A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack

Ronald Cramer
Institute for Theoretical Computer Science, ETH Zurich, 8092 Zurich, Switzerland
`cramer@inf.ethz.ch`

Victor Shoup
IBM Zurich Research Laboratory, Säumerstr. 4, 8803 Rüschlikon, Switzerland
`sho@zurich.ibm.com`

Feb. 16, 1998

Abstract

A new public key cryptosystem is presented that is provably secure against adaptive chosen ciphertext attack. The scheme is quite practical, and the proof of security relies only on standard intractability assumptions.

1 Introduction

In this paper, we present and analyze a new public key cryptosystem that is provably secure against adaptive chosen ciphertext attack (as defined by Rackoff and Simon [13]). The scheme is quite practical, requiring just a few exponentiations over a group, and the application of a hash function. Moreover, the proof of security relies only on standard intractability assumptions, namely, the hardness of the Diffie-Hellman decision problem in the underlying group, and the collision intractability of the hash function.

The hardness of the Diffie-Hellman decision problem is essentially equivalent to the semantic security of the basic El Gamal encryption scheme [6]. Thus, with the additional assumption of a collision-resistant hash function, and just a bit more computation, we get security against adaptive chosen ciphertext attack, whereas the basic El Gamal scheme is completely insecure against adaptive chosen ciphertext attack.

While there are several provably secure encryption schemes in the literature, they are all quite impractical. Also, there have been several practical cryptosystems that have been proposed, but none of them have been proven secure under standard intractability assumptions. The significance of our contribution is that it provides a scheme that is provably secure and practical at the same time. There appears to be no other encryption scheme in the literature that enjoys both of these properties simultaneously.

Chosen Ciphertext Security

The notion of *semantic security* (defined by Goldwasser and Micali [8]) captures the notion of security of a public key cryptosystem against chosen plaintext attack. It is now generally accepted that this is a basic requirement of a good cryptosystem. However, it also known that other, stronger attacks are possible, and moreover, security against these types of attacks are necessary to ensure the security of many higher-level protocols built on top of the cryptosystem.

A *chosen ciphertext attack* is one in which the adversary has access to a "decryption oracle," allowing the adversary to decrypt ciphertexts of his choice. Typically, one distinguishes between a weak form of this attack, known as a *lunch-time attack* (defined by Naor and Yung [12]), and the strongest possible form, known as an *adaptive chosen ciphertext attack* (defined by Rackoff and Simon [13]). In a lunch-time attack, the adversary queries the decryption oracle some number of times, after which, he obtains the target ciphertext that he wishes to cryptanalyze, and is not allowed to query the decryption oracle further. In an adaptive attack, the adversary may continue to query the decryption oracle after obtaining the target ciphertext, subject only to the (obviously necessary) restriction that queries to the oracle may not be identical to the target ciphertext.

Security against adaptive chosen ciphertext attack also implies *non-malleability* (defined by Dolev, Dwork and Naor [5]), meaning that an adversary cannot take an encryption of some plaintext and “massage” it into an encryption of a different plaintext that is related in some interesting way to the original plaintext.

Provably Secure Schemes. For many years, no public key system was shown to be secure under a chosen ciphertext attack. Naor and Yung [12] presented the first scheme provably secure against lunch-time attacks. Subsequently, Dolev, Dwork, and Naor [5] presented a scheme that is provably secure against adaptive chosen ciphertext attack.

Unfortunately, all of the known schemes provably secure under standard intractability assumptions are completely impractical (albeit polynomial time), as they rely on general and expensive constructions for non-interactive zero-knowledge proofs.

Practical Schemes. Damgard [4] proposed a practical scheme that he conjectured to be secure against lunch-time attacks; however, this scheme is not known to be provably secure, and is in fact demonstrably insecure against adaptive chosen ciphertext attack. Zheng and Seberry [16] propose practical schemes that are conjectured to be secure against chosen ciphertext attack, but again, no proof based on standard intractability assumptions is known. Lim and Lee [10] also proposed practical schemes that were later broken by Frankel and Yung [7].

In a different direction, Bellare and Rogaway [1, 2] have presented practical schemes that are provably secure against adaptive chosen ciphertext attack in an idealized model of computation where a hash function is represented by a random oracle.

While a proof of security in the random oracle model is certainly preferable to no proof at all, a proof in the “real world” would be even better.

Indeed, recent work by Canetti, Goldreich, and Halevi [3] show that there are cryptographic schemes that are secure in the random oracle model, but insecure in the real world—no matter what hash function is chosen. It is not yet clear what the implications of these results are. While it still seems that security in the random oracle model does give good heuristic evidence that a natural scheme is secure in the real world, these results certainly cast a bit of a cloud on the random oracle model, providing extra motivation to seek out practical schemes that are provably secure under standard intractability assumptions.

2 The Basic Scheme

We assume that we have a group G of prime order q, where q is large. We also assume that cleartext messages are (or can be encoded as) elements of G (although this condition can be relaxed, as will be discussed later). We also need a hash function H that hashes long strings to elements of $\mathbf{Z}_q$.

Key Generation. The key generation algorithm runs as follows. Random elements $g_1, g_2 \in G$ are chosen, and random elements

$$x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbf{Z}_q$$

are also chosen. Next, the group elements

$$c = g_1^{x_1} g_2^{x_2},\ d = g_1^{y_1} g_2^{y_2},\ h = g_1^{z_1} g_2^{z_2}$$

are computed. The public key is (g_1, g_2, c, d, h), and the private key is $(x_1, x_2, y_1, y_2, z_1, z_2)$.

Encryption. Given a message $m \in G$, the encryption algorithm runs as follows. First, it chooses $r \in \mathbf{Z}_q$ at random. Then it computes

$$u_1 = g_1^r,\ u_2 = g_2^r,\ e = h^r m,\ \alpha = H(u_1, u_2, e),\ v = c^r d^{r\alpha}.$$

The ciphertext is

$$(u_1, u_2, e, v).$$

Decryption. Given a ciphertext (u_1, u_2, e, v), the decryption algorithm runs as follows. It first computes $\alpha = H(u_1, u_2, e)$, and tests if

$$u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^\alpha = v.$$

If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs

$$m = e/(u_1^{z_1} u_2^{z_2}).$$

We should verify that the decryption of an encryption of a message yields the message. Since $u_1 = g_1^r$ and $u_2 = g_2^r$, we have

$$u_1^{x_1} u_2^{x_2} = g_1^{r x_1} g_2^{r x_2} = c^r.$$

Likewise, $u_1^{y_1} u_2^{y_2} = d^r$ and $u_1^{z_1} u_2^{z_2} = h^r$ Therefore, the test performed by the decryption algorithm will pass, and the output will be $e/h^r = m$.

3 Proof of Security

In this section, we prove the following theorem.

Theorem 1 *The above cryptosystem is secure against adaptive chosen ciphertext attack assuming that (1) the hash function H is collision resistant, and (2) the Diffie-Hellman decision problem is hard in the group G.*

Before going into the proof, we recall the meaning of the technical terms in the above theorem.

Security against adaptive chosen ciphertext attack. Security is defined via the following game played by the adversary.

First, the key generation algorithm is run, with a security parameter as input. Next, the adversary makes arbitrary queries to a "decryption oracle," decrypting ciphertexts of his choice.

Next the adversary chooses two messages, m_0, m_1, and sends these to an "encryption oracle." The encryption oracle chooses a bit $b \in \{0, 1\}$ at random, and encrypts m_b. The corresponding ciphertext is given to the adversary (the internal coin tosses of the encryption oracle, in particular b, are not in the adversary's view).

After receiving the ciphertext from the encryption oracle, the adversary continues to query the decryption oracle, subject only to the restriction that the query must be different than the output of the encryption oracle.

At the end of the game, the adversary outputs $b' \in \{0, 1\}$, which is supposed to be the adversary's guess of the value b. If the probability that $b' = b$ is $1/2 + \epsilon$, then the adversary's *advantage* is defined to be ϵ.

The cryptosystem is said to be secure against adaptive chosen ciphertext attack if the advantage of any polynomial-time adversary is negligible.

Collision resistant hash functions. A family of hash functions is collision resistant if given a random hash function H in the family, it is infeasible to find a collision, i.e., two strings $x \neq y$ such that $H(x) = H(y)$.

The Diffie-Hellman Decision Problem. Let G be a group of prime order q, and consider the following two distributions:

- the distribution $\mathbf{R}$ of quadruples (g_1, g_2, u_1, u_2), where g_1, g_2, u_1, u_2 are chosen at random.

- the distribution $\mathbf{D}$ of quadruples (g_1, g_2, g_1^r, g_2^r), where $g_1, g_2 \in G$ are chosen at random, and $r \in \mathbf{Z}_q$ is chosen at random.

An algorithm that solves the Diffie-Hellman decision problem is a statistical test that can distinguish the two distributions. That is, given a quadruple coming from one of the two distributions, it should output 0 or 1, and there should be a non-negligible difference between (a) the probability that it outputs a 1 given an input from $\mathbf{R}$, and (b) the probability that it outputs a 1 given an input from $\mathbf{D}$. The Diffie-Hellman decision problem is hard if there is no such polynomial-time statistical test.

Related to the Diffie-Hellman decision problem is the Diffie-Hellman problem (given g, g^x and g^y, compute g^{xy}), and the discrete logarithm problem (given g and g^x, compute x).

There are obvious polynomial-time reductions from the Diffie-Hellman decision problem to the Diffie-Hellman problem, and from the Diffie-Hellman problem to the discrete logarithm problem, but reductions in the reverse direction are not known. Moreover, these reductions are essentially the only known methods of solving the Diffie-Hellman or Diffie-Hellman decision problems. All three problems are widely conjectured to be hard, and have been used as assumptions in proving the security of a variety of cryptographic protocols. Some heuristic evidence for the hardness of all of these problems is provided in [14], where it is shown that they are hard in a certain natural, structured model of computation. See [15, 11] for further applications and discussion of the Diffie-Hellman decision problem.

It is perhaps worth pointing out that the hardness of the Diffie-Hellman decision problem is equivalent to the security of the basic El Gamal encryption scheme against chosen message attack. Recall that in the basic El Gamal scheme, we encrypt a message $m \in G$ as $(g^r, h^r m)$, where h is the public key of the recipient whose secret key is s with $h = g^s$. A chosen message attack is equivalent to a chosen ciphertext attack without access to a decryption oracle (i.e., it is a passive attack). On the one hand, if the Diffie-Hellman decision problem is hard, then the group element h^r could be replaced by a random group element without changing significantly the behavior of the attacker; however, if we perform this substitution, the message m is perfectly hidden, which implies security. On the other hand, if the Diffie-Hellman decision problem can be efficiently solved, we can feed the pair $(1, m)$, with $m \in G$ random, to the encryption oracle; then, if (u, v) is the ciphertext, we feed

(g, h, u, v) to a statistical test for the Diffie-Hellman decision problem. This can be used to tell if 1 or m was encrypted, since in the first case, (g, h, u, v) comes from $\mathbf{D}$, and in the second case, it comes from $\mathbf{R}$.

It is also worth pointing out here that the basic El Gamal scheme is completely insecure against adaptive chosen ciphertext attack. Indeed, given an encryption (u, v) of a message m, we can feed the $(u, v{\cdot}g)$ to the decryption oracle, which gives us $m \cdot g$.

Proof of Theorem

To prove the theorem, we will assume that there is an adversary that can break the cryptosystem, and show how to use this adversary to construct a statistical test for the Diffie-Hellman decision problem.

For the statistical test, we are given (g_1, g_2, u_1, u_2) coming from either the distribution $\mathbf{R}$ or $\mathbf{D}$. At a high level, our construction works as follows. We build a simulator that simulates the joint distribution consisting of adversary's view in its attack on the cryptosystem, and the bit b generated by the decryption oracle (which is not a part of the adversary's view). It will be clear from the construction that if the input happens to come from $\mathbf{D}$, the simulation of this joint distribution is perfect, and so the adversary has a non-negligible advantage. We then show that if the input happens to come from $\mathbf{R}$, then the adversary's view is essentially independent of b, and therefore the adversary's advantage is negligible. This immediately implies a statistical test distinguishing $\mathbf{R}$ from $\mathbf{D}$.

We now give the details of the simulator. The input to the simulator is (g_1, g_2, u_1, u_2). The simulator runs the key generation algorithm, using the given g_1, g_2. More specifically, the simulator chooses

$$x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbf{Z}_q$$

at random, and computes

$$c = g_1^{x_1} g_2^{x_2},\ d = g_1^{y_1} g_2^{y_2},\ h = g_1^{z_1} g_2^{z_2}.$$

The public key that the adversary sees is (g_1, g_2, c, d, h). The simulator knows the corresponding private key $(x_1, x_2, y_1, y_2, z_1, z_2)$.

The simulator answers decryption queries as in the actual attack, which it can do since it knows the private key.

We now describe the simulation of the encryption oracle. Given m_0, m_1, the simulator chooses $b \in \{0,1\}$ at random, and computes

$$e = u_1^{z_1} u_2^{z_2} m_b,\ \ \alpha = H(u_1, u_2, e),\ \ v = u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^{\alpha},$$

and outputs

$$(u_1, u_2, e, v).$$

That completes the description of the simulator. As we will see, when the input to the simulator comes from $\mathbf{D}$, the output of the encryption oracle is a perfectly legitimate ciphertext; however, when the input to the simulator comes from $\mathbf{R}$, the output of the decryption oracle will not be legitimate, in the sense that $\log_{g_1} u_1 \neq \log_{g_2} u_2$. This is not a problem, and indeed, it is crucial to the proof of security.

First, consider the joint distribution of the adversary's view and the bit b when the input comes from the distribution $\mathbf{D}$. Say $u_1 = g_1^r$ and $u_2 = g_2^r$. Then it is clear that $u_1^{x_1} u_2^{x_2} = c^r$, $u_1^{y_1} u_2^{y_2} = d^r$, and $u_1^{z_1} u_2^{z_2} = h^r$. From this is is clear that the joint distribution of the adversary's view and b is identical to that in the actual attack.

Second, consider the more interesting case of the distribution of the adversary's view and the bit b when the input comes from $\mathbf{R}$. We want to show that the adversary's view and b are essentially independent. This argument will be purely information theoretic, except that it will rely on the assumption that the adversary cannot find a collision in the hash function H.

First, some notation and terminology. Let $\log(\cdot)$ denote the logarithm to the base g_1, and let $w = \log g_2$. Let $u_1 = g_1^{r_1}$ and $u_2 = g_1^{wr_2}$. We may assume that $r_1 \neq r_2$, since this occurs with overwhelming probability. Also, let us define a tuple $(u_1', u_2', e', v') \in G^4$ to be a "valid ciphertext" if there exists $r' \in \mathbf{Z}_q$ such that $u_1' = g_1^{r'}$ and $u_2' = g_2^{r'}$. Otherwise, we will say it is an "invalid ciphertext."

Claim 1. If the decryption oracle rejects all invalid ciphertexts during the attack, then b is independently distributed from the adversary's view.

To see this, consider the pair $(z_1, z_2) \in \mathbf{Z}_q^2$. At the beginning of the attack, this is a random point on the line $z_1 + wz_2 = \log h$ (this is the information about (z_1, z_2) leaked by the public key). Moreover, if the decryption oracle only decrypts valid ciphertexts (u_1', u_2', e', v'), then the adversary obtains only linearly dependent relations $r'z_1 + r'wz_2 = r' \log h$ (since

$(u_1')^{z_1}(u_2')^{z_2} = g_1^{r'z_1} g_2^{r'z_2} = h^{r'}$). Thus, no further information about (z_1, z_2) is leaked.

Consider now the output of the simulated encryption oracle. We have

$$\begin{pmatrix} \log h \\ \log(e/m_b) \end{pmatrix} = \begin{pmatrix} 1 & w \\ r_1 & wr_2 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}.$$

Since the matrix in the above equation is nonsingular, for each choice of $b \in \{0, 1\}$, there exists exactly one solution (z_1, z_2). This implies that the distribution of b is independent of the adversary's view.

Claim 2. Assuming the adversary does not find a collision in H, then with overwhelming probability, the decryption oracle will reject all invalid ciphertexts during the attack.

To prove this claim, we study the distribution of $(x_1, x_2, y_1, y_2) \in \mathbf{Z}_q^4$ as seen by the adversary. From the adversary's view, this is essentially a random point on the line formed by intersecting the hyperplanes:

$$\begin{aligned} &x_1 + wx_2 = \log c, \\ &y_1 + wy_2 = \log d, \\ &r_1x_1 + wr_2x_2 + (\alpha r_1)y_1 + (\alpha wr_2)x_2 = \log v, \end{aligned}$$

where $\alpha = H(u_1, u_2, e)$. The first two equations come from the public key, and the third comes from the output of the encryption oracle.

Actually, the adversary obtains a bit more information about the point (x_1, x_2, y_1, y_2) when the decryption oracle rejects an invalid ciphertext, puncturing the above line at the point where it intersects a hyperplane. This only makes a negligible difference in the distribution (x_1, x_2, y_1, y_2), which we can safely ignore.

Also note that decrypting a valid ciphertext leaks no information about the point (x_1, x_2, y_1, y_2).

The above considerations imply that it suffices to consider what happens when the adversary presents a single invalid ciphertext $(u_1', u_2', e', v') \neq (u_1, u_2, e, v)$ to the decryption oracle.

First, assume that $(u_1', u_2', e') = (u_1, u_2, e)$. In this case, the hash values are the same, but $v' \neq v$ implies that the decryption oracle will certainly reject.

Second, assume that $(u_1', u_2', e') \neq (u_1, u_2, e)$. Let $\alpha' = H(u_1', u_2', e')$ and $\alpha = H(u_1, u_2, e)$. We are assuming, by collision intractability, that $\alpha' \neq \alpha$.

Let $u_1' = g_1^{r_1'}$ and $u_2' = g_1^{wr_2'}$, where $r_1' \neq r_2'$ (since the ciphertext is invalid). The decryption oracle will not reject if and only if $v' = v''$, where

$$v'' = (u_1')^{x_1}(u_2')^{x_2}((u_1')^{y_1}(u_2')^{y_2})^{\alpha'}.$$

Consider the 4×4 matrix

$$M = \begin{pmatrix} 1 & w & 0 & 0 \\ 0 & 0 & 1 & w \\ r_1 & wr_2 & \alpha r_1 & \alpha w r_2 \\ r_1' & wr_2' & \alpha' r_1' & \alpha' w r_2' \end{pmatrix}.$$

It will suffice to show that M is nonsingular, because then even when the adversary sees the first three entries of the vector

$$\begin{pmatrix} \log c \\ \log d \\ \log v \\ \log v'' \end{pmatrix} = M \begin{pmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \end{pmatrix},$$

the last entry of this vector will be independent of the adversary's view. But then the probability that $\log v' = \log v''$ is negligible, and when equality does not hold, the decryption oracle will reject.

To finish the proof, we only need to show that M is nonsingular. But for this, one can easily verify that

$$\det(M) = w^2(r_2 - r_1)(r_2' - r_1')(\alpha - \alpha') \neq 0.$$

That completes the proof of security.

4 Implementation Details and Variations

In this section, we briefly discuss some implementation details and possible variations of the basic encryption scheme.

(1) To define the group, we could choose a large prime p (say, 1024 bits long), such that $p-1 = 2q$, where q is also prime. Then the group G would be chosen to be the subgroup of index 2 in the group of units of integers modulo p. If we restrict a message to be an element of the set $\{1, \ldots, (p-1)/2\}$, then we can "encode" a message by squaring it modulo p, giving us an element in G. We can recover a message from its encoding by computing the unique square root of its encoding modulo p that is in the set $\{1, \ldots, (p-1)/2\}$.

(2) This yields an implementation that is reasonably efficient. However, it would be more practical to work in a smaller subgroup, and it would be nice to have a more flexible and efficient encoding scheme.

To do this, one could do the following. Choose a 1024-bit prime p such that $p-1 = qm$, where q is a prime with, say, 240-bits. The group G would then be the subgroup of order q in the multiplicative group of units modulo p. Then, instead of encoding a message as a group element, one could just view it as a bit string. The encryption algorithm would have to be modified, replacing $e = h^r m$ with $e = F(h^r) \oplus m$, where F is a function that maps a random element of G (as encoded as an integer modulo p) to a bit string of the same length as m that is computationally indistinguishable from a random bit string of the same length.

One way to implement F is as follows. First, hash the 1024-bit encoding of h^r down to, say, 56 bits using a random but publicly known 2-universal hash function. The left-over hash lemma [9] would imply that these 56 bits are fairly close to random. We can then use these 56 bits as a DES key, and generate as many pseudo-random bits as we need using DES in "counter mode." The security proof would then require the assumption that DES is a good pseudo-random permutation, which is quite reasonable. A more expensive pseudo-random bit generator could be used if a weaker intractability assumption were desired.

(3) Another, somewhat more efficient variant of the scheme runs as follows. The public key and encryption algorithm are the same, but the key generation and decryption algorithms are slightly different. In this variation, the private key consists of $(w, x, y, z) \in \mathbf{Z}_q^4$, and the public key is computed as

$$g_2 = g_1^w, \ c = g_1^x, \ d = g_1^y, \ h = g_1^z.$$

The test made by the decryption algorithm on input (u_1, u_2, e, v) is:

$$u_2 = u_1^w \text{ and } v = u_1^{x+y\alpha},$$

where $\alpha = H(u_1, u_2, e)$. If this test passes, the output of the encryption algorithm is $m = e/u_1^z$.

In the proof of security of this modified encryption scheme, one uses the *same* simulator as in the original proof. When one does this, the key generation and decryption algorithms of the simulator are no longer identical to those in the actual scheme, as they were in the original proof. Nevertheless, using arguments similar to those in the original proof, one can show that when the input to the simulator comes from the distribution $\mathbf{D}$, the simulation is nearly perfect. Thus, in this case, the adversary still has a nonnegligible advantage. However, when the input to the simulator comes from $\mathbf{R}$, the adversary has advantage zero, just as in the original proof.

References

[1] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, 1993.

[2] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology—Crypto '94*, pages 92–111, 1994.

[3] R. Canetti, O. Goldreich, and S. Halevi. The random oracle model, revisted. In *30th Annual ACM Symposium on Theory of Computing*, 1998. To appear.

[4] I. Damgard. Towards practical public key cryptosystems secure against chosen ciphertext attacks. In *Advances in Cryptology–Crypto '91*, pages 445–456, 1991.

[5] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, 1991.

[6] T. El Gamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.

[7] Y. Frankel and M. Yung. Cryptanalysis of immunized LL public key systems. In *Advances in Cryptology–Crypto '95*, pages 287–296, 1995.

[8] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[9] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random number generation from any one-way function. In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, 1989.

[10] C. H. Lim and P. J. Lee. Another method for attaining security against adaptively chosen ciphertext attacks. In *Advances in Cryptology–Crypto '93*, pages 420–434, 1993.

[11] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, 1997.

[12] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.

[13] C. Rackoff and D. Simon. Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology–Crypto '91*, pages 433–444, 1991.

[14] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology–Eurocrypt '97*, 1997.

[15] M. Stadler. Publicly verifiable secrete sharing. In *Advances in Cryptology–Eurocrypt '96*, pages 190–199, 1996.

[16] Y. Zheng and J. Seberry. Practical approaches to attaining security against adaptively chosen ciphertext attacks. In *Advances in Cryptology–Crypto '92*, pages 292–304, 1992.