Switched Threshold Signatures from K-Private PolyShamir Secret Sharing

Kristian L. McDonald

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 $O(1)$ 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 \textit{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.
# Contents

1 Introduction .............................. 1

2 Shamir Secret Sharing .............. 2
   2.1 Single-Secret Sharing ................. 2
   2.2 Sharing Multiple Secrets ........... 3
   2.3 Sharing Multiple (Power) Secrets 5

3 PolyShamir Secret Sharing ......... 6
   3.1 Sharing PolyShamir Secrets .......... 6
   3.2 Sharing Multiple PolyShamir Secrets 7
   3.3 Sharing Multiple PolyShamir Secrets Using Power Shares 8
   3.4 Generalising to t-out-of-n Secret Sharing Schemes 11

4 K-Private Secret Sharing .......... 12
   4.1 K-Privacy ............................. 12
   4.2 K-Security ............................ 17

5 Switched Threshold Signatures ..... 18
   5.1 Generalities .......................... 18
   5.2 Polynomial Signatures from Switched Pointcheval-Sanders Signatures 21
   5.3 Pointcheval-Sanders Signatures from Switched Polynomial Signatures 23

6 General Related Shares ............ 26

7 Conclusion ........................... 27
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 is 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 [Pod91], and verifiable secret sharing schemes [CGMA83] 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 $O(1)$ 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, $\alpha_c \overset{R}{\leftarrow} \mathbb{F}_p^*$, for $c \in [t-1]$.
  - Use $\bar{y}$ and the elements $\alpha_c$ 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 $sh_a \equiv y_a = f(x_a)$, is called a secret share or simply a share\footnote{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 $sh_a$.}.
  - Distribute a single share to each of the $n$ friends.
Secret recovery:

* Select a subset of \( t \) friends, labeled as \( T_s \subseteq [n] \), with \(|T_s| = t\), and request their shares.

* Reconstruct the polynomial \( f(x) \) using the shares \( s_b = y_b \), for \( b \in T_s \), 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 \( T_s \subseteq [n] \) respectively (with \(|T_s| = t\)); e.g. \( a \in [n] \) and \( b \in T_s \), 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 \( c \in T \), with \(|T| = t - 1 \), where \([t - 1] \subseteq [n] \) and \( T \subseteq T_s \subseteq [n] \).

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 \( \lambda \) is smaller than \( O(1/\text{poly}(\lambda)) \) and denoted as \( \text{negl}(\lambda) \).

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 \( P \) is said to be secure if any probabilistic polynomial time adversary \( A \) that obtains \( t - 1 \) shares \( \{y_c\}_{c \in T} \), with \(|T| = t - 1 \), has a negligible chance of outputting the corresponding secret \( \bar{y} \). Namely:

\[
\Pr \left[ F_p \leftarrow \mathcal{G}(1^\lambda); \ (\bar{y}, \{y_a\}_{a \in [t]}) \leftarrow P(\text{input}); \ t^{R} \leftarrow [t]; \ T \equiv [t] \setminus \{t\}; \ z \leftarrow A(\{y_c\}_{c \in T}) : \ z = \bar{y} \right] = \text{negl}(\lambda),
\]

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 \( \lambda \). The input data to protocol \( P \) may include the secret \( \bar{y} \) itself, as occurs for \( P = \text{Shamir} \). Protocol Shamir is unconditionally (i.e. information-theoretically) secure \([\text{Sha79]}\) 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. \([\text{BDSDC}^{+94}, \text{JMO94}]\)). 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:

- **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:
    \[
    \alpha_{(c,i)} \overset{R}{\leftarrow} \mathbb{F}^*_p, \quad \text{for} \quad i \in [q] \quad \text{and} \quad c \in [t-1],
    \]
    and define \( q \) polynomials:
    \[
    f_i(x) = \bar{y}_i + \sum_{c=1}^{t-1} \alpha_{(c,i)} x^c \quad \text{for} \quad i \in [q].
    \]
  - Evaluate the polynomials \( f_i(x) \) at the evaluation points \( \{x_a\}_{a \in [t]} \). Label the evaluations as \( y_{a,i} = f_i(x_a) \) and distribute a share \( sh_a \) to each of the \( n \) friends, where
    \[
    sh_a = (y_{a,1}, \ldots, y_{a,q}).
    \]
  - To reconstruct the secrets \( \{\bar{y}_i\}_{i \in [q]} \):
    * Request the shares \( sh_b \) from \( t \) friends (described by the index \( b \in \mathcal{T}_s \subset [n] \), where \( |\mathcal{T}_s| = t \).
    * 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 \( P_q \) is said to be secure if any probabilistic polynomial time adversary \( A \) 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:

\[
\Pr \left[ F_p \leftarrow \mathcal{G}(1^\lambda); \left\{ \{\bar{y}_i\}_{i \in [q]}, \{y_{a,i}\}_{a \in [t], i \in [q]} \right\} \leftarrow P_q(\text{input}); \bar{t} \overset{R}{\leftarrow} [t]; \mathcal{T} \equiv [t] \setminus \{\bar{t}\}; \left\{ z_i \right\}_{i \in [q]} \leftarrow A(\left\{ y_{c,i} \right\}_{c \in \mathcal{T}, i \in [q]}); z_i = \bar{y}_i, \forall i \in [q] \right] = \text{negl}(\lambda),
\]

where \text{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 \( P_q \) is said to be **strongly secure** if:

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

\[
\Pr \left[ P_p \leftarrow \mathcal{G}(1^\lambda); \ (\{\bar{y}_i\}_{i \in [q]}, \{y_{a,i}\}_{a \in [t], i \in [q]}) \leftarrow P_q(\text{input}); \ t \overset{R}{\leftarrow} [t]; \ T \equiv [t]\backslash\{t\}; \ \bar{z}_j \leftarrow A(\{y_{e,i}\}_{e \in T, i \in [q]}): \ \bar{z}_j \in \{\bar{y}_i\}_{i \in [q]} \right] = \text{negl}(\lambda),
\]

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. \( \blacksquare \)

### 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 \( \text{pwrSecret}_q \) may be appropriate:

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

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

3 PolyShamir Secret Sharing

This section discusses a variant of \( \text{Shamir} \) which may be generalised to obtain variant multi-
secret sharing protocols.

3.1 Sharing PolyShamir Secrets

A user (Alice) of protocol \( \text{Shamir} \) 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 \( \text{Shamir} \) 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 \( \text{polyShamir} \) is suitable for such cases:

- \( \text{polyShamir} \)
  - Randomly draw \( t \) field elements \( y_a \leftarrow \mathbb{F}_p^* \), 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 $sh_a = y_a$ 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 $A_{\text{polyShamir}}$ 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 polyShamir$_q$, one has the following theorem:

**Theorem 3.2.** The $q$-secret sharing scheme polyShamir$_q$ is secure.

**Proof.** 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 $\text{polyShamir}_q$ secret is sufficient to determine the polynomial $f(x)$ and obtain all other secrets (if $t-1$ shares are already known). Consequently $\text{polyShamir}_q$ is secure but not strongly secure. Users of $\text{polyShamir}_q$ 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 polyShamir$_q$, defined as follows:

- **polyShamir$_q$**:
  - Randomly draw $q \times t$ field elements, $y_{a,i} \leftarrow R_{F^*_p}$, 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:
    \[ s_{h_a} = (y_{a,1}, \ldots, y_{a,q}) \]  

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 polyShamir$_q$ follows immediately as the shares are uniformly distributed at random (similar to polyShamir).

**Theorem 3.3.** The $q$-secret sharing scheme polyShamir$_q$ is strongly secure.

**Proof.** Each secret generated by an execution of 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 polyShamir$_q$ is strongly secure. $\blacksquare$

### 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 pwrShares$_q$, an alternative $q$-secret generalisation of polyShamir, is defined as follows:
• pwrShares q:
  - Randomly draw t elements \( y_a \overset{R}{\leftarrow} \mathbb{F}_p^* \) 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, \( sk_a = y_a \), 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 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:

\[
\overline{sh}_a = (y_a, y_a^2, \ldots, y_a^q),
\]

for each friend \( a \in [t] \). Users need not distribute the full shares \( \overline{sh}_a \), as these may be derived from the (partial) shares \( sh_a = y_a \). 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 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 pwrShares q. In particular, the (partial) share length remains at \( O(1) \) elements, independent of the number of secrets q. Similar to polyShamir and polyShamir q, 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 pwrShares q is secure

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

\(^3\)If the friends are honest, the scheme also works if the friends directly draw their own share, \( sh_a = y_a \overset{R}{\leftarrow} \mathbb{F}_p \).
The above proof avails the following intuition. Consider the execution of polyShamir with secret \( \hat{y} \) 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 \( y_c, y_t \sim \mathbb{F}_p \), 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:
\[
\{ y^t_c \}_{c \in [t-1]} \cup \{ y^t_i \}, \quad \forall i \in [q].
\] (6)

These shares define \( q \) polynomials \( f_i(x) \), via the evaluations \( f_i(x_c) = y^t_i \) and \( f_i(x_t) = y^t_i \). Together with the evaluations \( \hat{y}_i = f_i(0) \), these parameters form a valid set under \( \text{pwrShares}_q \), such that:
\[
\hat{y}_i = \hat{y}_1 \leftarrow f_1(0) = f(0).
\] (7)

Thus, any set of valid parameters for polyShamir automatically defines a corresponding set of valid parameters for \( \text{pwrShares}_q \). More precisely, there is a one-to-one mapping from a valid polyShamir share \( (y_a) \), to a related full \( \text{pwrShares}_q \) share, with action
\[
y_a \rightarrow (y_a, y_a^2, \ldots, y_a^q).
\] (8)

Consequently if the \( q \)-secret scheme \( \text{pwrShares}_q \) 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 \( \text{pwrShares}_q \), and leveraging the security vulnerability of \( \text{pwrShares}_q \) to break polyShamir. Hence, security of the multi-secret scheme \( \text{pwrShares}_q \) 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 polyShamir from a set of valid parameters for Shamir, such that the \( i = 1 \) polynomial in polyShamir matches the single polynomial used in Shamir. Security vulnerabilities that reveal all \( q \) secrets in \( \text{pwrShares}_q \) could therefore be used to break Shamir.

Note that \( \text{pwrShares}_q \) 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 \( \text{pwrShares}_q \) and the multi-secret sharing schemes polyShamir and Shamir reflect the fact that the \( q \) polynomials \( f_i(x) \) are related in \( \text{pwrShares}_q \). To elaborate on this point, consider an adversary \( A_{\text{pwrShares}_q} \), acting against \( \text{pwrShares}_q \) 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 \( \hat{y}_i \), for \( i \in [q] \). In contrast, an adversary \( A_{\text{Shamir}_q} \), acting against Shamir, which obtains a complete set of shares \( \{ y_{a,j} \}_{a \in [t], j} \) for some fixed value of \( j \in [q] \), can generate the polynomial \( f_j(x) \) and obtain the secret \( \hat{y}_j = f_j(0) \), but cannot generate the polynomials \( f_i(x) \) for \( i \neq j \), so the secrets \( \hat{y}_{i \neq j} \) remain secure. Both adversaries have access to \( t \) field elements, yet \( A_{\text{pwrShares}_q} \) achieves a full break of \( \text{pwrShares}_q \).

---

4This point is discussed in Section 4.
whereas $A_{Shamir_q}$ only achieves a partial break of $Shamir_q$. To achieve a full break of $Shamir_q$, adversary $A_{Shamir_q}$ must instead obtain all $t \times q$ shares $\{y_{a,i}\}_{i \in [q], a \in [t]}$. Similar comments hold for $polyShamir_q$. Thus, in some sense the security of $pwrShares_q$ is more similar to that of $Shamir_q$, 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 $polyShamir_q$, where $t \times q$ shares are needed to achieve a full break. Speaking informally, $pwrShares_q$ 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.\footnote{This comment is unsurprising - $pwrShares_q$ leverages the same amount of randomness as e.g. $Shamir_q$, and consequently has similar security properties.}

It may appear strange that, to securely share $q$ secrets among $t$ friends using $polyShamir_q$, one must generate and distribute $t \times q$ secret field elements, whereas to share $q$ secrets among $t$ friends using $pwrShares_q$, one need only generate and distribute $t$ random elements. To better understand why $pwrShares_q$ 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

$$f_i(x_c) = \bar{y}_c + \sum_{c'=1}^{t-1} \alpha(c', i) x_{c'} = y_c'$$

$$f_i(x_t) = \bar{y}_t + \sum_{c'=1}^{t-1} \alpha(c', i) x_{c'} = Y_t', \quad (9)$$

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 $\overline{sh}_a = (y_a, y_a^2, \ldots, y_a^q)$, can be calculated with knowledge of the (partial) shares $sh_a = y_a$, the underlying system of equations remains under-determined unless an adversary knows \textit{all} partial shares, so the system cannot be solved. Accordingly, $pwrShares_q$ 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 polyShamir to a multi-secret sharing scheme with power secrets. Protocol 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 polyShamir, polyShamir$_q$ and pwrShares$_q$ may be generalised to $t$-out-of-$n$ secret sharing schemes. In scheme polyShamir, users construct polynomials by drawing $t$ shares at random, $y_a \overset{R}{\leftarrow} \mathbb{F}_p$, and identifying the shares with polynomial

---

This comment is unsurprising - $pwrShares_q$ leverages the same amount of randomness as e.g. $Shamir_q$, and consequently has similar security properties.
evaluations \( f(x_a) = y_a \), for all \( a \in [t] \).

Protocol \( \text{polyShamir} \) 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 \( \text{sh}_{a'} = y_{a'} \) 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 \( \text{polyShamir}_q \) and \( \text{pwrShares}_q \) 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 \( \text{sh}_{a'} = (y_{a'1}, \ldots, y_{a'q}) \) 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 \( \text{sh}_a = y_a \) (for \( a \in [t] \)), whereas other friends would receive longer shares \( \text{sh}_{a'} = (y_{a'1}, \ldots, y_{a'q}) \) (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 \( \text{sh}_a \rightarrow (y_a, y_a^2, \ldots, y_a^q) \) 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.

\section*{4 \textbf{K-Private Secret Sharing}}

\subsection*{4.1 K-Privacy}

To further discuss the differences between strongly secure \( q \)-secret sharing protocols such as \( \text{Shamir}_q \) and secure \( q \)-secret sharing protocols such as \( \text{pwrShares}_q \), 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 \( F_p^* \), is said to be \( K \)-private if any probabilistic polynomial time adversary \( A \) that obtains shares \( \{y_c\}_{c \in T} \) (with \( |T| = t-1 \)), corresponding to secret \( \tilde{y} \), may leverage the shares to deduce that there exists a set \( \Delta_T \subseteq F_p^* \) of cardinality \( |\Delta_T| = K \), such that \( \tilde{y} \in \Delta_T \).

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 \( \tilde{y} \) must reside within a set of candidate secrets \( \Delta_T \), 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

---

In the \( q \)-secret sharing schemes \( \text{polyShamir}_q \) and \( \text{pwrShares}_q \), a similar process is employed for multiple polynomials.

---

\( ^6 \)In the \( q \)-secret sharing schemes \( \text{polyShamir}_q \) and \( \text{pwrShares}_q \), a similar process is employed for multiple polynomials.
Protocol permits an adversary to deduce that the secret belongs to some subset \( \Delta_\tau \subseteq F_p^* \), 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 \textit{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 \texttt{polyShamir}, referred to as \( N\text{-polyShamir} \):

- **\( N\text{-polyShamir} \)**
  - Randomly draw \( t \) field elements \( y_a \leftarrow_{\text{R}} F_p^* \) 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 \( \bar{y} = f(0) \).
  - Distribute a single share \( sh_a = y_a^N \) 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 \( N\text{-polyShamir} \) is identical to \texttt{polyShamir} except that the shares \( sh_a = y_a^N \) are raised to the power of \( N \). Consequently all shares reside in the set of \( N \)-atic residues mod \( p \), namely \( sh_a \in \Delta_{N,p} \) for all \( a \in [t] \). Consider an adversary \( A \) that acquires \( t-1 \) shares \( \{y_c^N\}_{c \in \tau} \). Using knowledge of the protocol, the adversary may deduce that the missing share must also be an \( N \)-atic residue mod \( p \), i.e. \( y_i^N \in \Delta_{N,p} \). Thus, for each candidate value of the share \( y_i^N \), there exists a unique candidate value for the secret \( \tilde{z} \in F_p^* \), so the set of candidate valid secrets \( \tilde{Z} \), where \( \tilde{z} \in \tilde{Z} \), has cardinality \( |\tilde{Z}| = |\Delta_{N,p}| \). Moreover, a different adversary \( A' \) that acquires a non-identical set of \( t-1 \) shares \( \{y_c^N\}_{c \in \tau} \), can also deduce that the secret must reside within a set of candidate valid secrets \( \tilde{Z}' \), with \( \tilde{z}' \in \tilde{Z}' \), where \( |\tilde{Z}'| = |\Delta_{N,p}| \). In general, one may have \( \tilde{Z}' \neq \tilde{Z} \) but with \( \tilde{Z}' \cap \tilde{Z} \neq \emptyset \). The full secret space \( \tilde{Y} \) is a subset of \( F_p^* \), defined as the set of values \( \tilde{y} \in F_p^* \) 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 \( \tilde{Y} \) but distinct adversaries, which obtain different sets of shares, deduce that the secret must reside within different sets of candidate secrets such as \( \tilde{Z} \subseteq \tilde{Y} \) and \( \tilde{Z}' \subseteq \tilde{Y} \).

Combining the above information demonstrates that protocol \( N\text{-polyShamir} \) is \( |\Delta_{N,p}| \)-private. For the particular case with \( N = 1 \), one has \( 1\text{-polyShamir} = \text{polyShamir} \) and protocol \( 1\text{-polyShamir} \) is both \( |\Delta_{1,p}| \)-private and secure. However, as \( |\Delta_{1,p}| = |F_p^*| \), 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 \( 2\text{-polyShamir} \) 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 \( \bar{Z} \) of candidate secrets, of size \( |\bar{Z}| = |\Delta_{N,p}| \), 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 \( |\Delta_{p}| \)-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 \( N \text{polyShamir} \) 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 \( P_q \), with secret space \( \mathbb{F}_p^* \), is said to be \( K \)-private if any probabilistic polynomial time adversary \( A \) that obtains shares \( \{y_{c,i}\}_{c \in T, i \in [q]} \) (with \( |T| = t - 1 \), corresponding to secrets \( \{\bar{y}_i\}_{i \in [q]} \), may leverage the shares to deduce that there exists sets \( \Delta_{T,i} \subseteq \mathbb{F}_p^* \), such that \( y_i \in \Delta_{T,i} \) and \( |\Delta_{T,i}| = K_i \leq |\mathbb{F}_p^*| \) for all \( i \in [q] \). Moreover, there exists at least one value \( j \in [q] \) such that \( K_j < |\mathbb{F}_p^*| \). 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 \( \Delta_{T,i} \subseteq \mathbb{F}_p^* \), with \( |\Delta_{T,i}| = K_i \), 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 \( \Delta_{T,i} \), 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 \textit{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 \( K_j < |\mathbb{F}_p^*| \), 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 \( \text{pwrShares}_q \) is \( K \)-private.

**Proof.** Consider a PPT adversary \( A \) that obtains \( t - 1 \) shares \( \{y_{c,i}\}_{c \in T, i \in [q]} \), corresponding to an execution of \( \text{pwrShares}_q \) which generated secrets \( \{\bar{y}_i\}_{i \in [q]} \), with \( T = [t] \setminus \{\bar{t}\} \) for some \( \bar{t} \in [t] \). Protocol \( \text{pwrShares}_q \) 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...
For each candidate value of the $i$-th share, denoted $z_j^i \in \Delta_{j,p}$, there exists a corresponding candidate value for the secret (call it $\bar{z}_j$). The set of such candidate secrets $\bar{Z}_j$ (with $\bar{z}_j \in \bar{Z}_j$) has cardinality $|\bar{Z}_j| = |\Delta_{j,p}|$, in direct correspondence with the candidate values for the remaining share $y_j^i$. The adversary cannot a priori determine which of the candidate values for the $j$-th secret is the actual secret, as all elements of $\bar{Z}_j$ are viable secrets. Moreover, different adversaries $\mathcal{A}'$, which acquire distinct sets of $t-1$ shares $\{y_{c,i}^j\}_{c\in T', i\in [q]}$, where $T' = [t]\{i\}$ for some $i \in [t]$ with $i \neq t$, 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 $z_j^i \in \Delta_{j,p}$. Again, for each candidate share $z_j^i$, there exists a candidate secret $\bar{z}_j$, and the set of such candidate viable secrets $\bar{Z}_j$ has cardinality $|\Delta_{j,p}|$. Moreover, one always has $\bar{Z}_j \cap \bar{Z}_j' = \emptyset$ but, in general, one may have $\bar{Z}_j \neq \bar{Z}_j'$, 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 $\text{pwr Shares}_q$ is $K$-private and, in particular, the $i$-th secret is $|\Delta_{i,p}|$-private.

Protocol $\text{pwr Shares}_q$ is both secure and $K$-private. To further discuss the security properties of $\text{pwr Shares}_q$, it is useful to recall that, for a given set of $t-1$ 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 $\text{pwr Shares}_q$ is also $|\mathbb{F}_p^*|^{-1}$, independent of the strategy employed. This observation makes intuitive sense as any instance of polyShamir can be mapped to an instance of $\text{pwr Shares}_q$, implying that breaking the latter should in some sense be equivalent to breaking the former.\footnote{\textit{\textsuperscript{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 $\text{pwr Shares}_q$. For a given set of $t-1$ shares for $\text{pwr Shares}_q$, the remaining share resides in the field, $y_1 \in \mathbb{F}_p^*$, 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 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 $|\mathbb{F}_p^*|^{-1}$-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 $|\mathbb{F}_p^*|^{-1}$.

\footnote{\textit{\textsuperscript{7}}} Said differently, if it were easier to break $\text{pwr Shares}_q$ with access to $t-1$ sets of shares $\{y_{c,i}^j\}_{c \in T, i \in [q]}$, than it was to break polyShamir (or Shamir) with $t-1$ shares $\{y_{c,i}^j\}_{c \in T}$, one has discovered a new attack on single-secret sharing schemes - namely, map shares for a single-secret sharing scheme onto $\text{pwr Shares}_q$ shares and attack the latter to break the former.
Among the $q$ secrets produced by an execution of $\text{pwrShares}_q$, 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 $q$ $\text{pwrShares}_q$ 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 $y_i^t$ 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 $y_i^t$). Consequently the probability of successfully guessing both the $i$-th secret and the $(i \pm 1)$-th secret is:

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

(10)

As another concrete example, consider an execution of $\text{pwrShares}_q$ 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

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

(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 $\text{pwrShares}_q$ 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 $\text{pwrShares}_q$ 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 $\text{pwrShares}_2$ 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 $O(10 \log_2 p)$, though these techniques do not purportedly yield efficient methods for $i > 3$ [CKHK14]. Naive construction of $\Delta_{i>2,p}$ therefore costs $\geq O(10p \log_2 p)$, 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 $\text{pwrShares}_q$ is expected to be sufficiently large to ensure security of the $i$-th secret provided $|\Delta_{i,p}| \sim O(p)$.

### 4.2 K-Security

Protocol $\text{pwrShares}_q$ 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, $\text{pwrShares}_2$ 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 $\text{pwrShares}_2$ is $K$-secure.

**Proof.** By Theorem 3.4, $\text{pwrShares}_2$ 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. $\blacksquare$

$K$-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 $\text{pwrShares}_q$ 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 $\text{pwrSecret}_q$ and $\text{pwrShares}_q$ 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 [SAB19], 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:

- $\text{PS.Setup}(1^\lambda)$: On input a security parameter $\lambda$, outputs the public parameters $\text{pp} = (p, \mathbb{F}_p, G_1, G_2, G_T, g, \tilde{g}, e)$, for generators $(g, \tilde{g}) \in G_1 \times G_2$, and a type-3 pairing $e : G_1 \times G_2 \rightarrow G_T$.

- $\text{PS.Keygen(pp, q)}$: On input the public parameters $\text{pp}$ and an integer $q$, randomly selects
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 PolySig is defined by the following algorithms:

- **PolySig.Setup(1^λ)**: For a given security parameter \( \lambda \), outputs the public parameters \( pp = (p, \mathbb{F}_p, G_1, G_2, G_T, e) \), for generators \( (g, \tilde{g}) \in G_1 \times G_2 \), where \( e : G_1 \times G_2 \to G_T \) is a type-3 pairing.
- **PolySig.Keygen(pp, q)**: On input public parameters \( pp \) and integer \( q \), randomly selects two field elements, which are assigned to the secret key:

\[
\text{sk} = (z, y) \overset{R}{\leftarrow} (\mathbb{F}_p^*)^2.
\]

Computes the following group elements:

\[
(\tilde{Z}, \tilde{Y}_1, \ldots, \tilde{Y}_q) = (\tilde{g}^z, \tilde{g}^{y_1}, \ldots, \tilde{g}^{y_q}),
\]

and sets \( \text{pk} = (\tilde{g}, \tilde{Z}, \tilde{Y}_1, \ldots, \tilde{Y}_q) \).
- **PolySig.Sign(pp, 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 \( (z, y, \ldots, y_q) \), and randomly selects a group element \( \sigma_1 \overset{R}{\leftarrow} G_1^* \). Outputs a signature:

\[
\Sigma_{\text{PolySig}} = (\sigma_1, \sigma) = (\sigma_1, \sigma_1^{z+\sum_i m_i y_i})
\]

for messages \( \{m_i\}_{i \in [q]} \).
• PolySig.Verify\((pp, pk, \{m_i\}_{i \in [q]}, \Sigma)\): Parses \(\Sigma\) as \((\sigma_1, \sigma)\), 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 \text{accept}, otherwise \text{reject}.

The secret sharing protocols \text{pwrSecret}_q and \text{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 \(\sigma_1\). To leverage secret sharing and achieve threshold signatures, one needs all signing authorities to agree on a common value for \(\sigma_1\) [SAB+19]. A suitable common value for \(\sigma_1\) 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 \(h = H(\text{aux} || m_1 || \ldots || m_q)\), where \(\text{aux}\) denotes agreed-upon (optional) auxiliary information and the hash function maps field elements representing different attributes to the group \(\mathbb{G}_1\), \(H : \mathbb{F}_p \rightarrow \mathbb{G}_1\) [BLS01]. Thus, variants of both the PS and PolySig signature schemes may be defined, such that the PS signature is replaced with:

\[
\Sigma_{\text{PS}} = (h, \sigma) \quad \text{where} \quad \sigma = h^{z + \sum_i m_i y_i},
\]  

(18)

and the PolySig signature is replaced with:

\[
\Sigma_{\text{PolySig}} = (h, \sigma) \quad \text{where} \quad \sigma = h^{z + \sum_i m_i y^i}.
\]  

(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:\(^9\)

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

\[
h = H(\text{aux} || m_1 || \ldots || m_q) \in \mathbb{G}_1^*,
\]  

(20)

and outputs the signature \(\Sigma_{\text{PS}}\) for messages \(\{m_i\}_{i \in [q]}\) and auxiliary information \(\text{aux}\), where

\[
\Sigma_{\text{PS}} = (h, \sigma) = (h, h^{z + \sum_i m_i y_i}).
\]  

(21)

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

\[
\Sigma_{\text{PolySig}} = (h, \sigma) = (h, h^{z + \sum_i m_i y^i})
\]  

(22)

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

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 PolySig form is shared using protocol $\text{pwrSecret}_q$, 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 PolySwitchPS (i.e. polynomial-based master signatures formed by switching aggregated PS signatures) is defined by the following algorithms:

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

- **PolySwitchPS.TTPKeyGen$(pp, t, n, q)$**:  
  - On input the public parameters $pp$ and three integers $q$, $n$ and $t \leq n$, executes PolySig.Keygen$(pp, q)$, which generates two random field elements $(z, y) \leftarrow (\mathbb{F}_p^*)^2$, and outputs the master secret key $(sk)$ and master public key $(pk)$ as:
    \begin{align*}
    sk &= (z, y), \\
    pk &= (\tilde{Z}, \tilde{Y}_1, \tilde{Y}_2, \ldots, \tilde{Y}_q) \equiv (\tilde{g}^z, \tilde{g}^y, \tilde{g}^{y^2}, \ldots, \tilde{g}^{y^q}).
    \end{align*}

  - Defines $q + 1$ polynomials $v, w_i$, for $i \in [q]$, of degree $t - 1$, with coefficients in $\mathbb{F}_p$. Specifically, sets:
    \begin{align*}
    (z, y, y^2, \ldots, y^q) &= (v(0), w_1(0), w_2(0), \ldots, w_q(0)),
    \end{align*}
    by executing Shamir with input secret $z$, which outputs the random polynomial $v$ with $v(0) = z$, and by executing $\text{pwrSecret}_q$ with input secret $y$, which outputs $q$ polynomials $w_i$ with $w_i(0) = y^i$. 

  - Uses the shares $sh_a^{\text{Shamir}} = z_a$, and $sh_a^{\text{pwrSecret}_q} = (y_{a,1}, \ldots, y_{a,q})$, output by Shamir and $\text{pwrSecret}_q$, respectively, to provide each issuing authority $a \in [n]$ a secret key $sk_a$, comprised of $q + 1$ field elements obtained as polynomial evaluations:
    \begin{align*}
    sk_a &= (z_a, y_{a,1}, \ldots, y_{a,q}) \equiv (v(a), w_1(a), \ldots, w_q(a)),
    \end{align*}
    where $v(a)$ and $w_i(a)$ denote polynomial evaluations at the point $a$. 

  - Provides the signing authorities with a corresponding public key:
    \begin{align*}
    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}}).
    \end{align*}
• PolySwitchPS.Sign$(sk_a, q, \{m_i\}_{i \in [q]}, \text{aux})$: Executes PS.Sign$(pp, sk, \{m_i\}_{i \in [q]}, \text{aux})$, which parses $sk_a$ as $(z_a, y_{a,1}, \ldots, y_{a,q})$, computes $h = H(\text{aux}||m_1|| \ldots ||m_q)$, and outputs the PS signature $\Sigma_a = (h, \sigma_a)$, with

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

(27)

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

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

(28)

where $\ell_c$ is a Lagrange coefficient:

$$\ell_c = \prod_{b \in T, b \neq c} \left\{ \frac{0 - b}{c - b} \right\} \mod p,$$

(29)

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

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

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

(30)

Outputs accept if the check passes, otherwise outputs reject.

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

$$z = v(0) = \sum_{c \in T} v(c)L_c(0) = \sum_{c \in T} z_c \ell_c,$$

$$y^i = w_i(0) = \sum_{c \in T} w(c)L_c(0) = \sum_{c \in T} y_{c,i} \ell_c.$$ 

(31)

This reconstruction happens in the exponent of the signature element $\sigma$:

$$\sigma = \prod_{c \in T} \sigma_c^{\ell_c} = \prod_{c \in T} (h^{z_c + \sum_{i} m_{i,y_{c,i}}})^{\ell_c} = h^{z + \sum_{i} m_{i,y^i}},$$

(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 pwrSecret, to construct shares $sh_a$ containing $q+1$ elements. These shares are identified with the secret keys for signing authorities:

$$sk_a = (sh_a^{\text{Shamir}}, sh_a^{\text{pwrShares}_q}) = (z_a, y_{a,1}, \ldots, y_{a,q}).$$ 

(33)
• Algorithm PolySwitchPS.Sign is used by a signing authority to generate a (partial) signature \( \sigma_a \) under their secret key \( sk_a \). The outputted partial signatures \( \Sigma_a = (h, \sigma_a) \) have the standard PS form, namely \( \Sigma_{PS} = (h, \sigma) = (h, h^z \sum_{i=1}^{m} y_i) \).

• 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 \( \sigma \), 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 \( \text{pwrSecret}_q \). 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, \( sk = (z, y) \). 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 \( \text{pwrSecret}_q \) 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 \( sk = (z, y_1, \ldots, y_q) \), is shared using secret sharing scheme \( \text{pwrShares}_q \) 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:
• **PSSwitchPoly.Setup(λ):** On input a security parameter λ, executes PS.Setup(1^λ), which outputs the public parameters pp = (p, F_p, G_1, G_2, G_T, g, ĝ, e), where (g, ĝ) ∈ G_1 × G_2 are generators and e : G_1 × G_2 → G_T is a type-3 pairing.

• **PSSwitchPoly.TTPKeyGen(pp, t, q):**

  - On input the public parameters pp and integers q, t, executes polyShamir and pwrShares_q, which output the shares:

\[
\text{sh}_a^{\text{polyShamir}} = z_a \overset{R}{\leftarrow} F_p^* \quad \text{and} \quad \text{sh}_a^{\text{pwrShares}} = y_a \overset{R}{\leftarrow} F_p^* \quad \text{for} \quad a \in [t],
\]

respectively, and define q + 1 corresponding polynomials v and w_i, i ∈ [q]. Polynomial v, constructed by polyShamir, satisfies v(a) = z_a. Polynomials w_i(x), for i ∈ [q], are constructed by pwrShares_q using w_i(a) = y_a^i (for all a ∈ [t]).

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

\[
\text{sk}_a = (\text{sh}_a^{\text{polyShamir}}, \text{sh}_a^{\text{pwrShares}}) = (z_a, y_a), \quad \text{pk}_a = (\tilde{Z}_a, \tilde{Y}_a, \ldots, \tilde{Y}_{a,q}) \equiv (\tilde{g}^{z_a}, \tilde{g}^{y_1}, \tilde{g}^{y_2}, \ldots, \tilde{g}^{y_q}).
\]

  - Constructs a master secret key:

\[
\text{sk} = (z, y_1, \ldots, y_q) = (v(0), w_1(0), \ldots, w_q(0)),
\]

and sets the corresponding master public key:

\[
\text{pk} = (\tilde{Z}, \tilde{Y}_1, \tilde{Y}_2, \ldots, \tilde{Y}_q) \equiv (\tilde{g}^z, \tilde{g}^{y_1}, \tilde{g}^{y_2}, \ldots, \tilde{g}^{y_q}).
\]

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

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

• **PSSwitchPoly.Aggsig(\{\Sigma_a\}_{a \in [t]}):** Parses each \(\Sigma_a = (h, \sigma_a)\), for a ∈ [t], and outputs the PS signature:

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

where \(\ell_a\) is a Lagrange coefficient:

\[
\ell_a = \prod_{b \in [t], b \neq a} \left\{ \frac{0 - b}{a - b} \right\} \mod p.
\]
• **PSSwitchPoly.Verify(pk, Σ, \{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 \((h, \sigma)\), 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 \(sk = (z, y_1, \ldots, y_q)\). The secret key is reconstructed via interpolation:

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

which happens in the exponent of the element \(\sigma\):

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

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

- **PSSwitchPoly employs** the secret sharing scheme polyShamir, and the \(q\)-secret sharing scheme \(pwrShares_q\), to construct shares containing just two field elements. These shares are identified with the secret keys:

\[
sk_a = (sh_a^{\text{polyShamir}}, sh_a^{pwrShares_q}) = (z_a, y_a). \tag{44}
\]

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

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

which is the standard PolySig form, namely \(\Sigma_{\text{PolySig}} = (h, \sigma) = (h, h^{z + \sum_i m_i y_i})\).

- 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, \(\Sigma_{\text{PS}} = (h, h^{z + \sum_i m_i y_i})\). 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 \(pwrShares_q\).

- Because the partial signatures \(\sigma_a\) are PolySig signatures, the signing authorities’ secret keys only contain two field elements, \(sk_a = (z_a, y_a)\). In particular, these secret keys contain \(O(1)\) 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 $\text{pwrShares}_q$, 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 $\text{pwrShares}_q$ 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, $\text{sh}_a = y_a$, 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 $\text{pwrShares}_q$ 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 $\text{rltdShares}_q$ describes the more-general class of $q$-secret sharing protocols with related shares:

• $\text{rltdShares}_q$
  - Randomly draw $t$ field elements $y_a \xleftarrow{\$} \mathcal{M}_R$, 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 $\text{sh}_a = y_a$ 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 $\text{sh}_a = y_a$ to friends, from which both the friends and the original user can generate the full shares:

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

(47)

Depending on the form of $F$, the protocol may or may not be $K$-secure. Provided protocol $\text{rltdShares}_q$ 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 $sk_a = (z_a, y_a)$ 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 rltdSharesq can therefore be employed in place of Shamirq 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 pwrSecretq, 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 pwrSharesq, which leverages succinct shares containing $O(1)$ 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 pwrSharesq was shown to be $q$-secret sharing secure and $K$-private, such that each secret remains private within a set of cardinality $|\Delta_{i,q}|$ (i.e. the $i$-th secret is $|\Delta_{i,q}|$-private). Despite pwrSharesq, 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 ↔ 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


