Switched Threshold Signatures from K-Private PolyShamir Secret Sharing

Kristian L. McDonald[1]

Clearmatics
London, UK

Abstract

Variant secret sharing schemes deriving from Shamir's threshold secret sharing protocol [Sha79] are presented. Results include multi-secret sharing protocols using shares with $\mathcal{O}(1)$ elements, independent of the number of secrets. The new schemes achieve a weaker notion of security (they're secure rather than strongly secure) but maintain a property called K-privacy (inspired by k-anonymity [SS98]). K-privacy ensures that all secrets remain private with respect to a subset of the secret space, though the particular subset providing privacy may vary among adversaries that acquire distinct sub-threshold sets of shares. Depending on the number of secrets and the protocol details, secure K-private multi-secret sharing schemes may be "almost" strongly secure or may remain merely secure and K-private - a difference captured by the notion of K-security. Novel applications of the multi-secret sharing schemes are presented, realising a primitive called a *switched threshold signature*. Switched threshold signatures have the quirky property that aggregating a threshold number of signatures of one type (e.g. Pointcheval-Sanders signatures [PS16]) "switches" the signatures into a master signature of a different type. Collectively these results may permit efficiencies within, e.g., threshold credential issuance protocols.

[1]klmcd protonmail com

Contents

1 Introduction

Shamir's threshold secret sharing scheme permits t-out-of-n friends/share-holders to collaboratively retrieve a secret via polynomial interpolation [Sha79]. Blakely also (independently) introduced the notion of threshold secret sharing [Bla79]. Since these discoveries, secret sharing schemes have been widely researched and deployed, with many variant schemes appearing. Overviews of the various approaches can be found in, e.g., an early article by Stinson [Sti92] or the more-recent review by Beimal [Bei11]. Examples of the many variants in the literature include t-out-of-t threshold secret sharing [KMG+83], multi-secret sharing [KMG+83, BDSDC+94, JMO94] and protocols with dependent secrets (in which the shares have reduced complexity [FY92]). Secret sharing protocols were also generalised to permit *access structures*, allowing specified subsets of share-holders to reconstruct a secret while forbidding secret reconstruction by non-authorised sets of share-holders [ISN89, BL90]. The so-called monospan program approach, which can also accommodate multi-secret sharing, found use in this context [BI93, BL90].

Other variant protocols include pseudo-random secret sharing, in which randomness initially distributed via an execution of Shamir's scheme is subsequently used to generate multiple (pseudo-)random shared secrets without communicating [CDI05]. Variant threshold secret sharing protocols alleviate the need for a trusted party [Ped91], and verifiable secret sharing schemes [CGMA85] allow secret-owners to commit to a polynomial encoding of their secret by separately committing to each coefficient in the polynomial [Fel87]. The subsequent discovery of polynomial commitment schemes [KZG10] permitted the construction of verifiable secret sharing protocols leveraging commitments containing a single group element [KZG10]. Secret sharing protocols were also leveraged to construct threshold signature schemes (e.g. Shoup's well-known RSA threshold signature protocol [Sho00]) and threshold credential issuance schemes [SAB+19]. Generalisations of multi-linear secret sharing schemes permit the use of polynomials for shares [PCR19] and a recent work performed a fine-grained analysis of secret sharing schemes [CDS19].

The present work proposes generalisations of Shamir's secret sharing scheme. The focus is on multi-secret sharing protocols related to both Shamir's protocol and the variant in which the secret is generated during protocol execution. A number of multi-secret sharing protocols are presented, including one in which users share multiple secrets by issuing shares with $\mathcal{O}(1)$ elements. Such schemes permit a reduction in the work required to generate and distribute shares, which may be of interest for, e.g., threshold credential schemes such as Coconut [SAB+19]. However, the standard trade-off between security and efficiency holds, with the resulting protocol being secure and K-private (a notion inspired by k-anonymity [SS98]) but not strongly secure (as defined below). Secure K-private q-secret sharing schemes may be "almost" strongly secure, depending on the number of secrets q and the details of the protocol. For example, one new protocol is K-private and secure for all q, but has even stronger security properties for *particular* values of q. These properties motivate the notion of K-security as a kind of "middle ground" security category between K-private security and strong security.

Example applications of the multi-secret sharing schemes are also presented, realising a

primitive referred to as a *switched threshold signature*. A switched threshold signature is a type of threshold signature with quirky properties, such that aggregating a threshold number of signatures of one type "switches" the signatures to achieve a master signature of a different type. For example, leveraging the new multi-secret sharing schemes, switched threshold signature schemes are presented in which a threshold number of Pointcheval-Sanders signatures [PS16] are aggregated to form a polynomial-based [McD20] aggregate/master signature (and vice-versa). Variant switched threshold signatures leveraging Camenisch-Lysyanskaya signatures [CL04] also appear possible.

The layout of this paper is as follows. Section 2 briefly reviews Shamir's single-secret sharing protocol and a related multi-secret sharing protocol. This section mainly sets notations and establishes definitions (experts need only skim it). The variant of Shamir's protocol which generates the secret during protocol execution is discussed in Section 3, and multiple multi-secret generalisations of this scheme are presented. To better define the security properties of the new schemes, the notions of secret sharing K-privacy and K-security are introduced in Section 4. Applications of the new secret sharing schemes which realise switched threshold signatures appear in Section 5. Brief comments on a larger family of related multi-secret sharing schemes are offered in Section 6 and the paper concludes in Section 7.

2 Shamir Secret Sharing

2.1 Single-Secret Sharing

Shamir secret sharing allows an individual to store and recover a secret by distributing shares to n friends, such that t-out-of-n shares are required to reconstruct the secret [Sha79]. The single-secret sharing protocol Shamir is defined as follows:

- Shamir:
  - Select a secret value $\bar{y} \in \mathbb{F}_p^*$, and a set of n friends, labeled by $a \in [n] \equiv \{1, 2, \ldots, n\}$.
  - Draw $t-1$ field elements uniformly at random, $\alpha_c \xleftarrow{R} \mathbb{F}_p^*$, for $c \in [t-1]$.
  - Use $\bar{y}$ and the elements α_c to construct the polynomial $f(x) = \bar{y} + \sum_{c=1}^{t-1} \alpha_c x^c$.
  - Evaluate $f(x)$ at n distinct *evaluation points*, $x \in \{x_a\}_{a\in[n]}$. The pair (x_a, sh_a), with $\mathsf{sh}_a \equiv y_a = f(x_a)$, is called a *secret share* or simply a *share*.[1]
  - Distribute a single share to each of the n friends.

[1]For brevity, the evaluation point for a share, namely x_a, is not always explicitly shown - shares are denoted simply as sh_a. In all secret sharing schemes discussed in this work, either the friends must store the values of x_a (and return them when requested) or users must store a dictionary relating each friend a with an evaluation point x_a. Typically the evaluation points may be given convenient assignments, such as $x_a = a$, $\forall a \in [n]$. For additional security, users may keep the evaluation points confidential and only share the shares sh_a.

– Secret recovery:
  * Select a subset of t friends, labeled as $\mathcal{T}_* \subseteq [n]$, with $|\mathcal{T}_*| = t$, and request their shares.
  * Reconstruct the polynomial $f(x)$ using the shares $\mathsf{sh}_b = y_b$, for $b \in \mathcal{T}_*$, via interpolation.
  * Evaluate the polynomial at the origin to recover the secret, $f(0) = \bar{y}$.

Regarding notation, in this document indices a and b generally denote elements in the sets $[n]$ and $\mathcal{T}_* \subseteq [n]$ respectively (with $|\mathcal{T}_*| = t$); e.g. $a \in [n]$ and $b \in \mathcal{T}_*$, where $[n]$ denotes the set $\{1, 2, \ldots, n\}$. The index c generally denotes elements in a subset of $[n]$ with cardinality $t-1$, such as $c \in [t-1]$ or $c \in \mathcal{T}$, with $|\mathcal{T}| = t-1$, where $[t-1] \subset [n]$ and $\mathcal{T} \subset \mathcal{T}_* \subseteq [n]$. In schemes where $t = n$, the index a is also used as $a \in [t]$ (i.e. a labels the full set of friends/shares in both cases). A *negligible* quantity in security parameter λ is smaller than $\mathcal{O}(1/\mathrm{poly}(\lambda))$ and denoted as $\mathsf{negl}(\lambda)$.

There are multiple ways to formally define the notion of secret sharing security. Protocol Shamir requires users to first select a secret, then generate the shares. However, protocols discussed below require users to first generate shares, then obtain the secret. Accordingly, the following definition of secret sharing security, which is suitable for both classes of secret sharing schemes (i.e. secret-first or shares-first), is adopted.

Definition 2.1. *Secret Sharing Security.* A t-out-of-n ($t \leq n$) secret sharing protocol $\mathcal{P}$ is said to be *secure* if any probabilistic polynomial time adversary $\mathcal{A}$ that obtains $t-1$ shares $\{y_c\}_{c \in \mathcal{T}}$, with $|\mathcal{T}| = t-1$, has a negligible chance of outputting the corresponding secret $\bar{y}$. Namely:

$$\Pr\left[\begin{array}{c}\mathbb{F}_p \leftarrow \mathcal{G}(1^\lambda);\ (\bar{y}, \{y_a\}_{a\in[t]}) \leftarrow \mathcal{P}(\mathsf{input});\ \bar{t} \xleftarrow{R} [t];\ \mathcal{T} \equiv [t]\backslash\{\bar{t}\};\\ \bar{z} \leftarrow \mathcal{A}(\{y_c\}_{c\in\mathcal{T}}) :\ \bar{z} = \bar{y}\end{array}\right] = \mathsf{negl}(\lambda),$$

where input denotes (optional) input data which may include the secret $\bar{y}$.

This security definition requires that a probabilistic polynomial time (PPT) adversary which acquires $t-1$ shares has a negligible probability of determining the corresponding secret. Here $\mathcal{G}(1^\lambda)$ is an object generator that outputs a finite field $\mathbb{F}_p$, for prime p, on input a security parameter (in unary form). The input data to protocol $\mathcal{P}$ may include the secret $\bar{y}$ itself, as occurs for $\mathcal{P} =$ Shamir. Protocol Shamir is unconditionally (i.e. information-theoretically) secure [Sha79] since there is no strategy an adversary with $t-1$ shares can use to reconstruct the (under-determined) polynomial $f(x)$, beyond guessing the secret (or share). Security of Shamir with respect to Definition 2.1 follows immediately; the following theorem is stated without proof.

Theorem 2.1. *The secret sharing scheme* Shamir_q *is secure.*

2.2 Sharing Multiple Secrets

Some use cases require the storage and retrieval of multiple secrets (see e.g. [BDSDC+94, JMO94]). Protocol Shamir_q invokes q independent instantiations of Shamir to distribute

shares for q secrets among n friends, such that t-out-of-n friends must cooperate to recover the secrets:

- $\underline{\mathsf{Shamir}_q}$:
  - Select a set of q secrets, $\{\bar{y}_i\}_{i\in[q]}$, with $\bar{y}_i \in \mathbb{F}_p^*$, and a set of n friends.
  - Draw $q \times (t-1)$ field elements uniformly at random:

$$\alpha_{(c,i)} \xleftarrow{R} \mathbb{F}_p^*, \quad \text{for} \quad i \in [q] \quad \text{and} \quad c \in [t-1], \tag{1}$$

    and define q polynomials:

$$f_i(x) \;=\; \bar{y}_i + \sum_{c=1}^{t-1} \alpha_{(c,i)}\, x^c \quad \text{for} \quad i \in [q]. \tag{2}$$

  - Evaluate the polynomials $f_i(x)$ at the evaluation points $\{x_a\}_{a\in[n]}$. Label the evaluations as $y_{a,i} = f_i(x_a)$ and distribute a share sh_a to each of the n friends, where

$$\mathsf{sh}_a = (y_{a,1}, \ldots, y_{a,q}). \tag{3}$$

  - To reconstruct the secrets $\{\bar{y}_i\}_{i\in[q]}$:
    - Request the shares sh_b from t friends (described by the index $b \in \mathcal{T}_* \subset [n]$, where $|\mathcal{T}_*| = t$).
    - Reconstruct the polynomials $f_i(x)$ via interpolation, and evaluate the secret values as $\bar{y}_i = f_i(0)$, $\forall i \in [q]$.

The following generalisation of Definition 2.1 is adopted:

Definition 2.2. *q-Secret Sharing Security.* A t-out-of-n ($t \leq n$) q-secret sharing protocol $\mathcal{P}_q$ is said to be *secure* if any probabilistic polynomial time adversary $\mathcal{A}$ that obtains a full set of $t-1$ shares $\{y_{c,i}\}_{c\in\mathcal{T},i\in[q]}$, with $|\mathcal{T}| = t-1$, has a negligible chance of outputting the corresponding secrets $\{\bar{y}_i\}_{i\in[q]}$. Namely:

$$\Pr\left[\begin{array}{l}\mathbb{F}_p \leftarrow \mathcal{G}(1^\lambda);\ (\{\bar{y}_i\}_{i\in[q]}, \{y_{a,i}\}_{a\in[t],i\in[q]}) \leftarrow \mathcal{P}_q(\mathsf{input});\ \bar{t} \xleftarrow{R} [t];\ \mathcal{T} \equiv [t]\backslash\{\bar{t}\};\\ \qquad\qquad \{\bar{z}_i\}_{i\in[q]} \leftarrow \mathcal{A}(\{y_{c,i}\}_{c\in\mathcal{T},i\in[q]}) :\ \bar{z}_i = \bar{y}_i, \forall i \in [q]\end{array}\right] = \mathsf{negl}(\lambda),$$

where input denotes (optional) input which may include the secrets $\{\bar{y}_i\}_{i\in[q]}$.

This definition requires that PPT adversaries which acquire $t-1$ shares have a negligible probability of determining the full set of corresponding secrets. For protocols such as Shamir_q, the input data includes the secrets $\{\bar{y}_i\}_{i\in[q]}$. The following theorem follows immediately from the information-theoretic security property of Shamir (and is therefore stated without proof):

Theorem 2.2. *The q-secret sharing scheme Shamir_q is secure.*

In protocol Shamir, the polynomial $f(x)$ is drawn uniformly at random from the set of degree $t-1$ polynomials with evaluation $f(0) = \bar{y}$. In the q-secret sharing scheme Shamir_q, the set of q secrets $\{\bar{y}_i\}_{i\in[q]}$ are encoded using q distinct polynomials, $\{f_i(x)\}_{i\in[q]}$, such that each polynomial $f_i(x)$ is independently drawn uniformly at random from the set of degree $t-1$ polynomials with evaluation $f_i(0) = \bar{y}_i$. The security of Shamir therefore trivially carries over independently for each secret $\bar{y}_j \in \{\bar{y}_i\}_{i\in[q]}$ inputted into algorithm Shamir_q. Consequently one may define a stronger security property:

Definition 2.3. *Strong q-Secret Sharing Security.* A t-out-of-n ($t \leq n$) q-secret sharing protocol $\mathcal{P}_q$ is said to be *strongly secure* if:

1. Any probabilistic polynomial time adversary $\mathcal{A}$ that obtains a full set of $t-1$ shares $\{y_{c,i}\}_{c\in\mathcal{T},i\in[q]}$, with $|\mathcal{T}| = t-1$, has a negligible chance of outputting one of the corresponding secrets $\bar{y}_j \in \{\bar{y}_i\}_{i\in[q]}$. Namely:

$$\Pr\left[\begin{array}{c}\mathbb{F}_p \leftarrow \mathcal{G}(1^\lambda);\ (\{\bar{y}_i\}_{i\in[q]}, \{y_{a,i}\}_{a\in[t],i\in[q]}) \leftarrow \mathcal{P}_q(\mathsf{input});\ \bar{t} \xleftarrow{R} [t];\ \mathcal{T} \equiv [t]\backslash\{\bar{t}\};\\ \bar{z}_j \leftarrow \mathcal{A}(\{y_{c,i}\}_{c\in\mathcal{T},i\in[q]}) :\ \bar{z}_j \in \{\bar{y}_i\}_{i\in[q]}\end{array}\right] = \mathsf{negl}(\lambda),$$

   where input denotes (optional) input data which may include the secrets $\{\bar{y}_i\}_{i\in[q]}$.

2. A probabilistic polynomial time adversary that successfully determines one secret, say $\bar{y}_j \in \{\bar{y}_i\}_{i\in[q]}$, continues to have a negligible probability of determining the other secrets $\bar{y}_{j'\neq j} \in \{\bar{y}_i\}_{i\in[q]}$.

This definition requires PPT adversaries which acquire $t-1$ shares to have a negligible probability of determining *any* of the corresponding secrets, *and*, if an adversary does obtain one secret, they gain at most a negligible advantage in determining the other secrets. Strong q-secret sharing security therefore implies q-secret sharing security. Shamir_q satisfies this stronger security definition:

Theorem 2.3. *The q-secret sharing scheme Shamir_q is strongly secure*

Proof. Protocol Shamir_q invokes q independent instantiations of protocol Shamir, which is information-theoretically secure. Consequently the theorem follows immediately. ■

2.3 Sharing Multiple (Power) Secrets

Consider the scenario in which a user Alice wants to store and recover a set of q secrets which have the form $\{\bar{y}, \bar{y}^2, \ldots, \bar{y}^q\} = \{\bar{y}^i\}_{i\in[q]}$. Such a set is referred to as a *power secret* because the secrets are related as powers of a single secret $\bar{y}$. Clearly, in some use cases, Alice may leverage the fact that the secrets are related and need only store and recover the single secret $\bar{y}$, from which $\bar{y}^i$ may be calculated, for all $i \in [q]\backslash\{1\}$. In such cases, Alice can use Shamir to distribute and recover $\bar{y}$. However, other use cases may not permit recovery of the set $\{\bar{y}^i\}$ from the single value $\bar{y}$. For example, in threshold signature schemes, the secrets may

correspond to private keys, such that secret recovery only occurs "in the exponent" without direct knowledge of $\bar{y}$. If one needs to store and recover a power secret $\{\bar{y}^i\}_{i\in[q]}$, the q-secret sharing scheme $\mathsf{pwrSecret}_q$ may be appropriate:

- $\underline{\mathsf{pwrSecret}_q}$:
  - Select q secrets $\{\bar{y}^i\}_{i\in[q]}$, which are powers of a single secret $\bar{y} \in \mathbb{F}_p^*$.
  - Execute protocol Shamir_q using the secrets $\{\bar{y}^i\}_{i\in[q]}$.

Protocol $\mathsf{pwrSecret}_q$ is a simple implementation of Shamir_q with a power secret (i.e. valid secrets are ordered powers of the first secret). Strong q-secret sharing security of $\mathsf{pwrSecret}_q$ therefore follows immediately from the strong q-secret sharing security of Shamir_q.

3 PolyShamir Secret Sharing

This section discusses a variant of Shamir which may be generalised to obtain variant multi-secret sharing protocols.

3.1 Sharing PolyShamir Secrets

A user (Alice) of protocol Shamir specifies a secret $\bar{y}$ and uses $t-1$ randomly drawn field elements to define a degree $t-1$ polynomial $f(x)$, with evaluation $f(0) = \bar{y}$. Evaluations of $f(x)$ at arbitrary nonzero points are uniformly random as $f(x)$ is drawn uniformly at random from the set of degree $t-1$ polynomials with fixed evaluation $f(0) = \bar{y}$. There exists a variant of Shamir that similarly draws uniformly random field elements to construct a polynomial, with random evaluations, that encodes a secret. Imagine that Alice wishes to generate a secret value $\bar{y}$ and share the secret among t of her friends, such that all t shares are required to reconstruct $\bar{y}$. Such scenarios arise, e.g., if the shares are used in signature schemes that require all friends/authorities to sign a message in order to generate a valid master signature.[2] Furthermore, assume Alice is indifferent to the particular value of the secret but wants surety that she may reliably generate a secret and recover its value when needed. For example, the secret may be a cryptographic key whose particular value is irrelevant (i.e. random) but whose secrecy is paramount. The secret sharing scheme polyShamir is suitable for such cases:

- $\underline{\mathsf{polyShamir}}$
  - Randomly draw t field elements $y_a \xleftarrow{R} \mathbb{F}_p^*$, for $a \in [t]$.
  - Using interpolation, construct a polynomial $f(x)$ by defining $f(x_a) = y_a$, for specified evaluation points $\{x_a\}_{a\in[t]}$.

[2] In t-out-of-t secret sharing, provided at least one friend is honest, the secret cannot be reconstructed by a set of adversaries. For t-out-of-n secret sharing, honesty of a single friend is insufficient to preclude collusion - one requires at least $n-t+1$ honest friends.

- Identify the (random) secret value with the evaluation $f(0)$, namely $\bar{y} \leftarrow f(0)$.
- Distribute a single share $\mathsf{sh}_a = y_a$ to each of the t friends.
- To recover the secret $\bar{y}$:
  - Retrieve the t shares from the friends, reconstruct the polynomial $f(x)$ via interpolation, and perform the evaluation $\bar{y} = f(0)$.

Security of protocol polyShamir follows immediately:

Theorem 3.1. *The t-out-of-t secret sharing scheme* polyShamir *is secure.*

Proof. Protocol polyShamir is information-theoretically secure to any adversary with (at most) $t-1$ shares and is therefore secure, according to Definition 2.1. ■

Protocol polyShamir is a variant of Shamir that is appropriate when the secret value need not be specified prior to protocol execution. The label polyShamir reflects the fact that users *first* construct a random polynomial, *then* deduce their secret value. Security of polyShamir can also be proven via reduction to Shamir, using a mapping from any set of $t-1$ randomly drawn shares $\{y_c\}_{c\in S}$, for secret $\bar{y}$ in polyShamir, to the same set of shares for the same secret $\bar{y}$ in scheme Shamir. An adversary $\mathcal{A}_{\mathsf{polyShamir}}$ with a non-negligible success probability of determining a polyShamir secret could be leveraged to obtain a non-negligible probability of breaking the security of Shamir, contradicting Theorem 2.1.

Note that, in the above description of polyShamir, the user drew random shares and distributed them to friends. However, if the friends are honest, they may directly draw their own shares. Also, polyShamir can be generalised to achieve t-out-of-n secret sharing (see below) though appears more naturally suited for t-out-of-t secret sharing.

3.2 Sharing Multiple PolyShamir Secrets

In scheme polyShamir, the interpolated polynomial is defined directly by using randomly drawn shares as polynomial evaluations. There are multiple ways to generalise this scheme and construct q-secret sharing schemes. Three generalisations are discussed in what follows (two below; one in the subsequent subsection). The first generalisation is a minimal extension of polyShamir in which Alice defines q secrets, using q evaluations of the single randomly drawn polynomial $f(x)$. Specifically, Alice chooses q points x_i, such that $x_i \neq x_a$ for all $i \in [q]$ and all $a \in [t]$, and defines her secrets as the evaluations $\bar{y}_i = f(x_i)$. Recovery of all t shares allows Alice to reconstruct all q secrets using the single polynomial $f(x)$. Referring to this scheme as $\overline{\mathsf{polyShamir}}_q$, one has the following theorem:

Theorem 3.2. *The q-secret sharing scheme* $\overline{\mathsf{polyShamir}}_q$ *is secure.*

Proof. $\overline{\mathsf{polyShamir}}_q$ is information theoretically secure as an adversary with $t-1$ shares has insufficient information to reconstruct the polynomial $f(x)$. Thus, the probability of outputting the set of full secrets is negligible provided p is sufficiently large. ■

Note that knowledge of one $\overline{\mathsf{polyShamir}_q}$ secret is sufficient to determine the polynomial $f(x)$ and obtain all other secrets (if $t-1$ shares are already known). Consequently $\overline{\mathsf{polyShamir}_q}$ is secure but not strongly secure. Users of $\overline{\mathsf{polyShamir}_q}$ must tolerate all q secrets being encoded on a single polynomial, whose form is not known prior to initiating the protocol. Note that an analogous q-secret generalisation of Shamir, that encodes multiple secrets in a single polynomial, is not possible as a Shamir secret must be specified prior to protocol execution.

A second q-secret generalisation of polyShamir is obtained by following the recipe of Section 2.2, where independent instantiations of Shamir were used to construct protocol Shamir_q. Using independent instantiations of the single-secret scheme polyShamir produces the q-secret sharing scheme $\underline{\mathsf{polyShamir}_q}$, defined as follows:

- $\underline{\mathsf{polyShamir}_q}$:
  - Randomly draw $q \times t$ field elements, $y_{a,i} \xleftarrow{R} F_p^*$, for $a \in [t]$ and $i \in [q]$.
  - Define q polynomials via interpolation, using $f_i(x_a) = y_{a,i}$, for evaluation points $\{x_a\}_{a\in[k]}$.
  - Identify the q secrets with the evaluations at the origin, namely $\bar{y}_i \leftarrow f_i(0)$.
  - Distribute a shares containing q elements $y_{a,i}$, for fixed a, to each friend:

$$\mathsf{sh}_a = (y_{a,1}, \ldots, y_{a,q}). \tag{4}$$

  - To recover the secrets $\{\bar{y}_i\}_{i\in[q]}$:
    - Request the shares from all friends, construct the q polynomials $f_i(x)$ via interpolation using $f_i(x_a) = y_{a,i}$, and obtain the q secrets as $\bar{y}_i = f_i(0)$.

Security of $\underline{\mathsf{polyShamir}_q}$ follows immediately as the shares are uniformly distributed at random (similar to polyShamir).

Theorem 3.3. *The q-secret sharing scheme $\underline{\mathsf{polyShamir}_q}$ is strongly secure.*

Proof. Each secret generated by an execution of $\underline{\mathsf{polyShamir}_q}$ is generated by an independent execution of polyShamir, which is information-theoretically secure (Theorem 3.1). Consequently knowledge of one secret provides no additional information about the other secrets and protocol $\underline{\mathsf{polyShamir}_q}$ is strongly secure. ■

3.3 Sharing Multiple PolyShamir Secrets Using Power Shares

The method of directly drawing random shares, employed in polyShamir, avails an additional q-secret generalisation that is not available when one instead randomly draws the polynomial coefficients and evaluates the shares (as in Shamir). Protocol $\mathsf{pwrShares}_q$, an alternative q-secret generalisation of polyShamir, is defined as follows:

- $\underline{\mathsf{pwrShares}_q}$:
  - Randomly draw t elements $y_a \xleftarrow{R} \mathbb{F}_p^*$, for $a \in [t]$.
  - Via interpolation, define q polynomials $f_i(x)$, using the evaluations $f_i(x_a) = y_a^i$, for all $a \in [t]$, and all $i \in [q]$.
  - Identify the q secrets $\{\bar{y}_i\}_{i\in[q]}$ as the evaluations $\bar{y}_i \leftarrow f_i(0)$, $i \in [q]$.
  - Distribute one (partial) share, $\mathsf{sk}_a = y_a$, to each of the t friends.[3]
  - To recover the secrets $\{\bar{y}_i\}_{i\in[q]}$:
    - Retrieve the shares $\{y_a\}_{a\in[k]}$, reconstruct the polynomials $f_i(x)$ using interpolation and evaluate the secrets as $\bar{y}_i = f_i(0)$.

Protocol $\mathsf{pwrShares}_q$ is a q-secret sharing scheme with *power shares*, meaning the full shares required to define all q polynomials $f_i(x)$ via interpolation have the form:

$$\overline{\mathsf{sh}}_a = (y_a, y_a^2, \ldots, y_a^q), \tag{5}$$

for each friend $a \in [t]$. Users need not distribute the full shares $\overline{\mathsf{sh}}_a$, as these may be derived from the (partial) shares $\mathsf{sh}_a = y_a$. Consequently users need only generate and distribute the single field element y_a (per friend) to permit secret storage and reconstruction. Thus, relative to Shamir_q and $\mathsf{polyShamir}_q$, the number of randomly drawn and distributed field elements used as shares is reduced from $t \times q$ independent elements to t independent elements, making key generation and distribution less complex in $\mathsf{pwrShares}_q$. In particular, the (partial) share length remains at $\mathcal{O}(1)$ elements, independent of the number of secrets q. Similar to $\mathsf{polyShamir}$ and $\mathsf{polyShamir}_q$, $\mathsf{pwrShares}_q$ also requires users to first construct random polynomials and then derive their secrets by evaluating the polynomials.

Theorem 3.4. *The q-secret sharing scheme $\mathsf{pwrShares}_q$ is secure*

Proof. Security of $\mathsf{pwrShares}_q$ is proven by contradiction, using a reduction to the single-secret sharing scheme $\mathsf{polyShamir}$. Assume that there exists a probabilistic polynomial time adversary $\mathcal{A}_{\mathsf{pwrShares}_q}$, which has a non-negligible probability of successfully determining all q secrets produced by an execution of $\mathsf{pwrShares}_q$. Let challenger $\mathcal{C}$ execute protocol $\mathsf{polyShamir}$ and provide the probabilistic polynomial time adversary $\mathcal{A}_{\mathsf{polyShamir}}$ with $t-1$ shares, $\{y_c\}_{c\in\mathcal{T}}$, where $|\mathcal{T}| = t-1$, corresponding to secret $\bar{y}$. Algorithm $\mathcal{A}_{\mathsf{polyShamir}}$ is tasked with outputting a secret $\bar{y}_\mathcal{A}$, and wins the challenge if $\bar{y}_\mathcal{A} = \bar{y}$. Upon receipt of the shares $\{y_c\}_{c\in\mathcal{T}}$, $\mathcal{A}_{\mathsf{polyShamir}}$ passes them to $\mathcal{A}_{\mathsf{pwrShares}_q}$, which treats the shares as $\mathsf{pwrShares}_q$ shares. Algorithm $\mathcal{A}_{\mathsf{pwrShares}_q}$ outputs the secrets $\{\bar{z}_i\}_{i\in[q]}$, which, by assumption, have a non-negligible probability of being the actual secrets $\{\bar{y}_i\}_{i\in[q]}$, corresponding to the (partial) shares $\{y_c\}_{c\in\mathcal{T}}$, such that $\bar{y}_1 = \bar{y}$. Upon receipt of the secrets $\{\bar{z}_i\}_{i\in[q]}$, adversary $\mathcal{A}_{\mathsf{polyShamir}}$ sets $\bar{y}_\mathcal{A} = \bar{z}_1$, and outputs $\bar{y}_\mathcal{A}$ to $\mathcal{C}$. With non-negligible probability, $\bar{y} = \bar{y}_\mathcal{A}$ and $\mathcal{A}_{\mathsf{polyShamir}}$ wins the challenge. This result contradicts Theorem 3.1, which asserts that all adversaries against $\mathsf{polyShamir}$ have a negligible probability of success. Thus, probabilistic polynomial time adversary $\mathcal{A}_{\mathsf{pwrShares}_q}$ cannot exist and $\mathsf{pwrShares}_q$ is secure. ∎

[3] If the friends are honest, the scheme also works if the friends directly draw their own share, $\mathsf{sh}_a = y_a \xleftarrow{R} \mathbb{F}_p$.

The above proof avails the following intuition. Consider the execution of polyShamir with secret $\bar{y}$ encoded as the constant in polynomial $f(x)$, which is constructed via interpolation using the shares $\{y_c\}_{c\in[t-1]} \cup \{y_t\}$, where $f(x_c) = y_c$ and $f(x_t) = y_t$. The shares are drawn uniformly at random $y_c, y_t \xleftarrow{R} \mathbb{F}_p$, meaning the polynomial is selected uniformly at random from the set of degree $t-1$ polynomials. Using this set of valid polyShamir parameters, which contains t shares, one can always construct the following set of $q \times t$ shares:

$$\{y_c^i\}_{c\in[t-1]} \cup \{y_t^i\},\ \forall i \in [q]. \tag{6}$$

These shares define q polynomials $f_i(x)$, via the evaluations $f_i(x_c) = y_c^i$ and $f_i(x_t) = y_t^i$. Together with the evaluations $\bar{y}_i = f_i(0)$, these parameters form a valid set under $\mathsf{pwrShares}_q$, such that:

$$\bar{y} = \bar{y}_1 \leftarrow f_1(0) = f(0). \tag{7}$$

Thus, any set of valid parameters for polyShamir *automatically* defines a corresponding set of valid parameters for $\mathsf{pwrShares}_q$. More precisely, there is a one-to-one mapping from a valid polyShamir share (y_a), to a related full $\mathsf{pwrShares}_q$ share, with action

$$y_a \rightarrow (y_a, y_a^2, \ldots, y_a^q). \tag{8}$$

Consequently if the q-secret scheme $\mathsf{pwrShares}_q$ were insecure, one could always break the security of polyShamir by mapping a given instance of valid parameters for polyShamir to the corresponding set of valid parameters for $\mathsf{pwrShares}_q$, and leveraging the security vulnerability of $\mathsf{pwrShares}_q$ to break polyShamir. Hence, security of the multi-secret scheme $\mathsf{pwrShares}_q$ is expected to be related to the security of the single-secret sharing scheme polyShamir, as explicitly demonstrated above. Similar comments hold with respect to Shamir - by mapping an instance of Shamir to an instance of polyShamir, one can always derive a valid set of parameters for an execution of $\mathsf{polyShamir}_q$ from a set of valid parameters for Shamir, such that the $i = 1$ polynomial in $\mathsf{polyShamir}_q$ matches the single polynomial used in Shamir. Security vulnerabilities that reveal all q secrets in $\mathsf{pwrShares}_q$ could therefore be used to break Shamir.

Note that $\mathsf{pwrShares}_q$ is secure but not strongly secure - knowledge of a single secret allows an adversary to determine the value of the remaining share for that secret, which *may*, in turn, allow the adversary to determine the remaining share for other secrets.[4] The different security properties of $\mathsf{pwrShares}_q$ and the multi-secret sharing schemes $\mathsf{polyShamir}_q$ and Shamir_q reflect the fact that the q polynomials $f_i(x)$ are related in $\mathsf{pwrShares}_q$. To elaborate on this point, consider an adversary $\mathcal{A}_{\mathsf{pwrShares}_q}$, acting against $\mathsf{pwrShares}_q$, which obtains a complete set of (partial) shares $\{y_a\}_{a\in[t]}$. Such an adversary can generate all polynomials $f_i(x)$ and reconstruct all secrets $\bar{y}_i$, for $i \in [q]$. In contrast, an adversary $\mathcal{A}_{\mathsf{Shamir}_q}$, acting against Shamir_q, which obtains a complete set of shares $\{y_{a,j}\}_{a\in[t]}$, for some fixed value of $j \in [q]$, can generate the polynomial $f_j(x)$ and obtain the secret $\bar{y}_j = f_j(0)$, but cannot generate the polynomials $f_i(x)$ for $i \neq j$, so the secrets $\bar{y}_{i\neq j}$ remain secure. Both adversaries have access to t field elements, yet $\mathcal{A}_{\mathsf{pwrShares}_q}$ achieves a full break of $\mathsf{pwrShares}_q$

[4] This point is discussed in Section 4.

whereas $\mathcal{A}_{\mathsf{Shamir}_q}$ only achieves a partial break of Shamir_q. To achieve a full break of Shamir_q, adversary $\mathcal{A}_{\mathsf{Shamir}_q}$ must instead obtain all $t \times q$ shares $\{y_{a,i}\}_{i\in[q],a\in[t]}$. Similar comments hold for $\mathsf{polyShamir}_q$. Thus, in some sense the security of $\mathsf{pwrShares}_q$ is more similar to that of Shamir, for which a set of t shares is sufficient to achieve a full break (and reconstruct the single secret), than it is to Shamir_q and $\mathsf{polyShamir}_q$, where $t \times q$ shares are needed to achieve a full break. Speaking informally, $\mathsf{pwrShares}_q$ is a type of multi-secret sharing scheme with security properties resembling a single-secret sharing scheme, as a partial break *may* expose all secrets.[5]

It may appear strange that, to securely share q secrets among t friends using $\mathsf{polyShamir}_q$, one must generate and distribute $t \times q$ secret field elements, whereas to share q secrets among t friends using $\mathsf{pwrShares}_q$, one need only generate and distribute t random elements. To better understand why $\mathsf{pwrShares}_q$ remains secure, it is useful to consider how an adversary may attack the scheme. Assume that an adversary knows $t-1$ shares $\{y_c\}_{c\in[t-1]}$ and the evaluation points $\{x_c\}_{c\in[t-1]} \cup \{x_t\}$, but does not know the final share y_t. The adversary may construct the following equations

$$\begin{aligned} f_i(x_c) &= \bar{y}_i + \sum_{c'=1}^{t-1} \alpha_{(c',i)}\, x_c^{c'} = y_c^i \\ f_i(x_t) &= \bar{y}_i + \sum_{c'=1}^{t-1} \alpha_{(c',i)}\, x_t^{c'} = Y_t^i, \end{aligned} \tag{9}$$

where $Y_t = y_t$ denotes the unknown share, and both the coefficients $\alpha_{(c',i)}$ and the secrets $\bar{y}_i$ are unknown. This system contains $t \times q$ linear equations in the $1 + t \times q$ unknowns Y_t, $\alpha_{(c',i)}$ and $\bar{y}_i$. Consequently the system of equations cannot be solved and, in fact, the final share Y_t is randomly distributed. Thus, although the full set of shares required to reconstruct all polynomials $f_i(x)$, namely $\overline{\mathsf{sh}}_a = (y_a, y_a^2, \ldots, y_a^q)$, can be calculated with knowledge of the (partial) shares $\mathsf{sh}_a = y_a$, the underlying system of equations remains under-determined unless an adversary knows *all* partial shares, so the system cannot be solved. Accordingly, $\mathsf{pwrShares}_q$ is secure unless an adversary obtains either all t secret shares or $t-1$ secret shares and one or more secrets that allow the system of equations (9) to be solved.

Finally, note that it is not possible to generalise $\mathsf{polyShamir}$ to a multi-secret sharing scheme with power secrets. Protocol $\mathsf{polyShamir}$ generates random secrets during execution that will not, in general, be related as powers of a single secret.

3.4 Generalising to t-out-of-n Secret Sharing Schemes

The t-out-of-t secret sharing schemes $\mathsf{polyShamir}$, $\mathsf{polyShamir}_q$ and $\mathsf{pwrShares}_q$ may be generalised to t-out-of-n secret sharing schemes. In scheme $\mathsf{polyShamir}$, users construct polynomials by drawing t shares at random, $y_a \xleftarrow{R} \mathbb{F}_p$, and identifying the shares with polynomial

[5] This comment is unsurprising - $\mathsf{pwrShares}_q$ leverages the same amount of randomness as e.g. Shamir, and consequently has similar security properties.

evaluations $f(x_a) = y_a$, for all $a \in [t]$.[6] Protocol polyShamir may be generalised to a t-out-of-n secret sharing scheme by evaluating $f(x)$ at an additional $n - t$ points, $y_{a'} = f(x_{a'})$, and defining the secret keys $\mathsf{sh}_{a'} = y_{a'}$ for $a' \in \{t+1, t+2, \ldots, n\}$. This same process can be repeated for the t-out-of-t q-secret sharing schemes $\mathsf{polyShamir}_q$ and $\mathsf{pwrShares}_q$ by evaluating the multiple polynomials $f_i(x)$ at an additional $n - t$ points $x_{a'}$ and using the evaluations $y_{a',i} = f_i(x_{a'})$ to define the shares $\mathsf{sh}_{a'} = (y_{a',1}, \ldots, y_{a',q})$ for the additional $n - t$ friends. A user may now reconstruct the secrets by retrieving shares from t of the n friends and using interpolation.

In these generalisations to t-out-of-n threshold secret sharing schemes, the shares for the first t friends would be generated by drawing random field elements and assigning these directly to shares, but the shares for the additional $n-t$ friends are calculated as polynomial evaluations for polynomials that are drawn randomly. Thus, some friends would receive a short share $\mathsf{sh}_a = y_a$ (for $a \in [t]$), whereas other friends would receive longer shares $\mathsf{sh}_{a'} = (y_{a',1}, \ldots, y_{a',q})$ (for $a' \in \{t+1, t+2, \ldots, n\}$). This asymmetry among friends may be accommodated in the protocol by, e.g., distributing full shares $\mathsf{sh}_a \to (y_a, y_a^2, \ldots, y_a^q)$ to the first t friends (so all shares contain q elements) or handling long versus short shares differently during interpolation. Nonetheless, this asymmetry is not particularly appealing and one may reasonably argue that secret sharing schemes with power shares are better suited to t-out-of-t scenarios. The t-out-of-n implementations are not discussed further here.

4 K-Private Secret Sharing

4.1 K-Privacy

To further discuss the differences between strongly secure q-secret sharing protocols such as Shamir_q and secure q-secret sharing protocols such as $\mathsf{pwrShares}_q$, it is useful to introduce the notion of secret sharing K-privacy.

Definition 4.1. *Secret Sharing K-Privacy.* A t-out-of-n ($t \leq n$) secret sharing protocol $\mathcal{P}$, with secret space $\mathbb{F}_p^*$, is said to be K-*private* if any probabilistic polynomial time adversary $\mathcal{A}$ that obtains shares $\{y_c\}_{c \in \mathcal{T}}$ (with $|\mathcal{T}| = t - 1$), corresponding to secret $\bar{y}$, may leverage the shares to deduce that there exists a set $\Delta_{\mathcal{T}} \subseteq F_p^*$ of cardinality $|\Delta_{\mathcal{T}}| = K$, such that $\bar{y} \in \Delta_{\mathcal{T}}$.

This definition captures the idea that, upon receipt of $t - 1$ shares, the adversary may leverage the shares and auxiliary information about the protocol to determine that the secret $\bar{y}$ must reside within a set of candidate secrets $\Delta_{\mathcal{T}}$, of cardinality $K \leq |F_p^*|$. K-privacy is related to the notion of k-anonymity [SS98], whereby anonymity of an individual's data is provided with respect to subsets of size k within a larger data set, but the anonymising subsets for distinct individuals may differ. In the present context, the idea is that even if the

[6] In the q-secret sharing schemes $\mathsf{polyShamir}_q$ and $\mathsf{pwrShares}_q$, a similar process is employed for multiple polynomials.

protocol permits an adversary to deduce that the secret belongs to some subset $\Delta_{\mathcal{T}} \subseteq F_p^*$, the protocol ensures that, for any such PPT adversary with $t-1$ shares, the corresponding subset contains at least K candidate values for the secret which cannot be excluded *a priori*. The particular subset that provides K-privacy for a given adversary may vary among adversaries, allowing distinct adversaries to acquire different information about the secret. Nonetheless, a K-private protocol ensures that the secret remains K-private for all adversaries. Note that a secret sharing scheme with K-privacy may or may not be secure, depending on the properties of the scheme.

To exemplify K-privacy, consider the following single-secret sharing variant of polyShamir, referred to as NpolyShamir:

- NpolyShamir
  - Randomly draw t field elements $y_a \xleftarrow{R} \mathbb{F}_p^*$, for $a \in [t]$.
  - Construct a polynomial $f(x)$, defined by $f(x_a) = y_a^N$, for specified evaluation points $\{x_a\}_{a\in[t]}$.
  - Identify the (random) secret value with the evaluation $f(0)$, namely $\bar{y} \leftarrow f(0)$.
  - Distribute a single share $\mathsf{sh}_a = y_a^N$ to each of the t friends.
  - To recover the secret $\bar{y}$:
    - Retrieve the t shares from the friends, reconstruct the polynomial $f(x)$ via interpolation, and perform the evaluation $\bar{y} = f(0)$.

Protocol NpolyShamir is identical to polyShamir except that the shares $\mathsf{sh}_a = y_a^N$ are raised to the power of N. Consequently all shares reside in the set of N-atic residues mod p, namely $\mathsf{sh}_a \in \Delta_{N,p}$ for all $a \in [t]$. Consider an adversary $\mathcal{A}$ that acquires $t-1$ shares $\{y_c^N\}_{c\in\mathcal{T}}$. Using knowledge of the protocol, the adversary may deduce that the missing share must also be an N-atic reside mod p, i.e. $y_{\bar{t}}^N \in \Delta_{N,p}$. Thus, for each candidate value of the share $y_{\bar{t}}^N$, there exists a unique candidate value for the secret $\bar{z} \in \mathbb{F}_p^*$, so the set of candidate valid secrets $\bar{\mathcal{Z}}$, where $\bar{z} \in \bar{\mathcal{Z}}$, has cardinality $|\bar{\mathcal{Z}}| = |\Delta_{N,p}|$. Moreover, a different adversary $\mathcal{A}'$ that acquires a non-identical set of $t-1$ shares $\{y_c^N\}_{c\in\mathcal{T}'}$ can also deduce that the secret must reside within a set of candidate valid secrets $\bar{\mathcal{Z}}'$, with $\bar{z}' \in \bar{\mathcal{Z}}'$, where $|\bar{\mathcal{Z}}'| = |\Delta_{N,p}|$. In general, one may have $\bar{\mathcal{Z}}' \neq \bar{\mathcal{Z}}$ but with $\bar{\mathcal{Z}}' \cap \bar{\mathcal{Z}} \neq \emptyset$. The full secret space $\tilde{\mathcal{Y}}$ is a subset of $\mathbb{F}_p$, defined as the set of values $\tilde{y} \in \mathbb{F}_p^*$, such that there exists a polynomial $f(x)$ of degree $t-1$ with evaluations $f(x_a) = y_a^N$ and $f(0) = \tilde{y}$. All adversaries agree on the set $\tilde{\mathcal{Y}}$ but distinct adversaries, which obtain different sets of shares, deduce that the secret must reside within different sets of candidate secrets such as $\bar{\mathcal{Z}} \subseteq \tilde{\mathcal{Y}}$ and $\bar{\mathcal{Z}}' \subseteq \tilde{\mathcal{Y}}$.

Combining the above information demonstrates that protocol NpolyShamir is $|\Delta_{N,p}|$-private. For the particular case with $N = 1$, one has 1polyShamir = polyShamir and protocol 1polyShamir is both $|\Delta_{1,p}|$-private and secure. However, as $|\Delta_{1,p}| = |\mathbb{F}_p^*|$, the set of candidate secrets deduced by any adversary with $t-1$ shares is equal to the full secret space and the notion of K-privacy trivially holds. However, for $N = 2$ the protocol 2polyShamir is $|\Delta_{2,p}|$-private, so adversaries with distinct sets of $t-1$ shares deduce that there exist $|\Delta_{2,p}|$

corresponding candidate values for the secret, but the particular set $\bar{\mathcal{Z}}$ of candidate secrets, of size $|\bar{\mathcal{Z}}| = |\Delta_{N,p}|$, can differ among adversaries. The point here is that, with knowledge of the protocol, an adversary against 2polyShamir with $t-1$ shares can deduce that there are $|\Delta_{N,p}|$ candidate secrets which are consistent with their shares, and $|\Delta_{N,p}|$ may be less than the number of candidate secrets available before obtaining the shares (i.e. the full secret space). Protocol 2polyShamir is therefore $|\Delta_{2,p}|$-private and, for sufficiently large p, secure. Note, however, that K-privacy is non-trivial in this instance - in general, the set of $|\Delta_{N,p}|$ candidate secrets deduced by an adversary with $t-1$ shares can be a proper subset of the secret space, so an execution of NpolyShamir ensures privacy with respect to a set of candidate secrets of size $|\Delta_{N,p}|$, not the full secret space. As an example at the other extreme to the $N = 2$ case, consider the case with $N = p-1$, for which one has $|\Delta_{p-1,p}| = 1$ and the secret is uniquely determined. In this case, the scheme is 1-private and (trivially) insecure.

For present purposes, the following generalised version of K-privacy, applicable for q-secret sharing schemes, is of interest:

Definition 4.2. *q-Secret Sharing K-Privacy.* A t-out-of-n ($t \leq n$) q-secret sharing protocol $\mathcal{P}_q$, with secret space $\mathbb{F}_p^*$, is said to be *K-private* if any probabilistic polynomial time adversary $\mathcal{A}$ that obtains shares $\{y_{c,i}\}_{c\in\mathcal{T},i\in[q]}$ (with $|\mathcal{T}| = t-1$), corresponding to secrets $\{\bar{y}_i\}_{i\in[q]}$, may leverage the shares to deduce that there exists sets $\Delta_{\mathcal{T},i} \subseteq F_p^*$, such that $\bar{y}_i \in \Delta_{\mathcal{T},i}$ and $|\Delta_{\mathcal{T},i}| = K_i \leq |F_p^*|$ for all $i \in [q]$. Moreover, there exists at least one value $j \in [q]$ such that $K_j < |F_p^*|$. The i-th secret of a K-private q-secret sharing protocol is said to be *K_i-private.*

Here, the idea is that an adversary may deduce that each of the q secrets belongs to a particular subset $\Delta_{\mathcal{T},i} \subseteq F_p^*$, with $|\Delta_{\mathcal{T},i}| = K_i$, but cannot uniquely determine which of these K_i candidate values for the i-th secret is the actual secret. Different adversaries may determine that the i-th secret resides in distinct subsets $\Delta_{\mathcal{T},i}$, depending on which shares they receive, but for all adversaries there exist K_i candidate values for the i-th secret which the adversary cannot exclude *a priori*. For a K-private q-secret sharing scheme, the size of the sets providing privacy varies among secrets, such that the i-th secret is K_i-private; i.e. privacy of the i-th secret is ensured with respect to a set of size K_i. The requirement that there exists at least one value $j \in [q]$, with $K_j < |F_p^*|$, ensures that K-privacy is non-trivial for q-secret sharing (i.e. there is always at least one secret for which privacy is provided with respect to a subset of the secret space, rather than the full secret space). For q-secret sharing schemes, the label of "K-private" refers to the fact that adversaries may deduce that the secrets belong to subsets of the secret space, with the particular size of the subsets being specified by the individual values K_i.

Theorem 4.1. *The q-secret sharing scheme* $\mathsf{pwrShares}_q$ *is K-private.*

Proof. Consider a PPT adversary $\mathcal{A}$ that obtains $t-1$ shares $\{y_{c,i}\}_{c\in\mathcal{T},i\in[q]}$, corresponding to an execution of $\mathsf{pwrShares}_q$ which generated secrets $\{\bar{y}_i\}_{i\in[q]}$, with $\mathcal{T} = [t]\backslash\{\bar{t}\}$ for some $\bar{t} \in [t]$. Protocol $\mathsf{pwrShares}_q$ is secure, by Theorem 3.4, so the adversary has a negligible probability of determining all q secrets. However, the adversary may deduce that, for a given secret $\bar{y}_j \in \{\bar{y}_i\}_{i\in[q]}$, the absent final share must reside in the set of j-atic residues mod p, namely

$y_{\bar{t}}^j \in \Delta_{j,p}$. For each candidate value of the $\bar{t}$-th share, denoted $z_{\bar{t}}^j \in \Delta_{j,p}$, there exists a corresponding candidate value for the secret (call it $\bar{z}_j$). The set of such candidate secrets $\bar{\mathcal{Z}}_j$ (with $\bar{z}_j \in \bar{\mathcal{Z}}_j$) has cardinality $|\bar{\mathcal{Z}}_j| = |\Delta_{j,p}|$, in direct correspondence with the candidate values for the remaining share $y_{\bar{t}}^j$. The adversary cannot *a priori* determine which of the candidate values for the j-th secret is the actual secret, as all elements of $\bar{\mathcal{Z}}_j$ are viable secrets. Moreover, different adversaries $\mathcal{A}'$, which acquire distinct sets of $t-1$ shares $\{y'_{c,i}\}_{c\in\mathcal{T}',i\in[q]}$, where $\mathcal{T}' = [t]\backslash\{\bar{t}'\}$ for some $\bar{t}' \in [t]$ with $\bar{t}' \neq \bar{t}$, may also deduce that the remaining share for the j-th secret is a j-atic residue mod p, namely the candidate values for the final share are $z_{\bar{t}'}^j \in \Delta_{j,p}$. Again, for each candidate share $z_{\bar{t}'}^j$, there exists a candidate secret $\bar{z}'_j$, and the set of such candidate viable secrets $\bar{\mathcal{Z}}'_j$ has cardinality $|\Delta_{j,p}|$. Moreover, one always has $\bar{\mathcal{Z}}_j \cap \bar{\mathcal{Z}}'_j \neq \emptyset$ but, in general, one may have $\bar{\mathcal{Z}}_j \neq \bar{\mathcal{Z}}'_j$, so that different adversaries deduce that the j-th secret is private with respect to different subsets possessing the same cardinality $|\Delta_{j,p}|$. Assuming non-trivial values of $q > 1$, one has that $|\Delta_{2,p}| < |\mathbb{F}_p^*|$, so there always exists a value $j' \in [q]$ for $q > 2$ such that $K_{j'} < |\mathbb{F}_p^*|$. Thus, the q-secret sharing scheme $\mathsf{pwrShares}_q$ is K-private and, in particular, the i-th secret is $|\Delta_{i,p}|$-private. ■

Protocol $\mathsf{pwrShares}_q$ is both secure and K-private. To further discuss the security properties of $\mathsf{pwrShares}_q$, it is useful to recall that, for a given set of $t-1$ $\mathsf{polyShamir}$ (or Shamir) shares, the final share may take any value in the field, so an adversary may deduce that all values in $\mathbb{F}_p^*$ are viable candidate values for the remaining share. The set of candidate valid secrets is in one-to-one correspondence with these candidate values for the last share, giving a probability $|\mathbb{F}_p^*|^{-1}$ of successfully guessing the secret. Similarly, the probability of guessing *all q-secrets* produced by an execution of $\mathsf{pwrShares}_q$ is also $|\mathbb{F}_p|^{-1}$, *independent* of the strategy employed. This observation makes intuitive sense as any instance of $\mathsf{polyShamir}$ can be mapped to an instance of $\mathsf{pwrShares}_q$, implying that breaking the latter should in some sense be equivalent to breaking the former.[7] The equivalence is trivially transparent in the case of an adversary whose strategy is to attack the first secret $\bar{y}_1$ derived from an execution of $\mathsf{pwrShares}_q$. For a given set of $t-1$ shares for $\mathsf{pwrShares}_q$, the remaining share resides in the field, $y_t \in \mathbb{F}_p^*$, and the candidate values for the first secret are in one-to-one correspondence with the candidate values for the final share; i.e. $\bar{y}_1 \in \mathbb{F}_p^*$, as the first secret is $|\Delta_{1,p}|$-private and $|\Delta_{1,p}| = |\mathbb{F}_p^*|$. The probability of successfully guessing the first secret is $|\mathbb{F}_p^*|^{-1}$, but an adversary that successfully recovers the first secret can uniquely determine the final share and recover all secrets. Hence, the probability of successfully guessing all shares, by attacking the first secret, is equal to the probability of guessing a $\mathsf{polyShamir}$ secret, namely $|\mathbb{F}_p^*|^{-1}$. The same statement applies for an adversary that attacks the i-th secret whenever the i-th secret is $|\mathbb{F}_p^*|$-private; i.e. when $|\Delta_{i,p}| = |\mathbb{F}_p^*|$, the number of candidate valid secrets for the i-th secret is equal to the number of elements in the field, and the probability of successfully guessing the i-th secret is $|\mathbb{F}_p^*|^{-1}$. Determining the i-th secret when $|\Delta_{i,p}| = |\mathbb{F}_p^*|$ uniquely determines all q secrets, so the probability of determining all q secrets by guessing is again $|\mathbb{F}_p^*|^{-1}$.

[7] Said differently, if it were easier to break $\mathsf{pwrShares}_q$ with access to $t-1$ sets of shares $\{y_c^i\}_{c\in\mathcal{T},i\in[q]}$, than it was to break $\mathsf{polyShamir}$ (or Shamir) with $t-1$ shares $\{y_c\}_{c\in\mathcal{T}}$, one has discovered a new attack on single-secret sharing schemes - namely, map shares for a single-secret sharing scheme onto $\mathsf{pwrShares}_q$ shares and attack the latter to break the former.

Among the q secrets produced by an execution of $\mathsf{pwrShares}_q$, not all secrets will be $|\mathbb{F}_p^*|$-private. Specifically, whenever $|\Delta_{i,p}| < |\mathbb{F}_p^*|$, $|\Delta_{i,p}|$-privacy implies that the adversary has a greater probability of successfully guessing the i-th secret than of guessing e.g. the first secret. Despite this advantage, the probability of guessing all q $\mathsf{pwrShares}_q$ secrets remains at $|\mathbb{F}_p^*|^{-1}$. For example, one has $|\Delta_{2,p}| = |\mathbb{F}_p^*|/2$, so the second secret is $(|\mathbb{F}_p^*|/2)$-private and the probability of guessing the second secret is twice that of guessing the first secret. However, for a given value of the second secret, there are two candidate valid values for the first secret, such that the probability of successfully guessing both the first and second secrets is $(1/2) \times (|\mathbb{F}_p^*|/2)^{-1} = |\mathbb{F}_p^*|^{-1}$. More generally, for $|\Delta_{i,p}| < |\mathbb{F}_p^*|$, the probability of successfully guessing the i-th secret is $|\Delta_{i,p}|^{-1}$, and for a given value of the i-th secret, the probability of guessing the $(i \pm 1)$-th secret is $|\Delta_{i,p}|/|\mathbb{F}_p^*|$ (this result follows as each unique value for the i-th level share y_t^i maps into $|\mathbb{F}_p^*|/|\Delta_{i,p}|$ candidate values for the $(i+1)$-th share; similarly, there exist $|\mathbb{F}_p^*|/|\Delta_{i,p}|$ distinct values for the $(i-1)$-th level share $y_t^{(i-1)}$ that map to a given unique value of the share y_t^i). Consequently the probability of successfully guessing both the i-th secret and the $(i \pm 1)$-th secret is:

$$\Pr[(\text{guess } \bar{y}_i) \wedge (\text{guess } \bar{y}_{i\pm1})] = \frac{|\Delta_{i,p}|}{|\mathbb{F}_p^*|} \times \frac{1}{|\Delta_{i,p}|} = |\mathbb{F}_p^*|^{-1}. \tag{10}$$

As another concrete example, consider an execution of $\mathsf{pwrShares}_q$ with $q = p-1$, in which case the secret $\bar{y}_{q=p-1}$ is uniquely fixed as $|\Delta_{p-1,p}| = 1$. An adversary has a 100% probability of successfully guessing the $(p-1)$-th secret as it is 1-private and therefore uniquely determined. However, successfully determining the secret $\bar{y}_{q=p-1}$ delivers no advantage in guessing the secret $\bar{y}_{p-2}$, as the $(p-2)$-th secret is $|\Delta_{p-2,p}|$-private and $|\Delta_{p-2,p}| = |\mathbb{F}_p^*|$, giving

$$\Pr[(\text{guess } \bar{y}_{p-1}) \wedge (\text{guess } \bar{y}_{p-2})] = |\mathbb{F}_p^*|^{-1}. \tag{11}$$

Thus, the strategy of using knowledge of $\bar{y}_{p-1}$, to determine $\bar{y}_{p-2}$, is no more successful than merely trying to guess $\bar{y}_{p-2}$ (or the first secret). More generally one can consider sets of $j < q$ secrets, such that successfully guessing all j secrets uniquely fixes the value for all secrets. All such strategies have a success probability of $|\mathbb{F}_p^*|^{-1}$, independent of the sets chosen. Hence, attempting to leverage relationships between arbitrary pairs of secrets provides no gain.

Note that the above statements hold when an adversary guesses all secrets *together*, reflecting the fact that $\mathsf{pwrShares}_q$ is secure but not strongly secure. If, instead, an adversary were able to independently test its guesses for the i-th secret and confirm success/failure of each guess, the probability of guessing all secrets decreases. For example, if an adversary attacks secret $\bar{y}_2$ by guessing, there are $|\mathbb{F}_p^*|/2$ candidate values to differentiate. Once $\bar{y}_2$ is determined, however, a single guess is required to determine $\bar{y}_1$. Thus, instead of requiring at most $|\mathbb{F}_p^*| - 1$ guesses to determine $\bar{y}_1$ directly, an adversary requires at most $[(|\mathbb{F}_p^*|/2) - 1] + 1 = |\mathbb{F}_p^*|/2$ guesses to determine the values of both $\bar{y}_1$ and $\bar{y}_2$, and thus determine all q secrets. Implementations of $\mathsf{pwrShares}_q$ must therefore preclude the possibility of independently verifying the correctness of guesses for individual secrets, or only use values of q such that $|\Delta_{i,p}|$-privacy also ensures security. For example, with $q = 2$, the execution of $\mathsf{pwrShares}_2$ generates two secrets, which are $|\Delta_{1,p}|$-private and $|\Delta_{2,p}|$-private, such that the probability of independently guessing either $\bar{y}_1$ or $\bar{y}_2$ is automatically negligible provided p is chosen sufficiently large to ensure security of $\bar{y}_1$ (see Theorem 4.2 below).

Finally, note also that, in practice, an attacker must determine elements from the set of i-atic residues $\Delta_{i,p}$ to gain an advantage when attacking the i-th secret. Determining a set of i-atic residues itself incurs a cost. Efficient algorithms for calculating a cubic residue in $\mathbb{F}_p$ cost $\mathcal{O}(10\log_2 p)$, though these techniques do not purportedly yield efficient methods for $i > 3$ [CKHK14]. Naive construction of $\Delta_{i>2,p}$ therefore costs $\geq \mathcal{O}(10p\log_2 p)$, prior to undertaking any attack leveraging this knowledge.[8] Roughly speaking, for values of p sufficiently large to achieve adequate security in protocol Shamir, the set of possible values for the i-th secret in $\mathsf{pwrShares}_q$ is expected to be sufficiently large to ensure security of the i-th secret provided $|\Delta_{i,p}| \sim \mathcal{O}(p)$.

4.2 K-Security

Protocol $\mathsf{pwrShares}_q$ is K-private and secure but not strongly secure. Yet for $q = 2$, an adversary has a negligible probability of independently outputting either of the secrets. Thus, $\mathsf{pwrShares}_2$ appears to be an example of a protocol whose security properties reside somewhere between the notions of K-private q-secret sharing security and strong q-secret sharing security. The notion of q-secret sharing K-security is introduced to describe this "middle ground," in which there is a negligible probability that an adversary will output any secret generated by a protocol, and yet the secrets may not be independent (as required for strong q-secret sharing security).

Definition 4.3. *q-Secret Sharing K-Security.* A t-out-of-n ($t \leq n$) q-secret sharing protocol $\mathcal{P}_q$, with secret space $\mathbb{F}_p^*$, is said to be *K-secure* if it is (i) q-secret sharing secure; (ii) q-secret sharing K-private (implying that $K_i < |\mathbb{F}_p^*|$ for at least one secret); and (iii) probabilistic polynomial time adversaries have a negligible probability of successfully outputting *any* of the q secrets.

Theorem 4.2. *The 2-secret sharing scheme* $\mathsf{pwrShares}_2$ *is K-secure.*

Proof. By Theorem 3.4, $\mathsf{pwrShares}_2$ is secure and, by Theorem 4.1, it is K-private. Moreover, the second secret is $|\Delta_{2,p}|$-private, where $|\Delta_{2,p}| < |\mathbb{F}_p^*|$. Finally, one may always choose p sufficiently large such that both $|\mathbb{F}_p^*|^{-1}$ and $|\mathbb{F}_p^*/2|^{-1}$ are negligibly small, demonstrating K-security. ■

K-security obviously implies K-privacy. Moreover, for a K-secure q-secret sharing protocol, there is a negligible chance that a PPT adversary can output all q secrets together (i.e. q-secret sharing K-security implies q-secret sharing security). Furthermore, there is a negligible chance that a PPT adversary will output any individual secret in a K-secure q-secret sharing protocol (i.e. q-secret sharing K-security is stronger than q-secret sharing security). However, if an adversary acting against a K-secure q-secret sharing protocol manages to uncover one secret, they may gain an advantage in discovering other secrets. Thus, q-secret sharing K-security "almost" (but not quite) implies strong q-secret sharing security (i.e. q-secret sharing K-security is weaker than strong q-secret sharing security). The

[8] There are trivial exceptions to this claim - e.g. the set $\Delta_{p-1,p}$ always contains a single element.

notion of K-security explicitly indicates that the probability of determining the individual secrets is non-uniform; i.e. there exists at least one secret, in the set of q secrets, such that the probability of determining that secret is less than the probability of determining the other secrets.

In summary, the q-secret sharing protocol $\mathsf{pwrShares}_q$ is secure and K-private. Depending on the value of q, the protocol may also be K-secure or it may remain merely secure and K-private. For particular use cases, one may require q-secret sharing K-security, which restricts the allowed values of q, or q-secret sharing security and K-privacy may be sufficient, in which case q is less constrained.

5 Switched Threshold Signatures

Threshold signature schemes leverage secret sharing protocols to generate master signatures from partial signatures contributed by distinct signing authorities (see e.g. [Ped91]). For example, one may distribute shares for a (master) private key among a set of authorities, such that all signing authorities must sign a message to generate a valid signature under the master key. In this way, the signing authorities may collectively construct a valid signature under the master key without knowing the actual value of the master key [Ped91]. The secret sharing schemes described in previous sections can be employed in this context. To illustrate this point, example applications of $\mathsf{pwrSecret}_q$ and $\mathsf{pwrShares}_q$ are presented in what follows. These examples realise a primitive referred to as a *switched threshold signature*. Switched threshold signatures are a type of threshold signature with the quirky property that a threshold number of signatures of one type may be aggregated to form a master signature of a different type; i.e. the aggregated partial signatures are "switched" to a master signature with a distinct type. Two classes of switched threshold signature schemes are presented. These schemes switch between Pointcheval-Sanders (PS) signatures [PS16] and polynomial-based signatures [McD20], and may have utility in, e.g., threshold credential schemes such as Coconut [SAB+19], where the adoption of secret sharing schemes with shorter shares would permit efficiency gains.

5.1 Generalities

A variant of the Pointcheval-Sanders (PS) signature scheme [PS16] is used in the examples discussed below. Before describing the variant scheme, first recall the approach of PS. For a set of q messages $(m_1, \ldots, m_q) \in \mathbb{F}_p^q$, the signature scheme PS is specified by the following algorithms:

- $\mathsf{PS.Setup}(1^\lambda)$: On input a security parameter λ, outputs the public parameters $\mathsf{pp} = (p, \mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \tilde{g}, e)$, for generators $(g, \tilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$, and a type-3 pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.
- $\mathsf{PS.Keygen}(\mathsf{pp}, q)$: On input the public parameters pp and an integer q, randomly selects

$q+1$ field elements $(z, y_1, \ldots, y_q) \xleftarrow{R} (\mathbb{F}_p^*)^{q+1}$, and defines the secret key:

$$\mathsf{sk} = (z, y_1, \ldots, y_q). \tag{12}$$

Computes the following group elements:

$$(\tilde{Z}, \tilde{Y}_1, \ldots, \tilde{Y}_q) = (\tilde{g}^z, \tilde{g}^{y_1}, \ldots, \tilde{g}^{y_q}), \tag{13}$$

and sets the public key $\mathsf{pk} = (\tilde{g}, \tilde{Z}, \tilde{Y}_1, \ldots, \tilde{Y}_q)$.

- $\mathsf{PS.Sign}(\mathsf{pp}, \mathsf{sk}, \{m_i\}_{i\in[q]})$: On input the public parameters pp, secret key sk, and messages $\{m_i\}_{i\in[q]}$, parses sk as $(z, y_1, \ldots, y_q)$, and randomly selects a group element $\sigma_1 \xleftarrow{R} \mathbb{G}_1^*$. Outputs a signature:

$$\Sigma_{\mathsf{PS}} = (\sigma_1, \sigma) = (\sigma_1, \sigma_1^{z+\sum_i m_i y_i}), \tag{14}$$

for messages $\{m_i\}_{i\in[q]}$.

- $\mathsf{PS.Verify}(\mathsf{pp}, \mathsf{pk}, \{m_i\}_{i\in[q]}, \Sigma)$: Parses Σ as (σ_1, σ), parses pk as $(\tilde{Z}, \tilde{Y}_1, \ldots, \tilde{Y}_q)$, checks that $\sigma_1 \neq 1_{\mathbb{G}_1}$ and tests whether $e(\sigma_1, \tilde{Z} \cdot \prod_i \tilde{Y}_i^{m_i}) = e(\sigma, \tilde{g})$. If both checks are satisfied, outputs accept, otherwise outputs reject.

The security properties of this scheme were described by PS [PS16, PS18]. There exists a mapping between PS signatures and a related class of polynomial-based signature schemes [McD20]. The related multi-message signature scheme $\mathsf{PolySig}$ is defined by the following algorithms:

- $\mathsf{PolySig.Setup}(1^\lambda)$: For a given security parameter λ, outputs the public parameters $\mathsf{pp} = (p, \mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, for generators $(g, \tilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-3 pairing.

- $\mathsf{PolySig.Keygen}(\mathsf{pp}, q)$: On input public parameters pp and integer q, randomly selects two field elements, which are assigned to the secret key:

$$\mathsf{sk} = (z, y) \xleftarrow{R} (\mathbb{F}_p^*)^2. \tag{15}$$

Computes the following group elements:

$$(\tilde{Z}, \tilde{Y}_1, \ldots, \tilde{Y}_q) = (\tilde{g}^z, \tilde{g}^{y^1}, \ldots, \tilde{g}^{y^q}), \tag{16}$$

and sets $\mathsf{pk} = (\tilde{Z}, \tilde{Y}_1, \ldots, \tilde{Y}_q)$.

- $\mathsf{PolySig.Sign}(\mathsf{pp}, \mathsf{sk}, \{m_i\}_{i\in[q]})$: On input the public parameters pp, secret key sk, and messages $\{m_i\}_{i\in[q]}$, parses sk as (z, y), and randomly selects a group element $\sigma_1 \xleftarrow{R} \mathbb{G}_1^*$. Outputs a signature:

$$\Sigma_{\mathsf{PolySig}} = (\sigma_1, \sigma) = (\sigma_1, \sigma_1^{z+\sum_i m_i y^i}) \tag{17}$$

for messages $\{m_i\}_{i\in[q]}$.

- $\mathsf{PolySig.Verify}(\mathsf{pp}, \mathsf{pk}, \{m_i\}_{i\in[q]}, \Sigma)$: Parses Σ as (σ_1, σ), parses pk as $(\tilde{Z}, \tilde{Y}_1, \ldots, \tilde{Y}_q)$, checks that $\sigma_1 \neq 1_{\mathbb{G}_1}$ and tests whether $e(\sigma_1, \tilde{Z} \cdot \prod_i \tilde{Y}_i^{m_i}) = e(\sigma, \tilde{g})$. If both checks are satisfied, output accept, otherwise reject.

The secret sharing protocols $\mathsf{pwrSecret}_q$ and $\mathsf{pwrShares}_q$ generate relations between certain instantiations of the multi-message signature schemes PS and PolySig. However, to leverage these secret sharing protocols within multi-signer signature schemes, one must first overcome an obstacle. Both PS and PolySig require the signer to generate a random element σ_1. To leverage secret sharing and achieve threshold signatures, one needs all signing authorities to agree on a common value for σ_1 [SAB+19]. A suitable common value for σ_1 may be generated by leveraging a feature of Boneh-Lynn-Shacham signatures [BLS01] and taking the hash of the messages [SAB+19]. In the present context, one may define $h = H(\mathsf{aux}||m_1||\ldots||m_q)$, where aux denotes agreed-upon (optional) auxiliary information and the hash function maps field elements representing different attributes to the group $\mathbb{G}_1$, $H : \mathbb{F}_p \to \mathbb{G}_1$ [BLS01]. Thus, variants of both the PS and PolySig signature schemes may be defined, such that the PS signature is replaced with:

$$\Sigma_{\mathsf{PS}} = (h, \sigma) \quad \text{where} \quad \sigma = h^{z+\sum_i m_i y_i}, \tag{18}$$

and the PolySig signature is replaced with:

$$\Sigma_{\mathsf{PolySig}} = (h, \sigma) \quad \text{where} \quad \sigma = h^{z+\sum_i m_i y^i}. \tag{19}$$

In what follows, signature protocols with these modified signatures are still referred to as PS and PolySig schemes, respectively. The algorithms remain as above, except for the following modified definitions for the signing algorithms:[9]

- $\mathsf{PS.Sign}(\mathsf{pp}, \mathsf{sk}, \{m_i\}_{i\in[q]}, \mathsf{aux})$: Parses sk as $(z, y_1, \ldots, y_q)$, computes the group element:

  $$h = H(\mathsf{aux}||m_1||\ldots||m_q) \in \mathbb{G}_1^*, \tag{20}$$

  and outputs the signature Σ_{PS} for messages $\{m_i\}_{i\in[q]}$ and auxiliary information aux, where

  $$\Sigma_{\mathsf{PS}} = (h, \sigma) = (h, h^{z+\sum_i m_i y_i}). \tag{21}$$

- $\mathsf{PolySig.Sign}(\mathsf{pp}, \mathsf{sk}, \{m_i\}_{i\in[q]}, \mathsf{aux})$: Parses sk as (z, y), computes $h = H(\mathsf{aux}||m_1||\ldots||m_q) \in \mathbb{G}_1^*$, and outputs the signature:

  $$\Sigma_{\mathsf{PolySig}} = (h, \sigma) = (h, h^{z+\sum_i m_i y^i}) \tag{22}$$

  for messages $\{m_i\}_{i\in[q]}$ and auxiliary information aux.

Henceforth, the labels PS and PolySig refer to the protocols with the above-modified signing algorithms, PS.Sign and PolySig.Sign, respectively.

[9] The explicit argument provided for the hash function should be treated as symbolic - a particular implementation can specify the argument as desired, provided it is an appropriate function of the messages. For example, it may be preferable to include a nonce, or hash commitments to attributes, etc.

5.2 Polynomial Signatures from Switched Pointcheval-Sanders Signatures

The new secret sharing protocols can be leveraged to construct a switched threshold signature scheme in which master signatures under a polynomial-based private key are generated by switching aggregated PS signatures (issued by multiple signers); i.e. a threshold number of PS signatures are aggregated in a way that *switches* them to a polynomial-based master signature. The basic idea is that a master private key of the $\mathsf{PolySig}$ form is shared using protocol $\mathsf{pwrSecret}_q$, such that the share for the a-th signing authority forms their corresponding private key. After a threshold number of t signing authorities have signed a set of messages, these (partial) signatures may be aggregated to construct a valid signature under the master private key. The corresponding switched threshold signature scheme $\mathsf{PolySwitchPS}$ (i.e. polynomial-based master signatures formed by switching aggregated PS signatures) is defined by the following algorithms:

- $\mathsf{PolySwitchPS.Setup}(\lambda)$: On input a security parameter λ, executes $\mathsf{PolySig.Setup}(1^\lambda)$ and outputs the public parameters $\mathsf{pp} = (p, \mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \tilde{g}, e)$, where $(g, \tilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$ are generators and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-3 pairing.

- $\mathsf{PolySwitchPS.TTPKeyGen}(\mathsf{pp}, t, n, q)$:

  - On input the public parameters pp and three integers q, n and $t \leq n$, executes $\mathsf{PolySig.Keygen}(\mathsf{pp}, q)$, which generates two random field elements $(z, y) \xleftarrow{R} (\mathbb{F}_p^*)^2$, and outputs the master secret key (sk) and master public key (pk) as:

$$\begin{aligned} \mathsf{sk} &= (z, y), \\ \mathsf{pk} &= (\tilde{Z}, \tilde{Y}_1, \tilde{Y}_2, \ldots, \tilde{Y}_q) \equiv (\tilde{g}^z, \tilde{g}^{y^1}, \tilde{g}^{y^2}, \ldots, \tilde{g}^{y^q}). \end{aligned} \tag{23}$$

  - Defines $q+1$ polynomials v, w_i, for $i \in [q]$, of degree $t-1$, with coefficients in $\mathbb{F}_p$. Specifically, sets:

$$(z, y, y^2, \ldots, y^q) = (v(0), w_1(0), w_2(0), \ldots, w_q(0)), \tag{24}$$

    by executing Shamir with input secret z, which outputs the random polynomial v with $v(0) = z$, and by executing $\mathsf{pwrSecret}_q$ with input secret y, which outputs q polynomials w_i with $w_i(0) = y^i$.

  - Uses the shares $\mathsf{sh}_a^{\mathsf{Shamir}} = z_a$, and $\mathsf{sh}_a^{\mathsf{pwrSecret}_q} = (y_{a,1} \ldots, y_{a,q})$, output by Shamir and $\mathsf{pwrSecret}_q$, respectively, to provide each issuing authority $a \in [n]$ a secret key sk_a, comprised of $q+1$ field elements obtained as polynomial evaluations:

$$\mathsf{sk}_a = (z_a, y_{a,1} \ldots, y_{a,q}) \equiv (v(a), w_1(a), \ldots, w_q(a)), \tag{25}$$

    where $v(a)$ and $w_i(a)$ denote polynomial evaluations at the point a.

  - Provides the signing authorities with a corresponding public key:

$$\mathsf{pk}_a = (\tilde{Z}_a, \tilde{Y}_{a,1}, \ldots, \tilde{Y}_{a,q}) \equiv (\tilde{g}^{z_a}, \tilde{g}^{y_{a,1}} \ldots, \tilde{g}^{y_{a,q}}). \tag{26}$$

- PolySwitchPS.Sign($\mathsf{sk}_a, q, \{m_i\}_{i\in[q]}, \mathsf{aux}$): Executes PS.Sign($\mathsf{pp}, \mathsf{sk}, \{m_i\}_{i\in[q]}, \mathsf{aux}$), which parses sk_a as $(z_a, y_{a,1} \ldots, y_{a,q})$, computes $h = H(\mathsf{aux}||m_1|| \ldots ||m_q)$, and outputs the PS signature $\Sigma_a = (h, \sigma_a)$, with

$$\sigma_a = h^{z_a + \sum_{i\in[q]} m_i y_{a,i}}. \tag{27}$$

- PolySwitchPS.AggSig($\mathcal{T}, \{\Sigma_c\}_{c\in\mathcal{T}}$): Confirms that the set $\mathcal{T} \subseteq [n]$ satisfies $|\mathcal{T}| = t \leq n$. Parses each Σ_c as (h, σ_c), for $c \in \mathcal{T}$, and outputs the PolySig signature:

$$\Sigma = (h, \sigma) \leftarrow (h, \prod_{c\in\mathcal{T}} \sigma_c^{\ell_c}), \tag{28}$$

  where ℓ_c is a Lagrange coefficient:

$$\ell_c \;=\; \prod_{b\in\mathcal{T}, b\neq c} \left\{ \frac{0-b}{c-b} \right\} \bmod p, \tag{29}$$

  i.e., ℓ_c is an evaluation of the Lagrange polynomial L_c at the point zero, $\ell_c = L_c(0)$.

- PolySwitchPS.Verify($\mathsf{pp}, \mathsf{pk}, \Sigma, \{m_i\}_{i\in[q]}$): On input a set of messages $\{m_i\}_{i\in[q]}$, public key pk, and signature Σ, parses pk as $(\tilde{Z}, \tilde{Y}_1, \ldots, \tilde{Y}_q)$, parses Σ as (h, σ), and executes the check:

$$e(h, \tilde{Z} \cdot \prod_{i\in[q]} \tilde{Y}_i^{m_i}) = e(\sigma, \tilde{g}). \tag{30}$$

  Outputs accept if the check passes, otherwise outputs reject.

Note that the aggregated signature $\Sigma = (h, \sigma) = (h, \prod_{c\in\mathcal{K}} \sigma_c^{\ell_c})$ is a valid PolySig signature under the (master) secret key $\mathsf{sk} = (z, y)$. To observe this, note that the full secret key can be reconstructed via interpolation:

$$\begin{aligned} z \;&=\; v(0) \;=\; \sum_{c\in\mathcal{T}} v(c) L_c(0) \;=\; \sum_{c\in\mathcal{T}} z_c\, \ell_c, \\ y^i \;&=\; w_i(0) \;=\; \sum_{c\in\mathcal{T}} w(c) L_c(0) \;=\; \sum_{c\in\mathcal{T}} y_{c,i}\, \ell_c. \end{aligned} \tag{31}$$

This reconstruction happens in the exponent of the signature element σ:

$$\sigma \;=\; \prod_{c\in\mathcal{T}} \sigma_c^{\ell_c} \;=\; \prod_{c\in\mathcal{T}} (h^{z_c + \sum_i m_i y_{c,i}})^{\ell_c} \;=\; h^{z + \sum_i m_i y^i}, \tag{32}$$

where the final expression has the form of a PolySig signature. The following features of the switched threshold signature scheme PolySwitchPS are noted:

- PolySwitchPS employs the secret sharing scheme Shamir and the q-secret sharing scheme $\mathsf{pwrSecret}_q$ to construct shares sh_a containing $q+1$ elements. These shares are identified with the secret keys for signing authorities:

$$\mathsf{sk}_a = (\mathsf{sh}_a^{\mathsf{Shamir}}, \mathsf{sh}_a^{\mathsf{pwrShares}_q}) = (z_a, y_{a,1}, \ldots, y_{a,q}). \tag{33}$$

- Algorithm PolySwitchPS.Sign is used by a signing authority to generate a (partial) signature σ_a under their secret key sk_a. The outputted partial signatures $\Sigma_a = (h, \sigma_a)$ have the standard PS form, namely $\Sigma_{\mathsf{PS}} = (h, \sigma) = (h, h^{z+\sum_i m_i y_i})$.

- The "full" master secret key contains the elements $(z, y, y^2 \ldots, y^q)$. These elements are reconstructed via secret sharing in the exponent of element σ, from an aggregated signature $\Sigma = (h, \sigma)$, by algorithm PolySwitchPS.AggSig.

- Signing authorities generate PS signatures but the (master) aggregated threshold signature is *switched* to a PolySig signature. This transformation between partial PS signatures and aggregated PolySig signatures results from the use of $\mathsf{pwrSecret}_q$. The use of Shamir_q would instead convert the partial PS signatures into an aggregated PS signature. The switch from PS signatures to an aggregated PolySig signature is reflected in the fact that PolySwitchPS.TTPKeyGen executes PolySig.Keygen to generate PolySig-style master keys, whereas PolySwitchPS.Sign executes PS.Sign to generate PS-style partial signatures.

- Because the master/aggregated signature is switched to a PolySig signature, the master secret key need only contain two field elements, $\mathsf{sk} = (z, y)$. In particular, the length of the master secret key is independent of the number of messages q.

- The master public key pk, and signing authority public keys pk_a, contain the same number of elements. Consequently the Verify algorithm works for both aggregated and partial signatures, even though these distinct signatures have different forms (PolySig versus PS).

In summary, PolySwitchPS is a t-out-of-n switched threshold signature scheme that leverages Shamir and the q-secret sharing scheme $\mathsf{pwrSecret}_q$ to switch PS partial signatures into an aggregated PolySig master signature. The master secret key contains $\mathcal{O}(1)$ independent elements, irrespective of the number of messages.

5.3 Pointcheval-Sanders Signatures from Switched Polynomial Signatures

One can also leverage the new secret sharing schemes to construct switched threshold signatures that build master PS-style signatures by switching aggregated PolySig signatures issued by multiple signers. In this approach, a master secret key with the standard PS form, namely $\mathsf{sk} = (z, y_1 \ldots, y_q)$, is shared using secret sharing scheme $\mathsf{pwrShares}_q$, such that the share for the a-th signing authority forms their private key. When all signing authorities have signed a set of messages, the resulting partial signatures can be aggregated and switched to construct a signature under the master secret key. The corresponding switched threshold signature scheme PSSwitchPoly (i.e. PS-based master signatures formed by switching aggregated polynomial-based signatures) is defined by the following algorithms:

- $\mathsf{PSSwitchPoly.Setup}(\lambda)$: On input a security parameter λ, executes $\mathsf{PS.Setup}(1^\lambda)$, which outputs the public parameters $\mathsf{pp} = (p, \mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \tilde{g}, e)$, where $(g, \tilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$ are generators and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-3 pairing.

- $\mathsf{PSSwitchPoly.TTPKeyGen}(\mathsf{pp}, t, q)$:

  - On input the public parameters pp and integers q, t, executes $\mathsf{polyShamir}$ and $\mathsf{pwrShares}_q$, which output the shares:

$$\mathsf{sh}_a^{\mathsf{polyShamir}} = z_a \xleftarrow{R} \mathbb{F}_p^* \quad \text{and} \quad \mathsf{sh}_a^{\mathsf{pwrShares}_q} = y_a \xleftarrow{R} \mathbb{F}_p^* \quad \text{for} \quad a \in [t], \tag{34}$$

    respectively, and define $q+1$ corresponding polynomials v and w_i, $i \in [q]$. Polynomial v, constructed by $\mathsf{polyShamir}$, satisfies $v(a) = z_a$. Polynomials $w_i(x)$, for $i \in [q]$, are constructed by $\mathsf{pwrShares}_q$ using $w_i(a) = y_a^i$ (for all $a \in [t]$).

  - Using the $2t$ random field elements $\{z_a, y_a\}_{a\in[t]}$, provides each issuing authority $a \in [t]$ with a secret key and corresponding public key:

$$\begin{aligned} \mathsf{sk}_a &= (\mathsf{sh}_a^{\mathsf{polyShamir}}, \mathsf{sh}_a^{\mathsf{pwrShares}_q}) = (z_a, y_a), \\ \mathsf{pk}_a &= (\tilde{Z}_a, \tilde{Y}_{a,1}, \ldots, \tilde{Y}_{a,q}) \equiv (\tilde{g}^{z_a}, \tilde{g}^{y_a^1}, \tilde{g}^{y_a^2} \ldots, \tilde{g}^{y_a^q}). \end{aligned} \tag{35}$$

  - Constructs a master secret key:

$$\mathsf{sk} = (z, y_1, \ldots, y_q) = (v(0), w_1(0), \ldots, w_q(0)), \tag{36}$$

    and sets the corresponding master public key:

$$\mathsf{pk} = (\tilde{Z}, \tilde{Y}_1, \tilde{Y}_2, \ldots, \tilde{Y}_q) \equiv (\tilde{g}^z, \tilde{g}^{y_1}, \tilde{g}^{y_2}, \ldots, \tilde{g}^{y_q}). \tag{37}$$

- $\mathsf{PSSwitchPoly.Sign}(\mathsf{sk}_a, q, \{m_i\}_{i\in[q]}, \mathsf{aux})$: Executes $\mathsf{PolySig.Sign}(\mathsf{pp}, \mathsf{sk}, \{m_i\}_{i\in[q]}, \mathsf{aux})$, which parses sk_a as (z_a, y_a), computes $h = H(\mathsf{aux}||m_1|| \ldots ||m_q)$, and outputs the $\mathsf{PolySig}$ signature $\Sigma_a = (h, \sigma_a)$, with

$$\sigma_a = h^{z_a + \sum_{i\in[q]} m_i y_a^i}. \tag{38}$$

- $\mathsf{PSSwitchPoly.AggSig}(\{\Sigma_a\}_{a\in[t]})$: Parses each Σ_a as (h, σ_a), for $a \in [t]$, and outputs the PS signature:

$$\Sigma = (h, \sigma) \leftarrow (h, \prod_{a\in[t]} \sigma_a^{\ell_a}), \tag{39}$$

  where ℓ_a is a Lagrange coefficient:

$$\ell_a = \prod_{b\in[t], b\neq a} \left\{ \frac{0-b}{a-b} \right\} \bmod p. \tag{40}$$

- PSSwitchPoly.Verify($\mathsf{pk}, \Sigma, \{m_i\}_{i\in[q]}$): On input a set of messages $\{m_i\}_{i\in[q]}$, public key pk, and signature Σ, parses pk as $(\tilde{Z}, \tilde{Y}_1, \ldots, \tilde{Y}_q)$, parses Σ as (h, σ), and executes the check:

$$e(h, \tilde{Z} \cdot \prod_{i\in[q]} \tilde{Y}_i^{m_i}) = e(\sigma, \tilde{g}). \tag{41}$$

  Outputs accept if the check passes, otherwise outputs reject.

Note that the aggregated signature $\Sigma = (h, \sigma) = (h, \prod_{a\in[t]} \sigma_a^{\ell_a})$ is a valid PS signature under the master secret key $\mathsf{sk} = (z, y_1, \ldots, y_q)$. The secret key is reconstructed via interpolation:

$$\begin{aligned} z &= v(0) = \sum_{a\in[t]} v(a)L_a(0) = \sum_{a\in[t]} z_a\,\ell_a, \\ y_i &= w_i(0) = \sum_{a\in[t]} w(a)L_a(0) = \sum_{a\in[t]} y_a^i\,\ell_a, \end{aligned} \tag{42}$$

which happens in the exponent of the element σ:

$$\sigma = \prod_{a\in[t]} \sigma_a^{\ell_a} = \prod_{a\in[t]} (h^{z_a + \sum_i m_i y_a^i})^{\ell_a} = h^{z+\sum_i m_i y_i}. \tag{43}$$

The following features of this switched threshold signature scheme are noted:

- PSSwitchPoly employs the secret sharing scheme polyShamir, and the q-secret sharing scheme $\mathsf{pwrShares}_q$, to construct shares containing just two field elements. These shares are identified with the secret keys:

$$\mathsf{sk}_a = (\mathsf{sh}_a^{\mathsf{polyShamir}}, \mathsf{sh}_a^{\mathsf{pwrShares}_q}) = (z_a, y_a). \tag{44}$$

- Algorithm PSSwitchPoly.Sign generates partial signatures under secret key (z_a, y_a), with the form:

$$\Sigma_a = (h, \sigma_a) = (h, h^{z_a + \sum_{i\in[q]} m_i y_a^i}), \tag{45}$$

  which is the standard PolySig form, namely $\Sigma_{\mathsf{PolySig}} = (h, \sigma) = (h, h^{z+\sum_i m_i y^i})$.

- Conversely, the aggregated signatures:

$$\Sigma = (h, \sigma) = (h, h^{z+\sum_i m_i y_i}), \tag{46}$$

  have been switched from the PolySig form to the standard PS form, $\Sigma_{\mathsf{PS}} = (h, h^{z+\sum_i m_i y_i})$. Thus, signing authorities generate PolySig signatures for the messages but the (master) aggregated signature is actually a PS signature. This switching from PolySig signatures to an aggregated PS signature results from the use of $\mathsf{pwrShares}_q$.

- Because the partial signatures σ_a are PolySig signatures, the signing authorities' secret keys only contain two field elements, $\mathsf{sk}_a = (z_a, y_a)$. In particular, these secret keys contain $\mathcal{O}(1)$ elements, independent of the number of messages q.

- The Verify algorithm works for both partial (PolySig) signatures and aggregated master (PS) signatures, as the public keys for both signature types share a common form.

To summarise, PSSwitchPoly is a t-out-of-t switched threshold signature scheme that leverages the secret sharing scheme polyShamir, and the q-secret sharing scheme $\mathsf{pwrShares}_q$, to switch t aggregated PolySig signatures into a master PS signature. Signing authorities use secret keys containing just two field elements, independent of the number of messages signed.

6 General Related Shares

Before concluding, it is noted that the methods employed to construct protocol $\mathsf{pwrShares}_q$ can be generalised to construct related families of q-secret sharing schemes in which a single random value is leveraged to construct multiple shares. These more-general secret sharing schemes can also be used within threshold signature schemes. The general idea is that the protocol selects a random element y_a (per friend) as a partial share, $\mathsf{sh}_a = y_a$, and some prescribed process is employed to generate elements of the full shares, namely $y_{a,i} = F(y_a, i)$, for some function (or operation) F. For $\mathsf{pwrShares}_q$ one has $F(y_a, i) = y_a^i$. Other possibilities include generating the additional secrets using an alternative function or, e.g., a hash function H. Protocol $\mathsf{rltdShares}_q$ describes the more-general class of q-secret sharing protocols with related shares:

- $\underline{\mathsf{rltdShares}_q}$
  - Randomly draw t field elements $y_a \xleftarrow{R} \mathcal{M}_R$, for $a \in [t]$.
  - Generate $(q-1) \times t$ additional field elements $y_{a,i} \leftarrow F(y_a, i)$, for $i \in [q]$.
  - Construct q polynomials $f_i(x)$, defined by $f_i(x_a) = y_{a,i}$, for $i \in [q]$, at specified evaluation points $\{x_a\}_{a\in[t]}$.
  - Identify the q secrets with the evaluations $f_i(0)$, namely $\bar{y}_i \leftarrow f_i(0)$.
  - Distribute a single share $\mathsf{sh}_a = y_a$ to each of the t friends.
  - To recover the secrets $\{\bar{y}_i\}_{i\in[q]}$:
    - Retrieve the t shares from the friends, reconstruct the polynomials $f_i(x)$ via interpolation, and perform the evaluations $\bar{y}_i = f_i(0)$.

Provided F is deterministic, users need only distribute the (partial) shares $\mathsf{sh}_a = y_a$ to friends, from which both the friends and the original user can generate the full shares:

$$(y_{a,1}, y_{a,2}, \ldots, y_{a,q}) = (F(y_a, 1), F(y_a, 2), \ldots, F(y_a, q)). \tag{47}$$

Depending on the form of F, the protocol may or may not be K-secure. Provided protocol $\mathsf{rltdShares}_q$ possesses sufficient security properties (which requires specification of the function/algorithm F), it may be employed within threshold signature schemes, the details of

which are readily inferred from Section 5. Similar to the scheme PSSwitchPoly described in Section 5, each friend receives a secret key $\mathsf{sk}_a = (z_a, y_a)$ for random elements z_a, y_a, and signs messages $\{m_i\}_{i\in[q]}$ as $\Sigma_a = (h, \sigma_a)$ for group element h with $\sigma_a = h^{z_a + \sum_{i\in[q]} m_i y_{a,i}}$. The keys $y_{a,i}$ are constructed via Eq. (47). Aggregation of a threshold number of these signatures produces a master signature. Scheme $\mathsf{rltdShares}_q$ can therefore be employed in place of Shamir_q to construct switched threshold signature schemes of use for e.g. threshold credential schemes such as Coconut [SAB+19].

7 Conclusion

Generalisations of the single-secret sharing schemes Shamir and polyShamir were presented. Protocol Shamir was generalised to the q-secret sharing scheme $\mathsf{pwrSecret}_q$, in which users share multiple secrets whose elements are related as powers of a single secret (i.e. power secrets). Scheme polyShamir was generalised to the q-secret sharing scheme $\mathsf{pwrShares}_q$, which leverages succinct shares containing $\mathcal{O}(1)$ elements, independent of the number of secrets shared. Succinct (partial) shares were achievable as the "full" shares were related as powers of a single element (i.e. power shares). The notion of secret sharing K-privacy was introduced and the q-secret sharing protocol $\mathsf{pwrShares}_q$ was shown to be q-secret sharing secure and K-private, such that each secret remains private within a set of cardinality $|\Delta_{i,p}|$ (i.e. the i-th secret is $|\Delta_{i,p}|$-private). Despite $\mathsf{pwrShares}_q$ not being strongly q-secret sharing secure, it was shown that, for particular values of q, there is a negligible chance of a PPT adversary outputting any of the q secrets. This observation indicates that a security category exists between K-private q-secret sharing security and strong q-secret sharing security, motivating the notion of q-secret sharing K-security. Example applications of the new multi-secret sharing schemes were presented. These examples realised a primitive called a *switched threshold signature*, wherein a threshold number of signatures of one type are *switched* into a master signature of a different type. Switched threshold signature schemes were constructed in which an aggregated set of PolySig signatures was switched into a master PS signature (and vice-versa, for PolySig $\leftrightarrow$ PS). Finally, comments on a more general family of q-secret sharing schemes were offered.

Acknowledgments

The author thanks A. Rondelet, D. Tebbs, R. Toledo and M. Zajac for helpful discussions and comments on the manuscript.

References

[BDSDC+94] Carlo Blundo, Alfredo De Santis, Giovanni Di Crescenzo, Antonio Giorgio Gaggia, and Ugo Vaccaro. Multi-secret sharing schemes. In Yvo G. Desmedt,

editor, *Advances in Cryptology — CRYPTO '94*, pages 150–163, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. *Coding and Cryptology*, pages 11–46, 05 2011.

[BI93] Michael Bertilsson and Ingemar Ingemarsson. A construction of practical secret sharing schemes using linear block codes. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology — AUSCRYPT '92*, pages 67–79, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[BL90] Josh Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO' 88*, pages 27–35, New York, NY, 1990. Springer New York.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1979.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[CDI05] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *Theory of Cryptography*, pages 342–362, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[CDS19] Shion Samadder Chaudhury, Sabyasachi Dutta, and Kouichi Sakurai. Secret sharing schemes : A fine grained analysis. Cryptology ePrint Archive, Report 2019/1429, 2019. `https://eprint.iacr.org/2019/1429`.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, SFCS '85, pages 383–395, Washington, DC, USA, 1985. IEEE Computer Society.

[CKHK14] Gook Cho, Namhun Koo, Eunhye Ha, and Soonhak Kwon. New cube root algorithm based on the third order linear recurrence relations in finite fields. *Designs, Codes and Cryptography*, 75:483–495, 06 2014.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 56–72, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, SFCS '87, pages 427–438, Washington, DC, USA, 1987. IEEE Computer Society.

[FY92] Matthew Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 699–710, New York, NY, USA, 1992. ACM.

[ISN89] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

[JMO94] Wen-Ai Jackson, Keith M. Martin, and Christine M. O'Keefe. Multisecret threshold schemes. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 126–135, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[KMG+83] Ehud D. Karnin, Student Member, Jonathan W. Greene, Student Member, and Martin E. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, 29:35–41, 1983.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, pages 177–194, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[McD20] Kristian L. McDonald. The landscape of Pointcheval-Sanders signatures: Mapping to polynomial-based signatures and beyond. *In preparation*, 2020.

[PCR19] Anat Paskin-Chernivasky and Artiom Radune. On polynomial secret sharing schemes. Cryptology ePrint Archive, Report 2019/361, 2019. `https://eprint.iacr.org/2019/361`.

[Ped91] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In Donald W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, pages 522–526, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016*, pages 111–126, Cham, 2016. Springer International Publishing.

[PS18] David Pointcheval and Olivier Sanders. Reassessing security of randomizable signatures. In Nigel P. Smart, editor, *Topics in Cryptology – CT-RSA 2018*, pages 319–338, Cham, 2018. Springer International Publishing.

[SAB+19] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[Sho00] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 207–220, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[SS98] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, Harvard Data Privacy Lab, 1998.

[Sti92] D. R. Stinson. An explication of secret sharing schemes. *Designs, Codes and Cryptography*, 2:357–390, 1992.