Some Practical Public-Key Encryption Schemes in both Standard Model and Random Oracle Model

Le Trieu Phong and Ogata Wakaha

May 18, 2006

Abstract

In this paper, we present some more results about the security of the Kurosawa-Desmedt encryption scheme [9] and a variant of it [13]. We prove that after a modification, those schemes are secure against adaptive chosen-ciphertext attack not only under the decisional Diffie-Hellman assumption in standard model as before but also under the computational Diffie-Hellman assumption in the random oracle model. These results ensure that both the Kurosawa-Desmedt scheme and the variant have similar security merits as the Cramer-Shoup encryption scheme [4, 5], which is proposed as a standard [10].
key words: public-key encryption, standard model, ROM, DDH, CDH, IND-CCA

1 Introduction

1.1 Background

The notion of *chosen-ciphertext security* was introduced by Naor and Yung [12] and developed by Rackoff and Simon [14], and Dolev, Dwork, and Naor [6]. This notion is now largely considered as the "right" notion for encryption schemes.

In the random oracle model [3], many practical schemes secure against adaptive chosen-ciphertext attack(IND-CCA) have been proposed: OAEP+ [16], SAEP [1], RSA-OAEP [7] to name just a few.

Although the security analysis in the random oracle model gives us a strong evidence that the schemes are secure, it does not rule out all possible attacks.

In standard model, the first practical public key cryptosystem which is provably IND-CCA secure was discovered by Cramer and Shoup [4]. The security of the scheme is based on the hardness of the decisional Diffie-Hellman(DDH) problem. After that, in [15], Shoup presented a hybrid variant of the Cramer-Shoup cryptosystem. As a hybrid scheme, the variant is very flexible since messages can be arbitrary bit strings.

In [9], Kurosawa and Desmedt modified the hybrid scheme presented in [15], gaining a scheme, called KD scheme for short, which produces shorter ciphertexts and needs less exponentiations than the original one. However, their proof of security relied on the use of information theoretically secure functions KDF(key derivation function) and MAC(message authentication code), which makes the Kurosawa-Desmedt scheme as efficient as the original Cramer-Shoup for typical security parameters, as recently stated in [8]. In that paper, Gennaro and Shoup also presented a different proof of security for Kurosawa-Desmedt scheme, which showed that the scheme can be instantiated with any computationally secure KDF and MAC, thus extended its applicability and efficiency.

In an attempt to sharpen up KD scheme, Phong and Ogata [13] recently modified the scheme, obtaining a more efficient scheme called KD1, which could also be implemented with any computationally secure KDF and MAC. Both KD1 and KD were proved to be IND-CCA secure in standard model, under DDH assumption.

Continuing that approach, we will try to replace the DDH assumption ensuring the security of KD and KD1 schemes by a much weaker assumption, the computational Diffie-Hellman(CDH) assumption. The upcoming results show that the security of KD and KD1 schemes has good merits similar to those of the Cramer-Shoup encryption scheme [4, 5], which is proposed as a standard [10]. The results in this paper together with the results in [13] can also be seen as a complete and affirmative response to the belief raised by Gennaro and Shoup [8] about optimizations and CDH-based security of KD scheme.

1.2 Our Contribution

Our main goal is to show that KD and KD1 schemes possess security merits as a proposed standard [4, 5, 10] does. In particular, we prove that

- Both KD and KD1 are still IND-CCA secure under DDH assumption in both standard and random oracle model after a reasonable modification of their key derivation function KDF.
- Both KD and KD1 are IND-CCA secure under CDH assumption in the random oracle model.

By the results, KD1, which is more efficient, can be used in practice with the same security level as the proposed standard.

The rest of this paper is organized as follows: Section 2 reviews KD and KD1 schemes and their previous security results. Section 3 is about the modification of key deviation KDF and some more security results gained from that modification. Section 4 analyses KD and KD1 in ROM. We finally end up with some discussions in Section 5.

2 Review of KD and KD1 schemes

2.1 Basic Components and Assumptions

Both schemes make use of:

- a group G of prime order q, with (random) generators g_1 and g_2.
  Security assumption(DDH): Hard to distinguish (g_1^r, g_2^r) *from* $(g_1^r, g_2^{r'})$, *where* r *is a random element of* Z_q *and* r' *is a random element of* $Z_q \setminus \{r\}$.
- a message authentication code MAC, which is a function that takes two inputs, a key k and message $e \in \{0,1\}^*$, and produces a "tag" $t := MAC_k(e)$.
  Security assumption: For random k, *after obtaining* $t^* := MAC_k(e^*)$ *for (at most one) adversarially chosen* e^*, *it is infeasible for an adversary* A_{mac} *to compute a forgery pair, i.e., a pair* (e, t) *such that* $e \neq e^*$ *and* $t = MAC_k(e)$.

  Define $Adv_{MAC}(A_{mac}) = \Pr(A_{mac}$ succeeds$)$. The assumption ensures that $Adv_{MAC}(A_{mac})$ is negligible for all polynomial-time adversary A_{mac}.

- a symmetric key encryption scheme, with encryption algorithm E and decryption algorithm D, such that for key K and plaintext $m \in \{0,1\}^*$, $e := E_K(m)$ is the encryption of m under K, and for key K and ciphertext $e \in \{0,1\}^*$, $m := D_K(e)$ is the decryption of e under K.
  Security assumption(semantic security): hard to distinguish $E_K(m_0)$ and $E_K(m_1)$ for randomly chosen K and adversarially chosen m_0 and m_1, where m_0 and m_1 are of equal length.

- a key derivation function KDF, such that for $v \in G$, $KDF(v) = (k, K)$, where k is a message authentication key, and K is a symmetric encryption key.
  Security assumption: hard to distinguish $KDF(v)$ and (k, K), where v, k and K are random.

  Let A_{kdf} be an 0-or-1-output algorithm that takes as input a pair of message authentication key and symmetric encryption key. Define

$$\begin{aligned} Adv_{KDF}(A_{kdf}) \quad &= \quad \Pr[A_{kdf}(KDF(v)) \rightarrow 1] \\ &- \quad \Pr[A_{kdf}(k, K) \rightarrow 1]. \end{aligned}$$

  The assumption ensures that $Adv_{KDF}(A_{kdf})$ is negligible for all polynomial-time adversary A_{kdf}.

- a hash function $H : G \times G \rightarrow Z_q$.
  Security assumption (target collision resistance): given $u_1^* := g_1^r$ and $u_2^* := g_2^r$ for random $r \in Z_q$, hard to find $(u_1, u_2) \in G \times G \setminus \{(u_1^*, u_2^*)\}$ such that $H(u_1, u_2) = H(u_1^*, u_2^*)$.

Note that the key space for the message authentication code is assumed to consist of all bit strings of a given length, so that by a random key k, we mean a random bit string of appropriate length. Similarly for the symmetric encryption keys.

Note also that KDF and H may have associated keys, which are publicly known.

2.2 KD Scheme

KD scheme is described as follows:

Key Generation: The description of the group G is generated, along with random generators g_1 and g_2 for G. Any keys for KDF and H

are also generated. Then,

$$x_1, x_2, y_1, y_2 \xleftarrow{\$} Z_q, c \leftarrow g_1^{x_1} g_2^{x_2}, d \leftarrow g_1^{y_1} g_2^{y_2}.$$

The public key consists of the description of G, the generators g_1 and g_2, keys for KDF and H (if any), along with the group elements c and d. The private key consists of the public key, along with x_1, x_2, y_1, y_2.

Encryption of $m \in \{0,1\}^*$:

$r \xleftarrow{\$} Z_q$, $u_1 \leftarrow g_1^r \in G$, $u_2 \leftarrow g_2^r \in G$, $\alpha \leftarrow H(u_1, u_2) \in Z_q$, $v \leftarrow c^r d^{r\alpha} \in G$, $(k, K) \leftarrow KDF(v)$, $e \leftarrow E_K(m)$, $t \leftarrow MAC_k(e)$.

return $C := (u_1, u_2, e, t)$

Decryption of $C = (u_1, u_2, e, t)$:

$\alpha \leftarrow H(u_1, u_2)$, $v \leftarrow u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} \in G$, $(k, K) \leftarrow KDF(v)$.

If $t \neq MAC_k(e)$ then return `reject`

Else return $m \leftarrow D_K(e)$

2.3 KD1 Scheme

The public key and the encryption algorithm of both KD and KD1 are exactly identical. However, the key generation algorithm and the decryption algorithm of KD1 are different from those of KD, and are described as follows:

Key Generation: The description of the group G is generated, along with *one* random generators g_1 for G. Any keys for KDF and H are also generated. Then,

$$\omega \xleftarrow{\$} Z_q^*, g_2 \leftarrow g_1^{\omega}; x, y \xleftarrow{\$} Z_q, c \leftarrow g_1^x, d \leftarrow g_1^y.$$

The public key consists of the description of G, the generators g_1 and g_2, keys for KDF and H (if any), along with the group elements c and d. The private key consists of the public key, along with ω, x, y.

Decryption of $C = (u_1, u_2, e, t)$:

$\alpha \leftarrow H(u_1, u_2), v \leftarrow u_1^{x+y\alpha} \in G$, $(k, K) \leftarrow KDF(v)$.

If $u_2 \neq u_1^{\omega}$ or $t \neq MAC_k(e)$ then return `reject`

Else return $m \leftarrow D_K(e)$

2.4 Security of KD and KD1

The security of KD and KD1 is ensured by the following theorems.

Theorem 1 (Gennaro, Shoup [8]). *KD scheme is IND-CCA secure in standard model if the assumptions on its components described in the previous section hold. In particular,* $Adv_{KD}^{ind-cca}(A)$ *is negligible for all probabilistic, polynomial-time adversary A.*

Theorem 2 (Phong, Ogata [13]). *KD1 scheme is IND-CCA secure under DDH assumption in standard model.*
In particular, for all probabilistic, polynomial-time adversary A, there exist algorithms A_1 *and* A_2 *whose resources are essentially the same as those of A such that*

$$|Adv_{KD1}^{ind-cca}(A) - Adv_{KD}^{ind-cca}(A)| \leq Q_A(Adv_{KDF}(A_1) + Adv_{MAC}(A_2)),$$

where Q_A *is the number of decryption queries made by A.*

The proof of Theorem 2 is shown in Appendix C.

3 Modification of Key Deviation Function

The key deviation function KDF is modulated as follows: the input of it is now *two* elements from the group G instead of one as before, i.e., for $v, u_1 \in G$, $KDF(v, u_1) = (k, K)$, where k is a message authentication key, and K is a symmetric encryption key. This KDF is required to satisfy a new security assumption as follows:

New security assumption on KDF: *For all* $u_1 \in G$*, it is infeasible to distinguish* $KDF(v, u_1)$ *and* (k, K)*, where* v, k, K *are random.*

There are many constructions for the above KDF. One may use a pairwise independent hash function together with the Leftover Hash Lemma for constructing that KDF [5]. However, the construction is not quite efficient, since the order of the group G needs to be large, so that long enough keys can be generated(e.g., G should be at least of order 512 to generate keys (k, K) of 256 bit length). Therefore, in practice, we can simply use a dedicated cryptographic hash function, like SHA1, for KDF. We refer readers to [10] for the concrete construction of KDF using hash functions. From now on, KDF is considered as a hash function in this paper, so we can model it as a random oracle when needed.

Theorem 1 and Theorem 2 remain valid after the modification of KDF. In fact, what these theorems's security analysis(in [8, 13])

requires is that the output of KDF is indistinguishable from random keys (k, K) if v is random, which is the previous security assumption of KDF in Section 2.1. The requirement is clearly fulfilled by the new security assumption on KDF.
We can gain more from the modification of KDF. In particular, some results similar to Theorem 1 and Theorem 2 can be obtained if KDF is modeled as a random oracle. We first state those results.

Theorem 1^{rom}**.** *KD scheme is IND-CCA secure under DDH assumption in ROM. In particular, for all probabilistic, polynomial-time adversary A, the advantage* $Adv_{KD}^{ind-cca}(A)$ *is negligible if KDF is considered as a random oracle.*

Theorem 2^{rom}**.** *KD1 scheme is IND-CCA secure under DDH assumption, if KDF is modeled as a random oracle.*
In particular, for all probabilistic, polynomial-time adversary A, there exists algorithm A' *whose resources is essentially the same as those of A such that*

$$|Adv_{KD1}^{ind-cca}(A) - Adv_{KD}^{ind-cca}(A)| \leq Q_D(\frac{Q_R}{|G|} + Adv_{MAC}(A')),$$

where Q_D *and* Q_R *are respectively the number of decryption queries and random oracle queries made by A.*

Sketch of Proof. We first revise the proofs of Theorem 1 [8] and Theorem 2 [13] and then show how to adapt the proofs to ROM. Those proofs employed "game-playing technique", which changed the IND-CCA attack game into a final easy-to-deal-with game using many intermediate ones. Some games utilized the indistinguishability between the output of KDF and random bit strings in "$KeySpace$" as follows:

Game i	**Game** $i+1$
$\cdots$	$\cdots$
$v \xleftarrow{\$} G$	$v \xleftarrow{\$} G$
$(k, K) \leftarrow KDF(v, u_1)$	$(k, K) \leftarrow$ "$KeySpace$"
$\cdots$	$\cdots$

In standard model, Game i and Game $i+1$, which are Game $3'$ and Game $4'$ in Appendix C, are the same with overwhelming probability, $1 - \epsilon_{kdf}$, where ϵ_{kdf} is the probability that one can distinguish $KDF(v, u_1)$ and (k, K) where v, k, K are random. Now, in ROM, if KDF is modeled as a random oracle, it behaves as in Algorithm 1 to evaluate random oracle query (v, u_1).

Algorithm 1 Implemetation of random oracle KDF

1: **if** $(v, u_1, k, K) \in TableKDF$ for some k, K **then**
2: **return** (k, K)
3: **else**
4: $(k, K) \xleftarrow{\$}$ "$KeySpace$"
5: Add (v, u_1, k, K) to TableKDF
6: **return** (k, K)
7: **end if**

In Algorithm 1, TableKDF, initially empty, is a data structure that records all random oracle queries which have been asked before. Note that the number of elements in TableKDF is always less than Q_R, which is the total number of random oracle queries.

In ROM, Game $i+1$ may be different from Game i if the value v in Game $i+1$ has been in some previous random oracle query. The probability that this event occurs is less than $\frac{Q_R}{|G|}$ since v (in both games) is uniformly distributed over G. Thus Game i and Game $i+1$ are also the same with overwhelming probability in ROM as in standard model. This fact makes the proofs of Theorem 1 and Theorem 2 can be reused in ROM, resulting in Theorem 1^{rom} and Theorem 2^{rom}. □

4 Security Analysis of KD1 and KD in ROM

Theorem 3. *Both KD and KD1 schemes are IND-CCA secure under CDH assumption in the random oracle model.*

Proof. Since Theorem 2^{rom} ensures that the advantages of an adversary against KD and KD1 are almost the same, it is sufficient to prove that KD1 scheme is IND-CCA secure under CDH assumption when $KDF(\cdot, \cdot)$ is modeled as a random oracle.

Suppose that KD1 is IND-CCA insecure in G, i.e., there is a polynomial-time adversary A with non-negligible probability in the attack game, we will build a polynomial-time algorithm to solve CDH problem in G. Since KD1 is supposed to be insecure, Theorem 2^{rom} also provides us with a polynomial-time algorithm solving DDH problem in G, called DHP algorithm for short. We will use the adversary

A together with this DHP algorithm to build an efficient algorithm B solving CDH problem in G.

The input of algorithm B is a random triple $(g_1, u_1^*, \lambda) \in G^3$, and its output must be non-negligibly a value $v \in G$ such that $DHP(g_1, u_1^*, \lambda, v) = 1$. Algorithm B runs the adversary A as a sub-routine and also simulates the environment for A as follows:

- Public-key simulation:
  - $\omega \stackrel{\$}{\leftarrow} Z_q, g_2 \leftarrow g_1^{\omega}, u_2^* \leftarrow u_1^{*\omega}$
  - Key(if any) for H is generated, and $\alpha^* \leftarrow H(u_1^*, u_2^*)$
  - $y \stackrel{\$}{\leftarrow} Z_q, d \leftarrow g_1^y$, $c \leftarrow \lambda d^{-\alpha^*}$

  B gives the values $pk = (g_1, g_2, c, d, \text{Key of H})$ to A as the public key. Note that the simulated public key has the same probabilistic distribution as the "real" public key.
- Encryption oracle simulation: algorithm B processes the encryption oracle query (m_0, m_1) as follows:
  - $(k^*, K^*) \stackrel{\$}{\leftarrow}$ "$KeySpace$", $b \stackrel{\$}{\leftarrow} \{0, 1\}$
  - $e^* \leftarrow E_{K^*}(m_b)$, $t^* \leftarrow MAC_{k^*}(e^*)$

  B gives the computed values $C^* = (u_1^*, u_2^*, e^*, t^*)$ to A as the target ciphertext.
  It is worth noting that, in this simulated environment for A, the symmetric key K^* is truly random and is not used elsewhere except for encrypting m_b. This fact implies that the advantage of A in the simulated environment is equal to advantage of an efficient algorithm A_{enc} distinguishing $E_{K^*}(m_0)$ and $E_{K^*}(m_1)$. Denoting the advantage of A_{enc} by ϵ_{enc}, which is negligible by assumption, we have

$$Adv_{KD1}^{ind-cca}(A^S) \quad = \quad \epsilon_{enc}, \tag{1}$$

  where A^S means the adversary A in the simulated environment. Of course, equation (1) can only be accepted if the random oracle KDF and the decryption oracle of A are successfully simulated. The goal of algorithm B now is to simulate those oracles. In order to do that, B utilizes initially empty data structures $\mathcal{V}_1$, $\mathcal{V}_2$, $\mathcal{V}_3$, where
  - $\mathcal{V}_1$ is the set of all (v, u_1, k, K) for which B has assigned the keys (k, K) to $KDF(v, u_1)$. The manner an element is added to $\mathcal{V}_1$ is in Algorithm 2.

- $\mathcal{V}_2$ is the set of all (v, u_1) for which $(v, u_1, k, K) \in \mathcal{V}_1$ for some (k, K) and $DHP(g_1, u_1, cd^{H(u_1, u_1^\omega)}, v) = 1$. The manner an element is added to $\mathcal{V}_2$ is also in Algorithm 2.
- $\mathcal{V}_3$ is the set of all (u_1, u_2, k, K) which is added to $\mathcal{V}_3$ in Algorithm 3 for simulating the decryption oracle.

• Random Oracle Simulation: The simulation is given in Algorithm 2. It can be checked by inspection that this algorithm returns random keys (k, K) to A if (v, u_1) has not been queried before, and returns the same value if the query is identical, except some cases. These cases is very important and will be considered later.

Algorithm 2 Simulation of random oracle KDF with input (v, u_1)

$u_2 \leftarrow u_1^\omega$, $\alpha \leftarrow H(u_1, u_2)$
if $(v, u_1, k, K) \in \mathcal{V}_1$ for some (k, K) **then**
    return (k, K) to A
else if $DHP(g_1, u_1, cd^\alpha, v) = 0$ **then**
    $(k, K) \xleftarrow{\$}$ "$KeySpace$"
    Add (v, u_1, k, K) to $\mathcal{V}_1$
    return (k, K) to A
else
    if $u_1 = u_1^*$ **then**
        B halts and outputs v ▷ F_1 occurs
    else if $\alpha = \alpha^*$ **then**
        Abort the game ▷ F_2 occurs
    else ▷ $\alpha \neq \alpha^* \Rightarrow v \neq v^*$
        if $(u_1, u_2, k, K) \in \mathcal{V}_3$ for some (k, K) **then**
            return (k, K) to A
        else
            $(k, K) \xleftarrow{\$}$ "$KeySpace$
            Add (v, u_1, k, K) to $\mathcal{V}_1$
            Add (v, u_1) to $\mathcal{V}_2$
            return (k, K) to A
        end if
    end if
end if

- Decryption Oracle Simulation: The simulation is given in Algorithm 3. It may need some elaborations. In this algorithm, B do not know the input value v for $KDF(\cdot,\cdot)$, but it do know that v here must satisfy $DHP(u_1, cd^{H(u_1,u_2)}, v) = 1$. Thus the value v is surely determined by the input values u_1, u_2, and hence we can define $KDF(v, u_1)$ as in line 3 of Algorithm 3. The set $\mathcal{V}_3$ is to keep the defined value of $KDF(v, u_1)$.

Algorithm 3 Simulation of decryption oracle, with input (u_1, u_2, e, t)

1: **if** $(u_1, u_2, k, K) \in \mathcal{V}_3$ for some (k, K) **then**
2: **if** $u_2 \neq u_1^{\omega}$ or $t \neq MAC_k(e)$ **then**
3: **return** `reject`
4: **else**
5: **return** $m \leftarrow D_K(e)$
6: **end if**
7: **else**
8: **if** $(v, u_1, k, K) \in \mathcal{V}_1$ for some v, k, K **then**
9: **if** $u_2 \neq u_1^{\omega}$ or $t \neq MAC_k(e)$ **then**
10: **return** `reject`
11: **else**
12: **return** $m \leftarrow D_K(e)$
13: **end if**
14: **else**
15: $(k, K) \xleftarrow{\$}$ "$KeySpace$" ▷ defining $KDF(v, u_1)$
16: Add (u_1, u_2, k, K) to $\mathcal{V}_3$
17: **if** $u_2 \neq u_1^{\omega}$ or $t \neq MAC_k(e)$ **then**
18: **return** `reject`
19: **else**
20: **return** $m \leftarrow D_K(e)$
21: **end if**
22: **end if**
23: **end if**

Let F_1, F_2 be respectively the event that line 10 and line 12 in Algorithm 2 are executed. Note that the environment of A given by B and the actual environment in the IND-CCA attack game are the same

until $F_1 \vee F_2$ occurs. Thus, by the Difference Lemma(Appendix 1),

$$\begin{aligned} |Adv_{KD1}^{ind-cca}(A^R) - Adv_{KD1}^{ind-cca}(A^S)| &\leq \Pr[F_1 \vee F_2] \\ &\leq \Pr[F_1] + \Pr[F_2], \quad (2) \end{aligned}$$

where A^R is the adversary A in the real environment, which is the environment of the IND-CCA attack game.

The event F_1 occurs if and only if B outputs a value v such that

$$DHP(g_1, u_1^*, cd^{\alpha^*}, v) = 1.$$

Note that $cd^{\alpha^*} = \lambda$, and thus F_1 occurs if and only if B successfully solve the CDH problem in G.
The event F_2 occurs if and only if there is a collision in hash function, $H(u_1, u_2) = H(u_1^*, u_2^*)$ and $(u_1, u_2) \neq (u_1^*, u_2^*)$. Therefore the probability that F_2 occurs is equal to the advantage of an algorithm A_{hash} finding a (target) collision in hash function H while using substantially the same resources as A. Denote the advantage of A_{hash} by ϵ_{hash}, which is negligible by assumption, we have

$$\Pr[F_2] = \epsilon_{hash} \quad (3)$$

Recall that $Adv_{KD1}^{ind-cca}(A^R)$ is assumed to be non-negligible. Moreover, by (1), (2), and (3),

$$\Pr[F_1] \geq |Adv_{KD1}^{ind-cca}(A^R) - \epsilon_{enc}| - \epsilon_{hash}$$

The above inequality implies that B successfully solves the CDH problem with non-negligible probability, which concludes the proof. □

5 Discussions

The reduction in Theorem 3 is not very efficient as desired, since many DDH instances need to be solved in order to solve just one CDH instance. In particular, let T be the running time of DHP, then the running time of B is essentially that of A plus $O(T \cdot Q_R)$, where Q_R is the total number of random oracle queries.

However, that fact does not have much sway on the IND-CCA security of both KD and KD1. In fact, in certain group G, like Gap Diffie-Hellman group [11] where DDH is easy, while CDH remains hard, we really have a fast DHP, and the reduction in Theorem 3

become quite reasonable. On the other hand, if DDH is hard in G, the security of KD and KD1 is still ensured by Theorem 1 and 2.

In a word, both KD and KD1 have good merits such as the Cramer-Shoup cryptosystem [4, 5], which is proposed as a standard [10]. Since KD, KD1 and the Cramer-Shoup cryptosystem are quite similar, the designed components used in the proposed standard can be used in KD and KD1 as well. In practice, KD1, which is more efficient, and the proposed standard, can be used interchangeably with the same level of security.

References

[1] D. Boneh. Simplified OAEP for the RSA and Rabin Functions. CRYPTO 2001, pages 275-291, 2001

[2] M. Bellare and P. Rogaway. The Game-Playing Technique and its Applications to Triple Encryption. Draft 2.0, 2005. Available at http://eprint.iacr.org/2004/331.pdf

[3] M. Bellare and P. Rogaway, Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.ACM Conference on Computer and Communications Security, pages 62-73, 1993

[4] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. CRYPTO'98, pages 13-25, 1998

[5] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack, SIAM Jounal of Computing 33, pages 167-226, 2003

[6] D. Dolev, C.Dwork, and M. Naor. Non-malleable cryptography. STOC'91, pages 542-552, 1991

[7] E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern. RSA-OAEP Is Secure under the RSA Assumption. CRYPTO 2001, pages 260-274, 2001

[8] R. Gennaro and V. Shoup. A Note on an Encryption Scheme of Kurosawa and Desmedt. manuscript, 2005. Available at http://www.shoup.net

[9] K.Kurosawa and Y.Desmedt. A New Paradigm of Hybrid Encryption Scheme. CRYPTO'04, pages 426-442, 2004

[10] ISO 18033-2: An Emerging Standard for Public-Key Encryption. Final Committee Draft of December 6, 2004. Available at www.shoup.net

[11] A. Joux and K. Nguyen, Separating Decision Die-Hellman from Die-Hellman in Cryptographic Groups," Cryptology ePrint Archive, Report 2001/003, available at http://eprint.iacr.org/2001/003/

[12] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In STOC'90, 1990

[13] L. T. Phong and W. Ogata. On a Variation of Kurosawa-Desmedt Encryption Scheme. Cryptology ePrint Archive, Report 2006/031, 2006, `http://eprint.iacr.org/`

[14] C. Rackoff and D. Simon. Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. In CRYPTO'91, pages 433-444, 1991

[15] V. Shoup. Using Hash Function as a Hedge Against Chosen Ciphertext Attack. In EuroCrypt'00, pages 275-288, 2000

[16] V. Shoup. OAEP Reconsidered. CRYPTO 2001, pp. 239-259, 2001

[17] V. Shoup. Sequences of Games: a Tool for Taming Complexity in Security Proofs. manuscript, 2005. Available at www.shoup.net

A Security against Adaptive Chosen Ciphertext Attack

For the readers' convenience, we recall here the definition of IND-CCA security. The definition used here is essentially the same as the one in [15]. The attack scenario, which is also called (IND-CCA) attack game, has four stages as follows.

First, the key generation algorithm is run, generating the public key and private key for the cryptosystem. The adversary A, of course, obtains the public key, but not the private key.

Second, the adversary makes a series of arbitrary queries to a *decryption oracle*. Each query is a ciphertext C that is decrypted by the decryption oracle, making use of the private key of the cryptosystem. The resulting decryption is given to the adversary. The adversary is

free to construct the ciphertexts in an arbitrary way, namely it is *not* required to compute them using the encryption algorithm.

Third, the adversary prepares two messages m_0 and m_1 and gives these to an *encryption oracle*. The encryption oracle chooses $b \in \{0,1\}$ at random, encrypts m_b, and gives the resulting "target" ciphertext C^* to the adversary. The adversary is free to choose m_0 and m_1 in an arbitrary way, except that if message lengths are not fixed by the cryptosystem, then these two messages must nevertheless be of the same length.

Fourth, the adversary continues to submit ciphertexts C to the decryption oracle, subject only to the restriction that $C \neq C^*$.

Just before the adversary terminates, it outputs $\hat{b} \in \{0,1\}$, representing its "guess" of b.

The adversary's advantage in this attack scenario is defined to be the distance from 1/2 of the probability that $\hat{b} = b$, i.e.,

$$Adv^{ind-cca}(A) = |\Pr[\hat{b} = b] - \frac{1}{2}|.$$

A cryptosystem is defined to be *IND-CCA secure* if for any efficient adversary A, the value $Adv^{ind-cca}(A)$ is negligible.

B The Difference Lemma

We state here a useful lemma, which is called the *Difference Lemma*[1] (see [17]).

Lemma 1 (Difference Lemma)**.** *Let A, B, F be events defined in some probability distribution, and suppose that $A \wedge \overline{F} \iff B \wedge \overline{F}$. Then*

$$|\Pr[A] - \Pr[B]| \leq \Pr[F].$$

This lemma is used intensively in "game" technique, which will be used to analyze the security of the upcoming schemes.

C Proof of Theorem 2 [13]

Proof. Consider a probabilistic, polynomial-time adversary A. We will begin with the game used to define IND-CCA security of *KD* scheme.

[1]This lemma is called "The Fundamental Lemma of Game-Playing" in [2]

Game 0 (attack game on KD scheme)
This game is an interactive computation between A and a simulator. Initially, the simulator runs the key generation algorithm of KD scheme, obtaining the description of G, generators g_1, g_2, keys for KDF and H (if any), along with the values $x_1, x_2, y_1, y_2 \in Z_q$ and $c, d \in G$. The simulator gives the public key to A.

During the execution of the game, the adversary A makes a number of "decryption requests." Assume these requests are $C^{(1)}, \ldots, C^{(Q_A)}$, where

$$C^{(i)} = (u_1^{(i)}, u_2^{(i)}, e^{(i)}, t^{(i)}).$$

For each such request, the simulator decrypts the given ciphertext, and gives A the result. We denote by $\alpha^{(i)}$, $v^{(i)}$, $k^{(i)}$, $K^{(i)}$ the corresponding intermediate quantities computed by the decryption algorithm on input $C^{(i)}$.

The adversary may also make a single "challenge request." For such a request, the adversary submits two messages m_0, m_1 of equal length bit strings to the simulator; the simulator chooses $b \in \{0, 1\}$ at random, and encrypts m_b using the encryption algorithm of KD scheme, obtaining the "target ciphertext" $C^* = (u_1^*, u_2^*, e^*, t^*)$.

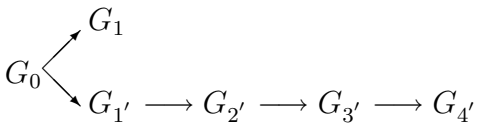
The only restriction on the adversary's requests is that after the "challenge request", subsequent decryption requests must not be the same as the target ciphertext.

At the end of the game, the adversary outputs $\hat{b} \in \{0, 1\}$.

Let T_0 be the event that $\hat{b} = b$ in this game. Advantage of the adversary with respect to KD scheme is defined as

$$Adv_{KD}^{ind-cca}(A) = |\Pr[T_0] - \frac{1}{2}|.$$

We will consider other games, which are slightly-modified versions of Game 0. The games will be built in the order as follows:



All those games are viewed as operating on the same underlying probability space, i.e., all the random variables `Coins`(of A), ω (in Game 1), x_1, x_2, y_1, y_2, r^* (for encrypting m_b), b take the same value in those games.

Game 1
This game is the same as Game 0, except that

- instead of being randomly chosen from G, the generator g_2 is generated as
$$\omega \stackrel{\$}{\leftarrow} Z_q, g_2 \leftarrow g_1^{\omega}.$$
- the simulator not only chooses x_1, x_2, y_1, y_2 randomly from Z_q but also puts $x := x_1 + \omega x_2$ and $y := y_1 + \omega y_2$. The values c and d are now computed as
$$c \leftarrow g_1^x, d \leftarrow g_1^y.$$
- in processing a decryption request $C = (u_1, u_2, e, t)$, the simulator proceeds as follows
  - $\alpha \leftarrow H(u_1, u_2), v \leftarrow u_1^{x+\alpha y}$
  - $(k, K) \leftarrow KDF(v)$
  - Test if $u_2 = u_1^{\omega}$ and $t = MAC_k(e)$; if this is not the case, then return `reject` .
  - Return $m \leftarrow D_K(e)$

It is obvious that Game 1 is really the attack game of A against KD1. Let T_1 be the event that $\hat{b} = b$ in this game. Thus the advantage of A with respect to KD1 scheme is

$$Adv_{KD1}^{ind-cca}(A) = |\Pr[T_1] - \frac{1}{2}|.$$

Therefore, in order to bound $|Adv_{KD1}^{ind-cca}(A) - Adv_{KD}^{ind-cca}(A)|$, it is sufficient to bound $|\Pr[T_1] - \Pr[T_0]|$.

Let F_1 be the event that some ciphertext is rejected in Game 1 by the decryption oracle, but would have passed the test of the decryption algorithm in Game 0. Game 0 and Game 1 are then the same until F_1 occurs. Thus $T_1 \wedge \overline{F_1}$ and $T_0 \wedge \overline{F_1}$ are identical. By the Difference Lemma,

$$|\Pr[T_1] - \Pr[T_0]| \leq \Pr[F_1].$$

We will use the below games $G_{1'}$, $G_{2'}$, $G_{3'}$, $G_{4'}$ to bound $\Pr[F_1]$.

Game $1'$

This game is the same as Game 0, except that

- instead of being randomly chosen from G, g_2 is generated by
$$\omega \stackrel{\$}{\leftarrow} Z_q, g_2 \leftarrow g_1^{\omega}.$$

- in processing a decryption request $C = (u_1, u_2, e, t)$, the simulator proceeds as follows
  - $\alpha \leftarrow H(u_1, u_2), v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$
  - $(k, K) \leftarrow KDF(v)$
  - Test if $u_2 = u_1^{\omega}$ and $t = MAC_k(e)$; if this is not the case, then return `reject` .
  - Return $m \leftarrow D_K(e)$

Note that Game 1 and Game $1^{'}$ are exactly identical from the viewpoint of the adversary A. In fact, if A submits a ciphertext $C = (u_1, u_2, e, t)$ with $u_2 \neq u_1^{\omega}$, then A will receive `reject` in the both games. On the other hand, if $u_2 = u_1^{\omega}$, then the value v in Game 1 and Game $1^{'}$ is the same, which ensures that the ciphertext is identically decrypted in those games. Thus a ciphertext is rejected in Game $1^{'}$ if and only if it is rejected in Game 1, and hence

$$\Pr[F_1] = \Pr[F_{1'}],$$

where $F_{1'}$ be the event that some ciphertext is rejected in Game $1^{'}$, but would have passed Game 0.

Let $F_{1'}^{(j)}$ be the event that the j^{th} ciphertext $C^{(j)}$ is rejected in Game $1^{'}$, but would have passed the test in Game 0. Then

$$\Pr[F_{1'}] \leq Q_A \max_{1 \leq j \leq Q_A} \{\Pr[F_{1'}^{(j)}]\}.$$

Note that $F_{1'}^{(j)}$ occurs if and only if $u_2^{(j)} \neq (u_1^{(j)})^{\omega}$ and $t^{(j)} = MAC_{k^{(j)}}(e^{(j)})$, where $k^{(j)}$ is the first part of $KDF((u_1^{(j)})^{x_1+y_1\alpha^{(j)}} (u_2^{(j)})^{x_2+y_2\alpha^{(j)}})$. Our task now is to bound $\Pr[F_{1'}^{(j)}]$.

Game $2^{'}$

This game is the same as Game $1^{'}$, except that the simulator now proceeds a decryption request $C = (u_1, u_2, e, t)$ as follows

$D01^{'}$: $\alpha \leftarrow H(u_1, u_2)$
$D02^{'}$: If $u_2 \neq u_1^{\omega}$ then
$D03^{'}$: $v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$
$D04^{'}$: $(k, K) \leftarrow KDF(v)$
$D05^{'}$: Test if $t = MAC_k(e)$; if this is not the case, then return `reject` .

$D06'$: $m \leftarrow D_K(e)$. Return `reject`
$D07'$: Else
$D08'$: $v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$
$D09'$: $(k, K) \leftarrow KDF(v)$
$D10'$: Test if $t = MAC_k(e)$; if this is not the case, then return `reject` .
$D11'$: $m \leftarrow D_K(e)$. Return m.

The change, which is purely conceptual, is that the simulator now considers two cases, $u_2 \neq u_1^{\omega}$ and $u_2 = u_1^{\omega}$ in decryption. Note that whenever line $D03'$ is reached, then the output is always `reject` . Moreover, $\Pr[F_{1'}^{(j)}] = \Pr[F_{2'}^{(j)}]$, where $F_{2'}^{(j)}$ is the event that line $D06'$ is executed in the j^{th} decryption request.

Game $3'$
This game is the same as Game $2'$, except that we change line $D03'$ as follows

$D03'$: $v \xleftarrow{\$} G$

Let $F_{3'}^{(j)}$ be the event that line $D06'$ is executed in the j^{th} decryption request in Game $3'$. We claim that $F_{3'}^{(j)} = F_{2'}^{(j)}$. This follows from the fact that $v = u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$, $c = g_1^{x_1+\omega x_2}$, and $d = g_1^{y_1+\omega y_2}$ are mutually independent and uniformly distributed over G if $u_2 \neq u_1^{\omega}$. In fact, if we put

$$\begin{aligned} r_1 &:= \log_{g_1} u_1, \\ r_2 &:= \log_{g_2} u_2, \end{aligned}$$

then the condition $u_2 \neq u_1^{\omega}$ implies $r_1 \neq r_2$, so the following values

$$\begin{aligned} \log_{g_1} c &= x_1 + \omega x_2, \\ \log_{g_1} d &= y_1 + \omega y_2, \\ \log_{g_1} v &= r_1(x_1 + \alpha y_1) + r_2(x_2 + \alpha y_2), \end{aligned}$$

are linearly independent. This means that v can take any value over G. Thus Game $3'$ and Game $2'$ are identical, and hence $F_{3'}^{(j)} = F_{2'}^{(j)}$.

Game $4'$
This game is the same as Game $3'$, except that we change line $D04'$ as follows

$D04^{'}$: $(k, K) \stackrel{\$}{\leftarrow}$ "$KeySpace$"

Let $F_{4'}^{(j)}$ be the event that line $D06^{'}$ is executed in the ^{jth} decryption request in Game $4^{'}$. It is clear that we can build an algorithm A_1, using similar resources to those of A, such that

$$|\Pr[F_{3'}^{(j)}] - \Pr[F_{4'}^{(j)}]| \leq Adv_{KDF}(A_1).$$

Note that the key k of the message authentication code in this game is completely random and is not used anywhere except as input for MAC. Thus, the probability that line $D06^{'}$ is executed in the j^{th} decryption request must be less than the probability that an algorithm A_2 can break the MAC, i.e.,

$$\Pr[F_{4'}^{(j)}] \leq Adv_{MAC}(A_2).$$

In fact, A_2 just employs A and its simulator, and returns (e, t) at $D05^{'}$ in Game $4^{'}$ whenever line $D06^{'}$ is executed. Summing up,

$$\begin{aligned}
\Pr[F_1] &= \Pr[F_{1'}] \\
&\leq Q_A \max_{1 \leq j \leq Q_A} \{\Pr[F_{1'}^{(j)}]\} \\
&= Q_A \max_{1 \leq j \leq Q_A} \{\Pr[F_{2'}^{(j)}]\} \\
&= Q_A \max_{1 \leq j \leq Q_A} \{\Pr[F_{3'}^{(j)}]\} \\
&\leq Q_A(Adv_{KDF}(A_1) + \max_{1 \leq j \leq Q_A} \{\Pr[F_{4'}^{(j)}]\}) \\
&\leq Q_A(Adv_{KDF}(A_1) + Adv_{MAC}(A_2)),
\end{aligned}$$

which completes the proof. □