Perfect Zero-Knowledge Arguments for NP Using any One-Way Permutation[*]

Moni Naor[†] Rafail Ostrovsky[‡] Ramarathnam Venkatesan[§] Moti Yung[¶]

Abstract

"Perfect zero-knowledge arguments" is a cryptographic primitive which allows one polynomial-time player to convince another polynomial-time player of the validity of an NP statement, without revealing any additional information (in the information-theoretic sense). Here the security achieved is *on-line*: in order to cheat and validate a false theorem, the prover must break a cryptographic assumption on-line *during the conversation*, while the verifier cannot find (ever) any information unconditionally. Despite their practical and theoretical importance, it was only known how to implement zero-knowledge arguments based on specific algebraic assumptions.

In this paper, we show a general construction, which can be based on *any* one-way permutation. The result is obtained by a construction of an information-theoretic secure bit-commitment protocol. The protocol is efficient (both parties are polynomial time) and can be based on any one-way permutation.

[*]A preliminary version of this paper appeared in *Advances in cryptology - Crypto 92 Proceedings*, Lecture Notes in Computer Science 740, Springer-Verlag, 1993.

[†]Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: naor@wisdom.weizmann.ac.il. Research supported by grants from the Israel Science Foundation administered by the Israeli Academy of Sciences and by a US-Israel Binational Science Foundation. Part of this work was done while the author was with the IBM Almaden Research Center.

[‡]Math and Cryptography Research Group, Bellcore, 445 South Street, Morristown, N.J. 07960, U.S.A. E-mail: rafail@bellcore.com. Part of this work was done while the author was at the International Computer Science Institute at Berkeley and the University of California at Berkeley.

[§]Microsoft Research, One Microsoft Way, Redmond, WA 98052, U.S.A. E-mail: venkie@microsoft.com

[¶]IBM Research, T.J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A. E-mail: moti@cs.columbia.edu.

1 Introduction

Reducing complexity assumptions for basic cryptographic primitives is a major current research program in cryptography. Characterizing the necessary and sufficient complexity conditions needed for primitives helps us develop the theoretical foundations of cryptography. Further, reducing requirements for a primitive may imply more concrete underlying functions for its practical implementations.

From this perspective, we study here the requirements for the existence of zero-knowledge (ZK) arguments for proving the "validity of an NP assertion". Informally, proving some fact in zero-knowledge (a notion introduced in [19]), is a way for one player (called "prover") to convince another player (called "verifier") that certain fact is true, while not revealing any additional information. In our setting, both players are polynomially bounded and the prover is presumed to have the witness for the proof of the NP statement. This has a large variety of applications in cryptography and distributed computing (see [19, 18]). In such applications, the prover may choose the NP-instance in such a way so that the witness is known (e.g. by evaluating a one-way function on some input) or possess some secret information that constitutes the witness. We must rely on complexity assumptions, since protocols for implementing the above task with polynomial time players imply the existence of one-way functions (see [22] and [35]). The assumptions could be used in two different ways:

1. Zero-knowledge *proofs* [19, 18]: The prover, even with infinite computational power, cannot convince the verifier to accept a false theorem. But, the proof itself is only *computationally secure*: i.e., if the verifier (or anyone overhearing the execution of the protocol) ever breaks the cryptographic assumption, say after 100 years, it *can* extract additional knowledge about the proof.

2. Perfect zero-knowledge *arguments* [6]: The verifier cannot extract additional information even if it is given infinite time (i.e., security is *perfect* or information theoretic); however, the prover (assumed to be polynomial-time) can cheat in its proof only if it manages to break the assumption *on-line during the execution of the protocol*. This is the reason to call it an "argument" rather than a "proof".

In many settings, ZK-arguments, which were introduced by Brassard, Chaum and Crépeau [6], may be preferable to ZK- proofs: the verifier must only be sure that the prover did not break the assumption *during their interaction* (which lasted, say, ten seconds or ten minutes). Notice that while assuring that the assumption can *never* be broken may be unreasonable, the assumption that something cannot be broken *during the next ten minutes* can be based on the current state of the art. On the other hand, the prover has absolute (i.e. information-theoretic) guarantee that no additional information is released, even if the verifier spends as much time as it desires trying (off-line) to extract it. Thus, the notion of zero-knowledge arguments is useful if there is a need to maintain the secrecy for very long time independent of the possible future advance of cryptanalysis.

So far the complexity assumptions needed for constructing perfect-zero-knowledge arguments were not general – they required specific algebraic assumptions. This is in contrast with zero-knowledge interactive proofs, which can be based on any one-way function. In this work we dispose of specific algebraic assumptions for zero-knowledge arguments:

Main result: If one-way permutations exist, then there exist perfect zero-knowledge arguments for any in $\mathcal{NP}$

We obtain this result by constructing an information-theoretically secure bit-commitment scheme which can be based on any one-way permutation. The scheme is efficient and thus *simulatable* by an expected polynomial time algorithm. We can then apply the known reduction of "perfectly-secure computationally-binding bit commitment" to "perfect ZK-argument". Most of the paper is devoted to the description of the bit-commitment scheme and its correctness and security proof.

1.1 Background

Past successes in establishing basic cryptographic primitives on general assumptions (initiated in [37]) have shown that various primitives, which were originally based on specific algebraic functions, can be based on the existence of general one-way functions or permutations. For example, Naor [30] showed that computationally secure bit commitments (i.e., bit commitments which *can be* broken off-line given sufficient resources) can be constructed from a pseudo-random generator. Such generators [1, 37] were first implemented based on a discrete logarithm assumption in [1] and following a sequence of papers [37, 26, 16, 17] it was shown that *any* one-way function suffices [20, 21]. Similarly, digital-signatures can now be based on any one-way function [31, 36]. Furthermore, these primitives (and primitives derived from them, e.g. identification) were shown to imply a one-way function (thus they are equivalent) [22].

Concerning secure proofs, Goldreich, Micali and Wigderson [18] showed that zero-knowledge *proofs* for $\mathcal{NP}$ can be done using computationally secure bit commitment protocols which, as indicated above, can be obtained from any one-way function. This applies to general $\mathcal{IP}$ proofs as well [24]. On the other hand, zero-knowledge proofs for non-trivial languages imply the existence of one-way functions [35].

In contrast to computational zero-knowledge *proofs*, the only known constructions for perfect zero-knowledge *arguments* for NP was under specific algebraic assumptions [6, 4, 24, 7, 3, 23] or under the assumption that *collision intractable hash functions* exists (first shown in [31]; see [11] for more information), which in turn is only known to be constructed under specific algebraic assumptions [5, 8, 9]. Our result gives the first general reduction: zero-knowledge NP-arguments can be constructed given *any* one-way permutation, by first constructing an information theoretically secure bit commitment.

1.2 Organization of the paper

In section 2, we give the model, the formal definitions of the problem, and the assumptions. Specifically, we present the model of interactive machines, the notion of commitment and of one-way functions and permutations and the definition of perfect zero-knowledge arguments. In Section 3 we present the new method for basing a perfectly-secure bit commitment on a one-way permutation and prove its security. In Section 4 we discuss possible extensions of our techniques. For completeness, we provide in Appendix A a comparison between this work and other recent work on commitments.

2 Model and Definitions

We now review the model and definitions of bit commitment, one-way permutations and perfect zero-knowledge arguments (a.k.a. computationally-sound proofs). In general we follow Goldreich [13]. The parties in the protocols are modeled as *Interacting Turing machines* as defined by [19] which share an access to a security parameter n, a common input and communication tapes. In addition each has an output tape, a private random tape (or string, a.k.a. as its coin-flips) and an auxiliary private input tape. When we say that a machine is polynomial-time it is polynomial in the security parameter (given in unary) and in general all other inputs (including the auxiliary) should be polynomial in the security parameter. We call a function $\rho(n)$ negligible if for all polynomials $p(n)$, $\rho(n) = o(1/p(n))$. That is, it is asymptotically smaller than all inverse polynomials.

Before we continue we should clarify a few issues regarding uniformity. Most cryptographic primitives come in two flavors: (i) uniform, where the adversary is assumed to be a probabilistic polynomial-time machine and (ii) non-uniform, where the adversary's computational power is modeled by polynomial sized circuit. (See [12, 13] for an extensive treatment of the subject.) A construction of one cryptographic primitive from another may be *uniformity preserving* meaning that the new primitive is secure against probabilistic polynomial-time adversaries if the original primitive is secure against such adversaries. Alternatively, it may be only *non-uniform*, meaning that the new primitive is secure only if the original primitive is secure against polynomial-sized circuits. (In all cases we are aware of, if the construction is uniformity preserving then it is also non-uniformity preserving, hence the usage of "only"; furthermore, this can be formalized to cover most cases.) Our construction of perfectly secure computationally binding bit-commitments from one-way permutation is uniformity preserving. However, when using such bit-commitments to construct zero-knowledge arguments for languages in NP some delicate issues that are beyond the scope of this paper arise. Therefore we provide only the non-uniform version of the zero-knowledge arguments and refer to [12] as the source for making the uniform case.

2.1 Commitment

A bit commitment protocol involves two interacting parties, the Sender and the Receiver. It can be thought of the Sender giving the Receiver a locked box with a secret bit inside. The receiver doesn't learn anything about the bit, but at a later stage, when the box is opened, it is sure that the contents of the box were not altered. More formally, a **bit commitment** protocol consists of two stages:

- The *commit* stage: the Sender $\mathcal{S}$ has a bit b on its input tape, to which she wishes to commit to the Receiver $\mathcal{R}$. The sender and the receiver exchange messages. At the end of this stage $\mathcal{R}$ has some information that "represents" b written on its output tape.

- The *reveal (opening) stage*: $\mathcal{S}$ and $\mathcal{R}$ exchange messages (where their output tapes from the commit stage are serving as input tapes for this stage). At the end of the exchange $\mathcal{R}$ writes on its output tape either "OK for bit b" or "NOT OK".

We should take care in defining what we mean by cheating in the context of information-theoretic commitment. Consider the following experiment: after the commit stage $\mathcal{S}$ is "split" into $\mathcal{S}_0$ and $\mathcal{S}_1$ and participates in two executions of the reveal protocol with two identical copies of $\mathcal{R}$ whose state is initialed to be that of $\mathcal{R}$ after the commit stage. If both executions end up with $\mathcal{R}$ writing "OK" on the tape, but the two bits written are not the same, then $\mathcal{S}$ is considered to have successfully cheated. More precisely, at any point in time the state of an interactive machine is determined by its random string r and the messages it received $\vec{m}$. The sender is specified by two machines $\{\mathcal{S}_0, \mathcal{S}_1\}$ so that when given the same random string $\mathcal{S}_0$ and $\mathcal{S}_1$ have *identical* behavior during the commit. I,e when sent the same input messages, they respond back with the same message (this is what it means that the $\mathcal{S}$ is split after the commit phase). If we have two interacting machines and we fix their random strings, then the outcome of their interaction is deterministic. We denote it by $\langle \mathcal{R}(r), \mathcal{S}(s)\rangle$ where r is the random string of $\mathcal{R}$ and s the random string of $\mathcal{S}$.

Definition 1 *We say that a sender $\mathcal{S} = \{\mathcal{S}_0, \mathcal{S}_1\}$* cheats *a receiver $\mathcal{R}$ with probability at most ρ if the following holds: the probability that the executions $\langle \mathcal{R}(r), \mathcal{S}_0(s)\rangle$ and $\langle \mathcal{R}(r), \mathcal{S}_1(s)\rangle$ end up following the reveal stage with "OK" but with two different bits, is at most ρ where the probability is over the choice of r and s.*

By a protocol we actually mean a family of protocols, indexed by the security parameter n. As is usual in computational based Cryptography, the security is a function of n. Note that in the definition below only the probability of cheating depends on n (but the security is independent of n).

Definition 2 *To be* **perfectly-secure computationally binding commitment**, *the protocol must obey the following for some negligible $\rho(n)$:*

1. *(*Viability*) If both players are honest (i.e. follow the protocol as specified), then for any input bit $b \in \{0,1\}$ the sender $\mathcal{S}$ gets, the receiver outputs at the end of the reveal stage the "OK for bit b" with probability one.*

2. *(*Security property:*) For* any *$\mathcal{R}'$ the distributions of the conversation between an (honest) $\mathcal{S}$ and $\mathcal{R}'$ in case $b = 0$ and $b = 1$ are identical. Note that the computational resources of $\mathcal{R}'$ are not bounded.*

3. *(*Binding property:*) The probability that any probabilistic polynomial time $\mathcal{S}' = \{\mathcal{S}'_0, \mathcal{S}'_1\}$ can successfully cheat is at most $\rho(n)$ where the probability is over the random tapes of $\mathcal{S}'$ and $\mathcal{R}$.*

4. *(*Efficiency*) $\mathcal{S}$ and $\mathcal{R}$'s algorithms can be executed in polynomial in the security parameter n time by a probabilistic Turing Machine.*

Remark 1 *Suppose that in property 1 above instead of requiring that the distributions in case $b = 0$ and in case $b = 1$ be identical we require that they will be close to each other to within $\rho(n)$ under, say, the L_1 norm. Then we get a* **statistically secure computationally binding commitment**. *This is good enough for many applications.*

2.2 One-way functions and permutations

We now define the underlying cryptographic primitive we assume. Let f be a length preserving function $f: \{0,1\}^* \to \{0,1\}^*$ computable in polynomial time. By $a \in_R A$ we mean that the element a is randomly chosen from the set A.

Definition 3 *f is* **one-way** *if for every probabilistic polynomial time algorithm $\mathcal{A}$, for all polynomials p and all sufficiently large n,*

$$Pr[f(x) = f(\mathcal{A}(f(x))) \mid x \in_R \{0,1\}^n] < \frac{1}{p(n)}$$

where the probability is over the random choices of x and the random tape of $\mathcal{A}$.

The above definition is of a *strong one-way function*. Its existence is equivalent to the existence of *weak one-way function* using Yao's amplification technique [37] or the more security preserving method of [15] which is applicable only to permutations or regular functions. (A weak one-way function has the same definition as above, but the hardness of inversion is smaller, i.e. its probability is inverse polynomially away from 1.)

If in addition f is 1-1 and length preserving then we say the f is a one-way permutation. For the construction of Section 3 we require a one-way permutation f. Note that the construction there assumes a one-way permutation f on $\{0,1\}^n$. Suppose that instead we have a one-way permutation $f' : S \mapsto S$ where $S \subset \{0,1\}^n$ is an easily recognizable and large set (non-negligible fraction of $\{0,1\}^n$), e.g. all numbers smaller than P where $2^{n-1} \leq P < 2^n$, as is the case in the number theoretic constructions. Then we can construct from it a weak one-way permutation $f : \{0,1\}^n \mapsto \{0,1\}^n$ by taking $f(x) = f'(x)$ if $x \in S$ and $f(x) = x$ otherwise. Using the amplification techniques of [37, 15] we can then obtain a *strong* one-way permutation on a domain $\{0,1\}^{n'}$ for n' linear in n.

The goal of this paper is to present a construction of perfectly-secure computationally binding commitment from any one-way permutation.

2.3 Perfect Zero-Knowledge Arguments

We now briefly discuss perfect zero-knowledge arguments (a.k.a computationally sound proof systems). The reason we are brief is that the paper does not deal with them directly, but their existence is a known consequence of the construction of the perfectly-secure computationally binding commitment protocol. For a more thorough discussion see [13].

In a proof system there are two interacting machines commonly called the prover P and the verifier V. The two parties share access to a security parameter n and a common input x which the prover "claims" is in a language $\mathcal{L}$. The prover should have in its auxiliary input tape a witness for this fact. In addition each party has an output tape, a private random tape and perhaps more information on their auxiliary private input tape. The three properties the proof system should have are (i) Completeness, meaning that if $x \in \mathcal{L}$ then the interaction should cause the verifier to write "ACCEPT" on its output tape (which we denote "$ACCEPT \in \langle P, V\rangle(x)$). (ii) Soundness which in this case is only computational, i.e. for any "bad" prover who is polynomially bounded, the probability that it makes the verifier write "ACCEPT" when $x \notin \mathcal{L}$ is small and (iii) Zero-knowledge, which here we require it to be perfect, i.e. for every "bad" verifier it is possible to simulate precisely its output and message distribution.

Definition 4 *(*perfect zero-knowledge arguments*) A pair of interactive machines (P, V) is a* perfect zero-knowledge arguments system for a language $\mathcal{L}$ *if both machines are polynomial-time and*

1. *(*Completeness*) For every $x \in \mathcal{L}$ there is a witness y such that*

$$\Pr[\text{“}ACCEPT\text{”} \in \langle P(y), V\rangle(x)] \geq \frac{2}{3}.$$

We say that the completeness is perfect *if for every $x \in \mathcal{L}$ there is a witness y such that*

$$\Pr[\text{“}ACCEPT\text{”} \in \langle P(y), V\rangle(x)] = 1.$$

2. *(*Computational soundness*) For every polynomial time interactive machine P' and for sufficiently large security parameter n, for every sufficiently long $x \notin \mathcal{L}$ and all auxiliary inputs y*

$$\Pr[\text{“}ACCEPT\text{”} \in \langle P(y), V\rangle(x)] \leq \frac{1}{3}.$$

3. *(*Perfect zero-knowledge*) For every verifier V' (with no bound on its computational resources) there is a simulator which is a probabilistic expected polynomial-time machine $M_{V'}$, such that on any positive instance $x \in \mathcal{L}$ and auxiliary input y for the prover and h for the verifier, the output $M_{V'}$ produces given x and h, the random variable $SIM_{V'}(x, h)$, is distributed identically to $\langle P(y), V'(h)\rangle(x)$.*

As is the case in general, the $(1/3, 2/3)$ gap can be made arbitrary large by *sequentially* repeating the protocol[1]. The major result we are interested in is that it is possible to obtain perfect zero-knowledge arguments given an information-theoretic secure bit commitments. We state the non-uniform version of the result. As mentioned in the beginning of Section 2, obtaining a uniform result can be done following [12].

Theorem 1 *([6, 18]) If non-uniform perfectly-secure computationally binding commitment exist, then every language $\mathcal{L} \in NP$ has a perfect zero-knowledge argument with perfect completeness.*

[1] It was recently shown that when an argument is repeated in *parallel* the gap does not necessary decrease [2].

3 Perfectly-Secure Simulatable Bit Commitment

We present a perfectly-secure bit commitment scheme and a proof of its security. To get the intuition, consider the following protocol:

- The sender $\mathcal{S}$ selects a random $x \in \{0,1\}^n$ and computes $y = f(x)$.
- The receiver $\mathcal{R}$ chooses a 2-to-1 hash function $h : \{0,1\}^n \mapsto \{0,1\}^{n-1}$ and sends its description to $\mathcal{S}$.
- $\mathcal{S}$ sends $w = h(y)$.
- At this point, from the receiver's point of view there are exactly two possible values for y, denoted y_0 and y_1 (i.e. $h(y_0) = h(y_1) = w$ and $y_0 < y_1$). Let $y = y_c$. To commit to b, the sender sends $d = b \oplus c$.
- To reveal b, $\mathcal{R}$ sends $x = f^{-1}(y)$.

As long as h is guaranteed to be 2-to-1, then it is equally likely that $y = y_0$ and $y = y_1$ so the security of $\mathcal{S}$ is maintained. I.e. even if R chooses h adverserialy, for any h which is 2-to-1, given $w = h(y)$ the probability that $y = y_0$ or $y = y_1$ is the same over $\mathcal{S}$ coin-flips. Therefore the distribution of (w, d) is independent of the value of b. If h is "random" enough (pair-wise independence is sufficient), then y is paired with a random y' and hence the chances that $\mathcal{S}$ may find $f^{-1}(y')$ are low. However, if $\mathcal{S}$ chooses y only *after* it learns of h, then it may be feasible to find x_0 and x_1 such that $h(f(x_0)) = h(f(x_1))$. Indeed, this is the case, unless $h \circ f$ is a *collision intractable* hash function, which we do not know how to construct under the assumption that one-way permutations exist[2].

In order to take care of "late choosers", the above protocol is refined and the hash function is disclosed gradually, in return for bits of information regarding y. The hash function is defined by an $(n-1) \times n$ binary matrix H of rank $n-1$ over $GF[2]$ and $h(x) = Hx$. The rows of H are revealed step by step, and in response for each row $\mathcal{S}$ sends the inner product of y and the row. The rest of the protocol is as above. We call this technique "interactive hashing". We note that a similar idea was proposed independently in a full information setting by Goldreich, Goldwasser and Linial [14].

Though a devious $\mathcal{S}$ cannot be forced to choose y at the beginning of the protocol, what we show is that there is enough freedom in $\mathcal{R}$'s movements that $\mathcal{S}$ can be forced (with non-negligible probability) to pair y with an arbitrary y'.

3.1 The Scheme

Let f be a strong one-way permutation on $\{0,1\}^n$. Let $\mathcal{S}$ denote the sender and $\mathcal{R}$ the receiver. In the beginning of the protocol, $\mathcal{S}$ is given a secret input bit b. $B(x, y)$ denotes the dot-product mod 2 of x and y.

Commit Stage.

Commit to a bit b.

1. The sender $\mathcal{S}$ selects $x \in_R \{0,1\}^n$ at random and computes $y \leftarrow f(x)$. $\mathcal{S}$ keeps both x and y secret from $\mathcal{R}$.
2. The receiver $\mathcal{R}$ selects $h_1, h_2, \ldots h_{n-1} \in \{0,1\}^n$ such that each h_i is a random vector over $GF[2]$ of the form $0^{i-1}1\{0,1\}^{n-i}$ (i.e. $i-1$ 0's followed by a 1 followed by an arbitrary choice for the last $n-i$ positions). Note that $h_1, h_2, \ldots h_{n-1}$ are linearly independent over $GF[2]$. We call $h_1, h_2, \ldots h_{n-1}$ $\mathcal{R}$'s queries.

[2]If f is indeed collision intractable, the resulting scheme is very close to the one proposed in [31] or [11].

3. For j from 1 to $n-1$
   - $\mathcal{R}$ sends h_j to $\mathcal{S}$.
   - $\mathcal{S}$ sends $c_j \leftarrow B(h_j, y)$ to $\mathcal{R}$.
4. At this point there are exactly two vectors $y_0, y_1 \in \{0,1\}^n$ such that for both $i \in \{0,1\}$, $c_j = B(y_i, h_j)$ for all $1 \le j \le n-1$. Define y_0 to be the lexicographically smaller of the two vectors. Both $\mathcal{S}$ and $\mathcal{R}$ compute y_0 and y_1 by solving the linear system[3]. Let $c \in \{0,1\}$ be such that $y = y_c$ (only $\mathcal{S}$ knows c).
5. $\mathcal{S}$ computes $d = b \oplus c$ and sends it to R.

Reveal Stage.

The receiver's $\mathcal{R}$'s input from the commit stage is $c_1, c_2, \ldots c_{n-1}$ and d, as well as $\mathcal{R}$'s queries $h_1, h_2, \ldots h_{n-1}$

1. $\mathcal{S}$ sends b and x to $\mathcal{R}$.
2. $\mathcal{R}$ verifies that $y = f(x)$ obeys $c_j = B(h_j, y)$ for all $1 \le j \le n-1$ and verifies that $y = y_c$ where $c = d \oplus b$.

3.2 Proof of security

Theorem 2 If f is a one-way permutation, then the scheme presented in Section 3.1 is a perfectly-secure computationally-binding bit commitment scheme.

Theorem 2 follows from the lemmata below, the Security Lemma and the Binding Lemma, respectively (the viability and efficiency of the scheme can be verified easily). The proof of the Security Lemma is relatively straightforward, but the Binding Lemma turned out to be trickier and required a delicate proof.

Lemma 1 (Security) *For* any *receiver $\mathcal{R}'$, the distribution of the conversations at the* commit *stage is independent of the value of the bit b.*

Proof : We show inductively on j, that for any choice of $h_1, h_2, \ldots h_j$ the conditional distribution of y given $h_1, h_2, \ldots h_j$ $c_1, c_2, \ldots c_j$ is uniform in the subspace defined by $h_1, h_2, \ldots h_j$ and $c_1, c_2, \ldots c_j$. The inductive step holds, since the linear independence of $h_1, h_2, \ldots, h_j$ implies that

$$\Pr[B(h_j, y) = 0 | h_1, h_2, \ldots h_{j-1}, c_1, c_2, \ldots, c_{j-1}] = \frac{1}{2}.$$

Thus, at Step 4 the probability that $c = 0$ (i.e. $y = y_0$) is exactly $\frac{1}{2}$, as y is distributed uniformly in $\{y_0, y_1\}$. Therefore, for any method of choosing the queries the distribution of

$$(h_1, h_2, \ldots h_{n-1}, c_1, c_2, \ldots, c_{n-1}, d)$$

is the same when $b = 0$ and $b = 1$. $\square$

Recall that we consider a cheating sender to be successful if following the commit stage it can make the receiver accept two different values as the bit committed. In our protocol that means that the cheating sender can find $x_0, x_1 \in \{0,1\}^n$ such that $x_0 \neq x_1$ but $y_0 = f(x_0)$ and $y_1 = f(x_1)$ are both consistent with $h_1, \ldots, h_{n-1}$ and $c_1, \ldots, c_{n-1}$. The "Binding" Lemma below states that if there exists a sender that can cheat with non-negligible probability, then it can be used to invert the presumed one-way permutation f on a non-negligible fraction of the inputs, contradicting our assumption.

[3]The way the queries are chosen implies that solving the system can be done in $O(n^2)$ time

Lemma 2 (Binding) *Assume there exists a probabilistic polynomial time $\mathcal{S}'(n)$ that following the commit stage can reveal to a honest receiver two different values for b with non-negligible probability $\varepsilon = \varepsilon(n)$ where the probability is over $\mathcal{S}'$ and the Receiver $\mathcal{R}$ coin flips. Then there exists a probabilistic polynomial time algorithm $\mathcal{A}$ that inverts f on non-negligible fraction of the y's in $\{0,1\}^n$.*

Proof : We describe how to construct an algorithm $\mathcal{A}$ for inverting f whose run time is larger than $\mathcal{S}'$'s by at most a $p(n, \frac{1}{\varepsilon})$ multiplicative factor and its probability of success in computing $f^{-1}(y)$ for $y \in_R \{0,1\}^n$ is at least $1/p(n, \frac{1}{\varepsilon})$ where p is some (fixed) polynomial.

We begin by making $\mathcal{S}'$ deterministic which can be done using standard techniques. Suppose that we choose an assignment to the random tape of $\mathcal{S}'$ and count the number of queries of $\mathcal{R}$ (i.e. $h_1, \ldots h_{n-1}$) on which $\mathcal{S}'$ succeeds in cheating. By assumption, if the assignment is random, then the expected fraction of such queries is at least ε. Let Ω be the set of assignments on which $\mathcal{S}'$ is successful on at least $\varepsilon/2$ of $\mathcal{R}$'s queries. By a simple counting argument we can conclude that Ω consists of at least $\frac{\varepsilon}{2}$ of the possible assignments. The algorithm $\mathcal{A}$ described below requires $\mathcal{S}'$ to be deterministic. Therefore we choose $m = \frac{2n}{\varepsilon}$ random assignments $\omega_1, \omega_2, \ldots, \omega_m$ and run m times the algorithm $\mathcal{A}$ with the random tape of $\mathcal{S}'$ initialized with $\omega_1, \omega_2, \ldots, \omega_m$. With probability $1 - (1 - \frac{\varepsilon}{2})^m \geq 1 - e^{-n}$ some $\omega_i \in \Omega$. Therefore from now on we assume that $\mathcal{S}'$ is deterministic and its probability of success over $\mathcal{R}$'s queries is at least $\frac{\varepsilon}{2}$.

Let T be the rooted tree of depth $n-1$ defined by the queries sent by $\mathcal{R}$. A node U_i at the ith level is defined by queries $h_1, h_2, \ldots, h_{i-1}$ where for all $1 \leq k \leq i-1$ the query h_k is of the form $0^{k-1}1\{0,1\}^{n-k}$. Each of U_i's 2^{n-i} outgoing edges corresponds to a query $\mathcal{R}$ may send in the ith round of the form $0^{i-1}1\{0,1\}^{n-i}$ and leads to a different node at the $(i+1)$th level. The behavior of $\mathcal{S}'$ specifies a labeling of the edges of T with $\{0,1\}$. For a node U_i defined by queries $h_1, h_2, \ldots, h_{i-1}$ the label of an edge h_i is the response c_i of $\mathcal{S}'$ to the query h_i in the ith round, given that the previous queries were $h_1, h_2, \ldots, h_{i-1}$. We denote it by $L_{\mathcal{S}'}(U_i, h_i)$. Given that $\mathcal{S}'$ is deterministic and that $\mathcal{A}$ has complete control over it, it is possible to compute this labeling.

For a leaf U_n defined by queries $h_1, h_2, \ldots h_{n-1}$, let $U_1, U_2, \ldots U_{n-1}$ be the nodes on the path from the root to U_n and let $\{y_0(U_n), y_1(U_n)\}$ be the set of images consistent with the labeling of $\mathcal{S}'$, i.e. $L_{\mathcal{S}'}(U_i, h_i) = B(y_b, h_i)$ for all $1 \leq i \leq n-1$ and $b \in \{0,1\}$. We say that the leaf U_n is good if given that $\mathcal{R}$'s queries $h_1, h_2, \ldots h_{n-1}$, then $\mathcal{S}'$ succeeds in opening the bit committed in two different ways: i.e. $\mathcal{S}'$ inverts both $y_0(U_n)$ and $y_1(U_n)$.

In general, given y, $\mathcal{A}$'s strategy is to try to find a good leaf U_n such that the labels $L_{\mathcal{S}'}$ on the edges leading to it are consistent with y, i.e. $y \in \{y_0(U_n), y_1(U_n)\}$. If U_n is indeed good, then it yields the inverses of $y_0(U_n)$ and $y_1(U_n)$ and hence of y. Such a leaf is found by developing the path node by node. Intuitively, for any labeling of T at any node U_i and for a y that is consistent with the labels leading to U_i the probability that $B(h_i, y) = L_{\mathcal{S}'}(U_i, h_i)$ for a random query h_i is $1/2$ (the intuition is that an inner product of random vector with two different vectors yields independent results). Therefore to find a node U_{i+1} consistent with y should take on the average two inspections of random h_i's. However, an important thing to note is that since $\mathcal{S}'$ may be cheating, its answers need not be consistent and that on the same query h_i the sender $\mathcal{S}'$ may give different answers depending on the previous queries. Therefore the above intuition is not accurate and this is the source of the difficulty in constructing and analyzing the invertor $\mathcal{A}$. Roughly speaking, we must use the randomness of y itself to argue that the label of a random h has a fair chance of agreeing with $B(h, y)$. We should also not "waste" this randomness too quickly, before getting close enough to a leaf.

Description of the inverting algorithm $\mathcal{A}$:

Recall our notation: $B(h, y)$ denotes the inner product of h and y, U_i is a node of level i defined by queries $h_1, h_2, \ldots h_{i-1}$ and $L_{\mathcal{S}'}(U_i, h_i)$ is the answer of $\mathcal{S}'$ on h_i, given that the previous queries were $h_1, h_2, \ldots h_{i-1}$.

$\mathcal{A}$ gets as an input a random image y in $\{0,1\}^n$ and it attempts to invert y. In order to compute $f^{-1}(y)$, $\mathcal{A}$ tries to find a good leaf u such that $y \in \{y_0(u), y_1(u)\}$. Obviously, if it finds such a leaf it can succeed in inverting y. Starting at the root, A develops node by node a path consistent with y. Fix j to be $n - 8(\log \frac{n}{\varepsilon} + 2)$. The algorithm $\mathcal{A}$ consists of $j-1$ rounds.

The state of $\mathcal{A}$ at the beginning of the ith round ($1 \leq i < j$) can be described by a node U_i of the ith level of the tree T defined by queries $h_1, h_2, \ldots h_{i-1}$. Let $U_1, U_2, \ldots U_{i-1}$ be the path from the root to U_i. The property that $\mathcal{A}$ maintains is that the and the labels $c_1, c_2, \ldots c_{i-1}$ along the path are consistent with y, i.e. for all $1 \leq k \leq i-1$ we have $c_k = L_{\mathcal{S}'}(U_k, h_k) = B(h_k, y)$.

At the ith round $\mathcal{A}$ performs the following: a random query $h \in_R \{h | h = 0^{i-1}1\{0,1\}^{n-i}\}$ is chosen. If the outgoing edge h is labeled properly, i.e. $L_{\mathcal{S}'}(U_i, h) = B(h, y)$, then $h_i \leftarrow h$ and the path is expanded to the new node U_i led by h_i. Otherwise, $\mathcal{S}'$ is reset to the state before its reply, and a new candidate for h_i is chosen. This is repeated until either a success or until there are no more candidates left, in which case $\mathcal{A}$ aborts.

If $\mathcal{A}$ reaches the jth level, it guesses the remaining $n-j$ queries $h_j, h_{j+1}, \ldots, h_{n-1}$ by choosing them uniformly at random from the proper sets of queries. $\mathcal{A}$ then checks whether the path to the leaf is labeled consistently with $B(y, h_k)$ for $k = j, \ldots, n-1$. If this is the case and the leaf reached is good, then $\mathcal{A}$ has succeeded in inverting y. Otherwise abort.

Analysis of the inverting algorithm $\mathcal{A}$

The rest of this proof is devoted to showing that $\mathcal{A}$ as defined above has probability at least $\frac{\varepsilon^{10}}{65e^3(4n)^8}$ for inverting y. Note that $\mathcal{A}$ as described above does not necessarily halt after a polynomial number of steps. However, as we shall see following Claim 7, we can limit the *total* number of unsuccessful attempts at finding consistent h's to $8n$ without decreasing significantly the probability that $\mathcal{A}$ succeeds in inverting y.

Notation: since we are dealing with several types of vectors of length n over $GF[2]$ we will distinguish them by referring to those vectors that are sent by $\mathcal{R}$ as queries, and to those vectors which may be the image that $\mathcal{A}$ attempts to invert as images. Let U_i be a node at the ith level of the tree T defined by $h_1, h_2, \ldots h_{i-1}$ and let $c_1, c_2, \ldots c_{i-1}$ be the labels $L_{\mathcal{S}'}$ assigned to the path to U_i. We say that $y \in \{0,1\}^n$ is an image in U_i if $B(h_k, y) = c_k$ for all $1 \leq k < i$. We denote the set of images of U_i by $\mathcal{I}(U_i)$; we know that $|\mathcal{I}(U_i)| = 2^{n-i+1}$. We say that $h \in \{0,1\}^n$ is a query of U_i if it is of the form $0^{i-1}1\{0,1\}^{n-i}$. There are 2^{n-i} queries at a node U_i of the ith level.

Let $A(U, y) = |\{h : h \text{ is a query of } U \text{ and } B(h, y) = L_{\mathcal{S}'}(U, h)\}|$. An image y is balanced in U_i, a node of the ith level if

$$\frac{1}{2}\left(1 - \frac{1}{n}\right) \leq \frac{A(U_i, y)}{2^{n-i}} \leq \frac{1}{2}\left(1 + \frac{1}{n}\right)$$

Hence for an image y that is balanced in U_i, roughly half of the answers to the queries at node U_i agree with y. An image y is fully balanced in U, a node of the jth level, if it is balanced in all the ancestors of U. Let $\mathcal{F}(U)$ be the set of $y \in \mathcal{I}(U)$ that are fully balanced in U. The motivation for considering fully balanced images is that the probability that $\mathcal{A}$ reaches a certain node U with an image $y \in \mathcal{F}(U)$ is close to what it would be in case $\mathcal{S}'$ was honest. For a set of queries H at a node U and an image y of U the discrepancy of y at H is

$$\left| |\{h \in H : L_{\mathcal{S}'}(U, h) = B(y, h)\}| - \frac{|H|}{2} \right|$$

i.e. the difference between the "expected" number of agreeing queries and the actual number of queries in H that agree with y. Finally, recall that $j = n - 8(\log \frac{n}{\varepsilon} + 2)$ and set $\gamma = n 2^{-\frac{5}{8}(n-j)}$.

Roadmap: Our main problem in analyzing algorithm $\mathcal{A}$ is in showing that no labeling $L_{\mathcal{S}'}$ can bias the walk towards a set of leaves containing a small subset of the images. Claims 1 and 2 show that for any labeling

$L_{\mathcal{S}'}$, for any node U almost all the images of U are fully balanced. The motivation for considering fully balanced images is expressed in Claim 3 by showing that the probability of $\mathcal{A}$ reaching a certain node U with an image $y \in \mathcal{F}(U)$ is close to what it would be in case $\mathcal{S}'$ was honest. This is also the reason $\mathcal{A}$ stops testing queries at level j and continues further by guessing the rest of the sequence: otherwise the nodes may be unbalanced and the probabilities too biased.

Though initially a non-negligible fraction of the leaves are good, there is a danger that $\mathcal{S}'$ leads $\mathcal{A}$ to those directions that have only few good leaves. Claims 4 and 5 say that this is not the case and that with reasonable probability when $\mathcal{A}$ reaches the jth level it has many good leaves whose images are fully balanced. Claim 6 implies that the probability that our random guess is correct is not far from being inversely proportional to the number of leaves of a subtree rooted at level j (which is polynomial). Finally, Claim 7 combines all the above to show that the probability of success is non-negligible.

Claim 1 *Let U be node of the ith level and let $H \subset \{h | h = 0^{i-1}1\{0,1\}^{n-i}\}$ be a subset of the queries of U of size at most 2^{n-j}. For any $h \in H$ let a_h be a random variable over $z \in_R \mathcal{I}(U)$ such that $a_h = 1$ if $B(h,z) = L_{\mathcal{S}'}(U,h)$ and 0 otherwise. Then,*

$$\Pr\left[\left|\sum_{h\in H} a_h - \frac{1}{2}|H|\right| \geq 2^{\frac{7}{8}(n-j)}\right] \leq 2^{-\frac{3}{4}(n-j)}.$$

Proof : First note that any pair of queries different $h', h'' \in H$ has the property that h'' is linearly independent of $h', h_1, h_2, \ldots h_{i-1}$. For any $h \in H$ we have that $\Pr[a_h = 1] = \frac{1}{2}$ and $\mathrm{Var}[a_h] = \frac{1}{4}$. For every $h' \neq h''$ the events $a_{h'}$ and $a_{h''}$ are pairwise independent (this follows from the linear independence of h' and h'') and hence

$$Var[\sum_{h\in H} a_h] = \frac{1}{4}\,|H| \leq 2^{n-j-2}.$$

We are essentially interested in

$$\Pr\left[\left|\sum_{h\in H} a_h - E[\sum_{h\in H} a_h]\right| \geq 2^{7/8(n-j)}\right] \tag{1}$$

since $E[\sum_h a_h] = \frac{1}{2}|H|$. By Chebyschev's inequality

$$\Pr\left[\left|\sum_{h\in H} a_h - E[\sum_{h\in H} a_h]\right| \geq \lambda\sqrt{Var\left[\sum a_h\right]}\right] \leq \frac{1}{\lambda^2}$$

Taking $\lambda = 2^{\frac{3}{8}(n-j)}$ we get that (1) is at most $2^{-\frac{3}{4}(n-j)}$. $\square$

Claim 2 *For any node U_j of level j and random $z \in \mathcal{I}(U_j)$ we have $\Pr[z \in \mathcal{F}(U_j)] \geq 1 - \gamma$ for $\gamma = n2^{-\frac{5}{8}(n-j)}$*

Proof : Let $U_1, U_2, \ldots U_{j-1}$ be the nodes on the path to U_j. We should show that for any U_i along the path most $z \in \mathcal{I}(U_j)$ are balanced. We cannot apply Claim 1 directly, since a random $y \in \mathcal{I}(U_j)$ is not random in $\mathcal{I}(U_i)$. To apply the Claim, we first take care of the queries of U_i that are *not* linearly independent of $h_i, \ldots h_{j-1}$. There are at most 2^{j-i} (out of 2^{n-i}) such queries and we (pessimistically) count them as contributing to the discrepancy. Let H' be the remaining queries of U_i. We partition them into 2^{j-i} subsets according to the values of bits $i+1$ through j. For each $\ell \in \{0,1\}^{j-i}$ let $H_\ell = \{h | h = 0^{i-1}1\ell\{0,1\}^{n-j}\} \cap H'$. Each H_ℓ is of size at most 2^{n-j} and has the following the important property.

Fact 1 *For every different $h', h'' \in H_\ell$ we have that $h_i, \dots h_{j-1}, h', h''$ are linearly independent.*

Proof : In any subset of $h_i, \dots h_{j-1}, h', h''$ that sums to $\vec{0}$ an even number of elements out of h_i, h', h'' must participate. Since h' and h'' are linearly independent of $h_i, \dots, h_{j-1}$, it is the case that h_i does not participate in the sum. However, since $h', h'' \in H_\ell$ and have the same bits in location i through j, no member of $h_{i+1}, \dots, h_{j-1}$ can participate in the sum. Since $h' \neq h''$ no vector from $h_i, \dots h_{j-1}, h', h''$ participates and we get the desired linear independence $\square$

Given this property we have that for $h', h'' \in H_\ell$ and a random $z \in_R \mathcal{I}(U_j)$ the random variables $a_{h'}$ and $a_{h''}$ are independent. Therefore, as in the proof of Claim 1 we have that for any $\ell \in \{0,1\}^{j-i}$

$$\Pr\left(\left|\sum_{h \in H_\ell} a_h - E[\sum_{h \in H_\ell} a_h]\right| > 2^{\frac{7}{8}(n-j)}\right) \leq 2^{-\frac{3}{4}(n-j)}. \tag{2}$$

Let b_ℓ be the indicator for the event $\left|\sum_{h \in H_\ell} a_h - E[\sum_{h \in H_\ell} a_h]\right| > 2^{\frac{7}{8}(n-j)}$. From (2) we know $\Pr[b_\ell] \leq 2^{-\frac{3}{4}(n-j)}$. By Markov's inequality we can conclude that

$$\Pr\left[\sum_{\ell \in \{0,1\}^{j-i}} b_\ell > \frac{2^{j-i}}{2^{\frac{1}{8}(n-j)}}\right] \leq 2^{-\frac{5}{8}(n-j)}.$$

That is, the probability that for more than a fraction $2^{-1/8(n-j)}$ of the ℓ's, the set H_ℓ has a discrepancy larger than $2^{\frac{7}{8}(n-j)}$ is at most $2^{-\frac{5}{8}(n-j)}$. Thus with probability at least $1 - 2^{-\frac{5}{8}(n-j)}$ the total discrepancy at node U_i is at most

$$2^{j-i} + 2^{-1/8(n-j)} 2^{n-j} 2^{j-i} + (1 - 2^{-1/8(n-j)}) 2^{\frac{7}{8}(n-j)} 2^{j-i} \leq 2 \cdot 2^{\frac{7n}{8} + \frac{j}{8} - i} = 2^{n-i} \cdot 2^{-\frac{1}{8}(n-j)+1}$$

where the first summand is an upper bound on the contribution of the queries not in H', the second the contribution of the H_ℓ's where $b_\ell = 1$ and the third the contribution of the H_ℓ's where $b_\ell = 0$. Hence for $z \in_R \mathcal{I}(U_j)$ with probability at least $1 - 2^{-\frac{5}{8}(n-j)}$ we have

$$2^{n-i-1} - 2^{n-i} \cdot 2^{-\frac{1}{8}(n-j)+1} \leq A(U_i, z) \leq 2^{n-i-1} + 2^{n-i} \cdot 2^{-\frac{1}{8}(n-j)+1}$$

and since $j = n - 8(\log \frac{n}{\varepsilon} + 2)$

$$\frac{1}{2}(1 - \frac{1}{n}) \leq \frac{1}{2} - 2^{-1/8(n-j)+1} \leq \frac{A(U_i, z)}{2^{n-i}} \leq \frac{1}{2} + 2^{-1/8(n-j)+1} \leq \frac{1}{2}(1 + \frac{1}{n})$$

The probability that z is balanced in all the levels is therefore at least $1 - n2^{-\frac{5}{8}(n-j)} = 1 - \gamma$. $\square$

Claim 3 *For any node U_j of level j and any $z \in \mathcal{F}(U_j)$*

$$\frac{1}{2^n e} \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}} \leq \Pr[\mathcal{A} \textit{ reaches } U_j \textit{ and } y = z] \leq \frac{e}{2^n} \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}}$$

where the probability is uniform over the choices of y and the coin-flips of $\mathcal{A}$

Proof : To get the first inequality,

$$\begin{aligned}
\Pr[\mathcal{A} \text{ reaches } U_j \text{ and } y = z] &= \frac{1}{2^n} \cdot \prod_{i=1}^{j-1} \frac{1}{A(U_i, z)} \\
&\geq \frac{1}{2^n} \cdot \prod_{i=1}^{j-1} \frac{1}{(1+1/n) \cdot 2^{n-i-1}} \\
&\geq \frac{1}{2^n(1+1/n)^n} \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}} \\
&\geq \frac{1}{2^n e} \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}}.
\end{aligned}$$

Similarly, for the second inequality

$$\begin{aligned}
\Pr[\mathcal{A} \text{ reaches } U_j \text{ and } y = z] &= \frac{1}{2^n} \cdot \prod_{i=1}^{j-1} \frac{1}{A(U_i, z)} \\
&\leq \frac{1}{2^n} \cdot \prod_{i=1}^{j-1} \frac{1}{(1-1/n) \cdot 2^{n-i-1}} \\
&\leq \frac{e}{2^n} \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}}. \qquad \square
\end{aligned}$$

Recall that a leaf U_n is *good* if given that R's queries lead to U_n, then $\mathcal{S}'$ succeeds in opening the bit committed in two different ways: i.e. $\mathcal{S}'$ inverts on both $y_0(U_n)$ and $y_1(U_n)$. Since we stop $n-j$ levels above the leaves we are interested in nodes that have many good leaves in the subtree below them. The reason we need many and not just one is that a single good node may not have any of its images in the set of fully balanced images at the root of the subtree. Call an internal node U good if at least $\frac{\varepsilon}{4}$ of the leaves at the subtree rooted at U are good. By assumption, the fraction of good leaves is at least $\frac{\varepsilon}{2}$. Therefore, the fraction of good nodes among those of any fixed level and in particular the jth level is at least $\frac{\varepsilon}{4}$, since all of them have the same number of leaves.

Claim 4 *The probability that $\mathcal{A}$ reaches some good node U_j of the jth level and $y \in \mathcal{F}(U_j)$ is at least $\frac{\varepsilon(1-\gamma)}{4e}$ where the probability is over the choice of y (the image $\mathcal{A}$ attempts to invert) and the coin-flips of $\mathcal{A}$.*

Proof : Let U_j be a good node of the j th level. Then,

$$\begin{aligned}
\Pr[\,\mathcal{A} \text{ reaches } U_j \text{ and } y \in \mathcal{F}(U_j)] &= \sum_{z \in \mathcal{F}(U_j)} \Pr[y = z \; and \; \mathcal{A} \; reaches \; U_j] \\
&\geq \sum_{z \in \mathcal{F}(U_j)} \frac{1}{e2^n} \cdot \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}} \\
&\geq \frac{2^{n-j+1}(1-\gamma)}{e2^n} \cdot \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}} \\
&= \frac{(1-\gamma)}{e} \prod_{i=1}^{j-1} \frac{1}{2^{n-i}}
\end{aligned}$$

Where the first inequality follows from Claim 3 and the second from Claim 2. Since there are $\prod_{i=1}^{j-1} 2^{n-i}$ nodes at the jth level and at least a fraction $\frac{\varepsilon}{4}$ of them are good, the probability that the image chosen is fully balanced at a good node of the jth level is at least $\frac{\varepsilon(1-\gamma)}{4e}$. $\square$

Claim 5 *In any good node U_j of level j the fraction of the good leaves at the subtree rooted in U_j that have at least one image in $\mathcal{F}(U_j)$ is at least $\frac{\varepsilon}{8}$.*

Proof : Any pair of images $y_1 \neq y_2$ in $\mathcal{I}(U_j)$ can be together in at most $1/2^{n-j}$ of the leaves of the subtree rooted at U_j: in any node U' along the way from U to the leaves and for random query h of U' we have $\Pr[B(h, y_1) = B(h, y_2)] = 1/2$. By Claim 2 there are at most $\gamma 2^{n-j+1}$ images in $\mathcal{I}(U_j)$ that are not fully balanced in U_j. Therefore the fraction of the leaves of the subtree rooted in U_j where both of their images are from $\mathcal{I}(U_j) \setminus \mathcal{F}(U_j)$ is bounded by

$$\binom{\gamma 2^{n-j+1}}{2} \cdot \frac{1}{2^{n-j}}$$

(i.e. the number of pairs of images from $\mathcal{I}(U_j) \setminus \mathcal{F}(U_j)$ times the fraction of leaves they can appear together). Since

$$\binom{\gamma 2^{n-j+1}}{2} \frac{1}{2^{n-j}} \leq 2\gamma^2 2^{n-j} = n^2 2^{-\frac{1}{4}(n-j)+1} = n^2 2^{-2(\log n/\varepsilon+2)+1} \leq \frac{\varepsilon^2}{8},$$

we have that at least $\frac{\varepsilon}{4} - \frac{\varepsilon^2}{8} \geq \varepsilon/8$ of the leaves are both good and have at least one image in $\mathcal{F}(U_j)$. $\square$

Claim 6 *For any good node U_j of level j and any $z \in \mathcal{F}(U_j)$, given that $\mathcal{A}$ reaches U_j and $y \in \mathcal{F}(U_j)$, the probability that $y = z$ is at least $\frac{1}{e^2 2^{n-j+1}}$ where the probability is over the choice y and the coin-flips of $\mathcal{A}$.*

Proof : For fixed U_i and $z \in \mathcal{F}(U_j)$ we would like to bound from below the value:

$$\frac{\Pr[\mathcal{A} \text{ reaches } U_j \text{ and } y = z]}{\Pr[\mathcal{A} \text{ reaches } U_j \text{ and } y \in \mathcal{F}(U_j)]}. \tag{3}$$

We know from the first inequality of Claim 3 that

$$\begin{aligned}
\Pr[\mathcal{A} \text{ reaches } U_j \text{ and } y \in \mathcal{F}(U_j)] &= \sum_{y' \in \mathcal{F}(U_j)} \Pr[\mathcal{A} \text{ reaches } U_j \text{ and } y = y'] \\
&\leq |\mathcal{F}(U_j)| \cdot \frac{e}{2^n} \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}} \\
&\leq |\mathcal{I}(U_j)| \cdot \frac{e}{2^n} \cdot \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}} \\
&\leq \frac{e \cdot 2^{n-j+1}}{2^n} \cdot \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}}.
\end{aligned}$$

On the other hand, from the second inequality of Claim 3, for any $z \in \mathcal{F}(U)$ we have that

$$\Pr[\mathcal{A} \text{ reaches } U_j \text{ and } y = z] \geq \frac{1}{e 2^n} \prod_{i=1}^{j-1} \frac{1}{2^{n-i-1}}$$

Therefore (3) is at least $\frac{1}{e^2 2^{n-j+1}}$. $\square$

Claim 7 *The probability that $\mathcal{A}$ is successful is at least $\frac{\varepsilon^{10}}{65e^3(4n)^8}$ where the probability is over the choice of the image y and $\mathcal{A}$ coin-flips.*

Proof : Define the events: (a) $\mathcal{A}$ reaches a good node U at level j and that $y \in \mathcal{F}(U)$ and (b) that $h_j, h_{j+1}, \ldots h_{n-1}$ define a path to a good leaf that has at least one image in $\mathcal{F}(U)$. Call this image z (select arbitrarily if both images are in $\mathcal{F}(U)$). If $y = z$ then $\mathcal{A}$ is successful. By Claim 6 we know that the probability that $y = z$ is at least $\frac{1}{e^2 2^{n-j+1}}$. The probability that (a) occurs is at least $\frac{\varepsilon(1-\gamma)}{4e}$ by Claim 4 and that (b) occurs given (a) is at least $\frac{\varepsilon}{8}$ by Claim 5. Therefore the probability that $\mathcal{A}$ succeeds is at least

$$\frac{\varepsilon(1-\gamma)}{4e} \cdot \frac{\varepsilon}{8} \cdot \frac{1}{e^2 2^{n-j+1}} = \varepsilon^2 \frac{(1-\gamma)}{32 \cdot e^3 \cdot 2^{n-j+1}} > \frac{\varepsilon^{10}}{65e^3(4n)^8}$$

where the last inequality follows from the fact that $j = n - 8(\log \frac{n}{\varepsilon} + 2)$. $\Box$

Note that we have considered $\mathcal{A}$ successful when y was fully balanced at level j, without taking into account the time it took for $\mathcal{A}$ to arrive at this position. However, given that y is fully balanced at level j, the probability that $\mathcal{A}$ had many unsuccessful candidates until it reached the jth level is small: we know that y is balanced at U_i for all $1 \leq i < j$ and therefore $A(U, y)/2^{n-i} > \frac{1}{4}$. Therefore the probability that $\mathcal{A}$ had to try more (in total) than $8n$ candidates for the h_i's until reaching level j is exponentially small in n. If we bound the run time of $\mathcal{A}$ by $8n^2$ (including the query time), then the probability of success is still at least $\frac{\varepsilon^{10}}{65e^3(4n)^8} - \exp(-n)$. If ε is non-negligible, then this is non-negligible as well. This concludes the Proof of Lemma 2 and Theorem 2 . $\Box$

3.3 Obtaining perfect ZK arguments

We have shown a uniform reduction from the existence of a one-way permutation to the existence of perfectly-secure computationally binding bit commitment protocols. The result holds in the non-uniform setting as well. Therefore, applying Theorems 1 and 2 we get

Corollary 1 If any non-uniformly secure one-way permutation exists, then there exist perfect zero-knowledge arguments for proving membership for all languages in NP.

4 Concluding remarks and possible extensions

We now review some technical and general issues arising from this work.

Probability of success: In the proof of the Binding Lemma we did not attempt to optimize the probability of success as a function of ε and the resulting polynomial is of rather high degree. However it seems that our method of designing algorithm $\mathcal{A}$ does not yield success probability that is linear in ε. It is interesting whether we can get the dependency to be *linear* in ε times some polynomial in n. This would make the reduction *linear preserving* in Luby's [28] terminology, whereas the current one is only *polynomial preserving*.

One-way permutations vs. functions: Where is the assumption that f is a permutation used? First it is needed for the Secrecy Lemma, in order to argue that $c_1, c_2, \ldots, c_{n-1}$ yield no information about y. Consider the case where f is an *almost* permutation, that is all but a negligible fraction of the strings in $\{0,1\}^n$ have exactly one pre-image.

Call a leaf u *secure* if both $y_0(u)$ and $y_1(u)$ have exactly one pre-image. If $\mathcal{R}$ and $\mathcal{S}$ reach a secure leaf, then $\mathcal{R}$ cannot guess b with probability better than $\frac{1}{2}$. Initially most leaves are secure, and δ, the fraction of insecure leaves is negligible. However, a devious receiver $\mathcal{R}'$ may bias the fraction of insecure leaves by its queries. Let $\delta = \delta_1, \delta_2, \ldots, \delta_{n-1}$ be the fractions of insecure leaves at an execution of the commit protocol. Suppose that c_i is random. Then for any strategy of $\mathcal{R}'$ the expected value of δ_{i+1} is δ_i and therefore $E[\delta_{n-1}] = \delta$. From Markov's inequality it follows that $\Pr[\delta_{n-1}] \neq 0$ is negligible. Note however that the c_i's are not quite random. Nevertheless, we can define a property similar to *balanced* that assures us that c_i is not far from being uniform in $\{0, 1\}$ and thus obtain the desired security property.

As for the binding requirement, the difference between the case where f is a permutation and an almost permutation is that y is not necessarily uniform in $\mathcal{I}(U)$, given that $\mathcal{A}$ reaches U. However, by a similar argument to the balanced property, with high probability the conditional distribution of y is not far from uniform in $\mathcal{I}(U)$.

In case f is a general one-way function the above arguments may fail miserably. For starters, most leaves will have the property that the number of pre-images y_0 and y_1 are different. Then there is the danger that a devious $\mathcal{R}'$ will skew the probability even further, making the guess of b extremely easy (so that even splitting b into $b = b_1 \oplus b_2 \oplus \ldots b_n$ would be futile).

Dynamic adversaries: We point out another advantage of perfectly-secure computationally-binding bit commitments (over computationally secure ones). Consider the following scenario which is a variant of one proposed by Oded Goldreich (personal communication) in order to model dynamic adversaries. There are n senders and receivers who perform a bit commitment protocol. The input bits given to the senders are drawn according to some joint distribution on which there is some auxiliary information. The commitments are performed separately and independently, but following the commit stage an adversary may decide (based on the communication exchanged) to "corrupt" $n/2$ of the senders who provide him with all their internal information, including the random string used in the protocol. The question is whether the remaining $n/2$ bits are still protected as they before. Since the bits may be related, the proper comparison should be with an weaker adversary that does not get to see the messages exchanged during the commit stage, but can ask to get the *value* of $n/2$ bits. Whatever the strong adversary can compute on the n bits should be computable by the weaker adversary (the computational power of both adversaries should be similar).

Intuitively, this should be the case, since the n parties act independently. However, attempts to prove this have been futile in case the bit commitment is *computationally secure*; the problem is in running a simulation, since the adversary gets to see the commitments *before* it decides which parties to corrupt, and the simulation is polynomially bounded. On the other hand, for perfectly-secure bit commitment it is the case that the remaining $n/2$ bits are protected information theoretically. The reason is that the messages sent during the commit stage are independent of the actual value of the bits, so a computationally powerful simulator may use the strong adversary to create a weak one (which in this case both of them are computationally unbounded).

Other applications of interactive hashing: The technique of interactive hashing presented here were useful in constructing fail-stop signatures [11] by replacing a collision-free one-way hash functions, and in designing zero-knowledge proofs from honest-verifier zero-knowledge proofs [34, 10]. It would be interesting to know if further applications of the techniques to reduction of computational complexity assumptions are possible.

One plausible scenario is replacing collision intractable hash functions used in the work of Kilian [25] and Micali [29] in order to reduce the communication complexity of NP arguments. Essentially, what is needed there is a commitment to a large string whose communication complexity is much smaller than the length of the string. Our protocol requires n^2 bits of communication in order to commit to a single bit, so it may seem not applicable to this problem. Note however that in case we use our protocol to commit to many bits, the queries $\mathcal{R}$ sends may be shared among the bit commitments giving us amortized complexity close to $n - 1$, still far from the desired $o(1)$.

Suppose that we give up the information-theoretic security of $\mathcal{S}$ and go for computational binding *and* security (i.e. both parties are protected "only" computationally). In this case, consider the following protocol: the sender commits to a seed of a pseudo-random sequence using a computationally secure scheme such as [30]. The bit-wise Xor of the pseudo-random sequence and the string is still pseudo-random and computationally protects the string. This Xored sequence is then partitioned into blocks of size n. Each of these blocks is then used as the x's in our protocol of Section 3.1. I.e. the commiter computes $f(x)$ and replies to $n-1$ successive queries $h_1, h_2, \ldots, h_{n-1}$ with $B(f(x), h_i)$. Steps 4 and 5 are not executed, since the commitment is really to x itself. As suggested above, the receiver's queries are shared between the blocks. To open the commitment the seed is revealed along with all the blocks (the x's). This yields amortized communication complexity for the commit phase of roughly $1-\frac{1}{n}$ per bit of the *original* string. Reducing the amortized communication complexity to $o(1)$ seems to be challenging.

Finally, an interesting question is whether the highly interactive nature of our protocol ($n-1$ rounds) essential?

Acknowledgments

We thank Joe Kilian, Dalit Naor and Omer Reingold for their generous advice. We thank the anonymous referees and the editor, Oded Goldreich, for their diligent reading of the paper, their many useful comments and suggestions and their patience.

References

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

[2] M. Bellare, R. Impagliazzo and M. Naor, *Does Parallel Repetition Lower the Error in Computationally Convincing Protocols?*, to appear, Proc. 38th Symp. on Foundations of Computer Science, 1997.

[3] M. Bellare, S. Micali and R. Ostrovsky, *The (True) Complexity of Statistical Zero Knowledge* Proc. 21st ACM Symp. on Theory of Computing, 1990, pp. 494–502.

[4] J. Boyar, S. Kurtz, and M. Krental, *A Discrete Logarithm Implementation of Perfect Zero-Knowledge Blobs*, Journal of Cryptology, vol 2, 1990, pp. 63-76.

[5] S. Brands, *An efficient off-line electronic cash system based on the representation problem*, CWI Technical Report, CS-R9323, 1993.

[6] G. Brassard, D. Chaum and C. Crépeau, *Minimum Disclosure Proofs of Knowledge*, J. Computer and System Sciences, vol 37, 1988, pp 156–189.

[7] G. Brassard and M. Yung, *One-Way Group Action*, Advances in Cryptology - Crypto '90, Lecture Notes in Computer Science 537, Springer Verlag, 1991, pp. 94–107.

[8] D. Chaum, E. van Heijst and B. Pfitzmann, *Cryptographically strong undeniable signatures, unconditionally secure for the signer*, Advances in Cryptology - Crypto '91, Lecture Notes in Computer Science 576, Springer Verlag, 1992, pp. 470–484.

[9] I. Damgård, *Collision Free Hash Functions and Public Key Signatures Schemes*, Advances in Cryptology - Eurocrypt '87, Lecture Notes in Computer Science 293, Springer-Verlag, 1988, pp. 203–216.

[10] I. Damgård, *Interactive Hashing can Simplify Zero-Knowledge Protocol Design without Complexity Assumptions*, Advances in Cryptology - Crypto '93, Lecture Notes in Computer Science 773, Springer-Verlag, 1994, pp. 100–109.

[11] I. Damgård, T. Pedersen and B. Pfitzmann, *On the Existence of Statistically Hiding Bit Commitment and Fail Stop Signature*, Advances in Cryptology - Crypto '93, Lecture Notes in Computer Science 773, Springer-Verlag, 1994, pp. 250–265.

[12] O. Goldreich, *A Uniform Complexity Treatment of Encryption and Zero-knowledge*, J. of Cryptology, vol 6, 1993, pp. 21–53.

[13] O. Goldreich, **Foundations of Cryptography (Fragments of a Book)**, 1995. Electronic publication: http://www.eccc.uni-trier.de/eccc/info/ECCC-Books/eccc-books.html (Electronic Colloquium on Computational Complexity).

[14] O. Goldreich S. Goldwasser and N. Linial, *Fault Tolerant Computation in the Full Information Model*, to appear in Siam J. on Computing. Extended abstract: Proc. IEEE 32nd Symp. on Foundations of Computer Science, 1991, pp. 447–457.

[15] O. Goldreich, R. Impagliazzo, L. Levin, R. Venkatesan, and D. Zuckerman, *Security Preserving Amplification of Hardness*, Proc. IEEE 31st Symp. on Foundations of Computer Science, 1990, pp. 318–326.

[16] O. Goldreich, H. Krawczyk, and M. Luby, *On the Existence of Pseudo-Random Generators* , SIAM J. Computing, vol 22, 1993, pp. 1163–1175.

[17] O. Goldreich and L. Levin, *A Hard-Core Predicate for any One-way Function*, Proc. 21st ACM Symp. on Theory of Computing, 1989, pp. 25–32.

[18] O. Goldreich, S. Micali and A. Wigderson, *Proofs that Yields Nothing But their Validity or all languages in NP have zero knowledge proof systems*, J. of the ACM, vol 38, 1991, pp. 691-729.

[19] S. Goldwasser, S. Micali and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, SIAM J. Computing, vol 18, 1989, pp. 186-208.

[20] J. Håstad, *Pseudo-Random Generators under Uniform Assumptions*, Proc. 21st ACM Symp. on Theory of Computing, 1990, pp. 395–404.

[21] I. Impagliazzo, L. Levin and M. Luby, *Pseudo-random generation from one-way functions*, Proc. 21st ACM Symp. on Theory of Computing, 1989, pp. 12-24.

[22] R. Impagliazzo and M. Luby, *One-way Functions are Essential for Complexity-Based Cryptography*, Proc. IEEE 30th Symp. on Foundations of Computer Science, 1989, pp. 230–235.

[23] R. Impagliazzo and M. Naor, *Efficient Cryptographic Schemes Provably as Secure as Subset-Sum*, J. of Cryptology, vol 9, 1996, pp. 199–216.

[24] R. Impagliazzo and M. Yung, *Direct Minimum-Knowledge Computations*, Advances in Cryptology - Crypto '87, Lecture Notes in Computer Science 293, Springer-Verlag, 1988, pp. 40–51.

[25] J. Kilian, *A note on efficient zero-knowledge proofs and arguments* Proc. 23rd ACM Symp. on Theory of Computing, 1992, pp. 723-732.

[26] L.A. Levin, One-way functions and pseudorandom generators. *Combinatorica*, Vol. 7, No. 4, 1987, pp. 357–363.

[27] L.A. Levin, *Average case complete problems*, SIAM Journal of Computing **15** (1986), pp. 285–286.

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

[29] S. Micali, *CS Proofs*, Proc. IEEE 35th Symp. of Foundations of Computer Science, 1995, pp. 436–453.

[30] M. Naor, *Bit Commitment Using Pseudo-Randomness*, Journal of Cryptology, vol 4, 1991, pp. 151–158.

[31] M. Naor and M. Yung, *Universal One-Way Hash Functions and their Cryptographic Applications*, Proc. 20th ACM Symp. on Theory of Computing, 1989, pp. 33–43.

[32] R. Ostrovsky, R. Venkatesan, M. Yung, *Fair Games Against an All-Powerful Adversary,* Sequences 91. (Also presented in Dimacs workshop on complexity and cryptography).

[33] R. Ostrovsky, R. Venkatesan, M. Yung, *Secure Commitment Against A Powerful Adversary,* STACS 92, Springer Verlag, Lecture Notes in Computer Science, 1992.

[34] R. Ostrovsky, R. Venkatesan, M. Yung, *Interactive Hashing Simplifies Zero-Knowledge Protocol Design*, Advance in Cryptology - Eurocrypt '93, Lecture Notes in Computer Science 765, Springer-Verlag, 1994.

[35] R. Ostrovsky and A. Wigderson, *One-Way Functions are Essential For Non-Trivial Zero-Knowledge Proofs*, Proc. 2nd Israeli Symp. on Theory of Computing and Systems, 1993, pp. 3–17.

[36] J. Rompel *One-way Functions are Necessary and Sufficient for Secure Signatures*, Proc. 21st ACM Symp. on Theory of Computing, 1990, pp. 387–394.

[37] A. C. Yao, *Theory and Applications of Trapdoor functions*, Proc. of the 23th Symp. on the Foundation of Computer Science, 1982, pp 80–91.

A Relation to recent work on bit-commitment

Bit-commitment (BC) protocols allow a Sender (Commiter) to be bound to a bit which is kept secret from the Receiver. Later, the Sender can "open" that bit in a unique way (i.e. like a sealed envelope). Recently, several models in which some parties are *required* to have computational power beyond polynomial-time were investigated. It is worth while pointing out the differences between those models and the current work.

By "From Strong to Weak BC", we call BC protocols, in which the binding is perfect, i.e. even an infinitely-powerful Sender cannot cheat, except with negligible probability, but where the security is computational, i.e. the Receiver is assumed to be polynomial-time and no such Receiver can figure out the bit committed with non-negligible advantage (if a complexity assumption holds). The combined results of [20, 21, 30] imply that if one-way functions exist, then there is a (Strong-to-Weak) BC which does *not* require the Sender (and of course the receiver) to do non-polynomial work, that is, it is an efficient protocol and the underlying assumption in this case is optimal [22].

The work in [33] investigated commitments between strong and polynomial-time players where the strong player actually needs to use its super-polynomial time power. Thus, the main issue in that paper is how the hardness assumptions change and can be relaxed when there is a large difference in computational power of players (rather than being polynomial-time for both players, as needed in cryptographic applications). It is shown that unless Distributional-NP=RP, a possibly weaker assumption than the existence of one-way functions, there is a (Strong-to-Weak) BC from a Sender with an ($NP \cup co-NP$) power to a polynomial-time Receiver; the Sender actually spends exponential-time in order to execute the protocol. (See [27] for definitions of hard-on-the average problems). Thus, when the Sender uses non-polynomial power this theoretical result relaxes the assumptions in [30].

By "from Weak to Strong BC" we denote BC in which the secrecy is information-theoretic, but the binding is computational, i.e. with high probability a polynomial-time commiter cannot change the value of the commitment (if a complexity assumption holds). In [33] it is also shown that given any one-way function, there is a (Weak-to-Strong) BC from a polynomial-time Sender to a ($PSPACE$) Receiver which actually spends exponential-time in order to execute the protocol. The result is based on an oblivious transfer protocols among unequal-power players from [32],

In contrast, in this paper, the protocols of both parties require only (low order) polynomial-time to execute. This is the appropriate model for cryptographic applications. We made no use of trapdoor properties, as BC's and secure interactive proofs do not need decryptions of arbitrary messages, but rather being able to display the pre-images of pre-specified messages.