Rethinking Chosen-Ciphertext Security under Kerckhoffs' Assumption

Seungjoo Kim[1], Masahiro Mambo[2], and Yuliang Zheng[3]

[1] KISA (Korea Information Security Agency),
78, Garag-Dong, Songpa-Gu, Seoul 138-803, Korea
skim@kisa.or.kr – http://www.crypto.re.kr
[2] Graduate School of Information Sciences, Tohoku University,
Kawauchi Aoba Sendai, 980-8576 Japan
mambo@icl.isc.tohoku.ac.jp – http://www.icl.isc.tohoku.ac.jp/~mambo/
[3] UNC Charlotte,
9201 University City Blvd, Charlotte, NC 28223
yzheng@uncc.edu – http://www.sis.uncc.edu/~yzheng/

Abstract. Any software claiming to cryptographically protect the data should use an encryption algorithm that meets public standards, and has an extensive history of independent cryptanalytic validation. However, even though they encrypt with strong encryption algorithm, most existing public-key cryptosystems, including RSA-OAEP, do not consider the "memory reconstruction attack" or the "memory core-dump attack" mounted by computer forensic software, information stealing viruses, or other accidental reasons. To deal with this situation, this paper attempts to analyze the existing provably secure cryptosystems under "Kerckhoffs' assumption" : an attacker knows all details of the cryptosystem except the key information, which security consequently rests entirely upon.

Keywords. Kerckhoffs' assumption, provable security, chosen-ciphertext security.

1 Introduction

A basic rule of cryptography is to use published, public algorithms and protocols. This principle, called *Kerckhoffs' assumption (also called Kerckhoffs' law or Kerckhoffs' principle)* was first stated in 1883 by Auguste Kerckhoffs : A cryptosystem should be designed to be secure if everything is known about it except the key information. It was reformulated (perhaps independently) by Claude Shannon as "the enemy knows the system". In that form it is called Shannon's Maxim.

Kerckhoffs' assumption was one of six design principles laid down by Kerckhoffs for military ciphers. Kerckhoffs' six cipher design principles were [22]:

1. The system must be practically, if not mathematically, undecipherable.

2. It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.
3. Its key must be communicable and retainable without the help of written notes, and changeable or modifiable at the will of the correspondents.
4. It must be applicable to telegraphic correspondence.
5. It must be portable, and its usage and function must not require the concourse of many people.
6. Finally, it is necessary, seeing the circumstances that the application commands, that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

Among the above six design principles, the second statement means that the method used to encipher data is known to the opponent, and that security must lie in the choice of key. A corollary of Kerckhoffs' second principle is that the fewer number of secrets a system has, the more secure it is. If the loss of any one secret causes the system to break, then the system with fewer number of secrets is necessarily more secure. The more secrets a system has, the more fragile it is. The fewer secrets, the more robust.

In this paper, we explore a generalization of Kerckhoffs' assumption fitted for the computer era. By using this generalization, we will study the security of the existing provably secure cryptosystems against chosen-ciphertext attack.

2 Security Definitions

Definition 1 (Generalized Kerckhoffs' Assumption). *Let's consider a Turing Machine(TM), a program for which is an encryption or a decryption algorithm. The TM has 5 tapes : input, output, random, work, and key.* Generalized Kerckhoffs' assumption *means an attacker knows all details of the TM except the key tape, where "all details of the TM" consist of*

– *(External Details : the original Kerckhoffs' assumption deals with only this situation) a triple (G, E, D) satisfying the following conditions and the bit sequence contained in the input/output tape of TM; and*

  - *key generation algorithm G : G, on input 1^k (the security parameter), produces a pair of encryption/decryption keys (e, d). The encryption key e is stored in the input tape, but the decryption key d is stored in the key tape.*
  - *encryption algorithm E : E takes as input a security parameter 1^k, an encryption key e from the range of $G(1^k)$, and a message $m \in \{0,1\}^k$, and produces as output the ciphertext $c \in \{0,1\}^*$[1].*

[1] We use the notation $c \in E(1^k, e, m)$ to denote c being an encryption of message m using key e with security parameter k. When clear, we use shorthand $c \in E_e(m)$, or $c \in E(m)$.

  - *decryption algorithm D : D takes as input a security parameter 1^k, a secret decryption key d from the range of $G(1^k)$, and a ciphertext c from the range of $E(1^k, e, m)$ and produces as output the message m.*

- *(Internal Details) The bit sequence contained in the random/work tape of TM. If giving the attacker the ability not only to read but also to write or modify the random/work tape of TM to get the faulty outputs, we can simulate the* fault attacks *[8].*

Note here that, when D_d(resp. E_e) is being applied to a target ciphertext itself(resp. a plaintext corresponding to a target ciphertext), the attacker cannot get access to the input/output/random/work tape of TM.

Under the assumption of Definition 1, attacks are classified based on what information an attacker has access to in addition to intercepted ciphertext. The most prominent classes of attack for public-key cryptosystems are : ciphertext-only attack, known-plaintext attack, chosen-plaintext attack, and chosen-ciphertext attack[29, 34]. In the following, we especially consider the chosen-ciphertext attack under the generalized Kerckhoffs' assumption. We note that the scope of our assumption is very broad and may apply to other cryptographic primitives as well (e.g., digital signatures).

Definition 2 (Chosen-Ciphertext Query under Generalized Kerckhoffs' Assumption). *Let k be a security parameter that generates matching encryption/decryption keys (e, d) for each user in the system. A* chosen-ciphertext query under generalized Kerckhoffs' assumption *is a process which, on input 1^k and e, obtains*

- *(External Details) the plaintext (relatively to d), in the output tape of decryption oracle, corresponding to a chosen ciphertext; or an indication that the chosen ciphertext is invalid[2]; and also*
- *(Internal Details) non-erased internal states contained in the random/work tape of decryption oracle decrypting the submitted ciphertext.*

[2] As explicitly pointed out in [21], we stress that the adversary may query the decryption oracle with invalid ciphertexts. Although seemingly useless, such attacks are not innocuous. The decryption oracle can for example be used to learn whether a chosen ciphertext is valid or not. From this single bit of information and by iterating the process, Bleichenbacher successfully attacked several implementations of protocols based on PKCS #1 v1.5 [9]. More recently, Manger pointed out the importance of preventing an attacker from distinguishing between rejections at the various steps of the decryption algorithm, say, using timing analysis [28]. The lesson is that implementors must ensure that the reasons for which a ciphertext is rejected are hidden from the outside world.

Definition 3 ([Static/Adaptive] Chosen-Ciphertext Attack under Generalized Kerckhoffs' Assumption). *A* static chosen-ciphertext attack[3] under generalized Kerckhoffs' assumption *consists of the following scenario:*

1. *On input a security parameter k, the key generation algorithm G is run, generating a public encryption key $e \in G(1^k)$ and a private decryption key $d \in G(1^k)$ for the encryption algorithm E. The attacker of course knows the public key e, but the private key d is kept secret.*
2. *[Find stage] The attacker makes polynomially (in k) many chosen-ciphertext queries (as in Definition 2) to a decryption oracle. (The attacker is free to construct the ciphertexts in an arbitrary way — it is certainly not required to compute them using the encryption algorithm.)*
3. *The attacker prepares two messages m_0, m_1 and gives these to an encryption oracle. The encryption oracle chooses $b \in_R \{0,1\}$ at random, encrypts m_b, and gives the resulting "target ciphertext" c' to the attacker. The attacker is free to choose m_0 and m_1 in an arbitrary way, except that they must be of the same length.*
4. *[Guess stage] The attacker outputs $b' \in \{0,1\}$, representing its "guess" on b.*

In an adaptive chosen-ciphertext attack under generalized Kerckhoffs' assumption, *the attacker has still access to the decryption oracle after having received the target ciphertext : a second series of polynomially (in k) many chosen-ciphertext queries may be run. The unique restriction is not to probe the decryption oracle with the target ciphertext c'.*

The success probability in the previous attack scenario is defined as

$$\Pr[b' = b] \ .$$

Here, the probability is taken over coin tosses of the adversary, the key generation algorithm G and the encryption algorithm E, and $(m_0, m_1) \in M^2$ where M is the domain of the encryption algorithm E.

The attack of Definition 3 is very powerful. The attacker not only can obtain the plaintexts corresponding to chosen ciphertexts, but can also invade user's computer and read the contents of its memory (i.e., non-erased internal data) [23, 24]. We believe that this attack model approximates more closely existing security systems, many of which are built on such operating systems as Unix and Windows where a reasonably privileged user can interrupt the operation of a computing process and inspect its intermeleged results at ease. Thus our attack model can deal with the "memory reconstruction attack [10]" or the "memory core-dump attack [23, 24]" mounted by computer forensic software[4], information stealing viruses, or other accidental reasons.

[3] In the past, this attack has also been called "lunch-time attack" or "midnight attack".

[4] Computer forensic is to gather and analyze data in a manner as free from distortion or bias as possible to reconstruct data or what has happened in the past on a system.

Definition 4 (Chosen-Ciphertext Security under Generalized Kerckhoffs' Assumption). *"An encryption scheme is* (t, q, ε)*-secure under* $(l_{\text{in}}, l_{\text{out}}, l_{\text{rand}}, l_{\text{work}})$*-generalized Kerckhoffs' assumption" means that*

- *We basically assume that an attacker does not know the decryption key* d *stored in the key tape of decryption oracle. Additionally, we suppose that at least* $(l_{\text{in}} + l_{\text{out}} + l_{\text{rand}} + l_{\text{work}})$*-bit sequence collectable from each input tape, output tape, random tape and work tape of decryption oracle is kept secret to the attacker.*
- *In the above assumptions, the chosen-ciphertext attacker of Definition 3 can break the encryption scheme with advantage less than* ε*, when s/he runs for time* t *and makes* q *queries to the decryption oracle.*

Note here that, under the generalized Kerckhoffs' assumption, the power of a chosen-ciphertext attack deeply depends on the four-tuple of $(l_{\text{in}}, l_{\text{out}}, l_{\text{rand}}, l_{\text{work}})$, so we can consider this four-tuple as one of the security evaluation criteria of a given cryptosystem. Thus, for example, when two encryption schemes based on the same cryptographic assumptions, E_e and E'_e, are given, we would say something like : "Even if both of an encryption scheme E_e and an E'_e are IND-CCA2 in terms of the definition of [4], when an attacker intentionally chooses and sends invalid ciphertexts, $(\delta_{\text{in}}, \delta_{\text{out}}, \delta_{\text{rand}}, \delta_{\text{work}})$ more bits of E'_e, except the key information, should be kept secret than E_e using the same security parameter as E'_e."

2.1 Forward Security versus Ours

Forward security is defined as the confidence that the compromise of a long-term private key does not compromise any earlier session keys. So, for example, even if the private decryption key is compromised and an eavesdropper has captured encrypted messages which has previously been sent, s/he will not be able to decrypt them.

Proactive cryptography [31], exposure-resilient cryptography [11], forward-secure signatures [2, 5], and key-insulated cryptography [15, 16] may all be viewed as different means of taking this approach.

While the forward security model deals with a resistance to *known-key attacks*, our model is motivated by work on *strong chosen-ciphertext attack with memory dump* [23, 24]. In our model, an attacker knows even the internal states of the system. The sole restriction is that the private decryption keys are inaccessible.

3 How to Measure the Amount of Information Appeared in Work Tape

To determine the amount of bit sequence collectable from work tape, first of all, we suppose a crypto library which provides the subroutines of Table 1, Table 2, Table 3 and Table 4 for manipulating arbitrary length integers over the integers and over finite fields[5]. Note here that the (unit) operations contained inside a subroutine must be executed without external interruptions (e.g., memory core dump, system crash, user signal to kill the transaction, and so on). After the execution being successfully completed, the internal variables used by subroutine are erased. Refer to [14, 23, 24] for details.

Table 1. Subroutines for basic calculations

Subroutine	Functionality
`Abs(x;a)`	x is the absolute value of a
`Negate(x;a)`	$x = -a$
`Add(x;a,b)`	$x = a + b$
`Sub(x;a,b)`	$x = a - b$
`Mul(x;a,b)`	$x = a \times b$
`DivRem(q,r;a,b)`	division of a by b, quotient in q, remainder in r
`Power(x;a,e)`	$x = a^e$
`Sqr(x;a)`	$x = a^2$
`GCD(x;a,b)`	$x = \gcd(a, b)$
`XGCD(d,s,t;a,b)`	$d = \gcd(a, b) = a \times s + b \times t$

Table 2. Subroutines for modular arithmetic

Subroutine	Functionality
`Mod(x;a,n)`	$x = a \bmod n$
`NegateMod(x;a,n)`	$x = -a \bmod n$
`AddMod(x;a,b,n)`	$x = a + b \bmod n$
`SubMod(x;a,b,n)`	$x = a - b \bmod n$
`MulMod(x;a,b,n)`	$x = a \times b \bmod n$
`InvMod(x;a,n)`	$x = a^{-1} \bmod n$
`PowerMod(x;a,e,n)`	$x = a^e \bmod n$
`SqrMod(x;a,n)`	$x = a^2 \bmod n$
`Jacobi(a,n)`	compute Jacobi symbol of $(a\|n)$

Now, we write pseudo-code to describe the given algorithm by using our crypto library, and then count the size of the temporary variables used for intermediate values or return value of subroutines. When writing the pseudo-code, we assume that, the data can be packed into a variable bit-by-bit(or byte-by-byte) as the bit-field structure of C programming language. Thus, for variables,

[5] We name the module according to the NTL(A Library for doing Number Theory) of Victor Shoup, and use ";" between output and input of all subroutines.

Table 3. Subroutines for bitwise operations

Subroutine	Functionality
`LeftShift(x;a,n)`	x is to shift the bits of a left by n
`RightShift(x;a,n)`	x is to shift the bits of a right by n (the sign is preserved)
`Bit_And(x;a,b)`	x is the bitwise AND of a and b
`Bit_Or(x;a,b)`	x is the bitwise OR of a and b
`Bit_Xor(x;a,b)`	x is the bitwise XOR of a and b

Table 4. Etc.

Subroutine	Functionality
`GenPrime(n;l,err)`	generating a random prime n of length l so that the probability that n is composite is bounded by $2^{-\mathrm{err}}$
`ProbPrime(n,NumTrials)`	perform up to `NumTrials` Miller-witness tests of n
`HashH(x;a[,b,···])`	x is the hash value of $H(a[,b,\cdots])$
`HashG(x;a[,b,···])`	x is the hash value of $G(a[,b,\cdots])$
`MemKill(a[,b,···])`	delete and set to zero the memory allocated for $a[,b,\cdots]$

we use the notation $V : V = v_{l-1}256^{l-1} + v_{l-2}256^{l-2} + \cdots + v_1 256 + v_0 \overset{\text{def}}{=} \{v_{l-1}v_{l-2}\cdots v_1 v_0\}$, where $0 \leq v_i < 256$. To indicate some consecutive bytes of an integer V, we use a shorter notation $[V]_a^b$, meaning a sequence of bytes from the b-th byte to a-th byte of v, $\{v_b \cdots v_a\}$.

Programming Style **:** Each programmer will, of course, have his or her own preferences in wiring style. Since the programming style may effect the memory use and the performance, we should restrict something on writing style of pseudo-code : (i) do not reuse variables. For example,

```
result = function1(input); result = function2(result);
```

(ii) do not use nested functions as

```
result = function2(function1(input));,
```

but call them on consecutive lines as

```
result1 = function1(input); result2 = function2(result1);.
```

From now on, when we express "a cryptosystem is implemented with the crypto library described in this section", we assume

- the given cryptosystem is implemented with the crypto library and the programming style defined in this section; and,
- operations contained inside a subroutine of library must be executed without external interruptions; and,
- decryption key itself appeared in the pseudo-code is always protected even if it is place on the work tape; and,
- the faulty behavior[8] of the cryptosystem is excluded from the evaluation.

4 Evaluation of Provably Secure Cryptosystems

For the last few years, many new schemes have been proposed with provable security against chosen-ciphertext attacks. Before 1994, only theoretical (i.e., not very practical) schemes were proposed. Then Bellare and Rogaway [7] came up with the *random oracle model* [6] and subsequently designed in [7] a generic padding, called OAEP (Optimal Asymmetric Encryption Padding), to transform a one-way (partially) trapdoor *permutation* into a chosen-ciphertext secure cryptosystem. Other generic paddings, all validated in the random oracle model, were later given by Fujisaki and Okamoto [18] (improved in [19]), by Pointcheval [33], and by Okamoto and Pointcheval [30]. Recently, Coron *et al.* proposed a generic IND-CCA2 conversion, which reduced the encryption size and/or speeded up the decryption process [12]. The first practical cryptosystem with provable security in the *standard model* is due to Cramer and Shoup [13]. They present an extended ElGamal encryption provably secure under the *decisional* Diffie-Hellman problem. (Cramer-Shoup system is also provably secure under the weaker *computational* Diffie-Hellman assumption in the random oracle model.)

In this section, we try to evaluate the existing provably secure cryptosystems under the generalized Kerckhoffs' assumption. Note here that, when evaluating the existing provably secure cryptosystems, we do not consider the computational difficulties of the underlying cryptographic assumptions.

4.1 RSA-OAEP

We give here a brief overview of RSA-OAEP. We describe the RSA-OAEP scheme as described in [32], but in our description, the decryption phase of RSA-OAEP is divided into three parts : (i) RSA decryption, (ii) validity test, and (iii) output.

Let k-byte string $n = pq$ denote an RSA modulus, which is the product of two large primes p and q. Furthermore, let e and d, satisfying $ed \equiv 1 \pmod{\mathrm{lcm}(p-1, q-1)}$, respectively denote the public encryption exponent and the private decryption exponent. We assume a hash function $H : \{0,1\}^{k-hLen-1 \text{ bytes}} \to \{0,1\}^{hLen \text{ bytes}}$ and a "generator" function $G : \{0,1\}^{hLen \text{ bytes}} \to \{0,1\}^{k-hLen-1 \text{ bytes}}$. The public parameters are $\{n, e, G, H\}$ and the secret parameters are $\{d, p, q\}$.

A $mLen$-byte plaintext message m is encrypted through RSA-OAEP as :

1. Form a data block DB of length $k - hLen - 1$ bytes as
$$DB = (H(L) \| \mathtt{0x00}^{k-mLen-2hLen-2} \| \mathtt{0x01} \| m),$$
where L is an optional label to be associated with the message; the default value for L, if L is not provided, is the empty string; and
2. Let $maskedDB = DB \oplus G(seed)$, with a random string $seed$ of length $hLen$-bytes[6]; and

[6] In [32], $maskedDB = DB \oplus MGF(seed, k - hLen - 1)$.

3. Let $maskedSeed = seed \oplus H(maskedDB)$[7]; and
4. Form an encoded message EM of length k bytes as

$$EM = (\texttt{0x00} \| maskedSeed \| maskedDB); \text{ and}$$

5. RSA encryption $c = EM^e \bmod n$.

Given a ciphertext c, m is recovered as :

```
(i) $\boxed{\textit{RSA decryption}}$

    1 : PowerMod($V1; c, d, n$);
        $/^* \; V1 \overset{\text{def}}{=} \{\underbrace{v1_{k-1}}_{\texttt{0x00}}\underbrace{v1_{k-2}\ldots\ldots v1_{k-hLen-1}}_{maskedSeed}\underbrace{v1_{k-hLen-2}\ldots\ldots v1_0}_{maskedDB}\} \; ^*/$
    2 : HashH($V2; [V1]_0^{k-hLen-2}$);
    3 : Bit_Xor($V3; [V1]_{k-hLen-1}^{k-2}, V2$);
    4 : HashG($V4; V3$);
    5 : Bit_Xor($V5; [V1]_0^{k-hLen-2}, V4$);
        $/^* \; V5 \overset{\text{def}}{=} \{\underbrace{v5_{k-hLen-2}\ldots\ldots v5_{k-2hLen-1}}_{H(L)}\underbrace{v5_{k-2hLen-2}\ldots\ldots v1_{mLen+1}}_{\texttt{0x00}^{k-mLen-2hLen-2}}$
        $\underbrace{v5_{mLen}}_{\texttt{0x01}}\underbrace{v5_{mLen-1}\ldots\ldots v5_0}_{m}\} \; ^*/$

(ii) $\boxed{\textit{Validity Test}}$

    If ($v5_{mLen} \neq \texttt{0x01} \; \bigvee \; [V5]_{k-2hLen-1}^{k-hLen-2} \neq H(L) \; \bigvee \; v1_{k-1} \neq \texttt{0x00}$) then
      • MemKill($V1, V2, V3, V4, V5$); and
      • Return("Invalid Ciphertext").

(iii) $\boxed{\textit{Output}}$

      • Return($[V5]_0^{mLen-1}$).
```

Theorem 1. *RSA-OAEP, implemented with crypto library of Section 3, is IND-CCA2 under* $(0, 0, 0, 2|n|)$*-generalized Kerckhoffs' assumption. Here,* $|n|$ *denotes the bit length of modulus* n.

Proof (Sketch). The problem with RSA-OAEP resides in that the validity test cannot be performed (i.e., the adversary cannot be detected) until *after* the RSA decryption is completed. Thus an attacker can freely mount an adaptive chosen-ciphertext attack and extract the partial information from internal data in the decryption oracle's memory [23, 24]. To prevent this type of attack, $V1 = \texttt{0x00} \| V3 \oplus V2 \| V5 \oplus V4$ should not leak out. Thus, the variables $V1$, $V3$ and $V5$ should be protected against the attacker.

[7] In [32], $maskedDB = seed \oplus MGF(maskedDB, hLen)$.

4.2 Abe Scheme

Let (n, e) be RSA public encryption key and d be private decryption key. We select a hash function $H : \mathbf{Z}_n^* \times \mathbf{Z}_n^* \rightarrow \{0,1\}^{l_m}$, $G : \{0,1\}^* \rightarrow \{0,1\}^l$ where l is a security parameter.

In Abe scheme[8], ciphertext C of message $m \in \{0,1\}^{l_m}$ is $C = (h, c, u, s)$ such that $h = r^e \bmod n$, $c = m \oplus H(h, r)$, $u = G(c, w^e \bmod n)$, $s = w \cdot r^u \bmod n$, where $r, w \in_U \mathbf{Z}_n^*$. Given a ciphertext C, m is recovered as follows. Contrary to RSA-OAEP, the validity test of Abe Scheme is performed *before* the RSA decryption :

```
(i) Validity Test

    1 : PowerMod($V1; s, e, n$);
    2 : PowerMod($V2; h, u, n$);
    3 : InvMod($V3; V2, n$);
    4 : MulMod($V4; V1, V3, n$);
    5 : HashG($V5; c, V4$);
    6 : Check if $u \neq V5$.

(ii) ElGamal decryption

    If ($u \neq V5$) then
        • MemKill($V1, V2, V3, V4, V5$); and
        • Return("Invalid Ciphertext").
    Else
        7 : PowerMod($V6; h, d, n$);
        8 : HashH($V7; h, V6$);
        9 : Bit_Xor($V8; c, V7$);

(iii) Output

        • Return($V8$).
```

Theorem 2. *Abe scheme, implemented with crypto library of Section 3, is IND-CCA2 under Kerckhoffs' assumption.*

Proof (Sketch). Suppose that, given a target ciphertext C, a chosen-ciphertext attacker modifies it and sends C' to a decryption oracle. Because Abe scheme is a chosen-ciphertext secure encryption scheme, the work tape of decryption oracle contains only $V1$, $V2$, $V3$, $V4$ and $V5$. However, even though the attacker can get access to these temporary variables, s/he has no advantage, since whatever the attacker can compute with $V1$, $V2$, $V3$, $V4$ and $V5$, s/he can also compute only with C' by her/himself.

[8] Although the motivation is quite different, a similar "validation-then-decryption"-type RSA scheme was imagined by the authors of this paper : ciphertext C of message m is (c, v_1, v_2) such that : $c = w^e \bmod n$, where $w = s \| t$, $s = m \oplus F(r)$, and $t = r \oplus G(s)$, $v_1 = H(n, e, c, ID_{sender}, x^{-e} \bmod n)$ with $x \in_R \mathbf{Z}_n^*$, $v_2 = x \cdot w^{v_1} \bmod n$.

4.3 Cramer-Shoup Scheme

We give here a brief overview of Cramer-Shoup scheme and refer the reader to [13] for details. Here, for simplicity, the same notation H is used, but function H may have different domain and image from that used in RSA-OAEP of Section 4.1 and Abe scheme of Section 4.2. Let $\mathcal{G} = \langle g \rangle$ be a cyclic group of large prime order q, generated by g.

The encryption of Cramer-Shoup scheme is $(c_1, c_2, c_3, c_4) = ({g_1}^s, {g_2}^s, {X_1}^s \cdot m, {X_2}^s \cdot {X_3}^{s \cdot H(c_1,c_2,c_3)})$ with public keys $X_1 = {g_1}^z$, $X_2 = {g_1}^{x_1} \cdot {g_2}^{x_2}$, and $X_3 = {g_1}^{y_1} \cdot {g_2}^{y_2}$. The decryption is as follows :

(i) $\boxed{\textit{Validity Test}}$

```
1 : HashH($V1; c_1, c_2, c_3$);
2 : MulMod($V2; y1, V1, q$);
3 : AddMod($V3; x_1, V2, q$);
4 : PowerMod($V4; c_1, V3, p$);
5 : MulMod($V5; y2, V1, q$);
6 : AddMod($V6; x_2, V5, q$);
7 : PowerMod($V7; c_2, V6, p$);
8 : MulMod($V8; V4, V7, p$);
9 : Check if $c_4 \neq V8$.
```

(ii) $\boxed{\textit{ElGamal decryption}}$

```
If ($c_4 \neq V8$) then
    • MemKill ($V1, V2, V3, V4, V5, V6, V7, V8$); and
    • Return("Invalid Ciphertext").
Else
    10 : NegateMod($V9; z, q$);
    11 : PowerMod($V10; c_1, V9, p$);
    12 : MulMod($V11; V10, c_3, p$);
```

(iii) $\boxed{\textit{Output}}$

```
• Return($V11$).
```

Theorem 3. *Cramer-Shoup scheme, implemented with crypto library of Section 3, is IND-CCA2 under* $(4|q|, 0, 0, 2|q|)$*-generalized Kerckhoffs' assumption.*

Proof (Sketch). For the scheme by Cramer and Shoup, some secret data are involved in the validity test. As a consequence, not only the private decryption key (i.e., z) of key tape but also the private validation keys (i.e., (x_1, x_2) and (y_1, y_2)) of input tape need to be kept secret. Also $V2$ and $V5$ of work tape should be protected.

4.4 Other Provably Secure Cryptosystems

Several variants of the basic RSA scheme[35] and the basic ElGamal scheme[17] were proposed in order to make it provably secure against chosen-ciphertext attacks. In this section, we review some of them in the chronological order of their appearance and analyze their resistance under our security model. Table 5 shows, when implemented with crypto library of Section 3, how many bits of information of decryption oracle except the decryption key should be protected from the attacker of Definition 3. Here, l_d denotes the bit length of decryption key of key tape.

Tsiounis and Yung [37]

- encryption: $(c_1, c_2, c_3, c_4) = (g^y, X^y \cdot m, g^k, y \cdot H(g, c_1, c_2, c_3) + k)$
- decryption:

```
 1) PowerMod($V1; g, c_4, p$);
 2) HashH($V2; g, c_1, c_2, c_3$);
 3) PowerMod($V3; c_1, V2, p$);
 4) MulMod($V4; V3, c_3, p$);
 5) If ($V1 \neq V4$) then
 6)    • MemKill($V1, V2, V3, V4$); and
 7)    • Return("Invalid Ciphertext").
    Else
 8)    • NegateMod($V5; x, q$);
 9)    • PowerMod($V6; c_1, V5, p$);
10)    • MulMod($V7; V6, c_2, p$);
11)    • Return($V7$).
```

Fujisaki and Okamoto [18]

- encryption: $(c_1, c_2) = (g^{H(m\|s)}, (m\|s) \oplus X^{H(m\|s)})$
- decryption:

```
1) PowerMod($V1; c_1, x, p$);
2) Bit_Xor($V2; V1, c_2$);
3) HashH($V3; V2$);
4) PowerMod($V4; g, V3, p$);
5) If ($c_1 \neq V4$) then
6)    • MemKill($V1, V2, V3, V4$); and
7)    • Return("Invalid Ciphertext").
8) Return($m$ of $V2$).
```

- attack: 1) set $(c_1', c_2') = (c_1, c_2 \oplus r)$ for a random r
  2) recover m from $(m'\| \cdots) \oplus r = m\| \cdots$

Fujisaki and Okamoto [19]

- encryption: $(c_1, c_2, c_3) = (g^{H(s,m)}, X^{H(s,m)} \cdot s, \mathcal{E}^{\mathsf{sym}}_{G(s)}(m))$

- decryption:
```
1) NegateMod($V1; x, q$);
2) PowerMod($V2; c_1, V1, p$);
3) MulMod($V3; V2, c_2, p$);
4) HashG($V4; V3$);
5) $V5 = \mathcal{D}^{\mathsf{sym}}_{V4}(c_3)$;
6) HashH($V6; V3, V5$);
7) PowerMod($V7; X, V6, p$);
8) MulMod($V8; V7, V3, p$);
9) If ($c_2 \neq V8$) then
10)     • MemKill($V1, V2, V3, V4, V5, V6, V7, V8$); and
11)     • Return("Invalid Ciphertext").
12) Return($V5$).
```
- attack: 1) set $(c_1', c_2', c_3') = (g^r \cdot c_1, X^r \cdot c_2, c_3)$ for a random r
  2) recover $m = m'$

Pointcheval [33]

- encryption: $(c_1, c_2, c_3) = (g^{H(m\|s)}, X^{H(m\|s)} \cdot k, (m\|s) \oplus G(k))$
- decryption:
```
1) NegateMod($V1; x, q$);
2) PowerMod($V2; c_1, V1, p$);
3) MulMod($V3; V2, c_2, p$);
4) HashG($V4; V3$);
5) Bit_Xor($V5; V4, c_3$);
6) HashH($V6; V5$);
7) PowerMod($V7; g, V6, p$);
8) If ($c_1 \neq V7$) then
9)      • MemKill($V1, V2, V3, V4, V5, V6, V7$); and
10)     • Return("Invalid Ciphertext").
11) Return($m$ of $V5$).
```
- attack: 1) set $(c_1', c_2', c_3') = (c_1, c_2, c_3 \oplus r)$ for a random r
  2) recover m from $(m'\| \cdots) \oplus r = m\| \cdots$

Baek, Lee, and Kim [3]

- encryption: $(c_1, c_2) = (g^{H(m\|s)}, (m\|s) \oplus G(X^{H(m\|s)}))$
- decryption:
```
1) PowerMod($V1; c_1, x, p$);
2) HashG($V2; V1$);
3) Bit_Xor($V3; V2, c_2$);
4) HashH($V4; V3$);
5) PowerMod($V5; g, V4, p$);
6) If ($c_1 \neq V5$) then
7)     • MemKill($V1, V2, V3, V4, V5$); and
8)     • Return("Invalid Ciphertext").
9) Return($m$ of $V3$).
```
- attack: 1) set $(c_1', c_2') = (c_1, c_2 \oplus r)$ for a random r
  2) recover m from $(m'\| \cdots) \oplus r = m\| \cdots$

Schnorr and Jakobsson [36]

- encryption: $(c_1, c_2, c_3, c_4) = (g^y, G(X^y) + m, H(g^s, c_1, c_2), s + c_3 \cdot y)$

```
• decryption:  1) PowerMod($V1; g, c_4, p$);
               2) NegateMod($V2; c_3, q$);
               3) PowerMod($V3; c_1, V2, p$);
               4) MulMod($V4; V1, V3, p$);
               5) HashH($V5; V4, c_1, c_2$);
               6) If ($c_3 \neq V5$) then
               7)     • MemKill($V1, V2, V3, V4, V5$); and
               8)     • Return(“Invalid Ciphertext”).
                  Else
               9)     • PowerMod($V6; c_1, x, p$);
              10)     • HashG($V7; V6$);
              11)     • SubMod($V8; c_2, V7, p$);
              12)     • Return($V8$).
```

Okamoto and Pointcheval [30]

- encryption: $(c_1, c_2, c_3, c_4) = (g^y, X^y \oplus R, \mathcal{E}^{\mathsf{sym}}_{G(R)}(m), H(R, m, c_1, c_2, c_3))$

```
• decryption:  1) PowerMod($V1; c_1, x, p$);
               2) Bit_Xor($V2; V1, c_2$);
               3) HashG($V3; V2$);
               4) $V4 = \mathcal{D}^{\mathsf{sym}}_{V3}(c_3)$
               5) HashH($V5; V2, V4, c_1, c_2, c_3$);
               6) If ($c_4 \neq V5$) then
               7)     • MemKill($V1, V2, V3, V4, V5$); and
               8)     • Return(“Invalid Ciphertext”).
               9) Return($V4$).
• attack:  1) set $(c'_1, c'_2, c'_3, c'_4) = (c_1, c_2, c_3, c'_4)$ with $c'_4 \neq c_4$
           2) recover $m = m'$
```

Table 5. Analysis of several RSA and ElGamal variants.

Variant	Type	l_d	$(l_{\text{in}}, l_{\text{out}}, l_{\text{rand}}, l_{\text{work}})$
RSA-OAEP [32, 7]	DtV	$\lvert\text{lcm}(p-1, q-1)\rvert$	$(0, 0, 0, 2\lvert n\rvert)$
Tsiounis-Yung [37]	VtD	$\lvert q\rvert$	$(0, 0, 0, 0)$
Cramer-Shoup [13]	VtD	$\lvert q\rvert$	$(4\lvert q\rvert, 0, 0, 2\lvert q\rvert)$
Fujisaki-Okamoto [18]	DtV	$\lvert q\rvert$	$(0, 0, 0, 2\lvert p\rvert)$
Fujisaki-Okamoto [19]	DtV	$\lvert q\rvert$	$(0, 0, 0, \lvert q\rvert + 2\lvert p\rvert + \lvert G(\cdot)\rvert)$
Pointcheval [33]	DtV	$\lvert q\rvert$	$(0, 0, 0, \lvert q\rvert + 2\lvert p\rvert + 2\lvert(m\Vert s)\rvert)$
Baek-Lee-Kim [3]	DtV	$\lvert q\rvert$	$(0, 0, 0, \lvert p\rvert + 2\lvert(m\Vert s)\rvert)$
Schnorr-Jakobsson [36]	VtD	$\lvert q\rvert$	$(0, 0, 0, 0)$
Okamoto-Pointcheval [30]	DtV	$\lvert q\rvert$	$(0, 0, 0, 2\lvert p\rvert + \lvert G(\cdot)\rvert + \lvert m\rvert)$
Abe [1]	VtD	$\lvert\text{lcm}(p-1, q-1)\rvert$	$(0, 0, 0, 0)$

DtV : Decryption-then-Validation
VtD : Validation-then-Decryption

5 Conclusion

In this paper we attempted to discuss the security of a cryptosystem under the sole assumption that the secret key is protected. Although putting additional assumptions to the ideal scenario at present, we hope that in the future cryptographers will improve our model and analyze the security of their schemes in this setting.

Acknowledgements

The authors are grateful to Marc Joye and Jaechul Sung for helpful discussions on the subject of this paper. We would also like to thank the anonymous referees for valuable comments.

References

1. M. Abe, "Securing "encryption + proof of knowledge" in the random oracle model", *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 277-289, Springer-Verlag, 2002.
2. R. Anderson, Invited lecture, *Fourth ACM Conference on Computer and Communications Security*, ACM, 1997.
3. J. Baek, B. Lee, and K. Kim, "Secure length-saving ElGamal encryption under the computational Diffie-Hellman assumption", *Information Security and Privacy (ACISP 2000)*, volume 1841 of *Lecture Notes in Computer Science*, pages 49–58, Springer-Verlag, 2000.
4. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes", *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45, Springer-Verlag, 1998.
5. M. Bellare and S. Miner, "A forward-secure digital signature scheme", *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448, Springer-Verlag, 1999.
6. M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols", *First ACM Conference on Computer and Communications Security*, pages 62–73, ACM Press, 1993.
7. M. Bellare and P. Rogaway, "Optimal asymmetric encryption", *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Springer-Verlag, 1995.
8. Bellcore Press Release, "New threat model breaks crypto codes", Sept. 1996, http://www.bellcore.com/PRESS/ADVSRY96/facts.html/.
9. D. Bleichenbacher, "A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS #1", *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12, Springer-Verlag, 1998.
10. S. Burnett and S. Paine, "RSA Security's official guide to cryptography", *RSA Press*, 2001.

11. R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai, "Exposure-resilient functions and all-or-nothing-transforms", *Advances in Cryptology – Eurocrypt '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 453–469, Springer-Verlag, 2000.
12. J.S. Coron, H. Handshuch, M. Joye, P. Paillier, D. Pointcheval, and C. Tymen, "GEM: A generic chosen-ciphertext secure encryption method", *Topics in Cryptology - CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 263–276, Springer-Verlag, 2002.
13. R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack", *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Springer-Verlag, 1998.
14. G. Di Crescenzo, N. Ferguson, R. Impagliazzo, and M. Jakobsson, "How to forget a secret", *Annual Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 500-509, 1999.
15. Y. Dodis, J. Katz, S. Xu, and M. Yung, "Strong key-insulated signature schemes", Unpublished Manuscript.
16. Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems", *Advances in Cryptology – Eurocrypt '02*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82, Springer-Verlag, 2002.
17. T. ElGamal, "A public key cryptosystems and a signature schemes based on discrete logarithms", *IEEE Transactions on Information Theory*, **IT-31**(4):469–472, 1985.
18. E. Fujisaki and T. Okamoto, "How to enhance the security of public-key encryption at minimum cost", *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68, Springer-Verlag, 1999.
19. E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes", *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–544, Springer-Verlag, 1999.
20. L.C. Guillou and J.-J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory", *Advances in Cryptology – EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128, Springer-Verlag, 1988.
21. M. Joye, J.-J. Quisquater, and M. Yung, "On the power of misbehaving adversaries", *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 208-222, Springer-Verlag, 2001.
22. A. Kerckhoffs, "La cryptographie militaire (Military Cryptography)", *Journal des sciences militaires*, vol.IX, pages 5-83, Jan. 1883, pages 161-191, Feb. 1883.
23. S. Kim, J.H. Cheon, M. Joye, S. Lim, M. Mambo, D. Won, and Y. Zheng "Strong adaptive chosen-ciphertext attack with memory dump (Or : The importance of the order of decryption and validation)", *Eighth IMA Conference on Cryptography and Coding 2001*, volume 2260 of *Lecture Notes in Computer Science* pages 114-127, Springer-Verlag, 2001.
24. S. Kim, J.H. Cheon, M. Joye, S. Lim, M. Mambo, D. Won, and Y. Zheng, "Security analysis of "provably" secure cryptosystems under strong adaptive chosen-ciphertext attack", *Technical Report of IEICE*, ISSN 0913-5685, ISEC2001-89, Vol.101, No.507, pages 17-24, 2001.
25. P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems", *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Springer-Verlag, 1996.

26. P. Kocher, J. Jaffe, and B. Jun, “Introduction to differential power analysis and related attacks”, 1998, http://www.cryptography.com/dpa/technical.
27. P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis”, *Advances in Cryptology – CRYPTO ’99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Springer-Verlag, 1999.
28. J. Manger, “A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS #1”, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 230–238, Springer-Verlag, 2001.
29. M. Naor and M. Yung, “Public-key cryptosystems provably secure against chosen ciphertext attacks”, *22nd Annual ACM Symposium on Theory of Computing*, pages 427-437, ACM Press, 1990.
30. T. Okamoto and D. Pointcheval, “REACT: Rapid enhanced-security asymmetric cryptosystem transform”, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 159–175, Springer-Verlag, 2001.
31. R. Ostrovsky and M. Yung, “How to withstand mobile virus attacks”, *Proceedings of the Tenth Annual ACM Symposium on Princiles of Distributed Computing – PODC ’91*, pages 51–59, ACM Press, 1991.
32. PKCS #1 v2.1 : RSA Cryptography Standard, June 14, 2002.
http://www.rsasecurity.com/rsalabs/pkcs/
33. D. Pointcheval, “Chosen-ciphertext security for any one-way cryptosystem”, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 129–146, Springer-Verlag, 2000.
34. C. Rackoff and D. Simon, “Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack”, *Advances in Cryptology – CRYPTO ’91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444, Springer-Verlag, 1992.
35. R.L. Rivest, A. Shamir, and L.M. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, *Communications of the ACM*, **21**(2):120–126, 1978.
36. C.P. Schnorr and M. Jakobsson, “Security of signed ElGamal encryption”, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 73–89, Springer-Verlag, 2000.
37. Y. Tsiounis and M. Yung, “On the security of ElGamal-based encryption”, *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134, Springer-Verlag, 1998.