Integrity, authentication and confidentiality in public-key cryptography

Houda Ferradi

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'École normale supérieure

Integrity, Authentication and Confidentiality in Public-Key Cryptography

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité Informatique

Soutenue par **Houda FERRADI**
le 22 septembre 2016

Dirigée par
David NACCACHE
École normale supérieure

COMPOSITION DU JURY

M. FOUQUE Pierre-Alain
Université Rennes 1
Rapporteur

M. YUNG Moti
Columbia University et Snapchat
Rapporteur

M. FERREIRA ABDALLA Michel
CNRS, École normale supérieure
Membre du jury

M. CORON Jean-Sébastien
Université du Luxembourg
Membre du jury

M. GOUBIN Louis
Université de Versailles Saint-Quentin-en-Yvelines
Membre du jury

M. PAILLIER Pascal
CryptoExperts
Membre du jury

M. TIBOUCHI Mehdi
NTT Secure Platform Laboratories
Invité

ENS ÉCOLE NORMALE SUPÉRIEURE | PSL RESEARCH UNIVERSITY PARIS

Intégrité, authentification et confidentialité en cryptographie à clé publique

Thèse de Doctorat

en vue de l'obtention du grade de

Docteur de l'École normale supérieure
(spécialité informatique)

présentée et soutenue publiquement le 22 septembre 2016 par

HOUDA FERRADI

devant le jury composé de :

Directeur de thèse :	David Naccache	(École normale supérieure)
Rapporteurs :	Pierre-Alain Fouque	(Université Rennes 1)
	Moti Yung	(Columbia University et Snapchat)
Examinateurs :	Michel Abdalla	(CNRS, École normale supérieure)
	Jean-Sébastien Coron	(Université du Luxembourg)
	Louis Goubin	(Université de Versailles Saint-Quentin-en-Yvelines)
	Pascal Paillier	(CryptoExperts)
Invité :	Mehdi Tibouchi	(NTT Secure Platform Laboratories)

École doctorale 386: Sciences mathématiques de Paris Centre
Unité de recherche: UMR 8548 - Département d'Informatique de l'École normale supérieure
Laboratoire de recherche affilié au CNRS et a INRIA

Integrity, Authentication and Confidentiality in Public-Key Cryptography

Doctorate Dissertation

submitted in fulfillment of the requirements for the degree of

Doctor of the École normale supérieure
(Specialty: Computer Science)

publicly defended and presented on September 22nd, 2016 by

HOUDA FERRADI

to the jury consisting of :

Supervisor :	David Naccache	(École normale supérieure)
Referees :	Pierre-Alain Fouque	(Université Rennes 1)
	Moti Yung	(Columbia University and Snapchat)
Examiners :	Michel Abdalla	(CNRS, École normale supérieure)
	Jean-Sébastien Coron	(Université du Luxembourg)
	Louis Goubin	(Université de Versailles Saint-Quentin-en-Yvelines)
	Pascal Paillier	(CryptoExperts)
Guest member :	Mehdi Tibouchi	(NTT Secure Platform Laboratories)

Doctoral School 386: Mathematical Sciences – Paris Centre
Research unit: UMR 8548 - The École normale supérieure's Computer Science Department
A research laboratory affiliated to CNRS and INRIA

ACKNOWLEDGMENTS

I very warmly thank *David Naccache* for his advisory role, continuous good humor, friendliness and for his scientific guidance throughout the years.

I am very grateful to *David Pointcheval* for admitting me in his research group and for funding my thesis through the French ANR Project ANR-12-INSE-0014 SIMPATIC.

I express my affection to the numerous members of the ENS' Cryptography and Security teams. I will always cherish the souvenirs of thought-provoking talks in the laboratory, of your friendship and of your witty humor. The feverish intellectual thrill of discovering together the cutting edge of scientific research at Eurocrypt'15, Eurocrypt'16, Crypto'16, CHES'16 and ACNS'16 was a unique experience that words can hardly express.

I am indebted to the team's permanent members for their guidance, life-wisdom and for entrusting me with the review of several papers. The ensuing exchanges with program committee members were an unforgettable scientific experience.

I am grateful to EIT ICT Labs for funding my participation at the 2014 Security & Privacy in Digital Life summer school (Trento) and to ENS for sponsoring my participation at the IACR 2016 School on Design for a Secure IoT, (Tenerife) and at the IACR 2015 School on Design and Security of Cryptographic Algorithms and Devices (Sardinia).

A tribute is due to *Rémi Géraud* for his friendship and efforts invested in our common papers and to *Fabrice Ben Hamouda* for his gentleness and dedication.

I thank my co-authors *Michel Abdalla*, *Ehsan Aerabi*, *A. Elhadi Amirouche*, *Fabrice Ben Hamouda*, *Thomas Bourgeat*, *Julien Bringer*, *Robin Champenois*, *Jean-Michel Cioranesco*, *Jérémie Clément*, *Simon Cogliani*, *Rémi Géraud*, *Marc Heinrich*, *Julien Jainski*, *Diana Maimuţ*, *Kostas (Konstantinos) Markantonakis*, *Mehari Msgna*, *Paul Melotti*, *David Naccache*, *Raja Naeem Akram*, *David Pointcheval*, *Assia Tria*, *Antoine Voizard*, *Jean Vuillemin*, *Amaury de Wargny* and *Hang Zhou*.

I am very grateful to the academic institutions who entrusted me with lecturing duties during my Ph.D.: Université Panthéon-Assas Paris II, Université Paris Diderot-Paris VII, Université Paris-XIII-Nord and, in particular, the École normale supérieure for letting me supervise full-fledged research projects at the *Informatique scientifique par la pratique* master course. It my hope that my efforts motivated my students and made my courses an intellectual adventure as much as a curricular obligation.

I express my recognition to *Michel Abdalla*, *Jean-Sébastien Coron*, *Pierre-Alain Fouque*, *Louis Goubin*, *Pascal Paillier*, *Mehdi Tibouchi* and *Moti Yung* for agreeing to serve in my thesis committee. I particularly thank my thesis referees *Pierre-Alain Fouque* and *Moti Yung* for their availability and their detailed comments on my work. I am very honored to have such a prestigious committee.

The research results contained in this thesis were supported by three high-tech firms. Seeing my scientific results applied in products used by millions of customers is a thrilling feeling, for which I warmly thank *Ingenico*, *Huawei* and *Tanker*. I thank Google for inviting me to the Münich Ph.D. Security Summit. The intense intellectual exchanges with Google's technical teams were an unforgettable experience.

Paris, Septembre 13th, 2016.
Houda Ferradi

CONTENTS

CHAPTER 1

INTRODUCTION

In 2016, the *New Encyclopedia Britannica* defines cryptology as: *"The practice of the enciphering and deciphering of messages in secret code in order to render them unintelligible to all but the intended receiver"*. Cryptology encompasses the study (*logia*, λογία) of both *cryptography* and *cryptanalysis*. The term cryptography comes from the Greek words: *kryptós* (κρπτός), which means "hidden" or "secret" and *graphein* (γράφειν) which means "writing".

Historically, cryptography was focused on the transforming of private correspondence into unreadable sequences of figures to protect the content of messages transferred from one geographic location to another.

Cryptography exists almost since the invention of writing. Nearly all ancient civilizations created some type of cryptic writing or cryptography. Until recently cryptography was considered as an art or as a game meant to construct, analyze and break codes. For centuries the discipline remained reserved to diplomats and military commanders. Modern cryptographic history began in the early 1900s with the advent of electrical communication technologies and the emergence of the Internet during the last twenty-five years.

One the main goals of modern cryptography is the designing of new cryptographic primitives and protocols. To prove the security of such algorithms, cryptographers proceed by steps: they first formally define security notions (this is done by defining a theoretical model capturing how an adversary could interact with the target system and the way in which we define "breaking" the system). We then design new schemes and prove their security within the framework of the previously designed model.

As we write these lines, the applications of cryptography abound. Cryptography is used for data encryption, message integrity checking, identity authentication, digital signatures, pseudo-random number generation, zero knowledge proofs, commitment schemes, e-voting, secret sharing, and secure multiparty computation (secure function evaluation), to name just a few.

The field is very active and thousands of scientific papers in cryptology are published every year.

1.1 Confidentiality Throughout History

To understand the notion of confidentiality we must travel back in time to 1900 B.C. when Egyptian scribes used hieroglyphs in an unconventional way, presumably to hide the the significance of text from laymen who did not know the meaning of the modified symbols [1]. The Greeks' celebrated scytale [2] was an ingenious yet simple instrument allowing to wrap a leather band around a stick, and then write the message on the band. When the band was unwrapped, the writing would appear meaningless. The

1. The practice is ancient although the best known examples are Ptolemaic hieroglyphs (second century B.C.).
2. σκυτάλη meaning "stick".

receiver of the message would have a scytale of an identical diameter and use it to decipher the message. As for the Romans, the most famous method providing message confidentiality was known as the *Caesar Shift Cipher* as attested by Suetonius in *De Vita Caesarum, LVI*:

Exstant et ad Ciceronem, item ad familiares domesticis de rebus, in quibus, si qua occultius perferenda erant, per notas scripsit, id est sic structo litterarum ordine, ut nullum verbum effici posset; quae si qui investigare et persequi velit, quartam elementorum litteram, id est D pro A et perinde reliquas commutet.

The Caesar Shift Cipher has long been used as a *Monoalphabetic Cipher*. It utilized the famous idea of shifting letters by an agreed upon number of positions (three positions was a common historical choice, this parameter is also called the *key*), and thus writing the message using the letter-shift. The receiver would then shift the letters backwards by the same number of positions and decipher the message. In other words, this method achieves a transformation by aligning two alphabets where the cipher alphabet is the plain alphabet rotated to the left (or to the right) by a given number of positions. For example to encrypt with three positions, 'A' is substituted by 'D','B' is substituted by 'E', etc... It is interesting to see why this encryption method is simple to break. All the attacker has to do is to go down the alphabet, juxtapositioning the start of the alphabet to each succeeding letter. At each iteration, the message is decrypted to see if it makes sense. When the result appears as a readable message, the code has been broken. Another method for breaking monoalphabetic ciphers consists in using frequency analysis introduced by the Arabs [3] circa 1000 C.E. This method is based on the principle that certain letters (for instance in English the letter 'E') are used more often than others. Armed with this knowledge, a person could go over a message and look for the repeated use, or the frequency of use, of a particular letter and try to identify frequently used letters.

The traditional doctrine of encryption has long remained based on keeping the encryption algorithm secret from the adversary. This postulate was called into question by Auguste Kerckhoffs in the 19th century and became obsolete [Ker83]. Kerckhoffs' fundamental idea that has revolutionized our insight of cryptography, was that a cryptosystem must remain secure even if an adversary knows everything about the system, except the key. Therefore security should only be based on keeping the key secret.

The author thinks that Kerckhoffs' concept also contributed to transform cryptography into a mathematical discipline. Mathematicians try to determine unknowns in knows systems. Hence if both the algorithm (system) and the key (unknown) are beyond the analysts reach there is nothing to work on mathematically.

In information theory, the security notion of *perfect privacy* refers to the situation where it is impossible for an adversary to extract any information about the plaintext from the ciphertext. However the big bulk of modern cryptography theory defines the notion of privacy in the sense of *infeasibility* rather than *impossibility* as adversaries are computationally bounded in practice.

In this computational-complexity approach, as long as the leaked information cannot be efficiently exploited (in polynomial time) then it is tolerated for a ciphertext to leak information about the plaintext [**Abdalla2001**]. The most famous approach for defining privacy is that of indistinguishability of ciphers, introduced by Goldwasser and Micali in [GM84].

1.2 Integrity, Authentication & Fairness

These three security functions are provided by both symmetric and asymmetric cryptography. Just like their real world counterparts, digital signatures (and message authentication codes – MACs) enable the receiver of digital information to verify the authenticity of its origin (authentication) and check that digital information is intact (integrity). In addition to providing authentication and integrity, digital signatures also provide a fundamental feature to cryptography called *non-repudiation*. Non-repudiation prevents the sender from denying that he did send the information.

Historically, data integrity and authentication were both provided by the physical support of the message (paper watermarks and wax seals). However, in the modern era reliance on physics became obviously

3. cf. Al-Kindi's "Manuscript on Deciphering Cryptographic Messages".

unsuited.

Thus digital signatures became a fundamental building block of most cryptographic applications. As we write these lines, digital signatures are essential in applications and commercial environments such as software distribution, financial transactions, contract management software and digital currencies.

The distinction between integrity and authentication is frequently blurred because integrity can also provide authentication. In essence, an integrity primitive would take as a parameter a message m and prove that the sender actually mixed his secret with m to attest m's origin. An authentication primitive does not involve any message (no "payload") and is only meant to check that the authenticated party actually knows a given secret. It follows that to achieve authentication the secret owner can just be challenged to attest the integrity of a random challenge m, chosen by the verifier. In practice, this is indeed the way in which numerous commercial products implement authentication using integrity primitives.

In many real-life applications, various problems persist even when confidentiality, integrity and authentication are enforced.

For instance, in a setting where mutually distrustful parties wish to compute some joint function of their private inputs (secure multiparty computation), *fairness* must be enforced. In particular, digital contract signing belongs to the wider concept of *fair exchange*, i.e., enabling two (or multiple) potentially distrustful parties to exchange digital signatures over a public channel through a process that ascertains that (1) each party obtains the other's signature, or (2) neither party does.

In a fair contract signing protocol Alice and Bob swap their commitments to a contract in a *fair* way. All known fair contract signing protocols published so far rely on trusted third parties [BGMR90; Rab83], arbitrators [ASW97; ASW02] or non-asymptotic computational power considerations [Blu83; EGL85; BGMR90].

Any multi-party computation can be securely computed [Yao86; GMW87b; Gol04; BGW88; CCD88] as long as there is a honest majority [Lin08]. In the case where there is no such majority, and in particular in the two-party case, it is (in general [4]) impossible to achieve both fairness and guaranteed output delivery [Lin08; Cle86].

4. See [GHKL08] for a very specific case where completely fair two-party computation can be achieved.

CHAPTER 2

MATHEMATICAL AND CRYPTOGRAPHIC PRELIMINARIES

We recall in this section some prerequisites necessary to understand the next chapters. We will start by introducing the notion of *Computational Hardness Assumption* which is a fundamental concept in modern cryptography, then we introduce several primitives used throughout this thesis.

2.1 Computational Hardness Assumptions

In public-key cryptography it is usually impossible to prove that a cryptographic primitive is secure in an absolute sense against an adversary with unlimited computational power. That led Diffie and Hellman to justify the concept of public key cryptography in the 1970s [DH76]. Henceforth cryptography has started to be oriented towards a weaker yet more realistic security notion called *computational security*.

The notion of computational security is formalized using the computational approach borrowed from complexity theory, assuming that there exist hard problems (hardness assumptions) which are impossible to solve when an adversary can only run in *reasonable* amount of time and succeed with *non-negligible* probability. The goal of this section is to recall the definitions of the main hardness assumptions encountered in cryptography, as well as the notions of "reasonable" and "non-negligible".

Definition 2.1 (polynomial Time Algorithm) *An algorithm A solving a given problem Q is called polynomial (or runs in polynomial time) if there exists a polynomial $p(.)$, such that the time taken by A to solve every instance of Q is upper bounded by $p(size(input))$, the polynomial $p(.)$ evaluated at the size of the input.*

Definition 2.2 (Probabilistic Polynomial Time (PPT)) *An algorithm A is called probabilistic polynomial if A uses random sources (for example, tosses coins) to decide its next actions during execution.*

Note that a random source is an idealized device that outputs a sequence of bits that are uniformly and independently distributed.

Definition 2.3 (Negligible Function) *A function f is called a negligible function if for every positive polynomial $p(.)$ there exists an integer N, such that, for all $n \geq N$, $f(n) \leq \frac{1}{p(n)}$. In others words, if a function is asymptotically smaller than the inverse of any fixed polynomial then it is considered as a negligible function.*

There are two main approaches in computational security: the *concrete* approach and the *asymptotic* approach. The first is a *practice-oriented provable security* paradigm developed by Bellare and Rogaway [BRW03], this methodology considers t-time attackers with ϵ-advantages. Asymptotic security considers PPT algorithms and "negligible" advantages. To quantify how close we come to the ideal, we explicitly bound the maximum success probability of adversary A running in certain time while attacking a primitive S:

Definition 2.4 *A scheme is (t, ϵ)-secure if every adversary running for time at most t succeeds in breaking the scheme with probability at most ϵ.*

We can also bound t by the number of atomic computations or by considering performance metrics such as CPU cycles, the number of gates in a circuit, the amount of communication etc. Computational complexity theory aims at determining the practical limits on what computers can and cannot do. The final goal of this endeavor is *the analysis of algorithms*, i.e. finding-out how fast computers can perform, knowing the amount of resources at hand (such as execution time and storage).

Furthermore, one last element considered in computational security is the fact that the security of cryptographic constructions is based on the hardness of specific problems. A given problem is considered *hard* if the success probability of any PPT algorithm solving it, is bounded by a negligible function. However, the hardness of problems is evaluated asymptotically since we cannot decide the size of parameters to choose. In practice, the choice is given by classifying hard problems according to the best algorithms allowing to solve them. This, in turn, translates into the computational power that an adversary must have in his possession to break the cryptographic primitive.

In this section, we recall the definitions of different cryptosystems based on the computational difficulty of inverting various *trapdoor functions*. The main conjecture related to these functions is the existence of *trapdoor one-way functions*: i.e. that an attacker cannot invert these functions in the computational complexity-theoretical sense, using only public data. We will discuss in section 2.3 the theoretical veracity of this assumption.

Definition 2.5 *A trapdoor one-way function is a function $f : X \to f(X)$ such that for all $x \in X$, it is easy to compute $y = f(x)$ (forward computation) whereas it is hard to compute its inverse i.e. $x = f^{-1}(y)$ (reverse computation or backward computation) without knowing a secret trapdoor information s.*

Consequently, choosing one assumption instead of another is a risky task for a cryptographer wishing to design a new cryptographic scheme since we must hope that a given underlying assumption has been properly and sufficiently evaluated by the scientific community. Such an evaluation must quantify the amount of time one could take to break the intended system's secrecy by making the backwards computations (while the forward computation is expected to remain easy). Note that the naive reverse computation algorithm takes $O(2^n)$ steps, such that n is the number of bits.

Definition 2.6 *A cryptosystem is secure if the fastest known algorithm A reversing the computation of its trapdoor function runs in exponential time. A cryptosystem is moderately secure if A runs in sub-exponential time, and a cryptosystem is considered insecure if A runs in polynomial time with respect to the size of the input.*

Thus, when we need to analyze a new cryptographic protocol relying upon a new assumption we need to compare that new assumption to the main hardness assumptions presented below by using the polynomial reduction method presented in section 2.4.1 and formalizing them, considering running time with respect to the size of the input of functions or their "key size".

The Discrete Logarithm Problem - DLP

Definition 2.7 (The Discrete Logarithm Problem - DLP) *Let g be a generator of group G (of order q). Given $g, p, h \in_R G$, find a such that $h = g^a$.*

Since the hardness of the DLP is the foundation of several cryptographic systems (e.g. Diffie–Hellman key agreement [DH76], ElGamal encryption and signature [El 84] or the Schnorr signature[Sch90]), it is important to take into account the best records achieved so far in solving the DLP.

The DLP's difficulty depends on the choice of the group G. The DLP takes sub-exponential time in $\mathbb{F}_p$ and is even harder (exponential) in elliptic curves $E(\mathbb{F}_p)$.

The last record in solving the discrete logarithm problem (16 June 2016), using the number field sieve, is the computation of a discrete logarithm modulo a 232-digit prime which roughly corresponds to the factoring of a 768-bits safe prime.

Definition 2.8 (The Computational Diffie-Hellman Problem - CDHP) *Given a finite cyclic group G of order q, a generator g of G, and two elements g^a and g^b, find the element g^{ab}.*

This problem introduced by Diffie and Hellman [DH76] is the cornerstone the celebrated Diffie-Hellman key exchange protocol. CDHP is the security foundation of many cryptosystems (for example ElGamal encryption).

CDH is trivially reduced to DLP: Given g^a and g^b we need to find g^{ab}, first we proceed by computing $\text{Dlog}(g, g^b)$ using an oracle, then we can compute $(g^a)^b = g^{ab}$. Thus it is straightforward to show that CDH is no harder than DLP.

Definition 2.9 (The Decisional Diffie-Hellman Problem - DDHP) *Given a finite cyclic group G, a generator g of G, three elements g^a, g^b and g^c, decide whether the elements g^c and g^{ab} are equal.*

The DDHP is a very important computational hardness assumption used in the ElGamal and Cramer–Shoup cryptosystems [CS98], to name just two. It is believed that CDH is a weaker assumption than DDHP since there are groups in which solving DDHP is easier than solving CDHP problems.

DLP versions for ECC The elliptic curve variant of definition 2.7 is given below.

Definition 2.10 (The Elliptic Curve Discrete Logarithm Problem - ECDLP) *Let E be an elliptic curve over the finite field $\mathbb{F}_p$ and let P and Q be points in $E(\mathbb{F}_p)$. The elliptic curve discrete logarithm problem consists in finding $n \in \mathbb{N}$ such that $Q = [n]P$.*

Pairing-Based Cryptography. Let g be a generator for a group G,of prime order q, and let e be a bilinear map on G, i.e. a function linear in each of its two arguments combining elements of two vector spaces to yield an element of a third vector space.

Definition 2.11 *Let G_1, G_2 be two additive cyclic groups of prime order p, and G_T a multiplicative cyclic group of order p. A pairing is an efficiently computable map $e : G_1 \times G_2 \to G_T$ satisfying*

- **bilinearity**: *$\forall a, b \in F_p^*$ and $\forall P \in G_1, Q \in G_2$ we have $e([a]P, [b]Q) = e(P, Q)^{ab}$*
- **non-degeneracy**: *$e(P, Q) \neq 1$*

Definition 2.12 (Type 1, 2, 3 Pairings [GPS08]) *We further categorize e as:*

- *Type 1, when $\mathbb{G}_1 = \mathbb{G}_2$;*
- *Type 2, when $\mathbb{G}_1 \neq \mathbb{G}_2$ but there exists an efficiently computable homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$, while there does not exist efficient such maps exists in the reverse direction;*
- *Type 3, when $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ exists, in either direction.*

Example 2.1 *Type 3 pairings can be constructed, for instance, on Barreto-Naehrig curves where $\mathbb{G}_1$ is the group of $\mathbb{F}_q$-rational points (of order p) and $\mathbb{G}_2$ is the subgroup of trace zero points in $E(\mathbb{F}_{q^{12}})[p]$.*

Definition 2.13 (The Bilinear Diffie-Hellman Problem - BDHP) *Let G, G_T be two cyclic groups of a large prime order p. Let $e \in G_T$ be a bilinear pairing. Given $g, g^x, g^y, g^z \in G^4$, compute $e(g, g)^{xyz} \in G_T$.*

Definition 2.14 (The Bilinear Decisional Diffie-Hellman Problem - BDDHP) *Let G, G_T be two cyclic groups of a large prime order p. Let $e \in G_T$ be a bilinear pairing. Given $g, g^x, g^y, g^z \in G$ and $e(g, g)^w \in G_T$, decide if $w = xyz \bmod p$.*

Obviously, BDHP $\Rightarrow$ BDDHP: Indeed if we can compute the bilinear pairing $e(g, g)^{xyz}$, then we solve BDDHP by comparing the result we got to the provided value $e(g, g)^w$.

Factorization-Related Problems - FACT and ERP In this section, we will recall the problem of factoring composite integers and several related problems. We will consider their security in the sense of complexity-theoretic reductions described in section 2.4.1. To determine which key sizes we have to choose for cryptosystems whose security relates to the hardness of the factoring problem, we need to investigate various algorithms solving factorization-related problems, for that, we have to study their asymptotic behavior, as well as their practical running times, based on experiments reported in the state of the art.

Definition 2.15 (Factoring Problem - FACT**)** *Given a positive integer N, find its prime factors, i.e., find the pairwise distinct primes p_i and positive integer powers e_i such that $N = p_1^{e_1} ... p_n^{e_n}$.*

It is generally believed that the most difficult setting for FACT is when $N = pq$ is the product of only two primes p and q of the same large size since the difficulty of FACT is nonuniform (i.e factoring integers N whose second largest prime factor is bounded by a polynomial in $\log N$ can be performed in polynomial time). It is straightforward to compute $N = pq$ in $O(k^2)$ time and (presumably) hard to invert this operation when p and q are pairwise distinct primes chosen randomly.

Another important notion to consider in the generation of p and q is the notion of *smoothness*.

Definition 2.16 *An integer is B-smooth if all its prime factors are smaller than B.*

The security requirement for FACT is that $p - 1$ and $q - 1$ should not be smooth.

Definition 2.17 (The e-th Root Problem - ERP**)** *Given a group G of unknown order, a positive integer $e < |G|$ and an element $a \in G$, find an element $b \in G$ such that $b^e = a$.*

Definition 2.18 (The RSA Problem - RSA**)** *The RSA problem is an ERP in $\mathbb{Z}_N$. Given (N, e, c) where $y \in \mathbb{Z}_N$ and $N = pq$, find x such that $c = x^e \bmod N$*

RSA relies on the difficulty for solving equations of the form $x^e = c \bmod N$, where e, c, and N are known and x is an arbitrary number. In other words, the security of RSA relies on the assumption that it is difficult to compute e-th roots modulo N, *i.e.* on ERP'S hardness in $\mathbb{Z}_N$.

The RSA Problem is clearly no harder than FACT since an adversary who can factor N can also compute the private key (p, q, d) from the public key (N, e). However, so far there are no proofs that the converse is true meaning that the RSA problem is only *apparently* as difficult as FACT: Whether an algorithm for solving the RSA Problem can be efficiently converted into an integer factoring algorithm is an important open problem. However, Boneh and Venkatesan [BDH99] have given evidence that such a reduction is unlikely to exist when the public exponent is very small, such as $e = 3$ or 17.

It is important to know which parameter sizes to choose when the RSA problem serves as the foundation of a cryptosystem. The current record for factoring general integers was annouced on December 12 in 2009, by a team including researchers from CWI, EPFL, INRIA and NTT. The consortium factored the RSA-768 (232-digit number) using the number field sieve (NFS) [Cas03]. This effort required the equivalent of almost 2000 computing years on a single core 2.2 GHz AMD Opteron. Now the NIST's recommendation is that future systems should use RSA keys with a minimum size of 3072 bits.

In 1994, Peter Shor [Sho97] introduced a quantum algorithm solving FACT in polynomial time. A discussion about the practical significance of the various quantum computing experiments conducted so far exceeds the scope of this introduction.

Definition 2.19 (Residuosity Problems - QRP**)** *Let $a, N, m \in \mathbb{N}$ with $\gcd(a, N) = 1$. a is called an m-th residue $\bmod N$ if there exists an integer x such that $a \equiv x^m \bmod N$.*

The residuosity problem may refer to *quadratic* or to *higher* residues.

Definition 2.20 *Let N be the product of two primes p and q. An element $a \in \mathbb{Z}_N$ is a quadratic residue modulo n (or a square) if there exists $w \in \mathbb{Z}_N$ such that $w^2 \equiv a \bmod N$. If there exist no such $w \in \mathbb{Z}_N$, a is called a quadratic non-residue.*

Definition 2.21 (The Quadratic Residuosity Problem - QRP**)** *Given $a, N \in \mathbb{N}$, $0 \leq a < N$, decide if a is a quadratic residue.*

Definition 2.22 (The Higher Residuosity Problem - HRP**)** *Given $a, N, m \in \mathbb{N}$, $0 \leq a < N$, $\gcd(a, N) = 1$ decide if a is an m-th residue.*

Note. QRP's intractability is the basis of the security of Goldwasser–Micali's cryptosystem [GM82], the first provably secure probabilistic public key encryption scheme [1]. Paillier's cryptosystem [Pai99] is the best known example of a scheme whose underlying hardness assumption is the HRP.

1. In the case of a probabilistic encryption scheme a message is encrypted into one of many possible ciphertexts.

2.2 Computational Security

Shannon introduced in [Sha48; Sha49] notions that form the mathematical foundations of modern cryptography. Shannon defined the concepts of *perfect secrecy* and defined the *entropy* of natural languages. Moreover, he provided the first security proofs using probability theory and established exact connections between provable security, key size, plaintext and ciphertext spaces.

The security of a cryptosystem relies on certain assumptions. Cryptographers distinguish between:
- Information-theoretically secure [2] cryptosystems that no amount of computation can break.
- Computationally secure cryptosystems, based on the computational intractability of breaking them.

In other words, no attack exists against unconditionally secure cryptosystems whereas attacks against computationally secure cryptosystems exist in theory but are intractable in practice.

Definition 2.23 *The entropy (or,* equiv., *uncertainty) $H(X)$ of a discrete random variable X with possible values x_i is defined as the expectation of the negative logarithm of the corresponding probability P [3]:*

$$H(X) = -\sum_i P(x_i) \log P(x_i) \,.$$

Definition 2.24 *A secret key cipher is perfect if and only if $H(M) = H(M|C)$, i.e., when the ciphertext C reveals no information about the message M.*

Corollary 2.1 *A perfect cipher is unconditionally (or information-theoretically) secure against ciphertext only attacks [4] (*COAs*).*

Complexity Theory

Complexity theory is the core of theoretical computer science. The main goal of complexity theory is to quantify the difficulty of solving specific decision problems and classify computational problems according to their inherent hardness. A set of problems of related complexity is referred to as a *complexity class*, which means a set of problems of related resource-based complexity i.e. problems solved by an abstract machine M using $O(f(n))$ of some resource ρ, where n is the input size of the input and ρ is typically time or memory (also called "space").

Whatever the algorithm used, a problem is considered as inherently difficult if its solution requires significant resources (e.g. number of atomic instructions, number of gates in a circuit, number of memory registers etc) these growth functions allow to determine practical bounds on what machines can and cannot do in practice.

Turing Machines The Turing Machine (TM) is the most commonly accepted model for the study of decision problems in complexity theory. It is a theoretical device introduced by Alan Turing in 1936 [Tur36] and the standard computational model on which the decision problems' theory is based. A TM consists of a finite program attached to a reading or writing head moving on an infinite tape. The tape is divided into squares, each capable of storing one symbol from a finite alphabet `Alph` which includes a blank symbol `blank`. Each machine has a specified input alphabet `Alph`, which is a subset of `Alph`, without `blank`. At a given point during a computation the machine is in a state q which is in a specified finite set Q of possible states. At first, a finite input string over `Alph` is written on adjacent squares of the tape, all other squares are blank (contain `blank`), the head scans the left-most symbol of the input string, and the machine is in the initial state q_0. At each step, the machine is in a state q and the head is scanning a tape square containing a tape symbol s. The action performed depends on the pair (q, s) and is specified by the machine's transition function τ. The action consists of printing a symbol on the scanned square, moving the head left or right one square, and assuming a new state.

2. Also called *unconditionally secure*.
3. assuming that $P(x_i) \neq 0$
4. We assume that an attacker has access only to ciphertexts.

Decision Problems and Language.

A *decision problem* is a problem in a formal system whose answer has to be a "yes" or "no".

Decision problems frequently raise from mathematical questions of *decidability*, which attempts to find-out whether there exists an effective algorithm to determine the existence of some object or its membership to a set. However, the most important problems in mathematics are known to be *undecidable*. A decision problem can also be regarded as a *formal language*, where the members of the language are instances whose output is "yes" (we say that a computer accepts a language L if it accepts for all questions q with $q \in L$), "no" (if the machine rejects) or if the input causes the machine to run forever (in which case the instance is also considered as a non-member).

We further give the definitions of language and the complexity classes: $\mathcal{P}$, $\mathcal{NP}$ and $\mathcal{BPP}$. In the following, we will only consider time complexities.

Definition 2.25 (Language) *L is a function: $\{0,1\}^n \to \{0,1\}$. We define L as a language, if for all $w \in \{0,1\}^n$:*

$$\begin{cases} w \in L & \iff \quad L(w) = 1 \\ w \notin L & \iff \quad L(w) = 0 \end{cases}$$

Definition 2.26 (Complexity Class $\mathcal{P}$) *A decision problem Q belongs to the class $\mathcal{P}$ if there exists a polynomial-time algorithm able to solve Q, i.e. if given an input of length n, there exists an algorithm that produces the answer to Q in a number of steps polynomial in n.*

Definition 2.27 (Complexity Class $\mathcal{NP}$) *A decision problem Q belongs to the class $\mathcal{NP}$ if a "yes" instance[5] of the problem Q can be verified in polynomial time.*

Definition 2.28 (Complexity Class $\mathcal{BPP}$) *A decision problem Q belongs to class $\mathcal{BPP}$ if Q can be solved with 2-sided error on a probabilistic Turing machine in polynomial time.*

***$\mathcal{P}$ versus $\mathcal{NP}$* Problem.** The *$\mathcal{P}$ versus $\mathcal{NP}$* problem is central in complexity theory. This fundamental question is listed by the Clay Mathematics Institute as a major pending question because of the wide implications of a solution. This problem can be simply stated as follows: "*is $\mathcal{P} = \mathcal{NP}$?*". An answer to the $\mathcal{P} = \mathcal{NP}$ question would determine whether problems that can be verified in polynomial time, like the factoring problem, can also be solved in polynomial time. However, If $\mathcal{P} \neq \mathcal{NP}$ is proved, then that would mean that there are problems in $\mathcal{NP}$ (such as $\mathcal{NP}$-complete problems) that are harder to compute than to verify.

2.3 One-Way Functions

Informally, it is believed that a *one-way function* (OWF) is a function f that is easy to compute in polynomial time (by definition), but hard to invert (meaning that there cannot exist a probabilistic (or deterministic) machine that can invert f in polynomial time). Thus, the existence of a OWF implies that $\mathcal{P} \neq \mathcal{NP}$. However, as we have discussed in the previous section (2.2), in the current state of complexity theory (i.e., $\mathcal{P} = \mathcal{NP}$?) it still unknown whether $\mathcal{P} \neq \mathcal{NP}$ implies the existence of OWFs. For that reason even if some OWF candidates (described in 2.1) are known to be $\mathcal{NP}$-complete, this does not necessarily imply their one-wayness. A *trapdoor one-way function* (TOWF) is a kind of OWF for which the inverse direction is easy to compute given a certain private information (the trapdoor), but difficult without it. TOWFs are the basis of signature schemes and public-key cryptosystems but also imply the existence of many other useful primitives (including pseudorandom generators, pseudorandom function families, bit commitment schemes, message authentication codes).

We can formalize this notion as following:

Definition 2.29 (One-Way Function) *A function f: $\{0,1\}^n \to \{0,1\}^n$ is one-way function if and only if: $\exists$ PPT M such that $\forall x$ $M(x) = f(x)$ and $\forall$ non-uniform PPT A we have that: $Pr[A(f(x)) \in f^{-1}(f(x))] = neg(|x|) = neg(n)$. Here $neg(.)$ denotes the negligible function notion of definition 2.3.*

5. An instance for which the purported answer is yes.

Definition 2.30 (The Security of a Scheme) *Let t, ε be positive constants with $\varepsilon > 1$. We say that a scheme is (t, ε)-secure if every adversary $\mathcal{A}$ running for at most t time units succeeds in breaking the scheme with probability at most ε.*

Definition 2.31 (The Asymptotic Security of a Scheme) *A scheme p is secure if every PPT adversary succeeds in breaking p with only negligible probability.*

We now turn our attention to proving the security of cryptosystems. We first give an intuition on how these proofs have to be constructed (reductionist proofs) and then discuss the models in which we can prove the security of a scheme.

2.4 Provable Security

2.4.1 Theoretical Framework

What is Provable Security?

Decidability, mentioned before, has a direct impact on provable security. In computability theory, the halting problem is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running or continue to run forever. Turing proved in 1936 that a general algorithm to solve the halting problem for all possible program-input pairs cannot exist. i.e. Turing proved that the halting problem is undecidable over TMs. A direct consequence of this proof is that a general algorithm $\mathcal{S}$ deciding the security for all possible program-input pairs cannot exist.

Indeed, given a candidate program H for the halting problem it suffices to transform H in a program $S = H|V$ where V is any artificially created security vulnerability. Submit S and its desired input to $\mathcal{S}$. If $\mathcal{S}$ outputs "secure" this reveals that V is never activated (and hence H runs forever). If $\mathcal{S}$ outputs "insecure" this reveals that V is reached (and hence H halts). In other words a program security oracle $\mathcal{S}$ can be transformed into a halting oracle $\mathcal{H}$. Which is known not to exist (Turing). This establishes the theoretical impossibility to decide security in the general case. [6]

In the same vain, we it is also possible to prove that if OWFs exist deciding security is impossible: let V be a vulnerability as introduced before. Let f be a OWF, s a secret and $y = f(s)$. We require the bit-length q of f's input to be significantly shorter than f's output [7]. Craft the following program [8]:

$$S(x) = \text{if } f(x) = y \text{ then run } V$$

Deciding the security of $S(x)$ amounts to deciding if some "magic preimage value" s happens to hit the hardwired value y that triggers the vulnerability V. Which is, precisely, the (decisional version of the) problem of inverting an OWF.

Paradoxically, OWFs are crucial for cryptographic security while being a theoretical argument in disfavor of general program security (which is anyhow a battle lost in advance, given Turing's halting argument above).

In the past, the traditional approach for evaluating the security of a cryptosystem was to search for attacks and consider a scheme secure as long as there were no known attacks contradicting its security. However, this approach is insufficient because we can never be sure that an attack does not exist. Hence, the security of a cryptosystem can only be considered heuristic (at best), given the possibility that a yet unknown attack exists cannot be excluded [Abd11].

6. For the sake of accuracy (and honesty) the above argument is about programs that are either useless (run forever) or insecure. Nothing is being said here about *useful real-life programs* (those that are both secure and halting). The argument is nonetheless correct.
7. e.g. size$(s) = 500$ bits and size$(y) = 1000$ bits
8. We assume that S aborts if x is longer than q.

As mentioned in section 2.1 there are two main security definition classes: The first is *unconditional or information-theoretic security* also called *perfect secrecy*: These proofs guarantee security against "all" attackers (all-powerful or unbounded adversaries). This category of proofs typically relies on combinatorial or information theoretic arguments [Sha48]. However, these proofs has practical limitations, for instance in the one-time pad keys must be at least as long as the message and be used only once, which is impractical.

In modern cryptography it is usually impossible to prove that a scheme is protected against unlimited adversaries. This requires to relax the definition of security, so that the best we can hope is construct schemes that are secure against computationally bounded adversaries, i.e. adversaries running in *probabilistic polynomial time*. Such proofs typically use complexity-theoretic techniques such as those discussed in section 2.2.

This led Goldwasser and Micali [GMR85] to introduce the foundations of *provable security* thirty years ago: a conceptual breakthrough for which they received the Turing award in 2013. They introduced the second approach for proving cryptosystem security, known as *provable security*, this school of thought relates the security of a cryptographic scheme to precise mathematically assumptions. As discussed in section 2.1, these assumptions can either be the existence of OWFs or specific TOWFs (e.g. the factoring problem).

We can formally define this notion as follows:

Definition 2.32 (Provable Security) *The problem X of contradicting the security of some cryptographic primitive or protocol can be reduced to a problem Y believed to be hard if there is a constructive polynomial-time transformation R that takes any instance of X and maps it to an instance of Y. Thus any algorithm solving Y can be transformed into an algorithm solving X. Hence if $X \in \mathcal{C}$, and if X is reducible to Y, then $Y \in \mathcal{C}$, where $\mathcal{C}$ denotes some complexity class.*

There are two types of *computational security* proof techniques, the first is *formal methods*, which computer-verifies the security of a scheme. That makes it possible to verify the system's properties better than by empirical testing and thus increase the level of security assurance.

The second main security proof technique is the concept introduced by Goldwasser and Micali in 1982 [GM82], called *reductionist proofs*.

To prove that a cryptographic protocol P is secure with respect to a computational problem or primitive S, one needs:

— A security definition that captures the adversary's capabilities and goals
— A statement of the computational problems
— A reduction showing how to transform an adversary that breaks the security goals of the P into a solver of S.

Reductionist Security Proofs

The provable security methodology called *reduction* transforms the breaking of the target protocol P into the solving of some hard mathematical problem S. This methodology has been widely successful in many cryptographic proofs. The process is an interplay between three algorithms: P, S and a third algorithm R, called *reduction*, transforming the solving of P into the solving of S. Thereby any hypothetical attacker A breaking (solving) P can be turned into an algorithm A' solving S.

Although provable security can tell us when a scheme is secure, it does not necessarily say *how secure* a scheme really is [MSA+11]. To measure the way in which security varies, a variable k (security parameter) is needed to quantify the size of inputs. Both the resource requirements to break the cryptographic protocol and the probability with which the adversary manages to break security should be expressed in terms of k. For that, two main approaches exist:

— The concrete security approach usually called *practice-oriented provable security* introduced by Bellare and Rogaway in [BR93]. This class of attackers called *probabilistic time-t algorithms* aims to give a precise estimation of the attacker's computational workfactor and success probability.

— The asymptotic security defines a class of attackers called *probabilistic polynomial time algorithms* (polynomial in a security parameter k). In that case cryptosystem designers prove that the attacker's success probability (advantage) is *negligible* in k as defined in definition 2.3.

2.4.2 The Random Oracle Paradigm

As mentioned before, theoretical work (for example to obtain an efficient scheme secure under a well known computational assumptions) often seems to achieve provable security only at the cost of efficiency. This is why we frequently must work in some idealized model of computation.

One such model is the well known *Random Oracle Model* (ROM) which is a powerful tool formalized by Mihir Bellare and Phillip Rogaway (1993) [BR93] as a mathematical abstraction (a theoretical black box). The ROM was firstly used in rigorous cryptographic proofs of certain basic cryptographic primitives, such as Full Domain Hash signatures [BR96] and OAEP encryption [BR95]. In the model, the interaction with such an oracle is available to both honest parties and adversaries.

Loosely speaking, cryptographers usually resort to the ROM when a cryptographic hash function H fulfills the role of a black box that responds to a query for the hash value of a bit-string M by giving a random value. For each query the oracle makes an independent random choice, except that it keeps a record of its past responses *H(M)* and repeats the same response if *M* is queried again.

Note that no function computable by a finite algorithm that can implement a true random oracle (which by definition requires an infinite description given the input's unrestricted size). Despite this, the best we can expect is that our concrete hash function mimics the behavior of a random oracle model in practice.

Reductionist proofs without random oracle assumptions are said to be in the standard model.

2.5 Digital Signatures

In this section we will recall the problem of data authentication and integrity in the asymmetric (public-key) setting. Here a sender needs to be assured that a message comes from a legitimate sender (authentication) and not from an attacker. This also includes the assurance that the message was not modified during transmission (integrity). MACs solved this problem but for the symmetric-key setting. By opposition to MACs, digital signatures have the advantage of being publicly verifiable and non-repudiable. Public *verifiability* implies the *transferability* of signatures and, thus, signatures prove useful in many applications and in public-key infrastructures. *Non-repudiation* and verifiability make digital signatures particularly suitable for contract signing purposes. Contract signing is that we will be analyzed in further depth in Section 4.2.

In their famous paper Diffie and Hellman [DH76], didn't only introduce the concept of public key cryptography but also the fundamental notion of *digital signature scheme.* Digital signatures are extremely important in cryptography. This primitive allows to ascertain the origin of information and is, as such, vital to web security, digital transactions and identity application, to name just a few of the numerous applications of digital signatures. There exist several types of digital signatures. Before recalling the main algorithms we will introduce theoretical framework and the security notions necessary for the proper definition of digital signatures.

2.5.1 General Framework

Definition 2.33 (Digital Signature Scheme) *A digital signature scheme Σ is a 3-tuple of algorithms: (*KEYGEN, SIGN, VERIFY*). *KEYGEN *is probabilistic algorithm.* SIGN *is usually probabilistic, but may, in some cases, be deterministic.* VERIFY *is usually deterministic.*

— KEYGEN algorithm: Let k be the security parameter and let 1^k be the input of the key generation algorithm KEYGEN. KEYGEN outputs a pair (pk, sk) of public and secret keys.

— SIGN algorithm: Given a message m and (pk, sk), SIGN outputs a signature σ.
— VERIFY algorithm: Given σ, m, pk, VERIFY tests if σ is a valid signature of m with respect to pk

The correctness of a digital signature scheme is defined as follows.

Definition 2.34 (Correctness of a Signature Scheme) *Let $\mathcal{M}$ be a message space. A signature scheme is said correct if, for any message $m \in \mathcal{M}$ the following experiment:*

$$(sk, pk) \leftarrow \text{KEYGEN}(\lambda), \quad \sigma \leftarrow \text{SIGN}(sk, m), \quad b \leftarrow \text{VERIFY}(pk, m, \sigma)$$

is such that $b =$ `True` except with probability negligible in λ.

2.5.2 Some Examples

A number of well known digital signature algorithms will further be recalled schematically hereafter: RSA in Figure 2.1, ElGamal in Figure 2.2, Schnorr in Figure 2.3 and Girault-Poupard-Stern in Figure 2.4.

The reason why we chose to present the ElGamal signatures is not only historical. Both ElGamal and Girault-Poupard-Stern can serve to implement the construction described in Section 4.2.

Also, Schnorr signatures are of particular interest for the results of Section 4.2, where Schnorr co-signature for two parties is described and proved.

An important observation about digital signature schemes, due to Diffie and Hellman, is that any public-key encryption scheme for which $\mathcal{C} = \mathcal{M}$, can be used as a digital signature scheme.

RSA (Rivest-Shamir-Adleman)

RSA is probably the most popular signature scheme to date. RSA signature and encryption share the same key generation algorithm. The security of both relies on the FACT assumption (*cf.* Section 2.1). A variety of RSA-based signature schemes appeared over time. Despite its elegant mathematical structure, instantiating RSA is subtle as shown in [Mis98; CNS99; CND+06].

Parameters and Key Generation	
	Choose large primes p, q and compute $n = p \cdot q$
	Choose e such that $\gcd(e, \varphi(n)) = 1$,
	where $\varphi(\cdot)$ is Euler's totient function [9]
Private	d such that $e \cdot d = 1 \bmod \varphi(n)$
Public	n, e
Signing Algorithm (message m)	
	$\sigma \leftarrow m^d \bmod n$
	Output σ as the signature of m
Verification Algorithm [10]	
	if $m = \sigma^e \bmod n$ **then**
	return `True`
	else
	return `False`

Figure 2.1 – RSA signature algorithm.

10. Let a be a positive integer. Euler's totient function denoted above $\varphi(a)$ represents the number of positive integers b such that $1 \leq b \leq a$ and $\gcd(a, b) = 1$.
10. For simplicity, we do not consider redundancy check in m for this illustrative signature scheme.

ElGamal

ElGamal's digital signature and encryption algorithms were introduced in [El 84]. [El 84] didn't include the hashing step added in Figure 2.2. Pointcheval-Stern's version of ElGamal signatures is provably secure against adaptive chosen-message attacks [PS96; PS00] in the ROM.

Nonetheless, Bleichenbacher [Ble96] has shown how malicious parameters can be generated. Bleichenbacher's attack is also applicable to Pointcheval-Stern's version of ElGamal.

Parameters and Key Generation	
	Large prime p such that $p-1$ contains a large prime factor $g \in \mathbb{Z}_p^*$.
Private	$x \in_R \mathbb{Z}_p^*$
Public	$p, g, y \leftarrow g^x \bmod p$
Signing Algorithm (message m)	
	Pick $k \in_R \mathbb{Z}_p^*$ such that $\gcd(k, p-1) = 1$ $r \leftarrow g^k \bmod p$ $e \leftarrow H(m)$ $s \leftarrow e - xrk^{-1} \bmod p-1$ If $s = 0$ repeat signature generation. Output $\{r, s\}$ as the signature of m
Verification Algorithm	

	$e \leftarrow H(m)$
	If $0 < r < p$ or $0 < s < p-1$ then return `False`
	if $g^e = y^r \cdot r^s \bmod p$ **then**
	  return `True`
	else
	  return `False`

Figure 2.2 – ElGamal's signature algorithm.

Schnorr

Schnorr signatures [Sch90] are an offspring ElGamal signatures[El 84]. Schnorr signatures are provably secure in the Random Oracle Model under the DLP assumption [PS96].

Schnorr signatures will be discussed in further detail in Section 4.2.

Girault-Poupard-Stern (GPS)

Originally, the Girault-Poupard-Stern (GPS) scheme [GPS06] was developed as an identification scheme whose underlying hardness assumptions were DLP and FACT. Applying the Fiat-Shamir transformation [FS87], GPS can be turned into a digital signature scheme in a straightforward manner.

GPS was the only identification scheme submitted to the NESSIE competition [NES]. NESSIE ended with 17 selected algorithms (of the initial 42), amongst which was GPS. The main advantage of GPS over other DLP based schemes is that the prover has only one exponentiation and one addition to perform. The exponentiation can be precomputed before receiving the challenge. If this is done the prover does not need to perform any modular reductions after receiving the challenge from the verifier.

Parameters and Key Generation	
	Large primes p, q such that $q \geq 2^{\kappa}$, where
	κ is the security parameter and
	$p - 1 \bmod q = 0$
	$g \in \mathbb{G}$ (a cyclic group of prime order q)
Private	$x \in_R \mathbb{Z}_q^*$
Public	$y \leftarrow g^x$
Signing Algorithm (message m)	
	Pick $k \in_R \mathbb{Z}_q^*$
	$r \leftarrow g^k$
	$e \leftarrow H(m, r)$
	$s \leftarrow k - ex \bmod q$
	Output $\{r, s\}$ as the signature of m
Verification Algorithm	
	$e \leftarrow H(m, r)$
	if $g^s y^e = r$ **then**
	return `True`
	else
	return `False`

Figure 2.3 – Schnorr's signature algorithm.

GPS is a version of Schnorr's signature algorithm designed to reduce on-line computation and thus allow on-the-fly signatures. The parameters of the GPS scheme have to be chosen by taking into consideration the attack presented by van Oorschot and Wiener in [OW96].

2.5.3 Security Notions for Digital Signatures

Bellare and Rogaway introduced the concept of "practice-oriented provable security" in a thread of papers of which the first one was [BR94]. This concept naturally results from the convergence of theory (notably [GM84]) and practice.

As a direct consequence, the ROM (see in Section 2.4.2) was introduced in [BR93]. The ROM allowed to prove the security of many cryptographic schemes by abstracting away the random-like properties of hash functions.

Using the ROM, Pointcheval and Stern [PS00] present security arguments for a large class of digital signatures. From their work emerged an approach that proposes computational reductions to well established problems for proving the security of digital signature schemes.

Security Notions. An efficient adversary $\mathcal{A}$ is modeled as a PPT algorithm.

We say that an adversary $\mathcal{A}$ "forges a signature on a new message" or "outputs a forgery" whenever $\mathcal{A}$ outputs a message/signature pair (m, σ) such that $\textsc{Verify}(pk, m, \sigma) = \texttt{True}$ and $\mathcal{A}$ was not previously given any signature on m. We say that $\mathcal{A}$ "outputs a strong forgery" whenever $\mathcal{A}$ outputs a message/signature pair (m, σ) such that $\textsc{Verify}(pk, m, \sigma) = \texttt{True}$ and $\mathcal{A}$ was not previously given the signature σ on the message m. Note that whenever $\mathcal{A}$ outputs a forgery then $\mathcal{A}$ also outputs a strong forgery.

Let Σ denote a signature scheme as defined in Section 2.5.1.

Parameters and Key Generation	
	$A, B, S \in \mathbb{N}$ s.t. $\lvert A\rvert \geq \lvert S\rvert + \lvert B\rvert + 80, \lvert B\rvert = 32, \lvert S\rvert > 140$
	κ is the security parameter and
	p, q primes and $n = pq$
	$g \in_R \mathbb{Z}_n$ s.t. $\gcd(g, n) = 1$.
Private	$s \in [1, S]$
Public	$I = g^s \bmod n$
Signing Algorithm (message m)	
	Pick $r \in_R [0, A-1]$
	$x \leftarrow g^r \bmod n$
	$c \leftarrow H(m, x)$
	$y \leftarrow r + c \cdot s$
	Output $\{c, y\}$ as the signature of m
Verification Algorithm	
	if $0 < c < B-1$ or $0 < y < A + (B-1) \cdot (S-1) - 1$
	return `False`
	Compute $c' \leftarrow H(m, g^y / I^c \bmod n)$
	if $c' = c$ **then**
	return `True`
	else
	return `False`

Figure 2.4 – Girault-Poupard-Stern's signature algorithm.

Definition 2.35 (EU-RMA Security) *A signature scheme Σ is* **existentially unforgeable under a random-message attack** *(EU-RMA) if for all polynomials p and all PPT adversaries $\mathcal{A}$, the success probability of $\mathcal{A}$ in the following experiment is negligible, as a function of k:*

1. *A sequence of $p = p(k)$ messages $m_1, ..., m_p$ are chosen uniformly at random from the message space*
2. *KeyGen(1^k) is run to obtain a key-pair (pk, sk)*
3. *Signature $\sigma_1 \leftarrow$ Sign$(sk, m_1), ..., \sigma_p \leftarrow$ Sign(sk, m_p) are computed*
4. *$\mathcal{A}$ is given pk and $\{(m_i, \sigma_i)\}_{i=1}^p$ and outputs (m, σ)*
5. *$\mathcal{A}$ succeeds if* Verify$(pk, m, \sigma) =$ `True` *and $m \notin (m_1, ..., m_p)$*

Σ is **strongly unforgeable under a random-message attack** (SU-RMA) if Σ complies with Definition 2.35 where step 5 is replaced by

5. $\mathcal{A}$ succeeds if Verify$(pk, m, \sigma) =$ `True` and $(m, \sigma) \notin \{(m_1, \sigma_1), ..., (m_p, \sigma_p)\}$

Definition 2.36 (EU-KMA Security) *A signature scheme Σ is* **existentially unforgeable under a known-message attack** *(EU-KMA) if for all polynomials p and all PPT adversaries $\mathcal{A}$, the success probability of $\mathcal{A}$ in the following experiment is negligible as a function of k:*

1. *$\mathcal{A}(1^k)$ outputs a sequence of $p = p(k)$ messages $m_1, ..., m_p$*
2. *KeyGen(1^k) is run to obtain a key-pair (pk, sk)*
3. *Signature $\sigma_1 \leftarrow$ Sign$(sk, m_1), ..., \sigma_p \leftarrow$ Sign(sk, m_p) are computed*
4. *$\mathcal{A}$ is given pk and $\{\sigma_i\}_{i=1}^p$ and outputs (m, σ)*
5. *$\mathcal{A}$ succeeds if* Verify$(pk, m, \sigma) =$ `True` *and $m \notin (m_1, ..., m_p)$*

We assume that $\mathcal{A}$ is a stateful algorithm, and in particular is allowed to maintain state between steps 1 and 4.

Σ is **strongly unforgeable under a known-message attack** (SU-KMA) if Σ complies with Definition 2.36 where step 5 is replaced by

5. $\mathcal{A}$ succeeds if Verify$(pk, m, \sigma) =$ `True` and $(m, \sigma) \notin \{(m_1, \sigma_1), ..., (m_p, \sigma_p)\}$

Definition 2.37 (EU-CMA Security) *A signature scheme Σ is* **existentially unforgeable under a chosen-message attack** *(EU-CMA) if for all PPT adversaries $\mathcal{A}$, the success probability of $\mathcal{A}$ in the following experiment is negligible as a function of k:*

1. $\text{KEYGEN}(1^k)$ *is run to obtain a key-pair (pk, sk)*
2. *$\mathcal{A}$ is given pk and allowed to interact with a signing oracle $\text{SIGN}(sk, \cdot)$, requesting signatures on as many messages as it likes. Let this attack be denoted as $\mathcal{A}_{pk}^{\text{SIGN}(\cdot)}$*
3. *Eventually, $\mathcal{A}$ outputs (m, σ)*
4. *$\mathcal{A}$ succeeds if $\text{VERIFY}(pk, m, \sigma) = \texttt{True}$ and $m \notin \mathcal{M}$*

The desired security notion for signature scheme is strong unforgeability under chosen message attack (SU-CMA). This notion is defined by the next security game:

Definition 2.38 (SU-CMA Security Game and Advantage) *The SU-CMA security game $G_{\Sigma}^{\text{SU-CMA}}$ is defined as a protocol between the challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:*

1. *$\mathcal{C}$ runs $(sk, pk) \leftarrow \text{KEYGEN}(\lambda)$ and sends pk to $\mathcal{A}$*
2. *$\mathcal{A}$ adaptively chooses messages $m_1 \dots m_k$ and sends them to $\mathcal{C}$*
3. *$\mathcal{C}$ responds to each message with the signature $\sigma_i = \text{SIGN}(pk, sk, m)$*
4. *The adversary sends a message and a forgery (m^*, σ^*) to $\mathcal{C}$*
5. *$\mathcal{C}$ outputs*

$$\begin{cases} 1 & \text{if } \text{VERIFY}(pk, m^*, \sigma^*) = \texttt{True} \text{ and if } (m^*, \sigma^*) \neq (m_i, \sigma_i) \text{ for all } i = 1 \dots k \\ 0 & \text{otherwise} \end{cases}$$

The advantage of an SU-CMA adversary $\mathcal{A}$ against the signature scheme Σ is defined as:

$$\text{Adv}_{\Sigma}^{\text{SU-CMA}}(\mathcal{A}) = \Pr\left[\text{Exp}_{\Sigma}^{\text{SU-CMA}}(\mathcal{A}) = 1\right]$$

Definition 2.39 (SU-CMA Security) *A signature scheme Σ is said to be SU-CMA secure if for any adversary $\mathcal{A}$ that runs in probabilistic polynomial time (PPT) in the security parameter λ, the adversary's advantage $\text{Adv}_{\Sigma}^{\text{SU-CMA}}(\mathcal{A})$ is negligible in λ.*

Observations. CMA security can be attained from weaker primitives. Such constructions, exceeding the purpose of this thesis, are presented in detail in [YK05].

2.6 Public-Key Cryptography

Since the introduction of public-key cryptography in the late 1970's [DH76], many candidate constructions were proposed based on the assumption that some mathematical problem is hard to solve.

Prime example of hard problems considered in the literature include: Factorizing a large number (FACT) [Rab79]; Finding the e-th root modulo (ERP) a composite n of arbitrary numbers, when n is the product of two large primes [RSA78]; Solving a large enough instance of the knapsack problem [NS97], [Mer78]; Finding the discrete logarithm (DLP) in a group of large prime order [DH76; Kob87; Mil86].

However not all these candidates survived the test of time: Merkle and Hellman's knapsack-based cryptosystem [Mer78], for instance, was broken by Shamir [SD82] using a now-classical lattice reduction algorithm [Len84].

Looking back on the history of public-key cryptography [Kob87] it seems that almost every new cryptosystem comes with its cortege of new assumptions. As observed by Naor [NY90], Goldwasser and Kalai [GK15], we face an increase in the number of new assumptions, which are often complex to define, difficult to interpret, and at times hard to untangle from the constructions which utilize them. While being instrumental to progress and a better understanding of the field, new assumptions often shed doubt on the real security of the schemes building on them.

2.6.1 General Framework

Consider a plaintext space $\mathcal{M} = \{0,1\}^m$, a key space $\mathcal{K} = \{0,1\}^k$ and a ciphertext space $\mathcal{C} = \{0,1\}^c$. Recall that a public-key cryptosystem is usually defined as follows:

— $Setup(k)$ takes as input a security parameter k and outputs public parameters pp;
— $KeyGen(pp) \in \mathcal{K}^2$ takes as input pp, and outputs a pair (sk, pk) of secret and public keys – we make the assumption that sk contains pk;
— $Encrypt(pk, m) \in \mathcal{C}$ takes as input pk and a message m, and outputs a ciphertext c;
— $Decrypt(sk, c) \in \{\mathcal{M}, \perp\}$ takes as input sk and c, and outputs m or $\perp$.

2.6.2 Security Notions for Public-Key Cryptography

In the following definitions, IND will denote indistinguishability and NM will denote non-malleability. IND was initially defined by Goldwasser and Micali [GM84], and NM by Dolev, Dwork and Naor [DDN00]. The NM notion will not be detailed or formally defined in this manuscript. We refer the reader to [BDJR97] for a precise description of this concept.

For defining IND we follow the description given in [BDPR98].

Experiments. Let A be a probabilistic algorithm and denote by $A(x_1, x_2, \ldots; r)$ is the result of running A on inputs $x_1, x_2, \ldots$ and coins r.

We let $y \leftarrow A(x_1, x_2, \ldots)$ denote the experiment of picking r at random and let $y = A(x_1, x_2, \ldots; r)$.

We say that y is a *potential output* of $A(x_1, x_2, \ldots)$ if $\exists r$ such that $A(x_1, x_2, \ldots; r) = y$.

IND. A public key encryption scheme $\mathcal{PKE}$ satisfies the property IND if the distributions A_{m_1} and A_{m_2} are *computationally indistinguishable*[11] for all $m_1, m_2 \in \mathcal{M}$ such that $|m_1| = |m_2|$ where

$$A_{m_i} = \left\{pk, \mathcal{PKE}.\textsc{Encrypt}(pk, m_i) : (pk, sk) \xleftarrow{\$} \mathcal{PKE}.\textsc{KeyGen}(\lambda)\right\}, \text{ for } i \in \{1,2\}.$$

The commonly desired security properties of public-key encryption are *indistinguishability under chosen plaintext attack* (IND-CPA) or semantic security, *indistinguishability under chosen ciphertext attack* (IND-CCA1) and *indistinguishability under adaptive ciphertext attack* (IND-CCA2) defined by the security games of Definitions 2.40, 2.42 and 2.44. Weaker security notions whose presentations we omit here are *one-wayness under chosen plaintext attack* and *under chosen ciphertext attack*, (OW-CPA and OW-CCA). We refer the reader to [Poi05] for a detail description of OW-CPA and OW-CCA.

Definition 2.40 (IND-CPA Security Game and Advantage) *The* IND-CPA *security game* $G^{\text{IND-CPA}}_{\mathcal{PKE}}$ *is defined as a protocol between the challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*:*

1. $\mathcal{C}$ *runs* $(sk, pk) \leftarrow \mathcal{PKE}.\textsc{KeyGen}(\lambda)$ *and sends* pk *to* $\mathcal{A}$
2. $\mathcal{A}$ *chooses two messages* m_0 *and* m_1 *and sends them to* $\mathcal{C}$
3. $\mathcal{C}$ *chooses a uniform random bit* b *and encrypts one of the two message accordingly:*

$$c \leftarrow \mathcal{PKE}.\textsc{Encrypt}(pk, m_b)$$

4. $\mathcal{A}$ *sends a guess* b' *to* $\mathcal{C}$
5. $\mathcal{C}$ *outputs* 1 *if the guess was correct, that is if* $b = b'$*,* 0 *otherwise*

The advantage of an IND-CPA *adversary* $\mathcal{A}$ *against this game is defined as:*

$$\text{Adv}^{\text{IND-CPA}}_{\mathcal{PKE}}(\mathcal{A}) = \Pr\left[G^{\text{IND-CPA}}_{\mathcal{PKE}}(\mathcal{A}) = 1\right] - \frac{1}{2}$$

11. Two probabilities are *computationally indistinguishable* if no efficient algorithm can make the difference between them.

Definition 2.41 (IND-CPA Security) *A public-key encryption scheme $\mathcal{PKE}$ is said to be* IND-CPA *secure if for any adversary $\mathcal{A}$ that runs in probabilistic polynomial time (PPT) in the security parameter λ, $\mathcal{A}$'s advantage $\mathrm{Adv}^{\text{IND-CPA}}_{\mathcal{PKE}}(\mathcal{A})$ is negligible in λ.*

Definition 2.42 (IND-CCA1 Security Game and Advantage) *The* IND-CCA1 *security game $G^{\text{IND-CCA1}}_{\mathcal{PKE}}$ is defined as a protocol between the challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:*

1. *$\mathcal{C}$ runs $(sk, pk) \leftarrow \mathcal{PKE}.\textsc{KeyGen}(\lambda)$ and sends pk to $\mathcal{A}$*
2. *$\mathcal{A}$ may perform polynomially many encryptions, calls to the decryption oracle based on arbitrary ciphertexts, or other operations*
3. *Eventually, $\mathcal{A}$ submits two distinct chosen plaintexts m_0 and m_1 to $\mathcal{C}$.*
4. *$\mathcal{C}$ chooses a random bit b and encrypts one of the two message accordingly:*

$$c \leftarrow \mathcal{PKE}.\textsc{Encrypt}(pk, m_b)$$

5. *$\mathcal{A}$ may* **not** [12] *make further calls to the decryption oracle*
6. *$\mathcal{A}$ sends a guess b' to $\mathcal{C}$*
7. *$\mathcal{C}$ outputs 1 if the guess was correct, that is if $b = b'$, 0 otherwise*

The advantage of an IND-CCA1 *adversary $\mathcal{A}$ against this game is defined as:*

$$\mathrm{Adv}^{\text{IND-CCA1}}_{\mathcal{PKE}}(\mathcal{A}) = \Pr\left[G^{\text{IND-CCA1}}_{\mathcal{PKE}}(\mathcal{A}) = 1\right] - \frac{1}{2}$$

Definition 2.43 (IND-CCA1 Security) *A public-key encryption scheme $\mathcal{PKE}$ is said to be* IND-CCA1 *secure if for any adversary $\mathcal{A}$ that runs in probabilistic polynomial time (PPT) in the security parameter λ, $\mathcal{A}$'s advantage $\mathrm{Adv}^{\text{IND-CCA1}}_{\mathcal{PKE}}(\mathcal{A})$ is negligible in λ.*

Definition 2.44 (IND-CCA2 Security Game and Advantage) *The* IND-CCA2 *security game $G^{\text{IND-CCA2}}_{\mathcal{PKE}}$ is defined as a protocol between the challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:*

1. *$\mathcal{C}$ runs $(sk, pk) \leftarrow \mathcal{PKE}.\textsc{KeyGen}(\lambda)$ and sends pk to $\mathcal{A}$*
2. *$\mathcal{A}$ may perform polynomially many encryptions, calls to the decryption oracle based on arbitrary ciphertexts, or other operations*
3. *Eventually, $\mathcal{A}$ submits two distinct chosen plaintexts m_0 and m_1 to $\mathcal{C}$*
4. *$\mathcal{C}$ chooses a uniform random bit b and encrypts one of the two message accordingly:*

$$c \leftarrow \mathcal{PKE}.\textsc{Encrypt}(pk, m_b)$$

5. *$\mathcal{A}$ may make further calls to the encryption or decryption oracles, but may not submit the challenge ciphertext c to $\mathcal{C}$*
6. *$\mathcal{A}$ sends a guess b' to $\mathcal{C}$*
7. *$\mathcal{C}$ outputs 1 if the guess was correct, that is if $b = b'$, 0 otherwise*

The advantage of an IND-CCA2 *adversary $\mathcal{A}$ against this game is defined as:*

$$\mathrm{Adv}^{\text{IND-CCA2}}_{\mathcal{PKE}}(\mathcal{A}) = \Pr\left[G^{\text{IND-CCA2}}_{\mathcal{PKE}}(\mathcal{A}) = 1\right] - \frac{1}{2}$$

Definition 2.45 (IND-CCA2 Security) *A public-key encryption scheme $\mathcal{PKE}$ is said to be* IND-CCA2 *secure if for any adversary $\mathcal{A}$ that runs in probabilistic polynomial time (PPT) in the security parameter λ, its advantage $\mathrm{Adv}^{\text{IND-CCA2}}_{\mathcal{PKE}}(\mathcal{A})$ is negligible in λ.*

The implication relations between the above security notions are given in Figure 2.5 and [BDPR98].

12. Step 5 stresses the difference between IND-CCA1 and IND-CCA2. Thus, we underline that for IND-CCA1, $\mathcal{A}$ will not be allowed to interact with the decryption oracle after step 4.

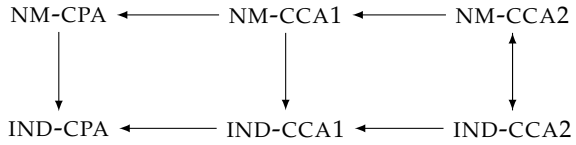


Figure 2.5 – Relations between public-key security notions.

2.7 Proof Systems

In mathematics and in cryptography, entities need to prove assertions (or statements) to each other. Various types of probabilistic proof systems emerged during the development of computer science over the last decade. The question raised revolved around *information leakage* i.e. how much extra information is the verifier going to learn from a proof, beyond the statement's truthfulness. We say that an encryption scheme is secure if the ciphertext does not allow the eavesdropper to compute any new (efficiently computable) function of the plaintext beyond what she could have computed without the ciphertext.

Very informally, a zero-knowledge proof allows one party, called *prover* ($\mathcal{P}$), to convince another party, called *verifier* ($\mathcal{V}$), that $\mathcal{P}$ knows some fact (a secret, a proof of a theorem, etc.) without revealing to $\mathcal{V}$ any information about that fact beyond its veracity. For example, The proof that $n = 670592745 = 12345 \cdot 54321$ is not a zero-knowledge proof that n is a composite integer.

This concern was first studied in the 1985 by Goldwasser, Micali and Rackoff [GMR85] who proposed a new notion called *zero knowledge proof* that became fundamental in modern cryptography.

In the following paragraphs we present the underlying ideas of zero-knowledge proof protocols, their history and their importance in modern cryptography.

2.7.1 Interactive Proofs

The notion of interactive proofs [13] was separately introduced by Babai [Bab86] and Goldwasser, Micali and Rackoff [GMR85]. An Interactive Proof System (IPS) for a language L, is a protocol between $\mathcal{P}$ and $\mathcal{V}$ where:

— $\mathcal{P}$ (all powerful) and $\mathcal{V}$ (PPT and assumed always honest) are given an input x.
— Through a protocol, $\mathcal{P}$ tries to convince to $\mathcal{V}$ that $x \in L$.
— At the end of the protocol, $\mathcal{V}$ outputs either "accept" if the proof is satisfactory or "reject" if not.

All interactive proof systems must comply with two requirements:
— *Completeness:* If the statement is true (i.e. $x \in L$) then a honest $\mathcal{P}$ is always able to convince $\mathcal{V}$ of this fact.
— *Soundness:* Otherwise, (i.e. if the statement is false) no $\mathcal{P}$ should not be able to convince $\mathcal{V}$ to output "accept" with probability higher than $\frac{1}{2}$.

2.7.2 Zero-Knowledge Proofs

In general, providing a proof results in giving-out new information or new knowledge [14].

As counter-intuitive as it may seem, Goldwasser, Micali and Rackoff [GMR85] showed that it is possible to prove statements without leaking information beyond the veracity of those statements (i.e. that $x \in L$). The conceptual tool allowing to do so is the *zero-knowledge proof* (ZKP).

13. Also called Arthur-Merlin proof systems, where Arthur represents the verifier $\mathcal{V}$ and Merlin represents the prover $\mathcal{P}$.
14. In the example given before, not only did the proof establish that 670592745 is composite, it also revealed its factors.

Using a ZKP, $\mathcal{P}$ will convince $\mathcal{V}$ that a statement S (theorem, secret, etc) is true. This will be done in a way that does not allow $\mathcal{V}$ to learn anything from the interaction with $\mathcal{P}$. [CGMW97] later showed that ZKPs can be constructed for any language in $\mathcal{NP}$ under the assumption that OWFs exist.

2.7.3 Applications

ZKPs proved to be very useful both in complexity theory and in cryptography.

In complexity theory, Boppana, Håstad, and Zachos [Rav87], show that ZKPs provide a means to convince ourselves that certain languages are not $\mathcal{NP}$-complete.

In cryptography, ZKPs were first motivated by the need to authenticate entities – which is a cornerstone of information security. ZKPs were then used to enforce honest behavior while maintaining privacy [15] during secure multiparty computation: a setting where several parties wish to securely compute some joint function of their private inputs [CGMW97].

2.7.4 Zero-Knowledge Proofs of Knowledge

In 1988 Feige, Fiat and Shamir [FFS88] went one step further ahead and noted that the intuition that $\mathcal{V}$ learns nothing from $\mathcal{P}$ isn't true. Indeed, the [GMR85] ZKP notion is somewhat misleading given that $\mathcal{P}$ reveals *one more bit of knowledge* to $\mathcal{V}$ (namely that $x \in L$). Therefore, [FFS88] introduced a novel form of proof systems called "knowledge about knowledge" or "Zero-knowledge proofs of knowledge" (ZKPPs). In ZKPPs the role of $\mathcal{P}$ is not to prove to $\mathcal{V}$ that $x \in L$, but to prove that he knows the member status of x with respect to L. Loosely speaking, a ZKPP guarantees that whenever $\mathcal{V}$ is convinced that $\mathcal{P}$ knows an information X, this X can be efficiently extracted from $\mathcal{P}$'s strategy. One natural application of ZKPPs is entity identification.

2.7.5 Non-Interactive Zero-Knowledge Proofs

A last noteworthy notion in this area is that of non-interactive ZKPs (NIZKPs) introduced by Blum, Feldman, and Micali in [BFM88]. A NIZKPs is functionally very close to a digital signature. Here $\mathcal{P}$ sends to $\mathcal{V}$ only one message which is, in a way, a sort of "frozen" ZKP verifiable offline without $\mathcal{P}$'s help. This allows to reduce interaction, which is desirable in many applications. [BFM88] present NIZKPs within the Common Reference String model, (CRS). This model captures the assumption that a trusted setup in which all involved parties get access to the same string CRS taken from some distribution D. exists. Schemes proven secure in the CRS model are secure given that the setup was performed correctly.

15. i.e. ensure that parties in a cryptographic protocol behave as they should.

CHAPTER 3

RESULTS & CONTRIBUTIONS

3.1 Thesis Results

The main results of this thesis are presented in this section. Nearly all those results were published as journal papers, conference papers or Internet pre-prints.

3.1.1 Fairness & Attestation in Cryptographic Protocols

Non-Interactive Attestations for Arbitrary RSA Prime Generation Algorithms

with Fabrice Ben Hamouda, Rémi Géraud and David Naccache

Abstract. RSA public keys are central to many cryptographic applications; hence their validity is of primary concern to the scrupulous cryptographer. The most relevant properties of an RSA public key (n, e) depend on the *factors* of n: are they properly generated primes? are they large enough? is e co-prime with $\phi(n)$? etc. And of course, it is out of question to reveal n's factors.

Generic non-interactive zero-knowledge (NIZK) proofs can be used to prove such properties. However, NIZK proofs are unpractical.

For moduli with some very specific properties, specialized proofs exist but such *ad hoc* proofs are hard to generalize or extend.

This paper proposes a new type of *general-purpose* compact non-interactive proofs, called *attestations*, allowing the key generator to convince any third party that n was properly generated.

The proposed construction applies to *any* prime generation algorithm, and is provably secure in the Random Oracle Model.

As a typical implementation instance, for a 138-bit security, verifying or generating an attestation requires $k = 1024$ prime generations. For this instance, each processed message will later need to be signed or encrypted 14 times by the final users of the attested moduli.

Note. Currently submitted. Section 4.1 of this thesis.

Legally Fair Contract Signing without Keystones

with Diana Maimuţ, Rémi Géraud, David Naccache and David Pointcheval

Abstract. In two-party computation, achieving both fairness and guaranteed output delivery is well known to be impossible. Despite this limitation, many approaches provide solutions of practical interest by weakening somewhat the fairness requirement. Such approaches fall roughly in three categories: "gradual release" schemes assume that the aggrieved party can eventually reconstruct the missing information; "optimistic schemes" assume a trusted third party arbitrator that can restore fairness in case of litigation; and "concurrent" or "legally fair" schemes in which a breach of fairness is compensated by the aggrieved party having a digitally signed cheque from the other party (called the keystone).

In this paper we describe and analyze a new contract signing paradigm that doesn't require keystones to achieve legal fairness, and give a concrete construction based on Schnorr signatures which is compatible with standard Schnorr signatures and provably secure.

Note. Published in the proceedings of *Applied Cryptography and Network Security (ACNS) 2016* [FGM+16a; FGM+16b]. Section 4.2 of this thesis.

3.1.2 Zero-Knowledge Proof Systems & Authentication Protocols

Slow Motion Zero Knowledge – Identifying With Colliding Commitments

with Rémi Géraud and David Naccache

Abstract. Discrete-logarithm authentication protocols are known to present two interesting features: The first is that the prover's commitment, $x = g^r$, claims most of the prover's computational effort. The second is that x does not depend on the challenge and can hence be computed in advance. Provers exploit this feature by pre-loading (or pre-computing) ready to use commitment pairs r_i, x_i. The r_i can be derived from a common seed but storing each x_i still requires 160 to 256 bits when implementing DSA or Schnorr.

This paper proposes a new concept called *slow motion zero-knowledge* (SM-ZK). SM-ZK allows the prover to slash commitment size (by a factor of 4 to 6) by combining classical zero-knowledge and a timing side-channel. We pay the conceptual price of requiring the ability to measure time but, in exchange, obtain communication-efficient protocols.

Note. Published in the proceedings of *the 12th International Conference on Information Security and Cryptology (INSCRYPT) 2016* [FGN16b; FGN16a]. Section 5.1 of this thesis.

Thrifty Zero-Knowledge: When Linear Programming Meets Cryptography

with Simon Cogliani, Rémi Géraud and David Naccache

Abstract. We introduce "thrifty" zero-knowledge protocols, or TZK. These protocols are constructed by introducing a bias in the challenge send by the prover. This bias is chosen so as to maximize the security versus effort trade-off. We illustrate the benefits of this approach on several well-known zero-knowledge protocols.

Note. To appear in the proceedings of *the 12th International Conference on Information Security Practice and Experience (ISPEC) 2016* [CFGN16b; CFGN16c]. Presented as well at the NATO Workshop on Secure Implementation of PQC 2016. Section 5.2 of this thesis.

Public-Key Based Lightweight Swarm Authentication

with Simon Cogliani, Bao Feng, Rémi Géraud, Diana Maimuţ, David Naccache, Rodrigo Portella do Canto and Guilin Wang

Abstract. We describe a lightweight algorithm performing whole-network authentication in a distributed way. This protocol is more efficient than one-to-one node authentication: it results in less communication, less computation, and overall lower energy consumption.

The security of the proposed algorithm can be reduced to the RSA hardness assumption, and it achieves zero-knowledge authentication of a network in a time logarithmic in the number of nodes.

Note. Currently submitted [CFH+16]. Section 5.3 of this thesis.

When Organized Crime Applies Academic Results

with Rémi Géraud and David Naccache and Assia Tria

Abstract. This paper describes the forensic analysis of what the authors believe to be the most sophisticated smart card fraud encountered to date. In 2010, Murdoch *et al.* [MDAB10] described a man-in-the-middle attack against EMV cards. [MDAB10] demonstrated the attack using a general purpose FPGA board, noting that "*miniaturization is mostly a mechanical challenge, and well within the expertise of criminal gangs*". This indeed happened in 2011, when about 40 sophisticated card forgeries surfaced in the field.

These forgeries are remarkable in that they embed two chips wired top-to-tail. The first chip is clipped from a genuine stolen card. The second chip plays the role of the man-in-the-middle and communicates directly with the point of sale (PoS) terminal. The entire assembly is embedded in the plastic body of yet another stolen card.

The forensic analysis relied on X-ray chip imaging, side-channel analysis, protocol analysis, and microscopic optical inspections.

Note. Published in the *Journal of Cryptographic Engineering (JCEN) 2015* [FGNT15; HRDA16]. Presented at the ENS' Cryptography seminar, KU Leuven's seminar and INRIA Prosecco's research seminar. Section 5.4 of this thesis. This result received extensive media coverage:

Website	Shortened URL
wired.com	http://tinyurl.com/hlf49na
uk.businessinsider.com	http://tinyurl.com/gn2rmru
csoonline.com	http://tinyurl.com/zjfzdt3
ibtimes.co.uk	http://tinyurl.com/j9tb4oj
arstechnica.com	http://tinyurl.com/pfuy2am
finextra.com	http://tinyurl.com/zqqclft
pymnts.com	http://tinyurl.com/h33elxp
cuinsight.com	http://tinyurl.com/hrusgq3
securityaffairs.co	http://tinyurl.com/hjgbcmz
tripwire.com	http://tinyurl.com/pnuxhy9

3.1.3 Exploring Interactions Between Natural Language, Vision & Encryption

Human Public-Key Encryption

with Rémi Géraud and David Naccache

Abstract. This paper proposes a public-key cryptosystem and a short password encryption mode, where traditional hardness assumptions are replaced by specific refinements of the CAPTCHA concept called Decisional and Existential CAPTCHAs.

The public-key encryption method, achieving 128-bit security, typically requires from the sender to solve one CAPTCHA. The receiver does not need to resort to any human aid. A second symmetric encryption method allows to encrypt messages using very short passwords shared between the sender and the receiver.

Here, a simple 5-character alphanumeric password provides sufficient security for all practical purposes. We conjecture that the automatic construction of Decisional and Existential CAPTCHAs is possible and provide candidate ideas for their implementation.

Note. To appear in the proceedings of Mycrypt 2016 [HRN16]: Paradigm-shifting Crypto. Section 6.1 of this thesis.

Honey Encryption for Language: Robbing Shannon to Pay Turing?

with Marc Beunardeau, Rémi Géraud and David Naccache

Abstract. Honey Encryption (HE), introduced by Juels and Ristenpart (Eurocrypt 2014, [JR14]), is an encryption paradigm designed to produce ciphertexts yielding plausible-looking but bogus plaintexts upon decryption with wrong keys. Thus brute-force attackers need to use additional information to determine whether they indeed found the correct key.

At the end of their paper, Juels and Ristenpart leave as an open question the adaptation of honey encryption to natural language messages. A recent paper by Chatterjee *et al.* [CBJR15] takes a mild attempt at the challenge and constructs a natural language honey encryption scheme relying on simple models for passwords.

In this paper we explain why this approach cannot be extended to reasonable-size human-written documents *e.g.* e-mails. We propose an alternative solution and evaluate its security.

Note: To appear in the proceedings of Mycrypt 2016 [MRN16]: Paradigm-shifting Crypto. Section 6.2 of this thesis.

3.1.4 Generalization & Applications of Hierarchical Identity-Based Encryption (HIBE)

Compact CCA2-Secure Hierarchical ID-Based Broadcast Encryption with Public Ciphertext Test

with Weiran Liu, Jianwei Liu, Qianhong Wu, Bo Qin and David Naccache

Abstract. This paper generalizes the concept of Hierarchical Identity-Based Encryption (HIBE) by proposing a new primitive called Hierarchical Identity-Based Broadcast Encryption (HIBBE). Similar to HIBE, HIBBE organizes users in a tree-like structure and users can delegate their decryption capability to their subordinates, which mirrors real-world hierarchical social organizations. Unlike HIBE merely allowing a single decryption path, HIBBE enables encryption to any subset of the users and only the intended users (and their supervisors) can decrypt. We define Ciphertext Indistinguishability against Adaptively Chosen-Identity-Vector-Set and Chosen-Ciphertext Attack (IND-CIVS-CCA2) which capture the most powerful attacks on HIBBE in the real world. We achieve this goal in the standard model in two steps. We first construct an efficient HIBBE Scheme (HIBBES) against Adaptively Chosen-Identity-Vector-Set and Chosen-Plaintext Attack (IND-CIVS-CPA) in which the attacker is not allowed to query the decryption oracle. Then we convert it into an IND-CIVS-CCA2 scheme at only a marginal cost, i.e., merely adding one on-the-fly dummy user at the first depth of hierarchy in the basic scheme without requiring any other cryptographic primitives. Furthermore, our CCA2-secure scheme natively allows public ciphertext validity test, which is a useful property when a CCA2-secure HIBBES is used to design advanced protocols.

Note: To appear in *Information Sciences 2016* (accepted). Published as an IACR ePrint [LLW+16b]. Section 6.3 of this thesis.

Improving Thomlinson-Walker's Software Patching Scheme Using Standard Cryptographic and Statistical Tools

with Michel Abdalla, Hervé Chabanne, Julien Jainski and David Naccache

Abstract. This talk will illustrate how standard cryptographic techniques can be applied to real-life security products and services. This article presents in detail one of the examples given in the talk. It is intended to help the audience follow that part of our presentation. We chose as a characteristic example a little noticed yet ingenious Microsoft patent by Thomlinson and Walker. The Thomlinson-Walker system distributes encrypted patches to avoid reverse engineering by opponents (who would then be able to launch attacks on unpatched users). When the proportion of users who downloaded the encrypted patch becomes big enough, the decryption key is disclosed and all users install the patch.

Note. Published in the proceedings of *the 10th International Conference on Information Security Practice and Experience (ISPEC) 2014* [MHH+14]. Section 6.4 of this thesis.

3.2 Additional Results

Several other papers, more related to the areas of information security and efficient implementations [GHNK16], [CFN13], [CFGN16a], [AAF+15; AAF+16], [FGM+15b; HRD+16], [FGM+15a], [BBC+14a; BBC+14b], [BFG+14; BFG+16] are not part of this thesis.

3.2.1 Trusted Computing for Embedded Devices: Defenses & Attacks

Secure Application Execution in Mobile Devices

with Mehari G. Msgna, Raja Naeem Akram, Konstantinos Markantonakis

Abstract. Smart phones have rapidly become hand-held mobile devices capable of sustaining multiple applications. Some of these applications allow access to services including health care, financial, online social networks and are becoming common in the smart phone environment. From a security and privacy point of view, this seismic shift is creating new challenges, as the smart phone environment is becoming a suitable platform for security- and privacy-sensitive applications. The need for a strong security architecture for this environment is becoming paramount, especially from the point of view of Secure Application Execution (SAE). In this chapter, we explore SAE for applications on smart phone platforms, to ensure application execution is as expected by the application provider. Most of the proposed SAE proposals are based on having a secure and trusted embedded chip on the smart phone. Examples include the GlobalPlatform Trusted Execution Environment, M-Shield and Mobile Trusted Module. These additional hardware components, referred to as secure and trusted devices, provide a secure environment in which the applications can execute security-critical code and/or store data. These secure and trusted devices can become the target of malicious entities; therefore, they require a strong framework that will validate and guarantee the secure application execution. This chapter discusses how we can provide an assurance that applications executing on such devices are secure by validating the secure and trusted hardware.

Note. Published in [GHNK16].

ARMv8 Shellcodes from 'A' to 'Z'

with Hadrien Barral, Rémi Géraud, Georges-Axel Jaloyan and David Naccache

Abstract. We describe a methodology to automatically turn arbitrary ARMv8 programs into alphanumeric executable polymorphic shellcodes. Shellcodes generated in this way can evade detection and bypass filters, broadening the attack surface of ARM-powered devices such as smartphones.

Note. Published in [BFG+14]. To appear in the proceedings of *the 12th International Conference on Information Security Practice and Experience (ISPEC) 2016* [BFG+16]

3.2.2 Creating Covert Channels & Preventing Their Exploitation

Communicating Covertly through CPU Monitoring

with Jean-Michel Cioranesco and David Naccache

Abstract. Covert channels, introduced by Lampson [W73] are communication channels not intended for information transfer. [W73] was the first to point out that varying the input/output computing ratio of a CPU could allow covert information exchange.

Recently Okamura and Oyama presented a CPU load covert channel between Xen virtual machines [KY10] where clients are connecting to different virtual machines running on the same physical CPU-core.

This covert channel exploits the fact client Alice is paused when client Bob is scheduled, allowing Bob to know if Alice is computing something.

In this column we investigate CPU load covert channels between clients running on a *multi-core* machine. We will show that how covert channels using CPU load are also possible between clients connected to a multi-core remote server.

Note. Published in the proceedings of *IEEE Computer Society 2013* [CFN13].

Process Table Covert Channels: Exploitation and Countermeasures

with Jean-Michel Cioranesco and Rémi Géraud and David Naccache

Abstract. How to securely run untrusted software? A typical answer is to try to isolate the actual effects this software might have. Such counter-measures can take the form of memory segmentation, sandboxing or visualization. Besides controlling potential damage this software might do, such methods try to prevent programs from peering into other running programs' operation and memory.

As programs, no matter how many layers of indirection in place, are really being run, they consume resources. Should this resource usage be precisely monitored, malicious programs might be able to communicate in spite of software protections.

We demonstrate the existence of such a covert channel bypassing isolation techniques and IPC policies. This covert channel that works over all major consumer OSes (Windows, Linux, MacOS) and relies on exploitation of the process table. We measure the bandwidth of this channel and suggest countermeasures.

Note. Published as an IACR ePrint [CFGN16a].

The Conjoined Microprocessor

with Ehsan Aerabi, A. Elhadi Amirouche, Rémi Géraud, David Naccache and Jean Vuillemin

Abstract. Over the last twenty years, the research community has devised sophisticated methods for retrieving secret information from side-channel emanations, and for resisting such attacks. This paper introduces a new CPU architecture called the Conjoined Microprocessor. The Conjoined Microprocessor randomly interleave the execution of two programs at very low extra hardware cost. We developed for the Conjoined Microprocessor a pre-processor tool that turns a target algorithm into two (or more) separate queues like Q_0 and Q_1 that can run in alternation. Q_0 and Q_1 fulfill the same operation as the original target algorithm.

Power-analysis resistance is achieved by randomly alternating the execution of Q_0 and Q_1, with different runs resulting in different interleaving. Experiments reveal that this architecture is indeed effective against CPA.

Note. Published as an IACR ePrint and in the proceedings of *IEEE International Symposium on Hardware Oriented Security and Trust (HOST) 2016* [AAF+15; AAF+16].

3.2.3 Efficient Hardware & Software Implementations

Regulating the Pace of von Neumann Correctors

with Jean-Michel Cioranesco, Rémi Géraud and David Naccache

Abstract. In a celebrated paper published in 1951 [Neu51], von Neumann presented a simple procedure allowing to correct the bias of random sources. This procedure introduces latency between the random outputs. On the other hand, algorithms such as stream ciphers, block ciphers or even modular multipliers usually run in a number of clock cycles which is independent of the operands' values: Feeding such HDL blocks with the inherently irregular output of such de-biased sources frequently proves tricky.

We propose an algorithm to compensate these irregularities, by storing or releasing numbers at given intervals of time. This algorithm is modeled as a special queue that achieves zero blocking probability and a near-deterministic service distribution (i.e. of minimal variance).

While particularly suited to cryptographic applications, for which it was designed, this algorithm also applies to a variety of contexts and constitutes an example of queue for which the buffer allocation problem is feasible.

Note. Published as an IACR ePrint [FGM+15a].

Backtracking-Assisted Multiplication

with Rémi Géraud, Diana Maimuţ, David Naccache and Hang Zhou

Abstract. This paper describes a new multiplication algorithm, particularly suited to lightweight microprocessors when one of the operands is known in advance. The method uses backtracking to find a multiplication-friendly encoding of one of the operands.

A 68HC05 microprocessor implementation shows that the new algorithm indeed yields a twofold speed improvement over classical multiplication for 128-byte numbers.

Note. Published as an IACR ePrint and in the pre-proceedings of *ArcticCrypt 2016* [FGM+15b; HRD+16].

New Algorithmic Approaches to Point Constellation Recognition

with Thomas Bourgeat, Julien Bringer, Herve Chabanne, Robin Champenois, Jeremie Clement, Marc Heinrich, Paul Melotti, David Naccache and Antoine Voizard

Abstract. Point constellation recognition is a common problem with many pattern matching applications. Whilst useful in many contexts, this work is mainly motivated by fingerprint matching. Fingerprints are traditionally modeled as constellations of oriented points called minutiae. The fingerprint verifier's task consists in comparing two point constellations. The compared constellations may differ by rotation and translation or by much more involved transforms such as distortion or occlusion. This paper presents three new constellation matching algorithms. The first two methods generalize an algorithm by Bringer and Despiegel. Our third proposal creates a very interesting analogy between mechanical system simulation and the constellation recognition problem.

Note. Published as an arXiv pre-print and in the proceedings of the *ICT Systems Security and Privacy Protection: 29th IFIP TC 11 International Conference, (SEC) 2014* [BBC+14a; BBC+14b].

3.2.4 Finding Security Flaws in Server Software

Security Researcher Acknowledgments Recipient from Microsoft Online Services.

Note. Officially listed on Microsoft's corporate website for finding and reporting a critical flaw in Microsoft OneDrive (March 2016 Security Researchers). As per a legal agreement with Microsoft, the details of this security flaw are confidential.

https://technet.microsoft.com/en-us/security/cc308575

3.3 Personal Bibliography

3.3.1 Journal Papers

[CFN13] Jean-Michel Cioranesco, Houda Ferradi, and David Naccache. « Communicating Covertly through CPU Monitoring ». In: *IEEE Security and Privacy* 11.6 (2013), pp. 71–73.

[HRDA16] Ferradi Houda, Géraud Rémi, Naccache David, and Tria Assia. « When organized crime applies academic results: a forensic analysis of an in-card listening device ». In: *Journal of Cryptographic Engineering* 6.1 (2016), pp. 49–59. ISSN: 2190-8516. DOI: 10.1007/s13389-015-0112-3. URL: http://dx.doi.org/10.1007/s13389-015-0112-3.

3.3.2 Conference Papers

[BBC+14a] Thomas Bourgeat, Julien Bringer, Hervé Chabanne, Robin Champenois, Jérémie Clément, Houda Ferradi, Marc Heinrich, Paul Melotti, David Naccache, and Antoine Voizard. « New Algorithmic Approaches to Point Constellation Recognition ». In: *ICT Systems Security and Privacy Protection: 29th IFIP TC 11 International Conference, SEC 2014, Marrakech, Morocco, June 2-4, 2014. Proceedings*. Ed. by Nora Cuppens-Boulahia, Frédéric Cuppens, Sushil Jajodia, Anas Abou El Kalam, and Thierry Sans. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 80–90. ISBN: 978-3-642-55415-5. DOI: 10.1007/978-3-642-55415-5_7. URL: http://dx.doi.org/10.1007/978-3-642-55415-5_7.

[MHH+14] Abdalla Michel, Chabanne Hervé, Ferradi Houda, Jainski Julien, and Naccache David. « Improving Thomlinson-Walker's Software Patching Scheme Using Standard Cryptographic and Statistical Tools ». In: *Information Security Practice and Experience: 10th International Conference, ISPEC 2014, Fuzhou, China, May 5-8, 2014. Proceedings*. Ed. by Huang Xinyi and Zhou Jianying. Cham: Springer International Publishing, 2014, pp. 8–14. ISBN: 978-3-319-06320-1. DOI: 10.1007/978-3-319-06320-1_2. URL: http://dx.doi.org/10.1007/978-3-319-06320-1_2.

[AAF+16] Ehsan Aerabi, A. Elhadi Amirouche, Houda Ferradi, Rémi Géraud, David Naccache, and Jean Vuillemin. « The Conjoined Microprocessor ». In: *2016 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016, McLean, VA, USA, May 3-5, 2016*. Ed. by Ryan A. Peter Y., Naccache David, and Quisquater Jean-Jacques. IEEE, 2016, pp. 67–70. ISBN: 978-3-662-49301-4. DOI: 10.1109/HST.2016.7495558. URL: http://dx.doi.org/10.1109/HST.2016.7495558.

[BFG+16] Hadrien Barral, Houda Ferradi, Rémi Géraud, Georges-Axel Jaloyan, and David Naccache. « ARMv8 Shellcodes from 'A' to 'Z' ». In: *The 12th International Conference on Information Security Practice and Experience (ISPEC 2016) Zhangjiajie, China, November 16-18, 2016. Proceedings*. Ed. by Chen Liqun and H. Robert Deng. Cham: Springer International Publishing, 2016.

[CFGN16b] Simon Cogliani, Houda Ferradi, Rémi Géraud, and David Naccache. « Thrifty Zero-Knowledge - When Linear Programming Meets Cryptography ». In: *The 12th International Conference on Information Security Practice and Experience (ISPEC 2016) Zhangjiajie, China, November 16-18, 2016. Proceedings*. Ed. by Chen Liqun and H. Robert Deng. Cham: Springer International Publishing, 2016.

[FGM+16a] Houda Ferradi, Rémi Géraud, Diana Maimuţ, David Naccache, and David Pointcheval. « Legally Fair Contract Signing Without Keystones ». In: *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Cham: Springer International Publishing, 2016, pp. 175–190. ISBN: 978-3-319-39555-5. DOI: 10.1007/978-3-319-39555-5_10. URL: http://dx.doi.org/10.1007/978-3-319-39555-5_10.

[FGN16b] Houda Ferradi, Rémi Géraud, and David Naccache. « Slow Motion Zero Knowledge Identifying with Colliding Commitments ». In: *Information Security and Cryptology: 11th International Conference, Inscrypt 2015, Beijing, China, November 1-3, 2015, Revised Selected Papers*. Ed. by Dongdai Lin, XiaoFeng Wang, and Moti Yung. Cham: Springer International Publishing, 2016, pp. 381–396. ISBN: 978-3-319-38898-4. DOI: 10.1007/978-3-319-38898-4_22. URL: http://dx.doi.org/10.1007/978-3-319-38898-4_22.

[GHNK16] Msgna Mehari G., Ferradi Houda, Akram Raja Naeem, and Markantonakis Konstantinos. « Secure Application Execution in Mobile Devices ». In: *The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*. Ed. by Ryan A. Peter Y., Naccache David, and Quisquater Jean-Jacques. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 417–438. ISBN: 978-3-662-49301-4. DOI: 10.1007/978-3-662-49301-4_26. URL: http://dx.doi.org/10.1007/978-3-662-49301-4_26.

[HRD+16] Ferradi Houda, Géraud Rémi, Maimuţ Diana, Naccache David, and Zhou Hang. « Backtracking-Assisted Multiplication ». In: *Arctic Crypt 2016, July 17-22, Longyearbyen, Svalbard, Norway. Pre-Proceedings*. 2016.

[HRN16] Ferradi Houda, Géraud Rémi, and David Naccache. « Human Public-Key Encryption ». In: *Mycrypt 2016: Paradigm-shifting Crypto Kuala Lumpur, Malaysia, December 1-2, 2016. Proceedings*. Ed. by Phan Raphael C.-W. and Yung Moti. Springer International Publishing, 2016.

[MRN16] Ferradi Houda Marc Beunardeau, Géraud Rémi, and David Naccache. « Honey Encryption for Language: Robbing Shannon to Pay Turing? » In: *Mycrypt 2016: Paradigm-shifting Crypto Kuala Lumpur, Malaysia, December 1-2, 2016. Proceedings*. Ed. by Phan Raphael C.-W. and Yung Moti. Springer International Publishing, 2016.

3.3.3 Manuscripts & Pre-Prints

[BFG+14] Hadrien Barral, Houda Ferradi, Rémi Géraud, Georges-Axel Jaloyan, and David Naccache. *ARMv8 Shellcodes from 'A' to 'Z'*. CoRR, abs/1608.03415. http://arxiv.org/abs/1608.03415. 2014.

[BBC+14b] Thomas Bourgeat, Julien Bringer, Hervé Chabanne, Robin Champenois, Jérémie Clément, Houda Ferradi, Marc Heinrich, Paul Melotti, David Naccache, and Antoine Voizard. *New Algorithmic Approaches to Point Constellation Recognition*. CoRR, abs/1405.1402. http://arxiv.org/abs/1405.1402. 2014.

[AAF+15] Ehsan Aerabi, A. Elhadi Amirouche, Houda Ferradi, Rémi Géraud, David Naccache, and Jean Vuillemin. *The Conjoined Microprocessor*. Cryptology ePrint Archive, Report 2015/974. http://eprint.iacr.org/2015/974. 2015.

[FGM+15a] Houda Ferradi, Rémi Géraud, Diana Maimuţ, David Naccache, and Amaury de Wargny. *Regulating the Pace of von Neumann Correctors*. Cryptology ePrint Archive, Report 2015/849. http://eprint.iacr.org/2015/849. 2015.

[FGM+15b] Houda Ferradi, Rémi Géraud, Diana Maimuţ, David Naccache, and Hang Zhou. *Backtracking-Assisted Multiplication*. Cryptology ePrint Archive, Report 2015/787. http://eprint.iacr.org/2015/787. 2015.

[FGNT15] Houda Ferradi, Rémi Géraud, David Naccache, and Assia Tria. *When Organized Crime Applies Academic Results - A Forensic Analysis of an In-Card Listening Device*. Cryptology ePrint Archive, Report 2015/963. http://eprint.iacr.org/2015/963. 2015.

[CFGN16a] Jean-Michel Cioranesco, Houda Ferradi, Rémi Géraud, and David Naccache. *Process Table Covert Channels: Exploitation and Countermeasures*. Cryptology ePrint Archive, Report 2016/227. http://eprint.iacr.org/2016/227. 2016.

[CFH+16] Simon Cogliani, Bao Feng, Ferradi Houda, Rémi Géraud, Diana Maimuţ, David Naccache, Rodrigo Portella do Canto, and Guilin Wang. *Public-Key Based Lightweight Swarm Authentication*. Cryptology ePrint Archive, Report 2016/750. http://eprint.iacr.org/2016/750. 2016.

[CFGN16c] Simon Cogliani, Houda Ferradi, Rémi Géraud, and David Naccache. *Thrifty Zero-Knowledge - When Linear Programming Meets Cryptography*. Cryptology ePrint Archive, Report 2016/443. http://eprint.iacr.org/2016/443. 2016.

[FGM+16b] Houda Ferradi, Rémi Géraud, Diana Maimuţ, David Naccache, and David Pointcheval. *Legally Fair Contract Signing Without Keystones*. Cryptology ePrint Archive, Report 2016/363. http://eprint.iacr.org/2016/363. 2016.

[FGN16a] Houda Ferradi, Rémi Géraud, and David Naccache. *Slow Motion Zero Knowledge Identifying With Colliding Commitments*. Cryptology ePrint Archive, Report 2016/399. http://eprint.iacr.org/2016/399. 2016.

[LLW+16b] Weiran Liu, Jianwei Liu, Qianhong Wu, Bo Qin, David Naccache, and Houda Ferradi. *Compact CCA2-secure Hierarchical Identity-Based Broadcast Encryption for Fuzzy-entity Data Sharing*. Cryptology ePrint Archive, Report 2016/634. http://eprint.iacr.org/2016/634. 2016.

CHAPTER 4

DESIGNING INTEGRITY PRIMITIVES

Summary

This chapter presents our research results in the area of *integrity*.

The core result of this chapter [1], detailed in Section 4.1, is a new attestation primitive allowing to prove the proper generation of RSA public keys. RSA public keys are central to many cryptographic applications; hence their validity is of primary concern to the scrupulous cryptographer. The most relevant properties of an RSA public key (n, e) depend on the *factors* of n: are they properly generated primes? are they large enough? is e co-prime with $\phi(n)$? etc. And of course, it is out of question to reveal n's factors.

Generic non-interactive zero-knowledge (NIZK) proofs can be used to prove such properties. However, NIZK proofs are not practical at all. Typically, such protocols turn out to be very specialized, and may not always be applicable (e.g., for some small values of e). For some very specific properties, specialized proofs exist but such *ad hoc* proofs are naturally hard to generalize.

Section 4.1 proposes a new type of *general-purpose* compact non-interactive proofs, called *attestations*, allowing the key generator to convince any third party that n was properly generated. The proposed construction applies to *any* prime generation algorithm, and is provably secure in the Random Oracle Model.

As a typical implementation instance, for a 138-bit security, verifying or generating an attestation requires $k = 1024$ prime generations. For this instance, each processed message will later need to be signed or encrypted 14 times by the final users of the attested moduli.

The second result in this chapter [2], detailed in Section 4.2, is a new form of contract co-signature, called *legal fairness*, that does not rely on third parties or arbitrators. The proposed protocol is efficient, compact, fully distributed, fully dynamic, and provably secure in the Random Oracle Model. The protocol is illustrated for two parties using Schnorr's signature scheme.

In two-party computation, achieving both fairness and guaranteed output delivery is well known to be impossible. Despite this limitation, many approaches provide solutions of practical interest by weakening somewhat the fairness requirement. Such approaches fall roughly in three categories: "gradual release" schemes assume that the aggrieved party can eventually reconstruct the missing information; "optimistic schemes" assume a trusted third party arbitrator that can restore fairness in case of litigation; and "concurrent" or "legally fair" schemes in which a breach of fairness is compensated by the aggrieved party having a digitally signed cheque from the other party (called the keystone). Section 4.2 describes and analyses a new contract signing paradigm that doesn't require keystones to achieve legal fairness, and give a concrete construction based on Schnorr signatures which is compatible with standard Schnorr signatures and provably secure.

1. Co-authored with Fabrice Ben Hamouda, Rémi Géraud and David Naccache.
2. Co-authored with Rémi Géraud, Diana Maimuţ, David Naccache and David Pointcheval.

In a way these two results complement each other: attestation certifies the integrity of *computation* whereas legal fairness certifies the integrity of *interaction*.

4.1 Non-Interactive Attestations for Arbitrary RSA Prime Generation Algorithms

4.1.1 Introduction

When provided with an RSA public key n, establishing that n is hard to factor might seem challenging: indeed, most of n's interesting properties depend on its secret factors, and even given good arithmetic properties (large prime factors, etc.) a subtle backdoor may still be hidden in n or e [And93; YY96; YY97; YY05a; YY05b].

Several approaches, mentioned below, focused on proving as many interesting properties as possible without compromising n. However, such proofs are limited in two ways: first, they might not always be applicable — for instance [KKM12; BY96; BY93] cannot prove that (n, e) define a permutation when e is too small. In addition, these *ad hoc* proofs are extremely specialized. If one wishes to prove some new property of n's factors, that would require modeling this new property and looking for a proper form of proof.

This section proposes a new kind of general-purpose compact non-interactive proof ω_n, called *attestation*. An attestation allows the key generator to convince any third party that n was properly generated. The corresponding construction, called an *attestation scheme*, applies to *any* prime generation algorithm $\mathcal{G}(1^P, r)$ where r denotes $\mathcal{G}$'s random tape, and P the size of the generated primes. The method can, for instance, attest that n is composed of primes as eccentric as those for which $\lfloor 9393 \sin^4(p^3) \rfloor = 3939$.

More importantly, our attestation scheme provides the first efficient way to prove that (n, e) defines a permutation for a small e, by making $\mathcal{G}$ only output primes p such that e is coprime with $p - 1$.

Our construction is provably secure in the Random Oracle Model. While polynomial, attestation and verification claim an important (yet feasible) computational effort. This might not be an issue given that attestation typically occurs only once during n's lifetime.

We present two variants: In the first, a valid attestation ω_n ensures that n contains at least two P-bit prime factors generated by $\mathcal{G}$ (if n is honestly generated, n must contain ℓ prime factors, for some integer $\ell \geq 2$ depending on the security parameter). In the second variant, a valid attestation $\omega_{\mathbf{n}}$ covers a set of moduli $\mathbf{n} = (n_1, \ldots, n_u)$ and ensures that at least one of these n_i is a product of two P-bit prime factors generated by $\mathcal{G}$.

We show that in most, if not all, cases, one or the other properties are sufficient.

Both variants are unified into a general attestation scheme [3] to encompass the entire gamut of tradeoffs offered by the concept.

4.1.1.1 Prior Work.

A long thread of papers deals with proving number-theoretic properties of composite moduli. The most general (yet least efficient) of these use non-interactive zero-knowledge (NIZK) proof techniques [CD98; GMW87a; BCC88]. Recent work by Groth [GOS06] establishes that there is a perfect NIZK argument for n being a properly generated RSA modulus. We distinguish between these *generic* proofs that can, in essence, prove anything provable [BGG+90] and *ad hoc* methods allowing to prove proper modulus generation in faster ways albeit for very specific $\mathcal{G}$s.

The first *ad hoc* modulus attestation scheme was introduced by Van de Graff and Peralta [GP88] and consists in proving that n is a Blum integer without revealing its factors. Boyar, Friedl and Lund [BFL90] present a proof that n is square-free. Leveraging [GP88; BFL90], Gennaro, Micciancio and Rabin [GMR98] present a protocol proving that n is the product of two "quasi-safe" primes [4]. Camenisch and Michels [CM99] give an NIZK proof that n is a product of two safe primes. Juels and Guajardo [JJ02] introduce a

3. I.e., use several multi-factor moduli.
4. A prime p is "quasi-safe" if $p = 2u^a + 1$ for a prime u and some integer a.

proof for RSA key generation with verifiable randomness. Besides its complexity, [JJ02]'s main drawback is that public parameters must be published by a trustworthy authority (TTP). Several authors [Mic93; BF97; CFT98; Mao98] describe protocols proving that n is the product of two primes p and q, without proving anything on p, q but their primality. Proving that $n = pq$ is insufficient to ascertain security (for instance, p may be too short). Hence, several authors (e.g., [LS98; Bou00; FO97; FO98; Mao98; CFT98]) introduced methods allowing to prove that p and q are roughly of identical sizes.

This works takes an *entirely different direction*: Given any generation procedure $\mathcal{G}$, we prove that $\mathcal{G}$ has been followed correctly during the generation of n. The new approach requires no TTPs, does not rely on n having any specific properties and attests that the correct prime generation algorithm has been used — with no restriction whatsoever on how this algorithm works.

As such, the concern of generating proper moduli (e.g. such that (N, e) define a permutation, but what constitutes a "proper" modulus may depend on the application) is entirely captured by the concern of choosing G appropriately. Our work merely attests that G was indeed used.

Cryptographic applications of attested RSA moduli abound. We refer the reader to [GMR98] or [Mao98] for an overview of typical applications of attested moduli. In particular, such concerns are salient in schemes where an authority is in charge of generating n (*e.g.* Fiat-Shamir or Guillou-Quisquater) and distributing private keys to users, or in the design of factoring-based verifiable secret-sharing schemes.

4.1.2 Outline of the Approach

The proposed attestation method is based on the following idea: fix $k \geq 2$, generate k random numbers $r_1, \ldots, r_k$ and define $h_i = \mathcal{H}(i, r_i)$ where $\mathcal{H}$ denotes a hash function. Let $p_i = \mathcal{G}(h_i)$ and:

$$N = \prod_{i=1}^{k} p_i$$

Define $(X_1, X_2) = \mathcal{H}'_2(N)$, where $\mathcal{H}'_2$ is a hash function which outputs two indices $1 \leq X_1 < X_2 \leq k$. We later show how to construct such an $\mathcal{H}'_2$. This defines $n = p_{X_1} \times p_{X_2}$ and

$$\omega_n = \{r_1, r_2, \ldots, r_{X_1-1}, \star, r_{X_1+1}, \ldots, r_{X_2-1}, \star, r_{X_2+1}, \ldots, r_k\}$$

Here, a star symbol ($\star$) denotes a placeholder used to skip one index. The data ω_n is called the *attestation* of n. The algorithm $\mathcal{A}$ used to obtain ω_n is called an *attestator*.

The attestation process is illustrated in Figure 4.1: the choice of the r_i determines N, which is split into two parts: n and N/n. Splitting is determined by d, which is the digest of N, and is hence unpredictable for the opponent.

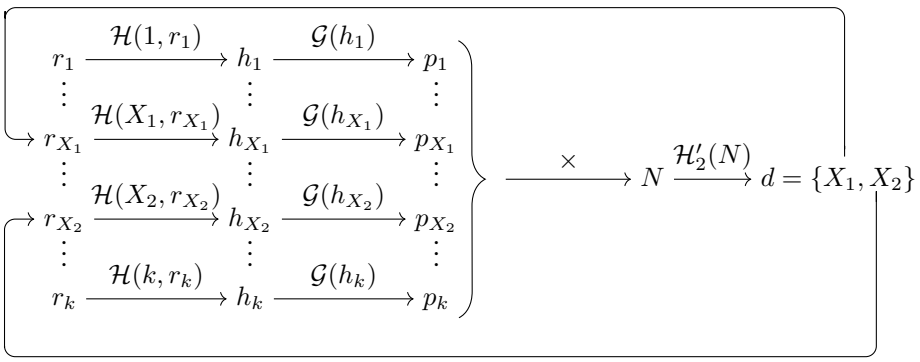


Figure 4.1 – The approach used to generate and validate an attestation.

Verifying the validity of such an attestation ω_n is performed as follows: all (non-star) values r_i in ω_n are fed to $\mathcal{G}$ to generate primes, that are multiplied together and by n. This gives back N. If by hashing

N and reading, as earlier, the digest of N (denoted d) as two values X_1 and X_2, we get the two exact starred positions X_1 and X_2 in ω_n, then ω_n is valid; else ω_n is invalid. The algorithm $\mathcal{V}$ we just described is called a *validator*. It is very similar to the attestator $\mathcal{A}$ mentioned above.

For a subtle reason, the r_i's are pre-processed into a set of values h_i before being fed into $\mathcal{G}$. The values h_i are generated by hashing the input r_is with their positions i. This serves two purposes: first, the hash welds together r_i and its position i in the list, which prevents the opponent from shuffling the p_is to his advantage; second, hashing prevents the opponent from manipulating the r_i's to influence $\mathcal{G}$'s output.

Evidently, as presented here, the method requires a very large k to achieve a high enough security level. The attacker, who chooses X_1, X_2, is expected to perform $k(k-1)/2$ operations to succeed. We circumvent this limitation using two techniques:

— The first technique uses ℓ indices $X_1, \ldots, X_\ell$ and not only $\ell = 2$. In RSA, security depends on the fact that n contains *at least* two properly formed prime factors. Hence we can afford to shorten k by allowing more factors in n. The drawback of using ℓ-factor moduli is an significant user slow-down as most factoring-based cryptosystems run in $O(\log^3 n)$. Also, by doing so, we prove that n *contains* a properly formed modulus rather than that n *is* a properly formed modulus.

— A second strategy consists in using $2u$ indices to form u moduli $n_1, \ldots, n_u$. Here, each user will be given u moduli and will process [5] each message u times. Thereby, total signature size and slow-down are only linear in ℓ. Encryption is more tricky: while for properly signing a message it suffices that *at least* one n_i is secure, when encrypting a message *all* n_i must be secure. Hence, to encrypt, the sender will pick u session keys κ_i, encrypt each κ_i using n_i, and form the global session-key $\kappa = \kappa_1 \oplus \ldots \oplus \kappa_u$. The target message will then be encrypted (using a block-cipher) using κ. In other words, it suffices to have at least *one* factoring-resistant n_i to achieve message confidentiality. Interestingly, to be secure a signature conceptually behaves as a logical "or", while encryption behaves as a logical "and".

The size of ω_n is also a concern in this simple outline. Indeed, as presented here ω_n is $O(kR)$ bits large, where R represents the bit size of the r_i [6]. Given the previous remark on k being rather large, this would result in very large attestations. Luckily, it turns out that attestation size can be reduced to $O(R \log k)$ using hash trees, as we explain in Section 4.1.6.

4.1.2.1 Note.

Multiplication in $\mathbb{N}$ is one implementation option. All we need is a *completely multiplicative operation*. For instance, as we have:

$$\left(\frac{a}{N}\right) = \left(\frac{a}{p_1}\right)\left(\frac{a}{p_2}\right)\cdots\left(\frac{a}{p_k}\right),$$

the hash of the product of the Jacobi symbols of the p_i with respect to the first primes $a_j = 2, 3, 5, \ldots$ [7] can equally serve as an index generator.

Before we proceed note that when generating a complete RSA key pair (n, e), it is important to ascertain that $\gcd(e, \phi(n)) = 1$. This constraint is easy to integrate into $\mathcal{G}$ [8]. All in all, what we prove is that with high probability, *the key was generated by the desired algorithm* $\mathcal{G}$, whichever this $\mathcal{G}$ happens to be.

4.1.3 Model and Analysis

4.1.3.1 Preliminaries and Notations

We now formally introduce the tools using which the method sketched in Section 4.1.2 is rigorously described and analyzed.

5. Sign, verify, encrypt, or decrypt.
6. Because $\mathcal{G}$ may destroy entropy, R must be large enough to make the function $\mathcal{G}(\mathcal{H}(i, r))$ is collision resistant.
7. This product is actually an a_j-wise exclusive-or.
8. A simple way to do so consists in re-running $\mathcal{G}$ with $r_i \| j$ (instead of r_i) for $j = 1, 2, \ldots$ until $\gcd(p_i - 1, e) = 1$.

Throughout this section, λ will denote a security parameter. The expression *polynomial time* will always refer to λ. The construction uses two cryptographic hash functions: a classical hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^R$ and a second hash function $\mathcal{H}'_d : \{0,1\}^* \to \mathcal{S}_d$ where $\mathcal{S}_d$ is the set of subsets of $\{1,\ldots,k\}$ of size d (for some positive integer d and k). $\mathcal{H}'$ can be constructed from a classical hash function using an unranking function [SW86] (see Appendix 4.1.9). Both hash functions will be modeled as random oracles in the security analysis.

Let $k \geq 2$. Moreover our attestation and validation algorithms always implicitly take λ as input. We denote by $|a|$ the bit size of a.

Let $\mathcal{G}(1^P, r)$ be a polynomial-time algorithm which, on input of a unary size P and of a random seed $r \in \{0,1\}^R$ produces a prime or a probably prime p of size P. The argument 1^P is often omitted, for the sake of simplicity. The size P of the primes is supposed to be a function of λ. We write $r_1 \xleftarrow{\$} \{0,1\}^R$ to indicate that the seed r_1 is chosen uniformly at random from $\{0,1\}^R$.

An attestation scheme for $\mathcal{G}$ is a pair of two algorithms $(\mathcal{A}, \mathcal{V})$, where

— $\mathcal{A}$ is an *attestation algorithm* which takes as input k random entries ($r_1,\ldots,r_k \in \{0,1\}^R$, in the sequel) and which outputs a tuple of moduli $\mathbf{n} = (n_1,\ldots,n_u)$ along with a bit string $\omega_{\mathbf{n}}$, called an *attestation*; u and k are integer parameters depending on λ; when $u = 1$, n_1 is denoted n;

— $\mathcal{V}$ is a *validation algorithm* which takes as input a tuple of moduli $\mathbf{n} = (n_1,\ldots,n_u)$ together with an attestation $\omega_{\mathbf{n}}$. $\mathcal{V}$ checks $\omega_{\mathbf{n}}$, and outputs True or False.

An *attestation scheme* must comply with the following properties:

— *Randomness.* If $r_1,\ldots,r_k$ are independent uniform random values, $\mathcal{A}(1^\lambda, r_1,\ldots,r_k)$ should output a tuple of moduli $\mathbf{n} = (n_1,\ldots,n_u)$ where each n_i is the product of ℓ random primes generated by $\mathcal{G}$. The positive integer $\ell \geq 2$ is a parameter depending on λ. More formally the two following distributions should be statistically indistinguishable:

$$\left\{ \mathbf{n} = (n_1,\ldots,n_u) \;\middle|\; \begin{array}{l} (r_1,\ldots,r_k) \xleftarrow{\$} \{0,1\}^R \\ (n_1,\ldots,n_u,\omega_n) \leftarrow \mathcal{A}(r_1,\ldots,r_k) \end{array} \right\}$$

$$\left\{ \mathbf{n} = (n_1,\ldots,n_u) \;\middle|\; \begin{array}{l} (r_1,\ldots,r_{\ell u}) \xleftarrow{\$} \{0,1\}^R \\ n_1 \leftarrow \mathcal{G}(r_1)\cdots\mathcal{G}(r_\ell), \ldots, n_u \leftarrow \mathcal{G}(r_{(u-1)\ell+1})\cdots\mathcal{G}(r_{u\ell}) \end{array} \right\}$$

— *Correctness.* The validator $\mathcal{V}$ always accepts an attestation honestly generated by the attestator $\mathcal{A}$. More precisely, for all $r_1,\ldots,r_k$:

$$\mathcal{V}\left(\mathcal{A}(1^\lambda, r_1,\ldots,r_k)\right) = \mathsf{True}.$$

— *Soundness.* No polynomial-time adversary $\mathcal{F}$ can output (with non-negligible probability) a tuple $\mathbf{n} = (n_1,\ldots,n_u)$ and a valid attestation $\omega_{\mathbf{n}}$ such that no n_i contains at least two prime factors generated by $\mathcal{G}$ with two distinct random seeds. More formally, for any polynomial-time adversary $\mathcal{F}$, the soundness advantage $\mathsf{Adv}^{\mathsf{snd}}(\mathcal{F})$ defined as

$$\Pr\left[(\mathbf{n} = (n_1,\ldots,n_u), \omega_{\mathbf{n}}) \xleftarrow{\$} \mathcal{F}(1^\lambda) \;\middle|\; \begin{array}{l} \mathcal{V}(n_1,\ldots,n_u,\omega_n) = \mathsf{True} \text{ and} \\ \forall i = 1,\ldots,u,\ \nexists s_1, s_2 \in \{0,1\}^R, \\ \qquad s_1 \neq s_2 \text{ and } \mathcal{G}(s_1)\cdot\mathcal{G}(s_2) \text{ divides } n_i \end{array} \right]$$

is negligible in λ.

We remark that when it is hard to find two seeds s_1 and s_2 such that $\mathcal{G}(s_1) = \mathcal{G}(s_2)$, then soundness basically means that one of the n_i's contains a product of two distinct primes generated by $\mathcal{G}$. In addition, when $\ell = 2$, if $\mathcal{V}$ rejects moduli of size different from $2P$ (the size of a honestly generated modulus), one of the n_i's is necessarily exactly the product of two prime factors generated by $\mathcal{G}$.

Table 4.1 summarizes the various parameters used in our construction (all are supposed to be function of λ). We now describe the following two variants:

— The multi-prime variant, where $\mathcal{A}$ only outputs one modulus (*i.e.* $u = 1$);

— The multi-modulus variant, where $\mathcal{A}$ outputs $u \geq 2$ two-factor moduli (*i.e.* $\ell = 2$).

Table 4.1 – Summary of the various parameters

λ	security parameter (all the other parameters are function of λ)
P	size of prime numbers p_i generated by $\mathcal{G}$
R	size of the seed used by $\mathcal{G}$ to generate a prime number
k	number of primes generated by the attestator $\mathcal{A}$, which is the *dominating cost of* $\mathcal{A}$
u	number of moduli output by $\mathcal{A}$ ($u = 1$ in the multi-prime variant, and $u \geq 2$ in the multi-modulus variant)
ℓ	number of factors of each modulus n_i: $\lvert n_i \rvert = \ell P$

Algorithm 1 Attestator $\mathcal{A}$ for the Multi-Prime Attestation Scheme ($u = 1$) with $\ell = 2$

Input: $r_1, \ldots, r_k$
Output: n, ω_n
  $N \leftarrow 1$
  for all $i = 1$ to k **do**
    $h_i \leftarrow \mathcal{H}(i, r_i)$
    $p_i \leftarrow \mathcal{G}(h_i)$
    $N \leftarrow N \times p_i$
  end for
  $X_1, X_2 \leftarrow \mathcal{H}'_2(N)$
  $\omega_n \leftarrow \{r_1, \ldots, r_{X_1-1}, \star, r_{X_1+1}, \ldots, r_{X_2-1}, \star, r_{X_2+1}, \ldots, r_k\}$
  $n \leftarrow p_{X_1} \times p_{X_2}$
  return n, ω_n

4.1.3.2 Multi-Prime Attestation Scheme ($u = 1$)

We now describe the algorithms $\mathcal{A}$ and $\mathcal{V}$ that generate and verify, respectively, an attestation along with an RSA public key, when $u = 1$ (only one modulus is generated). Algorithms in this Section are given for $\ell = 2$ (corresponding to the common case where $n = pq$) for the sake of clarity and as a warm-up.

Algorithms for arbitrary ℓ are particular cases of the general algorithms described in Section 4.1.4.1.

In Algorithms 1 and 2, a star symbol ($\star$) denotes a placeholder used to skip one index.

4.1.3.2.1 Generating an Attestation.

The attestator $\mathcal{A}$ is described in Algorithm 1. $\mathcal{A}$ calls $\mathcal{H}$ and $\mathcal{G}$.

In this setting, the attestation has size k. This size is reduced to $\log k$ using hash trees as described in Section 4.1.6.

4.1.3.2.2 Verifying an Attestation.

The validator $\mathcal{V}$ is described in Algorithm 2. $\mathcal{V}$ calls the *same* $\mathcal{H}$, $\mathcal{G}$, and unrank as $\mathcal{A}$.

4.1.3.2.3 Correctness:

The h_is are generated deterministically, therefore so are the p_is, and their product times n yields the correct value of N. unrank is also deterministic, therefore X_1 and X_2 are the same in Algorithms 1 and 2.

4.1.3.2.4 Randomness:

In the Random Oracle Model (for $\mathcal{H}$), the scheme's *randomness* is proven later in Section 4.1.5.1, as a particular case of the general scheme's [9] soundness.

9. C.f. Section 4.1.4.1

Algorithm 2 Validator $\mathcal{V}$ for the Multi-Prime Attestation Scheme ($u = 1$) with $\ell = 2$

Input: n, ω_n
Output: True or False
  $N \leftarrow n$
  for all $r_i \neq \star$ in ω_n **do**
    $h_i \leftarrow \mathcal{H}(i, r_i)$
    $p_i \leftarrow \mathcal{G}(h_i)$
    $N \leftarrow N \times p_i$
  end for
  $X_1, X_2 \leftarrow \mathcal{H}'_2(N)$
  if $r_{X_1} = \star$ and $r_{X_2} = \star$ and $\#\{r_i \in \omega_n \text{ s.t. } r_i = \star\} = 2$ and $|n| = \ell P$ **then**
    return True
  else
    return False
  end if

Algorithm 3 Attestator $\mathcal{A}$ for the Multi-Modulus Attestation Scheme ($u \geq 2$, $\ell = 2$)

Input: $r_1, \ldots, r_k$
Output: $\mathbf{n} = (n_1, \ldots, n_u), \omega_{\mathbf{n}}$
  $N \leftarrow 1$
  for all $i = 1$ to k **do**
    $h_i \leftarrow \mathcal{H}(i, r_i)$
    $p_i \leftarrow \mathcal{G}(h_i)$
    $N \leftarrow N \times p_i$
  end for
  $X_1, \ldots, X_{2u} \leftarrow \mathcal{H}'_{2u}(N)$
  $\omega_{\mathbf{n}} \leftarrow \{r_1, \ldots, r_{X_1-1}, \star, r_{X_1+1}, \ldots, r_{X_{u\ell}-1}, \star, r_{X_{u\ell}+1}, \ldots, r_k\}$
  for all $j = 1$ to u **do**
    $n_j \leftarrow p_{X_{2j}} \times p_{X_{2j+1}}$
  end for
  return $\mathbf{n} = (n_1, \ldots, n_u), \omega_{\mathbf{n}}$

Algorithm 4 Validator $\mathcal{V}$ for the Multi-Modulus Attestation Scheme ($u \geq 2$, $\ell = 2$)

Input: $\mathbf{n} = (n_1, \ldots, n_u), \omega_{\mathbf{n}}$
Output: True or False
  $N \leftarrow n_1 \times \cdots \times n_u$
  for all $r_i \neq \star$ in $\omega_{\mathbf{n}}$ **do**
    $h_i \leftarrow \mathcal{H}(i, r_i)$
    $p_i \leftarrow \mathcal{G}(h_i)$
    $N \leftarrow N \times p_i$
  end for
  $X_1, \ldots, X_{2u} \leftarrow \mathcal{H}'_{2u}(N)$
  if $r_j = \star$ for all $j = 1$ to u and $\#\{r_i \text{ s.t. } r_i = \star\} = 2u$ and $|n_1| = \cdots = |n_u| = 2P$ **then**
    return True
  else
    return False
  end if

4.1.4 Multi-Modulus Attestation Scheme ($u \geq 2$, $\ell = 2$)

The second variant consists in generating in a batch $u = \ell/2$ bi-factor moduli. The corresponding attestator and validator are given in Algorithms 3 and 4.

Algorithm 5 Attestator $\mathcal{A}$ for the General Attestation Scheme ($u \geq 1$, $\ell \geq 2$)

Input: $r_1, \ldots, r_k$
Output: $\mathbf{n} = (n_1, \ldots, n_u), \omega_{\mathbf{n}}$
  $N \leftarrow 1$
  for all $i = 1$ to k **do**
    $h_i \leftarrow \mathcal{H}(i, r_i)$
    $p_i \leftarrow \mathcal{G}(h_i)$
    $N \leftarrow N \times p_i$
  end for
  $X_1, \ldots, X_{u\ell} \leftarrow \mathcal{H}'_{u\ell}(N)$
  $\omega_{\mathbf{n}} \leftarrow \{r_1, \ldots, r_{X_1-1}, \star, r_{X_1+1}, \ldots, r_{X_{u\ell}-1}, \star, r_{X_{u\ell}+1}, \ldots, r_k\}$
  for all $j = 1$ to u **do**
    $n_j \leftarrow p_{X_{(\ell-1)j+1}} \times \cdots \times p_{X_{\ell j}}$
  end for
  return $\mathbf{n} = (n_1, \ldots, n_u), \omega_{\mathbf{n}}$

Algorithm 6 Validator $\mathcal{V}$ for the General Attestation Scheme ($u \geq 1$, $\ell \geq 2$)

Input: $\mathbf{n}, \omega_{\mathbf{n}}$
Output: True or False
  $N \leftarrow n_1 \times \cdots \times n_u$
  for all $r_i \neq \star$ in $\omega_{\mathbf{n}}$ **do**
    $h_i \leftarrow \mathcal{H}(i, r_i)$
    $p_i \leftarrow \mathcal{G}(h_i)$
    $N \leftarrow N \times p_i$
  end for
  $X_1, \ldots, X_{2u\ell} \leftarrow \mathcal{H}'_{u\ell}(N)$
  if $r_{X_j} = \star$ for $j = 1$ to ℓ and $\#\{r_i \text{ s.t. } r_i = \star\} = u\ell$ and $|n_1| = \cdots = |n_u| = \ell P$ **then**
    return True
  else
    return False
  end if

4.1.4.1 General Attestation Scheme

Algorithms 5 and 6 describe our general attestation scheme, for any $u \geq 1$ and $\ell \geq 2$. The previous multi-prime and multi-modulus schemes are illustrative particular cases of this scheme.

The *correctness* and *randomness* arguments are similar to those of Section 4.1.3.2. In addition, the attestation has size k. This size is brought down to $\ell u \log k$ using hash-trees as described in Section 4.1.6.

4.1.5 Security and Parameter Choice

4.1.5.1 Security

In this section, we prove that for correctly chosen parameters u, ℓ, k, the general attestation scheme defined in Section 4.1.4.1 (Algorithms 5 and 6) is sound. We recall that the two other properties required by an attestation scheme (namely correctness and randomness) were proven in previous sections.

More formally, we have the following theorem:

Theorem 4.1 *In the Random Oracle Model, the soundness advantage of an adversary making $q_{\mathcal{H}}$ queries to $\mathcal{H}$ and $q_{\mathcal{H}'}$ queries to $\mathcal{H}'$ is at most:*

$$(q_{\mathcal{H}'} + 1) \cdot \left(\frac{\ell u}{k - (\ell - 1)u + 1} \right)^{(\ell-1)u} + \frac{q_{\mathcal{H}} \cdot (q_{\mathcal{H}} - 1)}{2} \cdot p_{\mathcal{G}-\text{col}} \ ,$$

where $p_{\mathcal{G}-\mathrm{col}}$ is the probability that two $\mathcal{G}(r) = \mathcal{G}(s)$, when $r, s \xleftarrow{\$} \{0,1\}^R$.

We point out that $p_{\mathcal{G}-\mathrm{col}}$ must be small, otherwise the generated primes are unsafe in any case.

Proof: First, we denote by S_i the set of all prime numbers $\rho = \mathcal{G}(\mathcal{H}(i, r))$, for which (i, r) has been queried to $\mathcal{H}$ (for $i = 1, \ldots, k$). We remark that the probability that two such primes ρ are equal is at most $\frac{q_{\mathcal{H}} \cdot (q_{\mathcal{H}}-1)}{2} \cdot p_{\mathcal{G}-\mathrm{col}}$. This is the second term in the security bound.

In the sequel, we suppose that there are no collisions between the primes. Thus the sets S_i are pairwise disjoint.

Now assume that the adversary $\mathcal{F}$ has been able to forge a valid attestation $\omega_{\mathbf{n}}$ for $\mathbf{n} = (n_1, \ldots, n_u)$ and let $N = \beta \prod_{i=1}^{u} n_i$, where β stands for the product of all the primes generated from the elements of $\omega_{\mathbf{n}}$. As the attestation is valid, $|n_1| = \cdots = |n_u| = \ell P$. Let $N = \prod_{i=1}^{L} \rho_i$ be the prime decomposition of N. Up to reordering the sets S_i, there exists an integer t such that:

— none of $S_1, \ldots, S_t$ contains a factor ρ_i;

— each of $S_{t+1}, \ldots, S_k$ contains a factor ρ_i. We arbitrarily choose a prime $p_i \in S_i$ for $i = t+1, \ldots, k$.

We distinguish two cases:

— if $t < (\ell - 1) \cdot u$, then this means that N is divisible by $m = p_{t+1} \times \cdots \times p_k$. But we also know that N is divisible by $n_1 \times \cdots \times n_u$. As $|n_1 \times \cdots \times n_u| = \ell u P$, $|m| = (k-t)P > kP - (\ell-1)uP + P$, and $|N| = kP$, we have

$$|\gcd(n_1 \cdots n_u, m)| \geq |n_1 \cdots n_u| + |m| - |N| \geq (u+1)P.$$

This implies that $n_1 \times \cdots \times n_u$ is divisible by at least $u+1$ distinct primes among $p_{t+1}, \ldots, p_k$. By the pigeon-hole principle, at least one of the n_i's is divisible by two distinct primes generated as $\mathcal{G}(r_i)$ for two distinct seeds r_i (seeds have to be distinct, otherwise the two primes would be equal).

— if $t \geq (\ell - 1) \cdot u$, the adversary will only be able to generate a valid attestation if none of the indices $X_1, \ldots, X_{u\ell}$ (obtained by $\mathcal{H}'_{u\ell}(N)$) falls in $\{1, \ldots, t\}$. As $\{1, \ldots, k\} \setminus \{X_1, \ldots, X_{u\ell}\}$ is a random subset of $\{1, \ldots, k\}$ with $k - \ell u$ elements, the previous bad event ($\mathcal{F}$ is able to generate a valid attestation) corresponds to this set being a subset of $\{t+1, \ldots, k\}$ and happens with probability:

$$\frac{\binom{k-t}{k-\ell u}}{\binom{k}{k-\ell u}} = \frac{(k-t) \cdot (k-t-1) \cdots (k-\ell u+1)}{k \cdot (k-1) \cdots (k-\ell u+1)} \cdot \frac{(\ell u)!}{(\ell u - t)!}$$
$$\leq \frac{1}{(k-t+1)^t} \cdot (\ell u)^t \leq \left(\frac{\ell u}{k - (\ell-1)u + 1} \right)^{(\ell-1) \cdot u}.$$

Since $\mathcal{F}$ makes $q_{\mathcal{H}'}$ queries to $\mathcal{H}'$, we get the theorem's bound (where the $+1$ corresponds to the query necessary to verify $\mathcal{F}$'s attestation if he did not do it himself).

□

4.1.5.2 Typical Parameters and Complexity Analysis

Algorithms 5 and 6 have the following properties:

— Attestation size $|\omega_n| = 2u\ell R \log k$, using the hash-tree compression technique in Section 4.1.6

— λ-bit security approximately when:

$$\left(\frac{\ell u}{k - (\ell-1)u + 1} \right)^{(\ell-1)u} \leq 2^{-\lambda}$$

(according to the soundness bound given by Theorem 4.1, omitting the second part, which is negligible in practice);

— Attestation and validation times mostly consist in generating (or re-generating) the k primes. Validation time is very slightly faster than attestation time.

4.1.6 Compressing the Attestation

As mentioned above, providing an attestation ω_n "as is" might be cumbersome, as it grows linearly with k. However, it is possible to drastically reduce ω_n's size using the following technique.

The tree of Figure 4.2 is constructed as follows: Let h be some public hash function. Each non-leaf node C of the tree has two children, whose value is computed by $r_{x0} \leftarrow h(r_x, 0)$ and $r_{x1} \leftarrow h(r_x, 1)$ for the left child and the right child respectively, where r_x is the value of C. Given a root seed r, one can therefore reconstruct the whole tree. The leaf values can now be used as r_i's for the attestation procedure.

To compress ω_n we proceed as follows:
- Get the indices X_1 and X_2 from the attestation procedure;
- Identify the paths from X_1 up to the root, and mark them;
- Identify the paths from X_2 up to the root, and mark them;
- Send the following information:

$$\omega_n = \{\text{for all leaves } L, \text{highest-ranking unmarked parent of } L\}$$

This requires revealing at most $2\log_2 k$ intermediate higher-rank hashes [10] instead of the $k-2$ values required to encode ω_n when naively sending the seeds directly.

Generalization to $u\ell \geq 2$ is straightforward.

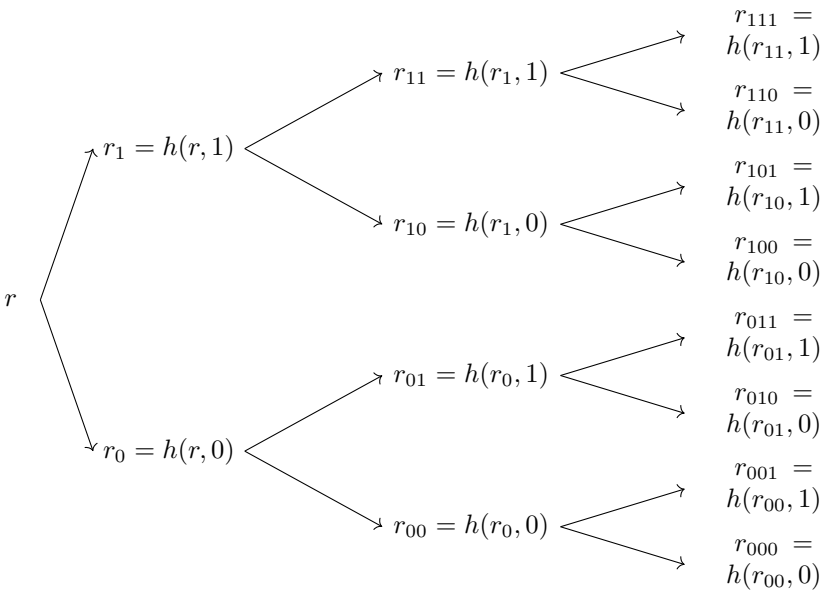


Figure 4.2 – Compressing ω_n using a hash tree.

4.1.7 Parameter Settings

Table 4.2 shows typical parameter values illustrating different trade-offs between security (λ), attestation size ($2u\ell R \log k$), modulus size (ℓ), the number of required moduli (u), and the work factors of $\mathcal{A}$ and $\mathcal{V}$ ($kt_{\mathcal{G}}$ where $t_{\mathcal{G}}$ is $\mathcal{G}$'s average running time). Table 4.3 provides the same information for the multi-modulus variant.

We (arbitrarily) consider that reasonable attestations and validations should occur in less than ten minutes using standard HSM such as the IBM 4764 PCI-X Cryptographic Co-processor [IBM] or Oracle's Sun Crypto Accelerator SCA 6000 [Ora]. When run with 7 threads in the host application, the 4764 generates on average 2.23 key-pairs per second (1,024 bits). The SCA 6000 (for which average key generation

10. I.e., we essentially only publish co-paths.

figures are not available) is about 11 times faster than the 4764 when processing RSA 1,024-bit keys. Hence we can assume that the SCA 6000 would generate about 24 key-pairs per second. We thus consider that average-cost current-date HSMs generate 10 key-pairs per second, *i.e.* 20 primes per second.

Spending ten minutes to generate or validate an attestation might not be an issue given that attestation typically occurs only once during n's lifetime. This means that a "reasonable" attestation implementation would use $k = 10 \times 60 \times 20 =$ 12,000. This gives $\ell = 10$ and $\ell = 6$ for the multi-prime and multi-modulus $\mathcal{A}$ (respectively) for $\lambda = 128$.

Note that in practical field deployments an attestation would be verified *once* by a trusted *Attestation Authority* and replaced by a signature on n (or $\mathbf{n}$).

According to the bounds of Theorem 4.1, we have

$$\lambda \geq -(\ell - 1)u \log_2 \left(\frac{\ell u}{k - (\ell - 1)u + 1} \right)$$

Table 4.2 – Some typical parameters for multi-factor attestation ($u = 2$). Each table entry contains λ for the corresponding choice of k and ℓ.

$\log_2 k$	Time	$\ell = 6$	$\ell = 8$	$\ell = 10$	$\ell = 12$	$\ell = 14$	$\ell = 16$	$\ell = 18$	$\ell = 20$
8	25 s	43	54	64	72	79	84	89	93
9	51 s	53	69	83	95	107	117	126	135
10	1.7 min	64	83	101	118	134	148	162	175
11	3.4 min	74	97	119	140	160	179	197	214
12	6.8 min	84	111	138	162	186	209	231	253
13	13.7 min	94	125	156	185	212	239	266	291
14	27.3 min	104	139	174	207	238	269	300	329
15	54.6 min	114	153	192	229	264	299	334	367
16	1.8 hrs	124	167	210	251	290	329	368	405
17	3.6 hrs	134	181	228	273	317	359	402	443
18	7.3 hrs	144	195	246	295	343	389	436	481
19	14.6 hrs	154	209	264	317	369	419	470	519
20	1.2 d	164	223	282	339	395	449	504	557
21	2.4 d	174	237	300	361	421	479	538	595

Table 4.2 is read as follows: we can see that taking for instance $\ell = 10$ and $\log_2 k = 13$ with the multi-factor version gives 156-bit security. In Table 4.3, taking $\ell = 10$ and $\log_2 k = 13$ with the multi-modulus version gives 285-bit security.

4.1.8 Conclusion and Further Research

The construction described in this section attests in a non-interactive way that n was properly generated using an arbitrary (publicly known) prime generator $\mathcal{G}$. The attestation is compact and publicly verifiable. As a result, any entity can convince herself of the modulus' validity before using it. Even though computation times may seem unattractive, we stress that attestation generation and verification only need to be performed once.

This work raises a number of interesting questions.

Committing to the primes p_i's might also be achieved using more involved tools such as pairings. For instance, given the commitments g^{p_1} and g^{p_2}, it is easy to check that $e(g^{p_1}, g^{p_2}) = e(g, g)^n$.

An interesting research direction consists in hashing $N \bmod v$ (instead of N) for some public v, to speed-up calculations. However, the condition $v > n$ must be enforced by design to prevent an opponent from using ω_n as the "attestation" of $n + tv$ for some $t \in \mathbb{N}$. Note that we did not adapt our security proof to this (overly?) simplified variant.

In general, any strategy allowing to reduce k without impacting λ would yield more efficient attestators. Also, generalizing and applying this approach to the parameter generation of other cryptographic problems, such as the discrete logarithm, may prove useful.

Finally, to date, no attestation method proves (without resorting to TTPs) that the random tape used for forming the primes was properly drawn. Like all other prior work articles cited in Section 4.1.1, we do not address this issue and assume that the random number that feeds $\mathcal{G}$ was not biased by the attacker.

4.1.9 Implementing the Second Hash Function $\mathcal{H}'$

To implement the second hash function $\mathcal{H}'_d$ from a classical hash function, we can apply an unranking hash function [SW86], which maps an integer (in some interval) to a subset $\{X_1, \ldots, X_u\} \subset \{0, \ldots, k-1\}$.

As an example, we describe here a simple (natural) unranking function. Let $\mathcal{H}''$ be a classical hash function with range $\{0, \ldots, M\}$, where $M = k(k-1)\cdots(k-u+1) - 1$. To hash a value N, we first compute $r \leftarrow \mathcal{H}''(N)$. Then we compute the integers $r_1, \ldots, r_d$ as in Algorithm 7.

Algorithm 7 Unranking algorithm

Input: r, k, u
Output: $r_1, \ldots, r_u$
  for all $i = 1$ to u **do**
    $r_i \leftarrow r \bmod (k-i+1)$
    $r \leftarrow r \text{ div } (k-i+1)$
  end for
  return $r_1, \ldots, r_u$

Algorithm 7 generates a mixed radix representation of r, hence any $r \in [0, M]$ can be represented this way. We now generate the unranking $X_1, \ldots, X_d$ iteratively as follows:

— $X_1 \leftarrow r_1$
— $X_{i+1} \leftarrow r_{i+1} + \#\{X_j \text{ s.t. } X_j \leq r_{i+1} \text{ for } j \leq i\}$

In other terms, we have a pool of M values, and for each i, one of these values is drawn and assigned to X_i. Hence it is easy to check that this provides a list of pairwise distinct integers.

This algorithm is simple and illustrates how unranking may be implemented. Alternative unranking methods can be found in [SW86].

Table 4.3 – Some typical parameters for multi-modulus attestation ($u = \ell/2$). Each cell contains λ for the corresponding choice of k and ℓ. Some choices of parameters are incompatible and are hence indicated by a dash.

$\log_2 k$	Time	$\ell = 6$	$\ell = 8$	$\ell = 10$	$\ell = 12$	$\ell = 14$	$\ell = 16$	$\ell = 18$	$\ell = 20$	$\ell = 30$	$\ell = 40$	$\ell = 50$	$\ell = 60$	$\ell = 70$	$\ell = 80$
7	12 s	39	46	33	-	-	-	-	-	-	-	-	-	-	-
8	25 s	56	79	93	92	69	11	-	-	-	-	-	-	-	-
9	51 s	71	109	145	173	191	194	176	131	-	-	-	-	-	-
10	1.7 min	87	138	193	246	295	338	371	391	169	-	-	-	-	-
11	3.4 min	102	167	239	315	393	469	542	611	801	519	-	-	-	-
12	6.8 min	117	195	285	383	487	594	704	814	1315	1600	1470	655	-	-
13	13.7 min	132	223	330	450	579	717	861	1011	1786	2505	3036	3248	2989	2064
14	27.3 min	147	251	375	516	671	838	1016	1204	2239	3342	4410	5347	6065	6468
15	54.6 min	162	279	420	582	762	959	1170	1396	2682	4150	5705	7267	8768	10143
16	1.8 hrs	177	307	465	648	853	1079	1324	1586	3121	4944	6964	9109	11319	13540
17	3.6 hrs	192	335	511	714	944	1199	1477	1777	3558	5731	8205	10914	13800	16814
18	7.3 hrs	207	363	556	780	1036	1319	1630	1967	3994	6514	9439	12702	16248	20030
19	14.6 hrs	222	391	601	846	1127	1439	1783	2157	4430	7296	10668	14480	18679	23217
20	1.2 d	237	419	646	912	1218	1559	1936	2347	4865	8076	11895	16255	21102	26391
21	2.4 d	252	447	691	978	1309	1679	2089	2537	5300	8857	13121	18027	23521	29558

4.2 Legally Fair Contract Signing Without Keystones

4.2.1 Introduction

When mutually distrustful parties wish to compute some joint function of their private inputs, they require a certain number of security properties to hold for that computation:

— *Privacy*: Nothing is learned from the protocol besides the output;
— *Correctness*: The output is distributed according to the prescribed functionality;
— *Independence*: One party cannot make their inputs depend on the other parties' inputs;
— *Delivery*: An adversary cannot prevent the honest parties from successfully computing the functionality;
— *Fairness*: If one party receives output then so do all.

Any multi-party computation can be securely computed [Yao86; GMW87b; Gol04; BGW88; CCD88] as long as there is a honest majority [Lin08]. In the case where there is no such majority, and in particular in the two-party case, it is (in general [11]) impossible to achieve both fairness and guaranteed output delivery [Lin08; Cle86].

4.2.1.1 Weakening Fairness.

To circumvent this limitation, several authors have put forth alternatives to fairness that try and capture the practical context (*e.g.* contract-signing, bank transactions, etc.). Three main directions have been explored:

1. *Gradual release models*: The output is not revealed all at once, but rather released gradually (*e.g.* bit per bit) so that, if an abort occurs, then the adversary has not learnt much more about the output than the honest party. This solution is unsatisfactory because it is expensive and may not work if the adversary is more computationally powerful [GHKL08; GL91; Pin03; GMPY06].
2. *Optimistic models*: A trusted server is setup but will not be contacted unless fairness is breached. The server is able to restore fairness afterwards, and this approach can be efficient, but the infrastructure requirements and the condition that the server be trusted limit the applicability of this solution [Mic03; ASW97; CC00]. In particular, the dispute-resolving third party must be endowed with functions beyond those usually required of a normal certification authority.
3. *Legally fair, or concurrent model*: The first party to receive output obtains an information dubbed the "keystone". The keystone by itself gives nothing and so if the first party aborts after receiving it, no damage has been done – if the second party aborts after receiving the result (say, a signature) then the first party is left with a useless keystone. But, as observed in [CKP04] for the signature to be enforced, it needs to be presented to a court of law, and legally fair signing protocols are designed so that this signature *and* the keystone give enough information to reconstruct the missing data. Therefore, if the cheating party wishes to enforce its signed contract in a court of law, it by doing so reveal the signature that the first party should receive, thereby restoring fairness [CKP04]. Legal fairness requires neither a trusted arbitrator nor a high degree of interaction between parties.

Lindell [Lin08] also introduces a notion of "legally enforceable fairness" that sits between legal fairness and optimistic models: a trusted authority may force a cheating party to act in some fashion, should their cheating be attested. In this case the keystone consists in a digitally signed cheque for an frighteningly high amount of money that the cheating party would have to pay if the protocol were to be aborted prematurely and the signature abused.

Concurrent Signatures. Chen *et al.* [CKP04] proposed a legally fair signature scheme based on ring signatures [RST01; AOS02] and designated verifier signatures [JSI96], that is proven secure in the Random Oracle Model assuming the hardness of computing discrete logarithms.

11. See [GHKL08] for a very specific case where completely fair two-party computation can be achieved.

Concurrent signatures rely on a property shared by ring and designated verifier signatures called "ambiguity". In the case of two-party ring signatures, one cannot say which of the two parties produced the signature – since either of two parties could have produced such an ambiguous signature, both parties can deny having produced it. However, within the ring, if A receives a signature then she knows that it is B who sent it. The idea is to put the ambiguity-lifting information in a "keystone". When that keystone is made public, both signatures become simultaneously binding.

Concurrent signatures schemes can achieve legal fairness depending on the context. However their construction is not *abuse-free* [BW00; GJM99]: the party A holding the keystone can always determine whether to complete or abort the exchange of signatures, and can demonstrate this by showing an outside party the signature from B with the keystone, before revealing the keystone to B.

Our Results. In this work we describe a new contract signing protocol that achieves legal fairness and abuse-freeness. This protocol is based on the well-known Schnorr signature protocol, and produces signatures *compatible* with standard Schnorr signatures. For this reason, and as we demonstrate, the new contract signing protocol is provably secure in the random oracle model under the hardness assumption of solving the discrete logarithm problem. Our construction can be adapted to other DLP schemes, such as most [12] of those enumerated in [HPM94], including Girault-Poupard-Stern [GPS06] and ElGamal [ElG84].

4.2.2 Preliminaries

4.2.2.1 Schnorr Signatures

Schnorr digital signatures [Sch90] are an offspring of ElGamal [ElG84] signatures. This family of signatures is obtained by converting interactive identification protocols (zero-knowledge proofs) into transferable proofs of interaction (signatures). This conversion process, implicitly used by ElGamal, was discovered by Feige, Fiat and Shamir [FFS88] and formalized by Abdalla, Bellare and Namprempre [AABN02].

Throughout this section, we will refer to the original Schnorr signature protocol as "classical" Schnorr. This protocol consists in four algorithms:

— $\mathsf{Setup}(\ell)$: On input a security parameter ℓ, this algorithm selects large primes p, q such that $q \geq 2^\ell$ and $p - 1 \bmod q = 0$, as well as an element $g \in \mathbb{G}$ of order q in some multiplicative group $\mathbb{G}$ of order p, and a hash function $H : \{0,1\}^* \to \{0,1\}^\ell$. The output is a set of public parameters $\mathsf{pp} = (p, q, g, \mathbb{G}, H)$.

— $\mathsf{KeyGen}(\mathsf{pp})$: On input the public parameters, this algorithm chooses uniformly at random $x \xleftarrow{\$} \mathbb{Z}_q^\times$ and computes $y \leftarrow g^x$. The output is the couple $(\mathsf{sk}, \mathsf{pk})$ where $\mathsf{sk} = x$ is kept private, and $\mathsf{pk} = y$ is made public.

— $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)$: On input public parameters, a secret key, and a message m this algorithm selects a random $k \xleftarrow{\$} \mathbb{Z}_q^\times$, computes

$$\begin{aligned} r &\leftarrow g^k \\ e &\leftarrow H(m \| r) \\ s &\leftarrow k - ex \bmod q \end{aligned}$$

and outputs $\langle r, s \rangle$ as the signature of m.

— $\mathsf{Verify}(\mathsf{pp}, \mathsf{pk}, m, \sigma)$: On input the public parameters, a public key, a message and a signature $\sigma = \langle r, s \rangle$, this algorithm computes $e \leftarrow H(m, r)$ and returns True if and only if $g^s y^e = r$; otherwise it returns False.

The security of classical Schnorr signatures was analyzed by Pointcheval and Stern [PS96; PS00] using the Forking Lemma. Pointcheval and Stern's main idea is as follows: in the Random Oracle Model, the opponent can obtain from the forger two valid forgeries $\{\ell, s, e\}$ and $\{\ell, s', e'\}$ for the same oracle query

12. In a number of cases, *e.g.* DSA, the formulae of s do not lend themselves to security proofs.

$\{m, r\}$ but with different message digests $e \neq e'$. Consequently, $r = g^s y^{-e} = g^{s'} y^{-e'}$ and from that it becomes straightforward to compute the discrete logarithm of $y = g^x$. Indeed, the previous equation can be rewritten as $y^{e-e'} = g^{s'-s}$, and therefore:

$$y = g^{\frac{s'-s}{e-e'}} \Rightarrow \text{Dlog}_g(y) = \frac{s'-s}{e-e'}$$

The Forking Lemma for Schnorr signatures is originally stated as follows:

Theorem 4.2 (Forking Lemma, [PS00]) *Let $\mathcal{A}$ be an attacker which performs within a time bound t_F an existential forgery under an adaptively chosen-message attack against the Schnorr signature, with probability ϵ_F. Assume that $\mathcal{A}$ makes q_h hashing queries to a random oracle and q_s queries to a signing oracle.*

Then there exists an adversary solving the discrete logarithm problem in subgroups of prime order in polynomial expected time.

Assume that $\epsilon_F \geq 10(q_s+1)(q_s+q_h)/q$, then the discrete logarithm problem in subgroups of prime order can be solved within expected time less that $120686\, q_h t_F/\epsilon_F$.

This security reduction loses a factor $O(q_h)$ in the time-to-success ratio. Note that recent work by Seurin [Seu12] shows that this is essentially the best possible reduction to the discrete logarithm problem.

4.2.2.2 Concurrent Signatures

Let us give a more formal account of legal fairness as described in [CKP04; Lin08] in terms of concurrent signatures. Unlike classical contract-signing protocol, whereby contractors would exchange full-fledged signatures (*e.g.* [Gol83]), in a concurrent signature protocol there are "ambiguous" signatures that do not, as such, bind their author. This ambiguity can later be lifted by revealing some additional information: the "keystone". When the keystone is made public, both signatures become simultaneously binding.

Let $\mathcal{M}$ be a message space. Let $\mathcal{K}$ be the keystone space and $\mathcal{F}$ be the keystone fix space.

Definition 4.1 (Concurrent signature) *A concurrent signature is composed of the following algorithms:*

- $\mathsf{Setup}(k)$*: Takes a security parameter k as input and outputs the public keys (y_A, y_B) of all participants, a function $\mathsf{KeyGen} : \mathcal{K} \to \mathcal{F}$, and public parameters pp describing the choices of $\mathcal{M}, \mathcal{K}, \mathcal{F}$ and KeyGen.*
- $\mathsf{aSign}(y_i, y_j, x_i, h_2, M)$*: Takes as input the public keys y_1 and y_2, the private key x_i corresponding to y_i, an element $h_2 \in \mathcal{F}$ and some message $M \in \mathcal{M}$; and outputs an "ambiguous signature"*

  $$\sigma = \langle s, h_1, h_2 \rangle$$

  where $s \in \mathcal{S}, h_1, h_2 \in \mathcal{F}$.
- $\mathsf{aVerify}(\sigma, y_i, y_j, M)$*: Takes as input an ambiguous signature $\sigma = \langle s, h_1, h_2 \rangle$, public keys y_i and y_j, a message M; and outputs a Boolean value, with the constraint that*

  $$\mathsf{aVerify}\,(\sigma', y_j, y_i, M) = \mathsf{aVerify}\,(\sigma, y_i, y_j, M)$$

  where $\sigma' = \langle s, h_2, h_1 \rangle$.
- $\mathsf{Verify}(k, \sigma, y_i, y_j, M)$*: Takes as input $k \in \mathcal{K}$ and σ, y_i, y_j, M as above; and checks whether $\mathsf{KeyGen}(k) = h_2$: If not it terminates with output False, otherwise it outputs the result of $\mathsf{aVerify}(\sigma, y_i, y_j, M)$.*

A valid concurrent signature is a tuple $\langle k, \sigma, y_i, y_j, M \rangle$ that is accepted by the Verify algorithm. Concurrent signatures are used by two parties A and B in the following way:

1. A and B run Setup to determine the public parameters of the scheme. We assume that A's public and private keys are y_A and x_A, and B's public and private keys are y_B and x_B.
2. Without loss of generality, we assume that A initiates the conversation. A picks a random keystone $k \in \mathcal{K}$, and computes $f = \mathsf{KeyGen}(k)$. A takes her own public key y_A and B's public key y_B and picks a message $M_A \in \mathcal{M}$ to sign. A then computes her ambiguous signature to be

   $$\sigma_A = \langle s_A, h_A, f \rangle = \mathsf{aSign}(y_A, y_B, x_A, f, M_A).$$

3. Upon receiving A's ambiguous signature σ_A, B verifies the signature by checking that
$$\mathsf{aVerify}(s_A, h_A, f, y_A, y_B, M_A) = \mathsf{True}$$
If this equality does not hold, then B aborts. Otherwise B picks a message $M_B \in \mathcal{M}$ to sign and computes his ambiguous signature
$$\sigma_B = \langle s_B, h_B, f\rangle = \mathsf{aSign}(y_B, y_A, x_B, f, M_B)$$
then sends this back to A. Note that B uses the same value f in his signature as A did to produce σ_A.
4. Upon receiving B's signature σ_B, A verifies that
$$\mathsf{aVerify}(s_B, h_B, f, y_B, y_A, M_B) = \mathsf{True}$$
where f is the same keystone fix as A used in the previous steps. If the equality does not hold, then A aborts. Otherwise A sends keystone k to B.

At the end of this protocol, both $\langle k, \sigma_A\rangle$ and $\langle k, \sigma_B\rangle$ are binding, and accepted by the Verify algorithm.

Remark Note that A has an the upper hand in this protocol: Only when A releases the keystone do both signatures become simultaneously binding, and there is no guarantee that A will ever do so. Actually, since A controls the timing of the keystone release (if it is released at all), A may only reveal k to a third party C but withhold it from B, and gain some advantage by doing so. In other terms, concurrent signatures can be *abused* by A [BW00; GJM99].

Chen *et al.* [CKP04] argue that there are situations where it is not in A's interest to try and cheat B, in which abuse-freeness is not necessary. One interesting scenario is credit card payment in the "four corner" model. Assume that B's signature is a payment to A. To obtain payment, A must channel *via* her acquiring bank C, which would communicate with B's issuing bank D. D would ensure that B receives both the signature and the keystone — as soon as this happens A is bound to her signature. Since in this scenario there is no possibility for A to keep B's signature private, fairness is eventually restored.

Example 4.1 *A concurrent signature scheme based on the ring signature algorithm of Abe* et al. *[AOS02] was proposed by Chen* et al. *[CKP04]:*

— Setup*: On input a security parameter ℓ, two large primes p and q are selected such that $q|p-1$. An element $g \in \mathbb{Z}_p^\times$ of order q is selected. The spaces $\mathcal{S} = \mathcal{F} = \mathbb{Z}_q$ and $\mathcal{M} = \mathcal{K} = \{0,1\}^*$ are chosen. Two cryptographic hash functions $H_1, H_2 : \{0,1\}^* \to \mathbb{Z}_q$ are selected and we set* KeyGen $= H_1$*. Private keys x_A, x_B are selected uniformly at random from $\mathbb{Z}_q$ and the corresponding public keys are computed as $g^{x_i} \bmod p$.*

— aSign*: The algorithms takes as input y_i, y_j, x_i, h_2, M, verifies that $y_i \neq y_j$ (otherwise aborts), picks a random value $t \in \mathbb{Z}_q$ and computes*
$$h = H_2\left(g^t y_j^{h_2} \bmod p \| M\right)$$
$$h_1 = h - h_2 \bmod q$$
$$s = t - h_1 x_i \bmod q$$
where $\|$ denotes concatenation. The algorithm outputs $\langle s, h_1, h_2\rangle$.

— aVerify*: This algorithm takes as input s, h_1, h_2, y_i, y_j, M and checks whether the following equation holds:*
$$h_1 + h_2 = H_2\left(g^s y_i^{h_1} y_j^{h_2} \bmod p \| M\right) \bmod q$$

The security of this scheme can be proven in the Random Oracle model assuming the hardness of computing discrete logarithms in $\mathbb{Z}_p^\times$.

4.2.2.3 Legal Fairness for Concurrent Signatures

A concurrent signature scheme is secure when it achieves existential unforgeability, ambiguity and fairness against an active adversary that has access to a signature oracle. We define these notions in terms of games played between the adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. In all security games, $\mathcal{A}$ can perform any number of the following queries:

— KeyGen queries: $\mathcal{A}$ can receive a keystone fix $f = \mathsf{KeyGen}(k)$ where k is chosen by the challenger [13].
— KeyReveal queries: $\mathcal{A}$ can request that $\mathcal{C}$ reveals which k was chosen to produce a keystone fix f in a previous KeyGen query. If f was not a previous KeyGen query output then $\mathcal{C}$ returns $\perp$.
— aSign queries: $\mathcal{A}$ can request an ambiguous signature for any message of his choosing and any pair of users [14].
— SKExtract queries: $\mathcal{A}$ can request the private key corresponding to a public key.

Definition 4.2 (Unforgeability) *The notion of existential unforgeability for concurrent signatures is defined in terms of the following security game:*

1. *The* Setup *algorithm is run and all public parameters are given to $\mathcal{A}$.*
2. *$\mathcal{A}$ can perform any number of queries to $\mathcal{C}$, as described above.*
3. *Finally, $\mathcal{A}$ outputs a tuple $\sigma = \langle s, h_1, f\rangle$ where $s \in \mathcal{S}, h_1, f \in \mathcal{F}$, along with public keys y_C, y_D and a message $M \in \mathcal{M}$.*

$\mathcal{A}$ wins the game if aVerify *accepts σ and either of the following holds:*
— *$\mathcal{A}$ did not query SKExtract on y_C nor on y_D, and did not query aSign on (y_C, y_D, f, M) nor on (y_D, y_C, h_1, M).*
— *$\mathcal{A}$ did not query aSign on (y_C, y_i, f, M) for any $y_i \neq y_C$, and did not query SKExtract on y_C, and f is the output of* KeyGen*: either an answer to a KeyGen query, or $\mathcal{A}$ can produce a k such that $k = \mathsf{KeyGen}(k)$.*

The last constraint in the unforgeability security game corresponds to the situation where $\mathcal{A}$ knows one of the private keys (as is the case if $\mathcal{A} = A$ or B).

Definition 4.3 (Ambiguity) *The notion of ambiguity for concurrent signatures is defined in terms of the following security game:*

1. *The* Setup *algorithm is run and all public parameters are given to $\mathcal{A}$.*
2. *Phase 1: $\mathcal{A}$ can perform any number of queries to $\mathcal{C}$, as described above.*
3. *Challenge: $\mathcal{A}$ selects a challenge tuple (y_i, y_j, M) where y_i, y_j are public keys and $M \in \mathcal{M}$. In response, $\mathcal{C}$ selects a random $k \in \mathcal{K}$, a random $b \in \{0, 1\}$ and computes $f = \mathsf{KeyGen}(k)$. If $b = 0$, then $\mathcal{C}$ outputs*

$$\sigma_1 = \langle s_1, h_1, f\rangle = \mathsf{aSign}(y_i, y_j, x_i, f, M)$$

Otherwise, if $b = 1$ then $\mathcal{C}$ computes

$$\sigma_2 = \langle s_2, h_2, f\rangle = \mathsf{aSign}(y_j, y_i, x_i, f, M)$$

but outputs $\sigma_2' = \langle s_2, f, h_2\rangle$ instead.
4. *Phase 2: $\mathcal{A}$ can perform any number of queries to $\mathcal{C}$, as described above.*
5. *Finally, $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$.*

$\mathcal{A}$ wins the game if $b = b'$ and if $\mathcal{A}$ made no KeyReveal query on f, h_1 or h_2.

Definition 4.4 (Fairness) *The notion of fairness for concurrent signatures is defined in terms of the following security game:*

1. *The* Setup *algorithm is run and all public parameters are given to $\mathcal{A}$.*
2. *$\mathcal{A}$ can perform any number of queries to $\mathcal{C}$, as described above.*
3. *Finally, $\mathcal{A}$ chooses two public keys y_C, y_D and outputs $k \in \mathcal{K}$ and $S = (s, h_1, f, y_C, y_D, M)$ where $s \in \mathcal{S}$, $h_1, f \in \mathcal{F}$, $M \in \mathcal{M}$.*

$\mathcal{A}$ wins the game if $\mathsf{aVerify}(S)$ *accepts and either of the following holds:*
— *f was output from a KeyGen query, no KeyReveal query was made on f, and* Verify *accepts $\langle k, S\rangle$.*
— *$\mathcal{A}$ can output $S' = (s', h_1', f, y_D, y_C, M')$ where* $\mathsf{aVerify}(S')$ *accepts and* $\mathsf{Verify}(k, S)$ *accepts, but* $\mathsf{Verify}(k, S')$ *rejects.*

This definition of fairness formalizes the idea that B cannot be left in a position where a keystone binds his signature to him while A's initial signature is not also bound to A. It does not, however, guarantee that B will ever receive the necessary keystone.

13. The algorithm KeyGen being public, $\mathcal{A}$ can compute $\mathsf{KeyGen}(k)$ for any k of her choosing.
14. Note that with this information and using KeyGen queries, $\mathcal{A}$ can obtain concurrent signatures for any message and any user pair.

4.2.3 Legally Fair Co-Signatures

4.2.3.1 Legal Fairness Without Keystones

The main idea builds on the following observation: Every signature exchange protocol is plagued by the possibility that the last step of the protocol is not performed. Indeed, it is in the interest of a malicious party to get the other party's signature without revealing its own. As a result, the best one can hope for is that a trusted third party can eventually restore fairness.

To avoid this destiny, the proposed paradigm does *not* proceed by sending A's signature to B and vice versa. Instead, we construct a *joint signature*, or *co-signature*, of both A and B. By design, there are no signatures to steal — and stopping the protocol early does not give the stopper a decisive advantage. More precisely, the contract they have agreed upon is the best thing an attacker can gather, and if she ever wishes to enforce this contract by presenting it to a court of law, she would confirm her own commitment to it as well as the other party's. Therefore, if one can construct co-signatures without intermediary individual signatures being sent, legal fairness can be achieved without keystones.

Since keystones can be used by the party having them to abuse the other [CKP04], the co-signature paradigm provides an interesting alternative to concurrent signatures.

Schnorr Co-signatures. To illustrate the new paradigm, we now discuss a legally fair contract-signing protocol built from the well-known Schnorr signature protocol, that produces signatures *compatible* with standard Schnorr signatures. This contract signing protocol is provably secure in the random oracle model under the hardness assumption of solving the discrete logarithm problem.

The construction can be adapted to other DLP schemes, such as most [15] of those enumerated in [HPM94], including Girault-Poupard-Stern [GPS06] and ElGamal [ElG84].

— Setup: An independent (not necessarily trusted) authority generates a classical Schnorr parameter-set p, q, g which is given to A and B. Each user U generates a usual Schnorr public key $y_U = g^{x_U}$ and publishes y_U on a public directory $\mathcal{D}$ (see Figure 4.3). To determine the co-signature public-key $y_{A,B}$ of the pair $\langle A, B\rangle$, a verifier consults $\mathcal{D}$ and simply computes $y_{A,B} = y_A \times y_B$. Naturally, $y_{A,B} = y_{B,A}$.

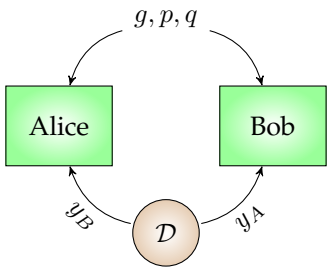


Figure 4.3 – Public directory $\mathcal{D}$ distributing the public keys.

— Cosign: To co-sign a message m, A and B compute a common r and a common s, one after the other. Without loss of generality we assume that B initiates the co-signature.
  — During the first phase (Figure 4.4), B chooses a private random number k_B and computes $r_B \leftarrow g^{k_B}$. He commits to that value by sending to A a message digest $\rho \leftarrow H(0\|r_B)$. A chooses a private random number k_A, computes $r_A \leftarrow g^{k_A}$ and sends r_A to B. B replies with r_B, which A checks against the earlier commitment ρ. Both parties compute $r \leftarrow r_A \times r_B$, and $e \leftarrow H(1\|m\|r)$, where m is the message to be co-signed.
  — During the second phase of the protocol, B sends $s_B \leftarrow k_B - e \times x_B \bmod q$ to A. A replies with $s_A \leftarrow k_A - e \times x_A \bmod q$. Both users compute $s \leftarrow s_A + s_B \bmod q$.

15. In a number of cases, *e.g.* DSA, the formulae of s do not lend themselves to security proofs.

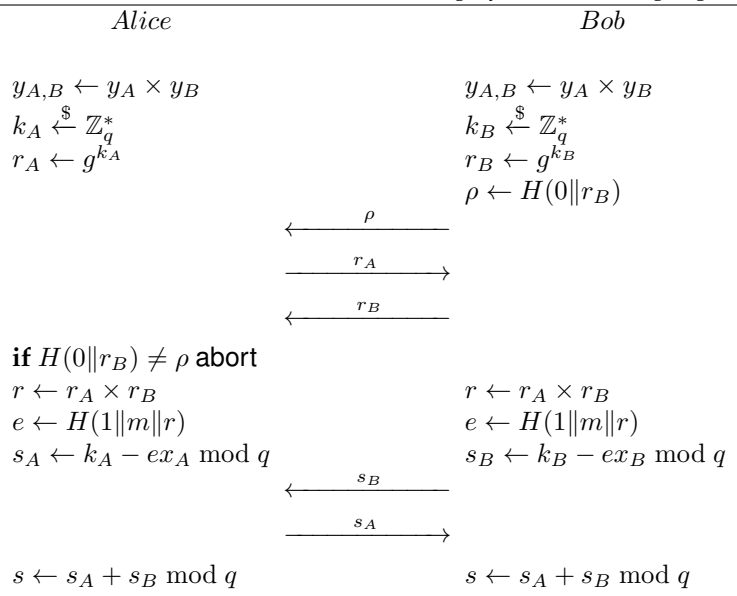


Figure 4.4 – Generating the Schnorr co-signature of message m.

— Verify: As in the classical Schnorr signature, the co-signature $\{r, s\}$ is checked for a message m by computing $e \leftarrow H(m\|r)$, and checking whether $g^s y^e = r$ (Figure 4.5). If the equality holds, then the co-signature binds both A and B to m; otherwise neither party is tied to m.

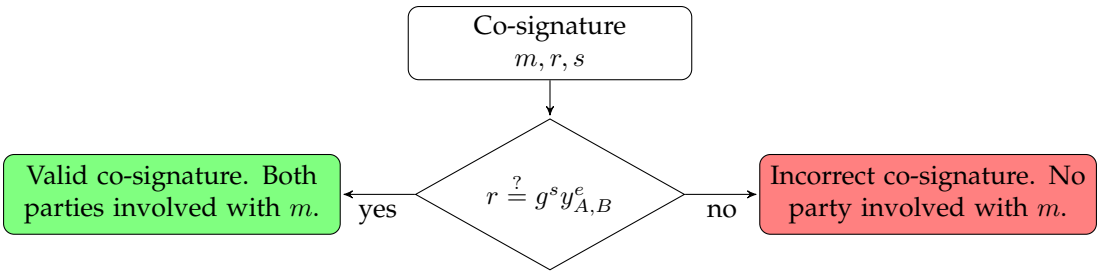


Figure 4.5 – Verification of a Schnorr co-signature m, r, s.

Remark Note that during the co-signature protocol, A might decide not to respond to B: In that case, A would be the only one to have the complete co-signature. This is a breach of fairness insofar as A can benefit from the co-signature and not B, but the protocol is abuse-free: A cannot use the co-signature as a proof that B, and B alone, committed to m. Furthermore, it is not a breach of legal fairness: If A presents the co-signature in a court of law, she *ipso facto* reveals her commitment as well.

Remark In a general fair-contract signing protocol, A and B can sign different messages m_A and m_B. Using the co-signature construction requires that A and B agree first on the content of a single message m.

Security Analysis The security of the co-signature scheme essentially builds on the unforgeability of classical Schnorr signatures. Since there is only one co-signature output, the notion of ambiguity does not apply *per se* — albeit we will come back to that point later on. The notion of fairness is structural in the fact that a co-signature, as soon as it is binding, is binding for *both* parties.

As for concurrent signatures, an adversary $\mathcal{A}$ has access to an unlimited amount of conversations and valid co-signatures, *i.e.* $\mathcal{A}$ can perform the following queries:

— Hash queries: $\mathcal{A}$ can request the value of $H(x)$ for a x of its choosing.

— CoSign queries: $\mathcal{A}$ can request a valid co-signature r, s for a message m and a public key $y_{C,D}$ of its choosing.
— Transcript queries: $\mathcal{A}$ can request a valid transcript $(\rho, r_C, r_D, s_C, s_D)$ of the co-signing protocol for a message m of its choosing, between users C and D of its choosing.
— SKExtract queries: $\mathcal{A}$ can request the private key corresponding to a public key.
— Directory queries: $\mathcal{A}$ can request the public key of any user U.

The following definition captures the notion of unforgeability in the co-signing context:

Definition 4.5 (Unforgeability) *The notion of unforgeability for co-signatures is defined in terms of the following security game between the adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

1. *The* Setup *algorithm is run and all public parameters are provided to $\mathcal{A}$.*
2. *$\mathcal{A}$ can perform any number of queries to $\mathcal{C}$, as described above.*
3. *Finally, $\mathcal{A}$ outputs a tuple $(m, r, s, y_{C,D})$.*

$\mathcal{A}$ wins the game if $\mathsf{Verify}(m, r, s) = \mathsf{True}$ *and there exist public keys $y_C, y_D \in \mathcal{D}$ such that $y_{C,D} = y_C y_D$ and either of the following holds:*
— *$\mathcal{A}$ did not query SKExtract on y_C nor on y_D, and did not query CoSign on $m, y_{C,D}$, and did not query Transcript on m, y_C, y_D nor m, y_D, y_C.*
— *$\mathcal{A}$ did not query Transcript on m, y_C, y_i for any $y_i \neq y_C$ and did not query SKExtract on y_C, and did not query CoSign on m, y_C, y_i for any $y_i \neq y_C$.*
We shall say that a co-signature scheme is unforgeable when the success probability of $\mathcal{A}$ in this game is negligible.

To prove that the Schnorr-based scheme described above is secure we use the following strategy: Assuming an efficient forger $\mathcal{A}$ for the co-signature scheme, we turn it into an efficient forger $\mathcal{B}$ for Schnorr signatures, then invoke the Forking Lemma to prove the existence of an efficient solver $\mathcal{C}$ for the discrete logarithm problem. All proofs hold in the Random Oracle model.

Since the co-signing protocol gives the upper hand to the last-but-one speaker there is an asymmetry: Alice has more information than Bob. Therefore we address two scenarios: When the attacker plays Alice's role, and when the attacker plays Bob's.

Theorem 4.3 *Let $\{y, g, p, q\}$ be a DLP instance. If $\mathcal{A}$ plays the role of Bob (resp. Alice) and is able to forge in polynomial time a co-signature with probability ϵ_F, then in the Random Oracle model $\mathcal{A}$ can break the DLP instance with high probability in polynomial time.*

The proof of Theorem 4.3, proceeds by splitting Theorem 4.3 in twain depending on whether $\mathcal{A}$ impersonates Bob or Alice:

Adversary Attacks Bob

Theorem 4.4 *Let $\{y, g, p, q\}$ be a DLP instance. If $\mathcal{A}_{Alice}$ plays the role of Alice and is able to forge in polynomial time a co-signature with probability ϵ_F, then in the Random Oracle model $\mathcal{A}_{Alice}$ can break that DLP instance with high probability in polynomial time.*

Proof: The proof consists in constructing a simulator $\mathcal{S}_{\text{Bob}}$ that interacts with the adversary and forces it to actually produce a classical Schnorr forgery. Here is how this simulator behaves at each step of the protocol.

1. *Key Establishment Phase:*
   $\mathcal{S}_{\text{Bob}}$ is given a target DLP instance $\{y, g, p, q\}$. As a simulator, $\mathcal{S}_{\text{Bob}}$ emulates not only Bob, but also all oracles and the directory $\mathcal{D}$ (see Figure 4.6).
   $\mathcal{S}_{\text{Bob}}$ injects the target y into the game, namely by posting in the directory the "public-key" $y_B \leftarrow y \times y_A^{-1}$.
   To inject a target DLP instance $y \leftarrow g^x$ into $\mathcal{A}$, the simulator $\mathcal{S}_{\text{Bob}}$ reads $y_{\mathcal{A}}$ from the public directory and poses as an entity whose public-key is $y_{\mathcal{S}} \leftarrow y \times y_{\mathcal{A}}^{-1}$. It follows that $y_{\mathcal{A},\mathcal{S}}$, the common public-key of $\mathcal{A}$ and $\mathcal{S}$ will be precisely $y_{\mathcal{A},\mathcal{S}} \leftarrow y_{\mathcal{S}} \times y_{\mathcal{A}}$ which, by construction, is exactly y.

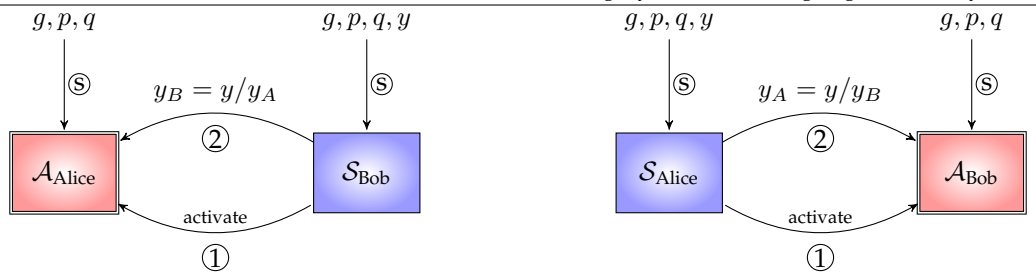


Figure 4.6 – The simulator $\mathcal{S}_{\text{Bob}}$ (left) or $\mathcal{S}_{\text{Alice}}$ (right) answers the attacker's queries to the public directory $\mathcal{D}$.

Then $\mathcal{S}_{\text{Bob}}$ activates $\mathcal{A}_{\text{Alice}}$, who queries the directory and gets y_B. At this point in time, $\mathcal{A}_{\text{Alice}}$ is tricked into believing that she has successfully established a common co-signature public-key set $\{g, p, q, y\}$ with the "co-signer" $\mathcal{S}_{\text{Bob}}$.

2. *Query Phase:*

   $\mathcal{A}_{\text{Alice}}$ will now start to present queries to $\mathcal{S}_{\text{Bob}}$. In a "normal" attack, $\mathcal{A}_{\text{Alice}}$ and Bob would communicate with a random oracle $\mathcal{O}$ representing the hash function H. However, here, the simulator $\mathcal{S}_{\text{Bob}}$ will play $\mathcal{O}$'s role and answer $\mathcal{A}_{\text{Alice}}$'s hashing queries.

   $\mathcal{S}_{\text{Bob}}$ must respond to three types of queries: *hashing queries*, *co-signature queries* and *transcript queries*. $\mathcal{S}_{\text{Bob}}$ will maintain an oracle table T containing all the hashing queries performed throughout the attack. At start $T \leftarrow \varnothing$. When $\mathcal{A}_{\text{Alice}}$ submits a hashing query q_i to $\mathcal{S}_{\text{Bob}}$, $\mathcal{S}_{\text{Bob}}$ answers as shown in Algorithm 1.

Algorithm 1: Hashing oracle simulation.

Input: A hashing query q_i from $\mathcal{A}$
if $\exists e_i, \{q_i, e_i\} \in T$ **then**
  $\rho \leftarrow e_i$
else
  $\rho \xleftarrow{\$} \mathbb{Z}_q^{\times}$
  Append $\{q_i, \rho\}$ to T
end if
return ρ

When $\mathcal{A}_{\text{Alice}}$ submits a co-signature query to $\mathcal{S}_{\text{Bob}}$, $\mathcal{S}_{\text{Bob}}$ proceeds as explained in Algorithm 2.

Algorithm 2: Co-signing oracle simulation.

Input: A co-signature query m from $\mathcal{A}_{\text{Alice}}$
$s_B, e \xleftarrow{\$} \mathbb{Z}_q^*$
$r_B \leftarrow g^{s_B} y^e$
Send $H(0 \| r_B)$ to $\mathcal{A}_{\text{Alice}}$
Receive r_A from $\mathcal{A}_{\text{Alice}}$
$r \leftarrow r_A \times r_B$
$u \leftarrow 1 \| m \| r$
if $\exists e' \neq e, \{u, e'\} \in T$ **then**
  abort
else
  Append $\{u, e\}$ to T
end if
return s_B

Finally, when $\mathcal{A}_{\text{Alice}}$ requests a conversation transcript, $\mathcal{S}_{\text{Bob}}$ replies by sending $\{m, \rho, r_A, r_B, s_B, s_A\}$ from a previously successful interaction.

3. *Output Phase:*

   After performing queries, $\mathcal{A}_{\text{Alice}}$ eventually outputs a co-signature m, r, s valid for $y_{\mathcal{A},\mathcal{S}}$ where $r = r_A r_B$ and $s = s_A + s_B$. By design, these parameters are those of a classical Schnorr signature and therefore $\mathcal{A}_{\text{Alice}}$ has produced a classical Schnorr forgery.

To understand $\mathcal{S}_{\text{Bob}}$'s co-signature reply (Algorithm 2), assume that $\mathcal{A}_{\text{Alice}}$ is an honest Alice who plays by the protocol's rules. For such an Alice, $\{s, r\}$ is a valid signature with respect to the common co-signature public-key set $\{g, p, q, y\}$. There is a case in which $\mathcal{S}_{\text{Bob}}$ aborts the protocol before completion: this happens when it turns out that $r_A \times r_B$ has been previously queried by $\mathcal{A}_{\text{Alice}}$. In that case, it is no longer possible for $\mathcal{S}_{\text{Bob}}$ to reprogram the oracle, which is why $\mathcal{S}_{\text{Bob}}$ must abort. Since $\mathcal{A}_{\text{Alice}}$ does not know the random value of r_B, such a bad event would only occur with a negligible probability exactly equal to q_h/q (where q_h is the number of queries to the hashing oracle).

Therefore, $\mathcal{A}_{\text{Alice}}$ is turned into a forger for the target Schnorr instance with probability $1 - q_h/q$. Since $\mathcal{A}_{\text{Alice}}$ succeeds with probability ϵ_F, $\mathcal{A}_{\text{Alice}}$'s existence implies the existence of a Schnorr signature forger of probability $\epsilon_S = (1 - q_h/q)\epsilon_F$, which by the Forking Lemma shows that there exists a polynomial adversary breaking the chosen DLP instance with high probability. □

Being an attacker, at some point $\mathcal{A}_{\text{Alice}}$ will output a forgery $\{m', r', s'\}$. From here on we use the Forking Lemma and transform $\mathcal{A}_{\text{Alice}}$ into a DLP solver as described by Pointcheval and Stern in [PS00, Theorem 14].

Adversary Attacks Alice. The case where $\mathcal{A}$ targets A is similar but somewhat simpler, and the proof follows the same strategy.

Theorem 4.5 *Let $\{y, g, p, q\}$ be a DLP instance. If $\mathcal{A}_{Bob}$ plays the role of Bob and is able to forge a co-signature with probability ϵ_F, then in the Random Oracle model $\mathcal{A}_{Bob}$ can break that DLP instance with high probability in polynomial time.*

Proof: [Theorem 4.5] Here also the proof consists in constructing a simulator, $\mathcal{S}_{\text{Alice}}$, that interacts with the adversary and forces it to actually produce a classical Schnorr forgery. The simulator's behavior at different stages of the security game is as follows:

1. *The Key Establishment Phase:*

   $\mathcal{S}_{\text{Alice}}$ is given a target DLP instance $\{y, g, p, q\}$. Again, $\mathcal{S}_{\text{Alice}}$ impersonates not only Alice, but also $\mathcal{O}$ and $\mathcal{D}$. $\mathcal{S}_{\text{Alice}}$ injects the target y into the game as described in Section 4.2.3.1. Now $\mathcal{S}_{\text{Alice}}$ activates $\mathcal{A}_{\text{Bob}}$, who queries $\mathcal{D}$ (actually controlled by $\mathcal{S}_{\text{Alice}}$) to get y_B. $\mathcal{A}_{\text{Bob}}$ is thus tricked into believing that it has successfully established a common co-signature public-key set $\{g, p, q, y\}$ with the "co-signer" $\mathcal{S}_{\text{Alice}}$.

2. *The Query Phase:*

   $\mathcal{A}_{\text{Bob}}$ will now start to present queries to $\mathcal{S}_{\text{Alice}}$. Here as well, $\mathcal{S}_{\text{Alice}}$ will play $\mathcal{O}$'s role and will answer $\mathcal{A}_{\text{Bob}}$'s hashing queries.

   Again, $\mathcal{S}_{\text{Alice}}$ must respond to hashing queries and co-signature queries. Hashing queries are answered as shown in Algorithm 1. When $\mathcal{A}_{\text{Bob}}$ submits a co-signature query to $\mathcal{S}_{\text{Alice}}$, $\mathcal{S}_{\text{Alice}}$ proceeds as explained in Algorithm 8.

   $\mathcal{S}_{\text{Alice}}$ controls the oracle $\mathcal{O}$, and as such knows what is the value of r_B that $\mathcal{A}_{\text{Bob}}$ is committed to. The simulator is designed to trick $\mathcal{A}_{\text{Bob}}$ into believing that this is a real interaction with Alice, but Alice's private key is not used.

3. *Output*:

   Eventually, $\mathcal{A}_{\text{Bob}}$ produces a forgery that is a classical Schnorr forgery $\{m, r, s\}$.

Algorithm 8 Co-signing oracle simulation for $\mathcal{S}_{\text{Alice}}$.

Input: A co-signature query m from $\mathcal{A}_{\text{Bob}}$
Receive ρ from $\mathcal{A}_{\text{Bob}}$
Query T to retrieve r_B such that $H(0\|r_B) = \rho$
$e, s_A \xleftarrow{\$} \mathbb{Z}_q$
$r \leftarrow r_B g^{s_A} y^e$
$u \leftarrow 1\|m\|r$
if $\exists e' \neq e, \{u, e'\} \in T$ **then**
  abort
else
  Append $\{u, e\}$ to T
end if
$r_A \leftarrow r \times r_B^{-1}$
Send r_A to $\mathcal{A}_{\text{Bob}}$
Receive r_B from $\mathcal{A}_{\text{Bob}}$; this r_B is not used by $\mathcal{S}_{\text{Alice}}$
Receive s_B from $\mathcal{A}_{\text{Bob}}$
return s_A

Algorithm 8 may fail with probability $1/q$. Using the Forking Lemma again, we transform $\mathcal{A}_{\text{Bob}}$ into an efficient solver of the chosen DLP instance. □

4.2.3.2 Concurrent Co-signatures

4.2.3.2.1 Proofs of Involvement. We now address a subtle weakness in the protocol described in the previous section, which is not captured by the fairness property *per se* and that we refer to as the existence of "proofs of involvement". Such proofs are not valid co-signatures, and would not normally be accepted by verifiers, but they nevertheless are valid evidence establishing that one party committed to a message. In a legally fair context, it may happen that such evidence is enough for one party to win a trial against the other — who lacks both the co-signature, and a proof of involvement.

Example 4.2 *In the co-signature protocol of Figure 4.4, s_B is not a valid Schnorr signature for Bob. Indeed, we have $g^{s_B} y_B^e = r_B \neq r$. However, Alice can construct $s' = s_B - k_A$, so that m, r, s' forms a valid classical signature of Bob* alone *on m.*

Example 4.2 illustrates the possibility that an adversary, while unable to forge a co-signature, may instead use the information to build a valid (mono-) signature. Note that Alice may opt for a weaker proof of involvement, for instance by demonstrating her possession of a valid signature using any zero-knowledge protocol.

A straightforward patch is to refrain from using the public keys y_A, y_B for both signature and co-signature — so that attempts at constructing proofs of involvement become vain. For instance, every user could have a key $y_U^{(1)}$ used for classical signature and for certifying a key $y_U^{(2)}$ used for co-signature [16]. If an adversary generates a classical signature from a co-signature transcript as in Example 4.2, she actually reveals her harmful intentions.

However, while this exposes the forgery — so that honest verifiers would reject such a signature — the perpetrator remains anonymous. There are scenarios in which this is not desirable, *e.g.* because it still proves that B agreed (with some unknown and dishonest partner) on m.

Note that the existence of proof of involvement is not necessary and depends on the precise choice of underlying signature scheme.

16. The key $y_U^{(2)}$ may be derived from $y_U^{(1)}$ in some way, so that the storage needs of $\mathcal{D}$ are the same as for classical Schnorr.

4.2.3.3 Security Model

It is important to make extremely clear the security model that we are targeting. In this situation an adversary $\mathcal{A}$ (possibly Alice or Bob) tries to forged signatures from partial and/or complete traces of co-signature interactions, which can be of two kinds :

1. Co-signatures between two parties, at least one of which did not take part in the co-signature protocol;
2. (Traditional) signatures of either party.

$\mathcal{A}$ succeeds if and only if one of these forgeries is accepted, which can be captured as the probability of acceptance of $\mathcal{A}$'s outputs, and the victim (purported mono-signatory, or co-signatory) doesn't have a co-signature with $\mathcal{A}$ [17].

Observe that due to the unforgeability of Schnorr signatures, the attacker must necessarily impersonate one of the co-signatories to achieve either of the two forgeries mentioned above (in fact, the strongest position is that of Alice, who has an edge over Bob in the protocol). This is the reason why the victim may have a co-signature of $\mathcal{A}$, so that this security model captures fairness.

In short, we propose to address such attacks in the following way:

1. By using a different key for co-signature and mono-signature;
2. By having Bob store specific co-signature-related information in non-volatile memory.

The reason for (1) is that it distinguishes between mono-signatures, and mono-signatures generated from partial co-signature traces. Thanks to this, it is easy for the verifier to detect a forgery, and perform additional steps.

The reason for (2) is twofold: On the one hand, it enables the verifier to obtain from Bob definitive proof that there was forgery; on the other hand, once the forgery has been identified, it makes it possible for the verifier to re-establish fairness binding the two real co-signatories together. Note that Bob is in charge of keeping this information secure, i.e. available and correct.

4.2.3.4 Concurrent Co-signatures

In the interest of fairness, the best we can ask is that if A tries to incriminate B on a message they both agreed upon, she cannot do so anonymously.

To enforce fairness on the co-signature protocol, we ask that the equivalent of a keystone is transmitted first; so that in case of dispute, the aggrieved party has a legal recourse. First we define the notion of an authorized signatory credential:

Definition 4.6 (Authorized signatory credential) *The data field*

$$\Gamma_{Alice,Bob} = \{Alice, Bob, k_A, \sigma(g^{k_A} \| Alice \| Bob)\}$$

is called an authorized signatory credential *given by Alice to Bob, where σ is some publicly known auxiliary signature algorithm using Alice's private key x_A as a signing key.*

Any party who gets $\Gamma_{\text{Alice,Bob}}$ can check its validity, and releasing $\Gamma_{\text{Alice,Bob}}$ is *by convention* functionally equivalent to Alice giving her private key x_A to Bob. A valid signature by Bob on a message m exhibited with a valid $\Gamma_{\text{Alice,Bob}}$ is *legally* defined as encompassing the meaning ($\Rrightarrow$) of Alice's signature on m:

$$\{\Gamma_{\text{Alice,Bob}}, \text{signature by Bob on } m\} \Rrightarrow \text{signature by Alice on } m$$

Second, the co-signature protocol of Figure 4.4 is modified by requesting that Alice provide t to Bob. Bob stores this in a local memory $\mathcal{L}$ along with s_B. Together, t and s_B act as a keystone enabling Bob (or a verifier, *e.g.* a court of law) to reconstruct $\Gamma_{\text{Alice,Bob}}$ if Alice exhibits a (fraudulent) signature binding Bob alone with his co-signing public key.

Therefore, should Alice try to exhibit as in Example 4.2 a signature of Bob alone on a message they both agreed upon (which is known as a fraud), the court would be able to identify Alice as the fraudster.

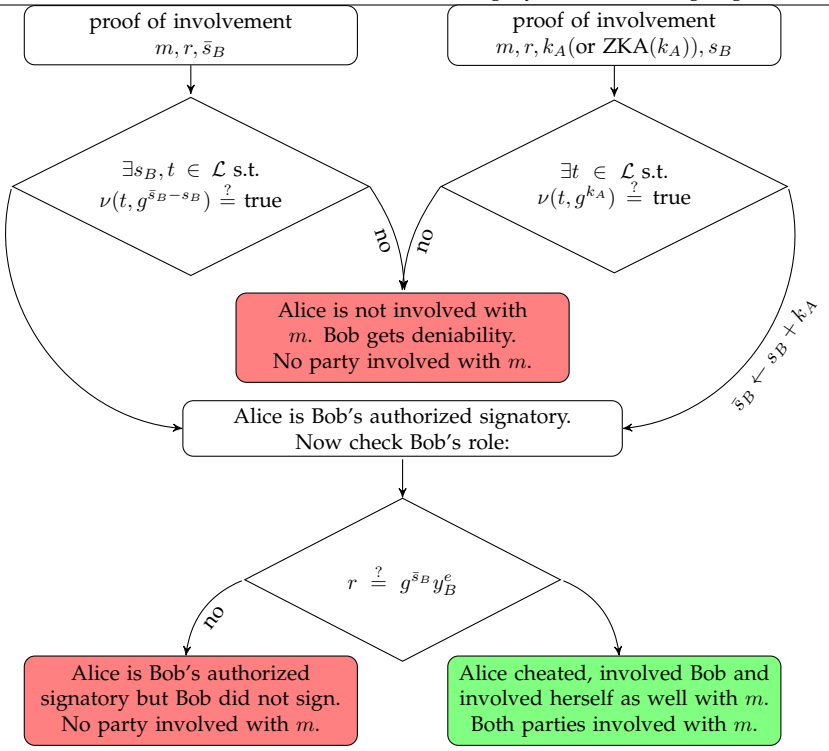


Figure 4.7 – The verification procedure: **proof of involvement**.

The modified signature protocol is described in Figure 4.8. Alice has only one window of opportunity to try and construct a fraudulent signature of Bob: by stopping the protocol at breakpoint ② and using the information s_B [18].

Indeed, if the protocol is interrupted before breakpoint ①, then no information involving m was released by any of the parties: The protocol's trace can be simulated without Bob's help as follows

$$\begin{aligned} s_B, r &\stackrel{\$}{\leftarrow} \mathbb{Z}_q \\ e &\leftarrow H(1 \| m \| r \| \text{Alice} \| \text{Bob}) \\ r_B &\leftarrow g^{s_B} y_B^e \\ r_A &\leftarrow r \times r_B^{-1} \\ t &\leftarrow \sigma(r_A \| \text{Alice} \| \text{Bob}) \\ \rho &\leftarrow H(0 \| r_B) \end{aligned}$$

and Bob has only received from Alice the signature of a random integer.

If Alice and Bob successfully passed the normal completion breakpoint ③, *both* parties have the co-signature, and are provably committed to m.

4.2.3.5 Willingness to Sign Attacks

David Pointcheval [Poi] pointed out a subtle attack that exceeds our model. In the scenario considered by Pointcheval Bob is willing to sign m. An attacker wanting to check this fact eavesdrops a legitimate co-signature session with Alice and replays r_A, t. Bob will proceed and pass breakpoint ① thereby revealing to the attacker his intent to co-sign m with Alice. A possible way to avoid this attack consists in

17. In particular, the question of whether Bob "intended" to sign is outside the scope of this security model.
18. If Bob transmits a wrong or incorrect s_B, this will be immediately detected by Alice as $r_B \neq g^{s_B} y_B^e$. Naturally, in such a case, Bob never sent any information binding him to the contract anyway.

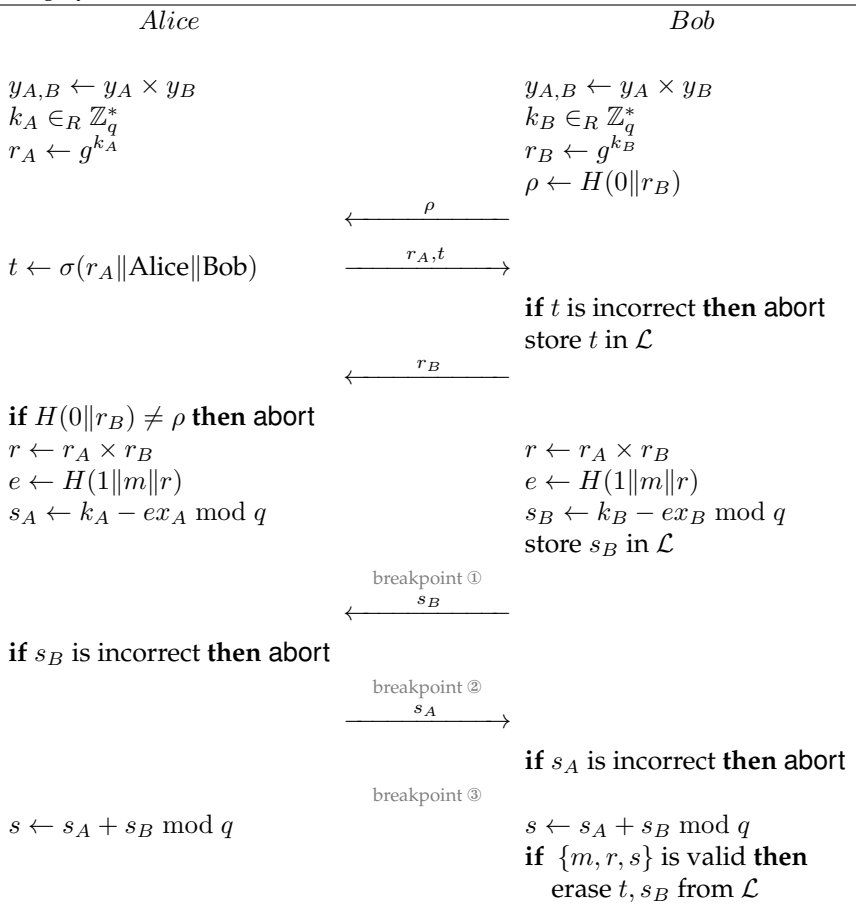


Figure 4.8 – The legally fair co-signature of message m.

having Bob send an auxiliary random challenge z along with ρ. The definition of t will then be modified to include z (*i.e.* $t = \sigma(r_A|z|\text{Alice}|\text{Bob})$). This will prevent the recycling (replay) of past protocol sessions. We conjecture that this countermeasure suffices to thwart these willingness to sign attacks although we did not prove that.

4.2.3.6 Conclusion and Further Work

In this section we described an alternative construction paradigm for legally fair contract signing that doesn't require keystones, but can be combined with them to provide additional power. The new paradigm produces co-signatures that bind a pair of users, and can be adapted to a number of DLP signature protocols. In the co-signature version of Schnorr's protocol, the resulting co-signatures have the same format as classical (single-user) signature. This paradigm guarantees fairness and abuse-freeness, and can be equipped with keystones to add functionalities such as whistle-blower traceability.

CHAPTER 5

DESIGNING AUTHENTICATION PROTOCOLS

Summary

This chapter presents our research results in the area of *authentication*.

Discrete-logarithm authentication protocols are known to present two interesting features: The first is that the prover's commitment, $x = g^r$, claims most of the prover's computational effort. The second is that x does not depend on the challenge and can hence be computed in advance. Provers exploit this feature by pre-loading (or pre-computing) ready to use commitment pairs r_i, x_i. The r_i can be derived from a common seed but storing each x_i still requires 160 to 256 bits when implementing DSA or Schnorr.

Section 5.1 [1] proposes a new concept called *slow motion zero-knowledge* (SM-ZK). SM-ZK allows the prover to slash commitment size (by a factor of 4 to 6) by combining classical zero-knowledge and a timing channel. We pay the conceptual price of requiring the ability to measure time but, in exchange, obtain communication-efficient protocols.

Section 5.2 [2] introduces "thrifty" zero-knowledge protocols, or TZK. These protocols are constructed by introducing a bias in the challenge send by the prover. This bias is chosen so as to maximize the security versus effort trade-off. We illustrate the benefits of this approach on several well-known zero-knowledge protocols.

Section 5.3 [3] presents a lightweight algorithm allowing a verifier to collectively identify a community of provers. This protocol is more efficient than one-to-one node authentication, resulting in less communication, less computation, and hence a smaller overall energy consumption. The protocol is provably secure, and achieves zero-knowledge authentication of a time linear in the degree of the spanning tree.

The proposed authentication protocol may be adapted to better fit constraints: in the context of Internet of Things (IoT), communication is a very costly operation. We describe versions that reduce the amount of data sent by individual nodes, while maintaining security.

Section 5.4 [4] describes the forensic analysis of what the authors believe to be the most sophisticated smart card fraud encountered to date. In a way, this section illustrates what can happen when authentication protocols are wrongly designed. In 2010, Murdoch *et al.* [MDAB10] described a man-in-the-middle attack against EMV cards. [MDAB10] demonstrated the attack using a general purpose FPGA board, noting that "*miniaturization is mostly a mechanical challenge, and well within the expertise of criminal gangs*". This indeed happened in 2011, when about 40 sophisticated card forgeries surfaced in the field. These forgeries are remarkable in that they embed two chips wired top-to-tail. The first chip is clipped from a

1. Co-authored with Rémi Géraud and David Naccache.
2. Co-authored with Simon Cogliani, Rémi Géraud and David Naccache.
3. Co-authored with Simon Cogliani, Rémi Géraud, Diana Maimuţ, David Naccache and Rodrigo Portella do Canto.
4. Co-authored with Rémi Géraud, David Naccache and Assia Tria.

genuine stolen card. The second chip plays the role of the man-in-the-middle and communicates directly with the point of sale (PoS) terminal. The entire assembly is embedded in the plastic body of yet another stolen card. The forensic analysis relied on X-ray chip imaging, side-channel analysis, protocol analysis, and microscopic optical inspections.

5.1 Slow Motion Zero Knowledge – Identifying With Colliding Commitments

5.1.1 Introduction

Authentication is a cornerstone of information security, and much effort has been put in trying to design efficient authentication primitives. However, even the most succinct authentication protocols require collision-resistant commitments. As proved by Girault and Stern [GS94a], breaking beyond the collision-resistance size barrier is impossible. The research work presented in this section shows that if we add the assumption that the verifier can measure the prover's response time, then commitment collision-resistance becomes unnecessary.

We call this new construction *slow-motion zero knowledge* (SM-ZK). As we will show, the parameter determining commitment size in SM-ZK protocols is the attacker's online computational power rather than the attacker's overall computational power.

As a result, SM-ZK allows a significant reduction (typically by a factor of 4 to 6) of the prover's commitment size. The prover's on-line computational effort remains unchanged (enabling instant replies in schemes such as GPS [GPS06]).

The prover's offline work is only slightly increased. The main price is paid by the verifier who has to solve a time-puzzle per session. The time taken to solve this time-puzzle determines the commitment's shortness.

Note that the use of pre-computations is not new: the nearly-instant on-line performance of Schnorr-like authentication protocols is known and largely commented upon [Roo97; MN94]. In the same vein, [NM95; Gir00] were the first to explore and formalize the use of time measurement during coupon-based authentication as a security improvement means. [Gir00] is, in itself, a continuation of [GS94a].

The major contribution of this work is thus a technique forcing a cheating prover to either attack the underlying zero-knowledge protocol or exhaust the space of possible replies in the presence of a time-lock function that slows down his operations. When this time-lock function is properly tuned, a simple time-out on the verifier's side rules out cheating provers. It is interesting to contrast this approach to the notion of *knowledge tightness* introduced by Goldreich, Micali and Widgerson [GMW91], and generalizations such as *precise/local ZK* introduced by Micali and Pass [MP06], which uses similar time-constraint arguments but to prove reduced knowledge leakage bounds.

5.1.2 Building Blocks

SM-ZK combines two existing building blocks that we now recall: three-pass zero-knowledge protocols and time-lock functions.

5.1.2.1 Three-Pass Zero-Knowledge Protocols

A Σ-protocol [HL10; Dam10; GMR85] is a generic 3-step interactive protocol, whereby a prover $\mathcal{P}$ communicates with a verifier $\mathcal{V}$. The goal of this interaction is for $\mathcal{P}$ to convince $\mathcal{V}$ that $\mathcal{P}$ knows some value – without revealing anything beyond this assertion. The absence of information leakage is formalized by the existence of a simulator $\mathcal{S}$, whose output is indistinguishable from the recording (trace) of the interaction between $\mathcal{P}$ and $\mathcal{V}$.

The three phases of a Σ protocol can be summarized by the following exchanges:

$$\mathcal{P} \xrightarrow{x} \xleftarrow{c} \xrightarrow{y} \mathcal{V}$$

Namely,

— The prover sends a *commitment* x to the verifier;
— The verifier replies with a *challenge* c;
— The prover gives a *response* y.

Upon completion, $\mathcal{V}$ may accept or reject $\mathcal{P}$, depending on whether $\mathcal{P}$'s answer is satisfactory. Such a description encompasses well-known identification protocols such as Feige-Fiat-Shamir [FFS88] and Girault-Poupard-Stern [Gir90].

Formally, let R be some (polynomial-time) recognizable relation, then the set $L = \{v \text{ s.t. } \exists w, (v, w) \in R\}$ defines a *language*. Proving that $v \in L$ therefore amounts to proving knowledge of a witness w such that $(v, w) \in R$. A Σ-protocol satisfies the following three properties:

— *Completeness*: given an input v and a witness w such that $(v, w) \in R$, $\mathcal{P}$ is always able to convince $\mathcal{V}$.
— *Special honest-verifier zero-knowledge*[5]: there exists a probabilistic polynomial-time simulator $\mathcal{S}$ which, given v and a c, outputs triples (x, c, y) that have the same distribution as in a valid conversation between $\mathcal{P}$ and $\mathcal{V}$.
— *Special soundness*: given two accepting conversations for the same input v, with different challenges but an identical commitment x, there exists a probabilistic polynomial-time extractor procedure $\mathcal{E}$ that computes a witness w such that $(v, w) \in R$.

Many generalizations of zero-knowledge protocols have been discussed in the literature. One critical question for instance is to compose such protocols in parallel [GMW91; MP06], or to use weaker indistiguishability notions (e.g., computational indistinguishability).

5.1.3 Commitment Pre-Processing

Because the commitment x does not depend on the challenge c, authors quickly noted that x can be prepared in advance. This is of little use in protocols where the creation of x is easy (e.g., Fiat-Shamir [FFS88]). Discrete-logarithm commitment pre-processing is a well-known optimization technique (e.g., [Roo97; MN94]) that exploits two properties of DLP:

1. In DLP-based protocols, a commitment is generated by computing the exponentiation $x = g^r$ in a well-chosen group. This operation claims most of the prover's efforts.
2. The commitment x being unrelated to the challenge c, can hence be computed in advance. A "pre-computed commitment" is hence defined as $\{r, x\}$ computed in advance by $\mathcal{P}$[6]. Because several pre-computed commitments usually need to be saved by $\mathcal{P}$ for later use, it is possible to derive all the r_i components by hashing a common seed.

Such pre-processing is interesting as it enables very fast interaction between prover and verifier. While the technique described in this work does not require the use of pre-processing, it is entirely compatible with such optimizations.

5.1.4 Time-Lock Puzzles

Time-lock puzzles [RSW96; MMV11] are problems designed to guarantee that they will take (approximately) τ units of time to solve. Like proof-of-work protocols [DN92], time-locks have found applications in settings where delaying requests is desirable, such as fighting spam or denial-of-service attacks, as well as in electronic cash [ABMW05; DGN03; DNW05].

Time-lock puzzles may be based on computationally demanding problems, but not all such problems make good time-locks. For instance, inverting a weak one-way function would in general not provide a good time-lock candidate [RSW96]. The intuition is that the time it takes to solve a time-lock should not be significantly reduced by using more computers (i.e., parallel brute-force) or more expensive machines.

5. Note that *special* honest-verifier zero-knowledge implies honest-verifier zero-knowledge.
6. Or for $\mathcal{P}$ by a trusted authority.

A time-lock puzzle is informally described as a problem such that there is a super-polynomial gap between the work required to generate the puzzle, and the parallel time required to solve it (for a polynomial number of parallel processors). The following definition formalizes this idea [Cio12].

Definition 5.1 (Time-lock puzzle) *A time-lock puzzle is the data two PPT algorithms $\mathcal{T}_G(1^k, t)$ (problem generator) and $\mathcal{T}_V(1^k, a, v)$ (solution verifier) satisfying the following properties:*

— *For every PPT algorithm $B(1^k, q, h)$, for all $e \in \mathbb{N}$, there exists $m \in \mathbb{N}$ such that*

$$\sup_{t \geq k^m, |h| \leq k^e} \Pr\left[(q, a) \leftarrow \mathcal{T}_G(1^k, t) \text{ s.t. } \mathcal{T}_V(1^k, a, B(1^k, q, h)) = 1\right]$$

is $\mathsf{negl}(k)$. Intuitively, $\mathcal{T}_G$ generates puzzles of hardness t, and B cannot efficiently solve any puzzle of hardness $t \geq k^m$ for some constant m depending on B.

— *There is some $m \in \mathcal{N}$ such that, for every $d \in \mathcal{N}$, there is a PPT algorithm $C(1^k, t)$ such that*

$$\min_{t \leq k^d} \Pr\left[(q, a) \leftarrow \mathcal{T}_G(1^k, t), v \leftarrow C(1^k, q) \text{ s.t. } \mathcal{T}_V(1^k, a, v) = 1 \text{ and } |v| \leq k^m\right]$$

is overwhelming in k. Intuitively, this second requirement ensures that for any polynomial hardness value, there exists an algorithm that can solve any puzzle of that hardness.

Rivest, Shamir and Wagner [RSW96], and independently Boneh and Naor [BN00] proposed a time-lock puzzle construction relying on the assumption that factorization is hard. This is the construction we retain for this work, and to the best of our knowledge the only known one to achieve interesting security levels. The original Rivest-Shamir-Wagner (RSW) time-lock [RSW96] is based on the "intrinsically sequential" problem of computing:

$$2^{2^\tau} \bmod n$$

for specified values of τ and an RSA modulus n. The parameter τ controls the puzzle's difficulty. The puzzle can be solved by performing τ successive squares modulo n.

Using the formalism above, the RSW puzzle can be described as follows:

$$\mathcal{T}_G(1^k, t) = \left((p_1 p_2, \min(t, 2^k)), (p_1, p_2, \min(t, 2^k))\right)$$

$$\mathcal{T}_V(1^k, (p_1, p_2, t'), v) = \begin{cases} 1 & \text{if } (v = v_1, v_2) \text{ and } v_1 = 2^{2^{t'}} \bmod n \text{ and } v_2 = n \\ 0 & \text{otherwise} \end{cases}$$

where p_1 and p_2 are $(k/2)$-bit prime numbers. Both solving the puzzle and verifying the solution can be efficiently done if p_1 and p_2 are known.

Good time-lock problems seem to be hard to find, and in particular there exist impossibility results against unbounded adversaries [MMV11]. Nevertheless, the RSW construction holds under a computational assumption, namely that factorization of RSA moduli is hard

5.1.5 Slow Motion Zero-Knowledge Protocols

5.1.5.1 Definition

We can now introduce the following notion:

Definition 5.2 (SM-ZK) *A* Slow Motion Zero-Knowledge *(SM-ZK) protocol $(\sigma, \mathcal{T}, \tau, \Delta_{max})$, where σ defines a Σ protocol, $\mathcal{T}$ is a time-lock puzzle, $\tau \in \mathbb{N}$, and $\Delta_{max} \in \mathbb{R}$, is defined by the three following steps of σ:*

1. *Commitment: $\mathcal{P}$ sends a commitment x to $\mathcal{V}$*
2. *Timed challenge: $\mathcal{V}$ sends a challenge c to $\mathcal{P}$, and starts a timer.*
3. *Response: $\mathcal{P}$ provides a response y to $\mathcal{V}$, which stops the timer.*

$\mathcal{V}$ accepts iif

— *y is accepted as a satisfactory response by σ; and*

— *x is a solution to the time-lock puzzle $\mathcal{T}$ with input (y, c) and hardness τ; and*

— *time elapsed between challenge and response, as measured by the timer, is smaller than* Δ_{max}.

Such a protocol must satisfy a new notion of soundness:

Definition 5.3 (Time-constrained soundness) *Let* $\tilde{\Sigma} = (\sigma, \mathcal{T}, \tau, \Delta)$ *be a slow-motion zero-knowledge protocol. Given two accepting conversations for the same input Let* $\epsilon' > 0$, *then there exists* $\tau > 0$ *such that, if a prover* $\mathcal{A}$ *is accepted with non-negligible probability* $\epsilon = 1/B^N + \epsilon'$ *by honest verifiers, then with overwhelming probability,* $\mathcal{A}$ *knows* s.

5.1.5.2 Commitment shortening

Commitments in a Σ-protocol are under the control of $\mathcal{P}$, which may be malicious. If commitments are not collision-resistant, the protocol's security is weakened. Hence commitments need to be long, and in classical Σ protocols breaking below the collision-resistance size barrier is impossible as proved by [GS94b].

However, as we now show, commitment collision-resistance becomes unnecessary in the case of SM-ZK protocols.

5.1.6 An Example Slow Motion Zero Knowledge

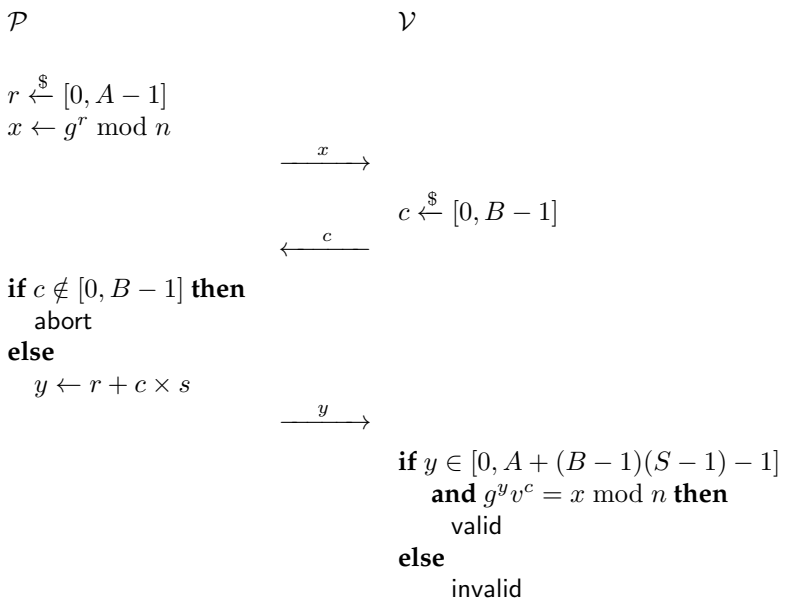


Figure 5.1 – Girault-Poupard-Stern identification protocol.

While SM-ZK can be instantiated with any three-pass ZK protocol, we will illustrate the construction using the Girault-Poupard-Stern (GPS) protocol [PS98; GPS06; Gir90], and a modification of the time-lock construction due to Rivest, Shamir and Wagner [RSW96].

5.1.6.1 Girault-Poupard-Stern Protocol

GPS key generation consists in generating a composite modulus n, choosing a public generator $g \in [0, n-1]$ and integers A, B, S such that $A \gg BS$. Choice of parameters depends on the application and is discussed in [GPS06]. Implicitly, parameters A, B, S are functions of the security parameter k.

The secret key is an integer $s \in [0, S-1]$, and the corresponding public key is $v = g^{-s} \bmod n$. Authentication is performed as in Figure 5.1.

$\mathcal{P}$ can also pre-compute as many values $x_i \leftarrow g^{r_i}$ as suitable for the application, storing a copy of r_i for later usage. The detailed procedure by which this is done is the following:

Figure 5.2 described one possible way in which pre-computed commitments are generated and used for GPS. In this figure, we delegate the computation to a trusted authority. That role can be played by $\mathcal{P}$ alone, but we leverage the authority to alleviate $\mathcal{P}$'s computational burden.

To efficiently generate a sequence of commitments, the authority uses a shared secret seed J and a cryptographic hash function H. Here J is chosen by $\mathcal{P}$ but it could be chosen by the authority instead.

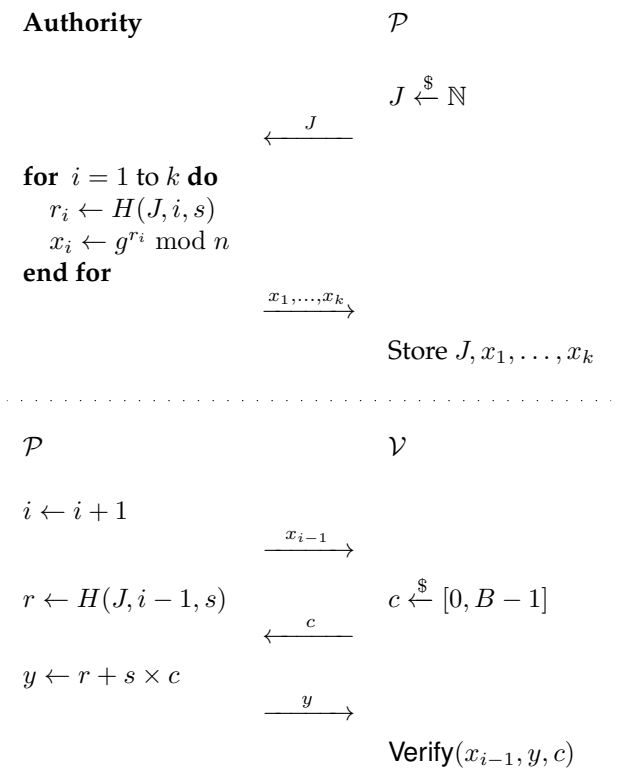


Figure 5.2 – Commitment pre-processing as applied to GPS. The first stage describes the preliminary interaction with a trusted authority, where pre-computed commitments are generated and stored. The second stage describes the interaction with a verifier. For the sake of clarity the range-tests on c and y were omitted. The trusted authority can be easily replaced by $\mathcal{P}$ himself.

5.1.6.2 GPS-RSW SM-ZK

We can now combine the previous building-blocks to construct a pre-processing scheme that requires little commitment storage.

The starting point is a slightly modified version of the RSW time-lock function $\tau \mapsto 2^{2^\tau}$. Let μ be some deterministic function (to be defined later) and $\overline{n}$ an RSA modulus different from the n used for the GPS, we define for integers τ, ℓ:

$$f_{\tau,\ell}(x) = \left(\mu(x)^{2^\tau} \bmod \overline{n}\right) \bmod 2^\ell.$$

Here, τ controls the puzzle hardness and ℓ is a parameter controlling output size.

The function $f_{\tau,\ell}$ only differs from the RSW time-lock in two respects: We use $\mu(x)$ instead of 2; and the result is reduced modulo 2^ℓ.

The motivation behind using a function μ stems from the following observation: An adversary knowing $x_1^{2^\tau}$ and $x_2^{2^\tau}$ could multiply them to get $(x_1 x_2)^{2^\tau}$. To thwart such attacks (and similar attacks based on the malleability of RSA) we suggest to use for μ a deterministic RSA signature padding function (e.g., the Full Domain Hash [BR94]).

The reduction modulo 2^ℓ is of practical interest, it is meant to keep the size of answers manageable. Of course, an adversary could brute-force all values between 0 and $2^\ell - 1$ instead of trying to solve the time-lock. To avoid this situation, ℓ and τ should be chosen so that solving the time-lock is the most viable option of the two.

Under the same assumptions as RSW (hardness of factorization), and if ℓ and τ are properly tuned, $f_{\tau,\ell}$ generates a time-lock problem.

Then, we adapt a construction of M'Raïhi and Naccache [MN94] to GPS[Gir90]. This is done by defining a secret J, a public hash function H, and computing the quantities:

$$x'_i = g^{H(J,i,s)} \bmod n$$

This computation can be delegated to a trusted authority. This is interesting in our case because the authority can compress these x'_i by computing $x_i = f_{\tau,\ell}(x'_i)$. Note that because the authority knows the factors of $\overline{n}$, computing the x_i is fast. $\mathcal{P}$ is loaded with k pre-computed commitments $x_1, \ldots, x_k$ as shown in Figure 5.3. The quantity k of pre-computed commitments depends on the precise application.

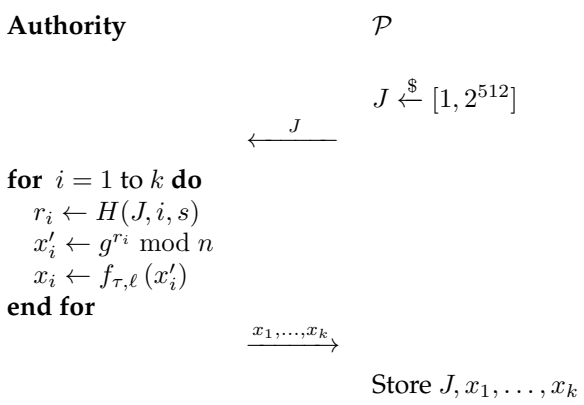


Figure 5.3 – Slow motion commitment pre-processing for GPS.

When $\mathcal{V}$ wishes to authenticate $\mathcal{P}$ the parties execute the protocol shown in Figure 5.4.

With a proper choice of τ, ℓ we can have a reasonable verification time (assuming that $\mathcal{V}$ is more powerful than $\mathcal{P}$), extremely short commitments (e.g., 40-bit ones) and very little on-line computations required from $\mathcal{P}$.

5.1.6.3 Choice of Parameters

What drives the choice of parameters is the ratio between:

— The time t it takes to a legitimate prover to compute y and transmits it. In GPS this is simply one multiplication of operands of sizes $\log_2 B$ and $\log_2 S$ (additions neglected), this takes time $\lambda \log(B) \log(S)$ for some constant λ (not assuming optimizations such as [Ber86] based on the fact that operand s is constant).

— The time T it takes for the fastest adversary to evaluate once the time-lock function $f_{\tau,\ell}$. T does not really depend on ℓ, and is linear in τ. We hence let $T = \nu\tau$. Note that there is no need to take into account the size of $\overline{n}$, all we require from $\overline{n}$ is to be hard to factor. That way, the slowing effect will solely depend on τ.

In a brute-force attack, there are 2^ℓ possibilities to exhaust. The most powerful adversary may run $\kappa \leq 2^\ell$ parallel evaluations of the time-lock function, and succeed to solve the puzzle in t time units with

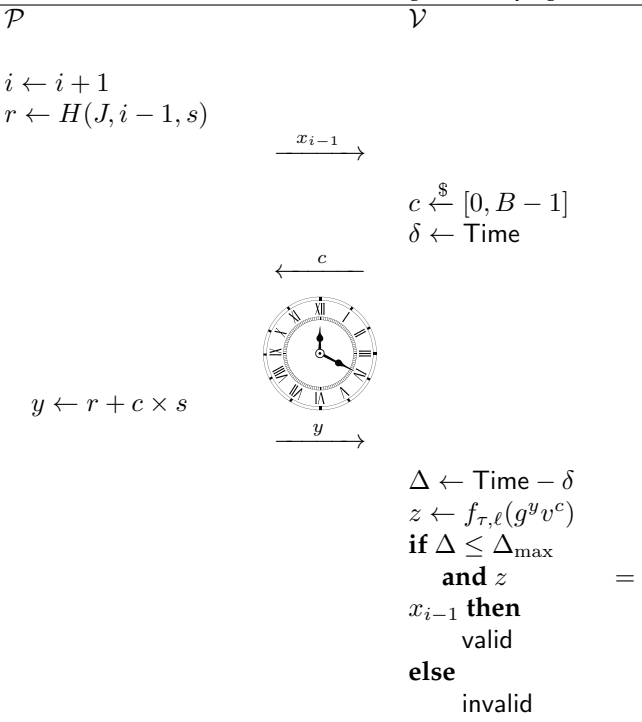


Figure 5.4 – Slow Motion GPS. Range tests on c and y omitted for the sake of clarity.

probability

$$\epsilon = \frac{\kappa t}{2^\ell T} = \frac{\kappa \log(B) \log(S) \lambda}{\nu 2^\ell \tau}$$

A typical instance resulting in 40-bit commitments is $\{\kappa = 2^{24}, T = 1, t = 2^{-4}, \epsilon = 2^{-20}\} \Rightarrow \ell = 40$. Here we assume that the attacker has 16.7 million (2^{24}) computers capable of solving one time-lock challenge per second ($T = 1$) posing as a prover responding in one sixteenth of a second ($t = 2^{-4}$). Assuming the least secure DSA parameters (160-bit q) this divides commitment size by 4. For 256-bit DSA the gain ratio becomes 6.4.

The time-out constant $\Delta_{\max}$ in Figure 5.4 is tuned to be as small as possible, but not so short that it prevents legitimate provers from authenticating. Therefore the only constraint is that $\Delta_{\max}$ is greater or equal to the time t it takes to the *slowest legitimate prover* to respond. Henceforth we assume $\Delta_{\max} = t$.

5.1.7 Security Proof

The security of this protocol is related to that of the standard GPS protocol analysed in [PS98; GPS06]. We recall here the main results and hypotheses.

5.1.7.1 Preliminaries

The following scenario is considered. A randomized polynomial-time algorithm Setup generates the public parameters $(\mathcal{G}, g, S)$ on input the security parameter k. Then a second probabilistic algorithm GenKey generates pairs of public and private keys, sends the secret key to $\mathcal{P}$ while the related public key is made available to anybody, including of course $\mathcal{P}$ and $\mathcal{V}$. Finally, the identification procedure is a protocol between $\mathcal{P}$ and $\mathcal{V}$, at the end of which $\mathcal{V}$ accepts or not.

An adversary who doesn't corrupt public parameters and key generation has only two ways to obtain information: either passively, by eavesdropping on a regular communication, or actively, by impersonating (in a possibly non protocol-compliant way) $\mathcal{P}$ and $\mathcal{V}$.

The standard GPS protocol is proven complete, sound and zero-knowledge by reduction to the *discrete logarithm with short exponent problem* [GPS06]:

Definition 5.4 (Discrete logarithm with short exponent problem) *Given a group $\mathcal{G}$, $g \in \mathcal{G}$, and integer S and a group element g^x such that $x \in [0, S-1]$, find x.*

5.1.7.2 Compressed Commitments For Time-Locked GPS

We now consider the impact of shortening the commitments to ℓ bits on security, while taking into account the time constraint under which $\mathcal{P}$ operates. The shortening of commitments will indeed weaken the protocol [GS94b] but this is compensated by the time constraint, as explained below.

Lemma 5.1 (Completeness) *Execution of the protocol of Figure 5.4 between a prover $\mathcal{P}$ who knows the secret key corresponding to his public key, and replies in bounded time $\Delta_{\max}$, and a verifier $\mathcal{V}$ is always successful.*

Proof: This is a direct consequence of the completeness of the standard GPS protocol [GPS06, Theorem 1]. By assumption, $\mathcal{P}$ computes y and sends it within the time allotted for the operation. This computation is easy knowing the secret s and we have

$$g^y v^c = g^{r_i + cs} v^c = x'_i g^{cs} v^c = x'_i v^{c-c} = x'_i$$

Consequently, $f_{\tau,\ell}(g^y v^c) = f_{\tau,\ell}(x'_i) = x_i$. Finally,

$$y = r + cs \leq (A-1) + (B-1)(S-1) < y_{\max}.$$

Therefore all conditions are met and the identification succeeds. □

Lemma 5.2 (Zero-Knowledge) *The protocol of Figure 5.4 is statistically zero-knowledge if it is run a polynomial number of times N, B is polynomial, and NSB/A is negligible.*

The proof follows [GPS06] and is the following:

Proof: The zero-knowledge property of the standard GPS protocol is proven by constructing a polynomial-time simulation of the communication between a prover and a verifier [GPS06, Theorem 2]. We adapt this proof to the context of the proposed protocol. The function δ is defined by $\delta(\mathsf{true}) = 1$ and $\delta(\mathsf{false}) = 0$, and $\wedge$ denotes the logical operator "and". For clarity, the function $f_{\tau,\ell}$ is henceforth written f.

The scenario is that of a prover $\mathcal{P}$ and a dishonest verifier $\mathcal{A}$ who can use an adaptive strategy to bias the choice of the challenges to try to obtain information about s. In this case the challenges are no longer chosen at random, and this must be taken into account in the security proof. Assume the protocol is run N times and focus on the i-th round.

$\mathcal{A}$ has already obtained a certain amount of information η from past interactions with $\mathcal{P}$. $\mathcal{P}$ sends a pre-computed commitment x_i. Then $\mathcal{A}$ chooses a commitment using all information available to her, and a random tape ω: $c_i\,(x_i, \eta, \omega)$.

The following is an algorithm (using its own random tape ω_M) that simulates this round:

Step 1. Choose $\overline{c_i} \xleftarrow{\$} [0, B-1]$ and $\overline{y_i} \xleftarrow{\$} [(B-1)(S-1), A-1]$ using ω_M.

Step 2. Compute $\overline{x_i} = f_{\tau,\ell}\left(g^{\overline{y_i}} v^{\overline{c_i}}\right)$.

Step 3. If $c_i\,(\overline{x_i}, \eta, \omega) = \overline{c_i}$ then return to step 1 and try again with another pair $(\overline{c_i}, \overline{y_i})$, else return $(\overline{x_i}, \overline{c_i}, \overline{y_i})$.[7]

7. The probability of success at step 3 is essentially $1/B$, and the expected number of executions of the loop is B, so that the simulation of N rounds runs in $O(NB)$: the machine runs in expected polynomial time.

The rest of the proof shows that, provided $\Phi = (B-1)(S-1)$ is much smaller than A, this simulation algorithm outputs triples that are indistinguishable from real ones, for any fixed random tape ω.

Formally, we want to prove that

$$\Sigma_1 = \sum_{\alpha,\beta,\gamma} \left| \Pr_{\omega_P}[(x,c,y) = (\alpha,\beta,\gamma)] - \Pr_{\omega_M}[(\overline{x},\overline{c},\overline{y}) = (\alpha,\beta,\gamma)] \right|$$

is negligible, i.e., that the two distributions cannot be distinguished by accessing a polynomial number of triples (even using an infinite computational power). Let (α,β,γ) be a fixed triple, and assuming a honest prover, we have the following probability:

$$\begin{aligned}
p &= \Pr_{\omega_P}[(x,c,y) = (\alpha,\beta,\gamma)] \\
&= \Pr_{0\le r<A}[\alpha = f(g^r) \wedge \beta = c(\alpha,\eta,\omega) \wedge \gamma = r + \beta s] \\
&= \sum_{r=0}^{A-1} \frac{1}{A} \delta\left(\alpha = f(g^\gamma v^\beta) \wedge \beta = c(\alpha,\eta,\omega) \wedge r = \gamma - \beta s\right) \\
&= \frac{1}{A} \delta\left(\alpha = f(g^\gamma v^\beta) \wedge \beta = c(\alpha,\eta,\omega) \wedge \gamma - \beta s \in [0, A-1]\right) \\
&= \frac{1}{A} \delta\left(\alpha = f(g^\gamma v^\beta)\right) \delta\left(\beta = c(\alpha,\eta,\omega)\right) \delta\left(\gamma - \beta s \in [0, A-1]\right).
\end{aligned}$$

where $f = f_{\tau,\ell}$.

We now consider the probability $\overline{p} = \Pr_{\omega_M}[(\overline{x},\overline{c},\overline{y}) = (\alpha,\beta,\gamma)]$ to obtain the triple (α,β,γ) during the simulation described above. This is a conditional probability given by

$$\overline{p} = \Pr_{\substack{\overline{y}\in[\Phi,A-1] \\ \overline{c}\in[0,B-1]}} \left[\alpha = f\left(g^{\overline{y}} v^{\overline{c}}\right) \wedge \beta = \overline{c} \wedge \gamma = \overline{y} \;\middle|\; \overline{c} = c\left(f\left(g^{\overline{y}} v^{\overline{c}}\right), \eta, \omega\right)\right]$$

Using the definition of conditional probabilities, this equals

$$\overline{p} = \frac{\Pr_{\substack{\overline{y}\in[\Phi,A-1] \\ \overline{c}\in[0,B-1]}} \left[\alpha = f\left(g^{\overline{y}} v^{\overline{c}}\right) \wedge \beta = \overline{c} \wedge \gamma = \overline{y}\right]}{\Pr_{\substack{\overline{y}\in[\Phi,A-1] \\ \overline{c}\in[0,B-1]}} \left[\overline{c} = c\left(f\left(g^{\overline{y}} v^{\overline{c}}\right), \eta, \omega\right)\right]}$$

Let us introduce

$$Q = \sum_{\substack{\overline{y}\in[\Phi,A-1] \\ \overline{c}\in[0,B-1]}} \delta\left(\overline{c} = c\left(f\left(g^{\overline{y}} v^{\overline{c}}\right), \eta, \omega\right)\right)$$

then the denominator in $\overline{p}$ is simply $Q/B(A-\Phi)$. Therefore:

$$\begin{aligned}
\overline{p} &= \sum_{\overline{c}\in[0,B-1]} \frac{1}{B} \Pr_{\overline{y}\in[\Phi,A-1]} \left[\alpha = f\left(g^{\overline{y}} v^{\overline{c}}\right) \wedge \gamma = \overline{y} \wedge \beta = \overline{c} = c(\alpha,\eta,\omega)\right] \frac{B(A-\Phi)}{Q} \\
&= \Pr_{\overline{y}\in[\Phi,A-1]} \left[\alpha = f\left(g^\gamma v^\beta\right) \wedge \gamma = \overline{y} \wedge \beta = c(\alpha,\eta,\omega)\right] \frac{A-\Phi}{Q} \\
&= \sum_{\overline{y}\in[\Phi,A-1]} \frac{1}{A-\Phi} \delta\left(\alpha = f\left(g^\gamma v^\beta\right) \wedge \gamma = \overline{y} \wedge \beta = c(\alpha,\eta,\omega)\right) \frac{A-\Phi}{Q} \\
&= \frac{1}{Q} \delta\left(\alpha = f\left(g^\gamma v^\beta\right)\right) \delta\left(\beta = c(\alpha,\eta,\omega)\right) \delta\left(\gamma \in [\Phi, A-1]\right)
\end{aligned}$$

We will now use the following combinatorial lemma:

Lemma 5.3 *If $h : \mathcal{G} \to [0, B-1]$ and $v \in \{g^{-s}, s \in [0, S-1]\}$ then the total number M of solutions $(c,y) \in [0,B-1] \times [\Phi, A-1]$ to the equation $c = h(g^y v^c)$ satisfies $A - 2\Phi \le M \le A$.*

Proof: [Proof of Lemma 5.3] [GPS06, Appendix A] □

Specializing Lemma 5.3 to the function that computes $c(f(g^{\overline{y}}v^{\overline{c}}), \eta, \omega)$ from $(\overline{c}, \overline{y})$ gives $A - 2\Phi \leq Q \leq A$. This enables us to bound Σ_1:

$$\begin{aligned}
\Sigma_1 &= \sum_{\alpha,\beta,\gamma} \left| \Pr_{\omega_P}[(x,c,y) = (\alpha,\beta,\gamma)] - \Pr_{\omega_M}[(\overline{x},\overline{c},\overline{y}) = (\alpha,\beta,\gamma)] \right| \\
&= \sum_{\alpha,\beta,\gamma \in [\Phi, A-1]} \left| \Pr_{\omega_P}[(x,c,y) = (\alpha,\beta,\gamma)] - \Pr_{\omega_M}[(\overline{x},\overline{c},\overline{y}) = (\alpha,\beta,\gamma)] \right| \\
&\quad + \sum_{\alpha,\beta,\gamma \notin [\Phi, A-1]} \Pr_{\omega_P}[(x,c,y) = (\alpha,\beta,\gamma)] \\
&= \sum_{\substack{\gamma \in [\Phi, A-1] \\ \beta \in [0, B-1] \\ \alpha = f(g^\gamma v^\beta)}} \left| \frac{1}{A}\delta(\beta = c(\alpha,\eta,\omega)) - \frac{1}{Q}\delta(\beta = c(\alpha,\eta,\omega)) \right| \\
&\quad + \left(1 - \sum_{\alpha,\beta,\gamma \in [\Phi, A-1]} \Pr_{\omega_P}[(x,c,y) = (\alpha,\beta,\gamma)] \right) \\
&= \left| \frac{1}{A} - \frac{1}{Q} \right| Q + 1 - \sum_{\substack{\gamma \in [\Phi, A-1] \\ \beta \in [0, B-1] \\ \alpha = f(g^\gamma v^\beta)}} \frac{1}{A}\delta(\beta = c(\alpha,\eta,\omega)) \\
&= \frac{|Q - A|}{A} + 1 - \frac{Q}{A}
\end{aligned}$$

Therefore $\Sigma_1 \leq 2|Q - A|/A \leq 4\Phi/A < 4SB/A$, which proves that the real and simulated distributions are statistically indistinguishable if SB/A is negligible. □

The last important property to prove is that if $\mathcal{V}$ accepts, then with overwhelming probability $\mathcal{P}$ must know the discrete logarithm of v in base g.

Lemma 5.4 (Time-constrained soundness) *Under the assumption that the discrete logarithm with short exponent problem is hard, and the time-lock hardness assumption, this protocol achieves time-constrained soundness.*

Proof: After a commitment x has been sent, if $\mathcal{A}$ can correctly answer with probability $> 1/B$ then he must be able to answer to two different challenges, c and c', with y and y' such that they are both accepted, i.e., $f_{\tau,\ell}(g^y v^c) = x = f_{\tau,\ell}(g^{y'} v^{c'})$. When that happens, we have

$$\mu\left(g^y v^c\right)^{2^\tau} = \mu\left(g^{y'} v^{c'}\right)^{2^\tau} \bmod \overline{n} \bmod 2^\ell$$

Here is the algorithm that extracts these values from the adversary $\mathcal{A}$. We write $\mathrm{Success}(\omega, \mathrm{c}_1, \ldots, \mathrm{c}_\mathrm{n})$ the result of the identification of $\mathcal{A}$ using the challenges $c_1, \ldots, c_n$, for some random tape ω.

Step 1. Pick a random tape ω and a tuple c of N integers $c_1, \ldots, c_N$ in $[0, B-1]$. If $\mathrm{Success}(\omega, \mathrm{c}) = \mathsf{false}$, then abort.

Step 2. Probe random N-tuples c' that are different from each other and from c, until $\mathrm{Success}(\omega, \mathrm{c}') = \mathsf{true}$. If after $B^N - 1$ probes a successful c' has not been found, abort.

Step 3. Let j be the first index such that $c_j \neq c'_j$, write y_j and y'_j the corresponding answers of $\mathcal{A}$. Output c_j, c'_j, y_j, y'_j.

This algorithm succeeds with probability $\geq \epsilon - 1/B^N = \epsilon'$, and takes at most $4\Delta_{\max}$ units of time [GPS06]. This means that there is an algorithm finding collisions in $f_{\tau,\ell}$ with probability $\geq \epsilon'$ and time $\leq 4\Delta_{\max}$.

Assuming the hardness of the discrete logarithm with short exponents problem, the adversary responds in time by solving a hard problem, where as pointed out earlier the probability of success is given by

$$\zeta = \frac{\kappa \log(B) \log(S) \lambda}{\nu 2^{\ell} \tau}$$

where κ is the number of concurrent evaluations of $f_{\tau,\ell}$ performed by $\mathcal{A}$. There is a value of τ such that $\zeta \ll \epsilon$. For this choice of τ, $\mathcal{A}$ is able to compute $f_{\tau,\ell}$ much faster than brute-force, which contradicts the time-lock hardness assumption. □

5.1.8 Conclusion and Further Research

The research work of this section introduced a new class of protocols, called Slow Motion Zero Knowledge (SM-ZK) showing that if we pay the conceptual price of allowing time measurements during a three-pass ZK protocol then commitments do not need to be collision-resistant.

Because of its interactive nature, SM-ZK does not yield signatures but seems to open new research directions. For instance, SM-ZK permits the following interesting construction, that we call a *fading signature*: Alice wishes to send a signed message m to Bob without allowing Bob to keep a long-term her involvement. By deriving $c \leftarrow H(x, m, \rho)$ where ρ is a random challenge chosen by Bob, Bob can can convince himself [8] that m comes from Alice. This conviction is however not transferable if Alice prudently uses a short commitment as described in this section.

8. If y was received before $\Delta_{\max}$.

5.2 Thrifty Zero-Knowledge: When Linear Programming Meets Cryptography

5.2.1 Introduction

Since their discovery, zero-knowledge proofs (ZKPs) [GMR85; BCC88] have found many applications and have become of central interest in cryptology. ZKPs enable a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that some mathematical statement is valid, in such a way that no knowledge but the statement's validity is communicated to $\mathcal{V}$. The absence of information leakage is formalized by the existence of a simulator $\mathcal{S}$, whose output is indistinguishable from the recording (trace) of the interaction between $\mathcal{P}$ and $\mathcal{V}$.

Thanks to this indistinguishability, an eavesdropper $\mathcal{A}$ cannot tell whether she taps a real conversation or the monologue of $\mathcal{S}$. $\mathcal{P}$ and $\mathcal{V}$, however, interact with each other and thus know that the conversation is real.

It may however happen, by sheer luck, that $\mathcal{A}$ succeeds in responding correctly to a challenge without knowing $\mathcal{P}$'s secret. ZKPs are designed so that such a situation is expected to happen only with negligible probability: Repeating the protocol renders the cheating probability exponentially small *if* the challenge at each protocol round is random. Otherwise, $\mathcal{A}$ may repeat her successful commitments while hoping to be served with the same challenges.

Classically, the protocol is regarded as ideal when the challenge distribution is *uniform* over a large set (for efficiency reasons, the cardinality of this set rarely exceeds 2^{128}). Uniformity, however, has its drawbacks: all challenges are not computationally equal, and some challenges may prove harder than others to respond to.

The research work of this section explores the effect of biasing the challenge distribution. Warping this distribution unavoidably sacrifices security, but it appears that the resulting efficiency gains balance this loss in a number of ZKPs. Finding the optimal distribution brings out interesting optimization problems which happen to be solvable exactly for a variety of protocols and variants. We apply this idea to improve on four classical ZK identification protocols that rely on very different assumptions: RSA-based Fiat-Shamir [FFS88], SD-based identification [Ste94], PKP-based identification [Sha90], and PPP-based identification [Poi95].

5.2.2 Preliminaries

5.2.2.1 Three-Round Zero-Knowledge Protocols

A Σ-protocol [HL02; Dam10; GMW91] is a generic 3-step interactive protocol, whereby a prover $\mathcal{P}$ tries to convince a verifier $\mathcal{V}$ that $\mathcal{P}$ knows a proof that some statement is true — without revealing anything to $\mathcal{V}$ beyond this assertion. The three phases of a Σ-protocol are illustrated by Figure 5.5.

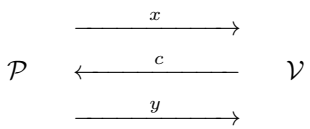


Figure 5.5 – Generic Σ-protocol.

Namely,
- $\mathcal{P}$ sends a *commitment* x to $\mathcal{V}$
- $\mathcal{V}$ replies with a *challenge* c;
- $\mathcal{P}$ provides a *response* y.

Upon completion, $\mathcal{V}$ may accept or reject $\mathcal{P}$, depending on whether $\mathcal{P}$'s response is satisfactory. In practice, the protocol will be repeated several times until $\mathcal{V}$ is satisfied.

An eavesdropper $\mathcal{A}$ should not be able to learn anything from the conversation between $\mathcal{P}$ and $\mathcal{V}$. This security notion is formalized by the existence of a simulator $\mathcal{S}$, whose output is indistinguishable from the interaction (or "trace") T between $\mathcal{P}$ and $\mathcal{V}$. Different types of zero-knowledge protocols exist, that correspond to different indistinguishability notions.

In *computational* zero-knowledge, $\mathcal{S}$'s output distribution is computationally indistinguishable from T, whereas in *statistical* zero-knowledge, $\mathcal{S}$'s output distribution must be statistically close to the distribution governing T: Thus even a computationally unbounded verifier learns nothing from T. The strongest notion of *unconditional* zero-knowledge requires that $\mathcal{A}$ cannot distinguish $\mathcal{S}$'s output from T, even if $\mathcal{A}$ is given access to both unbounded computational resources and $\mathcal{P}$'s private keys. The Fiat-Shamir protocol [FFS88] is an example of unconditional ZKP.

Definition 5.5 (Statistical Indistinguishability) *The statistical difference between random variables X and Y taking values in $\mathcal{Z}$ is defined as:*

$$\begin{aligned}\Delta(X,Y) &:= \max_{Z\subset\mathcal{Z}} |\Pr(X\in Z) - \Pr(Y\in Z)| \\ &= 1 - \sum_{z\in\mathcal{Z}} \min\{\Pr(X=z), \Pr(Y=z)\}\end{aligned}$$

We say that X and Y are statistically indistinguishable *if $\Delta(X,Y)$ is negligible.*

Finally, we expect $\mathcal{P}$ to eventually convince $\mathcal{V}$, and that $\mathcal{V}$ should only be convinced by such a $\mathcal{P}$ (with overwhelming probability). All in all, we have the following definition:

Definition 5.6 (Σ-protocol) *A Σ-protocol is a three-round protocol that furthermore satisfies three properties:*

- Completeness*: given an input v and a witness w such that vRw, $\mathcal{P}$ is always able to convince $\mathcal{V}$.*
- Zero-Knowledge*: there exists a probabilistic polynomial-time simulator $\mathcal{S}$ which, given (v,c), outputs triples (x,c,y) that follow a distribution indistinguishable from a valid conversation between $\mathcal{P}$ and $\mathcal{V}$.*
- Special Soundness*: given two accepting conversations for the same input v, and the same commitment x, but with different challenges $c_1 \neq c_2$, there exists a probabilistic polynomial-time algorithm $\mathcal{E}$ called extractor that computes a witness $w = \mathcal{E}(c_1,c_2,v,x)$ such that vRw.*

5.2.2.2 Security Efficiency

During a Σ-protocol, $\mathcal{P}$ processes c to return the response $y(x,c)$. The amount of computation $W(x,c)$ required for doing so depends on x, c, and on the challenge size, denoted k. Longer challenges — hence higher security levels — would usually claim more computations.

Definition 5.7 (Security Level) *Let $\mathcal{P}\leftrightarrow\mathcal{V}$ be a Σ-protocol, the* security level $S(\mathcal{P}\leftrightarrow\mathcal{V})$*: is defined as the challenge min-entropy*

$$S(\mathcal{P}\leftrightarrow\mathcal{V}) := -\min_c \log\Pr(c)$$

This security definition assumes that $\mathcal{A}$'s most rational attack strategy is to focus her efforts on the most probable challenge. From a defender's perspective, verifiers achieve the highest possible security level by sampling challenges from a uniform distribution.

Definition 5.8 (Work Factor) *Let $\mathcal{P}\leftrightarrow\mathcal{V}$ be a Σ-protocol, the* average work factor $W(\mathcal{P}\leftrightarrow\mathcal{V})$ *is defined as the expected value of $W(x,c)$:*

$$W(\mathcal{P}\leftrightarrow\mathcal{V}) := \mathbb{E}_{x,c}\left[W(x,c)\right]$$

Definition 5.9 (Security Efficiency) *Let $\mathcal{P}\leftrightarrow\mathcal{V}$ be a Σ-protocol, the* security efficiency *of $\mathcal{P}\leftrightarrow\mathcal{V}$, denoted $E(\mathcal{P}\leftrightarrow\mathcal{V})$, is defined as the ratio between $S(\mathcal{P}\leftrightarrow\mathcal{V})$ and $W(\mathcal{P}\leftrightarrow\mathcal{V})$:*

$$E(\mathcal{P}\leftrightarrow\mathcal{V}) := \frac{S(\mathcal{P}\leftrightarrow\mathcal{V})}{W(\mathcal{P}\leftrightarrow\mathcal{V})}$$

Informally, $E(\mathcal{P}\leftrightarrow\mathcal{V})$ represents[9] the average number of security bits per mathematical operation.

9. *i.e.* is proportional to

5.2.2.3 Linear Programming

Linear programming (LP) [Dan51; DT06a; DT06b; BV04] problems appear when a linear objective function must be optimized under linear equality and inequality constraints. These constraints define a convex polytope. General linear programming problems can be expressed in canonical form as:

$$\begin{array}{ll} \text{maximize} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ \text{and} & \mathbf{x} \geq 0 \end{array}$$

where $\mathbf{x}$ represents the vector of variables (to be determined), $\mathbf{c}$ and $\mathbf{b}$ are vectors of (known) coefficients and A is a (known) matrix of coefficients.

Linear programming is common in optimization problems and ubiquitous in logistics, operational research, and economics. Interestingly, linear programming has almost never surfaced in cryptography, save a few occasional appearances in error correcting codes [BGS94], or under the avatar of its NP-hard variant, integer programming [Len84].

Every linear problem can be written in so-called "standard form" where the constraints are all inequalities and all variables are non-negative, by introducing additional variables ("slack variables") if needed. Not all linear programming problems can be solved: The problem might be unbounded (there is no maximum) or unfeasible (no solution satisfies the constraints, *i.e.* the polytope is empty).

Many algorithms are known to solve LP instances, on the forefront Dantzig's Simplex algorithm [Dan51]. The Simplex algorithm solves an LP problem by first finding a solution compatible with the constraints at some polytope vertex, and then walking along a path on the polytope's edges to vertices with non-decreasing values of the objective function. When an optimum is found the algorithm terminates — in practice this algorithm has usually good performance but has poor worst-case behavior: There are LP problems for which the Simplex method takes a number of steps exponential in the problem size to terminate [DT06a; Mur83].

Since the 1950's, more efficient algorithms have been proposed called "interior point" methods (as opposed to the Simplex which evolves along the polytope's vertices). In particular, these algorithms demonstrated the polynomial-time solvability of linear programs [Kar84]. Following this line of research, approximate solutions to LP problems can be found using very efficient (near linear-time) algorithms [KY08; ZO14].

In this work we assume that some (approximate) LP solver is available. Efficiency is not an issue, since this solver is only used once, when the ZKP is designed

5.2.3 Optimizing $E(\mathcal{P} \leftrightarrow \mathcal{V})$

The new idea consists in assigning *different* probabilities to different c values, depending on how much it costs to generate their corresponding y values, while achieving a given security level. The intuition is that by choosing a certain distribution of challenges, we may hope to reduce $\mathcal{P}$'s total amount of effort, but this also reduces security. As we show, finding the best trade-off is equivalent to solving an LP problem.

Consider a set Γ of symbols, and a cost function $\eta : \Gamma \to \mathbb{N}$. Denote by $p_j := \Pr(i \mid i \in \Gamma_j)$ the probability that a symbol i is emitted, given that i has cost j. We wish to find this probability distribution.

Let Γ_j denote all symbols having cost j, *i.e.* such that $\eta(i) = j$. Let γ_j be the cardinality of Γ_j. The expected cost for a given choice of emission probabilities $\{p_j\}$ is

$$W = \mathbb{E}[\eta] = \sum_{i \in \Gamma} \eta(i) \Pr(i) = \sum_j j \times \gamma_j \times p_j$$

W is easy to evaluate provided we can estimate the amount of work associated with each challenge isocost class Γ_j. The condition that probabilities sum to one is expressed as:

$$1 = \sum_{i \in \Gamma} \Pr(i) = \sum_j \gamma_j p_j$$

Finally, security is determined by the min-entropy

$$S = -\log_2 \max_i \Pr(i) = -\log_2 \max_j p_j$$

Let $\epsilon = 2^{-S}$, so that $p_j \leq \epsilon$ for all j. The resulting security efficiency is $E = S/W = (-\log_2 \epsilon)/W$.

We wish to maximize E, which leads to the following constrained optimization problem:

$$\text{Given } \{\gamma_j\} \text{ and } \epsilon, \begin{cases} \text{minimize} & W = \sum_j j p_j \gamma_j \\ \text{subject to} & 0 \leq p_j \leq \epsilon \\ & \sum_j \gamma_j p_j = 1 \end{cases} \tag{5.1}$$

This is a linear programming problem [Dan51; DT06a; DT06b], that can be put in canonical form by introducing slack variables $q_j = \epsilon - p_j$ and turning the inequality constraints into equalities $p_j + q_j = \epsilon$. The solution, if it exists, therefore lies on the boundary of the polytope defined by these constraints.

Note that a necessary condition for an optimal solution to exist is that $\epsilon \geq 1/\sum_j \gamma_j$, which corresponds to the choice of the uniform distribution.

Exact solutions to Equation (5.1) can be found using the techniques mentioned in Section 5.2.2.3.

We call such optimized ZKP versions "thrifty ZKPs". Note that the zero-knowledge property is not impacted, as it is trivial to construct a biased simulator.

5.2.4 Thrifty Zero-Knowledge Protocols

The methodology described in Section 5.2.3 can be applied to any ZK protocol, provided that we can evaluate the work factor associated with each challenge class. As an illustration we analyze thrifty variants of classical ZKPs: Fiat-Shamir (FS, [FFS88]), Syndrome Decoding (SD, [Ste94]), Permuted Kernels Problem (PKP, [Sha90]), and Permuted Perceptrons Problem (PPP, [Poi95]).

5.2.4.1 Thrifty Fiat-Shamir

In the case of Fiat-Shamir [FFS88] (see [FFS88]), response to a challenge c claims a number of multiplications proportional to c's Hamming weight. We have $k = n$-bit long challenges. Here γ_j is the number of n-bit challenges having Hamming weight j, namely

$$\gamma_j = \binom{n}{j}$$

Note that the lowest value of ϵ for which a solution to Equation (5.1) exists is 2^{-n}, in which case $p_j = \epsilon$ is the uniform distribution, and $W = n/2$. Hence the original Fiat-Shamir always has $E = 2$.

Example 5.1 *Let $n = 3$. In that case Equation (5.1) becomes the following problem:*

$$\textit{Given } \epsilon, \begin{cases} \textit{minimize} & W = 3p_1 + 6p_2 + 3p_3 \\ \textit{subject to} & 0 \leq p_0, p_1, p_2, p_3 \leq \epsilon \\ & p_0 + 3p_1 + 3p_2 + p_3 = 1 \end{cases}$$

Security efficiency is $(-\log_2 \epsilon)/W$. Note that the original Fiat-Shamir protocol has $W = 3/2$ and security $S = 3$ bits, hence a security efficiency of $E = 2$, as pointed out previously.

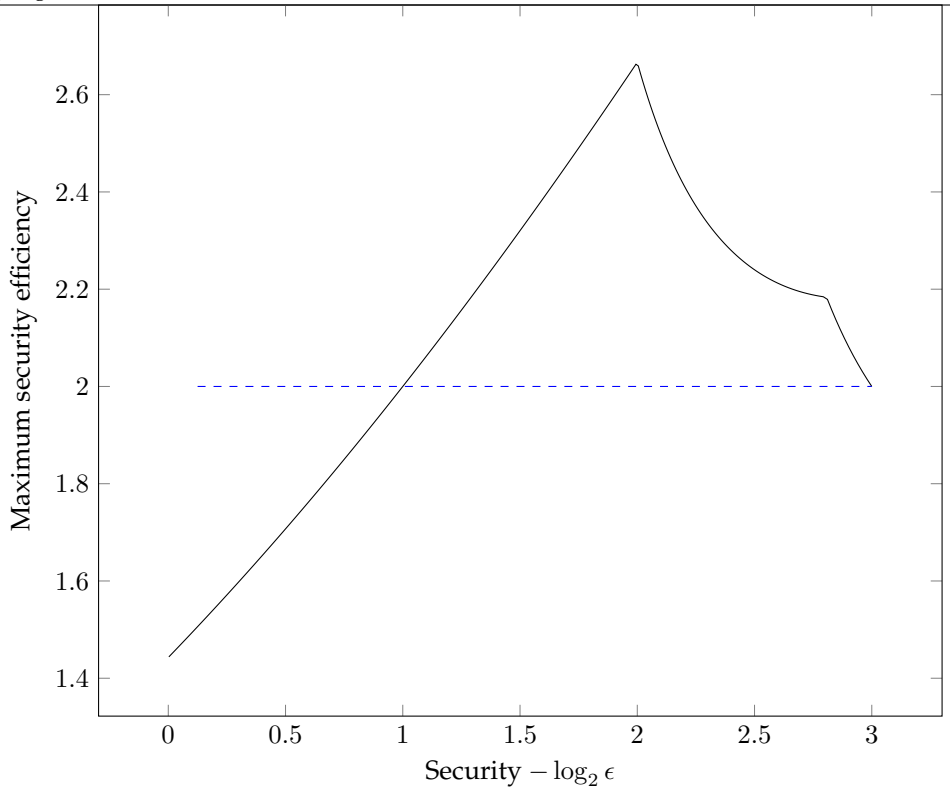


Figure 5.6 – Security efficiency for biased Fiat-Shamir with $n = 3$, as a function of ϵ. Standard Fiat-Shamir security efficiency corresponds to the dashed line.

Let for instance $\epsilon = 1/7$, for which the solution can be expressed simply as $p_0 = p_1 = p_2 = \epsilon$, and $p_3 = 1 - 7\epsilon$, yielding an effort

$$W = 9\epsilon + 3(1 - 7\epsilon) = 3(1 - 4\epsilon)$$

Therefore the corresponding security efficiency is $\frac{-\log_2 \epsilon}{3(1-4\epsilon)}$, which at $\epsilon = 1/7$ equals $7 \log_2 7/9 \simeq 2.18$. This is a 10% improvement over a standard Fiat-Shamir.

Remark We can compute the optimal distribution for any value of $\epsilon \geq 1/8$, *i.e.* choose the p_is that yields the maximum security efficiency $\hat{E}(\epsilon)$. The result of this computation is given in Figure 5.6. Corresponding optimal probabilities $\hat{p}_i$ are given in Figure 5.7.

Remark Figure 5.6 shows that $\hat{E}$ is not a continuously differentiable function of ϵ. The two singular points correspond to $\epsilon = 1/7$ and $\epsilon = 1/4$. These singular points correspond to optimal strategy changes: when ϵ gets large enough, it becomes interesting to reduce the probability of increasingly many symbols. This is readily observed on Figure 5.7 which displays the optimal probability distribution of each symbol group as a function of ϵ.

Example 5.2 *Solving Equation* (5.1) *for Fiat-Shamir with $n = 16$ gives Figure 5.8 which exhibits the same features as Figure 5.6, with more singular points positioned at $\epsilon = 2^{-4}, 2^{-7}, 2^{-9}$, etc.*

5.2.5 Thrifty SD, PKP and PPP

The authors implemented [10] the SD, PKP and PPP protocols, and timed their operation as a function of the challenge class. Only the relative time taken by each class is relevant, and can be used as a measure

10. Python source code is available upon request.

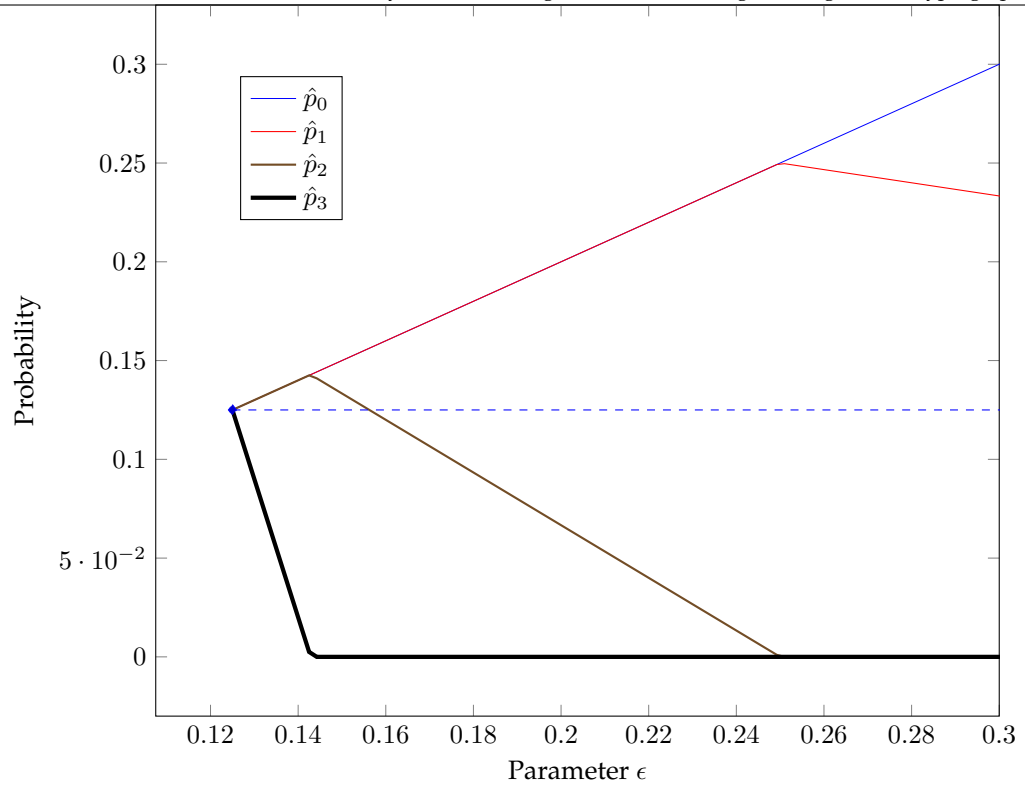


Figure 5.7 – Fiat-Shamir ($k = n = 4$) optimal probability distribution for challenges in group $j = 0, \ldots, 3$, as a function of ϵ. Branching happens at $\epsilon = 1/7$ and $\epsilon = 1/4$. Dashed line corresponds to the standard Fiat-Shamir distribution.

Table 5.1 – Challenge effort distribution for SD [Ste94], with a 16×16 parity matrix H, over 10^4 runs.

Challenge	Operations by prover	Time	Optimal p_i
0	Return y and σ	0 s ± 0.01	0.333
1	Compute $y \oplus s$	747.7 s ± 2	0.333
2	Compute $y \cdot \sigma$ and $s \cdot \sigma$	181.22 s ± 2	0.333

Table 5.2 – Challenge effort distribution for PKP [Sha90], over 10^7 runs.

Challenge	Operations by prover	Time	Optimal p_i
0	Compute W	390 s ± 2	0.5
1	Compute W and $\pi(\sigma)$	403 s ± 2	0.5

of $\mathcal{W}$. The methodology of Section 5.2.3 is then used to compute the optimal probability distributions and construct the thrifty variant of these protocols.

However, there is a peculiarity in these protocols: An adversary can correctly answer $(k-1)$ out of k possible challenges, requiring a legitimate prover to achieves more than $2/3$, $1/2$ and $3/4$ success rates respecitvely for SD, PKP and PPP. In this case, the attacker's optimal strategy is to bet on the most probable combination of $(k-1)$ challenges. Hence security is no longer measured by the min-entropy, but instead by $-\log_2 \min(p_i)$. In that case it is easily seen that the security efficiency cannot be improved, and linear optimisation confirms that the optimal parameters are that of uniform distributions.

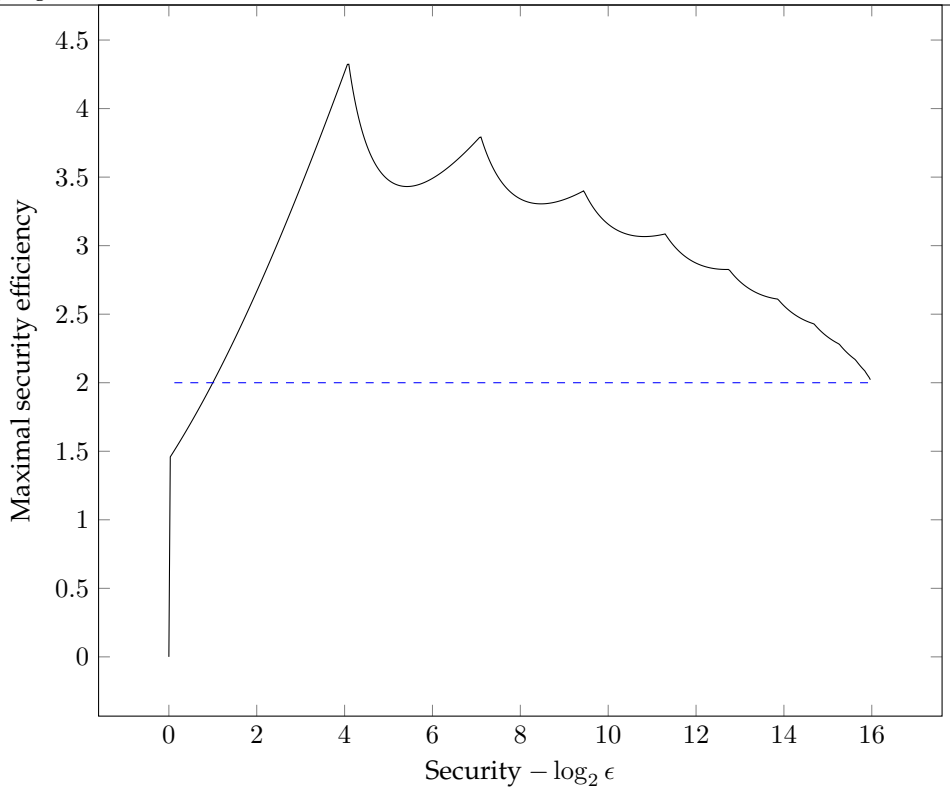


Figure 5.8 – Maximal security efficiency $\hat{E}$ for biased Fiat-Shamir with $n = 16$, as a function of security $-\log \epsilon$. Standard Fiat-Shamir security efficiency corresponds to the dashed line.

Table 5.3 – Challenge effort distribution for PPP [Poi95], over 10^6 runs.

Challenge	Operations by prover	Time	Optimal p_i
0	Return P, Q, W	0.206 s ± 0.05	0.25
1	Compute $W + Q^{-1}V$	6.06 s ± 0.05	0.25
2	Compute $Q(P(A))$ and $Q^{-1}V$	21.13 s ± 0.5	0.25
3	Compute $Q^{-1}V$	4.36 s ± 0.05	0.25

The result of measurements [11] and optimisations is summarized in Tables 5.1 to 5.3. For details about the protocols we refer the reader to the original descriptions. The code computing the thrifty Fiat-Shamir parameters is given in the appendix of this thesis.

11. Experiments were performed on a Intel Core i7-4712HQ CPU at 2.30 GHz, running Linux 3.13.0, Python 2.7.6, `numpy` 1.9.3, and `sympy` 0.7.6.1.

5.3 Public-Key Based Lightweight Swarm Authentication

We describe a lightweight algorithm performing whole-network authentication in a distributed way. This protocol is more efficient than one-to-one node authentication: it results in less communication, less computation, and overall lower energy consumption.

The proposed algorithm is provably secure, and achieves zero-knowledge authentication of a network in a time logarithmic in the number of nodes.

Related Work: Zero Knowledge (ZK) protocols have been considered for authentication of wireless sensor networks. For instance, Anshul and Roy [AR05] describe a modified version of the Guillou-Quisquater identification scheme [GQ88], combined with the μTesla protocol [PST+02] for authentication broadcast in constrained environments. We stress that the purpose of the scheme of [AR05], and similar ones, is to authenticate the base station.

Aggregate signature schemes such as [BGLS03; ZQWZ10] may be used to achieve the goal pursued here – however they are intrinsically non-interactive, and the most efficient aggregate constructions use elliptic curve pairings, which require powerful devices.

Closer to our concerns, [UMS11] describes a ZK network authentication protocol, but it only authenticates two nodes at a time, and the base station acts like a trusted third party. As such it takes a very large number of interactions to authenticate the network as a whole.

What we propose instead is a collective perspective on authentication and not an isolated one.

Organisation: Section 5.3.1 recalls the Fiat-Shamir authentication scheme and present a distributed algorithm for topology-aware networks. We describe our core idea, a distributed Fiat-Shamir protocol for IoT authentication, in Section 5.3.2. We analyze the security of the proposed protocol in Section 5.3.3. Section 5.3.4 provides several improvements and explores trade-offs between security, transmission and storage.

5.3.1 Preliminaries

5.3.1.1 Fiat-Shamir Authentication

The Fiat-Shamir authentication protocol [FS87] enables a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that $\mathcal{P}$ possesses a secret key without ever revealing the secret key [GMR85; FFS88].

The algorithm first runs a one-time setup, whereby a trusted authority publishes an RSA modulus $n = pq$ but keeps the factors p and q private. The prover $\mathcal{P}$ selects a secret $s < n$ such that $\gcd(n, s) = 1$, computes $v = s^2 \mod n$ and publishes v as its public key.

When a verifier $\mathcal{V}$ wishes to identify $\mathcal{P}$, he uses the protocol of Figure 5.9. $\mathcal{V}$ may run this protocol several times until $\mathcal{V}$ is convinced that $\mathcal{P}$ indeed knows the square root s of v modulo n.

Figure 5.9 describes the original Fiat-Shamir authentication protocol [FS87], which is *honest verifier* zero-knowledge [12], and whose security is proven assuming the hardness of computing arbitrary square roots modulo a composite n, which is equivalent to factoring n.

As pointed out by [FS87], instead of sending x, $\mathcal{P}$ can hash it and send the first bits of $H(x)$ to $\mathcal{V}$, for instance the first 128 bits. With that variant, the last step of the protocol is replaced by the computation of $H(y^2 \prod_{i=1}^{k} v_i^{-a_i} \mod n)$, truncated to the first 128 bits, and compared to the value sent by $\mathcal{P}$. Using this "short commitment" version reduces somewhat the number of communicated bits. However, it comes at the expense of a reduced security level. A refined analysis of this technique in given in [GS02].

12. This can be fixed by requiring $\mathcal{V}$ to commit to the a_i before $\mathcal{P}$ has sent anything, but this modification will not be necessary for our purpose.

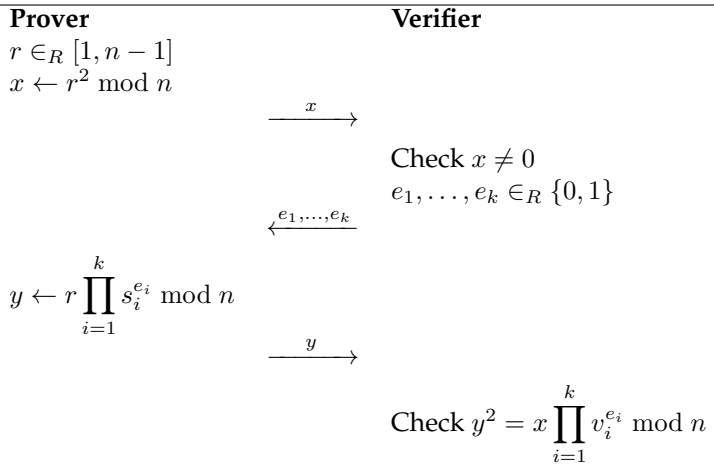


Figure 5.9 – Fiat-Shamir authentication protocol.

5.3.1.2 Topology-Aware Distributed Spanning Trees

Due to the unreliable nature of sensors, their small size and wireless communication system, the overall network topology is subject to change. Since sensors send data through the network, a sudden disruption of the usual route may result in the whole network shutting down.

5.3.1.2.1 Topology-Aware Networks. A *topology-aware* network detects changes in the connectivity of neighbors, so that each node has an accurate description of its position within the network. This information is used to determine a good route for sending sensor data to the base station. This could be implemented in many ways, for instance by sending discovery messages (to detect additions) and detecting unacknowledged packets (for deletions). Note that the precise implementation strategy does not impact the algorithm.

Given any graph $G = (V, E)$ with a distinguished vertex B (the base station), the optimal route for any vertex v is the shortest path from v to B on the minimum degree spanning tree $S = (V, E')$ of G. Unfortunately, the problem of finding such a spanning tree is **NP**-hard [SL07], even though there exist optimal approximation algorithms [SL07; LV08]. Any spanning tree would work for the proposed algorithm, however the performance of the algorithm gets better as the spanning tree degree gets smaller.

5.3.1.2.2 Mooij-Goga-Wesselink's Algorithm. The network's topology is described by a spanning tree W constructed in a distributed fashion by the Mooij-Goga-Wesselink algorithm [MGW03]. We assume that nodes can locally detect whether a neighbor has appeared or disappeared, *i.e.* graph edge deletion and additions.

W is constructed by aggregating smaller subtrees together. Each node in W is attributed a "parent" node, which already belongs to a subtree. The complete tree structure of W is characterized by the parenthood relationship, which the Mooij-Goga-Wesselink algorithm computes. Finally, by topological reordering, the base station $\mathcal{T}$ can be put as the root of W.

Each node in W has three local variables $\{\mathsf{parent}, \mathsf{root}, \mathsf{dist}\}$ that are initially set to a null value $\perp$. Nodes construct distributively a spanning tree by exchanging "M-messages" containing a root information, distance information and a type. The algorithm has two parts:

— *Basic*: maintains a spanning tree as long as no edge is removed (it is a variant of the union-find algorithm [CSRL01]). When a new neighbor w is detected, a discovery M-message $(\mathsf{root}, \mathsf{dist})$ is sent to it. If no topology change is detected for w, and an M-message is received from it, it is processed by Algorithm 9. Note that a node only becomes active upon an event such as the arriving of an M-message or a topology change.

— *Removal*: intervenes after the edge deletion so that the basic algorithm can be run again and give correct results.

Algorithm 9 Mooij-Goga-Wesselink algorithm, basic part.

Input: An M-message (r, d) coming from a neighbor w

```
1 (parent, root, dist) ← (⊥, ⊥, ⊥)
  if (r, d + 1) < (root, dist) then
2     parent ← w
      root ← r
      dist ← d + 1
      Send the M-message (root, dist) to all neighbors except w
3 end if
```

Algorithm 9 has converged once all topology change events have been processed. At that point we have a spanning tree [MGW03].

For our purposes, we may assume that the network was set up and that Algorithm 9 is running on it, so that at all times the nodes of the network have access to their parent node. Note that this incurs very little overhead as long as topology changes are rare.

5.3.2 Distributed Fiat-Shamir Authentication

5.3.2.1 The Approach

Given a k-node network $\mathcal{N}_1, ..., \mathcal{N}_k$, we may consider the nodes $\mathcal{N}_i$ as users and the base station as a trusted center $\mathcal{T}$. In this context, each node will be given only an [13] s_i. To achieve collective authentication, we propose the following Fiat-Shamir based algorithm:

— *Step 0:* Wait until the network topology has converged and a spanning tree W is constructed with Algorithm 9 presented in Section 5.3.1.2. When that happens, $\mathcal{T}$ sends an authentication request message (AR-message) to all the $\mathcal{N}_i$ directly connected to it. The AR-message may contain a commitment to e (*cf.* Step 2) to guarantee the protocol's zero-knowledge property even against dishonest verifiers.

— *Step 1:* Upon receiving an AR-message, each $\mathcal{N}_i$ generates a private r_i and computes $x_i \leftarrow r_i^2 \mod n$. $\mathcal{N}_i$ then sends an A-message to all its children, if any. When they respond, $\mathcal{N}_i$ multiplies all the x_j sent by its children together, and with its own x_i, and sends the result up to its own parent. This recursive construction enables the network to compute the product of all the x_is and send the result x_c to the top of the tree in d steps (where $d = \deg W$). This is illustrated for a simple network including 4 nodes and a base station in Figure 5.10.

— *Step 2:* $\mathcal{T}$ sends a random e as an authentication challenge (AC-message) to the $\mathcal{N}_i$ directly connected to it.

— *Step 3:* Upon receiving an AC-message e, each $\mathcal{N}_i$ computes $y_i \leftarrow r_i s_i^{e_i}$. $\mathcal{N}_i$ then sends the AC-message to all its children, if any. When they respond, $\mathcal{N}_i$ multiplies the y_j values received from all its children together, and with its own y_i, and sends the result to its own parent. The network therefore computes collectively the product of all the y_i's and transmits the result y_c to $\mathcal{T}$. This is illustrated in Figure 5.11.

— *Step 4:* Upon receiving y_c, $\mathcal{T}$ checks that $y_c^2 = x_c \prod v_i^{e_i}$, where $v_1, \ldots, v_k$ are the public keys corresponding to $s_1, \ldots, s_k$ respectively.

Note that the protocol may be interrupted at any step. In the version of the algorithm that we have just described, this results in a failed authentication.

13. This is for clarity. It is straightforward to give each node several private keys, and adapt the algorithm accordingly.

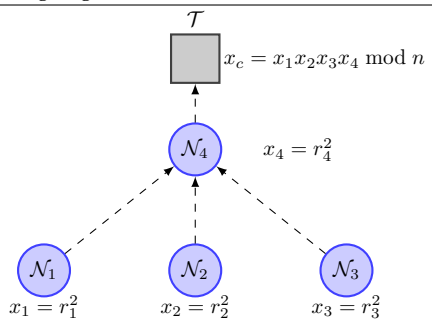


Figure 5.10 – The construction of x_c.

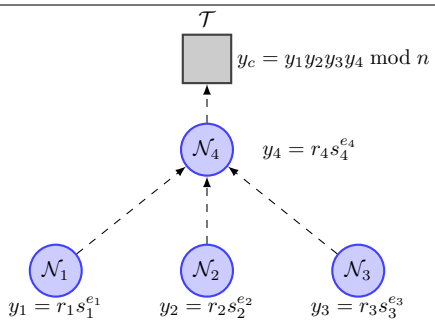


Figure 5.11 – The construction of y_c.

Figure 5.12 – The proposed algorithm running on a network. Each parent node aggregates the values computed by its children and adds its own information before transmitting the result upwards to the base station.

5.3.2.2 Back-up Authentication

Network authentication may fail for many reasons described and analyzed in detail in Section 5.3.3.3.3. As a consequence of the algorithm's distributed nature that we have just described, a single defective node suffices for authentication to fail.

This is the intended behavior; however there are contexts in which such a brutal answer is not enough, and more information is needed. For instance, one could wish to know *which* node is responsible for the authentication failure.

A simple back-up strategy consists in performing *usual* Fiat-Shamir authentication with all the nodes that still respond, to try and identify where the problem lies. Note that, as long as the network is healthy, using our distributed algorithm instead is more efficient and consumes less bandwidth and less energy.

Since all nodes already embark the hardware and software required for Fiat-Shamir computations, and can use the same keys, there is no real additional burden in implementing this solution.

5.3.3 Security Proofs

In this section we wish to discuss the security properties relevant to our construction. The first and foremost fact is that algorithm given in Figure 5.11 is *correct*: a legitimate network will always succeed in proving its authenticity, provided that packets are correctly transmitted to the base station $\mathcal{T}$ (possibly hopping from node to node) and that nodes perform correct computations.

The interesting part, therefore, is to understand what happens when such hypotheses do *not* hold.

5.3.3.1 Soundness

Lemma 5.5 (Soundness) *If the authentication protocol of Section 5.3.2.1 succeeds then with overwhelming probability the network nodes are genuine.*

Proof: Assume that an adversary $\mathcal{A}$ simulates the whole network, but does not know the s_i, and cannot compute in polynomial time the square roots of the public keys v_i. Then, as for the original Fiat-Shamir protocol [FS87], the base station will accept $\mathcal{A}$'s identification with probability bounded by $1/2^k$ where k is the number of nodes.

□

5.3.3.2 Zero-knowledge

Lemma 5.6 (Zero-knowledge) *The distributed authentication protocol of Section 5.3.2.1 achieves statistical zero-knowledge.*

Proof: Let $\mathcal{P}$ be a prover and $\mathcal{A}$ be a (possibly cheating) verifier, who can use any adaptive strategy and bias the choice of the challenges to try and obtain information about the secret keys.

Consider the following simulator $\mathcal{S}$:

Step 1. Choose $\overline{e} \in_R \{0,1\}^k$ and $\overline{y} \in_R [0, n-1]$ using any random tape ω'

Step 2. Compute $\overline{x} \leftarrow \overline{y}^2 \prod v_i^{\overline{e}_i}$ and output $(\overline{x}, \overline{e}, \overline{y})$.

The simulator $\mathcal{S}$ runs in polynomial time and outputs triples that are indistinguishable from the output of a prover that knows the corresponding private key.

If we assume the protocol is run N times, and that $\mathcal{A}$ has learnt information which we denote η, then $\mathcal{A}$ chooses adaptively a challenge using all information available to it $e(x, \eta, \omega)$ (where ω is a random tape). The proof still holds if we modify $\mathcal{S}$ in the following way:

Step 1. Choose $\overline{e} \in_R \{0,1\}^k$ and $\overline{y} \in_R [0, n-1]$ using any random tape ω'

Step 2. Compute $\overline{x} \leftarrow \overline{y}^2 \prod v_i^{\overline{e}_i}$

Step 3. If $e(\overline{x}, \eta, \omega) = \overline{e}$ then go to Step 1 ; else output $(\overline{x}, \overline{e}, \overline{y})$.

Note that the protocol is also "locally" ZK, in the sense that an adversary simulating ℓ out of k nodes of the network still has to face the original Fiat-Shamir protocol.

□

5.3.3.3 Security Analysis

5.3.3.3.1 Choice of Parameters. Let λ be a security parameter. To ensure this security level the following constraints should be enforced on parameters:

— The identification protocol should be run $t \geq \lceil \lambda / k \rceil$ times (according to Lemma 5.5), which is reasonably close to one as soon as the network is large enough;
— The modulus n should take more than $2^{\lambda t}$ operations to factor;
— Private and public keys are of size comparable to n.

5.3.3.3.2 Complexity. The number of operations required to authenticate the network depends on the exact topology at hand, but can safely be bounded above:

— Number of modular squarings: $2kt$
— Number of modular multiplications $\leq 3kt$

In average, each $\mathcal{N}_i$ performs only a constant (a small) number of operations. Finally, only $O(d)$ messages are sent, where d is the degree of the minimum spanning tree of the network. Pathological cases aside, $d = O(\log k)$, so that only a logarithmic number of messages are sent during authentication.

All in all, for $\lambda = 256$, $k = 1024$ nodes and $t = 1$, we have $n \geq 2^{1024}$, and up to 5 modular operations per node.

5.3.3.3.3 Root Causes of Authentication Failure. Authentication may fail for several reasons. This may be caused by network disruption, so that no response is received from the network – at which point not much can be done.

However, more interestingly, $\mathcal{T}$ may have received an invalid value of y_c. The possible causes are easy to spot:

1. A topology change occurred during the protocol:

— If all the nodes are still active and responding, the topology will eventually converge and the algorithm will get back to Step 0.
— If however, the topology change is due to nodes being added or removed, the network's integrity has been altered.

2. A message was not transmitted: this is equivalent to a change in topology.
3. A node sent a wrong result. This may stem from low battery failure or when errors appear within the algorithm the node has to perform (fault injection, malfunctioning, etc). In that case authentication is expected to fail.

5.3.3.3.4 Effect of Network Noise. Individual nodes may occasionally receive incorrect (ill-formed, or well-formed but containing wrong information) messages, be it during topology reconstruction (M-messages) or distributed authentication (A-messages). Upon receiving incorrect A or M messages, nodes may dismiss them or try and acknowledge them, which may result in a temporary failure to authenticate. An important parameter which has to be taken into account in such an authentication context is the number of children of a node (fanout). When a node with many children starts failing, all its children are disconnected from the network and cannot be contacted or authenticated anymore. While a malfunction at leaf level might be benign, the failure of a fertile node is catastrophic.

5.3.3.3.5 Man-in-the-Middle. An adversary could install itself between nodes, or between nodes and the base station, and try to intercept or modify communications. Lemma 5.6 proves that a passive adversary cannot learn anything valuable, and Lemma 5.5 shows that an active adversary cannot fool authentication.

It is still possible that the adversary *relays* information, but any attempt to intercept or send messages over the network would be detected.

5.3.4 Variants and Implementation Trade-offs

The protocol may be adapted to better fit operational constraints: in the context of IoT for instance communication is a very costly operations. We describe variants that aim at reducing the amount of information sent by individual nodes, while maintaining security.

5.3.4.1 Shorter Challenges Variant

In the protocol of Section 5.3.2, the *long* (say, 128-bit) challenge e is sent throughout the network to all individual nodes. One way to reduce the length of e without compromising security is the following:
— A *short* [14] (say, 80-bit) value e is sent to the nodes;
— Each node i computes $e_i \leftarrow H(e\|i)$, and uses e_i as a challenge;
— The base station also computes e_i the same way, and uses $\{e_1, ..., e_k\}$ to check authentication.
This variant does not impact security, assuming an ideal hash function H, and it can be used in conjunction with the other improvements described below.

5.3.4.2 Multiple Secret Variant

Instead of keeping one secret value s_i, each node could have multiple secret values $s_{i,1}, \ldots, s_{i,\ell}$. Note that these additional secrets need not be stored: they can be derived from a secret seed.

The multiple secret variant is described here for a single node, for the sake of clarity. Upon receiving a challenge e_i (assuming for instance that e_i was generated by the above procedure), each node computes

14. but sufficiently long in terms of entropy

a response

$$y_i \leftarrow r_i \prod_{j=1}^{\ell} s^{e_{i,j}} \bmod n$$

This can be checked by the verifier by checking whether

$$y_i^2 \stackrel{?}{=} x_i \prod_{j=1}^{\ell} v_{i,j}^{e_{i,j}} \bmod n.$$

To achieve swarm authentication, it suffices to perform aggregation as described in the protocol of Section 5.3.2 at intermediate nodes.

Using this approach, one can adjust the memory-communication trade-off, as the security level is $\lambda = t\ell$ (single-node compromission). Therefore, if $\ell = 80$ for instance, it suffices to authenticate once to get the same security as $t = 80$ authentications with $\ell = 1$ (which is the protocol of Section 5.3.2). This drastically cuts bandwidth usage, a scarce resource for IoT devices.

Furthermore, computational effort can be reduced by using batch exponentiation techniques [MN96; BGR98] to compute y_i.

5.3.4.3 Pre-computed Alphabet Variant

The security level we aim at is 80 bits. A way to further reduce computational cost is the following: each node chooses an alphabet of m words $w_0, \ldots, w_{m-1}$ (a word is a 32-bit value), and computes once and for all the table of all pairwise products $p_{i,j} = m_i m_j$. Note that each $p_{i,j}$ entry is 64 bits long.

The values s_i are generated by randomly sampling from the alphabet of ws. Put differently, s_i is built by concatenating u words (bit patterns) taken from the alphabet only.

We thus see that each s_i, which is an mu-bit integer, can take m^u possible values. For instance if $m = u = 32$ then s_i is a 1024-bit number chosen amongst $32^{32} = 2^{160}$ possible values. Thanks to the lookup table, word by word multiplications need not be performed, which provides a substantial speed-up over the naive approach.

The size of the lookup table is moderate, for the example given, all we need to store is $32 \times 31/2 + 32 = 528$ values. This can be further reduced by noting that the first lines in the table can be removed: 32 values are zeros, 31 values are the results of multiplications by 1, 30 values are left shifts by 1 of the previous line, 29 values are the sum of the previous 2 and 28 values are left shifts by 2. Hence all in all the table can be compressed into $528 - 32 - 31 - 29 - 28 = 408$ entries. Because each entry is a word, this boils-down to 1632 bytes only.

5.3.4.4 Pre-computed Combination Variant

Computational cost can be also cut down if we pre-compute and store some products, only to assemble them online during Fiat-Shamir authentication: in this variant the values of $s_{i,1,2} \leftarrow s_{i,1}s_{i,2}$, $s_{i,2,3} \leftarrow s_{i,2}s_{i,3}$, ... , etc. are stored in a lookup table.

The use of combined values $s_{i,a,b}$ in the evaluation of y results in three possible scenarios for each:

1. $s_a s_b$ appears in y – the probability of this occurring is $1/4$ – in which case one additional multiplication must be performed;
2. $s_a s_b$ does not appear in y – the probability of this occurring is $1/4$ – in which case no action is performed;
3. s_a or s_b appears, but not both – this happens with probability $1/2$ – in which case one single multiplication is required.

Consequently the expected number of multiplications is reduced by 25%, to wit $\frac{3}{4} \times 2^{m-1}$, where m is the size of e.

The method can be extended to work with a window of size $\kappa \geq 2$, for instance with $\kappa = 3$ we would pre-compute:

$$
\begin{aligned}
s_{i,3j,3j+1} &\leftarrow s_{i,3j} \times s_{i,3j+1} \\
s_{i,3j+1,3j+2} &\leftarrow s_{i,3j+1} \times s_{i,3j+2} \\
s_{i,3j,3j+2} &\leftarrow s_{i,3j} \times s_{i,3j+2} \\
s_{i,3j,3j+1,3j+2} &\leftarrow s_{i,3j} \times s_{i,3j+1} \times s_{i,3j+2}
\end{aligned}
$$

Following the same analysis above, the expected number of multiplications during the challenge-response phase is $\frac{7}{8} \times \frac{2^m}{3}$. The price to pay is that larger κ values claim more pre-computing and more memory.

More precisely, we have the following trade-offs, writing $\mu = 2^m \bmod \kappa$:

$$
\begin{aligned}
\text{Multiplications (expected)} &= 2^m \left(\frac{2^\kappa - 1}{2^\kappa} \left(\left\lfloor \frac{2^m}{\kappa} - 1 \right\rfloor \right) - \frac{2^\mu - 1}{2^\mu} \right) \\
\text{Pre-multiplications} &= \ell - 1 + \left((2^\kappa - \kappa - 1) \left\lfloor \frac{2^m}{\kappa} \right\rfloor \right) + (2^\mu - \mu - 1) \\
\text{Stored Values} &= (2^\kappa - 1) \left\lfloor \frac{2^m}{\kappa} \right\rfloor + (2^\mu - 1)
\end{aligned}
$$

where ℓ is the number of components of s_i.

5.4 When Organized Crime Applies Academic Results

5.4.1 Introduction

EMV [EMV; EMV08a; EMV08b; EMV08c] (Europay, MasterCard, Visa) is a global standard, currently managed by the public corporation EMVCo, specifying interactions between integrated circuit cards and PoS terminals. The standard also defines exchanges between cards and automatic teller machines (ATMs). Over the recent years, additional payment operators (such as JCB, AmericanExpress, China UnionPay and Discover) endorsed EMV. EMV cards rely on pre-existing physical, link, network, and transport layer protocols such as ISO/IEC 7816 and ISO/IEC 14443.

According to EMVCo's website, by Q4 2014 a third of card present transactions worldwide followed the EMV protocol, and 3.423 billion EMV cards were in circulation.

5.4.1.1 Brief Overview of an EMV Transaction

A typical EMV transaction breaks down into three phases: ① card authentication, ② cardholder verification and ③ transaction authorization.

During card authentication, the PoS explores the applications supported by the card (*e.g.* credit, debit, loyalty, ATM, etc.).

During cardholder verification, the PoS queries the PIN from the user and transmits it to the card. The card compares the PIN and responds by "yes" (`SW` code [15] `0x9000`) or "no" (`0x63CX` [16]).

Transaction authorization starts by feeding the card with the transaction details T (*e.g.* amount, currency, date, terminal ID, fresh randomness, etc.). The card replies with an authorization request cryptogram (ARQC) based on T. {ARQC, T} is sent to the issuer [17], who replies with an authorization request code (ARC) instructing the PoS how the transaction should proceed. The issuer also sends to the PoS an authorization response cryptogram (ARPC) which is a MAC of {ARQC, ARC}. ARPC is transmitted to the card that responds with a transaction certificate (TC) sent to the issuer to finalize the transaction.

We refer the reader to [MDAB10] for a comprehensive diagram illustrating these three phases.

5.4.1.2 Murdoch et al.'s Attack

The protocol vulnerability described in [MDAB10] is based on the fact that the card does not condition transaction authorization on successful cardholder verification.

Hence the attack consists in having the genuine card execute the first and last protocol phases, while leaving the cardholder verification to a man-in-the-middle device.

To demonstrate this scenario's feasibility, Murdoch *et al.* produced an FPGA-based proof-of-concept, noting that miniaturization remains a mechanical challenge.

5.4.1.3 Fraud in the Field

In May 2011, the French's bankers Economic Interest Group (GIE Cartes Bancaires) noted that a dozen EMV cards, stolen in France a few months before, were being used in Belgium. A police investigation was thus triggered.

15. Whenever a command is executed by a card, the card returns two status bytes called `SW1` and `SW2`. These bytes encode a success or a failure cause.
16. `X` denotes the number of further PIN verifications remaining before lock-up.
17. For our purposes, the issuer can be thought of as the bank.

Figure 5.13 – The judicial seizure. Personal information such as cardholder name are censored for privacy reasons.

Because transactions take place at well-defined geographic locations and at well-defined moments in time, intersecting the IMSIs [18] of SIM cards present near the crime scenes immediately revealed the perpetrators' SIM card details. A 25 years old woman was subsequently identified and arrested, while carrying a large number of cigarette packs and scratch games. Such larceny was the fraudsters' main target, as they resold these goods on the black market.

Investigators quickly put a name on most of the gang members. Four were arrested, including the engineer who created the fake cards. Arrests occurred in the French cities of Ezanville, Auchy-les-Mines and Rouvroy. About 25 stolen cards were seized, as well as specialized software and €5000 in cash.

The net loss caused by this fraud is estimated to stand below €600,000, stolen over 7,000 transactions using 40 modified cards.

A forensic investigation was hence ordered by Justice [Jus].

5.4.2 Physical Analysis

5.4.2.1 Optical Inspection

The forgery appears as an ISO/IEC 7816 smart card. The forgery's plastic body indicates that the card is a VISA card issued by Caisse d'Épargne (a French bank). The embossed details are: PAN [19] = `4978**********89`; expiry date in 2013 [20]; and a cardholder name, hereafter abridged as P.S. The forgery's backside shows a normally looking CVV [21]. Indeed, this PAN corresponds to a Caisse d'Épargne VISA card.

The backside is deformed around the chip area (Figure 5.14). Such a deformation is typically caused by heating. Heating (around 80°C) allows melting the potting glue to detach the card module.

The module looks unusual in two ways: ① it is engraved with the inscription "FUN"; and ② glue traces clearly show that a foreign module was implanted to replace the `**89` card's original chip (Figure 5.15).

The module is slightly thicker than normal, with the chip bulging somewhat through the card, making insertion into a PoS somewhat uneasy but perfectly feasible (Figure 5.16).

18. International Mobile Subscriber Identity.
19. Permanent Account Number (partially anonymized here).
20. Precise date removed for privacy reasons.
21. Card Verification Value.

Figure 5.14 – Deformation due to heating of the forgery's backside.

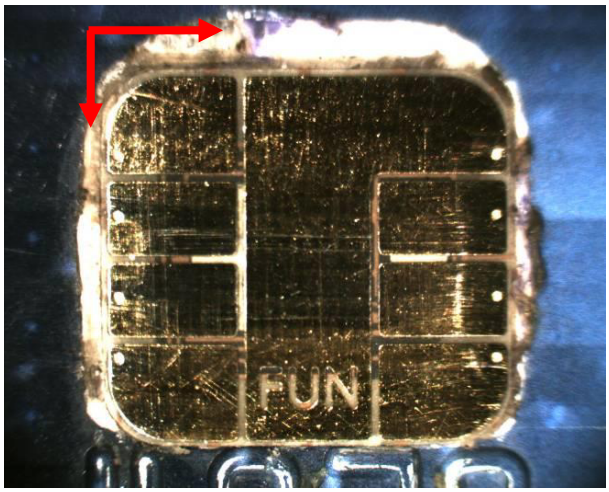


Figure 5.15 – Forgery's ISO module. Red arrows show glue traces.

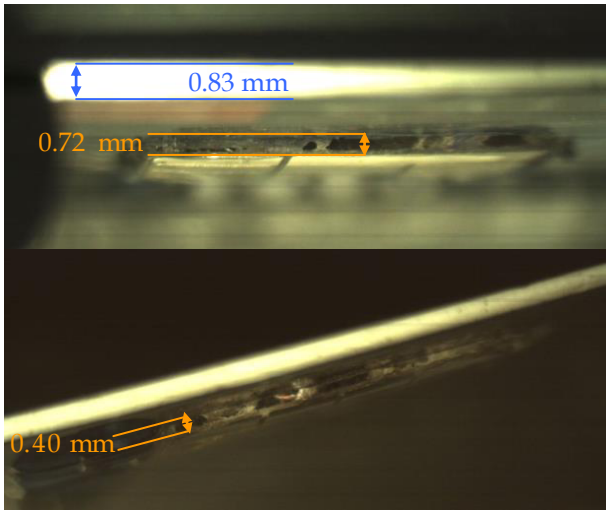


Figure 5.16 – Side-views of the forgery, showing that it is somewhat thicker than a standard card (0.83 mm). The extra thickness varies from 0.4 mm to 0.7 mm suggesting the existence of several components under the card module, besides the FUN card.

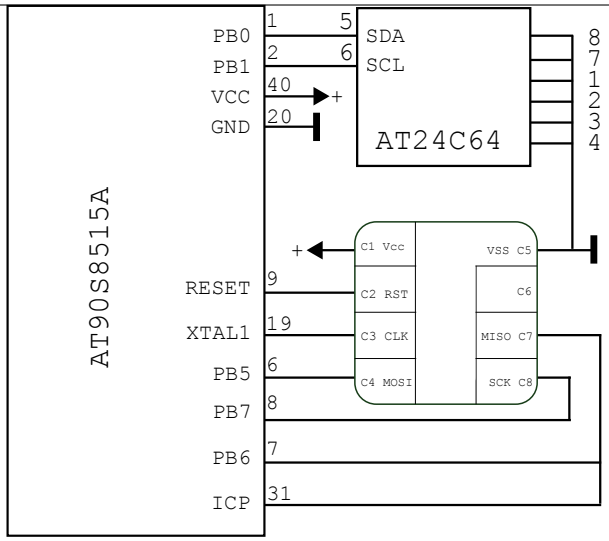


Figure 5.17 – The FUN card's inner schematics.

The "FUN" engraving indicates that the module belongs to a FUN card. FUN cards are open cards, widely used for hobbying and prototyping purposes.

The FUN module contains an Atmel AVR AT90S8515 microcontroller and an EEPROM memory AT24Cxx. The AVR has 8 kB of Flash memory and 512 bytes of internal EEPROM, 512 bytes of internal RAM and a few other resources (timer, etc.). The AT24Cxx has varying capacity depending on the exact FUN card model. For a FUNcard5, this capacity is 512 kB (Figure 5.17).

5.4.2.2 Magnetic Stripe Analysis

The magnetic stripe was read and decoded. The ISO1 and ISO2 tracks perfectly agrees with the embossed information. ISO3 is empty, as is usual for European cards.

5.4.2.3 X-Ray Analysis

X-ray analysis was performed using a Y.Cougar Microfocus Xylon imager. Figure 5.18 shows an unmodified FUN card, while Figure 5.19 is an X-ray image of the forgery.

X-ray analysis reveals, using false colors, the different materials composing the forged module (Figure 5.21). Legitimate connection wires are made of gold, but connections between the FUN card and the stolen chip underneath are made of another metal (copper, as will later appear after opening the forged card). Soldering was made using a classical mixture of silver and tin.

5.4.2.4 Probing non-ISO Contacts

FUN cards are programmed using specialized hardware.

Programming is done *via* the two unstandardized pins `MOSI` and `SCK`.

We tried to use programming hardware to read back the card's contents and reverse-engineer its software.

All reading attempts failed. FUN cards can be protected against reading at flashing time. Clearly, the fraudster enabled this protection.

It is possible to identify the chip using the programming hardware, but this uses writing commands that are invasive and possibly destructive. Therefore such an identification was not attempted.

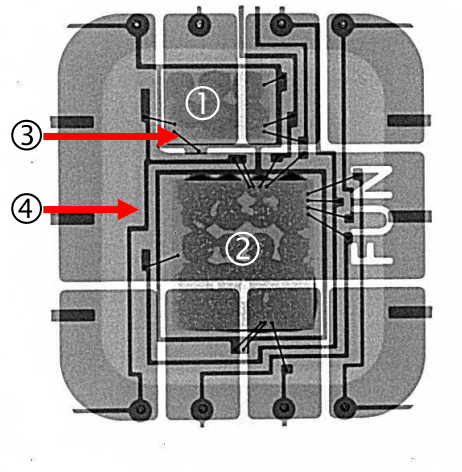


Figure 5.18 – FUN card X-ray analysis. ① External memory (AT24C64); ② Microcontroller (AT90S8515A); ③ Connection wires; ④ Connection grid.

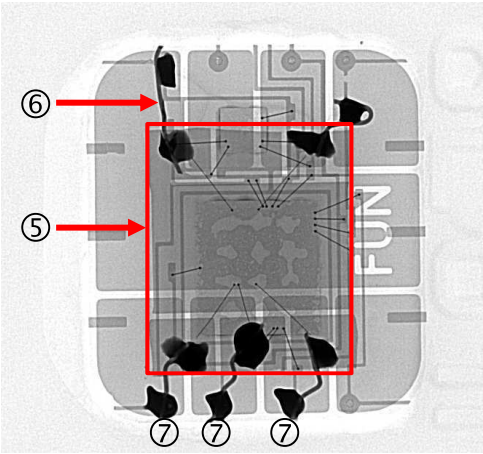


Figure 5.19 – Forgery X-ray analysis. ⑤ Stolen card's module; ⑥ Connection wires added by the fraudster; ⑦ Weldings by the fraudster (only three are pointed out here).

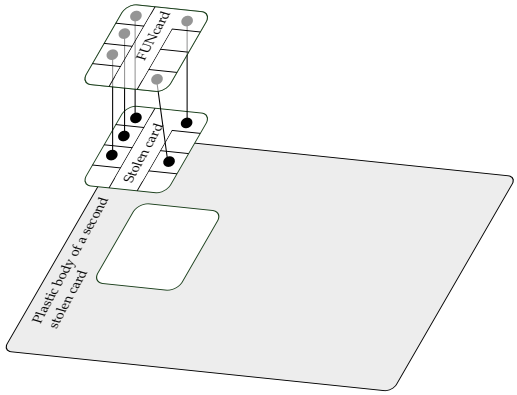


Figure 5.20 – Forgery structure suggested by Figure 5.19.

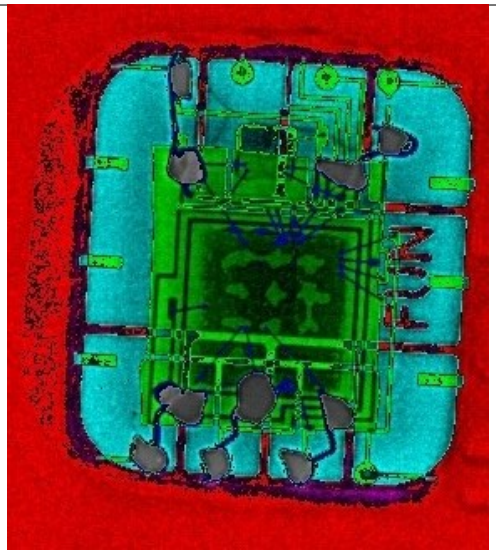


Figure 5.21 – False colors X-ray image of the forgery. Different colors correspond to different materials. The stolen chip is clearly visible in green.

5.4.2.5 ISO/IEC 7816 Compliance

We assumed that the forged card's software was rudimentary and did not fully comply with ISO/IEC 7816. The assumption was that the fraudsters contented themselves with a minimal implementation that works reasonably well under usual conditions. We hence analyzed the forgery's behavior at external clock frequencies close to the most extreme value (5 MHz) allowed by ISO/IEC 7816.

The forgery behaved normally up to 4.90 MHz. At 4.91 MHz, the forgery stopped responding to commands and only returned an ATR (Answer To Reset).

5.4.3 Protocol Analysis

The electronic exchanges between the forgery and the PoS were monitored using the `Cardpeek` [22] tool. `Cardpeek` allows monitoring the APDU commands sent to the forgery.

This is a read-only operation that does not alter the analyzed card.

When queried, the forgery responded with the following information: PAN = `4561**********79`; expiry date in 2011; and the cardholder name henceforth refered to as H.D. All this information is in blatant contradiction with data embossed on the card (mentioned in Section 5.4.2.1).

5.4.3.1 Application Selection

Application selection is performed by browsing the PSE [23], as described in [EMV08a].

5.4.3.1.1 Select 1PAY.SYS.DDF01. Selecting the DDF [24] named `1PAY.SYS.DDF01` succeeded [25].

22. See `http://code.google.com/p/cardpeek/downloads/list`.
23. Payment System Environment.
24. Directory Definition File.
25. Command: `00 A4 04 00 14`.

5.4.3.1.2 Browsing the Payment System Directory Records in the Payment System Directory were browsed in-order and revealed the presence of CB [26] and VISA applications.

ReadRecord SFI [27] 1 record #1 [28]: this SFI contains:
- Application AID `A0 00 00 00 42 10 10`
- Label: CB
- Priority: 1

ReadRecord SFI 1 record #2: this SFI contains:
- Application AID: `A0 00 00 00 03 10 10`
- Label: Visa DEBIT
- Priority: 2

Attempting ReadRecord SFI 1 record #3 returns a status word equal to `0x6A83`, *i.e.* "record not found". All applications have thus been found.

5.4.3.1.3 Select "VISA Debit" `Cardpeek` used the previously discovered AID to select the VISA Debit application [29].

5.4.3.2 Transaction Initialization

5.4.3.2.1 GetProcessingOptions (GPO) Next, we retrieved the card's processing options [30]. This data contains the AIP (Application Interchange Profile) and the AFL (Application File Locator) as defined in [EMV08c, Chapter 10.2]. The card claims that it supports:
- static authentication (SDA);
- dynamic authentication (DDA);
- cardholder verification;
- and external authentication.

The card furthermore requests risk management by the PoS.

AFL consists in a list of 4-byte blocks describing which records should be read. In our case, the following blocks were received:
- SFI #1, of record 01 to 01. No record of this SFI is used for "disconnected" mode data authentication.
- SFI #2, of record 01 to 02. Record #1 of this SFI is used for "disconnected" mode data authentication.
- SFI #3, of record 01 to 04. Record #1 of this SFI is used for "disconnected" mode data authentication.

5.4.3.2.2 SFI Records Having read the GPO data, the reader can access the SFI records.

ReadRecord SFI 2 record #1 (used for "disconnected" mode data authentication) contained the following VISA application information:
- Application creation and expiry date: between a date in 2009 and a date in 2011 (omitted here for privacy reasons).
- The card's PAN: `4561**********79`.

The Bank Identification Number of this PAN corresponds to a HSBC VISA card (4561), which is inconsistent with the information embossed on the card.

ReadRecord SFI 2 record #2 of the VISA application provided the following information:
- The list of objects to be included upon the first GenerateAC (CDOL1) (tags list + lengths)
- The list of objects to be included upon the second GenerateAC (CDOL2).
- The list of objects to be included upon internal authentication (DDOL):

26. Carte Bancaire.
27. Short File Identifier.
28. Command: `00 B2 xx 0C Le`, where `xx` is incremented as records are being read.
29. Command: `00 A4 04 00 07`.
30. Command: `80 A8 00 00 02` followed by a GetResponse command: `00 C0 00 00 20`.

— Tag `0x9F37` – "Unpredictable Number" (length: 4 octets)
— Cardholder's name: abridged here as H.D. for privacy reasons.

The chip belongs to Mr. H.D., which is also inconsistent with the information embossed on the card.

ReadRecord SFI 3 record #1, 2, 3, 4 (used for "disconnected" mode data authentication) contained actions codes, requested by the card to authorize a transaction, as well as a list of supported authentication methods, their public keys and certificates.

ReadRecord SFI 1 record #1 should have revealed the exact same information encoded in the ISO2 track. Instead, it contained, again, the following information:

— Account number: `4561**********79`
— Expiration date (YYMM): a date in 2011 (anonymised for privacy reasons)

ReadRecord SFI 4 record #1 indicated an empty record.

5.4.3.3 Authentications

5.4.3.3.1 InternalAuthenticate In smart card terms, an InternalAuthenticate[31] is an authentication of the card by the reader (*cf.* Chapter 6.5 of [EMV08b]). The reader requests that the card signs a random 4-byte number, as asked by the DDOL.

The reader accepted this authentication.

5.4.3.3.2 VerifyPIN (Cardholder verification) The reader checks that the card isn't blocked by reading the number of remaining PIN presentation tries[32]. There are 3 remaining tries before the card is blocked.

PIN codes are verified using the command VerifyPIN[33]. A correct PIN results in a `0x9000` status word.

Our experiments reveal that the PIN is always considered correct, regardless of P1 and P2, even for inconsistent values. The card accepts any PIN unconditionally.

5.4.3.4 Transaction

The reader gathers risk management data before starting a transaction.

5.4.3.4.1 GetData (ATC) The ATC (Application Transaction Counter) was requested[34].

The ATC sent by the card does not change, regardless of the number of transactions performed. This ATC is different from the one returned by the first GenerateAC (which is incremented at each transaction), and is therefore clearly false.

The ATC is forged to manipulate the PoS risk management routine, which would otherwise request to go on-line.

The above also applied to the reading of the last online ATC[35].

5.4.3.4.2 Risk management Based on available data, the reader performs risk management as described in Chapter 10.6.3 of [EMV08c]:

> "*If the required data objects are available, the terminal shall compare the difference between the ATC and the Last Online ATC Register with the Lower Consecutive Offline Limit to see if the limit has been exceeded.*"

31. Command: `00 88 00 00 04`.
32. Command: `80 CA 9F 17 04`.
33. Command: `00 20 00 80 08`.
34. Command: `80 CA 9F 36 05`.
35. Command: `80 CA 9F 13 05`.

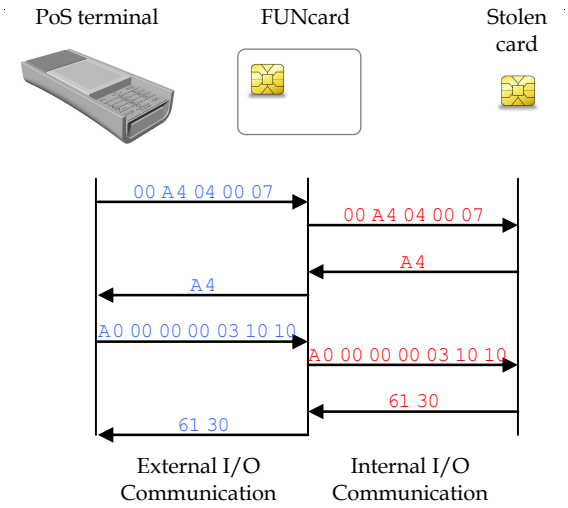


Figure 5.22 – The FUN card intercepts the Select command and replays it to the stolen card. Then the FUN card sends back the response to the PoS.

Here, ATC (`0x04`) – LOATC (`0x02`) > LCOL (`0x01`).

As the transaction log extracted from the card indicated, the fraudsters performed many small amount purchases to avoid on-line connection requests.

5.4.4 Side-Channel Power Analysis

Measuring variations in the device's power consumption enables detecting patterns that correspond to repeated operations. This is a common way to try and determine secret keys used in cryptographic operations. Although very rarely, side-channel analysis is also used by forensic experts (*e.g.* [SF13]).

Here, side-channel analysis will expose the fact that the forgery contains an underlying (legitimate) card, by analysing in detail the forgery's power trace when it is operated.

We shall contrast the "VerifyPIN" command, which does not propagate to the stolen card, with the "Select" command, which must be relayed to the stolen card.

5.4.5 EMV "Select" Command

The VISA application is selected based on its AID.

The sequence diagram of Figure 5.22 shows what should happen if the forgery indeed behaved as a "chip-in-the-middle" proxy.

Power consumption is measured and synchronized with the I/O between the card and the reader. However, internal communication between the FUN card and the stolen chip is witnessed on the power trace.

Figure 5.23 shows power consumption over time when the Select command is sent to the forgery. Patterns clearly appear during I/O activity. Some patterns can also be noticed *between* I/O operations, while there is no communication with the reader. A finer analysis shows that these patterns are made of sub-patterns, whose number is equal to the number of bytes exchanged. This confirms that communication is intercepted and re-transmitted by the FUN card, as illustrated in Figure 5.24.

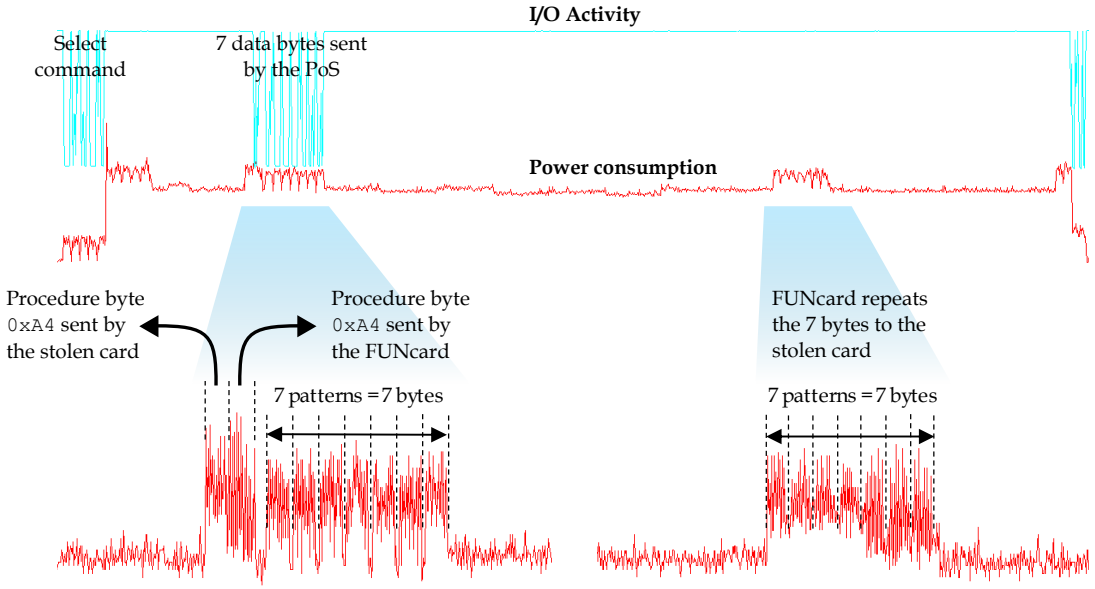


Figure 5.23 – The power trace analysis of the forgery during the Select command reveals a pattern that is repeated, despite the absence of I/O operations. It is readily observed that the pattern corresponds to the replay of data sent earlier by the PoS.

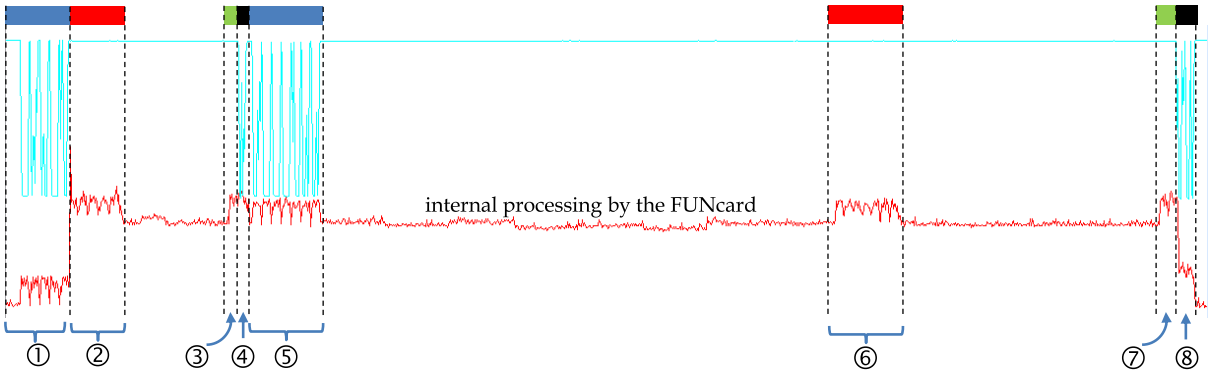


Figure 5.24 – ① PoS sends the ISO command `00 A4 04 00 07`; ② The command is echoed to the stolen card by the FUN card; ③ The stolen card sends the procedure byte `A4` to the FUN card; ④ The FUN card retransmits the procedure byte (`A4`) to the PoS; ⑤ The PoS sends the data `A0 00 00 00 03 10 10` to the FUN card; ⑥ The FUN card echoes `A0 00 00 00 03 10 10` to the stolen card; ⑦ The stolen card sends the status word (`SW1=61`, `SW2=30`) to the FUN card; ⑧ and the FUN card transmits `SW1 SW2` to the PoS. Communication: PoS → FUN card is shown in blue; FUN card → stolen card in red; Stolen card → FUN card in green and FUN card → PoS in black.

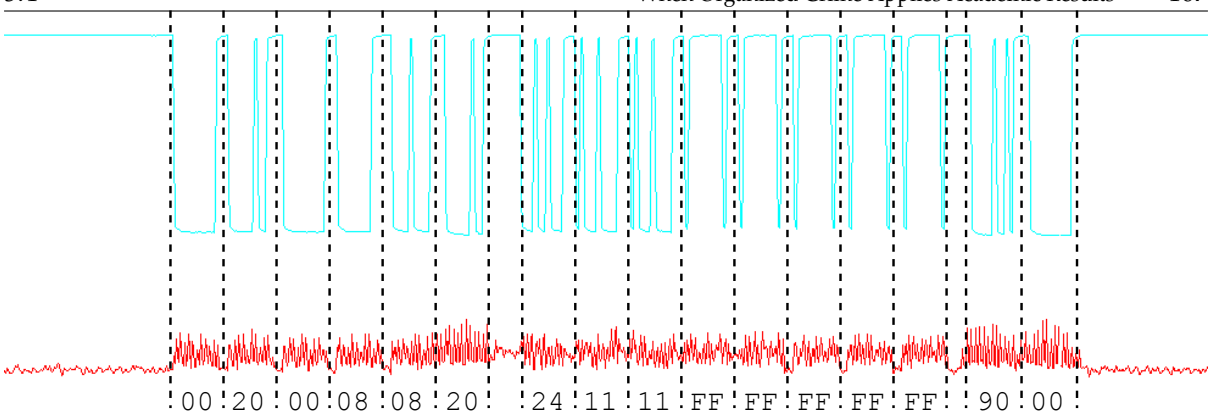


Figure 5.25 – Power trace of the forgery during the VerifyPIN command. Notice the absence of a re-transmission on the power trace before the returning of `SW1 SW2`.

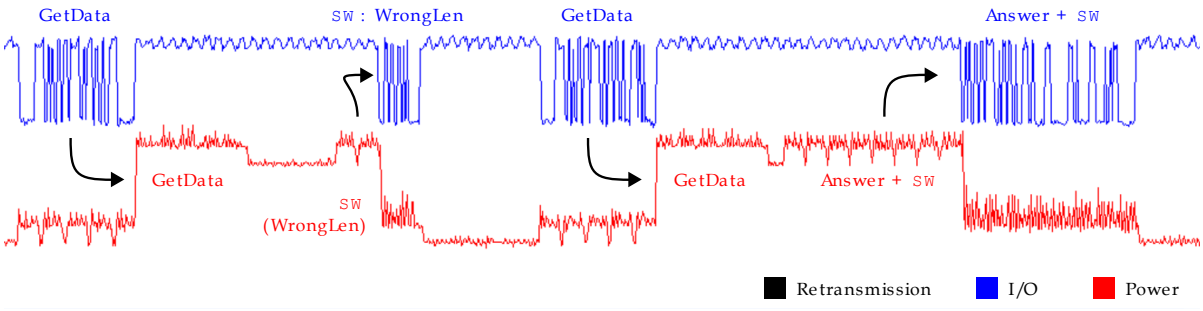


Figure 5.26 – Power consumption during a GetData command.

5.4.6 EMV "VerifyPIN" Command

We now turn our attention to the "VerifyPIN" command which, in the case of a proxy chip, would never be sent to the stolen chip.

As expected, this command is executed directly, as shown on the power trace of Figure 5.25. No operations between I/Os are witnessed here.

5.4.6.1 GetData commands

When sent a GetData command, the card seems to modify some values used for risk management purposes, so as to manipulate the PoS. The level of resolution offered by power trace analysis (Figure 5.26) is insufficient for seeing when this happens.

5.4.7 Destructive Analysis

We finally de-capsulated the forged module to analyze its internal structure. Results are shown in Figures 5.27, 5.28, and 5.29.

The `Vcc`, `RST`, `CLK`, `GND` contacts of the FUN card are connected to the corresponding pins of the stolen card (`Vcc` to `Vcc`, `RST` to `RST` etc.). However the stolen card's `IO` pin is connected to the `SCK` pin of the FUN card (Figure 5.30).

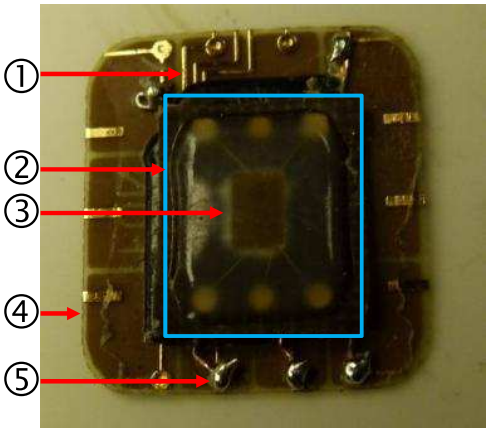


Figure 5.27 – ① Connection grid; ② Stolen card's module (outlined in blue); ③ Stolen card's chip; ④ FUN card module; ⑤ Weldings of connection wires.

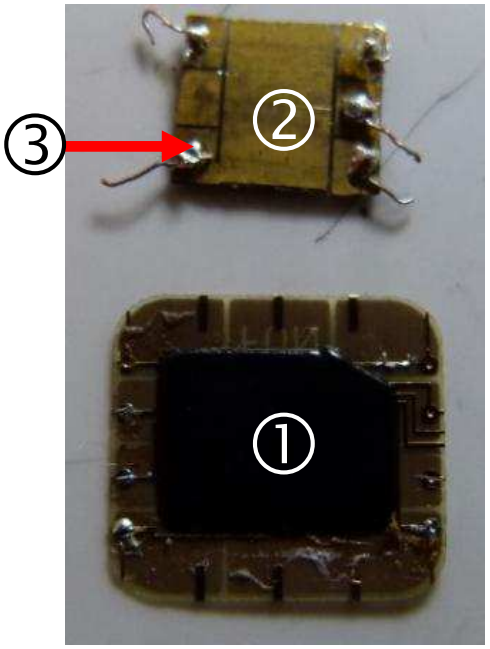


Figure 5.28 – ① FUN card module; ② genuine stolen card; ③ welded wire.

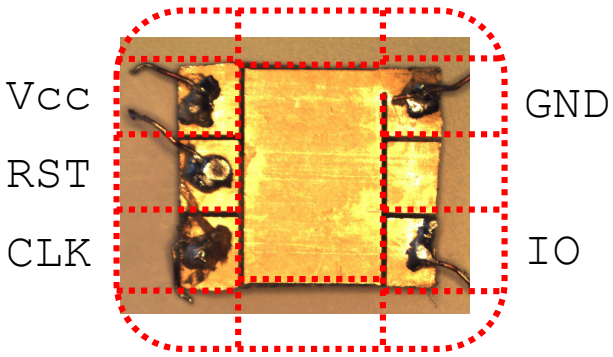


Figure 5.29 – Original EMV chip clipped by the fraudsters, with the cut-out pattern overlaid.

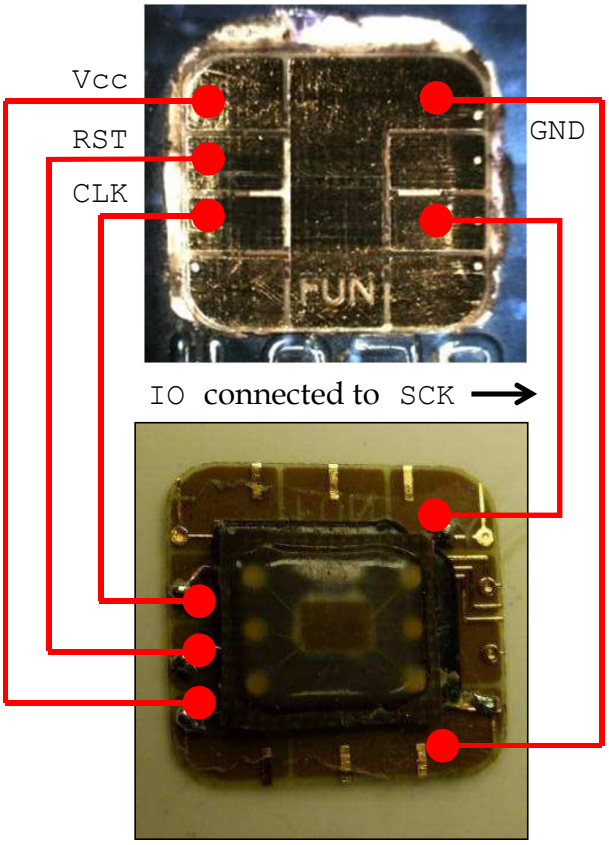


Figure 5.30 – Wiring diagram of the forgery.

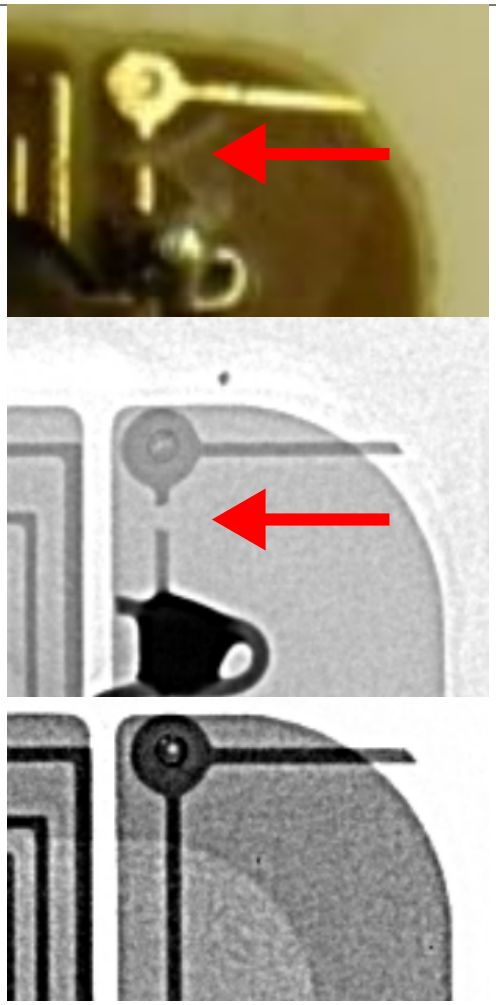


Figure 5.31 – Anti-forensics precautions taken by the perpetrator. Zoom on parts of Figures 5.30 (fraudulent card), 5.18 (X-ray of the fraudulent card), and 5.19 (unmodified FUN card). The abrasion and cut wire are clearly visible.

5.4.7.1 Anti-Forensic Countermeasures

Figure 5.31 shows that the perpetrators scratched the printed circuit copper track of SCK to conceal the traffic transiting *via* SCK. Recall that SCK is the most informative signal in the device because SCK is used by the FUN card to communicate with the stolen card.

During questioning by law enforcement, two reasons were advanced by the perpetrator for doing so. The first was, indeed, the intention to make forensic analysis harder. The second is way more subtle: using a software update, PoSs could be modified to spy the traffic on SCK. This would have allowed deploying a software countermeasure that would have easily detected forged cards.

5.4.8 Aftermath & Lessons Learned

The forensic report produced by the authors of this research work was sufficient for the court to condemn the perpetrators. During our testimony we underlined to the court that this case shows that organised crime is following very attentively advances in information security. We also noted that producing the forgery required patience, skill and craftsmanship. It is important to underline that, as we write these lines, the attack described in this research work is not applicable anymore, thanks to the activation of a

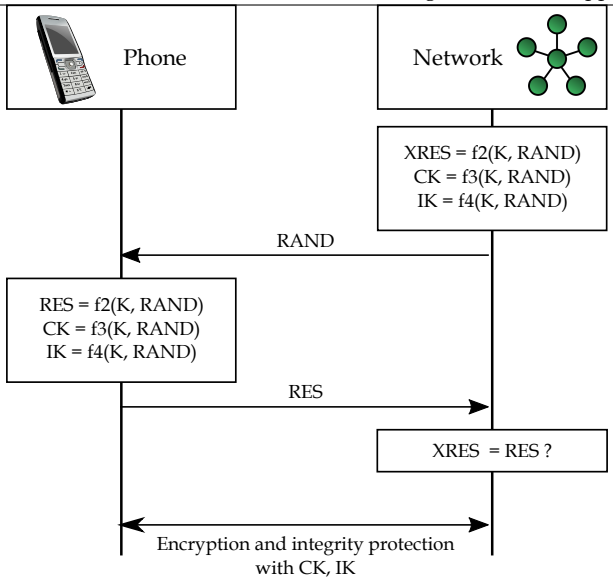


Figure 5.32 – 3G authentication protocol (simplified).

new authentication mode (CDA, Combined Data Authentication) and network level protections acting as a second line of defense. Until the deployment of CDA, this fraud was stopped using network-level counter-measures and PoS software updates. While we cannot detail the network-level countermeasures for confidentiality reasons [36], the following two fixes allowed to immediately stop the fraud:

5.4.8.0.1 Parity Faults We assumed that the fraudster only implemented the just-enough functionalities allowing to perform the fraud. This was indeed the case: when injecting byte-level parity errors into bytes transmitted from the PoS to the FUN card, the FUN card did not request byte re-transmission as mandated by the ISO/IEC 7816 standard. Coding, testing and deploying this countermeasure took less than a week.

5.4.8.0.2 Abnormal Applicative Behavior The forged card replies with a status word equal to `0x9000` to VerifyPIN commands sent *outside* of a transaction context (*e.g.* just after a reset). This is uncompliant with EMV specifications (where a PIN is necessarily attached to a previously selected application) and proves that the FUN card is not context-aware. Coding, testing and deploying this countermeasure was done overnight.

In addition, four other software-updatable countermeasures were developed and tested, but never deployed. These were left for future fraud control, if necessary.

Nonetheless, this case illustrates that, as a rule of thumb, an unmalleable cryptographic secure channel must always exist between cards and readers. Other (more expensive) solutions allowing to avoid man-in-the-middle devices consist in relying on physical attestation techniques such as [MMC06].

5.4.9 Other Applications of Miniature Spy Chips

The technique explained in this section can be generalized to attack non-payment devices.

5.4.9.1 Eavesdropping Mobile Communications

By extracting a chip from a FUN card and implanting it under the SIM connector of a smartphone, mobile communications can be monitored and decrypted. The demonstrator, on which we currently work, functions as follows: GSM and 3G communication confidentiality is based on session keys (denoted Kc, CK and IK) transmitted by the SIM to the phone. These session keys are derived from a random challenge (RAND) sent from the Authentication server (AuC) to the SIM (see Figure 5.32). A FUN card implanted under the reader can easily monitor these exchanges and record Kc, CK and IK in EEPROM.

While this happens, the opponent intercepts and records encrypted voice communications without decrypting them. It remains to extract the captured key material and transmit it to the attacker. This is far from being a trivial task given that, unlike the EMV fraud case that we have just analyzed, a FUN card implanted under a card reader does not actively control the SIM.

As strange as this may sound, as a matter of fact *it does*, assuming that the FUN card can read bits quicker than the phone (which is the case in practice). The ISO/IEC 7816 protocol relies on the fact that the `IO` signal connecting the card to the reader is pulled-up. This means that a party wishing to communicate pulls-down the `IO` and hence signals a zero to the other party. When the communicator's port is switched to high impedance, the line automatically goes up again. Hence, if we connect the FUN card's `IO` to the SIM connector's `IO`, both the FUN card and the legitimate SIM can signal zeros to the phone. In other words, the phone will see the information $b_f \wedge b_s$ where b_f and b_s (respectively) denote the bits sent by the FUN card and by the SIM.

To prove its identity to the network, the SIM returns to the AuC a response called SRES (or RES). Hence, the FUN card can intervene in the transmission of RES and force some of RES's bits to zero. Because RES is false the authentication will fail *but* information (in which the FUN card can embed Kc, CK or IK) will be broadcast to the attacker over the air. This precise information encoding problem was already considered by Rivest and Shamir in [RS82].

The implementation of this strategy is technical. It requires more than just turning bits to zero, because every byte sent from the SIM to the phone has a parity bit. Switching a single bit to zero means that the parity bit must also be flipped, which can only be done when the parity is one. Hence, the FUN card needs to compute the parity p of bits [0:6]. If $p = 0$ or bit 7 is zero, the FUN card remains quiet. Else, the FUN card pulls down the `IO` during bit 7 *and* during the parity. Another option consists in pulling down two data bits during transmission and leaving the parity unchanged.

5.4.9.2 Characterizing Unknown Readers

Consider the case of a border control device, produced and sold in small quantities, to carefully chosen clients. Users are given identification cards that interact with the device, but the description of the ISO commands exchanged between the card and the device is kept confidential. Exhausting all possible commands is impossible because critical-infrastructure cards usually embed a ratification counter that limits the number of unknown commands to 10 before definitively blocking the card.

An intelligence agency wishing to characterize the readers and understand how they work may construct a "chip-in-the-middle" command recorder based on a FUN card and a genuine identification card. The ISO command set could then be retrieved for later analysis.

5.4.9.3 Low-Cost Hardware Security Modules

In a number of industrial settings, keys, signatures or ciphertexts must be generated at a fast pace. A smart-card has relatively strong tamper resistance defenses but modest computational capabilities. Hardware Security Modules (HSMs) are expensive devices featuring both tamper-resistance and important computational capabilities.

36. These can potentially be efficient against yet unknown future forms of fraud.

Figure 5.33 – An industrial multi-SIM reader.

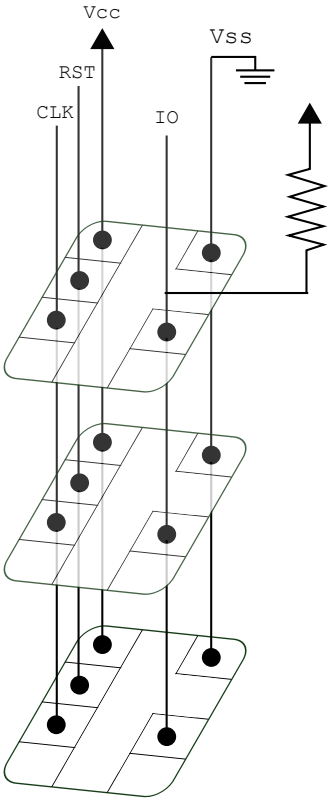


Figure 5.34 – Proposed low-cost HSM design based on SIM cards.

A number of manufacturers propose multi-smart-card readers (Figure 5.33). In such readers, a central processor establishes distinct one-to-one connections with each smart-card. An alternative HSM concept, illustrated in Figure 5.34, would consist in simply wiring card modules to each other. Power and clock supply would be common to all card modules. All card modules will be reset at once (given that `IO` is pulled up, the simultaneous emission of answers to reset will not cause signal conflicts). Communicating with individual modules will only require the (software) coding of a non-ISO protocol, where modules monitor `IO` and emit information to the reader while avoiding collisions.

CHAPTER 6

DESIGNING CONFIDENTIALITY BUILDING-BLOCKS

Summary

This chapter presents our research results in the area of *confidentiality*.

The research work presented in Section 6.1 [1] proposes a public-key cryptosystem and a short password encryption mode, where traditional hardness assumptions are replaced by specific refinements of the CAPTCHA concept called Decisional and Existential CAPTCHAs.

The public-key encryption method, achieving 128-bit security, typically requires from the sender to solve one CAPTCHA. The receiver does not need to resort to any human aid.

A second symmetric encryption method allows to encrypt messages using very short passwords shared between the sender and the receiver. Here, a simple 5-character alphanumeric password provides sufficient security for all practical purposes.

We conjecture that the automatic construction of Decisional and Existential CAPTCHAs is possible and provide candidate ideas for their implementation.

Honey Encryption (HE), introduced by Juels and Ristenpart (Eurocrypt 2014, [JR14]), is an encryption paradigm designed to produce ciphertexts yielding plausible-looking but bogus plaintexts upon decryption with wrong keys. Thus brute-force attackers need to use additional information to determine whether they indeed found the correct key.

At the end of their paper, Juels and Ristenpart leave as an open question the adaptation of honey encryption to natural language messages. A recent paper by Chatterjee *et al.* [CBJR15] takes a mild attempt at the challenge and constructs a natural language honey encryption scheme relying on simple models for passwords.

Section 6.2 [2] explains why this approach cannot be extended to reasonable-size human-written documents *e.g.* e-mails. We propose an alternative solution and evaluate its security.

Section 6.3 [3] generalizes the concept of Hierarchical Identity-Based Encryption (HIBE) by proposing a new primitive called Hierarchical Identity-Based Broadcast Encryption (HIBBE). Similar to HIBE, HIBBE organizes users in a tree-like structure and users can delegate their decryption capability to their subordinates, which mirrors real-world hierarchical social organizations. Unlike HIBE merely allowing a single decryption path, HIBBE enables encryption to any subset of the users and only the intended users (and their supervisors) can decrypt. We define Ciphertext Indistinguishability against Adaptively

1. Co-authored with Rémi Géraud and David Naccache.
2. Co-authored with Marc Beunardeau, Rémi Géraud and David Naccache.
3. Co-authored with Weiran Liu, Jianwei Liu, Qianhong Wu, Bo Qin and David Naccache.

Chosen-Identity-Vector-Set and Chosen-Ciphertext Attack (IND-CIVS-CCA2) which capture the most powerful attacks on HIBBE in the real world. We achieve this goal in the standard model in two steps. We first construct an efficient HIBBE Scheme (HIBBES) against Adaptively Chosen-Identity-Vector-Set and Chosen-Plaintext Attack (IND-CIVS-CPA) in which the attacker is not allowed to query the decryption oracle. Then we convert it into an IND-CIVS-CCA2 scheme at only a marginal cost, i.e., merely adding one on-the-fly dummy user at the first depth of hierarchy in the basic scheme without requiring any other cryptographic primitives. Furthermore, our CCA2-secure scheme natively allows public ciphertext validity test, which is a useful property when a CCA2-secure HIBBES is used to design advanced protocols.

The last research work of this chapter [4] illustrates how standard cryptographic techniques can be applied to real-life security products and services. Section 6.4 improves a little noticed yet ingenious Microsoft patent by Thomlinson and Walker. The Thomlinson-Walker system distributes encrypted patches to avoid reverse engineering by opponents (who would then be able to launch attacks on unpatched users). When the proportion of users who downloaded the encrypted patch becomes big enough, the decryption key is disclosed and all users install the patch.

4. Co-authored with Michel Abdalla, Hervé Chabanne, Julien Jainski and David Naccache.

6.1 Human Public-Key Encryption

6.1.1 Introduction

CAPTCHAs[5] [ABHL03] are problems that are hard to solve by computers, while being at the reach of most untrained humans. There might be many reasons why, at a particular time, a given type of CAPTCHA is considered hard for computers. The automated solving of CAPTCHAs may either require more computational power than is available, or algorithms have yet to be invented. It might well be that computers are inherently less efficient, or even incapable, at some tasks than human beings. Whichever the cause, several candidate CAPTCHAs are widely used throughout the Internet to keep robots at bay, or at least slow them down (*e.g.* [EDHS07; CGJ+08; Goo; CB03; NASK14; SHL+10]).

Most CAPTCHAs are used as human-interaction proofs [BL05] but their full potential as cryptographic primitives has not been leveraged so far despite a few exploratory papers. Early attempts [Dzi10; CHS06; ABHL03; CHS05] faced the inherent difficulty of *malleability*: given a CAPTCHA Q, an adversary could generate Q', whose solution gives a solution to Q. Thus the security of such constructions could only be evaluated against unrealistic "conservative adversaries" [KOPW13]. All in all, we propose to fill the gap by providing a finer taxonomy of CAPTCHAs as well as cryptosystems based on them, which can reach real-life security standards.

The organization of this section is as follows: Section 6.1.2 defines the classes of problems we are interested in, and estimates how many of those problems can be solved per time unit. We then refine the classical CAPTCHA concept into Decisional and Existential CAPTCHAs. Section 6.1.3 describes how to implement public-key encryption using Decisional CAPTCHAs; Section 6.1.4 describes a short password-based encryption mode that uses Existential CAPTCHAs to wrap high-entropy keys. Section 6.1.5 presents Decisional and Existential CAPTCHA candidates.

6.1.2 Preliminaries and Definitions

6.1.2.1 CAPTCHA Problems

Let $\mathcal{Q}$ be a class of problem instances, $\mathcal{A}$ a class of answers, and S a relation such that $S(Q, A)$ expresses the fact that "$A \in \mathcal{A}$ is a solution of $Q \in \mathcal{Q}$". Solving an instance Q of problem $\mathcal{Q}$ means exhibiting an $A \in \mathcal{A}$ such that $S(Q, A)$. We assume that for each problem there is one and only one solution, *i.e.* that S is bijective. This formal setting (similar to [KOPW13; CHS06]) allows us to provide more precise definitions.

Because CAPTCHAs involve humans and considerations about the state of technology, we do not pretend to provide formal mathematical definitions but rather clarifying definitional statements.

Definition 6.1 (Informal) *A given problem $\mathcal{Q} \in$ CP (CAPTCHA Problem) if no known algorithm can solve a generic instance $Q \in \mathcal{Q}$ with non-negligible advantage over $1/|\mathcal{A}|$, which is the probability to answer Q correctly at random; yet most humans can provide the solution A to a random $Q \in_R \mathcal{Q}$ with very high probability in reasonable time.*

In Definition 6.1, it is worth pointing out that future algorithms might turn out to solve efficiently some problems that evade today's computers' reach. As such, CP is not so much a complexity class as it is a statement about technology at any given point in time.

There exist today several approaches to building CAPTCHAs, based for instance on deformed word recognition, verbal tests, logic tests or image-based tasks. We are chiefly interested in those tests that can be automatically generated.

We extend CP in two ways:

5. "Completely Automated Public Turing test to Tell Computers and Humans Apart".

Definition 6.2 (Informal) *A given problem $\mathcal{Q} \in \mathsf{DCP}$ (Decisional CP) if $\mathcal{Q} \in \mathsf{CP}$ and, given a random instance $Q \in_R \mathcal{Q}$ and a purported solution A to Q, no known algorithm can decide whether A is a solution to Q,* i.e. *evaluate $S(Q, A)$, with non-negligible advantage over $1/|\mathcal{A}|$; while humans can determine with high probability $S(Q, A)$ in reasonable time.*

Finally, we introduce a further class of problems:

Definition 6.3 (Informal) *Let $\overline{\mathcal{Q}} \notin \mathsf{CP}$ be a set of "decoy data" which are not CAPTCHAs. A given problem $\mathcal{Q} \in \mathsf{ECP}$ (Existential CP) if $\mathcal{Q} \in \mathsf{CP}$ and, given a generic instance $Q \in \mathcal{Q}$ or a decoy $Q \in \overline{\mathcal{Q}}$, no known algorithm can decide whether $Q \in \mathcal{Q}$ with non-negligible advantage over $|\mathcal{Q}|/|\mathcal{Q} \cup \overline{\mathcal{Q}}|$; while humans can decide correctly if $Q \in \mathcal{Q}$ or $Q \in \overline{\mathcal{Q}}$ in reasonable time with high probability.*

Remark Definition 6.3 depends on the set $\overline{\mathcal{Q}}$. We silently assume that, for a given problem $\mathcal{Q}$, an appropriate $\overline{\mathcal{Q}}$ is chosen. This choice makes no difference.

When $\mathcal{Q}$ is not exhaustively searchable, Definition 6.3 means that a computer cannot decide whether a given Q is a CAPTCHA or not, let alone solve Q if Q is indeed a CAPTCHA.

Remark Definition 6.3 can be reformulated similarly to the IND-CPA [NY90] security game: we pick a random bit b and provide the adversary with Q_b, where $Q_0 \in \mathcal{Q}$ and $Q_1 \in \overline{\mathcal{Q}}$. The adversary is expected to guess b no better than at random unless it resorts to human aid.

Remark $\mathsf{ECP}, \mathsf{DCP} \subseteq \mathsf{CP}$, but there is no inclusion of ECP in DCP or *vice versa*. Informally, CP is about finding an answer, DCP is about checking an answer, and ECP is about recognizing a question.

Remark Solving a problem $Q \in \mathsf{CP}$ is either done using computers which by definition provide unreliable answers at best; or by asking a human to solve Q – effectively an oracle. However, there is a limit on the number of solutions humans can provide and on the rate at which humans can solve CAPTCHAs.

Consider a given $\mathcal{Q} \in \mathsf{CP}$ whose generic instances can be solved by a human in reasonable time. Let us estimate an upper bound b on the number of instances of $\mathcal{Q}$ that a human may solve during a lifetime. Assuming a solving rate of 10 instances per minute, and working age of 15–75 years, spent exclusively solving such problems, we get $b \sim 10^8$. Taking into account sleep and minimal life support activities, b can be brought down to $\sim 10^7$.

There should be no measurable difference between solving a problem in CP or in DCP, however it might be slightly simpler (and therefore quicker) for humans to *identify* whether a problem is a CAPTCHA without actually solving it. For simplicity we can assume that CAPTCHA recognition is ten times faster than CAPTCHA resolution.

There are various estimations on the cost of having humans solve CAPTCHAs. Some websites offer to solve 1000 CAPTCHAs for a dollar[6]. Of course, the oracle may employ more than one human, and be proportionally faster, but also proportionally more expensive.

6.1.3 Human Public-Key Encryption

We now describe a public-key cryptosystem using problems in DCP. Let $\mathcal{Q} \in \mathsf{DCP}$. We denote by $H(m)$ a hash function (*e.g.* SHA) and by $E_k(m)$ a block cipher (*e.g.* AES). Here, m is the plaintext sent by Bob to Alice.

— *Key-pair generation*: The public key pk is a list of b instances of $\mathcal{Q}$

$$\mathsf{pk} = \{Q_1, \ldots, Q_b\}$$

The private key is the set of solutions (in the CP sense) to the Q_i:

$$\mathsf{sk} = \{A_1, \ldots, A_b\}$$

i.e. for $1 \leq i \leq b$, $S(Q_i, A_i)$ holds true.

6. At a first glance, the previous figures imply that breaking a public-key (as defined in the next section) would only cost $\$10^4$. We make the economic nonlinearity conjecture there are no $\$10^4$ service suppliers allowing the scaling-up of this attack. In other words, if the solving demand d increases so will the price. We have no data allowing to quantify price(d).

— *Encryption*: Bob wants to send m to Alice. Bob picks k random problems $\{Q_{i_1}, \ldots, Q_{i_k}\}$ from Alice's pk, and solves them[7]. Let $\sigma \leftarrow \{A_{i_1}, \ldots, A_{i_k}\}$ and $\alpha \leftarrow \{i_1, \ldots, i_k\}$. Bob computes $\kappa \leftarrow H(\alpha)$ and $c \leftarrow E_\kappa(m)$, and sends (σ, c) to Alice.
— *Decryption*: Given σ, Alice identifies the set of indices α and computes $\kappa \leftarrow H(\alpha)$. Alice then uses κ to decrypt c and retrieve m. Decryption does *not* require any human help.

The general idea of this cryptosystem is somewhat similar to Merkle's puzzles [Mer78], however unlike Merkle's puzzle here security *is not quadratic*, thanks to problems in CP not being automatically solvable. We may assume that the A_is are pairwise different to simplify analysis.

Remark Indeed if $\mathcal{Q} \in$ CP it might be the case that a machine could decide if given A, Q the relation $S(A, Q)$ holds *without* solving Q. Hence $\mathcal{Q}$ must belong to DCP.

Remark A brute-force attacker will exhaust all $\binom{b}{k}$ possible values of α. Hence $\binom{b}{k}$ should be large enough. Given that $b \sim 10^7$ or $b \sim 10^8$, it appears that $k = 6$ provides at least 128-bit security.

Remark The main drawback of the proposed protocol is the size of pk. Assuming that each Q_i can be stored in 20 bytes, a pk corresponding to $b \sim 10^8$ would require 2 GB. However, given that CAPTCHAs are usually visual problems, it is reasonable to assume that pk might turn out to be compressible.

Remark Instead of sending back the solutions σ in clear, Bob could hash them individually. Hashing would only make sense as long as solutions have enough entropy to resist exhaustive search.

Remark It is possible to leverage the DCP nature of the Q_is in the following way: instead of sending a random permutation of solutions, Bob could interleave into the permutation d random values (decoy answers). Alice would spot the positions of these decoy answers and both Alice and Bob would generate $\alpha = \{i_1, \ldots, i_k, j_1, \ldots, j_d\}$ where j_d are the positions of decoys. Subsequently, security will grow to $\binom{b}{k+d}/d!$. This is particularly interesting since for $b = 10^7$, $k = 1$ and $d = 6$ we exceed 128-bit security. In other words, all the sender has to do is to *solve one CAPTCHA*.

Entropy can be further increased by allowing d to vary between two small bounds. In that case the precise (per session) value of d is unknown to the attacker.

6.1.4 Short Password-Based Encryption

In the following scenario Alice and Bob share a short password w. We will show how a message m can be securely sent from Alice to Bob using *only* w. This is particularly suited to mobile devices in which storing keys is risky.

Let $\mathcal{Q} \in$ ECP $\cap$ DCP.

— Alice generates a full size[8] key R and uses it to encrypt m, yielding $c_0 \leftarrow E_{0|R}(m)$. She generates an instance $Q \in \mathcal{Q}$, such that $S(P, R)$. Alice computes $c_1 \leftarrow E_{1|w}(P)$ and sends (c_0, c_1) to Bob.
— Bob uses w to decrypt c_1, and solves P. He thus gets the key R that decrypts c_0.

An adversary therefore faces the choice of either "attacking Shannon" or "attacking Turing", *i.e.* either automatically exhaust R, or humanly exhaust w. Each candidate w yields a corresponding P that cannot be computationally identified as a CAPTCHA. The adversary must hence resort to humans to deal with every possible candidate password.

Assuming that CAPTCHA identification by humans is ten times faster than CAPTCHA resolution, it appears that w can be a 5-character alphanumeric code[9].

Remark R must have enough entropy bits to provide an acceptable security level. R can be generated automatically on the user's behalf. As we write these lines we do now know if there exists $\mathcal{Q} \in$ ECP$\cap$DCP admitting 128-bits answers. If such $\mathcal{Q}$s do not exist, R could be assembled from several problem instances.

7. Here Bob must resort to human aid to solve $\{Q_{i_1}, \ldots, Q_{i_k}\}$.
8. *e.g.* 128-bit.
9. There are 64 alphanumeric characters, and $64^5 > 10 \times b$.

Remark In the above we assume that R is generated first, and then embedded into the solution of a problem instance P. All we require from R is to provide sufficient entropy for secure block cipher encryption. Hence, it might be easier to generate P first, and collect R afterwards.

Remark The main burden resting on Bob's shoulders might not be the solving on P but the keying of the answer R. 128 bits are encoded as 22 alphanumeric characters. Inputting R is hence approximately equivalent to the typing effort required to input a credit card information into e-commerce website interfaces [10]. Alternatively, Bob may as well read the solution R to a speech-to-text interface that would convert R into digital form.

Remark $Q \in \mathsf{ECP} \cap \mathsf{DCP}$ is necessary because the adversary may partially solve Q and continue using exhaustive search. Under such circumstances, c_0 serves as a clue helping the attacker to solve Q. If $Q \in \mathsf{ECP} \cap \mathsf{DCP}$, such a scenario is avoided.

6.1.5 DCP and ECP Candidate Instances

The above constructions assume that ECP and DCP instances exist and are easy to generate. Because ECP and DCP depend both on humans and on the status of technology, it is difficult to "prove" the feasibility of the proposed protocols.

We hence propose a DCP candidate an ECP candidates and submit them to public scrutiny.

6.1.5.1 DCP Candidate

Figure 6.1 – A DCP candidate constructed from an existing CP.

As a simple way to generate DCPs, we propose to start from a standard CP (*e.g.* a number recognition problem) and ask a further question about the answer. The further question should be such that its answer may correspond to numerous potential contents. For instance, the further question could be whether two sequences of digits recognized in an image Q sum up to $A = 91173$ or not (see Figure 6.1).

6.1.5.2 ECP Candidates

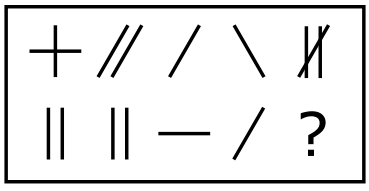


Figure 6.2 – An instance of a visual-logical task ECP problem. Recognizing objects in this image is insufficient to tell whether there is a solution, nor to compute the solution should there be one.

This section proposes a few candidate $\mathcal{Q}$ that we conjecture to belong to ECP.

The first step is to design a task that we think is challenging for computers. Despite recent progress (see *e.g.* [GBI+13]), computer vision is still expensive and limited. Most computer vision algorithms

10. PAN (16 characters), expiry date (4 characters) and a CVV (4 characters).

have to be trained specifically to recognize objects or features of a given kind (dog breeds, handwritten characters, etc.), and fail whenever the task at hand requires more than mere object identification. Even in that case, occlusion, distortion and noise cause drastic performance loss for most techniques. Many CAPTCHAs ideas rely on this to generate problem instances [CLSC05].

Even if image contents can be detected, we can still pose a hard challenge. Indeed, while computers excel at solving logical reasoning questions when those questions are encoded manually as logical formulae, state of the art algorithms fail at even the most basic questions when challenges are presented in visual form. Therefore, solving for instance a visual-logical task is a problem that is at least in DCP (see Figure 6.2).

Good ECP candidates for cryptographic purposes should be easy to generate, they should have enough possible solutions to thwart exhaustive search attempts, and it should be hard to tell automatically whether there is a solution at all.

6.1.5.3 **Temporal Sequence** ECP

Figure 6.3 – Three instances of the temporal sequence ECP problem. The problem consists in temporally arranging the pictures.

The intuition for this candidate is that although computer vision algorithms may reach human accuracy (and even beat it), humans can make use of external knowledge, which provides additional understanding of what is under scrutiny. Here the external knowledge is that real-life events abide by *causality*.

We provide k images (*e.g.* $k = 5$), each of which is a snapshot of some situation: buying goods, driving a car, dressing up, etc. The order of images is scrambled (some random images may be inserted as decoys) and the problem is to put images back in the correct order. This task, which we call *temporal sequence*, requires the contextual knowledge that some events can only happen after (or before) others. This is illustrated in Figure 6.3.

We conjecture that the temporal sequence task is both in DCP and in ECP.

One drawback of this approach is that to reach an 80-bit security level we need $k = 40$ images [11] which can be unwieldy. This may be solved by using ℓ collections of κ images, and tune ℓ, κ so that $(\kappa!)^\ell > 2^{80}$.

Temporal sequences may be automatically generated from videos, although it is not obvious how to ensure that sequences generated like this are always meaningful to humans.

Figure 6.4 – Visual Letter Recognition ECP: letters are concealed using an existing CP, and one digit is inserted into each sequence of letters. The ECP problem is to reorder the CAPTCHAs in increasing digit order, discarding all non-digit symbols. Here the solution consists in selecting the 4th, 5th, 2nd, 3rd, and 1st images, in that order.

6.1.5.4 **Visual Letter Recognition** ECP

Assume we have a CP problem $\mathcal{Q}$, whose instances can successfully conceal letters (a "one-letter" CAPTCHA). We provide k instances of $Q_1, \ldots, Q_k$ corresponding to answer letters $A_1, \ldots, A_k$, and ask for the alphabetically sorted list of these A_i.

As an example, we would generate instances of $\mathcal{Q}$ for the letters $\{A, M, T, O, B, R\}$, and ask for the solution ABMORT. Under the assumption that $\mathcal{Q} \in$ CP, determining whether a solution exists requires human aid. Therefore we conjecture that this problem belongs to ECP.

A further variant of this idea is illustrated in Figure 6.4.

Note that the visual letter recognition problem is DCP if an only if $\mathcal{Q} \in$ DCP.

6.1.5.5 **Honey Images** ECP

Another candidate problem is inspired by honey encryption [JR14; YKJ+15]. The idea is that any integer $1 \leq \ell \leq k$ would generate an image, but that only one value ℓ_{OK} generates a *meaningful* image [12]. All values $\ell \neq \ell_{\text{OK}}$ generate images in a way that makes them indistinguishable from meaningful images. The problem would then be to identify ℓ_{OK}, which we conjecture only humans can do reliably.

The main difficulty is that the notion of indistinguishability is tricky to define for images, and even harder to enforce: humans and computers alike use very specific visual cues to try and perform object recognition, which are hard to capture statistically. Following [YKJ+15], we may try and learn from a dataset how to properly encode images, but this is cumbersome in our context, especially when dealing with a large number of instances.

Our candidate is a simpler embodiment based on the following intuition: using biased noise (*i.e.* noise that is *not* random), we can elicit pareidolia in computer vision programs. Each candidate value of ℓ

11. There are $k!$ combinations, and $40! > 2^{80}$.
12. In the specific case of Figure 6.5, translation, rotation, mirroring as well as border cropping may also generate the meaningful image corresponding to ℓ_{OK}, but the overall proportion of such images remains negligible.

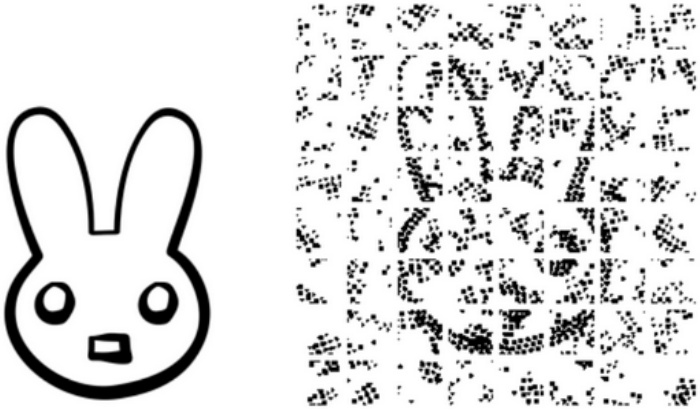


Figure 6.5 – A honey image ECP. Left: original image; right: $Q_{\ell_{\mathrm{OK}}}$, the transformed image for ℓ_{OK}.

Figure 6.6 – All values of ℓ other than ℓ_{OK} produce decoys whose statistical properties are conjectured to be indistinguishable from the correct image, with salient features but no real meaning.

would then correspond to some object being recognized – but only one of those is really relevant. We conjecture that only humans are able to pick this relevant object apart.

The authors implemented this idea. We start from a black and white picture of a clearly identifiable object (Figure 6.5 left, here $A =$ "rabbit"), turn it into a collection of black dots [13] (1). The picture is then cut into blocks which are shuffled and rotated (2). Finally, noise is added, under the form of black dots whose size is distributed as the size of black dots in the original picture (3). The image is then rotated back in place (Figure 6.5 right) to provide the challenge $Q_{\ell_{\mathrm{OK}}}$.

The motivation for this approach is as follows: (1) guarantees that individual pixels contain no information on luminescence, and geometric features (lines, gradients and corners) – each dot being circular destroys information about orientation; the shuffling and rotation of blocks in (2) is encoded as an integer ℓ; and (3) inserts decoy features, so that any shuffling/rotation would make geometric features appear (to lure a computer vision algorithm into detecting something).

Now, many decoys $Q_\ell \in \overline{\mathcal{Q}}, \ell \neq \ell_{\mathrm{OK}}$ can be generated easily from this image by shuffling and rotating blocks (Figure 6.6). Each decoy shares the same statistical properties as the correct (unshuffled) image, but has no recognizable content.

Our conjecture is that the human brain can perceive structures very efficiently and assign meaning to them. Many such structures are irrelevant and inserted so as to fool computer vision algorithms, but the familiar ones are immediately and intuitively grasped by humans. Consequently, although the original

13. For instance using an iteratively reweighted Voronoi diagram.

picture is severely deteriorated, we conjecture that it should still be possible for humans to tell noise and signal apart and identify correctly the contents of this image.

6.1.6 Further Applications

Figure 6.7 – Credit card PAN and expiry date, stored as a DCP instance.

Beyond their cryptographic interest, DCP and ECP tasks may have interesting applications in their own right.

One such application is the following: users may wish to store sensitive data as a DCP instance, for instance credit card information, instead of plaintext. Indeed, attackers often browse their victims' computers looking for credit card information, which is easy to recognize automatically. By storing credentials in an ECP the attacker's task can be made harder.

6.2 Honey Encryption for Language: Robbing Shannon to Pay Turing?

6.2.1 Introduction

Cryptography assumes that keys and passwords can be kept private. Should such secrets be revealed, any guarantee of confidentiality or authenticity would be lost. To that end, the set of possible secrets – the keyspace $\mathcal{K}$ – is designed to be very large, so that an adversary cannot possibly exhaust it during the system's lifetime.

In some applications however, the keyspace is purposely limited – for instance, passwords. In addition to the limited keyspace size, secret selection has a fundamental limitation: keys should be chosen uniformly at random – yet users routinely pick (the same) poor passwords. Consequently, key guessing is a guided process in which the adversary does not need to exhaust all possibilities. The deadly combination of low-entropy key generation and small keyspace make password-based encryption (PBE) particularly vulnerable [LHAS14].

The best security achievable by a PBE is measured by the min-entropy of the key distribution over $\mathcal{K}$:

$$\mu = -\log_2 \max_{k \in \mathcal{K}} p_k(k).$$

where p_k is the probability distribution of keys. The min-entropy captures how probable is the most probable guess. Conventional PBE schemes such as [Kal13] can be broken with constant effort with probability $O(2^{-\mu})$, but μ is in practice very small: [Bon12] reports $\mu < 7$ for passwords observed in a population of about 69 million users. If a message m were to be protected by such passwords, an adversary could easily recover m by trying the most probable passwords [14].

But how would the adversary *know* that the key she is trying is the correct one? A message has often some structure — documents, images, audio files for instance — and an attempt at decrypting with an incorrect key would produce something that, with high probability, does *not* feature or comply with this structure. The adversary can therefore tell apart a correct key from the incorrect ones, judging by how appropriate the decryption's output is. Mathematically, the adversary uses her ability to distinguish between the distribution of outputs for her candidate key k' and the distribution p_m of inputs she is expecting to recover.

Using such a distinguisher enables the attacker to try many keys, then select only the best key candidates. If there are not many possible candidates, the adversary can recover the plaintext (and possibly the key as well). In the typical case of password vaults, when one « master password » is used to encrypt a list of passwords, such an attack leads to a complete security collapse.

Example 6.1 *Assume that we wish to AES-decrypt what we know is an English word protected with a small 4 digits key:* $c \leftarrow \mathsf{Enc}_k(m)$. *An efficient distinguisher is whether* $m_{k'} \leftarrow \mathsf{Dec}_{k'}(c)$ *is made of letters belonging to the English alphabet. For instance, if*

$$c = \texttt{0f 89 7d 66 8b 4c 27 d7 50 fa 99 0c 5a d6 11 eb}$$

Then the adversary can distinguish between two candidate keys `5171` *and* `1431`*:*

$$m_{5171} = \texttt{48 6f 6e 65 79 00 00 00 00 00 00 00 00 00 00 00}$$
$$m_{1431} = \texttt{bd 94 11 05 a2 e5 a7 c8 48 57 87 2a 88 52 bc 7e}$$

Indeed, m_{5171} *spells out '*`Honey`*' in ASCII while* m_{1431} *has many characters that do not correspond to any letters. Exhausting all 4 digit keys yields only one message completely made of letters, hence* $k =$ `5171` *and the adversary succeeded in recovering the plaintext* m_{5171}.

To thwart such attacks, Juels and Ristenpart introduced Honey Encryption (HE) [JR14]. HE is an encryption paradigm designed to produce ciphertexts which, upon decryption with wrong keys, yields

14. Such passwords may be learned from password leaks [VCT14; WAdG09; JD12].

plausible-looking plaintexts. Thus brute-force attackers need to use additional information to decide whether they indeed found the correct key.

Mathematically, the decoding procedure in HE outputs candidate plaintexts distributed according to a distribution p_d close to the distribution p_m of real messages. This renders distinguishing attacks inoperant. The advantages of HE are discussed at length in [JR14] where the concept is applied to password-based encryption of RSA secret keys, credit card PINs and CVVs. In particular, HE does not reduce the security level of the underlying encryption scheme, but may act as an additional protection layer.

However, the applications of HE highlighted in [JR14] are very specific: Passwords protecting passwords (or passwords protecting keys). More precisely, low min-entropy keys protecting high min-entropy keys. The authors are wary not to extend HE to other settings and note that designing HE

> « *...for human-generated messages (password vaults, e-mail, etc.) (...) is interesting as a natural language processing problem.* » [JR14]

To give a taste of the challenge, realizing HE as Juels and Ristenpart defined it is equivalent to modeling the probability distribution of human language *itself*. A more modest goal is to restrict to subsets of human activity where choices are more limited, such as passwords — this is indeed the target of a recent paper by Chatterjee, Bonneau, Juels and Ristenpart [CBJR15], which introduces encoders for human-generated passwords they call « natural language encoders » (NLE). Chatterjee *et al.*'s approach to language is to model the distribution of messages using either a 4-gram generative Markov model or a custom-trained probabilistic grammar model. This works reasonably well for passwords.

A natural question is therefore: Could the same techniques be extended or generalized to human-generated documents *in general*? Chatterjee *et al.* hint at it several times, but never actually take a leap: The core reason is that these approaches do not scale well and fail to model even simple sentences – let alone entire documents.

6.2.2 Preliminaries

6.2.2.1 Notations

We write $x \xleftarrow{D} X$ to denote the sampling of x from X according to a distribution D, and $x \xleftarrow{\$} X$ when D is the uniform distribution.

6.2.2.2 Message Recovery Attacks

Let $\mathcal{M}$ be a message space and let $\mathcal{K}$ be a key space. We denote by p_m the message distribution over $\mathcal{M}$, and by p_k the key distribution over $\mathcal{K}$. Let Enc be any encryption scheme. The *message-recovery advantage* of an adversary $\mathcal{A}$ against Enc is defined as

$$\mathbf{Adv}^{\mathrm{MR}}_{\mathsf{Enc},p_m,p_k}(\mathcal{A}) = \Pr\left[\mathrm{MR}^{\mathcal{A}}_{\mathsf{Enc},p_m,p_k} = \mathsf{True}\right]$$

where the MR security game is described below. $\mathcal{A}$ may run for an unbounded amount of time, and make an unbounded number of queries to a random oracle.

This advantage captures the ability of an adversary knowing the distributions p_m, p_k to recover a message encrypted with Enc.

When key and message entropy are low, this advantage might not be negligible. However, using Honey Encryption, Juels and Ristempart show that $\mathcal{A}$'s advantage is bounded by $2^{-\mu}$, where $\mu = -\log \max_{k \in \mathcal{K}} p_k(k)$ is the min-entropy of the key distribution.

Game 1 Message recovery (MR) security game $\mathrm{MR}^{\mathcal{A}}_{\mathsf{Enc},p_m,p_k}$.

$K' \xleftarrow{p_k} \mathcal{K}$
$M' \xleftarrow{p_m} \mathcal{M}$
$C' \xleftarrow{\$} \mathsf{Enc}(K', M')$
$M \leftarrow \mathcal{A}(C')$
return $M == M'$

Figure 6.8 – $\mathsf{SAMP0}^{\mathcal{B}}_{\mathsf{DTE}}$

$x' \xleftarrow{\$} [0,1]$
$M' \leftarrow \mathsf{DTDecode}(x')$
$b \xleftarrow{\$} \mathcal{B}(M', x')$
return b

Figure 6.9 – $\mathsf{SAMP1}^{\mathcal{B}}_{\mathsf{DTE},p_m}$

$M' \xleftarrow{p_m} \mathcal{M}$
$x' \xleftarrow{\$} \mathsf{DTEncode}(M')$
$b \xleftarrow{\$} \mathcal{B}(M', x')$
return b

6.2.2.3 Distribution Transforming Encoding

Honey Encryption (HE), introduced by Juels and Ristenpart [JR14], is an encryption paradigm designed to produce ciphertexts yielding plausible-looking but bogus plaintexts called *honey messages* upon decryption with wrong keys. Thus brute-force attackers need to use additional information to determine whether they indeed found the correct key. The advantages of HE are discussed at length in [JR14] where the process is applied to password-based encryption of RSA secret keys and credit card numbers.

HE relies on a primitive called the *distribution transforming encoding* (DTE). The DTE is really the central object of HE, which is then used to encrypt or decrypt messages. A DTE is composed of two algorithms, DTEncode and DTDecode which map messages into numbers in the interval $[0,1]$ and back, *i.e.* such that

$$\forall M \in \mathcal{M}, \quad \mathsf{DTDecode}(\mathsf{DTEncode}(M)) = M.$$

More precisely, $\mathsf{DTEncode}\colon \mathcal{M} \to [0,1]$ is designed such that the output distribution of DTEncode is uniform over $[0,1]$ when the input distribution over $\mathcal{M}$ is specified and known — in other terms, DTDecode samples messages in $\mathcal{M}$ according to a distribution p_d close to p_m, with

$$p_d(M) = \Pr\left[M' = M \mid x \xleftarrow{\$} [0,1] \text{ and } M' \leftarrow \mathsf{DTDecode}(S)\right]$$

As such, DTEs cannot be arbitrary: They need to mimic the behavior of the cumulative distribution function and its inverse. More precisely, the closeness of p_d and p_m is determined by the advantage of an adversary $\mathcal{A}$ in distinguishing the games of Figures 6.8 and 6.9:

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{DTE},p_m} = \left|\Pr\left[\mathsf{SAMP1}^{\mathcal{A}}_{\mathsf{DTE},p_m} = 1\right] - \Pr\left[\mathsf{SAMP0}^{\mathcal{A}}_{\mathsf{DTE}} = 0\right]\right|$$

$\mathcal{A}$ is provided with either a real message and its encoding, or a fake encoding and its decoding. $\mathcal{A}$ outputs 1 or 0 depending on whether it bets on the former or the latter. A perfectly secure DTE is a scheme for which the indistinguishability advantage is zero even for unbounded adversaries (this is equivalent to $p_d = p_m$).

Having good DTEs is the central aspect of building a Honey Encryption scheme as well as the main technical challenge. Given a good DTE, the honey encryption and decryption of messages is provided by a variation of the "DTE-then-encrypt" construction described in Figures 6.10 and 6.11 where some symmetric encryption scheme $(\mathsf{ESEncode}, \mathsf{ESDecode})$ is used. In the "DTE-then-encrypt" paradigm, a message is first transformed by the DTE into an integer x in some range, and x (or rather, some binary representation of x) is then encrypted with the key. Decryption proceeds by decrypting with the key, then reversing the DTE.

Figure 6.10 – Algorithm HEnc^{ES} Figure 6.11 – Algorithm HDec^{ES}

$\mathsf{HEnc}^{ES}(K, M)$
$x \leftarrow \mathsf{DTEncode}(M)$
$C \leftarrow \mathsf{ESEncode}(x, K)$
return C

$\mathsf{HDec}^{H}(K, C)$
$x \leftarrow \mathsf{ESDecode}(K, C)$
$M \leftarrow \mathsf{DTDecode}(x)$
return M

6.2.3 Natural Language Encoding

Chaterjee *et al.* [CBJR15] developed an approach to generating DTEs based on two natural language models: an n-gram Markov model, and a custom probabilistic grammar tree.

6.2.3.1 Markov Model

The n-gram model is a local description of letters whereby the probability of the next letter is determined by the $n-1$ last letters:

$$\Pr[w_1 \cdots w_k] = \prod_{i=1}^{k} \Pr\left[w_i \mid w_{i-(n-1)} \cdots w_{i-1}\right]$$

It is assumed that these probabilities have been learnt from a large, consistent corpus.

Such models are language-independent, yet produce strings that mimic the local correlations of a training corpus — but, as Chomsky pointed out [Cho02; Cho56; Cho59], the output of such models lack the long-range correlations typical of natural language. The latter is not an issue though, as Chatterjee *et al.* train this model on passwords.

The model can be understood as a directed graph where vertices are labeled with n-grams, and edges are labeled with the cumulative probability from some distinguished root node. To encode a string it suffices to encode the corresponding path through this graph from the root — and decoding uses the input as random choices in the walk. Encoding and decoding can be achieved in time linear in message size.

6.2.3.2 Grammar Model

Probabilistic context-free grammars (PCFG) are language-dependent models that learn from a tagged corpus a set of grammatical rules, and then uses these rules to generate syntactically possible sentences. PCFGs are a compact way of representing a distribution of strings in a language.

Although it is known that context-free grammars do not capture the whole breadth of natural language, PCFGs are a good starting point, for such grammars are easy to understand, and from a given probabilistic context-free grammar, one can construct compact and efficient parsers [KM03]. The Stanford Statistical Parser, for instance, has been used by the authors to generate parse trees in this section.

Mathematically, a probabilistic context-free grammar G is a tuple of the form $(N, T, R, P, \textsc{Root})$ where N are non-terminal symbols, T are terminal symbols (disjoint from N), R are production rules, P is the set of probabilities on production rules and $\textsc{Root}$ is the start symbol. Every production rule is of the form $A \to b$, where $A \in N$ and $b \in (T \cup N)^*$.

Figure 6.12 shows a parse tree aligned with a sentence. Some grammatical rules can be read at every branching: S $\to$ NP VP, NP $\to$ DT VBN NN, NP $\to$ DT NN, *etc.*

Chaterjee *et al.* [CBJR15] rely on a password-specific PCFGs [WAdG09; JD12; VCT14; MYLL14; KKM+12] where grammatical roles are replaced by *ad hoc* roles.

The DTE encoding of a string is the sequence of probabilities defining a parse tree that is uniformly selected from all parse trees generating to the same string (see *e.g.* Figure 6.13, which provides an

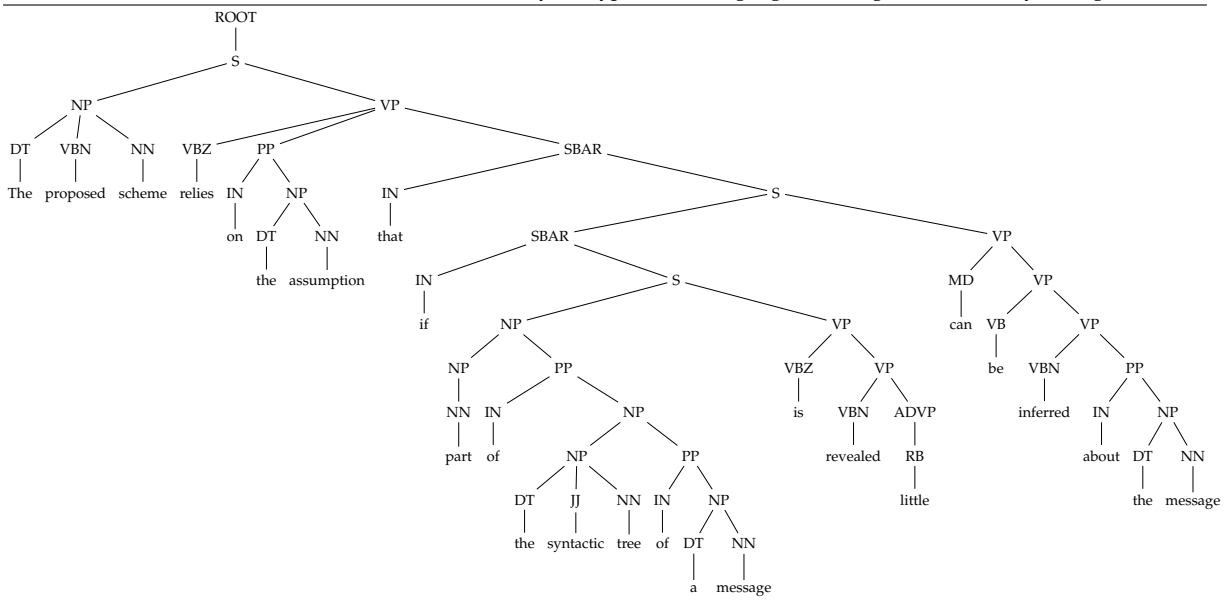


Figure 6.12 – Syntactic tree of an example sentence.

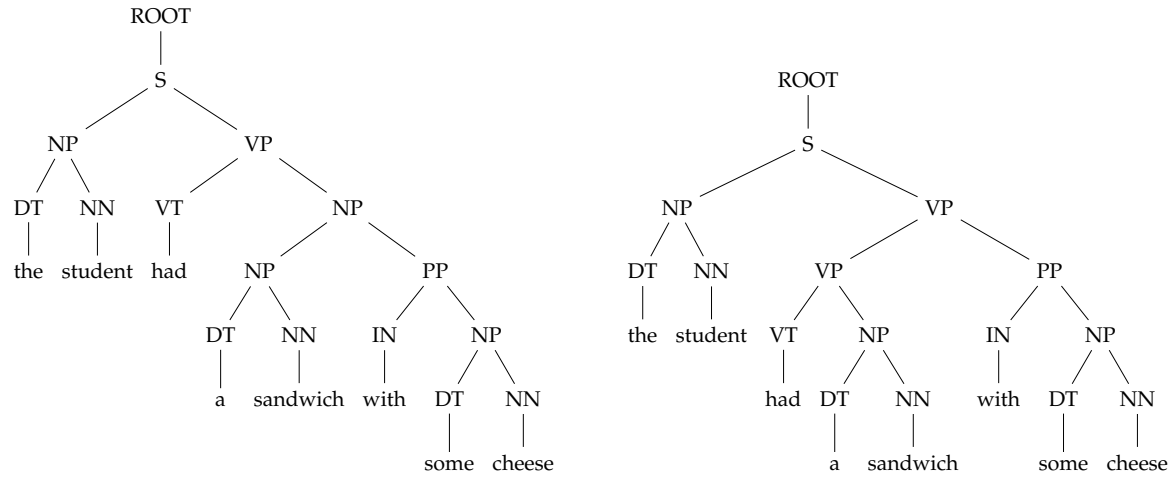


Figure 6.13 – Two possible derivations of the same sentence. Note that these derivations correspond to two possible meanings which are not identical.

example of two parse trees for a same sentence, amongst more than 10 other possibilities). Decoding just emits the string indicated by the encoded parse tree.

In the probabilistic context, the probability of each parse tree can be estimated. A standard algorithm for doing so is due to Cocke, Younger, and Kasami (CYK) [Coc69; You67; Kas65].

6.2.3.3 Generalized Grammar Model

The generalized idea relies on the assumption that if part of the syntactic tree of a message is revealed, little can be inferred about the message. To understand the intuition, consider the syntactic tree of the previous sentence (clause) shown in Figure 6.14.

As we can see, words are tagged using the *clause level*, *phrase level* and *word level* labels listed in Section 6.2.7.

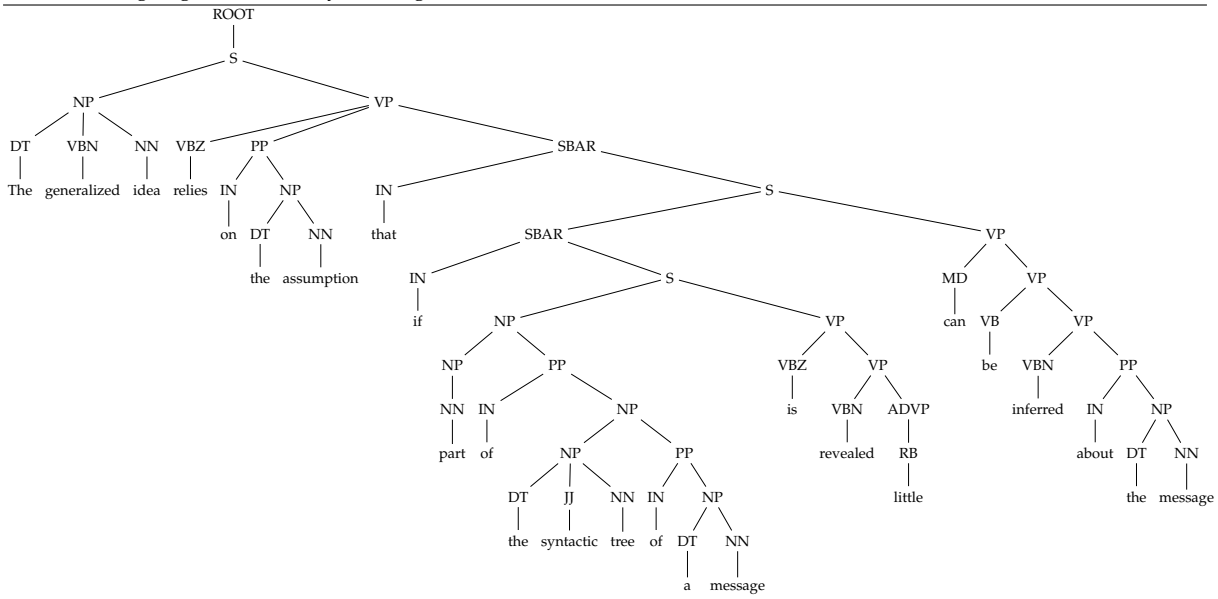


Figure 6.14 – Syntactic tree of an example sentence.

The idea underlying syntactic honey encryption consists in revealing a rewritten syntactic tree's word layer while encrypting words [15]. The process starts by a syntactic analysis of the message allowing to extract the plaintext's syntactic tree. This is followed by a projection at the word level. When applied to the previous example, we get the following projection (hereafter called *skeleton*).

Given a clause, we can automatically associate each word s_i to a label L_i [16]. For instance, if the third word of the clause is "relies", then $L_3 \leftarrow \text{VBZ}$. We denote by R_i the rank of the skeleton's i-th word in the dictionary of the category L_i. Finally we denote by $|X|$ the cardinality of the set X.

To map our ordered wordlist into a single integer, we note that because in the above example there are 5 DTs, 3 VBNs, 6 NNs, 2 VBZs, 6 INs, and 1 JJ, RB, MD and 1 VB, our specific clause is one amongst exactly B syntactically correct messages where:

$$B = |\text{DT}|^5|\text{VBN}|^3|\text{NN}|^6|\text{VBZ}|^2|\text{IN}|^6|\text{JJ}||\text{RB}||\text{MD}||\text{VB}|$$

We can thus map a clause skeleton into $\mathbb{N}$ by writing:

$$e \leftarrow \sum_{i=0}^{k-1} R_i \prod_{j=0}^{i-1} |L_j|$$

where, by typographic convention, $L_{-1} = 1$.

To get back the original clause, given e and the skeleton, we use the algorithm of Figure 6.15.

The skeleton is transferred in clear:

$$\begin{aligned} s = \ & \text{DT VBN NN VBZ IN DT NN IN IN NN IN DT JJ NN} \\ & \text{IN DT NN VBZ VBN RB MD VB VBN IN DT NN} \end{aligned}$$

Note that there is no need to tune precisely the plaintext size of the underlying block cipher because the decoding process for e stops automatically when i reaches $k-1$. In other words, we can randomize encryption at little cost by replacing e by $e + \mu B$ for some random integer μ.

15. We stress that unlike e.g. Kamouflage[BBBB10] which deals with passwords, syntacting honey encyrption applies to natural language.

16. Note that such a skeleton might be ambiguous in certain constructions, for instance in sentences such as "*Time flies like an arrow; fruit flies like a banana*".

Decoding

$\ell \leftarrow |L_0|$
for $i \leftarrow 0$ to $k-1$
  $R_i \leftarrow e \bmod \ell$
  $e \leftarrow (e - R_i)/\ell$
  $\ell \leftarrow \ell \times |L_{i+1}|$
  $\text{word}_i \leftarrow \text{Dictionary}_{L_i}(R_i)$

Figure 6.15 – Decoding algorithm.

The number e is then honey encrypted, thus attempting to protect the actual content of the plaintext sentence.

6.2.4 Limitations of Honey Encryption

As observed by [JR14], HE security does not hold when $\mathcal{A}$ has some side information about the target message. This puts strong constraints on HE's applicability to situations such as protecting RSA or HTTPS private keys. A second limitation is that the HE construction assumes that the key and message distributions are independent. When these distributions are correlated, $\mathcal{A}$ can identify a correct message by comparing that message with the decryption key that produced it. Similarly, encrypting two correlated messages under the same key enables $\mathcal{A}$ to identify correct messages.

Finally, constructing a DTE requires knowing the distribution p_m of messages in $\mathcal{M}$. As we will argue, this turns out to be extremely difficult to evaluate when $\mathcal{M}$ becomes a large enough space, such as human-generated messages (emails, *etc.*). In those cases, it might even turn out that adversaries know p_m *better than users*.

The methods described in Section 6.2.3 apply reasonably well to short passwords, but as we will now argue they cannot scale to deal with natural language as used in real-world scenarios such as: e-mails and written documents. The reason is threefold: First the methods require a huge amount of context-relevant information; Second, even when this information is available, the methods of [CBJR15] fail to produce convincing honey messages, *i.e.* messages that fool *automated tools* in telling them apart from real messages with high probability; Third, natural language HE may actually leak information about the underlying message.

6.2.4.1 Scaling NLE

The models developed for passwords in [CBJR15] can be extended: Markov models for instance can be configured to generate arbitrary-length messages. Instead of letters, such models can be trained to produce words, in accordance with some known distribution of n-grams. But while there are only a few English letters, a recent study of the English language [MSA+11] counts more than a million individual words in usage.

As a result, the memory required to store an n-gram database is of the order of $10^{6n} \approx 2^{20n}$. That becomes prohibitive not only in terms of storage, but also when access latency is taken into account. Applying directly the method of [CBJR15] to words (using $n = 5$) would require knowing, storing, and sharing 2^{100} bytes of data [17]. The real issue however is that measuring accurately 5-grams usage is extremely difficult in practice, so that most of this impossibly large database is essentially unknown [18].

Using grammars is one way to avoid this combinatorial explosion by keeping a simple and compact model of language. To that end, a sentence is parsed to reveal its grammatical structure as in Figures 6.12 and 6.13. Each word is labeled with an indication of its grammatical role (see Section 6.2.7).

17. This is conceptually similar to Borges' famous library [Bor41; Bor44].
18. See for instance http://www.ngrams.info/.

A sentence is therefore uniquely represented by a list of grammatical tags, and a list of integers denoting which word is used. The idea behind syntactic honey encryption consists in revealing the tags but honey encrypting the words. By construction, honey messages generated have the same syntax as the original message, which makes decryption with a wrong key yield an often *plausible* plaintext. For instance, a sentence such as s_1 = « Secure honey encryption is hard » could be honey decrypted as Chomsky's famous sentence s_2 =« Colorless green ideas sleep furiously » [Cho56], illustrating a sentence that is grammatically correct while being semantically void. Here s_1 and s_2 share the same syntax. To use this algorithm the communicating parties must agree on a dictionary that includes a set of labels and a parsing algorithm.

There are however two structural limitations to this grammatical approach. First, revealing the syntactic structure of a message leaks information: Unless the message is long enough, there might be only very few possible sentences with that given syntax. Second, a grammar is language-dependent — and furthermore, to some extent, there is variability within a given language [19]. The consequence of an inaccurate or incorrect tagging is that upon honey decoding, the sentence might be noticeably incorrect from the suitable linguistic standpoint.

This opens yet another research avenue. Automatically translate the sentence into an artificially created language where syntactic honey encryption would be very efficient. For instance translate French to Hindi, then perform honey encryption on the Hindi sentence.

6.2.4.2 Quality of NLE

The question of whether a honey message is "correct" in a given linguistic context can be rephrased: Is it possible, to an adversary having access to a large corpus (written in the same language), to distinguish honey messages from the legitimate plaintext?

It turns out that the two approaches to modeling natural language provide two ways to construct a distinguisher: We can compare a candidate to a reference, either statistically or syntactically. But we can actually do both *simultaneously*: We can use Web search engines to assess how often a given sentence or word is used [20]. This empirical measure of probability is interesting in two respects: First, an adversary may query many candidates and prune those that score badly; Second, the sender cannot learn enough about the distribution of *all* messages using that "oracle" to perform honey encryption.

The situation is that there is a measurable distance between the model (used by the sender) of language, and language itself (as can be measured by *e.g.* a search engine). Mathematically, the sender assumes an approximate distribution p_m on messages which is different from the real-world distribution $\widehat{p}_m$. Because of that, a good DTE in the sense of Figures 6.8 and 6.9 would, in essence, yield honey messages that follow p_m and not $\widehat{p}_m$. An adversary capable of distinguishing between these distributions can effectively tell honey messages apart.

What is the discrepancy between p_m and $\widehat{p}_m$? Since $\widehat{p}_m$ measures real-world usage, we can make the hypothesis that such messages correspond to human concerns, *i.e.* that they carry some meaning — in one word, what distinguishes p_m from $\widehat{p}_m$ is *semantics*.

6.2.4.3 Leaking Information

Another inherent limitation of HE is precisely that decryption of uniformly random ciphertexts produces in general the most probable messages. There are many situations in which linguistic constraints force a certain structure on messages, *e.g.* the position of a verb in a German sentence. As a consequence, there might be enough landmarks for a meaningful reconstruction (see also [RWJL06]).

19. An extreme example is William Shakespeare's use of inversion as a poetic device: "*If't be so, For Banquo's issue have I fil'd my mind,/ For them the gracious Duncan have I murther'd,/Put rancors in the vessel of my peace*" (*MacBeth*, III.1.8).

20. We may assume that communication with such services is secure, *i.e.* confidential and non-malleable, for the sake of argument.

To thwart such reconstruction attacks, it is possible to consider phrase-level defenses. Such defenses imply modifying the syntactic tree in a way which is both reversible and indistinguishable from other sentences of the language. Phrase-level defenses heavily depend on the language used. For instance the grammar of Latin, like that of other ancient Indo-European languages, is highly inflected; consequently, it allows for a large degree of flexibility in choosing word order. For example, *femina togam texuit*, is strictly equivalent to *texuit togam femina* or *togam texuit femina*. In each word the desinence (also called ending or suffix): *-a*, *-am* and *-uit*, and not the position in the sentence, marks the word's grammatical function. This specific example shows that even if the target language allows flexibility in word order, this flexibility does not necessarily imply additional security. Semitic languages, such as Arabic or Hebrew, would on the contrary offer very interesting phrase-level defenses. In semitic languages, words are formed by associating three-consonant verbs to structures. In Hebrew for example the structure *mi□□a□a* corresponds to the place where action takes place. Because the verb *drš* means *to teach* (or *preach*), and because the verb *zrk* means *to throw* (or *project*), the words *midraša*[21] and *mizraka* respectively mean "school" and "water fountain" (the place that projects (water)). This structure which allows, in theory, to build $O(ab)$ terms using $O(a)$ verbs and $O(b)$ and thus turns out to be HE-friendly.

6.2.5 Corpus Quotation DTE

We now describe an alternative approach which is interesting in its own right. Instead of targeting the whole breadth of human language, we restrict users to only quote from a known public document[22].

The underlying intuition is that, since models fail to capture with enough detail the empirical properties of language, we should think the other way around and start from an empirical source directly. As such, the corpus quotation DTE addresses the three main limitations of HE highlighted in Section 6.2.4: It scales, it produces realistic sentences (because they are actual sentences), and it does not leak structural information.

Consider a known public string $\mathfrak{M}$ (the "corpus"). We assume that $\mathcal{M}$ consists in contiguous sequence of words sampled from $\mathfrak{M}$, *i.e.* from the set of substrings of $\mathfrak{M}$. To build a DTE we consider the problem of mapping a substring $m \in \mathcal{M}$ to $[0, 1]$.

6.2.5.1 Interval Encoding of Substrings

Let M be the size of $\mathfrak{M}$, there are $|\mathcal{M}| = M(M-1)/2$ substrings denoted $m_{i,j}$, where i is the starting position and j is the ending position, with $i \leq j$. Substrings of the form $m_{i,i}$ are 1-letter long.

The DTE encoding of $m \in \mathcal{M}$ is a point in a sub-interval of $[0, 1]$, whose length is proportional to the probability $p_m(m)$ of choosing m. If p_m is uniform over $\mathcal{M}$, then all intervals have the same length and are of the form

$$I_k = \left] \frac{2k}{M(M-1)}, \frac{2(k+1)}{M(M-1)} \right].$$

where k is the index of $m \in \mathcal{M}$ for some ordering on $\mathcal{M}$. Decoding determines which I_k contains the input and returns k, from which the original substring can be retrieved. For more general distributions p_m, each substring $m_{i,j}$ is mapped to an interval whose size depends on $p_m(m)$.

6.2.5.2 Length-Dependent Distributions

Let's consider the special case where $p_m(m)$ depends only on the length of m. We will therefore consider the function $p : [1, M] \longrightarrow [0, 1]$ giving the probability of a substring of a given length. This captures some properties of natural languages such as Zipf's law [Hei01]: Short expressions and words are used much more often than longer ones. Note that part of this is captured by the fact that there are fewer long substrings than short ones.

21. The Arabic equivalent is *madrasa*.
22. The way some characters do in Umberto Eco's novel, *Il pendolo di Foucault*[Eco11].

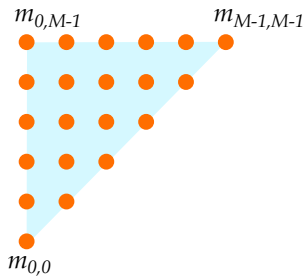


Figure 6.16 – Triangle representation T of the substrings $\mathcal{M} \subseteq \mathfrak{M}$. Substrings along right diagonals have equal length. The top-left point represents the entire corpus $\mathfrak{M}$.

Thus the encoding of a message $m_{i,j}$ is a random point in an interval of size $\ell(j-i)$ proportional to $p_m(m_{i,j}) = p(j-i)$:

$$\ell(k) = \frac{p(k)}{L}, \qquad L = \sum_{k=1}^{M} (M-k)p(k).$$

This ensures that

$$\sum_{k=1}^{M} (M-k)\ell(k) = 1.$$

The intervals associated to each substring are defined as follows. First, substrings $m_{i,j}$ are mapped via the map $\tau\colon m_{i,j} \mapsto (i,j)$ to a triangle (see Figure 6.16):

$$T = \{(i,j) \mid j \geq i \in [0, M-1]\} \subset \mathbb{N}^2.$$

Then points in T are mapped to $[0,1]$ using the function:

$$\Phi\colon (i,j) \mapsto (i-1)\ell(\mathrm{diag}(i,j)) + \sum_{k=1}^{\mathrm{diag}(i,j)-1} k\ell(k)$$

where $\mathrm{diag}(i,j) = M - 1 - (j-i)$ indicates on which upright diagonal (i,j) is. All in all, a substring $m_{i,j}$ is encoded using the following algorithm:

$$\mathsf{DTEncode} : m_{i,j} \mapsto (\Phi + \epsilon\ell \circ \mathrm{diag})\,(\tau(m_{i,j}))$$

where ϵ is sampled uniformly at random from $[0,1]$.

Encoding can be understood as follows: Substrings of equal length k are mapped by τ to points along a diagonal of constant $k = j - i$. The first diagonal is the whole corpus $\mathfrak{M}$ and the only substring of length M. The $(M-1-k)$-th diagonal is the set of substrings $\{m_{i,i+k} \mid i \in [0, M-1-k]\}$ of length k. Decoding is achieved by Algorithm 10, which takes a number $x \in [0,1]$ and returns the position $(i,j) = \Phi^{-1}(x)$ of the corresponding substring by determining the position in T. The idea is to count the segment length before x. At each iteration we update the segment length and the current position in the diagonal.

This decryption algorithm is linear in the number of substrings, *i.e.* it runs in time $O(M^2)$. We can speed things up using pre-computations, Algorithms 11 and 12 run in $O(M)$ time and memory.

6.2.6 Further Research

This work opens a number of interesting research directions:

Algorithm 10 Position of $\Phi^{-1}(x)$

Input: $x \in [0,1]$
Output: $(a,b) \in [|0,M|]^2$ such that $\Phi(a,b) = x$
  $i \leftarrow 0$
  $j \leftarrow 0$
  $k \leftarrow M$
  while $i < x$ **do**
    $i \leftarrow i + \ell(k)$
    $j \leftarrow j + 1$
    if $j \geq M - k + 1$ **then**
      $j \leftarrow 0$
      $k \leftarrow k - 1$
    end if
    return $(j-1, M+j-k-1)$
  end while

Algorithm 11 Pre-computation

Output: vector V such that intervals in $[V[i], V[i+1]]$ are the intervals of length $\ell(i)$
  let $V[1..M]$ be a vector of length M
  for $i \leftarrow 1$ to M **do**
    $V[i] \leftarrow V[i-1] + (M-i+1)\ell(i)$
  end for
  return V

Algorithm 12 Fast Decryption

Input: $x \in [0,1]$, V the result of Algorithm 11.
Output: $(a,b) \in [|0,M|]^2$ such that $\Phi(a,b) = x$
  $i \leftarrow 1$
  while $V[i] < x$ **do**
    $i++$
  end while
  $j \leftarrow (x - V[i])/\ell(i)$
  return $(j-1, M-i-1)$

6.2.6.1 Machine to Human HE:

Search engines, and more generally computational knowledge engines and answer engines such as Wolfram Alpha [23] provide users with structured answers that mimic human language. These algorithms generate messages using well-defined algorithmic process having a precise probability distribution which DTEs can be better modeled. Such sentences are hence likely to be safer to honey encrypt.

6.2.6.2 Automated Plaintext Pre-Processing:

A more advanced, yet not that unrealistic option consists in having a machine understand a natural language sentence m and re-encode m as a humanly understandable yet grammatically and syntactically simplified sentence m' having the same meaning for a human. Such an ontology-preserving simplification process will not modify the message's meaning while allowing the construction better DTEs.

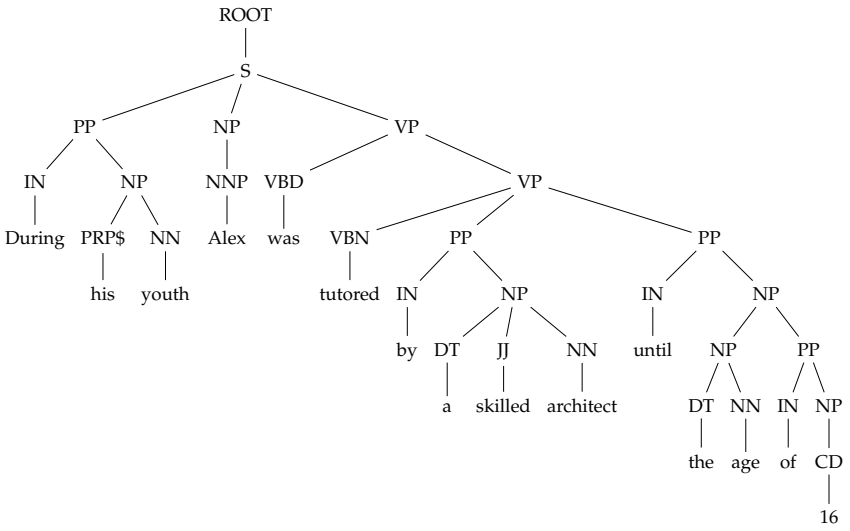
23. www.wolframalpha.com.

6.2.6.3 Adding Syntactic Defenses:

This work was mostly concerned by protecting messages at the *word* level. It is however possible to imagine the adding of defenses at the clause and at the phrase levels. Two simple clause-level protections consist in adding decoy clauses to the message, and shuffling the order of clauses in the message. Both transforms can be easily encoded in the ciphertext by adding an integer field interpreted as the rank of a permutation and a binary strong whose bits indicate which clauses should be discarded. Decryption with a wrong key will yield a wrong permutation and will remove useful skeletons from the message. It should be noted that whilst the permutation has very little cost, the addition of decoy skeletons impacts message length. It is important to use decoy skeletons that are indistinguishable from plausible skeletons. To that end the program can either pick skeletons in a huge database (*e.g.* the web) or generate them artificially.

6.2.6.4 Adding Phrase-Level Defenses:

Adding phrase-level defenses is also a very interesting research direction. A simple way to implement phrase-level defenses consists in adding outgrowths to the clause. An outgrowth is a collection of fake elements added using a specific rewriting rule. Note that information cannot be removed from the sentence. Here is an example of scrambling using outgrowths: the original clause m_0 is the sentence "During his youth Alex was tutored by a *skilled* architect until the age of 16". The syntactic tree of m_0 is:



The skeleton of m_0 is IN PRP$ NN NNP VBD VBN IN DT JJ NN IN DT NN IN CD.

Now consider the following rewrite rules:

PRP$ NN $\rightarrow$ PRP$ JJ NN
DT NN $\rightarrow$ DT JJ NN
IN DT JJ NN $\rightarrow$ IN DT NN CC IN DT JJ NN

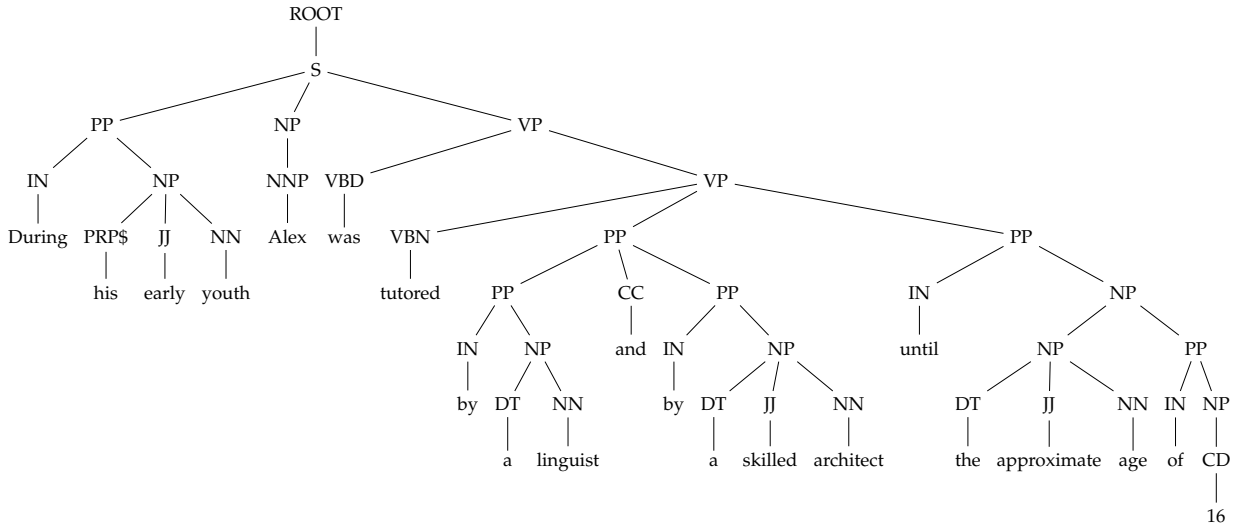
We can apply these rules to m_0 to obtain:

m_0	IN PRP$ NN NNP VBD VBN IN DT JJ NN IN DT NN IN CD
$m_1 \leftarrow r_1(m_0)$	IN PRP$ JJ NN NNP VBD VBN IN DT JJ NN IN DT NN IN CD
$m_2 \leftarrow r_2(m_1)$	IN PRP$ JJ NN NNP VBD VBN IN DT JJ NN IN DT JJ NN IN CD
$m_3 \leftarrow r_3(m_2)$	IN PRP$ JJ NN NNP VBD VBN IN DT NN CC IN DT JJ NN IN DT JJ NN IN CD

m_3 is a plausible skeleton that could have corresponded to the clause: "During his early youth Alex was tutored by a linguist and by a skilled architect until the approximate age of 16":

Table 6.1 – Partial list of grammatical roles.

	Clause Level
S	Simple declarative clause
SBAR	Clause introduced by a (possibly empty) subordinating conjunction.
	Phrase Level
ADVP	Adverb phrase
NP	Noun phrase
PP	Prepositional phrase
VP	Verb phrase
	Word Level
CC	Conjunction, coordinating
DT	Determiner
IN	Preposition or subordinating conjunction
JJ	Adjective
MD	Modal
NN	Noun, singular or mass
PRP	Pronoun, personal
PRP$	Pronoun, possessive
RB	Adverb
VB	Verb, base form
VBN	Verb, past participle
VBZ	Verb, third person singular present



It remains to show how to reverse the process to recover the original skeleton m_0. To that end, we include in the ciphertext a binary string indicating which outgrowths should be removed. Removal consists in scanning m_0 and identifying what could have been the result of rewriting. Scanning reveals one potential application of rule 1 (namely "his early youth"), two potential applications of rule 2 ("a skilled architect" and "the approximate age") and one potential application of rule 2 ("by a linguist and by a skilled architect"). Hence 4 bits suffice to identify and remove the outgrowths.

6.2.7 Grammatical Tags for English

6.3 Compact CCA2-Secure Hierarchical ID-Based Broadcast Encryption with Public Ciphertext Test

6.3.1 Introduction

Identity-Based Encryption (IBE), introduced by Shamir [Sha85], allows one to securely communicate with others if he/she knows their public identities. In IBE, users' recognizable identities such as their social security numbers, IPs or email addresses, are used as their public keys. A Private Key Generator (PKG) is used to generate secret keys associated with the users' public identities. One can encrypt to any user by specifying its recognizable identity and only the intended user can decrypt.

Hierarchical IBE (HIBE) extends IBE to endow a large number of users with a delegation mechanism. HIBE [GS02] organizes users in a tree-like structure which is consistent with the structure of many social organizations [DWQ+14; ZWD+14]. PKG's burden is shared by upper-level users who can delegate secret keys to their subordinates. In the encryption process, the sender associates the ciphertext with an identity vector instead of a single identity. Then only the users whose identities appear in the specified identity vector can decrypt. For instance, to securely communicate with a university professor, one just needs to encrypt with a specified identity vector "university: XXX | school: XXX | laboratory: XXX | professor: XXX". Then, only the individuals whose identity vectors match the prefix of the assigned identity vector are able to decrypt.

In applications similar to the above, one may have to simultaneously communicate with multiple users in hierarchical organizations. For example, a company may cooperate with a number of professors from different laboratories in a university to develop a new software system. The company can separately encrypt to these professors by specifying their respective decryption paths. However, this trivial solution incurs heavy encryption burden and long ciphertexts. Another application comes from IP-based multicast networks, in which all network nodes are organized in a hierarchy expressed in terms of IP addresses and subnet masks. Since sensitive contents in such networks can be easily intercepted by packet sniffers, efficient and secure cryptographic systems are needed. Applying existing HIBE Schemes (HIBES) in multicast networks would be a reasonable solution. However, HIBES gradually becomes inefficient when the number of nodes from different IP paths increases. We are interested in more practical solutions to such applications.

6.3.1.1 Our Contributions

Motivated by the above scenarios, we propose a new cryptographic primitive called Hierarchical Identity-Based Broadcast Encryption (HIBBE). Users in a tree-like structure can delegate their decryption capabilities to their subordinates, so that the burden of the PKG can be shared when the system hosts a large number of users. One can encrypt to any subset of the users and only the intended ones and their supervisors can decrypt.

We define Ciphertext Indistinguishability against Adaptively Chosen-Identity-Vector-Set and Chosen-Ciphertext attack (IND-CIVS-CCA2). In this notion, the attacker is simultaneously allowed to adaptively query for the secret keys of users recognized by identity vectors of its choice and to issue decryption queries for receiver identity vector sets at wish. Even such an attacker cannot distinguish the encrypted messages, provided that the attacker does not query for the secret keys of the target users or their supervisors. Clearly, this definition captures the most powerful attacks on HIBBE in the real world.

We obtain an IND-CIVS-CCA2 scheme in the standard model (without using random oracles) in two steps. We first construct an HIBBE Scheme (HIBBES) against Adaptively Chosen-Identity-Vector-Set and Chosen-Plaintext Attack (IND-CIVS-CPA) in the standard model, in which the attacker is not allowed to issue decryption queries. Then, at merely marginal cost, we convert the basic scheme into an IND-CIVS-CCA2 scheme by adding only one on-the-fly dummy user, rather than adding one hierarchy of users in existing conversions from a CPA-secure hierarchical encryption scheme to a CCA2-secure one. Both schemes have constant size ciphertext and are efficient in terms of communications in multi-receiver situations.

Compared with the preliminary version [LLWQ14] of the paper, in this extended work we give the formal security proof of the CPA security of the basic scheme; we further convert the CPA-secure HIBBES into a CCA2-secure HIBBES with compact design in the sense that the conversion does not require any other cryptographic primitives; we formally prove that the resulting scheme is CCA2-secure in the standard model. Our CCA2-secure HIBBES allows public ciphertext validity test which is useful for a third party, e.g., a firewall, to filter invalid spam and for system designers to design advanced protocols from HIBBE, e.g., publicly verifiable HIBBE and fair exchange of HIBBE-encrypted digital contents.

6.3.1.2 Related Work

Identity-Based Encryption. Since the concept of Identity-Based Encryption (IBE) was introduced by Shamir [Sha85], it took a long time for researchers to construct a practical and fully functional IBE Scheme (IBES). In 2001, Boneh and Franklin [BF01; BF03] precisely defined the security model of IBE and proposed the first practical IBES by using bilinear pairings. In the Boneh-Franklin security model, the adversary can adaptively request secret keys for the identities of its choice and can choose the challenge identity it wants to attack at any point during the key-requesting process, provided that the secret key for the challenging identity is not queried. The security of their IBES [BF01; BF03] requires cryptographic hash functions to be modeled as random oracles. Canetti *et al.* [CHK03; CHK04] formalized a slightly weaker security notion, called selective-ID security, in which the adversary must disclose the challenge identity before the public parameters are generated. They exhibited a selective-ID secure IBES without using random oracles. Since then, more practical IBES have been proposed that are shown to be secure without random oracles in the selective-ID security model [BB04] or in the standard security model [Wat05]. These schemes are secure against CPA. Interestingly, some recent works [BK05; BMW05; CHK04] showed CPA-secure IBES can be used to construct regular Public-Key Encryption systems with CCA2 security. Canetti, Halevi and Katz [CHK04] exhibited a generic conversion by adding a one-time signature scheme and hash the signature parameters as a special "identity" in encryption. Boneh and Katz [BK05] later presented a more efficient construction using a MAC to replace the one-time signature. More recently, Boyen *et al.* [BMW05] introduced a new technique that can directly obtain CCA2 security from some particular IBES without extra cryptographic primitives. Park *et al.* [PLL15] proposed a concrete CCA2-secure IBES with a tight security reduction in the random oracle model.

Broadcast Encryption. In Broadcast Encryption (BE) [FN94], a dealer is employed to generate and distribute decryption keys for users. A sender can encrypt to a subset of the users and only the privileged users can decrypt. This functionality models flexible secure one-to-many communication scenarios [QWZ+12]. Since the BE concept was introduced in 1994 [FN94], many BE Schemes have been proposed to gain preferable properties. We mention just a few of those properties, such as "Stateless Receivers" (after getting the broadcast secret keys, users do not need to update them) [DF03; HS02], "Fully Collusion Resistant" (even if all users except the receiver set collude, they can obtain no information about the plaintext) [BGW05], "Dynamic" (the dealer can dynamically recruit new members while the other members will not be affected) [DPP07], "Anonymity" (a receiver does not need to know who the other receivers are when decrypting ciphertexts) [LPQ12], and "Contributory Broadcast" (Anyone can send messages to any subset of the group members without a trusted key server) [WQZ+16].

Identity-Based Broadcast Encryption. Identity-Based Broadcast Encryption (IBBE) incorporates the idea of BE into IBE and recognizes the users in a BES with their identities, instead of indexes assigned by the system. When one needs to send confidential messages to multiple users, the sender in IBBE can efficiently encrypt the message once to multiple users and simply broadcasts the resulting ciphertext. Fully functional IBBE was formalized and realized by Delerablée with constant size ciphertexts and secret keys [Del07], although it is only selective-ID secure in the random oracle model. The up-to-date IBBE Schemes [GW09; RG09; KSAS15] are shown to be secure in the standard security model.

Hierarchical Identity-Based Encryption. Horwitz and Lynn [HL02] first proposed the concept of HIBE and presented a two-level HIBES in the same article. The first fully functional HIBE construction was proposed by Gentry and Silverberg [GS02]. The security relies on the Bilinear Diffie-Hellman assumption in the random oracle model. Subsequently, Boneh and Boyen [BB04] introduced HIBES in the selective-ID model without using random oracles. Boneh, Boyen and Goh [BBG05] presented a selective-ID

secure HIBE with constant size ciphertext. Gentry and Halevi [GH09] constructed a fully secure HIBES supporting polynomial hierarchy depth. In 2009, Waters [Wat09] proposed a new framework, called Dual System Encryption, for constructing fully secure IBES and HIBES. This approach has become a powerful tool for obtaining fully secure encryption schemes [LW10; LW12]. These plain HIBES are CPA-secure. The techniques in the previously reviewed conversions [BK05; BMW05; CHK04] can be extended to achieve CCA2-secure HIBES with CPA-secure ones by adding one extra hierarchy to the underlying CPA-secure HIBES.

Generalized Identity-Based Encryption. Boneh and Hamburg [BH08] proposed a general framework for constructing IBES, referred to as Generalized Identity-Based Encryption (GIBE), to incorporate different properties in IBE via a product rule. They also introduced an important instance of GIBE called Spatial Encryption (SE), showing that many GIBES are embedded in it, e.g., HIBE, inclusive IBE, co-inclusive IBE, in an identity-based like settings. HIBBE can also be derived from SE. However, the HIBBE derived from their SE only has selective and chosen-plaintext security. Very recently, Zhang *et al.* [ZYT14] suggested two fully secure and anonymous SE schemes, which not only obtain full security, but further protect the recipient identity privacy. Their constructions achieve CPA security and can be extended to CCA2 security, but also with the help of one-time signature schemes.

6.3.1.3 Organization of This Section

The rest of the section is organized as follows. In sub-section 6.3.2, we review composite order bilinear groups and the assumptions used in our constructions. Sub-section 6.3.3 formalizes HIBBE and its security definitions. We propose a secure HIBBES against Adaptively Chosen-Identity-Vector-Set and Chosen-Plaintext Attack in Sub-section 6.3.4. We then introduce a compact transformation that converts our CPA-secure HIBBES into a CCA2-secure one in Sub-section 6.3.5. Finally, we state our conclusions in Sub-section 6.3.6.

6.3.2 Preliminaries

6.3.2.1 Composite Order Bilinear Groups

Composite order bilinear groups were first introduced in [BGN05]. Let $\mathcal{G}$ be an algorithm which takes a security parameter λ as input and outputs the description of a bilinear group, $(N, \mathbb{G}, \mathbb{G}_T, e)$, where $N = p_1p_2p_3$ is a composite integer with three distinct large prime factors p_1, p_2 and p_3, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order N, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfying the following properties:

1. *Bilinearity*: for all $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_N$, $e(g^a, h^b) = e(g, h)^{ab}$;
2. *Non-degeneracy*: there exists at least an element $g \in \mathbb{G}$ such that $e(g, g)$ has order N in $\mathbb{G}_T$;
3. *Computability*: There exists an efficient algorithm (in polynomial time with respect to λ) computing the bilinear pairing $e(u, v)$ for all $u, v \in \mathbb{G}$.

In addition to these properties, the three subgroups of order p_1, p_2 and p_3 in $\mathbb{G}$ (we respectively denote them by $\mathbb{G}_{p_1}$, $\mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$) satisfy the orthogonality property:

$$\text{For all } h_i \in \mathbb{G}_{p_i} \text{ and } h_j \in \mathbb{G}_{p_j}, \text{ if } i \neq j, \text{ then } e(h_i, h_j) = 1$$

This special property will be an essential tool in our constructions and the security proofs.

6.3.2.2 Assumptions in Composite Order Bilinear Groups

We will use three static assumptions to prove the security of our HIBBES. These three assumptions, which were first introduced by Lewko and Waters [LW10], hold if it is hard to find a nontrivial factor of N. Let $\mathcal{G}$ be a group generating algorithm that outputs a composite order bilinear group $(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$. For ease of description, we let $\mathbb{G}_{p_ip_j}$ denote the subgroup of order p_ip_j in $\mathbb{G}$.

Let $g \overset{R}{\leftarrow} \mathbb{G}_{p_1}$ be a random generator of $\mathbb{G}_{p_1}$ and $X_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$ be a random element in $\mathbb{G}_{p_3}$. **Assumption 1** is that it is hard to determine whether T is a random element in $\mathbb{G}_{p_1p_2}$, or a random element in $\mathbb{G}_{p_1}$ given $D_1 = (g, X_3)$ as an input. We define the advantage of an algorithm $\mathcal{A}$ that outputs $b \in \{0, 1\}$ in solving the first assumption in $\mathbb{G}$ to be

$$Adv1_{\mathcal{A}}(\lambda) = \left|\Pr\left[\mathcal{A}\left(D_1, T \overset{R}{\leftarrow} \mathbb{G}_{p_1p_2}\right) = 1\right] - \Pr\left[\mathcal{A}\left(D_1, T \overset{R}{\leftarrow} \mathbb{G}_{p_1}\right) = 1\right]\right|$$

Definition 6.4 *Assumption 1 states that $Adv1_{\mathcal{A}}(\lambda)$ is negligible for all polynomial time algorithms $\mathcal{A}$.*

Let $g \overset{R}{\leftarrow} \mathbb{G}_{p_1}$ be a random generator of $\mathbb{G}_{p_1}$. Choose random elements $X_1 \overset{R}{\leftarrow} \mathbb{G}_{p_1}$, $X_2, Y_2 \overset{R}{\leftarrow} \mathbb{G}_{p_2}$ and $X_3, Y_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$. **Assumption 2** is that given $D_2 = (g, X_1X_2, X_3, Y_2Y_3)$ as an input, it is hard to determine whether T is a random element in $\mathbb{G}$ or a random element in $\mathbb{G}_{p_1p_3}$. We define the advantage of an algorithm $\mathcal{A}$ that outputs $b \in \{0, 1\}$ in solving the second assumption in $\mathbb{G}$ to be

$$Adv2_{\mathcal{A}}(\lambda) = \left|\Pr\left[\mathcal{A}\left(D_2, T \overset{R}{\leftarrow} \mathbb{G}\right) = 1\right] - \Pr\left[\mathcal{A}\left(D_2, T \overset{R}{\leftarrow} \mathbb{G}_{p_1p_3}\right) = 1\right]\right|$$

Definition 6.5 *Assumption 2 states that $Adv2_{\mathcal{A}}(\lambda)$ is negligible for all polynomial time algorithms $\mathcal{A}$.*

Similarly, let $g \overset{R}{\leftarrow} \mathbb{G}_{p_1}$ be a random generator of $\mathbb{G}_{p_1}$, $X_2, Y_2, Z_2 \overset{R}{\leftarrow} \mathbb{G}_{p_2}$ be random elements in $\mathbb{G}_{p_2}$, $X_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$ be a random element in $\mathbb{G}_{p_3}$, $\alpha, s \overset{R}{\leftarrow} \mathbb{Z}_N$ be random exponents chosen in $\mathbb{Z}_N$. **Assumption 3** states that, given $D_3 = (g, g^{\alpha}X_2, X_3, g^sY_2, Z_2)$ as an input, it is hard to determine whether T is $e(g, g)^{\alpha s}$, or a random element in $\mathbb{G}_T$. We define the advantage of an algorithm $\mathcal{A}$ that outputs $b \in \{0, 1\}$ in solving the third assumption in $\mathbb{G}$ to be

$$Adv3_{\mathcal{A}}(\lambda) = \left|\Pr\left[\mathcal{A}\left(D_3, T \leftarrow e(g, g)^{\alpha s}\right) = 1\right] - \left[\mathcal{A}\left(D_3, T \overset{R}{\leftarrow} \mathbb{G}_T\right)\right] = 1\right|$$

Definition 6.6 *Assumption 3 states that $Adv3_{\mathcal{A}}(\lambda)$ is negligible for all polynomial time algorithms $\mathcal{A}$.*

6.3.3 Syntax

6.3.3.1 Terminology and Notations

We introduce several notations to simplify the description of HIBBES. Table 6.2 summarizes these notations and their corresponding meanings that will be used in the paper.

Table 6.2 – Notations

Notation	Description	Notation	Description
λ	Security Parameter	PK	Public Key
MSK	Master Key	CT	Ciphertext
ID	Identity	**ID**	Identity Vector
$\mathrm{I}_{\mathbf{ID}}$	Identity Vector Position	$SK_{\mathbf{ID}}$	Secret Key for Identity Vector
$\|\mathbf{ID}\|$	Depth of **ID**	$S_{\mathbf{ID}}$	Identity Set Associated with **ID**
V	Identity Vector Set	$\mathbb{I}_{\mathbf{V}}$	Identity Vector Set Position
$\|\mathbf{V}\|$	Depth of **V**	$S_{\mathbf{V}}$	Identity Set Associated with **V**

We use $[a, b]$ to denote the integer set $\{a, a + 1, \cdots, b\}$. $|S|$ denotes the cardinality of the set S. For an identity vector $\mathbf{ID} = (ID_1, ID_2, \cdots, ID_d)$, we define $\|\mathbf{ID}\| = d$ as the depth of **ID** and $S_{\mathbf{ID}} = \{ID_1, \cdots, ID_d\}$ as the identity set associated with **ID**. The identity vector position of **ID** is defined by $\mathrm{I}_{\mathbf{ID}} = \{i : ID_i \in S_{\mathbf{ID}}\}$. Similarly, we define the maximal depth of an identity vector set as $\|\mathbf{V}\| = \max\{\|\mathbf{ID}\| : \mathbf{ID} \in \mathbf{V}\}$. The associated identity set $S_{\mathbf{V}}$ of **V** and the identity vector set position $\mathbb{I}_{\mathbf{V}}$ of **V** can be defined accordingly.

We slightly abuse the term prefix and define the prefix of an identity vector $\mathbf{ID} = (ID_1, \cdots, ID_d)$ as an identity vector set denoted by $\mathrm{Pref}(\mathbf{ID}) = \{(ID_1, \cdots, ID_{d'}) : d' \leq d\}$. Clearly, $|\mathrm{Pref}(\mathbf{ID})| = \|\mathbf{ID}\| = d$. We similarly define the prefix of an identity vector set **V** as $\mathrm{Pref}(\mathbf{V}) = \bigcup_{\mathbf{ID} \in \mathbf{V}} \mathrm{Pref}(\mathbf{ID})$.

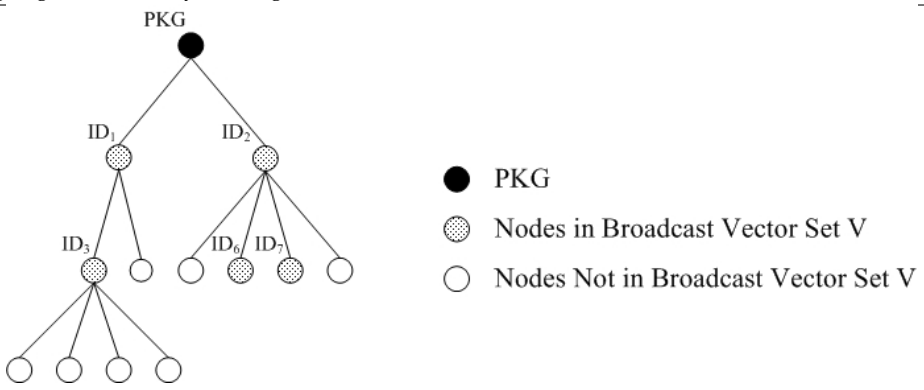


Figure 6.17 – A Typical Example of an HIBBES.

In practice, a user may have more than one identity or parent node. In this case, we treat these users as different users with the same identity. Hence, without loss of generality, we assume that each user has a unique identity vector and can have at most one parent node.

For example, assume that the users are organized as in Figure 6.17. For the user whose identity vector is $\mathbf{ID} = (\mathrm{ID}_1, \mathrm{ID}_3)$, we have that $\|\mathbf{ID}\| = 2$, $S_{\mathbf{ID}} = \{\mathrm{ID}_1, \mathrm{ID}_3\}$, and $\mathrm{I}_{\mathbf{ID}} = \{1, 3\}$. The prefix of **ID** is $\mathrm{Pref}(\mathbf{ID}) = \{(\mathrm{ID}_1), (\mathrm{ID}_1, \mathrm{ID}_3)\}$. Similarly, for the broadcast identity vector set $\mathbf{V} = \{(\mathrm{ID}_1, \mathrm{ID}_3), (\mathrm{ID}_2, \mathrm{ID}_6, \mathrm{ID}_7)\}$, we have that $\|\mathbf{V}\| = \max\{2, 3\} = 3$, $S_{\mathbf{V}} = \{\mathrm{ID}_1, \mathrm{ID}_3, \mathrm{ID}_2, \mathrm{ID}_6, \mathrm{ID}_7\}$, and $\mathbb{I}_{\mathbf{V}} = \{1, 3, 2, 6, 7\}$. The prefix of **V** is

$$\mathrm{Pref}(\mathbf{V}) = \{(\mathrm{ID}_1), (\mathrm{ID}_1, \mathrm{ID}_3), (\mathrm{ID}_2), (\mathrm{ID}_2, \mathrm{ID}_6), (\mathrm{ID}_2, \mathrm{ID}_6, \mathrm{ID}_7)\}$$

6.3.3.2 Hierarchical Identity-Based Broadcast Encryption (HIBBE)

A (D, n)-HIBBES consists of five polynomial time algorithms: **Setup**, **KeyGen**, **Delegate**, **Encrypt** and **Decrypt** defined as follows:

- **Setup**(D, n, λ). Takes as inputs the maximal depth D of the hierarchy, the maximal number n of users, and the security parameter λ. It outputs a masker key MSK and a public key PK.
- **Encrypt**(PK, M, **V**). Takes as inputs the public key PK, a message M in the message space $\mathcal{M}$, and a receiver identity vector set **V**. The algorithm outputs the ciphertext CT of the message M.
- **KeyGen**(MSK, **ID**). Takes as inputs the master key MSK and an identity vector **ID**. It outputs a secret key $SK_{\mathbf{ID}}$ for the user whose identity vector is **ID**.
- **Delegate**($SK_{\mathbf{ID}'}$, ID). Takes as inputs a secret key of a user whose identity vector is **ID**′ of depth d and an identity ID. It returns a secret key $SK_{\mathbf{ID}}$ for the user whose identity vector is $\mathbf{ID} = (\mathbf{ID}', ID)$.
- **Decrypt**(**V**, CT, $SK_{\mathbf{ID}}$). Takes as inputs a receiver identity vector set **V**, a ciphertext CT of a message M, and a secret key $SK_{\mathbf{ID}}$ of a user whose identity vector is **ID**. If $\mathbf{ID} \in \mathrm{Pref}(\mathbf{V})$, it returns M.

An HIBBES must satisfy the standard consistency constraint, namely for all $D \leq n \in \mathbb{N}$, all (PP, MSK) $\leftarrow$ **Setup**(D, n, λ), all $SK_{\mathbf{ID}} \leftarrow$ **KeyGen**(MSK, **ID**) or $SK_{\mathbf{ID}} \leftarrow$ **Delegate**($SK_{\mathbf{ID}'}$, ID) with $\|\mathbf{ID}\| \leq D$, all $M \in \mathcal{M}$, and all $CT \leftarrow$ **Encrypt**(PP, M, **V**) with $\|\mathbf{V}\| \leq D$ and $|S_{\mathbf{V}}| \leq n$, if $\mathbf{ID} \in \mathrm{Pref}(\mathbf{V})$, then **Decrypt**(**V**, CT, $SK_{\mathbf{ID}}$) = M.

We define Ciphertext Indistinguishability against Adaptively Chosen-Identity-Vector-Set and Chosen-Ciphertext Attack (IND-CIVS-CCA2) in HIBBE. In this security model, the adversary is allowed to obtain the secret keys associated with any identity vectors **ID** of its choice and to issue decryption queries for its chosen ciphertexts, provided that the adversary does not query for the secret keys of its chosen receivers or their supervisors, or for the challenge ciphertext as one of its chosen messages. We require that even such an adversary cannot distinguish the encrypted messages of its choice.

Formally, IND-CIVS-CCA2 security is defined through a game played by an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Both of them are given the parameters D, n and λ as inputs.

— **Setup**. $\mathcal{C}$ runs **Setup** algorithm to obtain the public key PK and gives it to $\mathcal{A}$.
— **Phase 1**. $\mathcal{A}$ adaptively issues two kinds of queries:
  — Secret key query for an identity vector **ID**. $\mathcal{C}$ generates a secret key for **ID** and gives it to $\mathcal{A}$.
  — Decryption query for the ciphertext CT with a receiver identity vector set $\mathbf{V}$. $\mathcal{C}$ responds by running algorithm **KeyGen** to generate a secret key $SK_{\mathbf{ID}}$ for an identity vector **ID** satisfying **ID** $\in$ Pref($\mathbf{V}$). It then decrypts the ciphertext CT and returns the resulting message to $\mathcal{A}$.
— **Challenge**. When $\mathcal{A}$ decides that **Phase 1** is over, it outputs two equal-length messages M_0 and M_1 on which $\mathcal{A}$ wishes to be challenged. Also, $\mathcal{A}$ outputs a challenge identity vector set $\mathbf{V}^*$ which contains all the users that $\mathcal{A}$ wishes to attack. The identity vector set $\mathbf{V}^*$ should be such that for all the secret key queries for **ID** issued in **Phase 1**, **ID** $\notin$ Pref($\mathbf{V}^*$). $\mathcal{C}$ flips a random coin $b \xleftarrow{R} \{0,1\}$ and encrypts M_b under the challenge identity vector set $\mathbf{V}^*$. $\mathcal{C}$ returns the challenge ciphertext CT^* to $\mathcal{A}$.
— **Phase 2**. $\mathcal{A}$ further adaptively issues two kinds of queries:
  — Secret key queries for identity vectors **ID** such that **ID** $\notin$ Pref($\mathbf{V}^*$).
  — Decryption queries for ciphertexts CT such that $CT \neq CT^*$.
  $\mathcal{C}$ responds the same as in **Phase 1**.
— **Guess**. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ and wins in the game if $b = b'$.

The advantage of such an $\mathcal{A}$ in attacking the (D, n)-HIBBES with security parameter λ is defined as

$$Adv_{\mathcal{A},D,n}^{IND-CIVS-CCA2}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

Definition 6.7 *A (D, n)-HIBBES is (τ, q, q_d, ϵ)-secure if for any τ-time IND-CIVS-CCA2 adversary $\mathcal{A}$ that makes at most q secret key queries and q_d decryption queries, we have that $Adv_{\mathcal{A},D,n}^{IND-CIVS-CCA2}(\lambda) < \epsilon$*

As usual, we define Ciphertext Indistinguishability against Adaptively Chosen-Identity-Vector-Set and Chosen-Plaintext Attack (IND-CIVS-CPA) for HIBBE as in the preceding game, with the constraint that $\mathcal{A}$ is not allowed to issue any decryption query. $\mathcal{A}$ is still able to adaptively issue secret key queries.

Definition 6.8 *A (D, n)-HIBBES is (τ, q, ϵ)-secure if for any τ-time IND-CIVS-CPA adversary $\mathcal{A}$ that makes at most q secret key queries, we have that $Adv_{\mathcal{A},D,n}^{IND-CIVS-CPA}(\lambda) < \epsilon$.*

It is challenging to achieve full (identity/identity-vector) security in BE and (H)IBE, some weaker security notions have been proposed to bridge security proofs or cater for special applications which require only moderate security levels. One useful security notion, called selective security, was first proposed by Canetti, Halevi, and Katz [CHK03; CHK04] in IBES. In this notion, $\mathcal{A}$ should commits ahead of time to the challenge identity it will attack. Similar security notions can also be found in HIBES [BB04] and IBBES [Del07]. A counterpart security notion can be naturally defined in HIBBES, by requiring the adversary in HIBBE to submit a challenge identity vector set before seeing the public parameters.

Another useful security notion, named semi-static security, can also be extended in HIBBES. This security notion was first defined by Gentry and Waters [GW09] in BES. In this notion, $\mathcal{A}$ must first commit to a set $\overline{S}$ before the **Setup** phase. $\mathcal{A}$ cannot query for secret key of any user in $\overline{S}$, but it can attack any target set $S^* \subseteq \overline{S}$. This security notion is weaker than full security but stronger than selective security, since $\mathcal{A}$ can partly decide which set is allowed to query adaptively. In HIBBES, a similar security notion can be defined by requiring $\mathcal{A}$ to submit an identity vector set $\overline{\mathbf{V}}$ before the **Setup** phase and later allow $\mathcal{A}$ to challenge any identity vector set $\mathbf{V}^* \subseteq$ Pref($\overline{\mathbf{V}}$). Recently, a practical HIBBES with moderate security result was proposed to meet this security notion [LLW+16a].

6.3.4 IND-CIVS-CPA Secure HIBBE with Constant Size Ciphertext

In this section, we propose an IND-CIVS-CPA secure HIBBE with constant size ciphertext over composite order bilinear groups of order $N = p_1 p_2 p_3$. Our starting point is the Lewko-Waters fully secure HIBES [LW10] which was inspired by the Boneh-Boyen-Goh selectively secure HIBES [BBG05]. To support

broadcast, every user in our system, instead of every depth of hierarchy in [BBG05; LW10], is associated with a random element for blinding its own identity vector in its secret key. Since users' identities have been randomized by different elements, users cannot reveal any information about other users' secret keys from their own ones.

We realize the functionalities in $\mathbb{G}_{p_1}$, while randomizing secret keys in $\mathbb{G}_{p_3}$. The $\mathbb{G}_{p_2}$ space, called semi-functional space, is only used in security proofs.

6.3.4.1 Basic Construction

We first assume that the identity vectors $\mathbf{ID} = (ID_1, \cdots, ID_k)$ at depth k are vector elements in $(\mathbb{Z}_N)^k$. We later extend the construction to identity vectors over $(\{0,1\}^*)^k$ by first hashing each component $ID_j \in S_{\mathbf{ID}}$ using a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_N$. We also assume that plaintexts are elements of $\mathbb{G}_T$. Similar to HIBES, we assume that users' positions in HIBBE are publicly known with the processing of **KeyGen**, **Delegate**, **Encrypt** and **Decrypt**. Our (D, n)-HIBBES works as follows.

Setup(D, n, λ). Run $(N, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$ to generate a composite integer $N = p_1p_2p_3$, two groups $\mathbb{G}$, $\mathbb{G}_T$ of order N, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Then, select a random generator $g \xleftarrow{R} \mathbb{G}_{p_1}$, two random elements $h \xleftarrow{R} \mathbb{G}_{p_1}$, $X_3 \xleftarrow{R} \mathbb{G}_{p_3}$, and a random exponent $\alpha \xleftarrow{R} \mathbb{Z}_N$. Next, pick random elements $u_i \xleftarrow{R} \mathbb{G}_{p_1}$ for all $i \in [1, n]$. The public key PK includes the description of $(N, \mathbb{G}, \mathbb{G}_T, e)$, as well as

$$(g, h, u_1, \cdots, u_n, X_3, e(g,g)^\alpha)$$

The master key is $MSK \leftarrow g^\alpha$.

KeyGen(MSK, **ID**). For an identity vector **ID** of depth $d \leq D$, the key generation algorithm picks a random exponent $r \xleftarrow{R} \mathbb{Z}_N$ and two random elements $A_0, A_1 \xleftarrow{R} \mathbb{G}_{p_3}$. It then chooses random elements $U_j \xleftarrow{R} \mathbb{G}_{p_3}$ for all $j \in [1, n] \backslash \mathrm{I}_{\mathbf{ID}}$ and outputs

$$SK_{\mathbf{ID}} \leftarrow \left(g^\alpha \left(h \cdot \prod_{i \in \mathrm{I}_{\mathbf{ID}}} u_i^{ID_i} \right)^r A_0, g^r A_1, \left\{ u_j^r U_j \right\}_{j \in [1,n] \backslash \mathrm{I}_{\mathbf{ID}}} \right)$$

Delegate($SK_{\mathbf{ID}'}$, ID). Given a secret key

$$SK_{\mathbf{ID}'} = \left(g^\alpha \left(h \cdot \prod_{i \in \mathrm{I}_{\mathbf{ID}'}} u_i^{ID_i} \right)^{r'} A'_0, g^{r'} A'_1, \left\{ u_j^{r'} U'_j \right\}_{j \in [1,n] \backslash \mathrm{I}_{\mathbf{ID}'}} \right) = \left(a_0, a_1, \{b_j\}_{j \in [1,n] \backslash \mathrm{I}_{\mathbf{ID}'}} \right)$$

the delegation algorithm generates a secret key for $\mathbf{ID} = (\mathbf{ID}', ID)$ as follows. It first picks a random exponent $t \xleftarrow{R} \mathbb{Z}_N$, and also chooses two random elements $R_0, R_1 \xleftarrow{R} \mathbb{G}_{p_3}$. Next, for all $j \in [1, n] \backslash \mathrm{I}_{\mathbf{ID}}$, it chooses random elements $T_j \xleftarrow{R} \mathbb{G}_{p_3}$. The algorithm outputs

$$SK_{\mathbf{ID}} = \left(a_0 \left(b_i^{ID}\right)_{i \in \mathrm{I}_{\mathbf{ID}} \backslash \mathrm{I}_{\mathbf{ID}'}} \left(h \prod_{i \in \mathrm{I}_{\mathbf{ID}}} u_i^{ID_i} \right)^t R_0, a_1 g^t R_1, \left\{ b_j u_j^t T_j \right\}_{j \in [1,n] \backslash \mathrm{I}_{\mathbf{ID}}} \right)$$

Note that by implicitly setting $r = r' + t \in \mathbb{Z}_N$, $A_0 = A'_0 U'^r_i R_0 \in \mathbb{G}_{p_3}$ with $i \in \mathrm{I}_{\mathbf{ID}} \backslash \mathrm{I}_{\mathbf{ID}'}$, $A_1 = A'_1 R_1 \in \mathbb{G}_{p_3}$, and $U_j = U'_j T_j \in \mathbb{G}_{p_3}$ for all $j \in [1, n] \backslash \mathrm{I}_{\mathbf{ID}}$, this delegated secret key can be written under the form

$$SK_{\mathbf{ID}} \leftarrow \left(g^\alpha \left(h \cdot \prod_{i \in \mathrm{I}_{\mathbf{ID}}} u_i^{ID_i} \right)^r A_0, g^r A_1, \left\{ u_j^r U_j \right\}_{j \in [1,n] \backslash \mathrm{I}_{\mathbf{ID}}} \right)$$

which is well-formed as if it were generated by the **KeyGen** algorithm. Hence, $SK_{\mathbf{ID}}$ is a properly distributed secret key for $\mathbf{ID} = (\mathbf{ID}', ID)$.

Encrypt(PP, M, $\mathbf{V}$). For the receiver identity vector set $\mathbf{V}$ the encryption algorithm picks a random exponent $\beta \xleftarrow{R} \mathbb{Z}_N$ and outputs the ciphertext

$$CT = (C_0, C_1, C_2) = \left(g^{\beta}, \left(h \cdot \prod_{i \in \mathbb{I}_{\mathbf{V}}} u_i^{ID_i} \right)^{\beta}, e(g,g)^{\alpha\beta} \cdot M \right)$$

Decrypt($\mathbf{V}$, CT, $SK_{\mathbf{ID}}$). Given the ciphertext $CT = (C_0, C_1, C_2)$, any user whose identity vector is $\mathbf{ID} \in \mathrm{Pref}(\mathbf{V})$ can use its secret key $SK_{\mathbf{ID}} = \left(a_0, a_1, \{b_j\}_{j \in [1,n] \setminus \mathrm{I}_{\mathbf{ID}}}\right)$ to compute

$$K = a_0 \cdot \prod_{j \in \mathbb{I}_{\mathbf{V}} \setminus \mathrm{I}_{\mathbf{ID}}} b_j^{ID_j}$$

Then it outputs the message by calculating

$$M = C_2 \cdot \frac{e(C_1, a_1)}{e(K, C_0)}$$

Soundness. If the ciphertext $CT = (C_0, C_1, C_2)$ is well-formed, then we have

$$K = a_0 \cdot \prod_{j \in \mathbb{I}_{\mathbf{V}} \setminus \mathrm{I}_{\mathbf{ID}}} b_j^{ID_j} = g^{\alpha} \left(h \cdot \prod_{i \in \mathbb{I}_{\mathbf{V}}} u_i^{ID_i} \right)^{r} \cdot \left(A_0 \prod_{j \in \mathbb{I}_{\mathbf{V}} \setminus \mathrm{I}_{\mathbf{ID}}} U_j \right)$$

Note that all random elements in $\mathbb{G}_{p_3}$ can be cancelled in the pairing operations due to the orthogonality property. Therefore, for the blinding factor in C_2, the following equalities hold:

$$\frac{e(C_1, a_1)}{e(K, C_0)} = \frac{e\left(\left(h \cdot \prod_{i \in \mathbb{I}_{\mathbf{V}}} u_i^{ID_i} \right)^{\beta}, g^r A_1 \right)}{e\left(g^{\alpha} \left(h \cdot \prod_{i \in \mathbb{I}_{\mathbf{V}}} u_i^{ID_i} \right)^{r} \cdot \left(A_0 \prod_{j \in \mathbb{I}_{\mathbf{V}} \setminus \mathrm{I}_{\mathbf{ID}}} U_j \right), g^{\beta} \right)} = \frac{e\left(\left(h \cdot \prod_{i \in \mathbb{I}_{\mathbf{V}}} u_i^{ID_i} \right)^{\beta}, g^r \right)}{e\left(g^{\alpha}, g^{\beta}\right) \cdot e\left(h \cdot \left(\prod_{i \in \mathbb{I}_{\mathbf{V}}} u_i^{ID_i} \right)^{r}, g^{\beta} \right)} = \frac{1}{e\left(g,g\right)^{\alpha\beta}}$$

It follows that $C_2 \cdot e(C_1, a_1)/e(K, C_0) = M \cdot e(g,g)^{\alpha\beta}/e(g,g)^{\alpha\beta} = M$.

6.3.4.2 Security Analysis

The security of our scheme is guaranteed by the following Theorem. In a high level, the proof of our HIBBES follows the proof framework of Lewko-Waters HIBES [LW10], with an extra effort to generate ciphertexts for supporting broadcast.

Theorem 6.1 *Let $\mathbb{G}$ be a group (of composite order N) endowed with an efficient bilinear map. Our HIBBES is IND-CIVS-CPA secure if all the three assumptions defined in Definition 6.4, Definition 6.5 and Definition 6.6 hold in $\mathbb{G}$.*

To prove the chosen-identity-vector-set and chosen-plaintext security of our scheme, we apply the Dual System Encryption technique introduced by Waters [Wat09] for obtaining adaptively secure IBES and HIBES. This technique has been shown to be a powerful tool for security proofs [LW10; LW12]. In a Dual System Encryption system, the ciphertexts and keys can take one of two indistinguishable forms: normal form and semi-functional form. Normal keys can decrypt normal or semi-functional ciphertexts, and semi-functional ciphertexts can be decrypted by normal or semi-functional keys. However, decryption will fail when one uses a semi-functional key to decrypt a semi-functional ciphertext. Since these two

kinds of keys and ciphertexts are indistinguishable, the simulator can replace all normal ciphertexts and keys with semi-functional ones in the security game. When all ciphertexts and keys are semi-functional, $\mathcal{A}$ obtains no information about the challenge ciphertext as none of the given keys are useful to decrypt the challenge ciphertext.

We first need to define the semi-functional key and the semi-functional ciphertext. They will only be used in the security proof. Let $g_2 \xleftarrow{R} \mathbb{G}_{p_2}$ be a random generator of $\mathbb{G}_{p_2}$, the semi-functional ciphertext and the semi-functional key are defined as follows:

Semi-Functional Ciphertext. Run **Encrypt** to construct a normal ciphertext $CT = (C'_0, C'_1, C'_2)$. Then, choose random exponents $x, y_c \xleftarrow{R} \mathbb{Z}_N$ and set

$$C_0 = C'_0, C_1 = C'_1 g_2^{x y_c}, C_2 = C'_2 g_2^{x}$$

Semi-Functional Key. For an identity vector **ID**, run **KeyGen** to generate its normal secret key

$$SK = (a'_0, a'_1, \{b'_j\}_{j \in [1,n] \setminus \mathrm{I_{ID}}})$$

Then, choose random exponents $\gamma, y_k \in \mathbb{G}_N$, $z_j \in \mathbb{G}_N$ for all $j \in [1, n] \setminus \mathrm{I_{ID}}$ and set

$$a_0 = a'_0 g_2^{\gamma}, a_1 = a'_1 g_2^{\gamma y_k}, \{b_j = b'_j g_2^{\gamma z_j}\}_{j \in [1,n] \setminus \mathrm{I_{ID}}}$$

It can be seen that **Decrypt** will correctly output the message M when decrypting a semi-functional ciphertext using a normal key or a semi-functional key since the added elements in $\mathbb{G}_{p_2}$ will be cancelled due to the orthogonality property. However, the blinding factor will be multiplied by the additional term $e(g_2, g_2)^{x\gamma(y_k - y_c)}$ when trying to decrypt the semi-functional ciphertext using a semi-functional key, unless $y_k = y_c$ with probability $\frac{1}{N}$. In this case, we call the key a *nominally semi-functional* key. In addition, in the semi-functional secret key, the exponent y_k used for blinding the second component a_1 and the exponents z_j used for blinding the third component a_2 are chosen randomly and only appear at most twice in the security game. Therefore, from $\mathcal{A}$'s view the components in $\mathbb{G}_{p_2}$ for the semi-functional secret keys look random so that it does not helpful for $\mathcal{A}$ to distinguish the semi-functional secret key from a normal one, except with negligible probability $\frac{1}{N}$ when *nominally semi-functional* secret keys is coincidentally generated.

We prove security by using a sequence of games:

- — **Game$_{\mathbf{Real}}$**. This game is the real HIBBE security game.
- — **Game$_{\mathbf{Restricted}}$**. This game is identical with **Game$_{\mathbf{Real}}$**, except that in **Phase 2**, $\mathcal{A}$ cannot ask for any identity vectors **ID** $= (ID_1, \cdots, ID_d)$ satisfying that $\exists$**ID**$^* = (ID_1^*, \cdots, ID_{d'}^*) \in$ Pref($\mathbf{V}^*$) with $d' \leq d$, s.t. $\forall i \in [1, d'], ID_i = ID_i^* \bmod p_2$, where $\mathbf{V}^*$ is the challenge identity vector set.
- — **Game$_{\mathbf{k}}$**. Suppose that $\mathcal{A}$ can make q secret key queries in **Phase 1** and **Phase 2**. This game is identical with the **Game$_{\mathbf{Restricted}}$**, except that the challenge ciphertext is semi-functional and the first k keys are semi-functional, while the rest of the keys are normal. We note that in **Game$_{\mathbf{0}}$**, only the challenge ciphertext is semi-functional; in **Game$_{\mathbf{q}}$**, the challenge ciphertext and all secret keys are semi-functional.
- — **Game$_{\mathbf{Final}}$**. This game is the same as **Game$_q$**, except that the challenge ciphertext is a semi-functional encryption of a random message in $\mathbb{G}_T$, not one of the messages given by $\mathcal{A}$.

Given a security parameter λ, we represent the advantages of winning in the games **Game$_{\mathbf{Real}}$**, **Game$_{\mathbf{Restricted}}$**, **Game$_{\mathbf{k}}$** and **Game$_{\mathbf{Final}}$** by $Adv^{\mathrm{CPA}}_{\mathrm{Real}}(\lambda)$, $Adv^{\mathrm{CPA}}_{\mathrm{Restricted}}(\lambda)$, $Adv^{\mathrm{CPA}}_{k}(\lambda)$ and $Adv^{\mathrm{CPA}}_{\mathrm{Final}}(\lambda)$, respectively. We show that these games are indistinguishable in the following four lemmas.

Lemma 6.2 *Suppose Assumption 2 defined in Definition 6.5 holds. Then there is no polynomial time algorithm that can distinguish **Game$_{Real}$** from **Game$_{Restricted}$** with non-negligible advantage.*

Proof: If there exists an adversary $\mathcal{A}$ that can distinguish $\mathbf{Game_{Real}}$ from $\mathbf{Game_{Restricted}}$ with advantage ϵ_R, then by the definition of $\mathbf{Game_{Restricted}}$, $\mathcal{A}$ can issue a secret key query for the identity vector $\mathbf{ID} = (ID_1, \cdots, ID_d)$ from others satisfying that

$$\exists \mathbf{ID}^* = (ID_1^*, \cdots, ID_{d'}^*) \in \text{Pref}(\mathbf{V}^*) \text{ with } d' \leq d, \text{ s.t. } \forall i \in [1, d'], ID_i = ID_i^* \bmod p_2$$

Then a factor of N can be extracted by computing $\gcd(ID_i - ID_i^*, N)$, from which we can build a similar algorithm described in the proof of Lemma 5 in [LW10] that can refute the second assumption with advantage $Adv2_{\mathcal{B}}(\lambda) \geq \epsilon_R/2$. We omit the details here for avoiding repetition. □

Compared with $\mathbf{Game_{Restricted}}$, the challenge ciphertext is replaced with a semi-functional one in $\mathbf{Game_0}$. Since $\mathcal{A}$ does not know the factor of $N = p_1p_2p_3$, it cannot determine whether the components of the challenge ciphertext are in $\mathbb{G}_{p_1}$ or in $\mathbb{G}_{p_1p_2}$. Hence $\mathcal{A}$ is unable to know of which form the given challenge ciphertext is. This implies indistinguishability between $\mathbf{Game_{Restricted}}$ and $\mathbf{Game_0}$. Formally, we have the following Lemma.

Lemma 6.3 *Suppose Assumption 1 defined in Definition 6.4 holds. Then there is no polynomial time algorithm that can distinguish* $\mathbf{Game_{Restricted}}$ *from* $\mathbf{Game_0}$ *with non-negligible advantage.*

Proof: Suppose that there exists an adversary $\mathcal{A}$ that can distinguish $\mathbf{Game_{Restricted}}$ from $\mathbf{Game_0}$ with advantage ϵ_0. Then we can construct an algorithm $\mathcal{B}$ that can refute Assumption 1 with advantage $Adv1_{\mathcal{B}}(\lambda) \geq \epsilon_0$. The input of $\mathcal{B}$ is the challenge tuple (g, X_3, T) of Assumption 1. $\mathcal{B}$ needs to determine whether T is in $\mathbb{G}_{p_1}$ or in $\mathbb{G}_{p_1p_2}$. $\mathcal{B}$ sets the public key as follows. It randomly chooses $\alpha \overset{R}{\leftarrow} \mathbb{Z}_N$ and $\gamma_i \overset{R}{\leftarrow} \mathbb{Z}_N$ for all $i \in [0, n]$. Then, it sets $h \leftarrow g^{\gamma_0}$ and $u_i \leftarrow g^{\gamma_i}$ for all $i \in [1, n]$. Finally, $\mathcal{B}$ gives the public key $PK \leftarrow (g, h, u_1, \cdots, u_n, X_3, e(g, g)^{\alpha})$ to $\mathcal{A}$. It keeps the master key $MSK \leftarrow g^{\alpha}$ to itself.

Assume that $\mathcal{A}$ issues a secret key query for the identity vector $\mathbf{ID} = (ID_1, \cdots ID_d)$. $\mathcal{B}$ chooses random elements $r, w_0, w_1 \overset{R}{\leftarrow} \mathbb{Z}_N$ and $v_j \overset{R}{\leftarrow} \mathbb{Z}_N$ for all $j \in [1, n] \backslash \mathrm{I}$, where $\mathrm{I} = \{i : ID_i \in S_{\mathbf{ID}}\}$. Then $\mathcal{B}$ returns a normal key

$$SK_{\mathbf{ID}} \leftarrow \left(g^{\alpha} \left(h \cdot \prod_{i \in \mathrm{I}} u_i^{ID_i} \right)^r X_3^{w_0}, g^r X_3^{w_1}, \left\{ u_j^r X_3^{v_j} \right\}_{j \in [1,n] \backslash \mathrm{I}} \right)$$

When $\mathcal{A}$ decides that the **Challenge** phase starts, it outputs two equal-length messages $M_0, M_1 \in \mathbb{G}_T$, together with a challenge identity vector set $\mathbf{V}^*$. $\mathcal{B}$ flips a random coin $b \overset{R}{\leftarrow} \{0, 1\}$, and returns the challenge ciphertext

$$CT^* \leftarrow (C_0^*, C_1^*, C_2^*) \leftarrow \left(T, T^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i}, M_b \cdot e(g, T)^{\alpha} \right)$$

where $\mathbb{I}^* = \{i : ID_i^* \in S_{\mathbf{V}^*}\}$.

Finally, $\mathcal{A}$ outputs a guess that it is in $\mathbf{Game_{Restricted}}$ or in $\mathbf{Game_0}$. $\mathcal{B}$ guesses $T \in \mathbb{G}_{p_1}$ if $\mathcal{A}$ decides it is in $\mathbf{Game_{Restricted}}$. Otherwise, $\mathcal{B}$ outputs $T \in \mathbb{G}_{p_1p_2}$.

If $T \in \mathbb{G}_{p_1}$, this is a normal ciphertext by implicitly setting $T \leftarrow g^{\beta}$. Hence, $\mathcal{B}$ is simulating $\mathbf{Game_{Restricted}}$. Otherwise, if $T \in \mathbb{G}_{p_1p_2}$, all components in this ciphertext contain elements in subgroup $\mathbb{G}_{p_2}$, thus it is a semi-functional ciphertext. In this case, $\mathcal{B}$ is simulating $\mathbf{Game_0}$. If $\mathcal{A}$ has advantage ϵ_0 in distinguishing $\mathbf{Game_{Restricted}}$ from $\mathbf{Game_0}$, $\mathcal{B}$ can distinguish the distribution of T with advantage $Adv1_{\mathcal{B}}(\lambda) \geq \epsilon_0$. □

Similarly, $\mathbf{Game_{k\text{-}1}}$ and $\mathbf{Game_k}$ are two indistinguishable games. The way to determine whether the k^{th} queried key is normal or semi-functional is to determine whether the key components are in $\mathbb{G}_{p_1p_3}$ or in $\mathbb{G}_N$. However, this is computationally difficult without factoring $N = p_1p_2p_3$. Hence, we have the following Lemma.

Lemma 6.4 *Suppose Assumption 2 defined in Definition 6.5 holds. Then there is no polynomial time algorithm that can distinguish* $\mathbf{Game_{k\text{-}1}}$ *from* $\mathbf{Game_k}$ *with non-negligible advantage.*

Proof: Suppose that there exists an adversary $\mathcal{A}$ that can distinguish $\mathbf{Game_{k\text{-}1}}$ from $\mathbf{Game_k}$ with advantage ϵ_k. Then we can construct an algorithm $\mathcal{B}$ that can refute Assumption 2 with advantage $Adv2_{\mathcal{B}}(\lambda) \geq \epsilon_k$. The input of $\mathcal{B}$ is the challenge tuple $(g, X_1X_2, X_3, Y_2Y_3, T)$ of Assumption 2. $\mathcal{B}$ has to answer T is in $\mathbb{G}_N$ or in $\mathbb{G}_{p_1p_3}$.

$\mathcal{B}$ runs exactly the same as **Setup** in the proof of Lemma 6.3. The public key can be published as $PK \leftarrow (g, h, u_1, \cdots, u_n, X_3, e(g,g)^\alpha)$ with $g \leftarrow g$, $h \leftarrow g^{\gamma_0}$ and $u_i \leftarrow g^{\gamma_i}$ for all $i \in [1,n]$. The master key is $MSK \leftarrow g^\alpha$ that is kept secret to $\mathcal{B}$.

When receiving the ℓ^{th} secret key query for identity vector $\mathbf{ID} = (ID_1, \cdots ID_d)$ with $\ell < k$, $\mathcal{B}$ creates a semi-functional key to response to the query. Denote $\mathrm{I} = \{i : ID_i \in S_{\mathbf{ID}}\}$. $\mathcal{B}$ chooses random elements $r, w_0, w_1 \overset{R}{\leftarrow} \mathbb{Z}_N$ and $v_j \overset{R}{\leftarrow} \mathbb{Z}_N$ for all $j \in [1,n]\backslash\mathrm{I}$. Then it returns the secret key

$$SK_{\mathbf{ID}} \leftarrow \left(g^\alpha \left(h \cdot \prod_{i \in \mathrm{I}} u_i^{ID_i} \right)^r (Y_2Y_3)^{w_0}, g^r (Y_2Y_3)^{w_1}, \left\{ u_j^r (Y_2Y_3)^{v_j} \right\}_{j \in [1,n]\backslash \mathrm{I}} \right)$$

This is a well-formed semi-functional key obtained by implicitly setting $g_2^\gamma = Y_2^{w_0}$ and $y_k = w_1/w_0$.

If $\mathcal{A}$ issues the ℓ^{th} secret key query for $k < l \leq q$, $\mathcal{B}$ calls the usual key generation algorithm to generate a normal secret key and returns it to $\mathcal{A}$.

When $\mathcal{A}$ issues the k^{th} secret key query for identity vector $\mathbf{ID}$ with $\mathrm{I} = \{i : ID_i \in S_{\mathbf{ID}}\}$, $\mathcal{B}$ chooses random exponents $w_0 \overset{R}{\leftarrow} \mathbb{Z}_N$ and $v_j \overset{R}{\leftarrow} \mathbb{Z}_N$ for all $j \in [1,n]\backslash\mathrm{I}$. It then outputs

$$SK_{\mathbf{ID}} \leftarrow \left(g^\alpha T^{\gamma_0 + \sum_{i \in \mathrm{I}} ID_i \cdot \gamma_i} X_3^{w_0}, T, \left\{ T^{\gamma_j} X_3^{v_j} \right\}_{j \in [1,n]\backslash \mathrm{I}} \right)$$

If $T \in \mathbb{G}_{p_1p_3}$, then all components in this secret key are in $\mathbb{G}_{p_1p_3}$. Hence it is a normal secret key. Otherwise, it is a semi-functional key by implicitly setting $y_k = \gamma_0 + \sum_{i \in \mathrm{I}} ID_i \cdot \gamma_i$.

When $\mathcal{B}$ receives two equal-length messages $M_0, M_1 \in \mathbb{G}_T$ and a challenge identity vector set $\mathbf{V}^*$ from $\mathcal{A}$, it chooses a random bit $b \overset{R}{\leftarrow} \{0,1\}$ and returns

$$CT^* \leftarrow (C_0^*, C_1^*, C_2^*) \leftarrow \left(X_1X_2, (X_1X_2)^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i}, M_b \cdot e(g, X_1X_2)^\alpha \right)$$

to $\mathcal{A}$, where $\mathbb{I}^* = \{i : ID_i^* \in S_{\mathbf{V}^*}\}$.

Note that this ciphertext is semi-functional with $y_c = \gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i$. Since from $\mathbf{Game_{Restricted}}$, the identity vector associating with the k^{th} secret key is not a prefix of the challenge receiver identity vector set modulo p_2, y_c and y_k will seem randomly distributed to $\mathcal{A}$ so that the relationship between y_c and y_k offers no help for $\mathcal{A}$ to distinguish the two games.

Although hidden from $\mathcal{A}$, the relationship between y_c and y_k is important: it prevents $\mathcal{B}$ from testing whether the k^{th} secret key is semi-functional by generating a semi-functional ciphertext for any identity vector set $\mathbf{V}$ with $\mathbf{ID} \in \text{Pref}(\mathbf{V})$ and decrypts it using the k^{th} key. Indeed, $\mathcal{B}$ only can generate a *nominally* semi-functional key for the k^{th} key query for $\mathbf{ID}$. Note that $y_k + \sum_{i \in \mathbb{I}\backslash\mathrm{I}} ID_i \cdot \gamma_i = y_c$, where $\mathrm{I} = \{i : ID_i \in S_{\mathbf{ID}}\}$ and $\mathbb{I} = \{i : ID_i \in S_{\mathbf{V}}\}$. Hence, if $\mathcal{B}$ tries to do that, then decryption will always work, even when the k^{th} key is semi-functional. So, using this method, $\mathcal{B}$ cannot test whether the k^{th} key for identity vector $\mathbf{ID}$ is semi-functional or not without $\mathcal{A}$'s help. Note that this is the only case the *nominally* semi-functional secret key is used. For other queried secret keys, the exponents used in the subgroup $\mathbb{G}_{p_2}$ are randomly chosen so that the secret keys are randomly blinded by the elements in $\mathbb{G}_{p_2}$ and helpless for $\mathcal{A}$ to win the security game.

$\mathcal{B}$ finally outputs $T \in \mathbb{G}_{p_1p_3}$ if $\mathcal{A}$ outputs that it is in $\mathbf{Game_{k\text{-}1}}$, or outputs $T \in \mathbb{G}_N$ if $\mathcal{A}$ answers that it is in $\mathbf{Game_k}$.

If $T \in \mathbb{G}_{p_1p_3}$, all components in the k^{th} secret key generated by $\mathcal{B}$ are in $\mathbb{G}_{p_1p_3}$. Hence it is a normal secret key. In this case, $\mathcal{B}$ is simulating $\mathbf{Game_{k\text{-}1}}$. Otherwise, if $T \in \mathbb{G}_N$, then the k^{th} secret key is semi-functional.

In this case, $\mathcal{B}$ is simulating $\mathbf{Game_k}$. If $\mathcal{A}$ has advantage ϵ_k in distinguishing these two games, $\mathcal{B}$ can also distinguish $T \in \mathbb{G}_{p_1p_3}$ from $T \in \mathbb{G}_N$ with advantage $Adv_{\mathcal{B}}(\lambda) \geq \epsilon_k$. □

Lemma 6.5 *Suppose Assumption 3 defined in Definition 6.6 holds. Then there is no polynomial time algorithm that can distinguish* $\mathbf{Game_q}$ *from* $\mathbf{Game_{Final}}$ *with non-negligible advantage.*

Proof: Suppose that there exists an adversary $\mathcal{A}$ that can distinguish $\mathbf{Game_q}$ from $\mathbf{Game_{Final}}$ with advantage ϵ_F. By invoking $\mathcal{A}$ as a blackbox, we build an algorithm $\mathcal{B}$ refuting the third assumption with advantage $Adv3_{\mathcal{B}}(\lambda) \geq \epsilon_F$. $\mathcal{B}$ is given the challenge tuple $(g, g^{\alpha}X_2, X_3, g^sY_2, Z_2, T)$ and is required to answer whether T is $e(g,g)^{\alpha s}$ or a random element in $\mathbb{G}_T$. $\mathcal{B}$ randomly chooses $\gamma_i \xleftarrow{R} \mathbb{Z}_N$ for all $i \in [0, n]$ and sets the public key

$$PK \leftarrow (g = g, h = g^{\gamma_0}, u_1 = g^{\gamma_1}, \cdots, u_n = g^{\gamma_n}, X_3, e(g,g)^{\alpha} = e(g^{\alpha}X_2, g))$$

When $\mathcal{A}$ requests a secret key for an identity vector $\mathbf{ID}$, $\mathcal{B}$ chooses random exponents $w_0, w_1, t_0, t_1 \xleftarrow{R} \mathbb{Z}_N$ and $v_j, z_j \xleftarrow{R} \mathbb{Z}_N$ for all $j \in [1, n] \backslash \mathrm{I}$, where $\mathrm{I} = \{i : ID_i \in S_{\mathbf{ID}}\}$. It outputs

$$SK_{\mathbf{ID}} \leftarrow \left(g^{\alpha}X_2 \left(h \cdot \prod_{i \in \mathrm{I}} u_i^{ID_i} \right)^r Z_2^{t_0} X_3^{w_0}, g^r Z_2^{t_1} X_3^{w_1}, \left\{ u_j^r Z_2^{z_j} X_3^{v_j} \right\}_{j \in [1,n] \backslash \mathrm{I}} \right)$$

Note that this secret key is semi-functional with $g_2^{\gamma} = Z_2^{t_0}$ and $y_k = t_1/t_0$.

In the challenge phase, $\mathcal{A}$ outputs two equal-length messages $M_0, M_1 \in \mathbb{G}_T$, and a challenge identity vector set $\mathbf{V}^*$. Denote $\mathbb{I}^* = \{i : ID_i^* \in S_{\mathbf{V}^*}\}$. $\mathcal{B}$ chooses a random bit $b \xleftarrow{R} \{0,1\}$ and outputs the resulting semi-functional ciphertext

$$CT^* \leftarrow (C_0^*, C_1^*, C_2^*) \leftarrow \left(g^sY_2, (g^sY_2)^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i}, M_b \cdot T \right)$$

Eventually, if $\mathcal{A}$ guesses that it is in $\mathbf{Game_q}$, $\mathcal{B}$ outputs $T \leftarrow e(g,g)^{\alpha s}$. Otherwise, $\mathcal{B}$ outputs $T \xleftarrow{R} \mathbb{G}_T$ when $\mathcal{A}$ answers that it is in $\mathbf{Game_{Final}}$.

If $T \leftarrow e(g,g)^{\alpha s}$, then $\mathcal{B}$ is simulating $\mathbf{Game_q}$ since CT^* is a semi-functional ciphertext of the message M_b. If $T \xleftarrow{R} \mathbb{G}_T$, then CT^* is a semi-functional ciphertext of a random message that is independent of M_b. In this case, $\mathcal{B}$ is simulating $\mathbf{Game_{Final}}$. Hence, if $\mathcal{A}$ has advantage ϵ_F in distinguishing these two games, then $\mathcal{B}$ has advantage $Adv3_{\mathcal{B}}(\lambda) \geq \epsilon_F$ in distinguishing the distribution of T. □

Finally, since all keys and ciphertexts are semi-functional in $\mathbf{Game_q}$, $\mathcal{A}$ can get no information about the challenge ciphertext since none of the given keys are useful to decrypt it. Therefore, $\mathcal{A}$ cannot notice that the challenge ciphertext has been replaced by a random element. This implies the indistinguishability between $\mathbf{Game_q}$ and $\mathbf{Game_{Final}}$.

With the above lemmas, these security games are indistinguishable and in the final game the encrypted message is information-theoretically hidden from $\mathcal{A}$. Therefore, the proof of Theorem 6.1 follows.

Proof: If the three assumptions hold, then for all polynomial time adversaries $\mathcal{A}$, $Adv1_{\mathcal{A}}(\lambda)$, $Adv2_{\mathcal{A}}(\lambda)$, and $Adv3_{\mathcal{A}}(\lambda)$ are all negligible probability. In $\mathbf{Game_{Final}}$, the ciphertext has been replaced with a random element of $\mathbb{G}_T$. The value of b chosen by the challenger is information-theoretically hidden from $\mathcal{A}$. By applying the Lemma 6.2, Lemma 6.3, Lemma 6.4 and Lemma 6.5, we have that

$$\begin{aligned}
\left| Adv_{\text{Real}}^{\text{CPA}}(\lambda) \right| &\leq \left| Adv_{\text{Real}}^{\text{CPA}}(\lambda) - Adv_{\text{Restricted}}^{\text{CPA}}(\lambda) + Adv_{\text{Restricted}}^{\text{CPA}}(\lambda) - \cdots - Adv_{\text{Final}}^{\text{CPA}}(\lambda) + Adv_{\text{Final}}^{\text{CPA}}(\lambda) \right| \\
&\leq \left| Adv_{\text{Real}}^{\text{CPA}}(\lambda) - Adv_{\text{Restricted}}^{\text{CPA}}(\lambda) \right| + \cdots + \left| Adv_{q}^{\text{CPA}}(\lambda) - Adv_{\text{Final}}^{\text{CPA}}(\lambda) \right| + \left| Adv_{\text{Final}}^{\text{CPA}}(\lambda) \right| \\
&\leq \epsilon_R + \epsilon_0 + \cdots + \epsilon_q + \epsilon_F \leq Adv1_{\mathcal{A}}(\lambda) + (q+2) \cdot Adv2_{\mathcal{A}}(\lambda) + Adv3_{\mathcal{A}}(\lambda)
\end{aligned}$$

Therefore, there is no polynomial time adversary that can break our HIBBES with non-negligible advantage. This completes the proof of Theorem 6.1. □

6.3.5 Compact IND-CIVS-CCA2 HIBBE with Short Ciphertexts

6.3.5.1 Basic Ideas

In this section, we construct an IND-CIVS-CCA2 secure (D, n)-HIBBES from our IND-CIVS-CPA secure $(D, n+1)$-HIBBES. We first provide an overview of the conversion. We add one "dummy user" with an on-the-fly "identity" to the system. This dummy user is at depth 1, i.e., a child of the PKG. No one is allowed to obtain the secret key for the dummy user. It will be used just for the ciphertext validity test. When encrypting a message M, the encryption algorithm first creates the ciphertext components C_0 and C_2, which are independent of the receiver's identity vector set. Then, the algorithm hashes these two elements using a collision resistant hash function, and assigns it as the on-the-fly "identity" of the dummy user. Finally, we compute the ciphertext component C_1, as in the encryption algorithm of CPA-secure scheme. We show that there is an efficient algorithm to verify whether the resulting ciphertext is valid or not. In one word, the ciphertext validity test can be done publicly, since the test only involves the ciphertext CT and the public key PK.

This technique is inspired by the Boyen-Mei-Waters technique [BMW05], which applies to Waters' adaptively secure IBE [Wat05] and Boneh-Boyen selective-ID secure IBE [BB04] to obtain CCA2-secure public key cryptosystems. Boyen *et al.* remarked that their technique can be extended to achieve CCA2-secure HIBES from some CPA-secure HIBES by adding one extra hierarchy to the underlying HIBES. Instead of introducing one extra hierarchy of users to our HIBBE, we just add one extra dummy user at the first level by exploiting the broadcasting feature to enforce ciphertext validation test. In this way, CCA2 security is achieved only at a marginal cost of one extra user.

6.3.5.2 The Resulting Construction

For simple description, we label the previous HIBBES as $\text{HIBBE}_{\text{CPA}}$ with algorithms $\textbf{Setup}_{\text{CPA}}$, $\textbf{KeyGen}_{\text{CPA}}$, $\textbf{Delegate}_{\text{CPA}}$, $\textbf{Encrypt}_{\text{CPA}}$, and $\textbf{Decrypt}_{\text{CPA}}$. Our CCA2-secure HIBBES is denoted by $\text{HIBBE}_{\text{CCA2}}$. Similar to $\text{HIBBE}_{\text{CPA}}$, we assume that the identity vectors $\textbf{ID} = (ID_1, \cdots, ID_k)$ at depth k are vector elements in $(\mathbb{Z}_N)^k$, and messages to be encrypted are elements in $\mathbb{G}_T$. Our resulting scheme works as follows:

Setup(D, n, λ). The system first runs $\textbf{Setup}_{\text{CPA}}(D, n+1, \lambda)$ to generate the public key

$$PK \leftarrow (g, h, u_1, \cdots, u_n, u_{n+1}, X_3, e(g,g)^{\alpha})$$

and the master key $MSK \leftarrow g^{\alpha}$. A collision resistant hash function $H : \mathbb{G} \times \mathbb{G}_T \rightarrow \mathbb{Z}_N$ is also included in the public key. We stress that the dummy user, associated with parameter u_{n+1}, is at depth 1 and no one is allowed to obtain its corresponding secret key.

KeyGen and **Delegate**. These two algorithms are identical to $\textbf{KeyGen}_{\text{CPA}}$ and $\textbf{Delegate}_{\text{CPA}}$.

Encrypt$(PK, M, \textbf{V})$. For a receiver identity vector set **V**, denote $\mathbb{I} = \{i : ID_i \in S_{\textbf{V}}\}$. The encryption algorithm first picks a random $\beta \xleftarrow{R} \mathbb{Z}_N$ and computes

$$(C_0, C_2) \leftarrow \left(g^{\beta}, e(g,g)^{\alpha\beta} \cdot M\right)$$

Then, the algorithm computes $ID_{n+1} \leftarrow H(C_0, C_2) \in \mathbb{Z}_N$ and constructs C_1 as

$$C_1 \leftarrow \left(h \cdot u_{n+1}^{ID_{n+1}} \cdot \prod_{i \in \mathbb{I}} u_i^{ID_i}\right)^{\beta}$$

Finally, the algorithm outputs the ciphertext $CT \leftarrow (C_0, C_1, C_2)$. Note that this ciphertext is a valid $\text{HIBBE}_{\text{CPA}}$ ciphertext for the receiver identity vector set $\mathbf{V} \cup \{(ID_{n+1})\}$.

Decrypt($\mathbf{V}$, CT, $SK_{\mathbf{ID}}$). Suppose the secret key for the user associated with identity vector **ID** is

$$SK_{\mathbf{ID}} = \left(a_0, a_1, \{b_j\}_{j \in [1,n+1] \setminus \mathrm{I}}\right)$$

where $\mathrm{I} = \{i : ID_i \in S_{\mathbf{ID}}\}$. Denote $\mathbb{I} = \{i : ID_i \in S_{\mathbf{V}}\}$. Before decrypting the ciphertext $CT = (C_0, C_1, C_2)$, the decryption algorithm needs to first verify whether the ciphertext is legitimate. It does this by randomly choosing elements $Z_3, Z_3' \overset{R}{\leftarrow} \mathbb{G}_{p_3}$ computing $ID_{n+1} = H(C_0, C_2) \in \mathbb{Z}_N$ and testing whether the following equation holds:

$$e(g \cdot Z_3, C_1) \overset{?}{=} e\left(C_0, \left(h \cdot u_{n+1}^{ID_{n+1}} \cdot \prod_{i \in \mathbb{I}} u_i^{ID_i} \cdot Z_3'\right)\right) \tag{6.1}$$

If so, the decryption algorithm runs $\textbf{Decrypt}_{\text{CPA}}$ $(\mathbf{V} \cup \{(ID_{n+1})\}, CT, SK_{\mathbf{ID}})$ to get message M. Otherwise, the ciphertext is invalid and the decryption algorithm simply outputs $NULL$.

Remark Note that the above ciphertext validity test can be done publicly since it only involves public parameters and ciphertexts. This property is useful for our scheme to build advanced protocols, e.g., publicly verifiable HIBBE encryption with CCA2 security. Also, it allows a gateway or firewall to filter spams (i.e., invalid ciphertexts) without requiring the secret keys of the receivers. Similar functionality has been applied to identify dishonest transactions in mobile E-commerce scenario [HYH+16].

Soundness. If the ciphertext is legitimate, then the following tuple

$$\left(g, C_0 = g^{\beta}, \left(h \cdot u_{n+1}^{ID_{n+1}} \cdot \prod_{i \in \mathbb{I}} u_i^{ID_i}\right), C_2 = \left(h \cdot u_{n+1}^{ID_{n+1}} \cdot \prod_{i \in \mathbb{I}} u_i^{ID_i}\right)^{\beta}\right)$$

is a valid Diffie-Hellman tuple. Note that elements $Z_3, Z_3' \in \mathbb{G}_{p_3}$ can be eliminated in both sides of Equation (6.1) with the orthogonality property. Accordingly, Equation (6.1) holds. Also, this ciphertext is a valid $\text{HIBBE}_{\text{CPA}}$ ciphertext for the receiver identity vector set $\mathbf{V} \cup \{(ID_{n+1})\}$ with $ID_{n+1} = H(C_0, C_2)$. Since $\mathbf{ID} \in \text{Pref}(\mathbf{V}) \subseteq \mathbf{V} \cup \{(ID_{n+1})\}$, the decryption algorithm can decrypt the ciphertext by invoking the underlying $\textbf{Decrypt}_{\text{CPA}}(\mathbf{V} \cup \{(ID_{n+1})\}, CT, SK_{\mathbf{ID}})$.

6.3.5.3 Security Analysis

We now allow decryption queries in all games defined previously in Section 6.3.4.2. Our simulation works as follows. When receiving a decryption query from the adversary, the simulator first checks Equation (6.1) to determine whether the ciphertext is valid. If the equality holds, the simulator generates a secret key for any identity vector **ID** satisfying that $\mathbf{ID} \in \text{Pref}(\mathbf{V})$, and then uses this key to decrypt the ciphertext. In the challenge phase, the simulator creates a challenge ciphertext $CT^* = (C_0^*, C_1^*, C_2^*)$ for the challenge identity vector set $\mathbf{V}^* \cup \{(ID_{n+1}^*)\}$, where $ID_{n+1}^* = H(C_0^*, C_2^*)$. Since the hash function H is collision resistant, the adversary is unable to make any valid ciphertext queries that would require the simulator to use a identity vector set $\mathbf{V} \cup \{(ID_{n+1}')\}$ with $ID_{n+1}' = ID_{n+1}^*$. Note that the adversary cannot issue secret key query for the dummy user because the dummy user is not available before the simulator produces the challenge ciphertext. Hence, the simulation can be done by invoking the underlying $\text{HIBBE}_{\text{CPA}}$.

Formally, the CCA2 security of the above scheme is guaranteed by the following Theorem.

Theorem 6.6 *Let $\mathbb{G}$ be a group (of composite order N) endowed with an efficient bilinear map. Suppose that all the three assumptions defined in Definition 6.4, Definition 6.5 and Definition 6.6 hold in $\mathbb{G}$. Then our $\text{HIBBE}_{\text{CCA2}}$ is IND-CIVS-CCA2 secure.*

Similarly to those in CPA security proofs, we denote these games by **GameCCA2$_{\text{Real}}$**, **GameCCA2$_{\text{Restricted}}$**, **GameCCA2$_{\text{k}}$** with $k \in [0, q]$ and **GameCCA2$_{\text{Final}}$** respectively. For a security parameter λ, we represent the advantages of winning in these games by $Adv^{\text{CCA2}}_{\text{Real}}(\lambda)$, $Adv^{\text{CCA2}}_{\text{Restricted}}(\lambda)$, $Adv^{\text{CCA2}}_{k}(\lambda)$ with $k \in [0, q]$, and $Adv^{\text{CCA2}}_{\text{Final}}(\lambda)$ respectively. The security of our HIBBE$_{\text{CCA2}}$ follows from the indistinguishability between the these games, assuming that the three assumptions defined in Section 6.3.2 hold.

Lemma 6.7 *Suppose that Assumption 2 holds. Then there is no polynomial time algorithm that can distinguish* ***GameCCA2$_{Real}$*** *from* ***GameCCA2$_{Restricted}$*** *with non-negligible advantage.*

Proof: The proof of this lemma is identical with the proof of lemma 6.2. □

Lemma 6.8 *There is no polynomial time algorithm that can distinguish* ***GameCCA2$_{Restricted}$*** *from* ***GameCCA2$_0$*** *with non-negligible advantage assuming that Assumption 1 holds.*

Proof: Assume that there exists an adversary $\mathcal{A}$ that can distinguish **GameCCA2$_{\text{Restricted}}$** from **GameCCA2$_0$** with advantage ϵ_0. We build an algorithm $\mathcal{B}$ that can refute Assumption 1 with advantage $Adv1_{\mathcal{B}}(\lambda) \geq \epsilon_0$. $\mathcal{B}$ takes the challenge tuple (g, X_3, T) as inputs. The goal of $\mathcal{B}$ is to determine whether T is an element in $\mathbb{G}_{p_1}$ or an element in $\mathbb{G}_{p_1 p_2}$. In the Setup phase, $\mathcal{B}$ randomly chooses exponents $\alpha \overset{R}{\leftarrow} \mathbb{Z}_N$ and $\gamma_i \overset{R}{\leftarrow} \mathbb{Z}_N$ for all $i \in [0, n+1]$. It sets $h \leftarrow g^{\gamma_0}$ and $u_i \leftarrow g^{\gamma_i}$ for all $i \in [1, n+1]$. Finally, $\mathcal{B}$ gives the public key

$$PK \leftarrow (g, h, u_1, \cdots, u_n, u_{n+1}, X_3, e(g,g)^{\alpha})$$

to $\mathcal{A}$. Note that $\mathcal{B}$ knows the master key $MSK \leftarrow g^{\alpha}$.

For a secret key query with identity vector $\mathbf{ID} = (ID_1, \cdots, ID_d)$ issued by $\mathcal{A}$, $\mathcal{B}$ runs the usual key generation algorithm to return the secret key.

When receiving a decryption query from $\mathcal{A}$ with a ciphertext $CT = (C_0, C_1, C_2)$ and a receiver identity vector set $\mathbf{V}$, $\mathcal{B}$ first computes $ID_{n+1} = H(C_0, C_2)$ and determines whether the ciphertext is valid by checking Equation (6.1) defined in Section 6.3.5.2. If the equality does not hold, then the ciphertext is invalid and $\mathcal{B}$ returns $NULL$. Otherwise, $\mathcal{B}$ generates a normal key for any user whose identity vector is $\mathbf{ID} \in \text{Pref}(\mathbf{V})$ using the master key g^{α}. Then, $\mathcal{B}$ uses this key to decrypt the ciphertext and returns the extracted message to $\mathcal{A}$.

In the challenge phase, $\mathcal{A}$ outputs two equal-length messages $M_0, M_1 \in \mathbb{G}_T$, together with a challenge identity vector set $\mathbf{V}^*$. Denote $\mathbb{I}^* = \{i : ID_i^* \in S_{\mathbf{V}^*}\}$. $\mathcal{B}$ flips a random coin $b \overset{R}{\leftarrow} \{0,1\}$ and returns the challenge ciphertext

$$CT^* \leftarrow (C_0^*, C_1^*, C_2^*) \leftarrow \left(T, T^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i + ID_{n+1}^* \cdot \gamma_{n+1}}, M_b \cdot e(g^{\alpha}, T)\right)$$

where $ID_{n+1}^* = H(C_0^*, C_2^*) = H(T, M_b \cdot e(g^{\alpha}, T))$.

Note that the components in the challenge ciphertext do not involve elements in $\mathbb{G}_{p_3}$. Therefore, for any randomly chosen elements $Z_3, Z_3' \overset{R}{\leftarrow} \mathbb{G}_{p_3}$, the challenge ciphertext is valid due to the following equalities:

$$\frac{e(g \cdot Z_3, C_1^*)}{e\left(C_0^*, \left(h \cdot u_{n+1}^{ID_{n+1}^*} \cdot \prod_{i \in \mathbb{I}^*} u_i^{ID_i^*}\right) \cdot Z_3'\right)} = \frac{e(g \cdot Z_3, T^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i + ID_{n+1}^* \cdot \gamma_{n+1}})}{e\left(T, g^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i + ID_{n+1}^* \cdot \gamma_{n+1}} \cdot Z_3'\right)} = 1$$

Finally, $\mathcal{A}$ outputs a guess of whether $\mathcal{A}$ is in **GameCCA2$_{\text{Restricted}}$** or in **GameCCA2$_0$**. If $\mathcal{A}$ guesses that $\mathcal{A}$ is in **GameCCA2$_{\text{Restricted}}$**, $\mathcal{B}$ outputs $T \in \mathbb{G}_{p_1}$. Otherwise, $\mathcal{B}$ concludes $T \in \mathbb{G}_{p_1 p_2}$.

The decryption query can be responded to perfectly, since $\mathcal{B}$ can generate normal keys for arbitrary identity vectors using the master key g^{α}. With the identical analysis showed in the proof of Lemma 6.2,

if $\mathcal{A}$ has advantage ϵ_0 in distinguishing **GameCCA2$_{\text{Restricted}}$** and **GameCCA2$_0$**, then $\mathcal{B}$ can determine the distribution of T with advantage $Adv1_{\mathcal{B}}(\lambda) \geq \epsilon_0$. □

Lemma 6.9 *If Assumption 2 holds, then there is no polynomial time algorithm that can distinguish **GameCCA2$_{k-1}$** from **GameCCA2$_k$** with non-negligible advantage.*

Proof: Assume an adversary $\mathcal{A}$ that can distinguish **GameCCA2$_{k-1}$** from **GameCCA2$_k$** with advantage ϵ_k. Then, by invoking $\mathcal{A}$ as a blackbox, we can construct an algorithm $\mathcal{B}$ that refutes Assumption 2 with advantage $Adv2_{\mathcal{B}}(\lambda) \geq \epsilon_k$. The input of $\mathcal{B}$ is an instance $(g, X_1X_2, X_3, Y_2Y_3, T)$ from the second assumption. $\mathcal{B}$ has to decide whether T is an element in $\mathbb{G}_N$ or an element in $\mathbb{G}_{p_1p_3}$. $\mathcal{B}$ randomly chooses $\alpha \overset{R}{\leftarrow} \mathbb{Z}_N$ and $\gamma_i \overset{R}{\leftarrow} \mathbb{Z}_N$ for all $i \in [1, n+1]$. It sends $\mathcal{A}$ the public key

$$PK \leftarrow (g, h, u_1, \cdots, u_n, u_{n+1}, X_3, e(g,g)^{\alpha})$$

with $h \leftarrow g^{\gamma_0}$ and $u_i \leftarrow g^{\gamma_i}$ for all $i \in [1, n+1]$. The master key is $MSK \leftarrow g^{\alpha}$ and is kept by $\mathcal{B}$.

When receiving the secret key query with an identity vector **ID** $= (ID_1, \cdots, ID_d)$, $\mathcal{B}$ runs the same as **Phase 1** in Lemma 6.4 to generate the secret key and returns it to $\mathcal{A}$.

When $\mathcal{A}$ issues a decryption query for a ciphertext $CT = (C_0, C_1, C_2)$ with a receiver identity vector set **V**, $\mathcal{B}$ sets $ID_{n+1} = H(C_0, C_2)$ and checks Equation (6.1) described in Section 6.3.5.2. If the equality holds, $\mathcal{B}$ creates a normal key for any identity vector **ID** $\in$ Pref(**V**) and returns the message decrypted from the ciphertext CT. Otherwise it returns $NULL$ since the ciphertext is invalid.

In the **Challenge** phase, $\mathcal{A}$ outputs two equal-length messages $M_0, M_1 \in \mathbb{G}_T$, together with an identity vector set $\mathbf{V}^*$ as the challenge identity vector set. Denote $\mathbb{I}^* = \{i : ID_i^* \in S_{\mathbf{V}^*}\}$. $\mathcal{B}$ chooses a random bit $b \overset{R}{\leftarrow} \{0,1\}$ and outputs the resulting ciphertext

$$CT^* \leftarrow (C_0^*, C_1^*, C_2^*) \leftarrow \left(X_1X_2, (X_1X_2)^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i + ID_{n+1}^* \cdot \gamma_{n+1}}, M_b \cdot e(g, X_1X_2)^{\alpha}\right)$$

where $ID_{n+1}^* = H(C_0^*, C_2^*) = H\left(X_1X_2, e(g, X_1X_2)^{\alpha}\right)$. Equation (6.1) holds for this ciphertext since for any $Z_3, Z_3' \overset{R}{\leftarrow} \mathbb{G}_{p_3}$,

$$\frac{e(g \cdot Z_3, C_1^*)}{e\left(C_0^*, \left(h \cdot u_{n+1}^{ID_{n+1}^*} \cdot \prod_{i \in \mathbb{I}^*} u_i^{ID_i^*}\right) \cdot Z_3'\right)} = \frac{e\left(g \cdot Z_3, (X_1X_2)^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i + ID_{n+1}^* \cdot \gamma_{n+1}}\right)}{e\left(X_1X_2, g^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i + ID_{n+1}^* \cdot \gamma_{n+1}} \cdot Z_3'\right)} = 1$$

Therefore, this ciphertext is valid.

Note that this ciphertext is semi-functional by implicitly setting

$$y_c = \gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i + ID_{n+1}^* \cdot \gamma_{n+1}$$

Since from **GameCCA2$_{\text{Restricted}}$**, $\mathcal{A}$ cannot issue a secret key query with the identity vector that is a prefix of the challenge receiver identity vector set module p_2, y_c and y_k will seem randomly distribute to $\mathcal{A}$. Therefore, the relationship between y_c and y_k does not give any advantage to $\mathcal{A}$ for distinguishing between the two games.

Though the relationship between y_c and y_k is hidden from $\mathcal{A}$, this special setting disallows $\mathcal{B}$ *itself* to test whether the k^{th} key for identity vector **ID** is semi-functional. The method is to generate a semi-functional ciphertext for any identity vector set **V** such that **ID** $\in$ Pref(**V**) and to decrypt it using the k^{th} key. If the k^{th} key is normal, the decryption is correct. However, if the k^{th} key is semi-functional, then by the definition of semi-functional secret key, the k^{th} key cannot decrypt the semi-functional ciphertext. In this way, $\mathcal{B}$ may have advantage 1 to answer $T \in \mathbb{G}_N$ or $T \in \mathbb{G}_{p_1p_2p_3}$ without $\mathcal{A}$'s help.

In fact, this well-designed secret key generated in the k^{th} key query disallows $\mathcal{B}$ to use this method. If $\mathcal{B}$ tries to do that, then no matter whether the k^{th} key is normal or semi-functional, decryption will always work, because $y_k + \sum_{i \in \mathbb{I} \setminus \mathrm{I}} ID_i \cdot \gamma_i + ID_{n+1} \cdot \gamma_{n+1} = y_c$, where $\mathrm{I} = \{i : ID_i \in S_{\mathbf{ID}}\}$ and $\mathbb{I} = \{i : ID_i \in S_{\mathbf{V}}\}$. In other words, for the k^{th} secret key query, $\mathcal{B}$ can only generate a *nominally* semi-functional key. Hence decryption is always correct by the definition of nominally semi-functional key given in Section 6.3.4.2.

Finally, if $\mathcal{A}$ outputs the guess that it is in **GameCCA2$_{\mathbf{k\text{-}1}}$**, $\mathcal{B}$ answers $T \in \mathbb{G}_{p_1 p_3}$. Otherwise, $\mathcal{A}$ outputs that it is in **GameCCA2$_{\mathbf{k}}$**, and $\mathcal{B}$ decides $T \in \mathbb{G}_N$.

With the similar reason in the proof of Lemma 6.4, if $\mathcal{A}$ has advantage ϵ_k in distinguishing **GameCCA2$_{\mathbf{k\text{-}1}}$** from **GameCCA2$_{\mathbf{k}}$**, $\mathcal{B}$ can distinguish $T \in \mathbb{G}_{p_1 p_3}$ from $T \in \mathbb{G}_N$ with advantage $Adv2_{\mathcal{B}}(\lambda) \geq \epsilon_k$. □

Lemma 6.10 *Suppose that Assumption 3 holds. Then there is no polynomial time algorithm that can distinguish* ***GameCCA2$_q$*** *from* ***GameCCA2$_{\text{Final}}$*** *with non-negligible advantage.*

Proof: Let $\mathcal{A}$ be an algorithm that can distinguish **GameCCA2$_{\mathbf{q}}$** from **GameCCA2$_{\mathbf{Final}}$** with advantage ϵ_F. By invoking $\mathcal{A}$ as a blackbox, we build an algorithm $\mathcal{B}$ refuting Assumption 3 with advantage $Adv3_{\mathcal{B}}(\lambda) \geq \epsilon_F$. The input of $\mathcal{B}$ is the challenge tuple $(g, g^{\alpha} X_2, X_3, g^s Y_2, Z_2, T)$ of Assumption 3. $\mathcal{B}$ has to answer whether T is $e(g,g)^{\alpha s}$ or a random element in $\mathbb{G}_T$. $\mathcal{B}$ randomly chooses $\gamma_i \xleftarrow{R} \mathbb{Z}_N$ for all $i \in [0, n+1]$ and sets the public key

$$PK \leftarrow (g = g, h = g^{\gamma_0}, u_1 = g^{\gamma_1}, \cdots, u_n = g^{\gamma_n}, u_{n+1} = g^{\gamma_{n+1}}, X_3, e(g,g)^{\alpha} = e(g^{\alpha} X_2, g))$$

When $\mathcal{A}$ requests a secret key for an identity vector **ID**, $\mathcal{B}$ chooses random exponents $w_0, w_1, t_0, t_1 \xleftarrow{R} \mathbb{Z}_N$ and $v_j, z_j \xleftarrow{R} \mathbb{Z}_N$ for all $j \in [1,n] \setminus \mathrm{I}$, where $\mathrm{I} = \{i : ID_i \in S_{\mathbf{ID}}\}$. Then, $\mathcal{B}$ outputs the secret key

$$SK_{\mathbf{ID}} \leftarrow \left(g^{\alpha} X_2 \left(h \cdot \prod_{i \in \mathrm{I}} u_i^{ID_i} \right)^{r} Z_2^{t_0} X_3^{w_0}, g^r Z_2^{t_1} X_3^{w_1}, \left\{ u_j^r Z_2^{z_j} X_3^{v_j} \right\}_{j \in [1,n] \setminus \mathrm{I}} \right)$$

Note that the resulting key is semi-functional.

When $\mathcal{B}$ receives a decryption query for a ciphertext $CT = (C_0, C_1, C_2)$ associated with a receiver identity vector set **V**, it first sets $ID_{n+1} = H(C_0, C_2)$. Then, $\mathcal{B}$ checks Equation (6.1) of Section 6.3.5.2 to verify the validity of CT. If the equality does not hold, $\mathcal{B}$ simply returns $NULL$. Otherwise, since $\mathcal{B}$ knows a random generator g of $\mathbb{G}_{p_1}$ and a random element $X_3 \in \mathbb{G}_{p_3}$, it can run the same algorithm described in **Phase 1** to generate a semi-functional secret key for **ID** $\in$ Pref(**V**) and use it to decrypt CT.

Although the generated secret keys are all semi-functional, $\mathcal{B}$ can use them to correctly respond the decryption queries. The reason is that $\mathcal{A}$ can only issue valid normal ciphertexts for decryption queries. One one hand, $\mathcal{A}$ cannot generate semi-functional ciphertexts for any identity vector sets **V** without the knowledge of the subgroup $\mathbb{G}_{p_2}$, except for the challenge identity vector set. Otherwise $\mathcal{A}$ can distinguish the preceding security games by issuing a secret key query for an identity vector **ID** $\in$ Pref(**V**) and try to decrypt by itself. This has been prevented in the CPA security proof. On the other hand, only semi-functional ciphertexts that can be obtained by $\mathcal{A}$ are the ones modified from the challenge ciphertext. However, any modifications done by $\mathcal{A}$ without the knowledge of the subgroup $\mathbb{G}_{p_2}$ for the challenge ciphertext can be detected by Equation (6.1). Therefore, any decryption queries for semi-functional ciphertexts would be prevented. The secret keys would only be used to decrypt normal ciphertexts and the decryption queries can be responded correctly.

When suitable, $\mathcal{A}$ outputs two equal-length messages $M_0, M_1 \in \mathbb{G}_T$, and a challenge identity vector set $\mathbf{V}^*$. Denote $\mathbb{I}^* = \{i : ID_i \in S_{\mathbf{V}^*}\}$. $\mathcal{B}$ chooses a random bit $b \xleftarrow{R} \{0,1\}$ and outputs the challenge ciphertext

$$CT^* \leftarrow (C_0^*, C_1^*, C_2^*) \leftarrow \left(g^s Y_2, (g^s Y_2)^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID_i^* \cdot \gamma_i + ID_{n+1}^* \cdot \gamma_{n+1}}, M_b \cdot T \right)$$

where $ID^*_{n+1} = H(C^*_0, C^*_2) = H\left(g^s Y_2, M_b \cdot T\right)$. Note that for any $Z_3, Z'_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$,

$$\frac{e(g \cdot Z_3, C^*_1)}{e\left(C^*_0, \left(h \cdot u_{n+1}^{ID^*_{n+1}} \cdot \prod_{i \in \mathbb{I}^*} u_i^{ID^*_i} \cdot Z'_3\right)\right)} = \frac{e\left(g \cdot Z_3, (g^s Y_2)^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID^*_i \cdot \gamma_i + ID^*_{n+1} \cdot \gamma_{n+1}}\right)}{e\left(g^s Y_2, g^{\gamma_0 + \sum_{i \in \mathbb{I}^*} ID^*_i \cdot \gamma_i + ID^*_{n+1} \cdot \gamma_{n+1}} \cdot Z'_3\right)} = 1$$

Hence CT^* is a valid ciphertext.

Finally, $\mathcal{B}$ answers $T \leftarrow e(g,g)^{\alpha s}$ if $\mathcal{A}$ outputs the guess that it is in **GameCCA2$_{\mathbf{q}}$**. Otherwise, $\mathcal{B}$ determines $T \overset{R}{\leftarrow} \mathbb{G}_T$ if $\mathcal{A}$ guesses that it is in **GameCCA2$_{\mathbf{Final}}$**.

Similar to the analysis of Lemma 6.5, $\mathcal{B}$ can distinguish $T \leftarrow e(g,g)^{\alpha s}$ from a random element in $\mathbb{G}_T$ with advantage $Adv3_{\mathcal{B}}(\lambda) \geq \epsilon_F$ if $\mathcal{A}$ has advantage ϵ_F in distinguishing **GameCCA2$_{\mathbf{q}}$** from **GameCCA2$_{\mathbf{Final}}$**. □

With the four lemmas described above, the security proof of Theorem 6.6 follows.

Proof: Since in **GameCCA2$_{\mathbf{Final}}$**, the ciphertext has been replaced with a random element in $\mathbb{G}_T$, the value of b chosen by the challenger is information-theoretically hidden from $\mathcal{A}$. Hence $\mathcal{A}$ can obtain no advantage in breaking our HIBBES. By combining the four lemmas shown previously, we have that

$$\begin{aligned}
\left|Adv^{\text{CCA2}}_{\text{Real}}(\lambda)\right| &\leq \left|Adv^{\text{CCA2}}_{\text{Real}}(\lambda) - Adv^{\text{CCA2}}_{\text{Restricted}}(\lambda) + Adv^{\text{CCA2}}_{\text{Restricted}}(\lambda) - \cdots - Adv^{\text{CCA2}}_{\text{Final}}(\lambda) + Adv^{\text{CCA2}}_{\text{Final}}(\lambda)\right| \\
&\leq \left|Adv^{\text{CCA2}}_{\text{Real}}(\lambda) - Adv^{\text{CCA2}}_{\text{Restricted}}(\lambda)\right| + \cdots + \left|Adv^{\text{CCA2}}_{q}(\lambda) - Adv^{\text{CCA2}}_{\text{Final}}(\lambda)\right| + \left|Adv^{\text{CCA2}}_{\text{Final}}(\lambda)\right| \\
&\leq \epsilon_R + \epsilon_0 + \cdots + \epsilon_q + \epsilon_F \leq Adv1_{\mathcal{A}}(\lambda) + (q+2) \cdot Adv2_{\mathcal{A}}(\lambda) + Adv3_{\mathcal{A}}(\lambda)
\end{aligned}$$

If the three assumptions hold, then for all polynomial time $\mathcal{A}$, $Adv1_{\mathcal{A}}(\lambda)$, $Adv2_{\mathcal{A}}(\lambda)$, and $Adv3_{\mathcal{A}}(\lambda)$ are all negligible probability. Hence for all polynomial time algorithms, the advantage of breaking our HIBBE$_{\text{CCA2}}$ is negligible. □

6.3.5.4 Efficient Tradeoff Between Ciphertext Size and Key Size

The public/secret key size and ciphertext size in (D, n)-HIBBE$_{\text{CCA2}}$ remain the same as those of the underlying $(D, n+1)$-HIBBE$_{\text{CPA}}$ system. The encryption algorithm needs only one more hash operation. The decryption algorithm does one more hash operation and one more extra test of Equation (6.1) in which a two-base pairing is required and $u_i^{ID_i}$ can be pre-computed for $i \in [1, n]$. Table 6.3 shows comparisons between our CPA-secure $(D, n+1)$-HIBBE and our CCA2-secure (D, n)-HIBBE in detail. In Table 6.3, the secret key $SK_{\mathbf{ID}}$ is associated with the identity vector **ID**, and the ciphertext CT is associated with the receiver identity vector set **V**. We denote τ_e as one exponent operation time in $\mathbb{G}$, τ_m as one multiplication operation time in $\mathbb{G}$, τ_p as one pairing operation time in $\mathbb{G}$, and τ_h as one hash operation time for the hash function H. From Table 6.3, it can be seen that the additional overheads are marginal.

Table 6.3 – Comparison Between CPA-secure $(D, n+1)$-HIBBE and CCA2-secure (D, n)-HIBBE

	$(D, n+1)$-HIBBE$_{\text{CPA}}$	(D, n)-HIBBE$_{\text{CCA2}}$
Active Users	$n+1$	n
PK Size	$n+5$	$n+5$
$SK_{\mathbf{ID}}$ Size	$n - \|\mathbf{ID}\| + 2$	$n - \|\mathbf{ID}\| + 2$
CT Size	3	3
Encryption Time	$(2 + \|S_{\mathbf{V}}\|) \cdot (\tau_e + \tau_m)$	$(2 + \|S_{\mathbf{V}}\|) \cdot (\tau_e + \tau_m) + \tau_h$
Decryption Time	$\leq (1 + \|S_{\mathbf{V}}\|) \cdot (\tau_e + \tau_m) + 2\tau_p$	$\leq (1 + \|S_{\mathbf{V}}\|) \cdot (\tau_e + \tau_m) + 4\tau_p + \tau_h$

HIBBE with Shorter Secret Keys. In our HIBBES, while the ciphertext contains only three group elements, the secret key for user at depth d contains $n - d + 2$ elements. In some scenarios, e.g., when the storage capacities of the receivers are limited, one may expect an efficient tradeoff between key size and ciphertext size. Note that users in an HIBBES are organized as a tree T with n nodes (PKG as the sink is not countered). We divide T into $\mathcal{T}$ subtrees with n_i nodes, where $i \in [1, \mathcal{T}]$. To achieve better balance, as shown in Figure 6.19, all the subtrees may be obtained in a way satisfying:

1. The number of nodes for each subtree is approximately equal. That is, for the i^{th} subtree with $i \in [1, \mathcal{T}]$, we have $n_i \approx n/\mathcal{T}$;
2. If possible, all subtrees share minimum number of higher-level nodes.

We then implement independent HIBBE instances in each subtree. When broadcasting, one encrypts the messages with each instance where the broadcast subsets are the intersection of the original broadcast set and the subtrees. Each receiver can decrypt the ciphertext component corresponding to its subtree. It is clear that, by using this subtree method, the key size is $O(\frac{n}{\mathcal{T}})$ and the ciphertext size is $O(\mathcal{T})$. By setting $\mathcal{T} = \sqrt{n}$, both the key size and the ciphertext size are $O(\sqrt{n})$.

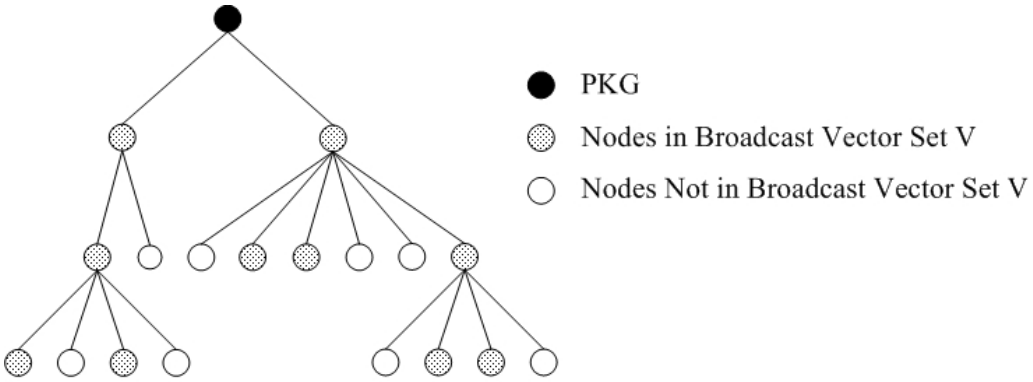


Figure 6.18 – Constant Size Ciphertext HIBBE.

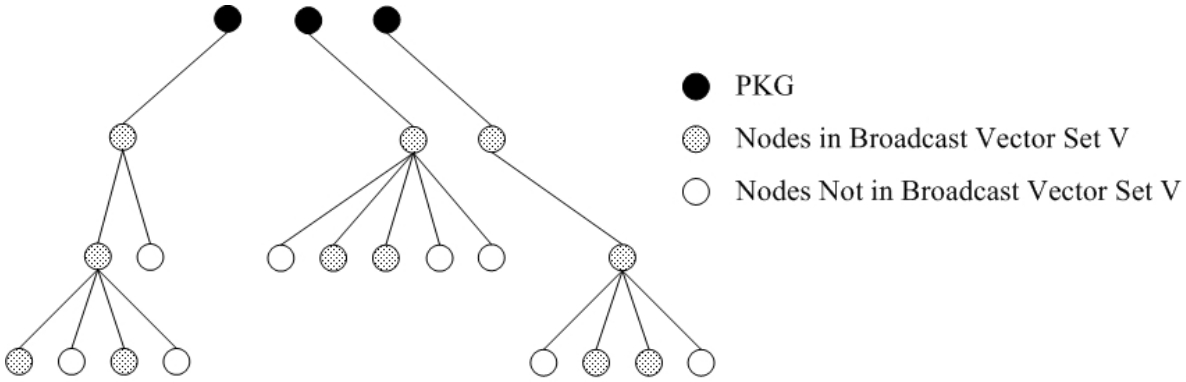


Figure 6.19 – Shorter Secret keys HIBBE.

6.3.6 Conclusion

This paper extended the functionality of HIBE to HIBBE, allowing users to encrypt to multiple receivers organized in hierarchy. The new cryptographic primitive offers a novel avenue to establish secure broadcast systems for distributed computation and communication applications. We constructed a chosen-plaintext secure HIBBES with short ciphertexts. We then proposed a transformation technique to convert our basic scheme to obtain chosen-ciphertext security. An interesting line of future research is to apply the conversion technologies to more general cryptosystems such as attribute based encryption, predicate encryption and functional encryption.

6.4 Improved Delayed Decryption for Software Patching

6.4.1 Introduction

In a little noticed yet ingenious patent [TW06], Thomlinson and Walker describe a very original software patching system. Thomlinson and Walker describe their invention as follows:

"...Computer programs are complex systems, and they typically have vulnerabilities that are not discovered until after the software is released. These vulnerabilities can be addressed after the initial software is released by distributing and installing an update to the software, which is designed to remedy, or protect against, the vulnerability. Typically, the vulnerability is discovered by the program's manufacturer, support entity, or partner before the vulnerability is generally known to the public.

One problem with an update is that the update can normally be reverse engineered to reveal the existence of the vulnerability that the update is attempting to fix, which can be an invitation to attackers to try to exploit the vulnerability on machines without the fix applied. If updates could be delivered to every machine at the same time, then the fact that the updates reveals the vulnerability would not be a significant problem, since all machines would be protected against the vulnerability at the same time that attackers learned of the vulnerability's existence. However, updates often take the form of large files, and there is not sufficient bandwidth, or other physical resources, to distribute the update to every machine at the same time. Thus, there is a window of time during which the update (and the vulnerability that it both fixes and reveals) is known to the public, but a significant number of machines are unprotected. It is desirable to update programs in such a manner that all, or a large number, of machines are protected very soon after the update is first made known to the public.

Updates can be provided in an encrypted form, such that being able to use the update (or to read it for reverse engineering purposes) requires a decryption key. The key can then be delivered after certain conditions have been met – e.g., only after the encrypted update has been delivered to a sufficient number of machines to ensure widespread protection, and/or after the update has undergone sufficient testing to ensure that it effectively remedies the vulnerability that it is designed to address. Since the key is small the key can be delivered to a large number of machines in a relatively short amount of time, as compared with how long it takes to distribute the update itself. Once the key is received by the machines on which the update is to be installed, the update, which has already been delivered in encrypted form, can be decrypted and installed. Since the update is encrypted, the update can be presumed not to be known to the world until the key is delivered. And, since the widespread distribution of the key takes a relatively short amount of time, the amount of time between when the update is first known, and the time at which a large number of machines are protected, is reduced, as compared with the time period that would exist if updates were distributed in unencrypted form ..."

While perfectly functional and useful, Thomlinson-Walker's original proposal suffers from two shortcomings:

Single Editor Support: Each software editor must manage his own keys. *i.e.* two editors cannot share keys without compromising the confidentiality of their respective patches.

Memory Increase: The list of published keys grows linearly with the number of updates. This is not a real-life problem because the number of software updates is usually small. However, it would be nice to come up with a system requiring only $O(1)$ or $O(\log^c n)$ memory for managing n updates [24].

The following sections will show how to improve Thomlinson-Walker's original proposal using standard cryptographic building-blocks such as one-way trapdoor functions, identity based encryption and tree-based hashing. The contribution of this invited talk is therefore the illustration of known techniques (*e.g.* [PQ10; RW96]) using a new problem rather than the design of new protocols. Throughout this section τ denotes the moment at which the key is disclosed.

24. Note that throughout this section complexities are expressed as a function of the number of updates and not as a function of the system's security parameter as is customary in cryptography. This is why, for instance, in Section 6.4.5, an IBE decryption operation is considered to require constant-time.

on day	reveal keys
1	r_{000}
2	r_{000}, r_{001}
3	r_{00}
4	r_{00}, r_{010}
5	r_{00}, r_{010}, r_{011}
6	r_{00}, r_{01}
7	r_0
8	r_0, r_{100}
9	r_0, r_{100}, r_{101}
10	r_0, r_{10}
11	r_0, r_{10}, r_{110}
12	$r_0, r_{10}, r_{110}, r_{111}$
13	r_0, r_{10}, r_{11}
14	r_0, r_1
15	r

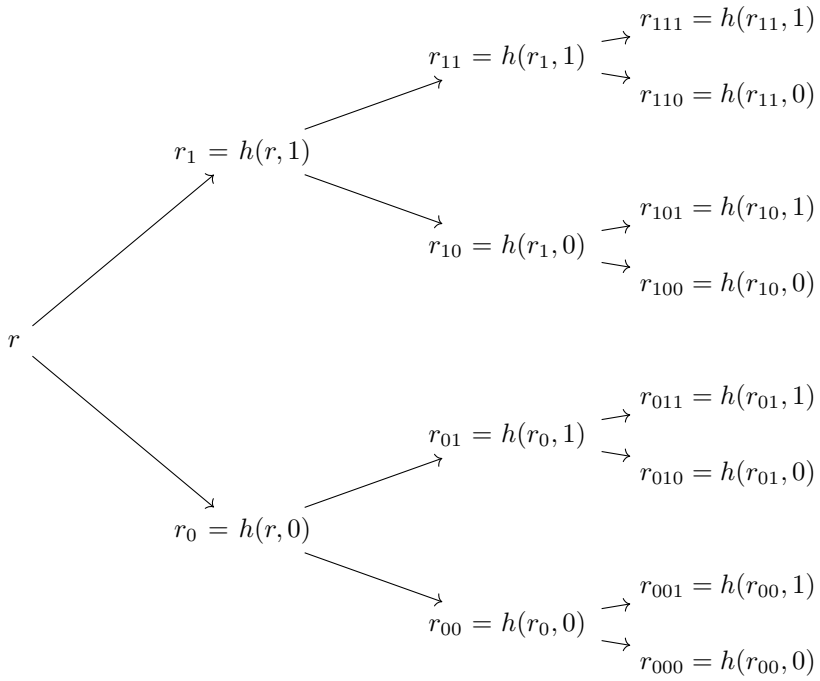


Figure 6.20 – Key tree example for 15 days.

6.4.1.1 Related Work:

The timed release of information is a widely researched area with an abundant bibliography. We do not overview these reference here but refer the reader to the excellent introduction found in [PQ10].

6.4.2 Single Editor, Constant Memory, Linear Time

We first present a single-editor patch management method that requires constant storage from the editor but claims from the client $O(n)$ time.

Let N be an RSA modulus [RSA78] generated by the editor. Let $3d = 1 \bmod \phi(N)$. The editor picks a random integer $r_0 \in \mathbb{Z}_N^*$. Everyday the editor computes $r_i = r_{i-1}^d \bmod N$ and updates the information on his website to (only) $\{N, r_i\}$. To retrieve the key of day $t < i$ the client (who knows the current date i) simply cubes r_i modulo N $i - t$ times to reach r_t. Note that exactly for the reasons described in [RW96], the client cannot speed-up computations and must spend $O(i - t)$ time to compute r_t from r_i.

This idea is also similar in concept to the reverse Canetti-Halevi-Katz [CHK03] scheme suggested in section 5.4 of [BBG05].

6.4.3 Single Editor, Polylogarithmic Memory, Polylogarithmic Time

We now use a hashing tree to achieve $O(\log^c n)$ time and storage. Instead of formally describing the algorithm, we illustrate the scheme's operation during 15 days. Pick a random r and derive a key tree by successive hashes as shown in Figure 6.20. The algorithm governing the management of this tree is straightforward.

6.4.4 Multiple Editors, Linear Memory, Constant Time

We will now extend Thomlinson-Walker's concept to multiple editors. As a typical example Microsoft, Google and Apple may want to use the same key distribution server for deploying patches for Windows, Chrome and iTunes without sharing any secret material. A technique for doing so was published by Mont *et alii* in [MHS03]. [MHS03] uses Identity Based Encryption (IBE). The concept of IBE was invented by Shamir in 1984 [Sha84]. It allows a party to encrypt a message using the recipient's identity as a public key. The corresponding private-key is provided by a central authority. The advantage of IBE over conventional public-key encryption is that it avoids certificate management, which greatly simplifies the implementation of secure communications between users. With an IBE scheme, users can simply use their email addresses as their identities. Moreover, the recipient does not need to be online to present a public-key certificate before the sender encrypts a message, and the sender does not have to be online to check the validity of the certificate.

More formally, an IBE scheme consists of four algorithms :

Setup generates the system's public parameters π and a private master key μ.

KeyGeneration takes as input an identity v and computes v's private key d_v using μ.

Encrypt encrypts messages for an identity v using π.

Decrypt decrypts ciphertexts for identity v using π and the private-key d_v.

[MHS03] considers time information as strings (*e.g.* [860]) and treats them as identities. A Trusted Third Party (TTP) generates π and maintains public list to which a new d_i is added every day. In other words, on day i the TTP reveals the keys $d_1, \ldots, d_i$. This allows different patch editors to encrypt patches into the future. The TTP also allows to preserve the editor's anonymity until τ. Indeed, an editor can post a patch on the TTP's website without indicating to which specific software the patch will be applied. This method forces all users to consult the list at date τ but increases operational security because it prevents the opponent from knowing in which software he has to look for flaws.

6.4.5 Multiple Editors, Polylogarithmic Memory, Polylogarithmic Time

Finally, it would be nice to combine all the previous desirable system features and provide memory-efficient and time-efficient multi-editor support. This can be achieved using Hierarchical IBE (HIBE) [HL02; GS02; BBG05]. HIBE generalizes IBE and allows to structure entities in a hierarchy. A level-i entity can distribute keys to its descendants but is unable to decrypt messages intended to ancestors and collaterals.

Just as an IBE, a HIBE comprises the algorithms Setup, KeyGeneration, Encrypt and Decrypt. However, while in IBE identities are binary strings, in a HIBE identities are ordered lists. As in IBE, Setup outputs $\{\pi, \mu\}$.

KeyGeneration takes as input an identity $(I_1, ..., I_k)$ and the private key $d[(I_1, ..., I_{k-1})]$ of the parent identity $(I_1, ..., I_{k-1})$ and outputs the private key $d[(I_1, ..., I_k)]$ for identity $(I_1, ..., I_k)$.

Encrypt encrypts messages for an identity $(I_1, ..., I_k)$ using π and Decrypt decrypts ciphertexts using the corresponding private key $d[(I_1, ..., I_k)]$.

We can hence adapt the tree construction of Section 6.4.3 as shown in Figure 6.21. We conveniently illustrate this idea for a week starting on Sunday and ending on Saturday.

on day	reveal keys
Sunday	d_{00}
Monday	d_{00}, d_{01}
Tuesday	d_0
Wednesday	d_0, d_{10}
Thursday	d_0, d_{10}, d_{11}
Friday	d_0, d_1
Saturday	μ

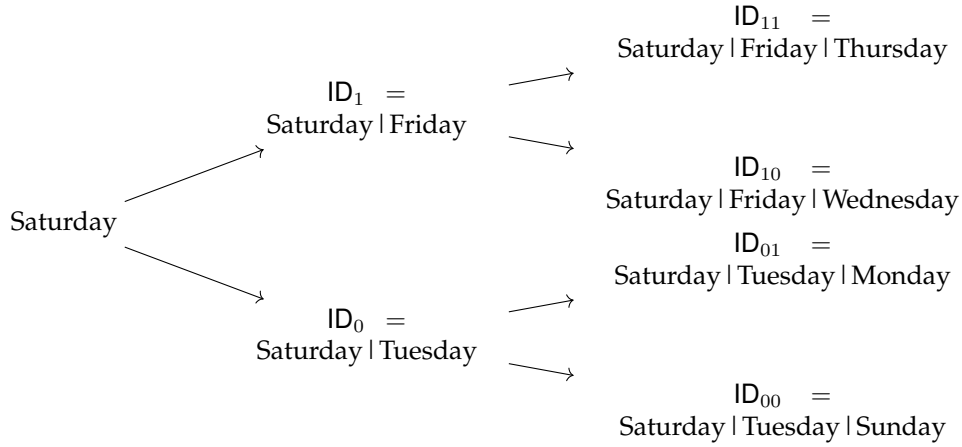


Figure 6.21 – HIBE tree example for 7 days. d_X denotes the secret key of identity ID_X.

Device connectivity		low exploit complexity $q = 0.1$	average exploit complexity $q = 0.01$	high exploit complexity $q = 0.001$
permanent	$p = 0.8$	5	14	24
usual	$p = 0.6$	3	8	12
intermittent	$p = 0.2$	2	3	5

Table 6.4 – Optimal τ values solved for various p, q probabilities.

6.4.6 How Long Should We Wait?

A last interesting question, totally unrelated to the above cryptographic discussion, is the determination of the optimal key release date τ. A plausible model can be the following: As an encrypted patch is announced, the opponent starts looking for the flaw. Let $\rho(t)$ be the probability that the vulnerability will be discovered by the opponent before t. Let $v(t)$ denote the proportion of users who downloaded the patch at time t. Here $\rho(0) = v(t) = 0$ and $\rho(\infty) = v(\infty) = 1$. It is easy to see that the optimal τ is the value that maximizes $(1 - \rho(t))v(t)$. It may be reasonable to assume that $v(t) \simeq 1 - p^t$ where p is the probability that a computer is not turned on by its owner during a day and $\rho(t) \simeq 1 - (1 - q)^t$ where q is the probability to independently discover the flaw after a one day's work. Resolution for this simplified model reveals that for most "reasonable" values (*e.g.* $1/6 \leq p \leq 2/3$ and $10^{-4} \leq q \leq 0.1$) τ would typically range somewhere between 1 and 20 days (Figure 6.22).

To see what this model imply in practice, we consider three typical device categories: permanently connected devices (*e.g.* mobile telephones), usually connected devices (*e.g.* PCs, tablets) and intermittently connected devices (*e.g.* smart-cards). Exploits of different technical difficulties were assigned the q values given in Table 6.4.

Table 6.4 confirms the intuition that (for a fixed p) τ increases with the exploit's complexity, *i.e.* the model takes advantage of the exploit's non-obviousness to spread the patch to more devices. In addition, (for a fixed q) τ increases with the device's connectivity as it appears better to patch only some devices rather

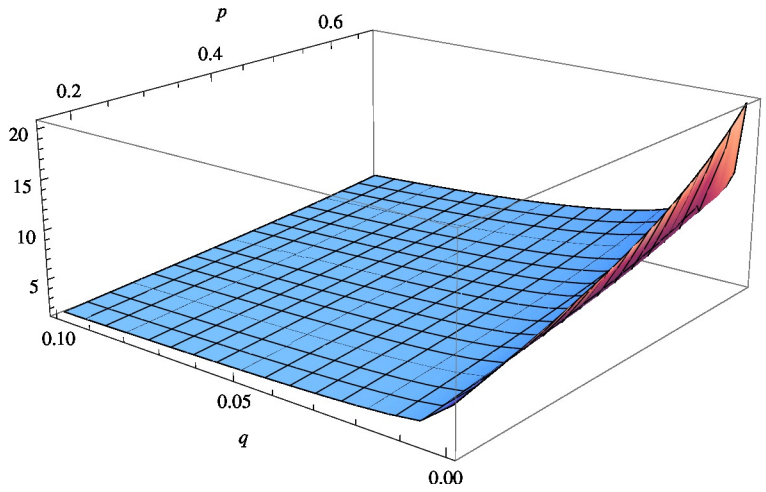


Figure 6.22 – Optimal τ for $\rho(t) = 1 - (1-q)^t$ and $v(t) \simeq 1 - p^t$. Solved for $1/6 \leq p \leq 2/3$ and $10^{-4} \leq q \leq 0.1$.

than let $v(t)$ slowly grow and maintain the entire device community at risk. We do not claim that this very simplified model accurately reflects reality.

CHAPTER 7

CONCLUSION AND FURTHER DEVELOPMENTS

7.1 Thesis Results & Contributions

This thesis addressed the three cornerstones of public-key cryptology: integrity, authentication and confidentiality.

The manuscript starts by an overview of the history of cryptography and by a summary of the mathematical preliminaries necessary for the presentation of our results.

We then present our results in three chapters, dealing respectively with integrity, authentication and confidentiality:

The chapter 4 presents our research results in the area of *integrity*.

The core result of the first chapter, detailed in Section 4.1, is a new attestation primitive allowing to prove the proper generation of RSA public keys. RSA public keys are central to many cryptographic applications; hence their validity is of primary concern to the scrupulous cryptographer. The most relevant properties of an RSA public key (n, e) depend on the *factors* of n: are they properly generated primes? are they large enough? is e co-prime with $\phi(n)$? etc. And of course, it is out of question to reveal n's factors.

Generic non-interactive zero-knowledge (NIZK) proofs can be used to prove such properties. However, NIZK proofs are not practical at all. Typically, such protocols turn out to be very specialized, and may not always be applicable (e.g., for some small values of e). For some very specific properties, specialized proofs exist but such *ad hoc* proofs are naturally hard to generalize.

Section 4.1 proposes a new type of *general-purpose* compact non-interactive proofs, called *attestations*, allowing the key generator to convince any third party that n was properly generated. The proposed construction applies to *any* prime generation algorithm, and is provably secure in the Random Oracle Model.

As a typical implementation instance, for a 138-bit security, verifying or generating an attestation requires $k = 1024$ prime generations. For this instance, each processed message will later need to be signed or encrypted 14 times by the final users of the attested moduli.

The second result in the first chapter, detailed in Section 4.2, is a new form of contract co-signature, called *legal fairness*, that does not rely on third parties or arbitrators. The proposed protocol is efficient, compact, fully distributed, fully dynamic, and provably secure in the Random Oracle Model. The protocol is illustrated for two parties using Schnorr's signature scheme.

In two-party computation, achieving both fairness and guaranteed output delivery is well known to be impossible. Despite this limitation, many approaches provide solutions of practical interest by weakening somewhat the fairness requirement. Such approaches fall roughly in three categories: "gradual release" schemes assume that the aggrieved party can eventually reconstruct the missing information; "optimistic schemes" assume a trusted third party arbitrator that can restore fairness in case of litigation; and "concurrent" or "legally fair" schemes in which a breach of fairness is compensated by the aggrieved party having a digitally signed cheque from the other party (called the keystone). Section 4.2 describes and analyses a new contract signing paradigm that doesn't require keystones to achieve legal fairness, and give a concrete construction based on Schnorr signatures which is compatible with standard Schnorr signatures and provably secure.

In a way these two results complement each other: attestation certifies the integrity of *computation* whereas legal fairness certifies the integrity of *interaction*.

Chapter 5 presents our research results in the area of *authentication*.

Discrete-logarithm authentication protocols are known to present two interesting features: The first is that the prover's commitment, $x = g^r$, claims most of the prover's computational effort. The second is that x does not depend on the challenge and can hence be computed in advance. Provers exploit this feature by pre-loading (or pre-computing) ready to use commitment pairs r_i, x_i. The r_i can be derived from a common seed but storing each x_i still requires 160 to 256 bits when implementing DSA or Schnorr.

Section 5.1 proposes a new concept called *slow motion zero-knowledge* (SM-ZK). SM-ZK allows the prover to slash commitment size (by a factor of 4 to 6) by combining classical zero-knowledge and a timing channel. We pay the conceptual price of requiring the ability to measure time but, in exchange, obtain communication-efficient protocols.

Section 5.2 introduces "thrifty" zero-knowledge protocols, or TZK. These protocols are constructed by introducing a bias in the challenge send by the prover. This bias is chosen so as to maximize the security versus effort trade-off. We illustrate the benefits of this approach on several well-known zero-knowledge protocols.

Section 5.3 presents a lightweight algorithm allowing a verifier to collectively identify a community of provers. This protocol is more efficient than one-to-one node authentication, resulting in less communication, less computation, and hence a smaller overall energy consumption. The protocol is provably secure, and achieves zero-knowledge authentication of a time linear in the degree of the spanning tree.

The proposed authentication protocol may be adapted to better fit constraints: in the context of Internet of Things (IoT), communication is a very costly operation. We describe versions that reduce the amount of data sent by individual nodes, while maintaining security.

Section 5.4 describes the forensic analysis of what the authors believe to be the most sophisticated smart card fraud encountered to date. In a way, this section illustrates what can happen when authentication protocols are wrongly designed. In 2010, Murdoch *et al.* [MDAB10] described a man-in-the-middle attack against EMV cards. [MDAB10] demonstrated the attack using a general purpose FPGA board, noting that "*miniaturization is mostly a mechanical challenge, and well within the expertise of criminal gangs*". This indeed happened in 2011, when about 40 sophisticated card forgeries surfaced in the field. These forgeries are remarkable in that they embed two chips wired top-to-tail. The first chip is clipped from a genuine stolen card. The second chip plays the role of the man-in-the-middle and communicates directly with the point of sale (PoS) terminal. The entire assembly is embedded in the plastic body of yet another stolen card. The forensic analysis relied on X-ray chip imaging, side-channel analysis, protocol analysis, and microscopic optical inspections.

Chapter 6 presents our research results in the area of *confidentiality*.

The research work presented in Section 6.1 proposes a public-key cryptosystem and a short password encryption mode, where traditional hardness assumptions are replaced by specific refinements of the CAPTCHA concept called Decisional and Existential CAPTCHAs.

The public-key encryption method, achieving 128-bit security, typically requires from the sender to solve one CAPTCHA. The receiver does not need to resort to any human aid.

A second symmetric encryption method allows to encrypt messages using very short passwords shared between the sender and the receiver. Here, a simple 5-character alphanumeric password provides sufficient security for all practical purposes.

We conjecture that the automatic construction of Decisional and Existential CAPTCHAs is possible and provide candidate ideas for their implementation.

Honey Encryption (HE), introduced by Juels and Ristenpart (Eurocrypt 2014, [JR14]), is an encryption paradigm designed to produce ciphertexts yielding plausible-looking but bogus plaintexts upon decryption with wrong keys. Thus brute-force attackers need to use additional information to determine whether they indeed found the correct key.

At the end of their paper, Juels and Ristenpart leave as an open question the adaptation of honey encryption to natural language messages. A recent paper by Chatterjee *et al.* [CBJR15] takes a mild attempt at the challenge and constructs a natural language honey encryption scheme relying on simple models for passwords.

Section 6.2 explains why this approach cannot be extended to reasonable-size human-written documents *e.g.* e-mails. We propose an alternative solution and evaluate its security.

Section 6.3 generalizes the concept of Hierarchical Identity-Based Encryption (HIBE) by proposing a new primitive called Hierarchical Identity-Based Broadcast Encryption (HIBBE). Similar to HIBE, HIBBE organizes users in a tree-like structure and users can delegate their decryption capability to their subordinates, which mirrors real-world hierarchical social organizations. Unlike HIBE merely allowing a single decryption path, HIBBE enables encryption to any subset of the users and only the intended users (and their supervisors) can decrypt.

We define Ciphertext Indistinguishability against Adaptively Chosen-Identity-Vector-Set and Chosen-Ciphertext Attack (IND-CIVS-CCA2) which capture the most powerful attacks on HIBBE in the real world. We achieve this goal in the standard model in two steps. We first construct an efficient HIBBE Scheme (HIBBES) against Adaptively Chosen-Identity-Vector-Set and Chosen-Plaintext Attack (IND-CIVS-CPA) in which the attacker is not allowed to query the decryption oracle. Then we convert it into an IND-CIVS-CCA2 scheme at only a marginal cost, i.e., merely adding one on-the-fly dummy user at the first depth of hierarchy in the basic scheme without requiring any other cryptographic primitives. Furthermore, our CCA2-secure scheme natively allows public ciphertext validity test, which is a useful property when a CCA2-secure HIBBES is used to design advanced protocols.

The last research work of the third chapter illustrates how standard cryptographic techniques can be applied to real-life security products and services. Section 6.4 improves a little noticed yet ingenious Microsoft patent by Thomlinson and Walker. The Thomlinson-Walker system distributes encrypted patches to avoid reverse engineering by opponents (who would then be able to launch attacks on unpatched users). When the proportion of users who downloaded the encrypted patch becomes big enough, the decryption key is disclosed and all users install the patch.

7.2 Personal Perspectives

Our results highlight the richness, the complexity but also the difficulty of modern cryptography. Cryptography is a hybrid discipline, blending mathematics, theoretical computer science, statistics, hardware and software. These ingredients are interwoven to construct increasingly complex primitives, protocols and real-life products. This complexity forces the practitioner to precisely formulate assumptions and to carefully implement building-blocks.

Our thesis illustrates the pace at which the field is rushing ahead: despite the fact that integrity, authentication and confidentiality are very basic functions researched since decades, there is still a lot of room for innovation and invention in cryptology.

Several eras can be distinguished in the evolution of modern cryptography:

The first two decades (1970s-1980s) were the *exploratory era* during which new primitives were discovered or designed at a fast pace. While many fundamental results were invented during this fertile exploratory era, the proposed advances were usually "big ideas" that, while frequently secure, lacked theoretical models.

The 1990s were the *modeling era* of modern cryptography. During the 1990s fundamental models and definitions were rigorously formalized and researchers got to agree about the meaning of common concepts such as security, adversaries or proofs.

The 2000s can be regarded as the *technical and industrial adoption era* where the formalization effort continued along with the sophistication of new groundbreaking concepts. In the 2000s the number of cryptographic commercial products and standards reached unprecedented records. It is easy to see that from the 2000s and on nearly no "low hanging fruit" results were discovered and most new advances required substantial mathematical theoretical imports (e.g. lattice theory).

The 2010s seem to be *boundary exploration era*, a decade during which the research community endeavors to push the boundaries of cryptographic functionalities, attempting to build features such as fully homomorphic encryption, multi-linear maps, indistinguishability obfuscation or functional encryption.

While it is difficult to glimpse into the future to predict where the field is heading and what a typical Eurocrypt 2076 program will look, the author would still like to formulate a number of hypotheses:

1. **An era of cryptographic thinking machines?** An important part of the cryptographer's future work will be to design and train machines that will design, prove and break new cryptographic protocols. Such machines will eventually be able to translate natural language security requirements into cryptographic protocols. Automated proofs and automated theorem proving will become a standard tool used by cryptographers.
2. **Beyond Turing machines?** Alternative computation paradigms (e.g. DNA, quantum, wetware, molecular scale electronics) will gain importance and, by ricochet, encourage cryptographic creativity.
3. **Integrating cryptography and biology?** Human DNA will replace passwords and noninvasive DNA readers will become pervasive. DNA will become an inexhaustible source of passwords during a human's lifetime. Organisms engineered to recognize human DNA and compute with it will allow access to information. In-brain artificial decryption organs [1] will become the ultimate end-to-end encryption means.
4. **An era of abstraction?** Some sub-areas of cryptography will evolve into purely theoretical proofs of existence disconnected from any practical implementability. The divide between practitioners and theorists will increase. It is likely that at some point cryptographers will undertake a systematic exploration of humanity's mathematical knowledge and evaluate, in each mathematical sub-area, what its cryptographic applications might be.

1. e.g. capable of sensing external physical stimuli carrying information, inserting the information directly into the human cranial nerves and vestibulocochlear nerve or even directly committing the decrypted information into the brain's memory.

5. **A post modular-multiplication era?** The advent of major advances in DLP and factorization will weaken the traditional links between number-theory and cryptology and favor the progressive reliance on other hard problems. It is also probable that radical advances will make all FACT-based and DLP-based cryptography disappear at once.
6. **Redefining the adversary?** As cryptographic protocols will get increasingly complex, so will the cheating scenarios and adversarial goals. Consequently, the importance of game theory in cryptography will increase. As we write these lines, proof methodologies define what is harmful and demonstrate its successful avoidance. In the future machines will infer alone what harmful is before harm is actually done.
7. **An era of cryptographic sufficiency?** A century of cryptographic research will mature and stabilize the field to a point where advances and discoveries will go past actual human needs. By 2040, standard cryptographic tools will suffice for satisfying 95% of human commercial (real-life) needs. Consequently the *practical* battle between the cryptographer and the cryptanalyst will be won by the cryptographer.
8. **An era of cryptographic bottlenecks?** The increasing pace at which information will be transmitted will become incompatible with digital-only cryptography. Hence physical channel security and analog cryptographic primitives (neuromorphic cryptoprocessors) are likely to gain in importance and emerge. Because these are based on physics, algorithms will be designed to take advantage of existing physical phenomena (rather than forcing physics to mimic mathematical behavior as is the case today).
9. **A shift in the demographics of cryptographic research?** The scientific impact of emerging countries in the community will certainly increase, while the relative percentage of major results discovered in Europe and north America will probably tend to decrease. More female researchers will enter the field and author more major results.
10. **An era of fast-following agencies?** The academic community's knowledge will surpass that of state agencies (such as the NSA) if that is not already the case. Intelligence agencies will devote a major part of their resources to software attacks, exploits and key theft methods rather than cryptography.

APPENDIX A

COMPUTING THRIFTY PARAMETERS

The following implementation uses Python 2.7 and the CVXOPT library[1] to solve the constrained optimization problem of Equation (5.1). Here the γ_j are computed for Fiat-Shamir, but could easily be adapted to other settings.

```
from cvxopt import matrix, solvers
from fractions import Fraction
import math

mul = lambda x,y: x*y

# Binomial coefficient \binom{n}{k}
def binom(n,k):
  return int(reduce(mul,(Fraction(n-i,i+1) for i in range(k)),1))

# Populations \gamma_k (for Fiat-Shamir)
def get_coeffsp(n):
  return [binom(n,k+1) for k in range(n)]

# Work coefficients k * \gamma_k (for Fiat-Shamir)
def get_coeffsw(n):
  r = get_coeffsp(n)
  return [(i+1)*c for i,c in enumerate(r)]

# Solve optimization problem for given n and epsilon
def solve_lp(epsilon, n):

  coeffsp = map(float, get_coeffsp(n))
  coeffsw = map(float, get_coeffsw(n))

  # Put the problem in canonical form, i.e.
  # construct matrix A and vectors b, c
  # such that the problem is in the form Ax + b <= c
  A = []
  for i in range(n):
    A += [[0.]*i + [1.] + [0.]*(n-i-1)]
  A += [map(lambda y:-y, coeffsp)]
  for i in range(n):
    A += [[0.]*i + [-1.] + [0.]*(n-i-1)]
  A += [coeffsp]
  A = matrix(A).trans()
  b = matrix([epsilon] * n + [epsilon-1.] + [0.] * n + [1.])
  c = matrix(coeffsw)

  # Solve the linear programming problem
  sol = solvers.lp(c, A, b)

  # Extract solution and append p0
  p0 = 1 - sum(i*w for i, w in zip(sol['x'], coeffsp))
  pi = [p0] + [i for i in sol['x']]

  # Compute total work (for Fiat-Shamir)
  w = sum(i * w for i, w in zip(sol['x'], coeffsw))

  # Compute total security
  sec = -math.log(epsilon, 2)
```

1. http://cvxopt.org/

```
  # Return security, work, efficiency, and optimal probabilities
  return (sec, w, sec/w, xi)

# Challenge bits
n = 16

# Number of sampling points
N = 500

# Smallest possible value of epsilon
mineps = 2**(-n)

# Save data to a file by uniformly sampling values of epsilon
f = open('output%s.txt'%n, 'w')
plabel = '\t'.join(['p%s'%(i) for i in range(n+1)])
f.write('i\teps\ts\tw\tse\t%s\n'%plabel)

for i in range(N):
   s = float(i)/N * n
   e = 2**(-s)
   s, w, se, xi = solve_lp(e, n)
   xi = '\t'.join(map(str, xi))
   f.write('%s\t%s\t%s\t%s\t%s\t%s\n'%(i,e,s,w,se,xi))

f.close()
```

Bibliography

[860] ISO 8601:2004. *Data elements and interchange formats Information interchange Representation of dates and times*. Tech. rep. URL: https://www.iso.org/obp/ui/#iso:std:iso:8601:ed-3:v1:en (cit. on p. 159).

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. « From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security ». In: *Advances in Cryptology – EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Apr. 2002, pp. 418–433 (cit. on p. 56).

[AAF+15] Ehsan Aerabi, A. Elhadi Amirouche, Houda Ferradi, Rémi Géraud, David Naccache, and Jean Vuillemin. *The Conjoined Microprocessor*. Cryptology ePrint Archive, Report 2015/974. http://eprint.iacr.org/2015/974. 2015 (cit. on pp. 33, 35).

[AAF+16] Ehsan Aerabi, A. Elhadi Amirouche, Houda Ferradi, Rémi Géraud, David Naccache, and Jean Vuillemin. « The Conjoined Microprocessor ». In: *2016 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016, McLean, VA, USA, May 3-5, 2016*. Ed. by Ryan A. Peter Y., Naccache David, and Quisquater Jean-Jacques. IEEE, 2016, pp. 67–70. ISBN: 978-3-662-49301-4. DOI: 10.1109/HST.2016.7495558. URL: http://dx.doi.org/10.1109/HST.2016.7495558 (cit. on pp. 33, 35).

[Abd11] Michel Abdalla. « Reducing The Need For Trusted Parties In Cryptography ». PhD thesis. École normale supérieure, 2011, pp. 7–11 (cit. on p. 17).

[ABHL03] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. « CAPTCHA: Using Hard AI Problems for Security ». In: *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*. Ed. by Eli Biham. Vol. 2656. Lecture Notes in Computer Science. Springer, 2003, pp. 294–311. ISBN: 3-540-14039-5. DOI: 10.1007/3-540-39200-9_18 (cit. on p. 117).

[ABMW05] Martín Abadi, Michael Burrows, Mark S. Manasse, and Ted Wobber. « Moderately hard, memory-bound functions ». In: *ACM Trans. Internet Techn.* 5.2 (2005), pp. 299–327. DOI: 10.1145/1064340.1064341 (cit. on p. 72).

[And93] RJ Anderson. « Practical RSA trapdoor ». In: *Electronics Letters* 29.11 (1993), pp. 995–995 (cit. on p. 43).

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. « 1-out-of-n Signatures from a Variety of Keys ». In: *Advances in Cryptology – ASIACRYPT 2002*. Ed. by Yuliang Zheng. Vol. 2501. Lecture Notes in Computer Science. Queenstown, New Zealand: Springer, Heidelberg, Germany, Dec. 2002, pp. 415–432 (cit. on pp. 55, 58).

[AR05] D. Anshul and S. Roy. « A ZKP-Based Identification Scheme for Base Nodes in Wireless Sensor Networks ». In: *Proceedings of the 20th ACM Symposium on Applied Computing - SAC'05*. ACM, 2005, pp. 319–323 (cit. on p. 89).

[ASW02] N. Asokan, V. Shoup, and M. Waidner. « Optimistic Fair Exchange of Digital Signatures ». In: *Advances in Cryptology - EUROCRYPT'98*. Vol. 1403. Lecture Notes in Computer Science. Springer, 2002, pp. 591–606 (cit. on p. 9).

[ASW97] N. Asokan, Matthias Schunter, and Michael Waidner. « Optimistic Protocols for Fair Exchange ». In: *ACM CCS 97: 4th Conference on Computer and Communications Security*. Zurich, Switzerland: ACM Press, Apr. 1997, pp. 7–17 (cit. on pp. 9, 55).

[Bab86] László Babai. « On Lovász' lattice reduction and the nearest lattice point problem ». In: *Combinatorica* 6.1 (1986), pp. 1–13 (cit. on p. 27).

[BB04] Dan Boneh and Xavier Boyen. « Efficient selective-ID secure identity-based encryption without random oracles ». In: *EUROCRYPT '04*. Vol. 3027. LNCS. Springer Berlin Heidelberg, 2004, pp. 223–238 (cit. on pp. 139, 143, 150).

[BBBB10] Hristo Bojinov, Elie Bursztein, Xavier Boyen, and Dan Boneh. « Kamouflage: Loss-resistant password management ». In: *Computer Security–ESORICS 2010*. Springer, 2010, pp. 286–302 (cit. on p. 130).

[BBC+14a] Thomas Bourgeat, Julien Bringer, Hervé Chabanne, Robin Champenois, Jérémie Clément, Houda Ferradi, Marc Heinrich, Paul Melotti, David Naccache, and Antoine Voizard. « New Algorithmic Approaches to Point Constellation Recognition ». In: *ICT Systems Security and Privacy Protection: 29th IFIP TC 11 International Conference, SEC 2014, Marrakech, Morocco, June 2-4, 2014. Proceedings*. Ed. by Nora Cuppens-Boulahia, Frédéric Cuppens, Sushil Jajodia, Anas Abou El Kalam, and Thierry Sans. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 80–90. ISBN: 978-3-642-55415-5. DOI: 10.1007/978-3-642-55415-5_7. URL: http://dx.doi.org/10.1007/978-3-642-55415-5_7 (cit. on pp. 33, 36).

[BBC+14b] Thomas Bourgeat, Julien Bringer, Hervé Chabanne, Robin Champenois, Jérémie Clément, Houda Ferradi, Marc Heinrich, Paul Melotti, David Naccache, and Antoine Voizard. *New Algorithmic Approaches to Point Constellation Recognition*. CoRR, abs/1405.1402. http://arxiv.org/abs/1405.1402. 2014 (cit. on pp. 33, 36).

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. « Hierarchical identity based encryption with constant size ciphertext ». In: *EUROCRYPT '05*. Vol. 3494. LNCS. Springer Berlin Heidelberg, 2005, pp. 440–456 (cit. on pp. 139, 143, 144, 158, 159).

[BCC88] G. Brassard, D. Chaum, and C. Crépeau. « Minimum disclosure proofs of knowledge ». In: *Journal of Computer and System Sciences* 37.2 (1988), pp. 156–189 (cit. on pp. 43, 82).

[BDH99] D. Boneh, G. Durfee, and N. Howgrave-Graham. « Factoring $N = p^r q$ for large r ». In: *Advances in Cryptology – CRYPTO'99*. Springer Berlin Heidelberg, 1999, pp. 326–337 (cit. on p. 14).

[BDJR97] M. Bellare, A. Desai, E. Jokipii, and Ph. Rogaway. « A Concrete Security Treatment of Symmetric Encryption ». In: *Proceedings of the 38th International IEEE Symposium on Foundations of Computer Science - FOCS'97*. 1997, pp. 394–403 (cit. on p. 25).

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and Ph. Rogaway. « Relations Among Notions of Security for Public-Key Encryption Schemes ». In: *Advances in Cryptology - CRYPTO'98*. Vol. 1462. Lecture Notes in Computer Science. Springer-Verlag, 1998, pp. 26–45 (cit. on pp. 25, 26).

[Ber86] Robert L. Bernstein. « Multiplication by Integer Constants ». In: *Softw., Pract. Exper.* 16.7 (1986), pp. 641–652. DOI: 10.1002/spe.4380160704 (cit. on p. 76).

[BF01] Dan Boneh and Matt Franklin. « Identity-based encryption from the Weil pairing ». In: *CRYPTO '01*. Vol. 2139. LNCS. Springer Berlin Heidelberg, 2001, pp. 213–229 (cit. on p. 139).

[BF03] Dan Boneh and Matthew Franklin. « Identity-based encryption from the Weil pairing ». In: *SIAM Journal on Computing* 32.3 (2003), pp. 586–615 (cit. on p. 139).

[BF97] Dan Boneh and M. Franklin. « Efficient generation of shared RSA keys ». In: *Advances in Cryptology – CRYPTO'97*. Springer Verlag, 1997, pp. 425–439 (cit. on p. 44).

[BFG+14] Hadrien Barral, Houda Ferradi, Rémi Géraud, Georges-Axel Jaloyan, and David Naccache. *ARMv8 Shellcodes from 'A' to 'Z'*. CoRR, abs/1608.03415. http://arxiv.org/abs/1608.03415. 2014 (cit. on pp. 33, 34).

[BFG+16] Hadrien Barral, Houda Ferradi, Rémi Géraud, Georges-Axel Jaloyan, and David Naccache. « ARMv8 Shellcodes from 'A' to 'Z' ». In: *The 12th International Conference on Information Security Practice and Experience (ISPEC 2016) Zhangjiajie, China, November 16-18, 2016. Proceedings*. Ed. by Chen Liqun and H. Robert Deng. Cham: Springer International Publishing, 2016 (cit. on pp. 33, 34).

[BFL90] J. Boyar, K. Friedl, and C. Lund. « Practical zero-knowledge proofs: Giving hints and using deficiencies ». In: *Advances in Cryptology – EUROCRYPT'89*. Springer Berlin Heidelberg, 1990, pp. 155–172 (cit. on p. 43).

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. « Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract) ». In: *20th Annual ACM Symposium on Theory of Computing*. Chicago, Illinois, USA: ACM Press, May 1988, pp. 103–112 (cit. on p. 28).

[BGG+90] M. Ben-Or, O. Goldreich, Sh. Goldwasser, J. Håstad, J. Kilian, S. Micali, and Ph. Rogaway. « Everything Provable is Provable in Zero-Knowledge ». In: *Advances in Cryptology – CRYPTO'88*. Springer New York, 1990, pp. 37–56 (cit. on p. 43).

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. « Aggregate and Verifiably Encrypted Signatures from Bilinear Maps ». In: *Advances in Cryptology — EUROCRYPT'03*. Vol. 2656. Lecture Notes in Computer Science. Springer, 2003, pp. 416–432 (cit. on p. 89).

[BGMR90] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. « A Fair Protocol for Signing Contracts ». In: *IEEE Transactions on Information Theory* 36.1 (1990), pp. 40–46 (cit. on p. 9).

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. « Evaluating 2-DNF formulas on ciphertexts ». In: *TCC '05*. Vol. 3378. LNCS. Springer Berlin Heidelberg, 2005, pp. 325–341 (cit. on p. 140).

[BGR98] M. Bellare, J. A. Garay, and T. Rabin. « Fast Batch Verification for Modular Exponentiation and Digital Signatures ». In: *Advances in Cryptology - EUROCRYPT'98*. Vol. 1403. Lecture Notes in Computer Science. Springer, 1998, pp. 236–250 (cit. on p. 95).

[BGS94] Jürgen Bierbrauer, K. Gopalakrishnan, and Douglas R. Stinson. « Bounds for Resilient Functions and Orthogonal Arrays ». In: *Advances in Cryptology – CRYPTO'94*. Ed. by Yvo Desmedt. Vol. 839. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1994, pp. 247–256 (cit. on p. 84).

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. « Collusion resistant broadcast encryption with short ciphertexts and private keys ». In: *CRYPTO '05*. Vol. 3621. LNCS. Springer Berlin Heidelberg, 2005, pp. 258–275 (cit. on p. 139).

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. « Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract) ». In: *20th Annual ACM Symposium on Theory of Computing*. Chicago, Illinois, USA: ACM Press, May 1988, pp. 1–10 (cit. on pp. 9, 55).

[BH08] Dan Boneh and Michael Hamburg. « Generalized identity based and broadcast encryption schemes ». In: *ASIACRYPT '08*. Vol. 5350. LNCS. Springer Berlin Heidelberg, 2008, pp. 455–470 (cit. on p. 140).

[BK05] Dan Boneh and Jonathan Katz. « Improved efficiency for CCA-secure cryptosystems built using identity-based encryption ». In: *CT-RSA '05*. Vol. 3376. LNCS. Springer Berlin Heidelberg, 2005, pp. 87–103 (cit. on pp. 139, 140).

[BL05] Henry S. Baird and Daniel P. Lopresti, eds. *Human Interactive Proofs, Second International Workshop, HIP 2005, Bethlehem, PA, USA, May 19-20, 2005, Proceedings*. Vol. 3517. Lecture Notes in Computer Science. Springer, 2005. ISBN: 3-540-26001-3 (cit. on p. 117).

[Ble96] D. Bleichenbacher. « Generating EIGamal Signatures Without Knowing the Secret Key ». In: *Advances in Cryptology — EUROCRYPT'96*. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 10–18 (cit. on p. 21).

[Blu83] Manuel Blum. « Coin flipping by telephone a protocol for solving impossible problems ». In: *ACM SIGACT News* 15.1 (1983), pp. 23–27 (cit. on p. 9).

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. « Direct chosen ciphertext security from identity-based techniques ». In: *CCS '05*. ACM, 2005, pp. 320–329 (cit. on pp. 139, 140, 150).

[BN00] Dan Boneh and Moni Naor. « Timed Commitments ». In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*. Ed. by Mihir Bellare. Vol. 1880. Lecture Notes in Computer Science. Springer, 2000, pp. 236–254. ISBN: 3-540-67907-3. DOI: 10.1007/3-540-44598-6_15 (cit. on p. 73).

[Bon12] Joseph Bonneau. « The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords ». In: *2012 IEEE Symposium on Security and Privacy*. San Francisco, California, USA: IEEE Computer Society Press, May 2012, pp. 538–552 (cit. on p. 125).

[Bor41] Jorge Luis Borges. *El Jardín de senderos que se bifurcan*. Editorial Sur, 1941 (cit. on p. 131).

[Bor44] Jorge Luis Borges. *Ficcione*. Editorial Sur, 1944 (cit. on p. 131).

[Bou00] F. Boudot. « Efficient proofs that a committed number lies in an interval ». In: *Advances in Cryptology – EUROCRYPT'00*. 2000, pp. 431–444 (cit. on p. 44).

[BR93] Mihir Bellare and Phillip Rogaway. « Random oracles are practical: A paradigm for designing efficient protocols ». In: *CCS '93*. ACM, 1993, pp. 62–73 (cit. on pp. 18, 19, 22).

[BR94] M. Bellare and Ph. Rogaway. « Entity Authentication and Key Distribution ». In: *Advances in Cryptology - CRYPTO'93*. Vol. 773. Lecture Notes in Computer Science. Springer-Verlag, 1994, pp. 232–249 (cit. on pp. 22, 76).

[BR95] Mihir Bellare and Phillip Rogaway. « Optimal Asymmetric Encryption ». In: *Advances in Cryptology – EUROCRYPT'94*. Ed. by Alfredo De Santis. Vol. 950. Lecture Notes in Computer Science. Perugia, Italy: Springer, Heidelberg, Germany, May 1995, pp. 92–111 (cit. on p. 19).

[BR96] Mihir Bellare and Phillip Rogaway. « The Exact Security of Digital Signatures: How to Sign with RSA and Rabin ». In: *Advances in Cryptology – EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Saragossa, Spain: Springer, Heidelberg, Germany, May 1996, pp. 399–416 (cit. on p. 19).

[BRW03] M. Bellare, Ph. Rogaway, and D. Wagner. « EAX: A Conventional Authenticated-Encryption Mode ». In: *IACR Cryptology ePrint Archive* 2003 (2003), p. 69 (cit. on p. 11).

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004 (cit. on p. 84).

[BW00] Birgit Baum-Waidner and Michael Waidner. « Round-Optimal and Abuse Free Optimistic Multi-party Contract Signing ». In: *Automata, Languages and Programming, 27th International Colloquium, ICALP 2000, Geneva, Switzerland, July 9-15, 2000, Proceedings*. Ed. by Ugo Montanari, José D. P. Rolim, and Emo Welzl. Vol. 1853. Lecture Notes in Computer Science. Springer, 2000, pp. 524–535. ISBN: 3-540-67715-1. DOI: 10.1007/3-540-45022-X_44 (cit. on pp. 56, 58).

[BY93] Mihir Bellare and Moti Yung. « Certifying cryptographic tools: The case of trapdoor permutations ». In: *Advances in Cryptology – CRYPTO'92*. Springer. 1993, pp. 442–460 (cit. on p. 43).

[BY96] Mihir Bellare and Moti Yung. « Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation ». In: *Journal of Cryptology* 9.3 (1996), pp. 149–166 (cit. on p. 43).

[Cas03] Michael Case. « A Beginner's Guide To The General Number ». In: *ECE 575 Data Security and Cryptography Project* (2003) (cit. on p. 14).

[CB03] Monica Chew and Henry S. Baird. « BaffleText: a human interactive proof ». In: *Document Recognition and Retrieval X, 22-23 January 2003, Santa Clara, California, USA, Proceedings*. Ed. by Tapas Kanungo, Elisa H. Barney Smith, Jianying Hu, and Paul B. Kantor. Vol. 5010. SPIE Proceedings. SPIE, 2003, pp. 305–316. ISBN: 0-8194-4810-9 (cit. on p. 117).

[CBJR15] Rahul Chatterjee, Joseph Bonneau, Ari Juels, and Thomas Ristenpart. « Cracking-Resistant Password Vaults Using Natural Language Encoders ». In: *2015 IEEE Symposium on Security and Privacy*. San Jose, California, USA: IEEE Computer Society Press, May 2015, pp. 481–498. DOI: 10.1109/SP.2015.36 (cit. on pp. 32, 115, 126, 128, 131, 165).

[CC00] Christian Cachin and Jan Camenisch. « Optimistic Fair Secure Computation ». In: *Advances in Cryptology – CRYPTO 2000*. Ed. by Mihir Bellare. Vol. 1880. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2000, pp. 93–111 (cit. on p. 55).

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. « Multiparty Unconditionally Secure Protocols (Extended Abstract) ». In: *20th Annual ACM Symposium on Theory of Computing*. Chicago, Illinois, USA: ACM Press, May 1988, pp. 11–19 (cit. on pp. 9, 55).

[CD98] R. Cramer and I. Damgård. « Zero-knowledge proof for finite field arithmetic, or: Can zero-knowledge be for free? » In: *Advances in Cryptology – CRYPTO'98*. Springer-Verlag, 1998, pp. 424–441 (cit. on p. 43).

[CFGN16a] Jean-Michel Cioranesco, Houda Ferradi, Rémi Géraud, and David Naccache. *Process Table Covert Channels: Exploitation and Countermeasures*. Cryptology ePrint Archive, Report 2016/227. http://eprint.iacr.org/2016/227. 2016 (cit. on pp. 33, 35).

[CFGN16b] Simon Cogliani, Houda Ferradi, Rémi Géraud, and David Naccache. « Thrifty Zero-Knowledge - When Linear Programming Meets Cryptography ». In: *The 12th International Conference on Information Security Practice and Experience (ISPEC 2016) Zhangjiajie, China, November 16-18, 2016. Proceedings*. Ed. by Chen Liqun and H. Robert Deng. Cham: Springer International Publishing, 2016 (cit. on p. 30).

[CFGN16c] Simon Cogliani, Houda Ferradi, Rémi Géraud, and David Naccache. *Thrifty Zero-Knowledge - When Linear Programming Meets Cryptography*. Cryptology ePrint Archive, Report 2016/443. http://eprint.iacr.org/2016/443. 2016 (cit. on p. 30).

[CFH+16] Simon Cogliani, Bao Feng, Ferradi Houda, Rémi Géraud, Diana Maimuţ, David Naccache, Rodrigo Portella do Canto, and Guilin Wang. *Public-Key Based Lightweight Swarm Authentication*. Cryptology ePrint Archive, Report 2016/750. http://eprint.iacr.org/2016/750. 2016 (cit. on p. 31).

[CFN13] Jean-Michel Cioranesco, Houda Ferradi, and David Naccache. « Communicating Covertly through CPU Monitoring ». In: *IEEE Security and Privacy* 11.6 (2013), pp. 71–73 (cit. on pp. 33, 34).

[CFT98] A. Chan, Y. Frankel, and Y. Tsiounis. « Easy come - easy go divisible cash ». In: *Advances in Cryptology – EUROCRYPT'98*. Springer-Verlag, 1998, pp. 561–575 (cit. on p. 44).

[CGJ+08] Richard Chow, Philippe Golle, Markus Jakobsson, Lusha Wang, and XiaoFeng Wang. « Making CAPTCHAs clickable ». In: *Proceedings of the 9th Workshop on Mobile Computing Systems and Applications, HotMobile 2008, Napa Valley, California, USA, February 25-26, 2008*. Ed. by Mirjana Spasojevic and Mark D. Corner. ACM, 2008, pp. 91–94. ISBN: 978-1-60558-118-7. DOI: 10.1145/1411759.1411783 (cit. on p. 117).

[CGMW97] Liqun Chen, Dieter Gollmann, Chris J. Mitchell, and Peter R. Wild. « Secret Sharing with Reusable Polynomials ». In: *ACISP 97: 2nd Australasian Conference on Information Security and Privacy*. Ed. by Vijay Varadharajan, Josef Pieprzyk, and Yi Mu. Vol. 1270. Lecture Notes in Computer Science. Sydney, NSW, Australia: Springer, Heidelberg, Germany, July 1997, pp. 183–193. DOI: 10.1007/BFb0027925 (cit. on p. 28).

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. « A forward-secure public-key encryption scheme ». In: *EUROCRYPT '03*. Vol. 2656. LNCS. Springer Berlin Heidelberg, 2003, pp. 255–271 (cit. on pp. 139, 143, 158).

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. « Chosen-ciphertext security from identity-based encryption ». In: *EUROCRYPT '04*. Vol. 3027. LNCS. Springer Berlin Heidelberg, 2004, pp. 207–222 (cit. on pp. 139, 140, 143).

[Cho02] Noam Chomsky. *Syntactic structures*. Walter de Gruyter, 2002 (cit. on p. 128).

[Cho56] Noam Chomsky. « Three models for the description of language ». In: *Information Theory, IRE Transactions on* 2.3 (1956), pp. 113–124 (cit. on pp. 128, 132).

[Cho59] Noam Chomsky. « On certain formal properties of grammars ». In: *Information and control* 2.2 (1959), pp. 137–167 (cit. on p. 128).

[CHS05] Ran Canetti, Shai Halevi, and Michael Steiner. « Hardness Amplification of Weakly Verifiable Puzzles ». In: *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*. Ed. by Joe Kilian. Vol. 3378. Lecture Notes in Computer Science. Springer, 2005, pp. 17–33. ISBN: 3-540-24573-1. DOI: 10.1007/978-3-540-30576-7_2 (cit. on p. 117).

[CHS06] Ran Canetti, Shai Halevi, and Michael Steiner. « Mitigating Dictionary Attacks on Password-Protected Local Storage ». In: *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*. Ed. by Cynthia Dwork. Vol. 4117. Lecture Notes in Computer Science. Springer, 2006, pp. 160–179. ISBN: 3-540-37432-9. DOI: 10.1007/11818175_10 (cit. on p. 117).

[Cio12] Oana Ciobotaru. « On the (non-) equivalence of UC security notions ». In: *Provable Security*. Springer, 2012, pp. 104–124 (cit. on p. 73).

[CKP04] Liqun Chen, Caroline Kudla, and Kenneth G. Paterson. « Concurrent Signatures ». In: *Advances in Cryptology – EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Interlaken, Switzerland: Springer, Heidelberg, Germany, May 2004, pp. 287–305 (cit. on pp. 55, 57, 58, 60).

[Cle86] Richard Cleve. « Limits on the Security of Coin Flips when Half the Processors Are Faulty (Extended Abstract) ». In: *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*. Ed. by Juris Hartmanis. ACM, 1986, pp. 364–369. ISBN: 0-89791-193-8. DOI: 10.1145/12130.12168 (cit. on pp. 9, 55).

[CLSC05] Kumar Chellapilla, Kevin Larson, Patrice Y. Simard, and Mary Czerwinski. « Designing human friendly human interaction proofs (HIPs) ». In: *Proceedings of the 2005 Conference on Human Factors in Computing Systems, CHI 2005, Portland, Oregon, USA, April 2-7, 2005*. Ed. by Gerrit C. van der Veer and Carolyn Gale. ACM, 2005, pp. 711–720. ISBN: 1-58113-998-5. DOI: 10.1145/1054972.1055070 (cit. on p. 121).

[CM99] J. Camenisch and M. Michels. « Proving that a number is the product of two safe primes ». In: *Advances in Cryptology – EUROCRYPT'99*. Springer-Verlag, 1999, pp. 107–122 (cit. on p. 43).

[CND+06] J.-S. Coron, D. Naccache, Y. Desmedt, A. Odlyzko, and J. P. Stern. « Index Calculation Attacks on RSA Signature and Encryption ». In: *Designs, Codes and Cryptography* 38.1 (2006), pp. 41–53 (cit. on p. 20).

[CNS99] J.-S. Coron, D. Naccache, and J. P. Stern. « On the Security of RSA Padding ». In: *Advances in Cryptology - CRYPTO'99*. Vol. 1666. Lecture Notes in Computer Science. Springer-Verlag, 1999, pp. 1–18 (cit. on p. 20).

[Coc69] John Cocke. « Programming languages and their compilers: Preliminary notes ». In: (1969) (cit. on p. 129).

[CS98] R. Cramer and V. Shoup. « A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack ». In: *Advances in Cryptology - CRYPTO'98*. Vol. 1462. Lecture Notes in Computer Science. Springer, 1998, pp. 13–25 (cit. on p. 13).

[CSRL01] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. 2nd. McGraw-Hill Higher Education, 2001 (cit. on p. 90).

[Dam10] I. Damgård. *On Σ Protocols*. http://www.cs.au.dk/~ivan/Sigma.pdf. 2010 (cit. on pp. 71, 82).

[Dan51] George B Dantzig. « Maximization of a Linear Function of Variables Subject to Linear Inequalities ». In: *Activity Analysis of Production and Allocation* (1951) (cit. on pp. 84, 85).

[DDN00] D. Dolev, C. Dwork, and M. Naor. « Non-Malleable Cryptography ». In: *SIAM Journal on Computing*. Society for Industrial and Applied Mathematics, 2000, pp. 542–552 (cit. on p. 25).

[Del07] Cécile Delerablée. « Identity-based broadcast encryption with constant size ciphertexts and private keys ». In: *ASIACRYPT '07*. Vol. 4833. LNCS. Springer Berlin Heidelberg, 2007, pp. 200–215 (cit. on pp. 139, 143).

[DF03] Yevgeniy Dodis and Nelly Fazio. « Public key broadcast encryption for stateless receivers ». In: *Digital Rights Management*. Vol. 2696. LNCS. Springer Berlin Heidelberg, 2003, pp. 61–80 (cit. on p. 139).

[DGN03] Cynthia Dwork, Andrew Goldberg, and Moni Naor. « On Memory-Bound Functions for Fighting Spam ». In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 426–444. ISBN: 3-540-40674-3. DOI: 10.1007/978-3-540-45146-4_25 (cit. on p. 72).

[DH76] W. Diffie and M. E. Hellman. « New Directions in Cryptography ». In: *IEEE Transactions on Information Theory* 22.6 (Nov. 1976), pp. 644–654 (cit. on pp. 11–13, 19, 24).

[DN92] Cynthia Dwork and Moni Naor. « Pricing via Processing or Combatting Junk Mail ». In: *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*. Ed. by Ernest F. Brickell. Vol. 740. Lecture Notes in Computer Science. Springer, 1992, pp. 139–147. ISBN: 3-540-57340-2. DOI: 10.1007/3-540-48071-4_10 (cit. on p. 72).

[DNW05] Cynthia Dwork, Moni Naor, and Hoeteck Wee. « Pebbling and Proofs of Work ». In: *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*. Ed. by Victor Shoup. Vol. 3621. Lecture Notes in Computer Science. Springer, 2005, pp. 37–54. ISBN: 3-540-28114-2. DOI: 10.1007/11535218_3 (cit. on p. 72).

[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. « Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys ». In: *Pairing '07*. Vol. 4575. LNCS. Springer Berlin Heidelberg, 2007, pp. 39–59 (cit. on p. 139).

[DT06a] George B Dantzig and Mukund N Thapa. *Linear programming 1: Introduction*. Springer Science & Business Media, 2006 (cit. on pp. 84, 85).

[DT06b] George B Dantzig and Mukund N Thapa. *Linear programming 2: Theory and extensions*. Springer Science & Business Media, 2006 (cit. on pp. 84, 85).

[DWQ+14] Hua Deng, Qianhong Wu, Bo Qin, Josep Domingo-Ferrer, Lei Zhang, Jianwei Liu, and Wenchang Shi. « Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts ». In: *Information Sciences* 275 (2014), pp. 370–384 (cit. on p. 138).

[Dzi10] Stefan Dziembowski. « How to Pair with a Human ». In: *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*. Ed. by Juan A. Garay and Roberto De Prisco. Vol. 6280. Lecture Notes in Computer Science.

Springer, 2010, pp. 200–218. ISBN: 978-3-642-15316-7. DOI: 10.1007/978-3-642-15317-4_14 (cit. on p. 117).
[Eco11] Umberto Eco. *Il pendolo di Foucault*. Bompiani, 2011 (cit. on p. 133).
[EDHS07] Jeremy Elson, John R. Douceur, Jon Howell, and Jared Saul. « Asirra: a CAPTCHA that exploits interest-aligned manual image categorization ». In: *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*. Ed. by Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson. ACM, 2007, pp. 366–374. ISBN: 978-1-59593-703-2. DOI: 10.1145/1315245.1315291 (cit. on p. 117).
[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. « A randomized protocol for signing contracts ». In: *Communications of the ACM* 28.6 (1985), pp. 637–647 (cit. on p. 9).
[El 84] T. El Gamal. « A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms ». In: *Advances in Cryptology - CRYPTO'84*. Vol. 196. Lecture Notes in Computer Science. Springer, 1984, pp. 10–18 (cit. on pp. 12, 21).
[ElG84] Taher ElGamal. « A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms ». In: *Advances in Cryptology – CRYPTO'84*. Ed. by G. R. Blakley and David Chaum. Vol. 196. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1984, pp. 10–18 (cit. on pp. 56, 60).
[EMV] EMVCo. http://www.emvco.com/specifications.aspx (cit. on p. 97).
[EMV08a] EMVCo. *EMV Specification (Book 1) – version 4.2*. June 2008 (cit. on pp. 97, 102).
[EMV08b] EMVCo. *EMV Specification (Book 2) – version 4.2*. June 2008 (cit. on pp. 97, 104).
[EMV08c] EMVCo. *EMV Specification (Book 3) – version 4.2*. June 2008 (cit. on pp. 97, 103, 104).
[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. « Zero-Knowledge Proofs of Identity ». In: *J. Cryptology* 1.2 (1988), pp. 77–94. DOI: 10.1007/BF02351717 (cit. on pp. 28, 56, 72, 82, 83, 85, 89).
[FGM+15a] Houda Ferradi, Rémi Géraud, Diana Maimuţ, David Naccache, and Amaury de Wargny. *Regulating the Pace of von Neumann Correctors*. Cryptology ePrint Archive, Report 2015/849. http://eprint.iacr.org/2015/849. 2015 (cit. on pp. 33, 36).
[FGM+15b] Houda Ferradi, Rémi Géraud, Diana Maimuţ, David Naccache, and Hang Zhou. *Backtracking-Assisted Multiplication*. Cryptology ePrint Archive, Report 2015/787. http://eprint.iacr.org/2015/787. 2015 (cit. on pp. 33, 36).
[FGM+16a] Houda Ferradi, Rémi Géraud, Diana Maimuţ, David Naccache, and David Pointcheval. « Legally Fair Contract Signing Without Keystones ». In: *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Cham: Springer International Publishing, 2016, pp. 175–190. ISBN: 978-3-319-39555-5. DOI: 10.1007/978-3-319-39555-5_10. URL: http://dx.doi.org/10.1007/978-3-319-39555-5_10 (cit. on p. 30).
[FGM+16b] Houda Ferradi, Rémi Géraud, Diana Maimuţ, David Naccache, and David Pointcheval. *Legally Fair Contract Signing Without Keystones*. Cryptology ePrint Archive, Report 2016/363. http://eprint.iacr.org/2016/363. 2016 (cit. on p. 30).
[FGN16a] Houda Ferradi, Rémi Géraud, and David Naccache. *Slow Motion Zero Knowledge Identifying With Colliding Commitments*. Cryptology ePrint Archive, Report 2016/399. http://eprint.iacr.org/2016/399. 2016 (cit. on p. 30).
[FGN16b] Houda Ferradi, Rémi Géraud, and David Naccache. « Slow Motion Zero Knowledge Identifying with Colliding Commitments ». In: *Information Security and Cryptology: 11th International Conference, Inscrypt 2015, Beijing, China, November 1-3, 2015, Revised Selected Papers*. Ed. by Dongdai Lin, XiaoFeng Wang, and Moti Yung. Cham: Springer International Publishing, 2016, pp. 381–396. ISBN: 978-3-319-38898-4. DOI: 10.1007/978-3-319-38898-4_22. URL: http://dx.doi.org/10.1007/978-3-319-38898-4_22 (cit. on p. 30).
[FGNT15] Houda Ferradi, Rémi Géraud, David Naccache, and Assia Tria. *When Organized Crime Applies Academic Results - A Forensic Analysis of an In-Card Listening Device*. Cryptology ePrint Archive, Report 2015/963. http://eprint.iacr.org/2015/963. 2015 (cit. on p. 31).

[FN94] Amos Fiat and Moni Naor. « Broadcast encryption ». In: *CRYPTO '93*. Vol. 773. LNCS. Springer Berlin Heidelberg, 1994, pp. 480–491 (cit. on p. 139).

[FO97] E. Fujisaki and T. Okamoto. « Statistical zero knowledge protocols to prove modular polynomial relations ». In: *Advances in Cryptology – CRYPTO'97*. Springer-Verlag, 1997, pp. 16–30 (cit. on p. 44).

[FO98] E. Fujisaki and T. Okamoto. « A practical and provably secure scheme for publicly verifiable secret sharing and its applications ». In: *Advances in Cryptology – CRYPTO'98*. Springer-Verlag, 1998, pp. 32–46 (cit. on p. 44).

[FS87] A. Fiat and A. Shamir. « How to Prove Yourself: Practical Solutions to Identification and Signature Problems ». In: *Advances in Cryptology - CRYPTO'86*. Vol. 263. Lecture Notes in Computer Science. Springer-Verlag, 1987, pp. 186–194 (cit. on pp. 21, 89, 92).

[GBI+13] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. « Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks ». In: *CoRR* abs/1312.6082 (2013). URL: http://arxiv.org/abs/1312.6082 (cit. on p. 120).

[GH09] Craig Gentry and Shai Halevi. « Hierarchical identity based encryption with polynomially many levels ». In: *TCC '09*. Vol. 5444. LNCS. Springer Berlin Heidelberg, 2009, pp. 437–456 (cit. on p. 140).

[GHKL08] S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. « Complete fairness in secure two-party computation ». In: *40th Annual ACM Symposium on Theory of Computing*. Ed. by Richard E. Ladner and Cynthia Dwork. Victoria, British Columbia, Canada: ACM Press, May 2008, pp. 413–422 (cit. on pp. 9, 55).

[GHNK16] Msgna Mehari G., Ferradi Houda, Akram Raja Naeem, and Markantonakis Konstantinos. « Secure Application Execution in Mobile Devices ». In: *The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*. Ed. by Ryan A. Peter Y., Naccache David, and Quisquater Jean-Jacques. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 417–438. ISBN: 978-3-662-49301-4. DOI: 10.1007/978-3-662-49301-4_26. URL: http://dx.doi.org/10.1007/978-3-662-49301-4_26 (cit. on pp. 33, 34).

[Gir00] Marc Girault. « Low-Size Coupons for Low-Cost IC Cards ». In: *Smart Card Research and Advanced Applications: IFIP TC8 / WG8.8 Fourth Working Conference on Smart Card Research and Advanced Applications September 20–22, 2000, Bristol, United Kingdom*. Ed. by Josep Domingo-Ferrer, David Chan, and Anthony Watson. Boston, MA: Springer US, 2000, pp. 39–49. ISBN: 978-0-387-35528-3. DOI: 10.1007/978-0-387-35528-3_3. URL: http://dx.doi.org/10.1007/978-0-387-35528-3_3 (cit. on p. 71).

[Gir90] Marc Girault. « An Identity-based Identification Scheme Based on Discrete Logarithms Modulo a Composite Number ». In: *Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*. Ed. by Ivan Damgård. Vol. 473. Lecture Notes in Computer Science. Springer, 1990, pp. 481–486. ISBN: 3-540-53587-X (cit. on pp. 72, 74, 76).

[GJM99] Juan A. Garay, Markus Jakobsson, and Philip D. MacKenzie. « Abuse-Free Optimistic Contract Signing ». In: *Advances in Cryptology – CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1999, pp. 449–466 (cit. on pp. 56, 58).

[GK15] Shafi Goldwasser and Yael Tauman Kalai. *Cryptographic Assumptions: A Position Paper*. Cryptology ePrint Archive, Report 2015/907. http://eprint.iacr.org/2015/907. 2015 (cit. on p. 24).

[GL91] Shafi Goldwasser and Leonid A. Levin. « Fair Computation of General Functions in Presence of Immoral Majority ». In: *Advances in Cryptology – CRYPTO'90*. Ed. by Alfred J. Menezes and Scott A. Vanstone. Vol. 537. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1991, pp. 77–93 (cit. on p. 55).

[GM82] S. Goldwasser and S. Micali. « Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information ». In: *Proceedings of the 14th Annual ACM Symposium on Theory of Computing - STOC'82*. ACM, 1982, pp. 365–377 (cit. on pp. 14, 18).

[GM84] S. Goldwasser and S. Micali. « Probabilistic Encryption ». In: *Journal of Computer and Systems Sciences* 28.2 (1984), pp. 270–299 (cit. on pp. 8, 22, 25).

[GMPY06] Juan A. Garay, Philip D. MacKenzie, Manoj Prabhakaran, and Ke Yang. « Resource Fairness and Composability of Cryptographic Protocols ». In: *TCC 2006: 3rd Theory of Cryptography Conference*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. Lecture Notes in Computer Science. New York, NY, USA: Springer, Heidelberg, Germany, Mar. 2006, pp. 404–428 (cit. on p. 55).

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. « The Knowledge Complexity of Interactive Proof-Systems ». In: *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*. STOC'85. Providence, Rhode Island, USA: ACM, 1985, pp. 291–304 (cit. on pp. 18, 27, 28, 71, 82, 89).

[GMR98] R. Gennaro, D. Micciancio, and T. Rabin. « An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products ». In: *Proceedings of the 5th ACM conference on Computer and Communications Security*. ACM, 1998, pp. 67–72 (cit. on pp. 43, 44).

[GMW87a] O. Goldreich, S. Micali, and A. Wigderson. « How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design ». In: *Advances in Cryptology – CRYPTO'86*. Springer-Verlag, 1987, pp. 171–185 (cit. on p. 43).

[GMW87b] Oded Goldreich, Silvio Micali, and Avi Wigderson. « How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority ». In: *19th Annual ACM Symposium on Theory of Computing*. Ed. by Alfred Aho. New York City,, New York, USA: ACM Press, May 1987, pp. 218–229 (cit. on pp. 9, 55).

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. « Proofs that Yield Nothing But Their Validity for All Languages in NP Have Zero-Knowledge Proof Systems ». In: *J. ACM* 38.3 (1991), pp. 691–729. DOI: 10.1145/116825.116852 (cit. on pp. 71, 72, 82).

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Vol. 2. Cambridge, UK: Cambridge University Press, 2004. ISBN: ISBN 0-521-83084-2 (hardback) (cit. on pp. 9, 55).

[Gol83] Oded Goldreich. « A Simple Protocol for Signing Contracts ». In: *Advances in Cryptology, Proceedings of CRYPTO '83, Santa Barbara, California, USA, August 21-24, 1983*. Ed. by David Chaum. Plenum Press, New York, 1983, pp. 133–136 (cit. on p. 57).

[Goo] Google. *Google reCAPTCHA*. URL: https://developers.google.com/recaptcha (cit. on p. 117).

[GOS06] J. Groth, R. Ostrovsky, and A. Sahai. « Perfect non-interactive zero knowledge for NP ». In: *Advances in Cryptology – EUROCRYPT 2006*. Springer, 2006, pp. 339–358 (cit. on p. 43).

[GP88] J. Van de Graaf and R. Peralta. « A simple and secure way to show the validity of your public key ». In: *Advances in Cryptology – CRYPTO'87*. Springer Berlin Heidelberg, 1988, pp. 128–134 (cit. on p. 43).

[GPS06] Marc Girault, Guillaume Poupard, and Jacques Stern. « On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order ». In: *J. Cryptology* 19.4 (2006), pp. 463–487. DOI: 10.1007/s00145-006-0224-0 (cit. on pp. 21, 56, 60, 71, 74, 77, 78, 80, 81).

[GPS08] Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. « Pairings for cryptographers ». In: *Discrete Applied Mathematics* 156.16 (2008), pp. 3113–3121 (cit. on p. 13).

[GQ88] L. C. Guillou and J.-J. Quisquater. « A Practical Zero-knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory ». In: *Advances in Cryptology - EUROCRYPT'88*. Vol. 330. Lecture Notes in Computer Science. Springer, 1988, pp. 123–128 (cit. on p. 89).

[GS02] Craig Gentry and Alice Silverberg. « Hierarchical ID-based cryptography ». In: *ASIACRYPT '02*. Vol. 2501. LNCS. Springer Berlin Heidelberg, 2002, pp. 548–566 (cit. on pp. 89, 138, 139, 159).

[GS94a] M. Girault and J. Stern. « On the Length of Cryptographic Hash-Values Used in Identification Schemes ». In: *Advances in Cryptology - CRYPTO'94*. 1994, pp. 202–215 (cit. on p. 71).

[GS94b] Marc Girault and Jacques Stern. « On the Length of Cryptographic Hash-Values Used in Identification Schemes ». In: *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*. Ed. by Yvo Desmedt. Vol. 839. Lecture Notes in Computer Science. Springer, 1994, pp. 202–215. ISBN: 3-540-58333-5. DOI: 10.1007/3-540-48658-5_21 (cit. on pp. 74, 78).

[GW09] Craig Gentry and Brent Waters. « Adaptive security in broadcast encryption systems (with short ciphertexts) ». In: *EUROCRYPT '09*. Vol. 5479. LNCS. Springer Berlin Heidelberg, 2009, pp. 171–188 (cit. on pp. 139, 143).

[Hei01] Christopher D. Manning and Heinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 2001. ISBN: 978-0-262-13360-9 (cit. on p. 133).

[HL02] Jeremy Horwitz and Ben Lynn. « Toward Hierarchical Identity-Based Encryption ». In: *Advances in Cryptology — EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques Amsterdam, The Netherlands, April 28 – May 2, 2002 Proceedings*. Ed. by Lars R. Knudsen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 466–481. ISBN: 978-3-540-46035-0. DOI: 10.1007/3-540-46035-7_31. URL: http://dx.doi.org/10.1007/3-540-46035-7_31 (cit. on pp. 82, 139, 159).

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010 (cit. on p. 71).

[HPM94] Patrick Horster, Holger Petersen, and Markus Michels. « Meta-El-Gamal Signature Schemes ». In: *ACM CCS 94: 2nd Conference on Computer and Communications Security*. Fairfax, Virginia, USA: ACM Press, 1994, pp. 96–107 (cit. on pp. 56, 60).

[HRD+16] Ferradi Houda, Géraud Rémi, Maimuţ Diana, Naccache David, and Zhou Hang. « Backtracking-Assisted Multiplication ». In: *Arctic Crypt 2016, July 17-22, Longyearbyen, Svalbard, Norway. Pre-Proceedings*. 2016 (cit. on pp. 33, 36).

[HRDA16] Ferradi Houda, Géraud Rémi, Naccache David, and Tria Assia. « When organized crime applies academic results: a forensic analysis of an in-card listening device ». In: *Journal of Cryptographic Engineering* 6.1 (2016), pp. 49–59. ISSN: 2190-8516. DOI: 10.1007/s13389-015-0112-3. URL: http://dx.doi.org/10.1007/s13389-015-0112-3 (cit. on p. 31).

[HRN16] Ferradi Houda, Géraud Rémi, and David Naccache. « Human Public-Key Encryption ». In: *Mycrypt 2016: Paradigm-shifting Crypto Kuala Lumpur, Malaysia, December 1-2, 2016. Proceedings*. Ed. by Phan Raphael C.-W. and Yung Moti. Springer International Publishing, 2016 (cit. on p. 32).

[HS02] Dani Halevy and Adi Shamir. « The LSD broadcast encryption scheme ». In: *CRYPTO '02*. Vol. 2442. LNCS. Springer Berlin Heidelberg, 2002, pp. 47–60 (cit. on p. 139).

[HYH+16] Jinguang Huan, Ye Yang, Xinyi Huang, Tsz Hon Yuen, Jiguo Li, and Jie Cao. « Accountable mobile E-commerce scheme via identity-based plaintext-checkable encryption ». In: *Information Sciences* 345 (2016), pp. 143–155 (cit. on p. 151).

[IBM] IBM. *4764 PCI-X Cryptographic Coprocessor*. See http://www-03.ibm.com/security/cryptocards/pcixcc/overperformance.shtml (cit. on p. 51).

[JD12] Markus Jakobsson and Mayank Dhiman. « The Benefits of Understanding Passwords ». In: *7th USENIX Workshop on Hot Topics in Security, HotSec'12, Bellevue, WA, USA, August 7, 2012*. Ed. by Patrick Traynor. USENIX Association, 2012 (cit. on pp. 125, 128).

[JJ02] A. Juels and J.Guajardo. « RSA key generation with verifiable randomness ». In: *Public Key Cryptography*. Springer Berlin Heidelberg, 2002, pp. 357–374 (cit. on pp. 43, 44).

[JR14] Ari Juels and Thomas Ristenpart. « Honey Encryption: Security Beyond the Brute-Force Bound ». In: *Advances in Cryptology – EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. Lecture Notes in Computer Science. Copenhagen, Denmark: Springer, Heidelberg, Germany, May 2014, pp. 293–310. DOI: 10.1007/978-3-642-55220-5_17 (cit. on pp. 32, 115, 122, 125–127, 131, 165).

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. « Designated Verifier Proofs and Their Applications ». In: *Advances in Cryptology – EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Saragossa, Spain: Springer, Heidelberg, Germany, May 1996, pp. 143–154 (cit. on p. 55).

[Jus] Ministère de la Justice (France). *French prosecution case number 1116791060.* (Cit. on p. 98).

[Kal13] Burton Kaliski. « PKCS#5: Password-Based Cryptography Specifications Version 2.0. » In: *Request for Comments 2898*. 2013 (cit. on p. 125).

[Kar84] Narendra Karmarkar. « A new polynomial-time algorithm for linear programming ». In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. ACM. 1984, pp. 302–311 (cit. on p. 84).

[Kas65] Tadao Kasami. *An efficient recognition and syntax analysis algorithm for context-free languages.* Tech. rep. DTIC Document, 1965 (cit. on p. 129).

[Ker83] Auguste Kerckhoffs. *La cryptographie militaire, ou, Des chiffres usités en temps de guerre: avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef*. Librairie militaire de L. Baudoin, 1883 (cit. on p. 8).

[KKM+12] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. « Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms ». In: *2012 IEEE Symposium on Security and Privacy*. San Francisco, California, USA: IEEE Computer Society Press, May 2012, pp. 523–537 (cit. on p. 128).

[KKM12] Saqib A Kakvi, Eike Kiltz, and Alexander May. « Certifying RSA ». In: *Advances in Cryptology–ASIACRYPT 2012*. Springer, 2012, pp. 404–414 (cit. on p. 43).

[KM03] Dan Klein and Christopher D Manning. « Accurate unlexicalized parsing ». In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics. 2003, pp. 423–430 (cit. on p. 128).

[Kob87] N. Koblitz. « Elliptic Curve Cryptosystems ». In: *Mathematics of Computation* 48.177 (1987), pp. 203–209 (cit. on p. 24).

[KOPW13] Abishek Kumarasubramanian, Rafail Ostrovsky, Omkant Pandey, and Akshay Wadia. « Cryptography Using Captcha Puzzles ». In: *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. Lecture Notes in Computer Science. Springer, 2013, pp. 89–106. ISBN: 978-3-642-36361-0. DOI: 10.1007/978-3-642-36362-7_7 (cit. on p. 117).

[KSAS15] Jongkil Kim, Willy Susilo, Man Ho Au, and Jennifer Seberry. « Adaptively secure identity-based broadcast encryption with a constant-sized ciphertext ». In: *IEEE Transactions on Information Forensics and Security* 10.3 (2015), pp. 679–693 (cit. on p. 139).

[KY08] Christos Koufogiannakis and Neal E. Young. « Beating Simplex for Fractional Packing and Covering Linear Programs ». In: *CoRR* abs/0801.1987 (2008). URL: http://arxiv.org/abs/0801.1987 (cit. on p. 84).

[KY10] Okamura Keisuke and Oyama Yoshihiro. « Load-based covert channels between Xen virtual machines ». In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. SAC '10. Sierre, Switzerland: ACM, 2010, pp. 173–180. ISBN: 978-1-60558-639-7. DOI: 10.1145/1774088.1774125. URL: http://doi.acm.org/10.1145/1774088.1774125 (cit. on p. 34).

[Len84] HW Lenstra. « Integer programming and cryptography ». In: *The Mathematical Intelligencer* 6.3 (1984), pp. 14–21 (cit. on pp. 24, 84).

[LHAS14] Zhiwei Li, Warren He, Devdatta Akhawe, and Dawn Song. « The Emperor's New Password Manager: Security Analysis of Web-based Password Managers ». In: *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*. Ed. by Kevin Fu and Jaeyeon Jung. USENIX Association, 2014, pp. 465–479 (cit. on p. 125).

[Lin08] Andrew Y. Lindell. « Legally-Enforceable Fairness in Secure Two-Party Computation ». In: *Topics in Cryptology – CT-RSA 2008*. Ed. by Tal Malkin. Vol. 4964. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer, Heidelberg, Germany, Apr. 2008, pp. 121–137 (cit. on pp. 9, 55, 57).

[LLW+16a] Weiran Liu, Jianwei Liu, Qianhong Wu, Bo Qin, and Yan Li. « Practical chosen-ciphertext secure hierarchical identity-based broadcast encryption ». In: *International Journal of Information Security* 15.1 (2016), pp. 35–50 (cit. on p. 143).

[LLW+16b] Weiran Liu, Jianwei Liu, Qianhong Wu, Bo Qin, David Naccache, and Houda Ferradi. *Compact CCA2-secure Hierarchical Identity-Based Broadcast Encryption for Fuzzy-entity Data Sharing*. Cryptology ePrint Archive, Report 2016/634. http://eprint.iacr.org/2016/634. 2016 (cit. on p. 33).

[LLWQ14] Weiran Liu, Jianwei Liu, Qianhong Wu, and Bo Qin. « Hierarchical Identity-Based Broadcast Encryption ». In: *ACISP '14*. Vol. 8544. LNCS. Springer Berlin Heidelberg, 2014, pp. 242–257 (cit. on p. 139).

[LPQ12] Benoît Libert, Kenneth G Paterson, and Elizabeth A Quaglia. « Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model ». In: *PKC '12*. Vol. 7293. LNCS. Springer Berlin Heidelberg, 2012, pp. 206–224 (cit. on p. 139).

[LS98] M. Liskov and B. Silverman. *A statistical-limited knowledge proof for secure RSA keys*. Manuscript. 1998 (cit. on p. 44).

[LV08] C. Lavault and M. Valencia-Pabon. « A Distributed Approximation Algorithm for the Minimum Degree Minimum Weight Spanning Trees ». In: *Journal of Parallel and Distributed Computing* 68.2 (2008), pp. 200–208 (cit. on p. 90).

[LW10] Allison Lewko and Brent Waters. « New techniques for dual system encryption and fully secure HIBE with short ciphertexts ». In: *TCC '10*. Vol. 5978. LNCS. Springer Berlin Heidelberg, 2010, pp. 455–479 (cit. on pp. 140, 143–145, 147).

[LW12] Allison Lewko and Brent Waters. « New proof methods for attribute-based encryption: Achieving full security through selective techniques ». In: *CRYPTO '12*. Vol. 7417. LNCS. Springer Berlin Heidelberg, 2012, pp. 180–198 (cit. on pp. 140, 145).

[Mao98] W. Mao. « Verifiable partial sharing of integer factors ». In: *Selected Areas in Cryptography – SAC'98*. Springer-Verlag, 1998, pp. 94–105 (cit. on p. 44).

[MDAB10] Steven J Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. « Chip and PIN is Broken ». In: *2010 IEEE Symposium on Security and Privacy*. IEEE. 2010, pp. 433–446 (cit. on pp. 31, 69, 97, 164).

[Mer78] Ralph C. Merkle. « Secure Communications Over Insecure Channels ». In: *Commun. ACM* 21.4 (1978), pp. 294–299. DOI: 10.1145/359460.359473 (cit. on pp. 24, 119).

[MGW03] A. J. Mooij, N. Goga, and J. W. Wesselink. *A Distributed Spanning Tree Algorithm for Topology-Aware Networks*. Technische Universiteit Eindhoven, Department of Mathematics and Computer Science, 2003 (cit. on pp. 90, 91).

[MHH+14] Abdalla Michel, Chabanne Hervé, Ferradi Houda, Jainski Julien, and Naccache David. « Improving Thomlinson-Walker's Software Patching Scheme Using Standard Cryptographic and Statistical Tools ». In: *Information Security Practice and Experience: 10th International Conference, ISPEC 2014, Fuzhou, China, May 5-8, 2014. Proceedings*. Ed. by Huang Xinyi and Zhou Jianying. Cham: Springer International Publishing, 2014, pp. 8–14. ISBN: 978-3-319-06320-1. DOI: 10.1007/978-3-319-06320-1_2. URL: http://dx.doi.org/10.1007/978-3-319-06320-1_2 (cit. on p. 33).

[MHS03] Marco Casassa Mont, Keith Harrison, and Martin Sadler. « The HP Time Vault Service: Exploiting IBE for Timed Release of Confidential Information ». In: *Proceedings of the 12th International Conference on World Wide Web*. WWW '03. Budapest, Hungary: ACM, 2003, pp. 160–169. ISBN: 1-58113-680-3. DOI: 10.1145/775152.775175. URL: http://doi.acm.org/10.1145/775152.775175 (cit. on p. 159).

[Mic03] Silvio Micali. « Simple and fast optimistic protocols for fair electronic exchange ». In: *22nd ACM Symposium Annual on Principles of Distributed Computing*. Ed. by Elizabeth Borowsky and Sergio Rajsbaum. Boston, Massachusetts, USA: Association for Computing Machinery, July 2003, pp. 12–19 (cit. on p. 55).

[Mic93] S. Micali. « Fair Public Key Cryptosystems ». In: *Advances in Cryptology – CRYPTO'92*. Springer Berlin Heidelberg, 1993, pp. 113–138 (cit. on p. 44).

[Mil86] V. S. Miller. « Use of Elliptic Curves in Cryptography ». In: *Advances in Cryptology - CRYPTO 85*. Vol. 218. Lecture Notes in Computer Sciences. Springer-Verlag, 1986, pp. 417–426 (cit. on p. 24).

[Mis98] J.-F. Misarsky. « How (Not) to Design RSA Signature Schemes ». In: *Public-Key Cryptography*. Vol. 1431. Lecture Notes in Computer Science. Springer-Verlag, 1998, pp. 14–28 (cit. on p. 20).

[MMC06] Keith Mayes, K Markantonakis, and C Chen. « Smart Card Platform Fingerprinting ». In: *The Global Journal of Advanced Card Technology* (2006). Ed. by Mark Locke, pp. 78–82 (cit. on p. 111).

[MMV11] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. « Time-Lock Puzzles in the Random Oracle Model ». In: *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*. Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 39–50. ISBN: 978-3-642-22791-2. DOI: 10.1007/978-3-642-22792-9_3 (cit. on pp. 72, 73).

[MN94] David M'Raïhi and David Naccache. « Couponing Scheme Reduces Computational Power Requirements for DSS Signatures ». In: *Proceedings of CardTech/SecurTech*. 1994, pp. 99–104 (cit. on pp. 71, 72, 76).

[MN96] D. M'Raïhi and D. Naccache. « Batch Exponentiation: A Fast DLP-Based Signature Generation Strategy ». In: *Proceedings of the 3rd ACM Conference on Computer and Communications Security - CCS'96*. ACM, 1996, pp. 58–61 (cit. on p. 95).

[MP06] Silvio Micali and Rafael Pass. « Local zero knowledge ». In: *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*. Ed. by Jon M. Kleinberg. ACM, 2006, pp. 306–315. ISBN: 1-59593-134-1. DOI: 10.1145/1132516.1132561 (cit. on pp. 71, 72).

[MRN16] Ferradi Houda Marc Beunardeau, Géraud Rémi, and David Naccache. « Honey Encryption for Language: Robbing Shannon to Pay Turing? » In: *Mycrypt 2016: Paradigm-shifting Crypto Kuala Lumpur, Malaysia, December 1-2, 2016. Proceedings*. Ed. by Phan Raphael C.-W. and Yung Moti. Springer International Publishing, 2016 (cit. on p. 32).

[MSA+11] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, and Jon Orwant. « Quantitative analysis of culture using millions of digitized books ». In: *Science* 331.6014 (2011), pp. 176–182 (cit. on pp. 18, 131).

[Mur83] Katta G Murty. « Linear programming ». In: (1983) (cit. on p. 84).

[MYLL14] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. « A Study of Probabilistic Password Models ». In: *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 2014, pp. 689–704. ISBN: 978-1-4799-4686-0 (cit. on p. 128).

[NASK14] Mir Tafseer Nayeem, Md. Mamunur Rashid Akand, Nazmus Sakib, and Md. Wasi Ul Kabir. « Design of a Human Interaction Proof (HIP) using human cognition in contextual natural conversation ». In: *IEEE 13th International Conference on Cognitive Informatics and Cognitive Computing, ICCI*CC 2014, London, UK, August 18-20, 2014*. IEEE, 2014, pp. 146–154. DOI: 10.1109/ICCI-CC.2014.6921454 (cit. on p. 117).

[NES] NESSIE. « List of NESSIE Submissions as Originally Submitted ». In: *NESSIE Project*. URL: https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions.html (cit. on p. 21).

[Neu51] J. von Neumann. « Various Techniques used in Connection with Random Digits ». In: *National Bureau of Standards Applied Math Series* 12 (1951), pp. 36–38 (cit. on p. 35).

[NM95] David Naccache and David M'Raihi. *Electronic Signature Method for Smart Cards*. filed April 20, 1995. 1995 (cit. on p. 71).

[NS97] D. Naccache and J. Stern. « A New Public-Key Cryptosystem ». In: *Advances in Cryptology - EUROCRYPT'97*. Vol. 1233. Lecture Notes in Computer Science. Springer, 1997, pp. 27–36 (cit. on p. 24).

[NY90] Moni Naor and Moti Yung. « Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks ». In: *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*. Ed. by Harriet Ortiz. ACM, 1990, pp. 427–437. ISBN: 0-89791-361-2. DOI: 10.1145/100216.100273 (cit. on pp. 24, 118).

[Ora] Oracle. *Sun Crypto Accelerator SCA 6000*. See http://www.oracle.com/us/products/servers-storage/036080.pdf (cit. on p. 51).

[OW96] P. C. van Oorschot and M. J. Wiener. « On Diffie-Hellman Key Agreement with Short Exponents ». In: *Advances in Cryptology — EUROCRYPT'96*. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 332–343 (cit. on p. 22).

[Pai99] P. Paillier. « Public-Key Cryptosystems Based on Composite Degree Residuosity Classes ». In: *Advances in Cryptology - EUROCRYPT'99*. Vol. 1592. Lecture Notes in Computer Science. Springer, 1999, pp. 223–238 (cit. on p. 14).

[Pin03] Benny Pinkas. « Fair Secure Two-Party Computation ». In: *Advances in Cryptology – EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, May 2003, pp. 87–105 (cit. on p. 55).

[PLL15] Jong Hwan Park, Kwangsu Lee, and Dong Hoon Lee. « New chosen-ciphertext secure identity-based encryption with tight security reduction to the bilinear Diffie-Hellman problem ». In: *Information Sciences* 325 (2015), pp. 256–270 (cit. on p. 139).

[Poi] D. Pointcheval (cit. on p. 67).

[Poi05] D. Pointcheval. « Advanced Course on Contemporary Cryptology ». In: Advanced Courses CRM Barcelona. Birkhäuser Publishers, Basel, June 2005. Chap. Provable Security for Public-Key Schemes, pp. 133–189 (cit. on p. 25).

[Poi95] David Pointcheval. « A New Identification Scheme Based on the Perceptrons Problem ». In: *Advances in Cryptology – EUROCRYPT'95*. Ed. by Louis C. Guillou and Jean-Jacques Quisquater. Vol. 921. Lecture Notes in Computer Science. Saint-Malo, France: Springer, Heidelberg, Germany, May 1995, pp. 319–328 (cit. on pp. 82, 85, 88).

[PQ10] Kenneth G. Paterson and Elizabeth A. Quaglia. « Time-Specific Encryption ». In: *Security and Cryptography for Networks: 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*. Ed. by Juan A. Garay and Roberto De Prisco. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–16. ISBN: 978-3-642-15317-4. DOI: 10.1007/978-3-642-15317-4_1. URL: http://dx.doi.org/10.1007/978-3-642-15317-4_1 (cit. on pp. 157, 158).

[PS00] David Pointcheval and Jacques Stern. « Security Arguments for Digital Signatures and Blind Signatures ». In: *J. Cryptology* 13.3 (2000), pp. 361–396. DOI: 10.1007/s001450010003 (cit. on pp. 21, 22, 56, 57, 64).

[PS96] D. Pointcheval and J. Stern. « Security Proofs for Signature Schemes ». In: *Advances in Cryptology - EUROCRYPT'96*. Vol. 1070. 1996, pp. 387–398 (cit. on pp. 21, 56).

[PS98] Guillaume Poupard and Jacques Stern. « Security Analysis of a Practical "on the fly" Authentication and Signature Generation ». In: *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Springer, 1998, pp. 422–436. ISBN: 3-540-64518-7. DOI: 10.1007/BFb0054143 (cit. on pp. 74, 77).

[PST+02] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. « SPINS: Security Protocols for Sensor Networks ». In: *Wirel. Netw.* 8.5 (Sept. 2002), pp. 521–534. ISSN: 1022-0038 (cit. on p. 89).

[QWZ+12] Bo Qin, Qianhong Wu, Lei Zhang, Oriol Farràs, and Josep Domingo-Ferrer. « Provably secure threshold public-key encryption with adaptive security and short ciphertexts ». In: *Information Sciences* 210 (2012), pp. 67–80 (cit. on p. 139).

[Rab79] M. O. Rabin. *Digitalized Signatures and Public-key Functions as Intractable as Factorization*. Tech. rep. Cambridge, MA, USA, 1979 (cit. on p. 24).

[Rab83] M. O. Rabin. « Transaction Protection by Beacons ». In: *Journal of Computer and System Sciences* 27.2 (1983), pp. 256–257 (cit. on p. 9).

[Rav87] Stathis Zachos. Ravi B. Boppana Johan Hastad. « CDoes co-NP have short interactive proofs? » In: *Information Processing Letters* Volume 25, Issue 2 (1987), pp. 127–132 (cit. on p. 28).

[RG09] Yanli Ren and Dawu Gu. « Fully CCA2 secure identity based broadcast encryption without random oracles ». In: *Information Processing Letters* 109.11 (2009), pp. 527–533 (cit. on p. 139).

[Roo97] Peter de Rooij. « On Schnorr's preprocessing for digital signature schemes ». In: *Journal of Cryptology* 10.1 (1997), pp. 1–16 (cit. on pp. 71, 72).

[RS82] Ronald L Rivest and Adi Shamir. « How to reuse a "write-once" memory ». In: *Information and control* 55.1 (1982), pp. 1–19 (cit. on p. 112).

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. « A Method for Obtaining Digital Signatures and Public-key Cryptosystems ». In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: http://doi.acm.org/10.1145/359340.359342 (cit. on pp. 24, 158).

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. « How to Leak a Secret ». In: *Advances in Cryptology – ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. Lecture Notes in Computer Science. Gold Coast, Australia: Springer, Heidelberg, Germany, Dec. 2001, pp. 552–565 (cit. on p. 55).

[RSW96] R Rivest, A Shamir, and D. Wagner. *Time-lock puzzles and timed-release crypto*. Technical report, MIT/LCS/TR-684. 1996 (cit. on pp. 72–74).

[RW96] Ronald L. Rivest and David A. Wagner. *Time-lock puzzles and timed-release crypto*. Tech. rep. 1996 (cit. on pp. 157, 158).

[RWJL06] Keith Rayner, Sarah J White, Rebecca L Johnson, and Simon P Liversedge. « Raeding wrods with jubmled lettres there is a cost ». In: *Psychological science* 17.3 (2006), pp. 192–193 (cit. on p. 132).

[Sch90] C.-P. Schnorr. « Efficient Identification and Signatures for Smart Cards ». In: *Advances in Cryptology - CRYPTO'89*. Vol. 434. Lecture Notes in Computer Science. Springer, 1990, pp. 239–252 (cit. on pp. 12, 21, 56).

[SD82] Adi Shamir and Whitfield Diffie. « A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem ». In: *In Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*. IEEE, 1982, pp. 145–152 (cit. on p. 24).

[Seu12] Yannick Seurin. « On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model ». In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Cambridge, UK: Springer, Heidelberg, Germany, Apr. 2012, pp. 554–571 (cit. on p. 57).

[SF13] T Souvignet and J Frinken. « Differential power analysis as a digital forensic tool ». In: *Forensic science international* 230.1 (2013), pp. 127–136 (cit. on p. 105).

[Sha48] C. E. Shannon. « A Mathematical Theory of Communication ». In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423 (cit. on pp. 15, 18).

[Sha49] C. E. Shannon. « Communication Theory of Secrecy Systems ». In: *Bell System Technical Journal* 28.4 (1949), pp. 656–715 (cit. on p. 15).

[Sha84] Adi Shamir. « Identity-Based Cryptosystems and Signature Schemes ». In: *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*. Vol. 196. Lecture Notes in Computer Science. Springer, 1984, pp. 47–53. DOI: 10.1007/3-540-39568-7_5 (cit. on p. 159).

[Sha85] Adi Shamir. « Identity-based cryptosystems and signature schemes ». In: *CRYPTO '84*. Vol. 196. LNCS. Springer Berlin Heidelberg, 1985, pp. 47–53 (cit. on pp. 138, 139).

[Sha90] Adi Shamir. « An Efficient Identification Scheme Based on Permuted Kernels (Extended Abstract) (Rump Session) ». In: *Advances in Cryptology – CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1990, pp. 606–609 (cit. on pp. 82, 85, 87).

[SHL+10] Graig Sauer, Jonathan Holman, Jonathan Lazar, Harry Hochheiser, and Jinjuan Feng. « Accessible privacy and security: a universally usable human-interaction proof tool ». In: *Universal Access in the Information Society* 9.3 (2010), pp. 239–248. DOI: 10.1007/s10209-009-0171-2 (cit. on p. 117).

[Sho97] Peter W. Shor. « Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer ». In: *SIAM J. Comput.* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397. DOI: 10.1137/S0097539795293172. URL: http://dx.doi.org/10.1137/S0097539795293172 (cit. on p. 14).

[SL07] M. Singh and L. C. Lau. « Approximating Minimum Bounded Degree Spanning Trees to within One of Optimal ». In: *Proceedings of the 39th annual ACM symposium on Theory of computing*. ACM. 2007, pp. 661–670 (cit. on p. 90).

[Ste94] Jacques Stern. « A New Identification Scheme Based on Syndrome Decoding ». In: *Advances in Cryptology – CRYPTO'93*. Ed. by Douglas R. Stinson. Vol. 773. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1994, pp. 13–21 (cit. on pp. 82, 85, 87).

[SW86] Dennis Stanton and Dennis White. *Constructive combinatorics*. Springer-Verlag New York, Inc., 1986 (cit. on pp. 46, 53).

[Tur36] A. Turing. « On Computable Numbers with an Application to the] Entscheidungsproblem[». In: *Proceeding of the London Mathematical Society* (1936), pp. 230–265 (cit. on p. 15).

[TW06] M. Thomlinson and Ch. Walker. *Distribution of encrypted software update to reduce attack winodow*. Tech. rep. US Patent 7876902 B2. Department of Computer Science, Michigan State University, Aug. 2006. URL: https://www.lens.org/lens/patent/US_7876902_B2 (cit. on p. 157).

[UMS11] S. K. Udgata, A. Mubeen, and S. L. Sabat. « Wireless Sensor Network Security Model Using Zero Knowledge Protocol ». In: *ICC*. IEEE, 2011, pp. 1–5 (cit. on p. 89).

[VCT14] Rafael Veras, Christopher Collins, and Julie Thorpe. « On Semantic Patterns of Passwords and their Security Impact ». In: *ISOC Network and Distributed System Security Symposium – NDSS 2014*. San Diego, California, USA: The Internet Society, Feb. 2014 (cit. on pp. 125, 128).

[W73] Lampson Butler W. « A note on the confinement problem ». In: *Commun. ACM* 16.10 (Oct. 1973), pp. 613–615. ISSN: 0001-0782. DOI: 10.1145/362375.362389. URL: http://doi.acm.org/10.1145/362375.362389 (cit. on p. 34).

[WAdG09] Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. « Password Cracking Using Probabilistic Context-Free Grammars ». In: *2009 IEEE Symposium on Security and Privacy*. Oakland, California, USA: IEEE Computer Society Press, May 2009, pp. 391–405 (cit. on pp. 125, 128).

[Wat05] Brent Waters. « Efficient identity-based encryption without random oracles ». In: *EUROCRYPT '05*. Vol. 3494. LNCS. Springer Berlin Heidelberg, 2005, pp. 114–127 (cit. on pp. 139, 150).

[Wat09] Brent Waters. « Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions ». In: *CRYPTO '09*. Vol. 5677. LNCS. Springer Berlin Heidelberg, 2009, pp. 619–636 (cit. on pp. 140, 145).

[WQZ+16] Qianhong Wu, Bo Qin, Lei Zhang, Josep Domingo-Ferrer, Oriol Farràs, and Jesús A.Manjón. « Contributory Broadcast Encryption with Efficient Encryption and Short Ciphertexts ». In: *IEEE Transactions on Computers* 65.2 (2016), pp. 466–479 (cit. on p. 139).

[Yao86] Andrew Chi-Chih Yao. « How to Generate and Exchange Secrets (Extended Abstract) ». In: *27th Annual Symposium on Foundations of Computer Science*. Toronto, Ontario, Canada: IEEE Computer Society Press, Oct. 1986, pp. 162–167 (cit. on pp. 9, 55).

[YK05] M. Yung and J. Katz. *Digital Signatures (Advances in Information Security)*. Springer-Verlag, 2005 (cit. on p. 24).

[YKJ+15] Ji Won Yoon, Hyoungshick Kim, Hyun-Ju Jo, Hyelim Lee, and Kwangsu Lee. « Visual Honey Encryption: Application to Steganography ». In: *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2015, Portland, OR, USA, June 17 - 19, 2015*. Ed. by Adnan M. Alattar, Jessica J. Fridrich, Ned M. Smith, and Pedro Comesaña Alfaro. ACM, 2015, pp. 65–74. ISBN: 978-1-4503-3587-4. DOI: 10.1145/2756601.2756606 (cit. on p. 122).

[You67] Daniel H. Younger. « Recognition and Parsing of Context-Free Languages in Time n^3 ». In: *Information and Control* 10.2 (1967), pp. 189–208. DOI: 10.1016/S0019-9958(67)80007-X (cit. on p. 129).

[YY05a] Adam L. Young and Moti Yung. « A Space Efficient Backdoor in RSA and Its Applications ». In: *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*. 2005, pp. 128–143. DOI: 10.1007/11693383_9. URL: http://dx.doi.org/10.1007/11693383_9 (cit. on p. 43).

[YY05b] Adam L. Young and Moti Yung. « Malicious Cryptography: Kleptographic Aspects ». In: *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*. 2005, pp. 7–18. DOI: 10.1007/978-3-540-30574-3_2. URL: http://dx.doi.org/10.1007/978-3-540-30574-3_2 (cit. on p. 43).

[YY96] Adam L. Young and Moti Yung. « The Dark Side of "Black-Box" Cryptography, or: Should We Trust Capstone? » In: *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*. Vol. 1109. Lecture Notes in Computer Science. Springer, 1996, pp. 89–103. ISBN: 3-540-61512-1. DOI: 10.1007/3-540-68697-5_8. URL: http://dx.doi.org/10.1007/3-540-68697-5_8 (cit. on p. 43).

[YY97] Adam L. Young and Moti Yung. « Kleptography: Using Cryptography Against Cryptography ». In: *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*. 1997, pp. 62–74. DOI: 10.1007/3-540-69053-0_6. URL: http://dx.doi.org/10.1007/3-540-69053-0_6 (cit. on p. 43).

[ZO14] Zeyuan Allen Zhu and Lorenzo Orecchia. « Using Optimization to Break the Epsilon Barrier: A Faster and Simpler Width-Independent Algorithm for Solving Positive Linear

Programs in Parallel ». In: *CoRR* abs/1407.1925 (2014). URL: http://arxiv.org/abs/1407.1925 (cit. on p. 84).

[ZQWZ10] L. Zhang, B. Qin, Q. Wu, and F. Zhang. « Efficient Many-to-One Authentication with Certificateless Aggregate Signatures ». In: *Computer Networks* 54.14 (2010), pp. 2482–2491 (cit. on p. 89).

[ZWD+14] Lei Zhang, Qianhong Wu, Josep Domingo-Ferrer, Bo Qin, and Peng Zeng. « Signatures in hierarchical certificateless cryptography: Efficient constructions and provable security ». In: *Information Sciences* 272 (2014), pp. 223–237 (cit. on p. 138).

[ZYT14] Mingwu Zhang, Bo Yang, and Tsuyoshi Takagi. « Anonymous spatial encryption under affine space delegation functionality with full security ». In: *Information Sciences* 277 (2014), pp. 715–730 (cit. on p. 140).

Résumé

Cette thèse présente des résultats appartenant aux trois thèmes fondamentaux de la cryptographie à clé publique: *l'intégrité*, *l'authentification* et *la confidentialité*. Au sein de chaque thème nous concevons des nouvelles primitives et améliorons des primitives existantes.

Le premier chapitre, dédié à l'intégrité, introduit une preuve non-interactive de génération appropriée de clés publiques RSA et un protocole de co-signature dans lequel tout irrespect de l'équité laisse automatiquement la partie lésée en possession d'une preuve de culpabilité incriminant la partie tricheuse.

Le second chapitre, ayant pour sujet l'authentification, montre comme une mesure de temps permet de raccourcir les engagements dans des preuves à divulgation nulle et comment des biais, introduits à dessin dans le défi, permettent d'accroitre l'efficacité de protocoles. Ce chapitre généralise également le protocole de Fiat-Shamir à plusieurs prouveurs et décrit une fraude très sophistiquée de cartes-à-puce illustrant les dangers de protocoles d'authentification mal-conçus.

Au troisième chapitre nous nous intéressons à la confidentialité. Nous y proposons un cryptosystème à clé publique où les hypothèses de complexité traditionnelles sont remplacées par un raffinement du concept de CAPTCHA et nous explorons l'application du chiffrement-pot-de-miel au langage naturel.

Nos dernières contributions concernent le chiffrement basé sur l'identité (IBE). Nous montrerons comment ajouter des fonctions d'émission à l'IBE hiérarchique et comment l'IBE permet de réduire la fenêtre temporelle de risque lors de la diffusion de mises à jour logicielles.

Mots Clés

cryptographie, intégrité, authentification, confidentiailité, chiffrement signatures numériques, équité, preuves à divulgation nulle.

Abstract

This thesis presents new results in three fundamental areas of public-key cryptography: *integrity*, *authentication* and *confidentiality*. In each case we design new primitives or improve the features of existing ones.

The first chapter, dealing with integrity, introduces a non-interactive proof for proper RSA public key generation and a contract co-signature protocol in which a breach in fairness provides the victim with transferable evidence against the cheater.

The second chapter, focusing on authentication, shows how to use time measurements to shorten zero-knowledge commitments and how to exploit bias in zero-knowledge challenges to gain efficiency. This chapter also generalizes Fiat-Shamir into a one-to-many protocol and describes a very sophisticated smart card fraud illustrating what can happen when authentication protocols are wrongly designed.

The third chapter is devoted to confidentiality. We propose public-key cryptosystems where traditional hardness assumptions are replaced by refinements of the CAPTCHA concept and explore the adaptation of honey encryption to natural language messages. Our final contributions focus on identity-based encryption (IBE) showing how to add broadcast features to hierarchical IBE and how to use IBE to reduce vulnerability exposure time of during software patch broadcast.

Keywords

cryptography, integrity, authentication, confidentiality, encryption, digital signatures, fairness, zero-knowledge proofs.