The Efficient ECDSA-based Adaptor Signature Schemes

Binbin Tu, Yu Chen, and Min Zhang

School of Cyber Science and Technology, Shandong University, Qingdao, China

Abstract. Adaptor signature can tie together transaction authorization and witness extraction and has become an important tool for solving the scalability and interoperability problems in the blockchain. Aumayr et al. first provide the formalization of the adaptor signature and present a provably secure ECDSA-based adaptor signature. However, their scheme requires zero-knowledge proof in the pre-signing phase to ensure the signer works correctly which leads to low efficiency.
In this paper, we construct efficient ECDSA-based adaptor signature schemes in which zero-knowledge proofs can be done offline, and give security proof based on ECDSA. The online pre-signing algorithm of our schemes is similar to the ECDSA signing algorithm except for modifying some parameters, and can enjoy the same efficiency as ECDSA. In particular, considering specific verification scenarios, such as (batched) atomic swaps, the pre-signature of our scheme can reduce the number of zero-knowledge proofs that further improves the efficiency. What's more, we develop batched atomic swaps and achieve the protocols efficiently based on our schemes. Lastly, we conduct an experimental evaluation, demonstrating that the performance of our ECDSA-based adaptor signature reduces online pre-signing time about 60% compared with the state-of-the-art ECDSA-based adaptor signature.

Keywords: Adaptor signature, ECDSA, ECDSA-based adaptor signature, batched atomic swaps

1 Introduction

Adaptor signatures (AS), also known as scriptless scripts, are introduced by Poelstra [19] and recently formalized by Aumayr et al. [3,4] AS can be seen as an extension over a digital signature with respect to an instance of a hard relation. Namely, the signer can generate a pre-signature with an instance of a hard relation, such that the pre-signature can be adapted into a full signature by using the witness of the hard relation. What's more, the witness can be extracted by using the pre-signature and the full signature. Therefore, AS can provide the following properties: (i) only the signer knowing the pre-signing key can generate the pre-signature; (ii)only the user knowing the witness of the hard relation can convert the pre-signature into a full signature; (iii) anybody can check the validity of the pre-signature and the corresponding signature and use them to extract the witness of the hard relation.

Tying together the signature and witness extraction, AS brings about various advantages of reducing the operations on-chain and supporting advanced functionality beyond the limitation of the blockchain's scripting language. AS has been shown highly useful in many blockchain applications such as payment channels [3,5,8,20,2], payment routing in payment channel networks [10,16,17,11], atomic swaps [9,14,11].

Poelstra [19] first gives a Schnorr-based AS that is limited to cryptocurrencies using Schnorr signatures [21] and thus is not compatible with those systems, prominently Bitcoin. Then, Moreno-Sanchez and Kate [18] present an ECDSA-based AS and its two-party version, but their schemes are not clear how to prove security. Malavolta et al. [16] present protocols for two-party adaptor signatures based on ECDSA [1]. However, they do not define AS as a stand-alone primitive and formalize the security definition for the threshold primitive and hence the security of their schemes has not been analyzed completely, such as the lack of the witness extractability. Until Aumayr et al. [3,4] first formalize AS as a standalone primitive and prove the security of their ECDSA-based AS based on the strong unforgeability of positive ECDSA in the Universal Composability (UC) framework [6]. They exquisitely modify the hard relation in [18], by adding a zero-knowledge proof such that they can extract the witness in the random model [13], namely "self-proving structure". However, in the pre-signing phase, the signer needs to use the random number as witness to compute a zero-knowledge proof to tie a pre-signing public parameter and a pre-signature that reduces the efficiency of online pre-signing.

1.1 Our Contributions

In this paper, we propose an ECDSA-based AS (ECDSA-AS) and use "self-proving structure" $I_Y = (Y, \pi_Y)$ [3] to prove the security based on positive ECDSA in UC framework [6]. And then, we develop two more efficient ECDSA-AS schemes ECDSA-AS1 and ECDSA-AS2 by running zero-knowledge proof offline. In particular, considering specific verification scenarios [1], such as (batched) atomic swaps, in which only the participants verifies the pre-signatures, ECDSA-AS1/2 can reduce the number of zero-knowledge proofs in pre-signing phase and further improves the efficiency. We briefly show the main techniques as follows:

ECDSA-based adaptor signature. ECDSA-AS can be seen as an extension of ECDSA with a hard relation $(I_Y = (Y = yG, \pi_Y), y)$, where $\pi_Y \leftarrow \mathsf{P}_Y(Y, y)$, P_Y denotes the proving algorithm in the zero-knowledge proof system [2]. We briefly introduce our ECDSA-AS as follows: $(Q = xG, x)$ denotes ECDSA verification key and signing key. The signer computes a pre-signing public parameter $Z =$

[1] Common verification scenarios require that everyone can verify the pre-signatures.
[2] The zero-knowledge proof system requires straight-line extractor, also namely online extractor [13]. We need this property in order to prove our ECDSA-AS secure [3].

xY and uses the signing key x as the witness to compute a zero-knowledge proof $\pi_Z \leftarrow \mathsf{P}_Z((G,Q,Y,Z),x)$ [3], then chooses a random number $k \leftarrow \mathbb{Z}_q$, and computes $r = f(KY)$ [4], $\hat{s} = k^{-1}(h(m) + rx) \bmod q$, and outputs the pre-signature $\hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$. The verification algorithm checks the validity of π_Z and $r =^? f(\hat{s}^{-1} \cdot h(m) \cdot Y + \hat{s}^{-1} \cdot r \cdot Z)$. The adaptor algorithm takes the witness y and the pre-signature $\hat{\sigma}$ as inputs to compute $s = \hat{s} \cdot y^{-1} \bmod q = k^{-1}y^{-1}(h(m)+rx) \bmod q$, and outputs ECDSA signature $\sigma = (r, s)$. The extraction algorithm can extract the witness from ECDSA signature and pre-signature by computing $y = \hat{s}/s$.

We also use "self-proving structure" $(I_Y = (Y, \pi_Y), y)$ following [3] to ensure provable security. Intuitively speaking, in the security proof, because the zero-knowledge proof system holds straight-line extractability [5], the simulator can extract the witness y from the instance (Y, π_Y), then it can use ECDSA signing oracle to get ECDSA signature $\sigma = (r, s)$ and simulates the pre-signing oracle by computing the pre-signature $\hat{s} = s \cdot y \bmod q$.

We observe that the pre-signing public parameter $Z = xY = yQ$ and the zero-knowledge proof π_Z in the pre-signing phase of our ECDSA-AS is independent of the message and the random number, so we can construct two efficient ECDSA-AS schemes ECDSA-AS1/2 by computing Z and π_Z offline, where ECDSA-AS1 uses the pre-signing key x as the witness to compute $(Z = xY, \pi_Z)$, and ECDSA-AS2 uses the witness y of hard relation (I_Y, y) as the witness to compute $(Z = yQ, \pi_Z)$.

Offline/online pre-signing. In [18,3], the signer computes the pre-signing public parameter $K = kY$ and proves the hard relation $((G, \hat{K} = kG, Y, K), k)$ satisfies equality of discrete logarithms $\pi_K \leftarrow \mathsf{P}((G, \hat{K}, Y, K), k)$, that is, there exists the witness k that is the *random number* used in the pre-signing algorithm, such that $\hat{K} = kG$ and $K = kY$.

In ECDSA-AS1, the signer computes the pre-signing public parameter $Z = xY$ and proves the hard relation $((G,Q,Y,Z),x)$ satisfies equality of discrete logarithms $\pi_Z \leftarrow \mathsf{P}_Z((G,Q,Y,Z),x)$, that is, there exists the witness x that is the *pre-signing key*, such that $Q = xG$ and $Z = xY$.

In ECDSA-AS2, the signer computes the pre-signing public parameter $Z = yQ$ and proves the hard relation $((G,Y,Q,Z),y)$ satisfies equality of discrete logarithms $\pi_Z \leftarrow \mathsf{P}_Z((G,Y,Q,Z),y)$, that is, there exists the witness y that is the *witness of hard relation* $(I_Y = (Y, \pi_Y), y)$, such that $Y = yG$ and $Z = yQ$.

Compared with [18,3], ECDSA-AS1/2 can compute the pre-signing public parameter Z and the zero-knowledge proof π_Z *offline*. In particular, in ECDSA-AS2, because of using y as the witness, the hard relation chooser who has the

[3] This zero-knowledge proof system does not require straight-line extractor. Such a proof can be derived by applying the Fiat-Shamir heuristic [12] to Chaum-Pedersen $\sum$-protocol [7] for the language comprising valid DDH tuples.

[4] The function f is defined as the projection to x-coordinate.

[5] The straight-line extractability property allows for extraction of a witness y for a statement Y from a proof π_Y in the random oracle model and is useful for models where the rewinding proof technique is not allowed, such as UC.

witness y and gets the verification key Q_i of any participants can generates all pre-signing public parameter $Z_i = yQ_i$ and the proof π_{Z_i} *in a batch and offline* for all participants.

Performance. We recall ECDSA-AS [18,3] and show the theoretical and experimental analysis of our ECDSA-AS schemes. The online pre-signing of ECDSA-AS1/2 can achieve the same level of efficiency as ECDSA. In particular, ECDSA-AS1/2 only computes once point multiplication operation, while ECDSA-AS in [18,3] need four times point multiplication operation in online pre-signing phase. Lastly, we realize our schemes based on the OpenSSL, and the implementation is given in https://github.com/tbb-tobebetter/ECDSA-AS. The experimental results show that ECDSA-AS1/2 reduces online pre-signing time about 60% compared with the state-of-the-art ECDSA-AS [3].

Applications. We observe that atomic swaps can be seen as *specific verification scenarios* in which the pre-signatures need not to be verified by anyone (including miners), because the pre-signature is not published on the blockchain. Based on this observation, constructing atomic swap based on ECDSA-AS1/2 can further reduce one zero-knowledge proof compared with ECDSA-AS [18,3], because two-party can compute and check the pre-signing public parameter $Z = xY = yQ$ without zero-knowledge proof π_Z.

We develop *batched atomic swaps*, in which one user U_0 can exchange different cryptocurrencies with many users U_i, $i \in [n]$ in a batch. Compared with running independently n times atomic swaps between U_0 and U_i, $i \in [n]$, batched atomic swaps can reduce the number of hard relations (Y, y) from n to 1. Batched atomic swaps can be applied to one party with many addresses (accounts) or one party with a lot of transactions that needs to exchange with many users once, such as the scenario of the Exchange. The batched atomic swaps can also be seen as specific verification scenarios, so constructing batched atomic swaps based on ECDSA-AS1/2 can further reduce $n-1$ pre-signing public parameters and $2n-1$ zero-knowledge proofs compared with ECDSA-AS [18,3].

2 Preliminaries

2.1 Notations

For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \cdots, n\}$, 1^λ denotes the string of λ ones. Throughout, we use λ to denote the security parameter. A function is negligible in λ, written $\mathsf{negl}(\lambda)$, if it vanishes faster than the inverse of any polynomial in λ. We denote a probabilistic polynomial-time algorithm by PPT. If S is a set then $s \leftarrow S$ denotes the operation of sampling an element s of S at random.

2.2 Hard Relation and Zero-Knowledge Proof

We recall the definition of a hard relation R with statement/witness pairs ($stat = (G, Y = yG), y$) [3]. Let $\mathsf{L_R}$ be the associated language defined as $\mathsf{L_R} = \{(G, Y) | \exists\, y$

s.t. $((G, Y), y) \in \mathsf{R}\}$. We say that R is a hard relation if the following holds: (i) There exists a PPT sampling algorithm $\mathsf{GenR}(1^\lambda)$ that on input 1^λ outputs a statement/witness pair $((G, Y), y) \in \mathsf{R}$; (ii) The relation is poly-time decidable; (iii) For all PPT $\mathcal{A}$, the probability of $\mathcal{A}$ on input (G, Y) outputting y is negligible.

We also recall the definition of a non-interactive zero-knowledge proof of knowledge (NIZKPoK) with straight-line extractors as introduced in [13]. More formally, a pair (P, V) of PPT algorithms is called a non-interactive zero-knowledge proof of knowledge with a straight-line extractor for a relation R, random oracle $\mathcal{H}$ and security parameter λ if the following holds: (i) Completeness: For any $((G, Y), y) \in \mathsf{R}$, it holds that $\mathsf{V}((G, Y), \pi \leftarrow \mathsf{P}((G, Y), y)) = 1$; (ii) Zero knowledge: There exists a PPT simulator S, which on input (G, Y) can simulate the proof π for any $((G, Y), y) \in \mathsf{R}$. (iii) Straight-line extractability: There exists a PPT straight-line extractor K with access to the sequence of queries to the random oracle and its answers, such that given $((G, Y), \pi)$, the algorithm K can extract the witness y with $((G, Y), y) \in \mathsf{R}$. For convenience, we omit the parameter G in this paper.

2.3 Signature Scheme

A signature scheme consists of three algorithms $\sum = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$.

- $\mathsf{Gen}(1^\lambda) \rightarrow (vk, sk)$. The key generation algorithm takes the security parameter as input and outputs a verification key vk and a secret key sk.
- $\mathsf{Sign}_{sk}(m) \rightarrow \sigma$. The signing algorithm takes the secret key sk and the message $m \in \{0,1\}^*$ as input, and outputs a signature σ.
- $\mathsf{Vrfy}_{vk}(m, \sigma) \rightarrow 0/1$. The verification algorithm takes the verification key vk, the message $m \in \{0,1\}^*$, the signature σ as input, and outputs 0 or 1.

The correctness is that for any $(vk,\, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and $m \in \{0,1\}^*$, we have $\mathsf{Vrfy}_{vk}(m,\, \mathsf{Sign}_{sk}(m)) \rightarrow 1$.

Definition 1 (SUF-CMA security). *A signature scheme $\sum$ is SUF–CMA secure if for every PPT adversary $\mathcal{A}$ there exists a negligible function* negl *such that:* $\Pr[sSigForge_{\mathcal{A},\sum}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where the experiment $sSigForge_{\mathcal{A},\sum}$ is defined as follows:*

$\text{sSigForge}_{\mathcal{A},\sum}(\lambda)$	$\mathcal{O}_{\text{Sign}_{sk}}(m)$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$	$\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}_{sk}}(\cdot)}(vk)$	$\mathcal{Q} = \mathcal{Q} \cup \{m, \sigma\}$
$\text{return}((m^*, \sigma^*) \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{vk}(m^*, \sigma^*))$	return σ

2.4 Adaptor Signature Scheme

An adaptor signature scheme [3] w.r.t. a hard relation $\mathsf{R} = \{Y, y\}$ and a signature scheme $\sum = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ consists of four algorithms $\mathit{\Pi}_{\mathsf{R},\sum} = (\mathsf{pSign},\, \mathsf{pVrfy},\, \mathsf{Adapt},\, \mathsf{Ext})$ defined as:

– $\mathsf{pSign}_{sk}(m, Y) \to \hat{\sigma}$: on input a pre-signing key sk, an instance Y and a message $m \in \{0,1\}^*$, outputs a pre-signature $\hat{\sigma}$.
– $\mathsf{pVrfy}_{vk}(m, Y, \hat{\sigma}) \to 0/1$: on input a verification key vk, a pre-signature $\hat{\sigma}$, an instance Y and a message $m \in \{0,1\}^*$, outputs a bit $b \in \{0,1\}$.
– $\mathsf{Adapt}(\hat{\sigma}, y) \to \sigma$: on input a pre-signature $\hat{\sigma}$ and a witness y, outputs a signature σ.
– $\mathsf{Ext}(\sigma, \hat{\sigma}, Y) \to y$: on input a signature σ, a pre-signature $\hat{\sigma}$ and an instance Y, outputs a witness y such that $(Y, y) \in \mathsf{R}$, or $\perp$.

In addition to the standard signature correctness, an AS has to satisfy pre-signature correctness.

Definition 2 (Pre-signature correctness). *An adaptor signature scheme $\Pi_{\mathsf{R},\sum}$ satisfies pre-signature correctness if for every λ, every message $m \in \{0,1\}^*$ and every statement/witness pair $(Y, y) \in \mathsf{R}$, the following holds:*

$$\Pr\left[\begin{array}{c|c} \mathsf{pVrfy}_{vk}(m, Y, \hat{\sigma}) \to 1 \wedge & \mathsf{Gen}(1^\lambda) \to (sk, vk) \\ \mathsf{Vrfy}_{vk}(m, \sigma) \to 1 \wedge & \mathsf{pSign}_{sk}(m, Y) \to \hat{\sigma} \\ (Y, y') \in \mathsf{R} & \mathsf{Adapt}(\hat{\sigma}, y) \to \sigma \\ & \mathsf{Ext}(\sigma, \hat{\sigma}, Y) \to y' \end{array}\right] = 1$$

We review the existential unforgeability under chosen message attack for AS (aEUF–CMA), pre-signature adaptability, and witness extractability [3].

Definition 3 (aEUF–CMA security). *An adaptor signature scheme $\Pi_{\mathsf{R},\sum}$ is aEUF–CMA secure if for every PPT adversary $\mathcal{A}$ there exists a negligible function* negl *such that:* $\Pr[aSigForge_{\mathcal{A},\Pi_{\mathsf{R},\sum}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where the experiment $aSigForge_{\mathcal{A},\Pi_{\mathsf{R},\sum}}$ is defined as follows:*

$\text{aSigForge}_{\mathcal{A},\Pi_{\mathsf{R},\sum}}(\lambda)$	$\mathcal{O}_{\text{Sign}_{sk}}(m)$
$\mathcal{Q} = \emptyset$	$\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$	$\mathcal{Q} = \mathcal{Q} \cup \{m\}$
$m \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}_{sk}}(\cdot), \mathcal{O}_{\text{pSign}_{sk}}(\cdot)}(vk)$	return σ
$(Y, y) \leftarrow \mathsf{GenR}(1^\lambda)$	
$\hat{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$	$\mathcal{O}_{\text{pSign}_{sk}}(m, Y)$
$\sigma \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}_{sk}}(\cdot), \mathcal{O}_{\text{pSign}_{sk}}(\cdot)}(\hat{\sigma}, Y)$	$\hat{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$
return $(m \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{vk}(m, \sigma))$	$\mathcal{Q} = \mathcal{Q} \cup \{m\}$
	return $\hat{\sigma}$

Definition 4 (Pre-signature adaptability). *An adaptor signature scheme $\Pi_{\mathsf{R},\sum}$ satisfies pre-signature adaptability if for any λ, any message $m \in \{0,1\}^*$, any statement/witness pair $(Y, y) \in \mathsf{R}$, any key pair $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and any pre-signature $\hat{\sigma}$ with $\mathsf{pVrfy}_{vk}(m, Y, \hat{\sigma}) \to 1$, we have $\Pr[\mathsf{Vrfy}_{vk}(m,\ \mathsf{Adapt}(\hat{\sigma},\ y)) \to 1] = 1$.*

The aEUF–CMA security together with the pre-signature adaptability ensures that a pre-signature for Y can be transferred into a valid signature if and only if the corresponding witness y is known.

Definition 5 (Witness extractability). *An adaptor signature scheme* $\Pi_{\mathsf{R},\sum}$ *is witness extractable if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that:* $\Pr[aWitExt_{\mathcal{A},\Pi_{\mathsf{R},\sum}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where the experiment* $aWitExt_{\mathcal{A},\Pi_{\mathsf{R},\sum}}$ *is defined as follows*

$\text{aWitExt}_{\mathcal{A},\Pi_{\mathsf{R},\sum}}(\lambda)$	$\mathcal{O}_{\text{Sign}_{sk}}(m)$
$\mathcal{Q} = \emptyset$	$\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$	$\mathcal{Q} = \mathcal{Q} \cup \{m\}$
$(m, Y) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}_{sk}}(\cdot),\mathcal{O}_{\text{pSign}_{sk}}(\cdot)}(vk)$	return σ
$\hat{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$	
$\sigma \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}_{sk}}(\cdot),\mathcal{O}_{\text{pSign}_{sk}}(\cdot)}(\hat{\sigma})$	$\mathcal{O}_{\text{pSign}_{sk}}(m, Y)$
$y' \leftarrow \mathsf{Ext}(\sigma, \hat{\sigma}, Y)$	$\hat{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$
return $(m \notin \mathcal{Q} \wedge (Y, y') \notin \mathsf{R}$	$\mathcal{Q} = \mathcal{Q} \cup \{m\}$
$\wedge \mathsf{Vrfy}_{vk}(m, \sigma))$	return $\hat{\sigma}$

The witness extractability guarantees that a valid signature/pre-signature pair $(\sigma, \hat{\sigma})$ for message/statement (m, Y) can be used to extract the corresponding witness y. There is one crucial difference between aWitExt and aSigForge: the adversary is allowed to choose the challenge instance Y. Hence, he knows a witness for Y so he can generate a valid signature on the forgery message m. However, this is not sufficient to win the experiment aWitExt. The adversary wins only if the valid signature does not reveal a witness for Y.

2.5 ECDSA

We review the ECDSA scheme [1] $\sum_{\text{ECDSA}} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ on a message $m \in \{0,1\}^*$ as follows. Let $\mathbb{G}$ be an Elliptic curve group of order q with base point (generator) G and let $pp = (\mathbb{G}, G, q)$ be the public parameter.

- $\mathsf{Gen}(pp) \to (Q, x)$: The key generation algorithm uniformly chooses a secret signing key $x \leftarrow \mathbb{Z}_q$, and calculates the verification key $Q = x \cdot G$, and outputs $(sk = x, vk = Q)$.
- $\mathsf{Sign}_{sk}(m) \to (r, s)$. The signing algorithm chooses $k \leftarrow \mathbb{Z}_q$ randomly and computes $r = f(kG)$ and $s = k^{-1}(h(m) + rx)$, where h is a hash function and f is defined as the projection to the x-coordinate.
- $\mathsf{Vrfy}_{vk}(m, \sigma) \to 0/1$. The verification algorithm computes $r' = f(s^{-1} \cdot (m' \cdot G + r \cdot Q))$. If $r = r' \bmod q$, outputs 1, otherwise, outputs 0.

Following [15,16,3], we also use the *positive* ECDSA which guarantees that if (r, s) is a valid signature.

3 ECDSA-based Adaptor Signature

In this section, we present a construction of ECDSA-AS $\Pi_{\mathsf{R},\sum} = (\mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Ext})$ w.r.t. a hard relation R and a ECDSA signature $\sum = (\mathsf{Gen},$

Sign, Vrfy). Let $(Q = xG, x)$ be the verification key and signing key of ECDSA. We define hard relations $\mathsf{R} = \{(I_Y = (Y, \pi_Y \leftarrow \mathsf{P}_Y(Y, y)), y)|\ Y = yG \wedge \mathsf{V}_Y(I_Y) = 1\}$ and $\mathsf{R}_Z = \{(I_Z = (G, Q, Y, Z), x)|Q = xG \wedge Z = xY\}$ where P_Y and V_Y denotes the proving and verification algorithm of a NIZKPoK with straight-line extractability [13], P_Z and V_Z denotes the proving and verification algorithm of a NIZK.

- $\mathsf{pSign}_{(vk,sk)}(m, I_Y) \to \hat{\sigma}$: on input a key-pair $(vk, sk) = (Q, x)$, a message m and an instance $I_Y = (Y, \pi_Y)$, the algorithm computes the pre-signing public parameter $Z = xY$, runs $\pi_Z \leftarrow \mathsf{P}_Z(I_Z = (G, Q, Y, Z), x)$, and chooses $k \leftarrow \mathbb{Z}_q$, computes $r = f(kY)$, $\hat{s} = k^{-1}(h(m) + rx) \bmod q$ and outputs the pre-signature $\hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$.
- $\mathsf{pVrfy}_{vk}(m, I_Y, \hat{\sigma}) \to 0/1$: on input the verification key $vk = Q$, a message m, an instance I_Y, and a pre-signature value $\hat{\sigma}$, the algorithm outputs $\perp$ if $\mathsf{V}_Z(I_Z) \to 0$, otherwise, it computes $r' = f(\hat{s}^{-1} \cdot (h(m) \cdot Y + r \cdot Z))$, and if $r' = r$, outputs 1, else outputs 0.
- $\mathsf{Adapt}(y, \hat{\sigma}) \to \sigma$: on input the witness y, and pre-signature $\hat{\sigma}$, the algorithm computes $s = \hat{s} \cdot y^{-1} \bmod q$ and outputs the signature $\sigma = (r, s)$.
- $\mathsf{Ext}(\sigma, \hat{\sigma}, I_Y) \to y$: on input the signature σ, the pre-signature $\hat{\sigma}$ and the instance I_Y, the algorithm computes $y = \hat{s}/s \bmod q$. If $(I_Y, y) \in \mathsf{R}$, it outputs y, else outputs $\perp$.

Note that in the pre-signing phase, our ECDSA-AS uses the signing key x as the witness to compute the pre-signing public parameter $Z = xY$ and zero-knowledge proof π_Z, then the later pre-signing operation is similar to original ECDSA signing algorithm except for modifying some parameters by using (Z, Y) as the verification key and base point instead of (Q, G).

Theorem 1. *Assuming that the positive ECDSA signature $\sum$ is SUF-CMA secure, and R is a hard relation, NIZKPoK and NIZK used in above scheme are secure, above ECDSA-AS $\Pi_{\mathsf{R},\sum}$ is secure in random oracle model.*

We prove that our ECDSA-AS scheme satisfies pre-signature adaptability, pre-signature correctness, aEUF–CMA security, and witness extractability as follows.

Lemma 1. (Pre-signature adaptability) *The ECDSA-based adaptor signature scheme $\Pi_{\mathsf{R},\sum}$ satisfies pre-signature adaptability.*

Proof. For any $(I_Y, y) \in \mathsf{R}$, $m \in \{0,1\}^*$, $G, Q, Y, Z \in \mathbb{G}$ and $\hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$. For $\mathsf{pVrfy}_{vk}(m, I_Y, \hat{\sigma}) \to 1$. That is, $Y = yG, Z = xY = yQ = xyG$, $\hat{K} = (h(m) \cdot \hat{s}^{-1})Y + r \cdot \hat{s}^{-1}Z = kY$, $r' = f(\hat{K}) = f(kY) = r$.

By definition of Adapt, we know that $\mathsf{Adapt}(\hat{\sigma}, y) \to \sigma$, where $\sigma = (r, s), s = \hat{s} \cdot y^{-1} = (yk)^{-1}(h(m) + rx) \bmod q$. Hence, we have

$$K' = (h(m) \cdot s^{-1})G + r \cdot s^{-1}Q = kY.$$

Therefore, $r' = f(K') = f(kY) = r$. That is $\mathsf{Vrfy}_{vk}(m, \sigma) \to 1$.

Lemma 2. (Pre-signature correctness) *The ECDSA-based adaptor signature scheme* $\Pi_{\mathsf{R},\sum}$ *satisfies pre-signature correctness.*

Proof. For any $x, y \in \mathbb{Z}_q$, $Q = xG, Y = yG$ and $m \in \{0,1\}^*$. For $\mathsf{pSign}_{(vk,sk)}(m, I_Y) \to \hat{\sigma} = (r,\ \hat{s},\ Z,\ \pi_Z)$, it holds that $Y = yG, Z = xY$, $\hat{s} = k^{-1}(h(m) + rx) \bmod q$ for some $k \leftarrow \mathbb{Z}_q$. Set $\hat{K} = (h(m) \cdot \hat{s}^{-1})Y + r \cdot \hat{s}^{-1}Z = kY$.

Therefore, $r' = f(\hat{K}) = f(kY) = r$, we have $\mathsf{pVrfy}_{vk}(m, I_Y, \hat{\sigma}) \to 1$. By Lemma 1, this implies that $\mathsf{Vrfy}_{vk}(m, \sigma) \to 1$, for $\mathsf{Adapt}(\hat{\sigma}, y) \to \sigma = (r, s)$. By the definition of Adapt, we know that $s = \hat{s} \cdot y^{-1}$ and hence

$$\mathsf{Ext}(\sigma, \hat{\sigma}, I_Y) = \hat{s}/s = \hat{s}/(\hat{s}/y) = y.$$

Lemma 3. (aEUF–CMA security) *Assuming that the positive ECDSA signature scheme* $\sum$ *is SUF–CMA secure,* R *is a hard relation, NIZKPoK and NIZK are secure, ECDSA-AS* $\Pi_{\mathsf{R},\sum}$ *as defined above is aEUF–CMA secure.*

Proof. We prove the aEUF–CMA security by reduction to the strong unforgeability of positive ECDSA signatures. Following [3], our ECDSA-AS uses the same hard relation $(I_Y = (Y, \pi_Y), y)$, where NIZKPoK_Y satisfies straight-line extractability, so the simulator can extract the witness from I_Y. Our proof works by showing that, for any PPT adversary $\mathcal{A}$ breaking aEUF–CMA security of the ECDSA-AS, we can construct a PPT simulator $\mathcal{S}$ who can break the SUF–CMA security of ECDSA. $\mathcal{S}$ has access to the signing oracle $\mathcal{O}_{\text{ECDSA-Sign}}$ of ECDSA and the random oracle $\mathcal{H}_{\text{ECDSA}}$. It needs to simulate oracle for $\mathcal{A}$, namely random oracle ($\mathcal{H}$), signing oracle ($\mathcal{O}_{\text{Sign}}$) and pre-signing oracle ($\mathcal{O}_{\text{pSign}}$).

The simulator $\mathcal{S}$ can use its oracle $\mathcal{O}_{\text{ECDSA-Sign}}$ and $\mathcal{H}_{\text{ECDSA}}$ to simulate $\mathcal{O}_{\text{Sign}}$ and $\mathcal{H}$. The main challenge is simulating $\mathcal{O}_{\text{pSign}}$ queries. Because $\mathcal{S}$ can extract the witness from I_Y, it can use its oracle $\mathcal{O}_{\text{ECDSA-Sign}}$ to get a full signature on m which is queried by $\mathcal{A}$, and transform the full signature into a pre-signature. What's more, S can use the zero-knowledge property of NIZK_Z to simulate π_Z for a statement (G, Q, Y, Z) without knowing the corresponding witness x.

We prove security by describing a sequence of games $G_0, \cdots, G_4$, where G_0 is the original aSigForge game. Then we show that for all $i = 0, \cdots, 3$, G_i and G_{i+1} are indistinguishable.

- Game G_0: This game corresponds to the original aSigForge game.
- Game G_1: This game works as G_0 with the exception that upon the adversary outputting a forgery σ^*. The game checks if completing the pre-signature $\hat{\sigma}$ using the secret value y results in σ^*. If yes, the game aborts.
- Game G_2: This game works as G_1 excepting that in $\mathcal{O}_{\text{pSign}}$, this game extracts a witness y' by executor K. The game aborts if $(I_Y, y') \notin \mathsf{R}$.
- Game G_3: This game works as G_2 excepting that this game extracts a witness y and calculates $Z = yQ$, and simulates a zero-knowledge proof π_S.
- Game G_4: In this game, upon receiving the challenge message m^* from $\mathcal{A}$, the game creates a full signature by executing the Sign algorithm and transforms the resulting signature into a pre-signature in the same way as in the previous game G_3 during the $\mathcal{O}_{\text{pSign}}$ execution.

There exists a simulator that perfectly simulates G_4 and uses $\mathcal{A}$ to win a positive ECDSA strongSigForge game.

– Signing oracle queries: Upon $\mathcal{A}$ querying $\mathcal{O}_{\text{Sign}}$ on input m, $\mathcal{S}$ forwards m to its oracle $\mathcal{O}_{\text{ECDSA-sign}}$ and forwards its response to $\mathcal{A}$.
– Random oracle queries: Upon $\mathcal{A}$ querying $\mathcal{H}$ on input x, if $H[x] = \perp$, then $\mathcal{S}$ queries $\mathcal{H}_{\text{ECDSA}}(x)$, otherwise the simulator returns $\mathcal{H}[x]$.
– Pre-signing oracle queries: Upon $\mathcal{A}$ querying $\mathcal{O}_{\text{pSign}}$ on input (m, I_Y), the simulator extracts y, and forwards m to $\mathcal{O}_{\text{ECDSA-sign}}$ and gets (r, s), then $\mathcal{S}$ computes $\hat{s} = s \cdot y$, $Z = yQ = xY$ and simulates a zero-knowledge proof π_S, and outputs $(r, \hat{s}, Z, \pi_S)$.
– In the challenge phase: Upon $\mathcal{A}$ outputting the challenge message m^*, $\mathcal{S}$ generates $(I_Y, y) \leftarrow \mathsf{GenR}(1^\lambda)$, forwards m^* to $\mathcal{O}_{\text{ECDSA-sign}}$ and gets (r, s). And then, S generates the pre-signature $\hat{\sigma}^*$ in the same way as during $\mathcal{O}_{\text{pSign}}$. Upon $\mathcal{A}$ outputting σ^*, the simulator outputs (m^*, σ^*) as its own forgery.

Therefore, the simulator S can simulate the views of $\mathcal{A}$. It remains to show that the forgery output by $\mathcal{A}$ can be used by the simulator to win the positive ECDSA strongSigForge game.

Claim 1 *Let Bad_1 be the event that G_1 aborts, then* $\Pr[Bad_1] \leq \mathsf{negl}_1(\lambda)$.

Proof. We prove this claim using a reduction to the hardness of the relation R. The simulator gets a challenge I_Y^*, and it generates a key pair $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ to simulate $\mathcal{A}$'s queries of $\mathcal{H}$, $\mathcal{O}_{\text{Sign}}$ and $\mathcal{O}_{\text{pSign}}$. This simulation of the oracles works as described in G_1. Upon receiving challenge message m^* from $\mathcal{A}$, $\mathcal{S}$ computes a pre-signature $\hat{\sigma} \leftarrow \mathsf{pSign}_{(vk,sk)}(m^*, I_Y^*)$, returns $\hat{\sigma}$ to $\mathcal{A}$ who outputs a forgery σ^*.

Assuming that Bad_1 happened (i.e. $\mathsf{Adapt}(\hat{\sigma}, y) = \sigma^*$), the simulator can extract $y^* \leftarrow \mathsf{Ext}(\sigma^*, \hat{\sigma}, I_Y^*)$. Since the challenge I_Y^* is an instance of the hard relation R and hence equally distributed to the public output of GenR. Hence the probability of $\mathcal{S}$ breaking the hardness of the relation is equal to the probability of the Bad_1 event.

Claim 2 *G_0, G_1, G_2, G_3 and G_4 are computationally indistinguishable.*

Proof. Since G_1 and G_0 are equivalent except if event Bad_1 occurs, it holds that $|\Pr[G_0 = 1] - \Pr[G_1 = 1]| \leq \mathsf{negl}_1(\lambda)$.

According to the straight-line extractability of the NIZKPoK_Y, for a witness y extracted from a proof π_Y of the instance I_Y such that $\mathsf{V}_Y(I_Y, \pi_Y) \rightarrow 1$, it holds that $(I_Y, y) \in \mathsf{R}$ except with negligible probability. It holds that $|\Pr[G_2 = 1] - \Pr[G_1 = 1]| \leq \mathsf{negl}_2(\lambda)$.

Due to the zero-knowledge property of the NIZK_Z, the simulator can compute a proof π_S which is computationally indistinguishable from a proof $\pi_Z \leftarrow \mathsf{P}((G, Q, Y, Z), x)$. Hence, it holds that $|\Pr[G_3 = 1] - \Pr[G_2 = 1]| \leq \mathsf{negl}_3(\lambda)$.

Following above proof, due to the zero-knowledge property of the NIZK_Z, G_4 is indistinguishable from G_3 and it holds that $|\Pr[G_4 = 1] - \Pr[G_3 = 1]| \leq \mathsf{negl}_2(\lambda)$.

Claim 3 (m^*, σ^*) *constitutes a valid forgery in positive ECDSA strongSigForge game.*

Proof. We show that (m^*, σ^*) has not been output by the oracle $\mathcal{O}_{\text{ECDSA-Sign}}$ before. Note that $\mathcal{A}$ has not previously made a query on the challenge message m^* to either $\mathcal{O}_{\text{Sign}}$ or $\mathcal{O}_{\text{pSign}}$. Hence, $\mathcal{O}_{\text{ECDSA-Sign}}$ is only queried on m^* during the challenge phase. As shown in game G_1, the adversary outputs a forgery σ^* which is equal to the signature σ output by $\mathcal{O}_{\text{ECDSA-Sign}}$ during the challenge phase only with negligible probability. Hence, $\mathcal{O}_{\text{ECDSA-Sign}}$ has never output σ^* on query m^* before and consequently (m^*, σ^*) constitutes a valid forgery for positive ECDSA strongSigForge game.

From the games G_0 to G_4, we get that $|\Pr[G_0 = 1] - \Pr[G_4 = 1]| \leq \mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda) + \mathsf{negl}_3(\lambda) + \mathsf{negl}_4(\lambda) \leq \mathsf{negl}(\lambda)$. Since $\mathcal{S}$ provides a perfect simulation of game G_4, we obtain:

$$\begin{aligned}\Pr[\text{aSigForge}_{\mathcal{A},\Pi_{\mathsf{R},\sum}}(\lambda) = 1] = \Pr[G_0 = 1] &\leq \Pr[G_4 = 1] + \mathsf{negl}(\lambda) \\ &\leq \Pr[\text{sSigForge}_{\mathcal{A},\sum}(\lambda) = 1] + \mathsf{negl}(\lambda).\end{aligned}$$

Lemma 4. (Witness extractability). *Assuming that the positive ECDSA scheme is SUF–CMA secure,* R *is a hard relation, NIZKPoK and NIZK are secure, ECDSA-AS* $\Pi_{\mathsf{R},\sum}$ *is witness extractable.*

Proof. Our proof is to reduce the witness extractability to the strong unforgeability of the positive ECDSA. Following the proof of Lemma 3, the simulator $\mathcal{S}$ can use its oracle $\mathcal{O}_{\text{ECDSA-Sign}}$ and $\mathcal{H}_{\text{ECDSA}}$ to simulate $\mathcal{O}_{\text{Sign}}$ and $\mathcal{H}$ of $\mathcal{A}$.

The main challenge is to simulate the pre-signing oracle $\mathcal{O}_{\text{pSign}}$. The crucial difference between aWitExt and aSigForge is that in the challenge phase of aSigForge, I_Y is chosen by challenger, but in the challenge phase of aWitExt, I_Y is chosen by $\mathcal{A}$. That is, $\mathcal{S}$ can not choose (I_Y, y). Following [3], our ECDSA-AS uses the same hard relation $(I_Y = (Y, \pi_Y), y)$, where NIZKPoK_Y satisfies straight-line extractability, so the simulator S can extract the witness y from challenge instance $I_Y = (Y, \pi_Y)$. And then, $\mathcal{S}$ forwards m to $\mathcal{O}_{\text{ECDSA-sign}}$ and gets the signature $\sigma = (r, s)$, then $\mathcal{S}$ computes $\hat{s} = s \cdot y$, $Z = yQ$ and simulates a zero-knowledge proof π_S, and outputs the pre-signature $\hat{\sigma} = (r, \hat{s}, Z, \pi_S)$.

Therefore, we can construct a simulator S following the proof of Lemma 3 except that in the challenge phase, S dose not generate the hard relation (I_Y, y) to get the witness y, but obtains the witness from the instance I_Y chosen by $\mathcal{A}$ based on the straight-line extractability. S can simulate the views of $\mathcal{A}$, so the simulator can win the positive ECDSA strongSigForge game if $\mathcal{A}$ can break the witness extractability of ECDSA-AS.

4 Fast ECDSA-based Adaptor Signature Schemes with Offline Pre-signing

In this section, we construct two fast ECDSA-AS schemes called ECDSA-AS1 and ECDSA-AS2, where ECDSA-AS1 uses the pre-signing key x as the witness to compute $(Z = xY, \pi_Z)$, and ECDSA-AS2 uses the witness y of hard relation (I_Y, y) as the witness to compute $(Z = yQ, \pi_Z)$.

In our ECDSA-AS, the pre-signing public parameter $Z = xY$ and the zero-knowledge proof $\pi_Z \leftarrow \mathsf{P}(I_Z = (G, Q, Y, Z), x)$ are independent of the message m and the random number k, so the signer can compute the pre-signing public parameter and the zero-knowledge proof *offline* before getting the message. ECDSA-AS1 can be designed from our ECDSA-AS directly with offline computing the pre-signing public parameter $Z = xY$ and the zero-knowledge proof π_Z. Refer to the section 3 for specific construction which is ignored here.

We construct efficient ECDSA-AS2 as follows. Formally, Let $(Q = xG, x)$ be the verification key and signing key of ECDSA. We define hard relations $\mathsf{R} = \{(I_Y = (Y, \pi_Y \leftarrow \mathsf{P}_Y(Y, y)), y)|\ Y = yG \wedge \mathsf{V}_Y(I_Y) = 1\}$ and $\mathsf{R}_Z = \{(I_Z = (G, Y, Q, Z), y)|Y = yG \wedge Z = yX\}$.

– $\mathsf{pSign}_{(vk,sk)}(m, I) \to \hat{\sigma}$: on input a key-pair $(vk, sk) = (Q, x)$, a message m and an instance $I = (I_Y, I_Z)$, the algorithm chooses $k \leftarrow \mathbb{Z}_q$, computes $r = f(kY)$, $\hat{s} = k^{-1}(h(m) + rx) \bmod q$ and outputs $\hat{\sigma} = (r, \hat{s})$.
– $\mathsf{pVrfy}_{vk}(m, I, \hat{\sigma}) \to 0/1$: on input the verification key $vk = Q$, a message m, an instance I, and a pre-signature value $\hat{\sigma}$, the algorithm computes $r' = f(\hat{s}^{-1} \cdot (h(m) \cdot Y + r \cdot Z))$, and if $r' = r$, outputs 1, else outputs 0.
– $\mathsf{Adapt}(y, \hat{\sigma}) \to \sigma$: on input the witness y, and pre-signature $\hat{\sigma}$, the algorithm computes $s = \hat{s} \cdot y^{-1} \bmod q$ and outputs the signature $\sigma = (r, s)$.
– $\mathsf{Ext}(\sigma, \hat{\sigma}, I_Z) \to y$: on input the signature σ, the pre-signature $\hat{\sigma}$ and the instance I_Z, the algorithm computes $y = \hat{s}/s \bmod q$. If $(I_Z, y) \in \mathsf{R}$, it outputs y, else outputs $\perp$.

Note that ECDSA-AS2 is similar to our ECDSA-AS except that the signer can compute the pre-signing public parameter $Z = yQ$ and zero-knowledge proof $\pi_Z \leftarrow \mathsf{P}_Z(I_Z = (G, Y, Q, Z), y)$ offline. Before running online pre-signing algorithm, the signer should check the validity of π_Y and π_Z offline to ensure Y and Z is correct.

Correctness. Following the proofs of lemma 1 and lemma 2, our ECDSA-AS1/2 schemes also satisfy pre-signature adaptability and pre-signature correctness.

Security. Our ECDSA-AS1/2 schemes embed the hard relation $(I_Y = (Y, \pi_Y), y)$ [3]. Therefore, in the security proof, the simulator can extract the witness y and simulate the pre-signing oracle. Following the proofs of lemma 3 and lemma 4, our ECDSA-AS1/2 schemes also satisfy aEUF–CMA security and witness extractability.

Offline/online pre-signing. Depending on whether or not the message is required, we could divide pre-signing into the offline and online phases. In the offline phase, the signer can generate the hard relation $I_Y = (Y, \pi_Y \leftarrow \mathsf{P}_Y(Y, y), y)$ and computes the pre-signing public parameter $Z = yQ$ and zero-knowledge proof $\pi_Z \leftarrow \mathsf{P}_Z(I_Z = (G, Y, Q, Z), y)$, and then checks the validity of the zero-knowledge proofs from other signers. In the online phase, the signer uses the correct Y and Z to run the online pre-signing algorithm to generate the pre-signature. As we can see, the online pre-signing algorithm is similar to original ECDSA signing algorithm except for modifying parameters by using (Z, Y) as the verification key and base point instead of (Q, G).

5 Performance and Experimental Results

5.1 Theoretical Analysis

As is shown in Table 1, we give the theoretical analysis of communication cost and efficiency about ECDSA-AS [18,3] and our ECDSA-AS schemes, respectively. For convenience, we omit some simple operations of modular multiplication and addition. The first ECDSA-AS proposed by Moreno-Sanchez et al. [18] does not provide provable security. Then Aumayr et al. [3] uses self-proving structure $(I_Y = (Y, \pi_Y), y)$, and gives a provably secure ECDSA-AS. But this scheme requires that proving $\hat{K} = kG$ and $K = kY$ satisfy equality of discrete logarithms with the witness k that is the random number in the pre-signing phase, so this part must be computed by the signer. What's more, for each message to be signed, the signer needs to choose a new random number and needs to compute a new pre-signing public parameter and a new zero-knowledge proof.

Our ECDSA-AS1/2 can use the witness x or y to prove $Z = xY$ and $Q = xG$ satisfy equality of discrete logarithms, or $Z = yQ$ and $Y = yG$ satisfy equality of discrete logarithms, and this part can be done offline. ECDSA-AS1/2 only computes once point multiplication operation, while ECDSA-AS in [18] and [3] need four times point multiplication operation in online pre-signing phase. In particular, in ECDSA-AS2, the hard relation chooser can help to compute all pre-signing public parameters $Z_i = yQ_i$ and zero-knowledge proofs for all other participants *in a batch and offline*.

Table 1: Communication Cost and Efficiency Comparison

Schemes	PK size	SK size	Pre-signature size	Online pSign	pVrfy	Batched zk proof	Offline zk proof	Provable security
ECDSA-AS [18]	$\vert\mathbb{G}\vert$	$\vert\mathbb{Z}_q\vert$	$\vert\mathbb{G}\vert + 4\vert\mathbb{Z}_q\vert$	4Exp	6Exp	$\times$	$\times$	?
ECDSA-AS [3]	$\vert\mathbb{G}\vert$	$\vert\mathbb{Z}_q\vert$	$\vert\mathbb{G}\vert + 4\vert\mathbb{Z}_q\vert$	4Exp	6Exp	$\times$	$\times$	$\surd$
Our ECDSA-AS	$\vert\mathbb{G}\vert$	$\vert\mathbb{Z}_q\vert$	$\vert\mathbb{G}\vert + 4\vert\mathbb{Z}_q\vert$	4Exp	6Exp	$\times$	$\surd$	$\surd$
Our ECDSA-AS2	$\vert\mathbb{G}\vert$	$\vert\mathbb{Z}_q\vert$	$2\vert\mathbb{Z}_q\vert$	Exp	6Exp	$\surd$	$\surd$	$\surd$

‡ $|\mathbb{G}|$ and $|\mathbb{Z}_q|$ denotes the size of the element in the group $\mathbb{G}$ and $\mathbb{Z}_q$, respectively. Exp denotes point multiplication operation.

5.2 Experimental Analysis

In order to evaluate the practical performance of our schemes, we implement the ECDSA-AS [3], our ECDSA-AS and ECDSA-AS2 based on the OpenSSL library.

The implementation is given in https://github.com/tbb-tobebetter/ECDSA-AS. The program is executed on an Intel Core i5 CPU 2.3 GHz and 8GB RAM running macOS High Sierra 10.13.3 system.

As depicted in Figure 1, we run our implementations on the standard NIST curves. We run ECDSA-AS [3], our ECDSA-AS and ECDSA-AS2 many times respectively, and show the efficiency of the online pre-signing and the verification operation. The average running times over 1000 executions of the online pre-signing operation in ECDSA-AS [3], our ECDSA-AS and ECDSA-AS1 are 174.75 μs, 198.91 μs and 74.74 μs. Therefore, ECDSA-AS2 is comparable to the state-of-the-art ECDSA-AS [3] and reduces online pre-signing time about 60%.

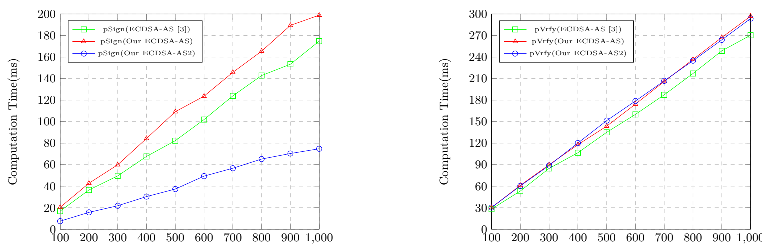

Fig. 1: Efficiency comparison of pre-signing and pre-verification operation

6 Application

6.1 Specific verification scenario

According to the definition of AS [3], the pre-signature should be verified by anyone. However, in some specific verification scenarios, such as the (batched) atomic swaps, AS does not require such a strong property, and the pre-signature is only verified by the participants. We briefly review atomic swaps [11] as follows: two users U_0 and U_1 want to exchange two different cryptocurrencies c_0 and c_1. U_0 (hard relation chooser) chooses a hard relation (I_Y, y) and sends I_Y to U_1; U_0 computes a pre-signature $\hat{\sigma}_0$ for spending c_0 to U_1; U_1 computes a pre-signature $\hat{\sigma}_1$ for spending c_1 to U_0; Both parties check the validity of the pre-signature from each other; Then, U_0 can use the witness y to adapt the full signature σ_1 and gets c_1 by publishing σ_1 on blockchain; U_1 can extract the witness y from ($\hat{\sigma}_1$, σ_1), and then adapts the full signature σ_0 and gets c_0 by publishing σ_0 on blockchain.

As we can see, in above atomic swaps, the pre-signatures do not need to be published on the blockchain, and are only verified by participants U_0 and U_1. ECDSA-AS1/2 can reduce some zero-knowledge proofs in this kind of specific verification scenario. To be specific, the zero-knowledge proof π_{Z_1} of the pre-signing public parameter Z_1 of U_1 can be removed, because U_0 can use the witness y to check the structure of $Z_1 = yQ_1$ without proof π_{Z_1}, but the zero-knowledge proof π_{Z_0} of U_0 cannot be removed. Furthermore, U_0 can help U_1 to compute the pre-signing public parameter $Z_1 = yQ_1$ offline, and U_1 can check $Z_1 = x_1 Y$ without zero-knowledge proof.

6.2 Batched atomic swaps

We develop batched atomic swaps from atomic swaps: U_0 (hard relation chooser) spends c_{0i} (transaction tx_{0i}) to U_i, $i \in [n]$ and U_i spends c_i (transaction tx_i) to U_0. For one hard relation (Y, y), U_0 can compute and send a batch of pre-signatures σ_{0i} for each users U_i. After checking the validity of pre-signatures σ_{0i}, each user U_i can compute the pre-signatures σ_i and sends it to U_0. Then U_0 can check the validity of all pre-signatures [6] and use the witness y to adapt full signatures σ_i and gets c_i by publishing signatures σ_i on blockchain. Lastly, each user can get signatures σ_i and extract the witness y, and adapts the full signature σ_{0i} and gets c_{0i} by publishing signatures σ_{0i} on blockchain.

$U_0((Q_0, x_0), Q_i, tx_{0i}), i \in [1, n]$		$U_i((Q_i, x_i), Q_0, tx_i), i \in [1, n]$
	Offline phase	
$\mathsf{GenR}(1^\lambda) \to (Y, y)$, $\mathsf{P}_Y(Y, y) \to \pi_Y$,		If $Z_i \neq x_i Y$ or $\mathsf{V}_Y(I_Y) \to 0$,
$Z_i = yQ_i$, $\mathsf{P}_Z(I_Z = (G, Y, Q_0, Z_0), y) \to \pi_{Z_0}$	$\xrightarrow{I_Y, Z_0, \pi_{Z_0}, Z_i}$	or $\mathsf{V}_Z((G, Y, Q_0, Z_0), \pi_{Z_0}) \to 0$, Output $\perp$
	Online phase	
$\hat{\sigma_{0i}} \leftarrow \mathsf{pSign}((X_0, x_0), (I_Y, I_Z), tx_{0i})$	$\xrightarrow{\hat{\sigma_{0i}}, tx_{0i}}$	If $\mathsf{pVrfy}(X_0, tx_{0i}, (I_Y, I_Z), \hat{\sigma_{0i}}) \to 0$, Output $\perp$
If $\exists\ i \in [n]$, $\mathsf{pVrfy}(X_i, tx_i, (I_Y, I_Z), \hat{\sigma_i}) \to 0$	$\xleftarrow{\hat{\sigma_i}, tx_i}$	else, $\hat{\sigma_i} \leftarrow \mathsf{pSign}((X_i, x_i), (I_Y, I_Z), tx_i)$
Output $\perp$		
else, $\sigma_i \leftarrow \mathsf{Adapt}(\hat{\sigma_i}, y)$		
Publish σ_i on blockchain	$\xrightarrow{\sigma_i}$	$y \leftarrow \mathsf{Ext}(\sigma_i, \hat{\sigma_i}, I_Y)$
		$\sigma_{0i} \leftarrow \mathsf{Adapt}(\hat{\sigma_{0i}}, y)$
		Publish σ_{0i} on blockchain

Fig. 2: Batched atomic swap protocol based on ECDSA-AS2

Our ECDSA-AS2 is more efficient for using in batched atomic swaps than ECDSA-AS [18,3]. We show the batched atomic swap protocol based on ECDSA-AS2 in fig 2. To be specific, in the offline phase, U_0 computes one pre-signing public parameters $Z_0 = yQ_0$ and one zero-knowledge proof $\pi_{Z_0} \leftarrow \mathsf{P}_Z(G, Q_0, Y, Z_0))$, and helps all other users U_i to compute n pre-signing public parameters $Z_i = yQ_i$ in a batch and offline. U_i can checks $\mathsf{V}_Z(G, Q_0, Y, Z_0), \pi_{Z_0}) \to 1$ and $Z_i = x_i Y$ without zero-knowledge proof π_{Z_i}. In the online phase, U_0 runs n times AS2.pSign and AS2.pVrfy and each U_i runs once AS2.pSign and AS2.pVrfy, where AS2.pSign and AS2.pVrfy is same to original ECDSA signing and verification algorithms except for using the pre-signing public parameter and the instance (Z, Y) as the verification key and base point instead of (Q, G).

[6] U_0 must check all pre-signatures, because any full signature is published on blockchain, the witness y can be extracted, and all coins can be taken.

In [18,3], each user uses the random number k as the witness to generate the pre-signing public parameter $K = kY$ and zero-knowledge proof π_K. For a batch of transactions, U_0 needs to choose n random numbers to generate individual pre-signing public parameter $K_{0i} = k_{0i}Y$ and zero-knowledge proof $\pi_{K_{0i}} \leftarrow \mathsf{P}(G, \hat{K_{0i}} = k_{0i}G, Y, K_{0i})$ for U_i, and U_i uses different random number k_i to compute pre-signing public parameter $K_i = k_iY$ and zero-knowledge proof $\pi_{K_i} \leftarrow \mathsf{P}(G, \hat{K_i} = k_iG, Y, K_i)$. What's more, U_0 needs to check the validity of all n proofs π_{K_i}, and U_i also needs to check the validity of $\pi_{K_{0i}}$.

As depicted in Table 2, we give a comparison of (batched) atomic swaps based on ECDSA-AS [18,3] or ECDSA-AS1/2. Our ECDSA-AS1/2 can move all computations of pre-signing public parameters and zero-knowledge proofs offline. Especially, in ECDSA-AS2, U_0 can generate them in a batch. What's more, since the batched atomic swaps can be seen as specific verification scenarios, batched atomic swaps based on ECDSA-AS [18,3] needs $2n$ pre-signing public parameters and $2n$ zero-knowledge proofs, but batched atomic swaps based on ECDSA-AS2 needs $n + 1$ pre-signing public parameters and just 1 zero-knowledge proof.

Table 2: Compare with ECDSA-AS [18,3] used in (batched) atomic swaps

Schemes	Users	Atomic swaps ($i = 1$)			Batched atomic swaps ($i \in [n]$)		
		Hard relation	Pre-signing		Hard relations	Pre-signing	
			Offline	Online		Offline	Online
ECDSA-AS [18]	U_0	(Y, y)	—	$\hat{\sigma_0} = (K_0, \pi_{K_0}, r_0, \hat{s_0})$	(Y, y)	—	$\hat{\sigma_{0i}} = (K_{0i}, \pi_{K_{0i}}, r_{0i}, \hat{s_{0i}})$
	U_i	—	—	$\hat{\sigma_1} = (K_1, \pi_{K_1}, r_1, \hat{s_1})$	—	—	$\hat{\sigma_i} = (K_i, \pi_{K_i}, r_i, \hat{s_i})$
ECDSA-AS [3]	U_0	(I_Y, y)	—	$\hat{\sigma_0} = (K_0, \pi_{K_0}, r_0, \hat{s_0})$	(I_Y, y)	—	$\hat{\sigma_{0i}} = (K_{0i}, \pi_{K_{0i}}, r_{0i}, \hat{s_{0i}})$
	U_i	—	—	$\hat{\sigma_1} = (K_1, \pi_{K_1}, r_1, \hat{s_1})$	—	—	$\hat{\sigma_i} = (K_i, \pi_{K_i}, r_i, \hat{s_i})$
Our ECDSA-AS1	U_0	(I_Y, y)	$Z_0 = x_0Y, \pi_{Z_0}$	$\hat{\sigma_0} = (r_0, \hat{s_0})$	(I_Y, y)	$Z_0 = x_0Y, \pi_{Z_0}$	$\hat{\sigma_{0i}} = (r_{0i}, \hat{s_{0i}})$
	U_i	—	$Z_i = x_iY$	$\hat{\sigma_1} = (r_1, \hat{s_1})$	—	$Z_i = x_iY$	$\hat{\sigma_i} = (r_i, \hat{s_i})$
Our ECDSA-AS2	U_0	(I_Y, y)	$Z_0 = yQ_0, Z_i = yQ_i, \pi_{Z_0}$	$\hat{\sigma_0} = (r_0, \hat{s_0})$	(I_Y, y)	$Z_0 = yQ_0, Z_i = yQ_i, \pi_{Z_0}$	$\hat{\sigma_{0i}} = (r_{0i}, \hat{s_{0i}})$
	U_i	—	—	$\hat{\sigma_1} = (r_1, \hat{s_1})$	—	—	$\hat{\sigma_i} = (r_i, \hat{s_i})$

7 Conclusions

In this paper, we propose an ECDSA-AS and give the security proof based on ECDSA. And then, we construct two more efficient ECDSA-AS schemes called ECDSA-AS1/2 than the state-of-the-art ECDSA-AS [3], by computing pre-signing public parameters and zero-knowledge proofs in a batch and offline. What's more, we develop batched atomic swaps which can reduce the number of hard relations in a batch compared with independently running atomic swaps. Considering specific verification scenarios, such as (batched) atomic swaps, our protocols based on ECDSA-AS1/2 can reduce the number of pre-signing public parameters and zero-knowledge proofs that further improves the efficiency.

References

1. American National Standards Institute: X9.62: Public key cryptography for the financial services industry: The elliptic curve digital signature algorithm (ecdsa) (2005)

2. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Bitcoin-compatible virtual channels. IACR Cryptol. ePrint Arch. **2020**, 554 (2020), https://eprint.iacr.org/2020/554
3. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized bitcoin-compatible channels. IACR Cryptol. ePrint Arch. **2020**, 476 (2020), https://eprint.iacr.org/2020/476
4. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized channels from limited blockchain scripts and adaptor signatures. In: ASIACRYPT 2021. pp. 635–664. Springer (2021)
5. Bitcoin Wiki: Payment channels (2018), https://en.bitcoin.it/wiki/Paymentchannels
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA. pp. 136–145. IEEE Computer Society (2001). https://doi.org/10.1109/SFCS.2001.959888, https://doi.org/10.1109/SFCS.2001.959888
7. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings. Lecture Notes in Computer Science, vol. 740, pp. 89–105. Springer (1992). https://doi.org/10.1007/3-540-48071-4_7, https://doi.org/10.1007/3-540-48071-4_7
8. Decker, C., Wattenhofer, R.: A fast and scalable payment network with bitcoin duplex micropayment channels. In: Stabilization, Safety, and Security of Distributed Systems - 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings. pp. 3–18. Springer (2015)
9. Deshpande, A., Herlihy, M.: Privacy-preserving cross-chain atomic swaps. In: Financial Cryptography and Data Security - FC 2020 International Workshops, AsiaUSEC, CoDeFi, VOTING, and WTSC, Kota Kinabalu, Malaysia, February 14, 2020, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12063, pp. 540–549. Springer (2020)
10. Eckey, L., Faust, S., Hostáková, K., Roos, S.: Splitting payments locally while routing interdimensionally. IACR Cryptol. ePrint Arch. **2020**, 555 (2020)
11. Esgin, M.F., Ersoy, O., Erkin, Z.: Post-quantum adaptor signatures and payment channel networks. In: Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12309, pp. 378–397. Springer (2020)
12. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. pp. 186–194 (1986). https://doi.org/10.1007/3-540-47721-7_12, https://doi.org/10.1007/3-540-47721-7_12
13. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3621, pp. 152–168. Springer (2005)
14. Gugger, J.: Bitcoin-monero cross-chain atomic swap. IACR Cryptol. ePrint Arch. **2020**, 1126 (2020), https://eprint.iacr.org/2020/1126

15. Lindell, Y.: Fast secure two-party ECDSA signing. In: Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II. pp. 613–644 (2017). https://doi.org/10.1007/978-3-319-63715-0_21, https://doi.org/10.1007/978-3-319-63715-0_21
16. Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M.: Anonymous multi-hop locks for blockchain scalability and interoperability. In: 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019. The Internet Society (2019)
17. Miller, A., Bentov, I., Bakshi, S., Kumaresan, R., McCorry, P.: Sprites and state channels: Payment networks that go faster than lightning. In: Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11598, pp. 508–526. Springer (2019)
18. Moreno-Sanchez, P., Kate, A.: Scriptless scripts with ecdsa. lightning-dev mailing list https://lists.linuxfoundation.org/pipermail/lightning-dev/attachments/20180426/fe978423/attachment-0001.pdf
19. Poelstra, A.: Lightning in scriptless scripts. mimblewimble team mailing list (2017), https://lists.launchpad.net/mimblewimble/msg00086.html
20. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016), https://lightning.network/lightning-network-paper.pdf
21. Schnorr, C.: Efficient identification and signatures for smart cards. In: Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. pp. 239–252 (1989)