An Efficient ECDSA-based Adaptor Signature and Its Two-Party Extension

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 and restricts its functionality extension.

In this paper, we construct efficient ECDSA-based adaptor signature schemes with offline proof and give security proof based on ECDSA. The pre-signing algorithm of our schemes is similar to the ECDSA signing algorithm except for modifying some parameters. Therefore, our schemes can enjoy the same efficiency and functionality extension as ECDSA. Meanwhile, our schemes can completely reuse the implementation of ECDSA and upgrade existing ECDSA application friendly. Furthermore, we define a two-party adaptor signature with interactive key generation and construct a fast two-party ECDSA-based adaptor signature following the structure of Lindell's two-party ECDSA. Last, we conduct an experimental evaluation, demonstrating that the performance of our ECDSA-based adaptor signature is similar to ECDSA and can compare with the state-of-the-art ECDSA-based adaptor signature.

Keywords: Adaptor signature, Two-party adaptor signature, ECDSA, two-party ECDSA

1 Introduction

Adaptor signatures (AS), also known as scriptless scripts, are introduced by Poelstra [1] and recently formalized by Aumayr et al. [2] 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 the pre-signing key, message, and an instance of a hard relation, such that the pre-signature can be adapted into a valid full signature by using the witness of the hard relation. The full signature can be verified in the same way as the original verification algorithm. 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 valid 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 [2,3,4,5,6], payment routing in payment channel networks (PCNs) [7,8,9,10], atomic swaps [11,12,10], and many others.

Poelstra [1] first gives a Schnorr-based AS that is limited to cryptocurrencies using Schnorr signatures [13] and thus is not compatible with those systems, prominently Bitcoin. Then, Moreno-Sanchez and Kate [14] present an ECDSA-based AS and its two-party version, but their schemes are not clear how to prove security. Malavolta et al. [8] present protocols for two-party adaptor signatures based on ECDSA [15]. However, they do not define AS as a stand-alone primitive and formalize the security definition for the threshold primitive and hence security for their schemes has not been analyzed completely. Until Aumayr et al. [2] 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 [16]. They exquisitely modify the hard relation in [14], by adding zero-knowledge proof such that they can extract the witness in the random model [17], namely "self-proving structure". However, their ECDSA-based AS requires another zero-knowledge proof in the pre-signing phase to ensure that the signer works correctly, which leads to low efficiency and restricts the functionality extension compared with original ECDSA [15], such as two-party [18,19,20] or threshold extension [21,22,23,24,25]. Then, Erwig et al. [26] define two-party adaptor signature schemes with aggregatable public keys and give a generic transformation from identification schemes. However, their definition is unsuitable for two-party ECDSA-based AS.

Unfortunately, the constructions of ECDSA-based AS and the formalization of two-party AS given in existing works have two limitations: (i) the state-of-the-art ECDSA-based AS [2] is not entirely satisfactory, such as it is not easy to be extended into two-party, and zero-knowledge proof in the pre-signature phase leads to low efficiency; (ii) missing the formalization of two-party adapter signature with an interactive key generation that is suitable for two-party ECDSA-based AS. Motivated by above discussions, we ask the following challenging question:

Is it possible to define a two-party adaptor signature with interactive key generation and construct an efficient ECDSA-based adaptor signature scheme that is easy to be extended into two-party?

1.1 Our Contributions

In this paper, we affirmatively answer the previous questions. First, we propose an ECDSA-based AS (ECDSA-AS) and use "self-proving structure" [2]

$I_Y = (Y, \pi_Y)$ to prove the security based on positive ECDSA in UC framework [16]. And then, we observe that the zero-knowledge proof used in the pre-signing phase is independent of the message and random number, so we can consider the "offline-proof technique" to improve the efficiency of the online pre-signing phase, and construct two efficient ECDSA-AS schemes. For convenience, we denote our two efficient ECDSA-AS schemes by ECDSA-ASv1 and ECDSA-ASv2. The difference between the two schemes is to use different witnesses to prove similar hard relations. ECDSA-ASv1 uses the pre-signing key x as a witness to prove the hard relation $((G, Q = xG, Y, Z = xY), x)$ satisfies equality of discrete logarithms. ECDSA-ASv2 uses the witness y of hard relation (I_Y, y) as a witness to prove the hard relation $((G, Y = yG, Q, Z = yQ), y)$ satisfies equality of discrete logarithms. With the offline zero-knowledge proof, the online pre-signing algorithm of our ECDSA-ASv1/2 is similar to the original ECDSA signing algorithm except for modifying some parameters by using (Z, Y) as verification key and base point instead of (Q, G) used in ECDSA. Therefore, our schemes can enjoy the same efficiency and functionality extension as ECDSA. Furthermore, we define two-party AS with interactive key generation and construct a fast two-party ECDSA-AS based on ECDSA-ASv2 following the structure of Lindell's two-party ECDSA [18].

Self-proving structure. Intuitively speaking, for hard relation $(Y = yG, y)$ embedded in AS, our schemes also use "self-proving structure" following [2] to ensure provable security. That is, we add an additional proof $\pi_Y \leftarrow \mathsf{P}(Y, y)$ in the hard relation $(I_Y = (Y, \pi_Y), y)$, where $\mathsf{V}(Y, \pi_Y) \rightarrow 1$, P and V denote the proving and verification algorithm in the proof systems. Benefiting from the above modification, upon receiving an instance $I_Y = (Y, \pi_Y)$, there is an extractor that can extract the witness y from π_Y in the random oracle model [2], online extractor as introduced in [17]. Thus, in the security proof, the simulator can extract the witness y and take advantage of the ECDSA signing oracle to simulate the pre-signing oracle.

Offline proof. In the pre-signing algorithm, our ECDSA-AS proves the hard relation $((G, Q, Y, Z), x)$ satisfies equality of discrete logarithms, that is, there exists the witness x that is pre-signing key, such that $Q = xG$ and $Z = xY$. However, in [2], they prove the hard relation $((G, K, Y, \hat{K}), k)$ satisfies equality of discrete logarithms, that is, there exists the witness k that is a random number used in the pre-signing algorithm, such that $K = kG$ and $\hat{K} = kY$. According to the structure $Z = xY = yQ$, our ECDSA-AS can also prove the hard relation $((G, Y, Q, Z), y)$ satisfies equality of discrete logarithms, 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 [2], our zero-knowledge proofs are independent of the message m and the random number k used in the pre-signature, so the proofs can be executed offline. At the same time, for same hard relation $(I_Y = (Y, \pi_Y), y)$, one zero-knowledge proof for $((G, Q, Y, Z), x)$ or $((G, Y, Q, Z), y)$ can be used to pre-signing many messages, but due to using the random number k as the

witness in [2], the size of zero-knowledge proof is linearly related to the number of pre-signatures.

Two-party extension. ECDSA-AS [2] uses the random number k as the witness that should be shared to many parties in constructing two-party or threshold ECDSA-AS, so the zero-knowledge proof of proving the equality of discrete logarithms for $((G, K, Y, \hat{K}), k)$ should be distributed, which may be difficult to achieve.

Because the pre-signing algorithm of our scheme is similar to the original ECDSA signing algorithm, the signer can run ECDSA signing algorithm by using Y as a base point and Z as a verification key instead of G and Q to generate the pre-signature. Our schemes can enjoy the same functionality extension as ECDSA. In particular, in ECDSA-ASv2, the two zero-knowledge proofs of hard relations (I_Y, y) and $((G, Y, Q, Z), y)$ are generated by the hard relation chooser, hence the signer only runs the ECDSA signing algorithm with modified parameter (Y, Z) to generate the pre-signature. Therefore, ECDSA-ASv2 can easily be transformed into two-party [18,19,20] or even threshold version [21,22,23,24,25] following existing ECDSA extension.

Since most efficient two-party ECDSA requires interactive key generation, the definition of two-party AS with aggregatable public keys [26] is unsuitable for two-party ECDSA-AS. Therefore, we define two-party AS with interactive key generation following the definition of two-party signature [18,19] and two-party AS [26], and then transform our ECDSA-ASv2 into a provably secure two-party ECDSA-AS.

Performance. Benefiting from holding the original ECDSA signing structure, our schemes can enjoy the same efficiency and completely reuse the hardware and software implementation of ECDSA except for modifying parameters, so it is friendly to upgrade existing ECDSA application. Then, We recall ECDSA-AS [14,2], and show the theoretical analysis of our ECDSA-AS. Our schemes can achieve the same level of efficiency as ECDSA and are more efficient than the state-of-the-art ECDSA-AS [14,2], in particular, ECDSA-ASv1/2 only computes once point multiplication operation, while ECDSA-AS in [14,2] need four times point multiplication operation in online pre-signing phase. Last, we realize our schemes based on the OpenSSL, and the running times of all algorithms are the level of microseconds. The experimental results show that our proposed schemes are practical.

2 Preliminaries

2.1 Notations

For $n \in \mathbb{N}$, 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 Adaptor Signature Scheme

An adaptor signature scheme [2] 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 $\Pi_{\mathsf{R},\sum} = (\mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Ext})$ defined as:

- $\mathsf{pSign}_{sk}(m, Y) \rightarrow \hat{\sigma}$: on input a pre-signing key sk, an instance Y and a message $m \in \{0,1\}^*$, outputs a pre-signature $\hat{\sigma}$.
- $\mathsf{pVrfy}_{vk}(m, Y, \hat{\sigma}) \rightarrow 0/1$: on input a verification key vk, a pre-signature $\hat{\sigma}$, an instance Y and a message $m \in \{0,1\}^*$, outputs a bit $b \in \{0,1\}$.
- $\mathsf{Adapt}(\hat{\sigma}, y) \rightarrow \sigma$: on input a pre-signature $\hat{\sigma}$ and a witness y, outputs a signature σ.
- $\mathsf{Ext}(\sigma, \hat{\sigma}, Y) \rightarrow y$: 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. Informally, it guarantees that an honestly generated pre-signature w.r.t. an instance $Y \in \mathsf{R}$ is a valid pre-signature and can be completed into a valid signature from which a witness for Y can be extracted.

Definition 1 (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:*

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

We now define the security properties of an AS scheme. We begin with the notion of unforgeability which is similar to the definition of existential unforgeability under chosen message attacks but additionally requires that producing a forgery σ for some message m is hard even given a pre-signature on m w.r.t. an instance $Y \in \mathsf{R}$. Let us emphasize that allowing the adversary to learn a pre-signature on the forgery message m is crucial since for our applications unforgeability needs to hold even in case the adversary learns a pre-signature for m without knowing a corresponding witness for Y. We formally define the existential unforgeability under chosen message attack for AS (aEUF–CMA).

Definition 2 (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:*

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

As discussed above, AS guarantees that a valid pre-signature w.r.t. Y can be completed to a valid signature if and only if the corresponding witness y for Y is known. An additional property that we will require is that any valid pre-signature w.r.t. Y (possibly produced by a malicious signer) can be completed into a valid signature using the witness y with $(Y, y) \in \mathsf{R}$. Notice that this property is stronger than the pre-signature correctness property, since we require that even maliciously produced pre-signatures can always be completed into valid signatures. The next definition formalizes the above discussion.

Definition 3 (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 $\mathsf{pVrfy}_{vk}(m, Y, \hat{\sigma}) \rightarrow 1$, we have $\Pr[\mathsf{Vrfy}_{vk}(m, \mathsf{Adapt}(\hat{\sigma}, y)) \rightarrow 1] = 1$.*

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. The last property that we are interested in is witness extractability. Informally, it 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.

Definition 4 (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 the following holds:*

$$\Pr[aWitExt_{\mathcal{A},\Pi_{\mathsf{R},\sum}}(\lambda) = 1] \leq \mathsf{negl}(\lambda),$$

where the experiment $aWitExt_{\mathcal{A},\Pi_{\mathsf{R},\sum}}$ is defined as follows

Let us stress that while the witness extractability experiment aWitExt looks fairly similar to the experiment aSigForge, there is one crucial difference; namely, the adversary is allowed to choose the forgery instance Y. Hence, we can assume that 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. The adversary wins only if the valid signature does not reveal a witness for Y.

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

2.3 ECDSA

We review the ECDSA scheme [15] $\sum = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ 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.

- $\mathsf{Gen}(pp) \rightarrow (Q, x)$: 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 $(sk = x, vk = Q)$.
- $\mathsf{Sign}_{sk}(m) \rightarrow (r, s)$. The signing algorithm chooses $k \leftarrow \mathbb{Z}_q$ randomly and computes $r = f(kG)$ and $s = k^{-1}(h(m) + rx)$, where $h : \{0,1\}^* \rightarrow \mathbb{Z}_q$ is a hash function modeled as a random oracle and $f : \mathbb{G} \rightarrow \mathbb{Z}_q$ is defined as the projection to the x-coordinate.
- $\mathsf{Vrfy}_{vk}(m, \sigma) \rightarrow 0/1$. 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.

According to the structure of ECDSA, if (r, s) is a valid signature for m, then so is $(r, -s)$. Therefore, $\sum$ does not satisfy SUF-CMA security which we need to prove the security of ECDSA-AS. In order to tackle this problem, we recall the Positive ECDSA scheme which guarantees that if (r, s) is a valid signature, then $|s| \leq (q-1)/2$. The positive ECDSA has already been used in other works such as [18].

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 $\sum = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$. Formally, we define hard relations $\mathsf{R} = \{(I_Y = (Y, \pi_Y), y) | Y = yG \wedge \mathsf{V}(I_Y) = 1\}$ and $\mathsf{R}_Z = \{(I_Z = (G, Q, Y, Z), x) | Q = xG \wedge Z = xY\}$, and denote by P the prover and by V the verifier of the proof system.

- $\mathsf{pSign}_{(vk,sk)}(m, I_Y) \rightarrow \hat{\sigma}$: on input a key-pair $(vk, sk) = (Q, x)$, a message m and an instance $I_Y = (Y, \pi_Y)$, the algorithm computes $Z = xY$, runs $\pi_Z \leftarrow \mathsf{P}(I_Z = (G, Q, Y, Z), x)$, and chooses $k \leftarrow \mathbb{Z}_q$, computes $r = f(kY)$, $\hat{s} = k^{-1}(h(m) + rx) \bmod q$ and outputs $\hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$.

- $\mathsf{pVrfy}_{vk}(m, I_Y, \hat{\sigma}) \to 0/1$: 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 $\mathsf{V}(I_Z) \to 0$, 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 $\perp$.

$\mathsf{pSign}_{(vk,sk)}(m, I_Y)$	$\mathsf{pVrfy}_{vk}(m, I_Y, \hat{\sigma})$
$(vk, sk) = (Q, x), Z = xY$	if $\mathsf{V}(I_Z) \to 0$
$\mathsf{P}(I_Z, x) \to \pi_Z$	outputs $\perp$.
$k \leftarrow \mathbb{Z}_q$	else, $r' = f(\hat{s}^{-1} \cdot h(m) \cdot Y + \hat{s}^{-1} \cdot r \cdot Z))$
$r = f(kY)$	If $r' = r$, output 1
$\hat{s} = k^{-1}(h(m) + rx) \bmod q$	else, output 0.
return $\hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$	return 0/1
$\mathsf{Adapt}(y, \hat{\sigma})$	$\mathsf{Ext}(\sigma, \hat{\sigma}, I_Y) \to y$
$\hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$	$\sigma = (r, s), \hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$
$s = \hat{s} \cdot y^{-1} \bmod q$	$y = \hat{s}/s \bmod q$
return $\sigma = (r, s)$	If $(I_Y, y) \in \mathsf{R}$, it returns y,
	else, it returns $\perp$

Fig. 1: ECDSA-based adaptor signature scheme

Note that in the pre-signing phase, our ECDSA-AS uses the pre-signing key x as the witness to prove the hard relation $((G, Q = xG, Y = yG, Z = xY), x)$ satisfies equality of discrete logarithms, while in [2], they use the random number k as the witness to prove the hard relation $((G, K = kG, Y = yG, \hat{K} = kY), k)$ satisfies equality of discrete logarithms. With this change, the zero-knowledge proof in our scheme can be executed offline, and in the online phase, the pre-signing algorithm is similar to the original ECDSA signing algorithm except for using (Z, Y) as verification key and base point instead of (Q, G). In addition, for the same hard relation (I_Y, y), our scheme only needs one zero-knowledge proof for pre-signing many different messages, but in [2], the zero-knowledge proof is linearly related to the number of pre-signatures.

Theorem 1. *Assuming that the positive ECDSA signature $\sum$ is SUF-CMA secure and R is a hard relation, the ECDSA-based adaptor signature $\Pi_{\mathsf{R},\sum}$ as defined in fig. 1 is secure in random oracle model.*

We prove that our ECDSA-AS scheme in fig. 1 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 $\mathsf{pVrfy}_{vk}(m, I_Y, \hat{\sigma}) \rightarrow 1$. 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 $\mathsf{Adapt}(\hat{\sigma}, y) \rightarrow \sigma$, 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 $\mathsf{Vrfy}_{vk}(m, \sigma) \rightarrow 1$.

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 $\mathsf{pSign}_{(vk,sk)}(m, I_Y) \rightarrow \hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$, 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 $\mathsf{pVrfy}_{vk}(m, I_Y, \hat{\sigma}) \rightarrow 1$. By Lemma 1, this implies that $\mathsf{Vrfy}_{vk}(m, \sigma) \rightarrow 1$, for $\mathsf{Adapt}(\hat{\sigma}, y) \rightarrow \sigma = (r, s)$. 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 and R is a hard relation, the ECDSA-based adaptor signature scheme $\Pi_{\mathsf{R},\sum}$ as defined above is aEUF–CMA secure.*

Proof. We prove the aEUF–CMA security of the ECDSA-AS by reduction to strong unforgeability of positive ECDSA signatures. 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 reduction algorithm $\mathcal{S}$ 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 $\mathcal{H}_{\text{ECDSA}}$, which it uses to simulate oracle queries for $\mathcal{A}$, namely random oracle ($\mathcal{H}$), signing oracle ($\mathcal{O}_{\text{Sign}}$) and pre-signing oracle ($\mathcal{O}_{\text{pSign}}$) queries.

The main challenge is simulating $\mathcal{O}_{\text{pSign}}$ queries since $\mathcal{S}$ can only get full signatures from its ECDSA signing oracle. Hence, $\mathcal{S}$ needs to transform full signatures into pre-signatures for $\mathcal{A}$. In order to do so, $\mathcal{S}$ needs to learn the witness y, for instance, I_Y, and simulate the zero-knowledge proof π_Z which proves $\exists\ x$ such that $Z = xY$ and $Q = xG$. More concretely, upon receiving a $\mathcal{O}_{\text{pSign}}$ query from $\mathcal{A}$ on input a message m and an instance I_Y, the simulator queries its signing oracle to obtain a full signature on m. Further, $\mathcal{S}$ learns the witness y, s.t. $Y = yG$, in order to transform the full signature into a pre-signature for $\mathcal{A}$. We make use of the extractability property of the zero-knowledge proof π_Y, in order to extract y and consequently transform a full signature into a

valid pre-signature. What's more, we use the zero-knowledge property of proving π_Z, which allows for the simulation of the proof for a statement without knowing the corresponding witness.

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, where the adversary $\mathcal{A}$ has to come up with a valid forgery for a message m, while having access to oracles $\mathcal{H}$, $\mathcal{O}_{\text{pSign}}$ and $\mathcal{O}_{\text{Sign}}$.
- Game G_1: This game works exactly 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 witness y results in σ^*. In that case, the game aborts.
- Game G_2: This game only applies changes to the $\mathcal{O}_{\text{pSign}}$ oracle as opposed to the previous game G_1. Namely, during the $\mathcal{O}_{\text{pSign}}$ queries, this game extracts a witness y by executing the algorithm K on inputs the instance I_Y, the proof π and the list of random oracle query $\mathcal{H}$. The game aborts if for the extracted witness y it does not hold that $(I_Y, y) \in \mathsf{R}$.
- Game G_3: This game extends the changes of the previous game G_2 to the $\mathcal{O}_{\text{pSign}}$ oracle by first creating a valid full signature σ by executing the Sign algorithm and then converting σ into a pre-signature using the extracted witness y. Further, the game calculates $Z = yQ = xY$, 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. $\mathcal{S}$ simulates $\mathcal{A}$'s oracle queries as follows:

- Signing oracle queries: Upon $\mathcal{A}$ querying the oracle $\mathcal{O}_{Sign}$ 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 the oracle $\mathcal{H}$ 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:
  1) Upon $\mathcal{A}$ querying the oracle $\mathcal{O}_{\text{pSign}}$ on input (m, I_Y), the simulator extracts y using the extractability of NIZK_{I_Y}, forwards m to oracle $\mathcal{O}_{\text{ECDSA-sign}}$ and parses the signature as (r, s).
  2) $\mathcal{S}$ generates a pre-signature from (r, s) by computing $\hat{s} = s \cdot y$.
  3) $\mathcal{S}$ computes $Z = yQ = xY$ and simulates a zero-knowledge proof π_S, proving that $Q = xG$ and $Z = xY$ satisfy equality of discrete logarithms. The simulator outputs $(r, \hat{s}, Z, \pi_S)$.

  In the challenge phase:

1) Upon $\mathcal{A}$ outputting the message m^* as the challenge message, $\mathcal{S}$ generates $(I_Y, y) \leftarrow \mathsf{GenR}(1^\lambda)$, forwards m^* to the oracle $\mathcal{O}_{\text{ECDSA-sign}}$ and parses the signature as (r, s).
2) The simulator generates the required pre-signature $\hat{\sigma}$ in the same way as during $\mathcal{O}_{\text{pSign}}$ queries.
3) Upon $\mathcal{A}$ outputting a forgery σ^*, the simulator outputs (m^*, σ^*) as its own forgery.

We emphasize that the main difference between the simulation and G_4 are syntactical, namely, instead of generating the public and secret keys and running the algorithm Sign and the random oracle $\mathcal{H}$, the simulator $\mathcal{S}$ uses its oracles $\mathcal{O}_{\text{ECDSA-Sign}}$ and $\mathcal{H}$. It remains to show that the forgery output by $\mathcal{A}$ can be used by the simulator to win a 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. More concretely, we construct a simulator $\mathcal{S}$ who can break the hardness of the hard relation assuming he has access to an adversary $\mathcal{A}$ that causes G_1 to abort with non-negligible probability. The simulator gets a challenge I_Y^*, upon which it generates a key pair $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ in order to simulate $\mathcal{A}$'s queries to the oracles $\mathcal{H}$, $\mathcal{O}_{\text{Sign}}$ and $\mathcal{O}_{\text{pSign}}$. This simulation of the oracles works as described in G_1. Eventually, 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 σ^* to $\mathcal{A}$ who outputs a forgery σ.

Assuming that Bad_1 happened (i.e. $\mathsf{Adapt}(\hat{\sigma}, y) = \sigma$), we know that due to the correctness property, the simulator can extract $y^* \leftarrow \mathsf{Ext}(\sigma, \sigma^*, I_Y^*)$ to obtain a valid statement/witness pair for the relation R, i.e.$(I_Y^*, y^*) \in \mathsf{R}$. First, we note that the view of $\mathcal{A}$ is indistinguishable from his view in G_1, 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. By our assumption, this is non-negligible which is the contradiction with the hardness of hard relation R.

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

Claim 2 *Let Bad_2 be the event that G_2 aborts during an $\mathcal{O}_{pSign}$ execution, then it holds that* $\Pr[Bad_2] \leq \mathsf{negl}_2(\lambda)$.

Proof. According to the online extractor property of the zero-knowledge proof, for a witness y extracted from a proof π of instance I_Y such that $\mathsf{V}(I_Y, \pi) \rightarrow 1$, it holds that $(I_Y, y) \in \mathsf{R}$ except with negligible probability in the security parameter. Since games G_2 and G_1 are equivalent except if event Bad_2 occurs, it holds that $|\Pr[G_2 = 1] - \Pr[G_1 = 1]| \leq \mathsf{negl}_2(\lambda)$.

Claim 3 *G_3 is computationally indistinguishable from the previous game G_2.*

Proof. This game extends the changes of the previous game to the $\mathcal{O}_{\text{pSign}}$ oracle by first creating a valid full signature σ by executing the Sign algorithm and

then converting σ into a pre-signature using the extracted witness y. Further, the game calculates $Z = yQ = xY = xyG$ and simulates a zero-knowledge proof π_S.

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

Claim 4 *G_4 is computationally indistinguishable from the previous game G_3.*

Proof. This proof follows above proof. Hence, G_4 is indistinguishable from G_3 and it holds that $\Pr[G_4 = 1] \leq \Pr[G_3 = 1] + \mathsf{negl}_4(\lambda)$.

Claim 5 *(m^*, σ^*) constitutes a valid forgery in positive ECDSA strongSigForge game.*

Proof. In order to prove this claim, we have to show that the tuple (m^*, σ^*) has not been output by the oracle $\mathcal{O}_{\text{ECDSA-Sign}}$ before. Note that the adversary $\mathcal{A}$ 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 ECDSA scheme is SUF–CMA-secure and R is a hard relation, the ECDSA-based adaptor signature scheme $\Pi_{\mathsf{R},\sum}$ as defined is witness extractable.*

Proof. Our proof is to reduce the witness extractability of $\Pi_{\mathsf{R},\sum}$ to the strong unforgeability of the positive ECDSA. In other words, assuming that there exists a PPT adversary $\mathcal{A}$ who wins the aWitExt experiment, we can construct a PPT reduction algorithm $\mathcal{S}$ that wins positive ECDSA strongSigForge experiment.

During the reduction, the main challenge is to simulate a pre-signing oracle. Unlike in the aSigForge experiment, in aWitExt experiment, $\mathcal{A}$ outputs the instance I_Y for relation R alongside the challenge message m^*, meaning that $\mathcal{S}$ does not choose the pair (I_Y, y). Fortunately, it is possible to extract y from the zero-knowledge proof embedded in I_Y. After extracting y, the same approach used in order to simulate the pre-signing oracle queries can be taken here as well.

We prove security by first describing a sequence of games $G_0, \cdots, G_4$, where G_0 is the original aWitExt 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 aWitExt game, where the adversary $\mathcal{A}$ has to come up with a valid signature σ for a message m, a given pre-signature $\hat{\sigma}$ and a given statement/witness pair (I_Y, y), while having access to oracles $\mathcal{H}$, $\mathcal{O}_{\text{pSign}}$ and $\mathcal{O}_{\text{Sign}}$, such that $(I_Y, \mathsf{Ext}(\sigma, \hat{\sigma}, I_Y)) \notin \mathsf{R}$.
- Game G_1: This game only applies changes to the $\mathcal{O}_{\text{pSign}}$ oracle as opposed to the previous game G_0. Namely, during the $\mathcal{O}_{\text{pSign}}$ queries, this game extracts a witness y by executing the algorithm K on inputs the instance I_Y, the proof π_Y and the list of random oracle query $\mathcal{H}$. The game aborts if for the extracted witness y it does not hold that $(I_Y, y) \in \mathsf{R}$.
- Game G_2: This game extends the changes to $\mathcal{O}_{\text{pSign}}$ from the previous game G_1. In the $\mathcal{O}_{\text{pSign}}$ execution, this game first creates a valid full signature σ by executing the Sign algorithm and converts σ into a pre-signature using the extracted witness y. Further, $\mathcal{S}$ calculates $Z = yQ$ and simulates a zero-knowledge proof π_S.
- Game G_3: In this game, we apply the same changes made in the game G_1 in oracle $\mathcal{O}_{\text{pSign}}$ to the challenge phase of the game. During the challenge phase, this game extracts a witness y by executing the algorithm K on inputs the instance I_Y, the proof π, and the list of random oracle queries $\mathcal{H}$. The game aborts if for the extracted witness y it does not hold that $(I_Y, y) \in \mathsf{R}$.
- Game G_4: In this game, we apply the same changes made in the game G_2 in oracle $\mathcal{O}_{\text{pSign}}$ to the challenge phase of the game. In the challenge phase, this game first creates a valid full signature σ by executing the Sign algorithm and converts σ into a pre-signature using the extracted witness y. Further, $\mathcal{S}$ calculates $Z = yQ$ and simulates a zero-knowledge proof π_S.

Having shown that the transition from the original aWitExt game (Game G_0) to Game G_4 is indistinguishable, it remains to show that there exists a simulator that perfectly simulates G_4 and uses $\mathcal{A}$ to win positive ECDSA strongSigForge game. $\mathcal{S}$ simulates $\mathcal{A}$'s oracle queries as follows:

- Signing oracle queries: Upon $\mathcal{A}$ querying the oracle $\mathcal{O}_{\text{Sign}}$ 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 the oracle $\mathcal{H}$ on input x, if $\mathcal{H}[x] = \perp$, then $\mathcal{S}$ queries $\mathcal{H}_{\text{ECDSA}}(x)$, otherwise the simulator returns $\mathcal{H}[x]$.
- Pre-signing oracle queries:
  1) Upon $\mathcal{A}$ querying the oracle $\mathcal{O}_{\text{pSign}}$ on input (m, I_Y), the simulator extracts y using the extractability of NIZK_{I_Y}, forwards m to oracle $\mathcal{O}_{\text{ECDSA-Sign}}$ and parses the signature as (r, s).
  2) $\mathcal{S}$ generates a pre-signature from (r, s) by computing $\hat{s} = s \cdot y$.
  3) $\mathcal{S}$ computes $Z = yQ = xY$ and simulates a zero-knowledge proof π_S, proving that $Q = xG$ and $Z = xY$ satisfy equality of discrete logarithms. The simulator outputs $(r, \hat{s}, Z, \pi_S)$.

In the challenge phase:

1) Upon $\mathcal{A}$ outputting the message (m^*, I_Y) as the challenge message, $\mathcal{S}$ extracts y using the extractability of NIZK_{I_Y}, forwards m^* to the oracle $\mathcal{O}_{\text{ECDSA-Sign}}$ and parses the signature as (r, s).
2) The simulator generates the required pre-signature $\hat{\sigma}$ in the same way as during $\mathcal{O}_{\text{pSign}}$ queries.
3) Upon $\mathcal{A}$ outputting a forgery σ, the simulator outputs (m^*, σ^*) as its own forgery.

We emphasize that the main difference between the simulation and G_4 are syntactical, namely, instead of generating the public and secret keys and running the algorithm Sign and the random oracle $\mathcal{H}$, the simulator $\mathcal{S}$ uses its oracles $\mathcal{O}_{\text{ECDSA-Sign}}$ and $\mathcal{H}_{\text{ECDSA}}$. It remains to show that the signature output by $\mathcal{A}$ can be used by the simulator to win a positive ECDSA strongSigForge game.

Claim 6 *Let Bad_1 be the event that G_1 aborts during an $\mathcal{O}_{pSign}$ execution, then it holds that* $\Pr[Bad_1] \leq \mathsf{negl}_1(\lambda)$.

Proof. According to the online extractor property of the zero-knowledge proof, for a witness y extracted from a proof π for instance I_Y such that $\mathsf{V}(I_Y) = 1$, it holds that $(I_Y, y) \in \mathsf{R}$ except with negligible probability. Since games G_1 and G_0 are equivalent except if event Bad_1 occurs, it holds that $|\Pr[G_0 = 1] - \Pr[G_1 = 1]| \leq \mathsf{negl}_1(\lambda)$.

Claim 7 *Let Bad_2 be the event that G_3 aborts during the challenge phase, then it holds that* $\Pr[Bad_2] \leq \mathsf{negl}_3(\lambda)$.

Proof. This proof is analogous to above proof. Since games G_2 and G_3 are equivalent except if event Bad_2 occurs, it holds that $|\Pr[G_2 = 1] - \Pr[G_3 = 1]| \leq \mathsf{negl}_3(\lambda)$.

Claim 8 *G_2 is computationally indistinguishable from the previous game G_1.*

Proof. This game extends the changes to $\mathcal{O}_{\text{pSign}}$ from the previous game. In the $\mathcal{O}_{\text{pSign}}$ execution, this game first creates a valid full signature σ by executing the Sign algorithm and converts σ into a pre-signature using the extracted witness y. Further, the game calculates $Z = yQ = xY$ and simulates a zero-knowledge proof π_S. Due to the zero-knowledge property of the zero-knowledge proof, the simulator can produce a proof $\pi_S \leftarrow \mathsf{S}((G, Q, Y, Z), 1)$ which is indistinguishable from a proof $\pi_Z \leftarrow \mathsf{P}((G, Q, Y, Z), x)$. Hence, G_2 is computationally indistinguishable from the previous game G_1. It holds that $\Pr[G_1 = 1] - \Pr[G_2 = 1] \leq \mathsf{negl}_2(\lambda)$.

Claim 9 *G_4 is computationally indistinguishable from the previous game G_3.*

Proof. This proof is analogous to above proof. G_4 is computationally indistinguishable from the previous game G_3, that is, it holds that $|\Pr[G_4 = 1] - \Pr[G_3 = 1]| \leq \mathsf{negl}_4(\lambda)$.

Claim 10 (m^*, σ^*) *constitutes a valid forgery in positive ECDSA strongSigForge game.*

Proof. In order to prove this claim, we have to show that the tuple (m^*, σ^*) has not been output by the oracle $\mathcal{O}_{\text{ECDSA-Sign}}$ before. Note that the adversary A has not previously made a query on the challenge message m^* to either $\mathcal{O}_{\text{pSign}}$ or $\mathcal{O}_{\text{Sign}}$. Hence, $\mathcal{O}_{\text{ECDSA-Sign}}$ is only queried on m^* during the challenge phase. If the adversary outputs a forgery σ^* which is equal to the signature σ output by $\mathcal{O}_{\text{ECDSA-Sign}}$ during the challenge phase, the extracted y would be in relation with the given public value I_Y. 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:

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

4 Fast ECDSA-based Adaptor Signature with Offline Zero-Knowledge Proof

In this section, according to the structure $Z = xY = yQ$, we can use the "offline proof technique" to construct two fast ECDSA-AS schemes called ECDSA-ASv1/2. For the structure $Z = xY$ in ECDSA-ASv1, the prover only proves that there exists witness x, such that $Q = xG$ and $Z = xY$ satisfy equality of discrete logarithms. For the structure $Z = yQ$ in ECDSA-ASv2, the prover only proves that there exists witness y, such that $Y = yG$ and $Z = yQ$ satisfy equality of discrete logarithms.

In our ECDSA-AS, the witness of hard relation $\mathsf{R}_Z = ((G, Q, Y, Z), x)$ is the pre-signing key x, and the zero-knowledge proof is $\pi_Z \leftarrow \mathsf{P}(I_Z = (G, Q, Y, Z), x)$, so the signer can compute the zero-knowledge proof offline before getting the message. Therefore, ECDSA-ASv1 can be designed from our ECDSA-AS directly with offline zero-knowledge proof. Refer to the section 3 for specific construction which is ignored here.

Because of the structure $Z = yQ$, by using y as the witness of hard relation $\mathsf{R}_Z = ((G, Y, Q, Z), y)$, and proving that there exists the witness y, such that $Y = yG$ and $Z = yQ$, that is, the zero-knowledge proof is $\pi_Z \leftarrow \mathsf{P}((G, Y, Q, Z), y)$, we can construct efficient ECDSA-ASv2 as follows. Formally, we define hard relations $\mathsf{R} = \{(I_Z = (G, Y, Q, Z, \pi_Y, \pi_Z), y) | Y = yG \wedge Z = yQ \wedge \mathsf{V}(Y, \pi_Y) = 1 \wedge \mathsf{V}((G, Y, Q, Z), \pi_Z) = 1\}$.

- $\mathsf{pSign}_{(vk,sk)}(m, I_Z) \rightarrow \hat{\sigma}$: on input a key-pair $(vk, sk) = (Q, x)$, a message m and an instance $I_Z = (G, Y, Q, Z, \pi_Y)$, the algorithm runs $\mathsf{V}(I_Z) \rightarrow 0$,

outputs $\perp$, otherwise, chooses $k \leftarrow \mathbb{Z}_q$, computes $r = f(kY)$, $\hat{s} = k^{-1}(h(m) + rx) \bmod q$ and outputs $\hat{\sigma} = (r, \hat{s})$.
- $\mathsf{pVrfy}_{vk}(m, I_Z, \hat{\sigma}) \rightarrow 0/1$: on input the verification key $vk = Q$, a message m, an instance I_Z, and a pre-signature value $\hat{\sigma}$, the algorithm outputs $\perp$ if $\mathsf{V}(I_Z) \rightarrow 0$, 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.
- $\mathsf{Adapt}(y, \hat{\sigma}) \rightarrow \sigma$: 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 $\sigma = (r, s)$.
- $\mathsf{Ext}(\sigma, \hat{\sigma}, I_Z) \rightarrow y$: 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 $\perp$.

$\mathsf{pSign}_{(vk,sk)}(m, I_Z)$	$\mathsf{pVrfy}_{vk}(m, I_Z, \hat{\sigma})$
if $\mathsf{V}(I_Z) \rightarrow 0$	if $\mathsf{V}(I_Z) \rightarrow 0$
output $\perp$.	outputs $\perp$.
else, $k \leftarrow \mathbb{Z}_q$	else, $r' = f(\hat{s}^{-1} \cdot h(m) \cdot Y + \hat{s}^{-1} \cdot r \cdot Z))$
$r = f(kY)$	If $r' = r$, output 1
$\hat{s} = k^{-1}(h(m) + rx) \bmod q$	else, output 0.
return $\hat{\sigma} = (r, \hat{s})$	return $0/1$

$\mathsf{Adapt}(y, \hat{\sigma})$	$\mathsf{Ext}(\sigma, \hat{\sigma}, I_Z) \rightarrow y$
$\hat{\sigma} = (r, \hat{s})$	$\sigma = (r, s), \hat{\sigma} = (r, \hat{s})$
$s = \hat{s} \cdot y^{-1} \bmod q$	$y = \hat{s}/s \bmod q$
return $\sigma = (r, s)$	If $(I_Z, y) \in \mathsf{R}$, it returns y,
	else, it returns $\perp$

Fig. 2: ECDSA-based adaptor signature with offline proof

Note that the construction of ECDSA-ASv1 is similar to our ECDSA-AS, except that the prover computes $\pi_Z \leftarrow \mathsf{P}(I_Z = (G, Q, Y, Z), x)$ offline. In the same way, ECDSA-ASv2 is similar to ECDSA-ASv2 except that the prover computes $\pi_Z \leftarrow \mathsf{P}(I_Z = (G, Y, Q, Z), y)$ offline.

Correctness. Following the lemma 1 and lemma 2, our ECDSA-ASv1/2 schemes also satisfy pre-signature adaptability and pre-signature correctness.

Security. Our ECDSA-ASv1/2 schemes embed the hard relation $(I_Y = (Y, \pi_Y), y)$ which satisfies the "self-proving structure" [2]. Therefore, in the security proof, the simulator can get the witness y and simulate the pre-signing oracle. Following the lemma 3 and lemma 4, our ECDSA-ASv1/2 schemes also satisfy aEUF–CMA security and witness extractability.

Comparison. In [2], the prover computes proof $\pi_Z \leftarrow \mathsf{P}(I_Z = (G, K, Y, \hat{K}), k)$ with the witness k which is the random number used in pre-signature. In ECDSA-ASv1, the prover computes proof $\pi_Z \leftarrow \mathsf{P}(I_Z = (G, Q, Y, Z), x)$ with the witness

x which is the pre-signing key. In ECDSA-ASv2, the prover computes proof $\pi_Z \leftarrow \mathsf{P}(I_Z = (G, Y, Q, Z), y)$ with the witness y which is also the witness of the hard relation $(I_Y = (Y, \Pi_Y), y)$.

As we can see, both zero-knowledge proofs of ECDSA-ASv1/2 are independent of the message and random number used in the pre-signing algorithm, so this part can be run offline to improve the efficiency of the online pre-signing phase. At the same time, for embedding the same hard relation $(I_Y = (Y, \pi_Y), y)$, one zero-knowledge proof for $((G, Q, Y, Z), x)$ or $((G, Y, Q, Z), y)$ can be used to pre-signing many messages, but due to using the random number k as the witness in [2], each pre-signature needs new random number, the size of zero-knowledge proof is linearly related to the number of pre-signatures.

What's more, the structure of pre-signature in ECDSA-ASv1/2 is similar to the original ECDSA except that the signer uses $Y = yG$ and $Z = yQ$ instead of G and $Q = xG$. Therefore, ECDSA-ASv1/2 can reuse the hardware and software implementation of the ECDSA signing algorithm.

Furthermore, in ECDSA-ASv1, the witness is pre-signing key x, so the prover must be the signer like [2], in which the witness is random number k. In ECDSA-ASv2, the witness is y, so the prover can be the hard relation chooser who chooses the hard relation $(I_Y = (Y, \pi_Y), y)$. In order to construct two-party or threshold ECDSA-AS, many parties need to share the pre-signing key x and the random number k, so the zero-knowledge proof of proving the equality of discrete logarithms for $((G, Q, Y, Z), x)$ or $((G, K, Y, \hat{K}), k)$ should be distributed, which may be difficult to achieve. However, in ECDSA-ASv2, the witness of $((G, Y, Q, Z), y)$ is y, which need not be shared, so zero-knowledge proof can be computed easily. Therefore, our ECDSA-ASv2 is easy to extend to distributed scenarios, following the extensions of existing ECDSA, such as two-party and threshold mode, etc.

5 Two-Party Adaptor Signature with Interactive Key Generation

We start with defining a two-party AS scheme with interactive key generation w.r.t. a hard relation R and two-party signature scheme $\sum_2 = (\mathsf{IGen}, \mathsf{ISign}, \mathsf{Vrfy})$. Our definition is inspired by the definitions from prior works [18,19,26].

A two-party AS scheme with interactive key generation is a tuple of protocols and algorithms $\Pi_{\mathsf{R},\sum_2} = (\mathsf{IpSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Ext})$, formally defined as:

- $\mathsf{IpSign}_{\langle sk_0, sk_1 \rangle}(m, Y) \rightarrow \hat{\sigma}$. The interactive pre-signing protocol takes in the common message $m \in \{0,1\}^*$, instance Y and pre-signing key sk_0, sk_1 from each party P_i, $i = 0, 1$, and outputs pre-signature $\hat{\sigma}$.
- $\mathsf{pVrfy}, \mathsf{Adapt}$ and Ext are non-interactive and identical to that of adaptor signature 2.2.

Note that the two-party pre-signature correctness and two-party pre-signature adaptability are similar to that of AS 2.2 except for using IGen and IpSign instead

of Gen and pSign. They are also similar to that of two-party AS [26] except for using interactive key generation instead of key generation and key aggregation.

The unforgeability security definition is similar to Def. 8 [18], except that the adversary can interact with additional oracle Π^b_{IpSign} in order to generate pre-signatures, as in Def. 2. More precisely, in the $\text{aSigForge}^b_{\mathcal{A},\Pi_{\mathsf{R},\sum_2}}(\lambda)$ experiment defined below, $\mathcal{A}$ obtains access to interactive, stateful key generation, signing and pre-signing oracles Π^b_{IKG}, $\Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}$, $\Pi^b_{\text{IpSign}_{\langle sk_{1-b},\cdot\rangle}}$ respectively. Above oracles simulate the honest party P_{1-b}. Note that the key generation oracle Π^b_{IKG} and the signing oracle $\Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}$ is identical to that of Def. 8. The key generation oracle, signing oracle and pre-signing oracle initialize difference machines M, M_{sid} and $M_{(pS,sid)}$, and both machines M_{sid} and $M_{(pS,sid)}$ have the state of M.

In the pre-signing phase, the interactive pre-signing oralce $\Pi^b_{\text{IpSign}_{\langle sk_{1-b},\cdot\rangle}}(\cdot,\cdot,\cdot)$ works as follows:

- Upon receiving (sid, m, Y) and this is the first oracle query with this identifier sid, the oracle initializes a new machine $M_{(pS,sid)}$ with the key share and any state stored by M at the end of the key generation phase. The $M_{(pS,sid)}$ executes as the party P_{1-b} with session identifier sid and input message m and instance Y to be signed.
- Upon receiving (sid, m, Y) and this is not the first oracle query with this identifier sid, the oracle hands $M_{(pS,sid)}$ the incoming message m and instance Y, and returns the next message sent by $M_{(pS,sid)}$. If $M_{(pS,sid)}$ concludes, then the output obtained by $M_{(pS,sid)}$ is returned.

Definition 5 (2-aEUF-CMA security) *The two-party adaptor signature $\Pi_{\mathsf{R},\sum_2}$ is unforgeable if for every PPT adversary $\mathcal{A}$ there exists a negligible function* negl *for $b \in \{0,1\}$, such that:* $\Pr[aSigForge^b_{\mathcal{A},\Pi_{\mathsf{R},\sum_2}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where the experiment $aSigForge^b_{\mathcal{A},\Pi_{\mathsf{R},\sum_2}}$ is defined as follows:*

$\text{aSigForge}^b_{\mathcal{A},\Pi_{\mathsf{R},\sum_2}}(\lambda)$	$\Pi^b_{\text{IKG}}(\cdot,\cdot)$
$(vk, sk_b) \leftarrow \mathcal{A}^{\Pi^b_{\text{IKG}}}(1^\lambda)$	$((vk, sk_b); (vk, sk_{1-b})) \leftarrow \Pi^b_{\text{IKG}}(\cdot,\cdot)$
$m^* \leftarrow \mathcal{A}^{\Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}, \Pi^b_{\text{IpSign}_{\langle sk_{1-b},\cdot\rangle}}}(vk, sk_b)$	
$(Y, y) \leftarrow \mathsf{GenR}(1^\lambda)$	$\Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m)$
$\hat{\sigma^*} \leftarrow \Pi^{\mathcal{A}}_{\text{IpSign}_{\langle sk_{1-b},\cdot\rangle}}(m^*, Y)$	$\sigma \leftarrow \Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m)$
$\sigma^* \leftarrow \mathcal{A}^{\Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}, \Pi^b_{\text{IpSign}_{\langle sk_{1-b},\cdot\rangle}}}(\hat{\sigma^*}, Y)$	$\mathcal{Q} = \mathcal{Q} \cup \{m\}$
return $m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{vk}(m^*, \sigma^*) = 1$	
	$\Pi^b_{\text{IpSign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m, Y)$
	$\hat{\sigma} \leftarrow \Pi^b_{\text{IpSign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m, Y)$
	$\mathcal{Q} = \mathcal{Q} \cup \{m\}$

Definition 6 (two-party witness extractability.) *The two-party adaptor signature $\Pi_{\mathsf{R},\sum_2}$ is witness extractable if for every PPT adversary $\mathcal{A}$ there exists a negligible function* negl *for $b \in \{0,1\}$, such that:* $\Pr[aWitExt^b_{\mathcal{A},\Pi_{\mathsf{R},\sum_2}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where the experiment $aWitExt^b_{\mathcal{A},\Pi_{\mathsf{R},\sum_2}}$ is defined as follows:*

$\mathrm{aWitExt}^b_{\mathcal{A},\Pi_{\mathsf{R},\sum_2}}(\lambda)$	$\Pi^b_{\mathrm{IKG}}(\cdot,\cdot)$
$(vk, sk_b) \leftarrow \mathcal{A}^{\Pi^b_{\mathrm{IKG}}}(1^\lambda)$	$((vk, sk_b); (vk, sk_{1-b})) \leftarrow \Pi^b_{\mathrm{IKG}}(\cdot,\cdot)$
$(m^*, Y^*) \leftarrow \mathcal{A}^{\Pi^b_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}}, \Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}}(vk, sk_b)$	
$\hat{\sigma^*} \leftarrow \Pi^{\mathcal{A}}_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}(m^*, Y)$	$\Pi^b_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m)$
$\sigma^* \leftarrow \mathcal{A}^{\Pi^b_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}}, \Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}}(\hat{\sigma^*}, Y)$	$\sigma \leftarrow \Pi^b_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m)$
$y' = \mathsf{Ext}(\sigma^*, \hat{\sigma^*}, Y^*)$	$\mathcal{Q} = \mathcal{Q} \cup \{m\}$
return $m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{vk}(m^*, \sigma^*) = 1 \wedge (Y^*, y') \notin \mathsf{R}$	$\Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m, Y)$
	$\hat{\sigma} \leftarrow \Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m, Y)$
	$\mathcal{Q} = \mathcal{Q} \cup \{m\}$

Note that the only difference between $\mathrm{aWitExt}^b_{\mathcal{A},\Pi_{\mathsf{R},\sum_2}}$ and $\mathrm{aSigForge}^b_{\mathcal{A},\Pi_{\mathsf{R},\sum_2}}$ is that here the adversary is allowed to choose the statement/witness pair (Y^*, y^*) and that the winning condition additionally requires that for the extracted witness $y' \leftarrow \mathsf{Ext}(\sigma^*, \sigma, Y^*)$ it holds that $(Y^*, y') \notin \mathsf{R}$.

A two-party AS scheme with interactive key generation is secure if it satisfies the properties 2-aEUF–CMA security, two-party pre-signature adaptability, and two-party witness extractability.

Compare with [18,26]. Compare with the unforgeability security definition of two-party signature [18], our two-party AS requires additional interactive pre-signing oracle where any PPT adversary $\mathcal{A}$ who corrupts one party $P_b, b \in \{0,1\}$ interacts with another party P_{1-b} to generate the pre-signature of the message and instance I_Y chosen by $\mathcal{A}$.

Compare with the security definition of two-party AS [26], our two-party AS requires additional interactive key generation oracle where $\mathcal{A}$ who corrupts one party $P_b, b \in \{0,1\}$ interacts with another party P_{1-b} to generate the verification key and pre-signing key. In [26], their scheme can aggregate public keys and avoid the interaction in the key generation phase, but their security definition is unsuitable for two-party ECDSA-AS, because of most efficient two-party ECDSA requires interactive key generation.

6 Two-Party ECDSA-based Adaptor Signature

In this section, we propose a fast two-party ECDSA-AS scheme based on ECDSA-ASv2. As we can see, the pre-signing algorithm of ECDSA-ASv2 is identical to original ECDSA signing algorithm, except for using $(Z = yQ, Y = yG)$ as the

verification key and base point instead of (Q, G) used in ECDSA. Therefore, our ECDSA-ASv2 can be easily transformed into two-party protocol following the structure of two-party ECDSA in [18].

We show the two-party ECDSA-ASv2 in figure 3. The protocol is similar to Lindell's two-party ECDSA except for using the base point Y instead of G in the signing phase, and using the verification key Z instead of Q in the verification phase. Our protocol is presented in the $\mathcal{F}_{\text{zk}}$ and $\mathcal{F}_{\text{com-zk}}$ hybrid model following [18]. For convenience, we review some hard relations in Supplementary Material C and some functionalities including $\mathcal{F}_{\text{ECDSA}}$, $\mathcal{F}_{\text{com}}$, $\mathcal{F}_{\text{zk}}$, and $\mathcal{F}^{\mathsf{R}}_{\text{com-zk}}$ in Supplementary Material D.

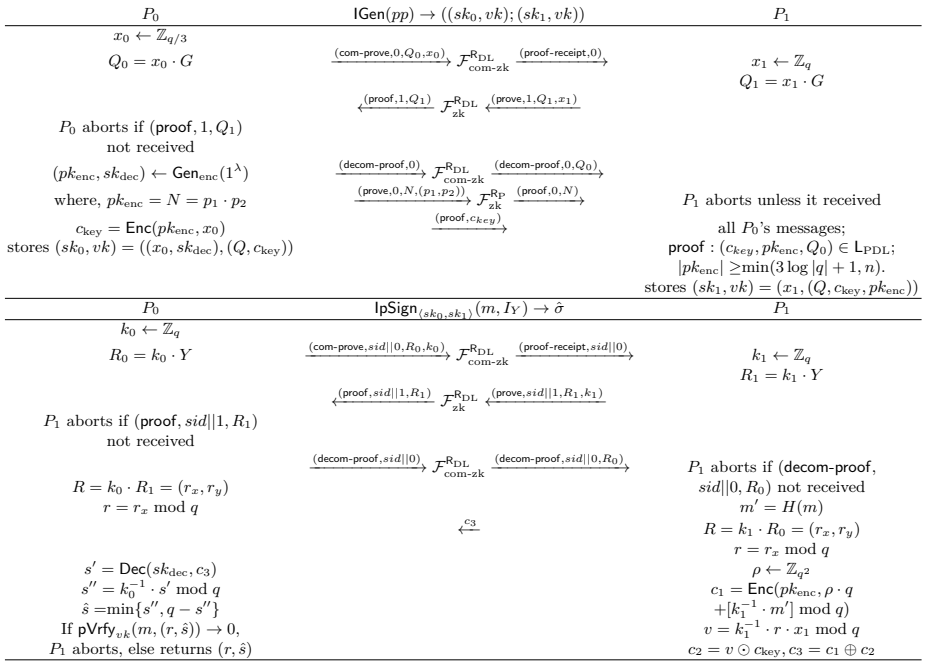

Fig. 3: Two-party ECDSA-ASv2 with interactive key generation

Theorem 2. *The two-party ECDSA-ASv2 in figure 3 is secure, if positive ECDSA is SUF-CMA secure and two-party ECDSA in [18] is secure and* R *is a hard relation.*

Following the lemma 1 and lemma 2, our two-party ECDSA-ASv2 scheme satisfies pre-signature adaptability and pre-signature correctness. Then, we prove two-party ECDSA-ASv2 satisfies 2-aEUF–CMA security and two-party witness extractability following the proof of lemma 3, 4 and the proof of ([18], Theorem

5.3). Before this, we show the interactive key generation oracle Π^b_{IKG}, the interactive signing oracle $\Pi^b_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}}$ and the pre-signing oracle $\Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}$ can be simulated.

The interactive key generation protocol Π^b_{IKG} and interactive signing protocol $\Pi^b_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}}$ are identical to that of two-party ECDSA in [18]. Intuitively speaking, in the key generation and signing protocol, both parties first run a type of "simulatable coin-tossing" in order to generate a random group element Q or R. Besides, P_0 use zero-knowledge proof for the language $\mathsf{L}_{\mathrm{PDL}}$ to ensure generation c_{key} correctly, and define Paillier-EC assumption to bypass decryption to check the correction of c_3. This is fully simulatable due to the extractability and equivocality of the proof and commitment [18]. Therefore, we can construct a simulator $\mathcal{S}$ just like ([18], Theorem 5.3), who can simulate honest P_{1-b} in the interactive key generation and interactive signing phase, when $P_b, b \in \{0,1\}$ is corrupted.

Claim 11 *The interactive key generation oracle Π^b_{IKG} and the interactive signing oracle $\Pi^b_{ISign_{\langle sk_{1-b},\cdot\rangle}}$ for $b \in \{0,1\}$ can be simulated.*

Proof. The simulator $\mathcal{S}$ can be constructed and the proof is following ([18], Theorem 5.3).

The difference between two-party ECDSA-ASv2 and two-party ECDSA [18] is that here the adversary is allowed to query additional pre-signing oracle $\Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}$. Benefiting from the pre-signing structure is similar to the signing structure of two-party ECDSA [18] except for using (Z, Y) as verification key and base point instead of (Q, G), meanwhile based on the self-proving structure $(I_Y = (Y, \pi_Y), y)$, so we can construct a simulator $\mathcal{S}$ who can gain the witness y from $I_Y = (Y, \pi_Y)$ and can simulate the pre-signing oracle $\Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}$ executed as honest P_{1-b}, when P_b is corrupted.

Claim 12 *The interactive pre-signing oracle $\Pi^b_{IpSign_{\langle sk_{1-b},\cdot\rangle}}$ for $b \in \{0,1\}$ can be simulated.*

Proof. We separately construct the simulator $\mathcal{S}$ to show how to simulate the pre-signing oracle for the case of corrupted P_0 and P_1 following ([18], Theorem 5.3). The main difference between our simulator and that of ([18], Theorem 5.3) is that our simulator needs to use the witness y gained from $I_Y = (Y, \pi_Y)$ and the ECDSA signature (r, s) outputted from $\mathcal{F}_{\mathrm{ECDSA}}$ to simulate the interactive pre-signing oracle, while in [18], their simulator only simulates the interactive signing oracle. Fortunately, according to Lemma 3, using the witness y to transform ECDSA signature (r, s) into a pre-signature easily. We construct the simulator as follows.

Simulating pre-signing - corrupted P_0. The idea behind the simulation is that both parties first run a "coin tossing" protocol to generate a random point $\hat{R}$, and then P_1 computes a ciphertext c_3 to P_0. Since the coin-tossing subprotocol

is simulatable like [18]. The simulator can gain the witness y from $I_Y = (Y, \pi_Y)$, so it can also generate the corrupted P_0's view of the decryption of c_3, given only the ECDSA signature $\sigma = (r, s)$ from $\mathcal{F}_{\text{ECDSA}}$.

1. Upon input (sid, m, I_Y), $\mathcal{S}$ gets the witness y, and sends $\mathsf{Sign}(sid, m)$ to $\mathcal{F}_{\text{ECDSA}}$ and receives back a signature $\sigma = (r, s)$.
2. Using the ECDSA verification procedure, $\mathcal{S}$ computes the point $R = kG$ where k is the random number used in signature $\sigma = (r, s)$.
3. $\mathcal{S}$ invokes $\mathcal{A}$ with input (sid, m, I_Y), and simulates the first three messages so that the result is R.
   (a) $\mathcal{S}$ receives $(\mathsf{com\text{-}prove}, sid||0, R_0, k_0)$ from $\mathcal{A}$.
   (b) If $R_0 = k_0 \cdot Y$ then $\mathcal{S}$ sets $R_1 = k_0^{-1} \cdot R$; else it chooses R_1 at random. $\mathcal{S}$ sends $\mathcal{A}$ the message $(\mathsf{proof}, sid||1, R_1)$.
   (c) $\mathcal{S}$ receives $(\mathsf{decom\text{-}proof}, sid||0)$ from $\mathcal{A}$. If $R_0 \neq k_0 \cdot Y$ then simulates P_1 aborting and sends abort to $\mathcal{F}_{\text{ECDSA}}$. Otherwise, it continues.
4. $\mathcal{S}$ chooses a random $\rho \leftarrow \mathbb{Z}_{q^2}$, computes $c_3 \leftarrow \mathsf{Enc}(pk_{\text{enc}}, [k_0 \cdot y \cdot s \bmod q] + \rho \cdot q)$, and internally sends c_3 to $\mathcal{A}$.

As we can see, $\mathcal{S}$ gets (r, s) from $\mathcal{F}_{\text{ECDSA}}$, that is, $s = k^{-1}(m' + rx) \bmod q$, $r = f(R) = f(kG)$, $m' = H(m)$. $\mathcal{S}$ can gain the witness y from I_Y. Therefore, $\mathcal{S}$ can simulate $\hat{R} = k \cdot Y$ with $R = kG$ and $\hat{s} = k^{-1}(m' + \hat{r}x)$ with $y \cdot s$, where $\hat{r} = f(\hat{R})$, and $\hat{\sigma} = (r, y \cdot s)$. That is,

$$f((m' \cdot Y + r \cdot Z)(y \cdot s)^{-1}) = f(kG) = r.$$

The difference between the view of $\mathcal{A}$ in a real execution and in the simulation is the way that $\hat{R}$ is computed. In a real execution, $\hat{R} = k_0 \cdot k_1 \cdot Y$, while $\hat{R} = R$ in the simulation is chosen randomly. Therefore, the distribution is identical. The zero-knowledge proofs and verifications are also identically distributed in the $\mathcal{F}_{\text{zk}}$,$\mathcal{F}_{\text{com-zk}}$-hybrid model. Another difference is c_3. In the simulation it is an encryption of $[k_0 \cdot y \cdot s \bmod q] + \rho \cdot q$, whereas in a real execution it is an encryption of $s' = k_1^{-1} \cdot (m' + rx) + \rho \cdot q$, where $\rho \in \mathbb{Z}_{q^2}$ is random (we stress q that all additions here are over the integers and not mod q, except for where it is explicitly stated in the protocol description). The fact that this is statistically close has already been shown in the proof of ([18], Theorem 4.5). This completes the proof of this simulation case.

Simulating pre-signing - corrupted P_1. The idea behind the simulation is that both parties first run a "coin tossing" protocol to generate a random point R, and then P_0 decrypts the ciphertext c_3 from P_1, if the result is correct, outputs the pre-signature. The coin-tossing subprotocol is simulatable like [18]. The simulator can gain the witness y from $I_Y = (Y, \pi_Y)$ and can simulate the pre-signature from ECDSA signature. The main challenge is that a simulator without the decryption key of Paillier scheme cannot detect the correctness of c_3. We solve this problem follows [18] based on the Paillier-EC assumption, by constructing a simulator with additional oracle $\mathcal{O}_c(c', \alpha, \beta)$ which outputs 1 if and only if $\mathsf{Dec}(sk_{\text{dec}}, c', \alpha, \beta) = \alpha + \beta \cdot x_0 \bmod q$.

1. Upon input (sid, m, I_Y), $\mathcal{S}$ computes the witness y from I_Y, sends $\mathsf{Sign}(sid, m)$ to $\mathcal{F}_{\text{ECDSA}}$ and receives back the ECDSA signature $\sigma = (r, s)$.
2. Using the ECDSA verification procedure, $\mathcal{S}$ computes the point $R = kG$.
3. $\mathcal{S}$ invokes $\mathcal{A}$ with input (sid, m, I_Y), and sends $\mathcal{A}$ $(\mathsf{proof\text{-}receipt}, sid||0)$ as if sent by $\mathcal{F}_{\mathsf{com\text{-}zk}}^{\mathsf{R}_{DL}}$.
4. $\mathcal{S}$ receives R_1 as $\mathcal{A}$ intends to send to P_0, and $(\mathsf{prove}, sid||1, R_1, k_1)$ as $\mathcal{A}$ intends to send to $\mathcal{F}_{\mathsf{zk}}^{\mathsf{R}_{DL}}$, and then verifies that $R_1 = k_1 \cdot Y$ and that R_1 is a non-zero point on the curve; otherwise, it simulates P_0 aborting.
5. $\mathcal{S}$ sets $R_0 = k_1^{-1} \cdot R$ and internally hands $(\mathsf{decom\text{-}proof}, sid||0, R_0)$ to $\mathcal{A}$ as if coming from $\mathcal{F}_{\mathsf{com\text{-}zk}}^{\mathsf{R}_{DL}}$.
6. $\mathcal{S}$ receives c_3 and computes $\alpha = k_1^{-1} \cdot m' \bmod q$, $\beta = k_1^{-1} \cdot r \cdot x_1 \bmod q$, and queries $\mathcal{O}_c(c_3, \alpha, \beta)$, if it gets $b = 0$, outputs abort, otherwise, outputs the pre-signature $\hat{\sigma} = (r, \hat{s})$, where $\hat{s} = min\{y \cdot s \bmod q, q - (y \cdot s \bmod q)\}$.

As we can see, $\mathcal{S}$ gets (r, s) from $\mathcal{F}_{\text{ECDSA}}$. $\mathcal{S}$ can gain the witness y from I_Y. Therefore, $\mathcal{S}$ can simulate $\hat{R} = k \cdot Y$ with $R = kG$ and $\hat{s} = k^{-1}(m' + \hat{r}x)$ with $y \cdot s$, where $\hat{r} = f(\hat{R})$, and $\hat{\sigma} = (r, y \cdot s)$. That is,

$$f((m' \cdot Y + r \cdot Z)(y \cdot s)^{-1}) = f(kG) = r.$$

The distribution of $\hat{R} = k_1 \cdot k_0 \cdot Y$ in a real execution and $\hat{R} = R$ in the simulation is identical. Following [18], based on the Paillier-EC assumption, $\mathcal{S}$ can check the correctness of c_3 outputted by $\mathcal{A}$. The proof follows ([18], Theorem 5.3). This completes the proof of this simulation case.

Lemma 5. *The two-party ECDSA-ASv2 satisfies 2-aEUF–CMA security.*

Proof. The proof of this lemma is similar to the proof of Lemma 3, based on Claim 11, 12. To be specific, we use hybrid games as the proof of Lemma 3, and use the simulator constructed in Claim 11, 12.

In the experiment $\text{aSigForge}^b_{\mathcal{A}, \Pi_{\mathsf{R}, \sum_2}}$ for $b \in \{0, 1\}$ 5, any PPT adversary $\mathcal{A}$ who corrupts P_b obtains access to the random oracle $\mathcal{H}_{\text{ECDSA}}$ and interactive, stateful key generation, signing and pre-signing oracles Π^b_{IKG}, $\Pi^b_{\text{ISign}_{\langle sk_{1-b}, \cdot \rangle}}$, $\Pi^b_{\text{IpSign}_{\langle sk_{1-b}, \cdot \rangle}}$. We can construct a simulator $\mathcal{S}$ following Claim 11, 12 that can simulate above oracles in each hybrid game in the proof of Lemma 3.

In the challenge phase, $\mathcal{S}$ needs to generate the hard relation (I_Y, y) and the proof of π_Z like Lemma 3. Because of holding the witness y, $\mathcal{S}$ can transform the full ECDSA signature into a pre-signature for $\mathcal{A}$. The difference is that since the witness of computes π_Z is also y, $\mathcal{S}$ can generate the proof π_Z directly without simulation in Lemma 3. Therefore, following the proof of Lemma 3 and the proof of Claim 11, 12, our two-party ECDSA-ASv2 satisfies 2-aEUF–CMA security in the UC framework.

Lemma 6. *The two-party ECDSA-ASv2 satisfies two-party witness extractability.*

Proof. The proof of this lemma is similar to the proof of Lemma 4, based on Claim 11, 12. To be specific, we use hybrid games as the proof of Lemma 4, and use the simulator constructed in Claim 11, 12.

In the experiment $\mathrm{aWitExt}^{b}_{\mathcal{A},\Pi_{\mathrm{R},\Sigma_2}}$ for $b \in \{0,1\}$ 6, any PPT adversary $\mathcal{A}$ who corrupts P_b obtains access to the random oracle $\mathcal{H}_{\mathrm{ECDSA}}$ and interactive, stateful key generation, signing and pre-signing oracles Π^{b}_{IKG}, $\Pi^{b}_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}}$, $\Pi^{b}_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}$. We can construct a simulator $\mathcal{S}$ following Claim 11, 12 that can simulate above oracles in each hybrid game in the proof of Lemma 4.

The difference between this proof and the above proof of Lemma 3 is that the hard relation (I_Y, y) and the proof of π_Z is generated by $\mathcal{A}$ in two-party ECDSA-ASv2. Since the self-proving structure of (I_Y, y), $\mathcal{S}$ can also gain the witness y and transform the full ECDSA signature into a pre-signature for $\mathcal{A}$. Especially, π_Z is generated by $\mathcal{A}$ and $\mathcal{S}$ can check its validity. Therefore, following the proof of Lemma 4 and the proof of Claim 11, 12, our two-party ECDSA-ASv2 satisfies two-party witness extractability in the UC framework.

7 Performance and Experimental Results

7.1 Theoretical Analysis

As is shown in Table 1, we give the theoretical analysis of communication cost and efficiency about ECDSA-AS [14,2] 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. [14] does not provide provable security. Then Aumayr et al. [2] uses self-proving structure $\{(I_Y = (Y, \pi_Y), y) | Y = yG \wedge \mathsf{V}(Y, \pi_Y) \rightarrow 1\}$, and gives a provably secure ECDSA-AS based on [14]. 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. For each message to be signed, the signer needs to choose a new random number and needs to compute a new zero-knowledge proof. At the same time, in order to transform ECDSA-AS into the two-party mode, the random number needs to be shared that leads to more complicated zero-knowledge proof.

Our ECDSA-AS schemes also use self-proving structure $\{(I_Y = (Y, \pi_Y), y) | Y = yG \wedge \mathsf{V}(Y, \pi_Y) \rightarrow 1\}$ to realize provable security. Meanwhile, in the pre-signing phase, our schemes 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. Both proofs are independent of the message and random number, so our schemes can use "offline proof" to improve the efficiency. ECDSA-ASv1/2 only computes once point multiplication operation, while ECDSA-AS in [14] and [2] need four times point multiplication operation. Benefiting from holding the ECDSA signing structure, our schemes can enjoy the same efficiency and completely reuse the hardware and software implementation of ECDSA. What's more, for embedding the same hard relation (I_Y, y), one zero-knowledge proof π_Z for $((G, Q, Y, Z), x)$ or $((G, Y, Q, Z), y)$ can be used to pre-signing many

messages, while the size of zero-knowledge proof [14,2] is linearly related to the number of pre-signatures.

Table 1: Communication Cost and Efficiency Comparison

Schemes	pk size	sk size	signature size	online pSign	pVrfy	provable security
ECDSA-AS	$\|\mathbb{G}\|$	$\|\mathbb{Z}_q\|$	$\|\mathbb{G}\|+4\|\mathbb{Z}_q\|$	4PM	6PM	?
ECDSA-AS	$\|\mathbb{G}\|$	$\|\mathbb{Z}_q\|$	$\|\mathbb{G}\|+4\|\mathbb{Z}_q\|$	4PM	6PM	$\surd$
Our ECDSA-AS	$\|\mathbb{G}\|$	$\|\mathbb{Z}_q\|$	$\|\mathbb{G}\|+4\|\mathbb{Z}_q\|$	4PM	6PM	$\surd$
Our ECDSA-ASv2	$\|\mathbb{G}\|$	$\|\mathbb{Z}_q\|$	$2\|\mathbb{Z}_q\|$	PM	6PM	$\surd$

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

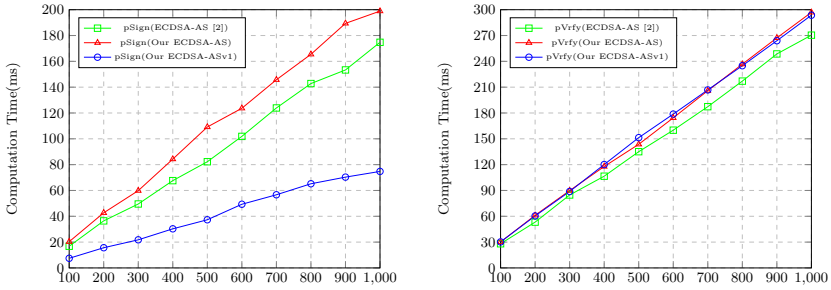

Fig. 4: Efficiency comparison of pre-signing and pre-verification operation

7.2 Experimental Analysis

In order to evaluate the practical performance of our schemes, we implement the ECDSA-AS [2], and our ECDSA-AS and ECDSA-ASv1 based on the OpenSSL library. 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 4, we run our implementations on the standard NIST curves. We run ECDSA-AS [2], our ECDSA-AS and ECDSA-ASv1 many times respectively, and show the efficiency of online pre-signing and a verification operation. Based on the experimental data and analysis results, the average running times of all algorithms in ECDSA-AS and ECDSA-ASv1 are the level of microseconds. Especially, the average running times over 1000 executions of the online pre-signing operation in ECDSA-AS [2], our ECDSA-AS and ECDSA-ASv1 are 174.75 μs, 198.91 μs and 74.74 μs. Therefore, our protocol can compare with the state-of-the-art ECDSA-AS [2].

8 Conclusions

In this paper, we propose an ECDSA-AS and give the security proof based on ECDSA in the random oracle model. Compared with existing ECDSA-AS [14,2], we can use the offline-proof technique to improve the efficiency of the online pre-signing algorithm and then construct two efficient ECDSA-AS called ECDSA-ASv1 and ECDSA-ASv2. For embedding the same hard relation $(I_Y, y) \in \mathsf{R}$, one proof π_Z of hard relation R_Z used in the pre-signing phase can be used to pre-signing many messages. What's more, the pre-signing algorithms of ECDSA-ASv1/2 are similar to the original ECDSA signing algorithm except for modifying some parameters, they can completely reuse the hardware and software implementation of ECDSA and upgrade existing ECDSA applications friendly. In addition, ECDSA-ASv2 uses the same witness y of hard relations (I_Y, y) and $((G, Y, Q, Z), y)$, which is independent of the pre-signing key and random number used in pre-signing phase and need not be shared in constructing two-party AS so it can be transformed into two-party extension easily. Last, we define the two-party AS with interactive key generation, and following the fast two-party ECDSA [18], we construct an efficient two-party ECDSA-AS from our ECDSA-ASv2.

References

1. Poelstra, A.: Lightning in scriptless scripts. mimblewimble team mailing list (2017), `https://lists.launchpad.net/mimblewimble/msg00086.html`
2. 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`
3. Bitcoin Wiki: Payment channels (2018), `https://en.bitcoin.it/wiki/Paymentchannels`
4. 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)
5. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016), `https://lightning.network/lightning-network-paper.pdf`
6. 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`
7. Eckey, L., Faust, S., Hostáková, K., Roos, S.: Splitting payments locally while routing interdimensionally. IACR Cryptol. ePrint Arch. 2020, 555 (2020)
8. 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)
9. 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)
10. 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)
11. 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)
12. Gugger, J.: Bitcoin-monero cross-chain atomic swap. IACR Cryptol. ePrint Arch. 2020, 1126 (2020), https://eprint.iacr.org/2020/1126
13. 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)
14. 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
15. American National Standards Institute: X9.62: Public key cryptography for the financial services industry: The elliptic curve digital signature algorithm (ecdsa) (2005)
16. 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
17. 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)
18. 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
19. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Two-party ECDSA from hash proof systems and efficient instantiations. In: Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III. pp. 191–221 (2019)
20. Yuen, T.H., Cui, H., Xie, X.: Compact zero-knowledge proofs for threshold ECDSA with trustless setup. In: Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12710, pp. 481–511. Springer (2021)
21. Gennaro, R., Goldfeder, S., Narayanan, A.: Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In: Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings. pp. 156–174 (2016), https://doi.org/10.1007/978-3-319-39555-5_9

22. Boneh, D., Gennaro, R., Goldfeder, S.: Using level-1 homomorphic encryption to improve threshold DSA signatures for bitcoin wallet security. In: Progress in Cryptology - LATINCRYPT 2017 - 5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, September 20-22, 2017, Revised Selected Papers. pp. 352–377 (2017)
23. Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 1837–1854 (2018)
24. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 1179–1194 (2018), https://doi.org/10.1145/3243734.3243859
25. Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Threshold ECDSA from ECDSA assumptions: The multiparty case. In: 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019. pp. 1051–1066 (2019), https://doi.org/10.1109/SP.2019.00024
26. Erwig, A., Faust, S., Hostáková, K., Maitra, M., Riahi, S.: Two-party adaptor signatures from identification schemes. In: Garay, J.A. (ed.) Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12710, pp. 451–480. Springer (2021)
27. Lindell, Y.: Highly-efficient universally-composable commitments based on the DDH assumption. In: Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6632, pp. 446–466. Springer (2011)
28. Fujisaki, E.: Improving practical uc-secure commitments based on the DDH assumption. In: Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9841, pp. 257–272. Springer (2016)
29. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols - Techniques and Constructions. Information Security and Cryptography, Springer (2010), https://doi.org/10.1007/978-3-642-14303-8

Supplementary Material

A Signature Scheme

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

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

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

Definition 7 (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:*

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

B Two-party Signature

We review two-party signature and its security in [18,19]. The two-party signature $\Pi = (\mathsf{IGen}, \mathsf{ISign}, \mathsf{Vrfy})$ consists of two distributed protocols and a verification algorithm as follows:

- $\mathsf{IGen}(pp) \to ((sk_0, vk); (sk_1, vk))$. The interactive key generation protocol takes in the public parameter pp and outputs the pre-signing key and verification key $(sk_0, vk); (sk_1, vk)$ for each party $P_i, i = 0, 1$.
- $\mathsf{ISign}_{\langle sk_0, sk_1 \rangle}(m) \to \sigma$. The interactive signing protocol takes in common message $m \in \{0,1\}^*$ and pre-signing key sk_0, sk_1 from each party $P_i, i = 0, 1$ and output the signature σ.
- $\mathsf{Vrfy}_{vk}(m, \sigma) \to 0/1$. The verification algorithm takes the verification key vk, the message $m \in \{0,1\}^*$, and the signature σ as input, and outputs either 0 or 1.

We review the security definition of two-party signature in [18,19]. In the experiment $\text{SigForge}^b_{\mathcal{A},\Pi}(\lambda)$, $b \in \{0,1\}$, a PPT adversary $\mathcal{A}$ corrupting the party P_b in protocol Π can interact with an oracle $\mathcal{O}_{\Pi^b}(\cdot,\cdot)$ to generate the verification key and pre-signing keys and sign any messages concurrently, which executes as another party P_{1-b}. $\mathcal{O}_{\Pi^b}(\cdot,\cdot)$ is defined such that the interactive key generation protocol Π^b_{IKG} is first run once and then the signing protocol $\Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}$ can be executed concurrently.

In the key generation phase, the interactive key generation oracle $\Pi^b_{\text{IKG}}(\cdot,\cdot)$ works as follows:

- Upon receiving $(0,0)$ first, the oracle initializes a machine M and acts as the party P_{1-b} to reply.
- Upon receiving $(0,m)$, the oracle hands M the message m as its next incoming message and returns M's reply.
- Upon receiving (sid,m) is received where $sid \neq 0$, the oracle returns $\perp$.

In the signing phase, the interactive signing oracle $\Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}(\cdot,\cdot)$ works as follows:

- Upon receiving (sid,m) and this is the first oracle query with this identifier sid, the oracle initializes a new machine M_{sid} with the key share and any state stored by M at the end of the key generation phase. The M_{sid} executes as the party P_{1-b} with session identifier sid and input message m to be signed.
- Upon receiving (sid,m) and this is not the first oracle query with this identifier sid, the oracle hands M_{sid} the incoming message m and returns the next message sent by M_{sid}. If M_{sid} concludes, then the output obtained by M_{sid} is returned.

The adversary $\mathcal{A}$ controlling P_b with oracle access to $\mathcal{O}_{\Pi^b}(\cdot,\cdot)$ "wins" if it can forge a signature on a message that is not queried. For a detailed explanation we refer the reader to [18].

Definition 8 *The two-party signature* $\Pi = (\mathsf{IGen}, \mathsf{ISign}, \mathsf{Vrfy})$ *is secure if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* negl *for* $b \in \{0,1\}$*, such that:* $\Pr[\mathit{SigForge}^b_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where the experiment* $\mathit{SigForge}^b_{\mathcal{A},\Pi}$ *is defined as follows:*

$\text{SigForge}^b_{\mathcal{A},\Pi}(\lambda)$	$\Pi^b_{\text{IKG}}(\cdot,\cdot)$
$(vk, sk_b) \leftarrow \mathcal{A}^{\Pi^b_{\text{IKG}}(\cdot,\cdot)}(1^\lambda)$	$((vk, sk_b); ((vk, sk_{1-b})) \leftarrow \Pi^b_{\text{IKG}}(\cdot,\cdot)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}(\cdot,\cdot)}(vk, sk_b)$	
return $\mathsf{Vrfy}_{vk}(m^*, \sigma^*) = 1 \wedge m^* \notin \mathcal{Q}$	$\Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m)$
	$\sigma \leftarrow \Pi^b_{\text{ISign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m)$
	$\mathcal{Q} = \mathcal{Q} \cup \{m\}$

C Hard Relation and Zero-Knowledge Proof

We recall the definition of a hard relation R with statement/witness pairs $(stat = (G, Y), y)$, where $Y = yG$ [2]. Let L_R be the associated language defined as $\mathsf{L}_\mathsf{R} = \{(G,Y)|\exists\ y \text{ s.t. } ((G,Y),y) \in \mathsf{R}\}$. 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 with online extractors as introduced in [17]. The online extractability property allows for extraction of a witness y for an instance (G,Y) from a proof π in the random oracle model and is useful for models where the rewinding proof technique is not allowed, such as UC. We will need this property in order to prove our ECDSA-AS scheme is secure. More formally, a pair (P,V) of PPT algorithms is called a non-interactive zero-knowledge proof of knowledge with an online 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 $\mathsf{V}((G,Y),\mathsf{P}((G,Y),y)) = 1$; (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) Online Extractor: There exists a PPT online 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 $((G,Y),y) \in \mathsf{R}$.

We review some hard relations and their corresponding zero-knowledge proof following [18].

1. *Proof that a Paillier public-key was generated correctly:* define the relation of valid Paillier public keys:

$$\mathsf{R}_\mathrm{P} = \{(N,\phi(N))|\mathsf{gcd}(N,\phi(N)) = 1\}$$

2. *Proof of knowledge of the discrete log of an Elliptic-curve point:* define the relation of discrete log values (relative to the given group).

$$\mathsf{R}_\mathrm{DL} = \{(\mathbb{G},G,q,P,w)|P = w \cdot G\}$$

3. *Proof of knowledge of the equality of discrete logs of two Elliptic-curve points:* define the relation of two discrete log values (relative to the given group).

$$\mathsf{R}_\mathrm{EDL} = \{(\mathbb{G},G,Q,q,X,Y,w)|X = w \cdot G \wedge Y = w \cdot Q\}$$

4. *Prove in zero-knowledge that a value encrypted in a given Paillier and the ciphertext is the discrete log of a given Elliptic curve point:* define the language where pk is a given Paillier public key and sk is its associated private key.

$$\mathsf{L}_\mathrm{PDL} = \{(c,pk,Q_1,\mathbb{G},G,q)|\exists\ (x_1,r) \text{ s.t. } c = \mathsf{Enc}(pk,x;r) \wedge Q_1 = x_1 \cdot G \wedge x_1 \in \mathbb{Z}_q\}$$

D Some Functionalities

We review the ECDSA ideal functionality $\mathcal{F}_{\text{ECDSA}}$ [18] between two parties P_0 and P_1 as follows. The functionality is defined with two functions: key generation and signing. The key generation is called once, and then any arbitrary number of signing operations can be carried out with the generated key.

The ECDSA Functionality $\mathcal{F}_{\text{ECDSA}}$.

- Upon receiving $\mathsf{Gen}(\mathbb{G}, G, q)$ from two parties P_0, P_1, then:
  1. Generate a pair of ECDSA keys (x, Q), where $x \leftarrow \mathbb{Z}_q^*$ is the secret signing key, and $Q = x \cdot G$ is the verification key.
  2. Send Q to both parties.
  3. Ignore future calls to Gen.
- Upon receiving $\mathsf{Sign}(sid, m)$ from P_0, P_1, if Gen was already called and sid has not been stored, then:
  1. Compute an ECDSA signature (r, s) on m.
  2. Send (r, s) to both parties, and store (sid, m).

We review some ideal functionalities used in [18,19], including the ideal commitment functionality $\mathcal{F}_{\text{com}}$, the ideal zero-knowledge functionality $\mathcal{F}_{\text{zk}}$, and the committed non-interactive zero-knowledge functionality $\mathcal{F}_{\text{com-zk}}^{\mathsf{R}}$. For a detailed explanation of these ideal functionalities we refer the reader to [18].

The ideal commitment functionality $\mathcal{F}_{\text{com}}$ [27,28] works with parties P_0 and P_1, formally defined as follows.

The Commitment Functionality $\mathcal{F}_{\text{com}}$.

- Upon receiving $(\mathsf{commit}, sid, x)$ from party P_i (for $i \in \{0, 1\}$), record (sid, i, x) and send $(\mathsf{receipt}, sid)$ to party P_{1-i} . If $(\mathsf{commit}, sid, *)$ is already stored, then ignore the message.
- Upon receiving $(\mathsf{decommit}, sid)$ from party P_i, if (sid, i, x) is recorded then send $(\mathsf{decommit}, sid, x)$ to party P_{1-i}.

A standard ideal zero-knowledge functionality [29] is defined by $((x, w), \emptyset) \leftarrow (\emptyset, (x, \mathsf{R}(x, w)))$. For a relation R, the ideal zero-knowledge functionality is denoted by $\mathcal{F}_{\text{zk}}^{\mathsf{R}}$, formally defined as follows.

The Zero Knowledge Functionality $\mathcal{F}_{\text{zk}}^{\mathsf{R}}$ for Relation R.

- Upon receiving $(\mathsf{prove}, sid, x, w)$ from a party P_i (for $i \in \{0, 1\}$): if $(x, w) \notin \mathsf{R}$ or sid has been previously used then ignore the message. Otherwise, send (proof, sid, x) to party P_{1-i}.

For a relation R, the commitments to non-interactive zero-knowledge proofs of knowledge used by the ideal commitment functionality $\mathcal{F}_{\text{com}}$ is denoted by $\mathcal{F}_{\text{com-zk}}^{\mathsf{R}}$, formally defined as follows.

The Committed NIZK Functionality $\mathcal{F}_{\textbf{com-zk}}^{\mathsf{R}}$ for Relation R.

- Upon receiving $(\mathsf{com\text{-}prove}, sid, x, w)$ from a party P_i (for $i \in \{0,1\}$): if $(x,w) \notin \mathsf{R}$ or sid has been previously used then ignore the message. Otherwise, store (sid, i, x) and send $(\mathsf{proof\text{-}receipt}, sid)$ to P_{1-i}.
- Upon receiving $(\mathsf{decom\text{-}proof}, sid)$ from a party P_i (for $i \in \{0,1\}$): if (sid, i, x) has been stored then send $(\mathsf{decom\text{-}proof}, sid, x)$ to P_{1-i}