Public-Key Encryption with Non-interactive Opening: New Constructions and Stronger Definitions

David Galindo[1], Benoît Libert[2], Marc Fischlin[3], Georg Fuchsbauer[4], Anja Lehmann[3], Mark Manulis[3], and Dominique Schröder[3]

[1] University of Luxembourg
david.galindo@uni.lu
[2] Université catholique de Louvain, Crypto Group, Belgium
[3] TU Darmstadt & CASED, Germany
[4] École normale supérieure, LIENS - CNRS - INRIA, Paris, France

Abstract. Public-key encryption schemes with non-interactive opening (PKENO) allow a receiver to non-interactively convince third parties that a ciphertext decrypts to a given plaintext or, alternatively, that such a ciphertext is invalid. Two practical generic constructions for PKENO have been proposed so far, starting from either identity-based encryption or public-key encryption with witness-recovering decryption (PKEWR). We show that the known transformation from PKEWR to PKENO fails to provide chosen-ciphertext security; only the transformation from identity-based encryption remains thus valid. Next, we prove that PKENO can alternatively be built out of robust non-interactive threshold public-key cryptosystems, a primitive that differs from identity-based encryption. Using the new transformation, we construct two efficient PKENO schemes: one based on the Decisional Diffie-Hellman assumption (in the Random-Oracle Model) and one based on the Decisional Linear assumption (in the standard model). Last but not least, we propose new applications of PKENO in protocol design. Motivated by these applications, we reconsider proof soundness for PKENO and put forward new definitions that are stronger than those considered so far. We give a taxonomy of all definitions and demonstrate them to be satisfiable.

Keywords: public-key encryption, non-interactive proofs, security definitions, constructions.

1 Introduction

Public-key encryption allows a receiver Bob to generate a pair of a private and a public key (sk_B, pk_B) such that anyone can encrypt messages under pk_B which can only be decrypted by Bob who knows sk_B. The primitive *public-key encryption with non-interactive opening* (PKENO), introduced by Damgård et al. [DHKT08], allows Bob to prove to a verifier Alice that a given ciphertext C decrypts to a certain message. By using PKENO, Bob can do so convincingly

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 333–350, 2010.
© Springer-Verlag Berlin Heidelberg 2010

and without further interaction, neither with Alice nor with the original sender of the ciphertext. More precisely, Bob runs a proving algorithm Prove on inputs his secret key sk_B and the intended ciphertext C, thereby generating a proof π. On the other hand, Alice runs a verification algorithm Ver on inputs Bob's public key pk_B, ciphertext C, a plaintext m, and an opening proof π. The soundness property guarantees that the verification algorithm outputs 1 if C was indeed an encryption of m, and 0 otherwise. An interesting feature of PKENO is that Bob can also convince Alice of the fact that a given ciphertext C is *invalid*, i.e., it is rejected by the decryption algorithm. PKENO turns out to be a useful primitive for protocol design. In addition to the use of PKENO in multiparty computation protocols, as highlighted in [DT07, DHKT08], we identify further applications, which we introduce below.

Secure Message Transmission with PKENO. One of the classical ways to realize secure message transmission in a public-key setting is to let the sender encrypt the message and then sign the ciphertext, i.e., the so-called *encrypt-then-sign* paradigm [ADR02] in which the transmitted ciphertext also includes a signature $\mathsf{Sign}(sk_s, \mathsf{Enc}(pk_r, pk_s||m)||pk_r)$, with sk_s being the signing key of the sender and pk_r the encryption key of the receiver. If the sender uses a standard PKE scheme, the receiver is in general not able to provide a non-repudiable proof for the origin of the received message m. To do so, the receiver should convincingly open the encryption $\mathsf{Enc}(pk_r, pk_s||m)$, which he cannot do, unless he is willing to expose his decryption key sk_r. Replacing PKE with PKENO allows the receiver to prove the origin for the decrypted *message*, and thus authenticated encryption with non-repudiation is achieved.

Group Signatures. The most common way to achieve anonymity in group signatures [CvH91] is the following: a group member first encrypts his membership certificate under the opener's public key while adding a non-interactive proof of validity of the encrypted data. The opening authority is then able to identify the signer by merely decrypting the ciphertext.

In the model of dynamic group signatures given by Bellare et al. [BSZ05], the opening authority is required to give a *proof* that it traced the correct user. Using PKENO rather than plain encryption enables the opener to do so in a simple manner. In the game modeling the anonymity of signatures in [BSZ05], an adversary is given an opening oracle that opens adversarially-chosen signatures and outputs proofs of correct opening. The security of the employed PKENO scheme (together with simulation-sound zero knowledge of the proof of well-formedness) ensures that an adversary cannot distinguish signatures from distinct users.

1.1 Our Contributions

Difficulty of Building PKENO. Damgård et al. [DHKT08] showed that a PKENO can be built out of Identity-Based Encryption (IBE). Although IBE can now be realized under a variety of assumptions and without bilinear maps (see [BGH07, GPV08, AB09, CHK09, Pei09] for instance), it remains a very specialized and powerful cryptographic primitive. Towards narrowing the gap between sufficient and necessary conditions for PKENO, it is interesting to see whether it can be obtained without resorting to all the functionalities provided by IBE (e.g. non-interactive user key derivation). In [DHKT08], the authors mentioned that PKENO can also be based upon a seemingly weaker primitive, called *public-key encryption with witness-recovering decryption* (PKEWR) [PW08]. In a PKEWR scheme, the receiver Bob is able to recover the random coins r used to encrypt a ciphertext C. Damgård *et al.* proposed to use the coins r as the proof, and verification proceeds by re-encrypting $C' = \mathsf{Enc}(pk_B, m; r)$ and checking whether $C = C'$. However, this approach can only be guaranteed to be sound for *valid* ciphertexts, i.e., ciphertexts that have been output by the encryption algorithm. As a consequence, for invalid ciphertexts "the coins used to construct C" might not be well defined. Indeed, we show in Section 4.1 how the (apparently) straightforward construction of PKENO out of PKEWR fails to provide security in the sense of [DHKT08]. This then motivates the quest for both new generic and concrete constructions for PKENO.

Non-Interactive Threshold Cryptosystems Imply PKENO. Somewhat surprisingly, we show that starting from a robust *non-interactive threshold cryptosystem* (TPKC), we can construct a PKENO scheme in a generic way. We only ask the threshold cryptosystem to satisfy some appropriate notion of decryption consistency. We emphasize that, although this notion is stronger than the one initially formalized by Shoup and Gennaro [SG98], it remains fairly mild in that most known robust threshold cryptosystems satisfy it.

Threshold cryptosystems distribute the ability to decrypt among several parties. The private decryption key is shared among n servers such that at least t servers are needed for decryption. If the *combiner* wishes to decrypt some ciphertext C, it sends C to the decryption servers. After receiving at least t partial decryption shares from the servers, the combiner is able to reconstruct the plaintext from these shares. A *robust* TPKC [SG98, BBH06] provides the additional property that, whenever the decryption of valid ciphertexts fails, the combiner can sieve out bad decryption shares and reveal the identity of the server having sent an invalid partial decryption. We show an efficient transformation from robust TPKC to PKENO. When applied to the schemes in [SG98, AT09], the conversion provides new practical PKENO schemes based on the Decisional Diffie-Hellman (in the random oracle model) and the Decisional Linear assumptions, respectively.

Stronger Soundness Definitions. The main motivation for introducing PKENO was protocol design: some player sends a message to Bob securely by encrypting it under Bob's public key. If Bob finds out (possibly later) that the message is somehow "invalid", he can convince other participants of this fact without getting back to the (possibly) dishonest sender. *Proof soundness* ensures that Bob can do so convincingly; in particular, it states that if a ciphertext C encrypts a message m, then Bob cannot make a proof for C being an encryption of a different message m' (including the case of invalid messages $m' = \perp$). In the game that formally defines this security notion [DHKT08, Gal09], the challenger produces a private/public key pair, hands it to the adversary, who outputs a message of which he receives an encryption C. The adversary wins if he outputs a different message and makes a valid proof that this was the opening.

Thus, previous definitions of proof soundness [DHKT08, Gal09] only considered the case of honestly chosen keys, where a malicious receiver tries to claim a different decryption result under the genuine keys. In real-world applications, however, the keys are usually chosen by the users themselves. It seems thus natural to let *the adversary* choose the keys in the security experiment to reflect this fact. Hence, we define two stronger flavors of proof soundness, where the first one is analogous to the original definition given by [DHKT08], but lets the adversary choose his keys. The second one is akin to the binding property of commitment schemes and states that no adversary can find a public key, a ciphertext with two messages and valid proofs for each of them. We relate all notions formally.

Note that strengthening proof soundness also makes sense for the other applications given above. It can be used towards reducing the need for trusted setup in group signatures: the opener could choose his opening key and add corresponding information to the public parameters. Strong proof soundness then guarantees non-frameability even in this setting.

A Note on PKENO From General Assumptions. In [DHKT08], Damgård et al. already discussed how to construct PKENO from general assumptions using general but rather inefficient non-interactive zero-knowledge (NIZK) proofs. The idea of the construction is as follows. The receiver commits initially to its secret key. Whenever the proof algorithm is executed, it outputs a non-interactive zero-knowledge proof showing that the secret key committed to corresponds to the public key, and that decryption of the ciphertext C indeed yields the message m. Although this construction satisfies the security definitions of [DHKT08], it does not seem to be sufficient for our stronger soundness definitions. In particular, this construction does not make any statements about "invalid" ciphertexts.

Nonetheless, we briefly discuss here how to modify the idea in order to satisfy our stronger definitions, obtaining a scheme under general assumptions meeting our security notions. In our modification, the encryption algorithm adds a NIZK proof showing the well-formedness of the ciphertext (somehow in the fashion of [NY90, Sah99]) under the public-key, allowing anyone to detect invalid ciphertexts. The prove algorithm then rejects any ciphertext whose NIZK proof is invalid. If, on the other hand, the NIZK proof in the ciphertext is valid, then the prove algorithm proceeds as before, computing a second NIZK proof as described by Damgård et al. We note that, in the scheme by [DHKT08] with weak soundness, the common reference string (CRS) for the NIZK proofs can be put into the honestly chosen public key. In contrast, for stronger soundness with adversarially chosen keys (as in our case), we need to assume that the CRS is a public parameter (common reference string model).

Future Work. We leave as an open problem the construction of an efficient PKENO scheme based on a standard assumption like the Decision Diffie-Hellman assumption in the standard model.

2 Preliminaries

Notation. If x is a string then $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If k is a natural number, then 1^k denotes the string of k ones. If S is a set then $s_1, \ldots, s_n \xleftarrow{\$} S$ denotes the operation of picking n elements s_i of S independently and uniformly at random. We write $\mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and by $z \leftarrow \mathcal{A}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and letting z be the output. The abbreviation PPT refers to "probabilistic polynomial-time" algorithms [Gol01].

2.1 Public Key Encryption with Non-interactive Opening

A PKENO scheme $\mathsf{PKENO} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Prove}, \mathsf{Ver})$ is a tuple of five PPT algorithms:

- Gen is a randomized algorithm taking as input a security parameter 1^k and returns a key pair (pk, sk), where the public key pk includes a description of the message space $\mathcal{M}_{pk}$.
- Enc is a probabilistic algorithm taking as inputs a public key pk and a message $m \in \mathcal{M}_{pk}$. It returns a ciphertext C.
- Dec is a deterministic algorithm that takes as inputs a ciphertext C and a secret key sk. It returns a message $m \in \mathcal{M}_{pk}$ or the special symbol $\perp$ meaning that C is invalid.
- Prove is a probabilistic algorithm taking as inputs a ciphertext C and a secret key sk. It returns a proof π.
- Ver is a deterministic algorithm taking as inputs a public key pk, a ciphertext C, a plaintext m and a proof π. It returns a result $res \in \{0, 1\}$ meaning accepted and rejected proof, respectively. In particular, $\mathsf{Ver}(pk, C, \perp, \pi) = 1$ must be interpreted as the verifier being convinced that C is an invalid ciphertext.

Correctness requires that for an honestly generated key pair $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$, it holds that:

- For all messages $m \in \mathcal{M}_{pk}$ we have $\Pr\left[\, \mathsf{Dec}\big(sk, \mathsf{Enc}(pk, m)\big) = m \,\right] = 1$.
- For all ciphertexts C, $\Pr\left[\, \mathsf{Ver}\big(pk, C, \mathsf{Dec}(sk, C), \mathsf{Prove}(sk, C)\,\big) = 1 \,\right] = 1$.

Security of PKENO is defined by indistinguishability under chosen-ciphertext and prove attacks (IND-CCPA) and proof soundness [DHKT08, Gal09]. We formally define both notions and propose strengthened definitions for proof soundness in Section 3.

Definition 1 (IND-CCPA security). *Let us consider the following game between a challenger and an adversary $\mathcal{A}$:*

Setup: *The challenger runs* $\mathsf{Gen}(1^k)$ *and gives pk to* $\mathcal{A}$*.*

Phase 1: *The adversary issues queries of the form:*

a) decryption query to an oracle $\mathsf{Dec}(sk, \cdot)$*;*
b) proof query to an oracle $\mathsf{Prove}(sk, \cdot)$*.*

These may be asked adaptively in that they may depend on the answers to previous queries.

Challenge: *At some point,* $\mathcal{A}$ *outputs two equal-length messages* $m_0, m_1 \in \mathcal{M}_{pk}$*. The challenger chooses a random bit* β *and returns* $C^\star \leftarrow \mathsf{Enc}(pk, m_\beta)$*.*

Phase 2: *As Phase 1, except that neither decryption nor proof queries on* $C^\star$ *are allowed.*

Guess: *The adversary* $\mathcal{A}$ *outputs a guess* $\beta' \in \{0,1\}$*. The adversary wins the game if* $\beta = \beta'$*.*

Define $\mathcal{A}$*'s advantage as* $\mathsf{Adv}^{\text{ind-ccpa}}_{\mathsf{PKENO},\mathcal{A}}(1^k) := \left|\Pr[\beta' = \beta] - \frac{1}{2}\right|$*. A scheme* PKENO *is called* indistinguishable against chosen-ciphertext and prove attacks (IND-CCPA secure) *if for every PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\text{ind-ccpa}}_{\mathsf{PKENO},\mathcal{A}}(\cdot)$ *is negligible.*

We recall the original definition [DHKT08, Gal09] of proof soundness under genuine keys:

Definition 2 (Proof Soundness). *Consider the following game between a challenger and an adversary* $\mathcal{A}$*:*

Stage 0: *The challenger runs* $\mathsf{Gen}(1^k)$ *and gives the output* (pk, sk) *to* $\mathcal{A}$*.*
Stage 1: *The adversary chooses a message* $m \in \mathcal{M}_{pk}$*.*
Stage 2: *The challenger computes* $C \leftarrow \mathsf{Enc}(pk, m)$ *and gives it to* $\mathcal{A}$ *which returns* (m', π')*.*

$\mathcal{A}$*'s advantage is defined as the probability*

$$\mathbf{Adv}^{\text{proof-snd}}_{\mathsf{PKENO},\mathcal{A}}(1^k) := \Pr\left[\mathsf{Ver}(pk, C, m', \pi') = 1 \wedge m' \neq m\right] .$$

A scheme PKENO *is* proof sound *if for every PPT adversary* $\mathcal{A}$ *its advantage is negligible.*

In the above definition it is understood that $\perp \notin \mathcal{M}_{pk}$ and that the adversary thus also wins if π' is a valid proof for $m' = \perp$. It is also worth insisting that, since the adversary obtains the private key at the beginning of the game, decryption or prove oracles would be redundant.

2.2 Robust Non-interactive Threshold Public-Key Cryptosystems

Non-interactive threshold public-key cryptosystems, as formalized in [SG98], distribute the ability to decrypt among several parties. The private decryption key is shared among n servers such that at least t servers are needed for decryption. If the *combiner* wishes to decrypt some ciphertext C, it sends C to the decryption servers. After receiving at least t partial decryption shares from the servers, the combiner is able to reconstruct the plaintext from these shares. A *robust* TPKC [SG98, BBH06] provides the additional property that whenever the decryption of valid ciphertexts fails, the combiner can sieve out bad decryption shares and reveal the identity of the server having sent an invalid partial decryption.

SYNTAX. We use the same syntax as Boneh-Boyen-Halevi [BBH06] and Shoup-Gennaro [SG98] for (robust) non-interactive threshold public-key cryptostyems (TPKC). Formally, a robust TPKC scheme

$$\mathsf{TPKC} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{ShareDecrypt}, \mathsf{ShareVerify}, \mathsf{Combine})$$

consists of the following algorithms:

$\mathsf{Setup}(n, t, 1^k)$ takes as input a security parameter 1^k and integers $t, n \in \mathbb{N}$ (with $1 \leq t \leq n$) denoting the number of decryption servers n and the decryption threshold t. It outputs a triple $(\mathsf{PK}, \mathbf{VK}, \mathbf{SK})$, where PK is the public key, $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$ is a vector of n private-key shares and $\mathbf{VK} = (\mathsf{VK}_1, \ldots, \mathsf{VK}_n)$ is the corresponding vector of verification keys. Decryption server i is given the share (i, SK_i) that allows to derive decryption shares for any ciphertext. For each $i \in \{1, \ldots, n\}$, the verification key VK_i is used to check the validity of decryption shares generated using SK_i.

$\mathsf{Encrypt}(\mathsf{PK}, M)$ is a randomized algorithm that given a public key PK and a plaintext M outputs a ciphertext C.

$\mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$ on input of a public key PK, a ciphertext C and a private-key share (i, SK_i), this (possibly randomized) algorithm outputs either a decryption share $\mu_i = (i, \hat{\mu}_i)$, or a special symbol $(i, \perp)$.

$\mathsf{ShareVerify}(\mathsf{PK}, \mathsf{VK}_i, C, \mu_i)$ takes as input PK, the verification key VK_i, a ciphertext C and a purported decryption share $\mu_i = (i, \hat{\mu}_i)$. It outputs either `valid` or `invalid`. In the former case, μ_i is said to be a valid decryption share.

$\mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, \{\mu_1, \ldots, \mu_t\})$ given PK, $\mathbf{VK}$, C and a set of t decryption shares $\{\mu_1, \ldots, \mu_t\}$, this algorithm outputs either a plaintext M or $\perp$ if the set contains invalid decryption shares.

It is required that the consistency of PK with $\mathbf{VK}$ be publicly checkable. Namely, for any t-subset V of $\mathbf{VK}$, there must be an efficient algorithm[1], which we call $\mathsf{CheckKeys}$ in the upcoming sections, allowing to make sure that V is a valid set of verification keys w.r.t. PK.

CORRECTNESS. For any $(\mathsf{PK}, \mathbf{VK}, \mathbf{SK})$ generated by $\mathsf{Setup}(n, t, 1^k)$, it is required that

1. For any ciphertext C, if $\mu_i = \mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$, where SK_i is the i^{th} private-key share in $\mathbf{SK}$, then $\mathsf{ShareVerify}(\mathsf{PK}, \mathsf{VK}_i, C, \mu_i) = \texttt{valid}$. We emphasize that this must hold even in the event that $\mu_i = (i, \perp)$ (i.e., if C is deemed invalid).
2. If C is the output of $\mathsf{Encrypt}(\mathsf{PK}, M)$ and $S = \{\mu_1, \ldots, \mu_t\}$ is a set of decryption shares such that $\mu_i = \mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$ for t distinct private-key shares in $\mathbf{SK}$, then $\mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, S) = M$.

[1] Although such an algorithm is not formally required in [SG98, BBH06], it implicitly exists in all known robust TPKC and it is convenient to be considered here.

The security of robust TPKC is defined via two properties. The first one is the usual notion of chosen-ciphertext security for public key encryption adapted to the TPKC setting, while the other one is termed *consistency of decryptions*. For the formal security definitions we refer to [SG98].

3 Stronger Proof Soundness Definitions

We define our stronger version of proof soundness with adversarially chosen keys, as well as a notion similar to the binding property of commitments. Jumping ahead, we note that both strengthenings imply the original soundness definition but are themselves incomparable. The application usually determines which version should be considered. Arguably, they are both somewhat more realistic to use than Definition 2 in certain applications such as multiparty protocols, where parties might be able to cheat by maliciously generating their public key.

Definition 3 (Strong Proof Soundness). *Consider the following game between a challenger and an adversary $\mathcal{A}$:*

Stage 1: $\mathcal{A}(1^k)$ *outputs a public key pk and a message* $m \in \mathcal{M}_{pk}$.
Stage 2: *The challenger computes* $C \leftarrow \mathsf{Enc}(pk, m)$ *and gives it to* $\mathcal{A}$*, which returns* (m', π').

$\mathcal{A}$'s advantage is defined as the probability

$$\mathbf{Adv}^{\text{s-proof-snd}}_{\mathsf{PKENO},\mathcal{A}}(1^k) := \Pr\left[\mathsf{Ver}(pk, C, m', \pi') = 1 \,\wedge\, m' \neq m\right] \;.$$

A PKENO scheme is strongly proof sound *if any PPT adversary $\mathcal{A}$ has negligible advantage.*

An alternative strong notion of soundness (with adversarially chosen keys) follows the idea that, for any ciphertext, one can only find one valid message-proof pair. We call this the *committing* property:

Definition 4 (Committing Property). *A PKENO scheme is* strongly committing *if for any PPT adversary $\mathcal{A}$ that outputs $(pk, C, m, \pi, m', \pi')$ on input 1^k the following probability is negligible:*

$$\mathbf{Adv}^{\text{s-com}}_{\mathsf{PKENO},\mathcal{A}}(1^k) := \Pr\left[\mathsf{Ver}(pk, C, m, \pi) = 1 = \mathsf{Ver}(pk, C, m', \pi') \,\wedge\, m \neq m'\right] \;.$$

The following shows that Definitions 3 and 4 are actually achievable—by a practical scheme.

Theorem 1. *Galindo's PKENO scheme* [Gal09] *is strongly proof sound and strongly committing.*

The proof is deferred to the full version, where we compare the different notions of proof of soundness, showing that Definitions 3 and 4 are incomparable while both are strictly stronger than the original notion of proof soundness (Def. 2). Comparing the new notions in the "Knowledge of Secret Key" (KOSK) model, where the adversary has to prove knowledge of the secret key, we further prove that the committing property is strictly stronger than strong soundness. We note that all our proofs preserve IND-CCPA security. As for the separation we further show that if there exists a proof-sound scheme which is also IND-CCPA, then there exists an IND-CCPA scheme which is *not* strongly committing (strongly proof sound, resp.) but still proof sound. It is also easy to see that the case of adversarially chosen keys is strictly stronger, independently of the question whether the PKENO scheme is IND-CCPA secure or not. These results are formally stated and proven in the full version.

4 On Generic Constructions for PKENO

In this section, we first show that an apparently straightforward PKENO construction (briefly) suggested in [DHKT08] fails to provide chosen-ciphertext security (as defined in that work). Next, we propose a simple and efficient transformation from robust TPKE to PKENO. Finally, we describe two concrete PKENO schemes obtained from this transformation. The first relies on the Decisional Diffie-Hellman assumption and the Random-Oracle Model, while the second relies on the Decisional Linear assumption and is proven secure in the standard model.

4.1 Witness-Recovering Encryption Does Not Suffice

In a PKEWR scheme, decryption recovers the random coins r used to encrypt a ciphertext C. Damgård *et al.* [DHKT08] proposed to use r as the opening proof for a PKENO scheme. Verification then proceeds by re-encrypting the plaintext m as $C' = \mathsf{Enc}(pk_B, m; r)$, checking whether $C = C'$, and accepting/rejecting the proof accordingly. A subtle issue arises when dealing with invalid ciphertexts C, as in this case the random coins might simply not exist, for instance if C is not in the range of the encryption algorithm. This could be exploited by an adversary to abuse the security of the resulting PKENO system. We illustrate this by sketching an IND-CCPA attack against the candidate PKENO scheme one would obtain from the IND-CCA secure encryption scheme[2] of Peikert and Waters [PW08].

Let $F(\cdot), G(\cdot, \cdot)$ be trapdoor functions that can be inverted knowing the corresponding secret keys sk_F, sk_G; let h be a pairwise independent hash function, and let $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a strongly unforgeable one-time signature scheme [Mer89]. Then, the challenge ciphertext of plaintext m_β in [PW08] is constructed as follows: choose a one-time key pair $(\mathsf{SSK}^\star, \mathsf{SVK}^\star) \leftarrow \mathcal{G}(1^k)$, choose $x^\star$ uniformly at random from a certain set of strings, and compute $C_0^\star = F(x^\star)$, $C_1^\star = G(\mathsf{SVK}^\star, x^\star)$, $C_2^\star = h(x^\star) \oplus m_\beta$, $\sigma^\star = \mathcal{S}(\mathsf{SSK}^\star, (C_0^\star, C_1^\star, C_2^\star))$. The ciphertext is then $C^\star = (\mathsf{SVK}^\star, C_0^\star, C_1^\star, C_2^\star, \sigma^\star)$.

[2] In this scheme, not all the sender's coins are retrieved upon decryption since the private key of the one-time signature is not recovered. However, these unrecovered coins have no impact in our setting.

We show how an IND-CCPA attacker can abuse the prove oracle in the IND-CCPA game to mount a successful distinguishing attack.

Given the challenge $C^\star$, the adversary chooses $(\mathsf{SSK}, \mathsf{SVK}) \leftarrow \mathcal{G}(1^k)$ and submits $C := \big(\mathsf{SVK}, C_0^\star, C_1^\star, C_2^\star, \mathcal{S}(\mathsf{SSK}, (C_0^\star, C_1^\star, C_2^\star))\big)$ to the prove oracle. C is invalid since SVK used in C is different from $\mathsf{SVK}^\star$ embedded in $C_1^\star$; but what could be a proof for this? In [PW08] one can decrypt by either inverting $C_0^\star = F(x^\star)$ or $C_1^\star = G(\mathsf{SVK}^\star, x^\star)$. Inverting $F(x^\star)$ and giving $x^\star$ out to the adversary would result in trivially recovering m_β; we are thus left with inverting $G(\mathsf{SVK}^\star, x^\star)$. Inversion of G is done using both the secret key sk_G and the 'tag' SVK. This will result in a pre-image $x \neq x^\star$, and the question is whether the targeted $x^\star$ can be recovered from x and the publicly available information. Alas, this property is not covered in the model by [PW08]. Indeed, for certain lossy-trapdoor functions $G(\cdot, \cdot)$ the knowledge of such a pre-image x allows recovering $x^\star$. For instance, for the functions by Rosen and Segev [RS08], $x = (\mathsf{SVK} - \mathsf{SVK}^\star) \cdot x^\star$ with $\mathsf{SVK}, \mathsf{SVK}^\star, x, x^\star$ being integers in a ring, and therefore $x^\star$ can be trivially recovered. This results in a successful IND-CCPA attack.

One could wonder whether PKEWR schemes in the Random-Oracle Model could be of any help here. It is rather straightforward to prove that the PKENO obtained by using the randomness as a proof in the Fujisaki and Okamoto [FO99] encryption scheme suffers from a similar attack. Finding a practical generic construction for PKENO from a primitive weaker than identity-based encryption represents therefore an open problem.

4.2 Stronger Decryption-Consistency Definitions for TPKC

In our generic construction, we need somewhat stronger flavors of decryption consistency. In the first one, we require the adversary's advantage to remain negligible in an enhanced game where the challenger reveals PK and *all* decryption shares $\mathsf{SK}_1, \ldots, \mathsf{SK}_n$ in the setup phase.

Definition 5 (Decryption Consistency with Known Secret Keys). *Let us consider the following game between a challenger and an adversary $\mathcal{A}$:*

Setup: *The challenger runs* $\mathsf{Setup}(n, t, 1^k)$ *to obtain a triple* $(\mathsf{PK}, \mathbf{VK}, \mathbf{SK})$*, where* $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$*, and sends* $(\mathsf{PK}, \mathbf{VK}, \mathbf{SK})$ *to the adversary* $\mathcal{A}$*.*

Output: *$\mathcal{A}$ generates a ciphertext C and two unequal sets $S = \{\mu_1, \ldots, \mu_t\}$ and $S' = \{\mu'_1, \ldots, \mu'_t\}$ of decryption shares.*

Define $\mathcal{A}$'s advantage $\mathsf{Adv}^{\text{s-dec-con}}_{\mathsf{TPKC},\mathcal{A}}(1^k)$ *as the probability that the following conditions hold:*

1. *All decryption shares in S and S' are valid decryption shares w.r.t. the verification key $\mathbf{VK}$ and the ciphertext C.*
2. *S and S' each contain decryption shares from t distinct servers.*
3. $\mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, S) \neq \mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, S')$*.*

A robust TPKC is decryption consistent with known secret keys *if, for every PPT adversary $\mathcal{A}$, the advantage* $\mathsf{Adv}^{\text{s-dec-con}}_{\mathsf{TPKC},\mathcal{A}}(1^k)$ *is negligible.*

We further strengthen the definition and let the adversary choose the keys on her own.

Definition 6 (Strong Decryption Consistency). *A robust TPKC is* strongly decryption consistent *if for every PPT adversary $\mathcal{A}$ the advantage in a game that is similar to the one above is negligible, when $\mathcal{A}$ is allowed to generate consistent encryption/verification keys* $(\mathsf{PK}, \mathbf{VK})$ *on her own* without *having to publish the vector of decryption shares* $\mathbf{SK}$.

4.3 Robust TPKC Implies PKENO

Let $\mathsf{TPKC} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{ShareDecrypt}, \mathsf{ShareVerify}, \mathsf{Combine})$ be a robust threshold cryptosystem providing chosen-ciphertext security and strong decryption consistency. We turn it into a secure PKENO scheme as follows. We can essentially restrict ourselves to the case of a single-user threshold scheme, $t = n = 1$, but nonetheless state the transformation for general parameters. We use the threshold cryptosystem in a straightforward way to encrypt messages. To decrypt ciphertexts in our derived PKENO scheme, we first generate the decryption shares locally and then run the combiner to recover the message. The decryption shares also act as a soundness proof and the share verification determines the proof verification for PKENO. Chosen-ciphertext security of the threshold cryptosystem guarantees IND-CCPA security of the resulting PKENO scheme—using the fact that in the attack on the threshold cryptosystem the adversary can request to see decryption shares, which translates to access to a Prove oracle in the IND-CCPA game. Additionally, decryption consistency of the underlying threshold scheme provides soundness of the PKENO.

- $\mathsf{Gen}(1^k)$ Choose arbitrary integers $t, n \in \mathbb{N}$ such that $1 \leq t \leq n$ and run $\mathsf{Setup}(n, t, 1^k)$ to obtain $\big(\mathsf{PK}, \mathbf{VK} = (\mathsf{VK}_1, \ldots, \mathsf{VK}_n), \mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)\big)$. The key pair (pk, sk) for PKENO is defined as $pk = (\mathsf{PK}, \mathbf{VK}, n, t)$, $sk = \mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$. The plaintext (resp. ciphertext) space of PKENO is the plaintext (resp. ciphertext) space of TPKC.
- $\mathsf{Enc}(pk, M)$ To encrypt M, parse pk as $pk = (\mathsf{PK}, \mathbf{VK}, n, t)$ and compute $C = \mathsf{Encrypt}(\mathsf{PK}, M)$.
- $\mathsf{Dec}(sk, C)$ To decrypt C, conduct the following steps:
  1. For $i = 1, \ldots, t$, compute $\mu_i = \mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$.
  2. If there exists $j \in \{1, \ldots, t\}$ such that $\mu_j = (j, \perp)$ return $\perp$.
  3. Otherwise return $M = \mathsf{Combine}(\mathsf{PK}, \mathsf{VK}, C, S)$, where $S = \{\mu_1, \ldots, \mu_t\}$ is a set of valid shares.
- $\mathsf{Prove}(sk, C)$ A proof for the ciphertext C is computed by parsing sk as $(\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$ and doing the following:
  1. For $i = 1, \ldots, t$, compute $\mu_i = \mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$.
  2. Return the set of decryption shares $\pi = \{\mu_1, \ldots, \mu_t\}$.

– $\mathsf{Ver}(pk, C, M, \pi)$ parse pk as $(\mathsf{PK}, \mathbf{VK}, n, t)$ and π as a set of shares $\{\mu_1, \ldots, \mu_t\}$.
 1. Return 0 if π contains less than t shares or if $(\mathsf{VK}_1, \ldots, \mathsf{VK}_t)$ is inconsistent with PK (namely, if $\mathsf{CheckKeys}(\mathbf{PK}, (\mathsf{VK}_1, \ldots, \mathsf{VK}_t)) = 0$).
 2. If there exists $j \in \{1, \ldots, t\}$ s.t. $\mathsf{ShareVerify}(\mathsf{PK}, \mathsf{VK}_j, C, \mu_j) = \mathtt{invalid}$, return 0. Otherwise return 1 if $M = \mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, \{\mu_1, \ldots, \mu_t\})$ and 0 otherwise.

Theorem 2. *Robust TPKC satisfying decryption consistency with known secret keys (resp. strong decryption consistency) implies PKENO with proof soundness (resp. strongly committing).*

The statement of the above theorem is implied by the following lemmas:

Lemma 1. *The above generic PKENO system provides IND-CCPA security if the underlying robust TPKC is IND-TCCA secure.*

Proof. Let $\mathcal{A}$ be an IND-CCPA adversary against PKENO. We show how it readily yields a chosen-ciphertext adversary $\mathcal{B}$ against the underlying TPKC.

$\mathcal{B}$ starts by choosing $S = \{1, \ldots, t-1\}$ as the set of decryption servers to corrupt and obtains $(\mathsf{PK}, \mathbf{VK})$ as well as $((1, \mathsf{SK}_1), \ldots, (t-1, \mathsf{SK}_{t-1}))$ from her own challenger. The PKENO adversary $\mathcal{A}$ is supplied with the public key $pk = (\mathsf{PK}, \mathbf{VK}, n, t)$ and starts making decryption and proving queries. Whenever $\mathcal{A}$ queries a proof for some ciphertext C, $\mathcal{B}$ is able to compute $\mu_i = \mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$ for $i = 1, \ldots, t-1$ since she knows $\mathsf{SK}_1, \ldots, \mathsf{SK}_{t-1}$. To obtain the missing decryption share, $\mathcal{B}$ asks her challenger to reveal $\mu_t = \mathsf{ShareDecrypt}(\mathsf{PK}, t, \mathsf{SK}_t, C)$, which allows constructing $\pi = \{\mu_1, \ldots, \mu_t\}$ as long as TPKC provides correctness. It is not hard to see that $\mathcal{A}$'s decryption queries can be dealt with exactly in the same way: instead of revealing the set $\{\mu_1, \ldots, \mu_t\}$, $\mathcal{B}$ returns the output of $\mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, \{\mu_1, \ldots, \mu_t\})$.

At the challenge step, $\mathcal{A}$ outputs equal-length messages M_0, M_1 that are transmitted to $\mathcal{B}$'s challenger. The latter replies with a challenge TPKC ciphertext $C^\star$, which $\mathcal{B}$ relays to $\mathcal{A}$. In the second stage, $\mathcal{A}$ is allowed to make further decryption/proof queries. Since these never involve the challenge ciphertext $C^\star$, $\mathcal{B}$ is always able to answer them by invoking her own challenger as in the first phase. The game ends with $\mathcal{A}$ outputting a bit $b \in \{0, 1\}$, which is also $\mathcal{B}$'s result. It is straightforward to observe that if $\mathcal{A}$ is successful then so is $\mathcal{B}$. □

Lemma 2. *The above generic PKENO scheme is sound (resp. strongly committing) if it builds on a robust TPKC satisfying decryption consistency with known secret keys (resp. strong decryption consistency).*

Proof. We first show that if an adversary $\mathcal{A}$ defeats the soundness of PKENO in the sense of Definition 2 then there exists an adversary $\mathcal{B}$ breaking the decryption consistency with known secret keys in TPKC with the same advantage.

Namely, our adversary $\mathcal{B}$ obtains PK, $\mathbf{VK}$ and $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$ from her challenger. The weak-soundness adversary $\mathcal{A}$ then receives $pk = (\mathsf{PK}, \mathbf{VK}, n, t)$, $sk = \mathbf{SK}$. In Stage 1 of the game, $\mathcal{A}$ chooses a plaintext m that $\mathcal{B}$ encrypts using the public key PK of TPKC. Upon receiving the resulting ciphertext $C =$

$\mathsf{Encrypt}(\mathsf{PK}, m)$, $\mathcal{A}$ attempts to produce a pair (m', π') such that $\mathsf{Ver}(pk, C, m', \pi') = 1$ and $m' \neq m$. Since π' is a valid proof, it can necessarily be parsed as a set $\{\mu'_1, \ldots, \mu'_t\}$ of valid decryption shares. The correctness property of TPKC implies that, since $\mathcal{B}$ knows $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$, it must be able to generate another set $\pi = \{\mu_1, \ldots, \mu_t\}$ of decryption shares such that

$$m = \mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, \{\mu_1, \ldots, \mu_t\}) \ .$$

It follows that the sets π and π' are valid t-sets of decryption shares that break the decryption consistency with known secret keys of TPKC.

Proving that the strong decryption consistency of TPKC implies the strong committing property of PKENO is fairly straightforward: from a strong-committingness adversary $\mathcal{A}$, we immediately obtain a strong-decryption-consistency adversary $\mathcal{B}$ that returns whatever $\mathcal{A}$ outputs. □

Since in the KOSK model any strongly committing PKENO scheme is also strongly proof sound, Lemma 2 implies that a strongly proof-sound PKENO scheme can be obtained from a strongly decryption-consistent TPKC. In general, however, it seems that strong decryption consistency is not sufficient to imply strong proof soundness as well.

It turns out that for concrete TPKC constructions, such as the schemes by Shoup and Gennaro [SG98] and Arita and Tsurudome [AT09], it is possible to set $n = t = 1$ for improved efficiency. For instance, the consistency check between PK and $(\mathsf{VK}_1, \ldots, \mathsf{VK}_t)$ becomes trivial in Step 1 of the verification algorithm. We recall those TPKC in the full paper and describe the resulting efficient PKENO schemes in the next section.

Remark 1. The reader might wonder whether an efficient transformation from PKENO to robust non-interactive threshold cryptosystem exists. The answer is in the affirmative if we allow[3] these primitives to support *labels* [Sho04]. A label is an arbitrary string that is given as additional input to every algorithm of the PKENO and TPKC primitives, except the key generation algorithms. Then, the transformations from standard PKE to (non-robust) non-interactive threshold cryptosystem by Dodis and Katz [DK05] yield robust TPKC when replacing PKE by PKENO. Due to space limitations, we omit the details here but they follow easily from [DK05, Section 4.2].

5 New PKENO Constructions Implied by TPKC

This section describes new concrete schemes obtained from the transformation in Section 4.3.

[3] The reason of this restriction is the difficulty of *efficiently* constructing a PKENO system supporting labels from an ordinary PKENO. The standard black-box technique to include labels (by simply appending them to the plaintext upon encryption) in any public key encryption scheme fails to preserve security (in the sense of Definition 1) in the context of PKENO.

5.1 PKENO without Pairings in the Random-Oracle Model

In [SG98], Shoup and Gennaro described two CCA2-secure threshold cryptosystems in the random-oracle model. We show in the full version of the paper that the most efficient scheme TDH2 satisfies strong decryption consistency (although a weaker notion of consistency was considered in [SG98]). This scheme makes use of a prime-order group $\mathbb{G}$ where the Decision Diffie-Hellman problem[4] is assumed to be hard. It is easily seen to give rise to the following PKENO system.

- $\mathsf{Gen}(1^k)$: chooses a group $\mathbb{G}$ of prime order $p > 2^k$, $g, \bar{g} \stackrel{\$}{\leftarrow} \mathbb{G}$ and $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, and sets $h = g^x$. The public key pk includes $g, h, \bar{g}$, the description of the plaintext space $\mathcal{M}_{pk} = \{0,1\}^l$, where l depends polynomially on k, and hash functions $H_0 : \mathbb{G} \to \{0,1\}^l$, $H_1, H_2 : \{0,1\}^* \to \mathbb{Z}_p$ (to be modeled as random oracles). The secret key $sk = x$.
- $\mathsf{Enc}(pk, m)$: to encrypt a message $m \in \{0,1\}^l$, it proceeds as follows. It chooses $r, s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, it sets $K = h^r$ and computes
$$c = H_0(h^r) \oplus m,\ u = g^r,\ w_1 = g^s,\ \bar{u} = \bar{g}^r,\ \overline{w}_1 = \bar{g}^s,\ f_1 = s + re_1,$$
where $e_1 = H_1(c, u, w_1, \bar{u}, \overline{w}_1)$. Let us note that $(w_1, \overline{w}_1, f_1)$ constitutes a non-interactive zero-knowledge proof of equality of discrete logarithms $\log_g u = \log_{\bar{g}} \bar{u}$ [CP92]. The ciphertext is $C = (c, u, \bar{u}, e_1, f_1)$.
- $\mathsf{Dec}(sk, C)$: given $sk = x$ and $C = (c, u, \bar{u}, e_1, f_1)$, the decryption algorithm first checks whether $e_1 = H_1(c, u, w_1, \bar{u}, \overline{w}_1)$, where $w_1 = g^{f_1}/u^{e_1}$, $\overline{w}_1 = \bar{g}^{f_1}/\bar{u}^{e_1}$. If this is not the case, it returns $\perp$, meaning that C is invalid. Otherwise, it returns $m = c \oplus H_0(u^x)$.
- $\mathsf{Prove}(sk, C)$: given $C = (c, u, \bar{u}, e_1, f_1)$ and the secret key $sk = x$, the algorithm first checks if $e_1 = H_1(c, u, w_1, \bar{u}, \overline{w}_1)$, where $w_1 = g^{f_1}/u^{e_1}$, $\overline{w}_1 = \bar{g}^{f_1}/\bar{u}^{e_1}$. If this is not satisfied, it returns $\emptyset$, meaning that the ciphertext is invalid. Otherwise it computes $K = u^x$, chooses $s \leftarrow \mathbb{Z}_p$ and returns $\pi = (K, e_2, f_2)$, where
$$w_2 = g^s,\ \overline{w}_2 = u^s,\ e_2 = H_2(K, w_2, \overline{w}_2),\ f_2 = s + xe_2\ .$$
Note that $(w_2, \overline{w}_2, f_2)$ constitutes a non-interactive zero-knowledge proof of equality of discrete logarithms $\log_g h = \log_u K$.
- $\mathsf{Ver}(pk, c, m, \pi)$: parses C as $(c, u, \bar{u}, e_1, f_1)$ and π as (K, e_2, f_2). Then it performs the following tests:
  1. $e_1 \stackrel{?}{=} H_1(c, u, w_1, \bar{u}, \overline{w}_1)$, where $w_1 = g^{f_1}/u^{e_1}$, $\overline{w}_1 = \bar{g}^{f_1}/\bar{u}^{e_1}$
  2. $e_2 \stackrel{?}{=} H_2(K, w_2, \overline{w}_2)$, where $w_2 = g^{f_2}/h^{e_2}$, $\overline{w}_2 = u^{f_2}/K^{e_2}$

  If these tests are both correct and $c \oplus H_0(K) = m$ it returns 1; and 0 otherwise. If Test 1 fails, it outputs 1 iff $\pi = \emptyset$ and $m = \perp$. In any other case (e.g., Test 2 fails or can not be computed because $\pi = \emptyset$) it outputs 0.

[4] A slightly less efficient threshold cryptosystem described in [SG98] relies on the Computational Diffie-Hellman assumption (in the random oracle model) and can be turned into a PKENO system in the same way.

Since the underlying threshold cryptosystem is IND-TCCA secure and strongly decryption consistent, it follows that the above PKENO is IND-CCPA secure and strongly committing.

5.2 PKENO Based on the Decision Linear Assumption

Recently, Arita and Tsurudome [AT09] described an efficient way to thresholdize the decryption algorithm of Kiltz's tag-based encryption scheme [Kil06] using bilinear maps to achieve robustness. Their scheme readily yields another PKENO with strong soundness since it also provides decryption consistency in the strongest sense. The security proof of the resulting scheme is in the standard model under the Decision Linear assumption [BBS04] (in bilinear groups), which is the infeasibility of distinguishing g^{c+d} from random given $(g, g^a, g^b, g^{ac}, g^{bd})$, where $a, b, c, d \stackrel{\$}{\leftarrow} \mathbb{Z}_p$.

One of the advantages of this PKENO scheme is that it can be used in CCA2-anonymous group signatures that rely on the *linear encryption* technique [BBS04]. For instance, it can be used to obtain simpler and more efficient proofs of correct opening (as required by the model of Bellare *et al.* [BSZ05] in the context of dynamic groups) in Groth's fully anonymous group signatures [Gro07]: such a proof only consists of two group elements and its verification only entails two pairing evaluations, which is significantly cheaper than checking a pairing-based non-interactive witness indistinguishable proof as in [Gro07].

The description hereafter requires a strongly unforgeable [Mer89, ADR02] one-time signature scheme $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ as in the original CHK transformation [CHK04], where we assume for simplicity that the scheme's verification keys SVK can be embedded in $\mathbb{Z}_p$ (else one should first hash the key with a target-collision resistant hash function). We note that shorter ciphertexts can be obtained using Waters' technique [Wat05] in the same way as in the encryption scheme of [BMW05, Section 3.1]: at the expense of longer public keys (comprising $O(k)$ group elements), ciphertext components SVK and σ can be eliminated.

- $\mathsf{Gen}(1^k)$: chooses groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^k$ that are equipped with a bilinear map $e \colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, $g \stackrel{\$}{\leftarrow} \mathbb{G}$ and $x, y, u, v \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. The public key pk comprises $(X, Y, U, V) = (g^x, g^y, g^u, g^v)$, the description of the plaintext space $\mathcal{M}_{pk} = \mathbb{G}$ and that of a strong one-time signature $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. The secret key is $sk = (x, y, u, v)$.
- $\mathsf{Enc}(pk, m)$: to encrypt a message $m \in \mathbb{G}$, the algorithm first generates a one-time signature key pair $(\mathsf{SSK}, \mathsf{SVK}) \leftarrow \mathcal{G}(1^k)$. It chooses $r, s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and computes

$$C_1 = X^r,\ C_2 = Y^s,\ D_1 = (g^{\mathsf{SVK}} U)^r,\ D_2 = (g^{\mathsf{SVK}} V)^s,\ E = m \cdot g^{r+s},$$

  and $\sigma = \mathcal{S}(\mathsf{SSK}, (C_1, C_2, D_1, D_2, E))$. The ciphertext is $C = (\mathsf{SVK}, C_1, C_2, D_1, D_2, E, \sigma)$.

– $\mathsf{Dec}(sk, C)$: given $sk = (x, y, u, v)$ and $C = (\mathsf{SVK}, C_1, C_2, D_1, D_2, E, \sigma)$, the algorithm checks if $\mathcal{V}(\mathsf{SVK}, \sigma, (C_1, C_2, D_1, D_2, E)) = 1$, $D_1 = C_1^{(\mathsf{SVK}+u)/x}$ and $D_2 = C_2^{(\mathsf{SVK}+v)/y}$. If these checks fail, it returns $\perp$. Otherwise, it outputs $m = E \cdot C_1^{-1/x} \cdot C_2^{-1/y}$.
– $\mathsf{Prove}(sk, C)$: given $C = (\mathsf{SVK}, C_1, C_2, D_1, D_2, E, \sigma)$ and $sk = (x, y, u, v)$, the algorithm returns $\emptyset$ if $\mathcal{V}(\mathsf{SVK}, \sigma, (C_1, C_2, D_1, D_2, E)) = 0$ or if $D_1 \neq C_1^{(\mathsf{SVK}+u)/x}$ or $D_2 \neq C_2^{(\mathsf{SVK}+v)/y}$. Otherwise it computes and returns $\pi = (\pi_1, \pi_2) = (C_1^{1/x}, C_2^{1/y})$.
– $\mathsf{Ver}(pk, c, m, \pi)$: parses C as $(\mathsf{SVK}, C_1, C_2, D_1, D_2, E, \sigma)$ and π as $(\pi_1, \pi_2) \in \mathbb{G}^2$ (and outputs 0 if they cannot be parsed properly). Then, it performs the following tests:

1. $\mathcal{V}(\mathsf{SVK}, \sigma, (C_1, C_2, D_1, D_2, E)) \stackrel{?}{=} 1,\ e(C_1, g^{\mathsf{SVK}}U) \stackrel{?}{=} e(X, D_1)$, $e(C_2, g^{\mathsf{SVK}}V) \stackrel{?}{=} e(Y, D_2)$.
2. $e(\pi_1, X) \stackrel{?}{=} e(g, C_1),\ e(\pi_2, Y) \stackrel{?}{=} e(g, C_2),\ E \stackrel{?}{=} m \cdot \pi_1 \cdot \pi_2$.

If both tests are correct, it returns 1. If Test 1 fails, it outputs 1 iff $\pi = \emptyset$ and $m = \perp$. In any other situation, it outputs 0.

In comparison with [Gal09] (if we assume that CCA2-security is acquired using the technique of [BMW05, Section 3.1] in both schemes), the above system provides faster decryption (since no pairing evaluation is needed) at the expense of longer ciphertexts whereas proofs are equally expensive to verify. Its main advantage over [Gal09], in our opinion, lies in its possible use to provide simple proofs of correct opening in pairing-based group signatures.

It is also worth mentioning that other cryptosystems [Kil07, Boy07] also admit CCA2-secure threshold variants which can be proved strongly decryption consistent. They thus imply strongly committing PKENO instances bearing similarities with the above scheme. The Paillier-based TPKE scheme of [FP01] can be proved decryption consistent in the known secret key setting (cf. Definition 5). Proving it strongly decryption consistent seems harder.

Acknowledgments

The work described in this paper was supported in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II, by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG), and through the Center for Advanced Security Research Darmstadt (`www.cased.de`). The second author acknowledges the financial support of the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.). The fourth author is supported by the French ANR 07-TCOM-013-04 PACE Project and EADS.

References

[AB09] Agrawal, S., Boyen, X.: Identity-based encryption from lattices in the standard model (July 2009) (manuscript)

[ADR02] An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)

[AT09] Arita, S., Tsurudome, K.: Construction of threshold public-key encryptions through tag-based encryptions. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 186–200. Springer, Heidelberg (2009)

[BBH06] Boneh, D., Boyen, X., Halevi, S.: Chosen ciphertext secure public key threshold encryption without random oracles. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 226–243. Springer, Heidelberg (2006)

[BBS04] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

[BGH07] Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), pp. 647–657 (2007)

[BMW05] Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM Conference on Computer and Communications Security 2005, pp. 320–329 (2005)

[Boy07] Boyen, X.: Miniature cca2 pk encryption: Tight security without redundancy. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 485–501. Springer, Heidelberg (2007)

[BSZ05] Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)

[CHK04] Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)

[CHK09] Cash, D., Hofheinz, D., Kiltz, E.: How to delegate a lattice basis. Cryptology ePrint Archive: Report 2009/351 (July 2009)

[CP92] Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)

[CvH91] Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)

[DHKT08] Damgård, I., Hofheinz, D., Kiltz, E., Thorbek, R.: Public-key encryption with non-interactive opening. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 239–255. Springer, Heidelberg (2008)

[DK05] Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005)

[DT07] Damgård, I., Thorbek, R.: Non-interactive proofs for integer multiplication. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 412–429. Springer, Heidelberg (2007)

[FO99] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)

[FP01] Fouque, P.-A., Pointcheval, D.: Threshold cryptosystems secure against chosen-ciphertext attacks. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 351–368. Springer, Heidelberg (2001)

[Gal09] Galindo, D.: Breaking and repairing Damgård et al. public key encryption scheme with non-interactive opening. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 389–398. Springer, Heidelberg (2009)

[Gol01] Goldreich, O.: Foundations of Cryptography - Basic Tools. Cambridge University Press, Cambridge (2001)

[GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: 40th Annual ACM Symposium on Theory of Computing (STOC 2008), pp. 197–206 (2008)

[Gro07] Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)

[Kil06] Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)

[Kil07] Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)

[Mer89] Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)

[NY90] Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attack. In: Proc. of the Twenty-Second Annual ACM Symposium on Theory of Computing, pp. 427–437. ACM, New York (1990)

[Pei09] Peikert, C.: Bonsai trees (or, arboriculture in lattice-based cryptography). Cryptology ePrint Archive: Report 2009/359 (July 2009)

[PW08] Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC, pp. 187–196. ACM, New York (2008)

[RS08] Rosen, A., Segev, G.: Efficient lossy trapdoor functions based on the composite residuosity assumption. Cryptology ePrint Archive, Report 2008/134 (2008), http://eprint.iacr.org/

[Sah99] Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th Annual Symposium on Foundations of Computer Science, pp. 543–553. IEEE Computer Society Press, Los Alamitos (1999)

[SG98] Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)

[Sho04] Shoup, V.: Draft ISO/IEC 18033-2: An emerging standard for public-key encryption. Technical report, ISO/IEC (2004), http://www.shoup.net/iso

[Wat05] Waters, B.: Efficient identity-based encryption without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)