Secure Multi-Party Computation and Privacy

Aurélien Dupin

THÈSE DE DOCTORAT

CentraleSupélec

Comue Université Bretagne Loire

École Doctorale N°601
Mathématiques et Sciences et Technologies de l'Information et de la Communication

Spécialité : Informatique

Par **Aurélien Dupin**

Secure Multi-Party Computation and Privacy

Calculs Multi-Parties et Vie Privée

Thèse présentée et soutenue à l'École Normale Supérieure de Paris, le 13 juin 2019

Unité de recherche : CIDRe

Thèse N° 2019-05-TH

Composition du jury

Directeurs de thèse :	Christophe Bidan	CentraleSupélec
	David Pointcheval	École Normale Supérieure
Encadrant industriel :	Renaud Dubois	Thales
Rapporteurs :	Sébastien Canard	Orange Labs
	Marine Minier	Université de Lorraine
Président :	Jean-Sébastien Coron	Université du Luxembourg
Examinateurs :	Duong Hieu Phan	Université de Limoges
	Jean-Marc Robert	École de Technologie Supérieure

ACKNOWLEDGEMENT

Tout mémoire de thèse commence par le remerciement de toutes les personnes impliquées dans l'élaboration de ladite thèse. C'est en rédigeant cette partie que je me rends compte que j'ai eu la chance de rencontrer énormément de personnes, avec lesquelles mes échanges ont été des plus enrichissants.

Je commence par remercier mes directeurs de thèse, Christophe Bidan et David Pointcheval, qui ont su me guider tout au long de ces trois dernières années, et sans lesquels je me serai probablement égaré. Apprendre auprès d'eux aura été extrêmement instructif et sympathique.

Je tiens aussi à remercier Renaud Dubois et Éric Garrido de m'avoir donné l'opportunité de réaliser ma thèse dans un cadre industriel. Il s'agissait pour moi d'une condition nécessaire pour me lancer dans un doctorat. Et quel cadre! Le souvenir de ce petit bureau étriqué au milieu d'un couloir avec vue sur la A86 m'emplie déjà de nostalgie!

Je remercie également Marine Minier et Sébastien Canard d'avoir consenti à être rapporteurs et pour leurs remarques pertinentes. Compte tenu de l'épaisseur de ce document, cela n'a pas dû être facile. J'espère que ce manuscrit vous aura intéressé et que sa lecture aura été agréable et instructive. Je remercie également Jean-Sébastien Coron, Duong Hieu Phan et Jean-Marc Robert d'avoir accepté d'être examinateurs pour ma soutenance.

Je remercie particulièrement Jean-Marc Robert, grâce à qui j'ai découvert l'univers de la cryptographie pendant mon master au Canada, et qui a par conséquent énormément influencé mon cursus universitaire et professionnel. Avec du recul, je ne regrette pas ce choix, bien au contraire.

Je souhaite aussi exprimer ma gratitude aux membres et anciens membres de l'équipe chiffre de Thales: Emeline, Renaud, Éric, Ange, Thomas R, Thomas P, Olivier B, Olivier O, Sylvain, David, Mickaël, Philippe, Alexandre, Sonia, Didier, Julien, Simon, Jean-Paul et Matthieu. Ils ont su m'accueillir dans leur équipe et me soutenir durant ces trois années de thèse. Je tiens à remercier tout particulièrement Thomas Prest pour avoir libéré son poste au bon moment –ce qui a largement simplifié ma recherche d'emploi–. Grâce à lui, l'équipe devra encore supporter ma présence pendant quelques années. Toujours grâce à lui, je vais enfin pouvoir profiter de la crème brulée du jeudi midi, et rien que pour ça, ça valait le coup de faire une thèse.

Je remercie également les membres de l'équipe crypto de l'ENS: Romain pour nous avoir affranchis de la notion de matin, Chloé pour son sens de la diplomatie, Mélissa pour m'avoir accompagné sur chaque continent, Balthazar pour ces moments mémorables, Geoffroy pour ses réponses concises, Georg pour avoir été mon meilleur élève, Michele M, Michele O, la com-

munauté italienne pour avoir si peu d'imagination pour les prénoms, Dahmun pour sa capacité à mentir, Brice pour m'avoir offert un voyage en Australie, Michel pour m'avoir permis de participer à l'organisation d'eurocrypt, Adrian –membre émérite de la Houda team–, Louiza pour ... je ne sais quoi mais merci beaucoup, Jérémy, Pooya, Antoine, Anca, Alain, Thierry, Rafael, Florian, Pierre-Alain, Houda, Léo, Hugo, Théo, Thibaut, Léonard, Édouard, Julia, Céline, Quentin, Damien, Fabrice, Azam, Ehsan, Pierrick pour m'avoir fait investir dans le marché de la crevette, mes quarante crevettes pour s'être entredévorées, Nespresso, César, les stagiaires.

Ma gratitude va également aux membres de l'équipe CIDre de CentraleSupélec. J'ai finalement passé très peu de temps avec cette équipe durant ces trois années, mais nos échanges ont toujours été très agréables et instructifs.

Je remercie aussi Megguy et mes parents pour leur soutien indéfectible pendant ces trois années et même avant, et mes plus proches amis –non-crypto– Axel, Antoine, Alexia et Amy de m'avoir permis de m'évader quand j'en avais besoin. Je leur suggère d'ailleurs de ne pas dépasser les remerciements lors de leur lecture. Non Alexia, il n'y a pas d'images.

Enfin, je tiens à remercier sincèrement, chaleureusement, amicalement, spontanément, cordialement, franchement, loyalement, réellement, directement, ouvertement et surtout simplement Alice, Bob, l'adversaire honnête-mais-curieux et la majorité honnête qui m'auront accompagné tout au long de cette thèse.

Si votre nom n'apparaît pas dans ces remerciements, c'est soit que j'ai oublié de vous mentionner –auquel cas je vous prierai de m'en excuser–, soit que vous n'avez aucunement contribué à cette thèse, comme la grande majorité des êtres peuplant cette planète –auquel cas vous n'auriez aucune raison de lire ce document–.

RÉSUMÉ

L'information numérique n'a pas cessé de se développer ces dernières décennies. Les données privées appartenant à des individus, des entreprises ou même des gouvernements sont devenues le fondement technique sur lequel s'appuient divers modèles économiques. Dans le domaine de la cryptographie, nous pensons généralement à des clés privées ou à des mots de passe, mais ce n'est pas ce type d'information qui nous intéresse dans cette thèse. Cette thèse s'intéresse aux données qui constituent le fondement même d'une entité. Pour une personne, il peut s'agir de sa situation économique (revenus, prêts, impôts ...) ou de sa santé (historique médical ...) ou plus simplement son âge, son adresse, ses opinions politiques ou toute autre information qu'il ou elle ne souhaite pas révéler à n'importe qui. Pour une entreprise, il peut s'agir d'une base de données de clients, d'employés, de sa situation économique ou toute information sur son fonctionnement interne.

Historiquement, la cryptographie s'intéresse à la conception de schémas de chiffrement permettant à des utilisateurs distants, qui se connaissent généralement et se font confiance, de communiquer de manière sécurisée malgré la présence d'un adversaire externe. Bien que cet objectif soit déjà compliqué à atteindre, son énoncé est simple à comprendre. Dans notre société moderne hyper connectée et centrée sur les informations personnelles, les choses deviennent malheureusement plus compliquées. Avec la diversification des appareils connectés et des réseaux sociaux, nous devons désormais communiquer, interagir et collaborer avec un grand nombre de parties, que nous ne connaissons souvent pas et auxquelles nous faisons encore moins confiance. Il est même possible que nous ayons des conflits d'intérêts avec certaines d'entre elles! À la différence de la cryptographie classique, il est donc indispensable de supposer que l'adversaire est interne et qu'il peut être une ou plusieurs parties. Malgré cela, il faut collaborer avec eux, ce qui implique souvent l'utilisation de nos données privées. Cette situation paradoxale nous amène donc à concevoir de nouveaux outils cryptographiques permettant de contrôler la fuite d'informations confidentielles alors même qu'elles sont communiquées et utilisées par des tiers auxquels nous ne faisons pas confiance.

De tels outils ont été formellement introduits en 1982 par Andrew Yao [Yao82] en tant que *calculs deux-parties sécurisés* (2PC). L'objectif du 2PC est de concevoir des protocoles autorisant deux utilisateurs à calculer coopérativement une fonction arbitraire de leurs données privées sans toutefois révéler lesdites données à la partie opposée. Yao donne une preuve de faisabilité en imaginant une solution au *problème des millionnaires*. Le problème est défini de la manière suivante: deux millionnaires souhaitent savoir lequel des deux est le plus riche sans

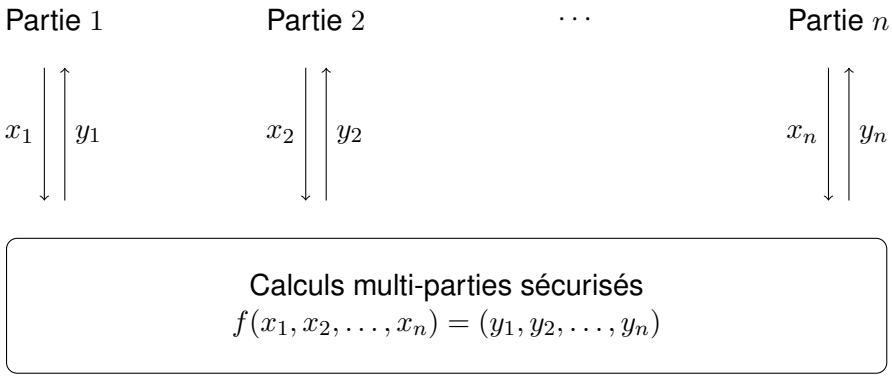


Figure 1: Calculs multi-parties sécurisés

révéler sa fortune à l'autre. Le protocole de Yao permet donc à deux utilisateurs de comparer de façon sécurisée deux valeurs privées. En 1986, Yao trouve une solution générale au problème du 2PC permettant ainsi de calculer n'importe quelle fonction, et non uniquement une comparaison. Sa solution prit plus tard le nom de *garbled circuits*. En quelques mots, un des utilisateurs va "chiffrer" la fonction à évaluer (vue comme un circuit Booléen) tandis que l'autre va pouvoir l'évaluer sans apprendre les valeurs intermédiaires.

Une généralisation naturelle des calculs deux-parties sécurisés est définie par les *calculs multi-parties sécurisés* (MPC). Ce nouveau problème peut être vu comme n participants cherchant à calculer une fonction de leurs paramètres privés d'une manière sécurisée, c'est-à-dire tel que l'exactitude du résultat et l'anonymat de leurs données soient assurés. Concrètement, si le participant i connait x_i pour $1 \leq i \leq n$, alors le MPC permet de déterminer $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$, tel que le participant i apprennent uniquement y_i, comme représenté dans la Figure 1. Dans la plupart des cas d'applications, les utilisateurs obtiennent le même résultat $y_1 = \cdots = y_n$. En 1987, Goldreich, Micali et Widgerson [GMW87] détaillent la première solution générale de MPC.

Comme mentionné précédemment, au contraire de la cryptographie traditionnelle, nous considérons dans les protocoles 2PC et MPC que l'adversaire est interne au protocole. Il est un des utilisateurs, voire plusieurs. Par conséquent, il est nécessaire de définir de nouveaux modèles d'attaquants. Ces adversaires peuvent être catégorisés selon leur capacité ou volonté à dévier des spécifications du protocole. Nous étudions essentiellement deux modèles: le modèle *semi-honnête* et le modèle *malveillant*.

Par définition, l'adversaire semi-honnête, ou honnête-mais-curieux, ne dévie pas du protocole, mais cherche à apprendre plus d'information qu'autorisé, en exploitant le résultat et les calculs intermédiaires. Il s'agit donc de vérifier que le protocole ne révèle pas d'information sensible par inadvertance. Les protocoles conçus dans ce modèle ont l'avantage d'être généralement très efficaces et ils sont souvent considérés comme une étape importante vers des mod-

èles de sécurité plus élevés.

À l'inverse, l'attaquant malveillant peut dévier arbitrairement du protocole afin d'obtenir des informations sur les données des autres parties. Dans ce modèle, la sécurité est souvent assurée par l'ajout de mécanismes garantissant que l'adversaire ne peut pas dévier du protocole ou que le protocole s'interrompra avant de divulguer des données privées. Toutefois, ce haut niveau de sécurité s'accompagne souvent d'une perte d'efficacité.

Dans ce contexte, cette thèse apporte diverses contributions. Les travaux de Yao [Yao86], connus sous le nom de "garbled circuits", sont une solution générale au problème de 2PC dans le modèle semi-honnête. Il est cependant vite apparu évident que cette solution n'était pas sécurisée face à un adversaire malveillant. En effet, ce dernier peut aisément modifier la fonction à évaluer sans que l'autre partie puisse s'en apercevoir. Ainsi, l'attaquant peut apprendre l'information de son choix. L'utilisation de "cut-and-choose" s'est révélée être une contre-mesure adéquate face à un tel adversaire: un grand nombre de circuits sont générés par l'attaquant et seule une portion est évaluée, les autres étant vérifiés par la partie opposée. Toutefois, cette solution s'accompagne d'un surcoût considérable.

Depuis, de nombreuses optimisations ont été réalisées, tant sur les "garbled circuits" que sur le "cut-and-choose". Néanmoins, il n'a jamais était défini clairement comment un adversaire malveillant pouvait corrompre un circuit à évaluer. C'est là qu'intervient la première contribution de cette thèse: nous définissons formellement quelles modifications du circuit l'adversaire peut faire sans que cela soit détecté par le participant honnête. Nous montrons que ses possibilités sont étonnamment limitées, du moins plus restreintes que ce que la précédente revue de littérature laisse suggérer. Nous analysons ensuite l'impact de cette étude sur des circuits réels et observons que certains circuits ne nécessitent pas l'utilisation de "cut-and-choose" pour être sûrs face à un adversaire malveillant.

La seconde contribution apporte un nouveau domaine d'application au MPC. En effet, nous étudions le cas des services basés sur la localisation, qui sont devenus de plus en plus présents ces dernières années. Toutefois, ces applications reposent aujourd'hui sur l'honnêteté des gens à transmettre leur véritable position. S'ils ont une motivation à tricher, ils peuvent le faire facilement. Les systèmes de preuves de localisation corrigent ce défaut en permettant à un prouveur d'obtenir des preuves de sa présence à un endroit et un instant donnés, à l'aide des témoins qui sont autour de lui. Il pourra ensuite fournir ces preuves à un vérifieur afin d'obtenir l'accès à un service. Cependant, on peut facilement concevoir que ces différents utilisateurs ne souhaitent pas révéler leur identité et leur position à chaque génération de preuve de localisation.

Notre seconde contribution est une conception du premier système de preuve de localisation respectueux de la vie privée. Grâce à l'utilisation de calculs multi-parties sécurisés, le prouveur est en mesure d'obtenir ses preuves, tout en garantissant que son identité et sa position ne sont pas révélés aux témoins. Réciproquement, l'identité et la position des témoins ne

sont révélées ni au prouveur, ni au vérifieur.

En contribution annexe de ce travail, nous concevons aussi un nouveau protocole de calcul de maximum sécurisé. Ce protocole permet à n participants de savoir lequel d'entre eux possède la plus grande valeur sans les révéler. À la différence des protocoles précédents nécessitant $O(n^2)$ opérations, notre solution ne requiert que $O(n \log(n))$ opérations, mais s'accompagne d'une petite divulgation d'information. Bien que nous l'ayons conçue spécifiquement dans le cadre des preuves de localisation, nous pensons que notre solution peut s'appliquer à de nombreux scénarios où cette fuite d'information est tolérable.

La dernière contribution porte sur l'étude de primitives facilement évaluables en calculs multi-parties. Plus spécifiquement, notre étude porte sur le générateur pseudo-aléatoire de Goldreich (PRG de Goldreich). Les générateurs pseudo-aléatoires localisés permettent d'étendre une petite chaîne aléatoire en une chaîne pseudo-aléatoire de plus grande taille, tel que chaque bit de sortie ne dépende que d'un nombre constant d de bits d'entrée. Cette particularité donne à cette primitive de nombreuses applications dans diverses branches de la cryptographie, et particulièrement en MPC grâce à sa faible complexité. En effet, cela rend cette famille de PRG facilement évaluable par un groupe de participants de sorte qu'aucun d'entre eux ne connaisse la chaîne aléatoire initialement utilisée.

Tandis que la sécurité théorique du PRG de Goldreich a été intensivement étudiée, aboutissant à de nombreux critères que doivent vérifier les paramètres pour être sécurisés, peu de résultats s'intéressent à la sécurité concrète et l'efficacité réelle de cette primitive. Motivés par les nombreuses applications théoriques et l'espoir de voir des instanciations pratiques ce celles-ci, nous initions une analyse de la sécurité réelle du PRG de Goldreich.

TABLE OF CONTENTS

INTRODUCTION

In the last few decades, the world has turned into a modern information-driven society. The everyday life of individuals, companies and governments is full of cases where various kinds of private information are valuable resources. While cryptographers might think of keys or passwords, these types of secrets are not the main concern in this thesis. Instead, this thesis is concerned with the data that is related to the primary business of a private person or a corporation. For a person, this might be information about his or her economic situation (incomes, loans, tax, ...), about his or her health condition (past or current diseases, allergies, ...) or more simply about his or her age, address, political tendencies and so on. Instead, for a company, it might be a customer database, the economic situation or information related to its internal functioning. For governments, it could be a list of tax-fraud suspects, positions of satellites, etc.

Cryptography has been historically dealing with the design of encryption schemes, permitting distant parties, who generally know and trust each other, to communicate securely even in the presence of an external eavesdropper. Although this goal is not trivial to reach, its purpose is easy to understand. Things get much more complicated in modern society: using electronic devices, we need to communicate, interact and do business with a large number of parties, some of whom we have never met, and most of whom we do not trust. Some of these parties may even have conflicts of interest with us! In this context, unlike in traditional cryptography, we have to assume that the adversary may be one or several of the inside participants. And yet, we have to do business with them, which often requires our private data. This paradoxical situation calls for cryptographic tools for controlling leakage of confidential data while they are being communicated and computed on, even in the case where their owner does not trust the parties he or she is interacting with.

Such tools were formally introduced in 1982 by Andrew Yao [Yao82] as *secure two-party computation* (2PC), which is a subfield of cryptography. The goal of 2PC is to create protocols allowing two parties to co-operatively compute an arbitrary function of their private inputs without sharing the clear value of their inputs with the opposing party. Yao gives evidence of feasibility with a solution to the so-called *Millionaires' Problem*. The problem is stated as follows: two millionaires wish to learn who is the richer without telling their actual wealth to the other. Beside the undeniable breakthrough that it represented in the upper class at that time, it also gave intuitions about feasibility of such problems. In 1986, Yao found a general solution for the two-party computation problem [Yao86]. His general solution later took the name of *garbled circuits*, which allows one of the participants to "encrypt" the function to evaluate (seen as a

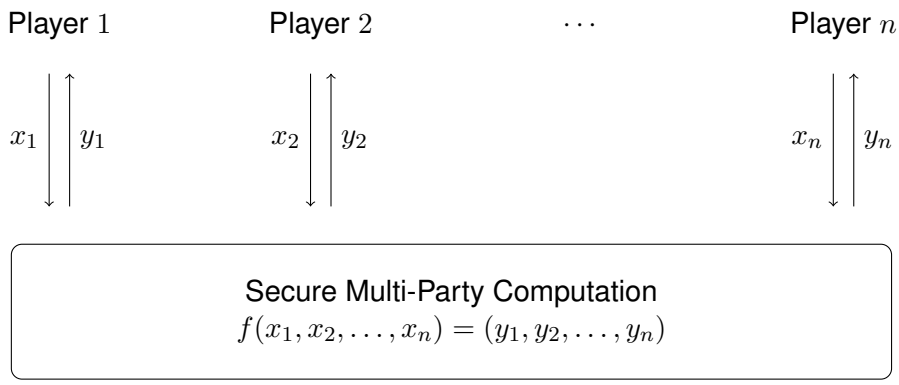


Figure 2: Multi-party computation

Boolean circuit) and the other to obliviously evaluate it on "encrypted" inputs, without leaking any intermediate value. Not only this solves the 2PC problem, but it is also very efficient, and even optimal, in terms of rounds of communication.

A natural generalization of the secure two-party computation is the secure *secure multi-party computation* (MPC). Secure multi-party computation can be defined as the problem of n participants to compute an agreed function of their inputs in a secure way, where security means guaranteeing the correctness of the output as well as the privacy of the users' inputs. Concretely, if participant i knows x_i for $1 \leq i \leq n$, then MPC allows to compute $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$, such that player i learns y_i but nothing more, as shown in Fig. 2. In most applications, the participants obtain the same output $y_1 = \cdots = y_n$.

In 1987, Goldreich, Micali and Widgerson [GMW87] detail the first general solution to secure multi-party computation. Their solution is based on *secret sharing*, which enables a participant to split his data into several shares that will be sent to each of the participants, with the guarantee that any individual share does not leak any sensitive information. The authors then show how to perform operations on these shares without revealing them.

Since then, both two-party and multi-party computations have attracted a lot of interest. New generic solutions have been designed and decades of optimizations have made garbled circuits and secret-sharing based solutions very efficient ([ZRE15, DPSZ12] and many others). Interestingly, custom protocols were also given, restricted to a single function and sometimes in some specific context. They are often more efficient than generic solutions.

Applications of Secure Multi-Party Computations

2PC and MPC enjoy a wide variety of applications. Let us describe two that have been commercialized: a private double auction system and a privacy-preserving data mining system.

Private Double Auction

A double auction is a process of trading goods when potential buyers and sellers submit their bids to an auctioneer. In this context, a bid is the quantity the bidder agrees to buy/sell for a given price. The auctioneer then determines the price p that clears the market: the supply matches the demand.

However, the auctioneer has access to all bids for any price $p' \neq p$. This brings new privacy issues. For example, if a seller initially agreed to sell at $p' \ll p$, this might leak sensitive information about his or her economic situation, and thus disadvantaging him or her in future negotiations. This is particularly true if the auctioneer has interests that conflicts with the bidders.

This tricky situation was met by the Danish sugar beet farmers in 2008. Here is the context: several thousand farmers produce sugar beets, which are sold to a single corporation (a buyer's monopoly). Farmers have contracts that give them rights and obligation to sell a certain amount of beets to this company at a certain price. These contracts can be traded between the farmers. Such trades were historically very limited and done via bilateral negotiations. However, due to several political factors, there was an urgent need to reallocate contracts between farmers: a nationwide double action was required. However, since a bid reveals the productivity and the economic situation of a farmer, they were not willing to let the corporation acting as an auctioneer, nor any other entity.

As explained in [BCD$^+$09], this situation was solved using secure multi-party computations. More precisely, secret sharing was used to split the farmers' bid between the parties. Then a private double auction algorithm was performed on the shared data, thus allowing to determine the price p and the quantity that each farmer committed to buy/sell for price p, without leaking any additional information. Since then, the system has been used several times by thousands of Danish farmers.

Privacy-Preserving Data Mining

In most countries, databases containing personal, medical or financial information about individuals are classified as sensitive and the corresponding laws specify who can gather and process them. However, this sensitive information plays a crucial role in medical, financial or social studies. Thus, one needs new mechanisms for conducting statistical surveys without compromising the privacy of the participants. The corresponding research area is commonly referred as privacy-preserving data mining.

Most approaches focus on anonymized inputs (through k-anonymity) or randomized inputs (roughly speaking, a small error is added to individuals' inputs). However, the nature of these solutions leads to a trade-off between privacy and accuracy of the outcomes. The more

anonymized or randomized the inputs are, the more privacy-preserving the system becomes but the more meaningless the statistical survey is. Also the security is preserved only on average.

Sharemind [BLW08] gives the first commercial solution based on secure multi-party computation. It is designed with secret sharing: the participants of the survey split their personal data among a few servers, that will perform operations on the shares. Sharemind thus allows to reach both privacy and accuracy of the outcomes, as long as the majority of the servers are honest.

In 2015, this solution was actually used for a large-scale statistical study in Estonia, as reported in [BKK+15]. Using Sharemind, social scientists managed to cross the Estonian Tax database with the Ministry of Education database in order to analyze the correlation between working during university studies and failing to graduate in time. In this context, MPC allows to compute meaningful statistics without leaking any information about incomes or degree course.

In 2016, a similar solution was deployed to analize the gender and ethnicity wage disparities in the Greater Boston Area [LVB+16]. Although 50 of the biggest employers had agreed to take part in this study, none would let sensitive employee wage data their servers and no institution was willing to gamble on hosting, and possibly losing, the data. MPC and secret sharing solved this issue.

Adversary Models

As shown in these two cases of application, unlike traditional cryptographic scenarios, such as encryption or signature, one must assume that the adversary in a 2PC or MPC protocol is one (or more in the case of MPC) of the participants engaged in the system. Therefore, it soon appeared essential to define new adversary models. Adversaries faced by the different protocols can be categorized according to how willing they are to deviate from the protocol. There are essentially two types of adversaries: the *semi-honest adversary* and the *malicious adversary*.

The semi-honest adversary does not deviate from the protocol specification but tries to gather more information than allowed out of the protocol. Thus, it is a weak security model that only prevents from inadvertent leakage of information between the parties. However, protocols in the semi-honest model are often very efficient and are generally considered as an important first step for achieving higher levels of security.

The malicious adversary may arbitrarily deviate from the protocol specification in its attempt to force the output or to learn more information on the other parties' inputs. Protocols that achieve security in this model provide a very high security guarantee. Security against malicious adversaries is often achieved by ensuring with cryptographic mechanisms that the participants

cannot deviate from the protocol, or the protocol will abort without leaking anything. However, using these tools generally leads to a reduction in efficiency.

In order to better understand the motivation and the possibilities of a malicious adversary, let us focus on the private double auction system previously described. Obviously, any participant has an incentive to force the output of double auction so that he or she buys/sells at any chosen price. If their is no countermeasure, the malicious participant may try to do so by sending inconsistent shared data to the other users. Similarly, the malicious participant can also deviate from the protocol during the evaluation of the double auction algorithm by computing another function (in order to change the output of the algorithm).

Alternatively, a malicious participant could also deviate from the protocol in order to learn more information than allowed, which can give a clear competitive advantage in the double auction. For example, he or she can try to make the protocol abort for some condition on the other parties inputs (e.g. some condition on the buying price of a specific concurrent). In this case, if the protocol is restarted, he can repeat the same attack under some other condition (e.g. the buying price of another concurrent), until he or she accumulates enough information to maximize the profit. Then, any party has to prove that he or she is running the expected computation (while keeping the manipulated data secret), which often leads to lower performances.

Note that in the malicious model, no distinction is made between the deviations that are undetected by the other parties and the corruptions that are detected (such as the abortion of the protocol). In some context, it is reasonable to consider that an adversary is willing to cheat only if the risk of getting caught is “not too high”. In the example previously described, if the honest parties can determine who is responsible for the abortion of the protocol, they may just ban the adversary and restart the protocol. Then, the adversary must remain undetected. This behavior is captured by the *covert adversary*. Although it has been less studied, this model can lead to more efficient solutions than in the malicious model.

Contributions

On the Leakage of Corrupted Garbled Circuits

The pioneering work of Yao [Yao86], known as garbled circuits, is a general solution to the secure two-party computation problem, which is extremely efficient in terms of rounds of communications, that is constant and optimal. It involves two parties: a generator that builds the garbled circuit to be evaluated, and an evaluator that executes it on its inputs. It was originally designed in the semi-honest model and it was clear that a malicious generator could modify the logic gates of the garbled circuit before sending it to the evaluator for execution.

Applying cut-&-choose to garbled circuits soon appeared to fix this issue, but requires to generate, transmit and evaluate a large number of garbled circuits, which can clearly lead to a serious overcost.

Since then, a lot of work has been made to optimize the garbled circuits, on the one hand [BMR90, NPS99, KS08, ZRE15], and the cut-&-choose, on the other hand [MF06, LP07, sS11, MR13, Lin13, sS13, AMPR14, WMK17]. The best of these approaches still requires s garbled circuits for a statistical security of 2^{-s} against malicious adversaries, thus resulting in a serious overhead compared to the semi-honest model.

However, all these techniques aim at avoiding any kind of modification on the circuit. Nevertheless, it has never been studied which modifications a malicious generator can make to a single garbled circuit, still leading to an accepted execution, and then why the cut-&-choose is necessary.

The first contribution of this thesis is to define formally what the adversary is able to corrupt. We prove that, for a large class of circuits, the malicious generator is limited to add NOT gates on the wires of his choice. Hence, his possibilities are much more restricted than what we could have expected from the previous state of the art. We also show some impacts of this result on real circuits and on cut-&-choose based solutions. Finally, we give a garbled-circuit solution against covert adversaries that is not based on cut-&-choose.

Location Proof System Based on Multi-Party Computations

We show how multi-party computations can help users to protect their privacy in everyday life. More specifically, we study the case of location-based services that have become quite popular (e.g. GPS, location-based advertising, augmented reality games). Their variety and their numerous users show it clearly. However, these applications rely on the people's honesty to use their real location. If they are motivated to lie about their position, they can easily do so. A location-proof system allows a prover to obtain proofs from nearby witnesses, for being at a given location at a given time. Such a proof can be used to convince a verifier later on. However, provers and witnesses may not want to broadcast their identity or their position each time they generate location proofs.

Many solutions have been designed in the last decade, but none protects perfectly the privacy of their participants. In this thesis, a solution is presented in which a malicious adversary, acting as a prover, cannot cheat on his position. It relies on multi-party computations and group-signature schemes to protect the private information of both the prover and the witnesses against any semi-honest participant.

Additionally, this thesis also gives a new secure multi-party maximum computation protocol for the specific context of location-proof systems. This protocol allows n users to know which one of them has the greatest value without revealing these values. It requires $O(n \log(n))$ com-

putations and communications, which greatly improves the previously known solutions having $O(n^2)$ complexities, but at the cost of some small leakage that we analyze. Although it is designed for our location-proof system, it can be applied to any scenario in which a small information leakage is acceptable.

On the Concrete Security of Goldreich's Pseudorandom Generator

Historically, the design of symmetric cryptographic primitives (such as block ciphers, pseudorandom generators, and pseudorandom functions) has been motivated by efficiency considerations (memory consumption, hardware compatibility, ease of implementation,...). The field of multi-party computation, where parties want to jointly evaluate a function on secret inputs, has led to the emergence of new efficiency considerations: the efficiency of secure evaluations of symmetric primitives is strongly related to parameters such as the circuit depth of the primitive, and the number of its AND gates. This observation has motivated the design of MPC-friendly symmetric primitives in several recent works (e.g. [ARS$^+$15, CCF$^+$16, MJSC16, GRR$^+$16]), that aim for an efficient secure evaluation.

Local pseudorandom generators allow to expand a short random string into a long pseudorandom string such that each output bit depends on a constant number d of input bits. Due to its extreme efficiency features, this intriguing primitive enjoys a wide variety of applications in cryptography and makes very promising candidate MPC-friendly PRGs. In the polynomial regime, where the seed is of size n and the output of size n^{s} for $\mathsf{s} > 1$, the only known solution is the *Goldreich's PRG*.

While the security of Goldreich's PRG has been deeply investigated, with a variety of results deriving provable security guarantees against class of attacks in some parameter regimes and necessary criteria to be satisfied by the underlying parameters, little is known about its concrete security and efficiency. Motivated by its numerous theoretical applications and the hope of getting practical instantiations for some of them, we initiate a study of the concrete security of Goldreich's PRG. Along the way, we develop a new guess-and-determine-style attack, and identify new criteria which refine existing criteria and capture the security guarantees of candidate local PRGs in a more fine-grained way.

Organization

The rest of this manuscript is organized as follows: Chapter 1 gives the basic cryptographic notions that will be used in the following chapters. Next, Chapter 2 defines formally how a malicious adversary can corrupt a circuit and what the impact on real circuits. Chapter 3 gives both practical and theoretical contributions to secure multi-party computations: we give the first

construction of location-proof system based on MPC and we also give a new secure maximum computation scheme. Finally, in Chapter 4, by cryptanalyzing it, we study the concrete efficiency of an MPC-friendly pseudorandom generator: the Goldreich's pseudorandom generator.

Personal Publications

[CDM+18] G. Couteau, A. Dupin, P. Méaux, M. Rossi, and Y. Rotella. On the concrete security of Goldreich's pseudorandom generator. In *ASIACRYPT 2018, Part II*, LNCS, pages 96–124. Springer, Heidelberg, December 2018.

[DPB18] A. Dupin, D. Pointcheval, and C. Bidan. On the leakage of corrupted garbled circuits. In *ProvSec 2018*, LNCS, pages 3–21. Springer, Heidelberg, 2018.

[DRB18] [1] A. Dupin, J.-M. Robert, and C. Bidan. Location-proof system based on secure multi-party computations. In *ProvSec 2018*, LNCS, pages 22–39. Springer, Heidelberg, 2018.

[1] This work was partially done during my Master at the École de Technologie Supérieure of Montreal (Canada), with the supervision of Jean-Marc Robert.

CHAPTER 1

PRELIMINARIES

In this chapter, we introduce all the basic tools like homomorphic encryption, zero-knowledge proof and oblivious transfer that will be used throughout this thesis. For comparison purpose, we also present some seminal solutions such as secret sharing.

Contents

Protocol 1.1: Yao's millionaires problem and its solution [Yao82]

Input: The two integers a and b of Alice and Bob, where $a, b \leq N$. The public key N_A of Alice and its encryption function $E_A(\cdot)$ and decryption function $D_A(\cdot)$.
Output: Bob learns whether $a \geq b$ or $a < b$.
Step 1: Bob picks a random $x \in \mathbb{Z}_{N_A}$, encrypts $k = E_A(x)$ and sends $k - b$.
Step 2: Alice does the following:
repeat
  Alice takes a random prime number p of size $|N_A|/2$.
  for $i = 1$ *à* N **do**
    Alice computes $y_i = D_A(k - b + i)$ and $z_i = y_i \mod p$.
  end
until $\forall i, \forall j \neq i, |z_i - z_j| \geq 2$;
Step 3: Alice sends $z_1, z_2, \ldots, z_a, z_{a+1} + 1, \ldots, z_N + 1$ and p to Bob.
Step 4: Bob read the b^{th} element sent by Alice, noted z'_b.
  Bob determines $a \geq b$ if and only if $z'_b = x \mod p$.

1.1 Yao's Millionaires' Problem

Yao's millionaires' problem is a well-known problem presented by Andrew Yao in the early eighties [Yao82], that introduced the concept of secure two-party computation. The problem discusses two millionaires, Alice and Bob, who are interested in knowing which of them is richer without revealing their actual wealth. The first solution was given by Yao himself [Yao82] and is presented in Protocol 1.1. It only uses of a public-key encryption scheme and does not require any homomorphic property.

The correctness of this protocol is easy to prove: remark that $y_b = x$, then $z_b = x \mod p$. If $a \geq b$, then we have $z'_b = z_b = x \mod p$. Otherwise, we have $z'_b = z_b + 1 \neq x \mod p$.

Although this protocol is correct, it is highly inefficient due to its exponential complexity. Since then, more efficient solutions have been designed, either from generic tools or with custom protocols. Such protocols are given in Chapter 3.

Since the problem is analogous to a more general problem where two participants have inputs a and b and the goal is to determine whether $a \geq b$ without leaking the clear values a and b, let us call these solutions *secure two-party comparison protocols*.

1.2 Adversary Models

Secure multi-party computation is about designing protocols for allowing participants to jointly compute a public function over their private inputs. If the parties follow the protocol specifications, then they are guaranteed that their private information remains secret. But what if one of the participants does not follow the instructions? What if one of them deviates from the proto-

col? If one of the participants does not follow the rules, it is very likely that the protocol leaks more information than allowed unless the protocol was specifically designed to resist such attacks.

Therefore, when designing MPC protocols, defining what kind of adversary we are dealing with is a crucial matter. The first adversary model that is generally considered is the semi-honest adversary, which is also referred as passive adversary or honest-but-curious adversary. The following definition is extracted from [HL10] :

Definition 1.1 (semi-honest adversary model)**.** *A semi-honest adversary follows the protocol specification exactly, but it may try to learn more information than allowed by looking at the messages that it received and its internal state.*

Of course, it is not always realistic to assume that all participants will behave correctly. In fact, designing protocols in the semi-honest adversary model is often seen as a first step toward more powerful adversaries. Malicious adversaries (also known as active adversaries), on the contrary, do not follow the rules. The following definition is also extracted from [HL10] :

Definition 1.2 (malicious adversary model)**.** *A malicious adversary may use any efficient attack strategy and thus may arbitrarily deviate from the protocol specification.*

Designing protocols against malicious adversaries ensures privacy for the participants however they behave, but often requires use of heavy cryptographic mechanisms. This may result in impractical solutions. Thus it is sometimes interesting to define intermediate adversary models, yet realistic.

In the malicious adversary model, there is no distinction between deviations that are detected by the other parties and deviations that are indistinguishable. Therefore, in many contexts, it is reasonable to consider that the adversary is willing to cheat only if the risk of getting caught is not too high. Then, we can define the covert adversary as in [AL07]:

Definition 1.3 (covert adversary with ϵ-deterrent)**.** *A covert adversary with ϵ-deterrent can deviate from the protocol as long as the probability of being caught by the honest parties is lower than ϵ.*

In this thesis, we consider those three kinds of adversaries.

1.3 Useful Tools for Multi-Party Computation

In this section, we detail three cryptographic tools that are particularly useful when designing secure multi-party computation protocols.

1.3.1 Homomorphic Encryption Schemes

Homomorphic encryption is a particular form of encryption that allows computations on encrypted data. It generates an encrypted result which, after decryption, matches the result of the operations as if they were computed on the plaintext.

Several kinds of homomorphic encryption schemes can be defined, depending on the nature of the operations they allow to perform:

- *Multiplicative encryption schemes*: these schemes allow to perform multiplication over encrypted data. The RSA [RSA78] and the ElGamal [ElG84] encryption schemes are two of them. Indeed, with these two schemes, multiplying two ciphertexts allows to obtain an encryption of the product of the two plaintexts.

- *Additive encryption schemes*: they allow to perform addition over encrypted data, and by extension a multiplication between an encrypted value and a clear value. The Paillier's cryptosystem [Pai99] is an example: multiplying two ciphertexts creates an encryption of the sum of the two plaintexts.

- *Fully Homomorphic Encryption (FHE) schemes*: they support both addition and multiplication, which makes any circuit evaluable over encrypted data. The first construction was given by Gentry in 2009 [Gen09] and has attracted a lot of interest. Despite major optimizations, the size of the ciphertexts and the overcost for performing operations are still an important issue.

In this thesis, we are mostly interested in additive schemes. Let us detail the Paillier's cryptosystem. Let p and q be two secret large prime numbers. The public and private keys are defined as follows:

$$\text{pk} = N = p \cdot q \text{ and sk} = \varphi(N) = (p-1)(q-1) \ .$$

Let $m \in \mathbb{Z}_N$ be the message to encrypt and $r \in \mathbb{Z}_N^*$ be a random number chosen by the encrypter. Then the encryption function is

$$\begin{aligned} E_N : \mathbb{Z}_N \times \mathbb{Z}_N^* &\to \mathbb{Z}_{N^2}^* \\ E_N(m, r) &= (1+N)^m \cdot r^N \mod N^2 \\ &= c \ . \end{aligned}$$

And the decryption function is

$$\begin{aligned} D_{\varphi(N)} &: \mathbb{Z}_{N^2}^* \to \mathbb{Z}_N \\ D_{\varphi(N)}(C) &= \frac{(c^{\varphi(N)} \mod N^2) - 1}{N} \cdot \varphi(N)^{-1} \mod N \\ &= m \ . \end{aligned}$$

The correctness of this decryption can be proven under the binomial theorem and Euler's theorem. Note that this encryption scheme is probabilistic and has the following homomorphic properties:

$$\begin{aligned} E_N(m_1, r_1) \cdot E_N(m_2, r_2) &= E_N(m_1 + m_2, r_1 r_2) \\ E_N(m_1, r)^{m_2} &= E_N(m_1 m_2, r) \ . \end{aligned}$$

These properties make this tool a very interesting primitive for secure multi-party computation.

The contributions of Chapter 3 heavily rely on these properties.

1.3.2 Zero-Knowledge Proof

A *zero-knowledge proof* (ZKP) is a protocol allowing a prover to convince a verifier that a given statement is true, without leaking any information apart from the fact that the statement is true. It has many cryptographic applications and most particularly in MPC, since it allows a user to prove that it has not deviated from a protocol. Informally, a proof must satisfy the following three properties:

1. *Correctness*: if the statement is true and the prover knows a proof of this, he will succeed in convincing the verifier.

2. *Soundness*: if the statement is false, no prover can convince the verifier of the truth of the statement, except with some small probability.

3. *Zero-knowledge*: if the statement is true, no verifier learns anything other than the fact that the statement is true.

The Schnorr protocol [Sch90] is an example of zero-knowledge proof permitting a prover to prove knowledge of discrete logarithm without revealing it. This protocol is shown in Protocol 1.2.

Informally, the correctness of this protocol can be easily proven using the fact that

$$g^a y^e = g^{r-ex} g^{ex} = g^r \ ,$$

Protocol 1.2: Schnorr protocol for proving knowledge of a discrete logarithm

Input: Let $\mathbb{G}$ be a public group of order q and generator g, where the discrete logarithm problem is hard. $x \in \mathbb{Z}_q^*$ is known only by the prover. Let $y = g^x$ be public.
Output: The verifier is convinced that the prover knows x
Commitment phase: the prover picks a random $r \in \mathbb{Z}_q^*$, computes the commitment $c = g^r$ and sends c to the verifier.
Challenge phase: the verifier picks a random challenge $e \in \mathbb{Z}_q^*$ and sends it to the prover.
Answer phase: the prover sends $a = r - e \cdot x \mod q$ to the verifier.
Verification: the verifier accepts the proof (c, e, a) if and only if $c = g^a \cdot y^e$

which indeed matches c. The soundness property can be proven by showing that an adversarial prover able to produce a valid tuple (c, e, a) without knowing x can be used to compute any discrete logarithm in $\mathbb{G}$ (and in fact x itself). The zero knowledge property relies on the existence of an efficient simulator that takes as input (y, e) and outputs a valid proof (c, e, a) without needing the secret x. Note that the simulator computes the answer a before the challenge c.

ZKP is very convenient for proving that no participant has deviated from the protocol. Thus, it allows to turn any protocol secure in the semi-honest model into a protocol secure in the malicious model. The efficiency of this transformation depends on the protocol to secure.

1.3.3 Oblivious Transfer

1 out of 2 Oblivious Transfer

1 out of 2 oblivious transfer, also known as 1-2 oblivious transfer or just oblivious transfer (OT), is a useful primitive for secure multi-party computation. The protocol involves two participants: a sender and a receiver. The sender has two messages m_0 and m_1 and the receiver has a bit b and wishes to receive m_b, while keeping b secret. The sender wants to ensure that the receiver learns only one of the two messages. Even, Goldreich, and Lempel gave the first solution to this problem [EGL82], using any public-key encryption scheme. This solution is described in Protocol 1.3.

As shown in Protocol 1.4, 1 out of 2 oblivious transfers can be made more efficient with additive homomorphic encryption schemes. Note that this protocol is only secure against semi-honest adversaries. Indeed, for example, a malicious receiver could send an encryption of 2 (instead of 0 or 1) during Step 1 and learn $m_1 - m_0$. This would be crucial if we consider two plaintext English messages: knowing the difference could allow to recover them both. This can be prevented by adding some zero-knowledge proofs.

Note that 1 out of 2 oblivious transfer of long messages (longer than the encryption scheme allows in Protocol 1.3 or 1.4) can be reduced to oblivious transfer of short strings using any pseudorandom generator G. Very briefly, the sender generates two keys k_0 and k_1 and sends

Protocol 1.3: Oblivious Transfer Protocol of Even et al. [EGL82]

Input: Alice has two secret messages m_0 and m_1 (of same size), a public encryption function $E_A(\cdot)$ and a decryption function $D_A(\cdot)$. Bob has a bit b.
Output: Bob learns m_b.
Step 1: Alice chooses two random strings r_0 and r_1 (same size as the ciphertexts) and sends them to Bob.
Step 2: Bob chooses a random string k (same size as the messages), computes $q = E_A(k) \oplus r_b$ ans sends it to Alice.
Step 3: Alice computes $k_0 = D_A(q \oplus r_0)$ and $k_1 = D_A(q \oplus r_1)$ and sends $(m_0 \oplus k_0, m_1 \oplus k_1)$ to Bob.
Step 4: Bob gets $m_b = (m_b \oplus k_b) \oplus k$.

Protocol 1.4: Oblivious Transfer Protocol from Additive Homomorphic Encryption Scheme

Input: Alice has two secret messages m_0 and m_1. Bob has a bit b, a public encryption function $E_B(\cdot)$ and a decryption function $D_B(\cdot)$.
Output: Bob learns m_b.
Step 1: Bob computes $E_B(b)$ and sends it to Alice.
Step 2: Alice computes homomorphically

$$E_B(m_b) = E_B(m_0 + b(m_1 - m_0))$$

and sends it to Bob.
Step 3: Bob decrypts and obtains m_b.

$m_0 \oplus G(k_0)$ and $m_1 \oplus G(k_1)$. The two parties then run a regular OT where the sender's input is (k_0, k_1). This trick is sometimes referred as *hybrid oblivious transfer*.

1 out of n and k out of n Oblivious Transfer

A natural generalization of this problem is the *1 out of n oblivious transfer* and then *k out of n oblivious transfer*. Both have been solved by Brassard, Crépeau and Robert [BCR87]. In the 1 out of n OT, the sender now has n secrets and the receiver wishes to obtain one of them. As before, the choice of the receiver and the other messages must remain secret. Note that it makes this primitive 2PC-complete[1]: if two parties have respective inputs x_1 and x_2 and want to compute $f(x_1, x_2)$, it "suffices" for the first party to compute $f(x_1, x_2')$ for every possible value of x_2' and then to act as the sender in a 1 out of n oblivious transfer protocol. The other party acts as the receiver and obliviously receives the ${x_2}^{th}$ messages, that is $f(x_1, x_2)$.

The solution of Brassard et al. [BCR87] allows to build 1 out of n OT from 1 out of 2 OT only. Then, the 1 out 2 OT is also 2PC-complete. Of course, when dealing with secure multi-party computation, more efficient solutions than the one just described are generally desirable.

[1] i.e. it is a general solution to the 2PC problem.

Protocol 1.5: Oblivious transfer extension of Ishai et al. [IKNP03]

Input: The sender has n pairs $(m_{j,0}, m_{j,1})$ of ℓ-bit messages, $1 \leq j \leq n$.
The receiver has n choices $b = (b_1, \ldots, b_n)$.
A security parameter λ and a random oracle $H : [m] \times \mathbb{F}_2^\lambda \to \mathbb{F}_2^\ell$.
Output: The receiver learns m_{j,b_j}, $1 \leq j \leq n$.
Step 1: The sender picks a random bit vector s of size λ.
Step 2: The receiver picks a random $n \times \lambda$ bit matrix T.
Step 3: The parties run λ OT protocols with reverse roles:
the receiver has inputs $(T^i, b \oplus T^i)$,
the sender has input s_i and obtains $s_i \cdot b \oplus T^i$, $1 \leq i \leq \lambda$.
Step 4: Let Q denote the $n \times \lambda$ matrix of values obtained by the sender.
For $1 \leq j \leq n$, the sender sends $((y_{j,0}, y_{j,1}))$ where

$$y_{j,0} = m_{j,0} \oplus H(j, Q_j) \text{ and } y_{j,1} = m_{j,1} \oplus H(j, Q_j \oplus s) \ .$$

Step 5: For $1 \leq j \leq n$, the receiver decrypts $m_{j,b_j} = y_{j,b_j} \oplus H(j, T_j)$.

In the rest of the thesis, we only consider 1 out of 2 oblivious transfer, that will be referred as oblivious transfer (OT) to improve readability.

Oblivious Transfer Extension

Due to its massive usage in secure protocols, efficiency is particularly crucial for oblivious transfer. In the two previous protocols, the bottleneck relies on the use of public-key encryption schemes, both in terms of computation and communication.

Therefore, the problem of extending a small number of OT to a large number of OT, with no additional asymmetric operations, has attracted a lot of interest. It has been solved by Ishai et al. [IKNP03] in the random oracle model[2]. In this model, the authors show that an arbitrary number of OT can be made from λ regular OT, where λ is a security parameter. Then, only $\mathcal{O}(\lambda)$ asymmetric operations are necessary for any number of OT.

Their solution is illustrated in Protocol 1.5. For a matrix M, we note M^i the i^{th} column of this matrix and M_j the j^{th} row.

The correctness of this protocol relies on the fact that $Q^i = s_i \cdot b \oplus T^i$. This implies that $Q_j = b_j \cdot s \oplus T_j$. Then, the definition of $y_{j,0}$ and $y_{j,1}$ can be developed as follows:

$$y_{j,0} = m_{j,0} \oplus H(j, b_j \cdot s \oplus T_j) \text{ and } y_{j,1} = m_{j,1} \oplus H(j, b_j \cdot s \oplus s \oplus T_j) \ .$$

It is now easy to see that the vector s (that is unknown to the receiver) will disappear in y_{j,b_j}. Similarly, the security relies on the fact that s does not disappear from $\overline{y_{j,b_j}}$. We refer the reader

[2] A random oracle is an oracle that responds to every unique query with a truly random response chosen uniformly from its output domain. If a query is repeated, then it responds the same way.

to the original paper for more details.

Then, this allows to build a large number of OT from λ OT where the roles are inverted, with a small communication overhead ($2n$ messages are sent in Step 4).

In some specific cases, Asharov et al. [ALSZ13] propose two optimizations to reduce this overhead: random-OT and correlated-OT.

Random-OT. In Protocol 1.5, consider that it is acceptable that the n pairs of messages $(m_{j,0}, m_{j,1})$ are chosen uniformly at random at the end of the protocol. Then, the authors suggest to define in Step 4:

$$m_{j,0} = H(j, Q_j) \text{ and } m_{j,1} = H(j, Q_j \oplus s) \ .$$

Therefore, there is no need to transmit $y_{j,0}$ and $y_{j,1}$, which completely removes the overhead. However, the sender does not know $m_{j,0}$ and $m_{j,1}$ before Step 4, which implies a deep change of functionality: the sender has no input but an output.

Correlated-OT. Similarly, consider that in each pair one of the messages (say $m_{j,0}$) can be chosen uniformly at random at the end of the protocol and the other message is correlated to the first ($m_{j,1} = f_j(m_{j,0})$). Then, the same trick can be applied:

$$m_{j,0} = H(j, Q_j) \text{ and } m_{j,1} = f_j(m_{j,0}) \ .$$

Therefore, only $y_{j,1}$ has to be transmitted, which lower by half the communication overcost. However, as for random-OT, $(m_{j,0}, m_{j,1})$ is outputted to the sender.

1.4 Garbled Circuits: a General Solution to the 2PC Problem

The seminal work of Yao [Yao86], which later took the name of *Yao's garbled circuits*, is a general solution to the two-party computation problem. It is extremely efficient in terms of number rounds of communications, which is constant and independent of the function to evaluate.

The target function is seen as a Boolean circuit: a collection of gates and wires to connect them. The two parties called generator and evaluator are responsible for respectively generating and evaluating the garbled circuit. At a high level, the generator prepares the garbled circuit by replacing the two possible values (0 and 1) of each wire by two random keys. He then "encrypt" and shuffle the truth table of each gate under these keys. These encrypted truth tables are given to the evaluator, along with the random keys representing the input of the generator.

After the evaluator retrieves the keys representing his input through oblivious transfer protocols, he can then start evaluating the circuit. Basically, knowing the encrypted truth table of

a gate and one key for each input wire is enough information to evaluate it and obtain the key matching the output. But it does not allow to decrypt the keys, and thus the evaluator can obliviously evaluate the entire circuit without learning any information. Finally, the evaluator decrypts the result by looking at some decryption table that is provided by the generator.

It was originally designed in the semi-honest adversary model, but can be adapted to malicious adversaries using further mechanisms, like cut-&-choose and ZKP.

Decades of optimizations have made this tool very practical. Indeed XOR gates of the Boolean circuit are no longer transmitted and are evaluated for free. Only non-linear operations, which can be seen as multiplications in $\mathbb{F}_2$, require some encryptions and decryptions.

Chapter 2 gives an extended introduction to garbled circuits, optimizations and countermeasures to malicious adversaries.

1.5 Secret Sharing: a General Solution to the MPC Problem

Secret sharing is about splitting a secret among a group of n participants, each of whom is given a *share* of the secret. The secret can then be reconstructed only if some predefined subset of shares are combined together. An individual share alone does not leak any information about the secret.

The work of Shamir [Sha79] introduced the first solution to (t, n)-*threshold secret sharing*: a specific case of secret sharing where any subset of t shares or more among n allows to efficiently reconstruct the secret. On the contrary, the knowledge of less than t shares reveals nothing. Shamir's secret sharing is based on a polynomial representation of the secret over a finite field $\mathbb{F}$. The only restriction is that $|\mathbb{F}| > n$, but we will assume for simplicity that $\mathbb{F} = \mathbb{Z}_p$ for some prime $p > n$.

Let s be the secret and $f_s(X) = \mathbb{F}[X]$ be a random polynomial of degree $t - 1$ such that $f_s(0) = s$. The secret is then shared by sending to participant P_j the share $s_j = f_s(j)$ for each $1 \leq j \leq n$. It is then trivial that the polynomial f_s (and therefore the secret s itself) can be reconstructed by any t shares or more, using the Lagrange interpolation.

Later, it has been demonstrated that if $t \leq (n-1)/2$, then the shamir's secret sharing becomes a general solution the multi-party computation problem in the semi-honest model. Indeed, it is easy to prove that it allows addition and multiplication over shared inputs, and therefore any function. For two secrets a and b, if each participant P_j has $f_a(j)$ and $f_b(j)$, then he can locally compute $f_{a+b}(j) = f_a(j) + f_b(j)$, and $f_{a+b}(X)$ is a polynomial of degree $t - 1$ and $f_{a+b}(0) = a + b$. Similarly, he can also compute $f_{ab}(j) = f_a(j) \cdot f_b(j)$, which is a valid share of $f_{ab}(0) = ab$. However, the polynomial $f_{ab}(X)$ is of degree $2t - 2$ and thus $2t - 1$ shares are necessary to reconstruct the secret. Since $t \leq (n-1)/2$, there are enough shares, but no further multiplication can be made without lowering the degree of the polynomial. To

perform this degree reduction, each participant P_j creates n new shares of his own share $f_{ab}(j)$ and sends one to each other participant. Each participant can then locally compute the Lagrange interpolation of the received shares and produce $f'_{ab}(j)$ where $f'_{ab}(X)$ is of degree t. This process allows to compute an unlimited number of multiplications but requires $\mathcal{O}(n^2)$ communications.

Note that, like for garbled circuits, addition is made for free whereas multiplication requires communications. Besides, remark that the number of rounds of communications is proportional to the depth of the circuit to evaluate. Therefore, secret sharing based solutions calls for new cryptographic primitives with minimal multiplicative and depth complexity, so that they can be computed efficiently via secret sharing.

There are plenty of schemes of secret sharing allowing computation over shares (see for example [DPSZ12]), but they all require a quadratic number of communications for performing a certain type of operation.

1.6 MPC-Friendly Primitives

Historically, the design of symmetric cryptographic primitives (such as block ciphers, pseudo-random generators, and pseudorandom functions) has been motivated by efficiency considerations (memory consumption, hardware compatibility, ease of implementation,...). The field of secure multi-party computation has led to the emergence of new efficiency considerations: the efficiency of secure evaluation of a symmetric primitive is strongly related to parameters such as its circuit depth, and the number of its AND gates. This is particularly true for garbled circuits and secret sharing as mentioned previously.

This observation has motivated the design of MPC-friendly symmetric primitives in several recent works (e.g. [ARS$^+$15, CCF$^+$16, MJSC16, GRR$^+$16]). Secure evaluation of such symmetric primitives enjoys a wide variety of applications.

Among many MPC-friendly primitives, Goldreich's pseudorandom generator has attracted a lot of interest due to its applications in many cryptographic constructions. A pseudorandom generator (PRG) maps a random seed to a longer pseudorandom string, with the guarantee that the output of the PRG cannot be distinguished from the uniform distribution. The Goldreich's pseudorandom generator can be defined as follows: let n be size of the seed, let m be the size of the output and let $(\sigma^1, \ldots, \sigma^m)$ be a list of m subsets of bits of the seed, such that each subset is of small size: for any $i \leq m$, $|\sigma^i| = d(n)$, where $d(n) \ll n$ (in actual instantiations, $d(n)$ can for example be logarithmic in n, or even constant). Fix a simple predicate $P : \{0,1\}^{d(n)} \mapsto \{0,1\}$, and define the function $f : \{0,1\}^n \mapsto \{0,1\}^m$ as follows: on input $x \in \{0,1\}^n$, for any subset S of $[n]$, let $x[\sigma]$ denote the subset of the bits of x indexed by σ. Compute $f(x)$ as $P(x[\sigma^1])|| \cdots ||P(x[\sigma^m])$ (that is, $f(x)$ is computed by applying the predicate P to all subsets of

the bits of x indexed by the sets $\sigma^1, \ldots, \sigma^m$).

This construction makes this PRG an interesting candidate for MPC-friendly since every output bit only depends on $d(n)$ bits of the seed. In practice $d(n)$ can be as small as five and the predicate can be limited to a single multiplication. In order to measure its efficiency, a study of concrete parameters of this PRG with some specific predicates is made in Chapter 4.

1.7 Regarding the Preprocessing Model

In this thesis, only the most general settings are covered but some more specific settings have been studied a lot by the cryptographic community. One of them particularly makes sense for multi-party computation: the *preprocessing model*.

In this model, the computation is separated in two phases: a preprocessing (or offline[3]) phase and an online phase. In the preprocessing phase, the participants do not know their inputs yet, but they have access to the function f to evaluate and they wish to compute everything that does not depend on the input (shared coin flipping, generation and transmission of a garbled circuit, ...). In the online phase, the parts that depend on the inputs are computed. It generally results in very efficient online phase.

This allows drastic efficiency improvement in several primitives, among which:

- *Oblivious transfer*: it is known since [Bea95] that oblivious transfers can be preprocessed on random inputs, and then "derandomized" in the online phase to OTs of chosen inputs.
- *Garbled Circuits*: the generation, the transmission and the OT phase of the garbled circuit protocol can be preprocessed. In the online phase, it just remains to derandomize the OTs and to evaluate the circuit. In the malicious setting, the cut-&-choose can also be made offline.
- *Secret sharing*: SPDZ [DPSZ12] is a special kind of secret sharing in the malicious adversary model, that takes advantage of the preprocessing model to preprocess multiplications over random data (that allows more efficient multiplications in the online phase) and proofs that no participant is deviating from the protocol.

As already noticed, this setting is out of the scope of the thesis, but the preliminaries of Chapter 2 and our results also apply in this model. This model is also one of the motivations of our studies in Chapter 4.

[3]The "offline" term can be misleading since it often requires interactions between the participants.

CHAPTER 2

ON THE LEAKAGE OF CORRUPTED GARBLED CIRCUITS

The seminal work of Yao [Yao86], known as garbled circuits, is a general solution to the secure two-party computation problem, which is extremely efficient in terms of rounds of communications, which is constant and optimal. The protocol designed by Yao, is asymmetric and involves two parties: the generator is responsible for creating the garbled circuit to be evaluated, and an evaluator is responsible for executing it on its inputs. Originally, it was designed in the semi-honest model, assuming that the generator correctly generates the circuit, and it was clear that a malicious generator could easily modify the logic gates of the garbled circuit before sending it to the evaluator for execution. Applying cut-&-choose to garbled circuits soon appeared to fix this issue, but requires to generate, transmit and evaluate a large number of garbled circuits.

Since then, a lot of work has been made to optimize both garbled circuits [BMR90, NPS99, KS08, ZRE15] and cut-&-choose based solutions [MF06, LP07, sS11, MR13, Lin13, sS13, AMPR14, WMK17]. Unfortunately, the best of these approaches still requires the generator to generate and transmit s garbled circuits for a statistical security of 2^{-s} against malicious adversaries, thus resulting in a serious overhead compared to the semi-honest model.

However, all these techniques based on cut-&-choose aim at avoiding any kind of modification on the circuit, without having to define them explicitely. Suprisingly, it has never been studied which modifications a malicious generator can make to a single garbled circuit, still leading to an accepted execution, and therefore why the cut-&-choose is necessary.

In this chapter, a detailed introduction to garbled circuits is first given. Hopefully, it should allow readers with a basic cryptographic background to be more familiar with this elegant tool. Our contribution comes after: we prove that, for a large class of circuits, the malicious generator is limited to add NOT gates on the wires of his choice. Hence, his possibilities are much more restricted than what we could have expected from the previous state of the art. Finally, we show some impacts of this result on real circuits and on cut-&-choose based solutions.

Contents

2.1 Preliminaries

Although garbled circuits have been heavily modified, optimized and formalized since the mid-80s, the original idea is due to the work of Andrew Yao [Yao86]. As mentioned in his paper, garbled circuits appear to be a general solution to the two-party computation problem, in the sense that it allows to solve this problem for any function f, even in the presence of semi-honest participants.

Informally, the generator G (one of the parties) is responsible for generating a garbled circuit representing the function f to evaluate, that one can see as an “encrypted" version of the circuit computing f. Roughly speaking, each wire of the circuit is associated two random keys (later called garbled keys) having hidden semantics 0 and 1. Then, “encrypted" truth tables (later called garbled gates or garbled truth tables) are provided to the evaluator E (the other party) to propagate garbled keys across gates, while keeping their semantics secret. Finally, E is responsible for evaluating it, so that he learns (and possibly returns) the result while keeping all intermediate values of this circuit secret.

The work of Beaver, Micali, and Rogaway [BMR90] is the first to introduce the term garbled circuit and also the first to give a construction based on symmetric primitives. Later, the first construction based on pseudorandom functions was given by Naor, Pinkas and Sumner [NPS99], allowing many other works to optimize garbled circuits and to make this tool very practical.

2.1.1 Formal Definition

Despite the fact that almost thirty years passed since the original work of Yao, the first general formalization of garbled circuits was made by Bellare, Hoang and Rogaway [BHR12].

A *garbling scheme* $\mathcal{G}$ consists in five components $\mathcal{G} = (\mathsf{Gb}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ev}, \mathsf{ev})$. A *garbling algorithm* Gb is a randomized algorithm that generates from the function $f : \{0,1\}^n \to \{0,1\}^m$ and a security parameter λ three functions (F, e, d): a *garbled function* F, an *encoding function* e and a *decoding function* d. An *encryption algorithm* Enc takes the input x of the function to evaluate and the encoding function e and returns a *garbled input* $X = e(x)$. This garbled input can be used along with F and the *evaluation algorithm* Ev to obtain the *garbled output* $Z = F(X)$. Finally, the decryption algorithm Dec and the decryption function d allow to decrypt this garbled output and to obtain the final result $z = f(x)$, which must be equal to the output of the *insecure evaluation algorithm* ev. Then, it is required that $f = d \circ F \circ e$. This decomposition of $\mathcal{G}$ is illustrated in Fig. 2.1.

The work of Bellare et al. [BHR12] also defines three security properties that capture the general case:

- *Privacy*: a party acquiring (F, X, d) does not learn more about the input x than the result

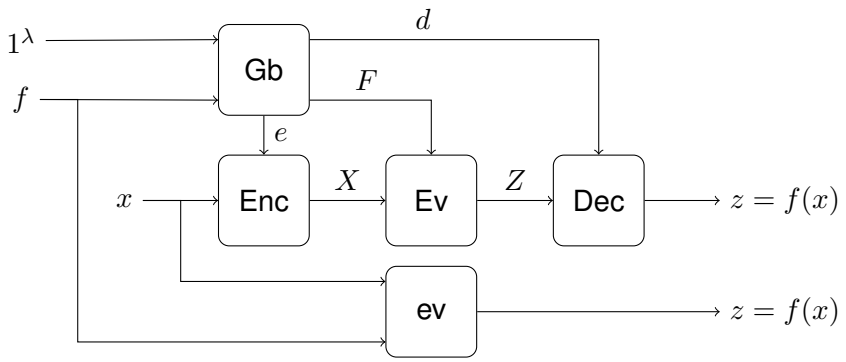


Figure 2.1: Components of a garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ev}, \mathsf{ev})$

$f(x)$ already allows to learn. Formally, there must exist a simulator $\mathcal{S}$ that takes as input $(1^\lambda, f, f(x))$ and whose output cannot be distinguished from (F, X, d) generated the usual way.

- *Obliviousness*: a party acquiring (F, X) does not learn any information about x. More specifically, there must exist a simulator $\mathcal{S}$ that takes as input $(1^\lambda, f)$ and whose output cannot be distinguished from (F, X) generated the usual way.

- *Authenticity*: a party acquiring (F, X) should be unable to produce a garbled output $Z^* \neq F(X)$, such that the decryption algorithm does not abort (i.e. $\mathsf{Dec}(d, Z^*) \neq \perp$), except with negligible probability.

In some specific cases, a garbling scheme may satisfy a subset only of these security properties, which may result in a more efficient solution.

Now that garbled circuits have been formally defined, let us see how it works in practice.

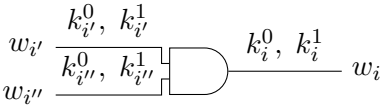
2.1.2 Simplest Garbling Scheme

In this part, a “de-optimized" version of [NPS99] is detailed. More specifically the version presented here is roughly equivalent to the original solution of Yao [Yao86], except that asymmetric operations are replaced by symmetric tools. This choice makes this part and the following optimizations much more accessible to non-specialists. Therefore, the presented scheme is only for educational purpose and, up to my knowledge, has never been published.

We consider a symmetric encryption scheme based on a hash function, noted H.

$$H : \mathbb{F}_{2^n} \to \mathbb{F}_{2^\lambda},\ n \geq \lambda$$

For a message $m \in \mathbb{F}_{2^\lambda}$, a key $k \in \mathbb{F}_{2^\lambda}$ and a salt i (of any size), the encryption function E



$H(k^0_{i'}\|\|k^0_{i''})$	$E^i_{k^0_{i'}}(E^i_{k^0_{i''}}(k^{g_i(0,0)}_i))$
$H(k^0_{i'}\|\|k^1_{i''})$	$E^i_{k^0_{i'}}(E^i_{k^1_{i''}}(k^{g_i(0,1)}_i))$
$H(k^1_{i'}\|\|k^0_{i''})$	$E^i_{k^1_{i'}}(E^i_{k^0_{i''}}(k^{g_i(1,0)}_i))$
$H(k^1_{i'}\|\|k^1_{i''})$	$E^i_{k^1_{i'}}(E^i_{k^1_{i''}}(k^{g_i(1,1)}_i))$

Figure 2.2: On the left side, a gate g_i with garbled keys at the input and output wires. On the right side, the corresponding non-shuffled garbled truth table.

of the scheme is defined as:

$$E^i_k(m) = m \oplus H(k|i)$$

The function f to evaluate is public and we assume that both parties already agreed on some public circuit representation of it C_f. We note x and y the respective inputs of the generator G and the evaluator E. Note that (x, y) refers to the input x defined in [BHR12] (see Section 2.1.1). We note w_i the i^{th} wire of C_f (for some arbitrary order), $\mathcal{I}$ the set of input wires of the circuit, $\mathcal{I}_X$ the input wires carrying x, $\mathcal{I}_Y$ the input wires carrying y ($\mathcal{I} = \mathcal{I}_X \cup \mathcal{I}_Y$) and $\mathcal{O}$ the set of output wires. Finally, we note g_i the gate that outputs w_i (unless $w_i \in \mathcal{I}$).

Garbling algorithm $(F, e, d) \leftarrow \mathsf{Gb}(1^\lambda, f)$: for each wire w_i of the circuit, the generator G randomly generates keys k^0_i and k^1_i (of size λ) having secret semantics 0 and 1. Let us call them the *garbled keys* for w_i. Then, for each gate g_i of the circuit, taking as input some wires $w_{i'}$ and $w_{i''}$, G generates a *garbled truth table* (also called *garbled table* or *garbled gate*) by hashing the input garbled keys and encrypting the outputs garbled keys with the corresponding input garbled keys. The identifier i of the gate is used as salt for the PRF used in the encryption scheme. The situation is illustrated in Fig. 2.2. The four rows of the garbled gate are then randomly shuffled by G. These garbled tables correspond to the garbled circuit F. Next, G computes the *commitment table* for each output wire of the circuit $w_i \in \mathcal{O}$. A commitment table is a mapping between the hashed garbled keys and their clear value, a trivial solution would be $(0, H(k^0_i)), (1, H(k^1_i))$. These commitment tables correspond to d and will be used to decrypt the result. Finally, garbled tables and commitment tables (in the formal definition F and d) are transmitted to the evaluator E. Note that G knows the garbled keys of all input wires $w_i \in \mathcal{I}$ and their clear value, which represent the encryption function e.

Encryption algorithm $X \leftarrow \mathsf{Enc}(e, (x, y))$: G knows e and x (his input). Therefore, for every wire $w_i \in \mathcal{I}_X$ carrying some input bit x_j of x, G can directly compute the garbled key $k^{x_j}_i$ and send it to E, while keeping the clear value of x private. These keys correspond to the first part of the garbled input X. The second part, that depends on the evaluator's private input y is a bit trickier to obtain. For every wire $w_i \in \mathcal{I}_Y$ carrying some input bit y_j of y, E wants to retrieve $k^{y_j}_i$ without revealing y_j, and G wants to ensure that only one of the two garbled keys k^0_i and

k_i^1 for w_i is learnt by E. Oblivious transfers allow to solve this issue. For efficiency reasons, OT-extension and random-OT[1] can be used. We refer the reader to Section 1.3.3 for more details about oblivious transfers and its optimizations.

Evaluation algorithm $Z \leftarrow \mathsf{Ev}(F, X)$: the evaluator E knows one (and only one) garbled key for each wire of $\mathcal{I}$ and these keys represent the input value x and y (although x is kept secret). He also knows the garbled table for each gate of the circuit and a commitment table for each wire of $\mathcal{O}$. Note that with a garbled table for g_i, as shown in Fig. 2.2, if E has two keys $k_{i'}^a$ and $k_{i''}^b$ for some hidden input bits a and b, then he is able to compute $k_i^{g_i(a,b)}$, while keeping a, b and $g_i(a,b)$ secret. More specifically, E computes and uses $H(k_{i'}^a||k_{i''}^b)$ to determine which row of the garbled table to decrypt. Then, he can decrypt $E^i_{k_{i'}^a}(E^i_{k_{i''}^b}(k_i^{g_i(a,b)}))$ to retrieve $k_i^{g_i(a,b)}$. To evaluate the entire circuit, E starts by evaluating, as just described, the first gate, the input garbled keys of which he already knows, and obtains the output garbled keys. He can then evaluate the entire circuit, one gate at a time, until he gets the garbled keys for the wires of $\mathcal{O}$ (that represent Z).

Decryption algorithm $z \leftarrow \mathsf{Dec}(d, Z)$: E has computed one garbled key $k_i^{z_j}$ for each wire $w_i \in \mathcal{O}$ (that represent Z) and has been given the commitment tables (for d) of the form $(0, H(k_i^0)), (1, H(k_i^1))$. Then, he can hash the garbled keys he has and reconstruct the clear result z, that must be equal to $f(x, y)$.

At the end of this decryption algorithm, E learns the result $z = f(x, y)$. Instead of naively returning this result to G, he returns Z (i.e. the garbled keys $k_i^{z_j}$ for each wire $w_i \in \mathcal{O}$). Since G knows d, he can also run the decryption algorithm. Thanks to the authenticity property of the garbling scheme, this convinces the generator G that z is indeed the correct evaluation of $f(x, y)$ and not an arbitrary value chosen by an adversarial evaluator.

An overview of the full protocol is presented in Fig. 2.3. Note that only three communications are necessary since the first one (the transmission of the garbled circuit) can be merged with the answer of the OT protocol. If it is not required that G learns the result, then only two are necessary, which is optimal in terms of rounds of communications.

The reason of the salt i used in the encryption scheme is to avoid linear dependencies between several garbled truth tables. As shown in Fig. 2.2, the same salt is used for an entire gate g_i. From this point, to simplify notations, this salt will be omitted unless necessary.

To sum up, Yao's garbled circuits are an elegant solution to solve the two-party computation problem, furthermore, in a constant number of rounds. However, the communication cost is linear in the size of the circuit to evaluate. Indeed, as shown previously, a garbled truth table has to be transmitted for each gate of the circuit. This transmission is the bottleneck of the protocol and decades of optimizations aimed at making it more efficient.

[1] Random-OT implies that the garbled keys of $\mathcal{I}_Y$ are known to the generator only after the OT protocol. Then, a part of the garbled circuit (the gates connected to $\mathcal{I}_Y$) cannot be pre-computed.

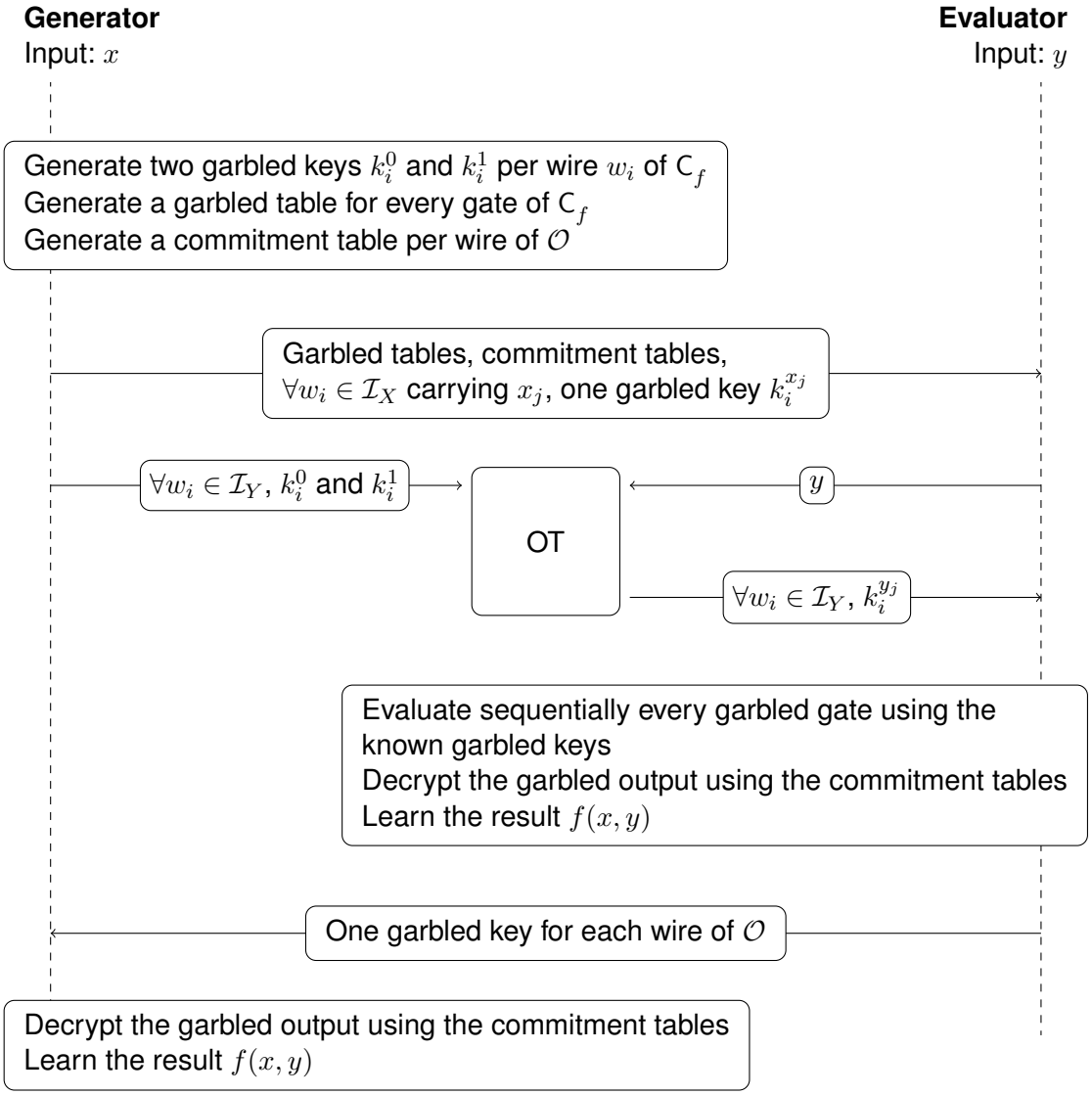
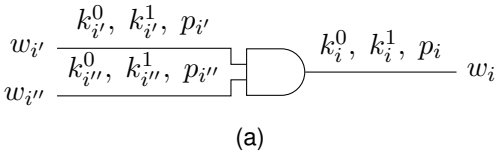


Figure 2.3: Garbled circuits protocol overview



(a)

~~0~~	~~0~~	$g_i(0,0)$
~~0~~	~~1~~	$g_i(0,1)$
~~1~~	~~0~~	$g_i(1,0)$
~~1~~	~~1~~	$g_i(1,1)$

(b)

$k_{i'}^0$	$k_{i''}^0$	$E_{k_{i'}^0}(E_{k_{i''}^0}(k_i^{g_i(0,0)}))$
$k_{i'}^0$	$k_{i''}^1$	$E_{k_{i'}^0}(E_{k_{i''}^1}(k_i^{g_i(0,1)}))$
$k_{i'}^1$	$k_{i''}^0$	$E_{k_{i'}^1}(E_{k_{i''}^0}(k_i^{g_i(1,0)}))$
$k_{i'}^1$	$k_{i''}^1$	$E_{k_{i'}^1}(E_{k_{i''}^1}(k_i^{g_i(1,1)}))$

(c)

~~$s(k_{i'}^{p_{i'}})=0$~~	~~$s(k_{i''}^{p_{i''}})=0$~~	$E_{k_{i'}^{p_{i'}}}(E_{k_{i''}^{p_{i''}}}(k_i^{g_i(p_{i'},p_{i''})}))$
~~$s(k_{i'}^{p_{i'}})=0$~~	~~$s(k_{i''}^{\overline{p_{i''}}})=1$~~	$E_{k_{i'}^{p_{i'}}}(E_{k_{i''}^{\overline{p_{i''}}}}(k_i^{g_i(p_{i'},\overline{p_{i''}})}))$
~~$s(k_{i'}^{\overline{p_i}})=1$~~	~~$s(k_{i''}^{p_{i''}})=0$~~	$E_{k_{i'}^{\overline{p_{i'}}}}(E_{k_{i''}^{p_{i''}}}(k_i^{g_i(\overline{p_{i'}},p_{i''})}))$
~~$s(k_{i'}^{\overline{p_{i'}}})=1$~~	~~$s(k_{i''}^{\overline{p_{i''}}})=1$~~	$E_{k_{i'}^{\overline{p_{i'}}}}(E_{k_{i''}^{\overline{p_{i''}}}}(k_i^{g_i(\overline{p_{i'}},\overline{p_{i''}})}))$

(d)

$E_{k_{i'}^{p_{i'}}}(E_{k_{i''}^{p_{i''}}}(k_i^{g_i(p_{i'},p_{i''})}))$
$E_{k_{i'}^{p_{i'}}}(E_{k_{i''}^{\overline{p_{i''}}}}(k_i^{g_i(p_{i'},\overline{p_{i''}})}))$
$E_{k_{i'}^{\overline{p_{i'}}}}(E_{k_{i''}^{p_{i''}}}(k_i^{g_i(\overline{p_{i'}},p_{i''})}))$
$E_{k_{i'}^{\overline{p_{i'}}}}(E_{k_{i''}^{\overline{p_{i''}}}}(k_i^{g_i(\overline{p_{i'}},\overline{p_{i''}})}))$

(e)

Figure 2.4: (a) a gate g_i with garbled keys and a permute bit at the input and output wires. (b) the clear truth table. (c) the unsecure garbled truth table. (d) the garbled truth table sorted by the select bit of the input keys. (e) the final garbled truth table.

2.1.3 The Point-and-Permute Trick

The point-and-permute trick of Beaver, Micali and Rogaway [BMR90] is an elegant optimization for garbled circuits to get rid of the input column of Fig. 2.2.

From now on, we call $s()$ the function that takes a garbled key as input and outputs the least significant bit of that key. This least significant bit of a garbled key is referred as *select bit*.

Consider the boolean gate g_i shown in Fig. 2.4.a and its corresponding (clear) truth table in Fig. 2.4.b. Observe that if one wanted to send a description of g_i, then the third column of Fig. 2.4.b would be sufficient, using the convention that the two first columns are sorted. The work of Beaver et al. [BMR90] manages to apply the same trick to garbled truth tables.

Roughly speaking, a random *permute bit* p_i is picked by G for every wire w_i of the circuit. A new constraint is added to the generation of the garbled keys: the select bit of a garbled key (the least significant bit[2]) is now the clear value masked with the permute bit. More specifically, the select bit of k_i^a is $a \oplus p_i$. Remark that it implies the select bit of every garbled key is arranged so

[2]Coding the select bit in the least significant bit of a key is a widespread convention. It could be any other bit or any other way of differentiating the two garbled keys of a wire. Without loss of generality, we use this convention in the rest of the chapter.

that the two garbled keys for a same wire have opposite select bits. Then we have the following facts:

$$
\begin{aligned}
&s(k_i^a) = a \oplus p_i \\
&s(k_i^{p_i}) = 0 \qquad\qquad s(k_i^{\overline{p_i}}) = 1 \\
&s(k_i^0) = p_i \qquad\qquad s(k_i^1) = \overline{p_i}
\end{aligned}
$$

Therefore, one can replace the values of the clear truth table (Fig. 2.4.b) by their respective garbled keys, as shown in Fig. 2.4.c. The resulting truth table is definitely insecure but can be sorted by the select bit (i.e. the least significant bit) of the input garbled keys. This situation is illustrated in Fig. 2.4.d. Using the convention that a garbled truth table is now sorted by the select bit of the input keys, the two first columns of Fig. 2.4.d become unnecessary. Then, G only has to generate and transmit four ciphertexts, as shown in Fig. 2.4.e. We stress that these four ciphertexts are no longer randomly shuffled. When evaluating, E just has to look at the select bit of the keys he knows to determine which one of the ciphertext he must decrypt.

More formally, the garbling algorithm $(F, e, d) \leftarrow \mathsf{Gb}(1^\lambda, f)$ is now changed as follows: for every wire w_i, G picks a random bit p_i, called permute bit. The select bit of a garbled key is the clear value masked with the permute bit. More specifically, the select bit of k_i^a is $a \oplus p_i$. Instead of being randomly shuffled, the garbled truth tables can be sorted by these select bits, as shown in Fig. 2.4.e. These tables can then be sent to E.

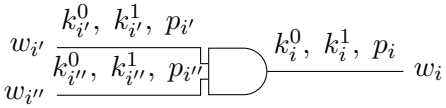
The evaluation algorithm $Z \leftarrow \mathsf{Ev}(F, X)$ has to be modified accordingly: when evaluating a gate, E uses the select bit of the garbled keys to determine which row he should decrypt. More specifically, for a gate g_i, if he has input garbled keys $k_{i'}^a$ and $k_{i''}^b$, then he can decrypt the $(2 \cdot s(k_{i'}^a) + s(k_{i''}^b))^{\text{th}}$ ciphertext and retrieve the relevant key $k_i^{g_i(a,b)}$.

Thanks to this optimization, the size of a garbled table is reduced to only four ciphertexts, thus reducing the communicational cost of garbled circuits by half.

2.1.4 The 25% Row-Reduction

Naor, Pinkas and Sumner [NPS99] introduced garbled row-reduction as a way of reducing the number of ciphertexts that describe a garbled gate. The main idea of their optimization is that instead of randomly picking two garbled keys for each wire w_i, one of them can be dependent of the input garbled keys of g_i (the gate that outputs w_i), and the garbling scheme still remains secure.

Concretely, the garbled keys are chosen such that the first ciphertext of a garbled truth table will always be the all-zeroes string and thus does not have to be transmitted. This situation is illustrated in Fig. 2.5. It implies that $k_i^{g_i(p_{i'}, p_{i''})}$ is the decryption of zero: $D_{k_{i''}^{p_{i''}}}(D_{k_{i'}^{p_{i'}}}(0))$.



~~$E_{k_{i'}^{p_{i'}}}(E_{k_{i''}^{p_{i''}}}(k_i^{g_i(p_{i'},p_{i''})})) = 0$~~
$E_{k_{i'}^{p_{i'}}}(E_{k_{i''}^{\overline{p_{i''}}}}(k_i^{g_i(p_{i'},\overline{p_{i''}})}))$
$E_{k_{i'}^{\overline{p_{i'}}}}(E_{k_{i''}^{p_{i''}}}(k_i^{g_i(\overline{p_{i'}},p_{i''})}))$
$E_{k_{i'}^{\overline{p_{i'}}}}(E_{k_{i''}^{\overline{p_{i''}}}}(k_i^{g_i(\overline{p_{i'}},\overline{p_{i''}})}))$

Figure 2.5: On the left side, a gate g_i with garbled keys and a permute bit at the input and output wires. On the right side, the corresponding garbled truth table with only three ciphertexts.

The garbling algorithm $(F, e, d) \leftarrow \mathsf{Gb}(1^\lambda, f)$ is therefore changed as follows: the generator G randomly chooses a permute bit and two garbled keys (according to the point-and-permute trick previously described) for every wire of $\mathcal{I}$. For every gate g_i taking as input some wires $w_{i'}$ and $w_{i''}$, G sets one of the garbled keys for w_i $k_i^{g_i(p_{i'},p_{i''})} = D_{k_{i''}^{p_{i''}}}(D_{k_{i'}^{p_{i'}}}(0))$, which also fixes the permute bit p_i. The other key $k_i^{\overline{g_i(p_{i'},p_{i''})}}$ is randomly chosen by G as usual (still according to the point-and-permute trick). G now generates the garbled truth tables as usual, except for the first ciphertext, which is the all-zeroes string by definition and thus does not have to be computed nor transmitted to the evaluator E.

Very few modifications are made to the evaluation algorithm $Z \leftarrow \mathsf{Ev}(F, X)$: for a gate g_i, if E has input garbled keys $k_{i'}^a$ and $k_{i''}^b$, if $(2 \cdot s(k_{i'}^a) + s(k_{i''}^b)) \neq 0$, then he can decrypt as usual. Otherwise, he simply assumes that the cipertext is the all-zeroes string and decrypts it.

Therefore, this optimization manages to lower the communication cost of garbled circuit by 25%. Since only one garbled key is randomly generated per wire, this work also drastically reduces the need for randomness generation, although the question of randomness generation is not explored further in this thesis.

Finally, another contribution of Naor, Pinkas and Sumner [NPS99] has to be mentioned: they describe a way to further reduce the size of garbled gates to only two ciphertexts, based on polynomial interpolation. Unfortunately, this second optimization is not compatible with the free-XOR trick that follows and is beaten by more recent works.

2.1.5 The Free-XOR Trick

The free-XOR trick of Kolesnikov and Schneider [KS08] allows to garble XOR gates for free. They observe that the difference between the two keys for a wire $k_i^0 \oplus k_i^1$ can be the same for all wires of the circuit, and the scheme still remains secure. The idea is to let G choose a *global offset* Δ that will be used to differentiate the two garbled keys for a same wire $k_i^0 \oplus k_i^1 = \Delta$. That way, when E has to evaluate a XOR gate, he just bitwise XOR the two input garbled keys to obtain the output garbled key. Note that, in order to make it compatible with the point-and-

permute technique, Δ has to be odd.

The garbling algorithm $(F, e, d) \leftarrow \mathsf{Gb}(1^\lambda, f)$ is optimized as follows: G starts by picking randomly an odd global offset Δ of size λ. For every wire $w_i \in \mathcal{I}$, he randomly chooses one of the garbled keys (say k_i^0 having hidden semantic 0), accordingly to the point-and-permute trick, and sets the other key to be the bitwise-XOR of the first and the global offset ($k_i^1 = k_i^0 \oplus \Delta$). For a non-XOR gate g_i, the key $k_i^{g_i(p_{i'}, p_{i''})}$ is computed as before (i.e. the decryption of zero), and the other key is set as $k_i^{\overline{g_i(p_{i'}, p_{i''})}} = k_i^{g_i(p_{i'}, p_{i''})} \oplus \Delta$. This also sets the permute bit p_i. The generation of garbled truth tables is made as usual for non-XOR gates. For a XOR gate g_i, having $w_{i'}$ and $w_{i''}$ as input, the two garbled keys for the output wire w_i are set at $k_i^0 = k_{i'}^0 \oplus k_{i''}^0$ and $k_i^1 = k_i^0 \oplus \Delta$. This implies that $p_i = p_{i'} \oplus p_{i''}$. No garbled truth tables are made for XOR gates.

For a XOR gate g_i, observe that:

$$\begin{cases} k_i^0 = k_{i'}^0 \oplus k_{i''}^0 \\ k_i^1 = k_i^0 \oplus \Delta \end{cases} \implies \begin{cases} k_i^0 = k_{i'}^0 \oplus k_{i''}^0 \\ k_i^0 = k_{i'}^1 \oplus k_{i''}^1 \\ k_i^1 = k_{i'}^1 \oplus k_{i''}^0 \\ k_i^1 = k_{i'}^0 \oplus k_{i''}^1 \end{cases}$$

Then, the evaluation algorithm $Z \leftarrow \mathsf{Ev}(F, X)$ has to be slightly modified: the evaluation of non-XOR gates remains unchanged. The evaluation of a XOR gates g_i is made by bitwise-XORing the input garbled keys together. If E has input garbled keys $k_{i'}^a$ and $k_{i''}^b$, the output garbled key is $k_i^{a \oplus b} = k_{i'}^a \oplus k_{i''}^b$.

Not only there is no ciphertext to transmit for XOR gates, but also the output garbled keys of any gate fully depend on the input keys and thus do not require any randomness to generate. Recursively, the garbled keys of the whole circuit fully depend on the garbled keys of $\mathcal{I}$.

Remark that it also implies that the random-OT optimization cannot be used any longer, since the two garbled keys of any wire of $\mathcal{I}_Y$ are correlated. Then the correlated-OT protocol should be used instead. Note that (as for the random-OT), the garbled keys of $\mathcal{I}_Y$, and thus the garbled keys of the entire circuit, are known to the generator only after the correlated-OT protocol. This means that the garbling algorithm comes after the exchange of the inputs. If this situation is not desirable, then the OT extension should be used alone. We refer the reader to Section 1.3.3 for more details about oblivious transfer.

In order to grasp the impact of this optimization, here are some statistics about real circuits, defined in [KSS09]. 66% of the gates of a multiplexer (of any size) are XOR gates. Thus the size of a multiplexer garbled circuit (in number of transmitted ciphertexts) is reduced by 66%. Similarly, the additioner circuit cost is reduced by 80%. The cost of the AES S-box, as designed in [BP11], is reduced by 73%.

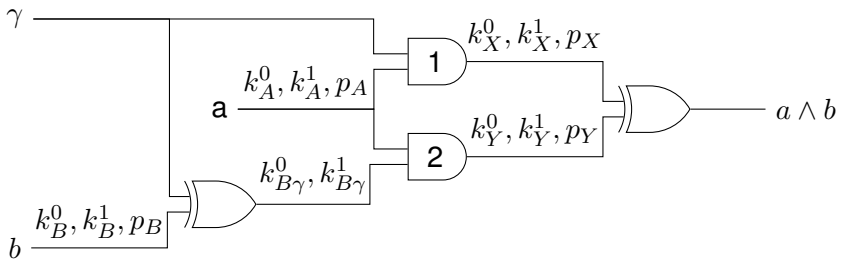


Figure 2.6: The two half-gates of an AND gate

2.1.6 The Two-Half-Gate Technique

More recently, Zahur, Rosulek and Evans [ZRE15] found a solution to reduce the size of a garbled gate to only two ciphertexts, and that remains compatible with the previous optimizations. In fact, this work is a very clever combination of the previous optimizations and is based on the following fact:

$$\forall \gamma \in \mathbb{F}_2, a \wedge b = \underbrace{(a \wedge \gamma)}_{\text{First half-gate}} \oplus \underbrace{(a \wedge (b \oplus \gamma))}_{\text{Second half-gate}}$$

Therefore, every AND gates of the circuit is implicitly replaced by the corresponding sub-circuit shown in Fig. 2.6, where γ is a bit randomly chosen by the generator G. A similar equation and a similar sub-circuit can be obtained from "any gate whose truth table contains an odd number of ones (e.g. AND, NAND, OR, NOR, etc.)." [ZRE15]. For simplicity, we only focus w.l.o.g. on AND gates.

An AND gate is then replaced by two AND gates and two (free) XOR gates. The two new AND gates are called half-gates and are defined as follows:

Definition 2.1 (Half-gate)**.** *AND gate for which one of the parties knows one of the inputs.*

Indeed, for the first half-gate $a \wedge \gamma$, G knows γ, and for the second half-gate $a \wedge (b \oplus \gamma)$, note that $b \oplus \gamma$ can be revealed to E without leaking b. At first sight, it does not look like an improvement, but the authors show that combining the previous optimizations in the particular case of half-gates, one can garble a half-gate with a single ciphertext. Thus, the total cost of the original AND gate is two ciphertexts.

In this section, we no longer need the symmetric encryption scheme E. Instead, we directly work with the hash function H.

$$H : \mathbb{F}_{2^n} \to \mathbb{F}_{2^\lambda},\ n \geq \lambda$$

Since an AND gate is now replaced by a fairly complex sub-circuit, we need to define a few notations that will be used to describe this optimization. We now call a and b the two

(presumably secret) input bits of the AND gate g_i and are carried by the wires w_A and w_B. $w_{B\gamma}$, w_X and w_Y respectively refer to the intermediate wires carrying $b \oplus \gamma$, $a \wedge \gamma$ and $a \wedge (b \oplus \gamma)$. According to these notations, the garbled keys and permute bits of these wires are as usual: $k^0_{B\gamma}$, $k^1_{B\gamma}$ and so on. All those notations are illustrated in Fig. 2.6. The wire carrying γ is fictive: it has no garbled key and no permute bit, this will be explained later.

Consider the first half-gate. G knows γ and we will use this knowledge to garble this gate with only two ciphertexts and then apply the previous optimizations to reduce it to only one. If $\gamma = 0$, then the half-gate must output k^0_X for any value of a. Therefore, we only need a two-ciphertext garbled table. If $\gamma = 1$, the half-gate must output k^a_X. We also need a two-ciphertext garbled table. In both cases, the garbled truth table is applied the point-and-permute trick (so that E can decrypt the $s(k^a_A)^{\text{th}}$ ciphertext) and the 25%-row reduction (so that the first ciphertext is the all-zeroes string). Therefore, only one ciphertext (that we call G) has to be transmitted. E can then evaluate it by decrypting either zero or G (depending on the value of $s(k^a_A)$) with the garbled key k^a_A that he knows.

The computation of $b \oplus \gamma$ is fictive. Since G knows γ, he can directly define $k^0_{B\gamma} = k^\gamma_B$, $k^1_{B\gamma} = k^{\overline{\gamma}}_B$ and $p_{B\gamma} = p_B \oplus \gamma$. Then, no generation or evaluation is needed. G gives $p_B \oplus \gamma$ to E, allowing him to learn the clear value $b \oplus \gamma$ of the garbled key $k^{b\oplus\gamma}_{B\gamma}$ obtained during the evaluation, without leaking the sensitive values b and γ.

Consider now the second half-gate. As just described, E knows $b \oplus \gamma$. If $b \oplus \gamma = 0$, then the half-gate must output 0. Then, the encryption of k^0_Y under $k^0_{B\gamma}$ must be provided. This first encryption is nullified, using the same trick as in the 25%-row reduction ($k^0_Y = H(k^0_{B\gamma})$). If $b \oplus \gamma = 1$, then the half-gate must output a. In that case, it gets trickier, G computes an encryption E of $k^0_Y \oplus k^0_A$ under $k^1_{B\gamma}$ ($E = k^0_Y \oplus k^0_A \oplus H(k^1_{B\gamma})$). Note that the point-and-permute trick is unnecessary since the evaluator E already knows the clear value of $b \oplus \gamma$. The ciphertext E is transmitted to E. When evaluating, if $b \oplus \gamma = 0$, he simply computes $k^0_Y = H(k^0_{B\gamma})$. Otherwise, he computes $k^{a\wedge(b\oplus\gamma)}_Y = E \oplus k^a_A \oplus H(k^1_{B\gamma})$ (this is correct since $k^a_A = k^0_A \oplus a\Delta$).

The generation of E and G is illustrated in Tab. 2.1. Very few modifications have to be made to this garbling algorithm to garble other "gate[s] whose truth table contains an odd number of ones (e.g. NAND, OR, NOR, etc.)." [ZRE15]. We will call later these gates *non-linear gates* in $\mathbb{F}_2$.

Table 2.1: Garbling the half-gates of an AND gate g_i

First half-gate		Second half-gate	
Garbled table if $\gamma = 0$	Garbled table if $\gamma = 1$	$b \oplus \gamma$	Garbled table
$k^0_X \oplus H(k^{p_A}_A \vert 2i) = 0$	$k^{p_A}_X \oplus H(k^{p_A}_A \vert 2i) = 0$	0	$k^0_Y \oplus H(k^0_{B\gamma} \vert 2i+1) = 0$
$k^0_X \oplus H(k^{\overline{p_A}}_A \vert 2i) = G$	$k^{\overline{p_A}}_X \oplus H(k^{\overline{p_A}}_A \vert 2i) = G$	1	$k^0_Y \oplus k^0_A \oplus H(k^1_{B\gamma} \vert 2i+1) = E$

Similarly to the previous sections, the identifier i of the gate is used as salt for the hash

function. More specifically, since we now have two (half-)gates, note that $2i$ and $2i+1$ are used respectively for the first and second half-gates. Unless necessary, we omit these salts in order to improve readability.

The evaluation of the half-gates is illustrated in Tab. 2.2. For gates other than AND, the evaluation remains exactly the same. Note that there are four different possible evaluations of the half-gates (one for each input combination), noted from K_1 to K_4. If E and G were correctly generated, then three of these results would collide, so that there are only two distinct garbled key for the output wire. For example, if g_i is an AND gate and if $p_A = 0$, then we should have $K_1 = K_2 = K_3$.

Table 2.2: Evaluating the half-gates of a gate g_i

Inputs		First half-gate	Second half-gate	Garbled output key
$k_A^{p_A}$	$k_{B\gamma}^0$	$H(k_A^{p_A})$	$H(k_{B\gamma}^0)$	$K_1 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0)$
$k_A^{p_A}$	$k_{B\gamma}^1$	$H(k_A^{p_A})$	$E \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A}$	$K_2 = E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A}$
$k_A^{\overline{p_A}}$	$k_{B\gamma}^0$	$G \oplus H(k_A^{\overline{p_A}})$	$H(k_{B\gamma}^0)$	$K_3 = G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0)$
$k_A^{\overline{p_A}}$	$k_{B\gamma}^1$	$G \oplus H(k_A^{\overline{p_A}})$	$E \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}}$	$K_4 = E \oplus G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}}$

2.1.7 Privacy-Free Garbled Circuits

Jawurek et al. [JKO13] demonstrated that garbled circuits can be used as a practical solution to zero-knowledge proof of knowledge protocols. The evaluator E (i.e. the prover) can prove any statement "$\exists x : f(x) = 1$" to the generator G (i.e. the verifier) without revealing x, using a single garbled circuit for f.

However, in this particular context, E has the input x and G has no input at all. Since E knows the entire input, he also knows the value of each intermediate wire of the garbled circuit, and thus there is no need to hide these values to E. Then, the privacy property as defined in Section 2.1.1 becomes unnecessary.

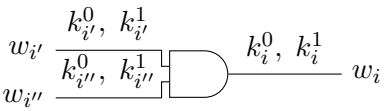
Frederiksen et al. [FNO15] showed that in this context, the size of the garbled circuits can be significantly reduced. The work of [ZRE15] provides an optimal garbling scheme in this context.

Since E knows every value, the garbled gates can be viewed as half-gates, and thus require a single ciphertext. More precisely, they are equivalent to the second half-gate, as defined in Section 2.1.6. For an AND gate, this ciphertext is $E = H(k_B^0) \oplus H(k_B^1) \oplus k_A^0$ and the evaluation algorithm is modified as presented in Tab. 2.3.

Although we focus only on the general case in the contributions of this chapter, they also apply to the privacy-free garbling scheme of [ZRE15]. Additionally, one of our constructions partially uses this specific scheme for efficiency reasons.

Table 2.3: Evaluating the privacy-free garbled gate

Inputs		Garbled output key
k_A^0	k_B^0	$K_1 = H(k_B^0)$
k_A^0	k_B^1	$K_2 = E \oplus H(k_B^1) \oplus k_A^0$
k_A^1	k_B^0	$K_3 = H(k_B^0)$
k_A^1	k_B^1	$K_4 = E \oplus H(k_B^1) \oplus k_A^1$



$H(k_{i'}^0 \| k_{i''}^0)$	$E_{k_{i'}^0}(E_{k_{i''}^0}(k_i^{g_i(0,0)}))$
$H(k_{i'}^0 \| k_{i''}^{1*})$	$E_{k_{i'}^0}(E_{k_{i''}^{1*}}(k_i^{g_i(0,1)}))$
$H(k_{i'}^1 \| k_{i''}^0)$	$E_{k_{i'}^1}(E_{k_{i''}^0}(k_i^{g_i(1,0)}))$
$H(k_{i'}^1 \| k_{i''}^{1*})$	$E_{k_{i'}^1}(E_{k_{i''}^{1*}}(k_i^{g_i(1,1)}))$

Figure 2.7: Example of selective failure attack. A corrupted garbled key $k_{i''}^{1*}$ is used for encryption in the garbled truth table instead of $k_{i''}^1$.

2.1.8 Corruption of Garbled Circuits

Now that we have seen how to garble a circuit, let us see how a malicious generator G can cheat. For simplicity, consider that the garbling scheme is unoptimized. There exists two kinds of corruptions: those that cannot be detected, since the evaluation always succeeds, and those that may lead the adversary to get caught, because of an invalid output (inconsistent with the commitments).

Selective Failure Attacks

We first consider the latter category, that leads to the so-called *selective failure attacks*. These are corruptions of the garbled circuit that make it executable only if a condition on internal values is met. If not, the protocol aborts: E does not obtain a correct output and thus cannot send back a result to G. Then, G learns whether the condition is met, but, if not, E detects the corruption and G gets caught. More specifically, the malicious G could use inconsistent keys to construct a garbled gate or to exchange inputs during the oblivious transfer step.

Let us see two examples, first, with the modification of an internal gate, and then with a corrupted oblivious transfer during the initialization step.

Alteration of an internal garbled gate. We consider an internal corrupted gate garbled as in Fig. 2.7. Suppose a key $k_{i''}^{1*}$ has been used for the generation of the garbled truth table instead of $k_{i''}^1$.

During the evaluation, if E gets $k_{i''}^1$, then he has no way of evaluating the gate and E is compelled to abort the protocol. If E gets $k_{i''}^0$, he can evaluate the gate as usual and does not

even notice the corruption.

If the protocol aborts, G learns that k_B^1 should have been used and E detects the attack. But if the protocol runs correctly, G learns the normal output, plus an internal bit k_B^0, and E does not detect it.

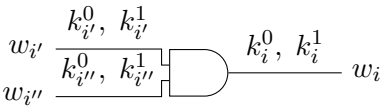
Note that if the point-and-permute technique is used, the attack is a bit different. E is always able to (possibly incorrectly) evaluate a gate. Therefore, after such a corrupted gate, E will get an inconsistent key, that will be used to evaluate the rest of the circuit. It will not be detected until the last gate of the circuit, the output of which will not match any value of the commitment table as defined in Section 2.1.2. Because he cannot return a valid output, E is forced to abort the protocol, and the leakage is exactly the same as previously described.

Remark that if G makes several such attacks in the circuits and if the protocol aborts, he does not know which attack (or both) lead to an abortion, which reduces the leakage of information. In that case, he learns much more information if the protocol succeeds.

Corruption during the oblivious transfer. We now consider E has some input bit b and G generates honestly the circuit using k_B^0 and k_B^1. However, during the oblivious transfer step, G uses k_B^0 and k_B^{1*}. Then, if $b = 1$, E gets an inconsistent key and the leakage of information is just as before. Note that the circuit itself is not modified, meaning that cut-&-choose based solutions (that will be introduced later) does not solve this issue. More specific and efficient solutions have been designed, such as *s-probe-resistant matrices* [LP07, sS13].

Roughly speaking, an s-probe-resistant matrix is a public Boolean matrix M used to encode the evaluator's input $y = My'$, such that every bit of y depends of at least s bits of y'. The function to evaluate now becomes $f'(x, y') = f(x, My')$. Then, E chooses a random y' that matches $y = My'$ and the rest of the garbled circuit protocol is performed using the function f'. It is now easy to see that G carries out $s' < s$ such selective failure attacks, then no information about y is leaked. Then, G has to perform at least s attacks, which the protocol to abort with probability $1-2^{-s}$ without leaking any meaningful information. In terms of efficiency, the garbling and the evaluation of M is free (it is only made of XOR gates), but the length of y' is slightly higher than the one of y. Different approaches aim at reducing this overcost.

Information vs. Detection. In both above cases, the malicious generator can be detected since the failure is part of the way to learn information. Hence, the adversary must make the protocol fail with non-negligible probability to learn something. In the rest of the chapter, we restrict the study to context where the potential gain of information is not worth the risk of getting caught by the honest party. Moreover, if the garbled circuit and the inputs were signed by the generator, the evaluator could easily prove to some authority that the garbled circuit is indeed non-executable. This seems reasonable in many real-life cases. We thus limit alterations



$H(k_{i'}^0 \|\| k_{i''}^0)$	$E_{k_{i'}^0}(E_{k_{i''}^0}(k_i^0))$
$H(k_{i'}^0 \|\| k_{i''}^1)$	$E_{k_{i'}^0}(E_{k_{i''}^1}(k_i^0))$
$H(k_{i'}^1 \|\| k_{i''}^0)$	$E_{k_{i'}^1}(E_{k_{i''}^0}(k_i^0))$
$H(k_{i'}^1 \|\| k_{i''}^1)$	$E_{k_{i'}^1}(E_{k_{i''}^1}(k_i^0))$

Figure 2.8: On the left side, a gate g_i supposed to be an AND gate. On the right side, the corrupted corresponding garbled table that always ouputs False.

to the garbled circuit that do never lead to a failure.

Undetectable Corruptions

In order to be undetectable, the corrupted circuit must keep the same topology and the outputs must match the values of the commitment tables.

For simplicity, consider the original garbling scheme of Section 2.1.2. A malicious generator G can easily make such a corruption by changing the functionality of a gate before garbling it. The example given in Fig. 2.8 shows how to turn (w.l.o.g.) an AND gate into (w.l.o.g.) a gate that always outputs False. Generalizing it to the whole circuit, a malicious generator can easily choose the output of the circuit. Similarly, he can also make the circuit output a part of (or even the entire) evaluator's private input. These modifications can be made arbitrarily by the generator and it will not be detected by the evaluator.

These two kinds of corruptions (selective failure attacks and undetectable corruptions) are traditionally prevented by cut-&-choose based solutions.

2.1.9 The Cut-&-Choose Paradigm

As described previously, if the generator G is malicious, he can construct a garbled circuit that computes a function that is different from the one that E and G agreed on, or he can make it abort under some conditions on E's inputs. A well-known approach for such problems in cryptography is the *cut-&-choose* technique.

Loosely speaking, in the context of garbled circuits, G generates a set of t garbled circuits for the function f and send them all to E. Then, E randomly chooses a subset of the t garbled circuits and asks G to *open* this subset. G complies and reveals all the garbled keys for the requested garbled circuits. E can then check that these garbled circuits are correctly garbled. If this verification fails, he can safely abort the protocol. Otherwise, the exchange of the inputs and the evaluation of the unopened garbled circuits is done as in the semi-honest case. Clearly, this solution solves the problem of G corrupting the garbled circuit. Indeed, an adversarial G now has to guess which subset will be randomly chosen by E. The number t of garbled circuits

depends on some *statistical* security parameter s [3].

However, it is not sufficient since it creates new problems within itself:

1. *Input consistency for the generator*: a malicious G can provide inconsistent input garbled keys for the t circuits. In other words, he can submit different values of x for the unopened circuits.

2. *Input consistency for the evaluator*: a malicious G can switch some garbled keys for an input wire of E for some circuits during the oblivious transfer phase (thus after the opening phase), resulting in different values of y (the difference of which is known to the adversary). Similarly, a malicious E can submit different values of y for the unopened circuits.

3. *Output consistency*: the evaluated circuits (i.e. those that are unopened) may output different results if some of them are corrupted by G. The protocol must specify what E should do since an abortion would leak one bit of information.

The state of the art provides a large number of solutions that answer these issues in different manners and that aim are reducing the number of garbled circuits t required to reach a security level s.

Input Consistency for the Generator

Two main approaches have been introduced to solve the first issue.

First, the work of Mohassel and Franklin [MF06] uses a commitment scheme with efficient proof-of-equality. This tool allows one to generate commitments (of zero and one in these settings) and then to efficiently prove that two (or more) commitments are for the same value (zero or one) without revealing any information. As shown in [Ped91], this scheme can be built from the ElGamal encryption scheme. Given a finite group $\mathcal{G}$ and a generator g, the commiter randomly picks $h \in \mathcal{G}$ and sends

$$\text{EGCommit}(h, m, r) = (g^r, h^r g^m) \ .$$

This commitment is computationally-hiding and perfectly-binding. Moreover, given two commitments EGCommit(h, m_1, r_1) and EGCommit(h, m_2, r_2), the commiter can prove the equality $m_1 = m_2$ by revealing $r_1 - r_2$. The verifier can then check that

$$\frac{h^{r_1} g^{m_1}}{h^{r_2} g^{m_2}} = h^{r_1 - r_2} \ .$$

[3] Small values of s are generally considered (i.e. 40 or 60) since an adversarial generator is given only one try to guess which circuits are to be opened and which circuits are to be evaluated.

Roughly speaking, the garbled keys for the input wires of G are no longer chosen randomly but are commitments of zero and one (i.e. EGCommit$(h, 0, r)$ and EGCommit$(h, 1, r)$). After the opening procedure, G can send a garbled key (i.e. a commitment EGCommit(h, x_i, r_j)) for each bit x_i and for each unopened circuit and then prove that all garbled keys for x_i commit to the same bit. The drawback of this solution is that the generation of the input garble keys requires some exponentiations, which is computationally more expensive

A different technique is presented by [LP07]: t sets of t pairs of commitments (without proof-of-equality) are generated by G for the two garbled keys for all input wires of G. All the $2t^2|x|$ commitments are transmitted to E. Then, cut-&-choose is also applied to the t sets. This solution does not use any exponentiation, but requires a large number of commitments. This solution is no longer used in more recent works and we refer the reader to this paper for more details.

Input Consistency for the Evaluator

A countermeasure against a malicious G trying to switch the garbled keys for an input bit y_i of E is to add a commitment table to the input wires of E. Or even more efficiently, another solution would be to fix the permute bit of these wires to zero. This allows E to check that all garbled keys he receives for his input y_i from the oblivious transfer phase have semantic values y_i.

The input consistency against a malicious evaluator can be forced by transmitting all the t garbled keys for the input bit y_i in a single oblivious transfer. Not only it solves this security issue, but it also makes the number of oblivious transfer protocols independent of t.

Output Consistency

In this part, we define what should happen if some evaluated circuits output different results, although all opened circuits were correctly garbled. It means that the adversarial generator has corrupted a few circuits and that these circuits were not opened. However, there are also some unopened circuits correctly garbled. This last issue is the trickiest one and has attracted a lot of interest.

The first solution, described in [MNP+04] is to have only one evaluated circuit and $t - 1$ opened circuits. This clearly solves the problem but is limited to very small values of the security parameter s, since it gives $t = 2^s$. Indeed, the adversary correctly guesses which circuit will be evaluated with probability $1/t$.

Later, several works [MF06, LP07, sS11] propose to open a constant fraction ct of the circuits and to output the majority result. In this case, the adversary must corrupt a majority of

evaluated circuits and none opened circuits. The probability of success of such an adversary is

$$\frac{\binom{t-ct}{\lceil\frac{t-ct}{2}\rceil}}{\binom{t}{\lceil\frac{t-ct}{2}\rceil}} < 2^{-s} .$$

The works of [MF06, LP07] originally specified a constant $c = 1/2$ but Shelat and Shen [sS11] later proved that an optimal setting is $c = 0.6$, which gives $t \approx 3s$.

More recently, Lindell [Lin13] designed an optimal solution that requires only $t = s$ garbled circuits. The core idea is to build a mechanism that allows E to learn the input x of the generator if and only if the evaluated circuits output different results (i.e. if G is malicious). Then, E learns x and can run an insecure evaluation $f(x, y)$ and return the honest result to the malicious G, who learns nothing more than in the honest case. Therefore, the adversary wins if he is able to guess which circuits will be opened and which will be evaluated. Moreover, the author suggests to open any circuit with probability $1/2$ (with the constraint that at least one is not opened). Then, the probability of success of the adversary is 2^{-s}. In [Lin13], this mechanism is based on small garbled circuits, which takes as input x and the output of the evaluated garbled circuits. However, these smaller garbled circuits also require to be secure against a malicious generator, thus the previous solution $t = 3s$ has to be applied.

Afshar et al. [AMPR14] get rid of this overcost by using zero-knowledge proofs and commitment schemes. Very briefly, the commitment scheme EGCommit is used to commit all input bits of G. This is no overcost, since it was already necessary for the input consistency issue.

$$\text{EGCommit}(h, x_i, r) = (g^r, h^r g^{x_i}) .$$

The garbled keys of the output wires of the circuits are arranged so that learning two different results allows E to retrieve the trapdoor $\log_g(h)$, and then the input x. Zero-knowledge proofs are used to convince E that obtaining two different results indeed allows to compute $\log_g(h)$.

2.2 Motivation of Our Work

As seen in the previous section, a lot of work has been made to optimize cut-&-choose based solutions, that aim at avoiding any kind of modification on the circuit. Nevertheless, it has never been studied which modifications a malicious generator can make to a single garbled circuit, still leading to an accepted execution, and then why the cut-&-choose is necessary.

Before the most recent general optimization of semi-honest garbling schemes of Zahur, Rosulek and Evans [ZRE15], such a study would have been meaningless. Indeed, it was obvious that an adversary could apply any modification of his choice as long as the topology of the circuit remains the same. Some examples are given in Section 2.1.8. In other words, any binary

gate could be turned into any other binary gate and the resulting corrupted garbled circuit would be still executable for any input.

However, the recent improvement of [ZRE15] manages to reduce the size of a garbled truth table to only two ciphers (instead of three since the work of Naor et al. [NPS99], or even four before that). Whereas this result can be seen as a nice improvement for an honest party, it is clearly an extra constraint for a malicious party, given that he can now change only two variables instead of three or four. Since then, it is not clear which modifications can actually be made, and we prove in this chapter that it is much more limited than suggested in the previous state-of-the-art.

More specifically, we prove that a malicious generator is limited to turn *non-linear gates* into other *non-linear gates*. We define non-linear gates and by opposition linear gates as follows:

Definition 2.2 (non-linear gate)**.** *A non-linear gate is "any gate whose truth table contains an odd number of ones (e.g. AND, NAND, OR, NOR, etc.)." [ZRE15]. A non-linear gate computes a non-linear operation in* $\mathbb{F}_2$*.*

Definition 2.3 (linear gate)**.** *A linear gate is a gate that computes a linear operation in* $\mathbb{F}_2$ *(e.g. XOR, XNOR, True, False, etc.).*

Then, such corruptions are equivalent to say that an adversary is only able to add NOT gates to a circuit or to allow abortion of the protocol.

The rest of this chapter is organized as follows: we first show how these modifications can be made and then prove that these are the only possible alteration. Finally, we show the impact of this contribution on real circuits.

2.3 Corruption of Optimized Garbled Circuits

We consider the garbling scheme of [ZRE15], as described in Section 2.1.6.

If G garbles the half-gates by switching some garbled keys, as shown in Tab. 2.4, it is easy to prove that the resulting gate computes $\bar{a} \wedge b$, and that the execution algorithm of E remains unchanged. Moreover, this modified garbled truth table is actually the correct way of garbling $\bar{a} \wedge b$.

Table 2.4: Turning $a \wedge b$ into $\bar{a} \wedge b$

First half-gate		Second half-gate	
Garbled table if $\gamma = 0$	Garbled table if $\gamma = 1$	$b \oplus \gamma$	Garbled table
$k_X^0 \oplus H(k_A^{p_A}) = 0$	$k_X^{\overline{p_A}} \oplus H(k_A^{p_A}) = 0$	0	$k_Y^0 \oplus H(k_{B\gamma}^0) = 0$
$k_X^0 \oplus H(k_A^{\overline{p_A}}) = G$	$k_X^{p_A} \oplus H(k_A^{\overline{p_A}}) = G$	1	$k_Y^0 \oplus k_A^1 \oplus H(k_{B\gamma}^1) = E$

Similarly, we show in Tab. 2.5 and Tab. 2.6 how to obtain a correct garbling of $a \wedge \bar{b}$ and $\overline{a \wedge b}$ from a corrupted AND gate.

Table 2.5: Turning $a \wedge b$ into $a \wedge \bar{b}$

First half-gate		Second half-gate	
Garbled table if $\gamma = 1$	Garbled table if $\gamma = 0$	$b \oplus \gamma$	Garbled table
$k_X^0 \oplus H(k_A^{p_A}) = 0$	$k_A^{p_A} \oplus H(k_A^{p_A}) = 0$	0	$k_Y^0 \oplus H(k_{B\gamma}^0) = 0$
$k_X^0 \oplus H(k_A^{\overline{p_A}}) = G$	$k_A^{\overline{p_A}} \oplus H(k_A^{\overline{p_A}}) = G$	1	$k_Y^0 \oplus k_A^0 \oplus H(k_{B\gamma}^1) = E$

Table 2.6: Turning $a \wedge b$ into $\overline{a \wedge b}$

First half-gate		Second half-gate	
Garbled table if $\gamma = 0$	Garbled table if $\gamma = 1$	$b \oplus \gamma$	Garbled table
$k_X^1 \oplus H(k_A^{p_A}) = 0$	$k_A^{\overline{p_A}} \oplus H(k_A^{p_A}) = 0$	0	$k_Y^0 \oplus H(k_{B\gamma}^0) = 0$
$k_X^1 \oplus H(k_A^{\overline{p_A}}) = G$	$k_A^{p_A} \oplus H(k_A^{\overline{p_A}}) = G$	1	$k_Y^0 \oplus k_A^0 \oplus H(k_{B\gamma}^1) = E$

Combining these three modifications, one can turn a AND gate into any of the eight non-linear gates. The example of the OR gate is also given Tab. 2.7, which is a combination of the three previous corruptions. Note that other ways exist to obtain the same results, but we chose these because they represent the honest ways of garbling $\bar{a} \wedge b$, $a \wedge \bar{b}$ and $\overline{a \wedge b}$, as described in [ZRE15].

Table 2.7: Turning $a \wedge b$ into $a \vee b$

First half-gate		Second half-gate	
Garbled table if $\gamma = 1$	Garbled table if $\gamma = 0$	$b \oplus \gamma$	Garbled table
$k_X^1 \oplus H(k_A^{p_A}) = 0$	$k_A^{p_A} \oplus H(k_A^{p_A}) = 0$	0	$k_Y^0 \oplus H(k_{B\gamma}^0) = 0$
$k_X^1 \oplus H(k_A^{\overline{p_A}}) = G$	$k_A^{\overline{p_A}} \oplus H(k_A^{\overline{p_A}}) = G$	1	$k_Y^0 \oplus k_A^1 \oplus H(k_{B\gamma}^1) = E$

These modifications can be made arbitrarily by the generator and it will not be detected by the evaluator unless a cut-&-choose solution is used. In the rest of the chapter, we are proving that no other modification can be made by a probabilistic polynomial-time adversary, or the protocol may abort.

2.4 Delimitation of the Corruption

Let us now prove that the above modifications and their combinations are the only ones that can be made by an adversarial generator G, if it does not want to get detected. We call f the function to evaluate and C_f a Boolean circuit representation of it.

We assume in this section that the (possibly corrupted) garbled circuit is executable for all inputs, since the adversary does not want to get detected.

Let us start with the obvious limitations. First, as already mentioned, the topology of the Boolean circuit to evaluate is public, which ensures that G cannot cheat on the number of gates or the way they are connected. Second, because of the free-XOR trick [KS08], XOR gates have no garbled truth tables to transmit, then they cannot be corrupted either.

But G can still garble "correctly" another circuit $\mathsf{C}_{f'}$ (computing some other function f' instead of f). By correct garbling, we mean that G garbles $\mathsf{C}_{f'}$ in accordance with the garbling algorithm (and its optimizations), and keeps the number of gates and the way they are connected to each other unchanged, as if f' was the correct function to evaluate. XOR gates of C_f must also be present in $\mathsf{C}_{f'}$. More specifically, we have the following restrictions :

1. Only two ciphers are sent for each non-linear gate.
2. XOR gates are not transmitted.
3. There is a global offset that differentiates the two garbled keys of each wire of $\mathsf{C}_{f'}$ (in accordance with the free-XOR trick [KS08]) and this offset is odd (as required by the point-and-permute technique [BMR90]).
4. $\mathsf{C}_{f'}$ is Boolean: for every wire of the circuit, there are two garbled keys.

The requirement of an odd offset follows from the fact that we took the convention that the select bit of a key is its least significant bit. If select bits were represented differently, the requirement would have to be changed accordingly.

It is obvious that the first two requirements are met. Otherwise, E will refuse to evaluate the circuit. In this section, we show that if the input wires of C_f are correctly garbled (i.e. have a common odd offset), then the rest of the circuit is also correctly garbled, or the protocol may abort. Thereafter, we provide a construction to ensure that input wires are correct. This will help to prove that the adversary is only able to turn a non-linear gate into another non-linear gate.

For the sake of simplicity, we consider that the original circuit is only composed of XOR and AND gates and we show later that the same result applies for the other gates.

Since the evaluation algorithm shown in Tab. 2.2 are the keystone of our proofs, we recall here in Tab. 2.8 a shorter version of it.

The correct generation of E and G are detailed in Section 2.1.6, but since we consider that G is malicious, we cannot make any assumption of their value.

2.4.1 Impossibility of Reducing the Number of Garbled Keys to One

The first thing to prove is that, for any garbled gate, there are at least two output garbled keys. Consider the case where an adversary wants to alter an AND gate (w.l.o.g.) so that it always

Table 2.8: Evaluating the half-gates

Select bits		Inputs		Garbled output key
0	0	$k_A^{p_A}$	$k_{B\gamma}^0$	$K_1 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0)$
0	1	$k_A^{p_A}$	$k_{B\gamma}^1$	$K_2 = E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A}$
1	0	$k_A^{\overline{p_A}}$	$k_{B\gamma}^0$	$K_3 = G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0)$
1	1	$k_A^{\overline{p_A}}$	$k_{B\gamma}^1$	$K_4 = E \oplus G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}}$

outputs True (or always False), whatever the inputs are. Then, he must choose E and G in Tab. 2.8, so that the four garbled output keys are equal. Then, we have the following system of equations:

$$\begin{cases} K_2 = K_1 \\ K_3 = K_1 \\ K_4 = K_1 \end{cases} \iff \begin{cases} E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ E \oplus G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \end{cases}$$

$$\iff \begin{cases} E = H(k_{B\gamma}^0) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} \\ G = H(k_A^{p_A}) \oplus H(k_A^{\overline{p_A}}) \\ k_A^{p_A} = k_A^{\overline{p_A}} \end{cases}$$

We thus proved here the following lemma:

Lemma 2.1. *For any garbled gate, if the first operand has two garbled keys with an odd offset, then the output wire has at least two possible garbled keys.*

Proof. If we indeed have $k_A^{p_A} \oplus k_A^{\overline{p_A}} = \Delta$ that is odd (i.e. the two garbled keys of the first operand have an odd offset), then we have $k_A^{p_A} \neq k_A^{\overline{p_A}}$ and the four keys cannot be equal. □

2.4.2 Impossibility of Three-Key Wires - Part 1

In the last part, we showed that if the input wires are correct, there are at least two garbled keys per wire. In this section, we aim at proving that there exists no wire having more than two possible garbled keys, while the circuit remains evaluable.

As described in Section 2.1.2, the garbled circuit is considered to have two commitments on the garbled keys of its output wires in $\mathcal{O}$. This ensures that output wires have at most two possible keys, or the protocol aborts when a third key is obtained. Then, if some wire of the circuit has three possible keys or more, then there must be a gate that reduces it to only two. We show that such a gate is impossible.

As defined in Section 2.1.3, $s()$ refers to the function that takes a garbled key as input and outputs the select bit of that key. This function tells the evaluator which line of Tab. 2.8 he should use while evaluating: $s(k_A^{p_A}) = 0$ and $s(k_A^0) = p_A$.

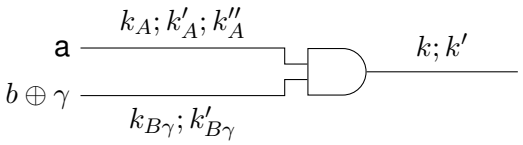


Figure 2.9: Reducing the number of keys of the first operand: Impossible

Since the previous notations are irrelevant if there are more than two keys or if the point-and-permute trick is not followed by the adversary, we now call k_X, k'_X the two distinct garbled key for a wire w_X, and k''_X a third garbled key when needed.

We remind that $H()$ is a hash function that is assumed to behave like a random function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^\lambda}$ and we expect the following problems to be computationally unfeasible by any polynomially bounded adversary :

1. Finding distinct $k_1, k'_1 \in \mathbb{F}_{2^\lambda}$, so that $H(k_1) = H(k'_1)$ requires $2^{\lambda/2}$ evaluations of $H()$ on average (Birthday paradox).

2. Finding distinct $k_1, k'_1 \in \mathbb{F}_{2^\lambda}$, so that $H(k_1) \oplus k_1 = H(k'_1) \oplus k'_1$ requires $2^{\lambda/2}$ evaluations of $H()$ on average (Equivalent to the birthday paradox).

3. For given i and j, finding $k_1, k_1^{'}, k_2, k_2^{'} \in \mathbb{F}_{2^\lambda}$, so that $k_1 \neq k_1^{'}$, $k_2 \neq k_2^{'}$ and $H(k_1|i) \oplus H(k_1^{'}|i) \oplus H(k_2|j) \oplus H(k_2^{'}|j) = 0$ requires $2^{\lambda/4}$ evaluation of $H()$ on average.

4. For given i and j, finding $k_1, k_1^{'}, k_2, k_2^{'} \in \mathbb{F}_{2^\lambda}$, so that $k_1 \neq k_1^{'}$, $k_2 \neq k_2^{'}$ and $H(k_1|i) \oplus k_1 \oplus H(k_1^{'}|i) \oplus k_1^{'} \oplus H(k_2|j) \oplus H(k_2^{'}|j) = 0$ requires $2^{\lambda/4}$ evaluations of $H()$ on average.

All these properties can be proven if H is modelled as a random oracle, using the birthday paradox bound. Note that in the definition of these problems, the adversary can freely choose the garbled keys k_1, $k_1^{'}$, k_2 and $k_2^{'}$, whereas for garbled gates, they are constrained by the garbling of the previous gates. Intuitively, solving these problems requires a lot more evaluations than listed above.

These properties lead to the following lemma, illustrated in Fig. 2.9:

Lemma 2.2. *For any garbled gate, if the first operand has at least three possible garbed keys, and the second has at least two, then the output wire has at least three garbled keys.*

Proof. We note k_A, k'_A and k''_A the three keys of the first operand and $k_{B\gamma}, k'_{B\gamma}$ the two keys of the second operand.

Suppose first that we have the following select bits $s(k_A) = s(k'_A) = 0$, $s(k_{B\gamma}) = 0$ and $s(k'_{B\gamma}) = 1$. We add no constraint on $s(k''_A)$. We can apply the evaluation algorithm for each

combination, as shown in Tab. 2.8, and we obtain the following set of garbled output keys:

$$\begin{cases} K_1 = H(k_A) \oplus H(k_{B\gamma}) \\ K_2 = E \oplus H(k_A) \oplus H(k'_{B\gamma}) \oplus k_A \\ K_3 = H(k'_A) \oplus H(k_{B\gamma}) \\ K_4 = E \oplus H(k'_A) \oplus H(k'_{B\gamma}) \oplus k'_A \end{cases}$$

We also have K_5 and K_6 that depends on k''_A. Then, the adversary has to reduce the number of keys to two. Thanks to the properties of the hash function, K_1 and K_3 are different. Then, he has to choose E that maps K_2 and K_4 to K_1 or K_3. If we try $K_2 = K_1$, then we get $K_4 = H(k'_A) \oplus H(k_{B\gamma}) \oplus k_A \oplus k'_A$.

Because the first operand has three keys, $k_A \neq k'_A$ and $K_4 \neq K_3$. Because of Property 2 of the hash function, K_4 cannot be mapped with K_1. A similar result would have been obtained if we first assumed $K_2 = K_3$. Then, the combination of select bits $s(k_A) = s(k'_A) = 0$, $s(k_{B\gamma}) = 0$ and $s(k'_{B\gamma}) = 1$ and any k''_A cannot be reduced to two garbled keys.

Let us see the case $s(k_A) = s(k'_A) = s(k_{B\gamma}) = s(k'_{B\gamma}) = 0$. We add no constraint on $s(k''_A)$. Then, we get:

$$\begin{cases} K_1 = H(k_A) \oplus H(k_{B\gamma}) \\ K_2 = H(k_A) \oplus H(k'_{B\gamma}) \\ K_3 = H(k'_A) \oplus H(k_{B\gamma}) \\ K_4 = H(k'_A) \oplus H(k'_{B\gamma}) \end{cases}$$

We also have K_5 and K_6 that depends on k''_A. Then, the adversary has to reduce the number of keys to two. Thanks to Properties 1 and 3 of the hash function, K_1, K_2, K_3 and K_4 are different.

Let us see the case $s(k_A) = s(k'_A) = s(k_{B\gamma}) = s(k'_{B\gamma}) = 1$. We add no constraint on $s(k''_A)$. Then, we get:

$$\begin{cases} K_1 = E \oplus G \oplus H(k_A) \oplus H(k_{B\gamma}) \oplus k_A \\ K_2 = E \oplus G \oplus H(k_A) \oplus H(k'_{B\gamma}) \oplus k_A \\ K_3 = E \oplus G \oplus H(k'_A) \oplus H(k_{B\gamma}) \oplus k'_A \\ K_4 = E \oplus G \oplus H(k'_A) \oplus H(k'_{B\gamma}) \oplus k'_A \end{cases}$$

We also have K_5 and K_6 that depends on k''_A. Then, the adversary has to reduce the number of keys to two. Note that in this particular case, the choice of E and G has no impact on the number of distinct keys. From Property 1 of the hash function, we know that $K_1 \neq K_2$. Property 2 allows to claim $K_1 \neq K_3$ and Property 4 $K_2 \neq K_3$. Then, K_1, K_2 and K_3 are different. In this particular case, it is even possible to prove that there cannot be less than six distinct keys.

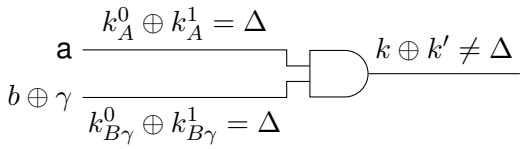


Figure 2.10: Modification of the offset: Impossible

Finally, let us see the case $s(k_A) = s(k'_A) = 1$, $s(k_{B\gamma}) = 0$ and $s(k'_{B\gamma}) = 1$. We add no constraint on $s(k''_A)$. Then, we get:

$$\begin{cases} K_1 = G \oplus H(k_A) \oplus H(k_{B\gamma}) \\ K_2 = E \oplus G \oplus H(k_A) \oplus H(k'_{B\gamma}) \oplus k_A \\ K_3 = G \oplus H(k'_A) \oplus H(k_{B\gamma}) \\ K_4 = E \oplus G \oplus H(k'_A) \oplus H(k'_{B\gamma}) \oplus k'_A \end{cases}$$

We also have K_5 and K_6 that depends on k''_A. Then, the adversary has to reduce the number of keys to two. Thanks to Property 1 of the hash function, K_1 and K_3 are different. Then, he has to choose E that maps K_2 and K_4 to K_1 or K_3. We describe below the case $K_2 = K_1$ and show that K_4 does not map any other key.

$$\begin{aligned} K_2 = K_1 &\iff E = H(k_{B\gamma}) \oplus H(k'_{B\gamma}) \oplus k_A \\ &\iff K_4 = G \oplus H(k'_A) \oplus H(k_{B\gamma}) \oplus k_A \oplus k'_A \end{aligned}$$

Because of Property 2 of the hash function, K_4 cannot be mapped with K_1 and $K_4 \neq K_3$ since $k_A \neq k'_A$. A similar result would have been obtained if we first assumed $K_2 = K_3$.

All other cases are changes of variables of the already studied cases, which ends the proof of Lemma 2.2. □

2.4.3 Impossibility of Three-Key Wires - Part 2

In this part, we study the opposite problem, where the second operand has at least three garbled keys and the first has at least two. The proof being trickier, we need to demonstrate Lemma 2.3 as a preliminary step.

Lemma 2.3. *For any gate, if the operands have two garbled keys and have the same odd offset, then the output wire has the same offset or at least three keys.*

Proof. This situation is illustrated in Fig. 2.10. Consider the case where the adversary wants to corrupt a garbled AND gate (w.l.o.g.) so that the offset is altered in the process. Then, he must choose such E and G in Tab. 2.8. We prove here that it cannot be done. As stated in

Section 2.1.6, there are four evaluation algorithms, the output of which, noted K_1 to K_4 collide so that there are at least two distinct results (from Lemma 2.2, the case $K_1 = K_2 = K_3 = K_4$ is already proven to be impossible).

First, let us see the case $K_1 = K_2 = K_3$. As expected, the output wire has the same odd offset Δ:

$$\begin{cases} K_1 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ K_2 = E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} = K_1 \\ K_3 = G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) = K_1 \\ K_4 = E \oplus G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} \end{cases} \Rightarrow \begin{cases} K_1 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ E = H(k_{B\gamma}^0) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} \\ G = H(k_A^{p_A}) \oplus H(k_A^{\overline{p_A}}) \\ K_4 = K_1 \oplus \Delta \end{cases}$$

The same result is obtained from the case $K_1 = K_2 = K_4$:

$$\begin{cases} K_1 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ K_2 = E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} = K_1 \\ K_3 = G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) \\ K_4 = E \oplus G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} = K_1 \end{cases} \Rightarrow \begin{cases} K_1 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ E = H(k_{B\gamma}^0) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} \\ G = H(k_A^{p_A}) \oplus H(k_A^{\overline{p_A}}) \oplus \Delta \\ K_3 = K_1 \oplus \Delta \end{cases}$$

The same result is obtained from the case $K_1 = K_3 = K_4$:

$$\begin{cases} K_1 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ K_2 = E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} \\ K_3 = G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) = K_1 \\ K_4 = E \oplus G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} = K_1 \end{cases} \Rightarrow \begin{cases} K_1 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ E = H(k_{B\gamma}^0) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} \\ G = H(k_A^{p_A}) \oplus H(k_A^{\overline{p_A}}) \\ K_2 = K_1 \oplus \Delta \end{cases}$$

The same result is obtained from the case $K_2 = K_3 = K_4$.

$$\begin{cases} K_1 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ K_2 = E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} \\ K_3 = G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) = K_2 \\ K_4 = E \oplus G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} = K_2 \end{cases}$$

$$\Longrightarrow \begin{cases} K_1 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ K_2 = E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} \\ E = G \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) \\ K_4 = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \oplus \Delta \implies K_4 = K_1 \oplus \Delta \end{cases}$$

We now prove that all other cases appear to be impossible.

$$\begin{cases} K_1 = K_2 \\ K_3 = K_4 \end{cases} \Leftrightarrow \begin{cases} E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) = E \oplus G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} \end{cases}$$
$$\Leftrightarrow \begin{cases} E = H(k_{B\gamma}^0) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} \\ E = H(k_{B\gamma}^0) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} \end{cases}$$

This implies that $k_A^{p_A} = k_A^{\overline{p_A}}$ and $\Delta = 0$. This is not possible since the operands have two distinct garbled keys and since the offset Δ is odd.

$$\begin{cases} K_1 = K_4 \\ K_2 = K_3 \end{cases} \Leftrightarrow \begin{cases} E \oplus G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) = E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} \end{cases}$$
$$\Leftrightarrow \begin{cases} E \oplus G = H(k_A^{p_A}) \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} \\ E \oplus G = H(k_A^{p_A}) \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} \end{cases}$$
$$\Leftrightarrow \Delta = 0$$

$$\begin{cases} K_1 = K_3 \\ K_2 = K_4 \end{cases} \Leftrightarrow \begin{cases} G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^0) = H(k_A^{p_A}) \oplus H(k_{B\gamma}^0) \\ E \oplus H(k_A^{p_A}) \oplus H(k_{B\gamma}^1) \oplus k_A^{p_A} = E \oplus G \oplus H(k_A^{\overline{p_A}}) \oplus H(k_{B\gamma}^1) \oplus k_A^{\overline{p_A}} \end{cases}$$
$$\Leftrightarrow \begin{cases} G = H(k_A^{p_A}) \oplus H(k_A^{\overline{p_A}}) \\ G = H(k_A^{p_A}) \oplus H(k_A^{\overline{p_A}}) \oplus k_A^{p_A} \oplus k_A^{\overline{p_A}} \end{cases}$$
$$\Leftrightarrow \Delta = 0$$

In the case of a XOR gate, the offset is also propagated, since garbled keys are simply XORed together.

$$\begin{cases} K_1 = k_A^0 \oplus k_B^0 \\ K_2 = k_A^0 \oplus k_B^0 \oplus \Delta = K_1 \oplus \Delta \\ K_3 = k_A^0 \oplus \Delta \oplus k_B^0 = K_1 \oplus \Delta \\ K_4 = k_A^0 \oplus \Delta \oplus k_B^0 \oplus \Delta = K_1 \end{cases}$$

This ends the proof of Lemma 2.3.

□

We now aim at concluding the last case with the following lemma:

Lemma 2.4. *If the input wires of the circuit have garbled keys with an odd global offset, then the garbled circuit cannot have a gate such that the second operand has at least three possible garbed keys, and the first has at least two, while the output wire has only two garbled keys.*

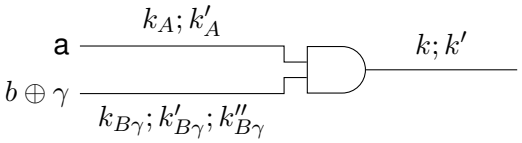


Figure 2.11: Reducing the number of keys of the second operand: Impossible

Proof. From Lemma 2.2, we know this is true if the two operands have at least three keys. We thus focus to the case where the first operand has two keys and the second operand has three keys, as illustrated in Fig. 2.11. For this proof, we consider that the input wires of $\mathcal{I}$ are correctly garbled : these wires have two garbled keys and they have an odd global offset Δ. We study the case of the first gate, called $\mathcal{F}$, of the circuit (in topological order) that has two garbled inputs for the first operand and three (or more) for the second.

Since $\mathcal{F}$ is the first of its kind in the circuit and because of Lemma 2.2, the sub-circuit that links the inputs of the circuit to the first operand wire of $\mathcal{F}$ have only wires with exactly two garbled keys. Moreover, since all input wires of this sub-circuit have the global offset Δ and because of Lemma 2.3, all wires of the sub-circuit, including the first operand of $\mathcal{F}$, have this same odd offset Δ.

Remark that an input wire of the circuit cannot have three keys. Then the three keys (or more) of the second operand of $\mathcal{F}$ come from a corrupted gate $\mathcal{F}'$ that outputs three distinct keys (or more). However, the two operand wires of $\mathcal{F}'$ have two possible garbled keys, and, with a similar approach, we can show that they have the same offset Δ as the first operand of $\mathcal{F}$.

Using the same convention as before, we call k_A and k'_A the keys of the first operand and $k_{B\gamma}$, $k'_{B\gamma}$ and $k''_{B\gamma}$ the keys of the second operand. As stated above, the computation of $k_{B\gamma}$, $k'_{B\gamma}$ and $k''_{B\gamma}$ engages the choice of Δ, and consequently $k_A \oplus k'_A$. We develop here the trickiest case $s(k_A) = s(k_{B\gamma}) = s(k'_{B\gamma}) = 0$ and $s(k'_A) = s(k''_{B\gamma}) = 1$, which gives the following set of keys:

$$\begin{cases} K_1 = H(k_A) \oplus H(k_{B\gamma}) \\ K_2 = H(k_A) \oplus H(k'_{B\gamma}) \\ K_3 = E \oplus H(k_A) \oplus H(k''_{B\gamma}) \oplus k_A \\ K_4 = G \oplus H(k'_A) \oplus H(k_{B\gamma}) \\ K_5 = G \oplus H(k'_A) \oplus H(k'_{B\gamma}) \\ K_6 = E \oplus G \oplus H(k'_A) \oplus H(k''_{B\gamma}) \oplus k'_A \end{cases}$$

Thanks to the property of the hash function, K_1 and K_2 are different. Then, the adversary must choose E and G, so that K_3 to K_6 collide with K_1 or K_2. Let us first consider the case $K_4 = K_1$.

$$K_4 = K_1 \implies \begin{cases} G = H(k_A) \oplus H(k'_A) \\ K_1 = K_4 = H(k_A) \oplus H(k_{B\gamma}) \\ K_2 = K_5 = H(k_A) \oplus H(k'_{B\gamma}) \\ K_3 = E \oplus H(k_A) \oplus H(k''_{B\gamma}) \oplus k_A \\ K_6 = E \oplus H(k_A) \oplus H(k''_{B\gamma}) \oplus k'_A \end{cases}$$

Then we have two more cases to enumerate : $K_3 = K_1$ and $K_3 = K_2$.

$$\begin{cases} K_4 = K_1 \\ K_3 = K_1 \end{cases} \implies \begin{cases} G = H(k_A) \oplus H(k'_A) \\ E = H(k_{B\gamma}) \oplus H(k''_{B\gamma}) \oplus k_A \\ K_1 = K_3 = K_4 = H(k_A) \oplus H(k_{B\gamma}) \\ K_2 = K_5 = H(k_A) \oplus H(k'_{B\gamma}) \\ K_6 = H(k_A) \oplus H(k_{B\gamma}) \oplus k_A \oplus k'_A \end{cases}$$

Then, K_6 is different from K_1 since Δ is odd and thus non-zero. Matching K_6 and K_2 is computationally unfeasible since it would require that $H(k_{B\gamma}) \oplus H(k'_{B\gamma}) = \Delta$ and we demonstrated above that the values of $k_{B\gamma}$ and $k'_{B\gamma}$ commits the value of Δ. The same result can be obtained if we assumed that $K_4 = K_2$ and/or $K_3 = K_2$

Let us now see the case $s(k_A) = s(k_{B\gamma}) = 0$ and $s(k'_A) = s(k'_{B\gamma}) = s(k''_{B\gamma}) = 1$. Using the four evaluation algorithms on each of the combinations, we have the following set of keys:

$$\begin{cases} K_1 = H(k_A) \oplus H(k_{B\gamma}) \\ K_2 = E \oplus H(k_A) \oplus H(k'_{B\gamma}) \oplus k_A \\ K_3 = E \oplus H(k_A) \oplus H(k''_{B\gamma}) \oplus k_A \\ K_4 = G \oplus H(k'_A) \oplus H(k_{B\gamma}) \\ K_5 = E \oplus G \oplus H(k'_A) \oplus H(k'_{B\gamma}) \oplus k'_A \\ K_6 = E \oplus G \oplus H(k'_A) \oplus H(k''_{B\gamma}) \oplus k'_A \end{cases}$$

Thanks to the property of the hash function, K_2 and K_3 are different. Then, the adversary must choose E and G, so that K_1 and K_4 to K_6 collide with K_2 or K_3. Let us first consider the

case $K_1 = K_2$, then the system of keys becomes:

$$K_1 = K_2 \implies \begin{cases} E = H(k_{B\gamma}) \oplus H(k'_{B\gamma}) \oplus k_A \\ K_1 = K_2 = H(k_A) \oplus H(k_{B\gamma}) \\ K_3 = H(k_A) \oplus H(k_{B\gamma}) \oplus H(k'_{B\gamma}) \oplus H(k''_{B\gamma}) \\ K_4 = G \oplus H(k'_A) \oplus H(k_{B\gamma}) \\ K_5 = G \oplus H(k'_A) \oplus H(k_{B\gamma}) \oplus k'_A \oplus k_A \\ K_6 = G \oplus H(k'_A) \oplus H(k_{B\gamma}) \oplus H(k'_{B\gamma}) \oplus H(k''_{B\gamma}) \oplus k'_A \oplus k_A \end{cases}$$

Then we have two more cases to enumerate : $K_4 = K_2$ and $K_4 = K_3$.

$$\begin{cases} K_1 = K_2 \\ K_4 = K_2 \end{cases} \implies \begin{cases} E = H(k_{B\gamma}) \oplus H(k'_{B\gamma}) \oplus k_A \\ G = H(k_A) \oplus H(k'_A) \\ K_1 = K_2 = K_4 = H(k_A) \oplus H(k_{B\gamma}) \\ K_3 = H(k_A) \oplus H(k_{B\gamma}) \oplus H(k'_{B\gamma}) \oplus H(k''_{B\gamma}) \\ K_5 = H(k_A) \oplus H(k_{B\gamma}) \oplus k'_A \oplus k_A \\ K_6 = H(k_A) \oplus H(k_{B\gamma}) \oplus H(k'_{B\gamma}) \oplus H(k''_{B\gamma}) \oplus k'_A \oplus k_A \end{cases}$$

$$\begin{cases} K_1 = K_2 \\ K_4 = K_3 \end{cases} \implies \begin{cases} E = H(k_{B\gamma}) \oplus H(k'_{B\gamma}) \oplus k_A \\ G = H(k_A) \oplus H(k'_A) \oplus H(k'_{B\gamma}) \oplus H(k''_{B\gamma}) \\ K_1 = K_2 = H(k_A) \oplus H(k_{B\gamma}) \\ K_3 = K_4 = H(k_A) \oplus H(k_{B\gamma}) \oplus H(k'_{B\gamma}) \oplus H(k''_{B\gamma}) \\ K_5 = H(k_A) \oplus H(k_{B\gamma}) \oplus H(k'_{B\gamma}) \oplus H(k''_{B\gamma}) \oplus k'_A \oplus k_A \\ K_6 = H(k_A) \oplus H(k_{B\gamma}) \oplus k'_A \oplus k_A \end{cases}$$

Note that in the two cases, K_5 and K_6 are simply switched around. Then, we only focus on the former case. K_5 is different from K_2 since Δ is odd and thus non-zero. Matching K_6 and K_3 is computationally unfeasible since it would require that $H(k_{B\gamma}) \oplus H(k'_{B\gamma}) = \Delta$ and we demonstrated above that the values of $k_{B\gamma}$ and $k'_{B\gamma}$ commits the value of Δ. The same result would be obtained if we assumed first that $K_1 = K_3$. Indeed, it would only be a permutation of the keys $k'_{B\gamma}$ and $k''_{B\gamma}$.

Let us see the case $s(k_A) = s(k_{B\gamma}) = s(k'_{B\gamma}) = s(k''_{B\gamma}) = 0$ and $s(k'_A) = 1$. Using the four

evaluation algorithms on each of the combinations, we have the following set of keys:

$$\begin{cases} K_1 = H(k_A) \oplus H(k_{B\gamma}) \\ K_2 = H(k_A) \oplus H(k'_{B\gamma}) \\ K_3 = H(k_A) \oplus H(k''_{B\gamma}) \\ K_4 = G \oplus H(k'_A) \oplus H(k_{B\gamma}) \\ K_5 = G \oplus H(k'_A) \oplus H(k'_{B\gamma}) \\ K_6 = G \oplus H(k'_A) \oplus H(k''_{B\gamma}) \end{cases}$$

This case is easier since the three first keys are different because of the properties of the hash function.

Let us see the case $s(k_A) = 0$ and $s(k'_A) = s(k_{B\gamma}) = s(k'_{B\gamma}) = s(k''_{B\gamma}) = 1$. Using the four evaluation algorithms on each of the combinations, we have the following set of keys:

$$\begin{cases} K_1 = E \oplus H(k_A) \oplus H(k_{B\gamma}) \oplus k_A \\ K_2 = E \oplus H(k_A) \oplus H(k'_{B\gamma}) \oplus k_A \\ K_3 = E \oplus H(k_A) \oplus H(k''_{B\gamma}) \oplus k_A \\ K_4 = E \oplus G \oplus H(k'_A) \oplus H(k_{B\gamma}) \oplus k'_A \\ K_5 = E \oplus G \oplus H(k'_A) \oplus H(k'_{B\gamma}) \oplus k'_A \\ K_6 = E \oplus G \oplus H(k'_A) \oplus H(k''_{B\gamma}) \oplus k'_A \end{cases}$$

The three first keys are different because of the properties of the hash function.

So far, we studied all the cases where $s(k_A) = 0$ and $s(k'_A) = 1$ (and by change of variables $s(k_A) \neq s(k'_A)$). All cases where $s(k_A) = s(k'_A)$ is equivalent to one already seen in the proof of Lemma 2.2 (it follows from the fact that we do not use the third key k''_A in the referred proof), which ends the proof of Lemma 2.4.

□

2.4.4 Impossibility of Turning a Non-Linear Gate into a Linear Gate

In Section 2.3, we showed how to turn a non-linear gate into any other non-linear gate. We will now prove that, since an adversarial generator is limited to Boolean circuits and cannot deviate from the global offset, he cannot turn a non-linear gate into a linear gate. We focus on the case of an AND gate.

Lemma 2.5. *For any non-linear gate, if the two operands have two garbled keys and have the same odd offset, then it cannot be turned into a linear gate.*

Proof. All cases are already studied in other proofs:

- The case $K_1 = K_2 = K_3 = K_4$ (the gate that output True or False) is already shown to be unsolvable from Lemma 2.1.
- The case $K_1 = K_2$ and $K_3 = K_4$ (the gate that always output the first operand a or always its negation $\bar{a}$) is proved to be impossible in the proof of Lemma 2.3.
- The case $K_1 = K_3$ and $K_2 = K_4$ (the gate that always output the second operand b or always its negation $\bar{b}$) is proved to be impossible in the proof of Lemma 2.3.
- The case $K_1 = K_4$ and $K_2 = K_3$ (the gate that computes $a \oplus b$ or $\overline{a \oplus b}$) is proved to be impossible in the proof of Lemma 2.3.

This ends the proofs of Lemma 2.5. □

Two particular cases of this lemma clearly reduce the possibilities of a malicious generator. First, an adversary cannot force the output of a non-linear gate, and thus cannot trivially force the output of the entire garbled circuit. Moreover, the adversary cannot alter a gate so that it always outputs the first input a ($K_1 = K_2$ and $K_3 = K_4$). This last example is interesting: it actually means that the malicious generator cannot modify the circuit so that the evaluator's inputs go directly to the output through the circuit.

2.4.5 About Other Non-Linear Gates

We showed in Section 2.3 how to turn a gate that computes $a \wedge b$ into $\bar{a} \wedge b$, $a \wedge \bar{b}$ and $\overline{a \wedge b}$. It appears that these alterations and their combinations are identical to the honest ways of garbling these respective gates, described in [ZRE15].

Then, an honest garbling of $\overline{a \wedge b}$ (or any other non-linear gate) can be obtained from a corruption of $a \wedge b$. Thus, there is no modification that can be made on $\overline{a \wedge b}$ and that cannot be made on $a \wedge b$. Therefore, any non-linear gate can only be turned into another non-linear gate.

2.4.6 Fitting Everything Together

Assembling the lemmata previously proved, we obtain Theorem 2.1, which is the main contribution of this chapter.

Theorem 2.1. *If all the operands of the first non-linear garbled gates can take the two values according to the evaluator's inputs (while the generator's inputs are fixed), and if there are output commitments, then the adversarial generator is limited to turn any non-linear gates into other non-linear gates.*

This theorem means that if we can guarantee that the first garbled gates (the non-linear gates that are the closest to the input wires) can take the two possible inputs, independently on each wire, according to the evaluator's choice, then all the garbled gates can only be altered into any non-linear gates.

Proof. Using Lemma 2.1, if the input wires of the first garbled gates all have two possible garbled keys, then there is no wire in the rest of the circuit that has only one possible key. Combining Lemmata 2.2 and 2.4, if the input wires of the first garbled gates of the circuit all have the same odd global offset and if the circuit has output commitments, then no wire of the rest of the circuit has more than two possible garbled keys. Moreover, with the same conditions, Lemma 2.3 shows that all wires share the same odd global offset. Then, Lemma 2.5 comes last and shows that non-linear gates can only be turned into other non-linear gates, and that this is the only possible corruption. □

It remains to study the conditions so that the starting point of this theorem is satisfied: all the inputs of the first non-linear gates have two possible garbled keys. How to guarantee some wires to have two possible garbled keys, with the same global odd offset? We will show below that it is possible to make sure that all the evaluator's inputs are converted into garbled keys with a common global odd offset. But there is no way to do the same for the generator's inputs. Indeed, the adversarial generator cannot be forced to choose his inputs after generating the garbled circuit. On the other hand, XOR gates cannot be corrupted, and so a XOR gate with an evaluator's input will necessarily have two distinct outputs. Hence, here are some interesting cases that will meet our above requirements:

- one wants to evaluate $f(y)$, for a public function f, so that the evaluator chooses y, but the generator will get the result;
- one evaluates $f(x, y)$, and any input wires of the first non-linear gates is either a y_j chosen by the evaluator, or $x_i \oplus y_j$, where x_i is chosen by the generator. Indeed, in both cases, y_j or $x_i \oplus y_j$, when x_i is fixed, the inputs of the first gates can take the two possible values according to y_j.

The latter case applies to a large class of circuits, including the addition, the greater-than (as defined in [KSS09]), the equality test, combination of those, or even more complex circuits, such as AES.

The former case is known as *privacy-free* garbled circuits. As described in Section 2.1.7, there are more efficient garbling schemes in this context. The work of [ZRE15] provides an optimal solution for this purpose. Our results also hold with this garbling scheme, but only the general solution is presented here.

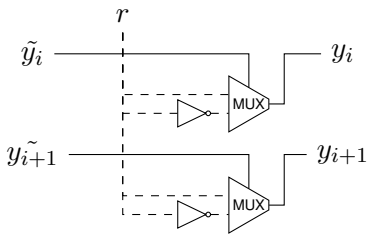


Figure 2.12: Overview of the sub-circuit

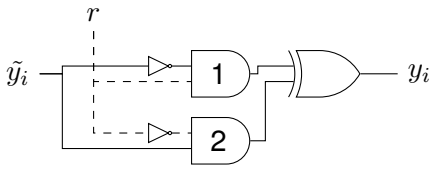


Figure 2.13: Implementation of the MUX

2.4.7 Ensuring the Correct Garbling of Input Wires

In this section, we describe a construction to guarantee that input wires of the evaluator E are correctly garbled by the generator G (i.e. all input wires of E share the same odd offset). Whether this can be solved using a modified version of correlated-OT (see Section 1.3.3) is an open question that deserves some attention. Instead, we propose to modify the circuit representation of the function to evaluate, by adding a sub-circuit in front of the original circuit[4]. This sub-circuit is illustrated in Fig. 2.12. Fig. 2.13 gives details of the multiplexer, but is not required for the correctness. As usual, we call x the input of G, and y the input of E.

Construction

The main idea is that rather than transmitting the input garbled keys of E through an oblivious transfer, the inputs are now connected to the outputs of this sub-circuit. The sub-circuit has the same number of inputs of E as the original circuit plus one: a bit r that is randomly chosen by E. For each input y_i of the original circuit, the sub-circuit has an input $\tilde{y_i} = y_i \oplus r$ and an output y_i. The new inputs are transmitted as usual through an oblivious transfer.

We also give restrictions on some permute bits: the permute bit of w_R (the wire carrying r) and $w_{\tilde{Y_i}}$ (carrying $\tilde{y_i}$) must be zero. Also the permute bit of w_{Y_i} (the wire carrying y_i) must be public. This is to ensure that G does not force the inputs of E during the oblivious transfer phase.

Because of r and of those permute bits, the protocol has to be slightly modified, as suggested by the following sketch:

1. G garbles the concatenation of the two circuits using the usual garbling scheme and sends it to E, along with his garbled input keys for x and the permute bit for w_{Y_i}, for all i;

2. E randomly picks a bit r;

[4]We do not modify the garbling scheme itself.

3. E and G perform oblivious transfers in order E to obtain the garbled keys of $\tilde{y_i}$ and r, and E checks that the select bits of these keys match the clear values or aborts. This ensures two possible keys for the evaluator's inputs;
4. E evaluates the sub-circuit and checks if the select bits of the keys for the input y match the clear value, or aborts;
5. E evaluates the rest of the circuit and returns the result.

Note that the listed steps can be grouped so that only four communications are needed. Since the functionality of the circuit is not changed by the sub-circuit (as long as the new input $\tilde{y}$ is chosen according to r), the correctness is preserved.

Analysis

Our security goal is to ensure that all output wires of the sub-circuit (i.e. inputs of the rest of the circuit) share the same odd global offset, or the protocol aborts for some specific inputs. To prove it, we need two more lemmata.

Lemma 2.6. *For any garbled gate, if the two operands have distinct but odd offsets, then the offset of the first operand is propagated to the output wire.*

Proof. The proof of this lemma is identical to the proof of Lemma 2.3. Indeed, in the proof of Lemma 2.3, the offset of the second operand ($k^0_{B\gamma} \oplus k^1_{B\gamma}$) never appears. □

Lemma 2.7. *For any XOR gate, if the offsets of the operands are different or if one of the operands has more than two garbled keys, there are at least four distinct garbled keys at the output.*

Proof. The proof of this lemma is trivial since the output keys of a XOR gate are the input keys XORed together. □

Let us analyze the propagation of offsets in one of the multiplexers of the sub-circuit. Remark that there cannot be only one possible garbled key for w_{Y_i}. Indeed, since the permute bit of this wire is known by the evaluator, then there must be at least two possible keys with opposite select bits. We consider the multiplexer illustrated in Fig. 2.13. We stress that the order of the operands matters. Let w_1 and w_2 refer to the output wires of the AND gates noted respectively 1 and 2. We also note Δ the offset of wire w_R carrying r and $\Delta_{\tilde{Y_i}}$ the offset of the wire carrying $\tilde{y_i}$. We can enumerate the different corruption cases:

1. The offsets Δ and $\Delta_{\tilde{Y_i}}$ are different but odd.
2. Δ is even and $\Delta_{\tilde{Y_i}}$ is odd.

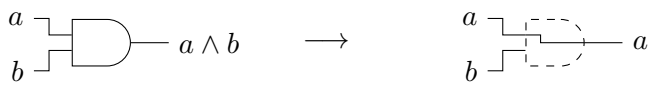


Figure 2.14: Impossible corruption

3. Δ is odd and $\Delta_{\tilde{Y}_i}$ is even.

4. Both offsets are even (distinct or not).

Consider the first case. According to Lemma 2.6, the different offsets propagate so that w_1 has offset $\Delta_{\tilde{Y}_i}$ and w_2 has offset Δ, or one of them two wires have more than two keys. In either case, using Lemma 2.7, the output of the XOR gate gives at least three different keys. Given that these three (or more) keys engages the value of Δ, we can show that it cannot be reduced back to two in the rest of the circuit, using the same method as for Lemma 2.4.

Consider the second case, if Δ is even, then the garbled keys of w_R have equal select bits. In other words, the select bit of one of the garbled keys does not match the clear value of r. Since r is known to evaluator and since the permute bit must be set to zero, this situation is detected and leads the evaluator to abort. The exact same reasoning works for the third and fourth cases.

We can now conclude that the output wires of the sub-circuit have exactly two possible garbled keys with the same odd global offset, or the protocol aborts for some inputs of the evaluator or some r.

2.5 Applications to Real Circuits

In the previous sections, we have defined precisely how a malicious generator can corrupt a garbled circuit. Turning non-linear gates into other non-linear gates is equivalent at adding NOT gates to the circuit. Then, we consider in this section that the adversary is able to add a NOT gate to any wire of the circuit. An important consequence is that a circuit cannot be modified so that the evaluator's inputs go through the gates to the outputs of the circuit. More precisely, the corruption of a gate as shown in Fig. 2.14 cannot be generated. Thus, the question "does a corrupted circuit leak more information than the original circuit?" turns out to be trickier than suggested in the previous works.

In this section, we don't provide a general answer, but we see the impact of corruptions on some real circuits. We measure this impact with the Shannon entropy of the evaluator's input. We call x and y the respective inputs of the generator and the evaluator. Let $z = f(x, y)$ be the function to evaluate and C_f a boolean circuit computing it. We note $\mathcal{C}_f$ the set of all circuits that can be obtained by corrupting C_f (i.e. by adding NOT gates to C_f). In other words, there

Algorithm 2.1: Finding the best corruption of a circuit

Input: A circuit C_f of N wires w_1 to w_N (arbitrary order).
Output: The corrupted circuit that leaks the most information.
Set an N-bits integer ω to zero.
Compute the initial entropy $H = H(Y|X = x, f(x, y))$.
while $\omega < 2^N$ **do**
  $\omega \leftarrow \omega + 1$
  $\mathsf{C}_{f'} \leftarrow \mathsf{C}_f$
  foreach ω_i **do**
    If $\omega_i = 1$, add a NOT gate to the wire w_i of $\mathsf{C}_{f'}$.
  end
  Compute the truth table of $\mathsf{C}_{f'}$.
  Compute the entropy $H' = H(Y|X = x', f'(x', y))$.
  if $H' < H$ **then**
    $H \leftarrow H'$
    $\mathsf{C}_{f'}$ becomes the best corrupted circuit so far.
  end
  return The best corrupted circuit found.
end

exists a corruption of C_f that leads to $\mathsf{C}_{f'}$, that computes some other function f', if and only if $\mathsf{C}_{f'} \in \mathcal{C}_f$. We formalize the problem as follows :

Problem 2.1. *For a circuit C_f, does it exist a corrupted circuit $\mathsf{C}_{f'} \in \mathcal{C}_f$, such that the obtained function f' leaks more information on the evaluator's input :*

$$H(Y|X = x, Z = f(x, y)) > H(Y|X = x', Z = f'(x', y)) \ ?$$

Remark that in the entropy equation, the generator knows x since this is his input. In our computations, we consider that the adversarial generator chooses his input in order to increase the leakage: i.e. he picks x that minimizes $H(Y|X = x, Z = f(x, y))$. Similarly, for a corrupted circuit, he also chooses x'.

To help us answer that question, we implemented a tool to exhaustively compute all corruptions $\mathsf{C}_{f'}$ of a circuit C_f and check if one of them leaks more information. More details about this tools are given in Algorithm 2.1.

2.5.1 The Greater-Than Function

Let us now see a practical example: the greater-than function, that returns a single bit (1 if $x > y$, 0 otherwise). Assuming the adversary takes the middle of the set as input (which leaks the most information), the original function leaks one bit of entropy. Since there is a single

output wire, whatever the modification made on the circuit, it does not leak more than one bit of entropy on y. But it is interesting to see that the adversary is limited in the choice of that bit. For example, if we consider the greater-than circuit defined in [KSS09], it cannot be modified to output the parity bit of y. This can be proven exhaustively for the 3-bit greater-than circuit and then recursively.

In the particular case of greater-than circuit, remark that the best strategy of an adversarial generator G, willing to retrieve the input y, consists in not modifying the circuit. If y is ℓ-bit long, then it would require ℓ evaluations for G to find y, and it cannot be reduced by corrupting it. Thus, in this context, using cut-&-choose based solutions does not enhance privacy (but ensures the correctness).

2.5.2 The Addition Function

Let us study now the addition function f, the circuit C_f of which is defined and optimized in [KSS09]. Consider that E has two inputs $y, y' \in \mathbb{F}_2^\ell$ and the generator none. This circuit computes the addition of y and y' in $\mathbb{F}_2^\ell$ (the carry bit is not returned). The original function f does not leak any information on y (or on y'). Up to $\ell = 10$, we exhaustively demonstrated that no modification leaks any information on y:

$$H(Y|Z = f'(y, y')) = H(Y|Z = f(y, y')) = \ell$$

Since the construction of [KSS09] uses serial 1-bit adders, this result can be extended recursively for larger values of ℓ.

2.5.3 The Equality-Test Function

Unfortunately, it is not the case for all circuits. Consider now the equality-test function that returns 1 if and only if $x = y$. The Boolean circuit we study for the 4-bit case is shown in Fig. 2.15. Inputs are 4-bit long and after the evaluation of the original function, it remains 3.66 bits of entropy.

This circuit is vulnerable to the addition of NOT gates. Indeed, we demonstrated exhaustively that the best corruption requires to add a single NOT gate, as shown in red in Fig. 2.15. Now, the remaining entropy is $H(Y|X = x', Z = f'(x', y)) = 3.01$ bits. Consequently, almost 1 bit is leaked by this function f'. Actually, f' returns $x_3 \oplus y_3$ if x_{0-2} and y_{0-2} are different (which happens with probability 7/8) and 0 otherwise. Clearly, this same attack would work (even with higher probability) for larger equality-test circuits.

But note that this attack is entirely based on the topological representation of the function. If we inverted the direction of the cascade of AND gates, as shown in Fig. 2.16, the leaked bit

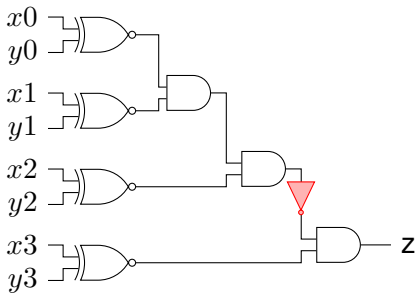


Figure 2.15: Circuit for the 4-bit-equality test and its best corrupted circuit in red

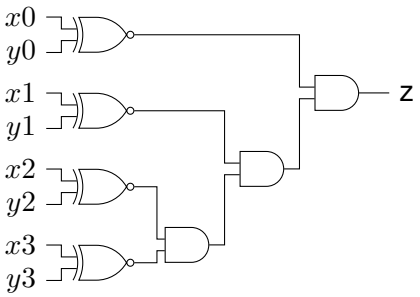


Figure 2.16: Another circuit for the 4-bit-equality test

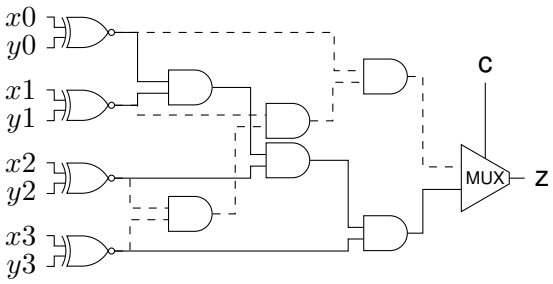


Figure 2.17: Improved circuit for the 4-bit-equality test

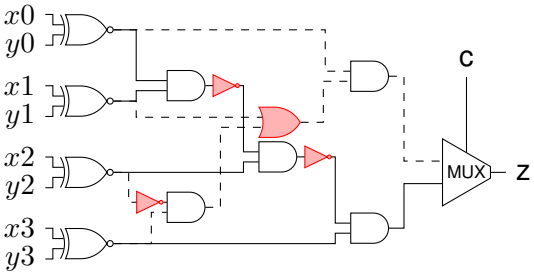


Figure 2.18: Best corrupted circuit for the improved 4-bit-equality test

would be $x_0 \oplus y_0$.

Since the leakage is dependent on the topology, we started investigating whether there exists a circuit (computing the same functionality) that has a reduced leakage in case of corruption. We discovered a generic construction of such circuits for any function f. Unfortunately, this fix also requires to increase the size of the circuit. In order to reduce the leakage, we actually take advantage of the fact that it depends on the topology of the circuit. In the case of the equality test (w.l.o.g.), we propose to evaluate two parallel sub-circuits with different topologies (for example the topologies of Fig. 2.15 and Fig. 2.16), and to output only one, randomly chosen by the evaluator E. This approach is illustrated in Fig. 2.17. A dashed sub-circuit performing differently the equality test is added to the previous one, and a multiplexer (described in [KS08]) allows E to choose which of the two results is returned (let c represent this choice). If the generator G is honest, the circuit remains correct : the sub-circuits have the same result and the multiplexer has no influence on the correctness. Otherwise, G does not know which one of the two sub-circuits has returned the result.

However, the attacker can still add NOT gates to this new circuit. Using the same method, we computed that the best corruption requires six NOT gates to be added, as illustrated in

Fig. 2.18. By studying the multiplexer of [KS08], we can show that there is no way for the adversary to force the choice of c. Then, we do not detail the multiplexer in the figures. The remaining entropy on the evaluator's input after evaluating this corrupted circuit is $H(Y|X = x', Z = f'(x', y)) = 3.35$ bits. Then, we have considerably reduced the leakage of information. Of course, one can reduce it even more by parallelizing more sub-circuits, but the size of the global circuit would drastically increase.

This work opens the problem of finding circuits that resist to the addition of NOT gates.

2.5.4 Trade-Off with Cut-&-Choose

We showed that, for some classes of circuits, there exists corrupted circuits that leak more information than the original function. In such cases, a cut-&-choose solution remains necessary if we want to avoid this leakage. Based on the fact that this leakage depends on the topology of the circuit, our results still allow to improve for free any cut-&-choose based solutions since [Lin13].

Since several garbled circuits are generated, we recommend to use different circuits of the same function (with different topologies). Then, even if the adversary manages to guess correctly which circuits are opened and which are evaluated, he is limited to corruptions that can be obtained from all unopened circuits and their respective topologies. Indeed, if different corrupted circuits do not compute the same (corrupted) function, then they may output different results, which allows the evaluator to learn the adversarial inputs thanks to [Lin13, AMPR14].

For example, let us consider the two circuits of Fig. 2.15 and 2.16 of the same function. Say that a cut-&-choose solution is used with half of the circuits with the first topology and the other half with the second. Assume that at least one circuit of each is unopened. Then, we demonstrated exhaustively that any corrupted function that can be obtained from both topologies does not leak any information on the evaluator's inputs more than the original function already does.

2.5.5 Garbled Circuits with Covert Adversaries

We showed that there exists circuits for which the addition of NOT does not advantage a malicious generator. However, such an adversary is still able to make selective failure attacks if he accepts the risk of getting caught. In this section, we design mechanisms to prevent from selective failure attacks. Unlike cut-&-choose, our only protects against selective failure attacks, but we believe it can lead to more efficient solutions.

In this part, we focus on a covert adversary with $1/2$-deterrent (as defined in Section 1.2) but the proposed solution could be adapted to any deterrence factor. Note that a deterrent factor of $1/2$ implies that an adversary is willing to cheat only if his probability of success is strictly

higher than his probability of getting caught. We believe this setting applies to many real world contexts.

Based on the observation that selective failure attack is very similar to the probing model in side channel analysis, the proposed scheme is heavily inspired by 1-order masking schemes. However, one cannot simply apply a masking scheme to garbled circuits. Indeed, one of the main differences is the ability of the adversary to modify a circuit before executing it.

Adapting a 1-Order Masking Scheme

In order to simplify notations, the AND operation $a \wedge b$ will just be noted ab for the rest of the chapter.

The purpose of this section is to show how to modify a circuit, so that any sensitive value is split in at least two wires, independently of the modifications (i.e. addition of NOT gates) possibly made on the circuit by an adversarial generator.

We note x and y the respective l-bit inputs (w.l.o.g. they have the same length) of the generator and the evaluator. They want to compute privately $f(x, y)$ and agreed on some circuit C_f computing this function. Let $d = \text{depth}(\mathsf{C}_f)$ be the *non-linear depth* of this circuit, defined as follows:

Definition 2.4 (non-linear depth)**.** *The non-linear depth of a circuit is the number of non-linear gates of the longest path from the inputs to the outputs of that circuit.*

This definition is somewhat similar to the usual depth of a circuit, with the particularity that this circuit would not have any linear gate. We also define a *layer* of a circuit as follows:

Definition 2.5 (layer of a circuit)**.** *The layer i of a circuit is the set of all non-linear gates, to which the longest path from the inputs crosses i non-linear gates.*

The evaluator E randomly chooses a d-bit mask r. We note r_i the i^{th} bit of r. The main idea of our scheme is to mask every wire of layer i with r_i and to replace every AND gate (or similarly any other non-linear gate) by the following sub-circuit, also shown in Fig. 2.19:

$$ab \oplus r_{i+1} = (a \oplus r_i)(b \oplus r_i) \oplus r_i(b \oplus r_i) \oplus r_i(a \oplus r_i) \oplus r_{i+1} \oplus r_i$$

Similarly, a XOR gate is replaced by the following free sub-circuit:

$$a \oplus b \oplus r_{i+1} = (a \oplus r_i) \oplus r_{i+1} \oplus (b \oplus r_i)$$

Note that if a wire carrying some bit $c \oplus r_i$ is needed in layer $j > i$, one can easily update the mask with the following free sub-circuit :

$$c \oplus r_j = (c \oplus r_i) \oplus r_j \oplus r_i$$

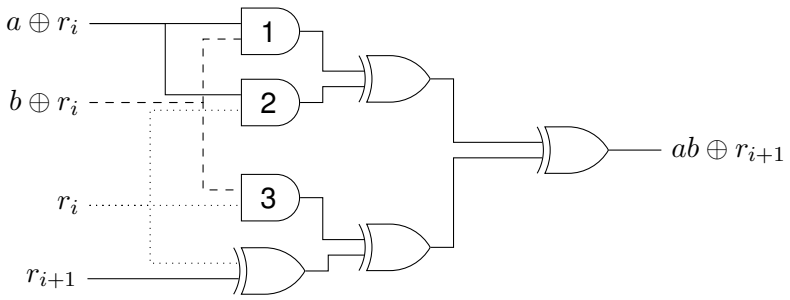


Figure 2.19: 1-order masked AND gate for garbled circuits

It is interesting to see that at no point a value stays unmasked. However, unlike in side-channel analysis, G is able to modify some gates of the circuit. Replacing one AND gate by three AND gates allows G to add NOT gates to any of them. But thanks to the new mask r_{i+1} and the impossibility of corrupting a XOR gate, there is no way to obtain an unmasked result. Thus, for the rest of this part, we consider that the only possible modifications of the previous global AND gate are:

$$ab \oplus \alpha_0 a \oplus \alpha_1 b \oplus \alpha_2 \oplus r_{i+1}$$

with the bits α_0, α_1 and α_2 of the generator's choice. Written differently, this corresponds to $\overline{a}b \oplus r_{i+1}$, $a\overline{b} \oplus r_{i+1}$, etc.. Thus, the modifications made on a global AND gate are no more than what we studied on a regular AND gate.

In terms of efficiency, we have replaced a single AND gate by three new AND gates. Then, one could expect the communication cost to be multiplied by 3. However, it can be reduced to a multiplicative overhead of 2 or even less by using the privacy free garbling scheme studied in Section 2.1.7. Indeed, r_i is known to E and we can take advantage of this knowledge to garble $r_i(a \oplus r_i)$ and $r_i(b \oplus r_i)$ (gates 2 and 3 of Fig. 2.19) with a single ciphertext. Then, the multiplicative overhead over the semi-honest settings is only two: the garbling of a global AND gate requires four ciphertexts.

Moreover, remark that these gates (2 and 3) can also be redundant with other gates of the circuit. For example, if another global AND gate computes $ac \oplus r_{i+1}$ in the same layer i, the gate labelled 2 is already defined. In that case, only seven ciphertexts are required to garble two global AND gates (instead of four in the semi-honest settings). Similarly, if another global AND gates computes $bc \oplus r_{i+1}$, then gates 2 and 3 are redundant and only nine ciphertexts are needed for three global AND, which represent a 1.5 multiplicative overhead, instead of 2 in the corresponding cut-&-choose solution.

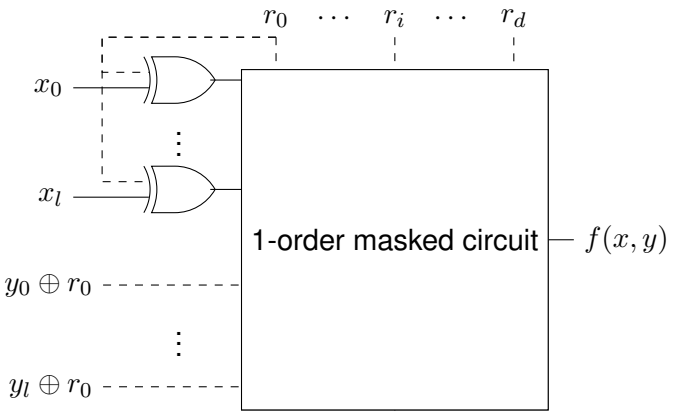


Figure 2.20: Inputs of an 1-order masked circuit

Garbling and Exchange of the Inputs

Before the inputs are exchanged, E randomly chooses a mask r of d bits. We note w_{X_j} the wire carrying x_j, w_{R_i} the wire carrying r_i, $w_{X_j \oplus R_i}$ and so on. We stress that there is no wire carrying y_j unmasked.

Then, the inputs wire of the masked garbled circuits are w_{X_j}, $w_{Y_j \oplus R_0}$ (for all $0 \leq j \leq l$) and w_{R_i} (for all $0 \leq i \leq d$). We stress that w_{X_j} is connected to a single gate: a XOR gate that outputs $w_{X_j \oplus R_0}$.

We recommend the permute bit of w_{R_i} and $w_{Y_i \oplus R_0}$ to be set to zero. In other words, for any of these wires, the least significant bit (select bit) of a garbled key is the clear value itself. E gets the garbled keys for these wires through a 1-out-of-2 oblivious transfer. He can then check that the select bits of the received keys match the input and the chosen mask. If not, the adversarial G was trying to make a selective failure attack, or an offset was even, and E can safely abort. This ensures that the offsets are odd for these wires, or the adversary is caught with probability one half. We remind that odd input offsets is a necessary condition to the global circuit security.

Still, it remains to ensure that an odd offset is used for w_{X_j}, the input of the generator. Obviously, the permute bit of this wire must be chosen randomly by G, in order to preserve his privacy. Because of this, the same trick cannot be applied. However, this wire is used in a single gate: a XOR gate that takes as input w_{X_i} and w_{R_0}. We just showed that an odd offset is used for w_{R_0} and we proved in Section 2.4 that the input wires of a XOR gate must have the same offset.

The overall circuit is shown in Fig. 2.20. In dashed are the wires the permute bit of which is set to zero (i.e. the least significant bit of a garbled key is the clear value itself).

Comparison with Cut-&-Choose

With the best general case cut-&-choose solution, two circuits would be necessary to be secure in the proposed setting. One of them would be opened and the other would be executed.

In comparison, our scheme requires a single circuit with three times more non-free gates than in the semi-honest settings. However, we showed that the garbling only requires twice the number of ciphertexts to transmit, compared to the semi-honest settings. Then, the communication cost is similar to the best cut-&-choose solution. Even better, when some gates share the same input, we showed that one less ciphertext has to be sent, thus reducing the multiplicative overhead below two.

On the other hand, our scheme requires a few more oblivious transfers, since the masking bits are randomly chosen by the evaluator. We remind that d masks are necessary for a circuit of depth d. We believe these are negligible with the size of the circuit and the number of oblivious transfers used for the inputs.

We remind that a covert adversary is still able to add NOT gates. This scheme only protects against selective failure attacks. However, it shows that more specific solutions than the traditional overkill cut-&-choose can be designed. Hopefully, optimizations of our scheme allowing to reduce the number of non-free gates would make our scheme much more competitive.

2.6 Conclusion

In the beginning of this chapter, a detailed introduction to garbled circuits, its major optimizations and cut-&-choose was made. After three decades of works on this very competitive research area, I believe it was necessary and I hope it will bring new people interested into the subject.

The main contribution of this chapter is to define precisely what alterations of a garbled circuit a malicious generator can make. We have proved that for a large class of circuits, the adversary is limited to turn non-linear gates into other non-linear gates and to make selective failure attack. This is equivalent to say that he can only add NOT gates to the wires of his choice, or to probe some wires with some probability of getting caught. This is drastically lower than the previous state-of-the-art suggests. We believe this work can lead to some more optimized secure solutions in the malicious setting, more efficient than the regular cut-&-choose schemes.

For circuits outside the class we define, what corruptions an adversary is able to make is still an open question. Our preliminary studies suggest that this question is highly non-trivial and may depend on the topology of the circuit being corrupted.

The second contribution is the analysis of the impact of NOT gates in real-life circuits. We show that some circuits do not leak more information when NOT gates are added, and thus cut-&-choose solutions are unnecessary to enhance the privacy security property. However, for

some other circuits, the addition of NOT gates can lead them to reveal more information, but in that case we give recommendations to improve cut-&-choose solutions for free.

For circuits that are resistant to the addition of NOT gates, we design an alternative to cut-&-choose to prevent selective failure attacks. Although it has roughly the same overcost than cut-&-choose based solutions, we believe this direction can lead to more efficient solutions.

Finally, our contribution also opens an interesting problem: can we define an OT protocol, such that the sender has n pairs of messages $(m_{j,0}, m_{j,1})$ with $m_{j,1} = m_{j,0} \oplus \Delta$ for all $1 \leq j \leq n$ (or more generally $m_{j,1} = f(m_{j,0})$ for an arbitrary function f)? The sender has n choices b_j and wishes to obtain m_{j,b_j} for all j, with the guarantee that $m_{j,1} = m_{j,0} \oplus \Delta$. Of course this can be achieved by adding commitments and zero-knowledge proofs, but it would be interesting to study if it can be solved with no overcost compared to the OT-extension protocol [IKNP03]. From a theoretical point of view, this could be achieved even more efficiently since all pairs of messages share the same relation.

CHAPTER 3

LOCATION PROOF SYSTEM BASED ON MULTI-PARTY COMPUTATIONS

In this chapter, we show how multi-party computations can help users to protect their privacy in everyday life. More specifically, we study the case of location-based services that have become quite popular (e.g. GPS, location-based advertising, augmented reality games). Their variety and their numerous users show it clearly. However, these applications rely on the people's honesty to use their real location. If they are motivated to lie about their position, they can easily do so. A *location-proof system* allows a *prover* to obtain proofs from nearby *witnesses*, for being at a given location at a given time. Such a proof can be used to convince a *verifier* later on. However, provers and witnesses may not want to broadcast their identity or their position each time they generate location proofs.

Many solutions have been designed in the last decade, but none protects perfectly the privacy of their participants. In this chapter, a solution is presented in which a malicious adversary, acting as a prover, cannot cheat on his position. It relies on multi-party computations and *group-signature schemes* to protect the private information of both the prover and the witnesses against any semi-honest participant.

Additionally, this chapter gives a new secure multi-party maximum computation protocol for the specific context of location-proof systems. This tool allows n users to know which one of them has the greatest value without revealing their values. It requires $O(n\log(n))$ computations and communications, which greatly improves the previously known solutions having $O(n^2)$ complexities, but at the cost of some small leakage that we analyze. Although it is designed for our location-proof system, it can be applied to any scenario in which a small information leakage is acceptable.

Contents

3.1 Introduction

Location-based services are now ubiquitous, mostly through our phones and vehicles. These services generally rely on the people's honesty to use their real location. Hence, they are limited to situations in which the people do not have any motivation to lie. However, for some services such as electronic voting, location-based access control, and law enforcement investigation, this is not the case. These services must be based on a *location-proof system* that allows a participant, called *prover*, to obtain proofs from nearby participants, called *witnesses*, asserting that he has been at a given location at a given time. Such a proof can be used later on to convince a service provider, called *verifier*.

Any location-proof system based on the interaction between a prover and his neighbours has some privacy issues. The prover may not want to broadcast his identity every time he needs location proofs. Similarly, witnesses may want to hide their identity and location. Hence, private information must be kept secret from all the participants but not from an independent trusted third party, called *judge*. Indeed, the judge must be allowed to retrieve the identities of the participants, in order to detect malicious collusions among them. In this chapter, we consider that an ideal location-proof system for such applications must then have the following properties [GKRT14].

1. *Correctness*: location proofs generated honestly by a prover with the collaboration of honest witnesses must always be accepted by the verifier.
2. *Unforgeability*: a prover cannot obtain/modify valid location proofs for a location where he is not, or at a different time.
3. *Non-transferability*: location proofs are valid only for the prover who generated them. They cannot be exchanged.
4. *Traceability*[1]: given a proof, the judge must be able to retrieve the identity of the witness who signed it.
5. *Location and identity privacy*: the location and the identity of the witnesses and the prover must be kept secret from other participants (except the judge).
6. *Unlinkability*: given two distinct location proofs, a participant cannot guess whether they have been generated by the same witness, nor whether they concern the same prover. This obviously does not stand for the judge.
7. *Storage sovereignty*: the prover is responsible for storing his own location proofs. No one is able to access them without the prover's agreement.

[1] The traceability property is new, it does not come from [GKRT14].

In this chapter, we propose the first privacy-aware location-proof system that fulfils all these properties. It relies on two protocols: a *location-proof gathering* protocol (allowing a prover to obtain proofs from witnesses) and a *location-proof verifying* protocol (allowing a verifier to validate the correctness of a proof). The first one ensures that both the prover and the witnesses keep their identity and their location secret. Once the location proofs have been obtained from witnesses, a prover must keep them securely and may use them later on to convince verifiers. For efficiency reasons, no centralized server is used during the gathering protocol.

The security of our solution is analyzed against *malicious* and *semi-honest* adversaries. The former is a prover trying to obtain invalid location proofs, whereas the latter is any participant (prover, witness or verifier) trying to obtain the private information on other participants. *Static collusions* between a prover and some of the witnesses against other witnesses are also considered.

3.2 Preliminaries

3.2.1 Group Signature Schemes

A *group signature scheme* is a technique introduced by Chaum and Van Heyst [Cv91] for allowing a member of a group to anonymously sign a message. A verifier is then able to check the validity of a signature but cannot determine which group member generated it. Every group member is given a unique secret key from a *group manager*, that possesses the master secret key. Therefore, the group manager is responsible for adding members to the group and revoking signature anonymity. Many solutions have been designed and they all meet the following requirements:

- *Soundness and completeness*: valid signatures by group members always verify correctly, and invalid signatures always fail verification.
- *Unforgeability*: only group members can generate valid group signatures.
- *Traceability*: given a valid signature, the group manager must be able to lift the anonymity of the signer.
- *Anonymity*: given a message and its signature, none should be able to determine the identity of the signer (except the group manager).
- *Unlinkability*: given two messages and their signatures, none should be able to determine whether they have been generated by the same group member.

Depending on the solutions, additional properties can be added. Particularly, the group manager is often divided in two entities: a membership manager and a revocation manager, this

allows to separate the two responsibilities. We do not consider this separation in the rest of the chapter.

More recently, Franklin and Zhang [FZ12] introduced a new property that breaks the unlinkability property in the very particular case where the two messages are the same:

- *Uniqueness*: given a single message and two signatures of this message, one can tell whether they have been generated by the same group member.

Such solutions are referred as *unique group signature schemes*.

3.2.2 Prior Location-Proof Systems

Several solutions that partially fulfil our objectives were proposed. Unfortunately, most of them require that the participants broadcast their identity and/or location. Sastry et al. [SSW03] introduced the notion of *secure location verification*. Their solution relies on the deployment of impersonal local access points to locate participants in a given region, using *distance-bounding protocols*. Furthermore, the identity and location of the prover have to be transmitted to allow access points to grant access to nearby location-based services. In [SW09], Saroiu et al. introduced the notion of location proofs. The prover can now ask access points to generate proofs that he can store until he has to convince a verifier. However, it still requires an infrastructure to be deployed and does not ensure privacy. Later, other approaches based on impersonal access points (Luo et al. [LH10a, LH10b] and Pham et al. [PHB+15]) start answering the privacy issues using hash functions and pseudonyms. Although the most recent of these schemes achieve a high level of privacy, it is still limited to regions where access points are already deployed.

A complete different approach has been used by Singelee et al. [SP05]. Instead of deploying impersonal devices, they have suggested to involve nearby users. These users, called witnesses, can run distance-bounding protocols with the prover to certify his location. Unfortunately, the scheme still does not provide any privacy property. The solution of Graham et al. [GG09] is somehow similar, but the verifier has to choose himself the witnesses among the nearby volunteers. It reduces the probability of collusion among the participants. Later, Zhu et al. proposed a new solution APPLAUS [ZC11] that protects identities through a set of pseudonyms. This allows the witnesses to generate location proofs without leaking their identity. However, all proofs (including pseudonyms and locations) are stored in a centralized authority, raising some privacy and efficiency issues. The protocol Link of Talasila et al. [TCB10] is also based on centralized system.

Finally, Gambs et al. [GKRT14] proposed a solution to get rid of the central authority and to ensure most privacy properties. Identities are protected with a group-signature scheme instead of pseudonyms and the positions of the witnesses are not transmitted. Unfortunately, the

	Echo [SSW03]	[SP05]	SLVPGP [GG09]	[SW09]	[LH10a]	Veriplace [LH10b]	Applaus [ZC11]	Link [TCB10]	[DCF12]	Props [GKRT14]	SecureRun[PHB+15]	Our work
Prover anonymity P: pseudonyms G: group signatures H: hash function					H				H	G	P	G
Witness anonymity P: pseudonyms G: group signatures NA: not applicable	NA			NA	NA	NA	P		H	G	NA	G
Prover location privacy	✓			✓	✓	✓		∼			✓	✓
Witness location privacy	NA			NA	NA	NA			✓	✓	NA	✓
Storage sovereignty					✓					✓	✓	✓
No infrastructure requirement		✓	✓				✓	✓	✓	✓		✓
Traceability	NA		✓	NA	NA	NA	✓	✓	✓		✓	✓

Table 3.1: Comparison of existing protocols

location of the prover is still learned by the witnesses. A comparison of all these schemes is provided in Tab. 3.1.

In comparison, our solution relies on multi-party computations and group signature schemes to protect the identity and the location of all participants. It assumes that the participants have phones/vehicles with directional antenna to locate their neighbours. Such a solution can complement classical distance-bounding protocols [BGG+16].

3.2.3 Secure Two-Party Comparison Protocol

Since the millionaires' problem was introduced by Yao [Yao82], it has attracted a lot of interest. Although the original solution of Yao solves the problem, its efficiency becomes prohibitive for large values (see Section 1.1). Generic solutions like garbled circuits or homomorphic secret sharing could be used to answer the problem, but it appears that some custom protocols are more efficient ([IG03, BK04, LT05] ...).

In this section, the technique of Lin and Tzeng [LT05] is detailed. We choose this one among others for efficiency reasons and because it is highly customizable.

Their solution uses either a multiplicative homomorphic encryption scheme or an additive

Protocol 3.1: Secure two-party comparison protocol of Lin and Tzeng [LT05].

Input: The l-bit private values a and b of A and B.
The encryption function $E_A(\cdot)$ with key N_A for an additive encryption scheme.
A hash function $h(\cdot)$.
Output: A determines whether $a > b$ or $a \leq b$.
Step 1: A does the following computations:
- Compute T_1^a.
- Create the l-element vector γ, so that $\gamma_i = h(T_1^a[i])$ if it exists, otherwise γ_i is a random value.
- Return $(E_A(\gamma_1), \cdots, E_A(\gamma_l))$ to B.

Step 2: B does the following steps:
- Compute T_0^b and pick a random permutation $\pi_B(\cdot)$.
- If $T_0^b[i]$ exists, homomorphically compute the l-element vector δ:

$$\begin{aligned} E_A(\delta_i) &= E_A(k_i \cdot (h(T_1^a[i]) - h(T_0^b[i]))) \\ &= (E_A(\gamma_i) \cdot E_A(-h(T_0^b[i])))^{k_i} \end{aligned}$$

  where $k_i \in_R \mathbb{Z}_{N_A}$.
- Otherwise, δ_i is a random non-zero value.
- Send $\pi_B(E_A(\delta_1), \cdots, E_A(\delta_l))$ to A.

Step 3: A does the following steps :
- Decrypt the shuffled vector δ.
- The vector δ contains 0 if and only if $a > b$.

homomorphic encryption scheme. We present the latter one in Protocol 3.1, since we will need the additive property in our scheme.

Given an integer x, let us define the following sets: $T_0^x = \{x_1x_2...x_{i-1}1 | x_i = 0\}$ and $T_1^x = \{x_1x_2...x_i | x_i = 1\}$, where x_1 is the most significant bit of x. Let $T_j^x[i]$ denote the i^{th} element of T_j^x, if it exists. Lin and Tzeng's protocol relies on the following observation:

$$\begin{aligned} a > b &\iff T_1^a \cap T_0^b \neq \emptyset \\ &\iff \exists i, T_1^a[i] = T_0^b[i] \end{aligned}$$

Note that $T_1^a \cap T_0^b$ contains at most one element. The two parties, having values a and b, locally compute these two sets and then privately determine the size of the intersection. This intersection is done in Step 2 by computing homomorphically

$$\delta_i = k_i \cdot (h(T_1^a[i]) - h(T_0^b[i])) \ ,$$

which equals 0 if $T_1^a[i] = T_0^b[i]$. Observe that the index i^* such that $T_1^a[i^*] = T_0^b[i^*]$ leaks how many most significant bits x and y have in common. Thus, the vector δ is shuffled in Step 2

before decryption in Step 3.

Our secure maximum computation protocol that is presented in Section 3.5 is based on a modified version of Lin and Tzeng's protocol.

3.2.4 Secure Multi-Party Maximum Protocol

This problem generalizes the millionaires' problem to n users. Every participant P_i has a value x_i and we want to know who possesses the maximum max$(x_i|1 \leq i \leq n)$, without revealing any of the n values including the maximum. An alternative goal would be to learn the maximal value while keeping hidden the identity of its owner and the other values.

Generic solutions based on secret sharing could be used to answer that problem, but they would imply a huge number of interactions between the participants in order to share the inputs and then to obtain the result. For this reason, many custom protocols have been specifically designed to answer it more efficiently.

Two different approaches have been studied. The first one consists in running a secure two-party comparison protocol for every pair of participants. Obviously, the main difficulty of this technique is to hide the results of these intermediate comparisons. The work of Zhang and Makedon [ZM05] elegantly solves this issue with a system of vectors that hide the private values of the participant. The main steps of this approach is given in Protocol 3.2[2]. Very roughly speaking, when comparing their value, the parties P_i and P_j cannot distinguish whether they are comparing x_i with x_j or $-x_i$ with $-x_j$. Only P_1 (that has a specific role) knows but does not have access to the result of the comparison. At the end of the $n^2 - n$ comparisons, P_1 is able to determine which participant has the maximal value without leaking any further information. Thus, this family of solutions requires $O(n^2)$ comparisons.

Other solutions, like [HMMB13], aim at computing the maximum bit by bit privately. Such solutions require $O(l \cdot n^2)$ communications, where l is the bit size of the inputs, which is then equivalent to the quadratic number of comparisons of the first approach.

As already mentioned, our scheme relies on a multi-party maximum protocol. Although any existing protocol would be sufficient, we design a new multi-party maximum protocol in Section 3.5 requiring $O(n \log(n))$ computations and communications. All previously known results have their complexity in $O(n^2)$. However, our construction is based on a trade-off between efficiency and privacy, but can be generalized to any scenario where a small information leakage is acceptable.

[2]The original solution of [ZM05] outputs the maximal value. For simplicity, the presented protocol has been slightly modified to output the identity of the owner instead.

Protocol 3.2: Overview of the secure multi-party maximum computation protocol of Zhang and Makedon [ZM05]

Input: Every participant P_i $(1 \leq i \leq n)$ has a distinct value x_i and an additive encryption function $E_i(\cdot)$.
Output: P_1 obtains i_{max}.
Step 1: **forall** $i \neq 1$ **do**
  P_i generates a vector $V_i = [x_i, -x_i, x_i, -x_i, ...]$ of size $2n$, encrypts every element and sends it to P_1. We note $V_{i,k}$ the k^{th} element of V_i.
end
Step 2: P_1 does the following steps:
  Generate a permutation $\pi(\cdot)$ that randomly switches the $2j^{\text{th}}$ and the $(2j+1)^{\text{th}}$ elements of a vector of size $2n$.
  Generate a random vector R of size $2n$.
  forall $i \neq 1$ compute homomorphically $V_i' = \pi(V_i) + R$ and send it to P_i.
Step 3: **forall** $i \neq 1$ **do**
  P_i decrypts V_i' and creates a vector T_i of size $2n$.
  forall $1 \leq j \leq n$ **do**
    P_i and P_j run a secure comparison protocol between $V_{i,2j}$ and $V_{j,2j}$, such that only P_i learns the result.
    If $V_{i,2j} > V_{j,2j}$, P_i sets $T_{i,2j} = 1$, else $T_{i,2j} = 0$.
    P_i and P_j do the same for $V_{i,2j+1}$ and $V_{j,2j+1}$.
  end
  P_i sets $T_{i,2i} = T_{i,2i+1} = 0$.
end
Step 4: P_1 does the final step:
  forall $i \neq 1$ **do**
    Generate $T_i' = [1, 0, 1, 0, ...]$ with $T_{i,2i}' = T_{i,2i+1}' = 0$.
    Note that $\pi(T_i') = T_i \iff P_i$ has the maximal value.
    P_1 and P_i privately check if $\pi(T_i') = T_i$, such that only P_1 learns the result.
  end

3.3 Problem Statement

Let us suppose the participants have devices (e.g. phone or vehicle) equipped with directional antennas, allowing to locate a transmitting device in $90°$-quadrants with respect to their position and orientation. Depending on his location and orientation, a witness would be able to locate a prover in one of the four reference orthogonal half-planes (*north*, *east*, *south*, *west*), as shown in Fig. 3.1.

Our location problem can therefore be stated as follows. Consider n witnesses having located a prover P in half-planes with respect to their position. Looking for a location proof, P wants to obtain an authenticated description of the intersection of these half-planes, as shown in Fig. 3.2, while the witnesses want to protect their identity and private information. This can be

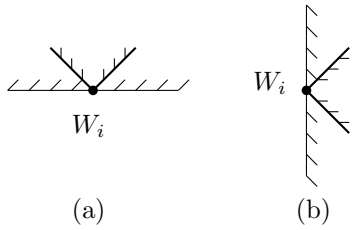


Figure 3.1: Half-planes from $90°$-quadrants

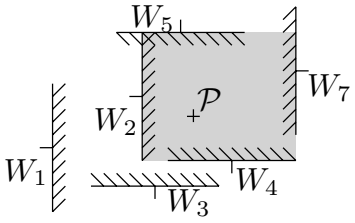


Figure 3.2: Intersecting half-planes

reduced to find the maximum (or minimum) of the private $x-$ or $y-$coordinates of the witnesses.

3.3.1 Location-Proof Generation Protocol Outline

Our method relies on an additive homomorphic encryption scheme, such as Paillier's cryptosystem [Pai99], and a unique group-signature scheme [FZ12]. The uniqueness property prevents that an accomplice of the prover P, or P himself, simulates the presence of multiple witnesses (Sybil attack). In our scheme, groups can be dynamically managed and each participant U has a signing key gsk_U. Let GS(gsk_U, m) denote the private signature function of the message m with gsk_U, and GV(g_v, m, σ) the public verification function that allows anyone to verify the signature σ of m with the public group key g_v.

Our solution also relies on a multi-party maximum computation protocol, allowing P to learn the encrypted maximum (or minimum for the north and east sides) of the private $x-$coordinate (or $y-$coordinate for the north and south sides) of the closest witness under the key of the verifier. We note $E_{\mathsf{V}}(x_{i_{min}})$, $E_{\mathsf{V}}(x_{i_{max}})$, $E_{\mathsf{V}}(y_{i_{min}})$ and $E_{\mathsf{V}}(y_{i_{max}})$ these results. Such a protocol can be obtained with small modifications of the original work of [ZM05] or from any other solution that uses an additive homomorphic encryption scheme. In later sections, we also design a new secure multi-party computation protocol that can be used instead. In comparison, it is much more efficient but has a small leakage. It consists of two sub-protocols, for which we give a brief overview for future references:

- Protocol 3.5: it is a secure multi-party maximum computation protocol that allows P to learn which witness is the closest to him, but not the encrypted coordinate.
- Protocol 3.7: if P knows which witness is the closest to him, this secure protocol allows P to retrieve its encrypted coordinate without leaking any further information.

Protocol 3.3 presents the outline of our approach. After sharing among all participants the ephemeral additive homomorphic public keys, and the directions in which the prover is located for all witnesses (Step 1), the idea is to find the intersection of the witness-defined orthogonal half-planes approximating the prover's position using multi-party computation protocols (Step 2-3), and generate a location proof from it (Step 4-5).

Protocol 3.3: Location proof generation

Input: Each participant U knows his position (x_i, y_i) and his group signature key gsk_U. The encryption function of the verifier $E_{\mathsf{V}}(\cdot)$ is public.
Output: P obtains an authenticated location proof from his neighbor witnesses.
Step 1: **Initialization**
P broadcasts a request: "*I'd like location proofs at time* τ".
forall *accepting witness* W_i **do**
Find the direction d_i of P (*North*, *South*, *West* or *East*).
Generate an ephemeral public key N_{W_i}.
Send back (d_i, N_{W_i}) to P.
end
P broadcasts to all witnesses $\mu = (\tau, N_{\mathsf{P}}, (d_i, N_{W_i})_{1 \leq i \leq n})$ and $GS(gsk_{\mathsf{P}}, \mu)$.
N_{P} is his ephemeral public key.
forall *accepting witness* W_i **do**
Find his key N_{W_i} in the properly signed message. If not, abort.
Return the signature $GS(gsk_{W_i}, \mu)$.
end
P broadcasts $\{GS(gsk_{W_i}, \mu) | 1 \leq i \leq n\}$.
forall *accepting witness* W_i **do**
Find if all the signatures are valid and different. If not, abort.
end
Step 2: **forall** *accepting witness* W_i **do**
Run a min/max computation protocol with all witnesses (Either [ZM05] or Protocol 3.5).
end
Step 3: P gets $E_{\mathsf{V}}(x_{i_{min}})$, $E_{\mathsf{V}}(x_{i_{max}})$, $E_{\mathsf{V}}(y_{i_{min}})$ and $E_{\mathsf{V}}(y_{i_{max}})$ (Either [ZM05] or Protocol 3.7).
Step 4: P transfers these encrypted results to all witnesses.
Step 5: All W_i sign the proof, using gsk_{W_i}, and send it to P (this will be detailed in Protocol 3.4).

3.3.2 Adversary Models

In this chapter, we stress that there are two different motivations for the prover. First, the main motivation of a malicious prover is to obtain a valid proof that he is at a given location at a given time, when in fact he is somewhere else. In this case, the prover has to deviate from the protocol, while remaining undetected. Otherwise, legitimate witnesses would abort and alert the judge.

On the other hand, a curious prover may be interested in getting information about his neighbours (identity or precise location). Since the identity of a witness relies on the security of the group-signature scheme used, the potential risk is low. At best, the prover can expect to get the location of an unknown participant.

The witnesses could be interested in discovering more information on their neighbours.

However, since a witness has far fewer possibilities than a prover, a malicious witness would be better to act as a prover with his neighbours.

Similarly, the verifier does not participate in the gathering location protocol and thus can only follow the semi-honest adversary model to try to get more information about the witnesses and the prover.

Finally, notice that a prover can always obtain a valid but faked location proof from accomplices. The verifier and the judge can always determine the number of witnesses having participated in the protocol. If they determine that this number is too low, they may reject the valid proof anyway.

To sum up, our scheme is secure against the following adversaries :

- A malicious prover willing to obtain fake location proofs.
- A semi-honest prover, witness or verifier trying to violate other participants' privacy.

In the rest of the chapter, Section 3.4 presents how to build encrypted location proofs against a malicious adversary, and how to verify them. In Section 3.5, a new solution to the secure multi-party maximum computation problem is described. It relies on a modified version of a classical two-party comparison protocol presented in Section 3.6 and is optimized in the context of our location-proof system.

3.4 Location-Proof Gathering and Verifying

Let us first assume that the prover P has obtained somehow the four encrypted optimum values $E_{\mathsf{V}}(x_{i_{min}})$, $E_{\mathsf{V}}(x_{i_{max}})$, $E_{\mathsf{V}}(y_{i_{min}})$ and $E_{\mathsf{V}}(y_{i_{max}})$ describing the rectangle in which he lies. Section 3.5 presents how to obtain them from his neighbouring witnesses. Unfortunately, nothing proves that he has not chosen these values himself and encrypted them with the verifier public key. The goal of Step 4 and Step 5 of Protocol 3.3 is specifically to prevent this malicious behaviour. In this section, we design a protocol allowing the witnesses to certify these optimum values. In this section, we will focus only on one of these values, say $E_{\mathsf{V}}(x_{i_{max}})$.

3.4.1 Location-Proof Gathering

Let us assume w.l.o.g. that the public key N_{V} of the verifier is 2048-bit long and that the witnesses are at most at one kilometre from the prover. If the scale of the grid system is one metre, the difference $x_{i_{max}} - x_i \leq 2^{10}$ uses at most $l_x = 10$ bits. We define $l_k = |N_V| - (l_x + 1)$. Our method for generating the location proofs is presented in Protocol 3.4. If a witness follows the

Protocol 3.4: Location-proof gathering protocol

Input: P knows $E_{\mathsf{V}}(x_{i_{max}})$. Each witness W_i has his value x_i and his signature key gsk_{W_i}. Each witness knows the number of participants n, $\mathsf{GS}(gsk_{\mathsf{P}}, \mu)$, and the verifier semi-homomorphic encryption function $E_{\mathsf{V}}(\cdot)$.
Output: P obtains a location proof from each witness.
Step 1: P broadcasts the randomized version of $E_{\mathsf{V}}(x_{i_{max}})$.
Step 2: **forall** *witness* W_i **do**
  Choose randomly $k_i \in_R [\![2^{l_x+1}; 2^{l_k} - 1]\!]$ and $r_i \in_R [\![-2^{l_x} + 1; 2^{l_x} - 1]\!]$.
  Compute $E_{\mathsf{V}}(k_i(x_{i_{max}} - x_i) + r_i) = (E_{\mathsf{V}}(x_{i_{max}}) \cdot E_{\mathsf{V}}(-x_i))^{k_i} \cdot E_{\mathsf{V}}(r_i)$.
  Send $E_{\mathsf{V}}(k_i(x_{i_{max}} - x_i) + r_i)$ to P.
end
Step 3: P broadcasts $\{E_{\mathsf{V}}(k_i(x_{i_{max}} - x_i) + r_i) | 1 \leq i \leq n\}$.
Step 4: **forall** *witness* W_i **do**
  Check the presence of $E_{\mathsf{V}}(k_i(x_{i_{max}} - x_i) + r_i)$. If not, abort.
  Define $\nu = ((E_{\mathsf{V}}(k_i(x_{i_{max}} - x_i) + r_i))_{1 \leq i \leq n}, E_{\mathsf{V}}(x_{i_{max}}), n, \mathsf{GS}(gsk_{\mathsf{P}}, \mu))$.
  Sign $\sigma_i = \mathsf{GS}(gsk_{W_i}, \nu)$ and send it to P.
end
Step 5: P stores ν, $\mathsf{GS}(gsk_{\mathsf{P}}, \nu)$ and all witness signatures σ_i.

protocol, the verifier would be able to retrieve the value $k_i(x_{i_{max}} - x_i) + r_i$, which is such that:

$$k_i(x_{i_{max}} - x_i) + r_i > \ 2^{l_x} \text{ iff } x_i < x_{i_{max}} \tag{1}$$

$$-2^{l_x} < k_i(x_{i_{max}} - x_i) + r_i < \ 2^{l_x} \text{ iff } x_i = x_{i_{max}} \tag{2}$$

$$k_i(x_{i_{max}} - x_i) + r_i < -2^{l_x} \text{ iff } x_i > x_{i_{max}} \tag{3}$$

If all the participants follow the protocols, Case (2) must happen at least once and Case (3) never. This can be confirmed by the verifier V. Thus, V can detect if a malicious prover deviates in Step 1 and uses an invalid value. On the other hand, if a malicious prover deviates in Step 3 and drops (or alters) some values, at least one witness can abort the protocol and alert the judge, by sending him any value signed by the prover (such as $GS(gsk_{\mathsf{P}}, \mu)$ of Protocol 3.3), which the judge can trace thanks to the properties of the group-signature scheme. Finally, the prover cannot deviate in Step 5 due to the unique group-signature scheme.

3.4.2 Security Properties of the Overall Process

We have now to argue that the overall process to obtain the location proofs respects all the security properties listed in the introduction.

Since the unique group signature scheme [FZ12] is unforgeable, the prover P cannot forge new proofs, except with his own key. In Step 5 of Protocol 3.3, such an opportunity is impossi-

ble. P would have to generate two distinct signatures on the same message, contradicting the uniqueness property of the signature scheme. In fact, the judge would identify any transgressing participant in this step, due to the traceability property of the signature scheme. Thus, the unforgeability and traceability properties of our location-proof protocol are ensured.

In Step 1 of Protocol 3.3, the prover broadcasts a message μ and its signature $\mathsf{GS}(gsk_{\mathsf{P}}, \mu)$. This links the timestamp and the n ephemeral keys of the witnesses. Since this signature is included in the final proofs signed by the witnesses, the location proof is valid only for the participant able to produce the valid signature $\mathsf{GS}(gsk_{\mathsf{P}}, \mu)$, confirming the non-transferability property of the protocol.

Due to the unlinkability property of the group signature scheme, the location proof associated to $\mathsf{GS}(gsk_{\mathsf{P}}, \mu)$ would not be linkable with another location proof associated to a different signature $\mathsf{GS}(gsk_{\mathsf{P}}, \mu')$ done by the same prover. Similarly, the signatures of the witnesses in Protocol 3.4 would also not be linkable. Thus, the unlinkability property of our location-proof protocol is guaranteed.

The privacy of the identities follows from the property of group signature scheme. Similarly, the privacy of the positions (x_i, y_i) relies on the semantic property of the encryption scheme and the randomization process (see Section 3.6.2). Unfortunately, the last step of Protocol 3.4 leaks some information through $E_{\mathsf{V}}(k_i(x_{i_{max}} - x_i) + r_i)$. The verifier can guess some bits of x_i. However, we can show that the Shannon entropy $H(X|Y = k_i(x_{i_{max}} - x_i) + r_i)$ is still close to $H(X|X \leq x_{i_{max}})$. For the parameters we consider, we computed that

$$H(X|Y = k_i(x_{i_{max}} - x_i) + r_i) \geq 0.85 \cdot H(X|X \leq x_{i_{max}}) \ .$$

The prover obtains his location proofs during Step 5. Then, he stores them until he needs to convince the verifier, ensuring the storage sovereignty property.

3.4.3 Location-Proof Verifying

Finally, the correctness property has to be shown. The prover P wants to convince the verifier V that $E_{\mathsf{V}}(x_{i_{max}})$ is indeed the maximum value. So, he sends:

- His position x, the message μ and his signature $\mathsf{GS}(gsk_{\mathsf{P}}, \mu)$. The message contains the timestamp τ and the number of witnesses n (Protocol 3.3).
- The randomized value of maximum $E_{\mathsf{V}}(x_{i_{max}})$ (Either [ZM05] or Protocol 3.7).
- The n proofs $E_{\mathsf{V}}(k_i(x_{i_{max}} - x_i) + r_i)$ and the witness signatures σ_i of $\nu = ((E_{\mathsf{V}}(k_i(x_{i_{max}} - x_i) + r_i))_{1 \leq i \leq n}, E_{\mathsf{V}}(x_{i_{max}}), n, \mathsf{GS}(gsk_{\mathsf{P}}, \mu))$ (Protocol 3.4).

The verifier proceeds to several verifications. He first decrypts $E_{\mathsf{V}}(x_{i_{max}})$ and checks if $x_{i_{max}} < x$. Then, he checks that the n proofs are generated by n distinct participants, different from P. This verification is based on the uniqueness property of the group signature scheme. All the signatures of the message ν must be different. The verifier also asks the judge to check that $\mathsf{GS}(gsk_{\mathsf{P}}, \mu)$ was generated using gsk_{P}, ensuring that P took place in the proof generation protocol. The final step is to make sure that $E_{\mathsf{V}}(x_{i_{max}})$ is indeed the maximum value of the witnesses. From the values of $E_{\mathsf{V}}(k_i(x_{max} - x_i) + r_i)$ in ν, the verifier can check that there is an index j s.t. $-2^{l_x} < k_j(x_{i_{max}} - x_j) + r_j < 2^{l_x}$, and that there is no index j s.t. $k_j(x_{i_{max}} - x_j) + r_j < -2^{l_x}$.

If all the verifications succeed, the verifier should be convinced that P was indeed at the east of $x_{i_{max}}$ at the given time. If any of these steps fails, it reveals a malicious action by either the prover or a witness. But unlike the prover, witnesses do not have any incentive to cheat. If some proofs are missing, the prover might have deleted them on purpose, or a witness may have aborted because of a deviation of the prover.

3.5 Secure Multi-Party Maximum Protocol

In this section, we introduce a new approach for a secure multi-party maximum protocol. The main purpose is to enable a third party (the prover in our context) to determine the owner of the maximum value among a set of n participants (or witnesses). The prover is the only party who gets a result from this protocol.

The basic idea is to use iteratively a dedicated secure two-party comparison protocol, that (i) enables the prover P to know which one of the two witnesses owns the greater private value without having to know this value, and (ii) guarantees that if one of the witnesses has already lost a comparison against another witness, the prover would not get any further information. We assume we have such a protocol. Indeed, we will give a construction in Section 3.6 (Protocol 3.6). Let "*iterative two-party comparison protocol*" refer to this tool.

3.5.1 The Protocol Description

Protocol 3.5 presents our approach for maximum computations. The prover gathers subsets of witnesses in a binary tree. In each node, the witnesses of the associated subset are paired and the secure iterative two-party comparison protocol is used. At the end of each round, the prover gets the results of these comparisons and can eliminate half of the remaining witnesses. If a witness does not participate in any further comparison, he can deduce that he was farther away from the prover than his latest paired witness. Similarly, if one keeps participating in the protocol, he knows he has won every previous comparison. Thus, the protocol should be adapted to

Protocol 3.5: Secure maximum computation based on binary tree

Input: The witnesses $S_1 = \{W_1, W_2, \cdots, W_n\}$. Each W_i has a private value x_i.
Output: P determines $i_{max} = \arg\max\{x_i | 1 \leq i \leq n\}$.
for $i = 1$ **to** $\lceil \log(n) \rceil$ **do**
  for $j = 2^{i-1}$ **to** $2^i - 1$ **do**
    Step 1: P does the following steps :
      $\mathcal{S} = \emptyset$
      if $|S_j|$ *is odd* **then**
        Select $Single \in_R S_j$ s.t. $Single$ is not marked.
        Mark the witness $Single$ and add it to $\mathcal{S}$.
      end
      Pair the elements of $S_j \setminus \mathcal{S}$ − pair the marked witnesses.
    Step 2: Each pair of witnesses uses the iterative secure two-party comparison protocol, and P obtains the index of the owner of the greater value.
    Step 3: P selects $k \in_R \{0, 1\}$ and computes the following sets:
      $S_{2j+k} = \mathcal{S} \cup \{\text{the set of the losing witnesses}\}$
      $S_{2j+\bar{k}} = \mathcal{S} \cup \{\text{the set of the winning witnesses}\}$
  end
end
Step 4: P determines the index Set i_{max} of this witness.

ensure that witnesses keep participating in the protocol even if they have been eliminated. However, the comparisons with eliminated witnesses must be randomized and meaningless for the prover.

First assume that the number of witnesses is a power of 2. In the initial round, the prover pairs the 2^k witnesses all together. Each of these pairs runs the two-party comparison protocol. At the end of the round, the winners and the losers are gathered independently. This process is then applied recursively on each subset. Hence, two witnesses would never be paired twice together. After i iterations, there would be 2^i subsets of 2^{k-i} witnesses. One of these subsets would contain only winners and all the others would contain only losers.

Consider now the general case of n witnesses. The prover pairs the witnesses. If there is an odd number of witnesses in a subset, one of them (called $Single$ in Protocol 3.5) would be *doubled*, and considered as both a winner and a loser.

Finally, notice that the witnesses do not communicate with each other directly. Otherwise, it would be simple to find out which one is closer to the prover due to the directional antennas. Thus, communications must go through P.

3.5.2 The Protocol Security

The security of our maximum computation protocol relies on these objectives: (1) the prover cannot get any information from the two-party comparison protocol if at least one of the wit-

nesses has been already eliminated previously, (2) the prover cannot get any information on the value of any witness, and (3) the witnesses cannot get any information from the comparison protocol.

The prover does the pairing and acts as the intermediary for the two-party comparison protocol. He can then observe all the messages exchanged between the witnesses. Thus, Objectives (1) and (2) rely on the security of the two-party comparison protocol. This will be addressed in Section 3.6.

Objective (3) relies on the indistinguishability of the subsets $\mathcal{S}_j$ in the round i of Protocol 3.5, for $2^{i-1} \leq j \leq 2^i - 1$. If the two-party comparison protocol is secure, the only way for a semi-honest witness to get any information on the comparisons is to find if he is in the subset of the winners. Since the indices of the subset are chosen randomly, any of them can be the subset of the winners.

3.5.3 The Protocol Analysis

The maximum computation problem has already been studied (e.g. [CFIK03, HBB12]). However, the computational and communication complexities of these solutions are in $O(n^2)$. Such complexities are not suitable for portable or embedded devices. In comparison, our method only requires $O(n \log(n))$ two-party comparisons, at the cost of leaking $n - 1$ comparison results involving winning witnesses. This follows directly from the underlying binary tree orchestrating the comparisons. The leaked information is not sufficient to order the witnesses.

In order to determine the complexity of Protocol 3.5, few facts must be proven. Since some witnesses may be doubled, they may be compared at least twice in any given round. We consider that the comparisons of a marked witness are resolved sequentially. In that case, two consecutive stages of comparisons are required for a round. The first step is to show that in any subset of witnesses at any round, there are at most two marked witnesses. This can be seen as an invariant of the protocol. Let us assume that a subset $\mathcal{S}_j$ contains at most two marked witnesses at the beginning of the round. If $|\mathcal{S}_j|$ is even, the subsets $\mathcal{S}_{2j}$ and $\mathcal{S}_{2j+1}$ may contain at most one marked element. Otherwise, if $|\mathcal{S}_j|$ is odd, one new witness would be marked, and the subsets $\mathcal{S}_{2j}$ and $\mathcal{S}_{2j+1}$ may contain at most two marked elements - the new one and an old one. Hence, for any subset of odd cardinality in a non-final round, there are at least one unmarked witness that can be marked and doubled if needed. Marking twice the same witness is unnecessary. As a corollary of this analysis, we have the following lemma:

Lemma 3.1. *Sets having two marked witnesses at the end of a round would contain one previously marked witness and a newly doubled witness.*

The second step is to show that any combination of comparisons can always be split into at most two stages in any given round. Consider the hypothetical cycle of comparisons between

marked witnesses in a given round.

$$\{W_{i_1}, W_{i_2}\}, \{W_{i_2}, W_{i_3}\}, \cdots, \{W_{i_k}, W_{i_1}\}$$

Each of these pairs belongs to a different subset of witnesses. If k is even, these comparisons can be split into two independent stages. This is optimal since a marked witness may have to be compared with two other witnesses. Now, if k is odd, alternate witnesses would have been just doubled in the round. By Lemma 3.1, this is impossible since the length of the cycle is odd. Hence, no cycle of comparisons of odd length may exist. Two stages per round are enough to orchestrate the comparisons. As a result, the total number of stages is greater than $\lceil \log(n) \rceil$ and lower than $2\lceil \log(n) \rceil$.

One can actually prove that no cycle can exist at all, but it does not improve the complexity further.

3.6 Secure Iterative Two-Party Comparison Protocol

In this section, we propose a specific two-party comparison protocol (Protocol 3.6) that enables a third party (the prover P) to know which one of the two participants (the witnesses A and B) owns the greater private value without having to know this value explicitly. This can be used iteratively, so that if one of the participants has already lost a comparison against another participant, he should not give any further information to the third party. Such a protocol can be obtained by adapting the protocol of Lin and Tzeng [LT05], chosen for efficiency.

Given an integer x, let us define the following sets for our comparison protocol: $T_0^x = \{x_1 x_2 ... x_{i-1} 1 | x_i = 0\}$ and $T_1^x = \{x_1 x_2 ... x_i | x_i = 1\}$. Let $T_j^x[i]$ denote the i^{th} element of T_j^x, if it exists. Lin and Tzeng's protocol relies on this lemma:

Lemma 3.2. *[LT05] For $x, y \in \mathbb{N}$, $x > y$ if and only if $T_1^x \cap T_0^y \neq \emptyset$.*

Our comparison protocol has been developed to be used in our multi-party maximum protocol presented in the previous section. It relies heavily on a probabilistic additive encryption scheme such as Paillier's cryptosystem [Pai99]. The participants use their ephemeral encryption keys broadcast in Protocol 3.3. These keys are signed by the prover and verified by all the nearby witnesses. This associates the keys to a particular session of the protocol. As mentioned previously, there should be no direct communication between the participants.

3.6.1 The Protocol Correctness

Let us first assume that the private values s_A and s_B have been initialized to zero by A and B, respectively. To simplify the notations, let us assume w.l.o.g. that the permutation functions are

Protocol 3.6: Secure iterative two-party comparison protocol determining which participant has the greater private value.

Input: The l-bit private values a and b of A and B. The encryption functions $E_A(\cdot)$, $E_B(\cdot)$ and $E_{\mathsf{P}}(\cdot)$, with keys N_A, N_B and N_{P}. The private values $E_{\mathsf{P}}(s_A)$ and $E_{\mathsf{P}}(s_B)$ of A and B, respectively. The hash function $h(\cdot)$.

Output: P determines whether $a > b$ or $a \leq b$.

Step 1: A does the following steps :

Compute T_1^a and the l-element vector γ, so that $\gamma_i = h(T_1^a[i])$ if it exists, otherwise, γ_i is simply a random value.
Pick a random $c \in_R \mathbb{Z}_{N_B}$.
Return $(E_A(\gamma_1), \cdots, E_A(\gamma_l))$ and $E_B(c)$ to B through P.

Step 2: B does the following steps after decrypting $E_B(c)$:

Compute T_0^b and the l-element vector δ
$E_A(\delta_i) = E_A(k_i(h(T_1^a[i]) - h(T_0^b[i])) + r_B)$
$\quad = (E_A(\gamma_i) \cdot E_A(-h(T_0^b[i])))^{k_i} \cdot E_A(r_B)$
where $k_i, r_B \in_R \mathbb{Z}_{N_A}$ s.t. $(k_i, N_A) = 1$. Otherwise, δ_i is a random value.;
Pick randomly a permutation $\pi_B(\cdot)$ and $\alpha, \beta \in_R \mathbb{Z}_{N_{\mathsf{P}}}$ s.t. $(\alpha, N_{\mathsf{P}}) = 1$.
Return $E_{\mathsf{P}}(s_B - r_B + c)$, $E_A(\alpha)$, $E_A(\beta)$ and
$(E_A(\delta_1^*), \cdots, E_A(\delta_l^*)) = \pi_B(E_A(\delta_1), \cdots, E_A(\delta_l))$ to A through P.

Step 3: A does the following steps :

Decrypt the elements $E_A(\delta_i^*)$ and compute the vector μ homomorphically
$E_{\mathsf{P}}(\mu_i) = E_{\mathsf{P}}((\delta_i^* - r_B + s_B + s_A + r_{A,i}) \cdot {r_{A,i}}^{-1})$
$\quad = (E_{\mathsf{P}}(\delta_i^* + r_{A,i}) \cdot E_{\mathsf{P}}(s_A) \cdot E_{\mathsf{P}}(s_B - r_B + c) \cdot E_{\mathsf{P}}(-c))^{{r_{A,i}}^{-1}}$
where $r_{A,i} \in_R \mathbb{Z}_{N_{\mathsf{P}}}$ s.t. $(r_{A,i}, N_{\mathsf{P}}) = 1$.
Return $(E_{\mathsf{P}}(\mu_1^*), \cdots, E_{\mathsf{P}}(\mu_l^*)) = \pi_A(E_{\mathsf{P}}(\mu_1), \cdots, E_{\mathsf{P}}(\mu_l))$,
where $\pi_A(\cdot)$ is a random permutation, to P.

Step 4: P decrypts the cyphertexts $E_{\mathsf{P}}(\mu_i^*)$.

If one of the elements of μ^* is equal to 1, then $a > b$ and P sets $s'_A = 0$. Otherwise, $a \leq b$ and P sets $s'_A = 1$. P returns $E_{\mathsf{P}}(s'_A)$ to A.

Step 5: A does the following steps, once α and β have been retrieved :

Update $E_{\mathsf{P}}(s_A) \leftarrow E_{\mathsf{P}}(s_A + k_A \cdot s'_A)$ using $E_{\mathsf{P}}(s_A) \cdot E_{\mathsf{P}}(s'_A)^{k_A}$, where $k_A \in_R \mathbb{Z}_{N_{\mathsf{P}}}$.
Return $E_{\mathsf{P}}(\alpha s'_B + \beta) = (E_{\mathsf{P}}(1) \cdot E_{\mathsf{P}}(s'_A)^{-1})^{\alpha} \cdot E_{\mathsf{P}}(\beta)$ to B through P,
since $s'_B = 1 - s'_A$.

Step 6: B does the following steps :

Retrieve $E_{\mathsf{P}}(s'_B) = (E_{\mathsf{P}}(\alpha s'_B + \beta) \cdot E_{\mathsf{P}}(-\beta))^{\alpha^{-1}}$.
Update $E_{\mathsf{P}}(s_B) \leftarrow E_{\mathsf{P}}(s_B + k_B \cdot s'_B)$ using $E_P(s_B) \cdot E_P(s'_B)^{k_B}$, where $k_B \in_R \mathbb{Z}_{N_{\mathsf{P}}}$.

the identity function. At the end of Step 2, there is an index i^* such that $\delta_{i^*} = r_B$, iff $a > b$. This follows from Lemma 3.2 and the fact that the hash function is collision-free. Consequently, at the end of Step 3, if s_A and s_B are both still equal to 0, there would be an element $\mu_{i^*} = r_{A_{i^*}} \cdot r_{A_{i^*}}^{-1} = 1$, if and only if $a > b$. Thus, P would know the result of the comparison. On the other hand, if at least one of the participants has randomized his private value $E_{\mathsf{P}}(s_*)$, due to a previous comparison, no element of the vector μ would be equal to 1, except if $\delta_i - r_B + s_A + s_B \equiv 0$

$\mod \mathbb{Z}_{N_\mathsf{P}}$. In any case, the result would be meaningless.

In Step 4, P obtains the result and defines $s'_A = 0$ if A has won the comparison ($a > b$) or $s'_A = 1$ otherwise. In Step 5, A updates homomorphically his value $s_A \leftarrow s_A + k_A \cdot s'_A$. Then it is easy to verify that if A has won every past comparison and the current one, then s_A remains null. But if A has lost at least one comparison, then s_A becomes or remains random. At the end of Step 5 and in Step 6, B (with the help of A) obtains homomorphically $s'_B = \overline{s'_A}$ and updates $s_B \leftarrow s_B + k_B \cdot s'_B$. Remark that $s'_B = 0$ if and only if B has won this comparison. Therefore, s_B remains zero if and only if B has not lost any comparison. This also implies that at the end of the protocol, at least one of s_A and s_B is randomized.

3.6.2 The Protocol Security

To prove the security of Protocol 3.6 against semi-honest polynomially-bounded adversaries trying to get more information on other participants, we have to show that these objectives are achieved: (1) A does not learn anything about b, (2) B does not learn anything about a, (3) P cannot find neither a nor b, (4) the result of the comparison is known only to P, (5) no one knows the first index i^* that differentiates a and b, (6) P eliminates A or B, (7) there is no information leaking if A or B has been already discarded, and (8) P cannot simulate A or B and have a coherent result.

Proof of Objective (1). At the beginning of Step 3, A learns $\pi_B(\delta_1, \cdots, \delta_l)$ with

$$\delta_i = k_i(h(T_1^a[i]) - h(T_0^b[i])) + r_B$$

Remember that $h(T_0^b[i])$ can be seen as an encoding of b. Let us prove that the vector δ does not leak any information about b.

W.l.o.g. assume that $\pi_B(\cdot)$ is the permutation identity. Let us take any value $b' \neq b$ and show that the same vector δ can be obtained from b' and thus does not leak any information.

If $a > b'$, let i^* be the index such that $T_1^a[i^*] = T_0^{b'}[i^*]$ and take $r_B = \delta_{i^*}$. On the other hand, if $a \leq b'$, we can choose arbitrarily r_B. Now if we take:

$$k_i = (\delta_i - r_B) \cdot (h(T_1^a[i]) - h(T_0^{b'}[i]))^{-1} \quad \forall i \neq i^*$$

then we obtain the same vector δ.

This can be generalized to permutation $\pi_B(\cdot)$. Hence, δ can be obtained from any value of b' with the same probability, and does not therefore leak any information about b.

It remains to prove that A does not learn the result of the comparison (part of Objective (4)) which would leak partial information about b. The result of the comparison (either in the

vector μ, the value s_A or s_B) is always encrypted under the public key of P. We assume the cryptosystem is semantically secure, which ends the proof of Objective (1). □

Proof of Objectif (2). Consider the information sent by A in Step 1. T_1^a gives a bit-encoding of a. Due to the semantic security of Paillier's cryptosystem, P and B cannot get any information on a (also part of Objective (3)).

Notice that the exact same γ (including random values) must be produced by A at any iteration. Otherwise, a collusion of P and B can set $E_A(\delta_i) = E_A(\gamma_i) \cdot E_A(\gamma_i')^{-1}$ and have an encoding of a. Either δ_i would be equal to 0, if $a_i = 1$, or be a random value, if $a_i = 0$.

The result of the comparison (either in the vector μ, the value s_A or s_B) is always encrypted under the public key of P. We assume the cryptosystem is semantically secure, which ends the proof of Objective (2). □

Proof of Objective (3). At the beginning of Step 4, P learns $\pi_A(\mu_1, \cdots, \mu_l)$ where

$$\mu_i = (\delta_i^* - r_B + s_B + s_A + r_{A,i}) \cdot {r_{A,i}}^{-1}$$

and $\delta_i^* = \delta_{\pi_B(i)}$. To simplify notations, assume that $\pi_A(\cdot)$ and $\pi_B(\cdot)$ are the identity permutation. If s_A or s_B is different from 0, this case is simple (Objective (7)): the vector μ follows an independent uniform distribution of $\mathbb{Z}_{N_\mathsf{P}}^l$. Thus, we only study the case:

$$\mu_i = (k_i(h(T_1^a[i]) - h(T_0^b[i])) + r_{A,i}) \cdot {r_{A,i}}^{-1}$$

Knowing that $a > b$, we will now show that for any couple (a', b') such that $a' > b'$, we can obtain the same vector μ with the same probability. In this case, let i^* be the index such that $T_1^{a'}[i^*] = T_0^{b'}[i^*]$. In this case, k_{i^*} can be chosen arbitrarily. For all other value $i \neq i^*$, $r_{A,i}$ can be chosen arbitrarily, and k_i can be defined as:

$$k_i = (\mu_i \cdot r_{A,i} - r_{A,i}) \cdot (h(T_1^{a'}[i]) - h(T_0^{b'}[i]))^{-1}.$$

Finally, if $a \leq b$, this is simpler. In this case, the index i^* is not defined, and the values of all $r_{A,i}$ and k_i are defined as above.

Thus, the same vector μ can be obtained. This can be done for any values of a' and b', as long as the result remains unchanged, and for any permutation $\pi_A(\cdot)$ and $\pi_B(\cdot)$. Therefore, the vector μ does not leak any information about a or b except whether $a > b$ or not, which ends the proof. □

Proof of Objective (4). Objectives (1) and (2) implies Objective (4). □

Proof of Objective (5). Due to the permutations, no information on the index i^* differentiating a and b can be inferred. □

Proof of Objective (6). In the last two steps, s_A and s_B are updated. Since P cannot infer the values of α and β in Step 2, it cannot manipulate the value of s'_B in Step 6 in such a way that $s'_B = 0$. At least one of the participants would then have his value $s_* \neq 0$. □

Proof of Objective (7). Notice that once s_A or s_B is a random number different than 0, μ^* follows an independent uniform distribution of $\mathbb{Z}^l_{N_\mathsf{P}}$. Hence, no conclusion follows from the value of μ^* in Step 4. □

Proof of Objective (8). Finally, note that δ and c are necessary to obtain the result and that they are encrypted respectively with A's and B's public keys. Assuming that these keys have been properly exchanged and have not been tampered with by P, a polynomially-bounded P cannot simulate A or B successfully. In such a case, the result of the protocol would then be meaningless. □

Let us briefly consider the collusion between A and P against B. In such a case, A and P accept to exchange all their private information. Due to $\pi_B(\cdot)$, A and P cannot obtain the index of the bit that differentiates a and b. Moreover, due to the multiplication of each element of μ by a distinct k_i, A and P cannot compute $h(T_1^a[i^*]) - h(T_0^b[i^*])$, except if the hashes are equal, which has been discarded anyhow. Thus, P does not discover more information with the help of A. Similarly, B and P do not gain more information neither. The index of the bit that differentiates a and b is hidden by the permutation $\pi_A(\cdot)$, and it is impossible to compute δ^* without knowing the values $r_{A,i}$ generated by A.

3.6.3 The Protocol Complexity

Following the fact that communications are made through P, any message sent between A and B is counted twice. The size of a public key N is denoted by $|N|$. Notice that ciphertexts are $2|N|$-bit long in Paillier's cryptosystem.

For any iteration, there are eight communications and $(10l + 22)|N|$ bits transferred. A maximum of $4l + 6$ cryptographic operations are computed by A, $2l + 8$ by B and only $l + 1$ by P. By cryptographic operations, we mean encryption, decryption and modular exponentiation. If either A or B was eliminated, P does not have to decrypt the result in Step 4: only one encryption is needed.

3.6.4 The Maximum Transfer

Using Protocols 3.5 and 3.6, the prover P knows the index i_{max} of the witness that has the maximum value. However, P needs to obtain $E_\mathsf{V}(x_{i_{max}})$, which corresponds to the maximum value encrypted with the verifier's public key. P does not want to inform which witness has

Protocol 3.7: Maximum transfer protocol

Input: P knows i_{max}. Each witness W_i has his values x_i and $E_{\mathsf{P}}(s_{W_i})$. Public keys N_{P} and N_{V} with functions $E_{\mathsf{P}}(\cdot)$ $E_{\mathsf{V}}(\cdot)$.
Output: P obtains $E_{\mathsf{V}}(x_{i_{max}})$.
Step 1: **forall** *witness* W_i **do**
  Generate a random number $\alpha_i \in_R \mathbb{Z}_{N_{\mathsf{P}}}$.;
  Compute $E_{\mathsf{P}}(\alpha_i + s_{W_i}) = E_{\mathsf{P}}(\alpha_i) \cdot E_{\mathsf{P}}(s_{W_i})$ and return it to P.
end
Step 2: P does the following steps:
  Compute $\alpha_{i_{max}}$ from $E_{\mathsf{P}}(\alpha_{i_{max}} + s_{W_{i_{max}}})$ received from $W_{i_{max}}$.
  Broadcast to all witnesses $E_{\mathsf{V}}(\alpha_{i_{max}})$.
Step 3: **forall** *witness* W_i *receiving* $E_{\mathsf{V}}(\alpha_{i_{max}})$ **do**
  Compute $E_{\mathsf{V}}(\alpha_{i_{max}} - \alpha_i + x_i) = E_{\mathsf{V}}(\alpha_{i_{max}}) \cdot E_{\mathsf{V}}(-\alpha_i) \cdot E_{\mathsf{V}}(x_i)$.
  and return it to P, only if it is the first request for that proof generation.
end
Step 4: P does the final steps:
  Receive $E_{\mathsf{V}}(x_{i_{max}})$ from $W_{i_{max}}$.
  Randomize it $E_{\mathsf{V}}(x_{i_{max}}) \leftarrow E_{\mathsf{V}}(x_{i_{max}}) \cdot r^{N_{\mathsf{V}}}$, for $r \in_R \mathbb{Z}^*_{N_{\mathsf{V}}}$.

been selected, but the discarded witnesses do not want to provide their location uselessly. Protocol 3.7 manages to reach both objectives. It relies on the fact that $W_{i_{max}}$ ends up with the internal value $E_{\mathsf{P}}(s_{W_{i_{max}}}) = E_{\mathsf{P}}(0)$ at the end of Protocol 3.6 (which correspond to s_A or s_B in Protocol 3.6). The other witnesses have a random s_{W_i}.

The security of Protocol 3.7 is easy to show. The security of all encrypted messages relies on the semantic security of the cryptosystem. In Step 1, P receives only random values from the witnesses. In Step 2, he picks one of them and broadcasts it back to all witnesses encrypted with the verifier's public key. A witness would return a meaningful value in Step 3 if and only if his internal random value α_i is the additive inverse of the value sent by P. In this case, the witness would return his encrypted position. Otherwise, he would return a random encrypted value. Finally, $E_{\mathsf{V}}(x_{i_{max}})$ is randomized to conceal it from the witness $W_{i_{max}}$. In terms of complexity, if broadcasting generates only one communication, $2n + 1$ messages of $2|N|$ bits are exchanged during the protocol.

This concludes our secure multi-party maximum protocol and allows to build our location-proof system more efficiently than with previous existing works. The complexity of the full location-proof system and of each sub-protocol is given in Section 3.7.

3.7 Complexity of the Overall System

We have detailed the computational and communication complexity in each sub-protocol, but we are now interested in the complexity of the overall location-proof system (Protocol 3.3), de-

	Cryptographic operations		Communication cost	
	Each witness	Prover	Communications	Bits sent
Protocol 1 (overall system)	negl + Protocols 3.4, 3.5, 3.7	negl + $4\times$ Protocols 3.4, 3.5, 3.7	$2m+3$	$(2m+1)(\|N\|+\|S\|)$
Protocol 3.4	negl	negl	$2n+2$	$(4n+2)\|N\|+m\|S\|$
Protocol 3.5	$< 2\lceil\log n\rceil$ $\times$Protocol 3.6	$\approx \frac{n}{2}\lceil\log n\rceil \times$ Protocol 3.6		
Protocol 3.6	$\leq 4l+6$	$l+1$ or 1	≤ 8	$\leq (10l+22)\|N\|$
Protocol 3.7	negl	negl	$2n+1$	$(4n+2)\|N\|$

Table 3.2: Complexity of the system

pending on the number of witnesses. For simplicity, let us assume there are $m = 4n$ witnesses, i.e. n in each direction. Let $|N|$ denote the size of the keys (the size of a ciphertext is simply $2|N|$ with the Paillier's cryptosystem) and $|S|$ denote the size of group signatures. We consider that the encryption, decryption functions and homomorphic operations are in $O(1)$.

Tab. 3.2 presents the number of cryptographic operations processed by the prover and by each witness, the number of communications and the bits exchanged during the different protocols. We only deal with the worst-case scenario: a marked witness for the computational complexity in $Protocol\ 3.5$, and only a witness A in Protocol 3.6. This can obviously be optimized by giving role B to marked witnesses as often as possible. The complexity of Protocol 3.5 is an approximation of the total number of comparisons. An exact formula is given in Section 3.5.3. In Protocol 3.6, it has been shown that P runs $l+1$ operations in $n-1$ comparisons, and only 1 otherwise. Thus, the number of operations done by the prover in Protocol 3.5 and 3.6 is approximately $(n-1)l + \frac{n}{2}\lceil\log n\rceil$.

To summarize, the global complexity, both in terms of computations and communication, is in $O(n\log n)$ for the prover and $O(\log n)$ for a witness. More specifically, each witness processes less than $\lceil\log n\rceil(4l+6)$ cryptographic computations and the prover makes less than $2n\lceil\log n\rceil(l+1)$. The overall system requires a total of $n(16\lceil\log n\rceil+24)+15$ communications.

In comparison, most previous location-based systems have a complexity for the prover in $O(n)$, and $O(1)$ for a witness. This is due to the fact that witnesses do not need to interact with each other. However, location privacy requires such interactions, and thus we do not reach the same objectives.

3.8 Conclusion

We have presented a privacy-aware location-proof system, allowing a prover to generate location proofs with the cooperation of nearby witnesses. Our solution is the first of its kind to provide both identity and location privacy. Our scheme relies on secure multi-party computa-

tions, allowing the prover to learn which participant is the closest, and thus to approximate more accurately the region in which he is. The proofs are then signed with a group signature scheme, protecting the identity of the participants and allowing the detection of any adversary trying to impersonate multiple witnesses. However, our scheme assumes that participants' devices are equipped with directional antennas. Although this is not a technological challenge, obtaining a similar level of privacy without these antennas is still an open problem.

As a second contribution, we also designed a new multi-party maximum computation based on a trade-off between efficiency and privacy. We showed that by leaking a few intermediate values, we can reduce the asymptotic cost to $O(n\log(n))$ instead of $O(n^2)$. Although it was originally designed specifically for our location-proof system, it can be applied to any scenario in which this leakage is acceptable.

CHAPTER 4

ON THE CONCRETE SECURITY OF GOLDREICH'S PSEUDORANDOM GENERATOR

Historically, the design of symmetric cryptographic primitives (such as block ciphers or pseudorandom generators) has been motivated by efficiency considerations (e.g. memory consumption, hardware compatibility). The field of multi-party computation, where parties want to jointly evaluate a function on secret inputs, has led to the emergence of new considerations: the efficiency of secure evaluations of primitives is strongly related to parameters such as the circuit depth of the primitive, and the number of its AND gates (as shown in Chapters 1 and 2). This observation has motivated the design of MPC-friendly primitives in several recent works (e.g. [ARS+15, CCF+16, MJSC16, GRR+16]), that aim for an efficient secure evaluation.

Local pseudorandom generators allow to expand a short random string into a long pseudorandom string, such that each output bit depends on a constant number d of input bits. Due to its extreme efficiency features, this intriguing primitive enjoys a wide variety of applications in cryptography and complexity and makes very promising candidate MPC-friendly PRGs. In the polynomial regime, where the seed is of size n and the output of size n^s for $s > 1$, the only known solution, commonly known as *Goldreich's PRG*, proceeds by applying a simple d-ary predicate to public random size-d subsets of the bits of the seed.

While the security of Goldreich's PRG has been deeply investigated, with a variety of results deriving provable security guarantees against class of attacks in some parameter regimes and necessary criteria to be satisfied by the underlying predicate, little is known about its concrete security and efficiency. Motivated by its numerous theoretical applications and the hope of getting practical instantiations for some of them, we initiate a study of the concrete security of Goldreich's PRG. Along the way, we develop a new guess-and-determine-style attack, and identify new criteria which refine existing criteria and capture the security guarantees of candidate local PRGs in a more fine-grained way.

Contents

4.1 Introduction

One of the most fundamental problems in cryptography is the question of what makes an efficiently computable function hard to invert. The quest for the simplest design which leads to a primitive resisting all known attacks is at the heart of both symmetric and asymmetric cryptography: while we might be able to build seemingly secure primitives by relying on more and more complex designs to thwart cryptanalysis attempts, such a "security by obscurity" approach is unsatisfying. Instead, as advocated almost two decades ago by Goldreich [Gol00], we should seek to construct the simplest possible function that we do not know how to invert efficiently.

Random Local Functions

In an attempt to tackle this fundamental problem, Goldreich suggested a very simple candidate one-way function as a promising target for cryptanalysis: let (n, m) be integers, and let $(\sigma^1, \ldots, \sigma^m)$ be a list of m subsets of $[n]$, such that each subset is of small size: for any $i \leq m$, $|\sigma^i| = d(n)$, where $d(n) \ll n$ (in actual instantiations, $d(n)$ can for example be logarithmic in n, or even constant). Fix a simple predicate $P : \{0,1\}^{d(n)} \mapsto \{0,1\}$, and define the function $f : \{0,1\}^n \mapsto \{0,1\}^m$ as follows: on input $x \in \{0,1\}^n$, for any subset σ of $[n]$, let $x[\sigma]$ denote the subset of the bits of x indexed by σ:

$$\sigma = [i_1, i_2, i_3] \implies x[\sigma] = [x_{i_1}, x_{i_2}, x_{i_3}]$$

Compute $f(x)$ as $P(x[\sigma^1])||\cdots||P(x[\sigma^m])$ (that is, $f(x)$ is computed by applying the predicate P to all subsets of the bits of x indexed by the sets $\sigma^1, \ldots, \sigma^m$). We call *random local functions* the functions obtained by instantiating this template.

In his initial proposal, Goldreich advocated instantiating the above methodology with $m \approx n$ and $d(n) = O(\log(n))$, and conjectured that if the subsets $(\sigma^1, \ldots, \sigma^m)$ form an expander graph[1], and for an appropriate choice of the predicate P, it should be infeasible to invert the above function f in polynomial time. While setting $d(n)$ to $O(\log(n))$ offers stronger security guarantees, the more extreme design choice $d(n) = O(1)$ (also discussed in Goldreich's paper) enhances the above candidate with an appealing feature: it enjoys constant input locality (which puts it into the complexity class NC^0)[2], hence it is highly parallelizable (it can be computed in constant parallel time). It appeared in subsequent works that a stronger variant of Goldreich's conjecture, which considers $m \gg n$ and claims that f is in fact a pseudorandom generator, was

[1] The subsets form an expander graph if for some k, every k subsets cover $k + \Omega(n)$ elements of $[n]$. In practice, it suffices to pick once for all the subsets $(\sigma^1, \ldots, \sigma^m)$ at random to guarantee that they will be expanding except with some small probability.

[2] Recall that NC^0 is the class of functions that can be computed by constant-depth circuits with bounded fan-in. In an NC^0 function, each bit of the output depends on a constant number of input bits

of particular interest; we will elaborate on this later on.

Local Pseudorandom Generators

The question of whether cryptographic primitives can exist in weak complexity classes such as NC^0 has attracted a lot of attention in the cryptographic community. A primitive of particular interest, which has been the focus of most works on the subject, is the notion of pseudorandom generators (PRGs), which are functions $G : \{0,1\}^n \mapsto \{0,1\}^m$ extending a short random seed into a longer, pseudorandom string. The existence of PRGs in NC^0 was first considered by Cryan and Miltersen in [CM01]. Remarkably, it was shown by Applebaum, Ishai, and Kushilevitz [AIK04, AIK08] that cryptographically secure pseudorandom generators (with linear stretch $m = O(n)$) exist in a complexity class as low as NC^0_4 (the class of constant depth, polysize circuits where each output bit depends on at most 4 input bits), under widely believed standard assumption for the case of PRG with sublinear stretch (such as factorization, or discrete logarithm), and under a specific intractability assumption related to the hardness of decoding "sparsely generated" linear codes, for the case of PRG with linear stretch. While this essentially settled the question of the existence of linear stretch PRGs in NC^0, an intriguing open question remained: could PRGs in NC^0 have polynomial stretch, $m = \mathsf{poly}(n)$?

Some early negative results were given by Cryan and Miltersen [CM01] (who ruled out the existence of PRGs in NC^0_3 with stretch $m > 4n$) and Mossel, Shpilka, and Trevisan [MST03] (who ruled out the existence of PRGs in NC^0_4 with stretch $m > 24n$). The authors of [CM01] also conjectured that any candidate PRG with superlinear stretch in NC^0 would be broken by simple, linear distinguishing tests[3]; this conjecture was refuted in [MST03], who gave a concrete candidate PRG in NC^0, by instantiating a random local function with $d = 5$, and the predicate

$$P_5 : (x_1, x_2, x_3, x_4, x_5) \mapsto x_1 + x_2 + x_3 + x_4 x_5 \ .$$

where the $+$ denotes the addition in $\mathbb{F}_2$ (i.e. the xor).

They proved that this PRG fools linear tests, even when m is a (sufficiently small) polynomial in n. By the previously mentioned negative result on PRGs in NC^0_4, this candidate PRG, which has locality 5, achieves the best possible locality. Recently, there has been a renewed interest in the study of this local PRG, now commonly known as Goldreich's PRG, and its generalizations [BQ09, App12, OW14, CEMT14, App15, ABR16, AL16, IPS08, LV17, BCG$^+$17].

[3] A linear test attempts to distinguish a string from random by checking whether the xor of a subset of the bits of the string is biased toward either 0 or 1.

4.1.1 Goals and Results

In this work, we continue the study of the most common candidate local pseudorandom generators. However, we significantly depart from the approach of previous works, in that we wish to analyze the *concrete* security of local PRGs. To our knowledge, all previous works were only concerned about establishing asymptotic security guarantees for candidate local PRGs, without providing any insight on, e.g. which parameters can be conjectured to lead to a primitive with a given bit-security. Our motivations for conducting this study are twofold.

- Several recent results, which we briefly overview in Section 4.2.4, indicate that (poly-stretch) local PRGs enjoy important theoretical applications. However, the possibility of instantiating these applications with concrete PRG candidates remains unclear, as their efficiency quickly deteriorates with the parameters of the underlying PRG. For example, the iO scheme of [LT17], which requires low-degree multilinear maps and therefore might be a viable approach to obtain efficiency improvements in iO constructions (as candidate high-degree multilinear maps are prohibitively expensive); however, it has a cost cubic in the seed size of a poly-stretch local PRG, which renders it practical only if we can safely use local PRGs with reasonably small seeds. Overall, we believe that there is a growing need for a better understanding of the exact efficiency of candidate local PRGs, and providing concrete estimations can prove helpful for researchers willing to understand which efficiency could potentially be obtained for local-PRG-based primitives.
- At a more theoretical level, previous works on (variants of) Goldreich's PRG have identified criteria which characterize the predicates susceptible to lead to secure local PRGs. Identifying such criteria is particularly relevant to the initial goal set up by Goldreich in [Gol00], which is to understand what characteristics of a function is the source of its cryptographic hardness, by designing the simplest possible candidate that resists all attacks we know of. However, existing criteria only distinguish predicates leading to insecure instances from those leading to instances for which no polynomial-time attack is known. We believe that it is also of particular relevance to this fundamental question to find criteria which capture in a more fine-grained way the cryptographic hardness of random local functions.

Our Results

We provide new cryptanalytic insights on the security of Goldreich's pseudorandom generator.

- *A new subexponential attack on Goldreich's PRG.* We start by devising a new attack on Goldreich's PRG. Our attack relies on a *guess-and-determine* technique, in the spirit of the recent attack [DLR16] on the FLIP family of stream ciphers [MJSC16]. The complexity

of our attack is $2^{O(n^{2-s})}$ where s is the stretch and n is the seed size. This complements O'Donnel and Witmer's result [OW14] showing that Goldreich's PRG is likely to be secure for stretch up to 1.5, with a more fine-grained complexity estimation. We implemented our attack[4] and provide experimental results regarding its concrete efficiency, for various seed size and stretch parameters.

- *Linearization and Gröbner attack.* We complement our study with an analysis of the efficiency of algebraic attacks *à la* Gröbner on Goldreich's PRG. While it is known that Goldreich's PRG (and its variants) provably resists such attacks for appropriate choices of (asymptotic) parameters [AL16], little is known about its exact security against such attacks for concrete choices of parameters. We evaluated the concrete security of Goldreich's PRG against a degree-two linearization attack. The existence of such an attack allows to derive bounds on Gröbner basis performance. Using an implemented proof of concept, we introduce heuristic bounds for vulnerable parameters.

We also generalize these results to other predicates in [CDM+18], but we refer the reader to this paper for more details.

4.1.2 Organization of the Chapter

Section 4.2 introduces necessary preliminaries on predicates and local pseudorandom generators and their applications in cryptography. Section 4.3 describes a guess-and-determine attack on Goldreich's PRG instantiated with the predicate P_5 and analyzes it. Section 4.4 investigates algebraic cryptanalysis of Goldreich's PRG with P_5, presenting a degree 2 linearization attack, and an attack using Gröbner basis approach. Finally, Section 4.5 considers the case of using Goldreich's PRG with ordered subset (as was initially advocated in [Gol00]) and provides indications that this weakens its concrete security.

4.2 Preliminaries

Throughout this chapter, n denotes the size of the seed of the PRGs considered. A probabilistic polynomial time algorithm (PPT, also denoted *efficient* algorithm) runs in time polynomial in the parameter n. A positive function f is *negligible* if for any polynomial p there exists a bound $B > 0$ such that, for any integer $k \geq B$, $f(k) \leq 1/|p(k)|$. An event depending on n occurs with *overwhelming probability* when its probability is at least $1 - \mathrm{negl}(n)$ for a negligible function negl. Given an integer k, we write $[k]$ to denote the set $\{1, \ldots, k\}$. Given a finite set S, the notation $X \xleftarrow{\$} S$ means a uniformly random assignment of an element of S to the variable X. Given a

[4] Our proof of concept can be found at `https://github.com/LuMopY/SecurityGoldreichPRG`.

string $x \in \{0,1\}^k$ for some k and a subset σ of $[k]$, we let $x[\sigma]$ denote the subsequence of the bits of x whose index belongs to σ. Moreover, the i-th bit of $x[\sigma]$ will be denoted by x_{σ_i}.

4.2.1 Hypergraphs

Hypergraphs generalize the standard notion of graphs (which are defined by a set of nodes and a set of edges, an edge being a pair of nodes) to a more general object defined by a set of nodes and a set of *hyperedges*, each hyperedge being an arbitrary subset of the nodes. We define an (n, m, d)-hypergraph G to be a hypergraph with n vertices and m hyperedges, each hyperedge having cardinality d. The hyperedges are assumed to be ordered from 1 to m, and each hyperedge $\{i_1, i_2, \ldots, i_d\}$ is ordered and satisfies $i_j \neq i_k$ for all $j \leq d$, $k \leq d$, $j \neq k$. We will consider hypergraphs satisfying some expansion property, defined below.

Definition 4.1 (Expander Graph)**.** *An (n, m, d)-hypergraph G, denoted $(\sigma^1, \ldots, \sigma^m)$, is* (α, β)-expanding *if for any $S \subset [m]$ such that $|S| \leq \alpha \cdot m$, it holds that $|\cup_{i \in S} \sigma^i| \geq \beta \cdot |S| \cdot d$.*

4.2.2 Predicates

The constructions of local pseudorandom generators that we will consider in this work rely on predicates satisfying some specific properties. Formally, a predicate P of arity d is a function $P : \{0,1\}^d \mapsto \{0,1\}$. We define below the two properties that were shown to be necessary for instantiating local PRGs:

- *Resiliency*: a predicate P is k-resilient if it has no correlation with any linear combination of up to k of its inputs. An example of predicate with maximal resiliency is the parity predicate (i.e. the predicate which xors all its inputs). For exemple, the predicate P_5

$$P_5(x_1, x_2, x_3, x_4, x_5) = x_1 + x_2 + x_3 + x_4 x_5$$

  is 2-resilient and cannot be 3-resilient since it is correlated with $x_1 + x_2 + x_3$.

- *Algebraic Immunity*: a predicate P has algebraic immunity e, referred to as $\mathsf{AI}(P) = e$, if the minimal degree of a non-null function g (called annihilator) such that $Pg = 0$ (or $(P+1)g = 0$) on all its entries is e. For example, $g(x) = 1 + x_1$ is an annihilator of $P(x) = x_1 x_2 x_3 x_4$ since:

$$(1 + x_1)(x_1 x_2 x_3 x_4) = x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_4 = 0$$

  A local PRG built from an AI-e predicate cannot be pseudorandom with a stretch n^e due to algebraic attacks.

Note that the algebraic immunity (also referred as rational degree in [AL16]) implies a lower bound on the degree and on the bit-fixing degree. Moreover, a high algebraic immunity implies at least the same degree. Hence, for now on, those two criteria are considered as the relevant criteria for evaluating the security of Goldreich's PRG.

We define a particular family of predicates which have been considered as a potential instantiation:

Definition 4.2 ($\mathsf{XOR}_\ell\mathsf{M}_k$ predicates)**.** *We call* $\mathsf{XOR}_\ell\mathsf{M}_k$ *predicate a predicate* P *of arity* $\ell + k$ *such that* M *is a predicate of arity* k *and:*

$$P(x_1, \ldots, x_\ell, z_1, \ldots, z_k) = \sum_{i=1}^{\ell} x_i + M(z_1, \ldots, z_k) \ .$$

We also define a subfamily of $\mathsf{XOR}_\ell\mathsf{M}_k$ predicates, which have been considered in [AL16]:

Definition 4.3 ($\mathsf{XOR}_\ell\mathsf{MAJ}_k$ predicates)**.** *We call* $\mathsf{XOR}_\ell\mathsf{MAJ}_k$ *predicate a predicate* P *of arity* $\ell+k$ *such that* P *is a* $\mathsf{XOR}_\ell\mathsf{M}_k$ *predicate such that* M *is the majority function in* k *variables:*

$$M(z_1, \ldots, z_k) = 1 \Leftrightarrow \mathsf{w}_H(z_1, \ldots, z_k) \geq \left\lceil \frac{k}{2} \right\rceil \ ,$$

where w_H *denotes the Hamming weight.*

4.2.3 Pseudorandom Generators

Definition

A pseudorandom generator is a deterministic process that expands a short random seed into a longer sequence, so that no efficient adversary can distinguish this sequence from a uniformly random string of the same length.

Definition 4.4 (Pseudorandom Generator)**.** *A* $m(n)$*-stretch pseudorandom generator, for a polynomial* m*, is an efficient uniform deterministic algorithm* PRG *which, on input a seed* $x \in \{0,1\}^n$*, outputs a string* $y \in \{0,1\}^{m(n)}$*. It satisfies the following security notion: for any probabilistic polynomial-time adversary* Adv*,*

$$\begin{aligned} &\Pr[y \xleftarrow{\$} \{0,1\}^{m(n)} : \mathsf{Adv}(\mathsf{pp}, y) = 1] \\ \approx &\Pr[x \xleftarrow{\$} \{0,1\}^n, y \leftarrow \mathsf{PRG}(x) : \mathsf{Adv}(\mathsf{pp}, y) = 1] \end{aligned}$$

Here $\approx$ *denotes that the absolute value of the difference of the two probabilities is negligible in the security parameters, and* pp *stands for the public parameters of the* PRG*.*

Roughly said, a PRG should ensure that no probabilistic polynomial-time adversary should be able to distinguish the output of the PRG from a uniformly random output of same size.

For any $n \in \mathbb{N}$, we denote PRG_n the function PRG restricted to n-bit inputs.

Definition 4.5 (Local Pseudorandom Generator)**.** *A pseudorandom generator* PRG *is* d-local *(for a constant* d*) if for any* $n \in \mathbb{N}$*, every output bit of* PRG_n *depends on at most* d *input bits.*

Goldreich's Pseudorandom Generator

Goldreich's candidate local PRGs form a family $\mathsf{F}_{G,P}$ of local PRGs: $\mathsf{PRG}_{G,P} : \{0,1\}^n \mapsto \{0,1\}^m$, parametrized by an (n, m, d)-hypergraph $G = (\sigma^1, \ldots, \sigma^m)$ (where m is polynomial in n), and a predicate $P : \{0,1\}^d \mapsto \{0,1\}$, defined as follows: on input $x \in \{0,1\}^n$, $\mathsf{PRG}_{G,P}$ returns the m-bit string $(P(x_{\sigma_1^1}, \ldots, x_{\sigma_d^1}), \ldots, P(x_{\sigma_1^m}, \ldots, x_{\sigma_d^m}))$.

Conjecture 4.1 (Informal)**.** *If* G *is a sufficiently expanding* (n, m, d) *hypergraph and* P *is a predicate with sufficiently high resiliency and high algebraic immunity, then the function* $\mathsf{PRG}_{G,P}$ *is a secure pseudorandom generator.*

Note that picking an hypergraph G uniformly at random suffices to ensure that it will be expanding with probability $1 - o(1)$. However, picking a random graph will always give a non-negligible probability of having an insecure PRG. To see that, observe that when the locality d is constant, a random hypergraph G will have two hyperedges containing the same vertices with probability $1/\mathrm{poly}(n)$; for any such graph G, the output of $\mathsf{PRG}_{G,P}$ on a random input can be trivially distinguished from random. Therefore, the security of random local functions is usually formulated non-uniformly, by stating that for a $1 - o(1)$ fraction of all hypergraphs G (and appropriate choice of P), no polytime adversary should be able to distinguish the output of $\mathsf{PRG}_{G,P}$ from random with non-negligible probability.

Fixed hypergraph versus random hypergraphs

Goldreich's candidates local pseudorandom generators require to use a sufficiently expanding hypergraph. Unfortunately, building concrete graphs satisfying the appropriate expansion properties is a non-trivial task. Indeed, all known concrete constructions of expanding bipartite hypergraphs fail to achieve parameters which would allow to construct a PRG with constant locality. Therefore, to our knowledge, in all works using local PRG (see e.g. [IKOS08, App13, Lin17, ADI$^+$17a, BCG$^+$17]), it is always assumed (implicitly or explicitly) that the hypergraph G of the PRG is picked uniformly at random (which makes it sufficiently expanding with probability $1 - o(1)$, even in the constant-locality setting) in a one-time setup phase. Therefore, this is the setting we assume for our cryptanalysis.

Notations

In the first part of this work, we focus on the predicate P_5, assuming that the subsets $\sigma^1, ..., \sigma^m$ are random subsets. The predicate P_5 can be regarded as a Boolean function of five variables:

$$P_5(x_1, x_2, x_3, x_4, x_5) = x_1 + x_2 + x_3 + x_4 x_5 \ .$$

The predicate P_5 has algebraic degree 2 and an algebraic immunity of 2, and is 2-resilient. Let n be the size of the input (i.e. the number of initial random bits). We define the stretch s and denote the size m of the output as $m = n^{\mathsf{s}}$. Let $x_1, \ldots, x_n \in \mathbb{F}_2$ be the input random bits and $y_1, \ldots, y_m \in \mathbb{F}_2$ be the output bits. The m public equations E_i for $1 \leq i \leq m$ are drawn as follows:

- a subsequence of $[n]$ of size 5 is chosen uniformly at random. Let us call it

$$\sigma^i = [\sigma_1^i, \sigma_2^i, \sigma_3^i, \sigma_4^i, \sigma_5^i] \ .$$

- E_i is the quadratic equation of the form

$$x_{\sigma_1^i} + x_{\sigma_2^i} + x_{\sigma_3^i} + x_{\sigma_4^i} x_{\sigma_5^i} = y_i \ .$$

The public system Σ that we consider is then defined with the m equations, that is $(E_i)_{1 \leq i \leq m}$.

Ordered and unordered

There are two different cases to consider:

- *The ordered case*: σ^i is ordered, i.e. $\sigma_1^i < \sigma_2^i < \sigma_3^i < \sigma_4^i < \sigma_5^i$.
- *The unordered case*: the order σ^i's elements is arbitrary.

However, we consider the unordered case in this chapter, as we will provide evidence that the vulnerabilities are even more important for the ordered case in Section 4.5.

Matrix inversion complexity

Our attacks of Section 4.3 require a sparse matrix inversion algorithm. We consider the Wiedemann's algorithm [Wie86], the complexity of which is $O(n^2)$ in this context, since there are less than $d \cdot n$ non-zero elements of our matrices. Other algorithms could be used, but the complexity of our attacks would have to be modified accordingly. For other Sections, with arbitrary matrices, we denote by ω the exponent for matrix inversion complexity $O(n^\omega)$.

4.2.4 Implications of Polynomial-Stretch Local Pseudorandom Generators

The original motivation for the study of local pseudorandom generators was the intriguing possibility of designing cryptographic primitives that can be evaluated in *constant time*, using polynomially many cores. While this is already a strong motivation in itself, it was observed in several works that the existence of (poly-stretch) local PRGs had a number of non-trivial implications, and is at the heart of feasibility results for several high-end cryptographic primitives. Beside being very good MPC-friendly candidates, local pseudorandom generators have impacts in the following (non-exhaustive) primitives:

- *Secure computation with constant computational overhead.* In the recent work [IKOS08], the authors explored the possibility of computing cryptographic primitives with essentially optimal efficiency, namely, constant overhead over a naive insecure implementation of the same task. One of their main results establishes the existence of constant-overhead two-party computation protocols for any boolean circuit, assuming the existence of poly-stretch local PRGs (and oblivious transfers). In a recent work [ADI$^+$17b], this result was extended to arithmetic circuits, using an arithmetic generalization of local PRGs.

- *Indistinguishability obfuscation (iO).* Introduced in the paper of Barak et al. [BGI$^+$01], iO is a primitive that has received a considerable attention in the past years, as a long sequence of works starting with [SW14] has demonstrated that iO had tremendous theoretical implications, to the point that it is often referred to as being a "crypto-complete" primitive. All known candidate constructions of iO rely, directly or indirectly, on a primitive called k-linear map, for some degree k. Recently, a sequence of papers (culminating with [LT17]) has attempted to find out the minimal k for which a k-linear map would imply the existence of iO (with the ultimate goal of reaching $k = 2$, as bilinear maps are well understood objects). These works have established a close relation between this value k and the existence of pseudorandom generators with poly-stretch, and locality k.[5]

- *Cryptographic capsules.* In [BCG$^+$17], Boyle et al. studied the homomorphic secret sharing (HSS). An important implication of HSS is that, assuming the existence of a local PRG with poly-stretch, one can obtain MPC protocols in the preprocessing model where the amount of communication between the parties is considerably smaller than the circuit size of the function, by constructing a primitive called cryptographic capsule which, informally, allows to compress correlated (pseudo-)random coins. MPC protocols with low-communication preprocessing have numerous appealing applications. However, the efficiency of the constructions of cryptographic capsule strongly depends on the locality

[5] The locality requirement can in fact be weakened to a related notion of *block locality*.

and seed size of the underlying local PRG (both should be as small as possible to get a reasonably efficient instantiation).

4.2.5 On the Security of Goldreich's PRG

In this section, we provide a brief overview of the state-of-the-art regarding the security of local pseudorandom generators. For a more detailed and well-written overview dating from 2015, we refer the reader to [App15].

Positive Results: Security against Class of Attacks

The seminal paper of Goldreich [Gol00] made some preliminary observations on necessary properties for a local one-way function. The predicate P must satisfy some non-degeneracy properties, such as being non-linear (otherwise, one could inverse the function using Gaussian elimination). It also noted that to avoid a large class of natural "backtracking" attacks, which make a guess on the values of bit inputs based on local observations and attempt to combine many local solutions into a global solution, the subsets $(S_1, \ldots, S_m)$ should be sufficiently *expanding*: for some k, every k subsets should cover $k + \Omega(n)$ elements of $[n]$. The security of Goldreich's candidate one-way function against a large class of backtracking algorithm was formally analyzed in [AHI05, CEMT14], where it was proven that two restricted types of backtracking algorithms (called "drunk" and "myopic" backtracking algorithms) take exponential time to invert the function (with high probability). They also ran experiments to heuristically evaluate its security against SAT solvers[6] (and observed experimentally an exponential increase in running time as a function of the input length).

The pseudorandomness of random local functions was originally analyzed in [MST03]. They proved (among other results) that the random local function instantiated with the predicate P_5 fools $\mathbb{F}_2$-linear distinguishers for a stretch up to $m(n) = n^{1.25-\varepsilon}$ (for an arbitrary small constant ε). This result was later extended to a larger stretch $n^{1.5-\varepsilon}$ in [OW14]. In the same paper, the authors proved that this candidate PRG is also secure against a powerful class of attacks, the Lasserre/Parrilo semidefinite programming (SDP) hierarchy, up to the same stretch. Regarding security against $\mathbb{F}_2$-linear attacks, a general dichotomy theorem was proven in [ABR12], which identified a class of *non-degenerate* predicates and showed that for most graphs, a local PRG instantiated with a non-degenerate predicate is secure against linear attacks, and for most graphs, a local PRG instantiated with a degenerate predicate is insecure against linear distinguishers. In general, to fool $\mathbb{F}_2$-linear distinguishers, the predicate should have high *algebraic degree* (in particular, a random local function instantiated with a degree-ℓ predicate cannot be

[6] The Boolean satisfiability problem (SAT) is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. If this is the case, the formula is called satisfiable.

pseudorandom for a stretch ℓ ($m \equiv n^{\ell}$), as it is broken by a straightforward Gaussian elimination attack).

Being pseudorandom seems to be a much stronger security property than being one-way. Nevertheless, in the case of random local functions, it was shown in [App12] that the existence of local pseudorandom generators follows from the existence of *one-way* random local functions (with sufficiently large output size).

Negative Results

The result of O'Donnell and Witmer [OW14] regarding security against SDP attacks is almost optimal, as attacks from this class are known to break the candidate for a stretch $\Theta(n^{1.5} \log n)$. More generally, optimizing SDP attacks leads to a polytime inversion algorithm for any predicate P which is (even slightly) correlated with some number c of its inputs, as soon as the output size exceeds $m \in \Omega(n^{c/2} + n \log n)$ [OW14, App15]. Therefore, a good predicate should have high *resiliency* (i.e. it should be k-wise independent, for a k as large as possible). This result shows, in particular, that a random local function with a constant locality d and with an output size $m > \mathsf{poly}(d) \cdot n$ is insecure when instantiated with a uniformly random predicate P. Combining this observation with the result of Siegenthaler [Sie84], which studied the correlation of d-ary predicates, gives a polytime inversion algorithm for any random local function implemented with a d-ary predicate, and with an output size $m \in \Omega(n^{1/2\lfloor 2d/3 \rfloor} \log n)$.

Bogdanov and Qiao [BQ09] studied the security of random local functions when the output is sufficiently larger than the input (*i.e.*, $m \geq Dn$, for a large constant D). They proved that for sufficiently large D, inverting a random local function could be reduced to finding an *approximate inverse* (*i.e.* finding any x' which is close to the inverse x in Hamming distance), by showing how to invert the function with high probability given an advice x' close to x. For random local function with an output size polynomial in n, $m = n^{\mathsf{s}}$ for some s, this leads to a subexponential-time attack [App15]: fix a parameter ε, assign random values to the $(1 - 2\varepsilon)n$ first inputs, and create a list that enumerates over all possible $2\varepsilon n$ assignments for the remaining variables. Then the list is guaranteed to contain a value x' that agree with the preimage x on a $(1/2 + \varepsilon)n$ fraction of the coordinates with good probability. By applying the reduction of [BQ09], using each element of the list as an advice string, one recovers the preimage in time $\mathsf{poly}(n) \cdot 2^{2\varepsilon n}$ provided that $m = \Omega(n/\varepsilon^{2d})$ (d is the arity of the predicate P). In the case of the 5-ary predicate P_5, this leads to an attack in subexponential-time $2^{O(n^{1-(\mathsf{s}-1)/2d})}$ (*e.g.* using $\mathsf{s} = 1.45$ gives an attack in time $2^{O(n^{0.955})}$).

By the previous observations, we know that the predicate of a random local function must have high resiliency and high algebraic degree to lead to a pseudorandom function. A natural question is whether this characterization is also sufficient; this question was answered negatively in [AL16], who proved that a predicate must also have high *bit-fixing degree* to fool

linear attacks.[7] In particular, this observation disproved a previous conjecture of Applebaum that XOR-AND predicates (which are natural generalizations of the predicate P_5) could lead to local PRGs with stretch greater than 2 that fools all linear tests (see [AL16, Corollary 1.3]).

In the same work, Applebaum and Lovett considered the class of algebraic attacks on local pseudorandom function, which are incomparable to linear attacks. An algebraic attack against a function $f : \{0,1\}^n \mapsto \{0,1\}^m$ starts with an output y and uses it to initialize a system of polynomial equations over the input variables $x = (x_1, \ldots, x_n)$. The system is further manipulated and extended until a solution is found or until the system is refuted. Applebaum and Lovett proved that a predicate must also have high *rational degree* to fool algebraic attacks (a predicate P has rational degree e if it is the smallest integer for which there exist degree e polynomials Q and R, not both zero, such that $PQ = R$). Indeed, if $e <$ s then P is not s-pseudorandom against algebraic attacks (see [AL16], Theorem 1.4).

In the symmetric cryptography community, the rational degree denotes the well-known *algebraic immunity* criterion on Boolean function that underlies the so-called *algebraic attacks* on stream ciphers [CM03, Cou03]. An algebraic immunity of e implies an r-bit fixing degree greater than or equal to $e - r$ ([DGM05], Proposition 1), giving that an high algebraic immunity guarantees both high rational degree and high bit fixing degree. The algebraic degree is equivalent to the 0-bit fixing degree, then it leads to the following characterization: a predicate of a random local function must have high resiliency and high algebraic immunity. In light of this characterization, the authors of [AL16] suggested the XOR-MAJ predicate as a promising candidate for building high-stretch local PRGs, the majority function having optimal algebraic immunity [DMS05].

Security against Subexponential Attacks

While there is a large body of work that studied the security of random local functions, leading to a detailed characterization of the parameters and predicates that lead to insecure instantiations, relatively little is known on the exact security of local PRGs instantiated with non-degenerated parameters. In particular, most papers only prove that some classes of polytime attacks provably fail to break candidates local PRGs; however, these results do not preclude the possible existence of non-trivial subexponential attacks (specifically, these polytime attacks do not "degrade gracefully" into subexponential attacks when appropriate parameters are chosen for the PRG; instead, they do always and provably not succeed).

To our knowledge, the only results in this regard are the proof from [AHI05, CEMT14] that many backtracking-type attacks require exponential time to invert a random local function, and the subexponential-time attack arising from the work of Bogdanov and Qiao [BQ09]. How-

[7] A predicate P has r-bit fixing degree e if the minimal degree of the restriction of P obtained by fixing r inputs is e

ever, as we saw above, the latter attack only gives a slightly-subexponential algorithm, in time $2^{O(n^{1-(\mathsf{s}-1)/2d})}$ for a d-ary predicate, and an n^{s}-stretch local PRG.

4.3 Guess & Determine Cryptanalysis of Goldreich's PRG with P_5

4.3.1 The Attack - Asymptotic Description

We first describe a distinguishing attack, where our adversary outputs 1 when the challenged bit-stream is considered as the PRG's output and 0 when it is considered as a random string.

At a high level, the attack works by collecting a large number of linear equations, by guessing well-chosen bits of the seed, seen as a vector x of n variables. When enough equations have been collected, two cases can occur.

- Either sufficiently many equations are linearly independent (as much as the number of variables). In this case, the attacker can invert a large subsystem of equations, obtain a candidate seed, and check it against the PRG output (therefore finding out whether the guesses were correct in the first place).
- Either most of the equations are linearly dependent. In this case, we show that this implies that the PRG output must pass a large number of linear tests, which a random string would be unlikely to all pass. We use this observation to mount a distinguishing attack.

We now proceed with the formal description of the attack. Our algorithm has the description of the PRG hardcoded (namely, an $(n, m, 5)$-hypergraph $G = (\sigma^1, \cdots, \sigma^m)$, where $m = n^{\mathsf{s}}$, and each $\sigma^i = (\sigma^i_1, \cdots, \sigma^i_5)$ is a size-5 subset of $[n]$). It takes as input an m-bit string $y = y_1 \cdots y_m$, and must distinguish whether y is a random string, or whether it is in the image of PRG_{G,P_5}. The algorithm starts by considering the following list of quadratic equations for $i = 1$ to m:

$$P_5(x_{\sigma^i_1}, \cdots, x_{\sigma^i_5}) = y_i.$$

We denote Q this list. The algorithm will proceed by constructing $O(m)$ linear equations from Q.

Selection Phase

The algorithm dynamically determines a "selected" subset of the quadratic equations and a subset Σ of $[m]$, which it will use in the guessing phase. The sets are constructed using the following greedy approach: set $j \leftarrow 1$ and mark all equations of Q as "unselected". In the j^{th} step, find the variable that appears in the largest number of quadratic terms over all equations in Q which are marked "unselected". Mark all the equations in which this variable appears in

a quadratic term as "selected", and add their indexes in Q to Σ, also add the linear equation corresponding to the affectation of this variable and count it as a "selected" equation. If the number s of equations marked as "selected" satisfies $s \geq n + j$, set $\ell \leftarrow j$, $y' \leftarrow y[\Sigma]$, and proceed to the guessing phase. Otherwise, set $j \leftarrow j + 1$ and continue.

Guessing Phase

In the previous phase, the algorithm has identified a subset of ℓ variables which appear overall in the quadratic term of $s \geq n + \ell$ selected quadratic (and linear) equations. In this step, the algorithm will enumerate over all 2^ℓ possible assignments for these variables, in some arbitrary fixed order. In each step, for $i = 1$ to 2^ℓ, the algorithm obtains a system of s linear equations by assigning a value in $\{0, 1\}$ to each of the ℓ variables across all selected quadratic equations. Let A_i denote the matrix of this system. We distinguish two cases:

- *Case 1.* $\mathsf{rank}(A_i) = n$. In this case, there is an $n \times n$ invertible submatrix of A_i. The algorithm extract this submatrix, let us denote it C_i. We also denote by y'_i the subsequence of y' indexed by the position of the rows of C_i in A_i. . The algorithm computes a candidate seed $x'_i \leftarrow C_i^{-1} y'_i$, and checks whether $\mathsf{PRG}_{G,P_5}(x'_i) = y$. If it holds, it outputs 1 and halts. Else, it sets $i \leftarrow i + 1$.

- *Case 2.* $\mathsf{rank}(A_i) < n$. In this case, there exists at least $\ell + 1$ linearly dependent rows of A_i. Let B_i denote the row echelon form of A_i, obtained through Gaussian elimination, and let G_i denote the (invertible) matrix of this transformation; that is, $B_i = G_i A_i$. Let $(v_1^\intercal, \ldots, v_{\ell+1}^\intercal)$ denote the last $\ell + 1$ rows of G_i. The algorithm checks whether $v_k^\intercal y' = 0$ for $k = 1$ to $\ell + 1$. If all checks pass, it outputs 1 and halts. Else, it sets $i \leftarrow i + 1$.

If the algorithm reaches $i = 2^\ell + 1$, it outputs 0 and halts.

4.3.2 Complexity Analysis

We now analyze the complexity of the algorithm. We first estimate the average value of ℓ obtained in the selection phase. We consider the list Q of all quadratic equations. For all i such that $1 \leq i \leq n$ let denote N_i^1 the number of occurrences of x_i in degree-two monomials.

Proposition 4.1 (Number of guesses)**.** *For any instance with n variables, m equations and c collisions, an upper bound on the sufficient number of guesses required to build $n - c$ linear equations is:*

$$\ell \leq \left\lfloor \frac{n^2}{2m} + 1 \right\rfloor . \tag{4.1}$$

Proof. Let us choose the variable with more occurrences, denoted w.l.o.g. x_1, as there are m equations, $\sum_{i=1}^{n} N_i^1 = 2m$, and therefore $N_1^1 \geq 2\frac{m}{n}$. Fixing the value of x_1 we get N_1^1 linear equations (plus the linear equation fixing the value of x_1). Since the value of x_1 is fixed, the remaining quadratic system of equations consists of $m - N_1$ equations in $n - 1$ unknowns $(x_2, \ldots, x_n)$. We recursively use this strategy:

For all j $(2 \leq j \leq n)$ we denote N_i^j the number of occurrences of x_i in a degree-two monomial in the system of equations obtained after fixing the $j-1$ first most appearing variables (as previously described) w.l.o.g. $x_1, \ldots, x_{j-1}$. Then, choosing the variable with higher N_i^j, w.l.o.g. x_j, the remaining quadratic system of equations consists of $m - N_1^1 - N_2^2 - \cdots - N_j^j$ equations in $n - j$ unknowns. So for all $1 \leq j \leq n$:

$$N_j^j \geq 2\frac{m - N_1^1 - N_2^2 - \cdots - N_{j-1}^{j-1}}{n - j + 1} \geq 2\frac{m}{n} ,$$

and we get $N_1^1 + N_2^2 + \cdots + N_j^j + j$ linear equations at this step with the value of $x_1, x_2, \ldots, x_j$ being fixed.

Take ℓ as the first value of j such that $N_1^1 + N_2^2 + \cdots + N_j^j + j \geq n + j$ (which is correctly defined as we only consider cases where $2m \geq n$). Then,

$$N_1^1 + N_2^2 + \cdots + N_{\ell-1}^{\ell-1} + \ell - 1 < n + \ell - 1 .$$

As for all $1 \leq j \leq \ell$, we have $N_j^j \geq 2\frac{m}{2}$ we get

$$2(\ell - 1)\frac{m}{n} < n .$$

So, the number of variables to guess ℓ is at most:

$$\left\lfloor \frac{n^2}{2m} + 1 \right\rfloor .$$

Note that since we consider the regime of superlinear stretch ($m = n^{\mathsf{s}}$ with $\mathsf{s} > 1$), the above implies that $\ell = o(n)$ (in fact, $\ell = O(n^{2-\mathsf{s}})$). □

We show further in Section 4.3.6 that experimental results are much better. It is worth noticing that the value obtained at Proposition 4.1 is the extreme case for the attacker and does not reflect the average case. However, this frequency of appearance is linked to a well-known problem of combinatorics in the context of balls-into-bins. At the second order, the maximum

load (i.e. the number of occurrences of the variable that appears the most) follows:

$$\Theta\left(\sqrt{\frac{m\ln n}{n}}+\frac{m}{n}\right),$$

where m corresponds to the number of balls and n to the number of bins (e.g. [JK77, KSC78]), which means we gain nothing asymptotically in average. This is related to us, but is not exactly the same, as in one monomial, one variable cannot be taken twice. However, we can lower the maximum that one variable appears with the classical setting of balls and bins by only considering the first variable, but also upper bound our exact probability distribution using twice the maximum load. Eventually, we can say that in average, the number of guesses is asymptotically the same as the worst case for the attacker.

Cost of the Selection Phase

The lemma below follows immediately:

Lemma 4.1. *The selection phase has complexity $O(\ell \cdot m)$ which is $O(n^2)$ with Equation 4.1 estimation.*

Cost of the Guessing Phase

For each $i \in \{1, \ldots, 2^\ell\}$, the algorithm executes either the procedure of Case 1 or the procedure of Case 2; finding out which case to execute requires computing the rank of an $s \times n$ matrix, with $s \approx n + \ell$. The cost of Case 1 is dominated by the inversion of an $n \times n$ matrix (since this cost is at least n^2, it dominates the cost of evaluating PRG_{G,P_5}, which is $O(m)$); the cost of Case 2 is dominated by the Gaussian elimination step. Observe that by construction, the matrix A_i (hence the submatrix C_i as well) is very sparse: each of its rows contains at most four nonzero entries. Therefore, we can apply Wiedemann algorithm [Wie86] and compute the rank of A_i, the inverse of C_i, or the row echelon form of A_i, in time $O(n \cdot (n+\ell)) = O(n^2)$ (since they can all be computed by making a constant number of black-box calls to an algorithm solving a sparse system of linear equations).

Combining the above calculations, the cost of the entire algorithm is dominated by

$$O(n^2 \cdot 2^\ell) = 2^{O(n^{2-\mathsf{s}})}.$$

Lemma 4.2. *The asymptotic complexity of the attack is*

$$O\left(n^2 2^{\frac{n^{2-s}}{2}}\right).$$

4.3.3 Success Probability

We now analyze the success probability of the algorithm. Let us first assume that y is in the image of the PRG: there exists x such that $y = \mathsf{PRG}_{G,P_5}(x)$. In this case, during the guessing phase, since the algorithm enumerates over all possible values for the ℓ selected variables, there must be an index i such that the selected variables have been assigned the correct value. Let i^* denote this index.

- If $\mathsf{rank}(A_{i^*}) = n$ (Case 1), the algorithm exactly recovers the right seed x by inverting the $n \times n$ subsystem, hence the check that $\mathsf{PRG}_{G,P_5}(x'_i) = y$ necessarily passes, hence the algorithm outputs 1 and halts with probability 1.

- If $\mathsf{rank}(A_{i^*}) < n$ (Case 2), observe that by construction, the last $\ell + 1$ rows of B_{i^*} are identically zero (since the number of zero rows at the end of the row echelon form of the matrix A_{i^*} is equal to the co-rank of A_{i^*}, which is at least $\ell + 1$). By assumption y is in the image of the PRG and i^* is the right guess, hence we have

$$G_{i^*}y' = G_{i^*}(A_{i^*}x) = B_{i^*}x,$$

  which implies that $G_{i^*}y'$ ends with at least $\ell+1$ zeroes (since the last $\ell+1$ rows of B_{i^*} are identically zero). Therefore, all checks of the algorithm necessarily pass, and it outputs 1 and halts with probability 1.

Hence, if y is in the image of the PRG, the algorithm always outputs 1. Let us now assume that y is a uniformly random m-bit string. Let us fix an arbitrary i between 1 and 2^ℓ. We analyze the probability that the algorithm outputs 1 on this i, where the probability is over the uniformly random choice of y. As previously, two cases can happen.

- If $\mathsf{rank}(A_i) = n$ (Case 1), the algorithm extracts a candidate seed x'_i. Note that this extraction is entirely independent of the choice of y. There are 2^n possible values of x'_i, hence 2^n possible values of $\mathsf{PRG}_{G,P_5}(x'_i)$. The probability (over a random choice of the m-bit string y) that y hits one of those values is equal to $2^n/2^m = 1/2^{m-n}$. Hence, the probability that the algorithm outputs 1 at step i, conditioned on case 1 happening, is upper bounded by $1/2^{m-n}$.

- If $\mathsf{rank}(A_i) < n$ (Case 2), the algorithm obtains $\ell+1$ vectors $(v_1, \ldots, v_{\ell+1})$. Note that since the v_i are rows of G_i, and G_i is invertible, the v_i are all linearly independent. Now, the probability that a uniformly random bit-vector y passes $\ell + 1$ linearly independent linear tests is at most $1/2^{\ell+1}$; therefore, the probability that the algorithm outputs 1 at step i, conditioned on case 2 happening, is upper bounded by $1/2^{\ell+1}$.

Since $\ell + 1 = O(n^2/m) = o(m - n)$, for a sufficiently large n we have

$$\frac{1}{2^{\ell+1}} > \frac{1}{2^{m-n}},$$

from which we get that for each i, the probability (over a random choice of y) that the algorithm outputs 1 is at most $1/2^{\ell+1}$. Taking a union bound over all possible choices of i, we get that the probability that there exists an index i for which the algorithm outputs 1 is at most $2^\ell \cdot 1/2^{\ell+1} = 1/2$. Hence, with probability at least $1/2$, the algorithm outputs 0.

Overall, the algorithm correctly distinguishes between $y = \mathsf{PRG}_{G,P_5}(x)$ and random y with probability at least $1/2(1 + 1/2) = 3/4$. Note that the success probability of the adversary can be made as close to 1 as one wishes, by collecting $n + \ell + \lambda - 1$ linear equations instead of $n + \ell$, for a security parameter λ; it is easy to check that this does not change the asymptotic complexity of the algorithm, and by the same analysis, the algorithm correctly outputs 0 when y is random with overwhelming probability at least $1 - 1/2^\lambda$.

4.3.4 Seed Recovery

The attack which we described above is a distinguishing attack: it breaks the pseudorandomness of the PRG in subexponential time $2^{O(n^{2-\varepsilon})}$. Observe that when $y = \mathsf{PRG}_{G,P_5}(x)$ for some x, if Case 1 happens at the step i^* corresponding to the right guess, the attack gives something stronger: it actually breaks the one-wayness of the PRG, by recovering the seed. Furthermore, our experimental evaluations (which we will discuss in Section 4.3.6) show that this is actually always the case: the algorithm systematically ends up in Case 1, and Case 2 never happens, leading to a seed recovery attack. In this section, we provide some theoretical support for this observation:

- we put forth a combinatorial assumption and prove that, under this assumption, there is a seed recovery algorithm which is a slight variation of our algorithm (and has the same complexity);
- we provide heuristic support for our combinatorial conjecture by relating it to existing results in mathematics.

Combinatorial Conjecture

We consider the following conjecture: set $\beta \leftarrow \lfloor n^2/2m + 1 \rfloor$, and define, for $i = 1$ to 2^β, $\mathcal{D}_{n,i}$ to be the distribution over $\mathbb{F}_2^{m \times n}$ obtained by sampling the hypergraph of Goldreich's PRG at random (with $d = 5$), selecting ℓ variables that appear in $n + \ell$ quadratic equations using the selection phase algorithm (see Section 4.3.1), and outputting the $n \times n$ matrix M_n of the linear

system obtained by setting all ℓ selected variables to the values indicated by the ℓ first bits of i (note that our analysis guarantees that $\ell \leq \beta$). We truncate to n equations for simplicity.

Hypothesis 4.1. *There exists a constant γ such that for every sufficiently large $n \in \mathbb{N}$, for every $i \leq 2^{\beta}$, the matrix M_i contains with overwhelming probability an invertible subsystem of $\gamma \cdot n$ equations, where the probability is taken over the coins of $M_i \xleftarrow{\$} \mathcal{D}_{n,i}$.*

Note that the conjecture is tailored to our particular attack, and could be easily generalized to more general PRG distributions and variable selection methods – indeed, we do consider generalizations and variants of this conjecture in the following sections. We first show that if Hypothesis 4.1 is verified, then there is a seed recovery attack on Goldreich's PRG instantiated with P_5. The attack is a simple variation of our previous algorithm, where in the guessing phase we do not consider case 2. Instead, the algorithm extracts a $\gamma n \times \gamma n$ invertible submatrix C_i of A_i (whose existence is guaranteed by Hypothesis 4.1), and uses it to recover a subsequence of γn bits of the seed x. Now, by applying the result of Bogdanov and Qiao [BQ09] on recovering a preimage from an approximate preimage of Goldreich's PRG, there exists a black-box polynomial-time reduction from an algorithm that recovers (with no error) $O(n^{(7-\mathsf{s})/8}) \ll \gamma \cdot n$ bits of the seed to an algorithm that fully recovers the seed.

Supporting the Conjecture

Unfortunately, the distributions $\mathcal{D}_{n,i}$ are quite complex, and it seems relatively difficult (and outside the scope of this work) to prove our conjecture. However, we can provide some heuristic support for the conjecture: variants of our conjecture with respect to simpler (and natural) distributions (which are close to the one we consider) follow from existing results in mathematics and computer science. Note that the $\mathcal{D}_{n,i}$ are distributions of random very sparse matrices, with at most 4 nonzero entries per row. We can consider two simpler natural distribution over very sparse matrices:

- the distribution D obtained by setting each entry of the matrix to be 1 with probability $4/n$, and 0 with probability $(n-4)/n$ (the Bernouilli distribution);
- the distribution D' obtained by sampling 4 random positions between 1 and n in each row, setting the entries at these positions to be 1, and setting all other entries of the row to be 0.

For the distribution D, simply looking at the entries that contain exactly a single 1 will give with high probability a $\gamma n \times \gamma n$ invertible submatrix (indeed, a permutation matrix),with $\gamma \approx 5 \cdot e^{-5}$. This gives a very loose lower bound on γ, but in fact, much stronger bounds are known for this distribution, at least in the case of random sparse *symmetric* matrices [BL10].

For the distribution D', the conjecture is very close to problems which have been studied in computer science under the name of Random XOR-SAT. In particular, the recent work of [PS16] gave a precise threshold value of c such that a random $c \cdot n \times n$ matrix contains an $n \times n$ invertible matrix with probability 1; this result implies in particular a (loose) lower bound of $\gamma = 1/c$ for our conjecture.

4.3.5 Concrete Instantiation of the Attack

We formulated our attack in an asymptotic sense, to obtain provable asymptotic efficiency guarantees. However, it is possible to obtain a much better concrete efficiency than the one achieved by our algorithm. A first observation is that even before the selection phase, we can collect several linear equations "for free" by looking at all quadratic equations where the quadratic terms are equal, and XORing them to cancel out the quadratic terms.

Finding All Collisions

We first define the notion of collisions between two quadratic equations.

Definition 4.6. *A collision is a couple* $(i, j) \in [m]^2$ *such that* $i \neq j$ *and* $\{\sigma_4^i, \sigma_5^i\} = \{\sigma_4^j, \sigma_5^j\}$.

Observe that any collision leads to a linear equation "for free": XORing the quadratic equations indexed by σ^i and σ^j, the terms $x_{\sigma_4^i} \cdot x_{\sigma_5^i}$ and $x_{\sigma_4^j} \cdot x_{\sigma_5^j}$ cancel out, leading to a linear equation. The algorithm first finds all collisions, and derives the corresponding linear equations. Let c be the number of linear equations obtained with this step. While the asymptotic number of such collisions is small, hence it does not change the asymptotic complexity, it turns out that this simple step already strongly reduces the concrete cost of the attack. Let c denote the number of linear equations obtained this way.

Note that finding all collisions can be reached with a tweaked sorting algorithm. The idea is to sort the equations $(E_i)_{1 \leq i \leq m}$ according to an order[8] on the quadratic term $x_{\sigma_4^i} x_{\sigma_5^i}$. And, each time an equality between two quadratic terms is found, one equation is removed and a new linear equation $E_i + E_j$ is derived. The complexity is dominated by the sorting complexity $O(m \cdot \log(m))$.

Avoiding the Bogdanov and Qiao Algorithm

Furthermore, as we already mentioned, we observe experimentally that Case 2 never happens. In all our experiments, the algorithm always ends up in Case 1, with a value of $\gamma > 0.90$. Note also that applying the result of Bogdanov and Qiao to obtain the seed from the approximate

[8] The order does not matter since only equalities are necessary, one can take the lexicographic order for example.

preimage is an overkill: this result actually only requires knowing an approximate preimage (but not necessarily which of the bits of the preimage are correct), while our attack gives us also the exact position of the correct bits of the preimage. Therefore, we can simply inject directly these $\gamma n > 0.90n$ values in our list of quadratic equations, which will turn a large fraction of them into linear equations, and hope to obtain the missing values directly from these linear equations. Our experiments show that this is indeed the case: after recovering a large fraction of the preimage, injecting the values in the quadratic equations always allows to recover the full seed. Our experiments show that this is the case with a large confidence gap: injecting only a small fraction $\gamma > 0.20$ of the preimage in the quadratic equations is sufficient to always recover the full seed.

Collecting Less Equations

Lastly, since Case 2 never happens, we do not need to collect $n + \ell$ linear equations: we can stop as soon as we collect $n - c$ linear equations in the guessing phase (leading to a total of n linear equations when adding the equations obtained through collisions – note that we were already truncating the matrices A_i and ignoring the last ℓ equations when formulating Hypothesis 4.1).

Assessing the Number of Collisions

For completeness, we analyze the asymptotic number of equations obtained through collisions. As previously noticed, collisions can be used to build linear equations. For example, let us assume we have the following two equations in Σ:

$$x_{\sigma_1^i} + x_{\sigma_2^i} + x_{\sigma_3^i} + x_{\sigma_4^i} x_{\sigma_5^i} = y_i \tag{E_1}$$

$$x_{\sigma_1^j} + x_{\sigma_2^j} + x_{\sigma_3^j} + x_{\sigma_4^i} x_{\sigma_5^i} = y_j \tag{E_2}$$

then adding Equation (E_1) and Equation (E_2) gives us the following linear equation:

$$x_{\sigma_1^i} + x_{\sigma_2^i} + x_{\sigma_3^i} + x_{\sigma_1^j} + x_{\sigma_2^j} + x_{\sigma_3^j} = y_i + y_j$$

However, we stress that if we had a third colliding equation:

$$x_{\sigma_1^k} + x_{\sigma_2^k} + x_{\sigma_3^k} + x_{\sigma_4^i} x_{\sigma_5^i} = y_k \tag{E_3}$$

then we could only produce a single other linear equation (w.l.o.g. (E_1) + (E_3)), since the other combination ((E_2) + (E_3)) would be linearly equivalent to the two previous linear equations.

Hence, this problem can be seen as a balls-into-bins problem: m balls are randomly thrown into $\binom{n}{2}$ bins and we want to know how many balls in average hit a bin that already contains at least one ball. Indeed, this number will approximate the value c of the algorithm.

Proposition 4.2 (Average number of collisions)**.** *Let n be the number of variables, and m be the number of equations, let C be the random variable counting the number of collisions on the degree-two monomials in the whole system. Then, the average number of collisions is:*

$$\mathbb{E}(C) = m - \binom{n}{2} + \binom{n}{2}\left(\frac{\binom{n}{2}-1}{\binom{n}{2}}\right)^m \in O(n^{2(s-1)}) \ .$$

Proof. We first consider individually the $\binom{n}{2}$ degree-two possible monomials. For each equation, the two variables of the degree-two monomial are taken uniformly from the n variables (with replacement), therefore the probability that the monomial indexed by i, j is taken follows a Bernouilli law with parameter $p = \frac{1}{\binom{n}{2}}$.

The random variable counting how many times the monomial indexed by i, j is selected follows a binomial law of parameters m and p. As a collision happens when the monomial has already been taken, we consider the random variable $C_{i,j}$ counting 0 if the monomial has been taken 0 or 1 times, $k-1$ otherwise. The expectation of $C_{i,j}$ is therefore

$$\mathbb{E}(C_{i,j}) = \sum_{k=2}^{m} P_{[B(m,p)=k]} \cdot (k-1),$$

where $P_{[B(m,p)=k]}$ stands for the probability for a random variable following a binomial distribution of parameters m and p to take the value k. The total number of collisions is obtained by summing the expectations of all the $C_{i,j}$.

$$
\begin{aligned}
\mathbb{E}(C) &= \sum_{i=1}^{n}\sum_{j=i+1}^{n}\mathbb{E}(C_{i,j}) = \sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=2}^{m} P_{[B(m,p)=k]}\cdot(k-1) \\
&= \binom{n}{2}\sum_{k=2}^{m} P_{[B(m,p)=k]}\cdot(k-1) = \binom{n}{2}\sum_{k=2}^{m}\binom{m}{k}p^k(1-p)^{m-k}\cdot(k-1) \\
&= \left[\binom{n}{2}\sum_{k=0}^{m}\binom{m}{k}p^k(1-p)^{m-k}\cdot(k-1)\right] - \left[\binom{n}{2}\sum_{k=0}^{1}\binom{m}{k}p^k(1-p)^{m-k}\cdot(k-1)\right] \\
&= \left[\binom{n}{2}\sum_{k=0}^{m}\binom{m}{k}p^k(1-p)^{m-k}\cdot(k-1)\right] + \binom{n}{2}p^0(1-p)^m \\
&= \left[\binom{n}{2}\sum_{k=0}^{m}k\binom{m}{k}p^k(1-p)^{m-k}\right] - \binom{n}{2} + \binom{n}{2}(1-p)^m \\
&= \left[\sum_{k=0}^{m}k\binom{m}{k}\left(\frac{1}{\binom{n}{2}}\right)^{k-1}\left(\frac{\binom{n}{2}-1}{\binom{n}{2}}\right)^{m-k}\right] - \binom{n}{2} + \binom{n}{2}(1-p)^m \text{ since } p=\frac{1}{\binom{n}{2}} \\
&= \left[\sum_{k'=-1}^{m-1}(k'+1)\binom{m}{k'+1}\left(\frac{1}{\binom{n}{2}}\right)^{k'}\left(\frac{\binom{n}{2}-1}{\binom{n}{2}}\right)^{m-1-k'}\right] - \binom{n}{2} + \binom{n}{2}(1-p)^m \\
&= \left[\sum_{k'=0}^{m-1}m\binom{m-1}{k'}\left(\frac{1}{\binom{n}{2}}\right)^{k'}\left(\frac{\binom{n}{2}-1}{\binom{n}{2}}\right)^{m-1-k'}\right] - \binom{n}{2} + \binom{n}{2}\left(\frac{\binom{n}{2}-1}{\binom{n}{2}}\right)^{m} \\
&= m - \binom{n}{2} + \binom{n}{2}\left(\frac{\binom{n}{2}-1}{\binom{n}{2}}\right)^{m}.
\end{aligned}
$$

Eventually this number can be estimated with a limited development:

$$
\begin{aligned}
\mathbb{E}(C) &= n^{\mathsf{s}} - \binom{n}{2} + \binom{n}{2}e^{ln\left(1-\frac{1}{\binom{n}{2}}\right)n^{\mathsf{s}}} = n^{\mathsf{s}} - \binom{n}{2} + \binom{n}{2}e^{\left(-\frac{1}{\binom{n}{2}}-\frac{1}{2\binom{n}{2}^2}+o\left[\frac{1}{\binom{n}{2}^3}\right]\right)n^{\mathsf{s}}} \\
&= n^{\mathsf{s}} - \binom{n}{2} + \binom{n}{2}e^{\left(-\frac{n^{1+\mathsf{s}}}{\binom{n}{2}}-\frac{n^{1+\mathsf{s}}}{2\binom{n}{2}^2}+o\left[\frac{n^{\mathsf{s}}}{\binom{n}{2}^3}\right]\right)} \\
&= n^{\mathsf{s}} - \binom{n}{2} + \binom{n}{2}\left(1-\frac{n^{\mathsf{s}}}{\binom{n}{2}}-\frac{n^{\mathsf{s}}}{2\binom{n}{2}^2}+o\left[\frac{n^{\mathsf{s}}}{\binom{n}{2}^3}\right]+\frac{n^{2\mathsf{s}}}{2\binom{n}{2}^2}+o\left[\frac{n^{2\mathsf{s}}}{\binom{n}{2}^2}\right]\right) \\
&= n^{\mathsf{s}} - \binom{n}{2} + \binom{n}{2} - n^{\mathsf{s}} - \frac{\binom{n}{2}n^{\mathsf{s}}}{2\binom{n}{2}^2} + \frac{\binom{n}{2}n^{2\mathsf{s}}}{2\binom{n}{2}^2} + o\left[\frac{\binom{n}{2}n^{2\mathsf{s}}}{\binom{n}{2}^2}\right] \\
&= -\frac{n^{\mathsf{s}}}{2\binom{n}{2}} + \frac{n^{2\mathsf{s}}}{2\binom{n}{2}} + o\left[\frac{n^{2\mathsf{s}}}{2\binom{n}{2}}\right] = O(n^{2(\mathsf{s}-1)}).
\end{aligned}
$$

□

Tab. 4.1 gives the evaluation of this formula for some set of parameters. Our experimental results (see Section 4.3.6) corroborate these expectations and show that the number of collisions is always very close to this expected average.

4.3.6 Experiments

Distribution of the number of collisions

The theoretical results of Table 4.1 are verified in practice, as shown in Fig. 4.1 for the particular case of $n = 1024$ and s $= 1.4$. As expected with the analytical formula, the number of collisions is very close to 254 in average. Moreover, our experimental results are very dense around the average, suggesting that the distribution has a low variance.

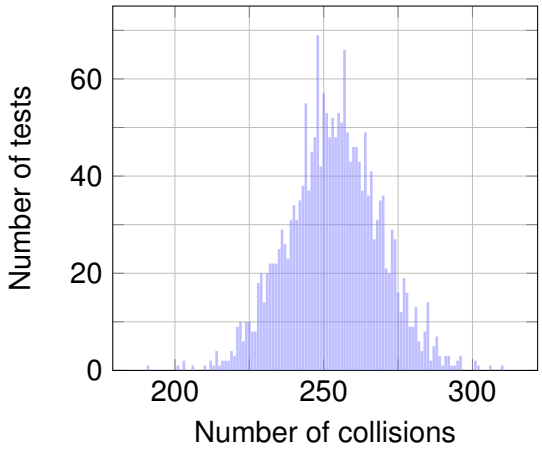


Figure 4.1: Number of collisions for $n = 1024$ and s $= 1.4$ with 2000 tests

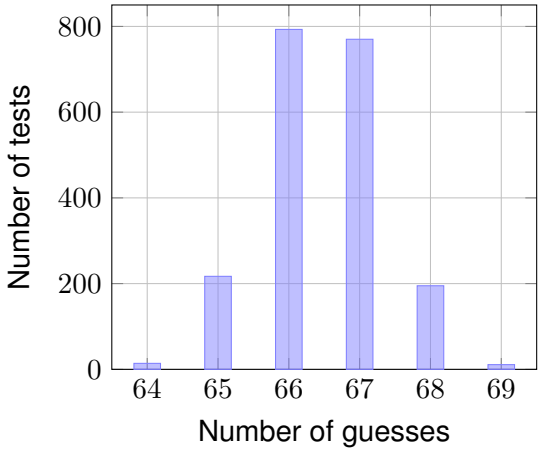


Figure 4.2: Number of guesses for $n = 2048$ and s $= 1.3$ with 2000 tests

Implementation of the attack

Since the original motivation of this work is to study the concrete security of Goldreich's PRG, it is important to practically check if the attack presented in Section 4.3.4 can be efficient when implemented. For this purpose, we provide a proof of concept in Python[9].

We first analyzed experimentally Hypothesis 4.1 and observed that we always obtain an invertible subsystem of at least $0.90 \cdot n$ equations, for all tested parameters ($2^8 \leq n \leq 2^{14}$ and $1 <$ s < 1.5). We also experimented that knowing only 20% of the seed allows to inject it in the quadratic system and to recover the remaining 80%, showing a large gap of confidence in our hypothesis.

[9]Our proof of concept can be found at `https://github.com/LuMopY/SecurityGoldreichPRG`

One can note that the practical attack should be on average more efficient than assessed theoretically. Indeed, the asymptotic complexity of Proposition 4.2 is estimated in the worst case and pessimistic approximations were made on $n - c$ and on the value of ℓ. Hence, we experimented this attack for different stretches and different values of n and we effectively noticed that the complexity on average is much smaller than the expected complexity. Table 4.2 represents the theoretical number of guesses necessary to recover the seed and Table 4.3 represents the average number of guesses actually needed in the experiment. Moreover, we also noticed that the number of guesses needed to invert the system has a very low variance, as shown in Fig. 4.2.

Table 4.1: Theoretical number of collisions (average case)

n	256	512	1024	2048	4096
$\mathsf{s} = 1.45$	142	269	506	946	1771
$\mathsf{s} = 1.4$	83	145	254	442	773
$\mathsf{s} = 1.3$	28	42	64	97	147

Table 4.2: Theoretical number of guesses (worst case)

n	256	512	1024	2048	4096
$\mathsf{s} = 1.45$	4	7	11	18	27
$\mathsf{s} = 1.4$	9	15	23	37	58
$\mathsf{s} = 1.3$	20	34	56	94	156

Table 4.3: Experimental number of guesses (average case)

n	256	512	1024	2048	4096
$\mathsf{s} = 1.45$	4	6	9	14	21
$\mathsf{s} = 1.4$	6	11	17	27	44
$\mathsf{s} = 1.3$	13	23	39	66	110

Table 4.4: Challenge parameters for seed recovery attacks. The first line contains the parameter n and below are represented the associated stretches s.

Operations	512	1024	2048	4096
$< 2^{80}$	1.120	1.215	1.296	1.361
$< 2^{128}$	1.048	1.135	1.222	1.295

This experiment enables to estimate the practical security of Goldreich's PRG against the guess and determine approach with 80 bits of security. Indeed, for one instance of the PRG, the complexity of the seed recovery can be easily derived from the number ℓ of guesses as $2^{\ell} n^{\omega}$. So to assess the 80 bits security, one can evaluate the average number of guesses necessary for one choice of (n, s) and check if the complexity is lower than 2^{80}. For that, for 30 values of $n \in [2^7, 2^{14}]$, we delimited the smallest stretch for which the average number of guesses allows a 80 bits attack. Each average has been done on 1000 measurements because the variance was very small. Fig. 4.3 represents the limit on vulnerable (n, s) parameters. Above the line, the parameters are on average insecure against the guess and determine attack.

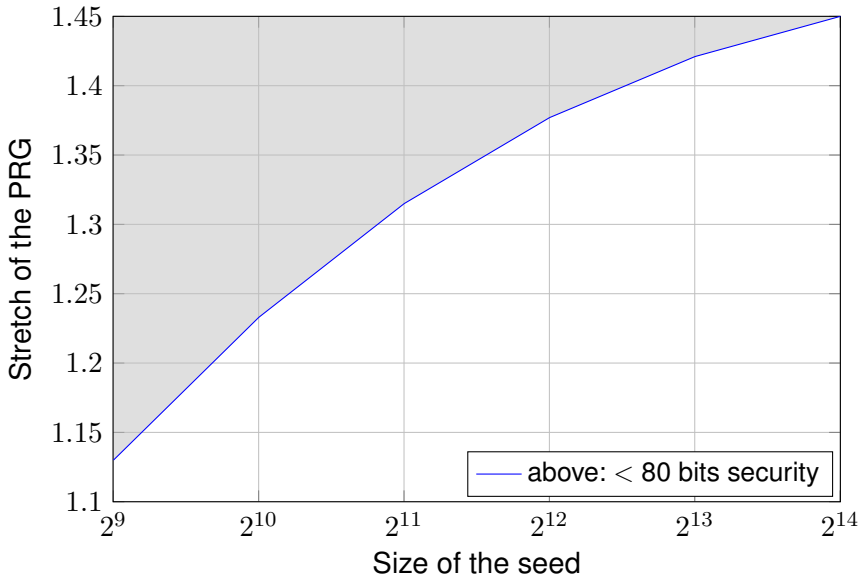


Figure 4.3: Limit stretch for vulnerable instances. The grey zone above the curve denotes the insecure choices of parameters.

Candidate Non-Vulnerable Parameters

We were able to estimate the practical range of parameters that appear to resist this attack. To assess them, we estimated the number of guesses necessary and deduced the bit security. With many measurements (1024 for each set of parameters), we could find the limit stretch for parameters that are, not vulnerable to our attack. The couples (n, s) that possess the maximal s with an expected security of 80 or 128 bits[10] are conjectured to be the limit for non-vulnerable parameters. These couples[11] are represented by the two lines in Fig. 4.4.

We also introduce certain parameters in Table 4.4 as challenges for improving the cryptanalysis of Goldreich's PRG. These parameters correspond to choices of the seed size and the stretch which cannot be broken in less than 2^{80} (resp. 2^{128}) operations with the attacks described in this chapter. Further study is required to assess confidence in the security level given by these parameters.

4.4 Algebraic Cryptanalysis of Goldreich's PRG with P_5

To complement the attacks of Section 4.3.1, we also provide an analysis of the efficiency of algebraic attacks with Gröbner basis on Goldreich's PRG. While it is known that Goldreich's

[10] We actually took a margin of 10% to take into account the possible improvements of our implementation.

[11] This curve should not be extrapolated because outside of its range, Gröbner attacks seem more powerful, see Fig. 4.10

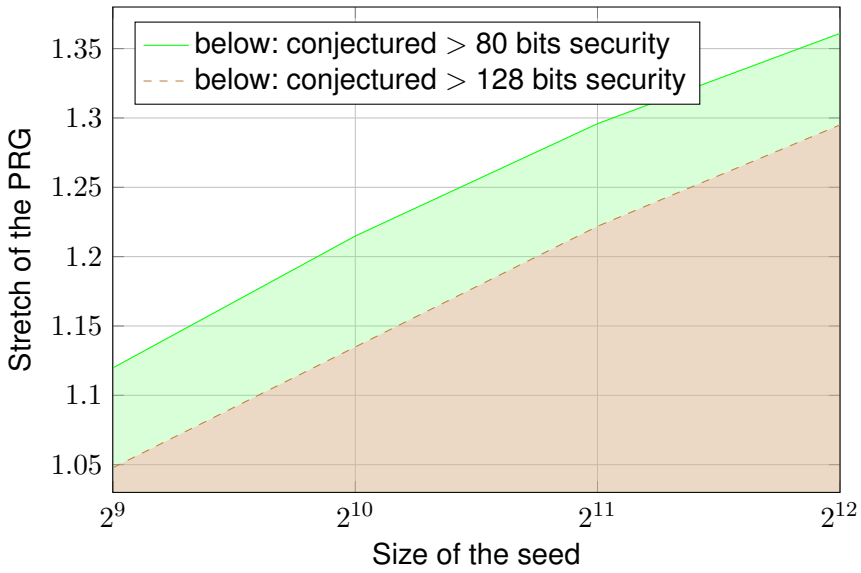


Figure 4.4: Limit stretch for conjectured non-vulnerable instances.

PRG (and its variants) provably resists such attacks for appropriate choices of (asymptotic) parameters ([AL16], Theorem 5.5), little is known about its exact security against such attacks for concrete choices of parameters.

In this section, we study the existence of polynomial attacks for s < 1.5. In fact, with the current literature, either s ≥ 1.5 and there is a polynomial inversion, or s < 1.5 and the only known attack is subexponential. The idea of this section is to offer some granularity on the parameters (n, s) instead of this abrupt limit for polynomial inversion. For this, we opt for a different algebraic approach without guess and determine. Instead of guessing values to transform the public system into a linear system in the seed, one might want to generate enough equations in order to linearize. This standard method has been introduced by Macaulay in [Mac64] and Lazard in [laz81]. The idea behind linearization is the assignment of an unknown variable for each of the monomials appearing in the system. For example, to each monomial $x_i x_j$, a variable $X_{i,j}$ will be assigned. Thereby, a linear system of equations with more unknowns of type $X_{i,j}$ remains to be solved. This linearization method has been improved in Gröbner basis computations due to Buchberger [Buc76] and later by Faugère with F4 [Fau99] and F5 [Fau02] algorithms.

Performance of a Gröbner basis strategy is hard to assess for the specific case of Goldreich's PRG with the existing theory (see [BFSyY] for complexity bounds on Boolean random quadratic systems). Indeed, Goldreich's PRG is far from a Boolean random quadratic system, it has a strong structure and is very sparse. These features should make Goldreich's PRG an easier target. In a first step, in order to give an intuition on how Gröbner basis algorithms would behave on Goldreich's PRG with predicate P_5, we provide an easy-to-understand degree-two

linearization attack . This polynomial attack leads to a practical seed recovery for certain parameters (n, s) and we can derive a heuristic bound for vulnerable (n, s) for 80 bits of security[12]. The existence of such an attack allows to estimate Gröbner basis algorithm complexity. Indeed, Gröbner basis algorithms use an optimized method to generate polynomials. So, their performance is at least as good as our linearization attack. Thus, from our linearization attack performance and complexity, we derive a heuristic bound on vulnerable (n, s) parameters against a Gröbner basis technique. This heuristic bound shows that a Gröbner basis approach may attack more parameters than the guess-and-determine technique (of Section 4.3) for high values of n.

4.4.1 A Polynomial Attack with Degree-Two Linearization

For a degree-two linearization, the number of variables will highly increase in comparison to the Section 4.3 case. Indeed, the total variables will include linear terms of shape x_i and quadratic terms of shape $x_i x_j$ where $i \neq j$. Thus, the total number of variables is

$$\mathcal{N}_{var}(n) = n + \binom{n}{2}.$$

To get a chance to invert a system with so many linearized variables, one needs to generate as many quadratic equations as possible. Fortunately, Goldreich's PRG with P_5 predicate has such a structure that allows any attacker to create a large number of new equations from the original system. Before showing how to generate these equations, let us introduce the principle of the attack assuming that a certain number of equations is drawn.

An Attack and its Complexity

Suppose that a Goldreich's PRG is drawn with parameters (n, s) and with c collisions. Suppose also that one can create a set of quadratic equations that contains $\mathcal{N}_{indep\ eqns}$ linearly independent ones. Only equations of degree exactly two are counted in $\mathcal{N}_{indep\ eqns}$. We sketch a seed recovery attack assuming that

$$0 \leq \mathcal{N}_{var}(n) - \mathcal{N}_{indep\ eqns} \leq c$$

and assess its complexity.

Step 1 From the system of $\mathcal{N}_{indep\ eqns}$, we create a linear system in matrix form.

[12]The case of 128 bits of security is harder to assess because a degree-three linearization must then be considered. This study is left for future work.

Step 2 We rewrite this system by separating the quadratic part and creating submatrices. Let q_i be the quadratic part of this new system and b_i be its linear part and y_i be its constant term.

$$\begin{aligned} q_1 + b_1 &= y_1 \\ &\vdots \\ q_{\mathcal{N}_{indep\ eqns}} + b_{\mathcal{N}_{indep\ eqns}} &= y_{\mathcal{N}_{indep\ eqns}} \end{aligned}$$

The linearization consists in solving $(q_i + b_i = y_i)_{i \in [\mathcal{N}_{indep\ eqns}]}$ by replacing each monomial with a variable and trying to invert a linear system of size $\mathcal{N}_{indep\ eqns} \cdot \mathcal{N}_{var}(n)$. We then rewrite the system in terms of matrices. Let $Q \in \mathbb{F}_2^{\mathcal{N}_{indep\ eqns} \cdot \binom{n}{2}}$ represent the coefficients of the quadratic polynomials q_i and $B \in \mathbb{F}_2^{\mathcal{N}_{indep\ eqns} \cdot n}$ represent the coefficients of the linear part b_i. Due to its sparseness, B is full rank with high probability. Figure 4.5 represents such matrices. The grey vector represents the list of quadratic variables of type $(x_i x_j)$, the light-grey vector represents the linear and constant variables.

Step 3 We compute the rank of matrix Q.

- If Q is full rank after deleting its columns of zero, then we invert the system by applying Gaussian elimination on $Q|B$ which is enough to recover the secret seed $x_1, \ldots, x_n$.
- Else Q is not full rank but the rank defect is bounded because of the condition $\mathcal{N}_{var}(n) - \mathcal{N}_{indep\ eqns} \leq c$. Indeed, the previous condition can be reformulated as $\mathcal{N}_{var}(n) > rank(Q|B) \geq \mathcal{N}_{var}(n) - c$. With the addition of the c linear equations obtained by collisions (that are linearly independant with high probability), the whole quadratic system becomes invertible.

  Remark 4.1. *In this precise case, we actually refined the computation in order to gain experimental complexity. For this, we rewrite the system differently as in Figure 4.6. We derive a matrix Λ for left kernel of Q. We multiply the system in Figure 4.6 by Λ and obtain a linear system as in Figure 4.7. With the addition of the c linear equations obtained by collisions, the inversion of the remaining linear system in x_i gives the secret seed with high probability.*

Since the costliest step in this attack is the inversion of a matrix of size $\binom{n}{2}$, the complexity is $O(n^{2 \cdot \omega})$. It then leads to the following proposition.

Proposition 4.3. *Let Goldreich's PRG be instantiated with n, s, and P_5. Let c be the number of collisions and $\mathcal{N}_{indep\ eqns}$ be the number of linearly independent quadratic polynomials gen-*

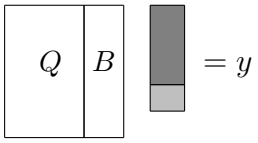


Figure 4.5: Linearized system

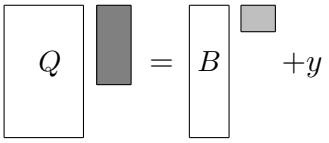


Figure 4.6: Rewritten system

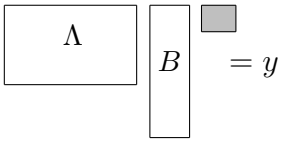


Figure 4.7: Linear system

erated with the previous generation. If $0 \leq \mathcal{N}_{var}(n) - \mathcal{N}_{indep\ eqns} \leq c$ *the previous algorithm recovers the seed with high probability with time complexity* $O(n^{2\cdot\omega})$*.*

Creating and Counting Quadratic Equations

In order to satisfy Proposition 4.3's hypothesis, one must draw $\mathcal{N}_{indep\ eqns}$ linearly independent quadratic equations such that

$$\mathcal{N}_{indep\ eqns} \geq \mathcal{N}_{var}(n) - c\ .$$

In order to achieve it, in the following we introduce a (non exhaustive) list of ways to create new quadratic polynomials. In each case, equations are grouped in a type. We denote by $\mathcal{N}_{T_i}$ the number of equations following from Type i. Unfortunately, predicting the linear dependencies with these new equations is a difficult task for a system with such a structure. For each type, we will remove all redundant equations (also with other types) and assess the number. The linear independence will only be conjectured from experiments.

Let us suppose that an instance of Goldreich's PRG with (n, s) is drawn and gives $m = n^{\mathsf{s}}$ equations $E_1, \ldots, E_m$ evaluated in the secret seed $x_1, \ldots, x_n$ such that for $i \in [m]$,

$$x_{\sigma_1^i} + x_{\sigma_2^i} + x_{\sigma_3^i} + x_{\sigma_4^i} x_{\sigma_5^i} = y_i \qquad (E_i)$$

where $y_1, \ldots, y_m \in \mathbb{F}_2$ is the output.

Type 0: the Original System

The first quadratic equations are the system itself composed of n^{s} quadratic equations. If the system has linear dependencies between equations, then a distinguisher is found and the PRG is broken. We then consider that all equations are linearly independent. All the new quadratic equations will come from this system. To avoid redundancy in the next constructions, we remove one equation from each collision, thus $\mathcal{N}_{T_0} = n^{\mathsf{s}} - c$.

Type 1: Generated Individually

New quadratic polynomials can be derived directly from each equation E_i with $i \in [m]$. Let us fix $i \in [m]$. In the field $\mathbb{F}_2$, the equation $x^2 = x$ gives

$$\begin{aligned} x_{\sigma_1^i} + x_{\sigma_2^i} + x_{\sigma_3^i} + x_{\sigma_4^i} x_{\sigma_5^i} = y_i \rightarrow x_{\sigma_1^i} x_{\sigma_5^i} + x_{\sigma_2^i} x_{\sigma_5^i} + x_{\sigma_3^i} x_{\sigma_5^i} + x_{\sigma_4^i} x_{\sigma_5^i} = y_i x_{\sigma_5^i} \\ \rightarrow x_{\sigma_1^i} x_{\sigma_4^i} + x_{\sigma_2^i} x_{\sigma_4^i} + x_{\sigma_3^i} x_{\sigma_4^i} + x_{\sigma_4^i} x_{\sigma_5^i} = y_i x_{\sigma_4^i} \ . \end{aligned}$$

Thus, the set of quadratic equations generated from E_i is

$$\{zE_i \mid \forall z \in \{x_{\sigma_4^i}, x_{\sigma_5^i}\}\} \ .$$

Then, considering all i in $[m]$, $2 \cdot \mathcal{N}_{T_0} = 2n^{\mathsf{s}} - 2c$ new equations can be created. A linear dependence in these equations would also lead to a distinguisher, then we consider that all these equations are linearly independent, thus $\mathcal{N}_{T_1} = 2n^{\mathsf{s}} - 2c$.

Remark 4.2. *If we combine equations of Type 0 with equations of Type 1, a small number of linear equations can follow. Indeed, take the following example*

$$x_{\sigma_1^i} + x_{\sigma_2^i} + x_{\sigma_3^i} + x_{\sigma_4^i} x_{\sigma_5^i} = y_i \rightarrow x_{\sigma_1^i} x_{\sigma_5^i} + x_{\sigma_2^i} x_{\sigma_5^i} + x_{\sigma_3^i} x_{\sigma_5^i} + x_{\sigma_4^i} x_{\sigma_5^i} = y_i x_{\sigma_5^i} \ .$$

If the quadratic monomials $x_{\sigma_1^i} x_{\sigma_5^i}$, $x_{\sigma_2^i} x_{\sigma_5^i}$ and $x_{\sigma_3^i} x_{\sigma_5^i}$ also appear in Type 0 equations, then each quadratic term can be replaced by the linear part. Thus, a new linear equation of weight up to 13 is created. The expected number of such linear equations is

$$\mathcal{N}_{extra\ lin}(n, \mathsf{s}) = 2 \cdot \mathcal{N}_{T_0} \cdot \left(\frac{\mathcal{N}_{T_0}}{n\binom{n}{2}} \right)^3 \approx 2^4 \cdot n^{4\mathsf{s}-6} \ .$$

This number is low, so these equations are added to the linear equations coming from collisions. In other words, from now on, $c \leftarrow c + \mathcal{N}_{extra\ lin}(n, \mathsf{s}) \approx c$.

Type 2: from Collisions

According to Definition 4.6, a collision is a couple (i, j) such that the sum of E_i and E_j generates a linear equation of shape $x_{\sigma_1^i} + x_{\sigma_2^i} + x_{\sigma_3^i} + x_{\sigma_1^j} + x_{\sigma_2^j} + x_{\sigma_3^j} = y_i + y_j$. Thus, the set of quadratic equations generated from a linear equation L is

$$\{zL \mid \forall z \in \{x_1, \ldots, x_n\}\} \ .$$

Then, $n \cdot c$ quadratic equations can be created. A linear dependence in these equations would lead to a distinguisher with success probability higher than $1/2$, then we consider that all these

equations are linearly independent, thus $\mathcal{N}_{T_2} = n \cdot c$.

Type 3: from Semi-Collisions

Let us first introduce the definition of a semi-collision.

Definition 4.7 (semi-collision)**.** *A semi-collision is a couple* $(i, j) \in [m]^2$ *such that* $i \neq j$*,* (i, j) *is not a collision and such that there exists a* $k \in [n]$ *so that*

$$k \in \{\sigma_4^i, \sigma_5^i\} \text{ and } k \in \{\sigma_4^j, \sigma_5^j\} .$$

For example, the following equations,

$$x_1 + x_2 + x_3 + x_7 x_{10} = y_1 \qquad (E_1)$$
$$x_4 + x_5 + x_6 + x_7 x_8 = y_2 \qquad (E_2)$$

induces $(1, 2)$ as a semi-collision because x_7 appears in both degree-2 monomials.

Lemma 4.3. *When a semi-collision* (i, j) *occurs, an extra quadratic equation of shape*

$$x_{\sigma_{5 \text{ or } 4}^j} E_i + x_{\sigma_{4 \text{ or } 5}^i} E_j$$

can be generated.

In the previous example, it is easy to see that one can generate a new quadratic equation.

$$x_8 x_1 + x_8 x_2 + x_8 x_3 + x_{10} x_4 + x_{10} x_5 + x_{10} x_6 = x_8 \cdot y_1 + x_{10} \cdot y_2 \qquad (x_8 \cdot E_1 + x_{10} \cdot E_2)$$

Lemma 4.4. *The total number of semi-collisions can be approximated by*

$$\mathcal{N}_{semi\ collisions} = n \binom{2n^{-1}(n^{\mathsf{s}} - c)}{2} .$$

Proof. Let p be the probability that a fixed variable x_i appears in the quadratic term of a fixed Type 0 quadratic equation. Thus, $\mathsf{p} = \frac{2}{n}$. For a variable x_i, there are on average $(m - c)\mathsf{p} = 2n^{-1}(n^{\mathsf{s}} - c)$ elements[13] that have x_i in their quadratic term. Inside this set of $2n^{-1}(n^{\mathsf{s}} - c)$ elements, there are $\binom{2n^{-1}(n^{\mathsf{s}}-c)}{2}$ couples. To get all the semi-collisions and collisions, we multiply the previous equation by n. This multiplication is accurate because this counting does not imply simple intersections. □

[13] This is a worst-case approximation.

Removing Redundant Zquations inside Type 3

If naively generated following Lemma 4.4's proof, many equations are redundant. To compute a correct assessment of the significant Type 3 equations, we will remove several redundant equations. Let us study a phenomenon that is at the origin of many redundancies. Look at the following example :

$$x_1 + x_2 + x_3 + x_{10}x_{11} = y_1 \qquad (E_1)$$
$$x_4 + x_5 + x_6 + x_{11}x_{12} = y_2 \qquad (E_2)$$
$$x_7 + x_8 + x_9 + x_{10}x_{12} = y_3 \qquad (E_3)$$

Among the three semi-collisions concerning x_{10}, x_{11} and x_{12}, one is exactly the sum of both other. Then, when a "cycle" of size 3 appears in the quadratic terms, one semi-collision should be ignored. This makes $\mathcal{N}_{T_3}$ smaller than $\mathcal{N}_{semi\ collisions}$. Let $\mathcal{N}_{cycles}$ be the expected number of these "cycles" of size three in a random instance Goldreich's PRG . $\mathcal{N}_{cycles}$ can be approximated by the following:

$$\mathcal{N}_{cycles} \approx \frac{1}{\binom{\binom{n}{2}}{3}} \cdot \binom{n}{3} \cdot \binom{m}{3} \in O(n^{3\mathsf{s}-3}) \ .$$

Then, the remaining number of linearly independent equations is upper bounded by

$$\mathcal{N}_{semi\ collisions} - \mathcal{N}_{cycles} \ .$$

One equation per cycle is removed and all other equations are kept and counted in $\mathcal{N}_{T_3}$.

$$\mathcal{N}_{T_3} = \mathcal{N}_{semi\ collisions} - \mathcal{N}_{cycles}$$

Proposition 4.4. *The total number of linearly independent quadratic equations that can be generated with the previous types of equations is upper-bounded by*

$$\mathcal{N}_{indep\ eqns} \leq \mathcal{N}_{T_0} + \mathcal{N}_{T_1} + \mathcal{N}_{T_2} + \mathcal{N}_{T_3} := \mathcal{N}_{eqn}(n, \mathsf{s}) \in O(n^{2\mathsf{s}-1}) \ .$$

Asymptotically, $\mathsf{s} < 1.5 \implies \mathcal{N}_{eqn}(n, \mathsf{s}) < \mathcal{N}_{var}(n)$ which makes the linearization impossible. This result comes with no surprise since it is part of the asymptotic security assumptions. However, for many instances (when $n < 2^{14}$), $\mathcal{N}_{eqn}(n, \mathsf{s}) \approx \mathcal{N}_{var}(n)$. In the next section, we provide conditions on n and s such that a polynomial seed recovery is possible with non-negligible probability.

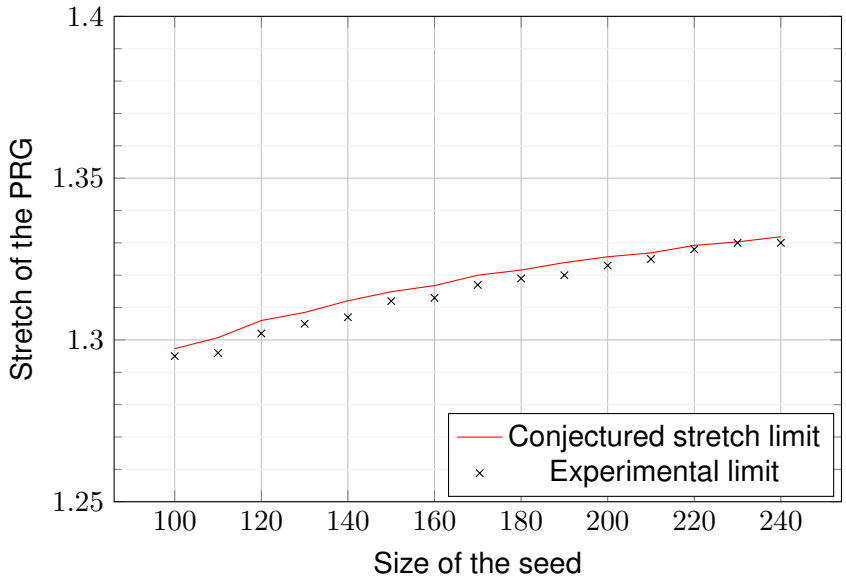


Figure 4.8: Experiment

Conjectured Bound on Vulnerable Parameters

Proposition 4.3 condition ($\mathcal{N}_{var}(n) - \mathcal{N}_{indep\ eqns} \leq c$) does not easily give a bound in terms of parameters. Indeed, $\mathcal{N}_{indep\ eqns}$ is hard to assess because the linear independence of equations form types 0, 1, 2 and 3 is non-trivial to prove.
However, extensive experiments on small parameters support $\mathcal{N}_{eqn}(n, \mathsf{s}) \approx \mathcal{N}_{indep\ eqns}$. That is what allows us to make the following conjectured limit parameters for this polynomial attack:

$$\mathcal{N}_{eqn}(n, \mathsf{s}) > \mathcal{N}_{var}(n) - c \qquad \text{(Heuristical limit)}$$

Experiment

We implemented this attack with a proof of concept using Magma CAS[14]. For each value $n \in \{100, 110, 120, \ldots, 240\}$, we found out that if (n, s) are such that $\mathcal{N}_{eqn}(n, \mathsf{s}) \gg \mathcal{N}_{var}(n) - c$, the attack succeeds with high probability which corroborates the theory. For a given n, we measured the limit stretch s for which the success probability goes under 50%. Indeed, in Fig. 4.8, the dots represent the experiments, the line corresponds to the equality $\mathcal{N}_{eqn}(n, \mathsf{s}) = \mathcal{N}_{var}(n) - c$ (Heuristical limit) which was computed discretely in another Magma code. The estimation of Heuristical limit was a worst-case assessment, so it is not surprising that some experimental limits are actually slightly below the line.

[14]The Magma code can be found at `https://github.com/LuMopY/SecurityGoldreichPRG`

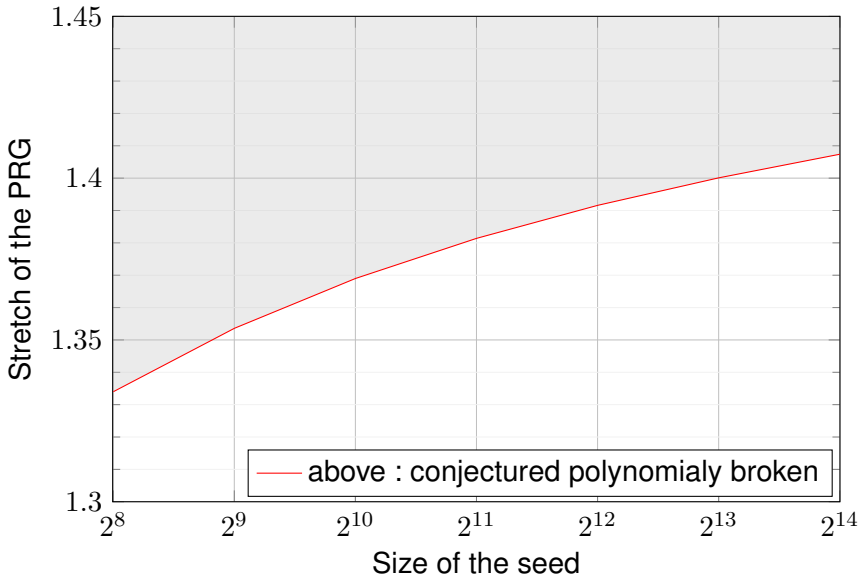


Figure 4.9: Extrapolation graph

Heuristic 4.1 (Extrapolation for greater size of seed)**.** *For any set of parameters* (n, s) *such that Equation Heuristical limit is verified, we conjecture that there is a polynomial seed recovery attack for Goldreich's PRG with* P_5 *with cost* $O(n^{2\omega})$.

We can notice that if $n < 2^{14}$ then the complexity is lower than 2^{80}.

In Fig. 4.9, we represent the extrapolated heuristic bound on (n, s). Above the line, the sets of parameters are conjectured to be vulnerable to this polynomial attack.

4.4.2 Gröbner Approach

An efficient alternative algebraic attack is using Gröbner basis algorithms such as Faugères F4 [Fau99] and F5 [Fau02]. It consists in a succession of linearization attempts where the degree of the linearization is incremented at each step. For each linearization attempt, all polynomial combinations are exhausted in a smart way in order to generate as many new equations as possible. However hard to assess (see Bardet, Faugère, Salvy and Yag's work [BFSyY]), Gröbner basis computation's complexity is dominated by Gaussian elimination on the smallest invertible Macaulay matrix. This Macaulay matrix contains coefficients associated with the monomials of a fixed degree. We denote by *degree of regularity* or D_{reg}, the degree of the monomials associated with the invertible Macaulay matrix. In [BFSyY], under certain hypotheses, the degree of regularity for a random Boolean quadratic system is upper bounded by

$$-n^{\mathsf{s}} + \frac{n}{2} + \frac{n}{2}\sqrt{2n^{2\mathsf{s}-2} - 10n^{\mathsf{s}-1} - 1 + 2(n^{\mathsf{s}-1}+2)\sqrt{n^{\mathsf{s}-1}(n^{\mathsf{s}-1}+2)}}\,.$$

This bound is too generic and does not represent what happens for practical (n, s). Goldreich's PRG structure allows to drastically reduce the degree of regularity. We conjecture an upper bound on the degree of regularity for certain parameters based on Section 4.4.1 attack results and that is observed to be true in our experiments.

Claim 4.1. *If the attack of Section 4.4.1 recovers the seed for one instance of Goldreich's PRG , the degree of regularity D_{reg} is 3 and drops to 2 for the resolution on this instance.*

The performance of Faugères F4 or F5 algorithm on Goldreich's PRG is strictly superior to the attack presented in Section 4.4.1. Indeed, the three types of equations found in Step 1 form a subset of the equations derived from Gröbner basis algorithm up to degree three. Then, if the subsystem is invertible with a degree-two linearization, Gröbner basis algorithm will also be able to invert it with a degree-two linearization. There is a subtlety because when computing the Gröbner basis, the maximal degree of polynomials involved is actually three: for finding semi-collisions, the quadratic polynomials need to be multiplied by a monomial. But then, once enough semi-collisions are found, the Gröbner basis algorithm falls back into solving a degree-two system. This phenomenon is called a *degree fall*.

Experimental Results

To experiment the performance, we used the Gröbner basis algorithms of Magma CAS. The Magma code is then very simple as it consists in computing GroebnerBasis(System,3) which calls a Boolean variant of Faugère F4 algorithm. For each computation, we checked that the degree fall happened and the inversion was done with a degree two. For each value $n \in \{100, 110, \ldots, 240\}$ and the conjectured limit stretch for 50% success, we ran 100 seed recoveries and Gröbner basis algorithm was able to recover around than 90% of the seeds. We finally conclude that according to the conducted experiments, Heuristic 4.1 is observed to be true for small values of n.

Remark 4.3. *We noticed that Gröbner basis performance was able to attack more parameters with lower stretches (often below $\mathsf{s} = 1.25$) with degree of regularity 2. So, some parameters below the heuristic bound may also be vulnerable.*

Increasing the Degree of Regularity

Since we consider 80 bits of security, we want the cost of a degree D_{reg} linearization to be doable with at most 2^{80} operations. A degree D_{reg} linearization corresponds to a Gaussian elimination on a system with $\binom{n}{n-D_{reg}}$ variables. Then, D_{reg} should verify:

$$\binom{n}{n-D_{reg}}^{\omega} < 2^{80} .$$

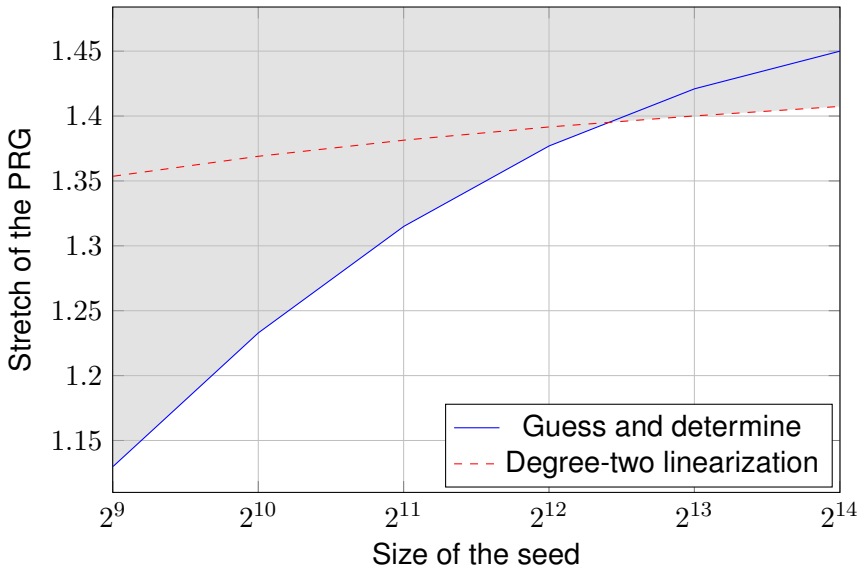


Figure 4.10: Limit stretch for vulnerable parameters with 80 bits of security against both guess and determine (Section 4.3) and degree-two linearization attacks (See Appendix 4.4). The grey zone above the curves denotes the insecure choices of parameters.

This implies that D_{reg} cannot be higher than 2 for $n > 512$. For $n \leq 512$, a degree-three linearization might solve more (n, s) instances. We leave this study as future work.

4.4.3 Conclusion

We described in Section 4.3 a guess-and-determine attack against Goldreich's PRG. In this section, we complement this result with an analysis of the security of Goldreich's PRG against a degree-two linearization attack (*à la* Gröbner). We represent on Figure 4.10 the range of parameters for which Goldreich's PRG is conjectured to have 80 bits of security against those two attacks. As illustrated in the graph, the guess-and-determine approach targets more parameters for low n while the linearization attack performs better for $n > 4000$.

Although Goldreich's PRG is conjectured to be theoretically secure for a stretch approaching 1.5 by an arbitrary constant, our analysis shows that a very large seed must be used to achieve at least 80 bits of security with such a stretch. In particular, if a stretch of 1.4 is needed, no seed smaller than 5120 bits should be used. Similarly, for a stretch as small as 1.1, the seed must be at least 512 bits long.

4.5 About the Ordered Case

In this section, we show that the additional structure given by an ordered Goldreich's PRG with the predicate P_5 brings a lower security level than the unordered case.

4.5.1 Guess and Determine

Although the ordered case seems highly non-trivial to analyze from a theoretical point of view, we give evidence that it brings a lower security level than the unordered case. Then, we also give some experimental measures to support our studies. Each subset is of the form:

$$\sigma^i = [\sigma_1^i, \sigma_2^i, \sigma_3^i, \sigma_4^i, \sigma_5^i], \text{ where } \sigma_1^i < \sigma_2^i < \sigma_3^i < \sigma_4^i < \sigma_5^i \ ,$$

and the equations are of the form:

$$x_{\sigma_1^i} + x_{\sigma_2^i} + x_{\sigma_3^i} + x_{\sigma_4^i} x_{\sigma_5^i} = y_i \ .$$

In this particular case, the average number of collisions is much higher than in the unordered case, since the lasts bits of the seed are drawn with a higher probability.

More formally, the average number of collisions is given by the following proposition:

Proposition 4.5 (Average number of collisions in the ordered case)**.** *Let n be the number of variables, and m be the number of equations, let C be the random variable counting the number of collisions on the degree-two monomials in the whole system. Then, the average number of collisions is:*

$$\mathbb{E}(C) = \sum_{i=1}^{n-1} (n-i) \left(-1 + mp_i + (1-p_i)^m\right) \ ,$$

where $p_i = \frac{\binom{i-1}{3}}{\binom{n}{5}}$.

Proof. We first consider individually the $\binom{n}{2}$ degree-two possible monomials. For each equation, the two variables of the degree-two monomial are taken after the three degree-one monomials, therefore the probability that the monomial indexed by i, j is taken follows a Bernouilli law with parameter $p_i = \frac{\binom{i-1}{3}}{\binom{n}{5}}$.

The random variable counting how many times the monomial indexed by i, j is selected follows a binomial law of parameters m and p_i. As a collision happens when the monomial has already been taken, we consider the random variable $C_{i,j}$ counting 0 if the monomial has been taken 0 or 1 times, $k-1$ otherwise. The expectation of $C_{i,j}$ is therefore:

$$\mathbb{E}(C_{i,j}) = \sum_{k=2}^{m} P_{[B(m,p_i)=k]} \cdot (k-1) \ ,$$

where $P_{[B(m,p_i)=k]}$ stands for the probability for a random variable following a binomial distribution of parameters m and p_i to take the value k.

The total number of collisions is obtained by summing the expectations of all the $C_{i,j}$:

$$\begin{aligned}
\mathbb{E}(C) &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mathbb{E}(C_{i,j}) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=2}^{m} P_{[B(m,p_i)=k]}\cdot(k-1)\\
&= \sum_{i=1}^{n-1}(n-i)\sum_{k=2}^{m}\binom{m}{k}p_i^k(1-p_i)^{m-k}\cdot(k-1)\\
&= \sum_{i=1}^{n-1}(n-i)\left[\left(\sum_{k=0}^{m}\binom{m}{k}p_i^k(1-p_i)^{m-k}\cdot(k-1)\right)+(1-p_i)^m\right]\\
&= \sum_{i=1}^{n-1}(n-i)\left[\left(\sum_{k=0}^{m}k\binom{m}{k}p_i^k(1-p_i)^{m-k}\right)-1+(1-p_i)^m\right]\\
&= \sum_{i=1}^{n-1}(n-i)\left[\left(\sum_{k=0}^{m}m\binom{m-1}{k-1}p_i^k(1-p_i)^{m-k}\right)-1+(1-p_i)^m\right]\\
&= \sum_{i=1}^{n-1}(n-i)\left[\left(m\sum_{k'=0}^{m-1}\binom{m-1}{k'}p_i^{k'+1}(1-p_i)^{m-1-k'}\right)-1+(1-p_i)^m\right]\\
&= \sum_{i=1}^{n-1}(n-i)\left[mp_i-1+(1-p_i)^m\right]
\end{aligned}$$

The penultimate line is obtained by fixing $k' = k-1$. □

In this very particular case, the average number of collisions and the number of guesses are hard to determine. Intuitively, we expect the last bits of the seed to be drawn more often in the monomials of degree two. As a consequence, the number of collisions is likely to be much higher. Also, the number of guesses should be greatly reduced , since we guess the bits of the seed that appears the most.

Our experimental results, shown in Table 4.5 and Table 4.6, support this intuition. Even better, for s $= 1.45$ we could not find a seed size n that forces the attacker to make at least one guess.

4.5.2 Algebraic Attack on the Ordered Case

Algebraic attacks are also more efficient for the ordered case since there is an additional structure. One can figure that, in that case, the number of equations derived from the three types method of Section 4.4.1 will find more collisions and semi-collisions. So, the limit stretch can be lowered in comparison to the non ordered case.

Table 4.5: Average number of collisions for the ordered case

n	256	512	1024	2048	4096
$\mathsf{s} = 1.45$	458	890	1703	3251	6162
$\mathsf{s} = 1.4$	271	488	873	1539	2709
$\mathsf{s} = 1.3$	95	145	221	341	520

Table 4.6: Average number of guesses for the ordered case

n	256	512	1024	2048	4096
$\mathsf{s} = 1.45$	0	0	0	0	0
$\mathsf{s} = 1.4$	0	1	2	5	9
$\mathsf{s} = 1.3$	6	10	17	30	50

4.6 Other Results

Additionally, our work [CDM+18] also has two important results that we do not detail in this document:

- *Generalization.* We generalize the guess-and-determine attack to the class of XOR-M predicates, which are divided into two parts, a linear part (the XOR part) and a non-linear part (the M part), XORed together. This captures all known candidate generalizations of Goldreich's PRG. By guessing the variables in the non-linear part, our attack takes subexponential time as soon as the stretch of the PRG is strictly above one. Importantly, our attack does not depend on the locality of the predicate, but only on the number of variables involved in the non-linear part. In a recent work [AL16], Applebaum and Lovett put forth an explicit candidate local PRG (of the form XOR-MAJ), as a concrete target for cryptanalytic effort. Our attack gives a new subexponential algorithm for attacking this candidate.
- *Extending the Applebaum-Lovett polynomial-time algebraic attack.* Applebaum and Lovett recently established that local pseudorandom generators can be broken in polynomial time, as long as the stretch s of the PRG is greater than the *rational degree* e of its predicate. We extend this result as follows: we show that the seed of a large class of local PRGs (which include all existing candidates) can be recovered in polynomial time whenever $\mathsf{s} \geq e - \log N_e / \log n$, where e is the rational degree, n is the seed size, and N_e is the number of independent annihilators of the predicate (or of its conjugate) [15] of degree at most e.

4.7 Conclusion and Open Questions

In this work, we described a guess-and-determine attack and a degree-two linearization attack (*à la* Gröbner) against Goldreich's PRG with predicate P_5. Although Goldreich's PRG is con-

[15] An annihilator of a predicate P is a non-zero polynomials Q such that $Q \cdot P = 0$, the conjugate of a predicate P is the predicate $P + 1$

jectured to be theoretically secure for a stretch approaching 1.5 by an arbitrary constant, our analysis shows that a very large seed must be used to achieve at least 80 bits of security with such a stretch. In particular, if a stretch of 1.4 is needed, no seed smaller than 5120 bits should be used. Similarly, for a stretch as small as 1.1, the seed must be at least 512 bits long. We also proved and experimented that even larger keys have to be considered for the ordered case.

This work then gives more fine-grained security parameters to consider when instantiating Goldreich's PRG. Although large seeds have to be considered for high stretches, this PRG still remains very efficient for small stretches closer to 1, thanks to its constant depth.

We also gave some challenge parameters in order to motivate the crypto community in cryptanalyzing this interesting PRG. Despite the fact that SAT Solvers are known to run in exponential time against such problems [CEMT14], it would be interesting to analyze how they behave against small seed sizes such as 512 or 1024 bits.

CHAPTER 5

CONCLUSION

Contributions to various theoretical and practical aspects of multi-party computation are presented in this thesis. These contributions lead to some interesting open problem.

Garbled circuits against malicious adversaries. The first contribution focuses on defining how a malicious adversary can corrupt a garbled circuit protocol. For a large class of circuits, we have shown that this adversary is much less powerful than what we could have expected from the previous state of the art. Formally, he is only able to add NOT gates and to make selective failure attacks. Therefore, it also suggests that cut-&-choose based solutions might be an overkill to achieve security in the malicious model and we may be able to design more specific and more efficient solutions. We leave it as an open question.

For circuits outside this class, what corruptions an adversary is able to make is still an open question. Our preliminary studies suggest that this question is highly non-trivial and may depend on the topology of the circuit being corrupted. However, they also suggest that the adversary is still very limited: only a few gates close to the generator's inputs can be modified more than with NOT gates.

Alternatively, it would also be very interesting to study whether any function can be represented by a circuit in the class we define, and with which overcost compared to an "intuitive" circuit.

Finally, this work also opens an interesting problem: can we define an oblivious transfer protocol, such that the sender has n pairs of messages $(m_{j,0}, m_{j,1})$ with $m_{j,1} = m_{j,0} \oplus \Delta$ for all $1 \leq j \leq n$ (or more generally $m_{j,1} = f(m_{j,0})$ for an arbitrary function f)? The sender has n choices b_j and wishes to obtain m_{j,b_j} for all j, with the guarantee that $m_{j,1} = m_{j,0} \oplus \Delta$. Designing such a protocol with no overcost compared to the OT-extension protocol remains an open problem.

MPC for location-based services. Our second contribution has both theoretical and practical impacts. It first shows that location-proof systems are a relevant field of applications for secure multi-party computation. It indeed allows users to prove their location without broadcasting it to everyone, enabling a wide variety of privacy-preserving location-based services.

However, it requires users devices to be equipped with directional antennas, it would be interesting to study whether we can achieve similar results without this hardware requirement, for example from distance bounding protocols.

We also provides a new secure maximum computation scheme that is asymptotically more efficient, in terms of communications and computations, than all prior works or generic solutions. However, this efficiency gain goes with a small leakage of information that we believe in many real-life scenarios, such as location-based services. We also believe that this trade-off between privacy and efficiency can be applied to many cryptographic primitives, such as private set intersection.

Analysis of MPC-friendly primitives. Finally, the last contribution describes a guess-and-determine attack and a degree-two linearization attack (*à la* Gröbner) against Goldreich's PRG with predicate P_5, which is a very interesting MPC-friendly candidate due to its extreme simplicity. Although Goldreich's PRG is conjectured to be theoretically secure for a stretch approaching 1.5 by an arbitrary constant, we show that a very large seed must be used to achieve at least 80 bits of security with such a stretch. In particular, if a stretch of 1.4 is needed, no seed smaller than 5120 bits should be used. Thus, Goldreich's PRG with predicate P_5 is limited to scenarios where a small stretch is sufficient. For example, for a stretch as small as 1.1, the seed can be as small as 512 bits. We then extended our study to other predicates with a particular interest for XOR-MAJ predicates and improved the theorem of [AL16], by taking into account the number of annihilators of the predicate.

In order to continue cryptanalyzing Goldreich's PRG, a possible direction would be to study the influence of the hamming weight of the seed on the predicates. For example, consider again the predicate P_5 that takes has input $(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, x_{i_5})$. Note that the choice of i_1 impacts the values of x_{i_2}, x_{i_3}, x_{i_4} and x_{i_5} since the positions i_j cannot be equal. Then, the input of P_5 has a small bias (that depends on the seed length and its hamming weight) and it appears that P_5 is balanced only for a uniformly random input. This ends up with a very small bias at the output of the predicate P_5, but since all the subsets are taken from the same seed, it might be possible to build a distinguisher from this observation. Our first studies on the question suggests that the two attacks on P_5 described in this document are much more efficient and allow to break more parameters. However, we believe this direction is of particular interest for other predicates with higher localities and higher stretches.

BIBLIOGRAPHY

[ABR12] B. Applebaum, A. Bogdanov, and A. Rosen. A dichotomy for local small-bias generators. In *TCC 2012*, *LNCS* 7194, pages 600–617. Springer, Heidelberg, March 2012.

[ABR16] B. Applebaum, A. Bogdanov, and A. Rosen. A dichotomy for local small-bias generators. *Journal of Cryptology*, 29(3):577–596, July 2016.

[ADI+17a] B. Applebaum, I. Damgård, Y. Ishai, M. Nielsen, and L. Zichron. Secure arithmetic computation with constant computational overhead. In *Crypto'17*, pages 223–254, 2017.

[ADI+17b] B. Applebaum, I. Damgård, Y. Ishai, M. Nielsen, and L. Zichron. Secure arithmetic computation with constant computational overhead. Cryptology ePrint Archive, Report 2017/617, 2017. `http://eprint.iacr.org/2017/617`.

[AHI05] M. Alekhnovich, E. A. Hirsch, and D. Itsykson. Exponential lower bounds for the running time of dpll algorithms on satisfiable formulas. *Journal of Automated Reasoning*, 35(1-3):51–72, 2005.

[AIK04] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004.

[AIK08] B. Applebaum, Y. Ishai, and E. Kushilevitz. On pseudorandom generators with linear stretch in nc 0. *Computational Complexity*, 17(1):38–69, 2008.

[AL07] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *TCC 2007*, *LNCS* 4392, pages 137–156. Springer, Heidelberg, February 2007.

[AL16] B. Applebaum and S. Lovett. Algebraic attacks against random local functions and their countermeasures. In *48th ACM STOC*, pages 1087–1100. ACM Press, June 2016.

[ALSZ13] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *ACM CCS 13*, pages 535–548. ACM Press, November 2013.

[AMPR14] A. Afshar, P. Mohassel, B. Pinkas, and B. Riva. Non-interactive secure computation based on cut-and-choose. In *EUROCRYPT 2014*, *LNCS* 8441, pages 387–404. Springer, Heidelberg, May 2014.

[App12] B. Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In *44th ACM STOC*, pages 805–816. ACM Press, May 2012.

[App13] B. Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM J. Comput.*, 42(5):2008–2037, 2013.

[App15] B. Applebaum. The cryptographic hardness of random local functions – survey. Cryptology ePrint Archive, Report 2015/165, 2015. `http://eprint.iacr.org/2015/165`.

[ARS+15] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In *EUROCRYPT 2015, Part I*, *LNCS* 9056, pages 430–454. Springer, Heidelberg, April 2015.

[BCD+09] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In *FC 2009*, *LNCS* 5628, pages 325–343. Springer, Heidelberg, February 2009.

[BCG+17] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, and M. Orrù. Homomorphic secret sharing: Optimizations and applications. In *ACM CCS 17*, pages 2105–2122. ACM Press, 2017.

[BCR87] G. Brassard, C. Crépeau, and J.-M. Robert. All-or-nothing disclosure of secrets. In *CRYPTO'86*, *LNCS* 263, pages 234–238. Springer, Heidelberg, August 1987.

[Bea95] D. Beaver. Precomputing oblivious transfer. In *CRYPTO'95*, *LNCS* 963, pages 97–109. Springer, Heidelberg, August 1995.

[BFSyY] M. Bardet, J.-C. Faugere, B. Salvy, and B. y. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *MEGA05, 2005. Eighth International Symposium on Effective Methods in Algebraic Geometry*.

[BGG+16] X. Bultel, S. Gambs, D. Gérault, P. Lafourcade, C. Onete, and J.-M. Robert. A prover-anonymous and terrorist-fraud resistant distance-bounding protocol. In *Proc. of WISec*, pages 121–133. ACM, 2016.

[BGI+01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO 2001*, *LNCS* 2139, pages 1–18. Springer, Heidelberg, August 2001.

[BHR12] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. In *ACM CCS 12*, pages 784–796. ACM Press, October 2012.

[BK04] I. F. Blake and V. Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In *ASIACRYPT 2004*, *LNCS* 3329, pages 515–529. Springer, Heidelberg, December 2004.

[BKK+15] D. Bogdanov, L. Kamm, B. Kubo, R. Rebane, V. Sokk, and R. Talviste. Students and taxes: a privacy-preserving social study using secure computation. Cryptology ePrint Archive, Report 2015/1159, 2015. `http://eprint.iacr.org/2015/1159`.

[BL10] C. Bordenave and M. Lelarge. The rank of diluted random graphs. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms*, pages 1389–1402. Society for Industrial and Applied Mathematics, 2010.

[BLW08] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In *ESORICS 2008*, *LNCS* 5283, pages 192–206. Springer, Heidelberg, October 2008.

[BMR90] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

[BP11] J. Boyar and R. Peralta. A depth-16 circuit for the AES s-box. Cryptology ePrint Archive, Report 2011/332, 2011. `http://eprint.iacr.org/2011/332`.

[BQ09] A. Bogdanov and Y. Qiao. On the security of goldreich's one-way function. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 392–405. Springer, 2009.

[Buc76] B. Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *SIGSAM Bull.*, 10(3):19–29, August 1976.

[CCF+16] A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In *FSE 2016*, *LNCS* 9783, pages 313–333. Springer, Heidelberg, March 2016.

[CDM+18] G. Couteau, A. Dupin, P. Méaux, M. Rossi, and Y. Rotella. On the concrete security of Goldreich's pseudorandom generator. In *ASIACRYPT 2018, Part II*, LNCS, pages 96–124. Springer, Heidelberg, December 2018.

[CEMT14] J. Cook, O. Etesami, R. Miller, and L. Trevisan. On the one-way function candidate proposed by goldreich. *ACM Transactions on Computation Theory (TOCT)*, 6(3):14, 2014.

[CFIK03] R. Cramer, S. Fehr, Y. Ishai, and E. Kushilevitz. Efficient multi-party computation over rings. In *EUROCRYPT 2003*, *LNCS* 2656, pages 596–613. Springer, Heidelberg, May 2003.

[CM01] M. Cryan and P. B. Miltersen. On pseudorandom generators in nc 0. In *International Symposium on Mathematical Foundations of Computer Science*, pages 272–284. Springer, 2001.

[CM03] N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In *EUROCRYPT 2003*, *LNCS* 2656, pages 345–359. Springer, Heidelberg, May 2003.

[Cou03] N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *CRYPTO 2003*, *LNCS* 2729, pages 176–194. Springer, Heidelberg, August 2003.

[Cv91] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT'91*, *LNCS* 547, pages 257–265. Springer, Heidelberg, April 1991.

[DCF12] B. Davis, H. Chen, and M. K. Franklin. Privacy-preserving alibi systems. In *ASIACCS 12*, pages 34–35. ACM Press, May 2012.

[DGM05] D. K. Dalai, K. C. Gupta, and S. Maitra. Cryptographically significant Boolean functions: Construction and analysis in terms of algebraic immunity. In *FSE 2005*, *LNCS* 3557, pages 98–111. Springer, Heidelberg, February 2005.

[DLR16] S. Duval, V. Lallemand, and Y. Rotella. Cryptanalysis of the FLIP family of stream ciphers. In *CRYPTO 2016, Part I*, *LNCS* 9814, pages 457–475. Springer, Heidelberg, August 2016.

[DMS05] D. K. Dalai, S. Maitra, and S. Sarkar. Basic theory in construction of Boolean functions with maximum possible annihilator immunity. Cryptology ePrint Archive, Report 2005/229, 2005. `http://eprint.iacr.org/2005/229`.

[DPB18] A. Dupin, D. Pointcheval, and C. Bidan. On the leakage of corrupted garbled circuits. In *ProvSec 2018*, LNCS, pages 3–21. Springer, Heidelberg, 2018.

[DPSZ12] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO 2012*, *LNCS* 7417, pages 643–662. Springer, Heidelberg, August 2012.

[DRB18] A. Dupin, J.-M. Robert, and C. Bidan. Location-proof system based on secure multi-party computations. In *ProvSec 2018*, LNCS, pages 22–39. Springer, Heidelberg, 2018.

[EGL82] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In *CRYPTO'82*, pages 205–210. Plenum Press, New York, USA, 1982.

[ElG84] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO'84*, *LNCS* 196, pages 10–18. Springer, Heidelberg, August 1984.

[Fau99] J.-C. Faugere. A new efficient algorithm for computing grobner bases (f4). *Journal of Pure and Applied Algebra*, 139(1):61 – 88, 1999.

[Fau02] J. C. Faugere. A new efficient algorithm for computing grobner bases without reduction to zero (f5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, pages 75–83, New York, NY, USA, 2002. ACM.

[FNO15] T. K. Frederiksen, J. B. Nielsen, and C. Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In *EUROCRYPT 2015, Part II*, *LNCS* 9057, pages 191–219. Springer, Heidelberg, April 2015.

[FZ12] M. K. Franklin and H. Zhang. Unique group signatures. In *ESORICS 2012*, *LNCS* 7459, pages 643–660. Springer, Heidelberg, September 2012.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GG09] M. Graham and D. Gray. Protecting privacy and securing the gathering of location proofs–the secure location verification proof gathering protocol. In *Proc. of MobiSec*, pages 160–171. Springer, 2009.

[GKRT14] S. Gambs, M.-O. Killijian, M. Roy, and M. Traoré. Props: A privacy-preserving location proof system. In *Proc. of SRDS*, pages 1–10. IEEE, 2014.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[Gol00] O. Goldreich. Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Report 2000/063, 2000. `http://eprint.iacr.org/2000/063`.

[GRR+16] L. Grassi, C. Rechberger, D. Rotaru, P. Scholl, and N. P. Smart. MPC-friendly symmetric key primitives. In *ACM CCS 16*, pages 430–443. ACM Press, October 2016.

[HBB12] O. Hasan, L. Brunie, and E. Bertino. Preserving privacy of feedback providers in decentralized reputation systems. pages 816–826. Elsevier Advanced Technology, 2012.

[HL10] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. ISC. Springer, Heidelberg, 2010.

[HMMB13] O. Hasan, J. Miao, S. B. Mokhtar, and L. Brunie. A privacy preserving prediction-based routing protocol for mobile delay tolerant networks. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 546–553. IEEE, 2013.

[IG03] I. Ioannidis and A. Grama. An efficient protocol for yao's millionaires' problem. In *Proc. of the 36th Annual Hawaii International Conference on System Sciences*, pages 6–pp. IEEE, 2003.

[IKNP03] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *CRYPTO 2003*, *LNCS* 2729, pages 145–161. Springer, Heidelberg, August 2003.

[IKOS08] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography with constant computational overhead. In *40th ACM STOC*, pages 433–442. ACM Press, May 2008.

[IPS08] Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. Cryptology ePrint Archive, Report 2008/465, 2008.

[JK77] N. Johnson and S. Kotz. *Urn models and their application: an approach to modern discrete probability theory*. Wiley Series in Probability and Statistics: Applied Probability and Statist ICS Sesction Series. Wiley, 1977.

[JKO13] M. Jawurek, F. Kerschbaum, and C. Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *ACM CCS 13*, pages 955–966. ACM Press, November 2013.

[KS08] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP 2008, Part II*, *LNCS* 5126, pages 486–498. Springer, Heidelberg, July 2008.

[KSC78] V. Kolchin, B. Sevastianov, and V. Chistiakov. *Random allocations*. Scripta series in mathematics. V. H. Winston, 1978.

[KSS09] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *CANS 09*, *LNCS* 5888, pages 1–20. Springer, Heidelberg, December 2009.

[laz81] D. lazard. Resolution des systemes d'equations algebriques. *Theoretical Computer Science*, 15(1):77 – 110, 1981.

[LH10a] W. Luo and U. Hengartner. Proving your location without giving up your privacy. In *Proc. of the HotMobile*, pages 7–12. ACM, 2010.

[LH10b] W. Luo and U. Hengartner. Veriplace: a privacy-aware location proof architecture. In *Proc. of SIGSPATIAL*, pages 23–32. ACM, 2010.

[Lin13] Y. Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In *CRYPTO 2013, Part II*, *LNCS* 8043, pages 1–17. Springer, Heidelberg, August 2013.

[Lin17] H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 599–629. Springer, Heidelberg, August 2017.

[LP07] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT 2007*, *LNCS* 4515, pages 52–78. Springer, Heidelberg, May 2007.

[LT05] H.-Y. Lin and W.-G. Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In *ACNS 05*, *LNCS* 3531, pages 456–466. Springer, Heidelberg, June 2005.

[LT17] H. Lin and S. Tessaro. Indistinguishability obfuscation from trilinear maps and blockwise local PRGs. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 630–660. Springer, Heidelberg, August 2017.

[LV17] A. Lombardi and V. Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In *TCC 2017, Part I*, LNCS, pages 119–137. Springer, Heidelberg, March 2017.

[LVB+16] A. Lapets, N. Volgushev, A. Bestavros, F. Jansen, and M. Varia. Secure multiparty computation for analytics deployed as a lightweight web application. Technical report, Computer Science Department, Boston University, 2016.

[Mac64] F. Macaulay. *The Algebraic Theory of Modular Systems*. Cambridge tracts in mathematics and mathematical physics. Stechert-Hafner Service Agency, 1964.

[MF06] P. Mohassel and M. Franklin. Efficiency tradeoffs for malicious two-party computation. In *PKC 2006*, *LNCS* 3958, pages 458–473. Springer, Heidelberg, April 2006.

[MJSC16] P. Méaux, A. Journault, F.-X. Standaert, and C. Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In *EUROCRYPT 2016, Part I*, *LNCS* 9665, pages 311–343. Springer, Heidelberg, May 2016.

[MNP+04] D. Malkhi, N. Nisan, B. Pinkas, Y. Sella, et al. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, pages 287—302. USENIX, 2004.

[MR13] P. Mohassel and B. Riva. Garbled circuits checking garbled circuits: More efficient and secure two-party computation. In *CRYPTO 2013, Part II*, *LNCS* 8043, pages 36–53. Springer, Heidelberg, August 2013.

[MST03] E. Mossel, A. Shpilka, and L. Trevisan. On e-biased generators in NC0. In *44th FOCS*, pages 136–145. IEEE Computer Society Press, October 2003.

[NPS99] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 129–139. ACM, November 1999.

[OW14] R. ODonnell and D. Witmer. Goldreich's prg: evidence for near-optimal polynomial stretch. In *Computational Complexity (CCC), 2014 IEEE 29th Conference on*, pages 1–12. IEEE, 2014.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, *LNCS* 1592, pages 223–238. Springer, Heidelberg, May 1999.

[Ped91] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*, pages 129–140. Springer, 1991.

[PHB+15] A. Pham, K. Huguenin, I. Bilogrevic, I. Dacosta, and J.-P. Hubaux. Securerun: Cheat-proof and private summaries for location-based activities. In *Proc. of TMC*, pages 2109–2123. IEEE, 2015.

[PS16] B. Pittel and G. B. Sorkin. The satisfiability threshold for k-xorsat. *Combinatorics, Probability and Computing*, 25(2):236–268, 2016.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[Sch90] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO'89*, *LNCS* 435, pages 239–252. Springer, Heidelberg, August 1990.

[Sha79] A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[Sie84] T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications (corresp.). *IEEE Transactions on Information theory*, 30(5):776–780, 1984.

[SP05] D. Singelee and B. Preneel. Location verification using secure distance bounding protocols. In *Proc. of MASS*, pages 7–14. IEEE, 2005.

[sS11] a. shelat and C.-H. Shen. Two-output secure computation with malicious adversaries. In *EUROCRYPT 2011*, *LNCS* 6632, pages 386–405. Springer, Heidelberg, May 2011.

[sS13] a. shelat and C.-H. Shen. Fast two-party secure computation with minimal assumptions. In *ACM CCS 13*, pages 523–534. ACM Press, November 2013.

[SSW03] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *Proc. of WISEC*, pages 1–10. ACM, 2003.

[SW09] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *Proc. of HotMobile*, pages 1–6. ACM, 2009.

[SW14] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[TCB10] M. Talasila, R. Curtmola, and C. Borcea. Link: Location verification through immediate neighbors knowledge. In *Proc. of MobiSec*, pages 210–223. Springer, 2010.

[Wie86] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE transactions on information theory*, 32(1):54–62, 1986.

[WMK17] X. Wang, A. J. Malozemoff, and J. Katz. Faster secure two-party computation in the single-execution setting. In *EUROCRYPT 2017, Part III*, *LNCS* 10212, pages 399–424. Springer, Heidelberg, May 2017.

[Yao82] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

[Yao86] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[ZC11] Z. Zhu and G. Cao. Applaus: A privacy-preserving location proof updating system for location-based services. In *Proc. of INFOCOM*, pages 1889–1897. IEEE, 2011.

[ZM05] S. Zhang and F. Makedon. Privacy preserving learning in negotiation. In *Proc. of SAC*, pages 821–825. ACM, 2005.

[ZRE15] S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *EUROCRYPT 2015, Part II*, *LNCS* 9057, pages 220–250. Springer, Heidelberg, April 2015.

Producteurs

Christophe BIDAN — David POINTCHEVAL

Coproducteurs

Renaud DUBOIS — Éric GARRIDO — Jean-Marc ROBERT

Postproducteurs

Sébastien CANARD — Jean-Sébastien CORON — Marine MINIER

Équipe de tournage

Costumes	Anca NITULESCU	Photographe en second	Emeline HUFSCHMITT
Cascades	Georg FUCHSBAUER		
	Michele ORRU	Eleboniste	Thomas PREST
Piano	Geoffroy COUTEAU	Jeux de société	Florian BOURSE
Logistique	Romain GAY		Antoine PLOUVIEZ
	Balthazar BAUER		Ange MARTINELLI
Traduction	Michele MINELLI		Mélissa ROSSI
	Razvan ROŞIE		Thomas RICOSSET
	Quoc Huy VU	Punchline	Chloé HÉBANT
Scénario	Pierrick MÉAUX	Houda team	Dahmun GOUDARZI
Chefs de production	Didier LE MAITRE		Adrian THILLARD
	Laurent FREREBEAU	Hackeurs	Anne VIGUIÉ
Fournisseur de café	Michel ABDALLA	administratifs	Karine BERNARD
Fournisseur de dlog	Edouard DUFOUR SANS		Lise-Marie BIVARD
		Soutien émotif	Megguy GUYON
Anonymous reviewer	Brice MINAUD	Soutien culinaire	PAPA et MAMAN

Figurants

Alexandre ANZALA-YAMAJAKO	Pooya FARSHIM	Simon MASSON
	Houda FERRADI	Thierry MEFENZA NOUNTU
Sonia BELAID	Mickael GEFFRAULT	Olivier ORCIÈRE
Fabrice BENHAMOUDA	Matthieu GIRAUD	Philippe PAINCHAULT
Olivier BERNARD	Junqing GONG	Alain PASSELÈGUE
Céline CHEVALIER	Julia HESSE	Julien PRAT
Jérémy CHOTARD	Louiza KHATI	Théo RYFFEL
Léo COLISSON	Jean KIEFFER	Quentin SANTOS
Rafael DEL PINO	Sylvain LACHARTRE	Damien VERGNAUD
Pierre-Alain DUPONT	David LEFRANC	Hoeteck WEE

Cette thèse est une oeuvre de fiction. Toute ressemblance avec la réalité serait purement fortuite.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

THALES

CentraleSupélec

Titre: Calculs Multi-Parties et Vie Privée

Mot clés : Multi-party computation, vie privée, garbled circuits, preuve de localisation, Goldreich's PRG

Resumé : Les calculs multi-parties sécurisés (MPC) sont une branche de la cryptographie qui a pour objectif de concevoir des solutions permettant à plusieurs parties de calculer ensemble une fonction de leurs données, tout en gardant ces données secrètes. Contrairement à la cryptographie classique, où l'on cherche à assurer la sécurité malgré la présence d'un adversaire extérieur, le MPC garantit la sécurité face à un adversaire interne contrôlant un ou plusieurs participants.

Cette thèse apporte à la fois des contributions théoriques et pratiques dans le domaine du MPC. D'un point de vue théorique, une étude est réalisée sur la corruption des "garbled circuits", qui sont une solution générale au problème à deux parties.

Sur un plan pratique, nous réalisons une cryptanalyse de certaines primitives propres au MPC, dans le but d'étudier leur efficacité réelle. Enfin, nous montrons que les services basés sur la position des utilisateurs peuvent prendre avantage du MPC pour devenir plus respectueux de la vie privée.

Title: Secure Multi-Party Computation and Privacy

Keywords : Multi-party computation, privacy, garbled circuits, location-proof, Goldreich's PRG

Abstract: Secure multi-party computation (MPC) is a subfield of cryptography that aims at designing protocols for parties to cooperatively compute a function over their inputs while keeping those inputs private. Unlike traditional cryptographic tools (encryption, signature, ...), where cryptography ensures security and integrity of communication or storage against an external eavesdropping adversary, MPC assures security against an internal adversary, that controls one or more of the actual participants.

Both theoretical and practical contributions to MPC are made in this thesis. From a theoretical point of view, we study the possible corruptions of garbled circuits, which is a general solution for the two-party case.

On a practical level, we cryptanalyze some MPC-friendly primitives in order to assess their concrete efficiency. Finally, we also show that MPC can be used to build privacy-preserving location-based services.