Non-Interactive CCA-Secure Threshold Cryptosystems with Adaptive Security: New Framework and Constructions

Benoît Libert[1] ⋆ and Moti Yung[2]

[1]Université catholique de Louvain, ICTEAM Institute (Belgium)
[2] Google Inc. and Columbia University (USA)

Abstract. In threshold cryptography, private keys are divided into n shares, each one of which is given to a different server in order to avoid single points of failure. In the case of threshold public-key encryption, at least $t \leq n$ servers need to contribute to the decryption process. A threshold primitive is said *robust* if no coalition of t malicious servers can prevent remaining honest servers from successfully completing private key operations. So far, most practical non-interactive threshold cryptosystems, where no interactive conversation is required among decryption servers, were only proved secure against static corruptions. In the adaptive corruption scenario (where the adversary can corrupt servers at any time, based on its complete view), all existing robust threshold encryption schemes that also resist chosen-ciphertext attacks (CCA) till recently require interaction in the decryption phase. A specific method (in composite order groups) for getting rid of interaction was recently suggested, leaving the question of more generic frameworks and constructions with better security and better flexibility (*i.e.*, compatibility with distributed key generation).

This paper describes a general construction of adaptively secure robust non-interactive threshold cryptosystems with chosen-ciphertext security. We define the notion of *all-but-one perfectly sound* threshold hash proof systems that can be seen as (threshold) hash proof systems with publicly verifiable and simulation-sound proofs. We show that this notion generically implies threshold cryptosystems combining the aforementioned properties. Then, we provide efficient instantiations under well-studied assumptions in bilinear groups (e.g., in such groups of prime order). These instantiations have a tighter security proof and are indeed compatible with distributed key generation protocols.

Keywords. Threshold cryptography, adaptive corruptions, public-key encryption, chosen-ciphertext security, non-interactivity, robustness.

⋆ This author acknowledges the Belgian Fund for Scientific Research (F.R.S.-F.N.R.S.) for his "Chargé de Recherches" fellowship and the BCRYPT Interuniversity Attraction Pole.

1 Introduction

Threshold cryptography [22, 23, 12] avoids single points of failure by splitting keys into $n > 1$ shares which are held by servers in such a way that at least t out of n servers should contribute to private key operations. In (t, n)-threshold cryptosystems, an adversary breaking into up to $t - 1$ servers should not jeopardize the security of the system.

Chosen-ciphertext security [45] (or IND-CCA for short) is widely recognized as the standard security notion for public-key encryption. Securely distributing the decryption procedure of CCA-secure public key schemes has proved to be a challenging task. As discussed in, e.g., [49, 25], the difficulty is that decryption servers should return their partial decryption results, called "decryption shares", before knowing whether the incoming ciphertext is valid or not and partial decryptions of ill-formed ciphertexts may leak useful information to the adversary.

The first solution to this problem was put forth by Shoup and Gennaro [49] and it requires the random oracle model [5], notably to render valid ciphertexts publicly recognizable. In the standard model, Canetti and Goldwasser [15] gave a threshold variant of the Cramer-Shoup encryption scheme [16]. Unfortunately, their scheme requires interaction among decryption servers to obtain robustness (*i.e.*, ensure that no coalition of $t - 1$ malicious servers can prevent uncorrupted servers from successfully decrypting) as well as to render invalid ciphertexts harmless. The approach of [15] consists in randomizing the decryption process in such a way that partial decryptions of invalid ciphertexts are uniformly random and thus meaningless to the adversary. To avoid the need to jointly generate randomizers at each decryption, shareholders can alternatively store a large number (*i.e.*, proportional to the expected number of decryptions) of pre-shared secrets, which does not scale well. Cramer, Damgård and Ishai suggested [20] a method to generate randomizers without interaction but it is only efficient for a small number of servers.

Other threshold variants of Cramer-Shoup were suggested [1, 40] and Abe notably showed [1] how to achieve optimal resilience (namely, guarantee robustness as long as the adversary corrupts a minority of $t < n/2$ servers) in the Canetti-Goldwasser system [15]. In the last decade, generic constructions of CCA-secure threshold cryptosystems with static security were put forth [24, 52].

Non-Interactive Schemes. As an application of the Canetti-Halevi-Katz (CHK) paradigm [18], Boneh, Boyen and Halevi [8] came up with the first fully *non-interactive* robust CCA-secure threshold cryptosystem with a security proof in the standard model: in their scheme, decryption servers can generate their decryption shares *without* any communication with other servers. Their scheme takes advantage of bilinear maps to publicly check the validity of ciphertexts, which considerably simplifies the task of proving security in the threshold setting. In addition, the validity of decryption shares can be verified in the same way, which provides robustness. Similar applications of the CHK methodology to threshold cryptography were studied in [13, 36].

Recently, Wee [52] defined a framework allowing to construct non-interactive threshold signatures and (chosen-ciphertext secure) threshold cryptosystems in a static corruption model. He left as an open problem the extension of his framework in the scenario of adaptive corruptions.

Adaptive Corruptions. Most threshold systems (including [49, 15, 24, 25, 8]) have been analyzed in a static corruption model, where the adversary chooses which servers it wants to corrupt *before* the scheme is set up. Unfortunately, adaptive adversaries – who can choose whom to corrupt at any time, as a function of their entire view of the protocol execution – are known (see, e.g., [19]) to be strictly stronger. As discussed in [15], properly dealing with adaptive corruptions often comes at some substantial expense like a lower resilience. For example, the Canetti-Goldwasser system can be proved robust and adaptively secure when the threshold t is sufficiently small (typically, when $t = O(n^{1/2})$) but supporting an optimal number of faulty servers is clearly preferable.

Assuming reliable erasures, Canetti *et al.* [14] devised adaptively secure protocols for the distributed generation of discrete-logarithm-based keys and DSA signatures. Their techniques were re-used later on [3] in proactive [44] RSA signatures. In 1999, Frankel, MacKenzie and Yung [26, 27] independently showed different methods to achieve adaptive security in the erasure-enabled setting.

Subsequently, Jarecki and Lysyanskaya [34] eliminated the need for erasures and gave an adaptively secure variant of the Canetti-Goldwasser threshold cryptosystem [15] which appeals to interactive zero-knowledge proofs but is designed to remain secure in concurrent environments. Unfortunately, their scheme requires a fair amount of interaction among decryption servers. Abe and Fehr [2] showed how to dispense with zero-knowledge proofs in the Jarecki-Lysyanskaya construction so as to prove it secure in (a variant of) the universal composability framework but without completely eliminating interaction from the decryption procedure. As in most threshold variants of Cramer-Shoup, hedging against invalid decryption queries requires an interactive (though off-line) randomness generation phase for each ciphertext, unless many pre-shared secrets are stored.

Recently, the authors of this paper showed [39] an adaptively secure variant of the Boneh-Boyen-Halevi construction [8] using groups of composite order and the dual system encryption approach [50, 38] that was initially applied to identity-based encryption [48, 10]. The scheme of [39] is based on a very specific use of the Lewko-Waters techniques [38], which limits its applicability to composite order groups and makes it hard to combine with existing adaptively secure distributed key generation techniques. Also, the concrete security of this initial scheme is not optimal as its security reduction is related to the number of decryption queries made by the adversary. To solve these problems, we need a new approach and different methods to analyze the security of schemes.

Our contribution. Motivated by an open question raised by Wee [52] and the limitations of [39], we define a general framework for constructing robust, adaptively secure and fully non-interactive threshold cryptosystems with chosen-ciphertext security. Our goal is to have simple and practical client/server protocols, as advocated in [49][Section 2.5], and even avoid the off-line interactive randomness generation stage which is usually needed in threshold versions of

Cramer-Shoup.

To this end, we also appeal to hash proof systems (HPS) [17] and take advantage of the property that, in security reductions using the techniques of [16, 17], the simulator knows the private keys, which is convenient to answer adaptive corruption queries. Indeed, when the reduction has to reveal the internal state of dynamically-corrupted servers, it is not bound to a particular set of available shares since it knows them all. At the same time, we depart from [15] in that the validity of ciphertexts is made publicly verifiable – which eliminates the need to randomize the decryption operation – using non-interactive proofs satisfying some form of simulation-soundness [46]: in the security reduction, the simulator should be able to generate a proof for a possibly false statement but the adversary should be unable to do it on its own, even after having seen a fake proof.

To this end, we define the notion of *all-but-one perfectly sound threshold hash proof systems* that can be seen as (threshold) hash proof systems [17] with *publicly* verifiable proofs (as opposed to designed-verifier proofs used in traditional HPS [17]). More precisely, each proof is associated with a tag, in the same way as ciphertexts are associated with tags in [41, 36]. Real public parameters are indistinguishable from alternative parameters that are generated in an *all-but-one* mode, which is only used in the security analysis. In the latter mode, non-interactive proofs are perfectly sound on all tags, except for a single specific tag where some trapdoor makes it possible to simulate proofs for false statements. While our primitive bears similarities with Wee's extractable hash proof systems [51, 52] (where hash proof systems are also associated with tags), it is different in that no extractability property is required and proofs are always used as proofs of membership.

Using all-but-one perfectly sound threshold hash proof systems, we generically construct adaptively secure robust non-interactive threshold cryptosystems with optimal resilience. An additional benefit of this approach is to provide a better concrete security as the security proof requires a constant number of game transitions whereas, in [39], the number of games is proportional to the number of decryption queries.

Then, we show three concrete instantiations using number theoretic assumptions in bilinear groups. The first one uses groups whose order is a product of two primes (whereas three primes are needed in [39]). Our second and third schemes rely on the Groth-Sahai proof systems [31] in their instantiations based on the Decision Linear [9] and symmetric eXternal Diffie-Hellman assumptions [47]. The latter two constructions operate over bilinear groups of prime order, which allows for a significantly better efficiency than composite order groups (as discussed in [28]) and makes them much easier to combine with known adaptively secure discrete-log-based distributed key generation protocols. For example, in the erasure-free setting, the protocols of [34, 2] can be used so as to eliminate the need for a trusted dealer at the same time as the reliance on reliable erasures.

2 Background and Definitions

2.1 Definitions for Threshold Public Key Encryption

A non-interactive (t, n)-threshold encryption scheme is a set of algorithms with these specifications.

Setup(λ, t, n)**:** given a security parameter λ and integers $t, n \in \mathsf{poly}(\lambda)$ (with $1 \leq t \leq n$) denoting the number of decryption servers n and the threshold t, this algorithm outputs $(PK, \mathbf{VK}, \mathbf{SK})$, where PK is the public key, $\mathbf{SK} = (SK_1, \ldots, SK_n)$ is a vector of private-key shares and $\mathbf{VK} = (VK_1, \ldots, VK_n)$ is a vector of verification keys. Decryption server i is given the private key share (i, SK_i). For each $i \in \{1, \ldots, n\}$, the verification key VK_i will be used to check the validity of decryption shares generated using SK_i.

Encrypt(PK, M)**:** is a randomized algorithm that, given a public key PK and a plaintext M, outputs a ciphertext C.

Ciphertext-Verify(PK, C)**:** takes as input a public key PK and a ciphertext C. It outputs 1 if C is deemed valid w.r.t. PK and 0 otherwise.

Share-Decrypt(PK, i, 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 a special symbol $(i, \perp)$ if **Ciphertext-Verify**$(PK, C) = 0$. Otherwise, it outputs a decryption share $\mu_i = (i, \hat{\mu}_i)$.

Share-Verify(PK, VK_i, C, μ_i)**:** takes in PK, the verification key VK_i, a ciphertext C and a purported decryption share $\mu_i = (i, \hat{\mu}_i)$. It outputs either 1 or 0. In the former case, μ_i is said to be a *valid* decryption share. We adopt the convention that $(i, \perp)$ is an invalid decryption share.

Combine$(PK, \mathbf{VK}, C, \{\mu_i\}_{i \in S})$**:** given PK, $\mathbf{VK}$, C and a subset $S \subset \{1, \ldots, n\}$ of size $t = |S|$ with decryption shares $\{\mu_i\}_{i \in S}$, this algorithm outputs either a plaintext M or $\perp$ if the set contains invalid decryption shares.

Chosen-ciphertext security. We use a game-based definition of chosen-ciphertext security which is akin to the one of [49, 8] with the difference that the adversary can adaptively decide which parties it wants to corrupt.

Definition 1. *A non-interactive (t, n)-Threshold Public Key Encryption scheme is secure against chosen-ciphertext attacks (or IND-CCA2 secure) and adaptive corruptions if no PPT adversary has non-negligible advantage in this game:*

1. *The challenger runs* **Setup**(λ, t, n) *to obtain PK, a vector of private key shares $\mathbf{SK} = (SK_1, \ldots, SK_n)$ and verification keys $\mathbf{VK} = (VK_1, \ldots, VK_n)$. It gives PK and $\mathbf{VK}$ to the adversary $\mathcal{A}$ and keeps $\mathbf{SK}$ to itself.*

2 *The adversary $\mathcal{A}$ adaptively makes the following kinds of queries:*
   - *Corruption query: $\mathcal{A}$ chooses $i \in \{1, \ldots, n\}$ and obtains SK_i. No more than $t - 1$ private key shares can be obtained by $\mathcal{A}$ in the whole game.*
   - *Decryption query: $\mathcal{A}$ chooses an index $i \in \{1, \ldots, n\}$ and a ciphertext C. The challenger replies with $\mu_i =$ **Share-Decrypt**(PK, i, SK_i, C).*

3. *The adversary $\mathcal{A}$ chooses two equal-length messages M_0, M_1 and obtains $C^\star =$ **Encrypt**(PK, M_β) for some random bit $\beta \overset{R}{\leftarrow} \{0, 1\}$.*

4. *$\mathcal{A}$ makes further queries as in step 2 but is not allowed to make decryption queries on $C^\star$.*
5. *$\mathcal{A}$ outputs a bit β' and is deemed successful if $\beta' = \beta$. As usual, $\mathcal{A}$'s advantage is measured as the distance $\mathbf{Adv}(\mathcal{A}) = |\Pr[\beta' = \beta] - \frac{1}{2}|$.*

CONSISTENCY. A (t, n)-Threshold Encryption scheme provides decryption consistency if no PPT adversary has non-negligible advantage in a three-stage game where stages 1 and 2 are identical to those of Definition 1 with the difference that the adversary $\mathcal{A}$ is allowed to obtain *all* private key shares (alternatively, $\mathcal{A}$ can directly obtain $\mathbf{SK}$ at the beginning of the game). In stage 3, $\mathcal{A}$ outputs a ciphertext C and two t-sets of decryption shares $\Gamma = \{\mu_1, \ldots, \mu_t\}$ and $\Gamma' = \{\mu'_1, \ldots, \mu'_t\}$. The adversary $\mathcal{A}$ is declared successful if

1. $\mathbf{Ciphertext\text{-}Verify}(PK, C) = 1$.
2. Γ and Γ' only consist of valid decryption shares.
3. $\mathbf{Combine}(PK, \mathbf{VK}, C, \Gamma) \neq \mathbf{Combine}(PK, \mathbf{VK}, C, \Gamma')$.

We note that condition 1 prevents an adversary from trivially winning by outputting an invalid ciphertext, for which distinct sets of key shares may give different results. This definition of consistency is identical to the one of [49, 8] with the difference that $\mathcal{A}$ can adaptively corrupt servers.

2.2 Hardness Assumptions in Composite Order Groups

In one occasion, we appeal to groups $(\mathbb{G}, \mathbb{G}_T)$ of composite order $N = p_1 p_2$, where p_1 and p_2 are primes, with a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ (*i.e.*, for which $e(g^a, h^b) = e(g, h)^{ab}$ for any $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_N$). In the notations hereafter, for each $i \in \{1, 2\}$, $\mathbb{G}_{p_i}$ stands for the subgroup of order p_i in $\mathbb{G}$.

Definition 2 ([11]). *In a group $\mathbb{G}$ of composite order N, the* **Subgroup Decision** *(SD) problem is given $(g \in \mathbb{G}_{p_1},\ h \in \mathbb{G})$ and η, to decide whether $\eta \in \mathbb{G}_{p_1}$ or $\eta \in_R \mathbb{G}$. The* **Subgroup Decision** *assumption states that, for any PPT distinguisher $\mathcal{D}$, the SD problem is infeasible.*

2.3 Assumptions in Prime Order Groups

We also use bilinear maps $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ over groups of prime order p. We will work in symmetric pairing configurations, where $\mathbb{G} = \hat{\mathbb{G}}$, and sometimes in asymmetric configurations, where $\mathbb{G} \neq \hat{\mathbb{G}}$.

In the symmetric setting $(\mathbb{G}, \mathbb{G}_T)$, we rely on the following assumption.

Definition 3 ([9]). *In a group $\mathbb{G}$ of prime order p, the* **Decision Linear Problem** *(DLIN) is to distinguish the distributions $(g, g^a, g^b, g^{ac}, g^{bd}, g^{c+d})$ and $(g, g^a, g^b, g^{ac}, g^{bd}, g^z)$, with $a, b, c, d, z \overset{R}{\leftarrow} \mathbb{Z}_p$. The* **Decision Linear Assumption** *is the intractability of DLIN for any PPT distinguisher $\mathcal{D}$.*

The problem amounts to deciding if vectors $\vec{g_1} = (g^a, 1, g)$, $\vec{g_2} = (1, g^b, g)$ and $\vec{g_3} = (g^{ac}, g^{bd}, g^{\delta})$ are linearly dependent (*i.e.*, if $\delta = c + d$) or not.

In *asymmetric* bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$, we assume the hardness of the Decision Diffie-Hellman (DDH) problem in $\mathbb{G}$ and $\hat{\mathbb{G}}$. This implies the unavailability of efficiently computable isomorphisms between $\hat{\mathbb{G}}$ and $\mathbb{G}$. This assumption is called **Symmetric eXternal Diffie-Hellman** (SXDH) assumption. Given vectors $\vec{u}_1 = (g, h)$, $\vec{u}_2 = (g^a, h^c)$ in $\mathbb{G}^2$ or $\hat{\mathbb{G}}^2$, the SXDH assumption asserts the infeasibility of deciding whether $\vec{u}_1$ and $\vec{u}_2$ are linearly dependent (*i.e.*, whether $a = c \bmod p$).

3 All-But-One Perfectly Sound Threshold Hash Proof Systems

Let $\mathcal{C}$, $\mathcal{K}$ and $\mathcal{K}'$ be sets and let $\mathcal{V} \subset \mathcal{C}$ be a subset. Let also $\mathcal{R}$ be a space where random coins can be chosen. We mandate that $\mathcal{V}$, $\mathcal{K}$, $\mathcal{K}'$ and $\mathcal{R}$ be of exponential size in λ, where $\lambda \in \mathbb{N}$ is a security parameter. In addition, $\mathcal{C}$, $\mathcal{V}$ and $\mathcal{C} \backslash \mathcal{V}$ should be efficiently samplable and we also require the set $\mathcal{K}$ to form a group for some binary operation, which is denoted by $\odot$ hereafter.

An *all-but-one perfectly sound threshold hash proof system* for $(\mathcal{C}, \mathcal{V}, \mathcal{K}, \mathcal{K}', \mathcal{R})$ is a tuple of algorithms ($\mathsf{SetupSound}$, $\mathsf{SetupABO}$, Sample, Prove, $\mathsf{SimProve}$, Verify, $\mathsf{PubEval}$, $\mathsf{SharePrivEval}$, $\mathsf{ShareEvalVerify}$, $\mathsf{Combine}$) of efficient algorithms with the following specifications.

SetupSound(λ, t, n)**:** given a security parameter $\lambda \in \mathbb{N}$ and integers $t, n \in \mathsf{poly}(\lambda)$, this algorithm outputs a public key pk, a vector of private key shares $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and verification keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$.

SetupABO$(\lambda, t, n, \mathsf{tag}^\star)$**:** takes as input a security parameter $\lambda \in \mathbb{N}$, integers $t, n \in \mathsf{poly}(\lambda)$ and a tag $\mathsf{tag}^\star$. It outputs a public key pk, private key shares $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$, the corresponding verification keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ as well as a simulation trapdoor τ. It is important that τ be independent of $\{\mathsf{sk}_i\}_{i=1}^n$.

Sample(pk)**:** is a probabilistic algorithm that takes as input a public key pk. It draws random coins $r \overset{R}{\leftarrow} \mathcal{R}$ and outputs an element $\Phi \in \mathcal{V}$ along with the random coins r that will serve as a witness explaining Φ as an element of $\mathcal{V}$.

Prove$(\mathsf{pk}, \mathsf{tag}, r, \Phi)$**:** takes in a public key pk, a tag tag, an element $\Phi \in \mathcal{V}$ and the random coins $r \in \mathcal{R}$ that were used to sample Φ. It generates a non-interactive proof $\pi_{\mathcal{V}}$ that $\Phi \in \mathcal{V}$.

SimProve$(\mathsf{pk}, \tau, \mathsf{tag}, \Phi)$**:** takes as input a public key pk and a simulation trapdoor τ produced by $\mathsf{SetupABO}(\lambda, t, n, \mathsf{tag}^\star)$, a tag tag and an element $\Phi \in \mathcal{C}$. If $\mathsf{tag} \neq \mathsf{tag}^\star$, the algorithm outputs $\perp$. If $\mathsf{tag} = \mathsf{tag}^\star$, the algorithm produces a simulated NIZK proof $\pi_{\mathcal{V}}$ that $\Phi \in \mathcal{V}$.

Verify$(\mathsf{pk}, \mathsf{tag}, \Phi, \pi_{\mathcal{V}})$**:** takes as input a public key pk, a tag tag, an element $\Phi \in \mathcal{C}$ and a purported proof $\pi_{\mathcal{V}}$. It outputs 1 if and only if $\pi_{\mathcal{V}}$ is deemed as a valid proof that $\Phi \in \mathcal{V} \subset \mathcal{C}$.

PubEval(pk, r, Φ)**:** takes as input a public key pk, an element $\Phi \in \mathcal{V}$ and the random coins $r \in_R \mathcal{R}$ such that $(r, \Phi) \leftarrow \mathsf{Sample}(\mathsf{pk})$. It outputs a value $K \in \mathcal{K}$, which is called *public evaluation* of Φ.

$\mathsf{SharePrivEval}(\mathsf{pk}, \mathsf{sk}_i, \varPhi)$: is a deterministic algorithm that takes in a public key pk, a private key share sk_i and an element $\varPhi \in \mathcal{C}$. It outputs a value $K_i \in \mathcal{K}'$, called *private evaluation share* and a proof π_{K_i} that K_i was evaluated correctly.

$\mathsf{ShareEvalVerify}(\mathsf{pk}, \mathsf{vk}_i, \varPhi, K_i, \pi_{K_i})$: given a public key pk, a verification key vk_i, an element $\varPhi \in \mathcal{C}$, a private evaluation share $K_i \in \mathcal{K}'$ and its proof π_{K_i}, this algorithm outputs 1 if π_{K_i} is considered as a valid proof of the correct evaluation of K_i. Otherwise, it outputs 0.

$\mathsf{Combine}(\mathsf{pk}, \varPhi, \{(K_i, \pi_{K_i})\}_{i \in S})$: takes as input a public key pk, an element $\varPhi \in \mathcal{C}$ and a set of t pairs $\{(K_i, \pi_{K_i})\}_{i \in S}$, where $S \subset \{1, \ldots, n\}$, each one of which consists of a private evaluation share $K_i \in \mathcal{K}'$ and its proof π_{K_i}. If $\mathsf{ShareEvalVerify}(\mathsf{pk}, \mathsf{vk}_i, \varPhi, K_i, \pi_{K_i}) = 0$ for some $i \in S$, it outputs $\perp$. Otherwise, it outputs a value $K \in \mathcal{K}$.

We also define this algorithm which is implied by the above ones but will be convenient to use.

$\mathsf{PrivEval}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i \in S}, \varPhi)$: given a public key pk, a set of private key shares $\{\mathsf{sk}_i\}_{i \in S}$ where S is an arbitrary t-subset of $\{1, \ldots, n\}$, and an element $\varPhi \in \mathcal{C}$, this algorithm outputs the result of $\mathsf{Combine}(\mathsf{pk}, \varPhi, \{(K_i, \pi_{K_i})\}_{i \in S})$ where $(K_i, \pi_{K_i}) \leftarrow \mathsf{SharePrivEval}(\mathsf{pk}, \mathsf{sk}_i, \varPhi)$ for each $i \in S$.

The following properties are required from these algorithms and the sets $(\mathcal{C}, \mathcal{V}, \mathcal{K}, \mathcal{K}', \mathcal{R})$.

(Setup indistinguishability): For any integers (λ, t, n) with $1 \leq t \leq n$ and any tag $\mathsf{tag}^\star$, the output of $\mathsf{SetupSound}(\lambda, t, n)$ is computationally indistinguishable from the outputs $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n, \{\mathsf{vk}_i\}_{i=1}^n)$ of $\mathsf{SetupABO}(\lambda, t, n, \mathsf{tag}^\star)$.

(Correctness and Public Evaluability on $\mathcal{V}$): For any $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n, \{\mathsf{vk}_i\}_{i=1}^n)$ returned by $\mathsf{SetupSound}$ or $\mathsf{SetupABO}$, if $(r, \varPhi) \xleftarrow{R} \mathsf{Sample}(\mathsf{pk})$ (and thus $\varPhi \in \mathcal{V}$), it holds that:

1. For any $i \in \{1, \ldots, n\}$, if $(K_i, \pi_{K_i}) \leftarrow \mathsf{SharePrivEval}(\mathsf{pk}, \mathsf{sk}_i, \varPhi)$, the private evaluation share $K_i \in \mathcal{K}'$ is *uniquely* determined by $(\mathsf{pk}, \mathsf{vk}_i)$ and $\varPhi$. Moreover, the proof π_{K_i} satisfies $\mathsf{ShareEvalVerify}(\mathsf{pk}, \mathsf{vk}_i, \varPhi, K_i, \pi_{K_i}) = 1$.
2. For any t-subset $S \subset \{1, \ldots, n\}$, combining the corresponding private evaluation shares allows recomputing the public evaluation of $\varPhi$: namely, $\mathsf{PubEval}(\mathsf{pk}, r, \varPhi) = \mathsf{PrivEval}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i \in S}, \varPhi)$.

(Universality): For any $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n, \{\mathsf{vk}_i\}_{i=1}^n)$ produced by $\mathsf{SetupSound}$ or $\mathsf{SetupABO}$ and any $\varPhi \in \mathcal{C} \backslash \mathcal{V}$, for any subset $\bar{S} \subset \{1, \ldots, n\}$ of size $|\bar{S}| = t-1$, the statistical distance

$$\varDelta[(\mathsf{pk}, \{\mathsf{vk}_i\}_{i=1}^n, \{\mathsf{sk}_i\}_{i \in \bar{S}}, \varPhi, \mathsf{PrivEval}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^t, \varPhi)), \\ (\mathsf{pk}, \{\mathsf{vk}_i\}_{i=1}^n, \{\mathsf{sk}_i\}_{i \in \bar{S}}, \varPhi, K)],$$

where $K \xleftarrow{R} \mathcal{K}$, should be negligible.

(All-But-One Soundness): For all integers (λ, t, n) such that $1 \leq t \leq n$, any tag $\mathsf{tag}^\star$ and any outputs $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n, \{\mathsf{vk}_i\}_{i=1}^n, \tau)$ of $\mathsf{SetupABO}(\lambda, t, n, \mathsf{tag}^\star)$, these conditions are satisfied.

1. For any $\mathsf{tag} \neq \mathsf{tag}^\star$, proofs are always perfectly sound. Namely, if a proof $\pi_\mathcal{V}$ satisfies $\mathsf{Verify}(\mathsf{pk}, \mathsf{tag}, \Phi, \pi_\mathcal{V}) = 1$ for some $\Phi \in \mathcal{C}$, then it necessarily holds that $\Phi \in \mathcal{V}$.
2. For any $\Phi \in \mathcal{C}$, the trapdoor τ allows generating as simulated a proof $\pi_\mathcal{V} \leftarrow \mathsf{SimProve}(\mathsf{pk}, \tau, \mathsf{tag}^\star, \Phi)$ such that $\mathsf{Verify}(\mathsf{pk}, \mathsf{tag}^\star, \Phi, \pi_\mathcal{V}) = 1$ (note that $\pi_\mathcal{V}$ is a proof for a false statement if $\Phi \in \mathcal{C} \backslash \mathcal{V}$). Moreover, if $\Phi \in \mathcal{V}$, the simulated proof $\pi_\mathcal{V}$ should be perfectly indistinguishable from a real proof (*i.e.*, that would be generated by Prove using a witness $r \in \mathcal{R}$ of the fact that $\Phi \in \mathcal{V}$).

(Simulatability of Share Proofs): For all (λ, t, n) with $1 \leq t \leq n$, any tag $\mathsf{tag}^\star$, any outputs $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n, \{\mathsf{vk}_i\}_{i=1}^n, \tau)$ of $\mathsf{SetupABO}(\lambda, t, n, \mathsf{tag}^\star)$ and any $\Phi \in \mathcal{C}$, the proofs π_{K_i} obtained as $(K_i, \pi_{K_i}) \leftarrow \mathsf{SharePrivEval}(\mathsf{pk}, \mathsf{sk}_i, \Phi)$ should be simulatable using the trapdoor τ instead of $\{\mathsf{sk}_i\}_{i=1}^n$. Using τ and $(\mathsf{pk}, \{\mathsf{vk}_i\}_{i=1}^n, \Phi)$, an efficient algorithm $\mathcal{S}$ should be able to produce simulated proofs π_{K_i} that are perfectly indistinguishable from real proofs.

(Consistency): For all (λ, t, n) with $1 \leq t \leq n$, any output $(\mathsf{pk}, \{(\mathsf{vk}_i, \mathsf{sk}_i)\}_{i=1}^n)$ of $\mathsf{SetupSound}(\lambda, t, n)$, given $(\mathsf{pk}, \{(\mathsf{vk}_i, \mathsf{sk}_i)\}_{i=1}^n)$, it should be computationally infeasible to come up with a triple $(\mathsf{tag}, \Phi, \pi_\mathcal{V})$ as well as two distinct t-sets $\Gamma = \{(K_{i_1}, \pi_{K_{i_1}}), \ldots, (K_{i_t}, \pi_{K_{i_t}})\}$ and $\Gamma' = \{(K'_{j_1}, \pi'_{K_{j_1}}), \ldots, (K'_{j_t}, \pi'_{K_{j_t}})\}$, with $i_k, j_k \in \{1, \ldots, n\}$ for each $k \in \{1, \ldots, t\}$, such that the following three conditions are satisfied: (i) $\mathsf{Verify}(\mathsf{pk}, \mathsf{tag}, \Phi, \pi_\mathcal{V}) = 1$; (ii) for each $k \in \{1, \ldots, t\}$, it holds that $\mathsf{ShareEvalVerify}(\mathsf{pk}, \mathsf{vk}_{i_k}, \Phi, K_{i_k}, \pi_{K_{i_k}}) = 1$ and $\mathsf{ShareEvalVerify}(\mathsf{pk}, \mathsf{vk}_{j_k}, \Phi, K'_{j_k}, \pi'_{K_{j_k}}) = 1$; (iii) Γ and Γ' result in distinct combinations: $\mathsf{Combine}(\mathsf{pk}, \Phi, \Gamma) \neq \mathsf{Combine}(\mathsf{pk}, \Phi, \Gamma')$.

(Subset Membership Hardness): membership in $\mathcal{C}$ should be easy to check but membership in $\mathcal{V}$ should not. Moreover, this should hold *even* if τ is given. Namely, for all integers (λ, t, n) such that $1 \leq t \leq n$, any tag $\mathsf{tag}^\star$ and any outputs $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n, \{\mathsf{vk}_i\}_{i=1}^n, \tau)$ of $\mathsf{SetupABO}(\lambda, t, n, \mathsf{tag}^\star)$, for any PPT distinguisher $\mathcal{D}$, it must hold that:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{SM}}(\mathcal{D}) = |\Pr[\mathcal{D}(\mathcal{C}, \mathcal{V}, C_1, \tau) = 1 | C_1 \xleftarrow{R} \mathcal{C} \backslash \mathcal{V}] \\ - \Pr[\mathcal{D}(\mathcal{C}, \mathcal{V}, C_0, \tau) = 1 | C_0 \xleftarrow{R} \mathcal{V}]| \in \mathsf{negl}(\lambda).\end{aligned}$$

In the definition of the subset membership hardness property, the trapdoor τ should not carry any side information helping the distinguisher. For this reason, the latter receives τ as part of its input.

4 Adaptively Secure Robust Non-Interactive CCA2-Secure Threshold Cryptosystems from All-But-One Perfectly Sound Threshold Hash Proof Systems

Let us assume sets $(\mathcal{C}, \mathcal{V}, \mathcal{K}, \mathcal{K}', \mathcal{R})$ for which we have an all-but-one perfectly sound threshold hash proof system $\varPi^{\mathsf{ABO\text{-}THPS}} = (\mathsf{SetupSound}, \mathsf{SetupABO}, \mathsf{Sample}, \mathsf{Prove}, \mathsf{SimProve}, \mathsf{Verify}, \mathsf{PubEval}, \mathsf{SharePrivEval}, \mathsf{ShareEvalVerify}, \mathsf{Combine})$ that satisfies the conditions specified in Section 3. We assume that messages are in $\mathcal{K}$. The generic construction of CCA2-secure threshold cryptosystem goes as follows.

Keygen(λ, t, n)**:** given integers $\lambda, t, n \in \mathbb{N}$, choose a one-time signature scheme $\varSigma = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$, generate $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n, \{\mathsf{vk}_i\}_{i=1}^n) \leftarrow \mathsf{SetupSound}(\lambda, t, n)$ and output $(PK, \mathbf{SK}, \mathbf{VK})$, where the vectors of private key shares and verification keys are defined as $\mathbf{SK} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and $\mathbf{VK} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, respectively. The public key is $PK = (\mathsf{pk}, \varSigma)$.

Encrypt(M, PK)**:** to encrypt a message $M \in \mathcal{K}$ using $PK = (\mathsf{pk}, \varSigma)$,

1. Generate a one-time signature key pair $(\mathsf{SSK}, \mathsf{SVK}) \leftarrow \varSigma.\mathsf{Gen}(\lambda)$.
2. Choose $r \overset{R}{\leftarrow} \mathcal{R}$, compute $(r, \varPhi) \leftarrow \mathsf{Sample}(\mathsf{pk}, r)$ and blind the message as $C_0 = M \odot \mathsf{PubEval}(\mathsf{pk}, r, \varPhi)$.
3. Generate a proof $\pi_{\mathcal{V}} \leftarrow \mathsf{Prove}(\mathsf{pk}, \mathsf{SVK}, r, \varPhi)$ that $\varPhi \in \mathcal{V}$ with respect to the tag SVK.
4. Output $C = (\mathsf{SVK}, C_0, \varPhi, \pi_{\mathcal{V}}, \sigma)$, where $\sigma = \varSigma.\mathsf{Sig}(\mathsf{SSK}, (C_0, \varPhi, \pi_{\mathcal{V}}))$.

Ciphertext-Verify$\big(PK, C\big)$**:** parse the ciphertext C as $C = (\mathsf{SVK}, C_0, \varPhi, \pi_{\mathcal{V}}, \sigma)$ and PK as $(\mathsf{pk}, \varSigma)$. Return 1 if it holds that $\varSigma.\mathsf{Ver}\big(\mathsf{SVK}, (C_0, \varPhi, \pi_{\mathcal{V}}), \sigma\big) = 1$ and $\mathsf{Verify}(\mathsf{pk}, \mathsf{SVK}, \varPhi, \pi_{\mathcal{V}}) = 1$. Otherwise, return 0.

Share-Decrypt(SK_i, C)**:** given $SK_i = \mathsf{sk}_i$ and $C = (\mathsf{SVK}, C_0, \varPhi, \pi_{\mathcal{V}}, \sigma)$, return $(i, \perp)$ if it turns out that **Ciphertext-Verify**$\big(PK, C\big) = 0$. Otherwise, compute a pair $(K_i, \pi_{K_i}) \leftarrow \mathsf{SharePrivEval}(\mathsf{pk}, \mathsf{sk}_i, \varPhi)$ and return $\mu_i = (i, \hat{\mu}_i)$ where $\hat{\mu}_i = (K_i, \pi_{K_i})$.

Share-Verify$\big(PK, VK_i, C, (i, \hat{\mu}_i)\big)$**:** parse C as $(\mathsf{SVK}, C_0, \varPhi, \pi_{\mathcal{V}}, \sigma)$. If $\hat{\mu}_i = \perp$ or if $\hat{\mu}_i$ cannot be properly parsed as a pair (K_i, π_{K_i}), return 0. Otherwise, return 1 if $\mathsf{ShareEvalVerify}(\mathsf{pk}, \mathsf{vk}_i, \varPhi, K_i, \pi_{K_i}) = 1$ and 0 otherwise.

Combine$(PK, \mathbf{VK}, C, \{(i, \hat{\mu}_i)\}_{i \in S})$**:** parse C as $(\mathsf{SVK}, C_0, \varPhi, \pi_{\mathcal{V}}, \sigma)$. Return $\perp$ if there exists $i \in S$ such that **Share-Verify**$\big(PK, C, (i, \hat{\mu}_i)\big) = 0$ or if **Ciphertext-Verify**$\big(PK, C\big) = 0$. Otherwise, compute the combined value $K = \mathsf{Combine}(\mathsf{pk}, \varPhi, \{(K_i, \pi_{K_i})\}_{i \in S}) \in \mathcal{K}$, which unveils $M = C_0 \odot K^{-1}$.

We observe that there is no need to bind the one-time verification key SVK to the ciphertext components $(C_0, \varPhi, \pi_{\mathcal{V}})$ in any other way than by using it as a tag to compute the non-interactive proof $\pi_{\mathcal{V}}$. Indeed, if the adversary attempts to re-use parts $(C_0^\star, \varPhi^\star, \pi_{\mathcal{V}}^\star)$ of the challenge ciphertext and simply replaces the one-time verification key $\mathsf{SVK}^\star$ by a verification key SVK of its own, it will be forced to compute a proof $\pi_{\mathcal{V}}$ that correspond to the same $\Phi^{\star}$ as in the challenge phase but under the *new* tag SVK. Our security proof shows that this is infeasible as long as $\Pi^{\mathsf{ABO\text{-}THPS}}$ satisfies the properties of setup indistinguishability and all-but-one soundness.

The consistency property of the threshold encryption scheme is trivially implied by that of $\Pi^{\mathsf{ABO\text{-}THPS}}$ and we focus on proving its IND-CCA security. In the threshold setting, adaptive security is achieved by taking advantage of the fact that, in security reductions using hash proof systems, the simulator typically knows the private key and can thus answer adaptive queries at will. At the same time, invalid ciphertexts are harmless as they are made publicly recognizable due to the use of non-interactive proofs of validity: as long as these proofs are perfectly sound in all decryption queries, the simulator is guaranteed not to leak too much information about the particular private key it is using.

The main problem to solve is thus to make sure that *only* the simulator can simulate a fake proof in the challenge phase and this is where the all-but-one soundness property is handy.

Theorem 1. *The above threshold cryptosystem is IND-CCA secure against adaptive corruptions assuming that: (i)* $\Pi^{\mathsf{ABO\text{-}THPS}}$ *is an all-but-one perfectly sound hash proof system; (ii)* Σ *is a strongly unforgeable one-time signature.*

Proof. The proof is given in the full version of the paper. □

5 Instantiations

5.1 Construction in Groups of Composite Order $N = p_1 p_2$

The construction relies on a hash proof system in a group $\mathbb{G}$ of composite order $N = p_1 p_2$ and it is conceptually close to the one in [33] (notably because it builds on a $\log p_2$-entropic hash proof system, as defined in [37]). The public key includes group elements $(g, X = g^x)$ in the subgroup $\mathbb{G}_{p_1}$ of order p_1 and the sets $\mathcal{C}$ and $\mathcal{V}$ are defined to be $\mathbb{G}$ and $\mathbb{G}_{p_1}$, respectively. The sampling algorithm returns $\Phi = g^r \in \mathbb{G}_{p_1}$ for a random exponent $r \overset{R}{\leftarrow} \mathbb{Z}_N$, which allows publicly evaluating $H(X^r) = H(\Phi^x)$ using a pairwise independent hash function $H : \mathbb{G} \to \{0,1\}^{\ell}$. Since the public key is independent of $x \bmod p_2$, for any $\Phi \in \mathbb{G}$ that has a non-trivial component of order p_2, the "hash value" Φ^x has exactly $\log p_2$ bits of min-entropy and the leftover hash lemma implies that $H(\Phi^x)$ is statistically close to the uniform distribution in $\{0,1\}^{\ell}$ when ℓ is sufficiently small.

In order to turn the scheme into an all-but-one perfectly sound threshold HPS, we need a mechanism that proves membership in the subgroup $\mathbb{G}_{p_1}$ and guarantees the perfect soundness of proofs of membership for all tags $\mathsf{tag} \in \mathbb{Z}_N$ such that $\mathsf{tag} \neq \mathsf{tag}^{\star}$. To this end, we use additional public parameters $(u, v) \in \mathbb{G}^2$ and a tag-dependent group element $u^{\mathsf{tag}} \cdot v$ will serve as a common reference string to generate a non-interactive proof that $\Phi \in \mathbb{G}_{p_1}$. Membership in $\mathbb{G}_{p_1}$ can be non-interactively proved using a technique that can be traced back to [30]. The proof consists of a group element $\pi_{\mathrm{SD}} \in \mathbb{G}$ satisfying the equality

$e(\varPhi, u^{\mathsf{tag}} \cdot v) = e(g, \pi_{\mathrm{SD}})$, which ensures that $\varPhi \in \mathbb{G}_{p_1}$ as long as $u^{\mathsf{tag}} \cdot v$ has a $\mathbb{G}_{p_2}$ component. In the public parameters produced by $\mathsf{SetupABO}$, the value $u^{\mathsf{tag}} \cdot v$ thus has to be in $\mathbb{G} \backslash \mathbb{G}_{p_1}$ for any $\mathsf{tag} \neq \mathsf{tag}^\star$ in such a way that generating fake proofs that $\varPhi \in \mathbb{G}_{p_1}$ is impossible. At the same time, $u^{\mathsf{tag}^\star} \cdot v$ should be in $\mathbb{G}_{p_1}$ so that fake proofs can be generated for $\mathsf{tag}^\star$.

SetupSound(λ, t, n)**:** choose a group $\mathbb{G}$ of composite order $N = p_1 p_2$ for large primes $p_i > 2^{l(\lambda)}$ for each $i \in \{1, 2\}$ and for some polynomial $l : \mathbb{N} \to \mathbb{N}$. Then, conduct the following steps

1. Pick $g \xleftarrow{R} \mathbb{G}_{p_1}$, $u, v \xleftarrow{R} \mathbb{G}$, $x \xleftarrow{R} \mathbb{Z}_N$ and set $X = g^x \in \mathbb{G}_{p_1}$.
2. Choose a random polynomial $P[X] \in \mathbb{Z}_N[X]$ of degree $t - 1$ such that $P(0) = x$. For each $i \in \{1, \ldots, n\}$, compute $Y_i = g^{P(i)} \in \mathbb{G}_{p_1}$.
3. Select a pairwise independent hash function $H : \mathbb{G} \to \{0, 1\}^\ell$, where $\ell \leq l(\lambda) - 2\lambda$. Note that the range $\mathcal{K} = \{0, 1\}^\ell$ of H forms a group for the bitwise exclusive OR operation $\odot = \oplus$.
4. Define private key shares $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ as $\mathsf{sk}_i = P(i) \in \mathbb{Z}_N$ for each $i = 1$ to n. The vector $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ is defined as $\mathsf{vk}_i = Y_i \in \mathbb{G}_{p_1}$ for each i and the public key consists of $\mathsf{pk} = \big((\mathbb{G}, \mathbb{G}_T), N, g, X, u, v, H\big)$. In addition, we have $(\mathcal{C}, \mathcal{V}, \mathcal{K}, \mathcal{K}', \mathcal{R}) = (\mathbb{G}, \mathbb{G}_{p_1}, \{0, 1\}^\ell, \mathbb{G}, \mathbb{Z}_N)$.

SetupABO$(\lambda, t, n, \mathsf{tag}^\star)$**:** is like $\mathsf{SetupSound}$ with the difference that, instead of being chosen uniformly in $\mathbb{G}$, v is defined as $v = u^{-\mathsf{tag}^\star} \cdot g^\alpha$ for some random $\alpha \xleftarrow{R} \mathbb{Z}_N$. The algorithm also outputs the simulation trapdoor $\tau = \alpha \in \mathbb{Z}_N$.

Sample(pk)**:** parse the public key pk as $\big((\mathbb{G}, \mathbb{G}_T), N, g, X, u, v, H\big)$. Choose $r \xleftarrow{R} \mathbb{Z}_N$, compute $\varPhi = g^r \in \mathbb{G}_{p_1}$ and output the pair $(r, \varPhi) \in \mathbb{Z}_N \times \mathbb{G}_{p_1}$.

Prove$(\mathsf{pk}, \mathsf{tag}, r, \varPhi)$**:** parse pk as $\big((\mathbb{G}, \mathbb{G}_T), N, g, X, u, v, H\big)$ and return $\perp$ if $\varPhi \neq g^r$. Otherwise, compute and return $\pi_{\mathrm{SD}} = (u^{\mathsf{tag}} \cdot v)^r$.

SimProve$(\mathsf{pk}, \tau, \mathsf{tag}, \varPhi)$**:** return $\perp$ if $\mathsf{tag} \neq \mathsf{tag}^\star$ or if $\varPhi \notin \mathbb{G}$. Otherwise, use the simulation trapdoor $\tau = \alpha \in \mathbb{Z}_N$ to compute and output $\pi_{\mathrm{SD}} = \varPhi^\alpha$.

Verify$(\mathsf{pk}, \mathsf{tag}, \varPhi, \pi_{\mathrm{SD}})$**:** return 1 iff $(\varPhi, \pi_{\mathrm{SD}}) \in \mathbb{G}^2$ and $e(\varPhi, u^{\mathsf{tag}} \cdot v) = e(g, \pi_{\mathrm{SD}})$.

PubEval$(\mathsf{pk}, r, \varPhi)$**:** on input of the public key $\mathsf{pk} = \big((\mathbb{G}, \mathbb{G}_T), N, g, X, u, v, H\big)$, return $\perp$ if $(r, \varPhi) \notin \mathbb{Z}_N \times \mathbb{G}$. Otherwise, output $K = H(X^r) \in \{0, 1\}^\ell$.

SharePrivEval$(\mathsf{pk}, \mathsf{sk}_i, \varPhi)$**:** return $\perp$ if $\varPhi \notin \mathbb{G}$. Otherwise, compute and return (K_i, π_{K_i}), where $K_i = \varPhi^{\mathsf{sk}_i} = \varPhi^{P(i)}$ and $\pi_{K_i} = \varepsilon$ is simply the empty string.

ShareEvalVerify$(\mathsf{pk}, \mathsf{vk}_i, \varPhi, K_i, \pi_{K_i})$**:** if $K_i \notin \mathbb{G}$, $\mathsf{vk}_i \notin \mathbb{G}$ or $\pi_{K_i} \neq \varepsilon$, return 0. Otherwise, return 1 if $e(g, K_i) = e(\varPhi, \mathsf{vk}_i)$. In any other situation, return 0 (the proof π_{K_i} is ignored in this instantiation since, given key $\mathsf{vk}_i = Y_i$, the private evaluation share K_i is directly verifiable).

Combine$(\mathsf{pk}, \varPhi, \{(K_i, \pi_{K_i})\}_{i \in S})$**:** return $\perp$ if there exists an index $i \in S$ such that $\mathsf{ShareEvalVerify}(\mathsf{pk}, \mathsf{vk}_i, \varPhi, K_i, \pi_{K_i}) = 0$. Otherwise, compute and output $K = H(\prod_{i \in S} K_i^{\Delta_{i,S}(0)}) = H(\varPhi^x) \in \mathcal{K}$.

Theorem 2. *The above construction is an all-but-one perfectly sound threshold hash proof system if the SD assumption holds in* $\mathbb{G}$. (The proof is given in the full version of the paper).

When the above all-but-one perfectly sound threshold hash proof system is plugged into the generic construction of Section 4, the resulting threshold cryptosystem bears resemblance with the scheme in [39], which makes use of groups whose order is a product of three primes. However, it is more efficient and its security proof is completely different as the dual system encryption approach [50] is not used here.

5.2 Construction in Prime Order Groups

This section presents an all-but-one threshold hash proof system based on the DLIN assumption in prime order bilinear groups. The public key comprises elements $(g, g_1, g_2, X_1, X_2) \in \mathbb{G}^5$, where $X_1 = g_1^{x_1} \cdot g^z$, $X_2 = g_2^{x_2} \cdot g^z$ and (x_1, x_2, z) are part of the private key. The sets $\mathcal{C}$ and $\mathcal{V} \subset \mathcal{C}$ consist of $\mathcal{C} = \mathbb{G}^3$ and $\mathcal{V} = \{(\Phi_1, \Phi_2, \Phi_3) = (g_1^{\theta_1}, g_2^{\theta_2}, g^{\theta_1+\theta_2}) \mid \theta_1, \theta_2 \in \mathbb{Z}_p\}$, respectively. For any $\Phi = (\Phi_1, \Phi_2, \Phi_3) \in \mathcal{V}$, the public evaluation algorithm computes $X_1^{\theta_1} \cdot X_2^{\theta_2}$, which can be privately evaluated as $\Phi_1^{x_1} \cdot \Phi_2^{x_2} \cdot \Phi_3^z$.

As in the previous instantiation, we append to elements $\Phi \in \mathcal{V}$ a non-interactive proof of their membership of $\mathcal{V}$ (*i.e.*, a proof that $(g, g_1, g_2, \Phi_1, \Phi_2, \Phi_3)$ is a linear tuple) and, in this case, the proof is obtained using the Groth-Sahai techniques. However, we cannot simply combine them with a DLIN-based hash proof system in the obvious way. The reason is that, using parameters produced by $\mathsf{SetupABO}$ and under the special tag $\mathsf{tag}^\star$, $\mathsf{SimProve}$ must be able to compute a fake non-interactive proof of the statement $\Phi \in \mathcal{V}$ for an element $\Phi \notin \mathcal{V}$. At the same time, we should make sure that, for any tag such that $\mathsf{tag} \neq \mathsf{tag}^\star$, it will be impossible to simulate such proofs. To solve this problem, we need a form of one-time simulation soundness [46] which can be possibly obtained from Groth's simulation-sound non-interactive proofs [29] or a more efficient variant suggested by Katz and Vaikuntanathan [35]. However, the specific language that we consider allows for even more efficient constructions: it is actually possible to build on the Groth-Sahai proofs essentially without any loss of efficiency.

The solution is as follows. After having sampled a tuple $\Phi = (\Phi_1, \Phi_2, \Phi_3) \in \mathcal{V}$, the sampler generates his proof using a Groth-Sahai CRS that depends on tag. Algorithm $\mathsf{SetupABO}$ produces parameters in the fashion of the all-but-one technique [7]: the tag-based CRS is perfectly WI on the special tag $\mathsf{tag}^\star$ (which allows generating NIZK proofs for this tag) and perfectly sound for any other tag, which makes it impossible to convincingly prove false statements on tags $\mathsf{tag} \neq \mathsf{tag}^\star$. Malkin, Teranishi, Vahlis and Yung [42] used a similar idea of message-dependent CRS in the context of signatures. A difference with [42] is that we do not need to extract witnesses from adversarially-generated proofs and only use them as proofs of membership.

Interestingly, the same technique can be applied to have a more efficient simulation-sound proof of plaintext equality in the Naor-Yung-type [43] cryptosystem in [35][Section 3.2.2]: the proof can be reduced from 60 to 22 group elements and the ciphertext size is decreased by more than 50%.

SetupSound(λ, t, n)**:** Choose a group $\mathbb{G}$ of prime order $p > 2^\lambda$ with generators $g, g_1, g_2, f_1, f_2 \xleftarrow{R} \mathbb{G}$.

1. Choose $x_1, x_2, z \xleftarrow{R} \mathbb{Z}_p$ and set $X_1 = g_1^{x_1} g^z$, $X_2 = g_2^{x_2} g^z$. Define the vectors $\vec{g_1} = (g_1, 1, g)$ and $\vec{g_2} = (1, g_2, g)$. Then, pick $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p$ and define $\vec{g_3} = \vec{g_1}^{\xi_1} \cdot \vec{g_2}^{\xi_2}$.
2. Choose $\phi_1, \phi_2 \xleftarrow{R} \mathbb{Z}_p$ and define vectors $\vec{f_1} = (f_1, 1, g)$, $\vec{f_2} = (1, f_2, g)$ and $\vec{f_3} = \vec{f_1}^{\phi_1} \cdot \vec{f_2}^{\phi_2} \cdot (1, 1, g)$.
3. Choose random polynomials $P_1[X], P_2[X], P[X] \in \mathbb{Z}_p[X]$ of degree $t-1$ such that $P_1(0) = x_1$, $P_2(0) = x_2$ and $P(0) = z$. For each $i = 1$ to n, compute $Y_{i,1} = g_1^{P_1(i)} g^{P(i)}$, $Y_{i,2} = g_2^{P_2(i)} g^{P(i)}$.
4. Define shares $\mathbf{SK} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ as $\mathsf{sk}_i = (P_1(i), P_2(i), P(i)) \in (\mathbb{Z}_p)^3$ for each $i \in \{1, \ldots, n\}$. Verification keys $\mathbf{VK} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ are defined as $\mathsf{vk}_i = (Y_{i,1}, Y_{i,2}) \in \mathbb{G}^2$ for each $i \in \{1, \ldots, n\}$ and the public key is

$$\mathsf{pk} = \Big((\mathbb{G}, \mathbb{G}_T),\ g,\ \vec{g_1},\ \vec{g_2},\ \vec{g_3},\ \vec{f_1},\ \vec{f_2},\ \vec{f_3},\ X_1,\ X_2 \Big).$$

As for the sets $(\mathcal{C}, \mathcal{K}, \mathcal{K}', \mathcal{R})$, they are defined as $\mathcal{C} = \mathbb{G}^3$, $\mathcal{K} = \mathcal{K}' = \mathbb{G}$ and $\mathcal{R} = (\mathbb{Z}_p)^2$, respectively. The subset $\mathcal{V} \subset \mathcal{C}$ consists of the language $(\varPhi_1, \varPhi_2, \varPhi_3) \in \mathbb{G}^3$ for which there exists $\theta_1, \theta_2 \in \mathbb{Z}_p$ such that $\varPhi_1 = g_1^{\theta_1}$, $\varPhi_2 = g_2^{\theta_2}$ and $\varPhi_3 = g^{\theta_1 + \theta_2}$.

SetupABO$(\lambda, t, n, \mathsf{tag}^\star)$**:** is like $\mathsf{SetupSound}$ with the following differences.

1. In step 1, $\vec{g_3}$ is set as $\vec{g_3} = \vec{g_1}^{\xi_1} \cdot \vec{g_2}^{\xi_2} \cdot (1, 1, g)^{-\mathsf{tag}^\star}$ so that $\vec{g_3} \notin \mathrm{span}(\vec{g_1}, \vec{g_2})$.
2. In step 2, the vectors $(\vec{f_1}, \vec{f_2}, \vec{f_3})$ are chosen so as to have $\vec{f_3} = \vec{f_1}^{\phi_1} \cdot \vec{f_2}^{\phi_2}$.
3. The algorithm also outputs the trapdoor $\tau = (\xi_1, \xi_2, \phi_1, \phi_2) \in (\mathbb{Z}_p)^4$.

Sample(pk)**:** choose $\theta_1, \theta_2 \xleftarrow{R} \mathbb{Z}_p$, compute $\varPhi = (\varPhi_1, \varPhi_2, \varPhi_3) = (g_1^{\theta_1}, g_2^{\theta_2}, g^{\theta_1 + \theta_2})$ and output $\big((\theta_1, \theta_2), \varPhi\big)$.

Prove$\big(\mathsf{pk}, \mathsf{tag}, (\theta_1, \theta_2), \varPhi\big)$**:** parse pk as $\big((\mathbb{G}, \mathbb{G}_T), g, \vec{g_1}, \vec{g_2}, \vec{g_3}, \vec{f_1}, \vec{f_2}, \vec{f_3}, X_1, X_2\big)$. Parse $\varPhi$ as $(\varPhi_1, \varPhi_2, \varPhi_3)$. Define[1] $\vec{g}_{\mathsf{tag}} = \vec{g_3} \cdot (1, 1, g)^{\mathsf{tag}}$ and use $\mathbf{g}_{\mathsf{tag}} = (\vec{g_1}, \vec{g_2}, \vec{g}_{\mathsf{tag}})$ as a Groth-Sahai CRS to generate a NIZK proof that $(g, g_1, g_2, \varPhi_1, \varPhi_2, \varPhi_3)$ is a linear tuple. More precisely, generate commitments $\vec{C}_{\theta_1}$, $\vec{C}_{\theta_2}$ to exponents $\theta_1, \theta_2 \in \mathbb{Z}_p$ (in other words, compute $\vec{C}_{\theta_i} = \vec{g}_{\mathsf{tag}}^{\theta_i} \cdot \vec{g_1}^{r_i} \cdot \vec{g_2}^{s_i}$ with $r_i, s_i \xleftarrow{R} \mathbb{Z}_p$ for each $i \in \{1, 2\}$) and a proof $\pi_{(\theta_1, \theta_2)}$ that they satisfy

$$\varPhi_1 = g_1^{\theta_1}, \qquad \varPhi_2 = g_2^{\theta_2}, \qquad \varPhi_3 = g^{\theta_1 + \theta_2}. \tag{1}$$

The whole proof π_{LIN} for (1) consists of $\vec{C}_{\theta_1}$, $\vec{C}_{\theta_2}$ and $\pi_{(\theta_1, \theta_2)}$ (see the full version of the paper for details about the generation of this proof) and requires 12 elements of $\mathbb{G}$.

SimProve$(\mathsf{pk}, \tau, \mathsf{tag}, \varPhi)$**:** parses pk as above, τ as $(\xi_1, \xi_2, \phi_1, \phi_2) \in (\mathbb{Z}_p)^4$ and $\varPhi$ as $(\varPhi_1, \varPhi_2, \varPhi_3) \in \mathbb{G}^3$. If $\mathsf{tag} \neq \mathsf{tag}^\star$, return $\perp$. Otherwise, the commitments $\vec{C}_{\theta_1}, \vec{C}_{\theta_2}$ and the proof π_{LIN} must be generated for the Groth-Sahai CRS

[1] We assume that tags are non-zero. This can be enforced by having Prove and Verify output $\perp$ when $\mathsf{tag} = 0$.

$\mathbf{g}_{\mathsf{tag}^\star} = (\vec{g_1}, \vec{g_2}, \vec{g}_{\mathsf{tag}^\star})$, where $\vec{g}_{\mathsf{tag}^\star} = \vec{g_3} \cdot (1,1,g)^{\mathsf{tag}^\star} = \vec{g_1}^{\,\xi_1} \cdot \vec{g_2}^{\,\xi_2}$, which is a Groth-Sahai CRS for the witness indistinguishability setting.

1. Using the trapdoor (ξ_1, ξ_2), simulate proofs for multi-exponentiation equations (see the full version of the paper for details as to how such proofs can be simulated). That is, generate $\vec{C}_{\theta_1}, \vec{C}_{\theta_2}$ as commitments to 0 and compute $\pi_{(\theta_1,\theta_2)}$ as a simulated proof that relations (1) hold.
2. Output $\pi_{\mathrm{LIN}} = (\vec{C}_{\theta_1}, \vec{C}_{\theta_2}, \pi_{(\theta_1,\theta_2)})$ that consists of perfectly hiding commitments and simulated NIZK proofs which, on the CRS $(\vec{g_1}, \vec{g_2}, \vec{g}_{\mathsf{tag}^\star})$, are distributed as real proofs.

Verify$(\mathsf{pk}, \mathsf{tag}, \Phi, \pi_{\mathrm{LIN}})$**:** parse pk and Φ as above. Also, parse the proof π_{LIN} as $(\vec{C}_{\theta_1}, \vec{C}_{\theta_2}, \pi_{(\theta_1,\theta_2)}) \in \mathbb{G}^{12}$. Then, compute $\vec{g}_{\mathsf{tag}} = \vec{g_3} \cdot (1,1,g)^{\mathsf{tag}}$ and use $\mathbf{g}_{\mathsf{tag}} = (\vec{g_1}, \vec{g_2}, \vec{g}_{\mathsf{tag}})$ as a Groth-Sahai CRS to verify π_{LIN}. If the latter is deemed as a valid proof for the relations (1), return 1. Otherwise, return 0.

PubEval$\big(\mathsf{pk}, (\theta_1, \theta_2), \Phi\big)$**:** parse pk and Φ as above. Return $\perp$ if $(\Phi_1, \Phi_2, \Phi_3) \neq (g_1^{\theta_1}, g_2^{\theta_2}, g^{\theta_1+\theta_2})$. Otherwise, compute and return $K = X_1^{\theta_1} \cdot X_2^{\theta_2} \in \mathcal{K}$.

SharePrivEval$(\mathsf{pk}, \mathsf{sk}_i, \Phi)$**:** parse sk_i as $(P_1(i), P_2(i), P(i)) \in (\mathbb{Z}_p)^3$ and return $\perp$ if $\Phi \notin \mathbb{G}^3$. Otherwise, return (K_i, π_{K_i}), where $K_i = \Phi_1^{P_1(i)} \Phi_2^{P_2(i)} \Phi_3^{P(i)} \in \mathcal{K}'$ and $\pi_{K_i} = (\vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P, \pi'_{K_i}) \in \mathbb{G}^{15}$ is a proof consisting of commitments $\vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P$ to exponents $P_1(i), P_2(i), P(i) \in \mathbb{Z}_p$ and a proof π'_{K_i} that these satisfy the equations

$$K_i = \Phi_1^{P_1(i)} \cdot \Phi_2^{P_2(i)} \cdot \Phi_3^{P(i)}, \qquad Y_{i,1} = g_1^{P_1(i)} g^{P(i)}, \qquad Y_{i,2} = g_2^{P_2(i)} g^{P(i)}. \quad (2)$$

The perfectly binding commitments $\vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P$ and the proof π'_{K_i} are generated using the vectors $\mathbf{f} = (\vec{f_1}, \vec{f_2}, \vec{f_3})$ as a Groth-Sahai CRS (in such a way that $\vec{C}_{P_1} = \vec{f_3}^{\,P_1(i)} \cdot \vec{f_1}^{\,r_{P_1}} \cdot \vec{f_2}^{\,s_{P_1}}$, for some $r_{P_1}, s_{P_1} \stackrel{R}{\leftarrow} \mathbb{Z}_p$, for example).

ShareEvalVerify$(\mathsf{pk}, \mathsf{vk}_i, \Phi, K_i, \pi_{K_i})$**:** parse vk_i as $(Y_{i,1}, Y_{i,2}) \in \mathbb{G}^2$ and return $\perp$ if (K_i, π_{K_i}) cannot be parsed as a tuple in $\mathbb{G} \times \mathbb{G}^{15}$. Otherwise, parse π_{K_i} as $\pi_{K_i} = (\vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P, \pi'_{K_i}) \in \mathbb{G}^{15}$ and return 1 if π'_{K_i} is a valid proof for equations (2). In any other situation, return 0.

Combine$(\mathsf{pk}, \Phi, \{(K_i, \pi_{K_i})\}_{i \in S})$**:** return $\perp$ if there is an index $i \in S$ for which $\mathsf{ShareEvalVerify}(\mathsf{pk}, \mathsf{vk}_i, \Phi, K_i, \pi_{K_i}) = 0$. Otherwise, compute

$$K = \prod_{i \in S} K_i^{\Delta_{i,S}(0)} = \Phi_1^{x_1} \cdot \Phi_2^{x_2} \cdot \Phi_3^{z} \in \mathcal{K}.$$

Theorem 3. *The above construction is an all-but-one perfectly sound threshold hash proof system assuming that the DLIN assumption holds in* $\mathbb{G}$. (The proof is given in the full version of the paper.)

The proof π_{LIN} takes 6 group elements whereas $\vec{C}_{\theta_1}, \vec{C}_{\theta_2}$ require 3 group elements each. If the scheme is instantiated using Groth's one-time signature [29] (which relies on the discrete logarithm assumption), SVK and σ demand 3 and 2 group elements, respectively. The whole ciphertext C thus consists of 21 group elements. Concretely, if each element has a representation of 512 bits, at the 128-bit security level, the ciphertext overhead amounts to 10240 bits.

From a computational standpoint, assuming that a multi-exponentiation with two base elements has roughly the same cost as a single-base exponentiation, the sender has to compute 19 exponentiations in $\mathbb{G}$ (we include the cost of generating SVK which incurs three exponentiations in Groth's one-time signature [29]). As for the verifier's workload, the validity of a ciphertext can be checked by computing a product of 12 pairings (which is more efficient than naively evaluating 12 individual pairings) using batch verification techniques as in [6].

In the full version of the paper, we show an even more efficient instantiation based on the Symmetric eXternal Diffie-Hellman assumption in prime order groups: only 6 pairing evaluations suffice to check $\pi_{\mathcal{V}}$.

Acknowledgements

We thank the anonymous reviewers and Carla Ràfols for useful comments.

References

1. M. Abe. Robust Distributed Multiplicaton with out Interaction. In *Crypto'99*, *LNCS* 1666, pp. 130–147, 1999.
2. M. Abe, S. Fehr. Adaptively Secure Feldman VSS and Applications to Universally-Composable Threshold Cryptography. In *Crypto'04*, *LNCS* 3152, pp. 317–334, 2004.
3. J. Almansa, I. Damgård, J.-B. Nielsen. Simplified Threshold RSA with Adaptive and Proactive Security. In *Eurocrypt'06*, *LNCS* 4004, pp. 593–611, 2006.
4. P. Barreto, M. Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. In *SAC'05*, *LNCS* 3897, pp. 319–331, 2005.
5. M. Bellare, P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, pp. 62–73, 1993.
6. O. Blazy, G. Fuchsbauer, M. Izabachène, A. Jambert, H. Sibert, D. Vergnaud. Batch Groth-Sahai. In *Applied Cryptography and Network Security (ACNS'10)*, *LNCS* 6123, pp. 218–235, 2010.
7. D. Boneh, X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Eurocrypt'04*, *LNCS* 3027, pp. 223–238, 2004.
8. D. Boneh, X. Boyen, S. Halevi. Chosen Ciphertext Secure Public Key Threshold Encryption Without Random Oracles. In *CT-RSA'06*, *LNCS* 3860, pp. 226–243, 2006.
9. D. Boneh, X. Boyen, H. Shacham. Short group signatures. In *Crypto'04*, LNCS 3152, pp. 41–55, 2004.
10. D. Boneh, M. Franklin. Identity-Based Encryption from the Weil Pairing. In *SIAM* J. of Computing 32(3), pp. 586–615, 2003. Earlier version in *Crypto'01*, *LNCS* 2139, 2001.
11. D. Boneh, E.-J. Goh, K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography Conference – TCC 2005*, *LNCS* 3378, pp. 325–341. Springer, 2005.

12. C. Boyd. Digital Multisignatures. In *Cryptography and Coding* (H.J. Beker and F.C. Piper Eds.), Oxford University Press, pp. 241–246, 1989.
13. X. Boyen, Q. Mei, B. Waters. Direct Chosen Ciphertext Security from Identity-Based Techniques. in *ACM CCS'05*, pp. 320–329, 2005.
14. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Adaptive Security for Threshold Cryptosystems. In *Crypto'99*, *LNCS* 1666, pp. 98–115, 1999.
15. R. Canetti, S. Goldwasser. An Efficient Threshold Public Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack. In *Eurocrypt'99*, *LNCS* 1592, pp. 90–106, 1999.
16. R. Cramer, V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Crypto'98*, LNCS 1462, pp. 13–25, 1998.
17. R. Cramer, V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Eurocrypt'02*, *LNCS* 2332, pp. 45–64, 2002.
18. R. Canetti, S. Halevi, J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In *Eurocrypt'04*, *LNCS* 3027, pp. 207–222, 2004.
19. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, T. Rabin. Efficient Multi-Party Computations Secure Against an Adaptive Adversary. In *Eurocrypt'99*, *LNCS* 1592, pp. 311–326, 1999.
20. R. Cramer, I. Damgård, Y. Ishai. Share Conversion, Pseudorandom Secret-Sharing and Applications to Secure Computation. In *TCC'05*, *LNCS* 3378, pp. 342–362, 2005.
21. I. Damgård. Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In *Crypto'91*, *LNCS* 576, pp. 445–456 1991.
22. Y. Desmedt. Society and Group Oriented Cryptography: A New Concept. In *Crypto'87*, *LNCS* 293, pp. 120–127, 1987.
23. Y. Desmedt, Y. Frankel. Threshold Cryptosystems. In *Crypto'89*, *LNCS* 435, pp. 307–315, 1989.
24. Y. Dodis, J. Katz. Chosen-Ciphertext Security of Multiple Encryption. In *TCC'05*, *LNCS* 3378, pp. 188–209, 2005.
25. P.-A. Fouque, D. Pointcheval. Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. In *Asiacrypt'01*, *LNCS* 2248, pp. 351–368, 2001.
26. Y. Frankel, P. MacKenzie, M. Yung. Adaptively-Secure Distributed Public-Key Systems. In *ESA'99*, *LNCS* 1643, pp. 4–27, 1999.
27. Y. Frankel, P. MacKenzie, M. Yung. Adaptively-Secure Optimal-Resilience Proactive RSA. In *Asiacrypt'99*, *LNCS* 1716, pp. 180–194, 1999.
28. D. Freeman. Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups. In *Eurocrypt'10*, *LNCS* 6110, pp. 44–61, 2010.
29. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Asiacrypt 2006*, *LNCS* 4284, pp. 444–459, 2006.
30. J. Groth, R. Ostrovsky, A. Sahai. Perfect non-interactive zero knowledge for NP. In *Eurocrypt'06*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, 2006.
31. J. Groth, A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Eurocrypt'08*, LNCS 4965, pp. 415–432, 2008.
32. J. Håstad, R. Impagliazzo, L. Levin, M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, vol. 28(4), pp. 1364–1396, 1999.
33. D. Hofheinz, E. Kiltz. The Group of Signed Quadratic Residues and Applications. In *Crypto'09*, *LNCS* 5677, pp. 637–653, 2009.

34. S. Jarecki, A. Lysyanskaya. Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures. In *Eurocrypt'00*, *LNCS* 1807, pp. 221–242, 2000.
35. J. Katz, V. Vaikuntanathan. Round-Optimal Password-Based Authenticated Key Exchange. In *TCC'11*, LNCS 6597, pp. 293–310, 2011.
36. E. Kiltz. Chosen-ciphertext security from tag-based encryption. In *TCC'06*, LNCS 3876, pp. 581–600, 2006.
37. E. Kiltz, K. Pietrzak, M. Stam, M. Yung. A New Randomness Extraction Paradigm for Hybrid Encryption. In *Eurocrypt'09*, *LNCS* 5479, pp. 590–609, 2009.
38. A. Lewko, B. Waters. New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In *TCC 2010*, LNCS 5978, pp. 455–479, 2010.
39. B. Libert, M. Yung. Adaptively Secure Non-Interactive Threshold Cryptosystems. In *ICALP 2011*, LNCS 6756, pp. 588–600, 2011.
40. P. MacKenzie. An Efficient Two-Party Public Key Cryptosystem Secure against Adaptive Chosen Ciphertext Attack. In *PKC'03*, LNCS 2567, pp. 47–61, 2003.
41. P. MacKenzie, M. Reiter, K. Yang. Alternatives to non-malleability: Definitions, constructions, and applications. In *TCC'04*, *LNCS* 2951, pp. 171–190. Springer, 2004.
42. T. Malkin, I. Teranishi, Y. Vahlis, M. Yung. Signatures resilient to continual leakage on memory and computation. In *TCC'11*, *LNCS* 6597, pp. 89–106, 2011.
43. M. Naor, M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC'90*, ACM Press, 1990.
44. R. Ostrovsky, M. Yung. How to Withstand Mobile Virus Attacks. In 10^{th} *ACM Symp. on Principles of Distributed Computing (PODC'91)*, 1991.
45. C. Rackoff, D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto'91*, *LNCS* 576, pp. 433–444, 1991.
46. A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In *FOCS'99*, pp. 543–553, 1999.
47. M. Scott. Authenticated ID-based Key Exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive: Report 2002/164.
48. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Crypto'84*, *LNCS* 196, pp. 47–53, 1984.
49. V. Shoup, R. Gennaro. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In *J. of Cryptology*, 15(2), pp. 75–96, 2002. Earlier version in *Eurocrypt'98*, *LNCS* 1403, pp. 1–16, 1998.
50. B. Waters. Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In *Crypto'09*, *LNCS* 5677, pp. 619–636, 2009.
51. H. Wee. Efficient Chosen-Ciphertext Security via Extractable Hash Proofs. In *Crypto'10*, *LNCS* 6223, pp. 314–332, 2010.
52. H. Wee. Threshold and Revocation Cryptosystems via Extractable Hash Proofs. In *Eurocrypt'11*, *LNCS* 6632, pp. 589–609, 2011.