Adaptor Multi-Signatures

A report submitted in fulfilment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering

by

Chaitra S. Gurjar (119CS0006)

under the guidance of

Dr. R. Kabaleeshwaran

April 27, 2023

Department of Computer Science and Engineering
Indian Institute of Information Technology Design and Manufacturing Kurnool
Kurnool, Andhra Pradesh 518008

Evaluation

Title of the Project : Adaptor Multi-Signatures

Name of the Student : Chaitra Gurjar

Examiner:

_____

Supervisor:

_____

Head of the Department:

_____

Date:

Place:

Plagiarism Report

This is to certify that B. Tech. project report entitled **Adaptor Multi-Signatures** submitted by **Chaitra S. Gurjar** holding Roll No. **119CS0006** under the supervision of **Dr. R. Kabaleeshwaran** in the **Department of Computer Science and Engineering** is an original research work done by the student.

The project report has been checked for the Plagiarism and the plagiarism report is submitted along with the project report for further processing.

- Originality content (including the contents from own publications): _____%
- Similarity Reproduction of the content from other sources: _____%

We are aware that any issue related to plagiarism in future will have to be addressed by the student and the concerned supervisor.

Date Student Signature

Certificate

I, **Chaitra S. Gurjar**, with Roll No: **119CS0006**, hereby declare that the material presented in this project report titled ***Adaptor Multi-Signatures*** represents original work carried out by me in the *Department of Computer Science and Engineering* at the *Indian Institute of Information Technology Design and Manufacturing Kurnool* during the years 2022-23.
With my signature, I certify that :

- I have not committed any plagiarism of intellectual property.
- I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the work presented in this report is carried out under my supervision, and is worthy of consideration for the requirements of the degree of Bachelor of Technology.

Advisor Name Advisor Signature

Copyright Transfer

Title of the Project : Adaptor Multi-Signatures

Name of the Student : Chaitra Gurjar

The undersigned hereby assigns to the Indian Institute of Information Technology Design and Manufacturing Kurnool all rights under copyright that may exist in and for the above project report submitted for the award of the B.Tech. degree.

Date Student Signature

Note: However, the author may reproduce or authorize others to reproduce material extracted in exactly the same words from the project report or a derivative of the project report for the personal use of the author given that the source and the Institute's copyright notice are indicated.

Acknowledgement

I would like to extend my sincere gratitude to my mentor *Dr. R. Kabaleeshwaran* for giving me the opportunity to work on this project. As a beginner in the research field, I got constant support, motivation and guidance from my mentor.

I could not have undertaken this journey without Honourable Director, *Prof. D.V.L.N. Somayajulu* ; Head of the Department, *Dr. K. Sathya Babu* ; Faculty Advisor, *Dr. R. Praneetha Sree* ; and all faculty members of the department. I would like to thank them for regular co-ordination, feedback and direction.

Lastly, I would like to mention my friends and family for the continuous encouragement, help and moral support.

Abstract

Adaptor Signature is the new and upcoming cryptographic scheme with important applications in layer-2 and peer-to-peer blockchain applications such as cross-currency swaps and payment channel networks. In this work, we introduce Adaptor Multi-Signatures allowing a group of users to use adaptor signatures. This may be essential for organizations with using multi-signatures for scriptless scripts. We define Adaptor Multi-Signatures to incorporate the amalgam of adaptor signatures and multi-signatures, through the basic Schnorr signature scheme. Schnorr signatures have a shorter signature size which can increase the capacity of a blockchain network, according to the BIP32 Proposal. They also enable more advanced functionality, such as multisignature transactions and threshold signatures, which advance functionality for the Bitcoin network. For Adaptor Multi-Signatures, we make an attempt to prove security properties of this scheme in the Random Oracle Model. We make use of MuSig-2 as a base for multi-signatures, which is a highly practical yet simple two-round multi-signature scheme. Finally, we explore the applications of this adaptor multi-signature for multi-signature wallets and multi-hop locks.

Contents

1 Introduction

Scriptless scripts [1] are a cryptographic technique that enables the execution of complex smart contract functions on blockchain networks without revealing the underlying contract details. Signatures that commit to a hidden value are known as Adaptor Signatures [2]. An adaptor reveals the hidden value when verified with a suitable signature. Alternately, the adaptor verifies the signature only when paired with the hidden value. As a result, adaptor signatures can be used effectively to enable locking in Bitcoin contracts, off-chain atomic swaps, generalized payment channel networks [2] and deterministic wallets [3].

Multi-signatures [4], on the other hand, are signature schemes that enable multiple parties to sign a single message collaboratively. This inherently increases security by making them less vulnerable to attacks by hackers or malicious actors who may attempt to steal private keys. They also allow multiple parties to share control over a single wallet or account. The MuSig-2 [5] scheme incorporates key aggregation method and two-round interactive signing to provide a more secure and efficient method for creating multi-signature transactions on the blockchain. It has the potential to be used in a wide range of applications, such as multi-signature wallets.

Adaptor Multi-Signature combines the features of both adaptor signatures and multi-signatures to create a secure and flexible signature scheme. This allow multiple parties to collaboratively sign a message while also enabling the adaptation of the signature to a hidden commit witness. This scheme relies on the reduction to underlying security models, while building on the advantages of both schemes. Adaptor multi-signatures have several potential applications, such as in multi-signature wallets or in the implementation of multi-hop locks. The purpose of this thesis is to provide concrete definitions and models for the concept of adaptor multi-signatures. Additionally, this will briefly examine the potential applications of adaptor multi-signatures.

1.1 Our Contribution

Recently, there has been a growing interest in multi-signature schemes, which allow multiple parties to collaboratively sign a message while producing only one signature. This makes signing for organizations with multiple parties convinient and verification of this signature easier. The MuSig2 [5] protocol is a notable example of a secure and efficient multi-signature scheme that has gained significant attention

in the cryptographic community. The protocol uses the concept of key aggregation and introduces nonces into the interactive signing protocol, making it practical for real-world applications.

Adaptor signature [1][2], on the other hand, is a relatively new cryptographic scheme that allows for the construction of more flexible and efficient signatures. Adaptor signatures tie message authentication to the disclosure of a secret value called the witness, making them ideal for applications such as payment channels and atomic swaps in blockchain technology. In this thesis, we propose a scheme that combines the MuSig2 protocol with Adaptor Signatures, which we refer to as the Adaptor Multi-Signature protocol. This can be an exciting advancement in the field of multi-signature schemes and adaptor signatures, with numerous potential applications in various fields such as multi-signature wallets and multi-hop locks. Our contribution is summarized as follows :

- We provide formal definitions for adaptor multi-signature scheme and its security properties.
- We provide a concrete instantiation for adaptor multi-signature by combining the Schnorr signature-based adaptor signature [2] and two round optimal Schnorr signature-based multi-signature (MuSig2) [5].
- We describe the potential applications for our adaptor multi-signature scheme.

1.2 Related Work

In their work, "MuSig2:Simple Two-Round Schnorr Multi-Signatures" [5], Nick Jonas *et al.* introduced the MuSig2 scheme in 2021, which enables a group of signers to produce a joint signature on a single message. They suggest MuSig2, a two-round multi-signature method that is straightforward and incredibly useful. It is the first system to simultaneously i) enable key aggregation ii) output standard Schnorr signatures iii) ensure security under concurrent signing sessions iv) use only two communication rounds and v) have a signer complexity that is similar to standard Schnorr Signatures.

The first formalization of definitions of adaptor signatures and their security was provided by Lukas Aumayr *et al.* in their work, "Generalized Channels from Limited Blockchain Scripts and Adaptor Signatures" [2] in 2021. Although this work focuses on generalized channels for off-chain interactions, they rely on adaptor signatures as

their base for this. Hence, they clearly d efine adaptor signatures as a stand-alone primitive and prove its security in terms of cryptographic games.

The generalization of adaptor signatures was constructed by Xianrui Quin *et al.* in their work, "Generic Adaptor Signature" [6] in 2021. A generic adaptor signature can be applied to discrete-logarithm(DL)-based, RSA-based and lattice-based signatures. It also gives the generic construction for blind adaptor signature and linkable ring adaptor signature. It can integrate different cryptocurrencies through a common signature scheme. It can be a tool to increase atomicity and privacy for cryptocurrency.

Lastly, the application of adaptor signatures in deterministic wallets was put forth by Andreas Erwig and Siavash Riahi in their work, "Deterministic Wallets for Adaptor Signatures" [7] in 2022. Adaptor Signatures with re-randomizable keys extends regular adaptor signatures by key re-randomization algorithms. It transforms the existing ECDSA-adaptor signature scheme into an adaptor signature with re-randomizable keys, which can be potentially used in deterministic wallets.

2 Preliminaries

2.1 Notation

A cyclic group $\mathcal{G}$ of order p with a generator g of $\mathcal{G}$ forms the group description as a triple $(\mathbb{G}, p, g)$. λ is the security parameter. The uniform random sampling of x from a set $\mathcal{X}$ is noted using $x \leftarrow \mathcal{X}$. The Schnorr signature scheme $\sum_{Sch}$ for a group $\mathbb{G}$ and the hard relation $\mathcal{R}$ is used as a basis for construction of any scheme. Given a game $Game_A$ parameterized by an adversary $\mathcal{A}$, we define the advantage of $\mathcal{A}$ in $Game_A$ as $Adv_{\mathcal{A}}^{Game}(\lambda) := Pr[Game_A(\lambda) = true]$. For an element $X \in \mathbb{G}$, the $\log_g(X)$ stands for the discrete logarithm of X in base g, i.e., the distinctive $x \in \mathbb{Z}_p$ such that $X = g^x$.

2.2 Definitions

2.2.1 Adaptor Signature

Definition. An adaptor signature scheme consisting of four algorithms $\Xi_{\mathcal{R},\sum} = (PreSign, Adapt, PreVerify, Extract)$ is defined with respec to a hard relation $\mathcal{R}$, a signature scheme $\sum = (ParGen, Signature, Verify)$ with the following rules: $PreSign_{sk}(m, Y)$ is a Probabilistic Polynomial Time algorithm which takes input a private key sk , a message$m \in \{0,1\}^*$ and some statement $Y \in L_{\mathcal{R}}$. It then outputs a pre-sig $\widetilde{\sigma}$; $PreVerify_{pk}(m, Y, \widetilde{\sigma})$ is a Deterministic Polynomial Time algorithm that on input a public key pk , a message $m \in \{0,1\}^*$, the statement $Y \in L_{\mathcal{R}}$ and pre-sig $\widetilde{\sigma}$, outputs a boolean b as true or false ; $Adapt(\widetilde{\sigma}, y)$ is a Deterministic Polynomial Time algorithm that on input a pre-sign $\widetilde{\sigma}$ and witness y corresponding to the statement Y, outputs a full sign σ; and $Extract(\sigma, \widetilde{\sigma}, Y)$ is a Deterministic Polynomial Time algorithm that on input the obtained sign σ, the pre-sign $\widetilde{\sigma}$ and statement $Y \in L_{\mathcal{R}}$, outputs the correct witness y such that (Y, y) are in the hard relation $\mathcal{R}$, or $\perp$.

Security. An adaptor signature scheme$\Xi_{\mathcal{R},\sum}$ is $aEUF\text{–}CMA$ secure, pre-signature adaptable and witness extractable. The formal proof of these security parameter is clearly mentioned in *L.Aumayr et al.*, 2021 [2].

2.2.2 MuSig-2

Definition. The two-round multi-signature scheme $\sum$ consists of the algorithms $(Setup, KeyGen, KeyAgg, (Sign, SignAgg, Sign', SignAgg', Sign''), Ver)$ as follows. System-wide parameters *par* are generated by the setup algorithm $Setup$ taking as input the security parameter. For notational simplicity, we assume that *par* is given as implicit input to all other algorithms.

- The key generation procedure follows a randomized algorithm which takes no explicit inputs and returns a private/public key pair $(sk, pk) \leftarrow KeyGen()$.
- $KeyAgg$ is a key aggregation procedure running deterministically which uses a multiset of public keys $L = pk_1, \ldots pk_n$ and returns an aggregate public key $\widetilde{pk} = KeyAgg$
  $(pk_1, \ldots, pk_n)$.
- Each signer i executes the interactive signature procedure $(Sign, SignAgg,$ $Sign', SignAgg', Sign'')$, which moves through a series of two communication rounds.
  - ⋆ $Sign$ does not require any specific inputs and returns for a signer i, their first-round secret state $state_i$ and a first-round output out_i. The deterministic algorithm $SignAgg$ aggregates the obtained first-round outputs from all signers $(out_1, \ldots out_n)$ in order to broadcast the single first-round output out to all signers.
  - ⋆ Correspondingly, $Sign'$ takes inputs $state_i, out_i$, the private key sk_i of signer i, a message m and additional public keys $(pk_2, \ldots pk_n)$ of all cosigners. out'_i which is this signer's second-round output and $state'_i$ which is some second-round secret state is returned. The deterministic algorithm $SignAgg'$ aggregates the second-round outputs from all signers $(out'_1, \ldots out'_n)$ into a single second-round output out' to be transmitted to all signers. Lastly, $Sign''$ takes $state'_i$, the second-round secret state of signer i and the aggregate second-round output out' to output a signature σ.
- When given an aggregate public key $\widetilde{pk}$, a message m, and a signature σ, the deterministic verification algorithm Ver returns $true$ if the signature is valid and $false$ otherwise.

Security. Given $\sum$, a two-round multi-signature scheme with key aggregation and let $EUF-CMA_{\sum}^{\mathcal{A}}$ be the game defined for $\sum$, then $\sum$ is existentially unforgeable under chosen message attacks for any PPT adversary $\mathcal{A}$. The formal proof of this security parameter is clearly mentioned in *Jonas Nick et al., 2021* [4].

3 Tools

3.1 Construction of Adaptor Signatures [8] [2]

In this section, we review Poelstra's Schnorr-based adaptor signature creation [8]. We need a technique to create pre-signatures that are dependent on the statement Y and reveal the matching witness y once the full signature is published. In order to expand Schnorr signatures to an adapter signature scheme : Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q and let $\mathcal{R} \subseteq \mathbb{G} \times Z_q$ be a relation defined as $\mathcal{R} := \{(Y, y) | Y = g^y\}$. All algorithms in the scheme (as well as those used by the adversary) are implicitly assumed to be parameterized by public parameters $\{(g, p)\}$ and to have access to a random oracle $\mathcal{H} : \{0,1\}^* \to Z_q$. The secret key is sampled $x \leftarrow Z_q$ and returned as x, and the public key is returned as $X = g^x$ by the key generation procedure. The signing algorithm works on a message $m \in \{0,1\}^*$.

PreSign (m, Y) :	**PreVerify** $(m, Y, \widetilde{\sigma})$:	**Adapt** $(\widetilde{\sigma}, y)$:	**Extract** $(\widetilde{\sigma}, \sigma, Y)$:
$k \leftarrow_\$ Z_q$	$K' = g^{\widetilde{s}} X^{-R} Y$	$s = \widetilde{s} + y$	$y' = s - \widetilde{s}$
$K = g^k \cdot Y$	$r' = \mathcal{H}(X \| \| K' \| \| m)$	return $\sigma = (R, s)$	if $(Y, y') \in \mathcal{R}$:
$R = \mathcal{H}(X \| \| K \| \| m)$	if $r = r'$: return 1		return y'
$\widetilde{s} = k + R \cdot x$	else : return 0		else :
return $\widetilde{\sigma} := (R, \widetilde{s})$			return $\perp$

Figure 1: Schnorr-based Adaptor Signature [2]

3.2 Construction of MuSig2 [5]

The group generation technique $GrGen$ and the integer ν, which indicates the number of nonces transmitted by each signer, are the parameters of the multi-signature scheme MuSig2 [5]. It is a simple and highly practical two-round multi-signature protocol. Below is a definition of the scheme :

Parameter Setup $\{Setup\}$: In this phase, the algorithm uses input parameter 1^λ to generate $(\mathbb{G}, p, g)$. It also selects three hash functions $\mathrm{H}_{sig}, \mathrm{H}_{agg}$ and H_{non} from $\{0,1\}^*$ to $\mathbb{Z}_p$. Finally this returns $((\mathbb{G}, p, g), \mathrm{H}_{sig}, \mathrm{H}_{agg}, \mathrm{H}_{non})$.

Key Generation $\{KeyGen\}$: A secret key x sampled from $\mathbb{Z}_q$ is selected by each signer and the respective public key $X = g^x$ is returned.

Key Aggregation $\{KeyAgg$ and $KeyAggCoeff\}$: For a multiset of public keys $L = \{X_1, \ldots X_n\}$ the algorithm finds key aggregation coefficient $a_i = KeyAggCoeff(L, X_i) \forall i$; where the $KeyAggCoeff(L, X)$ function is defined as

: $\mathrm{H}_{agg}(L, X)$. Finally, the aggregate key $\widetilde{X}$ is calculated and returned as $\widetilde{X} = \prod_{i=1}^{n} X_i^{a_i}$.

First Signing Round $\{Sign \text{ and } SignAgg\}$: Before determining the cosigners and the message to sign, each signer may complete the $Sign$ step.

$Sign$: For every nonce $j \in \{1, \dots \nu\}$, every signer (for e.g. for a local signer with index=1) samples $r_{1,j} \leftarrow_{\$} \mathbb{Z}_q$ and computes $R_{1,j} = g^{r_{1,j}}$. It finally returns ν nonces $\{R_{1,1}, \dots R_{1,\nu}\}$.

$SignAgg$: The aggregator receives outputs from 'n' signers $\{(R_{1,1}, \dots R_{1,\nu}), \dots (R_{n,1}, \dots R_{n,\nu})\}$ and calculates $R_j = \prod_{i=1}^{n} R_{i,j}, \forall j$. It finally outputs the set $(R_1, \dots R_\nu)$.

Second Signing Round $\{Sign' \text{ and } SignAgg'\}$: Let $\{x_1, X_1\}$ be the keys of a local signer with index=1 and $X_2 \dots X_n$ be the public keys of other signers. For the message m and the public key multiset $L = \{X_1, \dots X_n\}$:

$Sign'$: The signer calculates aggregate public key $\widetilde{X}$ using the $KeyAgg$ algorithm and the key aggregation coefficient $a_1 = KeyAggCoeff(L, X_1)$. When the output $\{R_1, \dots R_\nu\}$ is received from the first signing round, $b = \mathrm{H}_{non}\left(\widetilde{X}, (R_1, \dots R_\nu), m\right)$ is calculated by the signer. It finally outputs s_1 as follows :

$$
\begin{aligned}
R &= \prod_{j=1}^{\nu} R_j^{b^{j-1}} \\
c &= \mathrm{H}_{sig}\left(\widetilde{X}, R, m\right) \\
r_1 &= \sum_{j=1}^{\nu} r_{1,j} b^{j-1} \\
s_1 &= c a_1 x_1 + r_1
\end{aligned}
$$

$SignAgg'$: Upon receiving $(s_1, \dots s_n)$ from all the signers, it aggregates $s = \sum_{i=1}^{n} s_i \bmod p$. The signature $\sigma = (R, s)$ is the final output.

Verification $\{Verify\}$: Upon receiving $\left(\widetilde{X}, m, \sigma\right)$ the verifier calculates c by using the hash function H_{sig} and the signature is accepted if $g^s = R\widetilde{X}^c$.

4 Adaptor Multi-Signature

4.1 Definition

Let $\mathcal{R}$ be the hard relation for a pair (x, X) which relies on the discrete logarithm hard problem, such that $\mathcal{R} = (x \leftarrow \mathbb{Z}_p; X = g^x)$. The adaptor multi-signature is defined w.r.t. hard relation $\mathcal{R}$ and consists of algorithms $(Setup, KeyGen, StGen, PreSign(Sign, Sign', Sign', SignAgg'), PreVerify, Adapt, Extract)$ with the following syntax. This is defined for n signers and ω witnesses.

- System-wide parameters *par* are generated by the setup algorithm $Setup$ taking as input the security parameter. The randomized $KeyGen()$ takes no input and returns the secret and public key pair (x, X). The randomized $StGen()$ takes no input and returns the statement and witness pair (y, Y).
- The deterministic $KeyAgg()$ takes a multiset of public keys $L = \{X_1, \ldots X_n\}$ and returns an aggregate $\widetilde{X} = KeyAgg(L)$.
- The interactive pre-signature algorithm consisting of $(Sign, SignAgg, Sign', SignAgg')$ is run by each signer i for each witness k.
  - ⋆ $Sign$ does not require any specific inputs and returns for a signer i, their first-round secret states $state_{i,k}$ and first-round outputs $out_{i,k}$. The deterministic algorithm $SignAgg$ aggregates the obtained first-round outputs from all signers $(out_{1,1}, \ldots out_{n,\omega})$ into a set *out* consisting of ω first-round outputs to be sent to all signers.
  - ⋆ Correspondingly, $Sign'$ takes the first-round secret states $state_{i,k}$ of signer i, the aggregate first-round output *out*, the private key x_i, a message m, the statement Y_k and additional public keys $(X_2, \ldots X_n)$ of all cosigners, and returns the second-round output for this signer for each witness $out'_{i,k}$ and a secret second-round state $state_{i,k}$. The deterministic algorithm $SignAgg$ aggregates the received second-round outputs from all signers for a witness into a single second-round output out'_k to be transmitted to all signers. Lastly, the aggregator take the secret states of a signer for a witness $state_{i,k}$ and aggregates it to output the pre-signature $\widetilde{\sigma}$ for each witness.
- When provided with a message m, aggregate public key $\widetilde{X}$, the statement Y_k and a pre-signature $\widetilde{\sigma}$, $PreVerify$ deterministically verifies the signature and

returns true if it is valid and false otherwise.

- *Adapt* is a DPT algorithm that takes the input pre-signature $\widetilde{\sigma}$ and witness y, outputs a signature σ.
- *Extract* is a DPT algorithm that on input a signature σ, pre-signature $\widetilde{\sigma}$ and statement $Y \in L_{\mathcal{R}}$, outputs a witness y such that $(Y, y) \in \mathcal{R}$, or $\perp$.

4.2 Security Definitions

4.2.1 Pre-Signature Adaptability

Theorem 1. *The adaptor multi-signature scheme* $\Xi R_g, \sum_{Sch,MuSig2}$ *satisfies pre-signature adaptability and verification.*

Proof. For an arbitirary witness $y \in \mathbb{Z}_q$, message $m \in \{0,1\}^*$, public key $X \in \mathbb{G}$ and the pre-signature $\widetilde{\sigma} = (R, \widetilde{s}) \in \mathbb{Z}_q \times \mathbb{Z}_q$; we define the statement $Y = g^y$ and $s = \widetilde{s} + y$. Assume that $\text{PreVerify}\,(m, Y, \widetilde{\sigma}) = 1$, then :

$$\begin{aligned} R &= \mathcal{H}\left(X||g^{\tilde{s}}X^{-R}Y||m\right) \\ &= \mathcal{H}\left(X||g^{\widetilde{s}+y}X^{-R}||m\right) \\ &= \mathcal{H}\left(X||g^{s}X^{-R}||m\right) \end{aligned}$$

which implies that $\text{V}erify_{pk}\,(m, \sigma = (R, s)) = 1$. □

4.2.2 Pre-Signature Correctness

Theorem 2. *The adaptor multi-signature scheme* $\Xi R_g, \sum_{Sch,Musig2}$ *satisfies pre-signature correctness.*

Proof. For some $x, y \in \mathbb{Z}_q$ and $m \in \{0,1\}^*$; we define $X = g^x$ and $Y = g^y$. For $\widetilde{\sigma} \leftarrow PreSign\,(m, Y)$ it holds that $R = \mathcal{H}\left(X||g^k.Y||m\right)$ and $\tilde{s} = k + rx$, for some $k \in \mathbb{Z}_q$. Since

$$\begin{aligned} &\mathcal{H}\left(X \left\| g^{\tilde{s}}X^{-r}Y \right\| m\right) \\ =&\mathcal{H}\left(X \left\| g^{k+rx}g^{-xr}Y \right\| m\right) = R \end{aligned}$$

We have $PreVerify_X(m, Y, \widetilde{\sigma}) = 1$. This implies that $Verify_X(m, Y, \sigma) = 1$ for $\sigma = (R, s) := (R, \widetilde{s} + y) = Adapt_X(\widetilde{\sigma}, y)$. Finally,
$Extract(\sigma, \widetilde{\sigma}), Y) = s - \widetilde{s} = (\widetilde{s} + y) - \tilde{s} = y$ which completes the proof. □

4.2.3 Existential Unforgeability

Theorem 3. *Assume that the Schnorr digital signature scheme $\sum_{Sch}$ is SUF–CMA-secure and R_g is a hard relation. The multi-signature scheme MuSig2[GrGen, ν = 4] is EUF-CMA secure in the random oracle model for $H_{agg}, H_{non}, H_{sig} : \{0,1\}^* \to \mathbb{Z}_p$. Then adaptor multi-signature scheme $\Xi_{R_g, \sum_{Sch}, MuSig2}$, as defined in Fig. 2 is EUF–CMA secure.*

Proof. The formal proof and EUF-CMA game of the adaptor multi-signature scheme is defined in Section 4.2. □

4.2.4 Witness Extractability

Theorem 4. *The adaptor multi-signature scheme is witness extractable if for every PPT adversary $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ there exists a negligible function f such that $Pr\left[amWitExt_{\mathcal{A},\sum}(n) = 1\right] \leq f(n)$, where the experiment $amWitExt_{\mathcal{A},\sum}$ is defined as follows.*

$\underline{\mathbf{amWitExt}_{\mathcal{A},\sum}(n):}$	$\underline{\mathcal{O}_S(m):}$	$\underline{\mathcal{O}(m, Y):}$
$Q = \emptyset$	$\sigma \leftarrow Sign_{sk}(m)$	$\widetilde{\sigma} \leftarrow pSign_{sk}(m, Y)$
$(sk, pk) \leftarrow Gen(1^\lambda)$	$Q = Q \cup \{m\}$	$Q = Q \cup \{m\}$
$(m, Y, st) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(pk)$	return σ	return $\widetilde{\sigma}$
$\widetilde{\sigma} \leftarrow pSign(m, Y)$		
$\sigma \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\widetilde{\sigma}, st)$		
return $(Y, Ext(\sigma, \widetilde{\sigma}, Y)) \notin R \wedge m \notin Q \wedge Verify(m, \sigma))$		

Figure 2: Witness Extractability Game for Adaptor Multi-Signatures

Witness Extractability ensures that a witness y may be extracted using a pair of valid signatures and pre-signatures $(\sigma, \widetilde{\sigma})$ for the message and statement (m, Y). Therefore, a malicious verifier cannot construct a legitimate signature using a pre-signature $\widetilde{\sigma}$ without disclosing a witness for Y. Here, the forgery statement Y may be chosen by the adversary. We can thus presume that they know the witness for Y in order to produce a reliable signature for the forged message m. This is not enough to win the experiment, though. Only if a witness for Y is not revealed by a genuine signature, does the adversary win.

5 Construction of Adaptor Multi-Signature

5.1 Scheme

The adaptor multi-signature works with n signers, ν nonces sent by each signer and ω witnesses. It is parameterized by the group generation algorithm GrGen. The scheme is defined as follows (see also Figure 3) :

Parameter Setup {Setup} : In this phase, the algorithm uses input parameter 1^λ to generate $(\mathbb{G}, p, g)$. It also selects three hash functions $\mathrm{H}_{sig}, \mathrm{H}_{agg}$ and H_{non} from $\{0,1\}^*$ to $\mathbb{Z}_p$. Finally this returns $((\mathbb{G}, p, g), \mathrm{H}_{sig}, \mathrm{H}_{agg}, \mathrm{H}_{non})$.

Key and Statement Generation {KeyGen, StGen} : A secret key x sampled from $\mathbb{Z}_q$ is selected by each signer and the respective public key $X = g^x$ is returned. Each witness generates a random secret witness y sampled from $\mathbb{Z}_q$ and the corresponding statement $Y = g^y$.

Key Aggregation {KeyAgg and KeyAggCoeff} : For a multiset of public keys $L = \{X_1, \ldots X_n\}$ the algorithm finds key aggregation coefficient $a_i = \mathit{KeyAggCoeff}(L, X_i) \forall i$; where the $\mathit{KeyAggCoeff}(L, X)$ function is defined as : $\mathrm{H}_{agg}(L, X)$. Finally, the aggregate key $\widetilde{X}$ is returned as $\widetilde{X} = \prod_{i=1}^{n} X_i^{a_i}$.

First Pre-Signature Round {PreSign-Sign and PreSign-SignAgg} : Each signer completes the internal *Sign* an *SignAgg* algorithm for every witness before interacting with the other co-signers.

$\mathit{PreSign} - \mathit{Sign}$: For every nonce $j \in \{1, \ldots \nu\}$ and for every witness $kin\{1, \ldots \omega\}$ a signer (for e.g. for a local signer with index=1) samples $r_{1,j,k} \leftarrow_\$ \mathbb{Z}_q$ and computes $R_{1,j,k} = g^{r_{1,j,k}}$. It finally returns $\{\nu \times \omega\}$ nonces $\{R_{1,1,1}, \ldots R_{1,\nu,\omega}\}$ for the local signer with index=1.

$\mathit{PreSign} - \mathit{SignAgg}$: The aggregator receives outputs from n signers, each corresponding to ω witnesses $\{(R_{1,1,1}, \ldots R_{1,\nu,\omega}), \ldots (R_{n,1,1}, \ldots R_{n,\nu,\omega})\}$. It then calculates $R_{j,k} = \prod_{i=1}^{n} R_{i,j,k}, \forall j \forall k$. It finally outputs the set for each witness k as $R_{set_k} = (R_{1,k}, \ldots R_{\nu,k})$.

Second Pre-Signature Round {PreSign-Sign$'$ and PreSign-SignAgg$'$} : Let $\{x_1, X_1\}$ be the keys of a local signer with index=1 and $X_2 \ldots X_n$ be the public keys of other signers. For the message m and the public key multiset $L = \{X_1, \ldots X_n\}$:

Sign' : The signer calculates aggregate public key $\widetilde{X}$ using the KeyAgg algorithm

Setup $\left(1^{\lambda}\right)$:

$(\mathbb{G}, p, g) \leftarrow \text{GrGen}\left(1^{\lambda}\right)$

Select three hash functions

$\text{H}_{\text{agg}}, \text{H}_{\text{non}}, \text{H}_{\text{sig}} : \{0,1\}^{*} \rightarrow \mathbb{Z}_p$

par $:= \left((\mathbb{G}, p, g), \text{H}_{\text{agg}}, \text{H}_{\text{non}}, \text{H}_{\text{sig}}\right)$

return *par*

KeyGen () :

$x_i \leftarrow_{\$} \mathbb{Z}_p \; ; \; X_i := g^{x_i}$

return (x_i, X_i)

KeyAggCoef (L, X_i) :

return $(\text{H}_{\text{agg}}(L, X_i))$

KeyAgg (L) :

$\{X_1, \ldots, X_n\} := L$

$a_i := \text{H}_{\text{agg}}(L, X_i) \;\; , \forall i$

$\widetilde{X} := \prod_{i=1}^{n} X_i^{a_i}$

return $\left(\widetilde{X}\right)$

PreSign-Sign () :

For local signer with index i=1

$r_{1,j,k} \leftarrow_{\$} \mathbb{Z}_p \;\; , \forall j \forall k$

$R_{1,j,k} := g^{r_{1,j,k}} \;\; , \forall j \forall k$

$out_1 := \{\forall j \forall k, r_{1,j,k}\}$

$state_1 := \{\forall j \forall k, R_{1,j,k}\}$

return $(out_1, state_1)$

PreSign-SignAgg $(out_1, \ldots, out_n)$:

$\{r_{i,j,k}\} := out_i \;\; , \forall i$

$R_{j,k} := \prod_{i=1}^{n} R_{i,j,k} \;\; , \forall j \forall k$

$R_{set_k} := \{\forall j, R_{j,k}\} \;\; , \forall k$

$out := \{\forall k, R_{set_k}\}$

return (out)

StGen () :

$y_k \leftarrow_{\$} \mathbb{Z}_p \; ; \; Y_k := g^{y_k}$

return (y_i, Y_i)

PreSign-Sign′ $(state_1, out, x_1, m, (X_2 \ldots X_n))$:

$b_k = \text{H}_{\text{non}}(L || R_{set_k} || m) \;\; , \forall k$

$R_k = \left(\prod_{j=1}^{\nu} R_{j,k}^{b_k^{j-1}}\right) \cdot Y_k \;\; , \forall k$

$c_k = \text{H}_{\text{sig}}\left(\widetilde{X} || R_k || m\right) \;\; , \forall k$

$r_{1,k} = \sum_{j=1}^{\nu} r_{1,j,k} b_k^{j-1} \;\; , \forall k$

$s_{1,k} = c_k a_1 x_1 + r_{1,k} \;\; , \forall k$

return $\left(R_k, s_{1,k}\right)$

PreSign-SignAgg′ $\left((s_{1,k} \ldots s_{n,k}), R_k\right)$:

$\widetilde{s_k} := \sum_{i=1}^{n} s_{i,k}$

return $(\widetilde{\sigma} = (R_k, \widetilde{s_k}))$

PreVerify $\left(\widetilde{X}, m, \widetilde{\sigma}\right)$:

$(R_k, s_k) := \widetilde{\sigma}$

$c_k = \text{H}_{\text{sig}}\left(\widetilde{X}, R_k, m\right)$

return $\left(g^{s_k} = R_k \widetilde{X}^{c_k} \cdot Y^{-1}\right)$

Adapt $(\widetilde{\sigma}, y_k)$:

$(R_k, s_k) := \widetilde{\sigma}$

$s_k = \widetilde{s_k} + y$

return $(\sigma = (R_k, s_k))$

Extract $(\widetilde{\sigma}, \sigma, Y_k)$:

${y_k}' = s_k - \widetilde{s_k}$

if $\left(Y_k, {y_k}'\right) \in \mathbb{R}$: return ${y_k}'$

else : return $\perp$

Figure 3: Adaptor Multi-Signature Scheme

and the key aggregation coefficient $a_1 = KeyAggCoeff(L, X_1)$. When the output $\{R_{set_1}, \ldots R_{set_\omega}\}$ is received from the first signing round, $b_k = \text{H}_{non}\left(\widetilde{X}, R_{set_k}, m\right)$

is calculated by the signer for each witness. The statement Y is integrated to finally output $s_{1,k}$ as follows :

$$
\begin{aligned}
R_k &= \prod_{j=1}^{\nu} R_{j,k}^{b_k^{j-1}} \cdot Y_k \ , \forall k \\
c_k &= \mathrm{H}_{sig}\left(\widetilde{X}, R_k, m\right) \ , \forall k \\
r_{1,k} &= \sum_{j=1}^{\nu} r_{1,j,k} b_k^{j-1} \ , \forall k \\
s_{1,k} &= c_k a_1 x_1 + r_{1,k} \ , \forall k
\end{aligned}
$$

$SignAgg'$: Upon receiving $(s_{1,1} \dots s_{1,\omega}, \dots s_{n,1} \dots s_{n,\omega})$ from all the signers, it aggregates $\widetilde{s_k} = \sum_{i=1}^{n} s_{i,k} \bmod p$. The pre-signature $\widetilde{\sigma} = (R_k, \widetilde{s_k})$ is the final output.

Verification {Pre-Verify} : Upon receiving $\left(\widetilde{X}, m, \widetilde{\sigma}\right)$ the verifier calculates c_k by using the hash function H_{sig} and the signature is accepted if $g^{s_k} = R_k \widetilde{X}_k^c \cdot Y^{-1}$.

Adapt {Adapt} : Each witness y_k uses the pre-signature $\widetilde{\sigma}$ to adapt it into a full signature $\sigma = (R_k, \widetilde{s_k} + y)$.

Extract {Extract} : Each witness uses the pre-signature $\widetilde{\sigma}$ and the full signature σ to extract the committed hidden value $y_k = (\sigma - \widetilde{\sigma})$.

5.2 Security

The overall idea to prove the security of Adaptor Multi-signatures is to reduce it into the $EUF - CMA$ security of Adaptor Signatures and MuSig-2. In the reduction, a PPT adversary, $\mathcal{A}$ needs to simulate a pre-signature on the target message m for which a successful simulator $\mathcal{B}$ will later produce a forgery. . Concretely, upon a pre-sign query by $\mathcal{A}$ on some message m, the simulator $\mathcal{B}$ forwards this message to its own signing oracle and sends the resulting full signature back to $\mathcal{A}$. This is acheived by the similator $\mathcal{B}$ querying the underlying MuSig-2 signature oracle on the message m. Signature unforgeability must hold even if the adversary learns a pre-signature for m without being aware of a witness for Y, hence it is critical to permit the adversary to learn a pre-signature on the forgery message m. When $\mathcal{A}$ returns a full signature for m as its forgery, $\mathcal{B}$ can only use this forgery to break the strong unforgeability of MuSig-2. Hence the adaptor multi-signature is $EUF - CMA$ secure for a PPT adversary $\mathcal{D} = \{\mathcal{A}, \mathcal{B}\}$ such that there exists a negligible function f for which $\Pr\left[amSignForge_{\mathcal{D},\sum}(n) = 1\right] \leq f(n)$, where the

game $amSignForge_{\mathcal{D},\sum}$ is defined as follows.

$\underline{\text{amSignForge}^{\mathcal{D}}_{\sum}(\lambda)}$:

$ctrs := 0$

$S := \emptyset$

$S' := \emptyset$

$Q := \emptyset$

$(x_1, X_1) \leftarrow \text{K}eyGen()$

$(y, Y) \leftarrow \text{S}tGen()$

$((x_2, X_2) \dots (x_n, X_n)) \leftarrow \mathcal{A}^{KeyGen()}$

$L = \{((x_1, X_1) \dots (x_n, X_n))\}$

$(m^*, L) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{P}reSign}}(L, Y)$

$\widetilde{\sigma} \leftarrow \text{P}reSign(m^*, Y, L)$

$\sigma^* \leftarrow \mathcal{A}^{\text{P}reSign,\text{M}Sign}(\widetilde{\sigma}, st)$

$b := \text{V}erify(m^*, \sigma^*)$

return $b \wedge ((m^*, L) \in Q) \wedge (X_1 \in L)$

$\underline{\mathcal{O}_{\text{P}reSign}(m, L, Y)}$:

$\widetilde{\sigma} \leftarrow \mathcal{B}^{\mathcal{O}_{\text{Sign},\text{SignAgg},\text{Sign}',\text{SignAgg}'}}(m, L, Y)$

return $\widetilde{\sigma}$

$\underline{\mathcal{O}_{\text{M}Sign}(m, L)}$: :

$\sigma \leftarrow \mathcal{B}^{\mathcal{O}_{\text{M}uSig2}}(m, L)$

return $\widetilde{\sigma}$

$\underline{\mathcal{O}_{\text{Sign}}()}$:

$ctrs := ctrs + 1$

$k := ctrs; S := S \cup \{k\}$

$\forall j, r_{1,j} \leftarrow_{\$} \mathbb{Z}_p$

$\forall j, R_{1,j,k} := g^{1,j}$

$out_1 := \{\forall j, r_{1,j}\}$

$state_1 := \{\forall j, R_{1,j}\}$

return (out_1)

$\underline{\mathcal{O}_{SignAgg}(\forall i, out_i)}$:

$\forall j, R_j = \prod_{i=1}^{n} R_{i,j}$

$R_{set} := \{R_1, \dots R_\nu\}$

$out := R_{set}$

return (out)

$\underline{\mathcal{O}_{\text{Sign}'}(state_1, out)}$:

if $k \notin S$: return $\perp$

$b = \text{H}_{\text{non}}(L||R_{set}||m)$

$R = \prod_{j=1}^{\nu} R_j^{b^{j-1}}$

$c = \text{H}_{\text{sig}}\left(\widetilde{X}||R||m\right)$

$r_1 = \sum_{j=1}^{\nu} r_{1,j} b^{j-1}$

$s_1 = ca_1 x_1 + r_1$

$Q := Q \cup \{L, m\}$

$S := S \backslash \{k\}$

$S' := S' \cup \{k\}$

return (R, s_1)

$\underline{\mathcal{O}_{SignAgg'}(R, (s_1, \dots s_n))}$:

if $k \notin S'$: return $\perp$

$\widetilde{s} := \sum_{i=1}^{n} s_i$

$S' := S' \backslash \{k\}$

return $(\widetilde{\sigma} = (R, \widetilde{s}))$

Figure 4: EUF-CMA Security Reduction for Adaptor MultiSignature

6 Applications

6.1 Multi-Signature Wallets

Multisignature wallets are a type of digital wallet used for storing cryptocurrency that requires multiple signatures or approvals in order to authorize a transaction. The use of multisignature wallets is growing in popularity due to their enhanced security features, as they require multiple parties to sign off on a transaction before it can be executed. This reduces the risk of unauthorized access to the funds held in the wallet, as it requires the collusion of multiple parties to carry out a fraudulent transaction. One of the main advantages of multisignature wallets is that they provide greater control over the custody and management of funds. This is particularly useful in the context of organizations or businesses that require multiple parties to have access to funds or assets. In the context of adaptor signatures, deterministic wallets [7] have been proposed as a means of generating adaptor signatures in a more efficient and secure manner. By using adaptor deterministic wallets [3], signatures can be generated in a way that is completely deterministic and requires no interaction between the parties involved. Hence, using Adaptor Multi-Signature for deterministic wallets can be a novel potential application.

6.2 Multi-Hop Locks from Scriptless Scripts

Multi-hop locks are protocols that allow two parties to exchange coins and proof of payment without requiring a mutual funding Multi-Sig output. Instead, they are connected by hopping through intermediate nodes who are connected by having a shared funding Multi-Sig output. This builds on the concept of adaptor signatures, a type of signature scheme that enables off-chain execution of smart contracts without revealing the transaction details on the blockchain. Hence, the amalgam of Adaptor Multi-Signature can be a useful tool in executing these multi-hop locks.

7 Conclusion

In conclusion, Adaptor Multi-Signatures (AMS) is a promising area of research in the field of cryptography that has the potential to greatly enhance the security and privacy of multi-party transactions on blockchain networks. By combining the concepts of adaptor signatures and multi-signatures, AMS enables off-chain execution of multi-party transactions without revealing any sensitive information on the blockchain.

AMS has several advantages over existing multi-signature schemes. Firstly, it provides enhanced privacy and security by preventing the exposure of sensitive transaction details on the blockchain. Secondly, it enables the execution of complex multi-party transactions with greater efficiency and flexibility. Thirdly, it can be combined with other cryptographic primitives such as scriptless scripts to create even more powerful and versatile smart contracts.

Despite the potential benefits of AMS, there are still several challenges that need to be addressed in order to fully realize its potential. These include the need for standardization and interoperability across different blockchain networks, as well as the development of efficient and user-friendly tools for implementing and executing AMS.

Nevertheless, the recent advancements in AMS research have shown great promise in addressing these challenges, and we can expect to see further developments and innovations in this field in the coming years. Overall, AMS represents a significant step forward in the quest for more secure and efficient multi-party transactions on blockchain networks, and we look forward to seeing its continued progress in the future.

References

[1] P. Moreno-Sanchez and A. Kate, "Scriptless scripts with ecdsa," *Lightning-dev mailing list*, 2018.

[2] L. Aumayr, O. Ersoy, E. Andreas, and S. Riahi, "Generalized channels from limited blockchain scripts and adaptor signatures," *Advances in Cryptology–ASIACRYPT 2021*, 2021. [Online]. Available: https://eprint.iacr.org/2020/476.pdf.

[3] A. Erwig and S. Riahi, "Deterministic wallets for adaptor signatures," *Computer Security–ESORICS 2022*, 2019. [Online]. Available: https://eprint.iacr.org/2019/698.pdf.

[4] J. Nick, R. Tim, S. Yannick, and W. Pieter, "Musig-dn: Schnorr multi-signatures with verifiably deterministic nonces," *2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020. [Online]. Available: https://eprint.iacr.org/2020/1057.pdf.

[5] J. Nick, R. Tim, and S. Yannick, "Musig2: Simple two-round schnorr multi-signatures," *Advances in Cryptology–CRYPTO 2021*, 2021. [Online]. Available: https://eprint.iacr.org/2020/1261.pdf.

[6] X. Qin, C. Handong, and H. Y. Tsz, "Generic adaptor signature," *Cryptology ePrint Archive 2021*, 2021. [Online]. Available: https://eprint.iacr.org/2021/161.pdf.

[7] P. Das, S. Haust, and J. Loss, "A formal treatment of deterministic wallets," *2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019. [Online]. Available: https://eprint.iacr.org/2019/698.pdf.

[8] A. Poelstra, "Scriptless scripts," *Presentation Slides*, 2017. [Online]. Available: https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf.

[9] A. Erwig, S. Faust, H. Kristina, and M. Monosij, "Two-party adaptor signatures from identification schemes," *Public-Key Cryptography–PKC 2021: 24th IACR International Conference on Practice and Theory of Public Key Cryptography*, 2021. [Online]. Available: https://eprint.iacr.org/2021/150.pdf.

Adaptor Multi-Signatures

ORIGINALITY REPORT

4%
SIMILARITY INDEX

PRIMARY SOURCES

1	www.coursehero.com Internet	157 words — 3%
2	idr-lib.iitbhu.ac.in:8080 Internet	45 words — 1%
3	quizengine.blogspot.com Internet	19 words — < 1%
4	iiitk.irins.org Internet	18 words — < 1%
5	Yang Yang, Ximeng Liu, Robert H. Deng, Yingjiu Li. "Lightweight Sharable and Traceable Secure Mobile Health System", IEEE Transactions on Dependable and Secure Computing, 2020 Crossref	10 words — < 1%

EXCLUDE QUOTES	ON	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	ON	EXCLUDE MATCHES	OFF