electronics

Article

Threshold/Multi Adaptor Signature and Their Applications in Blockchains

Yunfeng Ji [1,2], Yuting Xiao [1,*], Birou Gao [1,2] and Rui Zhang [1,2,*]

[1] Institutite of Information Engineering, Chinese Academy of Sciences, Beijing 100085, China; jiyunfeng@iie.ac.cn (Y.J.); gaobirou@iie.ac.cn (B.G.)
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: xiaoyuting@iie.ac.cn (Y.X.); r-zhang@iie.ac.cn (R.Z.)

Abstract: Adaptor signature is a variant of digital signatures and useful for fair excheng in financial applications such as cryptocurrencies, to name a few, off-chain transaction protocols, atomic swaps and other privacy-enhancing mechanisms. However, similar to normal digital signatures, an adaptor signature also suffers from the loss of the secret key and single-point failure, which is insufficient in practice. In this paper, we address this constraint by introducing two new concepts as enhancements: multi-adaptor signatures and threshold adaptor signatures. First, we propose the formal security models for multi-adaptor signature and threshold adaptor signature. Then, we present specific schemes for these two primitives based on the commonly used blockchain signature scheme Schnorr and the post-quantum signature scheme Dilithium, respectively. Furthermore, we provide security proofs for these four schemes. Finally, we demonstrate interesting applications for blockchains, such as oracle-based conditional payment and n to n atomic swap.

Keywords: adaptor signature; multi-signature; threshold signature; blockchain; Schnorr; Dilithium

Citation: Ji, Y.; Xiao, Y.; Gao, B.; Zhang, R. Threshold/Multi Adaptor Signature and Their Applications in Blockchains. *Electronics* **2024**, *13*, 76. https://doi.org/10.3390/electronics13010076

Academic Editor: Mehdi Sookhak

Received: 13 November 2023
Revised: 4 December 2023
Accepted: 7 December 2023
Published: 23 December 2023

1. Introduction

Due to its decentralized, anonymous, traceable, and transparent nature, blockchain has extensive applications. However, existing blockchain applications, such as cryptocurrencies, face challenges like poor scalability and low throughput. Addressing these issues, payment channel networks (PCNs) [1,2] establish channels on-chain, enabling numerous off-chain transactions between users. This reduces on-chain transaction volume, increases transaction throughput, and lowers on-chain costs. As a crucial technology for building PCNs in blockchains, adapter signatures [3] serve as important building block in addressing issues such as poor scalability and low throughput.

An adaptor signature is a cryptographic primitive that enables a signer to create a pre-signature under its secret key, adaptable into a valid signature by a publisher possessing a specific secret value. If the finalized signature becomes public, the signer can extract the secret value employed by the publisher. In blockchain applications, adapter signatures can be utilized in atomic swap [4], enabling two parties to proceed cross-chain fair exchange.

Adaptor signatures can be viewed as an extension of digital signatures to address the lack of mutual trust. Aside from key generation, signing and verification algorithms, there is a stage called pre-signing, where a pre-signing algorithm produces a pre-signature, which can be verified by a pre-signature verification algorithm. To convert a pre-signature into a normal signature, an adaptation algorithm and evidence extraction algorithm come into place. Adaptor signatures possess two distinct capabilities, i.e., authorization and evidence extraction, achieved by the integration of hard relations. Currently, motivated by blockchains, many works proposed adaptor signature schemes based on Schnorr signatures [3,5] and ECDSA signatures [3,6]. Additionally, for long-term security, there is also work [7] that provided adaptor signatures based on lattice signatures.

In an adaptor signature system, only a single signer is considered, which is often insufficient in many scenarios. We consider the following two instances:

- In cryptocurrency transactions, the loss of the secrete key of the signer can lead to significant financial losses. If we extend the adaptor signatures to a threshold setting, called the threshold adaptor signature scheme, the loss of a single secret key share does not compromise the security of the system.
- In blockchain-based crowdfunding scenarios, transactions require the consent of every stakeholder. This needs the extension of adaptor signatures into multi-setting, ensuring that a valid signature can only be generated when all stakeholders agree on the transaction.

As we can see, adaptor signatures are not suitable for the above scenarios that require multi-signers, and basic multi-party signatures lack the features of adaptor signatures. Therefore, this paper focuses on multi/threshold adaptor signatures. In the case of (t, n) -threshold adaptor signatures, $t + 1$ out of n secret key share holders need to participate to generate a valid signature. Moreover, multi-adaptor signatures require the participation of all secret key share holders in the signing protocol.

1.1. Our Results and Technical Overview

In this paper, we formally study multi-adaptor signatures and threshold adaptor signatures. Our contributions are three-fold:

Formal Models. We proposed a formal model for the multi-adaptor signature and threshold adaptor signature with one witness holder, n signers $P_{i|i\in[n]}$, and public verifiers.

A multi-adaptor signature scheme $\mathsf{MASIG_R} = (\mathsf{MKGen}, \mathsf{MSign}, \mathsf{KAgg}, \mathsf{pMVerify}, \mathsf{MAdapt}, \mathsf{MVerify}, \mathsf{MExt})$ consists of one interactive protocol MSign and six non-interactive algorithms. The system works as follows:

- To start, the witness holder provides a statement Y, and each signer P_i will invoke the MKGen algorithm to generate its public-secret key pair (pk_i, sk_i).
- Next, given Y and a message m to be signed, the n signers jointly run the MSign protocol to generate a pre-signature $\tilde{\sigma}$ for m, then employ the KAgg algorithm to generate an aggregated key apk using $\{pk_i\}_{i\in[n]}$.
- Then, using apk and Y, the witness holder can verify the validity of $\tilde{\sigma}$ by invoking the $\mathsf{pMVerify}$ algorithm. If $\tilde{\sigma}$ is valid, the witness holder can further utilize the witness y (of the statement Y such that $(Y, y) \in \mathsf{R}$, where R denotes a binary relation provided as a public parameter) to transform $\tilde{\sigma}$ into a signature σ.
- Any verifier can use $\mathsf{MVerify}$ to verify the validity of σ.
- From σ and $\tilde{\sigma}$, all signers can obtain the witness y through the MExt algorithm.

For a threshold adaptor signature scheme $\mathsf{TASIG_R} = (\mathsf{TKGen}, \mathsf{TSign}, \mathsf{pTVerify}, \mathsf{TAdapt}, \mathsf{TVerify}, \mathsf{TExt})$, which differs from $\mathsf{MASIG_R}$ in terms of the key generation algorithm (TKGen), the signing protocol (TSign) and the public key aggregation algorithm (KAgg). At the end of each execution instance of the protocol TKGen, each signer P_i obtains a secret key share sk_i along with the corresponding public key pk. Given the statement Y and a message m, any $t + 1$ (out of n) signers can jointly generate a valid pre-signature using their secret key shares and the public key by running the protocol TSign. Since all signers use a common pk, the KAgg algorithm for aggregating public keys in $\mathsf{MASIG_R}$ is not required in $\mathsf{TASIG_R}$.

Our security definitions continue the security requirements of adaptor signature pre-signature adaptability and witness extractability. Pre-signature adaptability ensures that any valid pre-signature specific to Y can be completed into a valid signature using. Witness extractability guarantees that a valid tuple $(\sigma, \tilde{\sigma})$ for a tuple (m, Y) can be used to extract a corresponding witness y. In addition, we provide definitions of unforgeability for $\mathsf{MASIG_R}$ and $\mathsf{TASIG_R}$, respectively. For $\mathsf{MASIG_R}$, we require that in a signing protocol involving n signers, even if the adversary corrupts $n - 1$ of them, it should still be unable to forge a valid pre-signature. Unlike $\mathsf{MASIG_R}$, for the unforgeability of (t, n)-$\mathsf{TASIG_R}$, it needs that fewer than $t + 1$ signers should not be able to generate a valid pre-signature. The adversary,

who can corrupt at most t signers, is allowed to participate in the key generation protocol, but still, it should be unable to forge a pre-signature that can be verified.

Schemes. We construct $\mathsf{MASIG_R}$ and (t,n)-$\mathsf{TASIG_R}$ schemes based on Schnorr, which is the commonly used signature scheme in blockchain to meet the diverse application requirements. Additionally, considering the long-term security and the post-quantum signature standards established by NIST, we also construct $\mathsf{MASIG_R}$ and $(n-1,n)$-$\mathsf{TASIG_R}$ based on Dilithium. We also provide security proofs for our schemes, demonstrating that the schemes satisfy pre-signature adaptability, witness extractability, and unforgeability.

Our $\mathsf{MASIG_R}$ and (t,n)-$\mathsf{TASIG_R}$ schemes with Schnorr are based on the multi-signature in [8] and the threshold signature in [9], respectively. Our $\mathsf{MASIG_R}$ and $(n-1,n)$-$\mathsf{TASIG_R}$ schemes with Dilithium are based on the multi-signature and $(n-1,n)$ signature from [10]. Our schemes maintain the key generation processes of the original schemes with slight modifications to their signing protocols, introducing a statement in the commitment generation. Correspondingly, the pre-verify algorithm $\mathsf{pVerify}$ also involves the statement.

Applications. We present an application of multi-adaptor signature, an n to n atomic swap. In contrast to the atomic swap implemented with basic adaptor signatures, this approach effectively prevents economic losses resulting from the loss of a single secret key. For threshold adaptor signatures, we present an application in oracle-based conditional payment. The security of threshold adaptor signature ensures that a single malicious oracle cannot disrupt the payments. In addition, the payer retains the right to know that oracles are transferring funds to the payee. Furthermore, threshold and multi-adaptor signatures can also be applied to electronic voting and cross-chain crowdfunding, respectively.

1.2. Related Work

Due to the application advantages of adaptor signatures, Malavolta et al. [2] constructed an anonymous multi-hop lock protocol based on adaptor signatures and then build a secure payment channel network. Thyagarajan et al. [11] provided an efficient instantiation of a two-party general atomic swap protocol based on ECDSA/Schnorr adaptor signatures. Aumayr et al. [3] utilized adaptor signatures to build a generalized channels structure on a script-limited blockchain, enabling secure off-chain execution and enhancing blockchain scalability.

Threshold signature was first proposed by Desmedt and Frankel [12], and they gave a threshold signature scheme based on the RSA assumption. Motivated by blockchains, efficient threshold Schnorr/ECDSA signature [9,13–16] received much attention lately. For post-quantum threshold signature, Bendlin et al. [17] proposed a lattice-based (t,n)-threshold signature based on Peikert hash-and-sign signature. Damgård et al. [10] presented a lattice-based $(n-1,n)$-threshold signature by implementing Dilithium-G in a multiparty setting. Multi-signature schemes enable a group of signers possessed an own secret/public key pair to produce a single signature σ on a message m. A number of modern and practical multi-signature schemes [18–22] are proposed based on Schnorr.

Organization of the Rest of the Paper. In Section 2, we give the preliminary. We give a model for multi-adaptor signature and thresho1d adaptor signature in Sections 3 and 4, respectively. We also give specific schemes for the new primitives. In Section 5, we discuss two applications of multi and thresho1d adaptor signature.

2. Preliminary

We now revisit adaptor signatures as presented in [23]. An adaptor signature scheme (w.r.t a hard relation R) $\mathsf{ASIG_R} = (\mathsf{KGen}, \mathsf{pSign}, \mathsf{pVerify}, \mathsf{Adapt}, \mathsf{Verify}, \mathsf{Ext})$ can be described as follows:

- $\mathsf{KGen}(1^\lambda)$: on input a security parameter λ, the randomized algorithm outputs the secret key sk and public key pk. In short, $(pk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$.
- $\mathsf{pSign}(sk, m, Y)$: on input sk, a message $m \in \{0,1\}^*$ and a statement $Y \in \mathcal{L}_\mathsf{R}$, the randomized algorithm outputs a pre-signature $\tilde{\sigma}$. In short, $\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$.

- $\mathsf{pVerify}(pk, m, Y, \tilde{\sigma})$: on input pk, $m \in \{0,1\}^*$, $Y \in \mathcal{L}_{\mathsf{R}}$ and $\tilde{\sigma}$, the deterministic algorithm outputs a bit b_1. In short, $b_1 \leftarrow \mathsf{pVerify}(pk, m, Y, \tilde{\sigma})$.
- $\mathsf{Adapt}(\tilde{\sigma}, y)$: on input $\tilde{\sigma}$ and a witness y, this deterministic algorithm outputs a signature σ. In short, $\sigma \leftarrow \mathsf{Adapt}(\tilde{\sigma}, y)$.
- $\mathsf{Verify}(pk, m, \sigma)$: on input m, pk and σ, this deterministic algorithm outputs a bit b_2 which equals 1 if and only if σ is a valid signature on m under pk. In short, $b_2 \leftarrow \mathsf{Verify}(pk, m, \sigma)$.
- $\mathsf{Ext}(\sigma, \tilde{\sigma}, Y)$: on input σ, $\tilde{\sigma}$ and $Y \in \mathcal{L}_{\mathsf{R}}$, this deterministic algorithm outputs y such that $(Y, y) \in \mathsf{R}$, or $\perp$. In short, $y/\perp \leftarrow \mathsf{Ext}(\sigma, \tilde{\sigma}, Y)$.

An adaptor signature scheme should satisfy pre-signature correctness, and a secure adaptor signature scheme $\mathsf{ASIG}_{\mathsf{R}}$ should satisfy pre-signature adaptability, unforgeablity and witness extractability.

3. Multi Adaptor Signature

Here, we propose a formal model for multi-adaptor signature and the corresponding security requirements. We construct two secure multi-adaptor signature schemes based on Schnorr and Dilithium, respectively.

3.1. Syntax

Definition 1. *A multi-adaptor signature scheme (w.r.t a hard relation* R*)* $\mathsf{MASIG}_{\mathsf{R}}$ = *(*MKGen, MSign, KAgg, $\mathsf{pMVerify}$, MAdapt, $\mathsf{MVerify}$, MExt*) consists of the following polynomial time protocol and algorithms, we also give a flowchart of multi-adaptor signature scheme in Figure 1.*

- $\mathsf{MKGen}(1^\lambda)$*: on input a security parameter* λ *as input, this randomized algorithm returns the private signing key* sk_i *and public verification key* pk_i*. In short,* $(pk_i, sk_i) \leftarrow \mathsf{MKGen}(1^\lambda)$.
- $\mathsf{MSign}\left\langle \{P_i(sk_i, pk_i, m, Y)\}_{i\in[n]} \right\rangle$*: This probabilistic protocol is jointly ran by n signing players* $\{P_i\}_{i\in[n]}$ *which generates a pre-signature* $\tilde{\sigma}$*.* P_i*'s input is a statement* $Y \in \mathcal{L}_{\mathsf{R}}$*, a message* $m \in \{0,1\}^*$*, his private signing key* sk_i *and public key* pk_i *generated in* MKGen*. The protocol is allowed to abort. In short,* $\langle \tilde{\sigma}/\perp \rangle \leftarrow \mathsf{MSign}\left\langle \{P_i(sk_i, pk_i, m, Y)\}_{i\in[n]} \right\rangle$.
- $\mathsf{KAgg}(\{pk_i\}_{i\in[n]})$*: on input* $\{pk_i\}_{i\in[n]}$*, this deterministic algorithm outputs an aggregated public key* apk*. In short,* $apk \leftarrow \mathsf{KAgg}(\{pk_i\}_{i\in[n]})$.
- $\mathsf{pMVerify}(apk, m, Y, \tilde{\sigma})$*: on input a message* $m \in \{0,1\}^*$*, an aggregated public key* apk*, a statement* $Y \in \mathcal{L}_{\mathsf{R}}$ *and a pre-signature* $\tilde{\sigma}$*, this deterministic algorithm outputs a bit* b_1*. In short,* $b_1 \leftarrow \mathsf{pMVerify}(apk, m, Y, \tilde{\sigma})$.
- $\mathsf{MAdapt}(\tilde{\sigma}, y)$*: on input a pre-signature* $\tilde{\sigma}$ *and a witness* y*, this deterministic algorithm outputs a signature* σ*. In short,* $\sigma \leftarrow \mathsf{MAdapt}(\tilde{\sigma}, y)$.
- $\mathsf{MVerify}(apk, m, \sigma)$*: on input an aggregated public key* apk*, a message* m *and a signature* σ*, this deterministic algorithm outputs a bit* b_2 *which equals 1 if and only if* σ *is a valid signature on* m *under* pk*. In short,* $b_2 \leftarrow \mathsf{MVerify}(apk, m, \sigma)$.
- $\mathsf{MExt}(\sigma, \tilde{\sigma}, Y)$*: on input a signature* σ*, a pre-signature* $\tilde{\sigma}$ *and a statement* $Y \in \mathcal{L}_{\mathsf{R}}$*, this deterministic algorithm outputs a witness* y *such that* $(Y, y) \in \mathsf{R}$*, or* $\perp$*. In short,* $y/\perp \leftarrow \mathsf{MExt}(\sigma, \tilde{\sigma}, Y)$.

Correctness. A multi-adaptor signature scheme should also satisfy pre-signature correctness.

Definition 2. *A multi-adaptor signature* $\mathsf{MASIG}_{\mathsf{R}}$ *satisfies pre-signature correctness, if for all* $m \in \{0,1\}^*$*,* $\{(pk_i, sk_i)\}_{i\in[n]}$ *generated by* MKGen*,* $(Y, y) \in \mathsf{R}$*,* $\tilde{\sigma}$ *generated by* MSign*,* $apk \leftarrow \mathsf{KAgg}(\{pk_i\}_{i\in[n]})$*,* $\sigma \leftarrow \mathsf{MAdapt}(apk, \tilde{\sigma}, y)$*, and* $y' \leftarrow \mathsf{MExt}(apk, \sigma, \tilde{\sigma}, Y)$*, the following holds:*

$$\Pr[\mathsf{pMVerify}(apk, m, Y, \tilde{\sigma}) = 1 \wedge \mathsf{MVerify}(apk, m, \sigma) = 1 \wedge (Y, y') \in \mathsf{R}] \geq 1 - negl(\lambda).$$

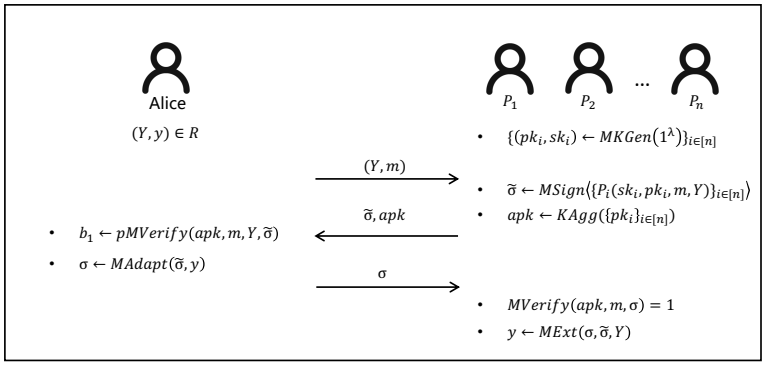

Figure 1. The flowchart of the multi adaptor signature.

3.2. Security Definitions

A multi-adaptor signature scheme $\mathsf{MASIG_R}$ is secure if it satisfies pre-signature adaptability, unforgeablity and witness extractability. We denote Q as the transcript of the interactions between adversary $\mathcal{A}$ and $\mathcal{O}_{Ms}$,$\mathcal{O}_{Mps}$. $\mathcal{O}_{Ms}$ is a signing oracle that for an input message m_j, $j \in [q_s]$, returns a valid public- verifiable signature σ_j, and $\mathcal{O}_{Mps}$ is a pre-signing oracle that returns a corresponding pre-signature $\langle\tilde{\sigma}_j\rangle \leftarrow \mathsf{MSign}\Big\langle\{P_i(sk_i, pk_i, m_j, Y)\}_{i\in[n]}\Big\rangle$. The formal definition of these properties are as follows.

Definition 3. *A multi-adaptor signature* $\mathsf{MASIG_R}$ *satisfies pre-signature adaptability, if for all* $n \in \mathbb{N}$ *and* $m \in \{0,1\}^*$*, aggregated public key* apk*,* $(Y, y) \in \mathsf{R}$ *and pre-signatures* $\tilde{\sigma} \leftarrow \{0,1\}^*$*; once we have that* $\mathsf{pMVerify}(apk, m, Y, \tilde{\sigma}) = 1$*, then the following holds:*

$$\Pr[\mathsf{MVerify}(pk, m, \mathsf{MAdapt}(pk, \tilde{\sigma}, y)) = 1] \geq 1 - negl(\lambda).$$

Without loss of generality, we assume there is a single honest player P_1. The unforgeablity and witness extractability can be described as follows.

Definition 4. *A multi-adaptor signature* $\mathsf{MASIG_R}$ *satisfies unforgeable, if for any PPT adversary* $\mathcal{A}$*, its advantage in the following experiment*

$$\begin{aligned} A^{uf}_{\mathsf{MASIG_R}} = \Pr[Q = \emptyset; (pk_1, sk_1) \leftarrow \mathsf{MKGen}(1^\lambda); (Y, y) \in \mathsf{R}; m^* \leftarrow \mathcal{A}^{\mathcal{O}_{Ms}, \mathcal{O}_{Mps}}(pk_1, Y); \\ \langle\tilde{\sigma}\rangle \leftarrow \mathsf{MSign}\Big\langle\{P_i(sk_i, pk_i, m, Y)\}_{i\in[n]}\Big\rangle; apk \leftarrow \mathsf{KAgg}(\{pk_i\}_{i\in[n]}); \\ \sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_{Ms}, \mathcal{O}_{Mps}}(\tilde{\sigma}, Y) : m^* \notin Q \wedge \mathsf{MVerify}(apk, m^*, \sigma^*) = 1] \end{aligned}$$

is negligible in λ*.*

Definition 5. *A multi-adaptor signature* $\mathsf{MASIG_R}$ *is witness extractable, if for any PPT adversary* $\mathcal{A}$*, its advantage in the following experiment*

$$\begin{aligned} A^{we}_{\mathsf{MASIG_R}} = \Pr[Q = \emptyset; (pk_1, sk_1) \leftarrow \mathsf{MKGen}(1^\lambda); (m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}_{Ms}, \mathcal{O}_{Mps}}(pk_1); \\ \langle\tilde{\sigma}\rangle \leftarrow \mathsf{MSign}\Big\langle\{P_i(sk_i, pk_i, m^*, Y^*)\}_{i\in[n]}\Big\rangle; \sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_{Ms}, \mathcal{O}_{Mps}}(\tilde{\sigma}, Y^*); apk \leftarrow \mathsf{KAgg}(\{pk_i\}_{i\in[n]}); \\ y \leftarrow \mathsf{MExt}(apk, \sigma^*, \tilde{\sigma}, Y^*) : m^* \notin Q \wedge (Y^*, y) \notin \mathsf{R} \wedge \mathsf{MVerify}(apk, m^*, \sigma^*) = 1] \end{aligned}$$

is negligible in λ*.*

3.3. Multi Adaptor Signature Based on Schnorr

Considering a p order cyclic group denoted as $\mathbb{G}$ with generator g, the discrete logarithm problem in $\mathbb{G}$ is hard. We consider the hard relation $\mathsf{R_s} = \{(Y, y) | Y = g^y\}$ and denote the hash functions H_a, H_n and H_s from $\{0,1\}^*$ to $\mathbb{Z}_p$. The scheme $\mathsf{S.MASIG_{R_s}}$=(S.MKGen, S.MSign, S.KAgg, S.pMVerify, S.MAdapt, S.MVerify, S.MExt) can be described as follows:

– $(pk_i, sk_i) \leftarrow \mathsf{S.MKGen}(1^\lambda)$.
Each signer P_i samples a random $x_i \leftarrow \mathbb{Z}_p$ and computes $X_i = g_i^x$, then generates the secret key $sk_i = x_i$ and public key $pk_i = X_i$.
– $\langle \tilde{\sigma}/\perp \rangle \leftarrow \mathsf{S.MSign}\Big\langle \{P_i(sk_i, pk_i, m, Y)\}_{i\in[n]} \Big\rangle$.
Given a message m and a statement $Y \in \mathcal{L}_{\mathsf{R_s}}$, for each $j \in \{1, \cdots, v\}$, each signer P_i generates random $r_{i,j} \leftarrow \mathbb{Z}_p$ and computes $R_{i,j} = g^{r_{i,j}}$. It then outputs and broadcasts the v nonces $(R_{i,1}, \cdots, R_{i,v})$. After receiving the nonces from others, each player computes $R_j = \prod_{i=1}^n R_{i,j}$ for each $j \in [v]$ and outputs $(R_1, \cdots, R_v)$. Each player computes $a_i \leftarrow \mathsf{H}_a(L, X_i)$, where $L = \{X_i\}_{i\in[n]}$, and $\tilde{X} \leftarrow \prod_{i=1}^n X_i^{a_i}$. Then, each signer P_i computes $o \leftarrow \mathsf{H}_n(\tilde{X}, (R_1, \cdots, R_v), m)$, $R \leftarrow Y \cdot \prod_{j=1}^v R_j^{o^{j-1}}$, $c \leftarrow \mathsf{H}_s(\tilde{X}, R, m)$ and $s_i \leftarrow ca_i x_i + \sum_{j=1}^v r_{i,j} b^{j-1} \mod p$. Broadcast s_i. After receiving all s_j form all players, P_i computes $\tilde{s} \leftarrow \sum_{i=1}^n s_i$. The signature of m is $\tilde{\sigma} = (R, \tilde{s})$.
– $apk \leftarrow \mathsf{S.KAgg}(\{pk_i\}_{i\in[n]})$.
Given $\{pk_i\}_{i\in[n]}$, the algorithm computes $a_i \leftarrow \mathsf{H}_a(L, X_i)$, where $L = \{X_i\}_{i\in[n]}$, and $\tilde{X} \leftarrow \prod_{i=1}^n X_i^{a_i}$. The aggregated public key is $\tilde{X}$.
– $b_1 \leftarrow \mathsf{S.pMVerify}(apk, m, Y, \tilde{\sigma})$.
Given an aggregated public key $\tilde{X}$, a message m, a statement $Y \in \mathcal{L}_{\mathsf{R_s}}$ and an adaptor signature $\tilde{\sigma} = (R, \tilde{s})$, the algorithm computes $c = \mathsf{H}_s(\tilde{X}, R, m)$ and accepts the adaptor signature if $g^{\tilde{s}} Y = R\tilde{X}^c$.
– $\sigma \leftarrow \mathsf{S.MAdapt}(\tilde{\sigma}, y)$.
Given an adaptor signature $\tilde{\sigma} = (R, \tilde{s})$ and a witness y, the algorithm outputs the signature $\sigma = (R, s)$, where $s = \tilde{s} + y$.
– $b_2 \leftarrow \mathsf{S.MVerify}(apk, m, \sigma)$.
Given a message m, an aggregated public key $\tilde{X}$, and a signature $\sigma = (R, s)$, the verifier computes $c = \mathsf{H}_s(\tilde{X}, R, m)$ and accepts the signature if $g^s = R\tilde{X}^c$.
– $y/\perp \leftarrow \mathsf{S.MExt}(\sigma, \tilde{\sigma}, Y)$.
Given an adaptor signature $\tilde{\sigma} = (R, \tilde{s})$, a signature $\sigma = (R, s)$ and a statement $Y \in \mathcal{L}_{\mathsf{R_s}}$, the algorithm can return the witness $y \leftarrow s - \tilde{s}$.

Theorem 1. $\mathsf{S.MASIG_R}$ = *(*S.MKGen, S.MSign, S.KAgg, S.pMVerify, S.MAdapt, S.MVerify, S.MExt*) is a secure multi-adaptor signature.*

Proof. As we can see, the scheme satisfies pre-signature correctness. We now show that $\mathsf{S.MASIG_R}$ satisfies pre-signature adaptable, unforgeablity and witness extractable.

Pre-signature adaptability. Assume $\mathsf{pMVerify}(apk, m, Y, \tilde{\sigma}) = 1$ holds, such that $g^{\tilde{s}} Y = R\tilde{X}^{\mathsf{H}_{sig}(\tilde{X}, R, m)}$. For $(Y, y) \in \mathsf{R_s}$, $g^{\tilde{s}+y} = R\tilde{X}^{\mathsf{H}_{sig}(\tilde{X}, R, m)}$. It direct implies with $\mathsf{MVerify}(apk, m, (s, R)) = 1$, with $s = \tilde{s} + y$ and $(s, R) = \mathsf{S.MAdapt}(\tilde{\sigma}, y)$. Therefore, the vaild pre-signature can be adapted in a vaild signature.

Unforgeability. If $\exists$ adversary $\mathcal{A}_{Uf}$ corrupts at most $n-1$ signing players can forge a valid confirmer signature, an efficient adversary $\mathcal{A}_{MS}$ can be constructed that can break the security of multi-signature scheme in [8]. For setup, $\mathcal{A}_{MS}$ generates $(pk_i, sk_i) \leftarrow \mathsf{MASIG_R}(1^\lambda)$, $(Y, y) \in R$ and sends $\{pk_i\}_{i\in[n]}$ to $\mathcal{A}_{Uf}$. When $\mathcal{A}_{Uf}$ queries $\mathcal{O}_{Ms}$ on message m_j, $\mathcal{A}_{MS}$ interacts with its own challenger to obtain a transcript (R_j, c_j, s_j) and gives (R_j, s_j) to $\mathcal{A}_{Uf}$. When $\mathcal{A}_{Uf}$ queries $\mathcal{O}_{Mps}$ on message m_k, $\mathcal{A}_{MS}$ obtains a transcript (R_k, c_k, s_k) and gives $(R_k \cdot Y, s_k)$ to $\mathcal{A}_{Uf}$. When $\mathcal{A}_{Uf}$ outputs a forged signature (m^*, σ^*) on a message $m^* \notin \mathcal{L}_1$, $\mathcal{A}_{MS}$ can provide a valid signature of scheme in [8].

Witness extractability. If $\exists$ adversary $\mathcal{A}_{WE}$ corrupts at most $n-1$ signing players can forge a valid confirmer signature, an efficient adversary $\mathcal{A}_{MS}$ can be constructed that can break the unforgeability of scheme in [8] or break the hardness of the relation R. When $\mathcal{A}_{WE}$ makes signing queries $\mathcal{A}_{MS}$ works as described in unforgeability. If $\mathcal{A}_{WE}$'s forgery $\sigma^* = \mathsf{S.MAdapt}(\tilde{\sigma}, y)$, $\mathcal{A}_{MS}$ can use $\mathcal{A}_{WE}$ to extract the witness and break the hardness of the relation R. Otherwise, $\mathcal{A}_{WE}$ forge a valid signature σ^* that satisfies $\mathsf{S.MVerify}(apk, m^*, \sigma^*) = 1$, which also a valid signature of multi-signature scheme in [8]. □

3.4. Multi Adaptor Signature Based on Dilithium

For a random matrix $\bar{\mathbf{A}} \leftarrow \mathcal{R}_q^{k\times(l+k)}$, we consider the hard relation $\mathsf{R_d} = \{(\mathbf{X}, \mathbf{x}) | \mathbf{X} = \bar{\mathbf{A}}\mathbf{x}\}$. We denote H_0: $\{0,1\}^* \rightarrow \{c \in \mathcal{R} : ||c||_\infty = 1 \wedge ||c||_1 = \kappa\}$ and COM=(CKGen, Commit, Open) a homomorphic commitment. The scheme $\mathsf{D.MASIG_R}$=(D.MKGen, D.MSign, D.KAgg, D.pMVerify, D.MAdapt, D.MVerify, D.MExt) can be described as follows:

- $(pk_i, sk_i) \leftarrow \mathsf{D.MKGen}(1^\lambda)$.
  Given a random matrix $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times l}$, each signer P_i samples $\mathbf{s}_i \leftarrow S_\eta^{l+k}$ and computes $\mathbf{t}_i \leftarrow \bar{\mathbf{A}}\mathbf{s}_i$, where $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}]$. The secret key $sk_i = \mathbf{s}_i$ and the public key $pk_i = (\bar{\mathbf{A}}, \mathbf{t}_i)$.
- $\langle \tilde{\sigma}/\perp \rangle \leftarrow \mathsf{D.MSign}\Big\langle \{P_i(sk_i, pk_i, m, \mathbf{X})\}_{i\in[n]} \Big\rangle$.
  Given a message m, a list of public keys $\{pk_i\}_{i\in[n]}$ and a statement $\mathbf{X} \in \mathcal{L}_{\mathsf{R_d}}$, each player P_i samples $\mathbf{y}_i \leftarrow D_\mu^{l+k}$, computes $\mathbf{w}_i \leftarrow \bar{\mathbf{A}}\mathbf{y}_i$, $com \leftarrow \mathsf{Commit}_{ck}(\mathbf{w}_i, r_i)$ with $r_i \leftarrow D(S_r)$, and broadcasts com_i. After receiving com_j for all $j \neq i$ and a random $r' \leftarrow D(S_r)$, P_i sets $com \leftarrow \sum_{j\in[n]} com_k$, derives a challenge $c_i \leftarrow \mathsf{H}_0(\mathbf{t}_i, com + \mathsf{Commit}_{ck}(\mathbf{X}, r'), m, \{pk_i\}_{i\in[n]})$ and computes a signature share $\mathbf{z}_i \leftarrow c_i\mathbf{s}_i + \mathbf{y}_i$. Then, P_i runs the rejection sampling on input $(c_i\mathbf{s}_i, \mathbf{z}_i)$, broadcasts $(\mathbf{z}_i, r_i)$ with probability $\min\{1, D_\mu^{l+k}(\mathbf{z}_i)/(M \cdot D_{c_i\mathbf{s}_i,\mu}^{l+k}(\mathbf{z}_i))\}$; otherwise broadcasts Restart. If some player broadcast Restart, P_i restart from sampling $\mathbf{y}_i$; otherwise P_i derives a per-user challenge $c_j \leftarrow \mathsf{H}_0(\mathbf{t}_j, com + \mathsf{Commit}_{ck}(\mathbf{X}, r'), m, \{pk_i\}_{i\in[n]})$, reconstructs $\mathbf{w}_j \leftarrow \bar{\mathbf{A}}\mathbf{z}_j - c_j\mathbf{t}_j$ then checks $||\mathbf{z}_j||_2 \leq B$ and $\mathsf{Open}_{ck}(com_j, r_j, \mathbf{w}_j) = 1$. If the check fails for some j, P_i broadcasts Abort; otherwise, P_i computes $\tilde{\mathbf{z}} \leftarrow \sum_{j\in[n]} \mathbf{z}_j$ and $r \leftarrow \sum_{j\in[n]} r_j$. The signature of m is $\tilde{\sigma} = (com, \tilde{\mathbf{z}}, r, r')$.
- $apk \leftarrow \mathsf{D.KAgg}(\{pk_i\}_{i\in[n]})$.
  The aggregated public key is $\{pk_i\}_{i\in[n]}$.
- $b_1 \leftarrow \mathsf{D.pMVerify}(apk, m, \mathbf{X}, \tilde{\sigma})$.
  Given an aggregated public key $apk = \{pk_i\}_{i\in[n]}$, a message m, an adaptor signature $\tilde{\sigma} = (com, \tilde{\mathbf{z}}, r, r')$, and statement $\mathbf{X} \in \mathcal{L}_{\mathsf{R_d}}$, the algorithm derives a per-user challenge $c_j \leftarrow \mathsf{H}_0(\mathbf{t}_j, com + \mathsf{Commit}_{ck}(\mathbf{X}, r'), m, \{pk_i\}_{i\in[n]})$ and reconstruct $\mathbf{w} = \bar{\mathbf{A}}\tilde{\mathbf{z}} - \sum_{j\in[n]} c_j\mathbf{t}_j$. Then, outputs $b_1 = 1$ if $||\mathbf{z}||_2 \leq B_n$ and $\mathsf{Open}_{ck}(com, r, \mathbf{w}) = 1$.
- $\sigma \leftarrow \mathsf{D.MAdapt}(\tilde{\sigma}, \mathbf{x})$.
  Given an adaptor signature $\tilde{\sigma} = (com, \tilde{\mathbf{z}}, r, r')$, and a witness $\mathbf{x}$, the algorithm outputs the signature $\sigma = (com', \mathbf{z}, r, r')$, where $\mathbf{z} = \tilde{\mathbf{z}} + \mathbf{x}$ and $com' = com + \mathsf{Com}_{ck}(\bar{\mathbf{A}}\mathbf{x}, r')$.
- $b_2 \leftarrow \mathsf{D.MVerify}(apk, m, \sigma)$.
  Given a message m, a signature $\sigma = (com', \mathbf{z}, r, r')$, aggregated public key $apk = \{pk_i\}_{i\in[n]}$, the algorithm derives a per-user challenge $c_j \leftarrow \mathsf{H}_0(\mathbf{t}_j, com', m, \{pk_i\}_{i\in[n]})$ and reconstruct $\mathbf{w} = \bar{\mathbf{A}}\mathbf{z} - \sum_{j\in[n]} c_j\mathbf{t}_j$. Then, outputs $b_1 = 1$ if $||\mathbf{z}||_2 \leq B_n$ and $\mathsf{Open}_{ck}(com', r, \mathbf{w}) = 1$.
- $\mathbf{x}/\perp \leftarrow \mathsf{D.MExt}(\sigma, \tilde{\sigma}, \mathbf{X})$.
  Given an adaptor signature $\tilde{\sigma} = (com, \tilde{\mathbf{z}}, r, r')$, a signature $\sigma = (com', \mathbf{z}, r, r')$, the algorithm can return $\tilde{\mathbf{z}} - \mathbf{z}$ as the witness $\mathbf{x}$.

Theorem 2. $\mathsf{D.MASIG_R}$ = *(*D.MKGen, S.MSign, D.KAgg, D.pMVerify, D.MAdapt, D.MVerify, D.MExt*) is a secure multi-adaptor signature.*

Proof. We now show $\mathsf{D.MASIG_R}$ satisfies pre-signature adaptability, unforgeablity and witness extractability. □

Pre-signature adaptability. If $\mathsf{pMVerify}(apk, m, \mathbf{X}, \tilde{\sigma}) = 1$, which means $\mathsf{Open}_{ck}(com, r, \bar{\mathbf{A}}\tilde{\mathbf{z}} - \sum_{j\in[n]} c_j\mathbf{t}_j) = 1$. For valid pair $(\mathbf{X}, \mathbf{x}) \in \mathsf{R_d}$, we can obtain $\mathsf{Open}_{ck}(com + \mathsf{Com}_{ck}(\bar{\mathbf{A}}\mathbf{x}, r'), r, \bar{\mathbf{A}}(\tilde{\mathbf{z}} + \mathbf{x}) - \sum_{j\in[n]} c_j\mathbf{t}_j) = 1$. It direct implies with $\mathsf{MVerify}(apk, m, (com', \mathbf{z}, r, r')) = 1$, with $\mathbf{z} = \tilde{\mathbf{z}} + \mathbf{x}$, $com' = com + \mathsf{Com}_{ck}(\bar{\mathbf{A}}\mathbf{x}, r')$ and $(com', \mathbf{z}, r, r') = \mathsf{S.MAdapt}(\tilde{\sigma}, \mathbf{x})$. Therefore, the vaild pre-signature can be adapted into a vaild signature.

Unforgeability and witness extractability. The proof is subsumed by the unforgeability and witness extractability proof of $\mathsf{S.MASIG_R}$. If adversary $\mathcal{A}_{DM}$ breaks the unforgeability

or witness extractability of $\mathsf{S.MASIG_R}$, an efficient adversary $\mathcal{A}_{MD}$ can be constructed that can break the unforgeability of scheme in [10] or the hardness of the relation R.

4. Threshold Adaptor Signature

In this section, we present a formal model for threshold adaptor signatures and construct two secure schemes based on Schnorr and Dilithium, respectively.

4.1. Syntax

Definition 6. *A (t, n)-threshold adaptor signature scheme (w.r.t a hard relation* R*)* $\mathsf{TASIG_R} = (\mathsf{TKGen}, \mathsf{TSign}, \mathsf{pTVerify}, \mathsf{TAdapt}, \mathsf{TVerify}, \mathsf{TExt})$ *can be described as follows, and we also give a flowchart of multi-adaptor signature scheme in Figure 2.*

- $\mathsf{TKGen}\left\langle \{P_i(1^\lambda)\}_{i\in[n]} \right\rangle$*: this probabilistic protocol is jointly run by n signing players* $\{P_i\}_{i\in[n]}$ *which takes a security parameter* λ *as public input. The private output of each signing player* P_i *is a signing secret key share* sk_i*, and the public output is the corresponding signing public key pk. The protocol is allowed to abort. In short,* $\left\langle \{P_i(pk, sk_i)\}_{i\in[n]} / \perp \right\rangle \leftarrow \mathsf{TKGen}\left\langle \{P_i(1^\lambda)\}_{i\in[n]} \right\rangle$.
- $\mathsf{TSign}\langle\{P_i(sk_i, pk, m, Y)\}_{i\in S}\rangle$*: this probabilistic protocol is jointly run by a subset* $S \subset \{P_i\}_{i\in[n]}$ *with* $|S| = t + 1$ *to generate a pre-signature* $\tilde{\sigma}$*. Each player* P_i*'s private input is his secret key share* sk_i*, while the public input consists of pk, m and* $Y \in \mathcal{L}_\mathsf{R}$*. The protocol is also allowed to abort. In short,* $\langle \tilde{\sigma} / \perp \rangle \leftarrow \mathsf{TSign}\langle\{P_i(sk_i, pk, m, Y)\}_{i\in S}\rangle$.
- $\mathsf{pTVerify}(pk, m, Y, \tilde{\sigma})$*: on input a public key pk, a statement* $Y \in \mathcal{L}_\mathsf{R}$*, a message* $m \in \{0,1\}^*$ *and a pre-signature* $\tilde{\sigma}$*, this deterministic algorithm outputs a bit* b_1*. In short,* $b_1 \leftarrow \mathsf{pTVerify}(pk, m, Y, \tilde{\sigma})$.
- $\mathsf{TAdapt}(\tilde{\sigma}, y)$*: on input a pre-signature* $\tilde{\sigma}$ *and a witness y, this deterministic algorithm outputs a signature* σ*. In short,* $\sigma \leftarrow \mathsf{TAdapt}(\tilde{\sigma}, y)$.
- $\mathsf{TVerify}(pk, m, \sigma)$*: on input a public key pk, a message m and a signature* σ*, this deterministic algorithm outputs a bit* b_2 *which equals 1 if* σ *is a valid signature. In short,* $b_2 \leftarrow \mathsf{TVerify}(pk, m, \sigma)$.
- $\mathsf{TExt}(\sigma, \tilde{\sigma}, Y)$*: on input a signature* σ*, a statement* $Y \in \mathcal{L}_\mathsf{R}$ *and a pre-signature* $\tilde{\sigma}$*, this deterministic algorithm outputs a witness y such that* $(Y, y) \in \mathsf{R}$*, or* $\perp$*. In short,* $y / \perp \leftarrow \mathsf{TExt}(\sigma, \tilde{\sigma}, Y)$.

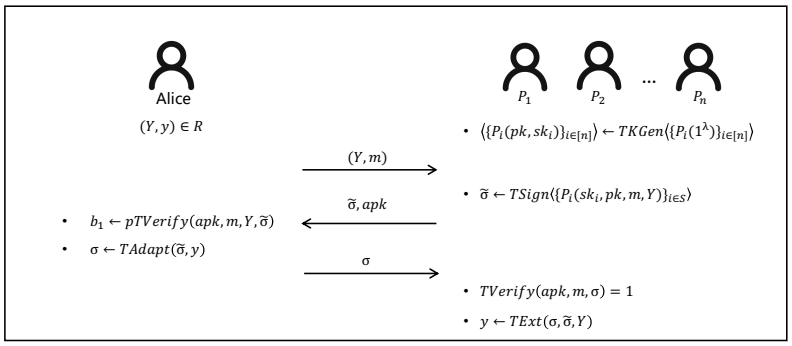

Figure 2. The flowchart of threshold adaptor signature.

Correctness. A threshold adaptor signature should satisfy pre-signature correctness.

Definition 7. *A threshold adaptor signature* $\mathsf{TASIG_R}$ *satisfies pre-signature correctness, if for all* $m \in \{0,1\}^*$*,* pk*,* $\{sk_i\}_{i\in[n]}$ *generated by* TKGen*,* $(Y, y) \in \mathsf{R}$*,* $\tilde{\sigma}$ *generated by* TSign*,* $\sigma \leftarrow \mathsf{TAdapt}(pk, \tilde{\sigma}, y)$*, and* $y' \leftarrow \mathsf{TExt}(pk, \sigma, \tilde{\sigma}, Y)$*, the following holds:*

$$\Pr[\mathsf{pTVerify}(pk, m, Y, \tilde{\sigma}) = 1 \wedge \mathsf{TVerify}(pk, m, \sigma) = 1 \wedge (Y, y') \in \mathsf{R}] \geq 1 - negl(\lambda).$$

4.2. Security Definitions

A secure threshold adaptor signature scheme $\mathsf{TASIG_R}$ should satisfy pre-signature adaptable,unforgeable and witness extractable. We denote Q as the transcript containing all the interactions between adversary $\mathcal{A}$ and $\mathcal{O}_{Tk}$, $\mathcal{O}_{Ts}$, $\mathcal{O}_{Tps}$. When a malicious adversary corrupts at most t signing players query $\mathcal{O}_{Tk}$ and $\mathcal{O}_{Ts}$, it can obtain the views of the protocols TKGen and TSign on input messages $m_1, m_2, \cdots, m_{q_t}$ which the adversary adaptively chose, respectively. $\mathcal{O}_{Tps}$ is a pre-signing oracle that for a message m_j, $j \in [q_s]$, returns a corresponding pre-signature $\langle \tilde{\sigma}_j \rangle \leftarrow \mathsf{TSign}\langle \{P_i(sk_i, pk, m_j, Y)\}_{i\in S}\rangle$.

Definition 8. *A threshold adaptor signature* $\mathsf{TASIG_R}$ *is pre-signature adaptable, if for all* $m \in \{0,1\}^*$, pk, $(Y, y) \in \mathsf{R}$ *and* $\tilde{\sigma} \leftarrow \{0,1\}^*$, *once we have that* $\mathsf{pTVerify}(pk, m, Y, \tilde{\sigma}) = 1$, *then the following holds:*

$$\Pr[\mathsf{TVerify}(pk, m, \mathsf{TAdapt}(pk, \tilde{\sigma}, y)) = 1] \geq 1 - negl(\lambda).$$

Definition 9. *A threshold adaptor signature* $\mathsf{TASIG_R}$ *satisfies unforgeable, if for any PPT adversary* $\mathcal{A}$, *its advantage in the following experiment*

$$A^{uf}_{\mathsf{TASIG_R}} = \Pr[Q = \emptyset; (Y, y) \in \mathsf{R}; (m^*, st) \leftarrow \mathcal{A}^{\mathcal{O}_{Tk}, \mathcal{O}_{Ts}, \mathcal{O}_{Tps}}(1^\lambda); \langle \tilde{\sigma} \rangle \leftarrow$$
$$\mathsf{TSign}\langle \{P_i(sk_i, pk, m^*, Y)\}_{i\in S}\rangle; \sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_{Ts}, \mathcal{O}_{Tps}}(\tilde{\sigma}, Y, st) : m^* \notin Q \wedge \mathsf{TVerify}(pk, m^*, \sigma^*) = 1]$$

is negligible in λ.

Definition 10. *A threshold adaptor signature* $\mathsf{TASIG_R}$ *is witness extractable, if for any PPT adversary* $\mathcal{A}$, *its advantage in the following experiment*

$$\begin{aligned} A^{we}_{\mathsf{TASIG_R}} = \Pr[Q = \emptyset; (m^*, Y^*, st) \leftarrow \mathcal{A}^{\mathcal{O}_{Tk}, \mathcal{O}_{Ts}, \mathcal{O}_{Tps}}(1^\lambda);& \\ \langle \tilde{\sigma} \rangle \leftarrow \mathsf{TSign}\langle \{P_i(sk_i, pk, m^*, Y^*)\}_{i\in S}\rangle; \sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_{Ts}, \mathcal{O}_{Tps}}(\tilde{\sigma}, Y^*, st);& \\ y \leftarrow \mathsf{TExt}(pk, \sigma^*, \tilde{\sigma}, Y^*) : m^* \notin Q \wedge (Y^*, y) \notin \mathsf{R} \wedge \mathsf{TVerify}(pk, m^*, \sigma^*) = 1]& \end{aligned}$$

is negligible in λ.

4.3. Threshold Adaptor Signature Based on Schnorr

Consider a p order cyclic group denoted as $\mathbb{G}$ with generator g, and the discrete logarithm problem in $\mathbb{G}$ is hard. We consider the hard relation $\mathsf{R_s} = \{(Y, y) | Y = g^y\}$ and denote H_s as a random oracle. The scheme $\mathsf{S.TASIG_R}$=(S.TKGen, S.TSign, S.pTVerify, S.TAdapt, S.TVerify, S.TExt) can be described as follows:

– $\left\langle \{P_i(pk, sk_i)\}_{i\in[n]} / \perp \right\rangle \leftarrow \mathsf{S.TKGen}\left\langle \{P_i(1^\lambda)\}_{i\in[n]} \right\rangle$.
Each player P_i, $i \in [n]$ performs Pedersen distributed key generation protocol. After the protocol, each P_i holds a value x_i that is their secret signing share sk_i and a public key $pk = (G, q, g, X)$.

– $\langle \tilde{\sigma} / \perp \rangle \leftarrow \mathsf{S.TSign}\langle \{P_i(sk_i, pk, m, Y)\}_{i\in S}\rangle$.
Let $S \subseteq [n], |S| = t + 1$ be the set of players participating in the signing protocol. Each player P_i can use S to determine the Lagrangian coefficients $\lambda_{i,S}$. Let H_1 be hash functions whose outputs are in $\mathbb{Z}^*_{q_s}$.
Each signing player P_i samples single-use nonces $(d_i, e_i) \leftarrow \mathbb{Z}^*_{q_s} \times \mathbb{Z}^*_{q_s}$, computes commitments $(D_i, E_i) = (g_s^{d_i}, g_s^{e_i})$, then broadcasts (D_i, E_i). When given a message m and a statement $Y \in \mathcal{L}_{\mathsf{R_s}}$, P_i creates the set B, where B is the ordered list of tuples $\langle (j, D_j, E_j) \rangle_{j\in S}$. Then, P_i computes the set of values $\rho_j = \mathsf{H}_1(j, m, B), j \in S$, the group commitment $R = \prod_{j\in S} D_j (E_j)^{\rho_j} \cdot Y$, the challenge $c = \mathsf{H}_s(R, pk, m)$, and $z_i = d_i + (e_i \rho_i) + \lambda_{i,S} sk_i c$. P_i securely deletes $((d_i, D_i), (e_i, E_i))$ from their local storage.

P_i broadcasts z_i. After received z_j, $j \neq i$ from other players, P_i checks the consistency of each z_j. If no check fails, the signature of m is $\tilde{\sigma} = (R, \tilde{z})$, where $\tilde{z} = \sum_{j \in S} z_j$.

– $b_1 \leftarrow \mathsf{S.pTVerify}(pk, m, Y, \tilde{\sigma})$.
Parse $\tilde{\sigma}$ as $(R, \tilde{z})$, and pk as (G, q, g, X), respectively, then compute $c = \mathsf{H}_s(R, X, m)$ and $R' = g^{\tilde{z}} Y X^{-c}$. Output 1 if and only if $R' = R$; otherwise, output 0.
– $\sigma \leftarrow \mathsf{S.TAdapt}(\tilde{\sigma}, y)$.
Given an adaptor signature $\tilde{\sigma} = (R, \tilde{z})$ and a witness y, the algorithm outputs the signature $\sigma = (R, z)$, where $z = \tilde{z} + y$.
– $b_2 \leftarrow \mathsf{S.TVerify}(pk, m, \sigma)$.
Parse σ as (R, z), and pk as (G, q, g, X), respectively, then compute $c = \mathsf{H}_s(R, X, m)$ and $R' = g^{z} X^{-c}$. Output 1 if and only if $R' = R$; otherwise, output 0.
– $y/\perp \leftarrow \mathsf{S.TExt}(\sigma, \tilde{\sigma}, Y)$.
Given an adaptor signature $\tilde{\sigma} = (R, \tilde{z})$, a signature $\sigma = (R, z)$ and a statement, the algorithm can return the witness $y \leftarrow z - \tilde{z}$.

Theorem 3. $\mathsf{S.TASIG_R}$=($\mathsf{S.TKGen}$, $\mathsf{S.TSign}$, $\mathsf{S.pTVerify}$, $\mathsf{S.TAdapt}$, $\mathsf{S.TVerify}$, $\mathsf{S.TExt}$) *is a secure threshold adaptor signature.*

Proof. The proof is subsumed by the security proof of $\mathsf{S.MASIG_R}$ with the only distinction that we need to provide a reduction to the scheme in [9] instead of scheme in [8] for unforgeability and witness extractability. If $\exists$ adversary $\mathcal{A}_{ST}$ can break the unforgeability or witness extractability of $\mathsf{S.TASIG_R}$, an efficient adversary $\mathcal{A}_{TS}$ can be constructed that can break the unforgeability of scheme in [9]. □

4.4. Threshold Adaptor Signature Based on Dilithium

For a random matrix $\bar{\mathbf{A}} \leftarrow \mathcal{R}_q^{k \times (l+k)}$, we consider the hard relation $\mathsf{R_d} = \{(\mathbf{X}, \mathbf{x}) | \mathbf{X} = \bar{\mathbf{A}}\mathbf{x}\}$. We denote and $\mathsf{COM} = (\mathsf{CKGen}, \mathsf{Commit}, \mathsf{Open})$ as a homomorphic commitment and H_0: $\{0,1\}^* \rightarrow \{c \in \mathcal{R} : ||c||_\infty = 1 \wedge ||c||_1 = \kappa\}$, $\mathsf{H}_1 : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$, $\mathsf{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^{l_2}$ as random oracles. The scheme $\mathsf{D.TASIG_R}$=($\mathsf{D.TKGen}$, $\mathsf{D.TSign}$, $\mathsf{D.pTVerify}$, $\mathsf{D.TAdapt}$, $\mathsf{D.TVerify}$, $\mathsf{D.TExt}$) can be described as follows.

– $\left\langle \{P_i(pk, sk_i)\}_{i \in [n]} / \perp \right\rangle \leftarrow \mathsf{D.TKGen}\left\langle \{P_i(1^\lambda)\}_{i \in [n]} \right\rangle$.
Each player P_i samples a random matrix share $\mathbf{A}_i \leftarrow \mathcal{R}_q^{k \times l}$ and generates a commitment $g_i \leftarrow \mathsf{H}_1(\mathbf{A}_i, i)$, broadcasts g_i. After receiving g_j for all $j \neq i$, P_i broadcasts $\mathbf{A}_i$. After receiving g_j for all $j \neq i$, P_i checks $\mathsf{H}_1(\mathbf{A}_j, j) = g_j$. If the check fails for some j, broadcasts Abort; otherwise P_i computes $\mathbf{A} \leftarrow \sum_{j \in [n]} \mathbf{A}_j$ and sets $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in R_q^{k \times (l+k)}$. P_i samples $\mathbf{s}_i \leftarrow S_\eta^{l+k}$ and computes $\mathbf{t}_i \leftarrow \bar{\mathbf{A}}\mathbf{s}_i$, respectively, generates a random oracle commitment $g_i' \leftarrow \mathsf{H}_2(\mathbf{t}_i)$, then broadcasts g_i'. After receiving g_i' for all $j \neq i$, P_i broadcasts $\mathbf{t}_i$. After receiving $\mathbf{t}_j$ for all $j \neq i$, P_i check $\mathsf{H}_2(\mathbf{t}_j, j) = g_j'$. If the check fails for some j, P_i broadcasts Abort; otherwise, the public key $pk = (\mathbf{t}, \mathbf{A})$, and P_i's secret key is $sk_i = \mathbf{s}_i$.
– $\langle \tilde{\sigma} / \perp \rangle \leftarrow \mathsf{D.TSign}\left\langle \{P_i(sk_i, pk, m, \mathbf{X})\}_{i \in [n]} \right\rangle$.
Given a message m and a statement $\mathbf{X} \in \mathcal{L}_{\mathsf{R_d}}$, each player samples $\mathbf{y}_i \leftarrow D_s^{l+k}$ and computes $\mathbf{w}_i \leftarrow \bar{\mathbf{A}}\mathbf{y}_i$, $com \leftarrow \mathsf{Commit}_{ck}(\mathbf{w}_i, r_i)$ with $r_i \leftarrow D(S_r)$, and broadcasts com_i. After receiving com_j for all $j \neq i$ and a random $r' \leftarrow D(S_r)$, P_i sets $com \leftarrow \sum_{j \in [n]} com_k$, derives a challenge $c \leftarrow \mathsf{H}_0(\mathbf{t}, com + \mathsf{Commit}_{ck}(\mathbf{X}, r'), m, pk)$ and computes a signature share $\mathbf{z}_i \leftarrow c\mathbf{s}_i + \mathbf{y}_i$. Then, P_i runs the rejection sampling on input $(c\mathbf{s}_i, \mathbf{z}_i)$, broadcasts $(\mathbf{z}_i, r_i)$ with probability $\min\{1, D_s^{l+k}(\mathbf{z}_i) / (M \cdot D_{c\mathbf{s}_i, s}^{l+k}(\mathbf{z}_i))\}$; otherwise broadcasts Restart. If some player broadcast Restart, P_i restart from sampling $\mathbf{y}_i$; otherwise P_i reconstructs $\mathbf{w}_j \leftarrow \bar{\mathbf{A}}\mathbf{z}_j - c\mathbf{t}_j$ then checks $||\mathbf{z}_j||_2 \leq B$ and $\mathsf{Open}_{ck}(com_j, r_j, \mathbf{w}_j) = 1$. If the check fails for some j, P_i broadcasts Abort; otherwise, P_i computes $\tilde{\mathbf{z}} \leftarrow \sum_{j \in [n]} \mathbf{z}_j$ and $r \leftarrow \sum_{j \in [n]} r_j$. The signature of m is $\tilde{\sigma} = (com, \tilde{\mathbf{z}}, r, r')$.
– $b_1 \leftarrow \mathsf{D.pTVerify}(pk, m, \mathbf{X}, \tilde{\sigma})$.

Given a public key $pk = (\mathbf{t}, \mathbf{A})$, a message m, an adaptor signature $\tilde{\sigma} = (com, \tilde{\mathbf{z}}, r, r')$, and statement $\mathbf{X} \in \mathcal{L}_{\mathsf{R_d}}$, the algorithm derives a challenge $c \leftarrow \mathsf{H}_0(\mathbf{t}, com + \mathsf{Commit}_{ck}(\mathbf{X}, r'), m, pk)$ and reconstructs $\mathbf{w} = \bar{\mathbf{A}}\tilde{\mathbf{z}} - c\mathbf{t}$. Then, outputs $b_1 = 1$ if $\mathsf{Open}_{ck}(com, r, \mathbf{w}) = 1$ and $||\tilde{\mathbf{z}}||_2 \leq B_n$.

– $\sigma \leftarrow \mathsf{D.TAdapt}(\tilde{\sigma}, \mathbf{x})$.
Given an adaptor signature $\tilde{\sigma} = (com, \tilde{\mathbf{z}}, r, r')$, and a witness $\mathbf{x}$, the algorithm outputs the signature $\sigma = (com', \mathbf{z}, r, r')$, where $\mathbf{z} = \tilde{\mathbf{z}} + \mathbf{x}$ and $com' = com + \mathsf{Com}_{ck}(\bar{\mathbf{A}}\mathbf{y}, r')$.

– $b_2 \leftarrow \mathsf{D.TVerify}(pk, m, \sigma)$.
Given a message m, a signature $\sigma = (com', \mathbf{z}, r, r')$, a public key $pk = (\mathbf{t}, \mathbf{A})$, the algorithm derives a challenge $c \leftarrow \mathsf{H}_0(\mathbf{t}, com', m, pk)$ and reconstructs $\mathbf{w} = \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}$. Then, outputs $b_1 = 1$ if $\mathsf{Open}_{ck}(com', r, \mathbf{w}) = 1$ and $||\mathbf{z}||_2 \leq B_n$.

– $\mathbf{x}/\perp \leftarrow \mathsf{D.TExt}(\sigma, \tilde{\sigma}, \mathbf{X})$.
Given an adaptor signature $\tilde{\sigma} = (com, \tilde{\mathbf{z}}, r, r')$, a signature $\sigma = (com', \mathbf{z}, r, r')$, the algorithm can return $\tilde{\mathbf{z}} - \mathbf{z}$ as the witness $\mathbf{x}$.

Theorem 4. $\mathsf{D.TASIG_R}$=(D.TKGen, D.TSign, D.pTVerify, D.TAdapt, D.TVerify, D.TExt) *is a secure threshold adaptor signature.*

Proof. The proof is subsumed by the security proof of $\mathsf{D.MASIG_R}$, differing only in presenting a reduction to the $(n-1, n)$ scheme, instead of the multi-signature scheme in [10] for unforgeability and witness extractability. □

5. Application

In this section, we present further applications for multi-adaptor signature and threshold adaptor signature.

5.1. n to n Atomic Swap

When cryptographic assets from two different blockchain networks need to be exchanged, atomic swaps can be utilized. This technology allows two parties to securely and verifiably exchange without relying on third-party trust.

We have constructed an atomic swap system (Figure 3) using multi-adaptor signature, which includes two entities, the transacting parties U_0 and U_1. To mitigate the risk of economic loss in the event of the loss of a single secret key holding a substantial amount, U_0 and U_1 can distribute cryptographic coins and their corresponding secret keys across n locations.

- Setup: U_0 publicly discloses statement Y, and both U_0 and U_1 invoke algorithm MKGen to generate their keys $\{(pk_{0i}, sk_{0i})\}_{i\in[n]}$ and $\{(pk_{1i}, sk_{1i})\}_{i\in[\bar{n}]}$ for their participation in this transaction.
- Locking Assets: The exchanging parties use a time-lock to restrict the pending currencies, with the time-lock primarily granting U_1 sufficient time to complete the exchange. That also prevents U_0 from withdrawing their currency after withdrawing U_1's currency.
- Generating Transactions: U_0 utilizes MSign to generate a pre-signature $\tilde{\sigma}_0$ for the exchange transaction tx_0 (i.e., U_0 transferring c_0 to U_1) based on statement Y, and invokes MAgg to generate an aggregated key apk_0, then sends $apk_0, \tilde{\sigma}_0$ to U_1. After U_1 verifies $\tilde{\sigma}_0$ by pMVerify, U_1 generate the aggregated key apk_1 and pre-signature $\tilde{\sigma}_1$ for the exchange transaction tx_1 base on Y, where tx_1 is that U_1 transferring c_1 to U_0, then sends $apk_1, \tilde{\sigma}_1$ to U_0.
- Broadcasting Transactions: U_0 verifies $\tilde{\sigma}_1$ and adapts the $\tilde{\sigma}_1$ into a complete signature value σ_1 using MAdapt, then broadcasts σ_1 to obtain c_1. Based on $\tilde{\sigma}_1$ and σ_1, U_1 can extract witness y using MExt, then U_1 adapts $\tilde{\sigma}_0$ into a complete signature value σ_0. U_1 broadcasts σ_0 to obtain the c_0.

Henceforth, n to n atomic swaps are able to be processed in batches simultaneously.

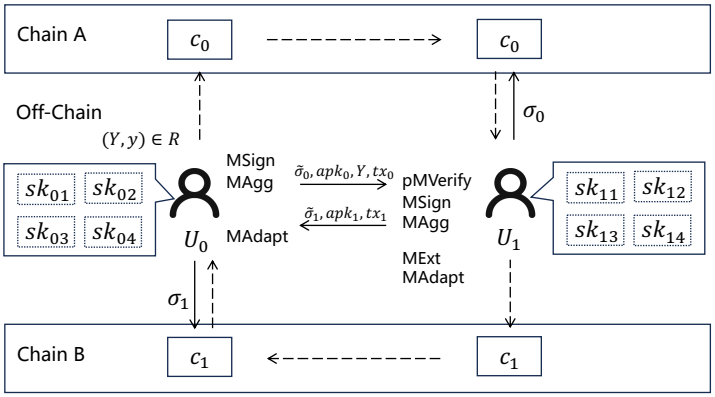

Figure 3. The atomic swap system with multi adaptor signature.

5.2. Oracle-Based Conditional Payment

Oracle-based conditional payments are a financial mechanism or smart contract arrangement that relies on external data oracles to trigger and execute a payment or transaction when specific conditions are met. The conditions can be anything that can be determined or verified by external data, such as the outcome of a sports event, weather conditions, stock prices, or any other event. These conditional payments are commonly used in blockchain and smart contract environments to automate financial agreements based on real-world events or data.

For example, Alice and Bob bet on a sports match. Alice bets USD 30 on team C, with the USD 30 held in escrow by oracles until the end of the game. Oracles serve as an intermediary layer connecting the blockchain with the real world. Once the match results are disclosed, oracles fetch the outcome from external data sources and execute payments based on the results. In the event that team C loses the match, oracles will transfer the USD 30 from Alice's account to Bob's account.

We now give an oracle-based conditional payment system using threshold adaptor signature and verifiable time signature. The system consists of a payer (A), a payee (B), and a set of (t, n) oracles ($\{O_i\}_{i\in[n]}$) utilized as watchtowers.

- Setup: A publishes its statement Y, and each oracle $O_i, i \in [n]$, runs the key generation protocol TKGen to generate pk and its own secret key sk_i.
- Escrowing funds: (t, n) oracles generate a verifiable time signature and send it to A, ensuring that A can redeem its funds after time T if the predefined conditions have yet to be fulfilled. Once A receives the verifiable time signature and checks its validity, A sends its signature to the oracles, which claims that the funds are escrowed in the address pk, then $\{O_i\}_{i\in[n]}$ check its validity.
- Condition monitoring: The oracles continuously monitor to determine whether the predefined conditions are met.
- Payment execution: When the conditions are met before time T, the oracles automatically execute the payment as per the agreed terms. $t + 1$ oracles collectively perform TASIG to generate a pre-signature $\tilde{\sigma}$ and send it to A. A calls TpVerify to verify $\tilde{\sigma}$, and for the valid $\tilde{\sigma}$, A uses its witness y and invokes TAdapt, transforming it into a signature that can be publicly verified. After B receives the signature, B can publish the signature σ and miners utilize TVerify to verify the signature.

We can observe that the securities of the threshold adaptor signature ensure that a single malicious oracle cannot disrupt the payment. Additionally, A has the right to be informed that the funds are being transferred to the payee. Only when A adapts the pre-signature into a publicly verifiable signature can B complete the receipt of funds.

6. Conclusions

Adaptor signatures have lots of applications in cryptocurrencies and blockchain, but may encounter issues such as secret key loss and single-point failure. To address this, we introduce multi-adaptor signatures and threshold adaptor signatures. We propose their security models and give four schemes based on Schnorr and Dilithium, respectively. Finally, two interesting applications are present, demonstrating that multi-adaptor signature and threshold adaptor signature can prevent system disruptions caused by the loss of a single secret key. In the future, for long-term security and broader application scenarios, we will focus on lattice-based (t, n)-threshold adaptor signatures.

Author Contributions: Methodology, Y.J. and B.G.; Writing (original draft preparation), Y.J. and Y.X.; Writing (review and editing), R.Z. and Y.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China (Grant No. 62202458).

Data Availability Statement: Data supporting this study are included within the article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

q	A prime number
$\mathbb{Z}_q$	The interval $[-\frac{q}{2}, \frac{q}{2}) \cap \mathbb{Z}$
Boldface small letters (e.g., $\mathbf{x}$)	Column vectors over $\mathbb{R}$ and $\mathbb{Z}$
Boldface capital letters (e.g., $\mathbf{A}$)	Matrices
$\mathbf{A}_1 \| \mathbf{A}_2$	The concatenation of matrices $\mathbf{A}_1$ and $\mathbf{A}_2$
$\mathcal{R} = \mathbb{Z}[x]/(f(x))$	A polynomial ring for an irreducible monic polynomial $f(x) = x^N + 1$, where N is a power of 2
$\mathcal{R}_q$	$\mathbb{Z}_q[x]/(f(x))$
Λ	A lattice in $\mathbb{Z}^n$
$\langle A(a_o); B(b_o) \rangle \leftarrow \mathsf{Pro}\langle A(a_i); B(b_i) \rangle$	In an interactive protocol Pro between parties A and B, A's (resp. B's) input is a_i (resp. b_i), where A's (resp. B's) output at the end of the execution is a_o (resp. b_o).
$\langle y \rangle \leftarrow \mathsf{Pro}\left\langle \{P_i(x_i)\}_{i \in [n]} \right\rangle$	A protocol with all parties receive the same output y.
$\mathcal{L}_{\mathsf{R}}$	An NP language and R as the associated binary relation

References

1. Decker, C.; Wattenhofer, R. A fast and scalable payment network with bitcoin duplex micropayment channels. In Proceedings of the SSS 2015, Edmonton, AB, Canada, 18–21 August 2015; Springer: Cham, Switzerland, 2015; pp. 3–18. [CrossRef]
2. Malavolta, G.; Moreno-Sanchez, P.; Schneidewind, C.; Kate, A.; Maffei, M. Anonymous multi-hop locks for blockchain scalability and interoperability. In Proceedings of the NDSS 2019, San Diego, CA, USA, 24–27 February 2019. [CrossRef]
3. Aumayr, L.; Ersoy, O.; Erwig, A.; Faust, S.; Hostkov, K.; Maffei, M.; Moreno-Sanchez, P.; Riahi, S. Generalized channels from limited blockchain scripts and adaptor signatures. In Proceedings of the ASIACRYPT 2021, Singapore, 6–10 December 2021; Springer: Cham, Switzerland, 2021; pp. 635–664. [CrossRef]
4. Chaum, D.; Pedersen, T.P. Wallet Databases with Observers. In Proceedings of the CRYPTO 1992, Santa Barbara, CA, USA, 16–20 August 1992; Springer: Berlin/Heidelberg, Germany, 1993; pp. 89–105. [CrossRef]
5. Erwig, A.; Faust, S.; Hostáková, K.; Maitra, M.; Riahi, S. Two-party adaptor signatures from identification schemes. In Proceedings of the PKC 2021, Virtual, 10–13 May 2021; Springer: Cham, Switzerland, 2021; pp. 451–480. [CrossRef]
6. Moreno-Sanchez, P.; Kate, A. Scriptless Scripts with ECDSA. 2018. Available online: https://lists.linuxfoundation.org/pipermail/lightning-dev/attachments/20180426/fe978423/attachment-0001.pdf (accessed on 6 December 2023).
7. Esgin, M.F.; Ersoy, O.; Erkin, Z. Post-quantum adaptor signatures and payment channel networks. In Proceedings of the European Symposium on Research in Computer Security, Guildford, UK, 14–18 September 2020; Springer: Cham, Switzerland, 2020; pp. 378–397. [CrossRef]
8. Nick, J.; Ruffing, T.; Seurin, Y. MuSig2: Simple two-round Schnorr multi-signatures. In Proceedings of the CRYPTO 2021, Virtual, 16–20 August 2021; Springer: Cham, Switzerland, 2021; pp. 189–221. [CrossRef]

9. Komlo, C.; Goldberg, I. FROST: Flexible round-optimized Schnorr threshold signatures. In Proceedings of the SAC 2020, Brno, Czech Republic, 30 March–3 April 2020; Springer: Cham, Switzerland, 2021; Volume 12804, pp. 34–65. [CrossRef]
10. [CrossRef] Damgrd, I.; Orlandi, C.; Takahashi, A.; Tibouchi, M. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *J. Cryptol.* **2022**, *35*, 14. [CrossRef]
11. Thyagarajan, S.A.; Malavolta, G.; Moreno-Sanchez, P. Universal atomic swaps: Secure exchange of coins across all blockchains. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–26 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1299–1316. [CrossRef]
12. Desmedt, Y.; Frankel, Y. Shared generation of authenticators and signatures. In Proceedings of the CRYPTO 1991, Santa Barbara, CA, USA, 11–15 August 1991; Springer: Cham, Switzerland, 1991; Volume 576, pp. 457–469. [CrossRef]
13. Gennaro, R.; Goldfeder, S. Fast multiparty threshold ECDSA with fast trustless setup. In Proceedings of the CCS 2018, Toronto, ON, Canada, 15–19 October 2018; pp. 1179–1194. [CrossRef]
14. Castagnos, G.; Catalano, D.; Laguillaumie, F.; Savasta, F.; Tucker, I. Bandwidth-efficient threshold EC-DSA. In Proceedings of the PKC 2020, Edinburgh, UK, 4–7 May 2020; Springer: Cham, Switzerland, 2020; Volume 12111, pp. 266–296. [CrossRef]
15. Canetti, R.; Gennaro, R.; Goldfeder, S.; Makriyannis, N.; Peled, U. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In Proceedings of the CCS 2020, Virtual, 9–13 November 2020; pp. 1769–1787. [CrossRef]
16. Castagnos, G.; Catalano, D.; Laguillaumie, F.; Savasta, F.; Tucker, I. Bandwidth-efficient threshold EC-DSA revisited: Online/offline extensions, identifiable aborts proactive and adaptive security. *Theor. Comput. Sci.* **2023**, *939*, 78–104. [CrossRef]
17. Bendlin, R.; Krehbiel, S.; Peikert, C. How to share a lattice trapdoor: Threshold protocols for signatures and (H) IBE. In Proceedings of the ACNS 2013, Banff, AB, Canada, 25–28 June 2013; Springer: Cham, Switzerland, 2013; Volume 7954, pp. 218–236. [CrossRef]
18. Nicolosi, A.; Krohn, M.N.; Dodis, Y.; Mazieres, D. Proactive Two-Party Signatures for User Authentication. In Proceedings of the NDSS 2003, San Diego, CA, USA, 27 February–3 March 2003.
19. Bellare, M.; Neven, G. Multi-signatures in the plain public-key model and a general forking lemma. In Proceedings of the CCS 2006, Alexandria, VA, USA, 30 October–3 November 2006; pp. 390–399.
20. Bagherzandi, A.; Cheon, J.H.; Jarecki, S. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Proceedings of the CCS 2008, Alexandria, VA, USA, 27–31 October 2008; pp. 449–458. [CrossRef]
21. Ma, C.; Weng, J.; Li, Y.; Deng, R. Efficient discrete logarithm based multi-signature scheme in the plain public key model. *Des. Codes Cryptogr.* **2010**, *54*, 121–133. [CrossRef]
22. Syta, E.; Tamas, I.; Visher, D.; Wolinsky, D.I.; Jovanovic, P.; Gasser, L.; Gailly, N.; Khoffi, I.; Ford, B. Keeping authorities "honest or bust" with decentralized witness cosigning. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 526–545. [CrossRef]
23. Aumayr, L.; Ersoy, O.; Erwig, L.; Faust, S.; Hostkov, K.; Maffei, M.; Moreno-Sanchez, P.; Riahi, S. Generalized Bitcoin-Compatible Channels. Cryptology ePrint Archive, Report 2020/476. Available online: http://hdl.handle.net/20.500.12708/40215 (accessed on 6 December 2023).