Digital Communications and Networks 11 (2025) 689–699

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Adaptor signature based on randomized EdDSA in blockchain

Yixing Zhu [a,*], Huilin Li [a], Mengze Li [b], Yong Yu [a]

[a] School of Computer Science, Shaanxi Normal University, Xi'an 710062, China
[b] School of Economics, Shandong Normal University, Jinan 250358, China

ARTICLE INFO

Keywords:
Blockchain
Adaptor signature
Randomized EdDSA
Payment channel

ABSTRACT

Adaptor signature, a new primitive that alleviates the scalability issue of blockchain to some extent, has been widely adopted in the off-chain payment channel and atomic swap. As an extension of standard digital signature, adaptor signature can bind the release of a complete digital signature with the exchange of a secret value. Existing constructions of adaptor signatures are mainly based on Schnorr or ECDSA signature algorithms, which suffer low signing efficiency and long signature length. In this paper, to address these issues, we propose a new construction of adaptor signature using randomized EdDSA, which has Schnorr-like structure with higher signing efficiency and shorter signature length. We prove the required security properties, including unforgeability, witness extractability and pre-signature adaptability, of the new adaptor signature scheme in the random oracle model. We conduct a comparative analysis with an ECDSA-based adaptor signature scheme to demonstrate the effectiveness and feasibility of our new proposal.

1. Introduction

Blockchain technology, which first emerged in 2008, has garnered great interest from both academia and industry. It is often hailed as one of the most disruptive innovations in the fourth industrial revolution. It is equipped with the properties of decentralization, tamper resistance, transparency and robustness. Due to these characteristics, blockchain is widely used in diverse fields including cloud storage [1], the internet of things [2], e-voting [3] and financial services [4,5]. The adoption of cryptocurrencies has further brought this technology into the public spotlight. Currently, well-established cryptocurrencies like Bitcoin, Ether,[1] etc., maintain a steadfast presence in the cryptocurrency market, consistently attracting both investors and the general public. In fact, the number of bitcoin transactions increased by approximately 35% in 2023, reaching a peak of more than 560,000 transactions per day[2] in April of that year. However, the rapid development of blockchain inevitably raises the scalability issue. The underlying consensus mechanism constrains transaction rates, resulting in a few number of transactions completed per second, which falls significantly short of practical demands. Currently, the majority of existing blockchain systems grapple with the challenge of low transaction throughput. As we know, bitcoin processes approximately 10 transactions per second, whereas other payment networks like Visa can handle a maximum of 4,000 transactions per second.

To tackle the scalability challenge in blockchain, a variety of solutions have been proposed, including both on-chain and off-chain approaches. On-chain method generally adopts a sharding strategy, which divides the network into fragments so that more transactions are processed simultaneously. The proposal of payment channels provides a comprehensive off-chain method to improve transaction efficiency. The payment channel denotes the creation of a new channel off-chain that specifically facilitates small and frequent payments between both parties without the need to go through the blockchain network. When a payment channel is opened, it solely requires interaction with the blockchain network during establishment and closure. This peer-to-peer off-chain transaction [6] greatly improves transaction output.

In 2017, Poelstra proposed the concept of scriptless scripts [7] as an extension of standard digital signature. Later, scriptless scripts were formalized as Adaptor Signatures (AS) and widely used to construct high-efficiency payment channels. In short, adaptor signatures bind the release of a complete signature with the exchange of a secret value. Firstly, the signer generates a pre-signature under their own key and sends it to the publisher of the secret value. The publisher embeds the secret value into pre-signature to generate a complete signature and

* Corresponding author.
E-mail addresses: zhuyixing@snnu.edu.cn (Y. Zhu), lihuilinlin@163.com (H. Li), LeMengze@163.com (M. Li), yuyong@snnu.edu.cn (Y. Yu).

[1] https://ethereum.org/zh/whitepaper/.
[2] https://www.blockchain.com/explorer.

https://doi.org/10.1016/j.dcan.2024.06.004
Received 18 January 2024; Received in revised form 21 May 2024; Accepted 6 June 2024
Available online 12 June 2024

posts the complete signature on-chain. Once the signer discovers the complete signature, he can extract the secret value from it and complete the exchange. With adaptor signature, only the publisher possessing the secret value can transform the proper pre-signature into a complete signature, and only the signer with access to both the pre-signature and complete signature can extract the secret value. Another method for constructing payment channel is to use Hash Time Lock Contract (HTLC) in lightning networks [8]. Compared with HTLC, the payment channel constructed by adaptor signature can resist wormhole attacks [9], and can provide enhanced security, privacy, and interoperability guarantees. In addition, adaptor signatures can also be used to design atomic swap [10] between different chain and Payment Channel Networks (PCNs).

Currently, adaptor signature schemes based on Schnorr and ECDSA have been developed to construct payment channels to alleviate the blockchain scalability issue. With the continuous development of blockchain and cryptocurrency, the Edwards Curve Digital Signature Algorithm (EdDSA) proposed by Bernstein [11] in 2012 has become one of the popular underlying signature algorithms for cryptocurrency, and about a quarter of cryptocurrencies, including Stellar,[3] Cardano[4] and Decred,[5] are using EdDSA due to its efficiency and security features. EdDSA [12] is a form of Schnorr-like signature scheme that uses potentially twisted Edwards curves. Compared with ECDSA, EdDSA has the following advantages: (1) EdDSA, with its Schnorr-like signature structure and use of twisted Edwards curves, significantly improves the efficiency of signing and verification; (2) EdDSA utilizes compact public keys, typically 32 or 57 bytes in size, and concise signatures, which are usually 64 or 114 bytes in length for Ed25519 and Ed448, respectively. This reduction in size substantially reduces storage requirements. However, due to its deterministic signature, EdDSA cannot be directly used to construct adaptor signatures because of the risk of leaking secret value [13]. Therefore, we utilize randomized EdDSA signature [14] without sacrificing efficiency. This approach allows us to develop a novel randomized EdDSA-based adaptor signature scheme, which can be applied to construct a more efficient and secure payment channel. Specifically, the signer and the intermediate nodes generate a pre-signature using the randomized EdDSA adaptor signature and lock the transaction. Subsequently, the publisher and the intermediate nodes use the secret to adapt the pre-signature to a full signature, gradually releasing the transaction until the signer obtains the secret.

1.1. Our contributions

Our main goal is to construct a randomized EdDSA-based adaptor signature scheme tailored for blockchain system.

- We propose a more efficient adaptor signature scheme based on randomized EdDSA signature, and provide a specific construction. It can be used to construct efficient and secure PCNs, alleviating blockchain scalability issues.
- We prove the correctness of the rEdDSA-based adaptor signature scheme, and demonstrate that the proposed scheme satisfies essential security properties, including pre-signature adaptability, unforgeability, and witness extractability.
- We evaluate computation cost and communication overhead of the proposed scheme, and conduct comparative experiments with the ECDSA-based scheme to analyze its effectiveness.

1.2. Related works

Poelstra [7] first introduced the concept of scriptless script in 2017. It is created solely through the faithful execution of a smart contract, offering higher privacy and security guarantees. In 2018, Malavolta et al. [9] implemented an Anonymous Multi-Hop Lock (AMHL) using scriptless Schnorr-based and ECDSA-based construction, respectively. And they demonstrated that AMHL with scriptless script offers higher efficiency than HTLC and provides enhanced security, privacy, and scalability.

In 2019, Fournier et al. [15] formalized adaptor signatures for the first time and proposed a one-time verifiably encrypted signatures for Bitcoin layer-2 protocol by using the concept of adaptor signature. To strengthen the definition and broaden its applicability, Aumayr et al. [6] formalized the adaptor signature as an independent primitive in 2020. Employing an ECDSA-based adaptor signature scheme, they constructed generalized channels, enabling users to execute any operations supported by the underlying blockchain in an off-chain manner. Subsequently, Erwig et al. [13], in 2021, proposed a generic adaptor signature transformation from Identification Schemes (ID), and proposed generic transformation for two-party aggregatable adaptor signatures based on ID, filling the gap in the generic construction of ID-based adaptor signatures. In 2022, Dai et al. [16] presented an unlinkable adaptor signature construction that can be derived from any signature scheme and any strongly random-self reducible relation. In 2023, Bao et al. [17] put forward a new identity-based adaptor signature scheme, demonstrating potential applications in blockchain systems. Following that Zhu et al. [18] devised a secure two-party adaptor signature based on IEEE P1363 standard identity-based signature.

Various approaches have been proposed to enhance post-quantum security, aiming to maintain security even in scenarios where adversaries have access to quantum computers. Esgin et al. [19] proposed the initial post-quantum adaptor signature based on the lattice hypothesis of Module-SIS and Module-LWE. Essentially, this signature demonstrates efficiency comparable to that of a standard lattice-based signature. Tairi et al. [20] designed an adaptor signature based on isogenies, formally proving the security of IAS against quantum adversaries. The above two works also illustrate the process of building post-quantum secure PCNs.

1.3. Organization

This paper is structured as follows. In section 1, we underscore the significance of adaptor signatures in addressing the scalability challenges encountered by blockchain, and then we review the related work of adaptor signature and summarize our contributions. In section 2, we present some cryptographic preliminaries that we employ. Section 3 reviews the basic concept and security model of adaptor signatures. In section 4, we describe the detailed construction and security proof of the proposed scheme. And we evaluate the computation cost and communication overhead of the proposed scheme in section 5, and finally we conclude and summarize our findings in section 6.

2. Preliminaries

2.1. Notation

We denote 1^n as security parameter of the proposed scheme. $x \xleftarrow{R} \mathcal{X}$ is denoted as uniform sampling a value x from set $\mathcal{X}$. We define the Probabilistic Polynomial Time (PPT) function $y \leftarrow \mathsf{F}(x)$, where x is the input and y is the output. The Deterministic Polynomial Time (DPT) algorithm is represented as $y := \mathsf{F}(x)$. Finally, we denote function u: $\mathbb{N} \rightarrow \mathbb{R}$ is negligible, if for every positive polynomial poly $(\cdot)$ there exists an integer $N_{poly} > 0$ such that for all $n > N_{poly}$, the following inequality holds

$$|u(n)| < \frac{1}{poly(n)}$$

[3] https://www.stellar.org/developers/guides/security.html.
[4] https://docs.cardano.org/.
[5] https://www.decred.org/.

2.2. Digital signature

Digital signature [13] (in a certain message space $\mathcal{M}$) generally uses triples $\mathsf{SIG}=(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, where (i) $\mathsf{KeyGen}\ (1^n)$ represents a PPT algorithm that receives the security parameter 1^n as input and generates a secret and public key pair (sk, pk), where sk is the signing key and pk is the verification key; (ii) $\mathsf{Sign}(m, sk)$ is a PPT or DPT algorithm that takes a message $m \in \mathcal{M}$ and secret key sk as inputs and outputs the signature σ; (iii) $\mathsf{Verify}(m, pk, \sigma)$ is a DPT algorithm that takes public key pk, message $m \in \mathcal{M}$ and signature σ as input, producing an output $b \in \{0,1\}$. If σ passes the verification, the algorithm outputs 1, otherwise it outputs 0. Moreover, if Sign is a DPT algorithm, we call SIG is *deterministic*. If there is only one signature σ for a message m, then SIG is a *unique signature*.

The fundamental security property of a digital signature is unforgeability, which is also known as (strong) existential unforgeability under chosen message attack (EUF–CMA or SUF–CMA). EUF–CMA means that a PPT adversary is unable to forge a valid signature for a new message m even if it has a public key pk and accesses the signing oracle. SUF–CMA means that it is unable to forge a valid signature for message m despite possessing the aforementioned abilities, and SUF–CMA provides a stronger security guarantee.

2.3. Hard relation

Let $\mathsf{R} \subseteq \mathcal{H}_s \times \mathcal{H}_w$ represent a relation of statement/witness pairs $(Y, y) \subseteq \mathcal{H}_s \times \mathcal{H}_w$, and define language $L_{\mathsf{R}} \subseteq \mathcal{H}_s$ associated to R as $L_{\mathsf{R}} := \{Y \subseteq \mathcal{H}_s | \ \exists y \in \mathcal{H}_w \ s.t. \ (Y, y) \in \mathsf{R}\}$. If R is a hard relation, then (i) there exists a PPT algorithm $\mathsf{Gen}(1^n)$ that takes security parameter as input and outputs a pair $(Y, y) \in \mathsf{R}$; (ii) The relation R is poly-time decidable; (iii) Given a statement $Y \in L_{\mathsf{R}}$, the probability that any PPT adversary $\mathcal{A}$ outputs a valid witness $y \in \mathcal{H}_w$ satisfying $(Y, y) \in \mathsf{R}$ is negligible.

2.4. Non-interactive zero knowledge proof

The concept of Non-Interactive Zero-knowledge (NIZK) proof was proposed by Manuel et al. [21] first in 1988, and has received extensive attention. NIZK proof refers to the direct verification without interaction between two parties. The definition of the NIZK proof system for hard relation $(Y, y) \in \mathsf{R}$ consists of two polynomial algorithms $(\mathsf{Prove}_{zk}, \mathsf{Verify}_{zk})$, specifically constructed as follows: (i) $\mathsf{Prove}_{zk}(Y, y)$: takes a statement Y and a witness y as inputs and outputs a proof π; (ii) $\mathsf{Verify}_{zk}(Y, \pi)$: takes a statement Y and a proof π as inputs. If the proof π passes the verification, it outputs 1; otherwise, it outputs 0. And the NIZK proof needs to satisfy the following properties: (i) Completeness: For all $(Y, y) \in \mathsf{R}$, the probability of $\mathsf{Verify}_{zk}(Y, \pi) = 0$ is negligible, in which $\pi \leftarrow \mathsf{Prove}_{zk}(Y, y)$; (ii) Soundness: For all $(Y, y) \notin \mathsf{R}$, the probability of $\mathsf{Verify}_{zk}(Y, \pi) = 1$ is negligible, in which $\pi \leftarrow \mathsf{Prove}_{zk}(Y, y)$; (iii) Zero knowledge: There exists a PPT simulator $\mathcal{S}$ that makes distributions $(Y, \pi \leftarrow \mathsf{Prove}_{zk}(Y, y))$ and $(Y, \pi \leftarrow S(Y))$ indistinguishable for all $(Y, y) \in \mathsf{R}$.

2.5. Randomized EdDSA signature

Randomized EdDSA signature (rEdDSA) [14] embeds a random value in EdDSA signature, forming a tuple of PPT algorithms $\mathsf{rEdDSA} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. Here, we denote the public parameters as $EdPP = (E_p, \mathbb{B}, q, B, b, H_1, H_2)$, where E_p is a twisted elliptic curve, and $\mathbb{B}$ is an additive cycle group with generator B and order q. b is the bit length of secret scalars. $H_1 : \{0,1\}^b \to \{0,1\}^{2b}$ and $H_2 : \{0,1\}^* \to Z_n^*$ are secure hash functions. The specific steps are as follows:

1. $\mathsf{KeyGen}(Edpp)$: Randomly selects a b-bits string t as private key, and calculates hash value $H_1(t) = (h_0, h_1, ..., h_{2b-1})$; it uses the first half of hash value $(h_0, h_1, ..., h_b)$ to calculate $sk_0 = 2^n + \sum_{c \le i < n} 2^i \cdot h_i$ (integer c is 2 or 3, and n satisfies $c \le n < b$), and calculates public key $pk = sk_0 \cdot B$; then it defines the second half of the hash value $(h_b, h_{b+1}, ..., h_{2b-1})$ as sk_1, and outputs a secret and public key pair (sk_0, sk_1, pk).
2. $\mathsf{Sign}(sk_0, sk_1, m)$: To sign a message m, this algorithm randomly selects a value $k \in Z_n^*$, and calculates a random nonce $r = H_2(sk_1||m||k) \bmod q$ and $R = r \cdot B$; it computes the signature $sig = r + H_2(R||pk||m)sk_0 \bmod q$, finally it outputs $\sigma = (sig, R)$.
3. $\mathsf{Verify}(pk, m, \sigma)$: outputs $b = 1$ if the equation $s \cdot B = R + H_2(R||pk||m) \cdot pk$ holds, otherwise, it outputs $b = 0$.

The randomized EdDSA signature scheme was proven unforgeable under chosen message attack [14]. According to forking lemma [22], if an adversary $\mathcal{A}$ forges a valid signature, then another pair of valid signature tuples can be generated by replaying. Using these two sets of forged signature tuples, we can easily recover the secret key sk_0. Therefore, the ability to forge a valid rEdDSA signature would indeed suggest that an adversary can solve an instance of the discrete logarithm problem. This goes against the core discrete logarithm assumption, which is a fundamental basis for many cryptographic protocols and systems.

3. Adaptor signature and security model

We review the definition of adaptor signature and its security model in this section [13].

3.1. Adaptor signature

Adaptor signature needs two parties: the signer and the publisher. The signer generates the pre-signature using the public key and secret key. The publisher, possessing a pair of hard relation, embeds the secret into the pre-signature to generate the full signature. An adaptor signature consists of the signature algorithm $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ and a hard relation $(Y, y) \in \mathsf{R}$, defined as $\Phi_{\mathsf{R,SIG}} = (\mathsf{pSign}, \mathsf{pVerify}, \mathsf{Adapt}, \mathsf{Ext})$. The algorithm is described as follows:

- $\mathsf{pSign}(sk, m, Y)$: takes a secret key sk, message $m \in \mathcal{M}$ and a statement $Y \in L_{\mathsf{R}}$ as inputs, it outputs an incomplete signature named pre-signature $\widetilde{\sigma}$.
- $\mathsf{pVerify}(pk, m, Y, \widetilde{\sigma})$: takes a public key pk, message $m \in \mathcal{M}$, a statement $Y \in L_{\mathsf{R}}$ and the pre-signature $\widetilde{\sigma}$ as inputs, it outputs a bit $b = 1$ if the pre-signature $\widetilde{\sigma}$ passes the verification. Otherwise, it outputs $b = 0$.
- $\mathsf{Adapt}(\widetilde{\sigma}, y)$: takes pre-signature $\widetilde{\sigma}$ and witness y as inputs, it outputs a complete signature σ.
- $\mathsf{Ext}(\widetilde{\sigma}, \sigma, Y)$: takes pre-signature $\widetilde{\sigma}$, complete signature σ and statement $Y \in L_{\mathsf{R}}$ as inputs, it outputs a witness y that satisfies $(Y, y) \in \mathsf{R}$, or Invalid.

Adaptor signature must satisfy correctness of a digital signature, formally defined as pre-signature correctness. It states that a correctly generated pre-signature $\widetilde{\sigma}$, signature σ and y can pass the verification. The adaptor signature needs to simultaneously satisfy the following properties: (i) Pre-signature adaptability: a signature σ generated by the pre-signature $\widetilde{\sigma}$ can only be validated after pre-signature $\widetilde{\sigma}$ passes the verification; (ii) aEUF–CMA security: a PPT adversary cannot forge a valid signature even with access to the pre-signing and signing oracles; (iii) Witness extractability: a PPT adversary who makes a statement Y^* and gives access to the pre-signing and signing oracles, cannot forge a valid signature.

3.2. Security model of adaptor signature

Here, our main goal is to introduce the correctness definition and the security properties of the adaptor signature.

$\mathbf{aSigForge}_{\mathcal{A},\Phi_{\mathrm{R,SIG}}}(n)$	$O_{\mathrm{Sign}}(m)$
$1: \mathcal{M} := \emptyset, pp \leftarrow \mathbf{Setup}(1^n)$	$1: \sigma \leftarrow \mathbf{Sign}(sk, m)$
$2: (sk, pk) \leftarrow \mathbf{KeyGen}(pp),$	$2: \mathcal{M} := \mathcal{M} \cup \{m\}$
$3: m^* \leftarrow \mathcal{A}^{O_{\mathrm{Sign}}, O_{\mathrm{pSign}}}(pk)$	$3:$ **return** σ
$4: (Y, y) \leftarrow \mathbf{GenR}(pp),$	$O_{\mathrm{pSign}}(m, Y)$
$5: \widetilde{\sigma} \leftarrow \mathbf{pSign}(sk, m^*, Y)$	$1: \widetilde{\sigma} \leftarrow \mathbf{pSign}(sk, m, Y)$
$6: \sigma^* \leftarrow \mathcal{A}^{O_{\mathrm{Sign}}, O_{\mathrm{pSign}}}(\widetilde{\sigma}, Y)$	$2: \mathcal{M} := \mathcal{M} \cup \{m\}$
$7: b_1 := m^* \notin \mathcal{M}$	$3:$ **return** $\widetilde{\sigma}$
$8: b_2 := \mathbf{Verify}(pk, m^*, \sigma^*)$	
$9:$ **return** $(b_1 \wedge b_2)$	

Fig. 1. The experiment of $\mathsf{aSigForge}^{\mathcal{A}}_{\Phi_{\mathrm{R,SIG}}}$.

In achieving the correctness of the digital signature, a secure adaptor signature must also ensure the correctness of the pre-signature and the hard relation $(Y, y) \in \mathsf{R}$ of extracted y, which is formally defined as follows:

Definition 2 *(Pre-signature correctness)*. An adaptor signature $\Phi_{\mathrm{R,SIG}}$ satisfies pre-signature correctness, if for any $m \in \mathcal{M}$, security parameter 1^n:

$$\Pr\left[\begin{array}{c|l} \mathbf{pVerify}(pk, m, Y, \widetilde{\sigma}) = 1 & pp \leftarrow \mathbf{Setup}(1^n) \\ \mathbf{Verify}(pk, m, \sigma) = 1 \wedge & (sk, pk) \leftarrow \mathbf{KeyGen}(pp) \\ \left(Y, y'\right) \in \mathrm{R} & (Y, y) \leftarrow \mathbf{GenR}(pp) \\ & \widetilde{\sigma} \leftarrow \mathbf{pSign}(sk, m, Y) \\ & \sigma := \mathbf{Adapt}(\widetilde{\sigma}, y) \\ & y' := \mathbf{Ext}(\sigma, \widetilde{\sigma}, Y) \end{array}\right] = 1$$

A secure adaptor signature needs to satisfy three fundamental security properties. One of these properties is termed as the pre-signature adaptability, which entails that upon generating a valid pre-signature $\widetilde{\sigma}$, the complete signature σ derived from $\widetilde{\sigma}$ remains valid. The formal definition is as follows:

Definition 3 *(Pre-signature adaptability)*. If for any public key pk, $m \in \mathcal{M}$, a statement and witness pair $(Y, y) \in \mathsf{R}$, and a pre-signature $\widetilde{\sigma}$, we have $\mathsf{pVerify}(pk, m, Y, \widetilde{\sigma}) = 1$, then $\mathsf{Verify}(pk, m, Adapt(\widetilde{\sigma}, y)) = 1$. That an adaptor signature $\Phi_{\mathrm{R,SIG}}$ satisfies pre-signature adaptability.

Adaptor signatures need to satisfy existential unforgeability under chosen message attack, which is similar to the unforgeability of digital signature. It is a guarantee for the signer. It means that even if adversary can access both the signing and pre-signing oracles, it cannot forge a valid signature for challenge message m^*. The formal definition is as follows:

Definition 4 *(aEUF–CMA security)*. An adaptor signature $\Phi_{\mathrm{R,SIG}}$ is aEUF-CMA security if the probability of any PPT adversary who wins in the experiment $\mathsf{aSigForge}^{\mathcal{A}}_{\Phi_{\mathrm{R,SIG}}}$ is negligible, that is $\Pr[\mathsf{aSigForge}^{\mathcal{A}}_{\Phi_{\mathrm{R,SIG}}}(1^n) = 1] \leq u(n)$. The experiment is described in Fig. 1.

Adaptor signature needs to satisfy the witness extractability, which is a guarantee for the publisher. It means that the signature forged by adversary for challenge message m^* under ability of forging statement Y^* and accessing the signing and pre-signing oracles cannot extract the valid witness y. The formal definition is as follows:

Definition 5 *(Witness extractability)*. An adaptor signature $\Phi_{\mathrm{R,SIG}}$ is witness extractability if the probability of any PPT adversary who wins in the experiment $\mathsf{aWitExt}^{\mathcal{A}}_{\Phi_{\mathrm{R,SIG}}}$ is negligible, that is $\Pr[\mathsf{aWitExt}^{\mathcal{A}}_{\Phi_{\mathrm{R,SIG}}}(1^n) = 1] \leq u(n)$. The experiment is described in Fig. 2.

$\mathbf{aWitExt}_{\mathcal{A},\Phi_{\mathrm{R,SIG}}}(n)$	$O_{\mathrm{Sign}}(m)$
$1: \mathcal{M} := \emptyset, Edpp \leftarrow \mathbf{Setup}(1^n)$	$1: \sigma \leftarrow \mathbf{Sign}(sk, m)$
$2: (sk, pk) \leftarrow \mathbf{KeyGen}(pp),$	$2: \mathcal{M} := \mathcal{M} \cup \{m\}$
$3: (m^*, Y^*) \leftarrow \mathcal{A}^{O_{\mathrm{Sign}}, O_{\mathrm{pSign}}}(pk)$	$3:$ **return** σ
$4: \widetilde{\sigma} \leftarrow \mathbf{pSign}(sk, m^*, Y^*)$	$O_{\mathrm{pSign}}(m, Y)$
$5: \sigma^* \leftarrow \mathcal{A}^{O_{\mathrm{Sign}}, O_{\mathrm{pSign}}}(\widetilde{\sigma})$	$1: \widetilde{\sigma} \leftarrow \mathbf{pSign}(sk, m, Y)$
$6: y := \mathbf{Ext}(\widetilde{\sigma}, \sigma^*, Y^*)$	$2: \mathcal{M} := \mathcal{M} \cup \{m\}$
$7: b_1 := m^* \notin \mathcal{M}$	$3:$ **return** $\widetilde{\sigma}$
$8: b_2 := (Y^*, y) \notin R$	
$9: b_3 := \mathbf{Verify}(pk, m^*, \sigma^*)$	
$10:$ **return** $(b_1 \wedge b_2 \wedge b_3)$	

Fig. 2. The experiment of $\mathsf{aWitExt}^{\mathcal{A}}_{\Phi_{\mathrm{R,SIG}}}$.

4. Randomized EdDSA-based adaptor signature

We present a novel adaptor signature based on rEdDSA signature, and provide a specific construction. We prove that the proposed scheme satisfies the necessary security properties of a secure adaptor signature.

4.1. Randomized EdDSA-based adaptor signature scheme

We construct a randomized EdDSA-based adaptor signature scheme, utilizing a universal rEdDSA digital signature and a hard relation R. The involved parties in this scheme are the signer and the publisher. As shown in Fig. 3, it consists of a tuples of algorithms $\Phi_{\mathrm{R,rEdDSA}} = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{GenR}, \mathsf{pSign}, \mathsf{pVerify}, \mathsf{Adapt}, \mathsf{Ext}\}$.

1. $Edpp \leftarrow \mathsf{Setup}(1^n)$
   Given a security parameter 1^n, it samples a twisted elliptic curve E_p and an additive cycle group $\mathbb{B}$ with generator B and order q. The system chooses secret rEdDSA scalar with b-bit length, secure hash functions $H_1 : \{0,1\}^b \rightarrow \{0,1\}^{2b}$ and $H_2 : \{0,1\}^* \rightarrow Z_n^*$, then it outputs the public parameter $EdPP = (E_p, \mathbb{B}, q, B, b, H_1, H_2)$.
2. $(sk_0, sk_1, pk) \leftarrow \mathsf{KeyGen}(Edpp)$
   The signer randomly selects a b-bits string t as private key and calculates hash value $H_1(t) = (h_0, h_1, ..., h_{2b-1})$. It utilizes $(h_0, h_1, ..., h_b)$ to calculate $sk_0 = 2^n + \sum_{c \leq i < n} 2^i \cdot h_i$, then calculates $pk = sk_0 \cdot B$, and defines the second half of the hash value $(h_b, h_{b+1}, ..., h_{2b-1})$ as sk_1. Finally the algorithm outputs its secret and public key pair (sk_0, sk_1, pk).
3. $((Y, y), \pi) \leftarrow \mathsf{GenR}(Edpp)$
   The publisher selects a witness $y \in Z_n^*$ and calculates $Y = y \cdot B$, sets hard relation $L_{\mathrm{rEd,R}} := \{Y, y : Y = y \cdot B\}$ and generates a NIZK proof $\pi = \mathsf{Prove}_{zk}(Y, y)$, it outputs a pair and a NIZK proof $((Y, y), \pi)$.
4. $\widetilde{\sigma} \leftarrow \mathsf{pSign}(sk_0, sk_1, m, Y, \pi)$
   To sign message m, the signer with (sk_0, sk_1) randomly selects a value $k \in Z_n^*$, then calculates the random nonce $r = H_2(sk_1 || m || k)$ mod q and $R_{pre} = r \cdot B$, then calculates $R_{sign} = R_{pre} + Y$ and $h = H_2(R_{sign} || pk || m)$. The signer computes $\widetilde{sig} = r + h \cdot sk_0$ mod q and finally outputs $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$.
5. $b \leftarrow \mathsf{pVerify}(pk, m, Y, \widetilde{\sigma})$
   With receiving a pre-signature $\widetilde{\sigma}$, the publisher parses $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$, then calculates $R' = R_{sign} - Y$, $h = H_2(R_{sign} || pk || m)$ and $b_{zk} = \mathsf{Verify}_{zk}(Y, \pi)$. If the equation $\widetilde{sig} \cdot B = R' + h \cdot pk \wedge b_{zk}$ holds, the algorithm outputs $b = 1$, otherwise it outputs $b = 0$.
6. $\sigma \leftarrow \mathsf{Adapt}(\widetilde{\sigma}, y)$

Setup (1^n)	**KeyGen** $(Edpp)$	**GenR** $(Edpp)$
1 : **return** $EdPP =$	1 : choose $t \in \mathcal{Z}_n^*$	1 : choose $y \in \mathcal{Z}_n^*$
2 : $(E_p, \mathbb{B}, q, B, b, H_1, H_2)$	2 : $H_1(t) = (h_0, h_1, ..., h_{2b-1})$	2 : $Y = y \cdot B$
	3 : $sk_0 = 2^n + \sum_{c \le i < n} 2^i \cdot h_i$	3 : $\pi = \mathbf{Prove}_{\mathbf{zk}}(Y, y)$
	4 : $sk_1 = (h_b, h_{b+1}, ..., h_{2b-1})$	4 : **return**$((Y, y), \pi)$
	5 : $pk = sk_0 \cdot B$	
	6 : **return**(sk_0, sk_1, pk)	
pSign (sk_0, sk_1, m, Y, π)	**pVerify** $(pk, m, Y, \widetilde{\sigma})$	**Adapt** $(\widetilde{\sigma}, y)$
1 : select $k \in \mathcal{Z}_n^*$	1 : parse $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$	1 : parse $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$
2 : $r = H_2(sk_1 \| m \| k)$	2 : $R' = R_{sign} - Y$	2 : $sig = \widetilde{sig} + y$
3 : $R_{pre} = r \cdot B$	3 : $h = H_2(R_{sign} \| pk \| m)$	3 : **return** $\sigma = (R_{sign}, sig)$
4 : $R_{sign} = R_{pre} + Y$	4 : $b_{zk} = \mathbf{Verify}_{\mathbf{zk}}(Y, \pi)$	**Ext** $(\widetilde{\sigma}, \sigma, Y)$
5 : $h = H_2(R_{sign} \| pk \| m)$	5 : **if** $(\widetilde{sig} \cdot B = R' + h \cdot pk) \wedge b_{zk}$	1 : parse $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$
6 : $\widetilde{sig} = r + h \cdot sk_0 \ mod \ q$	6 : $b = 1$	2 : parse $\sigma = (R_{sign}, sig)$
7 : **return** $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$	7 : **return**$(b_{zk} \wedge b)$	3 : $y = sig - \widetilde{sig}$
		4 : **if** $(Y, y) \in L_{rEd,R}$
		5 : **return** y

Fig. 3. The design of a randomized EdDSA-based adaptor signature scheme.

In order to convert a pre-signature $\widetilde{\sigma}$ to a complete signature σ, the publisher parses $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$ and calculates $sig = \widetilde{sig} + y$, then outputs complete signature $\sigma = (R_{sign}, sig)$.

7. $y \leftarrow \mathsf{Ext}(\widetilde{\sigma}, \sigma, Y)$
With receiving a pre-signature $\widetilde{\sigma}$ and a complete signature σ, the signer parses $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$ and $\sigma = (R_{sign}, sig)$, computes $y = sig - \widetilde{sig}$ and checks whether $(Y, y) \in L_{\mathsf{rEd,R}}$. If $(Y, y) \in L_{\mathsf{rEd,R}}$, it outputs y, otherwise it outputs Invalid.

4.2. Security proof

We provide detailed security proofs for each property in this section. It demonstrates that the proposed randomized EdDSA-based adaptor signature scheme satisfies pre-signature adaptability, pre-signature correctness, aEUF–rEdDSA-CMA security and witness extractability.

Theorem 1. *Assume that rEdDSA is an EUF-CMA security signature scheme, and R is a hard relation about $(Y, y) \in L_R$. The proposed randomized EdDSA-based adaptor signature scheme $\Phi_{R,rEdDSA}$ is a secure adaptor signature scheme in the random oracle model.*

Lemma 1 *(Pre-signature adaptability). For Definition 3, $\Phi_{R,rEdDSA}$ satisfies pre-signature adaptability under the assumptions of Theorem 1.*

Proof. For any message $m \in \mathcal{M}$, a valid statement and witness pair $(Y, y) \in \mathsf{R}$, a secret and public key pair (sk_0, sk_1, pk), and a pre-signature $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$, where $\widetilde{sig} = r + h \cdot sk_0$ and $h = H_2(R_{sign} \| pk \| m)$, assume $\mathsf{pVerify}(pk, m, Y, \widetilde{\sigma}) = 1$, then we have

$$\begin{aligned}(\widetilde{sig} + y) \cdot B &= (r + h \cdot sk_0 + y) \cdot B \\ &= r \cdot B + h \cdot pk + y \cdot B \\ &= R_{sign} + h \cdot pk \\ &= sig \cdot B\end{aligned}$$

Thus, the complete signature obtained from the pre-signature is a valid signature for the public key pk and message m. Consequently, once pre-signature successfully passes verification, the complete signature is valid. Proof completed.

Lemma 2 *(Pre-signature correctness). For Definition 2, $\Phi_{R,rEdDSA}$ satisfies pre-signature correctness under the assumptions of Theorem 1.*

Proof. For any $m \in \mathcal{M}$, a valid $(Y, y) \in \mathsf{R}$, a key pair (sk_0, sk_1, pk), a pre-signature $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$ from pre-signing algorithm, where $\widetilde{sig} = r + h \cdot sk_0$ and $h = H_2(R_{sign} \| pk \| m)$, then we have

$$\begin{aligned}\widetilde{sig} \cdot B &= (r + h \cdot sk_0) \cdot B \\ &= r \cdot B + h \cdot pk \\ &= R_{sign} - Y + h \cdot pk \\ &= R' + h \cdot pk\end{aligned}$$

Based on the validity of the above equation and the correctness of NIZK, we can show that $\mathsf{pVerify}(pk, m, Y, \widetilde{\sigma}) = 1$. And from Lemma 1, we have $\mathsf{Verify}(m, pk, \sigma) = 1$, where $\sigma = (R_{sign}, sig) = (R_{sign}, \widetilde{sig} + y) = \mathsf{Adapt}(\widetilde{\sigma}, y)$. Finally, we extract witness y from $\mathsf{Ext}(\widetilde{\sigma}, \sigma, Y)$, which satisfies $(Y, y) \in \mathsf{R}$. Proof completed.

Lemma 3 *(aEUF–rEdDSA-CMA security). For Definition 4, $\Phi_{R,rEdDSA}$ satisfies aEUF–rEdDSA-CMA security under the assumptions of Theorem 1.*

Proof. We demonstrate the aEUF–rEdDSA-CMA security of the randomized EdDSA-based adaptor signature scheme $\Phi_{\mathsf{R,rEdDSA}}$. This proof essentially involves a reduction to the EUF-CMA security of the underlying rEdDSA signature. Specifically, we assume that adversary $\mathcal{A}$ conducts an unforgeability experiment $\mathsf{aSigForge}^{\mathcal{A}}_{\Phi_{\mathsf{R,rEdDSA}}}$ with simulator $\mathcal{S}$. If adversary $\mathcal{A}$ wins the $\mathsf{aSigForge}^{\mathcal{A}}_{\Phi_{\mathsf{R,rEdDSA}}}$ experiment, then the simulator $\mathcal{S}$ can win $\mathsf{SigForge}$ experiment of rEdDSA signature or break the hardness of relation R.

In experiment, adversary $\mathcal{A}$ can conduct hash queries $\mathcal{H}$, signing oracle queries O_{Sign} and pre-signing oracle queries O_{pSign}, while the simulator $\mathcal{S}$ can access the rEdDSA signature signing oracle $O_{\mathrm{Sign}}(\cdot)$ and the random oracle $\mathcal{H}(\cdot)$ to respond to these queries. We show the game hops from G_0 to G_5 to explain the process of the reduction step by step, as shown in Fig. 4 and Fig. 5.

G_0: The simulation experiment involves adversary $\mathcal{A}$ successfully forging a valid signature σ^*. $\mathcal{A}$ has knowledge of the valid pre-signature $\widetilde{\sigma}$ of message m^*, and can have access to random oracle $\mathcal{H}$, signing oracle $O_{\mathrm{Sign}}(\cdot)$, and pre-signing oracle $O_{\mathrm{pSign}}(\cdot)$. Game G_0 is initial state of the experiment, hence $\Pr[\mathsf{aSigForge}^{\mathcal{A}}_{\Phi_{\mathsf{R,rEdDSA}}} = 1] = \Pr[G_0 = 1]$.

G_1: This step is similar to G_0, with only one additional check added. Following adversary $\mathcal{A}$ successfully forges the signature σ^*, witness y

Main in G_1	$O_{\text{pSign}}(m, Y, \pi)$ in G_2	$O_{\text{pSign}}(m, Y, \pi)$ in G_3
$1: \mathcal{M} := \emptyset, H := [\perp]$	$1: H' := H$	$1: H' := H$
$2: Edpp \leftarrow \textbf{Setup}(1^n)$	$2: \widetilde{\sigma} := (\widetilde{sig}, R_{sign}, \pi)$	$2: \sigma := (R_{sign}, sig) \leftarrow \textbf{Sign}(sk_0, sk_1, m)$
$3: (sk_0, sk_1, pk) \leftarrow \textbf{KeyGen}(Edpp)$	$3: \leftarrow \textbf{pSign}(sk_0, sk_1, m, Y, \pi)$	$3: \widetilde{sig} := sig - y$
$4: m^* \leftarrow \mathcal{A}^{O_{\text{Sign}}, O_{\text{pSign}}}(pk)$	$4: R_{pre} \leftarrow R_{sign} - Y$	$4: \widetilde{\sigma} := (\widetilde{sig}, R_{sign}, \pi)$
$5: ((Y, y), \pi) \leftarrow \textbf{GenR}(Edpp),$	$5: \textbf{if}\ (H'[R_{pre}\|\|m] \neq \perp$	$5: R_{pre} \leftarrow R_{sign} - Y$
$6: \widetilde{\sigma} := (\widetilde{sig}, R_{sign}, \pi)$	$6: \vee H'[R_{sign}\|\|m] \neq \perp)$	$6: \textbf{if}\ (H'[R_{pre}\|\|m] \neq \perp$
$7: \leftarrow \textbf{pSign}\left(sk_0, sk_1, m^*, Y, \pi\right)$	$7:$ Fail	$7: \vee H'[R_{sign}\|\|m] \neq \perp)$
$8: \sigma^* := (R^*_{sign}, sig^*) \leftarrow \mathcal{A}^{O_{\text{Sign}}, O_{\text{pSign}}}(\widetilde{\sigma}, Y)$	$8: \mathcal{M} := \mathcal{M} \cup \{m\}$	$8:$ Fail
$9: \textbf{if}\ (Y, sig^* - \widetilde{sig}) \in R$	$9: \textbf{return}\ \widetilde{\sigma}$	$9: x := R_{sign}\|\|m$
$10:$ Fail		$10: H[R_{pre}\|\|m] := H[x]$
$11: b_1 := m^* \notin \mathcal{M}$		$11: H[x] \xleftarrow{R} ChSet$
$12: b_2 := \textbf{Verify}(pk, m^*, \sigma^*)$		$12: \mathcal{M} := \mathcal{M} \cup \{m\}$
$13: \textbf{return}\ (b_1 \wedge b_2)$		$13: \textbf{return}\ \widetilde{\sigma}$

Fig. 4. Changes made in the game hops G_1 to G_3.

Main in G_4	Main in G_5
$1: \mathcal{M} := \emptyset, H := [\perp]$	$1: \mathcal{M} := \emptyset, H := [\perp]$
$2: Edpp \leftarrow \textbf{Setup}(1^n)$	$2: Edpp \leftarrow \textbf{Setup}(1^n)$
$3: (sk_0, sk_1, pk) \leftarrow \textbf{KeyGen}(Edpp)$	$3: (sk_0, sk_1, pk) \leftarrow \textbf{KeyGen}(Edpp)$
$4: m^* \leftarrow \mathcal{A}^{O_{\text{Sign}}, O_{\text{pSign}}}(pk)$	$4: m^* \leftarrow \mathcal{A}^{O_{\text{Sign}}, O_{\text{pSign}}}(pk)$
$5: ((Y, y), \pi) \leftarrow \textbf{GenR}(Edpp),$	$5: ((Y, y), \pi) \leftarrow \textbf{GenR}(Edpp)$
$6: H' := H$	$6: H' := H$
$7: \widetilde{\sigma} := (\widetilde{sig}, R_{sign}, \pi)$	$7: \sigma := (R_{sign}, sig) \leftarrow \textbf{Sign}\left(sk_0, sk_1, m^*\right)$
$8: \leftarrow \textbf{pSign}\left(sk_0, sk_1, m^*, Y, \pi\right)$	$8: \widetilde{sig} := sig - y$
$9: R_{pre} \leftarrow R_{sign} - Y$	$9: \widetilde{\sigma} := (\widetilde{sig}, R_{sign}, \pi)$
$10: \textbf{if}\ (H'[R_{pre}\|\|m^*] \neq \perp$	$10: R_{pre} \leftarrow R_{sign} - Y$
$11: \vee H'[R_{sign}\|\|m^*] \neq \perp)$	$11: \textbf{if}\ (H'[R_{pre}\|\|m^*] \neq \perp$
$12:$ Fail	$12: \vee H'[R_{sign}\|\|m^*] \neq \perp)$
$13: \sigma^* := (R^*_{sign}, sig^*) \leftarrow \mathcal{A}^{O_{\text{Sign}}, O_{\text{pSign}}}(\widetilde{\sigma}, Y)$	$13:$ Fail
$14: \textbf{if}\ (Y, sig^* - \widetilde{sig}) \in R$	$14: x := R_{sign}\|\|m^*$
$15:$ Fail	$15: H[R_{pre}\|\|m^*] = H[x]$
$16: b_1 := m^* \notin \mathcal{M}$	$16: H[x] \xleftarrow{R} ChSet$
$17: b_2 := \textbf{Verify}(pk, m^*, \sigma^*)$	$17: \sigma^* := (R^*_{sign}, sig^*) \leftarrow \mathcal{A}^{O_{\text{Sign}}, O_{\text{pSign}}}(\widetilde{\sigma}, Y)$
$18: \textbf{return}\ (b_1 \wedge b_2)$	$18: \textbf{if}\ (Y, sig^* - \widetilde{sig}) \in R$
	$19:$ Fail
	$20: b_1 := m^* \notin \mathcal{M}$
	$21: b_2 := \textbf{Verify}(pk, m^*; \sigma^*)$
	$22: \textbf{return}\ (b_1 \wedge b_2)$

Fig. 5. Changes made in the game hops G_4 and G_5.

can be extracted from forged signature σ^* and pre-signature $\widetilde{\sigma}$, computed as $y = sig^* - \widetilde{sig}$. If $(Y, y) \in \mathsf{R}$, then the game fails.

Claim. *The failure probability of game G_1 is negligible, it holds that $\Pr[fail_1] \leq u_1(n)$, where $fail_1$ represents the probability of G_1 event aborts and $u_1(n)$ represents the negligible function in n.*

Proof. Considering a reduction to the hardness of relation R. If the complete signature forged by adversary $\mathcal{A}$ can extract witness y that satisfies the hard relation, such that $(Y, y) \in \mathsf{R}$, then the simulator can break the hardness of relation R. Due to the inherent characteristics, the probability of breaking the hardness of relation is negligible. Then we have $\Pr[\mathsf{fail}_1] \leq u_1(n)$. Moreover, compared to G_1 and G_0, only the step of checking the extracted witness y is added, it holds that $\Pr[G_0 = 1] \leq \Pr[G_1 = 1] + u_1(n)$.

G_2: This step closely resembles G_1, with the only change made during the pre-signing oracle queries O_{pSign}. It is checked whether the random values R_{pre} and R_{sign} have been stored in the list H during pre-signing oracle queries O_{pSign}. If the random values are queried, the game fails.

Claim. *The failure probability of game G_2 is negligible, it holds that $\Pr[fail_2] \leq u_2(n)$, where $fail_2$ represents the probability of G_2 event aborts and $u_2(n)$ represents the negligible function in n.*

Proof. Random values are uniformly and randomly selected from the defined set whose size is p. As a PPT algorithm, $\mathcal{A}$ only executes a polynomial number of queries to the $\mathcal{H}$, O_{Sign} and O_{pSign}. Total number of queries is denoted as n. The probability of being queried for random value R_{pre} in the set is $\Pr[\mathcal{H}'[R_{pre}\|\|m] \neq \perp] \leq \frac{n}{p}$. Similarly, for random value R_{sign}, the probability of being queried also is $\Pr[\mathcal{H}'[R_{sign}\|\|m] \neq \perp] \leq \frac{n}{p}$. In summary, failure probability of game is related to the probabil-

$\mathcal{S}^{\mathrm{SIG}^{\mathrm{rEdDSA}},\mathcal{H}}$	$O_{\mathrm{pSign}}(m,Y,\pi)$	$O_{\mathrm{Sign}}(m)$
$1:\mathcal{M}:=\emptyset, H:=[\perp]$	$1:H':=H$	$1:\sigma\leftarrow\mathbf{Sign}(sk_0,sk_1,m)$
$2:Edpp\leftarrow\mathbf{Setup}(1^n)$	$2:\sigma:=(R_{sign},sig)\leftarrow\mathbf{Sign}(sk_0,sk_1,m)$	$2:x:=R_{sign}\|\|m$
$3:(sk_0,sk_1,pk)\leftarrow\mathbf{KeyGen}(Edpp)$	$3:\widetilde{sig}:=sig-y$	$3:H(x):=\mathcal{H}(x)$
$4:m^*\leftarrow\mathcal{A}^{O_{\mathrm{Sign}},O_{\mathrm{pSign}}}(pk)$	$4:\widetilde{\sigma}:=(\widetilde{sig},R_{sign},\pi)$	$4:\mathcal{M}:=\mathcal{M}\cup\{m\}$
$5:((Y,y),\pi)\leftarrow\mathbf{GenR}(Edpp),$	$5:R_{pre}\leftarrow R_{sign}-Y$	$5:\mathbf{return}\,\sigma$
$6:H':=H$	$6:\mathbf{if}\,(H'[R_{pre}\|\|m]\neq\perp$	
$7:\sigma:=(R_{sign},sig)\leftarrow\mathbf{Sign}\left(sk_0,sk_1,m^*\right)$	$7:\vee H'[R_{sign}\|\|m]\neq\perp)$	
$8:\widetilde{sig}:=sig-y$	$8:$ Fail	
$9:\widetilde{\sigma}:=(\widetilde{sig},R_{sign},\pi)$	$9:x:=R_{sign}\|\|m$	
$10:R_{pre}\leftarrow R_{sign}-Y$	$10:H[R_{pre}\|\|m]=\mathcal{H}[x]$	
$11:\mathbf{if}\,(H'[R_{pre}\|\|m^*]\neq\perp$	$11:H[x]\xleftarrow{R}\mathcal{H}(R_{pre}\|\|m)$	
$12:\vee H'[R_{sign}\|\|m^*]\neq\perp)$	$12:\mathcal{M}:=\mathcal{M}\cup\{m\}$	
$13:$ Fail	$13:\mathbf{return}\,\widetilde{\sigma}$	
$14:x:=R_{sign}\|\|m^*$	$\mathcal{H}(x)$	
$15:H[R_{pre}\|\|m^*]=\mathcal{H}[x]$	$1:\mathbf{if}\;H(x)=\perp$	
$16:H[x]\xleftarrow{R}\mathcal{H}(R_{pre}\|\|m^*)$	$2:\;H(x):=\mathcal{H}(x)$	
$17:\sigma^*:=(R^*_{sign},sig^*)\leftarrow\mathcal{A}^{O_{\mathrm{Sign}},O_{\mathrm{pSign}}}(\widetilde{\sigma},Y)$	$3:\mathbf{return}\;H(x)$	
$18:\mathbf{if}\,(Y,sig^*-\widetilde{sig})\in R$		
$19:$ Fail		
$20:\mathbf{return}\,(m^*,\sigma^*)$		

Fig. 6. The final simulation of experiment $\mathsf{aSigForge}^{\mathcal{A}}_{\Phi_{\mathsf{R,rEdDSA}}}$.

ity of these two random values being queried, then we have $\Pr[\mathsf{fail}_2]=\Pr[\mathcal{H}'[R_{pre}||m]\neq\perp]+[\mathcal{H}'[R_{sign}||m]\neq\perp]\leq\frac{2n}{p}\leq u_2(n)$.

From G_1 to G_2, only one step of checking for random values has been added during the pre-signing queries stage, thus we have $\Pr[G_1=1]\leq\Pr[G_2=1]+u_2(n)$.

G_3: This step is crucial part of game hops, as it modifies the pre-signing queries. The simulator replaces a pre-signature by a complete signature of underlying rEdDSA signature scheme and random values produced by random oracle $\mathcal{H}$. It sets random value R_{pre} to value R_{sign}, and then a fresh value R_{sign} is selected uniformly at random.

Due to the negligible probability that the random values related to the pre-signature have been queried in G_2, adversary $\mathcal{A}$ cannot determine whether the signature returned in the pre-signing oracle queries is a pre-signature or a complete signature. Thus we have $\Pr[G_2=1]=\Pr[G_3=1]$.

G_4: Once adversary $\mathcal{A}$ has pre-signature generated from the challenge message m^*, it is checked in the list H whether the random values R_{pre} and R_{sign} have been stored. If these random values are stored, the game fails.

Claim. *The failure probability of game G_4 is negligible, it holds that $\Pr[\mathsf{fail}_3]\leq u_3(n)$, where fail_3 represents the probability of G_4 event aborts and $u_3(n)$ represents the negligible function in n.*

Proof. This checking process is consistent with G_2. The probability of random values R_{pre} and R_{sign} already stored is $\Pr[\mathsf{fail}_3]=\Pr[\mathcal{H}'[R_{pre}||m^*]\neq\perp]+[\mathcal{H}'[R_{sign}||m^*]\neq\perp]\leq\frac{n}{p}+\frac{n}{p}\leq\frac{2n}{p}\leq u_3(n)$. Thus we have $\Pr[G_3=1]\leq\Pr[G_4=1]+u_3(n)$.

G_5: This is the final step of the experiment, similar to G_3. When adversary $\mathcal{A}$ is querying about the pre-signature for challenge message m^*, simulator replaces the pre-signature by the complete signature of the underlying rEdDSA signature scheme and random values produced by random oracle $\mathcal{H}$.

This proof process is consistent with G_3. Since the probability of random values being queried is negligible, adversary $\mathcal{A}$ cannot distinguish whether the generated signature is a pre-signature or a complete signature. Thus we have $\Pr[G_4=1]=\Pr[G_5=1]$.

From the above proof, it is clear that the transition from games G_0 to G_5 is indistinguishable. Next, we need to show that the signature forged by adversary $\mathcal{A}$ for challenge message m^* is valid in SigForge experiment. Final simulation is depicted in Fig. 6.

Claim. *(m^*,σ^*) is a valid message and signature pair in the game SigForge.*

Proof. To forge valid message and signature pair, simulator $\mathcal{S}$ can access signing oracle and random oracle to simulate the response of pre-signing oracle queries. In fact, it checks whether the message m^* has been queried to the signing oracle both in queries and challenge stage. If the challenge message m^* has been queried in $\mathsf{aSigForge}^{\mathcal{A}}_{\Phi_{\mathsf{R,rEdDSA}}}$ experiment, the game fails. Thus the message and signature pair (m^*,σ^*) successfully forged by adversary $\mathcal{A}$ in the game SigForge is valid.

The game hops of G_0 to G_5 have achieved a reduction from the initial experiment aSigForge to the underlying rEdDSA experiment SigForge. From the above proof, we have $\Pr[G_0=1]\leq\Pr[G_5=1]+u_1(n)+u_2(n)+u_3(n)$, where $\Pr[G_0=1]$ is the probability of the adversary $\mathcal{A}$ wins the unforgeability experiment $\mathsf{aSigForge}^{\mathcal{A}}_{\Phi_{\mathsf{R,rEdDSA}}}$, and $\Pr[G_5=1]$ is probability of adversary $\mathcal{A}$ wins SigForge experiment of the rEdDSA signature. Therefore, we obtain $\Pr[\mathsf{aSigForge}^{\mathcal{A}}_{\Phi_{\mathsf{R,rEdDSA}}}(1^n)=1]\leq\Pr[\mathsf{SigForge}^{\mathcal{S},\mathcal{A}}_{\Phi_{\mathsf{rEdDSA}}}(1^n)=1]+u(n)$, where $u(n)=u_1(n)+u_2(n)+u_3(n)$ is negligible.

The EUF-CMA properties of the underlying rEdDSA and the inherent characteristics of hard relation R ensure that the probability of adversary $\mathcal{A}$ successfully forging a valid signature is negligible. Proof completed.

Lemma 4 *(Witness extractability). For Definition 5, $\Phi_{R,rEdDSA}$ satisfies witness extractability under the assumptions of Theorem 1.*

Proof. To prove the witness extractability for $\Phi_{\mathsf{R,rEdDSA}}$, we provide a reduction to the unforgeability experiment of the underlying rEdDSA signature, but the information obtained by the adversary $\mathcal{A}$ is different. Specifically, we assume that adversary $\mathcal{A}$ conducts an experiment $\mathsf{aWitExt}^{\mathcal{A}}_{\Phi_{\mathsf{R,rEdDSA}}}$ with the simulator $\mathcal{S}$. If adversary $\mathcal{A}$ wins experiment

$O_{\text{pSign}}(m,Y,\pi)$ in $\boldsymbol{G_1}$	Main in $\boldsymbol{G_3}$	Main in $\boldsymbol{G_4}$
$1: H' := H$	$1: \mathcal{M} := \emptyset, H := [\perp]$	$1: \mathcal{M} := \emptyset, H := [\perp]$
$2: \widetilde{\sigma} := (\widetilde{sig}, R_{sign}, \pi)$	$2: Edpp \leftarrow \textbf{Setup}(1^n)$	$2: Edpp \leftarrow \textbf{Setup}(1^n)$
$3: \leftarrow \textbf{pSign}(sk_0, sk_1, m, Y, \pi)$	$3: (sk_0, sk_1, pk) \leftarrow \textbf{KeyGen}(Edpp)$	$3: (sk_0, sk_1, pk) \leftarrow \textbf{KeyGen}(Edpp)$
$4: R_{pre} \leftarrow R_{sign} - Y$	$4: (m^*, Y^*, \pi^*) \leftarrow \mathcal{A}^{O_{\text{Sign}}, O_{\text{pSign}}}(pk)$	$4: (m^*, Y^*, \pi^*) \leftarrow \mathcal{A}^{O_{\text{Sign}}, O_{\text{pSign}}}(pk)$
$5: \textbf{if}\ (H'[R_{pre}\|\|m] \neq \perp$	$5: H' := H$	$5: H' := H$
$6: \vee H'[R_{sign}\|\|m] \neq \perp)$	$6: \widetilde{\sigma} := (\widetilde{sig}, R_{sign}, \pi)$	$6: \sigma := (R_{sign}, sig) \leftarrow \textbf{Sign}\left(sk_0, sk_1, m^*,\right)$
$7:$ Fail	$7: \leftarrow \textbf{pSign}\left(sk_0, sk_1, m^*, Y^*, \pi^*\right)$	$7: \widetilde{sig} := sig - y$
$8: \mathcal{M} := \mathcal{M} \cup \{m\}$	$8: R_{pre} \leftarrow R_{sign} - Y^*$	$8: \widetilde{\sigma} := (\widetilde{sig}, R_{sign}, \pi)$
$9: \textbf{return}\ \widetilde{\sigma}$	$9: \textbf{if}\ (H'[R_{pre}\|\|m^*] \neq \perp$	$9: R_{pre} \leftarrow R_{sign} - Y^*$
$O_{\text{pSign}}(m,Y,\pi)\ in\ \mathbf{G_2}$	$10: \vee H'[R_{sign}\|\|m^*] \neq \perp)$	$10: \textbf{if}\ (H'[R_{pre}\|\|m^*] \neq \perp$
$1: H' := H$	$11:$ Fail	$11: \vee H'[R_{sign}\|\|m^*] \neq \perp)$
$2: \sigma := (R_{sign}, sig) \leftarrow \textbf{Sign}(m, sk)$	$12: \sigma^* \leftarrow \mathcal{A}(\widetilde{\sigma})$	$12:$ Fail
$3: \widetilde{sig} := sig - y$	$13: y := \textbf{Ext}\left(\widetilde{\sigma}, \sigma^*, Y^*\right)$	$13: x := R_{sign}\|\|m^*$
$4: \widetilde{\sigma} := (\widetilde{sig}, R_{sign}, \pi)$	$14: b_1 := m^* \notin \mathcal{M}$	$14: H[R_{pre}\|\|m^*] = H[x]$
$5: R_{pre} \leftarrow R_{sign} - Y$	$15: b_2 := (Y^*, y) \notin R$	$15: H[x] \xleftarrow{R} ChSet$
$6: \textbf{if}\ (H'[R_{pre}\|\|m] \neq \perp$	$16: b_3 := \textbf{Verify}\,(pk, m^*, \sigma^*)$	$16: \sigma^* \leftarrow \mathcal{A}(\widetilde{\sigma})$
$7: \vee H'[R_{sign}\|\|m] \neq \perp)$	$17: \textbf{return}\ (b_1 \wedge b_2 \wedge b_3)$	$17: y := \textbf{Ext}\left(\widetilde{\sigma}, \sigma^*, Y^*\right)$
$8:$ Fail		$18: b_1 := m^* \notin \mathcal{M}$
$9: x := R_{sign}\|\|m$		$19: b_2 := (Y^*, y) \notin R$
$10: H[R_{pre}\|\|m] = H[x]$		$20: b_3 := \textbf{Verify}\,(pk, m^*, \sigma^*)$
$11: H[x] \xleftarrow{R} ChSet$		$21: \textbf{return}\ (b_1 \wedge b_2 \wedge b_3)$
$12: \mathcal{M} := \mathcal{M} \cup \{m\}$		
$13: \textbf{return}\ \widetilde{\sigma}$		

Fig. 7. Changes made in the game hops G_1 to G_4.

$\mathsf{aWitExt}^{\mathcal{A}}_{\Phi_{\mathsf{R,rEdDSA}}}$, then the simulator $\mathcal{S}$ can win the unforgeability experiment $\mathsf{SigForge}$ of the rEdDSA signature. At this point, since $\mathcal{A}$ forges the statement Y^*, the reduction of hard relation is no longer considered.

Adversary $\mathcal{A}$ can conduct the oracle $\mathcal{H}$, O_{Sign} and O_{pSign} in experiment. And the simulator can access the rEdDSA signature signing oracle $O_{\text{Sign}}(\cdot)$ and the random oracle $\mathcal{H}(\cdot)$ to respond to these queries. We show the game hops of G_0 to G_4 to explain the process of the reduction step by step, as shown in Fig. 7.

G_0: The adversary $\mathcal{A}$ generates a statement Y^*. Under the condition of accessing oracle $\mathcal{H}(\cdot)$, $O_{\text{Sign}}(\cdot)$ and $O_{\text{pSign}}(\cdot)$, the adversary successfully forges a valid signature σ^* by acquiring the valid pre-signature for message m^*. At this point, the witness y extracted from pre-signature and forged signature does not satisfy the hard relation R, where $(Y^*, y) \notin \mathsf{R}$. Game G_0 is the initial state of experiment, therefore $\Pr[\mathsf{aWitExt}^{\mathcal{A}}_{\Phi_{\mathsf{R,rEdDSA}}} = 1] = \Pr[G_0 = 1]$.

G_1: This step closely resembles game G_2 of Lemma 3. During the pre-signing oracle queries, the simulator $\mathcal{S}$ checks if the random values R_{pre} and R_{sign} have been stored in the list H. If the random values are queried, the game fails.

Claim. *The failure probability of game G_1 is negligible, it holds that $\Pr[fail_1] \leq u_1(n)$, where $fail_1$ represents the probability of G_1 event aborts and $u_1(n)$ represents the negligible function in n.*

Proof. Random values are uniformly and randomly selected within defined set whose size is p. As a PPT algorithm, $\mathcal{A}$ only execute a polynomial number of queries to the $\mathcal{H}$, O_{Sign}, and O_{pSign}. Total number of queries is denoted n. The probability of being queried for random value R_{pre} is $\Pr[\mathcal{H}'[R_{pre}||m] \neq \perp] \leq \frac{n}{p}$. For random value R_{sign}, the probability of being queried is $\Pr[\mathcal{H}'[R_{sign}||m] \neq \perp] \leq \frac{n}{p}$. Thus we have $\Pr[\mathsf{fail}_1] = \Pr[\mathcal{H}'[R_{pre}||m] \neq \perp] + [\mathcal{H}'[R_{sign}||m] \neq \perp] \leq \frac{2n}{p} \leq u_1(n)$.

From G_0 to G_1, only one step of querying for random values has been added in the pre-signing queries, then we have $\Pr[G_0 = 1] \leq \Pr[G_1 = 1] + u_1(n)$.

G_2: Based on the above random values queries, this step uses the underlying rEdDSA signature to simulate pre-signing oracle queries. Simulator $\mathcal{S}$ replaces pre-signature by complete signature of underlying rEdDSA and the random values simulated by random oracle $\mathcal{H}$. It sets random value R_{pre} to value R_{sign}, and then a fresh value R_{sign} is selected uniformly at random.

Since G_1 indicates that the probability of the random values related to the pre-signature being queried is negligible, adversary $\mathcal{A}$ cannot determine whether the signature returned in the pre-signing oracle is a pre-signature or a complete signature. Thus we have $\Pr[G_1 = 1] = \Pr[G_2 = 1]$.

G_3: This step closely resembles G_1. After the adversary $\mathcal{A}$ has the pre-signature from challenge message m^*, it is checked whether the random values have been stored in the H. If these random values are queried, the game fails.

Claim. *The failure probability of game G_3 is negligible, it holds that $\Pr[fail_2] \leq u_2(n)$, where $fail_2$ represents the probability of G_3 event aborts and u_2 represents the negligible function in n.*

Proof. The specific process is consistent with G_1, and it can be obtained that $\Pr[\mathsf{fail}_2] = \Pr[\mathcal{H}'[R_{pre}||m^*] \neq \perp] + [\mathcal{H}'[R_{sign}||m^*] \neq \perp] \leq \frac{n}{p} + \frac{n}{p} \leq \frac{2n}{p} \leq u_2(n)$. From G_2 to G_3, only one step of checking for random values has been added before the adversary forges the signature, then we have $\Pr[G_2 = 1] \leq \Pr[G_3 = 1] + u_2(n)$.

G_4: This step is the last of the experiment, similar to G_2. $\mathcal{S}$ simulates pre-signature using complete signature by underlying rEdDSA and random values produced by random oracle $\mathcal{H}$ in challenge stage.

This proof process is consistent with G_2, because the probability of random values being queried is negligible, and adversary $\mathcal{A}$ cannot distinguish if generated signature is a pre-signature or a complete signature. Therefore we have $\Pr[G_3 = 1] = \Pr[G_4 = 1]$.

$\mathcal{S}^{\mathrm{SIG}^{\mathrm{rEdDSA}},\mathcal{H}}$	$O_{\mathrm{pSign}}(m,Y,\pi)$	$O_{\mathrm{Sign}}(m)$
$1:\mathcal{M}:=\emptyset, H:=[\perp]$	$1:H':=H$	$1:\sigma\leftarrow\mathbf{Sign}(sk_0,sk_1,m)$
$2:Edpp\leftarrow\mathbf{Setup}(1^n)$	$2:\sigma:=(R_{sign},sig)\leftarrow\mathbf{Sign}(sk_0,sk_1,m)$	$2:x:=R_{sign}\|\|m$
$3:(sk_0,sk_1,pk)\leftarrow\mathbf{KeyGen}(Edpp),$	$3:\widetilde{sig}:=sig-y$	$3:H(x):=\mathcal{H}(x)$
$4:(m^*,Y^*,\pi)\leftarrow\mathcal{A}^{O_{\mathrm{Sign}},O_{\mathrm{pSign}}}(pk)$	$4:\widetilde{\sigma}:=(\widetilde{sig},R_{sign},\pi)$	$4:\mathcal{M}:=\mathcal{M}\cup\{m\}$
$5:H':=H$	$5:R_{pre}\leftarrow R_{sign}-Y$	$5:\mathbf{return}\,\sigma$
$6:\sigma:=(R_{sign},sig)\leftarrow\mathbf{Sign}\left(sk_0,sk_1,m^*\right)$	$6:\mathbf{if}\,(H'[R_{pre}\|\|m]\neq\perp$	
$7:\widetilde{sig}:=sig-y$	$7:\vee H'[R_{sign}\|\|m]\neq\perp)$	
$8:\widetilde{\sigma}:=(\widetilde{sig},R_{sign},\pi)$	$8:$ Fail	
$9:R_{pre}\leftarrow R_{sign}-Y^*$	$9:x:=R_{sign}\|\|m$	
$10:\mathbf{if}\,(H'[R_{pre}\|\|m^*]\neq\perp$	$10:H[R_{pre}\|\|m]=\mathcal{H}[x]$	
$11:\vee H'[R_{sign}\|\|m^*]\neq\perp)$	$11:H[x]\xleftarrow{R}\mathcal{H}(R_{pre}\|\|m)$	
$12:$ Fail	$12:\mathcal{M}:=\mathcal{M}\cup\{m\}$	
$13:x:=R_{sign}\|\|m^*$	$13:\mathbf{return}\,\widetilde{\sigma}$	
$14:H[R_{pre}\|\|m^*]=\mathcal{H}[x]$	$\mathcal{H}(x)$	
$15:H[x]\xleftarrow{R}\mathcal{H}(R_{pre}\|\|m^*)$	$1:\mathbf{if}\,H(x)=\perp$	
$16:\sigma^*\leftarrow\mathcal{A}(\widetilde{\sigma})$	$2:H(x):=\mathcal{H}(x)$	
$17:\mathbf{return}\,(m^*,\sigma^*)$	$3:\mathbf{return}\,H(x)$	

Fig. 8. The final simulation of experiment $\mathsf{aWitExt}^{\mathcal{A}}_{\Phi_{\mathrm{R,rEdDSA}}}$.

From above proof, it is clear that the transition from games G_0 to G_4 is indistinguishable. We need to show that the signature forged by adversary $\mathcal{A}$ for challenge message m^* is valid in SigForge experiment. Final simulation is depicted in Fig. 8.

Claim. *(m^*,σ^*) is a valid message and signature pair in the game SigForge.*

Proof. Before adversary $\mathcal{A}$ forges a signature, it is checked whether the random values in pre-signature have been queried, which actually means checking whether the message m^* has been queried by signing oracle both in queries and challenge stage. If challenge message m^* has been queried in $\mathsf{aWitExt}^{\mathcal{A}}_{\Phi_{\mathrm{R,rEdDSA}}}$ experiment, the game fails. Thus we have that message and signature pair forged by adversary $\mathcal{A}$ in the game SigForge is valid.

The game hops of G_0 to G_4 have given a reduction from the initial experiment aWitExt to the underlying rEdDSA experiment SigForge. From the above proof, we have $\Pr[G_0=1]\leq\Pr[G_4=1]+u_1(n)+u_2(n)$, where $\Pr[G_0=1]$ is the probability of $\mathcal{A}$ wins the witness extractability $\mathsf{aWitExt}^{\mathcal{A}}_{\Phi_{\mathrm{R,rEdDSA}}}$ experiment, and $\Pr[G_5=1]$ is the probability of $\mathcal{A}$ wins the underlying rEdDSA signature SigForge experiment. Therefore, we obtain $\Pr[\mathsf{aWitExt}^{\mathcal{A}}_{\Phi_{\mathrm{R,rEdDSA}}}(1^n)=1]=\Pr[\mathsf{SigForge}^{S^{\mathcal{A}}}_{\Phi_{\mathrm{rEdDSA}}}(1^n)=1]+u(n)$, where $u(n)=u_1(n)+u_2(n)$.

Due to the EUF-CMA of underlying rEdDSA signature, the probability of adversary $\mathcal{A}$ successfully forging a valid signature is negligible. Proof completed.

5. Experimental evaluation

In this section, we conduct an evaluation of computation cost and communication overhead of rEdDSA-based adaptor signature scheme $\Phi_{\mathrm{R,rEdDSA}}$ to evaluate its efficiency. Firstly, we delve into the analysis of time cost for each algorithm in our proposed signature scheme, and compare it with an ECDSA-based signature scheme. Then we analyze the communication overhead of each algorithm and also draw comparisons with ECDSA-based signature scheme. In this experiment, we utilize a twisted Edwards curve Ed25519 as the instantiation object of rEdDSA. The simulation was executed on a desktop equipped with AMD Ryzen 7 3700X CPU @ 3.6 GHz and 16 GB RAM. The programming language employed was Python, and the compiler was JetBrains PyCharm version 2023.2.1.

Table 1
The time consumption of ECDSA-based and rEdDSA-based adaptor signature schemes.

Adaptor signatures	GenR	pSign	pVerify	Adapt	Ext
ECDSA-based	t_p+t_m	$2t_m+t_h+t_i$	$t_v+2t_m+t_h+t_i$	t_i	t_i
rEdDSA-based	t_p+t_m	t_m+2t_h	$t_v+t_m+t_h$	t_{add}	t_{add}

5.1. Computation cost

In terms of computation cost, we mainly evaluate GenR, pSign, pVerify, Adapt and Ext, the five algorithms in the adaptor signature. Here we define t_m as the computational time for point multiplication in group G, t_p and t_v as the computational time for NIZK.prove and NIZK.verify algorithms, t_h as the computational time for running a SHA-512 hash function, t_i and t_{add} as the computational time for modular inverse and modular addition operations in Z_n^*, respectively. Based on the above definitions, we present the time consumption of different algorithms in ECDSA-based and rEdDSA-based adaptor signature scheme, as depicted in Table 1.

In Table 1, the time consumption of GenR algorithm based on rEdDSA and ECDSA adaptor signatures is t_p+t_m, which varies in specific running times of due to differences in the underlying curve. Among the other four algorithms, ECDSA-based algorithm involves inverse operation compared with rEdDSA-based one, the running time is different. The specific running times of these five algorithms are depicted in Fig. 9. The rEdDSA-based adaptor signature exhibits lower time costs than ECDSA-based in both GenR and pSign algorithms, which is attributed to rEdDSA's utilization of a twisted Edwards curve and Schnorr-like structure. In pVerify phase, due to the existence of NIZK proof and pre-signature verification, the time cost of both pVerify algorithm is high. As the underlying curve of rEdDSA, the time cost of rEdDSA-based scheme is also less than that of ECDSA-based one. Since Adapt and Ext algorithms in rEdDSA-based only involve addition operation and in ECDSA-based only involve inverse operation, the time costs of these two adaptor signature algorithms are small. Following the above analysis, in comparison to ECDSA-based, the rEdDSA-based adaptor signature scheme has faster key generation, pre-signing and pre-signing verification algorithms, which improve the efficiency of adaptor signature.

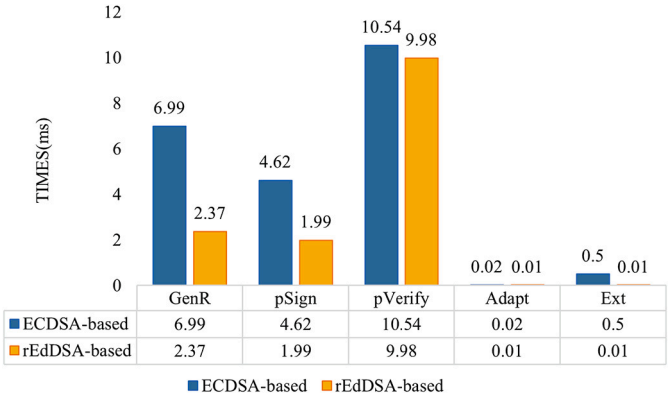

Fig. 9. Execution time of ECDSA-based and rEdDSA-based adaptor signature schemes.

Table 2
The communication overhead of ECDSA-based and rEdDSA-based adaptor signature schemes.

Schemes	Proof sizes	Value	Pre-signature sizes	Value
ECDSA-based	$3\|Z_n^*\| + \|G_1\|$	160 bytes	$4\|Z_n^*\| + \|G_1\|$	192 bytes
rEdDSA-based	$2\|Z_n^*\| + 2\|G_2\|$	128 bytes	$2\|Z_n^*\| + 2\|G_2\|$	128 bytes

5.2. Communication overhead

Among the ECDSA-based and rEdDSA-based adaptor signature schemes, since their public key, secret key and complete signature are not significantly different from the underlying ECDSA and rEdDSA signature scheme, the communication overhead discussed in this section mainly analyzes the size of the proof in GenR phase and pre-signature in pSign phase. In GenR phase, the size of the ECDSA-based proof is $3|Z_n^*| + |G_1|$. In the rEdDSA-based adaptor signature scheme, the form of proof is $((Y, y), \pi)$, where the size of generated NIZK proof π is $|Z_n^*| + |G_2|$, and the size of (Y, y) is also $|Z_n^*| + |G_2|$, so the proof size is $2|Z_n^*| + 2|G_2|$. In pSign phase, the size of the ECDSA-based pre-signature is $4|Z_n^*| + |G_1|$. In the rEdDSA-based adaptor signature scheme, the pre-signature forms as $\widetilde{\sigma} = (\widetilde{sig}, R_{sign}, \pi)$, where the size of NIZK proof π is $|Z_n^*| + |G_2|$, and the size of $\widetilde{sig}$ and R_{sign} is $|Z_n^*|$ and $|G_2|$, respectively. So the total size is $2|Z_n^*| + 2|G_2|$. In our experiments, we adapt Ed25519 as underlying curve and set $|Z_n^*| = 32$ bytes, $|G_1| = 64$ bytes, and $|G_2| = 32$ bytes. As listed in Table 2, the byte size of the proof and pre-signature calculated by the rEdDSA-based scheme is lower than that of the ECDSA-based scheme, resulting in significant savings in storage space.

6. Conclusions

Currently, EdDSA signature has become one of the popular underlying signature algorithms for cryptocurrencies. However, its deterministic signature nature poses an impossibility for a direct conversion into an adaptor signature. This limitation impacts the construction of off-chain payment channels in blockchain systems. We used EdDSA with random value to solve this problem, transforming the EdDSA signature algorithm into a random one, then designed a randomized EdDSA-based adaptor signature scheme. We also analyzed and proved in detail that the proposed adaptor signature scheme satisfies the defined adaptor signature security properties, and demonstrated its effectiveness through experimental evaluation.

CRediT authorship contribution statement

Yixing Zhu: Writing – original draft, Software, Formal analysis, Data curation, Conceptualization. **Huilin Li:** Writing – review & editing, Validation. **Mengze Li:** Writing – review & editing. **Yong Yu:** Writing – review & editing, Supervision, Resources, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yong Yu is a lead Guest Editor for Digital Communications and Networks and was not involved in the editorial review or the decision to publish this article. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported by the National Key R&D Program of China (2022YFB2701500), the National Natural Science Foundation of China (62272385, 62311540156), Shaanxi Distinguished Youth Project (2022JC-47), the Key Research and Development Program of Shaanxi (2021ZDLGY06-04), and Major Program of Shandong Provincial Natural Science Foundation for the Fundamental Research (ZR2022ZD03).

References

[1] M. Xie, Y. Yu, R. Chen, H. Li, J. Wei, Q. Sun, Accountable outsourcing data storage atop blockchain, Comput. Stand. Interfaces 82 (2022) 103628.
[2] Y. Yu, Y. Li, J. Tian, J. Liu, Blockchain-based solutions to security and privacy issues in the Internet of Things, IEEE Wirel. Commun. 25 (6) (2018) 12–18.
[3] H. Li, Y. Li, Y. Yu, B. Wang, K. Chen, A blockchain-based traceable self-tallying e-voting protocol in AI era, IEEE Trans. Netw. Sci. Eng. 8 (2) (2020) 1019–1032.
[4] Y. Yu, Y. Ding, Y. Zhao, Y. Li, Y. Zhao, X. Du, M. Guizani, Lrcoin: leakage-resilient cryptocurrency based on bitcoin for data trading in IOT, IEEE Int. Things J. 6 (3) (2018) 4702–4710.
[5] Y. Li, G. Yang, W. Susilo, Y. Yu, M.H. Au, D. Liu, Traceable monero: anonymous cryptocurrency with enhanced accountability, IEEE Trans. Dependable Secure Comput. 18 (2) (2019) 679–691.
[6] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, S. Riahi, Generalized channels from limited blockchain scripts and adaptor signatures, in: Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2021, pp. 635–664.
[7] A. Poelstra, Scriptless scripts, https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-05-milan-meetup/slides.pdf, 2017. (Accessed 10 March 2023).
[8] Technical Report, The bitcoin lightning network: scalable off-chain instant payments, https://lightning.network/lightning-network-paper.pdf, 2016. (Accessed 1 May 2023).
[9] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, M. Maffei, Anonymous multi-hop locks for blockchain scalability and interoperability, in: Network and Distributed Systems Security Symposium, 2019.
[10] A. Deshpande, M. Herlihy, Privacy-preserving cross-chain atomic swaps, in: International Conference on Financial Cryptography and Data Security, Springer, 2020, pp. 540–549.
[11] D.J. Bernstein, N. Duif, T. Lange, P. Schwabe, B.-Y. Yang, High-speed high-security signatures, J. Cryptogr. Eng. 2 (2) (2012) 77–89.
[12] S. Josefsson, I. Liusvaara, Edwards-curve digital signature algorithm (EdDSA), https://www.rfc-editor.org/rfc/rfc8032, 2017. (Accessed 13 June 2023).
[13] A. Erwig, S. Faust, K. Hostáková, M. Maitra, S. Riahi, Two-party adaptor signatures from identification schemes, in: IACR International Conference on Public-Key Cryptography, Springer, 2021, pp. 451–480.
[14] D. Yan, M. Xie, Y. Zhao, W. Wang, Y. Yu, Two-party eddsa signature scheme against differential fault attack, J. Softw. 34 (2) (2023) 915–931.
[15] L. Fournier, One-time verifiably encrypted signatures aka adaptor signatures, https://tinyurl.com/y4qxopxp, 2019. (Accessed 29 March 2023).
[16] W. Dai, T. Okamoto, G. Yamamoto, Stronger security and generic constructions for adaptor signatures, in: International Conference on Cryptology in India, Springer, 2022, pp. 52–77.
[17] Z. Bao, D. He, C. Peng, M. Luo, K.-K.R. Choo, An identity-based adaptor signature scheme and its applications in the blockchain system, IEEE Open J. Comput. Soc. 4 (2023) 231–242.
[18] X. Zhu, D. He, Z. Bao, C. Peng, M. Luo, Two-party adaptor signature scheme based on IEEE p1363 identity-based signature, IEEE Open J. Commun. Soc. (2023), https://doi.org/10.1109/OJCOMS.2023.3325106.
[19] M.F. Esgin, O. Ersoy, Z. Erkin, Post-quantum adaptor signatures and payment channel networks, in: European Symposium on Research in Computer Security, Springer, 2020, pp. 378–397.

[20] E. Tairi, P. Moreno-Sanchez, M. Maffei, Post-quantum adaptor signature for privacy-preserving off-chain payments, in: International Conference on Financial Cryptography and Data Security, Springer, 2021, pp. 131–150.
[21] B. Manuel, P. Feldman, S. Micali, Non-interactive zero-knowledge and its applications, in: ACM Symposium on Theory of Computing, ACM, 1988, pp. 103–112.
[22] D. Pointcheval, J. Stern, Security arguments for digital signatures and blind signatures, J. Cryptol. 13 (2000) 361–396.