# Kurosawa-Desmedt Key Encapsulation Mechanism, Revisited

Kaoru Kurosawa[1] and Le Trieu Phong[2]

[1] Ibaraki University
[2] NICT, Japan

**Abstract.** While the hybrid public key encryption scheme of Kurosawa and Desmedt (CRYPTO 2004) is provably secure against chosen ciphertext attacks (namely, IND-CCA-secure), its associated key encapsulation mechanism (KEM) is not IND-CCA-secure (Herranz et al. 2006, Choi et al. 2009). In this paper, we show a simple twist on the Kurosawa-Desmedt KEM turning it into a scheme with IND-CCA security under the decisional Diffie-Hellman assumption. Our KEM beats the standardized version of Cramer-Shoup KEM in ISO/IEC 18033-2 by margins of at least 20% in encapsulation speed, and 20% $\sim$ 60% in decapsulation speed. Moreover, the public and secret key sizes in our schemes are at least 160-bit smaller than those of the Cramer-Shoup KEM. We then generalize the technique into hash proof systems, proposing several KEM schemes with IND-CCA security under decision linear and decisional composite residuosity assumptions respectively. All the KEMs are in the standard model, and use standard, computationally secure symmetric building blocks.

**Keywords:** Kurosawa-Desmedt KEM, IND-CCA security, hash proof systems, standard model.

## 1 Introduction

### 1.1 Background

Key Encapsulation Mechanism (KEM) is an asymmetric encryption technique allows generating *simultaneously* a random key $K_s$ together with its encryption $C$, termed encapsulation. The key $K_s$ then will be used for long data encryption, while the encapsulation $C$ is used for sharing $K_s$. In other words, KEM serves as a delivery of secret keys used in symmetric encryption.

KEM implies public-key encryption (PKE). Indeed, it can be used to construct *hybrid* PKE, namely PKE with unrestricted message space, when combining with a data encapsulation mechanism (DEM) [13]. In practice, since the DEM part is already highly efficient, one usually concerns about the performance of the KEM part. Specific constructions of KEM are incorporated in the standards ISO/IEC 18033-2 [1], ANSI X9.44 [5], and can be considered for e-Government usage in the future [2]. KEM is widely yet implicitly used in the TLS Handshake Protocol [23].

In 2004, Kurosawa and Desmedt [24], improved upon the seminal work of Cramer and Shoup [12], published an efficient hybrid PKE, whose security proof was refined in [15], resisting chosen ciphetext attacks (IND-CCA) under the decisional Diffie-Hellman (DDH) assumption. Unlike Cramer-Shoup scheme, the KEM part of the Kurosawa-Desmedt scheme is not IND-CCA secure, as shown in 2006 in [11, 19]. In 2007, by creatively switching elements in the Kurosawa-Desmedt KEM, Kiltz [22] presented an IND-CCA-secure KEM, and yet under the less standard Gap Hashed Diffie-Hellman (GHDH) assumption. On the other hand, sticking to the DDH assumption, Abe, Gennaro, Kurosawa [4], and Hofheinz, Kiltz [20] showed the Kurosawa-Desmedt KEM only meets weakened notions of CCA security.

While weakened IND-CCA security as defined in [4,20] can be converted into IND-CCA security (see Section 1.4), there is still no direct security proof for any variant of the Kurosawa-Desmedt KEM. A summarization of these discussions is in Table 1.

**Table 1.** Classification of Kurosawa-Desmedt (KD) KEM and its variants.

| **Security** ($\downarrow$) **Assumption** ($\rightarrow$) | GHDH | DDH |
|---|---|---|
| Weakened IND-CCA | – | [4], [20] (KD KEM) |
| IND-CCA | [22] (dual KD KEM) | This paper (with direct proof) |

### 1.2 Our contributions

Our results can be categorized as follows.

**Theoretical contribution.** We show a slight twist on the insecure Kurosawa-Desmedt KEM turning it into an IND-CCA-*secure* one. Formally, we propose a variant of the Kurosawa-Desmedt KEM which can be proved IND-CCA-secure under the DDH assumption. That is, we fulfill Table 1 with the most "*desirable*" KEM in terms of security assumption (namely, DDH) and security notion (namely, IND-CCA).

The twist is simple. Details are discussed at length at the beginning of Section 3.1, but a high view is as follows. In the original Kurosawa-Desmedt KEM, the encapsulation of a symmetric key $v$ consists of group elements $(u_1, u_2)$. In our proposal, we do not return the whole $v$ as the shared symmetric key, but split it into two independent keys $k_s$ and $k_a$. The key $k_s$ is then returned as the shared key, while the key $k_a$ is internally used to authenticate the encapsulation $(u_1, u_2)$. This authentication step is important as it protects the KEM against adversarial decapsulation queries, and is novel to this work in the sense that, with the twist, previous security proof for hybrid PKE in [15] can be *as is* reused for the KEM case, without any loss factor to the main complexity assumption.

**Practical impact.** The result is not only of theoretical interest. Indeed, compared to the existing practice [1], namely the standardized ACE-KEM basing on

the same assumption in the standard model, we achieve more than 20% improvement over encapsulation speed, and at least 20% improvement over decapsulation speed in general. For specific choices of the base group such as prime-field NIST elliptic curves, the speed improvement on decapsulation can go up to 60%. These estimations are confirmed by experimental results in Section 3.2.

In sizes, the public and secret keys in our schemes are one group element, or at least 160-bit, smaller than those of the ACE-KEM. The encapsulation length is also slightly shorter. See Table 2 in Section 3.2 for details.

These improvements are significant, as frequently there are large amounts of asymmetric decryption work (e.g., in SSL/TLS servers), or several public and secret keys must be created and kept internally in memory (e.g., as in iOS devices [7] where asymmetric encryption enables file access on the background while the device is locked).

**DLIN-based and DCR-based extensions.** Our method can be extended to hash proofs systems. When coupling with known constructions of hash proof systems in the literature, we obtain KEMs under the decision linear (DLIN) and decisional composite residuosity (DCR) assumptions, respectively.

### 1.3 Other usage of KEM beyond hybrid encryption

While original application of KEM is hybrid PKE, the ability to output a shared symmetric key allows KEM to have other applications as well. For example, KEM can be used to build schemes for identification [6] and authenticated key exchange (AKE) [10, 16, 29]. In particular, Boyd et al. [10] showed that a one-round AKE protocol can be constructed from IND-CCA secure KEM, and Fujioka et al. [16] showed that a two-pass AKE protocol with weak perfect forward secrecy can be constructed from IND-CCA secure KEM. This additionally illustrates why KEM is preferable over PKE alone.

### 1.4 More related works

The proof given in [24] depends on some information theoretically secure components, which affects the efficiency of the hybrid PKE scheme. The refined proof in [15] weakens the components to computationally secure ones.

Already in [11, 19], it was remarked that, if one models the key derivation function as a random oracle and is content with a much stronger assumption than DDH, the Kurosawa-Desmedt KEM can be proved IND-CCA-secure.

Okamoto [26] presented a KEM derived from the Kurosawa-Desmedt hybrid PKE. The KEM is IND-CCA-secure under the DDH assumption, and yet theoretically relies on an arguably non-standard primitive called pseudo-random function with pairwise independent random sources.

We are informed by Takahiro Matsuda that constrained IND-CCA (CCCA) security [20] can be converted into standard IND-CCA security as done in [8] using essentially the same idea with this work. The transformation, while generic and applied to the original Kurosawa-Desmedt KEM, however has a loss factor

of 4 in the security reduction. Our approach in this paper puts aside constrained IND-CCA definition, giving a direct proof for the KEM and related schemes from hash proof systems and yielding a theoretically better loss factor of 1 to the main complexity assumptions (namely DDH, DLIN, and DCR).

In the same vein, LCCA-secure KEM as defined in [4] can be converted to IND-CCA-secure Tag-KEM [4, Theorem 3] which in turn yields hybrid PKE. The conversion again has a loss factor of 2 to the main complexity assumption. The application of Tag-KEM beyond hybrid PKE is arguably less clear than KEM.

The conversions from CCCA or LCCA security to CCA security, while generic, are of theoretical interests, since proving that a concrete scheme is CCCA-secure or LCCA-secure is apparently not easier than directly showing that scheme is IND-CCA-secure.

## 2 Preliminaries

**KEM.** A KEM consists of key generation $\mathsf{KG}$, encapsulation $\mathsf{Encap}$, and decapsulation $\mathsf{Decap}$ algorithms. $\mathsf{KG}(1^\kappa)$ with security parameter $\kappa$ outputs public key $pk$ and secret key $sk$. The algorithm $\mathsf{Encap}(pk)$ returns a pair $(C, K)$. Correctness holds if $\mathsf{Decap}(sk, C) = K$.

**IND-CCA security of KEM.** To define the security, consider the following game with adversary $\mathcal{A}$. First, $(pk, sk) \leftarrow \mathsf{KG}(1^\kappa)$ and $pk$ is given to $\mathcal{A}$. In the so-called find stage, $\mathcal{A}$ can query any $C$ of its choice to oracle $\mathsf{Decap}(sk, \cdot)$.

Then $\mathcal{A}$ invoke a challenge oracle. The oracle computes $(C^*, K^*) \leftarrow \mathsf{Encap}(pk)$, then takes $K_*$ randomly satisfying $|K^*| = |K_*|$, and chooses $b \xleftarrow{\$} \{0, 1\}$. The oracle returns challenge pair $(C^*, K(b))$ in which $K(0) = K^*$ and $K(1) = K_*$.

After that, in the guess stage, $\mathcal{A}$ can again access to the oracle $\mathsf{Decap}(sk, \cdot)$, but is not allowed to query $C^*$ to the decapsulation oracle. Finally, $\mathcal{A}$ returns $b'$ as a guess of the hidden $b$.

The KEM is IND-CCA-secure if the advantage

$$\mathbf{Adv}^{\mathrm{ind-cca}}_{\mathcal{A}}(\kappa) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

is negligible in $\kappa$ for all poly-time adversary $\mathcal{A}$.

Taking an element $a$ randomly from a set $A$ is notationally expressed by $a \xleftarrow{\$} A$. Let $\kappa$ be the security parameter. We requires following building blocks. Concrete schemes can be found in [1, Section 6].

**TCR.** A target collision resistant hash function $\mathsf{TCR} : \mathcal{E}(\kappa) \to \mathcal{R}(\kappa)$ is defined as follows. Given a target $x^* \xleftarrow{\$} \mathcal{E}(\kappa)$, it is hard for all poly-time adversary $\mathcal{A}$ to find $x \in \mathcal{E}(\kappa)$ satisfying $\mathsf{TCR}(x) = \mathsf{TCR}(x^*)$. Formally, the advantage

$$\mathbf{Adv}^{\mathsf{TCR}}_{\mathcal{A}}(\kappa) = \Pr[x \leftarrow \mathcal{A}(x^*) : \ x \neq x^* \wedge \mathsf{TCR}(x) = \mathsf{TCR}(x^*)]$$

is negligible for all poly-time adversary $\mathcal{A}$.

**Fig. 1.** Our IND-CCA-secure KEM under the DDH assumption.

| $\underline{\mathsf{KG}(1^\kappa):}$ | $\underline{\mathsf{Encap}(pk):}$ | $\underline{\mathsf{Decap}(sk, C):}$ |
|---|---|---|
| $g_1, g_2 \xleftarrow{\$} \mathbb{G}$ | $r \xleftarrow{\$} \mathbb{Z}_q$ | Parse $C = (u_1, u_2, t)$ |
| $(x_1, x_2, y_1, y_2) \xleftarrow{\$} \mathbb{Z}_q^4$ | $u_1 \leftarrow g_1^r, u_2 \leftarrow g_2^r$ | $\alpha \leftarrow \mathsf{TCR}(u_1, u_2)$ |
| $c \leftarrow g_1^{x_1} g_2^{x_2}$ | $\alpha \leftarrow \mathsf{TCR}(u_1, u_2)$ | $v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$ |
| $d \leftarrow g_1^{y_1} g_2^{y_2}$ | $v \leftarrow c^r d^{r\alpha}$ | $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$ |
| $pk \leftarrow (g_1, g_2, c, d)$ | $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$ | If $t = \mathsf{MAC}_{k_a}(u_1, u_2)$ |
| $sk \leftarrow (x_1, x_2, y_1, y_2)$ | $t \leftarrow \mathsf{MAC}_{k_a}(u_1, u_2)$ | return $k_s$ |
| Return $(pk, sk)$ | Return $C = (u_1, u_2, t)$ and $K = k_s$ | Else return $\perp$ |

**KDF.** We assume that there exists a key derivation function $\mathsf{KDF} : \mathcal{K}(\kappa) \to \{0,1\}^{2n(\kappa)}$ such that $\mathsf{KDF}(v)$ for random $v \in \mathcal{K}(\kappa)$ is computationally random over $\{0,1\}^{2n(\kappa)}$. Formally, the advantage

$$\mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{D}}(\kappa) = \left| \Pr_{v \xleftarrow{\$} \mathcal{K}(\kappa)} [\mathcal{D}(\mathsf{KDF}(v)) = 1] - \Pr_{(k,k') \xleftarrow{\$} \{0,1\}^{2n(\kappa)}} [\mathcal{D}(k, k') = 1] \right|$$

is negligible for all poly-time distinguishers $\mathcal{D}$.

**MAC.** A message authentication code $\mathsf{MAC} : \{0,1\}^{n(\kappa)} \times \mathcal{E}(\kappa) \to \{0,1\}^{\tau(\kappa)}$ takes inputs $k \in \{0,1\}^{n(\kappa)}$ and $x \in \mathcal{E}(\kappa)$ to compute tag $t = \mathsf{MAC}_k(x)$. For random key $k \xleftarrow{\$} \{0,1\}^{n(\kappa)}$, the adversary $\mathcal{A}$ is given at most one pair $(x^*,\ t^* = \mathsf{MAC}_k(x^*))$ where $x^*$ is of $\mathcal{A}$'s own choice. The adversary $\mathcal{A}$ then returns a pair $(x, t)$. It is required that the following advantage

$$\mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}}(\kappa) = \Pr[x \neq x^* \wedge t = \mathsf{MAC}_k(x)]$$

is negligible for all poly-time distinguishers $\mathcal{A}$.

## 3 Kurosawa-Desmedt KEM, revisited

Let $\mathbb{G} = \langle g \rangle$ be a group, generated by $g$, of prime public order $2^\kappa < q < 2^{\kappa+1}$ for security parameter $\kappa$.

The DDH assumption on $\mathbb{G}$ asserts that, for all poly-time distinguishers $\mathcal{D}$, non-unit random elements $g_1, g_2 \xleftarrow{\$} \mathbb{G}$, and $r \neq s \xleftarrow{\$} \mathbb{Z}_q$, the advantage

$$\mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{D}}(\kappa) = \left| \Pr[\mathcal{D}(g_1, g_2, g_1^r, g_2^r) = 1] - \Pr[\mathcal{D}(g_1, g_2, g_1^r, g_2^s) = 1] \right|$$

is negligible on parameter $\kappa$.

### 3.1 Our proposed KEM under DDH

The construction is depicted in Figure 1. In the construction, keys $k_s$ and $k_a$ are of $n$-bit length. In $\mathsf{Decap}$, if $u_1 \notin \mathbb{G}$ or $u_2 \notin \mathbb{G}$ then $\perp$ is returned immediately

at the beginning. The description of symmetric building blocks $\mathsf{TCR}$, $\mathsf{KDF}$, and $\mathsf{MAC}$ are in Section 2.

The main difference with the Kurosawa-Desmedt KEM is, in $\mathsf{Encap}(pk)$, the element $v$ is spitted in two keys $(k_s, k_a)$ by $\mathsf{KDF}$. Then, the key $k_a$ is used to authenticate elements $(u_1, u_2)$ inside $\mathsf{Encap}(pk)$, while the key $k_s$ is returned as the shared symmetric key. The crucial point here is the authentication of $(u_1, u_2)$ by the $\mathsf{MAC}$, which helps proving IND-CCA security of our proposal. This technique, while simple, has been neglected in the literature.

Perhaps it is illustrative to see how our KEM resists against the chosen ciphertext attack in [11,19] that breaks the Kurosawa-Desmedt KEM. Recall that, in the attack, the adversary first obtains the challenge encapsulation consisting of $(u_1^*, u_2^*)$. The adversary then queries the decapsulation oracle with query of form $((u_1^*)^r, (u_2^*)^r)$ where $r \in \mathbb{Z}_q$ is random of its own choice. In [11, 19], it is showed that, by only two such queries, the encapsulated symmetric key can be computed with overwhelming probability. In comparison, in our KEM, the tag $t$ is effective as a hedge against such malformed queries. When the adversary submits $(u_1, u_2, t) = ((u_1^*)^r, (u_2^*)^r, t)$, the corresponding $v$ can be proved randomly distributed under the DDH assumption (in the proof, see $\mathbf{Game}_4$). This means corresponding keys $(k_s, k_a) = \mathsf{KDF}(v)$ are randomly distributed. For the decapsulation not returning $\perp$, the adversary had to come up with the tag $t$ satisfying $t = \mathsf{MAC}_{k_a}((u_1^*)^r, (u_2^*)^r)$, which is computationally hard since $k_a$ is random and $\mathsf{MAC}$ is assumed secure.

Our use of $\mathsf{MAC}$ is different from the counterpart in the hybrid PKE [15] in its input. In [15], $\mathsf{MAC}$ is used to authenticate a symmetrically encrypted plaintext $e$. Namely, using our notations, in [15], $e \leftarrow \mathsf{SymmetricEncryption}_{k_s}(\mathtt{plaintext})$ and then $t \leftarrow \mathsf{MAC}_{k_a}(e)$. In contrast, in Figure 1, we take "early" $\mathsf{MAC}$ on $(u_1, u_2)$. Nevertheless, the resemblance between our KEM and the hybrid PKE allows us to re-utilize the proof in the hybrid encryption case.

### 3.2 Comparison and implementation

**Base group.** There are primarily two choices for the group $\mathbb{G}$ so that DDH assumption is believed holds true. The first choice is to take $\mathbb{G}$ as the order $q$, multiplicative subgroup of $\mathbb{Z}_p^*$ in which $p = 1 \pmod q$ is a prime. The elements in $\mathbb{G}$ are thus represented modulo $p$, and hence of $|p| = 1024$ bits (for 80-bit security) or $|p| = 3072$ bits (for 128-bit security). See [13] for more details.

The second choice of $\mathbb{G}$ is to take elliptic curve groups of order $q$. This choice reduces the length of element representation, since the length of $q$ in bits can be $|q| = 160$ (for 80-bit security), or $|q| = 256$ (for 128-bit security). See [25] for specific curves.

**Theoretical comparison** In Table 2, we compare our KEMs with the ACE-KEM in ISO/IEC 18033-2 [1], which refined the schemes in [12,13]. Both enjoys a tight security reduction to the DDH assumption. Since the tag size $|t|$ can be 128 in our KEMs, our encapsulation size is slightly shorter than ACE-KEM. The public key in our KEMs is one group element shorter.

**Table 2.** Comparison of KEMs in standard model based on the DDH assumption. Abbreviations in the table: **me** = multi-exponentiation, **se** = single-exponentiation, **gmc** = group membership check, **el** = group element.

| **Scheme** | **Assumption** | **Encap length** | **[Encap]; [Decap] main costs of computation** | $[pk, sk]$ **size** |
|---|---|---|---|---|
| ACE-KEM [1] | DDH | $3\|q\|$ | [1 **me**, 3 **se**]; [0 **me**, 3 **se**, 1 **gmc**] | [5 **el**, 4 **el**] |
| Ours, Figure 1 | DDH | $2\|q\| + \|t\|$ | [1 **me**, 2 **se**]; [1 **me**, 0 **se**, 2 **gmc**] | [4 **el**, 4 **el**] |

To compare computation costs, we consider ACE-KEM implemented a group of prime order $q$. We use the result that one multi-exponentiation in that group can be carried out in $(1 + 2/\log_2 \log_2 q) \log_2 q$ multiplications [9], therefore can be counted as approximately 1.2 single exponentiation, which is confirmed by experiments in Section C.

First, in groups where group membership checks are trivial, our KEM in Figure 1 needs just one multi-exponentiation, thus beating the ACE-KEM at dramatic margin of 60% in decapsulation speed. Examples of the groups include NIST elliptic curves [25] defined over prime fields (P-192, P-224, P-256, P-384, P-521) and binary fields (B-163, B-233, B-283, B-409, B-571).

Now assume that a group membership check is costly as one single exponentiation, while more efficient methods (e.g., using the Legendre symbol) may be available depending on the base group [13, Section 4.2]. Using abbreviations in Table 2, we count: 1 **me** = 1.2 **se**, 1 **gmc** = 1 **se**.

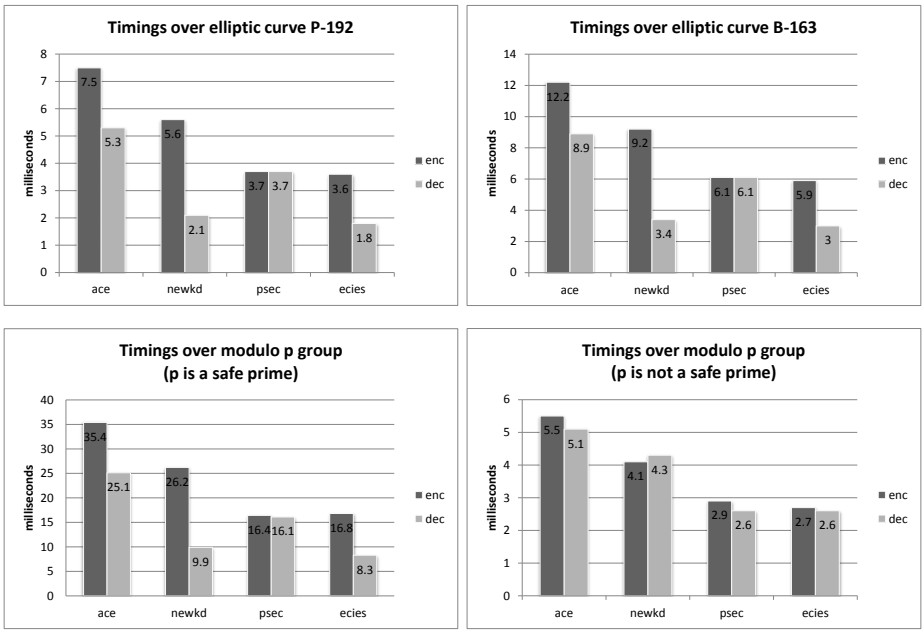
Thus our encapsulation needs 3.2 (se), while that for ACE-KEM is 4.2 (se), meaning more than 20% improvement in speed. For decapsulation, our schemes in Figures 1 and 7 require 3.2 or 3 (se), while that of ACE-KEM is 4 (se), yielding at least 20% improvement.

Our KEM decapsulation speed is even either faster or comparable with standardized PSEC-KEM and ECIES-KEM schemes whose security proofs are not in the standard model. Interested readers can find more details in Section B.

**Experimental comparison** ISO/IEC 18033-2 comes with a reference implementation, written by Anshuman Rawat and Victor Shoup (see website of [1]). The implementation, among others, includes ACE-KEM, PSEC-KEM, and ECIES-KEM. We add an implementation of our proposed KEM based on that library. Timings of encryption and decryption are reported in Figure 2, in which our scheme in Figure 1 is named "newkd". The codes in [1] neither speed up multi-exponentiation nor use Legendre symbol for group membership check. Our code elaborates on these aspects by

- employing a square-and-multiply algorithm for multi-exponentiation (recalled in Section C), and
- using Legendre symbol for group membership check in $\mathbb{G} \subset \mathbb{Z}_p^*$ where $p$ is a safe (aka, Sophie Germain) prime, namely $p = 2q + 1$ for a prime $q$.

**Fig. 2.** Average timings, taken over 10000 executions, over different base groups. Experiment is done over a laptop (Intel 2.0GHz CPU, 8GB RAM) running Ubuntu 12.04 LTS. The C compiler is g++ 4.6.3 using NTL 6.0.0 and GMP 5.1.1 libraries.




Over all groups, one can confirm by Figure 2 that our proposed "newkd" is more efficient than ACE-KEM in both encapsulation and decapsulation. The bar charts also fit above theoretical comparisons.

Whenever above speedup tricks are applicable, namely over NIST's elliptic curves or over $\mathbb{G} \subset \mathbb{Z}_p^*$ with safe prime $p$, one can confirm that our proposal's decapsulation is faster than PSEC-KEM, and is even comparable to ECIES-KEM.

Over a subgroup $\mathbb{G} \subset \mathbb{Z}_p^*$ where $p$ is not a safe prime, the decapsulation speed of "newkd" decreases. Here, two group membership checks, performed by two exponentiations, must be done since the Legendre symbol trick cannot be applied.

### 3.3 Security proof

This subsection is devoted to prove the following theorem.

**Theorem 1** *The KEM in Figure 1 is IND-CCA-secure under the DDH assumption.*

The following proof is similar to [15], adjusted for our KEM.

*Proof.* We will proceed in games, each of which is a modification of the previous one. Below, $\Pr[X_i] = \Pr[b' = b \text{ in } \mathbf{Game}_i]$.

$\mathbf{Game}_0$: This game is the IND-CCA attack game with an adversary $\mathcal{A}$. Recall that $\kappa$ is the security parameter, and $\mathbf{Adv}^{\mathrm{ind-cca}}_{\mathcal{A}}(\kappa) = |\Pr[b' = b] - \frac{1}{2}|$.

The challenge is $(C^*, K(b))$ where $C^* = (u_1^*, u_2^*, t^*)$. We denote by $r^*$, $\alpha^*$, $v^*$, $k_s^*, k_a^*$ the corresponding intermediate quantities. The key $K(b)$ is $(k_s^*, k_a^*)$ or random depending on the bit $b$.

$\mathbf{Game}_1$: The challenge oracle uses secrets $(x_1, y_1, x_2, y_2)$ to compute $v^*$. Namely,

$$v^* = (u_1^*)^{x_1 + \alpha^* y_1} (u_2^*)^{x_2 + \alpha^* y_2}$$

where $u_1^* = g_1^{r^*}, u_2^* = g_2^{r^*}$ and $\alpha^* = \mathsf{TCR}(u_1^*, u_2^*)$.

Moreover, for any query $(u_1, u_2, t)$ with $(u_1, u_2) \neq (u_1^*, u_2^*)$ and $\mathsf{TCR}(u_1, u_2) = \mathsf{TCR}(u_1^*, u_2^*)$, the decapsulation oracle returns $\perp$.

Then there exists a poly-time adversary $\mathcal{A}_1$ such that

$$|\Pr[X_0] - \Pr[X_1]| \leq \mathbf{Adv}^{\mathsf{TCR}}_{\mathcal{A}_1}(\kappa) \tag{1}$$

since the first change is notational, and the second one is based on the security of $\mathsf{TCR}$. More formally, $\mathcal{A}_1$ gets inputs $(u_1^*, u_2^*)$, and simulates the environment for $\mathcal{A}$ by generating the public and secret keys. $\mathcal{A}_1$ gives $\mathcal{A}$ the public key, and answers $\mathcal{A}$'s decapsulation queries using the secret key. In any decapsulation query $(u_1, u_2, t)$, if $(u_1, u_2) \neq (u_1^*, u_2^*)$ and $\mathsf{TCR}(u_1, u_2) = \mathsf{TCR}(u_1^*, u_2^*)$, then $\mathcal{A}_1$ stops the simulation and returns the pair $(u_1, u_2)$ as its output. The running time of $\mathcal{A}_1$ in the worst case is that of $\mathcal{A}$ plus time for doing arithmetic computations in $\mathbb{G}$ and time for some symmetric operations, so is of polynomial time.

$\mathbf{Game}_2$: In this game, elements $u_1^*$ and $u_2^*$ are computed as follows: $r_1^* \xleftarrow{\$} \mathbb{Z}_q, u_1^* \leftarrow g_1^{r_1^*}$, and $r_2^* \xleftarrow{\$} \mathbb{Z}_q \setminus \{r_1^*\}, u_2^* \leftarrow g_2^{r_2^*}$. Then there is a poly-time adversary $\mathcal{A}_2$ such that

$$|\Pr[X_1] - \Pr[X_2]| = \mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{A}_2}(\kappa). \tag{2}$$

The description of $\mathcal{A}_2$ is as follows. Its input is a tuple $(g_1, g_2, u_1^*, u_2^*)$. $\mathcal{A}_2$ itself generates the secret key, and then coupling with generators $g_1, g_2$ of $\mathbb{G}$, it computes the public key. Since $\mathcal{A}_2$ holds the secret key, it can answer all decapsulation queries from $\mathcal{A}$. The adversary $\mathcal{A}_2$ controls the hidden bit $b$, so that it can compare that bit with $\mathcal{A}$'s output bit $b$. In case $b' = b$, $\mathcal{A}_2$ returns 1; otherwise it returns 0. Any difference on the output $b'$ of $\mathcal{A}$ depending on tuple $(g_1, g_2, u_1^*, u_2^*)$ directly yields a difference on the probability $\mathcal{A}_2$ outputting 1, so that above equation claim is justified. The running time of $\mathcal{A}_2$ in the worst case is that of $\mathcal{A}$ plus time for doing arithmetic computations in $\mathbb{G}$ and time for some symmetric operations, so is of polynomial time.

$\mathbf{Game}_3$: This game makes use of $\omega \in \mathbb{Z}_q^*$ satisfying $g_2 = g_1^\omega$. With $\omega$, we can check in poly-time whether $\log_{g_1} u_1 = \log_{g_2} u_2$ by simply verifying $u_1^\omega = u_2$. Denote $\mathcal{V} = \{(u_1, u_2) \in \mathbb{G}^2 : u_1^\omega = u_2\}$. In this game, any decapsulation query $(u_1, u_2, t)$

**Fig. 3.** Oracles in $\mathbf{Game}_3$ for the proof of Theorem 1.

| Initialization of the game | Decapsulation of adversarial query $C = (u_1, u_2, t)$ |
|---|---|
| **I1:** $\omega \xleftarrow{\$} \mathbb{Z}_q^*$, $g_2 \leftarrow g_1^{\omega}$<br>**I2:** $(x_1, x_2, y_1, y_2) \xleftarrow{\$} \mathbb{Z}_q^4$<br>$c \leftarrow g_1^{x_1} g_2^{x_2}$, $d \leftarrow g_1^{y_1} g_2^{y_2}$<br>**I3:** $r_1^* \xleftarrow{\$} \mathbb{Z}_q$, $u_1^* \leftarrow g_1^{r_1^*}$<br>$r_2^* \xleftarrow{\$} \mathbb{Z}_q \setminus \{r_1^*\}$, $u_2^* \leftarrow g_2^{r_2^*}$<br>**I4:** $\alpha^* \leftarrow \mathsf{TCR}(u_1^*, u_2^*)$<br>$v^* \leftarrow (u_1^*)^{x_1+\alpha^* y_1} (u_2^*)^{x_2+\alpha^* y_2}$<br>**I5:** $(k_s^*, k_a^*) \leftarrow \mathsf{KDF}(v^*)$ | 1: $\alpha = \mathsf{TCR}(u_1, u_2)$<br>2: **if** $(u_1, u_2) \neq (u_1^*, u_2^*)$ and $\alpha = \alpha^*$ **then**<br>3: **return** $\perp$<br>4: **end if**<br>5: **if** $(u_1, u_2) = (u_1^*, u_2^*)$ **then**<br>6: **if** $t \neq \mathsf{MAC}_{k_a^*}(u_1^*, u_2^*)$ **then return** $\perp$<br>7: **else return** $k_s^*$<br>8: **else if** $(u_1, u_2) \notin \mathcal{V}$ **then**<br>9: $\alpha \leftarrow \mathsf{TCR}(u_1, u_2)$<br>10: $v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$<br>11: $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$<br>12: **if** $t \neq \mathsf{MAC}_{k_a}(u_1, u_2)$ **then return** $\perp$<br>13: **else return** $\perp$ {Rejection rule in $\mathbf{Game}_3$}<br>14: **else**<br>15: $\alpha \leftarrow \mathsf{TCR}(u_1, u_2)$<br>16: $v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$<br>17: $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$<br>18: **if** $t \neq \mathsf{MAC}_{k_a}(u_1, u_2)$ **then return** $\perp$<br>19: **else return** $k_s$<br>20: **end if** |

with $(u_1, u_2) \notin \mathcal{V}$ is rejected. The initialization and decapsulation oracle in this game are depicted in Figure 3.

Let $F_i$ ($i \geq 3$) be the event that a query is rejected at line 13 of the decapsulation oracle in $\mathrm{Game}_i$. Let $Q$ be the bound on the total number of decapsulation queries $\mathcal{A}$ makes, we have

$$|\Pr[X_2] - \Pr[X_3]| \leq Q \Pr[F_3]. \tag{3}$$

$\mathbf{Game}_4$: In this game, take $v^* \xleftarrow{\$} \mathbb{G}$ (at line **I4**) and $v \xleftarrow{\$} \mathbb{G}$ (at line 10 in the decapsulation). This is because

$$\begin{bmatrix} \log_{g_1} c \\ \log_{g_1} d \\ \log_{g_1} v^* \\ \log_{g_1} v \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \omega & 0 \\ 0 & 1 & 0 & \omega \\ r_1^* & r_1^*\alpha^* & r_2^*\omega & r_2^*\omega\alpha^* \\ r_1 & r_1\alpha & r_2\omega & r_2\omega\alpha \end{bmatrix}}_{M} \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$

and determinant $\det(M) = \omega^2(r_2^* - r_1^*)(r_2 - r_1)(\alpha - \alpha^*) \neq 0$ shows that $(c, d, v^*, v)$ are uniformly distributed as $(x_1, y_1, x_2, y_2)$ are. We have

$$\Pr[X_3] = \Pr[X_4] \tag{4}$$

$$\Pr[F_3] = \Pr[F_4]. \tag{5}$$

**Game**$_5$: At line **I5**, take $(k_s^*, k_a^*) \xleftarrow{\$} \{0,1\}^{2n}$. This is because $v^*$ is taken randomly in the previous game. Then there exists an adversary $\mathcal{A}_5$ against $\mathsf{KDF}$ such that

$$|\Pr[X_4] - \Pr[X_5]| \leq \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}_5}(\kappa). \tag{6}$$

The description of $\mathcal{A}_5$ is as follows. Its input is a string in $\{0,1\}^{2n}$. It uses the input for the keys $(k_s^*, k_a^*)$ at line **I5**, while generating the secret key and public key and others as in lines **I1** to **I4**. Since $\mathcal{A}_2$ holds the trapdoor for membership testing $\omega$ and the secret key, it can handle decapsulation queries as in Figure 3. When $\mathcal{A}$ returns $b'$, the adversary $\mathcal{A}_5$ checks whether $b'$ equals its chosen bit $b$. If $b' = b$, $\mathcal{A}_5$ returns 1. The running time of $\mathcal{A}_5$ in the worst case is that of $\mathcal{A}$ plus time for doing arithmetic computations in $\mathbb{G}$ and time for some symmetric operations, so is of polynomial time.

**Game**$_6$: At line 7 in the decapsulation, return $\perp$. This is because $(u_1, u_2) = (u_1^*, u_2^*)$ with probability $\frac{1}{q^2}$ before the challenge phase. Moreover, after the challenge phase when $(u_1^*, u_2^*, t^*)$ was already announced, querying $(u_1^*, u_2^*, t)$ with $t = \mathsf{MAC}_{k_a^*}(u_1^*, u_2^*)$ and $t \neq t^*$ to the oracle means the adversary can break the MAC. We have

$$|\Pr[X_5] - \Pr[X_6]| \leq Q\left(\frac{1}{q^2} + \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}_6}(\kappa)\right) \text{ and } \Pr[X_6] = \frac{1}{2} \tag{7}$$

since $(k_s^*, k_a^*)$ are perfectly random in this game.

The description of $\mathcal{A}_6$ is as follows. Its input is $(u_1^*, u_2^*, t^*)$ where $t^* = \mathsf{MAC}_{k_a^*}(u_1^*, u_2^*)$ for random key $k_a^*$. It generates the secret key and then simulates the environment for $\mathcal{A}$. Whenever $\mathcal{A}$ queries $(u_1, u_2, t)$ for decapsulation in which $t \neq t^*$ and $t = \mathsf{MAC}_{k_a^*}(u_1^*, u_2^*)$, the adversary $\mathcal{A}_6$ halts the simulation and returns $(u_1^*, u_2^*, t)$. The running time of $\mathcal{A}_6$ in the worst case is that of $\mathcal{A}$ plus time for doing arithmetic computations in $\mathbb{G}$ and time for some symmetric operations, so is of polynomial time.

**Game**$_{5'}$: Now we move back to consider **Game**$_4$ again. This game is the same as **Game**$_4$, except that, $(k_s, k_a) \xleftarrow{\$} \{0,1\}^{2n}$ at line 11. We have

$$|\Pr[F_4] - \Pr[F_{5'}]| \leq \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}'_5}(\kappa). \tag{8}$$

Since the MAC key has been turned random,

$$\Pr[F_{5'}] \leq \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}''_5}(\kappa) \tag{9}$$

in which, as a recall, $F_{5'}$ is the event that a query is rejected at line 13 of the decapsulation oracle in this game. The descriptions of adversaries $\mathcal{A}'_5$ against $\mathsf{KDF}$ and $\mathcal{A}''_5$ against $\mathsf{MAC}$ are similar to those in **Game**$_5$ and **Game**$_6$.

By (5), (8), (9), we have

$$\Pr[F_3] = \Pr[F_4] \leq \Pr[F_{5'}] + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}'_5}(\kappa) \leq \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}''_5}(\kappa) + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}'_5}(\kappa) \tag{10}$$

and by (1), (2), (3), (4), (6), (7), and the bound (10),

$$\mathbf{Adv}^{\mathrm{ind-cca}}_{\mathcal{A}}(\kappa) \le \mathbf{Adv}^{\mathsf{TCR}}_{\mathcal{A}_1}(\kappa) + \mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{A}_2}(\kappa) + Q\left(\mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}''_5}(\kappa) + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}'_5}(\kappa)\right) + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}_5}(\kappa) + Q\left(\frac{1}{q^2} + \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}_6}(\kappa)\right)$$

ending the proof. □

# 4 Generalization to universal hash proof system

## 4.1 Hash proof system

The notion of hash proof systems was introduced by Cramer and Shoup [14]. Let $\mathcal{SK}, \mathcal{PK}$, and $\mathcal{K}$ be sets of secret keys, public keys, and encapsulated symmetric keys. Let $\mathcal{E}$ be the set of all "valid" and "invalid" encapsulation, and $\mathcal{V} \subset \mathcal{E}$ be the set of all "valid" ones. To illustrate the above notation, in the DDH-based scheme, $\mathcal{SK} = \mathbb{G}^4$, $\mathcal{PK} = \mathbb{G}^2$, $\mathcal{E} = \mathbb{G}^2$, $\mathcal{K} = \mathbb{G}$, $\mathcal{V} = \{(g_1^r, g_2^r) : r \in \mathbb{Z}_q\}$.

The subset membership assumption says that $\mathcal{V}$ is indistinguishable from $\mathcal{E}$. If $\mathcal{V} = \{(g_1^r, g_2^r) : r \in \mathbb{Z}_q\}$ and $\mathcal{E} = \mathbb{G}^2$ as above, this is exactly the DDH assumption. Formally, the advantage

$$\mathbf{Adv}^{\mathrm{sm}}_{\mathcal{D}}(\kappa) = \left| \Pr_{U \xleftarrow{\$} \mathcal{E}}[\mathcal{D}(U) = 1] - \Pr_{U \xleftarrow{\$} \mathcal{V}}[\mathcal{D}(U) = 1] \right|$$

is negligible for all poly-time distinguishers $\mathcal{D}$.

A function $\Lambda_{sk} : \mathcal{E} \to \mathcal{K}$ is *projective* if there exists a projection $\mu : \mathcal{SK} \to \mathcal{PK}$ such that $pk = \mu(sk)$ defines $\Lambda_{sk} : \mathcal{V} \to \mathcal{K}$. Namely, for every $E \in \mathcal{V}$, the value $K = \Lambda_{sk}(E)$ is uniquely determined by $pk = \mu(sk)$ and $E$. As an example, in our scheme of Sect.3, $\Lambda_{sk}\big(E = (u_1, u_2)\big) = u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$ where $\alpha = \mathsf{TCR}(E)$.

A projective function $\Lambda_{sk}$ is called computationally universal-2 [20] if for all $E, E' \notin \mathcal{V}$ with $E \neq E'$,

$$\Big(pk, \Lambda_{sk}(E'), \Lambda_{sk}(E)\Big) \text{ and } \Big(pk, \Lambda_{sk}(E'), K\Big)$$

are computationally indistinguishable, where $sk \xleftarrow{\$} \mathcal{SK}$ and $K \xleftarrow{\$} \mathcal{K}$. Formally, consider an adversary $\mathcal{A} = (\mathcal{A}_{\mathrm{find}}, \mathcal{A}_{\mathrm{guess}})$ in the following experiment.

**Experiment** $\mathbf{Exp}^{\mathrm{cu2}}_{\mathcal{A}}(\kappa)$ :

$\big(group, \mathcal{SK}, \mathcal{PK}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \Lambda_{(\cdot)}(\cdot), \mu\big) \leftarrow \mathsf{Param}(1^\kappa)$

$sk \xleftarrow{\$} \mathcal{SK}, pk \leftarrow \mu(sk), E' \xleftarrow{\$} \mathcal{E} \setminus \mathcal{V}, K' \leftarrow \Lambda_{sk}(E')$

$(E \in \mathcal{E} \setminus \mathcal{V}, \mathtt{state}) \leftarrow \mathcal{A}_{\mathrm{find}}^{\mathsf{Eval}_{sk}(\cdot)}(pk, E', K')$

$b \xleftarrow{\$} \{0,1\}, K(0) \leftarrow \Lambda_{sk}(C), K(1) \xleftarrow{\$} \mathcal{SK}$

$b' \leftarrow \mathcal{A}_{\mathrm{guess}}(\mathtt{state}, K(b))$

If $b' = b$ then return 1 else return 0

**Fig. 4.** Our generic KEM from hash proof system $(\mathsf{Param}, \mathsf{Pub}, \mathsf{Priv})$.

| $\underline{\mathsf{KG}(1^\kappa)}$ : | $\underline{\mathsf{Encap}(pk)}$ : | $\underline{\mathsf{Decap}(sk, C)}$ : |
|---|---|---|
| Run $\mathsf{Param}$ to define | Take random witness $r$ | Parse $C = (E, t)$ |
| $(group, \mathcal{SK}, \mathcal{PK}, \mathcal{K},$ | $E = E(r) \xleftarrow{\$} \mathcal{V}$ | $v \leftarrow \mathsf{Priv}(sk, E)$ |
| $\mathcal{E}, \mathcal{V}, \Lambda_{(\cdot)}(\cdot), \mu)$ | $v \leftarrow \mathsf{Pub}(pk, E, r)$ | $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$ |
| $sk \xleftarrow{\$} \mathcal{SK}$ | $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$ | If $t = \mathsf{MAC}_{k_a}(E)$ |
| $pk \leftarrow \mu(sk)$ | $t \leftarrow \mathsf{MAC}_{k_a}(E)$ | return $k_s$ |
| Return $(pk, sk)$ | Return $C = (E, t)$ and $K = k_s$ | Else return $\perp$ |

where the oracle $\mathsf{Eval}_{sk}(F)$ returns $\Lambda_{sk}(F)$ if $F \in \mathcal{V}$ and $\perp$ otherwise. Computational universality requires that

$$\mathbf{Adv}^{\mathrm{cu2}}_{\mathcal{A}}(\kappa) = \Pr[\mathbf{Exp}^{\mathrm{cu2}}_{\mathcal{A}}(\kappa) = 1]$$

is negligible for all poly-time $\mathcal{A}$.

**Hash proof system.** A hash proof system $\mathcal{HPS}$ consists of three algorithms $(\mathsf{Param}, \mathsf{Pub}, \mathsf{Priv})$ described as follows. Algorithm $\mathsf{Param}(1^\kappa)$ first generates the description of $group$, $\mathcal{SK}$, $\mathcal{PK}$, $\mathcal{K}$, $\mathcal{E}$, $\mathcal{V}$, $\Lambda_{(\cdot)}(\cdot)$, and $\mu : \mathcal{SK} \to \mathcal{PK}$. Algorithm $\mathsf{Pub}(pk, E, r)$ returns $K = \Lambda_{sk}(E)$ for $E \in \mathcal{V}$, where the computation does not use $sk$ but makes use of $r$, a witness of the fact that $E \in \mathcal{V}$. Algorithm $\mathsf{Priv}(sk, E)$ returns $\Lambda_{sk}(E)$.

### 4.2 IND-CCA-secure KEM from hash proof systems

The KEM is depicted in Figure 4. The descriptions of symmetric building blocks $\mathsf{KDF}$ and $\mathsf{MAC}$ are in Section 2.

**Theorem 2** *The generic construction of KEM in Figure 4 is IND-CCA-secure.*

*Proof.* We proceed in games as follows.

**Game**$_0$: This game is the IND-CCA attack game with leakage. Without loss of generality, assume that $E^*, r^*$ are generated at the beginning of the game.

**Game**$_1$: Compute $\mathsf{Pub}(pk, E^*, r^*)$ in the challenge encapsulation as $\mathsf{Priv}(sk, E^*)$. This change is only notational since $\mathsf{Priv}(sk, E^*) = \mathsf{Pub}(pk, E^*, r^*) = \Lambda_{sk}(E^*)$ so that $\Pr[X_0] = \Pr[X_1]$.

**Game**$_2$: Take $E^* \xleftarrow{\$} \mathcal{C} \setminus \mathcal{V}$. We have

$$|\Pr[X_1] - \Pr[X_2]| \leq \mathbf{Adv}^{\mathrm{sm}}_{\mathcal{A}_2}(\kappa) \tag{11}$$

thanked to the subset membership problem. The running time of $\mathcal{A}_2$ in the worst case is that of $\mathcal{A}$ plus time for doing some computations in the hash proof systems and time for some symmetric operations, so is of polynomial time.

**Game**$_3$: Any decapsulation query $(E, t)$ with $E \neq E^*$ and $E \notin \mathcal{V}$ is answered by $\perp$. Let $Q$ be the total number of decapsulation queries, we have

$$|\Pr[X_2] - \Pr[X_3]| \leq Q \Pr[F_3] \tag{12}$$

where $F_3$ is the event that a query is rejected by the above rule. The initialization and the decapsulation oracle are depicted in Figure 5, in which $F_3$ happens whenever line 8 of decapsulation is reached.

**Fig. 5.** Oracles in **Game**$_3$ for the proof of Theorem 2.

| Initialization of the game | Decapsulation of adversarial query $C = (E, t)$ |
|---|---|
| **I1:** $\omega \xleftarrow{\$} \mathsf{Trapdoors}$<br>**I2:** $sk \xleftarrow{\$} \mathcal{SK}$, $pk \leftarrow \mu(sk)$<br>**I3:** $E^* \xleftarrow{\$} \mathcal{C} \setminus \mathcal{V}$<br>**I4:** $v^* \leftarrow \mathsf{Priv}(sk, E^*)$<br>**I5:** $(k_s^*, k_a^*) \leftarrow \mathsf{KDF}(v^*)$ | 1: **if** $E = E^*$ **then**<br>2: **if** $t \neq \mathsf{MAC}_{k_a^*}(E^*)$ **then return** $\perp$<br>3: **else return** $k_s^*$<br>4: **else if** $E \notin \mathcal{V}$ **then**<br>5: $v \leftarrow \mathsf{Priv}(sk, E)$<br>6: $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$<br>7: **if** $t \neq \mathsf{MAC}_{k_a}(E)$ **then return** $\perp$<br>8: **else return** $\perp$<br>9: **else**<br>10: $v \leftarrow \mathsf{Priv}(sk, E)$<br>11: $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$<br>12: **if** $t \neq \mathsf{MAC}_{k_a}(E)$ **then return** $\perp$<br>13: **else return** $k_s$<br>14: **end if** |

**Game**$_4$: In this game, take $v^* \xleftarrow{\$} \mathcal{K}$ (at line **I4**) and $v \xleftarrow{\$} \mathcal{K}$ (at line 5 in the decapsulation). We have

$$|\Pr[X_3] - \Pr[X_4]| \leq \mathbf{Adv}^{\mathrm{cu2}}_{\mathcal{A}_4}(\kappa) \tag{13}$$

$$|\Pr[F_3] - \Pr[F_4]| \leq \mathbf{Adv}^{\mathrm{cu2}}_{\mathcal{A}'_4}(\kappa) \tag{14}$$

where event $F_4$ happens whenever line 8 of decapsulation is reached in this game. The reasons are that $v = \Lambda_{sk}(E)$ is computationally random conditioned on $pk, v^* = \Lambda_{sk}(E^*)$; and that $v^* = \Lambda_{sk}(E^*)$ is computationally random conditioned on $pk, v$ thanks to the computational universality of the hash proof system.

**Game**$_5$: At line **I5**, take $(k_s^*, k_a^*) \xleftarrow{\$} \{0,1\}^{2n}$. This is because $v^*$ is taken randomly in the previous game. Then there exists an adversary $\mathcal{A}_5$ against $\mathsf{KDF}$ such that

$$|\Pr[X_4] - \Pr[X_5]| \leq \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}_5}(\kappa). \tag{15}$$

The description of $\mathcal{A}_5$ is the same as its counterpart in the proof of Theorem 1.

**Game**$_6$: At line 3 in the decapsulation, return $\perp$. This is because $E = E^*$ with probability $\frac{1}{|\mathcal{E}|}$ before the challenge phase. Moreover, after the challenge phase when $(E^*, t^*)$ was already announced, querying $(E^*, t)$ with $t = \mathsf{MAC}_{k_a^*}(E^*)$ and $t \neq t^*$ to the oracle means the adversary can break the MAC. We have

$$|\Pr[X_5] - \Pr[X_6]| \leq Q\left(\frac{1}{|\mathcal{E}|} + \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}_6}(\kappa)\right) \text{ and } \Pr[X_6] = \frac{1}{2} \tag{16}$$

since $(k_s^*, k_a^*)$ are perfectly random in this game.

The description of $\mathcal{A}_6$ is as follows. Its input is $(E^*, t^*)$ where $t^* = \mathsf{MAC}_{k_a^*}(E^*)$ for random key $k_a^*$. It generates the secret key and then simulates the environment for $\mathcal{A}$. Whenever $\mathcal{A}$ queries $(E, t)$ for decapsulation in which $t \neq t^*$ and $t = \mathsf{MAC}_{k_a^*}(E^*)$, the adversary $\mathcal{A}_6$ halts the simulation and returns $(E^*, t)$.

**Game**$_{5'}$: Now we move back to consider **Game**$_4$ again. This game is the same as **Game**$_4$, except that, $(k_s, k_a) \xleftarrow{\$} \{0,1\}^{2n}$ at line 6. We have

$$|\Pr[F_4] - \Pr[F_{5'}]| \leq \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}'_5}(\kappa). \tag{17}$$

The description of $\mathcal{A}_5$ is the same as its counterpart in the proof of Theorem 1. Since the MAC key $k_a$ has been turned random,

$$\Pr[F_{5'}] \leq \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}''_5}(\kappa) \tag{18}$$

in which, as a recall, $F_{5'}$ is the event that a query is rejected at line 8 of the decapsulation oracle in this game. The descriptions of adversaries $\mathcal{A}'_5$ against $\mathsf{KDF}$ and $\mathcal{A}''_5$ against $\mathsf{MAC}$ are similar to those in **Game**$_5$ and **Game**$_6$.

By (14), (17), and (18),

$$\Pr[F_3] \leq \mathbf{Adv}^{\mathrm{cu2}}_{\mathcal{A}'_4}(\kappa) + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}'_5}(\kappa) + \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}''_5}(\kappa) \tag{19}$$

Summing up (11), (12), (13), (15), (16), and (19),

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ind-cca}}_{\mathcal{A}}(\kappa) \leq\ & \mathbf{Adv}^{\mathrm{sm}}_{\mathcal{A}_2}(\kappa) + Q\left(\mathbf{Adv}^{\mathrm{cu2}}_{\mathcal{A}'_4}(\kappa) + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}'_5}(\kappa) + \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}''_5}(\kappa)\right) \\ & + \mathbf{Adv}^{\mathrm{cu2}}_{\mathcal{A}_4}(\kappa) + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}_5}(\kappa) + Q\left(\frac{1}{|\mathcal{E}|} + \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}_6}(\kappa)\right)\end{aligned}$$

ending the proof. □

### 4.3 Instantiation under the DLIN assumption

We use the HPS based on the decisional linear assumption (DLIN) given by [20]. In this HPS, $\mathcal{SK} = \mathbb{Z}_q^6$, $\mathcal{PK} = \mathbb{G}^4$, $\mathcal{K} = \mathbb{G}$. Also $\mathcal{E} = \mathbb{G}^3$ and $\mathcal{V} = \{(g_1^{r_1}, g_2^{r_2}, h^{r_1+r_2}) : r_1, r_2 \in \mathbb{Z}_q\}$, where $g_1, g_2, h \in \mathbb{G}$. The DLIN assumption asserts that $\mathcal{E}$ and $\mathcal{V}$ are indistinguishable. The projective function is

$$\Lambda_{sk}(u_1, u_2, u_3) = u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2} u_3^{z+\alpha z'} \Longleftrightarrow \Lambda_{sk}(u_1, u_2, u_3) = (c_1 d_1^{\alpha})^{r_1} (c_2 d_2^{\alpha})^{r_2}$$

using the same notations as in Figure 6. To check $E \in \mathcal{E} \setminus \mathcal{V}$ in Figure 5, use trapdoors $\log_{g_1} h \in \mathbb{Z}_q$ and $\log_{g_2} h \in \mathbb{Z}_q$.

**Fig. 6.** Our DLIN-based KEM (above) and DCR-based KEM (below).

| $\underline{\mathsf{KG}(1^\kappa):}$ | $\underline{\mathsf{Encap}(pk):}$ | $\underline{\mathsf{Decap}(sk,C):}$ |
|---|---|---|
| $g_1, g_2, h \xleftarrow{\$} \mathbb{G}$ | $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q$ | Parse $C = (u_1, u_2, u_3, t)$ |
| $(x_1, x_2, y_1, y_2) \xleftarrow{\$} \mathbb{Z}_q^4$ | $u_1 \leftarrow g_1^{r_1}, u_2 \leftarrow g_2^{r_2}$ | $\alpha \leftarrow \mathsf{TCR}(u_1, u_2, u_3)$ |
| $(z, z') \xleftarrow{\$} \mathbb{Z}_q^2$ | $u_3 \leftarrow h^{r_1+r_2}$ | $v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2} u_3^{z+\alpha z'}$ |
| $c_1 \leftarrow g_1^{x_1} h^z, c_2 \leftarrow g_2^{x_2} h^z$ | $\alpha \leftarrow \mathsf{TCR}(u_1, u_2, u_3)$ | $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$ |
| $d_1 \leftarrow g_1^{y_1} h^{z'}, d_2 \leftarrow g_2^{y_2} h^{z'}$ | $v \leftarrow (c_1 d_1^\alpha)^{r_1} (c_2 d_2^\alpha)^{r_2}$ | If $t = \mathsf{MAC}_{k_a}(u_1, u_2, u_3)$ |
| $pk \leftarrow (g_1, g_2, h,$ | $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$ | return $k_s$ |
| $c_1, d_1, c_2, d_2)$ | $t \leftarrow \mathsf{MAC}_{k_a}(u_1, u_2, u_3)$ | Else |
| $sk \leftarrow (x_1, x_2, y_1, y_2, z, z')$ | Return $C = (u_1, u_2, u_3, t)$ | return $\perp$ |
| Return $(pk, sk)$ | and $K = k_s$ | |

| $\underline{\mathsf{KG}(1^\kappa):}$ | $\underline{\mathsf{Encap}(pk):}$ | $\underline{\mathsf{Decap}(sk,C):}$ |
|---|---|---|
| $g \xleftarrow{\$} \mathbb{G}, g_2 \leftarrow g^{N_1}$ | $r \xleftarrow{\$} \{0, \ldots, N_1/4\}$ | Parse $C = (u, t)$ |
| $(x, y) \xleftarrow{\$} \mathcal{SK}$ | $u \leftarrow g_2^r \bmod N_1^2$ | $\alpha \leftarrow \mathsf{TCR}(u)$ |
| $c \leftarrow g_2^x \bmod N_1^2$ | $\alpha \leftarrow \mathsf{TCR}(u)$ | $v \leftarrow u^{x+y\alpha} \bmod N_1$ |
| $d = g_2^y \bmod N_1^2$ | $v \leftarrow (cd^\alpha)^r \bmod N_1$ | $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$ |
| $pk = (N_1, g_2, c, d)$ | $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$ | If $t = \mathsf{MAC}_{k_a}(u)$ |
| $sk \leftarrow (x, y)$ | $t \leftarrow \mathsf{MAC}_{k_a}(u)$ | return $k_s$ |
| Return $(pk, sk)$ | Return $C = (u, t)$ and $K = k_s$ | Else return $\perp$ |

**Lemma 1 (Lemma 6.3 in [20]).** *The above hash proof system is computationally universal-2 if* $\mathsf{TCR}$ *is target collision resistant.*

Our DLIN-based KEM appears in Figure 6. The symmetric building blocks are $\mathsf{TCR} : \mathbb{G}^3 \to \mathbb{Z}_q$, $\mathsf{KDF} : \mathbb{G} \to \{0,1\}^{2n}$, and $\mathsf{MAC} : \{0,1\}^n \times \mathbb{G}^3 \to \{0,1\}^\tau$. Security requirements are given in Section 2.

**Theorem 3** *The construction of KEM in Figure 6 is IND-CCA-secure under the DLIN assumption.*

*Proof.* Directly from Lemma 1 and Theorem 2. □

### 4.4 Instantiation under the DCR assumption

We use the HPS based on the decisional composite residuosity assumption (DCR) given in [20]. Let $p_1 = 2p_2 + 1$ and $q_1 = 2q_2 + 1$ be primes, where $p_2$ and $q_2$ are also primes. Let $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$. Let $\mathbb{G}$ be the subgroup of $Z_{N_1^2}^*$ with order $N_1 N_2$. Note that $\mathbb{G}$ is written as $\mathbb{G} = \mathbb{G}_{N_1} \cdot \mathbb{G}_{N_2}$ where $\mathbb{G}_{N_i}$ denotes a cyclic group of order $N_i$. Let $g$ be a generator of $\mathbb{G}$, so that $g_1 = g^{N_2}$ is a generator of $\mathbb{G}_{N_1}$ and $g_2 = g^{N_1}$ is a generator of $\mathbb{G}_{N_2}$.

In this HPS, $\mathcal{SK} = \{0, \ldots, \lfloor N_1^2/2 \rfloor\}^2$, $\mathcal{PK} = \mathbb{G}_{N_2}^2$, $\mathcal{K} = \mathbb{Z}_{N_1}$. Also $\mathcal{E} = \mathbb{G}$ and $\mathcal{V} = \{g_2^r \bmod N_1^2 : r \in \{0, \ldots, N_1/4\}\}$. The DCR assumption says that $\mathcal{E}$ and $\mathcal{V}$ are indistinguishable. To check $E \in \mathcal{E} \setminus \mathcal{V}$ in Figure 5, use trapdoor $N_2$.

The projection function is, using the same notation as in Figure 6,

$$\Lambda_{sk}(u) = u^{x+y\alpha} \bmod N_1 \Longleftrightarrow \Lambda_{sk}(u = g_2^r \bmod N_1^2) = (cd^\alpha)^r \bmod N_1.$$

**Lemma 2 (By [14, 20]).** *The above hash proof system is computationally universal 2 if* $\mathsf{TCR}$ *is target collision resistant.*

Our DLIN-based KEM appears in Figure 6, which uses symmetric building blocks $\mathsf{TCR} : \mathbb{Z}_{N_1^2} \to \mathbb{Z}_{\lfloor N_1^2/2 \rfloor}$, and $\mathsf{KDF} : \mathbb{Z}_{N_1} \to \{0,1\}^{2n}$, and $\mathsf{MAC} : \{0,1\}^n \times \mathbb{Z}_{N_1^2} \to \{0,1\}^\tau$ .

**Theorem 4** *The construction of KEM in Figure 6 is IND-CCA-secure under the DCR assumption.*

*Proof.* Directly from Lemma 2 and Theorem 2.

## References


1. International Organization for Standardization, Genève, Switzerland. ISO/IEC 18033-2:2006, Information technology — Security techniques — Encryption Algorithms — Part 2: Asymmetric Ciphers, 2006. Final Committee Draft available at `http://shoup.net/iso/`.
2. Cryptography Research and Evaluation Committees (CRYPTREC). Specifications of ciphers in the Candidate Recommended Ciphers List, March, 2013. `http://www.cryptrec.go.jp/english/method.html`.
3. M. Abdalla, M. Bellare, and P. Rogaway. DHIES: An encryption scheme based on the Diffie-Hellman problem, 2001. `http://cseweb.ucsd.edu/~mihir/papers/dhies.html`.
4. M. Abe, R. Gennaro, and K. Kurosawa. Tag-KEM/DEM: A new framework for hybrid encryption. *J. Cryptology*, 21(1):97–130, 2008.
5. American National Standards Institute. ANSI X9.44-2007: Key Establishment Using Integer Factorization Cryptography, 2007.
6. H. Anada and S. Arita. Identification schemes from key encapsulation mechanisms. *IEICE Transactions*, 95-A(7):1136–1155, 2012.
7. Apple Inc. iOS Security, October 2012. `https://www.apple.com/ipad/business/docs/iOS_Security_Oct12.pdf`.
8. J. Baek, D. Galindo, W. Susilo, and J. Zhou. Constructing strong KEM from weak KEM (or how to revive the KEM/DEM framework). In R. Ostrovsky, R. D. Prisco, and I. Visconti, editors, *SCN*, volume 5229 of *Lecture Notes in Computer Science*, pages 358–374. Springer, 2008.
9. D. J. Bernstein. Pippenger's exponentiation algorithm, 2002. `http://cr.yp.to/papers/pippenger.pdf`.
10. C. Boyd, Y. Cliff, J. M. G. Nieto, and K. G. Paterson. One-round key exchange in the standard model. *IJACT*, 1(3):181–199, 2009.
11. S. G. Choi, J. Herranz, D. Hofheinz, J. Y. Hwang, E. Kiltz, D. H. Lee, and M. Yung. The Kurosawa-Desmedt key encapsulation is not chosen-ciphertext secure. *Inf. Process. Lett.*, 109(16):897–901, 2009.
12. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.
13. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33:167–226, 2001.

14. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002.
15. Y. Desmedt, R. Gennaro, K. Kurosawa, and V. Shoup. A new and improved paradigm for hybrid encryption secure against chosen-ciphertext attack. *J. Cryptology*, 23(1):91–120, 2010.
16. A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 467–484. Springer, 2012.
17. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.
18. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptology*, 26(1):80–101, 2013.
19. J. Herranz, D. Hofheinz, and E. Kiltz. The Kurosawa-Desmedt key encapsulation is not chosen-ciphertext secure. *IACR Cryptology ePrint Archive*, 2006:207, 2006.
20. D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. Cryptology ePrint Archive, Report 2007/288, 2007. `http://eprint.iacr.org/`. Full version of a paper at CRYPTO 2007.
21. IEEE P1363a Committee. IEEE 1363a-2004: Standard Specifications For Public Key Cryptography – Amendment 1: Additional Techniques, 2004. `http://grouper.ieee.org/groups/1363/P1363a/`.
22. E. Kiltz. Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie-Hellman. In T. Okamoto and X. Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 282–297. Springer, 2007.
23. H. Krawczyk, K. G. Paterson, and H. Wee. On the security of the TLS protocol: A systematic analysis. In R. Canetti and J. A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 429–448. Springer, 2013.
24. K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2004.
25. National Institute of Standards and Technology. Recommended elliptic curves for federal government use, 1999. `http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf`.
26. T. Okamoto. Authenticated key exchange and key encapsulation in the standard model. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 474–484. Springer, 2007. Revised version available at `http://eprint.iacr.org/2007/473`.
27. PSEC-KEM website. `http://info.isl.ntt.co.jp/crypt/eng/psec/contents.html`.
28. The Standards for Efficient Cryptography Group. SEC 1: Elliptic Curve Cryptography, 2000. `http://www.secg.org/secg_docs.htm`.
29. K. Yoneyama. Compact authenticated key exchange from bounded CCA-secure KEM. In G. Paul and S. Vaudenay, editors, *INDOCRYPT*, volume 8250 of *Lecture Notes in Computer Science*, pages 161–178. Springer, 2013.

# A A variant of our DDH-based KEM

Here we describe a variant of the KEM in Figure 1. The encapsulation is the same, while key generation and decapsulation are different. In decapsulation, one needs one group membership check that $u_1 \in \mathbb{G}$, which in turn ensures that $u_2 \in \mathbb{G}$ and $v \in \mathbb{G}$.

**Fig. 7.** An optimization of the KEM in Figure 1.

| $\underline{\mathsf{KG}(1^\kappa):}$ | $\underline{\mathsf{Encap}(pk):}$ | $\underline{\mathsf{Decap}(sk, C):}$ |
|---|---|---|
| $g_1 \xleftarrow{\$} \mathbb{G}$ | $r \xleftarrow{\$} \mathbb{Z}_q$ | Parse $C = (u_1, u_2, t)$ |
| $(x, y, \omega) \xleftarrow{\$} \mathbb{Z}_q^3$ | $u_1 \leftarrow g_1^r, u_2 \leftarrow g_2^r$ | If $u_2 \neq u_1^\omega$: return $\perp$ |
| $g_2 \leftarrow g_1^\omega$ | $\alpha \leftarrow \mathsf{TCR}(u_1, u_2)$ | $\alpha \leftarrow \mathsf{TCR}(u_1, u_2)$ |
| $c \leftarrow g_1^x$ | $v \leftarrow c^r d^{r\alpha}$ | $v \leftarrow u_1^{x+\alpha y}$ |
| $d \leftarrow g_1^y$ | $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$ | $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$ |
| $pk \leftarrow (g_1, g_2, c, d)$ | $t \leftarrow \mathsf{MAC}_{k_a}(u_1, u_2)$ | If $t = \mathsf{MAC}_{k_a}(u_1, u_2)$ |
| $sk \leftarrow (x, y, \omega)$ | Return $C = (u_1, u_2, t)$ | return $k_s$ |
| Return $(pk, sk)$ | and $K = k_s$ | Else return $\perp$ |

# B Comparison with other KEMs in the random oracle model

In this section, we compare our DDH-based KEMs with other standardized schemes whose security were examined in the random oracle model. As will be clear below, the decapsulation cost in our KEMs is comparable, or even lesser, those in the KEMs.

**ECIES-KEM.** The scheme was originally developed by Abdalla, Bellare, and Rogaway [3]. Other names are DHES and as DHAES. Versions of the scheme are in ISO 18033-2 [1], IEEE 1363a [21], and SECG/SEC1 [28]. For comparison in Table 3, we use the version in [1].

**PSEC-KEM.** The scheme [27] was originally developed at Nippon Telegraph and Telephone corporation based on the work of Fujisaki and Okamoto [17] (refined in [18]). The KEM appears in ISO/IEC 18033 [1], and in the Candidate Recommended Ciphers List of CRYPTREC [2]. The one we use for comparison on Table 3 is described in [1].

**Comments on Table 3.** On decapsulation efficiency, ignoring group membership checks, then ECIES-KEM is the fastest with 1 (se), next comes ours in Figure 1 with 1 (me) counted as 1.2 (se), then PSEC-KEM with 2 (se).

Concerning security assumption, one has to make choices between the computational Diffie-Hellman assumption (CDH) in PSEC-KEM, gap CDH in ECIES-KEM (both in the random oracle model), or DDH in standard model in ours.

The seed length $|seed|$ must be set up so that $Q/2^{|seed|}$ is negligible where $Q$ is the number of decapsulation queries, so that roughly the encapsulation in PSEC-KEM is $|q|$ bits lesser than ours.

Our encapsulation is a bit less efficient than both ECIES-KEM and PSEC-KEM by 1 multi-exponentiation.

**Table 3.** Comparison between our KEMs, ECIES-KEM, PSEC-KEM. Abbreviations: rom = random oracle model, std = standard model, others are identical to Table 2.

| **Scheme** | **Assumption** and **Model** | **Encap length** | **[Encap]; [Decap] main costs of computation** | $[pk, sk]$ **size** |
|---|---|---|---|---|
| ECIES-KEM | gapCDH, rom | $\|q\|$ | [0 **me**, 2 **se**]; [0 **me**, 1 **se**, 1 **gmc**] | [1 **el**, 1 **el**] |
| PSEC-KEM | CDH, rom | $\|q\| + \|seed\|$ | [0 **me**, 2 **se**]; [0 **me**, 2 **se**, 1 **gmc**] | [1 **el**, 1 **el**] |
| Figure 1 | DDH, std | $2\|q\| + \|t\|$ | [1 **me**, 2 **se**]; [1 **me**, 0 **se**, 2 **gmc**] | [4 **el**, 4 **el**] |
| Figure 7 | DDH, std | $2\|q\| + \|t\|$ | [1 **me**, 2 **se**]; [0 **me**, 2 **se**, 1 **gmc**] | [4 **el**, 3 **el**] |

## C A speedup algorithm for multi-exponentiation

Algorithm 1 is what we use for computing multi-exponentiation in our implementation, which is a special case of the Straus's algorithm [9, Section 3]. Experimental results are depicted in Figure 8, in which "speedup" uses Algorithm 1, while "trivial" means computing $U^x$ and $V^y$ separately and then multiplying them together. For comparison purpose, timings for a single exponentiation are also drawn in "single-exp".

**Algorithm 1** Multi-Exp$(U, x, V, y)$ computes $Z = U^x V^y$ over group $\mathbb{G}$

**Require:** $U, V \in \mathbb{G}$ and positive integers $x, y \in \mathbb{Z}$
**Require:** Binary representations $x = x_n \cdots x_1$ and $y = y_n \cdots y_1$
1: $W \leftarrow UV$
2: $Z \leftarrow 1$
3: for $i$ from $n$ to 1 step $-1$
4: $\quad Z \leftarrow Z^2$
5: $\quad$ if $(x_i = 1$ and $y_i = 0)$, $Z \leftarrow Z \cdot U$
6: $\quad$ if $(x_i = 0$ and $y_i = 1)$, $Z \leftarrow Z \cdot V$
7: $\quad$ if $(x_i = 1$ and $y_i = 1)$, $Z \leftarrow Z \cdot W$
8: Return $Z$

In Figure 8, perhaps it it worth noting that exponentiations in group $\mathbb{G} = (\mathbb{Z}_p^*)^2 \subset \mathbb{Z}_p^*$ where $p = 2q + 1$ is a safe prime is relatively expensive. The reason is that $\mathbb{G}$'s order is $q$, which is as large as $p$, so that the exponents $x, y$ can be of the same magnitude of 1024 bits in length.

In contrast, when $p = \nu q + 1$ for $\nu > 2$ and $q$ is of 160 bits, exponents $x, y$ are of at most 160 bits in length, so that the computation becomes more efficient.

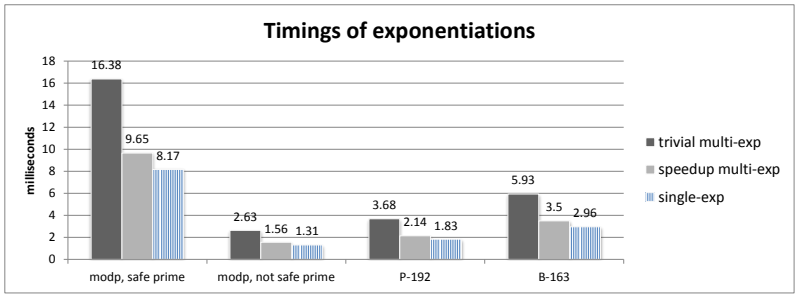



**Fig. 8.** Average timings of trivial and speedup computation of exponentiations, taken over 10000 executions, over various base groups. Experiment is done over a laptop (Intel 2.0GHz CPU, 8GB RAM) running Ubuntu 12.04 LTS. The C compiler is `g++` 4.6.3 using NTL 6.0.0 and GMP 5.1.1 libraries.