

Threshold Signature for Privacy-preserving Blockchain

Citation

Ricci, S., Dzurenda, P., Casanova-Marqués, R., Cika, P., 2022. Threshold Signature for Privacy-preserving Blockchain, in: Business Process Management: Blockchain, Robotic Process Automation, and Central and Eastern Europe Forum, Springer, pp. 100–115.

Year

2022

Version

Publisher's PDF (version of record)

Link to publication

https://link.springer.com/chapter/10.1007/978-3-031-16168-1_7

Published in

Springer, Cham

DOI

https://doi.org/10.1007/978-3-031-16168-1_7

License

This publication is copyrighted. You may download, display and print it for Your own personal use. Commercial use is prohibited.

Take down policy

If you believe that this document breaches copyright, please contact the authors, and we will investigate your claim.

BibTex entry

```
@inproceedings{BUT178490,  
  author="Sara {Ricci} and Petr {Dzurenda} and Raúl {Casanova-Marqués}  
and Petr {Číka}",  
  title="Threshold Signature for Privacy-preserving Blockchain",  
  booktitle="Business Process Management: Blockchain, Robotic Process  
Automation, and Central and Eastern Europe Forum",  
  doi="10.1007/978-3-031-16168-1_7",  
  year="2022",  
  month="august",  
  pages="100--115",  
}
```



Threshold Signature for Privacy-Preserving Blockchain

Sara Ricci¹(✉) , Petr Dzurenda¹ , Raúl Casanova-Marqués^{1,2} ,
and Petr Cika¹

¹ FEEC, Department of Telecommunications, Brno University of Technology, Brno,
Czech Republic

{ricci,dzurenda,casanova,cika}@vut.cz

² Institute of New Imaging Technologies, Universitat Jaume I, Castellón, Spain
<https://axe.vut.cz/>

Abstract. Threshold signatures received renewed interest in recent years due to their practical applicability to Blockchain technology. In this article, we propose a novel (n, t) -threshold signature scheme suitable for increasing security and privacy in Blockchain technology. Our scheme allows splitting a Blockchain wallet into multiple devices so that a threshold of them is needed for signing. This increases the security of the transactions, e.g., more devices need to be compromised to recover the key and permits, and the privacy, e.g., the signing is made anonymously on behalf of the group of users sharing the Blockchain wallet. Our experimental results show that the signing algorithm requires less than 10 ms in the cases of 10 devices involved.

Keywords: Threshold signature · Multi-signature · Blockchain · Secret sharing · Paillier cryptosystem · Schnorr protocol

1 Introduction

Threshold signatures belong to distribute signature family where a threshold number of participants have to cooperate to issue a signature that can be verified by a single public key. Even if they have been studied for a long time [13, 22], these signatures received renewed interest in recent years due to their practical applicability to Blockchain technology and electronic transactions, including cryptocurrencies such as Bitcoin [3].

A typical Blockchain consists of two parts: 1) a consensus mechanism for delegating the creation of new blocks including user transaction and 2) a signature scheme for user transactions verification [24]. A standard transaction, namely a single-signature transaction, involves only one private key, which is managed by a single device. On the contrary, a multi-signature transaction involves at least two keys that can be stored in different devices. This approach can bring several new beneficial features [17], e.g., 1) **increased security**: splitting the wallet keys between more user devices reduces the risk of compromising the wallet.

In fact, a malware is unlikely to infect them all. 2) **Joint accounts**: the transactions require the signatures of multiple users before the funds can be transferred, and 3) **wallet key backup**: if one key is lost in a “2-of-3” wallet, then the other two keys can be used to retrieve the wallet.

It is important to notice that Bitcoin already supports multi-signature transactions. These multi-signature wallets consist of several regular Bitcoin addresses. However, this approach has several privacy issues such as 1) several wallets of the same owner are publicly known, 2) the signing threshold and signer’s identity are also known, and 3) the size of the transaction grows with the number of wallet owners and signers. Threshold signatures can help reduce the amount of data stored in the Blockchain and solve privacy issues by compressing the signatures together while keeping verification still possible.

In a (n, t) -threshold signature, n parties can jointly generate a single public key from their n private shares of the key. The key can be used to securely sign messages if and only if t parties collaborate in the signing process. Moreover, no group of $t-1$ colluding parties should be able to recover the secret key. Threshold signatures are mainly based on RSA [12, 33] and Elliptic Curve Digital Signature Algorithm (ECDSA) signatures [9, 18, 20, 26, 27]. In the ECDSA signatures group, we can split the signatures in the $(2, 2)$ -threshold variant [7, 14, 25] and the more general (n, t) -threshold case [6, 8, 10, 11, 15, 18, 19, 26]. Note that (n, n) -threshold signatures are a particular case of (n, t) -threshold ones. At the moment, the most efficient threshold signature schemes rely on pairing-based cryptography [4, 5]. These schemes can perform signing operations in a single round among participants whereas the best non-pairing-based threshold schemes require multiple rounds of interaction during signing operations. These signatures are based on Schnorr’s protocol. For instance, [21, 34] require at least three rounds of communication during signing operations whereas FROST threshold signatures [23] needs only two rounds. Even if having more communication overload, these latter schemes guarantee robustness as a main feature, i.e., if any participant misbehaves, honest participants can detect this misbehavior, disqualify the misbehaving participant, and produce a signature as long as the threshold number of honest parties is achieved.

1.1 Contribution and Paper Structure

In this work, we proposed a novel (n, t) -threshold signature scheme suitable for increasing security and privacy in Blockchain technology. To do so, we provide a solution on how to securely split a Blockchain wallet between more devices. These devices can be held by a single user (i.e., it increases security, since more user devices are needed to sign Blockchain transactions) or by several collaborative users (i.e., it increases privacy, since the signing is made anonymously on behalf of the group of users sharing the Blockchain wallet). Our scheme is built on provable secure cryptographic primitives such as Schnorr signature [31], Pailler cryptosystem [29], and Shamir secret sharing scheme [32]. Furthermore, we implement our proposal with promising experimental results on a single board

computer using ARM Cortex-A72 processor widely used in the internet of things environment.

The paper is organized as follows. Section 2 outlines the used notation and cryptographic primitives used in our proposal. Section 3 introduces our (n, t) -threshold signature scheme. Section 4 presents the security analysis of our proposal. Section 5 shows the possible deployment of threshold signatures to Blockchain technology. Section 6 presents our experimental results. In the last section, we conclude this work.

2 Cryptographic Preliminaries

In this section, we introduce used notation and cryptographic primitives. From now on, the symbol “.” means “such that”, “ $|x|$ ” is the bitlength of x , and “ $||$ ” denotes the concatenation of two binary strings. We write $a \in_R A$ when a is sampled uniformly at random from A . Let \mathbb{G} be a additive cyclic group generated by elliptic curve E over final field \mathbb{F}_p and base point $g \in E(\mathbb{F}_p)$ of prime order q_{EC} , where $|q_{EC}| = \kappa$ and κ is a security parameter. A secure hash function is denoted as $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$, where κ is a security parameter. We describe the Proof of Knowledge (PK) and the Signature of Knowledge (SK) protocols using the notation introduced by Camenisch and Stadler. In particular, the protocol for proving the knowledge of discrete logarithm of c with respect to g is denoted as $\text{PK}\{\alpha : c = g^\alpha\}$ and the protocol for proving the knowledge of discrete logarithm of c with respect to g and message m is denoted as $\text{SK}\{\alpha : c = g^\alpha\}(m)$.

2.1 Schnorr Signature

Schnorr signature [31] is a digital signature scheme known for its simplicity, efficiency, and short signatures. It is based on the PK concept and it is frequently used in many cryptosystems, including privacy-enhancing schemes, such as group signatures, ring signatures, and attribute-based credentials. Using this scheme, the prover proves his/her statement on knowledge of a discrete logarithm (i.e., secret key $sk : pk = g^{sk}$) with respect to public parameters \mathbb{G}, g, q, pk . In contrast to the Schnorr identification protocol [31] which is a interactive 3-way protocol, the Schnorr signature scheme is non-interactive.

The scheme is depicted in Fig. 1. The prover commits a random number r , computes a challenge e by using a secure hash function \mathcal{H} , and finally responds by the proof z on the challenge e , secret key sk and message m .

Furthermore, Schnorr signatures have linear characteristics as shown in [16, 28]. This linearity property of Schnorr signatures can be used to construct a multi-signature, see Eq. 1 describing case 2-of-2 multi-signature.

$$\begin{aligned}
 pk &= pk_A * pk_B = g^{sk_A} * g^{sk_B} \\
 c &= c_A * c_B = g^{r_A} * g^{r_B} \\
 e &= \mathcal{H}(c||m) \\
 z &= z_A + z_B = (r_A - e * sk_A) + (r_B - e * sk_B)
 \end{aligned} \tag{1}$$

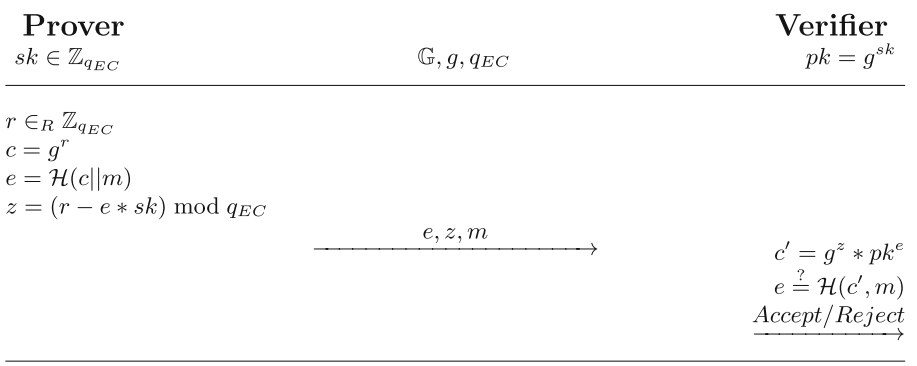


Fig. 1. Schnorr's signature of knowledge of discrete logarithm $SK\{sk : pk = g^{sk}\}(m)$.

The verification of 2-of-2 multi-signature is depicted in Eq. 2.

$$c' = g^z * pk^e = g^{(r_A - e * sk_A) + (r_B - e * sk_B)} * (g^{sk_A} * g^{sk_B})^e = g^{r_A} * g^{r_B} \quad (2)$$

2.2 Shamir Secret Sharing Scheme

Shamir secret share scheme [32] is a well-known (n, t) -threshold scheme where n denotes the number of shares involved and t the number of shares needed to reconstruct the secret k . This scheme is based on 1) unique polynomial property, i.e., there exists a unique $t-1$ -th degree polynomial that passes through t points in the plane, and 2) interpolation problem.

Let k be the secret, where k is in a field \mathbb{F}_q . The scheme involves a dealer who owns a secret and a set of n parties. The dealer chooses $t-1$ random elements a_1, \dots, a_{t-1} from \mathbb{F}_q independently with uniform distribution and defines a polynomial $P(x) = k + \sum_{i=1}^{t-1} a_i x^i$. The share of party j is the evaluation of the polynomial $\beta_j = P(\alpha_j)$, that is the pair (α_j, β_j) . The secret can be recovered with the following formula:

$$k = P(0) = \sum_{j=1}^t \beta_j * \prod_{m=1, m \neq j}^t \frac{\alpha_m}{\alpha_m - \alpha_j}. \quad (3)$$

Shamir secret sharing scheme has both properties [2]: 1) **Correctness** the secret k can be reconstructed by any authorized set of parties, and 2) **Perfect Privacy** every unauthorized set cannot learn anything about the secret (in the information theoretic sense) from their shares.

2.3 Paillier Cryptosystem

Paillier cryptosystem [29] is a probabilistic public-key algorithm for asymmetric encryption. This scheme runs in a RSA-modulo where P, Q are two large primes

of equal length. Figure 2 depicts its subroutines, namely **Keygen**, **Enc**, and **Dec** that states for key generation, encryption and decryption protocols, respectively.

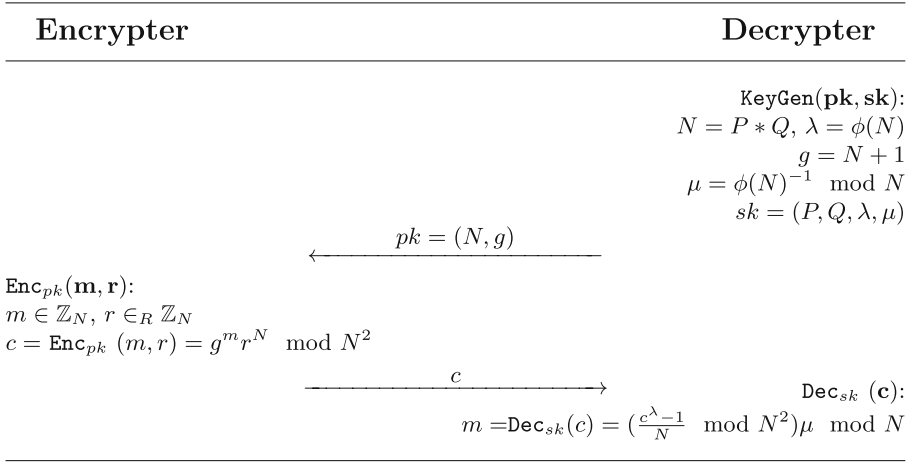


Fig. 2. Paillier cryptosystem.

Paillier scheme has additive homomorphic properties, i.e., allows doing additive operations on ciphertexts without corrupting the result. In particular, we will use the following two properties:

- $\text{Enc}_{pk}(m_1, r_1) * \text{Enc}_{pk}(m_2, r_2) = \text{Enc}_{pk}(m_1 + m_2, r)$,
- $\text{Enc}_{pk}(m_1, r)^{m_2} = \text{Enc}_{pk}(m_1 * m_2, r)$,

where m_1 and m_2 are two messages in \mathbb{Z}_N and r, r_1 , and r_2 are the random noises.

3 Proposed (n,t)-Threshold Scheme

In this section, we define the algorithms and entities of our (n,t)-threshold protocol. It employs two parties:

- **Signer**: jointly generates the signature in collaboration with other co-signers. The signer is typically a user which is accessing online services, such as cryptocurrency payments. The signing key (i.e., Blockchain Wallet secret key) is split and stored on different user devices such as smartphones, wearables, microcontrollers, and personal computers. In particular, a **Main Device** (MD) represents a user device with an activated signing mode. This mode is used by a signer when he/she wants to sign a Blockchain transaction. The main task

of the MD is to initiate all events, i.e., communication with Blockchain, creation of user transactions, and generation of a signature on the transaction. To do so, the MD has to run the signing protocol jointly with the co-signers' devices. Moreover, a **Secondary Device** (SD) represents a user device with activated co-signing mode. This mode is used by co-signers when they want to join a signing process of a Blockchain transaction initiated by the MD.

- **Verifier**: is a Blockchain node receiving and validating the user transactions. Among others, it checks the validity of the signature in the transaction over the Blockchain Wallet public key. The Verifier can be represented by a powerful computer as well as a computationally less powerful device such as a single-board computer or microcontroller.

The (n,t) -threshold signature scheme consists of the following three algorithms: 1) **Setup**, 2) **Signing**, and 3) **Verifying** that are described in Sects. 3.1, 3.2, and 3.3, respectively. Our (n,t) -threshold signature scheme needs that t out of n authorized signers collaborate to generate the signature. Let D_j be a signer's device performing the signature where $j = 1, \dots, n$. Each D_j owns its signing secret key k_j .

3.1 Setup Algorithm

Our scheme is based on Shamir protocol [32] and, therefore, requires that n signers agree on a polynomial $f(x)$ of degree $t-1$ that has $sk = \sum k_i$ as constant term, please see Sect. 2 for more details. Note that sk is the secret key used in the signature. We consider a polynomial with the following structure:

$$f(x) = (d_{t-1}^{(1)} + \dots + d_{t-1}^{(n)})x^{t-1} + \dots + (d_1^{(1)} + \dots + d_1^{(n)})x + sk, \quad (4)$$

where $d_j^{(i)}$ is privately generated by D_i for $i = 1, \dots, n$ and $j = 1, \dots, t-1$.

By using Paillier encryption [29], the polynomial $f(x)$ is evaluated in n points without disclosing its coefficients, i.e., the polynomial is not “built” and shared among the devices but only evaluated. Accordingly, each device D_j obtains the pair $(\alpha_j, f(\alpha_j))$ where $\alpha_j = j$ is publicly known and $f(\alpha_j)$ is kept secret. The **Setup** algorithm consists of two phases:

$(A_j, pk, pk_j, pk_{p,j}) \leftarrow \text{ParGen} \leftarrow (n, t, \kappa)$: each device D_j for $j = 1, \dots, n$ does as follow and with respect of a security parameter κ :

- generate at random $d_1^{(j)}, \dots, d_{t-1}^{(j)}$,
- generate the Paillier's key pair $(pk_{p,j}, sk_{p,j})$,
- generate at random k_j in $\mathbb{Z}_{q_{EC}}$,
- compute $pk_j = g^{k_j}$.

The values $A_j = (k_j, d_1^{(j)}, \dots, d_{t-1}^{(j)}, sk_{p,j})$ are privately stored in each device whereas $(pk_j, pk_{p,j})$ are made public. An agreed user compute the common key

$pk = \prod_{i=1}^n pk_i$ (i.e., Blockchain wallet public key). Note that the coefficients $d_i^{(j)}$ need to meet the following dis-equality:

$$|d_i^{(j)}| < \left(\frac{|N|}{n} - |q_{EC}|\right) \frac{2}{t(t-1)}, \quad (5)$$

for $i = 1, \dots, t-1$ and $j = 1, \dots, n$.

Lemma 1. Equation 5 allows the polynomial $f(x)$ to not be modified by applying Paillier cryptosystem.

Proof. Since we are going to encrypt $f(x)$ with Paillier cryptosystem, we need that its evaluation $f(\alpha_j)$ is smaller than Paillier modulus N for all $j = 1, \dots, n$. To do so, we need that $|f(\alpha_n)| = |(d_{t-1}^{(1)} + \dots + d_{t-1}^{(n)})\alpha_n^{t-1} + \dots + (d_1^{(1)} + \dots + d_1^{(n)})\alpha_n + \sum_i k_i| < |N|$, where $\alpha_i < \alpha_2 < \dots < \alpha_n$ by construction. Let d be equal to $\max_{i,j} d_i^{(j)}$, then

$$\begin{aligned} |f(\alpha_n)| &< n|d||\alpha_n|^{t-1} + \dots + n|d||\alpha_n| + n|q_{EC}| = \\ n|d|((t-1) * (t-2) * \dots * 1)|\alpha_n| + n|q_{EC}| &= \\ n|d|\frac{t*(t-1)}{2}|\alpha_n| + n|q_{EC}| &< |N|. \end{aligned}$$

Therefore, $|d| < \left(\frac{|N|}{n} - |q_{EC}|\right) \frac{2}{t(t-1)}$.

$(\alpha_j, f(\alpha_j)) \leftarrow \text{PolyEval} \leftarrow (\Lambda_j, pk_{p,j})$: all devices engage in the following step for the computation of each $f(\alpha_j)$ for $j = 1, \dots, n$. Let h be equal to $j+1$:

1. D_h generates random value $r_{j,h}$ and compute $\epsilon_{j,h} = \text{Enc}_{pk_{p,j}}(\alpha_j, r_h)$,
2. D_h generates random value $v_{j,h}$ and compute

$$c_h = \epsilon_{j,h}^{\alpha_j^{t-2} * d_{t-1}^{(h)}} * \epsilon_{j,h}^{\alpha_j^{t-3} * d_{t-2}^{(h)}} * \dots * \epsilon_{j,h}^{d_1^{(h)}} * \text{Enc}_{pk_{p,j}}(k_j, v_{j,h}),$$

3. if $h = j+1$, then D_h sends c_h to D_{h+1} ,
4. if $h \neq j$, then set $h = h+1 \bmod n$ and go to Step (1),
5. if $h = j$, then D_j computes

$$f(\alpha_j) = \text{Dec}_{sk_{p,j}}(c_{j-1}) + d_{t-1}^{(j)}\alpha_j^{t-1} + \dots + d_1^{(j)}\alpha_j + k_j.$$

The algorithm outputs for each device D_j the pair $(\alpha_j, f(\alpha_j))$ where $\alpha_j = j$ is publicly known and $f(\alpha_j)$ is kept secret. In Fig. 3, **PolyEval** phase is sketched in the case of five devices where D_1 acts as MD. At the end of the process D_1 obtains the evaluation of the $f(x)$ in α_1 , i.e., $f(\alpha_1)$.

3.2 Signing Algorithm

During the **Signing** algorithm, t out of n devices need to collaborate to generate the signature σ . To do so, t specific devices need to be agreed. Therefore, let $\mathcal{J}_t \subset \{1, \dots, n\}$ be the set of t indices such that D_j with $j \in \mathcal{J}_t$ has been selected for signing. The **Signing** algorithm consists of two phases:

$(\{s_j\}_{j \in \mathcal{J}_t}) \leftarrow \text{SessionKeyGen} \leftarrow (\mathcal{J}_t, \alpha_j, f(\alpha_j))$: each D_j with $j \in \mathcal{J}_t$ does as follow:

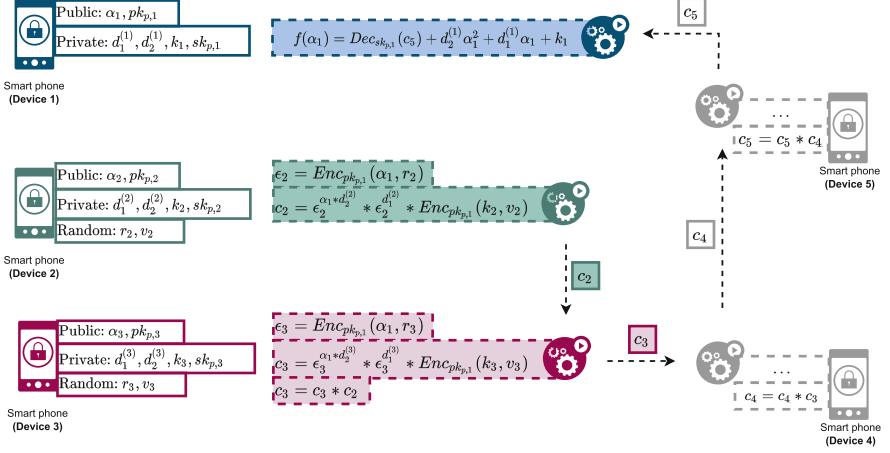


Fig. 3. PolyEval phase in Set-up algorithm of the proposed (n, t) -threshold signature.

- compute the session key:

$$s_j = f(\alpha_j) * \prod_{m \in \mathcal{J}_t \setminus \{j\}} \frac{\alpha_m}{\alpha_m - \alpha_j} \mod q_{EC},$$

- store s_j privately.

Note that s_j is the private session key of D_j used to generate σ_j and, therefore, s_j is kept secret.

$(\sigma) \leftarrow \text{SignatureGen} \leftarrow (\{s_j\}_{j \in \mathcal{J}_t}, m)$: at first, each device D_j with $j \in \mathcal{J}_t$ commits to its random value r_j . At second, the commitments c_j are sent to the MD device that aggregates them to receive the common commitment c . Then c is sent back to SDs with the signing message m (i.e., Blockchain transaction) so that each device D_j generates its signature fragment (z_j) on m . At last, the MD aggregates all signature fragments and outputs the Schnorr signature $\sigma = (e, z)$ on the Blockchain transaction which is sent to the Blockchain for validation, see Algorithm 1 for more details.

3.3 Verifying Algorithm

The **Verifying** algorithm allows the verifier (i.e., Blockchain node) to verify the signature (see Sect. 2.1 for more details).

$(0/1) \leftarrow \text{Verifying} \leftarrow (pk, \sigma, m)$: this is run by the verifier that verifies if the signature is valid as follow:

$$c' = g^z * pk^e \quad (6)$$

Algorithm 1. SignatureGen($\{s_j\}_{j \in \mathcal{J}_t}, m$)

```

1: for  $j \in \mathcal{J}_t$  do:                                 $\triangleright$  run privately by each  $D_j$  (i.e.,  $MD$  and  $SDs$ )
2:    $r_j \in_R \mathbb{Z}_{q_{EC}}$ 
3:    $c_j = g^{r_j}$                                       $\triangleright$   $SDs$  send  $c_j$  to  $MD$ 
4: end for
5:  $c = \prod_{j \in \mathcal{J}_t} c_j$                                 $\triangleright$  run by  $MD$ ,  $c$  and  $m$  sent to  $SDs$ 
6: for  $j \in \mathcal{J}_t$  do:                                 $\triangleright$  run privately by each  $D_j$  (i.e.,  $MD$  and  $SDs$ )
7:    $e = \mathcal{H}(c||m)$ 
8:    $z_j = r_j - e * s_j \bmod q_{EC}$                   $\triangleright$   $SDs$  send  $z_j$  to  $MD$ 
9: end for
10:  $z \leftarrow \sum_{j \in \mathcal{J}_t} z_j \bmod q_{EC}$             $\triangleright$  run by  $MD$ 
11: return  $\sigma = (e, z)$ 

```

$$e \stackrel{?}{=} \mathcal{H}(c' || m). \quad (7)$$

If Eq. 7 holds, the signature σ is accepted and the algorithm returns true, false otherwise.

4 Security Analysis

In this section, we prove the security of our threshold signature scheme. The signature is based on provable secure cryptographic primitives, namely Schnorr signature [31], Pailler cryptosystem [29], and Shamir secret sharing scheme [32].

Lemma 2. *The proposed threshold signature is existentially **unforgeable** under chosen-message attacks in the random oracle model assuming that the discrete logarithm problem is hard.*

Proof. This is based on the fact that our proposal is built on the Schnorr signature and its unforgeability is proven in [30], Lemma 2.

Lemma 3. *The proposed threshold signature is **sound and complete**, i.e., valid signatures will be always verified correctly, and invalid ones will always fail verification.*

Proof. In order to prove the completeness of the signature, Eq. 7 has to hold. This happens if:

1. the sum of the sessions keys s_i is equal to the private key $sk = \sum_{i=1}^n k_i$. This follows from Eq. 3, where $\beta_j = f(\alpha_j)$. In fact, $\sum_{j=1}^t s_i = \sum_{j=1}^t f(\alpha_j) * \prod_{m=1, m \neq j}^t \frac{\alpha_m}{\alpha_m - \alpha_j} = f(0) = sk$, where a re-labeling of the elements of \mathcal{J}_t is applied;
2. the commitment c' (i.e., Eq. 6) is correctly reconstructed

$$\begin{aligned}
 c' &= g^z * pk^e = g^{\sum_{i=1}^t (r_i - e * s_i)} (g^{sk})^e = g^{\sum_{i=1}^t r_i - e * sk} (g^{sk})^e = g^{\sum_{i=1}^t r_i} \\
 &= \prod_{i=1}^t g^{r_i} = \prod_{i=1}^t c_i = c.
 \end{aligned}$$

Soundness (sketch of proof by contradiction): if an unauthorized signer is able to produce at least two valid signatures $\sigma = (e, z)$ and $\sigma' = (e', z')$ without knowing sk , then Eq. 6 has to hold, i.e., both signatures pass the verification phase. Note that $1 = pk^{e-e'} * g^{z-z'}$ and, therefore, $pk = g^{\frac{z'-z}{e-e'}}$ where $sk = \frac{z'-z}{e-e'}$, i.e., the unauthorized signer knows the secret key sk .

Lemma 4. *The proposed threshold signature is **zero-knowledge**. This means that there exists a simulator \mathbb{S} that is able to efficiently generate a protocol transcript indistinguishable from a real protocol transcript without the knowledge of the private key sk .*

Proof. We prove the zero-knowledge property by constructing the zero-knowledge simulator \mathbb{S} . Let's assume that the simulator can program the random oracle \mathcal{H} in a way that on inputs \hat{c} , m outputs \hat{e} . Then, the simulator does as follows:

1. Randomly selects the response $\hat{z} \in_R \mathbb{Z}_q$.
2. Randomly selects the challenge $\hat{e} \in_R \mathbb{Z}_q$.
3. Computes the commitment $\hat{c} = pk^{\hat{e}} * g^{\hat{z}}$.

The simulator's output is computationally indistinguishable from the real protocol transcript, i.e., $(\hat{e}, \hat{z}) \cong_c (e, z)$, because all pairs are selected randomly and uniformly from the same sets.

Lemma 5. *The proposed threshold signature provides both secret sharing properties, i.e., **correctness and perfect privacy**.*

Proof. Note that the secret sk can be recovered with either the secret keys k_i or the session keys s_i . Therefore, correctness and perfect privacy properties need to be proven in both cases. For k_i , the proof is straightforward and follows the Shamir secret sharing scheme ones [2]. **Correctness:** each session key s_i secrecy relies on $f(x)$ one. In fact, if one knows $f(x)$, then can evaluate $f(x)$ in all α_i and reconstruct s_i . Thanks to PolyEval algorithm, any signer knows only partial values of each coefficient of $f(x)$. Accordingly, $f(x)$ can be reconstructed only knowing all $d_j^{(i)}$. **Perfect privacy:** the modulus and Paillier encryption prevent to have information on the coefficient $d_j^{(i)}$ of $f(x)$ and, therefore, to reconstruct the session keys s_i , where $sk = \sum_i s_i$.

5 Deployment of (n, t) -Threshold Scheme to the Blockchain

Two main use case scenarios for (n, t) -threshold scheme deployment to the Blockchain are depicted in Fig. 4. On one hand, the Multi-device wallet scenario aims at higher protection of the Blockchain wallet. The user owns a Blockchain wallet of which a secret key is split between his/her several wearable devices. In this case, we use a $(5, 2)$ -threshold signature where two of five devices have to collaborate to sign a transaction by the Blockchain wallet secret key. On the other

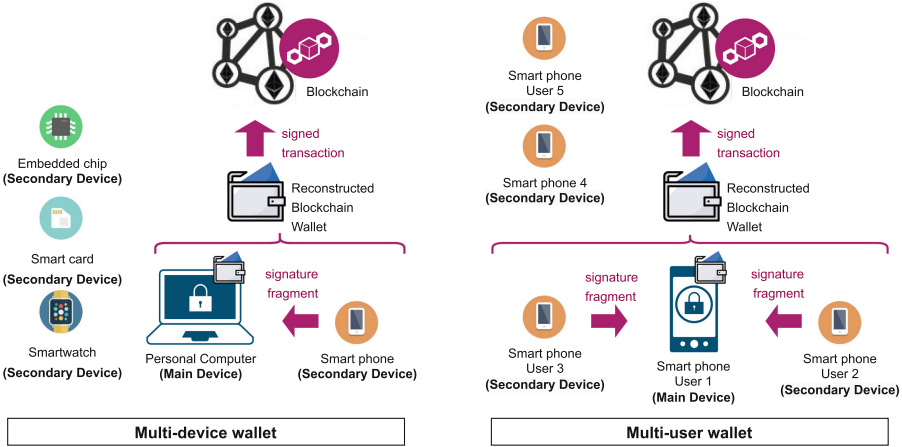


Fig. 4. Practical use cases of our (n, t) -threshold signatures in the Blockchain.

hand, the Multi-user wallet scenario focuses on stronger privacy protection of users and sharing property of the Blockchain wallet between several users. The users own a common Blockchain wallet of which a secret key is split between all of them. Here, we consider a $(5, 3)$ -threshold signature, where three of five users have to collaborate to reconstruct the Blockchain wallet, i.e., sign a transaction by the Blockchain wallet. The signature on the transaction is verifiable by the Blockchain wallet public key. However, no one is able to track back the signers, since, the signature is generated anonymously on behalf of the group of users sharing the wallet. In both scenarios, the verifier or even eavesdropper will learn nothing about the signers, i.e., the number of users/devices sharing the Blockchain wallet and the required threshold for reconstructing the wallet. In fact, there is only one group Blockchain wallet public key and several fragments of the Blockchain wallet secret key distributed between several users/devices.

6 Experimental Results

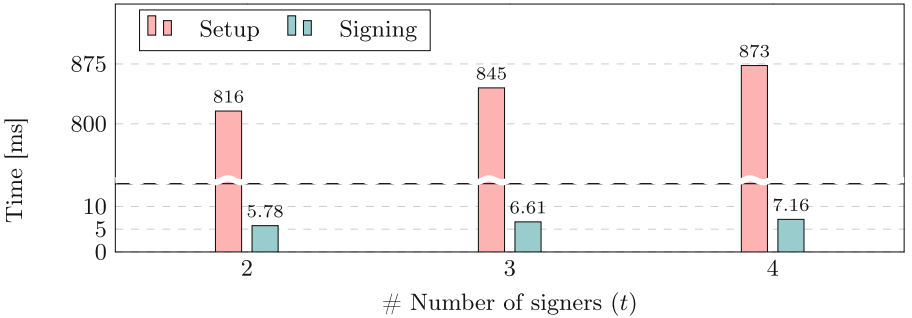
In this section, we provide experimental results of the proposed (n, t) -threshold scheme. We assess the execution time for all deployed algorithms (i.e., **Setup**, **Signing**, and **Verifying**) independently, as well as the overall execution time. We used one single Raspberry Pi 4 Model B with 4 GB of RAM to represent all system entities, i.e., **Signer**' MD and SDs devices, and the **Verifier**'s device. The testing application is written in C programming language and uses several external cryptographic libraries. The cryptographic core follows the key length recommendations defined by the National Institute of Standards and Technology (NIST) [1] for 112-bit security strength. We use Paillier cryptosystem with modulus size $|N| = 2048$ bits, where $N = P * Q$ and $|P| = |Q| = 1024$ bits primes and Shamir protocol with elements of 256-bit length sizes. Both protocols were

Table 1. Benchmarks in ms of the **Setup**, **Signing**, and **Verifying** algorithms for (5,3)-threshold scheme.

| Setup algorithm | | Signing algorithm | | |
|---------------------|----------|-------------------|-----------------------------|--------------------|
| ParGen | PolyEval | SessionKeyGen | SignatureGen (σ_j) | Total (σ) |
| 273.7 | 570.9 | 0.02 | 8.57 | 8.59 |
| Verifying algorithm | | Total | | |
| Verifying | | 870.44 | | |
| 16.95 | | | | |

implemented by using the **GMP** library . Furthermore, we use the **micro-ecc** library to implement the Schnorr signature over elliptic curve. Namely, we use standardized elliptic curve secp256r1 where $|p| = |q| = 256$ bits. Finally, we utilized **OpenSSL** library to perform SHA-256 hash algorithm.

We follow the environment model depicted in Fig. 3, with a total of five devices, of which three were required to perform the signature. Each device is simulated by a separated application thread. Table 1 shows the benchmarks of the **Setup** algorithm, the benchmarks of the **Signing** algorithm, and the benchmarks of the **Verifying** algorithm. In addition, we provide the whole protocol execution timings. Note that the benchmarks shown for **ParGen** algorithm relate to the execution time on each device (owing to running in parallel), whilst the polynomial evaluation refers to the overall execution time across all five devices. Regarding the **Signing** algorithm, the session key generation is performed in parallel on each device whereas the times for signature generation are divided into two parts: 1) the partial signature on each device and 2) the joint signature. Additionally, we run several experiments with varying numbers of devices and signers. Figure 5 depicts the execution timings of the protocol for $n = 5$ and $t = \{2, 3, 4\}$, whereas Fig. 6 illustrates the speed for $n = 10$ and $t = \{2, \dots, 9\}$. Since signature verification is consistent across all devices, we simply included

**Fig. 5.** Speed comparison of the threshold scheme for $n = 5$ and $t = \{2, 3, 4\}$.

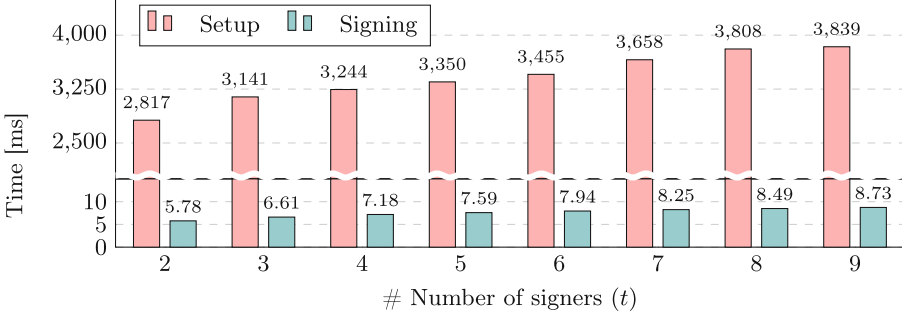


Fig. 6. Speed comparison of the threshold scheme for $n = 10$ and $t = \{2, \dots, 9\}$.

the **Setup** and **Signing** algorithms. The most costly component of the scheme is the **Setup** algorithm, which requires around 1s for (5, 4) setting and around 4s for (10, 9) setting. Fortunately, this procedure only has to be performed once. On the other side, the **Signing** algorithm requires less than 10 ms in all settings.

Since our benchmarks do not consider communication overhead, we compute how much data need to be sent and evaluate how long it would take via Ethernet and Bluetooth connections. During the **PolyEval** phase, the protocol requires to transfer 2,048 B between all devices in one round (i.e., $|c_i| = 512$ B, 5 devices deployed, i.e., $4 \cdot 512 = 2,048$ B). We consider parallel processing of all 5 rounds of the **PolyEval** phase. Using Transmission Control Protocol (TCP), the communication latency is negligible. However, the Bluetooth requires ca. 3s. The **SignatureGen** phase requires to transfer 160 B (i.e., $|c_j| = 64$ B, $|c| = 64$ B, and $|s_j| = 32$ B) between MD and one SD. Also in this case, we consider parallel communication processing with all signing SDs. TCP communication latency is again negligible whereas the Bluetooth communication takes ca. 3s.

7 Conclusion

In this article, we propose a novel (n, t) -threshold signature scheme suitable for increasing security and privacy in Blockchain technology. Our scheme allows securely splitting a Blockchain wallet between more devices that can be held by a single user or by several collaborative users. In the first case, the user's security is increased, since more user devices need to be compromised to sign Blockchain transactions. The former case increases user privacy, where a signature can be anonymously made on behalf of the group of users sharing the Blockchain wallet. Our experimental results show that the proposed signature can be practically deployed due to its fast signing phase that requires less than 10 ms when 10 devices are involved.

Acknowledgements. Research described in this paper was financed by the Technology Agency of the Czech Republic ‘DELTA 2 Programme’ under grant TM02000036. The authors gratefully acknowledge funding from European Union’s Horizon 2020

Research and Innovation programme under the Marie Skłodowska Curie grant agreement No. 813278 (A-WEAR: A network for dynamic wearable applications with privacy constraints, <http://www.a-wear.eu/>).

References

1. Barker, E.: Recommendation for key management part 1: general (revision 5). NIST Spec. Publ. Part 1 **800**(57), 1–171 (2020)
2. Beimel, A.: Secret-sharing schemes: a survey. In: Chee, Y.M., et al. (eds.) IWCC 2011. LNCS, vol. 6639, pp. 11–46. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20901-7_2
3. BitcoinCore: Technology roadmap - schnorr signatures and signature aggregation (2017). <https://bitcoincore.org/en/2017/03/23/schnorr-signature-aggregation/>
4. Boneh, D., Drijvers, M., Neven, G.: Compact multi-signatures for smaller blockchains. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 435–464. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_15
5. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptol. **17**(4), 297–319 (2004). <https://doi.org/10.1007/s00145-004-0314-9>
6. Canetti, R., Gennaro, R., Goldfeder, S., Makriyannis, N., Peled, U.: UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 1769–1787 (2020)
7. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Two-party ECDSA from hash proof systems and efficient instantiations. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 191–221. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_7
8. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Bandwidth-efficient threshold EC-DNA. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 266–296. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_10
9. Cogliati, B., et al.: Provable security of (tweakable) block ciphers based on substitution-permutation networks. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 722–753. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_24
10. Dalskov, A., Orlandi, C., Keller, M., Shrishak, K., Shulman, H.: Securing DNSSEC keys via threshold ECDSA from generic MPC. In: Chen, L., Li, N., Liang, K., Schneider, S. (eds.) ESORICS 2020. LNCS, vol. 12309, pp. 654–673. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59013-0_32
11. Damgård, I., Jakobsen, T.P., Nielsen, J.B., Pagter, J.I., Østergaard, M.B.: Fast threshold ECDSA with honest majority. In: Galdi, C., Kolesnikov, V. (eds.) SCN 2020. LNCS, vol. 12238, pp. 382–400. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57990-6_19
12. Damgård, I., Koprowski, M.: Practical threshold RSA signatures without a trusted dealer. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 152–165. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_10
13. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_28

14. Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Secure two-party threshold ECDSA from ECDSA assumptions. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 980–997. IEEE (2018)
15. Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Threshold ECDSA from ECDSA assumptions: the multiparty case. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 1051–1066. IEEE (2019)
16. Dzurenda, P., Ricci, S., Casanova-Marqués, R., Hajny, J., Cika, P.: Secret sharing-based authenticated key agreement protocol. In: The 16th International Conference on Availability, Reliability and Security, pp. 1–10 (2021)
17. Freemanlaw: Cryptocurrency transactions: Multi-signature arrangements explained. <https://freemanlaw.com/cryptocurrency-transactions-multi-signature-arrangements-explained/>
18. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1179–1194 (2018)
19. Gennaro, R., Goldfeder, S., Narayanan, A.: Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 156–174. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_9
20. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust threshold DSS signatures. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 354–371. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_31
21. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure applications of Pedersen’s distributed key generation protocol. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 373–390. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36563-X_26
22. Itakura, K.: A public-key cryptosystem suitable for digital multisignatures. NEC J. Res. Dev. **71**, 1–8 (1983)
23. Komlo, C., Goldberg, I.: FROST: flexible round-optimized schnorr threshold signatures. In: Dunkelman, O., Jacobson, Jr., M.J., O’Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 34–65. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81652-0_2
24. Li, C.Y., Chen, X.B., Chen, Y.L., Hou, Y.Y., Li, J.: A new lattice-based signature scheme in post-quantum blockchain network. IEEE Access **7**, 2026–2033 (2018)
25. Lindell, Y.: Fast secure two-party ECDSA signing. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 613–644. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_21
26. Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1837–1854 (2018)
27. MacKenzie, P., Reiter, M.K.: Two-party generation of DSA signatures. Int. J. Inf. Secur. **2**(3), 218–239 (2004)
28. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_3
29. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
30. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396 (2000)

31. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_22
32. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
33. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_15
34. Stinson, D.R., Strobl, R.: Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In: Varadharajan, V., Mu, Y. (eds.) ACISP 2001. LNCS, vol. 2119, pp. 417–434. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47719-5_33