A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms

TAHER ELGAMAL, MEMBER, IEEE

Abstract—**A new signature scheme is proposed, together with an implementation of the Diffie–Hellman key distribution scheme that achieves a public key cryptosystem. The security of both systems relies on the difficulty of computing discrete logarithms over finite fields.**

I. Introduction

In 1976, Diffie and Hellman [3] introduced the concept of public key cryptography. Since then, several attempts have been made to find practical public key systems (see, for example, [6], [7], [9]) depending on the difficulty of solving some problems. For example, the Rives–Shamir–Adleman (RSA) system [9] depends on the difficulty of factoring large integers. This paper presents systems that rely on the difficulty of computing logarithms over finite fields.

Section II shows a way to implement the public key distribution scheme introduced by Diffie and Hellman [3] to encrypt and decrypt messages. The security of this system is equivalent to that of the distribution scheme. Section III introduces a new digital signature scheme that depends on the difficulty of computing discrete logarithms over finite fields. It is not yet proved that breaking the system is equivalent to computing discrete logarithms. Section IV develops some attacks on the signature scheme, none of which seems to break it. Section V gives some properties of the system. Section VI contains a conclusion and some remarks.

II. The Public Key System

First, the Diffie–Hellman key distribution scheme is reviewed. Suppose that A and B want to share a secret K_{AB}, where A has a secret x_A and B has a secret x_B. Let p be a large prime and α be a primitive element mod p, both known. A computes $y_A \equiv \alpha^{x_A} \bmod p$, and sends y_A. Similarly, B computes $y_B \equiv \alpha^{x_B} \bmod p$ and sends y_B. Then the secret K_{AB} is computed as

$$\begin{aligned} K_{AB} &\equiv \alpha^{x_A x_B} \bmod p \\ &\equiv y_A^{x_B} \bmod p \\ &\equiv y_B^{x_A} \bmod p. \end{aligned}$$

Manuscript received February 6, 1984; revised November 12, 1984. This work was supported by the National Science Foundation under Grant ECS83 07741. The material in this paper was presented at the IEEE Crypto '84 Conference, August 1984, Santa Barbara, CA.

The author was with the Information Systems Laboratory, Stanford University, Stanford, CA. He is now with Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304, USA.

Hence both A and B are able to compute K_{AB}. But, for an intruder, computing K_{AB} appears to be difficult. It is not yet proved that breaking the system is equivalent to computing discrete logarithms. For more details refer to [3].

In any of the cryptographic systems based on discrete logarithms, p must be chosen such that $p-1$ has at least one large prime factor. If $p-1$ has only small prime factors, then computing discrete logarithms is easy (see [8]).

Now suppose that A wants to send B a message m, where $0 \le m \le p-1$. First A chooses a number k uniformly between 0 and $p-1$. Note that k will serve as the secret x_A in the key distribution scheme. Then A computes the "key"

$$K \equiv y_B^k \bmod p, \tag{1}$$

where $y_B \equiv \alpha^{x_B} \bmod p$ is either in a public file or is sent by B. The encrypted message (or ciphertext) is then the pair (c_1, c_2), where

$$c_1 \equiv \alpha^k \bmod p \qquad c_2 \equiv Km \bmod p \tag{2}$$

and K is computed in (1).

Note that the size of the ciphertext is double the size of the message. Also note that the multiplication operation in (2) can be replaced by any other invertible operation such as addition mod p.

The decryption operation splits into two parts. The first step is recovering K, which is easy for B since $K \equiv (\alpha^k)^{x_B} \equiv c_1^{x_B} \bmod p$, and x_B is known to B only. The second step is to divide c_2 by K and recover the message m.

The public file consists of one entry for each user, namely y_i for user i (since α and p are known for all users). It is possible that each user chooses his own α and p, which is preferable from the security point of view although that will triple the size of the public file.

It is not advisable to use the same value k for enciphering more than one block of the message, since if k is used more than once, knowledge of one block m_1 of the message enables an intruder to compute other blocks as follows. Let

$$c_{1,1} \equiv \alpha^k \bmod p \qquad c_{2,1} \equiv m_1 K \bmod p,$$

$$c_{1,2} \equiv \alpha^k \bmod p \qquad c_{2,2} \equiv m_2 K \bmod p.$$

Then $m_1/m_2 \equiv c_{2,1}/c_{2,2} \bmod p$, and m_2 is easily computed if m_1 is known.

Breaking the system is equivalent to breaking the Diffie–Hellman distribution scheme. First, if m can be computed from c_1, c_2, and y, then K can also be computed from y, c_1, and c_2 (which appears like a random

0018-9448/85/0700-0469$01.00

number since k and m are unknown). That is equivalent to breaking the distribution scheme. Second, (even if m is known) computing k or x from c_1, c_2, and y is equivalent to computing discrete logarithms. The reason is that both x and k appear in the exponent in y and c_1.

III. A Digital Signature Scheme

A new signature scheme is described in this section. The public file contains the same public keys for encrypting messages as well as verifying signatures.

Let m be a document to be signed, where $0 \le m \le p-1$. The public file still consists of the public key $y \equiv \alpha^x \bmod p$ for each user. To sign a document, a user A should be able to use the secret key x_A to find a signature for m in such a way that all users can verify the authenticity of the signature by using the public key y_A (together with α and p), and no one can forge a signature without knowing the secret x_A.

The signature for m is the pair (r, s), $0 \le r, s < p-1$, chosen such that the equation

$$\alpha^m \equiv y^r r^s \bmod p \tag{3}$$

is satisfied.

A. The Signing Procedure

The signing procedure consists of the following three steps.

1) Choose a random number k, uniformly between 0 and $p-1$, such that $\gcd(k, p-1) = 1$.

2) Compute

$$r \equiv \alpha^k \bmod p. \tag{4}$$

3) Now (3) can be written as

$$\alpha^m \equiv \alpha^{xr}\alpha^{ks} \bmod p, \tag{5}$$

which can be solved for s by using

$$m \equiv xr + ks \bmod (p-1). \tag{6}$$

Equation (6) has a solution for s if k is chosen such that $\gcd(k, p-1) = 1$.

B. The Verification Procedure

Given m, r, and s, it is easy to verify the authenticity of the signature by computing both sides of (3) and checking that they are equal.

Note 1: As will be shown in Section IV, the value of k chosen in step 1) should never be used more than once. This can be guaranteed, for example, by using as a "k generator" a DES chip used in the counter mode as a stream cipher.

IV. Some Attacks on the Signature Scheme

This section introduces some of the possible attacks on the signature scheme. Some of these attacks are easily shown to be equivalent to computing discrete logarithms over GF(p). It has not yet been proved that breaking the signature scheme is equivalent to computing discrete logarithms, or equivalent to breaking the distribution scheme. However, none of the attacks shown in this section appear to break the system. The reader is encouraged to develop new attacks, or find fast algorithms to perform one of the attacks described in this section. The attacks will be divided into two groups. The first group includes some attacks for recovering the secret key x, and in the second group we show some attacks for forging signatures without recovering x.

A. Attacks Aiming to Recover x

Attack 1: Given $\{m_i: i = 1, 2, \cdots, l\}$ documents, together with the corresponding signatures $\{(r_i, s_i): i = 1, 2, \cdots, l\}$, an intruder may try to solve l equations of the form (6). Since there are $l+1$ unknowns (since each signature uses a different k), the system of equations is underdetermined and the number of solutions is large. The reason is that each value for x yields a solution for the k_i since a system of linear equations with a diagonal matrix of coefficients will result. Since $p-1$ is chosen to have at least one large prime factor q, recovering $x \bmod q$ requires an exponential number of message-signature pairs.

Note 2: If any k is used twice in the signing, then the system of equations is uniquely determined and x can be recovered. So for the system to be secure, any value of k should never be used twice.

Attack 2: Trying to solve equations of the form (3) is always equivalent to computing discrete logarithms over GF(p), since both unknowns x and k appear in the exponent.

Attack 3: An intruder might try to develop some linear dependencies among the unknowns $\{k_i, i = 1, 2, \cdots, l\}$. This is also equivalent to computing discrete logarithms since if $k_i \equiv ck_j \bmod (p-1)$, then $r_i \equiv r_j^c \bmod p$, and if c can be computed then computing discrete logarithms is easy.

B. Attacks for Forging Signatures

Attack 4: Given a document m, a forger may try to find r, s such that (3) is satisfied. If $r \equiv \alpha^j \bmod p$ is fixed for some j chosen at random, then computing s is equivalent to solving a discrete logarithm problem over GF(p).

If the forger fixes s first, then r could be computed from the equation

$$r^s y^r \equiv A \bmod p. \tag{7}$$

Solving equation (7) for r is not yet proved to be at least as hard as computing discrete logarithms, but we believe that it is not feasible to solve (7) in polynomial time. The reader is encouraged to find a polynomial time algorithm for solving (7).

Attack 5: It seems possible that (3) can be solved for both r and s simultaneously, but we have not been able to find an efficient algorithm to do that.

Attack 6: The signature scheme allows the following attack, whereby the intruder, knowing one legitimate signature for one message, can generate other legitimate sig-

natures and messages. This attack does not allow the intruder to sign an arbitrary message and therefore does not break the system. This property exists in all the existing digital signature schemes and can be avoided by either requiring that m has to have a certain structure or by applying a one-way function to the message m before signing it.

Given a signature (r, s) for the message (m), then

$$\alpha^m = y^r r^s \bmod p.$$

Select integers A, B, and C arbitrarily such that $(Ar - Cs)$ is relatively prime to $p - 1$. Set

$$r' \equiv r^A \alpha^B y^C \bmod p,$$
$$s' \equiv sr'/(Ar - Cs) \bmod (p - 1),$$
$$m' \equiv r'(Am + Bs)/(Ar - Cs) \bmod (p - 1).$$

Then it is claimed that (r', s') signs the message (m'). Calculate

$$\begin{aligned} y^{r'} r'^{s'} &\equiv y^{r'} \left(r^A \alpha^B y^C\right)^{sr'/(Ar-Cs)} \\ &\equiv \left(y^{r'Ar - r'Cs + r'Cs} r^{Asr'} \alpha^{Bsr'}\right)^{1/(Ar-Cs)} \\ &\equiv \left(\left(y^r r^s\right)^{Ar'} \alpha^{Bsr'}\right)^{1/(Ar-Cs)} \\ &\equiv \alpha^{(mAr' + Bsr')/(Ar-Cs)} \\ &\equiv \alpha^{m'} \end{aligned}$$

(all calculations mod p).

As a special case, setting $A = 0$, legitimate signatures can be generated with corresponding messages without ever seeing any signatures:

$$r' \equiv \alpha^B y^C \bmod p,$$
$$s' \equiv -r'/C \bmod (p - 1),$$
$$m' \equiv -r'B/C \bmod (p - 1).$$

It can be shown that (r', s') signs (m').

V. Properties of Our System and Comparison to Other Signature Schemes and Public Key Systems

Let m be the number of bits in either p for the discrete logarithm problem or n for the integer factoring problem. Then the best known algorithm for both computing discrete logarithms and factoring integers (which is the function used in some of the existing systems such as the RSA system [9]) is given by (see [1], [5], [10])

$$O\left(\exp \sqrt{cm \ln m}\right), \tag{8}$$

where the best estimate for c is $c = 0.69$ for factoring integers (due to Schnorr and Lenstra [10]), as well as for discrete logarithms over GF(p) (see [5]). These estimates imply that we have to use numbers that are about the size of the numbers used in the RSA system in order to obtain the same level of security (assuming the current value for c for both the discrete logarithms problem and the integer factorization problem). So the size of the public file is larger than that for the RSA system. (For the RSA system, each user has one entry n as his public key, together with the encryption key in the public file.)

A. Properties of the Public Key System

As shown above, our system differs from the other known systems. First, due to the randomization in the enciphering operation, the cipher text for a given message m is not repeated, i.e., if we encipher the same message twice, we will not get the same cipher text $\{c_1, c_2\}$. This prevents attacks like a probable text attack where if the intruder suspects that the plain text is, for example, m, then he tries to encipher m and finds out if it was really m. This attack, and similar ones, will not succeed since the original sender chose a random number k for enciphering, and different values of k will yield different values of $\{c_1, c_2\}$. Also, due to the structure of our system, there is no obvious relation between the enciphering of m_1, m_2, and $m_1 m_2$, or any other simple function of m_1 and m_2. This is not the case for the known systems, such as the RSA system.

Suppose that p is of about the same size as that required for n in the case of the RSA system. Then the size of the cipher text is double the size of the corresponding RSA cipher text.

For the enciphering operation, two exponentiations are required. That is equivalent to about $2 \log p$ multiplications in GF(p). For the deciphering operation only one exponentiation (plus one division) is needed.

B. Properties of the Signature Scheme

For the signature scheme using the above arguments for the sizes of the numbers in our system and the RSA system, the signature is double the size of the document. Then the size of the signature is the same size as that needed for the RSA scheme, and half the size of the signature for the new signature scheme that depends on quadratic forms published by Ong and Schnorr [6], and also Ong, Schnorr, and Shamir [7] (since both systems are based on the integer factoring problem). The Ong–Schnorr–Shamir system has been broken by Pollard and new variations are being suggested. Thus, it is not clear at the present time whether a secure system based on modular equations can be found, and hence no further remarks will be made regarding these schemes.

Note that, since the number of signatures is p^2, while the number of documents is only p, each document m has a lot of signatures but any signature signs only one document.

For the signing procedure, one exponentiation (plus a few multiplications) is needed. To verify a signature, it seems that three exponentiations are needed, but it was pointed to the author by Shamir that only 1.875 exponentiations are needed. This is done by representing the three exponents m, r, s in their binary expansion. At each step square the number $\alpha^{-1} yr$ and divide by the necessary

factor to account for the different expansions of m, r, and s. The different multiples of α^{-1}, y, and r can be stored in a table consisting of eight entries. We expect that 0.875 of the time a multiplication is needed. That accounts for the 1.875 exponentiations needed.

VI. Conclusions and Remarks

The paper described a public key cryptosystem and a signature scheme based on the difficulty of computing discrete logarithms over finite fields. The systems are only described in $GF(p)$. The public key system can be easily extended to any $GF(p^m)$, but recent progress in computing discrete logarithms over $GF(p^m)$ where m is large (see [2, 5]) makes the key size required very large for the system to be secure. The subexponential time algorithm has been extended to $GF(p^2)$ [4] and it appears that it can be extended to all finite fields, but the estimates for the running time for the fields $GF(p^m)$ with a small m seem better at the present time. Hence, it seems that it is better to use $GF(p^m)$ with $m = 3$ or 4 for implementing a cryptographic system. The estimates for the running time of computing discrete logarithms and for factoring integers are the best known so far, and if the estimates remain the same, then, for the same security level, the size of the public key file and the size of the cipher text will be double the size of those for the RSA system.

Acknowledgment

The author would like to thank a referee for including Attack 6 described in Section IV.

References

[1] L. Adleman, "A subexponential algorithm for the discrete logarithm problem with applications to cryptography," in *Proc. 20th IEEE Symp. Foundations of Computer Science* 1979, pp. 55–60.

[2] D. Coppersmith, "Fast evaluation of logarithms in fields of characteristic two," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 587–594, 1984.

[3] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 472–492, 1976.

[4] T. ElGamal, "A subexponential-time algorithm for computing discrete logarithms over $GF(p^2)$," *IEEE Trans. Inform. Theory*, this issue.

[5] A. Odlyzko, "Discrete logarithms in finite fields and their cryptographic significance," *Proc. Eurocrypt 84*, to appear.

[6] H. Ong and C. Schnorr, "Signatures through approximate representations by quadratic forms," to appear.

[7] H. Ong, C. Schnorr, and A. Shamir, "An efficient signature scheme based on quadratic forms," in *Proc. 16th ACM Symp. Theoretical Computer Science*, 1984, pp. 208–216.

[8] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 106–110, 1978.

[9] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[10] C. Schnorr and H. W. Lenstra Jr., "A Monte Carlo factoring algorithm with finite storage," *Math. Comput.*, vol. 43, pp. 289–311, 1984.