Journal of Computer Security 21 (2013) 749–778
DOI 10.3233/JCS-130473
IOS Press

Publicly verifiable ciphertexts

Juan González Nieto [a], Mark Manulis [b], Bertram Poettering [c], Jothi Rangasamy [d] and Douglas Stebila [a,*]

[a] *Queensland University of Technology, Brisbane, QLD, Australia*
E-mails: {j.gonzaleznieto, stebila}@qut.edu.au
[b] *University of Surrey, Guildford, UK*
E-mail: mark@manulis.eu
[c] *Royal Holloway, University of London, Egham, UK*
E-mail: bertram.poettering@rhul.ac.uk
[d] *Society for Electronic Transactions and Security, Chennai, India*
E-mail: jothi.rangasamy@gmail.com

In many applications where encrypted traffic flows from an open (public) domain to a protected (private) domain there exists a gateway that bridges these two worlds, faithfully forwarding all incoming traffic to the receiver. We observe that the notion of indistinguishability against (adaptive) chosen-ciphertext attacks (IND-CCA2), which is a mandatory goal in face of active attacks in a public domain, can be relaxed to indistinguishability against chosen-plaintext attacks (IND-CPA) once the ciphertexts passed the gateway. The latter then acts as an IND-CCA2/CPA filter by first checking the validity of an incoming IND-CCA2-secure ciphertext, transforming it (if valid) into an IND-CPA-secure ciphertext, and finally forwarding it to the recipient in the private domain. Non-trivial filtering can result in reduced decryption costs on the recipient's side.

We identify a class of encryption schemes with *publicly verifiable ciphertexts* that admit generic constructions of IND-CCA2/CPA filters (with non-trivial verification). These schemes are characterized by existence of public algorithms that can distinguish ultimately between valid and invalid ciphertexts. To this end, we formally define public verifiability of ciphertexts for general encryption schemes, key encapsulation mechanisms and hybrid encryption schemes, encompassing public-key, identity-based and tag-based encryption flavours. We further analyze the security impact of public verifiability and discuss generic transformations and concrete constructions that enjoy this property.

Keywords: Ciphertext filtering, public key encryption, identity-based encryption, chosen ciphertext attacks

1. Introduction

Transmission of sensitive information over public networks necessitates the use of cryptographic protection. Modern cryptography offers various techniques, includ-

*Corresponding author: Douglas Stebila, GPO Box 2434 (2 George St, GP-O-617), Brisbane, Queensland 4001, Australia. Tel.: +61 7 3138 9566; Fax: +61 7 3138 2310; E-mail: stebila@qut.edu.au.

ing public key encryption (PKE) and identity-based encryption (IBE), by which the sender can use public information to encrypt a message only the intended receiver can decrypt. These two encryption flavours can be combined into a common syntax, called *general encryption* (GE) [1]. For longer messages, hybrid encryption schemes based on independent key and data encapsulation techniques are often more efficient (cf. KEM/DEM approach [12]).

The most standard security notion for encryption schemes is *indistinguishability* (IND) – a ciphertext may not leak any information about the encrypted message (except possibly its length) – whose definitions consider different types of attacks.

The strongest is an *adaptive chosen-ciphertext attack* (CCA2), in which an attacker can ask for decryption of ciphertexts of her own choice (other than the target ciphertext). IND-CCA2 security hence protects encrypted messages of honest senders despite the threat that receivers may also have to decrypt ciphertexts constructed by the adversary. Such threats exist if the network is susceptible to active attacks. In contrast, if senders are trustworthy and their messages are delivered over a network that protects authenticity, then security against *chosen-plaintext attacks* (CPA) would already provide sufficient confidentiality guarantees, possibly resulting in better performance.

1.1. IND-CCA2/CPA-*encrypted traffic filtering*

Consider an intermediate party, called a *gateway*, and assume that encrypted sender's messages are transmitted over a public network until they reach that gateway and are then forwarded by the gateway over a private network to the receiver. We assume that the gateway can be trusted to do this forwarding faithfully.

By the above reasoning, IND-CCA2 security would be required for the encrypted traffic from (possibly malicious) senders towards the gateway. But for messages on the internal network – including from the gateway to the receiver – IND-CPA security would be sufficient in practice to preserve confidentiality. If the gateway just forwards all (IND-CCA2) ciphertexts from the outside world without modification, all security goals remain satisfied, but perhaps we can improve efficiency for the receiver by having the gateway do some processing on ciphertexts before forwarding them.

An often observed difference between IND-CPA and IND-CCA2 schemes is that IND-CPA schemes successfully decrypt every given ciphertext, whereas IND-CCA2 schemes typically check ciphertexts for consistency and decrypt only those that are "well-formed" [8,12,21,22,24]. For such schemes, the gateway could act as a filter that would sort out inconsistent IND-CCA2 ciphertexts. There exist few IND-CCA2 schemes [4,33,34] that decrypt every ciphertext to a possibly meaningless (random) message. Such schemes are not well-suited to filtering since the gateway would need to know the receiver's private key to decide whether the message is meaningful,

which would in general be unacceptable since it requires trusting gateways for confidentiality, not just integrity.

In this paper, we are interested in solutions that allow an honest-but-curious gateway to transform IND-CCA2-secure traffic from a public network into IND-CPA-secure traffic for a private network at low cost and while fully preserving confidentiality of encrypted messages. Note that a simple approach where a gateway can decrypt ciphertexts and then re-encrypt messages for intended receivers cannot be applied here since the gateway is not supposed to learn the communication contents. The key step behind our solution is that the gateway is trusted to correctly perform a "validity check" of traffic from the public network before forwarding it on to the private network. Recipient devices on the private network can then (potentially) use a more efficient decryption procedure.

1.2. Applications of IND-CCA2/CPA *filtering*

Many applications could benefit from the described IND-CCA2/CPA filtering approach in the "sender–gateway–receiver" system model. In the following we give some application domains where the trust relationship between the gateway performing the filtering step on the incoming IND-CCA2-secure traffic and the receiver obtaining the resulting IND-CPA-secure traffic is sufficient to preserve the overall security of the communication, while resulting in better communication efficiency and network throughput.

1.2.1. Wireless sensor networks

A typical wireless sensor network (WSN) consists of many low-powered nodes that communicate with each other locally in a multi-hop fashion and use a single more powerful gateway device to communicate with the Internet.

Most battery-powered sensor nodes are particularly slow in processing public-key operations; for example, the popular Crossbow TelosB node (equipped with an 8 MHz processor and 10 kB RAM) performs one pairing computation in about 14.5 s (on a 256-bit curve), one modular exponentiation in about 10.2 s (with a 1024-bit exponent), and one elliptic curve scalar-point multiplication in about 1.5 s (on a 160-bit curve) [17,41]. Any usage of public-key operations on such nodes must be considered with care; this holds in contrast to highly efficient symmetric key algorithms (typically AES), for which sensor nodes often have hardware-based support. The majority of key distribution and secure initialization algorithms for WSNs (see Stelle et al. [40] for a recent overview) equip sensor nodes with shared keys for securing the multi-hop communication with other nodes and the gateway. Therefore, sensors only need to resort to more expensive asymmetric algorithms when communicating with the outside world.

In our filtering approach, the gateway could take IND-CCA2-secure traffic from senders in the outside world, check it for validity, and convert it to a simpler (IND-CPA-secure) format to reduce the processing costs for receiving sensor nodes.

1.2.2. Encrypted emails and DoS-resistance

As a second application example, mail servers (MTAs) generally receive emails over unprotected networks; email recipients contact these servers to access their emails after having established an end-to-end authenticated (and possibly encrypted) channel with them. This setting is for instance common to current web-based mail hosting servers, such as Google Mail, to which clients can connect using TLS. The trust assumption underlying this model matches the assumed trust relationship between the receiver and the gateway in our abstract "sender–gateway–receiver" scenario; the mail server acting as a gateway is trusted to guarantee the integrity of clients' emails. Today we observe an increasing use of mobile devices for email communication. Their power constraints are still known to be a limitation and processing encrypted emails can constitute an additional burden on the available resources.

A requirement to send and receive encrypted emails, although not widely adopted today, may eventually be prescribed by a policy of a company or institution. Mobile device users would then potentially face the risk of DoS attacks from processing encrypted emails or attachments. An attacker may send "garbage" emails, aiming to lure the user into performing the decryption operation that will result in a waste of resources. The DoS attack remains reasonable from the attacker's point of view if generation of "garbage" emails can be performed at a low cost without performing the actual encryption operation.

Our filtering approach may help thwart such attacks for emails and attachments encrypted with an IND-CCA2-secure cryptosystem. The idea is to let the mail server perform a "sanity check" and filter out inconsistent ciphertexts, thus saving the client from downloading and processing such emails.

1.2.3. Electronic elections

A frequent step within the electronic election process to guarantee vote privacy is to let voters submit their votes, encrypted under some public key as input to a mix-net comprised of several servers holding shares of the corresponding secret key to obtain a (publicly verifiable) permuted list of votes whilst hiding the permutation that was applied during the mixing process. For efficiency and re-randomization reasons, most mix-net approaches require homomorphic properties of the input ciphertexts. In order to ensure fair election, votes must however remain independent such that some sort of non-malleability for submitted ciphertexts becomes necessary. Wikström [42] identified the above problem and proposed a solution by defining a special class of cryptosystems, called *augmented cryptosystems*, where partial knowledge of the secret key can be used to essentially lift the non-malleability of ciphertexts, thus enabling their homomorphic processing. Wikström [42] proved that Cramer–Shoup PKE scheme [12] possesses all the properties of an augmented cryptosystem. In order to ensure public verifiability of the permuted votes, servers are required to reveal parts of the secret key. As acknowledged in [42], this is a major limitation of the approach since each new election process requires setting up a new public key.

Given that non-malleability under adaptive chosen ciphertext attacks is equivalent to IND-CCA2-security, the concept of augmented cryptosystems from [42] may at

first be seen helpful to realize IND-CCA2/CPA filtering. However, as discussed in the next section the requirement of using parts of the secret key to perform filtering in general can essentially be seen as a drawback. Moreover, using IND-CCA2/CPA filters where non-malleability property can be securely lifted without knowledge of the secret key would in fact enable re-usability of the public key and thus result in a more efficient solution to the problem addressed in [42].

1.3. Ciphertext consistency checks

IND-CCA2-secure encryption schemes where 'invalid' ciphertexts are filtered out based on explicit consistency checks seem very suitable for our purposes. Consistency checks can be either private or public: the check is private if it requires at least partial knowledge of the private key (such as in [12]); public checks do not require any secrets (e.g. in [8,21]).

We will focus on IND-CCA2-secure cryptosystems with *publicly verifiable ciphertexts*. Interestingly, public verifiability has been treated so far in a rather folklore manner, as a property of concrete schemes [8,21,22,24]. To make use of this property in general, for example to enable "black-box" constructions of higher-level security protocols from publicly verifiable encryption schemes, a more formal and thorough characterization of public verifiability is merited. We also note that public verifiability has been extensively addressed in a different context, namely with regard to *threshold encryption*, where, as observed initially by Lim and Lee [29] and then provably realized in several schemes [5,8,28,39], this property is useful to make the threshold decryption process of an IND-CCA2-secure threshold encryption scheme non-interactive and robust.

In our applications, public verifiability can immediately be used to detect and filter out invalid IND-CCA2 ciphertexts by trusting the gateway to perform the check. This filtering could also be performed for IND-CCA2 schemes with private consistency checks, as long as these checks need only parts of the private key that are by themselves not sufficient to break IND-CPA security. Existence of such IND-CCA2 schemes has been demonstrated by Persiano [32] through his concept of *trapdoor cryptosystems*. For instance, he proved that a trapdoor containing private-key components that are used in the consistency check of Cramer–Shoup PKE [12] cannot be used for an IND-CPA attack (although their disclosure allows malleability attacks). Being concerned about IND-CCA2-security, Persiano argued that existence of such trapdoors is a drawback. Taking a look at Persiano's trapdoor cryptosystems [32] from the perspective of our work, we observe that the gateway could indeed be given trapdoor information to check IND-CCA2 consistency *without* losing IND-CPA security. However, this approach would offer somewhat weaker guarantees in contrast to publicly verifiable schemes: if the delegated trapdoor keys are ever leaked, then IND-CCA2 security can never be recovered. This contrasts with our approach, in which the receiver always has the potential to obtain IND-CCA2 security at any particular time, simply by performing more operations.

1.4. Contributions

1.4.1. Definitions

We formalize the property of *publicly verifiable ciphertexts* for general encryption (Section 2), general KEMs (Section 3) and general KEM/DEM hybrid schemes (Section 4). Our definitions emphasize the role of public ciphertext consistency checks within the decryption procedure. In our approach, decryption algorithms of publicly verifiable schemes follow a strictly modular design where the consistency check can be performed independently of the remaining "lightweight" decryption procedure. Success or failure of the entire decryption procedure is indicated ultimately by the consistency check, which can be performed by any third party without access to any secret information. The only exception is the KEM/DEM approach, where we relax these conditions to account for decryption failures in the DEM part. Our definitions employ the syntax of generalized encryption by Abdalla et al. [1] which we extend to also capture *tag-based encryption* (TBE) [3,21] and to address KEMs and the KEM/DEM framework.

With these definitions, we first prove in Section 2.2 the very general statement that any IND-CCA2-secure scheme with publicly verifiable ciphertexts remains at least IND-CPA-secure if the underlying consistency check is outsourced from the decryption procedure. We focus in particular on the case where the verification algorithm is non-trivial, meaning that the public consistency check fails exactly when the IND-CCA2-secure scheme's decryption algorithm fails, as such publicly verifiable schemes can readily be used to build the aforementioned IND-CCA2/CPA filters.

1.4.2. Publicly verifiable PKEs

We provide several constructions (general and concrete) of IND-CCA2-secure schemes with public verifiability.

First, we show (Sections 2.3–2.4) that some well-known general ways for obtaining IND-CCA2 secure schemes readily offer a form of *inconclusive public verifiability*, meaning there may be ciphertexts that pass verification but still fail decryption. This holds in particular for the Canetti–Halevi–Katz (CHK) transform [9] and the non-interactive zero knowledge (NIZK)-based transform [30,36]. The result on CHK contrasts with the related transform by Boneh and Katz [7] that uses a message authentication code (MAC) and does not offer public verifiability.

Next, we give several concrete PKE schemes with publicly verifiable ciphertexts and lightweight decryption algorithms. Scheme 1 (Section 2.5) is inspired in part by a KEM by Kiltz and its security is based on the Hashed Diffie–Hellman assumption. It offers an especially lightweight decryption procedure for ciphertexts that pass its public verification, consisting of only a single exponentiation. The scheme also makes use of a one-time signature scheme for integrity checking. This contrasts with Scheme 2 (Section 2.6), which removes the need for the one-time signature scheme, making encryption and verification less expensive, albeit by increasing the public key by one group element.

1.4.3. Publicly verifiably KEMs and hybrid encryption

In addition to PKE, we also consider KEMs in Section 3 and give examples of public key-based, identity-based and tag-based KEMs with public verification. In Section 4, we look into the KEM/DEM paradigm and show that public verification of the KEM partially carries over to the hybrid scheme. More precisely, we define a notion of *partial* public verification for hybrid encryption schemes by linking a failure in the hybrid decryption process to a verification failure in either the KEM or the DEM, and show that, by outsourcing the KEM consistency check, the hybrid construction remains at least IND-CPA-secure.

1.4.4. Efficiency

Table 1 compares the efficiency of our schemes with some previous standard-model PKEs and KEMs, including previous schemes with public consistency checks. Our most computationally efficient Scheme 2 matches or even beats the efficiency of previous PKE schemes. Scheme 1 offers the smallest public keys, albeit with a slightly higher computational cost compared to other schemes. Note that compared to Hanaoka–Kurosawa KEM, our schemes are slightly less efficient but are in fact PKEs, whereas the aforementioned KEM requires the overhead of a DEM to achieve the functionality of PKE.

2. Publicly verifiable ciphertexts in general encryption

We start by recalling the definition of *general encryption* (GE) of Abdalla et al. [1], which we extend here to also address the notion of *tag-based encryption* (TBE) [3,21].

2.1. Definition: *General encryption*

A *general encryption* (GE) scheme $\mathsf{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ consists of four algorithms:

$\mathsf{PG}(1^k)$: On input security parameter 1^k, the parameter generation algorithm PG returns public parameters par and a master secret key msk. Public parameters include a description of an identity space IDSp, a message space MsgSp, and a tag space TagSp. Note that we assume all messages in MsgSp are of the same length.

$\mathsf{KG}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$: On input par, msk and $\mathsf{id} \in \mathsf{IDSp}$, the key generation algorithm KG produces an encryption key ek and decryption key dk.

$\mathsf{Enc}(\mathsf{par}, \mathsf{ek}, M, t)$: On input par, ek, a message $M \in \mathsf{MsgSp}$, and a tag $t \in \mathsf{TagSp}$, the encryption algorithm Enc outputs a ciphertext C.

$\mathsf{Dec}(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C, t)$: On input par, ek, dk, C and a tag $t \in \mathsf{TagSp}$, the decryption algorithm Dec returns either a plaintext message $M \in \mathsf{MsgSp}$, or $\perp$ to indicate that it rejects.

Table 1
Efficiency comparison of standard-model public key encryption schemes and KEMs

Scheme	PubVerify?	Assumptions	Tight?	Size of		Cost* of		
				public key	ciphertext overhead	Enc	Dec	Dec′
Decisional (DDH or DBDH)-based schemes								
Cramer–Shoup (CS1b) [12]	N	DDH, TCR	Y	$4G$	$3G$	$1ME + 3E$	$3E$	–
Boneh et al. (gap groups) [6]	N	DBDH, TCR	Y	$4G$	$3G$	$1ME + 2E$	$1.5E + 1P$	–
Boyen et al. (gap groups) [8]	Y	DBDH, CR	N	$(k+2)G$	$2G$	$1ME + 2E$	$1E + 1P$	$1P$
Lai et al. (gap groups) [27]	Y	DBDH, CR	Y	$4G$	$3G$	$1ME + 2E$	$2E + 1P$	$1E + 1P$
KEM: Kurosawa–Desmedt [26]	N	DDH, TCR	Y	$3G$	$2G$	$1ME + 2E$	$1ME$	–
HDH-based schemes								
Kiltz (gap groups) [23, Appx B]	Y	HDH, CR	N	$(k+1)G$	$2G$	$1ME + 2E$	$2E$	$1E$
KEM: Cash et al. (a) [10]	Y	HDH, TCR	Y	$4G$	$3G$	$1ME + 3E$	$3E$	$1E$
Hanaoka–Kurosawa Section 5 [18]	Y	HDH, TCR	Y	$4G$	$2G + \lvert\text{mac}\rvert$	$1ME + 2E$	$1ME$	$1E$
Scheme 1 (Fig. 4)	Y	HDH, TCR	Y	$2G$	$2G + \lvert\sigma\rvert + \lvert vk\rvert$	$1ME + 2E + 1S$	$2E + 1V$	$1E$
Scheme 2 (Fig. 5)	Y	HDH, CR	Y	$3G$	$2G + \mathbb{Z}_p$	$1ME + 2E$	$2E$	$1E$

Notes: G – size of group element; k – security parameter; *ME* – cost of group multi-exponentiation; E – cost of group exponentiation; P – cost of pairing; σ – one-time signature; S – cost of computing σ; V – cost for verifying σ; vk – verification key for σ; $\mathbb{Z}_p$ – integer mod p.

The above GE formalism encompasses public-key, identity-based and tag-based encryption flavours:

Public-Key Encryption (PKE): Let $\mathsf{msk} = \epsilon$ (an empty string) and let IDSp and TagSp contain a single fixed element. (Hence IDSp and TagSp can be omitted in the notation of PKE algorithms.)

Identity-Based Encryption (IBE): Let KG always output $\mathsf{ek} = \mathsf{id}$ and let TagSp contain a single fixed element. (Hence TagSp can be omitted in the notation of IBE algorithms.)

Tag-Based Encryption (TBE): Let $\mathsf{msk} = \epsilon$ and let IDSp contain a single fixed element. (Hence IDSp can be omitted in the notation of TBE algorithms.)

2.1.1. Correctness

A general encryption scheme $\mathsf{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is *correct* if, for all $(\mathsf{par}, \mathsf{msk}) \leftarrow \mathsf{PG}$, all identities $\mathsf{id} \in \mathsf{IDSp}$, all $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KG}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$, all plaintexts $M \in \mathsf{MsgSp}$ and all tags $t \in \mathsf{TagSp}$, we have $\mathsf{Dec}(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, \mathsf{Enc}(\mathsf{par}, \mathsf{ek}, M, t), t) = M$ with probability one, where the probability is taken over the coins of Enc.

2.1.2. Indistinguishability

The $\mathsf{IND\text{-}CCA2}/\mathsf{CPA}$ security games between a challenger and an adversary $\mathcal{A}$ are defined by the experiments in Fig. 1 (left column). The advantage of $\mathcal{A}$ in these games is defined as

$$\mathbf{Adv}^{\mathsf{IND\text{-}xxx}}_{\mathcal{A},\mathsf{GE}}(k) = \left|\Pr\left(\mathbf{Exp}^{\mathsf{IND\text{-}xxx},0}_{\mathcal{A},\mathsf{GE}}(k) = 1\right) - \Pr\left(\mathbf{Exp}^{\mathsf{IND\text{-}xxx},1}_{\mathcal{A},\mathsf{GE}}(k) = 1\right)\right|,$$

where $\mathsf{xxx} \in \{\mathsf{CPA}, \mathsf{CCA2}\}$. A GE scheme is $\mathsf{IND\text{-}xxx}$-secure if the advantage of any PPT adversary $\mathcal{A}$ in the corresponding game is negligible in the security parameter k.

2.2. General encryption with publicly verifiable ciphertexts

In our definition of general encryption with publicly verifiable ciphertexts, we require (a) the existence of a separate algorithm for ciphertext validation and (b) that the scheme's original decryption procedure can be logically divided into this public validation check and a following 'lightweight' decryption algorithm.

Definition 1 (Publicly verifiable GE). A general encryption scheme $\mathsf{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is said to be *publicly verifiable*[1] with respect to auxiliary algorithms Ver and Dec' if:

(1) $\mathsf{Dec}(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C, t)$ has the same input/output behavior as the following sequence of operations:

[1] Note a change in terminology compared to the extended abstract version of this work [16], where we used the term "strictly non-trivial" publicly verifiable.

$\mathbf{Exp}^{\mathsf{IND\text{-}xxx},b}_{\mathcal{A},\mathsf{GE}}(k)$:

(1) $U, V, \mathit{KList}, \mathit{DList} \leftarrow \emptyset$
(2) $(\mathsf{par}, \mathsf{msk}) \xleftarrow{\$} \mathsf{PG}(1^k)$
(3) $(st, M_0, M_1, \mathsf{id}^*, t^*) \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}_{\mathsf{EK}}, \mathcal{O}_{\mathsf{DK}}[, \mathcal{O}_{\mathsf{Dec}}]}(\mathsf{par})$
If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{EK}}(\mathsf{id})$:
(a) If $\mathsf{id} \in U$ then return $\perp$
(b) $U \leftarrow U \cup \{\mathsf{id}\}$
(c) $(\mathsf{ek}[\mathsf{id}], \mathsf{dk}[\mathsf{id}]) \xleftarrow{\$} \mathsf{KG}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$
(d) Append $(\mathsf{id}, \mathsf{ek}[\mathsf{id}], \mathsf{dk}[\mathsf{id}])$ to *KList*
(e) Answer $\mathcal{A}$ with $\mathsf{ek}[\mathsf{id}]$
If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{DK}}(\mathsf{id})$:
(a) If $\mathsf{id} \notin U$ then return $\perp$
(b) $V \leftarrow V \cup \{\mathsf{id}\}$
(c) Answer $\mathcal{A}$ with $\mathsf{dk}[\mathsf{id}]$ from *KList*
If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Dec}}(C, \mathsf{id}, t)$ (if $\mathsf{xxx} = \mathsf{CCA2}$):
(a) If $\mathsf{id} \notin U$ then return $\perp$
(b) $M \leftarrow \mathsf{Dec}(\mathsf{par}, \mathsf{ek}[\mathsf{id}], \mathsf{dk}[\mathsf{id}], C, t)$
(c) Append (C, id, t) to *DList*
(d) Answer $\mathcal{A}$ with M
(4) If $\mathsf{id}^* \notin U$ then return $\perp$
(5) $C^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{par}, \mathsf{ek}[\mathsf{id}^*], M_b, t^*)$
(6) $b' \xleftarrow{\$} \mathcal{A}_2^{\mathcal{O}_{\mathsf{EK}}, \mathcal{O}_{\mathsf{DK}}[, \mathcal{O}_{\mathsf{Dec}}]}(st, C^*)$
Answer queries as above
(7) If $\mathsf{id}^* \in V$ then return $\perp$
(8) If $(C^*, \mathsf{id}^*, t^*) \in \mathit{DList}$ then return $\perp$
(9) Return b'.

$\mathbf{Exp}^{\mathsf{IND\text{-}xxx},b}_{\mathcal{A},\mathsf{GKEM}}(k)$:

(1) $U, V, \mathit{KList}, \mathit{DList} \leftarrow \emptyset$
(2) $(\mathsf{par}, \mathsf{msk}) \xleftarrow{\$} \mathsf{PG}(1^k)$
(3) $(st, \mathsf{id}^*, t^*) \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}_{\mathsf{EK}}, \mathcal{O}_{\mathsf{DK}}[, \mathcal{O}_{\mathsf{Dec}}]}(\mathsf{par})$
If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{EK}}(\mathsf{id})$:
(a) If $\mathsf{id} \in U$ then return $\perp$
(b) $U \leftarrow U \cup \{\mathsf{id}\}$
(c) $(\mathsf{ek}[\mathsf{id}], \mathsf{dk}[\mathsf{id}]) \xleftarrow{\$} \mathsf{KG}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$
(d) Append $(\mathsf{id}, \mathsf{ek}[\mathsf{id}], \mathsf{dk}[\mathsf{id}])$ to *KList*
(e) Answer $\mathcal{A}$ with $\mathsf{ek}[\mathsf{id}]$
If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{DK}}(\mathsf{id})$:
(a) If $\mathsf{id} \notin U$ then return $\perp$
(b) $V \leftarrow V \cup \{\mathsf{id}\}$
(c) Answer $\mathcal{A}$ with $\mathsf{dk}[\mathsf{id}]$ from *KList*
If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Dec}}(C, \mathsf{id}, t)$ (if $\mathsf{xxx} = \mathsf{CCA2}$):
(a) If $\mathsf{id} \notin U$ then return $\perp$
(b) $K \leftarrow \mathsf{Decap}(\mathsf{par}, \mathsf{ek}[\mathsf{id}], \mathsf{dk}[\mathsf{id}], C, t)$
(c) Append (C, id, t) to *DList*
(d) Answer $\mathcal{A}$ with K
(4) If $\mathsf{id}^* \notin U$ then return $\perp$
(5) $(C^*, K_0^*) \xleftarrow{\$} \mathsf{Encap}(\mathsf{par}, \mathsf{ek}[\mathsf{id}^*], t^*)$
(6) $K_1^* \xleftarrow{\$} \mathsf{KeySp}(k)$
(7) $b' \xleftarrow{\$} \mathcal{A}_2^{\mathcal{O}_{\mathsf{EK}}, \mathcal{O}_{\mathsf{DK}}[, \mathcal{O}_{\mathsf{Dec}}]}(st, C^*, K_b^*)$
Answer queries as above.
(8) If $\mathsf{id}^* \in V$ then return $\perp$
(9) If $(C^*, \mathsf{id}^*, t^*) \in \mathit{DList}$ then return $\perp$
(10) Return b'.

Fig. 1. IND-CCA2/CPA security experiments for General Encryption (*left*) and General Key Encapsulation (*right*). Note that M_0 and $M_1 \in \mathsf{MsgSp}$ are by definition of the same length.

(a) $C' \leftarrow \mathsf{Ver}(\mathsf{par}, \mathsf{ek}, C, t)$
(b) If $C' = \perp$, then return $\perp$
(c) $M \leftarrow \mathsf{Dec}'(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C', t)$
(d) Return M

(2) Ver and Dec′ satisfy the following:

$\mathsf{Ver}(\mathsf{par}, \mathsf{ek}, C, t)$: Given public parameters par, encryption key ek, ciphertext C and tag t, this algorithm outputs either $\perp$ if the ciphertext fails the validation, or a (transformed) ciphertext C'. Note that Ver does not take any secrets as input.

$\mathsf{Dec}'(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C', t)$: This algorithm takes input public parameters par, encryption and decryption keys (ek, dk), ciphertext C' and tag t, and outputs a message $M \in \mathsf{MsgSp}$ or $\perp$.

(3) and it holds that, for all $(\mathsf{par}, \mathsf{msk}) \leftarrow \mathsf{PG}(1^k)$, $\mathsf{id} \in \mathsf{IDSp}$, $t \in \mathsf{TagSp}$ and $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KG}(\mathsf{par}, \mathsf{msk}, \mathsf{id})$,

(a) $\mathsf{Ver}(\mathsf{par}, \mathsf{ek}, C, t) = \perp \Leftrightarrow \mathsf{Dec}(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C, t) = \perp$ for all C, and
(b) there exists a ciphertext C for which $\mathsf{Dec}(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C, t) = \perp$.

Hereafter, when we say $\mathsf{Dec} = \mathsf{Dec}' \circ \mathsf{Ver}$, we mean that algorithm Dec is composed of two algorithms Ver and Dec' according to the above construction.

Condition 3(a) requires that successful public verification is both necessary and sufficient for the regular decryption algorithm not to fail. Condition 3(b) formally excludes IND-CCA2-secure schemes where Dec never outputs $\perp$ (cf. [4,33,34] where modified (challenge) ciphertexts decrypt to random messages, and it is left to the receiver to decide whether the decrypted message is "meaningful"). While condition 3(b) may not completely exclude degenerate schemes with "useless" public verification, it at least partially captures the intuition that the public verification operation should reject all ciphertexts that do not decrypt to "meaningful" messages.

Note that, formally, all IND-CCA2-secure general encryption schemes trivially achieve parts 1 and 2 of the definition, with respect to the trivial algorithms $\mathsf{Ver}(\mathsf{par}, \mathsf{ek}, C, t) := C$ and $\mathsf{Dec}' := \mathsf{Dec}$. However, such a trivial scheme does not achieve the part 3 of the definition. We are usually interested in the case where something non-trivial is occurring in Ver: where the consistency check is essential for successful decryption. (Note that this separation does not formally ensure that Dec' is more efficient than Dec, though in practice we are of course interested primarily in such schemes.) However, occasionally we consider cases where this is not necessarily the case, leading us to define:

Definition 2 (Inconclusive public verification). Let $\mathsf{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a general encryption scheme that is publicly verifiable with respect to auxiliary algorithms Ver and Dec'. Algorithm Ver is said to be an *inconclusive public verification* algorithm if it satisfies conditions 1 and 2 of Definition 1, but not necessarily condition 3.

The following Theorem 1 shows that any IND-CCA2-secure GE scheme with publicly verifiable ciphertexts remains at least IND-CPA-secure if its decryption algorithm Dec is replaced with Dec'. Since, in the original decryption procedure, a verification process may change the ciphertext, we must ensure that ciphertexts output by the encryption algorithm of the new scheme can be processed with Dec' algorithm. A potential (syntactical) mismatch is resolved via post-processing of original ciphertexts using Ver and by viewing this step as part of the new encryption algorithm:

Theorem 1. *Let* $\mathsf{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *be an* IND-CCA2*-secure general encryption scheme that is publicly verifiable with respect to* Ver *and* Dec'. *Let* $\mathsf{Enc}' := \mathsf{Ver} \circ \mathsf{Enc}$ (*where* $\circ$ *denotes the obvious composition*) *and* $\mathsf{GE}' := (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}', \mathsf{Dec}')$.

For every IND-CPA *adversary* $\mathcal{A}$ *against* GE′ *there exists an* IND-CCA2 *adversary* $\mathcal{B}$ *against* GE *such that, for all* $k \geqslant 0$, $\mathbf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\text{GE}'}(k) \leqslant \mathbf{Adv}^{\text{IND-CCA2}}_{\mathcal{B},\text{GE}}(k)$, *where* $\mathcal{B}$ *has* (*asymptotically*) *the same running time as* $\mathcal{A}$.

Proof. Let $\mathcal{A}$ be an adversary that breaks the IND-CPA security of GE′ and runs in time $t_{\mathcal{A}}$. We build an algorithm $\mathcal{B}$ running in time $t_{\mathcal{B}}$ that, using $\mathcal{A}$ as a sub-routine, breaks the IND-CCA2 security of GE. Let $\mathcal{C}_{\text{GE}}$ denote the challenger in the associated IND-CCA2 security game for GE.

Algorithm $\mathcal{B}$ interacts with $\mathcal{C}_{\text{GE}}$ and $\mathcal{A}$. With $\mathcal{A}$, $\mathcal{B}$ acts as a challenger playing the IND-CPA security game for GE′. In detail, $\mathcal{B}$ does the following: On input public par, $\mathcal{B}$ forwards them on to $\mathcal{A}$. At some point $\mathcal{A}$ outputs the challenge consisting of two messages M_0 and M_1, a target identity id^* and a target tag t^*. $\mathcal{B}$ forwards M_0 and M_1 along with id^* and t^* to GE challenger $\mathcal{C}_{\text{GE}}$, which in turn responds with a ciphertext C^* on M_b^* for a random bit b (unknown to $\mathcal{B}$). Since C^* is publicly verifiable, $\mathcal{B}$ hands $\bar{C}^* \leftarrow \mathsf{Ver}(\mathsf{par}, \mathsf{ek}[\mathsf{id}^*], C^*, t^*)$ as the challenge ciphertext over to $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs a bit b', which $\mathcal{B}$ uses as it own output.

Queries of $\mathcal{A}$ to the oracles $\mathcal{O}_{\text{EK}}$ and $\mathcal{O}_{\text{DK}}$ are answered by $\mathcal{B}$ as follows:

- $\mathcal{O}_{\text{EK}}(\mathsf{id})$: $\mathcal{B}$ queries $\mathcal{O}_{\text{EK}}(\mathsf{id})$ to $\mathcal{C}_{\text{GE}}$ and responds to $\mathcal{A}$ with whatever it receives from $\mathcal{C}_{\text{GE}}$. Note that $\mathcal{A}$ is allowed to query $\mathcal{O}_{\text{EK}}$ on id^*.
- $\mathcal{O}_{\text{DK}}(\mathsf{id})$: $\mathcal{B}$ queries $\mathcal{O}_{\text{DK}}(\mathsf{id})$ to $\mathcal{C}_{\text{GE}}$ and responds to $\mathcal{A}$ with whatever it receives from $\mathcal{C}_{\text{GE}}$. Note that $\mathcal{A}$ is not allowed to query $\mathcal{O}_{\text{DK}}$ on id^*.

The total running time of $\mathcal{B}$ is $t_{\mathcal{B}} \leqslant t_{\mathcal{A}} + t_{\mathsf{Ver}}$ with $t_{\mathcal{A}}$ being the running time of $\mathcal{A}$ and t_{Ver} being the execution time of Ver.

Given the above perfect simulation of oracles, $\mathcal{B}$ clearly breaks the IND-CCA2 security of GE whenever $\mathcal{A}$ breaks the IND-CPA security of GE′. □

2.3. Publicly verifiable ciphertexts through CHK transformation

Canetti, Halevi and Katz [9] described a method for constructing an IND-CCA2-secure public key encryption scheme PKE from any IND-CPA-secure identity-based encryption scheme IBE with identity-space $\{0,1\}^{\ell_s(k)}$ and any strongly unforgeable *one-time signature* scheme $\mathsf{OTS} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Vrfy})$ with verification key space $\{0,1\}^{\ell_s(k)}$ (see [9] for the syntax of OTS and the details of the construction; note that one-time signature schemes can be obtained from any one-way function [35]). Later, Kiltz [21] showed that the CHK transform also works if the IND-CPA-secure IBE scheme is replaced by a weakly IND-CCA2-secure tag-based encryption scheme TBE with tag-space $\{0,1\}^{\ell_s(k)}$. In both cases (whether based on IBE or TBE), the CHK transform picks a freshly chosen OTS signing key *sigk* to sign the ciphertext c produced by the underlying encryption scheme and outputs c together with the corresponding signature σ and the public key vk. Upon decryption, σ is verified before c gets decrypted with the decryption algorithm of the underlying scheme.

PKE.Enc(par, ek, M):

(1) $(vk, sigk) \xleftarrow{\$}$ OTS.KG(1^k)
(2) If IBE-based:
$c \leftarrow$ IBE.Enc(par, vk, M)
If TBE-based:
$c \leftarrow$ TBE.Enc(par, ek, M, vk)
(3) $\sigma \leftarrow$ OTS.Sign($sigk, c$)
(4) Return $C = (c, \sigma, vk)$

PKE.Ver(par, ek, C):

(1) $(c, \sigma, vk) \leftarrow C$
(2) If OTS.Vrfy(c, σ, vk) $= \perp$
then return $\perp$
(3) Return $C' = (c, vk)$

PKE.Dec′(par, ek, dk, C'):

(1) $(c, vk) \leftarrow C'$
(2) If IBE-based:
$usk_{vk} \leftarrow$ IBE.KG(par, dk, vk)
$M \leftarrow$ IBE.Dec(par, vk, usk_{vk}, c)
If TBE-based:
$M \leftarrow$ TBE.Dec(par, ek, dk, c, vk)
(3) Return M

Fig. 2. PKE with publicly verifiable ciphertexts from CHK transformation.

Figure 2 (which uses our GE notation) shows that, in both cases, the resulting PKE is at least inconclusively publicly verifiable with respect to PKE.Ver and PKE.Dec′. In the IBE-based case, ek $= \epsilon$ remains empty, while dk $=$ msk and par are output by IBE.PG. In the TBE-based case, ek and dk are output by TBE.KG using par generated by TBE.PG. The original IBE-based transform by Canetti et al. [9] and its TBE-based version from Kiltz [21] are obtained by setting PKE.Dec $=$ PKE.Dec′ $\circ$ PKE.Ver.

An interesting question with regard to the CHK transformation is whether it achieves (conclusive) public verifiability. The statement PKE.Ver(par, ek, C) $= \perp \Rightarrow$ PKE.Dec(par, ek, dk, C) $= \perp$ from Definition 1 holds by construction for any underlying IBE and TBE schemes. For the other direction, we need to take a closer look into the specification of the PKE.Dec′ algorithm. We observe that PKE.Dec′(par, ek, dk, C') $= \perp$ may occur only if $\perp$ is among the possible outputs of the underlying algorithms IBE.Dec or TBE.Dec. Given that the TBE scheme needs to be at least weakly IND-CCA2-secure, the TBE-based CHK transformation is inconclusive. The case of IBE-based CHK transformation is slightly different as the underlying IBE scheme needs to be at least IND-CPA-secure. This alone is not sufficient for obtaining conclusive public verification in general, since the decryption procedure of an IND-CPA-secure IBE scheme may still output $\perp$. For instance, an IBE scheme constructed in a hybrid manner from an IND-CPA-secure KEM and an IND-CCA2-secure DEM would remain IND-CPA-secure, yet the corresponding DEM decryption algorithm may still output $\perp$. On the other hand, if we consider only the class of IND-CPA-secure IBE schemes for which $\perp$ is not among the possible outputs of IBE.Dec (and this class represents the majority of direct IND-CPA-secure IBE constructions known today) then PKE.Dec(par, ek, dk, C) $= \perp \Rightarrow$ PKE.Ver(par, ek, C) $= \perp$ would hold for the resulting CHK transform, whose public verifiability would indeed be conclusive. This leads us to the following claim.

Claim 1. If the underlying IBE scheme is IND-CPA-secure and $\perp$ is not among the possible outputs of the IBE.Dec algorithm then the PKE construction in Figure 2 is publicly verifiable according to Definition 1.

2.4. Publicly verifiable ciphertexts using NIZKs

Basing on the Naor–Yung approach [30], any IND-CPA-secure public key encryption scheme $\mathsf{PKE}' = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ can be converted into an IND-CCA2-secure one by extending it with a non-interactive zero-knowledge (NIZK) proof (P, V) with simulation soundness, as proven by Sahai [36]. The private/public key pair of the resulting scheme, PKE, is given by $(\mathsf{dk}, \mathsf{ek}) = ((\mathsf{dk}_1, \mathsf{dk}_2), (\mathsf{ek}_1, \mathsf{ek}_2, \rho))$ where $(\mathsf{dk}_i, \mathsf{ek}_i)$, $i \in \{1, 2\}$, are obtained from two independent runs of $\mathsf{PKE}'.\mathsf{KG}$, and ρ is the common reference string of the NIZK proof system for languages of the form

$$\big\{(c_1, c_2, \mathsf{ek}_1, \mathsf{ek}_2)\colon\ \exists M, r_1, r_2 \text{ s.t. } c_1 = \mathsf{PKE}'.\mathsf{Enc}(\mathsf{par}, \mathsf{ek}_1, M; r_1) \\ \wedge\, c_2 = \mathsf{PKE}'.\mathsf{Enc}(\mathsf{par}, \mathsf{ek}_2, M; r_2)\big\},$$

where M and random tapes r_1, r_2 used in the encryption process take the role of the witness. As demonstrated in Fig. 3, IND-CCA2 schemes obtained by this transformation directly offer at least inconclusive public verifiability. This reasoning also applies to the NIZK-based constructions of Elkind and Sahai [14] and to the first IND-CCA2-secure PKE scheme by Dolev, Dwork and Naor [13] (that uses NIZK-proofs in a slightly different way). Although NIZK-based schemes are generally regarded as not efficient, we notice that their lightweight decryption procedure Dec' (if the scheme is viewed from the public verifiability perspective) is as efficient as that of our CHK-based schemes (cf. Fig. 2).

Observe that the NIZK-based transformation in fact offers conclusive public verification according to Definition 1 if it is used with an IND-CPA-secure PKE' scheme for which $\perp$ is not among the possible outputs of $\mathsf{PKE}'.\mathsf{Dec}$ algorithm. Although not

PKE.Enc(par, ek, M):	PKE.Ver(par, ek, C):
(1) $(\mathsf{ek}_1, \mathsf{ek}_2, \rho) \leftarrow \mathsf{ek}$	(1) $(\mathsf{ek}_1, \mathsf{ek}_2, \rho) \leftarrow \mathsf{ek}$
(2) $c_1 \leftarrow \mathsf{PKE}'.\mathsf{Enc}(\mathsf{par}, \mathsf{ek}_1, M; r_1)$	(2) $(c_1, c_2, \pi) \leftarrow C$
(3) $c_2 \leftarrow \mathsf{PKE}'.\mathsf{Enc}(\mathsf{par}, \mathsf{ek}_2, M; r_2)$	(3) If $V(\rho, (c_1, c_2, \mathsf{ek}_1, \mathsf{ek}_2), \pi) = \perp$ then return $\perp$
(4) $\pi \leftarrow P((M, r_1, r_2), (c_1, c_2, \mathsf{ek}_1, \mathsf{ek}_2), \rho)$	(4) Return $C' = c_1$
(5) Return $C = (c_1, c_2, \pi)$	
	PKE.Dec′(par, ek, dk, C'):
	(1) $(\mathsf{dk}_1, \mathsf{dk}_2) \leftarrow \mathsf{dk}$
	(2) $c_1 \leftarrow C'$
	(3) $M \leftarrow \mathsf{PKE}'.\mathsf{Dec}(\mathsf{par}, \mathsf{ek}_1, \mathsf{dk}_1, c_1)$
	(4) Return M

Fig. 3. PKE with publicly verifiable ciphertexts from NIZK-based transformation.

generally required, this property does hold for the majority of direct IND-CPA-secure public-key encryption schemes.

Claim 2. If the underlying PKE' scheme is IND-CPA-secure and $\perp$ is not among the possible outputs of the $\mathsf{PKE}'.\mathsf{Dec}$ algorithm then the PKE construction in Fig. 3 is publicly verifiable according to Definition 1.

It seems, however, that the Naor–Yung approach [30] and the NIZK-based approach from [14] could offer conclusive public verifiability per Definition 1 in general if the corresponding proof of validity is modified such that it proves not only the validity of the IND-CPA-secure encryption process but also that the encrypted message does not correspond to $\perp$, either using a dedicated symbol that represents $\perp$ within the message space of PKE' scheme or by considering $\perp$ as an output of some relation that can be proven within the NIZK proof. This leads us to the following claim.

Claim 3. If the underlying PKE' scheme is IND-CPA-secure, $\perp$ is among the possible outputs of the $\mathsf{PKE}'.\mathsf{Dec}$ algorithm, and π is a NIZK proof for languages of the form

$$\big\{(c_1, c_2, \mathsf{ek}_1, \mathsf{ek}_2)\colon \exists M, r_1, r_2 \text{ s.t. } c_1 = \mathsf{PKE}'.\mathsf{Enc}(\mathsf{par}, \mathsf{ek}_1, M; r_1) \\ \wedge\, c_2 = \mathsf{PKE}'.\mathsf{Enc}(\mathsf{par}, \mathsf{ek}_2, M; r_2) \wedge M \neq \perp\big\},$$

then the PKE construction in Fig. 3 is publicly verifiable according to Definition 1.

2.5. *Scheme* 1: *HDH-based PKE with publicly verifiable ciphertexts*

In this section, we propose a practical IND-CCA2-secure PKE scheme with (conclusive) public verification as per Definition 1, which hence is well-suited for IND-CCA2/CPA filters described in the introduction due to an especially lightweight algorithm Dec'. Our construction is inspired by the IND-CCA2 public-key KEM of Kiltz [22], which when plugged into a KEM/DEM framework would yield an IND-CCA2-secure PKE scheme (but only obtain partial public verification as discussed in Section 4). In contrast, we obtain a (conclusively) publicly verifiable PKE in a more direct way, by using the encapsulated key in [22] as a one-time pad for the message and by linking the resulting ciphertext components together with a one-time signature, whose verification key is in turn bound to the KEM ciphertext part through a tweak on the original scheme from [22].

2.5.1. *The scheme*

Our PG algorithm is similar to [22] except that it uses gap groups: $\mathsf{PG}(1^k)$ outputs public parameters $\mathsf{par} = (\mathbb{G}, p, g, \mathsf{DDH}, \mathsf{H})$ where $\mathbb{G} = \langle g \rangle$ is a multiplicative cyclic group of prime order p, $2^k < p < 2^{k+1}$, DDH is an efficient algorithm such that

PKE.KG(par):

(1) $x \xleftarrow{\$} \mathbb{Z}_p^*$
(2) $u \leftarrow g^x$, $v \xleftarrow{\$} \mathbb{G}$
(3) $\mathsf{ek} \leftarrow (u, v)$, $\mathsf{dk} \leftarrow x$
(4) Return $(\mathsf{ek}, \mathsf{dk})$

PKE.Enc(par, ek, M):

(1) $(u, v) \leftarrow \mathsf{ek}$
(2) $(vk, sigk) \xleftarrow{\$} \mathsf{OTS.KG}(1^k)$
(3) $r \xleftarrow{\$} \mathbb{Z}_p^*$, $c_1 \leftarrow g^r$
(4) $t \leftarrow \mathsf{TCR}(c_1, vk)$, $\pi \leftarrow (u^t v)^r$
(5) $K \leftarrow \mathsf{H}(u^r)$, $c_2 \leftarrow M \oplus K$
(6) $c \leftarrow (c_1, c_2, \pi)$
(7) $\sigma \leftarrow \mathsf{OTS.Sign}(sigk, c)$
(8) Return $C = (c, \sigma, vk)$

PKE.Ver(par, ek, C):

(1) $(u, v) \leftarrow \mathsf{ek}$
(2) $(c, \sigma, vk) \leftarrow C$
(3) $(c_1, c_2, \pi) \leftarrow c$
(4) $t \leftarrow \mathsf{TCR}(c_1, vk)$
(5) If $\mathsf{DDH}(c_1, u^t v, \pi) \neq 1$ or $\mathsf{OTS.Vrfy}(c, \sigma, vk) = \perp$ return $\perp$
(6) Return $C' = (c_1, c_2)$

PKE.Dec′(par, ek, dk, C'):

(1) $(c_1, c_2) \leftarrow C'$
(2) $x \leftarrow \mathsf{dk}$
(3) $K \leftarrow \mathsf{H}(c_1^x)$, $M \leftarrow c_2 \oplus K$
(4) Return M

Fig. 4. Scheme 1: PKE with publicly verifiable ciphertexts and small public keys, HDH-based, gap groups.

$\mathsf{DDH}(g^a, g^b, g^c) = 1 \Leftrightarrow c = ab \bmod p$, and $\mathsf{H} : \mathbb{G} \to \{0,1\}^{\ell_1(k)}$ is a cryptographic hash function such that $\ell_1(k)$ is a polynomial in k. We also use a strong one-time signature scheme $\mathsf{OTS} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Vrfy})$ with verification key space $\{0,1\}^{\ell_2(k)}$ such that $\ell_2(k)$ is a polynomial in k and a target collision resistant hash function $\mathsf{TCR} : \mathbb{G} \times \{0,1\}^{\ell_2(k)} \to \mathbb{Z}_p$. The message space is $\mathsf{MsgSp} = \{0,1\}^{\ell_1(k)}$. The scheme works as shown in Fig. 4.

2.5.2. Security analysis

First we give intuition why our scheme is IND-CCA2-secure. Let (c^*, σ^*, vk^*) be the challenge ciphertext. As we discussed above, without the CHK transform, the proposed PKE can be seen as a KEM/DEM combination which is at least IND-CPA-secure due to Herranz et al. [19]. As for the KEM, the Hashed Diffie–Hellman (HDH) assumption [2] can be used to prove the IND-CPA security of the resulting PKE. Note that the message does not depend on vk^*, and σ^* is just the signature on c^*. Therefore c^* being an output of the IND-CPA-secure scheme hides the value of the chosen b from the adversary.

We now claim that the IND-CCA2 adversary $\mathcal{A}$ may access a decryption oracle but gains no help in guessing the value of b. Suppose the adversary submits a ciphertext $(c', \sigma', vk') \neq (c^*, \sigma^*, vk^*)$ to the decryption oracle. Now there are two cases: (a) $vk' = vk^*$ or (b) $vk' \neq vk^*$. When $vk' = vk^*$, the decryption oracle will output $\perp$ as the adversary fails to break the underlying strongly unforgeable one-time signature scheme with respect to vk'. When $vk' \neq vk^*$, the attacker $\mathcal{B}$ against the HDH problem can set the public keys as seen in the IND-CCA2 security proof for the KEM by Kiltz [22] such that (1) $\mathcal{B}$ can answer except for the challenge ciphertext all decryption queries from $\mathcal{A}$ even without the knowledge of the secret key and (2) $\mathcal{B}$ solves HDH if $\mathcal{A}$ wins. This security analysis is reflected in the following theorem.

Theorem 2. *Assume that* TCR *is a target collision resistant hash function and* OTS *is a strongly unforgeable one-time signature scheme. Under the Hashed Diffie–Hellman assumption for* $\mathbb{G}$ *and* H, *PKE Scheme* 1 (PKE.KG, PKE.Enc, PKE.Dec $=$ PKE.Dec$'$ $\circ$ PKE.Ver) *specified in Fig.* 4 *is* IND-CCA2*-secure.*

Proof. Let $\mathcal{A}$ be a PPT adversary that breaks the IND-CCA2 security of the PKE scheme with non-negligible advantage, makes at most q decryption queries and runs in time $t_{\mathcal{A}}$. We construct an algorithm $\mathcal{B}$ running in time $t_{\mathcal{B}}$ that uses $\mathcal{A}$ as a subroutine and breaks the HDH assumption with non-negligible advantage.

Before describing the algorithm $\mathcal{B}$, we define the event Forge and find an upper bound for the probability that it occurs. Let (c^*, σ^*, vk^*) be the challenge ciphertext given by $\mathcal{B}$ to $\mathcal{A}$. Let Forge be the event that $\mathcal{A}$ submits to the decryption oracle a ciphertext $(c, \sigma, vk) \neq (c^*, \sigma^*, vk^*)$ such that $(c, \sigma) \neq (c^*, \sigma^*)$ but $\mathsf{OTS.Vrfy}(c, \sigma, vk^*) = 1$. This event also includes the case that such a query is submitted by $\mathcal{A}$ before it receives the challenge ciphertext and therefore $(c, \sigma, vk) \neq (c^*, \sigma^*, vk^*)$ is not needed in this case. This implies that $\mathcal{A}$ can be used to forge the underlying one-time signature scheme OTS with probability $\Pr_{\mathcal{A}}[\mathsf{Forge}]$. The scheme OTS being a strongly unforgeable one-time signature scheme implies that $\Pr_{\mathcal{A}}[\mathsf{Forge}]$ must be negligible in the security parameter k.

We now describe how $\mathcal{B}$ proceeds on input a random instance of the HDH problem $(u = g^a, g^b, W)$. The goal of $\mathcal{B}$ is to decide whether $W = \mathsf{H}(g^{ab})$ or W is a random bit string of appropriate length. $\mathcal{B}$ interacts with $\mathcal{A}$ as a challenger in the IND-CCA2 security game for PKE. In detail, $\mathcal{B}$ proceeds as follows.

Setup. $\mathcal{B}$ runs the key generation algorithm of OTS to generate $(vk^*, sigk^*)$. Then, $\mathcal{B}$ selects $d \xleftarrow{\$} \mathbb{Z}_p^*$, computes part of the challenge ciphertext for $\mathcal{A}$ to be $(c_1^*, \pi^*) \leftarrow (g^b, (g^b)^d)$. Now $\mathcal{B}$ computes $t^* \leftarrow \mathsf{TCR}(c_1^*, vk^*)$ and $v \leftarrow u^{-t^*} \cdot g^d$ and sets the public key as (u, v). We say a ciphertext $(c = (c_1, \pi, c_2), \sigma, vk)$ is consistent if $\pi = (u^t v)^r$ for $t \leftarrow \mathsf{TCR}(c_1, vk)$ and $r = \log_g c_1$. Note that for a consistent ciphertext, the setup of the public keys implies that $\pi = (u^t v)^r = (u^t u^{-t^*} g^d)^r = (u^r)^{t-t^*} c_1^d$ and $\mathsf{H}(u^r) = \mathsf{H}((\pi / c_1^d)^{(t-t^*)^{-1}})$. Then, $\mathcal{B}$ runs $\mathcal{A}$ on input the public key (u, v).

Decryption query: phase 1. Adversary $\mathcal{A}$ may query decryption oracle with a ciphertext (c, σ, vk) for which $\mathcal{B}$ proceeds as follows:

- If $\mathsf{OTS.Vrfy}(c, \sigma, vk) \neq 1$, then $\mathcal{B}$ returns $\perp$.
- If $\mathsf{OTS.Vrfy}(c, \sigma, vk) = 1$ and $vk = vk^*$, then the event Forge happens, so $\mathcal{B}$ halts and outputs a random bit.
- If $\mathsf{OTS.Vrfy}(c, \sigma, vk) = 1$ and $vk \neq vk^*$, then for $C = (c_1, \pi, c_2)$, $\mathcal{B}$ computes $t \leftarrow \mathsf{TCR}(c_1, vk)$ and $u^t v$ and verifies the consistency of the ciphertext by checking $\pi \stackrel{?}{=} (u^t v)^r$, i.e. $\mathcal{B}$ aborts if $\mathsf{DDH}(c_1, u^t v, \pi) \neq 1$. Otherwise there are three cases based on $t \leftarrow \mathsf{TCR}(c_1, vk)$:

Case 1: $t = t^*$ and $c_1 = c_1^*$: Since $\mathcal{B}$ hides c_1^* information theoretically from $\mathcal{A}$, the probability that $c_1 = c_1^*$ is at least q/p, with q being an upper bound on the number of decryption queries $\mathcal{A}$. In this case, $\mathcal{B}$ outputs a random bit and aborts.

Case 2: $t = t^*$ and $c_1 \neq c_1^*$: In this case $\mathcal{B}$ finds a collision $c_1 \neq c_1^*$ but $\mathsf{TCR}(c_1, vk) = \mathsf{TCR}(c_1^*, vk^*)$. So, $\mathcal{B}$ outputs the collision and aborts.

Case 3: $t \neq t^*$: In this case $\mathcal{B}$ decrypts the message successfully as $m \leftarrow \mathsf{H}((\pi/c_1^d)^{(t-t^*)^{-1}}) \oplus c_2$ and returns the message m to $\mathcal{A}$.

Challenge. At some point, $\mathcal{A}$ outputs two messages M_0 and M_1 of the same length. Using the already computed challenge ciphertext part $(c_1^*, \pi^*) \leftarrow (g^b, (g^b)^d)$ and $(vk^*, sigk^*)$, $\mathcal{B}$ computes $c_2^* \leftarrow W \oplus M_\delta$ for a random bit δ and sets the challenge ciphertext for $\mathcal{A}$ to be (c^*, σ^*, vk^*), where $c^* \leftarrow (c_1^*, \pi^*, c_2^*)$ and $\sigma^* \leftarrow \mathsf{OTS.Sign}(sigk^*, c^*)$.

Decryption query: phase 2. $\mathcal{A}$ may continue querying the decryption oracle except for the challenge ciphertext. $\mathcal{B}$ answers these queries as before.

Guess. $\mathcal{A}$ outputs a bit δ'. If $\delta = \delta'$, then $\mathcal{B}$ outputs $\gamma = 1$, which means that $\mathcal{B}$'s guess is $W = \mathsf{H}(g^{ab})$. If $\delta \neq \delta'$, then $\mathcal{B}$ outputs $\gamma = 0$, which means that $\mathcal{B}$'s guess is that W is a random string.

From the above we see that unless $\mathcal{B}$ receives c_1^* from $\mathcal{A}$ (Case 1) or finds a collision in TCR, $\mathcal{B}$ simulates the view of $\mathcal{A}$ perfectly, as in the original PKE security experiment.

If $\mathcal{A}$ wins, then $\mathcal{B}$ also wins. Therefore, we have $\forall k \geqslant 0$,

$$\mathbf{Adv}_{\mathcal{B}}^{\mathrm{HDH}}(k) \geqslant \mathbf{Adv}_{\mathcal{A},\mathsf{PKE}}^{\mathsf{IND\text{-}CCA2}}(k) - \mathbf{Adv}_{\mathsf{TCR},\mathsf{H}}^{\text{Hash-colli}}(k) - \Pr_{\mathcal{A}}[\mathsf{Forge}] - q/p.$$

The total running time of $\mathcal{B}$ is $t_{\mathcal{B}} \leqslant t_{\mathcal{A}} + O(t_{\mathbb{G}})$, where $t_{\mathcal{A}}$ is the running time of $\mathcal{A}$ and $t_{\mathbb{G}}$ is the time to perform a basic operation in $\mathbb{G}$. □

2.5.3. Efficiency

Recall that Table 1 compares the efficiency of our scheme with previous standard model PKEs and KEMs.

Note that compared to some of the KEMs in the table, Scheme 1 is less efficient. However to construct a PKE, adding a DEM to those KEMs impacts either performance, ciphertext size, or uses stronger cryptographic assumptions as noted by Kiltz and Galindo [25, Sections 5.2, 7.3].

In Scheme 1, public keys consist of 2 group elements, the ciphertext overhead is 2 group elements, a one-time signature and a one-time verification key, encryption requires 3.5 group exponentiations (using simultaneous exponentiation) and 1 one-time signature, verification requires 1 group exponentiation, 2 pairings, and 1

one-time signature verification, and lightweight decryption requires only one exponentiation.

Amongst existing PKE constructions with public consistency checks, only two seem to offer the same efficiency for lightweight decryption: Kiltz [22] describes a (direct) PKE construction (in addition to KEM) that is publicly verifiable with the same lightweight decryption cost of 1 group exponentiation, but at the cost of requiring public keys with the number of group elements being linear in the security parameter, as opposed to only 2 group elements in the public key of our scheme. Hanaoka and Kurosawa [18] describe a publicly verifiable KEM that, when combined with a DEM, would yield a partially (see Section 4) publicly verifiable PKE. Its lightweight decryption would require 1 group exponentiation (plus any costs from the DEM) but its public keys would contain 3 group elements, compared to 2 group elements in our scheme.

2.6. *Scheme* 2: *A more efficient HDH-based PKE with publicly verifiable ciphertexts*

We now describe another HDH-based scheme using the technique due to Lai et al. [27]. We use a parameter generation algorithm PG that is the same as in the HDH-based PKE schemes in Fig. 4. Additionally we use a collision-resistant hash function $\mathsf{CR}: \mathbb{G} \times \{0,1\}^{\ell_1(k)} \to \mathbb{Z}_p$. Our Scheme 2 with publicly verifiable ciphertexts according to Definition 1 is specified in Fig. 5. Compared with Scheme 1 in the previous subsection, Scheme 2 has shorter ciphertexts. Compared with Lai et al.'s scheme [27], Scheme 2 has smaller public keys and is more efficient.

Theorem 3. *Assume that* CR *is a collision resistant hash function. Under the Hashed Diffie–Hellman assumption for* $\mathbb{G}$ *and* H, *PKE Scheme* 2 ($\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec} = \mathsf{PKE.Dec}' \circ \mathsf{PKE.Ver}$) *specified in Fig.* 5 *is* IND-CCA2*-secure.*

$\mathsf{PKE.KG}(\mathsf{par})$:

(1) $x, y, z \xleftarrow{\$} \mathbb{Z}_p$
(2) $u \leftarrow g^x, v \leftarrow g^y, w \leftarrow g^z$
(3) $\mathsf{ek} \leftarrow (u, v, w)$
(4) $\mathsf{dk} \leftarrow (x, y, z)$
(5) Return $(\mathsf{ek}, \mathsf{dk})$

$\mathsf{PKE.Enc}(\mathsf{par}, \mathsf{ek}, m)$:

(1) $r, s \xleftarrow{\$} \mathbb{Z}_p$; $c_1 \leftarrow g^r$
(2) $K \leftarrow \mathsf{H}(u^r)$
(3) $c_2 \leftarrow m \oplus K$
(4) $t \leftarrow \mathsf{CR}(c_1, c_2)$
(5) $\pi \leftarrow (u^t v^s w)^r$
(6) $C \leftarrow (c_1, c_2, \pi, s)$

$\mathsf{PKE.Ver}(\mathsf{par}, \mathsf{ek}, C)$):

(1) $(c_1, c_2, \pi, s) \leftarrow C$
(2) $t \leftarrow \mathsf{CR}(c_1, c_2)$
(3) $C' \leftarrow (c_1, c_2)$
(4) If $\mathsf{DDH}(c_1, u^t v^s w, \pi) \neq 1$ then set $C' = \perp$
(5) Return C'

$\mathsf{PKE.Dec}'(\mathsf{par}, \mathsf{dk}, C')$:

(1) If $C' = \perp$, then return $\perp$
(2) $(c_1, c_2) \leftarrow C'$
(3) $K \leftarrow \mathsf{H}(c_1^x)$
(4) $m \leftarrow c_2 \oplus K$
(5) Return m

Fig. 5. Scheme 2: PKE with publicly verifiable ciphertexts and more efficient computation, HDH-based, gap group.

Proof. Let $\mathcal{A}$ be a PPT adversary that breaks the IND-CCA2 security of the PKE scheme with non-negligible advantage, makes at most q decryption queries and runs in time $t_{\mathcal{A}}$. We construct an algorithm $\mathcal{B}$ running in time $t_{\mathcal{B}}$ that uses $\mathcal{A}$ as a subroutine and breaks the HDH assumption with non-negligible advantage.

We now describe how $\mathcal{B}$ proceeds on input a random instance of the HDH problem $(u = g^a, g^b, W)$. The goal of $\mathcal{B}$ is to decide whether $W = \mathsf{H}(g^{ab})$ or W is a random bit string of appropriate length. $\mathcal{B}$ interacts with $\mathcal{A}$ as a challenger in the IND-CCA2 security game for PKE. In detail, $\mathcal{B}$ proceeds as follows.

Setup. $\mathcal{B}$ selects $x_v, x_w, y_v, y_w \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and sets $u \leftarrow g^a$, $v \leftarrow g^{ax_v} g^{y_v}$ and $w \leftarrow g^{ax_w} g^{y_w}$. Now $\mathcal{B}$ chooses a collision resistant hash function $\mathsf{CR} : \mathbb{G} \times \{0,1\}^{\ell_1(k)} \rightarrow \mathbb{Z}_p$ and sets the public key as (u, v, w). The private key (x, y, z) is $(a, ax_v + y_v, ax_w + y_w)$ that is unknown to $\mathcal{B}$.

We say a ciphertext $c = (c_1, c_2, \pi, s)$ is consistent if $\pi = (u^t v^s w)^r$ for $t \leftarrow \mathsf{CR}(c_1, c_2)$ and $r = \log_g c_1$. Note that for a consistent ciphertext, the setup of the public keys implies that $\pi = (u^t v^s w)^r = ((g^a)^t (g^{ax_v} g^{y_v})^s (g^{ax_w} g^{y_w}))^r = (g^{at} g^{asx_v} g^{sy_v} g^{ax_w} g^{y_w})^r = (g^{a(t+sx_v+x_w)} g^{sy_v+y_w})^r = g^{ar(t+sx_v+x_w)} g^{r(sy_v+y_w)}$ and hence

$$\mathsf{H}\big(u^r\big) = \mathsf{H}\big(\pi / c_1^{sy_v+y_w}\big)^{(t+sx_v+x_w)^{-1}}.$$

Then, $\mathcal{B}$ runs $\mathcal{A}$ on input the public key (u, v, w).

Decryption query: phase 1. Adversary $\mathcal{A}$ may query the decryption oracle with a ciphertext $c = (c_1, c_2, \pi, s)$ for which $\mathcal{B}$ proceeds as follows: First $\mathcal{B}$ computes $t \leftarrow \mathsf{CR}(c_1, c_2)$, $u^t v^s w$, and verifies the consistency of the ciphertext by checking $\pi \stackrel{?}{=} (u^t v^s w)^r$: $\mathcal{B}$ aborts if $\mathsf{DDH}(c_1, u^t v^s w, \pi) \neq 1$. Then $\mathcal{B}$ checks if $t + sx_v + x_w = 0$. If yes, then $\mathcal{B}$ aborts; otherwise, $\mathcal{B}$ decrypts the message successfully as $m \leftarrow \mathsf{H}(\pi / c_1^{sy_v+y_w})^{(t+sx_v+x_w)^{-1}} \oplus c_2$ and returns the message m to $\mathcal{A}$.

Challenge. At some point, $\mathcal{A}$ outputs two messages M_0 and M_1 of the same length. Now $\mathcal{B}$ constructs the challenge ciphertext as follows:

(1) Selects a random bit δ.
(2) Sets $c_2^* \leftarrow W \oplus M_\delta, c_1^* \leftarrow g^b, t^* \leftarrow \mathsf{CR}(c_1^*, c_2^*)$.
(3) Then, sets $s^* \leftarrow -(t^* + x_w)/x_v$ and $\pi^* \leftarrow (g^b)^{(s^* y_v + y_w)}$.
(4) Finally returns the ciphertext $c^* = (c_1^*, c_2^*, \pi^*, s^*)$ to $\mathcal{A}$.

Note that by construction the challenge ciphertext is a consistent ciphertext. Also it has the correct distribution whenever $W = \mathsf{H}(g^{ab})$ else it is independent of δ from $\mathcal{A}$'s view.

Decryption query: phase 2. $\mathcal{A}$ may continue querying the decryption oracle for a ciphertext $c \neq c^* = (c_1^*, c_2^*, \pi^*, s^*)$. $\mathcal{B}$ now does the following.

(1) Check whether $c = (c_1^*, c_2, \pi^*, s^*)$ and $\mathsf{CR}(c_1^*, c_2) = t^*$. In this case $\mathcal{B}$ finds a collision in CR. So, $\mathcal{B}$ outputs the collision and aborts.
(2) Check whether $t + sx_v + x_w = 0$ where $t \leftarrow \mathsf{CR}(c_1, c_2)$. If yes, $\mathcal{B}$ outputs a random bit and aborts, else $\mathcal{B}$ decrypts the message successfully as $m \leftarrow \mathsf{H}(\pi / c_1^{sy_v + y_w})^{(t+sx_v+x_w)^{-1}} \oplus c_2$ and returns the message m to $\mathcal{A}$.

Note that the values x_v and x_w are blinded using y_v and y_w, respectively and thus they are hidden from $\mathcal{A}$ initially. Also when $\mathcal{A}$ makes a decryption query on a ciphertext c, $\mathcal{B}$ outputs either $\perp$ if c fails the consistency check or the corresponding message. Thus answers to decryption queries do not leak any information to $\mathcal{A}$ about either x_v or x_w. However for the challenge ciphertext, we have that $t^* + s^* x_v + x_w = 0$ and this fact is known to the adversary. There are only p possible (x_v, x_w) pairs satisfying this equation and each pair is equally likely. Thus, the probability that $t + sx_v + x_w = 0$ is at most $1/p$.

Guess. $\mathcal{A}$ outputs a bit δ'. If $\delta = \delta'$, then $\mathcal{B}$ outputs $\gamma = 1$, which means that $\mathcal{B}$'s guess is $W = \mathsf{H}(g^{ab})$. If $\delta \neq \delta'$, then $\mathcal{B}$ outputs $\gamma = 0$, which means that $\mathcal{B}$'s guess is that W is a random string.

From the above we see that unless $\mathcal{B}$ aborts, $\mathcal{B}$ simulates the view of $\mathcal{A}$ perfectly, as in the original PKE security experiment.

If $\mathcal{A}$ wins, then $\mathcal{B}$ also wins. Therefore, we have that, for all $k \geqslant 0$,

$$\mathbf{Adv}_{\mathcal{B}}^{\mathsf{HDH}}(k) \geqslant \mathbf{Adv}_{\mathcal{A},\mathsf{PKE}}^{\mathsf{IND\text{-}CCA2}}(k) - \mathbf{Adv}_{\mathcal{A}}^{\mathsf{CR}}(k) - q/p.$$

The total running time of $\mathcal{B}$ is $t_{\mathcal{B}} \leqslant t_{\mathcal{A}} + \mathrm{O}(t_{\mathbb{G}})$, where $t_{\mathcal{A}}$ is the running time of $\mathcal{A}$ and $t_{\mathbb{G}}$ is the time to perform a basic operation in $\mathbb{G}$. □

3. Publicly verifiable ciphertexts in general KEMs

We now adapt the generalized encryption syntax to the setting of key encapsulation mechanisms and explore the public verifiability of general KEMs.

3.1. Definition: *General KEM*

A *general key encapsulation* mechanism (GKEM) is a tuple $\mathsf{GKEM} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$ of four algorithms that are defined almost identically to general encryption schemes (cf. Section 2.1). The difference to encryption schemes' PG and KG algorithms is that message space MsgSp is replaced by a key space KeySp. In addition, the modification to Encap and Decap (when compared to Enc and Dec, respectively) is that Encap outputs a ciphertext C and a session key $K \in \mathsf{KeySp}$, while Decap outputs either a key K, or $\perp$.

Correctness of GKEM and its security definition through advantage function $\mathbf{Adv}_{\mathcal{A},\mathsf{GKEM}}^{\mathsf{IND\text{-}xxx}}(k)$, $\mathsf{xxx} \in \{\mathsf{CPA}, \mathsf{CCA2}\}$, in the indistinguishability experiments are also defined analogously to the case of general encryption and are shown in the right side of Fig. 1.

3.2. General KEMs with publicly verifiable ciphertexts

Definition 3 (Publicly verifiable GKEM). A general key encapsulation mechanism $\mathsf{GKEM} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$ is said to be *publicly verifiable* with respect to auxiliary algorithms Ver and Decap' if $\mathsf{Decap} = \mathsf{Decap}' \circ \mathsf{Ver}$, and Ver and Decap' satisfy the same criteria as Ver and Dec' in Definition 1, with the proviso that Decap' outputs a key $K \in \mathsf{KeySp}$, not a message $M \in \mathsf{MsgSp}$.

Theorem 4 (whose proof is identical to that of Theorem 1 and is thus omitted) shows that any publicly verifiable IND-CCA2-secure GKEM scheme will remain at least IND-CPA-secure if the verification algorithm Ver is run by an honest-but-curious gateway. To account for a non-trivial verification process that may modify the ciphertext, we again apply post-processing to the output of the encapsulation algorithm (cf. discussion in Section 2.2).

Theorem 4. *Let* $\mathsf{GKEM} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$ *be an* IND-CCA2*-secure general KEM that is publicly verifiable with respect to auxiliary algorithms* Ver *and* Decap'. *Let* $\mathsf{Encap}' := \mathsf{Ver} \circ \mathsf{Encap}$ *and let* $\mathsf{GKEM}' := (\mathsf{PG}, \mathsf{KG}, \mathsf{Encap}', \mathsf{Decap}')$. *Then for every* IND-CPA *adversary* $\mathcal{A}$ *against* GKEM', *there exists an* IND-CCA2 *adversary* $\mathcal{B}$ *against* GKEM *such that, for all* $k \geqslant 0$, $\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathsf{GKEM}'}(k) \leqslant \mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{B},\mathsf{GKEM}}(k)$, *where* $\mathcal{B}$ *has* (*asymptotically*) *the same running time as* $\mathcal{A}$.

3.3. Constructions of publicly verifiable KEMs

We now present some examples of KEMs with publicly verifiable ciphertexts. First, we discuss the publicly verifiable construction of an identity-based KEM that we obtain immediately from the IND-CCA2-secure IB-KEM proposed by Kiltz and Galindo [24]. Parameters $\mathsf{par}' = (\mathbb{G}_1, \mathbb{G}_T, p, g, e, \mathsf{H})$ chosen by parameter generation algorithm $\mathsf{PG}(1^k)$ are such that $\mathbb{G}_1 = \langle g \rangle$ is a multiplicative cyclic group of prime order $p > 2^{2k}$, $\mathbb{G}_T$ is a multiplicative cyclic group of the same order, $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a non-degenerate bilinear map, and $\mathsf{H} : \mathsf{IDSp} = \{0,1\}^{\ell(k)} \to \mathbb{G}_1$ is Waters' hash function such that $\ell(k)$ is a polynomial in k. We also use a target collision resistant function $\mathsf{TCR} : \mathbb{G}_1 \to \mathbb{Z}_p$. Figure 6 details the scheme.

Note that by setting $\mathsf{KEM.Decap} = \mathsf{KEM.Decap}' \circ \mathsf{KEM.Ver}$ we immediately obtain the original Kiltz–Galindo IB-KEM [24]. It is easy to see that algorithm KEM.Ver is publicly verifiable as per Definition 3. Further, Kiltz and Galindo noted that ignoring all operations associated to the identity in their IB-KEM yields a simplified version of the IND-CCA2-secure public-key schemes from [8,21]. Therefore, by removing computations related to the ciphertext component c_2 and the key generation algorithm KG from Kiltz–Galindo's IB-KEM, Theorem 4 implies that we immediately obtain IND-CPA-secure publicly verifiable constructions of a public-key KEM and a tag-based KEM.

KEM.PG(1^k):

(1) Generate $\mathsf{par}' = (\mathbb{G}_1, \mathbb{G}_T, p, g, e, \mathsf{H})$
(2) $\alpha \xleftarrow{\$} \mathbb{G}_1$, $\mathsf{msk} \leftarrow \alpha$
(3) $u, v \xleftarrow{\$} \mathbb{G}_1$, $z \leftarrow e(g, \alpha)$
(4) $\mathsf{pk} \leftarrow (u, v, z)$
(5) $\mathsf{par} \leftarrow (\mathsf{par}', \mathsf{pk})$
(6) Return $(\mathsf{par}, \mathsf{msk})$

KEM.Encap(par, id):

(1) $(\mathsf{par}', \mathsf{pk}) \leftarrow \mathsf{par}$
(2) Parse par' and pk
(3) $r \xleftarrow{\$} \mathbb{Z}_p^*$, $c_1 \leftarrow g^r$
(4) $t \leftarrow \mathsf{TCR}(c_1)$
(5) $c_2 \leftarrow \mathsf{H}(\mathsf{id})^r$
(6) $c_3 \leftarrow (u^t v)^r$
(7) $K \leftarrow z^r \in \mathbb{G}_T$
(8) $C \leftarrow (c_1, c_2, c_3) \in \mathbb{G}_1^3$
(9) Return (C, K)

KEM.KG(par, msk, id):

(1) Parse $(\mathsf{par}', \mathsf{pk}) \leftarrow \mathsf{par}$ and par'
(2) $s \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{dk}[\mathsf{id}] \leftarrow (\alpha \cdot \mathsf{H}(\mathsf{id})^s, g^s)$
(3) Return dk[id]

KEM.Ver(par, pk, id, C):

(1) $(\mathsf{par}', \mathsf{pk}) \leftarrow \mathsf{par}$
(2) Parse par' and pk
(3) $(c_1, c_2, c_3) \leftarrow C$, $t \leftarrow \mathsf{TCR}(c_1)$
(4) If $e(g, c_3) \neq e(c_1, u^t v)$ or $e(g, c_2) \neq e(c_1, \mathsf{H}(\mathsf{id}))$, then return $\perp$
(5) Return $C' = (c_1, c_2)$

KEM.Decap′(par, id, dk[id], C'):

(1) $(\mathsf{par}', \mathsf{pk}) \leftarrow \mathsf{par}$
(2) Parse par'
(3) $(c_1, c_2) \leftarrow C'$, $(d_1, d_2) \leftarrow \mathsf{dk}[\mathsf{id}]$
(4) $K \leftarrow e(c_1, d_1)/e(c_2, d_2)$
(5) Return K

Fig. 6. Kiltz–Galindo IB-KEM with publicly verifiable ciphertexts.

4. Publicly verifiable ciphertexts in hybrid encryption

For its simplicity and flexibility, the KEM/DEM approach [12,37] to construct PKE schemes has attracted much attention and is used in several encryption standards [20,31,38]. It has been shown that if both the KEM and the DEM are secure against chosen-ciphertext attacks, then so is the resulting hybrid encryption scheme [12]. Herranz et al. [19] studied a variety of other necessary and sufficient conditions for KEMs and DEMs for the security of the hybrid construction. They showed that, in order to obtain IND-CCA2 security of the hybrid scheme, the KEM must be IND-CCA2-secure while the security requirement on the DEM can be relaxed from IND-CCA2 to IND-OTCCA that prevents *one-time* (adaptive) chosen-ciphertext attacks.

Therefore, when dealing with public verifiability of hybrid schemes, we must take into account the existence of consistency checks in the decapsulation of DEM, in addition to checks on the KEM part. Since DEM consistency checks are typically performed using the decapsulated key, hybrid schemes cannot usually provide the full form of public verification as specified in Definition 1. We show, however, that these schemes can offer a somewhat relaxed property, where only the KEM part is publicly verifiable, so successful public consistency check of the KEM part is a necessary but not a sufficient condition for the overall success of decryption. In the context of gateway-assisted IND-CCA2/CPA conversion, this property effectively allows outsourcing the consistency check of the KEM part to the gateway. In this

HGE.PG(1^k):	HGE.KG(par, msk, id):
(1) (par, msk) $\xleftarrow{\$}$ KEM.PG(1^k) (2) Return (par, msk)	(1) (ek, dk) $\xleftarrow{\$}$ KEM.KG(par, msk, id) (2) Return (ek, dk)
HGE.Enc(par, ek, M, t):	HGE.Dec(par, ek, dk, C, t):
(1) $(C_1, K) \leftarrow$ KEM.Encap(par, ek, t) (2) $C_2 \leftarrow$ DEM.Enc(K, M) (3) Return $C = (C_1, C_2)$	(1) $(C_1, C_2) \leftarrow C$ (2) $K \leftarrow$ KEM.Decap(par, ek, dk, C_1, t) (3) If $K = \perp$ then return $\perp$ (4) $M \leftarrow$ DEM.Dec(K, C_2) (5) Return M (possibly as $\perp$)

Fig. 7. Hybrid general encryption scheme HGE.

way, clients would only need to perform private consistency checks for the DEM part: in practice, the verification costs for DEMs are often much smaller than the verification costs for KEMs.

4.1. Definition: Hybrid general encryption

Let GKEM $=$ (PG, KG, Encap, Decap) be a general KEM scheme (as defined in Section 3.1) and let DEM $=$ (Enc, Dec) be a one-time symmetric key encryption scheme [19]. The two schemes are assumed to be compatible, so session keys output by KEM are appropriate for DEM.

A *hybrid general encryption* (HGE) scheme is a tuple HGE $=$ (PG, KG, Enc, Dec) of four algorithms as defined in Fig. 7.

4.1.1. Correctness

A hybrid general encryption scheme HGE $=$ (PG, KG, Enc, Dec) is *correct* if, for all (par, msk) output by HGE.PG, all plaintexts M, all identities id $\in$ IDSp, all (ek, dk) output by HGE.KG(par, msk, id), and all tags $t \in$ TagSp, we have HGE.Dec(par, ek, dk, HGE.Enc(par, ek, M, t), t) $= M$ with probability 1, where the probability is taken over the coins of HGE.Enc.

4.2. Hybrid general encryption with publicly verifiable ciphertexts

When defining public verifiability of HGE schemes with respect to Ver and Dec′, we can reuse most of Definition 1 for general encryption. Note that message M output by the lightweight decryption algorithm Dec′ could also be an error symbol $\perp$. However, as previously mentioned, in general, HGE do not satisfy condition 3(b) of Definition 1 since, for hybrid schemes, failure of the original decryption procedure HGE.Dec may not necessarily imply failure of the verification algorithm Ver′. For this reason we define the following relaxed notion:

Definition 4 (Partial public verification). Let HGE $=$ (PG, KG, Enc, Dec) be a hybrid general encryption scheme of the form given in Fig. 7. We say that HGE is

partially publicly verifiable with respect to auxiliary algorithms Ver and Dec′ if it satisfies conditions 1, 2 and 3(b) of Definition 1, as well as the following condition:

3(a)′ $(\mathsf{Ver}(\mathsf{par}, \mathsf{ek}, C, t) = \perp \vee \mathsf{DEM.Dec}(K, C_2) = \perp) \Leftrightarrow \mathsf{Dec}(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C, t) = \perp$ for all C, where $C = (C_1, C_2)$ and $K = \mathsf{KEM.Decap}(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C_1, t)$.

Condition 3(a)′ states that successful public verification by itself is not necessarily sufficient for the decryption algorithm to successfully decrypt. More precisely, if Ver succeeds, then the only reason that HGE.Dec fails is because of a failure in DEM.Dec.

The following theorem shows that, if the underlying general KEM is publicly verifiable, then the hybrid general encryption scheme is partially publicly verifiable and that by outsourcing verification of the KEM part the hybrid scheme remains at least IND-CPA-secure.

Theorem 5. *Let* $\mathsf{GKEM} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$ *be an* IND-CCA2*-secure general key encapsulation mechanism that is publicly verifiable with respect to* GKEM.Ver *and* GKEM.Decap′, $\mathsf{DEM} = (\mathsf{Enc}, \mathsf{Dec})$ *be an* IND-OTCCA*-secure data encapsulation mechanism, and* $\mathsf{HGE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *be the resulting hybrid general encryption scheme. Then the following hold*:

(1) HGE *is partially publicly verifiable with respect to* HGE.Ver *and* HGE.Dec′.

(2) *Let* $\mathsf{HGE}' := (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}', \mathsf{Dec}')$ *with* $\mathsf{HGE}'.\mathsf{Enc}' = \mathsf{HGE.Ver} \circ \mathsf{HGE.Enc}$ *and* $\mathsf{HGE}'.\mathsf{Dec}' = \mathsf{HGE.Dec}'$. *For any* IND-CPA *adversary* $\mathcal{A}$ *against* HGE′, *there exists an* IND-CPA *adversary* $\mathcal{B}_1$ *against* GKEM′ *and an* IND-OTCCA *adversary* $\mathcal{B}_2$ *against* DEM *such that*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathsf{HGE}'}(k) \leqslant \mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{B}_1,\mathsf{GKEM}'}(k) + \mathbf{Adv}^{\mathsf{IND\text{-}OTCCA}}_{\mathcal{B}_2,\mathsf{DEM}}(k) \quad \forall k \geqslant 0,$$

and such that $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *have* (*asymptotically*) *the same running time as* $\mathcal{A}$.

Proof.

Statement 1. The first statement of the theorem is proven as follows: if GKEM as used in the HGE construction is publicly verifiable (in the sense of Definition 3) then there exist two algorithms GKEM.Ver and GKEM.Decap′ such that $\mathsf{GKEM.Decap} = \mathsf{GKEM.Decap}' \circ \mathsf{GKEM.Ver}$.

We construct now two algorithms HGE.Ver and HGE.Dec′ as follows:

HGE.Ver(par, ek, C, t): Given public parameters par, the encryption key ek, a ciphertext C and a tag t, this algorithm first parses C into C_1 and C_2 and runs GKEM.Ver(par, ek, C_1, t). If GKEM.Ver($\cdot$) outputs $\perp$ the algorithm HGE.Ver also outputs $C' = \perp$. Otherwise the output of GKEM.Ver($\cdot$) is a new (transformed) ciphertext C'_1 and in this case the algorithm HGE.Ver outputs a ciphertext $C' = (C'_1, C_2)$.

$\mathsf{HGE.Dec}'(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C', t)$: The algorithm parses C' as (C'_1, C_2) and obtains $K \leftarrow \mathsf{GKEM.Decap}'(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C'_1, t)$. If $K = \perp$ then it outputs $\perp$. Otherwise, it runs $\mathsf{DEM.Dec}(K, C_2)$ and outputs its result (which could also be $\perp$).

From the above, it is easy to see that $\mathsf{HGE.Dec} = \mathsf{HGE.Dec}' \circ \mathsf{HGE.Ver}$. That is, $\mathsf{HGE.Dec}(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C, t)$ has the same input/output behavior as the following sequence of steps:

(1) $(C_1, C_2) \leftarrow C$
(2) $C' \leftarrow \mathsf{HGE.Ver}(\mathsf{par}, \mathsf{ek}, C, t)$
(3) If $C' = \perp$ then return $\perp$
(4) $M \leftarrow \mathsf{HGE.Dec}'(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C', t)$
(5) Return M (possibly as $\perp$)

This construction of HGE.Dec implies that HGE is at least inconclusively publicly verifiable with respect to HGE.Ver and HGE.Dec′. Now observe that if either HGE.Ver or HGE.Dec′ fails then so does HGE.Dec. By construction, $\mathsf{DEM.Dec}(K, C_2) = \perp$ leads to the failure of HGE.Dec′. Hence,

$$\mathsf{HGE.Ver}(\mathsf{par}, \mathsf{ek}, C, t) = \perp \vee \mathsf{DEM.Dec}(K, C_2) = \perp$$
$$\Rightarrow \quad \mathsf{HGE.Dec}(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C, t) = \perp. \tag{1}$$

As for the opposite implication observe that if HGE.Dec outputs $\perp$ then either $\mathsf{HGE.Ver}(\mathsf{par}, \mathsf{ek}, C, t) = \perp$ or $\mathsf{HGE.Dec}'(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C', t) = \perp$. Note that by construction, HGE.Dec′ fails if $\mathsf{GKEM.Decap}'(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C'_1, t) = \perp$ or $\mathsf{DEM.Dec}(K, C_2) = \perp$. Since $\mathsf{GKEM.Decap} = \mathsf{GKEM.Decap}' \circ \mathsf{GKEM.Ver}$ and GKEM.Ver is conclusive we have $\mathsf{GKEM.Decap}'(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C'_1, t) = \perp$ if and only if $C' = \perp$. Since $C' = \perp$ is equivalent to the failure of HGE.Ver we have

$$\mathsf{HGE.Dec}(\mathsf{par}, \mathsf{ek}, \mathsf{dk}, C, t) = \perp$$
$$\Rightarrow \quad \mathsf{HGE.Ver}(\mathsf{par}, \mathsf{ek}, C, t) = \perp \vee \mathsf{DEM.Dec}(K, C_2) = \perp. \tag{2}$$

Combining (1) and (2) shows that HGE.Ver is partially publicly verifiable as per Definition 4.

Statement 2. To prove the second statement, we first need to show that the hybrid general encryption scheme $\mathsf{HGE}' := (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}', \mathsf{Dec}')$ is IND-CPA-secure. Note that HGE′ is a general encryption scheme that is obtained through a hybrid construction of $\mathsf{GKEM}' := (\mathsf{PG}, \mathsf{KG}, \mathsf{Encap}', \mathsf{Decap}')$ and $\mathsf{DEM} = (\mathsf{Enc}, \mathsf{Dec})$, where $\mathsf{GKEM}'.\mathsf{Encap}' := \mathsf{GKEM.Ver} \circ \mathsf{GKEM.Encap}$ and $\mathsf{GKEM}'.\mathsf{Decap}' = \mathsf{GKEM.Decap}'$, as defined in Theorem 4, which also says that GKEM′ is IND-CPA-secure. This implies that HGE′ is obtained through combination of an IND-CPA-secure KEM and an IND-OTCCA secure DEM. The result of Herranz et al. [19], who showed that this combination achieves at least IND-CPA security, helps us to immediately conclude the proof of the second statement. □

4.3. Hybrid encryption schemes with publicly verifiable ciphertexts

Herranz et al. [19] showed that if an IND-CCA2-secure KEM is combined with an IND-OTCCA-secure DEM then the resulting hybrid encryption scheme is also IND-CCA2-secure. As shown by Cramer and Shoup [12], one can easily construct an IND-OTCCA-secure DEM by adding a one-time MAC to a one-time-secure DEM such as the one-time pad. Moreover, Theorem 5 states that if the underlying KEM is publicly verifiable then the resulting hybrid encryption scheme is publicly verifiable as well. We can thus immediately obtain from these two building blocks a range of publicly verifiable constructions of hybrid encryption schemes with partial public verifiability; for instance, we can apply publicly verifiable KEM constructions from Section 3.3.

In the case of the tag-based KEM/DEM approach, Abe et al. [3] showed that IND-CCA2-secure hybrid encryption can be obtained by combining an IND-CCA2-secure tag-based KEM with a one-time secure DEM. They also provide constructions of IND-CCA2-secure tag-based KEMs that they obtain generically from IND-CCA2-secure public-key KEMs and one-time MACs. Our publicly verifiable public-key-based KEM constructions from Section 3.3 can be used to instantiate their tag-based KEMs, resulting in further publicly verifiable hybrid encryption schemes.

5. Conclusion

We studied the notion of public verifiability for encryption schemes, KEMs and hybrid KEM/DEM encryption. Public verifiability allows any party to check the well-formedness of a ciphertext. If a semi-trusted intermediate party, such as a gateway, performs the verification, then the final recipient need not perform the verification and can simply decrypt. If the gateway does its job properly, then the overall scheme has IND-CCA2 security. Regardless of the gateway's behaviour, the recipient has the guarantee of IND-CPA security; moreover, the recipient can always choose to do the full verification to obtain full IND-CCA2 security. Compared with other outsourcing techniques, a scheme with public verifiability is compelling because it does not require the recipient to disclose its private keys to the gateway.

Our formal definitions for publicly verifiable schemes adopt and extend the generalized syntax of Abdalla et al. [1], to cover public-key based, identity-based, and tag-based settings. We demonstrate that the well-known CHK and NIZK-based transforms generally offer at least some public verification. We propose several direct constructions of publicly verifiable PKE schemes and KEMs. For hybrid KEM/DEM schemes, we show that, although fully conclusive public verification is not achievable, a relaxed notion of partial public verifiability of KEM ciphertexts can be obtained: this IND-CCA2/CPA filter still offers performance gains that can be useful for applications involving outsourcing ciphertext verification to an honest-but-curious gateway without losing confidentiality.

Acknowledgments

This research was supported by the Australian Technology Network (ATN) and German Academic Exchange Service (DAAD) Joint Research Cooperation Scheme. Juan González Nieto, Jothi Rangasamy and Douglas Stebila were further supported by the Australia–India Strategic Research Fund project TA020002. Mark Manulis acknowledges support through German Research Foundation (DFG) via grant MA 4957. This work was also supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE and by the Hessian LOEWE excellence initiative within CASED. The majority of this work was performed while Mark Manulis and Bertram Poettering were at TU Darmstadt and Jothi Rangasamy was at QUT.

References

[1] M. Abdalla, M. Bellare and G. Neven, Robust encryption, in: *TCC 2010*, D. Micciancio, ed., LNCS, Vol. 5978, Springer, 2010, pp. 480–497.

[2] M. Abdalla, M. Bellare and P. Rogaway, The oracle Diffie–Hellman assumptions and an analysis of DHIES, in: *CT-RSA 2001*, LNCS, Vol. 2020, Springer, 2001, pp. 143–158.

[3] M. Abe, R. Gennaro and K. Kurosawa, Tag-KEM/DEM: A new framework for hybrid encryption, *Journal of Cryptology* **21**(1) (2008), 97–130.

[4] M. Abe, E. Kiltz and T. Okamoto, Chosen ciphertext security with optimal ciphertext overhead, in: *ASIACRYPT 2008*, LNCS, Vol. 5350, Springer, 2008, pp. 355–371.

[5] D. Boneh, X. Boyen and S. Halevi, Chosen ciphertext secure public key threshold encryption without random oracles, in: *CT-RSA 2006*, LNCS, Vol. 3860, Springer, 2006, pp. 226–243.

[6] D. Boneh, R. Canetti, S. Halevi and J. Katz, Chosen-ciphertext security from identity-based encryption, *SIAM J. Comput.* **36**(5) (2007), 1301–1328.

[7] D. Boneh and J. Katz, Improved efficiency for CCA-secure cryptosystems built using identity-based encryption, in: *CT-RSA 2005*, LNCS, Vol. 3376, Springer, 2005, pp. 87–103.

[8] X. Boyen, Q. Mei and B. Waters, Direct chosen ciphertext security from identity-based techniques, in: *ACM CCS 2005*, V. Atluri, C. Meadows and A. Juels, eds, ACM, 2005, pp. 320–329.

[9] R. Canetti, S. Halevi and J. Katz, Chosen-ciphertext security from identity-based encryption, in: *EUROCRYPT 2004*, C. Cachin and J. Camenisch, eds, LNCS, Vol. 3027, Springer, 2004, pp. 207–222.

[10] D. Cash, E. Kiltz and V. Shoup, The twin Diffie–Hellman problem and applications, in: *Proc. EUROCRYPT 2008*, N. Smart, ed., LNCS, Vol. 4965, Springer, 2008, pp. 127–145; full version available at http://eprint.iacr.org/2008/067 and published as [11].

[11] D. Cash, E. Kiltz and V. Shoup, The twin Diffie–Hellman problem and applications, *J. Cryptology* **22**(4) (2009), 470–504; extended abstract published as [10].

[12] R. Cramer and V. Shoup, Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack, *SIAM J. Computing* **33**(1) (2003), 167–226.

[13] D. Dolev, C. Dwork and M. Naor, Non-malleable cryptography (extended abstract), in: *ACM STOC 1991*, ACM, 1991, pp. 542–552.

[14] E. Elkind and A. Sahai, A unified methodology for constructing public-key encryption schemes secure against adaptive chosen-ciphertext attack, Cryptology ePrint Archive, Report 2002/042, 2002, available at: http://eprint.iacr.org/2002/042.

[15] O. Goldreich and L.A. Levin, A hard-core predicate for all one-way functions, in: *Proc. STOC 1989*, D.S. Johnson, ed., ACM, 1989, pp. 25–32.

[16] J.M. González Nieto, M. Manulis, B. Poettering, J. Rangasamy and D. Stebila, Publicly verifiable ciphertexts (extended abstract), in: *Proc. Security and Cryptography for Networks (SCN) 2012*, LNCS, Vol. 7485, Springer, 2012, pp. 393–410.

[17] C.P.L. Gouvêa and J. López, Software implementation of pairing-based cryptography on sensor networks using the MSP430 microcontroller, in: *INDOCRYPT 2009*, LNCS, Vol. 5922, Springer, 2009, pp. 248–262.

[18] G. Hanaoka and K. Kurosawa, Efficient chosen ciphertext secure public key encryption under the computational Diffie–Hellman assumption, in: *ASIACRYPT 2008*, LNCS, Vol. 5350, Springer, 2009, pp. 308–325.

[19] J. Herranz, D. Hofheinz and E. Kiltz, KEM/DEM: Necessary and sufficient conditions for secure hybrid encryption, Cryptology ePrint Archive, Report 2006/265, 2006, available at: http://eprint.iacr.org/2006/265.

[20] H. Imai and A. Yamagishi, CRYPTREC project – Cryptographic evaluation project for the Japanese electronic government, in: *ASIACRYPT 2000*, LNCS, Vol. 1976, Springer, 2000, pp. 399–400.

[21] E. Kiltz, Chosen-ciphertext security from tag-based encryption, in: *TCC 2006*, LNCS, Vol. 3876, Springer, 2006, pp. 581–600.

[22] E. Kiltz, Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie–Hellman, in: *PKC 2007*, LNCS, Vol. 4450, Springer, 2007, pp. 282–297.

[23] E. Kiltz, Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie–Hellman, Cryptology ePrint Archive, Report 2007/36, 2007, available at: http://eprint.iacr.org/2007/036.pdf; preliminary version published as [22].

[24] E. Kiltz and D. Galindo, Direct chosen-ciphertext secure identity-based key encapsulation without random oracles, in: *ACISP 2006*, LNCS, Vol. 4058, Springer, 2006, pp. 336–347; full version published as [25].

[25] E. Kiltz and D. Galindo, Direct chosen-ciphertext secure identity-based key encapsulation without random oracles, *Theoretical Computer Science* **410**(47–49) (2009), 5093–5111; extended abstract published as [24].

[26] K. Kurosawa and Y. Desmedt, A new paradigm of hybrid encryption scheme, in: *CRYPTO 2004*, LNCS, Vol. 3152, Springer, 2004, pp. 426–442.

[27] J. Lai, R.H. Deng, S. Liu and W. Kou, Efficient CCA-secure PKE from identity-based techniques, in: *CT-RSA 2010*, LNCS, Vol. 5985, Springer, 2010, pp. 132–147.

[28] B. Libert and M. Yung, Adaptively secure non-interactive threshold cryptosystems, in: *ICALP 2011*, Part II, LNCS, Vol. 6756, Springer, 2011, pp. 588–600.

[29] C.H. Lim and P.J. Lee, Another method for attaining security against adaptively chosen ciphertext attacks, in: *CRYPTO 1993*, LNCS, Vol. 773, Springer, 1993, pp. 420–434.

[30] M. Naor and M. Yung, Public-key cryptosystems provably secure against chosen ciphertext attacks, *ACM STOC 1990*, ACM, 1990, pp. 427–437.

[31] NESSIE, Final report of European project IST-1999-12324: New European Schemes for Signatures, Integrity, and Encryption, April 2004, available at: https://www.cosic.esat.kuleuven.be/nessie/.

[32] G. Persiano, About the existence of trapdoors in cryptosystems, Manuscript, available at: http://libeccio.dia.unisa.it/Papers/Trapdoor/Trapdoor.pdf.

[33] D.H. Phan and D. Pointcheval, Chosen-ciphertext security without redundancy, in: *ASIACRYPT 2003*, LNCS, Vol. 2894, Springer, 2003, pp. 1–18.

[34] D.H. Phan and D. Pointcheval, OAEP 3-round: A generic and secure asymmetric encryption padding, in: *ASIACRYPT 2004*, LNCS, Vol. 3329, Springer, 2004, pp. 63–78.

[35] J. Rompel, One-way functions are necessary and sufficient for secure signatures, in: *STOC 1990*, ACM, 1990, pp. 387–394.

[36] A. Sahai, Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security, *FOCS 1999*, IEEE, 1999, pp. 543–553.

[37] V. Shoup, A proposal for an ISO standard for public key encryption (version 2.1), Manuscript, 2001, available at: http://shoup.net/papers.

[38] V. Shoup, ISO 18033-2: An emerging standard for public-key encryption, Final Committee Draft, December 2004, available at: http://shoup.net/iso/std6.pdf.
[39] V. Shoup and R. Gennaro, Securing threshold cryptosystems against chosen ciphertext attack, in: *EUROCRYPT 1998*, LNCS, Vol. 1403, Springer, 1998, pp. 1–16.
[40] S. Stelle, M. Manulis and M. Hollick, Topology-driven secure initialization in wireless sensor networks: A tool-assisted approach, in: *IEEE ARES 2012*, IEEE, 2012, pp. 28–37.
[41] H. Wang and Q. Li, Efficient implementation of public key cryptosystems on mote sensors, in: *ICICS 2006*, LNCS, Vol. 4307, Springer, 2006, pp. 519–528.
[42] D. Wikström, Simplified submission of inputs to protocols, in: *SCN 2008*, LNCS, Vol. 5229, Springer, 2008, pp. 293–308.