Improved Public Key Cryptosystems Secure against Chosen Ciphertext Attacks

Yuliang Zheng

The Centre for Computer Security Research
Department of Computer Science
The University of Wollongong
Wollongong, NSW 2522, AUSTRALIA
yuliang@cs.uow.edu.au

24 January 1994

Abstract

This short note describes an improvement to the first two of the three public key cryptosystems proposed by Zheng and Seberry, which are provably secure against chosen ciphertext attacks. The improvement removes a shortcoming with the original cryptosystems, which occurs when they are used for both confidentiality and sender authentication purposes.

1 Introduction

Zheng and Seberry presented in [ZS93] three practical public key cryptosystems that are secure against chosen ciphertext attacks. These cryptosystems are the first which feature both practicality and provable security against chosen ciphertext attacks, and have attracted notable attention from the research community [BR93, LL94]. In particular, main ideas in [ZS93] have been further developed by Bellare and Rogaway to design practical digital signature schemes and zero-knowledge protocols in [BR93].

In addition to their practicality and provable security, another attractive feature of the cryptosystems is that they can be employed in such applications as electronic mail systems and financial transactions, where both confidentiality and sender authenticity of messages are required. However, as was pointed out by Lim and Lee [LL94], when the first two cryptosystems are used for checking messages' origin, their sender authentication capability might be compromised by an "inside" enemy who can obtain pairs of message and ciphertext.

The main cause for this potential problem lies in the fact that during the enciphering process, a sender's secret information does not participate in the creation of a tag. As was noticed in [LL94], the shortcoming can be easily removed by letting the seed to the cryptographically strong pseudo-random number generator G be part of the input to the one-way hash function h (or the universal hash function h_s). A similar technique has been employed in encryption algorithms presented in [BR93]. In the following we first introduce notations to be used, and then describe how to change the two cryptosystems so that they get rid of the potential problem. For completeness, the third cryptosystem proposed Zheng and Seberry is also included in the descriptions.

2 Notation

The cryptosystems are reminiscent of the Diffie-Hellman cryptosystem [DH76] and El Gamal cryptosystem [ElG85] in their use of a n-bit (public) prime p and a (public) generator g of the multiplicative group $GF(p)^*$ of the finite field $GF(p)$. Here n is a security parameter which is greater than 512 bits, while the prime p must be chosen such that $p-1$ has a large prime factor [LO91]. In this note the alphabet $\Sigma = \{0, 1\}$ will be employed, and $|x|$ denotes length of a string x over Σ. Concatenation of string are denoted using the "$||$" symbol and the bit-wise XOR operations of two strings is symbolized using "$\oplus$". The notation $x_{[i \cdots j]}$ $(i \leq j)$ is used to indicate the substring obtained by taking the bits of string x from the i-th bit (x_i) to the j-th bit (x_j). The action of choosing an element x randomly and uniformly from a set S is denoted by $x \in_R S$. It should be noted that there is a natural one-to-one correspondence from strings in Σ^n to elements in the finite field $GF(2^n)$, and from strings in Σ^n to integers in $[0, 2^n - 1]$.

Other notations are as follows. G denotes a cryptographically strong pseudo-random string generator that stretches an n-bit input string into an output string whose length can be an arbitrary polynomial in n. Two examples of G follow. The first example is STRANDOM [Zhe93], a generator based on a one-way hash function called HAVAL [ZPS93]. Like its parent HAVAL, STRANDOM is very fast and extremely suitable for software implementation.

The second example is the generator based on the difficulty of computing discrete logarithms in finite fields [BM84, LW88, Per85]. This generator produces $O(\log n)$ bits output at each exponentiation. Compared with STRANDOM, this generator is much slower, although its security (or strength) can be formally proven under a reasonable assumption.

All messages to be encrypted are chosen from the set $\Sigma^{P(n)}$, where $P(n)$ is an arbitrary polynomial with $P(n) \geqq n$. Padding can be used for messages of length less than n bits. The polynomial $\ell = \ell(n)$ specifies the length of tags.

In the algorithms to be shown in the following, we assume that Bob (a sender) wants to send a $P(n)$-bit message m to Alice (a receiver) whose secret-public key pair is (x_A, y_A). Here Alice's secret key x_A is an element chosen randomly from $[1, p-1]$ and her public key is $y_A = g^{x_A} \bmod p$. Bob encrypts his messages using an enciphering algorithm, while Alice decrypts ciphertexts received using the corresponding deciphering algorithm. It is not necessary for Bob to have a pair of secret and public keys, unless sender authentication is required (see the following sections).

Following the notation in [ZS93], the first cryptosystem will be called $\mathcal{C}_{owh}$, the second $\mathcal{C}_{uhf}$ and the third $\mathcal{C}_{sig}$.

3 On $\mathcal{C}_{owh}$

This cryptosystem employs two cryptographic primitives, a cryptographically strong pseudo-random string generator denoted by G and a one-way hash function denoted by h. As discussed in Section 2, G expands a relatively short random input into a output of desired length, while guaranteeing the pseudo-randomness of the output string. In contrast, h compresses an arbitrarily long input string into an ℓ-bit output string. STRANDOM is an example of G [Zhe93], while HAVAL is an example of h [ZPS93].

3.1 Original $\mathcal{C}_{owh}$

Algorithm 1 $E_{owh}(y_A, p, g, m)$

```
    1. $x \in_R [1, p-1]$.
    2. $z = G(y_A^x \bmod p)_{[1 \cdots (P+\ell)]}$.
    3. $t = h(m)$.
    4. $c_1 = g^x \bmod p$.
    5. $c_2 = z \oplus (m \,||\, t)$.
    6. output $(c_1, c_2)$.
end
```

Algorithm 2 $D_{owh}(x_A, p, g, c_1, c_2)$

```
    1. $z' = G(c_1^{x_A} \bmod p)_{[1 \cdots (P+\ell)]}$.
    2. $w = z' \oplus c_2$.
    3. $m' = w_{[1 \cdots P]}$.
    4. $t' = w_{[(P+1) \cdots (P+\ell)]}$.
    5. if $h(m') = t'$ then
           output $(m')$
       else
           output $(\emptyset)$.
end
```

In the above description, the "$\emptyset$" symbol denotes the empty string.

Now suppose that Bob, the sender, also has a pair of secret and public keys denoted by (x_B, y_B), where x_B is chosen from $[1, p-1]$ uniformly at random and $y_B = g^{x_B} \bmod p$. Then the cryptosystem can be enhanced with a sender authentication capability by the following changes:

- add x_B to the input parameter list of E_{owh}, and y_B to the input parameter list of D_{owh}.
- change the step 2 of the enciphering algorithm to "2. $z = G(y_A^{x_B+x} \bmod p)_{[1 \cdots (P+\ell)]}$."
- change the step 1 of the deciphering algorithm to "1. $z' = G((y_B \cdot c_1)^{x_A} \bmod p)_{[1 \cdots (P+\ell)]}$."

The sender authentication capability guarantees that no third party can create a legal ciphertext between Alice and Bob. The capability, however, might be compromised by an attacker who can obtain "inside" information, namely, pairs of message and ciphertext between Alice and Bob. We look more closely at this insider attack. Assume that the attacker has a message-ciphertext pair $(m, (c_1, c_2))$. As m is known, $t = h(m)$ can be calculated. Thus the pseudo-random one-time pad z can be extracted. Now let $m^* \neq m$ is an arbitrary message. The attacker can calculate $t^* = h(m^*)$, and hence $c_2^* = z \oplus (m^* \mid\mid t^*)$. Clearly, (c_1, c_2^*) is a legal ciphertext between Alice and Bob. This shows that the attacker can impersonate either Alice or Bob.

It should be stressed that the above problem is associated only with authentication, and it does not have any influence on the security of the cryptosystem against chosen ciphertext attacks when it is used solely for message confidentiality.

3.2 Improved $\mathcal{C}_{owh}$

The potential impersonation problem shown above is remedied if the tag t depends also on the sender's secret information x_B, without knowing which the tag can not be created.

Algorithm 3 $E^*_{owh}(y_A, p, g, m)$

```
    1. $x \in_R [1, p-1]$.
    2. $r = y_A^x \bmod p$.
    3. $z = G(r)_{[1 \cdots (P+\ell)]}$.
    4. $t = h(m \mid\mid r)$.
    5. $c_1 = g^x \bmod p$.
    6. $c_2 = z \oplus (m \mid\mid t)$.
    7. output $(c_1, c_2)$.
end
```

Algorithm 4 $D^*_{owh}(x_A, p, g, c_1, c_2)$

```
    1. $r' = c_1^{x_A} \bmod p$.
    2. $z' = G(r')_{[1 \cdots (P+\ell)]}$.
    3. $w = z' \oplus c_2$.
    4. $m' = w_{[1 \cdots P]}$.
    5. $t' = w_{[(P+1) \cdots (P+\ell)]}$.
    6. if $h(m' \mid\mid r') = t'$ then
         output $(m')$
       else
         output $(\emptyset)$.
end
```

Note the order of computing $h(m \mid\mid r)$: the message m is compressed first, followed by the seed r.

The cryptosystem can be enhanced with a sender authentication capability, if Bob also has a pair of secret and public keys, (x_B, y_B). For the convenience of future reference, the enhanced cryptosystem is described in a separate section.

3.3 Improved $\mathcal{C}_{owh}$ with Sender Authentication

Algorithm 5 $E^*_{owh}(x_B, y_A, p, g, m)$

```
    1. $x \in_R [1, p-1]$.
    2. $r = y_A^{x_B + x} \bmod p$.
    3. $z = G(r)_{[1 \cdots (P+\ell)]}$.
    4. $t = h(m \,||\, r)$.
    5. $c_1 = g^x \bmod p$.
    6. $c_2 = z \oplus (m \,||\, t)$.
    7. output $(c_1, c_2)$.
end
```

Algorithm 6 $D^*_{owh}(x_A, y_B, p, g, c_1, c_2)$

```
    1. $r' = (y_B \cdot c_1)^{x_A} \bmod p$.
    2. $z' = G(r')_{[1 \cdots (P+\ell)]}$.
    3. $w = z' \oplus c_2$.
    4. $m' = w_{[1 \cdots P]}$.
    5. $t' = w_{[(P+1) \cdots (P+\ell)]}$.
    6. if $h(m' \,||\, r') = t'$ then
         output $(m')$
       else
         output $(\emptyset)$.
end
```

4 On $\mathcal{C}_{uhf}$

Like $\mathcal{C}_{owh}$, this cryptosystem uses a cryptographically strong pseudo-random string generator G. It also uses a universal class of hash functions [CW79, WC81]. An example of universal classes of hash functions is the collection of all k-variable affine functions on $GF(2^\ell)$. A function h_s in the class has the form of $h_s(x_1, x_2, \ldots, x_k) = \sum_{i=1}^{k} a_i x_i + b$, where $a_1, a_2, \ldots, a_k, b$ are elements from $GF(2^\ell)$, while $x_1, x_2, \ldots, x_k$ are variables in $GF(2^\ell)$. Note that a function in H compresses $k\ell$-bit input into ℓ-bit output strings, and itself is specified by $k+1$ elements from $GF(2^\ell)$ (i.e., $(k+1)\ell$

bits). By padding to input strings, functions in the class can compress strings whose lengths are not exactly $k\ell$.

$\mathcal{C}_{uhf}$ has the same potential problem when it is employed for authentication purposes, and the problem is removed in the same way as that for $\mathcal{C}_{owh}$.

4.1 Original $\mathcal{C}_{uhf}$

The original cryptosystem assumes a universal class of hash functions which map P-bit input into ℓ-bit output strings, where P is the length of messages. Each function h_s in the class is specified by a string of exactly Q_1 bits.

Algorithm 7 $E_{uhf}(y_A, p, g, m)$

```
    1. $x \in_R [1, p-1]$.
    2. $r = y_A^x \bmod p$.
    3. $z = G(r)_{[1 \cdots P]}$.
    4. $s = G(r)_{[(P+1) \cdots (P+Q_1)]}$.
    5. $c_1 = g^x \bmod p$.
    6. $c_2 = h_s(m)$.
    7. $c_3 = z \oplus m$.
    8. output $(c_1, c_2, c_3)$.
end
```

Algorithm 8 $D_{uhf}(x_A, p, g, c_1, c_2, c_3)$

```
    1. $r' = c_1^{x_A} \bmod p$.
    2. $z' = G(r')_{[1 \cdots P]}$.
    3. $s' = G(r')_{[(P+1) \cdots (P+Q_1)]}$.
    4. $m' = z' \oplus c_3$.
    5. if $h_{s'}(m') = c_2$ then
          output $(m')$
       else
          output $(\emptyset)$.
end
```

4.2 Improved $\mathcal{C}_{uhf}$

To improve the original cryptosystem, we assume a universal class of hash functions which map $P+n$-bit input into ℓ-bit output strings, where P is the length of messages and n is the length of the prime p. Each function h_s in the class is specified by a string of exactly Q_2 bits.

Algorithm 9 $E^*_{uhf}(y_A, p, g, m)$

```
    1. $x \in_R [1, p-1]$.
    2. $r = y_A^x \bmod p$.
    3. $z = G(r)_{[1 \cdots P]}$.
    4. $s = G(r)_{[(P+1) \cdots (P+Q_2)]}$.
    5. $c_1 = g^x \bmod p$.
    6. $c_2 = h_s(m \,||\, r)$.
    7. $c_3 = z \oplus m$.
    8. output $(c_1, c_2, c_3)$.
end
```

Algorithm 10 $D^*_{uhf}(x_A, p, g, c_1, c_2, c_3)$

```
    1. $r' = c_1^{x_A} \bmod p$.
    2. $z' = G(r')_{[1 \cdots P]}$.
    3. $s' = G(r')_{[(P+1) \cdots (P+Q_2)]}$.
    4. $m' = z' \oplus c_3$.
    5. if $h_{s'}(m') = c_2$ then
         output $(m' \,||\, r')$
       else
         output $(\emptyset)$.
end
```

4.3 Improved $\mathcal{C}_{uhf}$ with Sender Authentication

Like the improved cryptosystem $\mathcal{C}_{owh}$, Bob, the sender, is required to have a secret-public key pair (x_B, y_B), if sender authentication is to be achieved by $\mathcal{C}_{uhf}$.

Algorithm 11 $E^*_{uhf}(x_B, y_A, p, g, m)$

```
1. $x \in_R [1, p-1]$.
2. $r = y_A^{x_B+x} \bmod p$.
3. $z = G(r)_{[1 \cdots P]}$.
4. $s = G(r)_{[(P+1) \cdots (P+Q_2)]}$.
5. $c_1 = g^x \bmod p$.
6. $c_2 = h_s(m \mid\mid r)$.
7. $c_3 = z \oplus m$.
8. output $(c_1, c_2, c_3)$.
end
```

Algorithm 12 $D^*_{uhf}(x_A, y_B, p, g, c_1, c_2, c_3)$

```
1. $r' = (y_B \cdot c_1)^{x_A} \bmod p$.
2. $z' = G(r')_{[1 \cdots P]}$.
3. $s' = G(r')_{[(P+1) \cdots (P+Q_2)]}$.
4. $m' = z' \oplus c_3$.
5. if $h_{s'}(m') = c_2$ then
     output $(m' \mid\mid r')$
   else
     output $(\emptyset)$.
end
```

5 On $\mathcal{C}_{sig}$

The potential attack on $\mathcal{C}_{owh}$ and $\mathcal{C}_{uhf}$ discussed above does not apply to the third cryptosystem $\mathcal{C}_{sig}$. The cryptosystem is included here for future reference.

This cryptosystem also employs a cryptographically strong pseudo-random string generator G and a one-way hash function h.

Algorithm 13 $E_{sig}(y_A, p, g, m)$

```
    1. $x \in_R [1, p-1]$.
    2. $k \in_R [1, p-1]$ such that $\gcd(k, p-1) = 1$.
    3. $r = y_A^{x+k} \bmod p$.
    4. $z = G(r)_{[1 \cdots P]}$.
    5. $c_1 = g^x \bmod p$.
    6. $c_2 = g^k \bmod p$.
    7. $c_3 = (h(m) - xr)/k \bmod (p-1)$.
    8. $c_4 = z \oplus m$.
    9. output $(c_1, c_2, c_3, c_4)$.
end
```

Algorithm 14 $D_{sig}(x_A, p, g, c_1, c_2, c_3, c_4)$

```
    1. $r' = (c_1 \cdot c_2)^{x_A} \bmod p$.
    2. $z' = G(r')_{[1 \cdots P]}$.
    3. $m' = z' \oplus c_4$.
    4. if $g^{h(m')} \equiv c_1^{r'} \cdot c_2^{c_3} \pmod{p}$ then
         output $(m')$
       else
         output $(\emptyset)$.
end
```

When Bob has a secret-public key pair (x_B, y_B), a sender authentication capability can be added to the cryptosystem by replacing the random number x with Bob's secret key x_B. Here is the system with a sender authentication capability.

Algorithm 15 $E_{sig}(x_B, y_A, p, g, m)$

```
    1. $k \in_R [1, p-1]$ such that $\gcd(k, p-1) = 1$.
    2. $r = y_A^{x_B + k} \bmod p$.
    3. $z = G(r)_{[1 \cdots P]}$.
    4. $c_2 = g^k \bmod p$.
    5. $c_3 = (h(m) - x_B r)/k \bmod (p-1)$.
    6. $c_4 = z \oplus m$.
    7. output $(c_2, c_3, c_4)$.
end
```

Note that a ciphertext has only three, but not four, parts c_2, c_3 and c_4.

Algorithm 16 $D_{sig}(x_A, y_B, p, g, c_2, c_3, c_4)$

```
    1. $r' = (y_B \cdot c_2)^{x_A} \bmod p$.
    2. $z' = G(r')_{[1 \cdots P]}$.
    3. $m' = z' \oplus c_4$.
    4. if $g^{h(m')} \equiv y_B^{r'} \cdot c_2^{c_3} \pmod p$ then
         output $(m')$
       else
         output $(\emptyset)$.
end
```

Clearly, when enhanced with a sender authentication capability, the cryptosystem degenerates to a system employing both probabilistic encryption [BG85] and El Gamal's digital signature [ElG85].

Efficiency of $\mathcal{C}_{sig}$ can be improved by adapting Schnorr's digital signature scheme [Sch91], or SHS [Nat95], instead of El Gamal's.

In the case where the sender has more computing power than the receiver or simply is willing to do more computation, the calculation of $c_2^{c_3} \bmod p$ can be carried out by the sender. This has been pointed out in [HZ93].

Acknowledgments

This work was supported in part by the Australian Research Council under the reference number A49232172. Thanks go to Lim and Lee for pointing out and providing a solution to the potential problem.

References

[BG85] M. Blum and S. Goldwasser. An efficient probabilistic public key encryption scheme which hides all partial information. In G. R. Blakeley and D. Chaum, editors, *Advances in Cryptology — CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 289–299, Berlin, New York, Tokyo, 1985. Springer-Verlag.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73, New York, November 1993. The Association for Computing Machinery.

[CW79] J. Carter and M. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):472–492, 1976.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[HZ93] T. Hardjono and Y. Zheng. A practical digital multisignature scheme based on discrete logarithms. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology — AUSCRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 122–132, Berlin, New York, Tokyo, 1993. Springer-Verlag.

[LL94] C. H. Lim and P. J. Lee. Another method for obtaining security against chosen ciphertext attacks. In *Advances in Cryptology — CRYPTO'93*, Lecture Notes in Computer Science, Berlin, New York, Tokyo, 1994. Springer-Verlag. To appear.

[LO91] B. A. LaMacchia and A. M. Odlyzko. Computation of discrete logarithms in prime fields. *Designs, Codes and Cryptography*, 1:47–62, 1991.

[LW88] D. L. Long and A. Wigderson. The discrete logarithm hides $\mathrm{O}(\log n)$ bits. *SIAM Journal on Computing*, 17(2):363–372, 1988.

[Nat95] National Institute of Standards and Technology. Secure hash standard. Federal Information Processing Standards Publication FIPS PUB 180-1, U.S. Department of Commerce, April 1995.

[Per85] R. Peralta. Simultaneous security of bits in the discrete log. In Franz Pichler, editor, *Advances in Cryptology — EUROCRYPT'85*, volume 219 of *Lecture Notes in Computer Science*, pages 62–72, Berlin, New York, Tokyo, 1985. Springer-Verlag.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[WC81] M. Wegman and J. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

[Zhe93] Y. Zheng. STRANDOM — a cryptographically strong pseudo-random number generator based on HAVAL, July 1993. source code in the programming language C.

[ZPS93] Y. Zheng, J. Pieprzyk, and J. Seberry. HAVAL — a one-way hashing algorithm with varialbe length of output. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology — AUSCRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 83–104, Berlin, New York, Tokyo, 1993. Springer-Verlag.

[ZS93] Y. Zheng and J. Seberry. Immunizing public key cryptosystems against chosen ciphertext attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):715–724, June 1993.