The Foundations of Modern Cryptography

(Version Nr. 1.4)

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.
Email: oded@wisdom.weizmann.ac.il

July 14, 1997

Abstract

In our opinion, the Foundations of Cryptography are the paradigms, approaches and techniques used to conceptualize, define and provide solutions to natural cryptographic problems. In this essay, we survey some of these paradigms, approaches and techniques as well as some of the fundamental results obtained using them. Special effort is made in attempt to dissolve common misconceptions regarding these paradigms and results.

Contents

It is possible to build a cabin with no foundations,
but not a lasting building.
Eng. Isidor Goldreich (1906–1995)

1 Introduction

Cryptography is concerned with the construction of schemes which are robust against malicious attempts to make these schemes deviate from their prescribed functionality. Given a desired functionality, a cryptographer should design a scheme which not only satisfies the desired functionality under "normal operation", but also maintains this functionality in face of adversarial attempts which are devised after the cryptographer has completed his/her work. The fact that an adversary will devise its attack after the scheme has been specified makes the design of such schemes very hard. In particular, the adversary will try to take actions other than the ones the designer had envisioned. Thus, our approach is that it makes little sense to make assumptions regarding the specific *strategy* that the adversary may use. The only assumptions which can be justified refer to the computational *abilities* of the adversary. Furthermore, it is our opinion that the design of cryptographic systems has to be based on *firm foundations*; whereas ad-hoc approaches and heuristics are a very dangerous way to go. A heuristic may make sense when the designer has a very good idea about the environment in which a scheme is to operate, yet a cryptographic scheme has to operate in a maliciously selected environment which typically transcends the designer's view.

Providing firm foundations to Cryptography has been a major research direction in the last two decades. Indeed, the pioneering paper of Diffie and Hellman [46] should be considered the initiator of this direction. Two major (interleaved) activities have been:

1. Definitional Activity: The identification, conceptualization and rigorous definition of cryptographic tasks which capture natural security concerns; and

2. Constructive Activity: The study and design of cryptographic schemes satisfying definitions as in (1).

The definitional activity provided a definition of secure encryption [74]. The reader may be surprised: *what is there to define* (beyond the basic setting formulated in [46])? Let us answer with a question (posed by [74]): *should an encryption scheme which leaks the first bit of the plaintext be considered secure?* Clearly, the answer is negative and so some naive conceptions regarding secure encryption (e.g., "a scheme is secure if it is infeasible to obtain the plaintext from the ciphertext when not given the decryption key") turn out to be unsatisfactory. The lesson is that even when a natural concern (e.g., "secure communication over insecure channels") has been identified, work still needs to be done towards a satisfactory (rigorous) definition of the underlying concept. The definitional activity also undertook the treatment of unforgeable signature schemes [76]: One result of the treatment was the refutation of a "folklore theorem" (attributed to Ron Rivest) by which "a signature scheme that is robust against chosen message attack cannot have a proof of security". The lesson here is that unclear/unsound formulations (i.e., those underlying the above folklore paradox) lead to false conclusions.

Another existing concept which was re-examined is the then-fuzzy notion of a "pseudorandom generator". Although ad-hoc "pseudorandom generators" which pass some ad-hoc statistical tests may be adequate for some statistical samplings, they are certainly inadequate for use in Cryptography: For example, sequences generated by linear congruential generators are easy to predict [24, 58] and endanger cryptographic applications even when not given in the clear [8]. The alternative suggested in [22, 74, 120] is a robust notion of pseudorandom generators – such a generator produces sequences which are *computationally indistinguishable* from truly random sequences, and thus, can replace truly random sequences in any practical application. We mention that the notion of computational indistinguishability has played a central role in the formulation of other cryptographic concepts (such as secure encryption and zero-knowledge).

The definitional activity has identified concepts which were not known before. One well-known example is the introduction of zero-knowledge proofs [75]. A key paradigm crystallized in making the latter definition is the *simulation paradigm*: A party is said to have gained nothing from some extra information given to it if it can generate (i.e., simulate the receipt of) essentially the same information by itself (i.e., without being given this information). The simulation paradigm plays a central role in the related definitions of secure multi-party computations (with respect to varying settings such as in [97, 2, 73, 27]). However, it has been employed also in different settings such as in [13, 14, 31].

The definitional activity is an on-going process. Its more recent targets have included mobile adversaries (aka "proactive security") [108, 32, 81], Electronic Cash [36], Coercibility [30, 29], Threshold Cryptography [45], and more.

The constructive activity. As new definitions of cryptographic tasks emerged, the first challenge was to demonstrate that they can be achieved. Thus, the first goal of the constructive activity is to *demonstrate the plausibility* of obtaining certain goals. Thus, standard assumptions such as that the RSA is hard to invert were used to construct secure public-key encryption schemes [74, 120] and unforgeable digital schemes [76]. We stress that assuming that RSA is hard to invert is different from assuming that RSA is a secure encryption scheme. Furthermore, plain RSA (alike any deterministic public-key encryption scheme) is not secure (as one can easily distinguish the encryption of one *predetermined* message from the encryption of another). Yet, RSA can be easily transformed into a secure public-key encryption scheme by using a construction which is reminiscent of a common practice (of padding the message with random noise). We stress that the resulting scheme is not merely believed to be secure but rather its security is linked to a much simpler assumption (i.e., the assumption that RSA is hard to invert). Likewise, although plain RSA signing is vulnerable to "existential forgery" (and other attacks), RSA can be transformed into a signature scheme which is unforgeable (provided RSA is hard to invert). Using the assumption that RSA is hard to invert, one can construct pseudorandom generators [22, 120], zero-knowledge proofs for any NP-statement [68], and multi-party protocols for securely computing any multi-variant function [121, 69].

A major misconception regarding theoretical work in Cryptography stems from not distinguishing work aimed at demonstrating the plausibility of obtaining certain goals from work aimed at suggesting paradigms and/or constructions which can be used in practice. For example, the general results concerning zero-knowledge proofs and multi-party protocols [68, 121, 69] mentioned above are merely *claims of plausibility*: What they say is that any problem of the above type (i.e., any protocol problem as discussed in Section 7) can be solved in principle. This is a very valuable piece of information. Thus, if you have a specific problem which falls into the above category then you should know that the problem is solvable in principle. However, if you need to construct a real system then you should probably construct a solution from scratch (rather than employing the above general results). Typically, *some* tools developed towards solving the general problem may be useful in solving the specific problem. Thus, we distinguish three types of results:

1. *Plausibility results:* Here we refer to mere statements of the type "any NP-language has a zero-knowledge proof system" (cf., [68]).

2. *Introduction of paradigms and techniques which may be applicable in practice:* Typical examples include construction paradigms as the "choose n out of $2n$ technique" of [110], the "authentication tree" of [93, 95], the "randomized encryption" paradigm of [74], proof techniques as the "hybrid argument" of [74] (cf., [62, Sec. 3.2.3]), and many others.

3. *Presentation of schemes which are suitable for practical applications:* Typical examples include the public-key encryption schemes of [21], the digital signature schemes of [50, 49], the session-key protocols of [13, 14], and many others.

Typically, it is quite easy to determine to which of the above categories a specific technical contribution belongs. Unfortunately, the classification is not always stated in the paper; however, it is typically evident from the construction. We stress that all results we are aware of (and in particular all results cited in this essay), come with an explicit construction. Furthermore, the security of the resulting construction is explicitly related to the complexity of certain intractable tasks. In contrast to some uninformed beliefs, for each of these results there is an explicit translation of concrete intractability assumptions (on which the scheme is based) into lower bounds on the amount of work required to violate the security of the resulting scheme.[1] We stress that this translation can be invoked for any value of the security parameter. Doing so determines whether a specific construction is adequate for a specific application under specific reasonable intractability assumptions. In many cases the answer is in the affirmative, but in general this does depend on the specific construction as well as on the specific value of the security parameter and on what is reasonable to assume for this value. When we say that a result is suitable for practical applications (i.e., belongs to Type 3 above), we mean that it offers reasonable security for reasonable implementation values of the security parameter and reasonable assumptions.

Other activities. This essay is focused on the definitional and constructive activities mentioned above. Other activities in the foundations of cryptography include the exploration of new directions and the marking of limitations. For example, we mention novel modes of operation such as split-entities [16, 45, 96], batching operations [55], off-line/on-line signing [50] and Incremental Cryptography [6, 7]. On the limitation side, we mention [84, 66]. In particular, [84] indicates that certain tasks (e.g., secret key exchange) are unlikely to be achieved by using a one-way function in a "black-box manner".

Organization: Although encryption, signatures and secure protocols are the primary tasks of Cryptography, we start our presentation with basic paradigms and tools such as computational difficulty (Section 2), pseudorandomness (Section 3) and zero-knowledge (Section 4). Once these are presented, we turn to encryption (Section 5), signatures (Section 6) and secure protocols (Section 7). We conclude with some notes (Section 8), two suggestions for future research (Section 9) and some suggestions for further reading (Section 10).

This is a preliminary version of the current essay. It slightly expands on a version which will appear in Crypto97 (to appear in Springer's Lecture Notes in Computer Science, Vol. 1294). I intend to revise the essay in the future and make revisions available from `http://theory.lcs.mit.edu/~oded/tfoc.html`.

[1] The only exception to the latter statement is Levin's observation regarding the existence of a *universal one-way function* (cf., [90] and [62, Sec. 2.4.1]).

Part I
Basic Tools

2 Central Paradigms

Modern Cryptography, as surveyed here, is concerned with the construction of *efficient* schemes for which it is *infeasible* to violate the security feature. Thus, we need a notion of efficient computations as well as a notion of infeasible ones. The computations of the legitimate users of the scheme ought be efficient; whereas violating the security features (via an adversary) ought to be infeasible. Our notions of efficient and infeasible computations are "asymptotic": They refer to the running time as a function of the security parameter. This is done in order to avoid cumbersome formulations which refer to the actual running-time on a specific model for specific values of the security parameter. As discussed above one can easily derive such specific statements from the asymptotic treatment. Actually, the term "asymptotic" is misleading since, from the functional treatment of the running-time (as a function of the security parameter), one can derive statements for ANY value of the security parameter.

Efficient computations are commonly modeled by computations which are polynomial-time in the security parameter. The polynomial bounding the running-time of the legitimate user's strategy is fixed and typically explicit and small (still in some cases it is indeed a valuable goal to make it even smaller). Here (i.e., when referring to the complexity of the legitimate user) we are in the same situation as in any algorithmic research. Things are different when referring to our assumptions regarding the computational resources of the adversary. A common approach is to postulate that the latter are polynomial-time too, where the polynomial is NOT a-priori specified. In other words, the adversary is restricted to the class of efficient computations and anything beyond this is considered to be infeasible. Although many definitions explicitly refer to this convention, this convention is INESSENTIAL to any of the results known in the area. In all cases, a more general (and yet more cumbersome) statement can be made by referring to adversaries of running-time bounded by any function (or class of functions). For example, for any function $T:\mathbb{N}\mapsto\mathbb{N}$ (e.g., $T(n)=2^{\sqrt[3]{n}}$), we may consider adversaries which on security parameter n run for at most $T(n)$ steps. Doing so we (implicitly) define as infeasible any computation which (on security parameter n) requires more than $T(n)$ steps. A typical result has the form[2]

> *If RSA with n-bit moduli cannot be inverted in time $T(n)$ then the following construction (using security parameter n) is secure against adversaries operating in time $T'(n)=T(g(n))/f(n)$, where f and g^{-1} are explicitly given polynomials.*

However, most papers prefer to present a simplified statement of the form "if RSA cannot be inverted in polynomial-time then the following construction is secure against polynomial-time adversaries". This is unfortunate since it is the specific functions f and g, which are (sometimes explicit and) always implicit in the proof, that determine the practicality of the construction.[3]

[2] Actually, the form below is over-simplified. The actual statement refers also to the success probabilities of both attacks. It reads: If RSA with n-bit moduli cannot be inverted in time $T(n)$, with success probability greater than $\epsilon(n)$, then the following construction (using security parameter n) cannot be broken by adversaries operating in time $T'(n)$ with success probability greater than $\epsilon'(n)$, where $T'(n)$ and $\epsilon'(n)$ are related to $T(g(n))$ and $\epsilon(g(n))$ via explicit polynomial expressions and g^{-1} is an explicitly given polynomial. Specifically, $T(g(n))=\mathrm{poly}(n,T'(n))/\mathrm{poly}(\epsilon'(n))$ and $\epsilon(g(n))=\mathrm{poly}(\epsilon'(n))/\mathrm{poly}(n,T'(n))$. Typically, $T(g(n))=\mathrm{poly}(n/\epsilon'(n))\cdot T'(n)$ and $\epsilon(g(n))=\mathrm{poly}(\epsilon'(n))/\mathrm{poly}(T'(n))$.

[3] The importance of *explicitly* relating the security of the resulting scheme to the quantified intractability assumption has been advocated (and practiced) in a sequence of recent works by Bellare and Rogaway (cf., [10, p. 343]).

The smaller f and g^{-1}, the better. Our rule of thumb is that results with $g^{-1}(n) = O(n)$ (e.g., $g(n) = n/2$) are practical, whereas results with, say, $g^{-1}(n) = n^4$ (i.e., $g(n) = \sqrt[4]{n}$) are to be considered merely plausibility results.

Lastly we consider the notion of a **negligible probability**. The idea behind this notion is to have a robust notion of rareness: A rare event should occur rarely even if we repeat the experiment for a feasible number of times. That is, if we consider any polynomial-time computation to be feasible then any function $f : \mathbb{N} \mapsto \mathbb{N}$ so that $(1 - f(n))^{p(n)} < 0.01$, for any polynomial p, is considered negligible (i.e., f is negligible if for any polynomial p the function $f(\cdot)$ is bounded above by $1/p(\cdot)$). However, if we consider the function $T(n)$ to provide our notion of infeasible computation then functions bounded above by $1/T(n)$ are considered negligible (in n).

In the rest of this essay we adopt the simpler convention of defining infeasible computations as ones which cannot be conducted in polynomial-time. (However, we explicitly state the level of practicality of each of the results presented.) The interested reader is referred to [91] for a more general treatment.

2.1 Computational Difficulty

Modern Cryptography is concerned with the construction of schemes which are easy to operate (properly) but hard to foil. Thus, a complexity gap (i.e., between the complexity of proper usage and the complexity of defeating the prescribed functionality) lies in the heart of Modern Cryptography. However, gaps as required for Modern Cryptography are not known to exist – they are only widely believed to exist. Indeed, almost all of Modern Cryptography rises or falls with the question of whether one-way functions exist (e.g., see [79, 63, 115, 99, 68] for positive results and [90, 115, 107] for negative ones). One-way functions are functions which are easy to evaluate but hard (on the average) to invert.

Definition 1 (one-way functions [46]): *A function $f:\{0,1\}^* \mapsto \{0,1\}^*$ is called* one-way *if*

- easy direction: *there is an efficient algorithm which on input x outputs $f(x)$.*
- hard direction: *given $f(x)$, where x is uniformly selected, it is infeasible to find, with non-negligible probability, a preimage of $f(x)$. That is, any feasible algorithm which tries to do invert f may succeed only with negligible probability, where the probability is taken over the choices of x and the algorithm's coin tosses.*

Warning: the above definition, as well as all other definitions in this essay, avoids some technicalities and so is imprecise. The interested reader is referred to other texts (see Section 10).

2.2 Computational Indistinguishability

A central notion in Modern Cryptography is that of "effective similarity". The underlying idea is that we do not care if objects are equal or not – all we care is whether a difference between the objects can be observed by a feasible computation. In case the answer is negative, we may say that the two objects are equivalent as far as any practical application is concerned. Indeed, it will be our common practice to interchange such (computationally indistinguishable) objects.

Definition 2 (computational indistinguishability [74, 120]): *Let $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ be probability ensembles such that each X_n and Y_n ranges over strings of length n. We say that X and Y are* computationally indistinguishable *if for every feasible algorithm A the difference*

$$d_A(n) \stackrel{\text{def}}{=} |\Pr(A(X_n)=1) - \Pr(A(Y_n)=1)|$$

is a negligible function in n.

2.3 The Simulation Paradigm

A key question regarding the modeling of security concerns is how to express the intuitive requirement that an adversary "gains nothing substantial" by deviating from the prescribed behavior of an honest user. The approach initiated in [74, 75] is that the adversary *gains nothing* if whatever it can obtain by deviating from the prescribed honest behavior can also be obtained in an appropriately defined "ideal model". The definition of the "ideal model" captures what we want to achieve in terms of security, and so is specific to the security concern to be addressed. For example, an encryption scheme is considered secure (against eavesdropping) if an adversary which eavesdrops on a channel on which messages are sent, using this encryption scheme, gains nothing over a user which does not tap this channel. Thus, the encryption scheme "simulates" an ideal private channel between parties.

A notable property of the above simulation paradigm, as well as of the entire approach surveyed here, is that this approach is very liberal with respect to its view of the abilities of the adversary as well as to what might constitute a gain for the adversary. For example, we consider an encryption scheme to be secure only if it can simulate a private channel. Indeed, failure to provide such a simulation does NOT necessarily mean that the encryption scheme can be "broken" in some intuitively harmful sense. Thus, it seems that our approach to defining security is overly cautious. However, it seems impossible to come up with definitions of security which distinguish "breaking the scheme in a harmful sense" from "breaking it in a non-harmful sense": What is harmful is application-dependent, whereas a good definition of security ought to be application independent (as otherwise using the scheme in any new application will require a full re-evaluation of its security). Furthermore, since we are interested in secure schemes, there is no harm in employing overly cautious definitions, provided that this does not prevent us (or even disturb us) from constructing "good" schemes. We claim that this has been the case in the past. In most cases it has been possible to construct schemes which meet the overly cautious definitions (of security), and in other cases the difficulty to construct such schemes has demonstrated an inherent problem (e.g., [84, 66]).

3 Pseudorandomness

In practice "pseudorandom" sequences are used instead of truly random sequences in many applications. The underlying belief is that if an (efficient) application performs well when using a truly random sequence then it will perform essentially as well when using a "pseudorandom" sequence. However, this belief is not supported by previous characterizations of "pseudorandomness" (e.g., such as passing the statistical tests in Knuth's book or having large linear-complexity). In contrast, the above belief is an easy corollary of defining pseudorandom distributions as ones which are computationally indistinguishable from uniform distributions. We are interested in pseudorandom sequences which can be generated and determined by short random seeds. That is,

Definition 3 (pseudorandom generator [22, 120]): *Let $\ell : \mathbb{N} \mapsto \mathbb{N}$ be so that $\ell(n) > n$, $\forall n$. A* pseudorandom generator*, with* stretch function *ℓ, is an efficient* (deterministic) *algorithm which on input a random n-bit* seed *outputs a $\ell(n)$-bit sequence which is computationally indistinguishable from a uniformly chosen $\ell(n)$-bit sequence.*

We stress that pseudorandom sequences can replace truly random sequences not only in "ordinary" computations but also in cryptographic ones. That is, ANY cryptographic application which is secure when the legitimate parties use truly random sequences, is also secure when the legitimate parties use pseudorandom sequences. Various cryptographic applications of pseudorandom generators will be presented in the sequel, but first let us consider the construction of pseudorandom generators. A key paradigm is presented next. It uses the notion of a *hard-core* predicate [22] of a (one-way) function: The predicate b is a hard-core of the function f if b is easy to evaluate but $b(x)$ is hard to predict from $f(x)$. That is, it is infeasible, given $f(x)$ when x is uniformly chosen, to predict $b(x)$ substantially better than with probability 1/2. Intuitively, b "inherits *in a concentrated sense*" the difficulty of inverting f. (Note that if b is a hard-core of an efficiently computable 1-1 function f then f must be one-way.)

The iteration paradigm [22]: Let f be a 1-1 function which is length-preserving and efficiently computable, and b be a hard-core predicate of f. Then

$$G(s) = b(s) \cdot b(f(s)) \cdots b(f^{\ell(|s|)-1}(s))$$

is a pseudorandom generator (with stretch function ℓ), where $f^{i+1}(x) \stackrel{\text{def}}{=} f(f^i(x))$ and $f^0(x) \stackrel{\text{def}}{=} x$. As a concrete example, consider the permutation $x \mapsto x^2 \bmod N$, where N is the product of two primes each congruent to $3 \pmod 4$. We have $G_N(s) = \text{lsb}(s) \cdot \text{lsb}(s^2 \bmod N) \cdots \text{lsb}(s^{2^{\ell(|s|)-1}} \bmod N)$, where $\text{lsb}(x)$ is the least significant bit of x (which by [1, 118] is a hard-core of the modular squaring function). We note that for any one-way permutation f', the inner-product mod 2 of x and r is a hard-core of $f(x, r) = (f'(x), r)$ [67]. Thus, using any one-way permutation, we can easily construct pseudorandom generators.

The iteration paradigm is even more beneficial when one has a hard-core function rather than a hard-core predicate: h is called a *hard-core function* of f if h is easy to evaluate but, for a random $x \in \{0,1\}^*$, the distribution $f(x) \cdot h(x)$ is pseudorandom. (Note that a hard-core predicate is a special case.) Using a hard-core function h for f, we obtain the pseudorandom generator $G'(s) = h(s) \cdot h(f(s)) \cdot h(f^2(s)) \cdots$. In particular, assuming the intractability of the subset sum problem (for suitable densities) this paradigm was used in [83] to construct very efficient pseudorandom generators. Alternatively, encouraged by the results in [1, 80], we conjecture that the first $n/2$ least significant bits of the argument constitute a hard-core function of the modular squaring function for n-bit long moduli. This conjecture yields an efficient pseudorandom generator: $G'_N(s) = \text{LSB}_N(s) \cdot \text{LSB}_N(s^2 \bmod N) \cdot \text{LSB}_N(s^4 \bmod N) \cdots$, where $\text{LSB}_N(x)$ denotes the $0.5 \log_2 N$ least significant bits of x.

A plausibility result: By [79], pseudorandom generators exist if one-way functions exist. Unlike the construction of pseudorandom generators from one-way permutations, the known construction of pseudorandom generators from *arbitrary* one-way functions has no practical significance. It is indeed an important open problem to provide an alternative construction which may be practical and still utilize an *arbitrary* one-way function.

Pseudorandom Functions

Pseudorandom generators allow to efficiently generate long pseudorandom sequences from short random seeds. Pseudorandom functions (defined below) are even more powerful: They allow efficient direct access to a huge pseudorandom sequence (which is not feasible to scan bit-by-bit). Put in other words, pseudorandom functions can replace truly random functions in any application where the function is used in a black-box fashion (i.e., the adversary may obtain the value of the function at arguments of its choice but is not able to evaluate the function by itself).[4]

Definition 4 (pseudorandom functions [63]): *A* pseudorandom function *is an efficient* (deterministic) *algorithm which given an n-bit* seed, *s, and an n-bit* argument, *x, returns an n-bit string, denoted $f_s(x)$, so that it is infeasible to distinguish the responses of f_s, for a uniformly chosen s, from the responses of a truly random function.*

That is, the distinguisher is given access to a function and is required to distinguish a random function $f:\{0,1\}^n \mapsto \{0,1\}^n$ from a function chosen uniformly in $\{f_s : s \in \{0,1\}^n\}$. We stress that in the latter case the distinguisher is NOT given the description of the function f_s (i.e., the seed s), but rather may obtain the value of f_s on any n-bit string of its choice.[5]

Pseudorandom functions are a very useful cryptographic tool (cf., [64, 60] and Section 5): One may first design a cryptographic scheme assuming that the legitimate users have black-box access to a random function, and next implement the random function using a pseudorandom function.

From pseudorandom generators to pseudorandom functions [63]: Let G be a pseudorandom generator with stretching function $\ell(n) = 2n$, and let $G_0(s)$ (resp., $G_1(s)$) denote the first (resp., last) n bits in $G(s)$ where $s \in \{0,1\}^n$. We define the function ensemble $\{f_s : \{0,1\}^{|s|} \mapsto \{0,1\}^{|s|}\}$, where $f_s(\sigma_{|s|} \cdots \sigma_2\sigma_1) = G_{\sigma_{|s|}}(\cdots G_{\sigma_2}(G_{\sigma_1}(s))\cdots)$. This ensemble is pseudorandom.

An alternative construction of pseudorandom functions has been suggested in [102].

4 Zero-Knowledge

Loosely speaking, zero-knowledge proofs are proofs which yield nothing beyond the validity of the assertion. That is, a verifier obtaining such a proof only gains conviction in the validity of the assertion. Using the simulation paradigm this requirement is stated by postulating that anything that is feasibly computable from a zero-knowledge proof is also feasibly computable from the valid assertion alone. All the above refers to proofs as to interactive and randomized processes. That is, here a proof is a (multi-round) protocol for two parties, called verifier and prover, in which the prover wishes to convince the verifier of the validity of a given assertion. Such an *interactive proof* should allow the prover to convince the verifier of the validity of any true assertion, whereas NO prover strategy may fool the verifier to accept false assertions. Both the above *completeness* and *soundness* conditions should hold with high probability (i.e., a negligible error probability is allowed). The prescribed verifier strategy is required to be efficient. No such requirement is made with respect to the prover strategy; yet we will be interested in "relatively efficient" prover strategies (see below). Zero-knowledge is a property of some prover-strategies. More generally, we consider interactive machines which yield no knowledge while interacting with an arbitrary feasible adversary on a common input taken from a predetermined set (in our case the set of valid assertions).

[4] This is different from the *Random Oracle Model* of [12].

[5] Typically, the distinguisher stands for an adversary that attacks a system which uses a pseudorandom function. The values of the function on arguments of the adversary's choice are obtained from the legitimate users of the system who, *unlike the adversary*, know the seed s. The definition implies that the adversary will not be more successful in its attack than it could have been if the system was to use a truly random function. Needless to say that the latter system is merely a *Gedanken Experiment* (it cannot be implemented since it is infeasible to even store a truly random function).

Definition 5 (zero-knowledge [75]): *A strategy A is* zero-knowledge *on inputs from S if, for every feasible strategy B^*, there exists a feasible computation C^* so that the following two probability ensembles are computationally indistinguishable:*

1. $\{(A, B^*)(x)\}_{x \in S} \stackrel{\text{def}}{=}$ *the output of B^* when interacting with A on common input $x \in S$; and*

2. $\{C^*(x)\}_{x \in S} \stackrel{\text{def}}{=}$ *the output of C^* on input $x \in S$.*

Note that whereas A and B^* above are interactive strategies, C^* is a non-interactive computation. The above definition does NOT account for auxiliary information which an adversary may have prior to entering the interaction. Accounting for such auxiliary information is essential for using zero-knowledge proofs as subprotocols inside larger protocols (see [66, 70]). Another concern is that we prefer that the complexity of C^* be bounded as a function of the complexity of B^*. Both concerns are taken care of by a more strict notion of zero-knowledge presented next.

Definition 6 (zero-knowledge, revisited [70]): *A strategy A is* black-box zero-knowledge *on inputs from S if there exists an efficient* (universal) *subroutine-calling algorithm U so that for every feasible strategy B^*, the probability ensembles $\{(A, B^*)(x)\}_{x \in S}$ and $\{U^{B^*}(x)\}_{x \in S}$ are computationally indistinguishable, where U^{B^*} is algorithm U using strategy B^* as a subroutine.*

Note that the running time of U^{B^*} is at most the running-time of U times the running-time of B^*. Actually, the first term may be replaced by the number of times U invokes the subroutine. All known zero-knowledge proofs are in fact black-box zero-knowledge.

A general plausibility result: By [68], assuming the existence of commitment schemes, there exist (black-box) zero-knowledge proofs for membership in any NP-language.[6] Furthermore, the prescribed prover strategy is efficient provided it is given an NP-witness to the assertion to be proven. This makes zero-knowledge a very powerful tool in the design of cryptographic schemes and protocols.

Zero-knowledge as a tool: In a typical cryptographic setting, a user, referred to as A, has a secret and is supposed to take some steps depending on its secret. The question is how can other users verify that A indeed took the correct steps (as determined by A's secret and the publicly known information). Indeed, if A discloses its secret then anybody can verify that it took the correct steps. However, A does not want to reveal its secret. Using zero-knowledge proofs we can satisfy both conflicting requirements. That is, A can prove in zero-knowledge that it took the correct steps. Note that A's claim to having taken the correct steps is an NP-assertion and that A has an NP-witness to its validity (i.e., its secret!). Thus, by the above result, it is possible for A to efficiently prove the correctness of its actions without yielding anything about its secret. (However, in practice one may want to design a specific zero-knowledge proof, tailored to the specific application and so being more efficient, rather than invoking the general result above. Thus, the development of techniques for the construction of such proofs is still of interest – see, for example, [43].)

[6] NP is the class of languages having efficiently verifiable (and short) proofs of membership. That is, L is in NP if there exists a polynomial-time recognizable binary relation R_L and a polynomial ℓ so that $x \in L$ if and only if there exists y so that $|y| \leq \ell(|x|)$ and $(x, y) \in R_L$.

Some Variants

Perfect zero-knowledge arguments: This term captures two deviations from the above definition; the first being a strengthening and the second being a weakening. Perfect zero-knowledge strategies are such for which the ensembles in Definition 5 are identically distributed (rather than computationally indistinguishable). This means that the zero-knowledge clause holds regardless of the computational abilities of the adversary. However, *arguments* (aka *computationally sound proofs*) differ from interactive proofs in having a weaker soundness clause: it is infeasible (rather than impossible) to fool the verifier to accept false assertion (except with negligible probability) [25]. Perfect zero-knowledge arguments for NP were constructed using any one-way permutation [100].

Non-Interactive zero-knowledge proofs [20, 52]: Here the interaction between the prover and the verifier consists of the prover sending a single message to the verifier (as in "classical proofs"). In addition, both players have access to a "random reference string" which is postulated to be uniformly selected. Non-interactive zero-knowledge proofs are useful in applications where one of the parties may be trusted to select the abovementioned reference string (e.g., see Section 5.3). Non-interactive zero-knowledge arguments for NP were constructed using any trapdoor permutation [52, 88].

Zero-knowledge proofs of knowledge [75, 56, 5]: Loosely speaking, a system for proofs of knowledge guarantees that whenever the verifier is convinced that the prover knows X, this X can be efficiently extracted from the prover's strategy. One natural application of (zero-knowledge) proofs of knowledge is for *identification* [56, 51].

Relaxations of Zero-knowledge: Important relaxations of zero-knowledge were presented in [53]. Specifically, in *witness indistinguishable* proofs it is infeasible to tell which NP-witness to the assertion the prover is using. Unlike zero-knowledge proofs, this notion is closed under parallel composition. Furthermore, this relaxation suffices for some applications in which one may originally think of using zero-knowledge proofs.

Part II
Basic Utilities

5 Encryption

Both Private-Key and Public-Key encryption schemes consists of three efficient algorithms: *key generation*, *encryption* and *decryption*. The difference between the two types is reflected in the definition of security – the security of a public-key encryption scheme should hold also when the adversary is given the encryption key, whereas this is not required for private-key encryption scheme. Thus, public-key encryption schemes allow each user to broadcast its encryption key so that any user may send it encrypted messages (without needing to first agree on a private encryption-key with the receiver). Below we present definitions of security for private-key encryption schemes. The public-key analogies can be easily derived by considering adversaries which get the encryption key as additional input. (For private-key encryption schemes we may assume, without loss of generality, that the encryption key is identical to the decryption key.)

5.1 Definitions

For simplicity we consider only the encryption of a single message; however this message may be longer than the key (which rules out information-theoretic secrecy [116]). We present two equivalent definitions of security. The first, called *semantic security*, is a computational analogue of Shannon's definition of *perfect secrecy* [116]. The second definition views secure encryption schemes as ones for which it is infeasible to distinguish encryptions of any (known) pair of messages (e.g., the all-zeros message and the all-ones message). The latter definition is technical in nature and is referred to as *indistinguishability of encryptions*.

We stress that the definitions presented below go way beyond saying that it is infeasible to recover the plaintext from the ciphertext. The latter statement is indeed a minimal requirement from a secure encryption scheme, but we claim that it is way too weak a requirement: An encryption scheme is typically used in applications where obtaining specific partial information on the plaintext endangers the security of the application. When designing an application-independent encryption scheme, we do not know which partial information endangers the application and which does not. Furthermore, even if one wants to design an encryption scheme tailored to one's own specific applications, it is rare (to say the least) that one has a precise characterization of all possible partial information which endanger these applications. Thus, we require that it is infeasible to obtain any information about the plaintext from the ciphertext. Furthermore, in most applications the plaintext may not be uniformly distributed and some a-priori information regarding it is available to the adversary. We require that the secrecy of all partial information is preserved also in such a case. That is, even in presence of a-priori information on the plaintext, it is infeasible to obtain any (new) information about the plaintext from the ciphertext (beyond what is feasible to obtain from the a-priori information on the plaintext). The definition of semantic security postulates all of this. The equivalent definition of indistinguishability of encryptions is useful in demonstrating the security of candidate constructions as well as for arguing about their usage as part of larger protocols.

The actual definitions: In both definitions we consider (feasible) adversaries which obtain, in addition to the ciphertext, also auxiliary information which may depend on the potential plaintexts (but not on the key). By $E(x)$ we denote the distribution of encryptions of x, when the key is selected at random. To simplify the exposition, let us assume that on security parameter n the key generation produces a key of length n, whereas the scheme is used to encrypt messages of length n^2.

Definition 7 (semantic security (following [74])): *An encryption scheme is* semantically secure *if for every feasible algorithm, A, there exists a feasible algorithm B so that for every two functions $f,h : \{0,1\}^* \mapsto \{0,1\}^*$ and all sequences of pairs, $(X_n, z_n)_{n\in\mathbb{N}}$, where X_n is a random variable ranging over $\{0,1\}^{n^2}$ and $|z_n|$ is of feasible* (in n) *length,*

$$\Pr(A(E(X_n), h(X_n), z_n) = f(X_n)) < \Pr(B(h(X_n), z_n) = f(X_n)) + \mu(n)$$

where μ is a negligible function. Furthermore, the complexity of B should be related to that of A.

What this definition says is that a feasible adversary does not gain anything by looking at the ciphertext. That is, whatever information (captured by the function f) it tries to compute from the ciphertext, can be essentially computed as efficiently from the available a-priori information (captured by the function h). In particular, the ciphertext does not help in (feasibly) computing the least significant bit of the plaintext or any other information regarding the plaintext. This holds for any distribution of plaintexts (captured by the random variable X_n).

Definition 8 (indistinguishability of encryptions (following [74])): *An encryption scheme has* indistinguishable encryptions *if for every feasible algorithm, A, and all sequences of triples, $(x_n, y_n, z_n)_{n\in\mathbb{N}}$, where $|x_n| = |y_n| = n^2$ and $|z_n|$ is of feasible* (in n) *length, the difference*

$$d_A(n) \stackrel{\text{def}}{=} |\Pr(A(E(x_n), z_n) = 1) - \Pr(A(E(y_n), z_n) = 1)|$$

is a negligible function in n.

In particular, z_n may equal (x_n, y_n). Thus, it is infeasible to distinguish the encryptions of any two fix messages such as the all-zero message and the all-ones message.

Probabilistic Encryption: It is easy to see that a secure *public-key* encryption scheme must employ a probabilistic (i.e., randomized) encryption algorithm. Otherwise, given the encryption key as (additional) input, it is easy to distinguish the encryption of the all-zero message from the encryption of the all-ones message. The same holds for *private-key* encryption schemes when considering the security of encrypting several messages (rather than a single message as done above).[7] This explains the linkage between the above robust security definitions and the *randomization paradigm* (discussed below).

5.2 Constructions

It is common practice to use "pseudorandom generators" as a basis for private-key stream ciphers. We stress that this is a very dangerous practice when the "pseudorandom generator" is easy to predict (such as the linear congruential generator or some modifications of it which output a constant fraction of the bits of each resulting number – see [24, 58]). However, this common practice becomes sound provided one uses pseudorandom generators (as defined in Section 3).

[7] Here, for example, using a deterministic encryption algorithm allows the adversary to distinguish two encryptions of the same message from the encryptions of a pair of different messages.

Private-Key Encryption Scheme based on Pseudorandom Functions: The key generation algorithm consists of selecting a seed, denoted s, for such a function, denoted f_s. To encrypt a message $x \in \{0,1\}^n$ (using key s), the encryption algorithm uniformly selects a string $r \in \{0,1\}^n$ and produces the ciphertext $(r, x \oplus f_s(r))$. To decrypt the ciphertext (r, y) (using key s), the decryption algorithm just computes $y \oplus f_s(r)$. The proof of security of this encryption scheme consists of two steps (suggested as a general methodology in Section 3):

1. Prove that an idealized version of the scheme, in which one uses a uniformly selected function $f:\{0,1\}^n \mapsto \{0,1\}^n$, rather than the pseudorandom function f_s, is secure.

2. Conclude that the real scheme (as presented above) is secure (since otherwise one could distinguish a pseudorandom function from a truly random one).

Note that we could have gotten rid of the randomization if we had allowed the encryption algorithm to be history dependent (e.g., use a counter in the role of r). Furthermore, if the encryption scheme is used for FIFO communication between the parties and both can maintain the counter value then there is no need for the sender to send the counter value.

The randomization paradigm [74]: To demonstrate this paradigm suppose we have a trapdoor one-way permutation, $\{p_\alpha\}_\alpha$, and a hard-core predicate, b, for it.[8] The key generation algorithm consists of selecting at random a permutation p_α together with a trapdoor for it: The permutation (or rather its description) serves as the public-key, whereas the trapdoor serves as the private-key. To encrypt a single bit σ (using public key p_α), the encryption algorithm uniformly selects an element, r, in the domain of p_α and produces the ciphertext $(p_\alpha(r), \sigma \oplus b(r))$. To decrypt the ciphertext (y, τ) (using the private key), the decryption algorithm just computes $\tau \oplus b(p_\alpha^{-1}(y))$ (where the inverse is computed using the trapdoor (i.e., private-key)). The above scheme is quite wasteful in bandwidth; however, the paradigm underlying its construction is valuable in practice. For example, it is certainly better to randomly pad messages (say using padding equal in length to the message) before encrypting them using RSA than to employ RSA on the plain message. Such a heuristic could be placed on firm grounds if a conjecture analogous to the one mentioned in Section 3 is supported. That is, assume that the first $n/2$ least significant bits of the argument constitute a hard-core function of RSA with n-bit long moduli. Then, encrypting $n/2$-bit messages by padding the message with $n/2$ random bits and applying RSA (with an n-bit moduli) on the result constitutes a secure public-key encryption system, hereafter referred to as Randomized RSA.

An alternative public-key encryption scheme is presented in [21]. The encryption scheme augments the construction of a pseudorandom generator, given in Section 3, as follows. The key-generation algorithm consists of selecting at random a permutation p_α together with a trapdoor. To encrypt the n-bit string x (using public key p_α), the encryption algorithm uniformly selects an element, s, in the domain of p_α and produces the ciphertext $(p_\alpha^n(s), x \oplus G_\alpha(s))$, where $G_\alpha(s) = b(s) \cdot b(p_\alpha(s)) \cdots b(p_\alpha^{n-1}(s))$. (We use the notation $p_\alpha^{i+1}(x) = p_\alpha(p_\alpha^i(x))$ and $p_\alpha^{-(i+1)}(x) = p_\alpha^{-1}(p_\alpha^{-i}(x))$.) To decrypt the ciphertext (y, z) (using the private key), the decryption algorithm first recovers $s = p_\alpha^{-n}(y)$ and then outputs $z \oplus G_\alpha(s)$.

Assuming that factoring Blum Integers (i.e., products of two primes each congruent to 3 (mod 4)) is hard, one may use the modular squaring function in role of the trapdoor permutation above (see [21, 1, 118, 57]). This yields a secure public-key encryption scheme (depicted in Figure 1) with efficiency comparable to that of RSA. Recall that RSA itself is not secure (as it

[8] Hard-core predicates are defined in Section 3. Recall that by [67], every trapdoor permutation can be modified into one having a hard-core predicate.

private-key: Two $n/2$-bit primes p, q, each congruent to $3 \pmod 4$.

public-key: Their product $N \stackrel{\text{def}}{=} pq$.

encryption of message $x \in \{0,1\}^n$:

1. Uniformly select $s_0 \in \{1, ..., N\}$.
2. For $i = 1, .., n+1$, compute $s_i \leftarrow s_{i-1}^2 \bmod N$ and $\sigma_i = \text{lsb}(s_i)$.

The ciphertext is (s_{n+1}, y), where $y = x \oplus \sigma_1 \sigma_2 \cdots \sigma_n$.

decryption of the ciphertext (r, y). Let $d = 2^{-n} \bmod \phi(N)$ [precomputed].

1. Let $s_1 \leftarrow r^d \bmod N$.
2. For $i = 1, .., n$, compute $\sigma_i = \text{lsb}(s_i)$ and $s_{i+1} \leftarrow s_i^2 \bmod N$.

The plaintext is $y \oplus \sigma_1 \sigma_2 \cdots \sigma_n$.

Figure 1: The Blum–Goldwasser Public-Key Encryption Scheme [21].

employs a deterministic encryption algorithm), whereas Randomized RSA (defined above) is not known to be secure under standard assumption such as intractability of factoring (or of inverting the RSA function).[9]

5.3 Beyond eavesdropping security

The above definitions refer only to a "passive" attack in which the adversary merely eavesdrops on the line over which ciphertexts are being sent. Stronger types of attacks, culminating in the so-called Chosen Ciphertext Attack, may be possible in various applications. In such an attack, the adversary may obtain the plaintexts of ciphertexts of its choice (as well as ciphertexts of plaintexts of its choice) and its task is to obtain information about the plaintext of a different ciphertext. Clearly, the private-key encryption scheme based on pseudorandom functions (described above) is secure also against such attacks. Public-key encryption schemes secure against Chosen Ciphertext Attacks can be constructed, assuming the existence of trapdoor permutations and utilizing non-interactive zero-knowledge proofs [106] (which can be constructed under this assumption [52]).

Another issue is the *non-malleability* of the encryption scheme, considered in [47]. Here one requires that it should be infeasible for an adversary, given a ciphertext, to produce a valid ciphertext for a related plaintext. For example, given a ciphertext of a plaintext of the form $1x$, it should be infeasible to produce a ciphertext to the plaintext $0x$. It is easy to turn a private-key encryption scheme into a non-malleable one, by using a message authentication scheme on top. Non-malleable public-key encryption schemes are known to exist assuming the existence of trapdoor permutation [47].

6 Signatures

Again, there are private-key and public-key versions both consisting of three efficient algorithms: *key generation*, *signing* and *verification*. (Private-key signature schemes are commonly referred to

[9] Recall that Randomized RSA is secure assuming that the $n/2$ least significant bits constitute a hard-core function for n-bit RSA moduli. We only know that the $O(\log n)$ least significant bits constitute a hard-core function for n-bit moduli [1].

as *message authentication schemes* or *codes* (MAC).) The difference between the two types is again reflected in the definition of security. This difference yields different functionality (even more than in the case of encryption): Public-key signature schemes (hereafter referred to as signature schemes) may be used to produce signatures which are *universally verifiable* (given access to the public-key of the signer). Private-key signature schemes (hereafter referred to as message authentication schemes) are only used to authenticate messages sent among a small set of *mutually trusting* parties (since ability to verify signatures is linked to the ability to produce them). Put in other words, message authentication schemes are used to authenticate information sent between (typically two) parties, and the purpose is to convince *the receiver* that the information was indeed sent by the legitimate sender. In particular, message authentication schemes cannot convince *a third party* that the sender has indeed sent the information (rather than the receiver having generated it by itself). In contrast, public-key signatures can be used to convince third parties: A signature to a document is typically sent to a second party so that in the future this party may (by merely presenting the signed document) convince third parties that the document was indeed generated/sent/approved by the signer.

6.1 Definitions

We consider very powerful attacks on the signature scheme as well as a very liberal notion of breaking it. Specifically, the attacker is allowed to obtain signatures to any message of its choice. One may argue that in many applications such a general attack is not possible (as messages to be signed must have a specific format). Yet, our view is that it is impossible to define a general (i.e., application-independent) notion of admissible messages, and thus a general/robust definition of an attack seems to have to be formulated as suggested here. (Note that at worst, our approach is overly cautious.) Likewise, the adversary is said to be successful if it can produce a valid signature to ANY message for which it has not asked for a signature during its attack. Again, this defines the ability to form signatures to possibly "nonsensical" messages as a breaking of the scheme. Yet, again, we see no way to have a general (i.e., application-independent) notion of "meaningful" messages (so that only forging signatures to them will be consider a breaking of the scheme).

Definition 9 (unforgeable signatures [76]):

- *A* chosen message attack *is a process which on input a verification-key can obtain signatures* (relative to the corresponding signing-key) *to messages of its choice.*
- *Such an attack is said to* succeeds (in existential forgery) *if it outputs a valid signature to a message for which it has* NOT *requested a signature during the attack.*
- *A signature scheme is* secure (or unforgeable) *if every* feasible *chosen message attack succeeds with at most negligible probability.*

We stress that *plain* RSA (alike plain versions of Rabin's scheme [111] and DSS [98]) is not secure under the above definition. However, it may be secure if the message is "randomized" before RSA (or the other schemes) is applied (cf., [15]). Thus, the randomization paradigm (see Section 5) seems pivotal here too.

6.2 Constructions

Message authentication schemes can be constructed using pseudorandom functions (see [64] or the better constructions in [10, 9, 3]). However, as noted in [4], an *extensive* usage of pseudorandom functions seem an overkill for achieving message authentication, and more efficient schemes may be obtained based on other cryptographic primitives. We mention two approaches:

1. *Fingerprinting* the message using a scheme which is *secure against forgery provided that the adversary does not have access to the scheme's outcome* (e.g., using Universal Hashing [33]), and "*hiding*" the result using a *non-malleable* scheme (e.g., a private-key encryption or a pseudorandom function). (Non-malleability is not required in certain cases; see [119].)

2. *Hashing* the message *using a collision-free scheme* (cf., [41, 42]), and *authenticating* the result using a MAC which operates on (short) fixed-length strings [4].

Three central paradigms in the construction of *signature schemes* are the "refreshing" of the "effective" signing-key, the usage of an "authentication tree" and the "hashing paradigm".

The refreshing paradigm [76]: To demonstrate this paradigm, suppose we have a signature scheme which is robust against a "random message attack" (i.e., an attack in which the adversary only obtains signatures to randomly chosen messages). Further suppose that we have a *one-time* signature scheme (i.e., a signature scheme which is secure against an attack in which the adversary obtains a signature to a single message of its choice). Then, we can obtain a secure signature scheme as follows: When a new message is to be signed, we generate a new random signing-key for the one-time signature scheme, use it to sign the message, and sign the corresponding (one-time) verification-key using the fixed signing-key of the main signature scheme[10] (which is robust against a "random message attack") [50]. We note that one-time signature schemes (as utilized here) are easy to construct (see, for example [94]).

The tree paradigm [93, 76]: To demonstrate this paradigm, we show how to construct a general signature scheme using only a one-time signature scheme (alas one where an $2n$-bit string can be signed w.r.t an n-bit long verification-key). The idea is to use the initial singing-key (i.e., the one corresponding to the public verification-key) in order to sign/authenticate two new/random verification keys. The corresponding signing keys are used to sign/authenticate four new/random verification keys (two per a signing key), and so on. Stopping after d such steps, this process forms a binary tree with 2^d leaves where each leaf corresponds to an instance of the one-time signature scheme. The signing-keys at the leaves can be used to sign the actual messages, and the corresponding verification-keys may be authenticated using the path from the root. Pseudorandom functions may be used to eliminate the need to store the values of intermediate vertices used in previous signatures [60]. Employing this paradigm and assuming that the RSA function is infeasible to invert, one obtains a secure signature scheme [76, 60] in which the i^{th} message can be signed/verified in time $2\log_2 i$ slower than plain RSA. Using a tree of large fan-in and assuming that RSA is infeasible to invert, one may obtain a secure signature scheme [49] which for reasonable parameters is only 5 times slower than plain RSA (alas uses a much bigger key).[11] We stress

[10] Alternatively, one may generate the one-time key-pair and the signature to its verification-key ahead of time, leading to an "off-line/on-line" signature scheme [50].

[11] This figure refers to signing up-to 1,000,000,000 messages. The scheme requires a universal set of system parameters consisting of 1000–2000 integers of the size of the moduli. We believe that in *some* applications the storage/time trade-off provided by [49] may be preferred over [76, 60].

that plain RSA is not a secure signature scheme, whereas the security of its randomized version (mentioned above) is not known to be reducible to the assumption that RSA is hard to invert.

The hashing paradigm: A common practice is to sign real documents via a two stage process: First the document is hashed into a (relatively) short bit string, and next the basic signature scheme is applied to the resulting string. We note that this heuristic becomes sound provided the hashing function is *collision-free* (as defined in [41]). Collision-free functions can be constructed assuming the intractability of factoring [41]. One may indeed postulate that certain off-the-shelf products (as MD5 or SHA) are collision-free, but such assumptions need to be tested (and indeed may turn out false). We stress that using a hashing scheme in the above two-stage process without evaluating whether it is collision-free is a very dangerous practice.

A useful variant on the above paradigm is the use of *Universal One-Way Hash Functions* (as defined in [105]), rather than the collision-free hashing used above. In such a case a new hash function is selected per each application of the scheme, and the basic signature scheme is applied to both the (succinct) description of the hash function and to the resulting (hashed) string. (In contrast, when using a collision-free hashing function, the same function – the description of which is part of the signer's public-key – is used in all applications.) The advantage of using Universal One-Way Hash Functions is that their security requirement seems weaker than the collision-free condition (e.g., the former may be constructed using any one-way function [115], whereas this is not known for the latter).

A plausibility result: By [105, 115] signature schemes exist if and only if one-way functions exist. Unlike the constructions of signature schemes described above, the known construction of signature schemes from *arbitrary* one-way functions has no practical significance [115]. It is indeed an important open problem to provide an alternative construction which may be practical and still utilize an *arbitrary* one-way function.

7 Cryptographic Protocols

A general framework for casting cryptographic (protocol) problems consists of specifying a random process which maps n inputs to n outputs. The inputs to the process are to be thought of as local inputs of n parties, and the n outputs are their corresponding local outputs. The random process describes the desired functionality. That is, if the n parties were to trust each other (or trust some outside party), then they could each send their local input to the trusted party, who would compute the outcome of the process and send each party the corresponding output. The question addressed in this section is to what extent can this trusted party be "simulated" by the mutually distrustful parties themselves.

7.1 Definitions

For simplicity we consider the special case where the specified process is deterministic and the n outputs are identical. That is, we consider an arbitrary n-ary function and n parties which wish to obtain the value of the function on their n corresponding inputs. Each party wishes to obtain the correct value of the function and prevent any other party from gaining anything else (i.e., anything beyond the value of the function and what is implied by it).

We first observe that (one thing which is unavoidable is that) each party may change its local input before entering the protocol. However, this is unavoidable also when the parties utilize a trusted party. In general, the basic paradigm underlying the definitions of *secure multi-party computations* amounts to saying that situations which may occur in the real protocol, can be simulated in the ideal model (where the parties may employ a trusted party). Thus, the "effective malfunctioning" of parties in secure protocols is restricted to what is postulated in the corresponding ideal model. The specific definitions differ in the specific restrictions and/or requirements placed on the parties in the real computation. This is typically reflected in the definition of the corresponding ideal model – see examples below.

An example – computations with honest majority: Here we consider an ideal model in which any minority group (of the parties) may collude as follows. Firstly this minority shares its original inputs and decided together on replaced inputs[12] to be sent to the trusted party. (The other parties send their respective original inputs to the trusted party.) When the trusted party returns the output, each majority player outputs it locally, whereas the colluding minority may compute outputs based on all they know (i.e., the output and all the local inputs of these parties). A *secure multi-party computation with honest majority* is required to simulate this ideal model. That is, the effect of any feasible adversary which controls a minority of the players in the actual protocol, can be essentially simulated by a (different) feasible adversary which controls the corresponding players in the ideal model. This means that in a secure protocol the effect of each minority group is "essentially restricted" to replacing its own local inputs (independently of the local inputs of the majority players) before the protocol starts, and replacing its own local outputs (depending only on its local inputs and outputs) after the protocol terminates. (We stress that in the real execution the minority players do obtain additional pieces of information; yet in a secure protocol they gain nothing from these additional pieces of information.)

Secure protocols according to the above definition may even tolerate a situation where a minority of the parties aborts the execution. An aborted party (in the real protocol) is simulated by a party (in the ideal model) which aborts the execution either before supplying its input to the trusted party (in which case a default input is used) or after supplying its input. In either case, the majority players (in the real protocol) are able to compute the output although a minority aborted the execution. This cannot be expected to happen when there is no honest majority (e.g., in a two-party computation) [40].

Another example – two-party computations: In light of the above, we consider an ideal model where each of the two parties may "shut-down" the trusted (third) party at any point in time. In particular, this may happen after the trusted party has supplied the outcome of the computation to one party but before it has supplied it to the second. A *secure multi-party computation allowing abort* is required to simulate this ideal model. That is, each party's "effective malfunctioning" in a secure protocol is restricted to supplying an initial input of its choice and aborting the computation at any point in time. We stress that, as above, the choice of the initial input of each party may NOT depend on the input of the other party.

7.2 Constructions

General plausibility results: Assuming the existence of trapdoor permutations, one may provide secure protocols for any two-party computation (allowing abort) [121] as well as for any

[12] Such replacement may be avoided if the local inputs of parties are verifiable by the other parties. In such a case, a party (in the ideal model) has the choice of either joining the execution of the protocol with its correct local input or not join the execution at all (but it cannot join with a replaced local input). Secure protocols simulating this ideal model can be constructed as well.

multi-party computations with honest majority [69]. Thus, a host of cryptographic problems are solvable assuming the existence of trapdoor permutations. As stressed in the case of zero-knowledge proofs, we view these results as asserting that very wide classes of problems are solvable in principle. However, we do not recommend using the solutions derived by these general results in practice. For example, although Threshold Cryptography (cf., [45, 59]) is merely a special case of multi-party computation, it is indeed beneficial to focus on its specifics.

Analogous plausibility results were obtained in a variety of models. In particular, we mention secure computations in the private channels model [17, 35] and in the presence of mobile adversaries [108].

Part III
Concluding Comments

8 Some Notes

On information theoretic secrecy: Most of Modern Cryptography aims at achieving *computational* security; that is, making it infeasible (rather than impossible) for an adversary to break the system. The departure from *information theoretic* secrecy was suggested by Shannon in the very paper which introduced the notion [116]: In an information theoretic secure encryption scheme the private-key must be longer than the total entropy of the plaintexts to be sent using this key. This drastically restricts the applicability of (information-theoretic secure) private-key encryption schemes. Furthermore, notions such as public-key cryptography, pseudorandom generators, and most known cryptographic protocols cannot exist in an information theoretic sense.

On the need for and choice of assumptions: As stated in Section 2, most of Modern Cryptography is based on computational difficulty. Intuitively, this is an immediate consequence of the fact that Modern Cryptography wish to capitalize on the difference between feasible attacks and possible-but-infeasible attacks. Formally, the existence of one-way functions has been shown to be a necessary condition for the existence of secure private-key encryption [82], pseudorandom generators [90], digital signatures [115], "non-trivial" zero-knowledge proofs [107], and various basic protocols [82].

As we need assumptions anyhow, why not assume what we want? Well, first we need to know what we want. This calls for a clear definition of complex security concerns – an non-trivial issue which is discussed at length in previous sections. However, once a definition is derived how can we know that it can at all be met? The way to demonstrate that a definition is viable (and so the intuitive security concern can be satisfied at all) is to construct a solution based on a *better understood* assumption. For example, looking at the definition of zero-knowledge proofs [75], it is not a-priori clear that such proofs exists in a non-trivial sense. The non-triviality of the notion was demonstrated in [75] by presenting a zero-knowledge proof system for statements, regarding Quadratic Residuosity, which are believed to be hard to verify (without extra information). Furthermore, in contrary to prior beliefs, it was shown in [68] that the existence of commitment schemes[13] implies that any NP-statement can be proven in zero-knowledge. Thus, statements, which were not known at all to hold (and even believed to be false), where shown to hold by reduction to widely believed assumptions (without which most of Modern Cryptography collapses anyhow). Furthermore, reducing the solution of a new task to the assumed security of a well-known primitive typically means providing a construction which using the known primitive solves the new task. This means that we do not only know (or assume) that the new task is solvable but rather have a solution based on a primitive which, being well-known, typically has several candidate implementations. More on this subject below.

On the meaning of asymptotic results: Asymptotic analysis is a major simplifying convention. It allows to disregard specifics like the model of computation and to focus on the essentials of the problem at hand. Further simplification is achieved by identifying efficient computations with polynomial-time computations, and more importantly by identifying infeasible computations with

[13] Consequently, it was shown how to construct commitment schemes based on any pseudorandom generator [99], and that the latter exists if one-way functions exist [79].

ones which are not implementable in polynomial-time. However, none of these conventions is really essential for the theory discussed in this essay.[14]

As stated in Section 2, all know results (referring to computational complexity) consists of an explicit construction in which a complex primitive is implemented based on a simpler one. The claim of security in many papers merely states that if the resulting (complex) primitive can be broken in polynomial-time then so can the original (simpler) primitive. However, all papers provide an explicit construction showing how to use any breaking algorithm for the resulting primitive in order to obtain a breaking algorithm for the original primitive. This transformation does not depend on the running-time of the first algorithm; it typically uses the first algorithm as a black-box. Thus, the running-time of the resulting breaking algorithm (for the simpler primitive) is explicitly bounded in terms of the running-time of the given breaking algorithm (for the complex primitive). This means that for each of these results, one can instantiate the resulting (complex) scheme for any desired value of the security parameter, make a concrete assumption regarding the security of the underlying (simpler) primitive, and derive a concrete estimate of the security of the proposed implementation of the complex primitive.

The applicability of a specific theoretical result depends on the complexity of the construction and the relation between the security of the resulting scheme and the quantified intractability assumption. Some of these results seem applicable in practice, some only offer useful paradigm/techniques, and other only state the plausibility of certain results. In the latter cases it is indeed the task of the theory community to work towards the improvement of these results. In fact, many improvements of this type have been achieved in the past (and we hope to see more in the future). Following are some examples:

- A plausibility result of Yao (commonly attributed to [120]) on the existence of hard-core predicates, assuming the existence of one-way permutations, was replaced by a practical construction of hard-core predicates for any one-way functions [67].
- A plausibility result of Yao (commonly attributed to [120]) by which any weak one-way permutation can be transformed into an ordinary one-way permutation was replaced by an efficient transformation of weak one-way permutation into ordinary one-way permutation [65].
- A plausibility result of [68] by which one may construct Verifiable Secret Sharing schemes (cf., [39]), using any one-way function, was replaced by an efficient construction the security of which is based on DLP [54]. In general, many concrete problems which are solvable in principle (by the plausibility results of [68, 121, 69]) were given efficient solutions.

Forget the result, use its ideas: As stated above, some theoretical results are not directly applicable in practice. Still, in many cases these results utilize ideas which may be of value in practice. Thus, if you know (by a theoretical result) that a problem is solvable in principle, but the known construction is not applicable for your purposes, you may try to utilize some of its underlying ideas when trying to come-up with an alternative solution tailored for your own purposes. We note that the underling ideas are at least as likely to appear in the proof of security as in the construction itself.

The choice of assumptions, revisited. When constructing a solution to a cryptographic problem one may have a choice of which building blocks to use (e.g., one-way functions or pseudorandom

[14] As long as the notions of efficient and feasible computation are sufficiently robust and rich. For example, they should be closed under various functional compositions and should allow computations such as RSA.

functions). In a coarse sense these tools may look equivalent (e.g., one exists if and only if the other exists), but when deciding which to use in practice one should consider the actual level of security attributed to each of them and the "cost" of using each of them as a building block in a particular construction. In the latter term ("cost") we mean the relationship of the security of the building block to the security of the resulting solution. For further discussion the reader is referred to [3, Sec. 1.5]. *Turning the table around*, if we note that a specific primitive provides good security, when used as a building block in many constructions, then this may serve as incentive to focus attention on the implementation of this primitive. The last statement should be understood both as referring to the theory and practice of cryptography. For example, it is our opinion that the industry should focus on constructing fixed-length-key pseudorandom functions rather than on constructing fixed-length-key pseudorandom permutations (or, equivalently, private-key block ciphers).[15]

Security as a quantity rather than a quality: From the above it should be clear that our notions of security are quantitative in nature. They refers to the minimal amount of work required to break the system (as a function of the security parameter). Thus alternative constructions for the same task may (and need to) be compared based on the security they provide. This can be done whenever the underlying assumption are compareable.

"Too cautious" definitions: As stated in Sections 5 and 6, our definitions seem "too cautious" in the sense that they also imply things which may not matter in practice. This is an artifact of our approach to security which requires that the adversary gains *nothing* (rather than "gains nothing we care about") by its malicious actions. We stress two advantages of our approach. First it yields application-independent notions of security (since the notion of a "gain we care about" is application-dependent). Secondly, even when having a specific application in mind, it is close to impossible to come-up with a precise characterization of the set of "gains we care about". Thus, even in the latter case, our approach of depriving the adversary from *any* gain seems to be the best way to go.

"Provable Security": Some of the papers discussed in this essay use the term "provable security". The term is supposed to reflect the fact that these papers only make well-defined technical claims and that proofs of these claims are given or known to the authors. Specifically, whenever a term such as "security" is used, the paper offers or refers to a rigorous definition of the term (and the authors wish to stress this fact in contrast to prior papers where the term was used as an undefined intuitive phrase). We personally object to this terminology since it suggests the possibility that there can be technical claims[16] which are well-defined and others which are not, and among the former some can be stated even when no proof is known. This view is wrong: A technical claim must always be well-defined, and it must always have a proof (otherwise it is a conjecture – not a claim). There is room for non-technical claims, but these claims should be stated as opinions and such. In particular, a technical claim referring to security must always refer to a rigorous definition of security and the person making this claim must always know a proof (or state the claim as a conjecture).

Still, do consider specific attack (but as a last resort). We do realize that sometimes one is faced with a situation where all the paradigms described above offer no help. A typical example

[15] Not to mention that the latter can be efficiently constructed from the former [92, 103].
[16] We refer to theorems, lemmas, propositions and such.

occurs when designing an "atomic" cryptographic primitive (e.g., a one-way function). The first thing we suggest in such a case is to formulate precise specifications/assumptions regarding the security of this primitive. Once this is done, one may need to turn to ad-hoc methods for trying to test these assumptions (i.e., if the known attack schemes fail then one gains some confidence in the validity of the assumptions). For example, if we were to invent RSA today then we would have postulated that it is a trapdoor permutation. To evaluate the validity of our conjecture, we would have noted (as Rivest, Shamir and Adleman did in [114, Sec. IX]) that known algorithms for factoring are infeasible for reasonable values of the security parameter, and that there seems to be no other way to invert the function.

9 Two Suggestions for Future Research

A very important direction for future research consists of trying to "upgrade" the utility of some of the constructions mentioned above. In particular, we have mentioned four plausibility results: two referring to the construction of pseudorandom generators and signature schemes and two referring to the construction of zero-knowledge proofs and multi-party protocols. For the former two results, we see no fundamental reason why the corresponding constructions can not be replaced by reasonable ones (i.e., providing very efficient constructions of pseudorandom generators and signature schemes based on *arbitrary* one-way functions). Furthermore, we believe that working towards this goal may yield new and useful paradigms (which may be applicable in practice regardless of these results). As for the latter general plausibility results (i.e., the construction of zero-knowledge proofs and multi-party protocols), here there seem to be little hope for a result which may both maintain the generality of the results in [68, 121, 69] and yield practical solutions for each specific task. However, we believe that there is work to be done towards the development of additional paradigms and techniques which may be useful in the construction of schemes for specific tasks.

Another important direction is to provide results and/or develop techniques for guaranteeing that individually-secure protocols remain secure when many copies of them are run in parallel and, furthermore, obliviously of one another. Although some negative results are known [66], they only rule out specific approaches (such as the naive false conjecture that ANY zero-knowledge proof maintains its security when executed twice in parallel).

10 Some Suggestions for Further Reading

The intention of these suggestions is NOT to provide a scholarly account of the due credits but rather to provide sources for further reading. Thus, our main criteria is the readability of the text (not its novelty). The recommendations are arranged by subjects.

Computational Hardness, One-Way Functions, Pseudorandom Generators and Zero-Knowledge: For these, our favorite source is our own text [62].

Encryption Schemes: A good motivating discussion appears in [74]. The definitional treatment in [61] is the one we prefer, although it can be substantially simplified if one adopts non-uniform complexity measures (as done above). Further details on the constructions of public-key encryption schemes (sketched above) can be found in [74, 61] and [21, 1], respectively. For discussion of Non-Malleable Cryptography, which actually transcends the domain of encryption, see [47].

Signature Schemes: For a definitional treatment of *signature schemes* the reader is referred to [76] and [109]. Easy to understand constructions appear in [11, 50, 49]. Variants on the basic model are discussed in [109] and in [34, 86]. For discussion of *message authentication schemes* (MACs) the reader in referred to [4].

General Cryptographic Protocols: This area is both most complex and most lacking good expositions. Our own preference is to refer to [27] for the definitions and to [61] for the constructions. For a very nice brief survey, the reader is referred to [72].

New Directions: Incremental Cryptography [6, 7], Realizing the Random Oracle Model [28], Coercibility [30, 29], sharing of cryptographic objects [45, 44, 59], Private Information Retrieval [38, 37, 89], Cryptanalysis by induced faults [23], Visual Cryptography [104, 101] and many others.

Acknowledgments

I wish to thank Ran Canetti, Shafi Goldwasser and Hugo Krawczyk for helpful discussions. Special thanks to Hugo for carefully reading and commenting on an early draft.

Thanks also to Mihir Bellare, Gilles Brassard, Cynthia Dwork and Moni Naor for comments and corrections regarding previous versions.

References

[1] W. Alexi, B. Chor, O. Goldreich and C.P. Schnorr. RSA/Rabin Functions: Certain Parts are As Hard As the Whole. *SIAM Journal on Computing*, Vol. 17, April 1988, pages 194–209.

[2] D. Beaver. Foundations of Secure Interactive Computing. In *Crypto91*, Springer-Verlag Lecture Notes in Computer Science (Vol. 576), pages 377–391.

[3] M. Bellare, R. Canetti and H. Krawczyk. Pseudorandom functions Revisited: The Cascade Construction and its Concrete Security. In *37th IEEE Symposium on Foundations of Computer Science*, pages 514–523, 1996.

[4] M. Bellare, R. Canetti and H. Krawczyk. Keying Hash Functions for Message Authentication. In *Crypto96*, Springer Lecture Notes in Computer Science (Vol. 1109), pages 1–15.

[5] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *Crypto92*, Springer-Verlag Lecture Notes in Computer Science (Vol. 740), pages 390–420.

[6] M. Bellare, O. Goldreich and S. Goldwasser. Incremental Cryptography: the Case of Hashing and Signing. In *Crypto94*, Springer-Verlag Lecture Notes in Computer Science (Vol. 839), pages 216–233, 1994.

[7] M. Bellare, O. Goldreich and S. Goldwasser. Incremental Cryptography and Application to Virus Protection. In *27th ACM Symposium on the Theory of Computing*, pages 45–56, 1995.

[8] M. Bellare, S. Goldwasser and D. Micciancio. "Pseudo-random" Number Generation within Cryptographic Algorithms: the DSS Case. In *Crypto97*, Springer Lecture Notes in Computer Science (Vol. 1294).

[9] M. Bellare, R. Guerin and P. Rogaway. XOR MACs: New Methods for Message Authentication using Finite Pseudorandom Functions. In *Crypto95*, Springer-Verlag Lecture Notes in Computer Science (Vol. 963), pages 15–28.

[10] M. Bellare, J. Kilian and P. Rogaway. The Security of Cipher Block Chaining. In *Crypto94*, Springer-Verlag Lecture Notes in Computer Science (Vol. 839), pages 341–358.

[11] M. Bellare and S. Micali. How to Sign Given Any Trapdoor Function. *Journal of the ACM*, Vol. 39, pages 214–233, 1992.

[12] M. Bellare and P. Rogaway. Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. In *1st Conf. on Computer and Communications Security*, ACM, pages 62–73, 1993.

[13] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Crypto93*, Springer-Verlag Lecture Notes in Computer Science (Vol. 773), pages 232–249, 1994.

[14] M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *27th ACM Symposium on the Theory of Computing*, pages 57–66, 1995.

[15] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In *EuroCrypt96*, Springer Lecture Notes in Computer Science (Vol. 1070).

[16] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability. In *20th ACM Symposium on the Theory of Computing*, pages 113–131, 1988.

[17] M. Ben-Or, S. Goldwasser and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *20th ACM Symposium on the Theory of Computing*, pages 1–10, 1988.

[18] L. Blum, M. Blum and M. Shub. A Simple Secure Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*, Vol. 15, 1986, pages 364–383.

[19] M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-Interactive Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, Vol. 20, No. 6, pages 1084–1118, 1991. (Considered the journal version of [20].)

[20] M. Blum, P. Feldman and S. Micali. Non-Interactive Zero-Knowledge and its Applications. In *20th ACM Symposium on the Theory of Computing*, pages 103–112, 1988. See [19].

[21] M. Blum and S. Goldwasser. An Efficient Probabilistic Public-Key Encryption Scheme which hides all partial information. In *Crypto84*, Lecture Notes in Computer Science (Vol. 196) Springer-Verlag, pages 289–302.

[22] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, Vol. 13, pages 850–864, 1984.

[23] D. Boneh, R. DeMillo and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *EuroCrypt97*, Springer Lecture Notes in Computer Science (Vol. 1233), pages 37–51, 1997.

[24] J.B. Boyar. Inferring Sequences Produced by Pseudo-Random Number Generators. *Journal of the ACM*, Vol. 36, pages 129–141, 1989.

[25] G. Brassard, D. Chaum and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Science*, Vol. 37, No. 2, pages 156–189, 1988.

[26] G. Brassard and C. Crépeau. Zero-Knowledge Simulation of Boolean Circuits. In *Crypto86*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 223–233, 1987.

[27] R. Canetti. *Studies in Secure Multi-Party Computation and Applications.* Ph.D. Thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, June 1995.
Available from from `http://theory.lcs.mit.edu/~tcryptol/BOOKS/ran-phd.html`.

[28] R. Canetti. Towards Realizing Random Oracles: Hash Functions that Hide All Partial Information. In *Crypto97*, Springer Lecture Notes in Computer Science (Vol. 1294).

[29] R. Canetti, C. Dwork, M. Naor and R. Ostrovsky. Deniable Encryption. In *Crypto97*, Springer Lecture Notes in Computer Science (Vol. 1294).

[30] R. Canetti and R. Gennaro. Incoercible Multiparty Computation. In *37th IEEE Symposium on Foundations of Computer Science*, pages 504–513, 1996.

[31] R. Canetti, S. Halevi and A. Herzberg. How to Maintain Authenticated Communication in the Presence of Break-Ins. In *16th ACM Symposium on Principles of Distributed Computing*, 1997.

[32] R. Canetti and A. Herzberg. Maintaining Security in the Presence of Transient Faults. In *Crypto94*, Springer-Verlag Lecture Notes in Computer Science (Vol. 839), pages 425–439.

[33] L. Carter and M. Wegman. Universal Hash Functions. *Journal of Computer and System Science*, Vol. 18, 1979, pages 143–154.

[34] D. Chaum. Blind Signatures for Untraceable Payments. In *Crypto82*, Plenum Press, pages 199–203, 1983.

[35] D. Chaum, C. Crépeau and I. Damgård. Multi-party unconditionally Secure Protocols. In *20th ACM Symposium on the Theory of Computing*, pages 11–19, 1988.

[36] D. Chaum, A. Fiat and M. Naor. Untraceable Electronic Cash. In *Crypto88*, Springer-Verlag Lecture Notes in Computer Science (Vol. 403), pages 319–327.

[37] B. Chor and N. Gilboa. Computationally Private Information Retrieval. In *29th ACM Symposium on the Theory of Computing*, pages 304–313, 1997.

[38] B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan, Private Information Retrieval. In *36th IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[39] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In *26th IEEE Symposium on Foundations of Computer Science*, pages 383–395, 1985.

[40] R. Cleve. Limits on the Security of Coin Flips when Half the Processors are Faulty. In *18th ACM Symposium on the Theory of Computing*, pages 364–369, 1986.

[41] I. Damgård. Collision Free Hash Functions and Public Key Signature Schemes. In *EuroCrypt87*, Springer-Verlag, Lecture Notes in Computer Science (Vol. 304), pages 203–216.

[42] I. Damgård. A Design Principle for Hash Functions. In *Crypto89*, Springer-Verlag Lecture Notes in Computer Science (Vol. 435), pages 416–427.

[43] I. Damgård, O. Goldreich, T. Okamoto and A. Wigderson. Honest Verifier vs Dishonest Verifier in Public Coin Zero-Knowledge Proofs. In *Crypto95*, Springer-Verlag Lecture Notes in Computer Science (Vol. 963), pages 325–338, 1995.

[44] A. De-Santis, Y. Desmedt, Y. Frankel and M. Yung. How to Share a Function Securely. In *26th ACM Symposium on the Theory of Computing*, pages 522–533, 1994.

[45] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. In *Crypto89*, Springer-Verlag Lecture Notes in Computer Science (Vol. 435), pages 307–315.

[46] W. Diffie, and M.E. Hellman. New Directions in Cryptography. *IEEE Trans. on Info. Theory*, IT-22 (Nov. 1976), pages 644–654.

[47] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. In *23rd ACM Symposium on the Theory of Computing*, pages 542–552, 1991. Full version available from authors.

[48] C. Dwork, and M. Naor. Pricing via Processing or Combatting Junk Mail. In *Crypto92*, Springer-Verlag Lecture Notes in Computer Science (Vol. 740), pages 139–147.

[49] C. Dwork, and M. Naor. An Efficient Existentially Unforgeable Signature Scheme and its Application. To appear in *Journal of Cryptology*. Preliminary version in *Crypto94*.

[50] S. Even, O. Goldreich and S. Micali. On-line/Off-line Digital signatures. *Journal of Cryptology*, Vol. 9, 1996, pages 35–67.

[51] U. Feige, A. Fiat and A. Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology*, Vol. 1, 1988, pages 77–94.

[52] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String. In *31th IEEE Symposium on Foundations of Computer Science*, pages 308–317, 1990. To appear in *SIAM Journal on Computing*.

[53] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd ACM Symposium on the Theory of Computing*, pages 416–426, 1990.

[54] P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *28th IEEE Symposium on Foundations of Computer Science*, pages 427–437, 1987.

[55] A. Fiat. Batch RSA. *Journal of Cryptology*, Vol. 10, 1997, pages 75–88.

[56] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solution to Identification and Signature Problems. In *Crypto86*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 186–189, 1987.

[57] R. Fischlin and C.P. Schnorr. Stronger Security Proofs for RSA and Rabin Bits. In *EuroCrypt97*, Springer Lecture Notes in Computer Science (Vol. 1233), pages 267–279, 1997.

[58] A.M. Frieze, J. Håstad, R. Kannan, J.C. Lagarias, and A. Shamir. Reconstructing Truncated Integer Variables Satisfying Linear Congruences. *SIAM Journal on Computing*, Vol. 17, pages 262–280, 1988.

[59] P.S. Gemmell. An Introduction to Threshold Cryptography. In *CryptoBytes*, RSA Lab., Vol. 2, No. 3, 1997.

[60] O. Goldreich. Two Remarks Concerning the GMR Signature Scheme. In *Crypto86*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 104–110, 1987.

[61] O. Goldreich. *Lecture Notes on Encryption, Signatures and Cryptographic Protocol*. Spring 1989. Available from `http://theory.lcs.mit.edu/~oded/ln89.html`

[62] O. Goldreich. *Foundation of Cryptography – Fragments of a Book*. February 1995. Available from `http://theory.lcs.mit.edu/~oded/frag.html`

[63] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, Vol. 33, No. 4, pages 792–807, 1986.

[64] O. Goldreich, S. Goldwasser, and S. Micali. On the Cryptographic Applications of Random Functions. In *Crypto84*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 276–288, 1985.

[65] O. Goldreich, R. Impagliazzo, L.A. Levin, R. Venkatesan, and D. Zuckerman. Security Preserving Amplification of Hardness. In *31st IEEE Symposium on Foundations of Computer Science*, pages 318–326, 1990.

[66] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, Vol. 25, No. 1, February 1996, pages 169–192.

[67] O. Goldreich and L.A. Levin. Hard-core Predicates for any One-Way Function. In *21st ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[68] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, Vol. 38, No. 1, pages 691–729, 1991. See also preliminary version in *27th IEEE Symposium on Foundations of Computer Science*, 1986.

[69] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th ACM Symposium on the Theory of Computing,* pages 218–229, 1987.

[70] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology*, Vol. 7, No. 1, pages 1–32, 1994.

[71] O. Goldreich and R. Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *Journal of the ACM*, Vol. 43, 1996, pages 431–473.

[72] S. Goldwasser. Fault Tolerant Multi Party Computations: Past and Present. In *16th ACM Symposium on Principles of Distributed Computing*, 1997. Also available from `http://www.cs.cornell.edu/Info/People/chandra/podc97/newProgram.html`.

[73] S. Goldwasser and L.A. Levin. Fair Computation of General Functions in Presence of Immoral Majority. In *Crypto90*, Springer-Verlag Lecture Notes in Computer Science (Vol. 537), pages 77–93.

[74] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, Vol. 28, No. 2, pages 270–299, 1984. See also preliminary version in *14th ACM Symposium on the Theory of Computing*, 1982.

[75] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989.

[76] S. Goldwasser, S. Micali, and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, April 1988, pages 281–308.

[77] S. Goldwasser, S. Micali and P. Tong. Why and How to Establish a Private Code in a Public Network. In *23rd IEEE Symposium on Foundations of Computer Science*, 1982, pages 134–144.

[78] S. Goldwasser, S. Micali and A.C. Yao. Strong Signature Schemes. In *15th ACM Symposium on the Theory of Computing*, pages 431–439, 1983.

[79] J. Håstad, R. Impagliazzo, L.A. Levin and M. Luby. Construction of Pseudorandom Generator from any One-Way Function. To appear in *SIAM Journal on Computing*. Preliminary versions by Impagliazzo et. al. in *21st ACM Symposium on the Theory of Computing* (1989) and Håstad in *22nd ACM Symposium on the Theory of Computing* (1990).

[80] J. Håstad, A. Schrift and A. Shamir. The Discrete Logarithm Modulo a Composite Hides $O(n)$ Bits. *Journal of Computer and System Science*, Vol. 47, pages 376–404, 1993.

[81] A. Herzberg, S. Jarecki, H. Krawczyk and M. Yu. Proactive Secret Sharing, or How to Cope with Perpetual Leakage. In *Crypto95*, Springer-Verlag Lecture Notes in Computer Science (Vol. 963), pages 339–352.

[82] R. Impagliazzo and M. Luby. One-Way Functions are Essential for Complexity Based Cryptography. In *30th IEEE Symposium on Foundations of Computer Science*, pages 230-235, 1989.

[83] R. Impagliazzo and M. Naor. Efficient Cryptographic Schemes Provable as Secure as Subset Sum. *Journal of Cryptology*, Vol. 9, 1996, pages 199–216.

[84] R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. In *21st ACM Symposium on the Theory of Computing*, pages 44–61, 1989.

[85] R. Impagliazzo and M. Yung. Direct Zero-Knowledge Computations. In *Crypto87*, Springer-Verlag Lecture Notes in Computer Science (Vol. 293), pages 40–51, 1987.

[86] A. Juels, M. Luby and R. Ostrovsky. Security of Blind Digital Signatures. In *Crypto97*, Springer Lecture Notes in Computer Science (Vol. 1294).

[87] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. In *24th ACM Symposium on the Theory of Computing*, pages 723–732, 1992.

[88] J. Kilian and E. Petrank. An Efficient Non-Interactive Zero-Knowledge Proof System for NP with General Assumptions. To appear in *Journal of Cryptology*.

[89] E. Kushilevitz and R. Ostrovsky. Replication Is NOT Needed: A SINGLE Database, Computational PIR. TR CS0906, Department of Computer Science, Technion, May 1997. To appear in *38th IEEE Symposium on Foundations of Computer Science*, 1997.

[90] L.A. Levin. One-Way Function and Pseudorandom Generators. *Combinatorica*, Vol. 7, pages 357–363, 1987.

[91] M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton University Press, 1996.

[92] M. Luby and C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal on Computing*, Vol. 17, 1988, pages 373–386.

[93] R.C. Merkle. Protocols for public key cryptosystems. In *Proc. of the 1980 Symposium on Security and Privacy.*

[94] R.C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Crypto87*, Springer-Verlag Lecture Notes in Computer Science (Vol. 293), 1987, pages 369-378.

[95] R.C. Merkle. A Certified Digital Signature Scheme. In *Crypto89*, Springer-Verlag Lecture Notes in Computer Science (Vol. 435), pages 218–238.

[96] S. Micali. Fair Public-Key Cryptosystems. In *Crypto92*, Springer-Verlag Lecture Notes in Computer Science (Vol. 740), pages 113–138.

[97] S. Micali and P. Rogaway. Secure Computation. In *Crypto91*, Springer-Verlag Lecture Notes in Computer Science (Vol. 576), pages 392–404.

[98] National Institute for Standards and Technology. Digital Signature Standard (DSS), *Federal Register*, Vol. 56, No. 169, August 1991.

[99] M. Naor. Bit Commitment using Pseudorandom Generators. *Journal of Cryptology*, Vol. 4, pages 151–158, 1991.

[100] M. Naor, R. Ostrovsky, R. Venkatesan and M. Yung. Zero-Knowledge Arguments for NP can be Based on General Assumptions. In *Crypto92*, Springer-Verlag Lecture Notes in Computer Science (Vol. 740), pages 196–214.

[101] M. Naor and B. Pinkas. Visual Authentication and Identification. In *Crypto97*, Springer Lecture Notes in Computer Science (Vol. 1294).

[102] M. Naor and O. Reingold. Synthesizers and their Application to the Parallel Construction of Pseudo-Random Functions. In *36th IEEE Symposium on Foundations of Computer Science*, pages 170–181, 1995.

[103] M. Naor and O. Reingold. On the Construction of Pseudo-Random Permutations: Luby-Rackoff Revisited. In *29th ACM Symposium on the Theory of Computing*, pages 189–199, 1997.

[104] M. Naor and A. Shamir. Visual Cryptography. In *EuroCrypt94*, Springer-Verlag Lecture Notes in Computer Science (Vol. 950), 1995, pages 1–12.

[105] M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Application. *21st ACM Symposium on the Theory of Computing*, 1989, pages 33–43.

[106] M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. In *22nd ACM Symposium on the Theory of Computing*, pages 427-437, 1990.

[107] R. Ostrovsky and A. Wigderson. One-Way Functions are essential for Non-Trivial Zero-Knowledge. In *2nd Israel Symp. on Theory of Computing and Systems (ISTCS93)*, IEEE Computer Society Press, pages 3–17, 1993.

[108] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. In *10th ACM Symposium on Principles of Distributed Computing*, pages 51–59, 1991.

[109] B. Pfitzmann. *Digital Signature Schemes (General Framework and Fail-Stop Signatures)*. Springer Lecture Notes in Computer Science (Vol. 1100), 1996.

[110] M.O. Rabin. Digitalized Signatures. In *Foundations of Secure Computation* (R.A. DeMillo et. al. eds.), Academic Press, 1977.

[111] M.O. Rabin. Digitalized Signatures and Public Key Functions as Intractable as Factoring. MIT/LCS/TR-212, 1979.

[112] M.O. Rabin. How to Exchange Secrets by Oblivious Transfer. Tech. Memo TR-81, Aiken Computation Laboratory, Harvard U., 1981.

[113] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multi-party Protocols with Honest Majority. In *21st ACM Symposium on the Theory of Computing*, pages 73–85, 1989.

[114] R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *CACM*, Vol. 21, Feb. 1978, pages 120–126.

[115] J. Rompel. One-way Functions are Necessary and Sufficient for Secure Signatures. In *22nd ACM Symposium on the Theory of Computing*, 1990, pages 387–394.

[116] C.E. Shannon. Communication Theory of Secrecy Systems. *Bell Sys. Tech. J.*, Vol. 28, pages 656–715, 1949.

[117] A. Shamir. How to Share a Secret. *CACM*, Vol. 22, Nov. 1979, pages 612–613.

[118] U.V. Vazirani and V.V. Vazirani. Efficient and Secure Pseudo-Random Number Generation. *25th IEEE Symposium on Foundations of Computer Science*, pages 458–463, 1984.

[119] M. Wegman and L. Carter. New Hash Functions and their Use in Authentication and Set Equality. *Journal of Computer and System Science*, Vol. 22, 1981, pages 265–279.

[120] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[121] A.C. Yao. How to Generate and Exchange Secrets. In *27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.