



**FACULTY
OF MATHEMATICS
AND PHYSICS**
Charles University

MASTER THESIS

Berenika Richterová

**On security of practical adaptor
signatures**

Computer Science Institute of Charles University

Supervisor of the master thesis: Mgr. Pavel Hubáček, Ph.D.

Study programme: Mathematics for Information
Technologies

Prague 2025

I declare that I carried out this master thesis on my own, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In date

Author's signature

I would like to express my sincere gratitude to my supervisor, Mgr. Pavel Hubáček, Ph.D., for his time and guidance throughout the course of my thesis.

I am also deeply grateful to my family for their constant encouragement, patience, and support during my studies.

Title: On security of practical adaptor signatures

Author: Berenika Richterová

Institute: Computer Science Institute of Charles University

Supervisor: Mgr. Pavel Hubáček, Ph.D., Computer Science Institute of Charles University

Abstract: In cryptography, adaptor signatures extend standard digital signatures. They allow the signer to make a promise on an actual signature, which is then revealed in exchange for a secret value. This functionality of adaptor signatures has recently been used in many applications regarding cryptocurrencies, including conditional payments. This thesis studies the security of adaptor signatures for Schnorr signature and ECDSA motivated by the conditional payments. We identify limitations of existing security definitions when applied to practical adaptor signatures for ECDSA and provide an example of such a limitation in the context of the Bitcoin blockchain. To address these problems, we present new security definitions and prove that the practical adaptor signature for ECDSA satisfies our notion of security.

Keywords: adaptor signatures, conditional payments, ECDSA, blockchain

Název práce: Bezpečnost adaptovatelných podpisových schémat

Autor: Berenika Richterová

Ústav: Informatický ústav Univerzity Karlovy

Vedoucí diplomové práce: Mgr. Pavel Hubáček, Ph.D., Informatický ústav Univerzity Karlovy

Abstrakt: Adaptovatelné podpisy v kryptografii představují rozšíření standardních digitálních podpisů. Umožňují podepisujícímu vytvořit závazek k podpisu, který lze následně získat výměnou za tajnou hodnotu. Tato vlastnost adaptovatelných podpisů se v poslední době využívá v mnoha aplikacích souvisejících s kryptoměny včetně podmíněných plateb. Naše práce se zabývá bezpečností adaptovatelných podpisových schémat pro Schnorrové podpisy a ECDSA, motivovanou právě scénářem podmíněných plateb. Identifikujeme omezení stávajících definic bezpečnosti, pokud je uvažujeme vzhledem k praktickým adaptovatelným podpisům pro ECDSA, a představíme i konkrétní problém v kontextu Bitcoinu. Navrhujeme nové definice bezpečnosti, které předcházejí těmto problémům, a dokážeme, že praktický adaptovatelný podpis pro ECDSA splňuje námi definovanou bezpečnost.

Klíčová slova: adaptovatelné podpisy, podmíněné platby, ECDSA, blockchain

Contents

| | |
|---|-----------|
| Introduction | 6 |
| 1 Non-Interactive Zero-Knowledge Proofs of Equality of Discrete Logarithms | 8 |
| 1.1 Random Oracle Model | 8 |
| 1.2 Non-Interactive Zero-Knowledge Proofs | 9 |
| 1.3 Standard Construction | 12 |
| 1.4 Applications | 16 |
| 2 Adaptor Signatures | 20 |
| 2.1 Structure of Adaptor Signatures | 20 |
| 2.2 Adaptor Signatures for Practical Signature Schemes | 21 |
| 2.2.1 Schnorr Adaptor Signatures | 22 |
| 2.2.2 ECDSA Adaptor Signatures | 24 |
| 2.3 Robust Security of Adaptor Signatures | 27 |
| 2.3.1 Unforgeability | 27 |
| 2.3.2 Witness Extractability | 29 |
| 2.3.3 Pre-Signature Adaptability | 31 |
| 2.3.4 Security of Adaptor Signatures for Practical Signature Schemes | 31 |
| 2.4 Limits of Robust Security of Adaptor Signatures | 34 |
| 3 Security of Practical Adaptor Signatures for ECDSA | 38 |
| 3.1 Unforgeability of Adaptor Signatures for ECDSA | 38 |
| 3.1.1 Definition of Existential Unforgeability | 38 |
| 3.1.2 Proof of Existential Unforgeability | 39 |
| 3.2 Witness Extractability of Adaptor Signatures for ECDSA | 50 |
| 3.2.1 Definition of Witness Extractability | 50 |
| 3.2.2 Proof of Witness Extractability | 51 |
| 3.3 Pre-Signature Adaptability of Adaptor Signatures for ECDSA . . | 56 |
| 3.3.1 Definition of Pre-Signature Adaptability | 56 |
| 3.3.2 Proof of Pre-Signature Adaptability | 56 |
| 3.4 Pre-Signature Unforgeability of Adaptor Signatures for ECDSA . | 57 |
| 3.4.1 Definition of Pre-Signature Unforgeability | 57 |
| 3.4.2 Proof of Pre-Signature Unforgeability | 58 |
| 4 Applications of Adaptor Signatures | 63 |
| 4.1 Relative Efficiency | 63 |
| 4.2 Discreet Log Contracts | 64 |
| Conclusion | 68 |
| Bibliography | 69 |

Introduction

Payments based on conditions related to events in the future are a common part of every-day life; from paying rent on the first day of the month, to bets, TV subscriptions, and other contracts. In cryptocurrencies, with rapid development, this functionality has been made possible by *smart contracts*; pieces of code that execute transactions according to the given conditions. However, the conditions are then stored in the contract publicly on blockchain, making the reasons why the transaction is being carried out visible to the whole world. Moreover, not all cryptocurrencies support arbitrary smart contracts and, since they are not a cryptographic solution, their security depends on the practices of the developers. Security is, however, a crucial part of these conditional payments, and thus it is reasonable to consider different solutions. To illustrate conditional payments, we give the following toy example, which is supported by the basic functionality of cryptocurrencies such as Bitcoin, where, in order to execute the transaction, it has to be digitally signed and published on the blockchain.

Bob is planning to join a marathon run in his hometown. To support and motivate him, Alice proposes a bet. If Bob completes the marathon in 3.5 hours, Alice will give him ten Friendcoins, if in 4 hours, she will give him five Friendcoins, and if in 5 hours, it will be one Friendcoin. In this example, the event in the future is Bob's marathon run, and the conditions depend on the outcome of the event; his final time. Alice can send the money to Bob by digitally signing a message, a transaction, by which she basically agrees to transfer the signed amount of her money to him. However, in this case, she cannot sign the transaction immediately, as it is based on a condition in the future, since otherwise, Bob could take the money even before the marathon started. Instead, we want to be able to *pre-sign* a message, i.e. to produce a *pre-signature* that would later, with knowledge of a secret value, transform into a signature on the message. This is formalized as *adaptor signature*, which is a term introduced by Poelstra [1]. Adaptor signatures or similar ideas and concrete constructions of such signatures were already presented and used in the literature [2], [3], however, they were first formalized as a standalone cryptographic primitive by Aumayr et al. [4]. In the same work, Aumayr et al. also formalized the security of adaptor signatures. This is especially important, since the applicability of adaptor signatures is fundamentally dependent on their security. The security is multidimensional, as we need to consider every party involved in the concrete application. The application also heavily influences the needed security directions. In our example, we have a security towards Alice, the sender and signer of the transaction, and security towards Bob, the receiver of the transaction. Before the run, Alice uses the adaptor signature to pre-sign the three transactions, obtaining three pre-signatures, each relative to one of the outcomes. She then sends them to Bob, and when the run is finished, on a high level, the final time provides Bob with a secret value, which Bob uses to *adapt* the correct pre-signature into a signature. To protect Alice, the sender, we want the pre-signatures to be adaptable into real signatures only if the related outcome really happens, i.e., Bob receives ten Friendcoins only if his marathon time is really under 3.5 hours. On the other hand, to protect Bob, the receiver, we want the contrary. If the condition is met, that is, if Bob's time is really under 3.5

hours, we want to ensure that the pre-signature will be adaptable into a signature, and Bob will receive the promised prize.

Structure of the Thesis

In Chapter 1, we introduce *non-interactive zero-knowledge proof systems*. In certain constructions of adaptor signatures, these proof systems provide a guarantee of adaptability and are a crucial part of their security. We present a standard construction of such a system for equality of discrete logarithms, and then, we formally prove the relevant properties of this construction. We conclude this chapter with a proof system for a scheme in [5], which is also relevant to conditional payments. The proof system uses the described non-interactive zero-knowledge proof for equality of discrete logarithms to prove equality of two encrypted plaintexts.

In Chapter 2, we formally present *adaptor signatures* and their security from [4]. We describe specific constructions of adaptor signatures for practical digital signature schemes such as Schnorr signature and ECDSA, and consider their security. In this chapter, we utilize the theory and the construction of non-interactive zero-knowledge proofs for equality of discrete logarithm discussed in Chapter 1. We then illustrate the limits of the security definitions in [4]. Considering the security definitions for adaptor signatures for ECDSA, we show a theoretical attack in the context of Bitcoin elliptic curve, which suggests that the original security definitions in [4] are not suitable for practical adaptor signatures for ECDSA in relevant applications.

In Chapter 3, we present our main contribution. Since there is no proof of security for practical adaptor signatures for ECDSA, we present new security definitions that do not allow our attack from Chapter 2. Then, we prove that the practical adaptor signature for ECDSA satisfies our notion of security.

Finally, in Chapter 4, we compare the efficiency of the constructions of adaptor signatures described in Chapter 2. We conclude the thesis with an application of adaptor signatures proposed in [6], a Bitcoin solution for conditional payments, which is based on the practical adaptor signature for ECDSA considered in Chapter 3. We propose that the security proven in the previous chapter could extend to the security of this solution.

In summary, the main contribution of this thesis lies in the new results presented in Chapter 3; additionally, the theoretical attack presented in Chapter 2 is also new.

1 Non-Interactive Zero-Knowledge Proofs of Equality of Discrete Logarithms

In this chapter, we describe non-interactive zero-knowledge (NIZK) proof systems for the discrete logarithm equality relation. The properties of such systems are crucial for many applications in cryptography. In our case, they serve as a building block in the design and security analysis of adaptor signatures in Chapter 2. In fact, they are an inseparable part of an adaptor signature for ECDSA presented in [4], which we describe later. Adaptor signatures are, however, not the only application of NIZK proofs for discrete logarithm equality regarding conditional payments. In a scheme presented by Madathil et al. [5], which proposes an alternative solution to the conditional oracle payment problem, non-interactive zero-knowledge proofs are used to determine whether two certain ciphertexts encrypt the same plaintext. This is, again, an important part of the security of the scheme. We provide more details in Section 1.4. First, we describe the random oracle model, so that we can formally define NIZK proof systems.

1.1 Random Oracle Model

In cryptographic protocols and schemes, we often use hash functions and prove the security of the constructions under, e.g., the collision resistance of the hash. However, often, collision resistance or one-wayness is insufficient for a security proof. Then, it is common to analyze the protocol in an idealized model, where we assume stronger properties that enable the security proof. One of such idealized models of security is a *random-oracle model* (ROM) [7], where we model the hash as an oracle that is accessible for querying by every party involved in the scheme. Technically, a random oracle is a sequence $\{\mathcal{O}_n\}_{n \in \mathbb{N}}$ such that $\mathcal{O}_n: \{0, 1\}^n \rightarrow \{0, 1\}^{l(n)}$ takes a string of bits of length n on input and outputs a string of bits of length $l(n)$ for some function l . For every input string $x \in \{0, 1\}^n$, on which the oracle has not been queried, $\mathcal{O}_n(x)$ is a uniformly random string from $\{0, 1\}^{l(n)}$. On an input on which the oracle has already been queried, we always obtain the same output. For simplicity, we denote the random oracle simply by $\mathcal{O}: \{0, 1\}^* \rightarrow \{0, 1\}^*$, for $\{0, 1\}^*$ a set of strings of bits of arbitrary length. Note that the oracle implements a truly random function that can be evaluated only by a query to the oracle. This is often exploited in security analyses in ROM, specifically in proofs by reduction, where a random oracle is simulated to an adversary, i.e., the answers of the oracle on the adversary's queries are simulated since they can be sampled from an identical distribution. Since \mathcal{O} implements a random function, the oracle can thus be *programmed* by the reduction, i.e. the reduction can choose a value $\mathcal{O}(x)$ for any input x , when the output is distributed identically. We call a partial function $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ with a domain $\mathcal{D}(f) \subset \{0, 1\}^*$ a *programming*. We denote by $\mathcal{O}|_f$ a *random oracle*

restricted by the programming, which is defined as

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

To behave like a random oracle, we need $\mathcal{O}|_f$ to be identically distributed as \mathcal{O} . We will utilize this notation in the definitions in the following chapter. We call an algorithm that is allowed to query a random oracle an *oracle-aided* algorithm. Note that, for simplicity of notation, we call the oracle a random function, unifying the oracle and the random function it implements.

Random Oracle Model and Standard Model. For the use of cryptographic protocols in the real world, since we do not possess a random function, a random oracle is instantiated with some cryptographic hash function, for instance *SHA-256*. A security in ROM does not imply a security in the standard model, since the properties of the random oracle are clearly much stronger than the properties of any specific hash function. Moreover, the hash function is described to all parties in advance, and the parties can evaluate it on their own. This poses as a major difference from the random oracle. However, the security proofs in ROM are standard in modern cryptography, and, together with a reasonable choice of a cryptographic hash function used in the analyzed protocol, they seemly justify implementing the protocol in practice. See [7] for additional discussion.

1.2 Non-Interactive Zero-Knowledge Proofs

We now turn to the notion of non-interactive zero-knowledge proofs. In general, NIZK proofs w.r.t. a relation $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$ are proving the membership of a statement s to a language \mathcal{L}_R associated with R in the following way

$$\mathcal{L}_R = \{s \mid \exists w \text{ s.t. } (s, w) \in R\}.$$

We call w a witness for the statement s , and the tuple (s, w) a statement-witness pair. It is usual to denote the statements of a relation by x , however, since x is commonly used to denote the discrete logarithm and we follow this notation in the relations defined in the subsequent sections, for clarity, we use s instead. In the thesis, we also use the following notation. We denote by $\lambda \in \mathbb{N}_0$ a security parameter and by 1^λ a string of 1s of length λ . By $s \leftarrow X$, we denote a uniform sampling of the element s from the set X , and by $w \leftarrow \mathcal{A}(s)$ a situation, where a probabilistic algorithm \mathcal{A} outputs w on input s . For a deterministic algorithm, we use $w := \mathcal{A}(s)$. For brevity, when referring to algorithms, instead of probabilistic polynomial-time, we use PPT. Finally, by $|s|$, we denote the length of s as a binary string.

Hard Relations. In general, the security of cryptographic protocols often depends on some computational hardness assumption, i.e., the hypothesis that some problem cannot be solved efficiently. The security of the protocol is then reduced to the hardness of the underlying problem. In our context, we assume the hardness of relation R , where finding the prospective witness for the statement

in \mathcal{L}_R is, in some sense, difficult. In the context of NIZK proofs, while it is possible to define them w.r.t. any polynomial-time decidable relation, it is usually reasonable to consider such hard relations. Otherwise, it would be easy to check the membership by extracting w , making the construction of a NIZK proof unnecessary. We define such hard relations next.

Definition 1. *Let $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$ be a relation. We say that R is a hard relation if the following hold.*

(i) *R is polynomial-time decidable.*

(ii) *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} , there exists a negligible function negl satisfying*

$$\Pr[(s, \mathcal{A}(s, 1^\lambda)) \in R \mid (s, w) \leftarrow \text{GenR}(1^\lambda)] \leq \text{negl}(\lambda).$$

Hard relations naturally correspond to NP languages; for a hard relation R , \mathcal{L}_R is by definition in NP. Specifically, by definition, it is easy to decide if a given witness w proves that a statement s is in \mathcal{L}_R . However, it is difficult for any PPT adversary to produce a valid witness to a sampled statement.

Discrete Logarithm Relation. We now give a classical example of a hard relation used in cryptography.

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

$$DLOG := \{((q, \mathbb{G}, g, y), x) \mid y = g^x\},$$

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

Technically, the statement of the DLOG relation is the full 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 group generator g , and we consider only

$$\{((g, y), x) \in \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_q \mid y = g^x\}$$

to be statement-witness pairs of DLOG. Moreover, if we assume that DLOG is a hard relation, by the definition of the hardness of the relation in (ii) of Definition 1, we have the following. Any PPT adversary should not be able to produce with a non-negligible probability a witness of a DLOG statement sampled by GenR ; the sampling algorithm for DLOG. Since the statement includes the group \mathbb{G} where the discrete logarithm should be computed, each sample might produce a different \mathbb{G} . However, this does not translate to real-world applications, where the group \mathbb{G} is usually fixed and is believed to be secure enough, and where we want the relation to be hard even when sampling an element $y \in \mathbb{G}$ in the statement from the same group several times. If we fix a specific group, we cannot draw any conclusions about the hardness of the relation in the asymptotic sense. For this, we divide the sampling algorithm into two algorithms. For the DLOG relation, we consider a PPT algorithm GenG that on input 1^λ samples a group \mathbb{G} with a

generator g and a PPT algorithm **GenR**. The relation sampling algorithm **GenR** then on input 1^λ samples the discrete logarithm $x \leftarrow \mathbb{Z}_q$ uniformly randomly and outputs a statement-witness pair (g^x, x) . Note that this view on the hardness of the DLOG relation is stronger than in Definition 1, since now the group \mathbb{G} is considered sampled, and for any such group, it should still be infeasible to compute with non-negligible probability the discrete logarithm of an element $y \in \mathbb{G}$ sampled by **GenR**. We utilize this view in the following chapters. Groups, where DLOG is assumed to be hard, are the foundations of many applications in cryptography, including encryption and signature schemes.

Notice that the order q of \mathbb{G} is related to the security parameter λ for DLOG to be a hard relation. If q was polynomial in λ , then a PPT adversary could find the corresponding witness for condition (ii) with a non-negligible probability simply by guessing or by brute force. Specifically, it is clear that q^{-1} must be negligible in the security parameter λ .

Non-Interactive Zero-Knowledge Proofs. As established above, the purpose of NIZK proofs is for a prover to prove to a verifier the membership of statements in a language \mathcal{L}_R . In NIZK proof systems, the prover generates the proof autonomously and provides it to the verifier for verification, which is suitable for many applications in cryptography. This differs compared to the standard interactive proof systems, where the prover and verifier communicate. Since NIZK proofs remove communication, the prover cannot respond to the randomness chosen by the verifier as in interactive proof systems, which could potentially allow cheating. Instead, the prover and the verifier have access to a random oracle, which is unpredictable and thus can simulate the challenges chosen by the verifier. However, in NIZK proof systems, proving membership of statements is not the only property that we require. We also want the proof not to reveal any information additional about the potential witness.

Definition 3. *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} :*

- $\mathbf{P}^{\mathcal{O}}(s, w)$ 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 .
- $\mathbf{V}^{\mathcal{O}}(s, \pi)$ 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 (\mathbf{P}, \mathbf{V}) to satisfy the following properties:

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

$$\Pr[\mathbf{V}^{\mathcal{O}}(s, \pi) = 1 \mid \pi \leftarrow \mathbf{P}^{\mathcal{O}}(s, w)] = 1,$$

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

Computational soundness: For all oracle-aided PPT \mathcal{A} , there exists a negligible function negl such that

$$\Pr\left[|s| \geq \lambda \wedge s \notin \mathcal{L}_R \wedge \mathbf{V}^{\mathcal{O}}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)\right] \leq \text{negl}(\lambda),$$

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

Perfect Zero Knowledge: (\mathbf{P}, \mathbf{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 $(\pi, f) \leftarrow \mathcal{S}(s)$ such that $(\pi, \mathcal{O}|_f)$ is identically distributed as $(\mathbf{P}^{\mathcal{O}}(s, w), \mathcal{O})$.

Informally, for the NIZK proof to be complete, we want the verifier \mathbf{V} to always accept a proof generated by the honest prover \mathbf{P} for any statement in \mathcal{L}_R . On the other hand, we require that any PPT prover generates an accepting proof of a sufficiently long invalid statement of its choice only with a negligible probability. The length of the chosen statement is lower bounded by the security parameter λ to rule out the trivial case where, with growing λ , the adversary could output the same very short invalid statement with a valid proof, and convince the verifier with probability that would then be non-negligible in λ . Finally, the perfect zero-knowledge property captures the ability of the proof system to prove the membership of s in \mathcal{L}_R without leaking any new information about the witness w . The definition of the zero-knowledge property ensures that the verifier \mathbf{V} cannot distinguish between a random oracle \mathcal{O} and $\mathcal{O}|_f$, the oracle restricted by the programming given by the simulator. Given access to $\mathcal{O}|_f$, it also cannot distinguish between the proofs generated by the honest prover and the ones generated by the simulator.

1.3 Standard Construction

This section describes a standard construction of the NIZK proof system for the equality of two discrete logarithms [8] in cyclic groups of the same prime order q . Consider two elements y_1, y_2 of two such groups. Essentially, we want to prove without revealing any additional information, that for the discrete logarithms x_1, x_2 of y_1, y_2 w.r.t. some specified generators g_1, g_2 , we have $x_1 \equiv_q x_2$. For simplicity of notation, we use $=$ instead of \equiv_q . Let us define the corresponding relation.

Definition 4. We define a discrete logarithm equality relation as

$$DLEQ = \{((q, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, y_1, y_2), x) \mid y_1 = g_1^x \wedge y_2 = g_2^x\},$$

where $\mathbb{G}_1, \mathbb{G}_2$ are descriptions including the operations of two cyclic groups of the same prime order q generated by g_1, g_2 , respectively, $y_1 \in \mathbb{G}_1, y_2 \in \mathbb{G}_2$ and $x \in \mathbb{Z}_q$.

Similarly to the discussion regarding DLOG, for simplicity of notation, we assume that the group order q and the group descriptions $\mathbb{G}_1, \mathbb{G}_2$ are implicitly given by each of the group generators g_1, g_2 . From now on, we consider the statement-witness pairs of DLEQ to be elements of set

$$\{((g_1, g_2, y_1, y_2), x) \mid y_1 = g_1^x \wedge y_2 = g_2^x\}.$$

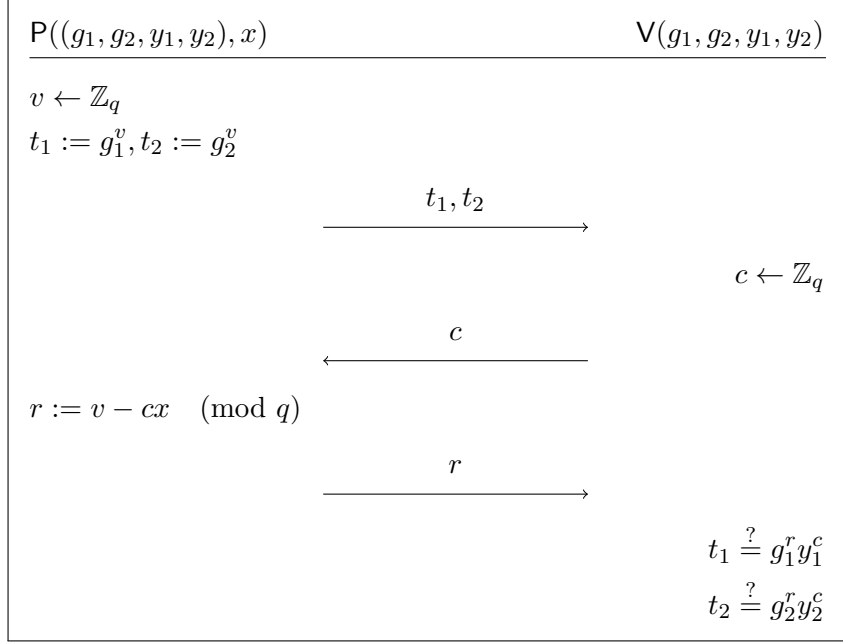


Figure 1.1 Interactive proof of the equality of discrete logarithms [8].

Note that Definition 3 is a general definition of a NIZK proof system. As the NIZK proofs are usually part of larger schemes, where the corresponding group is sampled and fixed, the adversary is expected to produce an invalid statement consisting of group elements of the expected length. To avoid technical problems, we require that the length of the elements of the group is greater than or equal to the security parameter, i.e., $\log(q) \geq \lambda$.

First, in Figure 1.1, we show a well-known interactive version of the required protocol [8], where the prover and the verifier communicate. The prover P has both the statement (g_1, g_2, y_1, y_2) and the witness x on input, whereas the verifier V gets only the statement. First, the prover samples a random element $v \in \mathbb{Z}_q$, computes elements $g_1^v \in \mathbb{G}_1, g_2^v \in \mathbb{G}_2$ and sends them to the verifier as a commitment. The verifier then returns a random challenge $c \in \mathbb{Z}_q$, and the prover responds with a corresponding $r \in \mathbb{Z}_q$ computed as $r := v - cx$. Finally, the verifier checks if both equations described in Figure 1.1 hold and outputs 1 or 0, accordingly. The prover convinces the verifier of the validity of the statement if the communication between P and V results in V accepting.

We present the standard NIZK proof system for DLEQ in Figure 1.2, for which we assume that both the prover P_{DLEQ} and the verifier V_{DLEQ} have access to a hash function $\mathsf{H}: \{0, 1\}^* \rightarrow \mathbb{Z}_q$. This protocol is derived from the interactive version by the standard Fiat-Shamir transformation [9]; the parties do not communicate, each party separately computes the challenge c with H and the specified group elements including the commitment. In this proof system, the final proof is a tuple (c, r) . The verifier obtains a proof (c, r) on input as well as a statement and verifies it as described in Figure 1.2.

For completeness, we prove in Theorem 3 that the protocol described in Figure 1.2 satisfies the NIZK properties in the random oracle model, that is, where the hash function H is modeled as a random oracle. We prove completeness and

| $\text{P}_{\text{DLEQ}}((g_1, g_2, y_1, y_2), x)$ | $\text{V}_{\text{DLEQ}}((g_1, g_2, y_1, y_2), \pi)$ |
|---|---|
| 1 : $v \leftarrow \mathbb{Z}_q$ | 1 : $(c, r) := \pi$ |
| 2 : $t_1 := g_1^v, t_2 := g_2^v$ | 2 : $t_1 := g_1^r y_1^c, t_2 := g_2^r y_2^c$ |
| 3 : $c \leftarrow \text{H}(g_1, g_2, y_1, y_2, t_1, t_2)$ | 3 : return |
| 4 : $r := v - cx \pmod{q}$ | $c = \text{H}(g_1, g_2, y_1, y_2, t_1, t_2)$ |
| 5 : return $\pi := (c, r)$ | |

Figure 1.2 Proof of the equality of discrete logarithms [8].

the zero-knowledge property of the construction fully in Theorem 3, however, for computational soundness, we first formulate the following lemmata. In Lemma 1, we compute an upper bound on the probability that, for an invalid DLEQ statement and chosen commitments, there exists a valid proof. We use Lemma 1 to prove Lemma 2, by which we obtain an upper bound on the probability that an arbitrary oracle-aided adversary produces an invalid DLEQ statement with a valid proof. In Theorem 3, we show that the computational soundness of the construction follows from Lemma 2.

Lemma 1. *For all statements $(g_1, g_2, y_1, y_2) \notin \mathcal{L}_{\text{DLEQ}}$ and all choices of commitments $t_1 \in \mathbb{G}_1, t_2 \in \mathbb{G}_2$ related to the proof in Figure 1.1, we have*

$$\Pr_c[\exists r : t_1 = g_1^r y_1^c \wedge t_2 = g_2^r y_2^c] \leq q^{-1},$$

where the probability is over the choice of the challenge c .

Proof. Let us fix a statement $(g_1, g_2, y_1 = g_1^{x_1}, y_2 = g_2^{x_2}) \notin \mathcal{L}_{\text{DLEQ}}$ and commitments $t_1 \in \mathbb{G}_1, t_2 \in \mathbb{G}_2$ as stated. Then $x_1 \neq x_2$ and there exist elements $v_1, v_2 \in \mathbb{Z}_q$ such that

$$t_1 = g_1^{v_1} \quad \text{and} \quad t_2 = g_2^{v_2}.$$

Therefore, for any challenge $c \leftarrow \mathbb{Z}_q$ and any response $r \in \mathbb{Z}_q$ for which the following holds

$$\begin{aligned} t_1 &= g_1^r y_1^c \\ t_2 &= g_2^r y_2^c, \end{aligned}$$

we obtain

$$g_1^{v_1} = g_1^r y_1^c = g_1^r g_1^{x_1 c}.$$

Analogously for t_2 , we get

$$g_2^{v_2} = g_2^r g_2^{x_2 c}.$$

For the exponents, we have

$$\begin{aligned} r + x_1 c &= v_1 \\ r + x_2 c &= v_2. \end{aligned}$$

By subtraction of the equations, we get

$$(x_1 - x_2)c = v_1 - v_2,$$

and since $x_1 \neq x_2$, we can prescribe c as

$$c = \frac{v_1 - v_2}{x_1 - x_2}.$$

Thus, we obtain an upper bound on the desired probability

$$\Pr_c[\exists r: t_1 = g_1^r y_1^c \wedge t_2 = g_2^r y_2^c] \leq \Pr_c\left[c = \frac{v_1 - v_2}{x_1 - x_2}\right] = \frac{1}{q},$$

since c is uniformly distributed in \mathbb{Z}_q . \square

Lemma 2. *Consider V_{DLEQ} to be the verifier described in Figure 1.2, where H is modeled as a random oracle \mathcal{O} . For all oracle-aided $\mathcal{A}^\mathcal{O}$ making ρ \mathcal{O} -queries, we have*

$$\Pr\left[|s| \geq \lambda \wedge s \notin \mathcal{L}_{DLEQ} \wedge V_{DLEQ}^\mathcal{O}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^\mathcal{O}(1^\lambda)\right] \leq (\rho + 1)q^{-1},$$

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

Proof. Consider an oracle-aided adversary $\mathcal{A}^\mathcal{O}$. By making an oracle query, $\mathcal{A}^\mathcal{O}$ fixes a choice of a statement $s := (g_1, g_2, y_1, y_2)$ and commitments t_1, t_2 , as they are part of the query. Assume a statement $s \notin \mathcal{L}_{DLEQ}$. Since \mathcal{O} is a random oracle, Lemma 1 gives

$$\Pr_c[\exists r: t_1 = g_1^r y_1^c \wedge t_2 = g_2^r y_2^c] \leq q^{-1},$$

where the probability is over the choice of challenge c . Thus, the probability of \mathcal{A} finding such a suitable r in one oracle-query is upper bounded by q^{-1} . Since the answers of the oracle are sampled uniformly randomly, the queries are independent, i.e. the answers of \mathcal{O} on several queries do not help predict the answer of a new query. The adversary can also output a proof (c, r) , for which \mathcal{A} did not query \mathcal{O} . Then, since \mathcal{O} is a random oracle, we have

$$\Pr_{\mathcal{O}}[c = \mathcal{O}(g_1, g_2, y_1, y_2, g_1^r y_1^c, g_2^r y_2^c)] = q^{-1}.$$

Thus, with ρ queries and $\log(q) \geq \lambda$, we obtain

$$\Pr\left[|s| \geq \lambda \wedge s \notin \mathcal{L}_{DLEQ} \wedge V_{DLEQ}^\mathcal{O}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^\mathcal{O}(1^\lambda)\right] \leq (\rho + 1)q^{-1}.$$

\square

We can now turn to the following theorem.

Theorem 3. *The protocol in Figure 1.2 is a NIZK proof system of equality of discrete logarithms in the random oracle model.*

Proof. Using the notation from Figure 1.2, we show:

Completeness: This follows directly by inspection of the protocol. Specifically, for all $((g_1, g_2, y_1, y_2), x) \in \text{DLEQ}$, we have

$$\begin{aligned} g_1^v y_1^c &= g_1^{v-cx} g_1^{cx} = g_1^v, \\ g_2^v y_2^c &= g_2^{v-cx} g_2^{cx} = g_2^v. \end{aligned}$$

| $\mathcal{S}_{\text{DLEQ}}(g_1, g_2, y_1, y_2)$ |
|--|
| 1 : $r \leftarrow \mathbb{Z}_q$ |
| 2 : $c \leftarrow \mathbb{Z}_q$ |
| 3 : $t_1 := g_1^r y_1^c, t_2 := g_2^r y_2^c$ |
| 4 : return $(\pi, f) := ((c, r), ((g_1, g_2, y_1, y_2, t_1, t_2) \mapsto c))$ |

Figure 1.3 Simulator $\mathcal{S}_{\text{DLEQ}}$.

Computational soundness: Consider an oracle-aided PPT $\mathcal{A}^{\mathcal{O}}$. By Lemma 2, we have

$$\Pr\left[|s| \geq \lambda \wedge s \notin \mathcal{L}_{\text{DLEQ}} \wedge \mathbf{V}_{\text{DLEQ}}^{\mathcal{O}}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)\right] \leq (\rho + 1)q^{-1},$$

where the probability is over the choice of \mathcal{O} and a randomness of \mathcal{A} , and ρ is a number of oracle queries. Since \mathcal{A} is a PPT adversary, ρ is polynomial in the security parameter λ . As $\log(q) \geq \lambda$, we have $q^{-1} \leq 1/2^\lambda$, which is negligible in the security parameter. Thus, the whole probability is negligible in λ .

Perfect Zero Knowledge: We construct the required simulator in Figure 1.3.

Consider a statement $(g_1, g_2, y_1, y_2) \in \mathcal{L}_{\text{DLEQ}}$. The simulator starts with sampling elements $c, r \leftarrow \mathbb{Z}_q$, which form the simulated proof (c, r) . Then, it computes t_1 and t_2 as $g_1^r y_1^c$ and $g_2^r y_2^c$, and sets the programming f to $(g_1, g_2, y_1, y_2, t_1, t_2) \mapsto c$.

Immediately, as c is chosen by \mathcal{S} uniformly at random, we get that \mathcal{O} and $\mathcal{O}|_f$ are distributed identically.

By the construction of $\mathcal{S}_{\text{DLEQ}}$, the simulated proof $(c, r) \leftarrow \mathbb{Z}_q \times \mathbb{Z}_q$. Since \mathbf{H} in the prover \mathbf{P}_{DLEQ} in Figure 1.2 is modeled as a random oracle and v is chosen uniformly as random, we have for the real proof $(c, r) \leftarrow \mathbb{Z}_q \times \mathbb{Z}_q$. Thus, they are distributed identically. \square

By \mathbf{P}_{DLEQ} , \mathbf{V}_{DLEQ} and $\mathcal{S}_{\text{DLEQ}}$ we denote a NIZK proof system for equality of discrete logarithms in general. In some cases, we will specify the construction shown in Figure 1.2 together with the corresponding simulator in Figure 1.3, which will also be referenced in the following chapters.

1.4 Applications

In the design of adaptor signatures for ECDSA, the use of a NIZK proof for discrete logarithm equality relation is rather straightforward, and we discuss it in detail in Section 2.2.2 of Chapter 2. Here, we illustrate the use of such NIZK proof that is still simple, although not as straightforward. Note that in this section, we use the original notation of the following scheme, which differs from the notation used throughout the rest of the thesis.

A scheme by Madathil et al. [5] provides a different approach to conditional future payments. They introduce Witness Encryption schemes based on Signatures (WES) w.r.t. a digital signature scheme $\text{DS} = (\text{KeyGen}, \text{Sign}, \text{Vrfy})$. We will

| KeyGen(1^λ) | Sign _{sk} (m) | Vrfy _{pk} (m, σ) |
|---------------------------------------|----------------------------|--|
| 1 : $x \leftarrow \mathbb{Z}_q$ | 1 : $(X, x) := \text{sk}$ | 1 : $X := \text{vk}$ |
| 2 : $X := g_0^x$ | 2 : $h := \text{H}_0(m)$ | 2 : return |
| 3 : return | 3 : $\sigma := h^x$ | $(e(g_0, \sigma) = e(X, \text{H}_0(m)))$ |
| $\text{sk} := (X, x), \text{vk} := X$ | 4 : return σ | |

Figure 1.4 BLS signature scheme.

formally define and discuss digital signatures in Chapter 2. Crucially, WES encrypts a plaintext w.r.t. (m, vk) , a pair of a message and a valid verifying key, i.e., a verifying key that is related to a valid key pair (sk, vk) for DS. Then, anyone providing a valid signature σ on m that verifies under this vk , i.e., $\text{Vrfy}_{\text{vk}}(m, \sigma) = 1$, can decrypt the ciphertext. Their specific candidate is based on the BLS signature scheme [10].

Consider now groups $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$ of prime order q , generators g_0, g_1 of $\mathbb{G}_0, \mathbb{G}_1$, respectively, an efficiently computable bilinear pairing $e: \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ and hash functions H_0, H_1 such that $\text{H}_0: \{0, 1\}^\lambda \rightarrow \mathbb{G}_1$ and $\text{H}_1: \mathbb{G}_T \rightarrow \{0, 1\}^\lambda$. We formally describe the BLS signatures in Figure 1.4. Crucially, a message $m \in \{0, 1\}^*$ is signed using a signing key $x \leftarrow \mathbb{Z}_q$ simply by computing $\text{H}_0(m)^x$. Then, a signature σ is verified by checking if $e(g_0, \sigma)$ and $e(\text{vk}, \text{H}_0(m))$ coincide for a verifying key vk .

For WES based on the BLS signatures by Madathil et al., during encryption, a pair of random coins $(r_1, r_2) \leftarrow \mathbb{Z}_q \times \mathbb{G}_T$ is chosen uniformly at random. An integer $r \in \mathbb{Z}_q$ is encrypted with these coins and w.r.t. $(m, \text{vk}) \in \{0, 1\}^\lambda \times \mathbb{G}_0$ as follows

$$\text{Enc}((\text{vk}, m), r; (r_1, r_2)) := (g_0^{r_1}, e(\text{vk}, \text{H}_0(m))^{r_1} \cdot r_2, \text{H}_1(r_2) + r),$$

where the plaintext r as a bit string is one-time padded with $\text{H}_1(r_2)$. Clearly, a BLS signature on m under vk allows to decrypt the ciphertext.

The scheme for conditional oracle-based payments by Madathil et al. using the above WES depends on the existence of a NIZK proof attesting that two WES ciphertexts are well-formed and encrypt the same plaintext with the same randomness during the encryption. Formally, we need a NIZK proof for a relation R defining the following language

$$\mathcal{L}_R = \left\{ (\text{vk}_1, \text{vk}_2, m_1, m_2, c_1, c_2) \left| \begin{array}{l} \exists r \in \mathbb{Z}_q, r_1 \in \mathbb{Z}_q, r_2 \in \mathbb{G}_T \text{ s.t.} \\ c_1 = \text{Enc}((\text{vk}_1, m_1), r; (r_1, r_2)) \wedge \\ c_2 = \text{Enc}((\text{vk}_2, m_2), r; (r_1, r_2)) \end{array} \right. \right\}, \quad (1.1)$$

where $\text{vk}_1, \text{vk}_2 \in \mathbb{G}_0, m_1, m_2 \in \{0, 1\}^\lambda$.

In [5], the authors did not give a construction of such a NIZK, and only suggested using a NIZK proof of equality of discrete logarithms. We now give an explicit formal description of the required proof in Figure 1.5, and, in Theorem 4, we show that this proof satisfies the properties of a NIZK proof.

For a statement-witness pair $(s, w) \in R$, the prover $\text{P}_{\mathcal{L}_R}$ in Figure 1.5 first parses each of the three parts of both ciphertexts c_1, c_2 appearing in the statement s . Then, it uses the second parts of c_1 and c_2 and the bilinear pairing e to produce

| $P_{\mathcal{L}_R}((\mathbf{vk}_1, \mathbf{vk}_2, m_1, m_2, c_1, c_2), (r, r_1, r_2))$ | $V_{\mathcal{L}_R}((\mathbf{vk}_1, \mathbf{vk}_2, m_1, m_2, c_1, c_2), \pi)$ |
|--|--|
| 1 : $c_1^1, v_1, c_1^3 := c_1$ | 1 : $c_1^1, v_1, c_1^3 := c_1$ |
| 2 : $c_2^1, v_2, c_2^3 := c_2$ | 2 : $c_2^1, v_2, c_2^3 := c_2$ |
| 3 : $u_1 := e(\mathbf{vk}_1, H_0(m_1))$ | 3 : $u_1 := e(\mathbf{vk}_1, H_0(m_1))$ |
| 4 : $u_2 := e(\mathbf{vk}_2, H_0(m_2))$ | 4 : $u_2 := e(\mathbf{vk}_2, H_0(m_2))$ |
| 5 : $h := u_1 u_2^{-1}$ | 5 : $h := u_1 u_2^{-1}$ |
| 6 : $\pi \leftarrow P_{\text{DLEQ}}((g_0, h, c_1^1, v_1 v_2^{-1}), r_1)$ | 6 : return $(V_{\text{DLEQ}}((g_0, h, c_1^1, v_1 v_2^{-1}), \pi)$ |
| 7 : return π | $\wedge (c_1^1 = c_2^1) \wedge (c_1^3 = c_2^3))$ |

Figure 1.5 NIZK proof for R defining \mathcal{L}_R (1.1).

a DLEQ statement-witness pair as described in the figure. Finally, it uses P_{DLEQ} , a NIZK prover of equality of discrete logarithms, to produce a proof π for this pair, and outputs π . The verification of π on the statement s proceeds as follows. The verifier $V_{\mathcal{L}_R}$ denotes the parts of c_1, c_2 and constructs the DLEQ statement exactly as the prover. Then, it checks if the first parts of c_1, c_2 are equal to the third parts of c_1, c_2 , respectively, and verifies π on the constructed DLEQ statement using V_{DLEQ} . Finally, $V_{\mathcal{L}_R}$ outputs 1 or 0, accordingly. The proof system $(P_{\text{DLEQ}}, V_{\text{DLEQ}})$ can be instantiated with the specific construction in Figure 1.2.

Theorem 4. *Proof in Figure 1.5 is a NIZK proof for relation R defining a language \mathcal{L}_R in (1.1) in the random oracle model.*

Proof. We show correctness, computational soundness and zero-knowledge property of a NIZK proof under the notation from Figure 1.5. Correctness and zero-knowledge property are straightforward. For computational soundness, we inspect the construction of $V_{\mathcal{L}_R}$ and show that if a statement $s \notin \mathcal{L}_R$, and $V_{\mathcal{L}_R}$ verifies on (s, π) , then the corresponding tuple $(g_0, h, c_1^1, v_1 v_2^{-1})$ defined in Figure 1.5 cannot be a valid DLEQ statement, i.e., we get $(g_0, h, c_1^1, v_1 v_2^{-1}) \notin \mathcal{L}_{\text{DLEQ}}$. This allows us to reduce computational soundness of the scheme to computational soundness of a NIZK proof for equality of discrete logarithms.

Correctness: For every statement-witness pair

$$((\mathbf{vk}_1, \mathbf{vk}_2, m_1, m_2, c_1, c_2), (r, r_1, r_2)) \in R,$$

the prover $P_{\mathcal{L}_R}$ creates a proof for ciphertexts c_1, c_2 of the form

$$c_1 = \text{Enc}((\mathbf{vk}_1, m_1), r) = \left(\underbrace{g_0^{r_1}}_{c_1^1}, \underbrace{e(\mathbf{vk}_1, H_0(m_1))^{r_1} \cdot r_2}_{v_1}, \underbrace{H_1(r_2) + r}_{c_1^3} \right),$$

$$c_2 = \text{Enc}((\mathbf{vk}_2, m_2), r) = \left(\underbrace{g_0^{r_1}}_{c_2^1}, \underbrace{e(\mathbf{vk}_2, H_0(m_2))^{r_1} \cdot r_2}_{v_2}, \underbrace{H_1(r_2) + r}_{c_2^3} \right).$$

For such pair of ciphertexts, clearly, the first and last coordinates are equal. It also holds

$$v_1 = u_1^{r_1} r_2 \quad \text{and} \quad v_2 = u_2^{r_1} r_2.$$

Thus, we obtain

$$\begin{aligned} c_1^1 &= g_0^{r_1}, \\ v_1 v_2^{-1} &= u_1^{r_1} r_2 (u_2^{r_1} r_2)^{-1} = (u_1 u_2^{-1})^{r_1}. \end{aligned}$$

We can see that the discrete logarithms of c_1^1 and $v_1 v_2^{-1}$ w.r.t. g_0 and $u_1 u_2^{-1}$, respectively, are equal. The rest follows from the completeness of the underlying NIZK proof of equality of discrete logarithms.

Computational soundness: Consider a PPT adversary \mathcal{A} , and, for a statement s , denote the corresponding tuple $(g_0, h, c_1^1, v_1 v_2^{-1})$ by \tilde{s} . Then, we have

$$\begin{aligned} &\Pr\left[|s| \geq \lambda \wedge s \notin \mathcal{L}_R \wedge \mathbf{V}_{\mathcal{L}_R}^{\mathcal{O}}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)\right] \\ &= \Pr\left[|\tilde{s}| \geq \lambda \wedge \tilde{s} \in \mathcal{L}_{\text{DLEQ}} \wedge s \notin \mathcal{L}_R \wedge \mathbf{V}_{\mathcal{L}_R}^{\mathcal{O}}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)\right] \\ &\quad + \Pr\left[|\tilde{s}| \geq \lambda \wedge \tilde{s} \notin \mathcal{L}_{\text{DLEQ}} \wedge s \notin \mathcal{L}_R \wedge \mathbf{V}_{\mathcal{L}_R}^{\mathcal{O}}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)\right], \end{aligned}$$

since $\log(q) \geq \lambda$ and $|s| \geq |\tilde{s}|$. By the definition of the verifier for R , we have

$$\mathbf{V}_{\mathcal{L}_R}^{\mathcal{O}}(s, \pi) = 1$$

if and only if

$$c_1^1 = c_2^1 \wedge c_1^3 = c_2^3 = c_2^3 \wedge \mathbf{V}_{\text{DLEQ}}^{\mathcal{O}}(\tilde{s}, \pi) = 1.$$

However, if the latter holds and $\tilde{s} \in \mathcal{L}_{\text{DLEQ}}$, then there exists $r_1 \in \mathbb{Z}_q$ such that

$$c_1^1 = c_2^1 = g_0^{r_1} \quad \text{and} \quad v_1 v_2^{-1} = (u_1 u_2^{-1})^{r_1},$$

which yields

$$v_1 u_1^{-r_1} = v_2 u_2^{-r_1}.$$

Thus, there exists $r_2 \in \mathbb{Z}_q$ such that

$$v_1 = e(\mathbf{vk}_1, \mathbf{H}_0(m_1))^{r_1} \cdot r_2 \quad \text{and} \quad v_2 = e(\mathbf{vk}_2, \mathbf{H}_0(m_2))^{r_1} \cdot r_2.$$

Since $c_1^3 = c_2^3$, there also exists r such that both ciphertexts encrypt r under the same random coins r_1 and r_2 . This means that $s \in \mathcal{L}_R$, which gives

$$\Pr\left[|\tilde{s}| \geq \lambda \wedge \tilde{s} \in \mathcal{L}_{\text{DLEQ}} \wedge s \notin \mathcal{L}_R \wedge \mathbf{V}_{\mathcal{L}_R}^{\mathcal{O}}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)\right] = 0.$$

We can continue with the following probability.

$$\begin{aligned} &\Pr\left[|\tilde{s}| \geq \lambda \wedge \tilde{s} \notin \mathcal{L}_{\text{DLEQ}} \wedge s \notin \mathcal{L}_R \wedge \mathbf{V}_{\mathcal{L}_R}^{\mathcal{O}}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)\right] \\ &\leq \Pr\left[|\tilde{s}| \geq \lambda \wedge \tilde{s} \notin \mathcal{L}_{\text{DLEQ}} \wedge \mathbf{V}_{\text{DLEQ}}^{\mathcal{O}}(s, \pi) = 1 \mid (\tilde{s}, \pi) \leftarrow \mathcal{B}^{\mathcal{O}}(1^\lambda)\right] \\ &\leq \mathbf{negl}(\lambda), \end{aligned}$$

where \mathbf{negl} is a negligible function and \mathcal{B} is a PPT adversary that uses \mathcal{A} to get (s, π) and computes the statement \tilde{s} from s . The last inequality follows from the computational soundness of the underlying NIZK proof system for equality of discrete logarithms. Together, we obtain

$$\Pr\left[|s| \geq \lambda \wedge s \notin \mathcal{L}_R \wedge \mathbf{V}_{\mathcal{L}_R}^{\mathcal{O}}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)\right] \leq \mathbf{negl}(\lambda),$$

which concludes the proof.

Perfect zero-knowledge: This follows immediately from the property of the perfect zero-knowledge of the NIZK proof system producing π . \square

2 Adaptor Signatures

2.1 Structure of Adaptor Signatures

Adaptor signatures extend standard digital signatures. First, we define standard *digital signature schemes*.

Definition 5. We define a digital signature scheme as a tuple of algorithms $\Sigma = (\text{KeyGen}, \text{Sign}, \text{Vrfy})$ such that

$\text{KeyGen}(1^\lambda)$ is a PPT key generation algorithm that takes the security parameter 1^λ on input and outputs pair (sk, pk) of a signing key sk and a verifying key pk .

$\text{Sign}_{\text{sk}}(m)$ is a PPT algorithm that takes a message $m \in \{0, 1\}^*$ and a signing key sk on input and outputs a signature σ .

$\text{Vrfy}_{\text{pk}}(m, \sigma)$ is a deterministic polynomial time algorithm that takes a message $m \in \{0, 1\}^*$, a signature σ and a verifying key pk on input, and outputs either 1 if σ is a valid signature on m under pk , and 0 otherwise.

We require Σ to satisfy the following property:

Signature correctness: For every message $m \in \{0, 1\}^*$, we have

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

For brevity, we simply say signature schemes when referring to digital signature schemes.

The main functionality of adaptor signatures allows us to produce an object called a *pre-signature*. We can imagine pre-signatures as commitments to signatures on a message, or, perhaps, as encrypted signatures. Pre-signatures are not valid signatures on their own, however, they can be transformed into one. Anyone who possesses a certain secret value can adapt the pre-signature into a valid signature. Conversely, from a pre-signature and the adapted signature, the corresponding secret value can be extracted. In fact, adaptor signatures enable an exchange between a signature and the secret value that is committed when producing the pre-signature. We present a definition of adaptor signatures as first formalized by Aumayr et al. [4]. Note that in the following, we consider the hard relation R to consist of statement-witness pairs (S, w) .

Definition 6. We define an adaptor signature scheme for a signature scheme $\Sigma = (\text{KeyGen}, \text{Sign}, \text{Vrfy})$ w.r.t. a hard relation R as a tuple of algorithms $\mathbf{a}\Sigma_R = (\Sigma, \text{pSign}, \text{pVrfy}, \text{Adapt}, \text{Ext})$ such that

$\text{pSign}_{\text{sk}}(m, S)$ is a PPT algorithm that takes a message $m \in \{0, 1\}^*$, a statement $S \in \mathcal{L}_R$ and a signing key sk relative to Σ on input, and outputs a pre-signature $\tilde{\sigma}$ on m w.r.t. S under the signing key sk .

$\text{pVrfy}_{\text{pk}}(m, S, \tilde{\sigma})$ is a deterministic polynomial time algorithm that takes a message $m \in \{0, 1\}^*$, a statement $S \in \mathcal{L}_R$, a pre-signature $\tilde{\sigma}$ and a verifying key pk relative to Σ on input, and outputs either 1 if the pre-signature is valid, and 0 otherwise.

$\text{Adapt}_{\text{pk}}(\tilde{\sigma}, w)$ is a deterministic polynomial time algorithm that takes a pre-signature $\tilde{\sigma}$, a witness w of a statement from relation R , and a verifying key pk on input and outputs a signature σ .

$\text{Ext}_{\text{pk}}(\sigma, \tilde{\sigma}, S)$ is a deterministic polynomial time algorithm that takes a signature σ , a pre-signature $\tilde{\sigma}$, a statement $S \in \mathcal{L}_R$, and a verifying key pk on input and outputs either a witness w such that $(S, w) \in R$, or \perp .

We require $\mathbf{a}\Sigma_R$ to satisfy the following property:

Pre-signature correctness: For every message $m \in \{0, 1\}^*$ and every pair $(S, w) \in R$, we have

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

As motivated above, we can imagine a pre-signature as a commitment to a signature on a message. This commitment is produced w.r.t. a statement $S \in \mathcal{L}_R$, and can be verified by the pre-signature verification algorithm pVrfy . Ideally, the valid pre-signature should imply that one can obtain the committed signature by the Adapt algorithm when possessing any witness of S in the relation R . Since R is a hard relation, this witness is, by definition, hard to compute from the statement S when sampled from GenR . We can, thus, see the witness as a secret key used to adapt the pre-signature into a signature. Conversely, one can use the Ext algorithm to extract such a witness from a signature and corresponding pre-signature. Note that each statement $S \in \mathcal{L}_R$ can be used only once, since after adaptation to a signature, the witness is no longer secret. Finally, the pre-signature correctness property is analogous to the standard signature correctness. Specifically, if everything is done correctly, we are guaranteed to obtain a valid pre-signature, signature and witness.

Note that in the rest of the thesis, we call sk also a secret key and pk a public key.

2.2 Adaptor Signatures for Practical Signature Schemes

In this section, we present adaptor signatures based on signature schemes that are widely used in practice, namely the Schnorr signature scheme [11] and ECDSA [12].

Note that from now on, we use additive notation. It is standard when working with groups of elliptic curve points, which we consider mainly in ECDSA, however, they can be also utilized in Schnorr signature. Moreover, for brevity, we omit the

| KeyGen(1^λ) | Sign _{sk} (m) | Vrfy _{pk} (m, σ) |
|--|-------------------------------------|---|
| 1 : $x \leftarrow \mathbb{Z}_q$ | 1 : $(X, x) := \mathbf{sk}$ | 1 : $X := \mathbf{pk}$ |
| 2 : $X := xG$ | 2 : $k \leftarrow \mathbb{Z}_q$ | 2 : $(r, s) := \sigma$ |
| 3 : return $\mathbf{sk} := (X, x), \mathbf{pk} := X$ | 3 : $r := \mathbf{H}(X \ kG \ m)$ | 3 : $r' := \mathbf{H}(X \ sG - rX \ m)$ |
| | 4 : $s := k + rx$ | 4 : return ($r = r'$) |
| | 5 : return (r, s) | |

Figure 2.1 Schnorr signature scheme.

public key \mathbf{pk} from the notation of algorithms $\mathbf{Adapt}_{\mathbf{pk}}$ and $\mathbf{Ext}_{\mathbf{pk}}$, since the adaptor signatures in this section do not use the public key for adapting and extraction.

The presented adaptor signature schemes are defined mainly w.r.t. the DLOG relation in Definition 2. Recall that by the discussion of Definition 2, a cyclic group $\mathbb{G} = \langle G \rangle$ of prime order q with a generator G is sampled by a group generating algorithm \mathbf{GenG} on 1^λ . For such a sampled group, we now consider the DLOG relation of the form

$$\text{DLOG} := \{(Y, y) \in \mathbb{G} \times \mathbb{Z}_q \mid Y = yG\}.$$

Set $pp = \{q, G\}$ to be public parameters of the presented schemes and assume that every algorithm is now parametrized by pp . We also assume that every party involved in the schemes, even the potential adversary, has access to pp and to a hash function \mathbf{H} that depends on the specific construction.

Note that the pre-signature correctness of the following adaptor signature schemes follows from the inspection of the respective algorithms.

2.2.1 Schnorr Adaptor Signatures

Schnorr Signature Scheme. First, we recall the Schnorr signature scheme [11], which consists of three algorithms ($\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Vrfy}$) formally defined in Figure 2.1. \mathbf{KeyGen} samples a uniform element x of \mathbb{Z}_q and outputs a secret key \mathbf{sk} consisting of a tuple (xG, x) , and a public key \mathbf{pk} consisting only of the group element xG . We denote this element by X . The signing algorithm \mathbf{Sign} then on an input message $m \in \{0, 1\}^*$ and a secret key \mathbf{sk} outputs a signature tuple $(r, s) \in \mathbb{Z}_q \times \mathbb{Z}_q$. First, it samples a uniform element k of \mathbb{Z}_q and computes r as an image of the hash $\mathbf{H}: \{0, 1\}^* \rightarrow \mathbb{Z}_q$, specifically $\mathbf{H}(X \| kG \| m)$. To implement this, one would use a cryptographic hash function and take the first $\log(q)$ bits of the resulting output. The element s is then computed as $k + rx$. The verification algorithm \mathbf{Vrfy} on a message $m \in \{0, 1\}^*$, a signature (r, s) and a public key \mathbf{pk} checks if $r = \mathbf{H}(X \| sG - rX \| m)$, under the previous notation.

To provide a broader context for the following construction of the adaptor signature, note that the Schnorr signature scheme can be seen as a non-interactive version of a proof of knowledge of the secret key \mathbf{sk} . Informally, this is a proof system in which a prover wants to convince a verifier of its knowledge of the secret part x of the secret key $\mathbf{sk} = (X, x)$, where x is a discrete logarithm of X w.r.t. G . The interaction would proceed very similarly as in the interactive version of the proof of equality of discrete logarithms in Figure 1.1, only with just t_1 sent

| $\text{pSign}_{\text{sk}}(m, Y)$ | $\text{pVrfy}_{\text{pk}}(m, Y, \tilde{\sigma})$ |
|--|---|
| 1 : $(X, x) := \text{sk}$ | 1 : $X := \text{pk}$ |
| 2 : $k \leftarrow \mathbb{Z}_q$ | 2 : $(r, \tilde{s}) := \tilde{\sigma}$ |
| 3 : $r := \text{H}(X \ kG + Y \ m)$ | 3 : $r' := \text{H}(X \ \tilde{s}G - rX + Y \ m)$ |
| 4 : $\tilde{s} := k + rx$ | 4 : return $(r = r')$ |
| 5 : return (r, \tilde{s}) | |
| $\text{Adapt}(\tilde{\sigma}, y)$ | $\text{Ext}(\sigma, \tilde{\sigma}, Y)$ |
| 1 : $(r, \tilde{s}) := \tilde{\sigma}$ | 1 : $(r, s) := \sigma$ |
| 2 : $s := \tilde{s} + y$ | 2 : $(\tilde{r}, \tilde{s}) := \tilde{\sigma}$ |
| 3 : return (r, s) | 3 : $y' := s - \tilde{s}$ |
| | 4 : if $(Y, y') \in R$ |
| | 5 : then return y' |
| | 6 : else return \perp |

Figure 2.2 Adaptor signature scheme for Schnorr signature w.r.t. DLOG relation [4].

and checked. By the notation of Schnorr’s signing algorithm **Sign**, the prover would send $K := kG$ as generated in **Sign**, the verifier would reply with a uniform element r of \mathbb{Z}_q and the prover would respond with the element s computed as in **Sign**. The verifier would then check if $sG - rX = K$. In the Schnorr signature scheme, this interaction is made non-interactive by the standard Fiat-Shamir transformation using a hash function H . The produced Schnorr signature (r, s) alone is a NIZK proof of the knowledge of x , and the algorithm **Vrfy** is the verifier of the system verifying the proof.

Schnorr Adaptor Signature Scheme. The notion of adaptor signatures for the Schnorr signature scheme was introduced by Poelstra [13, 1]. We describe a version of such an adaptor signature scheme w.r.t. the DLOG relation presented by Aumayr et al. [4]. However, we note that there exists a slightly different version of the adaptor signature for the Schnorr signature scheme by Fournier [3], which is also based on Poelstra’s idea. We choose to present the version by Aumayr et al., since it corresponds more to the standard definition of the Schnorr signature in Figure 2.1. We denote the adaptor signature scheme by $\text{aSchnorr}_{\text{DLOG}}$ and formally define it in Figure 2.2.

Moving from the Schnorr signature scheme to the adaptor signature scheme is straightforward. Consider a statement-witness pair $(Y, y) \in \text{DLOG}$. The main difference is that when **pSign** computes a pre-signature $(r, \tilde{s}) \in \mathbb{Z}_q \times \mathbb{Z}_q$ on m w.r.t. Y , it chooses a uniform randomness $k \leftarrow \mathbb{Z}_q$ as in **Sign**, however, r is computed as $\text{H}(X \| kG + Y \| m)$, i.e., the group element on input of 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} needs to be shifted accordingly by y , i.e., **Adapt** for a pre-signature $(r, \tilde{\sigma})$ and a witness y outputs $(r, \tilde{s} + y)$. Conversely, **Ext** obtains the witness y by subtracting $s - \tilde{s}$.

| KeyGen(1^λ) | Sign _{sk} (m) | Vrfy _{pk} (m, σ) |
|---|--------------------------------------|------------------------------------|
| 1 : $x \leftarrow \mathbb{Z}_q$ | 1 : $(X, x) := \text{sk}$ | 1 : $X := \text{pk}$ |
| 2 : $X := xG$ | 2 : $k \leftarrow \mathbb{Z}_q$ | 2 : $(r, s) := \sigma$ |
| 3 : return $\text{sk} := (X, x)$, $\text{pk} := X$ | 3 : $K := kG$ | 3 : $u := \text{H}(m)s^{-1}$ |
| | 4 : $r := f(K)$ | 4 : $v := rs^{-1}$ |
| | 5 : $s := k^{-1}(\text{H}(m) + rx)$ | 5 : $K' := uG + vX$ |
| | 6 : return $\sigma := (r, s)$ | 6 : return $(f(K') = r)$ |

Figure 2.3 ECDSA signature scheme.

Similarly to the Schnorr signature scheme, we can see this adaptor signature scheme as a NIZK proof of knowledge of the secret part x of the secret key sk . More specifically, take pSign to be a prover and pVrfy a verifier. Again, if we consider the corresponding interactive proof, now the prover sends the commitment kG shifted by Y that is publicly known, and the verifier responds with a random $r \leftarrow \mathbb{Z}_q$. However, the prover sends the final response as if the commitment was not shifted. Thus, to balance this and verify correctly, the verification condition also has to be shifted by Y , as done in the verification algorithm pVrfy . Then, as Schnorr signature, we can see the pre-signature as a NIZK proof of knowledge of discrete logarithm x of X .

2.2.2 ECDSA Adaptor Signatures

ECDSA Signature Scheme. We now recall the ECDSA signature scheme [12]. Note that for this scheme, the group \mathbb{G} is an elliptic curve group over a finite field \mathbb{F}_p . The group elements of \mathbb{G} can be seen as pairs of elements of \mathbb{F}_p , and we can thus talk about the first and second coordinate of the group element. The ECDSA signature scheme consists of a tuple of algorithms ($\text{KeyGen}, \text{Sign}, \text{Vrfy}$) formally defined in Figure 2.3. KeyGen is identical to the key-generation algorithm of the Schnorr signature scheme. The signing algorithm Sign on a message $m \in \{0, 1\}^*$ and secret key sk outputs a signature $(r, s) \in \mathbb{Z}_q \times \mathbb{Z}_q$ similarly to the Schnorr signature scheme. It also starts by sampling an element $k \leftarrow \mathbb{Z}_q$ uniformly randomly and computing a group element kG denoted by K . However, it computes r as $f(K)$, for a function $f: \mathbb{G} \rightarrow \mathbb{Z}_q$, which is typically a projection on the first coordinate of the group element K modulo q . The element s is computed as $k^{-1}(\text{H}(m) + rx)$, where $\text{H}: \{0, 1\}^* \rightarrow \mathbb{Z}_q$ is a hash function. Similarly to the Schnorr signature scheme, to implement this, one would use the first $\log(q)$ bits of the output of a cryptographic hash function. The verification algorithm Vrfy on message $m \in \{0, 1\}^*$, signature (s, r) and public key pk then checks if $r = f(\text{H}(m)s^{-1}G + rs^{-1}X)$.

ECDSA Adaptor Signature Scheme. The ideas of adaptor signature scheme for ECDSA were first presented by Moreno-Sanchez and Kate [2]. We formally define the adaptor signature scheme w.r.t./ DLOG relation in Figure 2.4 as formulated in [4], and we denote it by $\text{aECDSA}_{\text{DLOG}}$.

| $\text{pSign}_{\text{sk}}(m, Y)$ | $\text{pVrfy}_{\text{pk}}(m, Y, \tilde{\sigma})$ |
|--|---|
| 1 : $(X, x) := \text{sk},$ $k \leftarrow \mathbb{Z}_q$ | 1 : $X := \text{pk},$ $(r, \tilde{s}, K, \pi) := \tilde{\sigma}$ |
| 2 : $\tilde{K} := kG, K := kY,$ $r := f(K)$ | 2 : $u := \text{H}(m)\tilde{s}^{-1}$ 3 : $v := r\tilde{s}^{-1}$ |
| 3 : $\tilde{s} := k^{-1}(\text{H}(m) + rx)$ | 4 : $K' := uG + vX$ |
| 4 : $\pi \leftarrow \text{P}_{\text{DLEQ}}((G, Y, \tilde{K}, K), k)$ | 5 : return $((r = f(K))$ $\wedge \text{V}_{\text{DLEQ}}((G, Y, K', K), \pi))$ |
| 5 : return (r, \tilde{s}, K, π) | |
| $\text{Adapt}(\tilde{\sigma}, y)$ | $\text{Ext}(\sigma, \tilde{\sigma}, Y)$ |
| 1 : $(r, \tilde{s}, K, \pi) := \tilde{\sigma}$ | 1 : $(r, s) := \sigma$ |
| 2 : $s := \tilde{s}y^{-1}$ | 2 : $(r, \tilde{s}, K, \pi) := \tilde{\sigma}$ |
| 3 : return (r, s) | 3 : $y' := s^{-1}\tilde{s}$ 4 : if $(Y, y') \in \text{DLOG}$ then return y' else return \perp |

Figure 2.4 Adaptor signature scheme for ECDSA w.r.t. DLOG relation [4].

The final pre-signature of $\text{aECDSA}_{\text{DLOG}}$ consists of four elements $(r, \tilde{s}, K, \pi) \in \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{G} \times \{0, 1\}^*$. Similarly to the adaptor signature based on Schnorr signature, there are two main parts of the pre-signature, r and \tilde{s} , where r is supposed to be used in the corresponding adapted signature (r, s) , while \tilde{s} is meant to be adapted into s , which would make the pair a valid signature. Consider an instance $(Y, y) \in \text{DLOG}$. Again, as in $\text{aSchnorr}_{\text{DLOG}}$, the scheme computes the \tilde{s} part of the pre-signature exactly as it would compute s in the ECDSA Sign algorithm, depending on a randomness k , while the r part is produced with a shifted randomness ky . To obtain the correct element s of the adapted ECDSA signature, we need to adjust the randomness under \tilde{s} accordingly to ky by computing $s := \tilde{s}y^{-1}$. However, having only (r, \tilde{s}) as the pre-signature would not allow us to verify the validity of the pre-signature. Additionally, we need a NIZK proof system $(\text{P}_{\text{DLEQ}}, \text{V}_{\text{DLEQ}})$ for the equality of discrete logarithms, which we discussed in Chapter 1 together with a specific construction in ROM in Figure 1.2. In the pre-signature, the pair (r, \tilde{s}) is extended by adding element $K \in \mathbb{G}$ and a NIZK proof π produced by P_{DLEQ} , which are both specified in the following.

Concretely, to pre-sign a message m w.r.t. a statement Y , we sample a uniform element k of \mathbb{Z}_q and compute the group elements $\tilde{K} := kG$ and $K := kY$. The element r is computed as $f(K)$ and \tilde{s} as $k^{-1}(\text{H}(m) + rx)$. Then, $\text{P}_{\text{DLEQ}}((G, Y, \tilde{K}, K), k)$ produces a NIZK proof π that proves that \tilde{K} and K have the same discrete logarithm k w.r.t. G and Y , respectively. We can verify the pre-signature by computing $K' := \text{H}(m)\tilde{s}^{-1}G + r\tilde{s}^{-1}X$ and checking if $r = f(K)$ and if the proof π holds for the statement (G, Y, K', K) , i.e. if $\text{V}_{\text{DLEQ}}((G, Y, K', K), \pi) = 1$. To extract the witness y from the signature and the pre-signature, we compute $s^{-1}\tilde{s}$ and check if it is, in fact, the valid witness to Y .

| $\text{pSign}_{\text{sk}}(m, I_Y)$ | $\text{pVrfy}_{\text{pk}}(m, I_Y, \tilde{\sigma})$ |
|---|--|
| 1 : $(X, x) := \text{sk},$ $(Y, \pi_Y) := I_Y$ 2 : $k \leftarrow \mathbb{Z}_q$ 3 : $\tilde{K} := kG, K := kY,$ $r := f(K)$ 4 : $\tilde{s} := k^{-1}(\mathbf{H}(m) + rx)$ 5 : $\pi \leftarrow \mathbf{P}_{\text{DLEQ}}((G, Y, \tilde{K}, K), k)$ 6 : return (r, \tilde{s}, K, π) | 1 : $X := \text{pk}, (Y, \pi_Y) := I_Y,$ $(r, \tilde{s}, K, \pi) := \tilde{\sigma}$ 2 : $u := \mathbf{H}(m)\tilde{s}^{-1}$ 3 : $v := r\tilde{s}^{-1}$ 4 : $K' := uG + vX$ 5 : return $(\mathbf{V}_{\text{DLOG}}((G, Y), \pi_Y)$ $\wedge (r = f(K))$ $\wedge \mathbf{V}_{\text{DLEQ}}((G, Y, K', K), \pi))$ |
| $\text{Adapt}(\tilde{\sigma}, y)$ | $\text{Ext}(\sigma, \tilde{\sigma}, I_Y)$ |
| 1 : $(r, \tilde{s}, K, \pi) := \tilde{\sigma}$ 2 : $s := \tilde{s}y^{-1}$ 3 : return (r, s) | 1 : $(r, s) := \sigma$ 2 : $(r, \tilde{s}, K, \pi) := \tilde{\sigma}$ 3 : $y' := s^{-1}\tilde{s}$ 4 : if $(I_Y, y') \in \pi\text{DLOG}$ \quad then return y' \quad else return \perp |

Figure 2.5 Adaptor signature scheme for ECDSA w.r.t. the extended DLOG relation [4].

It is clear that the scheme is defined for $y \neq 0$, since it uses the inverse of y in \mathbb{Z}_q . Thus, in the rest of the thesis, we implicitly assume that GenR for DLOG never samples $y = 0$, and we treat the pair $(0, 0)$ as an invalid DLOG pair, i.e., $(0, 0) \notin \text{DLOG}$ and $Y = yG$ rejects for $y = 0$. 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} equals zero. Moreover, we also consider signatures and pre-signatures with such zero values invalid, and we assume that the algorithms always check and abort before inverting, if necessary.

Note that the simple scheme presented in this section is used in practice. Next, we show a more “robust” scheme suggested by Aumayr et al.

ECDSA Adaptor Signature Scheme with Extended Relation. Aumayr et al. presented in [4] a slightly different adaptor signature scheme for ECDSA. We describe the scheme formally in Figure 2.5. This scheme is analogous to the one depicted in Figure 2.4, however, it is constructed w.r.t. an extended DLOG relation. For this, we consider $(\mathbf{P}_{\text{DLOG}}, \mathbf{V}_{\text{DLOG}})$ to be a NIZK proof of knowledge of a discrete logarithm. As mentioned in the previous section, for $(Y, y) \in \text{DLOG}$, such a system proves knowledge of the discrete logarithm y . Specifically, we consider a relation

$$\pi\text{DLOG} := \{((Y, \pi_Y), y) \mid Y = yG \wedge \mathbf{V}_{\text{DLOG}}((G, Y), \pi_Y) = 1\}.$$

The statement of the relation is now not only a group element Y but also π_Y ; a NIZK proof of knowledge of the discrete logarithm of Y w.r.t. the generator G . We denote a statement of this relation by I_Y , i.e. $(I_Y, y) \in \pi\text{DLOG}$. Note that in this scheme, the relation for the key generation for ECDSA, DLOG; and the relation for the adaptor signature, πDLOG ; now differ. The relation πDLOG causes small changes in the adaptor signature scheme. Mainly, when verifying the pre-signature, we now also use a verifier V_{DLOG} to check if π_Y holds on (G, Y) . We denote this version of the adaptor signature scheme for ECDSA by $\text{aECDSA}_{\pi\text{DLOG}}$.

As we show in the following sections, proving security of the practical adaptor signature scheme for ECDSA in Figure 2.4 is problematic. Using the extended relation πDLOG in the “robust” adaptor signature scheme, Aumayr et al. [4] sidestepped the problems and proved the security of this “robust” scheme.

2.3 Robust Security of Adaptor Signatures

For any cryptographic protocol, it is necessary to analyze its security. Specifically, for the functionality of the adaptor signature schemes, one should consider several different directions of security that heavily depend on the exact application of the adaptor signature. Generally, when defining the security for cryptographic protocols, we want the definitions to protect all parties involved. For adaptor signatures, we talk mainly about a signer of a signature or a pre-signature, and a receiver/verifier of a signature or a pre-signature. In this section, we present the definitions by Aumayr et al. [4] that model the basic directions of security, which the adaptor signature schemes should satisfy.

2.3.1 Unforgeability

Existential Unforgeability for Adaptor Signatures. We begin with security notions that protect the signer. We recall that the standard existential unforgeability of signature schemes aims to protect the signer against a malicious forger. Informally, we want to ensure that only the signer, i.e., the owner of the secret key, can sign a message. Therefore, it should be computationally infeasible to produce a forged signature on any fresh message m with a non-negligible probability without the knowledge of a signing key. The definition of existential unforgeability states that this should be infeasible even when the PPT adversary is provided with access to a signing oracle and can ask for signatures on any chosen message. Since the adaptor signatures are an extension of signature schemes and operate in a similar context, it is desirable to require similar security. In a space where adaptor signatures are used, the adversary could potentially obtain an advantage from existing pre-signatures. Informally, we want the pre-signatures not to help the adversary to forge a signature on any message. Analogously to the signing oracle, it is reasonable to allow the adversary to have access to a pre-signing oracle, obtaining pre-signatures on messages of its choice. However, it is a question whether the choice of a statement S , with respect to which is the pre-signature created, should be unrestricted as well. Aumayr et al. [4] defined an *existential unforgeability under chosen message attack for adaptor signature* for an unrestricted pre-signing oracle, where the adversary can ask for pre-signatures w.r.t. any statement S , even for those where the adversary does not know the

| $\text{aRSigForge}_{\mathcal{A}, \text{a}\Sigma_R}(\lambda)$ | $\mathcal{O}_S(m)$ | $\mathcal{O}_{pS}(m, S)$ |
|---|---|---|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ | 1: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(S^*, w^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ | 3: return $\tilde{\sigma}$ |
| 4: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, S^*)$ | | |
| 5: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m^*, S^*)$ | | |
| 6: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | | |
| 7: return $(m^* \notin \mathcal{Q}$ $\wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | | |

Figure 2.6 $\text{aRSigForge}_{\mathcal{A}, \text{a}\Sigma_R}$ experiment [4].

corresponding witness w . We call the security definitions with such unrestricted access to the pre-signing oracle *robust*. We present the definition by Aumayr et al. in Definition 7.

Definition 7 (Robust existential unforgeability [4]). *We say that an adaptor signature scheme $\text{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R satisfies robust existential unforgeability under chosen message attack for adaptor signature (aREUF-CMA security) if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function negl such that*

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

for the experiment $\text{aRSigForge}_{\mathcal{A}, \text{a}\Sigma_R}$ defined in Figure 2.6.

The goal of the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the $\text{aRSigForge}_{\mathcal{A}, \text{a}\Sigma_R}$ experiment in Figure 2.6 is to forge a signature on a message $m^* \in \{0, 1\}^*$ of its choice, that would be valid under a given public key pk sampled by KeyGen . The adversary does not know the corresponding secret key sk . As an advantage during the experiment, \mathcal{A} can query the signing and pre-signing oracles $\mathcal{O}_S, \mathcal{O}_{pS}$ and obtain signatures and pre-signatures on any message m and, in the case of \mathcal{O}_{pS} , w.r.t. any statement $S \in \mathcal{L}_R$. First, \mathcal{A}_1 receives pk and a statement $S^* \in \mathcal{L}_R$ from an instance $(S^*, w^*) \in R$, which is sampled by GenR . On this input, \mathcal{A}_1 chooses m^* and produces a state st . Then, \mathcal{A}_2 receives st and a pre-signature $\tilde{\sigma}$ on m^* w.r.t. S^* pre-signed by pSign_{sk} . Finally, \mathcal{A}_2 outputs a signature σ^* . The adversary wins the experiment if σ^* is not only a valid signature on m^* , but also if \mathcal{A} did not query \mathcal{O}_S or \mathcal{O}_{pS} for any signature or pre-signature on m^* . This means that if the adversary wants to win the experiment, $\tilde{\sigma}$ is the only pre-signature on m^* that \mathcal{A} gets. Note that \mathcal{A} neither knows the witness w^* of S^* , nor chooses the statement S^* .

Pre-Signature Unforgeability. When considering the security towards the signer, we also need to discuss the unforgeability of a pre-signature. Since a valid pre-signature can be seen as a commitment to a signature and can lead to a valid signature, we want to ensure that it is infeasible to produce the pre-signature without possessing the corresponding secret key. In other words, we require for

| $\text{pRSigForge}_{\mathcal{A}, \mathfrak{a}\Sigma_R}(\lambda)$ | $\mathcal{O}_S(m)$ | $\mathcal{O}_{pS}(m, S)$ |
|--|--|--|
| 1 : $\mathcal{Q} := \emptyset$ | 1 : $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ | 1 : $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S)$ |
| 2 : $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ | 2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3 : $(S^*, w^*) \leftarrow \text{GenR}(1^\lambda)$ | 3 : return σ | 3 : return $\tilde{\sigma}$ |
| 4 : $(m^*, \tilde{\sigma}) \leftarrow \mathcal{A}^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, S^*)$ | | |
| 5 : return $(m^* \notin \mathcal{Q} \wedge \text{pVrfy}_{\text{pk}}(m^*, S^*, \tilde{\sigma}))$ | | |

Figure 2.7 $\text{pRSigForge}_{\mathcal{A}, \mathfrak{a}\Sigma_R}$ experiment [4].

any PPT adversary to be unable to produce a forged valid pre-signature on a fresh message with a non-negligible probability while given the access to the signing and pre-signing oracles. We have described the potential limits of the access to the pre-signing oracle above. Aumayr et al. defined *pre-signature unforgeability under chosen message attack* with the pre-signing oracle unrestricted as in the definition of robust existential unforgeability. We present their robust definition of pre-signature unforgeability (pREUF-CMA security) in Definition 8.

Definition 8 (Robust pre-signature unforgeability [4]). *We say that an adaptor signature scheme $\mathfrak{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R satisfies robust pre-signature unforgeability under chosen message attack (pREUF-CMA security) if for every PPT adversary \mathcal{A} there exists a negligible function negl such that*

$$\Pr[\text{pRSigForge}_{\mathcal{A}, \mathfrak{a}\Sigma_R}(\lambda) = 1] \leq \text{negl}(\lambda),$$

for the experiment $\text{pRSigForge}_{\mathcal{A}, \mathfrak{a}\Sigma_R}$ defined in Figure 2.7.

In the $\text{pRSigForge}_{\mathcal{A}, \mathfrak{a}\Sigma_R}$ experiment in Figure 2.7, the adversary \mathcal{A} needs to forge a pre-signature on a message m^* of its choice. This pre-signature should be valid under a public key pk sampled by KeyGen and w.r.t. a statement $S^* \in \mathcal{L}_R$ sampled in (S^*, w^*) by GenR . Both pk and S^* are given to \mathcal{A} on input, however, the adversary does not know the corresponding secret key or the witness to S^* . During the experiment, it can query \mathcal{O}_S and \mathcal{O}_{pS} for signatures and pre-signatures on any message and w.r.t. any statement $S \in \mathcal{L}_R$ as in the $\text{aRSigForge}_{\mathcal{A}, \mathfrak{a}\Sigma_R}$ experiment. To win, the adversary cannot ask \mathcal{O}_S or \mathcal{O}_{pS} for signatures or pre-signatures on m^* .

2.3.2 Witness Extractability

Recall the toy scenario used as a motivation in the introduction to the thesis, where Alice and Bob propose a bet on Bob's marathon time. Specifically, Alice will give Bob ten Friendcoins if he completes the marathon in 3.5 hours, five Friendcoins in 4 hours, and one Friendcoin in 5 hours. Bob can receive the money only by publishing a transaction signed by Alice. They now appoint Olivia to be the supervisor of the bet. Olivia publishes statements $S_1, S_2, S_3 \in \mathcal{L}_R$ sampled by GenR , each corresponding to one of the outcomes of the event, and, on the day of the marathon, she will publish a witness w_i of S_i , $(S_i, w_i) \in R$, related to the actual outcome that occurs. First, Alice produces three pre-signatures on

| $\mathbf{aRWitExt}_{\mathcal{A}, \mathbf{a}\Sigma_R}(\lambda)$ | $\mathcal{O}_S(m)$ |
|--|--|
| 1 : $\mathcal{Q} := \emptyset$ | 1 : $\sigma \leftarrow \mathbf{Sign}_{\mathbf{sk}}(m)$ |
| 2 : $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathbf{KeyGen}(1^\lambda)$ | 2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3 : $(m^*, S^*, \mathbf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\mathbf{pk})$ | 3 : return σ |
| 4 : $\tilde{\sigma} \leftarrow \mathbf{pSign}_{\mathbf{sk}}(m^*, S^*)$ | <hr/> $\mathcal{O}_{pS}(m, S)$ |
| 5 : $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \mathbf{st})$ | 1 : $\tilde{\sigma} \leftarrow \mathbf{pSign}_{\mathbf{sk}}(m, S)$ |
| 6 : return $((S^*, \mathbf{Ext}_{\mathbf{pk}}(\sigma, \tilde{\sigma}, S^*)) \notin R$ $\wedge m^* \notin \mathcal{Q} \wedge \mathbf{Vrfy}_{\mathbf{pk}}(m^*, \sigma^*))$ | 2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 3 : return $\tilde{\sigma}$ |

Figure 2.8 $\mathbf{aRWitExt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment [4].

transactions that would send ten, five, or one Friendcoin to Bob w.r.t. S_1, S_2, S_3 , respectively. She sends the pre-signatures to Bob, and he verifies them. After the marathon results, Olivia publishes w_i . Then, Bob will be able to use w_i to adapt the corresponding pre-signature into a signature, and claim his prize. Bob's final time is 4.5 hours. Olivia therefore publishes the witness w_3 and Bob can adapt the correct pre-signature into a signature, earning him one Friendcoin. However, Alice later finds one more transaction, which is signed by her, published, sending Bob ten Friendcoins. It turns out that Olivia colluded with Bob and provided him in secret also with w_1 ; a witness of the statement related to the outcome that did not happen. Alice needs to be able to extract the valid witness w_1 from the produced pre-signature and the published signature, convicting Olivia of the collusion. Informally, to protect the signer, we thus require for any PPT adversary to be unable to produce with a non-negligible probability a valid forged signature, from which it would not be possible to extract a valid witness of the corresponding statement. Formally, we present the security definition by Aumayr et al. [4] in Definition 9.

Definition 9 (Robust witness extractability [4]). *We say that an adaptor signature scheme $\mathbf{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R is robustly witness extractable if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function negl such that*

$$\Pr[\mathbf{aRWitExt}_{\mathcal{A}, \mathbf{a}\Sigma_R}(\lambda) = 1] \leq \mathit{negl}(\lambda)$$

for the experiment $\mathbf{aRWitExt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ defined in Figure 2.8.

The $\mathbf{aRWitExt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment in Figure 2.8 is similar to the $\mathbf{aRSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment. The PPT adversary \mathcal{A} again aims to forge a fresh signature σ^* on a message m^* chosen by \mathcal{A} , which would be valid under a given public key \mathbf{pk} sampled by \mathbf{KeyGen} . The adversary also has access to the signing and pre-signing oracles \mathcal{O}_S and \mathcal{O}_{pS} . Compared to $\mathbf{aRSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$, \mathcal{A} now chooses the statement S^* , w.r.t. which it obtains a pre-signature $\tilde{\sigma}$ on m^* , while previously, S^* was sampled together with its witness by \mathbf{GenR} . Moreover, the goal of the adversary is not only for the forgery σ^* to be valid, but also to be unable to extract a valid witness to S^* from the forged σ^* , the pre-signature $\tilde{\sigma}$ and the statement S^* . This

is formalized in the requirement of

$$(S^*, \text{Ext}_{\text{pk}}(\sigma^*, \tilde{\sigma}, S^*)) \notin R,$$

which is added to the winning conditions.

2.3.3 Pre-Signature Adaptability

Now, we turn to the security towards the verifier/receiver. For a pre-signature to act as a commitment to a signature on a message, we need the following property. Assume that a pre-signature $\tilde{\sigma}$ is valid under a public key pk and w.r.t. a statement S . The verifier possessing the witness to the statement S needs to be able to adapt $\tilde{\sigma}$ into a valid signature. Note that this property considers any valid pre-signatures, not only those produced by pSign . In this way, it is stronger than pre-signature correctness. We formally define such a property by Aumayr et al. as follows.

Definition 10 (Perfect pre-signature adaptability [4]). *We say that an adaptor signature scheme $\text{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R is perfectly pre-signature adaptable if for any message $m \in \{0, 1\}^*$, any instance $(S, w) \in R$, any public key pk and any valid pre-signature $\tilde{\sigma} \in \{0, 1\}^*$, $\text{pVrfy}_{\text{pk}}(m, S, \tilde{\sigma}) = 1$, we have*

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

2.3.4 Security of Adaptor Signatures for Practical Signature Schemes

The state of the art technique for proving both robust existential unforgeability and robust witness extractability of an adaptor signature scheme as defined in the previous sections is via a reduction to the strong existential unforgeability of the underlying signature scheme. We now recall a formal definition of the strong existential unforgeability property.

Definition 11 (Strong unforgeability). *Consider a signature scheme Σ . We say that Σ satisfies strong existential unforgeability under a chosen message attack (SUF-CMA security) if for every PPT adversary \mathcal{A} there exists a negligible function negl such that*

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

for the experiment $\text{strongSigForge}_{\mathcal{A}, \Sigma}$ defined in Figure 2.9.

In the $\text{strongSigForge}_{\mathcal{A}, \Sigma}$ experiment in Figure 2.9, the PPT adversary \mathcal{A} can interact with a signing oracle \mathcal{O}_S and can obtain signatures on messages of its choice. Informally, \mathcal{A} wants to produce a message m^* and a valid signature σ^* on m^* such that if the adversary previously queried \mathcal{O}_S on this message, the oracle did not answer with σ^* . The signature σ^* should be valid under a public key pk sampled by KeyGen and provided to \mathcal{A} on input. Notice that strong unforgeability implies weaker *existential unforgeability*, which we discussed earlier and where the adversary wins if it manages to produce a signature valid under pk on any message other than those that \mathcal{O}_S was queried on.

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

Figure 2.9 $\text{strongSigForge}_{\mathcal{A},\Sigma}$ experiment.

Security of Schnorr Adaptor Signatures. For the Schnorr adaptor signature scheme in Figure 2.2, the basic security as defined in the previous sections is resolved. In random oracle model, Aumayr et al. [4] reduced both robust existential unforgeability and robust witness extractability to the strong unforgeability of the Schnorr signature scheme. The reductions rely on the programmability of the random oracle, which is possible because of the structure of the scheme. Aumayr et al. also proved the perfect pre-signature adaptability of Schnorr adaptor signature as defined in Definition 10. They resolved the robust pre-signature unforgeability in general and showed that any adaptor signature scheme that satisfies robust existential unforgeability and perfect pre-signature adaptability then also satisfies robust pre-signature unforgeability. Gerhart et al. [14] defined even stronger definitions of security and proved that a version of the Schnorr adaptor signature satisfies these new definitions.

Security of ECDSA Adaptor Signatures. Similarly to the adaptor signature for the Schnorr signature scheme, one would want to apply analogous reductions of the robust existential unforgeability and robust witness extractability to strong unforgeability of the signature scheme. However, the ECDSA signature scheme is not strongly unforgeable. Recall now the following property of ECDSA signatures.

Claim 5. *Assume (r, s) is a valid ECDSA signature on a message m under a public key pk . Then, $(r, -s)$ is also a valid signature on m under pk .*

Proof. This follows immediately from the properties of elliptic curves, since

$$\begin{aligned} f(\mathbf{H}(m)(-s)^{-1}G + r(-s)^{-1}X) &= f(-(\mathbf{H}(m)s^{-1}G + rs^{-1}X)) \\ &= f(\mathbf{H}(m)s^{-1}G + rs^{-1}X) \\ &= r, \end{aligned}$$

where the last equality is given by the validity of the signature (r, s) . \square

For the security of the signature scheme and, thus, also for the adaptor signature scheme, this is not a desired property. However, we can handle this by adding a constraint to the verification of a signature and considering a (usual ECDSA valid) signature (r, s) to be valid only if

$$0 \leq s \leq (q-1)/2.$$

This restricted version of ECDSA is called *positive ECDSA* [4, 15, 16] and we denote it by ECDSA^+ . We will call s from the described interval *positive*. Beyond

the additional verification constraint, when considering ECDSA^+ compared to the standard ECDSA signature scheme in Figure 2.3, there is a change in the **Sign** algorithm. At the end of the signing algorithm of ECDSA^+ , after producing the standard ECDSA signature (r, s) , we check the positivity of s and change it to $-s$ if necessary for the signature to be valid.

Similarly, $\text{aECDSA}_{\text{DLOG}}^+$ is analogous to $\text{aECDSA}_{\text{DLOG}}$ in Figure 2.4 with the following changes.

Adapt: Before returning the adapted signature (r, s) , we again change s to $-s$, if it is necessary for the signature to be valid.

Ext: At the end of the algorithm, we now need to check if $(Y, y') \in \text{DLOG}$ or $(Y, -y') \in \text{DLOG}$, in which case we return y' or $-y'$, respectively. Otherwise, we return \perp as in $\text{aECDSA}_{\text{DLOG}}$.

The last change comes from the fact that the pre-signing algorithm remains unchanged, while the adapting algorithm now deals with the positivity of s . If (r, \tilde{s}, K, π) is a valid pre-signature w.r.t. $Y \in \mathcal{L}_{\text{DLOG}}$ and (r, s) is an ECDSA^+ signature adapted from the pre-signature, where the second coordinate was flipped, we get

$$s^{-1}\tilde{s}G = -Y.$$

Therefore, the additional check is needed.

It is assumed that ECDSA^+ is strongly unforgeable. Thus, in the rest of the thesis, we utilize the corresponding adaptor signature scheme $\text{aECDSA}_{\text{DLOG}}^+$. Since its construction is very similar to $\text{aECDSA}_{\text{DLOG}}$, for brevity, when talking about ECDSA^+ and $\text{aECDSA}_{\text{DLOG}}^+$, we will still refer to the algorithms in the original figures depicting ECDSA and $\text{aECDSA}_{\text{DLOG}}$, namely Figure 2.3 and Figure 2.4.

Simply considering ECDSA^+ , however, does not straightforwardly resolve the security of the corresponding adaptor signature scheme. In order to reduce the robust existential unforgeability or robust witness extractability of the adaptor signature to the strong unforgeability of ECDSA^+ , the reduction has to simulate the pre-signatures answered by the pre-signing oracle when queried by the adversary. By robustness of the definitions, the reduction needs to be able to simulate pre-signatures w.r.t. any statements Y chosen by the adversary. As we show in the following sections, this can be done given access to the signing oracle *if the witness y to the statement Y is known to the reduction*. However, it is not clear how to simulate the pre-signatures without the knowledge of both the secret key sk and the witness y . Aumayr et al. [4] sidestepped this problem by modifying the adaptor signature scheme, producing $\text{aECDSA}_{\pi\text{DLOG}}^+$ depicted in Figure 2.5. They constructed the scheme w.r.t. a relation πDLOG and assumed a NIZK proof of knowledge of discrete logarithm $(\text{P}_{\text{DLOG}}, \text{V}_{\text{DLOG}})$. Recall that this proof system produces a proof π_Y that proves the knowledge of a witness y for a statement $Y \in \mathcal{L}_{\text{DLOG}}$. Moreover, Aumayr et al. required the proof system to satisfy *online extractability*, which is a property that allows the reduction to extract y from π_Y in ROM. Thus, since now the reduction knows the corresponding witness, it can simulate a pre-signature w.r.t. arbitrary Y . The online extractability property was introduced by Fischlin [17], where a construction of such a proof system was presented. Informally, the proposed construction of NIZK proofs of knowledge

with online extractors relies on multiple repetitions of the proof, which would make the adaptor signature scheme less practical and the π DLOG statements $I_Y = (Y, \pi_Y)$ alone larger. Moreover, the standard DLOG is a relation that is generally used in the applications of adaptor signatures for ECDSA. Leveraging the described modifications, Aumayr et al. then proved the robust existential unforgeability and the robust witness extractability of $\mathbf{aECDSA}_{\pi\text{DLOG}}^+$. Since they considered adaptor signatures primarily in the context of payment channels, Aumayr et al. were interested in the robust properties. However, their theorems and proofs do not argue any security of $\mathbf{aECDSA}_{\text{DLOG}}^+$.

2.4 Limits of Robust Security of Adaptor Signatures

In this section, we show the limits of robust security definitions when considering $\mathbf{aECDSA}_{\text{DLOG}}^+$, specifically in practice. We consider $\mathbf{aECDSA}_{\text{DLOG}}^+$ deployed on the Bitcoin blockchain, which is one of the primary applications of the scheme, as the adaptor signatures are mostly relevant to cryptocurrencies. We outline a potential weakness of robust security definitions in this setting. First, let us define the Diffie-Hellman problem and the fixed Diffie-Hellman oracle.

Definition 12. *For a cyclic group \mathbb{G} of prime order q with a generator G , we define the Diffie-Hellman problem as follows: Given group elements aG, bG for uniformly random $a, b \leftarrow \mathbb{Z}_q$, return abG .*

Definition 13. *Consider a cyclic group \mathbb{G} of prime order q with a generator G and a group element $aG = P \in \mathbb{G}$. We define a fixed Diffie-Hellman oracle for P as a function FDH_P that on input $bG \in \mathbb{G}$ returns abG .*

Fixed Diffie-Hellman Oracle from Pre-Signing Oracle. As Fournier [3] noticed, when given unrestricted access to the pre-signing oracle \mathcal{O}_{pS} (as defined in robust security experiments in Figure 2.6 or Figure 2.8) while proving the security of $\mathbf{aECDSA}_{\text{DLOG}}$, a fixed instance of the Diffie-Hellman problem becomes easy. Specifically also for $\mathbf{aECDSA}_{\text{DLOG}}^+$, $\mathbf{pSign}_{sk}(m, Y)$ returns a tuple $\tilde{\sigma} = (r, \tilde{s}, K, \pi)$ such that

$$\tilde{s} = k^{-1}(\mathbf{H}(m) + rx) \quad \text{and} \quad K = kY$$

for some randomly chosen $k \leftarrow \mathbb{Z}_q$. Therefore, we have the following

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

We can see that with access to \mathcal{O}_{pS} (and thus to \mathbf{pSign}_{sk}), we obtain an access to a fixed Diffie-Hellman oracle.

| | |
|------------------|--|
| 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 | $= 11579208923731619542357098500868790785283756$ $4279074904382605163141518161494337$ a prime |
| $\log_2(q)$ | ≈ 256 |

Table 2.1 Elliptic curve secp256k1.

Observation 1. Given $\mathcal{O}_{pS}(\cdot)$ for $\text{aECDSA}_{\text{DLOG}}^+$, we can implement a call to the FDH_{pk} oracle fixed for a public key pk in

- 1 call to $\mathcal{O}_{pS}(\cdot)$
- 1 hash $\text{H}(\cdot)$
- 1 subtraction in \mathbb{Z}_q
- 1 inverse in \mathbb{Z}_q
- 2 multiplications in \mathbb{Z}_q
- 1 group operation in \mathbb{G}
- 2 scalar multiplications in \mathbb{G} .

For any $Y \in \mathbb{G}$, we have $\text{FDH}_{\text{pk}}(Y)$ simply by calling $\mathcal{O}_{pS}(m, Y)$ for any message m and transforming the output accordingly. We notice that this would lead to a security hazard, specifically, when the adaptor signature scheme $\text{aECDSA}_{\text{DLOG}}^+$ is intended to be used securely on Bitcoin.

Attack on secp256k1 Using Pre-Signing Oracle. For this section, let us consider \mathbb{G} to be the secp256k1 elliptic curve group used in Bitcoin public-key cryptography. According to the specifications of secp256k1 (see Table 2.1 or [18]), \mathbb{G} is cyclic with a generator G , of odd characteristic and of prime order q such that $\log_2(q) \approx 256$. For now, let us also consider the adaptor signature scheme $\text{aECDSA}_{\text{DLOG}}^+$ to be initialized over secp256k1 for both the DLOG relation and the ECDSA^+ signature scheme. By the above, when considering robust security experiments $\text{aRSigForge}_{\mathcal{A}, \text{a}\Sigma_R}$ and $\text{aRWitExt}_{\mathcal{A}, \text{a}\Sigma_R}$ with unrestricted access to the pre-signing oracle, the adversary \mathcal{A} in fact obtains a fixed Diffie-Hellman oracle FDH_{pk} for the fixed underlying public key pk .

We can take advantage of this and mount a type of meet-in-the-middle attack by Brown and Gallant [19]. This attack uses an available fixed Diffie-Hellman oracle FDH_P to compute a discrete logarithm of a group element $P \in \mathbb{G}$ in complexity depending on a factorization of $q - 1$, where \mathbb{G} is a cyclic group of prime order q . In our case, we would be able to compute a discrete logarithm of $X := \text{pk}$, i.e. a secret part x of sk such that $X = xG$. If done efficiently enough, it would break the security of the underlying ECDSA^+ signature scheme, as it would be possible to compute the corresponding secret key. Let us have $q = uv + 1$ for some positive integers u, 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-Steps Giant-Steps algorithm [20]. Specifically, we utilize access to the fixed Diffie-Hellman oracle to compute the group element $x^u G$, which would not

| 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 |
| 341948486974166000522343609283189 | 108.1 |

Table 2.2 Factorization of $q - 1$ for secp256k1.

be possible without the oracle, as x is secret. By [19], the time complexity of the attack is dominated by roughly

u calls to the FDH_{pk} oracle and $2(\sqrt{u} + \sqrt{v})$ 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 FDH_{pk} oracle as in Observation 1, the time complexity would be

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

where we omit the call to the pre-signing oracle \mathcal{O}_{ps} , as its unit cost is negligible in comparison to the cost of the other operations. Moreover, if we fix a message m , the transformation of the pre-signatures sampled by $\text{pSign}_{\text{sk}}(m, Y)$ to the answers of $\text{FDH}_{\text{pk}}(Y)$ always computes the same value $-\text{H}(m)$, leaving us with just one hash function evaluation and one subtraction in \mathbb{Z}_q that is needed for all FDH_{pk} calls. In comparison to the number of other operations, this can be also omitted in the following analysis, resulting in the complexity of

u inverse in \mathbb{Z}_q
 $2u$ multiplications in \mathbb{Z}_q
 u group operations in \mathbb{G}
 $(2u + 2\sqrt{u} + 2\sqrt{v})$ scalar multiplications in \mathbb{G} .

Moreover, when using a precomputation described in [19], 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} , where each 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 be thus upper bounded by 2^{23} group operations in \mathbb{G} .

A factorization of $q - 1$ for secp256k1 is given in Table 2.2. Consider u to be a product of the emboldened factors. Then $83 < \log_2(u) < 84$ and thus $u \approx \sqrt[3]{q}$, since $\log_2(\sqrt[3]{q}) \approx 85$. Let us now compute the rough time complexity of the attack

for such u as

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

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 lower bound for the space complexity is at least 2^{84} bits, which is vastly unrealistic. However, even though the attack is not feasible in practice, the rough time complexity of 2^{93} group operations, when given access to the fixed Diffie-Hellman oracle, shows a great weakening of the expected $\log_2(q)/2 \approx 128$ -bit security of the secp256k1 elliptic curve, i.e. computing the discrete logarithm for a randomly chosen element of the group with much less than 2^{128} group operations, which would be expected without access to the oracle. We formalize the results in the following statement.

Theorem 6. *Let \mathbb{G} be the secp256k1 elliptic curve group as specified in Table 2.1, $(\mathbf{sk}, \mathbf{pk})$ be an ECDSA⁺ key pair, and \mathcal{O}_{pS} be a pre-signing oracle for aECDSA⁺_{DLOG} such that, on input $m \in \{0, 1\}^*$ and $Y \in \mathbb{G}$, it outputs a pre-signature on m w.r.t. Y sampled by $\mathbf{pSign}_{\mathbf{sk}}$. The secret key \mathbf{sk} can then be computed in time dominated by 2^{93} group operations in \mathbb{G} and in space lower bounded by 2^{84} bits.*

Moreover, as suggested by Brown and Gallant [19], the space complexity could be reduced by using Pollard- ρ algorithm [21] instead of the Baby-Steps Giant-Steps algorithm. However, when trying to apply the ideas of the algorithm in [19] to the algorithm in [21], it was not clear to us how to compute the following group element efficiently enough so that the time complexity of the original attack would be preserved. Specifically, it was not clear, given $(x^u)^{a_i}G$ and access to $\text{FDH}_{\mathbf{pk}}$, how to efficiently compute $(x^u)^{2a_i}G$ for the unknown secret key x and an arbitrary a_i . Since it is beyond the scope of the thesis, we leave this open.

3 Security of Practical Adaptor Signatures for ECDSA

In this chapter, we present new definitions of security for adaptor signature schemes. The definitions are based on the robust security definitions presented in the previous sections, however, they are formalized to fit the context of the adaptor signature scheme $\mathbf{aECDSA}_{\text{DLOG}}^+$ described in Figure 2.4. Specifically, they do not permit the theoretical attack in Section 2.4. We formally prove that, under the necessary assumptions, $\mathbf{aECDSA}_{\text{DLOG}}^+$ satisfies the new security definitions.

3.1 Unforgeability of Adaptor Signatures for ECDSA

3.1.1 Definition of Existential Unforgeability

Motivated by the discussion in the previous sections, we present a new definition of *existential unforgeability under chosen message attack for adaptor signatures*. Our definition follows the robust existential unforgeability for adaptor signatures (Definition 7) but restricts adversary’s access to the pre-signing oracle. Previously, the adversary could query the oracle w.r.t. any chosen statement $S \in \mathcal{L}_R$. However, this definition was too robust in the setting of $\mathbf{aECDSA}_{\text{DLOG}}^+$. We modify the pre-signing oracle to answer two types of queries - either the queries w.r.t. the fixed statement S^* generated in the main part of the security experiment, or the queries w.r.t. any statement S for which the adversary provides a witness. Therefore, the adversary is not allowed to ask for a pre-signature w.r.t. the statements for which it does not know the witness. We formalize this notion in the following definition.

Definition 14. *We say that an adaptor signature scheme $\mathbf{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R is $\mathbf{aEUFCMA}$ secure if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function negl such that*

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

for the experiment $\mathbf{aSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ defined in Figure 3.1.

The $\mathbf{aSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment in Figure 3.1 differs from the $\mathbf{aRSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment only in the pre-signing oracle \mathcal{O}_{pS} . Otherwise, they proceed identically. Again, for a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, a key pair (sk, pk) and an instance $(S^*, w^*) \in R$ are generated by KeyGen and GenR , respectively, and \mathcal{A}_1 obtains pk and S^* on input. The objective of the adversary is still to forge a fresh signature on a message of its choice. The adversary \mathcal{A}_1 chooses the message $m^* \in \{0, 1\}^*$ and produces a state st , which is forwarded to \mathcal{A}_2 together with a pre-signature $\tilde{\sigma}$ on m^* w.r.t. S^* sampled by $\mathbf{pSign}_{\text{sk}}$. During the experiment, \mathcal{A} can query a signing oracle, which works as in the $\mathbf{aRSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment, and a pre-signing oracle \mathcal{O}_{pS} . The input arguments of the pre-signing oracle in the $\mathbf{aSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment now consist of a message $m \in \{0, 1\}^*$, a bit b and, if applicable, an statement-witness pair (S, w) of the relation R . The adversary

| $\text{aSigForge}_{\mathcal{A}, \text{a}\Sigma_R}(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|---|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(S^*, w^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, S^*)$ | $\mathcal{O}_{pS}(m, b, (S, w))$ |
| 5: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m^*, S^*)$ | 1: if $b = 0$ then |
| 6: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 7: return $(m^* \notin \mathcal{Q}$ | 3: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S^*)$ |
| $\wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge (S, w) \in R$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S)$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.1 $\text{aSigForge}_{\mathcal{A}, \text{a}\Sigma_R}$ experiment.

can ask for pre-signatures w.r.t. the fixed statement S^* by setting $b := 0$ and calling \mathcal{O}_{pS} on m and the bit b . By setting $b := 1$, the adversary can ask \mathcal{O}_{pS} to pre-sign a message w.r.t. any statement S , for which the adversary provides on input a valid statement-witness pair. More specifically, \mathcal{O}_{pS} answers with a pre-signature generated by $\text{pSign}_{\text{sk}}(m, S)$, when queried on a message m , bit $b = 1$ and a statement-witness pair $(S, w) \in R$. If the pair (S, w) , given by the adversary, is not a statement-witness pair of R , i.e., $(S, w) \notin R$, the pre-signing oracle returns \perp . Finally, \mathcal{A}_2 outputs a signature σ^* and wins the experiment, if σ^* is a valid signature on m^* under pk and if the adversary did not query the oracles for signatures or pre-signatures on the message m^* .

3.1.2 Proof of Existential Unforgeability

We now prove the defined existential unforgeability for $\text{aECDSA}_{\text{DLOG}}^+$. 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 the robust existential unforgeability of $\text{aECDSA}_{\pi\text{DLOG}}^+$ presented in [4]. However, we stress that we prove our notion of security for the original adaptor signature based on ECDSA^+ without modifying the scheme. We also do not need to utilize a NIZK proof of knowledge and its extraction algorithm in the reduction.

Recall that the construction of $\text{aECDSA}_{\text{DLOG}}^+$ depends on a NIZK proof system for equality of discrete logarithms. In Definition 3, we defined NIZK proof systems in the random oracle model, and thus, to provide a random oracle to the proof system, we would want to show the reduction in the same model. We can formalize the security in two ways. In the first one, we assume the existence of the NIZK proof system for equality of discrete logarithms. In the other one, we do not assume the existence of such a system, and we are still able to show the reduction

| $\mathcal{S}_{pS}^{\text{Sign}}(m, (Y, y))$ | |
|---|--|
| 1 : | $\sigma \leftarrow \text{Sign}(m)$ |
| 2 : | $\mathbf{parse}(r^*, s^*) := \sigma$ |
| 3 : | $b \leftarrow \{0, 1\}$ |
| 4 : | $s := (-1)^b s^*$ |
| 5 : | $\tilde{s} := sy$ |
| 6 : | $u := \mathbf{H}(m)s^{-1}$ |
| 7 : | $v := r^*s^{-1}$ |
| 8 : | $K := uG + vX$ |
| 9 : | $\tilde{K} := y^{-1}K$ |
| 10 : | $\pi_S \leftarrow S_{\text{DLEQ}}(G, Y, \tilde{K}, K)$ |
| 11 : | $\mathbf{return}(r^*, \tilde{s}, K, \pi_S)$ |

Figure 3.2 Simulator of pre-signatures \mathcal{S}_{pS} .

of the existential unforgeability in the random oracle model, since, as shown in Chapter 1, in ROM, we have a specific construction of the NIZK proof system for the equality of discrete logarithms. We show the formal proof for the first approach and formalize the second approach as a corollary, as the reduction is analogous. In what follows, in order to convey the main idea of the proofs, we assume a NIZK proof system for equality of discrete logarithms in the standard model. Such a system has the same properties as in Definition 3, however, the algorithms are not oracle-aided, and the corresponding simulator outputs only a simulated proof and omits the oracle programming. This allows us to operate with the assumed NIZK proof system modularly as with a black box. We use this approach in the following sections.

First, we prove the following lemma, which states that the algorithm presented in Figure 3.2 simulates pre-signatures for $\mathbf{aECDSA}_{\text{DLOG}}^+$ when given black-box access to the corresponding signing algorithm for a fixed secret key. We utilize this property in the upcoming reduction.

Lemma 7. *Let $\mathbf{aECDSA}_{\text{DLOG}}^+$ described in Figure 2.4 be an adaptor signature scheme for ECDSA^+ w.r.t. the DLOG relation, $\mathcal{S}_{pS}^{\text{Sign}}$ described in Figure 3.2 be a PPT simulator of pre-signatures, and let there exist a NIZK proof for equality of discrete logarithms with simulator $\mathcal{S}_{\text{DLEQ}}$.*

Then, for all key pairs $(\mathbf{sk}, \mathbf{pk})$ related to ECDSA^+ , messages $m \in \{0, 1\}^$ and all statement-witness pairs $(Y, y) \in \text{DLOG}$, we have that $\tilde{\sigma}$ produced as*

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

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

Proof. Fix a key pair $(\mathbf{sk}, \mathbf{pk})$, message m and an instance $(Y, y) \in \text{DLOG}$. Consider the algorithm of $\mathcal{S}_{pS}^{\text{Sign}_{\mathbf{sk}}}$, where \mathcal{S}_{pS} is given black-box access to $\text{Sign}_{\mathbf{sk}}$ defined for ECDSA^+ . We now show that it perfectly simulates the pre-signatures of $\mathbf{aECDSA}_{\text{DLOG}}^+$ generated by $\mathbf{pSign}_{\mathbf{sk}}$.

First, on m and (Y, y) , $\mathcal{S}_{pS}^{\text{Sign}_{sk}}$ samples a signature (r^*, s^*) using Sign_{sk} . We show that $\mathcal{S}_{pS}^{\text{Sign}_{sk}}$ at the end outputs a pre-signature that adapts into (r^*, s^*) .

Recall that, in ECDSA^+ , Sign_{sk} outputs signatures only with positive s , i.e., we know that

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

In fact, only half of the random coins $k \in \mathbb{Z}_q$ are actually used in Sign_{sk} , since if randomness $k \in \mathbb{Z}_q$ produces (r, s) , then $-k$ produces $(r, -s)$, and if $s \neq 0$, only one of them is positive. However, the pre-signing algorithm pSign_{sk} produces pre-signatures using any random coin in \mathbb{Z}_q and the positivity of the corresponding signature is not checked until the Adapt algorithm, where, by the above, for fixed y , half of the random coins cause the algorithm to flip s into $-s$. Thus, to simulate this and cover all random coins of \mathbb{Z}_q , the simulator \mathcal{S}_{pS} flips s^* into $-s^*$ with probability $1/2$. We denote it by s . Clearly, (r^*, s) is an ECDSA signature sampled using some randomness $l \leftarrow \mathbb{Z}_q$.

Next, the algorithm \mathcal{S}_{pS} reverses the standard adaptation of a pre-signature and computes \tilde{s} as sy . Then, it computes group elements

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

Afterward, it uses a simulator $\mathcal{S}_{\text{DLEQ}}$ for the proof of equality of discrete logarithms that exists by the zero-knowledge property of the NIZK proof system. By the same property, the simulated proofs are distributed identically on a DLEQ statement to the proofs of the actual prover of the system on the same statement and the corresponding witness. $\mathcal{S}_{\text{DLEQ}}$ produces a simulated proof π_S of the equality on input (G, Y, \tilde{K}, K) , proving that \tilde{K} and K have the same discrete logarithm w.r.t. G and Y , respectively. Finally, \mathcal{S}_{pS} outputs a pre-signature consisting of $(r^*, \tilde{s}, K, \pi_S)$.

By construction, we have

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

The simulated pre-signature thus adapts into the original sampled signature.

Now, we show that the simulated pre-signature is valid under pk . In the process of verification, we have

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

Therefore, π_S holds on (G, Y, K', K) . By the validity of the original signature, we have

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

since it holds for both (r^*, s^*) and $(r^*, -s^*)$. It is thus clear that the simulated pre-signature is valid.

Finally, by the construction of the pre-signature, and since $\mathcal{S}_{\text{DLEQ}}$ is a simulator for proof of equality of discrete logarithms, we can see that the simulated pre-signature is, in fact, a pre-signature generated by $\text{pSign}_{sk}(m, Y)$ corresponding to a randomness ly^{-1} , which concludes the proof. \square

We now turn to the actual existential unforgeability of $\text{aECDSA}_{\text{DLOG}}^+$.

| $\mathcal{G}_0(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|---|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(Y^*, y^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, Y^*)$ | $\mathcal{O}_{pS}(m, b, (Y, y))$ |
| 5: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m^*, Y^*)$ | 1: if $b = 0$ then |
| 6: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 7: return $(m^* \notin \mathcal{Q} \wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | 3: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, Y^*)$ |
| | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge Y = yG$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, Y)$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.3 Game \mathcal{G}_0 .

Theorem 8. Let $\text{aECDSA}_{\text{DLOG}}^+$ described in Figure 2.4 be an adaptor signature scheme for ECDSA^+ w.r.t. the DLOG relation, and $(\text{P}_{\text{DLEQ}}, \text{V}_{\text{DLEQ}})$ be a NIZK proof system for equality of discrete logarithms.

If the ECDSA^+ signature scheme is SUF-CMA secure and DLOG is a hard relation, then $\text{aECDSA}_{\text{DLOG}}^+$ is aEUF-CMA secure in the standard model.

Proof. We show the reduction of the existential unforgeability of $\text{aECDSA}_{\text{DLOG}}^+$ to the strong unforgeability of the ECDSA^+ signature scheme. For brevity, let us denote by \mathcal{G}_0 the $\text{aSigForge}_{\mathcal{A}, \text{aECDSA}_{\text{DLOG}}^+}$ experiment, which will be the starting game in the reduction. First, we modify the game \mathcal{G}_0 step by step into a series of games $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$, and show that if a PPT adversary \mathcal{A} wins \mathcal{G}_0 with probability p , it also wins \mathcal{G}_3 at least with probability that is negligibly close to p . Then, we construct a PPT forger \mathcal{F} that uses \mathcal{A} to produce a fresh pair (m^*, σ^*) ; a signature σ^* on a message m^* , winning the strong unforgeability experiment $\text{strongSigForge}_{\mathcal{F}, \text{ECDSA}^+}$ at least with the same probability as \mathcal{A} wins \mathcal{G}_3 , which completes the described reduction. 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 3.3. Consider a PPT adversary \mathcal{A} winning \mathcal{G}_0 with the corresponding probability.

| $\mathcal{G}_1(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|---|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(Y^*, y^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, Y^*)$ | $\mathcal{O}_{pS}(m, b, (Y, y))$ |
| 5: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m^*, Y^*)$ | 1: if $b = 0$ then |
| 6: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 7: if $\text{Adapt}(\tilde{\sigma}, y^*) = \sigma^*$ then | 3: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, Y^*)$ |
| abort | 4: return $\tilde{\sigma}$ |
| 8: return $(m^* \notin \mathcal{Q}$ | 5: else if $b = 1 \wedge Y = yG$ then |
| $\wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, Y)$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.4 Game \mathcal{G}_1 .

Game \mathcal{G}_1 : Game \mathcal{G}_1 in Figure 3.4 is similar to \mathcal{G}_0 with a difference in the main part of the game. After the game samples a pre-signature $\tilde{\sigma}$ and \mathcal{A}_2 outputs σ^* , we now check if σ^* coincides with a signature that is obtained by adapting the pre-signature $\tilde{\sigma}$ using y^* ; a witness sampled at the beginning of the game. In such case, the game aborts. This change will in the final reduction ensure that the pair (m^*, σ^*) produced by the forger \mathcal{F} is unique, 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.

Claim 9. *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 only with non-negligible probability. Then, we could break the hardness of DLOG relation with non-negligible probability and for a given random $Y^* \in \mathcal{L}_{\text{DLOG}}$ find its discrete logarithm y^* using \mathcal{A} . The PPT solver solving this would be similar to game \mathcal{G}_0 , however, it would not be generating the pair (Y^*, y^*) , rather than obtaining generated random $Y^* \in \mathbb{G}$ on input as a challenge. The solver would then use \mathcal{A} as in \mathcal{G}_0 and after obtaining σ^* , it would extract y' with $\text{Ext}(\sigma^*, \tilde{\sigma}, Y^*)$ as defined in Figure 2.4. As assumed, σ^* is with non-negligible probability the adapted signature $\text{Adapt}(\tilde{\sigma}, y^*)$, where y^* is a discrete logarithm of Y^* w.r.t. the generator G . Therefore, by the pre-signature correctness of $\text{aECDSA}_{\text{DLOG}}^+$, we have $y' = y^*$ at least with the same probability. Thus, there would exist a PPT solver computing the discrete

| $\mathcal{G}_2(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|--|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(Y^*, y^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, Y^*)$ | $\mathcal{O}_{pS}(m, b, (Y, y))$ |
| 5: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m^*, Y^*)$ | 1: if $b = 0$ then |
| 6: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 7: if $\text{Adapt}(\tilde{\sigma}, y^*) = \sigma^*$ then | 3: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m, (Y^*, y^*))$ |
| abort | 4: return $\tilde{\sigma}$ |
| 8: return $(m^* \notin \mathcal{Q}$ | 5: else if $b = 1 \wedge Y = yG$ then |
| $\wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m, (Y, y))$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.5 Game \mathcal{G}_2 .

logarithm of Y^* w.r.t. the generator G , breaking the hardness of the relation. This concludes the proof of Claim 9 \square

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

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

Game \mathcal{G}_2 : The difference in \mathcal{G}_2 shown in Figure 3.5 compared to game \mathcal{G}_1 is in the work of the pre-signing oracle. Instead of sampling a pre-signature via pSign_{sk} algorithm as in \mathcal{G}_1 , we simulate the pre-signature in both if-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 on the input. Thus, with the knowledge of the witness y , we can produce a pre-signature $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m, (Y, y))$ when the simulator is given a black-box access to Sign_{sk} . By Lemma 7, $\tilde{\sigma}$ 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 $\text{pSign}_{\text{sk}}(m, Y)$. Therefore, the adversary cannot distinguish between \mathcal{O}_{pS} 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 depicted in Figure 3.6 is similar to game \mathcal{G}_2 with a change in the main part of the game. We now simulate the pre-signature $\tilde{\sigma}$ on

| $\mathcal{G}_3(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|--|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(Y^*, y^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, Y^*)$ | $\mathcal{O}_{pS}(m, b, (Y, y))$ |
| 5: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m^*, (Y^*, y^*))$ | 1: if $b = 0$ then |
| 6: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 7: if $\text{Adapt}(\tilde{\sigma}, y^*) = \sigma^*$ then abort | 3: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m, (Y^*, y^*))$ |
| 8: return $(m^* \notin \mathcal{Q} \wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge Y = yG$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m, (Y, y))$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.6 Game \mathcal{G}_3 .

m^* w.r.t. the sampled statement Y^* using $\mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m^*, (Y^*, y^*))$ exactly as in the pre-signing oracle of game \mathcal{G}_2 . Therefore, precisely as in game \mathcal{G}_2 , by Lemma 7, 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 3.7 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 (sk, 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}_S^+ , which, on the input message m , samples an ECDSA⁺ signature on m signed using the unknown sk . Note that \mathcal{O}_S^+ is the signing oracle provided in the strong unforgeability experiment $\text{strongSigForge}_{\mathcal{F}, \text{ECDSA}^+}$. 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}_S or simulating pre-signatures by \mathcal{S}_{pS} , the forger \mathcal{F} now uses the provided signing oracle \mathcal{O}_S^+ instead of Sign_{sk} . Thus, the simulator \mathcal{S}_{pS} has access to query \mathcal{O}_S^+ . This change in the source of signatures is mostly syntactical, since, by the definition of \mathcal{O}_S^+ , the oracle and pSign_{sk} both sample signatures with the same distribution. For clarity, we describe how the forger simulates not only \mathcal{O}_S and \mathcal{O}_{pS} queries to the adversary, but also the pre-signature $\tilde{\sigma}$.

| $\mathcal{F}^{\mathcal{O}_S^+}(\text{pk})$ | $\mathcal{O}_S(m)$ |
|---|--|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \mathcal{O}_S^+(m)$ |
| 2: $(Y^*, y^*) \leftarrow \text{GenR}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, Y^*)$ | 3: return σ |
| 4: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathcal{O}_S^+}(m^*, (Y^*, y^*))$ | $\mathcal{O}_{pS}(m, b, (Y, y))$ |
| 5: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | 1: if $b = 0$ then |
| 6: return (m^*, σ^*) | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 3: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathcal{O}_S^+}(m, (Y^*, y^*))$ |
| | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge Y = yG$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathcal{O}_S^+}(m, (Y, y))$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.7 Forger \mathcal{F} .

Signature queries: When \mathcal{A} queries \mathcal{O}_S on a message m , the forger forwards the message to \mathcal{O}_S^+ . Upon \mathcal{O}_S^+ 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}_{pS} 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}_S^+ , 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 need to show that the forger \mathcal{F} plays $\text{strongSigForge}_{\mathcal{F}, \text{ECDSA}^+}$; the strong unforgeability experiment for ECDSA^+ , and wins with probability at least $\Pr[\mathcal{G}_3^{\mathcal{A}} = 1]$ when using \mathcal{A} . As described, the forger has access to query the signing oracle \mathcal{O}_S^+ from $\text{strongSigForge}_{\mathcal{F}, \text{ECDSA}^+}$. In this experiment, (sk, 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} .

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}_S or \mathcal{O}_{pS} on this

message. Since these oracles are simulated using \mathcal{O}_S^+ , it means that \mathcal{F} did not query \mathcal{O}_S^+ on m^* in these simulated queries. The only query on m^* was done by \mathcal{S}_{pS} to \mathcal{O}_S^+ when simulating the pre-signature $\tilde{\sigma}$. Let us now denote by σ_1 the signature that \mathcal{O}_S^+ answered. By the construction of \mathcal{S}_{pS} , we have

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

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 $\text{strongSigForge}_{\mathcal{F}, \text{ECDSA}^+}$ experiment. We have

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

which implies

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

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

We now formalize the proof of the existential unforgeability of $\text{aECDSA}_{\text{DLOG}}^+$ using the second approach, where we do not assume the existence of a NIZK proof for discrete logarithms. The core of the proof is analogous to the one given for Theorem 8, however, now, we prove the existential unforgeability in the random oracle model.

Theorem 10. *Let $\text{aECDSA}_{\text{DLOG}}^+$ described in Figure 2.4 be an adaptor signature scheme for ECDSA^+ w.r.t. the DLOG relation. If ECDSA^+ signature scheme is SUF-CMA secure and DLOG is a hard relation, then $\text{aECDSA}_{\text{DLOG}}^+$ is aEUFCMA secure in the random oracle model.*

Proof. By Chapter 1, there exists a NIZK proof for equality of discrete logarithms in the random oracle model. Specifically, we have a construction $(\text{P}_{\text{DLEQ}}, \text{V}_{\text{DLEQ}})$ of such a proof in Figure 1.2 with a corresponding simulator described in Figure 1.3. The reduction proving the statement would be analogous to the reduction in the proof of Theorem 8. However, since the existential unforgeability is considered in the random oracle model, we now have to consider the random oracle \mathcal{H} available to the adversary. We will refer to games $\mathcal{G}_0 - \mathcal{G}_3$ and the forger \mathcal{F} defined in the proof of Theorem 8. The random oracle \mathcal{H} is formally described in Figure 3.8 and is provided to the adversary \mathcal{A} both in games $\mathcal{G}_0 - \mathcal{G}_3$ and in the forger \mathcal{F} . The oracle stores the queries together with their answers in H . Internally, for an input x , on which the oracle has not been queried, $H[x]$ is \perp , otherwise it is the

| $\mathcal{H}(x)$ |
|--|
| 1 : if $H[x] = \perp$ then |
| 2 : $H[x] \leftarrow \mathbb{Z}_q$ |
| 3 : return $H[x]$ |

Figure 3.8 Random oracle \mathcal{H} .

output sampled previously for x . For new queries, the random oracle chooses the answer uniformly from \mathbb{Z}_q . We make the random oracle available to the NIZK proof system and consider oracle-aided algorithms $(P_{\text{DLEQ}}^{\mathcal{H}}, V_{\text{DLEQ}}^{\mathcal{H}})$ and $\mathcal{S}_{\text{DLEQ}}^{\mathcal{H}}$ when producing and verifying the proofs. Note that for the reduction, both the hash function used in the NIZK proof system and the hash function H in the scheme are modeled as a random oracle. We consider the proof of Theorem 8 and in the following, we summarize the most important changes.

- In every game, \mathcal{A} has access to the random oracle \mathcal{H} .
- The oracle-aided algorithms of the NIZK proof system in Figure 1.2 are used for the proofs.
- When the simulator of pre-signatures \mathcal{S}_{pS} uses $\mathcal{S}_{\text{DLEQ}}$ to simulate the NIZK proof, \mathcal{S}_{pS} receives (π_S, f) ; 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 $\mathcal{S}_{\text{DLEQ}}$ on input $(G, Y, \tilde{K}, K) \in \mathcal{L}_{\text{DLEQ}}$ returns programming

$$f: (G, Y, \tilde{K}, K, rG + c\tilde{K}, rY + cK) \mapsto c,$$

for $r, c \leftarrow \mathbb{Z}_q$. 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}_{pS} of the games starting from \mathcal{G}_2 in Figure 3.9. When \mathcal{S}_{pS} simulates the pre-signature $\tilde{\sigma}$, the pre-signing oracle \mathcal{O}_{pS} 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 NIZK properties of the proof system, the changes are indistinguishable to the adversary \mathcal{A} . Assume n_1, n_2, n_3 to be a number of queries of \mathcal{A} 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}_{pS} 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

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

Figure 3.9 Changes to the pre-signing oracle.

many \mathcal{O}_{pS} queries, and thus there exists a negligible function negl_1 such that

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

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

All changes considered, we obtain a negligible function negl such that

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

which completes the proof. □

Remark. We postpone the definition and proof of pre-signature unforgeability to Section 3.4, since we will utilize the definition of pre-signature adaptability that has yet to be presented.

3.2 Witness Extractability of Adaptor Signatures for ECDSA

3.2.1 Definition of Witness Extractability

Exactly as for the robust existential unforgeability of $\mathbf{aECDSA}_{\text{DLOG}}^+$, by the reasons discussed in the previous sections, we cannot allow an unrestricted pre-signing oracle in the definition of robust witness extractability (Definition 9) in context of $\mathbf{aECDSA}_{\text{DLOG}}^+$. We present a new definition of witness extractability, where the pre-signing oracle is modified as in the new definition of existential unforgeability for adaptor signatures. The adversary can now ask for a pre-signature on any message either w.r.t. the statement $S^* \in \mathcal{L}_R$, which is fixed in the main part of the experiment and which the adversary obtains on input, or w.r.t. any $S \in \mathcal{L}_R$ for which the adversary provides a valid witness w , i.e. $(S, w) \in R$. The new definition also differs from the robust definition in the main part of the experiment. We now generate the fixed statement S^* via **GenR** and provide it together with the witness w^* to the adversary, while the robust witness extractability allows the adversary to choose the fixed statement S^* . This allows us to simulate the pre-signature in the main part of the experiment, which enables the proof the witness unforgeability for $\mathbf{aECDSA}_{\text{DLOG}}^+$. We formalize witness unforgeability in the following definition.

Definition 15. *We say that an adaptor signature scheme $\mathbf{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R is witness extractable if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function negl such that*

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

for the experiment $\mathbf{aWitExt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ defined in Figure 3.10.

The experiment $\mathbf{aWitExt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ in Figure 3.10 is similar to the $\mathbf{aRWitExt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment in Figure 2.8. It starts by sampling a key pair (sk, pk) using **KeyGen**. In contrast to $\mathbf{aRWitExt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$, it samples a relation instance (S^*, w^*) using **GenR** and provides it to the PPT adversary \mathcal{A}_1 with pk on input. This is similar but different from the $\mathbf{aSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment, since now the adversary knows the witness to S^* . Next, the experiment proceeds as in the $\mathbf{aRWitExt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment, \mathcal{A}_1 chooses a message $m^* \in \{0, 1\}^*$ and outputs it with a state st . The experiment samples a pre-signature $\tilde{\sigma}$ on m^* w.r.t. S^* using $\mathbf{pSign}_{\text{sk}}$, and on $\tilde{\sigma}$ and st as input, \mathcal{A}_2 produces a signature σ^* . During the experiment, the adversary can query the signing oracle \mathcal{O}_S for signatures on any message, and the pre-signing oracle \mathcal{O}_{pS} for pre-signatures on any message w.r.t. two types of statements. As in experiment $\mathbf{aSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$, the adversary sets a bit b on input of \mathcal{O}_{pS} either to 0 if it asks for a pre-signature w.r.t. S^* , or to 1 if it asks for a pre-signature w.r.t. S which it provides as a tuple (S, w) with its witness to \mathcal{O}_{pS} . This ensures that the adversary cannot ask for pre-signatures w.r.t. statements for which it does not know a witness. The adversary wins if σ^* is a valid signature on m^* under pk , if the adversary did not query \mathcal{O}_S or \mathcal{O}_{pS} for signatures or pre-signatures on m^* and if the element extracted from $\sigma^*, \tilde{\sigma}$ and S^* by \mathbf{Ext}_{pk} is not a witness to S^* , which are requirements identical to robust witness extractability. Note that since the adversary obtains the witness to S^* , it can adapt the provided pre-signature $\tilde{\sigma}$ and,

| $\text{aWitExt}_{\mathcal{A}, \text{a}\Sigma_R}(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|---|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(S^*, w^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, S^*)$ | $\mathcal{O}_{pS}(m, b, (S, w))$ |
| 5: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m^*, S^*)$ | 1: if $b = 0$ then |
| 6: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, w^*, \text{st})$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 7: return $((S^*, \text{Ext}_{\text{pk}}(\sigma^*, \tilde{\sigma}, S^*)) \notin R$ $\wedge m^* \notin \mathcal{Q} \wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | 3: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S^*)$ |
| | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge (S, w) \in R$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S)$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.10 $\text{aWitExt}_{\mathcal{A}, \text{a}\Sigma_R}$ experiment.

thus, have one free valid signature on m^* , which is an advantage in comparison to the existential unforgeability for adaptor signatures. However, the adversary cannot output this signature since the extraction would yield the valid witness, breaking the last winning requirement.

3.2.2 Proof of Witness Extractability

We present the proof of witness extractability of $\text{aECDSA}_{\text{DLOG}}^+$ under the assumption of strong unforgeability of the underlying ECDSA^+ signature scheme. It proceeds similarly as the proof of robust witness extractability of $\text{aECDSA}_{\pi\text{DLOG}}^+$ in [4] and also as the proof of Theorem 14. We again reduce the witness unforgeability of the adaptor signature scheme to the strong unforgeability of ECDSA^+ . As in the discussion in the previous section, in the following statement, we assume the existence of NIZK proof of equality of discrete logarithms that allows us to show witness extractability in the standard model.

Theorem 11. *Let $\text{aECDSA}_{\text{DLOG}}^+$ described in Figure 2.4 be an adaptor signature scheme for ECDSA^+ w.r.t. the DLOG relation, and $(\text{P}_{\text{DLEQ}}, \text{V}_{\text{DLEQ}})$ be a NIZK proof system for equality of discrete logarithms. If ECDSA^+ signature scheme is SUF-CMA secure and DLOG is a hard relation, then $\text{aECDSA}_{\text{DLOG}}^+$ is witness extractable in the standard model.*

Proof. For brevity, let us denote the $\text{aWitExt}_{\mathcal{A}, \text{aECDSA}_{\text{DLOG}}^+}$ 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

| $\mathcal{G}_0(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|---|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(Y^*, y^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, (Y^*, y^*))$ | $\mathcal{O}_{pS}(m, b, (Y, y))$ |
| 5: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m^*, Y^*)$ | 1: if $b = 0$ then |
| 6: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 7: return $((Y^*, \text{Ext}_{\text{pk}}(\sigma^*, \tilde{\sigma}, Y^*)) \notin R$ $\wedge m^* \notin \mathcal{Q} \wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | 3: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, Y^*)$ |
| | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge Y = yG$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, Y)$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.11 Game \mathcal{G}_0 .

witness extractability of $\text{aECDSA}_{\text{DLOG}}^+$ to the strong unforgeability of ECDSA^+ is possible, since we are able to perfectly simulate the pre-signatures of $\text{aECDSA}_{\text{DLOG}}^+$ 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 3.11 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.

Game \mathcal{G}_1 : Game \mathcal{G}_1 in Figure 3.12 differs from game \mathcal{G}_0 in the pre-signing oracle \mathcal{O}_{pS} . The oracle in game \mathcal{G}_1 returns pre-signatures produced by the simulator $\mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}$ in Figure 3.2 given a black-box access to ECDSA^+ signing algorithm Sign_{sk} . 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 7, when given the witness, the simulator outputs valid pre-signatures that are distributed identically to the pre-signatures by pSign_{sk} on the same message and w.r.t. the same DLOG statement. Thus, \mathcal{G}_0 and \mathcal{G}_1 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 3.13. It is similar to \mathcal{G}_1 with a change in the main part of the game. We now generate the pre-signature $\tilde{\sigma}$ using the

| $\mathcal{G}_1(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|--|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(Y^*, y^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, (Y^*, y^*))$ | $\mathcal{O}_{pS}(m, b, (Y, y))$ |
| 5: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m^*, Y^*)$ | 1: if $b = 0$ then |
| 6: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 7: return $((Y^*, \text{Ext}_{\text{pk}}(\sigma^*, \tilde{\sigma}, Y^*)) \notin R$ $\wedge m^* \notin \mathcal{Q} \wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | 3: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m, (Y^*, y^*))$ |
| | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge Y = yG$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m, (Y, y))$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.12 Game \mathcal{G}_1 .

| $\mathcal{G}_2(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|--|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(Y^*, y^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, (Y^*, y^*))$ | $\mathcal{O}_{pS}(m, b, (Y, y))$ |
| 5: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m^*, (Y^*, y^*))$ | 1: if $b = 0$ then |
| 6: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 7: return $((Y^*, \text{Ext}_{\text{pk}}(\sigma^*, \tilde{\sigma}, Y^*)) \notin R$ $\wedge m^* \notin \mathcal{Q} \wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | 3: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m, (Y^*, y^*))$ |
| | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge Y = yG$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}(m, (Y, y))$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.13 Game \mathcal{G}_2 .

simulator $\mathcal{S}_{pS}^{\text{Sign}_{\text{sk}}}$ on $m^*, (Y^*, y^*)$; a message chosen by the adversary, and the DLOG instance sampled at the beginning of the game. Again, by Lemma 7, the adversary cannot distinguish between games \mathcal{G}_1 and \mathcal{G}_2 and thus

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

| $\mathcal{F}^{\mathcal{O}_S^+}(\text{pk})$ | $\mathcal{O}_S(m)$ |
|---|--|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \mathcal{O}_S^+(m)$ |
| 2: $(Y^*, y^*) \leftarrow \text{GenR}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(m^*, \text{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, (Y^*, y^*))$ | 3: return σ |
| 4: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathcal{O}_S^+}(m^*, (Y^*, y^*))$ | $\mathcal{O}_{pS}(m, b, (Y, y))$ |
| 5: $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_{pS}}(\tilde{\sigma}, \text{st})$ | 1: if $b = 0$ then |
| 6: return (m^*, σ^*) | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 3: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathcal{O}_S^+}(m, (Y^*, y^*))$ |
| | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge Y = yG$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \mathcal{S}_{pS}^{\mathcal{O}_S^+}(m, (Y, y))$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.14 Forger \mathcal{F} .

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 3.14 and differs from \mathcal{G}_2 in several places. It takes a public key pk on input and a black-box access to a signing oracle \mathcal{O}_S^+ , which for a query on a message m answers a sampled ECDSA⁺ signature on m signed with a secret key sk corresponding to pk . As in \mathcal{G}_2 , the adversary samples a DLOG instance (Y^*, y^*) by GenR . It forwards pk and (Y^*, y^*) to \mathcal{A}_1 . Then, it proceeds as in \mathcal{G}_2 . When the forger uses the simulator \mathcal{S}_{pS} to simulate the pre-signatures both in the main part of the algorithm of \mathcal{F} and in the pre-signing oracle \mathcal{O}_{pS} , the simulator is now given access to query \mathcal{O}_S^+ . The same access is also given to the signing oracle \mathcal{O}_S . 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}_S on m , the forger \mathcal{F} forwards m to \mathcal{O}_S^+ , 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}_{pS} 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}_S^+ on m . Upon \mathcal{O}_S^+ 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}_{pS} query, the forger then forwards this pre-signature back to \mathcal{A} .

However, these changes are mainly syntactical, as \mathcal{O}_S^+ works with the same distribution to Sign_{sk} in \mathcal{G}_2 . After adversary \mathcal{A}_2 produces σ^* , the forger outputs (m^*, σ^*) . Now, if pk given to \mathcal{F} is sampled by KeyGen , it is clear that \mathcal{A} cannot distinguish between \mathcal{G}_2 and \mathcal{F} .

It now suffices to show that \mathcal{F} plays $\text{strongSigForge}_{\mathcal{F}, \text{ECDSA}^+}$ experiment and wins with probability at least $\Pr[\mathcal{G}_2^{\mathcal{A}} = 1]$ when using \mathcal{A} . As in the proof of

Theorem 8, we denote the signing oracle in $\text{strongSigForge}_{\mathcal{F}, \text{ECDSA}^+}$ experiment by \mathcal{O}_S^+ . This is the signing oracle that \mathcal{F} is able to query. The strong unforgeability experiment samples (sk, pk) by 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 sign or the pre-signing oracle $\mathcal{O}_S, \mathcal{O}_{pS}$ on m^* . By the above, these queries are facilitated by forwarding m to \mathcal{O}_S^+ , which implies that \mathcal{O}_S^+ was not queried on m^* during this process. The only time when \mathcal{O}_S^+ 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}_S^+ answers with a signature that we denote by σ_1 . By the construction of \mathcal{S}_{pS} , we have

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

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

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

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

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

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

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

Together with the previous equalities, we have

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

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

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

Now, as in Theorem 10, if we consider witness extractability in a random oracle model, we do not have to assume the existence of a NIZK proof for equality of discrete logarithms. We obtain the following result, which follows from the proof of Theorem 11 and changes similar to the proof of Theorem 10.

Theorem 12. *Let $\text{aECDSA}_{\text{DLOG}}^+$ described in Figure 2.4 be an adaptor signature scheme for ECDSA^+ w.r.t. the DLOG relation. If ECDSA^+ signature scheme is SUF-CMA secure and DLOG is a hard relation, then $\text{aECDSA}_{\text{DLOG}}^+$ is witness extractable in the random oracle model.*

| |
|---|
| $\text{pSigAdapt}_{\mathcal{A}, \mathbf{a}\Sigma_R}(\lambda)$ |
| $1 : (m, \text{pk}, (S, w), \tilde{\sigma}) \leftarrow \mathcal{A}(1^\lambda)$ |
| $2 : \text{ return } ((S, w) \in R \wedge \text{pVrfy}_{\text{pk}}(m, S, \tilde{\sigma}) = 1 \wedge \text{Vrfy}_{\text{pk}}(m, \text{Adapt}(\tilde{\sigma}, w)) = 0)$ |

Figure 3.15 $\text{pSigAdapt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment.

3.3 Pre-Signature Adaptability of Adaptor Signatures for ECDSA

3.3.1 Definition of Pre-Signature Adaptability

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 of S . The drawback of the definition of perfect pre-signature adaptability in Definition 10 is that, in context of $\mathbf{aECDSA}_{\text{DLOG}}^+$, the pre-signature contains a NIZK proof, which by definition satisfies only computational soundness. Since computational soundness does not restrict the existence of valid proofs of invalid statements, there could exist a pre-signature, which would be valid under a public key and w.r.t. some statement. However, it would not be adaptable into a valid signature. The perfect pre-signature adaptability therefore does not fit the considered adaptor signature for ECDSA^+ . We now present a new definition of pre-signature adaptability.

Definition 16. *We say that an adaptor signature scheme $\mathbf{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R is pre-signature adaptable if, for every PPT adversary \mathcal{A} , there exists a negligible function negl such that*

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

for the experiment $\text{pSigAdapt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ defined in Figure 3.15.

The new definition together with the experiment $\text{pSigAdapt}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ in Figure 3.15 states that no PPT adversary can with a non-negligible probability choose a message m , a public key, a valid instance of the relation, and produce a valid pre-signature on m , which would not then adapt into a valid signature on m .

3.3.2 Proof of Pre-Signature Adaptability

We present a proof of pre-signature adaptability for $\mathbf{aECDSA}_{\text{DLOG}}^+$, where we assume a NIZK proof system for equality of discrete logarithms in the standard model. The proof in the random oracle model without this assumption would proceed analogously.

Theorem 13. *Let $\mathbf{aECDSA}_{\text{DLOG}}^+$ described in Figure 2.4 be an adaptor signature scheme for ECDSA^+ w.r.t. the DLOG relation, and $(\text{P}_{\text{DLEQ}}, \text{V}_{\text{DLEQ}})$ be a NIZK proof system for equality of discrete logarithms. Then, $\mathbf{aECDSA}_{\text{DLOG}}^+$ satisfies pre-signature adaptability in the standard model.*

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

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

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

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

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

Since $(G, Y, K', K) \in \mathcal{L}_{\text{DLEQ}}$, by the definition of DLEQ, there exists $k \in \mathbb{Z}_q$ such that

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

By definition, we also have $\text{Adapt}(\tilde{\sigma}, y) = (r, \tilde{s}y^{-1})$, where, without loss of generality, we assume that $\tilde{s}y^{-1}$ is positive. Since $\tilde{\sigma}$ is a valid pre-signature, we thus get

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

making $(r, \tilde{s}y^{-1})$ a valid signature on m .

Now, if a tuple $T := (m, \text{pk}, (Y, y), \tilde{\sigma})$ wins the $\text{pSigAdapt}_{\mathcal{A}, \text{a}\Sigma_R}$ experiment, it is clear that $\text{V}_{\text{DLEQ}}((G, Y, K', K), \pi) = 1$, even though $\tilde{T} := (G, Y, K', K) \notin \mathcal{L}_{\text{DLEQ}}$. This yields

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

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 $|\tilde{T}| \geq \lambda$, since $\log(q) \geq \lambda$. Finally, the last inequality follows from the computational soundness of the underlying NIZK proof system, which concludes the proof. \square

3.4 Pre-Signature Unforgeability of Adaptor Signatures for ECDSA

3.4.1 Definition of Pre-Signature Unforgeability

Finally, we revisit the pre-signature unforgeability. Exactly as in the security definitions of robust existential unforgeability or witness extractability, the unlimited pre-signing oracle of robust pre-signature unforgeability does not fit the context of $\text{aECDSA}_{\text{DLOG}}^+$. We modify the oracle accordingly, allowing the adversary

| $\text{pSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|---|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(S^*, w^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \tilde{\sigma}) \leftarrow \mathcal{A}^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, S^*)$ | $\mathcal{O}_{pS}(m, b, (S, w))$ |
| 5: return $(m^* \notin \mathcal{Q} \wedge \text{pVrfy}_{\text{pk}}(m^*, S^*, \tilde{\sigma}))$ | 1: if $b = 0$ then |
| | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 3: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S^*)$ |
| | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge (S, w) \in R$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S)$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.16 $\text{pSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment.

to query for pre-signatures either w.r.t. a chosen fixed statement, or w.r.t. a statement that the adversary provides with the corresponding witness on input. We define *pre-signature unforgeability under chosen message attack* (pEUF-CMA security) as follows.

Definition 17. *We say that an adaptor signature scheme $\mathbf{a}\Sigma_R$ for a signature scheme Σ w.r.t. a hard relation R is pEUF-CMA secure if for every PPT adversary \mathcal{A} there exists a negligible function negl such that*

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

for the experiment $\text{pSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ defined in Figure 3.16.

The experiment $\text{pSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ in Figure 3.16 is analogous to the experiment $\text{pRSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$. The adversary \mathcal{A} aims to forge a pre-signature on a chosen message m^* w.r.t. a given sampled statement S^* , which would be valid under a given sampled public key pk . 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^* .

3.4.2 Proof of Pre-Signature Unforgeability

As discussed in Section 2.3.4, Aumayr et al. [4] showed that robust pre-signature unforgeability follows from robust existential unforgeability and perfect pre-signature adaptability of the adaptor signature scheme. We now present an analogous reduction tailored to our new definitions. In general, the proof proceeds

| $\mathcal{G}_1(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|---|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(S^*, w^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \tilde{\sigma}) \leftarrow \mathcal{A}^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, S^*)$ | $\mathcal{O}_{pS}(m, b, (S, w))$ |
| 5: $\sigma^* := \text{Adapt}(\tilde{\sigma}, w^*)$ | 1: if $b = 0$ then |
| 6: return $(m^* \notin \mathcal{Q}$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| $\wedge \text{pVrfy}_{\text{pk}}(m^*, S^*, \tilde{\sigma})$ | 3: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S^*)$ |
| $\wedge \text{Vrfy}_{\text{pk}}(m^*, \sigma^*))$ | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge (S, w) \in R$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S)$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.17 Game \mathcal{G}_1 .

identically, however, compared to the proof in [4], We cannot assume that every valid pre-signature is adaptable, since our notion of pre-signature adaptability is weaker. 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 present a formal proof in the following.

Theorem 14. *Let $\mathbf{a}\Sigma_R$ be an adaptor signature scheme for a signature scheme Σ w.r.t. a hard relation R . If $\mathbf{a}\Sigma_R$ satisfies aEUF-CMA security and pre-signature adaptability, then $\mathbf{a}\Sigma_R$ is pEUF-CMA secure.*

Proof. We show a reduction of pre-signature unforgeability to existential unforgeability of $\mathbf{a}\Sigma_R$ assuming pre-signature adaptability. For brevity, we denote the $\text{pSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment in Figure 3.16 by \mathcal{G}_0 . Consider a PPT adversary \mathcal{A} that wins the $\text{pSigForge}_{\mathcal{A}, \mathbf{a}\Sigma_R}$ experiment with probability p .

First, we modify game \mathcal{G}_0 into game \mathcal{G}_1 in Figure 3.17. 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

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

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

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

Assume there exists a non-negligible function ν such that

| $\mathcal{B}(\lambda)$ | $\mathcal{O}_S(m)$ |
|---|---|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$ |
| 2: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(S^*, w^*) \leftarrow \text{GenR}(1^\lambda)$ | 3: return σ |
| 4: $(m^*, \tilde{\sigma}) \leftarrow \mathcal{A}^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, S^*)$ | $\mathcal{O}_{pS}(m, b, (S, w))$ |
| 5: return $(m^*, \text{pk}, (S^*, w^*), \tilde{\sigma})$ | 1: if $b = 0$ then |
| | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 3: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S^*)$ |
| | 4: return $\tilde{\sigma}$ |
| | 5: else if $b = 1 \wedge (S, w) \in R$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \text{pSign}_{\text{sk}}(m, S)$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.18 Adversary \mathcal{B} .

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

By the construction of the games, the difference is precisely the probability of $m^* \notin \mathcal{Q}$ and the pre-signature $\tilde{\sigma}$ being valid but not adaptable into a valid signature on m^* . We use this to break the pre-signature adaptability of $\text{a}\Sigma_R$. We construct a PPT adversary \mathcal{B} formally described in Figure 3.18. 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 tuple $(m^*, \text{pk}, (S^*, w^*), \tilde{\sigma})$. Clearly, \mathcal{B} plays $\text{pSigAdapt}_{\mathcal{B}, \text{a}\Sigma_R}$ experiment in Figure 3.15 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

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

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

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

Now, we construct a PPT forger $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2)$ that wins $\text{aSigForge}_{\mathcal{F}, \text{a}\Sigma_R}$ experiment using \mathcal{A} . We describe \mathcal{F} formally in Figure 3.19. Both \mathcal{F}_1 and \mathcal{F}_2 obtain access to query the signing and pre-signing oracles defined in $\text{aSigForge}_{\mathcal{F}, \text{a}\Sigma_R}$ experiment, we denote them by \mathcal{O}'_S and \mathcal{O}'_{pS} , respectively. \mathcal{F}_1 proceeds similarly to game \mathcal{G}_1 , however, it does not sample a key pair (sk, pk) by 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, the pre-signature is adapted by Adapt on $\tilde{\sigma}$ and the witness w^* into a signature σ^* , which

| $\mathcal{F}_1^{\mathcal{O}'_S, \mathcal{O}'_{pS}}(\text{pk}, \tilde{S})$ | $\mathcal{O}_S(m)$ |
|---|--|
| 1: $\mathcal{Q} := \emptyset$ | 1: $\sigma \leftarrow \mathcal{O}'_S(m)$ |
| 2: $(S^*, w^*) \leftarrow \text{GenR}(1^\lambda)$ | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| 3: $(m^*, \tilde{\sigma}) \leftarrow \mathcal{A}^{\mathcal{O}_S, \mathcal{O}_{pS}}(\text{pk}, S^*)$ | 3: return σ |
| 4: $\sigma^* := \text{Adapt}(\tilde{\sigma}, w^*)$ | $\mathcal{O}_{pS}(m, b, (S, w))$ |
| 5: $\text{st} := \sigma^*$ | 1: if $b = 0$ then |
| 6: return (m^*, st) | 2: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| $\mathcal{F}_2^{\mathcal{O}'_S, \mathcal{O}'_{pS}}(\tilde{\sigma}', \text{st})$ | 3: $\tilde{\sigma} \leftarrow \mathcal{O}'_{pS}(m, 1, (S^*, w^*))$ |
| 1: $\sigma^* := \text{st}$ | 4: return $\tilde{\sigma}$ |
| 2: return σ^* | 5: else if $b = 1 \wedge (S, w) \in R$ then |
| | 6: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ |
| | 7: $\tilde{\sigma} \leftarrow \mathcal{O}'_{pS}(m, 1, (S, w))$ |
| | 8: return $\tilde{\sigma}$ |
| | 9: else return \perp |

Figure 3.19 Forger $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2)$.

is consequently stored as a 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}_S and \mathcal{O}_{pS} as in game \mathcal{G}_1 with small changes. Instead of \mathcal{F}_1 sampling the signature and pre-signature using Sign and pSign algorithms, they are now sampled by querying \mathcal{O}'_S and \mathcal{O}'_{pS} .

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

Pre-signing queries: The adversary \mathcal{A} can still query \mathcal{O}_{pS} 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}'_{pS} on the m and the pair (S^*, w^*) . On the other hand, if \mathcal{A} queries \mathcal{O}_{pS} for a pre-signature w.r.t. a pair $(S, w) \in R$, \mathcal{F}_1 just forwards the query to \mathcal{O}'_{pS} and returns the answer to \mathcal{A} .

As explained in the previous reductions, the adversary \mathcal{A} cannot differentiate between the oracles in game \mathcal{G}_1 and those provided by \mathcal{F}_1 . If pk is sampled by KeyGen , it is clear that the adversary cannot differentiate between \mathcal{G}_1 and \mathcal{F}_1 at all. 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}_S and \mathcal{O}_{pS} . However, it immediately returns the signature σ^* encoded in st by \mathcal{F}_1 .

We now show that \mathcal{F} wins $\text{aSigForge}_{\mathcal{F}, \text{a}\Sigma_R}$ experiment with sufficient probability. In the experiment, pk is sampled correctly by KeyGen . Thus, for the message m^* chosen by \mathcal{F}_1 and the adapted signature σ^* produced by \mathcal{F}_2 , we have that σ^* is valid signature on m^* under pk and \mathcal{A} did not query to the oracles on m^* with at

least probability of \mathcal{A} winning game \mathcal{G}_1 . Since it implies that \mathcal{F} did not query the oracles on m^* , \mathcal{F} then wins the $\text{aSigForge}_{\mathcal{F}, \text{a}\Sigma_R}$ experiment using \mathcal{A} , and we have

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

Together, we obtain

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

for some negligible function negl_1 . Therefore, if there exists a PPT adversary \mathcal{A} winning $\text{pSigForge}_{\mathcal{A}, \text{a}\Sigma_R}$ experiment with non-negligible probability, and thus breaking pre-signature unforgeability of $\text{a}\Sigma_R$, we can construct a PPT forger \mathcal{F} winning $\text{aSigForge}_{\mathcal{F}, \text{a}\Sigma_R}$ experiment with also non-negligible probability, and thus breaking existential unforgeability of $\text{a}\Sigma_R$, which concludes the proof. \square

Specifically, for $\text{aECDSA}_{\text{DLOG}}^+$, we have existential unforgeability by Theorem 8 and pre-signature adaptability by Theorem 13. Thus, the pre-signature unforgeability of $\text{aECDSA}_{\text{DLOG}}^+$ follows as a corollary of Theorem 14.

Theorem 15. *Let $\text{aECDSA}_{\text{DLOG}}^+$ described in Figure 2.4 be an adaptor signature scheme for ECDSA^+ w.r.t. the DLOG relation, and $(\text{P}_{\text{DLEQ}}, \text{V}_{\text{DLEQ}})$ be a NIZK proof system for equality of discrete logarithms.*

If the ECDSA^+ signature scheme is SUF-CMA secure and DLOG is a hard relation, then $\text{aECDSA}_{\text{DLOG}}^+$ is pEUF-CMA secure in the standard model.

4 Applications of Adaptor Signatures

4.1 Relative Efficiency

In addition to security, when using adaptor signatures in practice, we are interested in efficiency of the respective schemes. By the construction of the presented adaptor signatures, it is clear that the most time-consuming part appears to be generating the pre-signatures and verifying them. In applications, this is usually done not once but sometimes a large number of times. The size of the produced pre-signatures as well as of instances of the corresponding hard-relation, is also important for the same reasons.

Efficiency of Adaptor Signatures for Practical Signature Schemes. We compare the efficiency of the pre-signing and verifying algorithms of the three adaptor signatures described in the previous chapter, namely $\mathbf{aSchnorr}_{\text{DLOG}}$, $\mathbf{aECDSA}_{\text{DLOG}}$, and $\mathbf{aECDSA}_{\pi\text{DLOG}}$, considered over a cyclic elliptic curve \mathbb{G} of prime order q in additive notation. For the NIZK proof of equality of discrete logarithms in the adaptor signatures for ECDSA, we use the construction described in Figure 1.2. We instantiate the NIZK proof of knowledge of a discrete logarithm needed in $\mathbf{aECDSA}_{\pi\text{DLOG}}$ with the standard Schnorr proof of knowledge, which we described in Section 2.2.1 when talking about the Schnorr signature scheme. As mentioned in Section 2.3.4, Aumayr et al. [4] required an online extraction property of the NIZK proof of knowledge, which can be achieved by multiple repetitions of the proof [17]. However, since we aim to illustrate the efficiency, we consider just one run of the algorithm, and thus it can be considered as a lower bound to the actual efficiency of $\mathbf{aECDSA}_{\pi\text{DLOG}}$. We also note that the hash function H may differ for each adaptor signature, as well as for the NIZK proof of knowledge of discrete logarithm and for the NIZK proof of equality of discrete logarithms.

In Table 4.1, we compare the efficiency of the \mathbf{pSign} algorithms of the three adaptor signatures, and we also compare the sizes of the produced pre-signatures. By $|\mathbb{G}|$ and $|\mathbb{Z}_q|$, we denote the size of the representation of elements of \mathbb{G} and \mathbb{Z}_q , respectively. In Table 4.2, we compare the efficiency of the respective verification algorithms \mathbf{pVrfy} . Table 4.3 then shows the sizes of the hard-relation instance $(Y, y) \in R$ for both the DLOG and πDLOG relations.

It is clear that, regarding the time and size complexity, $\mathbf{aSchnorr}_{\text{DLOG}}$ is the most efficient among the considered schemes. However, if we consider the use of adaptor signatures in the context of Bitcoin, for a long time, Bitcoin supported only the ECDSA signature scheme, and the Schnorr signature was added relatively recently in 2021 through BIP340 [22]. Therefore, in practice, ECDSA is still widely used and for this, one would prefer $\mathbf{aECDSA}_{\text{DLOG}}$ to $\mathbf{aECDSA}_{\pi\text{DLOG}}$, since it is the more efficient version of the adaptor signature for ECDSA. Moreover, there is only one implementation of the adaptor signature for the Schnorr signature scheme at the time of writing.

Note that the complexity of the adaptor signatures for ECDSA^+ compared to the corresponding schemes for ECDSA is identical in the verification algorithm,

| Operation \ AS algorithm | pSign | | |
|----------------------------------|--------------------------|----------------------------------|--|
| | aSchnorr _{DLOG} | aECDSA _{DLOG} | aECDSA _{πDLOG} |
| Gen. rand. el. of \mathbb{Z}_q | 1 | 2 | 2 |
| Scalar mult. in \mathbb{G} | 1 | 4 | 4 |
| Group op. in \mathbb{G} | 1 | - | - |
| $+\mathbb{Z}_q$ | 1 | 1 | 1 |
| $-\mathbb{Z}_q$ | - | 1 | 1 |
| $\cdot\mathbb{Z}_q$ | 1 | 3 | 3 |
| $-\mathbb{1}_{\mathbb{Z}_q}$ | - | 1 | 1 |
| $H(\cdot)$ | 1 | 2 | 2 |
| $f(\cdot)$ | - | 1 | 1 |
| Size of pre-signature | $2 \mathbb{Z}_q $ | $ \mathbb{G} + 4 \mathbb{Z}_q $ | $ \mathbb{G} + 4 \mathbb{Z}_q $ |

Table 4.1 Computational complexity of pSign and size of pre-signatures of adaptor signatures.

| Operation \ AS algorithm | pVrfy | | |
|------------------------------|--------------------------|------------------------|--|
| | aSchnorr _{DLOG} | aECDSA _{DLOG} | aECDSA _{πDLOG} |
| Scalar mult. in \mathbb{G} | 2 | 6 | 8 |
| Group op. in \mathbb{G} | 2 | 3 | 4 |
| $\cdot\mathbb{Z}_q$ | - | 2 | 2 |
| $-\mathbb{1}_{\mathbb{Z}_q}$ | - | 1 | 1 |
| $H(\cdot)$ | 1 | 2 | 3 |
| $f(\cdot)$ | - | 1 | 1 |

Table 4.2 Computational complexity of pVrfy of adaptor signatures.

| Relation R | DLOG | π DLOG |
|------------------------|---------------------------------|----------------------------------|
| Size of $(S, w) \in R$ | $ \mathbb{G} + \mathbb{Z}_q $ | $ \mathbb{G} + 3 \mathbb{Z}_q $ |

Table 4.3 Size of instances of relations R .

and increases only by one subtraction in \mathbb{Z}_q in the signing algorithm.

Adaptor Signatures on Bitcoin. For context, both ECDSA and Schnorr signature on the Bitcoin network are implemented on the elliptic curve secp256k1 specified in Table 2.1. For such a group \mathbb{G} , the secret keys sampled from \mathbb{Z}_q are 256-bit integers, and thus $|\mathbb{Z}_q|$ equals 32 bytes for both signature schemes. The public key pk for ECDSA is in compressed form of length 33 bytes and in uncompressed form of length 65 bytes, while for the Schnorr signature scheme, the public key is of length 32 bytes, the same length as the secret key.

4.2 Discreet Log Contracts

While the motivating application of adaptor signatures in [4] was mainly in generalized and payment channels on blockchain, our motivation lies in crypto-

graphic solutions to conditional payments. As we mentioned in Section 1.4, one of such solutions was proposed by Madathil et al. [5]. While their scheme has a proof of security, the construction is complicated, and to the best of our knowledge, not used in practice. Another solution, called *Discreet Log Contracts* (DLCs), which is deployed in practice, utilizes adaptor signatures in a straightforward way. DLCs were first proposed by Dryja [23], but we consider a version presented in [6]. This version is based on adaptor signatures for ECDSA, specifically in our notation, on $\text{aECDSA}_{\text{DLOG}}$.

Adaptor Signatures in DLC. At a high level, the crucial part of the adaptor-based DLCs works similarly to the toy example of Alice and Bob proposing a bet under Olivia’s supervision. We have two parties, the sender and the receiver, and additionally an oracle, which provides external data on blockchain and attests to events. Assume that the oracle attests to such an event that can result in multiple outcomes. The sender uses the adaptor signature scheme to pre-sign transactions m_i relevant to each of the possible outcomes o_i . In a basic version of the solution, the oracle publishes a random group element $R := rG$ ahead of the event and the sender computes the statements $Y_i \in \mathbb{G}$, referred to as *anticipation points*, as

$$Y_i := R + P \cdot \mathbf{H}(P \| R \| o_i),$$

for a hash function \mathbf{H} and the oracle’s public key $P := pG$ corresponding to a private key p . The sender uses the anticipation points Y_i to connect each outcome and the corresponding transaction that should be carried out when the outcome occurs. This is done precisely by the sender pre-signing each m_i w.r.t. Y_i and sending these pre-signatures to the receiver who verifies them. When the event occurs with the actual outcome o_k , the oracle then publishes a discrete logarithm y_k of Y_k , which it computes as

$$y_k := r + p \cdot \mathbf{H}(P \| R \| o_k),$$

since it knows both the sampled secret r and the secret key p . This enables the receiver to adapt the corresponding pre-signature into an actual signature, which when published on blockchain carries out the transaction.

Structure of DLC. The full DLC consists of two parties that want to exchange money according to a specific event in the future. Both parties lock their funds into a shared account and prepare transactions that move the funds according to the actual outcome of the event. Specifically, they proceed in the following steps.

1. Each party prepares pre-signatures on *contract execution transactions* according to the possible outcomes as described above, and they exchange them together with the signed *refund transactions*, which return the funds to each party if the oracle fails.
2. The parties sign *fund transactions*, which lock each party’s collateral into one shared address. This address allows access to the funds only with the signatures of both parties on a corresponding transaction.

3. When the outcome occurs and the oracle attests to this outcome by publishing y_k , the corresponding pre-signature is adapted into a signature on a contract execution transaction. Each party thus obtains a signature of the other party, which allows to move the funds from the shared address. This completes the contract.

Base Anticipation Points. In the above DLC solution, the parties, Alice and Bob, need to compute an anticipation point for each of the possible outcomes; however, the number of outcomes could be large. Consider, for instance, an event with numerical outcomes, such as the rate of a currency. Assume that the outcomes are in the form of bit strings of length n . Then, both parties need to compute 2^n anticipation points resulting in 2^n scalar multiplications in the group \mathbb{G} , where the scalar is computed by a hash function. To reduce the computational complexity for Alice and Bob, a different approach to anticipation points is presented in [6]. We show even more simplified version of this approach.

Instead of a statement Y_i for each outcome, Alice and Bob compute n *base anticipation points* B_i , $1 \leq i \leq n$, as

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

for a hash function \mathbf{H} and the oracle's public key P . Then, they compute the anticipation point Y_{o_i} for the outcome $o_i = b_1 \dots b_n \in \{0, 1\}^n$ as

$$Y_{o_i} := \sum_{\substack{1 \leq j \leq n \\ b_j=1}} B_j.$$

The oracle then attests to an event o_k by publishing the discrete logarithm y_{o_k} of Y_{o_k} , which it computes as a sum of discrete logarithms of the relevant base anticipation points

$$y_{o_k} := \sum_{\substack{1 \leq j \leq n \\ b_j=1}} r + p \cdot \mathbf{H}(P\|R\|j).$$

This approach reduces the number of scalar multiplications from 2^n to n in a trade-off with the number of group operations.

Security of DLC. Even though DLC is a solution intended to be used in practice, there is currently no formal analysis of security. We suggest that our new security definitions presented in Chapter 3 could be extended to fit the DLC setting if we consider $\mathbf{aECDSA}_{\text{DLOG}}^+$ as the underlying adaptor signature scheme. We illustrate it on existential unforgeability for adaptor signatures in Definition 14, which we basically motivated with the DLC setting in mind. At a high level, we could modify the $\mathbf{aSigForge}_{\mathbf{a}\Sigma_{R,A}}$ experiment, so that instead of one instance (Y^*, y^*) of relation R (in our case DLOG relation), the experiment would sample several such instances (Y_i^*, y_i^*) corresponding to the base anticipation points. The adversary could ask the pre-signing oracle for pre-signatures w.r.t. any binary combination of these statements Y_i^* , by providing the oracle with a binary string representing which statements are included in the combination. In other words, when called on a message m , bit $b = 0$ and string $s = s_1 \dots s_n \in \{0, 1\}^n$, the

oracle would answer with a pre-signature on m w.r.t.

$$Y := \sum_{\substack{1 \leq i \leq n \\ s_i = 1}} Y_i^*.$$

The adversary could still query for pre-signatures w.r.t. any statement Y , for which it provides a witness, and also for signatures on a chosen message m . Our existential unforgeability for adaptor signatures seems to be defined reasonably for the security of DLC. However, there is currently no definition of the security of DLC. Since its formalization is beyond the scope of the thesis, we leave it as an open problem for future research.

Conclusion

In this thesis, we investigated the security of practical adaptor signatures, specifically in the context of conditional oracle-based payments. The security of adaptor signatures has recently been studied by Aumayr et al. [4], Gerhart et al. [14], and partially by Fournier [3]. However, none of these works successfully proved the security of practical adaptor signature schemes for ECDSA.

In Chapter 2, we described why the robust security definition for adaptor signatures [4] might not fit the practical adaptor signature for ECDSA. To illustrate the limitations, we outlined a theoretical attack considering the robust security definition in the context of the practical adaptor signature for ECDSA on Bitcoin. In Chapter 3, we then presented new, more suitable security definitions, which do not allow the proposed attack. We proved that the practical adaptor signature for ECDSA satisfies the new notion of security. We concluded the thesis with an application of practical adaptor signatures for ECDSA to conditional payments, specifically Discreet Log Contracts (DLCs) [6]. We suggested that the new security definitions could be extended to accommodate the context of DLCs and potentially be used to prove the security of such constructions.

Bibliography

1. POELSTRA, Andrew. *Scriptless scripts* [online]. 2017. [visited on 2025-03-26]. Available from: <https://download.wpsoftware.net/bitcoin/wizardry/%20mw-slides/2017-05-milan-meetup/slides.pdf>.
2. MORENO-SANCHEZ, Pedro; KATE, Aniket. *Scriptless Scripts with ECDSA*. 2018. Available also from: <https://web.archive.org/web/20231115202140/https://lists.linuxfoundation.org/pipermail/lightning-dev/attachments/20180426/fe978423/attachment-0001.pdf>.
3. FOURNIER, Lloyd. *One-Time Verifiably Encrypted Signatures A.K.A. Adaptor Signatures* [online]. 2019. [visited on 2025-03-27]. Available from: <https://github.com/LLFourn/one-time-VES/blob/master/main.pdf>.
4. AUMAYR, Lukas; ERSOY, Oguzhan; ERWIG, Andreas; FAUST, Sebastian; HOSTÁKOVÁ, Kristina; MAFFEI, Matteo; MORENO-SANCHEZ, Pedro; RIAHI, Siavash. Generalized Channels from Limited Blockchain Scripts and Adaptor Signatures. In: TIBOUCHI, Mehdi; WANG, Huaxiong (eds.). *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*. Springer, 2021, vol. 13091, pp. 635–664. Lecture Notes in Computer Science. Available from DOI: 10.1007/978-3-030-92075-3_22.
5. MADATHIL, Varun; THYAGARAJAN, Sri Aravinda Krishnan; VASILOPOULOS, Dimitrios; FOURNIER, Lloyd; MALAVOLTA, Giulio; MORENO-SANCHEZ, Pedro. Cryptographic Oracle-based Conditional Payments. In: *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. Available also from: <https://www.ndss-symposium.org/ndss-paper/cryptographic-oracle-based-conditional-payments/>.
6. GUILLY, Thibaut Le; KOHEN, Nadav; KUWAHARA, Ichiro. Bitcoin Oracle Contracts: Discreet Log Contracts in Practice. In: *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2022, Shanghai, China, May 2-5, 2022*. IEEE, 2022, pp. 1–8. Available from DOI: 10.1109/ICBC54727.2022.9805512.
7. KATZ, Jonathan; LINDELL, Yehuda. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. ISBN 9781466570269. Available also from: <https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269>.
8. CHAUM, David; PEDERSEN, Torben P. Wallet Databases with Observers. In: BRICKELL, Ernest F. (ed.). *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*. Springer, 1992, vol. 740, pp. 89–105. Lecture Notes in Computer Science. Available from DOI: 10.1007/3-540-48071-4_7.

9. FIAT, Amos; SHAMIR, Adi. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: ODLYZKO, Andrew M. (ed.). *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*. Springer, 1986, vol. 263, pp. 186–194. Lecture Notes in Computer Science. Available from DOI: 10.1007/3-540-47721-7_12.
10. BONEH, Dan; LYNN, Ben; SHACHAM, Hovav. Short Signatures from the Weil Pairing. In: BOYD, Colin (ed.). *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*. Springer, 2001, vol. 2248, pp. 514–532. Lecture Notes in Computer Science. Available from DOI: 10.1007/3-540-45682-1_30.
11. SCHNORR, Claus-Peter. Efficient Identification and Signatures for Smart Cards. In: BRASSARD, Gilles (ed.). *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. Springer, 1989, vol. 435, pp. 239–252. Lecture Notes in Computer Science. Available from DOI: 10.1007/0-387-34805-0_22.
12. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Digital Signature Standard*. Washington, D.C., 2023. Tech. rep., Federal Information Processing Standards Publications (FIPS) 186-5. U.S. Department of Commerce. Available from DOI: 10.6028/NIST.FIPS.186-5.
13. POELSTRA, Andrew. *Scriptless scripts* [online]. 2017. [visited on 2025-03-26]. Available from: <https://download.wpsoftware.net/bitcoin/wizardry/%20mw-slides/2017-03-mit-bitcoin-expo/slides.pdf>.
14. GERHART, Paul; SCHRÖDER, Dominique; SONI, Pratik; THYAGARAJAN, Sri Aravinda Krishnan. Foundations of Adaptor Signatures. In: JOYE, Marc; LEANDER, Gregor (eds.). *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part II*. Springer, 2024, vol. 14652, pp. 161–189. Lecture Notes in Computer Science. Available from DOI: 10.1007/978-3-031-58723-8_6.
15. LINDELL, Yehuda. Fast Secure Two-Party ECDSA Signing. In: KATZ, Jonathan; SHACHAM, Hovav (eds.). *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*. Springer, 2017, vol. 10402, pp. 613–644. Lecture Notes in Computer Science. Available from DOI: 10.1007/978-3-319-63715-0_21.
16. MALAVOLTA, Giulio; MORENO-SANCHEZ, Pedro; SCHNEIDEWIND, Clara; KATE, Aniket; MAFFEI, Matteo. Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability. In: *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019. Available also from: <https://www.ndss-symposium.org/ndss-paper/anonymous-multi-hop-locks-for-blockchain-scalability-and-interoperability/>.

17. FISCHLIN, Marc. Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors. In: SHOUP, Victor (ed.). *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*. Springer, 2005, vol. 3621, pp. 152–168. Lecture Notes in Computer Science. Available from DOI: 10.1007/11535218_10.
18. CERTICOM RESEARCH. *SEC 2: Recommended Elliptic Curve Domain Parameters* [online]. 2010. [visited on 2025-04-01]. Available from: <https://www.secg.org/sec2-v2.pdf>.
19. BROWN, Daniel R. L.; GALLANT, Robert P. The Static Diffie-Hellman Problem. *IACR Cryptol. ePrint Arch.* 2004, p. 306. Available also from: <http://eprint.iacr.org/2004/306>.
20. SHANKS, Daniel. Class number, a theory of factorization, and genera. In: *Proceedings of Symposia in Pure Mathematics*. 1971.
21. POLLARD, John M. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*. 1978, vol. 32, no. 143, pp. 918–924.
22. WUILLE, Pieter; NICK, Jonas; RUFFING, Tim. *Schnorr Signatures for secp256k1* [online]. 2020. [visited on 2025-03-26]. Available from: <https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki>.
23. DRYJA, Thaddeus. *Discreet Log Contracts* [online]. 2017. [visited on 2025-03-26]. Available from: <https://adiabat.github.io/dlc.pdf>.