Novel Public Key Encryption Technique Based on Multiple Chaotic Systems

Ranjan Bose

Department of Electrical Engineering, IIT Delhi, Hauz Khas, New Delhi 110016, India

(Received 27 March 2005; published 26 August 2005)

Public key encryption was first introduced by Diffie and Hellman in 1976. Since then, the Diffie-Hellman key exchange protocol has been used in developing public key systems such as Rivest-Shamir-Adleman and elliptic curve cryptography. Chaotic functions, so far, have been used for symmetric cryptography only. In this Letter we propose, for the first time, a methodology to use multiple chaotic systems and a set of linear functions for key exchange over an insecure channel. To the best of our knowledge, this is the first Letter that reports the use of chaotic systems for public key cryptography. We have shown that the security of the proposed algorithm grows as $(NP)^m$, where N, P, and m are large numbers that can be chosen as the parameters of the cryptosystem.

DOI: 10.1103/PhysRevLett.95.098702 PACS numbers: 05.45.Vx, 89.70.+c

Cryptography is the study or science of secret writing, and a cryptosystem is a system in which either information is transformed into secret writing (called ciphertext) or ciphertext is transformed back to the above-mentioned information (called plain text). The transformation process just mentioned is controlled by what is called a ''key.'' Hence, to transform in either direction, the correct key is needed. Simmons classifies cryptosystems as either symmetric (secret key) or asymmetric (public key) [1].

In a secret-key cryptosystem, private conversation between two persons is established by using one key, known to both of them. This key is used for transformation to cipher text (enciphering) as well as transformation back to plain text (deciphering). A disadvantage of this scheme is the fact that the secrecy of communication depends upon the trustworthiness or reliability of the two persons. Another disadvantage is that in a multichannel scenario one has to keep a lot of keys secret to maintain private communication with different people.

In a public key cryptosystem, there exist two separate keys known as the enciphering key (to encipher) and the deciphering key (to decipher). These keys decipher in such a way that, knowing one of the keys, it is computationally infeasible to determine the other key. This concept was first introduced in 1976 by Diffie-Hellman in their seminal paper [2].

In the past few years, many types of chaos-generating systems have been proposed and analyzed in various fields. Specifically, chaos functions have found applications in cryptography as it can be used efficiently for random number generation [3,4]. Many cryptosystems based on chaos have been proposed recently [5–12]. These systems are based on the characteristics of chaotic systems like sensitivity to parameters and initial conditions, ergodicity, and mixing property, which are analogous to the requirements of pseudorandom coding and cryptography [13,14]. Many of these systems have been shown to possess weaknesses and vulnerability [15–18]. Some insightful suggestions have been proposed in [19–21], which recommend a focus of future research on the relationship of chaos and cryptography. In [19], the authors have introduced a couple of chaotic systems based on a pseudorandom bit generation scheme and have shown that, apart from satisfying the cryptographic properties, it compares well with conventional random number generators [22,23].

Chaotic systems, so far, have been used only for symmetric cryptography (also called secret-key encryption) where the *same* key is used for encoding and decoding. This Letter deals with asymmetric key encryption (also called public key encryption), which is often used to exchange the symmetric keys. In this Letter we have proposed, for the first time, a method for public key encryption using a combination of chaotic systems and linear systems. We have shown that the time taken to establish the key is of order NP, whereas the time for the adversary to break the system is $(NP)^m$, where N, P, and m are large numbers, defined by the user.

In 1976, Diffie and Hellman introduced public key cryptosystem in their seminal paper [2], which initiated a revolution in cryptography. They presented the first protocol with public keys, the so-called Diffie-Hellman (DH) protocol for public key distribution. The protocol allows two parties, Alice and Bob, who are connected by an authenticated but otherwise insecure channel, to generate a secret key, which is difficult to compute for an adversary Eve overhearing the communication between Alice and Bob.

The protocol works as follows. Let G be a finite cyclic group with order $|G|$ generated by g. In order to generate a mutual secret key, Alice and Bob secretly choose integers s_A and s_B, respectively, at random from the interval $[0, |G| - 1]$. Then they compute secretly $a_A = g^{s_A}$ and $a_B = g^{s_B}$, respectively, and exchange these group elements over the insecure public channel. Finally, Alice and Bob compute $a_{AB} = a_B^{s_A} = g^{s_A s_B}$ and $a_{BA} = a_A^{s_B} = g^{s_B s_A}$, respectively. It may be noted that $a_{AB} = a_{BA}$, and hence this quantity can be used as a secret key shared by Alice and Bob. Figure 1 shows a mechanical analog of the Diffie-

 © 2005 The American Physical Society

Hellman protocol [24]. This scheme has been used for developing public key cryptosystems such as Rivest-Shamir-Adleman [25] and elliptic curve cryptosystems [26–28].

Chaotic functions have found applications in cryptography, as it can be used efficiently for random number generation [6,8,9]. These systems are based on the characteristics of chaotic systems like sensitivity to parameters and initial conditions, ergodicity, and mixing property, which are analogous to the requirements of pseudorandom coding and cryptography [13,14]. But, they have been found to have some problems, which can be overcome by using a couple of chaotic systems as suggested in [19]. We next discuss, briefly, the system suggested in [19] and extend it to a general case of multiple chaotic systems.

Assume there are two different one-dimensional chaotic maps $F_1(x_1, p_1)$ and $F_2(x_2, p_2)$ such that $x_1(i+1) = F_1(x_1(i), p_1)$, $x_2(i+1) = F_2(x_2(i), p_2)$, where p_1, p_2 are control parameters, $x_1(0)$, $x_2(0)$ are initial conditions, and $\{x_1(i)\}$, $\{x_2(i)\}$ denote the two chaotic orbits.

Define a pseudorandom bit sequence $k_i = g(x_1(i), x_2(i))$, where

$$g(x_1, x_2) = \begin{cases} 1, & \text{if } x_1 > x_2, \\ \text{no output}, & \text{if } x_1 = x_2, \\ 0, & \text{if } x_1 < x_2. \end{cases} \tag{1}$$

Subject to certain requirements [19], the above generated sequence will have good cryptographic properties, and it is called "a couple of chaotic systems based pseudorandom number generator" (CCS-PRNG).

This can be generalized to the case of m chaotic maps. Let there be m different chaotic maps $F_1, F_2, \ldots, F_m$ defined as $x_1(i+1) = F_1(x_1(i), p_1)$, $x_2(i+1) = F_2(x_2(i), p_2), \ldots, x_m(i+1) = F_m(x_m(i), p_m)$, where $p_1, p_2, \ldots, p_m$ are control parameters, $x_1(0), x_2(0), \ldots, x_m(0)$ are the initial conditions, and $\{x_1(i)\}, \{x_2(i)\}, \ldots, \{x_m(i)\}$ denote the m chaotic orbits.

Define a pseudorandom sequence $k_i = g(x_1(i), x_2(i), \ldots, x_m(i))$, where

$$g(x_1, x_2, \ldots, x_m) = \begin{cases} 1, & \text{if } x_1 > x_2, x_1 > x_3, \ldots, x_1 > x_m, \\ 2, & \text{if } x_2 > x_1, x_2 > x_3, \ldots, x_2 > x_m, \\ & \vdots \\ r, & \text{if } x_r > x_1, x_r > x_2, \ldots, x_r > x_m, \\ & \vdots \\ m, & \text{if } x_m > x_1, x_m > x_2, \ldots, x_m > x_{m-1}. \end{cases} \tag{2}$$

Note that when there are two or more largest values in the set, the one with the lower index is chosen. It can easily be shown that this sequence will also have good cryptographic properties. We call this the "m-chaotic systems based pseudorandom number generator" (m-CS PRNG).

We first describe how to use a couple of chaotic systems (CCS) in conjunction with DH protocol for key exchange. Later we shall generalize it for m-CS. Suppose Alice and Bob wish to agree upon a key that will later be used in conjunction with a classical cryptosystem. They decide on a starting value x_0 and two linear functions $f_1(x)$ and $f_2(x)$. The linear functions satisfy the condition $f_1 \circ f_2 = f_2 \circ f_1$, and therefore the sequence of operation does not matter. Define a selection function, $h(x, k)$, as follows:

$$h(x, k) = \begin{cases} f_1(x) & \text{if } k = 0, \\ f_2(x) & \text{if } k = 1. \end{cases} \tag{3}$$

We can then write the iterative form as

$$x_i = h(x_{i-1}, k_i). \tag{4}$$

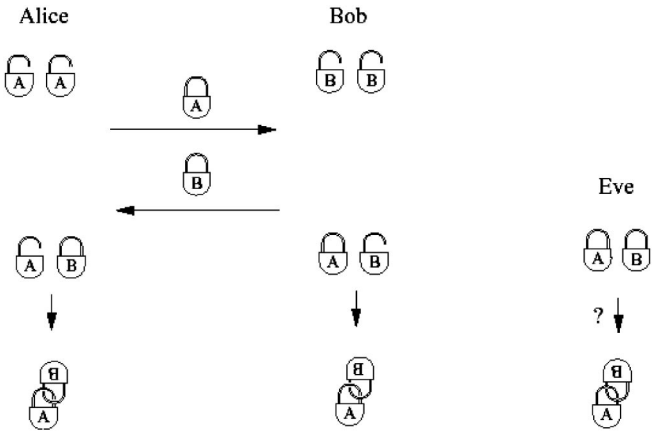

FIG. 1. A mechanical analog of the Diffie-Hellman protocol.

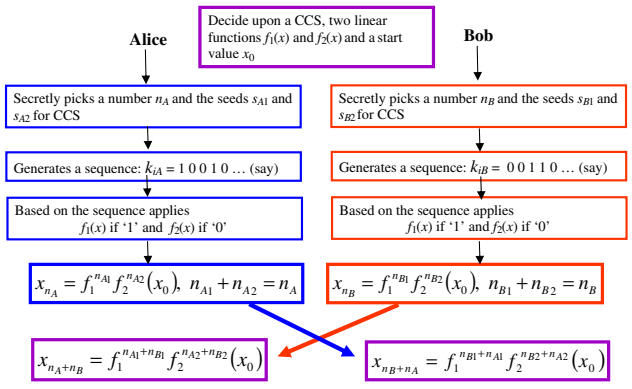

FIG. 2 (color online). The proposed key exchange protocol using CCS.

The key exchange algorithm based on CCS is described below.

Step 1: Alice and Bob, who wish to exchange the key, publicly agree on a common CCS, two linear functions $f_1(x)$ and $f_2(x)$, and a start value x_0, as depicted in Fig. 2.

Step 2: Alice secretly chooses two seeds s_{A1} and s_{A2} and a large number n_A and uses them as inputs to a CCS PRNG to generate a random bit sequence, k_{iA}, based on Eq. (1). n_A is the number of bits in the sequence generated. This sequence k_{iA} is provided as input to Eq. (4), along with x_0 to iterate and give resultant x_{nA}. The number n_A can be varied to vary the security level of the system, as we shall show later. Alice publishes x_{nA} as her public key.

Step 3: In a similar fashion, Bob secretly chooses two seeds s_{B1} and s_{B2} and a large number n_B to generate a bit sequence, k_{iB}, to use in conjunction with x_0 to generate his public key x_{nB}, which he publishes.

Step 4: Alice takes Bob's public key x_{nB} and, using it as the seed and taking the same sequence, k_{iA}, as generated earlier, performs another n_A iteration to obtain x_{nB+nA}.

Step 5: Similarly, Bob takes Alice's public key x_{nA} and, taking it as the seed and the same bit sequence, k_{iB}, as generated by him earlier, performs another n_B iteration to get x_{nA+nB}.

The important point to note is that since $f_1(x)$ and $f_2(x)$ are linear functions, i.e., they satisfy the condition $f_1 \circ f_2 = f_2 \circ f_1$, therefore the sequence of operation does not matter. So the resultant iterations of the two sets are equal, i.e., $x_{nB+nA} = x_{nA+nB}$. This value obtained can now be suitably mapped to a common key K, which can then be used to communicate over the insecure channel.

It should be observed here that the start value x_0 can be a vector, and, hence, the common key will also be a vector. An example of a linear function that operates on a vector (x_0) could be the fast Fourier transform (FFT).

The following example illustrates the algorithm. Let the two linear functions be $f_1(x) = \mathrm{FFT}(x)$ and $f_2(x) = 1.5x$. Let the start value $x_0 = [0.06\ 0.35\ 0.81\ 0.01\ 0.14]$. This is randomly chosen by one of the parties (Alice or Bob) and made public. Let the CCS, used by both Alice and Bob, be

$$F_1(x_1) = 4x_1(1 - x_1) \quad \text{and} \quad F_2(x_2) = 3.98x_2(1 - x_2). \tag{5}$$

Alice secretly picks a number $n_A = 10$, and Bob secretly picks a number $n_B = 12$. Here small values of n_A and n_B have been used to illustrate the point. Alice secretly picks and uses the seeds $\{0.83, 0.34\}$ for the CCS to generate her random bit stream

$$k_A = [\,1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1\,]. \tag{6}$$

The length of the bit stream is n_A. Here, the initial conditions (seeds) do not have to be communicated to the other party. Similarly, Bob secretly picks and uses the seeds $\{0.47, 0.61\}$ for the CCS to generate his random bit stream

$$k_B = [\,0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1\,]. \tag{7}$$

The length of the bit stream is n_B. Next, Alice operates the function $f_1(x) = \mathrm{FFT}(x)$ or $f_2(x) = 1.5x$ on x_0, depending on whether a "1" or a "0" is encountered in the bit stream, k_A. Alice then publishes the resulting output as her public key

$$x_{nA} = [\,36.63 \quad 87.89 \quad 6.26 \quad 514.60 \quad 223.31\,]. \tag{8}$$

The application of FFT an even number of times yields a real output. Similarly, Bob uses k_B and x_0 to generate his public key

$$x_{nB} = [\,82.44 \quad 197.77 \quad 14.10 \quad 1157.86 \quad 502.47\,]. \tag{9}$$

Now, Alice takes x_{nB} and uses k_A to generate the secret key

$$x_{nA+nB} = \left[\,52\,168.84 \quad 317\,968.63 \quad 732\,706.40 \quad 8920.06 \quad 125\,151.16\,\right], \tag{10}$$

which is identical to the key, x_{nB+nA}, obtained by Bob by using x_{nA} and k_B. Thus, Alice and Bob are able to exchange a secret key successfully.

We now generalize the case for the m-CS PRNG. In this case, we have m linear functions, $f_1(x), f_2(x), \ldots, f_m(x)$, and the corresponding selection function is defined as

$$h_m(x, k) = \begin{cases} f_1(x) & \text{if } k = 1, \\ f_2(x) & \text{if } k = 2, \\ \quad\vdots & \\ f_m(x) & \text{if } k = m. \end{cases} \tag{11}$$

We can then write the iterative form as

$$x_i = h_m(x_{i-1}, k_i). \tag{12}$$

The key exchange algorithm based on m-CS is described as follows.

Step 1: Alice and Bob, who wish to exchange the key, publicly agree on a common m-CS, m linear functions $f_1(x), f_2(x), \ldots, f_m(x)$ and a start value x_0.

Step 2: Alice secretly chooses m seeds $s_{A1}, s_{A2}, \ldots, s_{AL}$ and a large number n_A and uses them as inputs to an m-CS PRNG to generate a random sequence of numbers, k_{iA}, as given by Eq. (2). n_A is the length of the sequence generated. This sequence k_{iA} is provided as input to Eq. (12) along with x_0 to iterate and give resultant x_{nA}. Alice publishes x_{nA} as her public key.

Step 3: In a similar fashion, Bob secretly chooses m seeds $s_{B1}, s_{B2}, \ldots, s_{BL}$ and a large number n_B to generate a sequence of number, k_{iB}, to use in conjunction with x_0 to generate his public key x_{nB}, which he publishes.

Step 4: Alice takes Bob's public key x_{nB} and, using it as the seed and taking the same sequence, k_{iA}, as generated earlier, performs another n_A iteration to obtain x_{nB+nA}.

Step 5: Similarly, Bob takes Alice's public key x_{nA} and, taking it as the seed and the same sequence, k_{iB}, as generated by him earlier, performs another n_B iteration to get x_{nA+nB}.

It should be observed that the initial conditions for the m-CS are chosen locally by both the parties, as shown in steps 2 and 3 of the algorithm. There is no requirement of communicating the initial conditions. Thus the choice of the initial conditions does *not* complicate the implementation. The choice of different initial conditions changes only the random bit sequence being generated by the m-CS. Since the two parties do not have to communicate their initial conditions, the algorithm does not get affected by the problem of synchronization. The algorithm can be easily implemented in software as well as in hardware.

In order to break the system, one has to solve the Diffie-Hellman problem, i.e., finding x_{nA+nB} from x_{nA} and x_{nB}. If the linear functions $f_1(x), f_2(x), \ldots, f_m(x)$ are suitably chosen, one cannot easily guess the constituent operations that convert the seed x_0 to the public keys x_{nA} and x_{nB}. The only viable attack is the brute force attack, where the adversary has to try all the possible combinations of sequences. Let the value n_A and n_B be chosen in the range $[0, N]$. Let each of the linear functions $f_1(x)$, $f_2(x), \ldots, f_m(x)$ require on the order of P floating point operations to execute. Then, in order to establish the key (by Alice or Bob), it requires on the order of NP floating point operations. However, the adversary has to decide for every number in the sequence which of the linear functions to use. Therefore, the complexity to break the cryptosystem is on the order of $(NP)^m$. This technique provides three independent design parameters to fix the security of the cryptosystem: (i) The size of the key, N. (ii) The computational complexity of the linear functions, P. (iii) The number of linear functions, m.

In conclusion, we have proposed a novel technique for public key encryption using multiple chaotic systems. This is the first time chaotic systems are being used for public key encryption (asymmetric cryptography). The problem of synchronization between different chaotic systems has been circumvented. We have shown that the security of the proposed algorithm grows as $(NP)^m$, where N, P, and m are large numbers that can be chosen, independently, as parameters of the cryptosystem. The proposed algorithm is practical and can be easily implemented in software as well as in hardware.

[1] G. J. Simmons, ACM Comput. Surv. **11**, 305 (1979).
[2] W. Diffie and M. Hellman, IEEE Trans. Inf. Theory **22**, 644 (1976).
[3] T. Stojanovski and L. Kocarev, IEEE Trans. Circuits Syst. I **48**, 281 (2001).
[4] T. Stojanovski, J. Phil, and L. Kocarev, IEEE Trans. Circuits Syst. I **48**, 382 (2001).
[5] R. Matthews, Cryptologia **XIII**, 29 (1989).
[6] T. Habutsu, Y. Nishio, I. Sasase, and S. Mori, in *Advances in Cryptology—EuroCrypt '91* (Springer-Verlag, New York, 1991), p. 127.
[7] Z. Kotulski and J. Szczepanski, Ann. Phys. (Berlin) **6**, 381 (1997).
[8] R. Bose and A. Banerjee, in *Proceedings of the 7th International Conference of Advanced Computing and Communications (ADCOM'99)* (Tata McGraw-Hill, New Delhi, 1999), p. 318.
[9] G. Jakimoski and L. Kocarev, IEEE Trans. Circuits Syst. I **48**, 163 (2001).
[10] Z. Kotulski, J. Szczepanski, K. Grski, A. Paszkiewicz, and A. Zugaj, Int. J. Bifurcation Chaos Appl. Sci. Eng. **9**, 1121 (1999).
[11] M. S. Baptista, Phys. Lett. A **240**, 50 (1998).
[12] R. Bose, in *Proceedings of ICIAM 2003, Sydney, Australia* (Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003).
[13] J. Fridrich, Int. J. Bifurcation Chaos Appl. Sci. Eng. **8**, 1259 (1998).
[14] R. Brown and L. O. Chua, Int. J. Bifurcation Chaos Appl. Sci. Eng. **6**, 219 (1996).
[15] D. D. Wheeler, Cryptologia **XIII**, 243 (1989).
[16] E. Biham, in *Advances in Cryptology—EuroCrypt '91* (Ref. [6]), p. 532.
[17] H. Zhou and X. T. Ling, IEEE Trans. Circuits Syst. I **44**, 268 (1997).
[18] G. Alvarez, F. Montoya, M. Romera, and G. Pastor, Phys. Lett. A **276**, 191 (2000).
[19] S. Li, X. Mou, and Y. Cai, in *Proceedings of Progress in Cryptology: INDOCRYPT 2001*, Lecture Notes in Computer Science Vol. 2247 (Springer, Berlin, 2001), p. 316.
[20] S. Li, X. Zheng, X. Mou, and Y. Cai, Proc. SPIE-Int. Soc. Opt. Eng. **4666**, 149 (2002).
[21] L. Kocarev, IEEE Circuits Syst. Mag. **1**, 6 (2001).
[22] L. Y. Deg and H. Xu, ACM Trans. Model. Comput. Simul. **13**, 299 (2003).
[23] P. L'Ecuyer and R. Touzin, Stat. Comput. **14**, 5 (2004).
[24] U. M. Maurer and S. Wolf, Des. Codes Cryptogr. **19**, 147 (2000).
[25] R. L. Rivest, A. Shamir, and L. Adleman, Commun. ACM **21**, 120 (1978).
[26] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography* (Cambridge University Press, Cambridge, England, 1999).
[27] V. Miller, in *Advances in Cryptology: Proceedings of CRYPTO '85* (Springer-Verlag, Berlin, 1986), p. 417.
[28] N. Koblitz, Math. Comput. **48**, 203 (1987).