Lecture Notes in Computer Science 1561

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Ivan Damgård (Ed.)

Lectures on Data Security

Modern Cryptology in Theory and Practice

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editor

Ivan Bjerre Damgård
BRICS, University of Aarhus
Ny Munkegade, Building 540
DK-8000 Aarhus C, Denmark
E-mail: ivan@daimi.aau.dk

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Lectures on data security : modern cryptology in theory and practice / Ivan Damgård (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 1999
(Lecture notes in computer science ; 1561)
ISBN 3-540-65757-6

Cover illustration taken from the contribution by Stefan Wolf, pages 217 ff

CR Subject Classification (1998): E.3, G.2.1, D.4.6, K.6.5, F.2.1-2, C.2, J.1

ISSN 0302-9743
ISBN 3-540-65757-6 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10702913 06/3142 — 5 4 3 2 1 0 Printed on acid-free paper

Preface

In July 1998, a summer school in cryptology and data security was organized at the computer science department of Aarhus University, Denmark. This took place as a part of a series of summer schools organized by the European Educational Forum, an organization consisting of the research centers TUCS (Finland), IPA (Holland) and BRICS (Denmark, Aarhus). The local organizing committee consisted of Jan Camenisch, Janne Christensen, Ivan Damgaård (chair), Karen Møller, and Louis Salvail. The summer school was supported by the European Union.

Modern cryptology is an extremely fast growing field and is of fundamental importance in very diverse areas, from theoretical complexity theory to practical electronic commerce on the Internet. We therefore set out to organize a school that would enable young researchers and students to obtain an overview of some main areas, covering both theoretical and practical topics. It is fair to say that the school was a success, both in terms of attendance (136 participants from over 20 countries) and in terms of contents. It is a pleasure to thank all of the speakers for their cooperation and the high quality of their presentations.

A total of 13 speakers gave talks: Mihir Bellare, University of California, San Diego; Gilles Brassard, University of Montreal; David Chaum, DigiCash; Ronald Cramer, ETH Zürich; Ivan Damgård, BRICS; Burt Kaliski, RSA Inc.; Lars Knudsen, Bergen University; Peter Landrock, Cryptomathic; Kevin McCurley, IBM Research, Almaden; Torben Pedersen, Cryptomathic; Bart Preneel, Leuven University; Louis Salvail, BRICS; Stefan Wolf, ETH Zürich.

It was natural to take the opportunity kindly offered by Springer-Verlag to publish a set of papers reflecting the contents of the school. Although not all speakers were able to contribute, due to lack of time and resources, this volume does cover all the main areas that were presented. The intention of all papers found here is to serve an educational purpose: elementary introductions are given to a number of subjects, some examples are given of the problems encountered, as well as solutions, open problems, and references for further reading. Thus, in general we have tried to give an up-to-date overview of the subjects we cover, with an emphasis on insight, rather than on full-detail technical presentations. Several results, however, are in fact presented with full proofs. The papers have not been refereed as for a journal.

I would like to thank all of the authors for their contributions and the hard work and time they have invested.

Ivan Damgård

Practice-Oriented Provable-Security

Mihir Bellare

Dept. of Computer Science & Engineering, University of California at San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA
mihir@cs.ucsd.edu
URL: www-cse.ucsd.edu/users/mihir

1 Introduction

This short article is intended to complement my talk. I would like to try to introduce you to a certain, relatively new sub-area of cryptography that we have been calling *practice-oriented provable-security*. It is about applying the ideas of "provably security" to the derivation of practical, secure protocols. I believe it is a fruitful blend of theory and practice that is able to enrich both sides and has by now had some impact on real world security.

A few years ago, provable security was largely known only to theoreticians. This has been changing. We are seeing a growing appreciation of provable security in practice, leading in some cases to the use of such schemes in preference to other ones. Indeed it seems standards bodies and implementors now view provable security as an attribute of a proposed scheme. This means that a wider audience needs an understanding of the basic ideas behind provable security.

This article is directed at practioners and theoreticians alike. For the first I hope it will help to understand what provable security is and isn't, why it is useful, how to evaluate the provable security of a scheme, and where to look for such schemes. For the second group, it can serve to acquaint them with how the ideas with which they are familiar are being applied.

I will begin by describing the basic idea behind provable security. (For many of you, this will be mostly recall, but some novel viewpoints or examples may enter.) Next, I will discuss the practice-oriented approach. I will discuss its main ideas, the problems it has addressed, and briefly survey known results. I hope to leave you feeling there is scope here both for interesting research and for application.

2 Protocols, Primitives, Proofs and Practice

The basic task in cryptography is to enable to parties to communicate "securely" over an insecure channel, namely in a way that guarantees privacy and authenticity of their transmissions. (There are many other tasks as well, but we will begin by thinking about this basic one.)

I. Damgård (Ed.): Lectures on Data Security, LNCS 1561, pp. 1–15, 1999.

2.1 Protocols and Primitives: The Problem

PROTOCOLS: THE END GOAL. To enable secure communication, one wants cryptographic *protocols* or *schemes*. For example, an encryption scheme enables users to communicate privately. Such a scheme is specified by a pair $(\mathcal{E}, \mathcal{D})$ of algorithms. The first, run by the sender, takes a *key* and the *plaintext* M to create a *ciphertext* C, which is transmitted to the receiver. The latter applies $\mathcal{D}$, which takes a key and the received ciphertext to recover the plaintext. (Roughly, the security property desired is that an adversary can't learn anything useful about the plaintext given the ciphertext, but we will get into this more later.) They key could be a shared one (this is the private key or symmetric setting) or the keys for encryption and decryption could be different (the public key or asymmetric setting). Designing an encryption scheme means designing the two algorithms $\mathcal{E}$ and $\mathcal{D}$.

Similarly, a message authentication scheme (or protocol) enables parties to tag their data so that the recipient is assured that the data originates with the person claiming to have sent it and has not been tampered with on the way.

The design of such protocols is the end goal for the cryptographer. However, it is not an easy one to reach. What makes it reachable at present is that we have very good *primitives* on which to *base* these protocols.

PRIMITIVES: THE TOOLS. Julius Caesar also wanted to design protocols. He had a much harder time than we do today, because he didn't have DES or the RSA function.

The latter are examples of what I will call *atomic primitives*. Certainly, they are cryptographic objects of some sort. What is it that distinguishes them from protocols? The distinction is that in their purest and rawest state, atomic primitives don't solve any cryptographic problem we actually care about. We must *use* them appropriately to construct protocols to solve the problems that matter. For example, DES based CBC encryption is a way of using DES to do symmetric encryption. By first hashing a message and then decrypting under RSA we have a possible way to do digital signatures based on the RSA function. (Whether these ways are good or bad ways of accomplishing the goal is another question, to be addressed later.) Thus, atomic primitives are simple building blocks that must be put together to yield protocols.

Good atomic primitives are rare, as are people who understand their workings. Certainly, an important effort in cryptography is to design new atomic primitives and cryptanalyze them and old ones. This, however, is not the part of cryptography I want to talk about. The reason is that the design (or discovery) of good atomic primitives is more an art than a science. On the other hand, I'd like to claim that the design of protocols can be made a science.

THE QUESTION. We will view a cryptographer as an engine for turning atomic primitives into protocols. That is, we focus on protocol design under the assumption that good atomic primitives exist. Some examples of the kinds of questions we are interested in are these. What is the best way to encrypt a large text file using DES, assuming DES is secure? What is the best way to design a signature

scheme using the RSA function, assuming the latter is one-way? How "secure" are known methods for these tasks? What do such questions even mean, and can we find a scientific framework in which to ask and answer them?

THE PROBLEM. The problem with protocol design is that a poorly designed protocol can be insecure *even though the underlying atomic primitive is good*. An example is ECB (Electronic Code-Book) mode encryption with a block cipher. It is not a good encryption scheme because partial information about the plaintext leaks. Yet this is no fault of the underlying atomic primitive (typically DES). Rather, the atomic primitive was mis-used.

Indeed, lots of protocols are broken. Yet the good atomic primitives, like DES and RSA, have never been convincingly broken. We would like to build on the strength of atomic primitives in such a way that protocols can "inherit" this strength, not loose it!

2.2 Provable Security: Reductions

The idea of provable security was introduced in the pioneering work of Goldwasser and Micali [26]. They developed it in the particular context of asymmetric encryption, but it soon spread to be applied to other tasks. (Of these, the most basic were pseudorandomness [16,40,25] and digital signatures [27]).

WHAT IS PROVABLE SECURITY? The paradigm is as follows. Take some goal, like achieving privacy via encryption. The first step is to make a formal adversarial model and *define* what it *means* for an encryption scheme to be secure. With this in hand, a particular scheme, based on some particular atomic primitive, can be analyzed from the point of view of meeting the definition. Eventually, one shows that the scheme "works" via a *reduction*. The reduction shows that the *only way* to defeat the protocol is to break the underlying atomic primitive. In other words, there is no need to directly cryptanalyze the protocol: if you were to find a weakness in it, you would have unearthed one in the underlying atomic primitive. So you might as well focus on the atomic primitive. And if we believe the latter is secure, we *know*, without further cryptanalysis of the protocol, that the protocol is secure.

An important sub-part of the last step is that in order to enable a reduction one must also have a formal notion of what is meant by the security of the underlying atomic primitive: what attacks, exactly, does it withstand? For example, we might assume RSA is a one-way function.

Here is another way of looking at what reductions do. When I give you a reduction from the one-wayness of RSA to the security of my protocol, I am giving you a transformation with the following property. Suppose you claim to be able to break my protocol. Let P be the program that does this. My transformation takes P and puts a simple "wrapper" around it, resulting in a protocol P'. This protocol P' provably breaks RSA. Conclusion? As long as we believe you can't break RSA, there could be no such program P. In other words, my protocol is secure.

Those familiar with the theory of NP-completeness will recognize that the basic idea of reductions is the same. When we provide a reduction from SAT to some problem we are saying our problem is hard unless SAT is easy; when we provide a reduction from RSA to our protocol, we are saying the latter is secure unless RSA is easy.

Here, I think, is a beautiful and powerful idea. Some of us by now are so used to it that we can forget how innovative it was. And for those not used to it, it can be hard to understand (or, perhaps, believe) at first hearing, perhaps because it delivers so much. Protocols designed this way truly have superior security guarantees.

Nomenclature. In some ways the term "provable security" is misleading. As the above indicates, what is probably the central step is providing a model and definition, which does not involve proving anything. And one does not "prove a scheme secure:" one provides a reduction of the security of the scheme to the security of some underlying atomic primitive. For that reason, I sometimes use the term "reductionist security" to refer to this genre of work.

The complexity-theoretic approach. The precise formalization of provable security can take many forms. The theoretical literature has chosen, for the most part, to develop it in a complexity theoretic framework where one talks about "polynomial time" adversaries and transformations, and "negligible success probabilities." This approach was convenient for a field striving to develop a technical idea of great depth. Complexity-based cryptography has been remarkably successful, coming up with definitions for many central cryptographic primitives, and constructions based on "minimal assumptions." For a brief introduction to this body of work, refer to the recent survey by Goldreich [24].

In practice? The potential for the idea of provable security to impact practice is large. Yet its actual impact had been disappointingly small, in the sense that these ideas were reflected almost not at all in protocols used in practice. Here are some possible reasons.

In practice, block ciphers are the most popular atomic primitive, especially for private key cryptography. Yet the provable security line of work (prior to the development of the practice-oriented variant) omitted any treatment of schemes based on block ciphers: only number-theoretic atomic primitives were deemed adequate as a basis for protocol design. In particular some of the world's most used protocols, such as CBC MAC [1] or encryption [32,2], seemed to be viewed as outside the domain of provable security.[1]

The main generic disadvantage of the schemes delivered by the traditional provable security approach is that they are inefficient.[2] This is due in part to the complexity of the constructions. But it is also due in part to a reliance on inefficient atomic primitives. For example, a MAC would be constructed out of

[1] Luby and Rackoff [31] studied the Feistel structure behind DES, but what I am talking about is to look at protocols that use DES and ask about their security.

[2] Typically the gap relative to what is desirable in practice is enormous. In some cases it is small, but still seems enough to preclude usage.

a one-way function like RSA rather than out of a block cipher. This takes us back to the above.

Finally, some aspects of the complexity-theoretic approach unfortunately distanced provable security from practice. For example, practioners need numbers: how many cycles of adversary computation can the scheme withstand, how many bits is the security parameter? These are only loosely captured by "polynomials" or "negligible probabilities." To make provable security useful, reductions and security analyses must be concrete. Theoreticians will say, correctly, that this information can be obtained by looking at their proofs. But this view obscures the importance of working on improving the security of reductions.[3]

Practice-oriented provable security attempts to remedy this by appropriate paradigm shifts.

3 Practice-Oriented Provable Security

Practice-oriented provable security as I discuss it was introduced in a set of papers authored by myself and Phil Rogaway [8,7,6]. We preserve and focus on the two central ideas of the provable security approach: the introduction of *notions*, or *definitions* that enable us to think about protocols and atomic primitives in a systematic way, and the idea of doing reductions. But we modify the viewpoints, models, and problems treated. Here are some elements of the approach and work to date.

3.1 Using Block Ciphers

Block ciphers like the DES are the most ubiquitous tool in practical cryptographic protocol design. However, as indicated above, traditionally nothing was proved about protocols that use them. An important element of our line of work is to integrate block ciphers into the fabric of provable security. On the one hand we analyze existing schemes that use block ciphers to assess how well they meet strong, formal notions of security; on the other hand we design new schemes based on block ciphers and show they meet such notions. In the first category are our analyses of the CBC MAC [7] and analyses of various modes of operation of a block cipher [5]. In the second category are constructions like the XOR MAC [6] or the cascade [4].

Key to these results (and perhaps more important than any individual result) is that we treat block ciphers systematically by formally modeling them in some way. Specifically, the suggestion of [7], followed in the other works, was to model a block cipher as a *finite pseudorandom function* (FPRF) family. (The fundamental notion of a pseudorandom function family is due to Goldreich, Goldwasser and Micali [25]. The finite variant was introduced in [7].) Roughly, we are

[3] This is not to say concrete security has always been ignored. One person who from the beginning has systematically addressed concrete security in his works is Claus Schnorr. See any of his papers involving cryptographic reductions.

assuming that as long as you don't know the underlying key, the input-output behavior of a block cipher closely resembles that of a random function.

Thus, the theorems in the mentioned papers say that a scheme (eg. CBC MAC) is secure unless one can detect some deviation from random behavior in the underlying block cipher. Underlying this claim is a reduction, as usual in the provable security approach, showing how to break the cipher given any way to break the scheme based on it.

The idea of treating block ciphers as pseudorandom functions provides a fresh way of looking at block ciphers from both the design and usage perspective. On the one hand, this view can form the basis for analyses of many other block cipher based schemes. On the other hand, we suggest it be a design criterion for future block ciphers (a view that new efforts such as AES do seem to support) and that existing ciphers should be cryptanalyzed to see how well they meet this goal.

3.2 Concrete Security

Practice oriented provable security attempts to explicitly capture the inherently *quantitative* nature of security, via a *concrete* or *exact* treatment of security. Rather than prove asymptotic results about the infeasability of breaking a protocol in polynomial time, we present and prove "exact" or "concrete" reductions. Our results have the form: "If DES withstands an attack in which the adversary gets to see 2^{36} plaintext-ciphertext pairs, then our protocol is secure against an adversary who can run for t steps, for the following value of t." This enables a protocol designer to know exactly how much security he/she gets. And it brings a new dimension to protocols: rather than just being secure or non-secure, one can be "more" secure than another.

For example, the theorem of [7] characterizing the security of the CBC MAC says that an adversary who runs for time t and sees q correctly MACed messages has chance at most $\epsilon + (3q^2n^2 + 1)/2^l$ of correctly forging the MAC of a new message, where l is the block length of the underlying cipher, n is the number of blocks in any message to which the MAC applies, and ϵ captures the security of the cipher, specifically being the chance of detecting a deviation of the cipher from random behavior in time $t + O(nql)$ given nq input-output examples of the cipher under the same key. (This ϵ is of course a function of the key length of the underlying cipher, but the latter does not need to appear explicitly.) Thus, a user sees exactly how the chance of forgery increases with the number of messages MACed.

Another aspect of the concrete security treatment is to try to preserve as much as possible of the strength of the underlying atomic primitive in transforming it to the protocol. This means we aim for reductions as *strong* as possible. This is important because reduction strength translates directly to protocol efficiency in practice. A weak reduction means that to get the same level of security in our protocol we must use larger keys for the underlying atomic primitive, and this means slower protocols. If the reduction is strong, shorter keys will suffice

and the protocol is more efficient. Reduction quality plays a significant role in [7,6,10,12,4,5] all of which achieve tight or close to tight reductions.

We found that *improving* the concrete security was a rich and rewarding line of work, and thinking about it greatly increases understanding of the problem.

In [5] we also concern ourselves with how different formalizations of a notion (in this case, secure encryption) are affected when concrete security is an issue.

3.3 Security Versus Attacks

Practitioners typically think only about concrete attacks; theoreticians ignore them, since they prove the security. Under the practice oriented provable security approach, attacks and security emerge as opposite sides of the same coin, and complement each other. Attacks measure the degree of insecurity; our quantitative bounds measure the degree of security. When the two meet, we have completely characterized the security of the protocol.

For example, the security of the CBC MAC shown in [7] is the flip-side of attacks like those of Preneel and Van Oorschot [37]. (The latter say that the CBC MAC can be broken once $2^{l/2}$ messages have been MACed, where l is the block length of the underlying cipher. We say, roughly, that it *can't* be broken when *fewer* than this many messages are MACed.) Thus the results of [7,37] complement each other very well. Yet, the literature on these subjects does not reflect this duality appropriately.

We found that even when proofs are provided, much is to be gained by finding the best possible attacks. We find *new kinds* of attacks, which break the system as measured by our more stringent notions of security: an encryption scheme is broken of you can tell whether the message encrypted was 0 or 1, not just if you find the key. This is actually important in practice. Meanwhile, these attacks provide, effectively, the lower bounds to our concrete security analyses, telling us whether the proven security is optimal or not. Publications in which we assess the optimality of our reductions via attacks include [6,4,5].

3.4 The Random Oracle Model

Sometimes, using pseudorandom function families or one-way functions alone, we are not able to find schemes efficient enough for practice. This is true for example in the case of public key encryption or signatures. In such cases, we turn to the random oracle paradigm.

The random oracle paradigm was introduced in [9] as a bridge between theory and practice. The idea is a simple one: namely, provide all parties —good and bad alike— with access to a (public) function h; prove correct a protocol assuming h is truly random, ie. a random oracle; later, in practice, set h to some specific function derived in some way from a standard cryptographic hash function like SHA-1 [33] or RIPEMD-160 [21].

We used the random oracle paradigm most importantly to design OAEP [10] and PSS [12]. These are schemes for (public key) encryption and signature

(respectively), the most popular versions of which use RSA as the underlying primitive. (Both OAEP and PSS are, more accurately, padding or formatting mechanisms which are applied to a message before the appropriate RSA operation is applied.) They are as efficient as previously used or standardized schemes, but, unlike them, provably achieve strong notions of security in the random oracle model, assuming RSA is a one-way function.

RSA Corporation publishes a standard for RSA based encryption called PKCS#1. (It is a widely used standard, implemented in Netscape and other browsers, and used in SSL.) Much publicity was given recently to a chosen-ciphertext attack on PKCS#1 that was discovered by Bleichenbacher [15]. RSA Corporation has now revised the protocol, adopting OAEP in PKCS#1 v2.0 [38]. The rationale for that move is that our protocol had been *proven* to resist chosen-ciphertext attacks (indeed Bleichenbacher's attacks do not work on OAEP, even though at the time of the design of OAEP we had not thought of these specific attacks), and furthermore OAEP is just as practical as the original PKCS#1 protocol.

OAEP is also included in SET, the electronic payment protocol of MasterCard and Visa, where it is used to encrypt credit card numbers. Both OAEP and PSS are being proposed for the IEEE P1363 standard.

What's the point of the random oracle paradigm, and what does it buy you? It buys efficiency, plus, we claim, security guarantees which, although not at the same level as those of the standard provable security approach, are arguably superior to those provided by totally ad hoc protocol design. The last point merits some more discussion.

The random oracle paradigm should be used with care and understanding. It is important to neither over-estimate nor under-estimate what this paradigm buys you in terms of security guarantees. First, one must be clear that this is not standard provable security. The function h that we actually use in the final scheme is not random. Thus the question is: what has it bought us to have done the proof in the first place?

The overly skeptical might say the answer is "nothing." This is not quite true. Here is one way to see what it buys. In practice, attacks on schemes involving a SHA-1 derived h and number theory will often *themselves treat h as random.* We call such attacks *generic attacks.* In other words, cryptanalysis of these "mixed" schemes is usually done by assuming h is random. But then the proofs apply, and indeed show that such generic attacks will fail unless the underlying number-theoretic problems are easy. In other words, the analysis at least provably excludes a certain common class of attacks, namely generic ones.

It is important to choose carefully the instantiating function h. The intuition stated in [9] is that the resulting scheme is secure as long as the scheme and the hash function are sufficiently "independent," meaning the scheme does not itself refer to the hash function in some way. This is a fuzzy guideline which we hope to understand better with time.

An important step in our understanding of the random oracle model was taken by Canetti, Goldreich and Halevi [19]. They indicate that there exist

schemes secure in the random oracle model but insecure under any instantiation in which we substitute a function from a small family of efficiently computable functions. Their examples however are somewhat contrived, and this kind of situation does not arise with any of the "real" constructions in the literature.

In comparison with totally ad hoc design, a proof in the random oracle model has the benefit of viewing the scheme with regard to its meeting a strong and formal notion of security, even if this is assuming some underlying primitive is very strong. This is better than not formally modeling the security of the scheme in any way. This explains why the random oracle model is viewed in [9] as a "bridge between theory and practice."

Since we introduced this model, it has been used in other places, for example in the design and analysis of signature schemes [35,36,34], hash functions [13] and threshold encryption schemes [23].

3.5 New Notions: Session Key Distribution

"Entity authentication" is the process by which a party gains confidence in the identity of a communication partner. It is usually coupled with the distribution of a "session key." These are arguably the most basic problems for secure distributed computation— without a correct solution there can be no meaningful access control or accountability; there cannot even be reliable distribution of work across network resources. Despite a long history and a large literature, this problem rested on no meaningful formal foundation. This is more than an academic complaint: it is an area in which an informal approach has often lead to work which has subsequently been found to be wrong, and in some cases the flaws have taken years to discover.

In [8] we address the two party setting of the problem. It achieves provable security by providing a model, definitions, protocols, and proofs of correctness for these protocols under standard assumptions.

The three party case of this problem may be the most well-known. It was first addressed by Needham and Schroeder in 1978. Its most popular incarnation is the *Kerberos* system. However this system, and existing solutions, suffer from the same problems discussed above. In [11] we provide provably secure protocols for the three party session key distribution problem.

All our protocols are efficient and practical, viable alternatives to current systems. Some have been implemented. Our models have been used to study related key distribution problems, for example in [39].

4 What Provable Security Is and Isn't

Now that provable security is moving into practice, there are many people who although not trained as theoreticians, or even deeply interested in the details of research, need to take decisions involving claims about provable security. The kinds of things they need to know are: exactly what provable security provides and doesn't provide; how to compare different provably secure schemes; how to

validate a claim of provable security. So I would like to discuss some of these points here.

4.1 On Limitations

The above has explained what provable security provides, but it is also important to understand its limitations. The first of these is in the model considered. One must ask what kinds of attacks the model encompasses. In particular, there are classes of attacks that do not fall into the normal fabric of provable security; these include timing attacks [29], differential fault analysis [18,14], and differential power analysis [30]. (That is, models used in provable security do not encompass these attacks. One should note that this does not mean specific proven secure schemes fall to these attacks. It just means we do not claim to have *proven* that they do not fall to these attacks. In fact if you look at specific schemes, some fall to these attacks, others don't.) But it is worth investigating an extension of provable security that does include these attacks, a line of research suggested by Dan Boneh.

Even when using a proven secure scheme, security can compromised in a number of ways. Sometimes, requirements may have been overlooked: we proved security, but for the wrong problem or in the wrong model. Or, the protocol may be used incorrectly. For example, a setting such as key exchange might require a public key encryption scheme that is secure against chosen-ciphertext attack. It does little good to use a proven secure scheme that is only proven secure against chosen-plaintext attack. This is a question of understanding what requirements a higher level protocol imposes on the lower level primitive.

Or software may be buggy. If you implement the scheme incorrectly, obviously all bets are off. Similarly the environment may be improperly administered leading to loss of passwords or keys. There may be insider attacks. And so on.

4.2 On Assumptions

Proven security does not mean that attacks (of the kind modeled) are unconditionally guaranteed to fail. Remember that a scheme is proven secure given some *assumption*. For example, we may have an encryption scheme proven to resist chosen-plaintext attacks as long as the problem of factoring a number product of two primes is computationally infeasible. Or, as in examples above, that a message authentication scheme is secure as long as the underlying cipher behaves like a pseudorandom function family.

If these assumptions fail, of course the proof becomes worthless. (One should note that failure of an assumption does not necessarily lead to attacks on the scheme. It just means that the proof of security is no longer useful.) This means that a proof of security is worth more when the assumption is weaker, ie. less likely to fail. An important parameter of a proof of security is thus the underlying assumption: the weaker the better. In particular this becomes something to consider in comparing schemes. If you have a choice between two schemes, you

will of course take into account many things, such as performance, ease of implementation, exportability and so on. But on the security front, if both schemes are proven secure, the one making weaker assumptions is preferable.

Comparing provable guarantees has both a qualitative and quantitative aspect. Even when two schemes are based on the same assumptions, one may have better concrete security. (We discussed the concrete security approach in Section 3.2.) This means that there is less loss of security in the translation from the problem of the assumption to the scheme. An example is the signature schemes FDH and PSS [12]– both are proven secure in a random oracle model assuming RSA is one-way, but the reduction for PSS is tight and that for FDH is not, so the quantitative guarantee of PSS is better.

How does one compare assumptions to see which is weaker? Unfortunately it is not always possible. Indeed, in the bulk of cases, we do not know how to compare the assumptions underlying various proofs of security. But it is still important to know about this and know when they are comparable and when not.

To illustrate these issues let us look at public key encryption secure against chosen-ciphertext attack. We discussed (RSA based) OAEP [10] above: it is proven secure in the random oracle model assuming the RSA function is one-way.

Dolev, Dwork and Naor [22] had designed a scheme that resists chosen-ciphertext attack many years prior to this. Lets call this the DDN scheme. The security of the DDN scheme can be proven assuming RSA is a one-way function. Notice that this assumption is weaker than the one underlying OAEP, since OAEP assumes in addition that we have a hash function that behaves like a random oracle. As a consequence we can say that the provable security guarantee provided in the DDN scheme is superior to that of OAEP. In this case, a security comparison was possible.

More recently, Cramer and Shoup [20] introduced a new proven-secure encryption scheme which we call the CS scheme. Unlike the schemes we have been discussing up to now, it is not RSA based: it assumes the hardness of a certain version of the Diffie-Hellman problem. How does the security of the CS scheme compare to that of OAEP? That is more difficult to assess. The CS scheme does not use the random-oracle paradigm, which is a plus. But it assumes the hardness of the so-called Decisional Diffie Hellman problem. (See [17] for a nice discussion of this problem.) This is a strong assumption, and relatively new and un-studied one compared to the assumption that RSA is one-way. (It would be much more surprising if the RSA assumption failed than if the Decisional Diffie-Hellman assumption failed.) In particular, we do not know how this assumption compares to the assumptions underlying OAEP. So, while the fact that the CS scheme avoids random oracles is a point in its favor, it is not really possible to say that one of these schemes has better security guarantees than the other in practice, because the assumptions are incomparable.

If one had to choose a scheme in practice one would of course also consider cost. OAEP has the same cost as heuristic RSA schemes. The DDN scheme is

many orders of magnitude more expensive than any practical scheme, since it involves multiple signatures and zero-knowledge proofs, and thus is likely to be ruled out. The CS scheme is much cheaper than the DDN scheme, but still more expensive than OAEP. (Encryption in OAEP is only a few multiplications if a small RSA exponent is used; while in the CS scheme it is a few exponentiations. Decryption in the CS scheme is about five times as costly as in OAEP.) In some applications, this kind of increase may be tolerable; in others not. There is no unique answer.

4.3 Proofs and Definitions

Faced with a choice of protocols claiming to be provably secure, we discussed above some issues involved in comparing them. Another should be mentioned: verification of the claims. A scheme isn't provably secure just because it is claimed to be so. One should check that proper formal definitions of security have been provided so as to know *what* is being proved. One should be able to at least cursorily verify the claims. How? Remember that a reduction consists of an algorithm that is an attacker for the problem we are assuming hard, using as a subroutine an attacker for the scheme. Look at the least for a description of such an algorithm.

5 Going On

The above has discussed provable security and its practice oriented variant in a general way. Next I would like to illustrate the ideas by looking in more depth at a central problem: encryption. The goal is to motivate the need for strong and formal notions of security and then show how to to adapt the seminal notions of [26] (given in the asymmetric setting) to the symmetric setting. With concrete security definitions in hand, we will turn to analyzing popular encryption modes like CBC or CTR and gauge their merits. We want to answer questions like: are the secure? Which is "better"?

I did this in my talk, for the most part following [5], and refer the reader there for this materiel. Some day, I hope to extend this article by the inclusion of this and other materiel.

Acknowledgments

This presentation is based on ideas developed jointly with Phillip Rogaway.

This work is supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering.

References[4]

1. ANSI X9.9, "American National Standard for Financial Institution Message Authentication (Wholesale)," American Bankers Association, 1981. Revised 1986. 4
2. ANSI X3.106, "American National Standard for Information Systems – Data Encryption Algorithm – Modes of Operation," American National Standards Institute, 1983. 4
3. M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication," *Advances in Cryptology – Crypto* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
4. M. Bellare, R. Canetti and H. Krawczyk, "Psuedorandom functions revisited: The cascade construction and its concrete security," *Proceedings of the* 37th *Symposium on Foundations of Computer Science*, IEEE, 1996. 5, 7
5. M. Bellare, A. Desai, E. Jokipii and P. Rogaway, "A concrete security treatment of symmetric encryption," *Proceedings of the* 38th *Symposium on Foundations of Computer Science*, IEEE, 1997. 5, 7, 12
6. M. Bellare, R. Guérin and P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions," *Advances in Cryptology – Crypto* 95 *Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995. 5, 7
7. M. Bellare, J. Kilian and P. Rogaway, "The security of cipher block chaining," *Advances in Cryptology – Crypto* 94 *Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994. 5, 6, 7
8. M. Bellare and P. Rogaway, "Entity authentication and key distribution," *Advances in Cryptology – Crypto* 93 *Proceedings*, Lecture Notes in Computer Science Vol. 773, D. Stinson ed., Springer-Verlag, 1993. 5, 9
9. M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," *Proceedings of the First Annual Conference on Computer and Communications Security*, ACM, 1993. 7, 8, 9
10. M. Bellare and P. Rogaway, "Optimal asymmetric encryption – How to encrypt with RSA," *Advances in Cryptology – Eurocrypt* 95 *Proceedings*, Lecture Notes in Computer Science Vol. 921, L. Guillou and J. Quisquater ed., Springer-Verlag, 1995. 7, 11
11. M. Bellare and P. Rogaway, "Provably secure session key distribution– the three party case," *Proceedings of the* 27th *Annual Symposium on the Theory of Computing*, ACM, 1995. 9
12. M. Bellare and P. Rogaway, "The exact security of digital signatures: How to sign with RSA and Rabin," *Advances in Cryptology – Eurocrypt* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1070, U. Maurer ed., Springer-Verlag, 1996. 7, 11
13. M. Bellare and D. Micciancio, "A new paradigm for collision-free hashing: Incrementality at reduced cost," *Advances in Cryptology – Eurocrypt* 97 *Proceedings*, Lecture Notes in Computer Science Vol. 1233, W. Fumy ed., Springer-Verlag, 1997. 9
14. E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," *Advances in Cryptology – Crypto* 97 *Proceedings*, Lecture Notes in Computer Science Vol. 1294, B. Kaliski ed., Springer-Verlag, 1997. 10

[4] The best place to obtain any of my papers listed below is via `http://www-cse.ucsd.edu/users/mihir`. Here you will find the full, and most recent, versions.

15. D. Bleichenbacher, A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS #1, *Advances in Cryptology – Crypto* 98 *Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998. 8
16. M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits," *SIAM Journal on Computing*, Vol. 13, No. 4, November 1984, pp. 850–864. 3
17. D. Boneh, "The decision Diffie-Hellman problem," Invited paper for the Third Algorithmic Number Theory Symposium (ANTS), Lecture Notes in Computer Science Vol. 1423, Springer-Verlag, 1998. Available at `http://theory.stanford.edu/~dabo/abstracts/DDH.html`. 11
18. D. Boneh, R. DeMillo and R. Lipton, "On the importance of checking cryptographic protocols for faults," *Advances in Cryptology – Eurocrypt* 97 *Proceedings*, Lecture Notes in Computer Science Vol. 1233, W. Fumy ed., Springer-Verlag, 1997. 10
19. R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," *Proceedings of the* 30th *Annual Symposium on the Theory of Computing*, ACM, 1998. 8
20. R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," *Advances in Cryptology – Crypto* 98 *Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998. 11
21. H. Dobbertin, A. Bosselaers and B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD," *Fast Software Encryption*, Lecture Notes in Computer Science 1039, D. Gollmann, ed., Springer-Verlag, 1996. 7
22. D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography," *Proceedings of the* 23rd *Annual Symposium on the Theory of Computing*, ACM, 1991. 11
23. R. Gennaro and V. Shoup, "Securing threshold cryptosystems against chosen-ciphertext attack," *Advances in Cryptology – Eurocrypt* 98 *Proceedings*, Lecture Notes in Computer Science Vol. 1403, K. Nyberg ed., Springer-Verlag, 1998. 9
24. O. Goldreich, "On the foundations of modern cryptography," *Advances in Cryptology – Crypto* 97 *Proceedings*, Lecture Notes in Computer Science Vol. 1294, B. Kaliski ed., Springer-Verlag, 1997. 4
25. O. Goldreich, S. Goldwasser and S. Micali, "How to construct random functions," *Journal of the ACM*, Vol. 33, No. 4, October 1986, pp. 792–807. 3, 5
26. S. Goldwasser and S. Micali, "Probabilistic encryption," *J. of Computer and System Sciences*, Vol. 28, April 1984, pp. 270–299. 3, 12
27. S. Goldwasser, S. Micali and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal of Computing*, Vol. 17, No. 2, April 1988, pp. 281–308. 3
28. ISO 8372, "Information processing – Modes of operation for a 64-bit block cipher algorithm," International Organization for Standardization, Geneva, Switzerland, 1987.
29. P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," *Advances in Cryptology – Crypto* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996. 10
30. P. Kocher, "Differential power analysis," `http://www.cryptography.com/dpa/index.html`. 10
31. M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM J. Computation*, Vol. 17, No. 2, April 1988. 4

32. National Bureau of Standards, NBS FIPS PUB 81, "DES modes of operation," U.S Department of Commerce, 1980. 4
33. National Institute of Standards, FIPS 180-1, "Secure hash standard," April 1995. 7
34. K. Ohta and T. Okamoto, "On concrete security treatment of signatures derived from identification," *Advances in Cryptology – Crypto* 98 *Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998. 9
35. D. Pointcheval and J. Stern, "Security proofs for signatures," *Advances in Cryptology – Eurocrypt* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1070, U. Maurer ed., Springer-Verlag, 1996. 9
36. D. Pointcheval and J. Stern, "Provably secure blind signature schemes," *Advances in Cryptology – ASIACRYPT* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1163, M. Y. Rhee and K. Kim ed., Springer-Verlag, 1996. 9
37. B. Preneel and P. van Oorschot, "MD-x MAC and building fast MACs from hash functions," *Advances in Cryptology – Crypto* 95 *Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995. 7
38. RSA Laboratoris, "PKCS," `http://www.rsa.com/rsalabs/pubs/PKCS/`. 8
39. V. Shoup and A. Rubin, "Session key distribution using smart cards," *Advances in Cryptology – Eurocrypt* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1070, U. Maurer ed., Springer-Verlag, 1996. 9
40. A. C. Yao, "Theory and applications of trapdoor functions," *Proceedings of the* 23rd *Symposium on Foundations of Computer Science*, IEEE, 1982. 3

Introduction to Secure Computation

Ronald Cramer

Dept. of Comp. Sc., ETH Zurich.
cramer@inf.ethz.ch
http://www.inf.ethz.ch/personal/cramer

Abstract. The objective of this paper[1] is to give an elementary introduction to fundamental concepts, techniques and results of Secure Computation.
Topics covered include classical results for general secure computation by Yao, Goldreich & Micali & Wigderson, Kilian, Ben-Or & Goldwasser & Wigderson, and Chaum & Crépeau & Damgaard.
We also introduce such concepts as oblivious transfer, security against malicious attacks and verifiable secret sharing, and for some of these important primitives we discuss realization.
This paper is organized as follows.
Part I deals with oblivious transfer and secure (general) two-party computation.
Part II discusses secure general multi-party computation and verifiable secret sharing.
Part III addresses information theoretic security and presents detailed but elementary explanations of some recent results in Verifiable Secret Sharing and Multi-Party Computation.
The importance of theory and general techniques often lies in the fact that the true nature of security is uncovered and that this henceforth enables to explore what is "possible at all". This then motivates the search for concrete and often specialized realizations that are more efficient. Nevertheless, many principles developed as part of the general theory are fundamental to the design of practical solutions as well.

Part I
Secure Two-Party Computation

1 Oblivious Transfer and Match-Making

Suppose there are two politicians who want to find out whether they both agree on a certain matter. For instance, they may be discussing a controversial law that has been proposed. Clearly, they could decide just to announce to each other

[1] This paper is based on a lecture given by the author at the 1998 Aarhus Summerschool in Cryptography and Data Security. An updated and extended version may be obtained from the author in the near future.

I. Damgård (Ed.): Lectures on Data Security, LNCS 1561, pp. 16–62, 1999.

their opinion, and both of them could determine whether there is agreement. But this has a drawback that careful politicians may wish to avoid. If only one of them supports that controversial law, he may lose face.

In other words, what they need is a method allowing two players to figure out if they both agree but in such a way that if they don't, then any player that has rejected the matter has no clue about the other player's opinion. Moreover, they may want to be able to carry out the method over a distance.

Technically, we can model the situation as follows. There are two players, A and B, and each of them has a secret bit. Say that A has the bit a and B has the bit b.

They want to compute $a \cdot b$ (which corresponds to the logical AND of a and b, and hence it is 1 if and only if $a = b = 1$) so that

- *Correctness*: none of the players is led to accept a false result.
- *Fairness*: each learns the result $a \cdot b$.
- *Privacy*: each learns nothing more than what is implied by the result and the own input.

Indeed, if A for example holds $a = 0$, then $a \cdot b = 0$, regardless of the value of b. Therefore, B's choice b remains unknown to A in this case.

We construct a solution to this "Match-Making" problem based on an important primitive *Oblivious Transfer* (OT) (more precisely: "chosen one-out-of-two oblivious transfer"). An OT is a protocol between two players, a sender S and a receiver R, that achieves the following. S has two secret *input bits*, b_0 and b_1, and R has a secret *selection bit* s. At the end of the protocol, which may consist of a number of exchanges of information between S and R, R obtains the bit b_s, but without having obtained any information about the other bit b_{1-s} (sender security). On the other hand, S does not get any information about the selection bit s (receiver security).

We use $\mathrm{OT}(b_0, b_1, s) = b_s$ to denote the output of an OT protocol as a function of the inputs. It is useful to observe that b_s is actually equal to $(1 \oplus s)b_0 \oplus sb_1$, where the operations are the usual multiplication and addition of bits.

Sender		Receiver
In: $b_0, b_1 \in \{0,1\}$		**In**: $s \in \{0,1\}$
	$S \to R : \mathrm{OT}(b_0, b_1, s)$ $\longleftarrow$	
		Out: b_s

Although at this point it is not clear whether OT-protocols exist and if so, how to construct them, we can already solve the Match-Making problem by assuming an OT-protocol!

This is how. If A and B now execute the OT-protocol with A acting as the sender and B as the receiver, using $b_0 = 0$ and $b_1 = a$, and $s = b$ as their

respective inputs, we see that B gets the value ab as output. In other words, $\mathrm{OT}(0, a, b) = (b \oplus 1)0 \oplus ba = ab$. Finally, B simply reveals ab to A, so that both learn the result. And indeed, if $b = 0$, then $ab = 0$ no matter what a is and player B learns nothing more about a by the properties of OT. If $a = 0$, then from the fact that the OT-protocol doesn't leak information about b and the fact that in this case B returns 0 to A in the final step no matter what b is, A doesn't learn nothing more about b as a result of the complete protocol.

Note that with respect to correctness and fairness, we have to assume that B indeed sends the correct value ab to A. Furthermore, we must assume here that both players take their actual choices as input to the protocol. But in any case, we can say that the protocol is secure for both parties if they are *semi-honest*. This means that both follow the rules of the game, but may try to learn as much as possible about the other player's input. We must also assume that no *crash-failures occur*, i.e. both players remain operational throughout the protocol and don't fail.

1.1 Historical Notes

Oblivious Transfer was originally introduced by M. O. Rabin [71], in a slightly different way. Namely, in his definition, the sender has just one bit b, and at the end of the protocol the receiver gets the bit b with probability $1/2$. Otherwise the receiver gets "?", and doesn't receive the bit. The sender cannot tell what happened.

Even, Goldreich and Lempel [40] later defined OT as we use it here, except that they require the selection bit to be random. It turned out that Wiesner [74] had earlier devised a similar definition in unpublished work.

The definition used here has appeared in many works on OT.

Soon after the invention of OT by Rabin, M. Blum [16] has conceived *coin-flipping over the phone* and *certified electronic email* as applications of OT.

2 Variations and Other Applications of OT

The Match-Making protocol is in fact just a toy example. Oblivious Transfer is an important primitive with many powerful applications, as we shall see.

2.1 OT of Strings

Suppose that instead of bits b_0 and b_1, the sender in OT has two strings $x_0, x_1 \in \{0,1\}^n$. Can we perform an OT where the sender receives the string x_0 if his selection bit s is 0 and the string x_1 otherwise? Note that "bitwise" OT of the n pairs of bits x_0^i, x_1^i of x_0 and x_1 is clearly not an option, since a cheater can for instance learn bits of both strings, which contradicts the requirements of string OT (whose definition is the obvious extension of the definition of OT of bits).

A general approach to this problem of oblivious transfer of strings is due to Brassard, Crépeau and Sántha and appears in [17]. They define *zig-zag functions*.

Consider a function $h : \{0,1\}^m \to \{0,1\}^n$, and for an arbitrary subset J of $\{1,\ldots,m\}$ and arbitrary $y \in \{0,1\}^m$, let y_J denote $y = (y_1,\ldots,y_m)$ restricted to the $|J|$ bits y_i with $i \in J$.

A function h is a zig-zag if for any $I \subset \{1,\ldots,m\}$, there is a $J \in \{I, I^c\}$ such that for any $x \in \{0,1\}^n$ and for a uniformly random chosen h-pre-image y of x, y_J gives no information about $h(y) = x$.

In other words, for any fixed subset of the m bits of y, it holds that either this subset of the bits or the remaining bits give no information about $h(y) = x$, and which of the two cases actually hold, does not depend on x or y.

Given such a function h, this is how one can perform chosen one-out-of-two oblivious transfer of n-bit strings x_0 and x_1. First the sender selects random y_0 and y_1 such that $h(y_0) = x_0$ and $h(y_1) = x_1$. Say that the receiver wishes to get the string x_s. Then they execute for $i = 1 \ldots n$ the protocol $\mathrm{OT}(y_0^i, y_1^i, s)$. As a result, the receiver gets all bits of y_s, applies h to it and gets x_s. Clearly, the sender has no information about s.

Let's see why it is true that at least one of the strings x_0, x_1 remains as unknown to the receiver as before the protocol. It is clear, by inspection of the protocol and the properties of OT, that even if the receiver deviates from the steps above there is some $I \subset \{1,\ldots,m\}$ such that he receives at best $y_{0,I}$ and y_{1,I^c}. Let J, with $J = I$ or $J = I^c$, be as in the definition of a zig-zag function. Then the receiver obtains at best $y_{0,J}$ and y_{1,J^c} and he has no information about $h(y_0) = x_0$, *or* obtains y_{0,J^c} and $y_{1,J}$ and he has no information about $h(y_1) = x_1$, since J only depends on h and I.

Constructions of zig-zag functions can be based on linear codes. It is easy to see that it is sufficient to construct a binary matrix with n rows such that for any subset I of the columns it holds that I or I^c has maximal rank n. Finding preimages can be done efficiently using basic linear algebra. Here is a small example with $n = 2$ and $m = 3$: the first column has entries 1 and 0, the second 1 and 1, and the third 0 and 1. Examples for larger values of n can be found for instance using recursion in combination with Vandermonde matrices, working over extension fields [17].

2.2 Oblivious Common String Verification

We describe a nice application of oblivious string transfer due to Fagin, Naor and Winkler [41]. There are two players A and B, and each of them holds some secret n-bit string. Their goal is to obliviously verify whether those strings are equal: as a result both of them should learn whether the strings are equal, but nothing more than that. Obviously, a secure protocol for this task can be used as a means of identification in a number of scenarios.

This is how the FNW protocol works. A has $x = (x_1,\ldots,x_n) \in \{0,1\}^n$ and B has $y = (y_1,\ldots,y_n) \in \{0,1\}^n$ as private input. For $i = 1 \ldots n$, A selects random k-bit strings $r_{i,0}$ and $r_{i,1}$, and B selects random k-bit strings $s_{i,0}$ and $s_{i,1}$. The parameter k is a security parameter. In the following, if u and v are n-bit strings, then $u + v$ denotes the n-bit string whose i-th bit is equal to the sum (modulo 2) of the i-th bit of u and the i-th bit of v, $i = 1 \ldots n$.

Consider the bit strings $\alpha = \sum s_{i,x_i}$, $\beta = \sum r_{i,y_i}$, $\alpha' = \sum r_{i,x_i}$ and $\beta' = \sum s_{i,y_i}$. If $x = y$, then clearly $\alpha + \alpha' = \beta + \beta'$. Otherwise, these values are different with probability $1/2^k$ (so in order for the error to be small, k must be large).

Note that A and B can obtain the strings α, resp. β by one-out-of-two string OT. The values α' and β' can be computed by A, and B respectively from their own random choices and their input strings. This is the complete protocol.

A		B
In: $x \in \{0,1\}^n$		**In**: $y \in \{0,1\}^n$
For $i = 1 \ldots n$,		For $i = 1 \ldots n$,
$r_{i,0}, r_{i,1} \in \{0,1\}^k : random.$		$s_{i,0}, s_{i,1} \in \{0,1\}^k : random.$
	$A \rightarrow B$: For $i = 1 \ldots n$, $\mathrm{OT}(r_{i,0}, r_{i,1}, y_i)$	
	$B \rightarrow A$: For $i = 1 \ldots n$, $\mathrm{OT}(s_{i,0}, s_{i,1}, x_i)$	
$\alpha = \sum_{i=1}^n \mathrm{OT}(s_{i,0}, s_{i,1}, x_i)$		$\beta = \sum_{i=1}^n \mathrm{OT}(r_{i,0}, r_{i,1}, y_i)$
$\alpha' = \sum_{i=1}^n r_{i,x_i}$		$\beta' = \sum_{i=1}^n s_{i,y_i}$
	$\alpha + \alpha'$ $\longrightarrow$	
	$\longleftarrow$ $\beta + \beta'$	
Out: $\alpha + \alpha' \stackrel{?}{=} \beta + \beta'$		**Out**: $\alpha + \alpha' \stackrel{?}{=} \beta + \beta'$

Note that if one of the parties is honest, and the other party has some y_i that differs from x_i, then the latter receives a completely random string in the final exchange. There exists a variety of other solutions to this particular problem. See [37] for a more efficient solution based on OT. Both [41] and [37] additionally survey completely different approaches not based on OT.

2.3 A Reduction

Crépeau [33] has shown that the OT as defined by Rabin (Rabin-OT, see Section 1.1) and chosen one-out-of-two OT are the same in the sense that one can be simulated from the other in a blackbox fashion, vice versa.

Given chosen one-out-of-two OT as a subroutine, Rabin-OT can be simulated as follows. The sender in Rabin-OT has a bit b to be obliviously transferred. First, the sender selects bits b_0 and b_1 at random such that $b_0 \oplus b_1 = b$, and the receiver selects a random selection bit s. After they have executed $\mathrm{OT}(b_0, b_1, s)$, the sender selects a random bit t and sends (t, b_t) to the receiver. With probability $1/2$, t is different from s and hence the receiver obtains both bits b_0 and b_1 and computes $b = b_0 \oplus b_1$. Also with probability $1/2$, the receiver gets the bit he

already had, leaving him in the dark about the other bit by the properties of chosen one-out-of-two OT and hence about b. The sender doesn't know what happened, since he doesn't learn s.

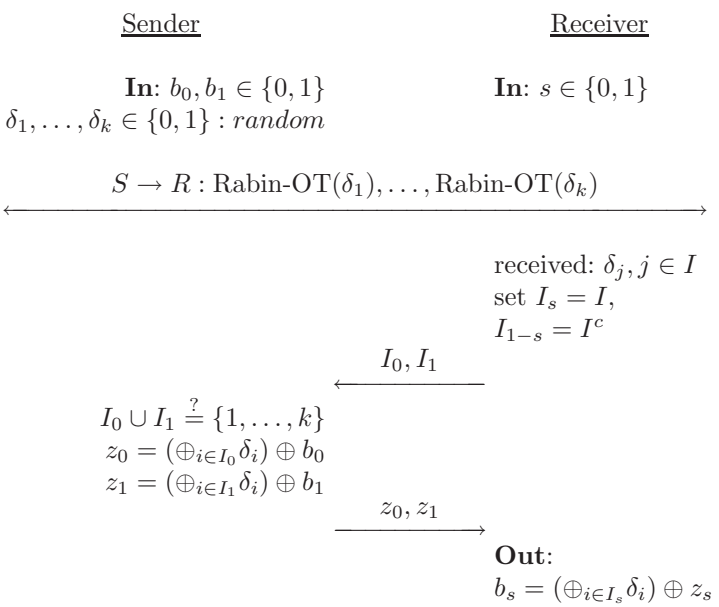
The interesting case is to simulate chosen one-out-of-two OT from Rabin-OT. So let the sender have input bits b_0 and b_1, and let the receiver have a selection bit s. Furthermore, Rabin-OT is at their disposal as a subroutine.

The sender chooses k random bits $\delta_1, \ldots, \delta_k$, where k is a*security parameter*. This value should be chosen large enough so that some error probability (to become clear later on) is small enough to be acceptable.

Next, using Rabin-OT, the sender transmits these bits one by one to the receiver, who is expected to receive roughly half of them. It is important to note that with probability $1 - 1/2^k$, at least one of the bits is not received, and with the same probability some bit is received.

Let $I \subset \{1, \ldots, k\}$ denote the collection of j such that δ_j has been received. Likewise, I^c refers to the bits that have not arrived. Having selection bit s, the receiver writes $I_s = I$ and $I_{1-s} = I^c$, and sends the ordered pair (I_0, I_1) to the sender. The sender now knows that the receiver obtained the bits corresponding to I_0 *or* I_1, but both events are equally likely from his point of view by the properties of Rabin-OT. Next, the sender adds all bits $\delta_i, i \in I_0$ to b_0 and the bits $\delta_i, i \in I_1$ to b_1, and sends the resulting two bits to the receiver. Since the latter knows all bits $\delta_i, i \in I_s$, he can recover the bit b_s, as required. It's clear that the sender has no clue about s.

Finally, consider a cheating receiver, who might define the sets I_0, I_1 differently, and perhaps learn more. However, if I_0 and I_1 cover the full set $\{1, \ldots, k\}$, then with probability $1 - 1/2^k$ at least one of the sets, say I_0, contains an index referring to a bit not received, which is hence completely unknown. In this case, the receiver doesn't learn b_0.



3 Constructions of OT-Protocols

For OT protocols to exist, we *must* make assumptions about the world in which the players operate, for instance related to the communication channel connecting the players, or their computational abilities.

However, besides its elegance and usefulness in protocol design, it is interesting to note that OT can be implemented under a wide variety of different such assumptions.

Among these, the difficulty of factoring large random composite integers, the Diffie-Hellman problem (related to the difficulty of computing discrete logarithms), and abstract, general assumptions such as the existence of trapdoor one-way permutations (which can be implemented under the RSA assumption) [54]. But physical assumptions suffice as well, such as the OT based on noisy communication channels of Crépeau and Kilian [36].

3.1 Necessity of Assumptions

Why doesn't OT exist unconditionally? Indeed suppose that a protocol for OT exists, making no assumptions on the computational abilities of the players, the communication channel or whatever.

Then there are programs used by sender and receiver to compute the exchanged messages, that given random strings and the input bits would operate deterministically. Moreover, we may assume that the players communicate over

a perfect channel, and that the players have infinite computing power. Say that the protocol achieves perfect correctness and always halts.

This leads to a contradiction as shown by the following (informal) argument, even if we assume that the players are semi-honest.

Given OT, there exists a protocol with similar characteristics for two players A and B to obliviously evaluate the AND of their input bits a and b (see Section 1). We show that such a protocol does not exist.

Let $\mathcal{T}$ denote the sequence of messages exchanged in a completed execution of the protocol ($\mathcal{T}$ is called *transcript*). Write $x_A \in \{0,1\}^*$ and $x_B \in \{0,1\}^*$ for the respective random strings used by A and B in the computation.

Say that $a = 0$. We argue that a semi-honest A having $a = 0$ as input can always figure out the value of B's input b, thus contradicting the security conditions.

If $b = 0$, then there exists a random string $x'_A \in \{0,1\}^*$ such that $\mathcal{T}$ is consistent with A having input $a = 1$ instead of $a = 0$. This follows from the fact that B, having $b = 0$ as input, has no information about A's input. Clearly, fixing the transcript $\mathcal{T}$ and an arbitrary x'_A, and setting $a = 1$, A can effectively decide whether the transcript is consistent with x'_A and $a = 1$. Since we do not assume limits on the computational power of the players, A eventually finds such string x'_A.

In case that $b = 1$, it is clearly impossible that $\mathcal{T}$ is consistent with $a = 1$ and some x''_A, since in this case flipping A's input from 0 to 1, changes the logical AND of the inputs: since we assumed perfect correctness, $\mathcal{T}$ cannot be consistent with two pairs of inputs (a, b) whose respective logical AND is different.

Therefore, A decides that $b = 0$ if there exists x'_A such that $\mathcal{T}$ is consistent with x'_A and $a = 1$, and decides that $b = 1$ if no such x'_A exists.

Similar arguments apply to the OR-function. Based on information-theory, one can find a more general argumentation.

3.2 Rabin-OT

We present a version of the original Rabin-OT [71]. Let n be the product of two distinct, large random primes p and q. By the assumption that factoring large random composite integers is infeasible [2], it is hard to retrieve p and q given just n.

However, it's easy to generate such n with known factorization. Testing primality can be done efficiently [3], and by the Prime Number Theorem, the fraction of primes smaller than K is roughly $1/\ln K$ for large K. Therefore, one can just select a random large integer and test it for primality. After some tries one finds a random integer that one knows is prime. Multiplying two such primes gives n.

[2] though certainly not *impossible*.

[3] i.e., certainly in practice. There is also a theoretical result by Adleman and Huang, extending a result by Goldwasser and Kilian, saying that primality can be tested in probabilistic polynomial time, with a negligible probability that no decision is made.

Rabin-OT is based on the number-theoretic fact that given two square roots x and z of a square y modulo n, that do not differ by a sign, one can efficiently compute p and q from those roots and n. Indeed, from $x^2 \equiv z^2 \bmod n$ we get $(x+z)(x-z) \equiv 0 \bmod n$. And since $x \not\equiv \pm z \bmod n$, n doesn't divide $(x+z)$ and doesn't divide $(x-z)$, yet it divides $(x+z)(x-z)$. This is only possible if p divides exactly one of the two terms, and q divides the other. We now just compute the greatest common divisor of n and $(x-z)$ and the greatest common divisor of n and $(x+z)$, to get both factors p and q. Note that greatest common divisor can be efficiently computed using for instance Euclid's algorithm.

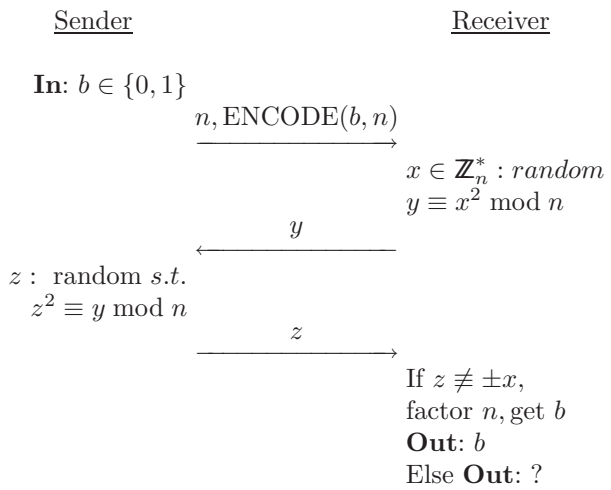
Each square y modulo n has four distinct square roots. Indeed, modulo each of the factors p and q, there are two square roots. Combining them with the Chinese Remainder Theorem, we get 4 distinct roots modulo n.

From the difficulty of factoring, and the analysis above, we conclude that that squaring modulo n is a *one-way function*, i.e. given just n and a random square y modulo n, it is infeasible to find a square root of y. Indeed, if this were not so, then one would select a random x, compute y as the square modulo n of x and compute a square root z of y given just y and n. With probability $1/2$, $x/z \bmod n$ is a non-trivial root of 1, and one can factor n efficiently.

On the other hand, if one knows p and q, computing a root of a square is efficient. It's easy to explain in the case that p and q are both 3 mod 4. Let y be a square modulo p, and write $z^2 \equiv y \bmod p$. Define $x \equiv y^{(p+1)/4} \bmod p$. Then $x^2 \equiv y^{(p+1)/2} \equiv z^{p+1} \equiv z^2 \equiv y \bmod n$. Same story for computing square roots modulo q. So if one has a square modulo n and one knows p and q (both of them 3 mod 4), one projects the problem modulo n on the factors p and q, computes square roots, and lifts it back with the Chinese Remainder Theorem. If p and q, are not both 3 mod 4, it's more complicated. We say that squaring modulo n is a *trapdoor* one-way function.

Without giving further details, we state that it is possible to encode a bit b as an integer modulo n using a public function ENCODE(b, n), such that it is hard to retrieve b given just ENCODE(b, n) and n, but easy given the trapdoor for n as well, i.e. its factorization.

The protocol works as follows. The sender encodes the bit b that is to be sent by Rabin-OT by computing ENCODE(b, n). After receiving n and ENCODE(b, n) from the sender, the receiver selects a random x modulo n and sends its square y modulo n to the sender. Note y perfectly hides which out of the four possible roots the receiver has chosen. The sender, knowing p and q, can efficiently compute a random square root z of y and returns it to the receiver. With probability $1/2$, z does not differ by a sign from x, and the receiver can factor n, and efficiently retrieve b from ENCODE(b, n). Otherwise, also with probability $1/2$, $z \equiv \pm x \bmod n$, and the receiver doesn't get the factorization of n, and hence doesn't get closer to learning b.



It is assumed that both players are semi-honest. For sender security we have to assume that the receiver is computationally bounded. The security of the receiver is unconditional.

3.3 OT Based on RSA

We give an example for chosen one-out-of-two OT based on RSA [72], the well-known public-key encryption scheme which R. Rivest, A. Shamir and L. Adleman introduced in 1978. We assume that both players are semi-honest. The sender selects two large random distinct primes p and q, and computes $n = pq$, the *modulus*. Next, the sender selects an integer *exponent* e such that e is relatively prime to $(p-1)(q-1)$. Let the integer d satisfy $de \equiv 1 \bmod (p-1)(q-1)$ (given p, q and e such d is easy to compute). Now we have $(x^e)^d = (x^d)^e \equiv x \bmod n$ for all x. The sender keeps d secret, and sends n, e (public key) to the receiver.

It has been proved by Alexi, Chor, Goldreich and Schnorr [1] that if a plain-text x is chosen at random, guessing the least significant bit of x, given just the ciphertext $y = x^e \bmod n$, n and e, significantly better than at random, is as hard as finding all bits of x. This is called a *hard-core bit* for the RSA function. Note that this result does not follow directly from the usual RSA-security assumption. That assumption only states that it is infeasible to recover *all* bits of x from y. In the protocol to follow, the sender in OT exploits the existence of hard-core bits to "mask" his bits b_0 and b_1.

The receiver, having selection bit s, chooses a random plain text $m \bmod n$ and computes the cipher text $c_s \equiv m^e \bmod n$. Let r_s denote the least-significant bit of the plain-text m.

The receiver selects the ciphertext c_{1-s} as a random integer modulo n and communicates the pair of ciphertexts (c_0, c_1) to the sender. The sender, knowing the secret RSA-key, computes for each of those ciphertexts their respective least-significant bits r_0 and r_1. Now the sender masks the bits b_0 and b_1 by setting

$b'_0 = b_0 \oplus r_0$ and $b'_1 = b_1 \oplus r_1$, and sending them to the receiver. The receiver recovers b_s by computing $b'_s \oplus r_s$. The bit b_{1-s} remains concealed, since he cannot guess r_{1-s} with high enough probability. Note that the selection bit s is unconditionally hidden from the sender, and that we have to assume that the receiver is semi-honest in order to guarantee sender security.

This is essentially the OT protocol of Goldreich, Micali and Wigderson [54], which not only works for RSA but any other trapdoor one-way permutation as well (though in general, more care has to be taken to define a hard-core bit).

4 General Secure Two-Party Computation

It is a natural question to ask which functions other than AND or string equality can be obliviously evaluated. It is the answer to this question that demonstrates the power of oblivious transfer: *all* functions f with finite domain and finite image can be obliviously evaluated. This is due to A. Yao [75], who based his result on the assumption that factoring integers is intractable. The protocol below shows the stronger result saying that the existence of OT is sufficient for this task. This is due to O. Goldreich and R. Vainish [55].

For simplicity, think of a function $f : \{0,1\}^{n_A} \times \{0,1\}^{n_B} \rightarrow \{0,1\}$, where n_A and n_B denote the number of input bits player A and B hold.

The function f is assumed to be efficiently computable (polynomial time on a Turing-machine) and both players have agreed on a *Boolean circuit* computing f (so in particular they both know f):

- a directed acyclic graph with
- $n_A + n_B$ input nodes, and one output node.
- The remaining vertices are labelled as binary negation, and two-input binary addition and multiplication gates. Note that these operations correspond to binary NOT, XOR and AND. The outputs of internal gates can be led to an arbitrary number of other gates (arbitrary fan-out).
- The topology of the graph dictates the flow of the values on which the computations are performed. More precisely, the circuit computes f in the sense that if one assigns the bits of any input strings a, b to the input nodes, and inductively propagates the values resulting from the computations performed on them (according to the logic of the gates), then the output node will be set to $f(a, b)$.

It is well known that all computable functions f can be computed by Boolean circuits and that a Boolean circuit computing f can be constructed with a number of nodes (gates) polynomial in the number of inputs (i.e. $n_A + n_B$ in this case) if f is efficiently computable.

The problem of oblivious function evaluation of f is as follows. Player A has input $a \in \{0,1\}^{n_A}$, and player B has input $b \in \{0,1\}^{n_B}$. For fixed input a, b, and a fixed circuit computing f, the computation graph is the graph representing the circuit but with the flow of the values written on the edges. For a given gate in the computation graph, we speak of the *actual inputs* and the *actual output*.

We try to devise a protocol for A and B to execute such that both learn $f(a, b)$, but none of them learns more than what is implied by $f(a, b)$ and the own input. In the following we assume that neither player crashes, and that both of them are semi-honest.

The protocol consists of three stages.

Input Sharing. For each of the n_A bits a_i of his input string a, A selects two bits $s_{i,A}$ and $s_{i,B}$ at random such that $s_{i,A} \oplus s_{i,B} = a_i$ and sends $s_{i,B}$ to B. Player B does the same to the n_B inputs bits b_i of his input string b, resulting in $t_{i,A}$ and $t_{i,B}$. This is an *additive secret sharing of the inputs*, and the s and t values above are called *shares*.

Computation. The computation proceeds inductively and in a gate by gate manner, possibly handling many gates in parallel. The players maintain the following *invariant*. The actual inputs to the current gate are additively shared. After processing of the current gate, there are uniformly random shares u_A (held by A) and u_B (held by B) such that $u_A \oplus u_B$ equals the actual output of the current gate and such that neither player has increased knowledge about the actual output.

Output Reconstruction: Each player reveals his share in the output bit of the computation. The sum of these shares equals the output bit $f(a, b)$.

It remains to be shown how this invariant is maintained for each of the three types of gates.

4.1 Addition-Gates

Let x_0, x_1 denote the actual input bits, and let $x = x_0 \oplus x_1$ denote the actual output bit. Then A holds $x_{0,A}$ and $x_{1,A}$, and B holds $x_{0,B}$ and $x_{1,B}$ such that $x_{0,A} \oplus x_{0,B} = x_0$ and $x_{1,A} \oplus x_{1,B} = x_1$.

Player A computes $x_A = x_{0,A} \oplus x_{1,A}$ as his share in the actual output bit x of the current gate. For B there is a similar program, resulting in a share x_B. We have $x = x_A \oplus x_B$.

4.2 Negation-Gates

These are simply handled by designating one player, say A, who just flips his share in the actual input bit, and takes the result as his share in the actual output bit. B just takes his share in the actual input bit as his share in the actual output bit.

4.3 Multiplication-Gates

A more interesting case is multiplication. Again, let x_0, x_1 denote the actual input bits. Then A holds $x_{0,A}$ and $x_{1,A}$, and B holds $x_{0,B}$ and $x_{1,B}$ such that $x_{0,A} \oplus x_{0,B} = x_0$ and $x_{1,A} \oplus x_{1,B} = x_1$.

Before we proceed, let's take a look at OT once more. Suppose that player A has some bit α and that player B has some bit β. How can they arrive at the

situation where they hold random additive shares in $\alpha \cdot \beta$ but neither of them has gained information about $\alpha \cdot \beta$?

Let ρ be a secret random bit chosen by player A. If A and B now execute the OT-protocol with A acting as the sender and B as the receiver, using $b_0 = \rho$ and $b_1 = \alpha \oplus \rho$, and $s = \beta$ as their respective inputs, we see that B gets the value $\alpha\beta \oplus \rho$ as output. In other words, $\mathrm{OT}(\rho, \alpha \oplus \rho, \beta) = (\beta \oplus 1)\rho \oplus \beta(\alpha \oplus \rho) = \alpha\beta \oplus \rho$. A then just takes ρ as his share, and B takes $\alpha\beta \oplus \rho$ as his share. We only need to argue that A and B do not gain knowledge about each other's inputs as a result. Clearly, the security of OT implies that A doesn't gain knowledge about β, since it is B's selection bit. Again by the security of OT, B learns only one of ρ and $\rho \oplus \alpha$, and since ρ was chosen at random by A, this doesn't increase B's knowledge about α.

Sender A		Receiver B
$\rho \in \{0,1\}$ random		
In: $\alpha \in \{0,1\}$		**In**: $\beta \in \{0,1\}$
	$A \to B : \mathrm{OT}(\rho, \alpha \oplus \rho, \beta)$ $\longleftrightarrow$	
Out: ρ		**Out**: $\alpha\beta \oplus \rho$

Now we return to handling the multiplication gates. Note that

$$x = x_0 \cdot x_1 = (x_{0,A} \oplus x_{0,B})(x_{1,A} \oplus x_{1,B}) =$$

$$x_{0,A}x_{1,A} \oplus x_{0,A}x_{1,B} \oplus x_{1,A}x_{0,B} \oplus x_{0,B}x_{1,B}.$$

Two executions of OT with, say, A as the sender are sufficient to get to the random additive shares of x. A selects random bits ρ_{01} and ρ_{10}.

1. $A \to B$: $\mathrm{OT}(\rho_{01}, \rho_{01} \oplus x_{0,A}, x_{1,B}) = \rho_{01} \oplus x_{0,A}x_{1,B}$.
2. $A \to B$: $\mathrm{OT}(\rho_{10}, \rho_{10} \oplus x_{1,A}, x_{0,B}) = \rho_{10} \oplus x_{1,A}x_{0,B}$.

A takes as his share in x the bit $x_A = x_{0,A}x_{1,A} \oplus \rho_{01} \oplus \rho_{10}$, and B takes $x_B = x_{0,B}x_{1,B} \oplus x_{0,A}x_{1,B} \oplus x_{1,A}x_{0,B} \oplus \rho_{01} \oplus \rho_{10}$ as his share.

4.4 Complexity of the Protocol

By inspection, an upperbound on the communication costs of executing the protocol is $O(|C|)$ OT's and $O(|C|)$ bits (the latter is from the initial input sharing), where $|C|$ denotes the number of gates in the circuit computing the function f. Handling many gates in parallel, the round complexity is upper bounded by the *depth* of the circuit C, i.e. the length of the longest path in the graph of C.

4.5 Security Discussion

In order that the above oblivious circuit evaluation protocol satisfies the required *correctness* and *privacy* properties we have to assume that both players are *semi-honest*, i.e. they follow the protocol and behave exactly as required, but each of them separately may try to deduce from the information available to them as a result of the protocol execution as much as possible about the other player's inputs.

It is easy to see that if one of the players is *malicious* and deviates from the protocol, he can make the other player accept a false result, while he in fact knows the correct one. With an adequate definition of what it means for OT to be secure against malicious attacks, the protocol above would be private though. For *fairness*, we have to assume that neither player crashes before termination of the protocol.

The intuition behind the analysis of privacy is that the invariant maintained guarantees that at each point in the execution of the protocol, the players hold random additive shares in the actual outputs so far and that the respective shares of each player does not increase knowledge about the actual output so far. It is only at the end of the protocol where they have random additive shares in the actual output that are exchanged, enabling the reconstruction of the actual output.

Therefore, another way to look at the protocol is by saying that, conceptually speaking, it simulates a *trusted host*: a third party who is and can be trusted by both players. Given such a third party, both players secretly send their inputs to the host, who returns the function value to both players. This is called an *ideal protocol*.

In an actual proof, one has to show that each player on his own, given just his input and the result of the computation, is able to generate efficiently a simulation of the protocol that is indistinguishable from the ideal protocol.

Later we present protocols for the same task, that are secure against much stronger adversaries than semi-honest ones in a much broader context, and in fact, the security principles outlined above are the basis for defining security there as well (Beaver [4], Micali/Rogaway [65], Goldreich [57], Canetti [21]).

5 Example

As an illustration, let's return to the problem of Oblivious Common String Verification. We show that the general protocol provides a solution for this problem. There are good reasons to prefer the solution of Fagin, Naor and Winkler, mainly because an appropriate OT withstanding attacks by malicious rather than semi-honest players renders the complete FNW solution secure against this kind of attack.

But if we may assume the players are semi-honest, the following protocol is just as good. Two players A and B each hold some secret n-bit string. Write $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ for their respective strings.

Write $f(x_1, \ldots, x_n, y_1, \ldots, y_n) = (x_1 \oplus y_1 \oplus 1) \cdots (x_n \oplus y_n \oplus 1)$. It follows that $f(x, y) = 1$ if and only if $x = y$.

From this formula for f we can easily derive a Boolean circuit: there are n pairs of input bits (x_i, y_i). For each such pair, the bits in it are first led through a binary addition gate, after which the result is passed through a negation gate. Now there are n intermediate results, which only have to be led through an n-input binary multiplication gate. To be consistent with our description, we first write the n-input binary multiplication gate as a tree of depth $\log n$ with two-input binary multiplication gates only, and lead the intermediate results into it.

By the method from Section 4 A and B can now obliviously verify whether or not they have the same string. Note that $2n$ oblivious transfers and $2 \log n$ rounds of communication suffice.

6 Dealing with Malicious Attacks

Unfortunately, most protocols presented so far only work if the players are semi-honest. We first indicate the failures that occur in the examples we have given, if one of the players is cheating and deviates from the protocol, i.e. carries out a *malicious* attack. The rest of this section deals with methods to enhance the security of OT-protocols, achieving security even in the presence of a malicious attacker. We stress that we still assume that the players do not crash before the end of the protocol, to ensure fairness.

As an example of a failure, although Rabin-OT is secure for the sender if the receiver is semi-honest and factoring large integers is hard, it is not clear that a receiver *deviating* from the steps required in the protocol couldn't "extract" the factorization from the sender, even without being able to factor large numbers efficiently *in general*. It might be true that there exists a single number modulo n such that a square root of it reveals the factorization of n. Given such a number it would be easy for the receiver to get the factorization of the sender's modulus, since the sender returns a square-root of *any* number modulo n that the receiver sends. Hence, the receiver would always get the bit b. On the other hand, if the sender would choose the modulus n as the product of three primes for instance, he can influence the probability with which the receiver gets the bit b.

Fischer, Micali and Rackoff [44] presented the first realization of OT *secure against malicious attacks*, i.e. it provides security for sender and receiver even if one of them deviates arbitrarily from the protocol.

It is easy to see that the scheme based on RSA we presented is totally insecure against malicious attacks by the receiver: nothing prevents the receiver from computing the ciphertext c_{1-s} in the same fashion as c_s, in which case the receiver retrieves both b_0 and b_1 at the end of the OT.

6.1 Notion of Security of Basic OT

We assume that at most one of the players carries out a malicious attack. These are the minimum (and for some applications sufficient) requirements we have to make in order for OT resisting malicious attacks to make sense.

1. If the sender is honest (so the bits b_0 and b_1 are well-defined) throughout the protocol, "no matter" how the receiver plays (note that if the receiver is corrupt, the selection bit s may not even be well-defined in general), at least one of the bits b_0, b_1 remains "completely" unknown to him. [4]
2. If the receiver is honest throughout the protocol, "no matter" how the sender plays, the selection bit s remains "completely" unknown to the sender. Moreover, the receiver always gets some bit, or else just aborts and the sender is deemed corrupt.

Under this definition, the string-equality protocol of [41] as presented in Section 2.2 is secure against a malicious attack by one of the players, for instance.

Beaver [9,10] has a simulation based definition of secure OT.

6.2 A General Solution in the Cryptographic Scenario

Goldreich, Micali and Wigderson [54] have a general defense against malicious attacks that works in principle for any OT based on intractability assumptions. We give an informal overview. It involves three other important primitives: *commitment schemes*, *mutually random coins* and *general zero knowledge techniques*. Interestingly, all these primitives (including OT) can be realized under the assumption that trapdoor one-way permutations exist.

Trapdoor One-Way Permutations. We assume that both players are restricted to probabilistic polynomial time computations, so that none of the players is computationally powerful enough to invert one-way permutations without knowing a trapdoor. More precisely, this means that if a trapdoor one-way permutation is selected at random by one party, then the other party, having access to the description of the forward function only, cannot efficiently invert a randomly chosen element from its range. The party knowing the trapdoor can efficiently invert the function. Why is it that one party does have the trapdoor while the other doesn't? This is by the existence of a special probabilistic polynomial time algorithm called *trapdoor permutation generator*. On input of a random bit string, the generator outputs a "random" one-way permutation *and* a corresponding trapdoor. RSA (see Section 3.3) is an example of a trapdoor one-way permutation.

[4] Actually, one must require that there is a bit s so that if b_s is *given* to the receiver, he still has no information about b_{1-s} This is to exclude the possibility that the receiver for instance learns $b_0 \oplus b_1$

Commitments. Conceptually, there is an analog between vaults and commitment schemes. Player A has some secret piece of information, and places the piece in a vault, locks it and memorizes the opening combination. He then passes the vault on to player B, to whom the secret information is *hidden* until he gets the secret opening information to open the vault. But in the mean time, player A cannot change the information stored in the vault, since it is no longer in his possession. Thus, the commitment is *binding*. At some later moment, player A can simply send the key of the vault to player B, who can then *open* it and read the information.

Cryptographic, non-physical realizations of commitment schemes, can for instance be based on RSA. Player A generates a key-pair $((n, e), (p, q))$ for RSA, and sends the public-key to player B. To commit to a bit b, A generates a random plaintext m, and computes the corresponding ciphertext c. Write ρ for its least significant bit. He sets $d = b \oplus \rho$ and sends (c, d) as the commitment to B. To open the commitment, A sends m and the bit b to B, who verifies that m is the plaintext corresponding to c and that d is equal to the sum of its least significant bit and b. The hiding property follows from the fact that the least significant bit is a hard-core bit (see Section 3.3). The commitment is binding since RSA is a permutation (if the public exponent e is a prime larger than the modulus n, for instance, B can efficiently verify that the public key defines a permutation without any further proofs from A, since then we have $\gcd((p-1)(q-1), e) = 1$ for sure and primality can be efficiently tested). Note that the binding property is unconditional and that the hiding property holds if B is polynomially bounded. In fact, it can be shown that *one-way permutations* are sufficient for commitments.

This seemingly innocent primitive has far reaching applications in cryptography. For instance, it is sufficient to implement general zero knowledge interactive proofs [53,56], a method that allows one to prove "anything provable" in zero knowledge, i.e. to convince a sceptical judge of the veracity of an assertion without giving anymore information away than the fact that the assertion is true. [5]

Mutually Random Coins. Another application of commitments is mutually random coins. Here players A and B want to establish a bit (or a string) that is random if one of them is honest. A simple protocol goes as follows. A selects a random bit b_A and sends to player B a commitment to it. Player B selects a random bit b_B and sends it to A, who opens the commitment. The bit b is defined as $b = b_A \oplus b_B$.

OT Secure against Malicious Attacks. Returning to the problem of defending against malicious attacks in OT, we now show how we can defend against these attacks by the techniques of [54].

[5] There is a vast literature dealing with general zero knowledge and commitment techniques, with many different flavours, styles and security and efficiency properties, but we do not discuss these any further here.

The key observation is that, for instance in the RSA based example of OT, if one of the players is honest the security of the other player is guaranteed. That's not what we want, since actually we want that a player's security is guaranteed if he is honest, no matter what the other player does. Nevertheless, in some sense this fact is the basis for achieving it: based on the primitives outlined above, an honest player can force the other player to be honest as well or else the protocol simply halts with no advantage for the corrupt player.

This has become an important design principle throughout the field of cryptography: often it is possible to start from a cryptographic protocol that is secure if its participants are semi-honest and to transform it into a protocol secure against malicious adversaries, by forcing each player to *prove* that he behaved as a semi-honest participant.

We start looking into the details. First of all, it's useful if the randomness used by each player is mutually random. However, it is in the interest of both players not to reveal their randomly chosen bits, for obvious security reasons. Say that each player needs at most l random bits. Then they execute the protocol for achieving a mutually random bit l times in parallel where the receiver is the committing party, and l times in parallel where the sender is the committing party. However, they do not open any of the commitments used. Note that in the first case this implies that the receiver knows the resulting mutually random bits whereas the sender does not. So the receiver can use these bits later on whenever they are required, and in fact we will explain how the sender can verify that the receiver used them, in a way that is secure for the receiver. The second case is of course similar, with the roles reversed.

Let's first look at the sender's security and let's look at the RSA-example from Section 3.3. The sender wants to make sure that the receiver gets at most one of the bits b_0, b_1. It is sufficient if the receiver can convince the sender of the veracity of the following assertion about the ciphertexts c_0, c_1. One of them is equal to some particular string of mutually random bits, and one of them is equal to the RSA-function applied to some other particular string of mutually random bits (in this case we refer to those mutually random bits that the receiver knows, but the sender doesn't). To protect the receiver in case he is honest, the means by which the receiver convinces the verifier of this assertion must be zero-knowledge.

Roughly speaking, it is now fairly easy though tedious for the sender and receiver to efficiently derive by themselves from what is known to both of them, a description of a function F and a function-value y such that the assertion about the ciphertexts c_0, c_1 is equivalent to saying that there exists an x with $F(x) = y$. Furthermore, if the receiver followed the protocol, he can actually efficiently determine such x.

The zero knowledge techniques from [53] are designed for exactly this technical situation! So in principle, the security of the sender can be guaranteed.

As to the receiver's security, his selection bit is protected by the fact that the proof of the assertion is zero knowledge.

Therefore, under fairly general intractability assumptions, OT that is secure against malicious attacks can be realized. However, it is very costly to resort to the powerful techniques underlying the defense. In concrete situations with specific implementations of OT, there may exist a more efficient way to enhance the security.

Oblivious Function Evaluation and Malicious Attacks. So, are we done and can we now use the OT with enhanced security directly in the general protocol from Section 4 and obtain general oblivious function evaluation secure against malicious attacks by one of the players?

No! There are many more things to be fixed first. For instance, in each current gate, the inputs used must be the same as the outputs of some earlier gates. Here a solution is to have both players always commit to their inputs at each current gate and have them prove to each other in zero knowledge that these commitments commit to the same values as the outputs of the gates that are supposed to deliver the inputs to the current gate. In particular, both players commit to their initial inputs.

Furthermore, using similar techniques as in the case of OT with strengthened security above, it is not so difficult anymore to handle the full set of instructions from Section 4 securely at all gates.

We return later to the techniques of GMW [54].

7 A Generic Solution

Another fundamental result is by Kilian [62], who shows constructively that OT is necessary and sufficient for general oblivious function evaluation, *even if* one of the players is malicious. From the previous section it should have become clear that this is by no means obvious.

Given OT as a black-box [6] and given a function to be obliviously evaluated and a circuit for it, there is a generic transformation that results in a set of protocols for the players to execute. It is immaterial how the OT works exactly: at those points in the protocols where OT is required, only calls are made to a black-box for doing OT. In the other direction, note that OT can be viewed as a oblivious evaluation of the function $f(b_0, b_1, s) = (s \oplus 1)b_0 \oplus sb_1$.

Another contribution of [62] concerns the round-complexity of general secure function evaluation, which is shown to be constant with polynomial size message complexity if the function can be computed by a polynomial size *formula* (i.e. the fan-out of the gates in the circuit is 1).

We do not overview a proof of Kilian's result, but only introduce some of its fundamental parts.

[6] It is beyond the scope of the present paper to discuss the exact security definition required for this result.

7.1 Commitment Based on OT

Kilian shows how a commitment scheme (which he attributes to Crépeau) can be simulated from OT as follows. Let b be the bit that player A wants to commit to. A and B agree on a security parameter k.

1. For $i = 1 \ldots k$, A selects a pair of random bits (r_i^0, r_i^1) such that $b = r_i^0 \oplus r_i^1$.
2. For $i = 1 \ldots k$, B selects a random bit s_i.
3. For $i = 1 \ldots k$, with A being the sender and B the receiver, they execute $OT(r_i^0, r_i^1, s_i)$
4. B takes the bits received, together with his own random choices, as A's commitment.
5. To open the commitment, A reveals the bit b and, for $i = 1 \ldots n$, the ordered pairs (r_i^0, r_i^1). Player B accepts the opening if and only if this information is consistent.

Player A's cheating probability is at most $1/2^k$: if A were able to open the commitment in two different ways, he would have to guess all of B's random bits, so the binding property is satisfied. The hiding property follows immediately from the definition of OT.

7.2 Committed Oblivious Transfer (COT)

COT is as OT, except that

1. Initially, the sender is committed to his input bits b_0, b_1, and the receiver is committed to his selection bit s.
2. At the end of the protocol, the receiver is committed to the received bit b_s.

An alternative proof of Kilian's result can be found in [35], who introduce COT and show that it is sufficient for secure function evaluation tolerating a malicious attacker, and that COT can be simulated from OT (they don't treat the constant round issue though). The latter construction involves so-called envelope techniques and error correcting codes.

8 Other Work

Some suggestions for further reading about defining OT secure against malicious attacks and constructions of secure OT: Beaver addresses the pitfalls in attempts to define OT secure against malicious attacks and presents solutions and constructions [7,9]. In [10], he gives a precise definition of OT so that when used in a multi-party computation protocol the protocol as a whole is secure against a malicious *adaptive* attacker.

The above references and Crépeau [34] (besides those references we already mentioned) contain a host of other interesting references.

Part II
General Secure Multi-party Computation

9 Introduction

The protocols from Sections 4 and 6.2 have an obvious extension from two players to n players guaranteeing correctness and privacy. This is done by using n-out-of-n additive sharing of bits and executions of OT between every pair of players. In this case, *privacy* of a single player is guaranteed even if the $n-1$ other players pool their complete views on the protocol. The extension to $n > 2$ players of the protocol from Section 6.2 is even secure against malicious attacks.

However, the fairness condition is only fulfilled by making a strong assumption on the behaviour of the players, since one party can leave the protocol knowing the result of the computation whereas the other remain ignorant about it, or simply disrupt it in an early stage.

An important contribution of Goldreich, Micali and Wigderson [54], is that they explain how *privacy* can be traded for *fairness*. In fact, they achieve the stronger property of *robustness*: it is not only infeasible for corrupted parties to walk away prematurely with the result of the computation and leaving the remaining players ignorant about it, they can't disrupt the computation at all: if the corrupt players leave the computation, the remaining ones will still be able to complete the computation.

More precisely, they show that even if at most a *minority* of the players perform a coordinated malicious attack, then correctness, privacy *and* robustness can be guaranteed.

Apart from GMW-techniques we discussed in Section 6.2, they employ what is called *verifiable secret sharing*, which was first introduced by B. Chor, S. Goldwasser, S. Micali and B. Awerbuch [27].

Before sketching the full protocol of GMW, we introduce secret sharing and verifiable secret sharing in the next sections.

10 Secret Sharing with Semi-Honest Participants

In a secret sharing scheme there is a *dealer* and a number of *agents*. The dealer holds some secret string s, and sends *shares* of s privately to each of the agents. These shares are computed in such a way that only certain specific subsets of the agents can *reconstruct* the secret s by pooling their shares, while others have no information about it.

Secret sharing was invented independently by A. Shamir [73] and B. Blakley [15] in 1979. Their solutions allow the dealer to consider any number n of agents and any *threshold* $t \leq n$, such that from any subset of size at least t of the shares the secret s can be reconstructed uniquely and efficiently, whereas sets containing less than t shares contain no information at all about s.

We explain Shamir's scheme which is based on *Lagrange-interpolation* over finite fields. We assume that all parties involved are *semi-honest*.

We use the following version of Lagrange Interpolation. Let K be a (finite) field. Suppose we are given any $t \geq 1$ points $(p_1, q_1), \ldots, (p_t, q_t)$ in the plane K^2, where the p_i's are all different. Then there is a unique polynomial $f(X) \in K[X]$ of degree smaller than t, that passes through these t points, i.e. $f(p_1) = q_1, \ldots, f(p_t) = q_t$.

First we discuss existence. For each $1 \leq i \leq t$ define the polynomial

$$f_i(X) = \frac{\prod_{1 \leq j \leq t, j \neq i}(X - p_j)}{\prod_{1 \leq j \leq t, j \neq i}(p_i - p_j)}.$$

Observe that each f_i has degree exactly $t - 1$ and that $f_i(p_i) = 1$ whereas $f_i(p_j) = 0$ if $j \neq i$.

But then it follows immediately that the following polynomial f does the trick (Lagrange interpolation formula).

$$f(X) = \sum_{1 \leq i \leq t} q_i \cdot f_i(X).$$

Note that $f(X)$ has degree *at most* $t - 1$. Indeed, it can be strictly smaller than $t - 1$.

As to uniqueness, note that if there were a polynomial $f'(X) \in K[X]$ of degree smaller than t that agrees with f on all t points, then the polynomial $f - f' \in K[X]$ has t zeroes while its degree is smaller than t. So $f - f'$ must be identical to the zero-polynomial, since it's well known that any polynomial $g \in K[X]$ has at most degree(g) zeroes unless it's the zero-polynomial.

To set up Shamir's secret sharing scheme, let K be a finite field with $|K| > n$, where n is the number of agents. Let $P_1, \ldots, P_n$ be distinct, non-zero elements of K, and let these values serve as "names" for the n agents. Let $1 \leq t \leq n$ be the threshold. The secret-space in which the dealer codes the secret s is K. For each $s \in K$, define $\Pi(t, s)$ as the set of all polynomials $f(X) \in K[X]$ such that degree(f) $< t$ and $f(0) = s$.

One can efficiently sample a random member from $\Pi(t, s)$ by setting the lowest-order coefficient to s and taking random elements from K for the remaining $t - 1$ coefficients. [7]

The field K, the threshold t, the names $P_1, \ldots, P_n$ and the protocol below are known to all players. We assume that for each agent, there is a separate private communication channel with the dealer (for instance one based on public key encryption).

- *Distribution Phase*: The dealer has a secret $s \in K$, and selects a random polynomial $f(X) \in \Pi(t, s)$ and sends $s_i = f(P_i)$ as share in s privately to player P_i, $i = 1 \ldots n$.

[7] Note that this does not necessarily mean that one generates a polynomial of degree exactly $t - 1$, since $0 \in K$ is also in the play.

– *Reconstruction Phase*: From collection of $\geq t$ shares, the corresponding players pool their shares and jointly reconstruct $f(X)$ and compute $f(0) = s$.

By Lagrange interpolation it is clear that reconstruction works as desired.

As to *privacy*, consider an arbitrary subset V of the agents of size $t-1$. Write $V = \{P'_1, \ldots, P'_{t-1}\} \subset \{P_1, \ldots, P_n\}$, and write $s'_1, \ldots, s'_{t-1}$ for the shares $f(P'_1), \ldots, f(P'_{t-1})$ of V.

Observe that for each $s' \in K$, the t points $(0, s')$, $(P'_1, s'_1), \ldots, (P'_{t-1}, s'_{t-1})$ uniquely determine a polynomial $f'(X) \in \Pi(t, s')$ that passes through all of them.

So from the joint view of the players in V, each secret is equally likely (take into account that the dealer chose f at random, given s) and hence the shares held by V give no information about the real secret s.

Note that since the joint view of any set of size $t-1$ gives no information about the secret, the view of a *smaller* subset doesn't give information about the secret either. This follows from the fact that a smaller subset holds even less information.

11 Verifiable Secret Sharing

In the presence of participants carrying out malicious attacks, there are two threats in Shamir's scheme.

– The dealer may send inconsistent shares, i.e. not all of them are simultaneously on some polynomial of degree smaller than t.
– At reconstruction, players may contribute false shares so that $\tilde{s} \neq s$ is reconstructed or nothing at all.

Note that if the malicious players coordinate well, the honest players cannot in general distinguish between "good shares" and "bad shares". Therefore, the honest players may not even be able to figure out who the malicious players are.

In this section we explain methods to remedy this situation.

11.1 Definition of Malicious Adversary

Before we define *verifiable secret sharing* (VSS) to remedy these threats, we make the model more precise and introduce some terminology. Consider a dealer and n agents. A malicious adversary is allowed to *corrupt* the dealer and any single subset of the n agents of size smaller than t. All other players are *honest*. Later during the execution of a protocol [8] the adversary is allowed to alter and control the behaviour of the corrupted players at his will, and even have them behave

[8] In many multi-party computation protocols, the dealer will in fact at the same time also be one of the agents. In this case, there are in total n players involved, and the condition on the adversary is equivalent to saying that he is allowed to corrupt any single subset of size less than t of the n players, without distinguishing between dealer and agents.

in a coordinated fashion. In particular the adversary can make some corrupted players crash. For simplicity, we assume that the adversary makes the choice of which subset to corrupt before anything happened, i.e. before the start of a protocol.

11.2 Definition of VSS

The following informal definition is based on a formal definition of VSS from [50].

1. If the dealer is honest, then the distribution of a secret s always succeeds, and the corrupted players gain no information about s as a result of the distribution phase. At reconstruction, the honest players recover s. These properties hold regardless of the behaviour of the corrupted players.
2. If the dealer is corrupt, then the following holds. Either the dealer is deemed corrupt by the honest players, and all of them abort the distribution phase. [9] Else, the distribution phase is accepted by the honest players and *some* value s is uniquely fixed by the information held by the honest players as a result of the distribution phase. In the reconstruction phase, the honest players recover this value s. These properties hold regardless of the behaviour of the corrupted players.

Note the absence of a secrecy condition in the corrupt dealer case: if the set of corrupted players includes the dealer, the adversary controlling them knows the secret. Therefore, it is only required that in this case the protocol is robust. The honest dealer case of course corresponds to what one would naturally require.

11.3 VSS Scheme

Here is a sketch of a generic construction of VSS based on a combination of Shamir's secret sharing scheme, commitments and zero-knowledge interactive proofs. Let the threshold t for Shamir's scheme satisfy $t - 1 < n/2$.

Commitments We assume that we have a commitment scheme for committing to $\log |K|$ bits, for instance obtained as a parallel version of the commitments from Section 6.2 based on RSA or trapdoor one-way permutations.

From a high level, a commitment protocol based on such primitives works as follows. There is a public, efficiently computable function "commit" whose description follows from the primitive chosen, and it takes as input a random m-bit string (for some m that will be clear from the primitive) and some $\log |K|$-bit string.

To commit to a $\log |K|$-bit string x, one chooses a random m-bit string ρ and computes $C = \text{commit}(x, \rho)$. Finally, one publishes C as a commitment.

To open, one publishes x and ρ. The opening is verified by checking that $\text{commit}(x, \rho) = C$. We call the string ρ the opening information of the commitment to x.

[9] Another possibility is that the honest players take some default set of shares.

Broadcast We now also assume that a primitive called *broadcast* is at the disposal of the participants. This is a mechanism by means of which any of the participants can make sure that a message he has for all players is received unaltered by the honest players, despite the possible presence of malicious adversaries. Furthermore, we assume that recipients can establish who is the originator of the message. This mechanism may be realized by physical means or may be simulated by a protocol among the players. It suffices to know that it can be realized using digital signatures for instance.

VSS Protocol Here is an informal overview of a VSS protocol due to [54], which is also a nice illustration of the power of zero knowledge techniques and commitments. We assume that all players are restricted to probabilistic polynomial time computations.

- *Distribution Phase*: The dealer has secret $s \in K$, and computes shares $s_1, \ldots, s_n$ of s as in Shamir's scheme. For $i = 1 \ldots n$, he computes a commitment C_i to s_i. After he has broadcast the commitments to all players, he proves in zero knowledge to all players $P_1, \ldots, P_n$ that the commitments contain shares consistent with some secret. If this proof is accepted, he sends s_i and the opening information for C_i privately to P_i, $i = 1 \ldots n$.
- *Reconstruction Phase*: As in Shamir's scheme, except that each player P_i not only broadcasts his share s_i, but also the opening information for C_i. For reconstruction of the secret s, the honest players only take those shares whose corresponding commitment is opened successfully.

We briefly analyze this protocol. Regarding the zero knowledge proof of consistency, we assume that it proceeds in such a way that consistency holds if and only if the proof is accepted by all honest players (except with negligible error of course).

There is a number of ways to achieve this, for instance by having each player separately and publicly (using the broadcast primitive) act as a verifier in a zero knowledge proof by the dealer, while all others verify whether the proof is accepting. Only if the dealer at some point returns a proof that is not accepting, the honest players accuse the dealer and abort. Note that if consistency does not hold, then with high probability the proof when verified by an honest player will fail.

The actual consistency statement that the dealer has to prove, could take the following form: there exist $s, a_1, \ldots, a_{t-1} \in K$, $\rho_1, \ldots, \rho_n \in \{0,1\}^m$ such that for $i = 1 \ldots n$, $C_i = \text{commit}(s + \sum_{1 \leq j \leq t-1} a_j P_i^j, \rho_i)$.

Such statements can be proved in zero knowledge by the methods of [53], for instance. [10]

If the dealer is honest, the distribution phase definitely succeeds. Privacy follows from the hiding property of the commitments (the corrupted players are

[10] A particularly efficient general zero knowledge protocol is given in [29]

polynomially bounded and hence cannot read the contents of commitments), the privacy of Shamir's scheme, and the zero knowledge property of the proofs.

Looking at the reconstruction phase, and assuming that the distribution phase was successful, we note that in the case of corrupt shares, the commitments cannot be successfully opened since this would contradict the binding property. Therefore, false shares are always found out and can henceforth be ignored by the honest players. In summary, the only malicious action the corrupt players can undertake is to refuse to participate in the reconstruction phase. But since there are at most $t-1$ corrupt players and since we assumed that the threshold t in Shamir's scheme satisfies $t-1 < n/2$, there are always t honest players [11] to reconstruct the secret s.

11.4 Other Work

Particularly efficient VSS based on specific intractability assumptions (discrete logarithms) are presented in [42] and [67]. See also [28]. In a later section we discuss information theoretic VSS.

12 GMW: Achieving Robustness

With VSS in hand, the GMW protocol first has each player VSS each of his inputs before the n-player extension of protocol from Section 6.2 is executed (see Section 9). At the end of the protocol, each player applies VSS again, this time to the (additive) shares in the result of the computation. This requires additional zero knowledge proofs (in the same style as before) showing that these additive shares are indeed shared with VSS.

If one of the players fails in this phase (or earlier) he is kicked out of the computation, and the remaining players back up to the beginning, reconstruct the failed player's input, and do the protocol over again, this time simulating the failed player openly. Note that up to $t-1$ corrupted parties are tolerated in this way. With a similar argument as in the case of VSS, this can be shown to be optimal. There are more efficient variants, see [57] for a full description and analysis of the GMW-result.

The analysis [57] of the actual [12] GMW-protocol is very complex and has to deal with many subtleties that have been suppressed in our informal overview.

[11] This argument can also be used to show that $t-1 < n/2$ is optimal, i.e. it is not only sufficient but also necessary.

[12] We have made a number of simplifications for ease of exposition. For instance, we have neglected *input independence.*: in reality one must make sure that the corrupted parties choose their inputs to the computation independently from the inputs of the honest parties. This can be achieved by having all players commit to their inputs and having them give a zero knowledge proof of knowledge showing they can open these commitments.

13 Other Work

Chaum, Damgaard and van de Graaf [23] present protocols where one of the players' input is unconditionally protected. Kilian, Micali, and Ostrovsky [63] show how oblivious transfers can be used in zero knowledge protocols. Galil, Haber and Yung [48] achieve greater efficiency with their computation protocols. Recently, Gennaro, Rabin and Rabin [52] presented particularly efficient protocols for the cryptographic model (see also [28]).

Part III
Information Theoretic Security

14 Introduction

In 1987, two independent papers by M. Ben-Or, S. Goldwasser and A. Wigderson [13], and D. Chaum, C. Crépeau and I. Damgaard [24] achieved a new breakthrough in the theory of general multi-party computations.

They demonstrated the existence of *information theoretically secure* general multi-party computation protocols.

The price to be paid is a smaller tolerance with respect to the number of maliciously behaving players. Whereas [54] tolerates any malicious minority under the assumption that the players are computationally bounded, the protocols of [13] and [24] tolerate any malicious subset of size less than a third of the total number of players, with no assumptions on the computational power of the adversary. However, both papers argue that this is essentially the best one can achieve.

A common feature of both papers is the use of Shamir's secret sharing scheme, and the general paradigm of compiling a protocol secure against semi-honest players into one secure against malicious players by forcing all players to prove that they behave as semi-honest ones. However, [13] relies on techniques from the theory of error correcting codes, while [24] is based on *distributed* commitments and zero-knowledge. The result from [13] achieves perfect correctness, while [24] has a negligibly small error probability. [13]

14.1 Model

We make the model of BGW [13] and CCD [24] a bit more precise.

Communication:

[13] An interesting side-contribution of [24], seemingly often overlooked, is that it employs general zero knowledge techniques and information theoretically secure commitments in a distributed setting, showing that general zero knowledge makes sense (in a distributed setting) even if for instance $NP = P$.

– The n players are arranged in a complete (synchronous) network.
– Untappable private channels between each pair of players are available.

Adversary:

– The adversary is allowed to corrupt any single subset of size k of the players before the start of the protocol.
– Exercising complete control over the corrupted players, the adversary is allowed to force the corrupted players into coordinated malicious attack on the protocol.

Function:

– Any efficiently computable function g with n inputs.

14.2 Results of CCD and BGW

There is a **correct**, **private**, **robust** polynomial time protocol evaluating g iff the adversary corrupts at most $k < n/3$ players. In the semi-honest case this is $k < n/2$. These bounds are optimal. [14]

14.3 Remark on Broadcast

For security against malicious attacks, both results require the availability of a broadcast channel [64]. It is clearly not an option to use digital signatures in this case, since this does not fit with context of information theoretically secure protocols.

However, broadcast among n players can be efficiently simulated even in the presence of at most $t - 1 < n/3$ malicious players (see for instance [43,49]). This bound is optimal.

14.4 Outline of this Part

Instead of explaining the techniques of [24] and [13], we will sketch proofs of their results based on recent developments in the theory of multi-party computation due to Gennaro, Rabin and Rabin [52] and Cramer, Damgård and Maurer [28].

We first treat the semi-honest case.

15 Semi-Honest Case

We show how n players can securely compute on shared secrets. More concretely, n players have shares in two secrets (according to Shamir's secret sharing scheme) and they wish to compute from these, random shares in the sum or the product of these secrets.

We first treat these two cases. Later we show how this allows the n players to compute an arbitrary function on shared secrets.

[14] Given a broadcast channel for free, a malicious minority can also be tolerated by the result of T. Rabin and M. Ben-Or [70,69] at the expense of a negligble correctness error.

15.1 Computing on Shared Secrets

Constants, Addition Suppose there are n players holding shares of two secrets s and s', both resulting from Shamir's secret sharing scheme, with parameters n and t. It is easy to see that shares for the sum $s + s'$ are obtained when each player simply adds his shares of s and s'. Moreover, if the players later reconstruct $s + s'$ from these new shares, no new information about s, s' is given away beyond what can be deduced from their sum $s + s'$.

If the dealer used polynomials f and g to compute the shares of s and s' respectively, the new shares "look" as if the dealer had used the polynomial $(f + g)$ to compute shares of $s + s'$.

Similarly, for any constant c *known* to all players, they compute shares for $c \cdot s$ (respectively, $c + s$) by multiplying (respectively, adding) each share by c.

Multiplication We explain a method introduced by R. Gennaro, T. Rabin and M. Rabin [52]. Suppose there are n players holding shares of two secrets s and s', both resulting from Shamir's secret sharing scheme, with parameters n and t.

The goal of the players is to jointly compute on their shares such that as a result they hold shares in the product $s \cdot s'$, also resulting from Shamir's secret sharing scheme with parameters n and t. Moreover, they require that these shares for $s \cdot s'$ are randomly generated, as if the (honest) dealer had not only distributed shares for s and s', but independently for $s \cdot s'$ as well.

If we consider the joint view of any set of $t-1$ players, we can observe that this randomness condition has the following effect. If $s \cdot s'$ is later reconstructed from the n shares of $s \cdot s'$, the shares revealed together with the complete information held by the $t - 1$ players, do not give information beyond ss' and what can be inferred from it.

Unlike the case of addition, this is not trivial to solve. The first protocols for this task appeared in [24] and [13], but the solution of [52] we explain here is elegant and simple.

Let f and g denote the polynomials used by the dealer. Let n denote the number of players (agents) and t the threshold. We assume that $t - 1 < n/2$.

We have: $f(0) = s$ and $g(0) = s'$ and both polynomials are of degree less than t. For $i = 1 \ldots n$, write $s_i = f(P_i)$ and $s'_i = g(P_i)$ for player P_i's shares in secrets s and s', respectively.

We are interested in $s \cdot s'$. Observe that the polynomial $f \cdot g$ satisfies $(f \cdot g)(0) = s \cdot s'$ and that it has degree at most $2t - 2 < n$. Furthermore, for $i = 1 \ldots n$, $(f \cdot g)(P_i) = s_i \cdot s'_i$, which is a value that player P_i can compute on his own.

Therefore, by Lagrange interpolation and by our assumption $t-1 < n/2$, the players at least hold enough information to define $f \cdot g$ uniquely.

Now comes the interesting point. First, there exists a fixed linear combination, whose coefficients $r_1, \ldots, r_n \in K$ only depend on the P_i's, over the "product-shares" $s_i \cdot s'_i$ that yields $s \cdot s'$. This is easy to see. By the Lagrange

interpolation formula we have

$$(f \cdot g)(0) = \sum_{1 \le i \le n} \left(\frac{\prod_{1 \le j \le n, j \ne i} -P_j}{\prod_{1 \le j \le n, j \ne i} (P_i - P_j)} \right) \cdot s_i s'_i.$$

So the values between brackets are the coefficients $r_1, \ldots, r_n$, and all of these can be computed from public information. A simple but important fact for the analysis to follow is that at least t of these values are non-zero. For suppose without loss of generality that $r_t = \ldots = r_n = 0$. Then we have, for instance, $(f \cdot \mathbf{1})(0) = s = \sum_{1 \le i \le t-1} r_i(s_i \cdot 1)$, where $\mathbf{1}$ denotes the polynomial $1 \in K[X]$. This would mean that players $P_1, \ldots, P_{t-1}$ can break Shamir's secret sharing scheme, a contradiction.

Of course the players don't want to keep these product-shares $s_i s'_i$ as their shares in $s \cdot s'$; first of all it changes the threshold, and second, these product-shares are by no means random shares of the secret $s \cdot s'$. In fact, reconstruction could reveal more than just $s \cdot s'$ in the sense that it could also reveal information about s, s' individually.

Therefore, they first *re-share* their product-shares: each player P_i acts as a dealer and distributes shares of his secret $s_i \cdot s'_i$ to all players (P_i included for completeness), using the same parameters n and t and resulting in share u_{ij} for player P_j, $j = 1 \ldots n$. Write h_i for the polynomial used by P_i.

Consider the polynomial

$$h(X) = \sum_{1 \le i \le n} r_i \cdot h_i(X).$$

This has degree $< t$, and $h(0) = \sum_{1 \le i \le n} r_i \cdot h_i(0) = \sum_{1 \le i \le n} r_i \cdot s_i s'_i = s \cdot s'$.

Therefore, when each player P_i now computes

$$v_i = \sum_{1 \le j \le n} r_j u_{ji},$$

P_i has a share v_i in $s \cdot s'$ resulting from the polynomial $h(X)$ of degree less than t.

As to privacy, it is sufficient to note that from the point of view of any coalition of the players of size $t - 1$ or smaller, the polynomial h contains at least one h_i contributed by a player outside the coalition, since at least t of the $r_1, \ldots, r_n$ are non-zero.

15.2 Protocol for Semi-Honest Participants

Based on the techniques for computing on shared secrets, we now present a general multi-party computation protocol (essentially due to [52]) secure if a semi-honest adversary has access to the complete information of at most $t - 1$ players, where $t - 1 < n/2$.

We assume that the function they wish to jointly compute is given as an arithmetic circuit over a finite field K with $|K| > n$. Arithmetic circuits are

similar to Boolean circuits (see Section 4), except that the computations take place over K instead of $GF(2)$. This is no restriction: if we fix an arbitrary K, then any function that is efficiently computable is also computable by a polynomial size arithmetic circuit over K. These are the types of gates we require: two-input addition- and multiplication gates, and one-input gates for addition or multiplication with a constant.

As in Section 4, the computation proceeds in a gate-by-gate manner, maintaining the invariant that at each point the players have random shares in the current intermediate results.

When they have processed the final output gate, all players broadcast their shares in the result, and reconstruct it.

Input Distribution Phase
Using Shamir's Secret Sharing Scheme, each player provides shares of his input to all players.

Computation Phase
If the current gate is addition, or addition/multiplication of a constant, they follow the steps from the first part of Section 15.1. If the current gate is multiplication, they follow the steps from the second part Section 15.1.

Reconstruction Phase
Each player broadcasts his share in the output, and all reconstruct the result.

15.3 Optimality of the Bound

Suppose there exists an integer $n > 1$ and a general n-party computation protocol secure if more than a strict minority of the players conspire (semi-honestly), i.e. the number of tolerable conspirators would be at least $n/2$. This would immediately imply a protocol for two players to evaluate for instance the AND-function obliviously (each of the players would simulate a different half of the n players). By the same arguments as in Section 3.1, this is impossible and hence the $t - 1 < n/2$ bound is optimal.

16 Dealing with Malicious Attacks

We first show how to turn Shamir's secret sharing scheme into a Verifiable Secret Sharing Scheme. Based on this, we construct distributed homomorphic commitments. Finally, we explain how to defend against malicious attacks in general multi-party computations.

These results are taken from Cramer, Damgaard and Maurer [28].

16.1 Verifiable Secret Sharing Scheme

Below we adopt a linear algebraic view on Shamir's secret sharing scheme, that some may find less intuitive than the explanation based on polynomial interpolation (though technically speaking it is definitely as elementary).

Our reasons for doing so are two-fold.

First, it opens the way to a verifiable secret sharing scheme that avoids the bi-variate polynomials and error correcting codes of [13].

Second, Brickell [18] points out how this linear algebraic view leads to a natural extension to a wider class of secret sharing schemes that are not necessarily of the threshold type. This has later been generalized to all possible so-called monotone access structures [15] Karchmer and Wigderson [61] based on a linear algebraic computational device called *monotone span program.*

Cramer, Damgård and Maurer [28] extend these results of Karchmer and Wigderson, by introducing a method to transform monotone span program based secret sharing schemes (Shamir's scheme is a particular instance) into verifiable secret sharing schemes. The enhancement is purely linear algebraic in nature and admits no analogous view based on polynomials. In fact, in the monotone span program model of [61], which deals with arbitrary monotone access structures and not just threshold ones, it is in general not possible to speak about polynomials. Therefore, one reaches further if one concentrates on the quintessential algebraic properties, instead of on the very specific language of polynomials.

We will not present the general VSS result of [28] here, but rather the threshold-case which has some nice extras over the general construction, that are mentioned but not detailed in [28]. The presentation is self-contained and doesn't require knowledge of [61].

Linear Algebraic View on Shamir's Secret Sharing Let K be a finite field, let M be a matrix with n rows and t columns, and with entries from K. We say that M is an (n,t)-Vandermonde matrix (over K) if there are $\alpha_1,\ldots,\alpha_n \in K$, all distinct and non-zero, such that the i-th row of M is of the form $(1,\alpha_i,\ldots,\alpha_i^{t-1})$ for $i = 1\ldots n$. Note that this implies that $|K| > n$.

For an arbitrary matrix M over K with n rows labelled $1\ldots n$, and for an arbitrary non-empty subset A of $\{1,\ldots,n\}$, let M_A denote the matrix obtained by keeping only those rows i with $i \in A$. If $A = \{i\}$, we write M_i. Similarly, for a vector $\mathbf{s} \in K^n$, $\mathbf{s}_A$ denotes those coordinates s_i of $\mathbf{s}$ with $i \in A$.

Let M_A^T denote the transpose of M_A and let $\mathrm{Im}M_A^T$ denote the K-linear span of the rows of M_A. We use $\mathrm{Ker}M_A$ to denote all linear combinations of the columns of M_A leading to $\mathbf{0}$, the *kernel* of M_A.

It is well-known that any square (i.e. number of rows is equal to number of columns) Vandermonde matrix has a non-zero determinant. If M is an (n,t)-Vandermonde matrix over K and $A \subset \{1,\ldots,n\}$, then we conclude that the rank of M_A is maximal (i.e. is equal to t, or equivalently, $\mathrm{Im}M_A^T = K^t$) if and only if $|A| \geq t$.

[15] This generalization has first been achieved by Ito, Nishizeki and Saito [60] and later by Benaloh and Leichter [11]. Both these results are based on elementary monotone formula complexity of the access structure ([60] is more restricted since it requires DNF formulae). However, the model of [61] is much more powerful in terms of efficiency. See also [14].

But more is true. Let $\boldsymbol{\epsilon}$ denote the column vector $(1, 0, \ldots, 0) \in K^t$. If $|A| < t$, then $\boldsymbol{\epsilon} \notin \mathrm{Im}M_A^T$, i.e. there is no $\boldsymbol{\lambda} \in K^{|A|}$ such that $M_A^T\boldsymbol{\lambda} = \boldsymbol{\epsilon}$.

This can be seen as follows. Suppose without loss of generality that $|A| = t-1$ and that there is such $\boldsymbol{\lambda}$. Consider the square matrix N_A obtained from M_A by deleting the first column (that consists of $t-1$ 1's). This matrix is "almost" a square Vandermonde matrix: it can be seen as a square Vandermonde matrix multiplied by a matrix that has zeroes everywhere, except that its diagonal consists of non-zero elements (in fact, α_i's with $i \in A$). It follows that N_A has a non-zero determinant. But then $M_A^T\boldsymbol{\lambda} = \boldsymbol{\epsilon}$ implies $N_A^T\boldsymbol{\lambda} = \mathbf{0}$ and $\boldsymbol{\lambda} \neq \mathbf{0}$. This is impossible since N_A is a square matrix with non-zero determinant.

Therefore we can say

$$\boldsymbol{\epsilon} \in \mathrm{Im}M_A^T \text{ if and only if } |A| \geq t.$$

We need some more basic linear algebra. For vectors $\mathbf{x}, \mathbf{y} \in K^t$, define the standard in-product (finite field case) as $\langle \mathbf{x}, \mathbf{y} \rangle = x_0y_0 + \ldots + x_{t-1}y_{t-1}$. We write $\mathbf{x} \perp \mathbf{y}$ when $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ and $\mathbf{x}$ is said to be orthogonal to $\mathbf{y}$, and vice-versa. For a K-linear subspace V of K^t, $V^\perp$ denotes the collection of elements of K^t that are orthogonal to all of V (the *orthogonal complement*), which is again a K-linear subspace.

For all subspaces V of K^t we have $V = (V^\perp)^\perp$. This is an elementary fact that can be proved in a number of ways. Here we exploit the fact that K is finite.

Say $\dim(V) = t'$, and choose any basis for V. Now $\mathbf{x} \in V^\perp$ if and only if $\langle \mathbf{x}, \mathbf{f} \rangle = 0$ for all vectors $\mathbf{f}$ in the chosen basis. So if we arrange those basis vectors as the rows of a matrix M (it follows that $V = \mathrm{Im}M^T$), we have $V^\perp = (\mathrm{Im}M^T)^\perp = \mathrm{Ker}M$. The latter equality simply follows by inspection.

By Gaussian Elimination ("sweeping") on the rows of M, we can bring it of course into a form where the first t' columns constitute the identity matrix.

The rows of this new M are still a basis for V, and therefore the relationships above still hold. We count the number of $\mathbf{x}$ such that $M\mathbf{x} = \mathbf{0}$, i.e. we count $|\mathrm{Ker}M|$. From M's form, it follows that for each selection of the last $t - t'$ coordinates of $\mathbf{x}$, there is a unique selection of its first t coordinates such that $M\mathbf{x} = \mathbf{0}$. Hence, $|V^\perp| = |\mathrm{Ker}M| = |K|^{t-t'}$. Therefore, by applying this fact once more, $|(V^\perp)^\perp| = |K|^{t'}$. Since $V \subset (V^\perp)^\perp$ from the definition, it now follows that $V = (V^\perp)^\perp$.

By application of this fact, it now follows that $\mathrm{Im}M_A^T = (\mathrm{Ker}M_A)^\perp$, and we can conclude that

$$\boldsymbol{\epsilon} \notin \mathrm{Im}M_A^T \text{ if and only if there exists } \boldsymbol{\kappa} \in K^t \text{ such that } M_A\boldsymbol{\kappa} = \mathbf{0} \text{ and } \kappa_1 = 1.$$

Another simple identity is that $\langle \mathbf{x}, M_A^T\mathbf{y} \rangle = \langle M_A\mathbf{x}, \mathbf{y} \rangle$ for all $\mathbf{x}, \mathbf{y}$ of adequate dimensions.

Now we can present and analyze Shamir's scheme in an alternative fashion.

Let there be n players, and let t be the threshold. Over a finite field K, let M be an (n, t)-Vandermonde matrix.

Distribution Phase: Let $s \in K$ be the secret. The dealer chooses a vector $\mathbf{b} \in K^t$ by setting its first coordinate b_1 to s, and selecting random elements from

K for the remaining coordinates. To player i he privately sends $s_i = M_i\mathbf{b}$ as share in s, for $i = 1 \ldots n$.

Reconstruction Phase: Let $A \subset \{1, \ldots, n\}$ with $|A| \geq t$. From their joint information, the players in A efficiently compute by elementary linear algebra $\boldsymbol{\lambda} \in K^{|A|}$ such that $M_A^T\boldsymbol{\lambda} = \boldsymbol{\epsilon}$. Write $M\mathbf{b} = \mathbf{s}$. Then $s = \langle \mathbf{b}, \boldsymbol{\epsilon} \rangle = \langle \mathbf{b}, M_A^T\boldsymbol{\lambda} \rangle = \langle M_A\mathbf{b}, \boldsymbol{\lambda} \rangle = \langle \mathbf{s}_A, \boldsymbol{\lambda} \rangle$, which they can compute efficiently.

It should be clear that reconstruction works as desired. Regarding privacy, let $|A| = t - 1$, and consider the joint information held by the players in A, i.e. $\mathbf{s}_A = M_A\mathbf{b}$. Let $\tilde{s} \in K$ be arbitrary, and let $\boldsymbol{\kappa}$ be such that $M_A\boldsymbol{\kappa} = \mathbf{0}$ and $\kappa_1 = 1$. Then $M_A(\mathbf{b} + (\tilde{s} - s)\boldsymbol{\kappa}) = \mathbf{s}_A$ and the first coordinate of the argument is equal to $\tilde{s}$. This means that, from the point of view of the players in A, $\mathbf{s}_A$ can be consistent with the secret $\tilde{s}$.

The number of $\tilde{b} \in K^t$ with $\tilde{b}_1 = \tilde{s}$ is clearly equal to $|\mathrm{Ker}(M_A)|$ (which is independent of $\tilde{s}$), and the players in A have no information about s (take into account that all coordinates of $\mathbf{b}$ except possibly the first have been chosen at random).

Towards VSS Let $t - 1 < n/3$. A fact that is also exploited in [13] is that a complete set of shares $\mathbf{s}$ with at most $t-1$ arbitrary errors still defines the secret s uniquely.

Indeed, let $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2 \in K^n$ be arbitrary vectors with Hamming-weight at most $t - 1$. Let $W \subset \{1, \ldots, n\}$ denote the indices of the coordinates that are simultaneously zero in both vectors. Note that $|W| \geq t$. Consider $\mathbf{s}_1 = M\mathbf{b}_1$ and $\mathbf{s}_2 = M\mathbf{b}_2$, and $\tilde{\mathbf{s}_1} = \mathbf{s}_1 + \boldsymbol{\delta}_1$, and $\tilde{\mathbf{s}_2} = \mathbf{s}_2 + \boldsymbol{\delta}_2$. Suppose that $\tilde{\mathbf{s}}_1 = \tilde{\mathbf{s}}_2$. Then we have $M_W(\mathbf{b}_1 - \mathbf{b}_2) = \mathbf{0}$. But since $|W| \geq t$, the first coordinate of the argument must be zero and hence $b_1 = b_2$.

Therefore, in principle and assuming that the dealer is honest, setting $t - 1 < n/3$ guarantees robustness against players handing in false shares. However, efficiency is a problem (even when assuming an honest dealer): how to decode a "disturbed" set of shares $\tilde{\mathbf{s}}$ and recover the secret. In [13], efficient standard error correction techniques are applied to a version of Shamir's scheme obtained by first passing to an extension field of K.

We first explain how this can be avoided (for the moment we still assume an honest dealer).

Consider the following variant of Shamir's scheme.

Distribution Phase: Let $s \in K$ be the secret. The dealer chooses a random *symmetric* matrix $R \in K^{t,t}$, subject to the condition that it has s in its upper left corner. For $i = 1 \ldots n$, the dealer sends privately to player i the (row-)vector $\mathbf{s}_i = M_iR$ as share in s. Write $\mathbf{b}$ for the first column of R, then the first coordinate of $\mathbf{s}_i$ is equal to $M_i\mathbf{b}$. This value is called player i's *actual share* in s.

Reconstruction Phase: For $i = 1 \ldots n$, each player i broadcasts his share $\tilde{\mathbf{s}}_i$. Consider the matrix C with n rows and n columns, whose entry in the i-th row and j-th column is 1 if and only if $M_j\tilde{\mathbf{s}}_i^T = \tilde{\mathbf{s}}_jM_i^T$. Throw away all rows

of C that have t or more zeroes. There will be at least t rows left. For each of the rows i left, take the first coordinate of the corresponding $\tilde{\mathbf{s}}_i$ as the actual share of player i. These at least t actual shares determine uniquely the secret s, according to Shamir's secret sharing scheme as before.

We first argue that the secret s is indeed efficiently reconstructed, assuming an honest dealer and at most $t-1$ arbitrarily corrupt players. First note that for all i, $(M_i R)^T = R M_i^T$ by symmetry of R. Hence, for all i, j we have

$$M_j \mathbf{s}_i^T = \mathbf{s}_j M_i^T.$$

From this we conclude that each player j holds a share $M_j \mathbf{s}_i^T$ in player i's actual share of s. Consider the case that a player i broadcasts a vector $\tilde{\mathbf{s}}_i$ that differs from his share $\mathbf{s}_i$ in the first coordinate (and possibly elsewhere as well), then for at most $t-1$ of the real $\mathbf{s}_j$'s we have $M_j \tilde{\mathbf{s}}_i^T = \mathbf{s}_j M_i^T$, since obviously, two complete sets of shares in Shamir's scheme (with parameters n, t) for different secrets can agree on at most $t-1$ of the shares. But we also have to take into account that not only player i may be cheating, he may also coordinate with $t-2$ other cheaters. Hence, an upperbound on the number of consistencies in this case is $(t-1)+(t-1) = 2t-2$. Therefore, there are at least t inconsistencies in this case.

On the other hand, if player i is honest then no matter how the corrupt players lie and cheat, they are going to cause at most $t-1$ inconsistencies in the i-th row of the consistency matrix C. Therefore, the procedure yields at least t good actual shares, sufficient for reconstructing s. Note that in the analysis we have only used the fact that the total information $\mathbf{s}_B$ received by the set of the honest players B is of the form $\mathbf{s}_B = M_B R$ for *some* symmetric R.

As to *privacy* we note the following. For vectors $\mathbf{v} = (v_1, \ldots, v_t) \in K^t$ and $\mathbf{w} = (w_1, \ldots, w_t) \in K^t$, the standard tensor product (matrix form)$\mathbf{v} \otimes \mathbf{w}$ is defined as a matrix with t rows and t columns such that the j-th column is equal to $v_j \mathbf{w}$. Note that $\mathbf{v} \otimes \mathbf{v}$ is a symmetric matrix. Privacy is argued in a similar way as in the case of the linear algebraic explanation of Shamir's scheme. Let $|A| \leq t-1$, and let $\boldsymbol{\kappa}$ satisfy $M_A \boldsymbol{\kappa} = \mathbf{0}$ and $\kappa_1 = 1$. Then $\boldsymbol{\kappa} \otimes \boldsymbol{\kappa}$ is symmetric, has 1 in its upper left corner and satisfies $M_A(\boldsymbol{\kappa} \otimes \boldsymbol{\kappa}) = \mathbf{0}$. This is then used to show that for each possible secret, the number of symmetric matrices with that secret in its upper left corner and consistent with the joint information of A, is the same.

Pairwise Checking Protocol We now drop the assumption that the dealer is honest, and build a "pair-wise checking protocol", where each pair of players exchange checking information, around the scheme above to obtain VSS. The pair-wise checking as such is quite similar to methods from e.g. [13] and [43].

Let B denote the set of honest players, and let S_B be the total information received by B in the distribution phase. By the analysis of the honest dealer case above, we are done if $S_B = M_B R$ for *some* symmetric matrix R.

Suppose that some "pair-wise checking protocol" performed right after the dealer distributed the shares (as in the scheme above) would guarantee that

$$M_B S_B^T = S_B M_B^T.$$

We show that this is sufficient to conclude the existence of such R. Since certainly $|B| \geq t$, we know that the span of the rows of M_B is all of K^t. Hence there exists a matrix N_B such that $M_B^T N_B$ is equal to the identity matrix with t rows and t columns. Hence we have $M_B(S_B^T N_B) = S_B$, and we can take $S_B^T N_B$ as R.

The following pairwise-checking protocol is appended to the distribution phase.

1. Each player i sends to each player j the value $M_j \mathbf{s}_i^T$. Player j checks that this is equal to $\mathbf{s}_j M_i^T$ (pairwise consistency check). In case of an inconsistency, player j broadcasts a complaint about the value received from player i.
2. In response to complaints, the dealer must broadcast the correct value $M_j \mathbf{s}_i^T$ for all complaints of players j about the values received from players i.
3. If any player j finds that the information broadcast by the dealer is still inconsistent, it is clear to player j that the dealer is corrupt, and he broadcasts a request that the dealer makes public all the information sent to player j. This counts as claiming that the dealer is corrupt. These accusing players remain passive until a decision is made in the final step.
4. The dealer must again broadcast all the requested information, and again this may result in some players accusing the dealer of being corrupt. This can repeat until the information broadcast by the dealer contradicts itself, or he has been accused by at least t players. Or else, no new complaints occur and the number of accusations is at most $t - 1$. The decision whether or not to accept the distribution phase is now taken as follows. In the first two cases, the dealer is deemed corrupt and is disqualified. In the last case, the distribution phase is accepted by the honest players. Accusing players accept the information broadcast for them as their shares.

We analyze the protocol. First, we look at the honest dealer case. The corrupt players do not get more information than in the protocol above that assumes an honest dealer (note that no honest player will request the honest dealer to make public the information sent to him by the dealer, because if the honest player complains about some player, the honest dealer will always send the correct value).

Furthermore, the corrupt players can cause at most $t - 1$ accusations, and hence the distribution phase is always accepted by the honest players if the dealer is honest.

Next, let's drop the assumption that the dealer is honest and let's assume that the distribution phase was accepted by the honest players. Then it is immediate that each honest player has a share that is consistent with the shares of all other honest players. Suppose that this is not the case. There must be at least one honest player that did not accuse the dealer (since there are at most $t - 1$ accusations and at most $t - 1$ corrupted players, and $2t - 2 < n$ since $t - 1 <$

$n/3$). Clearly, the shares held by the set of non-accusing honest players (which is non-empty by the above) must be pair-wise consistent. All other shares of honest players are broadcast, so if there were any inconsistency, a non-accusing honest player would have accused the dealer, which is in contradiction with our assumptions.

16.2 General Protocol Secure against Malicious Attacks

Consider the protocol for the semi-honest case. We would like to enhance it so that the following invariant is maintained. At each point in the (once again) gate-by-gate multi-party computation, the current intermediate results (i.e. the values at the current gate as propagated through the circuit from the actual inputs) are secret shared (as in the semi-honest case) and moreover, each player is *committed* to his shares.

Homomorphic Distributed Commitments Distributed commitments have similar binding and hiding properties as the commitments from Section 6.2, except that this time these properties hold unconditionally, i.e. regardless of the computing power of an adversary. Of course, this will be so only with respect to the adversary we have defined earlier, that corrupts less than $n/3$ of the players before the start of the protocol.

Based on VSS, this is how it works. If player j wants to commit to $s \in K$, the n players execute the distribution phase of VSS, where player j acts as the dealer and takes s as the secret. To open the commitment, the n players execute the reconstruction phase of VSS.

One can immediately see that given two distributed commitments to values s and s' respectively, a commitment to $s+s'$ is non-interactively created by having all players locally take the sum of the information they hold (i.e. the VSS-shares in s and s').

Similarly, they can take a commitment and non-interactively multiply or add in a known constant.

It is in this sense that we say that the commitments are homomorphic. To create a distributed commitment to ss', is more involved and is explained later on.

Maintaining the Invariant Now think of the commitments from above as abstract, black-box homomorphic commitments, and forget for the moment how we actually constructed them. Suppose the dealer in Shamir's scheme first commits to the secret s and the random choices $\rho_1, \ldots, \rho_{t-1}$. Then, by the homomorphic properties of the commitments and the fact that the shares in Shamir's scheme are linear combinations (with fixed public coefficients!) of the secret and the ρ_i's, the players can compute new commitments to these n shares by just doing local computations. This guarantees that the dealer is committed to consistent shares, i.e. the shares results from a correct (not necessarily random, but this is no problem) execution of the distribution phase of the secret sharing scheme.

The only thing the dealer now has to do, is to send privately to each player the share he is entitled to *and* the "opening information" of the commitment to this share [16], so that now the receiving player is committed to his share and can open it himself. We call this *Commitment Sharing Protocol* (CSP)

In the *Input Distribution Phase* of the general protocol, all players will secret share their inputs to the computation in the way we have just described.

In the *Computation Phase*, if the current gate is addition, or multiplication by a constant, the procedure is trivial by the homomorphic properties of commitments and Shamir's secret sharing scheme. The only real difficulty left is handling multiplication gates, which we will study separately.

In the *Output Reconstruction Phase*, each player merely opens the commitment to his share in the final result of the computation. Each player collects enough correct shares to reconstruct the result (output) of the computation.

Linear Secret Sharing Schemes We now set out to handle the multiplication gates. But first it is convenient to further explore our linear algebraic view.

Shamir's secret sharing scheme is a linear scheme in the sense that each share is a linear combination (with fixed, public constants) of the secret and random choices made by the dealer.

It is possible to take this point of view as the starting point for a class of secret sharing schemes [18,14,61]: general linear secret sharing schemes.

There are n players, and there is a public matrix M with d rows and e columns, in which each row is assigned to one of the players. Abstractly speaking, each of the d rows of M is labeled with exactly one element from $\{1, \ldots, n\}$, and we allow that some (or all) labels occur more than once. Write ψ for the function that associates a row with a player. For $A \subset \{1, \ldots, n\}$, let M_A denote those rows of M that are labeled with an element from the set A. If $A = \{i\}$, we write M_i.

To compute shares of a secret, the dealer chooses a vector $\mathbf{b}$ at random subject to the condition that the secret is in the first coordinate of the vector, and for $i = 1 \ldots n$ sends the vector $\mathbf{s}_i = M_i\mathbf{b}$ as share in s privately to player i.

In Shamir's scheme this matrix corresponds of course to the Vandermonde matrix, and each player is associated with exactly one row.

Recall from the linear algebra proof of Shamir's scheme that exactly those subsets of the players can reconstruct the secret, whose matrix (i.e. the submatrix that contains the rows associated with the subset) has $\boldsymbol{\epsilon}$ in its K-linear span of the rows. Other subsets have no information about the secret.

It can be shown by similar arguments as the ones used in the linear algebra proof of Shamir's scheme, that in the general linear scheme as defined above, exactly those subsets A can reconstruct the secret for which $\boldsymbol{\epsilon}$ is in the K-linear span of the rows of M_A. Other subsets have no information.

[16] The opening information for a share consists basically of all data needed to construct the commitment. It's easy to see that the dealer in fact has the required information. In reality, the process we describe needs to be augmented with a complain/satisfy procedure, like in VSS. This procedure is fairly straightforward in this case.

Now in general, the subsets that can reconstruct are not exactly all subsets of a certain cardinality. One can show that for any *monotone access structure* Γ, i.e. a collection of subsets of the n players with the property that if A is a member of Γ than any set containing A is in Γ as well, there is a linear secret sharing scheme such that the subsets that can reconstruct the secret are exactly the members of Γ. Again, other subsets have no information.

Let $\boldsymbol{\epsilon}$ denote the vector $(1, 0, \ldots, 0) \in K^e$. It is not hard to show that the subsets A that can reconstruct the secret are exactly those for which $\boldsymbol{\epsilon} \in \mathrm{Im} M_A^T$. The players in such a set A jointly recover a secret s by computing $s = \langle \mathbf{s}_A, \boldsymbol{\lambda} \rangle$, where $M_A^T \boldsymbol{\lambda} = \boldsymbol{\epsilon}$, and $\mathbf{s}_A$ are the shares held by A, i.e. $\mathbf{s}_A = M_A \mathbf{b}$.

The quadruple $\mathcal{M} = (K, M, \boldsymbol{\epsilon}, \psi)$ is called *monotone span program* [61]. This powerful device is said to compute an access structure Γ (or equivalently, a monotone Boolean function) if and only if it is the case that $\boldsymbol{\epsilon} \in \mathrm{Im} M_A^T$ exactly when A is a member of Γ.

We will also call the sets in this correponding access structure the sets "accepted" by $\mathcal{M}$. A set that is not accepted, is called "rejected".

So each linear secret sharing scheme can be viewed as derived from a monotone span program computing its access structure.

We now return to the multiplication protocol. Let M be a (n, t)-Vandermonde matrix over K with $t - 1 < n/2$. For vectors $\mathbf{s}, \mathbf{s}' \in K^n$, define their *star-product*

$$\mathbf{s} * \mathbf{s}' = (s_1 s_1', \ldots, s_n s_n') \in K^n.$$

For vectors $\mathbf{x}, \mathbf{y} \in K^t$, define their tensor product (this time a vector instead of a matrix)

$$\mathbf{x} \otimes \mathbf{y} = (x_1 y_1, \ldots, x_1 y_t, \ldots, x_t y_1, \ldots, x_t y_t) \in K^{t^2}.$$

For a matrix M, let $M_\otimes$ denote M except that each row $\mathbf{v}$ of M is replaced by $\mathbf{v} \otimes \mathbf{v}$.

Another way to view the multiplication protocol from Section 15.1 for Shamir's scheme is by saying that there exists a fixed vector $\mathbf{r} \in K^n$, which we call *recombination vector*, such that for all $\mathbf{b}, \mathbf{b}' \in K^t$, with respective first coordinates $s, s' \in K$, we have

$$\langle \mathbf{r}, \mathbf{s} * \mathbf{s}' \rangle = ss',$$

where $\mathbf{s} = M\mathbf{b}$ and $\mathbf{s}' = M\mathbf{b}'$.

Call this the *multiplication-property* of the secret sharing scheme. The existence of the vector $\mathbf{r}$ follows for instance from the analysis in Section 15.1, as well as a method for efficiently computing it. From the analysis it also follows that Shamir's scheme has the multiplication property if and only if $t - 1 < n/2$.

In the case of defense against *malicious* attacks in the multiplication protocol for Shamir's scheme and for reasons to become clear shortly, we need additionally that for all $B \subset \{1, \ldots, n\}$ with [17] $|B| \geq n - t + 1$ there exists a fixed vector $\mathbf{r}$ (depending on B) such that

$$\langle \mathbf{r}, \mathbf{s}_B * \mathbf{s}_B' \rangle = ss',$$

[17] these sets of course correspond to the potentially honest sets rather than the potentially corrupt sets of size at most $t - 1$

where $\mathbf{s}_B = M_B\mathbf{b}$ and $\mathbf{s}'_B = M_B\mathbf{b}'$ are arbitrary.

Call this the *strong multiplication-property* of the secret sharing scheme, and call $\mathbf{r}$ the *recombination vector* for the set B.

Note that if the strong multiplication-property is satisfied, then certainly also the multiplication-property is satisfied: just take $B = \{1, \ldots, n\}$.

We can also say that strong multiplication is satisfied exactly when for each B with at least $n - t + 1$ elements, M_B has multiplication.

If we now set $t - 1 < n/3$, then we see that for all B with $n - t + 1$ elements, M_B is an $(n - t + 1, t)$-Vandermonde matrix ("t out-of $n - t + 1$") and also that $t - 1 < (n - t + 1)/2$. If B has even more elements, this clearly holds as well. Therefore, strong multiplication is satisfied by the way we set the parameter t.

It will be helpful to further extend the linear algebraic view. Note that the definition of the multiplication-property makes no reference to Shamir's secret sharing or threshold access structures. We could require this property of a general linear secret sharing scheme. In fact, this is exactly the definition of *monotone span programs with multiplication* from [28]. For strong multiplication, the only change in the definition we make is to say that the property holds for all sets B that are the complement of a set that is rejected by the monotone span program (i.e. complements of sets that are not in the access structure).

It is proved [18] in [28] that $\mathcal{M} = (K, M, \boldsymbol{\epsilon}, \psi)$ is a monotone span program with multiplication if and only if

$$\boldsymbol{\epsilon} \otimes \boldsymbol{\epsilon} \in \mathrm{Im} M_{\otimes}^{T}.$$

Any vector $\mathbf{r}$ with $\boldsymbol{\epsilon} \otimes \boldsymbol{\epsilon} = M_{\otimes}^{T}\mathbf{r}$ is a recombination vector.

As to strong multiplication, let $\mathcal{M}_B$ be the monotone span program obtained by throwing away the rows corresponding to the complement B of a rejected set. Then it follows immediately that $\mathcal{M}$ has strong multiplication if and only if for all such B, $\mathcal{M}_B$ has multiplication.

We are now ready to state the properties we use in the explanation of defense against malicious attacks to follow. Now let $\mathcal{M}$ be a monotone span program with multiplication. We can now consider the linear secret sharing scheme based on $\mathcal{M}_{\otimes} = (K, M_{\otimes}, \boldsymbol{\epsilon} \otimes \boldsymbol{\epsilon}, \psi)$ and conclude that the set $\{1, \ldots, n\}$ is accepted by $\mathcal{M}_{\otimes}$. Hence, if the n players receive a complete set of shares $M_{\otimes}\mathbf{c}$, they can recover the secret, which is $\mathbf{c}$'s first coordinate. This follows from the observations about the connection between general linear secret sharing and monotone span programs above.

If $\mathcal{M}$ has strong multiplication, this is true for each subset B, whose complement is rejected by $\mathcal{M}$. This fact and the following technicality (which is proved directly from the definitions) are useful in what follows.

For any monotone span program $\mathcal{M}$, and for all $\mathbf{b}$ and $\mathbf{b}'$, we have

$$\mathbf{s} * \mathbf{s}' = M_{\otimes}(\mathbf{b} \otimes \mathbf{b}'),$$

where $\mathbf{s} = M\mathbf{b}$ and $\mathbf{s}' = M\mathbf{b}'$.

[18] This follows from the definition and the uniqueness of algebraic normal form.

The Commitment Multiplication Protocol The situation is as follows. There are two values s and s', and each of the n players is committed to his shares in s and s'.

We'd like to have a protocol by means of which the same can be enforced on ss'.

Of course the protocol from Section 15.1 comes in handy, but we will have to enhance it.

Let $\mathcal{M} = (K, M, \boldsymbol{\epsilon}, \psi)$ be the monotone span program underlying Shamir's secret sharing scheme with $t - 1 < n/3$.

Consider player i right before he re-shares $s_i s'_i$ in Section 15.1, where s_i and s'_i are his shares in s and s', respectively. In the current context we may assume that he is already committed to s_i and s'_i separately.

It is sufficient for our purposes here if player i could create a commitment to $s_i s'_i$ and convince the rest of the players that this is indeed a commitment to $s_i s'_i$.

Indeed, suppose we had such a method, then for re-sharing we would do as in Section 15.1 and additionally have each player i commit to his local product $s_i s'_i$, prove that the resulting commitment contains indeed $s_i s'_i$, commit to randomness needed for the basic Shamir's secret sharing, have all players compute non-interactively commitments to the shares, and have player i finally send their shares privately, plus the information needed to open the commitments their respective shares, just as in the CSP-protocol. After each player i has done so, they can compute their own shares in ss', commitments to all shares and opening information for the commitments to their own shares, using the recombination vector $\mathbf{r}$.

How can player i prove that a given commitment contains the product of the contents of two other given commitments?

We assume that $t - 1 < n/3$. Let M be an (n, t)-Vandermonde matrix. Then $\mathcal{M} = (K, M, \boldsymbol{\epsilon}, \psi)$ is with strong multiplication and ψ just associates the j-th row of M with the j-th player, $j = 1 \ldots, n$.

First, player i selects $\mathbf{b}$ at random such that $b_1 = s_i$ and $\mathbf{b}'$ at random such that $b'_1 = s'_i$. Next, he commits to all random coefficients of $\mathbf{b}$ and $\mathbf{b}'$ (commitments to s_i and s'_i are already available, by assumption). Then all players compute, non-interactively, commitments to the individual shares resulting from $\mathbf{u} = M\mathbf{b}$ and $\mathbf{u}' = M\mathbf{b}'$. Finally, player i sends shares u_j and u'_j privately to player j, $j = 1 \ldots, n$. So this is as the CSP-protocol, except that at this point it is not necessary to provide the opening information of the respective commitments.

Player i proceeds by committing to $s_i s'_i$, and to each of the t^2 coordinates of $\mathbf{b} \otimes \mathbf{b}'$. The players now compute non-interactively commitments to the n coordinates $\mathbf{v} = (v_1, \ldots, v_n)$ of $M_{\otimes}(\mathbf{b} \otimes \mathbf{b}')$.

Note that if player i indeed committed to the correct value $s_i s'_i$, for each j we must now have $u_j u'_j = v_j$, since $\mathbf{u} * \mathbf{u}' = M_{\otimes}(\mathbf{b} \otimes \mathbf{b}')$.

In any case, there is a vector $\mathbf{c}$ such that $\mathbf{v} = M_{\otimes}\mathbf{c}$. Write $\mathbf{u} * \mathbf{u}' = \mathbf{w}$. Consider the set B, defined as the complement of the set of players that the adversary actually corrupted (i.e. B consists of the honest players). Note that $|B| \geq n - t + 1$.

Since $\mathbf{u} * \mathbf{u}' = M_{\otimes}(\mathbf{b} \otimes \mathbf{b}')$ and since $\mathcal{M}$ has strong multiplication, B is accepted by $\mathcal{M}_{\otimes}$ and the set of shares $\mathbf{w_B}$ for B defines $s_i s'_i$ uniquely. Likewise, $\mathbf{v}_B$ defines a secret (i.e. $\mathbf{c}$'s first coordinate c_1) uniquely.

Therefore, if $c_1 \neq s_i s'_i$, there must be a $j \in B$ such that player j holds different shares for c_1 and $s_i s'_i$: if not, the reconstruction procedure for B (in the secret sharing scheme derived from $\mathcal{M}_{\otimes}$) applied to $\mathbf{w}_B$ and $\mathbf{v}_B$ would yield the same secrets.

Therefore, if player i did not commit to $s_i s'_i$ there is at least one honest player j that will notice an inconsistency and is going to complain. Upon that complaint, player i must open the commitments to $u_j u'_j$ and v_j so that all honest players conclude that player i is corrupt.

On the other hand, if player i is honest, then there are at most $t-1$ such complaints from the corrupted players, and each of them will not convince any honest player, since opening the commitments will show that the complaining player is corrupt rather than player i. Moreover, the information that becomes available in the course of handling these complaints, does not yield any new information (from the point of view of the corrupted players) about $s_i s'_i$.

16.3 Extensions

The protocol above [19] and its analysis are a special case of [28]. In fact, the basic framework behind it also works for any adversary that can be captured [20] by a monotone span program with (strong) multiplication.

However a lot of things have to be settled first. The VSS protocol we described is an optimization for the threshold case of the general VSS scheme from [28]. That scheme is based on arbitrary monotone span programs and we cannot in general assume as in the threshold case here, that the matrix corresponding to the honest players has maximal rank (this is essential in the analysis of the threshold VSS). However, one can show that the protocol, although in general not a VSS, is still a distributed commitment scheme. Based on these commitments, one can indeed construct VSS based on arbitrary monotone span programs.

Moreover, [28] provides a theory of monotone span programs with (strong) multiplication that shows that exactly those general (not necessarily threshold) adversaries are captured for which [59] demonstrates that secure computation tolerating them is possible at all. Therefore, the theory is complete. Upper bounds on the complexity of monotone span programs with (strong) multipli-

[19] We have not tried to optimize its efficiency, and we have been not very explicit about how to handle situations where players are found out to be corrupt. In any case, it is always possible to back-up to the beginning, and recover the inputs of corrupted players, after which the protocol is done over again with the corrupted players openly being simulated. There are other options which we do not discuss here.

[20] Loosely speaking, this requires a monotone span program with (strong) multiplication that rejects the sets in the adversary structure: a pre-determined collection of subsets of the players, out of which the actual adversary may pick an element and corrupt all the players in it.

cation are given [21] as well, that show significant improvements over previous approaches (similarly for VSS, but not requiring multiplication properties).

A remark about broadcast is in place. In case of general adverarsaries, information theoretically secure broadcasts protocols defending against threshold adversaries are in general not sufficient. Therefore, [28] uses the result of [45].

Also, the techniques extend to the model of [70], where broadcast is assumed (and cannot be simulated information theoretically) and an exponentially small error is tolerated (see also [30]). This is non-trivial, and we omit any of the details.

17 Other Work

We provide some suggestions for further reading (besides those references already given). This list is by no means complete and selection has been quite ad-hoc (This holds as well for the results covered in detail in this paper, with the exception of the classical results).

Adaptive adversaries, i.e. adversaries who do not necessarily select their vicitms before the start of the protocol but rather adaptively as the protocol is proceeding, are dealt with in [8] [20].

In [2] it is shown how general multi-party computations can be performed with polynomial complexity and a constant number of rounds of interaction, provided that the function to be evaluated is given as a polynomial size arithmetic formula (instead of circuit). Efficiency considerations (also using pre-processing) are discussed in [5,6].

This issue of a corrupt majority is studied in [3].

Secure multi-party computation in an asynchronous communication model is addressed in [12].

Loosely speaking, a proactively secure protocol is one secure against an attacker who in principle can corrupt an arbitrary number of players in the life-time of a system, except that in each time-frame less than, say, half of the players are corrupted and a majority is honest [66,46].

For lots of references and detailed explanations of some fundamental results, see for instance [47] and [19].

Regarding multi-party computation protocols for electronic cash or electronic voting, see for instance [22], [26], [32] and [31].

Threshold cryptography, i.e. efficient and secure distributed computation for specific functions was introduced in [39]. See for instance, [38], [51] and [68] for distributed RSA-protocols.

[21] Recently, in a revision of [28], the authors have proved that for all relevant functions f (i.e. Q2-functions), if a monotone span program of size m is given that computes such a function f, then there exists a monotone span program *with multiplication* that computes f as well and has **size** $O(m)$. Note that the novelty is in the last part of the claim. This implies, in a well-defined sense, that *linear secret sharing is "sufficient" for general secure multi-party computation*, where both existence and efficiency are taken into account.

18 Acknowledgements

Donald Beaver, Claude Crépeau, Ivan Damgård, Serge Fehr, Matthias Fitzi, Martin Hirt, Ueli Maurer, Săsa Radomirović and Berry Schoenmakers are kindly acknowledged for discussions, anwering questions, giving comments or remarks.

References

1. W. Alexi, B. Chor, O. Goldreich and C.P. Schnorr: *RSA and Rabin functions: Certain parts are as hard as the whole*, SIAM Journal on Computing, 17(2):194-209, April 1988. 25
2. J. Bar-Ilan and D. Beaver: *Non-cryptographic fault-tolerant computing in constant number of rounds of interaction*. In Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, pages 201-209, Edmonton, Alberta, Canada, 14-16 August 1989. 58
3. D. Beaver and S. Goldwasser: *Multiparty computation with faulty majority (extended announcement)*, In 30th Annual Symposium on Foundations of Computer Science, pages 468-473, Research Triangle Park, North Carolina, 30 October-1 November 1989. IEEE 58
4. D. Beaver: *Foundations of Secure Interactive Computing*, Proceedings of Crypto 91, Springer Verlag LNCS, vol. 576, pp. 420-432, Springer-Verlag, 1992. 29
5. D. Beaver: *Secure Multiparty Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority*, J. Cryptology 4:2 (1991), 75-122. 58
6. D. Beaver: *Efficient Multiparty Protocols Using Circuit Randomization*, Proceedings of Crypto '91, Springer-Verlag LNCS, 1992, 420-432. 58
7. D. Beaver: *How to break a "secure" oblivious transfer protocol.*, Eurocrypt '92, volume 658 of Lecture Notes in Computer Science, pages 285-296. Springer-Verlag, 24-28 May 1992. 35
8. D. Beaver and S. Haber: *Cryptographic protocols provably secure against dynamic adversaries*, volume 658 of Lecture Notes in Computer Science, pages 307-323, Springer-Verlag, 24-28 May 1992. 58
9. D. Beaver: *Equivocable Oblivious Transfer*, Proceedings of Eurocrypt '96, Springer–Verlag LNCS 1070, 1996, 119–130. 31, 35
10. D. Beaver: *Adaptively Secure Oblivious Transfer*, to appear in the Proceedings of Asiacrypt '98. 31, 35
11. J. Benaloh, J. Leichter: *Generalized Secret Sharing and Monotone Functions*, Proc. of Crypto '88, Springer Verlag LNCS series, pp. 25–35. 47
12. M. Ben-Or, R. Canetti, O. Goldreich: *Asynchronous Secure Computations*, Proc. ACM STOC '93, pp. 52–61. 58
13. M. Ben-Or, S. Goldwasser, A. Wigderson: *Completeness theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proc. ACM STOC '88, pp. 1–10. 42, 43, 44, 47, 49, 50
14. M. Bertilsson, I. Ingemarsson: *A Construction of Practical Secret Sharing Schemes using Linear Block Codes*, Proc. AUSCRYPT '92, LNCS 718 (1993), 67-79. 47, 53
15. G. R. Blakley: *Safeguarding Cryptographic Keys*, Proceedings of AFIPS 1979 National Computer Conference, vol. 48, N.Y., 1979, pp. 313–317. 36
16. M. Blum: *Three Applications of the Oblivious Transfer*, Technical report, Dept. EECS, University of California, Berkeley, CA, 1981. 18

17. G. Brassard, C. Crépeau and M. Sántha: *Oblivious Transfers and Intersecting Codes*, IEEE Transaction on Information Theory , special issue on coding and complexity, Volume 42, Number 6, pp. 1769-1780, November 1996. 18, 19
18. E. F. Brickell: *Some Ideal Secret Sharing Schemes*, J. Combin. Maths. & Combin. Comp. 9 (1989), pp. 105–113. 47, 53
19. R. Canetti: *Studies in Secure Multiparty Computation and Applications*, Ph. D. thesis, Weizmann Institute of Science, 1995. 58
20. R. Canetti, U. Feige, O. Goldreich, M. Naor: *Adaptively Secure Multi-Party Computation*, Proc. ACM STOC '96, pp. 639–648. 58
21. R. Canetti: *Security and Composition of Multiparty Cryptographic Protocols*, draft, presented at the 1998 Weizmann Workshop on Cryptography, Weizmann Institute of Science, Rehovot, Israel, June 1998. 29
22. D. Chaum: *Achieving Electronic Privacy*, Scientific American, August 1992. 58
23. D. Chaum, I. Damgård and J. vd Graaf: *Multi-Party Computations Ensuring Secrecy of Each Party's Input and Correctness of the Output*, Proceedings of Crypto'87 volume 293 of Lecture Notes in Computer Science, pages 87-119, 16-20, Springer-Verlag, 1988. 42
24. D. Chaum, C. Crépeau, I. Damgård: *Multi-Party Unconditionally Secure Protocols*, Proc. of ACM STOC '88, pp. 11–19. 42, 43, 44
25. D. Chaum: *Transaction Systems to make Big Brother Obsolete*, Communications of the ACM, vol. 28, no. 10, October 1985, pp. 1030–1044.
26. D. Chaum: *Untraceable Electroic Mail, Return Addresses, and Digital Pseudonyms*, Communications of the ACM, vol. 24, no. 2, 1985, pp. 84–88. 58
27. B. Chor, S. Goldwasser, S. Micali, B. Awerbuch: *Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults*, Proc. IEEE FOCS '85, pp. 383–395. 36
28. R. Cramer, I. Damgård and U. Maurer: *Span Programs and Secure Multi-Party Computation*, draft, presented at the 1998 Weizmann Workshop on Cryptography, Weizmann Institute of Science, Rehovot, Israel, June 1998. Completely revised version available from `http://www.inf.ethz.ch/personal/cramer`. 41, 42, 43, 46, 47, 55, 57, 58
29. R. Cramer, I. Damgård: *Zero Knowledge for Finite Field Arithmetic or: Can Zero Knowledge be for Free?*, Proc. of CRYPTO'98, Springer Verlag LNCS series. 40
30. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt and T. Rabin: *Efficient Multi-Party Computations with Dishonest Minority*, Proceedings of Eurocrypt '99, Springer Verlag LNCS. To appear. 58
31. R. Cramer, R. Gennaro and B. Schoenmakers : *A Secure and Optimally Efficient Multi-Authority Election Scheme*, Proceedings of EUROCRYPT '97, Konstanz, Germany, Springer Verlag LNCS, vol. 1233, pp. 103–118, May 1997. Journal version: Eur. Trans. Telecom, Vol. 8, No. 5, Sept./Oct. 1997.
32. R. Cramer, M. Franklin, B. Schoenmakers, M. Yung: *Secure Secret Ballot Election Schemes with Linear Work*, Proceedings of EUROCRYPT '96, Zaragoza, Spain, Springer Verlag LNCS, vol. 1070, pp. 72–83, May 1996. 58 58
33. C. Crépeau: *Equivalence between two flavours of oblivious transfers (abstract)*, Proceedings of Crypto '87 , volume 293 of Lecture Notes in Computer Science , pages 350-354. Springer-Verlag, 1988. 20
34. C. Crépeau: *Correct and Private Reductions among Oblivious Transfers* PhD thesis, Department of Elec. Eng. and Computer Science, Massachusetts Institute of Technology, 1990. 35

35. C. Crépeau, J.vd.Graaf and A. Tapp: *Committed Oblivious Transfer and Private Multiparty Computation*, proc. of Crypto 95, Springer Verlag LNCS series. 35
36. C. Crépeau and J. Kilian: *Achieving oblivious transfer using weakened security assumptions*, In 29th Symp. on Found. of Computer Sci. , pages 42-52. IEEE, 1988. 22
37. C. Crépeau and L. Salvail: *Oblivious Verification of Common String*, CWI Quarterly (Special Issue on Cryptography), 8 (2), June 1995. 20
38. A. De Santis, Y. Frankel, Y. Desmedt and M. Yung: *How to Share a Function Securely*, Proceedings of 26th Annual ACM STOC, pp. 522–522, 1994. 58
39. Y. Desmedt: *Threshold Cryptography*, European Transactions in Telecommunication, 5 (1994), 449–457. 58
40. S. Even, O. Goldreich and A. Lempel: *A Randomized Protocol for Signing Contracts*, Communications of the ACM, vol. 28, 1985, pp. 637–647. 18
41. R. Fagin, M. Naor and P. Winkler: *Comparing Common Secret Information without Leaking it*, Communications of the ACM, vol 39, May 1996, pp. 77–85. 19, 20, 31
42. P. Feldman: *A practical scheme for non-interactive verifiable secret sharing*, Proceedings of 28th Annual Symposium on Foundations of Computer Science, pages 427-437, Los Angeles, California, 12-14 October 1987. IEEE. 41
43. P. Feldman, S. Micali: *An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement*, SIAM J. Comp. Vol. 26, No. 4, pp. 873–933, August 1997. 43, 50
44. M. Fischer, S. Micali and C. Rackoff: *A Secure Protocol for Oblivious Transfer (extended abstract)*, presented at Eurocrypt '84. First published in Journal of Cryptology, 9(3):191–195, Summer 1996. 30
45. M. Fitzi and U. Maurer: *Efficient Byzantine Agreement Secure Against General Adversaries*, Proceedings of 12th International Symposium on Distributed Computing (DISC '98). 58
46. Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung: *Optimal-resilience proactive public-key cryptosystems*, Proceedings of 38th Annual Symposium IEEE FOCS pages 384-393, 1997. 58
47. M. Franklin: *Complexity and Security of Distributed Protocols*, Ph.D. thesis, Columbia University, New York, 1992. 58
48. Z. Galil, S. Haber and M. Yung: *Cryptographic computation: Secure fault-tolerant protocols and the public-key model*, Proceedings of Crypto '87, volume 293 of Lecture Notes in Computer Science, pages 135-155, 16-20 August 1987. Springer-Verlag, 1988. 42
49. J.A. Garay and Y. Moses: *Fully polynomial Byzantine agreement for n ¿ 3t processors in t + 1 rounds*, SIAM Journal on Computing, 27(1):247-290, February 1998.
50. R. Gennaro: *Theory and Practice of Verifiable Secret Sharing*, Ph.D.-thesis, MIT, 1995. 43 39
51. R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin: *Robust and efficient sharing of RSA functions*, Proceedings of CRYPTO '96, volume 1109 of Lecture Notes in Computer Science, pages 157-172, 18-22, 1996. 58
52. R. Gennaro, M. Rabin, T. Rabin, *Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography*, Proceedings of ACM PODC'98. 42, 43, 44, 45
53. O. Goldreich, S. Micali and A. Wigderson: *Proofs that Yield Nothing but the Validity of the Assertion, and a Methodology of Cryptographic Protocol Design*, Proceedings IEEE FOCS'86, pp. 174–187. 32, 33, 40

54. O. Goldreich, S. Micali and A. Wigderson: *How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority*, Proc. of ACM STOC '87, pp. 218–229. 22, 26, 31, 32, 34, 36, 40, 42
55. O. Goldreich and R. Vainish: *How to Solve any Protocol Problem: An Efficiency Improvement*, Proceedings of Crypto'87, volume 293 of Lecture Notes in Computer Science, pages 73-86, 16-20 August 1987. 26
56. O. Goldreich: *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, ISBN 3-540-64766-x, Springer-Verlag, Algorithms and Combinatorics, Vol. 17, 1998. 32
57. O. Goldreich: *Secure Multi-Party Computation* (working draft), Weizman Institute of Science, Rehovot, Israel, June 1998. Avaliable through the author's homepage `http://theory.lcs.mit.edu/ oded/`. 29, 41
58. S. Goldwasser, S. Micali and C. Rackoff: *The Knowledge Complexity of Interactive Proof Systems*, Proceedings of ACM STOC'85, pp. 291–304.
59. M. Hirt, U. Maurer: *Complete Characterization of Adversaries Tolerable in General Multiparty Computations*, Proc. ACM PODC'97, pp. 25–34. 57
60. M. Ito, A. Saito and T. Nishizeki: *Secret Sharing Scheme Realizing General Access Structures*, Proceedings IEEE Globecom '87, pp. 99–102, 1987. 47
61. M. Karchmer, A. Wigderson: *On Span Programs*, Proc. of Structure in Complexity, 1993. 47, 53, 54
62. J. Kilian: *Founding Cryptography on Oblivious Transfer*, ACM STOC '88, pp. 20–31. 34
63. J. Kilian, S. Micali and R. Ostrovsky: *Minimum resource zero-knowledge proofs (extended abstract)*, Proceedings of 30th Annual IEEE Symposium on Foundations of Computer Science, pages 474-479, November 1989, IEEE. 42
64. L. Lamport, R.E. Shostak and M.C. Pease: *The Byzantine generals problem*, ACM Transactions on Programming Languages and Systems, 4(3):382-401, July 1982. 43
65. S. Micali and P. Rogaway:*Secure Computation*, Manuscript, Preliminary version in Proceedings of Crypto 91. 29
66. R. Ostrovsky and M. Yung: *How to withstand mobile virus attacks*, Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing, pages 51-59, 1991 58
67. T. P. Pedersen: *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*, Proc. CRYPTO '91, Springer Verlag LNCS, vol. 576, pp. 129–140. 41
68. T. Rabin: *A Simplified Approach to Threshold and Proactive RSA*, Proceedings of Crypto '98, Springer Verlag LNCS, vol. 1462, pp. 89–104, 1998. 58
69. T. Rabin: *Robust Sharing of Secrets when the Dealer is Honest or Cheating*, J. ACM, 41(6):1089-1109, November 1994. 43
70. T. Rabin, M. Ben-Or: *Verifiable Secret Sharing and Multiparty Protocols with Honest majority*, Proc. ACM STOC '89, pp. 73–85. 43, 58
71. M. Rabin: How to Exchange Secrets by Oblivious Transfer, Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981. 18, 23
72. R. Rivest, A. Shamir and L. Adleman: *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of ACM, 21 (1978), pp. 120–126. 25
73. A. Shamir: *How to Share a Secret*, Communications of the ACM 22 (1979) 612–613. 36
74. S. Wiesner: *Conjugate Coding*, SIGACT News, vol. 15, no. 1, 1983, pp. 78-88; Manuscript written *circa* 1970, unpublished until it appeared in SIGACT News. 18
75. A. Yao: *Protocols for Secure Computation*, Proc. IEEE FOCS '82, pp. 160–164. 26

Commitment Schemes and Zero-Knowledge Protocols

Ivan Damgård

Aarhus University, BRICS (Basic Research in Computer Science, center of the Danish National Research Foundation)
ivan@daimi.aau.dk

Abstract. This article is an introduction to two fundamental primitives in cryptographic protocol theory: commitment schemes and zero-knowledge protocols, and a survey of some new and old results on their existence and the connection between them.

1 What's in this Article?

This article contains an introduction to two fundamental primitives in cryptographic protocol theory: commitment schemes and zero-knowledge protocols. We also survey some new and old results on their existence and the connection between them. Finally, some open research problems are pointed out.

Each of the two main sections (on commitments, resp. zero-knowledge) contain its own introduction. These can be read independently of each other. But you are well advised to study the technical sections on commitment schemes before going beyond the introduction to zero-knowledge.

The reader is assumed to have some basic knowledge about cryptography and mathematics, in particular public key cryptography and the algebra underlying the RSA and discrete log based public key systems. Concepts such as groups and homomorphisms will be used without further introduction.

2 Commitment Schemes

2.1 Introduction

The notion of *commitment* is at the heart of almost any construction of modern cryptographic protocols. In this context, making a commitment simply means that a player in a protocol is able to choose a value from some (finite) set and commit to his choice such that he can no longer change his mind. He does not however, have to reveal his choice - although he may choose to do so at some later time.

As an informal example, consider the following game between two players P and V:

1. P wants to commit to a bit b. To do so, he writes down b on a piece of paper, puts it in a box, and locks it using a padlock.

I. Damgård (Ed.): Lectures on Data Security, LNCS 1561, pp. 63–86, 1999.

2. P gives the box to V
3. If P wants to, he can later *open* the commitment by giving V the key to the padlock.

There are two basic properties of this game, which are essential to any commitment scheme:

- Having given away the box, P cannot anymore change what is inside. Hence, when the box is opened, we know that what is revealed really was the choice that P committed to originally. This is usually called the *binding property.*
- When V receives the box, he cannot tell what is inside before P decides to give him the key. This is usually called the *hiding property*

There are many ways of realizing this basic functionality, some are based on physical processes, e.g. noisy channels or quantum mechanics, while others are based on distributing information between many players connected by a network. We will say a bit more about this later, but for now we will concentrate on the scenario that seems to be the most obvious one for computer communication: commitments that can be realized using digital communication between two players.

As a very simple example of this kind of commitments, consider the case where P has a pair of RSA keys, where V (like anyone else) knows the public key with modulus n and public exponent e. To commit to a bit b, P can build a number x_b, which is randomly chosen modulo n, such that its least significant bit is b. Then he sends the encryption $C = x_b^e \bmod n$ to V. We do not prove anything formal about this scheme here, although that is in fact possible. But it should be intuitively clear that P is stuck with his choice of b since the encryption C determines all of x_b uniquely, and that V will have a hard time figuring out what b is, if he cannot break RSA. Thus, at least intuitively, the binding and hiding requirements are satisfied.

Why should we be interested in building such commitment schemes? Primarily because this simple functionality enables the construction of secure protocols that accomplish surprisingly complicated, even seemingly impossible tasks. We will see some examples of this in the section on zero-knowledge. But we can already now give an example of a simple problem that seems intractable without commitment schemes, namely *coin flipping by telephone.*

The following story was introduced by Manuel Blum: suppose our old friends Alice and Bob are getting a divorce. They are at the point where they cannot even stand facing each other, so they have to discuss over the phone how to split the furniture, the kids, etc. But one problem remains: *who gets the car?* Since they cannot agree, they decide to flip a coin. However, they quickly realize that this is easier said than done in this situation where of course they don't trust each other at all. Bob would not be very happy about a protocol where he announces $HEADS$, only to hear Alice reply on the phone: "Here goes...I'm flipping the coin....You lost!". How can we solve this? Well, certainly not by asking Alice to flip the coin and announce the result to Bob before he has chosen heads or tails; Alice would be just as unhappy as Bob was before. We seem to be in a

deadlock - neither party wants to go first in announcing their choice. However, this deadlock can be broken: using a commitment scheme, we get the following simple protocol:

1. Alice commits to a random bit b_A, and sends the resulting commitment C to Bob (you can think of C as being a locked box or an encryption, as you wish).
2. Bob chooses a random bit b_B and sends it to Alice.
3. Alice opens C to let Bob learn b_A, and both parties compute the result, which is $b = b_A \oplus b_B$.

It is not hard to argue intuitively that if the commitment is binding and hiding, then if at least one of Alice and Bob play honestly and chooses a bit randomly, then the result is random, no matter how the other party plays. A formal proof requires a more precise definition of commitments, which we will get to in the next section.

2.2 Defining Commitment Schemes

Two things are essential in the RSA example:

- The RSA key used does not come falling from the sky. There has to be an algorithm for generating it: some procedure that takes as input the length of modulus to generate, and then chooses randomly n and e, suitable for use as an RSA public key. In the example this algorithm would be run by P initially, and P must have some confidence that keys generated by this algorithm cannot be easily broken by V.
- When committing, it is essential that P makes random choices. The scheme above (in fact any scheme) would be completely insecure, if this was not the case (can you see why?). Thus the commitment sent to V must be a function of both the bit committed to, and of some random choices made by P.

Keeping this in mind, we can abstract the following general definition. It is somewhat simplified in that it does not cover all commitment schemes, but it covers the examples we will look at, and is enough to get a feeling for how such definitions work.

We will think of a commitment scheme as being defined by a a probabilistic polynomial time algorithm $\mathcal{G}$ called a *generator*. It takes as input 1^l where l is a security parameter and corresponds to e.g. the length of RSA modulus we want. It outputs a description of a function $\texttt{commit} : \{0,1\}^l \times \{0,1\} \to \{0,1\}^l$. where the idea is that a 0/1-values can be committed to. We stick to *bit*-commitments here for simplicity. We refer to the description of `commit` as the *public key* of the commitment scheme.

To use the scheme in practice, one first executes a *set-up phase* (once and for all) where either P or V runs $\mathcal{G}$, and sends a description of the resulting function `commit` to the other party. In some schemes it is necessary in addition to convince the other party that `commit` was correctly chosen, in case this is not

easy to verify from the description itself. Thus, one of the parties may *reject* in the set-up phase, meaning that it refuses to use the public key it received.

Assuming that the public key was accepted, to commit to a bit b, P chooses r at random from $\{0,1\}^l$ and computes the commitment $C \leftarrow \texttt{commit}(r,b)$. To open a commitment, r, b are revealed, and V checks that indeed $C = \texttt{commit}(r,b)$.

To define precisely the two essential properties of hiding and binding for this kind of commitment, we need to first define what it means for an entity, typically a probability, to be negligible – "too small to matter". There are different ways in which one can define what negligible means, from the point of view of a practical application, one might want to say that anything occurring with probability below a concrete bound, such as 2^{-50}, is negligible. In complexity based cryptography, one usually prefers an asymptotic definition: $\epsilon(l)$ is *negligible in* l if for any polynomial p, $\epsilon(l) \leq 1/p(l)$ for all large enough l. One motivation for this is that if we perform repeatedly an experiment in which a particular event occurs with negligible probability in l, then the expected number of repetitions before seeing an occurrence is superpolynomial in l. In this sense we can say that events that occur with negligible probability occur so seldom that polynomial time algorithms will never see them happening. We then have the following definitions:

- The binding property comes in two flavors.
  ***unconditional binding*:** Means that even with infinite computing power, P cannot change his mind after committing. In this case, P will run the generator and send the function `commit` to V. We require that if `commit` is correctly generated, then b is uniquely determined from $\texttt{commit}(r,b)$, and that an honest V accepts an incorrectly generated `commit` with at most negligible probability.
  ***computational binding*:** Means that unless you have "very large" computing resources, then you chances of being able to change your mind are very small. In this case, V will run the generator, so we can define it precisely as follows: take any probabilistic polynomial time algorithm P^* which takes as input a public key produced by the generator $\mathcal{G}$ on input 1^l. Let $\epsilon(l)$ be the probability (over the random choices of $\mathcal{G}$ and P^*) with which the algorithm outputs a commitment and two valid openings revealing distinct values. That is, it outputs C, b, r, b', r' such that $b \neq b'$ and $\texttt{commit}(r,b) = C = \texttt{commit}(r',b')$. Then $\epsilon(l)$ is *negligible in* l.
- The hiding property also comes in two flavors:
  ***unconditional hiding*:** Means that a commitment to b reveals (almost) no information about b, even to an infinitely powerful V. In this case V will run the generator and send the function `commit` to P. We require that if `commit` is correctly generated, then the distributions of $\texttt{commit}(r,0)$ and $\texttt{commit}(s,1)$ for random r, s are almost the same, meaning that one can be changed into the other by moving a negligible amount of probability mass. Furthermore an honest P should accept an incorrectly generated `commit` with at most negligible probability. In the best possible case, a commitment $\texttt{commit}(r,b)$ has distribution independent of b, and P never

accepts a bad `commit` function, i.e. commitments reveal no information whatsoever about the committed values. We then speak of *perfectly* hiding commitments.

***computational hiding*:** Means that a polynomially bounded V will have a hard time guessing what is inside a commitment. In this case, P will run the generator. A precise definition: consider any probabilistic polynomial time algorithm which takes as input a public key produced by the G on input 1^l, and a commitment $\texttt{commit}(r, b)$, where b is 0 or 1, and outputs a bit. Let $\epsilon_b(l)$ be the probability that 0 is produced as output when the commiment contained b. Then $|\epsilon_0(l) - \epsilon_1(l)|$ is negligible in l. This definition says that an adversary will not be able to tell efficiently which of the two given values is in a commitment, with probability much better than just guessing at random.

Before we continue, a word of warning about the definitions of the computational flavors of hiding and binding: They are based on the asymptotic behavior of an adversary as we increase the value of the security parameter. This is mathematically convenient when doing proofs, and has nice connections to standard complexity theory - but one should take care when evaluating the meaning in practice of results according to such a definition: it implicitly classifies everything that can be solved in probabilistic polynomial time as being "easy" and anything else as being "hard", and this distinction is not always accurate in real life. Even if a problem (such as breaking a commitment scheme) is asymptotically hard, it might still be easy in practice for those input sizes we want in a particular application. This does not at all mean that asymptotic security results are worthless, only that usage of a scheme in real life should always be supplemented with an analysis of practical state of the art of solutions to the (supposedly) hard problem we base ourselves on.

In any case, it is evident that the computational versions of the properties are more complicated to define than the unconditional ones. And since furthermore an unconditional guarantee is of course better, why don't we just build commitments that are both unconditionally binding *and* hiding? Well, unfortunately this is impossible!

Imagine we had such a scheme. Then, when P sends a commitment to e.g. 0 $C = \texttt{commit}(r, 0)$, there must exist an r', such that $C = \texttt{commit}(r', 1)$. If not, V could conclude that the committed value could not be 1, violating unconditional hiding. But then, if P has unlimited computing power, he can find r' and change his mind from 0 to 1, violating unconditional binding. This reasoning does not depend on the particular definition we have presented of commitment schemes. It extends to any protocol whatsoever for committing to a value in a two-player game. The basic reason for this is that the scenario by definition ensures that each player sees everything the other player sends.

There are several scenarios, however, where this reasoning does not apply. In a distributed scenario with many players, or in a two-party case where communication is noisy, it is no longer true that V sees exactly what P sends and vice versa. And in such cases, unconditional binding and hiding can in fact be obtained si-

multaneously. For commitment schemes in such scenarios, see e.g. [10, 3, 14, 9]. Note, however, that despite the fact that the reasoning does not apply to quantum communication either, bit commitment with unconditional security is not possible with quantum communication alone. More about his can be found in the article by Salvail in this volume.

2.3 Examples of Commitment Schemes

Many examples of commitment schemes have been suggested in the literature, see e.g. [4] for some basic ones or [11] for some later and more efficient examples.

Above, we have seen an example based on RSA with unconditional binding. This scheme also satisfies computational hiding, assuming that the RSA encryption function is hard to invert, although this is quite technically complicated to prove. It does *not* follow immediately, since a priori it might well be the case that the least significant bit of x is easy to compute from $x^e \bmod n$, even though all of x is hard to find. However in [1] it was proved that this is not the case: any algorithm that guesses the least siginificant bit of x with probability slightly better than $1/2$ can, by a randomized polynomial time reduction, be turned into one that inverts the RSA encryption function.

We now look at a general way to make commitment schemes with unconditional hiding. It turns out that such schemes can be constructed if we can efficiently generate group homomorphisms that are 1-way functions, i.e. they are easy to compute but hard to invert. A precise definition:

Definition 1 *A* Group Homomorphism Generator $\mathcal{H}$ *is a probabilistic polynomial time algorithm which on input* 1^l *outputs a description of two finite Abelian groups* G, H *and a homomorphism* $f : H \to G$*. Elements in* G, H *can be represented as* l*-bit strings, and the group operation and inverses in* G *and* H *can be computed in polynomial time. Finally, a uniformly chosen element in* H *can be selected in probabilistic polynomial time.*

$\mathcal{H}$ *is said to be* one-way *if in addition the following holds for any probabilistic polynomial time algorithm* A*: on input* f, y*, where* f *is selected by* $\mathcal{H}$ *on input* 1^l *and* y *is uniformly chosen in* $Im(f)$*, the probability that* A *outputs* $x \in H$ *such that* $f(x) = y$ *is negligible.*

As an example, consider any algorithm for generating a secure RSA modulus n. We can extend this to a homomorphism generator by choosing also a prime $q > n$, letting $G = H = Z_n^*$, the multiplicative group modulo n, and finally defining $f(x) = x^q \bmod n$. Assuming that RSA with modulus n and public exponent q is hard to invert, this clearly satisfies the requirements (recall that in general, f is a homomorphism, if $f(1) = 1$ and $f(xy) = f(x)f(y)$). Note that q, being larger than n, must be prime to $\phi(n)$ and so one can directly check from a description of f that it is surjective.

We can also base a generator on the discrete log problem. In this case, f would be of the form $f(x) = g^x \bmod p$ for a large prime p. We leave the details to the reader.

Given a generator as defined above, we define an unconditionally hiding scheme as follows. We assume for simplicity that given y, one can directly check if $y \in Im(f)$. This is trivially true of the RSA implementation above.

- **Set-up Phase**: the generator $\mathcal{G}$ for the commitment scheme is defined based on the group homorphism generator $\mathcal{H}$ as follows: it runs $\mathcal{H}$ on input 1^l. It then chooses a random element $x \in G$ and outputs f, G, H and $y = f(x)$. In the set-up phase, V runs $\mathcal{G}$ and sends the output f, G, H, y to P, who checks that $y \in Im(f)$.
- **Commit function**: is defined as a mapping from $H \times \{0,1\}$ to G. Concretely, $\texttt{commit}(r, b) = y^b f(r)$.
- **Hiding Property**: is unconditionally satisfied, since by assumption, P can verify without error that $y \in Im(f)$, and in this case a commitment to b will have distribution independent of b, namely the uniform distribution over $Im(f)$. This is because P chooses r uniformly in H, group homomorphisms are regular mappings (they map a fixed number of elements to one), and finally because multiplication by the constant y is a one-to-one mapping in the subgroup $Im(f) \leq G$. Thus these commitments are in fact perfectly hiding.
- **Binding Property**: Follows from the following fact: given any algorithm A that breaks the binding property of this scheme with success probability ϵ in time T_A. Then there exists an algorithm A' that inverts homomorphisms generated by $\mathcal{H}$ with success probability ϵ as well and in time T_A plus the time needed for one inversion and one multiplication in G.
  This is easy to show: we are given $f : H \to G, y$ and must invert f in point y. We run A on input f, G, H, y pretending this is the public key of a commitment scheme instance. A outputs in time T_A a commitment c and openings $r_0, 0$ and $r_1, 1$. We now output $x = r_0 r_1^{-1}$. We leave it to the reader to show that if indeed $r_0, 0$ and $r_1, 1$ are valid openings of c, then $f(x) = y$.

There are several things to notice about this scheme and its proof:

- In the set-up phase, it is essential that P is convinced that $y \in Im(f)$. It would be devastating if V could get away with selecting $y \notin Im(f)$ (can you see what would go wrong?). For the RSA example, this was not a problem: P can check for himself that f is surjective which implies that $y \in Im(f)$. In other cases, the set-up phase must be more elaborate in that V must convince P that the public key was correctly selected. This can be done using a zero-knowledge protocol (see Section 3). In particular it is always possible given any homomorphism f for V to convince P in zero-knowledge that $y \in Im(f)$, which is in fact enough for the scheme to be secure.
- The proof of the binding property is an example of so called proof by *black-box reduction*: we want to show that existence of cryptographic primitive $P1$ implies existence of primitive $P2$. In our case $P1$ = one-way group homomorphisms and $P2$ = unconditionally binding commitments schemes.
  To do this, we first make a construction that takes an instance of $P1$ and builds an instance of $P2$. This construction treats the instance of $P1$ as a

black-box: anything that satisfies the abtract requirements (e.g for being a one-way group homomorphism) will do. We then show that any algorithm that can break $P2$ can be used to build an algorithm that breaks $P1$ "just as efficiently". This is done by a reduction that treats the algorithm attacking $P2$ as a black-box: it doesn't care how the algorithm manages to break $P2$, it just uses the fact that it succeeds in doing so. We conclude that if the security properties of $P2$ are violated, so are those of $P1$, and conversely, if secure instances of $P1$ exist so do secure instances of $P2$.
This black-box paradigm has proven extremely productive in many areas of cryptography. See the article by Bellare in this volume for more information on this.

- The black-box reduction we built to show the binding property is actually much stronger than needed for the definitions: for that, it would have been enough if we had shown that the running time of A' was polynomial in T_A, and that the success probability of A' was a polynomial function of ϵ. Still, what we have done is far from being overkill: what we want, in practice as well as in theory, is basically to say that "breaking the comitment scheme is just as hard as it is to invert the homomorphism". And of course we can make this claim in a stronger sense, the more efficient a reduction we have. Hence if we want results that are meaningful not only in theory, but also in practice, it is important to try to obtain as efficient a reduction as possible in any proof of this type.
- Group homomorphisms can also be used to build unconditionally binding commitments, and to build schemes where one can commit to many bits in the same commitment. For details on this, see [11].

2.4 Theoretical Results of Existence of Commitment Schemes

It is easy to see that if any commitment scheme in the two-player model exists, then a one-way function must also exist. For example, in our definition, it is clear that the function `commit` must be one-way in order for the commitment scheme to be secure.

Hence, the optimal result would be to show existence of commitment schemes based only on the existence of one-way functions. Such a result is known for one type of commitment scheme, and follows from a result of Naor [29] (actually, Naor's result is the last in a chain of results linking one-way functions with commitments through other primitives such a pseudorandom generators, for references on this, see [29]):

Theorem 2. *If one-way functions exist, then commitment schemes with unconditional binding (and computational hiding) exist.*

For unconditionally hiding schemes, the situation is different. In [30], the following is proved:

Theorem 3. *If one-to-one surjective one-way functions exist, then commitment schemes with perfect hiding (and computational binding) exist.*

In [31], with generalizations to multi-bit commitments in [18], the following is proved:

Theorem 4. *If collision-intractable hash functions exist, then there exists commitment schemes with unconditional hiding (and computational binding).*

Loosely speaking, a collision intractable hash function is a function $h : \{0,1\}^k \to \{0,1\}^l$ such that $l < k$, h is easy to compute, but it is hard to find $x \neq y$ such that $h(x) = h(y)$ (although such values must of course exist – for a precise definition, see [15]).

Whereas the first two of these three basic results involve very complex reductions and therefore are of limited practical value, the third one can lead to very practical schemes.

There is no implication known in either direction between existence of one-way one-to-one functions and collision-intractable hash functions, so the last two results are in this sense "independent" from a theoretical point of view. The obvious question "does existence of one-way functions imply existence of unconditionally hiding commitments?" is a long standing open problem.

3 Zero-Knowledge Protocols

3.1 Introduction

In order for a modern computer network to offer services related to security, it is a basic necessity that its users have access to private information, in the form of e.g. passwords, PIN codes, keys to cryptosystems, keys to signature systems etc. If I know every bit of information that you know, it will be impossible for the rest of the system to tell us apart.

This introduces a basic problem when implementing such services: of course I want my private information to stay private, but as soon as I start using it as input when computing the messages I send to other parties on the net, this introduces the risk of leaking private information, in particular if the parties I interact with do not follow the protocols, but instead do their best to maliciously trick me into revealing my secrets. This dilemma can be solved if we use protocols on the net for which we can *control exactly* how much sensitive information is being released, even in presence of adversarial behavior. The concept of zero-knowledge, first introduced by Goldwasser, Micali and Rackoff [26], is one approach to the design of such protocols.

As an easy example, consider the classical user identification problem: we have a host computer that would like to verify the identity of users that try to log on. The classical solution is assign a private password to each user. When logging on, the user types his user name and password, this is sent to the host, who checks it against a stored list.

The security problems with this are many and well known. Let us concentrate here on the obvious problem that if an adversary eavesdrops the line, he can pick up the password, and then impersonate the user. When trying to solve this, the

immediate reaction might be to propose that we transport instead the password in a protected way. Perhaps we should just encrypt it?

But then we would be barking up the wrong tree. We have to ask ourselves first what the purpose of the protocol is. Is it to send the password from the user to the host? No! - we are trying to identify the user. What we have done intitally is to assign a secret (the password) to each user, so when someone types his user name, say xxx, this is equivalent to claiming that a certain statement is true, in this case "I know the secret corresponding to user name xxx".

The *only* thing the host needs to know here is only 1 bit of information, namely whether this statement is true or not. The real purpose of the protocol is to communicate this piece of knowledge to the host. Sending the secret of the user in clear is just one way, and not even a very intelligent way to do it.

In general, we could have the user and host conduct an interactive protocol, where at the end, the host can compute a one-bit answer saying whether the user was successfull in proving himself or not. Here of course we have to design the protocol such that if the user really knows the right secret, he will be successful, whereas the answer will be no, if the user is cheating and does not know the secret. If this is satisfied, we can say that the protocol really does communicate this 1 bit of knowledge saying whether the claim is true or not. But moreover, if we design the protocol correctly, we can actually obtain that it communicates *nothing more than this*. Which would mean that for example an eavesdropper listenting to the communication would just as far away from guessing the user's secret after seeing the conversation as he was before.

This leads to our first very loose definition of zero-knowledge: *a protocol is zero-knowledge if it communicates exactly the knowledge that was intended, and no (zero) extra knowledge.*

3.2 A Simple Example

One way to realize the scenario where each user has his own secret is to use a public key cryptosystem. So suppose each user A has a private key S_A known only to him, whereas everyone, including the host, knows the public key P_A.

Now, if the cryptosystem is any good, it must be the case that decrypting a ciphertext $C = P_A(M)$ is hard unless you know the private key. Hence, if you meet someone who is able to decrypt a ciphertext you send him, it is reasonable to conclude that he knows S_A, at least if you make sure that the message you encrypt is randomly chosen from a large set, such that the probability of guessing your choice is negligible. This suggests the following simple protocol, where we rename the players so that the description fits better with the definitions to follow: the user, who is the one wanting to convince the other about the truth of some claim will be called the *Prover* (P), and the host, who is interested in checking that the claim is true, will be called the *verifier* (V).

1. If the prover claims to be A, the verifier chooses a random message M, and sends the ciphertext $C = P_A(M)$ to the prover.
2. The prover decrypts C using S_A and sends the result M' to the verifier.
3. The verifier accepts the identity of the prover if and only if $M' = M$.

Let us look at this protocol from the point of view of both parties. Should the verifier be happy about this protocol? the answer is *yes* if the public key system used is secure: while the owner of S_A can always conduct the protocol successfully, an adversary who knows only the public key and a ciphertext should not be able to find the plaintext essentially better than by guessing at random.

Now what about security from the (honest) prover's point of view - is any unnecessary knowledge being communicated to the verifier here? At first sight, it may seem that everything is OK: if we consider the situation of the verifier just after sending C, then we might argue that since the verifier has just chosen the message M itself, it *already knows* what the prover will say; therefore it learns no information it didn't know before, and so the protocol is zero-knowledge.

But this reasoning is WRONG! It assumes that the verifier follows the protocol, in particular that C is generated as prescribed. This is of course unreasonable because nothing in the protocol allows the prover to check that the verifier is behaving honestly. This is more than a formal problem: assume that an adversary takes control of the verifier, and sends instead of a correctly generated C some ciphertext C' intended for the correct prover, that the adversary has eavesdropped elsewhere. And now, following the protocol, the unsuspecting prover will kindly decrypt C' for the adversary!

This is certainly not the kind of knowledge we wanted to communicate, and hence this protocol is definitely not zero-knowledge. How can we repair this protocol? The basic problem we saw is that when the verifier sends C, we are not sure if it really knows the corresponding plaintext M. If it did, we would be fine. However, the verifier will of course not be willing to reveal M immediately, since from its point of view, the purpose of the protocol is to test if the prover can compute M based only on C. And for the reasons we saw above, the prover will not be willing to go first in revealing M either. So we have a sort of deadlock situation similar to the one in the coin-flipping by telephone problem from the former section. Like that problem, this one can be solved using commitments.

Assume we have a commitment scheme that lets the prover commit to any message that can be encrypted by the public key system. Let $\texttt{commit}(r, M)$ denote a commitment to message M (using random choice r - we can always commit bit by bit if no more efficient methods are available). Then consider the following:

1. If the prover claims to be A, the verifier chooses a random message M, and sends the ciphertext $C = P_A(M)$ to the prover.
2. The prover decrypts C using S_A and sends a commitment to the result $\texttt{commit}(r, M')$ to the verifier.
3. The verifier sends M to the prover.
4. The prover checks if $M = M'$. If not he stops the protocol. Otherwise he opens the commitment, i.e. he sends r, M' to the verifier.
5. The verifier accepts the identity of the prover if and only if $M' = M$ and the pair r, M' correctly opens the commitment.

Proving formally that this repair works turns out to be surprisingly complicated, but possible. The necessary techniques can be found e.g. in [5,24]. Here, however, we are only interested in arguing informally why such a solution should have a chance of working: first, the protocol demonstrates that the prover can decrypt C based on C alone, since when the verifier finds the right plaintext inside the commitment, this shows that the prover knew it already in step 2, by the binding property of the commitment scheme. As for zero-knowledge, either the verifier knows M or not. If yes, then it can send the correct M in step 3, but then it already knows what it will find inside the commitment in step 5 and so learns nothing new. If not, then it cannot send the right value in step 3, the prover will stop the protocol, and the verifier will be left with an unopened commitment which by the hiding property is a useless piece of information that might represent any value whatsoever.

If nothing else, this example demonstrates first the fundamental role that commitments often play in protocol design, and second that we should *not* argue security of protocols based on what players *should be* doing according to the protocol, we must take any adversarial behavior into account. Finally, it also demonstrates one basic design principle for zero-knowledge protocols that continue to appear in all sorts of incarnations: have the prover demonstrate something the verifier already knows. The problem with this is, in the above protocol as in all protocols of this type, to ensure that the verifier does indeed know in advance what the prover will say. For other examples of this kind, see e.g. the graph nonisomorphism protocol from [25].

3.3 Definitions

Interactive Proof Systems and Proofs of Knowledge The protocols to follow will take place as interactions between two *Interactive Turing Machines*, i.e. ordinary probabilistic Turing machines that are in addition equipped with communication tapes allowing a machine to send and receive messages from the other one. A formal definition can be found in [26].

To define interactive proof systems, we assume that one machine, called the prover (P) has infinite computing power, and the other called the verifier (V) is polynomial time bounded. The machines get a common input string (usually called x). Running the machines on some input x results in V outputting *accept* or *reject* after which the machines halt. We say that the pair (P, V) accepts or rejects x accordingly. Finally a binary language L is given.

In the previous section, we talked about the intutive model where the prover claims that "a certain statement is true". We now specialize to the concrete case where the prover claims that a certain logical statement is true, namely that $x \in L$. This can be compared in the real world to convincing someone that a certain theorem is true. Concretely, we have the following definition [26]:

Definition 5 *The pair* (P, V) *is* an interactive proof system for L *if it satisfies the following two conditions:*

Completeness: *If $x \in L$, then the probability that (P, V) rejects x is negligible in the length of x.*

Soundness: *If $x \notin L$ then for any prover P^*, the probability that (P^*, V) accepts x is negligible in the length of x.*

What these conditions say is that first, the honest prover can always convince the verifier about a true statement, but that there is no strategy that convinces him about something false. Both conditions are required to hold except with negligible probability, and are in fact rather strong: even if the honest prover can convince the verifier using only polynomial computing time, there must be no way to cheat the verifier, even using infinite computing power.

There are two features that make this definition interesting, namely that interaction and error probabilities are allowed. It is easy to see that if the prover is only allowed to send a single message to the verifier, who should then be able to check without error that the input x is in L, we would only be redefining the class NP. But with these two features, the model becomes much more powerful in terms of the class of statements that can be proved, as we shall see.

There is a variant of this, known as *Proofs of Knowledge*, where the prover's claim has a different flavor: he claims to know a certain piece of information (such as a secret key corresponding to a given public one). Such proof systems can be defined in a similar model, where however the completeness and soundness properties are replaced by *knowledge completeness* and *knowledge soundness*. The first property simply says that if the prover knows the claimed information and follows the protocol, he can almost always convince the verifier. The second, loosely speaking, says that if some prover can, using whatever strategy, convince the verifier with substantial probability, then the prover knows the information in question. By "knowing the information" we mean that the prover can compute it, and that the time he needs to do so is roughly inversely proportional to the probability with which the verifier gets convinced. A precise definition can be found in [2].

Interactive Arguments Another variant of Interactive proof systems is known as *Interactive Arguments* and has perhaps more direct relations to practical protocols. In this type of protocol, we want the prover to be polynomial time, but on the other hand are only concerned about polynomial time provers cheating the verifier. This can be said to be a complexity theorist's way of modelling the situation where only realistic computing power is available to prover and verifier.

The simplest way to define an interactive argument for a language L, is to say that it is an interactive proof system, but with two changes:

- The honest prover is required to be probabilistic polynomial time, and its only advantage over the verifier is that it has a private auxiliary input. The completeness condition says that for every $x \in L$, there is an auxiliary input that allows the prover to convince the verifier almost always[1].

[1] In order for the protocol to be interesting at all, the prover must have some advantage - otherwise the verifier might as well go and solve the problem on his own.

– The soundness condition says "for any probabilistic polynomial time prover", in stead of "for any prover".

It turns out that this simplistic definition of soundness is not quite adequate in all cases, but it will do for us here. For a more complete set of definitions and a discussion of this, see [17].

Zero-Knowledge Zero-Knowledge can be seen as an extra property that an interactive proof system, a proof of knowledge or an interactive argument may have. Here, we want to express the requirement that whatever strategy the verifier follows, and whatever a priori knowledge he may have, he learns nothing except for the truth of the prover's claim. We do this by requiring that assuming the prover's claim is true, the interaction between the prover and verifier can be efficiently simulated *without interacting with the prover*.

A verifier that tries to cheat the prover can be modelled by an arbitrary probabilistic polynomial time machine V^* that gets an auxiliary input H of length some fixed polynomial in the length of the common input x. This represents a priori information that V^* could have e.g. from earlier executions of the protocol, which it may now use to trick the prover into revealing more information. By a conversation between P and V we mean the ordered concatenation of all messages sent between P and V in an execution of the protocol. We get the following [26]:

Definition 6 *An interactive proof system or argument (P, V) for language L is* zero-knowledge *if for every probabilistic polynomial time verifier V^*, there is a simulator M_{V^*} running in expected probabilistic polynomial time, such that for $x \in L$ and any auxiliary input H, the distribution of conversations output by M_{V^*} on input x, H is* computationally indistinguishable *from the distribution of conversations produced by (P, V^*) on input x and H (given to V^*).*

By "computationally indistinguishable", we mean the following: consider any probabilistic polynomial time distinguisher D, who gets as input $x \in L$ and H as above. In case 0 it also gets a conversation generated by (P, V^*) on this input, in case 1 it gets a simulated conversation generated from the same input. D outputs a bit, which we can think of as its guess at which case we're in. Let $p_i(x, H)$ be the probability that D outputs 0 from this experiment, assuming we are in case i, $i = 0, 1$. These probabilities are taken over the coin tosses used for producing the conversations as well as over internal coin tosses of D. Then computational indistinguishability means that $|p_0(x, H) - p_1(x, H)|$ is negligible in the length of x [2].

For some protocols, we can obtain that real and simulated conversations have exactly the same distribution, in this case we speak of *perfect zero-knowledge*.

[2] Usually, one allows D to be a non-uniform algorithm, i.e. it is specified by a family of polynomial size Boolean circuits – but this is not so important for our purposes in this paper

In other cases, the distributions are different, but very close to each other in the sense that the amount of probability mass one must move to change one into the other is negligible; then we speak of *statistical zero-knowledge*. Clearly, perfect zero-knowledge implies statistical zero-knowledge, which in turn implies computational zero-knowledge as defined above.

At first sight, the zero-knowledge definition may seem intuitively to contradict the proof system definition: first we say that the verifier should be convinced by talking to the prover. But then we require that the whole conversation can be efficiently simulated *without* talking to the prover – doesn't this mean that having a conversation with the prover cannot be convincing?

Fortunately, this is not the case. The explanation is that a simulator has some degrees of fredom that the prover doesn't have when executing the real protocol. In particular, the simulator can generate messages of a conversation in any order it wants - it can start with the last message first, and then try to find earlier messages that match. A real prover is forced by the verifier to proceed in the protocol with the correct time ordering of messages. And this is why it can be possible that even an infinite prover cannot cheat the verifier, and still a simulator with no special knowledge or computing power can simulate the conversation. For concreteness, see the example below.

3.4 An Example

We describe here a simple example taken from [25], namely a perfect zero-knowledge proof system for the graph isomorphism problem: the common input in this case is a pair of graphs G_0, G_1 each on n nodes, and the prover claims the graphs are isomorphic: there is a permutation π (an isomorphism) such that by permuting the nodes of G_0 according to π (and connecting two resulting nodes iff their preimages were connected in G_0), one obtains the graph G_1. We say that $\pi(G_0) = G_1$.

Note that no general probabilistic poly-time algorithm is known for deciding if two graphs are isomorphic. We will use n as a measure of the length of the input. In the protocol, we actually do not need P to be infinitely powerful, although the definition of proof systems allows this; it is enough that he knows an isomorphism π. The protocol works by repeating sequentially the following steps n times:

1. P chooses a random permutation ϕ on n points and sends $H = \phi(G_0)$ to V.
2. V chooses at random a bit b, and sends it to P.
3. If $b = 0$, P sets $\psi = \phi^{-1}$. Else he sets $\psi = \pi\phi^{-1}$. He sends ψ to V, who checks that $\psi(H) = G_b$, and rejects immediately if not.

The verifier accepts, only if all n iterations were completed successfully.

First, let us check that this is a proof system. Completeness is obvious: if indeed $\pi(G_0) = G_1$ and $\psi(G_0) = H$, then it follows trivially that V's check will be satisfied for both values of b. Soundness can be argued as follows: observe that we must prove something here assuming that the prover's claim is wrong, which

in this case means that G_0 is not isomorphic to G_1. Now assume that in one of the n iterations, P can answer both values of b with a permutations that satisfy V's check. Let ψ_0, ψ_1 be the permutations sent as response to $b = 0, 1$. Since V's checks are satisfied, we know that $\psi_0(H) = G_0$ and $\psi_1(H) = G_1$. It follows that G_0 is isomorphic to G_1 under the isomorphism $\psi_1\psi_0^{-1}$, a contradiction. Consequently, it must be the case that in all n iterations, the prover is able to answer at most one of the 2 possible values of b. Hence the probability of acceptance is at most 2^{-n}, which is certainly negligible in n.

Finally, let us show that the protocol is perfect zero-knowledge. To this end, we must build a simulator. The easiest way to think of a simulator usually is to think of it as an algorithm that tries to complete the protocol, playing the role of the prover, but of course without any special knowledge or computing power. Thus, a non-trivial trick is needed. In our case, we cannot just execute the protocol: we saw in the argument for soundness that knowing how to answer both of V's challenges at the same time implies we can compute an isomorphism between G_0 and G_1, and no efficient algorithm is known for this. However it *is* possible to prepare in such a way that one of the challenges can be answered. This is used in the following algorithm for a simulator M:

1. Start the machine V^*, which means giving it inputs G_0, G_1 (plus possibly some auxiliary input H) and supplying random input bits for V^*. These are needed since V^* is allowed to be a probabilistic algorithm; we choose the random bits here and keep them fixed for the rest of the simulation.
2. To simulate one iteration, execute the following loop:
   (a) Choose a bit b' and a permutation ψ at random. Set $H = \psi^{-1}(G_{b'})$ and send H to V^*.
   (b) Get b from V^*. If $b = b'$, output H, b, ψ and exit the loop. Else, reset V^* to its state just before the last H was chosen, and go to step 2a.

   If we have completed simulation of all n iterations at this point, then stop. Else start at Step 2a again.

So in simulating one iteration, the simulator prepares to answer question b', and hopes that this is the question V^* will ask. If this happens, we're in business and can complete the simulation of the current iteration. Otherwise we just pretend the bad case never happened by rewinding V^* and then we try again. At first sight, this rewinding technique can seem somewhat strange. However, it is essentially the same as rebooting your PC when it crashes: if we reach a configuration we don't like, we take the machine back to one we like better; so in this sense rewinding is an everyday experience[3].

To show that this simulator works, we need to show two things: M runs in expected polynomial time, and the distribution output by M is exactly the same as in a real protocol run.

Observe first, that by definition of zero-knowledge, we always prove correctness of a simulation assuming that P's claim is true, in our case this means that

[3] If your PC never crashes, you should be making a fortune in consultancy instead of reading this book!

G_0 is isomorphic to G_1. Let S be the set of all graphs isomorphic to G_0 (or G_1). It is straightforward to check that the distribution of H generated in the simulation is the same as in the real protocol, namely the uniform distribution over S. In particular it is independent of b'. It follows that the b chosen by V^* must be independent of b' as well, and so $Prob(b' = b) = 1/2$. Hence the expected number of times we do the loop to simulate one iteration is 2, and so the whole simulation takes expected time $2n$ times the time to go though the loop once, which is certainly polynomial in n.

Finally, the output distribution: The simulator produces for the i'th iteration a triple (H, b, ψ). First note that the candiate H's produced in step 2a are uniform over S. By independency of H and b' the decision to keep H or rewind and throw it out does not depend on the choice of H. Hence the H's actually output are also uniform over S, as in the real protocol. The b occurring in a triple is by construction always the value V^* would send having seen H (recall that we fix V^*'s random bits initially). And finally ψ is a random permutation mapping H to G_b, just as in the real protocol. Thus the output distribution of M matches the real protocol exactly.

This example demonstrates another basic design idea for zero-knowledge protocols: the prover is asked to answer one out of some set of questions. We set it up such that he can only answer all of them if his claim is true, but such that one can always prepare for answering any single question properly. For other examples of this type of protocol, see e.g. [11, 12, 13, 21, 27, 32].

3.5 Known General Results and Open Problems

Having seen a few examples of zero-knowledge proofs, it is natural to ask some more general questions:

- Which languages have interactive proofs?
- Which languages have (perfect/statistical) zero-knowledge interactive proofs?
- Can we compose several zero-knowledge protocols and obtain again a zero-knowledge protocol?

It turns out that the answers depend strongly on whether the prover (and cheating provers) are allowed infinite computing power, or only polynomial time, that is, if we are talking about proof systems or arguments.

Results on Interactive Proofs and Arguments For an unbounded prover, the first question has been answered recently by Shamir [33], where we define $IP = \{L |\ L$ has an interactive proof system$\}$:

Theorem 7. *$IP = PSPACE$, i.e. the statements that an all powerful prover can prove to a polynomially bounded verifier, are precisely those that can be verified using polynomially bounded memory (but possibly unbounded time).*

If the prover is polynomially bounded, it is clear that his only possible advantage over the verifier is that he may have more information than the verifier.

In this case, the best the prover can do to convince the verifier is to simply send his information, s, say, to the verifier who should then be able to check the prover's statement based on s, where some error probability is allowed. The class of languages allowing such probabilistic verification of membership given auxiliary knowledge is already well known as $NBPP$ or MA. So if we define Bounded-ProverIP to be the class of languages that have interactive arguments, then we have:

Theorem 8. *Bounded-Prover$IP = MA$*

Results on Zero-Knowledge We first look at the case of zero-knowledge interactive proofs. Let

$$ZKIP = \{L|\ L \text{ has a zero-knowledge interactive proof system}\}.$$

Goldreich, Micali and Wigderson [25] show that any $NP \subset ZKIP$ if commitment schemes with unconditional binding exist. This was extended to all of IP in [6]. This, together with Theorem 2 gives:

Theorem 9. *If one-way functions exist, then $ZKIP = IP$.*

It is natural to ask also about statistical and perfect zero-knowledge. Let $PZKIP$, $SZKIP$ denote the classes of languages with perfect, resp. statistical zero-knowledge proof systems. Except for the trivial $PZKIP \subset SZKIP \subset ZKIP$, very little is known with certainty. We know that a few languages with nice algebraic properties, such as graph isomorphism and quadratic residuosity[4] are in $PZKIP$. Also the complements of these languages are in $PZKIP$, and this is interesting since a problem such graph non-isomophism is not known to be in NP, and so it seems unlikely that $PZKIP \subset NP$. It also seems unlikely that the converse inclusion holds: Fortnow [20] has proved that if it does, then the polynomial hierachy collapses - something believed to be false by many complexity theorists. In fact this can be seen as evidence that the graph isomorphism problem is *not* NP-complete, one of the few real evidences that have been found.

A nice characterization of languages in $PZKIP$ or $SZKIP$ is an interesting open problem. We do know, however, some information on complete problems in $SZKIP$ [35], and that a proof system that is statistical zero-knowledge w.r.t. the honest verifier implies existence of a proof system that is statistical zero-knowledge in general [23].

Let us mention also a variant of the zero-knowledge concept, known as *non-interactive zero-knowledge*. In the non-interactive zero-knowledge model, an unbounded prover and a polynomial time verifier share access to a random string α. It is assumed as a part of the model, that α contains independent random bits. The prover must now convince the verifier that a common input x is in some language L by sending only 1 message σ (hence the "non-interactiveness"). The verifier then checks σ against x and α and accepts or rejects.

[4] This is the set of pairs of numbers n, a, where a is a square modulo n

This proof system is called sound if whenever $x \notin L$, no prover can make the verifier accept with non-negligble probability over the choice of α. It is zero-knowledge if the pair σ, α can be simulated with an indistinguishable distribution in expected polynomial time.

This model was introduced by Blum, de Santis, Micali and Persiano [7] to formalize the absolute minimal amount of interaction required to prove non-trivial statements in zero-knowledge.

To distinguish between all the relevant complexity classes now involved, we use the following notation: Let $NIZK$, $NIPZK$ and $NISZK$ denote the classes of languages with non-interactive computational, perfect and statistical zero-knowledge proof systems.

Lapidot and Shamir [28] have shown that

Theorem 10. *If one-to-one surjective one-way functions exist, then* $NP \subset NIZK$.

It is an open question whether any one-way function would be sufficient.

The non-interactive model is weaker than the normal interactive model in that interaction is not allowed, but in another respect stronger because a random string with correct distribution is assumed to be given "for free". It is therefore not immediately clear whether any language that has a non-interactive zero-knowledge proof system also has an interactive one and vice versa. In [16], Damgård shows:

Theorem 11. *We have that* $NIZK \subset ZKIP$, $NISZK \subset SKZIP$ *and that* $NIPZK \subset PZKIP$.

We already know that if one-way functions exist, $ZKIP = PSPACE$. This together with the fact that a non-interactive proof uses only a constant number of rounds provides very strong evidence that the first containment above is proper, since it is extremely unlikely that a constant number of rounds would be sufficient to prove all of IP. On the other hand, the corresponding questions for the classes where statistical or perfect zero-knowledge are required seem more open.

For the interactive argument model - which is the most interesting one in practice - the situation is again quite different. We have already seen that the only statements we can hope to prove at all are those in the class MA.

So the remaining question is whether we can prove any such statement in zero-knowledge, or even in perfect zero-knowledge.

In [4], Brassard Chaum and Crépeau show that any MA-language has a perfect zero-knowledge argument, if commitment schemes with unconditional hiding exist. It follows that

Theorem 12. *If one-to-one surjective one-way functions exist, resp. if collision-intractable hash functions exist, then any language in* MA *has a perfect, resp. statistical zero-knowledge interactive argument.*

There is currently no implication known either way between the two assumptions listed in this theorem. Proving the theorem assuming only existence

of one-way functions is a challenging open problem. Note that there is no conflict between this result and that of Fortnow mentioned above: Fortnow's result talks only about interactive proofs (and not arguments).

The concrete protocol constructions used to prove that all NP problems have zero-knowledge proof systems and arguments are in fact also proofs of knowledge. So equally general results on proofs of knowledge follow immediately.

On Composition of Zero-Knowledge Protocols In general, the *sequential* composition of two zero-knowledge protocols is again zero-knowledge. An example of this is the graph isomorphism protocol shown above – it is in fact the result of repeating sequentially a basic step several times, where each step is zero-knowledge.

However, if we try doing the repetitions in parallel, then the resulting protocol does not seem to be zero-knowledge: we would get a scenario where P would send many graphs $H_1, ..., H_n$ at once, V would send challenges $b_1, ...b_n$ and P would reply by $\psi_1, ..., \psi_n$. The resetting technique for simulation does not work anymore: we would be forced to try to guess in advance all the bits $b_1, ...b_n$, and it would take us expected exponential time before the guess was correct. The idea that doing the protocol in parallel is not zero-knowledge may seem counterintuitive at first sight: why should doing it in parallel tell V more about an isomorphism between G_0 and G_1? The answer is that while it might in fact be true that V learns nothing that could help him to compute such an isomorphism, this is not enough for zero-knowledge which requires that V learns *nothing whatsoever* that he could not compute himself. Indeed if the verifier computes its challenge bits as a one-way function of the $H_1, ..., H_n$ received, then it seems that conversation itself would be a piece of information that is difficult for V to generate on his own.

This discussion does not prove that the parallel version of the graph isomorphism protocol is not zero-knowledge, only that the resetting technique will not work for simulating it. However, Goldreich and Krawczyk [24] have shown that there exist protocols that are zero-knowledge, but where the parallel composition provably is not zero-knowledge.

A more complicated scenario which has been considered very recently is that of *concurrent zero-knowledge* where we allow arbitrary interleaving of different instances of protocols, i.e. while P is running a protocol with V_1, it starts doing (the same or) a different protocol with V_2, etc. There is no a priori time ordering fixed between messages sent in different protocols. We can ask whether this entire interaction is simulatable. There are results about this indicating that many well known protocols fail to be zero-knowledge in such a scenario, however, there are also ways around this problem. More information on this can be found in one of the latest papers on the subject by Dwork and Sahai [19], which also contains pointers to more material.

3.6 Applications of Zero-Knowledge

One basic application of zero-knowledge protocols that is important in theory as well as in practice is the usage of zero-knowledge protocols as subprotocols in larger constructions, this could be voting schemes, key distribution protocols, or in general any multiparty computation (see the article by Cramer in this volume for information and references on this). If we do not want to assume existence of secure channels, such constructions are usually not possible in the first place unless one-way functions exist. This means that in building such protocols we can assume without loss of generality that $NP \subset ZKIP$. And so whenever a player A sends a message in a protocol he can convince anybody else in zero-knowledge that he has computed his message according to the rules in the protocol. This follows since if the computation A was supposed to do is feasible in the first place, then the claim that the message is correctly computed can be verified in polynomial time given all A's data, and so is an NP-statement.

It follows that we can automatically transform any protocol that is secure assuming players follow the rules into one that is secure even if players deviate arbitrarily from the protocol. This oberservation was first made in [25].

This can be interesting in practice if the involved zero-knowledge proofs are efficient. However, this is not always the case if we are using the general theoretical results we have covered. While they show what is in principle possible, most of the actual protocol constructions occurring in the proofs of those results are not very attractive in practice.

As an example, we know that a zero-knowledge proof or argument can be given for any NP language, and this is proved by providing a zero-knowledge proof for an NP complete problem such as Boolean Circuit satisfiability (SAT). When we are given a concrete problem instance $x \in L$, where $L \in NP$, then to use the general result, we must first construct from x a Boolean circuit which is satisfiable precisely if $x \in L$, and then use the protocol for SAT.

This approach often results in very large circuits, for problem instances of interest in real life, typically at least 10.000 to 100.000 binary gates. It is therefore of interest to be able to construct instead an ad hoc zero-knowledge protocol for the problem in question, such as the graph isomorphism protocol above. A few problems are "nice" in this sense, in that they allow construction of particularly efficient protocols. This is often true of problems derived from number theory, and we mention some examples below. Still, there are also cases where the only solution we know is to use general techniques. This can be the case e.g. if P wants to show that for a given bit string y he knows x such that $h(x) = y$, where h is some cryptograhic hash function. Since such functions are usually constructed deliberately to have none of the nice algebraic properties that enable efficient zero-knowledge directly, we have to resort to the general techniques. SAT is often the natural NP complete problem to use, so efficient zero-knowledge protocols for SAT are of particular interest. Recent results by Cramer and Damgård in this direction show that one can prove satisfiability of a Boolean circuit while communicating only a number of bit commitments linear in the size of the circuit [11]. Using preprocessing, one can even reduce the proof to one message

containing 2 bits pr. gate in the circuit [12]. Thus, general techniques can in fact be practical in some cases.

Still, the largest potential for practical applications of zero-knowledge comes from extremely efficient protocols specially designed for particular problems such as the quadratic residuosity problem [21], the discrete logarithm problem [32], or the RSA root extraction problem [27]. The typical use here is for the classical user identification problem that we mentioned earlier: each user U gets a solution to a hard problem instance x_U, and can identify himself by proving in zero-knowledge that he knows a solution to x_U. By the zero-knowledge property, none of the proofs conducted by U will help an adversary to find a solution to x_U. Still, by the soundness property, an adversary can only impersonate U if he can find a solution to x_U. So if he succeeds it means he could find a solution to x_U from scratch, and this is not possible if the underlying problem is hard. Using a secure hash function, one can also use these (interactive) identification protocols to build (non-interactive) signature schemes [21]. These can be more efficient than RSA signatures, but have so far only conjectured security in the sense that we do not know how to reduce the security to any well established computational assumption.

The most efficient versions of these protocols yield error probability exponentially small in the security parameter, even though the communication required is only linear. Unfortunately, these protocols are only zero-knowledge against the *honest* verifier, and hence have no provable security in real life. Feige and Shamir [22] point out a possible way around this problem: the identification scenario does not really require the full power of zero-knowledge. It is enough if the protocol does not help the verifier (or anyone else) to find the provers secret (while zero-knowledge ensures that the verifier learns nothing new whatsoever). This is so since we can show that an adversary needs to know the prover's secret to impersonate the prover. Protocols with this weaker property are called *Witness Hiding* (WH), and might conceivably be easier to construct. In [13] Cramer, Damgård and Schoenmakers show that the efficient honest verifier zero-knowledge protocols of [32, 27] can be transformed into WH protocols while preserving the efficiency.

The results just mentioned and many others in the area of efficient zero-knowledge and WH protocols revolve around protocols of a particular form where P sends a message, V sends a random challenge, and P gives an answer that can be checked by V (this is the form of the basic step in the graph isomorphism protocol). While such protocols by themselves have only limited security properties (e.g. they either have large error probability or are only honest verifier zero-knowledge), it turns out that they can be used in a modular way in a number of constructions of protocols and signature schemes with simultaneously high efficiency and provable security. For instance, a prover can show that he knows at least t out of $n > t$ secrets without revealing which t secrets is involved [13, 34]. This can be important, e.g. in protocols where anonymity is desired. For a nice introduction to this entire area, see [8].

References

1. W.Alexi, B.Chor, O.Goldreich and C.P.Schnorr: *RSA and Rabin Functions: Certain parts are as hard as the Whole*, SIAM J.Computing, 17(1988), 194-209. 68
2. M. Bellare and and O. Goldreich: *On Defining Proofs of Knowledge*, Proceedings of Crypto '92, Springer Verlag LNCS, vol. 740, pp. 390–420. 75
3. M. Ben-Or, S. Goldwasser, A. Wigderson: *Completeness theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proc. ACM STOC '88, pp. 1–10. 68
4. G. Brassard, D. Chaum and C. Crépeau: *Minimum Disclosure Proofs of Knowledge*, JCSS, vol.37, pp. 156–189, 1988. 68, 81
5. J.Brandt, I.Damgaård, P.Landrock and T.Pedersen: *Zero-Knowledge Authentication Scheme with Secret Key Exchange*, J.Cryptology, vol 11(1998), 147-160. 74
6. M.Ben-Or, O.Goldreich, S.Goldwasser, J.Håstad, J.Kilian, S.Micali and P.Rogaway: *Everything Provable is Provable in Zero-Knowledge*, Proceedings of Crypto 88, Springer Verlag LNCS series, 37–56. 80
7. Blum, De Santis, Micali and Persiano: *Non-Interactive Zero-Knowledge*, SIAM J.Computing, Vol.20, no.6, 1991. 81
8. R.Cramer: *Modular Design of Secure, yet Practical Cryptographic Protocols*, PhD Thesis, University of Amsterdam 1996. 84
9. C.Crépeau: *Efficient Cryptographic Protocols based on Noisy Channels*, Proceedings of EuroCrypt 97, Springer Verlag LNCS series, vol.1233, p.306-317. 68
10. D. Chaum, C. Crépeau, I. Damgård: *Multi-Party Unconditionally Secure Protocols*, Proc. of ACM STOC '88, pp. 11–19. 68
11. R. Cramer and I. Damgård: *Linear Zero-Knowledge*, Proc. of STOC 97. 68, 70, 79, 83
12. R. Cramer and I. Damgård: *Zero-Knowledge Proofs for Finite Field Arithmetic; or Can Zero-Knowldge be for Free?*, Proceedings of Crypto 98, Springer Verlag LNCS series. 79, 84
13. R. Cramer, I. Damgård and B. Schoenmakers: *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*, Proceedings of Crypto '94, Springer verlag LNCS, vol. 839, pp. 174–187. 79, 84
14. C.Crépeau and J.Kilian:*Achieving Oblivious Transfer using Weakened Security Assumptions*, Proc. of FOCS 88, p.42-52. 68
15. I.Damgård: *Collision Free Hash Functions and Public Key Signature Schemes*, Proc. of EuroCrypt 87, Springer Verlag LNCS series. 71
16. I.Damgård: *Interactive Hashing can Simplify Zero-Knowledge Protocol Design Without Computational Assumptions*, Proc. of Crypto 93, Springer Verlag LNCS series. 81
17. I. Damgård and B. Pfitzmann: *Sequential Iteration of Interactive Arguments*, Proc. of ICALP 98, Springer Verlag LNCS series. 76
18. I. Damgaård, B. Pfitzmann and T.Pedersen: *Statsitical Secrecy and Multi-Bit Commitments*, IEEE Trans.Info.Theory, vol.44 (1998), 1143-1151. 71
19. C.Dwork and A.Sahai: *Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints*, Proc. of Crypto '98, Springer Verlag LNCS series. 82
20. L.Fortnow: *The complexity of Perfect Zero-Knowledge*, Adv. in Computing Research, vol.5, 1989, 327–344. 80
21. A.Fiat, A.Shamir: *How to Prove Yourself, Practical Solutions to Identification and Signature Problems*, proc. of Crypto 86, Springer Verlag LNCS series. 79, 84

22. U. Feige and A. Shamir: *Witness Indistinguishable and Witness Hiding Protocols*, Proc. of STOC '90. 84
23. O.Goldreich, A.Sahai, S.Vadhan: *Honest-Verifier Statistical Zero-Knowledge Equals General Statistical Zero-Knowledge*, Proc. of STOC '98. 80
24. O. Goldreich and A. Kahan: *How to Construct Constant-Round Zero-Knowledge Proof Systems for NP*, Journal of Cryptology, (1996) 9: 167–189. 74, 82
25. O. Goldreich, S. Micali and A. Wigderson: *Proofs that yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design*, Proceedings of FOCS '86, pp. 174–187. 74, 77, 80, 83
26. S. Goldwasser, S. Micali and C. Rackoff: *The Knowledge Complexity of Interactive Proof Systems*, SIAM J.Computing, Vol. 18, pp. 186-208, 1989. 71, 74, 76
27. L.Guillou and J.J.Quisquater: *A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing both Transmission and Memory*, Proc. of EuroCrypt 88, Springer Verlag LNCS 330. 79, 84
28. Lapidot and Shamir: *Publicly Verifiable Non-Interactive Zero-Knowledge Proofs*, Proc. of Crypto 90, Springer Verlag LNCS series. 81
29. M.Naor: *Bit Commitment using pseudo-randomness*, Proceedings of Crypto 89, Springer Verlag LNCS series. 70
30. M.Naor, R.Ostrovsky, S.Venkatesan, M.Yung: *Perfect Zero-Knowledge Arguments for NP using Any One-Way Permutation*, J.Cryptology, vol.11 (1998), 87-108. 70
31. M.Naor, M.Yung: *Universal One-Way hash Functions and their Cryptographic Applications*, Proc. of STOC '89, 33-43. 71
32. C. P. Schnorr: *Efficient Signature Generation by Smart Cards*, Journal of Cryptology, 4 (3): 161–174, 1991. 79, 84
33. A.Shamir: *IP=PSPACE*, Journal of the ACM, vol.39 (1992), 869-877. 79
34. A.De Santis, G.Crescenzo, G.Persiano, M.Yung: *On Monotone Formula Closure of SZK*, proc. of FOCS '94. 84
35. A.Sahai and S.Vadhan: *A Complete Promise Problem for Statistical Zero-Knowledge*, Proc. of FOCS '97. 80

Emerging Standards for Public-Key Cryptography

Burton S. Kaliski Jr.

RSA Laboratories
20 Crosby Drive
Bedford, MA 01730 USA
burt@rsa.com

Abstract. The transition from theory to industry standards presents many challenges, particularly in terms of what features are important and how they are to be specified. Public-key cryptography, now in its third decade, is in the midst of such a transition. With an introduction to the P1363 project *Standard Specifications for Public Key Cryptography*, this survey highlights some of the transitional challenges, and also describes several areas for further research motivated by the standards efforts.

1 Introduction

As public-key cryptography has now moved into its third decade, a maturing of available technology has occurred, as reflected by the widespread deployment of products based on public-key techniques, and the development of standards for public-key cryptography.

Standards have historically been developed for many reasons. Perhaps the most traditional motivation is that of a reference: standard time and standard measurements are two examples. Many standards today, particularly for communications, extend this notion of a reference definition to provide a basis for interoperability, as parties communicate according to a common set of protocols. For security, the protocols are further based on standards for underlying cryptographic techniques, including public-key cryptography.

Another motivation for standards is assurance of some kind of safety; here, fire resistance standards are a classic example. Assurance of security also plays a role in the development of public-key standards.

Public-key standards today tend to be converging on three families of algorithms, where a family is defined by the underlying hard problem on which security is based. The first two families are based on the difficulty of the discrete logarithm problem over finite fields and the difficulty of the elliptic curve discrete logarithm problem. The third is based on the difficulty of integer factorization. As shorthand, these families may be denoted DL, EC, and IF, respectively, and they are the subject of further discussion in the material that follows. For more background on those families, the reader is referred to other articles within this Volume.

I. Damgård (Ed.): Lectures on Data Security, LNCS 1561, pp. 87–104, 1999.

Organization	Web Page
ANSI X9F1	www.x9.org
IEEE P1363	grouper.ieee.org/groups/1363
ISO/IEC JTC1 SC27	www.iso.ch/meme/JTC1SC27.html
NIST	www.nist.gov

Table 1. Web pages of four organizations developing public-key standards.

As noted in an earlier survey [14], the set of standards is as broad as the set of applications, and it would be difficult to write (at least in a short paper) a full description of every standard involving public-key cryptography. However, much of the work is covered by four organizations, so it is possible to convey a reasonable sense of the overall activity by reviewing the four efforts. This is done in Section 2. One of the outcomes of the various work efforts is a general model for public-key standards, which is helpful as a framework for further work. This model, presented in Section 3, also illustrates the techniques in the IEEE P1363 draft standard.

Section 4 gives an interesting account standards development with respect to the "strong primes" issue. The interaction between research and standards development is a challenging one in this regard; new research results motivate different positions in standards, and new requirements from standards motivate new research. The "strong primes" issue is thus one relevant area of research. Several other areas prompted by recent standards development are considered in Section 5.

2 A Survey of Standards Efforts

This brief survey is mainly intended as "snapshot" of current activities as of October 1998 in four organizations: ANSI X9F1, IEEE P1363, ISO/IEC JTC1 SC27, and NIST. New activities are being added continually, and the reader is encouraged to consult the organizations' Web pages for further information (see Table 1). Also, in the interest of brevity in terms of the references (and in view of the ongoing nature of the standards projects), full bibliographic citations for the standards documents are not given. Titles and other information can generally be obtained from the Web pages or directly from the organizations.

2.1 ANSI X9F1

ANSI X9F1 (full name: Accredited Standards Committee X9, Financial Services — Data and Information Security — Cryptographic Tools) develops cryptographic tools for the financial services industry in the United States. Membership is by corporation and meetings are held quarterly. Balloting is conducted through X9F1's parent committees, X9 and X9F, and an approved document becomes an American National Standard.

Standard	Description	Status
ANSI X9.30	DL signatures (DSA)	approved 1995
ANSI X9.31	IF signatures (RSA, RW)	approved 1998
ANSI X9.42	DL key agreement (DH, MQV)	nearly complete
ANSI X9.44	IF key transport (RSA)	in preparation
ANSI X9.62	EC signatures (DSA)	in public comment
ANSI X9.63	EC key agreement / transport (DH, MQV)	in preparation
ANSI X9.80	Prime generation	in preparation

Table 2. ANSI X9F1 standards and draft standards for public-key techniques.

X9F1 has standards and draft standards for digital signatures and key establishment in each of the three families. A standard for prime generation, which underlies all three families, is in development. Table 2 lists the various efforts.

The renewed debate about "strong primes" (Section 4) has emerged as a result of ANSI X9F1's standardization efforts.

2.2 IEEE P1363

IEEE P1363 is developing a comprehensive standard for public-key cryptography in computer and communications systems. Membership is by individual and meetings are held quarterly. Balloting is conducted through P1363's sponsor, the IEEE Computer Society Microprocessor Standards Committee. An approved document becomes an IEEE Standard.

P1363 has a comprehensive draft standard about to begin ballot, whose title is the same as the working group's name, *IEEE P1363: Standard Specifications for Public Key Cryptography*. The draft standard includes a variety of public-key techniques from all three families as well as extensive material on the number-theoretic algorithms underlying the standard and on security considerations. P1363 defines schemes and primitives (see Section 3), but not protocols.

A new project, IEEE P1363a, will develop additional techniques to be added to the P1363 standard. That project has just started, and submissions of new techniques are currently being received.

2.3 ISO/IEC JTC1 SC27

ISO/IEC JTC1 SC27 (full name: International Organization for Standardization / International Electrotechnical Commission — Joint Technical Committee 1, Information Technology — Subcommittee 27, IT Security Techniques). Membership is by country, although experts participate in the three working groups of SC27. Meetings are held several times a year. Balloting is conducted through ISO and IEC, and an approved document becomes an international standard.

SC27 has projects involving many aspects of cryptography, with both symmetric and public-key techniques (often from multiple families) in the same set

Project	Description
ISO/IEC 9796	Signatures with message recovery
ISO/IEC 9798	Entity authentication
ISO/IEC 11770	Key agreement / transport
ISO/IEC 13888	Nonrepudiation
ISO/IEC 14888	Signatures with appendix

Table 3. Some ISO/IEC SC27 projects involving public-key techniques.

of documents, which often have multiple parts. Table 3 lists some of the current efforts. SC27 defines protocols as well as other techniques and does not make as strong a distinction between schemes and primitives as P1363 does.

2.4 NIST

NIST, the U.S. National Institute of Standards and Technology, develops standards for the U.S. government, including Federal Information Processing Standards (FIPS). The Computer Security Act (1987) gave NIST the charter for cryptography standards for the U.S. government. Although NIST submits documents for public review, there is no ballot process, and final approval is by the U.S. Secretary of Commerce.

NIST has two standards involving public-key techniques, FIPS PUB 186 (Digital Signature Standard), and FIPS PUB 196 (Entity Authentication Using Public Key Cryptography). A key agreement / exchange standard is also in preparation.

NIST is also developing the Advanced Encryption Standard (AES), which though not a public-key standard, is clearly a major achievement in terms of the synergy between standards development and research in cryptography.

2.5 Differences and Coordination

The four organizations, though developing standards based on related technology, have significant differences. ISO/IEC JTC1 SC27 and IEEE P1363 focus more on cryptographic building blocks and leave a fair amount of flexibility. ANSI X9F1 is oriented toward U.S. banking requirements and includes considerations relevant to auditing and validation of security components. NIST is oriented toward U.S. government requirements for unclassified data. These differences result in generally related but not necessarily compatible results.

Despite the differences, there is significant coordination. IEEE P1363 and ANSI X9F1 have overlapping membership and an informal understanding that ANSI X9F1 will adopt or "profile" IEEE P1363 specifications to meet banking requirements (although the reverse is also occurring, where IEEE P1363 generalizes some of the ANSI X9F1 specifications). NIST has stated that it will accept new ANSI X9F1 standards for government purposes, in addition to its existing

Digital Signature Standard, FIPS 186. (Interestingly, ANSI X9F1 had previously adopted FIPS 186 as the basis for ANSI X9.30.) ANSI X9F1 documents are promoted into the international standards process through the banking standards committee ISO TC68, which has some coordination with ISO/IEC JTC1 SC27.

2.6 Related Efforts

Another standards effort of interest is the Public-Key Cryptography Standards (PKCS) series (`www.rsa.com/rsalabs/pubs/PKCS/`) coordinated by RSA Laboratories. PKCS builds consensus among an informal, international audience of cryptography developers and is intended as a catalyst for more formal standards development. PKCS also follows the three-family model, with the RSA algorithm (IF family) covered in PKCS #1, Diffie-Hellman (DL family) in #3, and the EC family in the proposed #13.

Also of particular interest today are the standards being developed in the security area of the Internet Engineering Task Force (`www.ietf.org`), many of which involve public-key protocols. Some of the more notable efforts are Public-Key Infrastructure (X.509) (pkix), S/MIME Mail Security (smime), IP Security Protocol (ipsec) and Transport Layer Security (tls).

3 A General Model for Public-Key Standards

As standards for public-key cryptography have emerged, a classification of the types of public-key techniques has been developed as well. The classification, a result of attempts to specify public-key techniques in a common manner, provides a natural framework or model for new standards development, as well as for research into new techniques.

The primary characteristic of the model is the separation of public-key techniques into two "levels": primitives and schemes. Primitives are basic mathematical operations like RSA encryption, $c = m^e \bmod n$. Schemes are sets of related operations combining primitives with additional techniques, like signature operations that involve the additional technique of hashing. Primitives are intended for low-level implementation as in a crypto-accelerator, schemes are intended as components of high-level application protocols and services. In addition, schemes are intended to be "secure" on all messages they process, whereas primitives are assumed to be difficult to compute (or invert) only on average.

As examples of schemes and primitives, some techniques from IEEE P1363 will be mentioned. Background on the P1363 naming convention will be helpful here. The general form of a P1363 name consists of three fields:

$$\textit{family} \quad \textit{type} \quad - \quad \textit{instance}$$

where *family* is the two-character designation for the underlying hard problem (DL, EC or IF); *type* is a two- to four-character shorthand for the type of technique, and *instance* is the name of a particular instance of the given type.

With the focus on P1363, protocols are not covered in the general model presented here. Examples of protocols include entity authentication protocols where one party verifies another party's presence, and key establishment protocols where parties agree on or exchange a session key. Such protocols can readily be built out of the various schemes, and in general can be built in a generic fashion where it is only the type of scheme that matters, not the specific scheme. For instance, it is possible to build an entity authentication scheme from any signature scheme. Thus protocols need not be defined in terms of basic mathematical operations, which further justifies the separation between primitives and schemes.

3.1 Primitives

A *primitive* is a basic mathematical operation from which other cryptographic techniques can be built. By itself, a primitive provides a degree of computational security in that it may be difficult on average to compute a primitive (or perhaps to invert one) without access to a certain key.

Types of Primitive The following types of primitive are defined in P1363.

Secret value derivation. A *secret value derivation primitive* (denoted SVDP) combines one or more private keys with one or more public keys to produce a secret value. The same secret value can be obtained by combining the corresponding public keys with the corresponding private keys.

Secret value derivation is relatively new terminology, being introduced in P1363 for specifying basic operations like the Diffie-Hellman step that combines one party's public key, say y_B, with another party's private key, say x_A, to compute a value $z_{AB} = y_B^{x_A}$ (the exponentiation being performed in some group). Other aspects of Diffie-Hellman such as how keys are derived from the value z_{AB} or how the public/private key pairs are managed, are more properly parts of a scheme or protocol. The primitive isolates the basic mathematical part.

Secret value derivation primitives in P1363 include the following:

- DLSVDP-DH, basic Diffie-Hellman [9]
- DLSVDP-DHC, Diffie-Hellman with cofactor multiplication (see [15]), which protects against certain chosen-public-key attacks [20,17]
- DLSVDP-MQV, Menezes-Qu-Vanstone secret value derivation, involving two key pairs per party [15]
- DLSVDP-MQV, MQV with cofactor multiplication
- ECSVDP-DH, ECSDVP-DHC, ECSVDP-MQV, and ECSVDP-MQV, the elliptic curve analogs of the preceding primitives

Only the DL and EC families have secret value derivation primitives, an advantage of having common domain parameters to be shared among parties.

Signature and verification. A *signature primitive* (SP) processes a message representative (an input that contains information about a message, such

Family \ Type	SVDP	SP / VP	EP / DP
DL	DH, DHC, MQV, MQVC	NR, DSA	—
EC	DH, DHC, MQV, MQVC	NR, DSA	—
IF	—	RSA1, RSA2, RW	RSA

Table 4. Primitives in IEEE P1363, by family and type.

as the hash of the message) with a signer's private key to produce a signature. A corresponding *verification primitive* (VP) processes the signature with the signer's public key to recover the message representative, or processes the signature and the message representative to verify the signature. (In the former case, the primitive is said to have a *message recovery capability.*)

Signature and verification primitives in P1363 include:

- DLSP-NR / DLVP-NR, Nyberg-Rueppel signatures [23]; these have a message recovery capability
- DLSP-DSA / DLVP-DSA, generalizations of the NIST FIPS 186 Digital Signature Algorithm [22]
- ECSSP-NR / ECVP-NR and ECSP-DSA / ECVP-DSA, the elliptic curve analogs of the preceding primitives
- IFSP-RSA1 / IFVP-RSA1, basic RSA [28]
- IFSP-RSA2 / IFVP-RSA2, basic RSA with an "absolute value" step that saves one bit, as in ISO/IEC 9796 and ANSI X9.31
- IFSP-RW / IFVP-RW, Rabin-Williams signatures [26,31] with the one-bit savings

Encryption and decryption. An *encryption primitive* (EP) processes a message representative with a recipient's public key to produce a ciphertext. A corresponding *decryption primitive* processes the ciphertext with the recipient's private key to recover the message representative.

There is just one pair of encryption and decryption primitives:

- IFEP-RSA / IFDP-RSA, basic RSA

Encryption in the other families is typically based on secret value derivation primitives, so only the latter type of primitive need be defined for the DL and EC families.

To summarize, Table 4 lists the primitives according to family and type.

Examples DLSP-DSA generates a signature (r, s) from a message representative m with a private key (p, q, g, x). (The meaning of the individual items is not significant to this discussion — but the notation differs from P1363 to be more consistent with the original DSA specification [22].) DLSP-DSA computes the signature (r, s) as

$$r \leftarrow (g^k \bmod p) \bmod q$$
$$s \leftarrow k^{-1}(m + xr) \bmod q$$

where $(k, g^k \bmod p)$ is a freshly generated DL key pair. DLVP-DSA verifies the signature by computing

$$u_1 \leftarrow ms^{-1} \bmod q$$
$$u_2 \leftarrow rs^{-1} \bmod q$$

and then comparing

$$r \stackrel{?}{=} (g^{u_1} y^{u_2} \bmod p) \bmod q.$$

(Some testing for nonzero values is also included in the primitives.) The EC/DLSSA signature scheme is built from these primitives.

Other primitives in P1363 have a similar flavor, consisting of modular arithmetic and other group operations.

Implementation Primitives are likely to be implemented as low-level components of a system, for instance as functions interfacing to a cryptographic accelerator in a smart card. They are generally not directly accessible to applications, particularly since "raw" access to a primitive may provide a means of compromising a private key. Moreover, because a primitive is a mathematical operation, it may have properties that lead to potential attack if the primitive is employed directly to protect data. In addition, a primitive, being a basic operation, is limited in terms of the size of messages it can process. Because of the mathematical properties and the message size limitation, a primitive needs to be combined with other techniques in a scheme, as described next.

3.2 Schemes

A *scheme* is a set of related operations combining one or more primitives with additional techniques to enhance security and, possibly, to handle messages of arbitrary size. A scheme is intended to be secure for all messages it processes.

Types of Scheme Four types of scheme are defined in P1363. Each has one or two related operations, in addition to key management operations mentioned further below.

Key agreement. A *key agreement scheme* (KAS) includes a *key agreement operation* by which two parties can agree on a shared secret key. The key agreement operation typically combines a secret value derivation primitive with a *key derivation function*, where the key derivation function maps shared secret values produced by the primitive to one or more shared secret keys.

Key agreement schemes in P1363 include DL/ECKAS-DH1, based on Diffie-Hellman with one key pair per party; DL/ECKAS-DH2, with two key pairs per party (see [5] for some security analysis); and DL/ECKAS-MQV, based on MQV. Similar to the situation with primitives, only the DL and EC families have a key agreement scheme. Key agreement protocols can be defined for any

of the families, however, building on an encryption scheme in the case of the IF family.

SIGNATURE. A *signature scheme* includes a *signature generation operation* and a *signature verification operation* by which parties can verify the origin and integrity of a message. There are two flavors. In a *signature scheme with appendix* (SSA), a signature is provided to a verifier separate from the message. In a *signature scheme with message recovery* (SSR), the message is recovered from the signature. The operations combine signature and verification primitives with an *encoding method for signatures*, where the encoding method maps between arbitrary length messages and message representatives. Examples of encoding methods include a hash function and a hash function with padding.

Signature schemes in P1363 include DL/ECSSA, IFSSA, and IFSSR, each a general signature scheme combining primitives in the families with an encoding method. Signature schemes with message recovery for the DL and EC families are the subject of further work.

ENCRYPTION. An *encryption scheme* (ES) includes an *encryption operation* and a *decryption operation* by which parties can protect a message from disclosure. An authenticated encryption scheme can also verify the integrity of a message and can bind it to certain non-secret "control information" (see [13] for discussion). The operations combine encryption and decryption primitives with an *encoding message for encryption*.

There is just one encryption scheme in P1363, IFES, based on RSA. Encryption schemes for other families are the subject of further work.

The selection of encoding methods and key derivation functions is a delicate matter, as these additional techniques must address mathematical properties of the primitives and also be internally secure. For instance, an encoding method for signatures must produce a message representative in a way that overcomes any mathematical properties of the signature primitive. It must also be difficult to find two messages with the same message representative, or a messages with a given message representative. The internal properties are the ones most often studied for such encoding methods, but the mathematical considerations are also an appropriate area for research. (In fact, both are addressed together in the most recent encoding methods, such as OAEP [2] and PSS [3].)

Key management operations for the various schemes include key generation, key validation (terminology proposed by Don Johnson in a contribution to ANSI X9F1), and, depending on the family, domain parameter generation and domain parameter validation, where domain parameters are components common to a set of key pairs, such as an elliptic curve group in the EC family or a prime in the DL family. The validation operations, which are optional in P1363, are for verifying that a public key or a set of domain parameters satisfies its definition. The key management operations are complementary to the other operations in the schemes in the sense that they produce (and optionally verify) the keys that are input to the related scheme operations. (How parties obtain one another's public keys is a separate matter.)

Table 5 summarizes the schemes according to family and type.

Family \ Type	KAS	SSA	SSR	ES
DL/EC	DH1, DH2, MQV	DSA, NR	*open*	*open*
IF	—	RSA, RW	RSA, RW	RSA

Table 5. Schemes in IEEE P1363, by family and type.

Example An example scheme, building on the DSA primitive from the previous discussion, is DL/ECSSA, a signature scheme with appendix for the DL and EC families. DL/ECSSA is "generic" in that it can be based on any pair of DL and EC signature and verification primitives and any encoding method consistent with the primitives. It has six operations: the four key management operations, the signature generation operation, and the signature verification operation. The latter two are described here.

DL/ECSSA signature generation generates a signature (r, s) from a message M with a private key S. (For DLSP-DSA the private key would have the form (p, q, g, x) as above, though again meaning of the individual items is not significant to this discussion.) The operation computes the signature (r, s) by the following steps:

1. Apply the message encoding method to compute a message representative from the message: $m = \text{ENCODE}(M)$.
2. Apply the signature primitive to the message representative and the private key to produce a signature: $(r, s) = \text{DLSP-DSA}(S, M)$.

DL/ECSSA signature verification verifies the signature with a public key V by these steps (for a primitive such as DLVP-NR with a message recovery capability the steps would be somewhat different):

1. Apply the message encoding method to compute a message representative from the message: $m = \text{ENCODE}(M)$.
2. Apply the verification primitive to the message representative, the signature, and the public key to verify the signature: $\text{DLVP-DSA}(V, M, (r, s))$.

Implementation Scheme operations might be found as "mid-level" components, such as modules in a cryptographic service provider or library. They will typically be directly accessible to applications, in contrast to primitives. A sequence of scheme operations can then be carried out by an application, along with other message processing, in the form of a key establishment, entity authentication, or other security protocol.

4 Are "Strong Primes" Needed for RSA?

Standards development can place new requirements on existing cryptographic systems, challenging assumptions about what is necessary for security. An excellent example is found in the ongoing debate about whether so-called "strong primes" are needed for the RSA public-key cryptosystem.

4.1 1980s: Yes

The security of the RSA public-key cryptosystem depends, in part, upon the difficult of factoring large integers that are the product of two primes. A number of methods are available for solving this problem of integer factorization. Some are "general purpose" in that they apply equally well to all integers of a given size. Others are "special purpose," operating more effectively when the integer or its factors have a certain form.

One special-purpose method of particular interest is Pollard's $P-1$ Method [25]. This method can factor an RSA modulus $n = pq$ in about r operations where r is the largest prime factor of $p-1$. Because of this, it was recommended in the 1980s that a modulus be constructed so that the largest prime factors of $p-1$ and $q-1$ are large, say at least 100 bits long. Other special-purpose methods lead to other sets of strong prime conditions, such as "$P+1$" conditions and "$r-1$" conditions (r being the large factor of $P-1$). "Strong primes" are primes satisfying one or more of these conditions.

One standard developed during the 1980s, ITU-T (then CCITT) Recommendation X.509 (1988) [7], includes a number of these conditions in its description of RSA key generation. Strong primes are easy to generate: Gordon [11] gives a method for generating strong primes with only a small overhead compared to generation of random primes.

4.2 Early 1990s: No

Although strong primes were easy to generate and protected against certain attacks, were they necessary? This was the subject of an unpublished paper by Rivest in the early 1990s [27] (see also [29]). To resist general-purpose methods, the paper argued, the prime factors of an RSA modulus would need to be reasonably large. If the primes were sufficiently large and were generated at random, the paper continued, the primes would with high probability resist the various special-purpose methods as well, so strong prime conditions would add no protection in practice. At most, they gave a false sense of security.

This point became particularly clear with the development of the Generalized Number Field Sieve (GNFS) [6], which by increasing the required size of RSA moduli to resist a certain level of attack, made the special-purpose methods even less relevant.

The development that perhaps most convincingly argued against the need for strong primes (and for the need for large ones) was the Elliptic Curve Method (ECM) [16]. ECM, unlike the $P-1$ Method and other previous special-purpose methods, is equally effective on all primes of a given size. No special conditions on a prime, other than size, can defend against it. Thus, in a certain sense, every prime of a given size is a "weak prime" — including even primes strong against the $P-1$, $P+1$ and all other previous methods. Of course, primes large enough to resist GNFS will resist ECM as well.

By the mid-1990s, then, it seemed that standards for the RSA public-key cryptosystem should no longer include conditions on the primes, other than that

they be sufficiently large and random. Implementations should be able to include such conditions during RSA key generation, but to impose a general requirement no longer seemed necessary. The debate about whether an RSA modulus could be "weak" appeared settled.

4.3 Late 1990s: Maybe?

Another debate, however, was just beginning. Although a large random prime would resist attacks against outside opponents, what if a user deliberately generated a prime that was not random or not large enough? Indeed, what if the user deliberately generated a prime that was weak against the $P-1$ method? The user could do so by repeatedly generating RSA key pairs until one of the primes output as part of the RSA private key was obviously weak. (To detect the weakness, the user need only try to factor $p-1$ or $q-1$, perhaps by ECM.) A user might be motivated to do this if the user later could claim that, because the prime was weak, the resulting RSA modulus could easily be factored. The user could thereby attempt to repudiate a previously verified signature.

On the one hand, it was argued that a user would have a difficult time convincing a judge that the supposed weakness was the result of chance. Since it is unlikely that a random prime would be weak against the $P-1$ method, the claim would seem suspicious, particularly as to why an opponent would choose this one RSA modulus to factor with the $P-1$ method without knowing in advance whether the effort would succeed. (Although the user could know in advance whether a modulus could be factored by the $P-1$ method, there is no way for an outsider to determine this without actually trying to factor it.)

On the other hand, it was pointed out that the mere possibility that such a ruse might succeed was sufficient justification to prevent it.

In any case, a general consensus was emerging by the late 1990s that it was important to consider not only security against outside opponents, but security against insiders — the users — when constructing requirements for key generation.

This debate, which played out in the final stages of the development of ANSI X9.31, makes it clear that assumptions about what is necessary for security are continually evolving. (The verdict: ANSI X9.31 would require strong prime conditions.) It also raises some nice research problems about how users can prove their keys are properly generated, a topic which is considered further in the next section.

5 Research Areas

As already established, research in cryptography eventually finds its way into standards, though perhaps not necessarily as originally intended. Standards development likewise motivates additional research.

As an example, consider Table 5. As part of standards development, it became clear that certain types of techniques were better established in one family than

in another. This provided motivation for finding new techniques in the other families. Or consider the strong primes debate. The question about whether strong primes were necessary, cast in the new light of nonrepudiation, raised issues about proving that public keys satisfy certain properties. Most research is influenced in one way or another by application requirements, and standards, by defining a class of applications, thus have a significant impact on new research.

(As a side note, perhaps the most compelling example of how standards development can influence research can be found in the significant body of research surrounding the analysis of the Data Encryption Standard [21] and the development it is successor, the Advanced Encryption Standard.)

As it would have been difficult to cover all the standards involving public-key cryptography, so it is difficult to cover all the research problems motivated by those standards. However, four research areas have been particularly prominent in the development of P1363 and its addendum, P1363a, and these are described in further detail next.

5.1 Key Validation

As discussed above in the context of the strong primes issue, it can be important to have assurance that a public/private key pair has certain properties. More fundamentally, it can often be important to know that a given public key is, in fact, a public key.

There is an interesting definitional issue here. When specifying a cryptographic primitive, one usually assumes that public keys are valid; as validation may be expensive and can be performed elsewhere, there is little reason to specify the behavior of a primitive on an invalid public key. When specifying a protocol, however, one can no longer make this assumption. Thus, the definition of "public key" varies according to the type of technique. The transition between different types of technique, say a protocol and a primitive, can introduce potential security risks due to misunderstanding about whether a key is valid or not.

Public-key validation is primarily of interest in key agreement schemes, which combine one user's public key with another user's private key in a secret key derivation primitive. Effectively, this combination can open the door to a "chosen-public-key attack" where an opponent, by supplying an invalid public key, may be able to extract information about a private key. (The "small subgroup" attacks on the Diffie-Hellman and related primitives observed by Vanstone [20] and by Lim and Lee [17] illustrate the risks involved.) Key validation is one of the countermeasures to these concerns.

In encryption and signature schemes, public-key validation provides an additional level of assurance, but is less important than for key agreement schemes since there is no direct counterpart to the "chosen-public-key attack." (Chosen-message and chosen-ciphertext attacks are of greater concern.) One example of added assurance is that public-key validation can defend against the possibility that a user might repudiate a signature on the basis that the user's public key is invalid.

As one area for research, then, it would be worthwhile to refine existing security models to accommodate the possibility that public keys might be invalid. Such models could account for the possibility that a user might repudiate a signature, and issues such as when key validation is necessary could be addressed.

A user may perform key validation directly, or a perhaps rely on a certificate authority to perform key validation as part of issuing a certificate. The particular validation method depends on the type of key. For DL and EC public keys, validation involves a straightforward check that the public key satisfies its definition — that is, that the public key has the correct order in an intended group. This assumes, of course, that the correct order and the intended group, which are part of the DL or EC domain parameters, have also been validated, a process that can be carried out separately.

An alternative to a direct check of a public key's validity is an interactive proof of knowledge of the corresponding private key such as the one given by Chaum et al. [8].

For IF public keys, validation is more difficult. (As mentioned above, however, the need for validation of IF public keys is less pronounced, since there is no direct "chosen-public-key" attack.) No method is known, for instance, by which a user can check whether an RSA modulus is a product of two primes of similar size. A user can check whether a modulus is composite, of course, but to verify the number of primes involved appears to require an interactive proof with the holder of the prime factors such as the one given by van de Graaf and Peralta [30].

Recently, several techniques have been developed for proving additional properties about IF public keys. Liskov and Silverman give an interactive proof for the size of the prime factors [18]; Mao presents an alternate proof [19]. The proof given by Gennaro, Micciancio and Rabin [10] shows that there are two primes involved, each occurring exactly once as a factor for a certain class of moduli. The techniques can likely be improved, and further research on this problem is well motivated.

5.2 New Encryption Schemes

Another area of research interest concerns improvements to encryption schemes. In P1363, there is only one encryption scheme, IFES, based on the RSA encryption primitive. Schemes for the DL and EC families were not included since there were no established techniques in practice, and since it was possible to establish keys for conventional encryption schemes through the use of the DL and EC key agreement schemes.

Related to the broadening of encryption schemes to include the other families is the broadening of the schemes to include potentially larger messages. IFES, as defined, limits the size of the message it can encrypt to slightly less than the size of the RSA modulus. This is generally not a problem in practice as the RSA modulus is typically 96 bytes or more and the message is typically a symmetric key of 16 bytes or less, though further flexibility would be helpful.

A straightforward approach to EC encryption, however, would combine a 16-byte key with a secret value that is (say) 20 bytes long. This leaves little room for padding and other enhancements that may be necessary for security, and motivates further research on how to construct DL and EC schemes.

New and better encryption schemes for all three families were thus identified as a research objective during the development of P1363, and several contributions resulted that are now being considered for inclusion in P1363a (a full list of contributions can be found through the P1363 Web page). As this is still a relatively new area of research, further review and additional contributions are definitely welcome.

5.3 New Signature Schemes

The situation with signature schemes in P1363 is somewhat more complete than with encryption schemes, as there is at least one scheme for each family. However, here as well there is need for additional research, as only one of the families has a signature scheme giving message recovery, and as the latest results on "provable security" (e.g. [24]) have not yet been incorporated. In addition, the one signature scheme with message recovery, IFSSR, has a relatively older design. More recent schemes, such as PSS [3], have better security proofs.

For the DL and EC families, signature schemes with message recovery could be based on the Nyberg-Rueppel signature primitive, since it supports message recovery. A challenge here is that the verification primitive (DLVP-NR or ECVP-NR) can only recover a relatively small message — typically 20 bytes. Any redundancy necessary for security would further limit the size of the message.

The recent discussion on "target-collision-resistant" hash functions [4] can also provide insight into the appropriate design of new signature schemes.

5.4 Provable Security

"Provable" security, of course, remains a continual objective — whether a better understanding of the complexity of an underlying hard problem or an assurance of the connection between that hard problem and a particular cryptosystem. Proofs for primitives, schemes, and protocols are all important; the last of the three is perhaps the most important in practice, since it is through actual protocols that parties (including opponents) most often interact with one another. Since proofs of protocol security depend on security of the underlying schemes and primitives, however, security analysis for the other two levels is important as well.

The random oracle model [1] has provided significant insight into the design and security proof of schemes, but it has limitations, namely that in practice, the random oracle in the construction is instantiated with a particular method such as a hash function. Security proofs in the random oracle model generally contemplate a generic attack that works for any instantiation. In practice, one would like assurance about specific attacks involving a particular hash function as well (although, certainly, the absence of a generic attack is itself quite assuring).

As an example, one might ask how security results about RSA bits [12] apply to the OAEP construction [2]. Further research into "instantiated security" is thus another desirable research topic.

6 Conclusion

With all the standards development around public-key cryptography, it is clear that the technology has matured significantly, but story is far from over. Improvements to existing techniques, new techniques, and perhaps even completely different approaches are to be expected.

A lesson learned for future development is the importance of collaboration between research and standards. Inasmuch as standards are "best practice," they are an excellent avenue for applying research, and their continued success depends on ongoing research. Basic research in cryptography and the development of standards are thus quite closely related. Though in the past the efforts have been separated by a decade or more, hopefully, in the future, they will proceed more closely in step, as the promising results of additional knowledge continue to be made available for everyday use.

Acknowledgments

My thanks to Ivan Damgard for inviting me to give a lecture on this subject at the Summer School in Cryptology & Data Security, and to the Basic Research in Computer Science (BRICS) program for its support. As is my custom, I also thank God (Col. 3:17).

References

1. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73, ACM Press, 1993. 101
2. M. Bellare and P. Rogaway. Optimal asymmetric encryption — How to encrypt with RSA. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT '94*, pages 92–111, Springer, 1994. 95, 102
3. M. Bellare and P. Rogaway. The exact security of digital signatures — How to sign with RSA and Rabin. In U.M. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 399–416, Springer, 1996. 95, 101
4. M. Bellare and P. Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In B.S. Kaliski Jr., editor, *Advances in Cryptology — Crypto '97*, pages 470–484, Springer, 1997. 101
5. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *Cryptography and Coding: Sixth IMA International Conference*, pages 30–45, Springer, 1997. 94
6. J.P. Buhler, H.W. Lenstra Jr., and C. Pomerance. Factoring integers with the number field sieve. In A.K. Lenstra and H.W. Lenstra Jr., editors, *The Development of the Number Field Sieve*, pages 50–94, Springer, 1993. 97

7. *CCITT Recommendation X.509: The Directory — Authentication Framework*. CCITT, 1988. 97
8. D. Chaum, J.-H. Evertse, J. van de Graaf, and R. Peralta. Demonstrating possession of a discrete logarithm without revealing it. In A.M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, pages 200–212, Springer, 1987. 100
9. W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976. 92
10. R. Gennaro, D. Micciancio, and T. Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. To appear, *Proceedings of the Fifth ACM Conference on Computer and Communications Security (CCS-5)*, ACM Press, 1998. 100
11. J. Gordon. Strong RSA keys. *Electronics Letters* 20:514–546, June 7, 1984. 97
12. J. Håstad and M. Näslund. The security of individual RSA bits. To appear, *Proceedings of the 39th IEEE Computer Society Conference on Foundations of Computer Science (FOCS '98)*, IEEE Computer Society, 1998. 102
13. D. B. Johnson and S. M. Matyas. Asymmetric encryption: Evolution and enhancements. *RSA Laboratories' CryptoBytes*, 2(1):1,3–6, Spring 1996. 95
14. B. Kaliski. A survey of encryption standards. *IEEE Micro*, 74–81, December 1993. 88
15. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. *An Efficient Protocol for Authenticated Key Agreement*. Technical Report CORR 98-05, Dept. of C&O, University of Waterloo, Canada, March 1998 (revised August 28, 1998). 92
16. H.W. Lenstra Jr. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987. 97
17. C.H. Lim and P.J. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In B.S. Kaliski Jr., editor, *Advances in Cryptology — CRYPTO '97*, pages 249–263, Springer, 1997. 92, 99
18. M. Liskov and R.D. Silverman. A statistical limited-knowledge proof for secure RSA keys. Manuscript, 1998. 100
19. W. Mao. Verifiable partial sharing of the factors of an integer. To appear, *Proceedings of Selected Areas in Cryptography (SAC) '98*, Springer. 100
20. A. Menezes, M. Qu, and S. Vanstone. Key agreement and the need for authentication. Presented at *Public Key Solutions '95*, Toronto, Canada, November 1995. 92, 99
21. *Federal Information Processing Standard (FIPS) Publication 46-2: Data Encryption Standard*. National Institute of Standards and Technology (NIST), U.S. Department of Commerce, December 30, 1993. 99
22. *Federal Information Processing Standard (FIPS) Publication 186: Digital Signature Standard*. National Institute of Standards and Technology, U.S. Department of Commerce, 1994. 93
23. K. Nyberg and R. Rueppel. A new signature scheme based on DSA giving message recovery. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 58–61, ACM Press, 1993. 93
24. D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 387–398, Springer, 1996. 101
25. J.M. Pollard. Theorems on factorization and primality testing. *Proceedings of the Cambridge Philosophical Society*, 76:521–528, 1974. 97
26. M.O. Rabin. *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979. 93

27. R.L. Rivest. Are 'strong' primes needed for RSA? Manuscript, 1991. 97
28. R.L. Rivest, A. Shamir and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2), pages 120–126, February 1978. 93
29. R.D. Silverman. Fast generation of random, strong RSA primes. *RSA Laboratories' CryptoBytes*, 3(1):9–12, Spring 1997. 97
30. J. van de Graaf and R. Peralta. A simple and secure way to show the validity of your public key. In C. Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, pages 128–134, 1988. 100
31. H.C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, 26:726–729, 1980. 93

Contemporary Block Ciphers

Lars R. Knudsen

University of Bergen, Department of Informatics, Hi-techcenter
N-5020 Bergen, Norway

Abstract. This paper considers modern secret-key block ciphers. The theory behind the design and analysis of modern block ciphers is explained, and the most important known attacks are outlined. Finally the Advanced Encryption Standard is discussed.

1 Block Ciphers - Introduction

In the last few thousands of years encryption algorithms, also called ciphers, have been developed and used [18,28]. Many of the old ciphers are much too weak to be used in applications today because of the tremendous progress in computer technology. There are essentially two types of cryptosystems, one-key and two-key ciphers. In one-key ciphers the encryption of a plaintext and the decryption of the corresponding ciphertext is performed using the same key. Until 1976 when Diffie and Hellman introduced *public-key* or two-key cryptography [20] all ciphers were one-key systems, today called conventional or classical cryptosystems. Conventional cryptosystems are widely used throughout the world today, and new systems are published frequently. There are two kinds of one-key ciphers, stream ciphers and block ciphers. In stream ciphers, typically a long sequence of bits is generated from a short string of key bits, and is then added bitwise modulo 2 to the plaintext to produce the ciphertext. In block ciphers the plaintext is divided into blocks of a fixed length, which are then encrypted into blocks of ciphertexts using the same key. The interested reader will find a comprehensive treatment of early cryptology in [28].

A block cipher is called an *iterated cipher* if the ciphertext is computed by iteratively applying a round function several times to the plaintext. In each round a round key is combined with the text input. In other words, let G be a function taking two arguments, such that, it is invertible when the first argument is fixed. Then define

$$C_i = G(K_i, C_{i-1}),$$

where C_0 is the plaintext, K_i is the ith round key, and C_r is the ciphertext. A special kind of iterated ciphers are the *Feistel* ciphers. A Feistel cipher with block size $2n$ and r rounds is defined as follows. Let C_0^L and C_0^R be the left and right halves of the plaintext, respectively, each of n bits. The round function G operates as follows

$$C_i^L = C_{i-1}^R$$
$$C_i^R = F(K_i, C_{i-1}^R) + C_{i-1}^L,$$

I. Damgård (Ed.): Lectures on Data Security, LNCS 1561, pp. 105–126, 1999.

and the ciphertext is the concatenation of C_r^R and C_r^L. Note that F can be any function taking as arguments an n-bit text and a round key K_i and producing n bits. '+' is a commutative group operation on the set of n bit blocks. For the remainder of this paper we will assume that '+' is the exclusive-or operation ($\oplus$).

The Data Encryption Standard (DES) [55] is by far the most widely used iterated block cipher today. Around the world, governments, banks, and standards organisations have made the DES the basis of secure and authentic communication [65]. The DES is a Feistel cipher. However, the key size and the block size of the DES have become too small. Therefore the National Institute of Standards and Technology (NIST) in the U.S.A. has initiated the process of developing and to standardise a new encryption algorithm, the *Advanced Encryption Standard (AES)* [57], as a replacement for DES. This work is ongoing as this paper is written.

The remainder of this paper is organised as follows. § 2 lists and discusses the modes of operation for block ciphers used for encryption. § 3 discusses the theoretical and practical security of block ciphers. The most important methods of cryptanalysing block ciphers are given in § 4. § 5 discusses design principles of block ciphers and §6 reviews how to strengthen the DES. In §7 the Advanced Encryption Standard is discussed and some conjectures are made, and § 8 contains concluding remarks.

2 Modes of Operations

The most obvious and widespread use of a block cipher is for encryption. In 1980 a list of four modes of operation for the DES was published [56]. These four modes can be used with any block cipher and seem to cover most applications of block ciphers used for encryption [18]. In the following let $E_K(\cdot)$ be the permutation induced by using the block cipher E of block length n with the key K and let $P_1, P_2, ..., P_i, ...$ be the blocks of plaintexts to be encrypted. The Electronic Code Book (ECB) is the native mode, where one block at a time is encrypted independently of the encryptions of other blocks, $C_i = E_K(P_i)$, $P_i = E_K(C_i)$. In the Cipher Block Chaining (CBC) mode the encryption of a block depends on the encryptions of previous blocks. $C_i = E_K(P_i \oplus C_{i-1})$, $P_i = D_K(C_i) \oplus C_{i-1}$, where C_0 is a chosen initial value. The Cipher Feedback (CFB) mode is a stream cipher mode, where one m-bit character at a time is encrypted.

$$C_i = P_i \oplus \mathrm{MSB}_m(E_K(X_i))$$
$$X_{i+1} = \mathrm{LSB}_{n-m}(X_i) \parallel C_i$$

where X_1 is a chosen initial value, $\parallel$ denotes concatenation of blocks, MSB_s and LSB_s denote the s most and least significant bits respectively or equivalently the leftmost and rightmost bits respectively. Decryption is similar to encryption. Here m can be any number between 1 and the block length of the cipher. If the plaintext consists of characters, $m = 7$ or $m = 8$ is usually the well-chosen

parameter. The Output Feedback (OFB) mode is a second stream mode, where the stream bits are not dependent on the previous plaintexts, that is, only the stream bits are fed back, not the ciphertext as in CFB mode.

$$C_i = P_i \oplus \mathrm{MSB}_m(E_K(X_i))$$
$$X_{i+1} = \mathrm{LSB}_{n-m}(X_i) \parallel \mathrm{MSB}_m(E_K(X_i))$$

where X_1 is a chosen initial value. Decryption is equal to encryption. Both the CFB and OFB modes have two parameters, the size of the plaintext block and the size of the feedback value. In the above definition we have chosen them equal and will do so also in the following.

The ECB is the native mode, well-suited for encryption of keys of fixed length. It is not suited for the encryption of larger plaintexts, since equal blocks are encrypted into equal blocks. To avoid this, the CBC mode is recommended. Not only does a current ciphertext block depend on the current plaintext but also on all previous ciphertext blocks. In some applications there is a need for encryptions of characters, instead of whole blocks, e.g., the 8 bytes for the CBC mode of DES. For that purpose the CFB and OFB modes are suitable. It is often recommended to use the OFB mode only with full feedback, i.e., with $m = n$ (64 for the DES). It comes from the fact, that for $m < n$ the feedback function is not one-to-one, and therefore has a relatively short cycle [18] of length about $2^{n/2}$.

An important issue in the applications of the four modes is how an error in the transmission of ciphertexts is propagated. In the ECB mode an error in a ciphertext block affects only one plaintext block. A lost ciphertext block results in a lost plaintext block. An error in a ciphertext block in the CBC mode affects two plaintexts blocks. As an example, assume that ciphertext C_3 has an error and that all other ciphertext blocks are error-free, then $P_4 = D_K(C_4) \oplus C_3$ inherits the error from C_3 and $P_3 = E_K(C_3) \oplus C_2$ will be completely garbled. Here we assume that even a small change in the input to the block cipher will produce a randomly looking output. All other plaintexts will be decrypted correctly. A lost ciphertext block results in a lost plaintext block and an error in one addition plaintext block after which the mode synchronises itself. In the CFB mode an error in a ciphertext block C_i will be inherited by the corresponding plaintext block P_i, and moreover since X_{i+1} contains the garbled C_i the subsequent plaintexts blocks will be garbled until the X value is free of C_i, i.e., when C_i has been shifted out. In other words in CFB mode with m-bit ciphertexts, at most $n/m + 1$ plaintext blocks will be garbled. The case of lost ciphertext blocks is similar to that of the CBC mode. In the OFB mode, since the feedback is independent of the plaintexts and ciphertexts, a transmission error in a ciphertext block garbles only the corresponding plaintext block and is not propagated to other plaintext blocks. On the other hand, a lost ciphertext block will result in an infinite error propagation.

3 Security of Secret-Key Block Ciphers

When discussing the security of cryptographic systems one needs to define a model of the reality. We will use the model of Shannon [64]. The sender and the receiver share a common key K, which has been transmitted over a secure channel. The sender encrypts a plaintext P using the secret key K, sends C over an insecure channel to the receiver, who restores C into P using K. The attacker has access to the insecure channel and can intercept the ciphertexts (cryptograms) sent from the sender to the receiver. In this section we assume that the legitimate sender and receiver use a secret-key cipher $E_K(\cdot)$ of block size n (bits), where the key K is of size k. To avoid an attacker to speculate in how the legitimate parties have constructed their common key, the following assumption is made.

Assumption 1. *All keys are equally likely and a key K is always chosen uniformly random.*

Also we will assume that all details about the cryptographic algorithm used by the sender and receiver are known to the attacker, except for the secret key. This assumption is known as Kerckhoffs's Assumption [28].

Assumption 2. *The enemy cryptanalyst knows all details of the enciphering process and deciphering process except for the value of the secret key.*

For a fixed key, a block cipher is a permutation. There are totally 2^{n2^n} possible n-bit permutations. Thus, it would require $k = n2^n$ bits to specify all of them. With a block size of 64 bits or more this is a huge number. In a practical block cipher, the key size is much smaller, typically $k = 128$ or $k = 256$. A block cipher (system) with a k-bit key and blocks of n bits can be seen as an algorithm of how to select and specify 2^k of all 2^{n2^n} n-bit permutations.

3.1 Classification of Attacks

The possible attacks an attacker can do are classified as follows.

- *Ciphertext-only attack.* The attacker has obtained a set of intercepted ciphertexts.
- *Known plaintext attack.* The attacker obtains $P_1, P_2, ..., P_s$ a set of s plaintexts and the corresponding ciphertexts $C_1, C_2, ..., C_s$.
- *Chosen plaintext attack.* The attacker chooses *a priori* a set of s plaintexts $P_1, P_2, ..., P_s$ and obtains in some way the corresponding ciphertexts $C_1, C_2, ..., C_s$.
- *Adaptively chosen plaintext attack.* The attacker chooses a set of plaintexts $P_1, P_2, ..., P_s$ interactively as he obtains the corresponding ciphertexts $C_1, C_2, ..., C_s$. That is, the attacker chooses P_1, obtains C_1, **then** chooses P_2 etc.

– *Chosen ciphertext attacks.* For symmetric ciphers these are similar to those of chosen plaintext attack and adaptively chosen plaintext attack, where the roles of plain- and ciphertexts are interchanged.

Also, one can consider any combination of the above attacks. The chosen text attacks are obviously the most powerful attacks. In many applications they are however also unrealistic attacks. If the plaintext space contains redundancy, it will be hard for an attacker to 'trick' a legitimate sender into encrypting non-meaningful plaintexts and similarly hard to get ciphertexts decrypted, which do not yield meaningful plaintexts. But if a system is secure against an adaptively chosen plaintext/ciphertext attack then it is also secure against all other attacks. An ideal situation for a designer would be to prove that her system is secure against an adaptively chosen text attack, although an attacker may never be able to mount more than a ciphertext only attack.

3.2 Theoretical Secrecy

In his milestone paper from 1949 [64] Shannon defines perfect secrecy for secret-key systems and shows that they exist. Shannon's theory is described in many text books and here only a few of his results are stated. A secret-key cipher is *perfect* if for all P and all C it holds that $\Pr(P) = \Pr(P|C)$ [64]. In other words, a ciphertext C gives no information about the plaintext. This definition leads to the following result.

Corollary 1. *A perfect cipher is unconditionally secure against a ciphertext-only attack.*

As noted by Shannon the Vernam cipher, also called the *one-time pad*, is a perfect secret-key cipher. In the one-time pad the plaintext characters are exclusive-ored with independent key characters to produce the ciphertexts. However, the practical applications of perfect secret-key ciphers are limited, since it requires as many digits of secret key as there are digits to be enciphered [45]. Clearly, the above definition of a perfect cipher makes no sense when considering known or chosen plaintext attacks. A less stringent form of theoretical secrecy is possible, in terms of the *unicity distance*. It is the smallest integer s such that essentially only one value of the secret key K could have encrypted some plaintexts to the ciphertexts $C_1, ..., C_s$. The unicity distance depends on both the key size and on the redundancy in the plaintext space. Redundancy is an effect of the fact that certain plaintext characters appear more frequently than others. However, the unicity distance gives no indication of the computational difficulty in breaking a cipher, it is merely a *lower* bound on the amount of ciphertext blocks needed in a ciphertext-only attack. The concept of unicity distance can be adapted also to the known or chosen plaintext scenario. In these cases the redundancy of the plaintexts from the attacker's point of view is zero. Let k and n be the number of bits in the secret key respectively in the plaintexts and ciphertexts. If we assume that the keys are always chosen uniformly at random the unicity distance in a known or chosen plaintext attack is $\lceil k/n \rceil$.

3.3 Practical Secrecy

In the recent years cryptanalysis has been focused on finding the key K of a secret-key cipher. However, there are other serious attacks, which do not find the secret key. In the sequel Assumption 1 is used.

- *Total break.* An attacker finds the secret key K.
- *Global deduction.* An attacker finds an algorithm A, functionally equivalent to $E_K(\cdot)$ (or $D_K(\cdot)$) without knowing the key K.
- *Instance (local) deduction.* An attacker finds the plaintext (ciphertext) of an intercepted ciphertext (plaintext), which he did not obtain from the legitimate sender.
- *Information deduction.* An attacker gains some (Shannon) information about the secret key, the plaintexts or the ciphertexts, which he did not get directly from the sender and which he did not have before the attack.
- *Distinguishing algorithm.* An attacker is able to tell whether the attacked cipher is a randomly chosen permutation or one of the 2^k permutations specified by the secret key.

Clearly, this classification is hierarchical, that is, if a total break is possible, then a global deduction is possible and so on.

A global deduction is possible when a block cipher contains a "block structure". If certain subsets of the ciphertext are independent of certain subsets of the plaintext, then no matter how long the key is, the block cipher is vulnerable to a global deduction in a known plaintext attack. Also, in iterated block ciphers the round keys are sometimes generated in a one-way fashion [62,63,15,16]. So in attacks, which find the round keys, it may be impossible for the attacker to derive the actual value of the secret key, but at the same time the round keys enable the attacker to encrypt and decrypt. An instance deduction can be as dangerous as a total break, if the number of likely plaintexts is small. Consider the situation where the block cipher is used for encrypting a key in a key-exchange protocol. Here only one plaintext is encrypted and a total break is equal to an instance deduction. If the plaintext space is highly redundant an information deduction can be a serious problem. In general, the legitimate parties are often interested in that no information at all about the plaintexts and keys are obtained by any enemies. A distinguishing algorithm is the least serious attack. Let A be an attack (a distinguisher), which has access to a black box which is able to compute $E_K(\cdot)$ for K the secret key. When asked for the ciphertexts of plaintexts $P_1, \ldots, P_i$ the black box flips a coin whether to return $E_K(P_1), \ldots, E_K(P_i)$ or $\pi(P_1), \ldots, \pi(P_i)$ for a randomly chosen permutation π. The attack A has to decide whether the encryptions came from $E_K(\cdot)$ or π. The advantage of the attack is $\text{abs}(\Pr(A : \text{"it is } E_K(\cdot)\text{"} | E_K(\cdot) \text{ was used}) - \Pr(A : \text{"it is } E_K(\cdot)\text{"} | \pi \text{ was used}))$, that is, a number between 0 and 1. The higher the number the better the attacker's strategy.

In the following some trivial attacks applicable to all block ciphers are discussed. All block ciphers are totally breakable in a ciphertext-only attack, simply by trying all keys one by one and checking whether the computed plaintext is

meaningful, using only about N_{ud} ciphertext blocks, where N_{ud} is the unicity distance. This attack requires the computation of about 2^k encryptions. Also, there is the table look-up attack, where the attacker encrypts in a pre-computation phase a fixed plaintext P under all possible keys and sorts and stores all the ciphertexts. Thereafter the cipher is total breakable in a chosen plaintext attack requiring one chosen plaintext. There might be some keys encrypting P into the same ciphertext. Repeating the attack a few times with $P' \neq P$ will give a unique key. All block ciphers are globally/instance deducible under a known/chosen plaintext attack. Simply get and store all possible plaintext/ciphertext pairs. The running time of a deduction is the time of one table look-up.

The following result shows that a non-trivial information gain can be obtained when about the square root of all ciphertexts are available.

Theorem 1 ([34]). *Every n-bit block cipher used in the ECB, CBC or CFB mode is information deducible in a ciphertext-only attack with complexity about $2^{n/2}$.*

Note that the result of Theorem 1 is independent of the key size. This attack on CBC mode was named the *matching ciphertext attack* in [12]. Thus, it is recommended that a single key is used to encrypt at most $2^{n/2}$ ciphertext blocks.

Hellman [24] has presented a time-memory trade-off attack on any block cipher, which finds the secret key after $2^{2k/3}$ encryptions using $2^{2k/3}$ words of memory. The $2^{2k/3}$ words of memory are computed in a pre-processing phase, which takes the time of 2^k encryptions.

To estimate the complexity of a cryptanalytic attack one must consider at least the time it takes, the amount of data that is needed and the storage requirements. For an n-bit block cipher the following complexities should be considered. *Data complexity:* The amount of data needed as input to an attack. Units are measured in blocks of length n. Denote this complexity C_d. *Processing complexity:* The time needed to perform an attack. Time units are measured as the number of encryptions an attacker has to do himself. Denote this complexity C_p. *Storage complexity:* The words of memory needed to do the attack. Units are measured in blocks of length n. Denote this complexity C_s. As a rule of thumb, the complexity of an attack is taken to be the maximum of the three complexities, that is, $C_a = max(C_d, C_p, C_s)$. In general, there are some deviations from this rule and furthermore the three complexities are relative to the attacker. As an example, we may say that the above attack by Hellman on the DES has complexity $2^{2\times 56/3} \simeq 2^{38}$. Although the time of the pre-computation phase is 2^{56}, it is done only once after which any DES-key can be derived with a complexity of 2^{38}. On the other hand, the storage requirements may be unrealistic for most attackers, e.g., the attack on the DES will require about 1000 Gigabytes of memory.

4 Cryptanalysis of Block Ciphers

The history of cryptanalysis is long and at least as fascinating as the history of cryptography. As a single example, in 1917 in an article in "Scientific American"

the Vigenère cipher was claimed to be "impossible of translation" [19]. Today, it is an exercise in cryptology classes to illustrate that this claim is not true.

4.1 Attacks on Iterated Ciphers

In the following, P denotes the plaintext and C denotes the ciphertext. In most modern attacks on iterated ciphers, the attacker repeats his attack for all possible values of (a subset of) the bits in the last-round key. The idea is, that when he guesses the correct values of the bits of the key, he can compute bits of the ciphertexts after the second-last round, that is before the last round, whereas when he guesses wrongly, these bits will correspond to ciphertext bits encrypted with a wrong key. If there is a probabilistic correlation between the bits of the plaintexts, P, and the bits of the ciphertexts before the last round, $\tilde{C}$, denoted $\mathrm{cor}(P, \tilde{C})$, an attacker might be able to distinguish the correct guesses of the key in the last round from wrong guesses. If this is the case, the attacker can peel off one round of the cipher and do a similar attack on a cipher one round shorter to find the second-last round key etc. In some attacks it is advantageous to consider the first-round key instead of the last-round key or both at the same time, depending on the structure of the cipher, the number of key bits involved in each round etc. In iterated ciphers the correlation is often found by first identifying a correlation between inputs and outputs of the individual rounds and them combining them to a correlation over several rounds. The probability of this correlation can be calculated as the product of the probabilities of the individual round correlations, if they are independent. For most ciphers this independence is obtained by assuming that all round keys are independent. Although this is most often not the case, first of all, experiments have shown [6,34,49] that this leads to a good approximation to the real probability, secondly there seems to be no other way to compute the real probability. Denote by the *reduced cipher*, the cipher that one gets by removing the first and/or the final rounds of the original cipher. Let $\tilde{P}, \tilde{C}$ be the input bits and output bits respectively of the reduced cipher. Let $\tilde{K}$ be the key bits the attacker guesses in the attack (note that an attacker might not need to know all input and output bits of the reduced cipher). If the attacker guesses $\tilde{K}$ correctly, he can compute (bits of) $\tilde{P}, \tilde{C}$ from P, C. Let P', C' be the results the attacker obtains when he guesses $\tilde{K}$ wrongly. The probability of success of an iterated attack depends first of all on whether $\mathrm{cor}(\tilde{P}, \tilde{C})$ is different from $\mathrm{cor}(P', C')$, at least for some wrong guesses of $\tilde{K}$. In most attacks on iterated ciphers, an attacker repeats the basic attack a number of times and counts the values of $\tilde{K}$ which led to the expected $\mathrm{cor}(\tilde{P}, \tilde{C})$. Although some attacks in the literature do not have exactly this form, they can be translated into this general form (at least for illustration). A similar approach was taken in [67]. The signal-to-noise ratio (see [6] for the differential attack) is the expected number of times the correct guess of the key is counted over the expected number of times a wrong guess of the key is counted. Earlier it was believed that a necessary condition for the success of an iterated attack is that the signal-to-noise ratio is greater than one [6]. However, it was later discovered [60,9] that an attack can work in two ways: when $S/N > 1$ one looks for the

most suggested value of the key, and when $S/N < 1$ one looks for the least suggested value. Attacks where $S/N < 1$ are in principle as good as attacks where $S/N > 1$ but do not seem easier to find in general. In the following a number of iterated attacks are described. Since all of them have the above form, it suffices to describe how to detect and obtain the correlation of bits of the inputs and outputs of the reduced cipher.

4.2 Differential Cryptanalysis

The most general method of analysing conventional cryptosystems today is *differential cryptanalysis*, published by Biham and Shamir in 1990. The method has proved to be relatively efficient and has been applied to a wide range of iterated ciphers see e.g., [6,32]. Furthermore, it was the first attack which could (theoretically) recover DES keys in time less than the expected cost of exhaustive search [6,7]. In the following a brief description of differential cryptanalysis with respect to a general n-bit iterated cipher, cf., (1) is given.

First, one defines a *difference* between two bit strings, X and X' of equal length as

$$\Delta X = X \otimes (X')^{-1}, \tag{1}$$

where $\otimes$ is the group operation on the group of bit strings used to combine the key with the text input in the round function and where $(X)^{-1}$ is the inverse element of X with respect to $\otimes$. The idea behind this is, that the differences between the texts before and after the key is combined are equal, so the difference is independent of the key. In a strong encryption algorithm there will be some components which are non-linear in the $\otimes$-operation. In a differential attack one exploits that for certain input differences the distribution of output differences of the non-linear components is non-uniform.

Definition 1 (([6])). *An s-round* characteristic *is a series of differences defined as an $s+1$-tuple $\{\alpha_0, \alpha_1, \ldots, \alpha_s\}$, where $\Delta P = \alpha_0$, $\Delta C_i = \alpha_i$ for $1 \leq i \leq s$.*

Define p_i as the probability that inputs of difference α_{i-1} lead to output of difference α_i, where the probability is taken over all choices of the round key and the inputs to the ith round. In [44] the notion of a Markov cipher was introduced. In a Markov cipher this probability is independent of the actual inputs of the round and is calculated over all possible choices of the round key. Also in [44] it was shown that in a Markov cipher if the round keys K_i are independent, the p_i's are also independent and

$$\Pr(\Delta C_s = \alpha_s \mid \Delta P_0 = \alpha_0) = \prod_{i=1}^{s} \Pr(\Delta C_i = \alpha_i \mid \Delta C_{i-1} = \alpha_{i-1}). \tag{2}$$

In some differential attacks using an $(r-1)$-round characteristic only the plaintext difference ΔP and the last ciphertext difference ΔC_{r-1} need to be fixed. That is, the intermediate differences $\Delta C_1, \Delta C_2, \ldots, \Delta C_{r-2}$ can have any value. Lai and Massey introduced the notion of *differentials* [44].

Definition 2. *An s-round* differential *is a pair of differences $\{\alpha_0, \alpha_s\}$, where $\Delta P = \alpha_0$, $\Delta C_s = \alpha_s$.*

The probability of an s-round differential $(\Delta P, \Delta C_s)$ is the conditional probability that given an input difference ΔP at the first round, the output difference at the sth round will be ΔC_s. More formally, the probability of an s-round differential is given as

$$\Pr(\Delta C_s = \beta_s \mid \Delta P = \beta_0) = \sum_{\beta_1} \cdots \sum_{\beta_{s-1}} \prod_{i=1}^{s} \Pr(\Delta C_i = \beta_i \mid \Delta C_{i-1} = \beta_{i-1}), \tag{3}$$

where $\Delta C_0 = \Delta P$. A differential will, in general, have a higher probability than a corresponding characteristic. Differentials were used in [54] to construct cipher secure against differential attacks. Also, for some ciphers there is a significant advantage in considering differentials instead of characteristics [40].

In a differential attack the attacker does not know the key. Therefore in finding a good differential, the attacker computes the probabilities of differentials assuming that all the round keys are uniformly random and independent. However, the pairs of encryptions an attacker gets are encrypted using the same key, where the round keys are fixed and (can be) dependent. In [42] this problem is dealt with as follows

Definition 3 ((Hypothesis of stochastic equivalence)). *For virtually all high probability $(r-1)$-round differentials (α, β)*

$$Pr_P(\Delta C_1 = \beta \mid \Delta P = \alpha,\ K = k) \approx Pr_{P,K}(\Delta C_1 = \beta \mid \Delta P = \alpha,)$$

holds for a substantial fraction of the key values k.

In the differential attack on IDEA in [9], it was exploited that the hypothesis of stochastic equivalence does not hold for IDEA reduced to 3.5 rounds. A differential attack was mounted for which the S/N-ratio is one when the differential is averaged over all keys. When the key is fixed the S/N-ratio is different from one and the secret key can be recovered with sufficiently many pairs of plaintexts and ciphertexts. In [38] a differential attack on DEAL is described using a differential of probability zero. Also, recently a differential attack with $S/N < 1$ on Skipjack was announced [5].

Experiments have shown that the number of chosen plaintexts needed by the differential attack in general is approximately c/p, where p is the probability of the differential being used and c a small constant.

Higher Order Differentials In [43] a definition of higher order derivatives of discrete functions was given. Later higher order differentials were used to cryptanalyse ciphers presumably secure against conventional differential attacks [37]. In [27] these attacks were extended and applied to the cipher of [54]. A dth order differential is a collection of 2^d (first-order) differentials. The main idea in the

higher order differential attack is the fact that a dth order differential of a function of nonlinear order d is a constant. Consequently, a $d+1$st order differential of the function is zero. Assume that (a subset of) the output bits of the reduced cipher are expressible as a low-degree polynomial $p(\tilde{x}) \in GF(2)[\tilde{x_1}, \tilde{x_2}, \ldots, \tilde{x_i}]$, where $\tilde{x_1}, \tilde{x_2}, \ldots, \tilde{x_i}$ is a subset of input bits to the reduced cipher. If this polynomial has degree not higher than d, then $\sum_{\tilde{x} \in \mathcal{L}_d} p(\tilde{x}) = c$, where $\mathcal{L}_d$ denotes a d-dimensional subspace of $GF(2)^n$ and c a constant. This method was applied to the cipher example given in [54]. This cipher is "provably secure" against a differential attack but can be broken in a higher order differential attack with relatively low complexity.

Truncated Differentials In some ciphers it is possible and advantageous to predict the values of only parts of the differences after each round of the cipher. Let $\{\alpha_0, \alpha_1, \ldots, \alpha_s\}$, be an s-round *characteristic*. Then $\{\alpha'_0, \alpha'_1, \ldots, \alpha'_s\}$ is called a truncated characteristic, if α'_i is a subsequence of α_i. Truncated characteristics were used to some extent in [6] but only in the outer rounds of a cipher. Note that a truncated characteristic is a collection of characteristics and therefore reminiscent of a differential. A truncated characteristic contains all characteristics $\{\alpha''_0, \alpha''_1, \ldots, \alpha''_s\}$ for which $\text{trunc}(\alpha''_i) = \alpha'_i$, where $\text{trunc}(x)$ is the truncated value of x, where the truncation is not further specified here. The notion of truncated characteristics extends in a natural way to truncated differentials introduced in [37].

The truncated differentials were used in [39] to attack SAFER K [46,47]. Also, in [9] truncated differential attacks were presented on IDEA [44] and latest on Skipjack [5].

4.3 Linear Cryptanalysis

Linear cryptanalysis was proposed by Matsui in 1993 [48]. A preliminary version of the attack on FEAL was described in 1992 [51]. Linear cryptanalysis is a known plaintext attack in which the attacker exploits linear approximations of some bits of the plaintext and ciphertext. In the attack on iterated ciphers the linear approximations are obtained by combining approximations for each round under the assumption of independent round keys. The attacker hopes in this way to find an expression

$$(\tilde{P} \cdot \alpha) = (\tilde{C} \cdot \beta) \tag{4}$$

where α, β are n-bit strings and where '$\cdot$' denotes the dot product, which holds with probability $p_L \neq \frac{1}{2}$ over all keys, such that $|p_L - \frac{1}{2}|$, called the bias, is maximal. As in differential cryptanalysis one can define characteristics to be used in linear cryptanalysis.

The number of known plaintexts needed such that the relation (4) can be effectively detected is approximately $|p_L - 1/2|^{-2}$. The following result appears in [53].

Theorem 2. *If X and K are independent and K is uniformly distributed, then for all $a \in GF(2)^m$, $b \in GF(2)^n \in GF(2)^\ell$*

$$2^{-\ell} \sum_{k \in GF(2)^\ell} |\, P_X(X \cdot a + Y(X,k) \cdot b = 0) - 1/2 \,|^2 = \sum_{c \in GF(2)^\ell} |\, P_{X,K}(X \cdot a + Y(X,K) \cdot b + K \cdot c = 0) - 1/2 \,|^2$$

This theorem shows the similarity between the concept of differentials in differential cryptanalysis and in linear cryptanalysis. An expression of the form (4) is called a *linear hull*. Note that in [48] the linear approximations have the form $(\tilde{P} \cdot \alpha) = (\tilde{C} \cdot \beta) \oplus (K \cdot \gamma)$, where $(K \cdot \gamma)$ is an exclusive-or of round-key bits accumulated in the linear characteristic. The bias of the linear approximations is taken as the bias of the linear characteristic used. However, such an attack cannot be guaranteed to work in general. If there exist linear approximations such that $(\tilde{P} \cdot \alpha) = (\tilde{C} \cdot \beta) \oplus (K \cdot \gamma)$, and $(\tilde{P} \cdot \alpha) = (\tilde{C} \cdot \beta) \oplus (K \cdot \gamma')$ both with probability $p > 1/2$ but where $(K \cdot \gamma) \neq (K \cdot \gamma')$, then these two linear approximations may cancel the effect of each other. This was also noted in [3].

In Matsui's attack on the DES, experiments indicate that the bias of the linear hull is equal to the bias of a single characteristic [49]. It is further confirmed by computer experiments that the probability of (4) is close to 1/2 when the value of $\tilde{K}$ is wrong. It is estimated that the complexity of a linear attack on the DES with up to 16 rounds is about

$$N_P \simeq c \times |p_L - 1/2|^{-2}$$

where $c \leq 8$. The attack on the DES was implemented in 1994, required a total of 2^{43} known plaintexts [49] and is today the fastest, known key-recovery attack on the DES.

In [29] an improved linear attack using multiple linear approximations was given. In [41] a linear attack is shown using non-linear approximations in the outer rounds of an iterated cipher. For the DES none of these attacks have yet shown to offer an significant improvement compared to Matsui's linear attack. The attacks seem best suited for attacks on ciphers with large S-boxes.

4.4 Davies' Attack

In [17] a correlation attack on the DES was outlined. It exploits that the outputs from neighbouring S-boxes are not uniformly distributed. The correlation can be iterated to any number of rounds with a corresponding decrease in the probability. The attack was improved in [4] and finds the secret key of the DES using about 2^{50} known plaintexts, and is the third, known key-recovery attack which finds the secret key faster than by an exhaustive search.

4.5 Differential-Linear Attack

In [25] it was shown how to combine the techniques of differential and linear attacks. The attack is a chosen plaintext attack and considers pairs of plaintexts and ciphertexts, the bits of which are (partly) approximated by linear approximations. In particular, an attack on the DES reduced to 8 rounds was devised, which on input only 512 chosen plaintexts finds the secret key. It seems that the attack is not easily extended to more than 8 rounds of DES [25]. In [1] the differential-linear attack was applied to FEAL. The attack takes a long time, but only 12 chosen plaintexts are needed.

4.6 Other Variants

Several generalisations of the differential and linear attacks have been developed. In [67] a generalisation of both the differential and linear attacks, known as *statistical cryptanalysis* was introduced. It was demonstrated that a statistical attack on the DES included the linear attack by Matsui but without any significant improvement. The applications to other ciphers have not been demonstrated. In [22,23] two generalisations of the linear attack were given. However, none of them have yet proved to be much more efficient than the linear attack.

4.7 Interpolation Attack

In [27] the interpolation attack was introduced based on the following well-known formula. Let R be a field. Given $2n$ elements $x_1, \ldots, x_n$, $y_1, \ldots, y_n \in R$, where the x_is are distinct. Define

$$f(x) = \sum_{i=1}^{n} y_i \prod_{1 \le j \le n, j \ne i} \frac{x - x_j}{x_i - x_j}. \tag{5}$$

$f(x)$ is the only polynomial over R of degree at most $n-1$ such that $f(x_i) = y_i$ for $i = 1, \ldots, n$. Equation (5) is known as the *Lagrange interpolation formula* (see e.g., [10, page 185]).

In the interpolation attack an attacker constructs polynomials using inputs and outputs of the reduced cipher. This is particularly easy if the components in the cipher can be easily expressed as mathematical functions. The idea in the attack is, that if the constructed polynomials have a small degree, only few plaintexts and their corresponding ciphertexts are necessary to solve for the (key-dependent) coefficients of the polynomial. In an extended version of the attack meet-in-middle techniques are used to further reduce the degrees of the used polynomials [27].

Recently, a probabilistic version of the interpolation attack was introduced [26].

4.8 Non-surjective Attack

In [61] the non-surjective attack on iterated ciphers was described. It is applicable to Feistel ciphers where the round function is not surjective. In a Feistel cipher the plaintexts and corresponding ciphertexts give the exclusive-or of all outputs of the round function. Thus, if the round function is not surjective this gives information about intermediate values in the encryptions, which can be used in an attack.

4.9 Key Schedule Attacks

In this section we consider the key schedules of block ciphers. We consider an n-bit block cipher, where $E_K(\cdot)$ denotes encryption with the key K and $D_K(\cdot)$ denotes decryption. A weak key K, is a key for which encryption equals decryption, that is, $E_K(X) = D_K(X)$ for all n-bit texts X. A pair of semi-weak keys K, K^*, are keys for which encryption with one keys equals decryption with the other key, that is, $E_K(X) = D_{K^*}(X)$ for all n-bit texts X or equivalently, $D_K(X) = E_{K^*}(X)$ for all n-bit texts X. It is well-known that there are at least four weak keys and six pairs of semi-weak keys for the DES. In [11] it was shown that there are exactly 2^{32} fixed points for the DES used with a weak key.

If there are only a small number of weak keys they pose no problem for applications of encryption if the used keys are chosen uniformly at random. However, when block ciphers are used in hash modes where e.g., the key input can be chosen by the attacker in attempts to find collisions, they play an important role as demonstrated in [14,59].

[13] lists a large class of 2^{51} keys for IDEA, which can be easily identified using only a few plaintexts and ciphertexts. Note that IDEA uses 128-bit keys. In [68] it was shown that for 1 in 2^{15} keys for Blowfish a differential attack is faster than an exhaustive key search. [40] lists a large class of differentially weak keys for RC5 [62], keys for which a specific differential attack has improved performance.

Related Key Attacks There are several variants of this attack depending on how powerful the attacker is assumed to be.

1. Attacker gets encryptions under one key.
2. Attacker gets encryptions under several keys.
   (a) Known relation between keys.
   (b) Chosen relation between keys.

The first kind of attacks was introduced in [33], the second kind of attacks in [2]. Also, there are related key attacks on SAFER K [36] and on several other block ciphers [30].

Note that for the attacks of 2b above one must omit Assumption 1. It may be argued that the attacks with a chosen relation between the keys are unrealistic. The attacker need to get encryptions under several keys, in some attacks even

with chosen plaintexts. However there exist quite realistic settings, in which an attacker may succeed to obtain such encryptions, as argued in [30]. Also, there exists quite efficient methods to preclude the related key attacks [30,16].

5 Design of Block Ciphers

In this section we discuss some of the problems involved in the design of a block cipher. Two generally accepted design principles for practical ciphers are the principles of confusion and diffusion that were suggested by Shannon. Massey[45] interprets Shannon's concepts of confusion and diffusion [64] as follows *Confusion:* "The ciphertext statistics should depend on the plaintext statistics in a manner too complicated to be exploited by the cryptanalyst". *Diffusion:* "Each digit of the plaintext and each digit of the secret key should influence many digits of the ciphertext". These two design principles are very general and informal. Shannon also discusses two other more specific design principles. The first is to make the security of the system reducible to some known difficult problem. This principle has been used widely in the design of public-key systems, but not in secret-key ciphers. Shannon's second principle is to make the system secure against all known attacks, which is still the best known design principle for secret-key ciphers today.

There have been many suggestions in the past of more specific design principles, e.g. completeness, strict avalanche criterion, see [52, page 277-278]. However a specific cryptographic design principle should not be overvalued. Design principles should be seen as "guidelines" in the construction of ciphers, evolved from years of experience, and as necessary, but *not* sufficient requirements. There are many examples of this in the history of cryptography. We already mentioned the example of [27], where a block cipher "provably secure" against differential and linear attacks was broken by some other means.

5.1 Block and Key Size

It is clear from the discussion in Section 3.3 that if either the block or key size is too small or both, a block cipher is vulnerable to a brute force attack. These attacks are independent of the internal structure and intrinsic properties of an algorithm. Most block ciphers in use today have a block size of 64 bits. For these ciphers the birthday attacks of Theorem 1 require storage/collection of 2^{32} ciphertext blocks for a success of about one half. It may seem unlikely that a single key is used to process that many ciphertexts, and the storage of 2^{32} ciphertext blocks of each 64 bits will require about 2^5 Gigabytes of memory. However with the rapid increase in computing power and available storage media it can expected that in a few years this attack is very realistic. This has be taken into consideration in the ongoing development of the Advanced Encryption Standard, cf. later.

The key size of the DES is only 56 bits, which is too short. In [69,70] a design of an exhaustive search machine was given, which at the cost of 1 million US$

finds the secret key of the DES in average time 0.5 hours. In [8] is was estimated that with respect to an exhaustive key search a key size of at least 90 bits will suffice for the next 20 years.

5.2 Resistance against Differential and Linear Attacks

We consider an r-round iterated block cipher with round function G. Denote by p_d the highest probability of a non-trivial one-round differential achievable by the cryptanalyst. Let p be the probability of a linear approximation. Then $|p - 1/2|$ is called the bias. Recall that the success of a linear attack is proportional to the reciprocal value of the square of the bias of the used linear approximation. It has been shown how to treat differential and linear cryptanalysis in a similar way [50] by defining $q = (2p - 1)^2$. Let q_ℓ denote the highest such quantity for a one-round linear approximation. It is possible to lower bound the probability of any differential and any hull in an r-round iterated cipher expressed in terms of p_d and q_ℓ.

Theorem 3 ([34]). *Consider an r-round iterated cipher, which has independent round keys. Any s-round differential, $s \geq 1$, has a probability of at most p_d. Any s-round linear hull, $s \geq 1$, has a reciprocal squared bias of at most q_ℓ.*

For Feistel ciphers, Theorem 3 is trivial, since $p_d = q_\ell = 1$ when the right halves of the inputs are fixed. These differentials and hulls are called trivial one-round differentials and hulls. It is possible to lower bound the probabilities of differentials and hulls in a Feistel cipher expressed in terms of the most likely non-trivial one-round differential with probability p_{max} and the best non-trivial one-round linear hull with reciprocal squared bias of q_{max}.

Theorem 4 ([54,50]). *Consider an r-round Feistel cipher with independent round keys. Any s-round differential, $s \geq 4$, has a probability of at most $2p_{max}^2$. Any s-round linear hull, $s \geq 4$, has a reciprocal squared bias of at most $2q_{max}^2$.*

It has been shown that the round function in a Feistel cipher can be chosen in such a way that p_{max} and q_{max} are small [54,34].

5.3 Resistance against other Attacks

As mentioned earlier one should be careful not to focus too much on the resistance against a limited set of attacks, when constructing new block ciphers. In some cases other attacks become possible.

Let E be a n-bit r-round iterated block cipher. Assume that the nonlinear order of the ciphertext bits after one round is d and d^s after s rounds with a high probability. Then higher order differential attacks will in general not be possible after r rounds, if $d^r \simeq n$. One should take into account that the attacker may be able to guess key bits in the outer rounds of the cipher thereby attacking a cipher with a fewer number of rounds. Thus, if the nonlinear order should reach the block size after, say, $r - 2$ rounds.

It is yet unknown how to obtain exact security against truncated differential attacks. However, a truncated differential is a collection of differentials. Therefore, if the probabilities of all differentials can be bounded sufficiently low, this attack will have only small probability of succeeding.

The differential-linear attack will only work if both good linear hulls and good differentials exist. Thus, the techniques of the previous section also apply in this case.

The interpolation attack works particularly well when the outputs of one round of a cipher can be described as a polynomial of the input bits with relatively few nonzero coefficients. Thus, if a cipher consists of elements which cannot be described as such, it seems that the attack will not be possible. The probabilistic version of the interpolation attack might improve on this, but this has not been reported and needs further study.

The key-schedule attacks can be precluded by using only so-called strong key-schedules [35], see also [30,16].

6 Enhancing the Strength of the DES

Already in 1977 the DES was criticised for its short key length and it was suggested to use the DES in a triple encryption mode [21]. In a triple encryption with three independent keys K_1, K_2, and K_3, the ciphertext corresponding to P is $C = E_{K_3}(E_{K_2}(E_{K_1}(P)))$. One variant of this idea is well-known as two-key triple encryption, proposed in [66], where the ciphertext corresponding to P is $E_{K_1}(D_{K_2}(E_{K_1}(P)))$. Compatibility with a single encryption can be obtained by setting $K_1 = K_2$. However, whereas triple encryption is provably as secure as single encryption, a similar result is not known for two-key triple encryption. A two-key triple encryption scheme with a proof of security appeared in [16]. Another method of increasing the key size is DES-X, developed by Rivest. In DES-X the ciphertext corresponding to P is $C = E_K(P \oplus K_1) \oplus K_2$, where K is a 56-bit key, and K_1 and K_2 are 64-bit keys. Alternatively, $K_1 = K_2$ may be used. It was shown [31] that for attacks not exploiting the internal structure the effective key size of DES-X is $118 - \log_2 m$ bits, where m is the maximum number of plaintext/ciphertext pairs the attacker can obtain.

Although all these schemes increase the key lengths of the DES, the block lengths of 64 bits of these proposals are the same as for DES, and the matching ciphertext attack is still a problem.

7 The Advanced Encryption Standard

A better solution than those of the previous section seems to be to construct a new block cipher with larger keys and larger blocks to replace the DES, a cipher which at the same time is immune to all kinds of attacks reported so far in the cryptographic literature. Such an initiative was announced in January 1997 by the U.S. National Institute of Standards and Technology (NIST), the same institute that standardized DES in the 70's. The first workshop was held

April 15, 1997. NIST's intention is to standardize a new encryption algorithm, the *Advanced Encryption Standard (AES)* [57], as a replacement for DES. NIST encouraged parties world-wide to submit proposals for the new standard. Submission deadline was June 15, 1998; 15 proposals from all over the world were submitted and all proposals are now publicly available [58]. The proposals are required to support at least a block size of 128 bits, and three key sizes of 128, 192, and 256 bits. NIST hopes that the end result is a block cipher "with a strength equal to or better than that of Triple-DES and significantly improved efficiency." With the minimum requirements for the key sizes it is clear that an exhaustive key search will be infeasible for many years. Also with a block size of 128 bits the matching ciphertext attack requires a huge number of about 2^{64} ciphertext blocks to come into play.

The submitters of most of the algorithms claim a very high level of security. An exhaustive search for the key is often claimed to be the best attack, or it is claimed that an attacker would need all 2^{128} possible inputs and outputs to succeed.

However, we think that once a few candidates have been selected by NIST, the increased attention of the worlds cryptanalysts will result in new analysis and in levels of security much lower than claimed by the designers. In particular, we conjecture that (theoretical) key-recovery attacks with complexities in the neighborhood of 2^{100} or less will be found against most of the candidates (provided that they are looked at) in 5 to 10 years and therefore with a security level lower than the best known key-recovery attacks on triple-DES today. Also, a long-time conjecture is that the (theoretical) security level of the final candidate, or the final few candidates in case NIST should decide for several algorithms, will drop to less than 2^{64} in 30 years from now.

8 Conclusion and Open Problems

This paper considers contemporary block ciphers. In the last decade there has been a huge increase in the public knowledge regarding the security of secret-key block ciphers, most notably through the publication of the differential and linear attacks. Although this has enabled us to break many systems faster than by an exhaustive search for the key, the best known attacks on many of these systems are not very practical and require either the encryptions of unrealisticly many chosen or known plaintexts and/or a huge memory and processing time. The open problems in cryptanalysis of block ciphers are easy to spot: Break all unbroken block ciphers! And there is a lot of them.

References

1. K. Aoki and K. Ohta. Differential-linear attack on FEAL. *IEICE Trans. Fundamentals*, E79-A(1):20–27, 1996. 117
2. E. Biham. New types of cryptanalytic attacks using related keys. In T. Helleseth, editor, *Advances in Cryptology: EUROCRYPT'93, LNCS 765*, pages 398–409. Springer Verlag, 1993. 118

3. E. Biham. On Matsui's linear cryptanalysis. In A. De Santis, editor, *Advances in Cryptology: EUROCRYPT'94, LNCS 950*, pages 341–355. Springer Verlag, 1995. 116
4. E. Biham and A. Biryukov. An improvement of Davies' attack on DES. In A. De Santis, editor, *Advances in Cryptology: EUROCRYPT'94, LNCS 950*, pages 461–467. Springer Verlag, 1995. 116
5. E. Biham, A. Biryukov, and A. Shamir. "Impossible" cryptanalysis. Presented at the rump session of CRYPTO'98. 114, 115
6. E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer Verlag, 1993. 112, 113, 115
7. E. Biham and A. Shamir. Differential cryptanalysis of the full 16-round DES. In E.F. Brickell, editor, *Advances in Cryptology: CRYPTO'92, LNCS 740*, pages 487–496. Springer Verlag, 1993. 113
8. M. Blaze, W. Diffie, R.L. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener. Minimal key lengths for symmetric ciphers to provide adequate commercial security. Document, January 1996. 120
9. J.B. Borst, L.R. Knudsen, and V. Rijmen. Two attacks on IDEA. In W. Fumy, editor, *Advances in Cryptology: EUROCRYPT'97, LNCS 1233*, pages 1–13. Springer Verlag, 1997. 112, 114, 115
10. P.M. Cohn. *Algebra, Volume 1*. John Wiley & Sons, 1982. 117
11. D. Coppersmith. The real reason for Rivest's phenomenon. In H.C. Williams, editor, *Advances in Cryptology: CRYPTO'85, LNCS 218*, pages 535–536. Springer Verlag, 1986. 118
12. D. Coppersmith, D.B. Johnson, and S.M. Matyas. Triple DES cipher block chaining with output feedback masking. Technical Report RC 20591, IBM, October 1996. Presented at the rump session of CRYPTO'96. 111
13. J. Daemen, R. Govaerts, and J. Vandewalle. Weak keys for IDEA. In D.R. Stinson, editor, *Advances in Cryptology: CRYPTO'93, LNCS 773*, pages 224–231. Springer Verlag, 1993. 118
14. I.B. Damgård and L.R. Knudsen. The breaking of the AR hash function. In T. Helleseth, editor, *Advances in Cryptology: EUROCRYPT'93, LNCS 773*, pages 286–292. Springer Verlag, 1993. 118
15. I.B. Damgård and L.R. Knudsen. Multiple encryption with minimum key. In E. Dawson and J. Golic, editors, *Cryptography: Policy and Algorithms. International Conference, Brisbane, Queensland, Australia, July 1995, LNCS 1029*, pages 156–164. Springer Verlag, 1995. 110
16. I.B. Damgård and L.R. Knudsen. Two-key triple encryption. *The Journal of Cryptology*, 11(3):209–218, 1998. 110, 119, 121
17. D. Davies and S. Murphy. Pairs and triples of DES S-boxes. *The Journal of Cryptology*, 8(1):20–27, 1995. 116
18. D.W. Davies and W.L. Price. *Security for Computer Networks*. John Wiley & Sons, 1989. 105, 106, 107
19. D.E. Denning. *Cryptography and Data Security*. Addison-Wesley, 1982. 112
20. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. on Information Theory*, IT-22(6):644–654, 1976. 105
21. W. Diffie and M. Hellman. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, pages 74–84, 1977. 121
22. C. Harpes, G.G. Kramer, and J.L. Massey. A generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma. In L. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology - EUROCRYPT'95, LNCS 921*, pages 24–38. Springer Verlag, 1995. 117

23. C. Harpes and J.L. Massey. Partitioning cryptanalysis. In E. Biham, editor, *Fast Software Encryption, Fourth International Workshop, Haifa, Israel, January 1997, LNCS 1267*, pages 13–27. Springer Verlag, 1997. 117
24. M. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. on Information Theory*, IT-26(4):401–406, 1980. 111
25. M.E. Hellman and S.K. Langford. Differential–linear cryptanalysis. In Y. Desmedt, editor, *Advances in Cryptology: CRYPTO'94, LNCS 839*, pages 26–39. Springer Verlag, 1994. 117
26. T. Jakobsen. Cryptanalysis of block ciphers with probabilistic non-linear relations of low degree. In H. Krawczyk, editor, *Advances in Cryptology: CRYPTO'98, LNCS 1462*, pages 212–222. Springer Verlag, 1998. 117
27. T. Jakobsen and L. Knudsen. The interpolation attack on block ciphers. In E. Biham, editor, *Fast Software Encryption, Fourth International Workshop, Haifa, Israel, January 1997, LNCS 1267*, pages 28–40. Springer Verlag, 1997. 114, 117, 119
28. D. Kahn. *The Codebreakers*. MacMillan, 1967. 105, 108
29. B.S. Kaliski and M.J.B. Robshaw. Linear cryptanalysis using multiple approximations. In Y. Desmedt, editor, *Advances in Cryptology: CRYPTO'94, LNCS 839*, pages 26–39. Springer Verlag, 1994. 116
30. J. Kelsey, B. Schneier, and D. Wagner. Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and triple-DES. In Neal Koblitz, editor, *Advances in Cryptology: CRYPTO'96, LNCS 1109*, pages 237–251. Springer Verlag, 1996. 118, 119, 121
31. J. Kilian and P. Rogaway. How to protect DES against exhaustive key search. In Neal Koblitz, editor, *Advances in Cryptology: CRYPTO'96, LNCS 1109*, pages 252–267. Springer Verlag, 1996. 121
32. L.R. Knudsen. Block ciphers - a survey. To appear in the proceedings of the International Course on the State of the Art and Evolution on Computer Security and Industrial Cryptography 1997, to be published in the LNCS Series from Springer Verlag. 113
33. L.R. Knudsen. Cryptanalysis of LOKI'91. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology, AusCrypt 92, LNCS 718*, pages 196–208. Springer Verlag, 1993. 118
34. L.R. Knudsen. *Block Ciphers – Analysis, Design and Applications*. PhD thesis, Aarhus University, Denmark, 1994. 111, 112, 120
35. L.R. Knudsen. Practically secure Feistel ciphers. In R. Anderson, editor, *Fast Software Encryption - Proc. Cambridge Security Workshop, Cambridge, U.K., LNCS 809*, pages 211–221. Springer Verlag, 1994. 121
36. L.R. Knudsen. A key-schedule weakness in SAFER K-64. In Don Coppersmith, editor, *Advances in Cryptology - CRYPTO'95, LNCS 963*, pages 274–286. Springer Verlag, 1995. 118
37. L.R. Knudsen. Truncated and higher order differentials. In B. Preneel, editor, *Fast Software Encryption - Second International Workshop, Leuven, Belgium, LNCS 1008*, pages 196–211. Springer Verlag, 1995. 114, 115
38. L.R. Knudsen. DEAL - a 128-bit block cipher. Technical Report 151, Department of Informatics,University of Bergen, Norway, February 1998. Submitted as an AES candidate. 114
39. L.R. Knudsen and T. Berson. Truncated differentials of SAFER. In Gollmann D., editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 1996, LNCS 1039*, pages 15–26. Springer Verlag, 1995. 115

40. L.R. Knudsen and W. Meier. Improved differential attack on RC5. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO'96, LNCS 1109*, pages 216–228. Springer Verlag, 1996. 114, 118
41. L.R. Knudsen and M.P.J. Robshaw. Non-linear approximations in linear cryptanalysis. In U. Maurer, editor, *Advances in Cryptology: EUROCRYPT'96, LNCS 1070*, pages 224–236. Springer Verlag, 1996. 116
42. X. Lai. On the design and security of block ciphers. In J.L. Massey, editor, *ETH Series in Information Processing*, volume 1. Hartung-Gorre Verlag, Konstanz, 1992. 114
43. X. Lai. Higher order derivatives and differential cryptanalysis. In R. Blahut, editor, *Communication and Cryptography, Two Sides of One Tapestry*. Kluwer Academic Publishers, 1994. ISBN 0-7923-9469-0. 114
44. X. Lai, J.L. Massey, and S. Murphy. Markov ciphers and differential cryptanalysis. In D.W. Davies, editor, *Advances in Cryptology - EUROCRYPT'91, LNCS 547*, pages 17–38. Springer Verlag, 1992. 113, 115
45. J.L. Massey. Cryptography: Fundamentals and applications. Copies of transparencies, Advanced Technology Seminars, 1993. 109, 119
46. J.L. Massey. SAFER K-64: A byte-oriented block-ciphering algorithm. In R. Anderson, editor, *Fast Software Encryption - Proc. Cambridge Security Workshop, Cambridge, U.K., LNCS 809*, pages 1–17. Springer Verlag, 1994. 115
47. J.L. Massey. SAFER K-64: One year later. In B. Preneel, editor, *Fast Software Encryption - Second International Workshop, Leuven, Belgium, LNCS 1008*, pages 212–241. Springer Verlag, 1995. 115
48. M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology - EUROCRYPT'93, LNCS 765*, pages 386–397. Springer Verlag, 1993. 115, 116
49. M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Y.G. Desmedt, editor, *Advances in Cryptology - CRYPTO'94, LNCS 839*, pages 1–11. Springer Verlag, 1994. 112, 116
50. M. Matsui. New structure of block ciphers with provable security against differential and linear cryptanalysis. In D. Gollman, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 1996, LNCS 1039*, pages 205–218. Springer Verlag, 1996. 120
51. M. Matsui and A. Yamagishi. A new method for known plaintext attack of FEAL cipher. In R. Rueppel, editor, *Advances in Cryptology - EUROCRYPT'92, LNCS 658*, pages 81–91. Springer Verlag, 1992. 115
52. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. 119
53. K. Nyberg. Linear approximations of block ciphers. In A. De Santis, editor, *Advances in Cryptology - EUROCRYPT'94, LNCS 950*, pages 439–444. Springer Verlag, 1995. 115
54. K. Nyberg and L.R. Knudsen. Provable security against a differential attack. *The Journal of Cryptology*, 8(1):27–38, 1995. 114, 115, 120
55. National Bureau of Standards. Data encryption standard. Federal Information Processing Standard (FIPS), Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977. 106
56. National Bureau of Standards. DES modes of operation. Federal Information Processing Standard (FIPS), Publication 81, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., December 1980. 106
57. National Institute of Standards and Technology. Advanced encryption algorithm (AES) development effort. `http://www.nist.gov/aes`. 106, 122

58. National Institute of Standards and Technology. AES candidate algorithms. Descriptions available from NIST, see `http://www.nist.gov/aes`. 122
59. B. Preneel. *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, January 1993. 118
60. V. Rijmen. *Cryptanalysis and Design of Iterated Block Ciphers*. PhD thesis, Katholieke Universiteit Leuven, October 1997. 112
61. V. Rijmen, B. Preneel, and E. De Win. On weaknesses of non-surjective round functions. *Designs, Codes, and Cryptography*, 12(3):253–266, 1997. 118
62. R. Rivest. The RC5 encryption algorithm. In B. Preneel, editor, *Fast Software Encryption - Second International Workshop, Leuven, Belgium, LNCS 1008*, pages 86–96. Springer Verlag, 1995. 110, 118
63. B. Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). In R. Anderson, editor, *Fast Software Encryption - Proc. Cambridge Security Workshop, Cambridge, U.K., LNCS 809*, pages 191–204. Springer Verlag, 1994. 110
64. C.E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949. 108, 109, 119
65. M.E. Smid and D.K. Branstad. The Data Encryption Standard: Past and future. In G.J. Simmons, editor, *Contemporary Cryptology - The Science of Information Integrity*, chapter 1, pages 43–64. IEEE Press, 1992. 106
66. W. Tuchman. Hellman presents no shortcut solutions to DES. *IEEE Spectrum*, 16(7):40–41, July 1979. 121
67. S. Vaudenay. An experiment on DES - statistical cryptanalysis. In *Proceedings of the 3rd ACM Conferences on Computer Security, New Delhi, India*, pages 139–147. ACM Press, 1995. 112, 117
68. S. Vaudenay. On the weak keys of Blowfish. In Gollmann D., editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 1996, LNCS 1039*, pages 27–32. Springer Verlag, 1996. 118
69. M.J. Wiener. Efficient DES key search. Technical Report TR-244, School of Computer Science, Carleton University, Ottawa, Canada, May 1994. Presented at the Rump Session of CRYPTO'93. 119
70. M.J. Wiener. Efficient DES key search - an update. *CryptoBytes*, 3(2):6–8, 1998. 119

Primality Tests and Use of Primes in Public Key Systems

Peter Landrock

Mathematics Institute Aarhus University

Abstract. In the first part of this discussion, we first briefly discuss various prime generation methods, starting with the Rabin-Miller test, and then moving on to a very simple new deterministic test. After that we discuss various ways of constructing so-called strong primes, and why this better be avoided.

1 Rabin-Miller's Primality Test

The celebrated Rabin-Miller test is the most commonly used algorithm used for generating primes in public key schemes, be it RSA, DSA, Diffie-Hellman or elliptic curves in odd characteristic. The idea goes back to [1] and the probabilistic algorithm was introduced in [2].

This is a probabilistic algorithm that on input a(n odd) number can prove that it is composite or assert with some degree of certainty that it is a prime:

1. On input n, compute $n-1=2rh$, where h is odd.
2. Choose b uniformly in $[1,....,n-1]$
3. Then n passes if $b^h=1 \bmod n$ or if $b^{h2^i}=-1 \bmod n$ for some $i<r$.

By Fermat's Little Theorem, a prime always passes this test. The question is what we can say about a composite number that passes the Rabin-Miller test. Note that for all Carmichael numbers n, a base b chosen prime to n has a multiplicative order dividing $n-1$. There are infinitely many:

Definition: An integer n is called a Carmichael number if $\phi(n)|(n-1)$.

Consider a procedure that chooses odd k-bit numbers uniformly and output the first one that passes t iterations of Rabin's test.

Furthermore, let C denote the event that n is a composite, and let $T(t)$ denote the event that t iterations of the test outputs a composite number. Let $P(k,t)=P(C|T(t))$ denote the probability that this happens, where k is the bitlength of n.

1.1 Introductory Results

Notation: $M(k)$ is the set of odd numbers of bit length exactly k. Fix k and let n be an odd number of bitlength k.

Let $a(n)$ denote the elements of Z_n^*, for which the R-M test is positive. These elements do not form a subgroup, unfortunately. Let $\alpha(n)$ be the fraction of elements in $[1...n-1]$ for which R-M's test is positive.

I. Damgård (Ed.): Lectures on Data Security, LNCS 1561, pp. 127–133, 1999.

Lemma 1. *Let $n = p_1^{r_1} \cdot ... \cdot p_s^{r_s}$ be the decomposition of n into distinct prime factors. The fraction $\alpha(n)$ is bounded by*

- 2^{-s+1} *where s is the number of different prime divisors of n.*
- $2^{-s+1}q^{-1}$ *where*

$$q = \prod_{i=1}^{s} \frac{p_i^{r_i-1}u_i}{(h, p_i - 1)},$$

for u_i the odd part of $p_i - 1$ and h the odd part of $n - 1$.

Lemma 2. $\alpha(n) < 1/4$

Proof: See e.g. [MOR]

Thus it follows that $P(T(1)|C) \leq 1/4$, and hence that $P(T(t))|C) \leq 4 - t$. However, we are interested in $P(C|T(t)) = P(k, t))$!!

But even so, if we could prove e.g. that $P(C|T(1)) \leq 1/4$, a guaranteed error rate of 2^{64} would require 32 independent choices of bases for the Rabin-Miller test.

Experience shows that for the vast majorities of values of n, $\alpha(n)$ is very small, while for very few values of n, the maximal possible value just below $1/4$ is assumed. However, in [3], Paul Comba wrote:

"Unfortunately, the "vast majority" and the "very few" have not been quantified by mathematical analysis."

In [4] and further improvements in [5] partly based on [6], this analysis is provided. Earlier results by Erdös and Pommerance were not exact, but asymptotic only.

To evaluate this probability effectively, one needs to study average behaviour over the distribution of candidates, as first done in [7]. It is elementary to prove

Lemma 3. *With the notation above, we have*

$$P(k, t) \leq 4^{1-t}P(k, 1)/(1 - P(k, 1))$$

Indeed, this follows by Bayes' Theorem, $P(T(t))P(C|T(t)) = P(T(t)|C)P(C)$ where in particular $P(T(1))P(C|T(1)) = P(C)P(T(1)|C))$ From the former, we get

$$\begin{aligned} P(C|T(t)) &= P(T(t)|C)P(C)/P(T(t)) \\ &\leq 4^{-t+1}P(T(1)_C)P(C)/P(\neg C) \\ &= 4^{-t+1}P(T(1))P(C_T(1))/P(\neg C) \end{aligned}$$

where the last equality follows from the former and the inequality from the fact that $P(T(t)) \geq P(\neg C)$.

But as $P(\neg C|T(1))P(T(1)) = P(\neg C)P(T(1))|\neg C) = P(\neg C)$, due to the fact that the Rabin-Miller test is always positive on a prime (i.e. the incident $\neg C$), we have

$$\frac{P(T(1))}{P(\neg C)} = \frac{1}{P(\neg C|T(1))} = \frac{1}{1 - P(C|T(1))}$$

which inserted above yields the claim.

1.2 Estimates

We sketch the approach of [5] very briefly: In the following choose $k = 100$, say, and

$$2 \leq m \leq \sqrt{\frac{k}{2}}$$

We want to choose a subset C_m of M_k such that

1. C_m is small compared to M_k
2. The composite numbers n in $M_k \setminus C_m$ satisfy one of the following.
   - All prime divisors of n are smaller than $2^{k/m-1}$
   - Some prime divisor p in n is larger than $2^{k/m-1}$ and $(p-1)/(p-1, n-1) > 2^{m-1}$

The idea is to define C_m as the set of composite numbers in M_k for which neither condition holds:

$$C_m = \{n \in M_k | n \text{ is a composite with a prime divisor } p > 2^{k/m-1} \text{ such that } (p-1)/(p-1, n-1) \leq 2^{m-1} \}$$

Using the techniques above, it is easy to prove that $\alpha(n) < 2^{-m}$ for all $n \in M_k \setminus C_m$.

Example: Let $n = pqr$, where p, q and r are primes that are 3 mod 4, and n is a Carmichael number: Then $\alpha(n) = 1/4$. A specific example is $1729 = 7 \cdot 13 \cdot 19$, the celebrated taxi cab number. Hardy probably mentioned this number to Ramanujan at the famous visit to the hospital to cheer Ramanujan up by making the mock statement that this number was uninteresting. Hardy was very familiar with Carmichael's work and it is quite feasible that he thought such a statement might tricker off a reaction from the sad Ramanujan, as he (of course) too would recognise it as a Carmichael number. Ramanujan then astounded Hardy by pointing out that it is the smallest number which in two different ways may be written as a sum of two cubes.

By going through even more elaborate estimates but still along the same lines, these results may be dramatically improved. See [5] for details on the following table which in the (k, t)'th entry contains $-\log_2$ of the upper bound for $P(k, t)$. For instance, $P(150, 2) \leq 2^{-20}$.

$k \setminus t$	1	2	3	4	5	6	7	8	9	10
100	5	14	20	25	29	33	36	39	41	44
150	8	20	28	34	39	43	47	51	54	57
200	11	25	34	41	47	52	57	61	65	69
250	14	29	39	47	54	60	65	70	75	79
300	19	33	44	53	60	67	73	78	83	88
350	28	38	48	58	66	73	80	86	91	97
400	37	46	55	63	72	80	87	93	99	105
450	46	54	62	70	78	85	93	100	106	112
500	56	63	70	78	85	92	99	106	113	119
550	65	72	86	93	100	107	113	119	119	126
600	75	82	88	95	102	108	115	121	127	133

2 A Simple Deterministic Prime Generation Algorithm

The approach as such goes back to [8]: Start with a random prime p_1 of some limited size from a table and construct a prime $p_2 = k_1 p_1 + 1$. Continue with this process until a prime p_r is constructed of the right size. This was used by D. Wheeler to construct large primes in the 50ies. D. Wheeler always chose the coefficient k_i less than p_i. Compare to Theorem 101 in [8].

This can be improved as follows: Let p be a prime, and let $k = ap + b$, where $a, b < p$ are both odd. Set $n = kp + 1$.

Theorem 1. *Suppose there exists a t such that*

- $t^k \neq 1 \bmod n$
- $t^{kp} = 1 \bmod n$

Then n is a prime.

Note that n is in the range $p < n < p^3$.

Proof: We first observe that if n is composite, it must have a prime divisor q of the form $xp+1$, x even. Hence $n = qr$, where $r = yp+1$, y even. Thus $a = xy$ and $b = x + y$ are both even, a contradiction.

Note: It is easy to see that the condition that a, b be odd can be relaxed to the assumption that $b^2 - 4a$ not be a square integer.

Notice that the restriction on a and b only reduces the potential key space by a factor 4, i.e. two bits.

This test is much simpler than e.g. Ueli Maurer's construction of deterministic primes (see [9]), but gives the same uniform distribution properties, as we shall indicate:

What is the quality of the distribution of the primes constructed by this method?

We need an estimate of the probability that a random (odd) number n is divisible by primes up to a certain bound B only. Obviously, this probability equals

$$\prod_{3 \leq p \leq B} (1 - \frac{1}{p})$$

which by Merten's Theorem (see e.g. [8]) can estimated as

$$2e^{-\gamma}/\log(B) \approx 1/\log(B)$$

where γ is Euler's constant. Thus the distribution is linear in $\log(B)^{-1}$. This gives the following estimate for the fraction $\rho(e)$ of numbers x less than n whose largest prime factor is less than $x^{1/e}$.

e	$\rho(e)$
1.5	0.59453 48919
2.0	0.30685 28194
2.5	0.13031 95618
3.0	0.04860 83883
3.5	0.01622 95932
4.0	0.00491 09256
4.5	0.00137 01177
5.0	0.00035 47247
6.0	0.00001 96497
7.0	0.00000 08746
8.0	0.00000 00323
9.0	0.00000 00010

We observe that 95% of all (odd) numbers x have a prime divisor which is at least $x^{1/3}$.

Assuming that the distribution of prime divisors is independent of the fact that $n-1$ (or $n+1$ is a prime (which can be verified statistically), this result yields the probability that a prime p has the property that all prime divisors of $p-1$ or $p+1$ are below a certain bound. This in fact is also the argument behind not using strong primes above a certain limit (about 384 bits).

This estimate also indicates how the size of the prime p dividing $n-1$ is chosen if we start by choosing the bit length of the final candidate n: The distribution should be linear in $\log(p)^{-1}$. We may then successively call our algorithm to generate smaller and smaller primes, until we end up with a size we can look up in a table, and then go backwards in our construction using the theorem above.

Likewise, this estimate above yields that by test dividing with all primes up to say 256, we may discard about 80% $\approx (1 - 1/8)$ of all candidates. Thus we will speed up the prime generation time considerably by first test dividing with small primes and only then start our favorite algorithm up, and always with 2 as the first choice for the test base, as modular exponentiation of 2 is much faster than modular exponentiation in general.

For much more on this, see [10].

3 Constructing Strong Probabilistic Primes

The only problem to address here is how to construct a prime p such at the same time $p-1$ is divisible by the prime r and $p+1$ is divisible by a prime s. An

obvious tool is to use the Chinese Remainder Theorem. The obvious solution was first described in [11].

The basic idea is the following:

Calculate first some number a which is 1 mod r and -1 mod s, and then start examining all candidates of the form $n = a + 2jrs$, for $j = 1, 2, ...$ To find such a number a is of course easy. [12] suggests the number $a = ((2s^{r-2}) \bmod r)s - 1$, which has the advantage that it is quite small (Gordon suggested $(s^{r-1} - r^{s-1}) \bmod rs$).

Next test candidates of the form $n = a + 2jrs$ using the R-M test.

We observe that

- $n \bmod r = a \bmod r = 2 - 1 = 1$ by Fermat's Little Theorem
- $n \bmod s = a \bmod s = 0 - 1 = -1$.

In most applications, p is specified to be of a particular bit length, k. This is achieved by first constructing the primes r and s to satisfy that $\log r + logs$ is of size about $k - m \log k$, where m is small, say 4. Starting the algorithm of with $j = m \log k$, and then increasing j in alternating steps of 2 and 4 (to avoid the factor 3 (!)) the bit length of the final candidate will be k with a very high probability by the Prime Number Theorem.

The problem with this approach of course is that it only works if $rs < p$. Most algorithms seem to choose r and s of equal size, and this of course is very restrictive, and hence not recommendable at all. We do not have any exact estimate for the fraction of all good candidates which are accepted using this approach, but it is very, very small! Given our earlier discussions, it seems a much better idea to choose r randomly in the range $[p^{1/3}, p^{2/3}]$ or perhaps $[p^{1/2}, p^{2/3}]$ and then s accordingly, if "the customer" insists on strong primes.

Notice than alternative choice of a above is

$$a = ((-2r^{s-2}) \bmod s)r + 1$$

Hence candicates of the form

$$n = a + 2jrs = ((-2r^{s-2}) \bmod s + js)r + 1$$

are been considered, and Theorem 1 of Chapter 2 above may be invoked if $(j + 1)s < r^2$, e.g., if $s < r$, which is a reasonable assumption by the remarks above, and $j < r$, which is the intention of the whole approach anyway.

Final remarks: Any algorithm, however good, needs a random input of considerable size, called a random seed. This must originate from a random source, and this is a completely different story.

References

1. G.L. Miller, Riemann's hypothesis and test for primality, Journal of Computer and System Sciences 13 (1976), 300-317. 127
2. M.O. Rabin, Probabilistic algorithm for testing primality, Journal of Number Theory 12 (1980), 128-138. 127
3. P. Comba, Exponentiation Cryptosystems on the IBM PC, IBM Systems Journal, vol. 29,4, 1990. 128
4. I. Damgaard and P. Landrock: Improved bounds for the Rabin Primality Test, Proceedings of the IMA Conference on Cryptography and Coding III, 1991, Oxford University Press, ed. by M.J. Ganley. 128
5. I. Damgaard, P. Landrock and C. Pomerance: Average Case Error estimates for the Strong Probable Prime Test, Math. Computations 61 (1993), 177-194. 128, 129
6. S.-H. Kim and C. Pomerance, The probability that a random probable prime is a composite, Math. Computations 86 (1986), 259-279. 128
7. P. Beauchemin, G. Brassard, C. Crépeau, C. Goutier and C. Pomerance, (1988), The Generation of random Numbers that are Probably Prime, J. Cryptology vol. 1 (1988), 53-64. 128
8. G.H. Hardy and E.M. Wright, An Introduction to the Theory Numbers, Oxford University Press, fifth ed. (1978). 130, 131
9. U. Maurer, Fast generation of secure RSA-moduli with almost maximal diversity, Advances in Cryptology - EUROCRYPT'89, LNCS 547 (1991), 458-471. 130
10. J. Brandt, I. Damgaard and P. Landrock, Speeding up Prime Number Generation, Proc. of Asiacrypt'91, Springer LNCS, vol. 739. 131
11. J. Gordon, Strong RSA keys, Electronic Letters 20 (June 7, 1984), 514-516. 132
12. Mike Ganley (Electronic Letters 1990) 132

Signing Contracts and Paying Electronically

Torben P. Pedersen

Cryptomathic, Denmark
tpp@cryptomathic.dk

Abstract. Motivated by the increasing use of cryptography, in particular digital signatures, to secure electronic commerce this paper discusses applications of digital signatures. The aim is to give an overview of some problems, which on one hand are related to electronic commerce and, on the other hand, are challenging from a cryptographic point of view. The paper first deals with fundamental techniques for establishing a public key infrastructure and for creating non-repudiation tokens. The latter makes it possible to use digital signatures to solve disputes which is often the ultimate goal when using digital signatures in practice.
Next more advanced cryptographic protocols are discussed by giving an overview of protocols for fair exchange of signed documents as well as for implementing electronic cash (prepaid payment systems).

1 Introduction

Digital signatures were made possible by the invention of public key cryptography by Diffie and Hellman in the middle seventies (see [DH76]). In a public key (or asymmetric) crypto system a user has a key pair consisting of a private key known only to himself and a public key, which may be publicly announced and which must be known to all other parties communicating securely with the user in question. These keys are used in algorithms, which are also publicly known (usually standardised algorithms such as [DSS93] for digital signatures and [RSA78] for both confidentiality and digital signatures).

Now consider a party, A, having a public key pair (s, p), where s is the private key and p the public one. Other parties can send information *confidentially* to A by encrypting it under A's public key. A can retrieve the original information by deciphering the cipher text using s. As only A knows this key, A is the only person who can retrieve the encrypted information. Due to efficiency reasons public key cryptography is often used to encrypt symmetric keys, which are then used to encrypt a single message or used several times during a session.

In the above setting, A can digitally sign a message using his private key. This results in a digital signature which can be verified by anyone using A's public key. The verification process ensures that only someone knowing the private key corresponding to the public verification key (here A) could have produced the signature.

In the following we only consider public key cryptography for digital signatures and stress, that it is possible to have a public key scenario, which can be

I. Damgård (Ed.): Lectures on Data Security, LNCS 1561, pp. 134–157, 1999.

used for digital signatures but not for public key encryption. More precisely, inspired by the increase in electronic commerce, this paper focuses on applications of digital signatures to electronic payments and fair exchange of signed data. Obviously, electronic payments are fundamental to electronic commerce, while exchange protocols may be necessary to ensure fairness in business sessions involving parties that don't trust each other (e.g., a buyer may not want to sign a receipt for some goods unless he is sure to get the goods, and a merchant may not want to send the (electronic) goods, unless he gets a receipt).

Development of Digital Signature Schemes

While public key cryptography was introduced in [DH76], the first digital signature scheme was published a few years later in [RSA78]. Later, a number of digital signature schemes have been suggested, but only a few have survived extensive analysis in the cryptographic community. We now review the most notable ones.

The RSA system is based on the problem of factoring. The public key is a pair of numbers (n, e) where n is the product of two primes p and q, and e, the public exponent, is relatively prime to $\mathbf{lcm}(p-1, q-1)$. The secret key consists of n and d, where d is the inverse of e modulo $\mathbf{lcm}(p-1, q-1)$. The signature on some data, D (considered a positive number less than n) is

$$\sigma = D^d \pmod{n}.$$

This signature can be verified using the public key by computing D' as

$$D' = \sigma^e \pmod{n}$$

and verifying that the retrieved data D' is of the correct form.

While anybody who is able to factor n can make false signatures, it is not known whether breaking RSA requires the ability to factor (forgeries based on homomorphic properties are possible, but these can be prevented if the signed data, D contains sufficient redundancy).

However, Rabin presented in [Rab79] a variant of RSA where the public exponent is 2 (i.e., the exponentiation with the public exponent is replaced by squaring). Note that 2 is not a valid public exponent in RSA, but as computing square roots modulo a composite requires the ability to factor the composite, signing arbitrary messages in Rabin's scheme requires knowledge of the factorisation of the modulus (an interesting variant of Rabin's scheme was given in [Wil80]).

In 1984, ElGamal presented a signature scheme based on the difficulty of computing discrete logarithms in the mulitplicative group modulo a prime [EG85]. This scheme had a renaissance in the end of the eighties, as a variant of it was selected as public standard by NIST [DSS93]. The ElGamal and DSA signature schemes will be discussed in more detail in Section 2. These schemes have gained further interest as variants can be implemented in groups defined by elliptic curves over finite fields (e.g., "elliptic curve DSA" [1/S98a]).

At the end of the eighties a new paradigm for the construction of digital signatures was introduced (see [FS87,FFS88]). These signature schemes are derived from zero-knowledge identification protocols (see [GMR89] for information about zero-knowledge). A number of signature schemes have been based on this technique, e.g., [Sch90,GQ89].

Most of the applications discussed in this paper are generic in the sense that they can be based on any signature scheme. In these cases the schemes mentioned above are very good candidates.

Existence of Digital Signatures

One problem with practical digital signature schemes is that they are not "provably" secure (i.e., it cannot be shown that forging signatures requires the solution of a generally assumed hard problem). Even in Rabin's scheme, where signing arbitrary messages is equivalent to factoring, the signers secret key can be retrieved if an attacker is allowed to get signatures on arbitrary messages (see Section 2.1). Recently a number of practical signature schemes, including Schnorr signatures, have been proved secure in the random oracle model [BR93].

Security of digital signature schemes was not formally defined until [GMR88], which also presented a provably secure scheme based on the existence of claw-free pairs of trapdoor functions. Later the sufficient condition for making secure digital signature schemes was weakened in a series of papers. Most notably

- [BM92] showed how to make secure signatures given any trapdoor function (in particular secure digital signatures based on RSA was made possible, although not that practical);
- [NY89] based secure digital signatures on universal one-way hash functions, which can be constructed given any one-way permutation; and finally
- Rompel showed in [Rom90] how to construct universal one-way hash functions (and hence secure digital signatures) based on one-way functions. Existence of one-way functions is also a necessary condition for secure digital signatures.

This paper will not deal with these "theoretical" schemes but consider schemes, that are used in practice and some applications of these.

Overview

The next section defines secure digital signatures (based on [GMR88]) and discusses the security of some of the signature schemes mentioned above. The next four sections consider applications of signatures. First, Section 3 discusses public key infrastructures, which on one hand is an application of signatures, and on the other is a prerequisite for the deployment of public key techniques, and Section 4 describes a standard format for non-repudiation tokens. Methods for exchanging such tokens (more generally, contract signing) are described in Section 5, and Section 6 presents the principles behind some electronic payment systems.

2 Definition of Secure Digital Signatures

A signature scheme is defined by the following components:

Key Space A subset of $\{0,1\}^* \times \{0,1\}^*$ of pairs of private and public keys
Key Generator A probabilistic polynomial time algorithm, *gen*, which on input the security parameter k outputs a private key KS and a matching public key KP. All keys, messages and signatures are of polynomial length in k.
Message Space A subset of $\{0,1\}^*$.
Signature Space A subset of $\{0,1\}^*$.
Signing function A probabilistic polynomial time algorithm *sign*, which on input a message m and a private key KS outputs a signature $sign(m, KS)$.
Signature verification A binary verification function, *test*, which on input a message m, a public key KP and a signature σ outputs 1 if σ is valid with respect to KP and 0, if σ is invalid.

Often we shall just say that a signature scheme is defined by $(gen, sign, test)$ and not explicitly mention the key, message and signature spaces.

Sometimes (e.g., in [1/S98b] versus [1/S91]) one distinguishes between signatures with *appendix* and signatures giving *message recovery*. The above definition corresponds to the former, as the signature is appended to the message in the sense that both σ and m are required inputs to *test*. In signature schemes with appendix the signing process usually goes in two steps:

1. The message to be signed, m, is digested using a *cryptographic* hash function, $\mathcal{H}$, resulting in $D = \mathcal{H}(m)$.
2. The data D is processed using the private key of the signer resulting in a signature $sign(m, KS)$.

A potential signature σ on message m is verified in a similar two step process (see Figure 1):

1. Compute $D = \mathcal{H}(m)$.
2. Verify that σ matches the digest D using the public key.

In schemes giving message recovery (part of) the message is recovered during signature verification. Again the signing process goes in two steps:

1. Redundancy is added to the message, m resulting in data D.
2. The data D is processed using the private key of the signer resulting in a signature $sign(m, KS)$.

[1/S91], which can be used with RSA, prescribes that the input message is at most (roughly) half the length of the modulus so that there is room for redundancy. If the message is too long, only part of the message can be recovered, and the rest is used as input for the verification process. In that case the first step of the signature process goes as follows

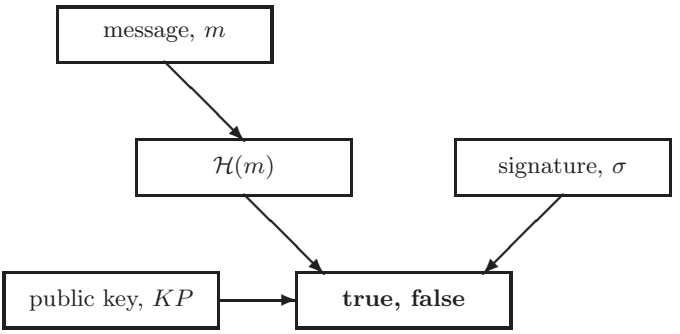


Fig. 1. Verifying signature with appendix

1. Write $m = m_1||m_2$, where $||$ denotes concatenation and construct $D = m_1||\mathcal{H}(m)$.

In this case *test* takes as input a partial message m_2, the potential σ and the public key and produces as output a pair (b, m), where

$$b = \begin{cases} 0 \text{ if } \sigma \text{ is invalid} \\ 1 \text{ if } \sigma \text{ is valid} \end{cases}$$

and m is the recovered message (only well-defined if $b = 1$). Testing goes in the following two steps (see Figure 2), given m_2, σ and KP:

1. Recover data D from σ using the public key.
2. Verify that the redundancy in D is correct and that D is of the form $D = m_1||\mathcal{H}(m_1||m_2)$. Return $m_1||m_2$ as the recovered message.

Thus in schemes with appendix as well as schemes giving message recovery we can identify in the signing and verification process a step which processes some data (denoted D) using the private key, respectively the alleged signature using the public key. This step will in the following be denoted as the *pure* part of the signature algorithm in order to differentiate it from the complete signature mechanism. Thus pure RSA is defined by the two modular exponentiation functions (with the private and public exponent).

2.1 Security of Digital Signatures

As mentioned previously [GMR88] provided the first thorough definition of digital signatures. The following definition is based on that paper.

The strength of a signature scheme is measured as the *achievement* under a given *attack*. The following four types of attacks are considered, with the most powerful mentioned last.

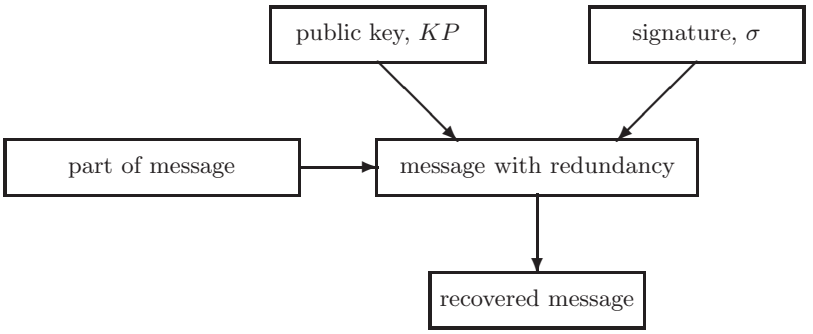


Fig. 2. Verifying signature with message recovery

Key only The adversary knows the public key of the signer.
Known message attack The adversary knows a number of message-signature pairs but cannot influence the distribution of these messages.
Chosen message attack The adversary makes a list of messages and gets the signature on each of these.
Adaptively chosen message attack In this type of attack the adversary can get the correct signature from the signer in a number of rounds. In each round the adversary can choose the messages to be signed based on the signatures received so far.

Note that the ability to carry out a chosen message attack (adaptively or not) is in many cases detrimental to the application of the signature scheme, as these attacks basically allow the adversary to get signatures on arbitrary messages. However, these scenarios are considered in order to allow for very strong attacks. This will be more clear when the achievements or goal of the attack is described. Let $\mathcal{M}$ denote the set of messages signed by the attacked signer during an attack. Then the following three different achievements are considered:

Total break The adversary obtains the secret key or other equivalent information allowing him to make signatures at will.
Selective forgery The adversary is able to make a signature on a message $m \notin \mathcal{M}$ chosen by himself.
Existential forgery The adversary is able to make a signature on some message $m \notin \mathcal{M}$ (the adversary may not control m).

The scheme is resistant to an attack with a given goal if for every polynomially bounded adversary and for every $c > 0$ the probability of achieving the goal is less than k^{-c} for k sufficiently large (where the probability is over the random choices of the attacker as well as the signer — k is the security parameter of the scheme).

Thus the highest level of security occurs when a scheme is secure against existential forgery under adaptively chosen message attacks.

2.2 Security of Practical Schemes

In the following we briefly describe the security of RSA, Rabin and DSA in terms of the definition given above.

RSA and Rabin Signatures It is easy to make an existential forgery in pure RSA with public key (n, e) by selecting a positive, random number $\sigma < n$ and computing $D = \sigma^e \bmod n$. However, if RSA is combined with a hash function, then it will be difficult to find a real message, m, such that $D = \mathcal{H}(m)$. Similarly, if redundancy is added to the message as part of signing and removed again as part of verification, then this attack will fail. The same goes for attacks based on the *homomorphic* property of RSA:

$$m_1^e m_2^e = (m_1 m_2)^e \bmod n.$$

Rabin's signature scheme is constructed such that the ability to make a total break in the pure scheme implies that the public modulus can be factored. Unfortunately, a chosen message attack enables the adversary to find the factorisation of n, as follows:

1. The adversary computes data $D = x^2 \bmod n$, where x is chosen at random and ask the signer to sign D.
2. The signature, σ, from the signer satisfies: $\sigma^2 = D \bmod n$.
3. As D contains no information about which of the four square roots of D the adversary knows, the adversary is able to factor n with probability $\frac{1}{2}$.

Thus, the security of pure Rabin signatures can be characterised as follows:

- existential forgery possible under known key attack
- total break possible under chosen message attack
- selective forgery impossible under known key attack (and under known message attacks, where, for example, the signer signs messages that are selected uniformly among the quadratic residues modulo n).

DSA and ElGamal Just as for RSA it is possible to make an existential forgery of pure DSA and ElGamal signatures. As mentioned previously, DSA is a variant of ElGamal. It uses system parameters p and q (both primes), where q is of length 160 bits and divides $p - 1$, and a number $g \in \mathbb{Z}_p^*$ of order q. The public key is an element $y \in \mathbb{Z}_p^*$ and the secret key is a number x between 1 and $q - 1$ such that

$$y = g^x \bmod p.$$

Signing some data $D \in \{0,1\}^{160}$, interpreted as a number between 0 and $q-1$, is a probabilistic process, where the signer chooses $k \in \mathbb{Z}_q^*$ at random and computes

$$r = (g^k \bmod p) \bmod q$$
$$s = (D + xr)/k \bmod q$$

Signature verification is done as follows:

1. Compute

$$u1 = D/s \bmod q$$
$$u2 = r/s \bmod q$$

2. The signature is valid if $r = \left(g^{u1} y^{u2} \bmod p\right) \bmod q$.

A correctly made signature will be accepted in this process as

$$g^{u1} y^{u2} = \left(g^{D+xr}\right)^{1/s} = g^k \bmod p.$$

The DSS standard prescribes that this signature system must be used with the hash function SHA-1 given in [SHS95]. This allows longer messages to be signed, and prevents some attacks that have been known since ElGamal signatures were introduced in [EG85]. Without a hash function it is possible to make an existential forgery as follows. Initially choose $a \in \mathbb{Z}_p$ of order q and $b \in \mathbb{Z}_q$ at random and compute

$$t = ay^b \bmod p$$
$$r = t \bmod q$$
$$s = r/b \bmod q$$

Let $u1$ and $u2$ be defined as above (D, the data to be signed are still unknown). Then we have to select a, b and D such that

$$g^{u1} y^{u2} = t \bmod p.$$

But this is equivalent to

$$g^D = y^{-r} t^s \bmod p$$

and hence

$$g^D = y^{-r} a^s y^{bs} = y^{-r} a^s y^r = a^s \bmod p.$$

Thus if instead of choosing a at random we select $a = g^c$ for a random c, then (r, s) is a correct signature on data $D = cs \bmod q$.

When "pure DSA" is used with SHA-1 as prescribed, the above forgery is not of much use, as the adversary would have to find a message m such that $\mathcal{H}(m) = D$ — a problem which has no publicly described solution. Even though there is some freedom in choosing, D, it is not known how to make even an existential forgery.

Schnorr Signatures These can for example be described in the same setting as DSA. Thus system parameters p, q and g as above are given. The public key is $y = g^x \bmod p$, where x is the private key (some times g^{-x} is used as public key, but that makes no real difference for our purposes).

A signature on a message m is most easily described as a proof of knowledge of the private key:

1. The prover (signer) chooses $r \in \mathbb{Z}_q$ at random and computes $a = g^r \bmod p$.
2. Compute a challenge $c = \mathcal{H}(a, m)$
3. The prover (signer) computes $z = r + cx \bmod q$.

Intuitively, this can be considered a proof of knowledge of x since x can be computed if the prover is able to answer correctly on two different challenges based on the same a.

A signature is the pair (c, z). It is correct if $c = \mathcal{H}(g^z h^{-c}, m)$. The security of this scheme depends on the properties of $\mathcal{H}$, but assuming $\mathcal{H}$ behaves like a random oracle Schnorr signatures are secure against existential forgeries under adaptively chosen message attacks.

3 Public Key Infrastructure

This and the following three sections describe practical aspects of using digital signatures. First, certificates are discussed, as these on one hand constitute a simple application of digitally signed messages and, on the other, enable other applications.

In order to use digital signatures it is usually necessary to have a *Public Key Infrastructure* (PKI) in place.[1] As a signed message is verified against the public key, it proves, assuming that the signature is not forged, that the message originates from the person knowing the private key corresponding to the public one. Thus the public key serves as the electronic identity, and the main purpose of a PKI is to link this electronic identity with the owners real identity (or in some cases with a pseudonym chosen by the owner).

In practice, this is done using *public key certificates*. A certificate is an electronic message stating that a given public key belongs to a certain person. It is issued and digitally signed by a third party called a *certification authority* (CA). Everybody knowing the public key of the CA can verify certificates issued by that CA and hence use the public keys in these certificates. Two certificate standards are given in [X5095,IS0].

While the CA is central for establishing a PKI, two other roles are often involved:

[1] A simple situation, where no particular PKI is required is in "star shaped topoligies", where all participants only communicate securely with a single central party. In that case the central party can maintain a table of the public keys of all other parties. These, in turn, only need to know the public key of the central party.

– A *registration authority* (RA) which verifies information about the user (in particular the identity of the user) and links the public key to the user. In some applications a number of local RAs are required where applicants must show up in person before getting a certificate.
– A *directory* (D) which maintains a register of public information about users and certificates. Certificates can be published in and hence retrieved from a directory. The LDAP protocol [YHK95] is a standard protocol for accessing such information.

A user registers at the registration authority and obtains a certificate from the certification authority. Later the certificate can be used by either including the certificate in the electronic messages or letting the counterpart obtain information about the certificate from a third party such as the directory or the CA itself. Figure 3 illustrates this.

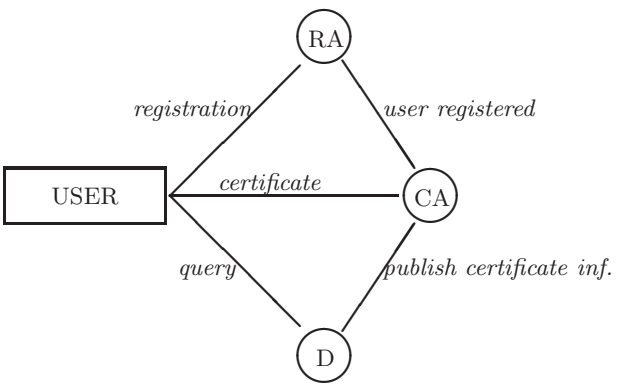


Fig. 3. Roles in a PKI

Since the purpose of a certificate is to link together a person and a public key, it is of course important that the identification of that person as well as the verification of the correctness of the key are done thoroughly: it must be ensured that the name of the person is correct and that the person acknowledges that the certified public key belongs to him.

Although other procedures exist, a certificate is typically issued as follows using the (local) RA and the CA:

1. The applicant registers at the RA. Several options are possible here
   – Electronic registration (e.g., identification against an email address). This is often done today, but should not be used if the certificate is to be used for non-repudiation (see Section 4).

   - Registration based on an already established relationship.
   - Physical registration, where the applicant must show up in person and the officer at the RA verifies the identity of the applicant.

   In the last two cases the applicant may have to sign (by hand) an application form (this form may contain a fingerprint of the public key, which is going to be certified).
2. The RA informs the CA about the registration.
3. The CA sends a so-called Initial Authentication Key (IAK) to the user in a sealed letter [PKI].
4. The applicant sends an electronic request for the certificate. In this request the applicant identifies himself using the IAK (e.g., by computing a MAC on the request) and proves knowledge of the private key corresponding to the public one to be certified.
5. The CA returns the requested certificate, if the request is ok (and the public key has not previously been certified).

However, no matter how many resources are put into the verification of the information in certificates when they are issued, the PKI must support means for revoking certificates. Most noteworthy, this will happen if the certified key-pair is (suspected to be) compromised, but it could also be necessary in less dramatic circumstance (e.g., if some information in the certificate is out-dated). In case a certificate is revoked, the PKI must make sure that the change of status is announced properly. This could for example be through announcements of certificate revocation lists (blacklists) or by providing an on-line service, which can always give the correct status of any given certificate in a secure way.

An important difficulty when establishing a PKI is to publish the public key of the CA. This key is at the heart of the security of any system, as all certificates issued by the CA are verified against this key. Thus someone who is able to replace that key with another one will be able to issue (false) certificates, and hence make signatures alleged to originate from someone else. Distribution of the key of the CA can either be done "out-of-band" (e.g., by publication in newspapers or in letters when a user is registered) or by certifying it using another CA. The latter may give rise to a hierarchy of CA's in which only the distribution of the root key will have to be done out-of-band.

4 Non-repudiation

Non-repudiation refers to the use of digital signatures for solving disputes. Thus the digital signature, which is made as part of a transaction should be stored and in the event of a dispute an arbiter must be able to verify it.

ISO provides a framework for non-repudiation in part I of ISO13888 [1/S97a], and in part III [1/S97b] specific non-repudiation tokens are defined. The most interesting are

Non-repudiation of origin Protects against the originators false denial of having originated the message.

Non-repudiation of delivery Protects against the recipients false denial of having received and recognised the contents of the message (ISO uses the term "non-repudiation of receipt" to refer to a proof that the recipient has just received the message).

In addition non-repudiation tokens are defined to support the electronic equivalent of registered mail.

The tokens all follow a similar structure, so let us take a look at the non-repudiation of origin token. It contains the following information, which is signed:

- A description of the non-repudiation policy for this token (i.e., what does the token prove).
- Identification of the originator.
- Identification of the intended recipient(s).
- Identification of the authority generating the token (usually the originator).
- Date and time when the token was generated.
- Date and time when the message was sent.
- Description of signature mechanism (including hash function).
- Hash value of the message.

The difficult (and yet essential) part to provide in this token is the time stamp, which is used to make the token unique and, in case of disputes, to determine the exact time of the generation of the token. A possible scenario where correct time stamps are important is:

> User A signs a message stating that he owes person B 1000 dollars and will pay this amount 2 months later. One month later A regrets and revokes the certificate claiming that his private key may be compromised. When B later wants to get his money, A refuses to pay claiming that the signature was made after the certificate was revoked.

Obviously, if the initial message from A had an unforgeable time stamp, A could not succeed with this claim.

In order to provide such time stamps a third party is needed. [1/S97a] defines a time stamp token to contain the signature of the Time Stamp Authority on a message containing

- A description of the non-repudiation policy for the time stamp
- Date and time when the time stamp was generated
- Description of signature mechanism (including hash function)
- Hash value of the message to be time stamped

Internet standards for time stamping protocols are developed in the PKIX working group [PKI]. Interestingly, the time stamp defined within PKIX in the current version provides the option of associating additional information to the token in order to prevent that the third party dates forward (e.g., the time stamp could contain the most recent closing value of the Dow Jones Average). However, as illustrated by the example above back dating is often a more serious problem.

Two cryptographic solutions to this problem were proposed in [HS91]. In one solution, the third party making the time stamps link the stamps together so that back dating (before the previous time stamp) is not feasible. This solution has the problem that solving a dispute involving time stamps may require as witness other parties having requested time stamps. The second solution proposes to distribute the time stamp among a number of third parties (in a random and unpredictable way).

5 Contract Signing

Contract signing refers to the problem of *fairly* exchanging signed documents between two parties. By fair is meant that each (honest) party sending a signed document is assured that if the other party gets his signature, then he will also get the required signature from that party.

Using a trusted third party this problem has a trivial solution: Both parties can send their signed documents to the third party, who after receiving both documents verifies the signatures, and then forwards the signed documents to the intended recipients.

As this solution is not satisfactory, it has been attempted to develop contract signing protocols relying less on trusted third parties. In theory, this problem is solved, since fair exchange of signed documents can be based on general cryptographic techniques such as [GMW86].

However, the search for more practical solution has attracted much attention (see [Dam94] for more references) and in the following we consider some of the suggested schemes. These can be grouped in three categories:

Gradual release of signature Here each party takes turns releasing one or more bits of the signature.

Gradual release of evidence Each party takes turns at releasing evidence, which a third party will use to decide a possible dispute. The distinguishing property compared to gradual release of the signature is that a party having more computing power than the other cannot use this advantage to generate additional evidence.

Optimistic protocols These protocols provide the exchange in such a way, that if at any point a cheating party stops and is able to compute the signature of the honest party, then the honest party has so much information that, with the help of the third party, the cheating party's signature can be computed.

5.1 Gradual and Verifiable Release of Signature

The concept of gradual and verifiable release of a secret was first introduced in [BCDvdG88], which shows how a party can gradually release a secret discrete logarithm (given two elements g and h of $\mathbb{Z}_p^*$, where p is a prime, the secret is the discrete logarithm of h with respect to g).

The idea is that in one round the party holding the secret can prove that it is in a certain interval, and in each round this interval is made smaller (e.g., the interval could be halved in size for each round corresponding to releasing one bit of the secret). In [Dam94] this technique was generalised using a special bit commitment scheme. While [BCDvdG88] allows release of signatures that can be expressed as the discrete logarithm, [Dam94] extended the scope to signatures that can be expressed in a well-defined way using this new commitment scheme (this includes RSA, Rabin and ElGamal signatures).

We do not go more into the details of these schemes but there is a rich literature on such schemes.

5.2 Gradual and Verifiable Release of Evidence

If signatures are exchanged using the technique of gradually releasing a secret, then a party, A with much computing power may have a substantial advantage over one with very little computing power, as A will be able to search (exhaustively) for missing bits. E.g., assume the signature is 320 bits (as in DSS). After each party has released, say, 270 bits of the signature then party A my decide to stop and search exhaustively for the remaining 50 bits. The other party having much less computing power may not be able to retrieve the missing bits before the signature is out-dated.

[BOGMR90] introduced a way to cope with this deficit. The exchange still takes place in a number of rounds, but in each round each party releases information, which increases the *probability* that an arbiter will accept the signature. Thus at any point party A's and B's signature will be valid with probability p_A and p_B, respectively. The idea is to keep p_A and p_B close to each other so that neither A nor B will have any significant advantage in stopping the protocol.

[BOGMR90] suggests to achieve this as follows. Initially A and B agree on a contract, c, setting the parameters for the exchange and sign this contract. This signature does not count as a signature on the contract to be signed, and they can freely exchange it. Now, in round i of the protocol, A signs a message saying "c is valid with probability p_i". This signed message is sent to B, who returns a similar, signed message to A. The probability, p_i, is increased gradually in each round (e.g., with 100 exchanges $p_i = \frac{i}{100}$ for $i = 1, 2, \ldots, 100$).

In case of a dispute a party will present the signed message containing the highest value of p_i to the arbiter, which simply decides to accept the signature with probability p_i (once the signature is accepted or rejected, all future verdicts have the same outcome).

The disadvantage of this protocol is that each party has to make and verify many signatures. In [Dam94] this was somewhat solved by replacing digital signature in each round with the preimage of a one-way function (e.g., a hash function). More precisely in the initial phase A signs c and $(\mathcal{H}(a_1), \mathcal{H}(a_2), \ldots, \mathcal{H}(a_n))$, where $a_1, a_2, \ldots, a_n$ are chosen at random. Similarly, B chooses $b_1, b_2, \ldots, b_n$ at random and signs c and the list $(\mathcal{H}(b_1), \mathcal{H}(b_2), \ldots, \mathcal{H}(b_n))$. In round i party A reveals a_i and B reveals b_i. After a_i and b_i are revealed, the arbiter will accept

A's and B's signatures, respectively, with probability $\frac{i}{n}$ (more generally arbitrary probabilities can be associated with each a_i and b_i as part of the initial contract).

While this is more efficient, it still requires a number of preimages to be signed and stored. This can be avoided based on the technique for micropayments presented in [Ped97,RS97,HSW96,AMS97], as the list $(\mathcal{H}(a_1), \mathcal{H}(a_2), \ldots, \mathcal{H}(a_n))$ can be replaced by $\mathcal{H}^n(a_0)$, where a_0 is chosen at random. B's list is similarly replaced by $\mathcal{H}^n(b_0)$, for a random b_0. In the i'th round A and B send $a_i = \mathcal{H}^{n-i}(a_0)$, respectively $b_i = \mathcal{H}^{n-i}(b_0)$, to each other. This reduces storage requirements, and the initially signed message will be shorter. In the i'th round each party will have to iterate $\mathcal{H}$ $n - i$ times in order to find a_i, but if a good hash function is chosen (e.g., SHA-1, [SHS95]) this can be done very fast. As A (B) only has to remember the previous b_i (a_i) verification requires a single computation with $\mathcal{H}$ in each step.

5.3 Optimistic Protocols

While the two types of exchange mentioned above both require a large number of rounds in order to gradually increase the faith in the signature, optimistic protocols aim at exchanging the signatures directly. In order to cope with dishonest parties, this is done in such a way that an honest party can be saved by a third party. The first optimistic protocols for fair exchanges were published in [ASW97]. In the following we sketch the protocols presented in [ASW98] for the case of Schnorr signatures. The following concepts are used:

Reduced signature Briefly, a reduced signature is a partial signature. It can be verified given the public key, and given additional information the correct signature can be derived from the reduced one. A reduced Schnorr signature (c, z) on message m is, for example, given by (c, u), where $u = g^z \bmod p$. It is correct if $c = \mathcal{H}(uh^{-c}, m)$.

Verifiable encryption of homomorphic inverse Given a surjective homomorphism $\varphi : G_1 \to G_2$, where G_1 and G_2 are groups, a verifiable encryption of the inverse of some $d \in G_2$ is an encryption of $\varphi^{-1}(d)$ for which it can be proved that decryption gives a pre-image of d.

In the following consider the homomorphism $\varphi : (\mathbb{Z}_q, +) \to (G_q, \cdot)$ defined by $x \mapsto g^x$, where G_q is a cyclic group of order q generated by g.

Assuming A wants to sign m_A and B the message m_B the protocol works as follows:

1. A computes a Schnorr signature (c_A, z_A) on m_A and the corresponding reduced signature (c_A, u_A), where $u_A = g^{z_A}$. A sends (c_A, u_A) to B.
2. B verifies the reduced signature: $c = \mathcal{H}(u_A h^{-c_A}, m_A)$. If it is invalid, B stops. Otherwise B signs m_B to get (c_B, z_B) and the corresponding reduced signature (c_B, u_B). The reduced signature is sent to A.
3. A verifies the reduced signature. If it is invalid, A stops. A encrypts z_A under the public key of the third party and proves that the encryption is an inverse of u_A under φ.

4. B aborts if the proof is not accepted. Otherwise B encrypts z_B under the public key of the third party and proves that the encryption is an inverse of u_B under φ.
5. If A rejects the proof, the third party is invoked. See below. Otherwise A sends z_A to B.
6. B verifies that $u_A = g^{z_A}$. If this is not satisfied B retrieves the signature with the help of the third party. Otherwise, B has the required signature and sends z_b to A.
7. A verifies that $u_B = g^{z_B}$. If this is satisfied, A has the required signature on m_B. Otherwise, A retrieves the signature with the help of the third party.

Please consult [ASW98] for details on verifiable encryption and a detailed proof of security. In the following the security of the protocol is described informally.

If both parties follow the protocol, they will end up getting each others signature. The interesting thing is what happens if something goes wrong. Up to and including Step 3 neither A nor B has revealed their respective signature. Thus after a failure nothing has to be done. The same is true if A's proof in Step 4 is rejected. So let's consider failures in Step 5, 6 and 7.

A failure in Step 7 means that B got A's signature, but A did not get B's. Now A can bring the encryption of z_B to T and get the decrypted value. T can do this without introducing security holes, as B would only have sent the encryption of z_B if he had received a correct encryption of z_A (hence T can help B if necessary).

A failure in Step 6 means that both A and B have received correct encryptions of z_B and z_A respectively. In this case, they will be able to get the decryptions from T. However, in order to get z_A, B must supply z_B so that A can get it in case B (being dishonest) claimed this error after getting the encryption of z_A in Step 4.

A failure in Step 5 means that A has provided a correct encryption of z_A, but B did not send one to A. Since B can get z_A from T by using the mechanism handling failures in Step 6 there are two possibilities, when A shows up at T:

- If B has not requested z_A, T can mark the exchange as aborted, and will never send z_A to B (if at some previous point A has requested z_A, T will not mark the protocol as aborted).
- If B at some point has requested z_A then T can immediately send z_B to A (as B had to supply it).

6 Electronic Payments

Payment systems allowing a person to pay electronically to another person are essential for electronic commerce. As a result a large number of (Internet) payment systems have recently been developed. It is out of the scope of this paper to describe all these systems. In stead a general model is presented and a few constructions of electronic cash are discussed, as these have drawn most attention in the cryptographic community.

6.1 Model

The following model is based on [AJSW97]. Electronic payments involve four parties (or roles, as one entity in principle can play different roles). These are

- the payer,
- the payee (receiving the money)
- the issuing bank (the bank or financial institution of the payer)
- and the acquiring bank (bank or financial institution of the payee)

In a clean payment system structure, the payer will only have to deal with the issuer and payee, while payee only has to deal with the payer and the acquirer. Thus we have relations as depicted in Figure 4.

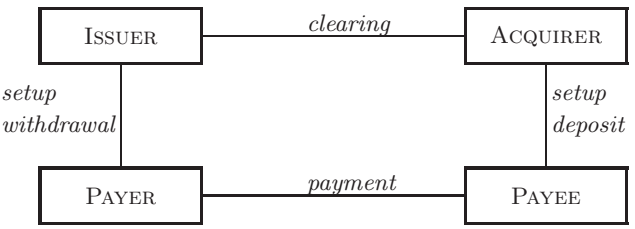


Fig. 4. Model of electronic payments

Most payment systems adhere to this model, but there are exceptions (e.g., some flows suggested by [FST95], and some options in [IBM], where the payer can contact the acquirer or the payee may have to contact the issuer).

Based on this model a number of different types of payment systems are possible.

Cash-like systems These are prepaid systems in which the user gets electronic tokens representing money from the issuer.

Account based systems Here the payer sends a token to the payee, which allows the acquirer, when the payee deposits it, to move the amount from the account of the payer to that of the payee. Credit-card payments and electronic cheques are common examples of such systems.

Indirect payment These are systems, where the payer instructs his bank directly to transfer money to the account of the payee. Such systems are typically applied from homebanking applications, where the payer initiates the payment, but they can also be initiated by the payee (based on agreements with the payer).

In the following we focus on cash-like systems, but first a few words on the principles of account based payment systems. These roughly work as follows

given any digital signature scheme. During set-up the user is given a certificate from the issuer on the public key-pair used to sign payment messages. The issuer uniquely links this key-pair to the (account of the) payer. A payment is simply a signed message enabling the acquirer to transfer the paid amount from the account of the payer to that of the payee. In SET encryption measures are provided, which makes it possible to tunnel credit card information from the payer to the acquirer through the payee, in such a way that the payee cannot read this information (see [MV97] for more details on SET). Electronic cheques, which provide an electronic equivalent of paper cheques, is another example of an account based system (see [FST95,Cry]).

6.2 Cash-like Payment Systems

In a cash-like system, the payer withdraws electronic money at the issuer. During payment some of this is transferred to the payee (this may happen as part of a protocol in which a number of messages are exchanged), and finally the payee can deposit the received money at the acquirer. Clearing between issuer and acquirer takes place afterwards.

As these systems resemble normal cash much effort has been put into the construction of such schemes supporting the same properties that real cash enjoys. In particular real cash can be used anonymously and it can be transferred between users (i.e., the payee can use received acting as a payer in another payment [CP93]). Furthermore, being prepaid, cash-like systems will only be used for minor amounts, and operating them should therefore be inexpensive and preferably fast (in terms of communication as well as computation).

As electronic cash can be authenticated (guaranteed) using digital signatures, the main security problem is to prevent forgeries through copying. There are basically two solutions to this:

- prevent copying using secure hardware (smart cards);
- prevent usage of copied cash by on-line queries to a central server.

For anonymous payments (and privacy protecting transactions in general) it is important to identify the level of anonymity offered. In order to define these properly, it is necessary to use the notion of a *view* as introduced by [GMR89]. We refer to [GMR89] for a formal definition and introduction of protocols and views. Informally, the view of a party involved in the execution of some protocol consists of

- All inputs given to that party.
- All messages received during the execution of the protocol.
- All random bits used by that party.

In principle one could add "All message sent by that party", but as these messages can be computed from the view defined above, they are usually excluded.

In the definition below both unconditional and computational anonymity is considered. The term "not feasible" means that under some cryptographic assumption (e.g., the difficulty of factoring) the required task cannot be done, while

"not possible" means that it cannot be done no matter how much computing power is available.

A protocol provides *untraceability* for a party, A, if given the view of one execution of the protocol with A it is not possible (feasible) to identify A (the execution can not be traced to A).

A protocol provides *unlinkability* for a party, A, if it provides untraceability and given all views of all other parties executing that protocol with A it is not possible (feasible) to see which views originate from an execution with the same party.

Obviously, a protocol only providing untraceability may not protect the anonymity of A, since linking all A's transactions together may identify A uniquely.

In the following we describe two prepaid systems providing *unconditional anonymity*, one on-line and one off-line. Both schemes are based on blind signatures, which are briefly explained next.

6.3 Blind Signatures

As introduced in [Cha83] blind signatures enables a signer to sign a message without seeing the message. More precisely, if the signer makes n signatures for some $n > 0$ and later sees n pairs (m_i, σ_i) where σ_i is the signature on m_i then the signer is not able to tell when he made any of these signatures.

At first, blind signatures seem like an odd idea, as we all learn to read documents carefully before signing them. However, blind signatures are intended to be used for anonymous electronic money. The idea is that a bank issuing electronic coins (or cheques) can sign these using a blind signature scheme. When the money has been spent and the recipient of the coin wants to deposit it, the acquirer can tell that the coin is valid (because of the signature), but the coin does not contain information which allows the issuer, acquirer and payee together to tell who originally withdrew it.

The most famous example of blind signatures is based on RSA. With the notation from the introduction this works as follows. In order to get a blind signature on a message, m, the signer is requested to sign the number $b = r^e \mathcal{H}(m) \bmod n$, where r is chosen at random. Thus the signer computes and returns $b^d \bmod n$, from which the required signature, $\mathcal{H}(m)^d$, can be computed as $(b^d)/r \bmod n$.

6.4 On-Line Coin System

Given any blind digital signature scheme (e.g., based on RSA as described above) an on-line coin system works as follows (see [Cha83]):

Withdrawal During withdrawal the payer gets a number of coins from the issuer. Each coin is represented by the issuer's signature, σ on $\mathcal{H}(m)$, where m is a random message selected by the payer. Different coins (denominations) can be implemented using different key-pairs at the issuer (as described in the certificate of the issuer).

Payment During payment of a coin the payer simply sends (m, σ) to the payee. The payee asks the acquirer if the received coin can be accepted (to ensure that it is not a copy of a previously spent coin) before accepting or rejecting the payment.

Deposit Done during the payment transaction.

Since the issuer when making a blind signature can get no (Shannon) information about (m, σ) the payment system provides unconditional unlinkability.

6.5 Off-Line Coin System

In the above system, the on-line query during the payment is necessary to prevent that coins are spent more than once. In an off-line system it is not possible to prevent such double-spending by cryptographic means alone (secure hardware is needed). In stead [CFN90] suggested to enable identification of double-spenders after the fact. Thus honest users are anonymous, while cheaters can be identified. This can for example be done as follows based on RSA ([Bra94,Bra95,BBC+94] present different schemes based on other blind signature schemes):

Withdrawal During withdrawal the payer prepares $2k$ messages of the form $\mathcal{H}(m_i)||\mathcal{H}(m_i \oplus Id)$ for $i = 1, 2, \ldots, 2k$, where Id is a unique identifier of the payer, and k is a security parameter. Each of these $2k$ message are blinded using a random number r_i (as in Section 6.3) and sent to the issuer.
The issuer request to get $(m_i, m_i \oplus Id)$ plus the corresponding blinding factors, r_i for k values of i. If all these are correct, the bank is assured that most of the remaining messages are also constructed correctly, and these are signed.
From the signature of the bank the payer can get a blind signature on each unrevealed message $\mathcal{H}(m_i)||\mathcal{H}(m_i \oplus Id)$.

Payment The payee selects a random k-bits challenge, $(c_1, c_2, \ldots, c_k)$ and sends it to the payer. The payer responds with $(m_i, \mathcal{H}(m_i \oplus Id))$ if $c_i = 0$ and $\mathcal{H}(m_i)||(m_i \oplus Id)$ if $c_i = 1$. The payee then verifies that the k signatures are correct.

Deposit During deposit the payee sends the received signed messages (plus signature) and the challenge to the acquirer.
The acquirer looks up if this coin has been used before. If they have been used with a different challenge c' then $c_i \neq c'_i$ for some i and hence the acquirer has both m_i and $m_i \oplus Id$ and knows the identity of the payer.

If the payer does not cheat it is not feasible to identify the payer unless preimages of $\mathcal{H}$ can be inverted. Please consult [CFN90] on how unconditional anonymity can be achieved.

6.6 Micropayment

The above systems require verification of at least one signatures during each payment. This may not be sufficiently efficient in systems requiring many payments of small amounts (in particular if this is done on a smart card). Such

payments can for example be made efficiently using the technique mentioned in Section 5. During payment the payer sends to the payee $a_n = \mathcal{H}^n(a_0)$ as well as a signature on a_n guaranteeing that a_n presents some value (say $n = 100$ and the total value is 10 dollars). After the payee has validated this signature, the payer can make a number of successive payments of 10 cents by sending the number $a_{n-i} = \mathcal{H}^{n-i}(a_0)$for the i'th payment. The payee only needs to remember the last value received. Validation of the i'th payment consists of verifying that $\mathcal{H}(a_{n-i}) = a_{n-(i-1)}$, which can be done very efficiently.

Development of efficient schemes for micropayment has received considerable attention recently. Other schemes are given in [DEC,JO97].

7 Conclusion

This paper has introduced the most widely applied digital signature schemes and discussed some applications of these related to electronic commerce. The security of the various protocols and mechanisms has only been described informally. As sound cryptographic design requires proofs based on acceptable cryptographic assumptions the reader is strongly encouraged to consult the original papers for full proofs of the security.

References

[1/S91] ISO/IEC JTC 1/SC27. Information technology - Security techniques - Digital signature schemes giving message recovery - Part 1: Mechanisms using redundancy. Final draft of ISO International Standard 9796-1, 1991. 137

[1/S97a] ISO/IEC JTC 1/SC27. Information technology - Security techniques - Non repudiation - Part 1: General Model. ISO International Standard 13888-1, 1997. 144, 145

[1/S97b] ISO/IEC JTC 1/SC27. Information technology - Security techniques - Non repudiation - Part 3: Using asymmetric techniques. ISO International Standard 13888-3, 1997. 144

[1/S98a] ISO/IEC JTC 1/SC27. Information technology - Security techniques - Digital signatures with appendix - Part 3: Certificate-based mechanisms. Final draft of ISO International Standard 14888-3, 1998. 135

[1/S98b] ISO/IEC JTC 1/SC27. Information technology - Security techniques - Digital signatures with appendix - Part 1: General. Final draft of ISO International Standard 14888-1, 1998. 137

[AJSW97] N. Asokan, Phil Janson, Michael Steiner, and Michael Waidner. State of the art in electronic payment systems. *IEEE Computer*, 30(9):28–35, September 1997. 150

[AMS97] R. Anderson, C. Manifavas, and C. Sutherland. NetCard – A Practical Electronic-Cash System. In *Security Protocols*, Lecture Notes in Computer Science, pages 49–58. Springer-Verlag, 1997. 148

[ASW97] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic Protocols for Fair Exchange. In *4th ACM Conference on Computer and Communications Security*, pages 6–17. ACM Press, April 1997. 148

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic Fair Exchange of Digital Signatures. In *Advances in Cryptology - proceedings of EUROCRYPT 98*, number 1403 in Lecture Notes in Computer Science, pages 591–606, Berlin, 1998. Springer-Verlag. 148, 149

[BBC+94] J.-P. Boly, A. Bosselars, R. Cramer, R. Michelsen, S. Mjølsnes, F. Muller, B.Pfitzmann, P. de Rooij, B. Schoenmakers, M. Schunter, L. Vallé, and M. Waidner. The ESPRIT Project CAFE — High Security Digital Payment Systems. In *Computer Security — ESORICS'94*, volume 875 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994. 153

[BCDvdG88] E. Brickell, D. Chaum, I. Damgård, and J van de Graaf. Gradual and Verifiable Release of a Secret. In *Advances in Cryptology - proceedings of CRYPTO 87*, Lecture Notes in Computer Science, Berlin, 1988. Springer-Verlag. 146, 147

[BM92] M. Bellare and S. Micali. How to Sign given any Trapdoor Permutation. *Journal of the Association for Computing Machinery*, 39:214–233, 1992. 136

[BOGMR90] Ben-Or, Goldreich, Micali, and Rivest. A Fair Protocol for Signing Contracts. *IEEE trans. on Information Theory*, 36:40–46, 1990. 147

[BR93] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *First ACM Conference on Computer and Communications Security*, 1993. 136

[Bra94] S. Brands. Untraceable Off-line Cash in Wallet with Observers. In *Advances in Cryptology - proceedings of CRYPTO 93*, Lecture Notes in Computer Science, pages 302 –318. Springer-Verlag, 1994. 153

[Bra95] S. Brands. Off-Line Electronic Cash Based on Secret-Key Certificates. In *Proceedings of LATIN'95*, 1995. Also available as CWI technical report, CS-R9506. 153

[CFN90] D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. In *Advances in Cryptology - proceedings of CRYPTO 88*, Lecture Notes in Computer Science, pages 319 – 327. Springer-Verlag, 1990. 153

[Cha83] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology - proceedings of CRYPTO 82*, pages 199 – 203, 1983. 152

[CP93] D. Chaum and T. P. Pedersen. Transferred Cash Grows in Size. In *Advances in Cryptology - proceedings of EUROCRYPT 92*, Lecture Notes in Computer Science, pages 390 – 407. Springer-Verlag, 1993. 151

[Cry] Cryptomathic. Mandate. See http://www.cryptomathic.dk. Describes a payment system based on electronic cheques. 151

[Dam94] I. Damgård. Practical and Provably Secure Release of a Secret and Exchange of Signatures. In *Advances in Cryptology - proceedings of EUROCRYPT 93*, number 765 in Lecture Notes in Computer Science, pages 200–217, Berlin, 1994. Springer-Verlag. 146, 147

[DEC] DEC. Milliscent. See http://www.millicent.digital.com/. 154

[DH76] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Trans. Inform. Theory*, IT–22(6):644–654, November 1976. 134, 135

[DSS93] Digital Signature Standard. Federal Information Processing Standards Publication 186, U.S. National Institute of Standards and Technology (NIST), February 1993. Draft. 134, 135

[EG85] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology - proceedings of CRYPTO 84*, Lecture Notes in Computer Science, pages 10 –18. Springer-Verlag, 1985. 135, 141

[FFS88] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988. 136

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - proceedings of EUROCRYPT 86*, Lecture Notes in Computer Science, pages 186 – 194. Springer-Verlag, 1987. 136

[FST95] FSTC. Electronic check proposal. Technical report, Financial Services Technology Consortium, 1995. 150, 151

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen Message Attack. *SIAM Journal on Computing*, 17(2):281 – 308, April 1988. 136, 138

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems. *SIAM Journal of Computation*, 18(1):186–208, 1989. 136, 151

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proceedings of the 27th IEEE Symposium on the Foundations of Computer Science*, pages 174–187, 1986. 146

[GQ89] 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 - proceedings of EUROCRYPT 88*, Lecture Notes in Computer Science, pages 123–128. Springer-Verlag, 1989. 136

[HS91] S. Haber and W.S. Stornetta. How To Time-Stamp a Digital Document. *Journal of Cryptology*, 3(2):99–111, 1991. 146

[HSW96] R. Hauser, M. Steiner, and M. Waidner. Micro-Payments based on iKP. Technical Report RZ 2791, IBM Zürich Research Laboratory, February 1996. 148

[IBM] IBM. See http://www.hrl.il.ibm.com/mpay/. 150

[IS0] Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT) — Application Level Syntax Rules, Part 9: Security Key and Certificate Management Message (KEYMAN). ISO/DIS 9735-9. 142

[JO97] S. Jarecki and A. M. Odlyzko. An efficient micropayment system based on probabilistic polling. In *Financial Cryptography*, Lecture Notes in Computer Science, pages 173–191. Springer-Verlag, 1997. 154

[MV97] Mastercard and Visa. *SET Secure Electronic Transactions Protocol*, version 1.0 edition, May 1997. Book One: Business Specifications, Book Two: Technical Specification, Book Three: Formal Protocol Definition. Available from http://www.setco.org/set_specifications.html. 151

[NY89] M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. In *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing*, pages 33–43, 1989. 136

[Ped97] T. Pedersen. Electronic Payments of Small Amounts. In *Security Protocols*, Lecture Notes in Computer Science, pages 59–68. Springer-Verlag, 1997. 148

[PKI] Public-Key Infrastructure (X.509) (pkix). Internet Working Group. The task of the working group is to develop standards needed to support an X.509-based PKI. 144, 145

[Rab79] M. O. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as factorization. Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, MIT, January 1979. 135

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, 1990. 136

[RS97] R. Rivest and A. Shamir. PayWord and MicroMint: Two Simple Micropayment Schemes. In *Security Protocols*, Lecture Notes in Computer Science, pages 69–88. Springer-Verlag, 1997. 148

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21, 1978. 134, 135

[Sch90] C. P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - proceedings of CRYPTO 89*, Lecture Notes in Computer Science, pages 239 – 252. Springer-Verlag, 1990. 136

[SHS95] Secure Hash Standard. Federal Information Processing Standards Publication 180-1, U.S. National Institute of Standards and Technology (NIST), April 1995. 141, 148

[Wil80] H. C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transaction on Information Theory*, 26(6):726 – 729, 1980. 135

[X5095] Information technology – open systems interconnection – the directory: Authentication framework. ISO/IEC 9594-8, 1995. See also Amendment 1 to ISO/IEC 9594-8:1995 – Certificate extensions. 142

[YHK95] Y. Yeong, T. Howes, and S. Kille. Lightweight Directory Access Protocol. RFC 1777, March 1995. 143

The State of Cryptographic Hash Functions

Bart Preneel*

Katholieke Universiteit Leuven, Dept. Electrical Engineering-ESAT
Kardinaal Mercierlaan 94, B–3001 Heverlee, Belgium
bart.preneel@esat.kuleuven.ac.be

Abstract. This paper describes the state of the art for cryptographic hash functions. Different definitions are compared, and the few theoretical results on hash functions are discussed. A brief overview is presented of the most important constructions, and some open problems are presented.

1 Introduction

Hash functions are well known in computer science. They compress a string of arbitrary length to a string of fixed length and are used to allocate as uniformly as possible storage for the records of a file. For cryptographic applications, additional security requirements are necessary: informally, one requires that they are hard to invert, and (in most cases) that it is hard to find colliding inputs. This is achieved by creating a mapping that associates to an input string a 'randomly looking' output string (note that formally this makes no sense, as hash functions are deterministic mappings). As a consequence of these properties, hash functions create a 'unique' relationship between the input and the hash value; there are of course many inputs corresponding to a single output, but it is hard to identify these.

Cryptographic hash functions can be used to protect the integrity of large amounts of information (such as the content of a hard disk, a set of financial transactions, a software program) by the integrity of a short string, the hash result. This protection can be achieved by digitally signing this short string [31], or by writing down the string on a piece of paper that is stored in a secure place. This is analogous to conventional message encryption, that replaces the secrecy of a large amount of information by the secrecy (and authenticity) of a secret key; typically this key is much shorter than the message.

Hash functions have been used (and sometimes abused) for many other cryptographic applications. Examples include

- the protection of pass-phrases (where the image of the pass-phrase under the hash function is stored in the computer, rather than the pass-phrase itself);
- the commitment to a string without revealing it (see Damgård *et al.* [27]).

* F.W.O. postdoctoral researcher, sponsored by the Fund for Scientific Research – Flanders (Belgium).

I. Damgård (Ed.): Lectures on Data Security, LNCS 1561, pp. 158–182, 1999.

– the construction of Message Authentication Codes (MACs), by introducing in the hash function a second parameter that is kept secret [5,7,70];
– key derivation, for example to derive session keys from a transaction number and a master key.

The last two applications assume that the hash function keyed by a second parameter yields a pseudo-random function, or a mapping that is hard to predict. Hash functions have also been used to instantiate 'random oracles' [9]; this requires even stronger properties.

The remainder of this paper is organized as follows. Section 2 presents the definitions and Sect. 3 discusses generic constructions and security results. Specific constructions are treated in Sect. 4, while conclusions and open problems are presented in Sect. 5.

2 Definitions

One distinguishes between hash functions that have only a single input, and hash functions that have a second input. The first type are sometimes called MDCs or Manipulation Detection Codes; many authors refer to this type simply as cryptographic hash functions, or even hash functions. If the second parameter is secret, one calls these functions keyed hash functions; an important subclass are MACs or Message Authentication Codes. Finally the second parameter can also be public; examples in this class are the UOWHFs or Universal One-Way Hash Functions.

Note that one should not confuse hash functions with checksums that are used for error detection or correction (such as the well known Cyclic Redundancy Checks or CRCs).

First we define one-way hash functions (OWHF) and collision resistant hash functions (CRHF). Both classes of hash functions are studied in this paper. Then we look at some related concepts, namely message authentication codes (MACs), universal one-way hash functions (UOWHFs) and universal hash functions.

In the following the hash function will be denoted with h, and its argument, i.e., the information to be protected with x. The image of x under the hash function h will be denoted with $h(x)$. It will be assumed that the description of the hash function h is publicly known, and that it does not require any secret information (except for the optional parameter, which may be secret). A second assumption is that given the inputs, the computation of the hash result must be *"easy."*

2.1 One-Way Hash Function (OWHF)

The concept of one-way hash functions was introduced by Diffie and Hellman in [31]. The first informal definition was apparently given by Merkle [59,60] and Rabin [73]. A **one-way hash function** is a function h satisfying the following conditions:

1. The argument x can be of *arbitrary* length and the result $h(x)$ has a *fixed* length of n bits (with $n \geq 64 \ldots 80$).
2. The hash function must be *one-way* in the sense that given a y in the image of h, it is "hard" to find a message x such that $h(x) = y$ (preimage resistant) and given x in the domain of h and $h(x)$ it is "hard" to find a message $x' \neq x$ such that $h(x') = h(x)$ (second preimage resistant).

Note that this last condition (finding a second preimage is "hard") differs from the intuitive concept of one-wayness, namely that it is "hard" to find a preimage x given only h and the value of $h(x)$. It is clear that for permutations or injective functions only preimage resistance is relevant. The relation between preimage resistance and second preimage resistance is discussed in [58,67].

The above definition is only informal. For example, one should specify the distribution that is used to select y respectively x, and specify what "hard" and "easy" means. The definition should also take into account that one can always invert functions with a small range (by exhaustive search) and that one can always precompute the function in a small set. There are many ways to formalize the definition, and it is a non-trivial exercise to show which of these are equivalent.

For a formal definition, we need to specify a model of computation. Rather than probabilistic Turing machines (that are used traditionally in cryptography), we will follow here Bellare and Rogaway [10] and use the RAM model including pointers (see for example [22]); execution time is measured with respect to that model. An *adversary* is a program for this model, written in some fixed programming language. The adversary has access to random bits.[1] The running time includes the actual execution time and the length of the program description.

The set of all integers will be denoted with $\mathbf{N}$. The alphabet considered is the binary alphabet $\Sigma = \{0, 1\}$. For $n \in \mathbf{N}$, Σ^n is the set of all binary strings of length n. The set of all strings of arbitrary length will be written as Σ^*. The size of a set S is denoted with $|S|$. Let h be a function with domain $D = \Sigma^*$ and range $R = \Sigma^n$. Note that in fact we will only consider inputs of bit length $l(n)$, with $l(n)$ a function that satisfies $l(n) > n$. This is not a real restriction, since it is not possible (in practice) to evaluate a function in inputs that are too large.

Definition 1. *A* **one-way** *hash function H is a function with domain $D = \Sigma^{l(n)}$ and range $R = \Sigma^n$ that satisfies the following conditions:*

– *preimage resistance: let x be selected uniformly in D and let M be an adversary that on input $h(x)$ uses time $\leq t$ and outputs $M(h(x)) \in D$. For each adversary M,*

$$\Pr_{x \in D} \{h(M(h(x))) = h(x)\} < \epsilon \,.$$

Here the probability is also taken over the random choices of M.

[1] Returning a random integer in the interval $[1, n]$ takes time $\Theta(\log n)$.

– *2nd preimage resistance: let x be selected uniformly in $\Sigma^{l(n)}$ and let M' be an adversary that on input x uses time $\leq t$ and outputs $x' \in D$ with $x' \neq x$. For each adversary M',*

$$\Pr_{x \in D} \{M'(x) = h(x)\} < \epsilon\,.$$

Here the probability is also taken over the random choices of M'.

The above definition is only meaningful if t/ϵ is large; it is clear that $t/\epsilon \leq 2^n$. One can specify a constant as required by a given application.

Note that this model does not take into account precomputation, which can be used to speed the computation of many preimages (as for example in Hellman [49]).

2.2 Collision Resistant Hash Function (CRHF)

The fact that a function is one-way does not mean that it is hard to find two colliding inputs; it will be shown in Sect. 3.2 that the effort to find a collision is only the square root of the effort to find a (2nd) preimage. This motivates the definition of a collision resistant function as a separate primitive. Note that some authors call this a collision free hash function [24,25,26], or a collision intractible hash function [89].

The first formal definition of a CRHF was given by Damgård [24,25]. An informal definition was given by Merkle in [60]. A **collision resistant hash function** is a function h satisfying the following conditions:

1. The argument x can be of *arbitrary* length and the result $h(x)$ has a *fixed* length of n bits (with $n \geq 128 \ldots 160$).
2. The hash function must be *one-way*, i.e., preimage resistance and second preimage resistant.
3. The hash function must be *collision resistant:* this means that it is "hard" to find two distinct messages that hash to the same result.

Finding a second preimage cannot be easier than finding a collision: therefore the second preimage condition in this definition is redundant. However, preimage resistance is not necessarily implied by collision resistance (note that it is required for certain applications). Damgård provides some definitions and conditions under which collision resistance implies preimage resistance [26]; see also Gibson's comment in [42].

Formalizing a collision resistant hash function is not as straightforward as formalizing a one-way hash function. One cannot specify that there should not exist an adversary that outputs a collision: since there are many colliding inputs that are very short, there will be many efficient adversaries that can output a pair of inputs that collide under x (just include two such inputs in the code). The way out of this problem is to define collision resistance as the property of a family $\mathcal{H}$ of hash functions, that is, a set of functions indexed by a parameter S. For simplicity, it will be assumed that S is taken from the parameter space Σ^s. Thus $\mathcal{H}$ is a mapping from $D \times \Sigma^s \to R$, and an individual function in this family is denoted with $h_S : D \to R$.

Definition 2. *A* **collision-resistant** *hash function $\mathcal{H}$ is a function family with domain $D = \Sigma^{l(n)}$ and range $R = \Sigma^n$ that satisfies the following conditions:*

- *the functions h_S are preimage resistant and second preimage resistant (cf. Definition 1).*
- *collision resistance: let F be a collision string finder that on input $S \in \Sigma^s$ uses time $\leq t$ and outputs either "?" or a pair $x, x' \in \Sigma^{l(n)}$ with $x' \neq x$ such that $h_S(x') = h_S(x)$. For each F,*

$$\Pr_S \{F(\mathcal{H}) \neq \text{“?”}\} < \epsilon\,.$$

Here the probability is also taken over the random choices of F.

Again this definition is only meaningful if t/ϵ is large; it will follow from Sect. 3.2 that an upper bound is $2.24 \cdot 2^{n/2}$.

2.3 Universal One-Way Hash Function

The concept of a UOWHF was introduced by Naor and Yung [64]. In [10], Bellare and Rogaway give a concrete definition (rather than an asymptotic one) and study practical constructions for this primitive under the name 'target collision resistant hash functions.'

Definition 3. *A* **universal one-way hash function** *$\mathcal{H}$ is a function family with domain $D = \Sigma^{l(n)}$ and range $R = \Sigma^n$, that satisfies the following condition:*

- *Let $F' = (F'_1, F'_2)$ be an adversary. F'_1 is run first; it is an algorithm that produces x and possibly some state information; this information is passed on to F'_2. Algorithm F'_2 is given S, x and the state information and outputs either "?" or an $x' \neq x$ such that $h_S(x') = h_S(x)$.*
  For each F' that runs in time $\leq t$ (that is, the sum of the running times of F'_1 and F'_2),

$$\Pr_S \{F'(\mathcal{H}) \neq \text{“?”}\} < \epsilon\,.$$

Here the probability is also taken over the random choices of F'.

The main difference with collision resistance is that here the input x is fixed first, and then the parameter S is chosen. This implies that finding collisions for a given value of S does not help an attacker.

Naor and Yung show that a UOWHF can be used to build a digital signature scheme [64]. Rompel [78] developed a (very inefficient) construction for a UOWHF based on any one-way function; in this way he reduces a digital signature scheme to a one-way function, which is the weakest possible assumption. A UOWHF can replace a CRHF in a digital signature scheme if the signer does not intend to repudiate her signatures (see also Sect. 3.2).

2.4 Message Authentication Code (MAC)

Message Authentication Codes have been used for a long time in the banking community. However, MAC algorithms with good cryptographic properties were only introduced in the late 1970s. The first reference to a MAC algorithm is a 1972 patent application by Simmons *et al.* (reference 10. in [81]).

Definition 4. *A* **MAC algorithm** *is a function h satisfying the following conditions:*

1. *The argument x can be of* arbitrary *length and the result $h_K(x)$ has a* fixed *length of n bits (with $n \geq 32 \ldots 64$).*
2. *Given h and x, it is "hard" to forge a MAC on a new message, that is, to determine $h_K(x)$ with a probability of success "significantly higher" than $1/2^n$. Even when many pairs $\{x_i, h_K(x_i)\}$ are known, where the x_i have been selected (sequentially) by the opponent, it is "hard" to compute $h_K(x')$ for any $x' \neq x_i$.*

The last attack is called an adaptive chosen text attack. One distinguishes between forgery attacks and key recovery attacks: a forgery allows to determine the MAC on a new message; a key recovery attack is much more serious as it allows to forge the MAC for an arbitrary message.

The exact security of a MAC algorithm can be expressed in terms of the running time t of the adversary, the number of known and (adaptively) chosen texts an adversary has access to, and the probability of success ϵ of the forgery attack (see e.g., [7]).

A MAC algorithm can be used for message authentication between a sender and a receiver who share a secret key K. In order to protect a message, the sender applies the MAC algorithm to the message and appends the resulting string to the message. On receipt of the message, the receiver recomputes the MAC and verifies that it corresponds to the transmitted MAC value. An active eavesdropper Eve can modify the message, but as she does not know the secret key, she cannot predict the MAC value for the modified message.

2.5 Universal Hash Functions

Universal hash functions are combinatorial objects, which implies that they can be defined without using a model of computation. They were introduced by Carter and Wegman [15,86], who show that they can be applied to efficient unconditionally secure message authentication. In this way they found practical constructions for the authentication codes introduced by Simmons in the 1970s [80]. The first published reference to authentication codes is Gilbert *et al.* [44]. Other applications of universal hash functions include interactive proof systems, pseudo-random number generation, complexity theory, and probabilistic algorithms.

A universal hash function is a mapping from a finite set A with size a to a finite set B with size b. For a given hash function h and for a pair (x, x') with

$x \neq x'$ the following function is defined:

$$\delta_h(x, x') = \begin{cases} 1 & \text{if } h(x) = h(x') \\ 0 & \text{otherwise.} \end{cases}$$

As above, a finite set of hash functions will be denoted *family* $\mathcal{H}$ of hash functions. Now $\delta_{\mathcal{H}}(x, x')$ is defined as $\sum_{h \in \mathcal{H}} \delta_h(x, x')$, or $\delta_{\mathcal{H}}(x, x')$ counts the number of functions in $\mathcal{H}$ for which x and x' collide. If a random choice of h is made, then for any two distinct inputs x and x', the probability that these two inputs yield a collision equals $\delta_{\mathcal{H}}(x, x')/|\mathcal{H}|$. For a universal hash function, the goal is to minimize this probability together with the size of $\mathcal{H}$.

Definition 5. *Let ϵ be any positive real number. An* **ϵ-almost universal** *family (or ϵ-AU family) $\mathcal{H}$ of hash functions from a set A to a set B is a family of functions from A to B such that for any distinct elements $x, x' \in A$*

$$|\{h \in \mathcal{H} : h(x) = h(x')\}| = \delta_{\mathcal{H}}(x, x') \leq \epsilon \cdot |\mathcal{H}| \ .$$

This definition states that for any two distinct inputs the probability for a collision is at most ϵ. In [15] the case $\epsilon = 1/b$ is called universal (the smallest possible value for ϵ is $(a - b)/b(a - 1)$).

Definition 6. *Let ϵ be any positive real number. An* **ϵ-almost strongly universal** *family (or ϵ-ASU family) $\mathcal{H}$ of hash functions from a set A to a set B is a family of functions from A to B such that*

- *for every $x \in A$ and for every $y \in B$, $|\{h \in \mathcal{H} : h(x) = y\}| = |\mathcal{H}|/b$,*
- *for every $x_1, x_2 \in A$ $(x_1 \neq x_2)$ and for every $y_1, y_2 \in B$ $(y_1 \neq y_2)$, $|\{h \in \mathcal{H} : h(x_1) = y_1, h(x_2) = y_2\}| \leq \epsilon \cdot |\mathcal{H}|/b$.*

The first condition states that the probability that a given input x is mapped to a given output y equals $1/b$. The second condition implies that if x_1 is mapped to y_1, then the conditional probability that x_2 (different from x_1) is mapped to y_2 is upper bounded by ϵ. The lowest possible value for ϵ equals $1/b$ and this family has been called strongly universal functions in [86]. Stinson shows that for this family the first condition in the definition follows from the second one [83].

An ϵ-almost strongly universal hash function family can be used in a similar way as a MAC algorithm. The secret key K chooses a function in the family; unlike the key for a MAC algorithm, the key can serve to authenticate a single message only. An ϵ-almost universal hash function family can also provide message authentication, but in addition its result needs to be encrypted (for example using a one-time pad).

3 Generic Constructions and Attacks

First a general model is presented for iterated hash functions. Next generic attacks are described, that is, attacks that are independent of the specific details of the hash function. This section is concluded with a discussion of generic security results for hash functions.

3.1 A General Model for Iterated Hash Functions

Most known hash functions are based on a compression function with fixed size inputs; they process every message block in a similar way. Lai and Massey call this an "iterated" hash function [55]. The information is divided into t blocks x_1 through x_t. If the total number of bits is not a multiple of the block length, the information is padded to the required length (using a so-called *padding rule*). The hash function can then be described as follows:

$$\begin{aligned} H_0 &= IV \\ H_i &= f(x_i, H_{i-1}) \qquad i = 1, 2, \ldots t \\ h(x) &= g(H_t)\,. \end{aligned}$$

The result of the hash function is denoted with $h(x)$ and IV is the abbreviation for Initial Value. The function f is called the *round* function or *compression* function, and the function g is called the *output transformation*. It is often omitted (that is, g is often the identity function). Two elements in this definition have an important influence on the security of a hash function: the choice of the padding rule and the choice of the IV. It is recommended that the padding rule is unambiguous (i.e., there do not exist two messages that can be padded to the same padded message); at the end one should append the length of the message; and the IV should be defined as part of the description of the hash function (this is called MD-strengthening after Merkle and Damgård). In some cases one can deviate from this rule, but this will make the hash function less secure and may lead to trivial collisions or second preimages.

An alternative model is a tree structure, that allows for increased parallelism and may result in different security conditions for the round function (see also Sect. 3.3). For the time being it is rarely used in practice.

The general model for MAC algorithms is similar to that of MDCs; the use of an output transformation is more common here. Bellare and Rogaway [10] show that the above model does not work for a UOWHF, and they introduce a different approach. Earlier Naor and Yung developed a different iterated construction for a UOWHF in [64].

3.2 Generic Attacks

This section gives an overview of the known general attacks methods on MDCs. A first class of attacks depend only on the size of the parameters, and not on the specific hash function. The second class depends on the properties of the round function f.

This taxonomy can be helpful to understand the security results presented in Sect. 3.3, but can also serve as a *caveat* for designers and users of hash functions.

Attacks Independent of the Algorithm These attacks depend only on the size n in bits of the hash result, and are independent of the specific details of

the algorithm. It is assumed that the MDC approximates a random function: if this is not the case this class of attacks will be even more successful. For the time being 2^{64} operations is considered to be on the edge of feasibility. In view of the fact that the speed of computers is multiplied by four every three years (this is one of the formulations of Moore's law), 2^{70} operations is sufficient for the next 5 to 10 years, but it will be only marginally secure within 15 years. For applications that require protection for 20 years, one should try to design the hash function such that an attack requires at least 2^{80} operations.

Random (2nd) Preimage Attack. The opponent selects a random message and hopes that a given hash result will be hit. If the hash function has the required "random" behavior, his probability of success equals $1/2^n$ with n the number of bits of the hash result. This attack can be carried out off-line and in parallel, which means that n should be at least 64.

If a significant number of messages can be attacked simultaneously, it is advisable to select a larger value of n. In that case it is preferable to use a UOWHF rather than a simple OWHF: as every instance will have a different parameter, this prevents an attacker from amortizing his effort over many targets.

Birthday Attack. The birthday paradox states that for a group of 23 people, the probability that at least two people have a common birthday exceeds 1/2. Intuitively one expects that the probability is much lower. However, the number of pairs of people in such a group equals $23 \cdot 22/2 = 253$. This can be exploited to find collisions for a hash function as follows: an adversary generates r_1 variations on a bogus message and r_2 variations on a genuine message. The expected number of collisions equals $r_1 \cdot r_2/n$. The probability of finding a bogus message and a genuine message that hash to the same result is given by $1 - \exp(-r_1 \cdot r_2/2^n)$, which is about 63% when $r = r_1 = r_2 = 2^{\frac{n}{2}}$. Finding the collision does not require r^2 operations: after sorting the data, which requires $O(r \log r)$ operations, comparison is easy. This attack was first pointed out by Yuval [87].

One can reduce the memory requirements for collision search by translating the problem to the detection of a cycle in an iterated mapping. Indeed, any mapping that is iterated on a finite set will eventually repeat, i.e., it will enter a cycle. If the mapping is a random mapping (rather than a permutation), the entry point to the cycle corresponds to a collision for the function (this algorithm fails if the starting point belongs to the cycle, but this event has a negligible probability). The detection of a cycle does not require storing all the values; for example, the technique of distinguished points can be used (one only stores special points, for example those points beginning with 30 zero bits). Cycle detection techniques were first applied to collision search by Quisquater [71]. The expected number of function evaluations of his algorithm is $2\sqrt{\pi/2} \cdot 2^{\frac{n}{2}}$; the storage requirements are negligible. In [85], van Oorschot and Wiener propose an efficient parallel variant of this algorithm: the speed-up is linear with the number of processors. They estimate that with a 10 million US$ machine, collisions for MD5 (with $n = 128$) can be found in 21 days (in 1994). In order to make a collision search infeasible, n should be at least 160 bits.

For digital signatures, hash functions need to be collision resistant since otherwise one can sign one message and later claim to have signed a different message, or be held accountable for a different message. There is no way to prevent a sender from performing this attack, although the occurrence of two messages that hash to the same value might make him suspect. Outsiders can perform the same attack if they can convince the signer to sign a message of their choice. The sender can protect himself through randomizing the message just prior to signing (or by randomizing the hash function as is done for a UOWHF, cf. Sect. 2.3).

Attacks Dependent on the Chaining This class of attacks depends on some high level properties of the compression function f.

Meet-in-the-Middle Attack. This attack is a variation on the birthday attack, but instead of comparing the hash results, one compares intermediate chaining variables. The attack enables an opponent to construct a (2nd) preimage, which is not possible for a simple birthday attack. The opponent generates r_1 variations on the first part of a bogus message and r_2 variations on the last part. Starting from the initial value and going backwards from the hash result, the probability for a matching intermediate variable is again given by $1 - \exp(-r_1 \cdot r_2/2^n)$. The only restriction that applies to the meeting point is that it cannot be the first or last value of the chaining variable. The cycle finding algorithm has been extended by Quisquater to perform a meet-in-the-middle attack with negligible storage [72]. The attack can be precluded by avoiding functions f that are invertible to the chaining variable H_{i-1} and to the message x_i (see also Theorem 1 in Sect. 3.3).

Further extensions of this attack have been proposed by Coppersmith [19] and Girault *et al.* [46] to break p-fold iterated schemes, i.e., weak schemes with more than one 'pass' over the message as proposed by Davies [28]. Other extensions take into account additional constraints on the message.

Fixed Point Attack. The idea of this attack is to look for an H_{i-1} and x_i such that $f(x_i, H_{i-1}) = H_{i-1}$. If the chaining variable is equal to H_{i-1}, it is possible to insert an arbitrary number of blocks equal to x_i without modifying the hash result. Producing collisions or a second preimage with this attack is only possible if the chaining variable can be made equal to H_{i-1}: this is the case if IV can be chosen equal to a specific value, or if a large number of fixed points can be constructed (e.g., if one can find an x_i for a significant fraction of H_{i-1}'s). Of course this attack can be extended to fixed points that occur after more than one iteration. This attack can be made more difficult by appending a block count and by fixing IV (MD-strengthening, see Sect. 3.1).

3.3 General Security Results

Research on hash functions has focussed on the question: which properties should be imposed on f to guarantee that h satisfies certain properties? Two partial

answers have been found to this question. The first result by Lai and Massey [55] gives necessary and sufficient conditions for f in order to obtain an *"ideally secure"* hash function h, that is, a hash function for which finding a preimage takes time $\Theta(2^n)$.

Theorem 1 (Lai–Massey). *Assume that the padding contains the length of the input string, and that the message x (without padding) contains at least 2 blocks. Then finding a second preimage for h with a fixed IV requires 2^n operations if and only if finding a second preimage for f with arbitrarily chosen H_{i-1} requires 2^n operations.*

Necessity of the condition is based on the following argument: if one can find a second preimage for f in 2^s operations (with $s < n$), one can find a second preimage for h in $2^{(n+s)/2+1}$ operations with a meet-in-the-middle attack (cf. Sect. 3.2).

A second result by Damgård [26] and independently by Merkle [60] states that for h to be a CRHF it is sufficient that f is a collision resistant function.

Theorem 2 (Damgård–Merkle). *Let f be a collision resistant function mapping l to n bits (with $l - n > 1$). If an unambiguous padding rule is used, the following construction yields a CRHF:*

$$H_1 = f(0^{n+1} \parallel x_1)$$
$$H_i = f(H_{i-1} \parallel 1 \parallel x_i) \qquad \text{for } i = 2, 3, \ldots t\,.$$

Here $\parallel$ denotes the concatenation of binary strings. The construction can be improved slightly,[2] and extended to the case where $l = n + 1$, at the cost of an additional assumption on f (see [26] for details and Gibson's comment in [42]). It can also be extended to a tree construction, which allows for increased parallelism [26].

We conclude this section with two other general results on the theory of hash functions. Damgård has showed in [25] that a collision resistant hash function can be constructed if claw-free permutations exist; Russell has slightly weakened this requirement to the existence of claw-free pseudo-permutations [79] (pseudo-permutations are functions that cannot be distinguished from permutations). Recently Simon [82] has provided a motivation to treat collision resistant hash functions as independent cryptographic primitives. He showed that no provable construction of a CRHF exists based on a "black box" one-way permutation, i.e., a one-way permutation treated as an oracle.

4 An Overview of Constructions

This section briefly discusses three types of MDCs: MDCs based on a block cipher, MDCs based on algebraic structures (modular arithmetic, knapsack, and lattice problems), and custom designed MDCs. For a more detailed discussion, the reader is referred to [67,68].

[2] One can get rid of the extra "0" and "1" bits.

4.1 MDCs Based on a Block Cipher

Several arguments can be given for designers of hash functions to base their schemes on existing encryption algorithms. The first argument is purely historical: DES [37] was the first standard commercial cryptographic primitive that was widely available; it seemed natural to construct hash functions based on this block cipher. A second argument is the minimization of the design and implementation effort: hash functions and block ciphers that are both efficient and secure are hard to design, and many examples that support this view can be found in the literature. Moreover, existing software and hardware implementations can be reused, which will decrease the cost. The major advantage however is that the trust in existing encryption algorithms can be transferred to a hash function. The main disadvantage of this approach is that custom designed hash functions are likely to be more efficient. This is particularly true because hash functions based on block ciphers require a key change after every encryption. Finally note that block ciphers may exhibit some weaknesses that are only important if they are used in a hashing mode. One also has to take into account that in some countries export restrictions for block ciphers may be tighter than those for hash functions.

The encryption operation E will be written as $y = E_K(x)$. Here x denotes the plaintext, y the ciphertext, and K the key. The size of the plaintext and ciphertext or the block length (in bits) will be denoted with r, while the key size (in bits) will be denoted with k. In the case of the well known block cipher DES, $r = 64$ and $k = 56$ [37]. The **hash rate** of a hash function based on a block cipher is defined as the number of r-bit input blocks that can be processed with a single encryption.

A distinction will be made between the cases $n = r$, $n = 2r$, and $n > 2r$. This is motivated by the fact that most proposed block ciphers have a block length of only 64 bits, and hence an MDC with a result at least twice the block length is necessary to obtain a CRHF. Other proposals are based on a block cipher with a large key, or on a block cipher with a modified key schedule.

Size of Hash Result Equal to the Block Length. If follows from Sect. 3.2 that these hash functions can only be collision resistant if the block length r is at least 128 bits to 160 bits. Most present day block ciphers have only a block length of 64 bits, but the AES (Advanced Encryption Standard), which NIST intends to publish by 2001, will have a block length of 128 bits.

All schemes of this type proposed in the literature have rate 1. The first 'secure' construction for such a hash function was the '85 scheme by Matyas *et al.* [57]:

$$H_i = E^{\oplus}_{s(H_{i-1})}(x_i) \stackrel{\text{def}}{=} E_{s(H_{i-1})}(x_i) \oplus x_i \,.$$

Here $s()$ is a mapping from the ciphertext space to the key space. This scheme has been included in ISO/IEC 10118–2 [51], and it forms the main building block for other hash functions based on block ciphers. This general construction is also used in several custom designed hash functions (cf. Sect. 4.3).

It is widely believed that this mapping is hard to invert, but there is no proof of this. It is not even clear which assumptions have to imposed on the block cipher to allow for such a proof. One can apply the following intuitive reasoning: either one chooses the plaintext x, but then one has to find the key corresponding to one plaintext/ciphertext pair, which is deemed to be infeasible; alternatively, one chooses the key, but then one has to find for a given key a plaintext/ciphertext pair with a known difference, which is also believed to be difficult. Therefore it is conjectured that if the block cipher is 'ideal' (i.e., a keyed random one-way permutation) no shortcut attacks exist, which implies that a collision attack requires $\Theta(2^{r/2})$ operations and a (2nd) preimage attack $\Theta(2^r)$ operations.

Preneel *et al.* show that 12 variants exist with a similar security level; they can be obtained by applying an affine transformation to the inputs of two basic schemes [69]. Moreover, the security level of these hash functions is limited by $\min(k, r)$, even if the size of some internal variables is equal to $\max(k, r)$. One such variant is widely known as the Davies-Meyer scheme (the real inventors are probably Matyas and Meyer):

$$H_i = E^{\oplus}_{x_i}(H_{i-1})\,.$$

It has the advantage that it extends more easily to block ciphers where key size and block size are different.

Size of Hash Result Equal to Twice the Block Length. The goal of *double block length* hash functions is to achieve a higher security level against collision attacks. Ideally a collision attack on such a hash function should require 2^r operations, and a (2nd) preimage attack 2^{2r} operations.

A series of proposals attempted to double the size of the hash result, for example by iterating a OWHF; all of these succumbed to a 'divide and conquer' attack. A large class of proposals of rate 1 has the following form:

$$H^1_i = E_{A^1_i}(B^1_i) \oplus C^1_i$$
$$H^2_i = E_{A^2_i}(B^2_i) \oplus C^2_i\,,$$

where A^1_i, B^1_i, and C^1_i are binary linear combinations of H^1_{i-1}, H^2_{i-1}, x^1_i, and x^2_i and where A^2_i, B^2_i, and C^2_i are binary linear combinations of H^1_{i-1}, H^2_{i-1}, x^1_i, x^2_i, and H^1_i. The hash result is equal to the concatenation of H^1_t and H^2_t. Knudsen *et al.* showed that for all hash functions in this class, a preimage attack requires at most 2^r operations, and a collision attack requires at most $2^{3r/4}$ operations (for most schemes this can be reduced to $2^{r/2}$) [53].

The few proposals that survive till today have rate less than 1. Two important examples are MDC-2 and MDC-4 with hash rate 1/2 and 1/4 respectively. They have been designed by Bracht *et al.* [13], and are also known as the Meyer-Schilling hash functions after the authors of the first paper describing these schemes [63]. MDC-2 has been included in ISO/IEC 10118-2 [51] (in a more

general form); it can be described as follows:

$$T_i^1 = E^{\oplus}_{u(H_{i-1}^1)}(x_i) = LT_i^1 \parallel RT_i^1 \qquad H_i^1 = LT_i^1 \parallel RT_i^2$$
$$T_i^2 = E^{\oplus}_{v(H_{i-1}^2)}(x_i) = LT_i^2 \parallel RT_i^2 \qquad H_i^2 = LT_i^2 \parallel RT_i^1 \,.$$

Here the variables H_0^1 and H_0^2 are initialized with the values IV_1 and IV_2 respectively, and the hash result is equal to the concatenation of H_t^1 and H_t^2. The ISO/IEC standard does not specify the block cipher; it also requires the specification of two mappings u, v from the ciphertext space to the key space such that $u(IV^1) \neq v(IV^2)$. The best known preimage and collision attacks on MDC-2 require 2^{3r} and 2^r operations respectively (Lai and Massey [55]). However, it is obvious that the compression function of MDC-2 is rather weak: preimage and collision attacks on the compression function require at most 2^r and $2^{r/2}$ operations (one fixes x_i and varies H_{i-1}^1 and H_{i-1}^2 independently).

One iteration of MDC-4 consists of the concatenation of two MDC-2 steps, where the plaintexts in the second step are equal to $H2_{i-1}$ and $H1_{i-1}$. The rate of MDC-4 is equal to 1/4. The best known attack to find a preimage for MDC-4 requires 2^{2r} operations. This shows that MDC-4 is probably more secure than MDC-2 against preimage attacks. However, a collision for the compression function of MDC-2 with a specified value for H_{i-1}^1 and H_{i-1}^2 also yields a collision for the compression function of MDC-4. Moreover, it has been demonstrated in by Preneel and Knudsen [67,54] that collisions for the compression function of MDC-4 can be found with $2^{3r/4}$ encryptions and the storage of $2^{3r/4}$ r-bit quantities.

Merkle describes an interesting proposal in [60], for which he proves that the compression function is collision resistant based on the assumption that the underlying single block length scheme is secure. The simplest scheme (with rate 1/18.3 for DES) can be described as follows:

$$H_i = \mathrm{chop}_{16}\left[E^{\oplus}_{0\|H_{i-1}^1}(H_{i-1}^2 \| x_i) \;\parallel\; E^{\oplus}_{1\|H_{i-1}^1}(H_{i-1}^2 \| x_i)\right] \,.$$

Here H_{i-1} is a string consisting of 112 bits, the leftmost 55 bits of which are denoted H_{i-1}^1, and the remaining 57 are denoted H_{i-1}^2; x_i consists of 7 bits only. The function chop_r drops the r rightmost bits of its argument. Note that this construction is similar to MDC-2 (but much slower). The most efficient proposal is more complex and use six invocations of the block cipher in two layers. Its hash rate is equal to 0.27 for DES. Merkle's proof for this proposal only showed a security level of $2^{52.5}$ against collisions; Preneel has improved this to 2^{56} [67].

Even the schemes in this class that provide optimal security do not offer long term collision resistance when used with DES; this will change with AES, which will have a block and key length of 128 bits (key lengths of 192 and 256 bits will also be provided).

Size of Hash Result Larger than Twice the Block Length. Knudsen and Preneel also design a collision resistant compression function, but with parallel encryptions only [54]. They show how a class of efficient constructions for

hash functions can be obtained by using non-binary error-correcting codes. Their schemes can achieve a provable security level against collisions equal to 2^r, $2^{3r/2}$ (or more) and this with rates larger than 1/2; the security proof reduces the security of this scheme to an assumption on the single block length hash functions. The internal memory of the scheme is however much larger than 2 or 3 blocks, which implies that an output transformation is required.

Other Constructions. Extending earlier work by Merkle [59], Lai and Massey [55] propose constructions for block ciphers with a key twice as long as the block length (Tandem Davies-Meyer and Abreast Davies-Meyer). Both schemes have rate equal to 1/2; the best known attacks for a preimage and a collision requires 2^{2r} respectively 2^r encryptions. Faster schemes in this class have been developed in [54].

Aiello and Venkatesan propose in [1] a construction to double the output of a random function. In order for it to be usable for hashing, one needs to define the key schedule of this larger 'block cipher'. The construction by Aiello, Haber, and Venkatesan [2] replaces the key schedule of DES by a function from the MDx-family (cf. Sect. 4.3); several instances are combined by choosing different (fixed) plaintexts.

4.2 MDCs Based on Algebraic Structures

First hash functions based on modular arithmetic are considered. Next hash functions based on knapsack problems and lattices are presented. This section is concluded with a short discussion of incremental hash functions.

MDCs Based on Modular Arithmetic. These hash functions are designed to use the modular arithmetic hardware that is required to produce digital signatures (for example, RSA [77]). The security of certain constructions can be based on the hardness of some number theoretic problems. Moreover these schemes are easily scalable. The disadvantage is that the underlying primitive has a rich mathematical structure; this can be exploited in attacks that use the homomorphic structure and the fixed points of modular exponentiation (trivial examples are 0 and 1); one also has to ensure that no small inputs occur.

A distinction is made between 'ad hoc' schemes, which have no provable reduction to the underlying hard problem, and provably secure schemes. Schemes in the first class are typically much more efficient, but many proposals have been broken; however, it seems that recently designers have been more conservative and designs survive longer.

Schemes Without Security Reduction. Most of these schemes uses a modulus N, that is, the product of two large primes. The size of N in bits (denoted with n) is typically between 512 and 1024. These hash functions can be useful in combination with RSA [77] as digital signature scheme. However, this choice poses the following practical problem: the person who has generated the modulus

knows its factorization, and hence he has a potential advantage over the other users of the hash function. One can try to design the hash function such that knowledge of the factorization does not help in attacking it (this is probably difficult to achieve). Alternatives are to use a trusted third party to generate the modulus (for example, the modulus of the Certification Authority), or to generate the modulus using a multi-party secure computation; recently practical solutions for such a computation have been developed by Boneh and Franklin [12] and Frankel *et al.* [40].

The most efficient schemes are based on modular squaring. An additional argument for squaring is that any algorithm that can extract modular square roots is reducible to a factoring algorithm (in random polynomial time). The best scheme seems to be of the following form: $H_i = (x_i \oplus H_{i-1})^2 \bmod N \ \oplus H_{i-1}$ [69].

It is essential to add redundancy to the message input. The first designs for this redundancy were not very successful (see for example Girault [45], Girault and Misarsky [47], and Coppersmith [20]). ISO/IEC SC27 has developed a new proposal, that is currently at the Final Draft International Standard (FDIS) level; it is called MASH-1 (for Modular Arithmetic Secure Hash) [51]:

$$H_i = ((x_i \oplus H_{i-1}) \vee A)^2 \pmod N \ \oplus H_{i-1}$$

here $A =$ `0xF00...00`, the four most significant bits in every byte of x_i are set to `1111`, and the output of the squaring operation is chopped to n bits. A complex output transformation is added, which consists of a number of applications of the compression function; its goal is to destroy all the remaining mathematical structure. The final result is at most $n/2$ bits. The best known preimage and collision attacks on MASH-1 require $2^{n/2}$ and $2^{n/4}$ operations [21]; they are thus not better than brute force attacks. MASH-2 is a variant of MASH-1 which uses exponent $2^8 + 1$ [51]. This provides for an additional security margin.

Schemes With a Security Reduction. For several schemes there exists a security reduction to a number theoretic problem that is believe to be difficult. However, they are very slow: typically they hash $\log_2 \log_2 N$ bits per modular squaring (or even per modular exponentiation).

Damgård provides two hash functions for which finding a collision is provably equivalent to factoring an RSA modulus [24]. Gibson proposes a construction based on the discrete logarithm problem modulo a composite [43]. A third approach uses the discrete logarithm problem in a group of prime order p denoted with G_p (Bellare *et al.* [6], after earlier work by Chaum *et al.* [18] and Brands). Every non-trivial element of G_p is a generator. The hash function uses t random elements α_i from G_p ($\alpha_i \neq 1$). The hash result is then computed as

$$H_{t+1} = \prod_{i=1}^{t} \alpha_i^{\tilde{x_i}} .$$

Here $\tilde{x_i}$ is obtained by considering the string x_i as the binary expansion of a number and prepending 1 to it. This avoids trivial collisions when x_i consists of all zeroes.

MDCs Based on Knapsack and Lattice Problems. The knapsack problem (which is a special case of the subset sum problem) of dimensions n and $\ell(n)$ can be defined as follows: given a set of n l-bit integers $\{a_1, a_2, \ldots, a_n\}$, and an integer S, find a vector x with components x_i equal to 0 or 1 such that

$$\sum_{i=1}^{n} a_i \cdot x_i = S \bmod 2^{\ell(n)} .$$

For application to hashing, one needs $n > \ell(n)$. The knapsack problem is known to be NP-hard; while this means that probably no feasible worst-case algorithms for this problem exists, this does not tell much about the hardness of a random instance. This problem was used in 1978 by Merkle and Hellman to construct the first public-key encryption system [62]. However, almost all public-key schemes based on the knapsack problem have been broken (see for example [65]), which has given the knapsack a bad reputation. The appeal of the knapsack problem (and related lattice based problems) lies in the fact that both hardware and software implementations are very fast compared to schemes based on number theoretic problems. Moreover, evaluation of a knapsack allows for significant parallelism. Finally, interesting security reductions can be proved: examples are the work for Impagliazzo and Naor [50] on knapsacks and that of Ajtai [3] for lattices; Ajtai was able to prove that if the shortest vector in a lattice problem is hard in the worst case, then the knapsack problem is hard on the average. However, some researchers believe that for realistic parameters, both these problems are relatively easy. If they are right, knapsack and lattice problems are not useful to practical cryptography.

Attacks on knapsacks often use the LLL lattice reduction algorithm [56] that finds the shortest vector in a lattice (the algorithm performs in practice much better than can be guaranteed). This reduction to the shortest vector problem only works for $\ell(n) > 1.0629 \cdot n$. Knapsack problems become more difficult when $n \approx \ell(n)$; however, the performance of the hash function decreases with the value $n - \ell(n)$. For $n = \ell(n)$, the best known attack requires time $O(2^{n/2})$ and space $O(2^{n/4})$. Impagliazzo and Naor summarize the state of the art in [50]. A different class of attacks are the algebraic attacks proposed by Camion and Patarin [14] and optimized by Patarin in [66]; these attacks tend to work better when $n \gg \ell(n)$. The scheme of Damgård [26] has been broken both using LLL [52] and using algebraic techniques [66]. It is for the time being an open problem whether a random knapsack with $n = 1024$, $l = 512$, and $\ell = 512$ is hard to solve.

Impagliazzo and Naor describe an efficient construction for a UOWHF (cf. Sect. 2.3) and provide a reduction of its security to that of the knapsack problem [50]. Ajtai introduced a function that is one-way (or preimage resistant) if the

problem of approximating the shortest vector in a lattice to polynomial factors is hard [3]. Goldreich *et al.* have proved that the function is in fact collision resistant [48].

Several multiplicative knapsacks have also been proposed; multiplicative notation is used for non-Abelian groups. The earliest proposal dates back to '77 (but it was quickly shown to be insecure). A recent example are the schemes by Zémor [88] and Tillich and Zémor [84]. Their security is based on the hardness of finding short factorizations in certain groups. In some cases one can even prove a lower bound on the Hamming distance between colliding messages. Attacks on these proposals (for certain parameters) can be found in [17,41]. Impagliazzo and Naor also extend their construction on a UOWHF to multiplication in a group [50].

Knapsack and lattice based hash functions have also the potential problem that trapdoors may be inserted when the vectors are generated. Therefore it is recommended that the instance generation is reproducible (for example, through the use of a pseudo-random string generator or a hash function).

Incremental Hash Functions. A hash function (or any cryptographic primitive) is called *incremental* if it has the following property: if the hash function has been evaluated for an input x, and a small modification is made to x, resulting in x', then one can update $h(x)$ in time proportional to the amount of modification between x and x', rather than having to recompute $h(x')$ from scratch. If a function is incremental, it is automatically parallelizable as well.

This concept was first introduced by Bellare *et al.* [6]. They also made a first proposal based on exponentiation in a group of prime order. Improved constructions were proposed by Bellare and Micciancio [8] that consist of two steps:

- First the message is divided into blocks; each block (together with its index) is hashed using a conventional collision resistant hash function (restricted to fixed length inputs). This is called the 'randomization' step as in the analysis the hash function is treated as an 'ideal' hash function or random oracle (which is a very demanding requirement).
- Next the different outputs are combined using a group operation. This can be a group of large prime order in which the discrete logarithm problem is hard, and modular addition. The first approach leads to a reduction to the discrete logarithm problem, while the second leads to a reduction to the 'weighted knapsack' problem.

The same techniques can also be used to improve the lattice based hash function. These schemes have the advantage that they are much more efficient than the other schemes studied in this section. However, this comes at a cost of requiring a collision resistant hash function, which also has to behave 'perfectly random.' This construction is remarkable, as it construct a collision resistant function based on a one-way property (but with specific algebraic structure, so there is no contradiction to the result of Simon [82] discussed in Sect. 3.3).

4.3 Custom Designed MDCs

This section discusses a selection of custom designed hash functions, i.e., algorithms that were especially designed for hashing operations. Most of these algorithms use the Davies-Meyer approach (cf. Sect. 4.1): the compression function is a block cipher, 'keyed' by the text input x_i; the plaintext is the value H_{i-1}, which is also added to the ciphertext (feedforward).

In 1990, R. Rivest proposed MD4 [75], a hash function with a 128-bit result based on 32-bit integer arithmetic. While this hash function proved to be not sufficiently strong, the innovative design ideas have influenced many other designs. The algorithms derived from it (with improved strength) are often called the *MDx-family*. This family contains the most popular hash functions in use today. Dobbertin has found collisions for MD4; his attack combines algebraic techniques and optimization techniques such as genetic algorithms [32,33]. It can be extended in such a way that even 'meaningful' collisions are obtained: the complete message (except for a few dozen bytes) is under complete control of the attacker. His attack also applies to the compression function of 'extended MD4' [75], which consists of concatenating two loosely coupled instances of MD4. Later Dobbertin showed that a reduced version of MD4 (2 rounds out of 3) is not preimage resistant [35].

Following early attacks on MD4 by Merkle and den Boer and Bosselaers [29], Rivest quickly designed a strengthened version, namely MD5 [76]. It was however shown by den Boer and Bosselaers [30] that the compression function of MD5 is not collision resistant (but their collisions are of the form $f(H_{i-1}, x_i) = f(H'_{i-1}, x_i)$, which is not immediately usable in practice). Dobbertin has extended his attack on MD4 to yield collisions for the compression function of MD5, i.e., $f(H_{i-1}, x_i) = f(H_{i-1}, x'_i)$, where he has some control over H_{i-1} [34]. It is believed that it is feasible to extend this attack to collisions for MD5 itself (that is, to take into account the IV).

A second improved variant of MD4, the Secure Hash Algorithm, was proposed by NIST [38] in 1993. The size of the hash result is increased from 128 to 160 bits and the message words are not simply permuted but encoded with a shortened cyclic code. After a few years, NSA discovered a certificational weakness in SHA; apparently collisions can be found in less than 2^{80} operations. Consequently a new release of the standard was published. The new algorithm is called SHA-1 [39]. Recently Chabaud and Joux have published an attack that finds collisions for SHA in 2^{61} operations [16]; it is probably similar to the (classified) attack developed earlier that prompted the upgrade to SHA-1.

Yet another improved version of MD4, called RIPEMD, was developed in the framework of the EEC-RACE project RIPE [74]. Due to partial attacks by Dobbertin [32], it was later upgraded by Dobbertin *et al.* to RIPEMD-128 and RIPEMD-160, that have a 128-bit and a 160-bit result respectively [36]. Variants with a 256 and 320-bit result have been introduced as well. Together with SHA-1, RIPEMD-128 and RIPEMD-160 are the three custom designed hash functions included in ISO/IEC 10118–3 [51].

Merkle suggested in 1989 a software oriented one-way hash function called Snefru [61]. It is based on large random substitution tables (2 Kbyte per pass) and allows for a variable size of the result (128 or 256 bits). Biham and Shamir have shown that Snefru-128 [11] is vulnerable to differential attacks. As a consequence it is recommended to use 6 and preferably 8 passes, preferably with a 256-bit hash result. However, these measures increase the size of the substitution tables and decrease the performance.

Two of the most recent designs are Tiger and Panama. Tiger was proposed by Anderson and Biham [4]. It is tuned towards 64-bit processors and mixes Snefru-type S-boxes (8 input bits and 64 output bits) with arithmetic operations. Panama is a design of Daemen and Clapp [23]; it has been designed to take advantage of the instruction-level parallelism in modern processors.

5 Conclusions and Open Problems

In spite of the popularity of cryptographic hash functions, very few theoretical results are known in this area; it is clear that further research is necessary to improve our understanding of these primitives. Collision resistance seems to be a condition that is particularly hard to analyze. Some open problems are whether it is possible to construct collision resistant hash functions based on weaker assumptions, and whether any theory can be developed to support the current constructions.

In the area of practical constructions, there is a need for more efficient hash functions, the security of which is better understood. For hash functions based on block ciphers, the main problem seems to be the assumptions on the block cipher. From an application viewpoint, multiplicative knapsacks seem to be very attractive (due to inherent parallelism and due to the fact that certain properties can be proved). However, further research is necessary to assess their security. Another research problem is to understand to which extent the current constructions provide other security properties such as pseudo-randomness and partial preimage resistance; both properties are related to the difficulty of inverting the hash function if part of the input is known.

References

1. W. Aiello, R. Venkatesan, "Foiling birthday attacks in length-doubling transformations. Benes: a non-reversible alternative to Feistel," *Advances in Cryptology, Proceedings Eurocrypt'96, LNCS 1070*, U. Maurer, Ed., Springer-Verlag, 1996, pp. 307–320. 172
2. W. Aiello, S. Haber, R. Venkatesan, "New constructions for secure hash functions," *Fast Software Encryption, LNCS 1372*, S. Vaudenay, Ed., Springer-Verlag, 1998, pp. 150–167. 172
3. M. Ajtai, "Generating hard instances of lattice problems," *Proc. 28th ACM Symposium on the Theory of Computing*, 1996, pp. 99–108. 174, 175
4. R. Anderson, E. Biham, "Tiger: A new fast hash function," *Fast Software Encryption, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 89–97. 177

5. M. Bellare, R. Canetti, H. Krawczyk, "Pseudorandom functions revisited: The cascade construction and its concrete security," *Proc. 37th Annual Symposium on the Foundations of Computer Science, IEEE*, 1996, pp. 514–523. Full version via `http://www-cse.ucsd.edu/users/mihir`. 159
6. M. Bellare, O. Goldreich, S. Goldwasser, "Incremental cryptography: the case of hashing and signing," *Advances in Cryptology, Proceedings Crypto'94, LNCS 839*, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 216–233. 173, 175
7. M. Bellare, J. Kilian, P. Rogaway, "The security of cipher block chaining," *Advances in Cryptology, Proceedings Crypto'94, LNCS 839*, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 341–358. 159, 163
8. M. Bellare, D. Micciancio, "A new paradigm for collision-free hashing: incrementality at reduced cost," *Advances in Cryptology, Proceedings Eurocrypt'97, LNCS 1233*, W. Fumy, Ed., Springer-Verlag, 1997, pp. 163–192. 175
9. M. Bellare, P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," *Proc. 1st ACM Conference on Computer and Communications Security*, ACM, 1993, pp. 62–73. 159
10. M. Bellare, P. Rogaway, "Collision-resistant hashing: towards making UOWHFs practical," *Advances in Cryptology, Proceedings Crypto'97, LNCS 1294*, B. Kaliski, Ed., Springer-Verlag, 1997, pp. 470–484. 160, 162, 165
11. E. Biham, A. Shamir, *"Differential Cryptanalysis of the Data Encryption Standard,"* Springer-Verlag, 1993. 177
12. D. Boneh, M. Franklin, "Efficient generation of shared RSA keys," *Advances in Cryptology, Proceedings Crypto'97, LNCS 1294*, B. Kaliski, Ed., Springer-Verlag, 1997, pp. 425–439. 173
13. B.O. Brachtl, D. Coppersmith, M.M. Hyden, S.M. Matyas, C.H. Meyer, J. Oseas, S. Pilpel, M. Schilling, *"Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function,"* U.S. Patent Number 4,908,861, March 13, 1990. 170
14. P. Camion, J. Patarin, "The knapsack hash function proposed at Crypto'89 can be broken," *Advances in Cryptology, Proceedings Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 39–53. 174
15. J.L. Carter, M.N. Wegman, "Universal classes of hash functions," *Journal of Computer and System Sciences*, Vol. 18, 1979, pp. 143–154. 163, 164
16. F. Chabaud, A. Joux, "Differential collisions: an explanation for SHA-1," *Advances in Cryptology, Proceedings Crypto'98, LNCS 1462*, H. Krawczyk, Ed., Springer-Verlag, 1998, pp. 56–71. 176
17. C. Charnes, J. Pieprzyk, "Attacking the SL_2 hashing scheme," *Advances in Cryptology, Proceedings Asiacrypt'94, LNCS 917*, J. Pieprzyk and R. Safavi-Naini, Eds., Springer-Verlag, 1995, pp. 322–330. 175
18. D. Chaum, E. van Heijst, B. Pfitzmann, "Cryptographically strong undeniable signatures, unconditionally secure for the signer," *Advances in Cryptology, Proceedings Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1992, pp. 470–484. 173
19. D. Coppersmith, "Another birthday attack," *Advances in Cryptology, Proceedings Crypto'85, LNCS 218*, H.C. Williams, Ed., Springer-Verlag, 1985, pp. 14–17. 167
20. D. Coppersmith, "Analysis of ISO/CCITT Document X.509 Annex D," *IBM T.J. Watson Center, Yorktown Heights, N.Y., 10598, Internal Memo*, June 11, 1989, (also ISO/IEC JTC1/SC20/WG2/N160). 173
21. D. Coppersmith, B. Preneel, "Comments on MASH-1 and MASH-2," February 21, 1995, ISO/IEC JTC1/SC27/N1055. 173

22. T. Cormen, C. Leierson, R. Rivest, *"Introduction to Algorithms,"* McGraw–Hill, 1992. 160
23. J. Daemen, C. Clapp, "Fast hashing and stream encryption with PANAMA," *Fast Software Encryption, LNCS 1372*, S. Vaudenay, Ed., Springer-Verlag, 1998, pp. 60–74. 177
24. I.B. Damgård, "Collision free hash functions and public key signature schemes," *Advances in Cryptology, Proceedings Eurocrypt'87, LNCS 304*, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp. 203–216. 161, 173
25. I.B. Damgård, "The application of claw free functions in cryptography," *PhD Thesis*, Aarhus University, Mathematical Institute, 1988. 161, 168
26. I.B. Damgård, "A design principle for hash functions," *Advances in Cryptology, Proceedings Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 416–427. 161, 168, 174
27. I.B. Damgård, T.P. Pedersen, B. Pfitzmann, "On the existence of statistically hiding bit commitment schemes and fail-stop signatures," *Advances in Cryptology, Proceedings Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 250–265. 158
28. D. Davies, W. L. Price, "The application of digital signatures based on public key cryptosystems," *NPL Report* DNACS 39/80, December 1980. 167
29. B. den Boer, A. Bosselaers, "An attack on the last two rounds of MD4," *Advances in Cryptology, Proceedings Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1992, pp. 194–203. 176
30. B. den Boer, A. Bosselaers, "Collisions for the compression function of MD5," *Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 293–304. 176
31. W. Diffie, M.E. Hellman, "New directions in cryptography," *IEEE Trans. on Information Theory*, Vol. IT–22, No. 6, 1976, pp. 644–654. 158, 159
32. H. Dobbertin, "RIPEMD with two-round compress function is not collisionfree," *Journal of Cryptology*, Vol. 10, No. 1, 1997, pp. 51–69. 176
33. H. Dobbertin, "Cryptanalysis of MD4," *Journal of Cryptology*, Vol. 11, No. 4, 1998, pp. 253–271. See also *Fast Software Encryption, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 53–69. 176
34. H. Dobbertin, "The status of MD5 after a recent attack," *CryptoBytes*, Vol. 2, No. 2, Summer 1996, pp. 1–6. 176
35. H. Dobbertin, "The first two rounds of MD4 are not one-way," *Fast Software Encryption, LNCS 1372*, S. Vaudenay, Ed., Springer-Verlag, 1998, pp. 284–292. 176
36. H. Dobbertin, A. Bosselaers, B. Preneel, "RIPEMD-160: a strengthened version of RIPEMD," *Fast Software Encryption, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 71–82.
See also `http://www.esat.kuleuven.ac.be/~bosselae/ripemd160`. 176
37. FIPS 46, *"Data Encryption Standard,"* Federal Information Processing Standard, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977 (revised as FIPS 46-1:1988; FIPS 46-2:1993). 169
38. FIPS 180, *"Secure Hash Standard,"* Federal Information Processing Standard (FIPS), Publication 180, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., May 11, 1993. 176
39. FIPS 180-1, *"Secure Hash Standard,"* Federal Information Processing Standard (FIPS), Publication 180-1, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., April 17, 1995. 176

40. Y. Frankel, P. D. MacKenzie, M. Yung, "Robust efficient distributed RSA-key generation," *Proc. 30th ACM Symposium on the Theory of Computing*, 1998. 173
41. W. Geiselmann, "A note on the hash function of Tillich and Zémor," *Cryptography and Coding. 5th IMA Conference*, C. Boyd, Ed., Springer-Verlag, 1995, pp. 257–263. 175
42. J.K. Gibson, "Some comments on Damgård's hashing principle," *Electronics Letters*, Vol. 26, No. 15, 1990, pp. 1178–1179. 161, 168
43. J.K. Gibson, "Discrete logarithm hash function that is collision free and one way," *IEE Proceedings-E*, Vol. 138, No. 6, November 1991, pp. 407–410. 173
44. E. Gilbert, F. MacWilliams, N. Sloane, "Codes which detect deception," *Bell System Technical Journal*, Vol. 53, No. 3, 1974, pp. 405–424. 163
45. M. Girault, "Hash-functions using modulo-n operations," *Advances in Cryptology, Proceedings Eurocrypt'87, LNCS 304*, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp. 217–226. 173
46. M. Girault, R. Cohen, M. Campana, "A generalized birthday attack," *Advances in Cryptology, Proceedings Eurocrypt'88, LNCS 330*, C.G. Günther, Ed., Springer-Verlag, 1988, pp. 129–156. 167
47. M. Girault, J.-F. Misarsky, "Selective forgery of RSA signatures using redundancy," *Advances in Cryptology, Proceedings Eurocrypt'97, LNCS 1233*, W. Fumy, Ed., Springer-Verlag, 1997, pp. 495–507. 173
48. O. Goldreich, S. Goldwasser, S. Halevi, "Collision-free hashing from lattice problems," *Theory of Cryptography Library*, `http://philby.ucsd.edu/cryptolib.html`, 96–09, July 1996. 175
49. M. Hellman, "A cryptanalytic time-memory tradeoff," *IEEE Trans. on Information Theory*, Vol. IT–26, 1980, pp. 401–406. 161
50. R. Impagliazzo, M. Naor, "Efficient cryptographic schemes provably as secure as subset sum," *Journal of Cryptology*, Vol. 9, No. 4, 1996, pp. 199–216. 174, 175
51. ISO/IEC 10118, *"Information technology – Security techniques – Hash-functions, Part 1: General"*, 1994, *"Part 2: Hash-functions using an n-bit block cipher algorithm,"* 1994, *"Part 3: Dedicated hash-functions,"* 1998, *"Part 4: Hash-functions using modular arithmetic,"* (FDIS) 1998. 169, 170, 173, 176
52. A. Joux, L. Granboulan, "A practical attack against knapsack based hash functions," *Advances in Cryptology, Proceedings Eurocrypt'94, LNCS 950*, A. De Santis, Ed., Springer-Verlag, 1995, pp. 58–66. 174
53. L.R. Knudsen, X. Lai, B. Preneel, "Attacks on fast double block length hash functions," *Journal of Cryptology*, Vol. 11, No. 1, Winter 1998, pp. 59–72. 170
54. L.R. Knudsen, B. Preneel, "Fast and secure hashing based on codes," *Advances in Cryptology, Proceedings Crypto'97, LNCS 1294*, B. Kaliski, Ed., Springer-Verlag, 1997, pp. 485–498. 171, 172
55. X. Lai, J.L. Massey, "Hash functions based on block ciphers," *Advances in Cryptology, Proceedings Eurocrypt'92, LNCS 658*, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 55–70. 165, 168, 171, 172
56. A. Lenstra, H. Lenstra, L. Lovász, "Factoring polynomials with rational coefficients," *Mathematischen Annalen*, Vol. 261, pp. 515–534, 1982. 174
57. S.M. Matyas, C.H. Meyer, J. Oseas, "Generating strong one-way functions with cryptographic algorithm," *IBM Techn. Disclosure Bull.*, Vol. 27, No. 10A, 1985, pp. 5658–5659. 169
58. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography,"* CRC Press, 1997. 160
59. R. Merkle, *"Secrecy, Authentication, and Public Key Systems,"* UMI Research Press, 1979. 159, 172

60. R. Merkle, "One way hash functions and DES," *Advances in Cryptology, Proceedings Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428–446. 159, 161, 168, 171
61. R. Merkle, "A fast software one-way hash function," *Journal of Cryptology*, Vol. 3, No. 1, 1990, pp. 43–58. 177
62. R. Merkle, M. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE Trans. on Information Theory*, Vol. IT–24, No. 5, 1978, pp. 525–530. 174
63. C.H. Meyer, M. Schilling, "Secure program load with Manipulation Detection Code," *Proc. Securicom 1988*, pp. 111–130. 170
64. M. Naor, M. Yung, "Universal one-way hash functions and their cryptographic applications," *Proc. 21st ACM Symposium on the Theory of Computing*, 1990, pp. 387–394. 162, 165
65. A.M. Odlyzko, "The rise and fall of knapsack cryptosystems," *Cryptology and Computational Number Theory*, C. Pomerance, Ed., Proc. Sympos. Appl. Math., Vol. 42, American Mathematical Society, 1990, pp. 75–88. 174
66. J. Patarin, "Collisions and inversions for Damgård's whole hash function," *Advances in Cryptology, Proceedings Asiacrypt'94, LNCS 917*, J. Pieprzyk and R. Safavi-Naini, Eds., Springer-Verlag, 1995, pp. 307–321. 174
67. B. Preneel, "Analysis and design of cryptographic hash functions," *Doctoral Dissertation*, Katholieke Universiteit Leuven, 1993. 160, 168, 171
68. B. Preneel, "Cryptographic primitives for information authentication — State of the art," *State of the Art in Applied Cryptography, LNCS 1528*, B. Preneel and V. Rijmen, Eds., Springer-Verlag, 1998, pp. 50–105. 168
69. B. Preneel, R. Govaerts, J. Vandewalle, "Hash functions based on block ciphers: a synthetic approach," *Advances in Cryptology, Proceedings Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 368–378. 170, 173
70. B. Preneel, P.C. van Oorschot, "MDx-MAC and building fast MACs from hash functions," *Advances in Cryptology, Proceedings Crypto'95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 1–14. 159
71. J.-J. Quisquater, J.-P. Delescaille, "How easy is collision search ? Application to DES," *Advances in Cryptology, Proceedings Eurocrypt'89, LNCS 434*, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 429–434. 166
72. J.-J. Quisquater, J.-P. Delescaille, "How easy is collision search. New results and applications to DES," *Advances in Cryptology, Proceedings Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 408–413. 167
73. M.O. Rabin, "Digitalized signatures," in *"Foundations of Secure Computation,"* R. Lipton, R. DeMillo, Eds., Academic Press, New York, 1978, pp. 155-166. 159
74. RIPE, *"Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040),"* *LNCS 1007*, A. Bosselaers and B. Preneel, Eds., Springer-Verlag, 1995. 176
75. R.L. Rivest, "The MD4 message digest algorithm," *Advances in Cryptology, Proceedings Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 303–311. 176
76. R.L. Rivest, "The MD5 message-digest algorithm," *Request for Comments (RFC) 1321*, Internet Activities Board, Internet Privacy Task Force, April 1992. 176
77. R.L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications ACM*, Vol. 21, February 1978, pp. 120–126. 172
78. J. Rompel, "One-way functions are necessary and sufficient for secure signatures," *Proc. 22nd ACM Symposium on the Theory of Computing*, 1990, pp. 387–394. 162

79. A. Russell, "Necessary and sufficient conditions for collision-free hashing," *Journal of Cryptology*, Vol. 8, No. 2, 1995, pp. 87–99. 168
80. G.J. Simmons, "A survey of information authentication," in *"Contemporary Cryptology: The Science of Information Integrity,"* G.J. Simmons, Ed., IEEE Press, 1991, pp. 381–419. 163
81. G.J. Simmons, "How to insure that data acquired to verify treat compliance are trustworthy," in *"Contemporary Cryptology: The Science of Information Integrity,"* G.J. Simmons, Ed., IEEE Press, 1991, pp. 615–630. 163
82. D. Simon, "Finding collisions on a one-way street: Can secure hash functions be based on general assumptions?" *Advances in Cryptology, Proceedings Eurocrypt'98, LNCS 1403*, K. Nyberg, Ed., Springer-Verlag, 1998, pp. 334–345. 168, 175
83. D.R. Stinson, "Universal hashing and authentication codes," *Designs, Codes, and Cryptography*, Vol. 4, No. 4, 1994, pp. 369–380. See also *Advances in Cryptology, Proceedings Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1992, pp. 74–85. 164
84. J.-P. Tillich, G. Zémor, "Hashing with SL_2," *Advances in Cryptology, Proceedings Crypto'94, LNCS 839*, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 40–49. 175
85. P.C. van Oorschot, M.J. Wiener, "Parallel collision search with application to hash functions and discrete logarithms," *Proc. 2nd ACM Conference on Computer and Communications Security*, ACM, 1994, pp. 210–218 (final version to appear in Journal of Cryptology). 166
86. M.N. Wegman, J.L. Carter, "New hash functions and their use in authentication and set equality," *Journal of Computer and System Sciences*, Vol. 22, No. 3, 1981, pp. 265–279. 163, 164
87. G. Yuval, "How to swindle Rabin," *Cryptologia*, Vol. 3, 1979, pp. 187–189. 166
88. G. Zémor, "Hash functions and Cayley graphs," *Designs, Codes, and Cryptography*, Vol. 4, No. 4, 1994, pp. 381–394. 175
89. Y. Zheng, T. Matsumoto, H. Imai, "Connections between several versions of one-way hash functions," *Proc. SCIS90, The 1990 Symposium on Cryptography and Information Security*, Nihondaira, Japan, Jan. 31–Feb. 2, 1990. 161

The Search for the Holy Grail in Quantum Cryptography

Louis Salvail

BRICS, Basic Research in Computer Science of the Danish National Research Foundation, Department of Computer Science,University of Århus, Ny Munkegade, building 540,DK-8000 Århus C, Denmark.
salvail@daimi.aau.dk

Abstract. In 1982, Bennett and Brassard suggested a new way to provide privacy in long distance communications with security based on the correctness of the basic principles of quantum mechanics. The scheme allows two parties, Alice and Bob, sharing no secret information in the first place, to exchange messages that nobody else can figure out. The only requirement is a quantum channel and a normal phone line connecting the two parties. The fact that quantum mechanics provides unconditional secure communications is a remarkable result that cannot be achieved by classical techniques alone. Apart from secure communication, cryptography is also interested in tasks that aim at protecting one party against a potentially dishonest peer. This scenario, called secure two-party computation, is usually modelled by a function $f(x_A, x_B)$ where x_A and x_B are Alice's and Bob's secret input respectively. They would like to execute a protocol that produces $z = f(x_A, x_B)$ to both parties without disclosing their secret input to the other party. The only information about a secret input that can be leaked toward the other party is what the output z itself discloses about it. Unlike secure communication, secure two-party computation does not assume that Alice and Bob are honest. One honest party's input should remain secret whatever the other party's behaviour. It is well-known that in order to find a protocol for secure two-party computation, one must have access to a secure bit commitment scheme. Unfortunately, in 1996 Mayers showed that no secure quantum bit commitment scheme exists. Similarly to the classical case (where trapdoor one-way functions are needed) quantum cryptography does not provide secure two-party computation for free. In this paper, we discuss the possibilities and limits of quantum cryptography for two-party computation. We describe the essential distinctions between classical and quantum cryptography in this scenario.

1 Introduction

Quantum cryptography aims at designing cryptographic protocols with security guaranteed by the fundamental laws of quantum mechanics. In 1982, Bennett and Brassard [1] proposed two quantum protocols: Quantum key distribution (QKD), and quantum coin tossing. Quantum key distribution allows two parties, Alice and Bob, who share no information to agree on a common secret key

I. Damgård (Ed.): Lectures on Data Security, LNCS 1561, pp. 183–216, 1999.

$\mathbf{k} \in \{0,1\}^l$ for some $l > 0$. Typically, once Alice and Bob share $\mathbf{k}$, Alice can encrypt any message $\mathbf{m} \in \{0,1\}^l$ as $\mathbf{c} = \mathbf{m} \oplus \mathbf{k}$. The ciphertext $\mathbf{c}$ is then sent to Bob over a normal channel that can be eavesdropped at will. It is well-known that this encryption method (called the one-time pad) does not leak information about $\mathbf{m}$ to an eavesdropper as long as $\mathbf{k}$ is unknown. This means that whenever a new message $\mathbf{m}$ has to be sent secretly, Alice and Bob first use QKD in order to get a fresh secret key $\mathbf{k}$ that is used for encrypting $\mathbf{m}$. The point here is that no classical method whatsoever can achieve this without relying upon some assumptions [42]. Classically, the security of secret-key exchange can be based upon a computing time limitation an attacker can spend in order to find the key [19]. However, it is very unlikely that one could prove that a secure classical cryptosystem would guarantee absolute security against eavesdroppers limited to spend only polynomial time. A proven security statement like this would imply that $\mathbf{P} \neq \mathbf{NP}$. On the other hand, if secret-key distribution is implemented quantumly then security can be achieved under the only assumption that the basic axioms of quantum mechanics are correct. This offers advantages compared to the classical cryptosystems since the notion of security is independent of the model of computation. This is important since it is possible that all practical public-key cryptosystems are secure against attackers modelled by Turing machines but not against attackers modelled by quantum Turing machines. As an example, RSA [39] and Diffie-Hellmann [19] cryptosystems are breakable by *quantum attackers* since the quantum computer can factorize and extract discrete logs in polynomial time [41].

The idea behind the Bennett-Brassard scheme for QKD [1,2] is that, any eavesdropper trying to get information by intercepting the communication on the quantum channel will be detected. This is because unknown quantum states cannot be observed without disturbing the state irreversibly. The disturbance can be detected by Alice and Bob by exchanging information over the public channel. The scheme ensures them that if they don't find too many errors it is because no threatening eavesdropping occurred during the quantum transmission. The key they are going to agree on should therefore be secret. Several papers have been written about the security of the Bennett-Brassard scheme. In [2], the scheme was shown secure against an attacker performing the so called *intercept-resend* attack. Intercept-resend attacks are the ones where the attacker keeps the original particles and resends others according to the outcome of a complete test (complete tests will be defined in section 3.1). The security of the scheme was shown against much stronger but still limited attackers in [6]. Very recently, the proof of security has been extended to cover all possible cases sound with quantum mechanics axioms [35]. It follows that quantum mechanics allows to achieve one of the most important cryptographic task without any assumption. Moreover, experimental implementations have demonstrated that quantum cryptography is also practical [2,37,43,24].

What about the other protocol introduced by Bennett and Brassard in 1982 [1]: Quantum coin tossing? A coin tossing protocol takes place between Alice and Bob and guarantees that a random bit $r \in \{0,1\}$ is generated [7]. Even when

one party is dishonest the outcome of the coin toss is random. This means that no dishonest party can influence the outcome. Unlike the protocol for QKD, it was already known by the authors that the proposal could be broken by a dishonest party able to produce and manipulate *entangled quantum states*. Loosely speaking, an entangled quantum state is the state of several particles such that:

- Observing one part of the system produces a random outcome and
- once the outcome is known, the state or the rest of the system is also known.

In other words, the state of each particle is correlated with the others. These states are rather difficult to prepare and were out of reach back in 1982. Today however, the entanglement needed in order to break the scheme can easily be produced in laboratory. From the beginning, coin tossing already appeared more difficult to achieve than QKD whereas classically, coin tossing is easier than secret-key distribution [23].

The coin tossing protocol proposed by Bennett and Brassard was in fact implementing a more powerful primitive called *bit commitment*. A bit commitment scheme allows Alice to commit to the value of a bit in a way that prevents Bob to learn it but also in a way that prevents Alice from changing her mind. A coin tossing is easily achieved using a bit commitment scheme:

- Alice commits on a random $r_A \in \{0,1\}$,
- Bob announces a random $r_B \in \{0,1\}$,
- Alice unveils r_A,
- Alice and Bob set $r = r_A \oplus r_B$.

The advantage of considering bit commitment is that it allows to prove knowledge of a statement without divulging it [10,20]. This kind of cryptographic task is important for solving natural cryptographic problems like identification, Zero-Knowledge proofs of Knowledge, etc... However there are tasks that even bit commitment cannot help to solve.

An *oblivious transfer* is a protocol that allows Alice to send Bob $x \in \{0,1\}$ in such a way that:

- Bob receives x with probability $\frac{1}{2}$ and knows it. When x is not received, Bob gets no information on x.
- Alice has no information on whether or not Bob received x.

Classically, it would be a major breakthrough if one could show that bit commitment and oblivious transfer can be based on the same computational assumptions [23]. Oblivious transfer seems strictly more powerful than bit commitment in the classical world. It allows to build bit commitment quite easily but the opposite will turn out to be true only if the existence of one-way functions implies the existence of trapdoor one-way functions. Coin tossing, bit commitment and oblivious transfer are all protocols involving two parties who want to cooperate while respecting their privacy. The most general task one can imagine in this model is the so called *secure two-party computation* (S2PC). A protocol for S2PC

is a generic protocol between Alice and Bob taking as input the description of a function $f : \{0,1\}^N \times \{0,1\}^N \to \{0,1\}^M$ and secret strings $x_A, x_B \in \{0,1\}^N$ for Alice and Bob respectively. The output is the value $f(x_A, x_B)$ that is made available to both parties. The protocol is secure if

1. it computes the correct output and
2. it leaks, to each player, no more information than $f(x_A, x_B)$ about the input of the other party.

Although S2PC seems quite general, an oblivious transfer is sufficient in order for a secure protocol to exists [26,18]. It follows that the most general primitive for solving any secure two-party computation is oblivious transfer.

From the above, it is natural to ask if oblivious transfer can be implemented quantumly. A positive answer would allow to base almost all modern cryptography upon the correctness of quantum mechanics, that is upon the laws of physics as we observe them. Oblivious transfer can therefore be seem as the Holy Grail of quantum cryptography.

1.1 Overview

Basically, quantum mechanics allows to transmit information in a way that is similar to a transmission through a binary symmetric channel. Quantum mechanics, by virtue of the uncertainty principle, allows to encode information in such a way that the receiver cannot decode it all the time. Measuring an arbitrary quantum state destroys it and does not extract all the information. Measurements are therefore not repeatable so the uncertainty about the measured state always remains. This inherent noisiness is at the basis of all quantum protocols including the one for secret-key distribution. Noisy channels, at least some of them, are powerful cryptographic primitives since they allow to build secure protocols for oblivious transfer [16]. In 1991, Bennett, Brassard, Crépeau and Skubiszewska proposed a quantum protocol for oblivious transfer [5]. Their protocol assume that Alice and Bob have access, as a black-box primitive, to a secure bit commitment scheme. Under this assumption, several results about the security of the scheme were shown [5,15,36,46]. The result of Yao [46], showed that the scheme is secure according to the laws of quantum mechanics and given bit commitment as a black-box. The result showed that bit commitment is sufficient to build a quantum oblivious transfer whereas classically this seems impossible.

There were reasons to be optimistic in 1995; the Holy Grail was in sight. Not for long though! In 1995, Mayers [32] broke the most serious candidate for quantum bit commitment [12] (although at that time it was even not considered as a candidate but as a genuine bit commitment scheme). Then, things got worse. In 1996, Mayers [33] and independently Lo and Chau [27] have given a general attack that can be applied on general quantum protocols for bit commitment. Mayers' construction [33,34] turns out to be so general that the existence of quantum bit commitment, with security relying merely upon the correctness of

quantum mechanics, was ruled out. Quantum bit commitment as its classical counterpart, needs extra assumption in order to be implemented. However, the classical and quantum assumptions can be of very different and independent nature [40]. It is of interest to have different sets of independent and realistic assumptions under which bit commitment and, more generally, oblivious transfer are possible. This allows to choose the model (classical or quantum) that suited the best the requirements of a particular application.

This paper describe the main steps in the search for secure quantum oblivious transfer. We shall see how quantum mechanics principles help in implementing a flavour of noisy channel as a primitive. We describe how to use this primitive to implement oblivious transfer given a black-box for bit commitment. We then describe Mayers' attack that breaks any quantum bit commitment. The description of the attack is a good starting point for getting accustomed to the weirdness of quantum information. It exhibits highly non classical behaviour and more importantly, it suggests how to look at quantum protocols in order not to *over classicize* their behaviour. It has been demonstrated many times, that thinking classically about the security of a quantum protocol can lead to false conclusions.

1.2 Content

In section 2, we introduce the mathematical concepts that are used throughout the paper. In section 3, we define quantum states and measurements using the standard physical representation. In section 4 we describe the standard way to encode obliviously classical information in quantum states. In section 5, we show how to reduce quantum oblivious transfer to the oblivious quantum encoding given a bit commitment scheme. In section 6 we describe Mayers' attack against any quantum bit commitment scheme. We conclude in section 7.

2 Mathematical Background

Here, we introduce a suitable vector space for the representation of quantum objects. We then introduce the definitions and basic properties of linear operators relevant to our discussion. More complete information can be found in almost any book about the basic of linear algebra.

2.1 Vectors and Vector Spaces

In quantum mechanics, states, system evolutions and measurements are all represented by objects in a complex vector space. An appropriate vector space is called Hilbert space which is, for our purposes, not different from the complex vector space with the scalar or inner product defined. In the following we denote by α^* the complex conjugate of any number $\alpha \in \mathbb{C}$. Let $\mathbf{u} = (u_1, \ldots, u_n), \mathbf{v} =$

$(v_1, \ldots, v_n) \in \mathcal{H}$ be two arbitrary vectors which belong in the same arbitrary Hilbert space $\mathcal{H}$. The inner product $\langle \mathbf{u}, \mathbf{v} \rangle \in \mathbb{C}$ between $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^{n} u_i^* v_i.$$

From the inner product (or scalar product) we define the norm (or length) $\|\mathbf{v}\|$ of vector $\mathbf{v} \in \mathcal{H}$ by $\|\mathbf{v}\|^2 = \langle \mathbf{v}, \mathbf{v} \rangle \in \mathbb{R}$. Two vectors $\mathbf{v}$ and $\mathbf{w}$ are orthogonal if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$. We say that a vector is *normalized* if its norm is 1. As usual, any vector $\mathbf{v} \in \mathcal{H}$ can be written as a linear combination of an infinite number of possible basis. In the following, $\mathcal{H}_n$ stands for the n-dimensional Hilbert space. A basis $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$ for $\mathcal{H}_n$ is said to be *orthonormal* if for all $1 \leq i \neq j \leq n$, we have that $\langle \mathbf{e}_i, \mathbf{e}_j \rangle = 0$ and $\|\mathbf{e}_i\| = 1$.

2.2 Dirac's Notation

A very popular notation for vectors and operators in an Hilbert space is the Dirac's notation. In Dirac's notation, vectors representing states are denoted by a *ket*. For any vector $\mathbf{v} = (v_1, \ldots, v_m) \in \mathcal{H}$, we write the state of a quantum attribute by $|\mathbf{v}\rangle$. One can see $|\mathbf{v}\rangle$ as the column vector:

$$|\mathbf{v}\rangle = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix}.$$

The *ket* notation allows to simplify expressions. In particular, it is often convenient to drop the description of vector $\mathbf{v}$ using only symbolic notations. One possible orthonormal basis for $\mathcal{H}_2$ is $+ = \{(1,0),(0,1)\}$. Basis $+$ is called the *standard* or *computational* or *rectilinear basis*. The orthonormal vectors for the standard basis are $+ = \{|\mathbf{0}\rangle, |\mathbf{1}\rangle\} = \{|\mathbf{0}\rangle_+, |\mathbf{1}\rangle_+\}$. Another important orthonormal basis in $\mathcal{H}_2$ is the *diagonal basis* $\times = \{(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), (\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) = \{|\mathbf{0}\rangle_\times, |\mathbf{1}\rangle_\times\}$.

Together with the *ket* comes the *bra* notation. If $\mathbf{v} = (v_1, \ldots, v_m) \in \mathcal{H}_m$ then the *bra* of $\mathbf{v}$ is noted $\langle \mathbf{v}|$ and is defined as $\langle \mathbf{v}| = (v_1^*, v_2^*, \ldots, v_m^*)$.

*Bra*s and *ket*s can be combined in order to denote operations. For $\mathbf{u} = (u_1, \ldots, u_m), \mathbf{v} = (v_1, \ldots, v_m) \in \mathcal{H}_m$ we have that $\langle \mathbf{v}|\mathbf{u}\rangle = \sum_{i=1}^{m} u_i^* v_i$ is the inner product between $\mathbf{u}$ and $\mathbf{v}$. Another operation sometime called the *dyadic* is denoted by $|\mathbf{u}\rangle\langle\mathbf{v}|$ and is such that

$$|\mathbf{u}\rangle\langle\mathbf{v}| = \begin{pmatrix} u_1 v_1^* & u_1 v_2^* & \ldots & u_1 v_m^* \\ u_2 v_1^* & u_2 v_2^* & \ldots & u_2 v_m^* \\ \vdots & \vdots & \vdots & \vdots \\ u_m v_1^* & u_m v_2^* & \ldots & u_m v_m^* \end{pmatrix}.$$

For any $\mathbf{v} \in \mathcal{H}_m$, $|\mathbf{v}\rangle\langle\mathbf{v}|$ is a matrix $V = \{v_{ij}\}_{1 \leq i,j \leq m}$ such that for all $i \neq j$ we have $v_{ij} = v_{ji}^*$ and $v_{ii} \in \mathbb{R}$. In the following and except when stated otherwise we

shall use vectors with only real components. In this case, the *bra* and the *ket* of vector $\mathbf{v}$ have the same components, the first one being $\mathbf{v}$ as a row vector and the later being $\mathbf{v}$ as a column vector. The inner product between $\mathbf{u} = (u_1, \ldots, u_m) \in \mathbb{R}^m$ and $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{R}^m$ is simply $\sum_{i=1}^{m} u_i v_i$.

2.3 Unitary Evolution

We shall see in next section that vectors in a Hilbert space represent quantum states. The possible evolution of a quantum state can always be described by a *unitary transformation*. We say that a transformation in a m-dimensional Hilbert space is *unitary* if it can be written as a bijective mapping between two orthonormal bases. The following transformation is unitary and acts in a 2-dimensional Hilbert space:

$$\begin{aligned} H : |\mathbf{0}\rangle &\mapsto \tfrac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle) \\ |\mathbf{1}\rangle &\mapsto \tfrac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle) \end{aligned}$$

Any unitary transformation acting in a m-dimensional Hilbert space can easily be written as a $m \times m$ matrix. We only have to label each column and each row by one vector of the basis $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$ we start with. The matrix entry labelled $(\mathbf{e}_i, \mathbf{e}_j)$ contains the complex number $\alpha_{i,j}$ that appears in front of vector $\mathbf{e}_j$ when the input state is $\mathbf{e}_i$. For example, the matrix form for H is:

$$H \;=\; \begin{array}{c|cc} & |\mathbf{0}\rangle & |\mathbf{1}\rangle \\ \hline |\mathbf{0}\rangle & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ |\mathbf{1}\rangle & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{array} \;=\; \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}.$$

In the following we will also use the *sign shift operator* S acting on vectors in $\mathcal{H}_2$ and defined as

$$\begin{aligned} S : |\mathbf{0}\rangle &\mapsto \;\; |\mathbf{0}\rangle \\ |\mathbf{1}\rangle &\mapsto -|\mathbf{1}\rangle \end{aligned}$$

For any vector $\mathbf{v} = (v_1, v_2) \in \mathcal{H}_2$, S applied on $\mathbf{v}$ produces the vector $\mathbf{v}' = (v_1, -v_2)$. The matrix representation of S is

$$S = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Any unitary transformation U has an inverse $U^{-1} = U^{\dagger}$ where $U^{\dagger}$ is the transposed conjugate of U (also called the Hermitian conjugate). One important property of unitary transforms is that they always preserved the inner product namely (i.e. for all $\mathbf{u}, \mathbf{v} \in \mathcal{H}$ we have that $\langle \mathbf{u}|\mathbf{v}\rangle = \langle U\mathbf{u}|U\mathbf{v}\rangle$).

Throughout this paper, we shall denote operators by capital letters. When we write $A \in \mathcal{H}$, we mean that A is an operator acting on vectors in $\mathcal{H}$.

2.4 Relevant Operators

A special case of operators, called Hermitians, will be useful in order to define what is a measurement of a quantum state. An operator $A \in \mathcal{H}_n$ is Hermitian if when A is expressed by $n \times n$ matrix $\{a_{ij}\}_{1 \leq i,j \leq n}$ we have that

1. for all $i \in \{1, \ldots, n\}$ the element $a_{ii} \in \mathbb{R}$. This means that all principal diagonal elements are real.
2. for all $i \neq j$, $a_{ij} = a_{ji}^*$.

An Hermitian operator A is always such that $A = A^\dagger$. When A contains only real elements, then A is Hermitian if and only if A is symmetric. Projections are special cases of Hermitian operators:

Definition 1. *An Hermitian operator P that satisfies $P = PP$ is called a* projection.

The condition $P = PP$ translates what we intuitively consider a projection, namely that a projection does not transform vectors that are parallel to the rays on which it projects. One can show that A is Hermitian in $\mathcal{H}_m$ if and only if it can be written for some $l \leq m$ as,

$$A = \sum_{i=1}^{l} a_i P_i \tag{1}$$

where the P_i's are projection operators projecting on mutually orthogonal rays. We say that $\mathbf{v}$ is an *eigenvector* with *eigenvalue* $a \in \mathbb{C}$ if A is such that $a\mathbf{v} = A\mathbf{v}$. The zero vector $\mathbf{0}$ is not an eigenvector but $a = 0$ is a possible eigenvalue. The set $E_A = \{a_i\}_{i=1}^l$ is the set of eigenvalues of A and the decomposition appearing in equation 1 is called the *spectral decomposition* of A. If $\#E_A = m$ then the spectral decomposition is unique and all projections are into orthogonal subspaces of dimension 1 (i.e. they project on rays). One can verify that all Hermitian operators have only real eigenvalues. The following projection operators are relevant to our discussion:

$$\mathbb{P}_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \mathbb{P}_{\frac{\pi}{4}} = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \mathbb{P}_{\frac{\pi}{2}} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \mathbb{P}_{\frac{3\pi}{4}} = \frac{1}{2}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}.$$

In the above, projection $\mathbb{P}_\alpha$ for $\alpha \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$ is the projection on the ray (i.e. one dimensional subspace) at angle α with vector $(1, 0)$. The projection operator $P_{\mathbf{v}}$ on the ray parallel to the normalized vector $\mathbf{v} \in \mathcal{H}$ is $P_{\mathbf{v}} = |\mathbf{v}\rangle\langle\mathbf{v}|$. For instance, the above projections $\mathbb{P}_0 = |\mathbf{0}\rangle\langle\mathbf{0}|$, $\mathbb{P}_{\frac{\pi}{4}} = |\mathbf{0}\rangle_\times\langle\mathbf{0}|$, $\mathbb{P}_{\frac{\pi}{2}} = |\mathbf{1}\rangle\langle\mathbf{1}|$, and $\mathbb{P}_{\frac{3\pi}{4}} = |\mathbf{1}\rangle_\times\langle\mathbf{1}|$.

The *trace* $\mathrm{Tr}(A)$ of an operator $A \in \mathcal{H}$, is the sum of its principal diagonal elements. More formally, we write

$$\mathrm{Tr}(A) = \sum_{\mathbf{e} \in \mathbf{E}} \langle \mathbf{e}|A\mathbf{e}\rangle \tag{2}$$

for any basis $\mathbf{E}$ for $\mathcal{H}$. It is easy to verify that any projection P is such that $\mathrm{Tr}(P) = 1$. The trace has the following properties:

1. $\mathrm{Tr}(A+B) = \mathrm{Tr}(A) + \mathrm{Tr}(B)$,
2. $\mathrm{Tr}(cA) = c\mathrm{Tr}(A)$ for $c \in \mathbb{C}$ and
3. $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$.

It follows from equation 1 that if A has eigenvalues E_A then

$$\mathrm{Tr}(A) = \sum_{a \in E_A} a\mathrm{Tr}(P_a) = \sum_{a \in E_A} a. \tag{3}$$

We shall see in section 3.4 that general quantum states are modelled by a special class of operators characterized by their traces:

Definition 2. *An operator D is a* density operator *if $Tr(D) = 1$.*

2.5 Space Extension

Two Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ can be merged together in order to get a larger one $\mathcal{H}$ containing both of them. Let m_1 and m_2 the dimension of $\mathcal{H}_1$ and $\mathcal{H}_2$ and let $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_{m_1}\}$ and $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_{m_2}\}$ be orthonormal bases for $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively. We define the *tensor product* operation "$\otimes$" that allows, given $\mathbf{E}$ and $\mathbf{F}$, to get a new orthonormal basis $\mathbf{H}$ for the $m_1 m_2$ dimensional Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$. The tensor product is dyadic operation acting upon vectors. If vector $\mathbf{e} = (e_1, \ldots, e_{m_1})$ and $\mathbf{f} = (f_1, \ldots, f_{m_2})$ then we define:

$$\mathbf{e} \otimes \mathbf{f} = \begin{pmatrix} e_1 f_1 \\ e_1 f_2 \\ \vdots \\ e_1 f_{m_2} \\ e_2 f_1 \\ \vdots \\ e_{m_1} f_{m_2} \end{pmatrix}. \tag{4}$$

It is now possible to define $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ as the Hilbert space generated by the orthonormal basis $\mathbf{H} = \{\mathbf{e}_1 \otimes \mathbf{f}_1, \mathbf{e}_1 \otimes \mathbf{f}_2, \ldots, \mathbf{e}_{m_1} \otimes \mathbf{f}_{m_2}\}$. The tensor product operation can also be generalized in order to deal with operators as well. Assume A is an operator in the m_1 dimensional Hilbert space $\mathcal{H}_1$ and A' is an operator in the m_2 dimensional Hilbert space $\mathcal{H}_2$. Assume $A = \{a_{ij}\}_{1 \le i,j \le m_1}$ and $A' = \{a'_{ij}\}_{1 \le i,j \le m_2}$ are expressed as $m_1 \times m_1$ and $m_2 \times m_2$ squares matrices respectively. The composite operator $A \otimes A' \in \mathcal{H}_1 \otimes \mathcal{H}_2$ is defined as

$$A \otimes A' = \begin{pmatrix} a_{11}A' & a_{12}A' & \ldots & a_{1m_1}A' \\ a_{21}A' & a_{22}A' & \ldots & a_{2m_1}A' \\ \vdots & \vdots & \vdots & \vdots \\ a_{m_1 1}A' & a_{m_1 2}A' & \ldots & a_{m_1 m_1}A' \end{pmatrix}.$$

3 Quantum States

In quantum cryptography, classical information is encoded in the state of a quantum system. In this section, we describe what is meant by a quantum state. We shall define *pure states* as a special case of all quantum states. Complete measurements of quantum states are also discussed. Finally, we introduce the most general quantum states allowed by the theory: *quantum mixture*.

3.1 Maximal Tests

Before giving the definition of a quantum state, it is convenient to introduce *maximal quantum tests* [38]. Suppose you want to observe the property of a quantum system that can possibly take N different values. If the test you devise allows to distinguish between all N possibilities, we say that it is a *maximal quantum test*. A N-outcome measurement of this property implements a maximal quantum test. A test that gives only partial information about the measured property is said to be a *partial test*.

3.2 Pure States

If a quantum system is prepared in such a way that one can devise a maximal quantum test that yields with certainty a particular outcome then we say that the quantum system is in *pure state*. It follows that measuring several times a pure state yields always the same outcome [38].

In quantum mechanics, pure states are described by normalized vectors in some Hilbert space. If the maximal test for a pure state has n possible outcomes then the state is described by a vector $|\phi\rangle \in \mathcal{H}_n$. The polarization state of a photon is the usual way to encode information in quantum cryptography. Pure states for the polarization of a photon can be tested by a 2-outcome maximal test. It follows that the polarization state (i.e. here we drop the word *pure* adopting the convention that unless stated otherwise a state is pure) is described by a normalized vector in $\mathcal{H}_2$. As an example, $|\mathbf{0}\rangle, |\mathbf{1}\rangle, \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle) = H|\mathbf{0}\rangle$ and $\frac{1}{\sqrt{2}}(-|\mathbf{0}\rangle + |\mathbf{1}\rangle) = H|\mathbf{1}\rangle$ are all possible states for the polarization of a photon. The pure state $|\mathbf{0}\rangle_\times = \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle_+ + |\mathbf{1}\rangle_+)$ is said to be in *superposition* of pure states $|\mathbf{0}\rangle_+$ and $|\mathbf{1}\rangle_+$.

It is easy to verify that the tensor product operation $|\phi\rangle \otimes |\phi'\rangle$ for $\phi \in \mathcal{H}$ and $\phi' \in \mathcal{H}'$ preserves the purity of the two quantum states $|\phi\rangle$ and $|\phi'\rangle$. That means that whenever $|\phi\rangle \in \mathcal{H}$ and $|\phi'\rangle \in \mathcal{H}'$ are brought together then the new composite system remains in pure state. This must be the case since the maximal test in $\mathcal{H}$ for $|\phi\rangle$ followed by the maximal test in $\mathcal{H}'$ for $|\phi'\rangle$ defined one maximal test in $\mathcal{H} \otimes \mathcal{H}'$ for $|\phi\rangle \otimes |\phi'\rangle$.

The time evolution of a pure state (and also for mixture as defined in section 3.4) is always unitary and any unitary transformation is a possible evolution of a quantum state. Let $U \in \mathcal{H}_{2^l}$ be any unitary transformation acting on vectors

in Hilbert space $\mathcal{H}_{2^l} = \bigotimes_{i=1}^{l} \mathcal{H}_2$. Let $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{2^l}\}$ be a basis for $\mathcal{H}_{2^l}$ and let $|\phi\rangle \in \mathcal{H}_{2^l}$ be any pure state in $\mathcal{H}_{2^l}$. We have that

$$U|\phi\rangle = U \sum_{j=1}^{2^l} \alpha_j |\mathbf{e}_j\rangle = \sum_{j=1}^{2^l} \alpha_j U|\mathbf{e}_j\rangle$$

for $\alpha_j \in \mathbb{C}$ and $\sum_j |\alpha_j|^2 = 1$. This means that U is in fact applied simultaneously to each element appearing in the superposition $|\phi\rangle$. This kind of parallel computation is very important for speeding up classical algorithms using quantum phenomena. As we shall see in section 6, it has also important consequences in cryptography.

3.3 Complete Measurements

We have seen that pure states are quantum states for which there exists a maximal test giving a predictable outcome (thus repeatable). Measurements are implementations of the testing procedures. Quantum mechanics define complete measurements as measurements implementing a maximal test for some quantum states. Formally,

Definition 3. *A* complete *or* Von Neumann *measurement of a quantum state in $\mathcal{H}_n$ is described by an Hermitian operator $M \in \mathcal{H}_n$ with n distinct eigenvalues $E_M = \{a_1, \ldots, a_n\}$. Each eigenvalue $a \in E_M$ is a possible outcome for the measurement.*

From definition 3, the outcomes of a complete measurement M are in one to one correspondence with the set of orthogonal projections $\mathbf{P}_M$ appearing in M's spectral decomposition, since the decomposition is unique when all eigenvalues are distinct. Let $P_a \in \mathbf{P}_M$ be the projection associated with eigenvalue $a \in E_M$. It is always possible to write $P_a = |\psi_a\rangle\langle\psi_a|$ for a normalized vector $|\psi_a\rangle$ that is an eigenvector of M. Definition 3 does not describe the behaviour of complete measurements but just the way they are modelled. In order to understand what is a complete measurement, we have to specify what is the probability to observe the outcome corresponding to any eigenvalues in E_M and what happens to the system once the outcome has been observed. This is where quantum measurements and consequently quantum states differ from the classical ones. When a system Φ in quantum state $|\phi\rangle \in \mathcal{H}_n$ is measured by a complete measurement M, the following is always satisfied:

- The outcome corresponding to $a \in E_M$ is obtained with probability $p_\phi(a) = \langle\phi|P_a|\phi\rangle$.
- If $a \in E_M$ is the outcome then the state of Φ after the measurement is $|\psi_a\rangle$.

Any normalized vector $|\phi\rangle \in \mathcal{H}_n$ can be tested maximally by a complete measurement M having projection $P_a = |\phi\rangle\langle\phi|$ in its spectral decomposition. The outcome of M applied upon $|\phi\rangle$ is predictable since the eigenvalue a satisfies $p_\phi(a) = \langle\phi|P_a|\phi\rangle = \langle\phi|\phi\rangle\langle\phi|\phi\rangle = 1$. It is always possible to find such an M so

that normalized vectors really describe pure states. Since projections and normalized vectors are in one to one correspondence, one can describe a pure state by a projection as well. It follows that a pure state $|\phi\rangle$ can always be written as the projection operator $P_\phi = |\phi\rangle\langle\phi|$. From definition 2 and equation 1 we have that any pure state P_ϕ is represented by a density operator but not all density operators represent pure states, as we shall see in next section.

We have seen that complete measurements in $\mathcal{H}_n$ are modelled by Hermitian operators $M \in \mathcal{H}_n$ having n distinct eigenvalues. The set of eigenvectors E_M for M defines a basis for $\mathcal{H}_n$. It follows that a complete measurement can also be described by a orthonormal basis $\mathbf{F}$ for $\mathcal{H}_n$ where each $\mathbf{v} \in \mathbf{F}$ is a possible outcome of M. Another equivalent way to specify a complete measurement is a set of the n orthogonal projections $\mathbf{P}_M$ in $\mathcal{H}_n$ appearing in M's spectral decomposition. Each projection $P \in \mathbf{P}_M$ is one of the possible orthogonal rays on which M projects the initial state. Using this representation of complete measurements, the following two complete measurements

$$\mathbb{M}_+ = \{\mathbb{P}_0, \mathbb{P}_{\frac{\pi}{2}}\} \text{ and } \mathbb{M}_\times = \{\mathbb{P}_{\frac{\pi}{4}}, \mathbb{P}_{\frac{3\pi}{4}}\}$$

will be used extensively in the following.

3.4 Mixed States

Suppose an observer is sitting next to a source of photons S. The dynamic of S is such that with probability $\frac{1}{2}$ a photon in state $|\mathbf{0}\rangle$ is sent and with probability $\frac{1}{2}$ a photon in state $|\mathbf{1}\rangle$ is sent. The behaviour of S can be described by a probability distribution $\mathcal{D}_S = \{(\frac{1}{2}, |\mathbf{0}\rangle), (\frac{1}{2}, |\mathbf{1}\rangle)\}$ over pure states in $\mathcal{H}_2$. Clearly, the next photon π that is going to be transmitted by S is not in pure state since no complete measurement can be defined such that the outcome will be predictable by the observer. To verify this, observe that if M represents a maximal test on $\mathcal{D}_S$ then we have that $p_{|\mathbf{0}\rangle}(a_0) = p_{|\mathbf{1}\rangle}(a_1) = 1$ where $a_0 \neq a_1$ are two eigenvalues of M. Let $p(a_0)$ and $p(a_1)$ be the probability to observe a_0 and a_1 respectively when the next photon transmitted by S is measured. We have that

$$p(a_0) = \frac{1}{2}p_{|\mathbf{0}\rangle}(a_0) = \frac{1}{2}p_{|\mathbf{1}\rangle}(a_1) = p(a_1) = \frac{1}{2}.$$

We conclude that no implementation of a maximal test is predictable when applied on the next particle produced by S. The quantum state transmitted by S is therefore not in pure state.

Definition 4. *A* quantum mixture *is a probability distribution over pure states in some Hilbert space $\mathcal{H}$. Moreover, any quantum state is a quantum mixture. In general we say that a quantum system is in a* mixed state *if it is not in pure state.*

Definition 4 does not say how a measurement behave when a mixed state is observed. Let $\mathcal{D} = \{(p_i, |\mathbf{s}_i\rangle)\}_{i=1}^{l}$ be an arbitrary quantum mixture in Hilbert

space $\mathcal{H}$. We define the operator $\rho_{\mathcal{D}} \in \mathcal{H}$ as

$$\rho_{\mathcal{D}} = \sum_{i=1}^{l} p_i |\mathbf{s}_i\rangle\langle\mathbf{s}_i|. \tag{5}$$

By definition, $\rho_{\mathcal{D}}$ is a density operator since each $|\mathbf{s}_i\rangle\langle\mathbf{s}_i|$ has trace 1. Equation 5 reminds us of the spectral decomposition except for the pure states $|\mathbf{s}_i\rangle$'s that are not necessarily orthogonal. Since $\rho_{\mathcal{D}}$ is Hermitian, it is always possible to write

$$\rho_{\mathcal{D}} = \sum_{i=1}^{l} p_i |\mathbf{s}_i\rangle\langle\mathbf{s}_i| = \sum_{i=1}^{m} \tilde{p}_i P_i \tag{6}$$

where for all $i \neq j$, P_i and P_j are orthogonal and $\sum_{i=1}^{m} \tilde{p}_i = 1$. One consequence of equation 6 is that two different mixtures $\mathcal{D}$ and $\mathcal{D}'$ may share the same density matrix. Let $P_{\mathbf{s}_i} = |\mathbf{s}_i\rangle\langle\mathbf{s}_i|$ be the projection operator associated with the pure state $|\mathbf{s}_i\rangle$. We have that

$$\rho_{\mathcal{D}} = \sum_{i=1}^{l} p_i P_{\mathbf{s}_i} = \sum_{i=1}^{m} \tilde{p}_i P_i = \rho_{\mathcal{D}'}$$

where $\mathcal{D}' = \{(\tilde{p}_i, P_i)\}_{i=1}^{m}$. The physical interpretation is that several and different physical preparations can produce the same physical state.

If we return to our interpretation of a quantum mixture as a probability distribution over pure states, it becomes clear how behave a complete measurement on it. Each time an observer performs a measurement on a quantum mixture $\mathcal{D}$, the measurement is applied on a random pure state $|\phi\rangle \in \mathcal{H}$ picked according to $\mathcal{D}$. Let $\rho_{\mathcal{D}}$ be the density operator for mixture $\mathcal{D} = \{(p_i, |\mathbf{s}_i\rangle)\}_i$. Let $M = \sum_i a_i P_i \in \mathcal{H}$ be a complete measurement with outcomes (or eigenvalues) $E_M = \{a_i\}_i$ and such that all P_i's are orthogonal. The behaviour of M when applied upon $\mathcal{D}$ satisfies the following:

- The probability $p_{\mathcal{D}}(a)$ that the complete measurement M gives the outcome $a \in E_M$ is

$$p_{\mathcal{D}}(a) = \sum_{(p,|\mathbf{s}\rangle)\in\mathcal{D}} p\langle\mathbf{s}|P_a|\mathbf{s}\rangle = \mathrm{Tr}(P_a \rho_{\mathcal{D}}) \tag{7}$$

  where P_a is the projection associated to the eigenvalue a in the spectral decomposition of M.
- After the outcome a has been observed,the state of the system becomes in pure state P_a.

Since the statistics of a measurement are completely specified by the density operator $\rho_{\mathcal{D}}$, it follows that two mixtures $\mathcal{D}$ and $\mathcal{D}'$ having the same density operator $\rho_{\mathcal{D}}$ behave the same when they are measured. We conclude that two

mixtures sharing the same density operator are indistinguishable by any physical process.

As an example, consider the mixture $\mathcal{D}$ produces by the source S and the new mixture $\mathcal{D}' = \{(\frac{1}{4}, |\mathbf{0}\rangle), (\frac{1}{4}, |\mathbf{1}\rangle), (\frac{1}{4}, |\mathbf{0}\rangle_\times), (\frac{1}{4}, |\mathbf{1}\rangle_\times)\}$ produces by the source S'. One can verify that

$$\begin{aligned}
\rho_{\mathcal{D}'} &= \frac{1}{4}(|\mathbf{0}\rangle\langle\mathbf{0}| + |\mathbf{1}\rangle\langle\mathbf{1}| + |\mathbf{0}\rangle_\times\langle\mathbf{0}| + |\mathbf{1}\rangle_\times\langle\mathbf{1}|) \\
&= \frac{1}{4}\left(\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & \frac{-1}{2} \\ \frac{-1}{2} & \frac{1}{2} \end{pmatrix}\right) \\
&= \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
&= \frac{1}{2}\mathbb{1} = \rho_{\mathcal{D}}.
\end{aligned}$$

It follows that no physical process can distinguish between sources S and S'. These two preparation methods are equivalent.

In the following we sometime denote quantum systems in $\mathcal{H}_2$ by *qubits*. As we have seen, a qubit cannot store more than 1 classical bit of information since any complete test on it has only two possible outcomes. This explains the analogy between "qubits" and "bits".

Henceforth, we shall write $\rho \in \mathcal{H}$, for a density operator ρ, if it acts on vectors in $\mathcal{H}$.

4 Oblivious Encoding of Information

In this section we shall see that the indistinguishability between quantum mixed states sharing the same density matrix leads to an encoding of classical information that cannot be recovered with 100% reliability by the receiver. This kind of encoding scheme is relevant to cryptography since it allows to perform non trivial cryptographic tasks. For instance consider the classical binary symmetric channel (BSC) that allows to send bits with error probability $0 < \epsilon < \frac{1}{2}$. The transmission of a classical bit through a BSC does not disclose all information to the receiver since the communication is noisy. The sender does not have all the information neither since (s)he does not know whether the receiver got the bit or its complement. Crépeau and Kilian [16] have shown that a BSC allows to build a secure oblivious transfer protocol and thus provides all the power needed for secure two-party computation. Noisy channels can also be used to implement secure secret-key distribution protocols as, for example, Wyner's wire-tap channel [45] or Maurer's secret-key agreement from common information [30]. This *oblivious* encoding of information is what we would like to achieve based on quantum mechanics. It would allow to see the quantum channel like a noisy channel thus providing the power needed for secure two-party computation.

4.1 The BB84 Coding Scheme

The BB84 coding scheme has been introduced by Bennett and Brassard [1] in order to achieve quantum secret-key distribution. As we shall see, the coding scheme can also be used in order to implement a wide variety of cryptographic tasks using the same quantum transmission procedure. The coding implements some kind of noisy transfer of a classical bit.

The idea behind the BB84 coding scheme is that classical bits 0 and 1 are encoded by non-orthogonal states and therefore cannot be distinguished perfectly by any measurement. For, we define the two following basis in $\mathcal{H}_2$:

- The *rectilinear basis* $+ = \{|\mathbf{0}\rangle_+, |\mathbf{1}\rangle_+\}$
- The *diagonal basis* $\times = \{|\mathbf{0}\rangle_\times, |\mathbf{1}\rangle_\times\}$.

Each vector in the rectilinear and diagonal basis will be the encoding of a classical bit. The following quantum transmission scheme is the main tool used in almost all quantum protocols. It is the standard quantum transmission between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$:

`BB84 Quantum Transmission`

1. $\mathcal{S}$ picks a random $b \in_R \{0,1\}$ and a random $\theta \in_R \{+,\times\}$,
2. $\mathcal{R}$ picks a random $\widehat{\theta} \in_R \{+,\times\}$,
3. $\mathcal{S}$ sends a photon π in quantum state $|b\rangle_\theta$ through the quantum channel,
4. $\mathcal{R}$ measures π with the complete measurement $\mathbb{M}_{\widehat{\theta}}$ and records the outcome

$$\widehat{b} = \begin{cases} 0 \text{ if } |\mathbf{0}\rangle_{\widehat{\theta}} \text{ is observed,} \\ 1 \text{ if } |\mathbf{1}\rangle_{\widehat{\theta}} \text{ is observed.} \end{cases}$$

One BB84 quantum transmission produces a photon π with polarization in mixed state $\mathcal{D}_{BB84} = \{(\frac{1}{4}, |\mathbf{0}\rangle_+), (\frac{1}{4}, |\mathbf{1}\rangle_+), (\frac{1}{4}, |\mathbf{0}\rangle_\times), (\frac{1}{4}, |\mathbf{1}\rangle_\times)\}$. From equation 6, the mixture $\mathcal{D}_{BB84}$ is described by the density operator

$$\begin{aligned} \rho_{BB84} &= \frac{1}{4}\left(|\mathbf{0}\rangle_+\langle\mathbf{0}| + |\mathbf{1}\rangle_+\langle 1| + |\mathbf{0}\rangle_\times\langle\mathbf{0}| + |\mathbf{1}\rangle_\times\langle 1|\right) \\ &= \frac{1}{2}\mathbb{1}. \end{aligned}$$

On the receiving end, $\mathcal{R}$ measures π either with the complete measurement $\mathbb{M}_+$ or with $\mathbb{M}_\times$, each being chosen with probability $\frac{1}{2}$. For any $\widehat{\theta} \in \{+,\times\}$ the Hermitian operator $\mathbb{M}_{\widehat{\theta}}$ with eigenvalues $E_{\widehat{\theta}} = \{0,1\}$ can be written as

$$\mathbb{M}_+ = \mathbb{P}_0 = |\mathbf{0}\rangle_+\langle\mathbf{0}| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ and } \mathbb{M}_\times = \mathbb{P}_{\frac{\pi}{4}} = |\mathbf{0}\rangle_\times\langle\mathbf{0}| = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

Suppose $\mathcal{S}$ sends π in state $|\mathbf{0}\rangle$ (i.e. when $\mathcal{S}$ chooses $b = 0$ and $\theta = +$) and $\mathcal{R}$ measures in basis $\widehat{\theta} = +$. The probability $p_+(0)$ that $\mathcal{R}$ gets the outcome 0 thus

setting $\widehat{b} = 0 = b$ is

$$p_+(0) = \langle \mathbf{0}|\mathbb{P}_0|\mathbf{0}\rangle = \langle \mathbf{0}| \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} |\mathbf{0}\rangle = \langle \mathbf{0}|\mathbf{0}\rangle = 1. \tag{8}$$

If $\mathcal{R}$ would have chosen measurement $\mathbb{M}_\times$ instead then the probability $p_\times(0)$ for $\mathcal{R}$ to decode correctly would be

$$p_\times(0) = \langle \mathbf{0}|\mathbb{P}_{\frac{\pi}{4}}|\mathbf{0}\rangle = \langle \mathbf{0}| \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} |\mathbf{0}\rangle = \langle \mathbf{0}|(\frac{1}{2}, \frac{1}{2})\rangle = \frac{1}{2}. \tag{9}$$

Equations 8 and 9 show the property of *obliviousness* of the BB84 quantum transmission. If $\mathcal{R}$ chooses $\widehat{\theta} = \theta$ then the decoded bit $\widehat{b} = b$ with probability 1. However, if $\mathcal{R}$ chooses $\widehat{\theta} \neq \theta$ then the decoded bit $\widehat{b}$ is completely random. The BB84 coding scheme is symmetric and behaves the same way if the basis θ is $\times$ instead of $+$ and if the bit $b = 1$ instead of 0. It follows that the probability p_s that $\widehat{b} = b$ is

$$p_s = \mathrm{P}\left(\widehat{\theta} = \theta\right) + \frac{1}{2}\mathrm{P}\left(\widehat{\theta} \neq \theta\right) = \frac{3}{4}. \tag{10}$$

From equation 10 we conclude that if $\mathcal{S}$ and $\mathcal{R}$ follow the protocol honestly then the BB84 quantum transmission implements a BSC with error probability $\frac{1}{4}$.

4.2 BB84 Is Oblivious

We now look at what happens when one party involved in a BB84 quantum transmission does not behave according the rules. We shall see what advantage a dishonest receiver $\mathcal{R}^*$ gets by choosing complete measurements different from $\mathbb{M}_+$ and $\mathbb{M}_\times$.

The goal for $\mathcal{R}^*$ is to figure out the bit b with better probability than $\frac{3}{4}$. In other words, $\mathcal{R}^*$ is looking for a complete measurement that allows to distinguish between $\mathcal{D}_0 = \{(\frac{1}{2}, |\mathbf{0}\rangle_+), (\frac{1}{2}, |\mathbf{0}\rangle_\times)\}$ and $\mathcal{D}_1 = \{(\frac{1}{2}, |\mathbf{1}\rangle_+), (\frac{1}{2}, |\mathbf{1}\rangle_\times)\}$ more accurately than measurements $\mathbb{M}_+$ and $\mathbb{M}_\times$. Let ρ_0 and ρ_1 be the density operators for $\mathcal{D}_0$ and $\mathcal{D}_1$ respectively. We have that

$$\rho_0 = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{pmatrix} \text{ and } \rho_1 = \begin{pmatrix} \frac{1}{4} & \frac{-1}{4} \\ \frac{-1}{4} & \frac{3}{4} \end{pmatrix} \tag{11}$$

Using equation 7, one can verify that

$$p_s = \mathrm{P}\left(\widehat{b} \neq b\right) = \frac{1}{4}\left(\mathrm{Tr}(\mathbb{P}_0\rho_0) + \mathrm{Tr}(\mathbb{P}_{\frac{\pi}{4}}\rho_0) + \mathrm{Tr}(\mathbb{P}_{\frac{\pi}{2}}\rho_1) + \mathrm{Tr}(\mathbb{P}_{\frac{3\pi}{4}}\rho_1)\right) = \frac{3}{4}.$$

Let $M_{\mathrm{B}} = \{\mathbb{P}_{\frac{\pi}{8}}, \mathbb{P}_{\frac{5\pi}{8}}\}$ be the complete measurement with possible outcomes $\mathbb{P}_{\frac{\pi}{8}} = |\mathbf{b_0}\rangle\langle\mathbf{b_0}|$ and $\mathbb{P}_{\frac{5\pi}{8}} = \mathbb{1}_2 - \mathbb{P}_{\frac{\pi}{8}} = |\mathbf{b_1}\rangle\langle\mathbf{b_1}|$ where $\mathbf{b_0} = (\cos\frac{\pi}{8}, \sin\frac{\pi}{8})$ and $\mathbf{b_1} =$

$(-\sin\frac{\pi}{8}, \cos\frac{\pi}{8})$. Assume $\mathcal{R}^*$ measures π with $M_{\mathtt{B}}$ and let $\tilde{p}_s(b)$ be the probability to get $\widehat{b} = b$ when ρ_b is sent. We have that

$$\tilde{p}_s(0) = \mathrm{Tr}(\mathbb{P}_{\frac{\pi}{8}}\rho_0) = \cos^2\frac{\pi}{8} \tag{12}$$

$$\tilde{p}_s(1) = \mathrm{Tr}(\mathbb{P}_{\frac{5\pi}{8}}\rho_1) = \cos^2\frac{\pi}{8}. \tag{13}$$

Equations 12 and 13 show that if $\mathcal{R}^*$ wants to maximize its information about b, he has advantage to apply measurement $M_{\mathtt{B}}$ on π. In this case, the probability to decode b correctly is about 85% instead of $\frac{3}{4}$ when $\mathbb{M}_+$ or $\mathbb{M}_\times$ is applied. We can show that $M_{\mathtt{B}}$ is in fact the measurement that maximizes the probability to decode b correctly. The spectral decomposition of density operators ρ_0 and ρ_1 is,

$$\rho_0 = \cos^2\frac{\pi}{8}|\mathbf{b_0}\rangle\langle\mathbf{b_0}| + \sin^2\frac{\pi}{8}|\mathbf{b_1}\rangle\langle\mathbf{b_1}| \text{ and } \rho_1 = \sin^2\frac{\pi}{8}|\mathbf{b_0}\rangle\langle\mathbf{b_0}| + \cos^2\frac{\pi}{8}|\mathbf{b_1}\rangle\langle\mathbf{b_1}|.$$

This means that $\mathcal{D}_0 = \{(\cos^2\frac{\pi}{8}, |\mathbf{b_0}\rangle), (\sin^2\frac{\pi}{8}, |\mathbf{b_1}\rangle)\}$ and $\mathcal{D}_1 = \{(\sin^2\frac{\pi}{8}, |\mathbf{b_0}\rangle), (\cos^2\frac{\pi}{8}, |\mathbf{b_1}\rangle)\}$. Therefore, sending b using the BB84 coding scheme behaves like if it was sent through a BSC with error probability $\sin^2\frac{\pi}{8}$ whatever measurement $\mathcal{R}$ performs. It follows that the quantum state ρ_b for any $b \in \{0,1\}$ does not carry more information than $\mathbf{H}(\cos^2\frac{\pi}{8}, \sin^2\frac{\pi}{8})$ about b. The BB84 coding scheme is therefore inherently oblivious.

The BB84 coding scheme hides completely $\mathcal{S}$'s basis $\theta \in \{+, \times\}$. To see this, consider the mixed state $\mathcal{D}_\theta$ corresponding to a photon π polarized in basis θ. We have that $\mathcal{D}_\theta = \{(\frac{1}{2}, |\mathbf{0}\rangle_\theta), (\frac{1}{2}, |\mathbf{1}\rangle_\theta)\}$. Let ρ_+ and $\rho_\times$ be the density operators corresponding to $\mathcal{D}_+$ and $\mathcal{D}_\times$ respectively. One can easily verify that,

$$\begin{aligned}\rho_+ &= \frac{1}{2}(|\mathbf{0}\rangle_+\langle\mathbf{0}| + |\mathbf{1}\rangle_+\langle\mathbf{1}|)\\ &= \frac{1}{2}(|\mathbf{0}\rangle_\times\langle\mathbf{0}| + |\mathbf{1}\rangle_\times\langle\mathbf{1}|)\\ &= \rho_\times.\end{aligned}$$

This implies that, given a BB84 photon π, it is impossible to figure out what basis θ has been used by $\mathcal{S}$. This holds for any quantum measurement $\mathcal{R}$ could perform on π. The basis θ is perfectly concealed by the BB84 coding scheme.

4.3 BB84 as a Quantum Primitive

The BB84 coding scheme is the quantum ingredient of most quantum protocols [1,2,12,17]. The difference between all these protocols is the classical communication taking place after the quantum transmission. The BB84 coding scheme is a kind of *universal* cryptographic primitive. Typically, a quantum protocol requires many BB84 transmissions upon which the classical part of the protocol is based. The parties involved in the classical part communicate only via the public channel. The classical phase is very often the only task dependent part of a quantum protocol.

In the following, we write $\langle (b,\theta),(\widehat{b},\widehat{\theta})\rangle \leftarrow \mathrm{BB84}_N$ to denote N independent BB84 quantum transmissions of photons $\pi_1,\pi_2,\ldots,\pi_N$. $\mathcal{S}$'s random bits are $b = b_1,b_2,\ldots,b_N$ and the N choices for the polarization bases are $\theta = \theta_1,\theta_2,\ldots,\theta_N \in \{+,\times\}^N$. The N particles $\pi_1,\pi_2,\ldots,\pi_N$ that are sent through the quantum channel are therefore in composite state $|b_1\rangle_{\theta_1} \otimes |b_2\rangle_{\theta_2} \otimes \ldots \otimes |b_N\rangle_{\theta_N} \in \mathcal{H}_{2^N}$. On each received particle π_i, $\mathcal{R}$ performs the measurement $\mathbb{M}_{\widehat{\theta}_i}$ for $\widehat{\theta}_i \in \{+,\times\}$ providing the outcome $\widehat{b}_i$.

5 From BB84 to Quantum Oblivious Transfer

The BB84 coding scheme shows similarities with the description of an oblivious transfer. In BB84, the receiver gets the bit b with probability $\frac{1}{2}$ (i.e. when $\widehat{\theta} = \theta$). The only difference between a BB84 transmission and an oblivious transfer is that in the BB84 case, $\mathcal{R}$ does not know if he receives the bit or not.

One way to tell $\mathcal{R}$ whether or not he gets b, would be for $\mathcal{S}$ to announce the basis θ used to transmit b. If the receiver finds out that $\widehat{\theta} = \theta$ then $\widehat{b} = b$. Otherwise, the bit received $\widehat{b}$ is not correlated with the bit sent. However, this method allows $\mathcal{R}$ to cheat and receive $\widehat{b} = b$ all the time! $\mathcal{R}$ just stores the photon he receives and waits (without disturbing it) for $\mathcal{S}$ to announce θ. Once $\mathcal{R}$ knows θ, he measures the photon with measurement $\mathbb{M}_\theta$ thus recovering b perfectly. One way to overcome this problem would be to require $\mathcal{R}$ to commit on $\widehat{\theta}$ and $\widehat{b}$ before $\mathcal{S}$ announces θ. With probability $\kappa > 0$, $\mathcal{S}$ asks $\mathcal{R}$ to open the commitment. $\mathcal{S}$ then verifies that whenever $\widehat{\theta} = \theta$ $\mathcal{R}$ obtained the outcome $\widehat{b} = b$. If it is not the case then $\mathcal{S}$ stops the execution. With probability $1-\kappa$, $\mathcal{S}$ announces θ allowing $\mathcal{R}$ to find out if he receives b. We have made a step forward but the method does not implement an oblivious transfer yet. $\mathcal{R}$ has still a probability $1-\kappa$ not to be asked to open the commitment. This allows him to take a chance and to commit on random values allowing him not to measure the received particle. The probability of not being caught remains better than $1-\kappa$ (i.e. in fact the probability of being caught is $\frac{\kappa}{4}$).

The above construction is the idea behind the quantum oblivious transfer protocol of Bennett, Brassard, Crépeau and Skuwbiszewska [5] called the BBCS protocol. Below, we present a slight modification of the BBCS protocol allowing Alice to send to Bob the bit x by oblivious transfer. N BB84 transmissions are performed out of which about one half have been received perfectly. One subset S_c, for $c \in \{0,1\}$, contains the positions i such that $\theta_i = \widehat{\theta}_i$ whilst the set S_{1-c} contains the positions i such that $\widehat{\theta}_i \neq \theta_i$. The two sets S_0 and S_1 are announced to Alice without telling her the bit c. Alice encodes the bit x she wants to transmit by OT using the bits in positions in S_q for a random $q \in \{0,1\}$. The encoding allows Bob to recover x if and only if $q = c$ which happens with probability exactly $\frac{1}{2}$. The protocol needs a bit commitment scheme in order to be implemented securely. Let us assume that $\mathtt{BC}(w)$, for $w \in \{0,1\}$, is a secure commitment of bit w.

BBCS QOT Scheme(x)

1. Alice and Bob execute $\langle (b,\theta),(\widehat{b},\widehat{\theta})\rangle \leftarrow \mathrm{BB84}_N$ where Alice is $\mathcal{S}$ and Bob is $\mathcal{R}$,
2. Bob sends to Alice the commitments $\{(\mathtt{BC}(\widehat{b}_i),\mathtt{BC}(\widehat{\theta}_i))\}_{i=1}^{N}$,
3. Alice selects a random subset of positions $I \subset \{1,\ldots,N\}$ that she announces to Bob,
4. Bob opens $\{(\mathtt{BC}(\widehat{b}_i),\mathtt{BC}(\widehat{\theta}_i))\}_{i\in I}$ allowing Alice to verify that for all $i \in I$ such that $\widehat{\theta}_i = \theta_i$ it is the case that $\widehat{b}_i = b_i$. If Alice finds errors she stops the execution else let $J = \{1,\ldots,N\} \setminus I$ be the set of untested positions,
5. Alice announces $\theta_J = \{\theta_i | i \in J\}$, Bob picks a random $c \in \{0,1\}$ and sets $S_c = \{i \in J | \theta_i = \widehat{\theta}_i\}, S_{1-c} = J \setminus S_c$,
6. Bob announces (S_0, S_1) to Alice (he keeps c secret),
7. Alice picks $q \in_R \{0,1\}$ and announces q together with $r = x \oplus \bigoplus_{i\in S_q} b_i$ to Bob,
8. If $q = c$ then Bob computes $x = r \oplus \bigoplus_{i\in S_c} \widehat{b}_i = r \oplus \bigoplus_{i\in S_c} b_i$ else Bob does not receive x.

The security of the scheme is based upon the inability for Bob to decode reliably the b_i's for all transmissions. Intuitively, the commitments ensure Alice that Bob measured completely the particles he received before she announces $\theta = \theta_1,\ldots,\theta_N$. Therefore, it should be the case that there exists a $z \in \{0,1\}$ such that the subset of positions S_z satisfies

$$|\mathrm{P}\left(\bigoplus_{i\in S_z} b_i = \bigoplus_{i\in S_z} \widehat{b}_i\right) - \frac{1}{2}| \leq 2^{-\alpha N}$$

for some $\alpha > 0$. If Bob follows the protocol then for each photon π_i we have that $\widehat{\theta}_i \neq \theta_i$ with probability $\frac{1}{2}$. We have seen that in this case, $\mathrm{P}\left(\widehat{b}_i = b_i | \widehat{\theta}_i \neq \theta_i\right) = \frac{1}{2}$. It follows that there exists $z \in \{0,1\}$ such that $\#\{i \in S_z | \widehat{\theta}_i \neq \theta_i\} \geq \frac{(1-\mu)\#S_z}{2}$ for any $\mu > 0$ as long as N is large enough. In that case, the bit $\bigoplus_{i\in S_z} b_i$ cannot be approximated by Bob. Since Alice encodes x in the XOR of all bits in S_q, for a random $q \in \{0,1\}$, with probability $\frac{1}{2}$ we have that $q = z$ and Bob is unable to obtain information about x.

5.1 Security and Generalized Measurements

In this section we quickly review what is known about the security of the BBCS protocol against dishonest parties that would take advantage of more elaborate quantum processes. Complete measurements as described in section 3.3, are not the only way an attacker can try to get extra information. Quantum mechanics allows generalized measurements to be performed. General measurements can extract information from a quantum state in such a way that the disturbance caused by the measurement process is minimized. In particular, if one is willing to get less information than what is achievable through a complete measurement, then a generalized measurement (also incomplete) of the the original quantum

state can be done with no complete destruction of the initial state. One example of an incomplete measurement is the measurement that does nothing. This is formally written as the identity operator $\mathbb{1}$ which has only one eigenvalue $a = 1$ and therefore is not a complete measurement (of course!). In general, incomplete measurements are modelled by Hermitian operators with fewer distinct eigenvalues than the dimension of the Hilbert space in which they operate. They cannot give more information than complete measurements do but can nevertheless be completed later in order to get a complete measurement. For example, it is always possible to apply the useless measurement $\mathbb{1}$ on a quantum state $|\phi\rangle$ and later measures the untouched state $|\phi\rangle$ with a complete measurement. The result is simply the same as if $|\phi\rangle$ would have been measured completely the first time. Or is it? One can see that even the useless measurement $\mathbb{1}$ allows to break BBCS if no commitment was used. An incomplete measurement that gives information about the observed state $|\phi\rangle$ must destroy a part of the initial state. In general, more distinct eigenvalues your measurement has, more destructive it is (an example of a non-trivial incomplete measurement is given in section 6.4). Incomplete measurements can be useful to an attacker involved in a quantum protocol (as we have seen with BBCS using no commitment). The reason is that between the time the attacker performs the incomplete measurement and the time the measurement is completed, some extra information is obtained (i.e. the bases θ in the case of BBCS). With this extra information, the completion of the measurement can be chosen more cleverly than before whilst giving more information than if a complete measurement would have been chosen regardless of the extra information.

We can already verify that Alice has no way to learn whether or not the bit x has been received by Bob, as long as the commitments are concealing. This, because Bob chooses randomly how to measure each photon π_i and never gives information that would allow Alice to figure out what measurements were performed (if the commitments were not concealing Alice could easily find out!). Therefore, given S_0 and S_1, Alice has no information about $c \in \{0, 1\}$ such that S_c contains the positions i where $\theta_i = \widehat{\theta}_i$. It follows that no matter what Alice tries, it is always the case that $\mathrm{P}\,(q = c) = \frac{1}{2}$. Only Bob could cheat the protocol by measuring photons $\pi_1, \ldots, \pi_N$ using measurements of its choice.

If we make the extra assumption that Bob only performs complete measurements then the security of the scheme can be shown. To see how, assume that Bob returns a commitment $\mathtt{BC}(\widehat{\theta}, \widehat{b})$ with the property that if $\theta = \widehat{\theta}$ then $\widehat{b} = b$ with probability 1. It follows that Bob's measurement is the complete measurement $\mathbb{M}_\theta$. Clearly, Bob cannot get b more than half the time even once he gets to know θ since after $\mathbb{M}_\theta$ has been performed, the state of the original photon is irreversibly destroyed. Another strategy for Bob would be to return a commitment that has a small but nonzero probability of being caught (i.e. $\widehat{\theta} = \theta$ but $\widehat{b} \neq b$) by applying complete measurements different than $\mathbb{M}_+$ and $\mathbb{M}_\times$. This strategy does not help Bob in increasing its chance to receive the bit x as shown in [15].

In [36], Bob was allowed to perform generalized measurements on single BB84 qubits. These measurements are strictly more powerful than complete measurements but were shown not to allow Bob to cheat the protocol neither. The final piece was provided by Yao [46] who showed that, given a perfectly secure bit commitment scheme, QOT is secure against any strategy allowed by quantum mechanics. The BBCS scheme can also be modified to deal with imperfect apparatus whilst remaining secure.

5.2 Classical vs. Quantum Cryptography

Yao's proof of security for the BBCS scheme holds relative to the existence of a secure bit commitment scheme. It follows that the scheme described above does not provide security for free (as it is for quantum key distribution) but rather, reduce the security of QOT to the security of bit commitment. Nevertheless, we achieved something classical cryptography does not: secure oblivious transfer based on bit commitment. Classically, bit commitment can be built from any one-way function but oblivious transfer requires trapdoor one-way functions. It is very unlikely that one can find a proof that one-way functions and trapdoor one-way functions are in fact the same thing [23]. In the classical world, bit commitment is a weaker primitive than oblivious transfer. On the other hand, Yao's proof has shown that quantumly, oblivious transfer is reducible to bit commitment. It follows that oblivious transfer can be based on a weaker assumption in the quantum world (i.e. the existence of one-way functions) than in the classical world.

6 Quantum Bit Commitment

The next important question is whether or not QOT can be shown secure under the only assumption that quantum mechanics is correct. This would allow to base any secure two-party computation upon the same principles than quantum key distribution [31,35,6]. The first attempt to find a secure quantum bit commitment scheme is as old as the first protocol for quantum key distribution [1]. This first scheme was known to be insecure but it was believed that a secure one could be found. Several attempts were made in order to fix the original scheme [11,12]. The last one was even claimed to be unbreakable [12]. Unfortunately, two years later Mayers found a subtle flaw in the last proposal [32]. Afterward, Mayers realized that the flaw he found was not only due to the particular broken protocol but could be applied to a large class of quantum protocol for bit commitment [33]. This has also been observed independently by Lo and Chau [27]. It is now known that no quantum bit commitment exists with security based only on the correctness of quantum mechanics axioms [33,34].

In this section, we shall look at the general idea behind Mayers' proof and see why quantum mechanics completely forbids the existence of bit commitment. Apart from being used in the proof of [33], concepts introduce here are of independent interest. In particular, they show the striking difference between classical

and quantum information. Quantum information will appear much more elusive than its classical counterpart.

6.1 Purification

In this section we shall discuss the main tool needed in order to prove Mayers' theorem. It is shown how a quantum mixture can be embedded in a pure state. This process is called *purification* of a mixed state.

We start by considering an example taken from the BB84 coding scheme. Let BB84(0) be the possible BB84 transmissions of classical bit $b = 0$:

BB84(0)

1. $\mathcal{S}$ picks a random $\theta \in_R \{+, \times\}$,
2. $\mathcal{S}$ sends a photon π in quantum state $|0\rangle_\theta$ through the quantum channel.

Clearly, the mixture associated with one transmission through BB84(0) is $\mathcal{D}_0 = \{(\frac{1}{2}, |0\rangle_+), (\frac{1}{2}, |0\rangle_\times)\}$ which has density operator ρ_0, as described in section 4.2. Now let us introduce a similar way to send one of the random state $|0\rangle_+$ and $|0\rangle_\times$ without requiring $\mathcal{S}$ to pick a random basis as in step 1 of BB84(0):

BB84*(0)

1. $\mathcal{S}$ prepares $|\Psi\rangle = \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle \otimes |0\rangle_+ + |\mathbf{1}\rangle \otimes |0\rangle_\times) \in \mathcal{H}_4$,
2. $\mathcal{S}$ keeps the first (the left one) particle and sends the other (the right one).
3. $\mathcal{S}$ measures in the standard basis "+" the particle he has kept. If the outcome is 0 then he sets $\theta = +$ otherwise he sets $\theta = \times$.

In BB84*(0), $\mathcal{S}$ never uses coin flips in order to determine which one of the two possible states $|0\rangle_+$ or $|0\rangle_\times$ is going to be sent. The coin is provided by adding an extra particle, called the auxiliary system (or ancilla), that is in superposition of the two possible outcomes of the coin toss. The auxiliary system is entangled with the particle that stores the qubit to be sent. When the the state of the auxiliary system is measured then the state of the qubit can be determined. Before the measurement, the states of the qubit and the auxiliary system were unknown. To see this, consider the standard complete measurement that $\mathcal{S}$ applies on $|\Psi\rangle$. The pure state $|\Psi\rangle$ can be written as

$$|\Psi\rangle = \frac{1}{\sqrt{2}} \left(\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \right) = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}.$$

When $\mathcal{S}$ executes $\mathbb{M}_+$, he will observe the outcome $\mathbb{P}_0$ (i.e. which is the projection on $|\mathbf{0}\rangle$) with probability

$$p(0) = \langle \Psi || (\mathbb{P}_0 \otimes \mathbb{1}_2) \Psi \rangle = \langle \Psi | \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \mathbb{1}_2 | \Psi \rangle = \frac{1}{2}.$$

This means that with probability $\frac{1}{2}$, $\mathcal{S}$ observes 0 and $|\Psi\rangle$ is projected in the state

$$|\Psi_0\rangle = |\mathbf{0}\rangle \otimes |0\rangle_+. \tag{14}$$

With probability also $p(1) = 1 - p(0) = \frac{1}{2}$, the standard measurement produces the outcome 1 that projects the original state $|\Psi\rangle$ into $|\Psi_1\rangle$ defined as

$$|\Psi_1\rangle = |\mathbf{1}\rangle \otimes |0\rangle_\times. \tag{15}$$

Equations 14 and 15 imply that the receiver $\mathcal{R}$ is going to receive $|0\rangle_+$ with probability $p(0) = \frac{1}{2}$ and $|0\rangle_\times$ with probability $p(1) = \frac{1}{2}$. On $\mathcal{R}$'s point of view, the mixed state he receives is $\mathcal{D}_0$ as it is for BB84(0). Since the density operators of BB84(0) and BB84*(0) are the same, $\mathcal{R}$ has no way to tell what preparation $\mathcal{S}$ is using to send the qubit.

Now, consider the mixed state $\mathcal{D}_\mathtt{B} = \{(\cos^2 \frac{\pi}{8}, |\mathbf{b_0}\rangle), (\sin^2 \frac{\pi}{8}, |\mathbf{b_1}\rangle)\}$, and its purification

$$|\Psi_\mathtt{B}\rangle = \cos\frac{\pi}{8}|\mathbf{0}\rangle \otimes |\mathbf{b_0}\rangle + \sin\frac{\pi}{8}|\mathbf{1}\rangle \otimes |\mathbf{b_1}\rangle.$$

If the leftmost particle is measured with the standard measurement $\mathbb{M}_+$ then with probability $p_\mathtt{B}(0) = \cos^2 \frac{\pi}{8}$ the outcome 0 will be observed. We see that $p_\mathtt{B}(0)$ and $p(0)$ (defined above) are not the same but, as we have seen in section 4.2, $\mathcal{D}_\mathtt{B}$ and $\mathcal{D}_0$ share the same density operator ρ_0. The two purifications $|\Psi\rangle$ and $|\Psi_\mathtt{B}\rangle$ are therefore two different purifications for the same mixed state.

It is always possible to replace a probabilistic procedure as BB84(0) by an equivalent one where no coin toss is necessary. Consider an arbitrary mixture $\mathcal{D} = \{(p_i, |\mathbf{s_i}\rangle)\}_{i=1}^{l}$ where each $|\mathbf{s_i}\rangle$ belongs to the Hilbert space $\mathcal{H}$. Let $\mathcal{H}_1$ be an Hilbert space of dimension $\mathtt{l} = 2^{\lceil \lg_2 l \rceil}$. A system $\Psi_\mathcal{D} \in \mathcal{H}_1 \otimes \mathcal{H}$ in pure state

$$|\Psi_\mathcal{D}\rangle = \sum_{i=1}^{l} \sqrt{p_i}|\mathbf{i}\rangle \otimes |\mathbf{s_i}\rangle \tag{16}$$

is called a *purification* of $\mathcal{D}$. The auxiliary system (the leftmost register) is used to store indices of all possible coin toss outcomes. Let $w \in \{1, \ldots, l\}$ be written in binary as $\text{Binary}(w) = w_0, w_1, \ldots, w_1$. A value for w is encoded in pure state $|\mathbf{w}\rangle = |\mathbf{w_0}\rangle \otimes |\mathbf{w_1}\rangle \otimes \ldots \otimes |\mathbf{w_1}\rangle \in \mathcal{H}_1$. The state of equation 16 is guaranteed, when the leftmost particle is measured with $\mathbb{M}_+$, to give the outcome w with probability p_w in which case the rightmost particle is projected in state $|\mathbf{s_w}\rangle$. This is exactly the behaviour of mixed state $\mathcal{D}$ that is provided by the entanglement of an auxiliary system with the pure states in $\mathcal{D}$.

One strange thing about purifications is that it allows to perform operations upon the result of a coin toss without knowing the outcome of the coin toss. For instance, in BB84*(0) it is not necessary for $\mathcal{S}$ to measure the register he keeps. Not measuring it changes nothing to what $\mathcal{R}$ will receive, it is still the mixed state $\mathcal{D}_0$ that is sent. But if $\mathcal{S}$ does not measure the kept register then he does not know what state has actually been transmitted although he knows that it has been chosen according to $\mathcal{D}_0$. The only way of doing this classically would be to require the sender to forget what he had done.

6.2 Purifying a Coin Toss

The most simple case of purification is probably the coin toss. Suppose that one instruction in a quantum protocol requires to flip a biased coin $\mathtt{C}(p)$ as follows

$$\mathtt{C}(p) = \begin{cases} 1 \text{ with probability } p, \\ 0 \text{ with probability } 1-p. \end{cases}$$

Unlike classically, it is possible to store a coin toss in a quantum memory without forcing the outcome. This is straightforward to achieve by preparing a quantum register $\Psi_{\mathtt{C}(p)}$ in state

$$|\Psi_{\mathtt{C}(p)}\rangle = \sqrt{p}\,|\mathbf{1}\rangle + \sqrt{1-p}\,|\mathbf{0}\rangle.$$

By measuring $|\Psi_{\mathtt{C}(p)}\rangle$ with measurement $\mathbb{M}_+$ one gets the outcome $\mathbb{P}_0$ with probability $1-p$ and the outcome $\mathbb{P}_{\frac{\pi}{2}}$ with probability p. As long as the measurement is not performed, the register $|\Psi_{\mathtt{C}(p)}\rangle$ keeps both possibilities in superposition. The coin toss itself is a quantum object. Classically, a coins toss does not exist until the outcome is known.

Assume that a quantum register is in mixed state $\rho \in \mathcal{H}$ and V_0 and V_1 are two unitary transforms acting on states in $\mathcal{H}$. One application of quantum coin toss is the purification of the sequence of instructions:

1. Pick $r \in \{0,1\}$ such that $\mathrm{P}\,(r=1) = p$,
2. Apply V_r to ρ for some arbitrary density operator $\rho \in \mathcal{H}$.

Let us define an unitary transformation $V \in \mathcal{H}_2 \otimes \mathcal{H}$ acting on a one qubit register $\Psi_{\mathtt{C}(p)}$ in addition to the register in state ρ. Transformation V simply applies V_0 to ρ if register $|\Psi_{\mathtt{C}(p)}\rangle = |\mathbf{0}\rangle$ and applies V_1 to ρ if $|\Psi_{\mathtt{C}(p)}\rangle = |\mathbf{1}\rangle$. Let $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$ be an orthonormal basis for $\mathcal{H}$. Transformation V is defined as

$$\begin{aligned} V : |\mathbf{0}\rangle \otimes |\mathbf{e}_1\rangle &\mapsto |\mathbf{0}\rangle \otimes V_0\,|\mathbf{e}_1\rangle \\ |\mathbf{0}\rangle \otimes |\mathbf{e}_2\rangle &\mapsto |\mathbf{0}\rangle \otimes V_0\,|\mathbf{e}_2\rangle \\ \vdots \qquad\qquad &\vdots \quad \vdots \\ |\mathbf{0}\rangle \otimes |\mathbf{e}_m\rangle &\mapsto |\mathbf{0}\rangle \otimes V_0\,|\mathbf{e}_m\rangle \\ |\mathbf{1}\rangle \otimes |\mathbf{e}_1\rangle &\mapsto |\mathbf{1}\rangle \otimes V_1\,|\mathbf{e}_1\rangle \\ \vdots \qquad\qquad &\vdots \quad \vdots \\ |\mathbf{1}\rangle \otimes |\mathbf{e}_m\rangle &\mapsto |\mathbf{1}\rangle \otimes V_1\,|\mathbf{e}_m\rangle. \end{aligned}$$

The fact that both V_0 and V_1 are unitary ensures that V is also unitary. Using a quantum coin toss and transformation V, one can purify the above instructions as follows:

1. Prepare a register in state $|\Psi_{\mathtt{C}(p)}\rangle$
2. Apply $V|\Psi_{\mathtt{C}(p)}\rangle \otimes \rho$.

It can easily be shown that both procedures generate the same mixture. Measuring the leftmost register allows to select the coin toss outcome and consequently which of V_0 or V_1 has been applied on the rightmost register.

In the above construction, the number of outcomes for the coin toss is irrelevant. Any coin toss distribution $D = \{(p_i, i)\}_i$ can be purified the same way.

6.3 Purifying a Measurement

The purification process is not only possible on $\mathcal{S}$'s side of the quantum channel. It can also be done on the receiving end. Typically, $\mathcal{R}$ is supposed to measure a particle π with some measurement M picked according to a distribution $D_M = \{(p_1, M_1), (p_2, M_2), \ldots, (p_l, M_l)\}$. A purification of such a process would allow to perform all possible measurements in superposition until $\mathcal{R}$ wants to know what measurement and what outcome he gets. When he does so, $\mathcal{R}$ gets the outcome of a measurement picked according distribution D_M.

Without loss of generality, let us assume that $D_M = \{(p_+, \mathbb{M}_+), (p_\times, \mathbb{M}_\times)\}$. The BB84 coding scheme corresponds to the special case $p_+ = p_\times = \frac{1}{2}$. Assume that a quantum register $\Psi_{\mathtt{C}(p_+)}$ in state $|\Psi_{\mathtt{C}(p_+)}\rangle \in \mathcal{H}_2$ contains a purification of the coin toss $\mathtt{C}(p_+)$ as described in the previous section. Let π be a qubit that $\mathcal{R}$ is supposed to measure according to D_M. We now define the unitary transformation $U_M \in \mathcal{H}_2 \otimes \mathcal{H}_2$ that perform the required purification:

$$
\begin{aligned}
U_M : \overbrace{|\mathbf{0}\rangle}^{\text{coin}} \otimes \overbrace{|\mathbf{0}\rangle}^{\pi} &\mapsto |\mathbf{0}\rangle \otimes |\mathbf{0}\rangle \\
|\mathbf{1}\rangle \otimes |\mathbf{0}\rangle &\mapsto \tfrac{1}{\sqrt{2}}|\mathbf{1}\rangle \otimes (|\mathbf{0}\rangle + |\mathbf{1}\rangle) \\
|\mathbf{0}\rangle \otimes |\mathbf{1}\rangle &\mapsto |\mathbf{0}\rangle \otimes |\mathbf{1}\rangle \\
|\mathbf{1}\rangle \otimes |\mathbf{1}\rangle &\mapsto \tfrac{1}{\sqrt{2}}|\mathbf{1}\rangle \otimes (|\mathbf{0}\rangle - |\mathbf{1}\rangle).
\end{aligned}
$$

The register containing the coin toss is the auxiliary system of the purification. Transformation U_M stores the measurement in the auxiliary system and stores the outcome in the system that encoded particle π initially. Let $|b\rangle_\theta$ be a BB84 qubit and let $|\Psi_{\mathtt{C}(p_+)}\rangle$ be the purification of an arbitrary coin toss. One can verify that

$$
\begin{aligned}
U_M(|\Psi_{\mathtt{C}(p_+)}\rangle \otimes |b\rangle_\theta) = \sqrt{p_+}|\mathbf{0}\rangle \otimes \left(\sqrt{p_+(0)}|\mathbf{0}\rangle \pm \sqrt{p_+(1)}|\mathbf{1}\rangle\right) \\
+\sqrt{p_\times}|\mathbf{1}\rangle \otimes \left(\sqrt{p_\times(0)}|\mathbf{0}\rangle \pm \sqrt{p_\times(1)}|\mathbf{1}\rangle\right)
\end{aligned}
$$

where $p_{\widehat{\theta}}(\widehat{b})$ is the probability of the outcome $\widehat{b}$ whenever the initial state is $|b\rangle_\theta$ and the measurement is $\mathbb{M}_{\widehat{\theta}}$. If the leftmost register is measured with $\mathbb{M}_+$ then the outcome $\mathbb{P}_0$ is obtained with probability p_+ and the rightmost register contains the possible outcomes of measurement $\mathbb{M}_+$ when applied to the BB84 state $|b\rangle_\theta$. Similarly, the outcome $\mathbb{P}_{\frac{\pi}{2}}$ is obtained with probability $p_\times$ and the rightmost register contains the possible outcomes of measurements $\mathbb{M}_\times$ when

applied on $|b\rangle_\theta$. If measurement $\mathbb{M}_+$ is applied on the rightmost particle first, an outcome $\widehat{b}$ is obtained without the measurement being completely specified. The leftmost register is in superposition of all possible measurements that can produce outcome $\widehat{b}$ when the initial state is $|b\rangle_\theta$. Purifying a random measurement and measuring the rightmost register (the outcome register) allows to get the outcome of an unknown measurement!

Suppose $\mathcal{R}$ is asked to perform a measurement $M \in \{\mathbb{M}_+, \mathbb{M}_\times\}$ according to distribution D_M on the particle π. Let $|b\rangle_\theta \in \mathcal{H}_2$ be an unknown BB84 state for π that is received by $\mathcal{R}$ through the quantum channel. The following implements a purification of this procedure given a register containing the coin toss $|\Psi_{\mathtt{C}(p_\times)}\rangle$ for choosing according to D_M:

1. $\mathcal{R}$ applies $|\Psi_M\rangle = U_M|\Psi_{\mathtt{C}(p_\times)}\rangle \otimes |b\rangle_\theta$.

The state $|\Psi_M\rangle$ contains a superposition of both possible measurements. If at some point after the BB84 transmission, $\mathcal{R}$ must announce the outcome of a random measurement $\mathbb{M}_{\widehat{\theta}}$ for $\widehat{\theta} \in \{+, \times\}$ according to D_M, then the measurement $\mathbb{M}_+$ applied to the rightmost register gives a possible outcome. To fix the measurement M, $\mathcal{R}$ only measures with $\mathbb{M}_+$ the leftmost register. If $\mathbb{P}_0$ is obtained then the selected measurement was $M = \mathbb{M}_+$ otherwise $M = \mathbb{M}_\times$ was selected. Applying U_M to a coin toss $\mathtt{C}(\frac{1}{2})$ register and a BB84 particle π purifies $\mathcal{R}$'s part of the BB84 transmission. The same technique can be used for sets of any N possible measurements by using a N-outcome quantum coin toss.

The measurement $\mathbb{M}_+$ performed by $\mathcal{R}$ on the leftmost register does nothing to the leftmost register and is formally defined as $M = \mathbb{M}_+ \otimes \mathbb{1}_2$. It is an incomplete measurement since it has only 2 distinct eigenvalues but acts in $\mathcal{H}_4$.

6.4 From One Purification to Another

In this section we shall argue that two purifications of the same mixed state are in fact equivalent. By equivalent we mean that one can transform a purification to another purification of the same mixture by acting only on the auxiliary part of the purification. This is a result of Hughston, Jozsa and Wootters [22].

Let us consider the unitary transformation $U^* = SH$ (see section 2.3) acting in $\mathcal{H}_2$ when applied on the auxiliary part of $|\Psi\rangle$:

$$\begin{aligned}
(U^* \otimes \mathbb{1}_2)|\Psi\rangle &= (SH \otimes \mathbb{1}_2)\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle \otimes |0\rangle_+ + |\mathbf{1}\rangle \otimes |0\rangle_\times) \\
&= \frac{1}{2}((|\mathbf{0}\rangle - |\mathbf{1}\rangle) \otimes |0\rangle_+ + (|\mathbf{0}\rangle + |\mathbf{1}\rangle) \otimes |0\rangle_\times) \\
&= \frac{1}{2}((|\mathbf{0}\rangle - |\mathbf{1}\rangle) \otimes (\cos\frac{\pi}{8}|\mathbf{b_0}\rangle - \sin\frac{\pi}{8}|\mathbf{b_1}\rangle) + \\
&\qquad (|\mathbf{0}\rangle + |\mathbf{1}\rangle) \otimes (\cos\frac{\pi}{8}|\mathbf{b_0}\rangle + \sin\frac{\pi}{8}|\mathbf{b_1}\rangle)) \\
&= \cos\frac{\pi}{8}|\mathbf{0}\rangle \otimes |\mathbf{b_0}\rangle + \sin\frac{\pi}{8}|\mathbf{1}\rangle \otimes |\mathbf{b_1}\rangle \\
&= |\Psi_{\mathtt{B}}\rangle.
\end{aligned}$$

Applying U^* on the auxiliary part of $|\Psi\rangle$ transforms purification $|\Psi\rangle$ into purification $|\Psi_{\mathtt{B}}\rangle$. This allows $\mathcal{S}$ to decide which preparation $\mathcal{D}_0$ or $\mathcal{D}_{\mathtt{B}}$ he wants to use even after the particle is gone! $\mathcal{S}$ just prepares the purification of $\mathcal{D}_0$ and sends to $\mathcal{R}$ the leftmost particle keeping the auxiliary system. If at some point $\mathcal{S}$ wants to change his mind and wants to prepare the photon already sent using preparation $\mathcal{D}_{\mathtt{B}}$ instead, then he just applies U^* upon the auxiliary part.

The above construction is not a coincidence. Any pair of purifications $|\Psi\rangle$ and $|\Psi'\rangle$ for the same density operator is always related by an unitary transformation acting only on the auxiliary part of the purifications [22]. Let $\Psi \in \mathcal{H}_m \otimes \mathcal{H}_n$ be a purification of the density operator $\rho \in \mathcal{H}_n$. The Schmidt decomposition [22,38] allows to write $|\Psi\rangle$ as a sum of *bi-orthogonal* terms. This means that there exists $r \leq \min(m,n)$ (depending only on ρ) and two sets of orthonormal vectors $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^r$ and $\mathbf{F} = \{\mathbf{f}_i\}_{i=1}^r$ in $\mathcal{H}_m$ and $\mathcal{H}_n$ respectively, such that[1]

$$|\Psi\rangle = \sum_{i=1}^{r} \sqrt{\alpha_i} |\mathbf{e}_i\rangle \otimes |\mathbf{f}_i\rangle \tag{17}$$

where as usual $\sum_i |\alpha_i|^2 = 1$. In equation 17, the set $\{\alpha_i\}_{i=1}^r$ is the set of eigenvalues of $\rho \in \mathcal{H}_n$. Let $|\Psi'\rangle \in \mathcal{H}_m \otimes \mathcal{H}_n$ be another purification of density operator $\rho \in \mathcal{H}_n$. We make the assumption that the auxiliary system in Ψ' belongs to the same Hilbert space $\mathcal{H}_m$ than the auxiliary system for Ψ. This can be done without loss of generality by taking the larger Hilbert space whenever the auxiliary systems for Ψ and Ψ' are defined in different Hilbert spaces. From the Schmidt decomposition, there exists two sets of orthonormal vectors $\mathbf{E}' = \{\mathbf{e}'_i\}_{i=1}^r$ and $\mathbf{F}' = \{\mathbf{f}'_i\}_{i=1}^r$ such that

$$|\Psi'\rangle = \sum_{i=1}^{r} \sqrt{\alpha_i} |\mathbf{e}'_i\rangle \otimes |\mathbf{f}'_i\rangle. \tag{18}$$

Clearly, the unitary transformation $W \in \mathcal{H}_m$ defined for all $i \in \{1, \ldots, r\}$ as

$$W : |\mathbf{e}_i\rangle \mapsto |\mathbf{e}'_i\rangle$$

is such that

$$(W \otimes \mathbb{1}_n)|\Psi\rangle = |\Psi'\rangle$$

since the subsystem in $\mathcal{H}_n$ is the same mixed state ρ in both purifications. In this case, it can be shown that $\mathbf{F} = \mathbf{F}'$.

[1] More precisely, let $|\Psi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ be an arbitrary pure state and let $\rho = |\Psi\rangle\langle\Psi|$ be the associated projection. Let $\rho_1 = \mathrm{Tr}_{\mathcal{H}_1}(\rho)$ and $\rho_2 = \mathrm{Tr}_{\mathcal{H}_2}(\rho)$ be the partial trace of ρ over $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively. It is always the case that ρ_0 and ρ_1 share the same nonzero eigenvalues (with the same multiplicity) $\{\alpha_i\}_{i=1}^r$ for $r \leq \min(\mathrm{Dim}(\mathcal{H}_1), \mathrm{Dim}(\mathcal{H}_2))$. The Schmidt polar form of $|\Psi\rangle$ is described in equation 17 and is such that vectors in $\{\mathbf{e}_i\}_{i=1}^r$ and vectors in $\{\mathbf{f}_i\}_{i=1}^r$ are orthogonal. This is why we call this a decomposition as a sum of *bi-orthogonal* terms.

6.5 Purifying a Quantum Protocol

The main idea behind Mayers' proof is that purifications can be applied not only to the cases we have seen previously but to any sequence of instructions that might occur in a protocol. One party can, without having any chance of being caught, execute his part of the protocol at the quantum level, meaning that every action is purified.

Let us first review what set of instructions one party involved in a quantum protocol should be able to perform. The instructions should be enough to execute what we considered intuitively a quantum protocol (that is basically a pair of algorithms usually not quantum connected by a classical and a quantum channels). The algorithm of party $\mathcal{P}$ defines, at each step, a transition function from the actual view $\mathcal{V} \in \mathbb{V}$ to the new view $\mathcal{V}' \in \mathbb{V}$ for an arbitrary set of possible views $\mathbb{V}$. The view $\mathcal{V}$ can be seen as the memory a player needs in order to complete the execution of the protocol. The following describes what $\mathcal{P}$ should be able to execute at step $h > 0$ given the view $\mathcal{V}_{h-1}$ after step $h-1$ (i.e. $\mathcal{V}_0$ is the initial secret input if needed):

1. Picks a random bit r such that $\mathrm{P}(r = 1) = p$ and sets $\mathcal{V}_h = \mathcal{V}_{h-1} \cup \{(h, r)\}$,
2. Computes a function $f : \mathbb{V} \to \mathbb{V}$ and sets $\mathcal{V}_h = \mathcal{V}_{h-1} \cup \{(h, f(\mathcal{V}))\}$,
3. Announces, through the classical channel, the value $v \in \{0, 1\}$ of some memory register and sets $\mathcal{V}_h = \mathcal{V}_{h-1} \cup \{(h, v)\}$,
4. Sends a qubit in state depending on the view $\mathcal{V}$ through the quantum channel,
5. Stores in memory a classical bit received through the classical channel,
6. Measures a qubit received through the quantum channel using measurement M chosen according the view $\mathcal{V}$. The outcome O_M is added to the actual view $\mathcal{V}_h = \mathcal{V}_{h-1} \cup \{(h, O_M)\}$ (note that not measuring the received qubit is also covered by this case since it is equivalent to apply measurement $\mathbb{1}$).

Intuitively, if one party $\mathcal{P}$ can execute all these instructions then $\mathcal{P}$ can execute any quantum protocol. As we have seen in sections 6.1,6.2, and 6.3, most of the above instructions can be purified if they are considered isolated. The only missing piece is how to compose them in a such a way that the properties of purification remain. Suppose $\mathcal{P}$ has a quantum memory $\mathtt{QM} \in \mathcal{H}$ where $\mathcal{H}$ is large enough for storing all possible states in $\mathbb{V}$. Suppose that initially $\mathcal{P}$'s quantum memory $\mathtt{QM} \in \mathcal{H}$ is in state $|\mathtt{QM}_0\rangle$ where $\mathtt{QM}_0$ is state V_0 encoded in quantum registers. During the course of actions, $\mathtt{QM}$ will evolve to a quantum mixture since mixed states will be received through the quantum channel and entangled registers will be sent. We denote by $\rho_{\mathtt{QM}}(h)$ the mixed state of $\mathtt{QM}$ after step $h > 0$. $\mathcal{P}$ purifies each of the above instructions as follows:

1. $\mathcal{P}$ prepares a new quantum register in state $|\Psi_{\mathtt{C}(p)}\rangle$. The quantum memory is now in state $\rho_{\mathtt{QM}}(h) = \rho_{\mathtt{QM}}(h-1) \otimes |\Psi_{\mathtt{C}(p)}\rangle$.
2. Let $U_f \in \mathcal{H}$ be an unitary transformation implementing f. It might be the case that $\mathcal{P}$ has to append few quantum registers in some pure state $|\phi\rangle$ in order to satisfy the requirement that U_f is unitary. The new state of $\mathtt{QM}$ is $\rho_{\mathtt{QM}}(h) = U_f(\rho_{\mathtt{QM}}(h-1) \otimes |\phi\rangle)$.

3. $\mathcal{P}$ applies the standard measurement $\mathbb{M}_+$ on the quantum register $R_{\mathbf{v}}$ containing $\mathbf{v}$. He announces 0 if the outcome is $\mathbb{P}_0$ and announces 1 if the outcome is $\mathbb{P}_{\frac{\pi}{2}}$. The new state $\rho_{\mathtt{QM}}(h)$ for $\mathtt{QM}$ can be computed in terms of $\rho_{\mathtt{QM}}(h-1)$ as described in equation 7.
4. $\mathcal{P}$ simply sends away the quantum register containing the qubit to be sent. This operation mixes the state of $\mathtt{QM}$. The new state $\rho_{\mathtt{QM}}(h)$ is $\rho_{\mathtt{QM}}(h-1)$ without register $R_{\mathbf{v}}$ (formally speaking $\rho_{\mathtt{QM}}(h)$ is the partial trace of $\rho_{\mathtt{QM}}(h-1)$ with respect to register $R_{\mathbf{v}}$). The state of the qubit can be determined by a sequence of coin tosses previously generated and other quantum registers. The purification is performed by an easy generalization of the method described in section 6.1.
5. $\mathcal{P}$ adds a new register in state $|b\rangle$ to $\mathtt{QM}$ where $b \in \{0,1\}$ is the bit received through the classical channel. The new state is $\rho_{\mathtt{QM}}(h) = \rho_{\mathtt{QM}}(h-1) \otimes |b\rangle$.
6. In this case, $\mathcal{P}$ does not store the outcome but all possible outcomes of all possible measurements as we have seen in section 6.3. It is always possible to determine an unitary transformation U_M which applies each measurement specified by the state of some registers in $\mathtt{QM}$. This is because the set of registers involved in the choice of the measurement behaves like a set of quantum coin tosses.

Suppose a protocol performed between $\mathcal{P}$ and $\mathcal{P}'$ has the property that the final view of $\mathcal{P}'$ corresponds to the mixed state $\rho' \in \mathcal{H}$. If $\mathcal{P}$ purifies each step then the state of the system Ψ that contains $\mathcal{P}$'s quantum memory $\mathtt{QM}$ plus all what $\mathcal{P}'$ has generated and received during the execution, is in pure state $|\Psi\rangle \in \mathcal{H} \otimes \mathcal{H}'$ where $\mathcal{H}'$ is the Hilbert space for $\mathcal{P}'$'s part of the system. Moreover, since $\mathcal{P}$'s behaviour is indistinguishable from the non-purified execution of the protocol (that is the main property of the purification process) we have that $|\Psi\rangle$ is a purification of ρ'.

To get to know more about how to purify a quantum protocol, consult [33] and [34].

6.6 Quantum Bit Commitment Is Impossible

We are now ready to conclude the impossibility of quantum bit commitment. Suppose $\mathtt{BC}$ is a candidate for a secure quantum bit commitment scheme between Alice, the sender, and Bob, the receiver. A secure protocol for bit commitment must be

Concealing: Let $\rho_{\mathtt{BC}}(0) \in \mathcal{H}'$ and $\rho_{\mathtt{BC}}(1) \in \mathcal{H}'$ be the density operator corresponding to the mixed state received by Bob when Alice commits 0 and 1 respectively. In order for the commitment to be concealing, it must be the case that $\rho_{\mathtt{BC}}(0) \approx \rho_{\mathtt{BC}}(1)$.

Binding: Once the committing phase completed, Alice can open with success only one bit b.

We show that if the concealing condition holds then necessarily the binding condition does not. First, if $\rho_{\mathtt{BC}}(0)$ and $\rho_{\mathtt{BC}}(1)$ are sensibly different then they can

be distinguished with good probability by a quantum measurement. For more information about how distinguishable are different density operators, consult [21]. In the following, we assume that $\rho_{\texttt{BC}}(0) = \rho_{\texttt{BC}}(1)$ instead of being approximatively the same. To see how to address the case where $\rho_{\texttt{BC}}(0)$ and $\rho_{\texttt{BC}}(1)$ are *close* but not identical, consult [32]. Alice's attack, that is described next, is the same in both cases.

Assume that Alice purifies the commitment of $b = 0$ using the technique describes in the last section. The resulting quantum system $\Psi_0 \in \mathcal{H} \otimes \mathcal{H}'$ that contains Alice's QM and what has been generated and received by Bob, is a purification of $\rho_{\texttt{BC}}(0)$. At revelation, Alice can open $b = 0$ since all information that was needed in order to commit honestly to $b = 0$, is still accessible in QM. After the revelation phase, Bob accepts the opening of $b = 0$ exactly as it is in the honest case (this is what purification is all about).

Alice could have purified the commitment of $b = 1$ instead. This would result in a quantum purification $\Psi_1 \in \mathcal{H} \otimes \mathcal{H}'$ for the mixed state $\rho_{\texttt{BC}}(1)$ corresponding to the commitment of $b = 1$. When Ψ_1 is created, QM contains all the necessary information to open $b = 1$. Since $\rho_{\texttt{BC}}(1) = \rho_{\texttt{BC}}(0)$ it follows that $|\Psi_1\rangle$ is a purification of $\rho_{\texttt{BC}}(0)$ as well.

Assume Alice wants to open $b = 1$. We now take full advantage of the purification of $\rho_{\texttt{BC}}(0)$. In last section, we have seen that for any pair of purifications Ψ_0 and Ψ_1 for the same density operator $\rho_{\texttt{BC}}(0)$ there exists an unitary transformation $W \in \mathcal{H}$ such that

$$|\Psi_1\rangle = (W \otimes \mathbb{1}_{\mathcal{H}'})|\Psi_0\rangle.$$

Moreover, the transformation W depends only upon the protocol specification and is independent on what Bob does. Alice can therefore open $b = 1$ after having applied W on her part of the system (i.e. which is QM) just by following the revelation protocol honestly.

In conclusion, here is the always successful attack against the quantum bit commitment scheme BC:

1. Alice purifies the commitment of $b = 0$,
2. If Alice wants to open $b = 0$, she executes the revelation protocol from her part of the purification stored in QM,
3. If Alice wants to open $b = 1$, she applies W on QM and follows the revelation protocol for $b = 1$.

This strategy is indistinguishable from the honest one and therefore can be applied to any candidate for a quantum bit commitment scheme. We conclude that no quantum bit commitment exists.

7 Conclusion

It is now clear that in quantum cryptography, security in two-party games is much more difficult to achieve than the security of Alice and Bob against the

world. Security against the world is what is needed in order to achieve secret-key distribution and that, quantum cryptography can do for free. However, two-party games involve two parties that, although collaborative, do not trust the integrity of the other. In this model, we discussed the fact that quantum oblivious transfer is reducible to bit commitment which is not known/expected to be true in the classical world. We have also seen that the security conditions for bit commitment cannot be met by any purely quantum process. After Mayers' had shown that no quantum bit commitment exists, the spontaneous attitude was to try taking advantage of subtle assumptions appearing in the theorem statement. Most of those approaches use classical assumptions that have to hold only temporarily. The goal being to build from such assumptions a commitment scheme that is both concealing and binding even after the assumption is withdrawn. Unfortunately, none of these attempts provided more than what classical cryptography alone provides [13]. Mayers' attack is now known to apply in scenarios lying beyond the original statement of the no-go theorem. It can also be shown that perfect quantum coin tossing is also impossible [28]. However, quantum bit commitment is possible under physical (not computational assumptions). In [40], it has been shown that if one party is restricted to perform a subset of all possible generalized quantum measurements then quantum bit commitment is possible. The subset of possible measurements can be chosen in such a way that the assumption is likely to hold in any practical situation that will occur in a foreseeable future. In other words, the existence of an unitary process that breaks a quantum protocol does not necessarily imply that it can be implemented in real life. There is an inherent asymmetry between the complexity of physical processes involved in the execution of quantum protocols and those involved in quantum algorithms breaking them. It is not clear if Mayers' attack will be implementable in real life for all practical quantum bit commitment protocols. It would be interesting to characterize the *physical complexity* of the attack against protocols designed to make it difficult to implement.

Although they aim at solving the same kind of problems, the structure of quantum and classical cryptography differ. In a particular situation, one may offer advantages over the other. One thing we did not talked about yet is the possibility to use hybrid systems. Quantum encoding of information, like the BB84 coding scheme, allows to send classical information in an oblivious way. The receiver does not know for sure what was the original classical bit, and the sender does not know whether or not the receiver got the bit sent. But the sender, by announcing the transmission basis θ, allows the receiver to determine whether or not he received the bit perfectly. This simple primitive, although not powerful enough to provide bit commitment, cannot be done classically using no assumptions. It would be interesting to see if it can be used in a purely classical setting in order to weakened the classical assumptions required for a particular task. We have already seen that it is the case for oblivious transfer based on bit commitment; what about other cases?

In conclusion, quantum information is more elusive than its classical counterpart. One must always take care when analyzing and reasoning about quantum

protocols. Although the Holy Grail is not achievable quantumly (nor classically), quantum cryptography offers a good alternative to classical cryptography. Quantum cryptography provides an independent framework to complexity-based cryptography and several open questions remain in order to get a better understanding of its possibilities and limits.

8 Acknowledgements

I thank Jan Camenisch, Ivan Damgård, and Stefan Dziembowski for comments on earlier drafts. I would like to express my gratitude to the organizing committee and, in particular, to Ivan Damgård for having brought to life such a successful event.

References

1. Bennett, C. H. and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing", *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, Bangalore, India, December 1984, pp. 175 – 179. 183, 184, 197, 199, 203
2. Bennett, C. H., F. Bessette, G. Brassard, L. Salvail and J. Smolin, "Experimental quantum cryptography", *Journal of Cryptology*, Vol. 5, no. 1, 1992, pp. 3 – 28. 184, 199
3. Bennett, C. H., G. Brassard, S. Breidbart and S. Wiesner, "Quantum cryptography, or unforgeable subway tokens", *Advances in Cryptology: Proceedings of Crypto 82*, August 1982, Plenum Press, pp. 267 – 275.
4. Bennett, C. H., G. Brassard, C. Crépeau and U. Maurer,"Generalized Privacy Amplification",IEEE Transaction on Information Theory, vol. 41, no. 6, november 1995, pp. 1915–1923.
5. Bennett, C. H., G. Brassard, C. Crépeau and M.–H. Skubiszewska, "Practical quantum oblivious transfer", Advances in Cryptology, Lecture Notes in Computer Sciences, Springer-Verlag, vol. 576, 1991, pp. 351–366. *Advances in Cryptology — Proceedings of Crypto '91*, August 1991, Springer – Verlag, pp. 351 – 366. 186, 200
6. Biham, E, G. Brassard,M. Boyer,J. van de Graaf and T. Mor, "Security of Quantum Key Distribution Against All Collective Attacks", Los Alamos preprint archive `quant-ph/9801022`, January 1998. 184, 203
7. Blum, M.,"Coin Tossing by Telephone: A Protocol for Solving Impossible Problems", Proceeding of the 24th IEEE Computer Conference, 1982, pp. 133–137; reprint in SIGACT News, vol.15, no.1, 1983,pp.23–27. 184
8. Buttler, W.T., R.J. Hughes, P.G. Kwiat, G.G. Luther, G.L. Morgan, J.E. Nordholt, C.G. Peterson and C.M. Simmons, "Free-space quantum key distribution",available at `http://xxx.lanl.gov/ps/quant-ph/9801006`, january 1998.
9. Brassard, G., "Recent developments in quantum cryptography", *Proceedings of Pragocrypt '96: 1st International Conference on the Theory and Applications of Cryptology*, Prague, October 1996.
10. Brassard, G., D. Chaum and C. Crépeau, "Minimum Disclosure Proofs of Knowledge", Journal of Computing and System Science, vol.37 , 1988, pp. 156–189. 185

11. Brassard, G., C. Crépeau, "Quantum bit commitment and coin tossing protocols", *Advances in Cryptology — Proceedings of Crypto '90*, August 1990, Springer – Verlag, pp. 49 – 61. 203
12. Brassard, G., C. Crépeau, R. Jozsa and D. Langlois, "A quantum bit commitment scheme provably unbreakable by both parties", *Proceedings of 34th Annual IEEE Symposium on the Foundations of Computer Science*, November 1993, pp. 362 – 371. 186, 199, 203
13. Brassard, G., C. Crépeau, D. Mayers and L. Salvail, "Defeating Classical Bit Commitments with a Quantum Computer", Los Alamos preprint archive `quant-ph/9806031`, June 1998. 213
14. Carter, J.L. and M.N. Wegman,"Universal Class of Hash Functions", Journal of Computer and System Sciences, vol. 18, 1979, pp.143–154.
15. Crépeau, C., "Quantum oblivious transfer", *Journal of Modern Optics*, Vol. 41, no. 12, December 1994, pp. 2445 – 2454. 186, 202
16. Crépeau, C. and Kilian, J., "Achieving oblivious transfer using weakened security assumptions", *Proceedings of 29th Annual IEEE Symposium on Foundations of Computer Science*, 1988, pp. 42 – 52. 186, 196
17. Crépeau, C. and L. Salvail, "Quantum oblivious mutual identification", Advances in Cryptology, proceedings of EUROCRYPT'95, Lecture Notes in Computer Sciences, Springer-Verlag, vol. 921, 1995, pp.133–146. 199
18. Crépeau, C., J. van de Graaf, and A. Tapp, "Committed Oblivious Transfer and Private Multi-Party Computation", Adavances in Cryptology, proceedings of Crypto'95, Lecture Notes in Computer Sciences, Springer-Verlag, Vol. 963, 1995, pp. 110 – 123. 186
19. Diffie, W. and M.E., Hellman,"New directions in cryptography", IEEE Transactions on Information Theory, vol. IT-22, 1976, pp. 644–654. 184
20. Goldreich, O., S. Micali, and A. Wigderson, "Proofs That Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems", Journal Assoc. Comput. Mach., vol. 38, 1991, pp. 691–729. 185
21. Fuchs, C.A. and J. van de Graaf,"Cryptographic Distinguishability Measures for Quantum Mechanical States", Los Alamos preprint archive `quant-ph/9712042`, December 1997. 212
22. Hughston, L. P., R. Jozsa,and W. K. Wootters, "A complete classification of quantum ensembles having a given density matrix", *Physics Letters A*, vol. 183, pp. 14 – 18, 1993. 208, 209
23. Impagliazzo, R and S. Rudich, "Limits on Provable Consequences of One-Way Permutations", in the 24th ACM conference on the theory of computing, 1989. 185, 203
24. Jacobs, B.C. and J.D. Franson,"Quantum cryptography in free space", Optics Letters, vol. 21, no. 22, November 15, 1996. 184
25. Jozsa, J., "Fidelity for mixed quantum states", *Journal of Modern Optics*, vol. 41(12), pp. 2315 – 2323, 1994.
26. Kilian, J.,"Founding Cryptography on Oblivious Transfer", Proceedings of the 20th Annual ACM Symposium on the Theory of Computing, Chicago, 1988, pp. 20 – 31. 186
27. Lo, H.–K. and H. F. Chau, "Is quantum bit commitment really possible?", *Physical Review Letters*, vol 78, pp. 3410 – 3413 (1997). 186, 203
28. Lo, H.–K. and H. F. Chau, "Why quantum bit commitment and ideal quantum coin tossing are impossible." Available at `http://xxx.lanl.gov/ps/quant-ph/9711065`. 213

29. Lo, H.–K. and H. F. Chau,"Security of Quantum Key Distribution", available at `http://xx.lanl.gov/list/quant-ph/9803006`, March 1998.
30. Maurer, U.M.,"Protocols for Secret Key Agreement by Public Discussion Based on Common Information",Advances in Cryptology, proceedings of CRYPTO'92, Lecture Notes in Computer Sciences vol. 740, Springer-Verlag, 1993, pp. 461–470. 196
31. Mayers, D., On the security of the quantum oblivious transfer and key distribution protocols, Advances in Cryptology, proceedings of Crypto'95, Lecture Notes in Computer Sciences, Springer-Verlag, vol. 963, 1995, pp.124–135. 203
32. Mayers, D., "The trouble with quantum bit commitment", LANL Report No. quant-ph/9603015 (to be published). The author first discussed the result in Montréal at a workshop on quantum information theory held in October 1995. 186, 203, 212
33. Mayers, D., "Unconditionally secure quantum bit commitment is impossible", submitted to *Fourth Workshop on Physics and Computation — PhysComp '96*, Boston, November 1996. 186, 203, 211, 216
34. Mayers, D., "Unconditionally secure quantum bit commitment is impossible", *Physical Review Letters*, vol 78, pp. 3414 – 3417 (1997). Note that this paper has the same title as [33] even though it uses a different approach. 186, 203, 211
35. Mayers, D.,"Unconditional security in Quantum Cryptography", available at `http://xx.lanl.gov/list/quant-ph/9802025`, february 1998. 184, 203
36. Mayers, D. and L. Salvail, "Quantum oblivious transfer is secure against all individual measurements", *Proceedings of the Third Workshop on Physics and Computation — PhysComp '94*, Dallas, November 1994, IEEE Computer Society Press, pp. 69 – 77. 186, 203
37. Muller, A., T. Herzog, B. Huttner, W. Tittel, H. Zbinden and N. Gisin, "Plug and Play systems for quantum cryptography", available at `http://xx.lanl.gov/list/quant-ph/9611042`, november 1996. 184
38. Peres, P., "Quantum Theory:Concepts and Methods", Fundamental Theories of Physics seriesm, Kluwer Academic Publishers,vol. 72, 1995. 192, 209
39. Rivest, R.L., A., Shamir and L.M, Adleman,"A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, vol.21, 1978,pp.120–126. 184
40. Salvail, L., "Quantum Bit Commitment from a Physical Assumption", Advances in Cryptology, proceedings of CRYPTO'98, Lecture Notes in Computer Sciences, Springer-Verlag, vol. 1462, 1998, pp. 338–353. 187, 213
41. Shor, P.,"Algorithms for Quantum Computation:Discrete Logarithms and Factoring", Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, 1994, pp. 124–134. 184
42. Shannon, C.E.,"A Mathematical Theory of Communication", Bell System Technical Journal, vol.27, 1948,pp.379–423 and pp.623–656. 184
43. Townsend, P., J. Rarity and P. Tapster,"Single photon interference in a 10km long optical fibre interferometer", electronic Letters, 29, 1993, pp.634–635. 184
44. Wiesner, S., "Conjugate coding", *Sigact News*, Vol. 15, no. 1, 1983, pp. 78 – 88; original manuscript written *circa* 1969.
45. Wyner, A.D.,"The wire-tap channel", Bell System Technical Journal,vol. 54, no. 8, 1975, pp. 1355–1387. 196
46. Yao, A. C.–C., "Security of quantum protocols against coherent measurements", Proceedings of the 26th Annual ACM Symposium on the Theory of Computing, 1995, pp. 67 – 75. 186, 203

Unconditional Security in Cryptography

Stefan Wolf

Computer Science Department
Swiss Federal Institute of Technology (ETH Zürich)
CH-8092 Zürich, Switzerland
wolf@inf.ethz.ch

Abstract. The fact that most presently-used cryptosystems cannot be rigorously proven secure and hence permanently face the risk of being broken motivates the search for schemes with unconditional security. The corresponding proofs however must be based on information theory rather than complexity theory. One reason for this is the lack of known lower bounds on the running time of algorithms solving certain computational problems such as the discrete-logarithm problem or the integer-factoring problem. At the beginning of an information-theoretic analysis of cryptosystems stands Shannon's definition of perfect secrecy, unquestionably the strongest possible security definition, and his well-known inequality giving a lower bound on the key length of every perfectly secret cipher, thus suggesting that such a high level of confidentiality cannot be realized in any practical scheme. This pessimism has later been qualified by several authors who showed that unconditional security can be achieved in many special but realistic scenarios. Some of these approaches are described in this introductory overview article.

1 Computational versus Information-Theoretic Security

The security of many presently-used cryptosystems, e.g., of all public-key cryptographic schemes, is based on the assumed hardness of computational problems in number theory such as the integer-factoring problem (e.g., RSA [28]) or the problem of computing discrete logarithms in certain finite cyclic groups (e.g., Diffie-Hellman [13]). Such a cryptosystem is called *computationally secure*.

Up to date, no practical cipher has been proven computationally secure. Note first of all that it is an inherent fact that computational security can only hold under certain assumptions on the adversary's computer resources. In other words, a computationally infinitely powerful opponent can break every system of this type by exhaustive search over the key space.

One reason for the lack of proofs of cryptographic security is that in complexity theory, actually proved lower bounds on the running time of algorithms solving specific problems are either rather weak (and useless in cryptography) or valid only in special computational models (e.g., [32]). Unfortunately, such bounds are not directly useful neither since it can never be guaranteed that the adversary is restricted to this particular model. So-called "provable computational security" is always conditional and means that an efficient reduction

I. Damgård (Ed.): Lectures on Data Security, LNCS 1561, pp. 217–250, 1999.

from a well-known problem that is believed to be hard, such as the discrete-logarithm problem or the decisional Diffie-Hellman problem, to breaking the proposed system can be given, thus showing that the cryptosystem is secure if some widely-accepted standard complexity assumption is true (e.g., [11]).

Finally, it has been shown that the integer-factoring as well as the discrete-logarithm problem can be solved in polynomial-time by a *quantum computer*, i.e., a computing device that is able to exploit certain effects from quantum mechanics [31]. The security of most public-key cryptographic protocols is based on the hardness of at least one of these problems.

Consequently, practical computational security is always conditional and additionally faces the risk of being broken by progress in the theory of efficient algorithms or in hardware engineering. On the other hand it appears desirable from both a scientific and practical point of view to design cryptosystems whose security is not based on any assumptions and can be proven rigorously. Because of the reasons discussed above, such security proofs must be based on *information theory* (i.e., probability theory) rather than complexity theory. There have been made various attempts at realizing this type of security, some of which we describe in this overview paper.

The outline of the article is as follows. We start with an introduction to some basic definitions and facts from probability and information theory (Section 2). Then, a definition of perfect secrecy, undoubtedly the strongest possible security definition in cryptography, is given (Section 3). Shannon's pessimistic theorem suggests that perfect secrecy is necessarily impractical. However, we describe a number of approaches that could qualify this pessimism. All these constructions have in common that some kind of limitations are needed on the amount of information that an opponent obtains. Realistic scenarios have been described where such an upper bound on the adversary's knowledge can for instance be based on noise, an inherent property of every physical communication channel (Section 4). Motivated by these examples, a model has been presented and analyzed that shows how two parties can generate a secret key from common randomness by communication over an insecure but authentic (or even completely insecure) channel (Sections 5 and 6).

2 Basic Concepts of Information Theory

Information theory goes back to Claude Shannon and his celebrated 1948 paper [30]. Examples of good and detailed introductions into the field are [10] or [5].

2.1 Probability-Theoretic Preliminaries

In this section we introduce some basic probability-theoretic concepts. For a detailed introduction see for example [14].

Let $\mathcal{X}$ be a countable set. The *distribution* P_X of a *discrete random variable* X with *range* $\mathcal{X}$ is a mapping

$$P_X : \mathcal{X} \longrightarrow \mathbf{R}_{\geq 0}$$

with $\sum_{x\in\mathcal{X}} P_X(x) = 1$. If $\mathcal{X} \subset \mathbf{R}$, the expectation of X is defined as

$$\mathrm{E}[X] := \sum_{x\in\mathcal{X}} x \cdot P_X(x) \ .$$

Let f be a convex function. Then we have

$$\mathrm{E}[f(X)] \geq f(\mathrm{E}[X]) \ . \tag{1}$$

Inequality (1) is called *Jensen's inequality*. Most of the basic inequalities in information theory follow directly from this inequality.

The *joint distribution* $P_{X_1X_2\cdots X_N}$ of N random variables is a probability distribution over the set $\mathcal{X}_1\times\mathcal{X}_2\times\cdots\times\mathcal{X}_N$. The random variables $X_1, X_2, \ldots, X_N$ are called *statistically independent* if

$$P_{X_1X_2\cdots X_N}(x_1, x_2, \ldots, x_N) = P_{X_1}(x_1) \cdot P_{X_2}(x_2) \cdots P_{X_N}(x_N)$$

for all $x_1, x_2, \ldots, x_N$, i.e., when the joint distribution equals the product of the *marginal distributions*.

An event $\mathcal{A}$ is a subset of the range of a random experiment. By $\mathrm{Prob}[\mathcal{A}]$ we denote the probability of $\mathcal{A}$, i.e., the sum of the probabilities of all the outcomes belonging to $\mathcal{A}$. The *conditional distribution* of X, given that the event $\mathcal{A}$ (with $\mathrm{Prob}[\mathcal{A}] > 0$) occurs, is defined as

$$P_{X|\mathcal{A}}(x) := \frac{\mathrm{Prob}[\{X = x\} \cap \mathcal{A}]}{\mathrm{Prob}[\mathcal{A}]} \ .$$

As a special case, a random variable can be conditioned on the event

$$\mathcal{A} := \{Y = y\}$$

that another random variable Y takes a particular value y. The resulting distribution

$$P_{X|Y}(x, y) := P_{X|Y=y}(x)$$

is called the *conditional distribution of X given Y*. Note that the function $P_{X|Y}(\cdot\,,\cdot)$ with two arguments is *not* a probability distribution on $\mathcal{X}\times\mathcal{Y}$, but for every $y \in \mathcal{Y}$, the function $P_{X|Y}(\cdot\,, y)$ is a distribution on $\mathcal{X}$.

2.2 Bar Kochba, Uncertainty, and Entropy

The following story has been reported about *Bar Kochba* (the "Son of the Star"), leader of the Jews during their independence war in 135 B.C., who defended his fortress heroically against a superior number of Romans [27].

"It is also said that Bar Kochba sent out a scout to the Roman camp who was captured and tortured, having his tongue cut out. He escaped from captivity and reported back to Bar Kochba, but being unable to talk, he could not tell in words what he had seen. Bar Kochba accordingly asked him questions which he could answer by nodding or shaking his head. Thus he acquired from his mute scout

the information he needed to defend the fortress. [...] It occurred to me that, if the story of Bar Kochba were true, then he would have been the forefather of information theory".

In the so-called *Bar-Kochba game*, one player has to find out, by asking yes/no-questions, what the second player has in mind. This game was extremely popular among writers in Budapest at the beginning of this century. Regardless of the (possibly adaptive) strategy of the questioner he cannot, with at most 20 questions, distinguish between more than 2^{20}, i.e., about one million, different objects (because there are only 2^{20} ways of answering the 20 questions differently). On the other hand, given that the object to be found comes from a set of size at most n, then $\lceil \log_2 n \rceil$ questions are always sufficient if the following strategy is used. Let a fixed encoding of all the objects by binary strings of length 20 be defined. Then, the strategy is to ask whether the first, second, ... bit of the encoding is 1.

This example shows the close relationship between the Bar-Kochba game and binary coding. For a random variable X that takes one of $n = 2^k$ values with equal probabilities, the minimal average number of questions in the Bar-Kochba game, as well as the minimal average codeword length of a prefix-free binary code, is k. Note that this bound cannot be beaten even if a strategy is used with variable codeword lengths for the different outcomes. We call this quantity the uncertainty or *entropy* of X, denoted by $H(X)$.

If the size of the range $\mathcal{X}$ of X is not a power of 2, then the average number of questions required obviously lies between $\lfloor \log_2 |\mathcal{X}| \rfloor$ and $\lceil \log_2 |\mathcal{X}| \rceil$. When combining r independent realizations of the random variable X, the optimal average number of questions required to learn all the outcomes together lies between $\lfloor \log_2 |\mathcal{X}|^r \rfloor$ and $\lceil \log_2 |\mathcal{X}|^r \rceil$. Taking such combinations into account, we obtain for the entropy of X that

$$\log_2 |\mathcal{X}| - \frac{1}{r} < \frac{\lfloor \log_2 |\mathcal{X}|^r \rfloor}{r} \leq H(X) \leq \frac{\lceil \log_2 |\mathcal{X}|^r \rceil}{r} < \log_2 |\mathcal{X}| + \frac{1}{r}$$

for all $r \geq 1$, hence

$$H(X) = \log_2 |\mathcal{X}| \ . \tag{2}$$

Equation (2) is called *Hartley's formula* and gives the entropy of a uniformly distributed random variable.

We consider an example of a random variable Y that is *not* uniformly distributed. Let $\mathcal{Y} = \{a, b, c, d\}$, with $P_Y(a) = 1/2$, $P_Y(b) = 1/4$, $P_Y(c) = P_Y(d) = 1/8$. We conclude from the above that two questions are always sufficient, hence $H(Y) \leq 2$. However, there is a better strategy of asking questions or equivalently, a prefix-free code with a shorter average codeword length, namely,

$$a \rightsquigarrow 0 \ , \quad b \rightsquigarrow 10 \ , \quad c \rightsquigarrow 110 \ , \quad d \rightsquigarrow 111 \ .$$

The average number of questions required when asking the bits of the codewords is

$$\frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = \frac{7}{4} \ (< 2) \ .$$

On the other hand, this code (or strategy of asking questions) is optimal. Note that in this example, the length of the codeword of a letter is $\log_2(1/p)$, where p is the probability of this letter. The quantity $\log_2(1/p)$ is sometimes called the *unexpectedness* of a elementary event with probability p.

A code is optimal if the length of every codeword is equal to the unexpectedness of the corresponding outcome. Hence, for a random variable X for which each probability p_i is of the form $p_i = 2^{-s_i}$ for an integer s_i, we have

$$H(X) = p_1 \log_2(1/p_1) + p_2 \log_2(1/p_2) + \cdots \ . \tag{3}$$

Equation (3) is called *Shannon's formula*, and is a generalization of Hartley's formula (2). By combining independent realizations of the random variable for the encoding, one obtains that this formula gives the entropy of any discrete random variable. The following definition was given by Shannon in 1948.

Definition 1. [30] *The* entropy $H(X)$ *of a random variable* X *with distribution* P_X *is given by*

$$H(X) = H(P_X) := \sum_{x \in \mathcal{X}} -P_X(x) \cdot \log_2 P_X(x) = \mathrm{E}\left[-\log_2 P_X\right] \ .$$

∘

The *joint entropy* of random variables $X_1, X_2, \ldots, X_N$ is the entropy of the joint distribution, i.e.,

$$H(X_1 X_2 \cdots X_N) := H(P_{X_1 X_2 \cdots X_N}) \ .$$

Moreover, Definition 1 also covers the case where the distribution is conditioned on an event $\mathcal{A}$. We write $H(X|\mathcal{A}) := H(P_{X|\mathcal{A}})$ or, if $\mathcal{A} = \{Y = y\}$,

$$H(X|Y = y) := H(P_{X|Y=y}) \ .$$

The entropy of a *binary* random variable with probability distribution $[p, 1-p]$ is given by the *binary entropy function*

$$h(p) := -p \log_2 p - (1-p) \log_2 (1-p)$$

(see Figure 1).

The entropy of a random variable X is always non-negative and upper bounded by the binary logarithm of the cardinality of the range, i.e.,

$$0 \leq H(X) \leq \log_2 |\mathcal{X}| \ . \tag{4}$$

The second inequality, which is intuitively clear when taking into account the discussion above, follows from Jensen's inequality for *concave* functions:

$$H(X) = \mathrm{E}\left[\log_2(1/P_X)\right] \leq \log_2(\mathrm{E}\left[1/P_X\right]) = \log_2 |\mathcal{X}| \ .$$

Equality on the left hand side of (4) holds if and only if there exists an element $x_0 \in \mathcal{X}$ with $P_X(x_0) = 1$, whereas equality on the right hand side is equivalent to the fact that X is uniformly distributed over $\mathcal{X}$, i.e., that $P_X(x) = 1/|\mathcal{X}|$ holds

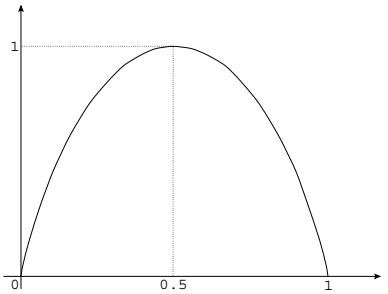


Fig. 1. The Binary Entropy Function

for all $x \in \mathcal{X}$. In the special case where the outcome of the random experiment is a binary string of length n, the second inequality of (4) implies that the entropy of the random variable can be equal to, but not exceed n.

For random variables X and Y, we have

$$H(XY) \leq H(X) + H(Y) \ , \tag{5}$$

with equality if and only if X and Y are statistically independent.

2.3 Conditional Entropy and Mutual Information

When considering inequality (5) it appears natural to interpret the (non-negative) quantity $H(XY) - H(X)$ as the entropy of the random variable Y when X is given.

Definition 2. The *conditional entropy of* Y *when given* X is defined as

$$H(Y|X) := H(XY) - H(X) \ . \tag{6}$$

∘

Note that in contrast to all previously introduced entropies such as $H(X) = H(P_X)$, $H(XY) = H(P_{XY})$, or $H(Y|X = x) = H(P_{Y|X=x})$, the conditional entropy $H(Y|X)$ is *not* the entropy of a specific probability distribution, but rather the expected value of the entropies $H(Y|X = x)$, i.e.,

$$H(Y|X) = \mathrm{E}_X[H(Y|X = x)] \ .$$

Equation (6) can be rewritten as

$$H(XY) = H(X) + H(Y|X) \ .$$

This *chain rule* can be generalized as follows. For random variables $X_1, \ldots, X_N$ and an event $\mathcal{A}$ we have

$$H(X_1X_2 \cdots X_N|\mathcal{A}) = H(X_1|\mathcal{A}) + H(X_2|X_1, \mathcal{A}) + \cdots + H(X_N|X_1X_2 \cdots X_{N-1}, \mathcal{A}).$$

It is a fundamental property of the conditional entropy that

$$H(Y|X) \leq H(Y) \ , \tag{7}$$

which is a consequence of inequality (5). (However, note that $H(Y|X = x) > H(Y)$ is possible, as the following example illustrates. Let Y be 100 independent flips of an unfair coin with Prob ["heads"] $= 99.9\%$, and let X be the number of "heads" in the sequence. Then, although of course $H(Y|X) < H(Y)$ holds, we have

$$1.141 \approx 100 \cdot h(0.999) = H(Y) < H(Y|X = 50) = \log_2 \left(\binom{100}{50} \right) \approx 96.35 \ .$$

Of course the event $\{X = 50\}$ is extremely unlikely.)

Informally spoken, inequality (7) can be interpreted as the fact that information can never increase uncertainty. More precisely, the quantity

$$I(Y;X) := H(Y) - H(Y|X) = H(X) + H(Y) - H(XY) \geq 0 \tag{8}$$

is the amount of information that X gives about Y. The last expression of (8) shows that $I(Y;X)$ is symmetric in its arguments, i.e., that

$$I(X;Y) = I(Y;X)$$

holds. The quantity $I(X;Y)$ is called *the mutual information between X and Y*. Analogously, one can define $I(X;Y|\mathcal{A}) := H(X|\mathcal{A}) - H(X|Y,\mathcal{A})$ and

$$I(X;Y|Z) := H(X|Z) - H(X|YZ) = \mathrm{E}_Z[I(X;Y|Z=z)] \ .$$

2.4 Graphical Representation of Information-Theoretic Quantities

Let X and Y be random variables. Then the quantities $H(XY)$, $H(X)$, $H(Y)$, $H(X|Y)$, $H(Y|X)$, and $I(X;Y)$ can be graphically represented as shown in Figure 2. The union of all inner regions corresponds to $H(XY)$. The representation

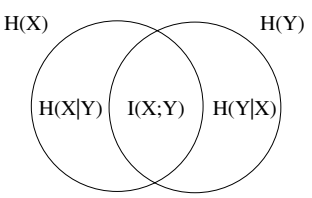


Fig. 2. Two Random Variables

has the property that the quantity corresponding to the disjoint union of some regions equals the sum of the quantities corresponding to these partial regions. For a detailed discussion of this measure-theoretic representation of information-theoretic quantities see [37].

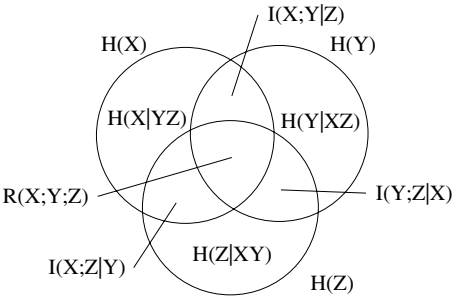


Fig. 3. Three Random Variables

The case of three random variables is shown in Figure 3. Note that the quantity corresponding to the region in the middle,

$$R(X;Y;Z) := I(X;Y) - I(X;Y|Z) \ ,$$

is symmetric in X, Y, and Z and can be negative. All the other regions represent information-theoretic quantities that are always non-negative.

Figure 4 illustrates independent symmetric bits X and Y and $Z := X \oplus Y$. Figure 5 shows a Markov chain.

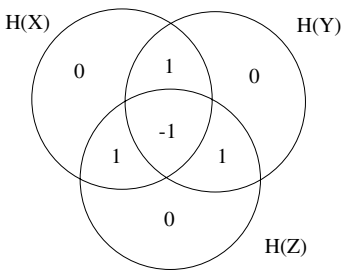


Fig. 4. $Z = X \oplus Y$

3 Perfect Secrecy and Shannon's Pessimistic Theorem

In the following we consider the problem of information-theoretically secure key generation and message transmission over an insecure channel. This section contains Shannon's definition of perfect secrecy of a cipher and his well-known theorem which appears to imply that unconditional security is necessarily completely impractical. In the following sections however it is demonstrated that information theory cannot be used only to prove such pessimistic results. It is somewhat surprising that when the models and security requirements are only

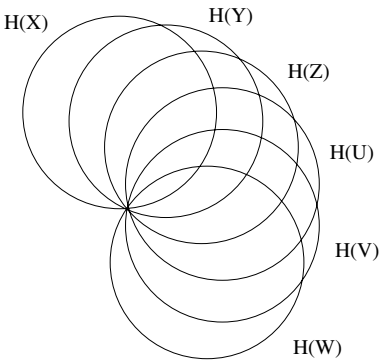


Fig. 5. A Markov Chain $X \to Y \to Z \to U \to V \to W$

slightly modified, then practical information-theoretic security can be achieved in many realistic scenarios.

Let us start with the classical scenario of a symmetric cryptosystem with message M, key K, and ciphertext C (see Figure 6). The following security

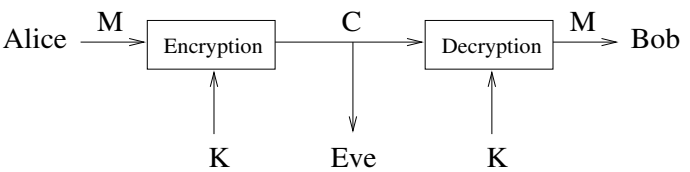


Fig. 6. A Symmetric Cryptosystem

definition appears to be the strongest possible for such a cryptosystem.

Definition 3. [29] *A cipher is called* perfectly secret *if the ciphertext reveals no information about the message, i.e., if* $I(M;C) = 0$ *holds.* ∘

Equivalent characterizations of this condition are that M and C are statistically independent, or that the best strategy of an eavesdropper who wants to obtain (information about) the message from the ciphertext is to use only the a priori knowledge about M and to discard C.

Perfect secrecy can even be achieved without any computation, as the example in Figure 7 shows. As everyone can easily see, the ciphertext alone reveals no information about the message at all in this example! (For more on "visual cryptography," see [26].)

This visual cipher is a graphical implementation of the *one-time pad* that was already proposed by Vernam in 1926 [34]. Here, the message is a string $M = [m_1, m_2, \ldots, m_N]$ of length N, and the key is a uniformly distributed N-bit string $K = [k_1, k_2, \ldots, k_N]$ which is independent of M. The ciphertext C is

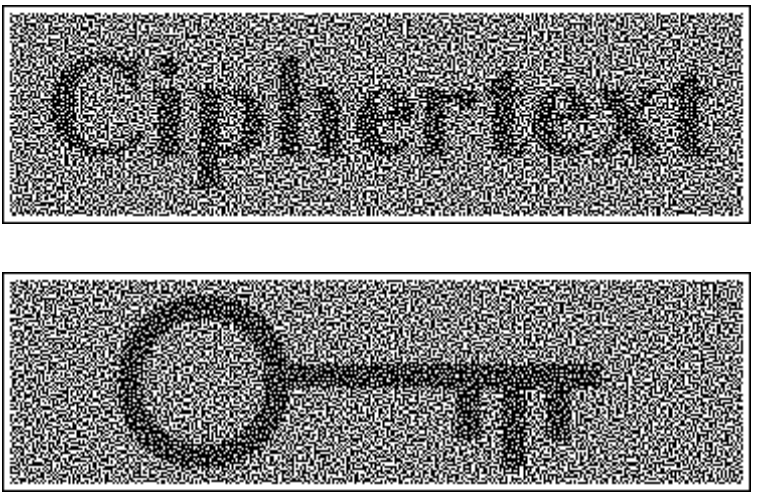


Fig. 7. Visual Decryption

computed from M and K by

$$C = [c_1, c_2, \ldots, c_N] = [m_1 \oplus k_1, m_2 \oplus k_2, \ldots, m_N \oplus k_N] =: M \oplus K \ .$$

The one-time pad is perfectly secret. To see this, observe first than when given the cleartext and the ciphertext, then the key is uniquely determined, i.e., $H(K|MC) = 0$. Furthermore, $I(K;C|M) = N$ (remember than N is the block length) follows then from $H(K) = N$ and $I(M;K) = 0$. Finally, $I(M;C) = 0$ holds because $H(C) \leq \log_2 |\mathcal{C}| = N$. A graphical representation of the quantities is given in Figure 8.

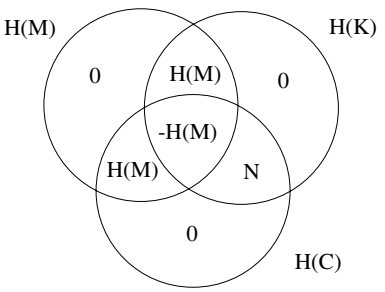


Fig. 8. Perfect Secrecy of the One-Time Pad

Unfortunately, the price one has to pay here for perfect secrecy is that the communicating parties must share a secret key which is at least as long as the

message (and can only be used once). In view of this property, the one-time pad appears to be quite impractical and can only offer an advantage in time: the key can be safely transmitted whenever this is possible, and the message can be secretly sent whenever this is needed.

However, Shannon showed that perfect secrecy cannot be obtained in a cheaper way, i.e., that the one-time pad is optimal with respect to key length.

Theorem 4. [29] *For every perfectly secret cryptosystem (with unique decodability), we have*

$$H(K) \geq H(M) \ .$$

For a proof of Shannon's theorem, note first that unique decodability means $H(M|CK) = 0$. The graphic representation of the involved quantities is given

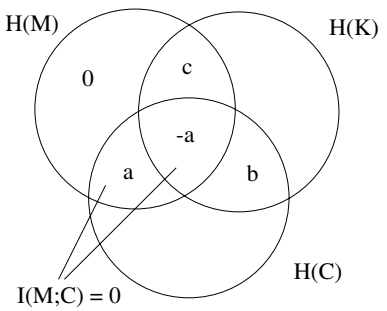


Fig. 9. The Proof of Shannon's Theorem

in Figure 9. We have $b \geq a$ because $I(C;K) \geq 0$, and

$$H(K) \geq b - a + c \geq a - a + c = H(M) \ .$$

This concludes the proof.

4 Optimistic Results by Limiting the Adversary's Information

Unfortunately, Shannon's theorem implies that perfect secrecy is possible only between parties who share a secret key of length at least equal to the entropy of the message to be transmitted. Hence every perfectly secret cipher is necessarily as impractical as the one-time pad. On the other hand, the assumption that the adversary has a *perfect* access to the ciphertext is overly pessimistic and unrealistic in general, since every transmission of a signal over a physical channel is subject to noise.

Motivated by this, many models have been presented and analyzed in which the information the adversary obtains is limited in some way, and which offer

the possibility of information-theoretically secure key agreement and, under the assumption that insecure channels are always available, secret message transmission (using the one-time pad with the generated secret key).

The condition that the opponent's knowledge is bounded can for instance be based on noise in communication channels [36],[12],[1],[21], on the fact that the adversary's memory is limited [22],[9], or on the uncertainty principle of quantum mechanics [2]. In this article, we describe a number of models that belong to the first category.

4.1 Wyner's Wire-Tap Channel

Consider the following (simple but generally unrealistic) situation first. Assume that two parties Alice and Bob are connected by an authentic and noiseless binary channel, and that a wiretapper Eve receives the bits sent over the channel with some error probability $\varepsilon > 0$. In other words, her wire-tap channel is a *binary symmetric channel (BSC)* with error probability ε (see Figure 10).

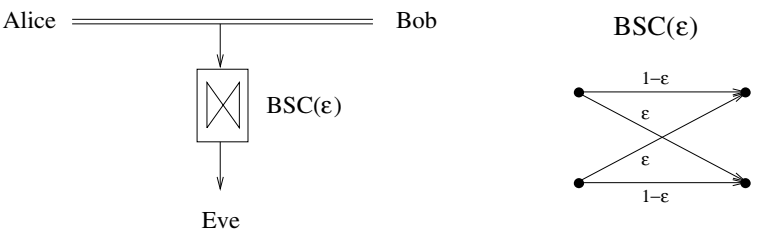


Fig. 10. A Binary-Symmetric Wire-Tap Scenario

In this situation, Alice can send a message bit M to Bob by sending an N-bit block $[X_1, X_2, \ldots, X_N]$, where $X_1, X_2, \ldots, X_{N-1}$ are independent and symmetric bits and X_N is such that

$$X_1 \oplus X_2 \oplus \cdots \oplus X_N = M \ .$$

Eve's error probability when guessing the bit M with the optimal strategy is

$$p = \frac{1 - (1 - 2\varepsilon)^N}{2} \ ,$$

and converges to $1/2$ exponentially fast in N. Moreover, the information that Eve obtains about M from the noisy versions of $X_1, X_2, \ldots, X_N$ does not exceed $1 - h(p)$. By repeating this process, Alice and Bob can agree on a highly secret key of arbitrary length.

The following, more general scenario of the *wire-tap channel* (see Figure 11) was introduced and analyzed by Wyner [36] and simplified by Massey [16]. In this setting, Alice and Bob are connected by a discrete memoryless channel

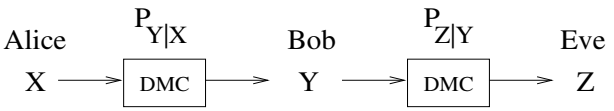


Fig. 11. Wyner's Wire-Tap Channel

(characterized by its conditional probability distribution $P_{Y|X}$), whereas Eve receives a noisy version Z of Bob's channel output Y. Alice chooses the input to the first channel according to some distribution P_X.

It was shown in [36] that in this scenario, Alice and Bob can agree on a highly secret key at a some rate in many situations (for instance in the case where all the random variables are binary and the channels are binary-symmetric with error probabilities not 1/2 and not 0 nor 1, respectively). Exact definitions of the security requirements to such a key, as well as of the secret-key generation rate, are given below.

However, the assumption that the adversary only receives a degraded version of the legitimate receiver's information is unrealistic in general. This fact motivated the study of generalizations of Wyner's model.

4.2 Broadcast Channels

Csiszár and Körner [12] considered the situation where the sender Alice is connected to the receiver Bob by a discrete memoryless channel (with conditional distribution $P_{Y|X}$), and where also the adversary Eve receives a noisy version Z of X over a different channel (characterized by $P_{Z|XY}$, i.e., the channels are not necessarily independent). As before, Alice chooses the channels' input X according to some distribution P_X. The broadcast scenario is illustrated in Figure 12.

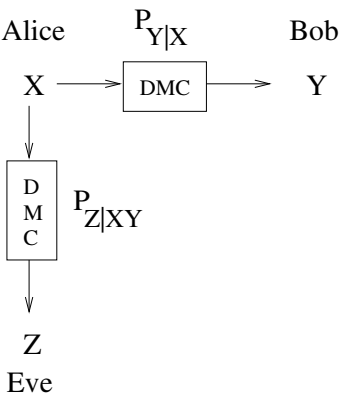


Fig. 12. The Broadcast-Channel Scenario

For this setting, the *secrecy capacity* $C_S(P_{YZ|X})$ has been defined as the maximal rate at which Alice and Bob can generate a virtually secret key. Without going into the details of the definitions and the key-generation protocols, we remark that both the size of the generated secret key as well as the amount of information leaked to the adversary are defined in terms of a *rate*, i.e., measured as average information *per channel use*.

In [12], the following lower bound on the secrecy capacity, depending on the conditional distribution $P_{YZ|X}$, has been proved:

$$C_S(P_{YZ|X}) \geq \max_{P_X} [I(X;Y) - I(X;Z)] \ . \tag{9}$$

In equality (9), the maximum is taken over all possible distributions P_X of X. Intuitively, this condition implies that if the legitimate partners initially have some advantage over Eve in terms of the information about each other's random variables, then this advantage can be fully exploited to generate a secret key.

However, if Alice and Bob have no such advantage to start with, then generally no secret-key agreement is possible in this model. Let us for instance consider the situation where the channels are independent and binary-symmetric with error probabilities ε and δ (see Figure 13). In this special scenario, the

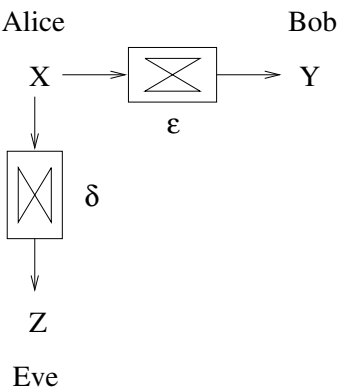


Fig. 13. Independent Binary-Symmetric Broadcast Channels

secrecy capacity is given by

$$C_S(\varepsilon, \delta) = \begin{cases} h(\delta) - h(\varepsilon) & \delta > \varepsilon \\ 0 & \text{otherwise} \ . \end{cases}$$

In other words, secret-key agreement is impossible unless Bob's channel is better than Eve's. Unfortunately, it may often be impossible to guarantee that the adversary's channel is noisier than the one of the legitimate partner.

4.3 The Power of Interaction

The following example, given in [21], illustrates how much more powerful interaction can be in contrast to one-way transmission for unconditionally secure key agreement. This is a motivation for the study of a more general model of secret-key agreement from common information by insecure two-way communication. We discuss this model in Section 5.

We start with the situation shown in Figure 13, where $0 < \delta \leq \varepsilon < 1/2$. As mentioned above, no secret-key agreement is possible. However, let us assume an *interactive* variant of this model with an additional noiseless and insecure but authentic channel. (Note that channels with virtually these properties often exist in reality, e.g., telephone lines.) Surprisingly, the situation is now entirely different although the additional channel can be perfectly overheard by Eve.

Observe first that the additional public-discussion channel allows to invert the direction of the noisy channel between Alice and Bob by the following trick. First, Alice chooses a random bit X and sends it over the noisy channel(s). This bit is received by Bob as Y and by Eve as Z. Bob, who wants to send the message bit C to Alice, computes $C \oplus Y$ and sends this over the noiseless public channel. Alice computes $(C \oplus Y) \oplus X$, whereas Eve can compute $(C \oplus Y) \oplus Z$. This perfectly corresponds to the situation where the direction of the main channel is inverted (see Figure 14).

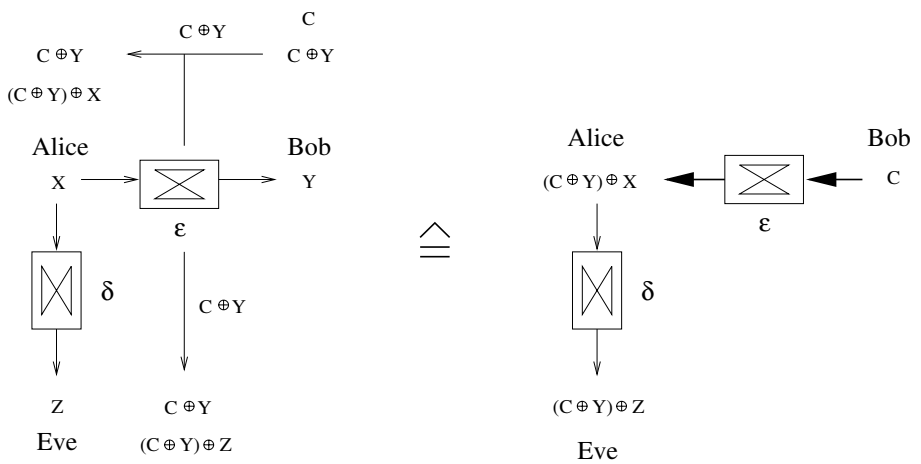


Fig. 14. Inverting the Main Channel

The second crucial observation is that this is exactly the binary-symmetric setting of Wyner's wire-tap channel of Section 4.1, allowing secret-key agreement at some rate. We conclude from this example that the possibility of feedback from Bob to Alice can substantially improve the legitimate partners' situation towards a wire-tapping adversary.

5 Interactive Secret-Key Agreement from Common Randomness

5.1 The Scenario and the Secret-Key Rate

Maurer has proposed the following interactive model of secret-key agreement by public discussion from common information [21]. The parties Alice and Bob who want to establish a mutual secret key have access to realizations of random variables X and Y, respectively, whereas the adversary knows a random variable Z. Let P_{XYZ} be the joint distribution of the random variables. Furthermore, the legitimate partners are connected by an insecure but authentic channel, i.e., a channel that can be passively overheard by Eve but over which no undetected active attacks by the opponent, such as modifying or inserting messages, are possible (see Figure 15).

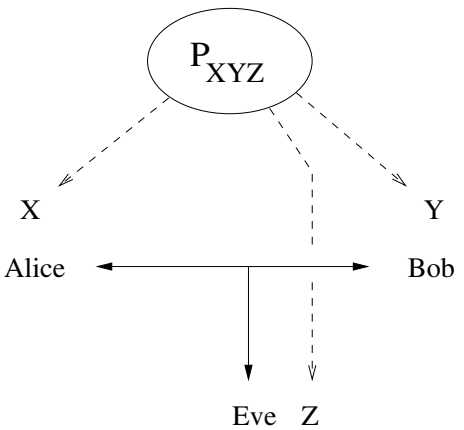


Fig. 15. Secret-Key Agreement by Public Discussion from Common Information

Note that it is natural to consider this model by the following reasons. First, it is an interactive (i.e., two-way) generalization of Wyner's and Csiszár and Körner's models. It is not necessary to assume the existence of noisy communication channels in this interactive setting because equivalents of such channels can be obtained by the same trick as shown in Section 4.3 for inverting the binary-symmetric channel. Secondly, the assumption that the parties have access to correlated randomness appears to be realistic in many contexts. An example of a possible physical implementation is described in Section 5.2.

In analogy to the previous models, where the channels could be used many times independently, we assume here that the parties have access to a number of independent realizations of the corresponding random variables. Consequently, the so-called *secret-key rate* is defined in this model as the maximal rate at which Alice and Bob can generate a highly secret key by communication over the

insecure channel, where the required number of channel uses from the definition of the secrecy capacity is replaced by the amount of randomness (i.e., the number of realizations of X and Y) necessary for the generation of a key of some length.

Definition 5. *The* secret-key rate $S(X;Y||Z)$ *of the distribution* P_{XYZ} *is the maximal number* R *with the following property. For every* $\varepsilon > 0$*, there is a number* N_0 *such that for all* $N \geq N_0$*, a protocol exists that uses authenticated public discussion and satisfies the following conditions. (We denote the block of the first* N *realizations of the random variable* X*,* $[X_1, X_2, \ldots, X_N]$*, by* X^N*, and analogous for* Y *and* Z*. Furthermore, let* U *be the entire communication held over the public channel during the execution of the protocol.) There exist* k*-bit strings* S *and* S' *with*

$$k > (R - \varepsilon)N \ , \tag{10}$$

$$H(S \,|\, X^N U) = 0 \ , \tag{11}$$

$$H(S' \,|\, Y^N U) = 0 \ , \tag{12}$$

$$\mathrm{Prob}\,[S \neq S'] < \varepsilon \ , \tag{13}$$

$$I(S\,;\, Z^N U) < \varepsilon \ , \tag{14}$$

$$H(S) > k - \varepsilon \ . \tag{15}$$

In other words, these conditions guarantee that Alice (11) and Bob (12) can generate almost uniformly distributed (15) keys of a certain length (10) that are equal with high probability (13) and about which the adversary has virtually no information (14). ◦

The notion of the secret-key rate is stronger than the one of secrecy capacity in the sense that in the definition of $C_S(P_{YZ|X})$, it was required that *the rate* at which Eve obtains information about the key is small, whereas here, the *total amount of information* about the entire key must be negligible. (However, one can show that the secret-key rates with respect to the weaker and the stronger definitions are equal [19].)

The secret-key rate is a quite fundamental and mathematically interesting property of a distribution P_{XYZ}. One challenging problem in this context is to enlighten the exact relationship between P_{XYZ} and $S(X;Y||Z)$, i.e., to determine the secret-key rate of a given distribution, or at least to decide whether the rate is non-zero and secret-key agreement is possible in principle in a particular situation. We discuss these questions in Section 6.

5.2 The Satellite Scenario and Phases of Secret-Key Agreement Protocols

The following realistic special scenario was proposed in [21] and completely analyzed in [25]. Assume that a satellite sends out random bits at very low signal power and that Alice, Bob, and Eve receive these bits over independent binary-symmetric channels with error probabilities α, β, and ε, respectively (see Figure 16).

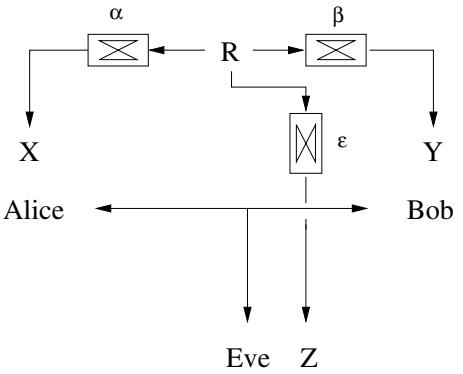


Fig. 16. The Satellite Scenario

In general, we may have to assume that Eve has a better antenna than the legitimate partners, and hence a possibly substantially lower error rate. It is a somewhat surprising fact that secret-key agreement is always possible in this scenario (unless Eve has a noiseless access to the satellite bits or either Alice or Bob obtains no information at all about these bits).

In the following, we describe a protocol for secret-key agreement in the satellite scenario. Such a protocol is often interpreted as consisting of three phases. As mentioned, Alice and Bob possibly start in a situation in which the adversary has an advantage over the legitimate partners with respect to the information about each other's random variables. The objective of the first phase, *advantage distillation*, is to generate an advantage over the opponent by exploiting the authenticity of the public channel. However, Alice and Bob do generally not share a mutual string after this phase. Hence, an interactive error-correction phase, *information reconciliation*, is required. Finally, the resulting mutual but only partially secret string must be transformed into a (shorter) highly secret string. This final phase is called *privacy amplification*. In the illustration of the three phases in Figure 17, the relations between the amounts of information that Bob's and Eve's knowledge provide about Alice's string are shown. The protocol steps are described in detail in the next three sections. An interactive demonstration of the phases is provided on the Internet [7].

5.3 Advantage Distillation

We assume the satellite scenario described in the previous section with error probabilities $0 \leq \alpha, \beta < 1/2$ and $0 < \varepsilon < \min\{\alpha, \beta\}$, i.e., the adversary has an initial advantage over the legitimate partners in terms of the error probabilities. Let us consider N independent realizations of the random variables. Then, we have

$$I(X^N; Y^N) = \sum_{i=1}^{N} I(X_i\,;\, Y_i) \;=\; N \cdot (1 - h(\alpha(1-\beta) + (1-\alpha)\beta)) \;,$$

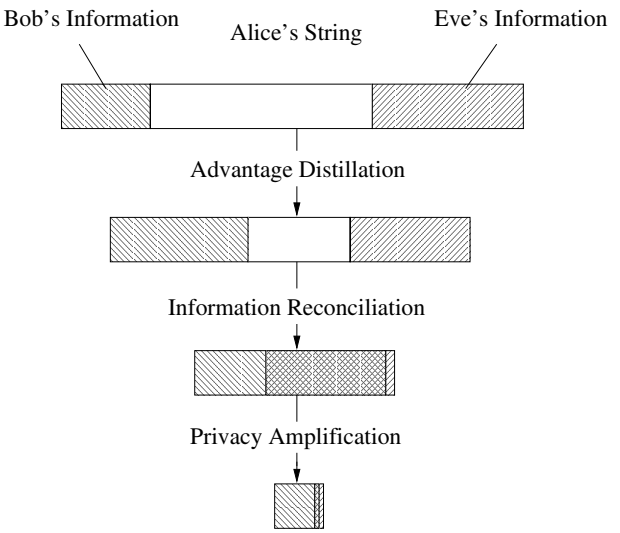


Fig. 17. Phases of a Secret-Key-Agreement Protocol

$$I(X^N;Z^N) = \sum_{i=1}^{N} I(X_i\,;\,Z_i) \;=\; N\cdot(1-h(\alpha(1-\varepsilon)+(1-\alpha)\varepsilon))\;,$$
$$I(Y^N;Z^N) = \sum_{i=1}^{N} I(Y_i\,;\,Z_i) \;=\; N\cdot(1-h(\beta(1-\varepsilon)+(1-\beta)\varepsilon))\;,$$

i.e.,

$$I(X^N;Y^N) < \min\left\{I(X^N;Z^N)\,,\; I(Y^N;Z^N)\right\}\;.$$

The basic idea of the advantage-distillation phase is that Alice and Bob use the noiseless discussion channel for exchanging information about their bits in an insecure but authentic way with the objective of identifying bits that are correct with a higher probability than others. We describe two different protocols that achieve this. The protocols are based on a repeat code and on the exchange of parity-check bits. The repeat-code protocol is simpler to describe, but very inefficient with respect to the required number of realizations of the random variables, whereas the parity-check protocol appears to be quite efficient. For a detailed analysis of the protocols, see for example [21],[20],[24].

Repeat-Code Protocol. The repeat-code protocol works as follows (see also Figure 18). Let N be a fixed parameter. Alice chooses a random bit C and computes

$$C^N \oplus X^N := [C\oplus X_1, C\oplus X_2, \ldots, C\oplus X_N]\;,$$

where C^N stands for the repeat-code block $[C, C, \ldots, C]$ of length N. She sends this "blinded" repeat-code block over the public channel. Bob computes from

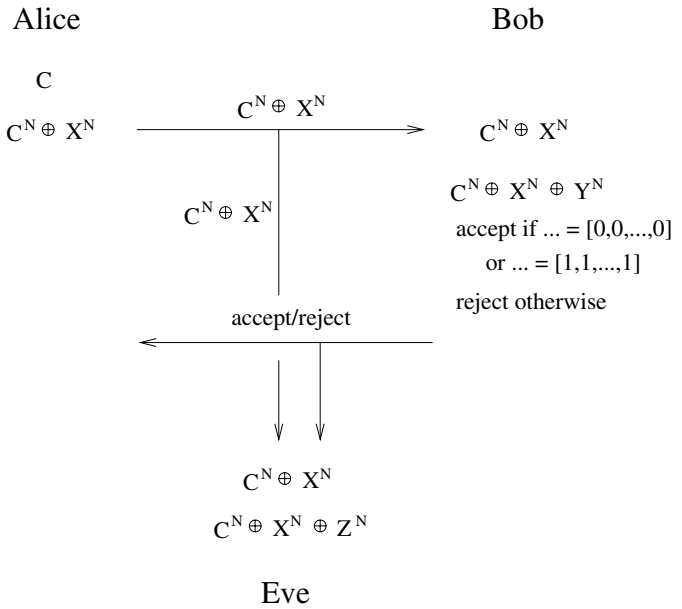


Fig. 18. The Repeat-Code Protocol

this the block $(C^N \oplus X^N) \oplus Y^N$, and sends an "accept" message over the discussion channel if and only if the resulting block is a repeat-code block $(C')^N = [C', C', \ldots, C']$. Note that this is exactly the situation where Alice and Bob have either the same bit in all positions, i.e., $X^N = Y^N$, or opposite values in each position, i.e., $X^N = Y^N \oplus 1^N$. It is intuitively clear that Alice and Bob not only obtain an arbitrarily low probability of the event that $C' \neq C$ for large N this way, but also that they improve their position compared to the opponent by accepting only in situations of apparently highly reliable transmission. However, also the adversary Eve, who can compute $(C^N \oplus X^N) \oplus Z^N$, takes advantage of a greater value of N. It is a somewhat surprising result that for *arbitrary* values of $\alpha, \beta < 1/2$, and $\varepsilon > 0$, Alice and Bob end up in an advantageous situation (both with respect to the error probabilities and to the information about each other's strings) for sufficiently large N.

We show this with respect to the error probabilities of Bob and Eve when guessing the bit C for the special case $\alpha = \beta$ (which we can assume without loss of generality because noise can always be added). We denote by α_{be} the probability that the single bit 0 sent by Alice over the conceptual channel (i.e., $C \oplus X$ is sent over the public channel) is received (i.e., decoded) by Bob as b and by Eve as e. Then we have

$$\begin{aligned}
\alpha_{00} &= (1-\alpha)^2(1-\varepsilon) + \alpha^2\varepsilon \ , \\
\alpha_{01} &= (1-\alpha)^2\varepsilon + \alpha^2(1-\varepsilon) \ , \\
\alpha_{10} &= \alpha_{11} \ = \ (1-\alpha)\alpha \ .
\end{aligned}$$

We assume that N is an even integer. The probability γ that Bob accepts the N-bit block sent by Alice and that $C' \neq C$ holds is

$$\gamma = (\alpha_{10} + \alpha_{11})^N \ ,$$

whereas the probability δ that Bob accepts and Eve guesses the bit incorrectly is lower bounded by $1/2$ times the probability of the event that the block $(C^N \oplus X^N) \oplus Z^N$ which Eve obtains consists of $N/2$ 0's and the same number of 1's, i.e.,

$$\delta \geq \frac{1}{2} \binom{N}{N/2} \alpha_{01}^{N/2} \cdot \alpha_{01}^{N/2} \approx \frac{1}{2} (2\sqrt{\alpha_{00}\alpha_{01}})^N \ .$$

Clearly, the actual message bit C is statistically independent of the block Eve receives if this event occurs. It is not difficult to see that

$$2\sqrt{\alpha_{00}\alpha_{01}} > \alpha_{10} + \alpha_{11}$$

holds for $\alpha < 1/2$ and $\varepsilon > 0$, meaning that Bob's error probability decreases asymptotically faster than Eve's and is hence smaller for sufficiently large N. One can even show that Eve has less information than Bob about the bit C for sufficiently large N.

Parity-Check Protocol. The second protocol we discuss uses parity-check bits and works as follows. Alice computes the parity bit $X_1 \oplus X_2$ and sends it over the public channel. Bob accepts if and only if $X_1 \oplus X_2 = Y_1 \oplus Y_2$, i.e., if the parities of Alice's and Bob's first two bits are equal. In this case, the values X_1 and Y_1 are chosen by Alice and Bob, respectively, for the next protocol round (whereas otherwise, the bits are discarded). This step is repeated a number of times. After this first round it may be necessary, depending on the initial error probabilities, to carry out some additional rounds (see Figure 19).

It is not difficult to see that r rounds of the parity-check protocol are equivalent to the repeat-code protocol with 2^r-bit blocks with respect to the resulting error probabilities. However, it is obvious that the parity-check protocol is much more efficient.

5.4 Information Reconciliation

During advantage distillation, the partners Alice and Bob compute (possibly distinct) strings S_A and S_B, respectively, about which the adversary also has some information. At the end of the key-agreement protocol however, Alice's and Bob's strings must be equal and highly secure, both with overwhelming probability. The information-reconciliation phase consists of interactive error correction and establishes the first of these two conditions.

After advantage distillation, Bob has more information about Alice's string than Eve has, and after information reconciliation, Bob should exactly know Alice's string. (A more general condition would be that after information reconciliation, Alice and Bob share a string that is equally long as S_A and S_B.) This

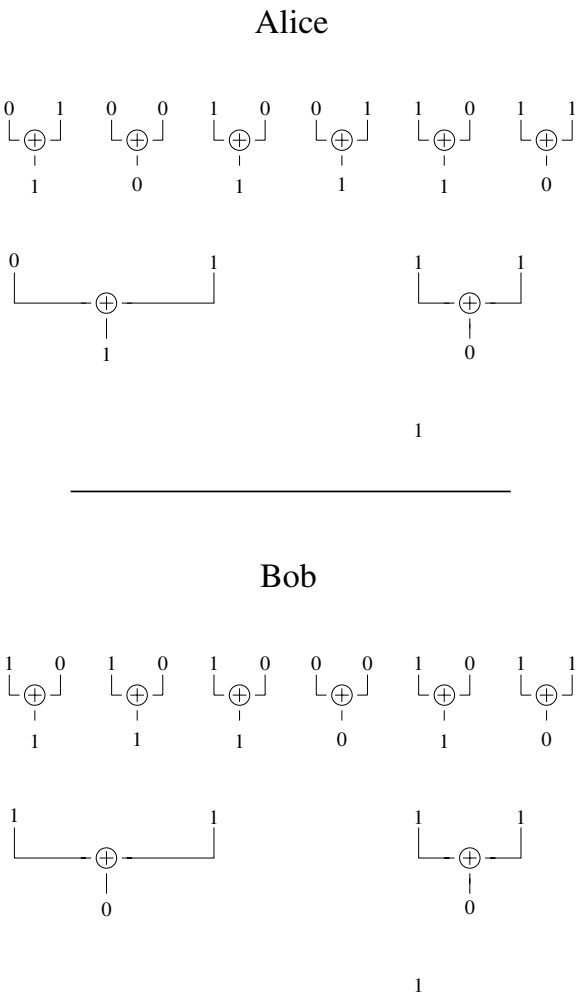


Fig. 19. Three Rounds of the Parity-Check Protocol

leads to a lower bound on the amount of error-correction information E that must be exchanged. Namely, Bob must know S_A completely with overwhelming probability when given S_B and E, i.e.,

$$0 \approx H(S_A \,|\, S_B, E) \geq H(S_A \,|\, S_B) - H(E) \ ,$$

and hence

$$H(E) \gtrsim H(S_A \,|\, S_B) \ .$$

On the other hand, the uncertainty of S_A from Eve's viewpoint can as well be reduced by $H(E)$ in the worst case when Eve learns E (see Figure 20).

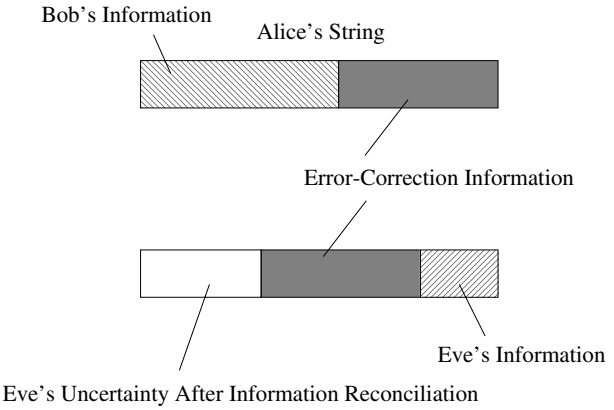


Fig. 20. The Effect of Information Leaked During Information Reconciliation

A good protocol for information reconciliation should both minimize the amount of information leaked to the adversary and be efficient. Examples of protocols satisfying both conditions are given in [6]. We sketch two examples of these protocols. The first is optimal with respect to the information leaked but completely inefficient, whereas the second protocol leaks more information but is more efficient as well. Let us assume that Alice and Bob have finished the advantage-distillation phase in the satellite model. In other words, Bob's string is a (good) estimate about Alice's string, i.e., the same string with a (small) number of errors.

Random-Label Protocol. The first, non-interactive, protocol works as follows. Alice randomly chooses a function f mapping $\{0,1\}^n \to \{0,1\}^m$ (where n is the length of S_A and S_B and m can be roughly equal to $H(S_A|S_B)$) among all such functions and sends (a description of) f together with $f(S_A)$ to Bob, who determines the string S'_A with minimal Hamming distance from S_B that satisfies $f(S'_A) = f(S_A)$. According to the discussion above, and because $m \approx H(S_A|S_B)$, this protocol is optimal with respect to the leaked information. However, it is completely inefficient, hence useless, by the following two reasons. First, the description of the random function f would require $m2^n$ bits. Furthermore, S'_A cannot be efficiently determined from f, $f(S_A)$, and S_B.

Binary-Search Protocol. The idea of the second protocol is to interactively detect the positions where Alice's and Bob's strings differ and to correct these errors. Alice and Bob start by comparing the parity bit, i.e., the XOR-sum, of the bits in randomly but identically chosen substrings S'_A and S'_B of S_A and S_B, respectively. If there are bit errors between the strings S_A and S_B, then the resulting parity bits differ with probability $1/2$ over the choice of the substrings.

If the parities are different, Alice and Bob have detected substrings containing an odd number of errors with respect to each other, and they can locate one of

them by partitioning the substring into two subsets of equal size, one of which clearly contains an odd number of errors as well (and has different parity sums). This splitting procedure is continued until the error is localized and can be corrected by Bob (see Figure 21).

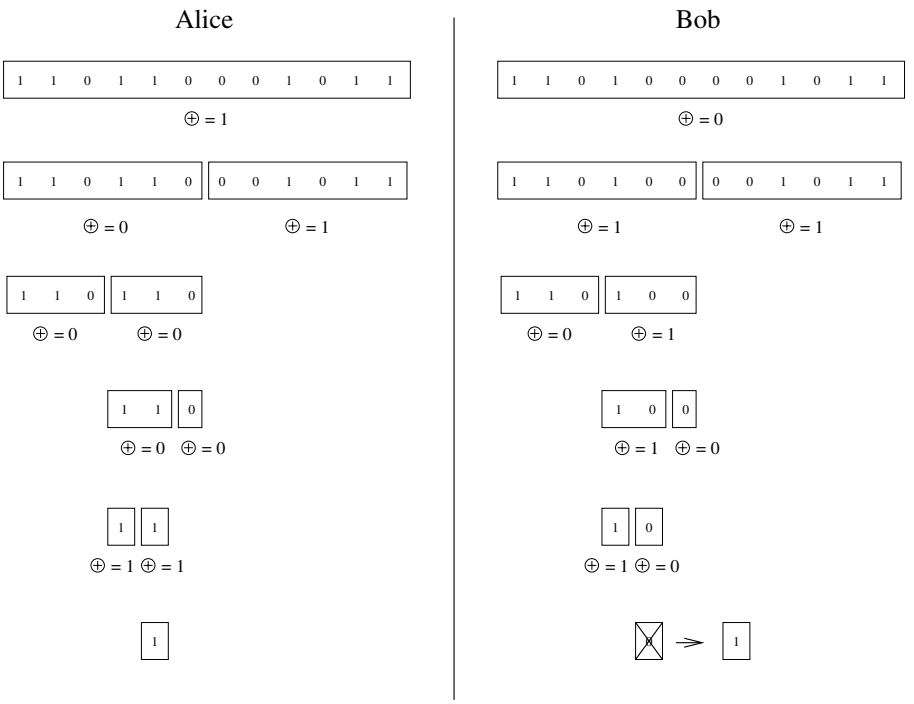


Fig. 21. Finding and Correcting an Error by Comparing Parities

Alice and Bob repeat this procedure until all the errors are found and corrected. If n is the length of the strings S_A and S_B, this protocol requires the exchange of $\lceil \log_2 n \rceil$ bits per error to be corrected. Hence it is efficient if the strings of Alice and Bob differ in only a few bit positions.

After information reconciliation, Alice and Bob have agreed on a mutual string S about which Eve has a possibly considerable amount of information consisting of both a priori knowledge but also information (e.g., physical bits or parities thereof) leaked during information reconciliation.

5.5 Privacy Amplification

Privacy amplification is the art of shrinking a partially secure string S to a highly secret string S' by public discussion. Hereby, the information of the ad-

versary about S can consist of physical bits, of parities thereof, or other types of information (see Figure 22).

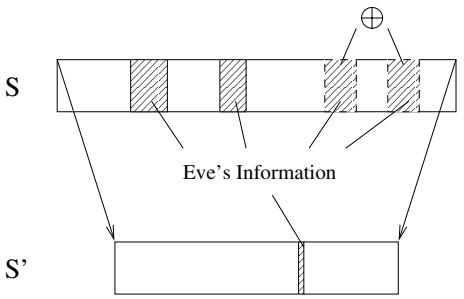


Fig. 22. Eliminating Eve's Knowledge by Privacy Amplification

The following questions related to privacy amplification were studied and answered in [4],[3]. What is a good technique of computing S' from S? What is the possible length of S', depending on this shrinking technique and on the adversary's (type and amount of) information about S?

It is quite clear that the best technique would be to compute S' (of length r) from the n-bit string S by applying a random function $f : \{0,1\}^n \to \{0,1\}^r$. However, Alice and Bob would have to exchange $r2^n$ bits of information to agree on such a function. On the other hand, there exist relatively small classes of functions with "random-like" properties. Examples are so-called *universal classes* of hash functions, which turned out to be useful for privacy amplification.

Definition 6. *A class $\mathcal{H}$ of functions h mapping a set $\mathcal{A}$ to a set $\mathcal{B}$ is called* universal *if for all $x, y \in \mathcal{A}$, $x \neq y$, we have*

$$\text{Prob}_{h \in_r \mathcal{H}}[h(x) = h(y)] = \frac{1}{|\mathcal{B}|} ,$$

where $h \in_r \mathcal{H}$ stands for the fact that h is chosen randomly in $\mathcal{H}$ according to the uniform distribution. In other words, a function that is chosen randomly from a universal class behaves like a completely random function with respect to collisions. ○

An example of a universal class of functions, mapping $\{0,1\}^n$ to $\{0,1\}^r$, of cardinality $2^{n \cdot r}$ are the *linear* functions. There exist even smaller classes. For more examples and lower bounds on the size of universal classes, see for example [33].

We analyze the following type of privacy amplification protocols. First, Alice chooses a random function h from a fixed universal class $\mathcal{H}$ of hash functions mapping n-bit strings to r-bit strings for some r to be determined, and sends (the description of) h publicly to Bob, i.e., also Eve learns h. Then Alice and Bob both compute $S' := h(S)$.

Let us consider the question how long the virtually secure string S' can be, depending on the type and amount of Eve's knowledge about S. Note first that the fact that Eve has some information about a string S is another way of saying that given Eve's entire knowledge $U = u$ about S, the random variable S is *not* uniformly distributed, i.e.,

$$H(S \,|\, U = u) < n \ .$$

In this case we say that Eve has $n - H(S|U = u)$ bits of (Shannon-) information about S. Because the resulting string S' must satisfy

$$H(S' \,|\, C, U = u) \approx r$$

(where r is the length of S' and C is the communication held over the public channel), privacy amplification can be interpreted as "distribution smoothing."

Intuitively, one might think that if Eve has t bits of information about S, then the length r of the resulting string S' can be roughly $n - t$ (see Figure 23). This fact was shown to be correct if Eve has *deterministic* information about

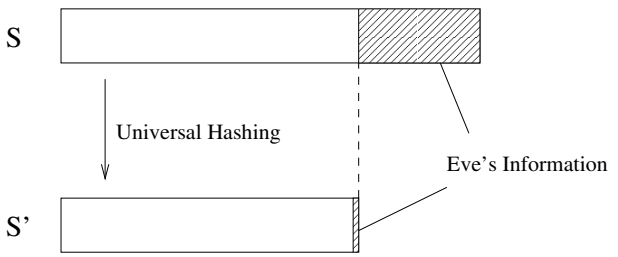


Fig. 23. Can Eve's Knowledge Be Simply Cut Away by Universal Hashing?

S, i.e., if Eve knows the value $g(S)$ for some fixed function g [4]. However, if Eve's information is not deterministic, it is *not* true in general that $n - t$ secure bits can be extracted when Eve has t bits of Shannon information about S, as the following example shows. Let $P_{S|U=u}(s_0) = 1/2$ for some $s_0 \in \{0,1\}^n$, and $P_{S|U=u}(s) = 1/(2 \cdot (2^n - 1))$ for all n-bit strings $s \neq s_0$. Then, we have $H(S|U = u) \approx n/2$, but no secure string S' (of any length, let alone $n/2$) can be extracted because Eve precisely knows S, hence also $S' = h(S)$, with probability $1/2$ (where h is the randomly chosen hash function). This means that S' cannot be highly secure.

The answer to the question what a suitable information (or entropy) measure is with the property that the above intuition (illustrated in Figure 23) is true, was given in [3] as follows.

Definition 7. *For a random variable X with distribution P_X, the* collision probability $P_C(X)$ *is defined as*

$$P_C(X) := \sum_{x \in \mathcal{X}} P_X(x)^2 \ .$$

The collision entropy *or* Rényi entropy (of order 2) of X *is*

$$H_2(X) := -\log_2(P_C(X)) = -\log_2(\sum_{x\in\mathcal{X}} P_X(x)^2) \ .$$

○

The collision probability is the probability that two independent realizations of the random variable X show the same value. Equivalently, it is the probability of guessing a realization of X correctly with the optimal strategy on the basis of an independent realization of X, where the distribution of X is unknown. Jensen's inequality implies

$$H_2(X) = -\log_2(\mathrm{E}\,[P_X]) \leq \mathrm{E}\,[-\log_2 P_X] = H(X) \ .$$

It was shown that Rényi entropy is a good information measure in the context of privacy amplification by universal hashing. Theorem 8 (see also Figure 24) implies that the intuitive fact illustrated in Figure 23 is true with respect to Rényi instead of Shannon information.

Theorem 8. [3] *Let S be an n-bit string with conditional distribution $P_{S|U=u}$ (given Eve's knowledge $U = u$ about S) and Rényi entropy $H_2(S|U = u)$, let G be the random variable corresponding to the random choice (with uniform distribution) of a member g of a universal class $\mathcal{H}$ of hash functions mapping n-bit strings to r-bit strings, and let $S' = G(S)$. Then*

$$r \geq H(S'|G, U = u) \geq H_2(S'|G, U = u) \geq r - \frac{2^{r-H_2(S\,|\,U=u)}}{\ln 2} \ .$$

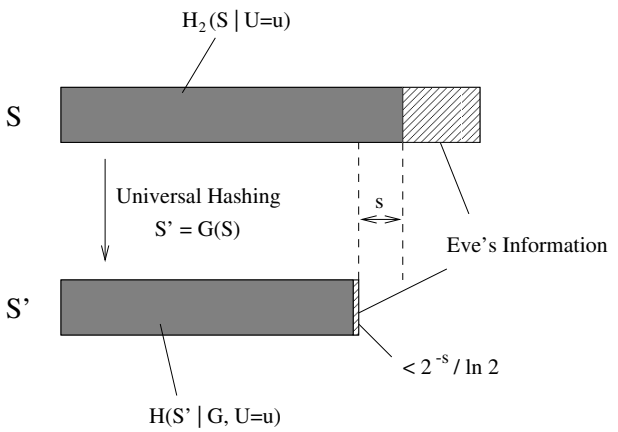


Fig. 24. Rényi Entropy Can Be Extracted by Universal Hashing

Intuitively, Theorem 8 states that if the length r of S' is chosen as

$$r := H_2(S \,|\, U = u) - s \ ,$$

where s is a security parameter, then the resulting string S' is highly secret, where the security increases exponentially in s.

Note that this result in not tight and can be improved in many cases. One reason for this is the counterintuitive fact that Rényi entropy can be increased by giving side information, so called *spoiling knowledge*. By using this property it was shown in [8] that Rényi entropy of order α, for $1 < \alpha < 2$, is a good measure with respect to privacy amplification as well.

One important question finally concerns the influence of the information exchanged during the information-reconciliation phase on the Rényi entropy of S from Eve's point of view, hence on the length of the key that can finally be generated. It was shown in [8] that learning r physical bits cannot reduce Rényi entropy by significantly more than r but with negligible probability.

6 Generalizing the Model

The scenario where the parties receive independent noisy versions of the same random source's signal was completely analyzed in [25],[23]. Possible real-world realizations of the required information source are a satellite sending random bits at low signal power, a pulsar, a deep-space radio source, or randomly polarized photons. However, many more general scenarios can be thought of where the parties receive a different type of correlated information. The assumptions that the parties obtain noisy versions of a common signal or that they have access to a great number of independent realizations of the same random experiment can be modified or dropped. An example is the scenario where Alice, Bob, and Eve obtain a number of playing cards from the same stack [15]. As another generalization, the adversary can be assumed to be more powerful. For instance, it may often be unrealistic to guarantee that the opponent is only a passive wire-tapper.

6.1 Arbitrary Random Variables

Let us have a closer look at the scenario of arbitrary correlated information, i.e., of an arbitrary random experiment P_{XYZ} with many independent realizations (see Figure 15). Note that this is exactly the setting for which the secret-key rate $S(X;Y||Z)$ is defined. In this general case it is a fundamental and natural problem to determine $S(X;Y||Z)$ for a given distribution P_{XYZ}, or at least to decide whether the quantity is non-zero. The following bounds depend on information-theoretic quantities directly derived from P_{XYZ}. The lower bound

$$\max\{I(X;Y) - I(X;Z)\,, I(Y;X) - I(Y;Z)\} \leq S(X;Y||Z)$$

is a consequence of the above-mentioned result by Csiszár and Körner [12] and states that an existing advantage over the adversary can be fully (and even

non-interactively) exploited to generate a secret key. As shown in the previous sections, this bound is *not* tight: Secret-key agreement can also be possible in scenarios where Alice and Bob start in a "bad" situation. On the other hand, the following upper bound was shown in [21]:

$$S(X;Y||Z) \leq \min\left\{I(X;Y)\,,\, I(X;Y|Z)\right\}\,. \tag{16}$$

The bound (16) is quite intuitive and states that Alice and Bob cannot extract a larger amount of secret key than the mutual information between their random variables X and Y (with and without giving Eve's random variable Z). However, this bound is not tight neither and can be improved as follows. Trying to reduce the quantity $I(X;Y|Z)$, the adversary Eve can send the random variable Z over a channel, characterized by $P_{\overline{Z}|Z}$, in order to generate the random variable $\overline{Z}$. Clearly,

$$S(X;Y||Z) \leq S(X;Y||\overline{Z}) \leq I(X;Y|\overline{Z}) \tag{17}$$

holds for every such $\overline{Z}$. This motivates the following definition of a new conditional information measure, *the intrinsic conditional mutual information between X and Y when given Z*, which is the infimum of $I(X;Y|\overline{Z})$, taken over all discrete random variables $\overline{Z}$ that can be obtained by sending Z over a channel, characterized by $P_{\overline{Z}|Z}$. The situation is illustrated in Figure 25. (Note that $R(X;Y;Z) \geq 0$ always holds for the particular $\overline{Z}$ which minimizes $I(X;Y|\overline{Z})$.)

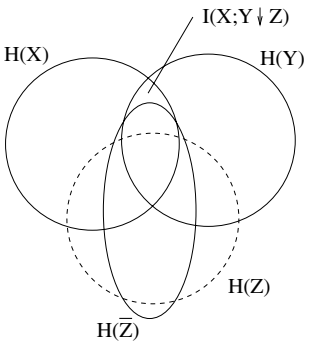


Fig. 25. The Intrinsic Conditional Information

Definition 9. *For a distribution P_{XYZ}, the* intrinsic conditional mutual information between X and Y when given Z, *denoted by $I(X;Y\downarrow Z)$, is*

$$I(X;Y\downarrow Z) := \inf\left\{I(X;Y|\overline{Z})\ :\ P_{XY\overline{Z}} = \sum_{z\in\mathcal{Z}} P_{XYZ}\cdot P_{\overline{Z}|Z}\right\}\,,$$

where the infimum is taken over all possible conditional distributions $P_{\overline{Z}|Z}$. ∘

Intuitively, the intrinsic conditional information $I(X;Y\downarrow Z)$ measures only the information between X and Y, which is possibly reduced by Z, but not the additional information brought in by giving Z. If for example X and Y are independent symmetric bits and $Z = X \oplus Y$, then we have $I(X;Y|Z) = 1$, but $I(X;Y\downarrow Z) = 0$.

It follows from the above that

$$S(X;Y||Z) \leq I(X;Y\downarrow Z) \ .$$

The fundamental problem of generally determining $S(X;Y||Z)$ for given P_{XYZ} has remained open, but there is some evidence that the intrinsic information is exactly the right quantity linking the secret-key rate with the joint distribution of X, Y, and Z.

Conjecture. $$S(X;Y||Z) = I(X;Y\downarrow Z) \ .$$

However, even the generally easier problem of completely characterizing the distributions P_{XYZ} for which $S(X;Y||Z) > 0$ holds, i.e., for which secret-key agreement is possible in principle, has not been fully answered yet (see Figure 26).

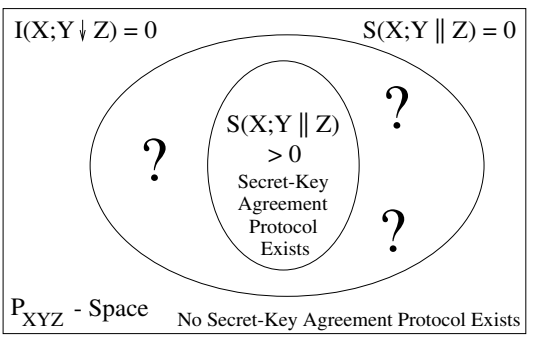


Fig. 26. Characterizing when Secret-Key Agreement Is Possible

6.2 Secret-Key Agreement Secure against ACTIVE Adversaries

In all the previous models, we have assumed that the adversary is only a passive wire-tapper or equivalently, that the public channel connecting Alice and Bob is authentic. In many cases, secret-key agreement is even possible when dropping this condition, i.e., when the adversary is able to modify or introduce messages without being detected. See [17],[23],[35] for a discussion and analysis of this model.

Note first that a protocol secure against active opponents cannot be guaranteed to work in every situation because Eve, who is assumed to have full control

over the public channel, can block the channel permanently, preventing any communication between the legitimate partners. Hence the best that can be achieved by such a protocol is that Alice and Bob detect an adversary's active attacks and reject the outcome of the protocol unless secret-key agreement is successful (see Figure 27). More precisely, it is required that if Eve chooses to remain passive,

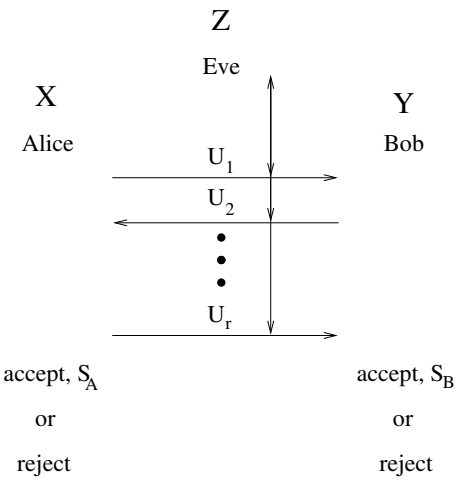


Fig. 27. Unconditional Security Against Active Opponents

then secret-key agreement is successful (as in the passive-adversary model). On the other hand, if Eve is active, then with overwhelming probability either Alice and Bob both reject the outcome of the protocol, or secret-key agreement is successful despite Eve's attacks. (Note that it is not requested that both Alice and Bob accept the outcome in the latter case. Such a perfect synchronization of the acceptance decisions cannot be achieved in the presence of an active adversary, who can always block the final message that makes the second party accept.)

Clearly, secret-key agreement can only be possible in the active-adversary scenario if Alice and Bob have some initial advantage over Eve in terms of the random variables X, Y, and Z. More precisely, this advantage must be such that Eve is not able to perfectly simulate Alice towards Bob and vice versa. In terms of the random variables, this is the condition that she cannot generate, using her random variable Z, a random variable $\overline{X}$ with the property that given only Y, $\overline{X}$ cannot be distinguished from X, and vice versa. Formally, this means that there do not exist conditional distributions $P_{\overline{X}|Z}$ or $P_{\overline{Y}|Z}$ such that either

$$P_{\overline{X}Y} = P_{XY}$$

or $P_{X\overline{Y}} = P_{XY}$ holds, respectively. If one of these distributions existed, secret-key agreement would be impossible because Bob could not tell Alice and Eve apart (or vice versa).

A surprising result however is that if secret-key agreement is possible also in the presence of an active adversary, then asymptotically the same key-generation rate as in the passive-adversary case can be achieved.

Finally, also privacy amplification can be executed in the case where the adversary is active. However, the restrictions on the opponent's knowledge about the partially secret key must be stronger [23],[35]. The idea is to use the string S twice, first as a key for unconditionally authenticating a message containing the description of a randomly chosen hash function, and as the argument for this function.

7 Concluding Remarks

We have described several techniques and results in the context of unconditional security in cryptography. The mentioned possibility and impossibility results can give a rough picture in what settings such provable confidentiality can be achieved. It is an important point in this context that despite Shannon's well-known pessimistic result, unconditional security is not necessarily impractical. A number of fundamental questions in this field are open today. In particular, the ultimate goal is the realization of a system that is practical and provably unconditionally secure simultaneously.

Acknowledgments

It is a pleasure to thank Ueli Maurer for initiating the author's interest in this exciting subject, and for many stimulating discussions. We also thank Ivan Damgård for the invitation to the Cryptology and Data Security Summer School.

References

1. R. Ahlswede and I. Csiszár, "Common randomness in information theory and cryptography - Part I: secret sharing," *IEEE Transactions on Information Theory*, vol. 39, no. 4, pp. 1121–1132, 1993. 228
2. C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, "Experimental quantum cryptography," *Journal of Cryptology*, vol. 5, no. 1, pp. 3–28, Springer-Verlag, 1992. 228
3. C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer, "Generalized privacy amplification," *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 1915–1923, 1995. 241, 242, 243
4. C. H. Bennett, G. Brassard, and J.-M. Robert, "Privacy amplification by public discussion," *SIAM Journal on Computing*, vol. 17, pp. 210–229, 1988. 241, 242
5. R. E. Blahut, *Principles and practice of information theory*, Addison-Wesley Publishing Company, 1988. 218
6. G. Brassard and L. Salvail, "Secret-key reconciliation by public discussion," *Advances in Cryptology - EUROCRYPT '93*, Lecture Notes in Computer Science, vol. 765, pp. 410–423, Springer-Verlag, 1994. 239

7. W. Brunner, C. Cachin, U. M. Maurer, and C. Vonäsch, *Demonstration system of secret-key agreement by public discussion*, ETH Zürich, 1996. http://www.inf.ethz.ch/department/TI/um/keydemo/ 234
8. C. Cachin, *Entropy measures and unconditional security in cryptography*, Ph. D. Thesis, ETH Zürich, Hartung-Gorre Verlag, Konstanz, 1997. 244
9. C. Cachin and U. M. Maurer, "Unconditional security against memory-bounded adversaries," *Advances in Cryptology - CRYPTO '97*, Lecture Notes in Computer Science, vol. 1294, pp. 292–306, Springer-Verlag, 1997. 228
10. T. M. Cover and J. A. Thomas, *Elements of information theory*, Wiley Series in Telecommunications, 1992. 218
11. R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," *Advances in Cryptology - CRYPTO '98*, Lecture Notes in Computer Science, vol. 1462, pp. 13–25, Springer-Verlag, 1998. 218
12. I. Csiszár and J. Körner, "Broadcast channels with confidential messages," *IEEE Transactions on Information Theory*, vol. IT-24, pp. 339–348, 1978. 228, 229, 230, 244
13. W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976. 217
14. W. Feller, *An introduction to probability theory and its applications*, 3rd edition, vol. 1, Wiley International, 1968. 218
15. M. J. Fischer and R. N. Wright, "Bounds on secret key exchange using a random deal of cards," *Journal of Cryptology*, vol. 9, no. 2, pp. 71–99, Springer-Verlag, 1996. 244
16. J. L. Massey, "A simplified treatment of Wyner's wire-tap channel," *Proceedings of the 21st Annual Allerton Conference of Communication, Control, and Computing*, Monticello, IL, pp. 268–276, 1983. 228
17. U. M. Maurer, "Information-theoretically secure secret-key agreement by NOT authenticated public discussion," *Advances in Cryptology - EUROCRYPT '97*, Lecture Notes in Computer Science, vol. 1233, pp. 209–225, Springer-Verlag, 1997. 246
18. U. M. Maurer, "The role of information theory in cryptography," *Codes and Ciphers*: 4th IMA Conference on Cryptography and Coding, Cirencester, UK, Dec. 1993, pp. 49-71, Southend-on-Sea, 1995. The Institute of Mathematics and its Applications.
19. U. M. Maurer, "The strong secret key rate of discrete random triples," *Communication and Cryptography – Two Sides of One Tapestry*, Kluwer Academic Publishers, pp. 271–285, 1994. 233
20. U. M. Maurer, "Protocols for secret key agreement based on common information," *Advances in Cryptology - CRYPTO '92*, Lecture Notes in Computer Science, vol. 740, pp. 461–470, Springer-Verlag, 1993. 235
21. U. M. Maurer, "Secret key agreement by public discussion from common information," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 733–742, 1993. 228, 231, 232, 233, 235, 245
22. U. M. Maurer, "Conditionally-perfect secrecy and a provably-secure randomized cipher," *Journal of Cryptology*, vol. 5, pp. 53–66, Springer-Verlag, 1992. 228
23. U. M. Maurer and S. Wolf, "Privacy amplification secure against active adversaries," *Advances in Cryptology - CRYPTO '97*, Lecture Notes in Computer Science, vol. 1294, pp. 307–321, Springer-Verlag, 1997.

24. U. M. Maurer and S. Wolf, "The intrinsic conditional mutual information and perfect secrecy," *Proc. of the 1997 IEEE Symp. on Information Theory*, Ulm, Germany, 1997 (abstract). To appear in *IEEE Transactions on Information Theory.* 244, 246, 248 235
25. U. M. Maurer and S. Wolf, "Towards characterizing when information-theoretic secret key agreement is possible," *Advances in Cryptology - ASIACRYPT '96*, Lecture Notes in Computer Science, vol. 1163, pp. 196–209, Springer-Verlag, 1996. 233, 244
26. M. Naor and A. Shamir, "Visual cryptography," *Advances in Cryptology - CRYPTO '94*, Lecture Notes in Computer Science, vol. 950, pp. 1–12, Springer-Verlag, 1995. 225
27. A. Rényi, *A diary on information theory*, Akadémiai Kiadó, Budapest, 1978. 219
28. R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978. 217
29. C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, pp. 656–715, 1949. 225, 227
30. C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948. 218, 221
31. P. W. Shor, "Algorithms for quantum computation: discrete log and factoring," *Proceedings of the 35th IEEE Symposium on the Foundations of Computer Science (FOCS '94)*, pp. 124–134, 1994. 218
32. V. Shoup, "Lower bounds for discrete logarithms and related problems," *Advances in Cryptology - EUROCRYPT '97*, Lecture Notes in Computer Science, vol. 1233, pp. 256–266, Springer-Verlag, 1997. 217
33. D. R. Stinson, "Universal hashing and authentication codes," *Advances in Cryptology - CRYPTO '91*, Lecture Notes in Computer Science, vol. 576, pp. 74–85, Springer-Verlag, 1992. 241
34. G. S. Vernam, "Cipher printing telegraph systems for secret wire and radio telegraphic communications," *Journal of the American Institute for Electrical Engineers*, vol. 55, pp. 109–115, 1926. 225
35. S. Wolf, "Strong security against active attacks in information-theoretic secret-key agreement," to appear in *Advances in Cryptology - ASIACRYPT '98*, Lecture Notes in Computer Science, Springer-Verlag, 1998. 246, 248
36. A. D. Wyner, "The wire-tap channel," *Bell System Technical Journal*, vol. 54, no. 8, pp. 1355–1387, 1975. 228, 229
37. R. W. Yeung, "A new outlook on Shannon's information measures," *IEEE Transactions on Information Theory*, vol. 37, no. 3, pp. 466–474, 1991. 223