Symmetric Public-Key Encryption

CUCS-191-85

Zvi Galil[1, 2, 3] Stuart Haber[1, 3] Moti Yung[1, 3, 4]

[1] Department of Computer Science, Columbia University
[2] Department of Computer Science, Tel Aviv University

Summary

Public-key encryption would seem to be inherently assymmetric, in that only messages sent to a user can be encrypted using his public key. We demonstrate that the use of interactive protocols for sending encrypted messages enables a symmetric use of public keys; we give cryptographic protocols for the following tasks:

1. Probabilistic encryption, using the same public key, both of messages that are sent to a particular user as well as of messages that the user sends to others, without compromising the key. We propose a public-key cryptosystem based on these protocols which has only one key, owned by a cryptographic server.
2. Authentication both of the sender and of the receiver of a probabilistically encrypted message.
3. Probabilistic encryption which is provably secure against both chosen-message and chosen-ciphertext attack.

---

[3] Supported in part by NSF grants MCS-8303139 and DCR-8511713.

[4] Supported in part by an IBM graduate fellowship.

1. Introduction

As introduced by Diffie and Hellman and further studied by many authors, public-key encryption would seem to be inherently assymmetric: messages sent to user A are encrypted using A's public key [6, 16]. This is true both for deterministic [15, 13] and for probabilistic [8, 3, 5] implementations of the Diffie-Hellman model.

In this paper we suggest that users follow an interactive protocol in order to send probabilistically encoded messages, and show how this allows the symmetric use of public keys. A's public key will be used to encode messages that are sent to A as well as to encode messages that A sends to others, without compromising the key. We contrast our protocol with previous interactive schemes, in which public-key encryption was used in order to distribute additional private keys that could be used symmetrically by pairs of users [9, 18]; our scheme enables symmetric use of the public key itself.

This capability is useful in a number of cryptographic settings. For example, it enables a casual user who is not registered in the central file of public keys to receive a private message. It can also be used in a cryptographic network with a trusted central server, through which all messages are routed; here only a single public key is needed (cf. [12]).

We extend our scheme so as to enable the symmetric authentication of an encoded message --- that is, the authentication both of the sender and of the receiver of the message. This is the first such scheme, in the setting of probabilistic encryption, that uses only the encryption keys.

Probabilistic encryption was proposed in order to hide from an eavesdropper all partial information about an encoded message. However, all of the systems discussed in the literature are vulnerable to chosen-ciphertext attack. We give a refinement of our protocol (based on [11]) which is provably secure against chosen-ciphertext attack. In addition, we give another symmetric public-key encryption scheme, this one based on a minimum-knowledge interactive proof-system, which is also chosen-ciphertext secure [7].

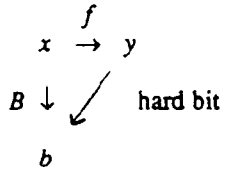
2. Background

In the model introduced by Diffie and Hellman, each user A in a public-key crypto-system has a public encryption algorithm E and a private decryption algorithm D. Any other user encrypts a message M that he wishes to send to A by computing the ciphertext $E(M)$; only A is capable of computing $D(E(M)) = M$ to recover the original message [15, 13]. In order that the ciphertext reveal no partial information about the message, it has been suggested that E and D be probabilistic algorithms [8, 3, 5].

We would like A to be able to use her own public key in order to send an encrypted message to another user B. In order to do this securely, so that no other users can decrypt the message, it seems necessary to make the transfer of a message depend on an interactive protocol between A and B. In this way B can help to choose the random input to the probabilistic encryption and decryption algorithms. In the next section we will show how to implement this idea; first we sketch the methods of probabilistic encryption that we will use.

The security of the protocols that we discuss in this paper relies on the existence of *hard bits*, that is, Boolean predicates B for which there is an efficient reduction to B of an assumedly intractable number-theoretic problem.

Specifically, we will assume that we are given functions of the following form. Let $D \subseteq \{0,1\}^n$ be a (non-sparse) set of n-bit strings, and suppose that $f: D \to D$ is a *one-way trapdoor permutation*. Suppose in addition that $B: D \to \{0,1\}$ is an (efficiently computable) Boolean predicate such that f^{-1} is efficiently reducible to the "hard bit" $B \circ f^{-1}$.*(Yao has shown that, even without such a predicate, f can be used to generate pseudo-random bits [17].)

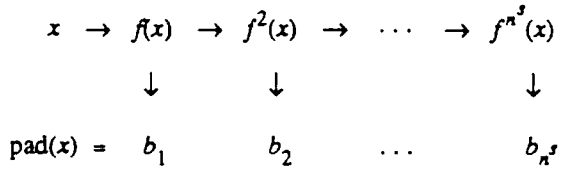


Such a function and its associated Boolean predicate may be used as a *cryptographically strong* generator of pseudo-random bits. Given any element $x \in D$, and two integers $j \le k$, we will define

$G(x, j, k)$

to be the bit-sequence

$B(f^j(x)),\ B(f^{j+1}(x)),\ \ldots,\ B(f^k(x))$.



If elements $x \in D$ are chosen at random, then the bit-sequences $\mathrm{pad}(x) = G(x, 1, n^s)$ are indistinguishable (in time polynomial in n) from truly random bit-sequences. That is, an efficient algorithm which could distinguish between the two sorts of sequences with non-negligible probability could in turn be converted to an efficient algorithm for computing f^{-1}, contradicting the assumption that f^{-1} is hard. (For a more complete account of this, see [4, 17, 2].) The provable security of these bit sequences permits us to use them to simulate one-time pads.

The schema just described is an abstraction of two different methods of pseudo-random bit-generation. That of Goldwasser, Micali, and Tong [9] requires an n-bit integer $N = pq$ which is the product of two primes satisfying either $p \equiv q \equiv 3 \bmod 8$, or $p \equiv q \equiv 7 \bmod 8$. The domain D is the set $\{x \in \mathbf{Z}_N^* \mid 0 < x < N/2, \left(\frac{x}{N}\right) = 1\}$; we define the function f by $f(x) = \pm x^2 \bmod N$, choosing either + or − so that $0 < f(x) < N/2$, and we define the predicate $B: D \to \{0,1\}$ by $B(x) = \mathrm{parity}(x)$. Both f and B can be efficiently computed. The trapdoor information for f is the factorization of N; this information enables the efficient computation of $(f^{-1})^j$ [5]. The security of the hard bit of this scheme was proved by Alexi, Chor, Goldreich, and Schnorr [1].

---

*We may also have $B: D \to \{0,1\}^k$, where B is a k-bit "predicate" all of whose bits are *simultaneously* hard. (In this case, the cryptographic applications -- modified in the obvious manner – are more efficient by a factor of k.) The proofs go through with little change.

The pseudo-random bit-generator of Blum, Blum, and Shub also involves squaring modulo a large integer N [3].

The first probabilistic encryption scheme, due to Goldwasser and Micali [8], does not use a pseudo-random bit-generator; instead, each bit of a message is encoded either as a random quadratic residue or as a random quadratic non-residue modulo a large integer N which is the product of two primes. More precisely, a public key in this scheme consists of N together with y, a quadratic non-residue mod N with Jacobi symbol +1. To encode a bit, one chooses $x \in \mathbf{Z}_N^*$ at random and sends either x^2 mod N (a random residue) or yx^2 mod N (a random non-residue), according to whether the bit is 0 or 1. For any bit-string s, we will use $E_{N,y}(s)$ to denote the set of possible encodings of s; thus, if s has length l then $E_{N,y}(s)$ consists of l-tuples of residues and non-residues mod N. Gaining any partial information about an encoded message is as hard as distinguishing residues from non-residues mod N, which appears to be an intractable problem without knowing the factorization of N. The trapdoor information which enables efficient decoding is exactly this factorization.

3. Symmetric Encryption Scheme

We describe here how to encode and decode, using a cryptographically strong bit-generator constructed with a trapdoor function as described above.

Let f be the trapdoor function whose specification is contained in user A's public file; that is, all users in the network can compute f quickly, but only A has the trapdoor information enabling her to compute f^{-1}. Generalizing the scheme of Blum and Goldwasser [5], we show how any other user B can send an encrypted message to A using f; we then describe how --- using the same function f --- A can send a securely encrypted message to B.

In both cases, the cleartext being sent is an l-bit message M (where l is polynomial in n.) For any element $x \in D$, we will use the notation $\text{pad}(x) = G(x, 1, l)$.

Protocol 1

In the 'forward' or usual direction, B chooses an element $x \in D$ at random, and computes pad(x), $C = \text{pad}(x) \oplus M$, and $X = f^{l+1}(x)$. B sends to A the encryption of M, namely the pair $[C, X]$. A can decode by computing $x = f^{-(l+1)}(X)$ and $C \oplus \text{pad}(x) = M$.

Encrypting messages in the opposite direction seems to require some additional communication. We propose the following protocol for A to send the message M to B.

Protocol 2

- A $\rightarrow$ B: "Hi"
- B chooses x at random in D, and computes pad(x) and $X = f^{l+1}(x)$
- B $\rightarrow$ A: X
- A computes $x = f^{-(l+1)}(X)$, pad(x), and $C = \text{pad}(x) \oplus M$
- A $\rightarrow$ B: C

- B computes $C \oplus \text{pad}(x) = M$

Notice that the information that is available to the eavesdropping adversary, namely X (which serves as an encoding of the seed x) and C, is the same in both protocols.

Under known-message attack, the present scheme is as secure as the problem of inverting f. To be more precise, assume that an eavesdropper witnesses polynomially many executions of one or both of the above protocols (polynomial in n), and that he knows the message M_i that was sent in the ith execution (either sent by user B_i to A, following Protocol 1, or sent by A to user B_i, following Protocol 2). Suppose that, after some polynomially bounded computation, the eavesdropper is able either (a) to correctly simulate the behavior of A in order to send a message of his (the eavesdropper's) choice; or (b) to decode (with probability non-negligibly better than 1/2) one bit of the next message that A sends or receives. Then our eavesdropper must be able to compute f^{-1} quickly. The proof relies on the (polynomial-time) indistinguishability of the pseudo-random bits of pad(x) from truly random bits.

We now show how the bit-by-bit probabilistic encryption scheme of Goldwasser and Micali can also be adapted so as to encode messages either to or from the owner of a public key. Let user A have the public key (N, y).

As in the original scheme, user B can send a message M to A by probabilistically computing an element $e \in E_{N,y}(M)$ and sending it to A. Knowing the trapdoor information, A can recover M from e. We will call this Protocol 1*.

In order for A to send an l-bit message M to B, the two users execute the following protocol.

Protocol 2*

- A $\rightarrow$ B: "Hi"
- B chooses $p \in \{0,1\}^l$ at random, and probabilistically computes $e \in E_{N,y}(p)$
- B $\rightarrow$ A: e
- A computes p and $C = p \oplus M$
- A $\rightarrow$ B: C
- B computes $C \oplus p = M$

As before, Protocol 2* is as secure as encryption in the usual direction.

3.1. Applications

An immediate application of these protocols is to allow encoded messages to be sent to casual users in a network without requiring that they undergo any special procedure such as having a key registered in a public-key library.

If a network includes a trusted central server, then any message can be sent via the server, encoded using the server's public key; when A wants to send a message to B, she sends it to the server using Protocol 1, and the server

sends it on to B using Protocol 2. In this case, only a single public key is needed; there is no need to initialize and maintain a public-key library. This may increase the security of the system and decrease its cost.

We observe that if this suggestion is implemented using bit-by-bit probabilistic encryption, i.e. with Protocol 1* and Protocol 2*, then it is easy to prove that an adversary gains no advantage whatsoever from over-hearing (what he knows to be) two encryptions of the same message.

4. Authentication

What has been presented so far is purely an encryption scheme. We now assume that each user has his own public key, and we consider the problem of user authentication.

We distinguish two authentication problems that arise when A sends a message to B. There is the problem of *sender authentication*, which is to convince B that it was indeed A who sent the message, and the complementary problem of *receiver authentication*, which is to convince A that it is indeed B who received the message.

In the usual public-key encryption scheme, in which messages to user A are encrypted using A's public key, receiver authentication is assured by the fact that only A knows the private key which is necessary for decryption; on the other hand, there is no automatic provision for authenticating the sender of a message. A malicious user C can send a message to A, claiming to be B.

In a standard deterministic public-key scheme, the usual modification to assure sender authentication is by means of a "digital signature", specifying that the message be further encrypted using the sender's private key; that is, B encrypts his message M to A as $D_B E_A(M)$ [6]. On the other hand, of the several proposed probabilistic public-key schemes, none seems to allow for an easy modification that assures sender authentication.

In our scheme, whereby A sends messages to others using her own public key, the authentication problems are correspondingly reversed. Sender authentication is guaranteed by the fact that only A knows the trapdoor information (for inverting f) which is necessary for encryption, while receiver authentication is no longer assured. That is, a malicious user C can masquerade as B in an execution of Protocol 2, since the only secret information in the scheme is that of A.

We suggest that a simple modification of the protocols presented in the last secton allows A to send M to B so that both A and B can authenticate their identities to each other. Let f_A, f_B denote the public encryption functions for users A and B, and let D_A, D_B be their domains. In the following protocol for A to send an authenticated l-bit message M to B, pad(x) and pad(y) are computed using f_A and f_B, respectively.

Protocol 3

- A $\rightarrow$ B: "Hi, this is A sending a message to B."
- B chooses x at random in D_A,
  and computes pad(x) and $X = f_A^{l+1}(x)$.
- B $\rightarrow$ A: X
- A computes $x = f_A^{-(l+1)}(X)$, pad(x),

chooses y at random in D_B,
computes pad(y), $Y = f_B^{l+1}(y)$, and $C = M \oplus \text{pad}(x) \oplus \text{pad}(y)$

- A $\rightarrow$ B: $[C, Y]$
- B computes $y = f_B^{-(l+1)}(Y)$, pad(y),
  and $C \oplus \text{pad}(x) \oplus \text{pad}(y) = M$

As with Protocols 1 and 2, this protocol is secure against known-message attack. Moreover, impersonating A or B is as hard as inverting f_A or f_B, respectively.

5. Chosen-Ciphertext Security

In a chosen-ciphertext attack, the adversary is allowed to have a ciphertext of his choice decrypted. Several proposed cryptosystems which are secure against weaker sorts of cryptanalytic attack are easily seen to be vulnerable to the chosen-ciphertext attack. In this section we show how introducing interaction to the cryptosystem enables us, for the first time, to achieve provable security against this attack.

For an example of the chosen-ciphertext attack, consider Protocol 2 specified above. B, the receiver of the message, can cheat in the following way. Instead of choosing x at random in D and continuing with the protocol, B chooses an element $y \in D$, setting $X = f^{l-1}(y) = f^{l+1}(x)$, where now x is an element which he does not know. However, B *does* know all but the first bit of pad(x). A, suspecting nothing amiss, sends B the encryption pad(x) $\oplus M$, all but the first bit of which B can easily decode. If the context of the message allows B to infer the value of that first bit, then he has learned the value of the hard bit $b_1 = B \bullet f^{-1}(y)$ of a number y of his choice. Since our original assumption was that inverting f is efficiently reducible to $B \bullet f^{-1}$, this is a successful attack.

$$\begin{array}{ccccccccccc} x = f^{-2}(y) & \rightarrow & f^{-1}(y) & \rightarrow & y & \rightarrow & \cdots & \rightarrow & f^{l-2}(y) & \rightarrow & X = f^{l-1}(y) = f^{l+1}(x) \\ & & \downarrow & & \downarrow & & & & \downarrow & & \\ \text{pad}(x) = & & b_1 & & b_2 & & \cdots & & b_l & & \end{array}$$

With the implementation using the bit-generator of Goldwasser, Micali, and Tong (see section 2), we now show how to refine Protocol 2 so that the resulting scheme is chosen-ciphertext secure. This refinement, based on the work of [11], is due to Silvio Micali.

Protocol 4

- A $\rightarrow$ B: "Hi"
- B chooses at random $x, x_1, x_2, \ldots, x_n$ in Z_N^*,
  and computes pad(x), $X = f^{l+1}(x)$, and $X_i = f^{l+1}(x_i)$ $(i = 1, \ldots, n)$
- B $\rightarrow$ A: X, X_i $(i = 1, \ldots, n)$
- A $\rightarrow$ B: a random subset $S \subset \{1, \ldots, n\}$ of size $n/2$

- B $\rightarrow$ A: $\{x_i | i \in S\}$ and $\{xx_j \bmod N | j \notin S\}$
- A checks that $f^{l+1}(x_i) = X_i$ for $i \in S$ and $f^{l+1}(xx_j) \equiv XX_j \bmod N$ for $j \notin S$;
  if so then A $\rightarrow$ B: $M \oplus \mathrm{pad}(x)$,
  otherwise A halts (detecting cheating)

This protocol is secure against an eavesdropper; furthermore, it is secure against chosen-ciphertext attack by B. The protocol ensures that if A does not halt the transaction, detecting cheating, then with very high probability B has not cheated. In fact, the refinement may be regarded as a protocol during which B proves to A that he knows the number x, without gaining any additional knowledge -- for example, about the integer N. (Such a protocol is called a *minimum-knowledge interactive proof system* [11, 7].)

The same protocol will work, *mutatis mutandis*, as long as D is a group (in which we can compute efficiently) and f is an automorphism of the group.

We now sketch another solution to the problem of the chosen-ciphertext attack. In order to avoid any chance that A's public key be compromised by B's clever choice of random input to a probabilistic encryption protocol, we require that A choose the input; we proceed as follows. The solution has two stages. In the first stage, A chooses a sequence of random bits and transmits them one by one to B; these transmissions, of course, must be cryptographically secure. This can be accomplished by means of the minimum-knowledge protocol introduced in [7]; following this protocol, A can prove to B the value of a Boolean predicate in such a way that no eavesdropper can tell whether that value is 0 or 1, and so that B gains no additional knowledge at all. In the second stage, the sequence of bits can be used as the seed for a pseudo-random bit-generator; in this way, A and B simulate a shared one-time pad (of length polynomial in the length of the seed exchanged in the first stage).

6. Conclusions

By extending the capabilities of public-key encryption, we have demonstrated the power that interaction adds to the capabilities of cryptographic systems. Further study of interactive protocols promises to be of great use to cryptography. We take note here of the work of Rackoff in rigorous modeling of cryptosystems, including interaction [14].

The problem remains open of specifying a cryptosystem which does *not* use interaction and proving it secure against chosen-ciphertext attack. Perhaps a first step in this direction would be to allow some sort of limited interaction. For example, in the signature scheme of [10], which is secure against the analogous attack, the dependence of any signature on the history of previously signed messages (or on a random-function construction) may be regarded as an instance of interaction with that history (or with the random-function generator).

Acknowledgments

We would like to thank Silvio Micali for his encouragement of and major contributions to this work.

References

[1] W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr.
RSA/Rabin bits are 1/2 + 1/poly(log *N*) secure.
In *Proc. 25th FOCS*, pages 449-457. IEEE, 1984.

[2] Angluin, Dana and Lichtenstein, David.
Provable Security of Cryptosystems: a Survey.
Technical Report YALEU/DCS/TR-288, Yale University, October, 1983.

[3] L. Blum, M. Blum, and M. Shub.
A simple secure pseudo-random number generator.
In *Crypto '82*. 1982.

[4] Blum, M. and Micali, S.
How to generate cryptographically strong sequences of pseudo-random bits.
In *Proc. 23rd FOCS*, pages 112-117. IEEE, 1982.

[5] M. Blum and S. Goldwasser.
An efficient probabilistic public-key encryption scheme which hides all partial information.
In *Crypto '84*. 1984.

[6] W. Diffie and M.E. Hellman.
New directions in cryptography.
IEEE Trans. on Inform. Theory IT-22:644-654, November, 1976.

[7] Z. Galil, S. Haber, and M. Yung.
A private interactive test of a Boolean predicate and minimum-knowledge public-key cryptosystems.
In *Proc. 26th FOCS*. IEEE, 1985.

[8] S. Goldwasser and S. Micali.
Probabilistic encryption and how to play mental poker keeping secret all partial information.
In *Proc. 14th STOC*, pages 365-377. ACM, 1982.

[9] S. Goldwasser, S. Micali, and P. Tong.
Why and how to establish a private code on a public network.
In *Proc. 23rd FOCS*, pages 134-144. IEEE, 1982.

[10] S. Goldwasser, S. Micali, and R.L. Rivest.
A "paradoxical" solution to the signature problem.
In *Proc. 25th FOCS*, pages 441-448. IEEE, 1984.

[11] S. Goldwasser, S. Micali, and C. Rackoff.
The knowledge complexity of interactive proof systems.
In *Proc. 17th STOC*, pages 291-304. ACM, 1985.

[12] R.M. Needham and M.D. Schroeder.
Using encryption for authentication in large networks of computers.
Communications of the ACM 21(12):993-99, December, 1978.

[13] M. Rabin.
Digitalized signatures and public-key functions as intractable as factorization.
Technical Report LCS/TR-212, MIT, January, 1979.

[14] C. Rackoff.
Cryptography: lecture notes.
1985.

[15] R.L. Rivest, A. Shamir, and L. Adleman.
A method for obtaining digital signatures and public key cryptosystems.
Communications of the ACM 21(2):120-126, February, 1978.

[16] G.J. Simmons.
Symmetric and assymmetric encryption.
Computing Surveys 11:305-330, December, 1979.

[17] A.C. Yao.
Theory and applications of trapdoor functions.
In *Proc. 23rd FOCS*, pages 80-91. IEEE, 1982.

[18] M. Yung.
Cryptoprotocols: subscription to a public-key, secret blocking and the multi-player mental poker game.
In *Crypto '84*. 1984.