Threshold and Revocation Cryptosystems via Extractable Hash Proofs

Hoeteck Wee*

Queens College, CUNY
hoeteck@cs.qc.cuny.edu

Abstract. We present a new unifying framework for constructing non-interactive threshold encryption and signature schemes, as well as broadcast encryption schemes, and in particular, derive several new cryptosystems based on hardness of factoring, including:

- a threshold signature scheme (in the random oracle model) that supports ad-hoc groups (i.e., exponential number of identities and the set-up is independent of the total number of parties) and implements the standard Rabin signature;
- a threshold encryption scheme that supports ad-hoc groups, where encryption is the same as that in the Blum-Goldwasser cryptosystem and therefore more efficient than RSA-based implementations;
- a CCA-secure threshold encryption scheme in the random oracle model;
- a broadcast encryption scheme (more precisely, a revocation cryptosystem) that supports ad-hoc groups, whose complexity is comparable to that of the Naor-Pinkas scheme; moreover, we provide a variant of the construction that is CCA-secure in the random oracle model.

Our framework rests on a new notion of *threshold extractable hash proofs*. The latter can be viewed as a generalization of the extractable hash proofs, which are a special kind of non-interactive zero-knowledge proof of knowledge.

1 Introduction

As the old saying goes, "Do not put all your eggs in one basket". Indeed, this is the basic principle underlying threshold cryptography, which distributes some cryptographic functionality amongst many users in such a way that: (1) any $t + 1$ parties can collectively compute the functionality; and (2) no colluding subset of t parties can compromise the security of the functionality. The two canonical applications of threshold cryptography are in public-key encryption and signature schemes, where the functionalities in consideration correspond to decryption and signing respectively. The approach was initiated in [19, 20, 21], and there is now a large body of work on threshold signature schemes [18, 27, 40, 26, 28, 29, 8, 34, 30] and threshold encryption schemes [41, 11, 24, 34, 9, 10].

* Supported by NSF CAREER Award CNS-0953626, and the US Army Research laboratory and the UK Ministry of Defence under agreement number W911NF-06-3-0001.

If we are willing to settle solely for pairings-based schemes, then the main questions of practical threshold encryption and signature schemes are essentially solved. The threshold signature schemes of Boneh, Lynn, Shacham [8] and threshold encryption schemes of Boyen, Mei and Waters [10, 9] are non-interactive (each party locally computes a signature/decryption share without any interaction with the other parties), guarantees robustness against corrupted parties (given a verification key, each party can check that the signature/decryption shares are well-formed) and are well-suited for use in ad-hoc groups such as MANETs ("mobile ad-hoc networks", which arise in many wireless and military settings). The latter requirement, articulated in the recent work of Gennaro et al. [30], means that the cryptographic protocol supports an identity space of exponential size and does not have any dependency on the total number of parties.

Given that the underlying principle of threshold cryptography is to avoid any single point of failure, it would be quite ironic of course to base all of threshold cryptography on pairings and discrete-log assumptions. A natural class of alternative assumptions would be that related to factoring, where many problems remain open. Here, virtually all threshold signature schemes are based on the RSA assumption; the only exception is the factoring-based scheme of Katz and Yung [34], which does not support ad-hoc groups. We also do not know of any threshold encryption schemes based on hardness of factoring which supports ad-hoc groups. More notably, we do not know of any practical CCA-secure threshold encryption scheme based on hardness of factoring, even in the random oracle model; this was posed as an open problem in [41]. Similarly, very little is known about factoring-based revocation cryptosystems, a primitive seemingly unrelated to threshold cryptosystems. These are a special kind of broadcast encryption schemes [23] where a sender broadcasts encrypted messages created in such a way that all but a small subset of recipients (the "revoked" users) can decrypt the message.

This work. We present a new unifying framework for constructing non-interactive threshold encryption and signature schemes, as well as revocation schemes, and in particular, derive several new cryptosystems based on hardness of factoring, including:

- a threshold signature scheme (in the random oracle model) that supports ad-hoc groups and implements the standard Rabin signature (namely, the end-result of running the protocol is a Rabin signature and anyone can verify that signature as if it were generated by a standard centralized signer);
- a threshold encryption scheme (in the standard model) that supports ad-hoc groups, where encryption is the same as that in the Blum-Goldwasser cryptosystem [5] and therefore more efficient than RSA-based implementations;
- a CCA-secure threshold encryption scheme (in the random oracle model), whose computation and communication complexity is roughly that of Shoup's threshold signature scheme [40] plus that of the Hofheinz-Kiltz CCA-secure encryption scheme [32].
- a revocation cryptosystem (in the standard model) that also supports ad-hoc groups, whose complexity is comparable to that of the Naor-Pinkas scheme [36]; moreover, we provide a variant of the construction that is CCA-secure in the random oracle model.

reference	assumption	security	ad-hoc
[18, 28]	RSA	CPA	×
[25, 24]	DCR	CPA, CCA (RO)	×
[34, 24]	QR	CPA, CCA (RO)	×
[34]	factoring	CPA	×
this work	factoring	CPA	✓
this work	factoring	CCA (RO)	×

Fig. 1. Summary of threshold PKEs from assumptions related to factoring

We refer to Figure 1 for a comparison with previous factoring-based threshold cryptosystems. We also note here that our framework also captures many of the aforementioned threshold and revocation cryptosystems based on pairings and discrete-log assumptions [8, 10, 36] (see Figure 2).

2 Overview of our constructions

We proceed to provide an overview of our framework and the constructions.

Threshold extractable hash proofs. We introduce the notion of a *threshold extractable hash proof system*, which generalizes our recent work [42]. Informally, these hash proof systems are like the Cramer-Shoup universal hash proofs [15] in that they are a special kind of non-interactive zero-knowledge proofs [6], except we replace the soundness requirement (corresponding to smoothness) with a "proof of knowledge property" [38, 17]. Specifically, the proofs are specified by a family of functions $\mathcal{H}_{\text{HK}}(\cdot,\cdot)$ indexed by a public key HK and takes two inputs, a tag and an instance u. We will require that $\mathcal{H}_{\text{HK}}(\cdot,\cdot)$ be efficiently computable either given the coin tosses used to sample the instance u, or a secret key for the corresponding tag. In addition, the family of functions is parametrized by a "threshold" value 1^t which plays the following role:

- $(t+1)$-EXTRACTABILITY: given any $t+1$ proofs for the same instance u on $t+1$ different tags, we may efficiently "extract" a witness s for the instance (this is mostly meaningful when computing the witness given only u is hard-on-average);
- t-SIMULATABILITY: on the other hand, any t proofs reveal no "useful" information about the witness, that is, there exists a simulator that can efficiently generate proofs for t different tags for an instance u without knowing the witness. The formal requirement is stronger, namely that the simulator can generate a random HK along with the secret keys for any t different tags.

We point out here that the case $t = 1$ corresponds to the "all-but-one" extractable hash proofs in [42]; that is, threshold extractable hash proofs may be regarded as a "all-but-t" analogue of extractable hash proofs.

From hash proofs to cryptosystems. With this informal overview of threshold extractable hash proofs, we can now outline how we derive threshold and revocation cryptosystems (see Figure 3 for the parameters). We note here that we are working in the model with a trusted dealer that generates HK and issues each party with identity ID with the secret key corresponding to the tag ID. This is well-suited for dynamic ad-hoc networks where any user can join the network at any time and register with the trusted dealer to obtain a secret key; however, in this work, we only address the setting where the threshold t is fixed once and for all.

- THRESHOLD ENCRYPTION: To encrypt a bit, we generate a random instance-witness pair (u, s), mask the bit using the hard-core bit of s, and publish u along with the masked bit. The decryption share from a party ID is simply a proof $\mathcal{H}_{\text{HK}}(\text{ID}, u)$. The functionality requirement follows from $(t+1)$-extractability – anyone can decrypt upon receiving the shares from any $t+1$ parties; moreover, t-simulatability prevents any t colluding parties from decrypting the message.

- THRESHOLD SIGNATURES: The signature for a message M is a witness s for the instance $\mathsf{H}(M)$ where $\mathsf{H}(\cdot)$ is a hash function modeled as a random oracle (a "full domain hash"). The signature share from a user ID is again a proof $\mathcal{H}_{\text{HK}}(\text{ID}, \mathsf{H}(M))$; functionality and security is exactly analogous to that for threshold encryption.

- REVOCATION CRYPTOSYSTEM: Here, we want to encrypt messages in such a way that any party outside a revoked set of t users $\text{ID}_1, \ldots, \text{ID}_t$ can decrypt. (To revoke fewer than t users, we may simply add "dummy" identities.) Again, to encrypt a bit, we generate a random instance-witness pair (u, s), mask the bit using the hard-core bit of s, and publish u along with the masked bit and t proofs $\mathcal{H}_{\text{HK}}(\text{ID}_1, u), \ldots, \mathcal{H}_{\text{HK}}(\text{ID}_t, u)$. Any party ID outside the revoked set can compute a $t+1$'th proof using the secret key for the tag ID and then derive s to decrypt.

We also show how to realize robustness for signatures following [28], and CCA security for threshold encryption and revocation schemes (with instantiations in the random oracle model). Our techniques for achieving CCA security follow the "all-but-one extractable hash proof" paradigm in [42, 12, 32], whereas most of the previous constructions [11, 22, 24] rely on the Cramer-Shoup [14, 15] or the Naor-Yung paradigm [37] which seem to be inherently limited to decisional assumptions.

Realizing threshold extractable hash proofs. We begin with an informal description of our approach for constructing threshold extractable hash proofs. The approach generalizes the direct constructions of all-but-one extractable hash proofs in [42] and provide a different perspective into those constructions. The basic idea is to exploit Shamir secret sharing in the exponent. More precisely, we sample a random degree t polynomial f subject to the constraint $f(0)$ is a special trapdoor such that $s = u^{f(0)}$. The master secret key is given by f and the public key HK is some "commitment" to the coefficients of f. The secret key corresponding to TAG is given by $f(\text{TAG})$ and $\mathcal{H}_{\text{HK}}(\text{TAG}, u) := u^{f(\text{TAG})}$. Computing $\mathcal{H}_{\text{HK}}(\text{TAG}, u)$ given the coin tosses for generating u corresponds to evaluating f in the exponent. Given the hash proofs for u

primitive	CDH/DDH	BDDH	factoring
threshold PKE	folklore	folkore	new
threshold signatures	[27] (RO)	[8]	new (RO)
broadcast PKE	[36]	[36]	new
CCA threshold PKE	[41, 24] (RO)	[10, 9]	new (RO)
CCA broadcast PKE	[22]	new	new (RO)

Fig. 2. Summary of previous and present constructions from CDH/DDH, BDDH and hardness of factoring. For most primitives and assumptions, our constructions match and improve upon existing constructions. We underline the references for the two settings where existing work achieve better parameters than our work. We note that the [22] scheme only achieves a relaxed notion of CCA security.

primitive	public key	secret key	ciphertext/signature	decryption/signing share
threshold PKE	$O(1)$	$O(1)$	$O(1)$	$O(1)$
threshold signatures	$O(1)$	$O(1)$	$O(1)$	$O(1)$
broadcast PKE	$O(t)$	$O(1)$	$O(t)$	$O(1)$
CCA threshold PKE	$O(1)$	$O(1)$	$O(1)$	$O(1)$
CCA broadcast PKE	$O(t)$	$O(1)$	$O(t)$	$O(1)$

Fig. 3. Complexity of our BDDH and factoring-based schemes, as measured by number of group elements. For broadcast PKE, t denotes the revocation threshold. Here, secret key refers to the user/server's secret key; the dealer's secret key has size $O(t)$.

corresponding to $t+1$ distinct tags, we may recover $u^{f(0)}$ via Lagrangian interpolation in the exponent. This is easy for discrete-log type settings since we know the order of the group. Generating simulated proofs or secret keys for any t distinct tags is easy since the evaluation of f at any t locations look random; the main technical complication comes in having to simulate a consistent HK (though that is again easy for discrete-log type settings).

The factoring-based construction is based on Rabin's trapdoor permutation. We begin with the simple observation that we may compute a square root of u by exponentiation to the secret value $(\phi(N)+1)/2 = 2^{-1} \pmod{\phi(N)}$. Here, we use secret sharing over the ring $\mathbb{Z}_{\phi(N)}$, whereas the previous factoring-based scheme in [34] applies secret sharing to the factorization of the modulus N. In order to perform Lagrangian interpolation over the ring $\mathbb{Z}_{\phi(N)}$ of unknown order, we build on ideas developed in the context of RSA-based schemes [40, 30] and the factoring-based cryptosystem in [32]. Informally, we set $f(0)$ to be $2^{-(t+1)\lceil \log N \rceil} \pmod{\phi(N)}$. Given the hash proofs $u^{f(\text{TAG})}$ corresponding to $t+1$ distinct tags, we may recover $u^{Df(0)}$, where D denotes an integer used to "clear the denominator" in the fractional Lagrangian coefficients and it depends on the tags used in the $t+1$ proofs. In order to support an identity space of exponential size, we bound the highest power of 2 that divides D by

$2^{-t\lceil \log N \rceil}$, following [30]. Given both u and $u^{Df(0)}$, we may then recover $s = u^{1/2}$ using Shamir's algorithm for "GCD in the exponent" [39].

In our constructions, we "commit" to the coefficients of f instead of the evaluations of f in HK, evaluating $u^{f(\text{TAG})}$ given the coin tosses used to sample u does not require interpolation; this appears to be the first time this property is exploited for RSA/factoring-based schemes and is important for handling ad-hoc networks in our revocation scheme.

Towards lattice-based instantiations. Looking forward, we plan to look into lattice-based instantiations of threshold extractable hash proofs, extending the ideas and results in [2, 3]. One possible starting point is the following construction: $\text{HK} := (\mathbf{A}_0, \ldots, \mathbf{A}_t) \leftarrow_{\text{R}} \mathbb{Z}_q^{m \times n}$ where $m = \text{poly}(n)$ and $f(\text{TAG}) := \mathbf{A}_0 + \mathbf{A}_1 \text{TAG} + \cdots + \mathbf{A}_t \text{TAG}^t$ for $\text{TAG} \in \mathbb{Z}_n \subset \mathbb{Z}_q$. The instance u is a perturbed lattice point $\mathbf{A} \cdot \mathbf{s} + \eta$ where $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathcal{H}_{\text{HK}}(\text{TAG}, u)$ is of the form $f(\text{TAG})\mathbf{s} + \eta$. Due to the interaction between the Lagrangian coefficients and the noise vectors, we will need to work with field sizes and approximation factors much larger than n^t (c.f. [3]). However, under sub-exponential hardness assumptions for lattice problems, we could still potentially get meaningful results for parameters such as $t = \sqrt{n}$, say.

3 Preliminaries and Definitions

A *key encapsulation mechanism* (KEM) $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with key-space $\{0,1\}^k$ consists three polynomial-time algorithms.Via $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^k)$ the randomized key-generation algorithm produces public/secret keys for security parameter 1^k; via $(C, K) \leftarrow \mathsf{Enc}(\text{PK})$, the randomized encapsulation algorithm creates a uniformly distributed symmetric key $K \in \{0,1\}^k$, together with a ciphertext C; via $K \leftarrow \mathsf{Dec}(\text{SK}, C)$, the possessor of secret key SK decrypts ciphertext C to get back a key K which is an element in $\{0,1\}^k$ or a special reject symbol $\perp$. For consistency, we require that for all k and all $(C, K) \leftarrow \mathsf{Enc}(\text{PK})$, we have $\Pr[\mathsf{Dec}(\text{SK}, C) = K] = 1$.

3.1 Binary Relations for Search Problems

Fix a family of (binary) relations R_{PP} indexed by a public parameter PP. We require that PP be efficiently samplable given a security parameter 1^k, and assume that all algorithms are given PP as part of its input. We omit PP henceforth whenever the context is clear. We will also require that R_{PP} be efficiently samplable, where the sampling algorithm is denoted by SampR. Intuitively, the relation R_{PP} corresponds to a hard search problem, that is, given a random u, it is hard to find s such $(u, s) \in \mathsf{R}_{\text{PP}}$. More formally, we say that a binary relation R_{PP} is *one-way* if:

- with overwhelming probability over PP, for all u, there exists at most one s such that $(u, s) \in \mathsf{R}_{\text{PP}}$; and
- there is an efficiently computable generator G such that $\mathsf{G}_{\text{PP}}(s)$ is pseudorandom even against an adversary that gets PP, u, where $(u, s) \leftarrow_{\text{R}} \mathsf{SampR}(\text{PP})$. (We will

also refer to G as extracting hard-core bits from s.) That is, the following quantity is negligible for all PPT $\mathcal{A}$:

$$\mathsf{AdvPRG}^{\mathcal{A}}(k) := \Pr\left[\begin{array}{c} (u,s) \leftarrow \mathsf{SampR}(\mathrm{PP}); \\ b = b' : K_0 \leftarrow \mathsf{G}(s); K_1 \leftarrow_{\mathrm{R}} \{0,1\}^k; b \leftarrow_{\mathrm{R}} \{0,1\}; \\ b' \leftarrow \mathcal{A}(\mathrm{PP}, u, K_b) \end{array}\right]$$

For relations where computing s given u is hard on average, we may derive a generator G_{PP} with a one-bit output via the Goldreich-Levin hard-core bit $\mathsf{GL}(\cdot)$ [31] (with the randomness in PP). In many cases as we shall see shortly, we may derive a linear number of hard-core bits by either iterating a one-way permutation or relying on decisional assumptions.

3.2 Threshold Extractable Hash Proofs.

We consider a family of hash functions $\mathcal{H}_{\mathrm{HK}}(\cdot,\cdot)$ indexed by a public key HK, that takes as input a tag and an instance. More formally, an *threshold extractable hash proof system* is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Pub}, \mathsf{Ext}, \mathsf{Priv})$ satisfying the following properties with overwhelming probability over $(\mathrm{PP}, \mathrm{SP})$:

(KEY GENERATION.**)** The set-up algorithm $\mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t)$ generates public keys HK and a master secret key MSK. Given a tag TAG, the share generation algorithm computes an associated key $\mathsf{ShareGen}(\mathrm{MSK}, \mathrm{TAG}) = \mathrm{SK}_{\mathrm{TAG}}$.

(PUBLIC EVALUATION.**)** For all HK, TAG and $(u,s) = \mathsf{SampR}(r)$: $\mathsf{Pub}(\mathrm{HK}, \mathrm{TAG}, r) = \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$.

(PRIVATE EVALUATION.**)** For all HK, TAG and u: $\mathsf{Priv}(\mathrm{SK}_{\mathrm{TAG}}, u) = \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$

($(t+1)$-EXTRACTION.**)** For all HK, u and all $t+1$ distinct tags $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_{t+1}$:

$$(u, \mathsf{Ext}(u, \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}_1, u), \ldots, \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}_{t+1}, u))) \in \mathsf{R}_{\mathrm{PP}}$$

We note here that Ext also receives as input the tags $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_{t+1}$ (omitted for notational simplicity) but does not require as input HK.

(t-SIMULATION.**)** For all $(\mathrm{PP}, \mathrm{SP}), 1^t$ and all $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_t$, the distribution of $(\mathrm{HK}, \mathrm{SK}_{\mathrm{TAG}_1}, \ldots, \mathrm{SK}_{\mathrm{TAG}_t})$ in the following experiments are statistically indistinguishable:

- the first is that obtained via key generation: $(\mathrm{HK}, \mathrm{MSK}) \leftarrow_{\mathrm{R}} \mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t)$ and $\mathrm{SK}_{\mathrm{TAG}_i} := \mathsf{ShareGen}(\mathrm{MSK}, \mathrm{TAG}_i)$ for $i = 1, \ldots, t$;
- the second is that given by $\mathsf{SetupSim}(\mathrm{PP}, \mathrm{TAG}_1, \ldots, \mathrm{TAG}_t)$

Finally, we say that a threshold extractable hash proof system is *publicly verifiable* if there is an efficient algorithm Ver that on input $(\mathrm{HK}, \mathrm{TAG}, u, \tau)$ outputs true iff $\tau = \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$.

4 Threshold Encryption Schemes

We define a threshold KEM (from which we can readily build a threshold encryption scheme):

(SHARING PHASE.**)** The set-up algorithm $\mathsf{Setup}(\text{PP}, \text{SP}, 1^t)$ generates a public key PK and a master secret key MSK. Given an identity ID, the share generation algorithm computes an associated key $\mathsf{ShareGen}(\text{MSK}, \text{ID}) = \text{SK}_{\text{ID}}$.

(ENCRYPTION.**)** The encapsulation algorithm $\mathsf{Enc}(\text{PK})$ generates (C, K), namely a random key K along with a ciphertext C.

(DECRYPTION.**)** The share decryption algorithm $\mathsf{ShareDec}(\text{ID}, C)$ takes an identity ID and computes the decryption share for that identity using its secret key SK_{ID}. Moreover, there's a combining algorithm that takes any $t+1$ decryption shares and outputs K.

Semantic Security. We define the advantage $\mathsf{AdvThEnc}^{\mathcal{A}}(k)$ to be:

$$\Pr\left[b = b' : \begin{array}{l} (\text{ID}_1^*, \ldots, \text{ID}_t^*) \leftarrow \mathcal{A}_1(1^k); \\ (\text{PK}, \text{MSK}) \leftarrow \mathsf{Gen}(1^k, 1^t); \\ \text{SK}_{\text{ID}_i^*} \leftarrow \mathsf{ShareGen}(\text{MSK}, \text{ID}_i^*), i = 1, \ldots, t; \\ (C, K_0) \leftarrow \mathsf{Enc}(\text{PK}); K_1 \leftarrow_{\text{R}} \{0,1\}^k; b \leftarrow_{\text{R}} \{0,1\}; \\ b' \leftarrow \mathcal{A}_2^{\mathsf{ShareDec}(\cdot, \mathsf{Enc}(\text{PK}))}(\text{PK}, \text{SK}_{\text{ID}_1^*}, \ldots, \text{SK}_{\text{ID}_t^*}, K_b, C) \end{array}\right]$$

Here, $\mathsf{ShareDec}(\cdot, \mathsf{Enc}(\text{PK}))$ denotes an oracle that given an input ID, computes a fresh ciphertext C using $\mathsf{Enc}(\text{PK})$ and returns $\mathsf{ShareDec}(\text{ID}, C)$ along with C. This captures the fact that the adversary may obtain decryption shares of fresh encryptions of known messages, and captures the security notion used in [16] with applications to secure voting. In the CCA setting, we provide $\mathcal{A}_2$ with oracle access to $\mathsf{ShareDec}(\cdot, \cdot)$, with the restriction that $\mathcal{A}_2$ is only allowed to query $\mathsf{ShareDec}(\cdot, \cdot)$ on ciphertexts different from the challenge ciphertext C. A threshold encryption scheme is said to be *indistinguishable against chosen plaintext attacks* (IND-CPA) if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{AdvThEnc}^{\mathcal{A}}(k)$ is a negligible function in k.

Decryption Consistency. We consider a notion of decryption consistency that for all ciphertexts C, there exists a unique value K such that for all (possibly malformed) $t+1$ decryption shares, the combining algorithm returns a value in $\{K, \perp\}$.

Theorem 1. *If* R_{PP} *is a one-way relation admitting a threshold extractable hash proof, then the threshold KEM shown in Figure 4 is* IND-CPA *secure.*

Proof. Given a PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the threshold encryption scheme, we construct a $\mathcal{B}$ that breaks the pseudorandomness of G as follows: on input (PP, u, K):

- Run $(\text{ID}_1^*, \ldots, \text{ID}_t^*) \leftarrow \mathcal{A}_1(1^k)$

Threshold PKE

(SHARING PHASE.) On input the security parameter 1^k and a threshold t, the dealer generates (PP, SP), runs $\mathsf{Setup}(\text{PP}, \text{SP}, 1^t) \to (\text{HK}, \text{MSK})$ and sets PK to be PP. A user with identity ID is given the share $\text{SK}_{\text{ID}} := \mathsf{ShareGen}(\text{MSK}, \text{ID})$.

(ENCRYPTION.) $\mathsf{Enc}(\text{PK})$: sample $(u, s) := \mathsf{SampR}(r)$, and return $(C, K) := (u, \mathsf{G}(s))$.

(DECRYPTION.) On input a ciphertext u, a user ID computes the decryption share $\sigma_{\text{ID}} := \mathsf{Priv}(\text{SK}_{\text{ID}}, u)$. Given $t+1$ decryption shares $\sigma_{\text{ID}_1}, \ldots, \sigma_{\text{ID}_{t+1}}$, the combining algorithm computes $s := \mathsf{Ext}(u, \sigma_{\text{ID}_1}, \ldots, \sigma_{\text{ID}_{t+1}})$ and returns $\mathsf{G}(s)$.

Fig. 4. Threshold encryption scheme from threshold hash proofs

- Run $\mathsf{SetupSim}(\text{PP}, \text{ID}_1^*, \ldots, \text{ID}_t^*)$ to get $(\text{HK}, \text{SK}_{\text{ID}_1^*}, \ldots, \text{SK}_{\text{ID}_t^*})$
- Output $\mathcal{A}_2((\text{PP}, \text{PK}), \text{SK}_{\text{ID}_1^*}, \ldots, \text{SK}_{\text{ID}_t^*}, K, u)$, simulating $\mathsf{ShareDec}(\cdot, \mathsf{Enc}(\text{PK}))$ using Pub.

It is easy to see that $\mathsf{AdvPRG}^{\mathcal{B}}(k) \approx \mathsf{AdvThEnc}^{\mathcal{A}}(k)$. □

5 Threshold Signature Schemes

A threshold signature scheme proceeds in two phases:

(SHARING PHASE.) The set-up algorithm $\mathsf{Setup}(\text{PP}, \text{SP}, 1^t)$ generates a verification VK and a master secret key MSK. Given an identity ID, the share generation algorithm computes an associated key SK_{ID}.

(SIGNATURE COMPUTATION PHASE.) The signature computation $\mathsf{ShareSign}(\cdot, \cdot)$ algorithm takes an identity ID and a message and computes the signature share for that identity using its secret key SK_{ID}. Moreover, there's a combining algorithm that takes any $t+1$ signature shares and outputs the actual signature σ.

Unforgeability. We define the advantage $\mathsf{AdvThSign}^{\mathcal{A}}(k)$ to be:

$$\Pr\left[\mathsf{Ver}(\text{PK}, \text{VK}, M^*, \sigma^*) = 1 : \begin{array}{l} (\text{ID}_1^*, \ldots, \text{ID}_t^*) \leftarrow \mathcal{A}_1(1^k); \\ (\text{VK}, \text{MSK}) \leftarrow \mathsf{Gen}(1^k, 1^t); \\ \text{SK}_{\text{ID}_i^*} \leftarrow \mathsf{ShareGen}(\text{MSK}, \text{ID}_i^*), i = 1, \ldots, t; \\ (M^*, \sigma^*) \leftarrow \mathcal{A}_2^{\mathsf{ShareSign}(\cdot,\cdot)}(\text{VK}, \text{SK}_{\text{ID}_1^*}, \ldots, \text{SK}_{\text{ID}_t^*}) \end{array}\right]$$

with the restriction that $\mathcal{A}_2$ never made a query to $\mathsf{ShareSign}(\cdot, \cdot)$ on the message M^*. A threshold signature scheme is said to be *existentially unforgeable* if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{AdvThSign}^{\mathcal{A}}(k)$ is a negligible function in k.

Construction. We assume that R_{PP} is efficiently verifiable and that there is a hash function H that maps the message space into instances of R_{PP}. The signature on a message M is a witness s such that $(\mathsf{H}(M), s) \in \mathsf{R}_{\text{PP}}$.

Threshold Signature Scheme

(SHARING PHASE.) On input the security parameter 1^k and a threshold t, the dealer generates (PP, SP), runs $\mathsf{Setup}(\text{PP}, \text{SP}, 1^t) \to (\text{HK}, \text{MSK})$ and sets VK to be PP. A user with identity ID is given the share $\text{SK}_{\text{ID}} := \mathsf{ShareGen}(\text{MSK}, \text{ID})$.

(SHARING PHASE.) On input the security parameter 1^k and a threshold t, the dealer generates (PP, SP), runs $\mathsf{Setup}(\text{PP}, \text{SP}, 1^t) \to (\text{HK}, \text{MSK})$ and sets VK to be PP. A user with identity ID is given the share $\text{SK}_{\text{ID}} := \mathsf{ShareGen}(\text{MSK}, \text{ID})$.

(SIGNATURE COMPUTATION PHASE.) On input a message M, the user ID computes $u := \mathsf{H}(M)$ and publishes the signature fragment $\sigma_{\text{ID}} := \mathsf{Priv}(\text{SK}_{\text{ID}}, u)$. Given $t+1$ signatures fragments $\sigma_{\text{ID}_1}, \ldots, \sigma_{\text{ID}_{t+1}}$, the signature is given by $\mathsf{Ext}(u, \sigma_{\text{ID}_1}, \ldots, \sigma_{\text{ID}_{t+1}})$.

(SIGNATURE VERIFICATION.) On input a key VK, a message M and a signature σ, the verification accepts iff $(\mathsf{H}(M), \sigma) \in \mathsf{R}_{\text{VK}}$.

Fig. 5. Threshold signatures from threshold hash proofs

Theorem 2. *If R_{PP} is a one-way relation admitting a threshold extractable hash proof, then the threshold signature shown in Figure 5 is existentially unforgeable in the random oracle model. Moreover, if the hash proof is publicly verifiable, then the signature scheme is robust.*

Proof. Given a PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the threshold signature scheme, we construct a $\mathcal{B}$ that breaks the one-wayness of R as follows: on input (PP, u):

- Run $(\text{ID}_1^*, \ldots, \text{ID}_t^*) \leftarrow \mathcal{A}_1(1^k)$
- Run $\mathsf{SetupSim}(\text{PP}, \text{ID}_1^*, \ldots, \text{ID}_t^*)$ to get $(\text{HK}, \text{SK}_{\text{ID}_1^*}, \ldots, \text{SK}_{\text{ID}_t^*})$
- Output $\mathcal{A}_2(\text{VK}, \text{SK}_{\text{ID}_1^*}, \ldots, \text{SK}_{\text{ID}_t^*})$

Suppose $\mathcal{A}_2$ requests for signatures on $M_1, \ldots, M_q$ and outputs a forgery on M^*. For each i, we sample $\mathsf{SampR}(r_i) := (u_i, s_i)$ and map $\mathsf{H}(M_i)$ to u_i for which we can compute any signature fragment using Pub. Finally, we map $\mathsf{H}(M^*)$ to u and thus a valid signature on M^* is a valid witness for u. It is easy to see that $\mathsf{AdvPRG}^{\mathcal{B}}(k) \approx \mathsf{AdvThSig}^{\mathcal{A}}(k) - q^2/2^k$ (where $q^2/2^k$ is an upper bound on the probability of a collision in the output of the random oracle). $\square$

6 Revocation Schemes

We define a revocation KEM:

(SHARING PHASE.**)** The set-up algorithm $\mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t)$ generates public keys HK and a master secret key MSK. Given an identity ID, the share generation algorithm computes an associated key $\mathsf{ShareGen}(\mathrm{MSK}, \mathrm{ID}) = \mathrm{SK}_{\mathrm{ID}}$.

(ENCRYPTION.**)** The encapsulation algorithm Enc takes PK and a set of t revoked users $\mathrm{ID}_1, \ldots, \mathrm{ID}_t$ generates (C, K), namely a random key K along with a ciphertext C.

(DECRYPTION.**)** The decapsulation algorithm Dec takes $\mathrm{SK}_{\mathrm{ID}}$ for any $\mathrm{ID} \neq \mathrm{ID}_1, \ldots, \mathrm{ID}_t$ and outputs K.

Semantic Security. We define the advantage $\mathsf{AdvBrEnc}^{\mathcal{A}}(k)$ to be:

$$\Pr\left[b = b' : \begin{array}{l} S = (\mathrm{ID}_1^*, \ldots, \mathrm{ID}_t^*) \leftarrow \mathcal{A}_1(1^k); \\ (\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathsf{Gen}(1^k, 1^t); \\ \mathrm{SK}_{\mathrm{ID}_i^*} \leftarrow \mathsf{ShareGen}(\mathrm{MSK}, \mathrm{ID}_i^*), i = 1, \ldots, t; \\ (C, K_0) \leftarrow \mathsf{Enc}(\mathrm{PK}, S); K_1 \leftarrow_{\mathrm{R}} \{0,1\}^k; b \leftarrow_{\mathrm{R}} \{0,1\}; \\ b' \leftarrow \mathcal{A}_2(\mathrm{PK}, \mathrm{SK}_{\mathrm{ID}_1^*}, \ldots, \mathrm{SK}_{\mathrm{ID}_t^*}, K_b, C) \end{array}\right]$$

In the CCA setting, we provide $\mathcal{A}_2$ with oracle access to $\mathsf{Dec}(\cdot, \cdot)$, with the restriction that $\mathcal{A}_2$ is only allowed to query $\mathsf{Dec}(\cdot, \cdot)$ on ciphertexts different from the challenge ciphertext. A revocation scheme is said to be *indistinguishable against chosen plaintext attacks* (IND-CPA) if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{AdvBrEnc}^{\mathcal{A}}(k)$ is a negligible function in k.

Theorem 3. *If* R_{PP} *is a one-way relation admitting a threshold extractable hash proof, then the broadcast KEM shown in Figure 6 is* IND-CPA *secure.*

Proof. Given a PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the broadcast KEM, we construct a $\mathcal{B}$ that breaks the pseudorandomness of G as follows: on input (PP, u, K):

- Run $(\mathrm{ID}_1^*, \ldots, \mathrm{ID}_t^*) \leftarrow \mathcal{A}_1(1^k)$
- Run $\mathsf{SetupSim}(\mathrm{PP}, \mathrm{ID}_1^*, \ldots, \mathrm{ID}_t^*)$ to get $(\mathrm{HK}, \mathrm{SK}_{\mathrm{ID}_1^*}, \ldots, \mathrm{SK}_{\mathrm{ID}_t^*})$
- Set $C := (u, \mathsf{Priv}(\mathrm{SK}_{\mathrm{ID}_1^*}, u), \ldots, \mathsf{Priv}(\mathrm{SK}_{\mathrm{ID}_t^*}, u))$.
- Output $\mathcal{A}_2((\mathrm{PP}, \mathrm{PK}), \mathrm{SK}_{\mathrm{ID}_1^*}, \ldots, \mathrm{SK}_{\mathrm{ID}_t^*}, K, C)$.

It is easy to see that $\mathsf{AdvPRG}^{\mathcal{B}}(k) \approx \mathsf{AdvBrEnc}^{\mathcal{A}}(k)$. □

7 Instantiations for the Diffie-Hellman Relation

We consider a family of groups $\mathbb{G}$ of prime order q that admits a bilinear map. The secret parameter is a random $\alpha \leftarrow_{\mathrm{R}} \mathbb{Z}_q$ and the public parameter PP is given by (g, g^α)

Revocation PKE

(SHARING PHASE.) On input the security parameter 1^k and a revocation threshold t, the dealer generates (PP, SP), runs $\mathsf{Setup}(\text{PP}, \text{SP}, 1^t) \to (\text{HK}, \text{MSK})$ and sets the public key PK to be (PP, HK). A user with identity ID is given the key $\text{SK}_{\text{ID}} := \mathsf{ShareGen}(\text{MSK}, \text{ID})$.

(ENCRYPTION.) In order to revoke users $\text{ID}_1, \ldots, \text{ID}_t$, $\mathsf{Enc}(\text{PK})$: sample $(u, s) := \mathsf{SampR}(r)$, compute $\tau_i := \mathsf{Pub}(\text{HK}, \text{ID}_i, u, r)$ for $i = 1, \ldots, t$ and return $(C, K) := ((u, \tau_1, \ldots, \tau_t), \mathsf{G}(s))$.

(DECRYPTION.) Any user ID not in the revoked set $\{\text{ID}_1, \ldots, \text{ID}_t\}$ may decrypt a ciphertext $C := (u, \tau_1, \ldots, \tau_t)$ as follows: compute $s := \mathsf{Ext}(u, \tau_1, \ldots, \tau_t, \mathsf{Priv}(\text{SK}_{\text{ID}}, u))$ and output $\mathsf{G}(s)$.

Fig. 6. Revocation scheme from threshold hash proofs

for a random $g \leftarrow_{\text{R}} \mathbb{G}$ and a random $\alpha \leftarrow_{\text{R}} \mathbb{Z}_q$. We consider the Diffie-Hellman relation

$$\mathsf{R}_{\text{PP}}^{\mathsf{dh}} = \left\{ (u, s) \in \mathbb{G} \times \mathbb{G} : s = u^{\alpha} \right\}$$

Note that $\mathsf{R}_{\text{PP}}^{\mathsf{dh}}$ is efficiently verifiable by computing a pairing. The associated sampling algorithm SampR picks a $r \leftarrow_{\text{R}} \mathbb{Z}_q$ and outputs $(g^r, g^{\alpha r})$.

Hard-core bits. Next, we explain how to obtain hard-core bits for $\mathsf{R}_{\text{PP}}^{\mathsf{dh}}$ under various assumptions.

- The CDH assumption [1] asserts that computing g^{ab} given (g, g^a, g^b) is hard on average; here, we may extract a single hard-core bit from s using $\mathsf{GL}(s)$.
- The Bilinear DDH (BDDH) assumption [7] asserts that $e(g, g)^{abc}$ is pseudorandom given g, g^a, g^b, g^c where g, g^a, g^b, g^c are random elements of a bilinear group. Under BDDH, we may extract a linear number of hard-core bits from s using:

  $$\mathsf{G}_{\text{PP}}^{\mathsf{bddh}}(s) := e(s, g^{\gamma}) \quad \left(\Rightarrow \mathsf{G}_{\text{PP}}^{\mathsf{bddh}}(g^{\alpha r}) = e(g, g)^{\alpha \gamma r} \right)$$

  where PP is now given by $(g, g^{\alpha}, g^{\gamma})$. In addition, we may improve efficiency by pre-computing the pairing and setting PP to be $(g, g^{\alpha}, e(g, g^{\gamma}))$ and computing $\mathsf{G}_{\text{PP}}^{\mathsf{bddh}}(g^r) := e(g, g^{\gamma})^r$. This construction extends naturally to the Gap Hashed DH assumption [35].

Threshold hash proof system. Fix the parameters $(\text{PP}, \text{SP}) = ((g, g^{\alpha}), \alpha)$; this also fixes a group $\mathbb{G}$ of prime order q. The tag space is given by $\mathbb{F}_q \setminus \{0\}$.

(KEY GENERATION.) Pick $a_1, \ldots, a_t \leftarrow_R \mathbb{Z}_q$ and set $f(x) := \alpha + a_1 x + \cdots a_t$.

- $\mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t)$ returns $\mathrm{HK} := (g^{a_1}, \ldots, g^{a_t})$ and $\mathrm{MSK} := f(x)$
- $\mathsf{ShareGen}(\mathrm{MSK}, \mathrm{TAG})$ returns $\mathrm{SK}_{\mathrm{TAG}} := f(\mathrm{TAG}) \in \mathbb{Z}_q$.

(PUBLIC/PRIVATE EVALUATION.) $\mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$ is given by $u^{f(\mathrm{TAG})} = (g^{f(\mathrm{TAG})})^r$ where $u := g^r$

- $\mathsf{Pub}(\mathrm{HK}, u, r)$ returns $\left(g^\alpha \cdot \prod_{i=1}^t (g^{a_i})^{\mathrm{TAG}^i}\right)^r$.
- $\mathsf{Priv}(\mathrm{SK}_{\mathrm{TAG}}, u)$ returns $u^{\mathrm{SK}_{\mathrm{TAG}}}$.

($(t+1)$-EXTRACTION.) Given $u, \mathrm{TAG}_1, \ldots, \mathrm{TAG}_{t+1}$, we have $(u, u^{f(0)}) \in \mathsf{R}_{\mathrm{PP}}$. In addition, we may write $f(0) = \sum_{i=1}^{t+1} L_i \cdot f(\mathrm{TAG}_i)$ where $L_i \in \mathbb{F}_q$ are the Lagrangian coefficients which may be efficiently computed given $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_{t+1}$. This means $u^{f(0)} = \prod_{i=1}^{t+1} u^{L_i \cdot f(\mathrm{TAG}_i)}$.

- $\mathsf{Ext}(u, \tau_1, \ldots, \tau_{t+1})$ returns $\prod_{i=1}^{t+1} \tau_i^{L_i}$.

(t-SIMULATION.) Pick $\gamma_1, \ldots, \gamma_t \leftarrow_R \mathbb{Z}_q$. This uniquely determines a degree t polynomial $f(x) = \alpha + a_1 x + \cdots + a_t x^t$ such that $f(\mathrm{TAG}_i) = \gamma_i$ for $i = 1, \ldots, t$. Moreover, $a_1, \ldots, a_t$ are given by the solution to the following linear system:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & \mathrm{TAG}_1 & \cdots & \mathrm{TAG}_1^t \\ \vdots & \vdots & & \vdots \\ 1 & \mathrm{TAG}_t & \cdots & \mathrm{TAG}_t^t \end{bmatrix} \begin{bmatrix} \alpha \\ a_1 \\ \vdots \\ a_t \end{bmatrix} = \begin{bmatrix} \alpha \\ \gamma_1 \\ \vdots \\ \gamma_t \end{bmatrix}$$

In particular, each of $a_1, \ldots, a_t$ may be written as a linear combination of $\alpha, \gamma_1, \ldots, \gamma_t$ (where the coefficients are efficiently computable given $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_t$) and therefore each of $g^{a_1}, \ldots, g^{a_t}$ may be written as a product of $g^\alpha, g^{\gamma_1}, \ldots, g^{\gamma_t}$ raised to the appropriate powers.

- $\mathsf{SetupSim}(\mathrm{PP}, 1^t)$ returns $\mathrm{HK} := (g^{a_1}, \ldots, g^{a_t})$ as computed above and $(\mathrm{SK}_{\mathrm{TAG}_1}, \ldots, \mathrm{SK}_{\mathrm{TAG}_t}) := (\gamma_1, \ldots, \gamma_t)$

Remark 1. Instead of setting $\mathrm{HK} := (g^{a_1}, \ldots, g^{a_t})$, we may also set $\mathrm{HK} := (g^{f(1)}, \ldots, g^{f(t)})$. Public evaluation and t-simulation then proceed via Lagrange interpolation in the exponent.

Public verifiability. Given $(\mathrm{HK}, u, \mathrm{TAG}, \tau)$, checking that $\tau = \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$ corresponds exactly to verifying that $(g, g^{f(\mathrm{TAG})}, u, \tau)$ is a valid DDH tuple. Given HK and TAG, we may compute $g^{f(\mathrm{TAG})}$ using $\left(g^\alpha \cdot \prod_{i=1}^t (g^{a_i})^{\mathrm{TAG}^i}\right)$. This implies public verifiability in bilinear groups.

For general groups that do not admit a bilinear pairing, we may realize public verifiability in the random oracle model following [41, Section 4.3] (also [28]). That is, we append to $\mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$ non-interactive zero-knowledge proof that $(g, g^{f(\mathrm{TAG})}, u, u^{f(\mathrm{TAG})})$

is a valid DDH tuple. Such a proof is derived by applying the Fiat-Shamir heuristic to Chaum-Pedersen Σ-protocol for the langugage comprising valid DDH tuples [13]; soundness of the verification algorithm follows immediately from soundness of the Σ-protocol. Handling public and private evaluation requires more care, and we proceed differently depending on the application:

- In the application to threshold cryptosystems, we instantiate the Chaum-Pedersen protocol so that there is an efficient prover given $f(\text{TAG})$ as a witness. This guarantees efficient private evaluation. For public evaluation, we rely on the zero-knowledge simulator, which in turn requires programming the random oracle. This is not an issue since we only rely on public evaluation in the proof of security.
- In the application to revocation cryptosystems, we instantiate the Chaum-Pedersen protocol so that there is an efficient prover given r such that $u = g^r$ as a witness. This guarantees efficient public evaluation. For private evaluation, we rely on the zero-knowledge simulator, which again requires programming the random oracle. This is not an issue since in the decryption algorithm, Ext ignores non-interactive zero-knowledge proof.

8 Instantiations from Hardness of Factoring

Fix a Blum integer $N = PQ$ for safe primes $P, Q \equiv 3 \pmod 4$ (such that $P = 2p+1$ and $Q = 2q + 1$ for primes p, q). Following [33], we work over the cyclic group of signed quadratic residues, given by the quotient group $\mathbb{QR}_N^+ := \mathbb{QR}_N / \pm 1$. $\mathbb{QR}_N^+$ is a cyclic group of order pq and is efficiently recognizable (by verifying that the Jacobi symbol is $+1$). In addition, the map $x \mapsto x^2$ is a permutation over $\mathbb{QR}_N^+$. Furthermore, assuming that factoring is hard on average and that safe primes are dense, the family of permutations $x \mapsto x^2$ (indexed by N) acting on the groups $\mathbb{QR}_N^+$ is one-way.

In our constructions, the public parameter PP comprises (N, g), where N is a random $2k$-bit Blum integer and g is chosen uniformly from $\mathbb{QR}_N^+$. We will henceforth assume that g is a generator for $\mathbb{QR}_N^+$, which happens with probability $1 - O(1/\sqrt{N})$. For signatures, we consider the relation

$$\mathsf{R}_{\text{PP}}^{\mathsf{sqr}} = \left\{ (u, s) \in \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ : u = s^2 \right\}$$

For encryption, we consider the relation:

$$\mathsf{R}_{\text{PP}}^{\mathsf{isqr}} = \left\{ (u, s) \in \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ : u = s^{2^k} \right\}$$

Here, we focus on the latter relation. The associated sampling algorithm SampR picks a random $r \in [(N-1)/4]$ and outputs $(g^{2^k r}, g^r)$. Note that the output distribution is statistically close to the uniform distribution over $\mathbb{QR}_N^+$ whenever g is a generator. Using the Blum-Blum-Shub (BBS) pseudorandom generator [4], we may extract k hard-core bits from s that are pseudorandom even given u, that is:

$$\mathsf{G}_{\text{PP}}^{\mathsf{bbs}}(s) := (\mathsf{lsb}_N(s), \mathsf{lsb}_N(s^2), \ldots, \mathsf{lsb}_N(s^{2^{k-1}}))$$

Basic idea. The next claim shows that we can do Shamir secret sharing over a composite modulus:

Claim (implicit in [40]). Fix two primes $p < q$. For any $t < p$ and any $t+1$ distinct values $v_1, \ldots, v_{t+1}$ in $\{1, 2, \ldots, p\}$, the map $\psi : (a_0, a_1, \ldots, a_t) \mapsto (f(v_1), f(v_2), \ldots, f(v_{t+1}))$ where $f(x) := a_0 + a_1 x + \cdots + a_t x^t$ defines a bijection from $\mathbb{Z}_{pq}^{t+1}$ to $\mathbb{Z}_{pq}^{t+1}$.

Proof (sketch). That ψ is injective follows from the fact that any polynomial of degree t over $\mathbb{Z}_{pq}$ that vanishes at the $t+1$ distinct values $v_1, \ldots, v_{t+1}$ must be identically 0 modulo p and modulo q and thus identically 0 modulo pq (by the Chinese remainder theorem). That ψ is surjective follows via Lagrange polynomial interpolation (since the pairwise differences $v_i - v_j$ are all coprime with pq). □

Threshold hash proof system. Fix the parameters $(\text{PP}, \text{SP}) = (N, \phi(N))$. The tag space is given by $Z_{\sqrt{N}/4}$. Note that $\sqrt{N}/4 \leq \min\{p, q\}$ so every valid tag is coprime with $\phi(N)/4$. SampR takes an additional $g \in \mathbb{QR}_N^+$ which is provided as part of HK, and $\mathsf{SampR}(r) := (u, s)$ where $s = g^{2^{tk} r}, u = s^{2^k} = g^{2^{(t+1)k} r}$.

(KEY GENERATION.**)** Pick $a_1, \ldots, a_t \leftarrow_{\text{R}} \mathbb{Z}_{\phi(N)/4}$ and set $f(x) := 2^{-(t+1)k} + a_1 x + \cdots a_t x^t$. In addition, pick $g \leftarrow_{\text{R}} \mathbb{QR}_N^+$.

- $\mathsf{Setup}(\text{PP}, \text{SP}, 1^t)$ returns $\text{HK} := (g, g^{2^{(t+1)k} a_1}, \ldots, g^{2^{(t+1)k} a_t})$ and $\text{MSK} := (f(x), \phi(N))$.
- $\mathsf{ShareGen}(\text{MSK}, \text{TAG})$ returns $\text{SK}_{\text{TAG}} := f(\text{TAG}) \pmod{\phi(N)/4}$.

(PUBLIC/PRIVATE EVALUATION.**)** $\mathcal{H}_{\text{HK}}(\text{TAG}, u)$ is given by $u^{f(\text{TAG})} = (g^{2^{(t+1)k} f(\text{TAG})})^r$ where $u := g^{2^{(t+1)k} r}$

- $\mathsf{Pub}(\text{HK}, u, r)$ returns $\left(g \cdot \prod_{i=1}^{t} (g^{2^{(t+1)k} a_i})^{\text{TAG}^i}\right)^r$.
- $\mathsf{Priv}(\text{SK}_{\text{TAG}}, u)$ returns $u^{\text{SK}_{\text{TAG}}}$.

($(t+1)$-EXTRACTION.**)** Given $\text{TAG}_1, \ldots, \text{TAG}_{t+1}$, we may efficiently compute the fractional Lagrangian coefficients $L_1, \ldots, L_{t+1}$ such that $f(0) = \sum_{i=1}^{t+1} L_i \cdot f(\text{TAG}_i) \pmod{\phi(N)/4}$. In addition, we may compute

$$D := \operatorname{lcm}\Big\{\prod\nolimits_{j \neq i} (\text{TAG}_i - \text{TAG}_j) : i \in [t+1]\Big\}.$$

We make the following observations: (1) $DL_1, \ldots, DL_{t+1}$ are all integers, so we may compute $u^{D \cdot f(0)} = \prod_{i=1}^{t+1} \tau_i^{DL_i}$; (2) let 2^c be the highest power of 2 that divides D, and we have $c \leq kt$ (since $|\text{TAG}_i - \text{TAG}_j| \leq 2^k$); and (3) given $u = s^{2^k}$ and $u^{2^{kt-c} D \cdot f(0)} = s^{2^{-c} D}$, we may efficiently recover s using Shamir's "GCD in the exponent" algorithm [39], since $\gcd(2^k, 2^{-c} D) = 1$.

- $\mathsf{Ext}(u, \tau_1, \ldots, \tau_{t+1})$ returns s as computed above.

(t-SIMULATION.) Pick $\gamma_1, \ldots, \gamma_t \leftarrow_R \mathbb{Z}_{N/4}$.[1] This uniquely determines a degree t polynomial $f(x) = 2^{-(t+1)k} + a_1 x + \cdots + a_t x^t$ such that $f(\text{TAG}_i) = \gamma_i \pmod{\phi(N)/4}$ for $i = 1, \ldots, t$. Moreover, we $a_1, \ldots, a_t$ are given by the solution to the following linear system:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & \text{TAG}_1 & \cdots & \text{TAG}_1^t \\ \vdots & \vdots & & \vdots \\ 1 & \text{TAG}_t & \cdots & \text{TAG}_t^t \end{bmatrix} \begin{bmatrix} 1 \\ 2^{(t+1)k} a_1 \\ \vdots \\ 2^{(t+1)k} a_t \end{bmatrix} = \begin{bmatrix} 1 \\ 2^{(t+1)k} \gamma_1 \\ \vdots \\ 2^{(t+1)k} \gamma_t \end{bmatrix}$$

Let $D := \prod_{1 \le i < j \le t} (\text{TAG}_i - \text{TAG}_j)$ (the determinant of the Vandermonde matrix on the left). Then, we may efficiently compute the integer values $D \cdot 2^{(t+1)k} a_1, \ldots, D \cdot 2^{(t+1)k} a_t$ (given $\gamma_1, \ldots, \gamma_t$ and $\text{TAG}_1, \ldots, \text{TAG}_t$) without computing any modular inverse.

- $\mathsf{SetupSim}(\text{PP}, 1^t)$ picks $\tilde{g} \leftarrow_R \mathbb{QR}_N^+$ and returns $(\text{SK}_{\text{TAG}_1}, \ldots, \text{SK}_{\text{TAG}_t}) := (\gamma_1, \ldots, \gamma_t)$ and $\text{HK} := (\tilde{g}^D, \tilde{g}^{D \cdot 2^{(t+1)k} a_1}, \ldots, \tilde{g}^{D \cdot 2^{(t+1)k} a_t})$.

Public verifiability. Following [28, Section 4], it suffices to construct a Σ-protocol for DDH-tuples over $\mathbb{QR}_N^+$, that is, (g, g^B, u, u^B). The honest sender is given the witness r such that $u = g^r$, picks $s \in [N^3]$ and sends $(g_0, g_1) := (g^s, g^{Bs})$. Upon receiving a challenge $e \in [N/4]$, it responds with $z := s + re$. The analysis is entirely analogous to that in [28].

9 Chosen-Ciphertext Security

9.1 Broadcast CCA

In this section, we construct revocation schemes secure against CCA attacks.

Public verifiability, signatures and random oracles. As stated, the construction also requires public verifiability and a one-time signature scheme. For factoring-based instantiations and Diffie-Hellman in general groups, we already rely on a random oracle to implement public verifiability, so we may as well also rely on the random oracle to instantiate an efficient signature scheme.[2] For Diffie-Hellman in bilinear groups (which satisfy public verifiability without random oracles), we can avoid the use of signatures and instead use a TCR – the ciphertext is given by (u, τ) where $\tau := \mathsf{Pub}(\text{PK}, \text{TAG}, u, r)$ and $\text{TAG} := \mathsf{TCR}(u)$, and as such, we obtain efficient constructions without random oracles.

[1] This yields a distribution that is statistically close to picking $\gamma_1, \ldots, \gamma_t \leftarrow_R \mathbb{Z}_{\phi(N)/4}$.

[2] The reason we need a signature scheme is that the addition of the non-interactive proof of membership in the random oracle to provide public verifiability means that (TAG, u) no longer uniquely determines an accepting proof.

Revocation PKE

(SHARING PHASE.) On input the security parameter 1^k and a revocation threshold t, the dealer generates (PP, SP), runs $\mathsf{Setup}(\text{PP}, \text{SP}, 1^{t+1}) \to (\text{HK}, \text{MSK})$ and sets the public key PK to be (PP, HK). A user with identity ID is given the key $\text{SK}_{\text{ID}} := \mathsf{ShareGen}(\text{MSK}, \text{ID})$.

(ENCRYPTION.) In order to revoke users $\text{ID}_1, \ldots, \text{ID}_t$, $\mathsf{Enc}(\text{PK})$:

1. generate a verification VKSIG for a one-time signature scheme;
2. sample $(u, s) := \mathsf{SampR}(r)$;
3. compute $\tau := \mathsf{Pub}(\text{PK}, \text{VKSIG}, u, r)$ and $\tau_i := \mathsf{Pub}(\text{HK}, \text{ID}_i, u, r)$ for $i = 1, \ldots, t$;
4. compute the signature σ on $(u, \tau, \tau_1, \ldots, \tau_t)$;
5. return $(C, K) := ((\text{VKSIG}, u, \tau, \tau_1, \ldots, \tau_t, \sigma), \mathsf{G}(s))$.

(DECRYPTION.) Any user ID not in the revoked set $\{\text{ID}_1, \ldots, \text{ID}_t\}$ may decrypt a ciphertext $C := (\text{VKSIG}, u, \tau, \tau_1, \ldots, \tau_t, \sigma)$ as follows:

1. verify proofs $\tau, \tau_1, \ldots, \tau_t$ and signature σ; output $\perp$ if any of these tests fails;
2. compute $s := \mathsf{Ext}(u, \tau, \tau_1, \ldots, \tau_t, \mathsf{Priv}(\text{SK}_{\text{ID}}, u))$ and output $\mathsf{G}(s)$.

Fig. 7. CCA-secure revocation scheme from threshold hash proofs

Theorem 4. *If R_{PP} is a one-way relation admitting a threshold extractable hash proof with public verifiability, then the above revocation scheme is* IND-CCA *secure.*

Proof (sketch). In the following, we write $(u^*, s^*) = \mathsf{SampR}(r), C^* = (u^*, \tau^*), K_0^*, K_1^*$ to denote the challenge ciphertext and keys chosen by the IND-CCA experiment, and we set VKSIG^* to denote the verification key used in computing C^*. We proceed via a sequence of games. We start with Game 0, where the challenger proceeds like in the standard IND-CCA game (i.e, K_0^* is a real key and K_1^* is a random key) and end up with a game where both K_0^* and K_1^* are chosen uniformly at random. Then, we show that all games are indistinguishable under the assumption that $\mathsf{G}(s)$ is pseudorandom even given u.

GAME 1: ELIMINATING COLLISIONS. We replace the decapsulation mechanism Dec with Dec' that outputs $\perp$ on inputs $(\text{VKSIG}, u, \tau, \sigma)$ such that $\text{VKSIG} = \text{VKSIG}^*$ but otherwise proceeds like Dec. We show that Games 0 and 1 are computationally indistinguishable, by arguing that Dec and Dec' essentially agree on all inputs. This follows readily from the security of the one-time signature.

GAME 2: DECAPSULATION WITH SetupSim. We modify the IND-CCA experiment from Game 1, we generate the keys using $\mathsf{SetupSim}(\text{PP}, \text{ID}_1^*, \ldots, \text{ID}_t^*, \text{VKSIG}^*)$, and we replace $\mathsf{Dec}'(\text{ID}, \cdot)$ with $\mathsf{Dec}^*(\text{ID}, \cdot)$:

On input $(\text{VKSIG}, u, \tau, \tau_1, \ldots, \tau_t, \sigma)$:
- if $\text{VKSIG} = \text{VKSIG}^*$, return $\perp$.
- if σ or any of $\tau, \tau_1, \ldots, \tau_t$ fails to verify, return $\perp$.
- compute $s := \mathsf{Ext}(u, \tau, \tau_1, \ldots, \tau_t, \mathsf{Priv}(\text{SK}_{\text{VK}^*}, u))$ and output $\mathsf{G}(s)$.

Here, we use the fact that in both Dec' and Dec^*, we run Ext with $t+2$ valid proofs and therefore it outputs the correct witness s.

GAME 3: ENCAPSULATION WITH Priv. We compute all $t+1$ proofs $\tau, \tau_1, \ldots, \tau_t$ in C^* using Priv instead of Pub and sign using the secret key corresponding to VK^*.

GAME 4: REPLACING $\mathrm{G}(s^*)$ WITH RANDOM. We generate K_0^* at random from $\{0,1\}^k$ instead of using $\mathsf{G}(s^*)$ (recall here that $(u^*, s^*) = \mathsf{SampR}(r)$). Observe that in Game 3, we never use knowledge of the witness s^* or randomness r associated with u^*. It follows from the pseudorandomness of G that Games 3 and 4 are computationally indistinguishable. Specifically, we may transform any distinguisher for Games 3 and 4 into a distinguisher K_0^* and $\mathsf{G}(s^*)$.

We conclude by observing that in Game 4, both K_0^* and K_1^* are identically distributed, so the probability that $b' = b$ is exactly $1/2$. □

9.2 Threshold CCA

In this section, we construct threshold encryption schemes secure against CCA attacks. The main technical difficulty is as follows: the simulator knows $f(\text{TAG}_1^*), \ldots, f(\text{TAG}_t^*)$, and needs to be able to compute $\mathcal{H}_{\text{HK}}(\text{TAG}, u) = u^{f(\text{TAG})}$ given any TAG, u. This is in order to simulate the $\mathsf{ShareDec}(\text{TAG}, \cdot)$, the decryption share for some user TAG. To handle this, we modify our basic threshold encryption scheme in three ways:

- The first modification is to add to the ciphertext which contains the instance u, a publicly-verifiable 1-threshold extractable hash proof —or, an all-but-one extractable hash proof following the terminology in [42]— for the relation $(u, u^{f(0)})$. For this, we need to turn to either pairings or the random oracle model. (Similar issues arise even in previous discrete-log based schemes not based on pairings.) This way, simulator will be able to recover the $t+1$ values $u^{f(0)}, u^{f(\text{TAG}_1^*)}, \ldots, u^{f(\text{TAG}_t^*)}$ for any well-formed ciphertext.
- Next, using interpolation, we can recover $u^{Df(\text{TAG})}$, where D is some factor we use to clear out the fractional Lagrangian coefficients. In the discrete-log based instantiations, we may compute D^{-1} and thus recover $u^{f(\text{TAG})}$. In the factoring-based instantation, we will modify the hash function $\mathcal{H}_{\text{HK}}(\text{TAG}, u)$ to be $u^{Df(\text{TAG})}$; as such, we can only support a fixed identity space of polynomial size, since we need to compute D in advance. Similarly, we will need to modify Ext so that it computes $u^{D^2 f(0)}$ via Lagrange interpolation and s from both $u^{D^2 f(0)}$ and u via Shamir's "GCD in the exponent" algorithm.
- Finally, we modify $\mathsf{ShareDec}(\cdot)$ so that it will only output the decryption share if the 1-threshold extractable hash proof verifies properly.

The details are deferred to the full version of this paper.

Acknowledgments.

I would like to thank Dan Boneh, Mario Szegedy and Moti Yung for insightful discussions. I am also very grateful to the anonymous referees for detailed and constructive feedback. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defense, or the UK Government.

References

[1] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *CT-RSA*, pages 143–158, 2001.

[2] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[3] R. Bendlin and I. Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, pages 201–218, 2010.

[4] L. Blum, M. Blum, and M. Shub. Comparison of two pseudo-random number generators. In *CRYPTO*, pages 61–78, 1982.

[5] M. Blum and S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In *CRYPTO*, pages 289–302, 1984.

[6] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *STOC*, pages 103–112, 1988.

[7] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[8] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[9] D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In *CT-RSA*, 2006.

[10] X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. In *ACM CCS*, pages 320–329, 2005.

[11] R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *EUROCRYPT*, pages 90–106, 1999.

[12] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[13] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO*, pages 89–105, 1992.

[14] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25, 1998.

[15] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002.

[16] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography*, pages 119–136, 2001.

[17] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *FOCS*, pages 427–436, 1992.

[18] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *STOC*, pages 522–533, 1994.

[19] Y. Desmedt. Society and group oriented cryptography: A new concept. In *CRYPTO*, pages 120–127, 1987.
[20] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO*, pages 307–315, 1989.
[21] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures (extended abstract). In *CRYPTO*, pages 457–469, 1991.
[22] Y. Dodis and N. Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Public Key Cryptography*, pages 100–115, 2003.
[23] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO*, pages 480–491, 1993.
[24] P.-A. Fouque and D. Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *ASIACRYPT*, pages 351–368, 2001.
[25] P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography*, pages 90–104, 2000.
[26] Y. Frankel, P. Gemmell, and M. Yung. Witness-based cryptographic program checking and robust function sharing. In *STOC*, pages 499–508, 1996.
[27] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *EUROCRYPT*, pages 354–371, 1996.
[28] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In *CRYPTO*, 1996.
[29] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT*, pages 295–310, 1999.
[30] R. Gennaro, S. Halevi, H. Krawczyk, and T. Rabin. Threshold RSA for dynamic and ad-hoc groups. In *EUROCRYPT*, pages 88–107, 2008.
[31] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32, 1989.
[32] D. Hofheinz and E. Kiltz. Practical chosen ciphertext secure encryption from factoring. In *EUROCRYPT*, pages 313–332, 2009.
[33] D. Hofheinz and E. Kiltz. The group of signed quadratic residues and applications. In *CRYPTO*, pages 637–653, 2009.
[34] J. Katz and M. Yung. Threshold cryptosystems based on factoring. In *ASIACRYPT*, pages 192–205, 2002.
[35] E. Kiltz. Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie-Hellman. In *Public Key Cryptography*, pages 282–297, 2007.
[36] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography*, pages 1–20, 2000.
[37] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.
[38] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO*, pages 433–444, 1991.
[39] A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Trans. Comput. Syst.*, 1(1):38–44, 1983.
[40] V. Shoup. Practical threshold signatures. In *EUROCRYPT*, pages 207–220, 2000.
[41] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *J. Cryptology*, 15(2):75–96, 2002.
[42] H. Wee. Efficient chosen-ciphertext security via extractable hash proofs. In *CRYPTO*, pages 314–332, 2010.