Minimum Resource Zero-Knowledge Proofs

(Extended Abstract)

Joe Kilian* Silvio Micali† Rafail Ostrovsky‡

Abstract

We consider several resources relating to zero-knowledge protocols: The number of envelopes used in the protocol, the number of oblivious transfers protocols executed during the protocol, and the total amount of communication required by the protocol.

We show that after a pre-processing stage consisting of $O(k)$ executions of Oblivious Transfer, *any* polynomial number of NP-theorems of any poly-size can be proved non-interactively and in zero-knowledge, based on the existence of *any* one-way function, so that the probability of accepting a false theorem is less then $\frac{1}{2^k}$.

1 Minimizing Envelopes

1.1 Envelopes as a resource.

[GMR] puts forward the somewhat paradoxical notion of a *zero-knowledge proof*, and exemplifies it for a few special classes of assertions. The introduction of ideal commitment mechanisms, known as envelopes, allows us to achieve greater generality. Proofs of any NP statements can be accomplished in perfect zero-knowledge, given the existence of envelopes [GMW].

*MIT, partially supported by NSF 865727-CCR and ARO DALL03-86-K-017.

†MIT, supported in part by NSF grant DCR-84-13577.

‡Boston University, supported in part by NSF grant DCR-86-07492.

An envelope may be thought of as a way to commit to a string s in such a way that the string may be revealed at a later point in time. Before s is revealed (but even after s has been committed), no information about s is revealed. When s is revealed, it is guaranteed to be the same as the value that was committed to.

In practice, one may implement envelopes using physical means (lead boxes, trusted parties, etc.) or by using cryptography. In the first case, it is obvious that minimizing the number of envelopes is a good idea. In the second case, there is an exhorbitant "start up" cost for each envelope. To commit a single bit requires one to write down a number that is on the order of k bits, where k is the security parameter. However, once the start up cost for an envelope has been paid, the amortized cost for each bit that has been comitted is very small.

To make this point more concrete, consider commital schemes based on factoring, in which $k \approx 500$ is reasonable in the light of current (1989) factoring technology. To commit $10,000$ bits individually, one must write down $5,000,000$ bits. However, one can commit $10,000$ bits in aggregate with only $11,000$ bits of communication.

Thus, it is vastly more efficient to implement zero-knowledge proof systems that use only a small number of large envelopes rather than a large number of small envelopes, all other things being equal.

1.2 How many envelopes do we need?

The early zero-knowledge protocols for NP all used an unbounded number of envelopes. More recently, Shamir [S] has found a constant envelope protocol for a natural variant of KNAPSACK. His protocol was of the following very simple form:

1. The prover puts down 3 multi-bit envelopes.
2. The verifier uniformly chooses two of them for the prover to open.

In other words, the protocol can be implemented using a single execution of 2 out of 3 oblivious string transfer (2–3 OT). That is, the prover has three strings, and the verifier is allowed to obtain 2 of them, without the provers knowing which ones were obtained. The verifier is guaranteed to reject with probability at least $\frac{1}{3}$ if the prover attempts to "prove" an incorrect theorem. In fact, the above protocols may be massaged to use only 2 envelopes, given a somewhat more complicated interaction.

While this result is very interesting from a theoretical perspective, it does not always give us the efficient protocols we desire. While one can always reduce ones NP theorem to a knapsack problem, these reductions are not guaranteed to be efficient. A theorem of size n may turn into a KNAPSACK problem of size n^3. The resulting 2-envelope, 2–3 OT protocol may be vastly less efficient than simply using a more direct protocol for the original problem. The advantages given by the simple form of the protocol can be more than offset by the inefficiencies of the problem transformation. More recently, Levin [L] has found a 3-envelope protocol for graph 3-colorability. However, this still does not help us with other NP problems.

1.3 A general transformation.

Ideally, one would like to directly and efficiently transform ones multi-envelope proof system into a three envelope proof sytem. We now show how to do this for a general class of protocols, which we call *subset revealing* protocols. A subset revealing protocol is one of the following form,

1. The prover commits a set of bits, $B = b_1, \ldots, b_n$. We make no restriction on how these bits are grouped into envelopes. Without loss of generality, we may assume that they are committed individually.

2. The verifier makes a query, q.

3. Based on q and B, the prover computes a set $I \subseteq [1, n]$. The prover reveals I along with b_i, for all $i \in I$.

The protocols of [GMW], [B], [BCC], [IY], and [S], among others fall into this class of protocols.

The two key operations in a subset revealing protocol are committing a set of bits and revealing an arbitrary subset of these bits. We now show a simple way to implement these operations using only 2 envelopes. We will in fact lose a little bit of security: The verifier will have a substantial, but not absolute, level of confidence that the revealed subset is equal to the originally committed bits. Still, this loss of security is small compared to the efficiencies obtained by using our technique.

Committing a set of bits.

To commit a set of bits, $B = b_1, \ldots, b_n$, the prover uniformly chooses a set of bits, $R = r_1, \ldots, r_n$. The prover makes one envelope, containing R, and another envelope, containing $B \oplus R$, where $\oplus$ denotes componentwise exclusive-or.

Decommitting an arbitrary subset.

To decommit a subset I of B, the prover reveals I, and for each $i \in I$, reveals r_i and $x_i \oplus r_i$. The verifier then either asks the prover to reveal R, or asks him to reveal $B \oplus R$. The verifier rejects if the revealed values are are not equal to the corresponding stated values. Otherwise, he computes b_i for $i \in I$.

Using this technique, we obtain the following theorem.

Theorem 1 Let (P, V) be a subset revealing protocol for L that achieves perfect zero-knowledge in the ideal envelope model, where $C(|x|)$ bits are committed for an input $x \in L$. Then there exists a 2 envelope protocol, (P', V') for L that achieves perfect zero-knowledge in the ideal envelope model, and commits $O(C(|x|)$ bits. For $x \in L$, (P', V') is as likely to accept as (P, V). For $x \notin L$, the probability that V' will reject is at least one half of the probability that V will reject.

2 Minimizing Oblivious Transfers

We have seen in the previous section that subset revealing zero-knowledge proofs can be efficiently transformed into zero-knowledge interactive proofs that use only two envelopes. The transformation preserves the probability of error and the number of transmitted bits, up to small constant factors. In this section we want to replace each envelope with a constant number of executions of Oblivious Transfer (OT). One obvious advantage of OT over envelopes is that it can be executed interactively at a distance.

A useful class of ZK proofs.

Our transformation will work with a special type of ZK proofs called *simple subset revealing* protocols.

In a single run of such a proof, the prover puts down a given number of envelopes and the verifier selects either to see the contents of all of them, or to have the prover choose and open a subset of them. Typically, each run allows the verifier to catch an error with probability 1/2.

Essentially, the value of q allowed to the verifier is binary. Furthermore, the set I used for $q = 0$ is always required to be the entire set of committed bits, which we denote by B.

ZK proofs of this sort not only exist, but are not "restrictive." The first such a proof was infact deviced by [B] for the NP-complete problem of Graph Hamiltonicity. Later, [BCC] and [IY] developed efficient simple subset revealing protocols for the circuit satisfiability problem.

OT, 1–2 OT, and 2–3 OT.

The notion of an OT was introduced in [R], where it was implemented, based on the difficulty of factoring integers, so to handle moderatly cheating parties. This protocol was strenghened by [FiMiRa] to handle totally cheating parties.

In this paper, we consider oblivious transfers on strings, rather than on single bits. The equivalance between the bitwise and the stringwise versions of various forms of oblivious transfer are established in [BCR] and [C].

Here we use a more powerful notion of an OT, known as 1–2 OT, due to [EGL]. 1–2 OT is a protocol for two players, called, respectively S (for Sender) and R (for Receiver). S has two secret strings m_0 and m_1. R secretly decides on a bit i, which indicates which strings (m_0 or m_1) she would like to get. After the execution of the protocol, the following is true: R learns the value of m_i, but obtains no new information about the other string m_{1-i}, no matter how much she cheats; and S obtains no new information about i, no matter how much he cheats.

It is easy to see that with a constant number of executions of OT, one can build a *2-out-of-3 OT* (2-3 OT for short). That is, the prover has three strings, and the verifier is allowed to obtain two of them *of his choice*, so that he will have no idea about the third one and the Sender will have no idea which two strings the verifier received.

2.1 ZK Proofs with 2-3 OT

As a result of Theorem 1, we can execute each run of a subset revealing ZK proof using only 2 envelopes as outlined in subsection 1.3. There, however, the prover needed to hear a message q from the verifier for computing the subset I_q. In a simple subset revealing ZK proof, however, the subset is chosen by the prover.

In the case of a simple subset revealing protocol, the transformed protocol is of the following simple form.

(computation stage)

1. Denote the set of committed bits by $B = b_1, \ldots, b_n$.
2. The prover uniformly chooses $R = r_1, \ldots, r_n$, and sets $S_1 = R$, and $S_2 = B \oplus R$. In other words, the prover follows the standard transformation, and set S_1 and S_2 to be equal to the contents of the two envelopes.
3. Finally, the prover computes I, the subset he would use if $q = 1$, and sets S_3 to be equal to I, along with r_i and $b_i \oplus r_i$ for all $i \in I$.

(interactive stage)

4. The prover and the verifier then run a 2–3 OT protocol on the three strings. The verifier uniformly chooses two of the three envelopes for the prover to reveal. If envelopes 1 and 2 are opened, the verifier reconstructs B, and accepts iff he would have accepted in the original protocol. If envelopes 1 and 3 or 2 and 3 are opened, the verifier rejects if he detects an inconsistency, as above. Otherwise, he reconstructs I and b_i for $i \in I$, and accepts iff he would have accepted in the original protocol.

Thus, the transformed protocol is turned into a simple execution of 2–3 OT.

Using this technique, we obtain the following theorem.

Theorem 2 Let (P, V) be a simple subset revealing protocol for L that achieves perfect zero-knowledge in the ideal envelope model, where $C(|x|)$ bits are committed for an input $x \in L$. Then there exists a 2–3 OT protocol, (P', V') for L that achieves perfect zero-knowledge in the ideal oblivious transfer model, and in which the total number of bits transferred is $O(C(|x|))$. If for $x \in L$, (P, V) always accepts, then (P, V) will always accept. For $x \notin L$, V' will reject with probability at least one-third the probability that V will reject.

Note that we use the fact that the when the verifier requests strings i and j, he knows which of the two strings he receives S_i and which string is S_j. Otherwise, the prover can cheat in our scheme (though this problem may be easily remedied).

In fact, for the transformed versions of the protocols of [B], [BCC], and [IY], the probability that the verifier will reject an incorrect theorem can be made to be $\frac{1}{3}$. Thus, an average of ≈ 1.7 rounds of the transformed protocol are needed to give the same transfer of confidence as one round of the original protocol.

3 Bounding Interaction

3.1 Introduction

"Classical" proofs do not hide knowledge, but they have a simple mechanics: the prover sends a single message to the verifier (the proof) who does not need to respond. By contrast, what makes zero-knowledge proofs work is the fact that they are *interactive*. That is, the prover and the verifier exchange messages back and forth during the proving process.

Unfortunately, interaction is a very expensive resource, and not always available. This motivates the study of how to bound the interaction required by an zero-knowledge proof as much as possible. Such savings will immediately translate into increased efficiencies for a large number of protocols (both in cryptography and in fault-tolerant computation) that use zero-knowledge proofs as subroutines.

We now show how to squeeze all of the interactions required for zero-knowledge proofs into a small initial preprocessing stage. This interactive preprocessing stage may occur even before the prover has decided which theorems he is to prove. After the preprocessing stage has been completed, the prover can prove polynomially many theorems of any polynomial size. To do this, we show how to appropriately modify protocols of the form given in Section 2.

3.2 Performing OT's "in advance."

The interaction in the protocols given in Section 2 is squeezed into executions of 2–3 OT. However, to implement 2–3 OT seems inherently interactive, since the verifier must choose which strings he is to receive. However, in these transformed protocols, the verifier chooses its two strings completely at random. This random choice might as well be made during an initial preprocessing stage (which may occur even before the theorems to be proven are known to the prover). However, in our applications, the prover may not know which strings it wishes to transfer until long after the end of the preprocessing stage.

Our first solution uses one-time pads. During the preprocessing stage, the prover chooses three random bit strings, R_1, R_2 and R_3. The prover and the verifier run 2–3 OT on these strings, with the verifier uniformly choosing which two it will receive. Later, when the prover and the verifier wish to perform 2–3 OT on strings S_1, S_2 and S_3, the prover simply sends the verifier $S_1 \oplus R_1, S_2 \oplus R_2$, and $S_3 \oplus R_3$, where $\oplus$ denotes bitwise exclusive-or. If the verifier received R_i during the preprocessing stage, it can trivially compute S_i. If the verifier learned nothing about R_i during the preprocessing stage, then it cannot learn anything about S_i from seeing $S_i \oplus R_i$.

Using this technique, arbitrarily many arbitrarily sized theorems may be proven. Note, however, that the random strings, R_i, must be large enough to accomodate all of the strings that will be transferred. Therefore, with this solution, one needs to know in advance an upper bound on the size and number of the theorems to be proven. The number of bits transferred in the preprocessing stage will grow polynomially with the size of the theorem that we wish to prove. The number of string transfers, however, depends solely on the probability of error. After only $O(k)$ transfers, the verifier will catch the prover with probability $1 - 1/2^k$ if he attempts to prove a false statement. This improves the result of [K], in which the number of oblivious transfers grows polynomially with the size of the theorem as well as with the probability of error.

3.3 Using pseudo-random generators to shorten the preprocessing phase.

A difficulty of the method so far is that the random strings transferred during the preprocessng stage grows with the size and number of theorems to be proved. We now show how to make do with small random strings, assuming the existence of cryptographically secure bit generators (CSBG) [BM,Y]. This assumption is equivalent to the existence of one-way functions [ILL].

Instead of making our one-time pads truly random, we use the output of a CSBG, G. Given an n-bit input, s, $G(s)$ produces an infinite sequence, $g_1^s, g_2^s, \ldots$, that is indistinguishable from a random string by any polynomially bounded judge. That is, for any constants c_1, c_2, c_3, no probabilistic Turing machine M, which runs in time n^{c_1}, can distinguish

$$g_1^s, \ldots, g_{n^{c_2}}^s$$

from a truly random string of the same length with probability greater than,

$$\frac{1}{2} + \frac{1}{n^{c_3}}.$$

After performing 2–3 OT on strings, R_1, R_2, R_3, we simply use them as seeds to our generator, G, thus implicitly defining 3 one-time pads of arbitrary size. To a polynomially bounded verifier, these pads are as secure as truly random one-time pads.

3.4 Independence

It should be noticed that while the probability of proving a single false theorem can be made to be less than $\frac{1}{2^k}$, the probability of successfully proving two false theorems is still 2^{-k}, rather then 2^{-2k}. In fact, if the Prover is extremely lucky in guessing exactly which seeds the verifier has received by initial Oblivious Transfer, he can prove essentially as many false theorems as he wants. In practice, this is not a worry, since for even moderately large k, 2^{-k} is such a low probability that one need not to care about what may happen with that probability.

The non-independence of our random choices my open other concerns in some scenarioes. That is, the prover may, on purpose, "xor" some "garbage" information with some of the pads. If the verifier accepts the proof none-the-less, the prover will know that the verifier did not receive the correposing seed in the preprocessing stage. This may be a worry in some applications, but not in the context of proving theorems. In fact, in our framework, the prover has no incentive to cheat. If he cheats a little he may be undetected, but that will not allow him to prove even a single false theorem. If he cheats enough to do that, he is caught almost certainly and not believed ever since. Presumably, it is in the prover's interest being able to send non-interactive zero-knowledge proofs. After all, he is the one who initiates the proving monologue!

4 Comparisons with Previous Work.

A model analogous to the one given in Section 3 was first presented in [DMP1]. [DMP1] show that after an interactive step of size n (i.e. one in which n bits are exchanged), a single theorem of size $\sqrt[3]{n}$ can be proved. Thus, the length of the interaction in the preprocessing stage constitutes a severe upper bound on the length of the theorem that can be proved. In our framework, we are able to prove polynomially many (in the security parameter) polynomially sized theorems. Note however, that the scheme proposed in [DMP1] is based solely on the existence of one-way functions, whereas we need the additional assumption of oblivious transfer (which is implemented using trapdoor permutations [GMW1]).

The results of Section 3 should also be contrasted with the *common random string* model, first put forth in [BFM] (see also [DMP2]). In the common random string model, the prover and the verifier are assumed to both have access to the same sequence of fair coin tosses (a common random string). Using this common random string, the prover gives zero-knowledge proofs using the same mechanics as in classical proofs. That is, he sends a string to the verifier who can then check its correctness, but cannot use it to extract any additional knowledge other then the validity of the theorem.

How "non-interactive" is this model? While the proving stage is non-interactive, it is hard to conceive that prover and verifier have agreed to a common random string without any prior interaction. Thus, one can depict the common random string model as a zero-knowledge proof system in which interaction takes place only in the beginning. This small amount of interaction is only devoted to generate the common random string.

The weakest known assumption for implementing the common random string model is that deciding quadratic residuosity modulo Blum integers (whose factorization is not known) is hard. In contrast, our protocols can be based on any trapdoor permutation.

It should also be pointed out that proofs and protocols in the common random string model were quite complex. For instance, the proof of the "many-theorems" result in the earlier versions of [BFM] and [DMP2] contained a subtke gap: it required, over than the stated number theoretic assumptions, a stronger property about pseudo-random generators. This stronger property is not needed in the final paper [BDFMP]. Another advantage of our result is its simplicity, both in the protocol and its proof.

5 Acknowledgements.

We would like to thank Shafi Goldwasser, Russell Impagliazzo, Leonid Levin, and Ramarathanam Venkatesan for valuable discussions.

6 References

[B1] Blum, M., "Three Applications of the Oblivious Transfer," *University of California, Berkeley*, 1981.

[B] Blum, "How to Prove a Theorem So No One Else Can Claim It", Zero Knowledge Proofs, ICM 1986.

[BeMi] Bellare, M., and S. Micali, "Non-Interactive Oblivious Transfer and Applications," CRYPTO 89.

[BCC] Brassard, Gilles, Claude Crépeau, and David Chaum, "Minimum Disclosure Proofs of Knowledge," manuscript.

[BFM] Blum, Manuel, Paul Feldman, and Silvio Micali, "Noninteractive zero-knowledge proofs and their applications," *Proceedings of the* 20^{th} *STOC*, ACM, 1988.

[BCR] Brassard, Gilles, Claude Crépeau, and Jean-Marc Robert. "Information Theoritic Reductions Among Disclosure Problems," *Proceedings of the* 27^{th} *FOCS*, IEEE, 1986, 168–173.

[BM] Blum, M., and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," *SIAM Journal on Computing*, Vol. 13, No. 4 (November 1984), 850-864.

[DMP] De Santis, A., S. Micali and G. Persiano, "Non-Interactive Zero Knowledge Proof Systems," *CRYPTO-87*

[DMP1] De Santis, A., S. Micali and G. Persiano, "Bounded-Interaction Zero-Knowledge proofs," *CRYPTO-88*.

[EGL] Even, S., O. Goldreich, and A. Lempel, "A Randomized Protocol for Signing Contracts," *CACM* 28 (6) 1985.

[GMW] Goldreich, O., S. Micali, and A. Wigderson, "Proofs that Yield Nothing but their Validity", Technical Report #498, Technion, 1988; preliminary version in *FOCS* 86.

[GMW2] Goldreich, O., S. Micali and A. Wigderson, "A Completeness Theorem for Protocols with Honest Majority," *STOC* 87.

[GMR] Goldwasser, S., S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proofs", *STOC* 85.

[ILL] Impagliazzo, Russell, Leonid Levin, and Mike Luby, "Pseudo-Random Generation from One-Way functions," *STOC-89*.

[IY] Impagliazzo, Russell and Moti Yung, "Direct Minimum Knowledge Computations," Proceedings, *Advances in Cryptology,* Crypto 1987.

[K] Kilian, Joe, "On The Power of Oblivious Transfer," *Proceedings of the* 20^{th} *STOC*, ACM, 1988.

[R] Rabin, M., "How to Exchange Secrets by Oblivious Transfer," Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[SS] Schrift, A. and Adi Shamir, "The Discreet Log is Very Discreet," to appear.

[S] A. Shamir, "A Zero-Knowledge Proof for Knapsacks", presented at a workshop on Probabilistic Algorithms, Marseille (March 1986).

[Y] Yao, A. C., "Theory and Applications of Trapdoor Functions," *Proceedings of the 23rd Annual IEEE Symposium on the Foundations of Computer Science*, IEEE (1982) 80-91.