A New Variant of the Cramer-Shoup KEM Secure against Chosen Ciphertext Attack

Joonsang Baek[1] Willy Susilo[2] Joseph K. Liu[1]
Jianying Zhou[1]

[1]Institute for Infocomm Research, Singapore
[2]University of Wollongong, Australia

Abstract

We propose a new variant of the Cramer-Shoup KEM (key encapsulation mechanism). The proposed variant is more efficient than the original Cramer-Shoup KEM scheme in terms of public key size and encapsulation cost, but is proven to be (still) secure against chosen ciphertext attack in the standard model, relative to the Decisional Diffie-Hellman problem.

1 Introduction

Motivation. At Crypto '98, Cramer and Shoup [7] proposed the first practical public key encryption (PKE) scheme whose security against adaptive chosen ciphertext attack (which we simply call "CCA" throughout this paper) can be proven without depending on the random oracle model [5]. This is a striking result as the chosen ciphertext security without random oracles could be achieved by only adding a few more exponentiations to the original ElGamal encryption scheme, in contrast to the computationally heavy solutions [9, 18] based on zero-knowledge proofs proposed before. Nearly seven years later, a major improvement on the performance of the Cramer-Shoup PKE scheme was made by Kurosawa and Desmedt [15]. They were able to obtain a very efficient hybrid PKE scheme by simplifying the Cramer-Shoup PKE scheme with the help of the "ciphertext authenticity checking" mechanism of the underlying symmetric encryption primitive. Afterwards, Hofheinz and Kiltz [12] came up with a dual version of the Kurosawa-Desmedt PKE scheme. Note that chosen ciphertext security of all these schemes are relative to the Decisional Diffie-Hellman (DDH) problem.

In the full version of their Crypto '98 paper, Cramer and Shoup [8] formulated a framework called "KEM/DEM (Key Encapsulation Mechanism/Data Encapsulation Mechanism)". A KEM is a public key scheme that outputs a (session) key taking public key as input. According to the KEM/DEM framework, a (hybrid) PKE scheme secure against CCA can be constructed in such a way that a key output by a CCA-secure KEM scheme[1] is used as a session key for an one-time CCA-secure DEM (i.e., symmetric encryption) scheme that encrypts a plaintext message.

[1]The CCA security notion for KEM will be defined in Section 2.

In the same paper, Cramer and Shoup proposed a KEM scheme based on the their original PKE scheme, which we denote by "CS-KEM", and showed it is CCA-secure assuming that the DDH problem is hard. Interestingly, however, it was shown [11] that the KEM scheme extracted from Kurosawa and Desmedt's hybrid PKE scheme, which we denote by "KD-KEM", does not satisfy full CCA-security even though the hybrid PKE scheme remains secure against CCA. Abe et al. [1] showed later that the KD-KEM scheme is actually secure against "LCCA (predicate-dependent CCA)" which is weaker than usual CCA-security of KEM. Similarly, the KEM scheme extracted from Hofheinz and Kilt's [12] hybrid PKE scheme, denoted "HK-KEM", was shown to be secure against CCCA (constrained CCA), which is also weaker than the usual CCA-security of KEM.

Hence, the CS-KEM scheme is, though less efficient than the KD-KEM and HK-KEM schemes, the only KEM scheme that is fully CCA-secure without random oracles, relative to the DDH problem. A remaining question is whether the performance of the CS-KEM scheme can be further improved. In this paper, we give a positive answer to this question.

Recent Developments. In 2007, Kiltz [14] proposed a KEM scheme whose CCA security is based on the Gap Hashed Diffie-Hellman (GHDH) problem. An interesting feature of this scheme is that different from the CS-KEM scheme, a key can be computed from one of the public key components used to create one ciphertext component. More precisely, let $pk = (q, g, c, d)$ be public key, where g is a generator of a group of prime order q; $c = g^x$ and $d = g^y$ for some random $(x, y) \in \mathbb{Z}_q^*$. In this scheme, a ciphertext and its corresponding key is computed as $(g^r, (c^\alpha d)^r)$ and $\mathsf{KDF}(c^r)$ respectively, where KDF denotes a key derivation function. As mentioned earlier, the public c used to create $(c^\alpha d)^r$ is reused to produce c^r. Note here that one cannot expect a computational gain even if c is reused. However, if d were reused, a computational cost could be reduced by computing $c^{r\alpha}$ and d^r separately to generate $(c^\alpha d)^r$ and using d^r as a key. Indeed, Lu et al. [16] recently showed that this modified version of Kiltz's KEM scheme is CCA-secure.

Our Contributions. We observe that it is also possible to apply the structure of Kiltz's KEM scheme to the CS-KEM scheme. As a result, we could construct a KEM scheme which is proven to be *fully* CCA-secure without random oracles assuming that the DDH problem is hard, but is more efficient than the CS-KEM scheme. The efficiency comes as a shorter public/private key pair and improved encapsulation speed. However, we honestly state that the improvement on the encapsulation speed would not be very much dramatic due to the advancement of fast multi-exponentiation algorithms [2, 6, 17], which makes the cost for computing double exponentiation very close to the cost of computing a single exponentiation. Nevertheless, the proposed scheme has a new structure, which reduces one group element of the public key of the CS-KEM scheme. We believe it is also theoretically interesting in that it shows yet another way of constructing a more efficient variant of the CS-KEM without sacrificing full CCA-security in contrast to the results of [15] and [12].

2 Preliminaries

In this section, we review the formal notion of key encapsulation mechanism (KEM) and its security against adaptive chosen ciphertext attack (CCA). We also review building blocks used in our construction of KEM which will be presented in Section 3.

Key Encapsulation Mechanism (KEM). The KEM scheme, denoted KEM, consists of the following algorithms [8, 20, 13].

- Key Generation: Taking 1^λ for a security parameter $\lambda \in \mathbb{Z}_{\geq 0}$ as input, this algorithm generates a public/private key pair (pk, sk).
- Encapsulation: Taking 1^λ and a public key pk as input, this algorithm generates a ciphertext/(symmetric) key pair (ψ, K).
- Decapsulation: Taking 1^λ, a private key sk and a ciphertext ψ as input, this algorithm outputs either a (symmetric) key K or the special symbol $\perp$, meaning "reject".

The security against CCA of KEM is defined as follows. Consider any attacker $\mathcal{A}$ and any value $\lambda > 0$ for security parameter in the following game $\mathsf{GameCCA}_{\mathcal{A}}^{\mathsf{KEM}}(\lambda)$ in which $\mathcal{A}$ interacts with the challenger.

Phase 1: The challenger runs the key generation algorithm providing 1^λ as input to generate a public/private key pair (pk, sk). The challenger then computes a challenge ciphertext ϕ^* and a key K_1^* by running the encapsulation algorithm. It also picks $K_0^* \in S_K$ at random, where S_K denotes the key space. It then picks $\beta \in \{0, 1\}$ at random and gives (pk, ϕ^*, K_β^*) to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ submits ciphertexts, each of which is denoted by ϕ. On receiving ϕ, the challenger runs the decapsulation algorithm on input ϕ and passes the resulting decapsulation to $\mathcal{A}$. At the end of this phase, $\mathcal{A}$ outputs its guess $\beta' \in \{0, 1\}$.

We define the output of the game to be 1 if $\beta' = \beta$, and 0 otherwise. $\mathcal{A}$'s success is defined by the probability

$$\mathbf{Adv}_{\mathcal{A},\mathsf{KEM}}^{\mathrm{CCA}}(\lambda) = \left| \Pr[\mathsf{GameCCA}_{\mathcal{A}}^{\mathsf{KEM}}(\lambda) = 1] - \frac{1}{2} \right|.$$

We say that KEM is CCA-secure if $\mathbf{Adv}_{\mathsf{KEM}}^{\mathrm{CCA}}(\lambda) = \max_{\mathcal{A}} \left\{ \mathbf{Adv}_{\mathcal{A},\mathsf{KEM}}^{\mathrm{CCA}}(\lambda) \right\}$ is negligible for any attacker $\mathcal{A}$.

The Decisional Diffie-Hellman Problem. We now review the definition of the Decisional Diffie-Hellman (DDH) problem. Let $\mathcal{D}$ be an attacker. Let $\mathbb{G}$ be a finite cyclic group generated by $g \in \mathbb{G}$. Let q be a prime order of $\mathbb{G}$, whose size depends on the security parameter λ. We define the DDH problem using the attacker $\mathcal{D}$'s advantage in distinguishing two distributions:

$$\begin{aligned} \mathbf{Adv}_{\mathcal{D},\mathbb{G}}^{\mathrm{DDH}}(\lambda) \;=\; & |\Pr[a \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_q; b \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_q : 1 \leftarrow \mathcal{D}(1^\lambda, g^a, g^b, g^{ab})] \\ - \; & \Pr[a \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_q; b \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_q; r \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_q : 1 \leftarrow \mathcal{D}(1^\lambda, g^a, g^b, g^r)]|. \end{aligned}$$

Equivalently [7, 8, 10],

$$\begin{aligned} \mathbf{Adv}_{\mathcal{D},\mathbb{G}}^{\mathrm{DDH}}(\lambda) \;=\; & |\Pr[w \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_q; g_2 \leftarrow g_1^w; r \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_q : 1 \leftarrow \mathcal{D}(1^\lambda, g_1, g_2, g_1^r, g_2^r)] \\ - \; & \Pr[w \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_q; g_2 \leftarrow g_1^w; r' \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_q \setminus \{r\} : 1 \leftarrow \mathcal{D}(1^\lambda, g_1, g_2, g_1^r, g_2^{r'})]|, \end{aligned}$$

where g_1 is the generator of $\mathbb{G}$.

We say that the DDH problem is hard if $\mathbf{Adv}_{\mathbb{G}}^{\mathrm{DDH}}(\lambda) = \max_{\mathcal{D}} \left\{ \mathbf{Adv}_{\mathcal{D},\mathbb{G}}^{\mathrm{CCA}}(\lambda) \right\}$ is negligible for any attacker $\mathcal{D}$.

Target Collision Resistant Hash Function (TCR). The security of the target collision resistant hash function denoted by H is defined as follows. Given a randomly chosen group element $x \in \mathbb{G}^n$, it is hard for an attacker $\mathcal{B}_1$ to find $y \neq x$ such that $\mathsf{H}(x) = \mathsf{H}(y)$. We define the attacker $\mathcal{B}_1$'s success probability by $\mathbf{Adv}_{\mathcal{B}_1,\mathsf{H}}^{\mathrm{COL}}(\lambda)$. We say that H is target collision-resistant if $\mathbf{Adv}_{\mathsf{H}}^{\mathrm{COL}}(\lambda) = \max_{\mathcal{B}_1} \left\{ \mathbf{Adv}_{\mathcal{B}_1,\mathsf{H}}^{\mathrm{COL}}(\lambda) \right\}$ is negligible for any attacker $\mathcal{B}_1$.

Key Derivation Function (KDF). In the proposed variant of the KD-KEM scheme, we will use the key derivation function denoted by KDF. Specifically, KDF takes a random element in the group $\mathbb{G}$ as input. Let l be the length of the output of KDF, which depends on the security parameter λ. We define the security of KDF with respect to an attacker $\mathcal{B}_2$ as follows. (Below, "ROR" stands for "real or random".)

$$\begin{aligned} \mathbf{Adv}_{\mathcal{B}_2,\mathsf{KDF}}^{\mathrm{ROR}}(\lambda) &= |\Pr[a \stackrel{\mathrm{R}}{\leftarrow} \mathbb{G} : 1 \leftarrow A(1^\lambda, \mathsf{KDF}(a))] \\ &- \Pr[a \stackrel{\mathrm{R}}{\leftarrow} \mathbb{G}; \mu \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^l : 1 \leftarrow A(1^\lambda, \mu)]|. \end{aligned}$$

We say that KDF is secure if $\mathbf{Adv}_{\mathsf{KDF}}^{\mathrm{ROR}}(\lambda) = \max_{\mathcal{B}_2} \left\{ \mathbf{Adv}_{\mathcal{B}_2,\mathsf{KDF}}^{\mathrm{ROR}}(\lambda) \right\}$ is negligible for any attacker $\mathcal{B}_2$.

Notice that the above security requirement on KDF is the same as that of the KDF functions used in [8, 10].

3 The Proposed Variant of the Cramer-Shoup KEM

Description. We describe our variant of the CS-KEM scheme, which we denote by "Π", as follows. (Readers are referred to the end of Section 1 for the underlying idea of our scheme.)

Key Generation: Pick a group $\mathbb{G}$ of prime order q and generators g_1 and g_2 of $\mathbb{G}$. Pick a target-collision resistant hash function $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and a key derivation function KDF. Then choose $(x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$ at random and compute

$$c = g_1^{x_1} g_2^{x_2}, \; d = g_1^{y_1} g_2^{y_2}.$$

Return public key $pk = (\mathbb{G}, q, g_1, g_2, c, d, \mathsf{H}, \mathsf{KDF})$ and private key $sk = (pk, x_1, x_2, y_1, y_2)$.

Encapsulation: Pick $r \in \mathbb{Z}_q^*$ at random and compute

$$u_1 = g_1^r, \; u_2 = g_2^r, \; \alpha = \mathsf{H}(u_1, u_2), \; v = c^r d^{r\alpha}, \; K = \mathsf{KDF}(u_1, c^r).$$

Return a ciphertext $\psi = (u_1, u_2, v)$ and a key K.

Decapsulation: Upon receiving $\psi = (u_1, u_2, v)$, compute

$$\alpha = \mathsf{H}(u_1, u_2), \; v = u_1^{x_1 + y_1\alpha} u_2^{x_2 + y_2\alpha}, \; K = \mathsf{KDF}(u_1, u_1^{x_1} u_2^{x_2})$$

If $v' = v$ then return K; otherwise, return $\perp$.

We show that the scheme Π is CCA-secure, relative to the DDH problem. More precisely, we prove the following theorem.

Theorem 1 *The KEM scheme Π is CCA-secure assuming that the DDH problem is hard and the underlying hash function H s target collision-resistant and key derivation function KDF is secure. More precisely, we have*

$$\mathbf{Adv}_{\Pi}^{CCA}(\lambda) \quad \leq \quad \mathbf{Adv}_{\mathbb{G}}^{DDH}(\lambda) + \mathbf{Adv}_{\mathsf{H}}^{COL}(\lambda) + \mathbf{Adv}_{\mathsf{KDF}}^{ROR}(\lambda) + \frac{q_D}{q}.$$

where λ denotes the security parameter and q_D is the number of queries to the decapsulation oracle.

Outline of Proof. The basic idea of the proof essentially follows that of the CS-KEM [8] and CS-PKE [7] schemes. Basically we need to show that by using a CCA-attacker for the scheme Π as a subroutine, a DDH attacker can solve the DDH problem: When the DDH attacker is given a right Diffie-Hellman tuple (g_1, g_2, g_1^r, g_2^r), it can perfectly simulate the environment of the CCA-attacker. On the other hand, when it is given $(g_1, g_2, g_1^r, g_2^{r'})$ where $r' \neq r$, the output of the decapsulation oracle will not be legitimate but we will show that this one won't be a problem.

In our proof, there is an important difference from the proof of the CS-KEM/CS-PKE schemes. Since the public key component c used to create $v = c^r d^{r\alpha}$ is "reused" to produce a key c^r, we need to assume that the attacker's view include c, d, v and c^r when breaking the confidentiality (i.e. "key indistinguishability") of the scheme Π. (Note that this is different from the CS-KEM/CS-PKE schemes in which an independent public key component is used to produce a key and hence v is not in the attacker's view.) By using an argument from linear algebra, we show that fortunately, this does not cause a problem. (Readers are referred to Equation (12).)

Proof. Fix an attacker $\mathcal{A}$ that breaks CCA-security of the scheme Π. Let $\mathcal{D}$ be an attacker for the DDH problem.

Simulation. The DDH attacker $\mathcal{D}$ simulates the environment of $\mathcal{A}$ as follows. Assume that $\mathcal{D}$ is given a DDH instance (g_1, g_2, u_1, u_2) where g_1 and g_2 are generators of a group $\mathbb{G}$ of prime-order q. $\mathcal{D}$ chooses $(x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$ at random and computes $c = g_1^{x_1} g_2^{x_2}$ and $d = g_1^{y_1} g_2^{y_2}$. $\mathcal{D}$ also chooses a hash function H and a key derivation function KDF, and gives $pk = (\mathbb{G}, q, g_1, g_2, c, d, \mathsf{H}, \mathsf{KDF})$ as a public key to $\mathcal{A}$.

When $\mathcal{A}$ queries ciphertexts to the decapsulation oracle in the find stage, $\mathcal{D}$ decapsulates them using (x_1, x_2, y_1, y_2).

$\mathcal{D}$ simulates the challenge ciphertext and the key as follows. $\mathcal{D}$ first sets $u_1^* = u_1$ and $u_2^* = u_2$, and computes $\alpha^* = \mathsf{H}(u_1^*, u_2^*)$, $v = (u_1^*)^{x_1 + y_1\alpha^*} (u_2^*)^{x_2 + y_2\alpha^*}$ and $K_1^* = \mathsf{KDF}(u_1^*, (u_1^*)^{x_1} (u_2^*)^{x_2})$. $\mathcal{D}$ also chooses K_0^* at random from the output space of KDF and picks $\beta \in \{0, 1\}$ at random. $\mathcal{D}$ finally gives $\mathcal{A}$ the challenge ciphertext-key pair $\psi^* = (u_1^*, u_2^*, K_\beta^*)$.

When $\mathcal{A}$ queries ciphertexts, all of which are different from ψ^*, to the decapsulation oracle in the find stage, $\mathcal{D}$ decapsulates them using (x_1, x_2, y_1, y_2).

Finally, when $\mathcal{A}$ outputs its guess β', $\mathcal{D}$ outputs 1 if $\beta' = \beta$; otherwise, outputs 0.

Analysis. We first analyze the case when $\mathcal{D}$ is given (g_1, g_2, u_2, u_2) such that $\log_{g_1} u_1 = \log_{g_2} u_2$. First, we prove the following lemma.

Lemma 1 *Let $r^* = \log_{g_1} u_1 = \log_{g_2} u_2$. Then we have*

$$\Pr[\mathcal{D}(1^\lambda, g_1, g_2, g_1^{r^*}, g_2^{r^*}) = 1] = \Pr[\mathsf{GameCCA}_{\mathcal{A}}^{\Pi}(\lambda) = 1]. \tag{1}$$

Proof. Note that since (x_1, x_2, y_1, y_2) is randomly chosen from $\mathbb{Z}_q^4$, the public key pk is distributed the same as that in the real attack.

By the simulation of the challenge ciphertext presented above, we have

$$\psi^* = (u_1^*, u_2^*, v^*) = (g_1^{r^*}, g_2^{r^*}, (g_1^{r^*})^{x_1 + y_1 \alpha^*} (g_2^{r^*})^{x_2 + y_2 \alpha^*}) = (g_1^{r^*}, g_2^{r^*}, c^{r^*} d^{r^* \alpha^*})$$

and

$$K_1^* = \mathsf{KDF}(u_1^*, (g_1^{r^*})^{x_1} (g_2^{r^*})^{x_2}) = \mathsf{KDF}(u_1^*, c^{r^*}).$$

Since K_0^* is drawn uniformly at random from the output space of KDF, (ψ^*, K_β^*) has the right distribution.

It remains to show that the output of the decapsulation oracle (both in the simulation and the real attack) has the right distribution. Now, we call a ciphertext $\psi = (u_1, u_2, v)$ is invalid if $\log_{g_1} u_1 \neq \log_{g_2} u_2$. We show that invalid ciphertexts are rejected except for negligible probability.

First, by the public key pk that $\mathcal{A}$ sees, we have the following equations:

$$\log_{g_1} c = x_1 + x_2 w \tag{2}$$

and

$$\log_{g_1} d = y_1 + y_2 w, \tag{3}$$

where $w = \log_{g_1} g_2$. Hence, one can view (x_1, x_2, y_1, y_2) as a random point on the plane defined by (2) and (3). From the challenge ciphertext, we have

$$\log_{g_1} v^* = r^*(x_1 + x_2 w + y_1 \alpha^* + y_2 w \alpha^*), \tag{4}$$

where $r^* = \log_{g_1} u_1^* = \log_{g_2} u_2^*$ and $\alpha^* = \mathsf{H}(u_1^*, u_2^*)$. Note that the challenge ciphertext (whether it is in the simulation or real attack) does not constrain (x_1, x_2, y_1, y_2) as the hyperplane defined by (4) contains the plane formed by the equations (2) and (3). Now consider the following equation obtained from the invalid ciphertext ψ:

$$\log_{g_1} v = r_1 x_1 + r_2 x_2 w + r_1 y_1 \alpha + r_2 y_2 w \alpha, \tag{5}$$

where $r_1 = \log_{g_1} u_1$ and $r_2 = \log_{g_1} u_2$ such that $r_1 \neq r_2$. If the decapsulation oracle does not reject ψ, the point (x_1, x_2, y_1, y_2) should lie on the hyperplane defined by (5). But observe that the equations (2), (3) and (5) are linearly independent, so the hyperplane defined by (5) intersects the plane formed by the equations (2) and (3) at a line. This happens with probability $1/q$, which is negligible. $\square$

We now analyze the case when $\mathcal{D}$ is given (g_1, g_2, u_2, u_2) such that $\log_{g_1} u_1 \neq \log_{g_2} u_2$. More precisely, we prove the following lemma.

Lemma 2 *Let $r_1^* = \log_{g_1} u_1$ and $r_2^* = \log_{g_2} u_2$, where $r_1^* \neq r_2^*$. Then we have β is independent from $\mathcal{B}$'s view.*

$$\Pr[\mathcal{D}(1^\lambda, g_1, g_2, g_1^{r_1^*}, g_2^{r_2^*}) = 1] \leq \frac{1}{2} + \mathbf{Adv}_{\mathsf{KDF}}^{ROR}(\lambda) + \mathbf{Adv}_{\mathsf{H}}^{COL}(\lambda) + \frac{q_D}{q}. \quad (6)$$

Proof. Recall that if a ciphertext $\psi = (u_1, u_2, v)$ is "invalid" then $\log_{g_1} u_1 \neq \log_{g_2} u_2$. Now, define $\mathsf{RejInvC}$ to be an event that the decapsulation oracle rejects all invalid ciphertexts. We first show that

$$\Pr[\beta' = \beta | \neg\mathsf{RejInvC}] \leq \frac{1}{2} + \mathbf{Adv}_{\mathsf{KDF}}^{\mathrm{ROR}}(\lambda). \quad (7)$$

To prove this, we consider the distribution of the point $(x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$ conditioned on $\mathcal{A}$'s view. Now assume that the decapsulation oracle decapsulates only valid ciphertexts, each of which is denoted by (u_1, u_2, v). Then we can get the equation

$$\log_{g_1} v = r(x_1 + x_2 w + y_1 \alpha + y_2 w \alpha), \quad (8)$$

where $r = \log_{g_1} u_1 = \log_{g_2} u_2$ and $\alpha = \mathsf{H}(u_1, u_2)$. Now assume that $\mathcal{A}$ even gets the key material $u_1^{x_1} u_2^{x_2}$ through querying the decapsulation oracle. (In fact, $\mathcal{A}$ only gets the key which is the output of KDF which "wraps" the key material $u_1^{x_1} u_2^{x_2}$.) Hence the following equation is in $\mathcal{A}$'s view:

$$\log_{g_1} u_1^{x_1} u_2^{x_2} = r(x_1 + x_2 w). \quad (9)$$

However, the equations (2), (3), (8) and (9) are linearly dependent. Hence, no information about the point (x_1, x_2, y_1, y_2) is leaked from the above equations.

Now consider the challenge ciphertext $\psi^* = (u_1^*, u_2^*, v^*)$ and the key $K_1^* = \mathsf{KDF}(u_1^*, (u_1^*)^{x_1}(u_2^*)^{x_2})$, produced by the simulation. Suppose that $\mathcal{A}$ gets the key material $(u_1^*)^{x_1}(u_2^*)^{x_2}$ at the worst case. Since v^* and $(u_1^*)^{x_1}(u_2^*)^{x_2}$ are in $\mathcal{A}$'s view, (x_1, x_2, y_1, y_2) should then satisfy the following equations

$$\log_{g_1} v^* = r_1^* x_1 + r_2^* x_2 w + r_1^* y_1 \alpha^* + r_2^* y_2 w \alpha^*, \quad (10)$$

where $r_1^* = \log_{g_1} u_1^*$, $r_2^* = \log_{g_2} u_2^*$ with $r_1^* \neq r_2^*$ and $\alpha^* = \mathsf{H}(u_1^*, u_2^*)$, and

$$\log_{g_1} (u_1^*)^{x_1} (u_2^*)^{x_2} = r_1^* x_1 + r_2^* x_2 w. \quad (11)$$

Now observe that

$$\det \begin{bmatrix} 1 & w & 0 & 0 \\ 0 & 0 & 1 & w \\ r_1^* & r_2^* w & r_1^* \alpha^* & r_2^* \alpha^* w \\ r_1^* & r_2^* w & 0 & 0 \end{bmatrix} = w^2 \alpha^* (r_1^* - r_2^*)^2 \neq 0. \quad (12)$$

Hence, the equations (2), (3), (10) and (11) are linearly independent. Note that $(u_1^*)^{x_1}(u_2^*)^{x_2}$ is distributed uniformly in $\mathbb{G}$ since r_1^* and r_2^* are chosen uniformly at random from $\mathbb{Z}_q$ and that K_0^* has been chosen uniformly at random and independently from anything else. Thus the distribution of β is independent from $\mathcal{A}$'s view under the assumption that KDF is secure and we get the bound (7).

We now show that the probability that the decapsulation oracle rejects all invalid ciphertexts is bounded by insecurity of hash function and some negligible probability.

$$\Pr[\mathsf{RejInvC}] \leq \mathbf{Adv}_{\mathsf{H}}^{\mathrm{COL}}(\lambda) + \frac{q_D}{q}, \tag{13}$$

where q_D denotes the number of the queries to the decapsulation oracle.

Suppose that $\mathcal{A}$ submits an invalid ciphertext $\psi = (u_1, u_2, v) \neq \psi^*$ to the decapsulation oracle. First, note that it is not possible that $(u_1, u_2) = (u_1^*, u_2^*)$ since $\psi \neq \psi^*$, we have $v \neq v^*$ and hence the decapsulation oracle will reject ψ straight away. Note also that it is possible that $(u_1, u_2) \neq (u_1^*, u_2^*)$ and $\alpha = \alpha^*$ *but* the probability that this happens is bounded by the insecurity of the hash function H since this event implies the violation of the target collision-resistance of H.

Thus, for up to q_D invalid ciphertexts such that $(u_1, u_2) \neq (u_1^*, u_2^*)$ and $\alpha \neq \alpha^*$. In this case, if the point (x_1, x_2, y_1, y_2) lied on the hyperplane defined by the following equation

$$\log_{g_1} v = r_1 x_1 + r_2 x_2 w + r_1 y_1 \alpha + r_2 y_2 w \alpha, \tag{14}$$

where $r_1 = \log_{g_1} u_1$ and $r_2 = \log_{g_1} u_2$, the decapsulation oracle would accept the ciphertext ψ. However, observe that

$$\det \begin{bmatrix} 1 & w & 0 & 0 \\ 0 & 0 & 1 & w \\ r_1^* & r_2^* w & r_1^* \alpha^* & r_2^* \alpha^* w \\ r_1 & r_2 w & r_1 \alpha & r_2 \alpha w \end{bmatrix} = w^2 (r_1 - r_2)(r_1^* - r_2^*)(\alpha^* - \alpha) \neq 0.$$

Hence, (2), (3), (10) and (14) are linearly independent, implying that the hyperplane defined by (14) intersects the line formed by intersecting (2), (3) and (10) at a point, which happens with negligible probability $1/q$. Considering that there are q_D decapsulation queries, we get (13).

$\square$

Note that from (7) and (13), we get

$$\begin{aligned} \Pr[\beta' = \beta] &= \Pr[\beta' = \beta | \neg\mathsf{RejInvC}] \Pr[\neg\mathsf{RejInvC}] + \Pr[\beta' = \beta | \mathsf{RejInvC}] \Pr[\mathsf{RejInvC}] \\ &\leq \Pr[\beta' = \beta | \neg\mathsf{RejInvC}] + \Pr[\mathsf{RejInvC}] \leq \frac{1}{2} + \mathbf{Adv}_{\mathsf{KDF}}^{\mathrm{ROR}}(\lambda) + \mathbf{Adv}_{\mathsf{H}}^{\mathrm{COL}}(\lambda) + \frac{q_D}{q}. \end{aligned}$$

Then, from the construction of $\mathcal{D}$, we have

$$\Pr[\mathcal{D}(1^\lambda, g_1, g_2, g_1^{r_1^*}, g_2^{r_2^*}) = 1] = \Pr[\beta' = \beta] \leq \frac{1}{2} + \mathbf{Adv}_{\mathsf{KDF}}^{\mathrm{ROR}}(\lambda) + \mathbf{Adv}_{\mathsf{H}}^{\mathrm{COL}}(\lambda) + \frac{q_D}{q}.$$

Combining the bounds from Lemma 1 and Lemma 2 (i.e. by subtracting (1) from (6)), we get the bound in the theorem statement. $\square$

4 Comparisons

In Table 1, we summarize the basic parameters such as public key, ciphertext of CS-KEM [8], KD-KEM [15], HK-KEM [12] and ours. We also summarize whether those schemes provide full CCA-security, assuming the hardness of the DDH problem.

Scheme	Public key	Ciphertext	Key	Full CCA
CS-KEM [8]	g_1, g_2, c, d, h	$g_1^r, g_2^r, (cd^\alpha)^r$	h^r	Yes
KD-KEM [15]	g_1, g_2, c, d	g_1^r, g_2^r	$(cd^\alpha)^r$	No
HK-KEM [12]	g_1, c, d, h	$g_1^r, (cd^\alpha)^r$	h^r	No
Ours	g_1, g_2, c, d	$g_1^r, g_2^r, (cd^\alpha)^r$	c^r	Yes

Table 1: Comparison of Our KEM Scheme with Other KEM Schemes

As one can notice from the above table, our scheme is clearly more efficient than the CS-KEM scheme. Although it is less efficient than the KD-KEM and HK-KEM schemes, an advantage of our scheme over those schemes is that our scheme provides full CCA-security. (Note that KD-KEM and HK-KEM are proven CCCA-secure [12], which is weaker than full CCA-security.)

In Table 2, we summarize the computational costs of the above-mentioned schemes. In the table, "E" stands for "Exponentiation" and "DE" stands for "Double Exponentiation", which is a special case of multi-exponentiation for two bases, e.g. $A^a B^b$.

Since there are many factors that determine running time of various multi-exponentiation algorithms [2, 17], it would be difficult to state decisively one double exponentiation is equivalent to how much of single exponentiation. (Note that if we use the naive approach that computes two single exponentiations separately and multiply them together, 1 DE = 2 E.) But if one adopts the "multi-exponentiation with a sliding window" algorithm assuming the unsigned binary representation of exponents as described in [2], one can obtain 1 DE = 1.39E if window size = 2 and the bit-length of $q = 256$. The figures in the parentheses in Table 2 are obtained based on this assumption.

Scheme	Enc. Cost	Dec. Cost
CS-KEM [8]	3E + 1DE (4.39E)	2DE (2.78E)
KD-KEM [15]	2E + 1DE(3.39E)	1DE (1.39E)
HK-KEM [12]	3.39E (2E + 1DE)	2E
Ours	4E	2.78E

Table 2: Comparison of Computational Costs

Notice from the above table that in terms of computational costs, the difference between our scheme and both KD-KEM and HK-KEM is less than one exponentiation.

Finally we remark that as done for CS-KEM and KD-KEM respectively in [8] and in [19], one can make the key generation and decapsulation algorithms of our KEM scheme more efficient. This is described in detail in Appendix A.

5 Conclusion

In this paper, we proposed a new variant of the Cramer-Shoup KEM (CS-KEM) scheme which is more efficient than the original Cramer-Shoup KEM and fully CCA-secure in the standard model, relative to the DDH problem. Our results shows that the original CS-KEM can further be optimized without losing full CCA-security.

References

[1] M. Abe, R. Genaro and K. Kurosawa, *Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM*, Cryptology ePrint Archive, Report 2005/027, 2005. (Last update: 11 October 2006.)

[2] R. M. Avanzi, *The Complexity of Certain Multi-Exponentiation Techniques in Cryptography*, Journal of Cryptology, Vol. 18 (4), pp. 357 – 373, Springer-Verlag, 2005.

[3] M. Bellare, R. Canetti and H. Krawczyk, *Keying Hash Functions for Message Authentication*, In Crypto '96, LNCS 1109, pp. 1–15, Springer-Verlag, 1996.

[4] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, *Relations Among Notions of Security for Public-Key Encryption Schemes*, In Crypto '98, LNCS 1462, pp. 26–45, Springer-Verlag, 1998.

[5] M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, In ACM-CCS '93, pp. 62–73, ACM, 1993.

[6] D.J. Bernstein, *Pippengers exponentiation algorithm*, Preprint, 2002. Available from http://cr.yp.to

[7] R. Cramer and V. Shoup, *A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack*, In Crypto '98, LNCS 1462, pp. 13–25, Springer-Verlag, 1998.

[8] R. Cramer and V. Shoup, *Design and Analysis of Practical Public-key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack*, SIAM Journal of Computing 33, pp. 167–226, 2003.

[9] D. Dolev, C. Dwork and M. Naor, *Non-malleable Cryptography*, In STOC '91, pp. 542–552, ACM, 1991.

[10] R. Gennaro and V. Shoup, *A Note on An Encryption Scheme of Kurosawa and Desmedt*, Cryptology ePrint Archive, Report 2004/294, 2004.

[11] J. Herranz, D. Hofheinz and E. Kiltz, *The Kurosawa-Desmedt Key Encapsulation is not Chosen-Ciphertext Secure*,Cryptology ePrint Archive, Report 2006/207, 2006.

[12] D. Hofheinz and E. Kiltz. *Secure Hybrid Encryption from Weakened Key Encapsulation*, In LNCS 4622, CRYPTO 2007, pp. 553–571, Springer-Verlag, 2007.

[13] ISO 18033-2, *An Emerging Standard for Public-Key Encryption*, 2004.

[14] E. Kiltz, *Chosen-Ciphertext Secure Key-Encapsulation Based on Gap Hashed Diffie-Hellman*, In PKC '07, LNCS 4450, pp. 282–297, Springer-Verlag, 2007.

[15] K. Kurosawa and Y. Desmedt, *A New Paradigm of Hybrid Encryption Scheme*, In Crypto '04, LNCS 3152, pp. 426–442, Springer-Verlag, 2004.

[16] X. Lu, X. Lai and D. He, *Improved efficiency of Kiltz07-KEM*, Cryptology ePrint Archive, Report 2008/312, 2008.

[17] B. Möller and A. Rupp, *Faster Multi-Exponentiation through Caching: Accelerating (EC)DSA Signature Verification*, In Security and Cryptography for Networks (SCN 2008), LNCS 5229, pp. 39–56, Springer-Verlag, 2008.

[18] M. Naor and M. Yung, *Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks*, In STOC '90, pp. 427–437, ACM, 1990.

[19] L. T. Phong and W. Ogata, *On Some Variations of Kurosawa-Desmedt Public-Key Encryption Scheme*, IEICE Transactions 90-A(1), pp. 226–230, 2007.

[20] V. Shoup, *A Proposal for an ISO Standard for Public Key Encryption (version 2.1)*, ISO/IEC JTC 1/SC 27, 2001.

A An Efficient Variant of Our KEM Scheme

Description. Adopting the techniques in [8, 19], one can design an efficient variant of our KEM scheme, which we denote by "$\widetilde{\Pi}$", as follows.

Key Generation: Pick a group $\mathbb{G}$ of prime order q and generator g_1 of $\mathbb{G}$. Pick a target-collision resistant hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q^*$ and a key derivation function KDF. Then choose $(w, x, y) \in \mathbb{Z}_q^3$ at random and compute

$$g_2 = g_1^w,\ c = g_1^x,\ d = g_1^y.$$

Return public key $pk = (\mathbb{G}, q, g_1, g_2, c, d, \mathsf{H}, \mathsf{KDF})$ and private key $sk = (pk, x, y, w)$.

Encapsulation: Pick $r \in \mathbb{Z}_q^*$ at random. Compute

$$u_1 = g_1^r,\ u_2 = g_2^r,\ \alpha = \mathsf{H}(u_1, u_2),\ v = c^r d^{r\alpha},\ K = \mathsf{KDF}(u_1, c^r).$$

Return ciphertext $\psi = (u_1, u_2, v)$ and key K.

Decapsulation: Upon receiving $\psi = (u_1, u_2, v)$, compute

$$\alpha = \mathsf{H}(u_1, u_2),\ u_2' = u_1^w\ , v' = u_1^{x+y\alpha},\ K = \mathsf{KDF}(u_1, u_1^x).$$

If $u_2' = u_2$ and $v' = v$ then return K; otherwise, return $\perp$.

The above scheme is also CCA-secure. Regarding this, we prove the following theorem.

Theorem 2 *If the KEM scheme* Π *(described in Section 3) is CCA-secure then the above KEM scheme* $\widetilde{\Pi}$ *is CCA-secure. More precisely, we have*

$$\mathbf{Adv}_{\widetilde{\Pi}}^{CCA}(\lambda) \quad \leq \quad \mathbf{Adv}_{\Pi}^{CCA}(\lambda) + \frac{q_D}{q}.$$

where λ *denotes the security parameter and* q_D *is the number of queries to the decapsulation oracle.*

Proof. Fix an attacker $\mathcal{A}$ for the scheme Π. Also, fix an attacker $\tilde{\mathcal{A}}$ for the scheme $\widetilde{\Pi}$.

Assume that $\mathcal{A}$ is provided with the public key $pk = (\mathbb{G}, q, g_1, g_2, c, d)$ and the private key $sk = (pk, x_1, x_2, y_1, y_2)$, where g_1 and g_2 are generators of $\mathbb{G}$ and $c = g_1^{x_1} g_2^{x_2}$ and $d = g_1^{y_1} g_2^{y_2}$. $\mathcal{A}$ simply gives $\tilde{\mathcal{A}}$ pk as the public key of the scheme Π. $\mathcal{A}$ sets $g_2 = g_1^w$ for some $w \in \mathbb{Z}_q$, $x = x_1 + wx_2$ and $y = y_1 + wy_2$. (Note that $\mathcal{A}$ does not the value w.) Since $c = g_1^{x_1} g_2^{x_2} = g_1^{x_1 + wx_2} = g^x$, $d = g_1^{y_1} g_2^{y_2} = g_1^{y_1 + wy_2} = g^x$ by definition of w and (x, y), the public key pk is distributed identically in both $\mathcal{A}$ and $\tilde{\mathcal{A}}$'s view.

When $\tilde{\mathcal{A}}$ queries a ciphertext $\psi = (u_1, u_2, v)$ to the decapsulation oracle in the find stage, $\mathcal{A}$ forwards it to its decapsulation oracle, gets a decapsulation result and sends it back to $\tilde{\mathcal{A}}$.

Sometime later, $\mathcal{A}$ gets a challenge ciphertext and a key pair $(\psi^*(= u_1^*, u_2^*, v^*), K_\beta)$, where $\beta \in \{0, 1\}$ is chosen at random, and forwards the pair to $\tilde{\mathcal{A}}$ as a challenge ciphertext of the scheme $\widetilde{\Pi}$ and a key.

When $\tilde{\mathcal{A}}$ queries a ciphertext $\psi = (u_1, u_2, v)$ to the decapsulation oracle in the guess stage, $\mathcal{A}$ forwards it to its decapsulation oracle, gets a decapsulation result and sends it back to $\tilde{\mathcal{A}}$.

When $\tilde{\mathcal{A}}$ outputs its guess, $\mathcal{A}$ outputs it as its guess.

We compute the probability that an invalid ciphertext $\psi = (u_1, u_2, v)$, which should have been rejected, is accepted by the simulated decapsulation oracle.

Since we have assumed that $\psi = (u_1, u_2, v)$ is invalid, the condition $[(u_1^w \neq u_2) \wedge (u_1^{x+y\alpha} = v)]$ or $[(u_1^w = u_2) \wedge (u_1^{x+y\alpha} \neq v)]$ or $[(u_1^w \neq u_2) \wedge (u_1^{x+y\alpha} \neq v)]$ holds. However, if the last two conditions held, the simulated decapsulation oracle would reject ψ. Hence the first condition $[(u_1^w \neq u_2) \wedge (u_1^{x+y\alpha} = v)]$ must hold when invalid ψ is not rejected by the simulated decapsulation oracle. Note that $u_1^w \neq u_2$ means $r_1 \neq r_2$ where $r_1 = \log_{g_1} u_1$ and $r_2 = \log_{g_1} u_2$. Note also that since $x = x_1 + wx_2$ and $y = x_y + wy_2$, $u_1^{x+y\alpha} = v$ is equivalent to

$$\begin{aligned} & [r_1\{(x_1 + wx_2) + (y_1 + wy_2)\alpha\}] \bmod q = [r_1(x_1 + y_1\alpha) + r_2 w(x_2 + y_2\alpha)] \bmod q \\ \iff \quad & w(r_1 - r_2)(x_1 + wy_2) = 0 \bmod q. \end{aligned}$$

As $r_1 \neq r_2$ by the assumption and $w \neq 0 \bmod q$, the above equation holds with probability $1/q$, which is negligible. Hence we get the bound in the theorem statement. $\square$