A formal Security Analysis of the EdDSA Signature Scheme

Ruhr-Universität Bochum

Fakultät für Informatik

Lehrstuhl für Kryptographie

Masterarbeit

von

Aaron Kaiser

EdDSA is a signature scheme which is widely used in practice due to its high performance. Despite the wide adoption of EdDSA, no tight security proof exists for the signature scheme. The only existing security proof analyzes the signature scheme as a canonical identification scheme onto which the Fiat-Schamir transformation is being applied, yielding a non-tight security proof.

In this thesis the security of EdDSA is analyzed, utilizing the random oracel model and the algebraic group model. Using this two methods yields a tight security proof using special variants of the discrete logarithm problem. This variant is the result of the key generation algorithm used in EdDSA. The hardness of this variant of the discrete logarithm problem is then analyzed in the generic group model.

In addition a proof in the single-user setting, a proof in the multi-user setting is also performed. This proof uses a variant of the one-more discrete logarithm, also because of the key generation algorithm.

Finally, it is shown that Ed25519 - a widely used instantiation of EdDSA - provides 125 bit security in the single-user setting and 124 bits of security in the multi-user setting. Ed448 - also a widely used instantiation of EdDSA - provides 221 bits of security in the single-user setting and 220 bits of security in the multi-user setting.

Contents

1 Introduction

The EdDSA signature scheme was first introduced in 2011 by Bernstein, Duif, Lange, Schwabe and Yang, instantiated as Ed25519 using the Edwards25519 twisted Edwards curve [1]. In 2015 Bernstein et al. published a new paper, which introduced a more general version the EdDSA signature scheme [2]. Due to its high performance and small signature size, the EdDSA signature scheme is very popular and widely used in applications such as TLS, SSH and the Signal protocol. From these papers came standards such as RFC 8032 [3] and FIPS 186-5 [4].

The original paper focused on the performance of the signature scheme and did not provide a formal security analysis of the signature scheme. The EdDSA signature scheme is closely related to the Schnorr signature scheme. Although EdDSA is related to the Schnorr signature scheme, the security proofs for Schnorr signature schemes do not apply to EdDSA. The EdDSA signature scheme uses the key prefixing modification and calculates its commitments deterministically, which does not weaken the security [5]. Besides these modifications, EdDSA also uses a different group structure, which is a prime order subgroup of a twisted Edwards curve, and clamps some bits of the private key to predefined values. Both of these modifications have not been well studied for the Schnorr signature scheme. The EdDSA signature scheme also specifies several variations of parsing the signature from a bitstring. One way of parsing the signature is to allow only one bitstring representation for a scalar and curve point, and another way is to allow multiple bitstring representations of the same scalar and curve point. This raises the question of whether the changes still result in a secure signature scheme.

The desired security notions for signature schemes is EUF-CMA or SUF-CMA security. These security notions require that no adversary is able to provide a forged signature from an arbitrary set of valid signatures for arbitrary messages. While EUF-CMA requires a forged signature for a message for which the adversary did not obtain a valid signature, SUF-CMA also counts as valid forgeries message/signature pairs that were not provided to the adversary, meaning that the adversary also wins if he is able to generate a new valid signature for a message given an already valid signature for that message.

The Schnorr signature scheme originates from a canonical identification scheme to which the Fiat-Schamir transformation is applied [6]. This transformation transforms the interactive identification scheme into a non-interactive one by making some of the values deterministic. By making the values also dependent on a message, the resulting transcript of the canonical identification scheme can be interpreted as the signature for that message [7].

In a 2020 paper, Brendel et al. showed that Ed25519 satisfies EUF-CMA and SUF-CMA security, depending on which EdDSA standard is used [8]. They did this by extracting the underlying canonical identification scheme, proving its security, and then proving the security of the constructed signature scheme via the Fiat-Schamir transformation. Due to the use of the reset lemma, the provided security proof is not tight.

Tightness is a property of a security proof. A security proof is said to be tight if the probability of success of an adversary $\mathcal{B}$ attacking problem B, constructed from adversary $\mathcal{A}$ attacking problem A, is at most smaller than the probability of success of $\mathcal{A}$ by a small constant factor.

Tight security proofs are desirable because they tightly bind the hardness of the underlying

assumption to the security of a cryptographic scheme. Without a tight security proof, it is not ruled out that an adversary may be discovered who needs considerably less effort to break the security of a cryptographic scheme compared to adversaries against its underlying assumption [9]. For that reason, much larger parameters must be used to securely instantiate the cryptographic scheme compared to the parameters needed to achieve the same level of security in the underlying assumption. This is undesired in practice, as usually a scheme becomes less efficient the larger its parameters are chosen.

For the Schnorr signature scheme, a tight security reduction can be achieved by using the algebraic group model and the random oracle model to directly show the EUF-CMA security using the discrete logarithm assumption, as shown by Fuchsbauer et al. [10], instead of analyzing it as a canonical identification scheme onto which the Fiat-Schamir transformation is applied.

This thesis uses a similar approach to the one in the paper by Fuchsbauer et al. [10] to achieve a tight security proof for EdDSA. The tight security proof is achieved by utilizing the algebraic group model and the random oracle model. However, some details of the EdDSA signature scheme have to be taken into account, which mainly is the different group structure and the key clamping, introduced by the key generation algorithm. Also, the way the signature is parsed has a major impact on the security guarantees of the EdDSA signature scheme. There are two variations how to parse the signature. One is called strict parsing and the other one is called lax parsing. Strict parsing allows only one bitstring representation of a scalar value, while lax parsing allows multiple bitstring representations of the same scalar value. Strict parsing ensures SUF-CMA security, while lax parsing only ensures EUF-CMA security.

Another important property of a signature scheme, also briefly mentioned in [8], is its multi-security. When looking at practical applications of a signature scheme, not only one user is using the signature scheme, but many users are involved, all of whom have their own key pair. In most cases, an adversary is satisfied with compromising one of the users. This leaves the question whether an adversary gains an advantage in compromising a single user if he is provided with many public keys and can request signatures for any of the provided public keys. The multi-user security of Schnorr-like signature schemes has been analyzed in several papers [11, 12], but none of them apply to EdDSA or give a tight reduction.

This thesis uses the same method of providing a tight security proof in the algebraic group model and the random oracle model to prove the security of EdDSA in the multi-user setting using a variant of the one more discrete logarithm assumption, which also takes the key clamping of EdDSA into account.

Finally, a concrete security level for common instantiations of the EdDSA signature scheme is provided by analyzing the hardness of these variants of the discrete logarithm problem and the one-more discrete logarithm problem in the generic group model.

The main contributions of this thesis are the following:

1. Providing the first tight security proof for EdDSA in the single-user setting.
2. Providing the first tight security proof for EdDSA in the multi-user setting.
3. Showing the actual bit security of several widely used instantiations of the EdDSA signature scheme.

2 Related Work

Standards for EdDSA The EdDSA signature scheme was introduced in 2011 by Bernstein et al. as the specific instance Ed25519, which is the EdDSA signature scheme instantiated with the twisted Edwards curve Edwards25519 [1]. Later in 2015, with a paper by Bernstein et al., a more general version of EdDSA was introduced, which mainly lifted some restrictions on the underlying finite field of the elliptic curve [2]. It also introduced a prehashing variant of EdDSA called HashEdDSA, while the original version is called PureEdDSA. In HashEdDSA, the message is hashed before the signature algorithm is invoked. This has advantages on memory-constrained devices because it does not have to store the entire message. In 2017, the IETF published a standard for EdDSA in its RFC 8032 [3]. This standard removes some ambiguity regarding the decoding of integers and points of the elliptic curve during signature verification. It also introduces a new variant of the signature scheme that includes an additional parameter named *context*. In addition to standardizing a general version of EdDSA, the RFC included parameters for specific instantiations Ed25519 and Ed448. In 2023, this standard was adopted by the NIST in its "Digital Signature Standard (DSS)" FIPS 186-5 [4].

Schnorr Signatures and Fiat-Schamir Transformation The EdDSA and Schnorr signature schemes have a similar structure. The Schnorr signature scheme has been introduced by Claus Peter Schnorr in 1991 [6]. It has proven to be a robust and efficient signature scheme and has undergone several security analyses [10–16]. The foundation of the Schnorr signature scheme is the canonical identification scheme [17] to which the Fiat-Schamir transformation [7] is applied. There are many proofs showing that the Fiat-Schamir transformation yields a secure signature scheme, using canonical identification schemes with different properties (e.g. [17–19]).

Related Proofs As mentioned above, there exists a paper proving the security of the Ed25519 signature scheme [8]. In this paper, the authors extracted the underlying canonical identification scheme from EdDSA and used the reset lemma from [20] to prove the impersonation security of the canonical identification scheme under the discrete logarithm assumption. This reduction turned out to be non-tight. They then reduced the EUF-CMA security of the Ed25519 signature scheme to the impersonation security of the underlying canonical identification scheme. To do this, they had to guess the position of the hash query in which they had to embed a challenge, further losing tightness.

A paper by Chalkias, Garillot and Nikolaenko analyzes the security of Ed25519 with respect to different signature decoding methods and the implementation of additional checks during the signature verification [21]. This paper also analyzes lesser known security properties such as strongly binding signatures, but already assumes SUF-CMA security of Ed25519. They also analyzed the impact of cofactorless vs. cofactored verification with respect to batch verification of Ed25519 signatures.

The multi-user security of EdDSA was briefly analyzed in a paper by Bernstein [11] after he exposed a flaw in a tight multi-user security proof for the Schnorr signature scheme by Galbraith, Malone-Lee, and Smart [22]. In this paper, Bernstein provided a tight security proof for the multi-user security of key-prefixed Schnorr signatures. The EdDSA signature scheme is also a key-prefixed version of a Schnorr signature. However, due to the clamping introduced in the key generation algorithm of EdDSA, these results do not apply directly to EdDSA. Attempting

to use the same method as in Bersteins paper would again result in a non-tight security proof, as already mentioned in the same paper.

In 2016, Kiltz et al. provided a tight bound on the multi-user security of Schorr signatures without the need for key-prefixing [12]. The tightness was a result of the random self-reducibility property of the underlying canonical identification scheme. Again, this property cannot be achieved by EdDSA due to the clamping introduced by the key generation algorithm.

Fuchsbauer et al. generated a tight security proof for the Schnorr signature scheme by using the algebraic group model [10]. They achieved this by using the representation of the commitment together with a forged signature to compute the discrete logarithm of the public key. This approach also looks promising for the EdDSA signature scheme and will be analysed in this thesis.

3 Preliminaries

3.1 Notation

3.1.1 General Notation

For an integer n, $\mathbb{Z}_n$ is defined as the residual ring $\mathbb{Z}/n\mathbb{Z}$. $a \leftarrow A$ denotes sampling the element a from a non-empty finite set A uniformly at random. $:=$ denotes a deterministic assignment of a variable. $\{0,1\}^n$ is the set of all bitstrings of length n, while $\{0,1\}^*$ denotes the set of finite bitstring of arbitrary length. (x, y) is a tuple of the two elements x and y. $\{x, y\}$ is a set of the elements x and y. At the beginning of a game a set is initialized to be the empty set $\{\}$. $\sum$ denotes a table and $\sum[x]$ denotes the value of the table at position x. Each position of the table is uninitialized at the beginning of a game. An uninitialized position in the table is denoted with the bottom symbol $\perp$. A function $f : \mathbb{N} \to \mathbb{R}$ is called negligible if for all polynomials p there exists a $N \in \mathbb{N}$ so that $\forall n \geq N : f(n) < \frac{1}{p(n)}$ is true. $\mathbf{S} \in f(x)$ denotes the set $\mathbf{S}$ of outputs of f given x as input. All algorithms are probabilistic polynomial time (ppt) unless stated otherwise. $o \leftarrow \mathcal{A}(I)$ denotes running the algorithm $\mathcal{A}$ with input I with uniform random coins and o describing its output. If $\mathcal{A}$ has additionally access to an oracle O this is denoted as $o \leftarrow \mathcal{A}^{O(\cdot)}(I)$. A security game consists of a main procedure and optionally some oracle procedures. When a game is played, the main procedure is run and adversary $\mathcal{A}$ is given some inputs and access to the oracle procedures. Based on the output of the adversary $\mathcal{A}$ and its oracle calls, the main procedure outputs 1 or 0 depending on whether the adversary $\mathcal{A}$ won the game. The message space of the signature scheme is defined as $\mathcal{M}$.

3.1.2 Algebraic Notation

A group description is denoted as a tuple $\mathbf{G} = (L, \mathbb{G}, B)$ with $\mathbb{G}$ being a cyclic group of prime order L generated by group element B. The group uses additive notation for its group law and group elements are denoted by uppercase letters A. Encoded group elements are denoted by underlining $\underline{A}$. Further information on the encoding of group elements can be found in section 4. It is assumed that there exists a group generation algorithm that, upon inputting 1^λ, outputs a group description $\mathbf{G}$ with L being λ bits in length.

3.2 Code-based reduction proofs

To perform the security proof of the EdDSA signature scheme, code-based game playing proofs are used. as introduced in [23]. In these proofs, an adversary is tasked to play (and win) against a predefined game. The game is defined by a set of instructions which are executed consecutively. At one point the game calls the adversary with some input and gets some output back from it. The game then decides, depending on the output of the adversary, whether it has won or not. In addition the adversary might get access to one or more procedures, of which the adversary is only able to observe the output of the procedure call given a specific input. Those procedures are called oracles. The adversary's advantage in a game is the adversary's ability to win the game more reliably than through the use of trivial attacks, such as guessing the answer to the game.

During the proof, these games are being modified until an adversary $\mathcal{B}$ against another problem can be constructed, that simulates the view of an adversary against the modified game. For each modification, it must be argued that there is only a negligible probability that this modification can be detected by an adversary. The adversary $\mathcal{B}$ is called a reduction. By constructing an adversary $\mathcal{B}$ against another problem it can be shown that any adversary attacking the modified game, simulated by adversary $\mathcal{B}$, can also be used to break another game, attacked by adversary $\mathcal{B}$. In other words, it says that if problem A can be reduced onto problem B, any algorithm solving problem A can be transformed into an algorithm solving problem B. This can be used to show the security of cryptographic schemes by transforming all adversaries $\mathcal{A}$ against the security of the cryptographic scheme into an adversary $\mathcal{B}$ attacking a hard mathematic assumption, to which it is believed that no efficient adversary exists.

3.2.1 Identical-Until-Bad Games

While modifying the games it has to be ensured that the advantage for an attacker to distinguish between the original and modified game is negligible. This can be achieved by constructing so called identical-until-bad games.

Definition 3.1 (Identical-until-bad games [23])**.** *Two games are called identical-until-bad games if they are syntactically equivalent except for instructions following the setting of a bad flag to true.*

Lemma 3.1 (Fundamental lemma of game-playing [23])**.** *Let G and H be identical-until-bad games and let $\mathcal{A}$ be an adversary. Then,*

$$|\Pr[G^{\mathcal{A}} \Rightarrow 1] - \Pr[H^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[bad]$$

This means that the advantage to distinguish between two identical-until-bad games is bound by the probability of the bad flag being set.

3.3 Digital Signature Scheme

A digital signature scheme is a method to ensure the authenticity of data. The signer, which is in the possession of a private key, generates a signature for a specific message. The verifier is then able to verify the authenticity of this data using the public key and the generated signature.

Definition 3.2. *A digital signature scheme SIG = (KeyGen,Sign,Verify) is a tuple of algorithms.*

> **KeyGen**: *The key generation algorithm, which upon receiving the security parameter as input outputs a matching tuple of public and private key.*
>
> **Sign**: *The signature algorithm, which upon receiving a secret key and a message, outputs a signature for that message.*
>
> **Verify**: *The verification algorithm, which upon receiving a public key, a message and a signature, outputs* 1 *if the signature gets accepted and* 0 *otherwise.*

For the digital signature scheme to be correct, it is required that $\forall(pk, sk) \in \mathsf{KeyGen}(par), m \in \mathcal{M}, \sigma \in \mathsf{Sign}(sk, m) : \mathsf{Verify}(pk, m, \sigma) = 1$

A common security notion for digital signature schemes is the existential unforgeability under chosen message attack (EUF-CMA) security. It requires that no adversary is able to forge a signature for a message to which they have not observed a valid signature, given a public key. A stronger notion, that is often used, is strong unforgeability under chosen message attack (SUF-CMA), which only requires the adversary to provide a message signature pair that has not been provided to the adversary. With this security notion, the adversary also wins if it is able to forge a new valid signature from an already valid one. Both of these notions are in the single-user setting. In the multi-user setting of these security notions, the adversary is supplied with N public keys and has to forge a signature for one of those public keys. In the following, the multi-user definitions of the EUF-CMA and SUF-CMA security notions are defined, respectively N-MU-EUF-CMA and N-MU-SUF-CMA. The single-user variant of these security notions can be seen as a special case of the multi-user definitions in which the adversary is only provided with one public key.

Definition 3.3 (N-MU-EUF-CMA)**.** *Let* $SIG = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a digital signature scheme and* N *be an integer. Let the* N*-MU-EUF-CMA game be defined in figure 1. SIG is* N*-MU-EUF-CMA secure if for all ppt adversaries* $\mathcal{A}$*, we have*

$$Adv^{N\text{-}\mathsf{MU\text{-}EUF\text{-}CMA}}_{SIG,\mathcal{A}}(\lambda) \coloneqq \Pr[N\text{-}\mathsf{MU\text{-}EUF\text{-}CMA}^{\mathcal{A}} \Rightarrow 1] \leq negl(\lambda).$$

Game N-MU-EUF-CMA
for $i \in \{1, 2, ..., N\}$
  $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\cdot)}(pk_1, pk_2, ..., pk_n)$
return $\exists i \in \{1, 2, ..., N\} : \mathsf{Verify}(pk_i, m^*, \sigma^*) \stackrel{?}{=} 1 \wedge (pk_i, m^*) \notin M$

Oracle *Sign* ($i \in \{1, 2, ..., N\}$, $m \in \mathcal{M}$)
$\sigma \leftarrow \mathsf{Sign}(sk_i, m)$
$M \coloneqq M \cup \{(pk_i, m)\}$
return σ

Figure 1: N-MU-EUF-CMA Security Game

Definition 3.4 (N-MU-SUF-CMA)**.** *Let* $SIG = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a digital signature scheme and* N *be an integer. Let the* N*-MU-SUF-CMA game be defined in figure 2. SIG is* N*-MU-SUF-CMA secure if for all ppt adversaries* $\mathcal{A}$*, we have*

$$Adv_{SIG,\mathcal{A}}^{N\text{-}\mathsf{MU\text{-}SUF\text{-}CMA}}(\lambda) \coloneqq \Pr[N\text{-}\mathsf{MU\text{-}SUF\text{-}CMA}^{\mathcal{A}} \Rightarrow 1] \leq negl(\lambda).$$

Game N-MU-SUF-CMA
for $i \in \{1, 2, ..., N\}$
  $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\cdot)}(pk_1, pk_2, ..., pk_n)$
return $\exists i \in \{1, 2, ..., N\} : \mathsf{Verify}(pk_i, m^*, \sigma^*) \stackrel{?}{=} 1 \wedge (pk_i, m^*, \sigma^*) \notin M$

Oracle *Sign* $(i \in \{1, 2, ..., N\},\ m \in \mathcal{M})$
$\sigma \leftarrow \mathsf{Sign}(sk_i, m)$
$M \coloneqq M \cup \{(pk_i, m, \sigma)\}$
return σ

Figure 2: N-MU-SUF-CMA Security Game

The N-MU-EUF-NMA security game is similar to the N-MU-EUF-CMA game. The only difference is that the adversary does not has access to an oracle to obtain valid signatures for arbitrary messages. Again the EUF-NMA security notation is a special case of the N-MU-EUF-NMA security notation with $N = 1$.

Definition 3.5 (N-MU-EUF-NMA)**.** *Let $SIG = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a digital signature scheme and N be an integer. Let the N-MU-EUF-NMA game be defined in figure 3. SIG is N-MU-EUF-NMA secure if for all ppt adversaries $\mathcal{A}$, we have*

$$Adv_{SIG,\mathcal{A}}^{N\text{-}\mathsf{MU\text{-}EUF\text{-}NMA}}(\lambda) \coloneqq \Pr[N\text{-}\mathsf{MU\text{-}EUF\text{-}NMA}^{\mathcal{A}} \Rightarrow 1] \leq negl(\lambda).$$

Game N-MU-EUF-NMA
for $i \in \{1, 2, ..., N\}$
  $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}(pk_1, pk_2, pk_n)$
return $\exists i \in \{1, 2, ..., N\} : \mathsf{Verify}(pk_i, m^*, \sigma^*) \stackrel{?}{=} 1$

Figure 3: N-MU-EUF-NMA Security Game

3.4 Security Assumptions

This thesis proves the security of the EdDSA signature scheme using two assumptions. The single-user security of EdDSA can be proven using the discrete logarithm assumption, while the multi-user security of EdDSA requires the stronger one-more discrete logarithm assumption. Both security assumptions are presented in this section.

3.4.1 Discrete Logarithm Problem

Definition 3.6 (Discrete Logarithm Problem)**.** *Let $\mathbb{G}$ be a cyclic group of order L with a generator B. The advantage of an adversary $\mathcal{A}$ is defined as following:*

$$Adv^{\mathsf{DLog}}_{\mathbb{G},\mathcal{A}} \coloneqq \Pr[a \stackrel{?}{=} a' | a \leftarrow \mathbb{Z}_L; a' \leftarrow \mathcal{A}(aB)].$$

3.4.2 One-More Discrete Logarithm

The one-more discrete logarithm assumption is stronger than the discrete logarithm assumption. In this assumption the adversary is supplied with N group elements and an oracle to obtain the discrete logarithm of up to $N-1$ group elements. The task of the adversary is to output the discrete logarithm for all supplied group elements.

Definition 3.7 (One-More Discrete Logarithm Problem [24])**.** *Let $\mathbb{G}$ be a cyclic group of order L with a generator B. Let the one-more discrete logarithm game be defined in figure 4. The advantage of an adversary $\mathcal{A}$ is defined as following:*

$$Adv^{\mathsf{OM\text{-}DLog}}_{\mathbb{G},\mathcal{A}} \coloneqq \Pr[\mathsf{OM\text{-}DLog}^{\mathcal{A}} \Rightarrow 1].$$

Game OM-DLog	**Oracle** CH()
$\mathbf{L} \coloneqq \{\}$	$N \coloneqq N + 1$
$N \coloneqq 0$	$a_i \leftarrow \mathbb{Z}_L$
$(a'_1, ..., a'_N) \leftarrow \mathcal{A}^{DL(\cdot),CH()}()$	$A_i \coloneqq a_i B$
return $\forall i \in \{1, 2, ..., N\} : a_i \stackrel{?}{=} a'_i$	$\mathbf{L} \coloneqq \mathbf{L} \cup \{a_i\}$
	return A_i

Figure 4: One-More Discrete Logarithm

The DL oracle outputs the discrete logarithm of the input element in respect to the generator B and is allowed to be called $N-1$ times.

3.5 Elliptic Curves

The EdDSA signature scheme has been defined using twisted Edwards curves as the underlying group structure. Twisted Edwards curves are a special form of elliptic curves. For the proofs performed in this thesis, no specific properties of twisted Edwards curves are used. Therefore, they will not be introduced in great detail. For more details on twisted Edwards curves, see the paper by Bernstein et al. [25]. The use of twisted Edwards curves in EdDSA is mainly for performance reasons [1].

The proofs only assume that the underlying group is albian, which is true for every elliptic curve. Later, the hardness of a special variant of the discrete logarithm assumption is analyzed in the generic group model, to calculate the concrete security level of EdDSA. For proofs in the generic group model to apply the underlying group must be generic, which is widely assumed to be true for elliptic curves. A group is generic if just the well-defined group operations can be performed on the group elements. For elliptic curves the additive group notation is used.

Elliptic curves also have a property called the cofactor. The cofactor of an elliptic curve refers to the number of points on the elliptic curve divided by the number of points in a particular subgroup. The EdDSA signature scheme is not defined to use the entire twisted Edwards curve

but instead uses the largest prime order subgroup of that twisted Edwards curve. Therefore, if the number of points on the twisted Edwards curve is N and the order of the prime order subgroup is L, the cofactor with respect to this subgroup is $\frac{N}{L}$.

3.6 Random Oracle Model (ROM)

Some of the following proofs are conducted in the random oracle model. The random oracle model was introduced by Bellare and Rogaway in 1993 [26]. In the random oracle model some primitives (in this case hash functions) are modeled as public random oracles. This means that instead of calling the hash function, the adversary has to call the random oracle provided by the challenger. This random oracle must behave like a true random function.

To simulate a truly random function in polynomial time, a process called "lazy-sampling" can be used. Lazy-sampling means that the challenger has a table that starts out empty. When the adversary queries a value from the random oracle, the challenger checks if that input is in the table. If the input is in the table, the challenger returns the output value according to the table. Otherwise, the challenger chooses an output value from a uniform random distribution and inserts it into the table for that particular input value. The challenger then returns that value.

The random oracle paradigm allows the challenger to observe and influence the behavior of the adversary. Since the random oracle behaves like a truly random function, the adversary must query the random oracle to know the output value for a given input value. Therefore, the challenger can observe any input value the adversary would have used to the hash function. Also, the challenger has the ability to program specific output values of the random oracle, as long as it is correctly distributed and consistent. Consistent means that at no time should the random oracle output two different values for the same input value.

3.7 Algebraic Group Model (AGM)

The algebraic group model was introduced in 2018 by Fuchsbauer et al. [27]. In the algebraic group model, all adversaries are modeled as being algebraic. This means that the adversary has to know a representation for each group element regarding all group elements the adversary received from the challenger. This representation has to be provided to the challenger for every group element the adversary outputs or inputs as an parameter to an oracle query. For example, if the adversary receives the group elements A and B from the challenger and at one point outputs group element C the adversary also has to output a vector $\overrightarrow{c} = (c_1, c_2)$ which satisfies: $C = c_1 A + c_2 B$. For the game proofs, the group element C, and its representation $\overrightarrow{c}$ is denoted as $[C]_{\overrightarrow{c}}$.

3.8 Generic Group Model (GGM)

Unlike the random oracle model or the algebraic group model the generic group model is not used to construct reductions from one problem to another. Rather, it is used to obtain information-theoretic lower bounds on the complexity of generic adversaries against a given problem. Generic algorithms are algorithms that perform only the defined group operations on group elements and do not exploit group-specific representations of the element.

The generic group model was first introduced by Shoup in 1997 [28]. In the generic group model the adversary does not work directly with the group elements directly. Instead the challenger provides the adversary with random bitstrings, called labels, instead of group elements. Each unique label represents an unique group element. The adversary can then use oracles to perform the defined group operations on the group elements, represented by labels. The challenger then responds with the label of the resulting group element. In this way, every structure of the group is hidden from the adversary, as it works only with random labels. The only operation the adversary can perform on its own is to compare group elements for equality by comparing labels.

In 2005, Maurer proposed an alternative definition of the generic group model [29]. The proofs conducted in this thesis will use the generic group model as defined by Shoup.

4 EdDSA Signatures

This section takes a closer look at the differences between the existing EdDSA specifications and the original Schnorr signature scheme. This section is partly inspired by [8].

As mentioned above, there are two papers by Bernstein et al., that define the EdDSA signature scheme [1,2]. The 2015 paper [2] describes a more generic version of the EdDSA signature scheme than the original publication [1]. According to [2], the EdDSA signature scheme is defined by 11 parameters, as shown in table 1. The paper also describes two variants of EdDSA. One is called PureEdDSA and the other is called HashEdDSA. HashEdDSA is a prehashing variant of the PureEdDSA signature scheme. This means that, in HashEdDSA, the message is being hashed by a hash function before it is signed or verified. Both variants can be described by the definition of the EdDSA signature scheme, by using a different preqhash function. In PureEdDSA the prehash function is simply the identity function. Another important variation in the EdDSA standard is the decoding of the signature. [2] describes two variations on how signatures can be decoded during verification. Both variations are described further in this section, as they have a major impact on the security of the EdDSA signature scheme.

There also exist two major standards for the EdDSA signature scheme. The first one is the RFC 8032, which was introduced by the IETF in 2017 [3]. In addition to publishing concrete parameterizations for the Ed25519 and Ed448 signature schemes, it also includes a variant of the EdDSA signature scheme that includes a context. The context is a separate string that can be used to separate the use of EdDSA between different protocols. As argued below, the inclusion of this context does not affect the security of the signature scheme and can be modeled as being part of the message.

The 2023 FIPS 186-5 standard [4] also includes the EdDSA signature scheme as specified in RFC 8032.

A version of the EdDSA signature scheme, representing all mentioned standards, is depicted in figure 5.

Parameter	Description
q	An odd prime power q. EdDSA uses an elliptic curve over the finite field $\mathbb{F}_q$.
b	An integer b with $2^{b-1} > q$. The bit size of encoded points on the twisted Edwards curve.
$Enc(\cdot)$	A $(b-1)$-bit encoding of elements in the underlying finite field.
$H(\cdot)$	A cryptographic hash function producing $2b$-bit output.
c	The cofactor of the twisted Edwards curve.
n	The number of bits used for the secret scalar of the public key.
a, d	The curve parameter of the twisted Edwards curve.
B	A generator point of the prime order subgroup of E.
L	The order of the prime order subgroup.
$H'(\cdot)$	A prehash function applied to the message prior to applying the Sign or Verify procedure.

Table 1: Parameter of the EdDSA signature scheme

KeyGen
$k \leftarrow \{0,1\}^b$
$(h_0, h_1, ..., h_{2b-1}) \coloneqq H(k)$
$s \leftarrow 2^n + \sum_{i=c}^{n-1} 2^i h_i$
$A \coloneqq sB$
return $(\underline{A}, k)$

Sign(k, m)
$(h_0, h_1, ..., h_{2b-1}) \coloneqq H(k)$
$s \leftarrow 2^n + \sum_{i=c}^{n-1} 2^i h_i$
$(r'_0, r'_1, ..., r'_{2b-1}) \coloneqq H(h_b|...|h_{2b-1}|m)$
$r \coloneqq \sum_{i=0}^{2b-1} 2^i r'_i$
$R \coloneqq rB$
$S \coloneqq (r + sH(\underline{R}|\underline{A}|m)) \pmod L$
return $\sigma \coloneqq (\underline{R}, S)$

Verify$(\underline{A}, \sigma \coloneqq (\underline{R}, S), m)$
return $2^c SB \stackrel{?}{=} 2^c R + 2^c H(\underline{R}|\underline{A}|m)A$

Figure 5: Generic description of the algorithms KeyGen, Sign and Verify used by the EdDSA signature scheme

4.1 Encoding of Group Elements

The encoding function encodes points on the twisted Edwards curve into a b-bit bitstring and vice versa. It is assumed that when a b-bit bitstring is decoded, the resulting point is either a valid point on the twisted Edwards curve or otherwise the decoding will fail. In this way, decoding a b-bit bitstring into a cuve point implicitly ensures that the decoded point is a valid point on the specified twisted Edwards curve. The encoding function does not ensure that each point has exactly one bitstring representation. This means that there may be multiple bitstrings mapping to the same curve point during decoding. The effect of this is included in the analysis.

4.2 Message Space

The message space $\mathcal{M}$ is defined as a bitstring of arbitrary length. To make the proof applicable to the EdDSA variant with context, the context can be modeled as part of the message.

Looking at the RFC and FIPS standards, the context is passed to a "dom" function which concatenates the context with some additional data. The resulting data is then passed as additional data to each hash function call during signature generation and verification. Since the proofs are performed in the random oracle model, the position of the data in the hash function

call, the actual content of the message, and the context are not relevant to the distribution of the random oracle. The context can be modeled as being part of the message, since the random oracle has the same uniform random distribution with or without the context.

4.3 Signature

The signature is defined as a $2b$ bitstring of the encoded curve points R concatenated with the b-bit little endian encoding of the scalar S.

The fact that S is defined as b-bit little-endian encoding poses a problem. It is possible that the decoded S is larger than the order L of the generator. The original paper [2] proposes two ways to handle decoded S values that are larger than L. The first approach is to replace S with $S \pmod{L}$ and continue verifying the signature. This is called lax parsing. The other approach is to reject all S values greater than L and fail the signature verification in that case. Parsing the integer in this way is called strict parsing.

The later proofs show that these two approaches lead to different security properties of the signature. Using strict parsing results in SUF-CMA security, while using lax parsing "only" ensures EUF-CMA security.

Both the RFC and FIPS standards require strict parsing.

4.4 Differences from Schnorr Signatures

As pointed out in [8], there are some minor differences from the traditional Schnorr signature that prevent existing proofs of the Schnorr signature scheme from being applied to EdDSA. This section points out the differences between the EdDSA signature scheme and the traditional Schnorr signature scheme.

4.4.1 Group Structure

Unlike the standard Schnorr signature scheme, which is defined over a prime order group, the EdDSA signature scheme is defined over a prime order subgroup of a twisted Edwards curve.

This may pose additional challenges, since working with group elements outside the prime order subgroup may have some unintended side effects. In the proofs using the algebraic group model, where this might become relevant, it is argued that the additional group structure of the twisted Edwards curves does not pose an additional threat to the scheme.

4.4.2 Private Key Clamping

Instead of choosing the secret scalar uniformly at random, as done in most other schemes, the secret scalar is generated by hashing a random bitstring, fixing some bits of the hash result to a specific value and then interpreting n bits of the result as the little endian representation of an integer.

To be more precise, from the lower b bits of the $2b$ bitstring the lowest c bit are set to 0, where c is the cofactor of the twisted Edwards curve, and the nth bit is set to 1. Then the first n bits are interpreted as the secret scalar s.

This is strictly less secure, in the sense of the discrete logarithm problem, than choosing the secret scalar uniformly at random. It also makes proofs in the multi-user setting more challenging, since rerandomization of a public key is not easily possible and therefore the multi-user security of EdDSA cannot be easily reduced onto the single-user security of EdDSA.

To overcome this challenge, specific variants of the discrete logarithm problem and the one-more discrete logarithm problem are introduced that take into account the specific key generation. The hardness of these problems is then studied in the generic group model.

Such a choice of the secret scalar should help to make the implementation constant time and to prevent the leakage of bits through side-channel attacks.

4.4.3 Key Prefixing

The EdDSA signature scheme also includes the public key as an additional input to the hash function when generating the challenge. This change does not reduce the security of the signature scheme and is mainly related to the multi-user security of the signature scheme. Whether key prefixing actually improves multi-user security is much debated [11, 12].

4.4.4 Deterministic Nonce Generation

The commitment is chosen as the result of a hash function instead of being chosen at random each time a signature is generated. This makes signature generation deterministic. Since the hash function can be modeled as a random oracle, the deterministic generation of the commitment does not pose any additional security risk, since it can be replaced by a random function, as shown in 4.5.

4.5 Replacing Hash Function Calls

To make it easier to work with the random oracle, the following proofs introduce a variant of the EdDSA signature scheme in which some calls to the hash function are replaced by direct sampling of a value at random or by using a random function. It is then shown that the advantage of winning the SUF-CMA game is roughly the same in both versions of the signature scheme.

Introducing EdDSA' The EdDSA' signature scheme is shown in figure 6. The difference from the original EdDSA signature scheme is that the value h is sampled uniformly at random from $\{0,1\}^{2b}$, and r' is the result of a call to random function instead of the hash function.

Theorem 4.1. *Let $\mathcal{A}$ be an adversary against SUF-CMA security of the EdDSA signature scheme. Then*

$$Adv^{SUF\text{-}CMA}_{EdDSA',\mathcal{A}}(\lambda) \leq Adv^{SUF\text{-}CMA}_{EdDSA,\mathcal{A}}(\lambda) + \frac{2(q_h+1)}{2^b}.$$

Proof Overview The different games used in the proof are depicted in figure 7. The proof uses the random oracle model. The main idea is that the values h and r_i look uniformly random to the adversary if he never queries the hash function with k or a value starting with $h_b|...|h_{2b-1}$.

KeyGen
$(h_0, h_1, ..., h_{2b-1}) \leftarrow \{0,1\}^{2b}$
$s \leftarrow 2^n + \sum_{i=c}^{n-1} 2^i h_i$
$A \coloneqq sB$
return $(\underline{A},\ k \coloneqq (s, h_b|...|h_{2b-1}))$

Sign$(k \coloneqq (s, h_b|...|h_{2b-1}),\ m)$
$(r'_0, r'_1, ..., r'_{2b-1}) \coloneqq RF(h_b|...|h_{2b-1}|m)$
$r \coloneqq \sum_{i=0}^{2b-1} 2^i r'_i$
$R \coloneqq rB$
$S \coloneqq (r + sH(\underline{R}|\underline{A}|m)) \pmod{L}$
return $\sigma \coloneqq (\underline{R}, S)$

Verify$(\underline{A}, \sigma \coloneqq (\underline{R}, S), m)$
return $2^c SB \stackrel{?}{=} 2^c R + 2^c H(\underline{R}|\underline{A}|m)A$

Figure 6: Generic description of the algorithms KeyGen, Sign and Verify used by the EdDSA' signature scheme

Since those values are unknown to the adversary, it is only able to guess those values, which is unlikely due to the high entropy of them. For this reason, these calls to the hash function can be replaced by sampling truly random values.

Formal Proof

Proof.

The proof will be conducted by gradually changing the game G_0, which is the SUF-CMA game for EdDSA, to G_4, which is the SUF-CMA game for EdDSA'. At each step it is argued that the change can be detected with at most negligible probability.

$\underline{G_0:}$ Let G_0 be defined in figure 7 by excluding all boxes except the black one. Clearly G_0 is the SUF-CMA game for EdDSA. By definition,

$$\mathrm{Adv}^{\text{SUF-CMA}}_{\text{EdDSA},\mathcal{A}}(\lambda) = \Pr[\text{SUF-CMA}^{\mathcal{A}}_{\text{EdDSA}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_1:}$ Let G_1 be defined by additionally including all blue boxes and excluding the black boxes. This change inlines the hash function calls and introduces two if conditions in the random oracle that set a bad flag if the abort condition is true. The inlining of the hash function calls ensures that the challenger does not trigger the abort conditions itself. Since the behavior of the game does not change, the changes are conceptual and the probability of winning the game is not affected. Hence,

$$\Pr[G_0^{\mathcal{A}} \Rightarrow 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_2:}$ G_2 now introduces the abort instruction in the red box. The game is aborted if the flag bad_1 is set. This abort instruction ensures that the adversary will not be able to get the hash value for the secret key k. For each individual query, the bad_1 flag is set with a probability at most $\frac{1}{2^b}$. The flag is set if the input to the hash function is equal to k. k is a value chosen

Game G_0 / G_1 / G_2 / G_3 / G_4

$k \leftarrow \{0,1\}^b$

$(h_0, h_1, ..., h_{2b-1}) := H(k)$ // G_0

if $\sum[k] = \perp$ **then** // $G_1 - G_3$
  $\sum[k] \leftarrow \{0,1\}^{2b}$
$(h_0, h_1, ..., h_{2b-1}) := \sum[k]$

$(h_0, h_1, ..., h_{2b-1}) \leftarrow \{0,1\}^{2b}$ // G_4

$s \leftarrow 2^n + \sum_{i=c}^{n-1} 2^i h_i$
$A := sB$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{H(\cdot),\mathsf{Sign}(\cdot)}(A)$
return $\mathsf{Verify}(A, m^*, \sigma^*) \wedge (m^*, \sigma^*) \notin \mathbf{Q}$

Oracle $H(m \in \{0,1\}^*)$

if $m = k$ **then** // $G_1 - G_4$
  $bad_1 := true$
  abort // $G_2 - G_4$
if m starts with $h_b|...|h_{2b-1}$ **then**
  $bad_2 := true$
  abort // $G_3 - G_4$
if $\sum[m] = \perp$ **then**
  $\sum[m] \leftarrow \{0,1\}^{2b}$
return $\sum[m]$

Oracle Sign $(m \in \mathcal{M})$

$(r'_0, r'_1, ..., r'_{2b-1}) := H(h_b|...|h_{2b-1}|m)$ // G_1

if $\sum[h_b|...|h_{2b-1}|m] = \perp$ **then** // $G_1 - G_3$
  $\sum[h_b|...|h_{2b-1}|m] \leftarrow \{0,1\}^{2b}$
$(r'_0, r'_1, ..., r'_{2b-1}) := \sum[h_b|...|h_{2b-1}|m]$

$(r'_0, r'_1, ..., r'_{2b-1}) = RF(h_b|...|h_{2b-1}|m)$ // G_4

$r := \sum_{i=0}^{2b-1} 2^i r'_i$
$R := rB$
$S := (r + sH(\underline{R}|\underline{A}|m)) \pmod{L}$
$\sigma := (\underline{R}, S)$
$\mathbf{Q} := \mathbf{Q} \cup \{(m, \sigma)\}$
return σ

Figure 7: Game $G_0 - G_4$

uniformly at random from $\{0,1\}^b$ and is hidden from the adversary. Therefore, the adversary can only guess this value. By the union bound over all hash queries q_h plus the one, which is performed by the challenger during signature verification, we obtain $\Pr[bad_1] \leq \frac{q_h+1}{2^b}$. Since G_1 and G_2 are identical-until-bad games with respect to the bad_1 flag, we have

$$|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[bad_1] \leq \frac{q_h+1}{2^b}.$$

$\underline{G_3:}$ G_3 now also introduces the abort instruction in the green box. This game also aborts if a message is queried that starts with $h_b|...|h_{2b-1}$. This abort instruction ensures that the adversary cannot obtain the discrete logarithm of the commitments by querying the hash function. For each individual query, the bad_2 flag is set with a probability at most $\frac{1}{2^b}$. The value h is the result of a random oracle call with k as input. Since the adversary is unable to query the random oracle with input k due to the abort condition introduced in G_2, the adversary has no information about h. Therefore, the adversary can only guess the value of h. By the union bound over all hash queries q_h plus the one hash, which is performed by the challenger during signature verification, we obtain $\Pr[bad_2] \leq \frac{q_h+1}{2^b}$. Since G_2 and G_3 are identical-until-bad games with respect to the bad_2 flag, we have

$$|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[bad_2] \leq \frac{q_h+1}{2^b}.$$

$\underline{G_4:}$ G_4 replaces the blue boxes in the main game and the *Sign* oracle with the orange boxes. With this change, the hash value h and the discrete logarithm of the commitments r' are randomly chosen instead of being the result of the hash function call. This change is only conceptual, since the aborts introduced in G_2 and G_3 ensure that the adversary cannot obtain these values from the hash function, and therefore these values are random to the adversary. Hence,

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \Pr[G_4^{\mathcal{A}} \Rightarrow 1].$$

Now G_4 is the same as SUF-CMA parameterized with EdDSA'. So we have

$$\Pr[G_4^{\mathcal{A}} \Rightarrow 1] = \mathrm{Adv}^{\text{SUF-CMA}}_{\text{EdDSA'},\mathcal{A}}(\lambda).$$

This proves theorem 4.1. □

The proof for the EUF-CMA security is the same as the proof for the SUF-CM security, with the only difference being the win condition for the adversary. Now that EdDSA' has been introduced, and it has been shown that the adversary cannot distinguish between these signature schemes in the SUF-CMA and EUF-CMA setting, the EdDSA' signature scheme is used instead

of the EdDSA signature scheme for the proofs in the following section. Using EdDSA' makes the proofs in the random oracle model easier.

5 The Security of EdDSA in a Single-User Setting

This section takes a closer look at the single-user security of the EdDSA signature scheme. This is done by showing the SUF-CMA and EUF-CMA security of EdDSA with different styles of signature parsing. The security based on the Ed-DLog assumption. The Ed-DLog assumption is a variation of the original discrete logarithm problem, which takes the key clamping during the key generation algorithm of EdDSA into account.

The two main theorems for the single-user security of EdDSA_{sp} and EdDSA_{lp} are:

Theorem 5.1 (Security of EdDSA with strict parsing in the single-user setting)**.** *Let $\mathcal{A}$ be an adversary against the SUF-CMA security of EdDSA with strict parsing, making at most q_h hash queries and q_o oracle queries, and $\mathbb{G}$ be a group of prime order L. Then,*

$$Adv^{SUF\text{-}CMA}_{\mathbb{G},\mathcal{A}}(\lambda) \leq Adv^{Ed\text{-}DLog}_{E,n,c,L,\mathcal{B}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+1)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}$$

Theorem 5.2 (Security of EdDSA with lax parsing in the single-user setting)**.** *Let $\mathcal{A}$ be an adversary against the EUF-CMA security of EdDSA with lax parsing, making at most q_h hash queries and q_o oracle queries, and $\mathbb{G}$ be a group of prime order L. Then,*

$$Adv^{EUF\text{-}CMA}_{\mathbb{G},\mathcal{A}}(\lambda) \leq Adv^{Ed\text{-}DLog}_{E,n,c,L,\mathcal{B}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+1)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}$$

The proof begins by showing that the EUF-NMA security of EdDSA implies the SUF-CMA/EUF-CMA security of EdDSA with different types of parsing, in the random oracle model. With this step, subsequent proofs can be performed without worrying about signature generation, and a unified chain of reduction can be used to prove the security of EdDSA with both parsing variants. Next, an algebraic intermediate game Ed-IDLOG is introduced. This intermediate game serves as a separation for proofs in the random oracle model and those in the algebraic group model. Finally, the intermediate game Ed-IDLOG is reduced to the special discrete logarithm variant Ed-DLog.

The chain of reductions can be depicted as:

$$\text{Ed-DLog} \overset{\text{AGM}}{\Rightarrow} \text{Ed-IDLOG} \overset{\text{ROM}}{\Rightarrow} \text{EUF-NMA} \overset{\text{ROM}}{\Rightarrow} \text{SUF-CMA}_{\text{EdDSA sp}}/\text{EUF-CMA}_{\text{EdDSA lp}}$$

5.1 EUF-NMA $\overset{\text{ROM}}{\Rightarrow}$ SUF-CMA$_{\text{EdDSA sp}}$

This section shows that the EUF-NMA security of EdDSA implies the SUF-CMA security of EdDSA with strict parsing using the random oracle model. The section begins with an intuition for the proof, followed by the detailed security proof.

$\underline{Sim}$ (A)
$\mathbf{ch} \leftarrow \{0,1\}^{2b}$
$s \leftarrow \{0,1\}^{2b}$
$S \coloneqq \sum_{i=0}^{2b-1} 2^i s_i \pmod{L}$
$R \coloneqq SB - \mathbf{ch}A$
return $(\underline{R}, \mathbf{ch}, S)$

Figure 8: *Sim*

Theorem 5.3 ([8]). *Let $\mathcal{A}$ be an adversary against SUF-CMA, making at most q_h hash queries and q_o oracle queries, and let $\mathbb{G}$ be a group of prime order L. Then,*

$$Adv^{SUF\text{-}CMA}_{\mathbb{G},\mathcal{A}}(\lambda) \leq Adv^{EUF\text{-}NMA}_{\mathbb{G},\mathcal{B}}(\lambda) + \frac{q_o q_h \lceil \frac{2^{2b}-1}{L} \rceil}{2^{2b}}.$$

Proof Overview The EUF-NMA security definition is close to the SUF-CMA security definition, but lacks the *Sign* oracle. To show that EUF-NMA security implies SUF-CMA security, the reduction must simulate the *Sign* oracle without knowledge of the private key.

The EdDSA signature scheme is based on the Schnorr signature scheme, which is a canonical identification scheme to which the Fiat-Shamir transformation is applied. This means that EdDSA roughly follows the structure of a canonical identification scheme by first computing a commitment R, computing a challenge $\mathbf{ch}$ using the hash function, and then computing the response S based on the commitment, challenge, and private key. The signature is the commitment and response tuple.

To generate a signature without knowing the private key, the challenge and response are chosen randomly, and the commitment is calculated based on the chosen challenge and response. The random oracle is then programmed to output the challenge given the commitment and the message as input. In this way, the resulting tuple of challenge and response is a valid signature for the given message.

For the reduction to be able to program the random oracle, the adversary must not have queried the hash function with this exact input before asking for the signature. Since the input to the hash query includes the commitment, which is the result of a random function and therefore unknown to the adversary prior to the *Sign* query. For this reason, the adversary can only guess it.

This method of simulating the *Sign* oracle and the resulting loss of advantage was first introduced in [8].

Formal Proof

Proof.

The proof begins by providing an algorithm that generates a correctly distributed tuple of commitment, challenge, and response. This algorithm is called *Sim* and is shown in figure 8.

Game G_0 / G_1 / G_2 / G_3

$(h_0, h_1, ..., h_{2b-1}) \leftarrow \{0,1\}^{2b}$
$s \leftarrow 2^n + \sum_{i=c}^{n-1} 2^i h_i$
$A := sB$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{H(\cdot),\mathsf{Sign}(\cdot)}(A)$
return $\mathsf{Verify}(A, m^*, \sigma^*) \wedge (m^*, \sigma^*) \notin \mathbf{Q}$

Oracle $H(m \in \{0,1\}^*)$

if $\sum[m] = \perp$ **then**
  $\sum[m] \leftarrow \{0,1\}^{2b}$
return $\sum[m]$

Oracle $\mathsf{Sign}\ (m \in \mathcal{M})$ // $G_0 - G_2$

$(r'_0, r'_1, ..., r'_{2b-1}) = RF(h_b|...|h_{2b-1}|m)$
$r := \sum_{i=0}^{2b-1} 2^i r'_i$
$R := rB$
$S := (r + sH(\underline{R}|\underline{A}|m)) \pmod L$ // G_0
if $\sum[\underline{R}|\underline{A}|m] \neq \perp$ **then** // $G_1 - G_2$
  $bad := true$
  $abort$ // G_2
if $\sum[\underline{R}|\underline{A}|m] = \perp$ **then**
  $\sum[\underline{R}|\underline{A}|m] \leftarrow \{0,1\}^{2b}$
$S := (r + s\sum[\underline{R}|\underline{A}|m]) \pmod L$
$\sigma := (\underline{R}, S)$
$\mathbf{Q} := \mathbf{Q} \cup \{(m, \sigma)\}$
return σ

Oracle $\mathsf{Sign}\ (m \in \mathcal{M})$ // G_3

$(R, \mathbf{ch}, S) \leftarrow Sim(A)$
if $\sum[\underline{R}|\underline{A}|m] \neq \perp$ **then**
  $bad := true$
  $abort$
$\sum[\underline{R}|\underline{A}|m] = \mathbf{ch}$
$\sigma := (\underline{R}, S)$
$\mathbf{Q} := \mathbf{Q} \cup \{(m, \sigma)\}$
return σ

Figure 9: Games $G_0 - G_3$

This procedure is taken from [8]. A proof can be found in the same paper. The formula for the min-entropy of the commitment R is also taken from that paper.

$\underline{G_0:}$ Let G_0 be defined in figure 9 by excluding all boxes except the black one. Clearly G_0 is the game SUF-CMA for EdDSA. By definition,

$$\mathrm{Adv}^{\text{SUF-CMA}}_{\mathbb{G},\mathcal{A}}(\lambda) = \Pr[\text{SUF-CMA}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_1:}$ G_1 is now defined by replacing the black box with the blue one. This change inlines the call to the hash function and introduces a bad flag in the *Sign* oracle, which is set in case the hash value for the challenge is already set before the *Sign* oracle is called. In this cases the adversary already queried the challenge for that signature, resulting in the challenger not being able to program the random oracle on that input. Without being able to program the random oracle the challenger is not able to generate a valid signature, without knowing the private key. This change is only conceptual, since it does not change the behavior of the oracle and only changes internal variables of the game. Therefore,

$$\Pr[G_0^{\mathcal{A}} \Rightarrow 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_2:}$ G_2 also contains the abort statement in the red box. The abort condition is triggered when the *bad* flag is set. Without loss of generality, it is assumed that the adversary queries the Sign oracle only once for each message, since the signature generated is deterministic and an adversary would not gain more information by multiple queries on the same message. For each individual signature query, the probability of the *bad* flag being set is at most $\frac{q_h}{2^{-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})}}$. The only parameter of the hash function that is unknown to the adversary prior to calling the Sign oracle is the commitment R. For an adversary to trigger the abort condition, he must guess the commitment R used during one of the Sign queries. $-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})$ is the min-entropy of R. r' is chosen uniformly at random from $\{0,1\}^{2b}$ and then reduced modulo L when multiplied by the generator B. At first there are 2^{2b} possible values for r'. After the reduction modulo L there are $min\{2^{2b}, L\}$ possible values for r'. If the values of L are less than 2^{2b} (which is the case in most instances of EdDSA), then the r''s are not uniformly distributed in $\mathbb{Z}_L$. Since an adversary could use this information, the min entropy of R must be considered, which takes this into account. By the Union bound over all oracle queries q_o we obtain $\Pr[bad] \leq \frac{q_o q_h}{2^{-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})}}$. Since G_1 and G_2 are identical-until-bad games, we have

$$|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[bad] \leq \frac{q_o q_h \lceil \frac{2^{2b}-1}{L} \rceil}{2^{2b}}.$$

Adversary $\mathcal{B}^{H(\cdot)}(A)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{H'(\cdot),\mathsf{Sign}(\cdot)}(A)$
return (m^*, σ^*)

Oracle $H'(m \in \{0,1\}^*)$
if $\sum[m] = \perp$ **then**
  $\sum[m] \coloneqq H(m)$
return $\sum[m]$

Oracle Sign $(m \in \mathcal{M})$
$(R, \mathbf{ch}, S) \leftarrow Sim(A)$
if $\sum[\underline{R}|\underline{A}|m] \neq \perp$ **then**
  $bad \coloneqq true$
  $abort$
$\sum[\underline{R}|\underline{A}|m] = \mathbf{ch}$
$\sigma \coloneqq (\underline{R}, S)$
$\mathbf{Q} \coloneqq \mathbf{Q} \cup \{(m, \sigma)\}$
return σ

Figure 10: Adversary $\mathcal{B}$ breaking EUF-NMA

$\underline{G_3:}$ G_3 replaces the Sign oracle with the Sign oracle in the green box. Now the signature is not generated by using the secret key, but by using the *Sim* procedure and manually setting the result of the hash function call. This change is conceptual only. *Sim* returns a correctly distributed tuple $(R, \mathbf{ch}, S)$, with $2^c SB = 2^c R + 2^c \mathbf{ch} A$, and it has been excluded that $H'(\underline{R}|\underline{A}|m)$ is set before calling the Sign oracle, so that the random oracle can be programmed to output $\mathbf{ch}$ when calling $H'(\underline{R}|\underline{A}|m)$. This ensures that $2^c SB = 2^c R + 2^c H'(\underline{R}|\underline{A}|m)A$, which means that $\sigma \coloneqq (\underline{R}, S)$ is a valid signature for the message m and was generated without using the private key s. Therefore,

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \Pr[G_3^{\mathcal{A}} \Rightarrow 1].$$

Finally, Game G_3 is well-prepared to show that there exists an adversary $\mathcal{B}$ satisfying

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \mathrm{Adv}^{\text{EUF-NMA}}_{\mathbb{G},\mathcal{B}}(\lambda). \tag{5.1}$$

To prove (5.1), an adversary $\mathcal{B}$ is defined that attacks EUF-NMA simulating the view of $\mathcal{A}$ in G_3. The adversary $\mathcal{B}$, formally defined in figure 10, is run in the EUF-NMA game, and the adversary $\mathcal{B}$ simulates *Sign* for the adversary $\mathcal{A}$. *Sign* is simulated perfectly. The hash queries of $\mathcal{A}$ are forwarded to the EUF-NMA challenger, when not set by the reduction itself.

Finally, consider $\mathcal{A}$'s output $(m^*, \sigma^* \coloneqq (\underline{R^*}, S^*))$. It is known that $2^c S^* B = 2^c R^* + 2^c H'(\underline{R^*}|\underline{A}|m^*)A$. If strict parsing is used to decode S from the signature, it is known that $0 \leq S < L$. Therefore, for every R, m pair, there is only one valid encoded S that satisfies the equation. This means that a new and valid signature cannot be generated by simply changing the S value of an already valid signature. Therefore, R or m must also be changed in order to create a new valid signature from another. Since R and m are inputs to the hash query used to generate the challenge, the result of this hash query is passed from the H hash oracle provided to the adversary $\mathcal{B}$ instead

of being set by $\mathcal{B}$ itself. Also, the existence of multiple encodings of the commitment R does not pose a problem, since if another representation of the same R is chosen, its hash value for the corresponding challenge has not been set by $\mathcal{B}$ and therefore must have been passed from the EUF-NMA challenger. For this reason, $H'(\underline{R^*}|\underline{A}|m^*) = H(\underline{R^*}|\underline{A}|m^*)$. Therefore,

$$\begin{aligned} 2^c S^* B &= 2^c R^* + 2^c H'(\underline{R^*}|\underline{A}|m^*)A \\ \Leftrightarrow 2^c S^* B &= 2^c R^* + 2^c H(\underline{R^*}|\underline{A}|m^*)A. \end{aligned}$$

This means that the forged signature of the adversary $\mathcal{A}$ is also a valid signature in the EUF-NMA game.

The runtime of adversary $\mathcal{B}$ is roughly the same as the runtime of adversary $\mathcal{A}$. Simulating a *Sign* query simply executes the ppt procedure *Sim* and sets the hash function output, the hash function H' simply forwards the query to the H hash function, and the adversary $\mathcal{B}$ simply calls $\mathcal{A}$ and outputs its forged signature.

This proves theorem 5.3. □

5.2 EUF-NMA $\overset{\text{ROM}}{\Rightarrow}$ EUF-CMA$_{\text{EdDSA lp}}$

This section shows that the EUF-NMA security of EdDSA implies the EUF-CMA security of EdDSA with lax parsing using the random oracle model. This proof is very similar to the proof of the SUF-CMA security of EdDSA with strict parsing. The modification of the games is the same as in the proof above, with the only difference being the winning condition, which is $\mathsf{Verify}(A, m^*, \sigma^*) \wedge m^* \notin \mathbf{Q}$. For this reason, this proof begins by showing the existence of an adversary $\mathcal{B}$ who breaks EUF-NMA security. The SUF-CMA security cannot be proved because there may be multiple encodings of S that map to the same $S \pmod L$, and therefore a new valid signature could be forged from an old one by simply choosing a different encoding of S, which would cause the output $H'(\underline{R^*}|\underline{A}|m^*)$ to be set by the reduction itself, and therefore the forged signature would not be a valid signature for the EUF-NMA challenger.

Theorem 5.4 ([8])**.** *Let $\mathcal{A}$ be an adversary against EUF-CMA, making at most q_h hash queries and q_o oracle queries, and $\mathbb{G}$ be a group of prime order L. Then,*

$$Adv^{EUF\text{-}CMA}_{\mathbb{G},\mathcal{A}}(\lambda) \leq Adv^{EUF\text{-}NMA}_{\mathbb{G},\mathcal{B}}(\lambda) + \frac{q_o q_h \lceil \frac{2^{2b}-1}{L} \rceil}{2^{2b}}.$$

Formal Proof

Proof.

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \mathrm{Adv}^{\text{EUF-NMA}}_{\mathbb{G},\mathcal{B}}(\lambda). \tag{5.2}$$

To prove (5.2), we define an adversary $\mathcal{B}$ attacking EUF-NMA that simulates the view of $\mathcal{A}$ in G_3. The adversary $\mathcal{B}$ formally defined in figure 11 is run in the EUF-NMA game and the

Adversary $\mathcal{B}^{H(\cdot)}(A)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{H'(\cdot),\mathsf{Sign}(\cdot)}(A)$
return (m^*, σ^*)

Oracle $H'(m \in \{0,1\}^*)$
if $\sum[m] = \perp$ **then**
  $\sum[m] \coloneqq H(m)$
return $\sum[m]$

Oracle Sign $(m \in \mathcal{M})$
1: $(R, \mathbf{ch}, S) \leftarrow Sim(A)$
2: **if** $\sum[\underline{R}|\underline{A}|m] \neq \perp$ **then**
3: $bad \coloneqq true$
4: $abort$
5: $\sum[\underline{R}|\underline{A}|m] = \mathbf{ch}$
6: $\sigma \coloneqq (\underline{R}, S)$
7: $\mathbf{Q} \coloneqq \mathbf{Q} \cup \{m\}$
8: **return** σ

Figure 11: Adversary $\mathcal{B}$ breaking EUF-NMA

adversary $\mathcal{B}$ simulates *Sign* for the adversary $\mathcal{A}$. *Sign* is simulated perfectly. The hash queries of $\mathcal{A}$ are forwarded to the EUF-NMA challenger, when not set by the reduction itself.

Finally, consider $\mathcal{A}$ output $(m^*, \sigma^* \coloneqq (\underline{R^*}, S^*))$. It is known that $2^c S^* B = 2^c R^* + 2^c H'(\underline{R^*}|\underline{A}|m^*)A$. Because we are in the EUF-CMA setting, the adversary $\mathcal{A}$ is required to provide a signature for a message m^* for which it has not requested a signature from the *Sign* oracle. Since the signature for the message m^* was not requested in the Sign oracle, the output of $H'(\underline{R^*}|\underline{A}|m^*)$ was not set by the adversary B, but must have been forwarded from the H hash oracle. For this reason, $H'(\underline{R^*}|\underline{A}|m^*) = H(\underline{R^*}|\underline{A}|m^*)$. Therefore,

$$\begin{aligned} & 2^c S^* B = 2^c R^* + 2^c H'(\underline{R^*}|\underline{A}|m^*)A \\ \Leftrightarrow\ & 2^c S^* B = 2^c R^* + 2^c H(\underline{R^*}|\underline{A}|m^*)A. \end{aligned}$$

This means that the forged signature of the adversary $\mathcal{A}$ is also a valid signature in the EUF-NMA game.

Since the adversary $\mathcal{B}$ is the same as in the proof above, the runtime is roughly the same as the runtime of $\mathcal{A}$, for the same reasons.

This proves theorem 5.4. □

5.3 Ed-IDLOG $\overset{\mathsf{ROM}}{\Rightarrow}$ EUF-NMA

This section shows that Ed-IDLOG implies the EUF-NMA security of the EdDSA signature scheme using the random oracle model. The section begins with the introduction of an intermediate game Ed-IDLOG, followed by an intuition of the proof and the detailed security proof.

Introducing Ed-IDLOG The intermediate game Ed-IDLOG is introduced to create a separation between proofs in the random oracle model and the algebraic group model. This is achieved

by replacing the random oracle with the *Chall* oracle, which takes a commitment and issues a challenge. This also removes the message and focuses on forging an arbitrary signature. The Ed-IDLOG game is shown in figure 12. The game has been inspired by the IDLOG game from [12].

Definition 5.1 (Ed-IDLOG)**.** *For an adversary $\mathcal{A}$. The advantage of $\mathcal{A}$ in the Ed-IDLOG game is defined as following:*

$$Adv^{\textit{Ed-IDLOG}}_{\mathbb{G},\mathcal{A}}(\lambda) \coloneqq |\Pr[\textit{Ed-IDLOG}^{\mathcal{A}} \Rightarrow 1]|.$$

Game Ed-IDLOG	**Oracle** *Chall* $(R_i \in \mathbb{G})$
$a \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$	$\mathbf{ch}_i \leftarrow \{0,1\}^{2b}$
$A \coloneqq aB$	$\mathbf{Q} \coloneqq \mathbf{Q} \cup \{(R_i, \mathbf{ch}_i)\}$
$s^* \leftarrow \mathcal{A}^{\textit{Chall}(\cdot)}(A)$	**return** $\mathbf{ch}_i$
return $\exists(R^*, \mathbf{ch}^*) \in \mathbf{Q} : R^* = 2^c s^* B - 2^c \mathbf{ch}^* A$	

Figure 12: Ed-IDLOG

Theorem 5.5. *Let $\mathcal{A}$ be an adversary against EUF-NMA. Then,*

$$Adv^{\textit{EUF-NMA}}_{\mathbb{G},\mathcal{A}}(\lambda) = Adv^{\textit{Ed-IDLOG}}_{\mathbb{G},\mathcal{B}}(\lambda).$$

Proof Overview The adversary must query the random oracle to obtain the hash value $H(\underline{R}|\underline{A}|m)$. The programmability of the random oracle can be used to embed the challenge from the *Chall* into the answer from the random oracle. In this way, a valid signature forgery also provides a valid solution to the Ed-IDLOG game.

Game G_0	**Oracle** $H(m \in \{0,1\}^*)$
$(h_0, h_1, ..., h_{2b-1}) \leftarrow \{0,1\}^{2b}$	**if** $\sum[m] = \perp$ **then**
$s \leftarrow 2^n + \sum_{i=c}^{n-1} 2^i h_i$	$\sum[m] \leftarrow \{0,1\}^{2b}$
$A \coloneqq sB$	**return** $\sum[m]$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{H(\cdot)}(A)$	
return $\mathsf{Verify}(A, m^*, \sigma^*)$	

Figure 13: G_0

Formal Proof

Proof.

This proof does not require any game hop, since the random oracle can be simulated using the *Chall* oracle.

$\underline{G_0:}$ Let G_0 be defined in figure 13. Clearly, G_0 is EUF-NMA. By definition,

$$\mathrm{Adv}^{\text{EUF-NMA}}_{\mathbb{G},\mathcal{A}}(\lambda) = \Pr[\text{EUF-NMA}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

G_0 is well-prepared to show that there exists an adversary $\mathcal{B}$ satisfying

$$\Pr[G_0^{\mathcal{A}} \Rightarrow 1] = \mathrm{Adv}^{\text{Ed-IDLOG}}_{\mathbb{G},\mathcal{B}}(\lambda). \tag{5.3}$$

Adversary $\mathcal{B}^{Chall(\cdot)}(A)$	**Oracle** $H(m \in \{0,1\}^*)$
$(m^*, \sigma^* := (\underline{R}, S)) \leftarrow \mathcal{A}^{H(\cdot)}(A)$	**if** $\sum[m] = \perp$ **then**
return S	**if** $\underline{R}\|\underline{A}\|m' := m \wedge R \in E$ **then**
	$\sum[m] \leftarrow Chall(2^c R)$
	else
	$\sum[m] \leftarrow \{0,1\}^{2b}$
	return $\sum[m]$

Figure 14: Adversary $\mathcal{B}$ breaking Ed-IDLOG

To prove (5.3), we define an adversary $\mathcal{B}$ attacking Ed-IDLOG that simulates the view of $\mathcal{A}$ in G_0. The adversary $\mathcal{B}$ formally defined in figure 14 is run in the Ed-IDLOG game, and the adversary $\mathcal{B}$ simulates the random oracle H for the adversary $\mathcal{A}$. H is perfectly simulated because the *Chall* oracle also outputs a uniformly random 2b-bit bitstring. For this reason, H returns a uniformly random 2b-bit bitstring for all queries, as expected.

Finally, consider $\mathcal{A}$'s output $(m^*, \sigma^* := (\underline{R}, S))$. It is known that:

$$\begin{aligned} 2^c SB &= 2^c R + 2^c H(\underline{R}|\underline{A}|m)A \\ \Leftrightarrow 2^c R &= 2^c SB - 2^c H(\underline{R}|\underline{A}|m)A \\ \Leftrightarrow 2^c R &= 2^c SB - 2^c Chall(2^c R)A \\ \Leftrightarrow R' &= 2^c SB - 2^c Chall(R')A \end{aligned}$$

Therefore, S is a valid solution for the Ed-IDLOG game.

The runtime of adversary $\mathcal{B}$ is roughly the same as the runtime of adversary $\mathcal{A}$, since it just outputs the solution of $\mathcal{A}$, and in the random oracle it either calls the *Chall* oracle or samples a value uniformly at random.

This proves theorem 5.5. □

5.4 Ed-DLog $\overset{\text{AGM}}{\Rightarrow}$ Ed-IDLOG

This section shows that Ed-DLog implies Ed-IDLOG using the algebraic group model. The section begins with an introduction to a special variant of the discrete logarithm problem, followed by an intuition of the proof, and finally a detailed security proof.

Introducing Ed-DLog The Ed-DLog game is a variant of the discrete logarithm game that represents the clearing and setting of bits in the secret scalar during EdDSA key generation. The only difference to the normal discrete logarithm game is that the secret scalars are not randomly chosen from $\mathbb{Z}_L$, where L is the order of the generator, but from the set $\{2^{n-1}, 2^{n-1}+2^c, ..., 2^n - 2^c\}$. This set represents all valid private keys according to the key generation algorithm. The hardness of this version of the discrete logarithm problem is further analyzed in section 7.1. The Ed-DLog game is illustrated in figure 15.

Definition 5.2 (Ed-DLog)**.** *For an adversary $\mathcal{A}$ we define its advantage in the Ed-DLog game as following:*

$$Adv^{Ed\text{-}DLog}_{\mathbb{G},\mathcal{A}}(\lambda) \coloneqq |\Pr[Ed\text{-}DLog^{\mathcal{A}} \Rightarrow 1]|.$$

Game Ed-DLog

$a \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$

$A \coloneqq aB$

$a' \leftarrow \mathcal{A}(A)$

return $a \stackrel{?}{=} a'$

Figure 15: Ed-DLog

Theorem 5.6. *Let $\mathcal{A}$ be an adversary against Ed-IDLOG with $\mathbb{G}$ being a cyclic group of prime order L, making at most q_o oracle queries. Then*

$$Adv^{Ed\text{-}IDLOG}_{\mathbb{G},\mathcal{A}}(\lambda) \leq Adv^{Ed\text{-}DLog}_{\mathbb{G},\mathcal{B}}(\lambda) + \frac{q_o \lceil \frac{2^{2b}-1}{L} \rceil}{2^{2b}}.$$

Proof Overview The adversary must call the *Chall* oracle with a commitment R to get a challenge from the challenger. Due to the nature of the algebraic group model, the adversary must also provide a representation of the group element R as a linear combination of all known group elements. Since only the generator of the group and the public key are known to the adversary, the representation looks like this: $R = r_1 B + r_2 A$. Together with a valid solution to the Ed-IDLOG game, this can be used to compute the discrete logarithm of the public key.

Formal Proof

Proof.

The proofs begin by showing that the only valid representation of a group element in the prime order subgroup is the one relative to all known elements in the subgroup and cannot include elements from outside the subgroup. This is followed by a discussion of the individual game-hops.

Game G_0 / G_1 / G_2

$a \leftarrow \{2^{n-1}, 2^{n-1}+8, ..., 2^n - 8\}$
$A := aB$
$s^* \leftarrow \mathcal{A}^{Chall(\cdot)}(A)$
return $\exists (R^*, \mathbf{ch}^*) \in \mathbf{Q} : R^* = 2^c(s^*B - \mathbf{ch}^*A)$

Oracle $Chall$ $([R_i]_{\overrightarrow{r_i}} \in \mathbb{G})$

Let $R_i = r_1 B + r_2 A$
$\mathbf{ch}_i \leftarrow \{0,1\}^{2b}$
If $2^c \mathbf{ch}_i \equiv -r_2 \pmod{L}$ **then** // $G_1 - G_2$
  $bad := true$
  $abort$ // G_2
$\mathbf{Q} := \mathbf{Q} \cup \{(R_i, \mathbf{ch}_i)\}$
return $\mathbf{ch}_i$

Figure 16: Games $G_0 - G_2$

AGM This proof is done in the algebraic group model. This means that the adversary has to provide a representation along each group element he provides to the reduction. The adversary must provide an element R which is an element in the prime order subgroup of the twisted Edwards curve. The question remains whether the representation should be defined relative to the prime order subgroup or the twisted Edwards curve. The answer to this question is that it is sufficient to define the representation relative to the prime order subgroup. The reason for this is given in the following paragraph.

The twisted Edwards curve E over the finite field $\mathbb{F}_q$ is a finite abelian group. Even though the group E may not be cyclic, the Fundamental Theorem of Finitely Generated Abelian Groups tells us that every finite abelian group can be uniquely decomposed into the direct product of cyclic subgroups [30]. This means that E can be written as $E = \langle a_1 \rangle \bigotimes \langle a_2 \rangle \bigotimes ... \bigotimes \langle a_n \rangle$. The set of generators for each of the cyclic groups is called the generating set of E. Let us recall a well-known theorem of algebra:

Theorem 5.7 (Characterization of Inner Direct Products [31])**.** *Let $N_1, ..., N_n$ be subgroups of a group $\mathbb{G}$. Following statements are equivalent:*

(1) $N_1, ..., N_n \trianglelefteq \mathbb{G}$ and $\mathbb{G} = N_1 \bigotimes ... \bigotimes N_n$.

(2) Each $x \in \mathbb{G}$ can uniquely be represented in the following way:

$$x = a_i \cdot ... \cdot a_n, a_i \in N_i$$

Due to Sylow's theorems, the decomposition must include the large prime order subgroup $\mathbb{G}$ used for EdDSA [32], and since twisted Edwards curves (like all elliptic curves) are abelian, each subgroup is also a normal subgroup. Together this means that the representation of each element $X \in E$ is unique relative to the generating set. Since each element $Y \in \mathbb{G}$ can be represented as $Y := yB$, where B is the generator of the prime order subgroup, this must be the only representation with respect to the generating set. This means that an adversary, in the algebraic group model, must provide a representation in the prime order subgroup $\mathbb{G}$.

The only two group elements in $\mathbb{G}$ provided to the adversary are the public key A and the generator B. Therefore, the representation of the element R, provided to the *Chall* oracle,

looks like $R = r_1 B + r_2 A$.

G_0**:** Let G_0 be defined in figure 16 by excluding all boxes. G_0 is the same as Ed-IDLOG. By definition,

$$\mathrm{Adv}^{\text{Ed-IDLOG}}_{\mathbb{G},\mathcal{A}}(\lambda) = \Pr[\text{Ed-IDLOG}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

G_1**:** Game G_1 is exactly the same as G_0 with the only change being that the bad flag is set inside an if condition. The bad flag is set when $2^c \mathbf{ch}_i = -r_2$. This represents cases where not all solutions from the adversary $\mathcal{A}$ can be used to calculate the discrete logarithm of A. This is only a conceptual change, since the behavior of the game does not change whether the flag is set or not. Hence,

$$\Pr[G_0^{\mathcal{A}} \Rightarrow 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1].$$

G_2 **:** The game G_2 is aborted if the bad flag is set. For each individual *Chall* query, the *bad* flag is set with probability at most $\frac{1}{2^{-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})}}$. $\mathbf{ch}_i$ is chosen by the game after the adversary has provided the representation of R_i and thus the value of r_2. This way the adversary has no way to choose $\mathbf{ch}_i$ after r_2 and therefore cannot influence the probability of the abort being triggered. $-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})$ is the min entropy of $\mathbf{ch}_i \pmod L$. $\mathbf{ch}_i$ is chosen uniformly at random from $\{0,1\}^{2b}$ and then reduced modulo L during the if condition check. By the union bound over all oracle queries q_o we obtain $\Pr[bad] \leq \frac{q_o}{2^{-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})}}$. Since G_1 and G_2 are identical-until-bad games, we have

$$|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[bad] \leq \frac{q_o \lceil \frac{2^{2b}-1}{L} \rceil}{2^{2b}}.$$

Finally, Game G_2 is well-prepared to show that there exists an adversary $\mathcal{B}$ satisfying

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \mathrm{Adv}^{\text{Ed-DLog}}_{\mathbb{G},\mathcal{B}}(\lambda). \tag{5.4}$$

To prove (5.4), we define an adversary $\mathcal{B}$ attacking Ed-DLog, which simulates the view of $\mathcal{A}$ in G_2. The adversary $\mathcal{B}$ formally defined in figure 17 is run in the Ed-DLog game and adversary $\mathcal{B}$ simulates *Chall* for adversary $\mathcal{A}$. The *Chall* oracle is simulated perfectly.

Finally, consider $\mathcal{A}$'s output s^*. We know that one $R^* = 2^c s^* B - 2^c \mathbf{ch}^* A$. We can use this together with the representation of R^* to get the following equation:

Adversary $\mathcal{B}(A)$	**Oracle** *Chall* $([R_i]_{\vec{r_i}} \in \mathbb{G})$
$s^* \leftarrow \mathcal{A}^{Chall(\cdot)}(A)$	Let $R_i = r_1 B + r_2 A$
If $\nexists [R^*]_{\vec{r^*}}, \mathbf{ch}^* : R^* = 2^c(s^* B - \mathbf{ch}^* A) \wedge$	$\mathbf{ch}_i \leftarrow \{0,1\}^{2b}$
$([R^*]_{\vec{r^*}}, \mathbf{ch}^*) \in \mathbf{Q}$ **then**	**If** $2^c \mathbf{ch}_i \equiv -r_2 \pmod{L}$ **then**
abort	$bad := true$
Let $R^* = r_1 B + r_2 A$	*abort*
return $(2^c s^* - r_1)(r_2 + 2^c \mathbf{ch}^*)^{-1}$	$\mathbf{Q} := \mathbf{Q} \cup \{([R_i]_{\vec{r_i}}, \mathbf{ch}_i)\}$
	return $\mathbf{ch}_i$

Figure 17: Adversary $\mathcal{B}$ breaking Ed-DLog

$$\begin{aligned} r_1 B + r_2 A &= 2^c s^* B - 2^c \mathbf{ch}^* A \\ \Leftrightarrow (r_2 + 2^c \mathbf{ch}^*) A &= (2^c s^* - r_1) B \\ \Leftrightarrow A &= (2^c s^* - r_1)(r_2 + 2^c \mathbf{ch}^*)^{-1} B \end{aligned}$$

Assuming that $r_2 + 2^c \mathbf{ch}^*$ is invertible in $\mathbb{Z}_L$ (i.e. not equal to 0), which is ensured due to the abort in G_2, both equations can be used to calculate the discrete logarithm of A.

Obviously, the runtime of $\mathcal{B}$ is roughly the same as the runtime of $\mathcal{A}$. The *Chall* just samples the challenge uniformly at random and returns it after checking the abort condition. After $\mathcal{A}$ has provided its solution, adversary $\mathcal{B}$ just does some additions, multiplications, and an inversion, which does not add much to its runtime.

This proves theorem 5.6. □

By combining the loss of advantage during all of the proofs above, combined with the loss introduced by EdDSA', a proof for theorem 5.1 and 5.2 is obtained.

6 The Security of EdDSA in a Multi-User Setting

Now that the single-user security of EdDSA got analyzed, we can take a look at its multi-user security. A common approach for Schnorr-like signature schemes is to show it using the random self-reducibility property of the canonical identification scheme as done in [12]. This approach does not work with the EdDSA signature scheme as the underlying identification scheme does not have this random self-reducibility property, since the reduction is not able to rerandomize a public key in a way preserving the distribution of the key generation algorithm. This is due to the fact that valid secret scalar always has to have the n-th bit set.

Therefore, a similar approach to the proof in the single-user setting is used. It is not possible to reduce onto the Ed-DLog problem directly since the adversary gets multiple public keys and therefore might not provide a representation of the commitment looking like $R = r_1 B + r_2 A$, which was needed for the discrete logarithm of the public key to be calculated. For this reason a variant of the one-more discrete logarithm assumption (OMDL) has to be used, as introduced in [24].

The proof starts by showing that the N-MU-EUF-NMA security of EdDSA implies N-MU-SUF-CMA security of EdDSA in the random oracle model. Next an intermediate game is introduced onto which the N-MU-EUF-NMA security of EdDSA is reduced. At last, the security of the intermediate game is reduced onto the security of the variant of the one-more discrete logarithm assumption.

The two main theorems for the multi-user security of EdDSA_{sp} and EdDSA_{lp} are:

Theorem 6.1 (Security of EdDSA with strict parsing in the multi-user setting)**.** *Let $\mathcal{A}$ be an adversary against the N-MU-SUF-CMA security of EdDSA with strict parsing, receiving N public keys and making at most q_h hash queries and q_o oracle queries, and $\mathbb{G}$ be a group of prime order L. Then,*

$$Adv^{N\text{-}MU\text{-}SUF\text{-}CMA}_{\mathbb{G},\mathcal{A}}(\lambda) \leq Adv^{N\text{-}Ed\text{-}DLog\text{-}Reveal}_{E,n,c,L,\mathcal{B}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+N)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}$$

Theorem 6.2 (Security of EdDSA with lax parsing in the multi-user setting)**.** *Let $\mathcal{A}$ be an adversary against the N-MU-EUF-CMA security of EdDSA with lax parsing, receiving N public keys and making at most q_h hash queries and q_o oracle queries, and $\mathbb{G}$ be a group of prime order L. Then,*

$$Adv^{N\text{-}MU\text{-}EUF\text{-}CMA}_{\mathbb{G},\mathcal{A}}(\lambda) \leq Adv^{N\text{-}Ed\text{-}DLog\text{-}Reveal}_{E,n,c,L,\mathcal{B}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+N)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}$$

The chain of reductions can be depicted as:

$$N\text{-Ed-DLog-Reveal} \overset{\text{AGM}}{\Rightarrow} \text{Ed-IDLOG} \overset{\text{ROM}}{\Rightarrow} N\text{-MU-EUF-NMA} \overset{\text{ROM}}{\Rightarrow} N\text{-MU-SUF-CMA}_{\text{EdDSA sp}}$$

$$N\text{-Ed-DLog-Reveal} \overset{\text{AGM}}{\Rightarrow} \text{Ed-IDLOG} \overset{\text{ROM}}{\Rightarrow} N\text{-MU-EUF-NMA} \overset{\text{ROM}}{\Rightarrow} N\text{-MU-EUF-CMA}_{\text{EdDSA lp}}$$

6.1 N-MU-EUF-NMA $\overset{\text{ROM}}{\Rightarrow}$ N-MU-SUF-CMA$_{\text{EdDSA sp}}$

This section shows that the N-MU-EUF-NMA security of the EdDSA signature scheme implies the N-MU-SUF-CMA security of the EdDSA signature scheme using the random oracle model. The section starts with providing an intuition of the proof, followed by the detailed security proof.

Theorem 6.3. *Let n and N be positive integer and $\mathcal{A}$ an adversary against N-MU-SUF-CMA, receiving N public keys and making at most q_h hash queries and q_o oracle queries. Then,*

$$Adv^{N\text{-}MU\text{-}SUF\text{-}CMA}_{\mathcal{A}}(\lambda) \leq Adv^{N\text{-}MU\text{-}EUF\text{-}NMA}_{\mathcal{B}}(\lambda) + \frac{q_o q_h\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}.$$

Proof Overview This proof closely follows the proof in section 5.1. The only difference of both security notions is the absence of the *Sign* oracle in N-MU-EUF-NMA. For this reason, the reduction must simulate the *Sign* oracle without the knowledge of the private keys. This is achieved by generating a valid and well-distributed tuple of commitment, challenge, and response using the *Sim* procedure, introduced in section 5.1, and then programming the random oracle to output that challenge for the corresponding input. The different games are shown in figure 18.

Game G_0 / G_1 / G_2 / G_3

for $j \in \{1, 2, ..., N\}$
  $(h_{j_0}, h_{j_1}, ..., h_{j_{2b-1}}) \leftarrow \{0,1\}^{2b}$
  $s_j \leftarrow 2^n + \sum_{i=c}^{n-1} 2^i h_{j_i}$
  $A_j := s_j B$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{H(\cdot), \mathsf{Sign}(\cdot,\cdot)}(A_1, A_2, ..., A_N)$
return $\exists j \in \{1, 2, ..., N\}$: $\mathsf{Verify}(A_j, m^*, \sigma^*) \wedge (A_j, m^*, \sigma^*) \notin \mathbf{Q}$

Oracle $H(m \in \{0,1\}^*)$
if $\sum[m] = \perp$ **then**
  $\sum[m] \leftarrow \{0,1\}^{2b}$
return $\sum[m]$

Oracle Sign $(j \in \{1, 2, ..., N\}, m \in \mathcal{M})$
// $G_0 - G_2$
$(r'_0, r'_1, ..., r'_{2b-1}) = RF(h_{j_b}|...|h_{j_{2b-1}}|m)$
$r := \sum_{i=0}^{2b-1} 2^i r'_i$
$R := rB$
$S := (r + sH(\underline{R}|\underline{A_j}|m)) \pmod{L}$ // G_0
if $\sum[\underline{R}|\underline{A_j}|m] \neq \perp$ **then** // $G_1 - G_2$
  $bad := true$
  $abort$ // G_2
if $\sum[\underline{R}|\underline{A_j}|m] = \perp$ **then**
  $\sum[\underline{R}|\underline{A_j}|m] \leftarrow \{0,1\}^{2b}$
$S := (r + s\sum[\underline{R}|\underline{A_j}|m]) \pmod{L}$
$\sigma := (\underline{R}, S)$
$\mathbf{Q} := \mathbf{Q} \cup \{(A_j, m, \sigma)\}$
return σ

Oracle Sign $(j \in \{1, 2, ..., N\}, m \in \mathcal{M})$ // G_3
$(R, \mathbf{ch}, S) \leftarrow Sim(A_j)$
if $\sum[\underline{R}|\underline{A_j}|m] \neq \perp$ **then**
  $bad := true$
  $abort$
$\sum[\underline{R}|\underline{A_j}|m] = \mathbf{ch}$
$\sigma := (\underline{R}, S)$
$\mathbf{Q} := \mathbf{Q} \cup \{(A_j, m, \sigma)\}$
return σ

Figure 18: Games $G_0 - G_3$

Formal Proof

Proof.

Now the original N-MU-SUF-CMA game is manipulated in a way that makes it possible to simulate signatures without the knowledge of the secret key. During each of the game-hops the probability for an adversary to detect this change is upper bounded.

$\underline{G_0:}$ Let G_0 be defined in figure 18 by excluding all boxes except the black one. G_0 is the N-MU-SUF-CMA for EdDSA. By definition,

$$\mathrm{Adv}^{N\text{-MU-SUF-CMA}}_{\mathrm{EdDSA},\mathcal{A}}(\lambda) = \Pr[N\text{-MU-SUF-CMA}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_1:}$ G_1 now is defined by replacing the black box with the blue one. This change inlines the call to the hash function and introduces a bad flag, which is set if the hash value is already set. The bad flag being set represents cases where the adversary already queried the random oracle for the challenge used for that signature and therefore the random oracle cannot be programmed. This results in the challenger not being able to produce a valid signature. This change is only conceptual, as it does not alter the behavior of the oracle. Therefore,

$$\Pr[G_0^{\mathcal{A}} \Rightarrow 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_2:}$ G_2 is defined by also introducing the abort instruction in the red box. Again, without loss of generality it is assumed that the adversary only queried each public key/message pair only once since the signatures are deterministic and the attacker would not gain any additional information by querying the *Sign* oracle multiple times with the same input. Since the commitment R is the only unknown input to the hash function, the probability of the bad flag being set for each individual *Sign* query is at most $\frac{q_h}{2^{-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})}}$. By the Union bound over all oracle queries q_o we obtain $\Pr[bad] \leq \frac{q_o q_h}{2^{-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})}}$. Since G_1 and G_2 are identical-until-bad games, we have

$$|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[bad] \leq \frac{q_o q_h \lceil \frac{2^{2b}-1}{L} \rceil}{2^{2b}}.$$

$\underline{G_3:}$ In G_3 the *Sign* oracle is replaced by the *Sign* oracle in the green box. Instead of calculating the response using the secret key, the *Sim* algorithm is used to generate a tuple of commitment, challenge, and response. Then the random oracle is programmed to output the specific challenge given $\underline{R}|\underline{A_j}|m$ as an input. This change is only conceptual, since *Sim* outputs a correctly distributed set and it was ruled out in earlier games that the random oracle was previously queries with this input. Hence,

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \Pr[G_3^{\mathcal{A}} \Rightarrow 1].$$

Finally, Game G_3 is well prepared to show that there exists an adversary $\mathcal{B}$ satisfying

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \mathrm{Adv}_{\mathcal{B}}^{N\text{-MU-EUF-NMA}}(\lambda). \tag{6.5}$$

Adversary $\mathcal{B}^{H(\cdot)}(A_1, A_2, ..., A_N)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{H'(\cdot),\mathsf{Sign}(\cdot,\cdot)}(A_1, A_2, ..., A_N)$
return (m^*, σ^*)

Oracle Sign $(j \in \{1, 2, ..., N\},\ m \in \mathcal{M})$
$(R, \mathbf{ch}, S) \leftarrow Sim(A_j)$
if $\sum[\underline{R}|\underline{A_j}|m] \neq \perp$ **then**
  $bad \coloneqq true$
  $abort$
$\sum[\underline{R}|\underline{A_j}|m] = \mathbf{ch}$
$\sigma \coloneqq (\underline{R}, S)$
$\mathbf{Q} \coloneqq \mathbf{Q} \cup \{(A_j, m, \sigma)\}$
return σ

Oracle $H'(m \in \{0,1\}^*)$
if $\sum[m] = \perp$ **then**
  $\sum[m] \coloneqq H(m)$
return $\sum[m]$

Figure 19: Adversary $\mathcal{B}$ breaking N-MU-EUF-NMA

To prove (6.5), we define an adversary $\mathcal{B}$ attacking N-MU-EUF-NMA that simulates $\mathcal{A}$'s view in G_2. Adversary $\mathcal{B}$ formally defined in figure 19 is run in the N-MU-EUF-NMA game and adversary $\mathcal{B}$ simulates *Sign* for adversary $\mathcal{A}$. *Sign* is simulated perfectly.

Finally, consider $\mathcal{A}$ output $(m^*, \sigma^* \coloneqq (\underline{R^*}, S^*))$. Every valid signature outputted by adversary $\mathcal{A}$ has to fulfill the following equation for one public key A_i: $2^c SB = 2^c R + 2^c H'(\underline{R}|\underline{A_i}|m)A_i$. Again there is only one valid encoded S for each R, m, A_i tuple that satisfies the verification equation. For the signature to be a valid forgery it must not be outputted by the *Sign* oracle for this specific m^* and A_i. No new valid signature can be generated from a valid one by just changing the S value. This means that either R, m or A_i have to be changed to generate a new valid signature from an already valid signature. Since all these parameters are part of the hash query to generate the challenge the resulting hash value has to be forwarded from the H hash oracle provided to the adversary $\mathcal{B}$. For this reason $H'(\underline{R^*}|\underline{A_i}|m^*) = H(\underline{R^*}|\underline{A_i}|m^*)$. Hence,

$$\begin{aligned} 2^c S^* B &= 2^c R^* + 2^c H'(\underline{R^*}|\underline{A_i}|m^*)A_i \\ \Leftrightarrow 2^c S^* B &= 2^c R^* + 2^c H(\underline{R^*}|\underline{A_i}|m^*)A_i \end{aligned}$$

Since the public keys and the results of the hash queries are forwarded from the N-MU-EUF-NMA challenger the forged signature from $\mathcal{A}$ in the N-MU-SUF-CMA game is also a valid forgery for the N-MU-EUF-NMA challenger.

In the main procedure the adversary $\mathcal{B}$ simply calls adversary $\mathcal{A}$ and outputs its forged signature. To simulate the hash function $\mathcal{B}$ simply forwards the queries to adversary $\mathcal{A}$ and to a signature $\mathcal{B}$ obtains the pair of commitment, challenge, and solution from the *Sim* procedure, which is

just samples two values and calculates the last one using a simple equation, and then programs its random oracle. Therefore, the runtime of adversary $\mathcal{B}$ is roughly the same as the runtime of adversary $\mathcal{A}$.

This proves theorem 6.3. □

6.2 N-MU-EUF-NMA $\overset{\text{ROM}}{\Rightarrow}$ N-MU-EUF-CMA$_{\text{EdDSA lp}}$

This section shows that N-MU-EUF-NMA security of EdDSA implies the N-MU-EUF-CMA security of EdDSA with lax parsing used in the random oracle model. This proof is very similar to the proof N-MU-SUF-CMA proof of EdDSA with strict parsing. The modification to the games are the same as in the proof above with the only modifications being in the win condition, which is $\exists j \in \{1, 2, ..., N\} : \mathsf{Verify}(A_j, m^*) \wedge (A_j, m^*) \notin \mathbf{Q}$. For this reason this proof starts at showing the existence of an adversary $\mathcal{B}$ breaking N-MU-EUF-NMA security. Similar to the proof in the single-user setting, the SUF-CMA security of EdDSA with lax parsing cannot be shown, as there are multiple valid encodings of S for one signature. This way the adversary would be able to generate a new valid signature from an obtained one by simply choosing a different encoding of S. This would result in the output of $H'(\underline{R^*}|\underline{A}|m^*)$ being programmed by the reduction itself and therefore the signature not being valid for the EUF-NMA challenger.

Theorem 6.4. *Let n and N be positive integers and $\mathcal{A}$ an adversary against N-MU-EUF-CMA, receiving N public keys and making at most q_h hash queries and q_o oracle queries. Then,*

$$Adv_{\mathcal{A}}^{N\text{-}MU\text{-}EUF\text{-}CMA}(\lambda) \leq Adv_{\mathcal{B}}^{N\text{-}MU\text{-}EUF\text{-}NMA}(\lambda) + \frac{q_o q_h \lceil \frac{2^{2b}-1}{L} \rceil}{2^{2b}}.$$

Formal Proof

Proof.

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \mathrm{Adv}_{\mathcal{B}}^{N\text{-MU-EUF-NMA}}(\lambda). \tag{6.6}$$

To prove (6.6), we define an adversary $\mathcal{B}$ attacking N-MU-EUF-NMA that simulates $\mathcal{A}$'s view on G_3. Adversary $\mathcal{B}$, formally defined in figure 20, is run in the N-MU-EUF-NMA game and simulates *Sign* for adversary $\mathcal{A}$. *Sign* is simulated perfectly.

Finally, consider $\mathcal{A}$ output $(m^*, \sigma^* := (\underline{R^*}, S^*))$. Every valid signature outputted by adversary $\mathcal{A}$ has to fulfill the following equation for one public key A_i: $2^c SB = 2^c R + 2^c H'(\underline{R}|\underline{A_i}|m)A_i$. Like in the single-user setting the adversary can create a new valid signature from an already valid one by choosing a different bitstring representation of the S value that maps to the same $S \pmod L$. Since we are in the N-MU-EUF-CMA setting the adversary has to forge a signature for a message m^* and public key A_i to which it has not queried a signature before. For this reason, the output of $H'(\underline{R^*}|\underline{A_i}|m^*)$ has not been set by the adversary $\mathcal{B}$, but was forwarded from the H hash oracle provided by the N-MU-EUF-NMA challenger. For this reason $H'(\underline{R^*}|\underline{A_i}|m^*) = H(\underline{R^*}|\underline{A_i}|m^*)$. Therefore,

Adversary $\mathcal{B}^{H(\cdot)}(A_1, A_2, ..., A_N)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{H'(\cdot),\mathsf{Sign}(\cdot,\cdot)}(A_1, A_2, ..., A_N)$
return (m^*, σ^*)

Oracle $H'(m \in \{0,1\}^*)$
if $\sum[m] = \perp$ **then**
  $\sum[m] \coloneqq H(m)$
return $\sum[m]$

Oracle Sign $(j \in \{1, 2, ..., N\},\ m \in \mathcal{M})$
$(R, \mathbf{ch}, S) \leftarrow Sim(A_j)$
if $\sum[\underline{R}|\underline{A_j}|m] \neq \perp$ **then**
  $bad \coloneqq true$
  $abort$
$\sum[\underline{R}|\underline{A_j}|m] = \mathbf{ch}$
$\sigma \coloneqq (\underline{R}, S)$
$\mathbf{Q} \coloneqq \mathbf{Q} \cup \{(A_j, m)\}$
return σ

Figure 20: Adversary $\mathcal{B}$ breaking N-MU-EUF-NMA

$$2^c S^* B = 2^c R^* + 2^c H'(\underline{R^*}|\underline{A_i}|m^*)A_i$$
$$\Leftrightarrow 2^c S^* B = 2^c R^* + 2^c H(\underline{R^*}|\underline{A_i}|m^*)A_i.$$

This shows that the forged signature from adversary $\mathcal{A}$ is also a valid forged signature for the N-MU-EUF-NMA challenger.

Since the adversary $\mathcal{B}$ is the same as in the proof above, its runtime is roughly the same as the runtime of adversary $\mathcal{A}$, for the same reason.

This proves theorem 6.4.

□

6.3 N-MU-Ed-IDLOG $\overset{\text{ROM}}{\Rightarrow}$ N-MU-EUF-NMA

This section shows that N-MU-Ed-IDLOG implies N-MU-EUF-NMA security of the EdDSA signature scheme using the Random Oracle Model. The section starts by first providing an intuition of the proof followed by the detailed security proof.

Introducing N-MU-Ed-IDLOG This game follows closely the definition of the Ed-IDLOG game. It again replaces the random oracle with the *Chall* oracle. The only difference to the Ed-IDLOG game is that the adversary gets access to N public keys. The adversary again has to output a valid result for any commitment challenge pair generated by the *Chall* oracle for any of the public keys. The N-MU-Ed-IDLOG game is depicted in figure 21.

Definition 6.1 (N-MU-Ed-IDLOG)**.** *Let n and N be positive integers. For an adversary $\mathcal{A}$, receiving N public keys as input, we define its advantage in the N-MU-Ed-IDLOG as following:*

$$Adv_{\mathcal{A}}^{N\text{-}MU\text{-}Ed\text{-}IDLOG}(\lambda) \coloneqq |\Pr[N\text{-}MU\text{-}Ed\text{-}IDLOG^{\mathcal{A}} \Rightarrow 1]|.$$

Game N-MU-Ed-IDLOG
for $i \in \{1, 2, ..., N\}$
  $a_i \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$
  $A_i \coloneqq a_i B$
$s^* \leftarrow \mathcal{A}^{Chall(\cdot)}(A_1, A_2, ..., A_N)$
return $\exists (R^*, \mathbf{ch}^*) \in \mathbf{Q}, i \in \{1, 2, ..., N\} \in : R^* = 2^c s^* B - 2^c \mathbf{ch}^* A_i$

Oracle *Chall* $(R_i \in \mathbb{G})$
$\mathbf{ch}_i \leftarrow \{0,1\}^{2b}$
$\mathbf{Q} \coloneqq \mathbf{Q} \cup \{(R_i, \mathbf{ch}_i)\}$
return $\mathbf{ch}_i$

Figure 21: N-MU-Ed-IDLOG

Theorem 6.5. *Let $\mathcal{A}$ be an adversary against N-MU-Ed-IDLOG. Then,*

$$Adv_{\mathcal{A}}^{N\text{-}MU\text{-}EUF\text{-}NMA}(\lambda) = Adv_{\mathcal{B}}^{N\text{-}MU\text{-}Ed\text{-}IDLOG}(\lambda).$$

Proof Overview Like the single-user setting the adversary has to query the random oracle to get the hash value $H(\underline{R}|\underline{A_i}|m)$. Again the programmability of the random oracle can be used to embed the challenge from *Chall* oracle into the answer of the random oracle. By embedding the challenge from the *Chall* oracle answer into the answer of the random oracle, a valid forgery of the signature also becomes a valid solution for the N-MU-Ed-IDLOG game.

Game G_0
for $i \in \{1, 2, ..., N\}$
  $(h_{i_0}, h_{i_1}, ..., h_{i_{2b-1}}) \leftarrow \{0,1\}^{2b}$
  $s_i \leftarrow 2^n + \sum_{i=c}^{n-1} 2^i h_i$
  $A_i \coloneqq s_i B$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{H(\cdot)}(A_1, A_2, ..., A_N)$
return $\exists i \in \{1, 2, ..., N\}$:
$\mathsf{Verify}(A_i, m^*, \sigma^*)$

Oracle $H(m \in \{0,1\}^*)$
if $\sum[m] = \perp$ **then**
  $\sum[m] \leftarrow \{0,1\}^{2b}$
return $\sum[m]$

Figure 22: G_0

Formal Proof

Proof.

Now it is argued that the *Chall* oracle can be used to simulate the hash function in a way that the answer of the N-MU-EUF-NMA adversary can be used as an valid solution for the N-MU-Ed-IDLOG challenger.

Let G_0 be defined in figure 22. Then G_0 is the same as N-MU-EUF-NMA with EdDSA. By definition,

$$\mathrm{Adv}^{N\text{-MU-EUF-NMA}}_{\mathrm{EdDSA},\mathcal{A}}(\lambda) = \Pr[N\text{-MU-EUF-NMA}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

G_0 is well-prepared to show that there exists an adversary $\mathcal{B}$ satisfying

$$\Pr[G_0^{\mathcal{A}} \Rightarrow 1] = \mathrm{Adv}^{N\text{-MU-Ed-IDLOG}}_{\mathbb{G},\mathcal{B}}(\lambda). \tag{6.7}$$

Adversary $\mathcal{B}^{Chall(\cdot)}(A_1, A_2, ..., A_N)$
$(m^*, \sigma^* := (\underline{R}, S)) \leftarrow \mathcal{A}^{H(\cdot)}(A_1, A_2, ..., A_N)$
return S

Oracle $H(m \in \{0,1\}^*)$
if $\sum[m] = \perp$ **then**
  if $\underline{R}|\underline{A}|m' := m \wedge R, A \in E$ **then**
    $\sum[m] \leftarrow Chall(2^c R)$
  else
    $\sum[m] \leftarrow \{0,1\}^{2b}$
return $\sum[m]$

Figure 23: Adversary $\mathcal{B}$ breaking Ed-IDLOG

To proof (6.7), we define an adversary $\mathcal{B}$ attacking N-MU-Ed-IDLOG that simulates $\mathcal{A}$'s view in G_0. Adversary $\mathcal{B}$ formally defined in figure 23 is run in the Ed-IDLOG game and adversary $\mathcal{B}$ simulates the random oracle H for the adversary $\mathcal{A}$. H is perfectly simulated because the *Chall* oracle also outputs a uniformly random 2b-bit bitstring. For this reason, H returns a uniformly random 2b-bit bitstring for all queries, as expected.

Finally, consider $\mathcal{A}$'s output $(m^*, \sigma^* := (\underline{R}, S))$. It is known that:

$$\begin{aligned} 2^c SB &= 2^c R + 2^c H(\underline{R}|\underline{A_i}|m)A_i \\ \Leftrightarrow 2^c R &= 2^c SB - 2^c H(\underline{R}|\underline{A_i}|m)A_i \\ \Leftrightarrow 2^c R &= 2^c SB - 2^c Chall(2^c R)A_i \\ R' &= 2^c SB - 2^c Chall(R')A_i \end{aligned}$$

Therefore, S is a valid solution for the N-MU-Ed-IDLOG game.

This proves theorem 6.5. □

6.4 N-Ed-DLog-Reveal $\overset{\mathsf{AGM}}{\Rightarrow}$ N-MU-Ed-IDLOG

This section shows that N-Ed-DLog-Reveal implies N-MU-Ed-IDLOG using the algebraic group model. The section starts by introducing a special variant of the one-more discrete logarithm problem followed by an intuition of the proof and at last giving a detailed security proof. The reduction cannot be directly performed using the Ed-DLog assumption, since the representation of the commitment contains more than one group element with unknown discrete

logarithm, because the adversary against N-MU-Ed-IDLOG receives multiple public keys as input. Therefore, a new assumption, based on the one-more discrete logarithm assumption, has to be introduced.

Introducing N-Ed-DLog-Reveal Similar to Ed-DLog, which is a variant of the discrete logarithm problem the N-Ed-DLog-Reveal is a variant of the one-more discrete logarithm problem, which represents the special distribution of secret keys resulting from the key generation algorithm of the EdDSA signature scheme. The only differences to the original one-more discrete logarithm game as introduced in [24] are that the secret scalars are chosen from the set $\{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$ which represents all valid secret scalars regarding the key generation algorithm and that the adversary is only able to query $N - 1$ discrete logarithms of the challenge group elements at once. This modification makes the assumption weaker than the original one-more discrete logarithm assumption. Since the resulting game is similar to the N-discrete logarithm problem with an additinal *Reveal* query, it is called N-Ed-DLog-Reveal. A lower bound on the hardness of the N-Ed-DLog-Reveal problem is further analyzed in section 7.2. The N-Ed-DLog-Reveal game is illustrated in figure 24.

Definition 6.2 (N-Ed-DLog-Reveal)**.** *Let n and N be positive integers. For an adversary $\mathcal{A}$, receiving N challenge group elements, we define its advantage in the N-Ed-DLog-Reveal game as following:*

$$Adv_{\mathcal{A}}^{N\text{-}Ed\text{-}DLog\text{-}Reveal}(\lambda) \coloneqq |\Pr[N\text{-}Ed\text{-}DLog\text{-}Reveal^{\mathcal{A}} \Rightarrow 1]|.$$

Game N-Ed-DLog-Reveal
for $i \in \{1, 2, ..., N\}$
  $a_i \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$
  $A_i \coloneqq a_i B$
$(a'_1, a'_2, ..., a'_N) \leftarrow \mathcal{A}^{Reveal(\cdot)}(A_1, A_2, ..., A_N)$
return $(a_1, a_2, ..., a_N) \stackrel{?}{=} (a'_1, a'_2, ..., a'_N)$

Oracle $Reveal(j \in \{1, 2, ..., N\})$ // max. one query
return $\{a_i | i \in \{1, 2, ..., N\} \backslash \{j\}\}$

Figure 24: N-Ed-DLog-Reveal

Theorem 6.6. *Let $\mathcal{A}$ be an adversary against Ed-IDLOG with $\mathbb{G}$ being a cyclic group of prime order L, receiving N public keys and making at most q_o oracle queries. Then*

$$Adv_{\mathbb{G},\mathcal{A}}^{N\text{-}MU\text{-}Ed\text{-}IDLOG}(\lambda) \leq Adv_{\mathbb{G},\mathcal{B}}^{N\text{-}Ed\text{-}DLog\text{-}Reveal}(\lambda) + \frac{q_o N \lceil \frac{2^{2b}-1}{L} \rceil}{2^{2b}}.$$

Proof Overview In the multi-user setting the adversary gets access to not only the generator B and one public key A but rather a set of public keys A_1 to A_N. For this reason, the representation of a group element, the adversary has to provide, looks the following: $R = r_1 B + r_2 A_1 + ... + r_{N+1} A_N$. Since there are multiple group elements with unknown discrete

logarithms it is not possible to directly calculate the discrete logarithm of one of the public keys given a valid forgery of a signature. Upon receiving a valid solution the *Reveal* oracle can be used to get the discrete logarithm of all the public keys except the one for which the solution is valid. This way it is again possible to construct a representation looking like $R = r_1B + r_2A_i$. Then it is again possible to calculate the discrete logarithm of A_i and win the N-Ed-DLog-Reveal game.

Game G_0 / G_1 / G_2
for $i \in \{1, 2, ..., N\}$
  $a_i \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$
  $A_i := a_iB$
$s^* \leftarrow \mathcal{A}^{Chall(\cdot)}(A_1, A_2, ..., A_N)$
return $\exists(R^*, \mathbf{ch}^*) \in \mathbf{Q}, i \in \{1, 2, ..., N\} : R^* = 2^c s^* B - 2^c \mathbf{ch}^* A_i$

Oracle *Chall* $([R]_{\vec{r}} \in \mathbb{G})$
Let $R = r_1B + r_2A_1 + ... + r_{N+1}A_N$
$\mathbf{ch} \leftarrow \{0,1\}^{2b}$
If $\exists i \in \{2, 3, ..., N+1\} : 2^c\mathbf{ch} \equiv -r_i \pmod{L}$ **then** // $G_1 - G_2$
  bad := *true*
  abort // G_2
$\mathbf{Q} := \mathbf{Q} \cup \{(R, \mathbf{ch})\}$
return ch

Figure 25: Games $G_0 - G_2$

Formal Proof

Proof.

Now the individual game-hops are analyzed and the probability, that an adversary can distinguish between two games, is upper bounded.

G_0**:** Let G_0 be defined in figure 25 by excluding all boxes. Clearly, G_0 is the N-MU-Ed-IDLOG. By definition,

$$\mathrm{Adv}^{N\text{-MU-Ed-IDLOG}}_{\mathbb{G},\mathcal{A}}(\lambda) = \Pr[N\text{-MU-Ed-IDLOG}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

G_1**:** G_1 is defined by including the if condition in the blue box, which sets a bad flag if the randomly chosen value $\mathbf{ch}$ fulfills $2^c\mathbf{ch} \equiv -r_i \pmod{L}$ for any $i \in \{2, 3, ..., N+1\}$. This represents challenges $\mathbf{ch}$ to which the solution might not be usable to break the discrete logarithm of one of the public keys, due to $(r_i + 2^c\mathbf{ch})$ not being invertible in $\mathbb{Z}_L$. Since only the bad flag is introduced this change does not influence the behavior of the game and is therefore only conceptual.

$$\Pr[G_0^{\mathcal{A}} \Rightarrow 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_2:}$ G_2 also includes the abort instruction in the red box. The abort is triggered if the bad flag is set to true. For each individual *Chall* oracle query the bad flag is set with a probability of $\frac{N}{2^{-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})}}$. With $2^{-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})}$ being the min-entropy of $\mathbf{ch}$ and N being the number of r_i with which the equation $2^c\mathbf{ch} \equiv -r_i \pmod{L}$ could evaluate to true. By the Union bound over all q_o oracle quries we obtain $\Pr[bad] \leq \frac{q_o N}{2^{-\log_2(\lceil \frac{2^{2b}-1}{L} \rceil 2^{-2b})}}$. Since G_1 and G_2 are identical-until-bad games, we have

$$|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[bad] \leq \frac{q_o N \lceil \frac{2^{2b}-1}{L} \rceil}{2^{2b}}.$$

Finally, Game G_2 is well-prepared to show that there exists an adversary $\mathcal{B}$ satisfying

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \mathrm{Adv}^{N\text{-Ed-DLog-Reveal}}_{\mathbb{G},\mathcal{B}}(\lambda). \tag{6.8}$$

Adversary $\mathcal{B}^{Reveal(\cdot)}(A_1, A_2, ..., A_N)$
$s^* \leftarrow \mathcal{A}^{Chall(\cdot)}(A_1, A_2, ..., A_N)$
If $\nexists([R^*]_{\vec{r^*}}, \mathbf{ch}^*) \in Q, i \in \{1, 2, ..., N\} : R^* = 2^c s^* B - 2^c \mathbf{ch}^* A_i$ **then**
  abort
Let $R^* = r_1^* B + r_2^* A_1 + ... + r_{N+1}^* A_N$
$r_b \coloneqq r_1$
$(a_1, ..., a_{i-1}, a_{i+1}, ..., a_N) \leftarrow Reveal(i)$
for $j \in \{1, 2, ..., N\} \backslash \{i\}$
  $r_b \coloneqq r_b + r_{j+1} a_j$ // $A_j = a_j B$
$a_i \coloneqq (2^c s^* - r_b)(r_i + 2^c \mathbf{ch}^*)^{-1}$ // $R^* = r_b B + r_i A_i$
return $(a_1, a_2, ..., a_N)$

Oracle *Chall* $([R]_{\vec{r}} \in \mathbb{G})$
Let $R = r_1 B + r_2 A_1 + ... + r_{N+1} A_N$
$\mathbf{ch} \leftarrow \{0,1\}^{2b}$
If $\exists i \in \{2, 3, ..., N+1\} : 2^c \mathbf{ch} \equiv -r_i \pmod{L}$ **then**
  $bad \coloneqq true$
  abort
$\mathbf{Q} \coloneqq \mathbf{Q} \cup \{(R, \mathbf{ch})\}$
return ch

Figure 26: Adversary $\mathcal{B}$ breaking N-Ed-DLog-Reveal

To prove (6.8), we define an adversary $\mathcal{B}$ attacking N-Ed-DLog-Reveal that simulates $\mathcal{A}$'s view in G_2. Adversary $\mathcal{B}$ formally defined in figure 26 is run in the N-Ed-DLog-Reveal game and adversary $\mathcal{B}$ simulates *Chall* for adversary $\mathcal{A}$. *Chall* is simulated perfectly.

Finally, consider $\mathcal{A}$'s output s^*. It is known that $R^* = 2^c s^* B - 2^c \mathbf{ch}^* A_i$ for one of the public keys and one tuple $(R^*, \mathbf{ch}^*)$ generated by the *Chall* oracle. Using the *Reveal* oracle we can get the discrete logarithms of all public keys but the one, for which s^* is a valid solution in the N-MU-Ed-IDLOG game. Together with the representation of R^*, provided during the *Chall* oracle call, and the discrete logarithms of the public keys we are able to generate a representation of R^*, which looks like $R^* = r_b B + r_i A_i$. By equating both equations we get:

$$\begin{aligned} r_b B + r_i A_i &= 2^c s^* B - 2^c \mathbf{ch}^* A_i \\ \Leftrightarrow (r_i + 2^c \mathbf{ch}^*) A &= (2^c s^* - r_b) B \\ \Leftrightarrow A &= (2^c s^* - r_b)(r_i + 2^c \mathbf{ch}^*)^{-1} B \end{aligned}$$

Assuming that $r_i + 2^c \mathbf{ch}^*$ is invertible in $\mathbb{Z}_L$ (i.e., not equal to 0), which is ensured by the abort in G_2 for all i, both equations can be used to calculate the discrete logarithm of A_i. Together with the discrete logarithms of the other public keys, which were obtained by the *Reveal* oracle, the adversary $\mathcal{B}$ is able to craft a valid solution for the N-Ed-DLog-Reveal challenger.

The runtime of adversary $\mathcal{B}$ is roughly the same as the runtime of adversary $\mathcal{A}$. In the main procedure the adversary $\mathcal{B}$ calls adversary $\mathcal{A}$, queries the *Reveal* oracle and performs some simple calculations to obtain the discrete logarithm of all public keys. In the *Chall* the adversary simply samples a 2b bitstring uniformly at random.

This proves theorem 6.6. □

By combining the loss of advantage during all of the proofs above, combined with the loss introduced by EdDSA', a proof for theorem 6.1 and 6.2 is obtained.

7 The Ed-GGM

The following section gives specific bounds on the difficulty of certain variations of the discrete logarithm and one-more discrete logarithm problems introduced in the previous proofs. These proofs are given in the generic group model. In the generic group model, group elements are represented as random bitstrings, and the adversary can only perform group operations by invoking an oracle.

In order to build a generic group model for twisted Edwards curves, it is essential to examine the group structure. As shown in section 5.4, a twisted Edwards curve can be uniquely decomposed into a collection of cyclic subgroups. The generating set for this twisted Edwards curve is defined as a set of generators for these cyclic subgroups. With a fixed generating set, any point on the twisted Edwards curve can be uniquely expressed as a linear combination of the generators in that set. Consequently, the adversary is given labels of the entire generator set as a description of the twisted Edwards curve. In addition, the adversary has access to a group operation oracle, GOp, which, given two labels and a bit indicating whether the group elements should be added or subtracted, returns the label of the resulting group element.

The labels are bitstrings of length $\lceil \log_2(L) \rceil$, with L being the order of the group.

7.1 Bounds on Ed-DLog

This section focuses on establishing a lower bound on the hardness of a modified version of the discrete logarithm problem in the generic group model. This variant is introduced in definition 5.2 and works similarly to the original discrete logarithm problem, except for the secret scalar generation, which is derived from the key generation algorithm of the EdDSA signature scheme. The following proof is given in the generic group model for twisted Edwards curves.

Theorem 7.1. *Let n and c be positive integers. Consider a twisted Edwards curve E with a cofactor of 2^c and a generating set consisting of $(B, E_2, ..., E_m)$. Among these, let B be the generator of the largest prime order subgroup with an order of L. Let $\mathcal{A}$ be a generic adversary making at most q_g group operations. Then,*

$$Adv^{Ed\text{-}DLog}_{E,n,c,L,\mathcal{A}} \leq \frac{(q_g + 3)^2 + 1}{2^{n-1-c}}.$$

Proof Overview This proof closely resembles the original proof on the lower bound for the discrete logarithm problem by Shoup [28]. The initial step involves working with the discrete logarithms of group elements rather than the actual group elements themselves. In the generic group model, this is equivalent as each group element can be uniquely represented by its discrete logarithms with respect to a generating set. For consistency the generating set is denoted as $(B, E_2, ..., E_m)$, with B being the generator of the prime order subgroup and E_2 to E_m being the generators of the other subgroups. Subsequently, the discrete logarithm in the prime order subgroup is replaced by an indeterminate. By doing this, the discrete logarithm in the prime order subgroup can be chosen after the adversary has provided their solution. As a result, the generic adversary can only guess the discrete logarithm in the prime order subgroup, since it is generated only after the adversary has already submitted their solution. Figure 27 shows the Ed-DLog game in the generic group model.

The following proof utilizes the Schwartz-Zippel lemma [33]. The Schwarz-Zippel lemma is defined as following:

Lemma 7.1 (Schwartz-Zippel lemma)**.** *Let L be a prime number and $P \in \mathbb{F}_L[X_1, ..., X_n]$ be a non-zero polynomial of total degree $d \geq 0$ over a field $\mathbb{F}_L$. Let S be a finite subset of $\mathbb{F}_L$ and let x be selected uniformly at random from S. Then*

$$\Pr[P(x) = 0] \leq \frac{d}{|S|}.$$

Formal Proof

Proof.

Let G_0 be the Ed-DLog game in the generic group model. In this proof, the discrete logarithm within the prime order subgroup of the group element A will be substituted with an indeterminate. Following that, it will be demonstrated that the challenger, by working with polynomials

Game Ed-DLog
$a \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$
$A \coloneqq aB$
$a^* \leftarrow \mathcal{A}^{GOp(\cdot,\cdot,\cdot)}(Enc(B), Enc(E_2), ..., Enc(E_m), Enc(A))$
return $a^* \stackrel{?}{=} a$

Oracle GOp$(x, y \in \mathbf{S}, b \in \{0, 1\})$
return $Enc(\sum^{-1}[x] + (-1)^b \sum^{-1}[y])$

Procedure Enc$(X \in E)$
If $\sum[X] = \perp$ **then**
  $\sum[X] \leftarrow \{0,1\}^{\lceil log_2(|E|)\rceil} \backslash \mathbf{S}$
  $\mathbf{S} \coloneqq \mathbf{S} \cup \{\sum[X]\}$
return $\sum[X]$

Figure 27: Ed-DLog in the generic group model

rather than actual discrete logarithms, makes errors in the simulation with negligible probability. Finally, it will be established that the discrete logarithm of the group element A can be selected after the adversary has submitted its solution for the game.

G_0 : Let G_0 be defined in figure 28 by excluding all boxes except the black ones. This is identical to the Ed-DLog in the generic group model. By definition,

$$\mathrm{Adv}^{\text{Ed-DLog}}_{E,n,c,L,\mathcal{A}} = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

G_1 : G_1 is defined by substituting some black boxes with blue ones, causing the challenger to work with discrete logarithms rather than group elements. This modification is undetectable to the adversary, as it only deals with labels representing group elements, and each group element can be uniquely represented by its discrete logarithms. These discrete logarithms are denoted by an integer vector, where each element corresponds to the discrete logarithm concerning the generator in the generating set. The addition of these vectors is carried out component-wise. This change remains conceptual, since it only changes how the challenger internally represents group elements. Each group element is still assigned the same label. Therefore,

$$\Pr[G_0^{\mathcal{A}} \Rightarrow 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1].$$

G_2 : In G_2, the blue boxes are replaced with the red ones, which involves replacing the discrete logarithm of group elements in the prime order subgroup with a polynomial. The polynomial has one indeterminant, denoted by Z, which represents the discrete logarithm of group element in the prime order subgroup, provided to the adversary as a challenge. Therefore, the polynomial

Game G_0 / G_1 /G_2 / G_3 / G_4

$a \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$ // $G_0 - G_4$

$A \coloneqq aB$ // G_0

$A \coloneqq (a, 0, ..., 0)$ // G_1

$P \coloneqq Z$ // $G_2 - G_4$
$A \coloneqq (P, 0, ..., 0)$

$a^* \leftarrow \mathcal{A}^{GOp(\cdot,\cdot,\cdot)}(Enc(B), Enc(E_2), ..., Enc(E_m), Enc(A))$

return $a^* \stackrel{?}{=} a$

Oracle $\mathrm{GOp}(x, y \in \mathbf{S}, b \in \{0, 1\})$

return $Enc(\sum^{-1}[x] + (-1)^b \sum^{-1}[y])$

Procedure $\mathrm{Enc}(X \in E)$ // G_0

Procedure $\mathrm{Enc}(X \in \mathbb{Z}_L \times \mathbb{Z}_{ord(E_2)} \times ... \times \mathbb{Z}_{ord(E_n)})$ // G_1

Procedure $\mathrm{Enc}(X \in \mathbb{Z}_L[Z] \times \mathbb{Z}_{ord(E_2)} \times ... \times \mathbb{Z}_{ord(E_n)})$ // $G_2 - G_4$
Let $X = (P, x_2, ..., x_n)$
$\mathbf{P} = \mathbf{P} \cup \{P\}$

if $\exists P_i \in \mathbf{P} : P_i(a) = P(a) \wedge P_i \neq P$ // $G_3 - G_4$
$bad \coloneqq true$
abort // G_4

$X \coloneqq (P(a), x_2, ..., x_n)$ // $G_2 - G_4$

If $\sum[X] = \perp$ **then**
$\sum[X] \leftarrow \{0, 1\}^{\lceil log_2(|E|) \rceil} \backslash \mathbf{S}$
$\mathbf{S} \coloneqq \mathbf{S} \cup \{\sum[X]\}$

return $\sum[X]$

Figure 28: $G_0 - G_4$

that serves as the discrete logarithm of the challenge in the prime order subgroup is simply $P = Z$. It is important to note that this change is only conceptual since the polynomial is ultimately evaluated at the secret scalar a in the Enc procedure. Hence,

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G_2^{\mathcal{A}} \Rightarrow 1].$$

G_3 : G_3 introduces the if condition within the green box. This condition checks if the challenger generated two distinct polynomials that would produce the same value when evaluated at a. This condition ensures that polynomials can be directly compared later on, rather than needing to evaluate them. If the if condition evaluates to true, a bad flag is set to true, indicating that the challenger might incorrectly assume that two discrete logarithms, represented by the polynomials, are different by only comparing the polynomials. This modification is purely

Game G_4 / G_5 /G_6 / G_7

$a \leftarrow \{2^{n-1}, 2^{n-1}+2^c, ..., 2^n - 2^c\}$ // $G_4 - G_6$
$P := Z$
$A := (P, 0, ..., 0)$
$a^* \leftarrow \mathcal{A}^{GOp(\cdot,\cdot,\cdot)}(Enc(B), Enc(E_2), ..., Enc(E_m), Enc(A))$
$a \leftarrow \{2^{n-1}, 2^{n-1}+8, ..., 2^n - 8\}$ // G_7
if $\exists P_i, P_j \in \mathbf{P} : P_i(a) = P_j(a) \wedge P_i \neq P$ // $G_6 - G_7$
  $bad := true$
  abort
return $a^* \stackrel{?}{=} a$

Oracle GOp$(x, y \in \mathbf{S}, b \in \{0,1\})$
  return $Enc(\sum^{-1}[x] + (-1)^b \sum^{-1}[y])$

Procedure Enc$(X \in \mathbb{Z}_L[Z] \times \mathbb{Z}_{ord(E_2)} \times ... \times \mathbb{Z}_{ord(E_n)})$
Let $X = (P, x_2, ..., x_n)$
$\mathbf{P} = \mathbf{P} \cup \{P\}$
if $\exists P_i \in \mathbf{P} : P_i(a) = P(a) \wedge P_i \neq P$ // $G_4 - G_5$
  $bad := true$
  abort
$X := (P(a), x_2, ..., x_n)$ // G_4
If $\sum[X] = \perp$ **then**
  $\sum[X] \leftarrow \{0,1\}^{\lceil log_2(|E|)\rceil} \backslash \mathbf{S}$
  $\mathbf{S} := \mathbf{S} \cup \{\sum[X]\}$
return $\sum[X]$

Figure 29: $G_4 - G_7$

conceptual, as it only affects internal variables and does not influence the game's behavior. Therefore,

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G_2^{\mathcal{A}} \Rightarrow 1].$$

G_4 : G_4 terminates if the bad flag defined in the previous game is set. This bad flag signifies situations where collisions of discrete logarithms would not be identified by merely comparing polynomials without evaluating them. The likelihood of the bad flag being set can be determined using the Schwartz-Zippel lemma. The set $\mathbf{P}$ is a set of all polynomials generated by the challenger and the polynomial P represents the newly generated one. During the encoding of a newly generated group element the challenger checks that no two distinct polynomials evaluate to the same value at a. For a fixed $P_i \in \mathbf{P} \neq P$ we define $P^* = P_i - P$. If and only if $P_i(a) = P(a)$ then $P^*(a) = 0$. Since $P^* \neq 0$, the degree of P^* being 1 and a being chosen uniformly at random from $\{2^{n-1}, 2^{n-1}+2^c, ..., 2^n - 2^c\}$ the Schwarz-Zippel lemma can be used to calculate the probability that $P^*(a) = 0$, which is $\Pr[P^*(a) = 0] \leq \frac{1}{2^{n-1-c}}$. Since the set $\mathbf{P}$

can hold at most $q_g + 3$ many polynomials (one per call to the group operation oracle GOp, and three by encoding the input to the adversary) by the Union bound over all polynomials in $\mathbf{P}$ the probability of bad being set, for each individual oracle query, is less or equal to $\frac{q_g+3}{2^{n-1-c}}$. By the Union bound over all oracle queries the probability of bad being set to true is $\Pr[bad] \leq \frac{(q_g+3)^2}{2^{n-1-c}}$. For this reason,

$$|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[bad] \leq \frac{(q_g+3)^2}{2^{n-1-c}}.$$

For improved readability, G_4 is also depicted in figure 29 by including only the black boxes and excluding all others. The subsequent game-hops are also illustrated in the same figure.

$\underline{G_5:}$ G_5 removes the evaluation of the polynomial in the Enc procedure. This alteration is purely conceptual, as the previous abort condition ensured that no two distinct polynomials would yield the same value upon evaluation. Consequently, it is feasible to work directly with the polynomials rather than evaluating them.

$$\Pr[G_4^{\mathcal{A}} \Rightarrow 1] = \Pr[G_5^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_6:}$ The difference in G_6 is that the abort condition was moved into the main game after the adversary provided its solution. Because of this change the secret scalar a is not being used before the adversary provided its solution to the challenger. Therefore, the secret scalar a can be chosen after the adversary provided its solution, which means that it has no better chance to guess the solution. To demonstrate that this alteration is solely conceptual, it will be proven that G_6 aborts if and only if G_5 would do the same.

G_5 aborts $\Rightarrow$ G_6 aborts: If G_5 aborts, it means that a polynomial P_i has been added to the set $\mathbf{P}$ during the call to the Enc procedure, which satisfies the abort condition. In G_6, the polynomials in the set $\mathbf{S}$ remain the same, since the instruction for adding polynomials to the set during the Enc procedure has not been altered between the games. After the adversary provides its solution, the challenger checks for any pair of polynomials in the set that meet the abort condition. Thus, G_6 will abort if G_5 would have aborted.

G_6 aborts $\Rightarrow$ G_5 aborts: If G_6 were to abort, the set $\mathbf{P}$ would contain a pair of polynomials that satisfy the abort condition. The distinction between G_6 and G_5 is that G_5 checks for the existence of such a pair immediately after inserting a new polynomial. Consequently, if G_6 were to abort, G_5 would also abort.

This proves that this change is only conceptual. Hence,

$$\Pr[G_5^{\mathcal{A}} \Rightarrow 1] = \Pr[G_6^{\mathcal{A}} \Rightarrow 1].$$

G_7 : The generation of the secret scalar a in G_7 occurs after the adversary has provided its solution. This modification is purely conceptual, as the secret scalar is not utilized prior to this point. Thus,

$$\Pr[G_6^{\mathcal{A}} \Rightarrow 1] = \Pr[G_7^{\mathcal{A}} \Rightarrow 1].$$

As a result, the adversary has no improved likelihood of computing its solution a^* other than guessing, given that the challenger does not select a until the adversary has submitted its solution. Since a being chosen uniformly at random from the set $\{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$ the probability for the adversary to win G_7 is:

$$\Pr[G_7^{\mathcal{A}} \Rightarrow 1] \leq \frac{1}{2^{n-1-c}}.$$

This proves theorem 7.1. □

7.2 Bounds on N-Ed-DLog-Reveal

This section provides a lower bound on the hardness of the modified version of the one-more discrete logarithm problem in the generic group model. The variant of the one-more discrete logarithm problem was introduced in the definition 6.2. N-Ed-DLog-Reveal differs from the original one-more discrete logarithm problem by only allowing the adversary to query the discrete logarithm of all challenges but one. Also the discrete logarithms of the group elements in the challenge to the adversary are chosen from a predefined set that is the result of the special key generation algorithm used in EdDSA. The following proof uses the generic group model for twisted Edwards curves. There already exists a proof for the one-more discrete logarithm problem in the generic group model [34]. This proof provides a lower bound on the original definition of the one-more discrete logarithm problem. This proof is not directly applicable to this definition of N-Ed-DLog-Reveal, since the secret scalars are not chosen uniformly at random from $\mathbb{Z}_L$ and the group structure is not just a prime order group. Since a more restricted version of the one-more discrete logarithm problem is used a simpler proof than that in [34] can be used, providing a better bound on N-Ed-DLog-Reveal.

Theorem 7.2. *Let n, N, c be positive integers. Consider a twisted Edwards curve E with a cofactor of 2^c and a generating set consisting of $(B, E_2, ..., E_m)$. Among these, let B be the generator of the largest prime order subgroup with an order of L. Let $\mathcal{A}$ be a generic adversary against N-Ed-DLog-Reveal receiving N group elements as challenge and making at most q_g group operations queries. Then,*

$$Adv_{E,n,c,L,\mathcal{A}}^{N\text{-}Ed\text{-}DLog\text{-}Reveal} \leq \frac{2(q_g + N + 2)^2 + 1}{2^{n-1-c}}.$$

Proof Overview This proof uses the same approach as the discrete logarithm proof in the generic group model by replacing the group elements with polynomials and choosing the challenge after the adversary provided its solution. The tricky part is that the adversary is able to query the discrete logarithms of all but one of the N group elements, provided to it as a

challenge. The proof starts by replacing all group elements with multivariate polynomials representing their discrete logarithms. The indeterminants of those polynomials are the discrete logarithms of each group element, provided to the adversary as challenges. Once the adversary requests the discrete logarithms for all but one group element of the challenge those discrete logarithms are chosen uniformly at random and all polynomials are partially evaluated. This leaves polynomials with just one indeterminate, representing the discrete logarithm of the last challenge. This challenge is then chosen after the adversary provided its solution, leaving the adversay no option but to guess the remaining discrete logarithm. The N-Ed-DLog-Reveal game in the generic group model is depicted in figure 30.

Game N-Ed-DLog-Reveal
for $i \in \{1, 2, ..., N\}$
  $a_i \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$
  $A_i \coloneqq a_i B$
$(a'_1, a'_2, ..., a'_N) \leftarrow \mathcal{A}^{GOp(\cdot,\cdot,\cdot), Reveal(\cdot)}(Enc(B), Enc(E_2), ..., Enc(E_m), Enc(A_1), ..., Enc(A_N))$
return $(a_1, a_2, ..., a_N) \stackrel{?}{=} (a'_1, a'_2, ..., a'_N)$

Oracle *Reveal*$(j \in \{1, 2, ..., N\})$ // max. one query
return $\{a_i | i \in \{1, 2, ..., N\} \backslash \{j\}\}$

Oracle GOp$(x, y \in \mathbf{S}, b \in \{0, 1\})$
  return $Enc(\sum^{-1}[x] + (-1)^b \sum^{-1}[y])$

Procedure Enc$(X \in E)$
If $\sum[X] = \perp$ **then**
  $\sum[X] \leftarrow \{0,1\}^{\lceil log_2(|E|) \rceil} \backslash \mathbf{S}$
  $\mathbf{S} \coloneqq \mathbf{S} \cup \{\sum[X]\}$
return $\sum[X]$

Figure 30: N-Ed-DLog-Reveal in the generic group model

Game G_0 / G_1 /G_2 / G_3 / G_4

for $i \in \{1, 2, ..., N\}$

$a_i \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$ // $G_0 - G_4$

$A_i := a_i B$ // G_0

$A_i := (a_i, 0, ..., 0)$ // G_1

$P_i := Z_i$ // $G_2 - G_4$

$A_i := (P_i, 0, ..., 0)$

$(a'_1, a'_2, ..., a'_N) \leftarrow \mathcal{A}^{GOp(\cdot,\cdot,\cdot), Reveal(\cdot)}(Enc(B), Enc(E_2), ..., Enc(E_m), Enc(A_1), ..., Enc(A_N))$

return $(a_1, a_2, ..., a_N) \stackrel{?}{=} (a'_1, a'_2, ..., a'_N)$

Oracle $Reveal(j \in \{1, 2, ..., N\})$

for $P_i \in \mathbf{P}$ // $G_3 - G_4$

Let $P_i = R_i + S_i, R_i \in \mathbb{Z}_L[Z_1, ..., Z_{j-1}, Z_{j+1}, ..., Z_N], S_i \in \mathbb{Z}_L[Z_j]$

$\mathbf{R} := \mathbf{R} \cup \{R_i\}$

if $\exists R_i, R_j \in \mathbf{R} : R_i(\vec{a}) = R_j(\vec{a}) \wedge R_i \neq R_j$

$bad_1 := true$

abort // G_4

for $P_i \in \mathbf{P}$

$\sum[R_i(\vec{a}) + S_i] = \sum[P_i]$

$P_i := R_i(\vec{a}) + S_i$

return $\{a_i | i \in \{1, 2, ..., N\} \backslash \{j\}\}$

Oracle $\mathrm{GOp}(x, y \in \mathbf{S}, b \in \{0, 1\})$

return $Enc(\sum^{-1}[x] + (-1)^b \sum^{-1}[y])$

Procedure $\mathrm{Enc}(X \in E)$ // G_0

Procedure $\mathrm{Enc}(X \in \mathbb{Z}_L \times \mathbb{Z}_{ord(E_2)} \times ... \times \mathbb{Z}_{ord(E_n)})$ // G_1

Procedure $\mathrm{Enc}(X \in \mathbb{Z}_L[Z_1, ..., Z_N] \times \mathbb{Z}_{ord(E_2)} \times ... \times \mathbb{Z}_{ord(E_n)})$ // $G_2 - G_4$

Let $X = (P, x_2, ..., x_n)$

$\mathbf{P} = \mathbf{P} \cup \{P\}$

$X := (P(\vec{a}), x_2, ..., x_n)$

If $\sum[X] = \perp$ **then**

$\sum[X] \leftarrow \{0, 1\}^{\lceil log_2(|E|) \rceil} \backslash \mathbf{S}$

$\mathbf{S} := \mathbf{S} \cup \{\sum[X]\}$

return $\sum[X]$

Figure 31: $G_0 - G_4$

Game G_4 / G_5 /G_6 / G_7 / G_8
for $i \in \{1, 2, ..., N\}$
$a_i \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$ // $G_4 - G_7$
$P_i := Z_i$
$A_i := (P_i, 0, ..., 0)$
$(a'_1, a'_2, ..., a'_N) \leftarrow \mathcal{A}^{GOp(\cdot,\cdot,\cdot), Reveal(\cdot)}(Enc(B), Enc(E_2), ..., Enc(E_m), Enc(A_1), ..., Enc(A_N))$
for $i \in \{1, 2, ..., N\}$ // G_8
if $a_i = \perp$
$a_i \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$
if $\exists P_i, P_j \in \mathbf{P} : P_i(\vec{a}) = P_j(\vec{a}) \wedge P_i \neq P_j$ // $G_5 - G_8$
$bad_2 := true$
abort // $G_6 - G_8$
return $(a_1, a_2, ..., a_N) \stackrel{?}{=} (a'_1, a'_2, ..., a'_N)$

Oracle $Reveal(j \in \{1, 2, ..., N\})$
for $i \in \{1, 2, ..., N\} \backslash \{j\}$ // G_8
$a_i \leftarrow \{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$
for $P_i \in \mathbf{P}$
Let $P_i = R_i + S_i, R_i \in \mathbb{Z}_L[Z_1, ..., Z_{j-1}, Z_{j+1}, ..., Z_N], S_i \in \mathbb{Z}_L[Z_j]$
$\mathbf{R} := \mathbf{R} \cup \{R_i\}$
if $\exists R_i, R_j \in \mathbf{R} : R_i(\vec{a}) = R_j(\vec{a}) \wedge R_i \neq R_j$
$bad_1 := true$
abort
for $P_i \in \mathbf{P}$
$\sum[R_i(\vec{a}) + S_i] = \sum[P_i]$
$P_i := R_i(\vec{a}) + S_i$
return $\{a_i | i \in \{1, 2, ..., N\} \backslash \{j\}\}$

Oracle $\mathrm{GOp}(x, y \in \mathbf{S}, b \in \{0, 1\})$
return $Enc(\sum^{-1}[x] + (-1)^b \sum^{-1}[y])$

Procedure $\mathrm{Enc}(X \in \mathbb{Z}_L[Z_1, ..., Z_N] \times \mathbb{Z}_{ord(E_2)} \times ... \times \mathbb{Z}_{ord(E_n)})$
Let $X = (P, x_2, ..., x_n)$
$\mathbf{P} = \mathbf{P} \cup \{P\}$
$X := (P(\vec{a}), x_2, ..., x_n)$ // $G_4 - G_6$
If $\sum[X] = \perp$ **then**
$\sum[X] \leftarrow \{0, 1\}^{\lceil log_2(|E|) \rceil} \backslash \mathbf{S}$
$\mathbf{S} := \mathbf{S} \cup \{\sum[X]\}$
return $\sum[X]$

Figure 32: $G_4 - G_8$

Formal Proof

Proof.

The proof starts by replacing group elements with polynomials. This happens in games G_1 and G_2. After that it is argued that the challenger makes a mistake in its simulation with only negligible probability by comparing polynomials directly instead of evaluating them. This is shown in $G_3 - G_6$. At last, since the polynomials are not evaluated during the simulation, one discrete logarithm is not used before the adversary provides its solution. Therefore, it can be chosen after the adversary provided its solution, which is shown in G_7 and G_8.

$\underline{G_0:}$ Let G_0 be depicted in figure 31 by excluding all boxes but the black ones. Clearly, this is equivalent to the N-Ed-DLog-Reveal game in the generic group model. Therefore,

$$\mathrm{Adv}^{N\text{-Ed-DLog-Reveal}}_{E,n,c,L,\mathcal{A}} = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_1:}$ G_1 now replaces the group elements in the challenger with their discrete logarithms. This change is purely conceptual, since the adversary only sees the labels of the group elements, and each group element can be uniquely identified by its discrete logarithm. As in the Ed-DLog proof, the discrete logarithm of a group element is denoted by an integer vector, where each element in the vector represents the discrete logarithm with respect to a generator from the generating set. For this reason,

$$\Pr[G_0^{\mathcal{A}} \Rightarrow 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_2:}$ G_2 replaces the blue boxes with the red ones. This change affects the discrete logarithm of the group elements in the prime order subgroup. The discrete logarithm is now represented as a multivariate polynomial. Each indeterminate of the polynomial represents the discrete logarithm of one of the group elements in the challenge to the adversary. The discrete logarithm of the group element in the challenge to the adversary is then instantiated with the indeterminate representing the discrete logarithm of that challenge, instead of the discrete logarithm itself. This change is only conceptual, since the polynomials are evaluated with the discrete logarithm vector of the group elements in the challenge before being compared in the Enc procedure. Hence,

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G_2^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_3:}$ G_3 introduces the bad_1 flag in the *Reveal* query. Without loss of generality the following explanation assumes that the adversary queries the *Reveal* oracle with input $j = N$. Each polynomial, generated by the challenger, is a linear multivariate polynomial of degree one. This is due to the fact that the challenger starts with linear multivariate polynomials of degree one in $\mathbb{Z}_L[Z_1, ..., Z_N]$ and only adds them to generate new polynomials. This means that each

polynomial $P_i \in \mathbb{Z}_L[Z_1, ..., Z_N]$, generated by the challenger, can be split into two polynomials $R_i \in \mathbb{Z}_L[Z_1, ..., Z_{N-1}], S_i \in \mathbb{Z}_L[Z_N]$ so that $P_i = R_i + S_i$, simply by distributing the monials between the polynomials R_i and S_i. The polynom S_i only contains the monial Z_n, while the polynom R_i contains the remaining monials and the constant. Now the polynomial P_i can be partially evaluated by setting $P_i = R_i(\vec{a}) + S_i$. For the simulation to be correct, when replacing the polynomial P_i with $R_i(\vec{a}) + S_i$, it has to be ensured that distinct polynomials stay distinct after being partially evaluated. To ensure this, it is necessary to check that no two distinct polynomials R_i, R_j result in the same value when evaluated with $\vec{a}$. In the case of this happening the bad_1 flag is set to true. Afterward, each generated polynomial is partially evaluated as described and the table $\sum$, which stores the association between group elements and labels, is updated to reflect this partial evaluation as well. From now on, each polynomial used by the challenger is in $\mathbb{Z}_L[Z_N]$. This change is purely conceptual, since the polynomials still get fully evaluated before being compared in the Enc procedure. Therefore,

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \Pr[G_3^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_4:}$ In G_4 the abort instruction in the orange box is introduced, which is executed after the bad_1 flag is set. The bad_1 flag is set if distinct polynomials result in the same polynomial, after being partially evaluated. To calculate the probability of this happening the Schwart-Zippel lemma can be utilized. For every $R_i, R_j \in \mathbf{R} \wedge R_i \neq R_j$ a polynomial $R^* := R_i - R_j$ can be constructed. If and only if $R_i(\vec{a}) = R_j(\vec{a})$ then $R^*(\vec{a}) = 0$. Since $R^* \neq 0$, the degree of R^* being 1 and $\vec{a}$ being chosen uniformly at random from $\{2^{n-1}, 2^{n-1} + 2^c, ..., 2^n - 2^c\}$ the Schwartz-Zippel lemma can be used to calculate the probability of $R^*(\vec{a}) = 0$, which is $\Pr[R^*(\vec{a}) = 0] \leq \frac{1}{2^{n-1-c}}$. The challenger can generate at most $q_g + N + 2$ many polynomials, one per group operation query GOp and $N + 2$ for encoding the input to the adversary. By the Union bound over all $(q_g + N + 2)^2$ possible pairs of polynomials an upper bound on the bad_1 flag being set can be calculated as $\Pr[bad_1] \leq \frac{(q_g+N+2)^2}{2^{n-1-c}}$. Since G_3 and G_4 are identical-until-bad games,

$$|\Pr[G_3^{\mathcal{A}} \Rightarrow 1] - \Pr[G_4^{\mathcal{A}} \Rightarrow 1]| \leq \frac{(q_g + N + 2)^2}{2^{n-1-c}}.$$

To improve the readability, G_4 is also depicted in figure 29 by only including the black boxes. The following game-hops are illustrated in the same figure.

$\underline{G_5:}$ G_5 introduces the check in the blue box. This check ensures that after the adversary provided its solution no two distinct polynomials where generated by the challenger that evaluate to the same value, when evaluated with the vector of discrete logarithms. If this happens the bad_2 flag is set. This change is only conceptual, as it only changes internal variables, which have no effect on the behavior of the challenger. Hence,

$$\Pr[G_4^{\mathcal{A}} \Rightarrow 1] = \Pr[G_5^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_6:}$ G_6 aborts if the bad_2 flag is set. The bad_2 flag is set if any two distinct polynomials evaluate to the same value, when evaluated with the vector of discrete logarithms. There are two cases. The first case is that the adversary has queried the *Reveal* oracle. The second case is that the adversary did not queried the *Reveal* oracle.

In the first case the adversary got the discrete logarithms of all but one challenge. Without loss of generality it is assumed that the adversary queried the discrete logarithm of all but the Nth group element. In this case all polynomials in $\mathbf{P}$ are in $\mathbb{Z}_L[Z_N]$, since at the time of the *Reveal* query all polynomials, generated up to this point, are partially evaluated and are in $\mathbb{Z}_Z[Z_N]$. All polynomials that are generated after this point are generated by the addition of the existing polynomials and are therefore also in $\mathbb{Z}_L[Z_N]$. In this case the Schwartz-Zippel lemma can be applied since the adversary has no information on the remaining discrete logarithm. This is the same scenario as in the Ed-DLog proof.

In the case where the adversary did not queried the *Reveal* oracle the adversary has no information on any of the discrete logarithms. All polynomials in $\mathbf{P}$ are in $\mathbb{Z}_Z[N_1, ..., Z_N]$. In this case the Schwartz-Zippel lemma can be applied, since the all discrete logarithms are chosen uniformly at random and the adversary has no information on them, prior to them being chosen.

The probability of bad_2 being true can be calculated using the Schwartz-Zippel lemma, as described in the game-hop to G_4. With the Union bound over all polynomial pairs in $\mathbf{P}$ the probability of bad_2 being true is $\Pr[bad_2] \leq \frac{(q_g+N+2)^2}{2^{n-1-c}}$. G_5 and G_6 are identical-until-bad games, therefore:

$$|\Pr[G_5^{\mathcal{A}} \Rightarrow 1] - \Pr[G_6^{\mathcal{A}} \Rightarrow 1]| \leq \frac{(q_g + N + 2)^2}{2^{n-1-c}}.$$

$\underline{G_7:}$ G_7 removes the evaluation of polynomials in the Enc procedure. It is argued that this change is only conceptual. When the evaluation of polynomials is removed, the polynomials are compared directly. Group elements represented by different polynomials are assigned different labels by the challenger. This is equivalent to the original definition as long as different polynomials do not evaluate to the same value, when evaluated with the discrete logarithms. This inconsistency in the simulation can be detected by the adversary when it gets some information on the discrete logarithms. This can either be during the query to the *Reveal* oracle or after the adversary provided its solution. In both cases there is an if condition checking for this inconsistency. If such an inconsistency is detected the game aborts. This change is only conceptual, since the different polynomials correspond to different group elements, in the cases where the game does not abort, and since the adversary only sees the labels it cannot detect whether the challenger works with polynomials or concrete discrete logarithms. Hence,

$$\Pr[G_6^{\mathcal{A}} \Rightarrow 1] = \Pr[G_7^{\mathcal{A}} \Rightarrow 1].$$

$\underline{G_8:}$ In G_8 the discrete logarithms of the challenge are only generated right before they are used. Since the discrete logarithms are not used during the Enc function anymore they the challenger can generate them not at the start of the game but only right before they are used. The discrete logarithms are only used during the inconsistency checks in the *Reveal* oracle or after the adversary has provided its solution. $N-1$ discrete logarithms are used in the *Reveal* oracle to check for inconsistencies and to partially evaluate the polynomials. After the adversary provided its solution the remaining discrete logarithms can chosen to fully evaluate all polynomials. This can be either all discrete logarithm, in the case that the adversary did not queried the *Reveal* oracle, or the remaining one, in the case that the adversary did queried the *Reveal* oracle. This change is only conceptual, since the initialization of variables is only moved right before the variable is used. Therefore,

$$\Pr[G_7^{\mathcal{A}} \Rightarrow 1] = \Pr[G_8^{\mathcal{A}} \Rightarrow 1].$$

Since at least one discrete logarithm is chosen after the adversary provided its solution, its only chance is to guess it. Therefore, the probability of the adversary of winning G_7 is upper bounded by the probability of it guessing that discrete logarithm. Hence,

$$\Pr[G_8^{\mathcal{A}} \Rightarrow 1] \leq \frac{1}{2^{n-1-c}}.$$

This proves theorem 7.2. □

8 Concrete Security of EdDSA

Now that a security bound on the complexity of an adversary breaking EdDSA has been established the concrete security of the signature scheme can be analyzed. The security level of a cryptographic scheme can be determined by analysing the success ratio of an adversary. The success ratio of an attacker can be determined by analyzing its success probability and its runtime. The success ratio is simply the advantage of an adversary devided by its runtime. This follows the approach for concrete security of Bellare and Ristenpart [35] but the definition of success ratio and bit security is taken from [36].

Definition 8.1 (Success Ratio [36])**.** *Let adversary $\mathcal{A}$ be an adversary with runtime Time($\mathcal{A}$) and advantage $Adv_{\mathcal{A}}$. Its success ratio is defined as following:*

$$SR(\mathcal{A}) = \frac{Adv_{\mathcal{A}}}{Time(\mathcal{A})}.$$

With this definition of the success ratio the bit security of a cryptographic scheme can be defined.

Definition 8.2 (Bit Security [36])**.** *A cryptographic scheme has κ bit security if the success ratio of all adversaries with a runtime $Time(\mathcal{A}) \leq 2^{\kappa}$ is upper bounded by $2^{-\kappa}$.*

This definition can be used to calculate the bit security of concrete instantiations of EdDSA. The most popular instantations of EdDSA are Ed25519 and Ed443, as they are also specified in the RFC and the NIST standard.

8.1 Ed25519

Theorem 8.1 (Ed25519 Bit Security)**.** *The Ed25519 signature scheme provides 125-bit security in the single-user setting and 124-bit security in the multi-user setting against generic adversaries.*

Ed25519 is one of the most widely used instantiations of EdDSA. According to the RFC it is supposed to provide around 128-bit of security. It uses the twisted Edwards curve Ed25519 and SHA-512 as a hash function [3] [4]. This provides the following values, needed to calculate the security level of Ed25519 according to the security proof in this thesis:

Parameter	**Value**
b	256
n	254
c	3
L	$2^{252} + 27742317777372353535851937790883648493$

Table 2: Parameter of Ed25519

Proof.

At first the runtime of the adversaries against Ed25519 in the single user setting is analyzed. This can be done by analyzing the runtime of an adversary $\mathcal{B}$ against Ed-DLog, since the runtime of both adversaries is roughly the same. The success probability of an adversary $\mathcal{B}$ in the Ed-DLog game is $\text{Adv}^{\text{Ed-DLog}}_{E,n,c,L,\mathcal{B}} \leq \frac{(q_g+3)^2+1}{2^{n-1-c}}$. When instantiated with the values for Ed25519, an adversary $\mathcal{B}$ is able to solve the Ed-DLog game with constant probability after about 2^{125} group operations. Therefore, the runtime of the adversary $\mathcal{B}$ in the Ed-DLog game can be upper bounded by 2^{125}. The runtime of an adversary $\mathcal{A}$ against Ed25519 is roughly the same as the adversary $\mathcal{B}$ against Ed-DLog and can therefore also be upper bounded by 2^{125}. This, together with the advantage of adversary $\mathcal{A}$, can be used to upper bound its success ratio.

Since the runtime of the adversary is upper bounded by 2^{125} the amount of hash quries q_h and group operations q_g can also be upper bounded by 2^{125}. A reasonable upper bound for the signing quries q_o is 2^{64}, as they are online and can not be computed by the adversary in secret. This provides following equation for the success ratio:

$$
\begin{aligned}
SR(\mathcal{A}) &\leq \frac{\mathrm{Adv}^{\text{SUF-CMA}}_{\mathbb{G},\mathcal{A}}(\lambda)}{Time(\mathcal{A})} \\
&\leq \frac{\mathrm{Adv}^{\text{Ed-DLog}}_{E,n,c,L,\mathcal{B}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+1)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}}{Time(\mathcal{A})} \\
&\leq \frac{\frac{(q_g+3)^2+1}{2^{n-1-c}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+1)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}}{Time(\mathcal{A})} \\
&\leq \frac{(2^{125}+3)^2+1}{2^{250}2^{125}} + \frac{2(2^{125}+1)}{2^{256}2^{125}} + \frac{2^{64}(2^{125}+1)2^{260}}{2^{512}2^{125}} \\
&\approx 2^{-125} + 2^{-316} + 2^{-188} \\
&\approx 2^{-125}
\end{aligned}
$$

This shows that Ed25519 provides 125-bit security in the single-user setting.

To get a security level in the multi-user setting an upper bound on the number of instances N is needed. In [12] Kiltz et al. mentioned that the existance of at lease $N = 2^{30}(\approx 1$ billion) public keys can be assumed. For the following calculations the number of instances is assumed to be $N \leq 2^{35}$. An adversary $\mathcal{B}$ against N-Ed-DLog-Reveal has a constant probability of winning the game after about 2^{125} group operations. Hence, its runtime is upper bounded by 2^{125}. The success ratio can then be calculated in the same way as it has been done in the single-user setting.

This provides a success ratio of:

$$
\begin{aligned}
SR(\mathcal{A}) &\leq \frac{\mathrm{Adv}^{N\text{-MU-SUF-CMA}}_{\mathbb{G},\mathcal{A}}(\lambda)}{Time(\mathcal{A})} \\
&\leq \frac{\mathrm{Adv}^{N\text{-Ed-DLog-Reveal}}_{E,n,c,L,\mathcal{A}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+N)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}}{Time(\mathcal{A})} \\
&\leq \frac{\frac{2(q_g+N+2)^2+1}{2^{n-1-c}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+N)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}}{Time(\mathcal{A})} \\
&\leq \frac{2(2^{125}+2^{35}+2)^2+1}{2^{250}2^{125}} + \frac{2(2^{125}+1)}{2^{256}2^{125}} + \frac{2^{64}(2^{125}+2^{35})2^{260}}{2^{512}2^{125}} \\
&\approx 2^{-124} + 2^{-316} + 2^{-188} \\
&\approx 2^{-124}
\end{aligned}
$$

This shows that Ed25519 provides 124-bit security in the multi-user setting.

This proofs theorem 8.1. □

8.2 Ed448

Theorem 8.2 (Ed448 Bit Security)**.** *The Ed448 signature scheme provides 221-bit security in the single-user setting and 220-bit security in the multi-user setting against generic adversaries.*

Another popular instantiation of the EdDSA signature scheme is Ed448. It uses the Ed448 twisted Edwards curve and SHAKE256 as hash function. It is supposed to provide around 224 bits of security and was also standardized by the IETF and NIST [3] [4]. The respective standards provide following values:

Parameter	**Value**
b	456
n	447
c	2
L	$2^{446}-138180668098951153520073867485154268803366924748821786098945475038 85$

Table 3: Parameter of Ed448

Proof.

This can be used to upper bound the success ratio of an adversary $\mathcal{A}$ against Ed448. To begin, the runtime of an adversary $\mathcal{B}$ against Ed-DLog is upper bounded, using the values from the Ed448 signature scheme. The adversary $\mathcal{B}$ achieves a constant probability of winning the Ed-DLog game after 2^{223} group operations. This also upper bounds its runtime. Now the success ratio of adversary $\mathcal{A}$ against Ed448 can be calculated as following:

$$
\begin{aligned}
SR(\mathcal{A}) &\leq \frac{\mathrm{Adv}^{\text{SUF-CMA}}_{\mathbb{G},\mathcal{A}}(\lambda)}{Time(\mathcal{A})} \\
&\leq \frac{\mathrm{Adv}^{\text{Ed-DLog}}_{E,n,c,L,\mathcal{B}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+1)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}}{Time(\mathcal{A})} \\
&\leq \frac{\frac{(q_g+3)^2+1}{2^{n-1-c}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+1)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}}{Time(\mathcal{A})} \\
&\leq \frac{(2^{223}+3)^2+1}{2^{444}2^{223}} + \frac{2(2^{223}+1)}{2^{456}2^{223}} + \frac{2^{64}(2^{223}+1)2^{466}}{2^{912}2^{223}} \\
&\approx 2^{-221} + 2^{-455} + 2^{-372} \\
&\approx 2^{-221}
\end{aligned}
$$

This shows that Ed448 provides 221-bit security in the single-user setting.

Now the same is done for the multi-user security of Ed448. This yields following upper bound for the success ratio:

$$
\begin{aligned}
SR(\mathcal{A}) &\leq \frac{\mathrm{Adv}^{N\text{-MU-SUF-CMA}}_{\mathbb{G},\mathcal{A}}(\lambda)}{Time(\mathcal{A})} \\
&\leq \frac{\mathrm{Adv}^{N\text{-Ed-DLog-Reveal}}_{E,n,c,L,\mathcal{A}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+N)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}}{Time(\mathcal{A})} \\
&\leq \frac{\frac{2(q_g+N+2)^2+1}{2^{n-1-c}} + \frac{2(q_h+1)}{2^b} + \frac{q_o(q_h+N)\lceil\frac{2^{2b}-1}{L}\rceil}{2^{2b}}}{Time(\mathcal{A})} \\
&\leq \frac{2(2^{223}+2^{35}+2)^2+1}{2^{444}2^{223}} + \frac{2(2^{223}+1)}{2^{456}2^{223}} + \frac{2^{64}(2^{223}+2^{35})2^{466}}{2^{912}2^{223}} \\
&\approx 2^{-220} + 2^{-445} + 2^{-372} \\
&\approx 2^{-220}
\end{aligned}
$$

This shows that Ed448 provides 220-bit security in the multi-user setting.

This proofs theorem 8.2. □

9 Conclusion

In this thesis it has been proven that EdDSA is tightly secure using the algebraic group model and the random oracle model. An algebraic attacker does not gain an advantage by attacking the signature scheme instead of attacking the underlying discrete logarithm problem directly, when taking the clamping of the private key into account. When using strict parsing of signatures the EdDSA signature scheme ensures SUF-CMA security and when using lax parsing the signature scheme still provides EUF-CMA security.

It has also been proven that the most common instantiations Ed25519 and Ed448 provide 125-bit of security and 221-bit of security respectively. This is weaker than the original discrete logarithm problem for the elliptic curves used, but was to be expected considering the clamping of the private key.

Moreover, it has been proven that the signature scheme does not lose much of its security considering a multi-user setting. More specific, with a generous assumption of the existence of $2^{35}(\approx 32$ billion) public keys the scheme loses only one bit of security.

According to the results of this thesis, EdDSA has been proven to be a secure signature scheme and that the modifications done to the original Schnorr signature scheme have very little affect on the security of the signature scheme. In fact, the only noticeable loss in security was introduced by the clamping of the private key.

Acknowledgments Many thanks to Dominik Hartmann and Eike Kiltz for the many discussions that helped me during the creation of this thesis.

References

[1] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," in *Cryptographic Hardware and Embedded Systems – CHES 2011* (B. Preneel and T. Takagi, eds.), vol. 6917 of *Lecture Notes in Computer Science*, (Nara, Japan), pp. 124–142, Springer, Heidelberg, Germany, Sept. 28 – Oct. 1, 2011.

[2] D. J. Bernstein, S. Josefsson, T. Lange, P. Schwabe, and B.-Y. Yang, "EdDSA for more curves." Cryptology ePrint Archive, Report 2015/677, 2015. `https://eprint.iacr.org/2015/677`.

[3] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," Request for Comments RFC 8032, Internet Engineering Task Force, Jan. 2017. Num Pages: 60.

[4] D. Moody, "Digital Signature Standard (DSS)," Tech. Rep. NIST FIPS 186-5, National Institute of Standards and Technology, Gaithersburg, MD, 2023.

[5] M. Bellare, B. Poettering, and D. Stebila, "From identification to signatures, tightly: A framework and generic transforms," in *Advances in Cryptology – ASIACRYPT 2016, Part II* (J. H. Cheon and T. Takagi, eds.), vol. 10032 of *Lecture Notes in Computer Science*, (Hanoi, Vietnam), pp. 435–464, Springer, Heidelberg, Germany, Dec. 4–8, 2016.

[6] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, pp. 161–174, Jan. 1991.

[7] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology – CRYPTO'86* (A. M. Odlyzko, ed.), vol. 263 of *Lecture Notes in Computer Science*, (Santa Barbara, CA, USA), pp. 186–194, Springer, Heidelberg, Germany, Aug. 1987.

[8] J. Brendel, C. Cremers, D. Jackson, and M. Zhao, "The provable security of Ed25519: Theory and practice," in *2021 IEEE Symposium on Security and Privacy*, (San Francisco, CA, USA), pp. 1659–1676, IEEE Computer Society Press, May 24–27, 2021.

[9] S. Chatterjee, A. Menezes, and P. Sarkar, "Another look at tightness," in *SAC 2011: 18th Annual International Workshop on Selected Areas in Cryptography* (A. Miri and S. Vaudenay, eds.), vol. 7118 of *Lecture Notes in Computer Science*, (Toronto, Ontario, Canada), pp. 293–319, Springer, Heidelberg, Germany, Aug. 11–12, 2012.

[10] G. Fuchsbauer, A. Plouviez, and Y. Seurin, "Blind schnorr signatures and signed ElGamal encryption in the algebraic group model," in *Advances in Cryptology – EUROCRYPT 2020, Part II* (A. Canteaut and Y. Ishai, eds.), vol. 12106 of *Lecture Notes in Computer Science*, (Zagreb, Croatia), pp. 63–95, Springer, Heidelberg, Germany, May 10–14, 2020.

[11] D. J. Bernstein, "Multi-user Schnorr security, revisited." Cryptology ePrint Archive, Report 2015/996, 2015. `https://eprint.iacr.org/2015/996`.

[12] E. Kiltz, D. Masny, and J. Pan, "Optimal security proofs for signatures from identification schemes," in *Advances in Cryptology – CRYPTO 2016, Part II* (M. Robshaw and J. Katz, eds.), vol. 9815 of *Lecture Notes in Computer Science*, (Santa Barbara, CA, USA), pp. 33–61, Springer, Heidelberg, Germany, Aug. 14–18, 2016.

[13] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Advances in Cryptology – EUROCRYPT'96* (U. M. Maurer, ed.), vol. 1070 of *Lecture Notes in Computer Science*, (Saragossa, Spain), pp. 387–398, Springer, Heidelberg, Germany, May 12–16, 1996.

[14] P. Paillier and D. Vergnaud, "Discrete-log-based signatures may not be equivalent to discrete log," in *Advances in Cryptology – ASIACRYPT 2005* (B. K. Roy, ed.), vol. 3788 of *Lecture Notes in Computer Science*, (Chennai, India), pp. 1–20, Springer, Heidelberg, Germany, Dec. 4–8, 2005.

[15] S. Garg, R. Bhaskar, and S. V. Lokam, "Improved bounds on security reductions for discrete log based signatures," in *Advances in Cryptology – CRYPTO 2008* (D. Wagner, ed.), vol. 5157 of *Lecture Notes in Computer Science*, (Santa Barbara, CA, USA), pp. 93–107, Springer, Heidelberg, Germany, Aug. 17–21, 2008.

[16] Y. Seurin, "On the exact security of Schnorr-type signatures in the random oracle model," in *Advances in Cryptology – EUROCRYPT 2012* (D. Pointcheval and T. Johansson, eds.), vol. 7237 of *Lecture Notes in Computer Science*, (Cambridge, UK), pp. 554–571, Springer, Heidelberg, Germany, Apr. 15–19, 2012.

[17] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre, "From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security," in *Advances in Cryptology – EUROCRYPT 2002* (L. R. Knudsen, ed.), vol. 2332 of *Lecture Notes in Computer Science*, (Amsterdam, The Netherlands), pp. 418–433, Springer, Heidelberg, Germany, Apr. 28 – May 2, 2002.

[18] K. Ohta and T. Okamoto, "On concrete security treatment of signatures derived from identification," in *Advances in Cryptology – CRYPTO'98* (H. Krawczyk, ed.), vol. 1462 of *Lecture Notes in Computer Science*, (Santa Barbara, CA, USA), pp. 354–369, Springer, Heidelberg, Germany, Aug. 23–27, 1998.

[19] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, pp. 361–396, June 2000.

[20] M. Bellare and A. Palacio, "GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks," in *Advances in Cryptology – CRYPTO 2002* (M. Yung, ed.), vol. 2442 of *Lecture Notes in Computer Science*, (Santa Barbara, CA, USA), pp. 162–177, Springer, Heidelberg, Germany, Aug. 18–22, 2002.

[21] K. Chalkias, F. Garillot, and V. Nikolaenko, "Taming the many EdDSAs." Cryptology ePrint Archive, Report 2020/1244, 2020. `https://eprint.iacr.org/2020/1244`.

[22] S. Galbraith, J. Malone-Lee, and N. P. Smart, "Public key signatures in the multi-user setting," *Information Processing Letters*, vol. 83, pp. 263–266, Sept. 2002.

[23] M. Bellare and P. Rogaway, "The security of triple encryption and a framework for code-based game-playing proofs," in *Advances in Cryptology – EUROCRYPT 2006* (S. Vaudenay, ed.), vol. 4004 of *Lecture Notes in Computer Science*, (St. Petersburg, Russia), pp. 409–426, Springer, Heidelberg, Germany, May 28 – June 1, 2006.

[24] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, "The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme," *Journal of Cryptology*, vol. 16, pp. 185–215, June 2003.

[25] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, "Twisted Edwards curves." Cryptology ePrint Archive, Report 2008/013, 2008. https://eprint.iacr.org/2008/013.

[26] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS 93: 1st Conference on Computer and Communications Security* (D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, eds.), (Fairfax, Virginia, USA), pp. 62–73, ACM Press, Nov. 3–5, 1993.

[27] G. Fuchsbauer, E. Kiltz, and J. Loss, "The algebraic group model and its applications," in *Advances in Cryptology – CRYPTO 2018, Part II* (H. Shacham and A. Boldyreva, eds.), vol. 10992 of *Lecture Notes in Computer Science*, (Santa Barbara, CA, USA), pp. 33–62, Springer, Heidelberg, Germany, Aug. 19–23, 2018.

[28] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Advances in Cryptology – EUROCRYPT'97* (W. Fumy, ed.), vol. 1233 of *Lecture Notes in Computer Science*, (Konstanz, Germany), pp. 256–266, Springer, Heidelberg, Germany, May 11–15, 1997.

[29] U. M. Maurer, "Abstract models of computation in cryptography (invited paper)," in *10th IMA International Conference on Cryptography and Coding* (N. P. Smart, ed.), vol. 3796 of *Lecture Notes in Computer Science*, (Cirencester, UK), pp. 1–12, Springer, Heidelberg, Germany, Dec. 19–21, 2005.

[30] C. Karpfinger and K. Meyberg, "Der Hauptsatz über endliche abelsche Gruppen," in *Algebra: Gruppen - Ringe - Körper* (C. Karpfinger and K. Meyberg, eds.), pp. 143–149, Berlin, Heidelberg: Springer, 2021.

[31] C. Karpfinger and K. Meyberg, "Direkte und semidirekte Produkte," in *Algebra: Gruppen - Ringe - Körper* (C. Karpfinger and K. Meyberg, eds.), pp. 83–102, Berlin, Heidelberg: Springer, 2021.

[32] C. Karpfinger and K. Meyberg, "Die Sätze von Sylow," in *Algebra: Gruppen - Ringe - Körper* (C. Karpfinger and K. Meyberg, eds.), pp. 115–129, Berlin, Heidelberg: Springer, 2021.

[33] J. T. Schwartz, "Fast Probabilistic Algorithms for Verification of Polynomial Identities," *Journal of the ACM*, vol. 27, pp. 701–717, Oct. 1980.

[34] B. Bauer, G. Fuchsbauer, and A. Plouviez, "The one-more discrete logarithm assumption in the generic group model." Cryptology ePrint Archive, Report 2021/866, 2021. https://eprint.iacr.org/2021/866.

[35] M. Bellare and T. Ristenpart, "Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme," in *Advances in Cryptology – EUROCRYPT 2009* (A. Joux, ed.), vol. 5479 of *Lecture Notes in Computer Science*, (Cologne, Germany), pp. 407–424, Springer, Heidelberg, Germany, Apr. 26–30, 2009.

[36] D. Hofheinz, T. Jager, and E. Kiltz, "Short signatures from weaker assumptions," in *Advances in Cryptology – ASIACRYPT 2011* (D. H. Lee and X. Wang, eds.), vol. 7073 of *Lecture Notes in Computer Science*, (Seoul, South Korea), pp. 647–666, Springer, Heidelberg, Germany, Dec. 4–8, 2011.