
Threshold Signatures

Seminar Essay
by Marlena Müller

Research Group 
Codes and Cryptography

Preface

This essay was written as part of the seminar *Modern Cryptography* at Paderborn University in the 2024 summer term and it is graded (together with a presentation on the essay topic) with a grade of 1.0 (very good).

Changes to the original essay were made for this publication and described in the Changelog.

This work is licensed under a Creative Commons
“Attribution-ShareAlike 4.0 International” license.



Contents

1	General Signature Schemes	1
1.1	Properties of Signature Schemes	1
1.2	Security Definitions	1
1.3	BLS Signature Scheme	2
1.3.1	Pairings on Elliptic Curves	2
1.3.2	Construction and Correctness of the BLS Signature Scheme	3
2	Threshold Signature Schemes	3
2.1	Security Definitions	4
2.2	Generic Threshold Signature Scheme	5
2.2.1	Security Analysis	5
2.3	BLS Threshold Signature Scheme	6
2.3.1	Correctness of the BLS threshold Signature Scheme	7
2.3.2	Security Analysis	8
2.4	Robustness	9
2.5	Accountability	10
2.6	Privacy	12
	References	13

1 General Signature Schemes

Signature schemes are a fundamental cryptographic primitive that allow a party to sign a message in a way that allows anyone to verify that the message was signed by the party. Signature schemes are used in many applications such as secure communication, secure software updates, and secure voting. Signature schemes are used to ensure that a message was not tampered with or that the message was signed by the party that claims to have signed it.

The signature scheme is the digital equivalent of a handwritten signature, although more formal and with proven security properties.

1.1 Properties of Signature Schemes

Definition 1.1 (Signature Scheme) *A signature scheme is a tuple of three algorithms $(Gen, Sign, Verify)$ defined over a message space \mathcal{M} and a signature space \mathcal{S} :*

- $Gen(1^\lambda) \rightarrow (pk, sk)$: *The key generation algorithm takes a security parameter λ as input and outputs a public key pk and a private key sk .*
- $Sign(sk, m) \rightarrow \sigma$: *The signing algorithm takes a private key sk and a message $m \in \mathcal{M}$ as input and outputs a signature $\sigma \in \mathcal{S}$.*
- $Verify(pk, m, \sigma) \rightarrow \{\mathbf{accept}, \mathbf{reject}\}$: *The verification algorithm takes a public key pk , a message $m \in \mathcal{M}$, and a signature $\sigma \in \mathcal{S}$ as input and outputs a value indicating whether the signature is valid.*

A signature scheme $(Gen, Sign, Verify)$, that satisfies the following properties for every $m \in \mathcal{M}$ and every possible output of Gen is said to be correct:

$$\Pr[Verify(pk, m, Sign(sk, m)) = \mathbf{accept}] = 1$$

In this document we will assume that a signature scheme is correct unless otherwise stated.

1.2 Security Definitions

We require signature schemes to adhere to certain properties in order to provide robust security guarantees. In this section, we will define the security properties that a signature scheme must satisfy in order to be considered secure.

Intuitively the definitions below will guarantee, that for every secure signature scheme nobody who knows the public key and has signatures for arbitrary messages, can forge a signature for a new message.

Definition 1.2 (EUF-CMA Security Game) *For a signature scheme $(Gen, Sign, Verify)$ we define the Existential Unforgeability under Chosen Message Attack (EUF-CMA) security game between an adversary \mathcal{A} and a challenger \mathcal{C} as follows:*

- The challenger \mathcal{C} runs $\text{Gen}(1^\lambda)$ to generate a public key pk and a private key sk . The public key pk is given to the adversary \mathcal{A} .
- The adversary can query \mathcal{C} for $q \in \mathbb{N}^+$ signatures of Messages $m_1, m_2, \dots, m_q \in \mathcal{M}$ and receive the corresponding signatures $\sigma_1, \sigma_2, \dots, \sigma_q$.
- \mathcal{A} outputs a pair (m, σ) where $m \notin \{m_1, m_2, \dots, m_q\}$ (i.e., m is a new message).

The adversary \mathcal{A} wins the game if $\text{Verify}(pk, m, \sigma) = \text{accept}$.

Definition 1.3 (EUF-CMA Advantage) *The advantage of the adversary \mathcal{A} against S in the EUF-CMA security game is defined as:*

$$\text{EUF-CMAadv}(\mathcal{A}, S) = \Pr[\mathcal{A} \text{ wins the EUF-CMA game}]$$

Definition 1.4 (Secure Signature Scheme) *A signature scheme $(\text{Gen}, \text{Sign}, \text{Verify})$ is said to be secure if for all probabilistic polynomial-time adversaries \mathcal{A} , the advantage of \mathcal{A} in the EUF-CMA security game is negligible.*

1.3 BLS Signature Scheme

We will now introduce our first signature scheme, the BLS (BonehLynnShacham) signature scheme. The BLS signature scheme is a signature scheme based on pairings on elliptic curves. Later we will use the BLS Signature Scheme to construct a threshold signature scheme.

1.3.1 Pairings on Elliptic Curves

Elliptic curves are a special type of group. For this essay we just need the fact, that elliptic curves follow group laws and that we can define a bilinear map on elliptic curves.

Definition 1.5 *A pairing on elliptic curves is a map $e : G_1 \times G_2 \rightarrow G_T$ where G_1 and G_2 are groups defined over elliptic curves and G_T is a group defined over a finite field. The map e needs to satisfy the following properties:*

1. **Bilinearity:** For all $u, u' \in G_1, v, v' \in G_2$: $e(u \cdot u', v) = e(u, v) \cdot e(u', v)$ and $e(u, v \cdot v') = e(u, v) \cdot e(u, v')$.
2. **Non Degeneracy:** $e(g_1, g_2)$ is a generator of G_T where g_1 and g_2 are generators of G_1 and G_2 respectively.

For the BLS signature scheme, we need a corollary of the bilinearity being:

$$e(g_1^\alpha, g_2) = e(g_1, g_2)^\alpha = e(g_1, g_2^\alpha) \text{ for all } g_1 \in G_1, g_2 \in G_2 \text{ and } \alpha \in \mathbb{Z}_q.$$

1.3.2 Construction and Correctness of the BLS Signature Scheme

Now we will look into the details of the BLS signature scheme as presented in [BS]. The BLS signature scheme is defined over a group G of prime order q with a bilinear map $e : G \times G \rightarrow G_T$ where G_T is a group of prime order p and some hash function $H : \mathcal{M} \rightarrow G$.

- The key generation algorithm Gen takes 1^λ as input and generates a public key pk and a secret key sk with $\alpha \leftarrow \mathbb{Z}_q$ and $pk := g^\alpha$ and $sk := \alpha$.
- The signing algorithm Sign takes the secret key sk and a message m as input and outputs a signature $\sigma := H(m)^\alpha$.
- The verification algorithm $\text{Verify}_{\text{BLS}}$ takes the public key pk , a message m , and a signature σ as input and outputs **accept** if $e(\sigma, g) = e(H(m), pk)$ and **reject** otherwise.

We can easily see, that the BLS signature scheme is correct, as the verification algorithm checks if $e(\sigma, g) = e(H(m)^\alpha, g) = e(H(m), g)^\alpha = e(H(m), g^\alpha) = e(H(m), pk)$, so

$$\Pr[\text{Verify}_{\text{BLS}}(pk, m, \text{Sign}(sk, m)) = \text{accept}] = 1$$

holds for all outputs of the key generation algorithm Gen and all messages m .

2 Threshold Signature Schemes

Threshold signature schemes are a generalization of signature schemes that allow a group of parties to jointly sign a message. In a threshold signature scheme, a group of parties generate a public key and multiple private keys such that any subset of at least a predefined size of the parties can jointly sign a message. But less parties can not sign the message. The signature generated by the parties is a valid signature that can be verified by anyone using the public key [Sho00].

[BS] formally defines a threshold signature schemes $(\text{Gen}, \text{Sign}, \text{Verify}, \text{Comb})$ over a finite message space \mathcal{M} and a finite signature space Σ as follows:

- The key generation algorithm Gen takes 1^λ , N and t as input and generates a public key pk , a combiner key pk_c and N private keys sk_1, sk_2, \dots, sk_n . Here 1^λ is the security parameter, N is the number of parties and t is the threshold number of parties required to sign a message.
- The signing algorithm Sign takes a message m and one of the private keys sk_i as input and outputs a partial signature σ_i .
- The combiner algorithm Comb takes the combiner key pk_c , a message m , and a set of partial signatures $\{\sigma_j \mid j \in \mathcal{J}\}$ as input. $\mathcal{J} \subseteq \{1, \dots, N\}$ is the t -sized set containing the parties contributing to the signature. The algorithm outputs a signature σ , or **blame**(J^*) where J^* should be the set of parties that are blamed for the signature not being valid.

- The verification algorithm `Verify` takes a message m , a signature σ , and the public key pk as input and outputs either `accept` or `reject`.

As usual, the signature scheme is correct if and only if

$$\Pr[V(pk, m, C(pk_c, m, \mathcal{J}, \{\text{Sign}(sk_j, m) \mid j \in \mathcal{J}\})) = \text{accept}] = 1$$

holds for all outputs of the key generation algorithm `Gen`, all at least t -sized subsets $J \subseteq \{1, \dots, N\}$ and all messages m .

2.1 Security Definitions

The definitions presented in this section are based on the work of [BS]. These definitions will provide a formal framework for analyzing the security of threshold signature schemes.

Definition 2.1 (Threshold Signature Security Game) *For a threshold signature scheme $S = (\text{Gen}, \text{Sign}, \text{Verify}, \text{Comb})$ over (\mathcal{M}, Σ) we define the threshold signature security game between an adversary \mathcal{A} and a challenger \mathcal{C} as follows:*

- *The adversary sends poly-bounded N and t and an $\mathcal{L} \subset \{1, \dots, N\}$ of size $t - 1$ to the challenger.*
- *The challenger runs $\text{Gen}(1^\lambda, N, t)$ to generate a public key pk , a combiner key pk_c and N private keys sk_1, sk_2, \dots, sk_N . pk, pk_c and sk_i for $i \in \mathcal{L}$ are given to the adversary.*
- *The adversary can send signing-queries to \mathcal{C} with messages m_j and receives $\sigma_{i,j} = \text{Sign}(sk_i, m_j)$ for $i \in \{1, \dots, N\}$.*
- *\mathcal{A} outputs a pair $(m, \sigma) \in \mathcal{M} \times \Sigma$ where $m \notin \{m_1, m_2, \dots, m_q\}$ (i.e., m is a new message).*

The adversary \mathcal{A} wins the game if $\text{Verify}(pk, m, \sigma) = \text{accept}$.

Definition 2.2 *The advantage of the adversary \mathcal{A} against S in the threshold signature security game is defined as:*

$$TSadv(\mathcal{A}, S) = \Pr[\mathcal{A} \text{ wins the game}]$$

Definition 2.3 (Secure Threshold Signature Scheme) *A threshold signature scheme $(\text{Gen}, \text{Sign}, \text{Verify}, \text{Comb})$ is said to be secure if for all polynomial-time adversaries \mathcal{A} , the advantage of \mathcal{A} in the threshold signature security game is negligible.*

The above definition, is a strong security definition, as the adversary knows the secret keys of $t - 1$ parties and all information, the combiner gets, which practically means, that $t - 1$ parties and the combiner are compromised. Also this rules out, any signature scheme, where the combiner can reconstruct the secret, from the partial signatures.

2.2 Generic Threshold Signature Scheme

We can construct a simple threshold signature scheme from any signature scheme. The idea is to generate N key pairs (pk_i, sk_i) for $i = 1, 2, \dots, N$ and let each party i sign the message with their private key sk_i . The public key pk and the combining key pk_c are the concatenation of all public keys pk_i . Thus formally

$(pk, pk_c, sk_1, \dots, sk_N) \leftarrow \text{Gen}(1^\lambda, N, t)$ with $pk := (pk_1, pk_2, \dots, pk_N)$.

The sign algorithm works as defined for the threshold signature scheme:

$\sigma_i \leftarrow S(sk_i, m)$.

The combiner algorithm works as follows: $C(pk_c, m, \mathcal{J}, \{\sigma_j \mid j \in \mathcal{J}\})$ outputs the concatenation of all partial signatures σ_j for $j \in \mathcal{J}$ and \mathcal{J} , if $|\mathcal{J}| = t$ and

$\forall j \in \mathcal{J} : \text{Verify}(pk_j, \sigma_j) = \text{accept}$. Otherwise it outputs $\text{blame}(J^*)$ where J^* is the set of parties which output invalid signatures σ_j .

The verification algorithm works as follows: $V(pk, m, \sigma)$ with $pk = (pk_1, \dots, pk_N)$ and $\sigma = (\{\sigma_j \mid j \in \mathcal{J}\}, \mathcal{J})$ outputs **accept** if $\text{Verify}(pk_j, \sigma_j) = \text{accept}$ for all $j \in \mathcal{J}$ and $|\mathcal{J}| = t$. Otherwise it outputs **reject**.

Later we will see that this simple threshold signature scheme is secure if the underlying signature scheme is secure.

The downside is, that the public key size is N times the size of the public key of the underlying signature scheme and the signature size is t times the size of the signature of the underlying signature scheme. This is not practical for large N and t , as the transfer time and storage requirements grow in practical scenarios.

2.2.1 Security Analysis

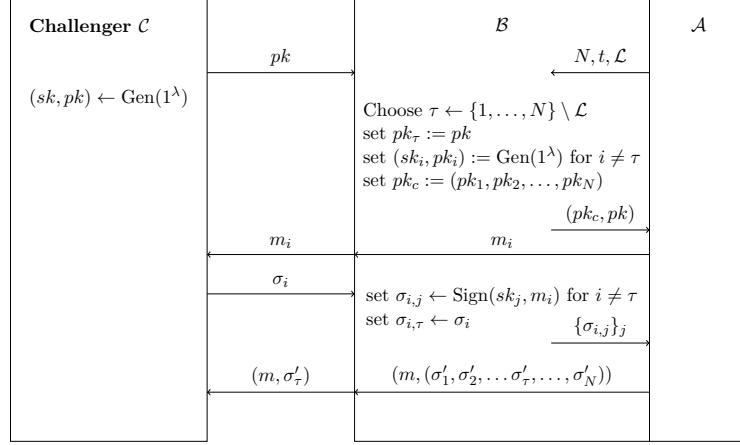
Now we will look into the security of the generic threshold signature scheme.

Theorem 2.4 *The generic threshold signature scheme is secure if the underlying signature scheme is secure.*

Proof. To prove that the generic threshold signature scheme is secure, we will show that using an adversary \mathcal{A} that wins the threshold signature security game with non-negligible probability, we can construct an adversary \mathcal{B} that wins the signature security game with non-negligible probability.

For this, we will construct \mathcal{B} as seen in Figure 1, as a simple wrapper around \mathcal{A} . \mathcal{B} wants to forge a signature for a message m . For this, \mathcal{B} simulates the threshold signature security game for \mathcal{A} . If \mathcal{A} wins the threshold signature security game for \mathcal{B} it has output a valid threshold signature for m . This signature contains t partial signatures, with at least 1 partial signature for which \mathcal{A} does not know the secret key. This is with probability of at least $\frac{1}{N-t+1}$, the signature corresponding to party τ , as τ is chosen uniformly at random and \mathcal{A} does not know the secret key of τ . If this happens, \mathcal{B} outputs the partial signature of τ as the forged signature for m , which is a valid signature for m .

If σ_τ is not part of the threshold signature, \mathcal{B} just outputs a random signature for m . So $\Pr[\mathcal{B} \text{ wins the game}] \geq \frac{1}{N-t+1} \Pr[\mathcal{A} \text{ wins the game}]$.


 Figure 1: Construction of \mathcal{B} from \mathcal{A}

As \mathcal{A} wins with non-negligible probability, \mathcal{B} wins with non-negligible probability as well, because N, t are polynomial bounded.

Because \mathcal{B} is a simple wrapper around \mathcal{A} , only computing efficiently computable functions, \mathcal{B} only generating polynomial many key-pairs and partial signatures, \mathcal{B} is polynomial-time, if \mathcal{A} is polynomial-time.

From this we can conclude, that the generic threshold signature scheme is secure, if the underlying signature scheme is secure. \square

2.3 BLS Threshold Signature Scheme

The BLS threshold signature scheme as seen in [BS] is based on the construction of the BLS signature scheme in Section 1.3.2.

To understand the BLS threshold signature scheme, we first need to look into the Shamir secret sharing scheme [Sha79]. The goal of this scheme is to split a secret into multiple shares, such that using a subset of at least a predefined size of shares can be used to reconstruct the secret.

Definition 2.5 (Shamir Secret Sharing Scheme) *The Shamir secret sharing scheme for sharing a secret between N parties such that any subset of at least t parties can reconstruct the secret works as follows:*

- The dealer gets a secret $\alpha \in \mathbb{Z}_q$ and chooses a polynomial $f(x) = \alpha + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1}$ with $a_i \leftarrow \mathbb{Z}_q$ for $i = 1, 2, \dots, t-1$.
- The dealer computes N shares $\alpha_1, \alpha_2, \dots, \alpha_N$ with $\alpha_i = f(i)$
- The dealer gives each party i the share α_i .

For reconstructing the secret α from a set \mathcal{J} of at least t parties, the parties compute: $f(x) = \sum_{i \in \mathcal{J}} \alpha_i \cdot \prod_{j \in \mathcal{J} \setminus \{i\}} \frac{x-j}{i-j}$ and can then reconstruct the secret α by calculating $f(0)$.

This construction works, because we can reconstruct a polynomial of degree $t - 1$ from t points, as stated in the fundamental theorem of algebra.

Using this we can now define the BLS threshold signature scheme:

The key generation algorithm Gen generates a public key pk , a combiner key pk_c and N private keys sk_1, sk_2, \dots, sk_N as follows:

- choose $\alpha \leftarrow \mathbb{Z}_q$ and set $pk := g^\alpha$
- use the secret sharing scheme to generate N shares $\alpha_1, \alpha_2, \dots, \alpha_N \in \mathbb{Z}_q$ of α
- set $sk_i := \alpha_i$ for $i = 1, 2, \dots, N$
- set $pk_c := (pk_1, pk_2, \dots, pk_N)$ with $pk_i := g^{\alpha_i}$

The signing algorithm Sign' for generating a partial signature works the same as the signing algorithm of the BLS signature scheme: output $\sigma_i := H(m)^{\alpha_i}$.

The combiner algorithm Comb' for combining the partial signatures with input pk_c, m, \mathcal{J} and $\{\sigma_j \mid j \in \mathcal{J}\}$ works as follows:

- check if all σ_j are valid signatures for the public key pk_j for $j \in \mathcal{J}$ and if $|\mathcal{J}| = t$
- if not output $\text{blame}(\mathcal{J}^*)$ with $\mathcal{J}^* = \{j \mid j \in \mathcal{J} \wedge \text{Verify}_{\text{BLS}}(pk_j, m, \sigma_j) = \text{reject}\}$
- otherwise output $\sigma := \prod_{j \in \mathcal{J}} (\sigma_j)^{\lambda_j}$ with $\lambda_j = \prod_{k \in \mathcal{J} \setminus \{j\}} \frac{k}{k-j}$.

The verification algorithm Verify' works the same way, as $\text{Verify}_{\text{BLS}}$ from section 1.3.2. One big advantage of the BLS threshold signature scheme is, that the public key size is the same as the public key size of the BLS signature scheme and the signature size is the same as the signature size of the BLS signature scheme, so they do not grow with the number of parties.

Also in comparison to the generic threshold signature scheme, where the t is known to anyone, only the combiner needs to know t in the BLS threshold signature scheme, and t can not be derived from any public information.

The BLS threshold signature scheme is correct if the underlying BLS signature scheme is correct. To prove this, we first need to look into the Shamir secret sharing scheme.

2.3.1 Correctness of the BLS threshold Signature Scheme

The BLS threshold signature scheme is correct, if the underlying BLS signature scheme is correct and the Shamir secret sharing scheme is correct.

Let \mathcal{J} be a subset of t parties, $\sigma_i = H(m)^{\alpha_i}$ for $i \in \mathcal{J}$ be the (correct) partial signatures of the parties in \mathcal{J} and m an arbitrary message.

As Verify' checks, if $e(\sigma, g) = e(H(m), pk)$, we need to show that $e(\sigma, g) = e(H(m), pk)$ holds for the output of Comb' . For this, we will show that $\sigma = \prod_{j \in \mathcal{J}} (\sigma_j)^{\lambda_j} = H(m)^\alpha$.

$$\begin{aligned}
 \sigma &= \prod_{j \in \mathcal{J}} (\sigma_j)^{\lambda_j} \\
 &= \prod_{j \in \mathcal{J}} (H(m)^{a_j})^{\lambda_j} \\
 &= \prod_{j \in \mathcal{J}} H(m)^{a_j \lambda_j} \\
 &= H(m)^{\sum_{j \in \mathcal{J}} a_j \lambda_j} \\
 &= H(m)^{\sum_{j \in \mathcal{J}} a_j \prod_{k \in \mathcal{J} \setminus \{j\}} \frac{k}{k-j}} \\
 &= H(m)^{\sum_{j \in \mathcal{J}} a_j \prod_{k \in \mathcal{J} \setminus \{j\}} \frac{0-k}{j-k}} \\
 &= H(m)^{f(0)} \\
 &= H(m)^\alpha
 \end{aligned}$$

So we have $e(\sigma, g) = e(H(m)^\alpha, g) = e(H(m), g)^\alpha = e(H(m), g^\alpha) = e(H(m), pk)$, so

$$\Pr[\text{Verify}'(pk, m, \text{Comb}'(pk_c, m, \mathcal{J}, \{\text{Sign}'(sk_j, m) \mid j \in \mathcal{J}\})) = \text{accept}] = 1$$

holds for all outputs of the key generation algorithm Gen, all t -sized subsets $\mathcal{J} \subseteq \{1, \dots, N\}$ and all messages m . Thus the BLS threshold signature scheme is correct.

2.3.2 Security Analysis

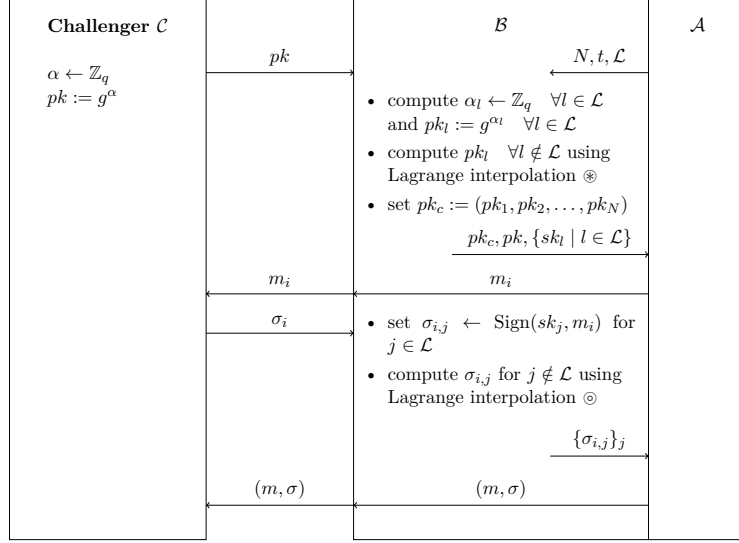
We will now look into the security of the BLS threshold signature scheme as seen in [BS].

Theorem 2.6 *The BLS threshold signature scheme is secure if the underlying BLS signature scheme is secure.*

Proof. To prove, that the BLS threshold signature scheme is secure, we will show that if an adversary \mathcal{A} wins the threshold signature security game, we can construct an adversary \mathcal{B} that wins the BLS signature security game with the same advantage. For this, we will construct \mathcal{B} as seen in Figure 2, as a simple wrapper around \mathcal{A} . Step \otimes works, as there is only one polynomial of degree $t-1$, that satisfies $f(i) = \alpha_i$ for $i \in \mathcal{L}$ and $f(0) = \alpha$. We can obtain this polynomial f by Lagrange interpolation and can now compute $f(l) = pk_l$ for $l \notin \mathcal{L}$.

Step \odot works, very similar. We know that $\{\sigma_{i,j}\}_j$ are points on a polynomial of degree $t-1$. Again we know t points (namely $f(0) = \sigma$, $f(l) = H(m)^{sk_l}$ for $l \in \mathcal{L}$), so we can reconstruct the polynomial and compute $\sigma_{i,j}$ for $j \notin \mathcal{L}$.

Using these steps \mathcal{B} can simulate the BLS signature security game for \mathcal{A} . If \mathcal{A} wins the threshold signature security game, \mathcal{B} wins the BLS signature security game as well, as


 Figure 2: Construction of \mathcal{B} from \mathcal{A}

both goals of the game are equivalent. Thus the advantage of \mathcal{B} is the same as the advantage of \mathcal{A} .

Because \mathcal{B} is a simple wrapper around \mathcal{A} , only computing efficiently computable functions, \mathcal{B} is polynomial-time, if \mathcal{A} is polynomial-time. From this we can conclude, that the BLS threshold signature scheme is secure, if the underlying BLS signature scheme is secure. \square

2.4 Robustness

We often want to make stronger assumptions about our threshold signature schemes, than just security. One of these assumptions is robustness.

For understanding robustness, we first need to look at the concept of “accurate blaming” and “consistency”.

Definition 2.7 (Accurate Blaming) *A threshold signature scheme $S = (Gen, Sign, Verify, Comb)$ over (\mathcal{M}, Σ) is said to have accurate blaming if for all outputs of the algorithm $Gen(1^\lambda, N, t)$ for all $m \in \mathcal{M}$ and all $J \subseteq \{1, \dots, N\}$ with $|J| = t$ and all $\sigma \in \Sigma$:*

$$Pr[Sign(sk_j, m) = \sigma_j \mid Comb(pk_c, m, J, \{\sigma_j \mid j \in J\}) = \text{blame}(J^*)] = 0 \text{ for all } j \in J^*$$

Intuitively this means, that J^* is the set of parties that are responsible for the signature not being valid. If a party is blamed, its output was an invalid signature. We want our threshold signature scheme to have accurate blaming, as we want to know, which parties are responsible for an invalid signature. Also, when more than t parties are willing to sign a message, and there are some parties that output invalid signatures, we can ignore the parties in J^* and use the remaining partial signatures to

construct a valid signature. If necessary this process is repeated until we have a valid signature or no parties are left.

Definition 2.8 (Consistent Threshold Signature Game) *For a threshold signature scheme $S = (\text{Gen}, \text{Sign}, \text{Verify}, \text{Comb})$ over (\mathcal{M}, Σ) we define the consistent threshold signature game between an adversary \mathcal{A} and a challenger \mathcal{C} as follows:*

- *The adversary sends poly-bounded N and t to the challenger.*
- *The challenger runs the algorithm $\text{Gen}(1^\lambda, N, t)$ and sends the complete output $(pk, pk_c, sk_1, \dots, sk_N)$ to the adversary.*
- *The adversary outputs a \mathcal{J} of size t and σ_j for $j \in \mathcal{J}$.*

The adversary \mathcal{A} wins the game if $\text{Comb}(pk_c, m, \mathcal{J}, \{\sigma_j \mid j \in \mathcal{J}\}) = \sigma$ holds with $\text{Verify}(pk, m, \sigma) = \text{reject}$.

Intuitively an adversary wins this game, if it can compute a series of (corrupted) partial signatures, such that the combiner outputs an invalid signature.

Definition 2.9 *Consistent Threshold Signature Scheme A threshold signature scheme $S = (\text{Gen}, \text{Sign}, \text{Verify}, \text{Comb})$ over (\mathcal{M}, Σ) is said to be consistent, if for all polynomial-time adversaries \mathcal{A} the probability that \mathcal{A} wins the consistent threshold signature game is negligible.*

Definition 2.10 *Robust Threshold Signature Scheme A threshold signature scheme $S = (\text{Gen}, \text{Sign}, \text{Verify}, \text{Comb})$ over (\mathcal{M}, Σ) is said to be robust, if it has accurate blaming and is consistent.*

A robust threshold signature scheme is beneficial in applications where we want that every signature, that is output by the combiner, is valid (due to consistency) and we can identify the parties that try to attack the signing process (due to accurate blaming). This is needed for applications where not every signee is trusted and we want to identify malicious parties. For example this is used for signatures in cryptocurrencies like Bitcoin [WNR20].

2.5 Accountability

We can also define so called accountable threshold signature schemes. The goal of accountable threshold signature schemes is to make it possible to identify the parties that participated in the signing of a message.

Definition 2.11 (Accountable Threshold Signature Scheme) *An accountable threshold signature scheme $S = (\text{Gen}, \text{Sign}, \text{Verify}, \text{Comb}, \text{Trace})$ over (\mathcal{M}, Σ) with a classical threshold signature scheme $(\text{Gen}, \text{Sign}, \text{Verify}, \text{Comb})$ with Trace defined as follows:*

- *Trace is an efficient, probabilistic algorithm that takes the public key pk , a message m and a signature σ as input and outputs a t -sized set \mathcal{J} of parties that participated in the signing of the message m with the signature σ .*

- We require correctness for the algorithm *Trace*, i.e. for all outputs of $Gen(1^\lambda, N, t)$, all $m \in \mathcal{M}$ and all $\mathcal{J} \subset \{1, \dots, N\}$ with $|\mathcal{J}| = t$:

$$Pr[\text{Trace}(pk, m, \text{Comb}(pk_c, m, \mathcal{J}, \{\text{Sign}(sk_j, m) \mid j \in \mathcal{J}\})) = \mathcal{J}] = 1$$

Definition 2.12 (Accountable Threshold Signature Game) *For an accountable threshold signature scheme $S = (Gen, Sign, Verify, Comb, Trace)$ over (\mathcal{M}, Σ) we define the accountable threshold signature game between an adversary \mathcal{A} and a challenger \mathcal{C} as follows:*

- The adversary sends poly-bounded N and t and a victim v to the challenger.
- The challenger runs $Gen(1^\lambda, N, t)$ and sends $(pk, pk_c, \{sk_i \mid i \neq v\})$ to the adversary.
- The adversary sends signing queries to the challenger with messages m_i and receives $\sigma_{i,j} = \text{Sign}(sk_v, m_i)$
- The adversary outputs a message m and a signature σ .

The adversary \mathcal{A} wins the game if:

- $Verify(pk, m, \sigma) = \text{accept}$,
- $v \in \text{Trace}(pk, m, \sigma)$ and
- $\forall i : m \neq m_i$

Definition 2.13 (Accountable Threshold Signature Advantage) *We define the advantage of the adversary \mathcal{A} against S in the accountable threshold signature game as:*

$$AccAdv(\mathcal{A}, S) = Pr[\mathcal{A} \text{ wins the game}]$$

Definition 2.14 (Accountable Threshold Signature Scheme) *A threshold signature scheme $S = (Gen, Sign, Verify, Comb, Trace)$ over (\mathcal{M}, Σ) is said to be accountable, if for all probabilistic polynomial-time adversaries \mathcal{A} the advantage of \mathcal{A} , in the accountable threshold signature game, is negligible.*

The construction of the generic threshold signature scheme yields accountability, if the underlying signature scheme is secure. Intuitively this is, because the adversary needs to output a signature σ containing the partial signature of the victim v . Because the underlying signature scheme is secure, the adversary can only output the partial signature of v with negligible probability.

The BLS threshold signature scheme is not accountable.

2.6 Privacy

Another property, a threshold signature schemes can have, is Privacy.

Intuitively a threshold signature scheme is private, if we can not deduct N , t or the parties that signed a message from the public information (i.e. the public key and the signature).

Definition 2.15 (Privacy Game) *For a threshold signature scheme $S = (Gen, Sign, Verify, Comb)$ and a non-threshold signature scheme $S' = (Gen', Sign', Verify')$ we define the game $b \in \{0, 1\}$ between an adversary \mathcal{A} and a challenger \mathcal{C} as follows:*

- *The adversary sends poly-bounded N and t with $t < N$ to the challenger.*
- *If $b = 0$ the challenger runs $(pk, pk_c, sk_1, \dots, sk_n) \leftarrow Gen(1^\lambda, N, t)$, if $b = 1$ the challenger runs $(pk, sk) \leftarrow Gen'(1^\lambda)$.
The challenger sends the public key pk to the adversary.*
- *The adversary sends queries (m_i, \mathcal{J}_i) with $m_i \in \mathcal{M}$ and $\mathcal{J}_i \subseteq \{1, \dots, N\}$ with $|\mathcal{J}_i| = t$ to the challenger.*
- *The challenger answers the queries with the corresponding signatures*

$$\sigma_i = \begin{cases} Comb(pk_c, m_i, \mathcal{J}_i, \{Sign(sk_j, m_i) \mid j \in \mathcal{J}_i\}) & \text{if } b = 0 \\ Sign'(sk, m_i) & \text{if } b = 1 \end{cases}$$

- *The adversary outputs a guess b' .*

The advantage of the adversary \mathcal{A} in the privacy game is defined as:

$$PrivAdv(\mathcal{A}, S, S') = |Pr[W_0] - Pr[W_1]|$$

with W_b being the event, that \mathcal{A} outputs 1 in the game b .

Definition 2.16 (Private Threshold Signature Scheme) *A threshold signature scheme $S = (Gen, Sign, Verify, Comb)$ is said to be private, if for all polynomial-time adversaries \mathcal{A} the advantage of \mathcal{A} in the privacy game is negligible.*

Theorem 2.17 *The generic threshold signature scheme is not private.*

Intuitively we can see that the generic threshold signature scheme is not private, as the adversary can just check, if the partial signatures are part of the threshold signature. If so, the adversary will output 1, otherwise 0. This will lead to $Pr[W_0] = 1$ and $Pr[W_1] = 0$ and an advantage of 1.

Theorem 2.18 *The BLS threshold signature scheme is private.*

We will give a prove intuition by showing, that the output of the BLS threshold signature scheme is indistinguishable from the output of the BLS signature scheme. In both schemes an α gets chosen as the secret. In the threshold BLS scheme, the secret gets shared using the Shamir secret sharing scheme. The public key is $pk = g^\alpha$ in both schemes.

When signing a message m in the threshold BLS scheme $H(m)^\alpha$ gets computed with Lagrange interpolation. In the non-threshold BLS scheme $H(m)^\alpha$ gets computed as well.

So in both cases the public key and the signature are exactly the same. Thus the adversary can not distinguish between the two schemes and only behave fully random, which yields $\Pr[W_0] = \Pr[W_1]$ and $\text{PrivAdv}(\mathcal{A}, S, S') = 0$. Because the advantage is negligible, the threshold BLS signature is private.

References

- [BS] Dan Boneh and Victor Shoup. A Graduate Course in Applied Cryptography.
- [Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
- [Sho00] Victor Shoup. Practical threshold signatures. In *Advances in Cryptology EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*, pages 207–220. Springer, 2000.
- [WNR20] Pieter Wuille, Jonas Nick, and Tim Ruffing. *Schnorr Signatures for secp256k1*, 2020.

Changelog

The following changes were made to the document, after the initial submission:

- Added the preface
- Added the license
- Changed the University Logo to the new Corporate Design
- Used the english version of the Research Group name
- Corrected definition 2.8, where it falsely stated that the adversary wins the game, if (among other conditions) $\text{Verify}(pk, m, \sigma) = \text{accept}$ holds. This condition was changed to $\text{Verify}(pk, m, \sigma) = \text{reject}$.