Adaptively Secure Non-Interactive Threshold Cryptosystems

Benoît Libert[1] ⋆ and Moti Yung[2]

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

Abstract. Threshold cryptography aims at enhancing the availability and security of decryption and signature schemes by splitting private keys into several (say n) shares (typically, each of size comparable to the original secret key). In these schemes, a quorum of at least ($t \leq n$) servers needs to act upon a message to produce the result (decrypted value or signature), while corrupting less than t servers maintains the scheme's security. For about two decades, extensive study was dedicated to this subject, which created a number of notable results. So far, most practical threshold signatures, where servers act non-interactively, were analyzed in the limited static corruption model (where the adversary chooses which servers will be corrupted at the system's initialization stage). Existing threshold encryption schemes that withstand the strongest combination of adaptive malicious corruptions (allowing the adversary to corrupt servers at any time based on its complete view), and chosen-ciphertext attacks (CCA) all require interaction (in the non-idealized model) and attempts to remedy this problem resulted only in relaxed schemes. The same is true for threshold signatures secure under chosen-message attacks (CMA).To date (for about 10 years), it has been open whether there are non-interactive threshold schemes providing the highest security (namely, CCA-secure encryption and CMA-secure signature) with scalable shares (*i.e.*, as short as the original key) and adaptive security. This paper answers this question affirmatively by presenting such efficient decryption and signature schemes within a unified algebraic framework.

Keywords. Threshold cryptography, encryption schemes, digital signatures, adaptive corruptions, non-interactivity.

1 Introduction

Threshold cryptography [21, 22, 11] avoids single points of failure by splitting cryptographic keys into $n > 1$ shares which are stored by servers in distinct locations. Cryptographic schemes are then designed in such a way that at least t out of n servers should contribute to private key operations in order for these to succeed. In (t, n)-threshold cryptosystems (resp. signature schemes), an adversary breaking into up to $t-1$ servers should be unable to decrypt ciphertexts (resp. generate signatures) on its own.

Designing secure threshold public key schemes has proved to be a highly non-trivial task. For example, the random oracle model [6] was needed to analyze the first chosen-ciphertext secure (or CCA-secure for short) threshold encryption systems put forth by Shoup and Gennaro [40]. Canetti and Goldwasser [15] gave a standard model implementation based on the Cramer-Shoup encryption scheme [16]. Their scheme, however, eliminates random oracles at the expense of using interaction between decryption servers to obtain robustness (*i.e.*, ensure that no dishonest minority deviating from the protocol can prevent uncorrupted servers from successfully decrypting) and to make sure that invalid ciphertexts do not reveal useful information to the adversary[1]. Other chosen-ciphertext-

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

[1] The approach of [15] consists in randomizing the decryption process in such a way that decryption queries on invalid ciphertexts result in meaningless partial decryptions. To avoid decryption servers to jointly generate random values, they can alternatively store a large number of pre-shared secrets.

secure threshold cryptosystems were suggested in [1, 36, 23, 8].

Non-Interactive Schemes. Using the innovative Canetti-Halevi-Katz (CHK) methodology [17], Boneh, Boyen and Halevi [8] showed the first *non-interactive* robust CCA-secure threshold cryptosystem with a security proof in the standard model (*i.e.*, without the random oracle idealization): in their scheme, decryption servers can compute their partial decryption result (termed "decryption share") *without* having to talk to each other and, in groups with a bilinear map, decryption shares contain built-in proofs of their validity, which guarantees robustness. These properties were obtained by notably taking advantage of the fact that, using bilinear maps, valid ciphertexts are publicly recognizable in the Boneh-Boyen identity-based encryption system [7]. Similar applications of the CHK methodology were studied in [12, 32, 4].

In the context of digital signatures, Shoup [41] described non-interactive threshold signatures based on RSA and providing robustness.

Adaptive Corruptions. Historically, threshold primitives (including [40, 15, 23, 27, 8]) have been mostly studied 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 and depending on the previously collected information – are known (see, e.g., [18]) to be strictly stronger and substantially harder to deal with. As discussed in [15], properly handling them sometimes requires to sacrifice useful properties. For example, the Canetti-Goldwasser system can be proved secure against adaptive corruptions when the threshold t is sufficiently small (typically, when $t = O(n^{1/2})$) but this comes at the expense of a lower resilience and schemes supporting a linear number of faulty servers seem preferable.

To address the above concerns, Canetti *et al.* [14] proposed a method to cope with adaptive corruptions assuming reliable erasures (*i.e.*, players must be able to safely erase their local data when they no longer need them) and also achieve proactive security [37]. In the case of proactive RSA signatures, this approach requires all servers to refresh their shares (by jointly computing a sharing of zero) after each distributed private key operation (effectively making schemes n-out-of-n rather than t-out-of-n for any $t \leq n$). This limitation was removed in [29] and [3], where simpler adaptively secure proactive RSA signatures are described. In 1999, Frankel, MacKenzie and Yung [28, 29] showed different techniques to achieve adaptive security while still using erasures.

Later on, Jarecki and Lysyanskaya [31] eliminated the need for erasures and gave an adaptively secure variant of the Canetti-Goldwasser CCA-secure threshold cryptosystem [15]. Unfortunately, their scheme – which is also designed to remain secure in concurrent environments – requires a lot of interaction between decryption servers. Abe and Fehr [2] showed how to extend Jarecki and Lysyanskaya's threshold version of Cramer-Shoup in the universal composability framework but without completely eliminating interaction from the decryption algorithm.

Recently, Qin *et al.* [38] suggested a non-interactive threshold cryptosystem (more precisely, a threshold broadcast encryption scheme whose syntax is similar to [19, 20]) with adaptive security. Its downside is its lack of scalability since private key shares consist of $O(n)$ elements, where n is the number of servers (while, in prior schemes, the share size only depends on the security parameter).

Our contribution. We give the first robust threshold cryptosystem which is simultaneously chosen-ciphertext secure under adaptive corruptions and non-interactive while being scalable (*i.e.*, providing short private keys). Unlike [38], our scheme features constant-size private key shares (where "constant" means independent of t and n) for public keys of comparable size. In addition, it is conceptually simple and relies on assumptions of constant-size whereas [38] relies on a "q-type"

assumption where the input is a sequence of the form $(g, g^{\alpha}, \ldots, g^{(\alpha^q)})$, for some secret $\alpha \in \mathbb{Z}_p$. Unlike [14], we do not have to perform proactive refreshes of private key shares after each decryption operation.

Our starting point is the identity-based encryption (IBE) system [9, 39] proposed by Lewko and Waters [34] and the elegant dual system approach introduced by Waters [43]. The latter has proved useful to demonstrate full security in identity and attribute-based encryption [43, 33, 34] but, to the best of our knowledge, it has not been applied to threshold cryptosystems so far. It is worth noting that the security proof of our scheme is not simply a direct consequence of applying the CHK paradigm to the Lewko-Waters results [34] as the treatment of adaptive corruptions does not follow from [17, 34]. Like [34], our proof uses a sequence of games. While we also use so-called semi-functional decryption shares and ciphertexts as in the IBE setting [34], we have to consider two distinct kinds of semi-functional ciphertexts and an additional step (which aims at making all private key shares semi-functional) is needed in the proof to end up in a game where proving the security is made simple.

We also describe a non-interactive threshold signature that follows the same line of development and which can be proven secure in the standard model under adaptive corruptions. This appears to be the first security result under adaptive corruptions for non-interactive threshold signatures in the standard model.

Technically speaking, the encryption scheme can be visualized as a variant of the Boneh-Boyen-Halevi threshold system [8] in groups whose order is a product $N = p_1p_2p_3$ of three primes, which are chosen at key generation. Interestingly, if the factorization of N is somehow leaked, the proof of security under static corruptions implied by [8] still applies and only the proof of adaptive security ceases to go through. We also believe the semantically-secure variant of our scheme (which is obtained by removing the appropriate "checksum values" allowing to hedge against chosen-ciphertext attacks) to be of interest in its own right since it is multiplicatively homomorphic (like the ElGamal encryption scheme [24]) and retains security under adaptive corruptions in the threshold setting. It can thus find applications in important protocols such as e-voting for example.

Organization. Section 2 recalls the definitions of threshold cryptosystems. The scheme and its CCA-security are analyzed in sections 3.1 and 3.2, respectively. A variant with shorter ciphertexts is described in section 4. Our threshold signature is presented in appendix B.

2 Background and Definitions

2.1 Definitions for Threshold Public Key Encryption

Definition 1. *A non-interactive (t, n)-threshold encryption scheme is a set of algorithms with the following specifications.*

Setup(λ, t, n)**:** takes as input 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 decryption threshold t. It outputs a triple $(PK, \mathbf{VK}, \mathbf{SK})$, where PK is the public key, $\mathbf{SK} = (SK_1, \ldots, SK_n)$ is a vector of n private-key shares and $\mathbf{VK} = (VK_1, \ldots, VK_n)$ is the corresponding vector of verification keys. Decryption server i is given the share (i, SK_i) that allows deriving decryption shares for any ciphertext. 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 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 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 definition of chosen-ciphertext security which is identical to the one of [40, 8] with the difference that the adversary can adaptively choose which parties it wants to corrupt.

Definition 2. *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 a public key 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. *$\mathcal{A}$ chooses two equal-length messages M_0, M_1. The challenger flips a fair coin $\beta \stackrel{R}{\leftarrow} \{0, 1\}$ and computes $C^\star =$* **Encrypt**(PK, M_β).
4. *$\mathcal{A}$ makes further queries as in step 2 but it is not allowed to make decryption queries on the challenge ciphertext $C^\star$.*
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 2 with the difference that the adversary $\mathcal{A}$ is allowed to obtain *all* private key shares. In stage 3, $\mathcal{A}$ outputs a ciphertext C and two t-sets of decryption shares $\Phi = \{\mu_1, \ldots, \mu_t\}$ and $\Phi' = \{\mu'_1, \ldots, \mu'_t\}$. The adversary $\mathcal{A}$ is declared successful if

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

We note that condition 1 aims at preventing 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 [40, 8] with the difference that $\mathcal{A}$ can adaptively corrupt decryption servers.

2.2 Hardness Assumptions in Bilinear Groups of Composite Order

We shall use groups $(\mathbb{G}, \mathbb{G}_T)$ of composite order $N = p_1p_2p_3$ endowed with an efficiently computable map (a.k.a. pairing) $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that: (1) $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}$; (2) if $e(g, h) = 1_{\mathbb{G}_T}$ for each $h \in \mathbb{G}$, then $g = 1_{\mathbb{G}}$. An important property of composite order groups is that pairing two elements of order p_i and p_j, with $i \neq j$, always gives the identity element $1_{\mathbb{G}_T}$.

In the following, for each $i \in \{1, 2, 3\}$, we denote by $\mathbb{G}_{p_i}$ the subgroup of order p_i. For all distinct $i, j \in \{1, 2, 3\}$, we call $\mathbb{G}_{p_ip_j}$ the subgroup of order p_ip_j. In this setting, we rely on the following assumptions introduced in [34].

Assumption 1 Given a description of $(\mathbb{G}, \mathbb{G}_T)$ as well as $g \xleftarrow{R} \mathbb{G}_{p_1}, X_3 \xleftarrow{R} \mathbb{G}_{p_3}$ and $T \in \mathbb{G}$, it is infeasible to efficiently decide if $T \in \mathbb{G}_{p_1p_2}$ or $T \in \mathbb{G}_{p_1}$.

Assumption 2 Let $g, X_1 \xleftarrow{R} \mathbb{G}_{p_1}, X_2, Y_2 \xleftarrow{R} \mathbb{G}_{p_2}, Y_3, Z_3 \xleftarrow{R} \mathbb{G}_{p_3}$. Given a description of $(\mathbb{G}, \mathbb{G}_T)$, a set of group elements (g, X_1X_2, Z_3, Y_2Y_3) and T, it is hard to decide if $T \in_R \mathbb{G}_{p_1p_3}$ or $T \in_R \mathbb{G}$.

Assumption 3 Let $g \xleftarrow{R} \mathbb{G}_{p_1}$, $X_2, Y_2, Z_2 \xleftarrow{R} \mathbb{G}_{p_2}$, $X_3 \xleftarrow{R} \mathbb{G}_{p_3}$ and $\alpha, s \xleftarrow{R} \mathbb{Z}_N$. Given a description of $(\mathbb{G}, \mathbb{G}_T)$, group elements $(g, g^\alpha X_2, X_3, g^s Y_2, Z_2)$ and T, it is infeasible to decide if $T = e(g, g)^{\alpha s}$ or $T \in_R \mathbb{G}_T$.

These assumptions are non-interactive and, in all of them, the number of input elements is constant (*i.e.*, independent of the number of adversarial queries).

3 A Robust Non-Interactive CCA2-Secure Threshold Cryptosystem with Adaptive Corruptions

Our starting point is applying the Canetti-Halevi-Katz [17] transform to a (conceptually equivalent) variant of the Lewko-Waters IBE [34] in the same way as [8] derives a CCA2-secure threshold cryptosystem from the Boneh-Boyen IBE [7]. We show that composite order groups and the techniques of [34] make it possible to handle adaptive corruptions in a relatively simple way and without having to refresh private key shares after each private key operation.

To this end, we apply a modification to the IBE scheme [34][Section 3]. The latter encrypts M under the identity $\mathsf{ID} \in \mathbb{Z}_N$ as $(M \cdot e(g, g)^{\alpha \cdot s}, g^s, (u^{\mathsf{ID}} \cdot v)^s)$ for a random exponent $s \in \mathbb{Z}_N$ and where the public key is $\big(g, u, v, e(g, g)^\alpha\big)$, with $g, u, v \in \mathbb{G}_{p_1}$. We implicitly use an IBE scheme where messages are encrypted as $(M \cdot e(g, h)^{\alpha \cdot s}, g^s, (u^{\mathsf{ID}} \cdot v)^s)$, where $h \neq g$ and $e(g, h)^\alpha$ is part of the public key.

Another difference is that, in order to ensure the consistency of these scheme (as defined in section 2.1), the ciphertext validation algorithm has to reject all ciphertexts containing components in the subgroup $\mathbb{G}_{p_3}$.

3.1 Description

In the description hereafter, the verification key of the one-time signature is interpreted as an element of $\mathbb{Z}_N$. In practice, longer keys can be hashed into $\mathbb{Z}_N$ using a collision-resistant hash function.

Setup(λ, t, n)**:** given a security parameter $\lambda \in \mathbb{N}$ and integers $t, n \in \mathsf{poly}(\lambda)$ (with $1 \leq t \leq n$), the algorithm does the following.

1. Choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of order $N = p_1p_2p_3$, with $p_1, p_2, p_3 > 2^\lambda$.
2. Choose $\alpha \xleftarrow{R} \mathbb{Z}_N$, $g, h, u, v \xleftarrow{R} \mathbb{G}_{p_1}$, $X_{p_3} \xleftarrow{R} \mathbb{G}_{p_3}$ and compute $e(g,h)^\alpha$.
3. Choose a strongly unforgeable one-time signature $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$.
4. Choose a polynomial $P[X] = \alpha + \alpha_1 X + \cdots + \alpha_{t-1}X^{t-1} \in \mathbb{Z}_N[X]$, for random coefficients $\alpha_1, \ldots, \alpha_{t-1} \xleftarrow{R} \mathbb{Z}_N$. Define the public key to be

$$PK = \Big((\mathbb{G}, \mathbb{G}_T),\ N,\ g,\ e(g,h)^\alpha,\ u,\ v,\ X_{p_3},\ \Sigma\Big)$$

and set private key shares $\mathbf{SK} = (SK_1, \ldots, SK_n)$ as $SK_i = h^{P(i)} \cdot Z_{3,i}$, for $i = 1$ to n, with $Z_{3,1}, \ldots, Z_{3,n} \xleftarrow{R} \mathbb{G}_{p_3}$. Verification keys are then set as $\mathbf{VK} = (VK_1, \ldots, VK_n)$ with $VK_i = e(g,h)^{P(i)}$ for $i = 1$ to n.

The public key PK and the verification key $\mathbf{VK}$ are made publicly available while, for each $i \in \{1, \ldots, n\}$, SK_i is given to decryption server i.

Encrypt(PK, m)**:** to encrypt $m \in \mathbb{G}_T$, generate a one-time signature key pair $(\mathsf{SSK}, \mathsf{SVK}) \leftarrow \mathcal{G}(\lambda)$. Choose $s \xleftarrow{R} \mathbb{Z}_N$ and compute

$$C = \big(\mathsf{SVK}, C_0, C_1, C_2, \sigma\big) = \Big(\mathsf{SVK},\ m \cdot e(g,h)^{\alpha \cdot s},\ g^s,\ (u^{\mathsf{SVK}} \cdot v)^s,\ \sigma\Big),$$

where $\sigma = \mathcal{S}(\mathsf{SSK}, (C_0, C_1, C_2))$.

Ciphertext-Verify$\big(PK, C\big)$**:** parse the ciphertext C as $(\mathsf{SVK}, C_0, C_1, C_2, \sigma)$. Return 1 if it holds that $\mathcal{V}(\mathsf{SVK}, (C_0, C_1, C_2), \sigma) = 1$, $e(C_j, X_{p_3}) = 1_{\mathbb{G}_T}$ for $j \in \{1, 2\}$ and $e(g, C_2) = e(C_1, u^{\mathsf{SVK}} \cdot v)$. Otherwise, return 0.

Share-Decrypt(i, SK_i, C)**:** Parse C as $\big(\mathsf{SVK}, C_0, C_1, C_2, \sigma\big)$ and SK_i as an element of $\mathbb{G}$. Return $(i, \perp)$ if **Ciphertext-Verify**$\big(PK, C\big) = 0$. Otherwise, choose $r \xleftarrow{R} \mathbb{Z}_N$, $W_3, W_3' \xleftarrow{R} \mathbb{G}_{p_3}$, compute and return $\mu_i = (i, \hat{\mu_i})$, where

$$\hat{\mu_i} = (D_{i,1}, D_{i,2}) = \big(SK_i \cdot (u^{\mathsf{SVK}} \cdot v)^r \cdot W_3,\ g^r \cdot W_3'\big). \tag{1}$$

Share-Verify$\big(PK, C, (i, \hat{\mu_i})\big)$**:** parse C as $(\mathsf{SVK}, C_0, C_1, C_2, \sigma)$. If $\hat{\mu_i} = \perp$ or $\hat{\mu_i} \notin \mathbb{G}^2$, return 0. Otherwise, parse $\hat{\mu_i}$ as a pair $(D_{i,1}, D_{i,2}) \in \mathbb{G}^2$ and return 1 if $e(D_{i,1}, g) = VK_i \cdot e(u^{\mathsf{SVK}} \cdot v, D_{i,2})$. In any other situation, return 0.

Combine$(PK, C, \{(i, \hat{\mu_i})\}_{i \in S})$**:** for each $i \in S$, parse the share $\hat{\mu_i}$ as $(D_{i,1}, D_{i,2})$ and return $\perp$ if **Share-Verify**$\big(PK, C, (i, \hat{\mu_i})\big) = 0$. Otherwise, compute

$$(D_1, D_2) = \big(\prod_{i \in S} D_{i,1}^{\Delta_{i,S}(0)}, \prod_{i \in S} D_{i,2}^{\Delta_{i,S}(0)}\big) = \big(h^\alpha \cdot (u^{\mathsf{SVK}} \cdot v)^{\tilde{r}} \cdot \tilde{W}_3,\ g^{\tilde{r}} \cdot \tilde{W}_3'\big),$$

for some $\tilde{W}_3, \tilde{W}_3' \in \mathbb{G}_{p_3}$ and $\tilde{r} \in \mathbb{Z}_{p_1}$. Using the pair (D_1, D_2), compute and output the plaintext $m = C_0 \cdot e(C_1, D_1)^{-1} \cdot e(C_2, D_2)$.

As far as efficiency goes, the ciphertext-validity check can be optimized by choosing $\omega_1, \omega_2 \xleftarrow{R} \mathbb{Z}_N$ and checking that $e(g \cdot X_{p_3}^{\omega_1}, C_2) = e(C_1, (u^{\mathsf{SVK}} \cdot v) \cdot X_{p_3}^{\omega_2})$, which rejects ill-formed ciphertexts with overwhelming probability and saves two pairing evaluations. Similar batch verification techniques

apply to simultaneously test t or more decryption shares using only two pairing evaluations[2].

We note that, following Boyen, Mei and Waters [12], it is possible to shorten ciphertexts by eliminating SVK and σ. In this case, C_2 is calculated using a hash value of (C_0, C_1) in place of SVK. Given that C_2 is no longer authenticated by a one-time signature, the security proof then requires an additional step to make sure that adversaries cannot modify the $\mathbb{G}_{p_2}$ component of C_2 in the challenge ciphertext. The details are of this optimized variant are provided in section 4.

We also observe that, as in [8], decryption shares can be seen as signature shares (for a message consisting of the verification key SVK) calculated by decryption servers. In appendix B, we show that the underlying threshold signature is secure against chosen-message attacks in the adaptive corruption scenario.

3.2 Security

The security proof departs from approaches that were previously used in threshold cryptography in that we do not construct an adversary against the centralized version of the scheme out of a CCA2 adversary against its threshold implementation. Instead, we directly prove the security of the latter using the dual encryption paradigm [43, 34].

Our proof proceeds with a sequence of games and uses semi-functional ciphertexts as in [34], and decryption shares. Still, there are two differences. First, two kinds of semi-functional ciphertexts (that differ in the presence of a component of order p_2 in the target group $\mathbb{G}_T$) have to be involved. The second difference is that we need to introduce semi-functional private key shares at some step of the proof and argue that they cannot be distinguished from real key shares. The proof takes advantage of the fact that, at each step of the sequence, the simulator knows either the $\mathbb{G}_{p_1}$ components of private key shares $\{h^{P(i)}\}_{i=1}^n$ or a "blinded" version $\{h^{P(i)} \cdot Z_{2,i}\}_{i=1}^n$ of those shares, for some $Z_{2,i} \in_R \mathbb{G}_{p_2}$, which suffices to consistently answer adaptive corruption queries.

Theorem 1. *The scheme is IND-CCA2 against adaptive corruptions assuming that Assumption 1, Assumption 2 and Assumption 3 all hold and that Σ is a strongly unforgeable[3] one-time signature.*

Proof. The proof proceeds using a sequence of games including steps similar to [34] and additional steps. As in [43, 34], the proof makes use of semi-functional ciphertexts and decryption shares (which are actually private keys in [34]). In addition, we also have to consider semi-functional private key shares. Another difference is that we need two kinds of semi-functional ciphertexts.

- Semi-functional ciphertexts of Type I are generated from a normal ciphertext (C_0', C_1', C_2') and some $g_2 \in \mathbb{G}_{p_2}$, by choosing random $\tau, z_c \xleftarrow{R} \mathbb{Z}_N$ and setting

$$C_0 = C_0', \qquad C_1 = C_1' \cdot g_2^{\tau}, \qquad C_2 = C_2' \cdot g_2^{\tau z_c}.$$

- Semi-functional ciphertexts of Type II are generated from a normal ciphertext (C_0', C_1', C_2') by choosing random $\tau, z_c, \theta \xleftarrow{R} \mathbb{Z}_N$ and setting

$$C_0 = C_0' \cdot e(g_2, g_2)^{\theta}, \qquad C_1 = C_1' \cdot g_2^{\tau}, \qquad C_2 = C_2' \cdot g_2^{\tau z_c}.$$

[2] Namely, t shares $\{\mu_i = (D_{i,1}, D_{i,2})\}_{i=1}^t$ can be batch-verified by drawing $\omega_1, \ldots, \omega_t \xleftarrow{R} \mathbb{Z}_N$ and testing if $e\big(g, \prod_{i=1}^t D_{i,1}^{\omega_i}\big) = \prod_{i=1}^t VK_i^{\omega_i} \cdot e\big(u^{\mathsf{SVK}} \cdot v, \prod_{i=1}^t D_{i,2}^{\omega_i}\big)$.

[3] Strong unforgeability refers to the infeasibility, after having obtained a message-signature pair (M, σ), of computing a new pair $(M^\star, \sigma^\star) \neq (M, \sigma)$.

- Semi-functional decryption shares are obtained from a normal decryption share $(D'_{i,1}, D'_{i,2})$ by picking $\gamma, z_k \xleftarrow{R} \mathbb{Z}_N$, $W_3, W'_3 \xleftarrow{R} \mathbb{G}_{p_3}$ and setting

$$D_{i,1} = D'_{i,1} \cdot g_2^{\gamma z_k} \cdot W_3, \qquad D_{i,2} = D'_{i,2} \cdot g_2^{\gamma} \cdot W'_3.$$

- Semi-functional private key shares $\{SK_i\}_{i=1}^n$ are obtained from normal shares $\{SK'_i\}_{i=1}^n$ by setting $SK_i = SK'_i \cdot Z_{2,i}$, where $Z_{2,i} \xleftarrow{R} \mathbb{G}_{p_2}$, for $i = 1$ to n.

The proof considers a sequence of $q + 6$ games. It starts with the real game Game_{real} followed by $\mathsf{Game}_{restricted}$, $\mathsf{Game}^*_{restricted}$ $\mathsf{Game}_0, \mathsf{Game}_1, \ldots, \mathsf{Game}_q$ and finally Game^*_q and Game_{final}.

$\mathsf{Game}_{restricted}$: is identical to Game_{real} with the difference that the challenger $\mathcal{B}$ rejects all post-challenge decryption queries $(\mathsf{SVK}, C_0, C_1, C_2, \sigma)$ for which $\mathsf{SVK} = \mathsf{SVK}^\star$, where $\mathsf{SVK}^\star$ denotes the one-time verification key included in the challenge ciphertext.

$\mathsf{Game}^*_{restricted}$: is identical to $\mathsf{Game}_{restricted}$ with the difference that the adversary $\mathcal{A}$ is not allowed to make decryption queries $(\mathsf{SVK}, C_0, C_1, C_2, \sigma)$ for which $\mathsf{SVK} = \mathsf{SVK}^\star \bmod p_2$.

Game_0: is identical to $\mathsf{Game}^*_{restricted}$ but the normal challenge ciphertext is replaced by a semi-functional ciphertext of Type I.

Game_k $(1 \leq k \leq q)$: in this game, the challenge ciphertext is a semi-functional ciphertext of Type I and the challenger $\mathcal{B}$ answers the first k decryption queries by returning semi-functional decryption shares. As for the last $q - k$ decryption queries, they are answered using normal decryption shares.

Game^*_q: is identical to Game_q with the following two differences.

- All private key shares are made semi-functional and thus contain a random $\mathbb{G}_{p_2}$ component.
- The Type I semi-functional challenge ciphertext is traded for a semi-functional ciphertext of Type II.

Game_{final}: is as Game^*_q but the Type II semi-functional challenge ciphertext is replaced by a semi-functional encryption of a random plaintext (instead of M_β). In this game, $\mathcal{A}$ has no information on the challenger's bit $\beta \in \{0, 1\}$ and cannot guess it with better probability than $1/2$.

As in [34], when a semi-functional decryption share is used (in combination with $t - 1$ normal decryption shares) to decrypt a semi-functional ciphertext, decryption only works when $z_k = z_c$, in which case the decryption share is called *nominally* semi-functional. For each $k \in \{1, \ldots, q\}$, the transitions between Game_{k-1} and Game_k is done in such a way that the distinguisher cannot directly decide (*i.e.*, without interacting with $\mathcal{A}$) whether the k^{th} decryption share is normal or semi-functional by generating this share for the challenge verification key $\mathsf{SVK}^\star$. Indeed, in such an attempt, the generated decryption share is necessarily either normal or nominally semi-functional, so that decryption succeeds either way.

Moreover, during the transition between Game_q and Game^*_q, we have to make sure that the distinguisher cannot bypass its interaction with the adversary and try to distinguish the two games by itself either. Should it attempt to decrypt the challenge ciphertext using the private key shares, the transition is organized in such a way that decryption succeeds regardless of whether the private key shares (resp. the challenge ciphertext) are normal or semi-functional (resp. semi-functional of Type I or II).

The proof is completed by lemma 1 to 6, which show that all games are computationally indistinguishable as long as the one-time signature is strongly unforgeable and Assumptions 1, 2, 3 hold. □

Lemma 1. *If the one-time signature Σ is strongly unforgeable,* Game_{real} *and* $\mathsf{Game}_{restricted}$ *are indistinguishable.*

Proof. The proof uses the classical argument saying that the only way for the adversary to create a legal decryption query $(\mathsf{SVK}^\star, C_0, C_1, C_2, \sigma)$ after the challenge phase is to break the strong unforgeability of Σ. Moreover, the challenge one-time verification key can be defined at the very beginning of the game. Hence, computing a valid pre-challenge decryption query involving $\mathsf{SVK}^\star$ would require the adversary to compute a valid signature without having seen a single signature (or even the verification key) and *a fortiori* break the security of Σ. □

Lemma 2. *Provided Assumption 1 and Assumption 2 both hold,* $\mathsf{Game}_{restricted}$ *and* $\mathsf{Game}^*_{restricted}$ *are indistinguishable.*

Proof. The proof is identical to the one of lemma 5 in [34]. Namely, the only situation where the two games are distinguishable is when the adversary $\mathcal{A}$ manages to come up with a ciphertext for which $\mathsf{SVK}^\star \neq \mathsf{SVK}$ but $\mathsf{SVK}^\star = \mathsf{SVK} \bmod p_2$. In this case, the challenger $\mathcal{B}$ can compute $\gcd(\mathsf{SVK} - \mathsf{SVK}^\star, N)$, which is necessarily a non-trivial factor of N. Depending on which factor is found, $\mathcal{B}$ can break either Assumption 1 or Assumption 2. □

The proofs of lemma 3 and 4 proceed exactly as in [34] and we give them in appendix A for completeness.

Lemma 3. *Under Assumption 1, no PPT adversary can distinguish* $\mathsf{Game}^*_{restricted}$ *and* Game_0.

Lemma 4. *Under Assumption 2, no PPT adversary can distinguish* Game_k *from* Game_{k-1} *for* $1 \leq k \leq q$.

In comparison with the security proof of [34], the novelty is the transition from Game_q to Game^*_q, which is addressed in lemma 5. This transition allows turning all private key shares into semi-functional shares in one step.

Lemma 5. *Under Assumption 2,* Game_q *and* Game^*_q *are indistinguishable.*

Proof. Towards a contradiction, we assume that a PPT adversary $\mathcal{A}$ can tell apart Game_q and Game^*_q. We construct an algorithm $\mathcal{B}$ that, given elements $(g, X_3, X_1X_2, Y_2Y_3, T)$, decides if $T \in_R \mathbb{G}_{p_1p_3}$ or $T \in_R \mathbb{G}$.

Algorithm $\mathcal{B}$ prepares PK by setting $X_{p_3} = X_3$, $u = g^a$ and $v = g^b$ with $a, b \xleftarrow{R} \mathbb{Z}_N$. It also picks $\alpha \xleftarrow{R} \mathbb{Z}_N$ and defines $e(g,h)^\alpha = e(g,T)^\alpha$. In addition, $\mathcal{B}$ chooses a random polynomial $P[X]$ of degree $t-1$ such that $P(0) = \alpha$. To prepare $\mathbf{SK} = (SK_1, \ldots, SK_n)$ and $\mathbf{VK} = (VK_1, \ldots, VK_n)$, $\mathcal{B}$ sets $SK_i = T^{P(i)} \cdot Z_{3,i}$, with $Z_{3,i} \xleftarrow{R} \mathbb{G}_{p_3}$, and $VK_i = e(g, SK_i)$ for $i = 1$ to n.

In the challenge phase, $\mathcal{A}$ outputs M_0, M_1 and $\mathcal{B}$ flips a coin $\beta \xleftarrow{R} \{0,1\}$. It generates a one-time signature key pair $(\mathsf{SSK}^\star, \mathsf{SVK}^\star) \leftarrow \mathcal{G}(\lambda)$ and computes

$$C_0^\star = M_\beta \cdot e(X_1X_2, T)^\alpha, \qquad C_1^\star = X_1X_2, \qquad C_2^\star = (X_1X_2)^{a \cdot \mathsf{SVK}^\star + b} \tag{2}$$

To generate a decryption share on behalf of decryption server i for a ciphertext $(\mathsf{SVK}, C_0, C_1, C_2, \sigma)$, $\mathcal{B}$ chooses $r, w, w' \xleftarrow{R} \mathbb{Z}_N$ and generates a semi-functional decryption share

$$(D_{i,1}, D_{i,2}) = (SK_i \cdot (u^{\mathsf{SVK}} \cdot v)^r \cdot (Y_2Y_3)^w,\ g^r \cdot (Y_2Y_3)^{w'}). \tag{3}$$

Whenever $\mathcal{A}$ decides to corrupt decryption server i, $\mathcal{B}$ simply reveals SK_i.

We note that, in the situation where $T \in \mathbb{G}_{p_1p_3}$, $\mathcal{B}$ is clearly playing Game_q. Now, let us consider what happens when $T \in_R \mathbb{G}$. In this case, T can be written as $T = g^{\gamma_1} g_2^{\gamma_2} g_3^{\gamma_3}$ for some random $\gamma_1 \in \mathbb{Z}_{p_1}$, $\gamma_2 \in \mathbb{Z}_{p_2}$, $\gamma_3 \in \mathbb{Z}_{p_3}$ and h is implicitly set as $h = g^{\gamma_1}$. From the simulator's standpoint, the $\mathbb{G}_{p_2}$ components of private key shares $\{SK_i\}_{i=1}^n$ are not independent since a polynomial of degree $t-1$ goes through them in the exponent: for each index $i \in \{1, \ldots, n\}$, we indeed have $SK_i = g^{\gamma_1 \cdot P(i)} \cdot g_2^{\gamma_2 \cdot P(i)} \cdot \tilde{Z}_{3,i}$, for some $\tilde{Z}_{3,i} \in_R \mathbb{G}_{p_3}$. In addition, if we write $X_1X_2 = g^s \cdot g_2^{\tau}$, for some $s \in \mathbb{Z}_{p_1}$, $\tau \in \mathbb{Z}_{p_2}$, the components

$$(C_0^\star, C_1^\star, C_2^\star) = \left(M_\beta \cdot e(g,h)^{\alpha \cdot s} \cdot e(g_2, g_2)^{\gamma_2 \cdot \tau \cdot \alpha},\ g^s \cdot g_2^{\tau},\ (u^{\mathsf{SVK}^\star} \cdot v)^s \cdot g_2^{\tau z_c}\right)$$

of the challenge ciphertext information-theoretically reveal $e(g_2, g_2)^{\gamma_2 \cdot \tau \cdot P(0)}$.

However, the public key does not reveal anything about $\alpha \bmod p_2$ and the distribution of $\mathbf{VK}$ is uncorrelated to $\{P(i) \bmod p_2\}_{i=1}^n$. Moreover, as decryption shares are generated as per (3), they perfectly hide $P(i) \bmod p_2$ in the first term SK_i of the product $D_{i,1}$. Hence, since the adversary $\mathcal{A}$ cannot obtain more than $t-1$ private key shares throughout the game, the correlation between the $\mathbb{G}_{p_2}$ components of $\{SK_i\}_{i=1}^n$ is information-theoretically hidden to $\mathcal{A}$. Indeed, given that $P[X] \in \mathbb{Z}_N[X]$ is chosen as a random polynomial of degree $t-1$, its evaluations $P(i) \bmod p_2$ are t-wise independent. In other words, for any $(t-1)$-subset $\mathcal{C} \subset \{1, \ldots, n\}$ chosen by $\mathcal{A}$, the values $\{g_2^{\gamma_2 \cdot P(i)}\}_{i \in \mathcal{C} \cup \{0\}}$ are statistically indistinguishable from a set of t random elements of $\mathbb{G}_{p_2}$. This means that, from $\mathcal{A}$'s view, $(C_0^\star, C_1^\star, C_2^\star)$ and $\{SK_i\}_{i \in \mathcal{C}}$ look like a Type II semi-functional ciphertext and a set of $t-1$ semi-functional private key shares, respectively. We conclude that, if $T \in_R \mathbb{G}$, the simulator $\mathcal{B}$ is actually playing Game_q^* with $\mathcal{A}$. □

In the proof of lemma 5, we note that $\mathcal{B}$ cannot distinguish $T \in_R \mathbb{G}_{p_1p_3}$ from $T \in_R \mathbb{G}$ by itself: if $\mathcal{B}$ tries to decrypt the challenge ciphertext (given by (2)) using the decryption shares $\{SK_i\}_{i=1}^n$, decryption recovers M_β in either case.

Lemma 6. *Under Assumption 3, no PPT adversary can distinguish* Game_q^* *from* Game_{final} *(the proof is deferred to the appendix A.3).*

Obviously, concealing the factorization of $N = p_1p_2p_3$ is crucial for the proof of adaptive security. Nevertheless, even if p_1, p_2 and p_3 are somehow revealed, the scheme can still be proved secure under static corruptions (using the same proof as [8]) under the Decision Bilinear Diffie-Hellman assumption [7] in $\mathbb{G}_{p_1}$.

Unlike [40, 8], where consistency holds statistically, we demonstrate consistency in the computational sense.

Theorem 2. *The scheme provides consistency if Assumption 1 holds.*

Proof. We describe an algorithm $\mathcal{B}$ breaking Assumption 1 using an adversary $\mathcal{A}$ against the consistency of the scheme.

Algorithm $\mathcal{B}$ takes as input the description of $(\mathbb{G}, \mathbb{G}_T)$ and elements (g, X_3) with the task of deciding if $T \in_R \mathbb{G}_{p_1}$ or $T \in_R \mathbb{G}_{p_1p_2}$. To this end, it begins by generating a public key exactly as in the proof of lemma 3. This implies that $\mathcal{B}$ knows $a, b \in \mathbb{Z}_N$ such that $u = g^a, v = g^b$ as well as the polynomial $P[X]$ (and in particular $\alpha = P(0)$), which allows answering all adversarial queries. At the end of the game, the adversary $\mathcal{A}$ is assumed to output a ciphertext $C = (\mathsf{SVK}, C_0, C_1, C_2, \sigma)$ such that $\mathbf{Ciphertext\text{-}Verify}(PK, C) = 1$ as well as two t-sets of valid decryption shares Φ, Φ' for

which $\mathbf{Combine}(PK, \mathbf{VK}, C, \Phi)$ and $\mathbf{Combine}(PK, \mathbf{VK}, C, \Phi')$ result in different plaintexts.

We remark that the ciphertext sanity check always rejects ciphertexts containing $\mathbb{G}_{p_3}$ components in C_1 and C_2. Hence, it is easy to see that the above situation can only occur if non-trivial $\mathbb{G}_{p_2}$ components appear in C_1 and/or C_2 as well as in at least one of the shares of Φ and Φ'. This means that $\eta = C_2/C_1^{a \cdot \mathsf{SVK}+b}$ is in $\mathbb{G}_{p_2}$ with overwhelming probability. Indeed, since the public key does not reveal anything about $a \bmod p_2$ and $b \bmod p_2$, we can only have $C_1 = g^s \cdot g_2^{\tau}$, $C_2 = (u^{\mathsf{SVK}} \cdot v)^s \cdot g_2^{\tau(a \cdot \mathsf{SVK}+b)}$ with negligible chance. Using $\eta \in \mathbb{G}_{p_2}$, $\mathcal{B}$ can break Assumption 1 since $e(T, \eta) \neq 1_{\mathbb{G}_T}$ if $T \in_R \mathbb{G}_{p_1 p_2}$. □

4 A Variant with Shorter Ciphertexts

It is tempting to use optimizations of the CHK paradigm to compress ciphertexts by eliminating the one-time signature and its verification key. To this end, it is possible to adapt the non-generic approach of Boyen, Mei and Waters [12].

The construction is similar to the one of [12][Section 3.1] in its thresholdized version. The intuition of that scheme is to use a hash value of ciphertext components $(C_0, C_1) = (M \cdot e(g,h)^{\alpha \cdot s}, g^s)$ as an "identity" for the underlying IBE scheme during the computation of the third ciphertext component. The main difference is that the use of composite order groups allows dispensing with the long public key (inherited from Waters' IBE scheme [42]) consisting of $O(\lambda)$ group elements.

In the system hereafter, the ciphertext overhead reduces to two elements in the subgroup $\mathbb{G}_{p_1}$ of the composite order group. Ciphertexts have the form $(C_0, C_1, C_2) = (M \cdot e(g,h)^{\alpha \cdot s}, g^s, (u^{\kappa} \cdot v)^s)$, where $\kappa = H(C_0, C_1)$. However, we have to deal with an additional difficulty which is inherent to the use of groups of composite order. Since the one-time signature was eliminated, there is no way to "authenticate" C_2 and prevent the adversary from re-randomizing the challenge ciphertext by introducing a $\mathbb{G}_{p_2}$ component in C_2 by itself. This can be solved by: (1) exploiting the ciphertext-validity check that rejects all ciphertexts where (C_1, C_2) have a $\mathbb{G}_{p_3}$ component (in this case, this check also comes into play in the proof of CCA2 security and not only in the proof of consistency); (2) adding one step in the security proof and show that, in the challenge ciphertext, the adversary cannot re-randomize the $\mathbb{G}_{p_2}$ part of C_2 without breaking some intractability assumption.

Setup(λ, t, n)**:** given a security parameter $\lambda \in \mathbb{N}$ and integers $t, n \in \mathsf{poly}(\lambda)$ such that $1 \leq t \leq n$, do the following.

1. Choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of composite order $N = p_1 p_2 p_3$, where $p_i > 2^{\lambda}$ for each $i \in \{1, 2, 3\}$.
2. Pick $\alpha \xleftarrow{R} \mathbb{Z}_N$, $g, h, u, v \xleftarrow{R} \mathbb{G}_{p_1}$, $X_{p_3} \xleftarrow{R} \mathbb{G}_{p_1}$ and compute $e(g,h)^{\alpha}$.
3. Choose a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_N$.
4. Define a polynomial $P[X] = \alpha + \alpha_1 X + \cdots + \alpha_{t-1} X^{t-1} \in \mathbb{Z}_N[X]$, for random coefficients $\alpha_1, \ldots, \alpha_{t-1} \xleftarrow{R} \mathbb{Z}_N$. Set the public key as

$$PK = \Big((\mathbb{G}, \mathbb{G}_T),\ N,\ g,\ e(g,h)^{\alpha},\ u,\ v,\ X_{p_3},\ H \Big)$$

and set private key shares $\mathbf{SK} = (SK_1, \ldots, SK_n)$ as $SK_i = h^{P(i)} \cdot Z_{3,i}$, where $Z_{3,i} \xleftarrow{R} \mathbb{G}_{p_3}$, for $i = 1$ to n. Verification keys $\mathbf{VK} = (VK_1, \ldots, VK_n)$ are then defined as $VK_i = e(g,h)^{P(i)}$ for $i = 1$ to n.

The public key PK and the verification key **VK** are publicized. For each $i \in \{1, \ldots, n\}$, the share SK_i is given to decryption server i.

Encrypt(PK, m)**:** to encrypt $m \in \mathbb{G}_T$, choose $s \xleftarrow{R} \mathbb{Z}_N$ and compute

$$C = \big(C_0, C_1, C_2\big) = \Big(m \cdot e(g,h)^{\alpha \cdot s},\ g^s,\ (u^\kappa \cdot v)^s \Big),$$

where $\kappa = H(C_0, C_1) \in \mathbb{Z}_N$.

Ciphertext-Verify$\big(PK, C\big)$**:** parse C as (C_0, C_1, C_2) and compute the hash value $\kappa = H(C_0, C_1)$. Return 1 if the equalities $e(g, C_2) = e(C_1, u^\kappa \cdot v)$ and $e(C_j, X_{p_3}) = 1_{\mathbb{G}_T}$ for $j \in \{1, 2\}$ both hold. Otherwise, return 0.

Share-Decrypt(i, SK_i, C)**:** Parse C as $\big(C_0, C_1, C_2\big)$ and SK_i as an element of $\mathbb{G}$. Return $(i, \perp)$ if **Ciphertext-Verify**$\big(PK, C\big) = 0$. Otherwise, choose $r \xleftarrow{R} \mathbb{Z}_N$, $W_3, W_3' \xleftarrow{R} \mathbb{G}_{p_3}$ and compute the decryption share

$$\hat{\mu_i} = (D_{i,1}, D_{i,2}) = \big(SK_i \cdot (u^\kappa \cdot v)^r \cdot W_3,\ g^r \cdot W_3'\big), \tag{4}$$

where $\kappa = H(C_0, C_1)$. Return $\mu_i = (i, \hat{\mu_i})$.

Share-Verify$\big(PK, C, (i, \hat{\mu_i})\big)$**:** parse C as (C_0, C_1, C_2). If $\hat{\mu_i} = \perp$ or $\hat{\mu_i} \notin \mathbb{G}^2$, return 0. Otherwise, parse $\hat{\mu_i}$ as $(D_{i,1}, D_{i,2}) \in \mathbb{G}^2$. Compute $\kappa = H(C_0, C_1)$ and return 1 if

$$e(D_{i,1}, g) = VK_i \cdot e(u^\kappa \cdot v, D_{i,2}).$$

Otherwise, return 0.

Combine$(PK, C, \{(i, \hat{\mu_i})\}_{i \in S})$**:** parse C as (C_0, C_1, C_2). For each $i \in S$, parse $\hat{\mu_i}$ as $(D_{i,1}, D_{i,2})$ and return $\perp$ if **Share-Verify**$\big(PK, C, (i, \hat{\mu_i})\big) = 0$. Otherwise, compute

$$(D_1, D_2) = \big(\prod_{i \in S} D_{i,1}^{\Delta_{i,S}(0)}, \prod_{i \in S} D_{i,2}^{\Delta_{i,S}(0)}\big)$$

and output the plaintext $m = C_0 \cdot e(C_1, D_1)^{-1} \cdot e(C_2, D_2)$.

The proof of the following theorem explains how the security proof of the scheme in section 3 can be modified to establish the security of the above system.

Theorem 3. *The scheme is IND-CCA2 secure against adaptive corruptions assuming that Assumption 1, Assumption 2 and Assumption 3 all hold and that H is a collision-resistant hash function.*

Proof. The proof follows the one of theorem 1 and we only outline the changes. Semi-functional ciphertexts, decryption shares and private key shares have exactly the same shape as in the scheme of section 3.

The proof considers a sequence of $q + 8$ games. Throughout this sequence, $C^\star = (C_0^\star, C_1^\star, C_2^\star)$ will always denote the challenge ciphertext and $\kappa^\star$ will stand for the corresponding hash value $\kappa^\star = H(C_0^\star, C_1^\star)$.

Game_{real}**:** proceeds like the real game. At the end of the game, the adversary $\mathcal{A}$ outputs a bit $\beta' \in \{0, 1\}$ and challenger outputs 1 if and only if $\beta' = \beta$.

$\mathsf{Game}_{restricted}$**:** is as Game_{real} with the difference that, in the challenge phase, the challenger halts and outputs 1 in the event that the adversary $\mathcal{A}$ previously made a decryption query (C_0, C_1, C_2) for which $C_1 = C_1^\star$.

$\mathsf{Game}^*_{restricted}$**:** is identical to $\mathsf{Game}_{restricted}$ but the challenger $\mathcal{B}$ rejects all post-challenge decryption queries (C_0, C_1, C_2) for which $(C_0, C_1) \neq (C_0^\star, C_1^\star)$ and $\kappa = H(C_0, C_1) = H(C_0^\star, C_1^\star) = \kappa^\star$.

$\mathsf{Game}^{**}_{restricted}$**:** is like $\mathsf{Game}^*_{restricted}$ but the adversary $\mathcal{A}$ is now disallowed to make post-challenge decryption queries (C_0, C_1, C_2) for which $\kappa = \kappa^\star \bmod p_2$ (although $\kappa \neq \kappa$), where $\kappa = H(C_0, C_1)$ and $\kappa^\star = H(C_0^\star, C_1^\star)$.

$\mathsf{Game}^{***}_{restricted}$**:** is as $\mathsf{Game}^{**}_{restricted}$ with one difference. Namely, the challenger $\mathcal{B}$ halts and outputs 1 if the adversary $\mathcal{A}$ manages to make a decryption query $C = (C_0, C_1, C_2)$ such that **Ciphertext-Verify**$(PK, C) = 1$ and for which $(C_0, C_1) = (C_0^\star, C_1^\star)$ and $C_2 \neq C_2^\star$.

Game_0**:** is identical to $\mathsf{Game}^{***}_{restricted}$ but the normal challenge ciphertext is turned into a semi-functional ciphertext of Type I.

Game_k $(1 \leq k \leq q)$**:** in this game, the challenge ciphertext is a Type I semi-functional ciphertext and the challenger $\mathcal{B}$ answers the first k decryption queries by outputting semi-functional decryption shares. The last $q - k$ decryption queries are normal.

Game^*_q**:** is like Game_q with two differences.

- All private key shares are made semi-functional and henceforth contain a random $\mathbb{G}_{p_2}$ component.
- The challenge ciphertext becomes a Type II (instead of Type I) semi-functional ciphertext.

Game_{final}**:** is as Game^*_q but the Type II semi-functional challenge ciphertext is now replaced by a semi-functional encryption of a random plaintext (rather than M_β). In this game, $\mathcal{A}$ the challenger's bit $\beta \in \{0, 1\}$ is perfectly independent of $\mathcal{A}$'s view and $\mathcal{A}$ can only guess it with probability 1/2.

It is easy to see that Game_{real} and $\mathsf{Game}_{restricted}$ are negligibly far apart since, until the challenge phase, $C_1^\star = g^s$ is independent of $\mathcal{A}$'s view. In $\mathsf{Game}_{restricted}$, the probability that the challenger halts in the challenge phase is at most q/p_1, which is negligible.

In $\mathsf{Game}^*_{restricted}$, the challenger has negligible chance of rejecting a ciphertext that would not have been rejected in $\mathsf{Game}_{restricted}$ as long as H is collision-resistant. As for the transition between $\mathsf{Game}^*_{restricted}$ and $\mathsf{Game}^{**}_{restricted}$, the indistinguishability of the two games is proved (under Assumption 1 and Assumption 2) exactly in the same way as in the proof of lemma 2.

The main difference with the proof of theorem 1 is the additional step proving the indistinguishability $\mathsf{Game}^{**}_{restricted}$ and $\mathsf{Game}^{***}_{restricted}$ in lemma 7 and all subsequent transitions then proceed as in the proof of theorem 1. □

Lemma 7. *As long as Assumption 1 holds, no PPT adversary can distinguish* $\mathsf{Game}^{**}_{restricted}$ *from* $\mathsf{Game}^{***}_{restricted}$.

Proof. We show that, if the adversary has significantly different behaviors in $\mathsf{Game}^{**}_{restricted}$ and $\mathsf{Game}^{***}_{restricted}$, there exists a PPT distinguisher $\mathcal{B}$ that breaks Assumption 1. This algorithm $\mathcal{B}$ takes as input (g, X_3, T) and decides if $T \in_R \mathbb{G}_{p_1}$ or $T \in_R \mathbb{G}_{p_1 p_2}$ using its interaction with the adversary.

Algorithm $\mathcal{B}$ generates the public key PK by choosing $h \xleftarrow{R} \mathbb{G}_{p_1}$ and setting $e(g, h)^\alpha$, $X_{p_3} = X_3$, $u = g^a$, $v = g^b$. It randomly chooses a polynomial $P[X]$ of degree $t - 1$ such that $P(0) = \alpha$ and

defines $SK_i = h^{P(i)} \cdot Z_{3,i}$, with $Z_{3,i} \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}$, for $i = 1$ to n. Decryption queries and private key share queries are processed by following exactly the specification of the scheme.

When $\mathcal{A}$ enters the challenge phase, it outputs $M_0, M_1 \in \mathbb{G}_T$ and $\mathcal{B}$ flips a coin $\beta \stackrel{R}{\leftarrow} \{0,1\}$ before returning a normal encryption of M_β. We denote by

$$C^\star = (C_0^\star, C_1^\star, C_2^\star) = (M \cdot e(g,h)^{\alpha \cdot s}, g^s, (u^{\kappa^\star} \cdot v)^s)$$

the resulting challenge ciphertext, where $\kappa^\star = H(C_0^\star, C_1^\star)$.

By hypothesis, $\mathcal{A}$ is able to notice the difference between the two games with non-negligible probability. However, the only situation where $\mathsf{Game}^{***}_{restricted}$ departs from $\mathsf{Game}^{**}_{restricted}$ is when $\mathcal{A}$ queries the decryption oracle with a valid ciphertext (C_0, C_1, C_2) such that $(C_0, C_1) = (C_0^\star, C_1^\star)$ and $C_2 \neq C_2^\star$. Since $e(g, C_2)$ equals $e(C_1, u^{\kappa^\star} \cdot v) = e(C_1^\star, u^{\kappa^\star} \cdot v) = e(g, C_2^\star)$, this necessarily means that C_2 and $C_2^\star$ have the same $\mathbb{G}_{p_1}$ component and only differ in that C_2 has a non-trivial component in $\mathbb{G}_{p_2}$ (recall that the ciphertext validation algorithm rules out the presence of a $\mathbb{G}_{p_3}$ component in C_2). Hence, our distinguisher $\mathcal{B}$ can compute $\eta = C_2/C_2^\star \in \mathbb{G}_{p_2}$, which allows deciding whether $T \in \mathbb{G}_{p_1}$ or $T \in \mathbb{G}_{p_1 p_2}$ (since we only have $e(T, \eta) \neq 1_{\mathbb{G}_T}$ in the latter case).

At the end of the game, $\mathcal{A}$ outputs a bit $\beta' \in \{0,1\}$ and $\mathcal{B}$ outputs 1 if $\beta' = \beta$. Otherwise, it outputs 0. □

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, 2006.
4. S. Arita, K. Tsurudome. Construction of Threshold Public-Key Encryptions through Tag-Based Encryptions. In *ACNS'09*, *LNCS* 5536, 2009.
5. N. Attrapadung, B. Libert. Homomorphic Network Coding Signatures in the Standard Model. In *PKC'11*, *LNCS* series, to appear, 2011.
6. M. Bellare, P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, 1993.
7. D. Boneh, X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Eurocrypt'04*, *LNCS* 3027, 2004.
8. D. Boneh, X. Boyen, S. Halevi. Chosen Ciphertext Secure Public Key Threshold Encryption Without Random Oracles. In *CT-RSA'06*, *LNCS* 3860, 2006.
9. 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*.
10. D. Boneh, M. Franklin. Efficient Generation of Shared RSA Keys. In *Crypto'97*, pp. 425–439, *LNCS* 1924, 1997.
11. C. Boyd. Digital Multisignatures. In *Cryptography and Coding* (H.J. Beker and F.C. Piper Eds.), Oxford University Press, pp. 241–246, 1989.
12. X. Boyen, Q. Mei, B. Waters. Direct Chosen Ciphertext Security from Identity-Based Techniques. in *ACM CCS'05*, 2005.
13. R. Canetti, O. Goldreich, S. Halevi. The random oracle methodology, revisited. Journal of the ACM 51(4), pp. 557–594, 2004. Earlier version in *STOC'98*, 1998.
14. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Adaptive Security for Threshold Cryptosystems. In *Crypto'99*, *LNCS* 1666, 1999.
15. R. Canetti, S. Goldwasser. An Efficient Threshold Public Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack. In *Eurocrypt'99*, *LNCS* 1592, 1999.
16. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Crypto'98*, LNCS 1462, 1998.

17. R. Canetti, S. Halevi, J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In *Eurocrypt'04*, *LNCS* 3027, 2004.
18. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, T. Rabin. Efficient Multi-Party Computations Secure Against an Adaptive Adversary. In *Eurocrypt'99*, *LNCS* 1592, 1999.
19. V. Daza, J. Herranz, P. Morillo, C. Ràfols. CCA2-Secure Threshold Broadcast Encryption with Shorter Ciphertexts. In *ProvSec'07*, *LNCS* 4784, 2007.
20. C. Delerablée, D. Pointcheval. Dynamic Threshold Public-Key Encryption. In *Crypto'08*, *LNCS* 5157, pp. 317–334, 2008.
21. Y. Desmedt. Society and Group Oriented Cryptography: A New Concept. In *Crypto'87*, *LNCS* 293, 1987.
22. Y. Desmedt, Y. Frankel. Threshold Cryptosystems. In *Crypto'89*, *LNCS* 435, 1989.
23. Y. Dodis, J. Katz. Chosen-Ciphertext Security of Multiple Encryption. In *TCC'05*, *LNCS* 3378, 2005.
24. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Crypto'84*, *LNCS 196*, Springer, 1985.
25. S. Goldwasser, S. Micali, R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.* 17(2), pp. 281–308, 1988.
26. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In *Eurocrypt'99*, *LNCS* 1592, 2002.
27. P.-A. Fouque, D. Pointcheval. Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. In *Asiacrypt'01*, *LNCS* 2248, 2001.
28. Y. Frankel, P. MacKenzie, M. Yung. Adaptively-Secure Distributed Public-Key Systems. In *ESA'99*, *LNCS* 1643, 1999.
29. Y. Frankel, P. MacKenzie, M. Yung. Adaptively-Secure Optimal-Resilience Proactive RSA. In *Asiacrypt'99*, *LNCS* 1716, 1999.
30. D. Freeman. Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups. In *Eurocrypt'10*, *LNCS* 6110, pp. 44–61, 2010.
31. S. Jarecki, A. Lysyanskaya. Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures. In *Eurocrypt'00*, *LNCS* 1807, 2000.
32. E. Kiltz. Chosen-ciphertext security from tag-based encryption. In *TCC'06*, LNCS 3876, 2006.
33. A. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In *Eurocrypt 2010*, LNCS 6110, Springer, 2010.
34. A. Lewko, B. Waters. New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In *TCC 2010*, LNCS 5978, Springer, 2010.
35. A. Lysyanskaya, C. Peikert. Adaptive Security in the Threshold Setting: From Cryptosystems to Signature Schemes. In *Asiacrypt'01*, *LNCS* 2248, 2001.
36. P. MacKenzie. An Efficient Two-Party Public Key Cryptosystem Secure against Adaptive Chosen Ciphertext Attack. In *PKC'03*, LNCS 2567, Springer, 2003.
37. R. Ostrovsky, M. Yung. How to Withstand Mobile Virus Attacks. In 10^{th} *ACM Symp. on Principles of Distributed Computing (PODC'91)*, 1991.
38. B. Qin, Q. Wu, L. Zhang, J. Domingo-Ferrer. Threshold Public-Key Encryption with Adaptive Security and Short Ciphertexts. In *ICICS'10*, *LNCS* 6476, 2010.
39. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Crypto'84*, *LNCS* 196, 1984.
40. V. Shoup, R. Gennaro. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In *Eurocrypt'98*, *LNCS* 1403, 1998.
41. V. Shoup. Practical Threshold Signatures. In *Eurocrypt'00*, *LNCS* 1807, 2000.
42. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Eurocrypt'05*, *LNCS* 3494, 2005.
43. B. Waters. Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In *Crypto'09*, *LNCS* 5677, 2009.

A Proof of Lemmas 3, 4 and 6

A.1 Proof of Lemma 3

As in [34], we show that, if the adversary has non-negligible chance of distinguishing $\mathsf{Game}^*_{Restricted}$ and Game_0, there is an algorithm $\mathcal{B}$ that, given (g, X_3, T), decides if $T \in_R \mathbb{G}_{p_1}$ or $T \in_R \mathbb{G}_{p_1 p_2}$.

The distinguisher $\mathcal{B}$ generates the public key PK by choosing $h \xleftarrow{R} \mathbb{G}_{p_1}$ and setting $e(g,h)^\alpha$,

$X_{p_3} = X_3$, $u = g^a$, $v = g^b$. It also chooses a random polynomial $P[X]$ of degree $t-1$ such that $P(0) = \alpha$ and sets $SK_i = h^{P(i)} \cdot Z_{3,i}$, where $Z_{3,i} \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}$, for $i = 1$ to n. It answers all decryption queries and all private key share queries according to the specification of the scheme since it knows all private key shares $\{SK_i\}_{i=1}^n$.

At the challenge phase, $\mathcal{A}$ outputs messages $M_0, M_1 \in \mathbb{G}_T$. The distinguisher $\mathcal{B}$ then picks $\beta \stackrel{R}{\leftarrow} \{0,1\}$ and computes

$$C_0^\star = M_\beta \cdot e(T,h)^\alpha, \qquad C_1^\star = T \qquad C_2^\star = T^{a \cdot \mathsf{SVK}^\star + b},$$

where $(\mathsf{SSK}^\star, \mathsf{SVK}^\star) \leftarrow \mathcal{G}(\lambda)$, and sets the challenge ciphertext as a tuple $(\mathsf{SVK}^\star, C_0^\star, C_1^\star, C_2^\star, \sigma)$, with $\sigma = \mathcal{S}(\mathsf{SSK}^\star, (C_0^\star, C_1^\star, C_2^\star))$.

If $T \in_R \mathbb{G}_{p_1}$, the ciphertext $(\mathsf{SVK}^\star, C_0^\star, C_1^\star, C_2^\star, \sigma)$ has the distribution of a normal ciphertext. If $T \in_R \mathbb{G}_{p_1 p_2}$, $(C_0^\star, C_1^\star, C_2^\star)$ has the distribution of a semi-functional ciphertext of Type I (*i.e.*, where $C_1^\star = g^s \cdot g_2^\tau$ and $C_2^\star = (u^{\mathsf{SVK}^\star} \cdot v)^s \cdot g_2^{\tau z_c}$, for some random $s \in \mathbb{Z}_{p_1}$, $\tau \in \mathbb{Z}_{p_2}^*$) for which $z_c = a \cdot \mathsf{SVK}^\star + b \bmod p_2$ (note that this value looks random to $\mathcal{A}$ since $\mathcal{A}$ has no information on $a \bmod p_2$ and $b \bmod p_2$ until the challenge phase). We conclude that $\mathcal{B}$ is playing $\mathsf{Game}^*_{Restricted}$ in the former case and Game_0 in the latter case. □

A.2 Proof of Lemma 4

Let $\mathcal{A}$ be a PPT adversary that is able to distinguish Game_{k-1} and Game_k for some $k \in \{1 \ldots, q\}$. It implies an algorithm $\mathcal{B}$ which, given $(g, X_3, X_1X_2, Y_2Y_3, T)$, decides if $T \in_R \mathbb{G}_{p_1 p_3}$ or $T \in_R \mathbb{G}$.

The distinguisher $\mathcal{B}$ generates PK by choosing $h \stackrel{R}{\leftarrow} \mathbb{G}_{p_1}$ and setting $e(g,h)^\alpha$, $X_{p_3} = X_3$, $u = g^a$, $v = g^b$. It also picks a random polynomial $P[X]$ of degree $t-1$ such that $P(0) = \alpha$ and defines $SK_i = h^{P(i)} \cdot Z_{3,i}$, with $Z_{3,i} \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}$, for $i = 1$ to n. It answers all private key share queries according to the specification of the scheme since it knows $\{SK_i\}_{i=1}^n$.

In the challenge phase, $\mathcal{A}$ outputs a pair of messages $M_0, M_1 \in \mathbb{G}_T$. The distinguisher $\mathcal{B}$ then flips a coin $\beta \stackrel{R}{\leftarrow} \{0,1\}$ and computes

$$C_0^\star = M_\beta \cdot e(X_1X_2, h)^\alpha, \qquad C_1^\star = X_1X_2 \qquad C_2^\star = (X_1X_2)^{a \cdot \mathsf{SVK}^\star + b},$$

where $(\mathsf{SSK}^\star, \mathsf{SVK}^\star) \leftarrow \mathcal{G}(\lambda)$. The above forms a Type I semi-functional encryption of M_β for which $z_c = a \cdot \mathsf{SVK}^\star + b$.

The way to answer decryption queries $\big(i, (\mathsf{SVK}^{(j)}, C_0^{(j)}, C_1^{(j)}, C_2^{(j)}, \sigma^{(j)})\big)$ depends on the index $j \in \{1, \ldots, q\}$ of the query.

- If $j < k$, $\mathcal{B}$ chooses $r, w_j, w'_j \stackrel{R}{\leftarrow} \mathbb{Z}_N$ and generates a semi-functional decryption share

$$(D_{i,1}, D_{i,2}) = (h^{P(i)} \cdot (u^{\mathsf{SVK}^{(j)}} \cdot v)^r \cdot (Y_2Y_3)^{w_j},\ g^r \cdot (Y_2Y_3)^{w'_j}).$$

- If $j > k$, $\mathcal{B}$ generates a normal decryption shares using the private key share SK_i as specified by the decryption algorithm.
- If $j = k$, $\mathcal{B}$ sets the decryption share as

$$(D_{i,1}, D_{i,2}) = (h^{P(i)} \cdot T^{a \cdot \mathsf{SVK}^{(k)} + b},\ T).$$

If $T \in_R \mathbb{G}_{p_1p_3}$, the k^{th} decryption share has the distribution of a normal decryption share. If $T \in_R \mathbb{G}$, it is distributed as a semi-functional decryption share (*i.e.*, where $D_{i,1} = h^{P(i)} \cdot (u^{\mathsf{SVK}^{(k)}} \cdot v)^r \cdot g_2^{\gamma \cdot z_k} W_3$ and $D_{i,2} = g^r \cdot g_2^{\gamma}$, for some random $r \in \mathbb{Z}_{p_1}$, $\gamma \in \mathbb{Z}_{p_2}^*$) for which $z_k = a \cdot \mathsf{SVK}^{(k)} + b \bmod p_2$. We observe that, as long as we have $\mathsf{SVK}^{(k)} \neq \mathsf{SVK}^\star \bmod p_2$, the values $z_k = a \cdot \mathsf{SVK}^{(k)} + b \bmod p_2$ and $z_c = a \cdot \mathsf{SVK}^\star + b \bmod p_2$ look independent from $\mathcal{A}$'s view. We conclude that $\mathcal{B}$ is playing Game_{k-1} if $T \in_R \mathbb{G}_{p_1p_3}$ and Game_k if $T \in_R \mathbb{G}$. □

A.3 Proof of Lemma 6

We show that, if the adversary $\mathcal{A}$ can distinguish the two games, there is a PPT distinguisher $\mathcal{B}$ against Assumption 3. This distinguisher $\mathcal{B}$ takes as input $(g, Z_2, Z_3, g^\alpha X_2, g^s Y_2, T)$ and has to decide if $T = e(g,g)^{\alpha s}$ or $T \in_R \mathbb{G}_T$.

To generate PK, algorithm $\mathcal{B}$ sets $h = g^{\gamma_1}$ for a randomly chosen $\gamma_1 \xleftarrow{R} \mathbb{Z}_N$. It then computes $e(g,h)^\alpha = e(g^\alpha X_2, h)$ and also sets $X_{p_3} = Z_3$, $u = g^a$ and $v = g^b$ with $a, b \xleftarrow{R} \mathbb{Z}_N$. It chooses a random polynomial $Q[X] \in \mathbb{Z}_N[X]$ of degree $t-1$ such that $Q(0) = 1$ and prepares the shares $\{SK_i\}_{i=1}^n$ and the verification keys $\{VK_i\}_{i=1}^n$ as $SK_i = (g^\alpha X_2)^{\gamma_1 Q(i)} \cdot Z_{2,i} \cdot Z_{3,i}$, with $Z_{2,i} \xleftarrow{R} \mathbb{G}_{p_2}$, $Z_{3,i} \xleftarrow{R} \mathbb{G}_{p_3}$, and $VK_i = e(g, SK_i)$ for each $i \in \{1, \ldots, n\}$.

To generate a decryption share for a ciphertext $C = (\mathsf{SVK}, C_0, C_1, C_2, \sigma)$, $\mathcal{B}$ picks $r, w, w' \xleftarrow{R} \mathbb{Z}_N$ and computes

$$(D_{i,1}, D_{i,2}) = \big(SK_i \cdot (u^{\mathsf{SVK}} \cdot v)^r \cdot (Z_2 Z_3)^w,\ g^r \cdot (Z_2 Z_3)^{w'}\big),$$

which forms a valid semi-functional decryption share.

In the challenge phase, $\mathcal{A}$ comes up with messages $M_0, M_1 \in \mathbb{G}_T$ and $\mathcal{B}$ flips a coin $\beta \xleftarrow{R} \{0,1\}$. To generate the challenge ciphertext, $\mathcal{B}$ generates a one-time signature key pair $(\mathsf{SSK}^\star, \mathsf{SVK}^\star) \leftarrow \mathcal{G}(\lambda)$, picks $\theta \xleftarrow{R} \mathbb{Z}_N$ and computes

$$C_0^\star = M_\beta \cdot T^{\gamma_1} \cdot e(Z_2, Z_2)^\theta, \qquad C_1^\star = g^s Y_2, \qquad C_2^\star = (g^s Y_2)^{a \cdot \mathsf{SVK}^\star + b}$$

before returning the ciphertext $(\mathsf{SVK}^\star, C_0^\star, C_1^\star, C_2^\star, \sigma)$. Whenever $\mathcal{A}$ decides to corrupt a decryption server $i \in \{1, \ldots, n\}$, $\mathcal{B}$ simply reveals SK_i.

If $T = e(g,g)^{\alpha s}$, we observe that $C = (\mathsf{SVK}^\star, C_0^\star, C_1^\star, C_2^\star, \sigma)$ forms a semi-functional encryption of Type II of M_β. If $T \in_R \mathbb{G}_T$, C is distributed as a Type II semi-functional encryption of a random plaintext. In this case, the challenge ciphertext carries no information on $\beta \in \{0,1\}$, which cannot be guessed by $\mathcal{A}$ with better probability than $1/2$. □

B Non-Interactive Threshold Signatures with Adaptive Corruptions

B.1 Definitions for Threshold Signatures

A threshold signature scheme is a tuple of algorithms $\Sigma = (\mathbf{Setup}, \mathbf{Share\text{-}Sign}, \mathbf{Share\text{-}Verify}, \mathbf{Verify}, \mathbf{Combine})$ such that:

Setup(λ, t, N)**:** takes as input a security parameter $\lambda \in \mathbb{N}$ and a pair of integers $t, N \in \mathsf{poly}(\lambda)$ such that $1 \leq t \leq N$. It outputs a public key PK and vector of private key share $\mathbf{SK} = (SK_1, \ldots, SK_n)$ and a corresponding vectors of verification keys $\mathbf{VK} = (VK_1, \ldots, VK_n)$.

Share-Sign(SK_i, M)**:** is a possibly randomized algorithm that takes in a message M and a private key share SK_i. It outputs a signature share σ_i.

Share-Verify$(PK, \mathbf{VK}, (i, \sigma_i), M)$**:** is a deterministic algorithm that takes as input a message M, the public key PK, the verification key $\mathbf{VK}$ and a pair (i, σ) consisting of an index $i \in \{1, \ldots, n\}$ and signature share σ_i. It outputs 1 or 0 depending on whether σ_i is deemed as a valid signature share or not.

Combine$(PK, \mathbf{VK}, M, \{(i, \sigma_i)\}_{i \in S})$**:** takes as input a public key PK, a message M and a subset $S \subset \{1, \ldots, n\}$ of size $t = |S|$ with pairs $\{(i, \sigma_i)\}_{i \in S}$ such that $i \in \{1, \ldots, n\}$ and σ_i is a signature share. This algorithm outputs either a full signature σ or $\perp$ if the set contains ill-formed signature shares.

Verify(PK, σ, M)**:** is a deterministic algorithm that takes as input a message M, the public key PK and a signature σ. It outputs 1 or 0 depending on whether σ is deemed valid share or not.

In the adaptive corruption setting, the security of non-interactive threshold signatures can be defined as follows, by extending the definition given by Goldwasser, Micali and Rivest [25].

Definition 3. *A threshold signature scheme Σ is existentially unforgeable under chosen-message attacks if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the following game.*

1. *The game begins with the challenger running* $\text{Setup}(\lambda, t, N)$ *to obtain PK,* $\mathbf{SK} = (SK_1, \ldots, SK_n)$ *and* $\mathbf{VK} = (VK_1, \ldots, VK_n)$*. The public key PK is given to the adversary $\mathcal{A}$.*
2. *$\mathcal{A}$ adaptively interleaves two kinds of queries.*
   - *Corruption query: at any time, $\mathcal{A}$ can choose to corrupt a server. To this end, $\mathcal{A}$ chooses $i \in \{1, \ldots, n\}$ and the challenge returns SK_i.*
   - *Signing query: $\mathcal{A}$ can also ask for a signature share on an arbitrary message M and the challenger responds by computing $\sigma_i \leftarrow$* **Share-Sign**(SK_i, M) *and returning σ to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a message $M^\star$ and a signature $\sigma^\star$. It wins if: (i) $M^\star$ was never submitted to the signing oracle; (ii) $\mathcal{A}$ did not obtain more than $t - 1$ private key shares in the game; (iii)* **Verify**$(PK, M^\star, \sigma^\star) = 1$.

$\mathcal{A}$'s advantage is defined as its success probability, taken over all coin tosses.

B.2 Construction

An adaptively secure threshold signature scheme in the standard model was notably described by Lysyanskaya and Peikert [35]. Unfortunately, servers have to run an interactive protocol to sign messages in their construction.

This section shows that, in its threshold version (and modulo a slight modification in the public key), the signature scheme that underlies the Lewko-Waters IBE scheme [34] can be proved secure under adaptive corruptions. The signing algorithm as basically identical to the algorithm that produces decryption shares in the cryptosystem of section 3.

The description below assumes that messages are elements of $\mathbb{Z}_N$. In practice, longer messages can be signed by applying a collision-resistant hash function.

Setup(λ, t, n)**:** given a security parameter $\lambda \in \mathbb{N}$ and integers $t, n \in \mathsf{poly}(\lambda)$ with $1 \leq t \leq n$, conduct the following steps.

1. Choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of order $N = p_1 p_2 p_3$, with $p_1, p_2, p_3 > 2^\lambda$.
2. Choose $\alpha \xleftarrow{R} \mathbb{Z}_N$, $g, h, u, v \xleftarrow{R} \mathbb{G}_{p_1}$, $X_{p_3} \xleftarrow{R} \mathbb{G}_{p_1}$ and compute $e(g, h)^\alpha$.

3. Define a polynomial $P[X] = \alpha + \alpha_1 X + \cdots + \alpha_{t-1} X^{t-1} \in \mathbb{Z}_N[X]$, for random coefficients $\alpha_1, \ldots, \alpha_{t-1} \xleftarrow{R} \mathbb{Z}_N$. Define the public key as

$$PK = \Big((\mathbb{G}, \mathbb{G}_T),\ N,\ g,\ e(g,h)^{\alpha},\ u,\ v,\ X_{p_3} \Big)$$

and the private key shares $\mathbf{SK} = (SK_1, \ldots, SK_n)$ as $SK_i = h^{P(i)} \cdot Z_{3,i}$, where $Z_{3,i} \xleftarrow{R} \mathbb{G}_{p_3}$, for $i = 1$ to n. Verification keys $\mathbf{VK} = (VK_1, \ldots, VK_n)$ are then defined as $VK_i = e(g,h)^{P(i)}$ for $i = 1$ to n.

PK and $\mathbf{VK}$ are made publicly available. For each $i \in \{1, \ldots, n\}$, the decryption server i is supplied with the private key share SK_i.

Share-Sign(i, SK_i, M)**:** to sign $M \in \mathbb{Z}_N$, using $SK_i = h^{P(i)} \cdot Z_{3,i}$, choose $r \xleftarrow{R} \mathbb{Z}_N$, $R_3, R_3' \xleftarrow{R} \mathbb{G}_{p_3}$ and compute the signature share

$$\sigma_i = \big(\sigma_{i,1}, \sigma_{i,2}\big) = \Big(SK_i \cdot (u^M \cdot v)^r \cdot R_3,\ g^r \cdot R_3' \Big).$$

Share-Verify$\big(PK, \mathbf{VK}, M, (i, \sigma_i)\big)$**:** parse σ_i as $(\sigma_{i,1}, \sigma_{i,2})$. If $\sigma_i = \perp$ or $\sigma_i \notin \mathbb{G}^2$, return 0. Otherwise, parse σ_i as $(\sigma_{i,1}, \sigma_{i,2}) \in \mathbb{G}^2$. Return 1 if the equality $e(\sigma_{i,1}, g) = VK_i \cdot e(u^M \cdot v, \sigma_{i,2})$ holds. Otherwise, return 0.

Combine$(PK, M, \{(i, \sigma_i)\}_{i \in S})$**:** for each $i \in S$, parse the signature share σ_i as $(\sigma_{i,1}, \sigma_{i,2})$ and return $\perp$ if **Share-Verify**$\big(PK, \mathbf{VK}, M, (i, \sigma_i)\big) = 0$. Otherwise, compute the combined signature $(\sigma_1, \sigma_2) = \big(\prod_{i \in S} \sigma_{i,1}^{\Delta_{i,S}(0)}, \prod_{i \in S} \sigma_{i,2}^{\Delta_{i,S}(0)} \big)$.

Verify$\big(PK, \mathbf{VK}, M, \sigma\big)$**:** parse σ as $(\sigma_1, \sigma_2) \in \mathbb{G}^2$. Return 1 if and only if

$$e(\sigma_1, g) = e(g,h)^{\alpha} \cdot e(u^M \cdot v, \sigma_2).$$

The security proof applies the dual system methodology in the context of signatures as was suggested in [43][Section 6] in prime order groups. The proof of the following theorem applies the same ideas in composite order groups and proceeds as in [5].

Theorem 4. *Under Assumption 1, Assumption 2 and Assumption 3, the scheme is secure against chosen-message attacks and adaptive corruptions.*

Proof. The proof considers a sequence of $q + 2$ games starting with the real attack game Game_{real} and ending with Game_q where proving the security is much easier. In the following we will denote by (i, M_j) the input of the j^{th} signing query, where $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, q\}$.

The sequence of games involves the following types of signatures

Type A signatures: are normal signatures $(\sigma_1, \sigma_2) = (h^{\alpha} \cdot (u^M \cdot v)^r \cdot R_3, g^r \cdot R_3')$.

Type B-1 signatures: are signatures of the form

$$(\sigma_1, \sigma_2) = \big(h^{\alpha} \cdot (u^M \cdot v)^r \cdot g_2^{\tau z_s} \cdot R_3, g^r \cdot g_2^{\tau} \cdot R_3' \big),$$

where $r \in \mathbb{Z}_{p_1}$, $\tau, z_c \in \mathbb{Z}_{p_2}$.

Type B-2 signatures: have the form $(\sigma_1, \sigma_2) = \big(h^{\alpha} \cdot (u^M \cdot v)^r \cdot g_2^{z_s} \cdot R_3, g^r \cdot R_3' \big)$, where $r \in \mathbb{Z}_{p_1}$, $\tau, z_c \in \mathbb{Z}_{p_2}$.

$\mathsf{Game}_{restricted}$**:** is identical to Game_{real} with the difference that the adversary $\mathcal{A}$ cannot output a forgery on a message $M^\star$ such that $M^\star = M_j \bmod p_2$ (although $M^\star \neq M_j \bmod N$), for some $j \in \{1, \ldots, q\}$.

Game_k $(1 \leq k \leq q)$**:** is defined as a hybrid game where the challenger $\mathcal{B}$ answers the first k signing queries by returning Type B signatures whereas the adversary obtains Type A signatures at its last $q - k$ signing queries.

The indistinguishability between $\mathsf{Game}_{restricted}$ and Game_{real} is proved using the usual argument in lemma 8. Then, lemmas 9 and 10 show that $\mathcal{A}$ has negligible chance of outputting Type B-1 or Type B-2 signatures unless either Assumption 1 or Assumption 2 fails to hold. Next, lemma 11 shows that, under Assumption 2, giving out semi-functional private key shares (instead of normal shares) does not increase $\mathcal{A}$'s probability to produce a Type B-1 or Type B-2 forgery. Finally, it is easy to prove that, when all signature shares and private key shares are semi-functional, any PPT forger necessarily contradicts Assumption 3. □

Lemma 8. *Any PPT adversary distinguishing* Game_{real} *from* $\mathsf{Game}_{restricted}$ *contradicts either Assumption 1 or Assumption 2.*

Proof. The proof is very similar to the one of lemma 1 (which is itself based on lemma 5 in [34]). □

Lemma 9. *In* $\mathsf{Game}_{restricted}$*, the adversary* $\mathcal{A}$ *has negligible chance of outputting a Type B-1 or a Type B-2 signature if Assumption 1 holds.*

Proof. We construct a distinguisher $\mathcal{B}$ that, on input of a tuple (g, X_3, T) decides if $T \in_R \mathbb{G}_{p_1}$ or $T \in_R \mathbb{G}_{p_1 p_2}$.

To this end, algorithm $\mathcal{B}$ interacts with the adversary $\mathcal{A}$ and generates the public key by drawing $a, b, \alpha \xleftarrow{R} \mathbb{Z}_N$ and setting $u = g^a$, $v = g^b$, $X_3 = X_{p_3}$ before giving $PK = (g, u, v, e(g,h)^\alpha, X_3)$ to $\mathcal{A}$. In addition, $\mathcal{B}$ picks a random polynomial $P[X]$ such that $P(0) = \alpha$. Private key shares are then defined as $SK_i = h^{P(i)} \cdot Z_{3,i}$, with $Z_{3,i} \xleftarrow{R} \mathbb{G}_{p_3}$, for $i = 1$ to n. The corresponding $\{VK_i\}_{i=1}^n$ are set as $VK_i = e(g, SK_i)$.

Since $\mathcal{B}$ knows all SK_i, it can perfectly answer signing queries and server corruption queries. By hypothesis, $\mathcal{A}$ eventually outputs a signature $\sigma = (\sigma_1, \sigma_2)$, which is either of the Type B-1 form $(h^\alpha \cdot (u^M \cdot v)^r \cdot g_2^{\tau z_s} \cdot R_3, g^r \cdot g_2^\tau \cdot R_3')$ or the Type B-2 form $(h^\alpha \cdot (u^M \cdot v)^r \cdot g_2^{z_s} \cdot R_3, g^r \cdot R_3')$. Since $\mathcal{B}$ knows $a, b \in \mathbb{Z}_N$, it can compute $\eta = \sigma_1 / (h^\alpha \cdot \sigma_2^{a \cdot M + b})$ which is an element of the subgroup $\mathbb{G}_{p_2 p_3}$. Unless σ is a Type B-1 signature for which $z_s = a \cdot M + b$ (which occurs with negligible probability since $a, b \bmod p_2$ are independent of $\mathcal{A}$'s view), η has a non-trivial component of order p_2. This implies that $\mathcal{B}$ can break Assumption 1 since it knows that $e(T, \eta) \neq 1_{\mathbb{G}_T}$ if $T \in_R \mathbb{G}_{p_1 p_2}$. □

Lemma 10. *The adversary outputs a Type A forgery with negligibly different probabilities in* Game_k *and* Game_{k+1} *if Assumption 2 holds.*

Proof. For the sake of contradiction, we assume that a forger $\mathcal{A}$ has significantly higher probability of outputting a Type A signature in Game_{k+1} than in Game_k. We outline a distinguisher $\mathcal{B}$ for Assumption 2. Algorithm $\mathcal{B}$ takes as input $(g, X_1 X_2, Z_3, Y_2 Y_3, T)$ and uses $\mathcal{A}$ to decide if $T \in \mathbb{G}$ or $T \in \mathbb{G}_{p_1 p_3}$. Recall that $\mathcal{A}$ must obtain Type B-1 signatures at its first k signing queries and Type A signatures at the last $q - k - 1$ queries. The k^{th}-query will be a Type A signature if $T \in \mathbb{G}_{p_1 p_3}$ and a Type B-1 signature if $T \in \mathbb{G}$.

It comes that $\mathcal{A}$ will produce a Type A forgery with about the same probability in either case

if Assumption 2 holds. We then show that the distinguisher $\mathcal{B}$ can indeed distinguish whether $\mathcal{A}$'s forgery will be of Type A or not with overwhelming probability.

To this end, $\mathcal{B}$ prepares PK by choosing $h \stackrel{R}{\leftarrow} \mathbb{G}_{p_1}$, $\alpha, a, b \stackrel{R}{\leftarrow} \mathbb{Z}_N$ and setting $u = g^a$, $v = g^b$. The public key $PK = \big(g, e(g,h)^\alpha, u, v, Z_3\big)$ is given to $\mathcal{A}$. Then, $\mathcal{B}$ can perfectly answer private key share queries (since it knows $\{SK_i\}_{i=1}^n$) and answers $\mathcal{A}$'s signing queries (i_j, M_j) depending on their index $j \in \{1, \ldots, q\}$.

Case $j < k$: to generate a signature share on message M_j on behalf of server $i_j \in \{1, \ldots, n\}$, $\mathcal{B}$ first chooses $r \stackrel{R}{\leftarrow} \mathbb{Z}_N$. It then chooses $w_1, w_2 \stackrel{R}{\leftarrow} \mathbb{Z}_N$, $R_3, R_3' \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}$. It computes a Type B-1 signature share $(\sigma_{i_j,1}, \sigma_{i_j,2})$ as

$$\sigma_{i_j,1} = SK_{i_j} \cdot (u^{M_j} \cdot v)^r \cdot (Y_2Y_3)^{w_1}, \qquad \sigma_{i_j,2} = g^r \cdot (Y_2Y_3)^{w_2}.$$

Case $j > k$: in this case, $\mathcal{B}$ simply computes a Type A signature using the private key share SK_{i_j} as specified by the signing algorithm.

Case $j = k$: to answer the k^{th} signing query (i_k, M_k), $\mathcal{B}$ first chooses $w_1, w_2 \stackrel{R}{\leftarrow} \mathbb{Z}_N$, $R_3, R_3' \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}$. It uses the challenge $T \in \mathbb{G}$ to compute

$$\sigma_{i_k,1} = SK_{i_k} \cdot T^{a \cdot M_k + b} \cdot R_3, \qquad \sigma_{i_k,2} = T \cdot R_3'.$$

It is easy to see that, in the situation where $T \in_R \mathbb{G}$, if we let g_2^τ be the $\mathbb{G}_{p_2}$ component of T for some $\tau \in \mathbb{Z}_{p_2}^*$, we obtain a Type B-1 signature where $z_s = (a \cdot M_k + b) \bmod p_2$. In contrast, if $T \in_R \mathbb{G}_{p_1p_3}$, the above forms a Type A signature.

At the end of the game, $\mathcal{A}$ outputs a message $M^\star$ with a forgery $\sigma^\star = (\sigma_1^\star, \sigma_2^\star)$ that satisfies the verification equation. At this stage, $\mathcal{B}$ returns 0 (meaning that it believes that $T \in_R \mathbb{G}_{p_1p_3}$) if $\sigma^\star$ is a Type A signature. If $\sigma^\star$ is a Type B-1 or Type B-2 signature, $\mathcal{B}$ returns 1 and bets that $T \in_R \mathbb{G}$.

To decide whether $\sigma^\star$ is a Type A signature or not, $\mathcal{B}$ uses its input element X_1X_2 and defines $\Theta = (X_1X_2)^{a \cdot M^\star + b}$. It interprets $\sigma^\star$ as a Type A signature if

$$\frac{e(\sigma_1^\star, X_1X_2)}{e(\sigma_2^\star, \Theta)} = e(X_1X_2, h)^\alpha. \tag{5}$$

It is easy to see that equation (5) can never be satisfied by a Type B-2 forgery and a Type B-1 forgery $\sigma^\star = (h^\alpha \cdot (u^M \cdot v)^r \cdot g_2^{\tau \cdot z_f} \cdot R_3, g^r \cdot g_2^\tau \cdot R_3')$ can only verify it if $z_f = a \cdot M^\star + b \bmod p_2$. Before the forgery stage however, the only information that $\mathcal{A}$ can potentially infer about $a, b \bmod p_2$ is the value $z_s = a \cdot M_k + b \bmod p_2$ of the k^{th} signing query. Since z_s and z_f are linearly independent as long as $M^\star \neq M_k \bmod p_2$, a Type B-1 signature can only satisfy the test with probability $1/p_2$ since $a, b \bmod p_2$ are uniformly distributed in $\mathbb{Z}_{p_2}$. □

Lemma 11. *If the adversary can output a Type B-1 or Type B-2 forgery with substantially higher probability in* Game_q^* *than in* Game_q*, there is a distinguisher for Assumption 2.*

Proof. We now assume that a forger $\mathcal{A}$ has higher chance of outputting a Type B-1 or Type B-2 forgery in Game_q^*. We show that $\mathcal{A}$ implies a distinguisher $\mathcal{B}$ for Assumption 2. Algorithm $\mathcal{B}$ takes as input $(g, X_1X_2, Z_3, Y_2Y_3, T)$ and decides if $T \in \mathbb{G}$ or $T \in \mathbb{G}_{p_1p_3}$.

The distinguisher $\mathcal{B}$ generates PK by choosing $\alpha, a, b \stackrel{R}{\leftarrow} \mathbb{Z}_N$ and setting $u = g^a$, $v = g^b$. It then sets $e(g,h)^\alpha = e(g,T)^\alpha$, which implicitly sets h as the $\mathbb{G}_{p_1}$ component of T. The public

key $PK = \big(g, e(g,T)^{\alpha}, u, v, Z_3\big)$ is given to $\mathcal{A}$. Also, $\mathcal{B}$ picks a random degree-t polynomial $P[X]$ such that $P(0) = \alpha$. Private key shares are set as $SK_i = T^{P(i)} \cdot Z_{3,i}$, with $Z_{3,i} \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}$, and the corresponding verification key $\mathbf{VK}$ is defined by $VK_i = e(g, SK_i)$ for $i = 1$ to n.

Then, $\mathcal{B}$ answers private key share queries by simply revealing the requested SK_i to $\mathcal{A}$. To generate a signature share on message M_j on behalf of server $i_j \in \{1, \ldots, n\}$, $\mathcal{B}$ picks $r \stackrel{R}{\leftarrow} \mathbb{Z}_N$, $w_1, w_2 \stackrel{R}{\leftarrow} \mathbb{Z}_N$, $R_3, R'_3 \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}$ and computes a Type B-1 signature share $(\sigma_{i_j,1}, \sigma_{i_j,2})$ as

$$\sigma_{i_j,1} = SK_{i_j} \cdot (u^{M_j} \cdot v)^r \cdot (Y_2Y_3)^{w_1}, \qquad \sigma_{i_j,2} = g^r \cdot (Y_2Y_3)^{w_2}.$$

At the end of the game, $\mathcal{A}$ outputs a message $M^{\star}$ with a forgery $\sigma^{\star} = (\sigma_1^{\star}, \sigma_2^{\star})$ that satisfies the verification equation. It means that $\sigma^{\star}$ is necessarily of the form $(\sigma_1^{\star}, \sigma_2^{\star}) = (h^{\alpha} \cdot (u^{M^{\star}} \cdot v)^r \cdot g_2^{\tau_1} \cdot R_3, g^r \cdot g_2^{\tau_2} \cdot R'_3)$, for some $r \in \mathbb{Z}_{p_1}$, $\tau_1, \tau_2 \in \mathbb{Z}_{p_2}$ and $R_3, R'_3 \in \mathbb{G}_{p_3}$. At this stage $\mathcal{B}$ computes $\eta = \sigma_1^{\star}/\sigma_2^{\star a \cdot M^{\star} + b}$ which is of the form $\eta = h^{\alpha} \cdot g_2^{\tilde{\tau}} \cdot \tilde{R}_3$ for some $\tilde{\tau} \in \mathbb{Z}_{p_2}$, $\tilde{R}_3 \in \mathbb{G}_{p_3}$ and where h is the $\mathbb{G}_{p_1}$ component of T. Then, $\mathcal{B}$ checks whether the relation

$$e(X_1X_2, \eta) = e(X_1X_2, T)^{\alpha} \tag{6}$$

is satisfied. If yes, $\mathcal{B}$ outputs 0 (which indicates that $T \in_R \mathbb{G}_{p_1p_3}$). If (6) is not satisfied, $\mathcal{B}$ outputs 1 (meaning that $T \in_R \mathbb{G}$).

We argue that $\mathcal{B}$ has non-negligible advantage as a distinguisher. To see this, we first note that, if $T \in_R \mathbb{G}_{p_1p_3}$, $\mathcal{B}$ is actually playing Game_q with $\mathcal{A}$ and relation (6) holds whenever $\tilde{\tau} = 0$. If $T \in_R \mathbb{G}$ (say $T = h \cdot g_2^{\gamma_2} \cdot g_3^{\gamma_3}$ for some $\gamma_2 \in \mathbb{Z}_{p_2}$ and $\gamma_3 \in \mathbb{Z}_{p_3}$), it is rather playing Game_q^* since $\mathcal{A}$ gathers at most $t-1$ private key shares $SK_i = h^{P(i)} \cdot g_2^{\gamma_2 \cdot P(i)} \cdot W_{3,i}$, with $W_{3,i} \in_R \mathbb{G}_{p_3}$, and their $\mathbb{G}_{p_2}$ components look independent from $\mathcal{A}$'s view.

Then, we observe that, if $\sigma^{\star}$ is a Type B-1 or Type B-2 signature, the expression of η is such that $\tilde{\tau} \neq 0$ with all but negligible probability (since $a, b \bmod p_2$ are completely independent of $\mathcal{A}$'s view). When $\tilde{\tau} \neq 0$, relation (6) can only be satisfied with negligible probability. Indeed, in the situation $T \in_R \mathbb{G}$, the right-hand-side member of (6) can be written

$$e(X_1X_2, T)^{\alpha} = e(X_1, h)^{\alpha} \cdot e(X_2, g_2)^{\gamma_2 \cdot P(0)}$$

and given that $e(X_1X_2, \eta) = e(X_1, h)^{\alpha} \cdot e(X_2, g_2)^{\tilde{\tau}}$, we remark that the equality (6) can only hold if $\tilde{\tau} = \gamma_2 \cdot P(0) \bmod p_2$. This only occurs with negligible probability since $\mathcal{A}$ obtains no information about $\alpha \bmod p_2 = P(0) \bmod p_2$ during the game.

We note that, when $\tilde{\tau} \neq 0$, the test (6) also fails in the case $T \in_R \mathbb{G}_{p_1p_3}$, which causes the distinguisher $\mathcal{B}$ to incorrectly answer. However, this is not a concern as lemmas 9 and 10 already guarantee that $\mathcal{A}$ must break some assumption to produce a Type B-1 or Type B-2 signature when $T \in_R \mathbb{G}_{p_1p_3}$ (recall that $\mathcal{B}$ plays Game_q if $T \in_R \mathbb{G}_{p_1p_2}$). Moreover, the hypothesis of the lemma implies that $\sigma^{\star}$ is more likely to give $\tilde{\tau} \neq 0$ if $T \in_R \mathbb{G}$ than if $T \in_R \mathbb{G}_{p_1p_3}$. By construction, the probability that $\mathcal{B}$ outputs 1 is thus higher in the former case. □

Lemma 12. *As long as Assumption 3 holds, no PPT adversary can output a Type A forgery in* Game_q^*.

Proof. We outline an algorithm $\mathcal{B}$ that takes as input $\big(g, g^{\alpha}X_2, X_3, g^sY_2, Z_2\big)$ and uses a Type A forger $\mathcal{A}$ to compute $T = e(g,g)^{\alpha s}$ (and *a fortiori* break Assumption 3). To this end, $\mathcal{B}$ generates the public key $PK = \big(g, e(g,h)^{\alpha}, u, v, X_{p_3}\big)$ by choosing $a, b, \gamma_1 \stackrel{R}{\leftarrow} \mathbb{Z}_N$ and setting $h = g^{\gamma_1}$, $X_{p_3} = X_3$,

$e(g,h)^{\alpha} = e(g^{\alpha}X_2, h)$ as well as $u = g^a$ and $v = g^b$. In addition, $\mathcal{B}$ picks a random $(t-1)$-degree polynomial $Q[X] \in \mathbb{Z}_N[X]$ such that $Q(0) = 1$ and defines private key shares are defined as $SK_i = (g^{\alpha} \cdot X_2)^{Q(i)} \cdot Z_{2,i} \cdot Z_{3,i}$ with $Z_{2,i} \stackrel{R}{\leftarrow} \mathbb{G}_{p_2}$, $Z_{3,i} \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}$. The corresponding $\{VK_i\}_{i=1}^n$ is defined as in previous lemmas.

Whenever the forger $\mathcal{A}$ decides to corrupt a server, $\mathcal{B}$ simply reveals the corresponding private key share. When $\mathcal{A}$ makes a signing query (i, M_j), $\mathcal{B}$ chooses $r \stackrel{R}{\leftarrow} \mathbb{Z}_N$, $w_1, w_2 \stackrel{R}{\leftarrow} \mathbb{Z}_N$, $R_3, R_3' \stackrel{R}{\leftarrow} \mathbb{G}_{p_3}$. It computes

$$\sigma_i = (\sigma_{i,1}, \sigma_{i,2}) = \big(SK_i \cdot (u^{M_j} \cdot v)^r \cdot (Z_2Z_3)^{w_1}, g^r \cdot (Z_2Z_3)^{w_2}\big)$$

which has the distribution of a Type B-1 signature.

The game ends with the adversary $\mathcal{A}$ outputting a message $M^{\star}$ as well as a Type-A signature $\sigma^{\star} = (\sigma_1^{\star}, \sigma_2^{\star}) = (h^{\alpha} \cdot (u^{M^{\star}} \cdot v)^r \cdot R_3, g^r \cdot R_3')$. Since $\sigma^{\star}$ is a Type A signature, $\sigma_1^{\star}, \sigma_2^{\star}$ have no $\mathbb{G}_{p_2}$ component and $\mathcal{B}$ can then compute

$$e(g,g)^{\alpha \cdot s} = \Big(\frac{e(g^s Y_2, \sigma_1^{\star})}{e\big((g^s Y_2)^{a \cdot M^{\star} + b}, \sigma_2^{\star}\big)}\Big)^{1/\gamma_1}.$$

It comes that $\mathcal{A}$'s advantage is thus negligible if Assumption 3 holds. □