IACR Communications in Cryptology
Vol. 0, No. 0, 32 pages.

On the use of ECDSA with hierarchical public key delegation in identity-based scenarios

Lucas C. Cardoso and Marcos A. Simplicio Jr

Universidade de São Paulo, São Paulo, Brazil

Abstract. In 2009, Galindo and Garcia (Africacrypt) proposed the use of concatenated Schnorr signatures for hierarchical public key delegation, creating an efficient identity-based signature scheme (IBS). Essentially, the scheme builds upon the Schnorr signature scheme to generate a primary signature, part of which is then used as a secret key to produce signatures on subsequent messages. The resulting IBS is proven secure against existential forgery under adaptive chosen-message attacks in the identity-based setting using variants of the Forking Lemma. In this paper, our goal is to answer the following question: would it be feasible to build upon the widely used elliptic curve digital signature algorithm (ECDSA) scheme to obtain a similarly secure and efficient IBS? We answer this affirmatively, opening interesting possibilities not only for identity-based signatures with ECDSA but also for solutions such as secure credential delegation. This latter application is of particular interest considering the wide support for ECDSA in web- and cloud-oriented authentication systems (e.g., those based on JSON Web Tokens). We prove the security of the resulting scheme by combining the Bijective Random Oracle model and the existential unforgeability game in an identity-based setup. Our results also show that, despite its non-linear nature and more convoluted verification process than Schnorr signatures, ECDSA allows one to obtain signatures of roughly the same size as those of Galindo-Garcia's scheme while maintaining efficient performance.

Keywords: ECDSA · identity-based signatures · secure credential delegation · security proof

1 Introduction

Digital signature schemes based on elliptic curves are quite popular in real-world systems due to their advantages compared to RSA [RSA78], such as smaller keys, shorter signatures, and faster signing procedures [LV01]. The Elliptic Curve Digital Signature Algorithm (ECDSA) [Nat23], proposed during the 1990s and first adopted by NIST in the early 2000s, is among the major representatives of this class of schemes. ECDSA ensures security properties such as data integrity, authenticity, and non-repudiation, and is broadly used in modern systems. Examples include cryptocurrencies (e.g., *Bitcoin* [Nak08] and *Ethereum* [Eth24]), network protocols (e.g., Transport Layer Security [Res18]), web authentication (e.g., JSON Web Tokens [JBS15]), among others.

Since its proposal, ECDSA's security has been analysed from various angles. From a practical perspective, numerous attacks exploit nonce leakages via lattice sieving and Fourier analysis [Rya18, BH19, GKL+21, ANT+20, XDWL23, GWHH25], underscoring the need for careful implementation and use of ECDSA to prevent side-channel key-recovery attacks. In the theoretical field, initial analyses of ECDSA focused on the hardness of the discrete logarithm problem (DLP) under the Generic Group Model (GGM), as discussed

E-mail: lcardoso@larc.usp.br (Lucas C. Cardoso), mjunior@larc.usp.br (Marcos A. Simplicio Jr)

This work is licensed under a "CC BY 4.0" license.

in [Nec94, Sho97, BPVY00]. In particular, the work of Brown was among the noticeable efforts towards proving the security of ECDSA in the GGM, assuming a strong idealisation of the conversion function f, which is responsible for mapping a curve point onto its x-coordinate modulo a prime number p (representing the order of the curve) [Bro02, Bro05]. Despite its promise, Stern et al. showed that this result was flawed in [SPMLS02]. The reasoning was that the analysis performed by Brown would imply the strong unforgeability of ECDSA, which contradicts the fact that ECDSA signatures are malleable – i.e., if $\sigma = (s, t)$ is a valid signature over message m, then so is $\widetilde{\sigma} = (-s, t)$. So far, security analyses of pure ECDSA in the literature are based on strong idealisations rather than more concrete assumptions (e.g., DLP), leading to the interesting research question of whether it is possible to achieve a more practical result.

Nevertheless, motivated by the broad adoption of ECDSA, many efforts have been directed at using the algorithm to build more advanced cryptographic schemes, often using alternative security assumptions. For example, Lindell builds an efficient two-party threshold scheme proven secure under a new security assumption related to the Paillier encryption scheme [Lin21]. Groth and Shoup analyse the security of ECDSA threshold signature schemes with key derivation and pre-signatures by idealising the conversion function [GS22]. Maire and Pulval-Dady use ECDSA itself as a security assumption to prove the security of a generic blind signature scheme employing zero-knowledge arguments in the MPC-in-the-head paradigm [MPD25].

In an analogous effort, this work aims to evaluate the suitability of ECDSA for building an efficient identity-based signature (IBS) scheme. Identity-based cryptosystems were first proposed by Shamir in 1985 [Sha85], as an alternative approach to a conventional public key infrastructure (PKI). In an identity-based setting, the users' public keys can be their regular identities, such as their full names, email addresses, or social security numbers. Such schemes usually involve a key generation center (KGC), a trusted entity responsible for issuing users' private keys associated with their identities. Among the possible strategies in the literature for building a KGC, one of the most efficient is to rely on hierarchical public key delegation with concatenated Schnorr signatures, as proposed by Galindo and Garcia in [GG09]. Using this construction as a starting point, and also recent contributions to the security analysis of ECDSA, our goal can then be summarised by the following research question:

> *Would it be feasible to build a secure IBS with hierarchical public key delegation using an ECDSA signature as the user's secret key?*

Related Works. The analysis of ECDSA under the Generic Group Model endured as the algorithm's main security result until the studies led by Fersch, Kiltz, and Poettering [FKP16, FKP17]. In their first work [FKP16], using both a non-idealised group and a partially idealised conversion function, the authors prove the security of ECDSA under adaptive chosen-message attacks in the discrete-logarithm setting, following the Bijective Random Oracle model (BROM) in the GenDSA framework. Later, in [FKP17], they formalised the semi-logarithm problem (SLP), a variant first conjectured by Brown as a possible problem to reduce the security of ECDSA in the random oracle model, without relying on a partial idealisation of the conversion function. These analyses, while based on slightly different models, served as a positive stimulus towards a more concrete understanding of ECDSA's security.

It was only until recent work by Hartmann and Kiltz [HK23] that the security of ECDSA under the BROM and its relation to the SLP were brought together. The authors focused on studying the programmability of the conversion function under the BROM, and showed that ECDSA's provable security requires idealisation of (part of) the conversion function, aiming to reduce the existential unforgeability under an adaptive chosen-message attack game to the discrete logarithm problem.

This new perspective on the projection function is also at the core of some recent ECDSA-based schemes. Examples include: the ECDSA-based blind signature scheme proposed by Qin, Cai, and Yuen [QCY21], proven secure under the algebraic variant of the BROM, and recent work by Cho, Fuchsbauer, O'Neil, and Sefranek [CFOS25] discussing a new non-interactive and non-falsifiable assumption, named *Circular Discrete Logarithm Assumption*, which can be used to prove the security of not only ECDSA, but also Schnorr [Sch90] and Sparkle+ [CKM23]. These results show the conditions needed to achieve provable security for ECDSA, and, albeit still relying on (partially) idealized models, they represent a promising approach to evaluate the security of new ECDSA-like schemes in multiple scenarios.

The particular scenario of interest in this study refers to identity-based signatures (IBS), especially highly efficient schemes like the one proposed by Galindo and Garcia in [GG09]. Their construction uses concatenated Schnorr signatures, a well-known technique studied since the early 1990s [Gir91] with applications in modern cryptographic schemes [MPSW19]. Essentially, the KGC issues one-time signatures on the identities of users, who, in turn, use part of the signature as their own private keys; one can then verify the validity of signatures created with this latter key by using the user's identity and the KGC's public key, leveraging the linearity properties of the Schnorr construction.

More generally, as observed by Barreto and Zanon [BZ23], a similar structure can be used to produce succinct non-interactive arguments of knowledge from one-time signatures, i.e., a party can prove possession of a valid signature over a first message by using part of that signature as a private key to sign a second message. This is also known as a Proof of Knowledge (PoK) of the signature, a direct consequence of the Sigma protocol's special soundness property and a highly useful structure when building authentication protocols. The construction of sigma protocols from ECDSA-like schemes has been a recurrent topic of study in an effort to leverage properties that would benefit authentication systems [AGM18, FHLM22, CLR24]. Meanwhile, alternative approaches usually lead to a more involved construction, such as the two-step setup adopted by Woo et al. [WOS+25]: *i*) encoding the ECDSA verification algorithm as a rank-one constraint system (R1CS) [BSCR+19], and then *ii*) using a zkSNARK to prove knowledge of a solution [BCCT12].

Contributions. In this paper, our contribution is threefold:

i) *An ECDSA-like IBS*: to the best of our knowledge, we present the first ECDSA-like identity-based signature scheme allowing ECDSA's signature concatenation. We revisit the technique employed by Galindo and Garcia's IBS scheme, with a slight modification: in addition to using the s-part of an ECDSA signature $\sigma = (s, t)$ as the user's private key, we also leverage the cyclic nature of the group to utilize the t-part when defining the generator for the signing phase. This opens up interesting possibilities in authentication systems with native support for ECDSA signatures (and not for Schnorr signatures), especially web- and cloud-oriented ones, e.g., OAuth systems that implement JSON Web Tokens according to RFC 7519 [JBS15] in frameworks like SPIFFE [SPI25]. This allows a user to add claims to a token T, and then sign the resulting delegation token T' by reusing part of the signature from T. When employed in delegated authentication schemes like Biscuits [CD25], this allows for more compact tokens and a faster delegation procedure, while preserving the scheme's security properties.

ii) *Security proof in the bijective random oracle model*: we prove the security of our proposed scheme following a similar strategy to the GenDSA's framework, but adapted to the identity-based setting, and the work of Hartmann and Kiltz when assessing the obstacles to the provable security of ECDSA. Our scheme is EUF-ID-CMA secure assuming the hardness of breaking ECDSA's EUF-CMA security and

the ψ-relative division resistance of a hash function (ψ-DR) under the BROM. We conclude the existential unforgeability under adaptive chosen-message and adaptive identity attack security in two steps. First, we reduce existential unforgeability under chosen-message attack to the unforgeability under key-only attack in the identity-based setting (Theorem 1). Then, we reduce its security to the hardness of forging an ECDSA signature under chosen-message attacks (Theorem 2). Figure 1 summarizes contributions *i*) and *ii*).

iii) *Comparison between our scheme and Galindo-Garcia's*: we evaluate the performance of our ECDSA-based IBS scheme and Galindo-Garcia's, both theoretically and experimentally, for different security levels. All experiments and implementations are available at https://github.com/lcupertino/ecdsa-ibs.

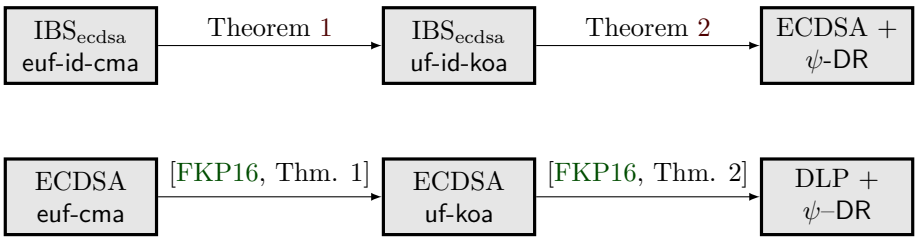

Figure 1: Summary of the security proof under the GenDSA's framework adapted to the identity-based setting. The first row shows the steps of our security proof, while the second shows the original contributions of [FKP16]. In the diagram, the symbol $\mathsf{A} \rightarrow \mathsf{B}$ represents a reduction from assumption A to assumption B.

Paper Outline. The rest of the manuscript is organized as follows. Section 2 defines the mathematical notation and the theoretical framework used throughout this work. Section 3 discusses the core ideas of the Galindo-Garcia identity-based signature scheme. Section 4 details the proposed scheme. In Section 5, we prove the security of our proposal using a game-based approach. Section 6 evaluates the performance of the resulting scheme compared to Galindo-Garcia's, and section 7 concludes the paper.

2 Preliminaries

2.1 Notation

Sets. For any finite set S, we use $x \xleftarrow{\$} S$ to denote the uniformly random sampling of an element $x \in S$ and $|S|$ to represent the number of elements in the set. We use $[n]$ to represent the numbers from 1 to n and $[m : n]$ to represent the numbers from m to $n \geqslant m$. Finally, we use $\{0,1\}^*$ to represent the set of all binary strings with arbitrary finite size.
Algebra. We write $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ as the ring of integers modulo a prime number p, and $\mathbb{Z}_p^* = \mathbb{Z}_p \backslash \{0\}$ when considering only its non-zero elements. Analogously, we use $\mathbb{G}$ to represent an Abelian group, written additively, and $\mathbb{G}^* = \mathbb{G}\backslash\{\mathcal{O}\}$ to denote its contents when excluding the neutral element $\mathcal{O}$ (in the case of elliptic curves, this is the point at infinity). Finally, we write E to denote an elliptic curve whose set of points $E(\mathbb{F}_q)$, for a prime number q, forms a group with generator G such that $|E(\mathbb{F}_q)| = p$ and $pG = \mathcal{O}$.
Security Games. During the security proof, we use the formalisation of Bellare and Rogaway for security games [BR04, BR06]. A game $\mathbf{G}$ is defined as a probabilistic experiment in which an adversary $\mathcal{A}$ interacts with an implicit challenger $\mathcal{C}$ that answers oracle queries sent by $\mathcal{A}$. More formally, a game $\mathbf{G}$ is represented by a sequence of programs $\mathbf{G} = (\textsc{Init}, \mathbf{O}_1, \ldots, \mathbf{O}_n, \textsc{Finalize})$, where $\mathbf{O}_{i \in [n]}$ are the *oracles* of the game. We write

$\mathcal{A}^{\mathbf{O}}$ when the attacker $\mathcal{A}$ has access to oracle $\mathbf{O}$. The binary output b of the i–th game $\mathbf{G}_i$ between the challenger $\mathcal{C}$ and the attacker $\mathcal{A}$ is denoted by $\mathbf{G}_i^{\mathcal{A}} \Rightarrow b$. The attacker *wins* if $b = 1$. To highlight changes between consecutive games, we use grey-colored boxes to represent the corresponding differences.

ORACLES. A query to an oracle is represented with a list of inputs in enclosed brackets. We write $q_{\mathbf{O}}$ to represent the maximum number of queries allowed to the oracle $\mathbf{O}$. For $i \in [q_{\mathbf{O}}]$, the attacker can query the oracle $\mathbf{O}$ on input $\langle x_i \rangle$ and obtain the output $y_i = \mathbf{O}(x_i)$, or the symbol $\perp$ to represent an abort. The table associated with an oracle is represented by $\mathcal{Q}$, and the entries of this table are defined according to the structure of the allowed queries.

2.2 Modelling ECDSA's Conversion Function

The conversion function is given by $f : E(\mathbb{F}_q) \mapsto \mathbb{Z}_p$, meaning that, given a point Q over $E(\mathbb{F}_q)$, the function f maps the point's x–coordinate over $\mathbb{Z}_p$, i.e., $f(Q) = x \bmod p$ with $Q = (x, y)$. It is important to notice that the conversion function does not preserve a visible algebraic structure in the sense that it does not inherit the group laws of its domain. Also, as observed by Fersch, Kiltz, and Poettering, practical instantiations of f are not bijective, and, in fact, the size of the pre-image of f is strictly less than the size of $|\mathbb{Z}_p|$. Guided by those pieces of evidence, the most recent analysis of ECDSA relies on modelling f by combining the disruptive algebraic nature with its real-world properties [FKP16]. The result is $f(x) = (\psi \circ \Pi \circ \varphi)(x)$, where each function φ, Π, and ψ is detailed next.

The function φ maps elements from $E(\mathbb{F}_q)$ to $\mathbb{A}$, a subset of $\{0,1\}^{\lambda}$, with λ defined as $\lceil \log q \rceil$, and $2^{\lambda} \leqslant q < 2^{\lambda+1}$. We represent by α a generic element of the set $\mathbb{A}$. The domain of φ and f are the same, corresponding to points of the elliptic curve, which are, in turn, mapped to their representation as binary strings of length λ following a suitable encoding. In our case, φ is a semi-injective (Definition 1) function such that any two elliptic curve points, P and P', with the same x–component are mapped onto the same element. The definition of the function φ is shown next.

Definition 1 (**Semi-Injective Function**, [HK23])**.** Let $\mathbb{G}$ be a prime-order group and $\mathbb{D}$ a set. A function $\varphi : \mathbb{G}^* \to \mathbb{D}$ is called semi-injective if the following conditions hold:

i) its range $\varphi(\mathbb{G}^*) \subseteq \mathbb{D}$ is efficiently decidable and

ii) it is either injective or a 2-to-1 function, i.e., $\varphi(x) = \varphi(y) \Longrightarrow x \in \{y, y^{-1}\}$.

The function Π maps elements from $\mathbb{A}$ to set $\mathbb{B}$, representing the conversion from binary strings $\alpha \in \mathbb{A} \subseteq \{0,1\}^{\lambda}$ to elements lying in the interval $[0,\, 2^{\lambda} - 1]$, which is consistent with [CMRR23, Appendix F1]. A generic element of $\mathbb{B}$ is represented by β. Lastly, the function ψ reflects the structure of f's range by mapping a point to its x-coordinate modulo p, implementing the characteristic bias of the range of f induced by the 2-to-1 property and the modular reduction [Nat23, Appendix A.4.1]. We represent by t every element such that $\psi(\beta) = t$, for some β lying in set $\mathbb{B}$.

The novelty of this model is the use of Π as a bijection to disrupt any algebraic link between the domain and the range of f, while building φ and ψ to be fast and efficiently computable functions. A direct consequence of these constructions is that the conversion function f is almost invertible [HK23], since the function φ is 2-to-1, ψ is invertible, and Π is bijective. This means that one can invert f, losing only information about the sign of the elliptic curve point. The idealisation of the conversion function is done exactly when modelling the behaviour of the function Π with a bijective random oracle, which is defined as follows.

Definition 2 (**Bijective Random Oracle**)**.** A bijective random oracle, hereby represented by **bRO**, is an idealised public bijection that maps elements from $\mathbb{A}$ to $\mathbb{B}$ and is accessible,

in both directions, via oracles. We write $\mathbf{\Pi}$ when considering the direct oracle, with queried values $\alpha \in \mathbb{A}$, returning values $\beta \in \mathbb{B}$, such that $\beta = \Pi(\alpha)$. Conversely, we use $\mathbf{\Pi}^{-1}$ to represent the inverse oracle, i.e., one that answers with $\alpha = \Pi^{-1}(\beta)$ when queried with value β. The bijective random oracle is represented by $\mathbf{bRO} = (\mathbf{\Pi}, \mathbf{\Pi}^{-1})$. Its specification is depicted in Figure 2.

Oracle $\mathbf{\Pi}(\alpha)$	**Oracle** $\mathbf{\Pi}^{-1}(\beta)$
00 **if** $(\alpha, \cdot) \in \Pi$:	00 **if** $(\cdot, \beta) \in \Pi$:
01 **return** $\Pi(\alpha)$	01 **return** $\Pi^{-1}(\beta)$
02 $\beta \xleftarrow{\$} \mathbb{B}$	02 $\alpha \xleftarrow{\$} \mathbb{A}$
03 **if** $(\cdot, \beta) \in \Pi$:	03 **if** $(\alpha, \cdot) \in \Pi$:
04 **abort**	04 **abort**
05 $\Pi \leftarrow \Pi \cup \{(\alpha, \beta)\}$	05 $\Pi \leftarrow \Pi \cup \{(\alpha, \beta)\}$
06 **return** β	06 **return** α

Figure 2: Specification of the bijective random oracle.

2.3 Identity-Based Signature Schemes

Identity-based signature schemes are a cryptographic primitive derived from the seminal work by Shamir [Sha85]. That work first introduced an identity-based setting in which the need to exchange private and public keys is eliminated due to the presence of a *trusted* authority that issues the user's secret key from a master secret. The trusted authority is commonly referred to as the key generation center (KGC) or Trusted Third Party (TTP). The definition of identity-based signatures is as follows.

Definition 3 (Identity-based Signature). An identity-based signature scheme IBS is a tuple of four efficient algorithms IBS = (Setup, KeyDeriv, Sign, Verify) with the following interfaces:

i) Setup(λ): given the security parameter λ as input, the trusted authority returns the master key pair $(\mathsf{msk}, \mathsf{mpk})$.

ii) KeyDeriv($\mathsf{msk}, \mathtt{id}$): based on the user's identifier $\mathtt{id}$, the trusted authority issues the user's secret key $\mathsf{sk}_{\mathtt{id}}$.

iii) Sign($\mathsf{sk}_{\mathtt{id}}, m, \mathtt{id}$): given the message m and the identifier $\mathtt{id}$ as inputs, the user outputs the corresponding signature σ using its private signing key $\mathsf{sk}_{\mathtt{id}}$.

iv) Verify($\sigma, m, \mathtt{id}, \mathsf{mpk}$): the verification algorithm outputs the value $b = 1$ if the signature is valid over the message m and the user's identity $\mathtt{id}$ under the master public key mpk, and $b = 0$ otherwise.

The security model used to prove the security of our proposed IBS is a variant of the known existential unforgeability under adaptive chosen-message attack [GM84], which is extended by appending the identity attack discussed by Gentry and Silverberg [GS02]. In this model, we consider the existence of two oracles. The first is the signature oracle $\mathbf{O}_{\mathsf{sig}}$, where, upon receiving a query on a message m and identity $\mathtt{id}$, it returns the signature over $m||\mathtt{id}$ using the secret key $\mathsf{sk}_{\mathtt{id}}$. The second oracle we consider is the key-extraction oracle $\mathbf{O}_{\mathsf{ext}}$, which is responsible for answering queries on a given identity $\mathtt{id}$ – similarly to the KGC or TTP in an identity-based scenario. The oracle outputs the user's secret key $\mathsf{sk}_{\mathtt{id}}$ bound to its identity $\mathtt{id}$. The specification of the oracles is depicted in Figure 3.

Oracle $\mathbf{O}_{\mathsf{ext}}(\mathtt{id})$	**Oracle** $\mathbf{O}_{\mathsf{sig}}(m, \mathtt{id})$
00 **if** $\langle \mathtt{id} \rangle \in \mathcal{Q}_{\mathsf{ext}}$:	00 **if** $\langle m, \mathtt{id} \rangle \in \mathcal{Q}_{\mathsf{sig}}$:
01 $\quad \mathsf{sk}_{\mathtt{id}} \leftarrow \mathbf{O}_{\mathsf{ext}}(\mathtt{id})$	01 $\quad \sigma \leftarrow \mathbf{O}_{\mathsf{sig}}(m, \mathtt{id})$
02 $\quad$ **return** $\mathsf{sk}_{\mathtt{id}}$	02 $\quad$ **return** σ
03 $\mathsf{sk}_{\mathtt{id}} \xleftarrow{\$} \textsc{KeyDeriv}(\mathsf{msk}, \mathtt{id})$	03 $\mathsf{sk}_{\mathtt{id}} \leftarrow \mathbf{O}_{\mathsf{ext}}(\mathtt{id})$
04 $\mathcal{Q}_{\mathsf{ext}} \leftarrow \mathcal{Q}_{\mathsf{ext}} \cup \{(\mathtt{id}, \mathsf{sk}_{\mathtt{id}})\}$	04 $\sigma \xleftarrow{\$} \textsc{Sign}(\mathsf{sk}_{\mathtt{id}}, m, \mathtt{id})$
05 **return** $\mathsf{sk}_{\mathtt{id}}$	05 $\mathcal{Q}_{\mathsf{sig}} \leftarrow \mathcal{Q}_{\mathsf{sig}} \cup \{(m, \mathtt{id}, \sigma)\}$
	06 **return** σ

Figure 3: Specification of the extraction (left) and signing oracles (right).

Now, we are ready to define the two central security models that will be used throughout this work. The first one is the existential unforgeability under chosen-message and chosen-identity attack, hereby referred to as EUF-ID-CMA, which is defined as follows:

Definition 4 (**EUF–ID–CMA**)**.** An adversary $\mathcal{A}$ is said to (t, ε)–break the existential unforgeability under chosen-message and identity attack game against the identity-based signature scheme IBS if, after receiving the master public key and given access to the extraction and signing oracles ($\mathbf{O}_{\mathsf{ext}}$ and $\mathbf{O}_{\mathsf{sig}}$, respectively), it can produce a forged signature on a new identity and a new identity-message pair that has never been queried before. This is done in time at most t and with an advantage given by

$$\Pr\left[\textsc{Verify}(\sigma^\star, m^\star, \mathtt{id}^\star, \mathsf{mpk}) = 1 \;\middle|\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \textsc{Setup}(\lambda) \\ (\sigma^\star, \mathtt{id}^\star, m^\star) \leftarrow \mathcal{A}^{\mathbf{O}_{\mathsf{ext}}, \mathbf{O}_{\mathsf{sig}}}(\mathsf{mpk}) \\ \langle \mathtt{id}^\star \rangle \notin \mathcal{Q}_{\mathsf{ext}} \cap \langle m^\star, \mathtt{id}^\star \rangle \notin \mathcal{Q}_{\mathsf{sig}} \end{array} \right] \geqslant \varepsilon.$$

A weaker but still useful security notion is the *unforgeability under key-only attacks*. In this setting, the games proceed as a usual chosen-message attack game, but with one difference: the attacker is not allowed to pose signing queries to the oracle $\mathbf{O}_{\mathsf{sig}}$. Instead, only the extraction oracle is preserved and works as shown in Figure 3. We define this notion in the identity-based setting as follows:

Definition 5 (**UF-ID-KOA**)**.** An adversary $\mathcal{A}$ is said to (t, ε)–break the unforgeability under key-only and identity attack game against the identity-based signature scheme IBS if, after receiving the master public key and given access to the extraction oracle, it can produce a forged signature on a new identity and a new identity-message pair that has never been queried. This is done in time at most t and with an advantage given by

$$\Pr\left[\textsc{Verify}(\sigma^\star, m^\star, \mathtt{id}^\star, \mathsf{mpk}) = 1 \;\middle|\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \textsc{Setup}(\lambda) \\ (\sigma^\star, \mathtt{id}^\star, m^\star) \leftarrow \mathcal{A}^{\mathbf{O}_{\mathsf{ext}}}(\mathsf{mpk}) \\ \langle \mathtt{id}^\star \rangle \notin \mathcal{Q}_{\mathsf{ext}} \end{array} \right] \geqslant \varepsilon.$$

2.4 Cryptographic Assumptions

In this section, we present the assumptions used throughout the security analysis of our scheme. We start with the formulation of the discrete logarithm problem:

Definition 6 (**Discrete Logarithm Problem**)**.** Let $(\mathbb{G}, G, p)$ be a group of prime order p and generator G. An adversary $\mathcal{A}$ is said to $(t_{\mathsf{dl}}, \varepsilon_{\mathsf{dl}})$–break the discrete logarithm problem if it runs in time at most t_{dl} and has an advantage

$$\Pr\left[\hat{x} = x \;\middle|\; \begin{array}{l} \hat{x} \leftarrow \mathcal{A}(X) \\ x \xleftarrow{\$} \mathbb{Z}_p^*, \; X = xG \end{array} \right] \geqslant \varepsilon_{\mathsf{dl}}.$$

Next, there are two problems regarding the security of hash functions that will be used throughout this work. The first one refers to the collision resistance of a hash function:

Definition 7 (**Collision Resistance of a Hash Function**)**.** Let $\mathsf{H} : \{0,1\}^* \mapsto \mathbb{Z}_p$ be a hash function. An adversary $\mathcal{A}$ is said to $(t_{\mathsf{cr}}, \varepsilon_{\mathsf{cr}})$–break the collision resistance of hash function H if it runs in time at most t_{cr} and with advantage

$$\Pr\left[\begin{array}{l|l} \mathsf{H}(m_1) = \mathsf{H}(m_0) & \begin{array}{l}(m_0, m_1) \leftarrow \mathcal{A}(\cdot) \\ m_0 \neq m_1 \\ m_0, m_1 \in \{0,1\}^*\end{array}\end{array}\right] \geqslant \varepsilon_{\mathsf{cr}}.$$

The second corresponds to a generalized version of this problem: the ψ–relative division resistance of a hash function introduced by [FKP16]. It is stated as follows.

Definition 8 (ψ–**Relative Division Resistance of a Hash Function**, [FKP16])**.** Let $\mathsf{H} : \{0,1\}^* \mapsto \mathbb{Z}_p$ be a hash function, $\mathbb{B}$ a set, and $\psi : \mathbb{B} \mapsto \mathbb{Z}_p$ a function. Let $\mathbb{B}^* = \psi^{-1}\left(\mathbb{Z}_p^*\right)$. An adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is said to $(t_{\mathsf{dr}}, \varepsilon_{\mathsf{dr}})$–breaks the ψ–relative division resistance of H if it runs in time at most t_{dr} and with advantage

$$\Pr\left[\begin{array}{l|l} \dfrac{\mathsf{H}(m_1)}{\psi(\beta_1)} = \dfrac{\mathsf{H}(m_0)}{\psi(\beta_0)} & \begin{array}{l}\beta_0, \beta_1 \xleftarrow{\$} \mathbb{B}^*, \\ (m_0, \Gamma) \xleftarrow{\$} \mathcal{A}_1(\beta_0), \\ m_1 \xleftarrow{\$} \mathcal{A}_2(\Gamma, \beta_1)\end{array}\end{array}\right] \geqslant \varepsilon_{\mathsf{dr}}.$$

Here, Γ is some arbitrary state information passed from $\mathcal{A}_1$ to $\mathcal{A}_2$.

The intuition behind Definition 8 is detailed next. For the function $\psi : \mathbb{B} \mapsto \mathbb{Z}_p$, an adversary receives two values, $\beta_0, \beta_1 \in \mathbb{B}^*$, for which, since $\mathbb{B}^* = \psi^{-1}(\mathbb{Z}_p)$, the existence of $\psi(\beta_0)$ and $\psi(\beta_1)$ is well-defined. The adversary attempts to find two messages such that $\psi(\beta_0)\mathsf{H}(m_1) = \psi(\beta_1)\mathsf{H}(m_0)$. Assuming $\psi(\beta_0) \neq \psi(\beta_1)$, the adversary's goal is to find a message such that $\mathsf{H}(m_0) = 0$, when $\psi(\beta_0) = 0$ (without loss of generality), or to find $\gamma = \psi(\beta_0)/\psi(\beta_1)$ such that $\mathsf{H}(m_0) = \gamma\mathsf{H}(m_1)$.

Definition 8 is central for a result regarding the security of ECDSA signatures considering the modelling of the conversion function according to section 2.2, which corresponds to the following lemma.

Lemma 1 ([FKP16])**.** *Let f be the ECDSA conversion function modelled as described in section 2.2. If an attacker can find messages m and $m^\star$ (not necessarily distinct) such that, for two values, β and $\beta^\star$, the relation $\dfrac{\mathsf{H}(m)}{\psi(\beta)} = \dfrac{\mathsf{H}(m^\star)}{\psi(\beta^\star)}$ holds, then it can produce a forgery on signature $\sigma^\star = (s^\star, t^\star)$ over message $m^\star$. In this case, the adversary is said to break the ψ–relative division resistance of the hash function H.*

The proof follows from manipulating the verification equation of the original signature. Let $\sigma = (s,t)$ be a valid signature over message m under public key X. That is, $\sigma = (s,t)$ satisfies the equation $f(s^{-1}(cG + tX)) = t$, with $c = \mathsf{H}(m)$, and $s, t \neq 0$. Recall that $t = \psi(\beta)$, with $\beta \in \mathbb{B}$. Consider an adversary that breaks the ψ-relative division resistance of the hash function H, i.e., the adversary finds another message $m^\star$ such that $tc^\star = t^\star c$ for $c^\star = \mathsf{H}(m^\star)$. Then, the signature $\sigma^\star = (s^\star, t^\star)$ is a valid signature over message $m^\star$ under public key X, with $s^\star = sc^\star/c$ and $t^\star = \psi(\beta^\star)$. This is because, in such a case, we would obtain the same verification equation $f((s^\star)^{-1}(c^\star G + t^\star X)) = t^\star$, for $s^\star = sc^\star/c$, and, hence, a valid forgery $\sigma^\star = (s^\star, t^\star)$ over message $m^\star$ and public key X.

Schnorr and ECDSA security analysis. Now we connect the previous results with the security analysis of ECDSA and its similarities with the one of the Schnorr signature

scheme. For analysing the Schnorr signature scheme, the General Forking Lemma of Bellare and Neven [BN06] is a historically used tool. The proof consists of finding collisions in R by manipulating the random oracle to respond to signing queries with values $R \leftarrow sG - cX$, with $s, c \xleftarrow{\$} \mathbb{Z}_p^*$, embedded in the answer, and then extracting the private signing key x. The idea is shown as follows: if the adversary can produce a forgery $\sigma^\star = (R^\star, s^\star)$ over the forged message $m^\star$ by rewinding the random oracle so that the R value is repeated, then it can extract the discrete logarithm $x = \mathrm{dlog}_G X$. Notice that, if $R = R^\star$ and, by the construction of the random oracle, $c \neq c^\star$, then we can write $sG - cX = s^\star G - c^\star X$, and thus derive the discrete logarithm

$$x = \frac{s^\star - s}{c^\star - c} \bmod p. \tag{1}$$

The security analysis of the ECDSA scheme follows a similar reasoning, but it is nevertheless more intricate than the Schnorr counterpart. Specifically, now the attacker tries to find collisions between the values of the conversion function, i.e., $t^\star = t$, so that $R^\star = \eta R$ for $|\eta| = 1$. In such a case, for two signatures $\sigma = (s, t)$ and $\sigma^\star = (s^\star, t^\star)$ over messages m and $m^\star$, correspondingly, we can derive the value $r^\star = \mathrm{dlog}_G R^\star$ from $(\eta s - s^\star)\, R^\star = (c - c^\star)\, G$, with $\eta s \neq s^\star$, and compute the discrete logarithm

$$x = \frac{s^\star c - \eta s c^\star}{t^\star\,(\eta s - s^\star)} \bmod p, \tag{2}$$

with digests $c = \mathsf{H}(m)$ and $c^\star = \mathsf{H}(m^\star)$. By modelling the projection function f as a bijective random oracle with a programmable part, we implicitly allow one to rewind the oracle until finding a collision (i.e., $R = \eta R^\star$) and extracting the discrete logarithm.

3 Galindo-Garcia IBS

In 2009, Galindo and Garcia proposed an IBS construction using two concatenated Schnorr signatures, leveraging the simplicity and efficiency of such scheme [GG09]. While the technique itself is not new, it was demonstrated to be useful in different scenarios, such as identity-based encryption schemes and primitives employed in public key infrastructures (PKI). The scheme is defined next by considering the following publicly accessible parameters: the group $\mathbb{G}$ of prime order p written multiplicatively and generated by g, and the secure hash functions H and G mapping elements from $\{0,1\}^*$ to $\mathbb{Z}_p$.

Definition 9 (**Galindo-Garcia IBS**)**.** The Galindo-Garcia IBS is represented by the tuple of four efficient algorithms, IBS = (Setup, KeyDeriv, Sign, Verify), with the following specifications:

i) Setup (λ): upon receiving the security parameter and having public access to the public parameters $\mathsf{pp} = (\mathbb{G}, g, p, \mathsf{H}, \mathsf{G})$, the trusted third party first randomly samples $x \xleftarrow{\$} \mathbb{Z}_p$ and defines $X = g^x$. Then, the master key pair $(\mathsf{msk}, \mathsf{mpk})$ is set as $\mathsf{msk} = x$ and $\mathsf{mpk} = X$. Finally, it returns $(\mathsf{msk}, \mathsf{mpk})$.

ii) KeyDeriv $(\mathsf{msk}, \mathtt{id})$: using the master secret key and the user's identity, the trusted authority proceeds as in the original specification of the Schnorr scheme's signing algorithm. It samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $R = g^r$, and computes the digest $c = \mathsf{H}(R, \mathtt{id})$. Finally, after computing $y = r + cx \bmod p$, it returns the user's secret key as $\mathsf{sk}_{\mathtt{id}} = (y, R)$.

iii) Sign $\left(\mathsf{sk}_{\mathtt{id}}, m, \mathtt{id}\right)$: on inputs the user's secret key $\mathsf{sk}_{\mathtt{id}} = (y, R)$, user's identity $\mathtt{id}$, and message m, the user randomly samples $a \xleftarrow{\$} \mathbb{Z}_p$, and computes the commitment $A = g^a$. Next, it computes the digest $d = \mathsf{G}(\mathsf{id}, A, m)$ followed by the value $b = a + yd \bmod p$ to output the signature $\sigma = (A, b, R)$.

iv) VERIFY $(\sigma, m, \mathtt{id}, \mathsf{mpk})$: the verification algorithm proceeds by first parsing the signature σ as (A, b, R), followed by computing both digests, $c = \mathsf{H}(R, \mathtt{id})$ and $d = \mathsf{G}(\mathtt{id}, A, m)$, to finally output true if the following equality holds:

$$g^b \stackrel{?}{=} A\,(RX^c)^d\,,$$

and false, otherwise.

The scheme is proven secure under the EUF-ID-CMA (Definition 4) game in the random oracle model (ROM) by constructing two reductions solving the DLP by means of the General Forking Lemma (Lemma 2) [BN06] and the Multiple Forking Lemma (Lemma 3) [BPW12], which are stated as follows.

Lemma 2 (**General Forking Lemma**, [BN06])**.** *Fix $q \in \mathbb{Z}^+$ and a set $\mathbb{S}$ such that $|\mathbb{S}| \geqslant 2$. Let $\mathcal{W}$ be a randomized algorithm that on input string x and elements $s_1, \ldots, s_\gamma \in \mathbb{S}$ returns a pair (I, σ) consisting of an integer $0 \leqslant I \leqslant \gamma$ and a string σ. Let $\mathcal{F}_\mathcal{W}$ be the forking algorithm associated with $\mathcal{W}$ such that, on input x, it returns $(1, \sigma_0, \sigma_1)$ if it succeeds and $(0, \perp, \perp)$ otherwise. Let $\mathcal{G}$ be a randomized algorithm that takes no input and returns a string. Let the accepting probability,* acc*, be defined as*

$$\mathsf{acc} = \Pr\left[I_0 \geqslant 1 \;\middle|\; \begin{array}{l} x \xleftarrow{\$} \mathcal{G},\ s_1^0, \ldots, s_\gamma^0 \xleftarrow{\$} \mathbb{S}, \\ (I_0, \sigma_0) \xleftarrow{\$} \mathcal{W}\left(x, s_1^0, \ldots, s_\gamma^0\right) \end{array}\right],$$

and the probability of returning a valid forgery

$$\mathsf{gfrk} = \Pr\left[b = 1 \;\middle|\; \begin{array}{l} x \xleftarrow{\$} \mathcal{G}, \\ (b, \sigma_0, \sigma_1) \xleftarrow{\$} \mathcal{F}_\mathcal{W}(x) \end{array}\right],$$

then the following holds

$$\mathsf{gfrk} \geqslant \mathsf{acc}\left(\frac{\mathsf{acc}}{\gamma} - \frac{1}{|\mathbb{S}|}\right).$$

Lemma 3 (**Multiple Forking Lemma**, [BPW12])**.** *Fix a positive integer q and a set $\mathbb{S}$ with at least two elements. Let n be and odd integer and $\mathcal{M}_{\mathcal{W},n}(x)$ be the multiple forking algorithm over $\mathcal{W}$ that returns the tuple (I, J, σ), with $0 \leqslant J < I \leqslant \gamma$. Algorithm $\mathcal{M}$ returns $(1, \sigma_0, \ldots, \sigma_n)$ in case of success and $(0, \perp)$ otherwise. Let $\mathcal{G}$ be a randomized algorithm that takes no input and returns a string. For the accepting probability,* acc*, defined as:*

$$\mathsf{acc} = \Pr\left[I_0 \geqslant 1 \cap J_0 \geqslant 1 \;\middle|\; \begin{array}{l} x \xleftarrow{\$} \mathcal{G},\ s_1^0, \ldots, s_\gamma^0 \xleftarrow{\$} \mathbb{S}, \\ (I_0, \sigma_0) \xleftarrow{\$} \mathcal{W}\left(x, s_1^0, \ldots, s_\gamma^0\right) \end{array}\right],$$

and the probability of succeeding in the multiple forking experiment, given by

$$\mathsf{mfrk} = \Pr\left[b = 1 \;\middle|\; \begin{array}{l} x \xleftarrow{\$} \mathcal{G}, \\ (b, \sigma_0, \ldots, \sigma_n) \xleftarrow{\$} \mathcal{M}_{\mathcal{W},n}(x) \end{array}\right],$$

the following inequality holds

$$\mathsf{mfrk} \geqslant \mathsf{acc}\left(\frac{\mathsf{acc}^n}{\gamma^{2n}} - \frac{n}{|\mathbb{S}|}\right).$$

Later, Chatterjee, Kamath, and Kumar highlighted some gaps in the formal analysis in the original paper of Galindo and Garcia and proposed a corrected version of the security proof, without compromising the IBS itself. We refer to the authors' original work for a more detailed discussion [CKK13]. All in all, the Galindo-Garcia IBS offers a smaller number of exponentiations and shorter signatures compared to pairing-based identity-based signatures [Her06, BLMQ05], relatively recent work at the time, and represents an attractive option in resource-constrained scenarios due to its simplicity and linearity.

4 Proposed Scheme

In this section, we describe our proposed scheme. It is constructed from the hierarchical public key delegation technique, where a trusted authority issues a one-time signature that is used as a private signing key to sign subsequent messages. In our case, however, due to the non-linear characteristics of the ECDSA scheme, proposing an ECDSA-based IBS should be done carefully, given the susceptibility to nonce-reuse attacks. Even further, for a private signing key $\mathsf{sk}_{\mathtt{id}} = (u_{\mathtt{id}}, t_{\mathtt{id}})$ corresponding to an ECDSA signature, we aim to use $u_{\mathtt{id}}$ as the private key and the point R that results $f(R) = t_{\mathtt{id}}$ as generator during the signing phase. After describing the scheme, we discuss the malleability property and other variants that could be considered to improve the performance. Our proposed scheme is presented as follows:

Definition 10. Our identity-based signature scheme $\text{IBS}_{\text{ecdsa}}$ is a tuple of four efficient algorithms $\text{IBS}_{\text{ecdsa}} = (\textsc{Setup}, \textsc{KeyDeriv}, \textsc{Sign}, \textsc{Verify})$:

i) $\textsc{Setup}(\lambda)$: upon input λ and public parameters $\mathsf{pp} = (E(\mathbb{F}_q), q, G, p, \mathsf{H}, \mathsf{G})$, where H and G are secure hash functions, the trusted third party authority randomly samples $x \xleftarrow{\$} \mathbb{Z}_p^*$, computes $X = xG$, and outputs the master key pair $(\mathsf{msk}, \mathsf{mpk}) = (x, X)$.

ii) $\textsc{KeyDeriv}(\mathsf{msk}, \mathtt{id})$: using the master secret key msk and the user's identifier $\mathtt{id}$, the trusted authority generates a random nonce r over $\mathbb{Z}_p$, sets $R = rG$, computes $t_{\mathtt{id}} = f(R)$, and the digest $c = \mathsf{H}(\mathtt{id})$. Finally, the trusted authority computes $u_{\mathtt{id}} = r^{-1}(c + t_{\mathtt{id}} x) \bmod p$ and outputs the user's private key $\mathsf{sk}_{\mathtt{id}} = (u_{\mathtt{id}}, t_{\mathtt{id}})$.

iii) $\textsc{Sign}(\mathsf{sk}_{\mathtt{id}}, m, \mathtt{id})$: using $u_{\mathtt{id}}$ as private signing key, the user recovers the point R from the correctness of the first signature, that is, $R = {u_{\mathtt{id}}}^{-1}(cG + t_{\mathtt{id}} X)$, and sets the verification key $U = u_{\mathtt{id}} R$. Then, the user computes a random nonce a, sets the commitment $A = aR$, uses the conversion function to compute $t = f(A)$, and the digest $d = \mathsf{G}(m, \mathtt{id})$, to finally computes $s = a^{-1}(d + t u_{\mathtt{id}}) \bmod p$ and outputs the signature $\sigma = (s, t, R)$.

iv) $\textsc{Verify}(\sigma, m, \mathtt{id}, \mathsf{mpk})$: the signature σ is parsed into (s, t, R) to extract $t_{\mathtt{id}} = f(R)$. The verifier computes digests $c = \mathsf{H}(\mathtt{id})$ and $d = \mathsf{G}(m, \mathtt{id})$, and using the master public key X, the verifier recovers $A = s^{-1}(dR + t(cG + t_{\mathtt{id}} X))$ and outputs true or false depending on whether the following equality holds $t = f(A)$.

Figure 4 details the complete scheme. Notice that the verification equation for the produced signature $\sigma = (s, t, R)$ follows from two observations: the first is that the signing algorithm proceeds as ECDSA signing over message m and identity $\mathtt{id}$ with private signing key $u_{\mathtt{id}}$, which is inherited from the ECDSA signature delegated to the user. The second observation is that $U = u_{\mathtt{id}} R$ mimics a public verification key, which, in identity-based scenarios, can be associated with the user's unique identity $\mathtt{id}$ (cf. Section 1). Since $u_{\mathtt{id}} R = cG + t_{\mathtt{id}} X$, we can rewrite the verification equation for signature $\sigma = (s, t, R)$ as $sA = dR + t(u_{\mathtt{id}} R)$, and the correctness follows.

Discussion. Malleability is an inherent property of ECDSA signatures. It states that if an ECDSA signature $\sigma = (s, t)$, with $t = f(R)$, over a message m is valid, then so is $\sigma' = (-s, t)$, with $t = f(-R)$. This follows directly from the verification equation and the 2-to-1 characteristic of the conversion function (cf. Section 2.2). Regarding the scheme's security against chosen-message attacks, the existential unforgeability property remains intact, since the attacker must forge a signature on a brand-new message; hence, malleability is rendered useless. Notice, however, that malleability can be combined with other vulnerabilities to produce malleability attacks [DW14].

$\underline{\text{SETUP}(\lambda)}$
00 $x \xleftarrow{\$} \mathbb{Z}_p^*$
01 $\mathsf{msk} \leftarrow x$
02 $X \leftarrow xG$
03 $\mathsf{mpk} \leftarrow X$
04 **return** $(\mathsf{msk}, \mathsf{mpk})$

$\underline{\text{KEYDERIV}(\mathsf{msk}, \mathtt{id})}$
05 $x \leftarrow \mathsf{msk}$
06 $r \xleftarrow{\$} \mathbb{Z}_p^*$
07 $R = rG$
08 **if** $R = \mathcal{O}$:
09 **abort**
10 $t_{\mathtt{id}} = f(R)$
11 **if** $t = 0$:
12 **abort**
13 $c \leftarrow \mathsf{H}(\mathtt{id})$
14 $u_{\mathtt{id}} \leftarrow r^{-1}(c + t_{\mathtt{id}} x) \bmod p$
15 **return** $\mathsf{sk}_{\mathtt{id}} = (u_{\mathtt{id}}, t_{\mathtt{id}})$

$\underline{\text{SIGN}(\mathsf{sk}_{\mathtt{id}}, m, \mathtt{id})}$
16 $(u_{\mathtt{id}}, t_{\mathtt{id}}) \leftarrow \mathsf{sk}_{\mathtt{id}}$
17 $X \leftarrow \mathsf{mpk}$
18 $c \leftarrow \mathsf{H}(\mathtt{id})$
19 $R \leftarrow u_{\mathtt{id}}^{-1}(cG + tX)$
20 $a \xleftarrow{\$} \mathbb{Z}_p^*$
21 $A \leftarrow aR$
22 **if** $A = \mathcal{O}$:
23 **abort**
24 $t = f(A)$
25 **if** $t = 0$:
26 **abort**
27 $d \leftarrow \mathsf{G}(m, \mathtt{id})$
28 $s \leftarrow a^{-1}(d + tu_{\mathtt{id}}) \bmod p$
29 $\sigma = (s, t, R)$
30 **return** σ

$\underline{\text{VERIFY}(\sigma, m, \mathtt{id}, \mathsf{mpk})}$
31 $(s, t, R) \leftarrow \sigma$
32 **if** $s = 0 \cup t = 0$:
33 **abort**
34 $X \leftarrow \mathsf{mpk}$
35 $c \leftarrow \mathsf{H}(\mathtt{id})$
36 $d \leftarrow \mathsf{G}(m, \mathtt{id})$
37 $t_{\mathtt{id}} \leftarrow f(R)$
38 $U \leftarrow cG + t_{\mathtt{id}} X$
39 $A \leftarrow s^{-1}(dR + tU)$
40 **if** $f(A) \neq t$:
41 **return** false
42 **return** true

Figure 4: Complete description of $\text{IBS}_{\text{ecdsa}}$.

Our scheme also preserves the malleability property of ECDSA signatures without revealing sensitive information to potential attackers. This can be observed from the verification equation for signature $\sigma = (s, t, R)$, written as $sA = dR + t(u_{\mathtt{id}} R)$, with $u_{\mathtt{id}} R = cG + t_{\mathtt{id}} X$. It can be seen that, by substituting the pair (s, A) by $(-s, -A)$, the left-hand side of the equation remains unchanged. Additionally, due to the semi-injective property of the conversion function (Definition 1), the value $t = f(-A)$ also remains the same after the substitution previously shown, and the right-hand side of the equation is preserved.

Interestingly, the R–part of the signature σ is not malleable, i.e., $\sigma' = (-s, t, -R)$, with $t = f(-A)$, does not produce a valid signature over the message-identity pair since it would be required that the sign of $u_{\mathtt{id}}$ also changes due to the malleability of the user's private key (which is an ECDSA signature). This means preserving the term $u_{\mathtt{id}} R = (-u_{\mathtt{id}})(-R)$ and obtaining a right-hand side equals to $d(-R) + t(u_{\mathtt{id}} R)$, and a left-hand side sA, which is an impossibility result since R differs from the point at infinity by construction (Figure 4) and d is rarely zero.

Variants of the scheme. Part of the importance of this work lies in some aspects when considering the usage of ECDSA-like schemes and identity-based signature schemes with hierarchical delegation of public keys. ECDSA performs, during the verification phase, the following steps to validate the signature $\sigma = (s, t)$ over the message m under the public key X: *i*) compute $c = \mathsf{H}(m)$, *ii*) recover point $R = s^{-1}(cG + tX)$, and *iii*) outputs the boolean value of $f(R) = t$.

When considering real-world implementations of other signature schemes (e.g., Schnorr's), additional information, such as public keys and commitments, is used as input to the hash function besides the messages themselves to avoid related-key attacks [BK03, MSM+16]. However, when embedding the commitment into the hash function, considering a construction analogous to Schnorr's, the verification process would be impossible since R is recovered based on digest c, which is equal to $\mathsf{H}(R, m)$ if the commitment serves as input to the hash function.

KeyDeriv($\mathsf{msk}, \mathtt{id}$)	Sign($\mathsf{sk}_{\mathtt{id}}, m, \mathtt{id}$)
00 $x \leftarrow \mathsf{msk}$	00 $(u_{\mathtt{id}}, t_{\mathtt{id}}) \leftarrow \mathsf{sk}_{\mathtt{id}}$
01 $X \leftarrow xG$	01 $c \leftarrow \mathsf{H}(X, \mathtt{id})$
02 $r \xleftarrow{\$} \mathbb{Z}_p$	02 $U = cG + t_{\mathtt{id}} X$
03 $R \leftarrow rG$	03 $R \leftarrow u_{\mathtt{id}}^{-1} U$
04 **if** $R = \mathcal{O}$:	04 $a \xleftarrow{\$} \mathbb{Z}_p$
05 **abort**	05 $A \leftarrow aR$
06 $t_{\mathtt{id}} = f(R)$	06 **if** $A = \mathcal{O}$:
07 **if** $t_{\mathtt{id}} = 0$	07 **abort**
08 **abort**	08 $t = f(A)$
09 $c \leftarrow \mathsf{H}(X, \mathtt{id})$	09 **if** $t = 0$:
10 $u_{\mathtt{id}} \leftarrow r^{-1}(c + t_{\mathtt{id}} x) \bmod p$	10 **abort**
11 $\mathsf{sk}_{\mathtt{id}} = (u_{\mathtt{id}}, t_{\mathtt{id}})$	11 $d \leftarrow \mathsf{G}(U, m, \mathtt{id})$
12 **return** $\mathsf{sk}_{\mathtt{id}}$	12 $s \leftarrow a^{-1}(d + t u_{\mathtt{id}}) \bmod p$
	13 $\sigma = (s, t, R)$
	14 **return** σ

Figure 5: Specification of the key derivation and signing algorithm for the $\text{IBS}^{\mathtt{v}_1}_{\text{ecdsa}}$ variant of our scheme.

A solution to this problem would be to consider using the public key X of the signer alongside the message to be signed m instead. In such a case, the digest can be calculated so that the point R is recovered. Then, the signing algorithm of our IBS would be written as in Figure 5. We denote this first variant of our scheme as $\text{IBS}^{\mathtt{v}_1}_{\text{ecdsa}}$.

Another possible implementation would be to use private signing keys as $\mathsf{sk}_{\mathtt{id}} = (u_{\mathtt{id}}, R)$ with $f(R) = t_{\mathtt{id}}$, thus, implicitly, obtaining an ECDSA signature. Using this format, it would be possible to use the commitment R as input to the hash function H, and modify the verification equation as follows: *i*) parse the signature as $(u_{\mathtt{id}}, R) \leftarrow \sigma$, *ii*) compute the digest $c = \mathsf{H}(R, \mathtt{id})$, *iii*) compute the projection of the commitment $t_{\mathtt{id}} = f(R)$, and *iv*) verify if the following equality holds $u_{\mathtt{id}} R = cG + t_{\mathtt{id}} X$.

In this case, when R is used as the generator the signing phase, one can uniformly sample $a \xleftarrow{\$} \mathbb{Z}_p$, compute the point $A = aR$, its projection along the x–axis $t = f(A)$, and the digest $d = \mathsf{G}(U, m, \mathtt{id})$, with $U = u_{\mathtt{id}} R$. Finally, the signer computes $s = a^{-1}(d + t u_{\mathtt{id}}) \bmod p$ and the output signature is $\sigma = (s, t, R)$. Verification occurs as usual ECDSA verification would, but requires the recovery of the point $U = cG + tX$. Then, the verifier computes $A' = s^{-1}(dR + tU)$ and outputs the boolean value of $f(A') \stackrel{?}{=} t$.

Compared to $\text{IBS}_{\text{ecdsa}}$ and $\text{IBS}^{\mathtt{v}_1}_{\text{ecdsa}}$, during the signing phase, using $u_{\mathtt{id}}$ as the private key while revealing point R avoids the need to conduct the verification of an ECDSA signature. This is because, in version $\mathtt{v}_2$, the recovery of point R (Line 03 of the Sign algorithm in Figure 5) would not be required, reducing the overhead of the operation. As will be shown in Section 6, these improvements will be crucial to improve the performance of the scheme.

5 Security Proof

In this section, we prove the security of our scheme $\text{IBS}_{\text{ecdsa}}$. Before starting the proof itself, we give some intuition about the difficulties inherent in modelling the reduction algorithm. First, to employ a reductionist argument aiming to prove the EUF-ID-CMA security (Definition 4) of our scheme against an adversary $\mathcal{A}$, we must construct an algorithm $\mathcal{R}$ that solves an equivalent problem by invoking $\mathcal{A}$. In our case, we follow a strategy

analogous to the one of Fersch, Kiltz, and Poettering [FKP16] and use a weaker security notion adapted to the identity-based setting – namely, UF-ID-KOA (Definition 5) – as an intermediate step in our proof.

This is because the partially idealised characteristics of the conversion function require different strategies during the security proof of ECDSA signatures. In the strategy adopted by [FKP16], for example, the reduction from EUF-CMA to UF-KOA depends on a non-rewinding programmable conversion function, while the reduction from UF-KOA to the DLP requires the usage of rewinding techniques via the Forking Lemma.

Another important observation is that we account for the probability of finding hash collisions, since the hash functions are not idealised. These build upon the need to balance the idealised characteristics (of part) of the conversion function, one of the pillars of the considered security assumptions.

Theorem 1. *Under the bijective random oracle model, if there exists a p.p.t. adversary* $\mathcal{F}$ *that* $(t, \varepsilon, q_{\mathsf{ext}}, q_{\mathsf{sig}}, q_{\mathbf{\Pi}})$*–breaks existential unforgeability of* $\mathrm{IBS}_{\mathrm{ecdsa}}$*, then there exists a p.p.t. adversary* $\mathcal{F}_0$ *that* $(t_0, \varepsilon_0, q_{\mathbf{\Pi}})$*–breaks existential unforgeability of* $\mathrm{IBS}_{\mathrm{ecdsa}}$ *under key-only and identity attack and p.p.t. adversaries,* $\mathcal{B}_1$ *and* $\mathcal{B}_2$*, that* (t_1, ε_1)*– and* (t_2, ε_2)*–break collision resistance of hash functions* H *and* G*, respectively, with*

$$\varepsilon \leqslant \varepsilon_0 + \varepsilon_1 + \varepsilon_2 + \frac{3Q\,(q_{\mathsf{ext}} + q_{\mathsf{sig}})}{(p-1)/2 - Q}, \tag{3}$$

where $t_0 \geqslant t + \mathcal{O}\,(q_{\mathsf{sig}} + q_{\mathsf{ext}})\,,\ t_1 + t_2 \geqslant t + \mathcal{O}\,(q_{\mathsf{ext}} + q_{\mathsf{sig}})$*, and* $Q = q_{\mathsf{ext}} + q_{\mathsf{sig}} + q_{\mathbf{\Pi}}$*.*

Proof. We start from game $\mathbf{G}_{\mathbf{0}}^{\mathcal{F}}$, defined as the usual EUF-ID-CMA game. After receiving the master public key X as the challenge, the adversary $\mathcal{F}$ tries to output a forged signature under the public key X over a message-id pair that has never been queried before (Definition 4). Thus, for the first game, we have:

$$\Pr\left[\mathbf{G}_{0}^{\mathcal{F}} \Rightarrow 1\right] = \varepsilon. \tag{4}$$

Game $\mathbf{G}_{\mathbf{1}}^{\mathcal{F}}$. This game results from modifying game $\mathbf{G}_{\mathbf{0}}^{\mathcal{F}}$ and introducing an abort condition related to the probability of finding a hash collision during the FINALIZE procedure (Lines 00 to 07 of Figure 6). If adversary $\mathcal{B}_1$ (or $\mathcal{B}_2$) breaks the collision resistance of hash function H (or G), then the simulation aborts and outputs the symbol $\perp$. Let $\textsc{Coll}_{\mathsf{H}}$ (and $\textsc{Coll}_{\mathsf{G}}$) denote the event when a collision is found for hash function H (G, respectively).

The advantage of adversary $\mathcal{B}_1$ succeeding is equal to the probability of occurrence of event $\textsc{Coll}_{\mathsf{H}}$

$$\Pr\left[\mathcal{B}_1 \text{ wins}\right] = \Pr\left[\textsc{Coll}_{\mathsf{H}}\right].$$

For adversary $\mathcal{B}_2$, however, the advantage of succeeding depends on *not* finding a collision for hash function H, that is

$$\Pr\left[\mathcal{B}_2 \text{ wins}\right] = \Pr\left[\textsc{Coll}_{\mathsf{G}} \cap \overline{\textsc{Coll}_{\mathsf{H}}}\right].$$

Since games $\mathbf{G}_{\mathbf{0}}^{\mathcal{F}}$ and $\mathbf{G}_{\mathbf{1}}^{\mathcal{F}}$ are identical unless adversaries $\mathcal{B}_1$ or $\mathcal{B}_2$ succed, we can write

$$\Pr\left[\mathbf{G}_{\mathbf{0}}^{\mathcal{F}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{\mathbf{1}}^{\mathcal{F}} \Rightarrow 1\right] \leqslant \Pr\left[\textsc{Coll}_{\mathsf{H}} \cup \left(\textsc{Coll}_{\mathsf{G}} \cap \overline{\textsc{Coll}_{\mathsf{H}}}\right)\right],$$

and, by the union bound,

$$\Pr\left[\mathbf{G}_{\mathbf{0}}^{\mathcal{F}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{\mathbf{1}}^{\mathcal{F}} \Rightarrow 1\right] \leqslant \varepsilon_1 + \varepsilon_2. \tag{5}$$

The time taken by adversaries, $\mathcal{B}_1$ and $\mathcal{B}_2$, to break the collision resistance of the hash functions depends on the maximum number of queries allowed to the extraction

$\underline{\textsc{Init}(\lambda)}$
00 $\Pi : \mathbb{A} \mapsto \mathbb{B}$
01 $x \xleftarrow{\$} \mathbb{Z}_p^*$
02 $X = xG$
03 $\mathsf{mpk} \leftarrow X$
04 **return** mpk

$\underline{\boldsymbol{\Pi}(\alpha)}$
00 **return** $\Pi(\alpha)$

$\underline{\boldsymbol{\Pi}^{-1}(\beta)}$
00 **return** $\Pi(\alpha)$

$\underline{\mathbf{O}_{\mathsf{ext}}(\mathtt{id})}$
00 **if** $\langle \mathtt{id} \rangle \in \mathcal{Q}_{\mathsf{ext}}$:
01 $\quad \mathsf{sk}_{\mathtt{id}} \leftarrow (u_{\mathtt{id}}, t_{\mathtt{id}})$
02 $\quad$ **return** $\mathsf{sk}_{\mathtt{id}}$
03 $r \xleftarrow{\$} \mathbb{Z}_p$
04 $R \leftarrow rG$
05 **if** $R = \mathcal{O}$:
06 $\quad$ **abort**
07 $\alpha \leftarrow \varphi(R)$
08 $\beta \leftarrow \boldsymbol{\Pi}(\alpha)$
09 $t_{\mathtt{id}} \leftarrow \psi(\beta)$
10 **if** $t_{\mathtt{id}} = 0$:
11 $\quad$ **abort**
12 $c \leftarrow \mathsf{H}(\mathtt{id})$
13 $u_{\mathtt{id}} \leftarrow r^{-1}(c + t_{\mathtt{id}} x)$
14 **if** $u_{\mathtt{id}} = 0$:
15 $\quad$ **abort**
16 $\mathsf{sk}_{\mathtt{id}} = (u_{\mathtt{id}}, t_{\mathtt{id}})$
17 **return** $\mathsf{sk}_{\mathtt{id}}$

$\underline{\mathbf{O}_{\mathsf{sig}}(m, \mathtt{id})}$
00 **if** $\langle m, \mathtt{id} \rangle \in \mathcal{Q}_{\mathsf{sig}}$
01 $\quad \sigma \leftarrow \mathbf{O}_{\mathsf{sig}}(m, \mathtt{id})$
02 $\quad$ **return** σ
03 $\mathsf{sk}_{\mathtt{id}} \leftarrow \mathcal{O}_{\mathsf{ext}}(\mathtt{id})$
04 $(u_{\mathtt{id}}, t_{\mathtt{id}}) \leftarrow \mathsf{sk}_{\mathtt{id}}$
05 $X \leftarrow \mathsf{mpk}$
06 $c \leftarrow \mathsf{H}(\mathtt{id})$
07 $R \leftarrow u_{\mathtt{id}}^{-1}(cG + tX)$
08 $a \xleftarrow{\$} \mathbb{Z}_p^*$
09 $A \leftarrow aR$
10 **if** $A = \mathcal{O}$:
11 $\quad$ **abort**
12 $\alpha \leftarrow \varphi(A)$
13 $\beta \leftarrow \Pi(\alpha)$
14 $t = \psi(\beta)$
15 **if** $t = 0$:
16 $\quad$ **abort**
17 $d \leftarrow \mathsf{G}(m, \mathtt{id})$
18 $s \leftarrow a^{-1}(d + tu_{\mathtt{id}})$
19 **if** $s = 0$:
20 $\quad$ **abort**
21 $\sigma \leftarrow (s, t, R)$
22 **return** σ

$\underline{\textsc{Finalize}(\sigma^\star, m^\star, \mathtt{id}^\star, \mathsf{mpk})}$
00 **for** $\langle \mathtt{id} \rangle \in \mathcal{Q}_{\mathsf{ext}}$:
01 $\quad$ **if** $\mathsf{H}(\mathtt{id}^\star) = \mathsf{H}(\mathtt{id})$:
02 $\quad\quad$ **abort**
03 **endfor**
04 **for** $\langle m, \mathtt{id} \rangle \in \mathcal{Q}_{\mathsf{sig}}$:
05 $\quad$ **if** $\mathsf{G}(m^\star, \mathtt{id}^\star) = \mathsf{G}(m, \mathtt{id})$:
06 $\quad\quad$ **abort**
07 **endfor**
08 **if** $\langle \mathtt{id}^\star \rangle \in \mathcal{Q}_{\mathsf{ext}}$:
09 $\quad$ **abort**
10 **if** $\langle m^\star, \mathtt{id}^\star \rangle \in \mathcal{Q}_{\mathsf{sig}}$:
11 $\quad$ **abort**
12 $X \leftarrow \mathsf{mpk}$
13 $(s, t, R) \leftarrow \sigma^\star$
14 **if** $s = 0 \cup t = 0$:
15 $\quad$ **abort**
16 **if** $R = \mathcal{O}$:
17 $\quad$ **abort**
18 $\alpha \leftarrow \varphi(R)$
19 $\beta \leftarrow \Pi(\alpha)$
20 $t_{\mathtt{id}^\star} \leftarrow \psi(\beta)$
21 $c \leftarrow \mathsf{H}(\mathtt{id})$
22 $U \leftarrow cG + tX$
23 **if** $U = \mathcal{O}$:
24 $\quad$ **abort**
25 $d \leftarrow \mathsf{G}(m^\star, \mathtt{id}^\star)$
26 $A^\star \leftarrow s^{-1}(dR + tU)$
27 $\alpha^\star \leftarrow \varphi(A^\star)$
28 $\beta \leftarrow \Pi(\alpha^\star)$
29 **return** $t \stackrel{?}{=} \psi(\beta^\star)$

Figure 6: Specification of games $\mathbf{G}_{\mathbf{0}}^{\mathcal{F}}$ and $\mathbf{G}_{\mathbf{1}}^{\mathcal{F}}$.

and signing oracles. The extraction oracle is updated either when a direct query to the extraction oracle $\mathbf{O}_{\mathsf{ext}}$ is posed by adversary $\mathcal{B}_1$ or when an indirect query (Line 03 of the specification of oracle $\mathbf{O}_{\mathsf{sig}}$) is posed by adversary $\mathcal{B}_2$. On the contrary, the signing oracle is updated when $\mathcal{B}_2$ queries $\mathbf{O}_{\mathsf{sig}}$. Hence, the time taken to perform the simulation is given by

$$t_1 + t_2 \geqslant t + \mathcal{O}\left(q_{\mathsf{ext}} + q_{\mathsf{sig}}\right).$$

Game $\mathbf{G}_{\mathbf{2}}^{\mathcal{F}}$. In this game, the specification of the bijective random oracle is modified by considering $\boldsymbol{\Pi} = \boldsymbol{\Pi}^S \cup \boldsymbol{\Pi}^O$ instead of randomly defining a function from $\mathbb{A}$ to $\mathbb{B}$ (Line 00 of the specification depicted in Figure 6). We start the game initializing each list associated with oracles $\boldsymbol{\Pi}^O$ and $\boldsymbol{\Pi}^S$ with empty values (Line 00 in Figure 7). During the execution of this game, the number of elements in the table corresponding to the non-programmable part of the bijection, $\boldsymbol{\Pi}^O$, serves as a parameter to monitor the number of elements in the function's $\boldsymbol{\Pi}^S$ table. Additionally, we highlight that the list associated with $\boldsymbol{\Pi}^O$ stores the queries relayed to the corresponding challenger. This construction is an important resource for proving the security of our scheme, as it introduces an abort condition for the case where the functions $\boldsymbol{\Pi}^S$ and $\boldsymbol{\Pi}^O$ become contradictory.

Due to the programmable characteristics of the extraction and the signing oracles, only the values corresponding to $\mathbf{\Pi}^S$ are updated during queries to these entities. Conversely, the non-programmable part of the bijection is captured by the specification of the bijective random oracle in Definition 2. Figure 7 depicts the description of game $\mathbf{G}_{\mathbf{2}}^{\mathcal{F}}$ with the described changes.

$\textsc{Init}(\lambda)$
00 $(\Pi^O, \Pi^S) \leftarrow (\varnothing, \varnothing)$
01 $x \xleftarrow{\$} \mathbb{Z}_p^*$
02 $X = xG$
03 $\mathsf{mpk} \leftarrow X$
04 **return** mpk

$\mathbf{\Pi}(\alpha)$
00 **if** $(\alpha, \cdot) \in \Pi$
01 **return** $\Pi(\alpha)$
02 $\beta \xleftarrow{\$} \mathbb{B} \backslash \mathsf{Rng}(\Pi^O)$
03 **if** $(\cdot, \beta) \in \Pi^S$
04 **abort**
05 $\Pi^O \leftarrow \Pi^O \cup \{(\alpha, \beta)\}$
06 **return** β

$\mathbf{\Pi}^{-1}(\beta)$
00 **if** $(\cdot, \beta) \in \Pi$
01 **return** $\Pi^{-1}(\beta)$
02 $\alpha \xleftarrow{\$} \mathbb{A} \backslash \mathsf{Dom}(\Pi^O)$
03 **if** $(\alpha, \cdot) \in \Pi^S$
04 **abort**
05 $\Pi^O \leftarrow \Pi^O \cup \{(\alpha, \beta)\}$
06 **return** α

$\mathbf{O}_{\mathsf{ext}}(\mathtt{id}_i)$
00 **if** $\langle \mathtt{id}_i \rangle \in \mathcal{Q}_{\mathsf{ext}}$:
01 $\mathsf{sk}_{\mathtt{id}_i} \leftarrow (u_{\mathtt{id}_i}, t_{\mathtt{id}_i})$
02 **return** $\mathsf{sk}_{\mathtt{id}_i}$
03 $r_i \xleftarrow{\$} \mathbb{Z}_p$
04 $R_i \leftarrow r_i G$
05 **if** $R_i = \mathcal{O}$:
06 **abort**
07 $\alpha_i \leftarrow \varphi(R_i)$
08 **if** $(\alpha_i, \cdot) \in \Pi$:
09 **abort**
10 $\beta_i \xleftarrow{\$} \mathbb{B}$
11 **if** $(\cdot, \beta_i) \in \Pi$:
12 **abort**
13 $\Pi^S \leftarrow \Pi^S \cup \{(\alpha_i, \beta_i)\}$
14 $t_{\mathtt{id}_i} \leftarrow \psi(\beta_i)$
15 **if** $t_{\mathtt{id}_i} = 0$:
16 **abort**
17 $c_i \leftarrow \mathsf{H}(\mathtt{id}_i)$
18 $s_i \leftarrow c_i + t_{\mathtt{id}_i} x$
19 **if** $s_i = 0$:
20 **abort**
21 $u_{\mathtt{id}_i} \leftarrow s_i / r_i$
22 $\mathsf{sk}_{\mathtt{id}} = (u_{\mathtt{id}}, t_{\mathtt{id}})$
23 **return** $\mathsf{sk}_{\mathtt{id}}$

$\mathbf{O}_{\mathsf{sig}}(m_j, \mathtt{id}_j)$
00 **if** $\langle m_j, \mathtt{id}_j \rangle \in \mathcal{Q}_{\mathsf{sig}}$
01 $\sigma_j \leftarrow \mathbf{O}_{\mathsf{sig}}(m_j, \mathtt{id}_j)$
02 **return** σ_j
03 $\mathsf{sk}_{\mathtt{id}_j} \leftarrow \mathcal{O}_{\mathsf{ext}}(\mathtt{id}_j)$
04 $(u_{\mathtt{id}_j}, t_{\mathtt{id}_j}) \leftarrow \mathsf{sk}_{\mathtt{id}_j}$
05 $X \leftarrow \mathsf{mpk}$
06 $c_j \leftarrow \mathsf{H}(\mathtt{id}_j)$
07 $R_j \leftarrow u_{\mathtt{id}_j}^{-1}(c_j G + t_{\mathtt{id}_j} X)$
08 $a_j \xleftarrow{\$} \mathbb{Z}_p^*$
09 $A_j \leftarrow a_j R_j$
10 **if** $A_j = \mathcal{O}$:
11 **abort**
12 $\alpha'_j \leftarrow \varphi(A_j)$
13 **if** $(\alpha'_j, \cdot) \in \Pi$:
14 **abort**
15 $\beta'_j \xleftarrow{\$} \mathbb{B}$
16 **if** $(\cdot, \beta'_j) \in \Pi$:
17 **abort**
18 $\Pi^S \leftarrow \Pi^S \cup \{(\alpha'_j, \beta'_j)\}$
19 $t_j = \psi(\beta'_j)$
20 **if** $t_j = 0$:
21 **abort**
22 $d_j \leftarrow \mathsf{G}(m_j, \mathtt{id}_j)$
23 $s'_j \leftarrow d_j + t_j u_{\mathtt{id}_j}$
24 **if** $s'_j = 0$:
25 **abort**
26 $s_j \leftarrow s'_j / a_j$
27 $\sigma \leftarrow (s_j, t_j, R_j)$
28 **return** σ_j

Figure 7: Specification of the oracles used during game $\mathbf{G}_2^{\mathcal{F}}$ with changes with respect to $\mathbf{G}_1^{\mathcal{F}}$ highlighted. We omit the specification of the FINALIZE procedure because it remains unchanged.

The simulation of game $\mathbf{G}_{\mathbf{2}}^{\mathcal{F}}$ aborts if the conditions portrayed in one of the following lines hold: Line 03 of $\mathbf{\Pi}$ and Line 03 of $\mathbf{\Pi}^{-1}$, Lines 08 and 11 of the extraction oracle $\mathbf{O}_{\mathsf{ext}}$, and Lines 13 or 16 of the signing oracle $\mathbf{O}_{\mathsf{sig}}$. We analyse each of the three abort events, S_1 to S_3, separately.

Event S_1. When the forger $\mathcal{F}$ queries the bijective random oracle, $\mathbf{bRO} = (\mathbf{\Pi}, \mathbf{\Pi}^{-1})$, directly or implicitly (i.e., query to the extraction or signing oracle), the number of elements present in the list Π^O corresponds to $Q = q_{\mathsf{ext}} + q_{\mathsf{sig}} + q_{\mathbf{\Pi}}$. Since the list Π is defined as $\Pi^O \cup \Pi^S$, the number of elements in Π^S is equals to $|\mathbb{B}| - Q = 2^\lambda - Q$, since $\mathbb{B} = [0, 2^\lambda - 1]$. Elements are added to the list Π^S in Line 13 of the extraction oracle and in Line 18 of the

signature oracle. Since only one element is added per query, we have a total of $q_{\mathsf{ext}} + q_{\mathsf{sig}}$ elements. Hence, for a total execution of $q_{\mathbf{\Pi}}(q_{\mathsf{ext}} + q_{\mathsf{sig}})$, the probability of occurrence of event S_1 is given by

$$\Pr[\mathsf{S}_1] \leqslant \frac{q_{\mathbf{\Pi}}\,(q_{\mathsf{ext}} + q_{\mathsf{sig}})}{2^{\lambda} - Q}. \tag{6}$$

Event S_2. To bound the abort probability for this event, we start with the condition depicted in Line 11 of the extraction oracle. There are a total of Q elements in the list Π and a maximum of q_{ext} queries to the extraction oracle, totalling Qq_{ext} elements present in a list of size $|\mathbb{B}| = 2^{\lambda}$. That is, a probability of occurrence bounded by $Qq_{\mathsf{ext}}/2^{\lambda}$.

For the event depicted in Line 08, we have to make use of the semi-injective property of the function φ. For the same number of elements, Qq_{ext}, we now consider a list of size $(p-1)/2$ and a probability bounded by $2Qq_{\mathsf{ext}}/(p-1)$. Hence, by the union bound,

$$\Pr[\mathsf{S}_2] \leqslant \frac{Qq_{\mathsf{ext}}}{2^{\lambda}} + \frac{2Qq_{\mathsf{ext}}}{p-1} \tag{7}$$

Event S_3. Notice that the same analysis conducted for event S_2 applies to event S_3. Lines 13 and 16 of the signature oracle correspond to Lines 08 and 11 of the extraction oracle, respectively. The only change is that the number of queries is bounded by q_{sig}, and not q_{ext}. Hence, due to the symmetry between S_2 and S_3, we have, by the union bound,

$$\Pr[\mathsf{S}_3] \leqslant \frac{Qq_{\mathsf{sig}}}{2^{\lambda}} + \frac{2Qq_{\mathsf{sig}}}{p-1} \tag{8}$$

Combining Equations 6, 7, and 8, using $p \leqslant 2^{\lambda+1} + 1$, and $q_{\mathbf{\Pi}} \leqslant Q$, we have

$$\Pr\left[\mathbf{G}_1^{\mathcal{F}} \Rightarrow 1\right] \leqslant \Pr\left[\mathbf{G}_2^{\mathcal{F}} \Rightarrow 1\right] + \frac{3Q\,(q_{\mathsf{ext}} + q_{\mathsf{sig}})}{(p-1)/2 - Q}. \tag{9}$$

Game $\mathbf{G}_3^{\mathcal{F}}$. From the specifications of Game 3 in Figure 8 and Game 2 in Figure 7 the difference resides in the different moments in which the abort can happen. In Game 2, the α_i value if defined accordingly to $\varphi(R_i)$ and then β_i is sampled over $\mathbb{B}$. In Game 3, the order of operations is inverted. Since the values of α and β are randomly chosen, both Games 2 and 3 are indistinguishable, allowing us to write:

$$\Pr\left[\mathbf{G}_3^{\mathcal{F}} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_2^{\mathcal{F}} \Rightarrow 1\right] \tag{10}$$

Forger $\mathcal{F}_0$. Game 3 relates to the case where $\mathcal{F}_0$ tries to break the key-only unforgeability of our scheme. The argument is analogue to the one in [FKP16]: assume $\mathcal{F}$ produces a forgery $\sigma^{\star} = (s^{\star}, t^{\star}, R^{\star})$, i.e., $s^{\star}a^{\star} = d^{\star} + t^{\star}u^{\star}$, on $\langle \mathtt{id}^{\star}, m^{\star} \rangle$. Then, we have two cases: $(\alpha'^{\star}, \beta'^{\star}) \in \Pi^S$ or $(\alpha'^{\star}, \beta'^{\star}) \in \Pi^O$. In the first case, the pair was defined in lines 22 or 25 of the extraction or signing oracle, respectively. Meaning, $\mathcal{F}_0$ knows a signature $\sigma_i = (s_i, t_i, R_i)$ such that the conditions of Equation 2 occur, and $\mathcal{F}_0$ extracts the user's secret key, thus winning the game. The case when the forged pair $(\alpha, \beta) \in \Pi^O$ follows with little adaptation, since Π^O contains the list of elements updated by querying the bijective random oracle. This means, $\mathcal{F}$ succeeded and so did $\mathcal{F}_0$, i.e.,

$$\Pr\left[\mathbf{G}_3^{\mathcal{F}} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_0^{\mathcal{F}_0} \Rightarrow 1\right]. \tag{11}$$

Combining Equations 4 to 11, we obtain the desired result. □

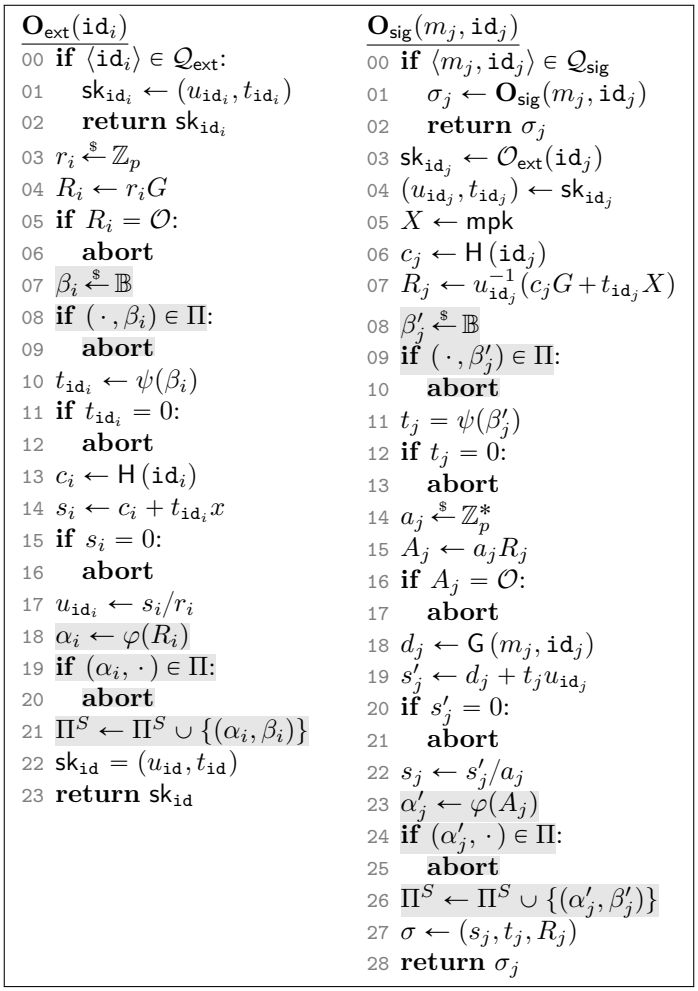

Figure 8: Specification of $\mathbf{G}_3^{\mathcal{F}}$ resulted from changes with respect to game $\mathbf{G}_2^{\mathcal{F}}$.

Theorem 2. *In the bijective random oracle model, if there exists an p.p.t adversary $\mathcal{F}_0$ that $(t, \varepsilon, q_{\mathsf{ext}}, q_{\mathbf{\Pi}})$–breaks the unforgeability under identity and key-only attack of* $\mathrm{IBS}_{\mathrm{ecdsa}}$*, then there exists a p.p.t. adversaries $\mathcal{R}$ that $(t_1, \varepsilon_1, q_{\mathsf{sig}}, q_{\mathbf{\Pi}})$–breaks the unforgeability under chosen-message attack of* ECDSA *and $\mathcal{B}$ that $(t_{\mathrm{dr}}, \varepsilon_{\mathrm{dr}})$–breaks the division resistance of hash function* G *where*

$$\varepsilon \leqslant \sqrt{2q_{\mathbf{\Pi}}\left(\varepsilon_1 + q_{\mathbf{\Pi}}\varepsilon_{\mathrm{dr}}\right)} + \varepsilon_1 + \frac{\left(q_{\mathbf{\Pi}} + q_{\mathsf{ext}}\right)^2}{2^{\lambda}},$$

with $t \geqslant t_1 + \mathcal{O}\left(q_{\mathbf{\Pi}} + q_{\mathsf{ext}}\right)$, $t \geqslant t_{\mathrm{dr}} + \mathcal{O}(q_{\mathbf{\Pi}})$, and $q_{\mathsf{sig}} \approx q_{\mathsf{ext}}$.

Again, before starting the proof of Theorem 2, we discuss some of the main ideas and difficulties found when modelling the security proof. To avoid the programmability of the extraction oracle by $\mathcal{R}$ when answering extraction queries posed by $\mathcal{F}$, we set the extraction oracle $\mathbf{O}_{\mathsf{ext}}$ to be a non-programmable oracle. This is because, to correctly simulate the extraction oracle, the forger $\mathcal{R}$ must somehow know the user's secret key a priori, which is considered impossible in practice.

Another important observation is that solely considering the non-programmability of the extraction oracle is insufficient to design a reduction that breaks, for instance, the

UF-KOA security of ECDSA, since the extraction oracle itself would behave as a signing oracle, thus violating Definition 5. Instead, we design the reduction to use the forger $\mathcal{F}_0$ to break the EUF-CMA security, because, in such a case, the conditions described in Definitions 4 (without the extraction oracle) and 5 would be fulfilled.

The reduction algorithm $\mathcal{R}$ does not possess any control over how the extraction oracle behaves, but it is capable of monitoring every query $(\mathtt{id}, \mathbf{O}_{\mathsf{ext}}(\mathtt{id}))$. When $\mathcal{F}$ queries the extraction oracle on $\mathtt{id}$, it receives the user's secret key associated with that identity, $\mathsf{sk}_{\mathtt{id}} = (u_{\mathtt{id}}, t_{\mathtt{id}})$. Algorithm $\mathcal{R}$ stores the queried and returned values, $\langle \mathtt{id}, u_{\mathtt{id}}, t_{\mathtt{id}} \rangle$, in a list, $\mathcal{Q}_{\mathsf{ext}}$. For the algorithm $\mathcal{R}$ to win the EUF-CMA game and output an ECDSA signature $\mathsf{sk}_{\mathtt{id}^\star} = (u_{\mathtt{id}^\star}, t_{\mathtt{id}^\star})$, the algorithm $\mathcal{F}_0$ must output $(m^\star, \mathtt{id}^\star, (s, t, R))$, with $\sigma^\star = (s, t, R)$ being the forged signature. Once it succeeds, the reduction $\mathcal{R}$ rewinds the oracle **bRO** and the algorithm $\mathcal{F}_0$ attempting to obtain a second forgery $\bar{\sigma}^\star = \left(\bar{s}, \bar{t}, R\right)$ over message $\bar{m}^\star$ and identity $\mathtt{id}^\star$. Finally, the reduction $\mathcal{R}$ uses the information from $\sigma^\star$ and $\bar{\sigma}^\star$ to derive $u_{\mathtt{id}^\star}$ and answers its own challenger with $(u_{\mathtt{id}^\star}, t_{\mathtt{id}^\star})$. At last, we use the lazy sampling technique along the proof of the theorem, we consider that the reduction knows the number of queries $q_{\mathbf{\Pi}}$.

Proof. The first game, $\mathbf{G}_0^{\mathcal{F}_0}$, corresponds to the chosen identity and key-only attack. Upon receiving the challenge value $\mathsf{mpk} = X$, the forger $\mathcal{F}_0$ tries to output a valid forged signature on a message m and an identity $\mathtt{id}^\star$ (not present in previous queries) by posing queries to the extraction oracle $\mathbf{O}_{\mathsf{ext}}$, that is

$$\Pr\left[\mathbf{G}_0^{\mathcal{F}_0} \Rightarrow 1\right] = \varepsilon. \tag{12}$$

Game $\mathbf{G}_1^{\mathcal{F}_0}$. Next, we introduce the abort condition during queries to the bijective random oracle and to the extraction oracle, implementing Π with lazy sampling [BR04]. In this case, we define two lists, $\mathbf{\Pi}_{\rightarrow}$ and $\mathbf{\Pi}_{\leftarrow}$, which store, respectively, α and β values. Analogously to Theorem 1, $\mathbf{\Pi}$ is defined as $\mathbf{\Pi}_{\rightarrow} \cup \mathbf{\Pi}_{\leftarrow}$, with abort conditions occurring when the two lists contradict each other.

However, instead of arguing that Π is composed of lists storing values obtained by means (or not) of programming the oracle, now, the notion of direct queries and inverse queries (left and right arrow subscripts, correspondingly). As will be discussed further, this is done to allow the use of the General Forking Lemma (Theorem 2) to rewind the oracle and obtain two forgeries as discussed in the prelude of the proof. We evaluate the abort probability in game $\mathbf{G}_1^{\mathcal{F}_0}$ accordingly to the abort condition in lines 5 in the specification of both $\mathbf{\Pi}$ and $\mathbf{\Pi}^{-1}$ (Figure 9). Notice that the bijective random oracle can be implicitly queried when the extraction oracle, hence an additional number of queries q_{ext} needs to be taken into account. In both cases, the probability of finding an α (or β) value already defined is bounded by $(q_{\mathbf{\Pi}} + q_{\mathsf{ext}})/|\mathbb{B}|$. By a simple counting argument, for a maximum of $q_{\mathbf{\Pi}} + q_{\mathsf{ext}}$ queries posed to the bijective random oracle, we can write

$$\Pr\left[\mathbf{G}_0^{\mathcal{F}_0} \Rightarrow 1\right] \leqslant \Pr\left[\mathbf{G}_1^{\mathcal{F}_0} \Rightarrow 1\right] + \frac{\left(q_{\mathbf{\Pi}} + q_{\mathsf{ext}}\right)^2}{2^\lambda}. \tag{13}$$

Game $\mathbf{G}_2^{\mathcal{F}_0}$. In this game, we introduce two changes. The first one is that the values of β are sampled a priori (Line 01 of the $\textsc{Init}(\lambda)$ procedure depicted in Figure 10) and spent in each query to $\mathbf{\Pi}$ if it does not abort, i.e., the queried value α is not an element of the list Π. The second is that the values α are resampled if they are already an element of the pre-image of the function φ (Lines 03 to 05 of the specification of oracle $\mathbf{\Pi}^{-1}$) when $\mathbf{\Pi}^{-1}$ is queried on value β.

Next, we argue that the distribution of games $\mathbf{G}_2^{\mathcal{F}_0}$ and $\mathbf{G}_1^{\mathcal{F}_0}$ are indistinguishable. This follows from two observations. The first is that the only difference between the two games with respect to the sampling of values β is the moment when they occur, since in

$\underline{\textsc{Init}(\lambda)}$
00 $(\Pi_{\rightarrow}, \Pi_{\leftarrow}) \leftarrow (\varnothing, \varnothing)$
01 $x \xleftarrow{\$} \mathbb{Z}_p^*$
02 $X = xG$
03 $\mathsf{mpk} \leftarrow X$
04 **return** mpk

$\underline{\boldsymbol{\Pi}(\alpha)}$
00 **if** $(\alpha, \cdot) \in \Pi$
01 **return** $\Pi(\alpha)$
02 $\beta \xleftarrow{\$} \mathbb{B}$
03 $(\cdot, \beta) \in \Pi$
04 **abort**
05 $\Pi_{\rightarrow} \leftarrow \Pi_{\rightarrow} \cup \{(\alpha, \beta)\}$
06 **return** β

$\underline{\boldsymbol{\Pi}^{-1}(\beta)}$
00 **if** $(\cdot, \beta) \in \Pi$
01 **return** $\Pi^{-1}(\beta)$
02 $\alpha \xleftarrow{\$} \mathbb{A}$
03 **if** $(\alpha, \cdot) \in \Pi$
04 **abort**
05 $\Pi_{\leftarrow} \leftarrow \Pi_{\leftarrow} \cup \{(\alpha, \beta)\}$
06 **return** α

$\underline{\textsc{Finalize}(\sigma^\star, m^\star, \mathtt{id}^\star, \mathsf{mpk})}$
00 $X \leftarrow \mathsf{mpk}$
01 $(s^\star, t^\star, R^\star) \leftarrow \sigma^\star$
02 **if** $s^\star = 0 \cup t^\star = 0$:
03 **abort**
04 **if** $R = \mathcal{O}$:
05 **abort**
06 $\alpha^\star \leftarrow \varphi(R^\star)$
07 $\beta^\star \leftarrow \Pi(\alpha^\star)$
08 $t_{\mathtt{id}^\star} \leftarrow \psi(\beta^\star)$
09 **if** $t_{\mathtt{id}^\star} = 0$:
10 **abort**
11 $c^\star \leftarrow \mathsf{H}(\mathtt{id}^\star)$
12 $d^\star \leftarrow \mathsf{G}(m^\star, \mathtt{id}^\star)$
13 $U^\star \leftarrow c^\star G + t_{\mathtt{id}^\star} X$
14 $A^\star \leftarrow (s^\star)^{-1}(d^\star R^\star + t^\star U^\star)$
15 **if** $A^\star = \mathcal{O}$:
16 **abort**
17 $\alpha'^\star \leftarrow \varphi(A^\star)$
18 $\beta'^\star \leftarrow \Pi(\alpha'^\star)$
19 **if** $\psi(\beta'^\star) \neq t^\star$:
20 **return false**
21 **return true**

Figure 9: Games $\mathbf{G}_0^{\mathcal{F}_0}$ and $\mathbf{G}_1^{\mathcal{F}_0}$ during the key-only experiment. Lines highlighted by gray colored boxes represents changes between game $\mathbf{G}_1^{\mathcal{F}_0}$ when compared to game $\mathbf{G}_0^{\mathcal{F}_0}$. We omit the specification of the extraction oracle $\mathbf{O}_{\mathsf{ext}}(\mathtt{id})$ because it is exactly the same as in Figure 7 with $\Pi_O = \Pi_{\rightarrow}$ and $\Pi_S = \Pi_{\leftarrow}$.

both cases, they are uniformly sampled over the set $\mathbb{B} = \left[0,\, 2^\lambda - 1\right]$, their distribution is the same. If during game $\mathbf{G}_1^{\mathcal{F}_0}$ they are sampled based on demand (Line 02 of Figure 9), during game $\mathbf{G}_2^{\mathcal{F}_0}$ they are only consumed, since $q_{\boldsymbol{\Pi}}$ values β are sampled during the $\textsc{Init}$ procedure. For the second change, the indistinguishability between the two distributions follows from the semi-injectiveness of the function φ. Hence, we conclude that

$$\Pr\left[\mathbf{G}_2^{\mathcal{F}_0} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_1^{\mathcal{F}_0} \Rightarrow 1\right]. \tag{14}$$

If the forger succeeds and produces a forgery on a backward query $(\alpha'^\star, \beta'^\star) \in \Pi_{\leftarrow}$, then, due to the resampling step during queries to $\boldsymbol{\Pi}^{-1}$, the forger knows the discrete logarithm a a priori. This means being capable of solving the equation

$$sA = dR + tU,$$

and extracting the discrete logarithm

$$u_{\mathtt{id}^\star} = \frac{s^\star \eta a - d^\star}{t^\star} \bmod p,$$

with $|\eta| = 1$ and $d^\star = \mathsf{G}(m, \mathtt{id}^\star)$.

From the forged signature $\sigma^\star$, the reduction $\mathcal{R}$ forwards the ECDSA signature $\mathsf{sk}_{\mathtt{id}^\star} = (u_{\mathtt{id}^\star}, t_{\mathtt{id}^\star})$ to its own challenger. The nature of the forgery $\sigma^\star = (s^\star, t^\star, R^\star)$ in game $\mathbf{G}_2^{\mathcal{F}_0}$ is restricted to queries made backwards. In what follows, game $\mathbf{G}_3^{\mathcal{F}_0}$ tackles the case when the forger succeeds with queries made forwards.

$\underline{\textsc{Init}(\lambda)}$
00 $(\Pi_{\rightarrow}, \Pi_{\leftarrow}) \leftarrow (\varnothing, \varnothing)$
01 $(\beta_i)_{i \in [q_{\Pi}]} \xleftarrow{\$} \mathbb{B}$
02 $x \xleftarrow{\$} \mathbb{Z}_p^*$
03 $X = xG$
04 $\mathsf{mpk} \leftarrow X$
05 **return** mpk

$\underline{\boldsymbol{\Pi}(\alpha)}$
00 **if** $(\alpha, \cdot) \in \Pi$
01 **return** $\Pi(\alpha)$
02 $k \leftarrow k + 1$
03 $\beta \leftarrow \beta_k$
04 $\beta \xleftarrow{\$} \mathbb{B}$
05 $(\cdot, \beta) \in \Pi$
06 **abort**
07 $\Pi_{\rightarrow} \leftarrow \Pi_{\rightarrow} \cup \{(\alpha, \beta)\}$
08 **return** β

$\underline{\boldsymbol{\Pi}^{-1}(\beta)}$
00 **if** $(\cdot, \beta) \in \Pi$
01 **return** $\Pi^{-1}(\beta)$
02 $\alpha \xleftarrow{\$} \mathbb{A}$
03 **if** $\alpha \in \varphi(\mathbb{G}^*)$:
04 $a \xleftarrow{\$} \mathbb{Z}_p^*$
05 $\alpha \leftarrow \varphi(aG)$
06 **if** $(\alpha, \cdot) \in \Pi$
07 **abort**
08 $\Pi_{\leftarrow} \leftarrow \Pi_{\leftarrow} \cup \{(\alpha, \beta)\}$
09 **return** α

$\underline{\textsc{Finalize}(\sigma^\star, m^\star, \mathtt{id}^\star, \mathsf{mpk})}$
00 $X \leftarrow \mathsf{mpk}$
01 $(s^\star, t^\star, R^\star) \leftarrow \sigma^\star$
02 **if** $s^\star = 0 \cup t^\star = 0$:
03 **abort**
04 **if** $R = \mathcal{O}$:
05 **abort**
06 $\alpha^\star \leftarrow \varphi(R^\star)$
07 $\beta^\star \leftarrow \Pi(\alpha^\star)$
08 $t_{\mathtt{id}^\star} \leftarrow \psi(\beta^\star)$
09 **if** $t_{\mathtt{id}^\star} = 0$:
10 **abort**
11 $c^\star \leftarrow \mathsf{H}(\mathtt{id}^\star)$
12 $d^\star \leftarrow \mathsf{G}(m^\star, \mathtt{id}^\star)$
13 $U^\star \leftarrow c^\star G + t_{\mathtt{id}^\star} X$
14 $A^\star \leftarrow (s^\star)^{-1}(d^\star R^\star + t^\star U^\star)$
15 **if** $A^\star = \mathcal{O}$:
16 **abort**
17 $\alpha'^\star \leftarrow \varphi(A^\star)$
18 $\beta'^\star \leftarrow \Pi(\alpha'^\star)$
19 **if** $\psi(\beta'^\star) \neq t^\star$:
20 **return false**
21 **return true**

Figure 10: Specifications of game $\mathbf{G}_1^{\mathcal{F}_0}$ and game $\mathbf{G}_2^{\mathcal{F}_0}$ with highlighted changes.

Game $\mathbf{G}_3^{\mathcal{F}_0}$. By the reasoning previously shown, the forger $\mathcal{F}$ succeeds if and only if an attacker obtains the user's secret key on the target identity $\mathtt{id}^\star$ with a query made backwards. This means that the winning condition of games $\mathbf{G}_3$ and $\mathbf{G}_2$ relates as follows:

$$\Pr\left[\mathbf{G}_2^{\mathcal{F}_0} \Rightarrow 1\right] \leqslant \Pr\left[\mathbf{G}_3^{\mathcal{F}_0} \Rightarrow 1\right] + \varepsilon_1. \tag{15}$$

When the forger succeeds with a query made forwards, the probability is given by the acc as in Theorem 2. Combining inequalities 12 to 15, we obtain

$$\Pr\left[\mathbf{G}_3^{\mathcal{F}_0} \Rightarrow 1\right] \geqslant \varepsilon - \varepsilon_1 - \frac{(q_{\boldsymbol{\Pi}} + q_{\mathsf{ext}})^2}{2^\lambda} \tag{16}$$

Line 20 highlights the abort when values have already been queried backwards, which is the case of game $\mathbf{G}_2^{\mathcal{F}_0}$. The reduction $\mathcal{R}$ uses the forger $\mathcal{F}_0$ with inputs the master public key $\mathsf{mpk} = X$, the values β_i, for $i \in [q_{\boldsymbol{\Pi}}]$, and random coins ρ. If the forger does not abort, it outputs the target index $i^\star \in [q_{\boldsymbol{\Pi}}]$ together with the forgery $(m^\star, \mathtt{id}^\star, (s^\star, t^\star, R^\star))$ on some message $m^\star$ and the target identity $\mathtt{id}^\star = \mathtt{id}_{i^\star}$.

To bound the abort probability, we follow the same method of the GenDSA framework and consider a two-stage stateful adversary $\mathcal{B}$ attempting to break the ψ-relative division resistance of the hash function G, but with an additional constraint. We impose that the

$\underline{\textsc{Init}(\lambda)}$
00 $(\Pi_{\rightarrow}, \Pi_{\leftarrow}) \leftarrow (\varnothing, \varnothing)$
01 $(\beta_i)_{i\in[q_{\Pi}]} \xleftarrow{\$} \mathbb{B}$
02 $x \xleftarrow{\$} \mathbb{Z}_p^*$
03 $X = xG$
04 $\mathsf{mpk} \leftarrow X$
05 **return** mpk

$\underline{\boldsymbol{\Pi}(\alpha)}$
00 **if** $(\alpha, \cdot) \in \Pi$
01 **return** $\Pi(\alpha)$
02 $k \leftarrow k + 1$
03 $\beta \leftarrow \beta_k$
04 $\beta \xleftarrow{\$} \mathbb{B}$
05 $(\cdot, \beta) \in \Pi$
06 **abort**
07 $\Pi_{\rightarrow} \leftarrow \Pi_{\rightarrow} \cup \{(\alpha, \beta)\}$
08 **return** β

$\underline{\boldsymbol{\Pi}^{-1}(\beta)}$
00 **if** $(\cdot, \beta) \in \Pi$
01 **return** $\Pi^{-1}(\beta)$
02 $\alpha \xleftarrow{\$} \mathbb{A}$
03 **if** $\alpha \in \varphi(\mathbb{G}^*)$:
04 $a \xleftarrow{\$} \mathbb{Z}_p^*$
05 $\alpha \leftarrow \varphi(aG)$
06 **if** $(\alpha, \cdot) \in \Pi$
07 **abort**
08 $\Pi_{\leftarrow} \leftarrow \Pi_{\leftarrow} \cup \{(\alpha, \beta)\}$
09 **return** α

$\underline{\textsc{Finalize}(\sigma^\star, m^\star, \mathtt{id}^\star, \mathsf{mpk})}$
00 $X \leftarrow \mathsf{mpk}$
01 $(s^\star, t^\star, R^\star) \leftarrow \sigma^\star$
02 **if** $s^\star = 0 \cup t^\star = 0$:
03 **abort**
04 **if** $R = \mathcal{O}$:
05 **abort**
06 $\alpha^\star \leftarrow \varphi(R^\star)$
07 $\beta^\star \leftarrow \Pi(\alpha^\star)$
08 $t_{\mathtt{id}^\star} \leftarrow \psi(\beta^\star)$
09 **if** $t_{\mathtt{id}^\star} = 0$:
10 **abort**
11 $c^\star \leftarrow \mathsf{H}(\mathtt{id}^\star)$
12 $d^\star \leftarrow \mathsf{G}(m^\star, \mathtt{id}^\star)$
13 $U^\star \leftarrow c^\star G + t_{\mathtt{id}^\star} X$
14 $A^\star \leftarrow (s^\star)^{-1}(d^\star R^\star + t^\star U^\star)$
15 **if** $A^\star = \mathcal{O}$:
16 **abort**
17 $\alpha'^\star \leftarrow \varphi(A^\star)$
18 $\beta'^\star \leftarrow \Pi(\alpha'^\star)$
19 **if** $\psi(\beta'^\star) \neq t^\star$:
20 **if** $(\alpha'^\star, \beta'^\star) \in \Pi_{\leftarrow}$:
21 **abort**
22 **for** $j \in [k]$:
23 **if** $\beta'^\star = \beta'_j$:
24 **return** $(j, (m^\star, \mathtt{id}^\star, \sigma^\star))$

Figure 11: Specification of games $\mathbf{G}_2^{\mathcal{F}_0}$ and $\mathbf{G}_3^{\mathcal{F}_0}$.

point $R^\star$ appears in both forgeries. This means obtaining two forgeries $\sigma^\star = (s^\star, t^\star, R^\star)$ and $\bar{\sigma}^\star = (\bar{s}^\star, \bar{t}^\star, R^\star)$ over $m^\star||\mathtt{id}^\star$ and $\bar{m}^\star||\mathtt{id}^\star$, correspondingly. The specification of the algorithms is depicted in Figure 12. Algorithm $\mathcal{W}_{\mathcal{F}_0}$ is a wrapper for algorithm $\mathcal{F}_0$, which succeeds when it outputs two valid signatures, which, in turn, lead us to $\Pr\left[\mathbf{G}_3^{\mathcal{F}_0} \Rightarrow 1\right] = \mathsf{acc}$.

Hence, for $t^\star = f(A^\star)$ and $\bar{t}^\star = f(\bar{A}^\star)$, we have $\bar{A}^\star = \eta A^\star$ by the semi-injective property of the function φ. For digests $d^\star = \mathsf{G}(m^\star, \mathtt{id}^\star)$ and $\bar{d}^\star = \mathsf{G}(\bar{m}^\star, \mathtt{id}^\star)$, we first extract the discrete logarithm of $A^\star$ with respect to $R^\star$

$$a^\star = \frac{\bar{d}^\star - d^\star}{\eta\bar{s}^\star - s^\star},$$

as long as adversary $\mathcal{B}$ does not solve the ψ-relative division of G, i.e., obtain $s^\star = \eta\bar{s}^\star$, which would render the obtention of $a^\star$ impossible. Then, the reduction can extract the user's secret key

$$u_{\mathtt{id}^\star} = \frac{\eta\bar{s}^\star d^\star - s^\star\bar{d}^\star}{t^\star(s^\star - \eta\bar{s}^\star)}, \tag{17}$$

and forwards $\mathsf{sk}_{\mathtt{id}^\star} = (u_{\mathtt{id}^\star}, t_{\mathtt{id}^\star})$ to its challenger, winning the EUF-CMA game. Overall, the probability of returning the $u_{\mathtt{id}^\star}$ is given by $\varepsilon_1 \geqslant \mathsf{gfrk} - q_{\boldsymbol{\Pi}}\varepsilon_{\mathrm{dr}}$. Thus, from the General

Reduction $\mathcal{W}_{\mathcal{F}_0}(X)$

$\rho \xleftarrow{\$} \mathbb{S}$
$(\beta_i)_{i\in[q_{\mathbf{\Pi}}]} \xleftarrow{\$} \mathbb{B}$
$\mathcal{L} \leftarrow \{\beta_1, \ldots, \beta_{q_{\mathbf{\Pi}}}\}$
$(i^\star, (m^\star, \mathtt{id}^\star, (s^\star, t^\star, R^\star)) \leftarrow \mathcal{W}_{\mathcal{F}_0}(X, \mathcal{L}; \rho)$
$\bar{\beta}_{i^\star}, \ldots, \bar{\beta}_{q_{\mathbf{\Pi}}} \xleftarrow{\$} \mathbb{B}$
if $\bar{\beta}_{i^\star} = \beta_{i^\star}$
  abort
$\mathcal{L} \leftarrow \{\beta_1, \ldots, \beta_{\mathtt{id}^\star - 1}\}$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{\bar{\beta}_{\mathtt{id}^\star}, \ldots, \bar{\beta}_{q_{\mathbf{\Pi}}}\}$
$(\bar{i}^\star, (\bar{m}^\star, \bar{\mathtt{id}}^\star, (\bar{s}^\star, \bar{t}^\star, R^\star)) \leftarrow \mathcal{W}_{\mathcal{F}_0}(X, \mathcal{L}; \rho)$
if $\bar{i}^\star \neq i^\star$
  abort
$u_{\mathtt{id}^\star} \leftarrow \dfrac{\eta\bar{s}^\star d^\star - s^\star \bar{d}^\star}{t^\star(s^\star - \eta\bar{s}^\star)}$
$t_{\mathtt{id}^\star} \leftarrow f(R^\star)$
return $(u_{\mathtt{id}^\star}, t_{\mathtt{id}^\star})$

Figure 12: The adversary $\mathcal{R}$ uses the wrapper $\mathcal{W}_{\mathcal{F}_0}$ to forge two signatures and derive the user's secret key and present it as part of the forged ECDSA signature.

Forking Lemma, we have

$$\varepsilon_1 + q_{\mathbf{\Pi}}\varepsilon_{\mathrm{dr}} \geqslant \mathsf{acc}\left(\frac{\mathsf{acc}}{q_{\mathbf{\Pi}}} - \frac{1}{2^\lambda}\right). \tag{18}$$

From Equation 18, we can bound the value ε by considering two cases:

Case 1. If $\dfrac{\mathsf{acc}}{2q_{\mathbf{\Pi}}} < 2^{-\lambda} < \dfrac{\mathsf{acc}}{q_{\mathbf{\Pi}}}$, then

$$\varepsilon \leqslant \varepsilon_1 + \frac{(q_{\mathbf{\Pi}} + q_{\mathsf{ext}})^2 + 2q_{\mathbf{\Pi}}}{2^\lambda}.$$

Case 2. If $\dfrac{\mathsf{acc}}{2q_{\mathbf{\Pi}}} \geqslant 2^{-\lambda}$, then

$$\varepsilon \leqslant \sqrt{2q_{\mathbf{\Pi}}\left(\varepsilon_1 + q_{\mathbf{\Pi}}\varepsilon_{\mathrm{dr}}\right)} + \varepsilon_1 + \frac{(q_{\mathbf{\Pi}} + q_{\mathsf{ext}})^2}{2^\lambda}$$

□

6 Analysis

In this section, we present the theoretical analysis of our ECDSA-based IBS (and variants), and also evaluate the empirical performance of the implementation of our scheme compared to the concrete instantiation of [GG09]. We begin with an analysis of the corresponding algorithms (Tables 1 and 2), evaluating the number of operations performed by each scheme. Next, for each scheme, we compare the signature sizes and formats. We consider instantiations over NIST safe elliptic curves over prime-order finite fields [Ran23] for security levels of 128, 192, and 256 bits, and assess the implications on signature sizes in compressed and uncompressed form. Lastly, we present the experimental performance of the proposed IBS (and variants) in comparison with the Galindo-Garcia scheme.

6.1 Theoretical evaluation

In Section 4, we describe different ECDSA-like identity-based schemes that support signature concatenation via the Hierarchical Public Key Delegation technique. Table 1 compares these different schemes in terms of complexity, considering the number of operations over the ring $\mathbb{Z}_p$, over the group $\mathbb{G}$, and hash computations. We use the following notation to represent the complexity of each algorithm. Sum, multiplication, and inversion over $\mathbb{Z}_p$ are represented by $\mathtt{S}$, $\mathtt{M}$, and $\mathtt{I}$, respectively. For point addition and multiplication by scalar we use $\mathtt{S}_{\mathbb{G}}$ and $\mathtt{M}_{\mathbb{G}}$. Lastly, for hash computations, we use $\mathtt{H}$. As shown in Table 1, our proposed scheme (and variants) displays a higher algorithmic complexity compared to Galindo and Garcia's scheme. This is expected, since ECDSA has more convoluted signing and verification processes than the Schnorr signature scheme underlying the Galindo-Garcia construction.

The proposed scheme $\mathrm{IBS}_{\mathrm{ecdsa}}$ and its variant $\mathrm{IBS}^{\mathtt{v}_1}_{\mathrm{ecdsa}}$ require the recovery of point R during the signing phase. This step is unavoidable because the point R is used as a generator when obtaining $A = aR$, where $a \xleftarrow{\$} \mathbb{Z}_p$, which is required for the computation of $t = f(A)$, part of the output signature σ. We are able to remove the inversion over $\mathbb{Z}_p$ and reduce the overall algorithmic complexity during the signing and verification phases in the $\mathrm{IBS}^{\mathtt{v}_2}_{\mathrm{ecdsa}}$ variant of our scheme. This improvement is obtained by making a slight modification to the user's secret key, writing it as $\mathtt{sk}_{\mathtt{id}} = (u_{\mathtt{id}}, R)$ and computing the value $t_{\mathtt{id}} = f(R)$ a posteriori, during the verification phase. This is intended to speed up computations and circumvent the need to recover R, which comes at the cost of essentially conducting an ECDSA verification on $\mathtt{sk}_{\mathtt{id}} = (u_{\mathtt{id}}, t_{\mathtt{id}})$ over identity $\mathtt{id}$ using the master public key.

Table 1: Algorithmic analysis of key derivation, sign, and verify algorithms comparing the variants of identity-based ECDSA and [GG09] scheme.

	KeyDeriv	Sign	Verify
[GG09]	$\mathtt{M}_{\mathbb{G}} + \mathtt{H} + \mathtt{S} + \mathtt{M}$	$\mathtt{M}_{\mathbb{G}} + \mathtt{H} + \mathtt{S} + \mathtt{M}$	$2\mathtt{S}_{\mathbb{G}} + 5\mathtt{M}_{\mathbb{G}} + 2\mathtt{H}$
$\mathrm{IBS}_{\mathrm{ecdsa}}$	$\mathtt{M}_{\mathbb{G}} + \mathtt{H} + \mathtt{S} + \mathtt{I} + 2\mathtt{M}$	$2\mathtt{M}_{\mathbb{G}} + 2\mathtt{H} + 2\mathtt{S} + 2\mathtt{I} + 4\mathtt{M}$	$5\mathtt{S}_{\mathbb{G}} + 2\mathtt{M}_{\mathbb{G}} + 2\mathtt{H} + \mathtt{I}$
$\mathrm{IBS}^{\mathtt{v}_1}_{\mathrm{ecdsa}}$	-	$2\mathtt{M}_{\mathbb{G}} + 2\mathtt{H} + 2\mathtt{S} + 2\mathtt{I} + 4\mathtt{M}$	$5\mathtt{S}_{\mathbb{G}} + 2\mathtt{M}_{\mathbb{G}} + 2\mathtt{H} + \mathtt{I}$
$\mathrm{IBS}^{\mathtt{v}_2}_{\mathrm{ecdsa}}$	-	$\mathtt{M}_{\mathbb{G}} + \mathtt{H} + \mathtt{S} + \mathtt{I} + 2\mathtt{M}$	$3\mathtt{S}_{\mathbb{G}} + 4\mathtt{M}_{\mathbb{G}} + 2\mathtt{H} + \mathtt{I}$

6.2 Concrete evaluation of signature size

Table 2 shows the signature formats of each IBS together with their respective sizes. We notice that the signatures obtained with all variants are slightly close to those in Galindo-Garcia's. For example, considering a concrete instantiation of the scheme for curve `secp256k1`, and points in compressed form, we obtain signatures with size 97 bytes instead of 98 bytes for the Galindo-Garcia scheme. This happens because each point comprises 32 bytes for the x–coordinate and 1 byte for the parity value associated with the point, which allows one to recover the y–coordinate by simple substitution in the elliptic curve equation. This one-byte difference is the same for other safe elliptic curves. Concretely, though, the encoding format used (e.g., [Por13, JL17, WNR20]) would result in signatures with varying sizes, but roughly the same size.

6.3 Performance: ECDSA vs. Schnorr signatures

In our experiments, we consider three elliptic curves over prime-order finite fields for detailed comparisons: `secp256k1`, `secp384r1`, and `secp521r1`, which provide security levels 128, 192, and 256, respectively. Our main implementations of Galindo-Garcia's and

Table 2: Comparison of signatures' format and size between all considered identity-based signature schemes.

	Signature format	Signature size
[GG09]	(A, b, R)	$\|\mathbb{Z}_p\| + 2\|\mathbb{G}^*\|$
$\text{IBS}_{\text{ecdsa}}$	(s, t, R)	$2\|\mathbb{Z}_p\| + \|\mathbb{G}^*\|$
$\text{IBS}^{\text{v}_1}_{\text{ecdsa}}$	(s, t, R)	$2\|\mathbb{Z}_p\| + \|\mathbb{G}^*\|$
$\text{IBS}^{\text{v}_2}_{\text{ecdsa}}$	(s, t, R)	$2\|\mathbb{Z}_p\| + \|\mathbb{G}^*\|$

our ECDSA-based identity-based signature variants are done using OpenSSL `v3.5.4` and the C programming language. While ECDSA is natively supported by OpenSSL's `evp` library, the Schnorr signature scheme had to be implemented from scratch, which was done using OpenSSL's `bignum` and `ec` libraries. The latter, albeit older and not necessarily as optimized as `evp` itself, was chosen because it allows for more direct access to specific values used during computation. In particular, and unlike `evp`, the `ec` library supports easy access to the s-part of the Schnorr signature, used as private signing key in the Galindo-Garcia IBS. Even though this difference in libraries is not expected to have much impact on our performance measurements, we nevertheless provide additional baseline comparisons using the `bitcoin-core` library [Bc25], which comes with highly optimized C-language implementations of ECDSA and Schnorr signatures over `sec256k1`.

We use the `gcc` compiler `v14.2.0` with the `-O3 -march=native` flag to run the benchmarks for every scheme implemented. Each test is performed a total of $10,000$ times, with a warm-up percentage of 10%. Our experiments are executed using an Intel(R) Core(TM) `i7-6500` CPU with a clock frequency of 2.50 GHz, and run on one thread.

Table 3 compares the performance between ECDSA and Schnorr using OpenSSL and the curve-specific implementation of the `bitcoin-core` library for curve `sec256k1`. Analogously, Table 4 extends the comparison for curves `secp384r1` and `secp521r1`, with both signing algorithms implemented with OpenSSL. We note that both tables display similar trends, with ECDSA performing faster at security levels 128 and 192, and slightly worse for $\lambda = 256$ bits, during verification. For all three security levels, the signing phase of the Schnorr signature scheme performs faster.

Table 3: Comparison between ECDSA and Schnorr signatures with OpenSSL (our implementation) and with the `bitcoin-core` library [Bc25] for curve `sec256k1`.

	ECDSA avg. performance in ms		Schnorr avg. performance in ms	
	OpenSSL	Baseline Result	OpenSSL	Baseline Result
KeyGen	0.59	0.29	0.56	0.29
Sign	0.59	0.39	0.56	0.27
Verify	0.53	0.58	1.23	0.62

Table 4: Comparative analysis of Schnorr and ECDSA implementations with OpenSSL at different bit-security values.

	$\lambda = 128$			$\lambda = 192$			$\lambda = 256$		
	KeyGen	Sign	Verify	KeyGen	Sign	Verify	KeyGen	Sign	Verify
Schnorr	0.557	0.558	1.125	0.224	0.261	0.725	0.226	0.248	0.664
ECDSA		0.593	0.529		0.316	0.696		0.398	0.745

6.4 Performance: ECDSA- vs. Schnorr-based IBS

We used OpenSSL to implement our ECDSA-based IBS variants and Galindo-Garcia's original Schnorr-based construction. For a fair comparison, we used the `ec` library to implement all schemes and their underlying operations, except for the Setup algorithm: since its construction is identical for all IBS under analysis, we prefer to report the performance numbers for our fastest implementations, obtained with the `evp` library. All tests were performed with the same setup described in Section 6.3, including compiler settings, number of executions, and underlying machine.

Table 5 shows the comparison between Galindo-Garcia's IBS[GG09] and the ECDSA variants described in this work. As expected, variants IBS and $\text{IBS}^{\text{v1}}_{\text{ecdsa}}$ have a higher computational complexity than Galindo-Garcia's scheme, thus increasing the gap during the signing phase compared to the reference IBS. The variant $\text{IBS}^{\text{v2}}_{\text{ecdsa}}$ reduces the time taken to sign, as it requires the computation of $t_{\texttt{id}} = f(R)$ only during verification, without needing to run ECDSA's verification algorithm on the user's secret key $\texttt{sk}_{\texttt{id}}$, the user's identity, and the master public key to recover point R. Verification of all three proposed IBS schemes is slower due to the convoluted verification process of ECDSA. Still, our results show that the numbers obtained remain in the same order of magnitude: a few milliseconds for verification, and hundreds of milliseconds for the key derivation and signing algorithms. Therefore, in scenarios where the ECDSA signature scheme is preferred due to its widespread support, the variants described herein represent viable approaches for implementing an efficient IBS scheme.

Table 5: Average time performance (in ms) for each IBS scheme depending on the security level λ.

	$\lambda = 128$			$\lambda = 192$			$\lambda = 256$		
	KeyDeriv	Sign	Verify	KeyDeriv	Sign	Verify	KeyDeriv	Sign	Verify
[GG09]	0.521	0.519	1.561	0.243	0.244	1.145	0.234	0.234	1.047
IBS	0.555	2.158	2.671	0.256	1.631	2.057	0.320	1.536	1.906
$\text{IBS}^{\text{v1}}_{\text{ecdsa}}$	0.574	2.231	2.761	0.320	1.682	2.123	0.320	1.568	1.947
$\text{IBS}^{\text{v2}}_{\text{ecdsa}}$	0.579	0.577	2.787	0.294	0.529	2.108	0.295	0.487	1.924

7 Conclusion

In this paper, we present the first ECDSA-like identity-based signature scheme constructed from the hierarchical delegation of public key technique. The schemes described allow the concatenation of ECDSA signatures by leveraging the cyclic property of the group to define a second generator during the signing phase. We discuss some possible variants for the construction, which can deliver performance improvements (especially during the signing phase) by modifying the structure of the user's secret key.

In terms of security analysis, we prove that the proposed construction is secure by combining the existential unforgeability under chosen-message attack in the identity-based setting – EUF-ID-CMA game – and the bijective random oracle model. For our analysis, we adapt the GenDSA framework to the identity-based setting. First, we reduce the security of our scheme to the UF-ID-KOA security notion. Then, we show that the latter allows us to reduce the security of the construction to the hardness of forging an ECDSA signature in the EUF-CMA game. We implement some variants of our proposal and compare them to Galindo-Garcia's IBS [GG09], an analogous construction based on Schnorr's signature scheme. Our results show that our scheme performs slightly worse than Galindo-Garcia's, while also slightly reducing the size of the obtained signatures.

All in all, the proposed construction represents an interesting approach in scenarios that may benefit from ECDSA's widespread support, including contemporary cloud-oriented authentication systems. It also opens up opportunities for advanced applications in secure credential management, such as hierarchical token delegation.

Aknowledgments

We thank the anonymous reviewers of IACR Communications in Cryptology for their shepherding that guided us to an improvement in the quality of this work. Also, we thank Thales B. Paiva for his important suggestions and discussions. This study was financed in part by Hewlett Packard Enterprise (HPE), by the Ripple's University Blockchain Research Initiative, and by the following Brazilian research agencies: FAPESP (grant 2020/09850-0), CNPq (grant 307732/2023-1), and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brazil (CAPES) – (Finance Code 001).

References

[AGM18] Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 643–673. Springer, Cham, August 2018. doi:10.1007/978-3-319-96878-0_22.

[ANT+20] Diego F. Aranha, Felipe Rodrigues Novaes, Akira Takahashi, Mehdi Tibouchi, and Yuval Yarom. LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 225–242, Virtual Event, USA, November 9–13, 2020. ACM Press. doi:10.1145/3372297.3417268.

[Bc25] Bitcoin-core. secp256k1: Optimized C library for EC operations on curve secp256k1. https://github.com/bitcoin-core/secp256k1, 2025.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 326–349, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery. doi:10.1145/2090236.2090263.

[BH19] Joachim Breitner and Nadia Heninger. Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security*, pages 3–20, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-32101-7_1.

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 491–506, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. doi:10.1007/3-540-39200-9_31.

[BLMQ05] Paulo Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT*

2005, pages 515–532, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11593447_28.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006. doi:10.1145/1180405.1180453.

[BPVY00] Ernest Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung. Design validations for discrete logarithm based signature schemes. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, pages 276–292, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. doi:10.1007/978-3-540-46588-1_19.

[BPW12] Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. *Journal of Cryptology*, 25(1):57–115, 01 2012. doi:10.1007/s00145-010-9082-x.

[BR04] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Paper 2004/331, 2004. URL: https://eprint.iacr.org/2004/331.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 409–426, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. doi:10.1007/11761679_25.

[Bro02] Daniel R. L. Brown. Generic groups, collision resistance, and ECDSA. Cryptology ePrint Archive, Paper 2002/026, 2002. URL: https://eprint.iacr.org/2002/026.

[Bro05] Daniel R. Brown. Generic groups, collision resistance, and ECDSA. *Design, Codes and Cryptography*, 35(1):119–152, 04 2005. doi:10.1007/s10623-003-6154-z.

[BSCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 103–128, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-17653-2_4.

[BZ23] Paulo Barreto and Gustavo Zanon. Blind signatures from zero-knowledge arguments. Cryptology ePrint Archive, Paper 2023/067, 2023. URL: https://eprint.iacr.org/2023/067.

[CD25] Geoffroy Couprie and Clément Delafargue. *Biscuit Documentation*, 07 2025. URL: https://doc.biscuitsec.org/.

[CFOS25] Gavin Cho, Georg Fuchsbauer, Adam O'Neill, and Marek Sefranek. Schnorr signatures are tightly secure in the ROM under a non-interactive assumption. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025*, pages 223–255, Cham, 2025. Springer Nature Switzerland. doi:10.1007/978-3-032-01887-8_8.

[CKK13] Sanjit Chatterjee, Chethan Kamath, and Vikas Kumar. Galindo-Garcia identity-based signature revisited. In Taekyoung Kwon, Mun-Kyu Lee, and

Daesung Kwon, editors, *ICISC 12: 15th International Conference on Information Security and Cryptology*, volume 7839 of *Lecture Notes in Computer Science*, pages 456–471, Seoul, Korea, November 28–30, 2013. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-37682-5_32.

[CKM23] Elizabeth Crites, Chelsea Komlo, and Mary Maller. Fully adaptive Schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 678–709, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-38557-5_22.

[CLR24] Sofia Celi, Shai Levin, and Joe Rowell. CDLS: Proving knowledge of committed discrete logarithms with soundness. In Serge Vaudenay and Christophe Petit, editors, *Progress in Cryptology - AFRICACRYPT 2024*, pages 69–93, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-64381-1_4.

[CMRR23] Lily Chen, Dustin Moody, Andrew Regenscheid, and Angela Robinson. Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters. Technical Report NIST Special Publication (SP) 800-186, National Institute of Standards and Technology, Gaithersburg, MD, 2023. doi:10.6028/NIST.SP.800-186.

[DW14] Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and MtGox. In Mirosław Kutyłowski and Jaideep Vaidya, editors, *Computer Security - ESORICS 2014*, pages 313–326, Cham, 2014. Springer International Publishing. doi:10.1007/978-3-319-11212-1_18.

[Eth24] Ethereum. Ethereum Accounts, September 2024. URL: https://ethereum.org/en/developers/docs/accounts/.

[FHLM22] Armando Faz-Hernández, Watson Ladd, and Deepak Maram. ZKAttest: Ring and group signatures for existing ECDSA keys. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography*, pages 68–83, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-030-99277-4_4.

[FKP16] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the provable security of (EC)DSA signatures. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1651–1662. ACM Press, October 2016. doi:10.1145/2976749.2978413.

[FKP17] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the one-per-message unforgeability of (EC)DSA and its variants. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 519–534, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-70503-3_17.

[GG09] David Galindo and Flavio D. Garcia. A Schnorr-like lightweight identity-based signature scheme. In Bart Preneel, editor, *Progress in Cryptology – AFRICACRYPT 2009*, pages 135–148, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-02384-2_9.

[Gir91] Marc Girault. Self-certified public keys. In Donald W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, pages 490–497, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. doi:10.1007/3-540-46416-6_42.

[GKL+21] Robert Granger, Thorsten Kleinjung, Arjen Lenstra, Benjamin Wesolowski, and Jens Zumbrägel. Computation of a 30750-bit binary field discrete logarithm. *Mathematics of computation*, 90(332):2997–3022, 2021. doi:10.1090/mcom/3669.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. doi:10.1016/0022-0000(84)90070-9.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, pages 548–566, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. doi:10.1007/3-540-36178-2_34.

[GS22] Jens Groth and Victor Shoup. On the security of ECDSA with additive key derivation and presignatures. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, pages 365–396, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-06944-4_13.

[GWHH25] Yiming Gao, Jinghui Wang, Honggang Hu, and Binang He. Attacking ECDSA with nonce leakage by lattice sieving: Bridging the gap with Fourier Analysis-based attacks. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024*, pages 3–34, Singapore, 2025. Springer Nature Singapore. doi:10.1007/978-981-96-0944-4_1.

[Her06] Javier Herranz. Deterministic identity-based signatures for partial aggregation. *The Computer Journal*, 49(3):322–330, 2006. doi:10.1093/comjnl/bxh153.

[HK23] Dominik Hartmann and Eike Kiltz. Limits in the provable security of ECDSA signatures. In Guy Rothblum and Hoeteck Wee, editors, *Theory of Cryptography*, pages 279–309, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-48624-1_11.

[JBS15] Michael Jones, John Bradley, and Nat Sakimura. JSON Web Token (JWT). RFC 7519, May 2015. doi:10.17487/RFC7519.

[JL17] S. Josefsson and I. Liusvaara. Edwards-Curve Digital Signature Algorithm (EdDSA). RFC 8032, January 2017. URL: https://www.rfc-editor.org/info/rfc8032/, doi:10.17487/RFC8032.

[Lin21] Yehuda Lindell. Fast secure two-party ECDSA signing. *Journal of Cryptology*, 34(4):44, 9 2021. doi:10.1007/s00145-021-09409-9.

[LV01] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, January 2001. doi:10.1007/s00145-001-0009-4.

[MPD25] Jules Maire and Alan Pulval-Dady. Blind ECDSA from the ECDSA assumption. *IACR Communications in Cryptology*, 2(3), 2025. doi:10.62056/ahjbhee-3.

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, Sep 2019. doi:10.1007/s10623-019-00608-x.

[MSM+16] Hiraku Morita, Jacob C. N. Schuldt, Takahiro Matsuda, Goichiro Hanaoka, and Tetsu Iwata. On the security of the Schnorr signature scheme and DSA against related-key attacks. In Soonhak Kwon and Aaram Yun, editors, *Information Security and Cryptology - ICISC 2015*, pages 20–35, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-30840-1_2.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *bitcoin.org*, 2008. URL: http://bitcoin.org/bitcoin.pdf.

[Nat23] National Institute of Standards and Technology. Digital Signature Standard (DSS). Technical Report 186-5, National Institute of Standards and Technology, Feb 2023. Approved February 2023. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 02 1994. doi:10.1007/BF02113297.

[Por13] T. Pornin. Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA). RFC 6979, August 2013. URL: https://www.rfc-editor.org/info/rfc6979/, doi:10.17487/RFC6979.

[QCY21] Xianrui Qin, Cailing Cai, and Tsz Hon Yuen. One-more unforgeability of blind ECDSA. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *ESORICS 2021: 26th European Symposium on Research in Computer Security, Part II*, volume 12973 of *Lecture Notes in Computer Science*, pages 313–331, Darmstadt, Germany, October 4–8, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-88428-4_16.

[Ran23] K. Randall. NIST Special Publication (SP) 800-186, Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters. https://csrc.nist.gov/pubs/sp/800/186/final, 4 2023. [Accessed 13-08-2025].

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018. doi:10.17487/RFC8446.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978. doi:10.1145/359340.359342.

[Rya18] Keegan Ryan. Return of the hidden number problem.: A widespread and novel key extraction attack on ecdsa and dsa. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):146–168, Nov. 2018. doi:10.13154/tches.v2019.i1.146-168.

[Sch90] C. P. Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York. doi:10.1007/0-387-34805-0_22.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 47–53, Berlin, Heidelberg, 1985. Springer. doi:10.1007/3-540-39568-7_5.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 256–266, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. doi:10.1007/3-540-69053-0_18.

[SPI25] SPIFFE. Secure Production Identity Framework for Everyone. (online), 2025. URL: https://spiffe.io/.

[SPMLS02] Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 93–110, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. doi:10.1007/3-540-45708-9_7.

[WNR20] P. Wuille, J. Nick, and T. Ruffing. Schnorr signatures for secp256k1. Bitcoin improvement proposals, 2020. URL: https://bips.xyz/340.

[WOS+25] Anna P. Y. Woo, Alex Ozdemir, Chad Sharp, Thomas Pornin, and Paul Grubbs. Efficient proofs of possession for legacy signatures. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 3291–3308, San Francisco, CA, USA, May 12–15, 2025. IEEE Computer Society Press. doi:10.1109/SP61157.2025.00080.

[XDWL23] Luyao Xu, Zhengyi Dai, Baofeng Wu, and Dongdai Lin. Improved attacks on (EC)DSA with nonce leakage by lattice sieving with predicate. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023, Issue 2:568–586, 2023. doi:10.46586/tches.v2023.i2.568-586.