On Actively Secure Fine-Grained Access Structures from Isogeny Assumptions

Fabio Campos[1,2] and Philipp Muth[3]

[1] RheinMain University of Applied Sciences, Wiesbaden, Germany
[2] Radboud University, Nijmegen, The Netherlands
campos@sopmac.de
[3] Technische Universität Darmstadt, Germany
philipp.muth@tu-darmstadt.de

Abstract. We present an actively secure threshold scheme in the setting of Hard Homogeneous Spaces (HHS) which allows fine-grained access structures. More precisely, we elevate a passively secure isogeny-based threshold scheme to an actively secure setting. We prove the active security and simulatability of our advanced schemes. By characterising the necessary properties, we open our schemes to a significantly wider field of applicable secret sharing schemes. Furthermore, we show that Shamir's scheme has our generalised properties, and thereby our approach truly represents a less restrictive generalisation.

Keywords: post-quantum cryptography · isogeny-based cryptography · threshold cryptography

1 Introduction

The principal motivation for a secret sharing scheme is to split private information into fragments and securely distribute these shares among a set of shareholders. Then, any collaborating set with a sufficient number of participants is able to reconstruct the shared private information, while the secret remains confidential to any unauthorised, that is not sufficiently large, subset of shareholders.

Since their introduction in the 1970s by Blakley [4] and Shamir [15], the field of secret sharing schemes, information theoretic and computational, has been studied extensively. In previous years, due to applications in blockchain and other scenarios, the interest in new developments and applications for secret sharing schemes has increased.

Post-quantum schemes have, however, only received little attention with respect to secret sharing. Recently, De Feo and Meyer [11] proposed a key exchange mechanism and a signature scheme making use of isogeny based public key cryptography for which the secret key is stored in a Shamir shared way. Their

Author list in alphabetical order; see https://www.ams.org/profession/leaders/culture/CultureStatement04.pdf.

approach enables decapsulation for the key exchange mechanism and signing for the signature scheme in a round-robin way without reconstructing the secret key in clear for any sufficiently large set of shareholders. Yet in applying Shamir's secret sharing scheme they restrict themselves to simple threshold access structures. Furthermore, their protocols are only passively secure, in that while a misbehaving shareholder cannot obtain information on the secret key shares of other shareholders participating in a decapsulation or a signing execution via maliciously formed inputs, his deviation from the protocol cannot be detected.

We aim to tackle both caveats by proposing an actively secure isogeny based key exchange mechanism, for which the secret key is secret shared by a trusted dealer. We further transform the key exchange mechanism into an actively secure signature scheme with shared secret key.

Our Contribution. Our contribution is manifold. First, we transfer the active security measures outlined in [2] from their setting of full engagement protocols to a setting of threshold secret sharing. We thereby open the active security measures to a wider field of application and improve upon their efficiency significantly. Second, we apply the adapted active security measures to propose an actively secure key exchange mechanism with secret shared secret key. Third, we present an actively secure signature scheme by applying a Fiat-Shamir transform to our key exchange mechanism. And fourth, we expand our key encapsulation mechanism and our signature scheme to a wider field of secret sharing schemes. For that we characterise the necessary properties for a secret sharing scheme and give several examples of compatible schemes.

Related work. Secret sharing schemes were first introduced by Blakley [4] and Shamir [15]. In both their approaches, secrets from the secret space $\mathbb{Z}_p := \mathbb{Z} \bmod p$ for prime p are shared by distributing interpolation points of randomly sampled polynomials. Damgård and Thorbek [9] presented a secret sharing scheme with secret space $\mathbb{Z}$. Thorbek [18] later improved their scheme. Yet their scheme is only computationally confidential, compared to the information theoretical confidentiality of Shamir and Blakley's schemes. Tassa [17] opened Shamir's scheme to a more general application by utilising the derivatives of the sharing polynomial to construct a hierarchical access structure,. These basic secret sharing schemes rely on the dealer providing honestly generated shares to the shareholders. Verifiable secret sharing schemes eliminate this drawback by providing the shareholders with the means to verify the correctness of the received shares with varying overhead. Examples of these are [1,13,16]. With minor efficiency losses, Herranz and Sáez [12] were able to achieve verifiable secret sharing for generalised access structures. Traverso et al. [19] proposed an approach for evaluating arithmetic circuits on secret shared in Tassa's scheme, that also enabled auditing the results. Cozzo and Smart [7] investigated the possibility of constructing shared secret schemes based on the Round 2 candidate signature schemes in the NIST standardization process[4]. Based on CSI-FiSh [3], De Feo and Meyer [11] introduced threshold variants of passively secure encryption and signature schemes in the Hard Homogeneous Spaces (HHS) setting. Cozzo and Smart [8] presented the first actively secure but not robust distributed signature scheme based on isogeny assumptions. In [2], the authors presented CSI-RAShi, a robust and actively secure distributed key generation protocol based on Shamir's secret sharing in the setting of HHS, which necessitates all shareholders to participate.

Outline. In Section 2 the terminology, primitives and security notions relevant for this work are introduced. Section 3 presents an actively secure threshold key exchange mechanism and proves our scheme's active security and simulatability. The actively secure signature scheme resulting from applying the Fiat-Shamir-transform to our key exchange mechanism is discussed in Section 4. Finally, the necessary properties for a secret sharing scheme to be compatible with our key exchange mechanism and signature scheme are characterised in Section 5 in order to enable applying a more general class of secret sharing schemes.

2 Preliminaries

Notation. Throughout this work we use a security parameter $\lambda \in \mathbb{N}$. It is implicitly handed to a protocol whenever needed, that is protocols with computational security. Information theoretic schemes and protocols such as secret sharing schemes used in this work do not require a security parameter.

For an indexed set $X = \{x_i\}_{i \in I}$, we denote the projection onto a subset $I' \subset I$ by $X_{I'} = \{x_i \in X : i \in I'\}$. The same holds for indexed tuples $(x_i)_{i \in I}$.

2.1 Secret Sharing Schemes

A secret sharing scheme is a cryptographic primitive that allows a dealer to share a secret among a set of shareholders. An instance is thus defined by a secret space G, a set of shareholders S and an access structure $\Gamma_{\mathcal{S}}$. A set $S' \in \Gamma_{\mathcal{S}}$ is called *authorised* and can from their respective shares reconstruct a shared secret. If the instance $\mathcal{S}$ is clear from the context, we omit the index in the access structure Γ. In this work, we consider monotone access structures, that is for any $A \subset B \subset S$ with $A \in \Gamma$, we also have $B \in \Gamma$.

A secret sharing instance $\mathcal{S}$ provides two algorithms: Share and Rec. A dealer executes $\mathcal{S}.\mathsf{Share}(s)$ to generate shares $s_1, \ldots, s_k$ of a secret s. A share s_i is assigned to a shareholder $P_{\phi(i)}$ via a surjective map $\phi : \{1, \ldots, k\} \to \{1, \ldots, n\}$ induced by $\Gamma_{\mathcal{S}}$. A set of shareholders $S' \in \Gamma_{\mathcal{S}}$ executes

$$\mathcal{S}.\mathsf{Rec}\Big(\{s_i\}_{P_{\phi(i)} \in S'}\Big)$$

[4] https://csrc.nist.gov/Projects/post-quantum-cryptography/Post-Quantum-Cryptography-Standardization

on their respective shares to retrieve a previously shared secret.

Definition 1 (Superauthorised sets). *For a secret sharing instance $\mathcal{S} = (G, S, \Gamma_{\mathcal{S}})$, we call a set $S' \subset S$* superauthorised*, if for any $P \in S'$, we have $S' \setminus \{P\} \in \Gamma_{\mathcal{S}}$. We denote the set of superauthorised sets of shareholders by $\Gamma_{\mathcal{S}}^{+}$.*

Any superauthorised set is also authorised.

Example 1 (Shamir's secret sharing). An instance of Shamir's famous secret sharing scheme consists of a set of $n > 0$ shareholders, a secret space $\mathbb{Z} \bmod p$, where p is a prime larger than n, and an access structure $\Gamma = \{S' \subset S : \#S' \geq t\}$ for a threshold $t \leq n$. A secret $s \in \mathbb{Z} \bmod p$ is shared by handing each shareholder P_i an interpolation point of a randomly sampled polynomial of degree $t-1$ with constant term s. Reconstruction is achieved via Lagrange interpolation, that is

$$s = \sum_{P_i \in S'} L_{i,S'} s_i = \sum_{P_i \in S'} \prod_{\substack{P_j \in S' \\ j \neq i}} \frac{j}{j-i} f(i)$$

for some $S' \in \Gamma$ and Lagrange interpolation coefficients $L_{i,S'}$. The set of superauthorised sets of shareholders is

$$\Gamma^{+} = \left\{ S^{+} \subset S : \#S^{+} \geq t+1 \right\}.$$

2.2 Hard Homogeneous Spaces

We present our key exchange mechanism and signature scheme in the context of *hard homogeneous spaces* (HHS). HHS were first discussed by Couveignes [6] in 2006. He defines a HHS $(\mathcal{E}, \mathcal{G})$ as a set $\mathcal{E}$ and a group $(\mathcal{G}, \odot)$ equipped with a transitive action $* : \mathcal{G} \times \mathcal{E} \to \mathcal{E}$. This action has the following properties:

– Compatibility: For any $g, g' \in \mathcal{G}$ and any $E \in \mathcal{E}$, we have $g * (g' * E) = (g \odot g') * E$.
– Identity: For any $E \in \mathcal{E}$, $i * E = E$ if and only if $i \in \mathcal{G}$ is the identity element.
– Transitivity: For any $E, E' \in \mathcal{E}$, there exists exactly one $g \in \mathcal{G}$ such that $g * E = E'$.

Definition 2 (Notation). *For a HHS $(\mathcal{E}, \mathcal{G})$ with a fixed $g \in \mathcal{G}$, let $p | \#\mathcal{G}$ be a fixed prime. We denote $[s]\, E := g^{s} * E$ for all $s \in \mathbb{Z}_p$ and all $E \in \mathcal{E}$.*

The following problems are assumed to be efficiently computable in a HHS $(\mathcal{E}, \mathcal{G})$, i.e., there exist polynomial time algorithms to solve them:

– Group operations on $\mathcal{G}$ (membership, inverting elements, evaluating $\odot$).
– Sampling elements of $\mathcal{E}$ and $\mathcal{G}$.
– Testing the membership of $\mathcal{E}$.
– Computing the transitive action $*$: given $g \in \mathcal{G}$ and $E \in \mathcal{E}$ as input, compute $g * E$.

Whereas the subsequent problems are assumed to be hard in a HHS $(\mathcal{E}, \mathcal{G})$.

Problem 1 (Group Action Inverse Problem (GAIP)). Given two elements $E, E' \in \mathcal{E}$ as input, the challenge is to provide $g \in \mathcal{G}$ with $E' = g * E$. Due to the transitivity property of HHS given instance of the GAIP has a solution.

Problem 2 (Parallelisation Problem). An instance of the *Parallelisation Problem* is defined by a triple $(E, E', F) \in \mathcal{E}^3$ with $E' = g * E$. The challenge is to provide F' with $F' = g * F$.

The intuitive decisional continuation of this problem is as follows.

Problem 3 (Decisional Parallelisation Problem). An instance of the *Decisional Parallelisation Problem* is defined by a base element $E \in \mathcal{E}$ and a triple (E_a, E_b, E_c) with $E_a = [a]E$, $E_b = [b]E$ and $E_c = [c]E$. The challenge is to distinguish whether $c = a + b$ or $c \leftarrow_{\$} \mathbb{Z}_p$ was randomly sampled.

Remark 1. It is obvious that the decisional parallelisation problem reduces to the parallelisation problem, which reduces to the group action inverse problem.

2.3 Threshold Group Action

Let s be a Shamir shared secret among shareholders $P_1, \ldots, P_n$, that is each P_i holds a share s_i of s, $i = 1, \ldots, n$. To compute $E' = [s]\,E$ for an arbitrary but fixed $E \in \mathcal{E}$ without reconstructing s, we have an authorised set of shareholders execute Algorithm 5. If it is executed successfully, we have by the compatibility property of $*$ and the repeated application of $E^k \leftarrow [L_{i,S'} s_i]\,E^{k-1}$ the result

$$E^{\#S'} = \left[\sum_{P_i \in S'} L_{i,S'} s_i\right] E = [s]\,E.$$

2.4 Piecewise Verifiable Proofs

A piecewise verifiable proof (PVP) is a cryptographic primitive in the context of hard homogeneous spaces and was first introduced in [2]. It is a compact non-interactive zero-knowledge proof of knowledge of a witness $f \in \mathbb{Z}_q[X]$ for a statement

$$x = ((E_0, E_1), s_1, \ldots, s_n), \tag{2.1}$$

with statement pieces $s_i = f(i)$ for $i = 0, \ldots, n$ and $E_1 = [s_0]\,E_0 \in \mathcal{E}$. A PVP provides a proving protocol $\mathsf{PVP}.P$, which takes a statement x of the form (2.1) and a witness f and outputs a proof $\left(\pi, \{\pi_i\}_{i=0,\ldots,n}\right)$, where (π, π_i) is a proof piece for s_i, $i = 0, \ldots, n$. The PVP also provides a verifying protocol $\mathsf{PVP}.V$, which takes an index $i \in \{0, \ldots, n\}$, a statement piece s_i and a proof piece (π, π_i) and outputs true or false. Let $\mathcal{R} = \{(x, f)\}$, where f is a witness for the statement x. The projection R_I for some $I \subset \{0, \ldots, n\}$ denotes (x_I, f).

Definition 3 (Completeness). *We call a PVP* complete*, if, for any $(x, f) \in \mathcal{R}$ and*

$$\left(\pi, \{\pi_i\}_{i=0,\dots,n}\right) \leftarrow \mathsf{PVP}.P\,(f, x)\,,$$

the verification succeeds, that is

$$\forall j \in \{0, \dots, n\} : \Pr\left[\mathsf{PVP}.V\,(j, x_j, (\pi, \pi_j)) = \mathsf{true}\right] = 1.$$

Definition 4 (Soundness). *A PVP is called* sound *if, for any adversary $\mathcal{A}$, any $I \subset \{0, \dots, n\}$ and any x for which there exists no f with $(x_I, f) \in \mathcal{R}_I$,*

$$\Pr\left[\mathsf{PVP}.V(j, x_j, (\pi, \pi_j)) = \mathsf{true}\right]$$

is negligible in the security parameter λ for all $j \in I$, where $\left(\pi, \{\pi_i\}_{i\in I}\right) \leftarrow \mathcal{A}(1^\lambda)$.

Definition 5 (Zero-knowledge). *A PVP is* zero-knowledge*, if for any $I \subset \{1, \dots, n\}$ and any $(x, f) \in \mathcal{R}$, there exists a simulator* Sim *such that for any polynomial-time distinguisher $\mathcal{A}$ the advantage*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{Sim}(x_I)}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{P(x,f)}(1^\lambda) = 1\right]\right|$$

is negligible in the security parameter λ, where P is an oracle that upon input (x, f) returns $\left(\pi, \{\pi_j\}_{j\in I}\right)$ with $\left(\pi, \{\pi_j\}_{j=0,\dots,n}\right) \leftarrow \mathsf{PVP}.P(f, x)$.

We refer to [2] for the precise proving and verifying protocols and the security thereof. In combination they state a complete, sound and zero-knowledge non-interactive PVP. A prover can hence show knowledge of a sharing polynomial f to a secret $s_0 = f(0)$ with shares $s_i = f(i)$. In Section 3, we adjust [2]'s proving protocol to our setting of threshold schemes, so that knowledge of a subset of interpolation points is proven instead of all interpolation points.

2.5 Zero-Knowledge Proofs for the GAIP

We give a non-interactive zero-knowledge proof protocol for an element $s \in \mathbb{Z}_p$ with respect to the group action inverse problem. That is, a prover shows the knowledge of s so that $E_i' = [s]\,E_i$, for $E_i, E_i' \in \mathcal{E}$ and $i = 1, \dots, m$, simultaneously, without revealing s.

The prover samples $b_j \in \mathbb{Z}_p$ and computes $\hat{E}_{i,j} \leftarrow [b_j]\,E_i$ for $i = 1, \dots, m$ and $j = 1, \dots, \lambda$. He then derives challenge bits

$$(c_1, \dots, c_\lambda) \leftarrow \mathcal{H}\left(E_1, E_1', \dots, E_m, E_m', \hat{E}_{1,1} \dots, \hat{E}_{m,\lambda}\right)$$

via a hash function $\mathcal{H} : \mathcal{E}^{(2+\lambda)m} \to \{0,1\}^\lambda$ and prepares the answers $r_j \leftarrow b_j - c_j s$, $j = 1, \dots, \lambda$. The proof $\pi = (c_1, \dots, c_\lambda, r_1, \dots, r_\lambda)$ is then published.

The verification protocol is straight forward: given a statement $(E_i, E_i')_{i=1,\dots,m}$ and a proof $\pi = (c_1, \dots, c_\lambda, r_1, \dots, r_\lambda)$, the verifier computes $\tilde{E}_{i,j} \leftarrow [r_j]\,E_i$ if

$c_j = 0$ and $\tilde{E}_{i,j} \leftarrow [r_j]\, E_i'$ otherwise, for $i = 1, \ldots, m$ and $j = 1, \ldots, \lambda$. He then generates verification bits $(\tilde{c}_1, \ldots \tilde{c}_\lambda) \leftarrow \mathcal{H}\Big(E_1, E_1', \ldots, E_m, E_m', \tilde{E}_{1,1} \ldots, \tilde{E}_{m,\lambda}\Big)$ and accepts the proof if $(c_1, \ldots, c_\lambda) = (\tilde{c}_1, \ldots, \tilde{c}_\lambda)$.

We sketch the proving and verifying protocols in Algorithm 6 and Algorithm 7, respectively. Again, we refer to [3] for the proof of completeness, soundness and zero-knowledge with respect to the security parameter λ.

2.6 The Adversary

We consider a static and active adversary. At the beginning of a protocol execution, the adversary corrupts a set of shareholders. The adversary is able to see their inputs and control their outputs. The set of corrupted shareholders cannot be changed throughout the execution of the protocol.

The adversary's aim is two-fold. On the one hand it wants to obtain information on the uncorrupted parties' inputs, on the other hand it wants to manipulate the execution of our protocol towards an incorrect output without detection.

2.7 Communication channels

Both our schemes assume the existence of a trusted dealer in the secret sharing instance. The shareholders' communication occurs in the execution of the decapsulation protocol of our key exchange mechanism and the signing protocol of our signature scheme.

The communication from the dealer to a shareholder must not be eavesdropped upon or tampered with, we hence assume secure private channels between the dealer and each shareholder. However, the communication between shareholders need not be kept private, thus we assume a simple broadcast channel between the shareholders. The means of how to establish secure private channels and immutable broadcast channels are out of scope of this work.

3 Key Exchange Mechanism

A key exchange mechanism is a cryptographic public key scheme that provides three protocols: KeyGen, Encaps and Decaps. These enable a party to establish an ephemeral key between the holder of the secret key. We present our actively secure key exchange mechanism with private key that is secret shared among a set of shareholders. An authorised subset can execute the Decaps protocol without reconstructing the secret key.

3.1 Public Parameters

We fix the following publically known parameters.

– A secret sharing instance $\mathcal{S}$ with shareholders $S = \{P_1, \ldots, P_n\}$, secret space $\mathbb{Z}_p$ and access structure Γ.

– A hard homogeneous space $(\mathcal{E}, \mathcal{G})$ with a fixed starting point $E_0 \in \mathcal{E}$.
– A fixed element $g \in \mathcal{G}$ with $\mathsf{ord} g = p$ for the mapping $[\cdot]\cdot : \mathbb{Z}_p \times \mathcal{E} \to \mathcal{E}; s \mapsto g^s E$.

We give our key exchange mechanism in the context of Shamir's secret sharing scheme and elaborate possible extensions to other, more general secret sharing schemes in Section 5.

3.2 Key Generation

A public and secret key pair is established by a trusted dealer (even an untrusted dealer is feasible by employing verifiable secret sharing schemes) executing Algorithm 1. For that he samples a secret key s and publishes the public key $\mathsf{pk} \leftarrow [s]E_0$. The secret key s is then shared among the shareholders $\{P_1, \ldots, P_n\}$ via $\mathcal{S}.\mathsf{Share}(s)$. The dealer shares each share s_i, $i = 1, \ldots, n$, once more with a sharing polynomial f_i. Each shareholder P_i, $i = 1, \ldots, n$, eventually receives s_i, f_i and $f_j(i)$, that is his share s_i of s, the polynomial f_i and a share $f_j(i)$ of each other s_j, $j \neq i$.

Algorithm 1: Key generation

```
Input: $\mathcal{S}$
$s \leftarrow_\$ \mathbb{Z}_p$
$\mathsf{pk} \leftarrow [s] E_0$
$\{s_1, \ldots, s_n\} \leftarrow \mathcal{S}.\mathsf{Share}(s)$
for $i = 1, \ldots, n$ do
    $f_i \leftarrow_\$ \mathbb{Z}_p [X]_{\leq k-1} : f_i(0) = s_i$
publish $\mathsf{pk}$
for $i = 1, \ldots, n$ do
    send $\left\{ s_i, f_i, \{f_j(i)\}_{j=1,\ldots,n} \right\}$ to $P_i$
```

This key generation protocol can be regarded as a "two-level sharing", where each share of the secret key is itself shared again among the shareholders. While this is not necessary for De Feo and Meyer's passively secure protocol, we require the two-level sharing in ensuring the active security of our key encapsulation mechanism.

3.3 Encapsulation

With a public key $\mathsf{pk} \in \mathcal{E}$ as input, the encapsulation protocol returns an ephemeral key $\mathcal{K} \in \mathcal{E}$ and a ciphertext $c \in \mathcal{E}$. Our encapsulation protocol is identical to the protocol of [11], thus we just give a short sketch and refer to De Feo's and Meyer's work for the respective proofs of security.

Algorithm 2: Encapsulation

```
Input: $\mathsf{pk}$
$b \leftarrow_{\$} \mathcal{G}$
$\mathcal{K} \leftarrow b * \mathsf{pk}$
$c \leftarrow b * E_0$
return $(\mathcal{K}, c)$
```

3.4 Decapsulation

A decapsulation protocol takes a ciphertext c and outputs a key $\mathcal{K}$. De Feo and Meyer [11] applied the threshold group action (Algorithm 5) so that an authorised set $S' \in \Gamma$ decapsulates a ciphertext c and produces an ephemeral key $[s]\, c = [s]\,(b * E_0) = b * ([s]\, E_0)$. For that, the shareholders agree on an arbitrary order of turns. With $E^0 := c$, the k^{th} shareholder P_i outputs $E^k = [L_{i,S'} s_i]\, E^{k-1}$ for $k = 1, \ldots, \#S'$. The last shareholder outputs the decapsulated ciphertext $E^{\#S'} = [s]\, c$. Their approach is simulatable. It does not leak any information on the shares s_i, yet it is only passively secure. Thus, a malicious shareholder can provide malformed input to the protocol and thereby manipulate the output of the computation towards incorrect results without the other parties recognising this deviation from the protocol. We extend their approach to enable the detection of misbehaving shareholders in a decapsulation. For that we maintain the threshold group action and apply the PVP and zero-knowledge proof for the group action inverse problem as layed out in Section 2.

3.5 Amending the PVP

In the PVP protocol sketched in Section 2, a prover produces a proof of knowledge for a witness polynomial f of the statement $((E_0, E_1), s_1, \ldots, s_n)$, where $E_0 \leftarrow_{\$} \mathcal{E}$, $E_1 = [s_0]\, E_0$ and $s_i = f(i)$ for $i = 0, \ldots, n$. He thereby proves knowledge of the sharing polynomial f of $s_0 = f(0)$.

This approach does not agree with the threshold group action, for which a shareholder P_i's output in the round-robin approach is $E^k \leftarrow [L_{i,S'} s_i]\, E^{k-1}$ rather than $E^k \leftarrow [s_i]\, E^{k-1}$, where E^{k-1} denotes the previous shareholder's output. Futhermore, authorised sets need not contain all shareholders. Example 2 illustrates a further conflict with of the PVP with the threshold group action.

Example 2. Let sk be a secret key generated and shared by KeyGen. That is each shareholder P_i holds

$$\left\{ s_i, f_i, \{ f_j(i) \}_{P_j \in S} \right\}.$$

Also let $S' \in \Gamma$ be a minimally authorised set executing Algorithm 5, i.e., for any $P_i \in S'$, $S' \backslash \{P_i\}$ is unauthorised. Thus, for any arbitrary but fixed $s_i' \in \mathbb{Z}_p$, there exists a polynomial $f_i' \in \mathbb{Z}_p[X]_{k-1}$ so that $f_i'(j) = L_{i,S'} f_i(j)$ and $R' = [f_i'(0)]\, R$

for any $R, R' \in \mathcal{E}$ and all $P_j \in S' \setminus \{P_i\}$. Therefore, P_i can publish

$$\left(\pi, \{\pi_j\}_{P_j \in S}\right) \leftarrow \mathsf{PVP}.P\left(\left((R, R'), (L_{i,S'} f_i(j))_{P_j \in S}\right), f_i'\right)$$

which is indistinguishable from

$$\mathsf{PVP}.P\left(\left((E_0, E_1), (L_{i,S'} f_i(j))_{P_j \in S}\right), L_{i,S'} f_i\right)$$

to $S' \setminus \{P_i\}$ with $E_0 \leftarrow\!\!\$\, \mathcal{E}$ and $E_1 = [L_{i,S'} s_i] E_0$. Thus, for a minimally authorised set S', the soundness of the PVP does not hold with respect to $P_i \in S'$ and f_i.

We resolve the conflicts by amending [2]'s PVP protocol, so that a shareholder $P_i \in S^*$ proves knowledge of a witness polynomial $L_{i,S^*} f_i$ for a statement

$$\left((R, R'), (f_i(j))_{P_j \in S^*}\right),$$

to a superauthorised set S^*, where $R \leftarrow\!\!\$\, \mathcal{E}$, $R' = [L_{i,S^*} f_i(0)] R = [L_{i,S^*} s_i] R$. The inputs of our amended proving protocol are the proving shareholder's index i, the witness polynomial f_i, the superauthorised set $S^* \in \Gamma^+$ and the statement $\left((R, R'), (f_i(j))_{P_j \in S^*}\right)$. The protocol can be found in Algorithm 8, in which $\mathcal{C}$ denotes a commitment scheme. The verifying protocol in turn has the prover's and the verifier's indices i and j, respectively, a set $S^* \in \Gamma^+$, a statement piece x_j and a proof piece (π, π_j) as input, where $x_j = (R, R') \in \mathcal{E}^2$ if $j = 0$ and $x_j \in \mathbb{Z}_p$ otherwise. The verifying protocol is given in Algorithm 9.

It is here, that the two-level sharing we introduced in Section 3.2 comes into play. We will have each shareholder P_i engaged in an execution of Decaps provide a PVP with respect to its share s_i of the secret key sk, that is then verified by each other participating shareholder with its respective share of s_i.

The definitions of soundness and zero-knowledge for a threshold PVP scheme carry over from the non-threshold setting in Section 2 intuitively, yet we restate the completeness definition for the threshold setting.

Definition 6 (Completeness in the threshold setting). *We call a threshold PVP scheme* complete *if, for any $S' \in \Gamma$, any $(x, f) \in \mathcal{R}$, any $P_i \in S'$ and $\left(\pi, \{\pi_j\}_{P_j \in S'}\right) \leftarrow \mathsf{PVP}.P(i, f, S', x_{S'})$, we have*

$$\Pr\left[\mathsf{PVP}.V(i, j, S', x_j, (\pi, \pi_j)) = \mathsf{true}\right] = 1 \textit{ for all } P_j \in S'.$$

The proofs for soundness, correctness and zero-knowledge for Beullens et al.'s [2] approach are easily transferred to our amended protocols, thus we do not restate them here.

We arrive at our decapsulation protocol, executed by a superauthorised set S^*: The partaking shareholders fix a turn order. A shareholder P_i's turn consists of the following steps.

1. If the previous shareholder's output E^{k-1} is not in $\mathcal{E}$, P_i outputs $\perp$ and aborts. The first shareholder's input E^0 is the protocol's input ciphertext c.

2. Otherwise P_i samples $R_k \leftarrow_{\$} \mathcal{E}$ and computes $R'_k \leftarrow [L_{i,S^*} s_i] R_k$.
3. P_i computes and publishes
$$\left(\pi^k, \{\pi_j^k\}_{P_j \in S^*}\right) \leftarrow \mathsf{PVP}.P\Big(i, f_i, S^*, \Big((R_k, R'_k), (f_i(j))_{P_j \in S^*}\Big)\Big).$$
4. P_i computes $E^k \leftarrow [L_{i,S^*} s_i] E^{k-1}$ and the zero-knowledge proof
$$zk \leftarrow \mathsf{ZK}.P\big((R_k, R'_k), (E^{k-1}, E^k), L_{i,S^*} s_i\big).$$
He publishes both.
5. Each shareholder $P_j \in S^* \setminus \{P_i\}$ verifies
$$\mathsf{PVP}.V\big(i, j, S^*, f_i(j), (\pi^k, \pi_j^k)\big) \wedge \mathsf{PVP}.V\big(i, 0, S^*, (R_k, R'_k), (\pi^k, \pi_0^k)\big) \quad (3.1)$$
and
$$\mathsf{ZK}.V\big((R_k, R'_k), (E^{k-1}, E^k), zk\big). \quad (3.2)$$
If (3.1) fails, P_j issues a complaint against P_i. If P_i is convicted of cheating by more than $\#S^*/2$ shareholders, decapsulation is restarted with an $S^{*\prime} \in \Gamma^+$, so that $P_i \notin S^{*\prime}$. If (3.2) fails, the decapsulation is restarted outright with $S^{*\prime} \in \Gamma^+$, so that $P_i \notin S^{*\prime}$.
6. Otherwise, P_i outputs E^k and finalises its turn.
7. The protocol terminates with the last shareholder's $E^{\#S^*}$ as output.

The combination of the PVP and the zero-knowledge proof in steps 3 and 4 ensure, that P_i has knowledge of the sharing polynomial $L_{i,S^*} f_i$ and also inputs $L_{i,S^*} f_i(0)$ to compute E^k. We give the precise protocol in Algorithm 3.

Definition 7. *A key exchange mechanism with secret shared private key is* correct, *if for any authorised set S', any public key* pk *and any* $(\mathcal{K}, c) \leftarrow \mathsf{Encaps}(\mathsf{pk})$, *we have* $\mathcal{K} = \mathcal{K}' \leftarrow \mathsf{Decaps}(c, S')$.

The correctness of our key exchange mechanism presented in Algorithm 1, Algorithm 2 and Algorithm 3 follows from the correctness of the threshold group action (Algorithm 5). Let sk be a secret key and $\mathsf{pk} = [\mathsf{sk}] E_0$ be the respective public key, that have been generated by KeyGen, thus each shareholder P_i holds a share s_i of sk, $i = 1, \ldots, n$. For an authorised set S' we therefore have
$$\mathsf{sk} = \sum_{P_i \in S'} L_{i,S'} s_i.$$

Furthermore, let $(\mathcal{K}, c) \leftarrow \mathsf{Encaps}(\mathsf{pk})$. To show correctness, $\mathcal{K}' = \mathcal{K}$ has to hold, where $\mathcal{K}' \leftarrow \mathsf{Decaps}(c, S')$. Now, after executing $\mathsf{Decaps}(c, S')$, we have $\mathcal{K}' = E^{\#S'}$ emerging as the result of the threshold group action applied to c. This gives us
$$\mathcal{K}' = \left[\sum_{P_i \in S'} L_{i,S'} s_i\right] c = [\mathsf{sk}] (b * E_0) = b * \mathsf{pk} = \mathcal{K}.$$

Algorithm 3: Decapsulation

```
Input: $c, S^*$
$E^0 \leftarrow c$
$k \leftarrow 0$
for $P_i \in S^*$ do
    if $E^k \notin \mathcal{E}$ then
        $P_i$ outputs $\perp$ and aborts.
    $k \leftarrow k+1$
    $R_k \leftarrow_\$ \mathcal{E}$
    $R'_k \leftarrow [L_{i,S^*} s_i] R_k$
    $\left(\pi^k, \{\pi_j^k\}_{P_j \in S^*}\right) \leftarrow \mathsf{PVP}.P\left(i, f_i, S^*, ((R_k, R'_k), (f_i(j))_{P_j \in S^*})\right)$
    $P_i$ publishes $(R_k, R'_k)$ and $\left(\pi^k, \{\pi_j^k\}_{P_j \in S^*}\right)$
    $E^k \leftarrow [L_{i,S^*} s_i] E^{k-1}$
    $zk^k \leftarrow \mathsf{ZK}.P\big((R_k, R'_k), (E^{k-1}, E^k), L_{i,S^*} s_i\big)$
    $P_i$ publishes $(E^k, zk^k)$
    for $P_j \in S^* \setminus \{P_i\}$ do
        if $\mathsf{ZK}.V\big((R_k, R'_k), (E^{k-1}, E^k), zk^k\big) = \mathsf{false}$ then
            return $\mathsf{Decapsulation}(c, S^{*\prime})$ with $S^{*\prime} \in \Gamma \wedge P_i \notin S^{*\prime}$
        if $\mathsf{PVP}.V\big(i, j, S^*, f_i(j), (\pi^k, \pi_j^k)\big) = \mathsf{false} \vee$
         $\mathsf{PVP}.V\big(i, 0, S^*, (R_k, R'_k), (\pi^k, \pi_0^k)\big) = \mathsf{false}$ then
            $P_j$ publishes $f_i(j)$
            if $P_i$ is convicted then
                return $\mathsf{Decapsulation}(c, S^{*\prime})$ with $S^{*\prime} \in \Gamma \wedge P_i \notin S^{*\prime}$
return $\mathcal{K} \leftarrow E^k$
```

The decapsulation is executed by superauthorised sets $S^* \in \mathit{\Gamma}^+ \subset \mathit{\Gamma}$. This shows that our key exchange mechanism is correct.

3.6 Security

There are two aspects of security to consider:

– Active security: A malicious shareholder cannot generate his contribution to the decapsulation protocol dishonestly without being detected. We prove this by showing that an adversary that can provide malformed inputs without detection can break the PVP or the zero-knowledge proof of knowledge.

– Simulatability: An adversary that corrupts an unauthorised set of shareholders cannot learn any information about the uncorrupted shareholders' inputs from an execution of the decapsulation protocol. We show this by proving the simulatability of Decaps.

Active security

Theorem 1. *Let $S^* \in \Gamma^+$ and let* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$ *be a public/secret key pair, where* sk *has been shared. Also let* $(\mathcal{K}, c) \leftarrow \mathsf{Encaps}(\mathsf{pk})$. *Denote the transcript of* $\mathsf{Decaps}(c, S^*)$ *by*

$$\left(E^k, (R_k, R'_k), \left(\pi^k, \{\pi_j^k\}_{P_j \in S^*}\right), zk^k\right)_{k=1,\dots,\#S^*}.$$

Let $P_i \in S^*$ be an arbitrary but fixed shareholder. If $\mathsf{Decaps}(c, S^*)$ *terminated successfully and $P_{i'}$'s output was generated dishonestly, then there exists an algorithm that breaks the soundness property of* PVP *or* ZK.

Proof. Let $P_{i'}$ be the malicious shareholder and let k' be the index of $P_{i'}$'s output in the transcript. Since $\mathsf{Decaps}(c, S^*)$ terminated successfully, we have

$$\mathsf{PVP}.V\left(i', j, S^*, f_{i'}(j), \left(\pi^{k'}, \pi_j^{k'}\right)\right) = \mathsf{true} \tag{3.3}$$

$$\mathsf{PVP}.V\left(i', 0, S^*, (R_{k'}, R'_{k'}), \left(\pi^{k'}, \pi_0^{k'}\right)\right) = \mathsf{true} \tag{3.4}$$

$$\mathsf{ZK}.V\left(\left(E^{k'-1}, E^{k'}\right), (R_{k'}, R'_{k'}), zk^{k'}\right) = \mathsf{true} \tag{3.5}$$

for all $P_j \in S^* \setminus \{P_{i'}\}$. $E^{k'}$ was generated dishonestly, thus we have

$$E^{k'} = [\alpha]\, E^{k'-1}, \text{ for some } \alpha \neq L_{i',S^*} s_{i'}.$$

We distinguish two cases: $R'_{k'} \neq [\alpha]\, R_{k'}$ and $R'_{k'} = [\alpha]\, R_{k'}$.

In the first case, $P_{i'}$ published a zero-knowledge proof $zk^{k'}$ so that (3.5) holds, where $E^{k'} = [\alpha]\, E^{k'-1}$ yet $R'_{k'} \neq [\alpha]\, R_{k'}$. $P_{i'}$ thus broke the soundness property of the zero-knowledge proof.

In the second case, $P_{i'}$ published $\left(\pi^{k'}, \left\{\pi_j^{k'}\right\}_{P_j \in S^*}\right)$ so that (3.3) and (3.4) hold for all $P_j \in S^* \setminus \{P_{i'}\}$ and for $j = 0$. Thus, $P_{i'}$ proved knowledge of a witness polynomial f' with

$$f'(j) = L_{i',S^*} f_{i'}(j) \tag{3.6}$$

for all $P_j \in S^* \setminus \{P_{i'}\}$ and $R'_{k'} = [f'(0)]\, R_{k'}$, that is $f'(0) = \alpha$. Since f' has degree at most $k-1$, it is well-defined from (3.6). Thus, we have $f' \equiv L_{i',S^*} f_{i'}$, where $f_{i'}$ is the polynomial with which $s_{i'}$ was shared, i.e., $f_{i'}(0) = s_{i'}$. This gives us $\alpha = f'(0) = L_{i',S^*} f_{i'}(0) = L_{i',S^*} s_{i'}$. We arrive at a contradiction, assuming the soundness of the PVP.

Simulatability We show that an adversary who corrupts an unauthorised subset of shareholder does not learn any additional information from an execution of the decapsulation protocol.

Definition 8 (Simulatability). *We call a key exchange mechanism* simulatable*, if for any HHS $(\mathcal{E}, \mathcal{G})$ with security parameter λ and any compatible secret sharing instance $\mathcal{S}$, there exists a polynomial-time algorithm* Sim *so that for any polynomial-time adversary $\mathcal{A}$ the advantage*

$$\mathsf{Adv}^{\text{dist-transcript}}_{\mathcal{A},\mathsf{Sim}}((\mathcal{E},\mathcal{G}),\mathcal{S}) := \left|\Pr\left[\mathsf{Exp}^{\textit{dist-transcript}}_{\mathcal{A},\mathsf{Sim}}(\mathcal{S})\right] - \frac{1}{2}\right|$$

in the security game $Exp^{\textit{dist-transcript}}_{\mathcal{A},\mathsf{Sim}}(\mathcal{S})$ (Algorithm 10) is negligible in λ.

Theorem 2. *If the* PVP *protocol and the GAIP* ZK *protocol employed are zero-knowledge, then the decapsulation protocol (Algorithm 3) is simulatable.*

Proof. We give a finite series of simulators, the first of which simulates the behaviour of the uncorrupted parties faithfully and the last of which fulfills the secrecy requirements. This series is inspired by the simulators, that [2] gave for the secrecy proof of their key generation algorithm, yet differs in some significant aspects. The outputs of the respective simulators will be proven indistinguishable, hence resulting in the indistinguishability of the first and last one. As a slight misuse of the notation, we denote the set of corrupted shareholders by $\mathcal{A}$, where $\mathcal{A}$ is the adversary corrupting an unauthorised set of shareholders. This means P_i is corrupted iff $P_i \in \mathcal{A}$.

The input for each simulator is a ciphertext c, a derived key $\mathcal{K}$ and the adversary's knowledge after KeyGen was successfully executed, that is

$$\left\{s_i, f_i, \{f_j(i)\}_{P_j \in S^* \setminus \mathcal{A}}\right\}_{P_i \in \mathcal{A}}.$$

1. The adversary corrupted an unauthorised set $\mathcal{A}$, hence each share of the secret key is uniformly distributed from his view. Sim^1 samples a polynomial $f'_i \in \mathbb{Z}_p[X]_{k-1}$ with
$$\forall P_j \in \mathcal{A} : f'_i(j) = f_i(j)$$
uniformly at random for each $P_i \in S^* \setminus \mathcal{A}$. Since $\mathcal{A}$ is unauthorised, f'_i exists. Sim^1 then proceeds by honestly producing the output of each $P_i \in S^* \setminus \mathcal{A}$ according to the decapsulation protocol, i.e., it samples $R_k \leftarrow_\$ \mathcal{E}$, computes $R'_k \leftarrow [L_{i,S^*} f'_i(0)]\, R_k$ and outputs
$$\mathsf{PVP}.P\Big(i, f'_i, S^*, \big((R_k, R'_k), (f'_i(j))_{P_j \in S^*}\big)\Big),$$
$$E^k \leftarrow [L_{i,S^*} f'_i(0)]\, E^{k-1}$$
and
$$\mathsf{ZK}.P\big((R_k, R'_k), (E^{k-1}, E^k), L_{i,S^*} f'_i(0)\big),$$
where k is the index of P_i's output in the transcript. Since, for all $P_i \in S^* \setminus \mathcal{A}$, the real share $s_i = f_i(0)$ of P_i is information theoretically hidden to the adversary, the resulting transcript is identically distributed to a real transcript.

2. Let i' denote the index of the last honest party in the execution of the decapsulation protocol and k' the index of its output. Sim^2 behaves exactly as Sim^1 with the exception, that it does not compute the PVP itself but calls the simulator $\mathsf{Sim}^{\mathsf{PVP}}$ for the PVP to generate the proof $\left(\pi^{k'}, \left\{\pi_j^{k'}\right\}\right)$ for the statement $\left(\left(R_{k'}, R'_{k'}\right), \left(f_{i'}(j)\right)_{P_j \in S^*}\right)$. Since the PVP is zero-knowledge, Sim^2's output is indistinguishable from that of Sim^1.

3. Sim^3 behaves identical to Sim^2 apart from not generating the zero-knowledge proof for $P_{i'}$ itself, but outsourcing it to the simulator for the zero-knowledge proof. That is Sim^3 hands tuples $(R_{k'}, R'_{k'})$ and $\left(E^{k'-1}, E^{k'}\right)$ to $\mathsf{Sim}^{\mathsf{ZK}}$ and publishes its answer as the zero-knowledge proof. With ZK being zero-knowledge, the output of Sim^3 is indistinguishable from that of Sim^2.

4. The final simulator Sim^4 enforces the correct decapsulation output, that is $E^{\#S^*} = \mathcal{K}$. Since, for $P_j \in \mathcal{A}$, s_j was provided as input and $P_{i'}$ is the last honest shareholder in the order of decapsulation execution, Sim^4 computes
$$\sum_{P_j \in S'} L_{j,S^*} s_j,$$
where S' contains the shareholders, whose turn is after $P_{i'}$'s. To achieve the correct output of the decapsulation E, Sim^4 thus sets
$$E^{k'} \leftarrow \left[- \sum_{P_j \in S'} L_{j,S^*} s_j\right] E$$
instead of $E^{k'} \leftarrow \left[L_{i',S^*} s'_{i'}\right] E^{k'-1}$. Assuming the soundness of the PVP as well as of the zero-knowledge proof, this guarantees the result to be $E^{\#S^*} = E$, since
$$E^{\#S^*} = \left[\sum_{P_j \in S'} L_{j,S^*} s_j\right] E^{k'} = E$$
holds. It remains to show, that the output of Sim^4 cannot be distinguished from that of Sim^3. The following reasoning is similar to that of [2], yet for completeness we give a reduction $\mathcal{B}'$, that uses a distinguisher $\mathcal{A}'$, that distinguishes Sim^3 from Sim^4, to break the decisional parallelisation problem. We highlight the necessary modifications.

Let (E_a, E_b, E_c) be an instance of the decisional parallelisation problem with base element c. $\mathcal{B}'$ computes
$$E^{k'} \leftarrow \left[\sum_{P_j \in S^* \setminus (S' \cup \{P_{i'}\})} L_{j,S^*} s_j\right] E_a.$$

With $s_{i'}$ looking uniformly distributed from $\mathcal{A}$'s view, this choice of $E^{k'}$ is indistinguishable from $E^{k'} = [L_{i',S^*} s'_{i'}]\, E^{k'-1}$. $\mathcal{B}'$ furthermore does not sample $R_{k'} \leftarrow\!\!\$\, \mathcal{E}$ but puts $R_{k'} \leftarrow E_b$ and $R'_{k'} \leftarrow E_c$. The resulting transcript is handed to $\mathcal{A}'$ and $\mathcal{B}'$ outputs whatever $\mathcal{A}'$ outputs.
Comparing the distributions, we see that

$$E^{k'} = [a]\, E^{k'-1} = [a] \left(\left[\sum_{P_j \in S^* \setminus (S' \cup \{P_{i'}\})} L_{j,S^*} s_j \right] c \right)$$

if and only if $E_a = [a]c$, where $s_j := s'_j$ for $P_j \notin \mathcal{A}$. Furthermore, $R'_{k'} = [a]R_{k'}$ is equivalent to $E_c = [a]E_b$. In the case of $E_a = [a]c$ and $E_c = [a]E_b$, the transcript handed to $\mathcal{A}'$ is identically distributed to Sim^3's output. If, on the other hand, (E_a, E_b, E_c) is a random triple, then the transcript follows the same distribution as Sim^4's output. $\mathcal{B}'$ thus breaks the DPP with the same advantage as $\mathcal{A}'$ distinguishes Sim^3 from Sim^4.

Sim^4 outputs a transcript of the decapsulation protocol with input c and output $\mathcal{K}$ that cannot be distinguished from the output of Sim^1, which is indistinguishable from a real execution protocol.

3.7 Efficiency

Each shareholder engaged in an execution of the decapsulation protocol has one round of messages to send. The messages of the k-th shareholder consist of the tuple (R_k, R'_k), a PVP proof $\left(\pi^k, \{\pi^k_j\}_{P_j \in S^*}\right)$, the output E^k and the zero-knowledge proof zk. Thus, the total size of a shareholder's messages is

$$\begin{aligned} &2x + 2c + \lambda k \log p + 2\lambda(\#S^*) + x + \lambda k \log p + \lambda \\ =&3x + 2c + \lambda\,(1 + 2(\#S^*) + 2k \log p) \end{aligned}$$

where x is the size of the bit representation of an element of $\mathcal{E}$ and c is the size of a commitment produced in $\mathsf{PVP}.P$. Assuming x, c and the secret sharing parameters k and p to be constant, the message size is thus linear in the security parameter λ with moderate cofactor.

4 Actively Secure Secret Shared Signature Protocols

We convert the key exchange mechanism in Algorithm 1, Algorithm 2 and Algorithm 3 into an actively secure signature scheme with secret shared signing key.

A signature scheme consists of three protocols: key generation, signing and verifying. We transfer the unmodified key generation protocol from the key exchange mechnism in Section 3 to our signature scheme. The signing protocol is derived from the decapsulation protocol (Algorithm 3) by applying the Fiat-Shamir-transformation, the verifying protocol follows straightforward. The protocols are given in Algorithm 4 and Algorithm 11.

We concede, that applying active security measures to a signature scheme to ensure the correctness of the resulting signature is counter-intuitive, since the correctness of a signature can easily be checked through the verifying protocol. Yet verification returning false only shows that the signature is incorrect, a misbehaving shareholder cannot be identified this way. An actively secure signature scheme achieves just that. An identified cheating shareholder can hence be excluded from future runs of the signing protocol.

Similar to [3], the results from [10] on Fiat-Shamir in the QROM can be applied to our setting as follows. First, in the case without hashing, since the sigma protocol has special soundness [3] and in our case perfect unique reponses, [10] shows that the protocol is a quantum proof of knowledge. Further, in the case with hashing, the collapsingness property implies that the protocol has unique responses in a quantum scenario.

4.1 Instantiations

As a practical instantiation, we propose the available parameter set for CSIDH-512 HHS from [3]. Currently no other instantiation of the presented schemes seems feasible in a practical sense. Furthermore, according to recent works [14,5] CSIDH-512 may not reach the initially estimated security level.

5 Generalising the Secret Sharing Schemes

We constructed the protocols above in the context of Shamir's secret sharing protocol [15]. The key exchange mechanism in Section 3 as well as the signature scheme in Section 4 can be extended to more general secret sharing schemes. In the following, we characterise the requirements that a secret sharing scheme has to meet in order to successfully implement the key exchange mechanism and the signature scheme.

5.1 Compatibility Requirements

Definition 9 (Independent Reconstruction). *We say a secret sharing instance $\mathcal{S} = (S, \Gamma, G)$ is* independently reconstructible*, if, for any shared secret $s \in G$, any $S' \in \Gamma$ and any shareholder $P_i \in S'$, P_i's input to reconstructing s is independent of the share of each other engaged shareholder $P_j \in S'$.*

A secret sharing scheme compatible with our key exchange mechanism and signature scheme has to be independently reconstructible, since each shareholder's input into the threshold group action is hidden from every other party by virtue of the GAIP.

Algorithm 4: Secret Shared Signing Algorithm

```
Input: $m, S^*$
$\left(E_1^0, \ldots, E_\lambda^0\right) \leftarrow (E_0, \ldots, E_0)$
$k \leftarrow 0$
for $P_i \in S^*$ do
    $k \leftarrow k+1$
    for $l \in 1, \ldots, \lambda$ do
        $P_i$ samples $b_{il} \leftarrow_{\$} \mathbb{Z}_q[X]_{\leq k-1}$
        $P_i$ publishes $R_{il}^k \leftarrow_{\$} \mathcal{E}$
        $P_i$ publishes ${R'_{il}}^k \leftarrow [b_{il}(0)]\, R_{il}^k$
        $P_i$ publishes $\left(\pi, \{\pi_j\}_{P_j \in S^*}\right) \leftarrow$
          $\mathsf{PVP}.P\Big(i, b_{il}, S^*, \Big(\Big(R_{il}^k, {R'_{il}}^k\Big), (b_{il}(l))_{P_j \in S^*}\Big)\Big)$
        $P_i$ outputs $E_l^k \leftarrow [b_{il}(0)]\, E_l^{k-1}$
        $P_i$ publishes $zk \leftarrow \mathsf{ZK}.P\left(\left(R_{il}^k, {R'_{il}}^k\right), \left(E_l^{k-1}, E_l^k\right), b_{il}(0)\right)$
        if $\mathsf{ZK}.V\left(\left(R_{il}^k, {R'_{il}}^k\right), \left(E_l^{k-1}, E_l^k\right), zk\right) = \mathsf{false}$ then
            restart without $P_i$
$(c_1, \ldots, c_\lambda) \leftarrow \mathcal{H}\left(E_1^{\#S^*}, \ldots, E_\lambda^{\#S^*}, m\right)$
for $P_i \in S^*$ do
    for $l \in 1, \ldots, \lambda$ do
        $P_i$ outputs $z_{il} = b_{il} - c_l \cdot L_{i,S^*} \cdot s_i$
        for $P_j \in S^*$ do
            $P_j$ computes $b'_{il}(j) \leftarrow z_{il}(j) + c_l L_{i,S^*} f_i(j)$
            and verifies
            $\mathsf{PVP}.V(i, j, S^*, b'_{il}(j), \pi, \pi_j) \wedge$
             $\mathsf{PVP}.V\left(i, 0, S^*, \left(R_{il}^k, {R'_{il}}^k\right), \pi, \pi_0\right)$
            if $P_i$ *is convicted of cheating* then
                restart without $P_i$
for $l \in 1, \ldots, \lambda$ do
    $z_j \leftarrow \sum_{P_i \in S^*} z_{ij}$
return $((c_1, \ldots, c_\lambda), (z_1, \ldots, z_\lambda))$
```

Definition 10 (Self-contained reconstruction). *An instance $\mathcal{S} = (S, \Gamma, G)$ of a secret sharing scheme is called* self-contained*, if, for any authorised set S', the input of any shareholder $P_i \in S'$ in an execution of* Rec *is an element of G.*

It is necessary, that $G = \mathbb{Z}_p$ for some prime p holds to enable the mapping $\cdot \mapsto [\cdot]$. This requirement may be loosened by replacing $\cdot \mapsto [\cdot]$ appropriately. To enable two-level sharing, it has to hold that for a share $s_i \in \mathcal{S}.\mathsf{Share}(s)$ of a secret s, $s_i \in G$ holds. The secret sharing scheme also has to allow for a PVP scheme, that is compatible with a zero-knowledge proof for the GAIP.

5.2 Examples of secret sharing schemes

- It is evident, that Shamir's approach fulfills all aforementioned requirements. In fact, the two-level sharing and the PVP have been tailored to Shamir's polynomial based secret sharing approach.
- Tassa [17] extended Shamir's approach of threshold secret sharing to a hierarchical access structure. To share a secret $s \in \mathbb{Z}_p$ with prime p, a polynomial f with constant term s is sampled. Shareholders of the top level of the hierarchy are assigned interpolation points of f as in Shamir's scheme. The k-th level of the hierarchy receives interpolation points of the $k-1$st derivative of f. The shares in Tassa's scheme are elements of $\mathbb{Z}_p$ themselves. The key generation (Algorithm 1) can easily be transferred to this setting, as each shareholder receives a description of the polynomial utilised in sharing his share. Hence all derivatives and their respective interpolation points can easily be computed. Reconstructing a shared secret is achieved via Birkhoff interpolation, the execution of which is independent and self-contained. The zero-knowledge proof (Algorithm 6 and Algorithm 7) as well as the piecewise verifiable proof (Algorithm 8 and Algorithm 9) thus directly transfer to Tassa's approach utilising the appropriate derivatives in the verifying protocols. The decapsulation and the signing protocols hence can be executed with adjustments only to the verifying steps.
- In 2006, Damgard and Thorbek proposed a linear integer secret sharing scheme [9] with secret space $\mathbb{Z}$. Given an access structure Γ, a matrix M is generated in which each shareholder is assigned a column so that iff $S' \in \Gamma$, the submatrix $M_{S'}$ has full rank. A secret s is shared by multiplying a random vector v with first entry s with M and sending the resulting vector entries to the respective shareholders. Reconstruction follows intuitively. Their scheme hence further generalises Tassa's with respect to secret space and feasible access structures. With the secret space $\mathbb{Z}$ their approach is not compatible with the mapping $\cdot \mapsto [\cdot]$ and our PVP scheme. Thus, neither our key exchange mechanism nor our signature scheme can in its current form be instantiated with Damgard's and Thorbek's scheme.

6 Conclusion

In this work, we presented an actively secure key exchange mechanism based on Shamir's secret sharing scheme and derived a signature scheme from it. The active security measures consist of a piecewise verifiable proof and a zero-knowledge proof for the GAIP, that in combination prove the knowledge of the correct share of the secret key and ensure its use in the protocol. For that we reworked the piecewise verifiable proof and zero-knowledge proof introduced in [2] to fit the threshold setting of Shamir's secret sharing and applied it to the threshold group action of [11]. Active security and simulatability were proven under the assumption of hardness of the decisional parallelisation problem.

Furthermore, we characterised the properties necessary for a secret sharing scheme in order for our key exchange mechanism and signature scheme to be based on it. We gave examples and counter-examples of secret sharing schemes compatible with our approach to demonstrate its limits. We thereby demonstrated that cryptographic schemes with secret shared private key in the HHS setting are not limited to threshold schemes, but applicable to more general access structures.

7 Acknowledgements

We thank Lena Ries, Luca De Feo, and Michael Meyer for inspiring discussions. Philipp Muth was funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297.

References

1. Beth, T., Knobloch, H., Otten, M.: Verifiable secret sharing for monotone access structures. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993. pp. 189–194. ACM (1993), https://doi.org/10.1145/168588.168612
2. Beullens, W., Disson, L., Pedersen, R., Vercauteren, F.: CSI-RAShi: distributed key generation for CSIDH. In: Cheon, J.H., Tillich, J. (eds.) Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20-22, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12841, pp. 257–276. Springer (2021), https://doi.org/10.1007/978-3-030-81293-5_14
3. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11921, pp. 227–247. Springer (2019), https://doi.org/10.1007/978-3-030-34578-5_9
4. Blakley, G.R.: Safeguarding cryptographic keys. In: Merwin, R.E., Zanca, J.T., Smith, M. (eds.) 1979 National Computer Conference: June 4–7, 1979, New York, New York. AFIPS Conference proceedings, vol. 48, pp. 313–317. AFIPS Press, pub-AFIPS:adr (1979)
5. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 -

39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12106, pp. 493–522. Springer (2020), https://doi.org/10.1007/978-3-030-45724-2_17

6. Couveignes, J.M.: Hard homogeneous spaces. IACR Cryptol. ePrint Arch. p. 291 (2006), http://eprint.iacr.org/2006/291
7. Cozzo, D., Smart, N.P.: Sharing the LUOV: threshold post-quantum signatures. In: Albrecht, M. (ed.) Cryptography and Coding - 17th IMA International Conference, IMACC 2019, Oxford, UK, December 16-18, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11929, pp. 128–153. Springer (2019), https://doi.org/10.1007/978-3-030-35199-1_7
8. Cozzo, D., Smart, N.P.: Sashimi: Cutting up CSI-FiSh secret keys to produce an actively secure distributed signing protocol. In: Ding, J., Tillich, J. (eds.) Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12100, pp. 169–186. Springer (2020), https://doi.org/10.1007/978-3-030-44223-1_10
9. Damgård, I., Thorbek, R.: Linear integer secret sharing and distributed exponentiation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings. Lecture Notes in Computer Science, vol. 3958, pp. 75–90. Springer (2006), https://doi.org/10.1007/11745853_6
10. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the fiat-shamir transformation in the quantum random-oracle model. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11693, pp. 356–383. Springer (2019). https://doi.org/10.1007/978-3-030-26951-7_13, https://doi.org/10.1007/978-3-030-26951-7_13
11. Feo, L.D., Meyer, M.: Threshold schemes from isogeny assumptions. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12111, pp. 187–212. Springer (2020), https://doi.org/10.1007/978-3-030-45388-6_7
12. Herranz, J., Sáez, G.: Verifiable secret sharing for general access structures, with application to fully distributed proxy signatures. In: Wright, R.N. (ed.) Financial Cryptography, 7th International Conference, FC 2003, Guadeloupe, French West Indies, January 27-30, 2003, Revised Papers. Lecture Notes in Computer Science, vol. 2742, pp. 286–302. Springer (2003), https://doi.org/10.1007/978-3-540-45126-6_21
13. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 129–140. Springer (1991), https://doi.org/10.1007/3-540-46766-1_9
14. Peikert, C.: He gives c-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia,

May 10-14, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12106, pp. 463–492. Springer (2020), https://doi.org/10.1007/978-3-030-45724-2_16

15. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979), http://doi.acm.org/10.1145/359168.359176
16. Stadler, M.: Publicly verifiable secret sharing. In: Maurer, U.M. (ed.) Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding. Lecture Notes in Computer Science, vol. 1070, pp. 190–199. Springer (1996), https://doi.org/10.1007/3-540-68339-9_17
17. Tassa, T.: Hierarchical threshold secret sharing. In: Naor, M. (ed.) Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2951, pp. 473–490. Springer (2004), https://doi.org/10.1007/978-3-540-24638-1_26
18. Thorbek, R.: Proactive linear integer secret sharing. IACR Cryptol. ePrint Arch. p. 183 (2009), http://eprint.iacr.org/2009/183
19. Traverso, G., Demirel, D., Buchmann, J.: Performing computations on hierarchically shared secrets. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10831, pp. 141–161. Springer (2018), https://doi.org/10.1007/978-3-319-89339-6_9

Appendix A Algorithms

Algorithm 5: Threshold group action

```
Input: $E, S'$
$E^0 \leftarrow E$
$k \leftarrow 0$
for $P_i \in S'$ do
    if $E^k \notin \mathcal{E}$ then
        $P_i$ outputs $\perp$ and aborts.
    else
        $k \leftarrow k + 1$
        $P_i$ outputs $E^k \leftarrow [L_{i,S'} s_i] \, E^{k-1}$
return $E^k$
```

Algorithm 6: The ZK proving protocol for the GAIP

```
Input: $s, (E_i, E_i')_{i=1,\ldots,m}$
for $j = 1, \ldots, \lambda$ do
    $b_j \leftarrow_{\$} \mathbb{Z}_p$
    for $i = 1, \ldots, m$ do
        $\hat{E}_{ij} \leftarrow [b_j] \, E_i$
$(c_1, \ldots, c_\lambda) \leftarrow \mathcal{H}\Big(E_1, E_1', \ldots, E_m, E_m', \hat{E}_{1,1}, \ldots, \hat{E}_{m,\lambda}\Big)$
for $j = 1, \ldots, m$ do
    $r_j \leftarrow b_j - c_j s$
return $\pi \leftarrow (c_1, \ldots, c_\lambda, r_1, \ldots, r_\lambda)$
```

Algorithm 7: The ZK verifying protocol for the GAIP

```
**Input:** $\pi, (E_i, E_i')_{i=1,\ldots,m}$
Parse $(c_1, \ldots, c_\lambda, r_1, \ldots, r_\lambda) \leftarrow \pi$
**for** $i = 1, \ldots, m$ *and* $j = 1, \ldots, \lambda$ **do**
    **if** $c_j == 0$ **then**
        $\tilde{E}_{i,j} \leftarrow [r_j]\, E_i$
    **else**
        $\tilde{E}_{i,j} \leftarrow [r_j]\, E_i'$
$(c_1', \ldots, c_\lambda') \leftarrow \mathcal{H}\Big(E_1, E_1', \ldots, E_m, E_m', \tilde{E}_{1,1}, \ldots, \tilde{E}_{m,\lambda}\Big)$
**return** $(c_1, \ldots, c_\lambda) == (c_1', \ldots, c_\lambda')$
```

Algorithm 8: Proving protocol of the threshold PVP

```
**Input:** $i, f, S^*, \Big((E_0, E_1), (f_i(j))_{P_j \in S^*}\Big)$
**for** $l \in 1, \ldots, \lambda$ **do**
    $b_l \leftarrow_\$ \mathbb{Z}_N[x]_{\leq k-1}$
    $\hat{E}_l \leftarrow [b_l(0)]\, E_0$
$y_0, y_0' \leftarrow_\$ \{0,1\}^\lambda$
$C_0 \leftarrow \mathcal{C}\Big(\hat{E}_1 \| \ldots \| \hat{E}_\lambda, y_0\Big)$
$C_0' \leftarrow \mathcal{C}(E_0 \| E_1, y_0')$
**for** $P_j \in S^*$ **do**
    $y_j, y_j' \leftarrow_\$ \{0,1\}^\lambda$
    $C_j \leftarrow \mathcal{C}(b_1(j) \| \ldots \| b_\lambda(j), y_j)$
    $C_j' \leftarrow \mathcal{C}\big(L_{i,S^*} \cdot f_i(j), y_j'\big)$
$C \leftarrow (C_j)_{P_j \in S^*}$
$C' \leftarrow \big(C_j'\big)_{P_j \in S^*}$
$c_1, \ldots, c_\lambda \leftarrow \mathcal{H}(C, C')$
**for** $l \in 1, \ldots, \lambda$ **do**
    $r_l \leftarrow b_l - c_l \cdot L_{i,S^*} \cdot f$
$\mathbf{r} \leftarrow (r_1, \ldots, r_\lambda)$
$\Big(\pi, \{\pi_j\}_{P_j \in S^*}\Big) \leftarrow \Big((C, C', \mathbf{r}), \big\{\big(y_j, y_j'\big)\big\}_{P_j \in S^*}\Big)$
**return** $\Big(\pi, \{\pi_j\}_{P_j \in S^*}\Big)$
```

Algorithm 9: Verifying protocol of the threshold PVP

```
Input: $i, j, S^*, x_j, (\pi, \pi_j)$
parse $(C, C', \mathbf{r}) \leftarrow \pi$
parse $(y_j, y'_j) \leftarrow \pi_j$
$c_1, \ldots, c_\lambda \leftarrow \mathcal{H}(C, C')$
if $j == 0$ then
    if $C'_j \neq \mathcal{C}(x_j, y'_j)$ then
        return false
    for $l \in 1, \ldots, \lambda$ do
        $\tilde{E}_l \leftarrow [r_l(0)] E_{c_l}$
    return $C_0 == \mathcal{C}(\tilde{E}_1 \| \ldots \| \tilde{E}_\lambda, y_0)$
else
    if $C'_j \neq \mathcal{C}(L_{i,S^*} x_j, y'_j)$ then
        return false
    return $C_j == \mathcal{C}(r_1(j) + c_1 \cdot L_{i,S^*} \cdot x_j \| \ldots \| r_\lambda(j) + c_\lambda \cdot L_{i,S^*} \cdot x_j, y_j)$
```

Algorithm 10: The security game $\mathsf{Exp}^{\text{dist-transcript}}_{\mathcal{A},\mathsf{Sim}}(S)$

```
Input: $\mathcal{S}$
$b \leftarrow_\$ \{0,1\}$
$S^* \leftarrow_\$ \Gamma^+$
$S' \leftarrow_\$ 2^{S^*} \setminus \Gamma$
$\left(\{s_i, f_i, f_j(i)\}_{P_i, P_j \in S}, \mathsf{pk}\right) \leftarrow \mathsf{KeyGen}(\mathcal{S})$
$(\mathcal{K}, c) \leftarrow \mathsf{Encaps}(\mathsf{pk})$
$t_0 \leftarrow \mathsf{Sim}\left(\mathcal{K}, c, \{s_i, f_i, f_j(i)\}_{P_i \in S', P_j \in S}\right)$
$E^0 \leftarrow E_0$, $k \leftarrow 0$
for $P_i \in S^*$ do
    $k \leftarrow k + 1$
    $E^k \leftarrow [L_{i,S^*} s_i] E^{k-1}$
    $R_k \leftarrow_\$ \mathcal{E}$
    $R'_k \leftarrow [L_{i,S^*} s_i] R_k$
    $\left(\pi^k, \{\pi^k_j\}_{P_j \in S^*}\right) \leftarrow \mathsf{PVP}.P\left(i, f_i, S^*, \left((R_k, R'_k), (L_{i,S^*} f_i(j))_{P_j \in S^*}\right)\right)$
    $zk^k \leftarrow \mathsf{ZK}.P\left((R_k, R'_k), (E^{k-1}, E^k), L_{i,S^*} s_i\right)$
$t_1 \leftarrow \left(E^k, \left(\pi^k, \{\pi^k_j\}_{P_j \in S^*}\right), zk^k\right)_{k=1,\ldots,\#S^*}$
$b' \leftarrow \mathcal{A}(t_b)$
return $b == b'$
```

Algorithm 11: Signature verification protocol

```
**Input:** $m, s, \mathsf{pk}$
parse $(c_1, \ldots, c_\lambda, z_1, \ldots, z_\lambda) \leftarrow s$
**for** $j = 1, \ldots, \lambda$ **do**
    **if** $c_j == 0$ **then**
        $E'_j \leftarrow [z_j]\, E_0 = \left[\sum_{P_i \in S^*} b_{ij}\right] E_0$
    **else**
        $E'_j \leftarrow [z_j]\, \mathsf{pk} = \left[\sum_{P_i \in S^*} b_{ij} - L_{i,S^*} s_i + s\right] E_0$
$(c'_1, \ldots, c'_\lambda) \leftarrow \mathcal{H}(E'_1, \ldots, E'_\lambda, m)$
**return** $(c_1, \ldots, c_\lambda) == (c'_1, \ldots, c'_\lambda)$
```