Compact CCA-Secure Encryption for Messages of Arbitrary Length

Masayuki Abe[1], Eike Kiltz[2], and Tatsuaki Okamoto[1]

[1] Information Sharing Platform Laboratories
NTT Corporation, Japan
{abe.masayuki,okamoto.tatsuaki}@lab.ntt.co.jp
[2] CWI Amsterdam, The Netherlands
kiltz@cwi.nl

Abstract. This paper proposes a chosen-ciphertext secure variant of the ElGamal public-key encryption scheme which generates very compact ciphertexts for messages of *arbitrary length*. The ciphertext overhead (i.e., the difference between ciphertext and plaintext) is one group element only. Such a property is particularly useful when encrypting short messages such as a PIN or a credit card number in bandwidth-critical environments. On top of the compact overhead, the computational cost for encryption and decryption are almost the same as plain ElGamal encryption. The security is proven based on the strong Diffie-Hellman assumption in the random oracle model.

1 Introduction

One of the most fundamental and best studied public-key encryption schemes is the ElGamal encryption scheme [13] that works over a group $\mathbb{G}$ of prime-order q with a generator g. The public-key is $h = g^x$ for a random secret index $x \in \mathbb{Z}_q$ and a ciphertext for plaintext m consists of the tuple (u, c), where $u = g^r$ and $c = m \oplus H(h^r)$. Here $H : \mathbb{G} \to \{0,1\}^{|m|}$ is some public hash function. Due to its simplicity the scheme is very efficient:

- its constituting encryption and decryption algorithms are roughly as efficient as computing one discrete exponentiation;
- its ciphertext overhead (i.e., the size of the ciphertext minus the size of the plaintext message [23]) consists of only one group element — independently of the length of the plaintext message.

However, the major drawback of the ElGamal scheme is its relatively poor security. The scheme can be proved semantically secure only against *passive attacks* and obviously insecure against stronger *active attacks*, i.e., chosen-ciphertext attacks which is nowadays considered to be the standard security notion. In this work we are interested in the question if it is at all possible to "upgrade" the security properties of the ElGamal encryption scheme to chosen-ciphertext security while at the same time retaining its high computational efficiency and compact ciphertexts.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-00468-1_29
S. Jarecki and G. Tsudik (Eds.): PKC 2009, LNCS 5443, pp. 377–392, 2009.

1.1 Our Contribution

We propose the most efficient IND-CCA secure ElGamal type encryption scheme whose ciphertext overhead consists of only one group element irrelevant to the length of the plaintext. Such a compact ciphertext overhead with IND-CCA security has been possible only when the message is sufficiently long. (See next section for details and Table 1 for summary.) On the other hand, important messages can be very short; A PIN number typically consists of 4 digits. And such a short information would be often used in applications that have sharp limitations in bandwidth or storage where saving even small amount of bits is critical. All known schemes however can't encrypt short messages securely or require extra overhead to provide chosen cipehrtext security.

In our scheme, the computational efficiency and the size of public/private keys is the same as hashed ElGamal's. It can be instantiated with any passively secure symmetric cipher, such as the one-time pad. Our scheme can be used as a plug-in alternative to ElGamal encryption that brings higher level of security. They share the same key generation algorithm, hash functions, and symmetric cipher. Furthermore, ciphertexts have the same algebraic structure.

IND-CCA-security of our scheme is proven in the random oracle model assuming the hardness of the strong Diffie-Hellman problem [1, 21].[1] The security reduction is almost tight, i.e., the respective advantages are related within a factor of $\log_2 \lambda$, where λ is the security parameter. Even though the reduction is not exactly tight, it only loses $\log_2 \lambda \approx 6$ bits of security for a security level of $\lambda = 80$ bits. Adapting the twinning technique from [9] we furthermore propose a variant of our scheme that can be proved secure under the Computational Diffie-Hellman (CDH) assumption.

1.2 Related Work

We now revisit the known frameworks to immunize the ElGamal scheme secure against chosen-ciphertext attacks, specifically focusing on the above properties of efficiency and ciphertext overhead, and point out their respective shortcomings. Here we mostly restrict ourselves to schemes that are secure in the random oracle model [4]. For quick overview, see Table 1. For an overview concerning schemes based on *trapdoor permutations* (such as OAEP) we refer to [3].

Schemes adding extra integrity. The first method achieves chosen-ciphertext security by adding an ephemeral integrity check to the ciphertext, essentially consisting of the tag of a message authentication code (MAC). Examples of such schemes are given in [4, 22] but the best known instance is probably DHIES [1] by Abdalla, Bellare and Rogaway. DHIES is as efficient as ElGamal in terms of computational efficiency. The scheme, however, has one group element

[1] The strong DH problem is computing g^{ab}, given random (g, g^a, g^b) and a restricted DDH oracle $\mathsf{DDH}_{g,a}(g^{\hat{b}}, g^{\hat{c}})$ which outputs 1 iff $a\hat{b} = \hat{c}$. Note that this is a weaker assumption than the related gap DH problem which grants access to a full DDH oracle.

plus a MAC tag of the ciphertext overhead, which is thus longer than that of ElGamal's. Its security can also be proved based on the strong Diffie-Hellman assumption in the random oracle model.[2] We remark that other schemes obtained using generic transformations (for example the Fujisaki-Okamoto transform [14] or the transformations from [10, 11]) also suffer from a similar overhead.

THE KEM/DEM FRAMEWORK. The second method is to construct a hybrid encryption scheme by combining key encapsulation (KEM) with CCA secure data encapsulation (DEM) [12]. The latter is typically constructed by adding a MAC to a one-time (passively) secure symmetric cipher (just like DHIES) but one can instead use a length-preserving strong pseudorandom permutation (PRP, e.g., [16]) to avoid such an overhead [16]. For (hashed) ElGamal this was done by Kurosawa and Matsuo [20] to obtain a variant of DHIES [1] whose ciphertexts are as compact as the ones from ElGamal.

However, there are two drawbacks with this approach. First, the strong PRPs obtained from modes of operation are generally considered to be complicated and not very efficient in practice: for example, they are all "two-pass schemes" meaning that plaintext/ciphertext need to be passed twice at encryption/decryption. Second, more importantly, a strong PRP can only *securely* encrypt messages whose length is at least one block length of the underlying symmetric cipher, i.e., typically at least 128 bits. (Smaller messages can of course be padded to one block thereby adding overhead.) In fact, this drawback is an inherent artifact of the security properties of DEMs that are CCA-secure and length-preserving (DEMs for which ciphertext-size equals plaintext-size): any length-preserving CCA-secure DEM encrypting messages of m bits can be trivially broken in time complexity 2^m.[3] Consequently, length-preserving CCA secure DEMs encrypting short messages (e.g, $m = 10$ bits) cannot exist and therefore the KEM/DEM framework is not suitable for the purpose of obtaining chosen-ciphertext secure encryption with compact ciphertexts for short messages.

TAG-KEM/DEM FRAMEWORK. Another generic method, known as the Tag-KEM/DEM framework [2], accepts the most simple form of a DEM such as a one-time pad. On the other hand, the framework is slightly more demanding on the KEM part. In particular, tag-KEMs whose ciphertexts consist only of a single group element are not known to exist so far.

[2] We remark that there also exists a standard model proof for DHIES based on the "Hashed Oracle Diffie-Hellman" assumption which is an interactive assumption involving a hash function that is close to the strong DH assumption in the random oracle model.

[3] The attack is as follows. Given an a challenge ciphertext $e^* \in \{0,1\}^m$ of an unknown message $m^* \in \{0,1\}^m$ an attacker queries the decryption oracle for all ciphertexts $e \in \{0,1\}^m \setminus \{e^*\}$. Since the cipher is length-preserving and hence a permutation on $\{0,1\}^m$ the challenge message must be the one not output by the decryption oracle. In general, a CCA-secure DEM that adds r bits of redundancy can be broken in with 2^{m+r} decryption queries.

Table 1. Efficiency comparison among some ElGamal-type CCA schemes instantiated in a group of order 2λ bits, where λ is the security parameter. An element in the group is assumed to be representable with 2λ bits, one MAC tag with λ bits. "Computation" counts the number of single exponentiations and costs for other building blocks. Here we adopt the convention that 1 multi-base exponentiation is counted as 1.5 standard exponentiations. "mac" (or "hash") counts the cost for MAC-ing (or Hash-ing, resp.) a (potentially long) message. "sprp" counts the extra cost needed for the strong PRP compared to the simple one-time secure symmetric encryption. All other computations are ignored. The assumptions are as follows. CDH: Computational DH; SDH: Strong-DH; GDH: Gap-DH; DDH: Decisional-DH. All but KD are in the random oracle model. See also footnote 2 for DHIES and KM.

Scheme	Ciphertext Overhead	Message Size	Computation Enc.	Computation Dec.	Assumption
ElGamal [13]	2λ	any	2	1	DDH (CPA only)
KD [19]	5λ	any	3.5+mac	2.5+mac	DDH
DHIES [1]	3λ	any	2+mac	1+mac	SDH
Twin DHIES [9]	3λ	any	3+mac	1.5+mac	CDH
KM [20]	2λ	$\lvert m\rvert > \lambda$	2+sprp	1+sprp	SDH
Twin KM [9]	2λ	$\lvert m\rvert > \lambda$	3+sprp	1.5+sprp	CDH
Boyen [6]	2λ	$\lvert m\rvert > 2\lambda$	3+2·hash	2+2·hash	GDH
Basic scheme (§3.2)	2λ	any	2.5+hash	1+hash	SDH
Twin scheme (§3.4)	2λ	any	3.5+hash	2+hash	CDH

Others. In independent work another Diffie-Hellman type scheme was proposed by Boyen [6]. His construction primarily focuses on a tight security reduction. However, compared to ElGamal it requires twice as much computation for encryption and decryption. More importantly, it inherits the limitation from the KEM/DEM framework and can only securely encrypt messages longer than one group element (or 2λ bits): for small messages Boyen's security proof is no longer valid and does not seem to be fixable with the same ideas.

Summary. Compared to hashed ElGamal, all known Diffie-Hellman type encryption schemes suffer from either non-optimal ciphertext expansion; or decreased computational efficiency; or a restriction in the message length; or a combination of these. Surprisingly, until today no chosen-ciphertext secure encryption scheme is known that encrypts *small messages* with only one group element of ciphertext overhead. The most efficient scheme encrypting small (and there even arbitrary) messages is DHIES whose ciphertext overhead sums to one group element plus λ bits for the MAC. (Here λ is the security parameter, e.g., $\lambda = 80, 128$ bits.)

1.3 Technical Overview

Our scheme follows the Tag-KEM/DEM framework [2], which combines symmetric and asymmetric encryption. Our scheme is designed in a redundancy-free manner so that decryption always outputs a message (and never rejects). If any part of a given ciphertext is modified, the corresponding plaintext message

becomes completely unrelated to the original plaintext. Such a property is obtained by using a hash of the symmetric part of the ciphertext (the "tag") to compute the asymmetric part of the ciphertext. Thus, modifying any part of a given ciphertext affects (directly or indirectly) the corresponding session-key and results that the symmetric part is decrypted differently.

Dealing with very short messages brings an unexpected technical difficulty in proving security; since the input to the hash function is short, its output distribution is very limited even when the hash function is modeled as a random oracle. More concretely, the adversary can exhaustively search for a short input that is consistent with a preliminary obtained output. We therefore cannot directly use the "oracle patching technique" (i.e., programmability) when the message is short. On the other hand, the security proof for long inputs can be done using a combination of techniques from [1, 2, 5].

The above difficulty is overcome by constructing two different reductions that deal with short and long messages separately and randomly use one of them hoping that the adversary chooses challenge messages of length that fits to the selected reduction algorithm. More details with intuition are shown in the proof outline given right after Theorem 1 in Section 3.3, and full proof details are in Section 4.

2 Definitions

Notation. Throughout the paper we denote by $\lambda \in \mathbb{N}$ the security parameter, i.e., we use choose our primitives such that an adversary requires "complexity" $\approx 2^{\lambda}$ to break them. Typical choices are $\lambda = 80, 128, \ldots$ bits.

Public Key Encryption. We recall the usual definitions for chosen ciphertext security. Let PKE be a public-key encryption scheme. Consider the usual chosen ciphertext attack game, played between a challenger and a adversary A:

1. Given the security parameter λ, the challenger generates a public key/secret key pair, and gives the public key to A;
2. A makes a number of *decryption queries* to the challenger; each such query is a ciphertext c; the challenger decrypts c, and sends the result to A;
3. A makes one *challenge query*, which is a pair of messages (m_0, m_1); the challenger chooses $\beta \in \{0, 1\}$ at random, encrypts m_β, and sends the resulting ciphertext c^* to A;
4. A makes more *decryption queries*, just as in step 2, but with the restriction that $c \neq c^*$;
5. A outputs $\hat{\beta} \in \{0, 1\}$.

The advantage $\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{A},\mathsf{PKE}}(\lambda)$ is defined to be $|\Pr[\hat{\beta} = \beta] - 1/2|$. The scheme PKE is said to be secure against chosen ciphertext attack if for all efficient adversaries A, the advantage $\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{A},\mathsf{PKE}}(\cdot)$ is negligible.

If we wish to analyze a scheme PKE in the random oracle model, then hash functions are replaced by random oracle queries as appropriate, and both challenger and adversary are given access to the random oracle in the above attack game.

Length-preserving symmetric encryption. A symmetric encryption scheme $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ consists of two deterministic algorithms that are used to encrypt and decrypt a message $m \in \{0,1\}^*$ with a symmetric key K from key-space $\{0,1\}^{\lambda_e}$. SE is length-preserving if for all keys K and message lengths $\ell \geq 1$, $\mathsf{E}_K(\cdot)$ is a permutation on $\{0,1\}^\ell$. We require SE to be secure against one-time attacks, where an adversary C has to distinguish $\mathsf{E}_K(m)$ from a randomly chosen bit-string of length $|m|$, where K is randomly chosen and message m is chosen by the adversary. Let $\mathbf{Adv}^{\mathrm{ot}}_{\mathsf{C},\mathsf{SE}}$ be the advantage function of an adversary C in the above distinguishing game. We say SE is secure against one-time attacks if $\mathbf{Adv}^{\mathrm{ot}}_{\mathsf{C},\mathsf{SE}}$ is negligible for all efficient adversaries C.

3 Compact CCA-Secure Encryption

3.1 Building Blocks

Prime-order groups. Let $q \approx 2^{2\lambda}$ be a prime, $\mathbb{G}$ be a (multiplicative) group of order q. In this section all arithmetics are done in $\mathbb{G}$ unless otherwise noted. The Diffie-Hellman problem is given random $(g, g^a, g^b) \in \mathbb{G}^3$, compute $g^{ab} \in \mathbb{G}$. The strong Diffie-Hellman problem [1] is the same as the DH problem, but given access to a an oracle $\mathsf{DDH}_{g,a}(\cdot,\cdot)$ that on input $g^{\hat{b}}, g^{\hat{c}}$ outputs 1 iff $a\hat{b} = \hat{c} \bmod q$. We remark that in pairing groups (also known gap groups [21]) the DDH oracle can be efficiently instantiated and hence the strong-DH problem is equivalent to the DH problem (and the gap-DH problem [21]). The advantage $\mathbf{Adv}^{\mathrm{sdh}}_{\mathsf{B},\mathbb{G}}$ is defined to be the probability that an adversary B solves the above strong-DH problem. The strong-DH assumption holds in $\mathbb{G}$ if for all efficient adversaries B, the advantage $\mathbf{Adv}^{\mathrm{sdh}}_{\mathsf{B},\mathbb{G}}$ is negligible.

Symmetric Encryption. Let $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ be a length-preserving symmetric encryption scheme that takes keys $K \in \{0,1\}^{\lambda_e}$ of length $\lambda_e = 2\lambda$ and encrypts arbitrary messages $m \in \{0,1\}^*$. We require that SE treats short and long messages in a different way as follows.

Long Messages. For all messages $m \in \{0,1\}^\ell$ of length $\ell > \lambda_e$ (larger than $|K|$), SE is secure against one-time attacks. A practical construction first expands K to a bit-string $K' \in \{0,1\}^\ell$ using a pseudo-random generator (key-derivation function), and then uses K' as an xor-based one-time pad to mask m, i.e., $e = m \oplus K'$.

Short Messages. For all messages $m \in \{0,1\}^\ell$ of length $\ell \leq \lambda_e$ (shorter than $|K|$), SE acts as a perfectly secure one-time pad by using the first ℓ bits of K as an xor-based one-time pad to mask m.

We remark that it is further possible to relax the condition for encryption of short messages to a computational one.

3.2 The Basic Scheme

Let $\mathbb{G}$ be group of prime order $q \approx 2^{2\lambda}$ and $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ be a length-preserving symmetric ciphertext using keys of length $\lambda_e \geq 2\lambda$. Let $G : \{0,1\}^* \to \mathbb{Z}_q$ and $H : \mathbb{G} \to \{0,1\}^{\lambda_e}$ be hash functions that will be modeled as random oracles.

Key-Generation. The public key consists of a random group element $g \in \mathbb{G}$ and $h = g^x \in \mathbb{G}$, where the secret key $x \in \mathbb{Z}_q$ is a random index.

Encryption. To encrypt a plaintext $m \in \{0,1\}^\ell$ of arbitrary length $\ell \geq 1$, a random $r \in \mathbb{Z}_q$ is picked and the symmetric key is computed as

$$K = H(g^r) \in \{0,1\}^{\lambda_e} .$$

Next, the ciphertext is computed as

$$e = \mathsf{E}_K(m), \quad u = \left(g^{G(e)} \cdot h\right)^r .$$

The ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$.

Decryption. Using the secret-key $x \in \mathbb{Z}_q$, decrypting a ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$ is done by first computing the symmetric key

$$K = H\left(u^{\frac{1}{G(e)+x}}\right) \in \{0,1\}^{\lambda_e}$$

and then computing the plaintext as $m = \mathsf{D}_K(e)$. Note that decryption never rejects.

We note that if $u = 1 \in \mathbb{G}$, encryption fails. This happens with negligible probability $1/q$. For simplicity this negligible encryption error will be ignored for the remainder of the paper. An error-free variant can be obtained by adjusting x and G.

To verify correctness note that $u = (g^{G(e)}h)^r = g^{r\cdot(G(e)+x)}$ which implies $u^{1/(G(e)+x)} = g^r$.

3.3 Security

Theorem 1. *Assume the strong-DH assumption holds in $\mathbb{G}$, G and H are modeled as random oracles, and SE is a symmetric cipher with the properties specified in Subsection 3.1. Then the above PKE scheme is* IND-CCA *secure.*

In particular, suppose A is an IND-CCA *adversary that runs in time τ and makes at most q_h hash queries to H, q_g hash queries to G, and q_d decryption queries. Then there exists a strong DH adversary B that runs in time at most $\tau + O(q_h q_d \cdot T)$ and makes at most $O(q_h q_d)$ oracle calls and an adversary C against SE that runs in time at most $\tau + O(q_h q_d \cdot T)$, where we denote by T the unit time to perform one exponentiation in the group $\mathbb{G}$; moreover,*

$$\mathbf{Adv}^{\mathrm{cca}}_{\mathsf{A}}(\lambda) \leq 2\lambda_e \cdot \mathbf{Adv}^{\mathrm{sdh}}_{\mathsf{B}}(\lambda) + 2\mathbf{Adv}^{\mathrm{ot}}_{\mathsf{C},\mathsf{SE}}(\lambda) + \frac{q_h + 2q_g}{2^{\lambda_e}} + \frac{q_h}{q} ,$$

where $\lambda_e \geq 2\lambda$ is the length of the symmetric keys used in SE.

Proof Outline. At a high level the reduction works as follows. Given a strong DH problem instance (z, z^a, z^b) the strong DH adversary defines the public-key in such a way that $g = z^a$ and $h = z \cdot g^{-w^*}$, for random w^*. This implicitly defines the private-key as $x = \frac{1}{a} - w^*$. This setup of the public-key is inspired by "IBE techniques" [5] and allows the strong DH adversary to simulate decryption of all ciphertext except the challenge itself. (We remark that it was also successfully applied in the proofs of other encryption scheme, see, e.g., [7, 18, 17, 9].) Furthermore, a technique from [1] is used to "patch" random-oracle and decryption oracle and make them consistent by using the DDH oracle provided by the strong-DH experiment. Here the crucial property is to be able to check efficiently if for a given ciphertext $u \in \mathbb{G}$, an $R \in \mathbb{G}$, and an index $w \neq w^*$ it holds that $R = u^{\frac{1}{w+x}}$ or not. The latter can be done by querying the DDH oracle on $\mathsf{DDH}_{z,a}(u \cdot R^{w^*-w}, R)$. The challenge ciphertext is set to (u^*, e^*), where $u^* = z^b$ and e^* is a random bitstring that equals the length of one of the challenge messages, m_0 or m_1. Furthermore, $G(e^*)$ is defined as w^* such that ${u^*}^{1/(x+G(e^*))} = z^{ab}$. Hence an IND-CCA adversary has no chance in guessing the challenge bit without querying H for z^{ab}.

One difficulty in the proof is that $w^* = G(e^*)$ is implicitly set at the beginning of the reduction. If the IND-CCA adversary decides to get challenged on long messages, it is unlikely that the random ciphertext e^* of the same length is asked to G before the challenge query is made. So the strong DH adversary *never* returns w^* to any query to G made before the challenge query. However, if the IND-CCA adversary decided to get challenged on very short messages, it may decide to query all possible e^* of that length to G before making the challenge query. So the strong DH adversary *has to* return w^* to one of these queries to be successful in this case. Accordingly, the DH adversary has to behave contrarily in the two cases, but which case happens only depends on the adversary.

To handle the above situation, we actually construct two independent strong DH adversaries where at least one of the adversaries will be successful. The first strong DH adversary is successful if the length of the challenge message is smaller than the key-length λ_e. If this is the case the reduction hopes to guess the *length* of the challenge message correctly in order to be able to define the symmetric part of the challenge ciphertext at the beginning of the experiment. Overall, given the challenge message is sufficiently small, this first strong DH adversary wins if it correctly guessed the length of the challenge ciphertext which happens with probability $1/\lambda_e$. The second strong DH adversary is successful if the length of the challenge message is larger than the key-length λ_e. In that case it hopes that the symmetric part of the challenge ciphertext remains hidden from the IND-CCA adversary's view until the actual challenge query is done. This makes it possible to patch $G(e^*)$ after the challenge query was done. Overall, given the challenge message is sufficiently large, this second strong DH adversary wins as long as the IND-CCA adversary did not guess the symmetric part of the challenge ciphertext which happens with probability $q_g/2^{\lambda_e}$. Combining the two strong DH adversaries explains the loss in the security reduction. A formal proof is given in Section 4.

3.4 The Twin Scheme: Security from Standard Diffie-Hellman

Using the "twinning technique" from [9] one can obtain a scheme which can be proven secure under the standard (i.e., not strong) Diffie-Hellman assumption by replacing $H(g^r)$ by $H(g^r, \hat{g}^r)$, where $\hat{g} = g^{\hat{x}}$ is another element from the public key. The twinned scheme is defined follows.

Key-Generation. The public key consists of a random group elements g, $h = g^x$, and $\hat{g} = g^{\hat{x}}$, where the secret key $x, \hat{x} \in \mathbb{Z}_q^2$ are random indices.

Encryption. To encrypt a plaintext $m \in \{0,1\}^\ell$ of arbitrary length $\ell \geq 1$, a random $r \in \mathbb{Z}_q$ is picked and the symmetric key is computed as

$$K = H(g^r, \hat{g}^r) \in \{0,1\}^{\lambda_e} .$$

Next, the ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$ is computed as

$$e = \mathsf{E}_K(m), \quad u = \left(g^{G(e)} \cdot h\right)^r .$$

Decryption. Using the secret-key $x, \hat{x} \in \mathbb{Z}_q$, decrypting a ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$ is done by first computing the symmetric key

$$K = H\left(u^{\frac{1}{G(e)+x}}, u^{\frac{\hat{x}}{G(e)+x}}\right) \in \{0,1\}^{\lambda_e}$$

and then computing the plaintext as $m = \mathsf{D}_K(e)$.

Theorem 2. *Assume the DH assumption holds in $\mathbb{G}$, G and H are modeled as random oracles, and SE is a symmetric cipher with the properties specified in Subsection 3.1. Then the above PKE scheme is* IND-CCA *secure.*

Proof sketch. The Strong Twin Diffie-Hellman (STDH) assumption [9] is as follows. Given $z, z^a, z^{\hat{a}}, z^b$, computing the tuple $(z^{ab}, z^{\hat{a}b})$ is computationally infeasible, even with access to a Twin DDH oracle $\mathsf{TDDH}_{z,a,\hat{a}}(z^c, z^d, z^{\hat{d}})$ that outputs 1 iff $ac = d$ and $\hat{a}c = \hat{d}$. Since it was shown in [9] that the standard CDH assumption implies the STDH assumption, it is sufficient to prove security of the above scheme assuming the STDH assumption. We now sketch this reduction, focusing mostly on the differences to the proof of Theorem 1.

Given a STDH problem instance $(z, z^a, z^{\hat{a}}, z^b)$ the STDH adversary defines the public-key in such a way that $g = z^a$, $\hat{g} = z^{\hat{a}}$, and $h = z \cdot g^{-w^*}$, for random w^*. This implicitly defines the private-keys as $x = \frac{1}{a} - w^*$ and $\hat{x} = \frac{\hat{a}}{a}$. The challenge ciphertext is set to (u^*, e^*), where $u^* = z^b$ and e^* is a random bitstring that equals the length of one of the challenge messages, m_0 or m_1. Furthermore, $G(e^*)$ is defined as w^* such that $u^{*1/(x+G(e^*))} = z^{ab}$ and $u^{*\hat{x}/(x+G(e^*))} = z^{\hat{a}b}$. Hence an IND-CCA adversary has no chance in guessing the challenge bit without querying H for the tuple $(z^{ab}, z^{\hat{a}b})$, which is exactly the challenge for the STDH adversary.

The decryption queries are simulated by patching the random oracles using the Twin DDH oracle $\mathsf{TDDH}_{z,a,\hat{a}}(\cdot,\cdot,\cdot)$. Here the crucial property is again to be

able to check efficiently if for a given ciphertext $u \in \mathbb{G}$, a tuple $(R_1, R_2) \in \mathbb{G}^2$, and an index $w \neq w^*$ it holds that $(R_1, R_2) = (u^{\frac{1}{w+x}}, u^{\frac{\hat{x}}{w+x}})$ or not. Define $\hat{R} = u \cdot R_1^{w^*-w}$. Using the equation for $x, \hat{x}$, the latter condition can be proved equivalent to $(R_1, R_2) = (\hat{R}^a, \hat{R}^{\hat{a}})$ which can be checked by calling $\mathsf{TDDH}_{z,a,\hat{a}}(\hat{R}, R_1, R_2)$. The remaining proof is similar to the one of Theorem 1. We remark that the concrete security bound of Theorem 2 is essentially the same as the one of Theorem 1 plus some small additive statistical parameter stemming from the twinning technique.

3.5 Efficiency

We note that in both schemes the value $u = (g^{G(e)} \cdot h)^r$ from encryption can be computed using one multi-exponentiation which is about as efficient as one standard exponentiation. For our basic scheme the parameters for key-generation, encryption and decryption, as well as for the size of the public/secret keys are exactly the same as for ElGamal. Furthermore, as in ElGamal, our scheme's ciphertext overhead only consists of one group element, independent of the message. We remark that implemented on certain elliptic curves this accounts to $\log q \approx 2\lambda$ bits overhead for conjectured λ bits of security (which is optimal due to the generic attacks on the discrete logarithm problem in $\mathbb{G}$). For an efficiency comparison with related schemes we refer to Table 1 in Section 1.

4 Proof of Theorem 1

We proceed in games, starting with Game 0 which is the original IND-CCA experiment run with an adversary A. Each Game i, $i \geq 0$, is entirely defined through the algorithms $\textsc{Setup}_i$, $\textsc{Challenge}_i$, $\textsc{Decrypt}_i$, $\textsc{HashH}_i$, and $\textsc{HashG}_i$, describing the adversary's view. If a game does not mention one of the above implementing algorithms then it is assumed to be unchanged compared to the last game. In Game i, $i \geq 0$, we define X_i as the event that $\beta = \hat{\beta}$. We make the general convention that all terms with a superscript asterisks $(^*)$ are (or at least will be in a future game) connected to the challenge ciphertext; for example, K^* is the symmetric key used for generating the challenge ciphertext.

Game 0. This game implements the IND-CCA experiment. We make the following conventions how the appearing random-variables are computed throughout the experiment.
$\triangleright\textsc{Setup}_0$. Given a generator $z \in \mathbb{G}$, the experiment picks random $a \in \mathbb{Z}_q^*$, $w^* \in \mathbb{Z}_q$, $B \in \mathbb{G}$ and defines

$$A = z^a, \quad R^* = B^a \,. \tag{1}$$

Next, it defines the public key as

$$g := A, \quad h := z \cdot A^{-w^*}.$$

(Note that this implicitly defines the secret key as $x = 1/a - w^*$.) We will now equivalently rewrite the rest of the IND-CCA experiment using these definitions of public/secret key.

$\triangleright$CHALLENGE$_0$. The challenge ciphertext (u^*, e^*) for message m_β of length ℓ^* is generated as

$$K^* = H(R^*), \quad e^* = \mathsf{E}_{K^*}(m_\beta), \quad u^* = B \cdot R^{*\,G(e^*)-w^*}, \tag{2}$$

where R^* and B are taken from (1).

$\triangleright$DECRYPT$_0$. Decryption of a ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$ is done as follows.

$$R = u^{\frac{a}{1+a\cdot(G(e)-w^*)}}, \quad K = H(R), \quad m = \mathsf{D}_K(e)\,. \tag{3}$$

$\triangleright$HASHH$_0$/HASHG$_0$. Hash queries to H and G are answered with random values from the respective domains.

One can easily verify that Game 0 is an equivalent rewrite of the original IND-CCA experiment. Hence,

$$|\Pr[X_0] - 1/2| = \mathbf{Adv}^{\mathrm{cca}}_{\mathsf{A}}(\lambda)\,. \tag{4}$$

Game 1. $\triangleright$HASHG$_1$=HASHG$_0$, but abort if $G(e) = G(e^*)$ for any value $e \neq e^*$.

Since there are at most q_g hash queries to G, each of them taking an independent random value in $\mathbb{Z}_q$, we have that the experiment aborts with probability at most q_g/q. Since Game 0 and 1 are equivalent until the new abort happens,

$$|\Pr[X_1] - \Pr[X_0]| \leq q_g/q\,.$$

Game 2. $\triangleright$HASHH$_2$=HASHH$_1$, but abort if $H(R^*)$ is asked before CHALLENGE$_2$ is executed.

Since R^* is perfectly hidden from A's view until CHALLENGE$_2$ is executed, we have

$$|\Pr[X_2] - \Pr[X_1]| \leq q_h/q\,.$$

Game 3. We now use an alternative (but equivalent) simulation of random oracle H and the decryption oracle. To make the simulation consistent we introduce the following two tables which are initially empty.

- $List_H[R] = K$ means $H(R) = K$, for $R \in \mathbb{G}$.
- $List_D[u, e] = K$ means that $K \in \{0,1\}^{\lambda_e}$ is the symmetric key that was used to decrypt ciphertext (u, e), for $u \in \mathbb{G}$ and $e \in \{0,1\}^\ell$.

$\triangleright$DECRYPT$_3$. Decryption of a ciphertext $(u, e) \in \mathbb{G} \times \{0,1\}^\ell$ is done as follows. As in (3) of DECRYPT$_2$, first the "algebraic key" $R = u^{\frac{a}{1+a(G(e)-w^*)}}$ is computed. Next, the symmetric key K is computed according to the following case distinction.

Case D1: If $List_H[R]$ is defined, then use $K = List_H[R]$.

Case D2: If there exists an entry $(\hat{u}, \hat{e})$ in $List_D$ such that $R = \hat{u}^{\frac{a}{1+a(G(\hat{e})-w^*)}}$, then use $K = List_D[\hat{u}, \hat{e}]$.
Case D3: In the second phase, if $R = u^{*\frac{a}{1+a(G(e^*)-w^*)}}$, then use $K = H(R')$, where $R' = (u/u^*)^{1/(G(e)-G(e^*))}$. (Note that $R' = R^*$ and, since $G(e) \neq G(e^*)$, R' is well-defined.)
Case D4: Otherwise, pick a random key $K \in \{0,1\}^{\lambda_e}$.

In all cases define $List_D[u, e] := K$ and return $m = \mathsf{D}_K(e)$.
▷HASHH$_3$. A hash query $H(R)$ is answered as follows.

Case H1: If $List_H[R]$ is defined then return this value.
Case H2: If there exists an entry $List_D[u, e]$ such that $R = u^{\frac{a}{1+a(G(e)-w^*)}}$ then use $K = List_D[u, e]$.
Case H3: Otherwise, pick a random K.

In all cases define $List_H[R] := K$ and return $H(R) = K$.
▷CHALLENGE$_3$=CHALLENGE$_2$, but now CHALLENGE$_3$ explicitly defines $List_H[R^*] := K^*$ for uniform $K^* \in \{0,1\}^{\lambda_e}$ (and not implicitly through a call to $H(R^*)$ as in (2)).

It is easy to verify that this does not change the behavior of the oracles.

$$\Pr[X_3] = \Pr[X_2] \,. \tag{5}$$

The next lemma shows that using a DDH oracle, DECRYPT$_3$ and HASHH$_3$ can be simulated without explicit knowledge of $a = \log_z A$.

Lemma 1. *Algorithms* DECRYPT$_3$ *and* HASHH$_3$ *can be efficiently simulated depending on the values z and $A = z^a$, plus making maximal $O(q_H q_D)$ calls to an oracle $\mathsf{DDH}_{z,a}(\cdot,\cdot)$.*

Proof. We start with DECRYPT$_3$. The problem is that the simulation cannot compute the value R without knowing a. However, it can perform the checks using $A = z^a$ and the DDH oracle. In case D1, to check if $List_H[R]$ is defined the simulator checks if there exists an entry $\hat{R}$ in $List_H$ satisfying $\hat{R} = (u/\hat{R}^{G(e)-w^*})^a$. The latter one is equivalent to $\mathsf{DDH}_{z,a}(u/\hat{R}^{G(e)-w^*}, \hat{R}) = 1$. In case D2, check if $\mathsf{DDH}_{z,a}(\hat{u}^{G(e)-w^*}/u^{G(\hat{e})-w^*}, u/\hat{u}) = 1$ for some entry $(\hat{u}, \hat{e})$ from $List_D$. The case D3 is similar. Furthermore, simulation of HASHH$_3$ can be done in a similar way.

Note that, using the DDH oracle, Game 3 can be simulated only knowing the values $z, B, A = z^a$. The value $R^* = B^a$ is only needed for the execution of CHALLENGE$_3$, i.e., to generate the element u^* of the challenge ciphertext (2) and to define $List_H[R^*] := k^*$. Now we would like to "define" $G(e^*) := w^*$ to be able to use $u^* := B$ (independent of R^*) for the generation of the challenge ciphertext. However, this seems difficult since e^* depends on the input m_0 and m_1 of the adversary. In particular, m_0 could be small in which case e^* (which is of the same length as m_0) can be guessed by A.

Game 4. From this game on we slightly change the way the experiment is executed. The experiment initially flips a random coin $c \in \{0,1\}$ representing a guess if the adversary wants to get challenged on short or long messages. Depending on c, the experiment is executed using the algorithms DECRYPT_4=DECRYPT_3, HASHH_4=HASHH_3, HASHG_4=HASHG_3, SETUP_4^c=SETUP_3, and CHALLENGE_4^c, where CHALLENGE_4^0 and CHALLENGE_4^1 are defined as follows.
$\triangleright\text{CHALLENGE}_4^0$=$\text{CHALLENGE}_3$, with the difference that it aborts (and returns a random bit $\hat{\beta}$) if $|m_0| > \lambda_e$. (Here m_0 is one of the challenge messages.)
$\triangleright\text{CHALLENGE}_4^1$=$\text{CHALLENGE}_3$, with the difference that it aborts if $|m_0| \leq \lambda_e$.

Since bit c is independent from the adversaries view, we have

$$\Pr[X_3] - \frac{1}{2} = 2(\Pr[X_4] - \frac{1}{2}) \,. \tag{6}$$

In Game i ($i \geq 4$), we define $\Pr[X_i^{\text{short}}] = \Pr[X_i \mid |m_0| \leq \lambda_e]$ and $\Pr[X_i^{\text{large}}] = \Pr[X_i \mid |m_0| > \lambda_e]$ such that

$$\begin{aligned}\Pr[X_4] &= \Pr[X_4^{\text{short}}] \cdot \Pr[|m_0| \leq \lambda_e] + \Pr[X_4^{\text{large}}] \cdot \Pr[|m_0| > \lambda_e] \\ &\leq \max\{\Pr[X_4^{\text{short}}], \Pr[X_4^{\text{large}}]\} \,.\end{aligned}$$

In the following games we will bound $\Pr[X_4^{\text{short}}]$ and $\Pr[X_4^{\text{large}}]$ separately.

Game 5. $\triangleright\text{SETUP}_5^0$=$\text{SETUP}_4^0$, but additionally a random bit-string $\ell^*_{\text{guess}} \in \{1, \ldots, \lambda_e\}$ is selected.
$\triangleright\text{CHALLENGE}_5^0$=$\text{CHALLENGE}_4^0$, but it aborts if $\ell^* \neq \ell^*_{\text{guess}}$, where ℓ^* is the length of the challenge message m_0 submitted by A.

Since ℓ^*_{guess} is picked uniformly from $\{1, \ldots, \lambda_e\}$, independent of the adversary's view, we have

$$\Pr[X_4^{\text{short}}] - \frac{1}{2} = \lambda_e \cdot (\Pr[X_5^{\text{short}}] - \frac{1}{2}) \,. \tag{7}$$

Game 6. $\triangleright\text{SETUP}_6^0$=$\text{SETUP}_5^0$, but additionally a bit-string $e^* \in \{0,1\}^{\ell^*_{\text{guess}}}$ is selected, uniformly at random.
$\triangleright\text{HASHG}_6^0$. Since e^* is now determined at the beginning, HASHG_6^0 can now program G such that $G(e^*) = w^*$.
$\triangleright\text{CHALLENGE}_6^0$=$\text{CHALLENGE}_5^0$. If $\ell^* = \ell^*_{\text{guess}}$ (i.e., if CHALLENGE_5^0 did not abort) it uses the tuple $(u^* := B, e^*)$ as challenge ciphertext. (Since $G(e^*) = w^*$, this is a correctly distributed ciphertext for m_β.) Furthermore, the experiment defines $List_H[R^*] := K^*$, where $K^* = K_1^* || K_2^*$ with $K_1^* = e^* \oplus m_\beta$ and K_2^* is a random bit-string of length $\lambda_e - \ell^*$.

Consider the distribution of K^* and e^* used in CHALLENGE_5^0 and CHALLENGE_6^0. By the properties of SE and since $\ell^* = \ell^*_{\text{guess}} \leq \lambda_e$, in CHALLENGE_5^0 a random key K^* is used and e^* is computed as $e^* = K_0^* \oplus m_\beta$, where K_1^* is the ℓ^* bit prefix of K^*. In CHALLENGE_6^0 a random symmetric ciphertext e^* is used together with a random key K^* that encrypts m_β to e^*. Therefore, K^* and e^* have the same joint distribution in both games and

$$\Pr[X_5^{\text{short}}] = \Pr[X_6^{\text{short}}] \,. \tag{8}$$

Game 7. $\triangleright\textsc{Challenge}_7^0$=$\textsc{Challenge}_6^0$, but $List_H[R^*]$ is left undefined.

Clearly, the view of adversary A in Games 6 and 7 is the same until it queries $H(R^*)$ in the second phase. We claim that there exists an adversary B against strong DH with

$$|\Pr[X_7^{\text{short}}] - \Pr[X_6^{\text{short}}]| \leq \mathbf{Adv}_{\mathsf{B}}^{\text{sdh}}(\lambda)\ . \tag{9}$$

Adversary B runs the experiment from this game. By Lemma 1, $\textsc{HashH}_7$=$\textsc{HashH}_3$ and $\textsc{Decrypt}_7$=$\textsc{Decrypt}_3$ can be simulated using the values z, A, and B. If $H(R^*)$ is queried, B will notice using the DDH oracle and outputs $R^* = z^{ab}$.

Now it is clear that in case $c = 0$ the experiment is independent of the challenge key K^* and hence also of the challenge bit β.

$$\Pr[X_7^{\text{short}}] = 1/2\ . \tag{10}$$

Game 8. $\triangleright\textsc{Challenge}_8^1$. For generating the challenge ciphertext (u^*, e^*) for message m_β of length ℓ^* the experiments picks a random $K^* \in \{0,1\}^{\lambda_e}$, defines $List_H[R^*] = K^*$, and computes

$$e^* = \mathsf{E}_{K^*}(m_\beta), \quad u^* = B.$$

The experiment aborts if $G(e^*)$ was already defined. Define this event as F_8^{large}. Otherwise, it defines $G(e^*) := w^*$ such that by (2) (u^*, e^*) is a correct ciphertext for m_β. Then,

$$|\Pr[X_8^{\text{large}}] - \Pr[X_7^{\text{large}}]| \leq \Pr[F_8^{\text{large}}]\ . \tag{11}$$

Game 9. $\triangleright\textsc{Challenge}_9^1$=$\textsc{Challenge}_8^1$, but after the challenge ciphertext is defined, the value $List_H[R^*]$ remains undefined.

Clearly, we have $\Pr[F_9^{\text{large}}] = \Pr[F_8^{\text{large}}]$, where F_9^{large} is the event that $G(e^*)$ was queried before the challenge ciphertext is defined. Furthermore, similar to Game 7, there exists an adversary B against the strong DH problem with

$$|\Pr[X_9^{\text{large}}] - \Pr[X_8^{\text{large}}]| \leq \mathbf{Adv}_{\mathsf{B}}^{\text{sdh}}(\lambda)\ . \tag{12}$$

Game 10. $\triangleright\textsc{Challenge}_{10}^1$=$\textsc{Challenge}_9^1$, but a random $e^* \in \{0,1\}^{\ell^*}$ is used for the challenge ciphertext (independent of m_β). Since a distinguisher between a random symmetric ciphertext e^* and $e^* = \mathsf{E}_{K^*}(m_\beta)$ (for known m_β) immediately implies an one-time adversary C against SE, we have

$$|\Pr[X_{10}^{\text{large}}] - \Pr[X_9^{\text{large}}]| \leq \mathbf{Adv}_{\mathsf{C},\mathsf{SE}}^{\text{ot}}(\lambda)\ . \tag{13}$$

Furthermore, since Game 10 is now independent of β, we have

$$\Pr[X_{10}^{\text{large}}] = 1/2\ . \tag{14}$$

Finally, e^* is a uniform element from $\{0,1\}^{\ell^*}$ with $\ell^* \geq \lambda_e$. Since adversary A makes maximal q_g hash queries to G, we have

$$\Pr[F_9^{\text{large}}] = \Pr[F_{10}^{\text{large}}] \leq q_g/2^{\lambda_e} \ . \tag{15}$$

Summing up the probabilities we obtain

$$\begin{aligned}\mathbf{Adv}^{\text{cca}}_{\mathsf{A}}(\lambda) &\leq \frac{q_h}{2^{\lambda_e}} + \frac{q_g}{q} + 2\max\{\lambda_e \cdot \mathbf{Adv}^{\text{sdh}}_{\mathsf{B}}(\lambda)\ ,\ \mathbf{Adv}^{\text{sdh}}_{\mathsf{B}}(\lambda) + \mathbf{Adv}^{\text{ot}}_{\mathsf{C},\mathsf{SE}}(\lambda) + \frac{q_g}{2^{\lambda_e}}\} \\ &\leq \frac{q_h + 2q_g}{2^{\lambda_e}} + \frac{q_g}{q} + 2\lambda_e \cdot \mathbf{Adv}^{\text{sdh}}_{\mathsf{B}}(\lambda) + 2\mathbf{Adv}^{\text{ot}}_{\mathsf{C},\mathsf{SE}}(\lambda) \ .\end{aligned}$$

Furthermore, the running times of A (and C) is bounded by the running time of B plus the time of performing additional $O(q_H q_D)$ group operations (by Lemma 1).

References

[1] Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)

[2] Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 128–146. Springer, Heidelberg (2005); also available at IACR e-print 2005/027 and 2004/194

[3] Abe, M., Kiltz, E., Okamoto, T.: Chosen Ciphertext Security with Optimal Ciphertext Overhead. In: Advances in Cryptology – Asiacrypt 2008. LNCS, vol. 5350, pp. 355–371. Springer, Heidelberg (2008); also available at IACR e-print 2008/374

[4] Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: First ACM Conference on Computer and Communication Security, pp. 62–73. Association for Computing Machinery (1993)

[5] Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

[6] Boyen, X.: Miniature CCA2 PK encryption: Tight security without redundancy. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 485–501. Springer, Heidelberg (2007)

[7] Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM CCS 2005, pp. 320–329. ACM Press, New York (2005)

[8] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: Proceedings of the 30th annual ACM Symposium on Theory of Computing, pp. 209–218 (1998)

[9] Cash, D.M., Kiltz, E., Shoup, V.: The Twin Diffie-Hellman Problem and Applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)

[10] Coron, J., Handschuh, H., Joye, M., Paillier, P., Pointcheval, D., Tymen, C.: GEM: A generic chosen-ciphertext secure encryption method. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 263–276. Springer, Heidelberg (2002)

[11] Coron, J., Handschuh, H., Joye, M., Paillier, P., Pointcheval, D., Tymen, C.: Optimal chosen-ciphertext secure encryption of arbitrary-length messages. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 17–33. Springer, Heidelberg (2002)
[12] Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
[13] El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
[14] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
[15] Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28, 270–299 (1984)
[16] Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
[17] Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
[18] Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on Gap Hashed Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007), http://eprint.iacr.org/2007/036
[19] Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
[20] Kurosawa, K., Matsuo, T.: How to remove MAC from DHIES. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 236–247. Springer, Heidelberg (2004)
[21] Okamoto, T., Pointcheval, D.: The gap-problems: a new class of problems for the security of cryptographic schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
[22] Okamoto, T., Pointcheval, D.: REACT: Rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, p. 159. Springer, Heidelberg (2001)
[23] Phan, D.H., Pointcheval, D.: Chosen-ciphertext security without redundancy. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 1–18. Springer, Heidelberg (2003)