On the BUFF Security of ECDSA with Key Recovery

Keita Emura*[1,2]

[1]Kanazawa University, Japan
[2]AIST, Japan

August 27, 2026

Abstract

In the usual syntax of digital signatures, the verification algorithm takes a verification key in addition to a signature and a message, whereas in ECDSA with key recovery, which is used in Ethereum, no verification key is input to the verification algorithm. Instead, a verification key is recovered from a signature and a message. In this paper, we explore BUFF security of ECDSA with key recovery (KR-ECDSA), where BUFF stands for Beyond UnForgeability Features (Cremers et al., IEEE S&P 2021). As a result, we show that KR-ECDSA provides BUFF security, except weak non-resignability (wNR). It is particularly noteworthy that the KR-ECDSA verification algorithm takes an Ethereum address addr as input. This address is defined as the rightmost 160 bits of the Keccak-256 hash of the corresponding ECDSA verification key. Crucially, the algorithm verifies that the hash of the recovered verification key matches addr. Our security analysis shows that the procedure of checking whether the hash value of the recovered verification key is equal to the address is mandatory to provide BUFF security. We also discuss whether wNR is mandatory in Ethereum or not. To clarify which part is mandatory to provide BUFF security in KR-ECDSA, we show that the original ECDSA does not provide any BUFF security. As a by-product of the analysis, we show that one of our BUFF attacks also works against Aumayr et al.'s ECDSA-based adaptor signature scheme (ASIACRYPT 2021) and Qin et al.'s blind adaptor signature scheme (IEEE S&P 2023), which is based on Aumayr et al.'s scheme. We emphasize that the attack is positioned outside of their security models.

1 Introduction

Signatures over Elliptic Curves. ECDSA (Elliptic Curve Digital Signature Algorithm) has gained widespread adoption in blockchain communities since it is used for verifying transactions in the actual systems. The security of ECDSA has been widely evaluated. To name a few, Vaudenay [41] introduced a domain parameter shifting attack against ECDSA where the base point G is replaced by an adversary. Fersch et al. [18] proved that ECDSA is existentially unforgeable against chosen message attack (EUF-CMA) under the elliptic curve discrete logarithm (ECDLP) assumption in the bijective random oracle model, and Brown [5] proved that ECDSA is strongly EUF-CMA secure in the generic group model (we will further discuss the strong EUF-CMA security of ECDSA in Section 4.1). Hartmann and Kiltz [24] focused on the programmability of the conversion function which maps an elliptic curve point into its x-coordinate modulo the group

*k-emura@se.kanazawa-u.ac.jp

order, and showed that an algebraic security reduction for ECDSA can only exist if the security reduction is allowed to program the conversion function.

Brendel et al. [4] evaluated the security of Ed25519 [3]. In addition to (strong) EUF-CMA, they considered BUFF (Beyond UnForgeability Features) security [1, 8]. Aulbach et al. [1] defined that a signature scheme is said to have full BUFF security if it satisfies S-CEO (Strong Conservative Exclusive Ownership), S-DEO (Strong Destructive Exclusive Ownership), MBS (Message Bound Signatures), and wNR (weak Non-Resignability). We will introduce these BUFF security notions in detail in Section 2. Intuitively, the BUFF security (except MBS) considers cases that an adversary produces another verification key that works against a signature which is valid under the original verification key. MBS considers another scenario that an adversary produces a signature and a verification key together with two different messages where the signature is valid with each message under the verification key.

ECDSA with Key Recovery. In the usual syntax of digital signatures, the verification algorithm takes a verification key in addition to a signature and a message, whereas in ECDSA with key recovery (we denote it KR-ECDSA hereafter[1]), which is used in Ethereum, no verification key is input to the verification algorithm. Instead, a verification key is recovered from a signature and a message. Moreover, the verification algorithm of KR-ECDSA takes an Ethereum address addr as input (we regard addr as a part of a signature), which is defined as the rightmost 160 bits of the Keccak-256 hash of the corresponding ECDSA verification key, and checks that the hash value of the recovered verification key is equal to addr. Moreover, it also checks whether a part of a signature value s is $0 < s < q/2 + 1$ where q is the order of the underlying elliptic curve. We refer to this check as the format checking. See Section 3 for the algorithm descriptions.

Our Motivation. Remarkably, the security of KR-ECDSA, including BUFF security, has not been explicitly analyzed so far, to the best of our knowledge. More precisely, Ethereum Yellow Paper [42] refers the document by Johnson, Menezes, and Vanstone [27] that does not explicitly analyze KR-ECDSA. Moreover, Groth and Shoup [23] analyzed ECDSA with additive key derivation and presignatures but did not analyze KR-ECDSA. Even if BUFF security mainly focuses on PQC (Post-Quantum Cryptography) signatures in [1, 8, 15, 29], analyzing whether a currently-employed signature scheme provides BUFF security is also important. Furthermore, using signature schemes that do not satisfy BUFF security in cryptocurrency transactions may lead to the following potential issues. For example, under S-CEO security, the adversary is given a signature, a message, and the original verification key, and attempts to output a different verification key. The adversary wins if the signature is also accepted under this adversary-generated verification key. In Ethereum, verification keys correspond to addresses. Therefore, such an attack undermines the notion of ownership of transaction issuers, allowing a different user (the S-CEO adversary) to claim that they issued the transaction. Under S-DEO security, an adversary is able to modify the content of the message (i.e., the transaction). An adversary breaking MBS security may pose the most severe threat. The adversary outputs a verification key, a signature, and two messages, i.e., the adversary can change only the message while keeping both the verification key and the signature unchanged. In other words, this leaves room for a transaction that has already been issued to be modified afterward. These scenarios represent potential risks rather than confirmed real-world threats. Indeed, at least based on our analysis, we do not consider these weaknesses to be severe enough to lead to critical incidents in practice. However, since transactions are signed by KR-ECDSA in Ethereum and high amounts of cryptocurrency are exchanged, any potential attack against KR-ECDSA should be identified/recognized. Concretely, according to the Ethereum Daily

[1]Ethereum Yellow Paper [42] denotes the ECDSA variant recoverable ECDSA. In this paper, we intentionally denote it key recovery to clarify what the recoverable value is.

Transformation	Signature	S-CEO	S-DEO	M-S-UEO	MBS	NR
PS-1	$\mathsf{Sign}(\mathsf{sk}, m)$, $H(m)$		✓		✓	
PS-2	$\mathsf{Sign}(\mathsf{sk}, m)$, $H(\mathsf{vk})$	✓	✓	✓		
BUFF-lite	$\mathsf{Sign}(\mathsf{sk}, m)$, $H(m, \mathsf{vk})$	✓	✓	✓	✓	
BUFF	$\mathsf{Sign}(\mathsf{sk}, H(m, \mathsf{vk}))$, $H(m, \mathsf{vk})$	✓	✓	✓	✓	✓

Table 1: BUFF Transformations [8]

	Strong EUF-CMA	S-CEO	S-DEO	M-S-UEO	MBS	wNR
ECDSA	✓ (w/ the format checking)					
KR-ECDSA	✓	✓	✓	✓	✓	

We regard ECDSA with the format checking is strongly EUF-CMA secure due to the result by Groth and Shoup [23]. See Section 4.1.
Cremers et al. [8] have mentioned that ECDSA does not provide MBS.

Table 2: Summary of Our Result

Transactions Chart[2], the highest recorded number of transactions so far was 1,961,144 on Sunday, January 14, 2024. More recently, on Wednesday, October 1, 2025, the number of transactions reached 1,704,159. This indicates that KR-ECDSA is executed more than one million times per day. This fact demonstrates that KR-ECDSA is highly relevant to Ethereum. Due to this motivation, we analyze the BUFF security of KR-ECDSA.

We note that generic transformations were summarized by Cremers et al. [8] (See Table 1) that add BUFF security to any signature scheme. We briefly introduce them here because they may fit the signature form of KR-ECDSA. We denote the transformations given by Pornin and Stern [34] are PS-1 and PS-2, respectively, and the transformations given by Cremers et al. [8] are BUFF-lite[3] and BUFF, respectively. Let Sign be a signing algorithm of the underlying signature scheme, $(\mathsf{vk}, \mathsf{sk})$ is a pair of verification key and signing key. We omit PS-3: $(\mathsf{Sign}(\mathsf{sk}, H(m, \mathsf{vk})))$ from the table because it assumes that no weak key exists.[4]

By regarding addr as $H(\mathsf{vk})$ in KR-ECDSA, a signature of KR-ECDSA consists of $(\mathsf{Sign}(\mathsf{sk}, H(m))$, $H(\mathsf{vk}))$. Since $H(\mathsf{vk})$ is added to a signature, KR-ECDSA is at least secure as a signature scheme transformed by PS-2: $(\mathsf{Sign}(\mathsf{sk}, m)$, $H(\mathsf{vk}))$ that provides M-S-UEO (Malicious Strong Universal Exclusive Ownership). This observation shows that KR-ECDSA provides M-S-UEO. Cremers et al. [8] gave implication results: S-CEO and S-DEO are equivalent to S-UEO (Strong Universal Exclusive Ownership), and S-UEO is implied by M-S-UEO. Thus, KR-ECDSA provides S-CEO and S-DEO. However, it is not clear whether KR-ECDSA provides MBS and wNR or not since the signature form $(\mathsf{Sign}(\mathsf{sk}, H(m)), H(\mathsf{vk}))$ is not the same as those of PS-1, PS-2, BUFF-lite, and BUFF summarized in Table 1.

Our Contribution. In this paper, we evaluate strong EUF-CMA security and BUFF security of KR-ECDSA. Our result is summarized in Table 2. As mentioned above, by regarding addr as $H(\mathsf{vk})$, KR-ECDSA provides M-S-UEO, S-CEO, and S-DEO, due to PS-2. In addition,

[2] `https://etherscan.io/chart/tx`

[3] We refer to the ePrint version posted on October 2023 [7] which was updated according to the result by Don et al. [14]

[4] Düzlü and Struck [16] claimed that MBS rules out the possibility of weak keys, and showed that if the original signature schemes satisfies MBS, then the PS-3 transform is sufficient to achieve BUFF security. Janneck [26] introduced random-message validity (RMV) and showed that PS-3 provides NR if the underlying signature scheme satisfies RMV.

- We formally prove that KR-ECDSA is strongly EUF-CMA secure if ECDSA with the format checking is strongly EUF-CMA secure in Section 4.1 (Theorem 1).
- We formally prove that KR-ECDSA provides MBS if the hash function is collision resistant (with respect to the rightmost 160 bits of the hash) in Section 4.3 (Theorem 2).
- We give a concrete wNR attack against KR-ECDSA in Section 4.4.

To clarify which part is mandatory to provide BUFF security in KR-ECDSA, we show that the original ECDSA does not provide any BUFF security.

- We give concrete S-CEO and S-DEO attacks against ECDSA in Section 4.2.

Note that Cremers et al. [8] have mentioned that ECDSA does not provide MBS.

As a by-product of the analysis, we show that our S-DEO attack against the original ECDSA also works against Aumayr et al.'s ECDSA-based adaptor signature scheme [2] and Qin et al.'s blind adaptor signature scheme [35], which is based on Aumayr et al.'s ECDSA-based adaptor signature scheme.

Research Ethics. In this paper, we showed that KR-ECDSA does not provide wNR and gave a concrete attack. Potentially, this may cause an issue in the actual Ethereum transactions. However, as mentioned in the discussion, we would like to claim that the attack is not a realistic threat in Ethereum. We emphasize that we never deny the meaningfulness of NR in other contexts such as [13, 17, 25]. For example, Jackson et al. [25] showed that the Dynamically Recreatable Key protocol [28] is insecure if the signature scheme used in the protocol does not provide NR. We also did not find a vulnerability in the applications of (blind) adaptor signatures. See Section 5 for detail. We also declare all authors do not have Ethereum accounts.

2 Definitions

Let $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. The key generation algorithm KeyGen takes a security parameter λ as input and outputs a verification and signing key pair $(\mathsf{vk}, \mathsf{sk})$. The signing algorithm Sign takes sk and a message to be signed M as input and outputs a signature σ. The verification algorithm Verify takes vk, M, and σ as input and outputs 0 or 1. The correctness requires that for any $\lambda \in \mathbb{N}$, any $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and any M, $\mathsf{Verify}(\mathsf{vk}, M, \mathsf{Sign}(\mathsf{sk}, M)) = 1$ holds with overwhelming probability in the security parameter λ.

2.1 Strong EUF-CMA

The strong unforgeability is defined as follows.

Definition 1 (Strong EUF-CMA) *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and gives vk to $\mathcal{A}$. $\mathcal{C}$ initiates $\mathsf{SigSet} = \emptyset$. $\mathcal{A}$ is allowed to issue signing queries M. $\mathcal{C}$ generates $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$ and returns σ to $\mathcal{A}$. $\mathcal{C}$ stores (M, σ) to SigSet. Finally, $\mathcal{A}$ outputs (M^*, σ^*). We say that $\mathcal{A}$ wins if*

$$\mathsf{Verify}(\mathsf{vk}, M^*, \sigma^*) = 1 \wedge (M^*, \sigma^*) \notin \mathsf{SigSet}$$

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{strong}_{\mathcal{A},\mathsf{Sig}}(\lambda) = \Pr[\mathcal{A}\ wins]$. We say that Sig provides strong existential unforgeability against chosen message attack (strong EUF-CMA) if $\mathsf{Adv}^{strong}_{\mathcal{A},\mathsf{Sig}}(\lambda)$ is negligible for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$. If $\mathcal{A}$ is not allowed to send M^* as a signing query, then we say that Sig is EUF-CMA secure.

2.2 S-CEO, S-DEO, and M-S-UEO

Intuitively, S-CEO (Strong Conservative Exclusive Ownership) guarantees that no adversary can produce a new verification key that accepts a signature which is valid under the original verification key. That is, for (M^*, σ^*) where $\mathsf{Verify}(\mathsf{vk}, M^*, \sigma^*) = 1$, the adversary wins if it produces $\mathsf{vk}^* \neq \mathsf{vk}$ where $\mathsf{Verify}(\mathsf{vk}^*, M^*, \sigma^*) = 1$ holds. The S-CEO security is formally defined as follows.

Definition 2 (S-CEO) *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and gives vk to $\mathcal{A}$. $\mathcal{C}$ initiates $\mathsf{SigSet} = \emptyset$. $\mathcal{A}$ is allowed to issue signing queries M. $\mathcal{C}$ generates $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$ and returns σ to $\mathcal{A}$. $\mathcal{C}$ stores (M, σ) to SigSet. Finally, $\mathcal{A}$ outputs $(\mathsf{vk}^*, M^*, \sigma^*)$. We say that $\mathcal{A}$ wins if*

$$\mathsf{Verify}(\mathsf{vk}^*, M^*, \sigma^*) = 1 \wedge (M^*, \sigma^*) \in \mathsf{SigSet} \wedge \mathsf{vk}^* \neq \mathsf{vk}$$

hold. Here, $(M^*, \sigma^*) \in \mathsf{SigSet}$ implies $\mathsf{Verify}(\mathsf{vk}, M^*, \sigma^*) = 1$. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\text{S-CEO}}_{\mathcal{A},\mathsf{Sig}}(\lambda) = \Pr[\mathcal{A}\ \text{wins}]$. We say that Sig provides S-CEO if $\mathsf{Adv}^{\text{S-CEO}}_{\mathcal{A},\mathsf{Sig}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.

Basically, S-DEO (Strong Destructive Exclusive Ownership) guarantees the same as those of S-CEO except that $\mathcal{A}$ produces a different message.

Definition 3 (S-DEO) *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and gives vk to $\mathcal{A}$. $\mathcal{C}$ initiates $\mathsf{SigSet} = \emptyset$. $\mathcal{A}$ is allowed to issue signing queries M. $\mathcal{C}$ generates $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$ and returns σ to $\mathcal{A}$. $\mathcal{C}$ stores (M, σ) to SigSet. Finally, $\mathcal{A}$ outputs $(\mathsf{vk}^*, M^*, \sigma^*)$. Let $(M, \sigma^*) \in \mathsf{SigSet}$. $\mathcal{A}$ wins if $\mathsf{Verify}(\mathsf{vk}^*, M^*, \sigma^*) = 1$, $M^* \neq M$, and $\mathsf{vk}^* \neq \mathsf{vk}$. Here, $(M, \sigma^*) \in \mathsf{SigSet}$ implies $\mathsf{Verify}(\mathsf{vk}, M, \sigma^*) = 1$. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\text{S-DEO}}_{\mathcal{A},\mathsf{Sig}}(\lambda) = \Pr[\mathcal{A}\ \text{wins}]$. We say that Sig provides S-DEO if $\mathsf{Adv}^{\text{S-DEO}}_{\mathcal{A},\mathsf{Sig}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.*

Finally, we introduce M-S-UEO (Malicious Strong Universal Exclusive Ownership) that is the strongest variant of the exclusive ownership notions. We assume that the adversary $\mathcal{A}$ implicitly takes a security parameter and other parameter, i.e., (E, G, p, q) in the case of (KR-)ECDSA.

Definition 4 (M-S-UEO) *The adversary $\mathcal{A}$ outputs $(M_1, M_2, \sigma, \mathsf{vk}_1, \mathsf{vk}_2)$. Since $\mathcal{A}$ produces vk_1 and vk_2, no signing query is defined. We say that $\mathcal{A}$ wins if*

$$\mathsf{Verify}(\mathsf{vk}_1, M_1, \sigma) = 1 \wedge \mathsf{Verify}(\mathsf{vk}_2, M_2, \sigma) = 1 \wedge \mathsf{vk}_1 \neq \mathsf{vk}_2$$

hold. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\text{M-S-UEO}}_{\mathcal{A},\mathsf{Sig}}(\lambda) = \Pr[\mathcal{A}\ \text{wins}]$. We say that Sig provides M-S-UEO if $\mathsf{Adv}^{\text{M-S-UEO}}_{\mathcal{A},\mathsf{Sig}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.

In PS-2 (See Table 1), a signature contains $H(\mathsf{vk})$. Observe that a targeted signature σ^* is produced via a signing query in the definitions of S-CEO and S-DEO. Thus, no adversary can produce a new verification key vk^* unless it breaks the collision resistance of the hash function H. This is the main reason why that PS-2 provides S-CEO and S-DEO security. Moreover, again due to the hash function H, no adversary can produce two distinct verification keys vk_1 and vk_2 satisfying $H(\mathsf{vk}_1) = H(\mathsf{vk}_2)$. This is the main reason why that PS-2 provides M-S-UEO security.

2.3 MBS

Intuitively, MBS (Message Bound Signatures) guarantees that no adversary $\mathcal{A}$ can produce two distinct messages that are accepted by a signature and verification key pair. If MBS does not hold, then $\mathcal{A}$ could switch a message after producing a signature, and could claim that it actually signed a different message. Here, $\mathcal{A}$ is allowed to introduce vk (and thus $\mathcal{A}$ is allowed to know the corresponding sk or $\mathcal{A}$ may not generate vk via the KeyGen algorithm). The MBS security is formally defined as follows. We assume that the adversary $\mathcal{A}$ implicitly takes a security parameter and other parameter, i.e., (E, G, p, q) in the case of (KR-)ECDSA.

Definition 5 (MBS) *The adversary $\mathcal{A}$ outputs $(M_1, M_2, \sigma, \mathsf{vk})$. Since $\mathcal{A}$ produces vk, no signing query is defined. We say that $\mathcal{A}$ wins if*

$$\mathsf{Verify}(\mathsf{vk}, M_1, \sigma) = 1 \wedge \mathsf{Verify}(\mathsf{vk}, M_2, \sigma) = 1 \wedge M_1 \neq M_2$$

hold. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{MBS}_{\mathcal{A},\mathsf{Sig}}(\lambda) = \Pr[\mathcal{A}\ wins]$. We say that Sig provides MBS if $\mathsf{Adv}^{MBS}_{\mathcal{A},\mathsf{Sig}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.

2.4 wNR

Intuitively, NR (Non-Resignability) guarantees that no adversary $\mathcal{A}$ can produce a signature and a verification key when $\mathcal{A}$ obtains a signature but does not know the message. For example, (KR-)ECDSA computes $h = H(M)$ and h is used for signing. Then, $\mathcal{A}$ may not be able to obtain M even if h is revealed (in the sense of one-wayness of H). Note that Don et al. [13] showed that the original NR is actually not achievable where auxiliary information can be used to leak crucial information that is sufficient to attack NR. There is no auxiliary information in a weaker form of NR (wNR) [14].

Definition 6 (wNR) *Let $\mathcal{A}_0$ and $\mathcal{A}_1$ be adversaries and $\mathcal{C}$ be the challenger. $\mathcal{C}$ generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and gives vk to $\mathcal{A}_0$. $\mathcal{A}_0$ declares M^*. $\mathcal{C}$ generates $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M^*)$ and gives (vk, σ) to $\mathcal{A}_1$.[5] We remark that $\mathcal{A}_1$ is not explicitly given M^* since no state information is shared between $\mathcal{A}_0$ and $\mathcal{A}_1$. $\mathcal{A}_1$ produces $(\mathsf{vk}^*, \sigma^*)$. We say that $(\mathcal{A}_0, \mathcal{A}_1)$ win if*

$$\mathsf{Verify}(\mathsf{vk}^*, M^*, \sigma^*) = 1 \wedge \mathsf{vk} \neq \mathsf{vk}^*$$

hold. The advantage of $(\mathcal{A}_0, \mathcal{A}_1)$ is defined as $\mathsf{Adv}^{wNR}_{(\mathcal{A}_0,\mathcal{A}_1),\mathsf{Sig}}(\lambda) = \Pr[(\mathcal{A}_0, \mathcal{A}_1)\ wins]$. We say that Sig provides wNR if $\mathsf{Adv}^{wNR}_{(\mathcal{A}_0,\mathcal{A}_1),\mathsf{Sig}}(\lambda)$ is negligible for all PPT adversaries $(\mathcal{A}_0, \mathcal{A}_1)$.

3 Key Recovery Signatures and KR-ECDSA

3.1 Key Recovery Signatures

First, we introduce key-recovery signatures to clarify the security models of KR-ECDSA. Since this is clear from the context and the inputs to the algorithm, we use the same algorithm names as those for standard signatures. Let $\mathsf{Sig}_{\mathsf{KR}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a key recovery signature scheme. The key generation algorithm KeyGen takes a security parameter λ as input and outputs a verification

[5] Don et al. [13] introduced a variant of NR where sk is given to an adversary. We do not consider the notion in this paper since our attack completes even if an adversary does not have sk.

and signing key pair $(\mathsf{vk}, \mathsf{sk})$. The signing algorithm Sign takes sk and a message to be signed M as input and outputs a signature σ. The verification algorithm Verify takes M, and σ as input and outputs 0 or 1. The Verify algorithm implicitly recovers a verification key. We assume that the Verify algorithm is deterministic, i.e., the verification key is deterministically recovered from the pair (M, σ). That is, in MBS for key recovery signatures (defined later), the winning condition implicitly requires the recovered keys from (M_1, σ) and (M_2, σ) to coincide. The correctness requires that for any $\lambda \in \mathbb{N}$, any $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and any M, $\mathsf{Verify}(M, \mathsf{Sign}(\mathsf{sk}, M)) = 1$ holds with overwhelming probability in the security parameter λ.

On Security Models for Key Recovery Signatures. Since the Verify algorithm does not take vk as input, we need to clarify security models that fit the syntax of KR-ECDSA. Basically, they can be analogously defined as follows.

Strong EUF-CMA: $\mathcal{C}$ generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and gives vk to $\mathcal{A}$. $\mathcal{C}$ initiates $\mathsf{SigSet} = \emptyset$. $\mathcal{A}$ is allowed to issue signing queries M. $\mathcal{C}$ generates $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$ and returns σ to $\mathcal{A}$. $\mathcal{C}$ stores (M, σ) to SigSet. Finally, $\mathcal{A}$ outputs (M^*, σ^*). Let vk^* be the verification key which is recovered from (M^*, σ^*) in $\mathsf{Verify}(\sigma^*, M^*)$. We say that $\mathcal{A}$ wins if $\mathsf{Verify}(M^*, \sigma^*) = 1 \wedge (M^*, \sigma^*) \notin \mathsf{SigSet} \wedge \mathsf{vk}^* = \mathsf{vk}$.

S-CEO: $\mathcal{C}$ generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and gives vk to $\mathcal{A}$. $\mathcal{C}$ initiates $\mathsf{SigSet} = \emptyset$. $\mathcal{A}$ is allowed to issue signing queries M. $\mathcal{C}$ generates $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$ and returns σ to $\mathcal{A}$. $\mathcal{C}$ stores (M, σ) to SigSet. Finally, $\mathcal{A}$ outputs (M^*, σ^*) and let vk^* be the verification key which is recovered by (M^*, σ^*). We say that $\mathcal{A}$ wins if $\mathsf{Verify}(M^*, \sigma^*) = 1 \wedge (M^*, \sigma^*) \in \mathsf{SigSet} \wedge \mathsf{vk}^* \neq \mathsf{vk}$ hold.

S-DEO: $\mathcal{C}$ generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and gives vk to $\mathcal{A}$. $\mathcal{C}$ initiates $\mathsf{SigSet} = \emptyset$. $\mathcal{A}$ is allowed to issue signing queries M. $\mathcal{C}$ generates $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$ and returns σ to $\mathcal{A}$. $\mathcal{C}$ stores (M, σ) to SigSet. Finally, $\mathcal{A}$ outputs (M^*, σ^*) and let vk^* be the verification key which is recovered by (M^*, σ^*). Let $(M, \sigma^*) \in \mathsf{SigSet}$. $\mathcal{A}$ wins if $\mathsf{Verify}(M^*, \sigma^*) = 1 \wedge M^* \neq M \wedge \mathsf{vk}^* \neq \mathsf{vk}$.

M-S-UEO: $\mathcal{A}$ outputs $(M_1, M_2, \sigma, \mathsf{vk}_1, \mathsf{vk}_2)$ where vk_1 (resp. vk_2) is the verification key recovered from (M_1, σ) (resp. (M_2, σ)) in $\mathsf{Verify}(M_1, \sigma)$ (resp. $\mathsf{Verify}(M_2, \sigma)$). We say that $\mathcal{A}$ wins if $\mathsf{Verify}(M_1, \sigma) = 1 \wedge \mathsf{Verify}(M_2, \sigma) = 1 \wedge \mathsf{vk}_1 \neq \mathsf{vk}_2$ hold.

MBS: $\mathcal{A}$ outputs $(M_1, M_2, \sigma, \mathsf{vk})$, We say that $\mathcal{A}$ wins if $\mathsf{Verify}(M_1, \sigma) = 1 \wedge \mathsf{Verify}(M_2, \sigma) = 1 \wedge M_1 \neq M_2$ hold and vk is recovered from both (M_1, σ) and (M_2, σ) in $\mathsf{Verify}(M_1, \sigma)$ and $\mathsf{Verify}(M_2, \sigma)$, respectively.

wNR: $\mathcal{C}$ generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and gives vk to $\mathcal{A}_0$. $\mathcal{A}_0$ declares M^*. $\mathcal{C}$ generates $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M^*)$ and gives (vk, σ) to $\mathcal{A}_1$. $\mathcal{A}_1$ produces σ^*. We say that $(\mathcal{A}_0, \mathcal{A}_1)$ win if $\mathsf{Verify}(M^*, \sigma^*) = 1 \wedge \mathsf{vk} \neq \mathsf{vk}^*$ hold where vk^* is the verification key recovered from (M^*, σ^*) in $\mathsf{Verify}(M^*, \sigma^*)$.

3.2 KR-ECDSA

According to Ethereum Yellow Paper [42], let $\mathcal{B}_{96..255}$ be the rightmost 160 bits of the input. Then, $\mathsf{addr} = \mathcal{B}_{96..255}(H(\mathsf{vk}))$. Let p and q be prime numbers, $H : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function, $E/\mathbb{F}_p : y^2 = x^3 + ax + b$ be an elliptic curve with order q defined over $\mathbb{F}_p$, and $G \in E(\mathbb{F}_p)$ be a base point. We assume that each algorithm implicitly takes (E, G, p, q) as input. We denote

the coordinate of a point $R \in E(\mathbb{F}_p)$ as $R = (R_x, R_y)$. Here, there are two y-coordinates for R_x satisfying $R_y^2 = R_x^3 + aR_x + b$. To uniquely recover R from R_x, KR-ECDSA introduces the flag v that decides whether R_y is greater than $q/2$ or not. That is, $R = (R_x, R_y)$ is uniquely determined by (R_x, v).

$\mathsf{KeyGen}(1^\lambda)$**:** Choose $d \xleftarrow{\$} \mathbb{Z}_q$ and compute $P = dG$. Output $\mathsf{vk} = P$, $\mathsf{sk} = (d, P)$, and $\mathsf{addr} = \mathcal{B}_{96..255}(H(P))$.

$\mathsf{Sign}(\mathsf{sk}, M)$**:** Parse $\mathsf{sk} = (d, P)$. Choose $r \xleftarrow{\$} \mathbb{Z}_q$, and compute $h = H(M)$ and $R = rG$. Let $R = (R_x, R_y)$ and v be the flag. Compute $s = \frac{h + dR_x}{r} \bmod q$. If $0 < s < q/2 + 1$, then output a signature $\sigma = (\mathsf{addr}, s, R_x, v)$ where $\mathsf{addr} = \mathcal{B}_{96..255}(H(P))$. Otherwise, set $s' = -s \bmod q$ and $\bar{v}$ is set the opposite flag of v, and output a signature $\sigma = (\mathsf{addr}, s', R_x, \bar{v})$.

$\mathsf{Verify}(\sigma, M)$**:** Parse $\sigma = (\mathsf{addr}, s, R_x, v)$. Output 0 if $0 < s < q/2 + 1$ does not hold. Otherwise, compute $R = (R_x, R_y)$ from (R_x, v), and recover $P = \frac{s}{R_x}(R - \frac{h}{s}G)$. Output 1 if $\mathsf{addr} = \mathcal{B}_{96..255}(H(P))$ and 0 otherwise.

The verification algorithm of the original ECDSA (that additionally takes P as input) outputs 1 if the x-coordinate of $\frac{h}{s}G + \frac{R_x}{s}P$ is R_x since

$$\frac{h}{s}G + \frac{R_x}{s}P = \frac{h}{s}G + \frac{R_x d}{s}G = \frac{h + R_x d}{s}G = rG = R$$

holds. From this verification equation,

$$P = \frac{s}{R_x}(R - \frac{h}{s}G)$$

holds.

We remark that $\frac{h}{-s}G + \frac{R_x}{-s}P = -R$ holds when $\frac{h}{s}G + \frac{R_x}{s}P = R$. Since the x-coordinates of R and $-R$ are the same, if the range of s is not checked, then $(\mathsf{addr}, -s, R_x, \bar{v})$ is also a valid signature if $(\mathsf{addr}, s, R_x, v)$ is a valid signature. KR-ECDSA explicitly checks the range of s.

4 Security Analysis of KR-ECDSA

4.1 Strong EUF-CMA

We remark that KR-ECDSA is not strongly EUF-CMA secure if the format checking (whether $0 < s < q/2 + 1$ or not) is not employed. The concrete attack is described as follows. An adversary issues a signing query M and obtains $\sigma = (\mathsf{addr}, s, R_x, v)$. Then, the adversary outputs $\sigma^* = (\mathsf{addr}, -s \bmod q, R_x, \bar{v})$ ($\bar{v}$ is set the opposite flag of v). Here, σ^* is a valid signature on M and $\sigma \neq \sigma^*$. That is, the format checking is at least mandatory to provide strong EUF-CMA security. The above attack works against the original ECDSA that contradicts the proof by Brown [5] where ECDSA is strongly EUF-CMA secure if the hash function is collision resistant in the generic group model. Stern et al. [37] mentioned that "*What goes wrong here is the adequacy of the model. The proof is correct but the underlying model is flawed, since it disallows production of malleable signatures*". Fersch et al. [18] introduced GenDSA which is an abstract signature framework that subsumes both DSA and ECDSA in unmodified form. They introduced a function φ and proved that GenDSA is EUF-CMA secure if φ is semi-injective, the hash function H is secure, and the discrete logarithm problem is hard in the bijective random oracle model. Though they also mentioned that

GenDSA is strongly unforgeable if and only if φ is injective, they mentioned that "*our overall results in a nutshell are: DSA signatures are strongly unforgeable, and ECDSA signatures are existentially unforgeable.*" Aumayr et al. [2] called ECDSA with the format checking the positive ECDSA scheme, and mentioned that it is assumed to be strong EUF-CMA. However, no formal security proof was given. Groth and Shoup [23] said that two ECDSA signatures (s, R_x) and $(-s, R_x)$ are equivalent "up to sign" and said that ECDSA is strongly unforgeable "up to sign" if it is hard to construct a valid signature on a message other than one that is equivalent up to sign. They mentioned that strong unforgeability up to sign implies that ECDSA can be trivially converted to a strongly secure signature scheme, simply by requiring that a valid signature (s, R_x) satisfies $0 < s < q/2 + 1$. According to the above situation, we assume that ECDSA with the format checking is strongly EUF-CMA secure.

Theorem 1 *KR-ECDSA is strongly EUF-CMA secure if ECDSA with the format checking is strongly EUF-CMA secure.*

Proof. Let $\mathcal{A}$ be an adversary of EUF-CMA against KR-ECDSA and $\mathcal{C}$ be the challenger of strong EUF-CMA against ECDSA with the format checking. We construct a reduction algorithm $\mathcal{R}$ that breaks the EUF-CMA security of the ECDSA by using $\mathcal{A}$ as follows. $\mathcal{C}$ sends $\mathsf{vk}^* = P^*$ to $\mathcal{R}$. $\mathcal{R}$ forwards P^* to $\mathcal{A}$. Since $\mathsf{addr} = \mathcal{B}_{96..255}(H(P^*))$ is determined by P^*, $\mathcal{A}$ also knows addr.

When $\mathcal{A}$ sends a signing query M to $\mathcal{R}$, $\mathcal{R}$ sends M to $\mathcal{C}$ as a signing query and obtains (s, R_x). Here, due to the format checking procedure, $0 < s < q/2 + 1$ holds. $\mathcal{R}$ computes $\frac{h}{s}G + \frac{R_x}{s}P = R$ where $h = H(M)$ and determines v by $R = (R_x, R_y)$, and returns $\sigma = (\mathsf{addr}, s, R_x, v)$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs $\sigma^* = (\mathsf{addr}, s^*, R_x^*, v^*)$ and M^* where for $h^* = H(M^*)$ and R^* which is determined by (R_x^*, v^*), $\frac{s^*}{R_x^*}(R^* - \frac{h^*}{s^*}G) = P^*$, $0 < s^* < q/2 + 1$, and $\mathsf{addr} = \mathcal{B}_{96..255}(H(P^*))$ hold.

If $\mathcal{A}$ did not send M^* as a signing query, then $\mathcal{R}$ outputs (s^*, R_x^*) and M^* that breaks the strong EUF-CMA security of ECDSA with the format checking. Otherwise, let (s, R_x) be a signature on M^* that $\mathcal{R}$ obtains from $\mathcal{C}$ as the response to the signing query M^*, and $\sigma = (\mathsf{addr}, s, R_x, v)$ be the response to $\mathcal{A}$ as the signing query M^*. Due to the winning condition, $\mathcal{A}$ did not obtain σ^* as the response to the signing query M^*. That is, $\sigma^* \neq \sigma$. Since $0 < s < q/2 + 1$, the corresponding v is uniquely determined from (s, R_x). Moreover, P^* is uniquely recovered from (s, R_x, v). Since H is a deterministic function, $\mathsf{addr} = \mathcal{B}_{96..255}(H(P^*))$ is also uniquely determined from P^*. To sum up, if $\sigma^* \neq \sigma$, then $(s^*, R_x^*) \neq (s, R_x)$ holds. $\mathcal{R}$ outputs (s^*, R_x^*) and M^* that breaks the strong EUF-CMA security of ECDSA with the format checking. □

As a remark, strong EUF-CMA security prevents to re-randomize a signature. That is, a signature is non-malleable. Since signature malleability could lead to replay attacks, strong EUF-CMA security seems critical for blockchain applications.

4.2 S-CEO, S-DEO, and M-S-UEO

To clarify which part is mandatory to provide S-CEO, S-DEO, and M-S-UEO in KR-ECDSA, we show that the original ECDSA does not provide S-CEO and S-DEO as follows (this implies the original ECDSA does not provide M-S-UEO since S-CEO and S-DEO are equivalent to S-UEO, and S-UEO is implied by M-S-UEO). We remark that Pornin and Stern [34] gave an attack for DSA which they called second key construction, and claimed that the attack can trivially be applied to ECDSA. Their attack changes the base point G and can be regarded as a variant of the domain parameter shifting attack [41]. In our attack, we follow the definitions of S-CEO and S-DEO, and do not change G.

Our S-CEO Attack. Let $\sigma = (s, R_x)$ be a ECDSA signature on M and $P = dG$ be a ECDSA verification key. That is, $\frac{h}{s}G + \frac{R_x}{s}P = R$ and $R = (R_x, R_y)$ holds. An S-CEO adversary can produce another verification key $P^* = \frac{-2h}{R_x}G - P$. Then,

$$\begin{aligned}\frac{h}{s}G + \frac{R_x}{s}P^* &= \frac{h}{s}G + \frac{R_x}{s}(\frac{-2h}{R_x}G - P)\\ &= \frac{h}{s}G - \frac{2h}{s}G - \frac{R_x}{s}P\\ &= -(\frac{h}{s}G + \frac{R_x}{s}P)\\ &= -R\end{aligned}$$

hold. Thus, $\sigma = (s, R_x)$ is a valid ECDSA signature on M under P^* and $P \neq P^*$ (if $d \neq \frac{h}{R_x}$ that holds with overwhelming probability). Clearly, our S-CEO attack above does not work for ECDSA with the format checking and highlights other effectiveness of the format checking besides providing strong EUF-CMA. We emphasize that it does not formally prove ECDSA with the format checking provides S-CEO security here.

Our S-DEO Attack. Similarly, the original ECDSA does not provide S-DEO as follows. Let $\sigma = (s, R_x)$ be a ECDSA signature on M and P be a ECDSA verification key. An S-DEO adversary chooses $M^* \neq M$, computes $h^* = H(M^*)$ and $\frac{h}{s}G + \frac{R_x}{s}P = R$, and sets $P^* = \frac{s}{R_x}(R - \frac{h^*}{s}G)$. Then,

$$\begin{aligned}\frac{h^*}{s}G + \frac{R_x}{s}P^* &= \frac{h^*}{s}G + \frac{R_x}{s}(\frac{s}{R_x}(R - \frac{h^*}{s}G))\\ &= R\end{aligned}$$

hold. Thus, $\sigma = (s, R_x)$ is a valid ECDSA signature on M^* under P^* and $P \neq P^*$ and $M \neq M^*$. Remark that $P = P^*$ is equivalent to $h = h^*$. Since $M \neq M^*$, we can assume that $h \neq h^*$ due to the collision resistance of H.

Clearly, our S-DEO attack above works for ECDSA with the format checking. As a by-product of our S-DEO attack above, we show that the attack also works against Aumayr et al.'s ECDSA-based adaptor signature scheme [2]. See Section 5.

KR-ECDSA provides M-S-UEO (and thus S-CEO and S-DEO). By regarding addr as $H(\mathsf{vk})$ in KR-ECDSA, a signature of KR-ECDSA consists of $(\mathsf{Sign}(\mathsf{sk}, H(m)), H(\mathsf{vk}))$. More precisely, for addr, let the address holder who has $\mathsf{sk} = d$ issue a transaction M with a signature σ. To verify σ, the verifier knows addr. Thus, even if an adversary produces $P^* \neq P$, the attack is prevented by checking $\mathsf{addr} := \mathcal{B}_{96..255}(H(P^*))$ holds. Due to PS-2: $(\mathsf{Sign}(\mathsf{sk}, m), H(\mathsf{vk}))$ that provides M-S-UEO, KR-ECDSA provides M-S-UEO (and thus S-CEO and S-DEO) if the hash function is collision resistant (with respect to the rightmost 160 bits of the hash).

4.3 MBS

To clarify which part is mandatory to provide MBS in KR-ECDSA, we show that the original ECDSA does not provide MBS as follows. Note that Cremers et al. [8] have mentioned that ECDSA does not provide MBS, and the following attack is essentially the same as that of Stern et al. (See Section 4. Duplicates in ECDSA in [37]) and Vaudenay (See Section 2.1. Signature Manipulation in ECDSA in [40]).

Let $\sigma = (s, R_x)$ be a ECDSA signature and $P = dG$ be a ECDSA verification key. For a point $R = (R_x, R_y)$, we denote $(R)_x = R_x$ and $(R)_y = R_y$, respectively. We consider when σ is valid

under P for both M_1 and M_2 and $M_1 \neq M_2$. Due to the collision resistance of H, for $h_1 = H(M_1)$ and $h_2 = H(M_2)$, $h_1 \neq h_2$ holds if $M_1 \neq M_2$. If $\frac{h_1}{s}G + \frac{R_x}{s}P = \frac{h_2}{s}G + \frac{R_x}{s}P$, then $h_1 = h_2$. Thus, $\frac{h_1}{s}G + \frac{R_x}{s}P \neq \frac{h_2}{s}G + \frac{R_x}{s}P$ must hold. The possibility is $(\frac{h_1}{s}G + \frac{R_x}{s}P)_x = (\frac{h_2}{s}G + \frac{R_x}{s}P)_x = R_x$ but $(\frac{h_1}{s}G + \frac{R_x}{s}P)_y \neq (\frac{h_2}{s}G + \frac{R_x}{s}P)_y$, i.e., $\frac{h_1}{s}G + \frac{R_x}{s}P = (R_x, R_y)$ and $\frac{h_2}{s}G + \frac{R_x}{s}P = (R_x, -R_y)$. From this relation, we obtain $\frac{h_1}{s}G + \frac{R_x}{s}P + \frac{h_2}{s}G + \frac{R_x}{s}P = \mathcal{O}$ and $h_1 + h_2 = -2dR_x$. From this relation, we can attack ECDSA as follows.

1. Choose distinct M_1 and M_2, and let $h_1 = H(M_1)$ and $h_2 = H(M_2)$.
2. Choose $r \xleftarrow{\$} \mathbb{Z}_q$ and compute $R = rG$. We denote $R = (R_x, R_y)$.
3. Define $d = -\frac{h_1+h_2}{2R_x}$ and compute $P = dG$.
4. Compute $s = \frac{h_1-h_2}{2r}$
5. Output $(M_1, M_2, (s, R_x), P)$.

Here, $\sigma = (s, R_x)$ is valid under P for both M_1 and M_2 since $\frac{h_1}{s}G + \frac{R_x}{s}P = \frac{h_1+dR_x}{s}G = rG = (R_x, R_y)$ and $\frac{h_2}{s}G + \frac{R_x}{s}P = -rG = (R_x, -R_y)$ hold. Clearly, the MBS attack above does not work for ECDSA with the format checking and highlights other effectiveness of the format checking besides providing strong EUF-CMA. We emphasize that it does not formally prove ECDSA with the format checking provides MBS security here.

KR-ECDSA provides MBS. The main observation of the above attack is the ECDSA verification algorithm just checks the x-coordinate of $\frac{h}{s}G + \frac{R_x}{s}P$. At the first sight, the attack is prevented in KR-ECDSA by additionally checking $0 < s < q/2 + 1$. We observe that the verification algorithm recovers R from (R_x, v) that implicitly prevents to use $-R$. That is, the format checking is unnecessary at least for providing MBS. Formally, we prove that KR-ECDSA provides MBS as follows. Although our security proof is somewhat trivial, the analysis clarifies an important point: a single signature can be associated with two different transactions if a weak hash function, such as RIPEMD, is employed (for example, to improve efficiency), since collisions can be found efficiently.

Theorem 2 *KR-ECDSA provides MBS if H is a collision resistant hash function.*

Proof. Let $\mathcal{A}$ be an adversary of MBS. $\mathcal{A}$ outputs $(M_1, M_2, (\mathsf{addr}, s, R_x, v))$. From (R_x, v), $R = (R_x, R_y)$ is uniquely determined. Let $h_1 = H(M_1)$, $h_2 = H(M_2)$, $P_1 = \frac{s}{R_x}(R - \frac{h_1}{s}G)$, and $P_2 = \frac{s}{R_x}(R - \frac{h_2}{s}G)$. If $P_1 \neq P_2$, then the collision resistance is broken since $\mathsf{addr} = \mathcal{B}_{96..255}(H(P_1)) = \mathcal{B}_{96..255}(H(P_2))$ hold. Thus, $P_1 = P_2$ and then $h_1 = h_2$ holds. Then the collision resistance is broken since $M_1 \neq M_2$. In both cases, we can reduce the collision resistance of the hash function. In the strict sense, the usual collision resistance does not guarantee that $\mathcal{B}_{96..255}(H(P_1)) \neq \mathcal{B}_{96..255}(H(P_2))$ holds for $P_1 \neq P_2$. Here, we assume that H is collision resistant with respect to the rightmost 160-bit of the hash. □

4.4 wNR

Here, we show that KR-ECDSA does not provide wNR, and discuss its influence.

Our wNR Attack. We construct adversaries $(\mathcal{A}_0, A_1)$ as follows. $\mathcal{C}$ generates $\mathsf{vk} = P$ and gives P to $\mathcal{A}_0$. $\mathcal{A}_0$ declares a message M (We do not require any restriction here, and a random message is sufficient). $\mathcal{C}$ computes $\sigma = (\mathsf{addr}, s, R_x, v)$ on M and sends σ to $\mathcal{A}_1$. Again, $\mathcal{A}_1$ is not explicitly

given M since no state information is shared between $\mathcal{A}_0$ and $\mathcal{A}_1$. $\mathcal{A}_1$ chooses $s^* \in \mathbb{Z}_q$ such that $0 < s^* < q/2 + 1$, sets $r^* = 1/s^*$, and computes $R^* = r^*G$. $\mathcal{A}_1$ implicitly defines $d^* = \frac{1-h}{R_x^*}$ where $R^* = (R_x^*, R_y^*)$. We remark that $\mathcal{A}_1$ does not explicitly know h but can compute $hG = sR - R_xP$ since $\frac{h}{s}G + \frac{R_x}{s}P = R$ holds and R is uniquely determined from (R_x, v). $\mathcal{A}_1$ computes P^* such that

$$P^* = \frac{1}{R_x^*}G - \frac{1}{R_x^*}(sR - R_xP) = \frac{1}{R_x^*}G - \frac{h}{R_x^*}G = \frac{1-h}{R_x^*}G = d^*G$$

$\mathcal{A}_1$ outputs $(\mathsf{addr}^*, s^*, R_x^*, v^*)$ where $\mathsf{addr}^* = \mathcal{B}_{96..255}(H(P^*))$ and v^* is determined from $R^* = (R_x^*, R_y^*)$. Here, $0 < s^* < q/2 + 1$ holds. Moreover, $R^* = (R_x^*, R_y^*)$ is uniquely determined from (R_x^*, v^*), and P^* can be recovered such that

$$\frac{s^*}{R_x^*}(R^* - \frac{h}{s^*}G) = \frac{1}{R_x^*}\frac{1}{r^*}(r^*G - r^*hG) = \frac{1-h}{R_x^*}G = P^*$$

and thus $\mathsf{addr}^* = \mathcal{B}_{96..255}(\frac{s^*}{R_x^*}(R^* - \frac{h}{s^*}G))$ holds. That is, the verification algorithm outputs 1. Here, the probability that $P = P^*$ holds is negligible since M and d are randomly chosen and $H(M) = 1 - dR_x^*$ holds with negligible probability. □

Similarly, ECDSA also does not provide wNR. The attack is essentially the same as above: (s^*, R_x^*) is a valid signature under P^* since $\frac{h}{s^*}G + \frac{R_x^*}{s^*}P^* = r^*G$ holds.

Is wNR Mandatory in Ethereum? First of all, we would like to claim that the above attack is not a realistic threat in Ethereum due to the following reasons. In the actual usage, the verifier knows both σ and M since M is a transaction. More precisely, for addr, let the address holder who has $\mathsf{sk} = (d, P)$ issue a transaction M with a signature σ. To verify σ, the verifier knows both addr and M. Then, the verifier recovers P from σ and M and checks $\mathsf{addr} = \mathcal{B}_{96..255}(H(P))$ holds or not. Thus, it is quite unnatural to assume that adversaries just know σ without knowing the corresponding transaction M. When M is opened as a transaction, anyone can re-sign M by introducing a new key pair $(\mathsf{vk}^*, \mathsf{sk}^*)$ and run $\mathsf{Sign}(\mathsf{sk}^*, M)$. This is inevitable even if the signature scheme provides wNR. Thus, even if KR-ECDSA can provide wNR with some additional costs, e.g., via the BUFF transformation, this provides only limited benefits.

In our wNR attack, adversaries produce addr^*. This might be a natural situation since adversaries produce $(\mathsf{vk}^*, \sigma^*)$ in the definition of wNR, addr^* is a part of σ^*, an address holder usually produces a signature, and addr^* is determined by vk^*. Here, we pay attention to the fact that addr is stored on the blockchain. Again, addr has the same roles of appending $H(\mathsf{vk})$ to a signature as in PS-2: $(\mathsf{Sign}(\mathsf{sk}, m), H(\mathsf{vk}))$ and provides S-CEO and S-DEO. We observe that there is a difference between the case that addr is stored on the blockchain and the case that $H(\mathsf{vk})$ is just added to a signature such as PS-2. To clear the difference, we borrow the explanation for the reason why the PS-2 transformation does not provide NR given by Cremers et al. [8]:

> "*NR is in general not achieved since the signature of the original scheme σ may contain the message directly, allowing the adversary to re-sign this message under a new key*".

That is, if an adversary can extract m from $\mathsf{Sign}(\mathsf{sk}, m)$, then the adversary generates a new key pair $(\mathsf{vk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}(1^\lambda)$ and outputs $(\mathsf{Sign}(\mathsf{sk}', m), H(\mathsf{vk}'))$. This attack is possible in the case that $H(\mathsf{vk})$ is just added to a signature. However, when addr is stored on the blockchain, introducing other $H(\mathsf{vk}')$ is prevented due to the immutability of the blockchain. From this perspective, the definition of wNR where adversaries produce addr^* seems too much in the Ethereum context.

To sum up, wNR is unnecessary in the Ethereum context from our understanding. We emphasize that we never deny the meaningfulness of NR in other contexts such as [13,25]. For example, Jackson et al. [25] showed that the Dynamically Recreatable Key protocol [28] is insecure if the signature scheme used in the protocol does not provide NR.

5 On ECDSA-based Adaptor Signatures

In this section, we show that our S-DEO attack against the original ECDSA given in Section 4.2 also works against Aumayr et al.'s ECDSA-based adaptor signature scheme [2]. We emphasize that the attack is positioned outside of their security model. Our attack is currently theoretical in the sense that we did not find a vulnerability in the applications of adaptor signatures, atomic swap, payment channels [33], private coin mixing [32, 35], and oracle-based payments [32]. However, it would be meaningful to explicitly give our attack here because, potentially, our S-DEO attack may cause an issue in an application of adaptor signatures. We remark that the definitions given by Gerhart et al. [22], which provides the strongest security definitions of adaptor signatures so far (to the best of our knowledge), do not consider BUFF security.

Let Rel be a hard relation. We denote $(Y, y) \in \mathsf{Rel}$ for a statement Y and a witness y. An adaptor signature $\mathsf{AS} = (\mathsf{pSign}, \mathsf{pVerify}, \mathsf{Adapt}, \mathsf{Ext})$ w.r.t. a signature scheme $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ and a hard relation Rel consists of four algorithms defined as follows. Let $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$. Dai et al. [10] defined unlinkability. It guarantees that an adapted signature and a signature generated by the Sign algorithm are indistinguishable and Gerhart et al. [22] also employed unlinkability. Liu et al. [31] defined witness hiding that is implied by unlinkability. Ciampi et al. [6] followed the Liu et al. definition. Due to the situation, unlinkability is a natural requirement and thus we use the same notation σ as an adapted signature as in a signature generated by the Sign algorithm.

Definition 7 (Adaptor Signatures)

$\mathsf{pSign}(\mathsf{sk}, M, Y)$ *: The pre-signature algorithm takes a signing key* sk*, a message* M *to be signed, and a statement* Y *as input, and outputs a pre-signature* $\tilde{\sigma}$*.*

$\mathsf{pVerify}(\mathsf{vk}, M, \tilde{\sigma}, Y)$ *: The verification algorithm for a pre-signature takes a verification key* vk*,* M*,* $\tilde{\sigma}$*, and* Y *as input, and outputs 0 or 1.*

$\mathsf{Adapt}(\mathsf{vk}, M, \tilde{\sigma}, y)$ *: The adaption algorithm takes* vk*,* M*,* $\tilde{\sigma}$*, and a witness* y *as input, and outputs an adapted signature* σ*.*

$\mathsf{Ext}(\mathsf{vk}, M, Y, \tilde{\sigma}, \sigma)$ *: The extraction algorithm takes* vk*,* M*,* Y*,* $\tilde{\sigma}$*, and* σ*, and outputs* y *or* $\perp$*.*

Next, we define a S-DEO security for adaptor signatures. Since ECDSA with the format checking is not S-DEO secure, we consider a S-DEO security for pre-signatures. Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. We emphasize that the following definition is weak (but sufficient to show our attack) in the sense that $\mathcal{A}$ is not allowed to send a pre-signature query ($\mathcal{C}$ chooses a message M and sends the pre-signature $\tilde{\sigma}$ to $\mathcal{A}$) and $\mathcal{A}$ is also not allowed to issue a signing query. First, $\mathcal{C}$ generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives vk to $\mathcal{A}$. $\mathcal{C}$ chooses M and $(Y, y) \in \mathsf{Rel}$, and sends $(M, Y, \tilde{\sigma})$ to $\mathcal{A}$ where $\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, M, Y)$. $\mathcal{A}$ outputs (vk^*, M^*) where $\mathsf{vk}^* \neq \mathsf{vk}$, $M^* \neq M$, and $\mathsf{pVerify}(\mathsf{vk}^*, M^*, \tilde{\sigma}, Y) = 1$.

Aumayr et al. [2] proposed an adaptor signature w.r.t. ECDSA with format checking (they called the scheme the positive ECDSA scheme). Since we focus on the ECDSA-based adaptor signature

scheme, for $\mathbb{G} = E(\mathbb{F}_p)$ where $|E(\mathbb{F}_p)| = q$, we define three elliptic curve oriented relations employed in the scheme as follows: $\mathsf{Rel}_G : \mathbb{G} \times \mathbb{Z}_q$ is defined as

$$\mathsf{Rel}_G := \{(Y, y) : Y = yG\}$$

Rel'_G is defined as

$$\mathsf{Rel}'_G := \{((Y, \pi_Y), y) : (Y, y) \in \mathsf{Rel}_G \wedge \mathsf{V}_G(Y, \pi_Y) = 1\}$$

where π_Y is a non-interactive zero-knowledge proof of knowledge for y satisfying $Y = yG$ and $(\mathsf{P}_G, \mathsf{V}_G)$ is a proof system (i.e., a prover and a verifier) for Rel_G. $\mathsf{Rel}_{(Y,G)} : \mathbb{G}^2 \times \mathbb{Z}_q$ is defined as

$$\mathsf{Rel}_{(Y,G)} := \{((\tilde{R}, R), r) : \tilde{R} = rY \wedge R = rG\}$$

Here, $(\mathsf{P}_{(P,G)}, \mathsf{V}_{(P,G)})$ is a proof system for $\mathsf{Rel}_{(Y,G)}$. Aumayr et al. employed a function f. We explicitly define f that takes a point R and outputs the x-coordinate R_x. Let $\sigma = (s, R_x)$ be a ECDSA signature, $\mathsf{vk} = P$ and $\mathsf{sk} = d$ such that $P = dG$. We remark that we sometime denote y with the meaning "the y-coordinate of a point". We describe the Aumayr et al. adaptor signature scheme as follows.

Aumayr et al. ECDSA-based Adaptor Signature Scheme.

$\mathsf{pSign}(\mathsf{sk}, M, (Y, \pi_Y))$: Parse $\mathsf{sk} = d$. Choose $r \xleftarrow{\$} \mathbb{Z}_q$. Compute $h = H(M)$, $R = rG$, $\tilde{R} = rY$ which we denote $\tilde{R} = (\tilde{R}_x, \tilde{R}_y)$, $\tilde{s} = \frac{h + d\tilde{R}_x}{r} \bmod q$. If $\tilde{s} > q/2 + 1$, then set $\tilde{s} := -\tilde{s} \bmod q$. Compute $\pi \leftarrow \mathsf{P}_{(Y,G)}((\tilde{R}, R), r)$. Output $\tilde{\sigma} = (\tilde{s}, \tilde{R}_x, \tilde{R}, \pi)$.

$\mathsf{pVerify}(\mathsf{vk}, M, \tilde{\sigma}, (Y, \pi_Y))$: Parse $\mathsf{vk} = P$. If $\mathsf{V}_G(Y, \pi_Y) = 0$, then output 0. If $\tilde{s} \leq q/2 + 1$ does not hold, then output 0. Compute $h = H(M)$ and $R = \frac{h}{\tilde{s}}G + \frac{\tilde{R}_x}{\tilde{s}}P$. If $\mathsf{V}_{(Y,G)}((\tilde{R}, R), \pi) = 0$, then output 0. Output 1 if $(\tilde{R})_x = \tilde{R}_x$ and 0 otherwise.

$\mathsf{Adapt}(\mathsf{vk}, M, \tilde{\sigma}, y)$: Parse $\tilde{\sigma} = (\tilde{s}, \tilde{R}_x, \tilde{R}, \pi)$. Compute $s = \tilde{s}y^{-1} \bmod q$ and output $\sigma = (s, \tilde{R}_x)$.

$\mathsf{Ext}(\mathsf{vk}, M, (Y, \pi_Y), \tilde{\sigma}, \sigma)$: Parse $\tilde{\sigma} = (\tilde{s}, \tilde{R}_x, \tilde{R}, \pi)$ and $\sigma = (s, \tilde{R}_x)$. Compute $y = s^{-1}\tilde{s} \bmod q$. Output y If $((Y, \pi_Y), y) \in \mathsf{Rel}'_G$, and $\perp$, otherwise.

Our S-DEO attack is described as follows. $\mathcal{C}$ chooses $d \in \mathbb{Z}_q$ and computes $P = dG$ and (Y, π_Y). $\mathcal{C}$ chooses M and sends $(M, (Y, \pi_Y), \tilde{\sigma})$ to $\mathcal{A}$ where $\tilde{\sigma} = (\tilde{s}, \tilde{R}_x, \tilde{R}, \pi)$ is a pre-signature on M. $\mathcal{A}$ chooses $M^* \neq M$ and computes $h^* = H(M^*)$, $R = \frac{h}{\tilde{s}}G + \frac{\tilde{R}_x}{\tilde{s}}P$, and

$$P^* := \frac{\tilde{s}}{\tilde{R}_x}(R - \frac{h^*}{\tilde{s}}G)$$

Here $\mathsf{pVrfy}(P^*, M^*, \tilde{\sigma}, (Y, \pi_Y)) = 1$ holds since $\mathsf{V}_G(Y, \pi_Y) = 1$ and $0 < \tilde{s} < q/2 + 1$ hold and $\mathsf{V}_{(Y,G)}((\tilde{R}, R), \pi) = 1$ also holds since

$$\frac{h^*}{\tilde{s}}G + \frac{\tilde{R}_x}{\tilde{s}}P^* = \frac{h^*}{\tilde{s}}G + \frac{\tilde{R}_x}{\tilde{s}}(\frac{\tilde{s}}{\tilde{R}_x}(R - \frac{h^*}{\tilde{s}}G)) = R$$

hold. $\mathcal{A}$ outputs (P^*, M^*) and breaks the S-DEO security.

The main reason why the attack works is that the format checking does not affect the attack. Due to the format checking, our S-CEO attack given in Section 4.2 and the MBS attack given in Section 4.3 do not work since they compute $-R$ in the verification procedure. We remark that this

fact does not formally prove that the Aumayr et al. adaptor signature scheme is S-CEO and MBS secure.

As a subsequent work of Aumayr et al., Tu et al. [39] proposed two ECDSA-based adaptor signature schemes which they called $\mathsf{ECDSA\text{-}AS}_{sk}$ and $\mathsf{ECDSA\text{-}AS}_{wit}$. They focused on the fact that the pSign algorithm above computes $\pi \leftarrow \mathsf{P}_{(Y,G)}((\tilde{R}, R), r)$ for a random r chosen in the algorithm. That is, π is not pre-computable. In the $\mathsf{ECDSA\text{-}AS}_{sk}$ scheme, $\mathsf{sk} = d$ is regarded as the witness instead of r. Compute $Z = dY$ and a proof π of d satisfying $P = dG$ and $Z = dY$. Here, π is pre-computable (i.e., the signer can compute π offline). The $\mathsf{pVerify}$ algorithm checks the validity of π and checks whether or not the x-coordinate of $\frac{h}{\tilde{s}}Y + \frac{\tilde{R}_x}{\tilde{s}}Z$ and $\tilde{R}_x$ are the same. Obviously, our S-DEO attack does not work since the validity check of π prevents to use $P^* \neq P$. In the $\mathsf{ECDSA\text{-}AS}_{wit}$ scheme, compute $Z = dY$ and a proof π of y satisfying $Y = yG$ and $Z = yP$. Again, π is pre-computable. The $\mathsf{pVerify}$ algorithm checks the validity of π and checks whether or not the x-coordinate of $\frac{h}{\tilde{s}}Y + \frac{\tilde{R}_x}{\tilde{s}}Z$ and $\tilde{R}_x$ are the same. Our S-DEO attack does not work since $(\log_G Y)P^* \neq Z$. Though this fact does not formally prove that the Tu et al. adaptor signature schemes are S-DEO secure, the construction methodology to reduce online computations may be applicable to provide S-DEO security.

As a building block of BlindHub, Qin et al. [35] introduce a blind adaptor signature scheme based on Aumayr et al.'s ECDSA-based adaptor signature scheme. Next, we show that our S-DEO attack works against the blind adaptor signature scheme as well. Again, we emphasize that the attack is positioned outside of their security model. Basically, the blind adaptor signature scheme is the same as that of the adaptor signature scheme except the interactive signing protocol between a user and a signer as follows.

Qin et al.'s ECDSA-based Blind Adaptor Signature Scheme (Signing).

User. Compute $h = H(M)$ and a proof-of-knowledge π_h of the pre-image of h. Send (h, π_h) to the signer who manages the verification key P.

Signer. If the proof π_h is not correct, then abort. Otherwise, basically the signer runs the original pre-signing algorithm: Choose $r \xleftarrow{\$} \mathbb{Z}_q$. Compute $R = rG$, $\tilde{R} = rY$ which we denote $\tilde{R} = (\tilde{R}_x, \tilde{R}_y)$, $\tilde{s} = \frac{h + d\tilde{R}_x}{r} \bmod q$. If $\tilde{s} > q/2 + 1$, then set $\tilde{s} := -\tilde{s} \bmod q$. Compute $\pi \leftarrow \mathsf{P}_{(Y,G)}((\tilde{R}, R), r)$. Return $\tilde{\sigma} = (\tilde{s}, \tilde{R}_x, \tilde{R}, \pi)$ to the user.

User. Abort if $\tilde{s} = 0$ or $\mathsf{pVerify}(P, M, \tilde{\sigma}, (Y, \pi_Y)) = 0$.

Since $\tilde{\sigma} = (\tilde{s}, \tilde{R}_x, \tilde{R}, \pi)$ is the same as that of the original adopter signature scheme, the user can produce (P^*, M^*) that $\mathsf{pVerify}(P^*, M^*, \tilde{\sigma}, (Y, \pi_Y)) = 1$ holds and $P^* \neq P$ and $M^* \neq M$. We note that an adversary (user) chooses M unlike the S-DEO attack for Aumayr et al.'s ECDSA-based adaptor signature scheme, due to the syntax of blind signatures. Clarifying an impact of the S-DEO attack (and other BUFF security) on the blind adaptor signature scheme and applications of adapt-r signatures are left as future work of this paper.

6 Remark on Domain Parameter Shifting Attack

Vaudenay explains the domain parameter shifting attack [41] as follows: *an adversary tries to modify the subgroup generator G to modify the elliptic curve a and b domain parameters.* Here, we also consider the case that G is modified to another point G' on the same elliptic curve. As a concrete example, we introduce CVE-2020-0601 reporting that a spoofing vulnerability exists in

Windows CryptoAPI (Crypt32.dll).[6] The attack behind is a domain parameter shifting attack: for $\mathsf{vk} = P = dG$, an adversary chooses a random $t \in \mathbb{Z}_q$ and sets $G' = \frac{1}{t}P$. The adversary, who does not know $\mathsf{sk} = d$, can produce a valid ECDSA signature (s, R_x) on arbitrary M under P with the forged base point G' such that $h = H(M)$, $r \xleftarrow{\$} \mathbb{Z}_p$, $R = rG' = (R_x, R_y)$, and $s = \frac{h+tR_x}{r} \bmod q$. Then, $\sigma = (s, R_x)$ is a valid ECDSA signature under P with G'. Since P can be recovered by $P = \frac{s}{R_x}(R - \frac{h}{s}G')$, the attack works for KR-ECDSA.

This attack can be prevented in the BUFF security context by simply defining $\mathsf{vk} = (G, P)$. Then, adding $H(\mathsf{vk})$ to a signature as in PS-2 prevents to produce another base point. We remark that the modification of verification key form does not fit KR-ECDDA since Ethereum address is defined as $\mathcal{B}_{96..255}(H(P))$. Thus, the domain parameter checking is mandatory in KR-ECDSA even if it provides S-CEO, S-DEO, M-S-UEO, and MBS.

7 Conclusion

In this paper, we analyzed the security of KR-ECDSA and showed that KR-ECDSA provides strong EUF-CMA security, S-CEO, S-DEO, M-S-UEO, and MBS but does not provide wNR. Since the original ECDSA does not provide S-CEO, S-DEO, and MBS, KR-ECDSA is more secure than the original ECDSA in the sense of BUFF security. We also show that our S-DEO attack against the original ECDSA works against Aumayr et al.'s ECDSA-based adaptor signature scheme and Qin et al.'s blind adaptor signature scheme. Struck and Weishäupl [38] introduced a general framework of binding properties that encompasses existing BUFF security. Considering their extended BUFF security in the (KR-)ECDSA context is also an interesting future work.

Though BUFF security has been somewhat overlooked in enhanced signatures (such as (blind) adaptor signatures as mentioned in this paper), some results have been reported. Fujita et al. [20] analyzed aggregate signatures and multi signatures in terms of the exclusive ownership context. They also mentioned that opening soundness introduced in [36] considered exclusive ownership in the group signature context. Fischlin et al. [19] considered BUFF security in the threshold signature context. They proposed a generic compiler that transforms any threshold signature scheme to satisfy BUFF security. Using the compiler, threshold ECDSA protocols (e.g., [9, 11, 12, 21, 30, 43]) provide BUFF security even if ECDSA does not provide BUFF security. Providing such a compiler for adaptor signatures is left as future work of this paper.

Acknowledgements. This work was supported by JSPS KAKENHI Grant Numbers JP21K11897 and JP25H01106.

References

[1] Thomas Aulbach, Samed Düzlü, Michael Meyer, Patrick Struck, and Maximiliane Weishäupl. Hash your keys before signing: BUFF security of the additional NIST PQC signatures. In *Post-Quantum Cryptography*, pages 301–335, 2024.

[2] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In *ASIACRYPT*, pages 635–664, 2021.

[3] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In *CHES*, pages 124–142, 2011.

[6] `https://nvd.nist.gov/vuln/detail/CVE-2020-0601`

[4] Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. The provable security of Ed25519: Theory and practice. In *IEEE S&P*, pages 1659–1676, 2021.

[5] Daniel R. L. Brown. Generic groups, collision resistance, and ECDSA. *Designs, Codes and Cryptography*, 35(1):119–152, 2005.

[6] Michele Ciampi, Xiangyu Liu, Ioannis Tzannetos, and Vassilis Zikas. Universal adaptor signatures from blackbox multi-party computation. In *CT-RSA*, pages 375–398, 2025.

[7] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. *IACR Cryptology ePrint Archive*, page 1525, 2020. Version 1.4.1, October 2023.

[8] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In *IEEE S&P*, pages 1696–1714, 2021.

[9] Handong Cui, Kwan Yin Chan, Tsz Hon Yuen, Xin Kang, and Cheng-Kang Chu. Bandwidth-efficient zero-knowledge proofs for threshold ECDSA. *The Computer Journal*, 67(4):1265–1278, 2024.

[10] Wei Dai, Tatsuaki Okamoto, and Go Yamamoto. Stronger security and generic constructions for adaptor signatures. In *INDOCRYPT*, pages 52–77, 2022.

[11] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *IEEE S&P*, pages 1051–1066, 2019.

[12] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Threshold ECDSA in three rounds. In *IEEE S&P*, pages 3053–3071, 2024.

[13] Jelle Don, Serge Fehr, Yu-Hsuan Huang, Jyun-Jie Liao, and Patrick Struck. Hide-and-seek and the non-resignability of the BUFF transform. In *TCC*, pages 347–370, 2024.

[14] Jelle Don, Serge Fehr, Yu-Hsuan Huang, and Patrick Struck. On the (in)security of the BUFF transform. In *CRYPTO*, pages 246–275, 2024.

[15] Samed Düzlü, Rune Fiedler, and Marc Fischlin. Buffing FALCON without increasing the signature size. In *Selected Areas in Cryptography*, pages 131–150, 2024.

[16] Samed Düzlü and Patrick Struck. The role of message-bound signatures for the beyond unforgeability features and weak keys. In *ISC*, pages 61–80, 2024.

[17] Serge Fehr, Yu-Hsuan Huang, and Julia Kastner. Sandwich BUFF: achieving non-resignability using iterative hash functions. In *TCC*, pages 235–265, 2025.

[18] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the provable security of (EC)DSA signatures. In *ACM CCS*, 2016.

[19] Marc Fischlin, Aikaterini Mitrokotsa, and Jenit Tomy. BUFFing threshold signature schemes. In *Public-Key Cryptography*, pages 137–168, 2025.

[20] Yuuki Fujita, Yusuke Sakai, Kyosuke Yamashita, and Goichiro Hanaoka. On key substitution attacks against aggregate signatures and multi-signatures. In *ACM APKC*, pages 54–63, 2025.

[21] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *Applied Cryptography and Network Security*, pages 156–174, 2016.

[22] Paul Gerhart, Dominique Schröder, Pratik Soni, and Sri Aravinda Krishnan Thyagarajan. Foundations of adaptor signatures. In *EUROCRYPT*, pages 161–189, 2024.

[23] Jens Groth and Victor Shoup. On the security of ECDSA with additive key derivation and presignatures. In *EUROCRYPT*, pages 365–396, 2022.

[24] Dominik Hartmann and Eike Kiltz. Limits in the provable security of ECDSA signatures. In *TCC*, pages 279–309, 2023.

[25] Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse. Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In *ACM CCS*, pages 2165–2180, 2019.

[26] Jonas Janneck. Bird of prey: Practical signature combiners preserving strong unforgeability. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT, Part II*, pages 222–250, 2026.

[27] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA). `https://web.archive.org/web/20170921160141/http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf`.

[28] Tiffany Hyun-Jin Kim, Cristina Basescu, Limin Jia, Soo Bum Lee, Yih-Chun Hu, and Adrian Perrig. Lightweight source authentication and path validation. In *ACM SIGCOMM*, pages 271–282, 2014.

[29] Mukul Kulkarni and Keita Xagawa. Strong existential unforgeability and BUFF securities of mpc-in-the-head signatures. *Des. Codes Cryptogr.*, 94(7):159, 2026.

[30] Yehuda Lindell. Fast secure two-party ECDSA signing. In *CRYPTO*, pages 613–644, 2017.

[31] Xiangyu Liu, Ioannis Tzannetos, and Vassilis Zikas. Adaptor signatures: New security definition and a generic construction for NP relations. In *ASIACRYPT*, pages 168–193, 2024.

[32] Varun Madathil, Sri Aravinda Krishnan Thyagarajan, Dimitrios Vasilopoulos, Lloyd Fournier, Giulio Malavolta, and Pedro Moreno-Sanchez. Cryptographic oracle-based conditional payments. In *NDSS*. The Internet Society, 2023.

[33] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and state channels: Payment networks that go faster than lightning. In *Financial Cryptography and Data Security*, pages 508–526, 2019.

[34] Thomas Pornin and Julien P. Stern. Digital signatures do not guarantee exclusive ownership. In *Applied Cryptography and Network Security*, pages 138–150, 2005.

[35] Xianrui Qin, Shimin Pan, Arash Mirzaei, Zhimei Sui, Oguzhan Ersoy, Amin Sakzad, Muhammed F. Esgin, Joseph K. Liu, Jiangshan Yu, and Tsz Hon Yuen. BlindHub: Bitcoin-compatible privacy-preserving payment channel hubs supporting variable amounts. In *IEEE S&P*, pages 2462–2480, 2023.

[36] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. On the security of dynamic group signatures: Preventing signature hijacking. In *Public Key Cryptography*, pages 715–732, 2012.

[37] Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In *CRYPTO*, pages 93–110, 2002.

[38] Patrick Struck and Maximiliane Weishäupl. A framework for advanced signature notions. In *Public-Key Cryptography, Part II*, pages 156–188, 2026.

[39] Binbin Tu, Min Zhang, and Chen Yu. Efficient ECDSA-based adaptor signature for batched atomic swaps. In *ISC*, pages 175–193, 2022.

[40] Serge Vaudenay. The security of DSA and ECDSA. In *Public Key Cryptography*, pages 309–323, 2003.

[41] Serge Vaudenay. Digital signature schemes with domain parameters: Yet another parameter issue in ECDSA. In *ACISP*, pages 188–199, 2004.

[42] Gavin Wood. Ethereum yellow paper (Shanghai version 9fde3f4-2024-09-02), 2024. `https://ethereum.github.io/yellowpaper/paper.pdf`.

[43] Guy Zyskind, Avishay Yanai, and Alex Pentland. Unstoppable wallets: Chain-assisted threshold ECDSA and its applications. In *ASIACCS*. ACM, 2024.