Practical Adaptor Signatures: Security and Applications*

Pavel Hubáček[1,2], Kristýna Mašková[1,2], and Berenika Richterová[3]

[1] Institute of Mathematics, Czech Academy of Sciences, Prague, Czech Republic
{hubacek,maskova}@math.cas.cz
[2] Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic
[3] Eindhoven University of Technology, Eindhoven, Netherlands
b.richterova@tue.nl

Abstract. Adaptor signatures allow a signer to publish a pre-signature that can be transformed into a valid signature by anyone once a secret witness is learned. Poelstra first suggested this primitive to bypass the limited scripting capabilities of Bitcoin. These schemes were later formalized by Aumayr et al. (ASIACRYPT 2021) and refined by Dai et al. (INDOCRYPT 2022) and Gerhart et al. (EUROCRYPT 2024). In this work, we revisit the constructions of adaptor signatures deployed in practice without any formal proof of security.

First, we demonstrate that the current security model does not capture the ECDSA adaptor signature that underlies most real-world systems. Second, we propose a relaxed definition and prove that it is satisfied by the ECDSA adaptor construction under the strong unforgeability of ECDSA. Finally, focusing on oracle-based conditional payments, we formulate the first security model for adaptor-based Discreet Log Contracts (DLCs) and show that our relaxed notion suffices for their security.

Keywords: Adaptor Signatures · ECDSA · Discreet Log Contracts.

* This is the full version of the paper published in the proceedings of SCN 2026.

Table of Contents

1 Introduction . . . 3
1.1 Our Contributions . . . 3
1.2 Related Work . . . 4
1.3 Technical Overview . . . 4
2 Preliminaries . . . 6
2.1 Hard Relations and Non-Interactive Zero-Knowledge . . . 6
2.2 Adaptor Signatures . . . 8
3 Robust Security of Adaptor Signatures . . . 12
3.1 Limits on Robust Security of Adaptor Signatures . . . 12
4 Security of ECDSA Adaptor Signatures . . . 15
4.1 Unforgeability of ECDSA Adaptor Signatures . . . 15
4.2 Witness Extractability of ECDSA Adaptor Signatures . . . 24
4.3 Pre-Signature Adaptability of ECDSA Adaptor Signatures . . . 29
4.4 Pre-Signature Unforgeability of ECDSA Adaptor Signatures . . . 30
4.5 Extractability of ECDSA Adaptor Signatures . . . 34
4.6 Witness Hiding of ECDSA Adaptor Signatures . . . 38
5 Relative Efficiency . . . 38
5.1 Efficiency of Adaptor Signatures for Practical Signature Schemes . . . 39
6 Adaptor-Based Discreet Log Contracts . . . 40
6.1 Defining DLCs . . . 40
6.2 DLCs from Adaptor Signatures . . . 42
6.3 Security of Discreet Log Contracts . . . 43

1 Introduction

Digital signatures are a cornerstone of modern cryptography, providing authentication, integrity, and non-repudiation for virtually every security-critical protocol. Adaptor signatures (AS), first articulated by Poelstra [Poe] to enable advanced protocols for cryptocurrencies with limited expressivity of smart contracts, extend this functionality by binding a signature to a hard relation. A signer outputs a publicly verifiable "pre-signature" that simultaneously (i) commits to a valid signature on a given message and (ii) hides a witness for the relation. Anyone who later learns a valid witness can adapt the pre-signature into a standard signature, while the pair of artifacts (pre-signature, signature) reveals the witness to the original signer. This duality has made adaptor signatures a powerful primitive for enforcing conditional transfers in blockchain systems, where they offer dramatic fee savings and stronger privacy compared to traditional hash-lock techniques. Applications include payment channel networks [MMS+19, DW15, MBB+19], private coin mixing [GMM+22, QPM+23], and oracle-based conditional payments [GKK22, MTV+23].

Because parties get access to both pre-signatures and signatures, the security of adaptor signatures must encompass more than, e.g., classical unforgeability. Aumayr et al. [AEE+21] were the first to study the security of adaptor signatures; they formalized adaptor unforgeability, pre-signature adaptability, and witness extractability, but recent analyses have identified subtle gaps. Gerhart et al. [GSST24] showed that schemes proven secure under the original definition can still break higher-level protocols such as private coin-mixing and oracle-based payments. Independently, Liu, Tzannetos, and Zikas [LTZ24] observed that the required properties do not prevent the witness from leaking through the adapted signature itself, and introduced the *witness-hiding property* to close this loophole.

1.1 Our Contributions

In this work, we revisit the security of adaptor signatures as deployed in practice; specifically, we study the ECDSA adaptor signatures [MSK18, Fou19]. Although Schnorr adaptor signatures are both theoretically and technically preferable, ECDSA adaptor signatures dominate practical implementations due to legacy constraints in systems like Bitcoin. Despite their adoption in practice, the ECDSA adaptor signature currently lacks a formal proof of security.

Original vs. extended ECDSA adaptor signatures. Aumayr et al. [AEE+21] initiated the formal study of adaptor signatures, analyzing constructions for both Schnorr and ECDSA. While they proved the security of Schnorr adaptor signatures, they did not resolve the security of the ECDSA adaptor proposed by Fournier [Fou19] (tracing back to Moreno-Sanchez and Kate [MSK18]). Instead, they proposed a more complex *extended variant* that augments Fournier's construction with an additional NIZK *proof of knowledge*. They proved the security of this extended variant in the ROM, assuming the strong unforgeability of ECDSA and relying on online-extractable NIZK proofs of knowledge. To the best of our knowledge, the security of the original variant, which underpins most real-world applications (including [Lav, 101, Bas, Eig, Ato, Far, P2P]), remains unproven in the literature.

We present the first proof of security for the original ECDSA adaptor signatures [Fou19]. Additionally, we formalize and prove the security of a deployed protocol for oracle-based conditional payments over Bitcoin that leverages these signatures. To this end, we present the following set of results.

1. *Refined definition of security for ECDSA-based AS:* We formalize Fournier's observation [Fou19] that unrestricted access to pre-signatures enables a *security degradation* (specifically, a reduction in bit-security) on curves like secp256k1, demonstrating that previous security definitions are incompatible with the original ECDSA adaptor signatures. Motivated by this, we propose a new set of security definitions that restrict the pre-signing oracle to meaningfully capture the security of ECDSA adaptors.
2. *Proof of security for ECDSA adaptor signatures:* We prove that the original ECDSA adaptor signature [Fou19] satisfies our refined notions of security. Our proofs share a structural similarity with those for the extended variant by Aumayr et al. [AEE+21], as their underlying strategy is natural. Crucially, however, we leverage our refined definitions to avoid the necessity of extracting witnesses from the adversary in the reduction, thereby removing the need for the additional NIZK proofs of knowledge.

3. *Formalization and proof of security for ECDSA-based DLCs:* We provide the first formal definition of adaptor-based Discreet Log Contracts (DLCs) [Dry17], a previously informal mechanism for oracle-based conditional payments on Bitcoin. This formalization enables us to articulate precise security notions for DLCs. We prove that existing constructions based on the original ECDSA adaptor signatures satisfy these notions, confirming their soundness for practical deployment.

1.2 Related Work

Close to our motivation is the work of Tu, Zhang, and Yu [TZY22], who suggested ways to address the overhead introduced by the additional NIZK proof of knowledge in the *extended* ECDSA adaptor signature of Aumayr et al. [AEE+21]. However, we analyze the original and deployed ECDSA adaptor signature.

Alternative constructions. Erwig et al. [EFH+21] presented a generic transformation of an identification-based signature scheme into an adaptor signature scheme. Dai, Okamoto, and Yamamoto [DOY22] identified gaps in the formulations of Aumayr et al. [AEE+21] and introduced extractability, unique extractability, and unlinkability as additional notions of security for adaptor signatures. Liu et al. [LTZ24] extended the work of Dai et al. [DOY22] by introducing the *witness hiding* property for adaptor signatures. Gerhart et al. [GSST24] constructed adaptor signatures for other classical signature schemes such as Camenisch-Lysyanskaya or Boneh-Boyen-Shacham signatures. There are also proposals for adaptor signatures with plausible resiliency to quantum attacks [EEE20, TMM21]. Baecker et al. [BGKS25] recently proposed an extension of adaptor signatures to the threshold setting, where a group of shareholders of a witness w and a group of shareholders of a signing key sk wish to fairly exchange w for a signature on a transaction under sk.

Oracle-based conditional payments. Discreet Log Contracts were proposed by Dryja [Dry17] and later formalized generically based on adaptor signatures. Le Guilly, Kohen, and Kuwahara [GKK22] presented various theoretical optimizations of DLCs based on ECDSA but did not provide a security proof for DLCs. A Rust implementation of DLCs maintained by Le Guilly and others [Rus] has been used by various practical projects [101, Ato, Lav, Bas, Eig, Far, P2P]. Madathil et al. [MTV+23] presented a more complex construction for oracle-based conditional payments that achieves strong security guarantees in the presence of multiple oracles attesting simultaneously. However, we are unaware of any deployment of their construction in practice. Based on the universal compatibility framework, Tairi, Moreno-Sanchez, and Schneidewind [TMS23] presented a general framework for analyzing blockchain applications based on adaptor signatures.

1.3 Technical Overview

The Power of the Pre-Signing Oracle. In a security experiment modeling adaptor signatures, the adversary has access to a pre-signing oracle to model its access to multiple pre-signatures on various messages. When defining a specific security notion, we must decide how to restrict the adversary's access to pre-signatures. A conservative choice, taken by Aumayr et al. and follow-up works, is to let the adversary request pre-signatures on arbitrary messages w.r.t. arbitrary instances of the hard relation. Such a choice leads to a robust security notion. However, it also makes the proof of security significantly more challenging. Specifically, when designing a reduction to, e.g., the strong unforgeability of the underlying signature scheme, we must be able to answer such an unrestricted adversary's queries to the pre-signature oracle, despite only having access to the signing oracle for the underlying signature scheme.

For Schnorr adaptor signatures, Aumayr et al. managed to simulate unrestricted pre-signature queries given access to a Schnorr signing oracle. For the ECDSA adaptor signature, however, it is unclear how to simulate a pre-signature w.r.t. an arbitrary instance Y of DLOG.[4] If the reduction knows the discrete logarithm of Y, it can produce a pre-signature distributed identically to a genuine pre-signature from a single ECDSA signature obtained from the signing oracle. This motivated Aumayr et al. to alter the hard relation

[4] Aumayr et al. also explicitly state the issue with simulating pre-signing queries for the original ECDSA adaptor; cf. the discussion in Section 5.1 of their work.

so that its instances are pairs (Y, π), where Y is a DLOG instance and π is a NIZK proof of knowledge of the discrete logarithm of Y. The pre-signature oracle can then reject queries containing invalid proofs π for Y. Intuitively, this prevents the adversary from submitting a pre-signature query "without knowing" the discrete logarithm of Y. More importantly, due to the (online) proof-of-knowledge property, the reduction can extract the corresponding DLOG solution for Y from an accepting proof π and simulate the pre-signature. There are two downsides to the above approach. First, the scheme becomes significantly more complex because we extend the relation with an online-extractable NIZK-PoK of the discrete logarithm, which introduces considerable overhead [Fis05]. Second, it leaves the security of the original ECDSA adaptor from [Fou19] unresolved.

The Inadequacy of Unrestricted Access to ECDSA Pre-Signatures. As it turns out, we cannot allow unrestricted access to the pre-signing oracle for the ECDSA adaptor signature. Fournier [Fou19] already observed that it is possible to efficiently simulate a fixed Diffie-Hellman (FDH) oracle w.r.t. the ECDSA public key considered in the security experiment. He concluded that one should try to prove the security of the ECDSA adaptor signature in the presence of an oracle solving the Computational Diffie-Hellman problem. However, to the best of our knowledge, this approach was never realized as a formal proof. On the other hand, it is known that access to an FDH oracle might significantly speed up attacks on the DLOG problem in the group [BG04]. In Section 3.1, we estimate the complexity of the above vulnerability over the secp256k1 curve (used, e.g., in Bitcoin) when the FDH oracle is implemented via ECDSA pre-signing queries. Our estimate confirms that it would indeed contradict the conjectured bit-security of the DLOG problem on secp256k1. Thus, we conclude that we cannot allow unrestricted access to a pre-signing oracle when modelling the security of ECDSA adaptor signatures.

Security of Adaptor Signatures with a Restricted Pre-Signing Oracle. To resolve the issue with unrestricted access to the ECDSA pre-signing oracle, we present alternative definitions of extractability, adaptor unforgeability, witness extractability, and adaptability that restrict the adversary's access to pre-signatures. This leads to a weaker notion of security compared to the robust definitions from previous works. However, we can now use our new definitions for a *meaningful analysis* of the security of the original ECDSA adaptor signature scheme. Moreover, we show that security established against adversaries with restricted access to the pre-signing oracle is sufficient in the context of oracle-based conditional payments, as we discuss next.

Security of Discreet Log Contracts. The motivating application of adaptor signatures in [AEE+21] was mainly in blockchain payment channels. We study their applications in oracle-based conditional payments. Madathil et al. [MTV+23] proposed a scheme for this use case. While their scheme has a proof of security, the construction is complex and, to the best of our knowledge, has not seen adoption in practice. Discreet Log Contracts (DLCs) offer a simple, deployed alternative built on ECDSA adaptor signatures [GKK22].

At a high level, the core idea behind adaptor-based DLCs is to link transactions to events using pre-signatures. The setting consists of three roles: a sender, a receiver, and an oracle. Assume that the oracle attests to an event that can result in multiple outcomes. The sender uses the adaptor signature scheme to pre-sign transactions tx_o relevant to each of the possible outcomes o. The oracle publishes a random group element $R := rG$ ahead of the event, and for each outcome o, the sender computes a statement $T_o \in \mathbb{G}$, referred to as an *anticipation point*, as

$$T_o := R + P \cdot \mathsf{H}(P \parallel R \parallel o),$$

for a hash function H and the oracle's public key $P := pG$ corresponding to a secret key p. The anticipation points T_o connect each outcome and the corresponding transaction that should be carried out when the outcome occurs. This is done by the sender pre-signing each tx_o w.r.t. T_o and sending these pre-signatures to the receiver who verifies them. When the event occurs with the actual outcome o^*, the oracle then publishes a discrete logarithm t_{o^*} of T_{o^*}, since it knows both the sampled secret r and the secret key p. This enables the

receiver to adapt the corresponding pre-signature into a valid signature and publish the resulting transaction on the blockchain.

We formalize DLCs and introduce two security notions: *unforgeability* and *redeemability*. Unforgeability extends classic signature security, preventing adversaries from signing without knowledge of the secret key. Redeemability guarantees that any honest pre-signature can be redeemed once the oracle's secret is revealed.

Unforgeability follows from the extractability of the underlying adaptor signature scheme, while redeemability comes from pre-signature adaptability. Our proof of unforgeability additionally requires the One-More-Discrete-Logarithm ($\mathsf{OMDL}[1]$) assumption [BNPS03]. This assumption allows for a single call to a DLOG oracle, while requiring the adversary to solve two DLOG challenges. In our proof, we modify the unforgeability game to include an additional extraction check. It is not immediately clear whether the probability of success in the modified game differs negligibly from the original. To argue this, we would ideally rely on the hardness of the discrete logarithm problem. However, simulating the attestation oracle requires revealing a witness that we do not know. At that point, we rely on the one call to the DLOG oracle, as permitted by the $\mathsf{OMDL}[1]$ assumption.

2 Preliminaries

For a probability distribution D, we write $x \leftarrow D$ to denote sampling x from D. A finite set S in place of D denotes the uniform distribution on S. We denote the security parameter by λ. For an algorithm $\mathcal{A}$, $y \leftarrow \mathcal{A}(x)$ denotes that $\mathcal{A}$ outputs y on input x. If the algorithm $\mathcal{A}$ is specifically DPT (deterministic polynomial time), then we sometimes write $y := \mathcal{A}(x)$. We say that a function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ of the security parameter λ is negligible if it approaches zero faster than any inverse polynomial. In that case, we write $\varepsilon(\lambda) \in \mathsf{negl}(\lambda)$. We call a partial function $f\colon \{0,1\}^* \to \{0,1\}^*$ with domain $\mathcal{D}(f) \subseteq \{0,1\}^*$ a programming function. We denote by $\mathcal{O}|_f$ a random oracle restricted by the programming function, which is defined as

$$\mathcal{O}|_f(x) = \begin{cases} f(x) & \text{if } x \in \mathcal{D}(f), \\ \mathcal{O}(x) & \text{otherwise.} \end{cases}$$

2.1 Hard Relations and Non-Interactive Zero-Knowledge

Definition 1. *We say that $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is a* hard relation *if R is polynomial-time decidable, and there exists a PPT sampling algorithm GenR that, on 1^λ, outputs a pair $(S, w) \in R$ such that, for every PPT adversary $\mathcal{A}$,*

$$\Pr\left[(S, \mathcal{A}(S, 1^\lambda)) \in R \;\middle|\; (S, w) \leftarrow \mathsf{GenR}(1^\lambda)\right] \in \mathsf{negl}(\lambda).$$

Definition 2. *We define the* discrete logarithm relation *as*

$$\mathsf{DLOG} := \{((q, \mathbb{G}, G, Y), y) \mid Y = yG\},$$

where $Y \in \mathbb{G}$, $y \in \mathbb{F}_q$, and $\mathbb{G}$ is a description (including the corresponding operation) of a cyclic additive group of prime order q generated by G.

Technically, the statement of the relation DLOG is the complete tuple $(q, \mathbb{G}, G, Y)$. For simplicity of notation, we assume that the order q and the group description $\mathbb{G}$ are given by the generator G, and we consider only

$$\{((G, Y), y) \in \mathbb{G} \times \mathbb{G} \times \mathbb{F}_q \mid Y = yG\}$$

to be the statement-witness pairs of DLOG. Assume DLOG is hard as per Definition 1. Then no PPT adversary can output, with non-negligible probability, a witness for a DLOG statement generated by GenR. Because each statement specifies the group $\mathbb{G}$, every sample may lie in a different group. In practice, however, we fix one trusted group $\mathbb{G}$ and still expect hardness when sampling many elements $Y \in \mathbb{G}$. With a single fixed group, we lose asymptotic guarantees, so we split the sampling into two stages. A PPT algorithm PGen, on

$\mathsf{P}_{\mathsf{DLEQ}}((G_1, G_2, Y_1, Y_2), x)$	$\mathsf{V}_{\mathsf{DLEQ}}((G_1, G_2, Y_1, Y_2), \pi)$
1 : $v \leftarrow \mathbb{F}_q$	1 : $(c, z) := \pi$
2 : $T_1 := vG_1, T_2 := vG_2$	2 : $T_1 := zG_1 + cY_1, T_2 := zG_2 + cY_2$
3 : $c \leftarrow \mathsf{H}(G_1 \parallel G_2 \parallel Y_1 \parallel Y_2 \parallel T_1 \parallel T_2)$	3 : **return**
4 : $z := v - cx \pmod q$	$\quad c = \mathsf{H}(G_1 \parallel G_2 \parallel Y_1 \parallel Y_2 \parallel T_1 \parallel T_2)$
5 : **return** $\pi := (c, z)$	

Fig. 1: Proof of the equality of discrete logarithms [CP93].

input 1^λ, first selects a group $\mathbb{G}$ with generator G. A second PPT algorithm GenR, on input 1^λ, chooses $y \leftarrow \mathbb{F}_q^*$ uniformly and returns (yG, y). This view is stronger than Definition 1: for every group from PGen, computing the discrete logarithm of $Y \in \mathbb{G}$ sampled by GenR must still be infeasible with non-negligible probability.

Non-interactive zero-knowledge proofs. Constructions of ECDSA adaptor signatures rely on non-interactive zero-knowledge proofs (NIZKs) of equality of discrete logarithms. In general, NIZK proofs w.r.t. a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ allow proving that a statement s belongs to the associated language $\mathcal{L}_R = \{s \mid \exists\, w \text{ s.t. } (s, w) \in R\}$. We call w a witness for the statement s, and the tuple (s, w) a statement-witness pair. Importantly, a zero-knowledge proof does not leak the witness.

Definition 3. *We define the* equality-of-discrete-logarithms relation *as*

$$\mathsf{DLEQ} := \{((G_1, G_2, Y_1, Y_2), x) \in \mathbb{G}^4 \times \mathbb{F}_q \mid Y_1 = xG_1 \wedge Y_2 = xG_2\}.$$

As in the simplified presentation of DLOG*, we omit the group order and description from the statement. We denote the associated language by* $\mathcal{L}_{\mathsf{DLEQ}}$*.*

NIZKs can be constructed either using an honestly generated Common Reference String (CRS) or in the Random Oracle Model (ROM). In practice, one would likely use the NIZK presented in Figure 1, obtained from the Chaum-Pedersen sigma protocol for equality of discrete logarithms [CP93] via the Fiat–Shamir transform, which gives a NIZK in the programmable ROM.

Definition 4. *Let R be a polynomial-time decidable relation. We define a* non-interactive zero-knowledge proof system in the programmable random oracle (RO) model for the relation R *as a tuple of polynomial-time algorithms with access to a random oracle $\mathcal{O}$:*

- $\mathsf{P}^{\mathcal{O}}(s, w)$ *is an oracle-aided* PPT *prover algorithm that takes a statement $s \in \mathcal{L}_R$ and a witness w for s as input, and outputs a proof π.*
- $\mathsf{V}^{\mathcal{O}}(s, \pi)$ *is a deterministic oracle-aided polynomial-time verifier algorithm that takes as input a statement s and a proof π, and outputs either 1 if the proof is accepted, or 0 if it is rejected.*

We require (P, V) *to satisfy the following properties.*

Completeness: *For all $(s, w) \in R$, we have*

$$\Pr\left[\mathsf{V}^{\mathcal{O}}(s, \pi) = 1 \;\middle|\; \pi \leftarrow \mathsf{P}^{\mathcal{O}}(s, w)\right] = 1,$$

where the probability is over the choice of $\mathcal{O}$ and the randomness of P*.*

Computational soundness: *For every oracle-aided* PPT *adversary $\mathcal{A}$,*

$$\Pr\left[|s| \geq \lambda \wedge s \notin \mathcal{L}_R \wedge \mathsf{V}^{\mathcal{O}}(s, \pi) = 1 \;\middle|\; (s, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)\right] \in \mathsf{negl}(\lambda),$$

where the probability is over the choice of $\mathcal{O}$ and the randomness of $\mathcal{A}$.

$\mathsf{strongSigForge}_{\mathcal{A},\Sigma}(\lambda)$	$\mathcal{O}_{\mathsf{Sign}}(m)$
1 : $\mathcal{Q} := \emptyset$	1 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	2 : $\mathcal{Q} := \mathcal{Q} \cup \{(m, \sigma)\}$
3 : $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pk})$	3 : **return** σ
4 : **return** $((m^*, \sigma^*) \notin \mathcal{Q} \land \mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*))$	

Fig. 2: $\mathsf{strongSigForge}_{\mathcal{A},\Sigma}$ experiment.

Perfect zero knowledge: (P, V) *is perfect zero-knowledge if there exists a* PPT *simulator* $\mathcal{S}$ *such that, for all* $(s, w) \in R$*, it produces a proof and a programming function* $(\pi, f) \leftarrow \mathcal{S}(s)$ *such that* $(\pi, \mathcal{O}|_f)$ *is identically distributed as* $(\mathsf{P}^{\mathcal{O}}(s, w), \mathcal{O})$*.*

In the standard model, we must use a CRS and, therefore, the proof system is extended via a generator for the CRS. The soundness error is then computed w.r.t. an honestly generated CRS. On the other hand, the simulator establishing the zero-knowledge property also receives a trapdoor that allows simulation of the proof. If we allow a CRS, we could, for example, use the pairing-based NIZK by Groth and Sahai [GS08]. A CRS would complicate the setup of the adaptor scheme because neither the receiver nor the sender can generate it alone without violating security.

2.2 Adaptor Signatures

Adaptor signatures extend standard digital signature schemes, which we recall together with the formal definition of their *strong* existential unforgeability next.

Definition 5. *A* digital signature scheme *consists of a tuple of algorithms* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *such that*

- $\mathsf{KeyGen}(1^\lambda)$ *is a* PPT *algorithm that takes as input the security parameter* 1^λ *and outputs a pair* $(\mathsf{sk}, \mathsf{pk})$ *consisting of a secret signing key* sk *and a public verification key* pk*.*
- $\mathsf{Sign}(\mathsf{sk}, m)$ *is a* PPT *algorithm that takes as input a message* $m \in \{0,1\}^*$ *and a signing key* sk*, and outputs a signature* σ*.*
- $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$ *is a deterministic polynomial-time algorithm that takes as input a verification key* pk*, a message* $m \in \{0,1\}^*$*, and a signature* σ*, and outputs 1 if* σ *is a valid signature on* m *under* pk *and 0 otherwise.*

Any Σ *must satisfy* signature correctness*, i.e., for all* $m \in \{0,1\}^*$*,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{pk}, m, \sigma) = 1 \;\middle|\; \begin{array}{l}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)\end{array}\right] = 1.$$

Definition 6. *A signature scheme* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *satisfies* strong existential unforgeability under a chosen-message attack (SUF-CMA security) *if, for every* PPT *adversary* $\mathcal{A}$*,*

$$\Pr[\mathsf{strongSigForge}_{\mathcal{A},\Sigma}(\lambda) = 1] \in \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{strongSigForge}_{\mathcal{A},\Sigma}$ *is defined in Figure 2.*

In the following definition of adaptor signatures due to Aumayr et al. [AEE+21], the hard relation R is modelled as a set of statement–witness pairs (S, w). A *pre-signature* serves as a commitment—or encrypted version—of a valid signature on a given message with respect to some statement $S \in \mathcal{L}_R$. While not valid on

$\mathsf{KeyGen}(1^\lambda)$	$\mathsf{Sign}(\mathsf{sk}, m)$	$\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$
$1:\ x \leftarrow \mathbb{F}_q$	$1:\ (X, x) := \mathsf{sk}$	$1:\ X := \mathsf{pk}$
$2:\ X := xG$	$2:\ k \leftarrow \mathbb{F}_q$	$2:\ (r, s) := \sigma$
$3:\ \mathsf{sk} := (X, x)$	$3:\ r := \mathsf{H}(X \parallel kG \parallel m)$	$3:\ r' := \mathsf{H}(X \parallel sG - rX \parallel m)$
$4:\ \mathsf{pk} := X$	$4:\ s := k + rx$	$4:\ \textbf{return}\ (r = r')$
$5:\ \textbf{return}\ \mathsf{sk}, \mathsf{pk}$	$5:\ \textbf{return}\ (r, s)$	

Fig. 3: Schnorr signature scheme.

its own, a pre-signature can be efficiently "adapted" (i.e., transformed) into a valid signature by anyone who knows a witness for S. Moreover, given both the pre-signature and the corresponding valid signature, this witness can be efficiently extracted. Adaptor signatures thus enable the controlled exchange of a signature for the hidden value embedded in the pre-signature.

Definition 7. *An* adaptor signature scheme *for a signature scheme* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *w.r.t. a hard relation* R *is a tuple of algorithms* $\mathsf{a}\Sigma_R = (\Sigma, \mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Ext})$ *such that*

- $\mathsf{pSign}(\mathsf{sk}, m, S)$ *is a* PPT *algorithm that takes as input a signing key* sk *relative to* Σ*, a message* $m \in \{0,1\}^*$*, and a statement* $S \in \mathcal{L}_R$*, and outputs a pre-signature* $\tilde{\sigma}$ *on* m *w.r.t.* S *under* sk*.*
- $\mathsf{pVrfy}(\mathsf{pk}, m, S, \tilde{\sigma})$ *is a deterministic polynomial-time algorithm that takes as input a verification key* pk *relative to* Σ*, a message* $m \in \{0,1\}^*$*, a statement* $S \in \mathcal{L}_R$*, and a pre-signature* $\tilde{\sigma}$*, and outputs 1 if the pre-signature is valid and 0 otherwise.*
- $\mathsf{Adapt}(\mathsf{pk}, \tilde{\sigma}, w)$ *is a deterministic polynomial-time algorithm that takes a public key* pk*, a pre-signature* $\tilde{\sigma}$*, and a witness* w *for a statement in the relation* R *as input, and outputs a signature* σ*.*
- $\mathsf{Ext}(\mathsf{pk}, \sigma, \tilde{\sigma}, S)$ *is a deterministic polynomial-time algorithm that takes a public key* pk*, a signature* σ*, a pre-signature* $\tilde{\sigma}$*, and a statement* $S \in \mathcal{L}_R$ *as input, and outputs either a witness* w *such that* $(S, w) \in R$ *or* $\perp$*.*

Any $\mathsf{a}\Sigma_R$ *must satisfy* pre-signature correctness*, i.e., for all* $m \in \{0,1\}^*$ *and every pair* $(S, w) \in R$*,*

$$\Pr\left[\begin{array}{l|l} \mathsf{pVrfy}(\mathsf{pk}, m, S, \tilde{\sigma}) = 1 & (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \wedge\ \mathsf{Vrfy}(\mathsf{pk}, m, \sigma) = 1 & \tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, S), \\ \wedge\ (S, w') \in R & \sigma := \mathsf{Adapt}(\mathsf{pk}, \tilde{\sigma}, w), \\ & w' := \mathsf{Ext}(\mathsf{pk}, \sigma, \tilde{\sigma}, S) \end{array}\right] = 1.$$

The adaptor signatures considered in this work do not use a public key for adaptation or extraction, so we omit it from the notation for Adapt and Ext in the rest of the paper.

We present two adaptor signatures based on signature schemes that are widely used in practice, namely the Schnorr signature scheme [Sch90] and ECDSA [Nat23].

Schnorr adaptor signature scheme. Moving from the Schnorr signature scheme [Sch90] in Figure 3 to the adaptor signature scheme in Figure 4 is straightforward. Consider a statement-witness pair $(Y, y) \in \mathsf{DLOG}$. The main difference is that when pSign computes a pre-signature $(r, \tilde{s}) \in \mathbb{F}_q \times \mathbb{F}_q$ on m w.r.t. Y, it samples $k \leftarrow \mathbb{F}_q$ uniformly at random, as Sign does; however, it computes r as $\mathsf{H}(X \parallel kG + Y \parallel m)$, so the group element in the input to H is shifted by Y. The element $\tilde{s}$ is computed as $k + rx$. This means that the randomness under r is in fact $k + y$, while $\tilde{s}$ is computed as s in Sign, depending on the randomness k. Thus, to adapt and obtain a valid signature, $\tilde{s}$ must be shifted by y; that is, on input a pre-signature $(r, \tilde{s})$ and a witness y, Adapt outputs $(r, \tilde{s} + y)$. Conversely, Ext obtains the witness y by subtracting $s - \tilde{s}$.

$\mathsf{pSign}(\mathsf{sk}, m, Y)$

1 : $(X, x) := \mathsf{sk}$
2 : $k \leftarrow \mathbb{F}_q$
3 : $r := \mathsf{H}(X \parallel kG + Y \parallel m)$
4 : $\tilde{s} := k + rx$
5 : **return** $(r, \tilde{s})$

$\mathsf{pVrfy}(\mathsf{pk}, m, Y, \tilde{\sigma})$

1 : $X := \mathsf{pk}$
2 : $(r, \tilde{s}) := \tilde{\sigma}$
3 : $r' := \mathsf{H}(X \parallel \tilde{s}G - rX + Y \parallel m)$
4 : **return** $(r = r')$

$\mathsf{Adapt}(\tilde{\sigma}, y)$

1 : $(r, \tilde{s}) := \tilde{\sigma}$
2 : $s := \tilde{s} + y$
3 : **return** (r, s)

$\mathsf{Ext}(\sigma, \tilde{\sigma}, Y)$

1 : $(r, s) := \sigma$
2 : $(\tilde{r}, \tilde{s}) := \tilde{\sigma}$
3 : $y' := s - \tilde{s}$
4 : **if** $(Y, y') \in \mathsf{DLOG}$ **then**
5 : **return** y'
6 : **else return** $\bot$

Fig. 4: Schnorr adaptor signature [AEE+21].

$\mathsf{KeyGen}(1^\lambda)$

1 : $x \leftarrow \mathbb{F}_q$
2 : $X := xG$
3 : $\mathsf{sk} := (X, x)$
4 : $\mathsf{pk} := X$
5 : **return** sk, pk

$\mathsf{Sign}(\mathsf{sk}, m)$

1 : $(X, x) := \mathsf{sk}$
2 : $k \leftarrow \mathbb{F}_q^*$
3 : $K := kG$
4 : $r := f(K)$
5 : $s := k^{-1}(\mathsf{H}(m) + rx)$
6 : **if** $0 \le s \le (q-1)/2$
7 : **return** $\sigma := (r, s)$
8 : **else**
9 : **return** $\sigma := (r, -s)$

$\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$

1 : $X := \mathsf{pk}$
2 : $(r, s) := \sigma$
3 : $u := \mathsf{H}(m)s^{-1}$
4 : $v := rs^{-1}$
5 : $K' := uG + vX$
6 : **return** $(f(K') = r \wedge$
$0 \le s \le (q-1)/2)$

Fig. 5: ECDSA$^+$ signature scheme.

ECDSA adaptor signature scheme. We now recall the ECDSA signature scheme [Nat23]. Since we rely on *strong* unforgeability in this work, we consider the so-called *positive* ECDSA presented in Figure 5, which we denote ECDSA$^+$. The scheme is specified by an elliptic curve group $\mathbb{G}$ over a finite field $\mathbb{F}_p$. Each group element can be represented as a pair of elements of $\mathbb{F}_p$, and thus we refer to its first and second coordinates. Besides a cryptographic hash H, the scheme uses a function $f\colon \mathbb{G} \to \mathbb{F}_q$, which is typically a projection onto the first coordinate of the group element modulo q. The positive variant differs from standard ECDSA in its handling of the sign of the s component; it ensures that s always lies in the interval $[0, (q-1)/2]$. If an s computed during signing exceeds this bound, it is replaced with $-s$. Note that standard ECDSA is not strongly unforgeable: if (r, s) is a valid ECDSA signature, then $(r, -s)$ is also valid. However, the restriction on s in ECDSA$^+$ eliminates this issue, and it is widely conjectured to be strongly unforgeable. The ECDSA$^+$ variant is also commonly used in blockchain protocols such as Bitcoin.

The ideas underlying the ECDSA-based adaptors were presented by Moreno-Sanchez and Kate [MSK18]. We formally define the ECDSA adaptor signature scheme w.r.t. the DLOG relation [Fou19] in Figure 6, and we denote it by $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$. To sign a message m under a public key $X = xG$ in ECDSA, a signer samples

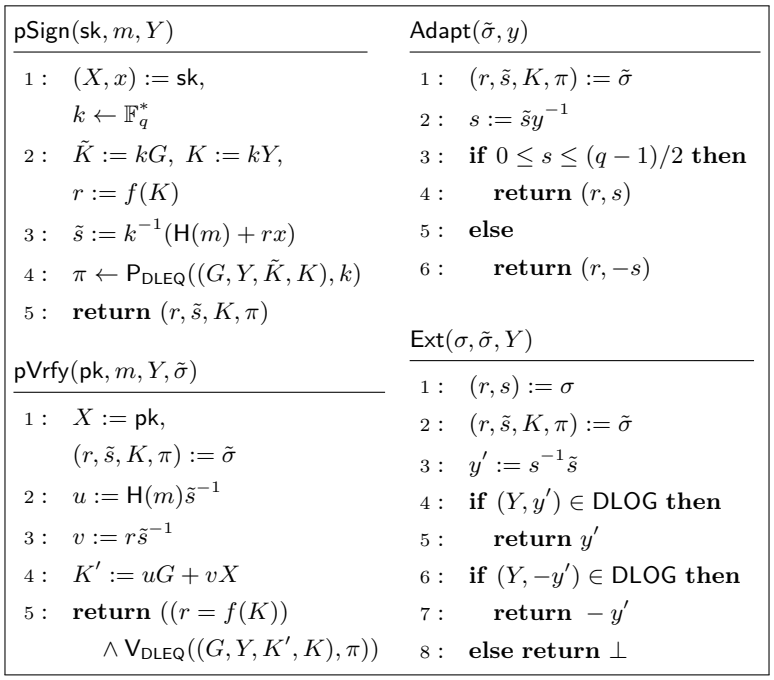

Fig. 6: Adaptor signature scheme $\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}$ for ECDSA^{+} w.r.t. the DLOG relation.

$k \leftarrow \mathbb{F}_q^*$ uniformly at random. It computes the nonce $R := kG$ and sets r as the projection $f(R)$. The signature s is derived as $k^{-1}(\mathsf{H}(m) + rx)$. Verification entails checking whether the point R' reconstructed from s^{-1}, $\mathsf{H}(m)$, and r satisfies $f(R') = r$. Moving to the adaptor setting with an instance $Y = yG$ of DLOG, the pre-signing algorithm pSign modifies how the randomness is utilized. It samples $k \leftarrow \mathbb{F}_q^*$ but computes the coordinate r using $K := kY$, which effectively shifts the nonce by the witness y. The pre-signature component $\tilde{s}$ is then computed as $k^{-1}(\mathsf{H}(m) + rx)$, similarly to an ECDSA signature but relative to the shifted r. Notice that, for (r, s) to be a valid ECDSA signature, the underlying nonce must be $K = kY = (ky)G$. This implies the effective randomness for the valid signature is ky, leading to $s = (ky)^{-1}(\mathsf{H}(m) + rx)$. Therefore, to adapt the pre-signature $(r, \tilde{s})$ into a valid signature (r, s), one must multiply $\tilde{s}$ by y^{-1} and output $(r, \tilde{s}y^{-1})$. Conversely, given a pre-signature and a signature pair, one can recover the witness y by computing $\tilde{s}s^{-1}$.

Crucially, the consistency of the pre-signature with the message cannot be guaranteed solely by the structure of $\tilde{s}$. To convince a verifier that $\tilde{s}$ can indeed be adapted using y, pSign must explicitly prove that the randomness k is consistent. Specifically, the pre-signature includes the shifted nonce K and a NIZK proof π that $\log_G \tilde{K} = \log_Y K$, where $\tilde{K} := kG$ and $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$ denotes the NIZK proof system for equality of discrete logarithms. The pre-signature verification algorithm pVrfy reconstructs $\tilde{K}$ from $\tilde{s}$ and verifies the proof π. This ensures that the pre-signature constitutes a valid commitment to the signature encrypted by the witness y.

The scheme is well-defined only for $y \neq 0$, since it uses the inverse of y in $\mathbb{F}_q$. Throughout this work, we therefore require GenR for DLOG to resample when $y = 0$, and the ECDSA adaptor scheme rejects statement–witness pairs with $y = 0$, including $(Y, y) = (0, 0)$. To prevent similar problems with signatures and pre-signatures, we assume that the signing and pre-signing algorithms Sign and pSign resample if any of the values r, s, or $\tilde{s}$ is zero. We also consider signatures and pre-signatures containing such zero values invalid and assume that the algorithms check and abort before performing an inversion when necessary.

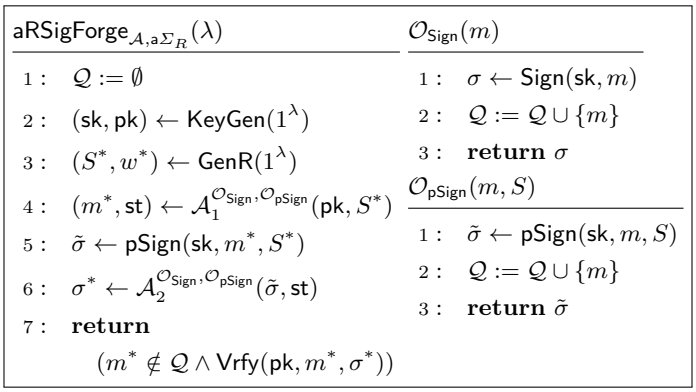

Fig. 7: The aRSigForge$_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment [AEE+21].

3 Robust Security of Adaptor Signatures

Standard existential unforgeability for signature schemes guarantees that only the secret-key holder can generate valid signatures, making it computationally infeasible to forge a signature on any fresh message m—even for a PPT adversary with oracle access to signatures on chosen messages.

Adaptor signatures aim to achieve a similar security guarantee: pre-signatures should not enable an adversary to forge valid signatures. To formalize this, the adversary is granted access to a pre-signing oracle—similar to the standard signing oracle—that returns pre-signatures for messages of its choice. A key question in defining security is how tightly access to this oracle should be restricted. A conservative approach allows it to accept arbitrary statements S, even if the adversary lacks the corresponding witness w. This is the approach taken by Aumayr et al. [AEE+21] in their definition of *existential unforgeability under chosen-message attack for adaptor signatures*. We refer to such definitions as *robust* and restate their definition in Definition 8.

Definition 8 (Robust existential unforgeability [AEE+21]). *An adaptor signature scheme* $\mathsf{a}\Sigma_R$ *for a signature scheme* Σ *w.r.t. a hard relation* R *satisfies* robust existential unforgeability under a chosen-message attack for adaptor signatures (aREUF-CMA security) *if, for every* PPT *adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\Pr[\mathsf{aRSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda) = 1] \in \mathsf{negl}(\lambda),$$

where the experiment aRSigForge$_{\mathcal{A},\mathsf{a}\Sigma_R}$ *is defined in Figure 7.*

The goal of the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the aRSigForge$_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment in Figure 7 is to forge a signature on a message $m^* \in \{0,1\}^*$ of its choice under a public key pk generated by KeyGen, without access to the corresponding secret key sk. During the experiment, $\mathcal{A}$ has oracle access to both the signing oracle $\mathcal{O}_{\mathsf{Sign}}$ and the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$, which return signatures and pre-signatures on arbitrary messages m. In particular, $\mathcal{O}_{\mathsf{pSign}}$ allows requests for any statement $S \in \mathcal{L}_R$. The adversary wins if it outputs a valid signature σ^* on m^*, without ever querying $\mathcal{O}_{\mathsf{Sign}}$ or $\mathcal{O}_{\mathsf{pSign}}$ on m^*. That is, the only pre-signature on m^* seen by $\mathcal{A}$ is the one given during the challenge phase, and this pre-signature corresponds to a statement S^* chosen by the experiment, for which $\mathcal{A}$ does not know the witness w^*.

3.1 Limits on Robust Security of Adaptor Signatures

In this section, we argue that *robust* security definitions for adaptor signatures, such as the one given in Definition 8, are inadequate when applied to the aECDSA$^+_{\text{DLOG}}$ construction. In particular, we show that, with unrestricted access to pre-signatures for aECDSA$^+_{\text{DLOG}}$, we can construct an efficient fixed Diffie–Hellman oracle, which in turn can speed up algorithms for solving the discrete logarithm problem beyond the conjectured bit-security level.

E	$y^2 = x^3 + 7$ over $\mathbb{F}_p$
p	$= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$ a prime
$\log_2(p)$	≈ 256
Order q of E	$= 115792089237316195423570985008687907852837564279074904382605163141518161494337$ a prime
$\log_2(q)$	≈ 256

Table 1: Elliptic curve secp256k1.

We rely on a variant of the computational Diffie-Hellman (CDH) problem, where the first secret scalar a is fixed, leading to the notion of a fixed Diffie–Hellman oracle [BG04].

Definition 9. *Consider a cyclic group $\mathbb{G}$ of prime order q with a generator G and a group element $aG = P \in \mathbb{G}$. The* fixed Diffie-Hellman oracle for P *is a function FDH_P that on input $bG \in \mathbb{G}$ returns abG.*

Fournier [Fou19] observed that when the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$ accepts unrestricted queries, i.e., on arbitrary statements $S = Y$, the $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ scheme effectively grants access to such a fixed Diffie–Hellman oracle for the public key $X = xG$. To see this, consider the output of $\mathsf{pSign}(\mathsf{sk}, m, Y)$, i.e., a pre-signature $\tilde{\sigma} = (r, \tilde{s}, K, \pi)$ such that $\tilde{s} = k^{-1}(\mathsf{H}(m) + rx)$ and $K = kY$ for uniform $k \leftarrow \mathbb{F}_q^*$. It follows that

$$\begin{aligned} \tilde{s}K &= \left(kk^{-1}\left(\mathsf{H}(m) + rx\right)\right) Y = (\mathsf{H}(m) + rx)Y \\ \tilde{s}K - \mathsf{H}(m)Y &= rxY \\ r^{-1}(\tilde{s}K - \mathsf{H}(m)Y) &= xY. \end{aligned}$$

Thus, for an arbitrary Y, we can use the above derivation to compute xY, implementing the $\mathsf{FDH}_{\mathsf{pk}}$ oracle for the public key $\mathsf{pk} = X = xG$.

Observation 1 *Given $\mathcal{O}_{\mathsf{pSign}}(\cdot)$ for aECDSA^+_{DLOG}, we can implement a call to the $\mathsf{FDH}_{\mathsf{pk}}$ oracle fixed for a public key pk via*

1 call to $\mathcal{O}_{\mathsf{pSign}}(\cdot)$
1 hash $\mathsf{H}(\cdot)$
1 subtraction in $\mathbb{F}_q$
1 inverse in $\mathbb{F}_q$
2 multiplications in $\mathbb{F}_q$
1 group operation in $\mathbb{G}$
2 scalar multiplications in $\mathbb{G}$.

Next, we show that providing an adversary with a fixed Diffie-Hellman oracle for the public key would lead to a significant degradation in the conjectured bit-security level of the discrete logarithm problem. Specifically, we consider the common instantiation of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ in Bitcoin.

Security degradation for secp256k1 under unrestricted pre-signature oracle access. We instantiate the adaptor signature scheme $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ over the secp256k1 elliptic curve group [Cer], which is used, e.g., for public-key cryptography in Bitcoin. According to the secp256k1 specification (see Table 1 or [Cer]), $\mathbb{G}$ is a cyclic group of prime order q with generator G, the underlying field has odd characteristic, and $\log_2(q) \approx 256$. We assume that $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ uses secp256k1 for both the DLOG relation and the ECDSA$^+$ signature scheme. By the above, when considering robust security experiments such as $\mathsf{aRSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ with unrestricted access to the pre-signing oracle, the adversary $\mathcal{A}$ in fact obtains a fixed Diffie-Hellman oracle $\mathsf{FDH}_{\mathsf{pk}}$ for the fixed underlying public key pk.

We can exploit this setup to weaken the assumed security of the Bitcoin curve secp256k1. We use the meet-in-the-middle attack of Brown and Gallant [BG04], which uses an available fixed Diffie-Hellman oracle FDH_P to compute the discrete logarithm of a group element $P \in \mathbb{G}$ with complexity depending on a factorization of $q-1$, where $\mathbb{G}$ is a cyclic group of prime order q. In our case, we could compute the discrete logarithm of $X := \mathsf{pk}$, namely the secret component x of sk such that $X = xG$. If done efficiently enough, this would break the security of the underlying ECDSA$^+$ signature scheme by revealing the corresponding secret key.

Theorem 2. *Let* $(\mathsf{sk}, \mathsf{pk})$ *be a key pair for ECDSA*$^+$ *over the secp256k1 elliptic curve group. Given* pk *and access to the pre-signing oracle* $\mathcal{O}_{\mathsf{pSign}}$ *for* aECDSA^+_{DLOG} *defined in Figure 7, the secret key* sk *can be computed in time dominated by* 2^{93} *group operations.*

Proof. Let $q = uv + 1$ for positive integers u and v. The attack computes the discrete logarithm in two parts, first by finding a certain value modulo u and then modulo v, both using the baby-step giant-step algorithm [Sha71]. Specifically, we use the fixed Diffie-Hellman oracle to compute the group element $x^u G$, which would not be possible without the oracle because x is secret. Brown and Gallant [BG04] show that the time complexity of the attack is dominated by approximately

$$u \text{ calls to the } \mathsf{FDH}_{\mathsf{pk}} \text{ oracle and } 2(\sqrt{u} + \sqrt{v}) \text{ scalar multiplications in } \mathbb{G},$$

and is generally minimized for $u \approx \sqrt[3]{q}$. For brevity, we write GO and SM instead of group operation and scalar multiplication, respectively. In our case, with the cost of a call to the $\mathsf{FDH}_{\mathsf{pk}}$ oracle as in Theorem 1, the time complexity would be

$$\begin{aligned} u \text{ (1 } \mathsf{H}(\cdot)\text{, 1 subtr. in } \mathbb{F}_q\text{, 1 inv. in } \mathbb{F}_q\text{, 2 multipl. in } \mathbb{F}_q\text{, 1 GO in } \mathbb{G}\text{, 2 SM in } \mathbb{G}) \\ + 2(\sqrt{u} + \sqrt{v}) \text{ SM in } \mathbb{G}, \end{aligned}$$

where we omit the call to the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$, as its unit cost is negligible in comparison to the cost of the other operations. Moreover, if we fix a message m, transforming the pre-signatures sampled by $\mathsf{pSign}(\mathsf{sk}, m, Y)$ into the answers of $\mathsf{FDH}_{\mathsf{pk}}(Y)$ always uses the same value $-\mathsf{H}(m)$, so a single hash evaluation and field negation suffice for all $\mathsf{FDH}_{\mathsf{pk}}$ calls. In comparison with the number of other operations, these costs can also be omitted from the following analysis, resulting in the complexity

$$\begin{aligned} &u \text{ inverses in } \mathbb{F}_q \\ &2u \text{ multiplications in } \mathbb{F}_q \\ &u \text{ group operations in } \mathbb{G} \\ &(2u + 2\sqrt{u} + 2\sqrt{v}) \text{ scalar multiplications in } \mathbb{G}. \end{aligned}$$

Moreover, when using the precomputation described by Brown and Gallant [BG04], a scalar multiplication in $\mathbb{G}$ costs at most $\log_2(q)/4 \approx 256/4 = 2^6$ group operations in $\mathbb{G}$. The precomputation consists of $2 \cdot 16 \cdot \log_2(q) < 2^{13}$ scalar multiplications of specific group elements in $\mathbb{G}$, each of which can be computed using the *double-and-add* method in $\log_2(q) < 2^8$ point additions and point doublings in $\mathbb{G}$. The precomputation can thus be upper-bounded by 2^{23} group operations in $\mathbb{G}$.

A factorization of $q-1$ for secp256k1 is given in Table 2. Let u be the product of the bold factors. Then $83 < \log_2(u) < 84$ and thus $u \approx \sqrt[3]{q}$, since $\log_2(\sqrt[3]{q}) \approx 85$. We now estimate the time complexity of the attack for this choice of u as

$$\begin{aligned} &2^{84} \text{ inv. in } \mathbb{F}_q + 2^{85} \text{ mult. in } \mathbb{F}_q + \left(2^{84} + 2^6(2 \cdot 2^{84} + 2 \cdot 2^{42} + 2 \cdot 2^{84})\right) \text{ GO in } \mathbb{G} \\ &< 2^{84} \text{ inverses in } \mathbb{F}_q + 2^{85} \text{ multipl. in } \mathbb{F}_q + \left(2^{84} + 2^{91} + 2^{49} + 2^{91}\right) \text{ GO in } \mathbb{G} \\ &< 2^{84} \text{ inverses in } \mathbb{F}_q + 2^{85} \text{ multipl. in } \mathbb{F}_q + 2^{93} \text{ GO in } \mathbb{G}. \end{aligned}$$

This result includes the described precomputation. Since the attack is a type of meet-in-the-middle attack, the space complexity should also be discussed. In our case, the space complexity is at least 2^{84} bits. □

Factor f of $q-1$	$\approx \log_2(f)$
2^6	6
3	1.6
149	7.2
631	9.3
107361793816595537	56.6
174723607534414371449	67.2
3419484869741660005223436092831 89	108.1

Table 2: Factorization of $q-1$ for secp256k1.

Although the attack is not feasible in practice, the rough time complexity of 2^{93} group operations under unrestricted pre-signing oracle access shows a significant weakening of the expected $\log_2(q)/2 \approx$ 128-bit security of the secp256k1 elliptic curve, since it computes the discrete logarithm of a random group element using much less than the expected 2^{128} group operations.

Moreover, as proposed by Brown and Gallant [BG04], the space complexity of the attack can be reduced by using the Pollard-ρ algorithm [Pol78] instead of the baby-step giant-step algorithm. However, when attempting to adapt the ideas from [BG04] to the Pollard-ρ setting, it was unclear how to efficiently compute certain group elements without increasing the overall time complexity of the attack.

4 Security of ECDSA Adaptor Signatures

In this section, we propose new security definitions for adaptor signature schemes, tailored to the $\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}$ construction in Figure 6. These definitions are inspired by the robust formulations introduced by Aumayr et al. [AEE+21], but are modified to avoid the issues discussed in Section 3.1. Instead of reiterating the robust definitions, we present the modified notions directly. We then formally prove that, under the necessary assumptions, $\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}$ satisfies the new security definitions.

4.1 Unforgeability of ECDSA Adaptor Signatures

We introduce a refined notion of *existential unforgeability under chosen-message attack for adaptor signatures*. Our definition restricts the adversary's access to the pre-signing oracle. Previously, the adversary could query the oracle w.r.t. any chosen statement $S \in \mathcal{L}_R$. We modify the pre-signing oracle such that it answers two types of queries: either 1) queries w.r.t. a fixed statement S^* chosen by the experiment, or 2) queries w.r.t. any statement S for which the adversary provides a witness.

Definition 10. *An adaptor signature scheme* $\mathsf{a}\Sigma_R$ *for a signature scheme* Σ *w.r.t. a hard relation* R *is* aEUF-CMA secure *if, for every* PPT *adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda) = 1] \in \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{aSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ *is defined in Figure 8.*

Note that the $\mathsf{aSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment in Figure 8 differs from the $\mathsf{aRSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment in Figure 7 only in the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$; otherwise, they proceed identically.

We prove that $\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}$ satisfies the existential unforgeability defined above. The proof reduces the existential unforgeability to the strong unforgeability of the underlying ECDSA^{+} signature scheme and proceeds analogously to the state-of-the-art proof of *robust* existential unforgeability of the *extended* ECDSA adaptor presented by Aumayr et al. [AEE+21]. However, we stress that we prove our notion of security for

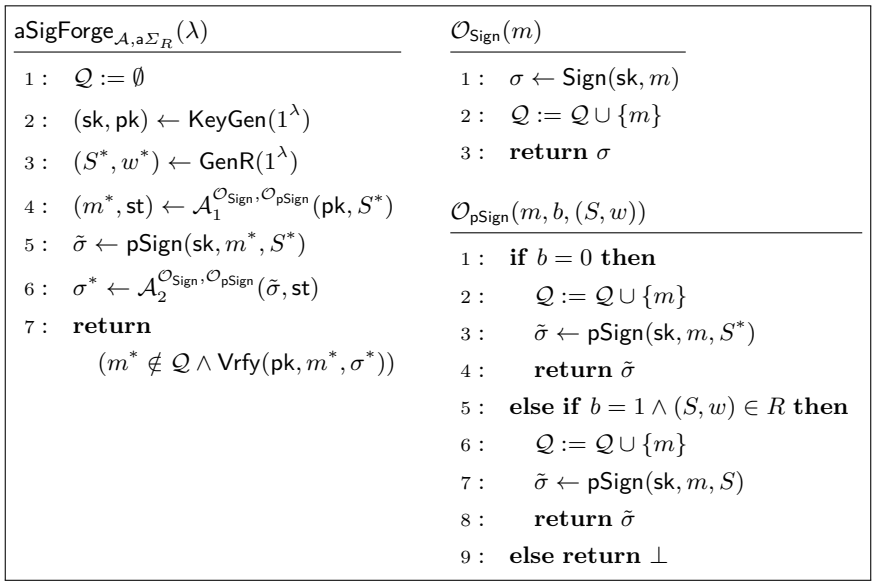

Fig. 8: $\mathsf{aSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment.

the original adaptor signature based on ECDSA$^+$ without modifying the scheme. In particular, we do not need to utilize a NIZK *proof of knowledge* and its online extraction algorithm in the reduction.

Recall that the construction of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ depends on a NIZK proof for equality of discrete logarithms (see Section 2.1). There are two approaches for obtaining such NIZKs. In order to obtain a proof in the standard model, we can work in the Common Reference String (CRS) setting and rely on a NIZK constructed under some hardness assumption [GS08]. In practice, one would likely use a NIZK obtained from the Chaum-Pedersen sigma protocol for equality of discrete logarithms [CP93] via the Fiat–Shamir transform, which gives a NIZK in the (programmable) Random Oracle Model (ROM). Our result can thus be stated as follows.

Theorem 3. *Let* aECDSA^+_{DLOG}*, described in Figure 6, be an adaptor signature scheme for ECDSA*$^+$ *w.r.t. the* DLOG *relation. If the ECDSA*$^+$ *signature scheme is* SUF-CMA *secure and* DLOG *is a hard relation, then* aECDSA^+_{DLOG} *is* aEUF-CMA *secure in the random oracle model.*

Additionally, if $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$ *is a perfectly zero-knowledge NIZK proof system for equality of discrete logarithms, then the scheme* aECDSA^+_{DLOG} *is* aEUF-CMA *secure in the standard model.*

We show that the existential unforgeability of the adaptor $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ reduces to the strong unforgeability of ECDSA$^+$. Assuming that an adversary $\mathcal{A}$ can break the $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}$ experiment with non-negligible probability, we construct a forger $\mathcal{F}$ that uses $\mathcal{A}$ to win the strong-unforgeability experiment $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ with at least the same probability.

We simulate the adversary's environment given access only to a signing oracle, as required in the strong-unforgeability experiment. While simulating the signing oracle $\mathcal{O}_{\mathsf{Sign}}$ is straightforward, simulating the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$ is more challenging, since producing a valid pre-signature typically requires knowledge of the secret signing key sk, which the forger lacks. However, if the adversary provides a witness for the statement to be pre-signed, we can simulate pre-signatures using just the signing oracle. Our modified definition of existential unforgeability ensures that, in the reduction, we always have a witness that allows us to simulate a pre-signature. The simulation procedure is described in Figure 9.

The reduction then proceeds via standard game-hopping. Starting from the $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ forgery game $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}$, we define a sequence of games $\mathcal{G}_1, \mathcal{G}_2$ and $\mathcal{G}_3$ that gradually shift toward the ECDSA$^+$ strong unforgeability experiment. The key steps involve replacing real pre-signatures with simulated ones and introducing an abort condition to ensure that the final forgery is not trivially derived from a known

$\mathcal{S}_{pS}^{\mathsf{Sign}}(m,(Y,y))$

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk},m)$
parse $(r,s) := \sigma$
$b \leftarrow \{0,1\}$
$s := (-1)^b s$
$\tilde{s} := sy$
$u := \mathsf{H}(m)s^{-1}$
$v := rs^{-1}$
$K := uG + vX$
$\tilde{K} := y^{-1}K$
$\pi_S \leftarrow \mathcal{S}_{\mathsf{DLEQ}}((G,Y,\tilde{K},K))$
return $(r,\tilde{s},K,\pi_S)$

Fig. 9: Simulator of pre-signatures $\mathcal{S}_{pS}^{\mathsf{Sign}}$ for $\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}$.

pre-signature. We prove that these transformations are indistinguishable to $\mathcal{A}$, up to a negligible difference in success probability. A crucial step in the argument requires the DLOG assumption to ensure that $\mathcal{A}$ cannot compute the forgery simply by adapting a pre-signature using a known witness. It is also essential that the simulated pre-signatures from Figure 9 are both valid and identically distributed to those generated by the real pSign algorithm, which is argued in a standalone lemma.

Lemma 1. *Let* $\mathsf{aECDSA}^{+}_{DLOG}$*, described in Figure 6, be an adaptor signature scheme for ECDSA*$^{+}$ *w.r.t. the* DLOG *relation. Let* $\mathcal{S}_{pS}^{\mathsf{Sign}}$*, described in Figure 9, be a* PPT *simulator of pre-signatures, and suppose that the NIZK proof system for equality of discrete logarithms has a perfect zero-knowledge simulator* $\mathcal{S}_{\mathsf{DLEQ}}$*.*

Then, for every key pair $(\mathsf{sk},\mathsf{pk})$ *related to ECDSA*$^{+}$*, message* $m \in \{0,1\}^*$*, and statement–witness pair* $(Y,y) \in \mathsf{DLOG}$*, the value* $\tilde{\sigma}$ *produced as*

$$\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(m,(Y,y)),$$

is a pre-signature on m *w.r.t.* Y *that is valid under* pk*, and is distributed identically to* $\mathsf{pSign}(\mathsf{sk},m,Y)$*.*

Proof. Fix a key pair $(\mathsf{sk},\mathsf{pk})$, message m and an instance $(Y,y) \in \mathsf{DLOG}$. Consider the algorithm of $\mathcal{S}_{pS}^{\mathsf{Sign}(\mathsf{sk},\cdot)}$, which has black-box access to the signing oracle $\mathsf{Sign}(\mathsf{sk},\cdot)$ for ECDSA^{+}. We write $\mathsf{Sign}(\mathsf{sk},\cdot)$ to signify that the oracle works like the algorithm Sign, with the difference that the input sk is fixed, while the message part of the input works as expected. We now show that $\mathcal{S}_{pS}$ perfectly simulates the output of the pre-signing algorithm $\mathsf{pSign}(\mathsf{sk},\cdot,\cdot)$ of $\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}$.

On input $(m,(Y,y))$, the simulator $\mathcal{S}_{pS}^{\mathsf{Sign}(\mathsf{sk},\cdot)}$ begins by querying the signing oracle to obtain a signature (r^*,s^*). We demonstrate that $\mathcal{S}_{pS}$ can construct a pre-signature that adapts into this exact signature.

Recall that in ECDSA^{+}, the signing algorithm outputs signatures with only positive s values, i.e.,

$$0 \le s^* \le (q-1)/2.$$

This means that only half of the randomness values $k \in \mathbb{F}_q$ are actually used in signing: if $k \in \mathbb{F}_q$ produces (r,s), then $-k$ produces $(r,-s)$, and only one of these will satisfy the positivity condition for s.

In contrast, the pre-signing algorithm pSign accepts any randomness $k \in \mathbb{F}_q$ and does not enforce positivity of s until the Adapt step. For a fixed y, half of the values k lead to a final signature with s and half with $-s$ after adaptation. To simulate this uniformly over all $k \in \mathbb{F}_q$, the simulator $\mathcal{S}_{pS}$ flips the sign of s^* with probability $1/2$ and sets $s := \pm s^*$ accordingly. The resulting pair (r^*,s) satisfies the ECDSA verification

equation for some randomness $l \leftarrow \mathbb{F}_q$, although it does not necessarily satisfy the positivity condition of ECDSA$^+$.

Next, the simulator $\mathcal{S}_{pS}$ reverses the adaptation step to recover the pre-signature component $\tilde{s}$ by computing $\tilde{s} := sy$. It then constructs the group elements

$$K := \mathsf{H}(m)s^{-1}G + r^*s^{-1}X \quad \text{and} \quad \tilde{K} := y^{-1}K.$$

To simulate the NIZK proof of discrete logarithm equality, $\mathcal{S}_{pS}$ invokes the simulator $\mathcal{S}_{\mathsf{DLEQ}}$ provided by the zero-knowledge property of the proof system. Since we assume perfect zero knowledge, the simulated proof π_S is identically distributed to a real proof generated on the same statement using the corresponding witness. The simulator $\mathcal{S}_{\mathsf{DLEQ}}$ is run on the tuple $(G, Y, \tilde{K}, K)$, proving that $\tilde{K}$ and K share the same discrete logarithm with respect to G and Y, respectively. Finally, $\mathcal{S}_{pS}$ outputs the simulated pre-signature $(r^*, \tilde{s}, K, \pi_S)$.

By construction, this pre-signature correctly adapts to the signature (r^*, s^*), as

$$\mathsf{Adapt}((r^*, \tilde{s}, K, \pi_S), y) = \mathsf{Adapt}((r^*, sy, K, \pi_S), y) = (r^*, s^*).$$

We now verify that this simulated pre-signature is also valid under pk. During verification, the recomputed value

$$K' := \mathsf{H}(m)\tilde{s}^{-1}G + r^*\tilde{s}^{-1}X = y^{-1}(\mathsf{H}(m)s^{-1}G + r^*s^{-1}X) = \tilde{K},$$

matches the expected input to the NIZK verifier. Thus, π_S is a valid proof for the statement (G, Y, K', K).

Moreover, since (r^*, s^*) is a valid ECDSA signature (and flipping the sign preserves validity), we have

$$f(K) = r^*.$$

Hence, all conditions for pre-signature validity are satisfied.

Finally, because the pre-signature was constructed with randomness l corresponding to the signature (r^*, s), the effective randomness used in the pre-signature is ly^{-1}. Therefore, the simulated output is distributed identically to a real pre-signature generated by $\mathsf{pSign}(\mathsf{sk}, m, Y)$, establishing the lemma. □

Proof (Theorem 3). We show the reduction of the existential unforgeability of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ to the strong unforgeability of the ECDSA$^+$ signature scheme. For brevity, let us denote by $\mathcal{G}_0$ the $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}$ experiment, which will be the starting game in the reduction. First, we modify game $\mathcal{G}_0$ step by step into games $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$ and show that, if a PPT adversary $\mathcal{A}$ wins $\mathcal{G}_0$ with probability p, it wins $\mathcal{G}_3$ with probability at least $p - \mathsf{negl}(\lambda)$. Then, we construct a PPT forger $\mathcal{F}$ that uses $\mathcal{A}$ to output a fresh message–signature pair (m^*, σ^*) and wins the strong-unforgeability experiment $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ with at least the probability that $\mathcal{A}$ wins $\mathcal{G}_3$. Finally, we apply the reduction and prove the statement.

For each game $\mathcal{G}_i$, we describe both the differences in the experiments and the changes in the probabilities of $\mathcal{A}$ winning $\mathcal{G}_i$ compared to the previous game $\mathcal{G}_{i-1}$. We also explain why the games are indistinguishable to $\mathcal{A}$. Let

$$\Pr[\mathcal{G}_i^{\mathcal{A}}(\lambda) = 1]$$

be the probability of $\mathcal{A}$ winning game $\mathcal{G}_i$.

Game $\mathcal{G}_0$: To aid the reader in following the proof, we describe the starting game $\mathcal{G}_0$ in Figure 10. Consider a PPT adversary $\mathcal{A}$ winning $\mathcal{G}_0$ with the corresponding probability.

Game $\mathcal{G}_1$: Game $\mathcal{G}_1$ in Figure 11 differs from $\mathcal{G}_0$ only in the main part of the game. After the game samples a pre-signature $\tilde{\sigma}$ and $\mathcal{A}_2$ outputs σ^*, we check whether σ^* coincides with the signature obtained by adapting $\tilde{\sigma}$ using y^*, the witness sampled at the beginning of the game. In such a case, the game aborts. This change ensures in the final reduction that the pair (m^*, σ^*) produced by the forger $\mathcal{F}$ is fresh, i.e., $\mathcal{F}$ did not obtain σ^* as an answer when querying the provided signing oracle on m^*.

Note that for the adversary, the game remains the same. We now show that game $\mathcal{G}_1$ aborts only with negligible probability.

$\mathcal{G}_0(\lambda)$

1 : $\mathcal{Q} := \emptyset$
2 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
3 : $(Y^*, y^*) \leftarrow \mathsf{GenR}(1^\lambda)$
4 : $(m^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\mathsf{pk}, Y^*)$
5 : $\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m^*, Y^*)$
6 : $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\tilde{\sigma}, \mathsf{st})$
7 : **return** $(m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*))$

$\mathcal{O}_{\mathsf{Sign}}(m)$

1 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : **return** σ

$\mathcal{O}_{\mathsf{pSign}}(m, b, (Y, y))$

1 : **if** $b = 0$ **then**
2 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : $\quad \tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y^*)$
4 : $\quad$ **return** $\tilde{\sigma}$
5 : **else if** $b = 1 \wedge Y = yG$ **then**
6 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
7 : $\quad \tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y)$
8 : $\quad$ **return** $\tilde{\sigma}$
9 : **else return** $\perp$

Fig. 10: Game $\mathcal{G}_0$.

$\mathcal{G}_1(\lambda)$

1 : $\mathcal{Q} := \emptyset$
2 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
3 : $(Y^*, y^*) \leftarrow \mathsf{GenR}(1^\lambda)$
4 : $(m^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\mathsf{pk}, Y^*)$
5 : $\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m^*, Y^*)$
6 : $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\tilde{\sigma}, \mathsf{st})$
7 : **if** $\mathsf{Adapt}(\tilde{\sigma}, y^*) = \sigma^*$ **then** **abort**
8 : **return** $(m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*))$

$\mathcal{O}_{\mathsf{Sign}}(m)$

1 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : **return** σ

$\mathcal{O}_{\mathsf{pSign}}(m, b, (Y, y))$

1 : **if** $b = 0$ **then**
2 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : $\quad \tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y^*)$
4 : $\quad$ **return** $\tilde{\sigma}$
5 : **else if** $b = 1 \wedge Y = yG$ **then**
6 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
7 : $\quad \tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y)$
8 : $\quad$ **return** $\tilde{\sigma}$
9 : **else return** $\perp$

Fig. 11: Game $\mathcal{G}_1$.

Claim. Assuming DLOG is a hard relation, game $\mathcal{G}_1$ aborts with negligible probability.

Proof. Suppose, to the contrary, that game $\mathcal{G}_1$ aborts with non-negligible probability. Then, using $\mathcal{A}$, we could break the hardness of the DLOG relation with non-negligible probability by finding the discrete logarithm y^* of a uniformly sampled $Y^* \in \mathcal{L}_{\mathsf{DLOG}}$. The PPT solver would proceed as in game $\mathcal{G}_0$, except that it would receive a randomly generated $Y^* \in \mathbb{G}$ as its challenge instead of generating the pair (Y^*, y^*). The solver would then use $\mathcal{A}$ as in $\mathcal{G}_0$ and after obtaining σ^*, it would extract y' with $\mathsf{Ext}(\sigma^*, \tilde{\sigma}, Y^*)$ as defined in Figure 6. With non-negligible probability, σ^* is the adapted signature $\mathsf{Adapt}(\tilde{\sigma}, y^*)$, where y^* is a discrete logarithm of Y^* w.r.t. the generator G. Therefore, by the pre-signature correctness of

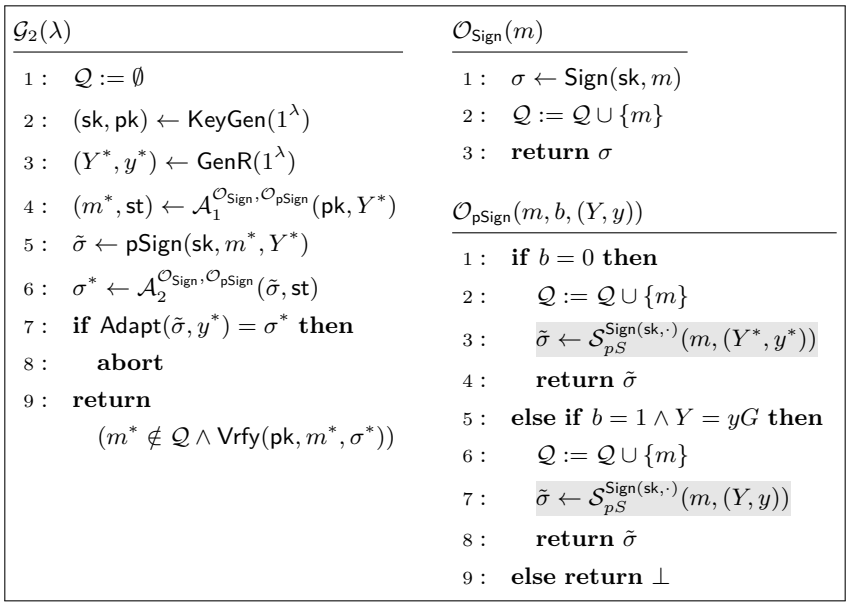

Fig. 12: Game $\mathcal{G}_2$.

$\mathsf{aECDSA}^+_{\mathrm{DLOG}}$, we have $y' = y^*$ with at least the same probability. Thus, there would exist a PPT solver computing the discrete logarithm of Y^* w.r.t. the generator G, breaking the hardness of the relation. This concludes the proof of the claim. ■

Let $\mathsf{negl}(\lambda)$ denote the negligible probability that game $\mathcal{G}_1$ aborts for the adversary $\mathcal{A}$. Then

$$\Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1] \geq \Pr[\mathcal{G}_0^{\mathcal{A}}(\lambda) = 1] - \mathsf{negl}(\lambda).$$

Game $\mathcal{G}_2$: Game $\mathcal{G}_2$ in Figure 12 differs from game $\mathcal{G}_1$ only in the pre-signing oracle. Instead of sampling a pre-signature using the pSign algorithm as in $\mathcal{G}_1$, we simulate the pre-signature in both branches as follows. Note that for both $b = 0$ and $b = 1$, we know the discrete logarithm of the required Y w.r.t. the group generator G. More specifically, when $b = 0$, it is y^* from the DLOG instance (Y^*, y^*) sampled at the beginning of the experiment. When $b = 1$, the adversary must provide the discrete logarithm as part of the query. Thus, knowing the witness y, the simulator can produce a pre-signature $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(m, (Y, y))$ when given black-box access to Sign. By Lemma 1, $\tilde{\sigma}$ is a valid pre-signature on m w.r.t. Y, under the public key pk corresponding to sk, and it is distributed identically to $\mathsf{pSign}(\mathsf{sk}, m, Y)$. Therefore, the adversary cannot distinguish between $\mathcal{O}_{\mathsf{pSign}}$ in games $\mathcal{G}_2$ and $\mathcal{G}_1$. We then have

$$\Pr[\mathcal{G}_2^{\mathcal{A}}(\lambda) = 1] = \Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1].$$

Game $\mathcal{G}_3$: Game $\mathcal{G}_3$ in Figure 13 differs from $\mathcal{G}_2$ only in its main procedure. We now simulate the pre-signature $\tilde{\sigma}$ on m^* w.r.t. the sampled statement Y^* using $\mathcal{S}_{pS}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(m^*, (Y^*, y^*))$ exactly as in the pre-signing oracle of game $\mathcal{G}_2$. Therefore, precisely as in game $\mathcal{G}_2$, by Lemma 1, we have

$$\Pr[\mathcal{G}_3^{\mathcal{A}}(\lambda) = 1] = \Pr[\mathcal{G}_2^{\mathcal{A}}(\lambda) = 1].$$

Forger $\mathcal{F}$: The forger $\mathcal{F}$ in Figure 14 is similar to game $\mathcal{G}_3$ and is constructed to simulate $\mathcal{G}_3$ to the adversary. The differences compared to the game $\mathcal{G}_3$ are in the main part of the algorithm of $\mathcal{F}$. The first change is in the sampling of the key pair $(\mathsf{sk}, \mathsf{pk})$. While $\mathcal{G}_3$ samples it on its own by KeyGen, $\mathcal{F}$ obtains pk on input. It does not know the corresponding sk. As for the next change, the forger $\mathcal{F}$ has black-box access to a signing oracle $\mathcal{O}^+_{\mathsf{Sign}}$, which, on the input message m, samples an ECDSA$^+$ signature on m signed using

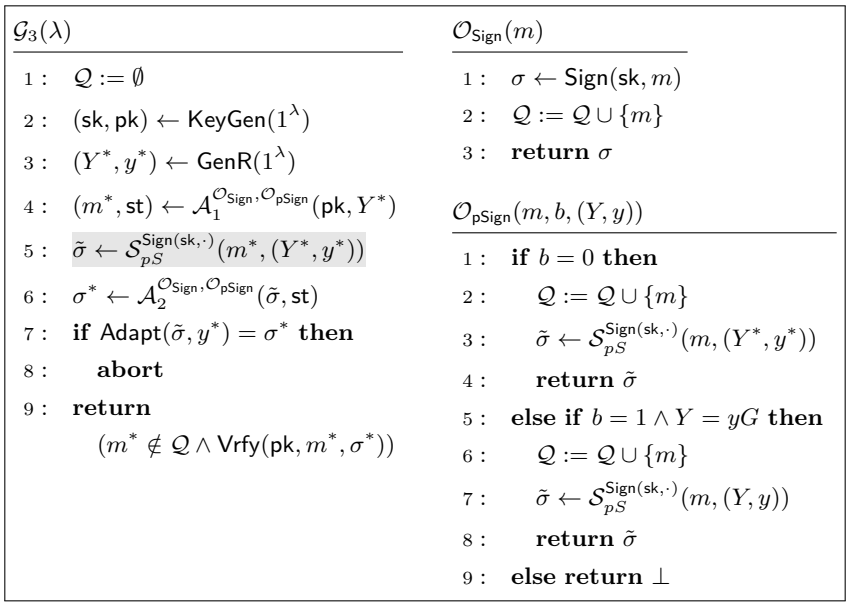

Fig. 13: Game $\mathcal{G}_3$.

the unknown sk. Note that $\mathcal{O}^+_{\mathsf{Sign}}$ is the signing oracle provided in the strong-unforgeability experiment $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$. The forger $\mathcal{F}$ starts by sampling a DLOG instance (Y^*, y^*) using GenR. Then, it passes pk and Y^* to the adversary $\mathcal{A}_1$ on input. The forger continues as game $\mathcal{G}_3$, receiving a chosen message m^* and a state st from $\mathcal{A}_1$, simulating a pre-signature $\tilde{\sigma}$ using $\mathcal{S}_{pS}$ and forwarding $\tilde{\sigma}$ together with the state st to $\mathcal{A}_2$. However, after that, when $\mathcal{A}_2$ produces a signature σ^*, the forger immediately returns the pair (m^*, σ^*). Moreover, when sampling signatures in $\mathcal{O}_{\mathsf{Sign}}$ or simulating pre-signatures by $\mathcal{S}_{pS}$, the forger $\mathcal{F}$ now uses the provided signing oracle $\mathcal{O}^+_{\mathsf{Sign}}$ instead of Sign. Thus, the simulator $\mathcal{S}_{pS}$ has access to query $\mathcal{O}^+_{\mathsf{Sign}}$. This change in the source of signatures is mostly syntactical because $\mathcal{O}^+_{\mathsf{Sign}}$ and $\mathsf{Sign}(\mathsf{sk}, \cdot)$ sample signatures with the same distribution. For clarity, we describe how the forger simulates not only $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{pSign}}$ queries to the adversary, but also the pre-signature $\tilde{\sigma}$.

Signature queries: When $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}$ on a message m, the forger forwards the message to $\mathcal{O}^+_{\mathsf{Sign}}$. Upon $\mathcal{O}^+_{\mathsf{Sign}}$ answering with a signature, the forger returns the received signature to $\mathcal{A}$.

Pre-signature queries and $\tilde{\sigma}$: When $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{pSign}}$ on a message m and a statement Y, the forger forwards m and Y with a corresponding witness y, which is either y^* or a witness provided by $\mathcal{A}$, to the simulator $\mathcal{S}_{pS}$. Since the simulator has access to $\mathcal{O}^+_{\mathsf{Sign}}$, it queries it for a signature on m and upon receiving the signature, it produces a pre-signature on m w.r.t. Y and returns it to $\mathcal{F}$. The forger then forwards the pre-signature to $\mathcal{A}$. When producing the pre-signature $\tilde{\sigma}$, the forger uses $\mathcal{S}_{pS}$ identically.

It is also clear that if the public key pk given to $\mathcal{F}$ on input is generated randomly by KeyGen, then the adversary $\mathcal{A}$ cannot distinguish between the forger $\mathcal{F}$ and game $\mathcal{G}_3$, since then the tuple (pk, Y^*), which $\mathcal{A}$ receives in $\mathcal{G}_3$, as well as $\tilde{\sigma}$ and the signature and pre-signature queries are simulated perfectly.

We now show that the forger $\mathcal{F}$ participates in the strong-unforgeability experiment $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ for ECDSA^+ and wins with probability at least $\Pr[\mathcal{G}_3^{\mathcal{A}}(\lambda) = 1]$ when using $\mathcal{A}$. As described, the forger has access to query the signing oracle $\mathcal{O}^+_{\mathsf{Sign}}$ from $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$. In this experiment, $(\mathsf{sk}, \mathsf{pk})$ is sampled by KeyGen and $\mathcal{F}$ then obtains pk on input. Thus, the public key pk that $\mathcal{A}$ receives through $\mathcal{F}$ is sampled with the correct distribution, and by the above, $\mathcal{A}$ cannot distinguish between the game $\mathcal{G}_3$ and the forger $\mathcal{F}$.

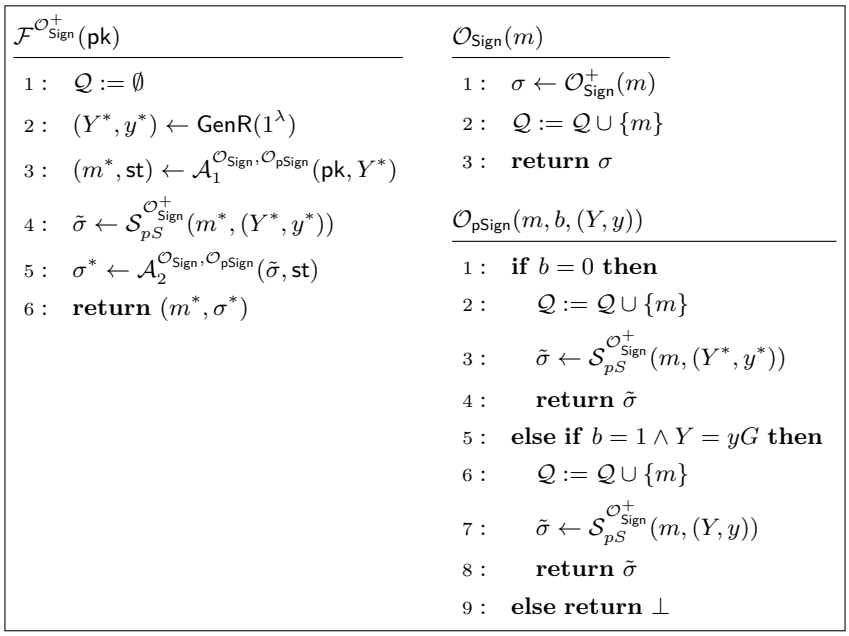

Fig. 14: Forger $\mathcal{F}$.

Moreover, if $\mathcal{A}$ wins game $\mathcal{G}_3$, the signature σ^* produced by $\mathcal{F}$ is a valid signature on the chosen message m^* and $\mathcal{A}$ did not query $\mathcal{O}_{\mathsf{Sign}}$ or $\mathcal{O}_{\mathsf{pSign}}$ on this message. Since these oracles are simulated using $\mathcal{O}^+_{\mathsf{Sign}}$, it means that $\mathcal{F}$ did not query $\mathcal{O}^+_{\mathsf{Sign}}$ on m^* in these simulated queries. The only query on m^* was done by $\mathcal{S}_{pS}$ to $\mathcal{O}^+_{\mathsf{Sign}}$ when simulating the pre-signature $\tilde{\sigma}$. Let us now denote by σ_1 the signature that $\mathcal{O}^+_{\mathsf{Sign}}$ answered. By the construction of $\mathcal{S}_{pS}$, we have

$$\sigma_1 = \mathsf{Adapt}(\tilde{\sigma}, y^*).$$

Since game $\mathcal{G}_3$ did not abort, we know that $\sigma^* \neq \sigma_1$ and thus

$$(m^*, \sigma^*) \neq (m^*, \sigma_1).$$

Therefore, if $\mathcal{A}$ wins $\mathcal{G}_3$, the forger $\mathcal{F}$ wins $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ experiment. We have

$$\Pr[\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}(\lambda) = 1] \geq \Pr[\mathcal{G}_3^{\mathcal{A}}(\lambda) = 1],$$

which implies

$$\begin{aligned}\Pr[\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}(\lambda) = 1] &\geq \Pr[\mathcal{G}_3^{\mathcal{A}}(\lambda) = 1] \\ &\geq \Pr[\mathcal{G}_0^{\mathcal{A}}(\lambda) = 1] - \mathsf{negl}(\lambda) \\ &= \Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}(\lambda) = 1] - \mathsf{negl}(\lambda).\end{aligned}$$

This completes the reduction from the existential unforgeability of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ to the strong unforgeability of ECDSA$^+$. Specifically, for a contradiction, if a PPT adversary $\mathcal{A}$ wins $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}$ with non-negligible probability, we can construct a forger $\mathcal{F}$ using $\mathcal{A}$ that also wins the $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ experiment with non-negligible probability. This would break the strong unforgeability assumption, which concludes the proof.

Proof in the random oracle model. Next, we explain the changes to the above proof when relying on a ROM-based NIZK. By Section 2.1, there exists a NIZK proof for equality of discrete logarithms in the

$\mathcal{O}_{\mathsf{pSign}}(m, b, (Y, y))$

1 : $\mathsf{H}' := \mathsf{H}$
2 : **if** $b = 0$ **then**
3 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
4 : $\quad (\tilde{\sigma}, (G, Y^*, \tilde{K}, K, T_1, T_2) \mapsto c) \leftarrow \mathcal{S}_{pS}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(m, (Y^*, y^*))$
5 : $\quad$ **if** $\mathsf{H}'[G \parallel Y^* \parallel \tilde{K} \parallel K \parallel T_1 \parallel T_2] \neq \bot$ **then**
6 : $\quad\quad$ **abort**
7 : $\quad$ **else**
8 : $\quad\quad \mathsf{H}[G \parallel Y^* \parallel \tilde{K} \parallel K \parallel T_1 \parallel T_2] := c$
9 : $\quad\quad$ **return** $\tilde{\sigma}$
10 : **else if** $b = 1 \wedge Y = yG$ **then**
11 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
12 : $\quad (\tilde{\sigma}, (G, Y, \tilde{K}, K, T_1, T_2) \mapsto c) \leftarrow \mathcal{S}_{pS}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(m, (Y, y))$
13 : $\quad$ **if** $\mathsf{H}'[G \parallel Y \parallel \tilde{K} \parallel K \parallel T_1 \parallel T_2] \neq \bot$ **then**
14 : $\quad\quad$ **abort**
15 : $\quad$ **else**
16 : $\quad\quad \mathsf{H}[G \parallel Y \parallel \tilde{K} \parallel K \parallel T_1 \parallel T_2] := c$
17 : $\quad\quad$ **return** $\tilde{\sigma}$
18 : **else return** $\bot$

Fig. 15: Changes to the pre-signing oracle.

random oracle model. Specifically, we have a construction $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$ of such a proof in Figure 1 with a corresponding simulator. The reduction is analogous to the standard-model reduction above. However, because the existential-unforgeability experiment is now considered in the random oracle model, we must account for the random oracle H available to the adversary. The oracle H is initialized as an empty table and, on query input x, checks whether a value $\mathsf{H}[x]$ has already been defined. If not, it samples a value uniformly at random from $\mathbb{F}_q$, stores it as $\mathsf{H}[x]$, and returns this value. This oracle is accessible to the adversary $\mathcal{A}$ in all games $\mathcal{G}_0$–$\mathcal{G}_3$ as well as to the forger $\mathcal{F}$. We also make the random oracle available to the NIZK proof system, and consider oracle-aided algorithms $(\mathsf{P}^{\mathsf{H}}_{\mathsf{DLEQ}}, \mathsf{V}^{\mathsf{H}}_{\mathsf{DLEQ}})$ and $\mathcal{S}^{\mathsf{H}}_{\mathsf{DLEQ}}$ when producing and verifying proofs. For the reduction, the NIZK proof system and the scheme use the same random oracle H. In what follows, we summarize the most important modifications required for the proof of Theorem 3 in this setting.

- In every game, $\mathcal{A}$ has access to the random oracle H.
- The oracle-aided algorithms of the NIZK proof system in Figure 1 are used for the proofs.
- When the pre-signature simulator $\mathcal{S}_{pS}$ uses $\mathcal{S}^{\mathsf{H}}_{\mathsf{DLEQ}}$ to simulate the NIZK proof, $\mathcal{S}_{pS}$ receives (π_S, f), consisting of a proof and a programming of the random oracle. Moreover, $\mathcal{S}_{pS}$ now outputs the simulated pre-signature together with the programming f. Recall that $\mathcal{S}^{\mathsf{H}}_{\mathsf{DLEQ}}$ on input $(G, Y, \tilde{K}, K) \in \mathcal{L}_{\mathsf{DLEQ}}$ returns a programming
$$f\colon (G, Y, \tilde{K}, K, zG + c\tilde{K}, zY + cK) \mapsto c,$$
for $z, c \leftarrow \mathbb{F}_q$. We stress that here, the programming f is not the same as the projection f used in the ECDSA signature scheme.
- We formally describe the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$ of the games starting from $\mathcal{G}_2$ in Figure 15. When $\mathcal{S}_{pS}$ simulates the pre-signature $\tilde{\sigma}$, the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$ now has to ensure that the programming is consistent with the previous answers of the random oracle. Therefore, if the random oracle has been queried on the input defined by the programming, the game aborts. Otherwise, it programs the random

oracle accordingly and returns the pre-signature to the adversary. By this and the properties of the NIZK proof system, the changes are indistinguishable to the adversary $\mathcal{A}$. Let n_1, n_2, and n_3 denote the numbers of queries that $\mathcal{A}$ makes to the random oracle, the signing oracle, and the pre-signing oracle, respectively. Since $\mathcal{A}$ is a PPT adversary, n_1, n_2, n_3 are polynomial in the security parameter λ. As at least T_1 is a group element chosen uniformly at random, the probability of the game aborting in each $\mathcal{O}_{\mathsf{pSign}}$ call is upper bounded by

$$\frac{n_1 + n_2 + n_3}{q},$$

which is a negligible probability in the security parameter. Then, the probability of game $\mathcal{G}_2$ aborting is also negligible, since $\mathcal{A}$ has only polynomially many $\mathcal{O}_{\mathsf{pSign}}$ queries, and thus there exists a negligible function negl_1 such that

$$\Pr[\mathcal{G}_2^{\mathcal{A}}(\lambda) = 1] \geq \Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1] - \mathsf{negl}_1(\lambda).$$

– The simulation of the pre-signature $\tilde{\sigma}$ in the main part of game $\mathcal{G}_3$ is modified in the same way.
– All described changes transfer also to the forger $\mathcal{F}$. Moreover, the forger $\mathcal{F}$ obtains on input the random oracle H^+ provided by the experiment $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$. The forger then substitutes the uniform sampling in the programming and the answers of H with the answers of H^+.

Considering all changes, we obtain a negligible function negl such that

$$\Pr[\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}(\lambda) = 1] \geq \Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}(\lambda) = 1] - \mathsf{negl}(\lambda),$$

which completes the proof. □

4.2 Witness Extractability of ECDSA Adaptor Signatures

As with robust existential unforgeability for $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$, the robust definition of witness extractability from [AEE+21] assumes an unrestricted pre-signing oracle. For brevity, we omit restating that version and instead present a revised definition tailored to our setting, in which the pre-signing oracle is restricted analogously to the modified definition of existential unforgeability. In this refined definition, the adversary may request pre-signatures on messages either with respect to a fixed statement $S^* \in \mathcal{L}_R$, provided as part of the experiment's input, or with respect to any statement $S \in \mathcal{L}_R$ for which it supplies a valid witness w, i.e., such that $(S, w) \in R$.

Additionally, our definition differs from the robust one in the setup of the challenge. Instead of allowing the adversary to choose the fixed statement S^*, we generate it via the randomized generator GenR for the hard relation and provide both S^* and the corresponding witness w^* to the adversary. This choice enables simulation of pre-signatures in the main part of the experiment and seems crucial for proving witness extractability for $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$.

Definition 11. *An adaptor signature scheme $\mathsf{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R is* witness extractable *if, for every* PPT *adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,*

$$\Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda) = 1] \in \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{aWitExt}_{\mathcal{A},\mathsf{a}\Sigma_R}$ is defined in Figure 16.

The experiment $\mathsf{aWitExt}_{\mathcal{A},\mathsf{a}\Sigma_R}$ in Figure 16 closely mirrors the robust witness extractability experiment defined in [AEE+21]. It starts by sampling a key pair $(\mathsf{sk}, \mathsf{pk})$ using KeyGen. In contrast to [AEE+21], it samples a relation instance (S^*, w^*) using GenR, gives S^* together with pk to $\mathcal{A}_1$, and gives w^* to $\mathcal{A}_2$ together with the pre-signature and state. Throughout the experiment, the adversary can query the signing oracle $\mathcal{O}_{\mathsf{Sign}}$ for signatures on any message, and the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$ for pre-signatures on any message w.r.t. two types of statements. It may either request pre-signatures with respect to S^* by setting a bit $b = 0$, or provide a pair (S, w) with $b = 1$ to request a pre-signature under $S \in \mathcal{L}_R$ for which it knows a valid witness.

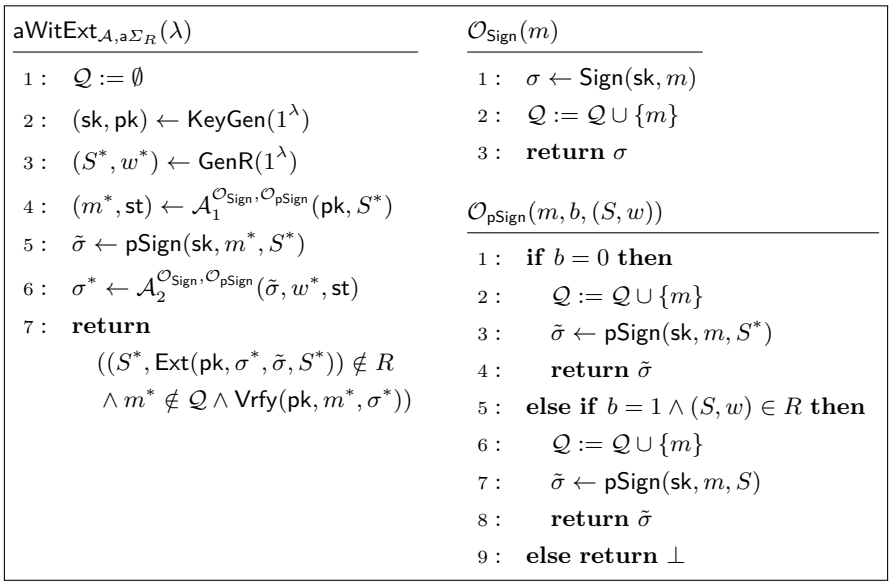

Fig. 16: $\mathsf{aWitExt}_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment.

The adversary wins if σ^* is a valid signature on m^* under pk, if the adversary did not query $\mathcal{O}_{\mathsf{Sign}}$ or $\mathcal{O}_{\mathsf{pSign}}$ for signatures or pre-signatures on m^*, and if the element extracted from σ^*, $\tilde{\sigma}$, and S^* by Ext is not a witness to S^*. Note that since the adversary is given w^*, it can adapt the provided pre-signature $\tilde{\sigma}$ into a valid signature, effectively giving it one "free" forgery. However, this forgery is disqualified by the extraction requirement, which would recover w^* and cause the adversary to lose.

The witness-extractability proof for $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ follows the robust proof for $\mathsf{aECDSA}^+_{\pi\mathsf{DLOG}}$ in [AEE+21], with modifications analogous to those in the proof of Theorem 3. As in the discussion in the previous section, in the following statement, we assume the existence of a NIZK proof of equality of discrete logarithms that allows us to show witness extractability in the standard model.

Theorem 4. *Let* aECDSA^+_{DLOG}*, described in Figure 6, be an adaptor signature scheme for ECDSA*$^+$ *w.r.t. the* DLOG *relation. If the ECDSA*$^+$ *signature scheme is* SUF-CMA *secure and* DLOG *is a hard relation, then* aECDSA^+_{DLOG} *is witness extractable in the random oracle model.*

Additionally, if $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$ *is a perfectly zero-knowledge NIZK proof system for equality of discrete logarithms, then the scheme* aECDSA^+_{DLOG} *is witness extractable in the standard model.*

Proof. For brevity, let us denote the $\mathsf{aWitExt}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}$ experiment by $\mathcal{G}_0$. In the proof, we first gradually modify the game $\mathcal{G}_0$ into games $\mathcal{G}_1, \mathcal{G}_2$ and show that if a PPT adversary $\mathcal{A}$ wins $\mathcal{G}_0$ with probability p, then it wins both $\mathcal{G}_1$ and $\mathcal{G}_2$ with the same probability. Then, we construct a forger $\mathcal{F}$ from $\mathcal{G}_2$ which, using $\mathcal{A}$, forges a signature on a message m^* of its choice and breaks the strong unforgeability property with probability at least p. Such a tight reduction of the witness extractability of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ to the strong unforgeability of ECDSA$^+$ is possible, since we are able to perfectly simulate the pre-signatures of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ w.r.t. any statement Y, if we know the corresponding discrete logarithm of Y. Let

$$\Pr[\mathcal{G}_i^{\mathcal{A}}(\lambda) = 1]$$

be the probability of $\mathcal{A}$ winning the game $\mathcal{G}_i$.

Game $\mathcal{G}_0$: We define game $\mathcal{G}_0$ in Figure 17 to aid the reader in following the proof. Consider now a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ winning game $\mathcal{G}_0$ with the corresponding probability.

$\mathcal{G}_0(\lambda)$

$\mathcal{Q} := \emptyset$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(Y^*, y^*) \leftarrow \mathsf{GenR}(1^\lambda)$
$(m^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\mathsf{pk}, Y^*)$
$\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m^*, Y^*)$
$\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\tilde{\sigma}, y^*, \mathsf{st})$
return $((Y^*, \mathsf{Ext}(\sigma^*, \tilde{\sigma}, Y^*)) \notin \mathsf{DLOG}$
$\wedge\, m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*))$

$\mathcal{O}_{\mathsf{Sign}}(m)$

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
return σ

$\mathcal{O}_{\mathsf{pSign}}(m, b, (Y, y))$

if $b = 0$ **then**
  $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
  $\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y^*)$
  return $\tilde{\sigma}$
else if $b = 1 \wedge Y = yG$ **then**
  $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
  $\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y)$
  return $\tilde{\sigma}$
else return $\perp$

Fig. 17: Game $\mathcal{G}_0$.

$\mathcal{G}_1(\lambda)$

$\mathcal{Q} := \emptyset$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(Y^*, y^*) \leftarrow \mathsf{GenR}(1^\lambda)$
$(m^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\mathsf{pk}, Y^*)$
$\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m^*, Y^*)$
$\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\tilde{\sigma}, y^*, \mathsf{st})$
return $((Y^*, \mathsf{Ext}(\sigma^*, \tilde{\sigma}, Y^*)) \notin \mathsf{DLOG}$
$\wedge\, m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*))$

$\mathcal{O}_{\mathsf{Sign}}(m)$

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
return σ

$\mathcal{O}_{\mathsf{pSign}}(m, b, (Y, y))$

if $b = 0$ **then**
  $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
  $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(m, (Y^*, y^*))$
  return $\tilde{\sigma}$
else if $b = 1 \wedge Y = yG$ **then**
  $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
  $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(m, (Y, y))$
  return $\tilde{\sigma}$
else return $\perp$

Fig. 18: Game $\mathcal{G}_1$.

Game $\mathcal{G}_1$: Game $\mathcal{G}_1$ in Figure 18 differs from game $\mathcal{G}_0$ in the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$. The oracle in game $\mathcal{G}_1$ returns pre-signatures produced by the simulator $\mathcal{S}_{pS}^{\mathsf{Sign}}$ in Figure 9, given black-box access to the ECDSA$^+$ signing algorithm Sign. This is possible since, by the definition of the pre-signing oracle, game $\mathcal{G}_1$ always knows the witness of the statement w.r.t. which $\mathcal{A}$ queries the oracle. It is y^*, the witness for Y^*, if $\mathcal{A}$ chooses $b = 0$, or the witness y of Y provided by the adversary if it sets $b = 1$. By Lemma 1, when given the witness, the simulator outputs valid pre-signatures that are distributed identically to those produced by pSign on the same message and w.r.t. the same DLOG statement. Thus, $\mathcal{G}_0$ and $\mathcal{G}_1$

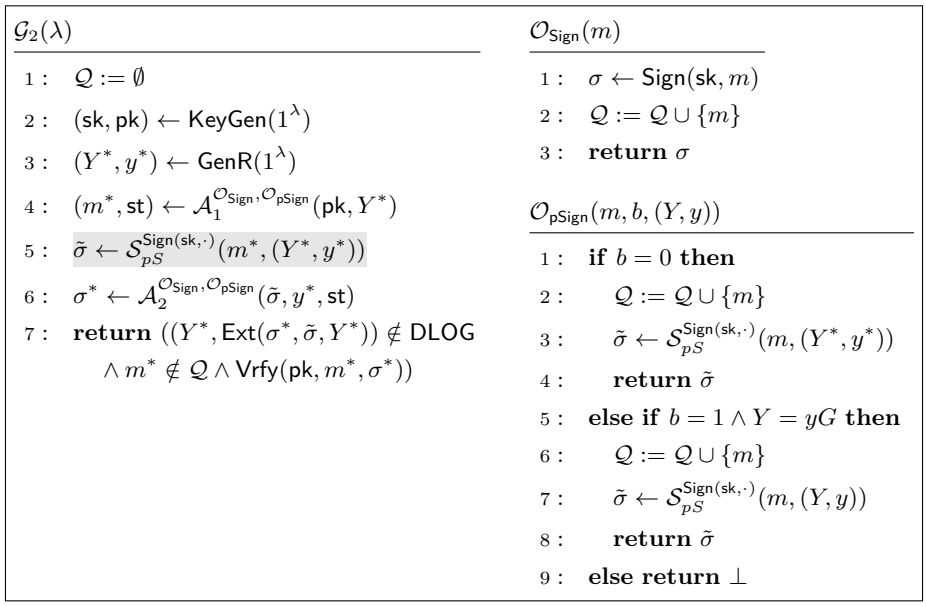

Fig. 19: Game $\mathcal{G}_2$.

are indistinguishable in the view of $\mathcal{A}$ and we have

$$\Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1] = \Pr[\mathcal{G}_0^{\mathcal{A}}(\lambda) = 1].$$

Game $\mathcal{G}_2$: We define game $\mathcal{G}_2$ in Figure 19. It differs from $\mathcal{G}_1$ only in the main part of the game. We now generate the pre-signature $\tilde{\sigma}$ using the simulator $\mathcal{S}_{pS}^{\mathsf{Sign}}$ on the adversary's chosen message m^* and the DLOG pair (Y^*, y^*) sampled at the beginning of the game. Again, by Lemma 1, the adversary cannot distinguish between games $\mathcal{G}_1$ and $\mathcal{G}_2$ and thus

$$\Pr[\mathcal{G}_2^{\mathcal{A}}(\lambda) = 1] = \Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1].$$

Forger $\mathcal{F}$: We construct forger $\mathcal{F}$ to simulate the game $\mathcal{G}_2$ to the adversary $\mathcal{A}$. The forger $\mathcal{F}$ is defined in Figure 20 and differs from $\mathcal{G}_2$ in several places. It takes a public key pk as input and has black-box access to a signing oracle $\mathcal{O}_{\mathsf{Sign}}^+$, which, on a query m, returns a sampled ECDSA$^+$ signature on m under the signing key sk corresponding to pk. As in $\mathcal{G}_2$, the forger samples a DLOG instance (Y^*, y^*) using GenR. It forwards pk and Y^* to $\mathcal{A}_1$ and later gives $\tilde{\sigma}$, y^*, and st to $\mathcal{A}_2$, proceeding otherwise as in $\mathcal{G}_2$. When simulating pre-signatures both in its main procedure and in $\mathcal{O}_{\mathsf{pSign}}$, $\mathcal{F}$ gives $\mathcal{S}_{pS}$ access to $\mathcal{O}_{\mathsf{Sign}}^+$. The same access is also given to the signing oracle $\mathcal{O}_{\mathsf{Sign}}$. For clarity, we now describe how the forger $\mathcal{F}$ simulates signatures and pre-signatures to $\mathcal{A}$.

Signature queries: When $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}$ on m, the forger $\mathcal{F}$ forwards m to $\mathcal{O}_{\mathsf{Sign}}^+$, obtains a signature as an answer and returns this signature to $\mathcal{A}$.

Pre-signature queries and $\tilde{\sigma}$: The pre-signature $\tilde{\sigma}$ in the main part of the forger and all queries of $\mathcal{A}$ to $\mathcal{O}_{\mathsf{pSign}}$ on m w.r.t. Y are simulated as follows. $\mathcal{F}$ forwards m and Y together with the corresponding witness y of Y to $\mathcal{S}_{pS}$, which queries $\mathcal{O}_{\mathsf{Sign}}^+$ on m. Upon $\mathcal{O}_{\mathsf{Sign}}^+$ answering with a signature, $\mathcal{S}_{pS}$ uses this signature to create the required pre-signature and outputs it to $\mathcal{F}$. In the case of the $\mathcal{O}_{\mathsf{pSign}}$ query, the forger then forwards this pre-signature back to $\mathcal{A}$.

However, these changes are syntactic because $\mathcal{O}_{\mathsf{Sign}}^+$ has the same output distribution as $\mathsf{Sign}(\mathsf{sk}, \cdot)$ in $\mathcal{G}_2$. After the adversary $\mathcal{A}_2$ produces σ^*, the forger outputs (m^*, σ^*). If the public key pk given to $\mathcal{F}$ is sampled by KeyGen, then $\mathcal{A}$ cannot distinguish between $\mathcal{G}_2$ and $\mathcal{F}$.

$\mathcal{F}^{\mathcal{O}^+_{\mathsf{Sign}}}(\mathsf{pk})$

1 : $\mathcal{Q} := \emptyset$
2 : $(Y^*, y^*) \leftarrow \mathsf{GenR}(1^\lambda)$
3 : $(m^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\mathsf{pk}, Y^*)$
4 : $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathcal{O}^+_{\mathsf{Sign}}}(m^*, (Y^*, y^*))$
5 : $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\tilde{\sigma}, y^*, \mathsf{st})$
6 : **return** (m^*, σ^*)

$\mathcal{O}_{\mathsf{Sign}}(m)$

1 : $\sigma \leftarrow \mathcal{O}^+_{\mathsf{Sign}}(m)$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : **return** σ

$\mathcal{O}_{\mathsf{pSign}}(m, b, (Y, y))$

1 : **if** $b = 0$ **then**
2 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : $\quad \tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathcal{O}^+_{\mathsf{Sign}}}(m, (Y^*, y^*))$
4 : $\quad$ **return** $\tilde{\sigma}$
5 : **else if** $b = 1 \wedge Y = yG$ **then**
6 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
7 : $\quad \tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathcal{O}^+_{\mathsf{Sign}}}(m, (Y, y))$
8 : $\quad$ **return** $\tilde{\sigma}$
9 : **else return** $\perp$

Fig. 20: Forger $\mathcal{F}$.

It now suffices to show that $\mathcal{F}$ participates in the $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ experiment and wins with probability at least $\Pr[\mathcal{G}_2^{\mathcal{A}}(\lambda) = 1]$ when using $\mathcal{A}$. As in the proof of Theorem 3, we denote the signing oracle in the experiment $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ by $\mathcal{O}^+_{\mathsf{Sign}}$. This is the signing oracle that $\mathcal{F}$ is able to query. The strong-unforgeability experiment samples $(\mathsf{sk}, \mathsf{pk})$ using KeyGen. Thus, by the above, $\mathcal{A}$ cannot distinguish between $\mathcal{G}_2$ and $\mathcal{F}$.

Assume now that $\mathcal{A}$ wins game $\mathcal{G}_2$ for a sampled pair (Y^*, y^*) with a message m^* and a signature σ^*. Then, σ^* is a valid signature on m^* verifiable under pk that was given to $\mathcal{A}$. We also know that $\mathcal{A}$ did not query the signing or pre-signing oracles $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{pSign}}$ on m^*. By the above, these queries are facilitated by forwarding m to $\mathcal{O}^+_{\mathsf{Sign}}$, which implies that $\mathcal{O}^+_{\mathsf{Sign}}$ was not queried on m^* during this process. The only time when $\mathcal{O}^+_{\mathsf{Sign}}$ is queried on m^* is when $\mathcal{S}_{pS}$ simulates the pre-signature $\tilde{\sigma}$ on m^* in the main part of the algorithm of $\mathcal{F}$. The signing oracle $\mathcal{O}^+_{\mathsf{Sign}}$ answers with a signature that we denote by σ_1. By the construction of $\mathcal{S}_{pS}$, we have

$$\sigma_1 = \mathsf{Adapt}(\tilde{\sigma}, y^*).$$

If $\sigma^* = \sigma_1$, then

$$(Y^*, \mathsf{Ext}(\sigma^*, \tilde{\sigma}, Y^*)) = (Y^*, \mathsf{Ext}(\mathsf{Adapt}(\tilde{\sigma}, y^*), \tilde{\sigma}, Y^*)) = (Y^*, y^*) \in \mathsf{DLOG},$$

by the pre-signature correctness of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$. However, since $\mathcal{A}$ wins, we have

$$(Y^*, \mathsf{Ext}(\sigma^*, \tilde{\sigma}, Y^*)) \notin \mathsf{DLOG}.$$

Therefore, $\sigma^* \neq \sigma_1$ and $\mathcal{O}^+_{\mathsf{Sign}}$ never answered σ^* to a query on m^*. Thus, the forger $\mathcal{F}$ wins the experiment $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ with the pair (m^*, σ^*). We have

$$\Pr[\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}(\lambda) = 1] \geq \Pr[\mathcal{G}_2^{\mathcal{A}}(\lambda) = 1].$$

$\mathsf{pSigAdapt}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda)$

1 : $(m, \mathsf{pk}, (S, w), \tilde{\sigma}) \leftarrow \mathcal{A}(1^\lambda)$

2 : **return** $((S, w) \in R \wedge \mathsf{pVrfy}(\mathsf{pk}, m, S, \tilde{\sigma}) = 1$
$\wedge\ \mathsf{Vrfy}(\mathsf{pk}, m, \mathsf{Adapt}(\mathsf{pk}, \tilde{\sigma}, w)) = 0)$

Fig. 21: $\mathsf{pSigAdapt}_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment.

Together with the previous equalities, we have

$$\begin{aligned}\Pr[\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}(\lambda) = 1] &\geq \Pr[\mathcal{G}_2^{\mathcal{A}}(\lambda) = 1] \\ &= \Pr[\mathcal{G}_0^{\mathcal{A}}(\lambda) = 1] \\ &= \Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}(\lambda) = 1]\end{aligned}$$

which concludes the reduction of the witness extractability of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ to the strong unforgeability of ECDSA^+.

Specifically, assume that there exists a PPT adversary $\mathcal{A}$ that wins the $\mathsf{aWitExt}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}$ experiment with a non-negligible probability. Then, we can construct a PPT forger $\mathcal{F}$ that wins $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ with non-negligible probability at least as large, breaking the assumed strong-unforgeability property.

The transition to the random oracle model proceeds analogously to that of Theorem 3. □

4.3 Pre-Signature Adaptability of ECDSA Adaptor Signatures

The notion of pre-signature adaptability captures that a valid pre-signature w.r.t. a statement S should be adaptable into a valid signature using the witness w.

Definition 12 (Perfect pre-signature adaptability [AEE+21]). *An adaptor signature scheme $\mathsf{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R is* perfectly pre-signature adaptable *if, for any message $m \in \{0,1\}^*$, instance $(S, w) \in R$, public key pk, and pre-signature $\tilde{\sigma} \in \{0,1\}^*$ such that $\mathsf{pVrfy}(\mathsf{pk}, m, S, \tilde{\sigma}) = 1$, we have*

$$\mathsf{Vrfy}(\mathsf{pk}, m, \mathsf{Adapt}(\mathsf{pk}, \tilde{\sigma}, w)) = 1.$$

The drawback of the definition of perfect pre-signature adaptability from [AEE+21] is that, in the context of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$, the pre-signature contains a NIZK proof that satisfies only computational soundness. Since computational soundness of the NIZK does not rule out every valid proof of a false statement, there may exist a pre-signature that verifies under a public key and statement but cannot be adapted into a valid signature. Perfect pre-signature adaptability is therefore unsuitable for $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$. Below, we present an alternative computational notion of pre-signature adaptability.

Definition 13. *An adaptor signature scheme $\mathsf{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R is* pre-signature adaptable *if, for every PPT adversary $\mathcal{A}$,*

$$\Pr[\mathsf{pSigAdapt}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda) = 1] \in \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{pSigAdapt}_{\mathcal{A},\mathsf{a}\Sigma_R}$ is defined in Figure 21.

The new definition together with the experiment $\mathsf{pSigAdapt}_{\mathcal{A},\mathsf{a}\Sigma_R}$ in Figure 21 states that no PPT adversary can, with a non-negligible probability, choose a message m, a public key pk, and a valid instance of the relation (S, w), and produce a valid pre-signature on m that fails to adapt into a valid signature.

To prove the statement, we show a reduction of the pre-signature adaptability of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ to the computational soundness of the underlying NIZK proof system.

Theorem 5. *Let* $\mathsf{aECDSA}^{+}_{DLOG}$*, described in Figure 6, be an adaptor signature scheme for ECDSA*$^{+}$ *w.r.t. the* DLOG *relation. Then,* $\mathsf{aECDSA}^{+}_{DLOG}$ *satisfies pre-signature adaptability in the random oracle model.*

Additionally, if $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$ *is a NIZK proof system for equality of discrete logarithms, then the scheme* $\mathsf{aECDSA}^{+}_{DLOG}$ *satisfies pre-signature adaptability in the standard model.*

Proof. To prove the statement, we reduce the pre-signature adaptability of $\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}$ to the computational soundness of the underlying NIZK proof system $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$. Consider a PPT adversary $\mathcal{A}$ that wins the $\mathsf{pSigAdapt}_{\mathcal{A},\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}}$ experiment with probability p.

First, observe that the following holds for any message $m \in \{0,1\}^*$, public key pk, relation instance $(Y, y) \in \mathsf{DLOG}$, and pre-signature $\tilde{\sigma} = (r, \tilde{s}, K, \pi) \in \mathbb{F}_q \times \mathbb{F}_q \times \mathbb{G} \times \{0,1\}^*$. Let $X = \mathsf{pk}$. If $\tilde{\sigma}$ is a valid pre-signature on m under pk and the statement constructed during verification for $\mathsf{V}_{\mathsf{DLEQ}}$ is an actual DLEQ statement, i.e., if

$$\mathsf{pVrfy}(\mathsf{pk}, m, Y, \tilde{\sigma}) = 1 \quad \text{and} \quad (G, Y, \tilde{s}^{-1}\mathsf{H}(m)G + \tilde{s}^{-1}rX, K) \in \mathcal{L}_{\mathsf{DLEQ}},$$

we have $\mathsf{Vrfy}(\mathsf{pk}, m, \mathsf{Adapt}(\tilde{\sigma}, y)) = 1$. To show this, let us denote the element in the statement

$$K' := \tilde{s}^{-1}\mathsf{H}(m)G + \tilde{s}^{-1}rX.$$

Since $(G, Y, K', K) \in \mathcal{L}_{\mathsf{DLEQ}}$, Definition 3 implies that there exists $k \in \mathbb{F}_q$ such that

$$K' = kG \quad \text{and} \quad K = kY.$$

Set $s_0 = \tilde{s}y^{-1}$. By definition, $\mathsf{Adapt}(\tilde{\sigma}, y)$ returns (r, s_0) if s_0 is positive and $(r, -s_0)$ otherwise. Since $\tilde{\sigma}$ is a valid pre-signature, in the first case we get

$$f(y\tilde{s}^{-1}\mathsf{H}(m)G + y\tilde{s}^{-1}rX) = f(yK') = f(ykG) = f(kY) = f(K) = r,$$

making (r, s_0) a valid signature on m. In the second case, verification recomputes the negation of the group element in the display above. For the ECDSA projection f, we have $f(-P) = f(P)$ for every $P \in \mathbb{G}$, so $(r, -s_0)$ is also a valid signature on m.

If a tuple $T = (m, \mathsf{pk}, (Y, y), \tilde{\sigma})$ wins the $\mathsf{pSigAdapt}_{\mathcal{A},\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}}$ experiment, then the DLEQ proof verifies, i.e., $\mathsf{V}_{\mathsf{DLEQ}}((G, Y, K', K), \pi) = 1$, even though $\tilde{T} := (G, Y, K', K) \notin \mathcal{L}_{\mathsf{DLEQ}}$. This yields

$$\begin{aligned} p &\leq \Pr\left[\mathsf{pVrfy}(\mathsf{pk}, m, Y, \tilde{\sigma}) = 1 \wedge \tilde{T} \notin \mathcal{L}_{\mathsf{DLEQ}} \,\middle|\, T \leftarrow \mathcal{A}(1^\lambda)\right] \\ &\leq \Pr\left[\mathsf{V}_{\mathsf{DLEQ}}(\tilde{T}, \pi) = 1 \wedge \tilde{T} \notin \mathcal{L}_{\mathsf{DLEQ}} \wedge \left|\tilde{T}\right| \geq \lambda \,\middle|\, (\tilde{T}, \pi) \leftarrow \mathcal{B}(1^\lambda)\right] \\ &\leq \mathsf{negl}(\lambda), \end{aligned}$$

for a negligible function negl and a PPT adversary $\mathcal{B}$ that uses $\mathcal{A}$ to produce the tuple T, computes $\tilde{T}$ from T and outputs $\tilde{T}$ together with a valid proof π from the pre-signature $\tilde{\sigma}$. Moreover, we have $\left|\tilde{T}\right| \geq \lambda$, since $\log(q) \geq \lambda$. Finally, the last inequality follows from the computational soundness of the underlying NIZK proof system, which concludes the proof.

In the random oracle model, the same reduction applies using the computational soundness of the Fiat–Shamir proof in Figure 1. □

4.4 Pre-Signature Unforgeability of ECDSA Adaptor Signatures

Exactly as in the security definitions of robust existential unforgeability or witness extractability, the unlimited pre-signing oracle of robust pre-signature unforgeability [AEE+21] does not fit the context of $\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}$. We modify the pre-signing oracle analogously to the previous definitions. We define *pre-signature unforgeability under chosen-message attack* (pEUF-CMA security) as follows.

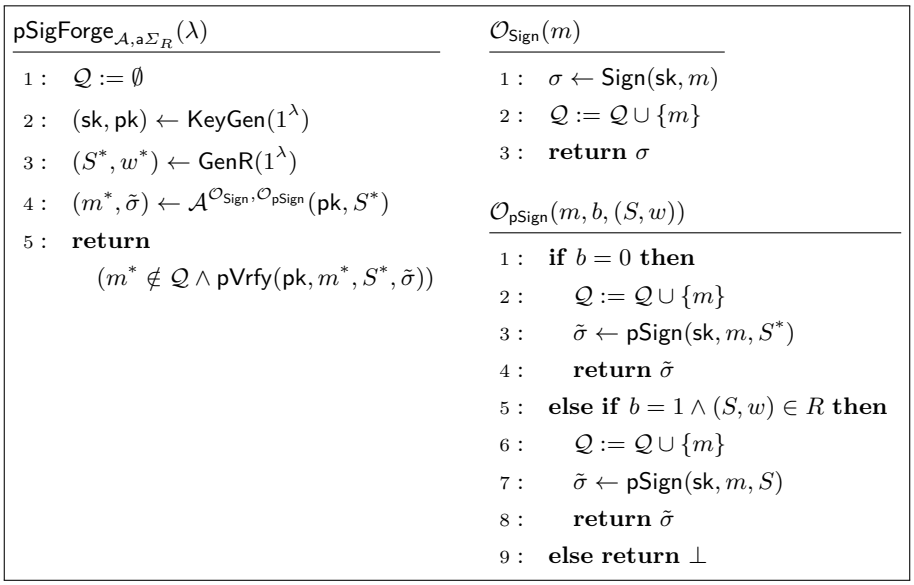

Fig. 22: $\mathsf{pSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment.

Definition 14. *An adaptor signature scheme* $\mathsf{a}\Sigma_R$ *for a signature scheme* Σ *w.r.t. a hard relation* R *satisfies* pre-signature unforgeability under chosen-message attack (pEUF-CMA security) *if, for every* PPT *adversary* $\mathcal{A}$*,*

$$\Pr[\mathsf{pSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda) = 1] \in \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{pSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ *is defined in Figure 22.*

The experiment $\mathsf{pSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ in Figure 22 is analogous to the robust pre-signature-unforgeability experiment $\mathsf{pRSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ from [AEE+21]. The adversary $\mathcal{A}$ aims to produce, under the sampled public key pk, a valid pre-signature on a chosen message m^* w.r.t. the sampled statement S^*. The adversary can ask for signatures on chosen messages, and, as described, for pre-signatures w.r.t. two types of statements, either w.r.t. S^* by setting $b := 0$, or w.r.t. S, which it provides with the witness w to the oracle by choosing $b := 1$. To win, $\mathcal{A}$ cannot query the signing or the pre-signing oracle on the message m^*.

Aumayr et al. [AEE+21] showed that robust pre-signature unforgeability follows from robust existential unforgeability and perfect pre-signature adaptability of the adaptor signature scheme. We present an analogous reduction tailored to our new definitions. While the proof largely mirrors that in [AEE+21], it must reflect the fact that $\mathsf{aECDSA}^{+}_{\mathrm{DLOG}}$ satisfies only a *computational* notion of pre-signature adaptability. At this point in the proof, we need to construct an adversary that would break the assumed pre-signature adaptability, resulting in a contradiction. We now present a formal proof.

Theorem 6. *Let* $\mathsf{a}\Sigma_R$ *be an adaptor signature scheme for a signature scheme* Σ *w.r.t. a hard relation* R*. If* $\mathsf{a}\Sigma_R$ *satisfies* aEUF-CMA *security and pre-signature adaptability, then* $\mathsf{a}\Sigma_R$ *is* pEUF-CMA *secure.*

Proof. We show a reduction of pre-signature unforgeability to existential unforgeability of $\mathsf{a}\Sigma_R$ assuming pre-signature adaptability. For brevity, we denote the $\mathsf{pSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment in Figure 22 by $\mathcal{G}_0$. Consider a PPT adversary $\mathcal{A}$ that wins the $\mathsf{pSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment with probability p.

First, we modify game $\mathcal{G}_0$ into game $\mathcal{G}_1$ in Figure 23. Game $\mathcal{G}_1$ differs from the original experiment in the winning requirements. For $\mathcal{A}$ to win $\mathcal{G}_1$, the returned pre-signature must now be adaptable into a valid signature on m^* under pk. It is clear that $\mathcal{A}$ cannot distinguish between games $\mathcal{G}_0$ and $\mathcal{G}_1$. We show that the difference between p and the probability of $\mathcal{A}$ winning $\mathcal{G}_1$ is negligible in the security parameter λ. Denote the latter probability by

$$\Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1].$$

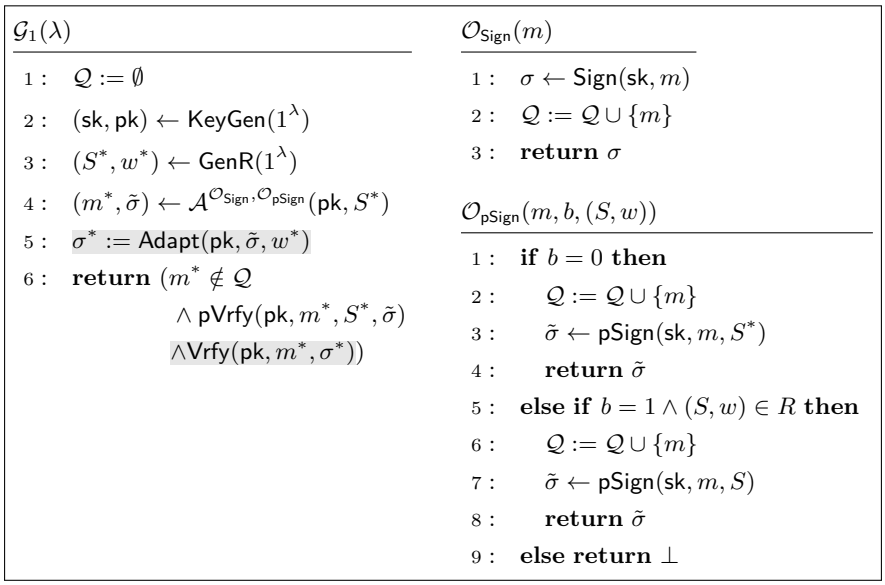

Fig. 23: Game $\mathcal{G}_1$.

We want to show that there exists a negligible function negl_1 such that

$$\mathsf{negl}_1(\lambda) \geq p - \Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1].$$

Assume there exists a non-negligible function ν such that

$$p - \Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1] \geq \nu(\lambda).$$

By the construction of the games, the difference is precisely the probability that $m^* \notin \mathcal{Q}$ and that the pre-signature $\tilde{\sigma}$ is valid but not adaptable into a valid signature on m^*. We use this to break the pre-signature adaptability of $\mathsf{a}\Sigma_R$. We construct a PPT adversary $\mathcal{B}$ formally described in Figure 24. This adversary proceeds identically to game $\mathcal{G}_1$; however, after $\mathcal{B}$ uses $\mathcal{A}$ to produce a message m^* and a pre-signature $\tilde{\sigma}$ as in $\mathcal{G}_1$, it immediately returns a tuple $(m^*, \mathsf{pk}, (S^*, w^*), \tilde{\sigma})$. The adversary $\mathcal{B}$ participates in the $\mathsf{pSigAdapt}_{\mathcal{B}, \mathsf{a}\Sigma_R}$ experiment in Figure 21, and, since (S^*, w^*) is sampled by GenR, it is a valid instance of the relation R. By the assumption above, $\mathcal{B}$ wins the experiment with probability

$$\Pr[\mathsf{pSigAdapt}_{\mathcal{B}, \mathsf{a}\Sigma_R}(\lambda) = 1] \geq p - \Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1] \geq \nu(\lambda),$$

which breaks the assumed pre-signature adaptability. Thus, there exists a negligible function negl_1 such that

$$\Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1] \geq p - \mathsf{negl}_1(\lambda).$$

Now, we construct a PPT forger $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2)$ that wins the experiment $\mathsf{aSigForge}_{\mathcal{F}, \mathsf{a}\Sigma_R}$ using $\mathcal{A}$. We describe $\mathcal{F}$ formally in Figure 25. Both $\mathcal{F}_1$ and $\mathcal{F}_2$ can query the signing and pre-signing oracles defined in the $\mathsf{aSigForge}_{\mathcal{F}, \mathsf{a}\Sigma_R}$ experiment; we denote these oracles by $\mathcal{O}'_{\mathsf{Sign}}$ and $\mathcal{O}'_{\mathsf{pSign}}$, respectively. $\mathcal{F}_1$ proceeds similarly to game $\mathcal{G}_1$; however, it does not sample a key pair $(\mathsf{sk}, \mathsf{pk})$ using KeyGen as in $\mathcal{G}_1$. Instead, it gets a public key pk together with a statement $\tilde{S} \in \mathcal{L}_R$ on input. It begins with sampling a relation instance (S^*, w^*) using GenR and then it forwards the public key pk and the sampled statement S^* to the adversary $\mathcal{A}$, which on this input returns a message m^* and a pre-signature $\tilde{\sigma}$ as in $\mathcal{G}_1$. Next, $\mathcal{F}_1$ computes $\sigma^* := \mathsf{Adapt}(\mathsf{pk}, \tilde{\sigma}, w^*)$ and stores it as the state st. $\mathcal{F}_1$ returns a tuple (m^*, st). The adversary $\mathcal{A}$ used in $\mathcal{F}_1$ has access to query the signing and pre-signing oracles $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{pSign}}$ as in game $\mathcal{G}_1$ with small changes. Instead of sampling signatures and pre-signatures with Sign and pSign, $\mathcal{F}_1$ obtains them by querying $\mathcal{O}'_{\mathsf{Sign}}$ and $\mathcal{O}'_{\mathsf{pSign}}$.

$\mathcal{B}(\lambda)$

1 : $\mathcal{Q} := \emptyset$
2 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
3 : $(S^*, w^*) \leftarrow \mathsf{GenR}(1^\lambda)$
4 : $(m^*, \tilde{\sigma}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\mathsf{pk}, S^*)$
5 : **return** $(m^*, \mathsf{pk}, (S^*, w^*), \tilde{\sigma})$

$\mathcal{O}_{\mathsf{Sign}}(m)$

1 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : **return** σ

$\mathcal{O}_{\mathsf{pSign}}(m, b, (S, w))$

1 : **if** $b = 0$ **then**
2 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : $\quad \tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, S^*)$
4 : $\quad$ **return** $\tilde{\sigma}$
5 : **else if** $b = 1 \wedge (S, w) \in R$ **then**
6 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
7 : $\quad \tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, S)$
8 : $\quad$ **return** $\tilde{\sigma}$
9 : **else return** $\bot$

Fig. 24: Adversary $\mathcal{B}$.

Signature queries: To answer a query of $\mathcal{A}$ to $\mathcal{O}_{\mathsf{Sign}}$ on a message m, the forger $\mathcal{F}_1$ simply queries $\mathcal{O}'_{\mathsf{Sign}}$ on m and forwards the answered signature to $\mathcal{A}$.

Pre-signature queries: The adversary $\mathcal{A}$ can still query $\mathcal{O}_{\mathsf{pSign}}$ for pre-signatures on m w.r.t. the sampled fixed statement S^*. To answer those queries, since $\mathcal{F}_1$ knows the corresponding witness w^*, it samples a pre-signature using $\mathcal{O}'_{\mathsf{pSign}}$ on m and the pair (S^*, w^*). On the other hand, if $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{pSign}}$ for a pre-signature w.r.t. a pair $(S, w) \in R$, $\mathcal{F}_1$ just forwards the query to $\mathcal{O}'_{\mathsf{pSign}}$ and returns the answer to $\mathcal{A}$.

As explained in the previous reductions, the adversary $\mathcal{A}$ cannot distinguish between the oracles in game $\mathcal{G}_1$ and those provided by $\mathcal{F}_1$. If pk is sampled by KeyGen, the adversary cannot distinguish between $\mathcal{G}_1$ and $\mathcal{F}_1$. Note that $\mathcal{F}_1$ ignores the statement $\tilde{S}$ on input.

The other part of the forger, $\mathcal{F}_2$, obtains a pre-signature $\tilde{\sigma}'$ and a state st on input. It can also query $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{pSign}}$. However, it immediately returns the signature σ^* encoded in st by $\mathcal{F}_1$.

We now show that $\mathcal{F}$ wins the $\mathsf{aSigForge}_{\mathcal{F},\mathsf{a}\Sigma_R}$ experiment with sufficient probability. In the experiment, pk is sampled correctly by KeyGen. Thus, with at least the probability that $\mathcal{A}$ wins $\mathcal{G}_1$, σ^* is a valid signature on m^* under pk, and $\mathcal{A}$ has queried neither oracle on m^*. Consequently, $\mathcal{F}$ wins the $\mathsf{aSigForge}_{\mathcal{F},\mathsf{a}\Sigma_R}$ experiment using $\mathcal{A}$, and we have

$$\Pr[\mathsf{aSigForge}_{\mathcal{F},\mathsf{a}\Sigma_R}(\lambda) = 1] \geq \Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1].$$

Together, we obtain

$$\begin{aligned}\Pr[\mathsf{aSigForge}_{\mathcal{F},\mathsf{a}\Sigma_R}(\lambda) = 1] &\geq \Pr[\mathcal{G}_1^{\mathcal{A}}(\lambda) = 1] \\ &\geq \Pr[\mathsf{pSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda) = 1] - \mathsf{negl}_1(\lambda),\end{aligned}$$

for some negligible function negl_1. Therefore, if a PPT adversary $\mathcal{A}$ wins the $\mathsf{pSigForge}_{\mathcal{A},\mathsf{a}\Sigma_R}$ experiment with non-negligible probability and thus breaks the pre-signature unforgeability of $\mathsf{a}\Sigma_R$, we can construct a PPT forger $\mathcal{F}$ that wins the $\mathsf{aSigForge}_{\mathcal{F},\mathsf{a}\Sigma_R}$ experiment with non-negligible probability. This breaks the existential unforgeability of $\mathsf{a}\Sigma_R$ and concludes the proof. □

Specifically for $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$, we have existential unforgeability by Theorem 3 and pre-signature adaptability by Theorem 5. Thus, pre-signature unforgeability of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ follows as a corollary of Theorem 6.

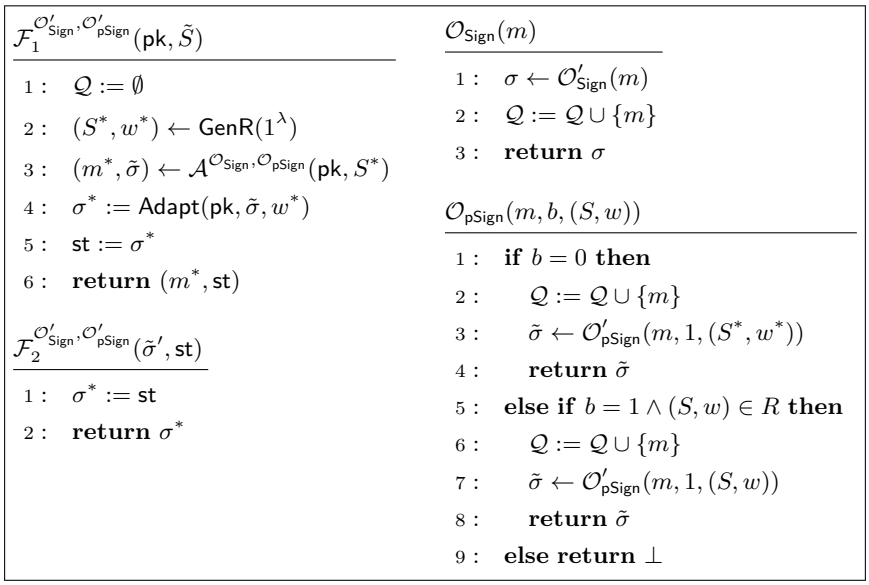

Fig. 25: Forger $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2)$.

Corollary 1. *Let* aECDSA^+_{DLOG}*, described in Figure 6, be an adaptor signature scheme for ECDSA*$^+$ *w.r.t. the* DLOG *relation. If the ECDSA*$^+$ *signature scheme is* SUF-CMA *secure and* DLOG *is a hard relation, then* aECDSA^+_{DLOG} *is* pEUF-CMA *secure in the random oracle model.*

Additionally, if $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$ *is a perfectly zero-knowledge NIZK proof system for equality of discrete logarithms, then the scheme* aECDSA^+_{DLOG} *is* pEUF-CMA *secure in the standard model.*

4.5 Extractability of ECDSA Adaptor Signatures

As adaptor signatures were applied in broader contexts, it became evident that standard definitions of unforgeability and witness extractability were insufficient to prevent additional forms of leakage or misuse. To address these limitations, Dai et al. [DOY22] introduced a stronger notion of *extractability*, which unifies the guarantees of unforgeability and witness extractability. Crucially, this definition mitigates the leaky pre-signature vulnerability formalized by Gerhart et al. [GSST24]. In prior models, the adversary receives a single pre-signature on the target message m^* regarding the sampled statement S and is prohibited from querying the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$ on m^*. Consequently, a scheme could leak partial signature information within the pre-signature yet still be deemed secure under those weaker definitions. The notion of extractability resolves this by permitting the adversary to obtain multiple pre-signatures on the target message m^*. We recall their definition below.

Definition 15. *An adaptor signature scheme* $\mathsf{a}\Sigma_R$ *for a signature scheme* Σ *w.r.t. a hard relation* R *satisfies* robust extractability *if, for every* PPT *adversary* $\mathcal{A}$*,*

$$\Pr[\mathsf{RExt}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda) = 1] \in \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{RExt}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda)$ *is defined in Figure 26.*

Again, robust extractability allows the adversary to query the pre-signing oracle on any statement S, without requiring a corresponding witness w. Below, we present our definition of extractability, which allows only restricted access to the pre-signing oracle.

$\mathsf{RExt}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda)$

$\mathcal{Q}_1 := \emptyset, \mathcal{Q}_2 := \emptyset, b := 1$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{NewY}}, \mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\mathsf{pk})$
assert $\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*)$
assert $(m^* \notin \mathcal{Q}_1)$
for $(S, \tilde{\sigma}) \in \mathcal{Q}_2[m^*]$:
  if $(S, \mathsf{Ext}(\mathsf{pk}, \sigma^*, \tilde{\sigma}, S)) \in R$
    $b := 0$
return b

$\mathcal{O}_{\mathsf{NewY}}(\lambda)$

$(S, w) \leftarrow \mathsf{GenR}(1^\lambda)$
return S

$\mathcal{O}_{\mathsf{Sign}}(m)$

$\mathcal{Q}_1 := \mathcal{Q}_1 \cup \{m\}$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
return σ

$\mathcal{O}_{\mathsf{pSign}}(m, S)$

$\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, S)$
$\mathcal{Q}_2[m] := \mathcal{Q}_2[m] \cup \{(S, \tilde{\sigma})\}$
return $\tilde{\sigma}$

Fig. 26: $\mathsf{RExt}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda)$ experiment of Dai et al. [DOY22].

$\mathsf{Ext}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda)$

$\mathcal{Q}_1 := \emptyset, \mathcal{Q}_2 := \emptyset, b := 1$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{NewY}}, \mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{pSign}}}(\mathsf{pk})$
assert $\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*)$
assert $(m^* \notin \mathcal{Q}_1)$
for $(S, \tilde{\sigma}) \in \mathcal{Q}_2[m^*]$:
  if $(S, \mathsf{Ext}(\mathsf{pk}, \sigma^*, \tilde{\sigma}, S)) \in R$
    $b := 0$
return b

$\mathcal{O}_{\mathsf{NewY}}(\lambda)$

$(S, w) \leftarrow \mathsf{GenR}(1^\lambda)$
return S

$\mathcal{O}_{\mathsf{Sign}}(m)$

$\mathcal{Q}_1 := \mathcal{Q}_1 \cup \{m\}$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
return σ

$\mathcal{O}_{\mathsf{pSign}}(m, (S, w))$

if $(S, w) \notin R$ **then**
  return $\perp$
$\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, S)$
$\mathcal{Q}_2[m] := \mathcal{Q}_2[m] \cup \{(S, \tilde{\sigma})\}$
return $\tilde{\sigma}$

Fig. 27: $\mathsf{Ext}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda)$ experiment.

Definition 16. *An adaptor signature scheme* $\mathsf{a}\Sigma_R$ *for a signature scheme* Σ *w.r.t. a hard relation* R *is* extractable *if, for every* PPT *adversary* $\mathcal{A}$*,*

$$\Pr[\mathsf{Ext}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda) = 1] \in \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Ext}_{\mathcal{A},\mathsf{a}\Sigma_R}(\lambda)$ *is defined in Figure 27.*

Note that this context differs from previous settings, where we allowed two types of queries: those on a fixed sampled statement S^* and those requiring a witness. In extractability, however, no fixed statement-witness pair is sampled. Instead, the adversary must output a message–signature pair (m^*, σ^*) such that extraction fails for every statement–pre-signature pair recorded for m^* by the pre-signing oracle.

We show that, under the assumption of strong unforgeability, the adaptor signature scheme $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ satisfies extractability.

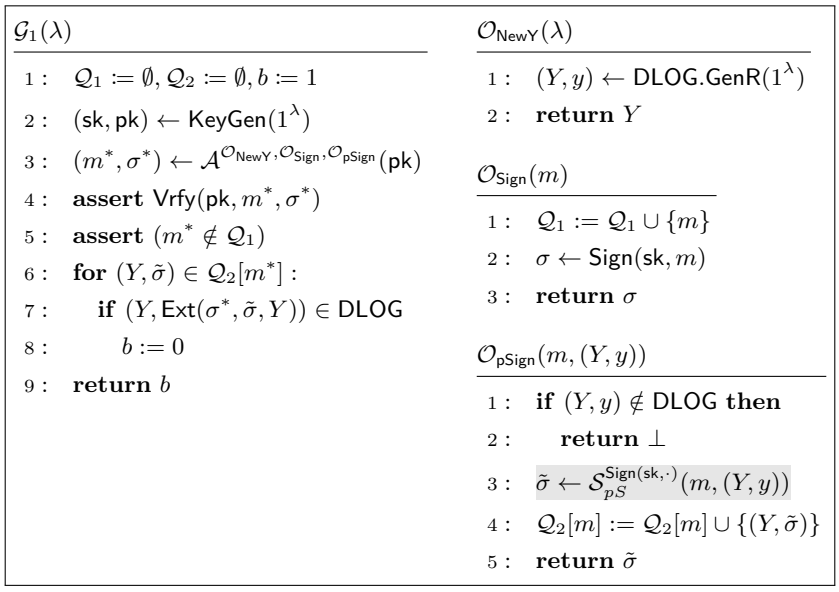

Fig. 28: Game $\mathcal{G}_1$.

Theorem 7. *Let* aECDSA^+_{DLOG}*, described in Figure 6, be an adaptor signature scheme for ECDSA*$^+$ *w.r.t. the* DLOG *relation. If the ECDSA*$^+$ *signature scheme is* SUF-CMA *secure and* DLOG *is a hard relation, then* aECDSA^+_{DLOG} *is extractable in the random oracle model.*

Additionally, if $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$ *is a perfectly zero-knowledge NIZK proof system for equality of discrete logarithms, then the scheme* aECDSA^+_{DLOG} *is extractable in the standard model.*

Proof. We reduce the extractability of $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$ to the strong unforgeability of the underlying ECDSA$^+$ signature scheme. Consider first a PPT adversary $\mathcal{A}$ winning the experiment $\mathsf{Ext}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}$ with probability

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Ext}}(\lambda) := \Pr[\mathsf{Ext}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}(\lambda) = 1].$$

For brevity, we denote the experiment $\mathsf{Ext}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}$ as game $\mathcal{G}_0$. We then modify game $\mathcal{G}_0$ to construct a forger $\mathcal{F}$ that uses $\mathcal{A}$ to produce a fresh message–signature pair (m^*, σ^*) and thus wins the experiment $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$.

We define game $\mathcal{G}_1$ in Figure 28. It differs from game $\mathcal{G}_0$ only in the work of the pre-signing oracle $\mathcal{O}_{\mathsf{pSign}}$. Instead of sampling a pre-signature on the given message m w.r.t. a DLOG statement Y, game $\mathcal{G}_1$ uses $\mathcal{S}_{pS}$ to simulate the pre-signature. This is possible because, when the adversary $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{pSign}}$, it provides together with Y a discrete logarithm y of Y w.r.t. G. The simulator $\mathcal{S}_{pS}$ is given black-box access to the signing algorithm Sign of the underlying ECDSA$^+$ signature scheme. By Lemma 1, the adversary $\mathcal{A}$ cannot distinguish between the pre-signatures answered by the pre-signing oracles in $\mathcal{G}_0$ and $\mathcal{G}_1$. Therefore, we have

$$\mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda) = \mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_0}(\lambda).$$

We can now construct the forger $\mathcal{F}$ playing $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ in a straightforward way from game $\mathcal{G}_1$. The forger is defined in Figure 29. In comparison to $\mathcal{G}_1$, $\mathcal{F}$ does not sample the key pair $(\mathsf{sk}, \mathsf{pk})$; instead, it receives pk as its challenge. It also has access to query the signing oracle from the experiment $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$, which we, for clarity, denote by $\mathcal{O}'_{\mathsf{Sign}}$. The remaining changes are syntactic: the forger uses $\mathcal{O}'_{\mathsf{Sign}}$ to answer queries to $\mathcal{O}_{\mathsf{Sign}}$ and to simulate pre-signatures in $\mathcal{O}_{\mathsf{pSign}}$.

Since pk is sampled by KeyGen, the forger $\mathcal{F}$ perfectly simulates $\mathcal{G}_1$ for the adversary $\mathcal{A}$. Next, we observe that if $\mathcal{A}$ produces (m^*, σ^*) that wins $\mathcal{G}_1$, the tuple also wins $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$. By the conditions in

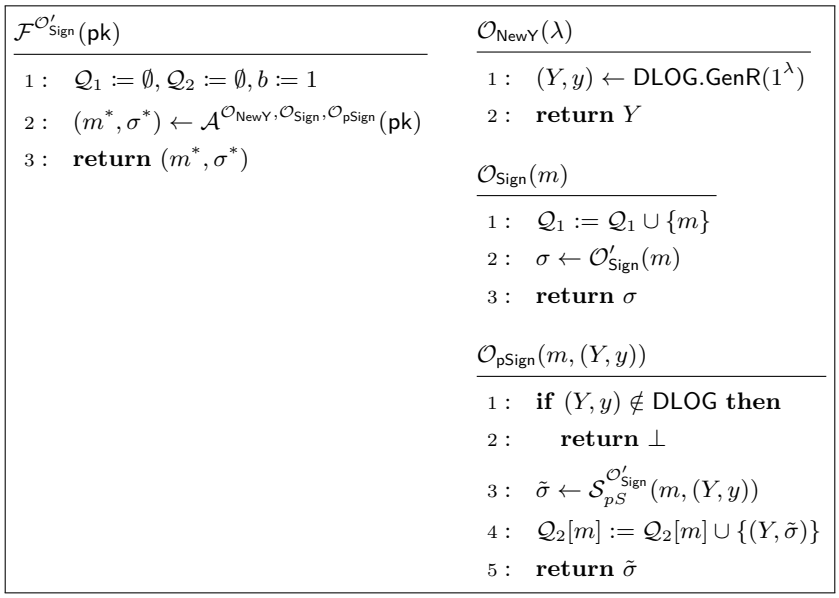

Fig. 29: Forger $\mathcal{F}$.

$\mathcal{G}_1$, (m^*, σ^*) is a valid signature on m^* that verifies under pk. Moreover, since $m^* \notin \mathcal{Q}_1$, the adversary did not query $\mathcal{O}_{\mathsf{Sign}}$ (and thus also $\mathcal{O}'_{\mathsf{Sign}}$) on m^*. Thus, it suffices to show that the query made by $\mathcal{S}_{pS}$ to $\mathcal{O}'_{\mathsf{Sign}}$ on m^* did not return σ^*. However, this follows from the remaining winning condition of $\mathcal{G}_1$. Assume $\mathcal{S}_{pS}$ queried $\mathcal{O}'_{\mathsf{Sign}}$ on m^*, obtained σ^* and produced a pre-signature $\tilde{\sigma}$. By the construction of $\mathcal{S}_{pS}$, we have

$$\sigma^* = \mathsf{Adapt}(\tilde{\sigma}, y),$$

where y is the discrete logarithm of Y with respect to which the pre-signature is simulated. By pre-signature correctness, it follows that

$$(Y, \mathsf{Ext}(\sigma^*, \tilde{\sigma}, Y)) = (Y, \mathsf{Ext}(\mathsf{Adapt}(\tilde{\sigma}, y), \tilde{\sigma}, Y)) \in \mathsf{DLOG}.$$

We could thus extract a valid witness for Y. However, as (m^*, σ^*) wins game $\mathcal{G}_1$, this is a contradiction. Therefore, (m^*, σ^*) is a fresh tuple and wins the experiment of strong unforgeability, resulting in

$$\mathsf{Adv}^{\mathcal{F}}_{\mathsf{strongSigForge}}(\lambda) \geq \mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda).$$

Together, we have

$$\mathsf{Adv}^{\mathcal{F}}_{\mathsf{strongSigForge}}(\lambda) \geq \mathsf{Adv}^{\mathcal{A}}_{\mathsf{Ext}}(\lambda),$$

which shows that, if a PPT adversary $\mathcal{A}$ wins $\mathsf{Ext}_{\mathcal{A},\mathsf{aECDSA}^+_{\mathrm{DLOG}}}$ with non-negligible probability, the corresponding forger $\mathcal{F}$ also wins $\mathsf{strongSigForge}_{\mathcal{F},\mathrm{ECDSA}^+}$ with non-negligible probability. This contradicts the assumed strong unforgeability of ECDSA^+, which concludes the proof.

The transition to the random oracle model proceeds analogously to that of Theorem 3. □

Note that our extractability game and the robust variant share the same winning condition. The only difference lies in the design of the pre-signing oracle: in our setting, each query must be accompanied by a valid witness, while the robust oracle accepts arbitrary statements without requiring one. As a result, robust extractability implies extractability, since one can simulate an extractability adversary using the robust oracle by simply discarding the witness.

Observation 8 *Let $\mathsf{a}\Sigma_R$ be an adaptor signature for a signature scheme Σ w.r.t. a relation R. If $\mathsf{a}\Sigma_R$ satisfies robust extractability, then $\mathsf{a}\Sigma_R$ satisfies extractability.*

$\mathsf{WitHid}_{\mathcal{A},\mathsf{Sim},\mathsf{a}\Sigma_R,b}(\lambda)$

1 : $(Y, y) \leftarrow \mathsf{GenR}(1^\lambda)$
2 : **return** $\mathcal{A}^{\mathsf{Chall}_b}(Y)$

$\mathsf{Chall}_0(\mathsf{sk}, \mathsf{pk}, m)$

1 : **if** $(\mathsf{sk}, \mathsf{pk})$ is not a valid key pair **then**
2 : **return** $\perp$
3 : $\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, Y)$
4 : $\sigma \leftarrow \mathsf{Adapt}(\mathsf{pk}, \tilde{\sigma}, y)$
5 : **return** σ

$\mathsf{Chall}_1(\mathsf{sk}, \mathsf{pk}, m)$

1 : **if** $(\mathsf{sk}, \mathsf{pk})$ is not a valid key pair **then**
2 : **return** $\perp$
3 : $\sigma \leftarrow \mathsf{Sim}(\mathsf{sk}, \mathsf{pk}, m, Y)$
4 : **return** σ

Fig. 30: $\mathsf{WitHid}_{\mathcal{A},\mathsf{Sim},\mathsf{a}\Sigma_R,b}(\lambda)$ experiment.

4.6 Witness Hiding of ECDSA Adaptor Signatures

In blockchain applications, adapted signatures become public, and it is important to ensure that they cannot be used for extracting the associated witness without the pre-signatures. Witness hiding, introduced by Liu et al. [LTZ24], captures this property. The adversary aims to distinguish whether it receives simulated signatures or signatures that are adapted from pre-signatures. The adversary has the additional power to choose a valid key pair that will be used for signing, which makes the adversary stronger than just an eavesdropper. Formally, we define witness hiding next.

Definition 17. *An adaptor signature scheme* $\mathsf{a}\Sigma_R$ *for a signature scheme* Σ *w.r.t. a hard relation* R *is* witness hiding *if there exists a* PPT *simulator* Sim *such that, for every* PPT *adversary* $\mathcal{A}$*,*

$$|\Pr[\mathsf{WitHid}_{\mathcal{A},\mathsf{Sim},\mathsf{a}\Sigma_R,0}(\lambda) = 1] - \Pr[\mathsf{WitHid}_{\mathcal{A},\mathsf{Sim},\mathsf{a}\Sigma_R,1}(\lambda) = 1]| \in \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{WitHid}_{\mathcal{A},\mathsf{Sim},\mathsf{a}\Sigma_R,b}(\lambda)$*,* $b \in \{0, 1\}$*, is defined in Figure 30.*

In the context of adaptor signatures for ECDSA, witness hiding is straightforward. Consider the simulator Sim that simply returns a signature on the given message using the ECDSA signing algorithm and the given signing key. Then, the adapted signatures are distributed identically to the signatures produced by the simulator, which establishes the following theorem.

Theorem 9. *Let* aECDSA^+_{DLOG}*, described in Figure 6, be an adaptor signature scheme for ECDSA*$^+$ *w.r.t. the* DLOG *relation. Then,* aECDSA^+_{DLOG} *is witness hiding.*

5 Relative Efficiency

In addition to security, when using adaptor signatures in practice, we are interested in the efficiency of the respective schemes. In the schemes we considered, the most computationally intensive tasks are generating and verifying pre-signatures. These operations are often performed repeatedly in real-world applications, sometimes at large scale. For the same reason, the sizes of the pre-signatures and associated relation instances also play a critical role in overall performance. In this section, we compare the efficiency of the Schnorr adaptor signature, the ECDSA adaptor signature, and the extended ECDSA adaptor signature from [AEE+21].

Operation \ AS algorithm	pSign		
	$\mathsf{aSchnorr}_{\mathrm{DLOG}}$	$\mathsf{aECDSA}_{\mathrm{DLOG}}$	$\mathsf{aECDSA}_{\pi\mathrm{DLOG}}$
Gen. rand. el. of $\mathbb{F}_q$	1	2	2
Scalar mult. in $\mathbb{G}$	1	4	4
Group op. in $\mathbb{G}$	1	-	-
$+_{\mathbb{F}_q}$	1	1	1
$-_{\mathbb{F}_q}$	-	1	1
$\cdot_{\mathbb{F}_q}$	1	3	3
$(\cdot)^{-1}_{\mathbb{F}_q}$	-	1	1
$\mathsf{H}(\cdot)$	1	2	2
$f(\cdot)$	-	1	1
Size of pre-signature	$2\lvert\mathbb{F}_q\rvert$	$\lvert\mathbb{G}\rvert + 4\lvert\mathbb{F}_q\rvert$	$\lvert\mathbb{G}\rvert + 4\lvert\mathbb{F}_q\rvert$

Table 3: Computational complexity of pSign and size of pre-signatures of adaptor signatures.

Operation \ AS algorithm	pVrfy		
	$\mathsf{aSchnorr}_{\mathrm{DLOG}}$	$\mathsf{aECDSA}_{\mathrm{DLOG}}$	$\mathsf{aECDSA}_{\pi\mathrm{DLOG}}$
Scalar mult. in $\mathbb{G}$	2	6	8
Group op. in $\mathbb{G}$	2	3	4
$\cdot_{\mathbb{F}_q}$	-	2	2
$(\cdot)^{-1}_{\mathbb{F}_q}$	-	1	1
$\mathsf{H}(\cdot)$	1	2	3
$f(\cdot)$	-	1	1

Table 4: Computational complexity of pVrfy for adaptor signatures.

Relation R	DLOG	πDLOG
Size of $(S, w) \in R$	$\lvert\mathbb{G}\rvert + \lvert\mathbb{F}_q\rvert$	$\lvert\mathbb{G}\rvert + 3\lvert\mathbb{F}_q\rvert$

Table 5: Size of instances of relations R.

5.1 Efficiency of Adaptor Signatures for Practical Signature Schemes

We compare the efficiency of the pre-signing and verification algorithms of the three adaptor signatures described in the preceding sections, namely $\mathsf{aSchnorr}_{\mathrm{DLOG}}$, $\mathsf{aECDSA}_{\mathrm{DLOG}}$, and $\mathsf{aECDSA}_{\pi\mathrm{DLOG}}$, instantiated over a cyclic elliptic-curve group $\mathbb{G}$ of prime order q in additive notation. We instantiate the NIZK proof of knowledge of a discrete logarithm needed in $\mathsf{aECDSA}_{\pi\mathrm{DLOG}}$ with the standard Schnorr proof of knowledge. Aumayr et al. [AEE+21] required an online-extractable NIZK proof of knowledge, which can be obtained by repeating the proof multiple times [Fis05]. To illustrate the efficiency, we consider a single repetition, yielding a lower bound on the actual computational cost of $\mathsf{aECDSA}_{\pi\mathrm{DLOG}}$. We also note that the hash function H may differ among the adaptor signatures, the NIZK proof of knowledge of a discrete logarithm, and the NIZK proof of equality of discrete logarithms.

In Table 3, we compare the efficiency of the pSign algorithms of the three adaptor signatures and the sizes of the resulting pre-signatures. By $|\mathbb{G}|$ and $|\mathbb{F}_q|$, we denote the size of the representation of elements of $\mathbb{G}$ and $\mathbb{F}_q$, respectively. In Table 4, we compare the efficiency of the respective verification algorithms pVrfy. Table 5 summarizes the sizes of statement–witness pairs $(S, w) \in R$ for the DLOG and πDLOG relations.

In terms of time and size complexity, $\mathsf{aSchnorr}_{\mathrm{DLOG}}$ is the most efficient among the schemes considered. However, in the context of Bitcoin, ECDSA has historically been the supported signature scheme, with Schnorr signatures only introduced in 2021 via BIP340 [WNR]. As a result, ECDSA remains widely used in practice. For such settings, $\mathsf{aECDSA}_{\mathrm{DLOG}}$ is preferable to $\mathsf{aECDSA}_{\pi_{\mathrm{DLOG}}}$ due to its greater efficiency. Additionally, as of this writing, there is only a single known implementation of Schnorr-based adaptor signatures.

Compared with the corresponding ECDSA schemes, adaptor signatures for ECDSA$^+$ have the same verification cost and require only one additional subtraction in $\mathbb{F}_q$ during signing.

Adaptor signatures on Bitcoin. Both ECDSA and Schnorr signatures on the Bitcoin network are implemented over the elliptic curve secp256k1, as specified in Table 1. For this group $\mathbb{G}$, secret keys are sampled from $\mathbb{F}_q$ and are 256-bit integers, meaning $|\mathbb{F}_q| = 32$ bytes for both schemes. In ECDSA, the public key pk is typically represented in a 33-byte compressed form or a 65-byte uncompressed form. In contrast, Schnorr public keys have a fixed length of 32 bytes, matching the size of the secret key.

6 Adaptor-Based Discreet Log Contracts

While the motivating application of adaptor signatures in [AEE+21] focused on generalized payment channels, it was later recognized that their properties are naturally suited to realizing conditional payments. One such protocol was proposed by Madathil et al. [MTV+23]. However, despite possessing a security proof, the construction is complex and, to the best of our knowledge, has not seen practical adoption. A more practical and widely deployed solution is known as *Discreet Log Contracts* (DLCs) [GKK22], which use adaptor signatures directly. Although DLCs were originally proposed by Dryja [Dry17], we follow the formalization by Le Guilly et al. [GKK22], which is based on ECDSA adaptor signatures.

At a high level, DLCs enable two parties to agree on a set of conditional payments, with the outcome determined by an external oracle. Crucially, the protocol runs without relying on a trusted intermediary to hold funds. Consider a scenario where Alice and Bob wager 1 BTC on a football match: if Team A wins, Alice receives the funds; otherwise, they go to Bob. They prepare two transactions—one paying Alice and one paying Bob—each tied to a specific outcome. These transactions are called Contract Execution Transactions (CETs). These CETs are encapsulated as promises and bound to the respective outcomes via anticipation points, ensuring that they can be publicly verified but not yet executed. Once the match concludes, the oracle publishes an attestation corresponding to the actual outcome. This attestation allows the winning party to adapt the relevant promise into a valid signature and broadcast the redeeming transaction. The design guarantees that only the transaction corresponding to the oracle's attested outcome can be finalized.

In this section, we provide a formal definition of Discreet Log Contracts (Section 6.1) and recall their realization using adaptor signatures (Section 6.2). This formalization enables us to articulate precise security notions for DLCs, and we prove that these notions are satisfied by existing constructions based on ECDSA adaptor signatures (Section 6.3).

6.1 Defining DLCs

Below, we give our formal definition of a DLC scheme. The separation of the components among the parties is illustrated in Figure 31.

Definition 18. *A* Discreet Log Contract (DLC) *scheme for a digital signature scheme* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is a tuple of algorithms* $\mathsf{DLC} = (\mathsf{OracleKGen}, \mathsf{EventKGen}, \mathsf{Promise}, \mathsf{PromiseVrfy}, \mathsf{Attest}, \mathsf{AttestVrfy}, \mathsf{Redeem})$ *such that*

- $\mathsf{OracleKGen}(1^\lambda)$ *is a* PPT *algorithm that takes as input a security parameter* 1^λ *and outputs an oracle key pair* $(\mathsf{osk}, \mathsf{opk})$ *consisting of an oracle secret key* osk *and an oracle public key* opk.
- $\mathsf{EventKGen}(1^\lambda)$ *is a* PPT *algorithm that takes as input a security parameter* 1^λ *and outputs an event key pair* $(\mathsf{esk}, \mathsf{epk})$ *consisting of an event secret key* esk *and an event public key* epk.

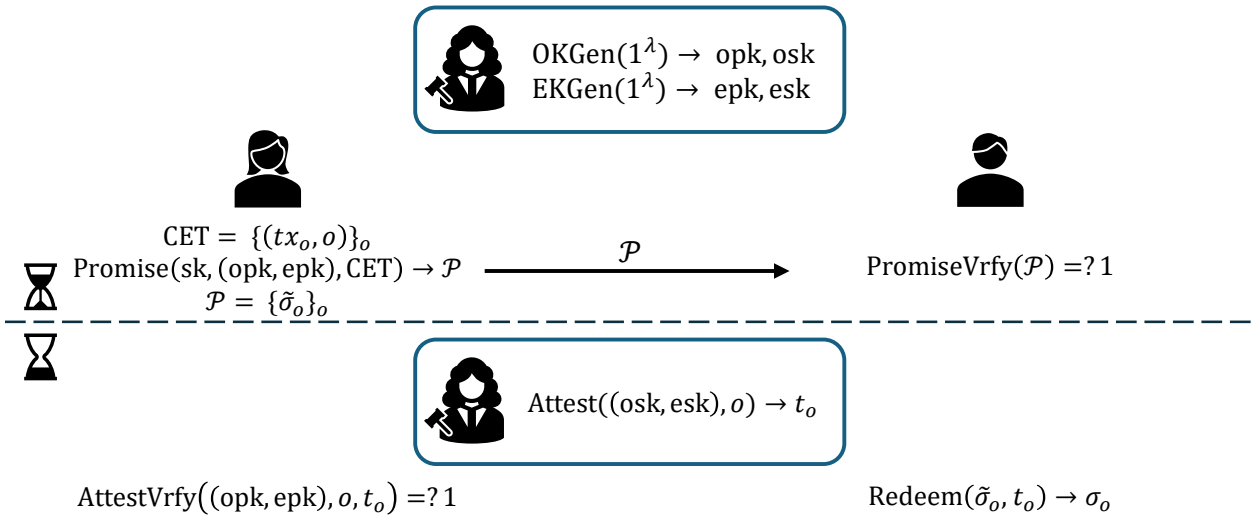

Fig. 31: DLC execution between an oracle (box), sender (left) and receiver (right).

- $\mathsf{Promise}(\mathsf{sk}, (\mathsf{opk}, \mathsf{epk}), \mathsf{CET})$ *is a* PPT *algorithm that takes as input a signing key* sk*, a pair of oracle and event public keys* $(\mathsf{opk}, \mathsf{epk})$*, and a set of contract execution transactions* $\mathsf{CET} = \{(tx_o, o)\}_o$*, and outputs a tuple* $\mathcal{P} = (\tilde{\sigma}_o)_o$ *of promises, where each* $\tilde{\sigma}_o$ *is tied to a transaction* tx_o *and its corresponding outcome* o*.*
- $\mathsf{PromiseVrfy}(\mathsf{pk}, (\mathsf{opk}, \mathsf{epk}), \tilde{\sigma}_o, tx_o, o)$ *is a* DPT *algorithm that takes as input a public key* pk*, a pair of oracle and event public keys* $(\mathsf{opk}, \mathsf{epk})$*, a promise* $\tilde{\sigma}_o$*, a transaction* tx_o*, and an outcome* o*, and outputs* 1 *if the promise is valid for the transaction and outcome and otherwise outputs* 0*.*
- $\mathsf{Attest}((\mathsf{osk}, \mathsf{esk}), o)$ *is a* DPT *algorithm that takes as input a pair of oracle and event secret keys* $(\mathsf{osk}, \mathsf{esk})$ *and an outcome* o*, and outputs an attestation* t_o *corresponding to* o*.*
- $\mathsf{AttestVrfy}((\mathsf{opk}, \mathsf{epk}), o, t)$ *is a* DPT *algorithm that takes as input a pair of oracle and event public keys* $(\mathsf{opk}, \mathsf{epk})$*, an outcome* o*, and an attestation* t*, and outputs* 1 *if the attestation corresponds to the outcome and otherwise outputs* 0*.*
- $\mathsf{Redeem}(\mathsf{pk}, \tilde{\sigma}_o, t_o)$ *is a* DPT *algorithm that takes as input a public key* pk*, a pre-signature* $\tilde{\sigma}_o$ *on a transaction* tx_o *corresponding to an outcome* o*, and an attestation* t_o*, and returns a signature* σ_{tx_o} *on* tx_o*.*

For every outcome o *and transaction* tx*, a* DLC *scheme must satisfy* promise correctness*, i.e.,*

$$\Pr\left[\mathsf{PromiseVrfy}(\mathsf{pk}, (\mathsf{opk}, \mathsf{epk}), \tilde{\sigma}, tx, o) = 1 \;\middle|\; \begin{array}{l} (\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OracleKGen}(1^\lambda), \\ (\mathsf{esk}, \mathsf{epk}) \leftarrow \mathsf{EventKGen}(1^\lambda), \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \tilde{\sigma} \leftarrow \mathsf{Promise}(\mathsf{sk}, (\mathsf{opk}, \mathsf{epk}), \{(tx, o)\}) \end{array}\right] = 1,$$

and attest correctness*, i.e.,*

$$\Pr\left[\mathsf{AttestVrfy}((\mathsf{opk}, \mathsf{epk}), o, t_o) = 1 \;\middle|\; \begin{array}{l} (\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OracleKGen}(1^\lambda), \\ (\mathsf{esk}, \mathsf{epk}) \leftarrow \mathsf{EventKGen}(1^\lambda), \\ t_o := \mathsf{Attest}((\mathsf{osk}, \mathsf{esk}), o) \end{array}\right] = 1.$$

$\mathsf{OracleKGen}(1^\lambda)$

$(P, p) \leftarrow \mathsf{DLOG.GenR}(1^\lambda)$
return (p, P)

$\mathsf{EventKGen}(1^\lambda)$

$(R, r) \leftarrow \mathsf{DLOG.GenR}(1^\lambda)$
return (r, R)

$\mathsf{Promise}(\mathsf{sk}, (P, R), \mathsf{CET})$

parse $\mathsf{CET} := \{(tx_o, o)\}_o$
$\mathcal{Q} = \emptyset$
for $(tx_o, o) \in \mathsf{CET}$:
  $T_o = R + P \cdot \mathsf{H}(P \parallel R \parallel o)$
  $\tilde{\sigma}_o \leftarrow \mathsf{pSign}(\mathsf{sk}, tx_o, T_o)$
  $\mathcal{Q} = \mathcal{Q} \cup \{\tilde{\sigma}_o\}$
return $\mathcal{Q}$

$\mathsf{PromiseVrfy}(\mathsf{pk}, (P, R), \tilde{\sigma}_o, tx_o, o)$

$T_o = R + P \cdot \mathsf{H}(P \parallel R \parallel o)$
return $\mathsf{pVrfy}(\mathsf{pk}, tx_o, T_o, \tilde{\sigma}_o)$

$\mathsf{Attest}((p, r), o)$

$t_o = r + p \cdot \mathsf{H}(P \parallel R \parallel o)$
return t_o

$\mathsf{AttestVrfy}((P, R), o, t)$

$T_o = R + P \cdot \mathsf{H}(P \parallel R \parallel o)$
$T = tG$
return $(T_o = T)$

$\mathsf{Redeem}(\mathsf{pk}, \tilde{\sigma}_o, t_o)$

$\sigma \leftarrow \mathsf{Adapt}(\mathsf{pk}, \tilde{\sigma}_o, t_o)$
return σ

Fig. 32: Construction of DLCs using DLOG-based adaptor signatures.

6.2 DLCs from Adaptor Signatures

DLCs can be naturally realized using adaptor signatures. The core idea is to link transactions to real-world events via pre-signatures. The sender intends to create a set of promises, each corresponding to a distinct potential outcome o. To achieve this, adaptor signatures are used to pre-sign transactions tx_o associated with each outcome o. In the standard instantiation, the oracle publishes a random event key $R := rG$ prior to the event. For every outcome o, the sender computes an *anticipation point* $T_o \in \mathbb{G}$, defined as

$$T_o := R + P \cdot \mathsf{H}(P \parallel R \parallel o),$$

where H is a hash function and $P := pG$ is the oracle's public key (with private key p). These anticipation points T_o cryptographically bind each outcome to the specific transaction intended for execution upon that outcome. This connection is realized by creating promises, specifically by pre-signing each tx_o with respect to the statement T_o. These promises are computed by the sender and transmitted to the receiver for verification. Intuitively, the adaptor signatures guarantee that the pre-signatures can be completed into valid signatures given knowledge of the scalar t_o such that $T_o = t_o G$. When the event concludes with the actual outcome o^*, the oracle attests to this result by publishing the discrete logarithm t_{o^*} of the anticipation point T_{o^*}. Using the event secret r and the long-term secret p, the oracle computes t_{o^*} as

$$t_{o^*} := r + p \cdot \mathsf{H}(P \parallel R \parallel o^*).$$

Note that the overhead for the oracle is independent of the number of possible outcomes, as it does not need to know all the anticipation points in advance. Possession of this attestation t_{o^*} empowers the receiver to redeem the corresponding transaction tx_{o^*} by adapting the pre-signature into a standard signature. Crucially, this finalization requires no further interaction with the sender. Consequently, the receiver can settle the contract on-chain even if the sender becomes unresponsive following an unfavorable outcome. The formal construction of a DLC scheme based on a DLOG-based adaptor signature scheme is presented in Figure 32.

Base anticipation points. In the naive DLC formulation, the sender and receiver must compute an anticipation point for every conceivable outcome. However, this approach scales poorly when the outcome space is large. Consider, for instance, an event with numerical outcomes, such as the exchange rate of

a currency. If the outcomes are represented as n-bit strings, the parties must compute 2^n anticipation points, requiring 2^n scalar multiplications in $\mathbb{G}$. To mitigate this computational complexity, Le Guilly et al. [GKK22] proposed an optimization using base anticipation points. Here, we describe a simplified variant of this technique.

Instead of deriving a distinct anticipation point T_o for every outcome, the sender computes n *base anticipation points* B_i (for $1 \leq i \leq n$) as

$$B_i := R + P \cdot \mathsf{H}(P \parallel R \parallel i),$$

using the hash function H and the oracle public key P. Subsequently, the sender constructs the anticipation point T_o for a specific outcome o with binary representation $o = b_1 \dots b_n \in \{0,1\}^n$ by summing the relevant base points.

$$T_o := \sum_{\substack{1 \leq i \leq n \\ b_i = 1}} B_i.$$

To attest to an outcome $o^* = b_1^* \dots b_n^*$, the oracle reveals the discrete logarithm t_{o^*} of the aggregate point T_{o^*}. This scalar is the sum of the discrete logarithms of the active base anticipation points.

$$t_{o^*} := \sum_{\substack{1 \leq i \leq n \\ b_i^* = 1}} \left(r + p \cdot \mathsf{H}(P \parallel R \parallel i)\right).$$

This method reduces the number of scalar multiplications from exponential (2^n) to linear (n), at the cost of additional group operations.

6.3 Security of Discreet Log Contracts

Having formalized the structure of DLCs in Definition 18, we now define the security properties that a DLC scheme must satisfy. These properties guarantee that the protocol behaves correctly, even in the presence of adversarial participants. We introduce two primary security definitions for DLCs: *unforgeability*, which protects senders, and *redeemability*, which protects receivers.

We stress that our proofs rely only on extractability and pre-signature adaptability, rather than on their robust variants, showing that our relaxed definitions provide meaningful security guarantees.

Unforgeability. Unforgeability protects the sender by ensuring that no adversary can produce a valid signature on a transaction without possession of the sender's secret key. In the context of DLCs, this definition extends standard unforgeability by granting the adversary additional power: it may observe promises created for various transactions corresponding to different outcomes, as well as a single attestation released by the oracle. Crucially, access to the attestation oracle is restricted to a single invocation per event key R, including repeated queries for the same outcome. Permitting multiple queries for the same event key would allow the adversary to recover the event secret key esk due to the linearity of the Schnorr-like construction. Therefore, the security model mandates that the event key must be refreshed for each event, ensuring the adversary never witnesses multiple attestations under the same R.

Definition 19. *A DLC scheme for a signature scheme Σ w.r.t. a hard relation R is* unforgeable *if, for every* PPT *adversary $\mathcal{A}$,*

$$\Pr[\mathsf{Unforgeability}_{\mathcal{A},\mathsf{DLC}}(\lambda) = 1] \in \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Unforgeability}_{\mathcal{A},\mathsf{DLC}}$ *is defined in Figure 33.*

To establish unforgeability of the adaptor-based DLC scheme presented in Figure 32, we rely on the One-More Discrete Logarithm assumption. Let $\mathcal{O}_{\mathsf{DLOG}}$ denote an oracle that takes as input $Y \in \mathbb{G}$ and returns its discrete logarithm, namely a value $y \in \mathbb{F}_q$ such that $Y = yG$. The One-More Discrete Logarithm problem (OMDL) captures the computational hardness of solving for the discrete logarithms of $n+1$ challenge group elements, given at most n queries to the $\mathcal{O}_{\mathsf{DLOG}}$ oracle [BNPS03].

$\mathsf{Unforgeability}_{\mathcal{A},\mathsf{DLC}}(\lambda)$

$\mathcal{Q}_1 := \emptyset, \mathcal{Q}_2 := \emptyset, \mathcal{Q}_3 := \emptyset, b := 0$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(p, P) \leftarrow \mathsf{OracleKGen}(1^\lambda)$
$(r, R) \leftarrow \mathsf{EventKGen}(1^\lambda)$
(tx^*, σ^*)
$\quad \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Promise}}, \mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{Attest}}}(\mathsf{pk}, (P, R))$
assert $|\mathcal{Q}_3| \leq 1$
if $|\mathcal{Q}_3| = 1$ **then**
$\quad$ **parse** $\mathcal{Q}_3 := \{o^*\}$
$\quad b := ((\cdot, o^*, tx^*) \notin \mathcal{Q}_1)$
else
$\quad b := 1$
return
$\quad \mathsf{Vrfy}(\mathsf{pk}, tx^*, \sigma^*) \wedge (tx^* \notin \mathcal{Q}_2) \wedge b$

$\mathcal{O}_{\mathsf{Promise}}(\mathsf{CET})$

parse $\mathsf{CET} := \{(tx_o, o)\}_o$
$\mathcal{P} \leftarrow \mathsf{Promise}(\mathsf{sk}, (P, R), \mathsf{CET})$
for $\tilde{\sigma}_o \in \mathcal{P}$:
$\quad \mathcal{Q}_1 := \mathcal{Q}_1 \cup \{(\tilde{\sigma}_o, o, tx_o)\}$
return $\mathcal{P}$

$\mathcal{O}_{\mathsf{Sign}}(tx)$

$\mathcal{Q}_2 := \mathcal{Q}_2 \cup \{tx\}$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, tx)$
return σ

$\mathcal{O}_{\mathsf{Attest}}(o)$

assert $\mathcal{Q}_3 = \emptyset$
$\mathcal{Q}_3 := \{o\}$
$t_o \leftarrow \mathsf{Attest}((p, r), o)$
return t_o

Fig. 33: $\mathsf{Unforgeability}_{\mathcal{A},\mathsf{DLC}}(\lambda)$ experiment.

$\mathsf{OMDL}^{\mathcal{A}}_{\mathsf{PGen},n}(\lambda)$

$(\mathbb{G}, q, G) \leftarrow \mathsf{PGen}(1^\lambda)$
$c := 0$
for $i = 1, \ldots, n+1$:
$\quad x_i \leftarrow \mathbb{F}_q^*; \quad X_i := x_i G$
$(x'_1, \ldots, x'_{n+1}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{DLOG}}}(\mathbb{G}, q, G, X_1, \ldots, X_{n+1})$
return $(x_i = x'_i$ for all $i \in [n+1]) \wedge (c \leq n)$

$\mathcal{O}_{\mathsf{DLOG}}(Y)$

$c := c + 1$
return $\mathrm{dlog}_G(Y)$

Fig. 34: The $\mathsf{OMDL}[n]$ problem.

Definition 20. *Let λ be a security parameter and $n \in \mathbb{N}$. The $\mathsf{OMDL}[n]$ problem is hard if, for all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\mathsf{OMDL}^{\mathcal{A}}_{\mathsf{PGen},n}(\lambda) = 1\right] \in \mathsf{negl}(\lambda),$$

where the $\mathsf{OMDL}^{\mathcal{A}}_{\mathsf{PGen},n}$ experiment is introduced in Figure 34.

Sampling the challenge exponents from $\mathbb{F}_q^*$ matches the oracle and event key-generation algorithms. For $\mathsf{OMDL}[1]$, this differs from sampling them from $\mathbb{F}_q$ only when at least one of the two challenge elements is the identity, an event of probability at most $2/q$.

Under the $\mathsf{OMDL}[1]$ assumption and collision resistance of the coefficient hash function, we prove that extractable adaptor signatures yield unforgeable DLCs.

Theorem 10. *Let $\mathsf{a}\Sigma_{DLOG}$ be an adaptor signature scheme for a signature scheme Σ defined w.r.t. the DLOG relation. If the $\mathsf{OMDL}[1]$ problem is hard, the coefficient hash function H used in Figure 32 is collision resistant, and $\mathsf{a}\Sigma_{DLOG}$ satisfies extractability, then the $\mathsf{a}\Sigma_{DLOG}$-based DLC scheme is unforgeable.*

$\mathcal{G}_1^{\mathcal{A}}(\lambda)$

$\mathcal{Q}_1 := \emptyset, \mathcal{Q}_2 := \emptyset, \mathcal{Q}_3 := \emptyset$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(p, P) \leftarrow \mathsf{OracleKGen}(1^\lambda)$
$(r, R) \leftarrow \mathsf{EventKGen}(1^\lambda)$
$(tx^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Promise}}, \mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{Attest}}}(\mathsf{pk}, (P, R))$
assert $|\mathcal{Q}_3| \leq 1$
if $|\mathcal{Q}_3| = 1$ **then**
  parse $\mathcal{Q}_3 := \{o^*\}$
  $b := ((\cdot, o^*, tx^*) \notin \mathcal{Q}_1)$
else
  $b := 1$
for $(\tilde{\sigma}_o, o) \in \mathcal{Q}_1[tx^*]$:
  $T_o := R + P \cdot \mathsf{H}(P \parallel R \parallel o)$
  $y := \mathsf{Ext}(\mathsf{pk}, \sigma^*, \tilde{\sigma}_o, T_o)$
  if $(T_o, y) \in \mathsf{DLOG}$ **then**
    abort
return $\mathsf{Vrfy}(\mathsf{pk}, tx^*, \sigma^*) \wedge (tx^* \notin \mathcal{Q}_2) \wedge b$

$\mathcal{O}_{\mathsf{Promise}}(\mathsf{CET})$

parse $\mathsf{CET} := \{(tx_o, o)\}_o$
$\mathcal{P} := \emptyset$
for $(tx_o, o) \in \mathsf{CET}$:
  $T_o := R + P \cdot \mathsf{H}(P \parallel R \parallel o)$
  $\tilde{\sigma}_o \leftarrow \mathsf{pSign}(\mathsf{sk}, tx_o, T_o)$
  $\mathcal{P} := \mathcal{P} \cup \{\tilde{\sigma}_o\}$
  $\mathcal{Q}_1 := \mathcal{Q}_1 \cup \{(\tilde{\sigma}_o, o, tx_o)\}$
return $\mathcal{P}$

$\mathcal{O}_{\mathsf{Sign}}(tx)$

$\mathcal{Q}_2 := \mathcal{Q}_2 \cup \{tx\}$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, tx)$
return σ

$\mathcal{O}_{\mathsf{Attest}}(o)$

assert $\mathcal{Q}_3 = \emptyset$
$\mathcal{Q}_3 := \{o\}$
$t_o := r + p \cdot \mathsf{H}(P \parallel R \parallel o)$
return t_o

Fig. 35: Game $\mathcal{G}_1^{\mathcal{A}}(\lambda)$.

Proof. To prove the statement, we reduce the unforgeability of $\mathsf{a}\Sigma_{\text{DLOG}}$-DLC to the extractability of $\mathsf{a}\Sigma_{\text{DLOG}}$. For a PPT adversary $\mathcal{A}$, we denote the initial experiment $\mathsf{Unforgeability}_{\mathcal{A},\mathsf{a}\Sigma_{\text{DLOG}}\text{-DLC}}$ in Figure 33 by $\mathcal{G}_0$ and define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_0}(\lambda) := \Pr\left[\mathcal{G}_0^{\mathcal{A}}(\lambda) = 1\right] = \Pr\left[\mathsf{Unforgeability}_{\mathcal{A},\mathsf{a}\Sigma_{\text{DLOG}}\text{-DLC}}(\lambda) = 1\right].$$

We modify $\mathcal{G}_0$ into a game $\mathcal{G}_1$ and show that $\mathcal{A}$ wins $\mathcal{G}_1$ with probability at least $\mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_0}(\lambda) - \mathsf{negl}(\lambda)$. Using $\mathcal{G}_1$, we construct a PPT forger $\mathcal{F}$ that uses $\mathcal{A}$ to win $\mathsf{Ext}_{\mathcal{F},\mathsf{a}\Sigma_{\text{DLOG}}}(\lambda)$.

First, we modify $\mathcal{G}_0$ into a game $\mathcal{G}_1$, which we formally define in Figure 35. The game $\mathcal{G}_1$ is similar to $\mathcal{G}_0$ with the following difference. Recall that in the unforgeability experiment, $\mathcal{A}$ outputs (tx^*, σ^*), where σ^* is a signature on a transaction tx^*. The game now ensures that the signature σ^* cannot be used to extract a valid witness for any of the previously used anticipation points. Specifically, for every query to $\mathcal{O}_{\mathsf{Promise}}$ containing tx^*, the game records the corresponding outcome o and the returned pre-signature $\tilde{\sigma}_o$. By the construction of $\mathsf{a}\Sigma_{\text{DLOG}}$-DLC, $\tilde{\sigma}_o$ is a pre-signature on tx^* w.r.t. the anticipation point T_o corresponding to o. The game then checks whether σ^* and $\tilde{\sigma}_o$ extract to a valid witness for T_o. In that case, the game aborts.

For now, assume that

$$\mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda) \geq \mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_0}(\lambda) - \mathsf{negl}(\lambda),$$

and continue with the main part of the proof.

In Figure 36, we define a forger $\mathcal{F}$ that uses an adversary winning $\mathcal{G}_1$ to win the $\mathsf{Ext}_{\mathcal{F},\mathsf{a}\Sigma_{\text{DLOG}}}(\lambda)$ experiment from Figure 27. We write $\mathcal{O}'$ for the extractability experiment's oracles and denote the query sets of $\mathcal{O}'_{\mathsf{Sign}}$ and $\mathcal{O}'_{\mathsf{pSign}}$ by $\mathcal{Q}'_1$ and $\mathcal{Q}'_2$, respectively. The forger $\mathcal{F}$ samples (p, P) and (r, R) as in the key-generation algorithms and runs $\mathcal{A}$ on input $(\mathsf{pk}, (P, R))$, providing the oracles $\mathcal{O}_{\mathsf{Promise}}$, $\mathcal{O}_{\mathsf{Sign}}$, and $\mathcal{O}_{\mathsf{Attest}}$ from $\mathcal{G}_1$. It simulates Sign and pSign using $\mathcal{O}'_{\mathsf{Sign}}$ and $\mathcal{O}'_{\mathsf{pSign}}$, respectively. The adversary eventually outputs (tx^*, σ^*), which the forger outputs in the extractability experiment.

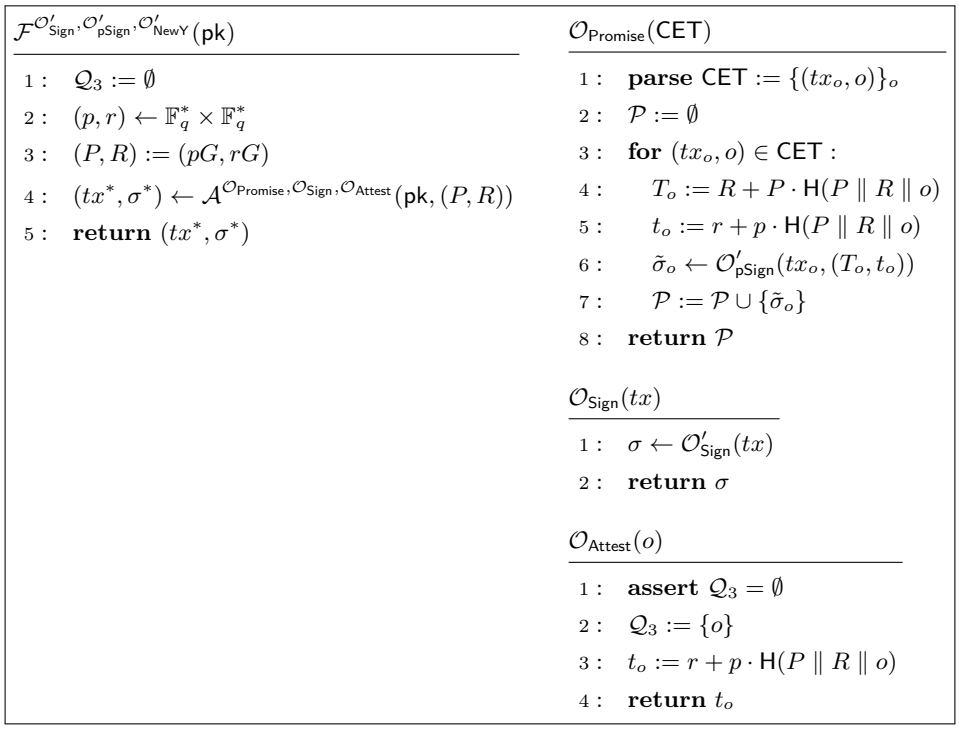

Fig. 36: Forger $\mathcal{F}$.

We now show that the forger wins the extractability experiment in Figure 27 with probability at least $\mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)$. If $\mathcal{A}$ wins $\mathcal{G}_1$, then $\mathsf{Vrfy}(\mathsf{pk}, tx^*, \sigma^*) = 1$. Moreover, $tx^* \notin \mathcal{Q}'_1$ because this condition could fail only if $\mathcal{A}$ had queried $\mathcal{O}_{\mathsf{Sign}}$ on tx^*. Finally, the additional condition in $\mathcal{G}_1$ ensures that $(T_o, \mathsf{Ext}(\mathsf{pk}, \sigma^*, \tilde{\sigma}_o, T_o)) \notin \mathsf{DLOG}$ for every $(T_o, \tilde{\sigma}_o) \in \mathcal{Q}'_2[tx^*]$.

As a result, $\mathsf{Adv}^{\mathcal{F}}_{\mathsf{Ext}}(\lambda) \geq \mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda) \geq \mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_0}(\lambda) - \mathsf{negl}(\lambda)$. Thus, if $\mathsf{a}\Sigma_{\mathrm{DLOG}}$ is extractable, then $\mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_0}(\lambda)$ is negligible.

It remains to show that $\mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda) \geq \mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_0}(\lambda) - \mathsf{negl}(\lambda)$. For contradiction, assume that $\mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_0}(\lambda) - \mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)$ is non-negligible. As the games differ only in the gray part, this means that the probability that $\mathcal{G}_1$ aborts in that part is non-negligible. We show that if the probability that $\mathcal{G}_1$ aborts is non-negligible, then we can break the OMDL assumption introduced in Figure 34. We construct a solver $\mathcal{B}$ in Figure 37.

The solver receives challenge elements X_1 and X_2 in the $\mathsf{OMDL}[1]$ problem and simulates the adversary from $\mathcal{G}_1$ with $P = X_1$ and $R = X_2$. To simulate the oracles from $\mathcal{G}_1$, it suffices to know the secret key sk and the pair (P, R), except when answering $\mathcal{O}_{\mathsf{Attest}}$. Computing an attestation in the usual way would require the discrete logarithms p and r of X_1 and X_2. The $\mathsf{OMDL}[1]$ problem allows the solver one query to $\mathcal{O}_{\mathsf{DLOG}}$, which it uses to answer the adversary's single allowed query to $\mathcal{O}_{\mathsf{Attest}}$.

When $\mathcal{A}$ causes $\mathcal{G}_1$ to abort in the added check, there is a pair $(\tilde{\sigma}_o, o) \in \mathcal{Q}_1[tx^*]$ such that $(T_o, y) = (T_o, \mathsf{Ext}(\mathsf{pk}, \sigma^*, \tilde{\sigma}_o, T_o)) \in \mathsf{DLOG}$. We find this pair by iterating over $\mathcal{Q}_1[tx^*]$. Suppose that $\mathcal{A}$ queried $\mathcal{O}_{\mathsf{Attest}}$ on exactly one outcome o^*. We then have

$$T_o = R + P \cdot \mathsf{H}(P \parallel R \parallel o)$$
$$T_{o^*} = R + P \cdot \mathsf{H}(P \parallel R \parallel o^*).$$

This system of two linear equations lets us solve for both p and r. Indeed, whenever $\mathcal{G}_0$ wins, its condition b ensures that $o \neq o^*$. Let $\mathsf{Bad}_{\mathsf{H}}$ be the event that $\mathsf{H}(P \parallel R \parallel o) = \mathsf{H}(P \parallel R \parallel o^*)$ for such distinct outcomes. If the experiment evaluates H on at most Q_{H} distinct inputs, then the birthday bound gives

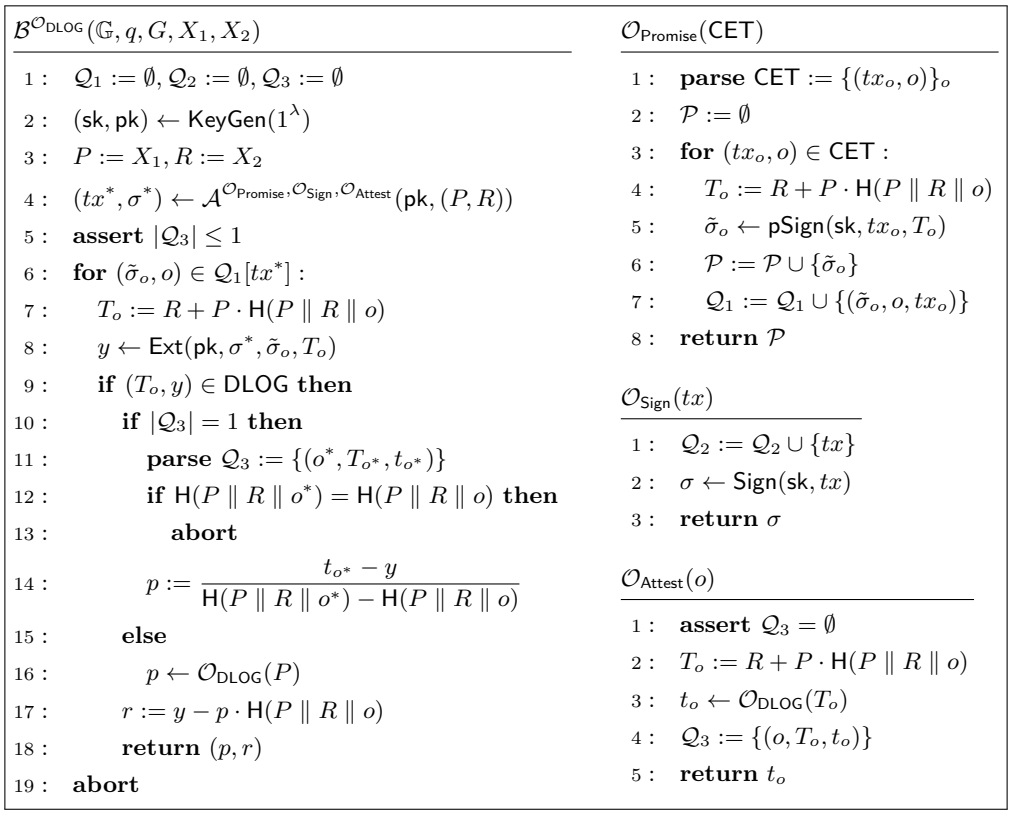

Fig. 37: Solver $\mathcal{B}$ for the $\mathsf{OMDL}[1]$ problem.

$\Pr[\mathsf{Bad}_\mathsf{H}] \leq Q_\mathsf{H}(Q_\mathsf{H}-1)/(2q)$ in the random oracle model. In the standard model, a non-negligible probability of Bad_H contradicts collision resistance of H. Outside Bad_H, the denominator used by $\mathcal{B}$ is nonzero.

If $\mathcal{A}$ does not query $\mathcal{O}_{\mathsf{Attest}}$, then we instead query $\mathcal{O}_{\mathsf{DLOG}}$ on P to obtain p such that $P = pG$ and solve for r in the same way.

Whenever $\mathcal{G}_0$ wins but $\mathcal{G}_1$ aborts and Bad_H does not occur, $\mathcal{B}$ solves the $\mathsf{OMDL}[1]$ problem, and hence

$$\mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_0}(\lambda) - \mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda) \leq \Pr[\mathsf{OMDL}^{\mathcal{B}}_{\mathsf{PGen},1}(\lambda) = 1] + \Pr[\mathsf{Bad}_\mathsf{H}].$$

The hardness of $\mathsf{OMDL}[1]$ and collision resistance of H make the right-hand side negligible, so

$$\mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda) \geq \mathsf{Adv}^{\mathcal{A}}_{\mathcal{G}_0}(\lambda) - \mathsf{negl}(\lambda),$$

which completes the proof. □

Combining Theorems 7 and 10 yields the following concrete result for DLCs based on $\mathsf{aECDSA}^+_{\mathrm{DLOG}}$.

Corollary 2. *If the ECDSA$^+$ signature scheme satisfies strong unforgeability, the* DLOG *relation is hard, and the* $\mathsf{OMDL}[1]$ *problem is hard, then the DLC scheme in Figure 32 instantiated with* aECDSA^+_{DLOG} *is unforgeable in the random oracle model. Under the same hardness assumptions, if* $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$ *is a perfectly zero-knowledge NIZK proof system for equality of discrete logarithms and the coefficient hash function* H *is collision resistant, then the scheme is unforgeable in the standard model.*

DLCs instantiated with Schnorr adaptor signatures. Gerhart et al. [GSST24] show that Schnorr-based adaptor signatures satisfy robust extractability if the Schnorr signature scheme is strongly unforgeable. Thus,

$\mathsf{Redeemability}_{\mathcal{A},\mathsf{DLC}}(\lambda)$

$(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OracleKGen}(1^\lambda)$
$(\mathsf{esk}, \mathsf{epk}) \leftarrow \mathsf{EventKGen}(1^\lambda)$
$(\mathcal{P}, \mathsf{pk}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{opk}, \mathsf{epk})$
parse $\mathcal{P} := \{(\tilde{\sigma}_o, tx_o, o)\}_o$
for $(\tilde{\sigma}_o, tx_o, o) \in \mathcal{P}$:
  assert $\mathsf{PromiseVrfy}(\mathsf{pk}, (\mathsf{opk}, \mathsf{epk}), \tilde{\sigma}_o, tx_o, o)$
  $t_o := \mathsf{Attest}((\mathsf{osk}, \mathsf{esk}), o)$
  $\sigma := \mathsf{Redeem}(\mathsf{pk}, \tilde{\sigma}_o, t_o)$
  if $\mathsf{Vrfy}(\mathsf{pk}, tx_o, \sigma) = 0$ **then**
    return 1
return 0

Fig. 38: $\mathsf{Redeemability}_{\mathcal{A},\mathsf{DLC}}(\lambda)$ experiment.

by Theorem 10, the DLC scheme in Figure 32 instantiated with Schnorr adaptor signatures is unforgeable assuming strong unforgeability of Schnorr signatures, hardness of OMDL[1], and collision resistance of the coefficient hash function H.

Redeemability. Redeemability protects the receiver by guaranteeing that, if the sender issues a promise and the oracle publishes a correct attestation for an outcome, then the receiver can redeem the corresponding transaction. To formalize this notion, we let an adversary choose a public key and a set of promises, which correspond to pre-signatures in adaptor-based DLCs, on transactions for arbitrary outcomes. Each promise must verify with respect to the oracle and event public keys, and the adversary wins if any promise cannot be redeemed correctly.

Definition 21. *A DLC scheme for a signature scheme Σ w.r.t. a hard relation R is* redeemable *if, for every* PPT *adversary $\mathcal{A}$,*

$$\Pr[\mathsf{Redeemability}_{\mathcal{A},\mathsf{DLC}}(\lambda) = 1] \in \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Redeemability}_{\mathcal{A},\mathsf{DLC}}$ *is defined in Figure 38.*

We prove that pre-signature adaptability of adaptor signatures implies redeemability of adaptor-based DLCs.

Theorem 11. *Let $\mathsf{a}\Sigma_{DLOG}$ be an adaptor signature scheme for a signature scheme Σ defined w.r.t. the* DLOG *relation. If $\mathsf{a}\Sigma_{DLOG}$ satisfies pre-signature adaptability, then the $\mathsf{a}\Sigma_{DLOG}$-based DLC scheme in Figure 32 satisfies redeemability.*

Proof. Let $\mathcal{A}$ be an adversary that wins the $\mathsf{Redeemability}_{\mathcal{A},\mathsf{DLC}}(\lambda)$ experiment with non-negligible probability. We construct a forger $\mathcal{F}$ in Figure 39 that wins the $\mathsf{pSigAdapt}_{\mathcal{F},\mathsf{a}\Sigma_{\mathsf{DLOG}}}(\lambda)$ experiment of the underlying adaptor signature scheme with at least the same probability.

The forger $\mathcal{F}$ first generates the oracle and event keys and simulates $\mathcal{A}$ on input $(1^\lambda, P, R)$. The adversary $\mathcal{A}$ returns a public key pk and a list $\mathcal{P}$ of tuples $(\tilde{\sigma}_o, tx_o, o)$, where $\tilde{\sigma}_o$ is a pre-signature on tx_o for the outcome o. Using the previously generated keys, the forger computes T_o and t_o and checks which pre-signature fails to redeem correctly. For such an outcome o, it outputs $(tx_o, \mathsf{pk}, (T_o, t_o), \tilde{\sigma}_o)$.

By the construction of DLCs from adaptor signatures, $\mathsf{Redeem}(\mathsf{pk}, \tilde{\sigma}_o, t_o)$ is defined as $\mathsf{Adapt}(\mathsf{pk}, \tilde{\sigma}_o, t_o)$. For this outcome, we have

$$\mathsf{Vrfy}(\mathsf{pk}, tx_o, \mathsf{Adapt}(\mathsf{pk}, \tilde{\sigma}_o, t_o)) = 0,$$

$\mathcal{F}(1^\lambda)$

$(p, P) \leftarrow \mathsf{OracleKGen}(1^\lambda)$
$(r, R) \leftarrow \mathsf{EventKGen}(1^\lambda)$
$(\mathcal{P}, \mathsf{pk}) \leftarrow \mathcal{A}(1^\lambda, P, R)$
parse $\mathcal{P} := \{(\tilde{\sigma}_o, tx_o, o)\}_o$
for $(\tilde{\sigma}_o, tx_o, o) \in \mathcal{P}$:
  assert $\mathsf{PromiseVrfy}(\mathsf{pk}, (P, R), \tilde{\sigma}_o, tx_o, o)$
  $t_o := r + p \cdot \mathsf{H}(P \parallel R \parallel o)$
  $T_o := R + P \cdot \mathsf{H}(P \parallel R \parallel o)$
  $\sigma := \mathsf{Redeem}(\mathsf{pk}, \tilde{\sigma}_o, t_o)$
  if $\mathsf{Vrfy}(\mathsf{pk}, tx_o, \sigma) = 0$ **then**
    return $(tx_o, \mathsf{pk}, (T_o, t_o), \tilde{\sigma}_o)$
abort

Fig. 39: Forger $\mathcal{F}$ breaking pre-signature adaptability.

so the pre-signature cannot be adapted into a valid signature. However, the forger also verified that

$$\mathsf{PromiseVrfy}(\mathsf{pk}, (P, R), \tilde{\sigma}_o, tx_o, o) = 1,$$

which implies $\mathsf{pVrfy}(\mathsf{pk}, tx_o, T_o, \tilde{\sigma}_o) = 1$.

The final requirement $(T_o, t_o) \in \mathsf{DLOG}$ follows from attestation correctness in Definition 18, which gives $t_o G = T_o$. Thus, $\mathcal{F}$ wins the pre-signature adaptability experiment with at least the probability that $\mathcal{A}$ wins the redeemability experiment. Consequently, pre-signature adaptability of the underlying $\mathsf{a}\Sigma_{\mathrm{DLOG}}$ implies redeemability of the DLC scheme. □

The following corollary follows immediately from Theorems 5 and 11.

Corollary 3. *Let* aECDSA^+_{DLOG}*, described in Figure 6, be an adaptor signature scheme for ECDSA*$^+$ *w.r.t. the* DLOG *relation. Then the DLC protocol constructed using* aECDSA^+_{DLOG} *satisfies redeemability in the random oracle model. Additionally, if* $(\mathsf{P}_{\mathsf{DLEQ}}, \mathsf{V}_{\mathsf{DLEQ}})$ *is a NIZK proof system for equality of discrete logarithms, then the protocol satisfies redeemability in the standard model.*

Acknowledgments. Pavel Hubáček and Kristýna Mašková were partially supported by the Academy of Sciences of the Czech Republic (RVO 67985840), Czech Science Foundation GAČR grant No. 25-16311S, and by Zircuit. Kristýna Mašková was supported by the grant SVV–2026–260822. Berenika Richterová contributed to this work during her master's studies at Charles University in Prague. Berenika Richterová was supported by the European Research Council (ERC), grant number 101160608 "SYMPZON". Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

101. 10101.finance. https://10101.finance/. Accessed: 2025-05-15.

AEE+21. 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 Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 635–664, Singapore, December 6–10, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-92075-3_22.

Ato. atomic.finance. https://atomic.finance/. Accessed: 2025-05-15.

Bas. Basicswap dex. https://github.com/basicswap. Accessed: 2025-05-15.

BG04. Daniel R. L. Brown and Robert P. Gallant. The static Diffie-Hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004. URL: https://eprint.iacr.org/2004/306.

BGKS25. Ruben Baecker, Paul Gerhart, Jonathan Katz, and Dominique Schröder. Fair exchange for decentralized autonomous organizations via threshold adaptor signatures. Cryptology ePrint Archive, Report 2025/388, 2025. URL: https://eprint.iacr.org/2025/388.

BNPS03. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003. doi:10.1007/s00145-002-0120-1.

Cer. Certicom Research. Sec 2: Recommended elliptic curve domain parameters. URL: https://www.secg.org/sec2-v2.pdf.

CP93. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105, Santa Barbara, CA, USA, August 16–20, 1993. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-48071-4_7.

DOY22. Wei Dai, Tatsuaki Okamoto, and Go Yamamoto. Stronger security and generic constructions for adaptor signatures. In Takanori Isobe and Santanu Sarkar, editors, *INDOCRYPT 2022*, volume 13774 of *LNCS*, pages 52–77, Kolkata, India, December 11–14, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-22912-1_3.

Dry17. Thaddeus Dryja. Discreet log contracts, 2017. URL: https://adiabat.github.io/dlc.pdf.

DW15. Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In Andrzej Pelc and Alexander A. Schwarzmann, editors, *SSS 2015*, volume 9212 of *LNCS*, pages 3–18. Springer, 2015. doi:10.1007/978-3-319-21741-3_1.

EEE20. Muhammed F. Esgin, Oguzhan Ersoy, and Zekeriya Erkin. Post-quantum adaptor signatures and payment channel networks. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 378–397, Guildford, UK, September 14–18, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-59013-0_19.

EFH+21. Andreas Erwig, Sebastian Faust, Kristina Hostáková, Monosij Maitra, and Siavash Riahi. Two-party adaptor signatures from identification schemes. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 451–480, Virtual Event, May 10–13, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-75245-3_17.

Eig. Eigenwallet. https://github.com/eigenwallet/core. Accessed: 2025-05-15.

Far. Farcaster. https://github.com/farcaster-project. Accessed: 2025-05-15.

Fis05. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168, Santa Barbara, CA, USA, August 14–18, 2005. Springer Berlin Heidelberg, Germany. doi:10.1007/11535218_10.

Fou19. Lloyd Fournier. One-time verifiably encrypted signatures a.k.a. adaptor signatures, 2019. URL: https://github.com/LLFourn/one-time-VES/blob/master/main.pdf.

GKK22. Thibaut Le Guilly, Nadav Kohen, and Ichiro Kuwahara. Bitcoin oracle contracts: Discreet log contracts in practice. In *ICBC 2022*, pages 1–8. IEEE, 2022. doi:10.1109/ICBC54727.2022.9805512.

GMM+22. Noemi Glaeser, Matteo Maffei, Giulio Malavolta, Pedro Moreno-Sanchez, Erkan Tairi, and Sri Aravinda Krishnan Thyagarajan. Foundations of coin mixing services. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1259–1273, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. doi:10.1145/3548606.3560637.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-78967-3_24.

GSST24. Paul Gerhart, Dominique Schröder, Pratik Soni, and Sri Aravinda Krishnan Thyagarajan. Foundations of adaptor signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 161–189, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-58723-8_6.

Lav. Lava: Non-custodial loans. https://github.com/lava-xyz/loans-paper/blob/main/loans_v3.pdf. Accessed: 2025-05-15.

LTZ24. Xiangyu Liu, Ioannis Tzannetos, and Vassilis Zikas. Adaptor signatures: New security definition and a generic construction for NP relations. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 168–193, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore. doi:10.1007/978-981-96-0888-1_6.

MBB+19. Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and state channels: Payment networks that go faster than lightning. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 508–526, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-32101-7_30.

MMS+19. Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *NDSS 2019*, San Diego, CA, USA, February 24–27, 2019. The Internet Society. doi:10.14722/ndss.2019.23330.

MSK18. Pedro Moreno-Sanchez and Aniket Kate. Scriptless scripts with ECDSA, 2018. URL: https://web.archive.org/web/20231115202140/https://lists.linuxfoundation.org/pipermail/lightning-dev/attachments/20180426/fe978423/attachment-0001.pdf.

MTV+23. Varun Madathil, Sri Aravinda Krishnan Thyagarajan, Dimitrios Vasilopoulos, Lloyd Fournier, Giulio Malavolta, and Pedro Moreno-Sanchez. Cryptographic oracle-based conditional payments. In *NDSS 2023*, San Diego, CA, USA, February 27– March 3, 2023. The Internet Society.

Nat23. National Institute of Standards and Technology. Digital signature standard. Technical Report FIPS 186-5, U.S. Department of Commerce, Washington, D.C., 2023. doi:10.6028/NIST.FIPS.186-5.

P2P. P2P-derivatives. https://github.com/p2pderivatives/p2pderivatives-client. Accessed: 2025-05-15.

Poe. Andrew Poelstra. Scriptless scripts. URL: https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf.

Pol78. John M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978. doi:10.2307/2006496.

QPM+23. 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 *2023 IEEE Symposium on Security and Privacy*, pages 2462–2480, San Francisco, CA, USA, May 21–25, 2023. IEEE Computer Society Press. doi:10.1109/SP46215.2023.10179427.

Rus. rust-dlc. https://github.com/p2pderivatives/rust-dlc. Accessed: 2025-05-15.

Sch90. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, New York, USA. doi:10.1007/0-387-34805-0_22.

Sha71. Daniel Shanks. Class number, a theory of factorization and genera. In *Proceedings of Symposium in Pure Mathematics*, volume 20, pages 415–440. American Mathematical Society, 1971.

TMM21. Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. Post-quantum adaptor signature for privacy-preserving off-chain payments. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part II*, volume 12675 of *LNCS*, pages 131–150, Virtual Event, March 1–5, 2021. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-64331-0_7.

TMS23. Erkan Tairi, Pedro Moreno-Sanchez, and Clara Schneidewind. LedgerLocks: A security framework for blockchain protocols based on adaptor signatures. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 859–873, Copenhagen, Denmark, November 26–30, 2023. ACM Press. doi:10.1145/3576915.3623149.

TZY22. Binbin Tu, Min Zhang, and Chen Yu. Efficient ECDSA-based adaptor signature for batched atomic swaps. In Willy Susilo, Xiaofeng Chen, Fuchun Guo, Yudi Zhang, and Rolly Intan, editors, *ISC 2022*, volume 13640 of *LNCS*, pages 175–193, Bali, Indonesia, December 18–22, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-22390-7_12.

WNR. Pieter Wuille, Jonas Nick, and Tim Ruffing. Schnorr signatures for secp256k1. URL: https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki.