Kurosawa-Desmedt Key Encapsulation Mechanism, Revisited and More[⋆]

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

[1] Ibaraki University, Japan
kurosawa@mx.ibaraki.ac.jp
[2] NICT, Japan
phong@nict.go.jp

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 widely known as not IND-CCA-secure. In this paper, we present a direct proof of IND-CCA security thanks to a simple twist on the Kurosawa-Desmedt KEM. 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
- up to 60% in decapsulation speed,

which are verified by both theoretical comparison and experimental results. The efficiency of decapsulation can be even

- about 40% better than the decapsulation of the PSEC-KEM in ISO/IEC 18033-2
- only slightly worse than the decapsulation of the ECIES-KEM in ISO/IEC 18033-2

which is of independent interest since the security of both PSEC-KEM and ECIES-KEM are argued using the controversial random oracle heuristic in contrast to ours.

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.

We finally show that, with additional simple yet innovative twists, the KEMs can be proved resilient to certain amount of leakage on the secret key. Specifically with the DDH-based scheme, a fraction of $1/4 - o(1)$ of the secret key can be leaked, and when conditioned on a fixed leakage rate, we obtain the most efficient leakage-resilient KEMs regarding computation and storage.

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

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.

Key encapsulation mechanism is perhaps a modern way to position public-key encryption thanks to its flexibility. First, since the symmetric key K_s is returned as a result of encapsulation, there is no need for ad-hoc padding to map bit strings into algebraic message space as in traditional

[⋆] Full version of a paper accepted to the 7th International Conference on Cryptology in Africa (AFRICACRYPT 2014), with major extensions described in Section 1.5.

public-key encryption (PKE). Such padding is hard-to-be-done-right, and may lead to devastating attack [9]. Second, after the key K_s is conveyed, encrypted data under that symmetric key can come in stream, and there is completely unnecessary to buffer the whole ciphertext before decryption.

KEM implies 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 [26].

In 2004, Kurosawa and Desmedt [27], 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, 21]. In 2007, by creatively switching elements in the Kurosawa-Desmedt KEM, Kiltz [24] 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 [22] showed the Kurosawa-Desmedt KEM only meets weakened notions of CCA security.

While weakened IND-CCA security as defined in [4,22] 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 (↓) Assumption (→)	GHDH	DDH
Weakened IND-CCA	–	[4], [22] (KD KEM)
IND-CCA	[24] (dual KD KEM)	*This paper (with direct proof)*

1.2 Our contributions

We provide a direct proof of IND-CCA security for a modified version of Kurosawa-Desmedt KEM. We then implement our KEM to highlight its efficiency over previous constructions. We generalize the mechanism to other cryptographic assumptions via hash proof systems. Finally, we show some simple twists turning our schemes leakage-resilient while maintaining the same efficiency. Details are given below.

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 impacts. 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, and
- 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 theoretical estimations are checked by experimental results in Section 3.2. These improvements are significant, as frequently there are large amounts of asymmetric encryption and decryption works, e.g., in SSL/TLS servers. Indeed, due to these practicalities, we think that our variants of the Kurosawa-Desmedt KEM deserve direct and dedicated proofs of security.

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.

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. See Section 4.

Leakage-resilient extensions. Above KEMs have their leakage-resilient variants, described in Section 6. In particular, keeping the same speed improvements as above, we present a DDH-based KEM secure even if 18.85% amount of secret key's information is leaked when the KEM operates over NIST's elliptic curve P-521. The rate goes up to 21.87% when considering group $\mathbb{Z}_p^*$ where p is of 1024 bit length. While these cannot reach beyond rate 1/4 as in [32], conditioned on the same (or approximate) leakage rates, our proposal outperforms the scheme in [32]. See Section 6.4.

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, 17, 34]. 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. [17] 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 [27] 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, 21], 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.

Using essentially the same idea with this work, Baek et al. [7] showed that constrained IND-CCA (CCCA) security [22] can be converted into standard IND-CCA security. 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 addition, we remark that Hanaoka and Kurosawa [20] showed a MAC-free conversion from CCCA to CCA security. While going through CCCA security as in [20] has benefits such as yielding new constructions with ciphertext size improvements, we think that a straight proof of security without any redundant loss factor is more intuitive and easier to follow.

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 being 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.

1.5 Additional content over the conference version

A short version of this paper is in [29]. Newly added materials are mainly in Sections 5 and 6 showing how to simplify the KEMs in previous sections, and make them leakage-resilient.

2 Preliminaries

2.1 Key encapsulation mechanisms

KEM. A KEM consists of key generation KG, encapsulation Encap, and decapsulation Decap algorithms. $\mathsf{KG}(1^\kappa)$ with security parameter κ 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 $\mathcal{A}$ invokes a challenge oracle who computes $(C^*, K^*) \leftarrow \mathsf{Encap}(pk)$, then takes K_* randomly satisfying $|K^*| = |K_*|$, and chooses $b \overset{\$}{\leftarrow} \{0, 1\}$. 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

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

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

Leakage-resilient IND-CCA security of KEM. The definition is the same as above, except that in the find stage, the adversary $\mathcal{A}$ can additionally submit any circuit f to receive the leakage $f(sk)$ on the secret key. The leakage queries can be adaptive, and yet the total leakage length in bits must be bounded by a number L. The leakage rate is defined as the fraction $L/|sk|$ where $|sk|$ is the length of the secret key.

It is worth noting that leakage queries are not allowed after the challenge phase. The reason is, suppose such query is permitted, the adversary can take $f =$ last bit of $\mathsf{Decap}_{sk}(\cdot)$ to receive $f(C^*)$ and then compare with the last bits of $K(b)$. Therefore, even one-bit leakage resilience is impossible in this case.

2.2 Symmetric building blocks

Taking an element a randomly from a set A is notationally expressed by $a \overset{\$}{\leftarrow} A$. Let κ 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 $x^* \overset{\$}{\leftarrow} \mathcal{E}(\kappa)$, 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

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

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

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

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

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 $k \overset{\$}{\leftarrow} \{0,1\}^{n(\kappa)}$, 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

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

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

Note that the definition treats MAC as a function where $\mathcal{E}(\kappa)$ contains both messages and randomness (if any), the security notion already captures *strong* unforgeability against chosen-message attacks.

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

$\mathsf{KG}(1^\kappa)$:	$\mathsf{Encap}(pk)$:	$\mathsf{Decap}(sk, C)$:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$	$r \xleftarrow{\$} \mathbb{Z}_q$	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 $\perp$

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 κ.

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

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

is negligible on parameter κ.

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 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 TCR, KDF, and MAC are in Section 2.2.

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 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 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, 21] 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, 21], 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 MAC is assumed secure.

Our use of MAC is different from the counterpart in the hybrid PKE [15] in its input. In [15], MAC is used to authenticate a symmetrically encrypted plaintext e. Namely, using our notations, in [15], $e \leftarrow \mathsf{SymmetricEncryption}_{k_s}(\texttt{plaintext})$ and then $t \leftarrow \mathsf{MAC}_{k_a}(e)$. In contrast, in Figure 1,

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. The ($\sim$ **gmc**) indicates that group membership checks may be very efficient compared to exponentiation.

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**]

we take "early" 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 with ACE-KEM

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 [30] 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.

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 [8], therefore can be counted as approximately 1.2 single exponentiation, which also is supported by experimental results in Section A.

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% (computed by $(1 - 1.2/3) \cdot 100\%$) in decapsulation speed. Examples of the groups include NIST elliptic curves [30] 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 Figure 1 require 3.2 (se), while that of ACE-KEM is 4 (se), yielding at least $(1 - 3.2/4) \cdot 100\% = 20\%$ improvement.

3.3 Comparison with random oracle model's KEMs

Table 3 compares our DDH-based KEMs with other standardized schemes whose security were examined in the random oracle model. Remarkably, the decapsulation cost in our KEMs is comparable, or even lesser, those in the KEMs.

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**	**Encap**	**[Encap]**; **[Decap]**	$[pk, sk]$ **size**
	and **Model**	**length**	**main costs of computation**	
ECIES-KEM	gapCDH, rom	$\lvert q\rvert$	$[0\ \mathbf{me} + 2\ \mathbf{se}]$; $[0\ \mathbf{me} + 1\ \mathbf{se} + (1\ \mathbf{gmc})]$	$[1\ \mathbf{el}, 1\ \mathbf{el}]$
PSEC-KEM	CDH, rom	$\lvert q\rvert + \lvert seed\rvert$	$[0\ \mathbf{me} + 2\ \mathbf{se}]$; $[0\ \mathbf{me} + 2\ \mathbf{se} + (1\ \mathbf{gmc})]$	$[1\ \mathbf{el}, 1\ \mathbf{el}]$
Figure 1	DDH, std	$2\lvert q\rvert + \lvert t\rvert$	$[1\ \mathbf{me} + 2\ \mathbf{se}]$; $[1\ \mathbf{me} + 0\ \mathbf{se} + (2\ \mathbf{gmc})]$	$[4\ \mathbf{el}, 4\ \mathbf{el}]$

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 [23], and SECG/SEC1 [33]. For comparison in Table 3, we use the version in [1].

PSEC-KEM. The scheme [31] was originally developed at Nippon Telegraph and Telephone corporation based on the work of Fujisaki and Okamoto [18] (refined in [19]). 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). Therefore, the decapsulation of our proposal is $(1-1.2/2)\cdot 100\% = 40\%$ faster than that of PSEC-KEM, while a little slower than ECIES-KEM.

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.

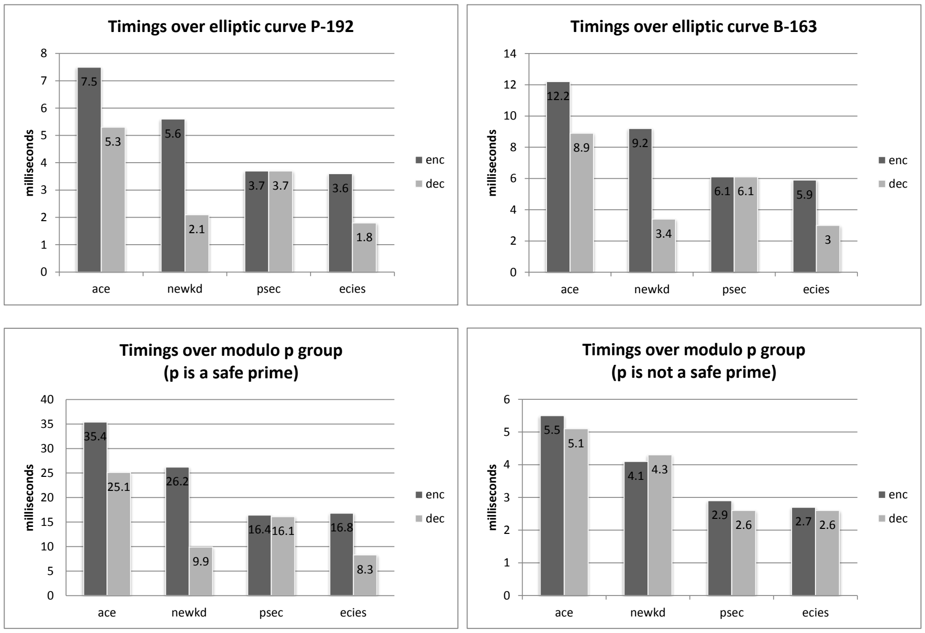
Our encapsulation is a bit less efficient than both ECIES-KEM and PSEC-KEM by 1 multi-exponentiation.

3.4 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 (see Section A for details), and

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.



- using Legendre symbol for group membership check in $\mathbb{G} \subset \mathbb{Z}_p^*$ where p is a “safe” prime, namely $p = 2q + 1$ for a prime q (Sophie Germain prime).

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.

3.5 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. A quantitative reduction is given in eq.(11) in the following proof.*

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, $\Pr[X_i] = \Pr[b' = b \text{ in } \mathbf{Game}_i]$.

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

The challenge is $(C^*, K(b))$ where $C^* = (u_1^*, u_2^*, t^*)$. We denote by r^*, α^*, 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.

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

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

since the first change is notational, and the second one is based on the security of 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.

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

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

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.

Game$_3$: This game makes use of $\omega \in \mathbb{Z}_q^*$ satisfying $g_2 = g_1^{\omega}$. With ω, 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) 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 Game$_i$. Let Q be the bound on the total number of decapsulation queries $\mathcal{A}$ makes, we have

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

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

Fig. 3. Oracles in **Game**$_3$ for the proof of Theorem 1.

Initialization of the game

```
I1: $\omega \xleftarrow{\$} \mathbb{Z}_q^*$, $g_2 \leftarrow g_1^{\omega}$
I2: $(x_1, x_2, y_1, y_2) \xleftarrow{\$} \mathbb{Z}_q^4$
    $c \leftarrow g_1^{x_1} g_2^{x_2}$, $d \leftarrow g_1^{y_1} g_2^{y_2}$
I3: $r_1^* \xleftarrow{\$} \mathbb{Z}_q$, $u_1^* \leftarrow g_1^{r_1^*}$
    $r_2^* \xleftarrow{\$} \mathbb{Z}_q \setminus \{r_1^*\}, u_2^* \leftarrow g_2^{r_2^*}$
I4: $\alpha^* \leftarrow \mathsf{TCR}(u_1^*, u_2^*)$
    $v^* \leftarrow (u_1^*)^{x_1+\alpha^* y_1} (u_2^*)^{x_2+\alpha^* y_2}$
I5: $(k_s^*, k_a^*) \leftarrow \mathsf{KDF}(v^*)$
```

Decapsulation of adversarial query $C = (u_1, u_2, t)$

```
1: $\alpha = \mathsf{TCR}(u_1, u_2)$
2: if $(u_1, u_2) \neq (u_1^*, u_2^*)$ and $\alpha = \alpha^*$ then
3:     return $\perp$
4: end if
5: if $(u_1, u_2) = (u_1^*, u_2^*)$ then
6:     if $t \neq \mathsf{MAC}_{k_a^*}(u_1^*, u_2^*)$ then return $\perp$
7:     else return $k_s^*$
8: else if $(u_1, u_2) \notin \mathcal{V}$ then
9:     $\alpha \leftarrow \mathsf{TCR}(u_1, u_2)$
10:    $v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$
11:    $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$
12:    if $t \neq \mathsf{MAC}_{k_a}(u_1, u_2)$ then return $\perp$
13:    else return $\perp$ {Rejection rule in Game$_3$}
14: else
15:    $\alpha \leftarrow \mathsf{TCR}(u_1, u_2)$
16:    $v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$
17:    $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$
18:    if $t \neq \mathsf{MAC}_{k_a}(u_1, u_2)$ then return $\perp$
19:    else return $k_s$
20: end if
```

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

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 KDF such that

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

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 ω 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 KDF and $\mathcal{A}_5''$ against 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),

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ind-cca}}_{\mathcal{A}}(\kappa) \leq\ & \mathbf{Adv}^{\mathsf{TCR}}_{\mathcal{A}_1}(\kappa) + \mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{A}_2}(\kappa) + Q\left(\mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}_5''}(\kappa) + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}_5'}(\kappa)\right)\\ & + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}_5}(\kappa) + Q\left(\frac{1}{q^2} + \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}_6}(\kappa)\right)\end{aligned} \tag{11}$$

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

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

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

A function $\Lambda_{sk} : \mathcal{E} \times \mathsf{Seed} \to \mathcal{K}$ is *projective* if there exists a projection $\mu : \mathcal{SK} \to \mathcal{PK}$ such that $pk = \mu(sk)$ defines Λ_{sk} restricted on subset $\mathcal{V} \times \mathsf{Seed}$ of $\mathcal{E} \times \mathsf{Seed}$. Namely, for every $E \in \mathcal{V}$ and $sd \in \mathsf{Seed}$, the value $K = \Lambda_{sk}(E, sd)$ is uniquely determined by $pk = \mu(sk)$ and E, sd. Note that

- Before Section 6, the set Seed is empty. As an example, in our scheme of Section 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 $\alpha = \mathsf{TCR}(E)$. This is the original definition of hash proof system in [14].
- In Section 6, Seed is a set of bit strings of fixed length. As an example, in our scheme of Section 6.2, $\Lambda_{sk}\big(E = (u_1, u_2), sd\big) = u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$ where $\alpha = \mathsf{TCR}(E, sd)$ for $sd \in \mathsf{Seed}$. This is an extension of hash proof system, first explicitly given in [28].

A projective function Λ_{sk} is called computationally universal-2 [22] if for all $E, E' \notin \mathcal{V}$ with $(E, sd) \neq (E', sd')$, the tuples

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

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 $\mathbf{Exp}_{\mathcal{A}}^{\mathrm{cu2}}(\kappa)$, in which the oracle $\mathsf{Eval}_{sk}(F, s)$ returns $\Lambda_{sk}(F, s)$ if $F \in \mathcal{V}$ and $s \in \mathsf{Seed}$; and $\perp$ otherwise. Computational universality requires that

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

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

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

```
Run $\mathsf{Param}(1^\kappa)$ to generate
$\big(group, \mathcal{SK}, \mathcal{PK}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \Lambda_{(\cdot)}(\cdot), \mu, \mathsf{Seed}\big)$
$sk \xleftarrow{\$} \mathcal{SK}, pk \leftarrow \mu(sk), E' \xleftarrow{\$} \mathcal{E} \setminus \mathcal{V}$
$sd' \xleftarrow{\$} \mathsf{Seed}, K' \leftarrow \Lambda_{sk}(E', sd')$
$(E, sd, \mathtt{st}) \leftarrow \mathcal{A}_{\mathrm{find}}^{\mathsf{Eval}_{sk}(\cdot,\cdot)}(pk, E', sd', K')$
where $(E, sd) \neq (E', sd')$ and $E \in \mathcal{E} \setminus \mathcal{V}$
$b \xleftarrow{\$} \{0,1\}, K(0) \leftarrow \Lambda_{sk}(E, sd), K(1) \xleftarrow{\$} \mathcal{SK}$
$b' \leftarrow \mathcal{A}_{\mathrm{guess}}(\mathtt{st}, K(b))$
If $b' = b$ then return 1 else return 0
```

Hash proof system. A hash proof system $\mathcal{HPS}$ consists of 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 KDF and MAC are in Section 2.2.

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

$\mathsf{KG}(1^\kappa)$:	$\mathsf{Encap}(pk)$:	$\mathsf{Decap}(sk, C)$:
Run 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 $\perp$

Theorem 2 *The generic construction of KEM in Figure 4 is IND-CCA-secure. A quantitative reduction is given in eq.(21) in the following proof.*

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 $\mathsf{Priv}(sk, E^*) = \mathsf{Pub}(pk, E^*, r^*) = \varLambda_{sk}(E^*)$ so that $\Pr[X_0] = \Pr[X_1]$.

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

$$|\Pr[X_1] - \Pr[X_2]| \leq \mathbf{Adv}^{\mathrm{sm}}_{\mathcal{A}_2}(\kappa) \tag{12}$$

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

$$|\Pr[X_2] - \Pr[X_3]| \leq Q \Pr[F_3] \tag{13}$$

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.

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

$$|\Pr[X_3] - \Pr[X_4]| \leq \mathbf{Adv}^{\mathrm{cu2}}_{\mathcal{A}_4}(\kappa) \tag{14}$$

$$|\Pr[F_3] - \Pr[F_4]| \leq \mathbf{Adv}^{\mathrm{cu2}}_{\mathcal{A}'_4}(\kappa) \tag{15}$$

where event F_4 happens whenever line 8 of decapsulation is reached in this game. The reasons are that $v = \varLambda_{sk}(E)$ is computationally random conditioned on $pk, v^* = \varLambda_{sk}(E^*)$; and that $v^* = \varLambda_{sk}(E^*)$ 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 KDF such that

$$|\Pr[X_4] - \Pr[X_5]| \leq \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}_5}(\kappa). \tag{16}$$

Fig. 5. Oracles in $\mathbf{Game}_3$ for the proof of Theorem 2.

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

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

$\mathbf{Game}_6$: 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

$$|\Pr[X_5] - \Pr[X_6]| \leq Q\left(\frac{1}{|\mathcal{E}|} + \mathbf{Adv}_{\mathcal{A}_6}^{\mathsf{MAC}}(\kappa)\right) \text{ and } \Pr[X_6] = \frac{1}{2} \tag{17}$$

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).

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

$$|\Pr[F_4] - \Pr[F_{5'}]| \leq \mathbf{Adv}_{\mathcal{A}_5'}^{\mathsf{KDF}}(\kappa). \tag{18}$$

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,

$$\Pr[F_{5'}] \leq \mathbf{Adv}_{\mathcal{A}_5''}^{\mathsf{MAC}}(\kappa) \tag{19}$$

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 KDF and $\mathcal{A}_5''$ against MAC are similar to those in $\mathbf{Game}_5$ and $\mathbf{Game}_6$.

By (15), (18), and (19),

$$\Pr[F_3] \leq \mathbf{Adv}_{\mathcal{A}_4'}^{\mathrm{cu2}}(\kappa) + \mathbf{Adv}_{\mathcal{A}_5'}^{\mathsf{KDF}}(\kappa) + \mathbf{Adv}_{\mathcal{A}_5''}^{\mathsf{MAC}}(\kappa) \tag{20}$$

Summing up (12), (13), (14), (16), (17), and (20),

$$\mathbf{Adv}^{\mathrm{ind-cca}}_{\mathcal{A}}(\kappa) \leq \mathbf{Adv}^{\mathrm{sm}}_{\mathcal{A}_2}(\kappa) + Q\left(\mathbf{Adv}^{\mathrm{cu2}}_{\mathcal{A}'_4}(\kappa) + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}'_5}(\kappa) + \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}''_5}(\kappa)\right)$$
$$+\mathbf{Adv}^{\mathrm{cu2}}_{\mathcal{A}_4}(\kappa) + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{A}_5}(\kappa) + Q\left(\frac{1}{|\mathcal{E}|} + \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{A}_6}(\kappa)\right) \qquad (21)$$

ending the proof. □

4.3 Instantiation under the DLIN assumption

We use the HPS based on the decisional linear assumption (DLIN) given by [22]. 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

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

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**).

$\mathsf{KG}(1^\kappa)$:	$\mathsf{Encap}(pk)$:	$\mathsf{Decap}(sk, C)$:
$g_1, g_2, h \xleftarrow{\$} \mathbb{G}$	$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q$	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 $K = k_s$	

$\mathsf{KG}(1^\kappa)$:	$\mathsf{Encap}(pk)$:	$\mathsf{Decap}(sk, C)$:
$g \xleftarrow{\$} \mathbb{G}, g_2 \leftarrow g^{N_1}$	$r \xleftarrow{\$} \{0, \dots, N_1/4\}$	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 [22]). *The above hash proof system is computationally universal-2 if* 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 2.2.

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 [22]. 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, $\mathcal{SK} = \{0, \ldots, \lfloor N_1^2/2 \rfloor\}^2$, $\mathcal{PK} = \mathbb{G}^2_{N_2}$, $\mathcal{K} = \mathbb{Z}_{N_1}$. 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, 22]). *The above hash proof system is computationally universal 2 if* 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.

5 How to remove the MAC

As a MAC is a symmetric primitive, its costs including size and computation are already modest compared to exponentiations, so why we bother removing it from the proposed schemes? The reasons are as follows: by removing MAC,

- We rely on less computational assumptions for security.
- While not much, we still gain some efficiency regarding size and computation.
- The MAC-free schemes can be used as a starting point for leakage-resilient variants.

As also discussed in Section 1.4, the MAC-free conversion from CCCA to CCA security has appeared in [20] by a different method from ours. The work [20] did not move further to leakage resilient variants of the MAC-free schemes.

Fig. 7. Our IND-CCA-secure KEM under the DDH assumption, without MAC.

$\mathsf{KG}(1^\kappa)$:	$\mathsf{Encap}(pk)$:	$\mathsf{Decap}(sk, C)$:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$	$r \xleftarrow{\$} \mathbb{Z}_q$	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 $\boxed{t = k_a}$
$sk \leftarrow (x_1, x_2, y_1, y_2)$	$\boxed{t \leftarrow k_a}$	return k_s
Return (pk, sk)	Return $C = (u_1, u_2, t)$ and $K = k_s$	Else return $\perp$

5.1 DDH-based KEM without MAC

The DDH-based KEM without MAC is given in Figure 7. The difference with the scheme in Figure 1 is given in the boxes.

Removing MAC. A careful examination on the proof of Theorem 1 crystallizes the idea, and let us paraphrase it:

> referring to Figure 7, *instead of* $\mathsf{MAC}_{k_a}(u_1, u_2)$, *just returning* k_a *for authentication!*

In other words, the string $t = k_a$ servers as an authenticator for each (u_1, u_2). The replacement works well because the string k_a is computationally random and hence unique for each encapsulation (u_1, u_2), as shown in the proof of Theorem 1 (**Game**$_5$ and **Game**$_{5'}$, specifically). This is an additional merit of the direct proof.

We have the following theorem, which *directly* proves the security of the KEM. The proof is almost the same as that for Theorem 1, with necessary modifications related to the MAC part.

Theorem 5 *The KEM in Figure 7 is IND-CCA-secure under the DDH assumption. A quantitative reduction is given in eq.(22) in the following proof.*

Proof. The proof goes along the lines with that of Theorem 1, so we only specify the differences.

- Tags $\mathsf{MAC}_{k_a}(\cdot)$ in lines 6, 12, 18 of Figure 3 are plainly replaced by k_a in the decapsulation oracle.
- In **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^*, k_a^*) was already announced, querying (u_1^*, u_2^*, t) with $t = k_a^*$ is forbidden. We have

$$\left|\Pr[X_5] - \Pr[X_6]\right| \leq \frac{Q}{q^2} \text{ and } \Pr[X_6] = \frac{1}{2}.$$

- In **Game**$_{5'}$

$$\Pr[F_{5'}] \leq 2^{-|k_a|}$$

where $|k_a|$ is the length of k_a in bits, and $F_{5'}$ is the event that a decryption query is rejected at line 13 of the decapsulation oracle in this game.

Fig. 8. Our generic KEM from hash proof system $(\mathsf{Param}, \mathsf{Pub}, \mathsf{Priv})$ without MAC.

$\mathsf{KG}(1^\kappa)$:	$\mathsf{Encap}(pk)$:	$\mathsf{Decap}(sk, C)$:
Run 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 $\boxed{t = k_a}$
$pk \leftarrow \mu(sk)$	$\boxed{t \leftarrow k_a}$	return k_s
Return (pk, sk)	Return $C = (E, t)$ and $K = k_s$	Else return $\perp$

- Having the above changes, the final quantitative reduction becomes

$$\mathbf{Adv}_{\mathcal{A}}^{\mathrm{ind-cca}}(\kappa) \leq \mathbf{Adv}_{\mathcal{A}_1}^{\mathsf{TCR}}(\kappa) + \mathbf{Adv}_{\mathcal{A}_2}^{\mathrm{ddh}}(\kappa) + Q\left(2^{-|k_a|} + \mathbf{Adv}_{\mathcal{A}_5'}^{\mathsf{KDF}}(\kappa)\right)$$
$$+\mathbf{Adv}_{\mathcal{A}_5}^{\mathsf{KDF}}(\kappa) + \frac{Q}{q^2}, \qquad (22)$$

ending the proof. □

5.2 Generalization: HPS-based KEM without MAC

The extension basing on hash proof system is given in Figure 8, in which the boxes show the difference with the MAC-based version previously given in Figure 4. Again, one-time secure $\mathsf{MAC}_{k_a}(\cdot)$ is simply replaced by k_a, which is pseudo-random and tied to each E in the encapsulation $(E, t = k_a)$. We have the following theorem.

Theorem 6 *The generic construction of KEM in Figure 8 is IND-CCA-secure. A quantitative reduction is given in eq.(25) in the following proof.*

Proof. The proof is similar to that of Theorem 2, so let us only show the differences.

- In $\mathbf{Game}_3$, $\mathsf{MAC}_{k_a}(\cdot)$ in Figure 5 is replaced by k_a. Particularly, lines 2, 7, 12 are changed.
- In $\mathbf{Game}_6$, (17) is turned to

$$|\Pr[X_5] - \Pr[X_6]| \leq \frac{Q}{|\mathcal{E}|} \text{ and } \Pr[X_6] = \frac{1}{2} \qquad (23)$$

in which $\frac{1}{|\mathcal{E}|}$ is the probability $E = E^*$ before the challenge phase and Q is the number of decryption queries.
- In $\mathbf{Game}_{5'}$, (19) is changed to

$$\Pr[F_{5'}] \leq 2^{-|k_a|} \qquad (24)$$

where $|k_a|$ is the length of k_a, and $F_{5'}$ is the event that line 8 in the modified Figure 5 is executed, namely the adversary can come up with a tag satisfying $t = k_a$ where $E \notin \mathcal{V}$.
- The final reduction becomes

$$\mathbf{Adv}_{\mathcal{A}}^{\mathrm{ind-cca}}(\kappa) \leq \mathbf{Adv}_{\mathcal{A}_2}^{\mathrm{sm}}(\kappa) + Q\left(\mathbf{Adv}_{\mathcal{A}_4'}^{\mathrm{cu2}}(\kappa) + \mathbf{Adv}_{\mathcal{A}_5'}^{\mathsf{KDF}}(\kappa) + 2^{-|k_a|}\right)$$
$$+\mathbf{Adv}_{\mathcal{A}_4}^{\mathrm{cu2}}(\kappa) + \mathbf{Adv}_{\mathcal{A}_5}^{\mathsf{KDF}}(\kappa) + \frac{Q}{|\mathcal{E}|} \qquad (25)$$

ending the proof. □

6 Leakage-resilient extensions

The simplicity in authentication of schemes in Section 5 gives raise to leakage resilience of secret keys. In this section, we provide leakage-resilient variants of the schemes in Section 5, basing on following properties.

- Exactly, *unpredictability* is the requirement on the authenticator k_a in the proofs of Theorems 5 and 2. As k_a is psuedo-random as seen in Section 5, it becomes unpredictable in the presense of some leakage.
- By leakage, the symmetric key k_s becomes unpredictable as well, but we can fix that by using a cryptographic extractor converting unpredictable to uniformly random sources.
- In the proofs in Section 5, the simulator owns the secret key, which makes the simulation of leakage queries possible.

Note that the first property does not hold for the schemes in Sections 3 and 4, as the MAC's key must be random to claim its security.

6.1 Additional preliminaries: entropy and extractors

The statistical distance of random variables X, Y over a finite domain Ω is

$$\mathbf{SD}(X;Y) = \frac{1}{2}\sum_{a\in\Omega}\big|\Pr[X=a]-\Pr[Y=a]\big|.$$

The min-entropy of X is $\mathbf{H}_\infty(X) = -\log_2(\max_x \Pr[X=x])$. Thus

$$\Pr[X=x] \le \max_x \Pr[X=x] = 2^{-\mathbf{H}_\infty(X)}.$$

The average min-entropy of X conditioned on Y is

$$\tilde{\mathbf{H}}_\infty(X|Y) = -\log_2\left(E_{y\leftarrow Y}\left[2^{-\mathbf{H}_\infty(X|Y=y)}\right]\right),$$

as defined in [16], which also proved the following result.

Lemma 3 (Lemma 2.2 in [16]). *If Y has 2^λ possible values and Z is any random variable, then*

$$\tilde{\mathbf{H}}_\infty(X|Y,Z) \ge \tilde{\mathbf{H}}_\infty(X,Y|Z) - \lambda \ge \tilde{\mathbf{H}}_\infty(X|Z) - \lambda \ge \mathbf{H}_\infty(X,Z) - \lambda.$$

When applying the lemma in our context, Y stands for the leakage on secret key X, while Z is another information on X such as given by the public key. The lemma then says that, given a leakage amount of λ bits, the secret key's entropy is decreased by λ. Hereafter, when referring to entropy, we mean average min-entropy unless otherwise stated.

A function $\mathsf{Ext} : \{0,1\}^{n_s} \times \mathsf{Seed} \to \{0,1\}^l$ is called a $(n_{\mathrm{etp}}, \epsilon_{\mathsf{Ext}})$-randomness extractor if for all pairs of random variables (X, I) such that X is an n_s-bit string satisfying $\tilde{\mathbf{H}}_\infty(X|I) \ge n_{\mathrm{etp}}$,

$$\mathbf{SD}\Big((\mathsf{Ext}(X, sd), sd, I); (\mathsf{rand}, sd, I)\Big) \le \epsilon_{\mathsf{Ext}},$$

where $sd \overset{\$}{\leftarrow} \mathsf{Seed}$ and $\mathsf{rand} \overset{\$}{\leftarrow} \{0,1\}^\ell$. In other words, $\mathsf{Ext}(X, sd)$ is nearly random given sd and I (when ϵ_{Ext} is small enough). Randomness extractors can be efficiently realized via pairwise independent hash functions.

6.2 Leakage-resilient DDH-based scheme

The scheme is given in Figure 9, in which Seed is a set of seeds used in an extractor $\mathsf{Ext} : \{0,1\}^{n_s} \times \mathsf{Seed} \to \{0,1\}^{\ell}$ where n_s and ℓ are integers.

Fig. 9. Our leakage-resilient IND-CCA-secure KEM under the DDH assumption.

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

Theorem 7 *The KEM in Figure 9 is IND-lrCCA-secure under the DDH assumption with leakage rate $\frac{1}{4} - o(1)$. A quantitative reduction is given in eq.(27) in the following proof.*

Proof. The proof differs with that of Theorem 5 at the following points:

- In all games, every leakage query with an arbitrary function f is answered by $f(sk)$ in which $sk = (x_1, x_2, y_1, y_2)$ is the secret key held by the simulator.
- We assume that $\mathsf{KDF}(v)$ has the same entropy as v for all $v \in \mathbb{G}$, which can be fulfilled if KDF is injective as in that case $\Pr[\mathsf{KDF}(v) = \mathsf{KDF}(w)] = \Pr[v = w]$ $\forall v, w \in G$. Concrete descriptions of KDF are in Section 6.4.
- In $\mathbf{Game}_{5'}$, with adversarial t and simulator's generated k_a,

  $$\Pr[F_{5'}] = \Pr[t = k_a] \leq 2^{-\tilde{\mathbf{H}}_\infty(k_a|\text{leakage so far})} \leq 2^{-(|k_a|-L)} \tag{26}$$

  where $|k_a|$ is the length of k_a in bits, and $F_{5'}$ is the event that a decryption query is rejected at line 13 of the decapsulation oracle in this game (see Figure 10). The last inequality comes by applying Lemma 3

  $$\tilde{\mathbf{H}}_\infty(k_a|\text{leakage so far}) \geq \tilde{\mathbf{H}}_\infty(k_a) - \tilde{\mathbf{H}}_\infty(\text{leakage so far}) \geq |k_a| - L$$

  as k_a is random of $|k_a|$ bit length and L is the bound on leakage in bits.
- Having the above changes, the quantitative reduction becomes

  $$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\text{ind}-\text{cca}}(\kappa) \leq \mathbf{Adv}_{\mathcal{A}_1}^{\mathsf{TCR}}(\kappa) + \mathbf{Adv}_{\mathcal{A}_2}^{\text{ddh}}(\kappa) + Q\left(2^{-|k_a|+L} + \mathbf{Adv}_{\mathcal{A}_5'}^{\mathsf{KDF}}(\kappa)\right) \\ + \mathbf{Adv}_{\mathcal{A}_5}^{\mathsf{KDF}}(\kappa) + \frac{Q}{q^2}.\end{aligned} \tag{27}$$

We now examine the leakage rate. Seeing (27), we set $-|k_a| + L = -128$ or equivalently $|k_a| - 128 = L$. Since $|k_a| \approx \log_2 q - |k_s|$ where q is the order of the DDH group, we have

$$\log_2 q - |k_s| - 128 \approx L.$$

Fig. 10. Oracles in $\mathbf{Game}_3$ for the proof of Theorem 7.

Initialization of the game

```
I1: $\omega \xleftarrow{\$} \mathbb{Z}_q^*$, $g_2 \leftarrow g_1^{\omega}$
I2: $(x_1, x_2, y_1, y_2) \xleftarrow{\$} \mathbb{Z}_q^4$
    $c \leftarrow g_1^{x_1} g_2^{x_2}$, $d \leftarrow g_1^{y_1} g_2^{y_2}$
I3: $r_1^* \xleftarrow{\$} \mathbb{Z}_q$, $u_1^* \leftarrow g_1^{r_1^*}$
    $r_2^* \xleftarrow{\$} \mathbb{Z}_q \setminus \{r_1^*\}, u_2^* \leftarrow g_2^{r_2^*}$
I4: $\alpha^* \leftarrow \mathsf{TCR}(u_1^*, u_2^*)$
    $v^* \leftarrow (u_1^*)^{x_1+\alpha^* y_1}(u_2^*)^{x_2+\alpha^* y_2}$
I5: $(k_s^*, k_a^*) \leftarrow \mathsf{KDF}(v^*)$
I6: $sd^* \leftarrow \mathsf{Seed}$
I7: $K^* \leftarrow \mathsf{Ext}(k_s^*, sd^*)$
```

Decapsulation of query $C = (u_1, u_2, t, sd)$

```
$\alpha = \mathsf{TCR}(u_1, u_2, sd)$
if $(u_1, u_2, sd) \neq (u_1^*, u_2^*, sd^*)$ and $\alpha = \alpha^*$ then
    return $\perp$
end if
if $(u_1, u_2, sd) = (u_1^*, u_2^*, sd^*)$ then
    if $t \neq k_a^*$ then return $\perp$
    else return $K^*$
else if $(u_1, u_2) \notin \mathcal{V}$ then
    $\alpha \leftarrow \mathsf{TCR}(u_1, u_2, sd)$
    $v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$
    $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$
    if $t \neq k_a$ then return $\perp$
    else return $\perp$ {Rejection rule in $\mathbf{Game}_3$}
else
    $\alpha \leftarrow \mathsf{TCR}(u_1, u_2, sd)$
    $v \leftarrow u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$
    $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$
    if $t \neq k_a$ then return $\perp$
    else return $\mathsf{Ext}(k_s, sd)$
end if
```

The leakage rate is

$$\frac{L}{|sk|} \approx \frac{L}{4\log_2 q} \approx \frac{\log_2 q - |k_s| - 128}{4\log_2 q} = \frac{1}{4} - \frac{|k_s| + 128}{4\log_2 q} \tag{28}$$

converging to $\frac{1}{4}$ when $\log_2 q$ of the DDH group becomes sufficiently large. □

6.3 Discussions

What if taking $t = k_a \oplus k_s$? For a random variable $k \in \{0,1\}^{|k_s|}$, let $\mathtt{E}$ be the event $k = k_s$, (26) becomes

$$\begin{aligned}\Pr[F_{5'}] &= \Pr[t = k_a \oplus k_s] = \Pr[t = k_a \oplus k|\mathtt{E}] \cdot \Pr[\mathtt{E}] \\ &\leq 2^{-\tilde{\mathbf{H}}_\infty(k_a|\text{leakage in } k_a)} \cdot 2^{-\tilde{\mathbf{H}}_\infty(k_s|\text{leakage in } k_s)} \\ &\leq 2^{-(|k_a|-L_a)} \cdot 2^{-(|k_s|-L_s)} = 2^{-|k_a|-|k_s|+L},\end{aligned}$$

where $L = L_a + L_s$, so that the reduction in (27) becomes

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\text{ind-cca}}(\kappa) \leq\ &\mathbf{Adv}_{\mathcal{A}_1}^{\mathsf{TCR}}(\kappa) + \mathbf{Adv}_{\mathcal{A}_2}^{\text{ddh}}(\kappa) + Q\left(2^{-|k_a|-|k_s|+L} + \mathbf{Adv}_{\mathcal{A}_5'}^{\mathsf{KDF}}(\kappa)\right) \\ &+\mathbf{Adv}_{\mathcal{A}_5}^{\mathsf{KDF}}(\kappa) + \frac{Q}{q^2},\end{aligned}$$

so that we can set $-|k_a| - |k_s| + L = -128$, or equivalently $L \approx \log_2 q - 128$ since $|k_a| + |k_s| \approx \log_2 q$. The new leakage rate is

$$\frac{L}{|sk|} \approx \frac{L}{4\log_2 q} \approx \frac{\log_2 q - 128}{4\log_2 q} = \frac{1}{4} - \frac{32}{\log_2 q} = \frac{1}{4} - o(1) \tag{29}$$

which is a little better, yet still in the same order of previous rate given in (28).

Optimal leakage rate. The rate $\frac{1}{4} - o(1)$ is *seemingly* optimal for the scheme for following reason. Suppose that the rate could be $\frac{1}{4}$. Given the challenge ciphertext $C = (u_1, u_2, t, sd)$, the adversary asks for leakage $f(sk) = u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$, which is v exactly in the scheme in Figure 9. Then the adversary can compute keys k_s and finally get the encapsulated key $K = \mathsf{Ext}(k_s, sd)$. The word "seemingly" was used due to the fact that the attack is in fact out of the security model, as leakage query is not allowed after the challenge phase.

6.4 Leakage rates on concrete groups

NIST's curves. Let us first consider P-521. The group contains points $(x, y) \in \mathbb{Z}_{p_{521}}^2$ for $p_{521} = 2^{521} - 1$ satisfying following equation for a constant b

$$y^2 = x^3 - 3x + b \pmod{p_{521}}.$$

It has order q of 521-bit length, so $\log_2 q \approx 521$. Plugging this value into (29), we have the leakage rate

$$\textbf{(over P-521)}\ \frac{1}{4} - \frac{32}{\log_2 q} \approx \frac{1}{4} - \frac{32}{521} \approx 18.85\%.$$

The injective $\mathsf{KDF}(v)$ for $v = (v_x, v_y) \in \mathbb{Z}_{p_{521}}^2$ is defined as $\mathsf{KDF}(v) = v_x \in \mathbb{Z}_{p_{521}}$ represented as a bit string of 521 bits.

The computations work with other NIST's curves over prime fields. For example,

$$\textbf{(over P-192)}\ \frac{1}{4} - \frac{32}{\log_2 q} \approx \frac{1}{4} - \frac{32}{192} \approx 8.33\%.$$

Subgroup of $\mathbb{Z}_p^*$. Take DDH group $\mathbb{G} = (\mathbb{Z}_p^*)^2$ where $p = 2q + 1$, so that $\mathbb{G}$ is of order q. If $\log_2 p \approx 1024$, then $\log_2 q \approx 1023$. Using this value in (29), we have the leakage rate

$$\frac{1}{4} - \frac{32}{\log_2 q} \approx \frac{1}{4} - \frac{32}{1023} \approx 21.87\%.$$

The injective $\mathsf{KDF} : \mathbb{G} \to \{0, 1\}^{1023}$ can be defined as in [28]

$$\mathsf{KDF}(v) = \begin{cases} v & \text{if } 0 < v < \frac{p}{2} \\ p - v & \text{if } \frac{p}{2} < v < p \end{cases}$$

where v and $p - v$ in the left is interpreted as bit strings of 1023 bits.

Comparison. At Asiacrypt 2013, a DDH-based leakage resilient PKE scheme was presented [32]. Since the PKE scheme has message space of bit strings, it can be easily modified into a KEM. The scheme in [32] is characterized by a number $\mathfrak{n} = \lceil 5/(2 - 4\delta) \rceil$ for leakage rate $\delta \in [0, 1/2)$. The main merit of [32] over ours is that the rate δ can be set larger than 1/4. The other merit is that the group can be fixed (say P-192) while $\mathfrak{n}$ gets large to obtain good rate (say 29.16%). Nevertheless, sine $\mathfrak{n} \geq 3$ at the bottom line, conditioned on the same $0 \leq \delta < 1/4$ and the same DDH group, our scheme performs better regarding computation and sizes, as illustrated in Table 4.

Table 4. Comparison of leakage-resilient KEMs in standard model based on the DDH assumption. Abbreviations in the table: **me** = multi-exponentiation, **se** = single-exponentiation, **el** = group element.

Scheme	Leakage rate δ	Encap length	[Encap]; [Decap] main costs of computation	$[pk, sk]$ size
Qin-Liu [32]	$0\% \sim 8.33\%$ ($\mathfrak{n} = 3$) $18.75\% \sim 24.99\%$ ($\mathfrak{n} = 5$) $25.11\% \sim 29.16\%$ ($\mathfrak{n} = 6$)	$(\mathfrak{n}+2)\lvert q\rvert$ + small	$[(2\mathfrak{n}+2)$ **se**]; [$\mathfrak{n}$ **me** + $\mathfrak{n}$ **se**]	$[(\mathfrak{n}^2+\mathfrak{n}+2)$ **el**, $2\mathfrak{n}$ **el**]
Ours, Figure 9	8.33% over P-192 18.85% over P-521 21.87% over $\mathbb{G} = (\mathbb{Z}_p^*)^2$	$2\lvert q\rvert$ + small	[1 **me** + 2 **se**]; [1 **me** + 0 **se**]	[4 **el**, 4 **el**]

Fig. 11. Our generic leakage-resilient KEM from hash proof system (Param, Pub, Priv).

```
KG(1^κ) :
Run Param to define
    (group, SK, PK, K,
     E, V, Λ_(·)(·), μ)
sk ←$ SK
pk ← μ(sk)
Return (pk, sk)
```

```
Encap(pk) :
Take random witness r
E = E(r) ←$ V, sd ←$ Seed
v ← Pub(pk, E, r, sd)
(k_s, k_a) ← KDF(v)
t ← k_a, K ← Ext(k_s, sd)
Return C = (E, t, sd) and K
```

```
Decap(sk, C) :
Parse C = (E, t, sd)
v ← Priv(sk, E, sd)
(k_s, k_a) ← KDF(v)
If t = k_a
    return Ext(k_s, sd)
Else return ⊥
```

6.5 Generalization: leakage-resilient KEM from HPS

The scheme is given in Figure 11. The Pub and Priv algorithms of the hash proof system need an additional input sd, which is also the seed of the randomness extractor.

Theorem 8 *The generic construction of KEM in Figure 11 is IND-CCA-secure. A quantitative reduction is given in eq.(31) in the following proof.*

Proof. The proof differs with that of Theorem 6 at the following points:

- In all games, every leakage query with an arbitrary function f is answered by $f(sk)$ in which sk is the secret key held by the simulator.
- We assume that $\mathsf{KDF}(v)$ has the same entropy as v for all $v \in \mathbb{G}$, which can be fulfilled if KDF is injective as in that case $\Pr[\mathsf{KDF}(v) = \mathsf{KDF}(w)] = \Pr[v = w]\ \forall v, w \in G$.
- In **Game**$_3$, the initialization and decryption oracle are depicted in Figure 12.
- In **Game**$_{5'}$, (24) is changed to

$$\Pr[F_{5'}] \leq 2^{-|k_a|+L} \tag{30}$$

  where $|k_a|$ is the length of k_a, and $F_{5'}$ is the event that line 8 in the modified Figure 12 is executed, namely the adversary can come up with a tag satisfying $t = k_a$ where $E \notin \mathcal{V}$.
- The final reduction becomes

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\mathrm{ind-cca}}(\kappa) \leq\ & \mathbf{Adv}_{\mathcal{A}_2}^{\mathrm{sm}}(\kappa) + Q\left(\mathbf{Adv}_{\mathcal{A}_4'}^{\mathrm{cu2}}(\kappa) + \mathbf{Adv}_{\mathcal{A}_5'}^{\mathsf{KDF}}(\kappa) + 2^{-|k_a|+L}\right) \\ & + \mathbf{Adv}_{\mathcal{A}_4}^{\mathrm{cu2}}(\kappa) + \mathbf{Adv}_{\mathcal{A}_5}^{\mathsf{KDF}}(\kappa) + \frac{Q}{|\mathcal{E}|}\end{aligned} \tag{31}$$

  ending the proof. □

Fig. 12. Oracles in $\mathbf{Game}_3$ for the proof of Theorem 8.

Initialization of the game

```
I1: $\omega \xleftarrow{\$} \mathsf{Trapdoors}$, $sd^* \leftarrow \mathsf{Seed}$
I2: $sk \xleftarrow{\$} \mathcal{SK}$, $pk \leftarrow \mu(sk)$
I3: $E^* \xleftarrow{\$} \mathcal{C} \setminus \mathcal{V}$
I4: $v^* \leftarrow \mathsf{Priv}(sk, E^*, sd^*)$
I5: $(k_s^*, k_a^*) \leftarrow \mathsf{KDF}(v^*)$
I6: $K^* \leftarrow \mathsf{Ext}(k_s^*, sd^*)$
```

Decapsulation of adversarial query $C = (E, t, sd)$

```
if $(E, sd) = (E^*, sd^*)$ then
    if $t \neq k_a^*$ then return $\perp$
    else return $K^*$
else if $E \notin \mathcal{V}$ then
    $v \leftarrow \mathsf{Priv}(sk, E, sd)$
    $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$
    if $t \neq k_a$ then return $\perp$
    else return $\perp$
else
    $v \leftarrow \mathsf{Priv}(sk, E, sd)$
    $(k_s, k_a) \leftarrow \mathsf{KDF}(v)$
    if $t \neq k_a$ then return $\perp$
    else return $\mathsf{Ext}(k_s, sd)$
end if
```

Instantiation under the DLIN assumption. Using the same notation as in Section 4.3 with

$$\alpha = \mathsf{TCR}(u_1, u_2, u_3, sd), \text{ and } \Lambda_{sk}(u_1, u_2, u_3, sd) = u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2} u_3^{z+\alpha z'}.$$

Since the secret key contains 6 elements in $\mathbb{Z}_q$, the leakage rate becomes $1/6 - o(1)$.

Instantiation under the DCR assumption. Using the same notation as in Section 4.4 with

$$\alpha = \mathsf{TCR}(u, sd), \text{ and } \Lambda_{sk}(u, sd) = u^{x+y\alpha} \bmod N_1.$$

Since the secret key $sk \in \mathcal{SK} = \{0, \dots, \lfloor N_1^2/2 \rfloor\}^2$, and hence $|sk| \approx 4|N_1|$, while $|k_a| + |k_s| \approx |N_1|$, the leakage rate becomes $1/4 - o(1)$.

7 Conclusion

While the Kurosawa-Desmedt KEM has the reputation that it is not IND-CCA-secure, we show simple modifications that make the scheme achieve the security. Our variant can be implemented on top of ISO/IEC 18033-2 since the underlying base group and symmetric building blocks are identical.

We also prepare alternatives IND-CCA-secure under the DLIN assumption, and the DCR assumption, in any setting the DDH assumption cannot be used. We also achieve leakage-resilient yet highly efficient variants by modest changes on the KEMs.

Acknowledgment

We are grateful to Takahiro Matsuda for informing us about reference [7]. We also thank Qiong Huang and the anonymous reviewers for comments that help refining this manuscript. Yoshinori Aono is also thanked for his help in debugging the C code of [1].

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. 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.
8. D. J. Bernstein. Pippenger's exponentiation algorithm, 2002. `http://cr.yp.to/papers/pippenger.pdf`.
9. D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs #1. In Krawczyk [25], pages 1–12.
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 Krawczyk [25], pages 13–25.
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. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
17. 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.
18. 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.
19. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptology*, 26(1):80–101, 2013.
20. G. Hanaoka and K. Kurosawa. Between hashed DH and computational DH: Compact encryption from weaker assumption. *IEICE Transactions*, 93-A(11):1994–2006, 2010.
21. J. Herranz, D. Hofheinz, and E. Kiltz. The Kurosawa-Desmedt key encapsulation is not chosen-ciphertext secure. *IACR Cryptology ePrint Archive*, 2006:207, 2006.
22. 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.
23. IEEE P1363a Committee. IEEE 1363a-2004: Standard Specifications For Public Key Cryptography – Amendment 1: Additional Techniques, 2004. `http://grouper.ieee.org/groups/1363/P1363a/`.
24. 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.
25. H. Krawczyk, editor. *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*. Springer, 1998.

26. 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.
27. 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.
28. K. Kurosawa, R. Nojima, and L. T. Phong. New leakage-resilient CCA-secure public key encryption. *J. Mathematical Cryptology*, 7(4):297–312, 2013.
29. K. Kurosawa and L. T. Phong. Kurosawa-Desmedt key encapsulation mechanism, revisited. In *the 7th International Conference on Cryptology in Africa - Morocco (Africacrypt 2014)*, 2014. To appear.
30. 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`.
31. PSEC-KEM website. `http://info.isl.ntt.co.jp/crypt/eng/psec/contents.html`.
32. B. Qin and S. Liu. Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter. In K. Sako and P. Sarkar, editors, *ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 381–400. Springer, 2013.
33. The Standards for Efficient Cryptography Group. SEC 1: Elliptic Curve Cryptography, 2000. `http://www.secg.org/secg_docs.htm`.
34. 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 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 [8, Section 3]. Experimental results are depicted in Figure 13, 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 $y = y_n \cdots y_1$
$W \leftarrow UV$
$Z \leftarrow 1$
for $i$ from $n$ to 1 step $-1$
    $Z \leftarrow Z^2$
    if ($x_i = 1$ and $y_i = 0$), $Z \leftarrow Z \cdot U$
    if ($x_i = 0$ and $y_i = 1$), $Z \leftarrow Z \cdot V$
    if ($x_i = 1$ and $y_i = 1$), $Z \leftarrow Z \cdot W$
Return $Z$
```

In Figure 13, 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.

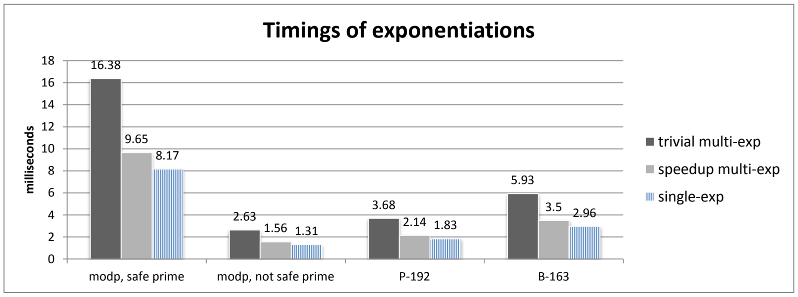


Fig. 13. Average timings of trivial and speedup computation of exponentiations, taken over 10000 executions, over various base groups. Experiment is done over a laptop (Intel 2.0GHz CPU, 8GB RAM) running Ubuntu 12.04 LTS. The C compiler is g++ 4.6.3 using NTL 6.0.0 and GMP 5.1.1 libraries.

B 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. More specifically, the algorithms KG and Decap are changed accordingly by setting $g_2 \leftarrow g_1^{\omega}$ in which $\omega \xleftarrow{\$} \mathbb{Z}_q^*$. The secret is made one element shorter. 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 $v \in \mathbb{G}$. Therefore, decapsulation can be slightly faster in the cases where group membership checks are costly (namely, in a subgroup of $\mathbb{Z}_p^*$ where p is not a safe prime). See Table 3 in Section 3.3 for a comparison with other schemes.

Fig. 14. A variant of the KEM in Figure 1.

$\mathsf{KG}(1^\kappa)$:	$\mathsf{Encap}(pk)$:	$\mathsf{Decap}(sk, C)$:
$g_1 \xleftarrow{\$} \mathbb{G}$	$r \xleftarrow{\$} \mathbb{Z}_q$	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 $\perp$