A NOVEL LOWER COST CRYPTO-SCHEME BASED ON THE THEORY OF SHARING SECRETS

Chao-Wen CHAN and Chin-Chen CHANG

Abstract: This article presents an information permutation and breaking scheme to construct a low-cost encryption scheme. The proposed encryption scheme preserves the security requirements of a general encryption scheme. The presented research uses information entropy to depict the one-way property and tries to use entropy to study the one-way property. The authors also present an estimation of the reasonable size of the seed set of a pseudo-random number generator.

Keywords: Entropy, One-Way Hash Function, One-Way Property, Pseudo-Random Number Generator.

Mobile computing continues to grow in importance. Machines that are designated to perform mobile computing are notebook PCs, PDAs, and mobile phones. In general, these machines have lower power than desktop PCs, and one of the main applications of these machines is communication. The owners of these machines communicate with each other or to some host machines. The business users of these lower power machines may need to exchange secret messages with other machines. The main technique to protect these secret messages is computer cryptography. The most popular cryptosystems are public-key cryptosystems.[1,2] However, encryption and decryption operations in these cryptosystems are expensive. It may take a long time to perform these operations. In addition, the length of the message that can be encrypted or decrypted is completely determined by the cryptosystem itself. However, the length of a general message is, in general, longer than the above mentioned length. The brute force method partitions the long message into blocks, so that each resulting block can be encrypted or decrypted using only one public-key operation. Thus, a message is partitioned into t blocks, then, we encrypt or decrypt it requiring t public-key operations. This will take a long time for a mobile machine.

A scheme to reduce the cost of the public-key encryption or decryption of a long message uses a session key of a symmetric-key cryptosystem to encrypt it. Then, we

use a public-key cryptosystem to encrypt the session key. Finally, we send the encrypted session key and the message encrypted by the session key to the receiver. After receiving the encrypted message, the receiver uses a public-key operation to decrypt the encrypted session key, then, uses the session key to decrypt the encrypted message. In the scheme, the symmetric-key cryptosystem[3] reduces the public-key operation of a long message to a symmetric session key which can be properly operated by a public-key operation. However, the symmetric-key encrypted message contains all the information found in the original message. So, the public-key cryptosystem and the symmetric-key cryptosystem must prevent all attacks to ensure the security of the scheme.

The proposed scheme uses a permutation-like breaking information scheme to partition a long message into t blocks. By the interpolating theory and Shamir's (t,n)-threshold scheme,[4] we try to reduce the encryption of a long message into the encryption of some small blocks. Moreover, we will reduce the security problem of the encryption of a long message to that of a single public-key operation.

The section that follows presents the main idea of the proposed scheme. Then, the authors summarize the security problems of the main components of the scheme. Details of the proposed scheme are presented afterwards, followed by a comprehensive analysis. Conclusions of the presented research are given in the last section.

Main Idea

In the previous section, the authors point out that their approach to the problem of reducing the cost of a public-key encryption of a long message is to use a permutation-like information breaking mechanism to re-permute the long message and partition it into several blocks, say t blocks, such that no one can recover the original message unless s/he has all t blocks. It is a well-known scheme, called (t,n)-threshold secret sharing scheme. Using a (t,t)-threshold scheme, one can distribute, or partition, a secret into t shadows such that any $t' < t$ shadows can not recover the original secret and t distinct shadows can recover it. Let the secret be denoted by s. In Shamir's (t,n)-threshold scheme,[5] s is placed at the constant coefficient position of a polynomial of degree $t-1$ with other coefficients being randomly selected. We use the resulting random polynomial to generate n distinct points. Then, we can use any t out of these n distinct points to reconstruct the random polynomial, the unique interpolating polynomial of these t distinct points, and recover s by retrieving the constant coefficient of it. It should be noted that any point set of size less than t can not uniquely recover the random polynomial. The distribution of the secret in Shamir's scheme is depicted diagrammatically below:

secret	s		
	$\downarrow$		
random polynomial	$f(x) = s + \sum_{i=1}^{t-1} a_i X^i$	$\leftarrow$	$\{a_i\}_{i=1}^{t-1}$
	$\downarrow$		
random distinct n points	$\{(x_i,\ f(x_i))\}_{i=0}^{n-1}$.	$\leftarrow$	$\{x_i\}_{i=0}^{n-1}$

With the property of modulo arithmetic operations, for instance $(\mathrm{mod}\, P)$ with P being prime, any change of a bit in s will be distributed over all bits of the resulting point set $\{(x_i,\ f(x_i))\}_{i=0}^{n-1}$. The recovering of the secret in Shamir's scheme is depicted diagrammatically below:

distinct t points	$\rightarrow$	$\{(x_{j_i},\ y_{j_i})\}_{i=0}^{t-1}$.	$\subset$	$\{(x_i,\ f(x_i))\}_{i=0}^{n-1}$.
		$\downarrow$		
		interpolating polynomial to these points		
		$\downarrow$		
		$f(X) = \sum_{i=1}^{t-1} a_i X^i$		
		$\downarrow$		
		$s = f(0)$		

It is obvious that the secret s can be placed at the position of any coefficient of the random polynomial without loss of any security condition. And, a plaintext message seems to be a random number in general. With this insight, we may treat a plaintext message as a random polynomial function, and use it to generate a set of distinct points by giving a set of random distinct abscissas, $\{x_i\}_{i=0}^{t-1}$. The following diagram can be used to depict the permutation procedure:

plaintext message	$(a_0, a_1, \cdots, a_{t-1}) \in Z_P^t$		
	$\downarrow$		
	$f(X) = \sum_{i=1}^{t-1} a_i X^i \in Z_P[X]$		
random abscissas	$\downarrow$	$\leftarrow$	$\{x_i\}_{i=0}^{t-1} \subset Z_P$
	$\{(x_i,\ f(x_i))\}_{i=0}^{t-1} \subset Z_P \times Z_P$		
	$\downarrow$		
ciphertext	$(f(x_0), f(x_1), \ldots, f(x_{t-1})) \in Z_P^t$		

It can be seen that any set of distinct abscissas, $\{x_i\}_{i=0}^{t-1} \subset Z_P$, uniquely determines an one-one linear transformation from Z_P^t to Z_P^t. Let denote the linear transformation by $F_{\vec{x}}$, then, the above permutation procedure can be written in the form of the following matrix equation:

$$\vec{y} \equiv F_{\vec{x}} \vec{a} \pmod{P}, \tag{1}$$

where $\vec{a}, \vec{y} \in Z_P^t$ and $y_j = \sum_{i=0}^{t-1} a_i x_j^i$. Note that $F_{\vec{x}}$ is a Vandermonde matrix. Let $\vec{A}$, and $\vec{Y}$ denote random vectors (or sequences of random variables). That is, $\vec{A} = (A_0, A_1, \ldots, A_{t-1})$, $\vec{Y} = (Y_0, Y_1, \ldots, Y_{t-1})$, and A_i, Y_i are random variables. Then Equation (1) implies

$$\mathrm{H}(\vec{Y} \mid \vec{A}, \vec{x}) = 0, \text{ or} \tag{2}$$

$$\mathrm{H}(\vec{A} \mid \vec{Y}, \vec{x}) = 0, \tag{3}$$

provided $\vec{x} = (x_0, x_1, \ldots, x_{t-1})$, ($x_i \neq x_j$ if $i \neq j$), where $H(|)$ denotes the conditional entropy function. It should be noted that if $\vec{a}$ happens to be an eigenvector of $F_{\vec{x}}$, then $\vec{y}$ is also an eigenvector of $F_{\vec{x}}$. In general, $\vec{a}$ is unlikely to be an eigenvector of $F_{\vec{x}}$, because $\vec{a}$ represents a plaintext message which can be seen as a randomly selected vector. In this situation, one can control the abscissa set $\{x_i\}_{i=0}^{t-1}$ to prevent it. Or, we will propose a scheme to make the probability of $\vec{a}$ being an eigenvector of $F_{\vec{x}}$ to be small. Given $\vec{x}$ and $\vec{y}$, one can determine whether $\vec{y}$ is an eigenvector of $F_{\vec{x}}$ or not. However, according to the theory of Shamir's threshold scheme, we have

$$H(A_i) = H(A_i \mid Y_1, \ldots, Y_{t-1}, \vec{X}). \tag{4}$$

That is, without the knowledge of Y_0, one cannot decide whether $\vec{Y}$ is an eigenvector of $F_{\vec{x}}$ or not.

In summary, we may permute a plaintext message by a linear transformation which is uniquely determined by randomly selecting a secret set of distinct abscissas. Without all the permuted information, one cannot learn any information about the original plaintext message. In the next section, we will briefly review the notions of pseudo-random number generator and a one-way hash function.

Related Topics

In this section, we will briefly review two important notions, one-way hash function and pseudo-random number generator. We shall use a one-way hash function to generate the message digest of a secret message and a pseudo-random number generator to generate a set of abscissas. Using a pseudo-random number generator by feeding a random number seed to generate a set of abscissas replaces the notion of the compression of a set of abscissas into a secret seed. A vector consists of the message digest and the chance that the original message is an eigenvector of a linear transformation will likely be decreased.

One-Way Hash Functions

Informally, a function $Y = f(X)$ has one-way property which means that given any $X = x$ in the domain of f, it is easy to compute a $Y = y$ in the co-domain of f such that $y = f(x)$, but it is hard to invert. That is, given a $Y = y$ in the co-domain of f, it is hard to compute a $X = x$ such that $y = f(x)$. To easily formulate the one-way property, we may assume that domain and co-domain of all functions are sets of binary strings $\{0,1\}^*$. Formally, a one-way function can be defined as follows:

Definition 1 One-way Function[6] *Let* $f : \{0,1\}^* \to \{0,1\}^*$ *be a function from binary strings to binary strings.* f *is called a* one-way function *if it satisfies the following conditions:*

1. f *is injective, and for all* $x \in \{0,1\}^*$, $|x|^{\frac{1}{k}} < |f(x)| < |x|^k$ *for some integer* $k > 0$, *where* $|x|$ *denotes the length of* x. *In other words,* $f(x)$ *is at most polynomially longer or shorter than* x.
2. f *is in* **FP**, *that is, given any* x, *the function value* $f(x)$ *can be computed in polynomial time.*
3. *The most important condition is that* f^{-1}, *the inverse of* f, *is not in* **FP**. *That is, there is no polynomial-time algorithm which, given a binary string y, either computes a* x *such that* $y = f(x)$ *or returns "no," if no* x *satisfies the equation* $y = f(x)$.

It can be noticed that, since a one-way function f is injective, x can be uniquely recovered from $f(x)$ by trying all x of appropriate length. The key point is that there is no polynomial-time algorithm that achieves this. Until now, no function can be proved to be a one-way function. However, there are two functions that many people

suspect are one-way functions. The first is the integer multiplication function and the second is the exponentiation function modulo a prime.

If $Y = f(X)$ is a one-way function, then $H(Y \mid X) = 0$ and $H(X \mid Y) = 0$, where $H(|)$ is the conditional entropy function. But, by the one-way property, the equation $H(X \mid Y) = 0$ provides no useful information to invert a one-way function.

A function is called a hash function if it can take a binary string of any length and produce a binary string of fixed length. That is, a hash function is of the following general form:

$$\mathtt{Hash}_m : \{0,1\}^* \rightarrow [0, m-1], \tag{5}$$

for some positive integer m and $[0, m-1]$ denotes the set of integers $\{0, 1, 2, \ldots, m-1\}$. It is obvious that $\mathtt{Hash}_m$ cannot be an injection for all possible integers m. In effect, $\mathtt{Hash}_m$ retrieves some attributes of its input and modifies or prunes them into a fixed length of $\lceil m \rceil$. Since we will use a hash function to generate a message digest in the proposed scheme, the hash function has to satisfy a security condition, which is called the collision-free property. It is impossible that a hash function satisfies the true collision-free properties. However, there are hash functions that can satisfy some pseudo-collision-free properties.

Definition 2 Weakly Collision-free Property[7] *A hash function* $\mathtt{Hash}_m$ *is* weakly collision-free *if, given* x, *there is no polynomial-time algorithm to compute a* $x' \neq x$ *such that* $\mathtt{Hash}_m(x') = \mathtt{Hash}_m(x)$.

Definition 3 Strong Collision-free Property[8] *A hash function* $\mathtt{Hash}_m$ *is* strong collision-free *if there is no polynomial-time algorithm to compute two* x *and* x' *such that* $x \neq x'$ *and* $\mathtt{Hash}_m(x) = \mathtt{Hash}_m(x')$.

Definition 4 One-way Hash Function[9] *A hash function* $\mathtt{Hash}_m$ *is* one-way *if, given a* y, *there is no polynomial-time algorithm to compute a* x *such that* $\mathtt{Hash}_m(x) = y$.

Note that a one-way hash function is not a one-way function. It has been proven that a strong collision-free hash function must be a one-way hash function. In this article, we will assume that if $Y = \mathtt{Hash}_m(X)$ is a one-way hash function, then

$$H(X) \geq H(X \mid Y) >> 0. \tag{6}$$

The above equation implies that directly computing X from Y might be of no use than directly trying out all possible appropriate X or directly guessing X.

Pseudo-Random Number Generators

A random number generator can generate a sequence of numbers, such that having observed the first $n-1$ generated numbers, we still cannot predict the n-th number to be generated by the generator. A random number generator can be illustrated by a sequence of random variables X_0, X_1, …, X_j, X_{j+1}, …, in which each random variable X_j obeys a probability distribution and they satisfy the following equation:

$$H(X_j) = H(X_j \mid X_0, X_1, \ldots, X_{j-1}). \tag{7}$$

That is, in a real random sequence, we learn no information about the future from the history of the random sequence. However, in a practical situation, we always assume that all X_j will obey the same probability distribution, $p(X)$, and assume that the history of such a random number sequence provides information about its sample space and probability distribution, i.e., $X_j = X$ for all j. Even in this ideal situation, Equation (7) still holds. Another problem of the real random number generators is that the generated random number sequence cannot be likely reproduced by the same random number generator. In our proposed scheme, we really need a pseudo-random number generator. A pseudo-random number generator can generate a set of number sequences indexed by a set of binary strings called random number seeds. Thus, the general functional form of a pseudo-random number generator is as follows:

$$\mathrm{PRNG}_m^{n_O} : Z_{n_O} \to Z_m^N, \tag{8}$$

where m and n_O are positive integers. The parameter m denotes the sample space, Z_m, of the pseudo-random sequences generated by $\mathrm{PRNG}_m^{n_O}$, and the parameter n_O depicts the set of the seeds, Z_{n_O}. Let $\mathrm{PRNG}_m^{n_O}(i) = x_{i,0}, x_{i,1}, \ldots, x_{i,j}, x_{i,j+1}, \ldots$. Suppose that the elements in $\mathrm{PRNG}_m^{n_O}[Z_{n_O}]$ are arranged such that

$$\frac{\sum_{(i \in Z_{n_O}) \wedge (x_{i,j} = x)} I}{n_O} \approx p(x) \text{ for each } j, x \text{ and} \tag{9}$$

$$\frac{\sum_{(0 \le j < l) \wedge (x_{i,j} = x)} I}{l} \to p(x) \text{ for each } i, x \text{ when } l \to \infty. \tag{10}$$

In Equation (9) PRNG_m^n is required to transform the uniformly distributed random variable I, the seed, into a random variable X'_j with probability distribution $p(X)$ for each j. We may use the random variable X'_j to simulate the random variable $X_j = X$. Now, we rewrite (8) as an equation of random variables.

$$\mathrm{PRNG}_m^{n_O}(I) = X'_0, X'_1, X'_2, \ldots, X'_j, X'_{j+1}, \ldots. \tag{11}$$

Based on the above equation and real pseudo-random number generators, we may reasonably assume that:

$$H(X'_0, X'_1, \cdots, | I) = 0, \tag{12}$$

$$H(X'_j) \geq H(X'_j \mid X'_0, \cdots, X'_{j-1}) >> 0, \text{ and} \tag{13}$$

$$H(I) \geq H(I \mid X'_0, \cdots, X'_j) >> 0, \tag{14}$$

for all $j < j_O$, where j_O is an integer description of the one-way property of $\mathrm{PRNG}_m^{n_O}$. The above assumptions state that a pseudo-random number generator will be assumed to behave like a one-way function if the corresponding j_O is sufficiently large. Now, we will discuss the parameter n_O of $\mathrm{PRNG}_m^{n_O}$. n_O is the size of the set of seeds or the size of $\mathrm{PRNG}_m^{n_O}[Z_{n_O}]$. By assumption $X'_0, X'_1, X'_2, \ldots$ are independent, identically distributed $\sim p(X)$, and we have

$$-\frac{1}{j} \lg p(X'_0, X'_1, \cdots, X'_j) \to H(X) \tag{15}$$

in probability.

The equation is the so-called Asymptotic Equipartition Property (AEP). In the following, we list a definition related to AEP[10]:

Definition 5 Typical Set *The typical set* $T_\varepsilon^{(j)}$ *with respect to* $p(X)$ *is the set of sequences* $(x_0, x_1, \ldots, x_{j-1}) \in (Z_m)^j$ *with the property:*

$$2^{-j(H(X)+\varepsilon)} \leq p(x_0, x_1, \ldots, x_{j-1}) \leq 2^{-j(H(X)-\varepsilon)}. \tag{16}$$

We can see that if $(x_0, x_1, \ldots, x_{j-1}) \in T_\varepsilon^{(j)}$ then

$$H(X)-\varepsilon \le -\frac{1}{j}\lg p(x_0,x_1,\cdots,x_{j-1}) \le \mathrm{H(X)}+\varepsilon. \qquad (17)$$

Obviously, we require that $\mathrm{PRNG}_m^{n_O}[Z_{n_O}] \subset T_\varepsilon^{(j_O)}$ for some reasonable small ε. However, the size of $T_\varepsilon^{(j_O)}$ is smaller than $2^{j_O H(X)+\varepsilon}$. Suppose that $\mathrm{PRNG}_m^{n_O}$ is injection. Then, we have:

$$n_O = |\,\mathrm{PRNG}_m^{n_O}[Z_{n_O}]\,| \le |\,T_\varepsilon^{(j_O)}\,| \le 2^{j_O H(X)+\varepsilon}. \qquad (18)$$

Thus, we may select n_O as close to $2^{j_O H(X)}$ as possible.

In the scheme proposed in this article, we will use a pseudo-random number generator to compress a set of secret abscissas, or as a secret abscissas generator. Suppose that $\mathrm{PRNG}(X,Q,N)$, $Q<\phi(N)$, is a random number generator that can generate a $(\phi(N),Q)$-combination sequence x_0, x_1, …, x_{Q-1}, where x_i's are distinct and $x_i \in Z_N^*$. $\mathrm{PRNG}(X,Q,N)$ can be implemented by selecting first Q items of $\mathrm{PRNG}_N^{n_O}(X)$ that are prime to N. Note that $|\mathrm{PRNG}(Z_{n_O},Q,N)| \le \binom{\phi(N)}{Q}$. Using Pascal's triangle, we may require $Q=\frac{\phi(N)}{2}$ or $Q\approx\frac{\phi(N)}{2}$ to make $|\mathrm{PRNG}(Z_{n_O},Q,N)|$ as large as possible. We call a pseudo-random number generator of the form $\mathrm{PRNG}(X,Q,N)$ secure if it satisfies the above-mentioned conditions.

In the next section, we will propose a low cost encryption scheme based on the notions elaborated here.

Proposed Scheme

To present the proposed scheme, we will first list the assumptions that will be used in the scheme.

1. Let P be a public large prime number.
2. Let $\mathrm{PRNG}(X,Q,N)$ be a public secure pseudo-random number generator. In addition, for each positive integer pair (t,n), $t<\phi(n)$, $\mathrm{PRNG}(x,t,n)$ will generate a $(\phi(n),t)$-combination integer sequence x_0, x_1, …, x_{t-1}, randomly selected from Z_n^* given a random seed $X=x$.

3. Suppose that the public-private key pairs of Alice, the sender, and Bob, the receiver, are (e_A, d_A) and (e_B, d_B), respectively. In addition, $E_e()$ and $D_d()$ are the associated encryption and decryption algorithms.

4. Let $\mathtt{Hash}_N(X, M)$ be a public secure one-way hash function that is unlikely to have occurred collisions, such that for all x, n and a positive integer m, $\mathtt{Hash}_n(x, m) \in Z_n$.

5. The sender, Alice, maintains a table to record the used hash values. The length of the table is at most L, where L reflects the security degree of the one-way hash function $\mathtt{Hash}_N(X, M)$.

Based on the above assumptions, we have the following properties:

1. By the assumption of $\mathtt{PRNG}(X, Q, N)$, we have $H(X) \geq H(X \mid X_0, X_1, \ldots, X_t, Q, N) >> 0$, where X denotes the random variable of random seed, the random variable vector $(X_0, X_1, \ldots, X_t) = \mathtt{PRNG}(X, t, N)$, $H()$ denotes the entropy function and $H(|)$ denotes the conditional entropy function. Or, given $N = P$ and $Q = t$, we have $H(X) \geq H(X \mid X_0, X_1, \ldots, X_t) >> 0$. However, by the assumption of PRNG() being secure, we have $H(X) \sim H(X \mid X_0, X_1, \ldots, X_t)$, where ~ denotes the "almost equal to" symbol.

2. Suppose that $Y = E_e(X)$ and $X = D_d(Y)$, where X denotes the random variable of plain-text, Y the random variable of cipher-text, e the random variable of the public key, and d the random variable of private key. We have $H(d) = H(d \mid e)$, $H(Y \mid X, e) = 0$ and $H(X \mid Y, d) = 0$.

3. Suppose that $Y = \mathtt{Hash}_N(X, M)$. In general, we have $H(Y \mid X, N, M) = 0$. But, by the assumption of $\mathtt{Hash}_N()$ being secure, we will have $H(X \mid Y, N, M) >> 0$ where >> denotes the "much greater than" symbol. Or, given $N = P$ and $M = m$, we have $H(X \mid Y) >> 0$.

Below, we first present the proposed encryption scheme, then, the proposed decryption scheme.

Proposed Encryption Sub-scheme

Suppose that Alice wants to send the secret message $(a_1, a_2, \ldots, a_t) \in Z_P^t$ to Bob secretly. Then, she performs the steps outlined in Algorithm 1.

Algorithm 1 Proposed Encryption Scheme

Input: The large prime number P, the pseudo random number generator $\mathrm{PRNG}(X,T,N)$, the hash function $\mathrm{Hash}_N(X,M)$, the public-key encryption algorithm $E_e(X)$, the decryption algorithm $D_d(X)$, the private key d_A, the public key e_B, the message $(a_1,a_2,\ldots,a_t)\in Z_P^t$, and a published m.

Output: The encrypted message $(y_0,y_1,\ldots,y_t)$ authenticated by the private key owner which is to be sent to the public key owner.

Step 1. [Compute the Message Digest] Compute the message digest $a_0=\mathrm{Hash}_P(a_1||a_2||\ldots||a_t,m)$, where the expression $a_1||a_2||\ldots||a_t$ denotes the catenation of a_1, a_2, …, a_{t-1} and a_t. (If a_0 had been used, Alice needs to modify the message $(a_1,a_2,\ldots,a_t)$ to compute a new a_0 that was never used. Alice, then, appends a_0 to a history table. If the length of the history table is bigger than L, Alice needs to publish a new m value, construct a new history table of length 0, and repeat the step again.)

Step 2. [Construct a Secret Polynomial] Use the secret message $(a_0,a_1,a_2,\ldots,a_t)$ to construct the secret polynomial $f(X)=a_0+a_1X+\cdots+a_tX^t \pmod P$.

Step 3. [Generate a Secret Random $(P-1,\ t+1)$-Combination Sequence] Randomly select a random seed x and generate a random sequence $\mathrm{PRNG}(x,t+1,P)=(\mathrm{x}_0,x_1,\ldots,x_t)$.

Step 4. [Generate Encrypted Message] Compute the encrypted message $[\,f(x_0),\ f(x_1),\ \ldots,\ f(x_t)\,]$.

Step 5. [Generate Encrypted Random Seed with $f(x_0)$] Use the private key d_A and the public key e_B to compute $E_{e_B}(D_{d_A}(x||f(x_0)))$.

Step 6. [The Resulting Encrypted Message] $(y_0,y_1,\ldots,y_t)=[\,E_{e_B}(D_{d_A}(x||f(x_0))),\ f(x_1),\ f(x_2),\ldots,\ f(x_t)\,]$.

Note that after performing Algorithm 1, the message $(a_1,a_2,\ldots,a_t)$ was first signed with the private key d_A owned by Alice. The authentication and integrity of the message can be verified by the public key e_B of the owner, Bob, using his private

key, d_B and Alice's public key e_A. Alice then sends the resulting encrypted message $(y_0, y_1, \ldots, y_t)$ to the receiver, Bob, over an insecure channel.

Proposed Decryption Sub-scheme

Upon receiving $(y_0, y_1, \ldots, y_t)$, Bob can use Algorithm 2 to decrypt the message. Algorithm 2 uses the public key e_A and the private key d_B to verify the encrypted message, i.e., that it was sent from the public key owner and the receiver is the private key owner. Algorithm 2 also uses the one-way hash function $\texttt{Hash}_N(X, M)$ to verify the integrity of the secret message $(a_1, a_2, \ldots, a_t)$.

In the next section, the authors will analyze the security of the proposed scheme.

Algorithm 2 Proposed Decryption Scheme

Input: The large prime number P, the pseudo random number generator $\texttt{PRNG}(X, T, N)$, the hash function $\texttt{Hash}_N(X, M)$, the public-key encryption algorithm $E_e(X)$, the decryption algorithm $D_d(X)$, the private key d_B, the public key e_A, and the encrypted message $(y_0, y_1, \ldots, y_t)$.

Output: Reject the encrypted message or accept the decrypted message $(a_1, a_2, \ldots, a_t) \in Z_P^t$ whose authentication, sent from the public key owner, and integrity have been verified by the public key owner.

Step 1. [Recover the Secret Seed] Use the public key e_A and the private key d_B to decrypt the random seed $x \,||\, f(x_0) = E_{e_A}(D_{d_B}(E_{e_B}(D_{d_A}(x \,||\, f(x_0)))))$.

Step 2. [Recover the $(P-1,\ t+1)$-Combination Sequence] Use the seed x to generate the $(P-1,\ t+1)$-combination sequence, $\texttt{PRNG}(x, P, t+1) = (x_0, x_1, x_2, \ldots, x_t)$.

Step 3. [Recover the Secret Message] Compute the interpolating polynomial $f(x) \in Z_P[X]$ from the point sets $\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_t, f(x_t))\} \subset Z_P \times Z_P$. The coefficients of the polynomial $f(x) \in Z_P[X]$ would be the secret message $(a_0, a_1, \ldots, a_t)$.

Step 4. [Verify the Message Digest] Check whether $a_0 = \texttt{Hash}_P(a_1 \,||\, a_2 \,||\, \ldots \,||\, a_t, m)$ or not. If the equality does not hold, reject the message. If it passes the verification, accept the message as being sent from the public key e_A owner and its integrity has not been destroyed.

Analysis

The attacks to the proposed scheme can be classified into the following classes:

1. Type 1 attack is eavesdropping or passive wiretapping. In general, the attack cannot be detected. This is the reason why we construct a cryptosystem to encrypt sensitive or secret message for preventing eavesdropping. In this attack we assume that the sender, Alice, and the receiver, Bob, are trusted parties. The eavesdropper, Eve, wants to learn the secret message from the encrypted message.
2. Type 2 attack is tampering or active wiretapping. The attacker, Mallory, modifies the encrypted message or forges an encrypted message to fool either Alice or Bob or the two of them.
3. Type 3 attack is an attack coming from the receiver. Bob forges a message and claims that it has been sent by Alice.
4. Type 4 attack is an attack coming from the sender. Alice wants to deny a message which she has sent to Bob.

It has to be noted that if the proposed scheme cannot prevent type 2 attacks, Alice can use the weak point to deny having sent a sensitive message and Bob can claim that he has received a message that Alice has never sent. However, types 3 and 4 attacks are more powerful than type 2 attacks because Alice and Bob have more key information than Mallory.

Below, we restate the assumptions listed in the previous section.

Let $(a_1, a_2, \ldots, a_t)$ represents the random vector of plain-text to be encrypted, and $(y_0, y_1, \ldots, y_t)$ - the random vector of the resulting encrypted cipher-text. Let x represents the random variable of the random number seed chosen by Alice, $(\mathrm{x}_0, x_1, x_2, \ldots, x_t) = \mathtt{PRNG}(x, t+1, P)$. And, let (e, d) denote the random vector that represents the public-private key pair. In addition, let $y_{x,0} = D_{d_A}(x \,||\, f(x_0))$ and $y_0 = E_{e_B}(y_{x,0})$.

1. $H(a_1', a_2', \ldots, a_t' \mid a_0, a_1, \ldots, a_t) \gg 0$, where $H(a_0 \mid a_1, a_2, \ldots, a_t) = 0$ (i.e., $a_0 = \mathtt{Hash}_P(a_1 \,||\, a_2 \,||\ldots||\, a_t, m)$) and $(a_1', a_2', \ldots, a_t')$ is the random vector, distinct from $(a_1, a_2, \ldots, a_t)$, such that $a_0 = \mathtt{Hash}_P(a_1' \,||\, a_2' \,||\ldots||\, a_t', m)$.
2. $H(x \mid x_0, x_1, \ldots, x_t, t+1, P) \gg 0$, where $H(x_0, x_1, \ldots, x_t \mid x, t+1, P) = 0$.
3. $H(e \mid d) = 0$ and $H(d \mid e) \gg 0$.

4. $H(y_{x,0} \mid (x \| f(x_0))) >> 0$, where $H(y_{x,0} \mid (x \| f(x_0)), d_A) = 0$, and $H(y_{x,0} \mid y_0) >> 0$, where $H(y_{x,0} \mid y_0, d_B) = 0$.

Type 1 Attack to the Proposed Scheme

The type 1 attack is a ciphertext-only attack. The proposed scheme uses a random seed x to generate a linear transformation on Z_P^{t+1}, and the matrix representation of the linear transformation is the so-called Vandermonde matrix, such as the following:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^t \\ 1 & x_1 & x_1^2 & \cdots & x_1^t \\ & & \vdots & & \\ 1 & x_t & x_t^2 & \cdots & x_t^t \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_t \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_t) \end{pmatrix}. \tag{19}$$

According to the assumption about the pseudo-random number generator $\mathtt{PRNG}(X, Q, N)$, the $(P-1, t+1)$-combination integer sequence $(x_0, x_1, \ldots, x_t)$ generated by $\mathtt{PRNG}(x, t+1, P)$ is a set of distinct integers, i.e. $x_i \neq x_j \in Z_P$ whenever $i \neq j$. Thus, the corresponding Vandermonde matrix is nonsingular. That is, given a random seed x and $(f(x_0), f(x_1), \ldots, f(x_t))$, Eve can determine a unique $(a_0, a_1, \ldots, a_t)$. Thus, we have

$$H(\vec{f}_x \mid \vec{a}', \vec{x}) = 0, \text{ and} \tag{20}$$

$$H(\vec{f}_x \mid \vec{x}) = H(\vec{a}' \mid \vec{x}), \tag{21}$$

where $\vec{f}_x = (f(x_0), f(x_1), \ldots, f(x_t))$, $\vec{a}' = (a_0, a_1, \ldots, a_t)$, and $\vec{x} = (x_0, x_1, \ldots, x_t)$. However, the secret random seed x and $f(x_0)$ are encrypted by the public-key e_B, and only Bob can decrypt it and no one else can do that providing the public key encryption cryptosystem is secure. Without x and $f(x_0)$, Eve cannot learn any information about the secret message $(a_0, a_1, \ldots, a_t)$. According to the theory of interpolating polynomials, we have

$$H(\vec{a}', \vec{x}) = H(\vec{a}', \vec{x} \mid \vec{y}'), \text{ or} \tag{22}$$

$$H(\vec{a}') = H(\vec{a}' \mid \vec{x}, \vec{y}'), \tag{23}$$

where $\overrightarrow{y'} = (y_1, y_2, \ldots, y_t) = (f(x_1), f(x_2), \ldots, f(x_t))$. Suppose that Alice uses the random seed x repeatedly. Then, Eve may have learnt the $(P-1, t+1)$-combination sequence $(x_0, x_1, \ldots, x_t)$. However, according to the theory of interpolating polynomials, without $f(x_0)$ Eve cannot uniquely determine the polynomial function $f(X)$. According to the theory of Shamir's (t, n)-threshold secret sharing scheme, the secret message $(a_0, a_1, \ldots, a_t)$ is protected by a perfect secret sharing scheme providing the public-key cryptosystem is secure. That is, Eve has to solve the following problem to perform the attack.

$$\begin{aligned} &\text{Given } y_0, \\ &\text{compute } x \,||\, f(x_0) \text{ such that} \\ &y_0 = E_{e_B}(D_{d_A}(x \,||\, f(x_0))). \end{aligned} \tag{24}$$

Not considering the signing operation, suppose that Alice uses a public-key cryptosystem to encrypt the secret message $(a_1, a_2, \ldots, a_t)$. It will need t encryption operations to perform the encryption of the message. In the proposed scheme, we use a pseudo-random number generator operation and a $(t+1)\times(t+1)$-linear transformation operation to replace the $t-1$ public-key operations. (Because the complexity of the public-key encryption operation is the same as the complexity of the public-key decryption operation, we call either one of them a public-key operation.) In general, a public-key operation is costly. So, the proposed scheme is less costly. For the same reason, the decryption operation of the proposed scheme uses a pseudo-random number generator operation and an interpolating polynomial operation to replace the $t-1$ public-key operations.

In conclusion, given a secure public-key cryptosystem and not considering the signing operation the proposed scheme can prevent the ciphertext-only attacks; it uses two pseudo-random number generator operations, a linear transformation operation and an interpolating polynomial operation to reduce the number of required public-key operations to two.

Type 2 Attack to the Proposed Scheme

If Mallory intercepts an encrypted message—simply replaces one y_i with y_i', and inserts it back into the message—Bob will discover the attack in Step 4 of Algorithm 2. To make such an attack meaningful, she has to perform an attack of type 1 to the encrypted message. However, without the help of Alice or Bob, she cannot compute the secret message $(a_0, a_1, \ldots, a_t)$ providing the public-key cryptosystem is secure. For this reason, we consider type 2 attack, a known-plaintext

attack, to the proposed scheme. In addition, Mallory cannot learn the sequence $(x_0, x_1, \ldots, x_t)$ from the two sequences $(a_0, a_1, \ldots, a_t)$ and $(y_0, y_1, \ldots, y_t)$ unless Alice repeatedly uses the same secret seed x. It is unlikely to happen. However, we assume that Mallory knows the three sequences $(x_0, x_1, \ldots, x_t)$, $(a_0, a_1, \ldots, a_t)$ and $(y_0, y_1, \ldots, y_t)$. And, she wants to change a_i to a_i', where $0 < i \le t$.

After computing the digest message $a_0' = \mathtt{Hash}_P(a_1 || \ldots || a_i' || \ldots || a_t, m)$, Mallory, then, computes:

$$\begin{pmatrix} f'(x_0) \\ f'(x_1) \\ \vdots \\ f'(x_t) \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^t \\ 1 & x_1 & x_1^2 & \cdots & x_1^t \\ & & \vdots & & \\ 1 & x_t & x_t^2 & \cdots & x_t^t \end{pmatrix} \begin{pmatrix} a_0' \\ a_1 \\ \vdots \\ a_i' \\ \vdots \\ a_t \end{pmatrix}, \tag{25}$$

where $f'(X) = a_0' + a_1 X + a_2 X^2 + \cdots + a_i' X^i + \cdots + a_t X^t$. Now, Mallory's attack is reduced to computing y_0' using the following equation:

$$y_0' = E_{e_B}(D_{d_A}(x || f'(x_0))). \tag{26}$$

By the assumption about the security of the pseudo random number generator, Mallory cannot use the sequence $(x_0, x_1, \ldots, x_t)$ to learn the secret random seed x. Even though she uses a new random seed x' to compute a new sequence, ($f''(x_0')$, …, $f''(x_t')$), she still needs the private key d_A to compute a new $y_0'' = E_{e_B}(D_{d_A}(x' || f''(x_0')))$. Mallory uses the following equation to compute $f(x_0)$ for learning the secret seed x.

$$\begin{pmatrix} f(x_0) \\ y_1 \\ \vdots \\ y_t \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^t \\ 1 & x_1 & x_1^2 & \cdots & x_1^t \\ & & \vdots & & \\ 1 & x_t & x_t^2 & \cdots & x_t^t \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_t \end{pmatrix}. \tag{27}$$

Now, Mallory will obtain the following equation:

$$y_0 = E_{e_B}(D_{d_A}(x || f(x_0))). \tag{28}$$

From the above equation, without Bob's help, Mallory cannot learn the secret random seed x either. Suppose that Mallory and Bob want to fool Alice. However, without the private key d_A, Mallory cannot compute y'_0. Thus, Mallory and Bob cannot fool Alice.

In conclusion, if the public-key cryptosystem is secure, only Alice can compute the value y_0; no one else can do that. Thus, the proposed scheme uses two public-key operations to prevent type 2 attacks. All the published information e_A, e_B, and $(y_0, y_1, \ldots, y_t)$ does not provide useful information to accomplish type 2 attacks even with the knowledge about the repeated uses of the same random seed x.

Type 3 Attack to the Proposed Scheme

Bob can perform all attacks that Mallory can perform. In addition, Bob has an additional piece of information, the private key d_B. Using this private key, Bob can easily get the secret random seed x. So, we suppose that Bob wants to forge a message $(a'_1, a'_2, \ldots, a'_t)$, then claim that it was sent from Alice. Bob may use the hash function $\mathtt{Hash}_N(X, M)$ to compute $a'_0 = \mathtt{Hash}_P(a'_1 || \ldots || a'_t, m)$. He randomly selects a seed x' and generates a $(P-1, t+1)$-combination sequence $(x'_0, x'_1, \ldots, x'_t)$. Then, he computes

$$\begin{pmatrix} f'(x'_0) \\ f'(x'_1) \\ \vdots \\ f'(x'_t) \end{pmatrix} = \begin{pmatrix} 1 & x'_0 & x'^2_0 & \cdots & x'^t_0 \\ 1 & x'_1 & x'^2_1 & \cdots & x'^t_1 \\ & & \vdots & & \\ 1 & x'_t & x'^2_t & \cdots & x'^t_t \end{pmatrix} \begin{pmatrix} a'_0 \\ a'_1 \\ \vdots \\ a'_t \end{pmatrix}, \tag{29}$$

where $f'(X) = a'_0 + a'_1 X + a'_2 X^2 + \cdots + a'_t X^t$. But, Bob does not have the private key d_A; he, therefore, cannot directly use the formula $y'_0 = E_{e_B}(D_{d_A}(x' || f'(x'_0)))$ to compute y'_0. Consider the following equation,

$$x' || f'(x'_0) = E_{e_A}(D_{d_B}(y'_0)). \tag{30}$$

Suppose that $\mathtt{Proj}_i(\mathtt{PRNG}(X, N)) = X_i$ is the *i*-th projection function. Equation (30) can be rewritten as:

$$x' || f'(\mathtt{Proj}_0(\mathtt{PRNG}(x', P))) = E_{e_A}(D_{d_B}(y'_0)). \tag{31}$$

Table 1: History of Messages that Alice has Sent.

$a_{0,i}$	$\bar{x}_i$	$\bar{x}_{0,i}$	$y_{0,i}$
$a_{0,1}$	$\bar{x}_1$	$\bar{x}_{0,1}$	$y_{0,1}$
$a_{0,2}$	$\bar{x}_2$	$\bar{x}_{0,2}$	$y_{0,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{0,ml}$	$\bar{x}_{ml}$	$\bar{x}_{0,ml}$	$y_{0,ml}$

Thus, Bob's attack is reduced to solving the following simultaneous equations:

$$X \,||\, f'(\texttt{Proj}_0(\texttt{PRNG}(X,P))) = E_{e_A}(D_{d_B}(Y_0)), \tag{32}$$

$$f'(\texttt{Proj}_0(\texttt{PRNG}(X,P))) = \sum_{i=0}^{t} a'_i(\texttt{Proj}_0(\texttt{PRNG}(X,P)))^i \ (\text{mod P}) \tag{33}$$

$$a'_0 = \texttt{Hash}_P(a'_1 \,||\ldots||\, a'_t, m)\cdot b \tag{34}$$

For any $(a'_1, a'_2, \ldots, a'_t) \in Z_P^t$ and any random seed x', there should exist a solution (x', y'_0) of the above simultaneous equations since Alice can compute such a solution. However, given y'_0, Equation (32) uniquely determines a x'. But x' does not necessarily satisfy Equation (33). Note that if (x', y'_0) is a solution, the computation of y'_0 from x'_0 requires a one-way function $\texttt{PRNG}(X,T,N)$. Under our assumptions, Bob has to exhaustively search for y'_0 to compute x' such that it satisfies Equation (33). Suppose that Bob controls the value of a'_0 in order to find a proper sequence $(a'_1, a'_2, \ldots, a'_t)$ that satisfies Equation (33). However, in order to do that, Bob needs to break the secure one-way hash function $\texttt{Hash}_N(X,M)$. Now, we add one more assumption to increase Bob's abilities. Bob has collected all messages that Alice had sent to him and constructed Table 1, where $(a_{1,i}, a_{2,i}, \ldots, a_{t,i})$ is the i-th message that Alice has sent,

$$a_{0,i} = \texttt{Hash}_P(a_{1,i} \,||\, a_{2,i} \,||\ldots||\, a_{t,i}, m),) \tag{35}$$

$$f_i(X) = \sum_{j=0}^{t} a_{i,j} X^j, \tag{36}$$

$\bar{x}_i$ is the i-th random seed, $\bar{x}_{0,i}$ is the first random number generated by $\mathrm{PRNG}(\bar{x}_i, t+1, P)$, and

$$y_{0,i} = E_{e_B}(D_{d_A}(x^i \,||\, f_i(\bar{x}_{0,i}))). \tag{37}$$

It should be noted that the computation of $a_{0,i}$ is independent from the choice of the random seed $\bar{x}_i$. Thus, if ml is sufficiently large, there is a chance that Bob can change the value of $(a'_1, a'_2, \ldots, a'_t)$ such that a'_0 can be found in Table 1. Bob then selects the corresponding random seed $\bar{x}_i$ and uses the corresponding $y_{0,i}$ as y'_0. Alice cannot deny the resulting encrypted message $(y'_0, y'_1, \ldots, y'_t)$. However, Alice controls the length of the history table such that it is always smaller than L. So, we have $ml < L$. This means that ml can be too large and the new hash value will unlikely collide with the used hash values.

In conclusion, the proposed scheme can effectively prevent Bob's attack.

Type 4 Attack to the Proposed Scheme

Type 4 attack is performed by the sender, Alice. She wants to deny sending a secret message to Bob. If Bob can perform the type 3 attack, Alice can easily deny a message which she has sent. In a previous subsection, we have shown that without the help of Alice Bob is unlikely to perform a successful type 3 attack. Note that y_0 is protected by three one-way-like functions, $\mathrm{Hash}_N()$, $\mathrm{PRNG}()$, and $D_{d_A}()$, and only Alice can perform the public-key operation $D_{d_A}()$. Thus, the y_0 value of an encrypted message can be easily computed only by the sender, Alice. If Alice denies an encrypted message, Bob can publish the random seed x, y_0, and $D_{d_B}(y_0)$, ($f(x_0)$, $f(x_1)$, …, $f(x_t)$). Anyone can use the published information to prove that the encrypted message has been sent from Alice to Bob. Alice cannot deny it.

In summary, the proposed scheme provides undeniable services.

Conclusions

In this article, we use an information permutation scheme (a $(t+1, t+1)$-threshold scheme) to permute and break a message into a sequence of sub-blocks, such that without all sub-blocks we cannot recover the original message. The information breaking scheme is a polynomial-time operation. By encrypting a sub-block, we can protect the whole message. Thus, it is a low-cost scheme. In addition, we have shown that the proposed scheme preserves such cryptographic operations as information

authentication, information integrity, unforgeability, and undeniablility, while still maintaining a low cost.

First, the authors have presented the proposed information permutation and breaking scheme, and analyzed its security. The evidence for its efficiency is that it is a matrix multiplication modulo a prime. Next, the security requirements of one-way hash function and secure pseudo-random number generator have been given. An important contribution of the presented work is that the authors use information entropy to describe the notion of the one-way property that is believed to be a problem in computational complexity. This article has also established estimation of the size of the set of the random number seeds to a secure pseudo-random number generator that does not consider the algorithm that implements the generator.

Notes:

[1] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Communications of the ACM* 21, no. 2 (February 1978): 120–126.

[2] Certificateless Public Key Cryptography.

[3] Joan Daemen and Vinvent Rijmen, "Advance Encryption Standard," (Computer Security Division of the Information Technology Laboratory at the National Institute of Standards and Technology, U.S. Department of Commerce's Technology Administration, October 2000), <http://csrc.nist.gov/CryptoToolkit/aes> (3 December 2004).

[4] Adi Shamir, "How to Share a Secret," *Communications of the ACM* 22, no. 11 (November 1979): 612–613.

[5] Shamir, "How to Share a Secret."

[6] Christos H. Papadimitriou, *Computational Complexity* (Reading, Massachusetts, Menlo Park, California: Addison-Wesley Publishing Company, 1994).

[7] Douglas R. Stinson, *Cryptography: Theory and Practice* (Boca Raton, London, Tokyo: CRC Press Inc., 1995).

[8] Stinson, *Cryptography: Theory and Practice*.

[9] Stinson, *Cryptography: Theory and Practice*.

[10] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, (New York, Chichester, Brisbane, Toronto, Singapore: John Wiley & Sons, Inc., 1991).

CHAO-WEN CHAN was born in Hsinchu, Taiwan, Republic of China, on 30 December 1957. He obtained a B.S. degree in Mathematics from National Tsing Hua University, Hsinchu, Taiwan, in 1981 and a M.S. degree in Mathematics from National Tsing Hua University, Hsinchu, Taiwan, in 1983. He is currently a Ph.D. candidate in Computer Science and Information Engineering at the National Chung Cheng University. His research interests include Cryptography, Image Processing, and Information Security. *Address for correspondence:* Department of Computer Science and Information Engineering, National Chung Cheng University, 160, San-Hsing, Min-Hsiung, Chiayi 621, Taiwan. *Phone*: 886-5-2720411 ext. 33100; *Fax*: 886-5-2720859; *E-mail*: ccwen@cs.ccu.edu.tw.

CHIN-CHEN CHANG see page 88.