On Actively Secure Fine-grained Access Structures from Isogeny Assumptions

Philipp Muth
Technische Universität Darmstadt,
Germany
muth@seceng.informatik.tu-darmstadt.de

Fabio Campos
Max Planck Institute for Security and Privacy,
Germany
campos@sopmac.de

***Abstract*—We present an actively secure threshold scheme in the setting of Hard Homogenous Spaces (HHS) which allows fine-grained access structures. More precisely, we elevate a given passively secure isogeny based threshold scheme to an actively secure setting. We prove the active security and simulatability of our advanced schemes. By defining some characterising properties, we are able to expand the range of secret sharing schemes which support the given scheme. 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

I. Introduction

The principal motivation for a secret sharing scheme is to split private information into fragments and securely distribute these shares among a group of participants. Then, any collaborating set with a sufficient number of participants/shares is able to reconstruct the shared private information.

Since their introduction in the 1970s by Blakley [1] and Shamir [2], 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 [3], [4], [5], [6].

Post-quantum schemes have, however, only received little attention with respect to secret sharing. Recently, De Feo and Meyer [7] 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 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, he can also not be detected when providing malformed inputs. 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 [8] from their setting of full engagement protocols to a setting of threshold secret sharing. We thereby achieve higher versatility as well as greater efficiency for said measures. Second, we propose an actively secure key exchange mechanism with secret shared secret key utilising the adapted measures for active security. Third, we present an actively secure signature scheme by applying a Fiat-Shamir-transform to our key exchange mechanism. And forth, we characterise the necessary properties for a secret sharing scheme to be compatible with our key exchange mechanism and signature scheme, hence we open those schemes to a significantly wider field of applications.

Related work. Cozzo and Smart [9] investigated the possibility of constructing shared secret schemes based on the Round 2 candidate signature schemes in the NIST standardization process[1]. Based on CSI-FiSh [10], De Feo and Meyer [7] introduced threshold variants of passively secure encryption and signature schemes in the Hard Homogenous Spaces

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

(HHS) setting. Cozzo and Smart [11] presented the first actively secure but not robust distributed signature scheme based on isogeny assumptions. In [8], 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 II the terminology, primitives and security notions relevant for this work are introduced. Section III 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 IV. Finally, the necessary properties for a secret sharing scheme to be compatible with our key exchange mechanism and signature scheme are characterised in Section V in order to enable applying a more general class of secret sharing schemes.

II. Preliminaries

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 are not affected by this.

A. Secret Sharing Schemes

A secret sharing scheme is a cryptographic primitive to distribute a secret s from a secret space among a set of shareholders. An instance $\mathcal{S}$ is defined by a secret space G, a set of shareholders S and an access structure $\Gamma_{\mathcal{S}} \subset 2^S$. A set $S' \in \Gamma$ 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 $A, B \subset S$ with $A \in \Gamma$ and $B \supset A$, 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 $s_1, \dots, s_k$ of a secret s. A share s_i is assigned to a shareholder $P_{\phi(i)}$ via a surjective map $\phi : \{1, \dots, k\} \to \{1, \dots, n\}$ induced by $\Gamma_{\mathcal{S}}$. A set of shareholders $S' \in \Gamma_{\mathcal{S}}$ executes

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

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 of shareholders S' *superauthorised*, if for any $P \in S'$, we have

$$S' \setminus \{P\} \in \Gamma_{\mathcal{S}}.$$

The set of superauthorised sets of shareholders is denoted by $\Gamma^+_{\mathcal{S}}$.

Any superauthorised set is also authorised.

Example 2 (Shamir's secret sharing)
Shamir's secret sharing scheme is defined by $S = \{P_1, \dots, P_n\}$, the access structure $\Gamma = \{S' \subset S \colon \#S' \geq t\}$ for some fixed threshold $1 \leq t \leq n$ and the secret space $G = \mathbb{Z}_p := \mathbb{Z} \bmod p$ for some prime $p > n$. To share a secret $s \in \mathbb{Z}_p$, a random polynomial $f \in \mathbb{Z}_p[X]$ of degree $t-1$ with constant term s is sampled and the shares are defined by

$$s_i = f(i)$$

for $i = 1, \dots, n$. The assigning function ϕ is simply the identity function, thus P_i's share is $f(i)$, $1 \leq i \leq n$. Reconstruction for an authorised set S' is achieved via Lagrange interpolation, that is

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

where $L_{i,S'}$ denotes the Lagrange interpolation coefficients.

The superauthorised sets Γ^+ are

$$\{S' \subset S \colon \#S' > t\}.$$

B. 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 [12] 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 $g * E = E'$.

For ease of notation we define the following.

Definition 3 (Notation)
For a secret sharing instance $\mathcal{S}$ with secret space $\mathbb{Z}_p$ and a hard homogeneous space $(\mathcal{E}, \mathcal{G})$, where $p | \#\mathcal{G}$,

we fix a $g \in \mathcal{G}$ with order p and denote from now on

$$[s]\, E := g^s * E$$

for all $s \in \mathbb{Z}_p$ and all $E \in \mathcal{E}$.

The following problems are assumed to be easily 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 hard homogeneous spaces, such a g always exists, thus any 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
It is obvious that the decisional parallelisation problem reduces to the parallelisation problem, which in turn reduces to the group action inverse problem.

C. Threshold Group Action

Assume, that a secret s has been shared in a Shamir secret sharing instance, thus each shareholder P_i holds a share s_i of s, $i = 1, \ldots, n$. Let E be an arbitrary but fixed element of $\mathcal{E}$. The action $E' \leftarrow [s]\, E$ can be computed by any authorised set S' without reconstructing s by executing the protocol given in algorithm 1.

If algorithm 1 is executed without aborting, 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

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

Algorithm 1: 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

D. Piecewise Verifiable Proofs

A piecewise verifiable proof (PVP) is a cryptographic primitive in the context of hard homogeneous spaces and was first introduced in [8]. It is a compact non-interactive zero-knowledge proof of knowledge of a witness $f \in \mathbb{Z}_q[X]$ for a statement

$$((E_0, E_1), s_1, \ldots, s_n), \tag{II.1}$$

where $E_1 = [s_0]\, E_0 \in \mathcal{E}$ and $s_i = f(i)$ for $i = 0, \ldots, n$. A PVP provides two protocol. The proving protocol $\mathsf{PVP}.P$ takes a statement x of the form (II.1) and a witness f as input and outputs a proof $\left(\pi, \{\pi_i\}_{i=0,\ldots,n}\right)$, where (π, π_i) is a proof piece for the partial statement x_i, $i = 0, \ldots, n$. The verifying protocol $\mathsf{PVP}.V$ takes an index $i \in \{0, \ldots, n\}$, a statement piece x_i and a proof piece (π, π_i) as input and outputs true or false.

Let $\mathcal{R}$ denote the set of all tuples $\{(x, f)\}$, where f is a witness for the statement x. Furthermore, for $I \subset \{0, \ldots, n\}$, we let $\mathcal{R}_I$ denote the set of partial relations $\{(x_I, f)\}$, where there exists $(x, f) \in \mathcal{R}$ so that $x|_I = x_I$.

Definition 4 (Completeness)
We call a PVP *complete*, if, for any $(x, f) \in \mathcal{R}$ and

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

the verification succeeds with overwhelming probability, i.e.,

$$\Pr[\mathsf{PVP}.V\,(j, x_j, (\pi, \pi_j)) = \mathsf{true}] = 1$$

for all $j \in \{0, \ldots, n\}$.

Definition 5 (Soundness)
A PVP is called *sound* if, for any adversary $\mathcal{A}$, any $I \subset \{0, \ldots, n\}$ and any statement x for which there exists no witness f with $(x_I, f) \in \mathcal{R}_I$,

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

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 6 (Zero-knowledge)
A PVP is *zero-knowledge*, if for any $I \subset \{1, \ldots, n\}$ 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 $\left(\pi, \{\pi_j\}_{j=0,\ldots,n}\right) \leftarrow \mathsf{PVP}.P(f, x)$.

We refer to [8] 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 III, we adjust [8]'s proving protocol to our setting of threshold schemes, so that knowledge of a subset of interpolation points for a modified sharing polynomial is proven instead of all interpolation points.

E. Zero-Knowledge Proofs for the GAIP

We define a proving and a verifying protocol to non-interactively prove knowledge of an element $s \in \mathbb{Z}_p$ in zero-knowledge 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 $i = 1, \ldots, m$, simultaneously, without revealing s.

To prove the knowledge of s, the prover samples $b_j \in \mathbb{Z}_p$ and computes

$$\hat{E}_{i,j} \leftarrow [b_j]\, E_i$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, \lambda$. He then derives challenge bits $(c_1, \ldots, c_\lambda) \leftarrow \mathcal{H}\left(E_1, E_1', \ldots, E_m, E_m', \hat{E}_{1,1} \ldots, \hat{E}_{m,\lambda}\right)$ via a hash function $\mathcal{H} : \mathcal{E}^{(2+\lambda)m} \to \{0,1\}^\lambda$ and prepares the answers $r_j \leftarrow b_j - c_j s$, $j = 1, \ldots, \lambda$. The proof $\pi = (c_1, \ldots, c_\lambda, r_1, \ldots, r_\lambda)$ is then published.

The verification protocol is straight forward: for a statement $(E_i, E_i')_{i=1,\ldots,m}$ and a proof $\pi = (c_1, \ldots, c_\lambda, r_1, \ldots, r_\lambda)$, 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 $j = 1, \ldots, \lambda$. Then he generates verification bits $(\tilde{c}_1, \ldots \tilde{c}_\lambda) \leftarrow \mathcal{H}\left(E_1, E_1', \ldots, E_m, E_m', \tilde{E}_{1,1} \ldots, \tilde{E}_{m,\lambda}\right)$ and accepts the proof if $(c_1, \ldots, c_\lambda) = (\tilde{c}_1, \ldots, \tilde{c}_\lambda)$.

A sketch of the proving and verifying protocols can be found in algorithm 2 and algorithm 3, respectively.

Algorithm 2: 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**
    $\hat{E}_{ij} \leftarrow [b_j]\, E_i$
$(c_1, \ldots, c_\lambda) \leftarrow \mathcal{H}\left(E_1, E_1', \ldots, E_m, E_m', \hat{E}_{1,1}, \ldots, \hat{E}_{m,\lambda}\right)$
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 3: 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
    $\tilde{E}_{i,j} \leftarrow [r_j]\, E_i'$
$(c_1', \ldots, c_\lambda') \leftarrow \mathcal{H}\left(E_1, E_1', \ldots, E_m, E_m', \tilde{E}_{1,1}, \ldots, \tilde{E}_{m,\lambda}\right)$
return $(c_1, \ldots, c_\lambda) == (c_1', \ldots, c_\lambda')$

We again refer to [8] for the proof of the presented algorithms being complete, sound and zero-knowledge with respect to the security parameter λ.

F. 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 falsify the output of the execution of our protocol without being detected.

G. Communication channels

Both our schemes assume the existence of a trusted dealer in addition to the shareholders engaged in a secret sharing instance. The dealer samples and shares a private key and publishes the according public key. The shareholders store the private key and execute the multiparty protocols for decapsulation in our key exchange mechanism (KEM) and signing in our signature scheme.

For the communication between the dealer and the shareholders we assume secure private channels. Thus messages sent through these channels cannot be tampered with or eavesdropped upon without detection. For the communication between the shareholders, however, a simple public channel is sufficient, since all messages sent by shareholders are being broadcast. The means of how to establish secure private channels and immutable broadcast channels are out of scope of this work.

III. Key Exchange Mechanism

We present our actively secure key exchange protocol with secret shared private key. A key exchange mechanism (KEM) provides three functions: KeyGen, Encaps and Decaps.

A. Public Parameters

We fix the following public parameters.

- A Shamir sharing instance $\mathcal{S}$ with shareholders $\mathcal{S} = \{P_1, \dots, P_n\}$, threshold $t < n$ and secret space $\mathbb{Z}_p$. In Section V we elaborate possible extensions to other, more general secret sharing schemes.
- A hard homogeneous space $(\mathcal{E}, \mathcal{G})$ with fixed starting point $E_0 \in \mathcal{E}$.
- An element $g \in \mathcal{G}$ with $\mathsf{ord} g = p$ for the mapping $[\cdot] : \mathcal{G} \times \mathcal{E} \to \mathcal{E}; s \mapsto g^s E$.

B. Key Generation

We assume the existence of a trusted dealer, yet even an untrusted dealer can be accommodated with little overhead. For key generation, the dealer samples the secret key $s \in \mathbb{Z}_p$ and publishes the public key $\mathsf{pk} \leftarrow [s]E_0$. The secret key s is shared via $(s_1, \dots, s_n) \leftarrow \mathcal{S}.\mathsf{Share}(s)$. Then, each share s_i, $1 \le i \le n$, is shared once more, resulting in n sets of n shares each.

$$\forall i = 1, \dots, n\colon \{s_{i1}, \dots, s_{in}\} \leftarrow \mathcal{S}.\mathsf{Share}(s_i)$$

The dealer then sends the following set to shareholder P_i, $i = 1, \dots, n$:

$$\left\{ s_i, \{s_{ij}\}_{j=1,\dots,n}, \{s_{ki}\}_{k=1,\dots,n} \right\},$$

thus each shareholder P_i receives his share s_i of the secret key s. He also receives the shares by which s_i was shared and his shares of each other shareholder's share of s.

For ease of notation we denote the polynomial with which the secret key s was shared by f and the polynomial with which s_i was shared by f_i, where $i = 1, \dots, n$.

This key generation protocol can be considered as a "two-level sharing", where each share of the secret key is itself shared again among the shareholders. A sketch of it can be found in algorithm 4.

Algorithm 4: Key generation

$s \leftarrow_{\$} \mathbb{Z}_p$
$\mathsf{pk} \leftarrow [s]\, E_0$
$\{s_1, \dots, s_n\} \leftarrow \mathcal{S}.\mathsf{Share}(s)$
for $i = 1, \dots, n$ **do**
  $\{s_{i1}, \dots, s_{in}\} \leftarrow \mathcal{S}.\mathsf{Share}(s_i)$
publish pk
for $i = 1, \dots, n$ **do**
  send $\left\{ s_i, \{s_{ij}\}_{j=1,\dots,n}, \{s_{ki}\}_{k=1,\dots,n} \right\}$ to P_i

C. 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 [7], thus we just give a short sketch and refer to De Feo's and Meyer's work for the respective proofs of security.

Algorithm 5: Encapsulation

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

D. Decapsulation

A decapsulation protocol takes a ciphertext c and outputs a key $\mathcal{K}$.

De Feo and Meyer [7] applied the threshold group action (algorithm 1) to a ciphertext c to have an authorised set of shareholders compute $[s]\,c = [s]\,(b * E_0) = b * ([s]\,E_0)$, thereby decapsulating the ciphertext and obtaining the session key. While their approach is simulatable and thereby does not leak any information on the shares of the secret key, it is only passively secure. Thus, a malicious shareholder can provide malformed input to the protocol and thereby falsify the output without being detected.

We extend their approach to enable detecting misbehaving shareholders in a decapsulation. For that we maintain the threshold group action, yet we apply the PVP and zero-knowledge proof layed out in Section II. Since the PVP does not fit our setting of threshold group action, we first discuss the necessary modifications to the PVP. We then present our actively secure decapsulation protocol.

1) Amending the PVP: In the PVP protocol, that we sketched in Section II, a prover produces a proof of knowledge for a witness polynomial f of the statement

$$\left((E_0, E_1), s_1, \ldots, s_n\right),$$

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 does not fit the threshold group action, since, for an authorised set S', a shareholder P_i's contribution to the round-robin approach is not $E^k \leftarrow [s_i]\,E^{k-1}$, where E^{k-1} denotes the previous shareholder's output, but $E^k \leftarrow [L_{i,S'} s_i]\,E^{k-1}$. Authorised sets also do not necessarily contain all shareholders $\{P_1, \ldots, P_n\}$. The following example illustrates a further conflict with of the PVP with the threshold group action.

Example 7
Let sk be a secret key generated and shared by KeyGen. That is each shareholder P_i holds

$$\left\{ s_i, \{s_{ij}\}_{P_j \in S}, \{s_{ji}\}_{P_j \in S} \right\}.$$

Also let $S' \in \Gamma$ be a minimally authorised set executing algorithm 1, i.e., for any $P_i \in S'$, $S' \setminus \{P_i\}$ 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 $f'_i(j) = L_{i,S'} s_{ij}$ and $R' = [f'_i(0)]\,R$ for any $R, R' \in \mathcal{E}$. Thus, P_i can publish $\left(\pi, \{\pi_j\}_{P_j \in S'}\right)$ with

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

which to $S' \setminus \{P_i\}$ is indistinguishable from

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

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 [8]'s PVP protocol, so that, for a superauthorised set S^*, a shareholder $P_i \in S^*$ proves knowledge of a witness polynomial $L_{i,S^*} f_i$ for a statement

$$\left((R, R'), (L_{i,S^*} s_{ij})_{P_j \in S^*}\right),$$

where $R \leftarrow\!\!\$\, \mathcal{E}$, $R' = [L_{i,S^*} s_i]\,R$, $s_{ij} = f_i(j)$ for $P_j \in S^*$ and s_i was shared via f_i. The inputs of our amended proving protocol are the proving shareholder's index i, the witness polynomial f, the superauthorised set $S^* \in \Gamma^+$ and the statement $\left((R, R'), (s_{ij})_{P_j \in S^*}\right)$. The protocol can be found in algorithm 6. By $\mathcal{C}$ we denote 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 7.

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

Definition 8 (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

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

for all $P_j \in S'$.

The proofs for soundness, correctness and zero-knowledge for Beullens et al.'s [8] 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.

Algorithm 6: Proving protocol of the threshold PVP

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

Algorithm 7: 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 $C_0 == \mathcal{C}\left(\tilde{E}_1 \| \ldots \| \tilde{E}_\lambda, y_0\right)$
else
  if $C_j' \neq \mathcal{C}\left(L_{i,S^*} x_j, y_j'\right)$ **then**
    return false
  return $C_j == \mathcal{C}(r_1(j) + c_1 \cdot L_{i,S^*} \cdot x_j \| \ldots \| r_\lambda(j) + c_\lambda \cdot L_{i,S^*} \cdot x_j, y_j)$

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 $\left(\pi^k, \left\{\pi_j^k\right\}_{P_j \in S^*}\right) \leftarrow \mathsf{PVP}.P\left(i, f_i, S^*, \left((R_k, R_k'), (s_{ij})_{P_j \in S^*}\right)\right)$
4) Each shareholder $P_j \in S^* \setminus \{P_i\}$ verifies $\mathsf{PVP}.V\left(i, j, S^*, s_{ij}, \left(\pi^k, \pi_j^k\right)\right)$ and $\mathsf{PVP}.V\left(i, 0, S^*, (R_k, R_k'), \left(\pi^k, \pi_0^k\right)\right)$. If either should fail, P_j issues a complaint against P_i and publishes s_{ij}. If P_i is convicted of cheating by more than $\#S^*/2$ shareholders, decapsulation is restarted with an $S^{*\prime} \in \Gamma^+$, so that $P_i \notin S^{*\prime}$.
5) If the PVP was accepted, P_i computes $E^k \leftarrow [L_{i,S^*} s_i]\, E^{k-1}$ as well as the zero-knowledge proof $zk \leftarrow \mathsf{ZK}.P\left((R_k, R_k'), \left(E^{k-1}, E^k\right), L_{i,S^*} s_i\right)$. He publishes both.
6) If $\mathsf{ZK}.V\left((R_k, R_k'), \left(E^{k-1}, E^k\right), zk\right)$ fails to verify, decapsulation is restarted with a set $S^{*\prime} \in \Gamma^+$, where $P_i \notin S^{*\prime}$.
7) Otherwise, P_i's turn is finalised and the next shareholder continues with E^k as input from P_i.
8) 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 5 ensure, that P_i not only has knowledge of the sharing polynomial $L_{i,S^*} f_i$ but also inputs $L_{i,S^*} f_i(0)$ to compute E^k. The precise protocol can be found in algorithm 8.

Definition 9
A KEM 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 KEM presented in algorithm 4, algorithm 5 and algorithm 8 follows from the correctness of the threshold group action (algorithm 1). Let pk be a public key and $\mathsf{sk} = [\mathsf{pk}]\, E_0$ be the respective secret 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 $\mathcal{K}' \leftarrow$

Algorithm 8: 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.
  $k \leftarrow k+1$
  $R_k \leftarrow_{\$} \mathcal{E}$
  $R'_k \leftarrow [L_{i,S^*} s_i]\, R_k$
  $\left(\pi^k, \left\{\pi^k_j\right\}_{P_j \in S^*}\right) \leftarrow \mathsf{PVP}.P\left(i, f_i, S^*, ((R_k, R'_k), (L_{i,S^*} s_{ij})_{P_j \in S^*})\right)$
  P_i publishes (R_k, R'_k) and $\left(\pi^k, \left\{\pi^k_j\right\}_{P_j \in S^*}\right)$
  Each $P_j \in S^* \setminus \{P_i\}$ checks
  $b_j \leftarrow \mathsf{PVP}.V\left(i, j, S^*, L_{i,S^*} s_{ij}, \left(\pi^k, \pi^k_j\right)\right)$
  if $b_j = \mathsf{false}$ *for some* P_j **then**
    P_j publishes s_{ij}
    if P_i *is convicted* **then**
      return $\mathsf{Decapsulation}\left(c, S^{*\prime}\right)$ *with* $S^{*\prime} \in \Gamma \wedge P_i \notin S^{*\prime}$
  $E^k \leftarrow [L_{i,S^*} s_i]\, E^{k-1}$
  $zk^k \leftarrow \mathsf{ZK}.P\left(\left(R_k, R'_k\right), \left(E^{k-1}, E^k\right), L_{i,S^*} s_i\right)$
  P_i publishes $\left(E^k, zk\right)$
  Each $P_j \in S^* \setminus \{P_i\}$ checks
  if $\mathsf{ZK}.V\left(\left(R_k, R'_k\right), \left(E^{k-1}, E^k\right), zk\right) = \mathsf{false}$ **then**
    return $\mathsf{Decapsulation}\left(c, S^{*\prime}\right)$ *with* $S^{*\prime} \in \Gamma \wedge P_i \notin S^{*\prime}$
return $\mathcal{K} \leftarrow E^k$

$\mathsf{Decaps}(c, S')$. Now, after executing $\mathsf{Decaps}(c, S')$, we have $\mathcal{K}' = E^{\#S'}$, which emerges as a result of applying the threshold group action to c. This gives us

$$\mathcal{K}' = \left[\sum_{P_i \in S'} L_{i,S'} s_i\right] c = [\mathsf{sk}]\,(b * E_0) = b * \mathsf{pk} = \mathcal{K}.$$

Our decapsulation is executed by superauthorised sets S^*, which are authorised. This shows that our KEM is correct.

E. 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 either 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.

1) Active security:

Theorem 10
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

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

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

$$\mathsf{PVP}.V\left(i', j, S^*, L_{i',S^*} s_{i'j}, \left(\pi^{k'}, \pi_j^{k'}\right)\right) = \text{true} \tag{III.1}$$

$$\mathsf{PVP}.V\left(i', 0, S^*, \left(R_{k'}, R'_{k'}\right), \left(\pi^{k'}, \pi_0^{k'}\right)\right) = \text{true} \tag{III.2}$$

$$\mathsf{ZK}.V\left(\left(E^{k'-1}, E^{k'}\right), \left(R_{k'}, R'_{k'}\right), zk^{k'}\right) = \text{true} \tag{III.3}$$

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

$$E^{k'} = [\alpha]\, E^{k'-1},$$

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 (III.3) 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 (III.1) and (III.2) 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

$$f'(j) = L_{i',S^*} s_{ij} \tag{III.4}$$

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 (III.4). 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 $\alpha = f'(0) = L_{i',S^*} f_{i'}(0) = L_{i',S^*} s_{i'j}$. We arrive at a contradiction, assuming the soundness of the PVP. ■

2) 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. For that we prove the simulatability of the decapsulation.

Theorem 11
The decapsulation protocol presented in algorithm 8 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 [8] 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

$$\left\{s_i, \{s_{ij}\}_{P_i \in S^*}, \{s_{ji}\}_{P_j \in S^* \setminus \mathcal{A}}\right\}_{P_i \in \mathcal{A}}.$$

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
$$\forall P_j \in \mathcal{A} : f'_i(i) = s_{ij}$$
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 $\mathsf{PVP}.P\left(i, f'_i, S^*, \left((R_k, R'_k), (L_{i,S^*} s_{ij})_{P_j \in S^*}\right)\right)$, $E^k \leftarrow [L_{i,S^*} s'_i]\, E^{k-1}$ and $\mathsf{ZK}.P\left((R_k, R'_k), \left(E^{k-1}, E^k\right), L_{i,S^*} f'_i(0)\right)$, where k is the index of P_i's output in the transcript, $s_{ij} := f'_i(j)$ for $P_j \in S^* \setminus \mathcal{A}$ and $s'_i := f'_i(0)$. Since, for all $P_i \in S^* \setminus \mathcal{A}$, s_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 $\left(\left(R_{k'}, R'_{k'}\right), \left(L_{i,S^* s_{i'j}}\right)_{P_j \in S^*}\right)$. 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 $\left(R_{k'}, R'_{k'}\right)$ 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 $E^{k'} \leftarrow [L_{i',S^*} s'_{i'}]\, E^{k'-1}$. 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 [8], 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 $R_{k'} \xleftarrow{\$} \mathcal{E}$ 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

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

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. ■

F. *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 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.

IV. Actively Secure Secret Shared Signature Protocols

We convert the key exchange mechanism in algorithm 4, algorithm 5 and algorithm 8 into an actively secure signature scheme with secret shared signing key. 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.

A signature scheme consists of three protocols: key generation, signing and verifying. We transfer the unmodified key generation protocol from the KEM in section III to our signature scheme. The

signing protocol is derived from the decapsulation protocol (algorithm 8) by applying the Fiat-Shamir-transformation, the verifying protocol follows straightforward. The protocols are given in algorithm 9 and algorithm 10.

Both simulatability and active security of the signing protocol can be proven in a manner similar to that of Theorem 10. Thus we skip the proofs deeming them only little instructive.

V. Generalising the secret sharing schemes

We constructed the protocols above in the context of Shamir's secret sharing protocol [2]. The key exchange mechanism in section III as well as the signature scheme in section IV can be extended to more general secret sharing schemes. We first list the requirements that a secret sharing scheme has to meet in order to successfully implement the KEM and the signature scheme, then we give some examples of secret sharing schemes that fulfill said requirements.

A. Compatibility requirements

Definition 12 (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 KEM and signature scheme has to have indenpendent reconstruction, since each shareholder's input into the threshold group action is hidden from every other party by virtue of the GAIP.

Definition 13 (Self-contained reconstruction)
A secret sharing instance $\mathcal{S} = (S, \Gamma, G)$ has *self-contained reconstruction*, if, for any $S' \in \Gamma$ and any share $s \in G$, the input of each shareholder $P_i \in S'$ to reconstructing s is in G, so that reconstruction can be represented as an iterated application of the action of G.

For the secret space, $G = \mathbb{Z}_p$ for some prime p is necessary to enable the mapping $\cdot \mapsto [\cdot]$. This requirement may be loosened by replacing $\cdot \mapsto [\cdot]$ with an appropriate alternative. Also, for a secret s and any $s_i \in \{s_1, \ldots, s_k\} \leftarrow \mathcal{S}.\mathsf{Share}(s)$, $s_i \in G$ has to hold to enable key generation with two-level sharing.

Furthermore, a PVP scheme, that is compatible with the secret sharing scheme, has to exist and agree with a zero-knowledge proof with respect to the GAIP.

B. Examples of secret sharing schemes

We give two examples of secret sharing schemes that fulfill the aforementioned conditions and two counter examples.

- It is evident, that Shamir's approach fulfills all aforementioned requirements. We point out, that the two-level sharing and the PVP have been tailored to Shamir's polynomial based secret sharing approach.
- Tassa [13] introduced a hierarchical secret sharing scheme also based on sharing via a randomly sampled polynomial. To share a secret s, a polynomial f in $\mathbb{Z}_p[X]$ is sampled with constant term s. Shareholders of the top rank in the hierarchy are assigned interpolation points of f. The second rank is assigned points on the first derivative, in short, shareholders of the k-th rank obtain interpolation points of the $k-1$st derivative. With the shares being in Z_p, this enables the necessary two-level sharing. The polynomial based sharing approach agrees with the PVP protocol given above with some minor adjustments. Thus, transferring the KEM and the signature scheme to Tassa's secret sharing can easily be achieved.
- In 2006, Damgard and Thorbek proposed a linear integer secret sharing scheme [14]. They enable a wide range of access structures by representing a given access strucure as a sharing matrix. For an integer secret s to be shared, a random vector with first entry s is sampled and multiplied with the sharing matrix. Thus reconstruction for an authorised set is achieved by inverting the corresponding submatrix corresponding and multiplying the set of their shares with the inverted matrix. With s and the shares being unbounded, Damgard's and Thorbek's scheme is not compatible with the mapping $\cdot \mapsto [\cdot]$. Also their scheme does not comply with our PVP scheme. In its current form, our KEM and signature scheme cannot be instantiated with [14]'s approach. If a suitable PVP and substitution for $\cdot \mapsto [\cdot]$ can be found, an instantiation with their scheme is feasible.
- Additive secret sharing is the simplest of secret sharing schemes. For a given secret s, each shareholder P_i receives a share s_i, $i = 1, \ldots, n$, with $s = \sum_{P_i} s_i$. Additive secret sharing has self-contained as well as independent reconstruction. Yet it is a full threshold secret sharing scheme, that is $\Gamma = \{S\} = \{\{P_1, \ldots, P_n\}\}$. Thus for any $P_i \in S$, the remaining sharehold-

Algorithm 9: 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 $\left(\pi, \{\pi_j\}_{P_j \in S^*}\right) \leftarrow \mathsf{PVP}.P\left(i, b_{il}, S^*, \left(\left(R_{il}^k, {R'_{il}}^k\right), (b_{il}(l))_{P_j \in S^*}\right)\right)$
    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 $b'_{il}(j) \leftarrow z_{il}(j) + c_l L_{i,S^*} s_{ij}$
      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 $\left((c_1, \ldots, c_\lambda), (z_1, \ldots, z_\lambda)\right)$

Algorithm 10: Signature verification protocol

Input: m, s, 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 $(c_1, \ldots, c_\lambda) == (c'_1, \ldots, c'_\lambda)$

ers $\{P_1, \ldots, P_n\} \setminus \{P_i\}$ form an unauthorised set. Thus active security cannot be provided for the threshold group action, making additive secret sharing incompatible with our KEM and signature scheme.

VI. Conclusion and Future Work

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 [8] to fit the threshold setting of Shamir's secret sharing and applied it to the threshold group action of [7]. 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 KEM and signature scheme to be based on it. We gave [2] and [13] as two examples of secret sharing schemes that fulfill the necessary conditions and to which our scheme can feasible be transfered to. The linear integer secret sharing scheme of [14] is incompatible with our schemes, yet a workaround does not seem unfeasible. With additive secret sharing we also gave a counter-example, to show the limits of our model. We thereby demonstrated that cryptographic schemes with secret shared private key in the HHS setting are not limited to threshold secret sharing schemes, but a wider variety of schemes and access structures can be utilised.

For future work we envision transferring a wider range of public key cryptographic schemes to a secret shared setting in hard homogeneous spaces, that is not restricted to Shamir's approach. Another promising direction is improving the efficiency of the PVP and zero-knowledge proof to reduce the message size, since the main portion of communication is spent on these security measures and thereby improving the overall efficiency considerably. Also, generalising our scheme to enable an even wider field of secret sharing schemes to be applied is an interesting task.

VII. 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] G. R. Blakley, "Safeguarding cryptographic keys," *Proceedings of AFIPS 1979 National Computer Conference*, vol. 48, pp. 313–317, 1979.

[2] A. Shamir, "How to share a secret," *Communications of the Association for Computing Machinery*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[3] Y. Lindell and A. Nof, "Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM, 2018, pp. 1837–1854. [Online]. Available: https://doi.org/10.1145/3243734.3243788

[4] J. Doerner, Y. Kondi, E. Lee, and a. shelat, "Threshold ECDSA from ECDSA assumptions: The multiparty case," in *2019 IEEE Symposium on Security and Privacy*. San Francisco, CA, USA: IEEE Computer Society Press, May 19–23, 2019, pp. 1051–1066.

[5] J. Doerner, Y. Kondi, E. Lee, and A. Shelat, "Secure two-party threshold ECDSA from ECDSA assumptions," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 980–997. [Online]. Available: https://doi.org/10.1109/SP.2018.00036

[6] L. T. A. N. Brandao, M. Davidson, and A. Vassilev, "NIST roadmap toward criteria for threshold schemes for cryptographic primitives," Jul 2020. [Online]. Available: http://dx.doi.org/10.6028/NIST.IR.8214A

[7] L. De Feo and M. Meyer, "Threshold schemes from isogeny assumptions," in *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, ser. Lecture Notes in Computer Science, A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, Eds., vol. 12111. Edinburgh, UK: Springer, Heidelberg, Germany, May 4–7, 2020, pp. 187–212.

[8] W. Beullens, L. Disson, R. Pedersen, and F. Vercauteren, "CSI-RAShi: Distributed key generation for CSIDH," Cryptology ePrint Archive, Report 2020/1323, 2020, https://eprint.iacr.org/2020/1323.

[9] D. Cozzo and N. P. Smart, "Sharing the LUOV: Threshold post-quantum signatures," in *17th IMA International Conference on Cryptography and Coding*, ser. Lecture Notes in Computer Science, M. Albrecht, Ed., vol. 11929. Oxford, UK: Springer, Heidelberg, Germany, Dec. 16–18, 2019, pp. 128–153.

[10] W. Beullens, T. Kleinjung, and F. Vercauteren, "CSI-FiSh: Efficient isogeny based signatures through class group computations," in *Advances in Cryptology – ASIACRYPT 2019, Part I*, ser. Lecture Notes in Computer Science, S. D. Galbraith and S. Moriai, Eds., vol. 11921. Kobe, Japan: Springer, Heidelberg, Germany, Dec. 8–12, 2019, pp. 227–247.

[11] D. Cozzo and N. P. Smart, "Sashimi: Cutting up CSI-FiSh secret keys to produce an actively secure distributed signing protocol," in *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, J. Ding and J.-P. Tillich, Eds. Paris, France: Springer, Heidelberg, Germany, Apr. 15–17 2020, pp. 169–186.

[12] J.-M. Couveignes, "Hard homogeneous spaces," Cryptology ePrint Archive, Report 2006/291, 2006, https://eprint.iacr.org/2006/291.

[13] T. Tassa, "Hierarchical threshold secret sharing," in *TCC 2004: 1st Theory of Cryptography Conference*, ser. Lecture Notes in Computer Science, M. Naor, Ed., vol. 2951. Cambridge, MA, USA: Springer, Heidelberg, Germany, Feb. 19–21, 2004, pp. 473–490.

[14] I. Damgård and R. Thorbek, "Linear integer secret sharing and distributed exponentiation," in *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, ser. Lecture Notes in Computer Science, M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds., vol. 3958. New York, NY, USA: Springer, Heidelberg, Germany, Apr. 24–26, 2006, pp. 75–90.