How to Prove All NP Statements in Zero-Knowledge and a Methodology of Cryptographic Protocol Design

(Extended Abstract)

Oded Goldreich
Dept. of Computer Sc.
Technion
Haifa, Israel

Silvio Micali
Lab. for Computer Sc.
MIT
Cambridge, MA 02139

Avi Wigderson
Inst. of Math. and CS
Hebrew University
Jerusalem, Israel

ABSTRACT

Under the assumption that encryption functions exist, we show that all languages in NP possess zero-knowledge proofs.

That is, it is possible to demonstrate that a CNF formula is satisfiable without revealing any other property of the formula. In particular, without yielding neither a satisfying assignment nor weaker properties such as whether there is a satisfying assignment in which $x_1 = TRUE$, or whether there is a satisfying assignment in which $x_1 = x_3$ etc.

The above result allows us to prove two fundamental theorems in the field of (two-party and multi-party) cryptographic protocols. These theorems yield automatic and efficient transformations that, given a protocol that is correct with respect to an extremely weak adversary, output a protocol correct in the most adversarial scenario. Thus, these theorems imply powerful methodologies for developing two-party and multi-party cryptographic protocols.

1. INTRODUCTION

A fundamental measure proposed by Goldwasser, Micali and Rackoff [GMR] is that of the amount of knowledge released during an interactive proof. Informally, an interactive proof is a two-party protocol through which one party (*the prover*) can convince his counterparts (*the verifier*) in the validity of some statement concerning a common input. (The prover should be able to do so if and only if the statement is indeed valid.) Loosely speaking, an interactive proof system is called zero-knowledge if whatever the verifier

Work done while the first author was in the Laboratory for Computer Science, MIT, partially supported by an IBM Postdoctoral Fellowship, and NSF Grant DCR-8509905. The second author was supported by NSF Grant DCR-8413577 and an IBM Faculty Development Award. Work done while the third author was in Mathematical Sciences Research Institute, Berkeley.

A.M. Odlyzko (Ed.): Advances in Cryptology - CRYPTO '86, LNCS 263, pp. 171-185, 1987.

could generate in polynomial-time after interacting with the prover, he could also generate in polynomial-time when just told by a trusted oracle that the assertion is indeed valid. In other words, zero-knowledge proofs have the remarkable property of being both convincing and yielding nothing except that the assertion is indeed valid.

Despite their importance, very few examples of (non-trivial [1]) zero-knowledge proofs have been known until recently. Furthermore, all previously known proofs were of languages in $NP \cap Co\text{-}NP$ and heavily relied on special "symmetric" properties of "Number Theoretic" languages. The much more general potential offered by the notion of zero-knowledge proofs has remained immaterialized.

In this extended abstract, we first show how to construct zero-knowledge interactive proofs for every language in NP. This yields an extremely powerful cryptographic tool: the *ability to prove any NP statement in a zero-knowledge manner.* In particular, the generality of this tool allows an untrusted party to prove that he is behaving according to a predetermined protocol, without yielding any of his secrets. We example the power of this tool by three concrete applications. However, the general effect of this result is demonstrated in its generic application as part of a compiler which translates protocols operating in a weak adversary model to protocols which achieve the same goals in the most adversarial environment.

1.1 What is an interactive proof system

It is traditional to view NP as the class of languages whose elements posses short proofs of membership. A "proof that $x \in L$" is a witness w_x such that $P_L(x, w_x)=1$, where P_L is a polynomial-time computable Boolean predicate associated to the language L such that $P_L(x, y)=0$ for all y if x is not in L. The witness must have length polynomial in the length of the input x, but needs not be computable from x in polynomial time. A slightly different point of view is to consider NP as the class of languages L for which a powerful prover may prove membership in L to polynomial-time deterministic verifiers. The interaction between the prover and the verifier, in this case, is trivial: the prover sends a witness ("proof") and the verifier computes for polynomial time to verify that it is indeed a proof.

This formalism was recently generalized by allowing more complex interaction between the prover and the verifier and by allowing the verifier to toss coins and to be convinced by overwhelming statistical evidence [GMR, B]. The motivation of Goldwasser, Micali and Rackoff for this generalization was to *consider the most general manner in which one party can prove theorems to another party, and to study the "amount of knowledge revealed in such interactions"* [GMR]. This generalization is crucial for establishing the non-triviality of the notion of zero-knowledge proofs (see Remarks 4 and 5).

---

1) All languages in BPP have trivial zero-knowledge proofs, in which the prover tells the verifier nothing; the verifier can test membership in BPP languages by himself.

An interactive proof system for a language L is a protocol (i.e. a pair of local programs) for two probabilistic interactive machines called the *prover* and the *verifier*. We denote these predetermined programs by P and V, respectively. Initially both machine have access to a common input tape. The two machines send messages to one another through two communication tapes. Each machine only sees its own tapes, the common input tape and the communication tapes. In particular, it follows that one machine cannot monitor the internal computation of the other machine nor read the other's coin tosses, current state, program etc. The verifier is bounded to a number of steps which is polynomial in the length of the common input, after which he stops either in an *accept* state or in a *reject* state. At this point we put no restrictions on the local computation conducted by the prover.

We require that, whenever the verifier is following his predetermined program V, the following two conditions hold:

1) *Completeness of the interactive proof system*: If the common input x is in L and the prover runs his predetermined program P, then the verifier accepts x with probability $\geq 1-|x|^{-c}$, for every constant $c>0$. In other words, the prover can convince the verifier that $x \in L$.

2) *Validity of the interactive proof system*: If the common input x is NOT in L, then for every program P' run by the prover the verifier rejects x with probability $\geq 1-|x|^{-c}$, for every constant $c>0$. In other words, the prover cannot fool the verifier.

Remark 1: Note that it does not suffice to require that the verifier cannot be fooled by the predetermined prover P (such a mild condition would have presupposed that the "prover" is trusted by the verifier). We require that no matter how the prover plays, he will fail to "prove" incorrect statements.

Remark 2: As is the case with NP, the conditions imposed on acceptance and rejection are not symmetric. Therefore the existence of an interactive proof for the language L does not imply its existence for the complement of L.

Remark 3: The above "definition" follows the one of Goldwasser, Micali and Rackoff [GMR]. A different definition due to Babai [B], restricts the verifier's actions to generating random strings, sending them to the prover, and evaluating a deterministic polynomial-time predicate at the end of the interaction. In other words, in Babai's framework the coin tosses are public, while in the more general definition of [GMR] the verifier may use a private coin (the output of which may not be revealed to the prover). Designing proof systems seems to be much simpler in the [GMR] model, but making statements about them seems easier if one restricts oneself to Babai's model. Surprisingly, the two models are equivalent as far as language recognition is concerned [GS].

Remark 4: The ability to toss coins is crucial to the non-triviality of the notion of an interactive proof system. Suppose that a language L has an interactive proof system in which the verifier does not toss coins. Then, without loss of generality, this proof system is a trivial one: The prover just guesses the legal conversation, sends it to the verifier which just verifies its validity in deterministic polynomial-time. (It follows that $L \in NP$.)

1.2 What is a zero-knowledge proof

Intuitively, a zero-knowledge proof is a proof which yields nothing but its validity. This means that for all practical purposes, "whatever" can be done after interacting with a zero-knowledge prover, can be done when just believing that the assertion he claims is indeed valid. (In "whatever" we mean not only the computation of functions but also the generation of probabilistic distributions.) Thus, zero-knowledge is a property of the predetermined prover: its robustness against attempts of an arbitrary (polynomial-time) verifier to extract knowledge from him via the interaction. This is captured by the formulation appearing in [GMR] and sketched below.

Denote by $V^*(x)$ the probability distribution generated by a machine V^* which interacts with (the prover) P on the common input $x \in L$. We say that the proof system is *zero-knowledge* if for all probabilistic polynomial-time machines V^*, there exists a probabilistic polynomial-time algorithm M_{V^*} that on input x can produce a probability distribution $M_{V^*}(x)$ that is polynomially-indistinguishable from the distribution $V^*(x)$. (For every algorithm A, let $p_A(x)$ denote the probability that A outputs 1 on input x and an element chosen according to the probability distribution $D(x)$. Similarly, $p_A{}'(x)$ is defined with respect to the probability distribution $D'(x)$. $D(\cdot)$ and $D'(\cdot)$ are *polynomially-indistinguishable* if for every probabilistic polynomial-time algorithm A, $|p_A(x)-p_A{}'(x)| \leq |x|^{-c}$, for every constant $c>0$ and for all sufficiently long x. This notion originates from [GM] and in [Y].)

Remark 5: It is easy to see that if a language L has a zero-knowledge proof system in which only one message is sent, then $L \in BPP$. Thus, the non-triviality of the interaction is a necessary condition for the non-triviality of the notion of zero-knowledge.

1.3 Previous results concerning interactive proofs

In section 1.1, we implicitly discussed the classes of languages having k-move interactive proof systems (i.e. k message exchanges). Let $IP(k)$ denote the class of languages membership in which can be proved through a general interaction consisting of k messages, and let $RIP(k)$ denote languages proven through the restricted type interaction in which the verifier tosses "public coins". Babai [B] showed that for every constant k, $RIP(k)=RIP(2) \subseteq NP^B$ for almost all oracles B. This means that his restricted hierarchy collapses. Goldwasser and Sipser [GS] showed that, surprisingly, for every k, $IP(k) \subseteq RIP(k+3)$. Both the above results say nothing about preservation of zero-knowledge by the transformations.

Several Number Theoretic languages, not known to be in BPP, have been previously shown to have zero-knowledge proof systems. The first language for which such a proof system has been demonstrated is Quadratic Non-Residuosity [GMR]. Other zero-knowledge proof systems were presented in [GMR], [GHY], and [G]. All these languages are known to lie in $NP \cap Co\text{-}NP$.

1.4 Our Results

In this extended abstract, we present only our results which are directly related to cryptography. For these results, we assume the existence of an *arbitrary* secure encryption function. We first show how to prove any NP statement in zero-knowledge. Next we use this ability to develop a methodology of cryptographic protocol design.

We have omitted from this extended abstract our results which are not related to cryptography. These result, which *do not rely on any assumptions,* consists of zero-knowledge interactive proof systems for Graph Isomorphism and Graph Non-Isomorphism. The mere existence of an interactive proof system for Graph Non-Isomorphism is interesting, since Graph Non-Isomorphism is not known to be in NP. For details see our paper [GMW].

1.5 Related Work

Using the intractability assumption of quadratic residuosity, Brassard and Crepeau have discovered independently (but subsequently) zero-knowledge proof systems to all languages in NP [BC1]. These proof systems heavily rely on *particular properties of quadratic residues* and do not seem to extend to arbitrary encryption functions. Recently, Brassard and Crepeau showed that *if factoring is intractable* then every NP language has a "zero-information" interactive proof system [BC2]. It should be stressed that the protocol they proposed constitutes an interactive proof provided that factoring is intractable. In other words, the validity of the interactive proofs depends on an intractability assumption; while in this paper and in [BC1] the proofs do not rely on such an assumption. On the positive side, the protocol presented in [BC2] is "zero-information" in the following strong sense: for every verifier program V^* there is an algorithm M_{V^*}, such that the probability distribution generated by M_{V^*} (on input $x \in L$) is *identical* to the probability distribution generated by V^* (when interacting with the prover on the input x).

Independently, Chaum [Cha] discovered a protocol which is very similar to the one in [BC2]. Chaum also proposed an interesting application of such "zero-information proofs". His application is to a setting in which the verifier may have infinite computing power while the prover is restricted to polynomial-time computations (see also [CEGP]). In such a setting it makes no sense to have the prover demonstrate properties (as membership in a language) to the verifier. However, the prover may wish to demonstrate to the verifier that he "knows" something without revealing what he "knows". More specifically, given a SAT formulae, the prover wishes to convince the verifier that he "knows" a satisfying assignment in a manner that would yield no information which of the satisfying assignments he knows. A definition of the notion of "a program knowing a satisfying assignment" can be derived from [GMR].

1.6 Organization of the Paper

In Section 2 we state our assumptions, and introduce some conventions. In Section 3 we show how to use any one-way permutation in order to construct a zero-knowledge

interactive proof for any language in NP. In Section 4 we example the cryptographic applications of the above result. In Section 5 we outline the fundamental theorems for two-party and multi-party cryptographic protocols.

2. Preliminaries

Throughout this paper we assume the existence of an *arbitrary* secure encryption schemes in the sense of Goldwasser and Micali [GM]. Such schemes exist if unapproximable predicates exist [GM]. The existence of unapproximable predicates has been shown by Yao to be a weaker assumption than the existence of one-way permutations [Y]. In particular, the infeasibility assumption of factoring (equivalently: the assumption that squaring modulo a composite integer is a one-way permutation) implies that the least significant bit of the modular square root is an unapproximable predicate [ACGS]. Note that the existence of one-way permutation is the basis for most of the works and results in "modern cryptography". See for example [DH, RSA, R1, GM, GMRiv, BM].

An encryption scheme secure as in [GM] is a probabilistic polynomial-time algorithm f that on input x and a random string r, outputs an encryption $f(x,r)$. Decryption is unique, that is $f(x,r)=f(y,s)$ implies $x=y$.

Notations: Let A be a set.

1) $Sym(A)$ denote the set of permutations over A.
2) When writing $a \in_R A$, we mean an element chosen at random with uniform probability distribution from the set A.

3. Zero-Knowledge Proofs for All Languages in NP

We begin by presenting a zero-knowledge interactive proof for graph 3-colourability. Using this interactive proof, we present zero-knowledge proofs for every language in NP.

3.1 A Zero-Knowledge Proof for Graph 3-Colourability

The common input to the following protocol is a graph $G(V,E)$. In the following protocol, the prover needs only to be a probabilistic polynomial-time machine which gets a proper 3-colouring of G as an auxiliary input. Let us denote this colouring by ϕ ($\phi: V \rightarrow \{1,2,3\}$). Let $n = |V|$, $m = |E|$. For simplicity, let $V = \{1,2,...,n\}$.

The following four steps are executed m^2 times, each time using independent coin tosses.

1) The prover chooses a random permutation of the 3-colouring, encrypts it, and sends it to the verifier. More specifically, the prover chooses a permutation $\pi \in_R Sym(\{1,2,3\})$, and random r_v's, computes $R_v = f(\pi(\phi(v)), r_v)$ (for every $v \in V$), and sends the sequence $R_1, R_2, ..., R_n$ to the verifier.
2) The verifier chooses at random an edge $e \in_R E$ and sends it to the prover.
3) If $e=(u,v) \in E$ then the prover reveals the colouring of u and v and "proves" that they correspond to their encryptions. More specifically, the prover sends

$(\pi(\phi(u)),r_u)$ and $(\pi(\phi(v)),r_v)$ to the verifier. If $e \notin E$ then the prover stops.

4) (The verifier checks the "proof" provided in step (3).)
The verifier checks whether $R_u = f(\pi(\phi(u)),r_u)$, $R_v = f(\pi(\phi(v)),r_v)$, $\pi(\phi(u)) \neq \pi(\phi(v))$, and $\pi(\phi(u)),\pi(\phi(v)) \in \{1,2,3\}$. If either condition is violated the verifier *rejects* and stops. Otherwise the verifier continues to the next iteration.

If the verifier has completed all m^2 iterations then it *accepts*.

The reader can easily verify the following facts: When the graph is 3-colourable and both prover and verifier follow the protocol then the verifier always accepts. When the graph is not 3-colourable and the verifier follows the protocol then no matter how the prover plays, the verifier will reject with probability at least $(1-m^{-1})^{m^2} = \exp(-m)$. Thus, the above protocol constitutes an interactive proof system for 3-colourability.

Proposition: If $f(\cdot,\cdot)$ is a secure probabilistic encryption, then the above protocol constitutes a zero-knowledge interactive proof system for 3-colourability.

proof's sketch: It is clear that the above prover conveys no knowledge to the SPECIFIED verifier. We need however to show that our prover conveys no knowledge to all possible verifiers, including cheating ones that deviate arbitrarily from the protocol.

Let V^* be an arbitrary fixed program of a probabilistic polynomial-time machine interacting with the prover P, specified by the protocol. We will present a probabilistic polynomial-time machine M_{V^*} that generates a probability distribution which is polynomially indistinguishable from the probability distribution induced on V^*'s tapes during its interaction with the prover P. In fact it suffices to generate the distribution on the random tape and the communication tape of V^*.

Our demonstration of the existence of such M_{V^*} is constructive: given an interactive program V^*, we use it in order to construct the machine M_{V^*}. The way we use V^* in this construction does not correspond to the traditional notion of (a subroutine) reduction [K, C], but rather to a more general notion of reduction suggested in [AHU, pp. 373-374]. The machine M_{V^*} monitors the execution of V^*. In particular, M_{V^*} chooses the random tape of V^*, reads messages from V^*'s communication tape, and writes messages to V^*'s communication tape. Typically, M_{V^*} tries to guess which edge the machine V^* will ask to check. M_{V^*} encrypts an illegal colouring of G such that it can answer V^* in case it (M_{V^*}) is lucky. The cases in which M_{V^*} fails will be ignored: M_{V^*} will just rewind V^* to the last success, and try its luck again. It is crucial that from the point of view of V^* the case which leads to M_{V^*} success and the case which leads to M_{V^*} failure are polynomially indistinguishable.

The machine M_{V^*} monitoring V^*, starts by choosing a random tape r for V^*. M_{V^*} places r on its record tape and proceeds in m^2 rounds as follows.

1) M_{V^*} picks an edge $(u,v) \in_R E$ and a pair of integers $(a,b) \in_R \{(i,j): 1 \leq i \neq j \leq 3\}$ at random. M_{V^*} chooses random r_i's and

computes $R_i = f(c_i, r_i)$, where $c_i = 0$ for $i \in V - \{u, v\}$, $c_u = a$ and $c_v = b$. M_{V^*} places the sequence of R_i's on the communication tape of V^*.

2) M_{V^*} reads e from the communication tape of V^*. If $e \notin E$ (V^* cheats) then M_{V^*} appends the R_i's and e to its record tape, outputs the record tape, and stops. If $e \neq (u, v)$ (unlucky for M_{V^*}) then M_{V^*} rewinds V^* to the configuration at the beginning of the current round, and repeats the current round with new random choices. If $e = (u, v)$ (lucky for M_{V^*}) then M_{V^*} proceeds as follows: First, it places (a, r_u) and (b, r_v) on the communication tape of V^*. Second, it appends the R_i's, e, (a, r_u) and (b, r_v) to its record tape; and finally, it proceeds to the next round.

If all rounds are completed then M_{V^*} outputs its record and halts. A technical lemma (to be stated and proved in the final paper) guarantees that the three possible "answers" of the verifier (i.e. $e \notin E$, $e \in E - \{(u, v)\}$ and $e = (u, v)$) occur with essentially the same probability as in the interaction of V^* and the real prover. Thus, the probability that the simulation of a particular round requires more than $k \cdot m$ rewinds is smaller than 2^{-k}, and M_{V^*} terminates in polynomial time. The only difference between the probability distribution of the true interactions and the distribution generated by M_{V^*} is that the first contain probabilistic encryptions of colourings while the second contains probabilistic encryptions of mostly 0's. However, a second technical lemma (postponed to the final paper) asserts that this difference is indistinguishable in probabilistic polynomial-time.

Remark 6: The above protocol needs m^2 rounds. In the final version of our paper we will present two alternative ways of modifying the above protocol so to get a four-round zero-knowledge protocol for graph 3-colorability. In both modifications the idea is to have the verifier send the prover "encryptions" of all his questions (i.e. which edge he wants to check for each copy of the coloured graph) before the prover sends to the verifier the corresponding coloured graphs. By this, the verifier commits himself to a test before seeing the encrypted colourings (equivalently, the tests are only a function of the common input and the random coin tosses of the verifier). *How can the verifier encrypt his questions?* This is the point in which the two modifications differ.

1) Assuming the intractability of integer factorization, the verifier encrypts as follows. The prover first randomly chooses a Blum integer N (i.e. a composite integer which is the product of two large primes each congruent to 3 modulo 4). To encrypt the bit $\sigma \in \{0,1\}$, the verifier randomly chooses a residue $r \in Z_N^*$ with Jacobi Symbol $(-1)^\sigma$, computes $s = r^2 \bmod N$, sends s to the prover and proves (in zero-knowledge) that he "knows" a square root of s (consult [FMRW, GMR]). Note that even with infinite computing power, the prover can not know σ. To reveal σ, the verifier presents r. Assuming the intractability of factoring, the verifier can not "change his mind" about σ.

2) A trivial solution follows by modifying the definition of an interactive proof such that the prover is also restricted to polynomial-time computation, and his

"computational advantage" over the verifier is merely in having an auxiliary input. Note that this is the natural cryptographic scenario.

Remark 7: The above protocol can be easily modified to yield a zero-information protocol that constitutes a proof system if factoring is intractable. The modification consists of having the prover encrypt the colouring by using a Blum integers selected by the verifier (analogue to (1) in Remark 6). More details will be given the final version of our paper.

3.2 Zero-Knowledge proofs for all Languages in NP

Incorporating the standard reductions into the protocol of Section 3.1, we get

Theorem 1: If $f(\cdot,\cdot)$ is a secure probabilistic encryption, then every NP language has a zero-knowledge interactive proof system.

Proof: For every language $L \in NP$ the protocol incorporates a fixed reduction of L to 3-colourability. Each party computes the 3-colourability instance from the common input, and then the prover proves to the verifier that this instance is 3-colourable (using the protocol of section 3.1). **QED**

Slightly less obvious is the proof of the following Theorem 2 that adapts Theorem 1 to the cryptographic scenario, in which all players are bounded to efficient computation.

Theorem 2: If $f(\cdot,\cdot)$ is a secure probabilistic encryption, every language in NP has a zero-knowledge interactive proof system whose prover is a probabilistic polynomial-time machine which gets a NP proof as an auxiliary input.

Proof: We would like the parties to proceed as in the proof of Theorem 1. The problem is whether the prover is powerful enough to execute his role in that protocol. Note that if the prover is given a colouring of the reduced 3-colourability instance then he can follow the instructions of the protocol in Section 3.1. However, the prover is only given a NP proof for the membership in an arbitrary language in NP. The difficulty is resolved by noticing that the standard reductions used in the protocol *efficiently* transform also the witnesses to the corresponding instances (see details below).

Most known Karp-reductions have the property that, given a NP-proof to the original instance, one can easily obtain a NP-proof for the reduced instance. Let L_1 be a language which is Karp-reducible to the language L_2 by the polynomial-time function t. Let $L \in NP$ and $x \in L$, then we denote by $w(x)$ a witness for x (i.e. $P_L(x, w(x))=1$, where P_L is the polynomial-time predicate associated to L). If there exist a polynomial-time computable function g such that for every instance $x_1 \in L_1$ we have $w(t(x_1)) = g(w(x_1))$ then we say that L_1 is *Levin-reducible* to L_2. (This is "half the condition" in the definition of polynomial reducibility as appeared in Levin's paper

"Universal Search Problems" [L].) Thus, it suffices to verify that the generic reduction to SAT, and the "popular" reductions of SAT to 3SAT and of 3SAT to 3C, are all in fact Levin-reductions. **QED**

Remark 8: Theorem 1 can be generalized to show that not only NP is in zero-knowledge, but also "probabilistic NP" is. In other words, if $f(\cdot,\cdot)$ is a secure probabilistic encryption, then for every fixed k every language in $IP(k)$ has zero-knowledge proof systems. The same holds for Theorem 2 and all languages in $RIP(k)$ (note that Goldwasser and Sipser's transformation of $IP(k)$-protocols to $RIP(k)$-protocols requires the prover to conduct approximate counting). For more details see [GMW].

4. Examples of Particular Applications of Theorem 2

Theorem 2 has a dramatic effect on the design of cryptographic protocols. Typically these protocols must cope with the problem of the parties convincing each other that they are sending messages which are computed according to the protocol. Such proofs should be carried out without yielding any secret knowledge. Since it is alway possible to give NP proofs that the messages are computed properly, we can now give zero-knowledge proofs of this fact. Let us demonstrate this point, by using Theorem 2 to present simple solutions to three problems, which until recently were considered extremely difficult or even impossible. The more general implications of Theorem 2, are outlined in the preceding chapter.

A central notion in the field of cryptography is that of a secret. By a secret we mean a piece of data that once given can be recognized as the desired one. More formally, a *secret* s is *recognizable through* $g(s)$ if g is a one-way function. For example, the factorization (p,q) of a composite integer $N=p\cdot q$ is a secret recognizable through N. The digital signature $S_U(m)$ of user U to the message m is a secret recognizable through the message m and the public-key of U.

4.1 Oblivious Transfer of Arbitrary Secrets

The notion of Oblivious Transfer, suggested by Rabin [R2], has attracted alot of attention within the study of cryptographic protocols. An *Oblivious Transfer* is a two-party protocol through which one party (unknowingly) transfers with probability 1/2 a large amount of knowledge to his counterpart, and yields no knowledge otherwise [R2, EGL]. Initially, the *sender* (S) knows a secret s recognizable through $g(s)$, and the *receiver* (R) knows $g(s)$. If both parties follow the protocol then R gets s with probability 1/2. If R follows the protocol then *for* S, the a-posteriori probability that R got s equals the a-priori probability. Rabin required that an attempt by S to reduce the probability that R receives s is detected [R2] with very high probability; while Even, Goldreich and Lempel only required that such an attempt is detected with probability 1/2 [EGL].

In the following we will assume that factoring is hard. For the case that the secret is the factorization of a given integer, a protocol satisfying Rabin's conditions was

presented by Fischer, Micali, Rackoff and Wittenberg [FMRW] (modifying [R2]). This protocol easily extends to arbitrary secrets, but in this case detection of "cheating" is only guaranteed with probability 1/2. It has been conjectured that no protocol can meet Rabin's condition (i.e. allow to always detect attempts to reduce the probability of a transfer) for arbitrary secrets [EGL].

Using Theorem 2, we show that the conjecture in [EGL] is false. The proposed protocol proceeds as follows: First, the sender encrypt the secret s using a randomly chosen composite N. Next, the sender provides the receiver with a zero-knowledge proof that the encrypted message is indeed the desired secret (note that this is a NP statement). Finally, the sender uses the [FMRW] Oblivious Transfer to send the factorization of N such that it is received with probability 1/2.

Remark 9: Recently, we presented an Oblivious Transfer protocol based on the security of an arbitrary public-key encryption function.

4.2 Verifiable Secret Sharing

The notion of a verifiable secret sharing was presented by Chor, Goldwasser, Micali, and Awerbuch [CGMA], and constitutes a powerful tool for multi-party protocol design. A *verifiable secret sharing* is a $n+1$-party protocol through which a *sender* (S) can distribute to the *receivers* (R_i's) *pieces* of a secret s recognizable through $g(s)$. The n pieces satisfying the following three conditions (with respect to $1 \leq l < u \leq n$):

1) It is infeasible to obtain any partial information about the secret from any l pieces;

2) Given any u messages the entire secret can be easily computed;

3) Given a piece it is easy to verify that it belongs to a set satisfying condition (2).

The notion of a verifiable secret sharing differs from Shamir's secret sharing [Sha], in that the secret is recognizable and that the pieces should be verifiable as authentic (i.e. condition (3)).

We will consider solutions which are polynomial in n and in the security parameter. The first solution, presented in [CGMA], relies on RSA (resp. factoring) and works for $l = O(log\ n)$ (resp. $l = O(log\ \log n)$, see also [CGG]). Relying on the difficulty of testing quadratic residuosity this solution was improved, independently by [FM] and [AGY], to allow $l = \alpha n$ and $u = (1-\alpha)n$ for every fixed $\alpha < 1/2$. Recently, Feldman [F] presented a solution allowing $u = l+1 \leq n$, assuming the intractability of the discrete logarithm function. Most of the above solutions are conceptually very complicated.

Combining Theorem 2 with Shamir's scheme [Sha], we present a conceptually simple solution allowing $u = l+1 \leq n$, assuming the existence of *arbitrary* one-way permutations. To share a secret $s \in Z_p$ recognizable through $g(s)$, the sender proceeds as follows: First, the sender chooses at random a l-degree polynomial over Z_p^* and evaluates it in n fixed points (these are the pieces in Shamir's scheme). Next, the sender encrypts the ith piece using the Public-Key of the ith receiver, and sends all encrypted secrets to all receivers. Finally, the sender provides each receiver with a zero-knowledge proof that the encrypted messages correspond to the evaluation of a single polynomial over Z_p^*

(note that this is a NP statement).

Recently, Benaloh has presented a much more efficient solution based on the intractability of quadratic residuosity [Bena].

4.3 Proving that a String is Pseudorandom

The notion of a pseudorandom bit generator, suggested by Blum and Micali [BM] and Yao [Y], is central to cryptography. A *pseudorandom bit generator* is an efficient deterministic program which stretches a randomly selected n-bit long seed into a longer bit sequence which is polynomially-indistinguishable from a random string [BM, Y]. A *pseudorandom function generator* is an efficient deterministic program that uses a random n-bit seed to construct an oracle which is polynomially-indistinguishable from a random oracle [GGM].

Using Theorem 2, a party which has selected the seed can present zero-knowledge proofs that the sequence/function he is producing/implementing is indeed pseudorandom.

5. Two Theorems for Cryptographic Protocols

In this section, we present an extremely powerful methodology for designing correct cryptographic protocols. The methodology consists of efficient "correctness and privacy preserving" transformations of protocols from a weak adversary model to the most adversarial model. These transformations are informally summarized as follows

> **Informal Theorem A**: There exist an efficient compiler transforming a protocol P designed for $n=2t+1$ honest players, to a cryptographic protocol P' that achieves the same goals even if t of its n players are faulty. Faulty players are allowed to deviate from P' in an arbitrary but polynomial-time way.

In the formal statement of the corresponding Theorem, we avoid talking about "achieving goals". The "goal of a protocol" is a semantic object that is not well understood. Instead, we make statements about well understood syntactic objects: the probability distribution on the tapes of interactive machines. In the final version of this paper we will define the notions of a "correctness preserving compiler" and a "privacy preserving compiler". Both notions will be defined as relations between the probability distribution on the tapes of interactive machines during the execution of protocol P (in a weak adversarial environment) and the distribution on these tapes during the execution of P' (in a strong adversarial environment). Loosely speaking, "preserving correctness" means that whatever a party could compute after participating in the original protocol P, he could also compute when following the transformed protocol P', properly. "Preserving privacy" means that whatever a set of dishonest players can compute after participating in P', the corresponding players in P can compute when sharing their "knowledge" after participating in P. Similarly we formalize the following

> **Informal Theorem B**: There exist an efficient compiler transforming a two-party protocol P that is correct in a fail-stop model, to a cryptographic two-party-protocol P' that achieves the same goals even if one of the players deviates from

P' in an arbitrary but polynomial-time way.

The proofs of the above Theorems make primary use of Theorem 2 to allow a machine to "prove" to other machines that a message it sent is computed according to the protocol. In addition, these proofs make innovative use of most of the cryptographic techniques developed in recent years. Essential ingredients in the proof of Theorem A are the notions of verifiable secret sharing and simultaneous broadcast proposed and first implemented by Chor, Goldwasser, Micali, and Awerbuch [CGMA]. An essential ingredient in the proof of Theorem B is Blum's "coin flipping into the well" [Blu].

Further Improvements

Theorem A constitutes a procedure for automatically constructing fault-tolerant protocols, the goal of which is to compute a predetermine function of the private inputs scattered among the players. This procedure takes as input a distributed specification of the function (i.e. a protocol for honest players), not the function itself. It is guaranteed that this procedure will output a fault-tolerant protocol for computing this very function (i.e. the "correctness" condition) and that the "privacy" present in the specification will be preserved. Thus, the degree of privacy offered by the output fault-tolerance protocol depends on the specification, and not on the function to be computed. Furthermore, for some functions f it seems to be difficult to write a distributed specification (protocol for honest players) which offers the maximum degree of privacy.

Recently (see forthcoming paper [GMW2]), we found a polynomial-time algorithm which on input a *Turing machine* specification of a n-ary function f, outputs a protocol for n honest players which offers maximum privacy. Namely, at the termination of the protocol, each subset of players can compute from their joint local history only whatever they could have computed from their corresponding local inputs and the value of the function. Thus, we achieve for any n-ary function what Benaloh [Bena] has achieved for the addition and multiplication functions.

Combined with the compiler of Theorem A, our algorithm constitutes an automatic generator of fault-tolerant protocols. This may be viewed as a completeness theorem for fault tolerant distributed computation.

ACKNOWLEDGEMENTS

It is our pleasure to thank Benny Chor and Shafi Goldwasser for many helpful discussions concerning this work.

REFERENCES

[AHU] Aho, A.V., J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publ. Co., 1974.

[ACGS] Alexi, W., B. Chor, O. Goldreich, and C.P. Schnorr, "RSA and Rabin Functions: Certain Parts Are As Hard As The Whole", to appear in *SIAM Jour. on Computing.* Extended Abstract in *Proc. 25th FOCS*, 1984.

[AGY] Alon, N., Z. Galil, and M. Yung, "A Fully Polynomial Simultaneous Broadcast in the Presence of Faults", preprint, 1985.

[B] Babai, L., "Trading Group Theory for Randomness", *Proc. 17th STOC*, 1985, pp. 421-429.

[Bena] Benaloh, (Cohen) J.D., "Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret", these proceedings.

[Blu] Blum, M., "Coin Flipping by Phone", *IEEE Spring COMPCOM*, pp. 133-137, February 1982.

[BM] Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM Jour. on Computing*, Vol. 13, 1984, pp. 850-864.

[BC1] Brassard, G., and C. Crepeau, "Zero-Knowledge Simulation of Boolean Circuits", manuscript 1986.

[BC2] Brassard, G., and C. Crepeau, "Non-Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond", manuscript, 1986.

[BD] Broder, A.Z., and D. Dolev, "Flipping Coins in Many Pockets (Byzantine Agreement on Uniformly Random Values", *Proc. 25th FOCS*, 1984, pp. 157-170.

[Cha] Chaum, D., "Demonstrating that a Public Predicate can be Satisfied Without Revealing Any Information About How", these proceedings.

[CEGP] Chaum, D., J.H. Evertse, J. van de Graaf, and R. Peralta, "Demonstrating Possession of a Discrete Logarithm without Revealing It", these proceedings.

[CGG] Chor, B., O. Goldreich, and S. Goldwasser, "The Bit Security of Modular Squaring given Partial Factorization of the Modulos", *Proc. of Crypto85*, to appear (1986).

[CGMA] Chor, B., S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults", *Proc. 26th FOCS*, 1985, pp. 383-395.

[C] Cook, S.A., "The Complexity of Theorem Proving Procedures", *Proc. 3rd STOC*, pp. 151-158, 1971.

[DH] Diffie, W., and M.E. Hellman, "New Directions in Cryptography", *IEEE Trans. on Inform. Theory*, Vol. IT-22, No. 6, November 1976, pp. 644-654.

[EGL] Even, S., O. Goldreich, and A. Lempel, "A Randomized Protocol for Signing Contracts", *CACM*, Vol. 28, No. 6, 1985, pp. 637-647.

[F] Feldman, P., "A Practical Scheme for Verifiable Secret Sharing", manuscript, 1986.

[FM] Feldman, P., and S., Micali, in preparation, 1985.

[FMRW] Fischer, M., S. Micali, C. Rackoff, and D.K. Wittenberg, "An Oblivious Transfer Protocol Equivalent to Factoring", in preparation. Preliminary versions were presented in *EuroCrypt84* (1984), and in the *NSF Workshop on Mathematical Theory of Security*, Endicott House (1985).

[GHY] Galil, Z., S. Haber, and M. Yung, "A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems", *Proc. 26th*

FOCS, 1985, pp. 360-371.

[GJ] Garey, M.R., and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.

[G] Goldreich, O., "A Zero-Knowledge Proof that a Two-Prime Moduli Is Not a Blum Integer", unpublished manuscript, 1985.

[GGM] Goldreich, O., S. Goldwasser, and S. Micali, "How to Construct Random Functions", *Proc. of 25th Symp. on Foundation of Computer Science*, 1984, pp. 464-479. To appear in *Jour. of ACM*.

[GMW] Goldreich, O., S. Micali, and A. Wigderson, "Proofs that Yield Nothing But their Validity", in preparation. An extended abstract will appear in the proceedings of *27th FOCS*, 1986.

[GMW2] Goldreich, O., S. Micali, and A. Wigderson, "How to Automatically Generate Correct and Private Fault-Tolerant Protocols", in preparations.

[GM] Goldwasser, S., and S. Micali, "Probabilistic Encryption", *JCSS*, Vol. 28, No. 2, 1984, pp. 270-299.

[GMR] Goldwasser, S., S. Micali, and C. Rackoff, "Knowledge Complexity of Interactive Proofs", *Proc. 17th STOC*, 1985, pp. 291-304.

[GMRiv] Goldwasser, S., S. Micali, and R.L. Rivest, "A *Paradoxical* Signature Scheme", *Proc. 25th FOCS*, 1984.

[GS] Goldwasser, S., and M. Sipser, "Arthur Merlin Games versus Interactive Proof Systems", *Proc. 18th STOC*, pp. 59-68, 1986.

[K] Karp, R.M., "Reducibility among Combinatorial Problems", *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher (eds.), Plenum Press, pp. 85-103, 1972.

[L] Levin, L.A., "Universal Search Problems", *Problemy Peredaci Informacii 9*, pp. 115-116, 1973. Translated in *problems of Information Transmission 9*, pp. 265-266.

[RSA] Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Comm. of the ACM*, Vol. 21, February 1978, pp. 120-126.

[R1] Rabin, M.O., "Digitalized Signatures as Intractable as Factorization", MIT/LCS/TR-212, 1979.

[R2] Rabin, M.O., "How to Exchange Secrets by Oblivious Transfer", unpublished manuscript, 1981.

[Sha] Shamir, A., "How to Share a Secret", *CACM*, Vol. 22, 1979, pp. 612-613.

[Y] Yao, A.C., "Theory and Applications of Trapdoor Functions", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, 1982, pp. 80-91.