MIT/LCS/TM-537

GUARANTEED PARTIAL KEY-ESCROW

Silvio Micali

August 1995

Guaranteed Partial Key-Escrow

by

Silvio Micali
Laboratory for Computer Science
MIT

Rough Draft

1 Introduction

At Crypto 92, the author put forward a new approach to key-escrow, whereby (1) trustees have guaranteed pieces of a secret key, (2) too few trustees cannot easily reconstruct the key, but (3) sufficiently many trustees can easily reconstruct the *entire* key. An additional example of key-escrow scheme with similar properties is provided by the Clipper Chip [2].

This guaranteed-piece approach has generated much attention; in particular, see the works of Kilian and Leighton [3], Micali and Sidney [4], Lenstra, Winkler and Yacobi [5], Frankel and Yung [6], which also suggest to enlarge the notion of key escrow so as to incorporate additional properties.

Guaranteed-piece key escrow can be used for a variety of applications, including enabling court-authorized line-tapping in a democratic country, internal auditing of a private corporation, retrieval of a archived data, etc. The new approach is preferable to older ones. In particular, requiring that each secret decryption key to be short, makes retrieving encrypted data not just doable, but doable by any one. Also, Beth's suggestion of escrowing every secret key (in its entirety) with a single trustee does not provide much guarantees about the secrecy of such keys.

Another development on key escrow was publicized at Crypto 95 by Adi Shamir, who advocated a notion of key escrow in which trustees possess pieces of a user's secret key so that together they can reconstruct a major portion of the secret key, but not all of it. We refer to this type of key escrow as *partial key-escrow*.

In this technical report, we wish to *put forward* and *construct* partial key-escrow systems that guarantee an additional crucial property: *ensuring* *that the pieces of the secret key possessed by the trustees are indeed correct.* Indeed, since no one (except, possibly, for the right user) knows the secret key, a trustee cannot judge whether the piece of secret key in his possession, together with the pieces of other trustees, enables one to reconstruct easily a main portion of the secret key of a user. It is clear that if a partial key-escrow system is used in the context of law-enforcement, or in the context or enabling a private organization to monitor, selectively, the communications of its employees, or in the context of retrieving encrypted and archived data, or for most other context, guaranteeing the correctness of each trustee-piece is crucial. We call partial key-escrow systems enjoying this fundamental guaranteed-piece property *Guaranteed Partial Key-Escrow* systems, GPKE systems for short.

GPKE systems ensure that any ciphertext within the system can be decrypted, with the collaboration of the trustees, by means of a moderatly-hard effort. Indeed, if one is guaranteed that the portion of a secret key not in the hands of the trustees consists of a short string, then reconstructing the entire secret key with the collaboration of the trustees requires only a moderatly-hard effort. In fact, one might always rely on exhaustive search for reconstructing the missing portion of the secret key.

On the other hand, without the collaboration of the trustees, reconstructing a secret key in a GPKE system is practically impossible. Indeed, secret keys may in a GPKE system be arbitrarily long. For this reason, GPKE systems are much preferable to systems where secret keys are required to be suitably short. In fact, these latter systems are moderatly hard to crack not only by some designated authority (such as Government, Board of Directors, etc., when they can count on the collaboration of the trustees!) but also by any one else!

Thus GPKE systems are ideally suitable to be used with current export control laws that forbid the export of cryptosystems with too long keys. Indeed, though the secret key of a GPKE system is very long for everyone else, for —say— law enforcement it is relatively short, assuming that the right circustances occur which permit the trustees to collaborate with law enforcement.

GPKE may also lessen the worries that some user may have in escrowing her key. Indeed, she would not surrender her entire secret key, and thus not even an entire collection of malicious trustees can trivially get hold of her key.

If the advantages that GPKE may bring about are clear, it is less clear that GPKE systems can be practically implemented. Let us then show that GPKE systems can be practically implemented in public-key and private-key scenarios, in software and in hardware. For further discussion on the complex social, legal, and technical issues surrounding key-escrow at large we refer to the article of Dorothy Denning [22].

2 GPKE Systems Based on the Diffie-Hellman Scheme

The famous Diffie-Hellman cryptosystem [9] is particularly suitable for law enforcement applications. Indeed, knowledge of the secret key of a user allows one to monitor both the in-coming and the outgoing encrypted traffic of that user.

Typically, in the Diffie-Hellman cryptosystem there are a prime, p, and an element, g, of high order in Z_p^* (preferably g is a generator) that are common to all users in the system. A secret key then consists of an integer x (preferably randomly selected integer in the interval $[1, p-1]$), and the corresponding public key is $g^x \bmod p$. Thus the secret key is the discrete logarithm (in base g) of the public key. Traditionally, p is chosen to be 512-bit long.

To obtain a GPKE system, we propose giving a single trustee almost all the bits of x (e.g., all but 80) so as to guarantee him that the bits in his possession are correct. At a high level, represent the secret key x as the sum of two numbers: a 512-bit number z and a 80-bit number y. The idea then is to give the trustee $g^y \bmod p$, $g^z \bmod p$, z (in a private manner), and a zero-knowledge proof that the secret key relative to g^y is suitably short (roughly, a proof that the discrete log of g^y is suitably short that does not reveal y itself).

The trustee can verify that g^z times g^y equals $g^x \bmod p$. (This check tells him that if he knew the discrete log of g^z and g^y, then the sum of these discrete logs mod $p-1$ would be the secret key relative to g^x.) The trustee can also check that g raised to the power z mod p indeed yields g^z. Thus, if he is proven that y is 80-bit long (without being revealed y itself, as demonstrated below), he would be guaranteed that y could be found by exhaustive search with only a moderatly-hard effort, and thus that the entire secret key x could be found by means of a moderatly-hard effort.

While above we have envisaged a single trustee that receives the key z, the method can be extended to work with a multiplicity of trustees. For instance, there may be n trustees: $T_1, ..., T_n$. In this case, let $z_1, \ldots, z_n$ be n random integers between 1 and $p-1$ whose sum mod $p-1$ equals z, and let each trustee T_i be privately given z_i, while the values g^{z_i} mod p are made known. Then, by having each T_i check that z_i is the correct discrete log of g^{z_i} mod p and that the product of all the g^{z_i} is congruent to g^z mod p, they can verify that, if they reveal to some entity their values z_i, then that entity can reconstruct z very easily, and thus the entire secret key x with only a moderatly-hard additional effort. At the same time, however, no individual trustee, nor any group with less than n trustees can easily reconstruct z (nor x).

If so wanted, one can also ensure that sufficiently many trustees (rather than all of them) can easily reconstruct z, while sufficiently few of them (rather than $n-1$) cannot reconstruct z without a very hard effort. For instance, one can use one of the methods disclosed by Micali [1]. Thus, we can also ensure that sufficiently many trustees can reconstruct the entire secret key x with only a moderatly-hard effort, while sufficiently few of them can reconstruct x only with a very hard effort.

We now must demonstrate that it is possible to prove that the discrete log of g^y is suitably small without revealing it. Proofs of this type are called zero-knowledge proofs. Zero-knowledge proofs have been introduced by Goldwasser, Micali, and Rackoff [12] and are by now well-known in the cryptographic literature. Typically, a zero-knowledge proof of a statement S proceeds by having the Prover present a Verifier two problems, P1 and P2, which are claimed to be both solvable. P1 and P2 are chosen in a special matched way, so that given solutions to both of them one can easily find a proof of S; while given a solution to just one of the two problems does not provide any help in finding the proof of S. After the Prover presents him with P1 and P2, the Verifier chooses at random one of the two problems, and the Prover provides its solution (but not a solution to the other one). This process is repeated several times, choosing the matching P1 and P2 anew each time. Since at each iteration the matching P1 and P2 are randomly chosen (and since there are enormously many such problem pairs) the Verifier will not get both solutions for a matching P1-P2 pair, thus the procedure does not reveal the verifier the proof of S. At the same time the procedure convinces the verifier that the statement S must be true. In fact, if it were false, then either P1 or P2 must be unsolvable (because the existence of a solution for both P1 and P2 implies the existence of a proof of S). Thus, at each iteration the Verifier has probability $\geq 1/2$ of choosing a problem for which the Prover cannot provide a solution. Thus, for instance, if S is false, the probability that the Prover can, ten times in a row, provide a solution to the problem chosen by the Verifier is less than one in a thousand.

Since the statement that the discrete log of g^y is small (i.e., 80-bit long) "is in NP," and since Goldreich, Micali, and Wigderson [18] have shown that there exists a general method for proving in zero-knowledge any NP-property, we could apply their proof-method to the property of interest to us. However, such an approach is hardly practical. Indeed, solving a specific problem by a general tool which disregards the problem's own characteristics rarely yields efficient solutions. We thus put forward below a method that exploits the specific properties of the discrete logarithm problem (without any reduction to other problems). The method is zero-knowledge and is described relative to a single trustee, since we have seen how it can be modified to work with more trustees. Methods that are not exactly zero-knowledge, but are still sufficient for our purposes, can also be derived along similar lines.

A ZERO-KNOWLEDGE METHOD. For completeness, we also describe a process for choosing in Z_p^* an element g of high order. Recall that the order of an element e is k if k is the minimum i such that $e^i \equiv 1 \bmod p$, in which case for any two different integers i and j between 1 and k, $g^i \not\equiv g^j \bmod p$. Since the number of elements in Z_p^* is $p-1$, the order of an element may range between 1 and $p-1$. In any case, however, the order of an element evenly divides $p-1$. Since Z_p^* is cyclic, we know that there exist elements g (called generators) whose order is $p-1$. (The number of generators is $\phi(p-1)$, and $\phi(k) > k/(6 \log\log k)$ for all $k > 3$ as proven by Rosser and Schoenfield [19].) Thus, if g is a generator, any number X between 1 and $p-1$ can be expressed as a power of g mod p. This power is unique in the interval $[1, p-1]$; that is, for any $X \in [1, p-1]$, there is a unique $x \in [1, p-1]$ such that $X = g^x$ mod p. This unique power x is called the discrete log of X (in base g and mod p).

Computing the discrete log of X on inputs X, p, g, and the factorization of $p-1$, is widely believed to be computationally intractable provided that

x is sufficiently large and sufficiently random. (The computational difficulty of the so called *Discrete Logarithm Problem* underlies the security of many a cryptographic scheme, including the Diffie-Hellman one.) The higher the order of the base, the harder is to find discrete logs. Indeed, if g is a generator, the discrete log in base g of an element must be found in the full set $[1, p-1]$ (which includes all 511-bit integers); but if an element e has order $k < p-1$ (e.g., if k is a 80-bit integer), then the discrete log of an element (in base e), which is a power of e, must be found in the smaller set $[1, k]$ (e.g., among the 80-bit numbers). Thus, the security of the Diffie-Hellman scheme increases with the order of g and is considered maximum when g is a generator. For this reason, we believe that, in the Diffie-Hellman cryptosystem, g should be a generator. But, whether a generator or an element of high order, we also believe that g should be *proven to have high order*. Else, a user may suspect that g may be of low order, so that, *de facto*, the system is one in which every secret key is required to be short. According to the above two beliefs, let us describe our Diffie-Hellman based GPKE when g is proven to be a generator, though our method also works for other choices of the base.

Assume that we wish to construct a GPKE based on the Diffie-Hellman where the portion of a secret key not escrowed with the trustee consists of 80-bits. Then, we choose p, g and a such that g is a generator mod p and the order of the element $A = g^a \bmod p$ is 80-bit long. (For simplicity, we also consider the value of a to be common to all users in the system. It should be noted, however, that we may also let a be different for each user.)

To this end, we suggest choosing p so that $p - 1 = 2qQ$, where q is a 80-bit prime (and, preferably, Q has a large prime factor, which makes finding discrete logs mod p harder). Primes with similar structure are constructed within the Digital Signature Standard [11]. In practice, one may even first select a 80-bit prime q and then randomly select large (e.g., 431-bit) primes Q until $p = 2qQ + 1$ is prime. Then, one may select g at random between 1 and $p-1$ until a generator is found, and finally set $a = 2Q$ —thus $A = g^{2Q}$ mod p.

Given the abundance of generators in Z_p^*, one needs to try relatively few elements. Moreover, it is easy to realize that the selected element g is a generator. Indeed, we must show that $g^i \not\equiv 1 \bmod p$ for any i dividing $p-1$. While at first glance it appears that too many values of i must be tried, it is not hard to see that, in our case (i.e., when $p-1 = 2qQ$, where both q and Q are primes), it suffices to compute g^{qQ}, g^{2Q}, and g^{2q} mod p, and verify that none of these values equals 1. If g is a generator, then $A = g^{2Q}$ has order q. Indeed, if A^i equaled 1 mod p for some $i < q$, then $g^I = 1$ mod p, where $I = 2Qi < p - 1$, contradicting the assumption that g is a generator.

Given our p, g, and a (and thus A), we suggest to choose a secret key for the Diffie-Hellman system to be $x = ar + R$ mod p, where r is randomly chosen between 1 and q (and thus is a 80-bit integer) and R is randomly chosen between 1 and $p - 1$ (and thus is a 512-bit integer). Thus, as usual, x is randomly chosen between 1 and $p - 1$. The public key corresponding to x is, as usual, g^x mod p. The trustee is then given $B = g^{ar}$ mod p and $C = g^R$ mod p, as well as (privatly!) the value R. Thus, he can verify that R is the discrete log of C mod p and that the product of B and C is concruent to g^x mod p. Thus, he knows that, if he could compute the discrete log of B mod p, he could easily compute the discrete log of g^x (i.e., the entire secret key) by adding mod $p - 1$ the value R and the discrete log of B.

However, we still need to have a guarantee that the discrete log of B is computable by a modestly-hard effort. To this effect, it is enough to have a zero-knowledge proof that B belongs to the subgroup generated by $A = g^a$ mod p. In fact, the order of A is guaranteed to be 80-bit long, and thus if B is proved to be a power of A mod p, then this secret power, r, cannot be more than 80-bit long, and thus recoverable only by a moderately hard computation (particularly if a "data-base" approach is used to find r). Once r is found, reconstructing the secret key x by computing ar+R mod $p - 1$ is very easy.

Now, the fact that an element B belongs to the subgroup generated by some elemnet A is already known to be provable in zero-knowledge due to the work of Tompa and Woll [20]. In essence, the Prover presents the Verifier with two random elements in the subgroup, I and J, whose product equals B, the Verifier chooses one of them at random, and the Prover releases the discrete log of that element in base A.

A NON-INTERACTIVE METHOD. Notice that the above zero-knowledge proof requires interaction between Prover and Verifier. In our application, however, interaction may not be practical. Indeed, it is unlikely that, in a large country, a single trustee can interact about every partially-escrowed secret key so as to verify that he has a genuine main piece.

To dispose of interaction, we may thus adopt a technique put forward by Fiat and Shamir [10] in the context of digital signature schemes. For instance, the Prover may choose a sequence S of 100 pairs of randomly selected I-J pairs: $S = I_1J_1 \ldots I_{100}J_{100}$. Then, he applies a given pseudo-random function H to this sequence so as to obtain a 100-bit result: $b_1 \ldots b_{100}$. (Think of H as a one-way function mapping any string to a sufficiently random 100-bit string.) Then, for each i between 1 and 100, he releases one value R_i as follows. If $b_i = 0$ R_i is the discrete log (base A) of I_i; else, R_i the discrete log (base A) of J_i. The sequence S together with the Sequence $R_1 \ldots R_{100}$ is a string, $\sum$, vouching that B is in the subgroup generated by A.

The string $\sum$ can be verified as follows. First, the pseudo-random function H is evaluated on S so as to obtain the 100 bits $b_1 \ldots b_{100}$. Then, for each i between 1 and 100, A^{R_i} mod p is computed and it is checked whether the resulting value is I_i (if $b_i = 0$) or J_i (if $b_i = 1$).

Though the bits b_i are not chosen randomly by the Verifier (indeed they are determined by the sequence S via the function H), they are chosen in a way that is random enough for our purposes. Indeed, if B were not in the subgroup generated by A, then, for each i, either I_i or J_i is not in that group, and thus its discrete log in base A does not exist. Thus, in order to construct a string proving that B is in the subgroup, a cheating Prover should be able to construct a special sequence $S' = I'_1J'_1 \ldots I'_{100}J'_{100}$ that, under H, yields 100 bits b'_i such that each b'_i selects exactly the only value between I_i and J_i which has a discrete log base A.

Notice that if the bits b'_i were randomly chosen, then the probability that they would "select" exactly the only 100 elements of S' having a discrete log base A is a remotely small; namely, one in 2^{100}. Thus, even if the bits b'_i obtained via H are not truly random, lots and lots of sequences S' should be tried before one is found such that the bits $H(S')$ select the "right" 100 elements.

So far we have argued that the sequence S together with the sequence of the R_i provide convincing evidence that B is in the subgroup generated by A. It should also be noted that this piece of evidence does not betray the discrete log (base A) of B. It is by now widely believed that if the original interactive proof was zero-knowledge, then the so obtained non-interactive proofs (i.e., strings) do not reveal in practice any significant additional information either. (Indeeed, this way of replacing interaction forms the base of many a cryptographic schemes and has been formally advocated by Bellare and Rogaway [13].) Indeed, extracting the discrete log of B from S and the R_i appears to be a formidable task.

Another way to prove non-interactively that B belongs to the subgroup generated by A without revealing its discrete log is using the non-interactive zero=knowledge proofs à la Blum, Micali and Feldman [14] [15].

In sum, it is also possible to prove that a given partial key-escrow system is a GPKE systems without any interaction, that is, by just providing a special string for each secret key. Of course, if so want, one may use special strings just to reduce the amount of interaction.

3 GPKE Systems Based on on the RSA

- NOTE: A GPKE System based on the RSA has been independently found by Shafi Goldwasser [21].

Let us now quickly describe a software GPKE system based on the RSA cryptosystem In it, a public key consists of a composite number n, the corresponding secret key of the prime factorization of n, and the encryption of a message $m \in Z_n^*$ (i.e., between 1 and n and relatively prime with n) of the value $m^e \bmod n$, where e is an exponent relatively prime with $\phi(n)$ (where ϕ represents Euler's totient function). Traditionally, n is chosen to be the product of two large primes.

We instead propose to chose n to be the product of 4 primes, p_1, p_2, p_3 and p_4, where p_1 and p_2 are large (say 512-bit each) and p_3 and p_4 are small (e.g., 150-bit each). Then p_1 and p_2 could be revealed to a trustee, while p_3 and p_4 will remain secret. It has been shown by Chor, Goldwasser, Micali, and Awerbuch [8] that, as long as two primes of an RSA public key remain secret, then the security of the system is essentially at least equal to that of an RSA system in which the public key consists of only those two secret primes. Thus, the trustee still faces an RSA cryptosystem whose public key is the product of p_3 and p_4. This number is however much more easily factored than the original n! On the other hand, everyone else faces at least an RSA cryptosystem with the original n, and thus a system whose security is at least that of a traditional RSA system with a 1024-bit public key.

Notice that the above described key-escrow system is a GPKE one because the trustee can easily verify to have almost all (but not all) the secret key corresponding to n. Indeed, it can verify that n is 1324-bit long, that p_1 and p_2 are different primes, that each of them is 512-bit long, and that each of them divides n. In sum we have a system that is moderately hard to crack (but not already cracked!) for the trustee, while it is very hard to crack for everyone else.

Above, we have envisaged using four primes and having a single trustee. Indeed, because in a GPKE system the trustees do not collectively possess the entire secret key, having a single trustee may be quite acceptable. In any case, it should be realized that the above system can be generalized so as to work with an arbitrary number of primes and with many trustees, each possessing (if so wanted) a different guaranteed piece of the secret key (while all together they still do not possess the entire secret key). For instance, there may be six distinct primes dividing n, $p_1, \ldots, p_6$, where $p_1, \ldots, p_4$ are 512-bit long and p_5 and p_6 150-bit long, and there may be four trustees, $T_1, \ldots, T_4$, where each T_i knows p_i. In this case at least 3 trustees must collaborate in order to make the reconstruction of n's entire factorization only moderatly hard. To ensure that each trustee possesses a different prime, proper digits of these primes may be made known (which will not compromise the secrecy of n's factorization, because in general it would be enogh to reveal the content of two bit-locations in our primes). Needless to say, each T_i can easily check that the value of those particular digits in his own prime are correct.

Of course, if so wanted, it can also be arranged for redundancy, so that the pieces of secret keys held by the trustees are not totally disjoint. For instance, one may insist that the same piece is given to more than one trustee.

It may also be arranged that the number of trustees exceed that of the primes in n's factorization. For instance, if n is the product of four primes, where p_1 and p_2 are long and p_3 and p_4 short as in our first example, the one could make known the products $n' = (p_1 \cdot p_2)$, and $n'' = (p_3 \cdot p_4)$ and then share the factorization of n' with arbitrarily many trustees as shown by Micali [1].

Needless to say, above, the values 512 and 150 are mere suggestions, and the right balance between the effort of factoring n without the help of the trustees and that of factoring n with the trustees' help may be obtained by choosing the values that seem most appropriate. (In particular, each prime may be chosen of a different length.)

4 Hardware-Based GPKE Sytems

Let us now describe some hardware methods to construct guaranteed partial key-escrow systems. In the simplest embodiment, assume that secure chips are manufactured that (like the Clipper chip) have a special decryption key inside. Assume that 100-bit is deemed to be a secure length for these keys. Then, a special agent can provide the chip with a 50-bit key, and the chip can choose the other 50 bits in a random number (e.g.by a noise diod); or the user chooses the other 50. Then an enemy faces a chip with a secret 100-bit key while the special agent (and thus law enforcement) a chip with just a 50-bit key.

This goal can also be accomplished by secure hardware when it is up to the user of the chip to choose the entire 100-bit key of the chip. The chip may in fact reveal the first 50 bits of this key to a special agent. For instance, it sends them along together with each ciphertext, encrypted with an internal key that only the special agent knows. Alternatively, before the chip starts functioning it is necessary that it receives a special signal from the special agent, and the special agent will give this activation signal only after the chip sends him an encrypted version of the first 50 bits of its chosen secret key. Of course 100 and 50 are variables, it is not necessary that the bits in possession of the special agent are the first 50, and other precautions can be taken to make sure that the chip works as desired.

References

[1] S. Micali. *Fair Cryptosystems.* Proc. Crypto 92.

[2] National Institute of Standards and Technology. *Escrowed Encryption Standard.* Federal Information Processing Standards, PUB 185, 9 February 1994.

[3] J. Kilian and T. Leighton. *Fair Cryptosystems, revisited.* Advances in Cryptology, Proc. CRYPTO 95, Lectures Notes in Computer Science, Springer, 1995, pp. 208-221.

[4] S. Micali and R. Sidney. *A simple method for generating and sharing pseudo-random functions, with applications to Clipper-like key escrow*

systems. Advances in Cryptology, Proc. CRYPTO 95, Lectures Notes in Computer Science, Springer, 1995, pp. 185-196.

[5] A. Lenstra, P. Winkler, Y. Yacobi. *A key escrow systems with warrant bounds.* Advances in Cryptology, Proc. CRYPTO 95, Lectures Notes in Computer Science, Springer, 1995, pp. 197-207.

[6] Y. Frankel and M. Yung. *Escrow encryption systems visited.* Advances in Cryptology, Proc. CRYPTO 95, Lectures Notes in Computer Science, Springer, 1995, pp. 208-235.

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

[8] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. Proc. 26 ann. IEEE Symp. on Foundations of Computer Science, IEE, New York, 1986, pp. 383-395.

[9] W. Diffie and M. Hellman. *New directions in cryptography.* IEEE Trans. Inform. Theory, IT-22, Vol. 6 (1976), IEEE, New York, pp. 644-654.

[10] A. Fiat and A. Shamir. *How to Prove Yourselves: Practical Solutions of Identification and Signature Problems.* Proc. Crypto 86, Springer Verlag, 263, 1987, pp.186-194.

[11] National Institute of Standard and Technology. *Digital Signature Standard (DSS).* Federal Information Processing Standards PUB 186, May 19, 1994.

[12] S. Goldwasser and S. Micali and C. Rackoff. *The Knowledge Complexity of Interactive Proof Systems.* SIAM J. Comput., 18, 1989, pp. 186-208. An earlier version of this result informally introducing the notion of a proof of knowledge appeared in Proc. 17th Annual Symposium on Theory of Computing, 1985, pp. 291-304. (Earlier yet versions include "Knowledge Complexity," submitted to the 25th Annual Symposium on the Foundations of Computer Science, 1984.)

[13] M. Bellare and P. Rogaway. *Random Oracles are practical: a paradigm for designing efficient protocols.* Proc. 1st ACM Conf. on Computer and Communication Security, ACM, November 1993.

[14] M. Blum, P. Feldman, and S. Micali. *Non-Interactive Zero-Knowledge Proof Systems and Applications.* STOC 1988.

[15] M. Blum, A. De Santis, S. Micali, and G. Persiano. *Non-Interactive Zero-Knowledge.* SIAM J. on Comp. 1991.

[16] M. Blum and S. Micali. *How to Generate Cryptographically-Strong Sequences of Pseudo-Random Bits.* SIAM J. on Comp. vol 13, 1984

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

[18] O. Goldreich, S. Micali, and A. Wigderson. *Proofs that Yield Nothing But Their Validity or All Languages in $\mathcal{NP}$ have Zero-Knowledge Proof Systems.* J. of the ACM, Vol 38, No. 1, July 1991, pp. 691-729.

[19] J. Rosser and L. Schoenfield. *Approximate formulas for some functions of prime numbers.* Illinois J. Math., Vol. 6 (1962), pp. 64-94.

[20] M. Tompa and H. Woll. *Random Self-Reducibility and Zero-knowledge Interactive Proofs of Possession of Information.* Proc. 28th Conference on Foundations of Computer Science, 1987, pp. 472-482.

[21] S. Goldwasser. *Private Communication.* Sep 7, 1995.

[22] D. Denning. *To tap or not to tap.* Comm. of the ACM, March 1993, Vol. 3, pp. 25-44.