event `onBB(m,s)`. For completeness we show that the event `reject` is unreachable. Again, we connect the processes directly to not rely on channels in order to avoid false attacks. We note that CryptoVerif does not rely on any cryptographic hypothesis for proving soundness or completeness of this test, hence the adversary has an advantage of 0 in both cases.

```
let bulletinboard =
  in(chBB,(b:sblocksize,s:signature));
  event onBB(b,s);
  insert bids(b,s).

let testrvw =
  in(chW,(m:sblocksize,s:signature));
  get bids(=m,=s) in
    event accept(m,s).

process
  ((testrvw) |
   (!i<=N bulletinboard))
```

Listing 4.20 – The CryptoVerif code for checking soundness of rv_w.

For the output verification tests $ov_{\{l,w,s\}}$, again no cryptographic assumptions are needed. For completeness we only require correctness of the encryption scheme, i.e. that

$$dec(enc(m, pkgen(r), r2), r) = m \tag{4.9}$$

```
let bidder =
  new k1s:skeyseed;
  insert keys(spkgen(k1s));
  let k1 = sskgen(k1s) in
  new s1:seed;
  let offer1:sblocksize =
   bitToSign(enc(bidval,b1,s1)) in
  new ss1:sseed;
  let offer1s = sign(offer1,k1,ss1) in
  insert offers(offer1);
  event send(offer1,offer1s);
  authority.

let authority =
  get keys(k1) suchthat check(offer1,k1,offer1s) in
  let dec11 = dec(signToBit(offer1),p1) in
  if(dec11 = injbot(bidval)) then
    let m = offer1 in testrvw
  else
    let dec21 = dec(signToBit(offer1),p2) in
    if(dec21 = injbot(bidval))
      then let m = offer1 in testrvw.

let testrvw =
  get offers(=m) in yield else
    event reject().

process
 in(start,());
 new p1s:keyseed; new p2s:keyseed;
 let p1 = skgen(p1s) in
 let p2 = skgen(p2s) in
 let p1p = pkgen(p1s) in
 let p2p = pkgen(p2s) in
 let b1 = p1p in
 bidder
```

Listing 4.21 – The CryptoVerif code for checking completeness of rv_w.

holds for any message m, secret key r and seed $r2$. For soundness we also need two properties about the keys:

— we assume that there is a function `pkgen` which computes the corresponding public key to a given secret key, and

— if several different secret keys share the same public key, all decryptions (i.e. using any of the private keys sharing the same public key) give the same result.

This is in particular the case for the ElGamal encryption scheme used in the implementation proposed by Sako: given a secret key x one can compute the public key $y = g^x \mod n$, and any secret key x' with $g^{x'} = g^x \mod n$ decrypts correctly. Completeness can again be verified automatically for the base case with two bidders and one price[13], however we were unable to prove soundness, even in CryptoVerif's manual mode. This is due to the heuristic employed in CryptoVerif's handling of equations, prohibiting the application of equations where both sides are complex terms, i.e. not a simple variable [Bla13]. Hence, we prove Theorem 71 manually.

Theorem 71 *The test for $ov_{\{l,w,s\}}$ (see Listing 4.22 on page 237) is sound with advantage 0 if the following conditions hold:*

— *we assume that there is a function* `pkgen` *which computes the corresponding public key to a given secret key, and*

— $\texttt{dec}(\texttt{enc}(m, \texttt{pkgen}(r), r2), r) = m$ *holds for any message m, secret key r and seed $r2$*

— *if several different secret keys share the same public key, all decryptions (i.e. using any of the private keys sharing the same public key) give the same result: if* $\texttt{pkgen}(r) = \texttt{pkgen}(r')$ *then* $\texttt{dec}(\texttt{enc}(m, \texttt{pkgen}(r), r2), r) = \texttt{dec}(\texttt{enc}(m, \texttt{pkgen}(r), r2), r')$

Proof Consider the test in Listing 4.22 on page 237. We have to show that it is sound, i.e. that it will only accept correct outcomes. Similar to the symbolic case above, we show that when the test accepts the provided data, this corresponds to a correct outcome. More precisely, we show that if a bid is accepted as a winning bid, it actually decrypts to the constant under the private key corresponding to the claimed price, and that none of the other bids decrypts under any of the keys corresponding to higher prices.

Consider the first acceptance in line 9. By the test `pk1=pkgen(p1)` in line 3 we know that `p1` is a correct secret key for the published public key `pk1`, i.e. by the hypotheses decrypts to the same value as the original secret key (the key used to generate the public key used do encrypt). Note that the test in line 3 is

[13] Similar to the symbolic proof we can show that this extends to the general case using an analog proof.

important, otherwise an attacker could submit a wrong secret key to claim that no ciphertext decrypts to `bidval`. Hence the following tests to check that the decryptions are correct (lines 3-4) use a correct key. Thus, the tests following in lines 5-7 operate on correctly decrypted values, and only accept if the announced winner corresponds to the observed (correct by the previous tests) decryptions.

If none of the decryptions gives the constant `bidval`, i.e. no bidder chose the price corresponding to the key `pk1`, the same procedure is repeated for the next key. Finally, if even for the last key no decryption returns the constant, the test stills as there was apparently no valid bid for any of the prices – this might correspond to bidders spoiling their bids on purpose.

Finally, the adversary's advantage is 0, as none of the used hypothesis gives the adversary any advantage. □

Remark In Section 4.3.5.3 we discussed the attack by Abdalla et al. [ABN10] on fairness. It turns out that this attack can be discovered if the verification is carried out as specified in Listing 4.22 on the next page.

Recall the attack: the dishonest bidder uses the randomness 0 to encrypt his bid, hence the resulting ciphertext has the form $(1, bidval)$, and decrypts correctly under any key to the value *bidval*. This allows a colluding authority to open the bogus bid at a price of his convenience, for example at price $x_m + 1$ if x_m is the value of the highest bid.

This however is not possible if the verification is carried out correctly. If the authority decides to open the bogus bid at price $x_m + 1$, he has to prove that the bogus bid does not decrypt to the value *bidval* for any key corresponding to a price higher than $x_m + 1$. Since the bogus bid decrypts under any key, he is unable to do so. The only possible way to pass the verification test is to open the bogus bid at the highest possible price and to declare the dishonest bidder a winner at this price, since then no other keys are used in the verification. This however gives the bidder no advantage over simply submitting a bid for the highest price using a correct encryption, hence the attack does not compromise fairness if the verification is carried out correctly.

Note however that our relatively abstract modeling of the encryption scheme and its correctness is not able to identify the underlying problem of using a particular random seed to construct an illegal ciphertext: equation (4.9) states nothing about the result of a decryption using a secret key that does not correspond to the public key used for encryption. To detect this issue, we would need to explicitly model the ElGamal encryption and its mathematical properties.

§ 4.4.3.2. Protocol by Curtis et al.. Our second case study is the protocol by Curtis et al. [CPS07] from Section 4.3.5.1.

```
let testov =
 in(chW,((m,s,w,p),(pk1,...,pkm),p1,(c1,m1),...,(cn,mn)));
 if pk1 = pkgen(p1) && dec(c1,p1) = m1 && ... &&
  dec(cn,p1) = mn then
    if p = one && ((w = one && m1 = bidval) || ... ||
     (w = n && m1 <> bidval && ... && m(n-1) <> bidval &&
     mn = bidval))
      then
       out(chOV, OK)
    else
      if m1 <> bidval && ... && mn <> bidval &&
       p <> one then
        ...
        in(chW,(pm,(c1,m1),...,(cn,mn));
        if pkm = pkgen(pm) && dec(c1,pm) = m1 && ... &&
         dec(cn,pm) = mn then
          if p = m && ((w = one && m1 = bidval)
           || ... || (w = n && m1 <> bidval && ... &&
           m(n-1) <> bidval && mn = bidval)) then
            out(chOV, OK)
          else
           if m1 <> bidval && ... && mn <> bidval &&
            p <> m then
             out(chOV, OK)
           else
             out(chOV, KO)
        else
          out(chOV, KO)
      ...
 else
   out(chOV, KO).
```

Listing 4.22 – The soundness and completeness code for $ov_{\{l,w,s\}}$.

```
let testrvw(chP1:channel,...,chPn:channel,chW:channel,
            chRVW:channel) =
  in(chP1,(p1:bitstring,c1:bitstring,h1:bitstring));
  ...
  in(chPn,(pn:bitstring,cn:bitstring,hn:bitstring));
  in(chW,(p:bitstring));
  if (p1 = p) || ... || (pn = p) then
    out(chRVW,OK)
  else
    out(chRVW,KO).
```

Listing 4.23 – The test rv_w.

```
let testrvw(chP1:channel,...,chPn:channel,chW:channel,
            chRVW:channel) =
  in(chP1,(p1:bitstring,c1:bitstring,h1:bitstring));
  event onList(p1);
  ...
  in(chPn,(pn:bitstring,cn:bitstring,hn:bitstring));
  event onList(pn);
  in(chW,(p:bitstring));
  if (p1 = p) || ... || (pn = p) then
    out(chRVW,OK);
    event accept(p)
  else
    out(chRVW,KO);
    event reject().
```

Listing 4.24 – The test rv_w with events.

With respect to our model of verifiability, we have the set of bidders $\mathcal{B}$ and a seller S. We do not need to specify the type of bids *Bid* since the protocol supports any type of bids. The bids are published when the auctioneer reveals the symmetric key, i.e. $\mathbb{L}$ contains bids of the following type: $(Pseudo \times PEnc(Bid) \times Hash)$, where *Pseudo* is the type of pseudonyms, *PEnc* is a public-key encryption and *Hash* are hash values. The function *getPrice* simply decrypts the encrypted bid (the second entry of the tuple). The function *isReg* returns *true* if and only if the hash value is correct, the pseudonym was actually attributed by the RA and the bid was submitted correctly signed by the bidder with this pseudonym. The protocol is independent of the used auction mechanism and hence does not define *win*. The seller simply decrypts all bids and can then apply any function *win*. He publishes the winning price and the winning bidders pseudonym, and *winBid* denotes the index of the bid containing this pseudonym.

As a concrete instance of this abstract model, we employ the model in the Applied π-Calculus already used in Section 4.3.5.1.

```
let testrvs(k1:pkey,...,kn:pkey,
            chH1:channel,...,chHn:channel,
            chP1:channel,...,chPn:channel,chRVS:channel) =
  in(chH1,(h1:bitstring, s1:bitstring));
  ...
  in(chHn,(hn:bitstring,sn:bitstring));
  in(chP1,(p1:bitstring,c1:bitstring,h1:bitstring));
  ...
  in(chPn,(pn:bitstring,cn:bitstring,hn:bitstring));
  if (h1 = ha && checksign(s1,k1) = h1) && ... &&
    (hn = ha && checksign(sn,kn) = hn) then
      out(chRVS,OK)
  else
      out(chRVS,KO).
```

Listing 4.25 – The test rv_s.

```
let testrvs(k1:pkey,...,kn:pkey,
            chH1:channel,...,chHn:channel,
            chP1:channel,...,chPn:channel,chRVS:channel) =
  in(chH1,(h1:bitstring, s1:bitstring));
  ...
  in(chHn,(hn:bitstring,sn:bitstring));
  in(chP1,(p1:bitstring,c1:bitstring,h1':bitstring));
  ...
  in(chPn,(pn:bitstring,cn:bitstring,hn':bitstring));
  if (h1 = h1' && checksign(s1,k1) = h1) && ... &&
     (hn = hn' && checksign(sn,kn) = hn) then
      out(chRVS,OK);
      event accept(h1,s1,p1,c1,...,hn,sn,pn,cn)
  else
      out(chRVS,KO);
      event reject().
```

Listing 4.26 – The test rv_s with events.

Analysis. As already noted in Section 4.3.5.1, the protocol does not specify the output computation, hence we do not consider the tests ov_s, ov_l and ov_w.

For rv_w we propose the following test: checking if the pseudonym of the winner appears in the list of bids. Using ProVerif we were able to show that this test (Listing 4.23 on page 238) is sound and complete, using the same techniques as in the case of the protocol by Sako: we instrument the process with events (Listing 4.24 on page 238), and show that any `accept(p)` is preceded by a corresponding `onList(p)`. Similarly we show that if a honest seller announces one of the bidders as the winner, the event `reject` is unreachable[14]. In both cases we only consider the case with one bidder, however we can show that this generalizes to an arbitrary number of bidders similarly to Theorem 69 on page 222.

For rv_s the situation is more complicated. Essentially, a verifier can only check if the hash was correctly signed when previously submitted to the RA as all other data is encrypted or otherwise inaccessible. We implemented such a test in ProVerif (Listing 4.25 on the previous page) and were able to prove its completeness using the same technique as above: we show that the event `reject` is not reachable if all parties are honest. To check soundness we verify if each event `accept` is preceded by the corresponding `send`, where send is an event executed by the bidders when submitting the bid. ProVerif immediately finds an attack: since the messages from the RA to the seller are not authenticated, anybody can modify all values except for the hash, in particular the pseudonyms. Hence there is no assurance that a pseudonym actually corresponds to a bidder. Even if the messages from the RA were authenticated, a verifier still needs to trust the RA, since the RA can still manipulate the pseudonyms, for example swap two pseudonyms or create fake ones. Note also that if we want to ensure privacy of the bids the first message containing the signed hash also needs to be encrypted (cf. Section 4.3.5.1), hence even this insufficient check is not possible any more.

Curtis et al. explicitly state that the RA needs to be trusted for correctness. However, they also propose a property "robustness" which states that an auction protocol should be able to handle corrupt behavior. Clearly, their protocol is not robust to dishonest behavior by the RA, and this cannot be fixed using verification checks, as they also would require trust in the RA. We argue that basing verifiability tests on such a trust assumption at least partly contradicts the main point of verifiability, which is to back up trust assumptions by providing evidence that a trusted party actually behaved honestly.

Again, the code used is available on our website [Dre13].

[14] As we consider only one bidder, we do not need to know how the winner is determined, we simply assume that the one participating bidder will be declared the winner.

4.4.4 — Summary

In this section we defined verifiability for auction protocols using a high-level model that can be instantiated with different concrete model, including symbolic and computational ones. Our definitions cover first-price auctions as well as second-price or even bulk good auctions.

We applied our definitions on two case studies. Our first example, the protocol by Sako, ensures verifiability as it provides sound and complete verification test. We showed this in a symbolic setting using ProVerif and in a computational setting using CryptoVerif. The protocol by Curtis et al. is our second case study and suffers from several problems. For example the bids lack authentication in the last step of the protocol, and one needs to trust the registration authority to manage the pseudonyms correctly.

4.5 Towards True Bidder-Verifiable Auctions

In the previous section we defined and analyzed verifiability. As discussed in the introduction, the goal of verifiability is to enable the participants to verify the correct execution of the protocol in order to reduce the necessary trust hypothesis. We were able to show that the protocol by Sako, one of our two case studies, fulfills our definition of verifiability. However, to achieve this, it relies on cryptographic operations such as signatures and public-key cryptography.

Yet, achieving verifiability through cryptography is somewhat disingenuous: any participant lacking cryptographic expertise cannot ascertain for himself that the verification procedure is indeed correct, and is thus forced to trust the judgment of cryptographic experts that designed this procedure. This means that to some extent the verification procedure does not reduce the necessary trust, but simply shifts the trust from the parties participating in the protocol execution to the cryptographers that designed the protocol.

The same problem was already identified in electronic voting. In 2004, Chaum [Cha04] argued along the same lines that the verifiable cryptographic voting protocols proposed up to this date did little to empower actual voters to verify elections, as even the verification procedures were to complex for a normal voter to understand. This view was shared by the German Constitutional Court in its decision on electronic voting machines: “the use of electronic voting machines requires that the essential steps of the voting and of the determination of the result can be examined by the citizen reliably and *without any specialist knowledge of the subject*” [Bun09]. To address this issue, Chaum proposed a voting protocol using visual cryptography: the ballot was distributed over two layers that on top of each other showed the voter’s choice. Then, one layer was destroyed, leaving the voter with a layer full of random dots from which no choice can be inferred. However,

anyone can verify that the system accurately recorded this layer – *without* any cryptographic expertise. Chaum called this "true voter-verifiable elections", hence our idea of "true bidder-verifiable auctions".

In this section we explore this idea of "true bidder-verifiable auctions" based on physical properties. We propose two examples of protocols ensuring "true bidder-verifiability". Although these protocols have some limitations with respect to scalability, they illustrate how we can realize secure auctions by exclusively relying on physical objects and their properties. Since we already established our formal security definitions for auctions protocols (Fairness, Authentication, Privacy and Verifiability), we also propose a first approach allowing to model physical objects and their properties in the Applied π-Calculus, in an attempt to apply our previous definitions on the new physical protocols.

Inspired by Sako's protocol, we propose a physical protocol called *Cardako*[15]. This protocol does not rely on cryptography nor on trusted parties, yet retains the verifiability, privacy, authentication and fairness properties of Sako's protocol. We provide a formal analysis of these security properties in ProVerif, modeling their physical properties using a special equational theory. Although ensuring privacy for the losing bidders (if the authorities are trusted), both the Sako and the Cardako protocol publicly reveal the winner.

To improve privacy, we propose *Woodako*[16]: a physical auction protocol that only reveals the winner and winning price to the seller and the winner, but not to the losing bidders (similar to [Bra06]). However, we show that the result remains verifiable even for losing bidders. We build a concrete prototype of this protocol, and verify the security properties with the help of ProVerif.

We argue that both protocols can be understood by a non-expert in the spirit of "true bidder-verifiable auctions".

4.5.1 — The "Cardako" Protocol

The Cardako protocol is a practical implementation of the main concepts of Sako's protocol [Sak00] using office material.

§ 4.5.1.1. Description. In the Cardako protocol each bidder has a piece of cardboard marked with a number of positions corresponding to possible prices in descending order from left to right (see Figure 4.4 on the facing page). A bidder chooses his price (in Figure 4.4, the second-highest price p_{m-1}), and punches a hole in the position corresponding to his bid. Next, he inserts the card into a special envelope of the same size with marks corresponding to the different prices on the outside (see Figure 4.5 on the next page), and signs on the outside of the

[15] A **Card**board version of S**ako**'s protocol.
[16] **Wood**en box based implementation of S**ako**'s protocol.

envelope. The envelope is sealed and shown to all other bidders so that they can check the signature.

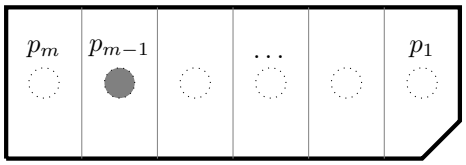

Figure 4.4 – Cardako bid for price p_{m-1}.

Once all bidders have finished creating bids and shown their signatures, the bidders swap envelopes and jointly try to punch a needle through the marked areas of each envelope, starting with the highest possible price. If this succeeds, they find a hole and hence a bid for this price. The signature on the outside then allows the identification of the winner. If this fails for all bids, the bidders repeat the procedure for the second price, etc.. Since the order of the prices is important, the cards and envelopes are designed such that they can only fit together in one way. This ensures that the card cannot accidentally or maliciously be turned around.

To fully ensure verifiability, the protocol must also ensure that only eligible bidders can bid. This is achieved through the verification of the signatures on the envelopes by the seller and bidders when bids are posted.

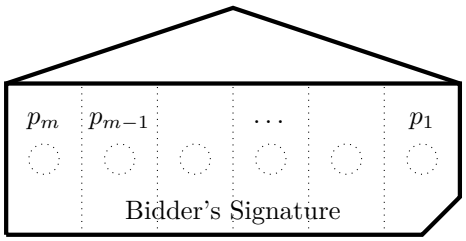

Figure 4.5 – Cardako bidding envelope.

§ 4.5.1.2. Security Properties. The Cardako protocol relies on the physical properties of the cardboard and the envelopes: Nobody can see from the outside the contents of a bid, but by trying to punch with a needle the bidders can test if it was an offer for a certain price. It offers verifiability similar to Sako's protocol as well as non-cancellation and non-repudiation due to the signatures and the swapping of the envelopes. It ensures fairness since no premature information is leaked (*Weak Non-Interference*) and due to the joint bid opening no cheating is possible (*Highest Price Wins*).

Obviously a malicious bidder can open an envelope and read its contents – but this is actually similar to Sako's protocol, where dishonest authorities can

break privacy. The difference is that in Cardako such a behavior will be detected by the other bidders, since the envelope is damaged and the other parties are in the same room. An extension to improve privacy could be to put the signed envelopes into slightly bigger and indistinguishable envelopes after the signature has been verified by the other parties. These envelopes can be posted into a ballot box to break the link between a bidder and his bid. Hence a malicious bidder will only be able to break the privacy of a random bid, but not necessary the one he is interested in.

A positive side effect of using indistinguishable envelopes is that ties can be detected and resolved fairly without revealing the identities of the tied bidders (unlike Sako's protocol). This happens as follows: first, the protocol determines which envelopes contain winning bids. As the envelopes are indistinguishable from one another, the identity of the tied bidders is not revealed yet. The tie is then broken by selecting a random envelope (e.g. by rolling a die, or drawing an envelope from a hat).

§ 4.5.1.3. Formal Analysis. We model the bidders as processes exchanging messages (envelopes or real communication messages), however we also need to model the physical properties of the objects used. Our approach consists in modeling the properties using an equational theory. We describe the envelope using a function `envelope` that is created using a random seed to hide its contents and can only be opened using that seed, similar to a cryptographic commitment. However we also have functions `testone`, `testtwo`, ..., `testm`, that can test for a certain value without opening the envelope (i.e., the needle tests[17]):

$$\begin{aligned}
\texttt{open}(\texttt{envelope}(\mathit{content}, k), k) &= \mathit{content} \\
\texttt{testone}(\texttt{sign}(\texttt{envelope}(x, k), sk)) &= \begin{cases} \texttt{true} & \text{if } x = \texttt{one}, \\ \texttt{false} & \text{otherwise} \end{cases} \\
\texttt{testtwo}(\texttt{sign}(\texttt{envelope}(x, k), sk)) &= \begin{cases} \texttt{true} & \text{if } x = \texttt{two}, \\ \texttt{false} & \text{otherwise} \end{cases} \\
&\vdots \\
\texttt{testm}(\texttt{sign}(\texttt{envelope}(x, k), sk)) &= \begin{cases} \texttt{true} & \text{if } x = \texttt{m}, \\ \texttt{false} & \text{otherwise} \end{cases} \\
\texttt{checksign}(\texttt{sign}(m, k), \texttt{pk}(k)) &= m \\
\texttt{getmessage}(\texttt{sign}(m, k)) &= m \\
\texttt{getpubkey}(\texttt{sign}(m, k)) &= \texttt{pk}(k)
\end{aligned}$$

[17] Note that technically the needle tests cannot be modeled like this in a normal equational theory due to the case distinction, however we can implement this in ProVerif using extended destructors. A variant for a normal equational theory would be $\texttt{testone}(\texttt{sign}(\texttt{envelope}(\texttt{one}, k), sk)) = \texttt{true}$.

```
let bidder(b:bid,sk:skey,chBox:channel,chBoxOut:channel,
           chSyncS1:channel,...,chSyncSn:channel,
           chSyncR:channel,chWon:channel) =
  new k:key;
  let offer:letter = envelope(b,k) in
  let sb:signedletter = sign(offer,sk) in
  event bid(sb,k);
  out(chBox,(sb,k));
  in(chBoxOut,(env:signedletter,s:key));
  if (testone(env) = true) then
    let sb:bid = open(getmessage(env),s) in
    event won(env,s);
    out(chWon,(sb,getpubkey(env)));
    out(chSyncS1,true);
    ...
    out(chSyncSn,true)
  else
    out(chSyncS1,false);
    ...
    out(chSyncSn,false);
    in(chSyncR,found1:bool);
    ...
    in(chSyncR,foundn:bool);
    if (found1 = false) && ... &&
     (foundn = false) then
      ...
      if (testn(env) = true) then
        let sb:bid = open(getmessage(env),s) in
        event won(env,s);
        out(chWon,(sb,getpubkey(env)));
        out(chSyncS1,true);
        ...
        out(chSyncSn,true)
      else
        out(chSyncS1,false);
        ...
        out(chSyncSn,false);
        in(chSyncR,found:bool).
```

Listing 4.27 – The bidder.

```
let processSwap(chBoxIn1:channel,...,chBoxInN:channel,
                chBoxOut1:channel,...,chBoxOutN:channel) =
  in(chBoxIn1,(b1:signedletter,k1:key));
  event recBid(b1,k1);
  ...
  in(chBoxInN,(bn:signedletter,kn:key));
  event recBid(bn,kn);
  out(chBoxOut1,(bn,kn)) |
  ...
  out(chBoxOutN,(b1,k1)).
```

Listing 4.28 – The swap process.

```
process
  new k1:skey; new k2:skey;
  new chBoxIn1:channel; new chBoxIn2:channel;
  let pk1 = pk(k1) in let pk2 = pk(k2) in
  out(chPKI,(pk1,pk2));
  processSwap(chBoxIn1,chBoxIn2,chBoxOut1,chBoxOut2) |
  bidder(b1,k1,chBoxIn1,chBoxOut1,chSync1,chSync2,chWon) |
  bidder(b2,k2,chBoxIn2,chBoxOut2,chSync2,chSync1,chWon)
```

Listing 4.29 – An instance with two bidders.

The last three equations model signatures. The first equation allows to verify a signature, the second equation to obtain the signed message, and the last one to identify the person who signed the message.

The honest participants obviously only test for the values that they are supposed to, but dishonest participants can also apply tests they are not supposed to execute. To model the fact that any party in possession of an envelope can open it, the bidders give away the random seed they used to create it when giving away the envelope.

The process model is given in Listings 4.27 on the preceding page, 4.28 and 4.29. The bidder (Listing 4.27 on the previous page) encloses his bid inside the envelope and sends the envelope together with the key to the synchronization and swap process (Listing 4.28). This process waits until he has all bids, and then redistributes them in inverse order over all bidders. This is to model that once all envelopes are ready, they are swapped between the bidders. The bidders then receive the envelopes, and test for the first (i.e. highest) price. If this succeeds, they announce the winner and stop. If it fails, they let the other bidders know, and wait if any of the other bidders finds a winner. If yes, they stop, otherwise they test for the next possible price, and so on. The verification tests (Listings 4.30 on the next page, 4.31 on the facing page and 4.32 on page 248) are straightforward: To check the result, one repeats the necessary needle test, and

```
let testov(chWin:channel,chOV:channel,
           chBid1:channel,...,chBidn:channel) =
  in(chWin,(price:bid,id:pskey));
  in(chBid1,(b1:signedletter,s1:key));
  ...
  in(chBidn,(bn:signedletter,sn:key));
  if ((price = one) && (getpubkey(b1) = id) &&
   (testone(b1)) then
    out(chOV,OK)
  ...
  else if (price = one) && (getpubkey(bn) = id) &&
   (testone(bn)) then
    out(chOV,OK)
  ...
  else if (price = m) && (getpubkey(b1) = id) &&
   (testtwo(b1)) && not(testone(b1)) && ... &&
   not(testone(b2)) then
    out(chOV,OK)
  ...
  else if (price = m) && (getpubkey(bn) = id) &&
   (testtwo(b2)) && not(testone(b1)) && ... &&
   not(testone(b2)) then
    out(chOV,OK)
  else
    out(chOV,KO).
```

Listing 4.30 – The verification test for $ov_{\{l,w,s\}}$.

```
let testrvs(chRVS:channel,pk1:pskey,...,pkn:pskey,
            chBoxIn1:channel,...,chBoxInN:channel) =
  in(chBoxIn1,(b1:signedletter,k1:key));
  ...
  in(chBoxInN,(bn:signedletter,kn:key));
  if checksign(b1,pk1) = getmessage(b1) && ... &&
   checksign(bn,pkn) = getmessage(bn) then
    out(chRVS,OK)
  else
    out(chRVS,KO).
```

Listing 4.31 – The verification test for rv_s.

```
let testrvw(chRVW:channel,chW:channel,
            chBox1:channel,...,chBoxN:channel) =
 in(chBox1,(b1:signedletter,k1:key));
 ...
 in(chBoxN,(bn:signedletter,kn:key));
 in(chW,(b:bid,id:pskey));
 if (open(getmessage(b1),k1) = b && getpubkey(b1) = id)
  || ... ||
  (open(getmessage(bn),kn) = b && getpubkey(bn) = id) then
   out(chRVW, OK)
 else
   out(chRVW, KO).
```

Listing 4.32 – The test rv_w.

to check the registration one checks the signatures and if the announced winner is among the initial bids.

This model allows us to repeat the same verification steps as for Sako's protocol[18] and to conclude that the protocol ensures *Non-Repudiation*, *Non-Cancellation*, *Weak Non-Interference*, *Highest Price Wins* and *Verifiability*.

When verifying *Privacy*, ProVerif finds the obvious attacks of opening the envelopes or testing for all values. If however we keep the seeds private and remove all tests except for the required ones (in an attempt to model the fact that bidders do not want to risk being caught when breaking the rules) ProVerif is able to prove secrecy of the losing bids. All the above verifications succeed within a few seconds on a standard office PC.

The full code is available online [Dre13].

4.5.2 — The "Woodako" Protocol

To improve the privacy of the Cardako protocol, we developed *Woodako*, which relies on a special wooden box. Our prototype is designed for 3 bidders and 5 possible prices, but such a box can be built for any number n of bidders and any number k of prices. Figure 4.6 on the facing page shows all components of the box. The Woodako auction system uses:

— Five black marbles per bidder, each size represents one price.

— Six layers ($L0$ – $L5$): Layers $L0$ and $L1$ are made of transparent plexiglass and have no holes. The other layers are made of wood and contain four holes per column. The size of these holes corresponds to the size of the marbles, i.e. the holes in L2 are only big enough for the smallest marbles, the holes in L3 for the second-smallest etc.

[18] As in the case of Sako's protocol, we only check the base case using ProVerif, however we can show that this generalizes to any number of bidders using similar proofs.

— Three top layers $T1$, $T2$, $T3$. Each layer is associated with a bidder.

— Two inclined layers: these are placed below the layers, near the bottom of the box.

— Locks and keys: each bidder and the seller has a set of locks and keys.

— One front side made of wood that closes the box and contains holes to insert the extremities of the layers. These extremities will stick out and so constitute a place where the parties can put locks. These locks are used to ensure security properties regarding that layer, for example that it cannot be removed unless everybody agrees.

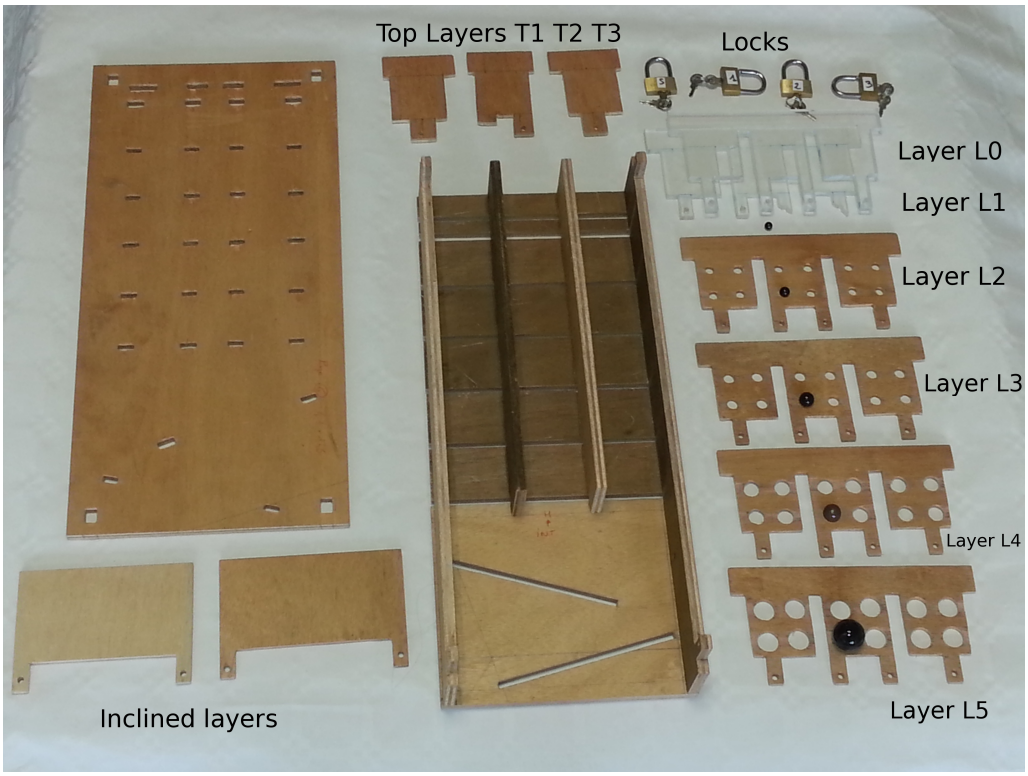

Figure 4.6 – The Woodako prototype.

§ 4.5.2.1. Description. The wooden box carries out the important steps of the auction in a secure way through its physical properties. The box (see Figure 4.7 on the next page) is composed of three columns and seven horizontal plus two inclined layers. Each column (the left, middle and right part of the box) corresponds to one bidder. The top layers $T1$, $T2$ and $T3$ are used to achieve confidentiality of the bid of each bidder, as the marbles (corresponding to the bids) are inserted underneath. The transparent layer $L0$ is used to lock the bids, once they are made, to achieve non-repudiation and non-cancellation. The five lower

Figure 4.7 – Inside our Woodako prototype, where layers $L0$ and $L1$ are removed.

horizontal layers $L1$ – $L5$ are used to determine the winning price in a private way. Finally, the two inclined layers are intended to make it impossible to know from which column a marble fell by guiding all of them to the same spot in the bottom left part of the box.

The general idea is as follows: Each bidder will place his bid, modeled by a marble of a certain size, in the top part of the box. We use five different sizes, the smallest one representing the highest possible price, and the biggest one representing the lowest possible price. In the bidding phase, all marbles are inserted into the box onto solid layer $L0$. In the opening phase, layer $L0$ is removed. Below there is layer $L1$ with holes big enough to only let the smallest marbles pass through. Below layer $L1$, there is layer $L2$ with bigger holes (the size of the next biggest marble), and so on. If a bidder inputs the highest possible price, i.e. the smallest marble, this marble will fall through all layers once the solid layer is removed, hence revealing the winning price – but not the winning bidder, thanks to the inclined layers. If nobody inserted the smallest marble, no marble will fall through and the participants can then remove the next layer to check for the second highest price, and so on.

All layers $L0$ – $L5$ are equipped with four locks, one for each of the three bidders, plus one for the seller. This ensures that a layer can only be removed if all parties agree to do so. Similarly, the removable front side of the box is attached using four locks in the four corners (cf. Figure 4.8 on page 253), one for each bidder plus one for the seller. This allows the parties to inspect the interior of the box before starting the protocol.

The topmost layer consists of three independent parts $T1$, $T2$ and $T3$ that each bidder can use to secure his bid (i.e. his marble inside the box, cf. Figure 4.8 on page 253) from the other participants. Once all bids are inserted, the transparent layer $L0$ is inserted just below and locked by all four parties to ensure non-cancellation (cf. Figure 4.7 on the facing page). Once the winning price is determined, the bidders can open their column by removing their lock on Ti and check through the transparent layer if their part of the box is empty or not, i.e. if they won or not (cf. Figure 4.10 on page 255).

Similarly the seller can remove the two inclined layers at the bottom to check if a marble is present inside a column or not (cf. Figure 4.11 on page 255). The first solid layer $L1$ of the price determination part is transparent to allow the participants to check at the start of the protocol if each bidder inserted exactly one marble. Note also that all participants are always in presence of the box to be able to detect misbehavior of somebody.

The protocol is then broken down into 4 phases:

1) Initialization: Each participant can check all the material and see the inside of the box as in Figure 4.7 on the preceding page to convince himself of the

correct design of the machine. The seller gives black marbles of different sizes to each bidder. The smallest marble corresponds to the biggest price and the biggest marble represents the lowest price. Moreover the seller and each bidder have a set of padlocks and keys (as in Figure 4.8 on the facing page). Once all bidders have checked the box and received their material (in the case of our prototype five marbles and eight padlocks with keys), the seller closes the box with the front side. The seller and each bidder put a padlock on the box (on each corner of Figure 4.9 on page 254, marked with 1, 2, 3, and S). The seller places the layers $L1$ – $L5$ in the box, but neither the individual top layers $T1$, $T2$ and $T3$ nor the transparent layer $L0$. The seller also places the two inclined layers in the bottom of the box. Finally, he puts one lock on each layer on the middle column, and all four locks on the inclined layers. He also assigns a column to each bidder.

2) Bidding Phase: Each bidder selects a marble corresponding to the price he wants to bid and puts it in his column without showing the marble to the other parties. He then closes his column using his top layer Ti and secures it using one of his locks. He also puts locks on the five layers $L1$ – $L5$ below. In Figure 4.8 on the facing page you can see the box after bidder number 2 assigned to the middle column has made his bid. Once all bids are made and all locks in place, the seller introduces the transparent plexiglass layer $L0$, i.e. in the hole between the individual top layers and the first full layer $L1$. Finally each participant puts a lock on plexiglass layer $L0$.

3) Opening Phase: The seller and all bidders verify that each bidder inserted exactly one marble by removing the inclined layers (to which the seller has the keys) and looking through the holes of layers $L2$ – $L5$ and the plexiglass layer $L1$ from below. After the inclined layers have been reinstalled and locked by the seller, all participants remove their lock on the layer $L1$, and the seller removes it. If somebody chose to bid the highest possible price, i.e. insert the smallest possible marble, it will now fall down through all the holes (since all lower layers have bigger holes) and all participants know the winning price, yet not the winner. If no marble falls down, they repeat this process with the next layer below corresponding to the next price. In Figure 4.9 on page 254, we see the back of the box once the two first prices have been tested. The inclined layers are there to hide from which column the marble fell, as all marbles will end up in the bottom left part independently of where they came from (cf. Figure 4.7 on page 250).

4) Verification Phase: Once a marble has fallen down, each bidder can open his lock on his top layer Ti and individually check if his marble is still inside. In Figure 4.10 on page 255, bidder number two notes that his marble is still inside the box, so he did not win. Similarly the seller can remove the two inclined layers and check for each column, whether there is still a marble inside, hence determining the winner (the column with no marble). An example is given in

Figure 4.8 – The Woodako box after the bid of bidder number two.

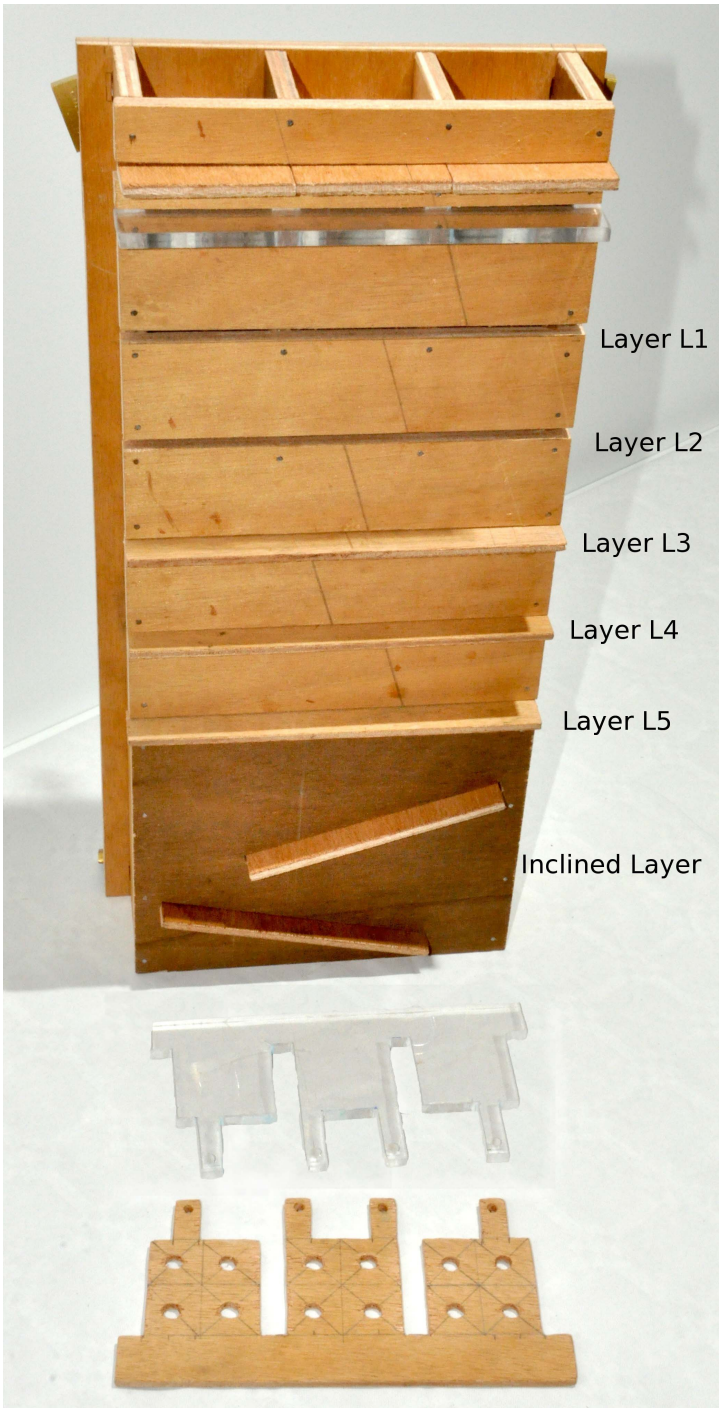

Figure 4.9 – The Woodako box after two prices have been tested.

Figure 4.11: the left bidder won since his column is empty, and the two others lost, as their marbles are still there (highlighted by the yellow circles).

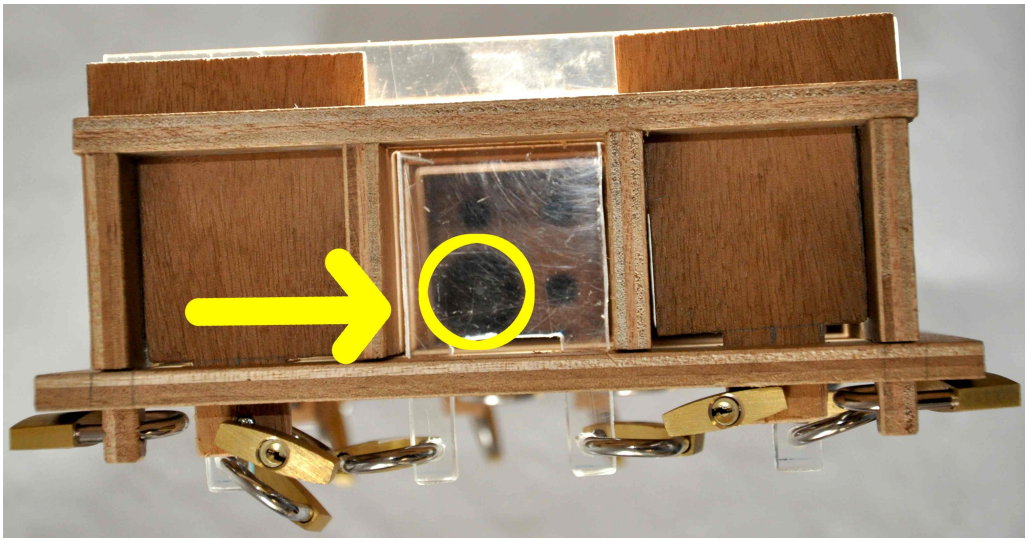

Figure 4.10 – Bidder verifiability (i.e. view from top).

Figure 4.11 – Seller verifiability (i.e. view from bottom).

Resolving ties: Note that in the case of a tie two (or more) marbles will fall down at the same time. Thus everybody knows that there is a tie, the seller can also identify the tied parties, and the bidders know if they are tied or not. Note also that a tied party can prove to anybody that he is tied by opening his top and showing that his compartment is empty. To resolve the situation either an external tie-breaking mechanism can be used (e.g. rolling a die), or the auction can simply be restarted. Using an external mechanism implies revealing the identity of the tied parties or trusting the seller, since he is the only one who knows who is

tied. If privacy is the main concern and the seller is not to be trusted, the auction can simply be restarted and giving the bidders the chance to modify their bids. Note that Sako's protocol (our inspiration) also reveals the identity of the tied parties.

§ 4.5.2.2. Securities Properties. We now argue how the properties defined Sections 4.3 and 4.4 are achieved by our protocol, as long as there is at least one honest party following the protocol (i.e. one bidder or the seller).

Non-cancellation and non-repudiation. Everybody can see in which column a bidder inserted his marble. Due to the fact that the layer $L0$ is locked by all the participants, nobody can change his price during the execution of the protocol. Hence nobody can cancel his bid. Similarly nobody can deny that it was his marble that fell down as the seller and the concerned bidder can verify in which column a marble is still present. Moreover the check at the beginning of the opening phase ensures due to the transparent layer $L1$ that there is exactly one marble per bidder.

Fairness. We consider the two aspects defined in Section 4.3.2:

— *Highest-Price-Wins:* By the design of the box and the holes of different size in layers $L2$ – $L5$, the highest price offered by a bidder which is represented by the smallest marble is the first marble to fall down. No bidder can make a larger marble drop before a smaller one.

— *Weak-Non-Interference:* For a given set of bidders no information about the bids is leaked until the end of the bidding phase, since each bidder can choose his marble privately and drop it into the box in such a way that nobody can identify its size.

Privacy. The winner is only known to the seller and himself, but everybody knows the winning price. The inclined layers prevent anybody else from determining the winner by observing from which column the marble fell[19]. Once a marble has dropped, the winner can check if his column is empty by unlocking his top layer Ti and looking inside. As shown in Figure 4.11 on the preceding page, the seller can also determine the winner by removing the inclined layers

[19]Note that with two layers as shown in Figure 4.7 on page 250 there is a side-channel attack: If the marble falls down in the rightmost column, one can hear the sound of a falling marble only once, whereas in the case of the other two columns the marble falls down twice. However there are some simple solutions: one can extend both layers further to the right so that the marbles fall down twice independently of their original column, or one can use something similar to a "bean machine", i.e. several rows of pins, arranged so that the falling marble hits a pin in each row. The idea is that the marble has a 50% chance of falling down on either side of the pin, hence arrives at a random location on the bottom.

and checking which column is empty. Since the remaining marbles are too big to fall through the holes, the seller can only see if there is a marble, but will be unable to determine its size, as all marbles have the same color. This preserves the secrecy of the losing bids. The losing bidders can also open their top layers Ti and verify if their marbles are still inside as shown in Figure 4.10 on page 255. This leaks no information about the winner, yet they know the price from the moment when the marble falls as each layer corresponds to a price.

Verifiability. The registration is done at the beginning of the protocol by the seller, and all participants can check if only the registered bidders participate by inserting a marble into the box. Hence the protocol ensures registration verifiability. Outcome verifiability is achieved by the fact that each participant can check the box and the mechanism at the beginning of the protocol, and that each bidder can check at the end whether he lost or won by opening his top layer Ti. The seller can also verify the outcome by opening the bottom of the box.

§ 4.5.2.3. Formal Analysis. To formally verify the security properties of this protocol we need to model the properties of the box. Again, we represent the current state of the box (Figure 4.7 on page 250) by an object denoted `machine(·)`. We use an equational theory to model possible changes to it. A `machine(·)` has the following parameters, where the index i represents a bidder among the n bidders, j a price among the m prices, and s the seller:

— b_{ji} representing the different compartments (for each bidder and price, i.e. above $L1$ to $L5$ for bidder one, two and three in our prototype) of the box, which can be empty or contain a marble of a certain size.

— l_{ji} and l_{js} represent the locks (or rather: the keys necessary to open the locks) that need to be opened to remove a layer Lj (where $j \in \{1, \ldots, n\}$) from the "sieve" part of the machine, one for each bidder and one for the seller.

— t_i are the locks used by the bidders to close the top layer Ti after they inserted their bid.

— p_i and p_s are the locks on the plexiglass layer $L0$.

— b_s represents the locks by the seller on the inclined layers at the bottom (for simplicity we model only one instead of four).

— w_k ($k \in \{1, \ldots, n * m\}$) represent the lower left part of the box where the "winning" marbles that have fallen down end up. We need multiple variables since all marbles fall down if all layers are removed. To simplify the equational theory we have different variables for each price, as this allows us to have independent equations for removing each layer – otherwise we need to take the current state of the w_ks into account, which further increases the number

of equations.

We also define the following functions:

— `check_window`: takes as input a machine and returns the w_k to check if a marble has fallen down. This function dose not require any key to be applied.

— `price_j`: takes as input a machine, the keys l_{ji} and l_{js}, and returns a machine where the layer j was removed and potentially marbles have fallen down.

— `open_top_i`: takes as input a machine and the key t_i, and returns the contents of all b_{ji} for bidder j. This corresponds to the bidder verification check by looking through the plexiglass layer $L0$.

— `open_bottom`: takes as input a machine and the key b_s, and returns a vector indicating if the columns contain marbles or not. This corresponds to the seller verification check.

— `change_top_i`: takes as input a machine, all the keys p_k (k among all the bidders), p_s and the single t_i and a new marble to place into bidder i's top compartment.

Consider an example of two bidders and two prices. Suppose the first bidder bids the highest possible price (constant `one`), and the second bidder bids the lower price `two`. Then the initial state of the machine m is:

$$m = \texttt{machine}(\texttt{one}, \texttt{two}, \texttt{empty}, \texttt{empty}, l_{11}, l_{12}, l_{1s}, l_{21}, l_{22}, l_{2s}, \\ t_1, t_2, p_1, p_2, p_s, b_s, \texttt{empty}, \texttt{empty}, \texttt{empty}, \texttt{empty})$$

If we compute $m_1 = \texttt{price\_one}(m, l_{11}, l_{12}, l_{1s})$ we obtain

$$m_1 = \texttt{machine}(\texttt{empty}, \texttt{empty}, \texttt{empty}, \texttt{two}, l_{11}, l_{12}, l_{1s}, l_{21}, l_{22}, \\ l_{2s}, t_1, t_2, p_1, p_2, p_s, b_s, \texttt{one}, \texttt{empty}, \texttt{empty}, \texttt{empty})$$

Any party can apply `check_window` on m_1 to obtain $(\texttt{one}, \texttt{empty}, \texttt{empty}, \texttt{empty})$ and hence observe that a bidder won at price one. The seller can determine that bidder one is the winner by computing $\texttt{open\_bottom}(m_1, b_s) = (\texttt{empty}, \texttt{something})$. Similarly bidder one can check he is the winner by doing $\texttt{open\_top\_1}(m_1, t_1) = (\texttt{empty}, \texttt{empty})$, and bidder two can verify his marble is still in the box with $\texttt{open\_top\_2}(m_1, t_2) = (\texttt{empty}, \texttt{two})$.

Since the number of parameters of the machine depends on the number of bidders and prices, we are unable to define them in a general way in ProVerif. However we have developed a python script that generates the necessary equations for a given number of bidders and prices. For the functions `price_j` and `open_bottom` this also consists in enumerating all possible cases based on the possible bid values.

```
let procMachine(chBox1:channel,...,chBoxN:channel,
 chKey11:channel,...,chKey1n:channel,chKey1s:channel,...,
 chKeym1:channel,...,chKeymn:channel,chKeyms:channel,
 chKeyT1:channel,...,chKeyTn:channel,chKeyP1:channel,...,
 chKeyPn:channel,chKeyPS:channel,chKeyBS:channel,
 chMachine1:channel,...,chMachineN:channel,
 chMachineS:channel,chMachineP:channel)=
  in(chBox1,ball1:ball);
  recBid(ball1,one);
  in(chKey11,l11:key);
  ...
  in(chKeym1,lm1:key);
  in(chKeyT1,t1:key);
  in(chKeyP1,p1:key);
  out(chMachineP,machine(ball1,empty,...,empty,empty,...,
   empty,...,empty,...,empty,l11,none,...,none,...,
   lm1,none,...,none,t1,none,...,none,p1,none,...,none,
   none,empty,...,empty));
  ...
  in(chBoxN,balln:ball);
  recBid(balln,n);
  in(chKey1n,l1n:key);
  ...
  in(chKeymn,lmn:key);
  in(chKeyTn,tn:key);
  in(chKeyPn,pn:key);
  out(chMachineP,machine(ball1,...,balln,empty,...,
   empty,...,empty,...,empty,l11,...,l1n,none,...,
   lm1,...,lmn,none,t1,...,tn,p1,...,pn,none,none,
   empty,...,empty));
  in(chKey1s,l1s:key);
  ...
  in(chKeyms,lms:key);
  in(chKeyPS,ps:key);
  in(chKeyBS,bs:key);
  out(chMachine1,machine(ball1,...,balln,empty,...,
   empty,...,empty,...,empty,l11,...,l1n,l1s,...,
   lm1,...,lmn,lms,t1,...,tn,p1,...,pn,ps,bs,
   empty,...,empty));
  ...
  out(chMachineN,machine(ball1,...,balln,empty,...,
   empty,...,empty,...,empty,l11,...,l1n,l1s,...,
   lm1,...,lmn,lms,t1,...,tn,p1,...,pn,ps,bs,
   empty,...,empty));
  out(chMachineS,machine(ball1,...,balln,empty,...,
   empty,...,empty,...,empty,l11,...,l1n,l1s,...,
   lm1,...,lmn,lms,t1,...,tn,p1,...,pn,ps,bs,
   empty,...,empty)).
```

Listing 4.33 – The process generating the machine.

```
let bidder(chBox:channel,chKey1:channel,...,chKeym:channel,
 chKeyT:channel,chKeyP:channel,b:ball,chKey:channel,
 chMachine:channel,chBid:channel,chWon:channel) =
  bid(b,one);
  out(chBox,b);
  new key1:key;
  ...
  new keym:key;
  new keyT:key;
  new keyP:key;
  out(chKey1,key1);
  ...
  out(chKeym,keym);
  out(chKeyT,keyT);
  out(chKeyP,keyP);
  in(chMachine,m:box);
  out(chKey,key1);
  in(chKey,k1:key);
  ...
  in(chKey,kn:key);
  let m1 = price_one(m,key1,k1,...,kn) in
  let (w1:ball,...,wm:ball) = check_window(m1) in
  if (w1 = one) then
    if((open_top1(m1,keyT)) = (empty,empty)) then
      event won(one,one)
    else
      0
  else
    ...
    out(chKey,keym);
    in(chKey,k1:key);
    ...
    in(chKey,kn:key);
    let mm = price_m(m(m-1),keym,k1,...,kn) in
    let (w1:ball,...,wm:ball) = check_window(mm) in
    if (w(n-1)m = m) then
      if(open_top1(mm,keyT) = (empty,empty)) then
        event won(m,one)
      else
        0
    else
      0.
```

Listing 4.34 – The process for the first bidder.

```
let seller(chKey1s:channel,...,chKeyms:channel,
 chKeyPs:channel,chKeyBs:channel,chKey:channel,
 chMachine:channel) =
  new key1:key;
  ...
  new keym:key;
  new keyP:key;
  new keyB:key;
  out(chKey1s,key1);
  ...
  out(chKeyms,keym);
  out(chKeyPs,keyP);
  out(chKeyBs,keyB);
  in(chMachine,m:box);
  in(chKey,k1:key);
  ...
  in(chKey,kn:key);
  out(chKey,key1);
  let m1 = price_one(m,k1,...,kn,key1) in
  let (w1:ball,...,wmm:ball) = check_window(m1) in
  if (w1 = one) then
    if((open_bottom(m1,keyB)) =
     (empty,something,...,something)) then
      event won(one,one)
    else
      ...
      if((open_bottom(m1,keyB)) =
       (something,...,something,empty)) then
        event won(one,n)
          else 0
  else
    in(chKey,k1:key);
    ...
    in(chKey,kn:key);
    out(chKey,keym);
    let mm = price_m(m(m-1),k1,...,kn,keym) in
    let (w1:ball,...,wmm:ball) = check_window(mm) in
    if (w(n-1)m = m) then
      if (open_bottom(mm,keyB) =
       (empty,something,...,something)) then
        event won(m,one)
      else
        if (open_bottom(mm,keyB) =
         (something,...,something,empty)) then
          event won(m,n)
        else 0
    else 0.
```

Listing 4.35 – The seller process.

```
process
  new chBox1:channel;
  new chKey11:channel;
  new chKey21:channel;
  new chKey31:channel;
  new chKeyT1:channel;
  new chKeyP1:channel;
  new chMachine1:channel;
  new chBox2:channel;
  new chKey12:channel;
  new chKey22:channel;
  new chKey32:channel;
  new chKeyT2:channel;
  new chKeyP2:channel;
  new chMachine2:channel;
  new chKey1s:channel;
  new chKey2s:channel;
  new chKey3s:channel;
  new chKeyPs:channel;
  new chKeyBs:channel;
  new chMachineS:channel;
  procMachine(chBox1,chBox2,chKey11,chKey12,chKey1s,
    chKey21,chKey22,chKey2s,chKey31,chKey32,chKey3s,
    chKeyT1,chKeyT2,chKeyP1,chKeyP2,chKeyPs,chKeyBs,
    chMachine1,chMachine2,chMachineS,chMachineP) |
  bidder(chBox1,chKey11,chKey21,chKey31,chKeyT1,
    chKeyP1,b1,chKey,chMachine1) |
  bidder2(chBox2,chKey12,chKey22,chKey32,chKeyT2,
    chKeyP2,b2,chKey,chMachine2) |
  seller(chKey1s,chKey2s,chKey3s,chKeyPs,chKeyBs,
    chKey,chMachineS)
```

Listing 4.36 – An instance with two bidders and three possible prices.

The script also generates a process `procMachine` (Listing 4.33 on page 259) that receives the marbles and all the keys from the bidders and the seller, and sends the resulting machine to all participants. They can then execute the "computation" on their copy of the machine, and only need to exchange the keys necessary. Note that after each bidder inserted his marble and added his locks, an intermediate state of the machine is published on channel `chMachineP`. Note also that this is an over-approximation since we create many copies of the same machine, which can even evolve differently, although this is not possible in the real world.

The ProVerif code for the first bidder is shown in Listing 4.34 on page 260. He sets up his keys and sends them together with his bid/marble to the process `procMachine` over private channels. Once he receives the machine object, publishes his first key and waits for the other bidders keys for the first layer, and the applies the function `price_one` to remove the first layer. Then he checks using `check_window` if somebody won, and if this is the case, he checks using `open_top1` if he himself won.

The seller is described by the code in Listing 4.35 on page 261. Similarly to the bidder he sets up his keys and sends them to the process `procMachine` over private channels. He then receives the machine and waits for the keys to remove the first layer. He published his key and applies the function `price_one`. If somebody won, he can determine the winner using `open_bottom`.

Note that in our model a bidder may insert at most one marble into the box, whereas in the real world he could try to insert several. This may lead to attacks on e.g. non-repudiation or non-cancellation: a bidder could insert two prices (marbles). The seller will observe one marble falling down when the first layer is removed. However, when he opens the bottom of the box to check for the winner, each column still contains a marble. To prevent this, the bidders and the seller check at the beginning if there is exactly one marble per column. Hence we argue that the approximation in our model is correct.

The above model allows us to prove using ProVerif that Woodako ensures *Non-Repudiation*, *Non-Cancellation*, *Weak Non-Interference*, *Highest Price Wins* and *Verifiability*.

For *Non-Repudiation* we consider two situations: In the first situation, we only assume an honest seller, and the event `bid` is executed by the process `procMachine` when he receives the marble. This corresponds to the seller observing the bidder inserting his marble into the box. In the second situation we only assume one honest bidder, he also executes the `won` events as in Listing 4.34 on page 260. In both situations ProVerif concludes successfully.

In the case of *Non-Cancellation* we removed the events `won` from the bidder process, they are only executed by the seller as we consider a dishonest bidder trying to cancel its bid.

We prove *Weak Non-Interference* using the `noninterf` command of ProVerif.

When verifying *Privacy* we consider two cases: If the seller is dishonest, the protocol only ensures secrecy of the losing bids, but the winner and winning price are revealed. If the seller is honest, the winner stays anonymous, and only the winning price is revealed. We can prove both results in ProVerif, but the verification takes approximately 24 hours for the dishonest seller and 36 hours for the honest seller case. This is due to the complexity of the equational theory and the equivalence proof. All other properties can be proved within a few seconds. Note that – as above – we only consider the base case (i.e. two bidders) due to the complexity of the equational theory for higher number of bidders and possible prices. Note also that we have to remove the publication of the intermediate states of the machine from process `procMachine` to avoid false attacks. Otherwise ProVerif finds an attack, where the functions `price_j` are applied for example on the first intermediate state using the published keys from bidder one, allowing to test only his bid for this price. This is obviously not possible in reality, as this intermediate state does not exist any more. Moreover we proved above that we have Weak-Noninterference, which means that the published intermediate states do not leak any information until the end of the bidding phase. Hence, as they are not existent any more after the end of the bidding phase, we can safely remove them in the analysis of Privacy.

For *Highest Price Wins* we consider an instance with two bidders, where one is honest and the other is corrupted. Since events can only be executed by honest bidders, he executes all `won` events, even if the other bidder wins (slightly differing from Listing 4.34 on page 260). This is possible since in the case he observes a marble falling down with his column still not empty, he can conclude that the other bidder won.

Concerning *Verifiability* we note that there is no need to design a test for rv_w: If a marble falls down, it must have been inside the box before, hence the winner is one of the participants, and there is nothing to verify.

For rv_s the test is obvious, but difficult to model: Anybody sees and verifies that the marbles and locks were added by eligible bidders, and for the rest of the process the locks on the transparent layer ensure that the bids cannot be tampered with. To model this, we use functions `hide`, `unhide` , `auth`, `authcheck` and `openauth`. The first two functions model the hiding of the marbles in the hand of the bidder, the remaining function model the implicit authentication using a secret (corresponding to the physical identity of the bidder), similar to a signature. We have the following equations:

$$\begin{aligned} \texttt{unhide}(\texttt{hide}(object, rand, hidesec), hidesec) &= object \\ \texttt{authcheck}(\texttt{auth}(object, authsec), \texttt{genPublic}(authsec)) &= object \\ \texttt{openauth}(\texttt{auth}(object, authsec)) &= object \end{aligned}$$

The ProVerif code used is shown in Listing 4.37. Note that above we did not model this part of the set-up phase explicitly, we simply modeled the interaction between the bidder and the machine as secret channels. This has the following reason: If we used the model with "hiding" and "authentication" above, the machine would be checking the authentication of the bidders, which is simply not the case. On the contrary, as the machine does not ensure that it is used by the correct participants, somebody else could insert his marble instead of an honest participant, and this would not be prevented by the machine itself. However, this will be detected by the other participants as we show here, and thus they will not continue the protocol. Hence we can argue that if the machine ends up being set up properly, authentication was correctly ensured, and hence that the model based on secret channels used above is realistic.

```
let bidder(chBox:channel,authsec:secret,hidesec:secret,
           b:ball) =
  new r:random;
  let hid = hide(b,r,hidesec) in
  event bid(hid);
  out(chBox,(auth(hid,authsec))).

let testrvs(chBox1:channel,,...,chBoxn:channel,
            authpub1:public,...,authpubn:public) =
  in(chBox1,(v1:authenticated));
  ...
  in(chBoxn,(vn:authenticated));
  let content1 = openauth(v1) in
  ...
  let contentn = openauth(vn) in
  if authcheck(v1,authpub1) = content1 && ... &&
   authcheck(vn,authpubn) = contentn then
    event accepted(content1,...,contentc)
  else
    event rejected().
```

Listing 4.37 – The code used to verify *testrvs*.

Concerning ov_l, ov_w and ov_s, we use the verification functions designed for this purpose: A losing bidder checks that his column is not empty, a winning bidder that his column is empty, and the seller opens the bottom to identify the winner. Actually, these functions are already employed during the winner determination process, hence we only check that this is sound and complete, i.e. that they only accept a correct outcome, and accept the outcome if everybody behaves honestly. We check this using the reachability of an event `bad`, similar to the previous case studies.

4.5.3 — Summary

We argued that verifiability of an auction should not depend on cryptographic expertise: without understanding, there is no meaningful verifiability. With that in mind, we proposed two protocols based on physical objects, inspired by the cryptographic auction protocol due to Sako. In both cases, we can achieve the desired security properties *without* cryptography or trusted parties[20].

In our first proposal, the Cardako protocol, each bidder marks their bid by making a hole in their piece of cardboard, which is then put into an envelope. Using a needle, it is possible to detect if a hole is in a certain place (i.e., if the envelope contains a bid for this price), without opening the envelope, and hence determine the winner without revealing the contents of the losing bids. We modeled the physical process of testing the envelope for a certain price using an special equational theory in ProVerif, which allows us to apply the exact same analysis as for the cryptographic protocols in previous sections. The analysis successfully proved *Non-Repudiation*, *Non-Cancellation*, *Weak Non-Interference*, *Verifiability*, and *Highest Price Wins*. For privacy, an issue was automatically found: dishonest participants may open an envelope or test an envelope for all possible bids. A mitigation is that such actions are readily detectable by all, as any handling of the envelopes occurs in public, and any such test either breaks the envelope or leaves a hole that can be detected by the other bidders. If we assume that such actions are not undertaken, we can show Strong Bidding-Price Secrecy using ProVerif.

To achieve an even higher level of privacy, we introduced the Woodako protocol. This protocol is again inspired by Sako's protocol, and again replaces cryptography and trusted parties by physical properties. This time, bids are represented by marbles, where smaller marbles denote higher bids. Bidders place the marble corresponding to their bid in their designated column in an entirely mechanical machine. Then, the first layer below all columns is removed, leaving a new layer with holes the size of the smallest marble. If at least one marble falls through, there is a winner, otherwise this layer is removed and the next layer with larger holes is now the base layer. We argued that Woodako achieves *Non-Repudiation*, *Non-Cancellation*, *Weak Non-Interference*, *Verifiability*, and *Highest Price Wins*. Moreover, this argumentation did not require any expert knowledge to understand, nor did it hinge on correct behavior by trusted parties.

Finally, we formally analyzed the Woodako protocol, again modeling physical properties in equational theory. The model of our physical implementation was proven correct with respect to the mentioned security properties using ProVerif. As the seller knows the winning bidder, a dishonest seller can reveal the winner.

[20] In the case of Cardako misbehaving parties could break some properties, however any such breach is immediately detected.

As such, we automatically proved privacy for all bidders including anonymity of the winner in case of an honest seller, and simple privacy for losing bidders in case of a dishonest seller.

We admit that both protocols have practically issues. Neither of them scales well for higher numbers of bidders or possible prices, and both require all bidders to be in the same room at least during the winner determination phase. Hence we need to find more practical protocols.

Moreover, the soundness of our models is a major direction of future work. How can we argue that a model is sound with respect to the physical work and its infinite amount of possibilities of interactions and combinations of objects? This is an important issue in particular for coercion-resistance: physical objects can be manipulated in many ways in order to create receipts or divulge information using side-channels.

Additionally, the proofs we currently have for both protocols are not generic in the number of bidders and bids. As future work, we are looking to provide a fully generic proof. One main issue is the state space of the Woodako protocol: it scales too fast for automated proofs with higher numbers of bidders and prices.

4.6 Conclusion

In this chapter we discussed auction protocols. We identified many desirable security properties:

— Fairness, including Strong Noninterference, Weak Noninterference and Highest Price Wins

— Authentication, including Non-Repudiation and Non-Cancellation

— Privacy, including Strong Bidding-Price Secrecy, Bidding-Price Unlinkability, Weak Anonymity, Strong Anonymity, as well as Receipt-Freeness and Coercion-Resistance

— Verifiability, including Registration and Integrity Verifiability as well as Outcome Verifiability

We defined the Fairness, Authentication and Privacy properties in the Applied π-Calculus and automatically analyzed three case studies using ProVerif: the protocols due to Curtis et al. [CPS07], Brandt [Bra06] and Sako [Sak00]. We were able to prove all properties except for receipt-freeness and coercion-resistance for the protocol due to Sako, but identified several problems with the other two. The protocol by Brandt completely lacks authentication. The protocol by Curtis et al. leaks some information about the bids, and also lacks authentication in the final messages between the registration authority and the seller.

In the next part of the chapter, we proposed a high-level model of Verifiability in auctions. We defined soundness and completeness conditions for the necessary verification tests, inspired by the perspectives of the participants. Our definitions were initially designed for first-price auctions, however we also gave a generalized version that can accommodate second-price or bulk-good auctions.

We then showed using the example of Sako's auction protocol how this definition can be instantiated in the symbolic model as well as in the computational model. We were able to prove that the verification test are sound and complete, using ProVerif in the symbolic model, and CryptoVerif in the computational model. We also analyzed the protocol by Curtis et al., and identified several issues. For example, some messages lack the necessary authentication, and the registration authority needs to be trusted for the management of pseudonyms.

In the last part of the chapter we discussed true bidder-verifiable auctions. Starting from the observation that auction protocols typically heavily rely on cryptography to achieve verifiability, which makes them too complex for many bidders to understand, we explored the idea of ensuring verifiability (as well as other properties) only using physical properties of physical objects.

We proposed two protocols: Cardako and Woodako. Cardako is a simple implementation of Sako's protocol using office material, whereas Woodako uses a wooden machine to compute the winner in a private, but verifiable way. We argued that both protocols ensure the desired properties, and also analyzed them formally in ProVerif. To model the properties of the physical objects we used special equational theories.

4.6.1 — Limitations and Future Work.

Most of our Fairness and Authentication definitions are only meaningful for first-price auctions, however a generalization to other auction types appears to be feasible. We would also like to refine the model used for the protocol by Brandt to capture that the losing bidders do not learn the winner. Then we would like to fix the issues identified with the protocols by Brandt and Curtis et al., and prove the security of the resulting protocols. This however implies dealing with some limitations of ProVerif, in particular concerning state. Similarly we often had to extend the ProVerif proofs manually to cover the general case, it would be desirable to obtain these results directly. There are already some extensions of ProVerif addressing the limitations, for example StatVerif [ARR11] allows to model stateful processes. We would also like to test our notions of Receipt-Freeness and Coercion-Resistance on suitable protocols.

Concerning our computational analysis of the protocol by Sako, it would be interesting to refine the proof by replacing the abstract public key decryption with the concrete ElGamal cryptosystem to obtain an even stronger result.

Finally concerning our work on true bidder-verifiable auctions, we would like to have more practical protocols. Another direction for future work is the soundness of models including physical objects - how can we obtain models that are arguably sound with respect to all the possibilities present in the physical world?

Chapter 5

Conclusion

In this chapter we summarize our results. We also discuss the limitations of our work, and directions for future research.

Contents

5.1 Summary 272

5.2 Limitations and Directions for Future Research 274

5.1 Summary

We introduced the context and motivated this work in Chapter 1.

In Chapter 2, we recalled the syntax and semantics of the Applied π-Calculus. We also identified two important subclasses of processes: Normed and finite processes. Normed processes are processes that have a finite complete trace, and finite processes are processes where all complete traces are finite. We showed that any normed process P can be rewritten as the parallel composition of prime factors in a unique (up to $\sim_l$) way, i.e. such that $P \sim_l P_1|\ldots|P_n$. Each factor P_i is prime in the sense that any further decomposition into $P_i \sim_l P_i^A|P_i^B$ implies that at least one of the factors is trivial, i.e. either $P_i^A \sim_l 0$ or $P_i^B \sim_l 0$. We also showed that any finite process P can be decomposed uniquely up to $\approx_l$ into (weakly) parallel prime factors $P \approx_l P_1|\ldots|P_n$.

We then discussed privacy in eVoting in Chapter 3. We proposed a taxonomy of privacy in eVoting protocols in the Applied π-Calculus. It accounts for different attacker capabilities (insider or outsider), special attacks (e.g. forced-abstention) as well as different levels of coercion (simple vote-privacy, receipt-freeness or coercion-resistance). As case studies we discussed the protocols by Fujioka et al. [FOO92], Okamoto [Oka96], Lee et al. [LBD+03] and Bingo Voting [BMQR07]. The protocol by Fujioka et al. ensures Vote-Privacy for an insider but is not secure against forced abstention attacks. Receipt-Freeness against an insider is ensured by the protocol by Okamoto, yet it remains vulnerable to forced abstention attacks. Both protocols can be secured against forced abstention attacks by using a private channel to a trusted administrator. The protocol by Lee et al. ensures Coercion-Resistance, but only against an outsider, and is vulnerable to forced abstention attacks. Finally Bingo Voting achieves Coercion-Resistance against an insider and is secure against forced abstention attacks, if the list of participating voters is not publicly announced.

In the next part, we developed generalized privacy definitions to accommodate protocols supporting weighted votes. We proposed such definitions for Privacy, Receipt-Freeness and Coercion-Resistance, with and without one or several corrupted voters. For Receipt-Freeness and Coercion-Resistance we also considered multiple coerced voters. Moreover, we established precise links between the new and the previous notions in the taxonomy. We considered a variant of the protocol by Fujioka et al. as a case study. Additionally we showed that for protocols ensuring a certain modularity condition the notions with and without corrupted bidders are equivalent. We were also able to prove that in such a case Multi-Voter Coercion-Resistance and Single-Voter Coercion-Resistance as well as Multi-Voter Receipt-Freeness and Single-Voter Receipt-Freeness coincide.

In Chapter 4, we analyzed electronic auction protocols. We started by giving

a formal model of auction protocols in the Applied π-Calculus, and by defining fairness, authentication and privacy properties. We defined Non-Repudiation, Non-Cancellation, Weak and Strong Non-Interference, Highest Price Wins as well as different notions of Privacy, Receipt-Freeness and Coercion-Resistance. These different privacy notions allow to accommodate protocols where the winner stays anonymous or not, and where losing bids are public (yet unlinkable to the bidders) or remain private. We also provided generalized notions for auctions that are not first-price auctions, and provided a formal link to the previous notions. As case studies, we analyzed the protocols by Curtis et al. [CPS07], Brandt [Bra06] and Sako [Sak00]. The protocol by Sako turned out to ensure all properties except for receipt-freeness and coercion-resistance, yet the authority needs to be trusted for privacy. The protocols by Curtis et al. and Brandt both suffered from several shortcoming, in particular both lacked authentication of some messages which can be exploited by an attacker. For the protocol by Curtis et al. we also provided some corrections.

In the next part of the chapter we analyzed Verifiability in electronic auction protocols. We gave a new, high-level model of an auction protocol and the necessary verification tests. This definition can be instantiated in the symbolic or the computational model. To illustrate this, we provided a symbolic (i.e. using ProVerif and a manual generalization) and a computational (using mostly CryptoVerif and one manual proof) proof of Verifiability for Sako's protocol. We also analyzed the protocol by Curtis et al. using ProVerif, and again identified several flaws.

Finally we explored the idea of "true bidder-verifiable auctions", i.e. auctions that can be verified without any specialist knowledge. To achieve this, we proposed two protocols ensuring verifiability only through physical properties. The first one, called "Cardako", only uses office material: bids are holes in a piece of cardboard, which is then inserted into an envelope. This allows to test if a bid contains a specific price, without revealing the bid entirely. We argued informally that the protocol ensures all the desired properties, and also proposed a modeling of the physical objects and their properties in ProVerif using a special equational theory to apply our formal definitions. Using this model we identified an attack on privacy if the parties are dishonest, however this attack can be detected by the other bidders.

Our second protocol is called "Woodako" and relies on a wooden box determining the winner in a secure way based on the properties of the box. In this case bids are marbles (of different sizes), and the first marble to come out of the machine determines the winner. In addition to Cardako this protocol is secure even if all other parties are dishonest, and reveals the winner only to the winner and the seller – the losing bidders only learn that they lost. Again, we provided

informal and formal arguments for its security. Similarly to Cardako we model the box using a special equational theory and use ProVerif to conclude.

5.2 Limitations and Directions for Future Research

Concerning our unique decomposition results, we see two lines of future work. The first line is concerned with the extension of a parallel decomposition to infinite processes containing replication. A first approach in this direction was proposed by Hirschkoff and Pous [HP10] for a subset of CCS with top-level replication. They introduced the concept of a *seed* of a process P. It is defined as the process Q, Q bisimilar to P, of least size in terms of prefixes whose number of replicated components is maximal among the processes of least size. They showed that this representation is unique. Moreover, they showed that this generalizes to the Restriction-Free-π-Calculus (i.e. without "ν"). It remains however open if a similar result can be obtained for the full calculus.

The second line of research is to find an efficient algorithm that converts a process into its unique decomposition. It is unclear if such an algorithm exists and can be efficient, as simply deciding if a process is finite can be non-trivial.

A more technical relic is that we did not prove that Strong Labeled Bisimilarity is closed under the application of contexts, although we expect such a result to hold due to its close similarity to Weak Labeled Bisimilarity.

Similarly, we did not give a formal proof showing that we can replace a process P synchronizing (i.e. containing a `sync`) with other processes within a context $C[\_]$, with a bisimilar (modulo the synchronization) processes $P' \approx_l P$, i.e. that we have $C[P] \approx_l C[P']$, although again we expect this result to hold.

With respect to our work on Privacy we also see two main directions for future work. Firstly most of our proofs were manual, and it would be great to mechanize more of these proofs as they tend to be long and cumbersome. The main obstacles are too complex equational theories (as in the case of the protocol by Okamoto in Section 3.3.5.2) and the universally quantified attacker context in the definition of Coercion-Resistance. A step in this direction was undertaken by Smyth et al. [SAR13]. They propose to replace the complex equational theory with a simpler, but equivalent one which can be then treated by ProVerif. Moreover, KISS [CDK12] and AKISS [CCK12] can deal with more complex equational theories. KISS can verify the static equivalence of frames, which allowed us to verify the static equivalence of the final frames in the proof of the protocol by Okamoto. AKISS can prove trace equivalence in a calculus similar to the Applied π-Calculus. However, it is still an open problem to automatically reason about the universally quantified context in the definitions of Coercion-Resistance.

A second direction for future work is a translation of our definitions to a

computational setting. In our symbolic model we do not consider probabilities, i.e. we call a protocol secure if there is a possibility for the voter to escape coercion, even if an attacker has a noticeable probability of detecting this. A computational variant of our definition would be able to take this into account.

Concerning our work on eAuctions, we see many possibilities for future work. A first idea is to try to fix all the identified issues with the protocols by Curtis et al. and Brandt. As we already noted, this also implies dealing with some of the limitations of ProVerif. Moreover, the current model of the protocol by Brandt is a simplification, it would be interesting to extend this to a more precise model.

When using ProVerif we often were unable to prove the general case directly, and had to provide a manual proof to show that the result holds for any number of bidders. An extension of ProVerif with loops could permit to obtain the general proof directly.

Additionally, none of our case studies turned out to ensure any notion of Receipt-Freeness or Coercion-Resistance. The auction protocol by Abe and Suzuki [AS02] was shown to ensure Receipt-Freeness in the model by Dong et al. [DJP11], it would be interesting to see if this generalizes to our model.

To strengthen our computational proof of verifiability of Sako's protocol, we would like to refine the model by replacing the abstract primitive "secure and correct encryption" with the concrete ElGamal encryption.

Finally, we would like to develop more practical and scalable true bidder-verifiable auction protocols. However even the current protocols raise many interesting questions for future work with respect to the application of formal methods to protocols that combine cryptography with physical properties. The main question is how to model physical objects and their interactions in a realistic way. Given the richness of the physical world this is not obvious, and the question of soundness is a challenging one. For cryptographic protocols we usually consider soundness with respect to the computational model (although this still abstracts away from implementation errors and side-channels), yet when considering physical objects it is not clear with respect to which model our reasoning should be "sound".

Moreover, in [DDL13] we provided a detailed cryptanalysis of Brandt's fully private auction protocol [Bra06]. Although we did not discuss the results in this thesis, in particular the discovered flaws and proposed fixes, it would be interesting to provide a formal (computational) proof of the corrected protocol.

Chapter 6

Résumé en Français

CETTE partie contient des résumés de tous les chapitres précédents en français.

Contents

6.1 Introduction 278

6.1.1 Contributions 280

6.1.2 Publications précédentes 282

6.2 Le π-calcul appliqué et la décomposition unique des processus 282

6.3 Les protocoles de vote 283

6.3.1 Taxonomie 284

6.3.2 Votes pondérés 286

6.4 Les protocoles de vente aux enchères 287

6.5 Conclusion 289

6.5.1 Perspectives 290

6.1 Introduction

De plus en plus d'échanges commerciaux se passent sur internet, par exemple la vente des biens – Amazon.com a atteint en 2012 un chiffre d'affaires de plus de 61 milliards de dollars [Ama13] – ou les ventes aux enchères en ligne via des sites tels que eBay dont le chiffre d'affaires en 2012 fût de plus que 14 milliards de dollars [Don12]. D'autre part, les administrations et le gouvernement utilisent de plus en plus des systèmes informatiques pour des applications d'administration. Par exemple le vote électronique, qui est utilisé par exemple en Estonie [Est] ou dans certains cantons suisses [Gen13, Reg13].

Pour faire interagir différents systèmes ou implémenter une application comme le vote ou les ventes aux enchères en ligne, de nombreux protocoles ont été développés. Ces protocoles définissent l'interaction entre les différents participants, et sont conçus pour assurer des propriétés de sécurité (comme l'authentification ou le secret) aussi bien que des propriétés fonctionnelles (comme la disponibilité ou une réponse en temps réel).

Malheureusement le développement de tels protocoles est difficile et sensible aux erreurs. Une approche pour maîtriser cette problématique est l'utilisation des méthodes formelles. Les méthodes formelles utilisent des modèles formelles comme des logiques dédiées, des algèbres de processus ou des arguments statistiques ou probabilistes pour analyser la sécurité d'un système. Elles permettent non seulement de découvrir des erreurs, mais aussi de prouver qu'un système est sûr par rapport à une propriété précise dans un modèle donné.

Le « Common Criteria for Information Technology Security Evaluation » [Com12a], un standard international pour la certification des systèmes d'information critiques, exige une évaluation formelle du système pour les deux niveaux de certification les plus hauts, les « Evaluation Assurance Levels (EALs) » 6 et 7 [Com12b].

L'utilisation des méthodes formelles a connu des nombreux succès. Depuis les travaux importants de Dolev et Yao [DY81, DY83] et Millen [Mil84] sur les protocoles à clé publique, le développement de la logique BAN [BAN90], et les résultats bien connus de Lowe [Low96] sur l'analyse automatique du protocole de Needham-Schroeder, beaucoup de faiblesses dans des protocoles et standards existants et déployés ont été identifiées. Par exemple Mitchell, Shmatikov et Stern [MSS98] ont découvert des anomalies dans SSL (« Secure Socket Layer ») 3.0, et Delaune, Kremer et Steel [DKS10] ont constaté plusieurs faiblesses dans le standard PKCS#11 pour les clés USB cryptographiques.

Mais les méthodes formelles peuvent aussi non seulement servir à identifier des failles de sécurité, mais aussi à obtenir des preuves de sécurité. Par exemple He et al. [HSD+05] ont donné une preuve modulaire du standard IEEE 802.11i

et du protocole « Transport Layer Security (TLS) », un des protocoles les plus utilisés sur internet. Plus récemment, une implémentation entièrement vérifiée de TLS a été réalisée [BFK+13].

Les principaux défis restent le choix du modèle, la formalisation des propriétés de sécurité, et le développement des outils de vérification automatique.

Dans cette thèse nous étudions deux applications principales : le vote électronique et les ventes aux enchères électroniques. Les systèmes de vote électroniques ont été utilisés dans de nombreux pays du monde, et maintenant même des systèmes de vote en ligne sont introduits, par exemple en Estonie [Est], dans plusieurs cantons suisses [Gen13, Reg13] et même pour les français expatriés [Min13]. Vue l'importance du vote dans les démocraties modernes, les exigences de sécurité élevées amènent à des discussions controversées [Par07, UK 07, Min08, Bun09].

La deuxième application, les ventes aux enchères électroniques, a connu un grand succès : par exemple eBay avait plus que 112 millions d'utilisateurs actifs, et plus que 350 millions d'offres en 2012 [Don12]. Comme il s'agit d'un processus compétitif – les participants cherchent à payer le prix le plus bas possible – engageant des sommes d'argent considérables, des fraudes sont courantes [NTJ13] et la sécurité est un enjeu majeur.

Dans les deux domaines d'application nous considérons des systèmes complexes et des propriétés non-triviales comme le respect de la vie privée, l'équité et la vérifiabilité. Le respect de la vie privée peut simplement vouloir dire le secret du vote (ou de l'offre), mais aussi l'anonymat du votant ou de l'enchérisseur. L'équité est souvent liée au respect de la vie privée car par exemple des résultats préliminaires peuvent influencer les votants restants, mais inclut aussi la protection contre la fraude, et le respect des règles des protocoles de vente aux enchères. Finalement la vérifiabilité permet aux participants d'un protocole de vérifier à la fin que le protocole s'est bien déroulé, et que les résultats annoncés sont corrects. Ceci est particulièrement intéressant pour les systèmes complexes, car la vérification peut être plus simple que l'exécution du protocole.

Puisque les définitions en langue naturelle sont souvent imprécises, un défi majeur pour la vérification formelle est le développement des définitions formelles précises, qui couvrent tous les aspects de la propriété en question, et qui sont adaptées à la vérification automatique.

Le but de cette thèse est de proposer des modèles et des définitions qui permettent de vérifier des propriétés de sécurité dans les deux contextes. Pour le vote, nous nous concentrons sur les propriétés d'anonymat, qui incluent l'absence de reçu et la protection contre la coercition, c'est-à-dire contre un intrus qui essaye de forcer les votants à voter pour un candidat de son choix. Pour les ventes aux enchères nous proposons des modèles et des définitions pour les propriétés d'anonymat, d'équité et de vérifiabilité. De plus, nous examinons plusieurs études

de cas, ce qui amène aussi bien à des preuves de sécurité qu'à la découverte de plusieurs failles de sécurité. Finalement nous présentons aussi un résultat théorique dans le π-calcul appliqué qui nous permet de prouver l'équivalence de plusieurs notions d'anonymat.

6.1.1 — Contributions

Dans le chapitre 2, nous rappelons la syntaxe et la sémantique du π-calcul appliqué, qui est utilisé de manière récurrente dans la thèse. Nous présentons ainsi deux résultats de décomposition unique pour des processus dans le π-calcul appliqué. Nous commençons par la définition de deux sous-classes de processus : les processus *normés* et *finis*, c'est-à-dire les processus qui possèdent au moins une trace complète (c'est-à-dire qui se termine dans un état ou aucune transition n'est possible) finie, et les processus où toutes les traces complètes sont finies respectivement. Dans un premier temps, nous démontrons que tout processus normé peut se réécrire de manière unique (modulo la bisimulation forte) sous forme de composition parallèle de plusieurs facteurs premiers, c'est-à-dire nous avons $P \sim_l P_1|\ldots|P_n$ où tous les P_i sont premiers, c'est-à-dire ne peuvent pas être décomposés dans des facteurs non-triviaux. Dans un deuxième temps, nous prouvons que tout processus fini (c'est-à-dire toutes les traces sont finies) peut se réécrire de manière unique (modulo la bisimulation faible) sous forme de composition parallèle de plusieurs facteurs premiers : $P \approx_l P_1|\ldots|P_n$. Ces résultats démontrent l'existence d'une forme normale, et impliquent un résultat d'annulation sur les processus : nous obtenons que $A|B \sim C|B$ implique $A \sim C$. Ce résultat est utilisé au chapitre 3 dans une preuve d'équivalence de différentes notions de sécurité.

Dans le chapitre 3 nous analysons le respect de la vie privée dans les systèmes de vote électroniques. Dans un premier temps, nous proposons une taxonomie formelle du respect de la vie privée dans le π-calcul appliqué. Cette taxonomie tient compte de différentes capacités de l'intrus (par exemple un intrus interne qui participe aussi au vote, ou un intrus externe qui est juste observateur), de différents types d'attaques (l'abstention forcée), et du niveau de coercition employé par l'intrus. Nous appliquons cette taxonomie dans plusieurs études de cas : les protocoles de Fujioka et al. [FOO92], de Okamoto [Oka96], de Lee et al [LBD$^+$03], et Bingo Voting [BMQR07].

Dans un deuxième temps, nous généralisons les notions du respect de la vie privée pour prendre en compte les votes pondérés, par exemple dans le cas d'une société où les votes seraient proportionnels aux nombres d'actions de chaque votant. Dans un tel cas les définitions précédentes fondées sur la permutation ne sont plus adaptées, car une permutation peut changer le résultat de l'élection et donc rendre les deux situations trivialement distinguables. Notre solution consiste à s'abstraire du résultat, et à considérer toutes les distributions de votes qui

amènent au même résultat. Nous appliquons ces nouvelles notions à un protocole inspiré par le protocole d'Eliasson and Zúquete [EZ06], qui implémente des votes pondérés. De plus, nous établissons des liens précis entre les notions généralisées, et les notions de la taxonomie précédente : nous démontrons que les deux sont équivalentes sous l'hypothèse que les votes ont tous le même poids. Dans une étape supplémentaire, nous démontrons que si le protocole est fini et satisfait une certaine condition de modularité (ce qui est le cas dans beaucoup d'exemples), la coercition d'un ou de plusieurs votants est équivalente. Ces conditions permettent aussi de prouver que le cas avec des votants corrompus par l'intrus peut se réduire au cas sans votants corrompus. Cela permet de simplifier les preuves de sécurité, car il suffit d'analyser le cas sans votants corrompus et avec un seul votant sous coercition.

Dans le chapitre 4 nous analysons les protocoles de vente aux enchères en ligne. Dans la première partie nous proposons des définitions des propriétés d'authentification comme la non-répudiation et la non-annulation, des propriétés d'équité comme la non-interférence des offres et la protection contre la fraude, et de la protection de la vie privée de l'anonymat dans le π-calcul appliqué. Dans la suite nous analysons trois études de cas : les protocoles de Curtis et al. [CPS07], de Brandt [Bra06] et de Sako [Sak00]. Nous identifions automatiquement grâce à l'outil ProVerif plusieurs problèmes pour les deux premiers, et obtenons des preuves automatiques en ProVerif pour le dernier.

Dans la deuxième partie nous analysons la vérifiabilité des protocoles d'enchère. Nous proposons une définition abstraite, qui peut s'instancier aussi bien dans le modèle symbolique que dans le modèle calculatoire. Dans la suite, nous étudions deux exemples : les protocoles de Sako et Curtis et al.. Pour le protocole de Sako, nous proposons une preuve symbolique en utilisant ProVerif (et des généralisations manuelles), et une preuve calculatoire en utilisant CryptoVerif (et une preuve manuelle). Pour le protocole de Curtis et al. nous employons ProVerif et identifions plusieurs faiblesses.

Dans la dernière partie nous explorons l'idée des « enchères vraiment vérifiables par les enchérisseurs », c'est-à-dire des protocoles qui assurent la vérifiabilité sans utiliser des opérations cryptographiques complexes, et qui sont donc compréhensibles même pour une personne non spécialiste. Nous proposons d'utiliser des propriétés physiques de certains objets, et développons deux protocoles inspirés du protocole proposé par Sako. Le premier s'appelle « Cardako » et n'utilise que du matériel de bureau, c'est-à-dire du carton et des enveloppes. Le deuxième s'appelle « Woodako » et utilise une boîte en bois qui détermine le gagnant de manière privée et vérifiable. Bien que ces protocoles aient leur limitations pour passer à l'échelle, ils illustrent qu'on peut réaliser des enchères simplement en utilisant des objets physiques. Nous proposons aussi d'appliquer les définitions

formelles proposées plus tôt à ces protocoles, et discutons leur vérification formelle automatique en ProVerif en utilisant une théorie équationnelle particulière qui modélise les propriétés physiques. Avec cette approche, nous obtenons que les deux ont les propriétés souhaitées.

Finalement, nous résumons nos résultats dans le chapitre 5 et discutons des perspectives.

6.1.2 — Publications précédentes

Une grande partie du travail présenté dans cette thèse a déjà été publiée avant la publication de cette thèse à l'occasion de plusieurs conférences internationales.

Les résultats sur la décomposition unique du chapitre 2 ont été présentés à FoSSaCS 2013 [DELL13]. Les premiers résultats du travail présenté dans le chapitre 3 ont été exposés à FPS 2011 [DLL11], ICC-SFCS 2012 [DLL12b] et à ESORICS 2012 [DLL12a]. Une grande partie des résultats du chapitre 4 a aussi été publiée : à POST 2013 [DLL13] et à ASIACCS 2013 [DJL13].

Bien que cela ne fasse pas partie du travail présenté dans cette thèse, nous avons aussi publié une analyse détaillée du protocole par Brandt à Africacrypt 2013 [DDL13].

6.2 Le π-calcul appliqué et la décomposition unique des processus

Le π-calcul appliqué [AF01] est une algèbre de processus. Il s'agit d'une variante du π-calcul adaptée à la vérification des protocoles. Dans cette algèbre, les participants du protocole sont décrits sous forme de processus qui peuvent interagir, par exemple échanger des messages. Tous les messages sont composés de termes, qui sont évalués par rapport à une théorie équationnelle qui modélise par exemple les propriétés des opérations cryptographiques (par exemple $\mathtt{dec}(\mathtt{enc}(m,k),k) = m$ pour un chiffrement symétrique).

Les grammaires pour la construction des termes et des processus sont données dans les figures 2.1, 2.2 and 2.3 : le processus 0 ne fait rien, la composition parallèle $P|Q$ exécute P et Q en parallèle, et la réplication $!P$ exécute un nombre infini de copies de P en parallèle. $\nu n.P$ crée un nom frais et restreint n, puis continue tant que P. $\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$ se comporte comme P si $N =_E M$, ou comme Q sinon. Le processus $\mathtt{in}(u,x).P$ reçoit un message sur le canal u, l'affecte à la variable x et puis continue comme P. La sémantique est donnée dans les figures 2.4, 2.5 et 2.6. Cette sémantique permet de définir plusieurs notions d'équivalence, notamment la bisimulation forte ($\sim_l$) et la bisimulation faible ($\approx_l$). Dans la bisimulation forte chaque transition d'un côté doit être simulée

par exactement une transition de l'autre côté, dans le cas de la bisimulation faible une transition peut être simulée par plusieurs transitions (internes).

Dans une algèbre de processus comme le π-calcul appliqué une question naturelle est celle de la décomposition en facteurs parallèles : étant donné un processus P, existent-ils des processus $P_1, \ldots, P_n$ tel que P peut se réécrire comme $P_1|\ldots|P_n$? Et si cette décomposition existe, est-elle unique ?

Une telle décomposition a plusieurs applications : elle sert de forme normale, ce qui simplifie certaines preuves, elle permet de vérifier l'équivalence de deux processus en vérifiant qu'ils ont la même forme normale, ou encore elle implique un résultat d'annulation, c'est-à-dire que $P|Q \approx P|R$ implique $Q \approx R$.

Dans ce chapitre nous démontrons que dans le π-calcul appliqué tout processus « normé » P peut se réécrire de manière unique modulo la bisimulation forte sous la forme suivante

$$P \sim_l P_1|\ldots|P_n$$

où $P_1, \ldots, P_n$ sont *premiers*, c'est-à-dire qu'ils ne peuvent pas se décomposer dans des facteurs non-triviaux (Théorèmes 5 et 6). Un processus est « normé » s'il possède au moins une trace complète (c'est-à-dire qui se termine dans un état ou aucune transition n'est possible) finie.

De manière similaire, nous prouvons que tout processus « fini » (c'est-à-dire toutes les traces complètes sont finies) P peut se réécrire de manière unique modulo la bisimulation faible sous la forme suivante

$$P \approx_l P_1|\ldots|P_n$$

où $P_1, \ldots, P_n$ sont premiers (Théorèmes 13 et 14).

Dans les deux cas nous obtenons un résultat d'annulation, c'est-à-dire que

— $P|Q \sim_l P|R$ implique $Q \sim_l R$ pour P, Q et R normés (Lemme 7)

— $P|Q \approx_l P|R$ implique $Q \approx_l R$ pour P, Q et R finis (Lemme 15)

Ce dernier résultat permet dans le chapitre 3 de démontrer l'équivalence de plusieurs notions d'anonymat pour les protocoles de vote.

6.3 Les protocoles de vote

Puisque le vote est un acte central de participation dans les démocraties modernes, de nombreuses propriétés de sécurité ont été proposées :

— *Correction* : le résultat annoncé correspond à la somme de tous les votes.

— *Éligibilité* : seulement les votants enregistrés peuvent voter, et au plus une fois.

— *Équité* : le processus de vote est équitable, notamment aucun résultat prélimi-

naire n'est publié afin de ne pas influencer les autres votants.

— *Robustesse* : le protocole peut tolérer des votants avec un comportement incorrect.

— *Vérifiabilité* : le protocole possède des mécanismes qui permettent de vérifier l'exactitude du résultat.

— *Respect de la vie privée* : tous les votes restent secrets.

La dernière propriété est cruciale pour assurer que les votants soient libres de leurs choix. Dans la littérature, elle est souvent divisée en plusieurs notions (p.ex. dans [DKR09, LSB$^+$09, SC11, SB13, Jon09, DHvdG$^+$13]) :

— *Secret du vote* : Tous les votes restent secrets pour un observateur extérieur.

— *Absence de reçu* : Un votant ne peut pas créer un reçu qui permet de prouver à une tierce partie qu'il a voté pour un certain candidat. Ceci permet d'éviter l'achat des votes.

— *Résistance à la coercition* : Même si un votant interagit avec un intrus pendant tout le processus de vote, l'intrus ne peut pas savoir avec certitude si le votant a suivi ses ordres ou voté différemment.

— *Protection contre l'abstention forcée* : Un intrus ne peut pas forcer un votant à s'abstenir.

— *Indépendance des votes* : Aucun votant ne peut lier son vote au vote d'un autre votant[1].

— *Secret perpétuel* : Le secret des votes n'est pas seulement assuré au moment de l'élection, mais aussi sur le long terme, quand la puissance de calcul de l'intrus aura considérablement augmenté.

Dans ce chapitre nous étudions toutes ces notions, sauf le secret perpétuel.

6.3.1 — Taxonomie

Dans le contexte du vote, le respect de la vie privée est difficile à définir formellement à cause de la grande variété d'attaques et de capacités de l'intrus. Pour systématiser notre définition, nous proposons une taxonomie basée sur une approche modulaire.

Dans notre scénario, l'intrus cible un votant (le *votant ciblé*) et essaye de connaître son vote. Si l'intrus connaît les votes de tous les autres votants, il peut déduire son vote à partir du résultat affiché. Nous supposons donc que l'intrus ne

[1]La possibilité de copier des votes peut compromettre le secret des votes si le nombre de votants est faible, ou si un grand nombre de votants est corrompu. Supposons une instance avec trois votants : le troisième votant peut copier le vote du premier et le soumettre comme son vote. Cela résulte en au moins deux votes pour le candidat choisi par le premier votant ; ce dernier peut donc être déduit du résultat du vote.

connaît pas le vote d'au moins un autre votant (le *votant de compensation*).

Nous exprimons le secret comme une équivalence observationnelle entre deux situations. Dans la première situation, le votant ciblé vote comme l'intrus le souhaite, dans la deuxième il vote différemment. Le votant de compensation contrebalance le vote changé en changeant son propre vote pour assurer que le résultat reste le même dans les deux situations. Si le protocole assure le secret du vote, l'intrus devrait donc être incapable de distinguer les deux situations.

En partant des définitions de résistance à la coercition, de l'absence de reçu et de secret du vote nous proposons de factoriser trois dimensions : communication entre le votant ciblé et l'intrus, intrus interne ou externe, et protection contre l'abstention forcée.

1. *Communication entre le votant ciblé et l'intrus :* Nous considérons trois niveaux différents :
   (a) Dans le cas le plus simple, l'intrus observe simplement les données publiques. Ce cas s'appelle « Swap-Vote-Privacy », noté « $SwVP$ ».
   (b) Dans le deuxième cas le votant ciblé essaye de construire un reçu en révélant toutes ces données privées pour convaincre l'intrus qu'il a voté pour un certain candidat. Ce cas s'appelle « Swap-Receipt-Freeness », noté « $SwRF$ ».
   (c) Dans le cas le plus fort, le votant prétend être complétement sous contrôle de l'intrus, c'est-à-dire qu'il révèle toutes ces données privées et suit les instructions de l'intrus. Ce cas s'appelle « Swap-Coercion-Resistance », noté « $SwCR$ ».

   Intuitivement « Swap-Coercion-Resistance » est plus fort que « Swap-Receipt-Freeness », qui est plus fort que « Swap-Vote-Privacy » ($SwCR > SwRF > SwVP$).
2. *Intrus interne ou externe :* L'intrus peut contrôler un autre votant légitime (ni le votant ciblé, ni le votant de compensation). Dans ce cas il peut par exemple essayer de copier le vote du votant ciblé. Dans nos définitions, nous distinguons deux cas :
   (a) L'intrus est externe, noté « O »
   (b) L'intrus est interne, noté « I »

   Intuitivement le deuxième cas est plus fort ($I > O$).
3. *Protection contre l'abstention forcée :* Un protocole peut permettre à un votant de voter, même si l'intrus essaye de l'en empêcher. Contrairement à la littérature [JCJ05, BHM08], nous définissions cette propriété indépendamment de la résistance à la coercition. Nous exigeons l'équivalence observationnelle
   (a) dans tous les cas, même si un votant est forcé à s'abstenir. Ce cas s'appelle « security against Forced-Abstention-Attacks », noté « FA ».

(b) seulement si le votant ciblé ne s'abstient pas. Ce cas s'appelle « Participation Only », noté « PO ».

Dans cette dimension « security against Forced-Abstention-Attacks » est plus fort que « Participation Only » ($FA > PO$).

La propriété la plus forte est donc $SwCR^{I,FA}$, la plus faible $SwVP^{O,PO}$. Toute la hiérarchie des notions est présentée dans la figure 3.1. De plus, nous avons appliqué notre définition modulaire dans plusieurs études de cas, et les résultats sont résumés dans la table 3.1.

6.3.2 — Votes pondérés

Les définitions présentées dans la section précédente sont basées sur l'idée de permutation de votes. Cela les rend inadaptées pour une situation où les votes sont pondérés, comme dans une société anonyme, où les votes sont proportionnels aux nombres d'actions de chaque votant. Considérons l'exemple suivant : Alice a 50% des actions, et Bob et Carol 25% chacun. Les cas où Alice et Bob permutent leurs votes sont donc facilement distinguables si par exemple Carol vote « Oui » tout le temps, car le résultat est différent : 75% ou 50% pour oui. Nous notons qu'il existe encore des situations qui donnent le même résultat : par exemple on obtient 50% « Oui » si Alice vote « Oui » et Bob et Carol « Non », ou si Alice vote « Non » et Bob et Carol votent « Oui ».

Afin de résoudre ce problème, nous proposons une généralisation qui prend en compte les votes pondérés : au lieu d'exiger l'équivalence observationnelle pour deux situations où deux votants ont permutés leurs votes, nous exigeons l'équivalence observationnelle de toutes les situations qui publient le même résultat, indépendamment de la distribution des votes. Nous démontrons aussi que cette définition est équivalente à la précédente si les votes ont tous le même poids.

Comme étude de cas nous utilisons une variante du protocole d'Eliasson et Zúquete [EZ06], et proposons une preuve partiellement automatisée avec ProVerif.

Dans une deuxième généralisation des définitions précédentes, nous proposons des définitions d'absence de reçu et de résistance à la coercition où plusieurs votants sont sous attaque par l'intrus. Nous démontrons ensuite que cette généralisation est équivalente à la définition précédente si le protocole est *modulaire* (c'est-à-dire qu'on peut composer et décomposer des instances) et *fini* (c'est-à-dire que les processus sont finis). De plus, sous les mêmes hypothèses, nous démontrons que les cas d'un intrus interne et externe sont équivalentes (c'est-à-dire que la corruption des votants ne rapporte rien à l'intrus pour un protocole avec ces propriétés). Nous prouvons aussi que les protocoles de Fujioka et al. [FOO92], d'Okamoto [Oka96] et Bingo Voting [BMQR07] satisfont ces hypothèses. Pour ces réductions nous utilisons le résultat d'annulation du chapitre 2.

Tout cela amène à une grande hiérarchie de notions, qui est résumée dans les

figures 3.2 et 3.3.

6.4 Les protocoles de vente aux enchères

Les ventes aux enchères représentent une méthode simple pour la vente de biens ou de services : un *vendeur* propose un bien ou un service, et les *enchérisseurs* soumettent des offres.

Dans la littérature beaucoup de propriétés de sécurité ont été proposées, ici nous analysons les suivantes en détail :

— **Équité :** D'abord un protocole de vente aux enchères scellées ne devrait divulguer aucune information sur les autres enchérisseurs et leurs offres jusqu'à la fin de la phase de soumission des offres, pour empêcher des tactiques non équitables. Nous appelons cette propriété « *Weak* » ou « *Strong Noninterference* », selon que le nombre de participants est divulgué ou pas. De plus, un protocole ne devrait pas permettre à un participant malhonnête de gagner la compétition même qu'il n'a pas soumis l'offre la plus élevée. Cette propriété s'appelle « *Highest Price Wins* ».

— **Authentification** : Du point de vue du vendeur il est important d'assurer la *non-répudiation*, c'est-à-dire qu'une fois le gagnant annoncé, l'enchérisseur ne peut pas nier qu'il a soumis l'offre gagnante. De plus, dans certains systèmes d'enchère, il est interdit d'annuler une offre – il faut donc exiger la *non-annulation*.

— **Respect de la vie privée :** Dans une vente aux enchères scellées le secret des offres est une propriété bien connue, et l'anonymat des acheteurs est souvent souhaité également. Nous distinguons plusieurs notions : « *Secrecy of Bids* », « *Bidding-Price Unlinkability* », « *Weak Anonymity* » et « *Strong Anonymity* », « *Receipt-Freeness* » et « *Coercion-Resistance* ». « Secrecy of Bids » garantit que les offres perdantes restent secrètes. Dans le cas de « Bidding-Price Unlinkability » une liste des offres peut être publique, mais le lien entre la liste et les enchérisseurs perdants doit rester secret. « Strong Anonymity » implique que tous les participants (y compris le gagnant) restent anonymes, et que les offres perdantes restent secrètes. Dans le cas de « Weak Anonymity » la liste des offres peut également être publique, mais le lien avec les enchérisseurs (anonymes) reste secret. « Receipt-Freeness » assure qu'un enchérisseur peut créer des faux reçus pour prouver qu'il a enchérit une offre particulière, et « Coercion-Resistance » implique que les enchérisseurs peuvent résister à la coercition.

— **Vérifiabilité** : Un protocole vérifiable permet aux enchérisseurs et au vendeur de vérifier que le gagnant et le prix ont été correctement calculés, et qu'il n'y

avait pas d'offre illégitime soumise par un extérieur.

Nous proposons des définitions formelles de toutes les propriétés d'équité, d'authentification et de respect de la vie privée dans le π-calcul appliqué. Pour les notions de respect de la vie privée nous obtenons une hiérarchie, qui est résumée dans la figure 4.3.

Pour la définition de la vérifiabilité nous utilisons une approche différente : dans un modèle abstrait nous définissons les tests nécessaires pour vérifier le bon déroulement du protocole et les conditions que ces tests doivent satisfaire. Un protocole est donc vérifiable s'il possède tous les tests, et si les tests satisfont les conditions. Ce modèle peut – par exemple – s'instancier dans le π-calcul appliqué.

Nous appliquons toutes les définitions précédentes avec ProVerif sur trois études de cas : les protocoles de Curtis et al. [CPS07], de Brandt [Bra06], et de Sako [Sak00]. Seulement le protocole de Sako s'avère sûr, les deux autres contiennent plusieurs faiblesses, essentiellement liées à des problèmes d'authentification.

La définition de vérifiabilité peut aussi s'instancier dans le modèle calculatoire. Pour illustrer cela, nous prouvons que le protocole par Sako assure la vérifiabilité aussi dans le modèle calculatoire. Pour ce faire nous utilisons majoritairement CryptoVerif, mais aussi une preuve manuelle car une limite de CryptoVerif ne permet pas de finir cette preuve automatiquement.

Le but de la vérifiabilité est de permettre aux utilisateurs de vérifier le bon déroulement du protocole. En même temps, pour assurer cette propriété, les protocoles utilisent des opérations cryptographiques complexes. Ces opérations sont très puissantes, mais souvent trop complexes pour être vraiment comprises par un utilisateur lambda, qui doit donc faire confiance aux experts qui ont développé (et peut-être vérifié formellement) la procédure de vérification. Pour proposer une solution à cette problématique, nous avons développé deux protocoles d'« enchères vraiment vérifiable par les enchérisseurs », c'est-à-dire des protocoles qui garantissent la vérifiabilité sans utiliser la cryptographie complexe – mais basés sur des objets et propriétés physiques.

Le premier protocole, une variante du protocole de Sako basé sur du carton et des enveloppes, s'appelle « Cardako[2] ». Il peut être réalisé entièrement avec du matériel de bureau. Le deuxième protocole s'appelle « Woodako[3] ». Il s'agit aussi d'une variante du protocole de Sako, mais implémenté avec une boîte en bois, qui permet de désigner le gagnant de manière privée et vérifiable. Pour consolider notre affirmation que ces protocoles sont effectivement sûrs et vérifiables, nous les analysons formellement en ProVerif en utilisant les mêmes définitions que pour les protocoles cryptographiques. Pour modéliser les objets physiques et leurs

[2] **Card**board S**ako**
[3] **Wood**en Box S**ako**

propriétés nous utilisons une théorie équationnelle particulière, ce qui permet de conclure que les protocoles sont sûrs.

6.5 Conclusion

Nous avons introduit notre travail et son contexte dans le chapitre 1.

Dans le chapitre 2, nous avons rappelé le π-calcul appliqué, sa syntaxe et sa sémantique. Nous avons identifié deux sous-classes de processus : des processus *normés*, et des processus *finis*. Nous avons démontré que tout processus normé P peut se réécrire de manière unique (modulo $\sim_l$) comme composition parallèle de facteurs premiers : $P \sim_l P_1|\ldots|P_n$. Chaque facteur est premier dans le sens où il ne peut pas se décomposer en facteurs non-triviaux. De plus, nous avons démontré que tout processus fini peut se décomposer de manière unique (modulo $\approx_l$) en processus premiers, c'est-à-dire que $P \approx_l P_1|\ldots|P_n$.

Dans la suite, chapitre 3, nous nous sommes intéressées au vote électronique. Nous avons proposé une taxonomie du respect de la vie privée dans le π-calcul appliqué. Elle prend en compte différents pouvoirs de l'intrus (interne ou externe), des attaques spécifiques (comme l'abstention forcée), et des niveaux de coercition différents (secret du vote, absence de reçu, résistance à la coercition). Comme études de cas, nous avons analysé les protocoles de Fujioka et al. [FOO92], Okamoto [Oka96], Lee et al. [LBD+03] et Bingo Voting [BMQR07]. Le protocole de Fujioka et al. garantit le secret du vote pour un intrus interne, mais est vulnérable aux attaques d'abstention forcée. Bien qu'il reste vulnérable aux attaques d'abstention forcée, le protocole d'Okaomoto garantit l'absence de reçu. Les deux protocoles peuvent être sécurisés contre les attaques d'abstention forcée avec un canal privé et un administrateur honnête. Le protocole de Lee et al. assure la résistance à la coercition, mais seulement contre un intrus externe, et reste vulnérable aux attaques d'abstention forcée. Finalement Bingo Voting garantit la résistance à la coercition pour un intrus interne et est protégé contre les attaques d'abstention forcée, si la liste des participants n'est pas publiée.

Dans la deuxième partie du chapitre, nous avons développé des notions généralisées pour prendre en compte des votes pondérés. Nous avons proposé des définitions du secret du vote, de l'absence de reçu et de la résistance à la coercition, avec et sans votants corrompus par l'intrus. Pour l'absence de reçu et la résistance à la coercition nous avons aussi développé des notions où un seul ou plusieurs votants sont sous attaque. De plus, nous avons établi des liens formels avec les définitions précédentes de notre taxonomie. Nous avons analysé une variante du protocole par Fujioka et al. comme étude de cas, et prouvé que si un protocole est fini et respecte une condition de modularité, les cas avec un seul ou plusieurs votants sous attaque sont équivalents. Nous avons obtenu un résultat similaire

pour les cas avec et sans votants corrompus.

Dans le chapitre 4 nous avons analysé les protocoles de vente aux enchères. Après avoir proposé notre modèle, nous avons défini plusieurs notions d'équité, d'authentification et de respect de la vie privée : non-répudiation, non-annulation, « Weak » et « Strong Non-Interference », « Highest Price Wins » et différentes notions de respect de la vie privée. Ces notions permettent de capturer des protocoles où le gagnant reste anonyme ou non, et où les offres sont secrètes ou juste anonymes. De même, nous avons proposé des notions pour des protocoles qui ne sont pas des protocoles d'enchère où le gagnant paye le prix de l'offre la plus élevée (enchères au premier prix), mais par exemple le prix de l'offre la deuxième plus élevée (enchères au deuxième prix). Comme études de cas, nous avons discuté les protocoles de Curtis et al. [CPS07], Brandt [Bra06] et Sako [Sak00]. Le protocole de Sako garantit toutes les propriétés sauf l'absence de reçu et la résistance à la coercition. Pour les deux autres protocoles, nous avons identifié plusieurs faiblesses, essentiellement liés à des problèmes d'authentification.

Dans la suite de ce chapitre, nous avons analysé la vérifiabilité pour les protocoles d'enchère. Nous avons proposé une définition abstraite de la vérifiabilité qui peut s'instancier dans le modèle symbolique et calculatoire. Pour illustrer cela, nous avons complété une étude du protocole de Sako dans les deux modèles en utilisant ProVerif et CryptoVerif (et des preuves manuelles). De plus nous avons aussi analysé le protocole de Curtis et al. en ProVerif, et identifié plusieurs faiblesses.

Finalement nous avons exploré le concept des « enchères vraiment vérifiables par les enchérisseurs », c'est-à-dire des protocoles qui peuvent être vérifiés sans connaissances spéciales. Pour cela nous avons proposé deux protocoles inspirés du protocole de Sako : « Cardako » et « Woodako ». Cardako n'utilise que du matériel de bureau, c'est-à-dire du carton et des enveloppes. Woodako est basé sur une boîte en bois qui permet de déterminer le gagnant de manière privée et vérifiable. Nous analysons les deux en ProVerif en utilisant des théories équationnelles particulières pour modéliser le comportement des objets physiques (les enveloppes, la boîte en bois). Cela permet d'appliquer les mêmes définitions que celles appliquées aux protocoles cryptographiques.

6.5.1 — Perspectives

Par rapport aux résultats de décomposition unique, nous avons identifié deux pistes principales. La première consiste à étendre l'approche aux processus infinis. Un premier résultat dans cette direction a été obtenu par Hirschkoff et Pous [HP10] pour un sous-ensemble de CCS avec de la réplication uniquement niveau le plus haut. Ils ont introduit le concept de *seed*, c'est-à-dire le processus bisimilaire au processus initial de taille minimale, mais avec un nombre maximal de processus

sous réplication. Ils ont prouvé que cette représentation est unique, et que ce résultat se généralise au π-calcul sans restriction (sans ν). Il reste ouvert si un résultat similaire peut être obtenu pour le π-calcul intégral.

Une deuxième piste consiste à proposer un algorithme efficace qui calcule la décomposition unique d'un processus. Il n'est pas certain qu'un tel algorithme existe, car simplement déterminer si un processus est fini peut être non-trivial.

De plus, il reste deux autres résultats à démonter : il reste à prouver que la bisimulation forte est clos sous l'application de contexte (comme son pendant faible), et qu'on peut remplacer des processus contenant la synchronisation avec des processus bisimilaires, et que le résultat reste bisimilaire.

Par rapport aux résultats concernant la vie privée dans les protocoles de vote nous avons identifié deux pistes. La première consiste à automatiser nos preuves (qui pour la plupart étaient manuelles). Pour cela les défis principaux sont des théories équationnelles complexes, et le contexte qui représente l'intrus dans la définition de la résistance à la coercition. La deuxième piste est la traduction de nos définitions symboliques dans le modèle calculatoire, pour par exemple pouvoir raisonner sur la probabilité qu'un intrus devine correctement le vote du votant ciblé.

Concernant nos résultats sur les ventes aux enchères, nous voyons plusieurs pistes. Une première idée est d'essayer de corriger les problèmes identifiés avec les protocoles de Curtis et al. et Brandt. Cela implique de surmonter certaines limites de ProVerif, et de préciser le modèle utilisé pour le protocole de Brandt.

Dans nos applications ProVerif était souvent incapable de prouver le cas général directement, ce qui a nécessité des preuves de généralisation manuelles. Une extension de ProVerif permettant de traiter des boucles et de l'état global est donc nécessaire.

De plus, aucun de nos protocoles ne garantit l'absence de reçu ou la résistance à la coercition. Il serait intéressant d'analyser un tel protocole, comme celui d'Abe et Suzuki [AS02].

Pour renforcer notre preuve calculatoire de vérifiabilité pour le protocole de Sako, il serait intéressant de remplacer la primitive de chiffrement abstraite par le chiffrement d'ElGamal concret.

Finalement, nous aimerions développer des protocoles d'« enchères vraiment vérifiables par les enchérisseurs » qui passent facilement à l'échelle. De plus, la modélisation formelle des propriétés physiques soulève encore beaucoup de questions : comment être sûr que le modèle est assez précis et permet de détecter toutes les attaques ?

Par ailleurs, dans [DDL13] nous avons proposé une analyse détaillée du protocole de Brandt. Bien que cela ne soit pas présenté dans cette thèse, il serait intéressant faire une preuve calculatoire démontrant que les corrections que nous

avons proposées sont suffisantes, et que le protocole est maintenant sûr.

List of Tables

2.1 Summary of unique factorization results for the Applied π-Calculus 45

3.1 Results of the case studies 114

List of Figures

2.1 Grammar for terms 16
2.2 Grammar for Plain Processes 17
2.3 Grammar for Extended Processes 17
2.4 Structural Equivalence 19
2.5 Internal Reduction 19
2.6 Labeled semantics 20
2.7 Channel/Link Passing in the Applied π-Calculus 25
2.8 Scope extrusion in the Applied π-Calculus 25

3.1 Hierarchy of privacy notions with examples 69
3.2 Hierarchy of generalized privacy notions 150
3.3 Links between the generalized and swap-based privacy notions . . 151

4.1 Relations among the privacy notions 169
4.2 Relations among the privacy notions including corrupted bidders . 169
4.3 Relations between the privacy notions with and without corrupted bidders 170
4.4 Cardako bid for price p_{m-1}. 243
4.5 Cardako bidding envelope. 243
4.6 The Woodako prototype. 249
4.7 Inside our Woodako prototype, where layers $L0$ and $L1$ are removed. 250
4.8 The Woodako box after the bid of bidder number two. 253
4.9 The Woodako box after two prices have been tested. 254
4.10 Bidder verifiability (i.e. view from top). 255
4.11 Seller verifiability (i.e. view from bottom). 255

Bibliography

[AAA+12] Alessandro Armando, Wihem Arsac, Tigran Avanesov, Michele Barletta, Alberto Calvi, Alessandro Cappai, Roberto Carbone, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Gabriel Erzse, Simone Frau, Marius Minea, Sebastian Mödersheim, David von Oheimb, Giancarlo Pellegrino, Serena Elisa Ponta, Marco Rocchetto, Michaël Rusinowitch, Mohammad Torabi Dashti, Mathieu Turuani, and Luca Viganò. The avantssar platform for the automated validation of trust and security of service-oriented architectures. In *Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, volume 7214 of *LNCS*, pages 267–282, Tallinn, Estonia, 2012. Springer. 10

[AB05a] Martín Abadi and Bruno Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. *Journal of the ACM*, 52(1):102–146, January 2005. 11

[AB05b] Xavier Allamigeon and Bruno Blanchet. Reconstruction of Attacks against Cryptographic Protocols. In *Proceedings of the 18th IEEE Computer Security Foundations Workshop (CSFW-18)*, pages 140–154. IEEE Computer Society, June 2005. 11

[ABB+05] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuellar, Paul Hankes Drielsma, Pierre-Cyrille Heám, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05)*, pages 281–285, Berlin, Heidelberg, 2005. Springer-Verlag. 10

[ABF07] Martín Abadi, Bruno Blanchet, and Cédric Fournet. Just fast keying in the pi calculus. *ACM Transactions on Information and System Security (TISSEC)*, 10(3):1–59, July 2007. 165

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In *Proceedings of the 7th Theory of Cryptography Conference (TCC'10)*, volume 5978 of *LNCS*, pages 480–497. Springer, 2010. 211, 236

[AC04] Alessandro Armando and Luca Compagna. Satmc: a sat-based model checker for security protocols. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA'04)*, volume 3229 of *LNAI*, pages 730–733. Springer-Verlag, September 2004. 10

[ACKR13] Myrto Arapinis, Véronique Cortier, Steve Kremer, and Mark Ryan. Practical everlasting privacy. In *Proceedings of the 2nd Conference on Principles of Security and Trust (POST'13)*, volume 7796 of *LNCS*, pages 21–40, Rome, Italy, 2013. Springer. 54

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *Proceedings of the 17th Usenix Security Symposium*, pages 335–348. USENIX Association, 2008. 6

[ADMPQ09] Ben Adida, Olivier De Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: analysis of real-world use of helios. In *Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE'09)*. USENIX Association, 2009. 117

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, New York, 2001. ACM. 9, 16, 19, 21, 22, 282

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: the spi calculus. In *Proceedings of the 4th ACM conference on Computer and communications security (CCS '97)*, pages 36–47, New York, NY, USA, 1997. ACM. 9

[Ama13] Amazon.com. Form 10-k, annual report, filing date jan 30, 2013. Available at `http://pdf.secdatabase.com/1562/0001193125-13-028520.pdf`. Retrieved 29/08/2013., Januar 2013. 6, 278

[AR00] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Proceedings of the International Conference IFIP on Theoretical Computer Science (TCS'00)*, pages 3–22, London, UK, 2000. Springer-Verlag. 9

[ARR11] Myrto Arapinis, Eike Ritter, and Mark D. Ryan. Statverif: Verification of stateful processes. In *Proceedings of the IEEE 24th Computer Security Foundations Symposium (CSF'11)*, pages 33–47, Washington, DC, USA, 2011. IEEE Computer Society. 268

[AS02] Masayuki Abe and Koutarou Suzuki. Receipt-free sealed-bid auction. In *Proceedings of the 5th International Conference on Information Security (ISC'02)*, volume 2433 of *LNCS*, pages 191–199. Springer-Verlag, 2002. 157, 160, 275, 291

[BAF08] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, February–March 2008. 11

[BAN90] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990. 6, 9, 278

[BBF+11] Jesper Bengtson, Karthikeyan Bhargavan, Cédric Fournet, Andrew D. Gordon, and Sergio Maffeis. Refinement Types for Secure Implementations. *ACM Transactions on Programming Languages and Systems*, 33(2):8:1–8:45, February 2011. 9

[BBU13] Michael Backes, Fabian Bendun, and Dominique Unruh. Computational soundness of symbolic zero-knowledge proofs: Weaker assumptions and mechanized verification. In *Proceedings of the Second International Conference Principles of Security and Trust (POST'13)*, volume 7796 of *LNCS*, pages 206–225. Springer, 2013. 9

[BC12] Bruno Blanchet and David Cadé. *CryptoVerif – Computationally Sound, Automatic Cryptographic Protocol Verifier – User Manual*. INRIA Paris-Rocquencourt, France, October 2012. Available at `http://prosecco.gforge.inria.fr/personal/bblanche/cryptoverif/`. 231

[BCD+09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam

Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In *Proceedings of the 13th International Conference on Financial Cryptography and Data Security (FC'09)*, volume 5628 of *LNCS*, pages 325–343. Springer, 2009. 157

[BCP+11] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting helios for provable ballot privacy. In *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS'11)*, volume 6879 of *LNCS*, pages 335–354, Leuven, Belgium, 2011. Springer. 6, 52, 53

[BCPW12] David Bernhard, Véronique Cortier, Olivier Pereira, and Bogdan Warinschi. Measuring vote privacy, revisited. In *ACM Conference on Computer and Communications Security (CCS'12)*, pages 941–952. ACM, 2012. 52

[BCSS11] David Basin, Srdjan Capkun, Patrick Schaller, and Benedikt Schmidt. Formal Reasoning about Physical Properties of Security Protocols. *ACM Transactions on Information and System Security*, pages 1–28, 2011. 162

[BDK07] Michael Backes, Markus Dürmuth, and Ralf Küsters. On simulatability soundness and mapping soundness of symbolic cryptography. In *Proceedings of the 27th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *LNCS*, pages 108–120. Springer, 2007. 9

[BDKL10] Gilles Barthe, Marion Daubignard, Bruce M. Kapron, and Yassine Lakhnech. Computational indistinguishability logic. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*, pages 375–386. ACM, 2010. 11

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *Proceedings of the 18th Annual International Cryptology Conference (CRYPTO'98)*, pages 26–45, London, UK, 1998. Springer-Verlag. 231

[Ben96] Josh Daniel Cohen Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, December 1996. 161

[BFG10] Karthikeyan Bhargavan, Cédric Fournet, and Andrew D. Gordon. Modular verification of security protocol code by typing. In *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'10)*, pages 445–456, 2010. 9

[BFK+13] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, and Pierre-Yves Strub. Implementing tls with verified cryptographic security. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'13)*, pages 445–459, 2013. 7, 279

[BGHZB11] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella-Béguelin. Computer-aided security proofs for the working cryptographer. In *Proceedings of the 31st Annual Cryptology Conference (CRYPTO'11)*, volume 6841 of *LNCS*, pages 71–90, Santa Barbara, CA, USA, 2011. Springer. 11

[BGZB09] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella-Béguelin. Formal certification of code-based cryptographic proofs. In *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'09)*, pages 90–101. ACM, 2009. 11

[BHKO04] Yohan Boichut, Pierre-Cyrille Héam, Olga Kouchnarenko, and Frédéric Oehl. Improvements on the genet and klay technique to automatically verify security protocols. In *Proceedings of the Workshop on Automated Verification of Infinite States Systems (AVIS'04), colocated with ETAPS'04*, 2004. 10

[BHM08] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proceedings of the Computer Security Foundations Symposium (CSF'08)*, pages 195–209. IEEE Computer Society, 2008. 50, 54, 61, 285

[Bla01] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proceedings of the 14th Computer Security Foundations Workshop (CSFW'14)*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001. IEEE Computer Society. 10, 158

[Bla02] Bruno Blanchet. From Secrecy to Authenticity in Security Protocols. In *Proceedings of the 9th Static Analysis Symposium (SAS'02)*, volume 2477 of *LNCS*, pages 342–359, Madrid, Spain, September 2002. Springer Verlag. 11

[Bla03] Matt Blaze. Rights amplification in master-keyed mechanical locks. *IEEE Security & Privacy*, 1(2):24–32, 2003. 162

[Bla04a] Bruno Blanchet. Automatic Proof of Strong Secrecy for Security Protocols. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'04)*, pages 86–100, Oakland, California, May 2004. 11

[Bla04b] Matt Blaze. Safecracking for the computer scientist. Technical report, U. Penn CIS Departement Technical Report, December 2004. Available at `http://www.crypto.com/papers/safelocks.pdf`. 162

[Bla06a] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'06)*, pages 140–154, Oakland, California, May 2006. 11, 159, 231

[Bla06b] Matt Blaze. Towards a broader view of security protocols. In *Proceedings of the Security Protocols Workshop 2004*, volume 3957 of *LNCS*, pages 106–120. Springer Verlag, 2006. 162

[Bla07] Bruno Blanchet. Computationally sound mechanized proofs of correspondence assertions. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF'07)*, pages 97–111, Venice, Italy, July 2007. IEEE. 11

[Bla08] Bruno Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Transactions on Dependable and Secure Computing*, 5(4):193–207, October–December 2008. Special issue IEEE Symposium on Security and Privacy 2006. 11

[Bla13] Bruno Blanchet. Re: Lien exact entre proverif et le pi-calcul applique [e-mail]. Personal Communication, May 2013. 63, 235

[BMQR07] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In *E-Voting and Identity*, volume 4896 of *LNCS*, pages 111–124. Springer, 2007. 12, 50, 52, 54, 69, 88, 114, 152, 272, 280, 286, 289

[BMU12] Michael Backes, Ankit Malik, and Dominique Unruh. Computational soundness without protocol restrictions. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'12)*, pages 699–711, Raleigh, NC, USA, 2012. ACM. 9

[BMV05] David A. Basin, Sebastian Mödersheim, and Luca Viganò. Ofmc: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, 2005. 10

[BP05] Michael Backes and Birgit Pfitzmann. Relating symbolic and cryptographic secrecy. *IEEE Transactions on Dependable and Secure Computing*, 2(2):109–123, 2005. 9

[BP06] Bruno Blanchet and David Pointcheval. Automated security proofs with sequences of games. In *Proceedings of the 26th Annual International Cryptology Conference (CRYPTO'06)*, volume 4117 of *LNCS*, pages 537–554, Santa Barbara, CA, USA, August 2006. Springer. 11

[BP09] Jens-Matthias Bohli and Andreas Pashalidis. Relations among privacy notions. In *Proceedings of the 13th International Conference on Financial Cryptography and Data Security (FC'09)*, pages 362–380. Springer-Verlag, 2009. 51

[BPW06] Michael Backes, Birgit Pfitzmann, and Michael Waidner. Soundness limits of dolev-yao models. In *Workshop on Formal and Computational Cryptography (FCC'06), affiliated with ICALP'06*, June 2006. 9

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *Proceedings of the 18th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'12)*, volume 7658 of *LNCS*, pages 626–643, Beijing, China, 2012. Springer. 6

[Bra02] Felix Brandt. A verifiable, bidder-resolved auction protocol. In *Proceedings of the 5th AAMAS Workshop on Deception, Fraud and Trust in Agent Societies (Special Track on Privacy and Protection with Multi-Agent Systems)*, pages 18–25, 2002. 159

[Bra06] Felix Brandt. How to obtain full privacy in auctions. *International Journal of Information Security*, 5:201–216, 2006. 13, 14, 155, 158, 159, 160, 162, 194, 202, 212, 213, 242, 267, 273, 275, 281, 288, 290

[BS08] Felix Brandt and Tuomas Sandholm. On the existence of unconditionally privacy-preserving auction protocols. *ACM Transactions on Information and System Security*, 11:6:1–6:21, 2008. 159

[BSC13] Bruno Blanchet, Ben Smith, and Vincent Cheval. *ProVerif – Automatic Cryptographic Protocol Verifier – User Manual and Tutorial*. INRIA Paris-Rocquencourt, Paris, France, 1.87beta6 edition, March 2013. Available at `http://prosecco.gforge.inria.fr/personal/bblanche/proverif/manual.pdf`. 195

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the 26th Symposium on Theory of Computing (STOC '94)*, pages 544–553, New York, NY, USA, 1994. ACM. 52, 160, 161

[Bun09] Bundesverfassungsgericht (Germany's Federal Constitutional Court). Use of voting computers in 2005 bundestag election unconstitutional. Press release 19/2009 `http://www.bundesverfassungsgericht.de/en/press/bvg09-019en.html`, March 2009. 7, 49, 241, 279

[Cac99] Christian Cachin. Efficient private bidding and auctions with an oblivious third party. In *Proceedings of the 6th Conference on Computer and Communications Security (CCS'99)*, pages 120–127. ACM Press, 1999. 157

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS'01)*, pages 136–145, Las Vegas, Nevada, USA, 2001. IEEE Computer Society. 51

[CB13a] David Cadé and Bruno Blanchet. Proved generation of implementations from computationally-secure protocol specifications. In *Proceedings of the 2nd Conference on Principles of Security and Trust (POST'13)*, volume 7796 of *LNCS*, pages 63–82, Rome, Italy, March 2013. Springer. 11

[CB13b] Vincent Cheval and Bruno Blanchet. Proving more observational equivalences with ProVerif. In *Proceedings of the 2nd Conference on Principles of Security and Trust (POST'13)*, volume 7796 of *LNCS*, pages 226–246, Rome, Italy, March 2013. Springer. 11

[CCK12] Rohit Chadha, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocols. In *Proceedings of the 21st European Symposium on Programming Languages and Systems (ESOP'12)*, volume 7211 of *LNCS*, pages 108–127, Tallinn, Estonia, 2012. Springer. 11, 75, 152, 274

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'08)*, volume 0, pages 354–368, Los Alamitos, CA, USA, 2008. IEEE Computer Society. 52

[CDK12] Ştefan Ciobâcă, Stéphanie Delaune, and Steve Kremer. Computing knowledge in security protocols under convergent equational theories. *Journal of Automated Reasoning*, 48(2):219–262, 2012. 11, 86, 152, 274

[CF85] Josh Daniel Cohen and Michael John Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS'85)*, pages 372–382, Portland, Oregon, USA, October 1985. IEEE Computer Society. 52, 161

[CG96] Ran Canetti and Rosario Gennaro. Incoercible multiparty computation (extended abstract). In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS'96)*, pages 504–513, 1996. 51

[CH88] Thierry Coquand and Gérard P. Huet. The calculus of constructions. *Information and Computation*, 76(2/3):95–120, 1988. 11

[Cha04] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004. 162, 241

[Chr93] Søren Christensen. *Decidability and Decompostion in Process Algebras*. PhD thesis, School of Computer Science, University of Edinburgh, 1993. 16, 24

[Cit13] Citizens Information Board. Proportional representation. `http://www.citizensinformation.ie/en/government_in_ireland/elections_and_referenda/voting/proportional_representation.html`, 2013. Retrieved 24/10/2013. 55, 161

[CK94] Claude Crépeau and Joe Kilian. Discreet solitary games. In *Proceedings of the 13th Annual International Cryptology Conference (CRYPTO'93)*, volume 773 of *LNCS*, pages 319–330. Springer, 1994. 162

[CKW11] Véronique Cortier, Steve Kremer, and Bogdan Warinschi. A survey of symbolic methods in computational analysis of cryptographic systems. *Journal of Automated Reasoning*, 46(3-4):225–259, 2011. 9

[CLC08] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security (CCS'08)*, pages 109–118, Alexandria, Virginia, USA, 2008. ACM. 9

[CLHKS12] Hubert Comon-Lundh, Masami Hagiya, Yusuke Kawamoto, and Hideki Sakurada. Computational soundness of indistinguishability properties without computable parsing. In *Proceedings of the 8th International Conference on Information Security Practice and Experience (ISPEC'12)*, volume 7232 of *LNCS*, pages 63–79, Hangzhou, China, 2012. Springer. 9

[CLK03] Xiaofeng Chen, Byoungcheon Lee, and Kwangjo Kim. Receipt-free electronic auction schemes using homomorphic encryption. In *Proceedings of the 6th Conference on Information Security and Cryptology*, volume 2971 of *LNCS*, pages 259–273. Springer, 2003. 157

[CLN09] Cas J. F. Cremers, Pascal Lafourcade, and Philippe Nadeau. Comparing state spaces in automatic security protocol analysis. In *Formal to Practical Security*, volume 5458 of *LNCS*, pages 70–94, Berlin, Heidelberg, 2009. Springer-Verlag. 11

[CMFP+06] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On some incompatible properties of voting schemes. In *Proceedings of the IAVoSS Workshop on Trustworthy Elections, 2006.*, 2006. 50

[Com12a] Common Criteria for Information Technology Security Evaluation. `http://www.commoncriteriaportal.org/`, 2012. 6, 278

[Com12b] Common Criteria for Information Technology Security Evaluation. Part 3: Security assurance components. `http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R4.pdf`, September 2012. 6, 278

[Coq] The coq proof assistant. `http://coq.inria.fr/`. 11

[CPP13] Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS'13)*, volume 8134 of *LNCS*, pages 481–498, Egham, UK, 2013. Springer. 52

[CPS07] Brian Curtis, Josef Pieprzyk, and Jan Seruga. An efficient eAuction protocol. In *Proceedings of the 7th Conference on Availability, Reliability and Security (ARES'07)*, pages 417–421. IEEE Computer Society, 2007. 13, 155, 158, 159, 162, 194, 212, 213, 236, 267, 273, 281, 288, 290

[Cre08a] Cas J. F. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Proceedings of the 20th International Conference on Computer Aided Verification (CAV'08)*, volume 5123 of *LNCS*, pages 414–418, Princeton, USA, 2008. Springer. 10

[Cre08b] Cas J. F. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security (CCS'08)*, pages 119–128, Alexandria, Virginia, USA, 2008. ACM. 10

[DDL13] Jannik Dreier, Jean-Guillaume Dumas, and Pascal Lafourcade. Brandt's fully private auction protocol revisited. In *Proceedings of the 6th International Conference on Cryptology in Africa (AFRICACRYPT'13)*, volume 7918 of *LNCS*, pages 88–106, Cairo, Egypt, 2013. Springer. 14, 275, 282, 291

[DELL13] Jannik Dreier, Cristian Ene, Pascal Lafourcade, and Yassine Lakhnech. On unique decomposition of processes in the applied pi-calculus. In *Proceedings of the 16th International Conference Foundations of Software Science and Computation Structures (FOSSACS'13)*, volume 7794 of *LNCS*, pages 50–64, Rome, Italy, 2013. Springer. 14, 282

[DHvdG+13] Denise Demirel, Maria Henning, Jeroen van de Graaf, Peter Y. A. Ryan, and Johannes Buchmann. Prêt à voter providing everlasting privacy. In *Proceedings of the 4th International Conference E-Voting and Identify (Vote-ID'13)*, volume 7985 of *LNCS*, pages 156–175, Guildford, UK, 2013. Springer. 49, 284

[DJL13] Jannik Dreier, Hugo Jonker, and Pascal Lafourcade. Defining verifiability in e-auction protocols. In *Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security (ASIACCS'13)*, pages 547–552, Hangzhou, China, 2013. ACM. 14, 282

[DJP11] Naipeng Dong, Hugo Jonker, and Jun Pang. Analysis of a receipt-free auction protocol in the applied pi calculus. In *Proceedings of the*

7th Workshop on Formal Aspects in Security and Trust (FAST'10), volume 6561 of *LNCS*, pages 223–238, Pisa, Italy, 2011. Springer-Verlag. 160, 169, 170, 177, 275

[DKP09] Stéphanie Delaune, Steve Kremer, and Olivier Pereira. Simulation based security in the applied pi calculus. In *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'09)*, volume 4 of *LIPIcs*, pages 169–180, IIT Kanpur, India, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. 52

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17:435–487, December 2009. 49, 50, 54, 55, 56, 59, 60, 63, 66, 67, 68, 71, 73, 77, 103, 104, 105, 149, 201, 284

[DKS10] Stéphanie Delaune, Steve Kremer, and Graham Steel. Formal security analysis of pkcs#11 and proprietary extensions. *Journal of Computer Security*, 18(6):1211–1245, 2010. 6, 278

[DLL11] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Vote-independence: A powerful privacy notion for voting protocols. In *4th Canada-France MITACS Workshop on Foundations and Practice of Security (FPS'11), Revised Selected Papers*, volume 6888 of *LNCS*, pages 164–180, Paris, France, 2011. Springer. 14, 282

[DLL12a] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Defining privacy for weighted votes, single and multi-voter coercion. In *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS'12)*, volume 7459 of *LNCS*, pages 451–468, Pisa, Italy, 2012. Springer. 14, 282

[DLL12b] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. A formal taxonomy of privacy in voting protocols. In *Proceedings of IEEE International Conference on Communications (ICC'12)*, pages 6710–6715, Ottawa, ON, Canada, 2012. IEEE. 14, 282

[DLL13] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Formal verification of e-auction protocols. In *Proceedings of the 2nd Conference on Principles of Security and Trust (POST'13)*, volume 7796 of *LNCS*, pages 247–266, Rome, Italy, 2013. Springer Verlag. 14, 282

[DLMS04] Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, April 2004. 9

[dMPQ07a] Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Simulation-based analysis of e2e voting systems. In *First International Conference on E-Voting and Identity (VOTE-ID'07), Revised Selected Papers*, volume 4896 of *LNCS*, pages 137–149, Bochum, Germany, 2007. Springer. 52

[dMPQ07b] Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Simulation-based analysis of e2e voting systems. In *Frontiers of Electronic Voting*, volume 07311 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007. 52

[Don12] John Donahoe. To our stockholders. `http://files.shareholder.com/downloads/ebay/2534998457x0x651324/1743d367-5b70-42c5-b1c3-57dfc4981443/eBay_ShareholderLetter2012.pdf`, 2012. Retrieved 17/06/2013. i, iii, 6, 7, 278, 279

[DPH11] Trajce Dimkov, Wolter Pieters, and Pieter H. Hartel. Portunes: Representing attack scenarios spanning through the physical, digital and social domain. In *Proceedings of the Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'10)*, volume 6186 of *LNCS*, pages 112–129. Springer Verlag, 2011. 162

[Dre13] Jannik Dreier. The code and scripts used to automatically verify the examples is available at `http://www-verimag.imag.fr/~dreier/papers/thesis-code.zip`, September 2013. 64, 75, 86, 120, 195, 207, 222, 231, 240, 248

[DY81] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols (extended abstract). In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science (FOCS'81)*, pages 350–357, Nashville, Tennessee, USA, October 1981. IEEE Computer Society. 6, 8, 278

[DY83] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983. 6, 8, 9, 278

[eBa13] eBay Inc. Standard selling fees. `http://pages.ebay.com/help/sell/fees.html`, 2013. Retrieved 01/08/2013. 158

[EG82] Shimon Even and Oded Goldreich. On the security of multi-party ping-pong protocols. In *Proceedings of the Annual International Cryptology Conference (CRYPTO'82)*, page 315, Santa Barbara, California, USA, 1982. Plenum Press, New York. 9

[EG85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of the Annual International Cryptology Conference (CRYPTO'84)*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc. 207

[Est] Estonian National Electoral Committee. Internet voting in estonia. `http://www.vvk.ee/public/dok/Internet_Voting_in_Estonia.pdf`. Retrieved 17/06/2013. i, iii, 6, 7, 49, 278, 279

[EZ06] Charlott Eliasson and André Zúquete. An electronic voting system supporting vote weights. *Internet Research*, 16(5):507–518, 2006. 12, 50, 115, 117, 281, 286

[FKS11] Cédric Fournet, Markulf Kohlweiss, and Pierre-Yves Strub. Modular code-based cryptographic verification. In *Proceedings of the 18th ACM conference on Computer and Communications Security (CCS'11)*, pages 341–350, New York, NY, USA, 2011. ACM. 11

[FLA11] Laurent Fousse, Pascal Lafourcade, and Mohamed Alnuaimi. Benaloh's dense probabilistic encryption revisited. In *Proceedings of the 4th International Conference on Cryptology in Africa (AFRICACRYPT'11)*, volume 6737 of *LNCS*, pages 348–362. Springer, 2011. 52

[FLPQ13] Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Robust encryption, revisited. In *Proceedings of the 16th International Conference on Practice and Theory in Public-Key Cryptography (PKC'13)*, volume 7778 of *LNCS*, pages 352–368, Nara, Japan, 2013. Springer. 211

[FNW96] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Communications of the ACM*, 39(5):77–85, 1996. 162

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Proceedings of the Workshop on the Theory and Application of Cryptographic*

Techniques (AUSCRYPT'92), volume 718 of *LNCS*, pages 244–251. Springer Berlin / Heidelberg, Gold Coast, Queensland, Australia, 1992. 12, 50, 54, 68, 69, 114, 115, 117, 152, 160, 272, 280, 286, 289

[FR95] Matthew K. Franklin and Michael K. Reiter. The design and implementation of a secure auction service. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'95)*, pages 2 –14, may 1995. 160

[Fra69] French election law "code électoral" - article l68. Available at `http://www.legifrance.gouv.fr/affichCodeArticle.do?idArticle=LEGIARTI000006353178&cidTexte=LEGITEXT000006070239&dateTexte=20130716`, Mai 1969. Retrieved 16/07/2013. 99

[Gen13] Geneva State Chancellery. The geneva internet voting project. `http://www.ge.ch/evoting/english/doc/final-livret-anglais.pdf`, March 2013. i, iii, 6, 7, 49, 278, 279

[GFN09] Nataliya Guts, Cédric Fournet, and Francesco Zappa Nardelli. Reliable evidence: Auditability by typing. In *Proceedings of the 14th European Symposium on Research in Computer Security (ESORICS'09)*, volume 5789 of *Lecture Notes in Computer Science*, pages 168–183, Saint-Malo, France, 2009. Springer. 161

[GM92] Jan Friso Groote and Faron Moller. Verification of parallel systems via decomposition. In *Proceedings of the Third International Conference on Concurrency Theory (CONCUR'92)*, pages 62–76, London, UK, 1992. Springer-Verlag. 24

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing - Special issue on cryptography*, 17(2):281–308, April 1988. 231

[HP10] Daniel Hirschkoff and Damien Pous. On bisimilarity and substitution in presence of replication. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP'10)*, volume 6199 of *LNCS*, pages 454–465. Springer, 2010. 45, 274, 290

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proceedings of the 19th Annual*

Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT'00), volume 1807 of *LNCS*, pages 539–556. Springer, 2000. 160

[HSD+05] Changhua He, Mukund Sundararajan, Anupam Datta, Ante Derek, and John C. Mitchell. A modular correctness proof of ieee 802.11i and tls. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 2–15, Alexandria, VA, USA, 2005. ACM. 6, 278

[HTK98] Michael Harkavy, J. D. Tygar, and Hiroaki Kikuchi. Electronic auctions with private bids. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, pages 61–74, 1998. 157, 160

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society (WPES'05)*, pages 61–70. ACM, 2005. 50, 52, 54, 61, 116, 285

[Jon09] Hugo Jonker. *Security Matters: Privacy in Voting and Fairness in Digital Exchange*. PhD thesis, Eindhoven University of Technology and University of Luxembourg, 2009. 49, 50, 53, 284

[JZF03] Rui Joaquim, André Zúquete, and Paulo Ferreira. Revs - a robust electronic voting system. In *IADIS International Conference e-Society 2003*, Lisboa, Portugal, 2003. 50, 115, 117

[KHT98] Hiroaki Kikuchi, Michael Harkavy, and J. D. Tygar. Multi-round anonymous auction protocols. In *In Proceedings of the First IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, pages 62–69. Springer-Verlag, 1998. 160

[KL07] Bogdan Księżopolski and Pascal Lafourcade. Attack and revision of electronic auction protocol using ofmc. *Annales UMCS Informatica 2007*, pages 171–183, 2007. 160

[KR03] Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. *Journal of Computer Security*, 11(3):399–429, 2003. 161

[KR05] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *Proceedings of the 14th European Symposium On Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005. 73

[KR11] Simon Kramer and Peter Y. A. Ryan. A modular multi-modal specification of real-timed, end-to-end voter-verifiable voting systems. In *Proceedings of the International Workshop on Requirements Engineering for Electronic Voting Systems (REVOTE'11)*, pages 9–21, 2011. 52

[Kri02] Vijay Krishna. *Auction Theory.* Academic Press, 2002. 157

[KRS10] Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *LNCS*, pages 389–404. Springer, 2010. 50, 161

[KS12] Robert Künnemann and Graham Steel. Yubisecure? formal security analysis results for the yubikey and yubihsm. In *8th International Workshop on Security and Trust Management (STM'12), Revised Selected Papers*, volume 7783 of *LNCS*, pages 257–272, Pisa, Italy, 2012. Springer. 6

[KSR10] Petr Klus, Ben Smyth, and Mark D. Ryan. Proswapper: Improved equivalence verifier for proverif. `http://www.bensmyth.com/proswapper.php`, 2010. 57, 201

[KT09a] Ralf Küsters and Tomasz Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'09)*, pages 251–266. IEEE Computer Society, 2009. 53

[KT09b] Ralf Küsters and Max Tuengerthal. Computational soundness for key exchange protocols with symmetric encryption. In *Proceedings of the 2009 ACM Conference on Computer and Communications Security (CCS'09)*, pages 91–100, Chicago, Illinois, USA, 2009. ACM. 9

[KTV10a] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: definition and relationship to verifiability. In *Proceedings of the 17th Conference on Computer and Communications Security (CCS'10)*, pages 526–535. ACM, 2010. 161

[KTV10b] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A game-based definition of coercion-resistance and its applications. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*. IEEE Computer Society, 2010. 52, 153

[KTV10c] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Proving Coercion-Resistance of Scantegrity II. In *Proceedings of the 12th International Conference on Information and Communications Security (ICICS'10)*, volume 6476 of *LNCS*, pages 281–295. Springer, 2010. 52

[KTV11] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'11)*, pages 538–553. IEEE Computer Society, 2011. 52

[KV09] Francis Klay and Laurent Vigneron. Automatic methods for analyzing non-repudiation protocols with an active intruder. In *Proceedings of the 5th International Workshop on Formal Aspects in Security and Trust (FAST'09)*, pages 192–209. Springer-Verlag, 2009. 161, 167

[LAN02] Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure vickrey auctions without threshold trust. In *Proceedings of the 6th Conference on Financial Cryptography (FC'02)*, volume 2357 of *LNCS*, pages 87–101. Springer, 2002. 157, 159

[LBD+03] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proceedings of the International Conference on Information and Communications Security (ICICS'03)*, volume 2971 of *LNCS*, pages 245–258. Springer Berlin / Heidelberg, 2003. 12, 50, 52, 54, 69, 103, 112, 114, 152, 272, 280, 289

[Liu11] Jia Liu. A proof of coincidence of labeled bisimilarity and observational equivalence in applied pi calculus. Technical Report ISCAS-SKLCS-11-05, Laboratory for Computer Science, Institute of Software, Chinese Academy of Sciences, 2011. Available at `http://lcs.ios.ac.cn/~jliu/`. 22, 45

[Liu13] Jia Liu. Re: Coincidence of labeled bisimilarity and observational equivalence in the applied pi calculus [e-mail]. Personal Communication, June 2013. 45

[LJP10] Lucie Langer, Hugo Jonker, and Wolter Pieters. Anonymity and verifiability in voting: understanding (un)linkability. In *Proceedings of the 12th International Conference on Information and Communications Security (ICICS'10)*, pages 296–310. Springer-Verlag, 2010. 53

[LKM01] Byoungcheon Lee, Kwangjo Kim, and Joongsoo Ma. Efficient public auction with one-time registration and public verifiability. In *Proceedings of the Second International Conference on Cryptology in India: Progress in Cryptology (INDOCRYPT'01)*, pages 162–174. Springer-Verlag, 2001. 160

[LL12] Jia Liu and Huimin Lin. A complete symbolic bisimulation for full applied pi calculus. *Theoretical Computer Science*, 458:76–112, 2012. 22

[Low96] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS'96)*, pages 147–166, London, UK, 1996. Springer-Verlag. 6, 278

[LSB+09] Lucie Langer, Axel Schmidt, Johannes Buchmann, Melanie Volkamer, and Alexander Stolfik. Towards a framework on the security requirements for electronic voting protocols. In *Proceedings of the First International Workshop on Requirements Engineering for e-Voting Systems (RE-VOTE'09)*, pages 61–68, Atlanta, Georgia, USA, 2009. IEEE Computer Society. 49, 53, 284

[LSBV10] Lucie Langer, Axel Schmidt, Johannes Buchmann, and Melanie Volkamer. A taxonomy refining the security requirements for electronic voting: Analyzing helios as a proof of concept. In *Proceedings of the Fifth International Conference on Availability, Reliability and Security (ARES'10)*, pages 475–480, Krakow, Poland, 2010. IEEE Computer Society. 53

[Lut12] Bas Luttik. Unique parallel decomposition in branching and weak bisimulation semantics. Technical report, 2012. Available at http://arxiv.org/abs/1205.2117v1. 24, 25

[LV10] Jing Liu and Laurent Vigneron. Design and verification of a non-repudiation protocol based on receiver-side smart card. *IET Information Security*, 4(1):15 –29, March 2010. 161

[LvO05] Bas Luttik and Vincent van Oostrom. Decomposition orders – another generalisation of the fundamental theorem of arithmetic. *Theoretical Computer Science*, 335(2-3):147–186, 2005. 24, 25

[MCB10] Simon Meier, Cas J. F. Cremers, and David A. Basin. Strong invariants for the efficient construction of machine-checked protocol

security proofs. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 231–245, Edinburgh, United Kingdom, 2010. IEEE Computer Society. 10

[Mil84] Jonathan K. Millen. The interrogator: A tool for cryptographic protocol security. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'84)*, pages 134–141, Oakland, California, USA, 1984. IEEE Computer Society. 6, 278

[Mil89] Robin Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989. 16

[Min08] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties (Netherland's Ministry of the Interior and Kingdom Relations). Stemmen met potlood en papier (voting with pencil and paper), May 2008. 7, 49, 279

[Min13] Ministère des Affaires étrangères. Le vote par internet. `http://www.diplomatie.gouv.fr/fr/vivre-a-l-etranger/vivre-a-l-etranger-vos-droits-et/elections-legislatives-partielles/dates-et-modalites-de-vote/`, 2013. Retrieved 17/06/2013. 7, 49, 279

[MM93] Robin Milner and Faron Moller. Unique decomposition of processes. *Theoretical Computer Science*, 107(2):357–363, 1993. 24

[MN05] Tal Moran and Moni Naor. Basing cryptographic protocols on tamper-evident seals. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 285–297, Lisbon, Portugal, 2005. Springer. 162

[MN06a] Tal Moran and Moni Naor. Polling with physical envelopes: A rigorous analysis of a human-centric protocol. In *Proceedings of the 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'06)*, volume 4004 of *LNCS*, pages 88–108, St. Petersburg, Russia, 2006. Springer. 162

[MN06b] Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Proceedings of the 26th Annual International Cryptology Conference (CRYPTO'06)*, volume 4117 of *LNCS*, pages 373–392. Springer, 2006. 50, 52

[Mol89] Faron Moller. *Axioms for Concurrency*. PhD thesis, School of Computer Science, University of Edinburgh, 1989. 24

[MP13] Catherine Meadows and Dusko Pavlovic. Formalizing physical security procedures. In *Proceedings of the 8th Workshop on Security and Trust Management (STM'12)*, volume 7783 of *LNCS*, pages 193–208. Springer-Verlag, 2013. 162

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part i. *Information and Computation*, 100(1):1–40, 1992. 16

[MR10] Aybek Mukhamedov and Mark D. Ryan. Identity escrow protocol and anonymity analysis in the applied pi-calculus. *ACM Transactions on Information System Security*, 13(41):1–29, 2010. 89

[MSCB13] Simon Meier, Benedikt Schmidt, Cas Cremers, and David A. Basin. The tamarin prover for the symbolic analysis of security protocols. In *Proceedings of the 25th International Conference on Computer Aided Verification (CAV'13)*, volume 8044 of *LNCS*, pages 696–701, Saint Petersburg, Russia, 2013. Springer. 10

[MSS98] John C. Mitchell, Vitaly Shmatikov, and Ulrich Stern. Finite-state analysis of ssl 3.0. In *Proceedings of the 7th USENIX Security Symposium*, pages 16–16, Berkeley, CA, USA, 1998. USENIX Association. 6, 278

[NP97] Moni Naor and Benny Pinkas. Visual authentication and identification. In *Proceedings of the 17th Annual International Cryptology Conference (CRYPTO'97)*, volume 1294 of *LNCS*, pages 322–336. Springer, 1997. 162

[NP00] Uwe Nestmann and Benjamin C. Pierce. Decoding choice encodings. *Information and Computation*, 163(1):1–59, 2000. 18

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 129–139, 1999. 157, 159

[NS78] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978. 8

[NTJ13] V. M. Noufidali, Jobin S. Thomas, and Felix Arokya Jose. E-auction frauds - a survey. *International Journal of Computer Applications*, 61(14):41–45, January 2013. 7, 157, 279

[Oka96] Tatsuaki Okamoto. An electronic voting scheme. In *Proceedings of the IFIP World Conference on IT Tools*, pages 21–30, 1996. 12, 50, 54, 69, 77, 114, 152, 272, 280, 286, 289

[OM01] Kazumasa Omote and Atsuko Miyaji. A practical english auction with one-time registration. In *Proceedings of the 6th Australasian Conference on Information Security and Privacy (ACISP'01)*, volume 2119 of *LNCS*, pages 221–234. Springer, 2001. 157, 159, 160

[Par07] Participants of the Dagstuhl Conference on Frontiers of E-Voting. Dagstuhl accord, 2007. 7, 49, 279

[PH05] Catuscia Palamidessi and Oltea Mihaela Herescu. A randomized encoding of the pi-calculus with mixed choice. *Theoretical Computer Science*, 335(2-3):373–404, 2005. 18

[PSKT01] Carsten Passch, William Song, Weidong Kou, and Chung-Jen Tan. Online auction protocols: A comparative study. In *Proceedings of the Second International Symposium on Topics in Electronic Commerce (ISEC'01)*, pages 170–186, London, UK, 2001. Springer-Verlag. 159

[Reg13] Regierungsrat Basel-Stadt. e-voting: Testbetrieb für elektronisches abstimmen und wählen. `http://www.regierungsrat.bs.ch/staatskanzlei/e-voting.htm`, June 2013. i, iii, 6, 7, 49, 278, 279

[RG95] Todd E. Rockoff and Michael Groves. Design of an internet-based system for remote dutch auctions. *Internet Research*, 5:10–16, 1995. 157

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. 207

[SA99] Frank Stajano and Ross J. Anderson. The cocaine auction protocol: On the power of anonymous broadcast. In *Proceedings of the Third International Workshop on Information Hiding (IH'99)*, volume 1768 of *LNCS*, pages 434–447. Springer, 1999. 159

[Sak00] Kazue Sako. An auction protocol which hides bids of losers. In *Proceedings of the 3rd Workshop on Practice and Theory in Public Key Cryptosystems (PKC'00)*, volume 1751 of *LNCS*, pages 422–432. Springer, 2000. 13, 155, 158, 159, 162, 194, 207, 212, 213, 218, 219, 222, 225, 231, 242, 267, 273, 281, 288, 290

[SAR13] Ben Smyth, Myrto Arapinis, and Mark Ryan. Translating between equational theories for automated reasoning. In *Workshop on Foundations of Computer Security (FCS'13)*, 2013. 152, 274

[SB13] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. In *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS'13)*, volume 8134 of *LNCS*, pages 463–480, Egham, UK, 2013. Springer. 49, 52, 284

[SC11] Ben Smyth and Veronique Cortier. Attacking and fixing helios: An analysis of ballot secrecy. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium (CSF'11)*, pages 297–311. IEEE, 2011. 6, 49, 50, 54, 61, 284

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth. In *Proceedings of the 14th Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT'95)*, volume 921 of *LNCS*, pages 393–403. Springer, 1995. 160, 161

[SMCB12] Benedikt Schmidt, Simon Meier, Cas J. F. Cremers, and David A. Basin. Automated analysis of diffie-hellman protocols and advanced security properties. In *Proceedings of the 25th IEEE Computer Security Foundations Symposium (CSF'12)*, pages 78–94, Cambridge, MA, USA, 2012. IEEE. 10

[Smy11] Ben Smyth. *Formal verification of cryptographic protocols with automated reasoning.* PhD thesis, School of Computer Science, University of Birmingham, 2011. 73, 75

[SRKK10] Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjieh. Towards automatic analysis of election verifiability properties. In *Proceedings of the Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'10)*, volume 6186 of *LNCS*, pages 146–163. Springer, 2010. 161, 165

[SS99] Stuart G. Stubblebine and Paul F. Syverson. Fair on-line auctions without special trusted parties. In *Proceedings of the 3rd Conference on Financial Cryptography (FC'99)*, volume 1648 of *LNCS*, pages 230–240. Springer, 1999. 159, 160

[Sub98] Srividhya Subramanian. Design and verification of a secure electronic auction protocol. In *Proceedings of the 17th IEEE Symposium on*

Reliable Distributed Systems (SRDS'98), pages 204–, Washington, DC, USA, 1998. IEEE Computer Society. 160

[Tur06] Mathieu Turuani. The CL-Atse Protocol Analyser. In *Proceedings of the 17th International Conference on Term Rewriting and Applications (RTA'06)*, volume 4098 of *LNCS*, pages 277–286, Seattle, WA, USA, 2006. 10

[UK 07] UK Electoral Commission. Key issues and conclusions: May 2007 electoral pilot schemes. `http://www.electoralcommission.org.uk/elections/modernising_elections/May2007`, August 2007. Retrieved 19/06/2013. 7, 49, 279

[UMQ10] Dominique Unruh and Jörn Müller-Quade. Universally composable incoercibility. In *Proceedings of the 30th Annual Cryptology Conference (CRYPTO'10)*, volume 6223 of *LNCS*, pages 411–428. Springer, 2010. 51

[VM07] Melanie Volkamer and Margaret McGaley. Requirements and evaluation procedures for evoting. In *Proceedings of the The Second International Conference on Availability, Reliability and Security (ARES'07)*, pages 895–902, Vienna, Austria, 2007. IEEE Computer Society. 51