

Hardening Signature Schemes via Derive-then-Derandomize: Stronger Security Proofs for EdDSA

Mihir Bellare¹, Hannah Davis¹, and Zijing Di²

¹ University of California San Diego
mihir@eng.ucsd.edu, h3davis@eng.ucsd.edu

² Stanford University
zidi@stanford.edu

Abstract. We consider a transform, called Derive-then-Derandomize, that hardens a given signature scheme against randomness failure and implementation error. We prove that it works. We then give a general lemma showing indistinguishability of a class of constructions that apply a shrinking output transform to an MD-style hash function. Armed with these tools, we give new proofs for the widely standardized and used EdDSA signature scheme, improving prior work in two ways: (1) we give proofs for the case that the hash function is an MD-style one, reflecting the use of SHA512 in the NIST standard, and (2) we improve the tightness of the reduction so that one has guarantees for group sizes in actual use.

1 Introduction

In designing schemes, and proving them secure, theoreticians implicitly assume certain things, such as on-demand fresh randomness and correct implementation. In practice, these assumptions can fail. Weaknesses in system random-number generators are common and have catastrophic consequences. (An example relevant to this paper is the well-known key-recovery attack on Schnorr signatures when signing reuses randomness. Another striking example are Ps and Qs attacks [25, 29].) Meanwhile, implementation errors can be exploited, as shown by Bleichenbacher’s attack on RSA signatures [15].

In light of this, practitioners may try to “harden” theoretical schemes before standardization and usage. A prominent and highly successful instance is EdDSA, a hardening of the Schnorr signature scheme proposed by Bernstein, Duif, Lange, Schwabe, and Yang (BDLSY) [14]. It incorporates explicit, simple key-derivation, makes signing deterministic, adds protection against sidechannel attacks via “clamping,” and for simplicity confines itself to a single hash function, namely SHA512. The scheme is widely standardized [34, 27] and used [26].

There is however a subtle danger here, namely that the hardening attempt introduces new vulnerabilities. In other words, hardening needs to be done right; if not, it may even “soften” the scheme! Thus it is crucial that the hardened scheme

be vetted via a proof of security. This is of particular importance for **EdDSA** given its widespread deployment. In that regard, Brendel, Cremers, Jackson and Zhao (BCJZ) [16] showed that **EdDSA** is secure if the Discrete-Log (DL) problem is hard and the hash function is modeled as a random oracle. This is significant as a first step but has at least two important limitations: (1) Due to the extension attack, a random oracle is not an appropriate model for the **SHA512** hash function **EdDSA** actually uses, and (2) the reduction is so loose that there is no security guarantee for group sizes in use today.

Extrapolating **EdDSA**, the first part of this paper defines a general hardening transform on signature schemes called Derive-then-Derandomize (**DtD**), and proves its soundness. Next we prove the indistinguishability of a general class of constructions, that we call shrink-MD; it includes the well-studied chop-MD construction [19] and also the modulo-a-prime construction arising in **EdDSA**. Armed with these results, the second part of the paper returns to give new proofs for **EdDSA** that in particular fill the above gaps. We begin with some background.

RESPECTING HASH STRUCTURE IN PROOFS. Recall that the MD-transform [31, 20] defines a hash function $H = \mathbf{MD}[h] : \{0, 1\}^* \rightarrow \{0, 1\}^{2k}$ by iterating an underlying compression function $h : \{0, 1\}^{b+2k} \rightarrow \{0, 1\}^{2k}$. (See Section 2 for details.) **SHA256** and **SHA512** are obtained in this way, with (b, k) being $(512, 128)$ and $(1024, 256)$, respectively. This structure gives rise to attacks, of which the most well known is the extension attack. The latter allows an attacker given $t \leftarrow \mathbf{MD}[h](e_2 \| M)$, where e_2 is a secret unknown to the attacker and $M \in \{0, 1\}^*$ is public, to compute $t' = \mathbf{MD}[h](e_2 \| M')$, for some $M' \in \{0, 1\}^*$ of its choice. This has been exploited to violate the UF-security of the so-called prefix message authentication code $\text{pMAC}_{e_2}(M) = H(e_2 \| M)$ when H is an MD-hash function; **HMAC** [4] was designed to overcome this.

A proof of security of a scheme (such as **EdDSA**) that uses a hash function H will often model H as a random oracle [10], in what we'll call the (H, H) -model: scheme algorithms, and the adversary, both have oracle access to the same random H . However the presence of the above-discussed structure in “real” hash functions led Dodis, Ristenpart and Shrimpton (DRS) [21] to argue that the “right” model in which to prove security of a scheme that uses $H = \mathbf{MD}[h]$ is to model the compression function h —rather than the hash function $H = \mathbf{MD}[h]$ —as a random oracle. We'll call this the $(\mathbf{MD}[h], h)$ -model: the adversary has oracle access to a random h , with scheme algorithms having access to $\mathbf{MD}[h]$. There is now widespread agreement with the DRS thesis that proofs of security of MD-hash-using schemes should use the $(\mathbf{MD}[h], h)$ model.

Giving from-scratch proofs in the $(\mathbf{MD}[h], h)$ model is, however, difficult. Maurer, Renner and Holenstein (MRH) [30] show that if a construction \mathbf{F} is indistinguishable (abbreviated *indiff*) and a scheme is secure in the (H, H) model, then it remains secure in the $(\mathbf{F}[h], h)$ model. (This requires the game defining security of the scheme to be single-stage [38], which is true for the relevant ones here.) Unfortunately, $\mathbf{F} = \mathbf{MD}$ is provably *not* *indiff* [19], due exactly to the extension attack. So the MRH result does not help with **MD**. This led to a

search for indiff variants. DRS [21] and YMO [42] (independently) offer public-indiff and show that it suffices to prove security, in the $(\mathbf{MD}[h], h)$ model, of schemes that use \mathbf{MD} in some restricted way. However, EdDSA does not obey these restrictions. Thus, other means are needed.

THE EdDSA SCHEME. The Edwards curve Digital Signature Algorithm (EdDSA) is a Schnorr-based signature scheme introduced by Bernstein, Duif, Lange, Schwabe and Yang [14]. Ed25519, which uses the Curve25519 Edwards curve and SHA512 as the hash function, is its most popular instance. The scheme is standardized by NIST [34] and the IETF [27]. It is used in TLS 1.3, OpenSSH, OpenSSL, Tor, GnuPG, Signal and WhatsApp. It is also the preferred signature scheme of the Corda, Tezos, Stellar and Libra blockchain systems. Overall, IANIX [26] reports over 200 uses of Ed25519. Proving security of this scheme is accordingly of high importance.

Figure 4 shows EdDSA on the right, and, on the left, the classic Schnorr scheme [40] on which EdDSA is based. The schemes are over a cyclic, additively-written group \mathbb{G} of prime order p with generator B . The public verification key is A . The Schnorr hash function has range $\mathbb{Z}_p = \{0, \dots, p-1\}$, while, for EdDSA, function H_1 has range $\{0, 1\}^{2k}$ where k , the bit-length of p , is 256 for Ed25519. Functions H_2, H_3 have range \mathbb{Z}_p .

EdDSA differs from Schnorr in significant ways. While the Schnorr secret key s is in \mathbb{Z}_p , the EdDSA secret key sk is a k -bit string. This is hashed and the $2k$ -bit result is split into k -bit halves $e_1 || e_2$. A Schnorr secret-key s is derived by applying to e_1 a clamping function CF that zeroes out the three least significant bits of e_1 . (Note: This means s is *not* uniformly distributed over \mathbb{Z}_p .) Clamping increases resistance to side-channel attacks [14]. Signing is made deterministic by a standard de-randomization technique [23, 33, 9, 12], namely obtaining the Schnorr randomness r by hashing the message M with a secret-key dependent string e_2 . We note that all of H_1, H_2, H_3 are instantiated via the same hash function, namely SHA512.

PRIOR WORK AND OUR QUESTIONS. Recall that the security goal for a signature scheme is UF (UnForgeability under Chosen-Message Attack) [24]. Schnorr is well studied, and proven UF under DL (Discrete Log in \mathbb{G}) when H is a random oracle [37, 1]. The provable security of EdDSA, however, received surprisingly little attention until the work of Brendel, Cremers, Jackson and Zhao (BCJZ) [16]. They take the path also used for Schnorr and other identification-based signature schemes [37, 1], seeing EdDSA as the result of the Fiat-Shamir transform on an underlying identification scheme EdID that they define, proving security of the latter under DL, and concluding UF of EdDSA under DL when H is a random oracle. This is an important step forward, but the BCJZ proof [16] remains in the (H, H) model. We ask and address the following two questions.

1. Can we prove security in the $(\mathbf{MD}[h], h)$ model? The NIST standard [34] mandates that Ed25519 uses SHA512, which is an MD-hash function. Accordingly, as explained above, the BCJZ proof [16], being in the (H, H) model, does not guarantee security; to do the latter, we need a proof in the $(\mathbf{MD}[h], h)$ model.

The gap is more than cosmetic. As we saw above with the example of the prefix MAC, a scheme could be secure in the (H, H) model, yet totally insecure in the more realistic $(MD[h], h)$ model, and thus also in practice. And EdDSA skirts close to the edge: line 14 is using the prefix-MAC that the extension attack breaks, and overlaps in inputs across the three uses of H could lead to failures. Intuitively what prevents attacks is that the MAC outputs are taken modulo p , and inputs to H in two of the three uses involve secrets. Thus, we'd expect that the scheme is indeed secure in the $(MD[h], h)$ model.

Proving this, however, is another matter. We already know that MD is not indiff. It is public indiff [21, 42], but this will not suffice for EdDSA because H_1, H_2 are being called on secrets. We ask, first, can EdDSA be proved secure in the $(MD[h], h)$ model, and second, can this be done in some modular way, rather than from scratch?

2. Can we improve reduction tightness? The reduction of BCJZ [16] is so loose that, in the 256-bit curve over which Ed25519 is implemented, it guarantees little security. Let's elaborate. Given an adversary A_{UF} violating the UF-security of EdDSA with probability ϵ_{UF} , the reduction builds an adversary A_{DL} breaking DL with probability $\epsilon_{DL} = \epsilon_{UF}^2/q_h$ where q_h is the number of H -queries of A_{UF} and the two adversaries have about the same running time t . (The square arises from the use of rewinding, analyzed via the Reset Lemma of [8].) In an order p elliptic curve group, $\epsilon_{DL} \approx t^2/p$ so we get $\epsilon_{UF} = t \cdot \sqrt{q_h/p}$. Ed25519 has $p \approx 2^{256}$. Say $t = q_h = 2^{70}$, which (as shown by BitCoin mining capability) is not far from attacker reach. Then $\epsilon_{DL} = 2^{-116}$ is small but $\epsilon_{UF} = 2^{70} \cdot 2^{-(256-70)/2} = 2^{-23}$ is in comparison quite high.

Now, one might say that one would not expect better because the same reduction loss is present for Schnorr. The classical reductions for Schnorr [37, 1] did indeed display the above loss, but that has changed: recent advances for Schnorr include a tighter reduction from DL [39], an almost-tight reduction from the MBDL problem [5] and a tight reduction from DL in the Algebraic Group Model [22]. We'd like to put EdDSA on par with the state of the art for Schnorr. We ask, first, is this possible, and second, is there a modular way to do it that leverages, rather than repeats, the (many, complex) just-cited proofs for Schnorr?

CONTRIBUTIONS FOR EdDSA. We simultaneously simplify and strengthen the security proofs for EdDSA as follows.

1. Reduction from Schnorr. Rather than, as in prior work, give a reduction from DL or some other algebraic problem, we give a simple, direct reduction from Schnorr itself. That is, we show that if the Schnorr signature scheme is UF-secure, then so is EdDSA. Furthermore, the reduction is *tight* up to a constant factor. This allows us to leverage prior work [39, 5, 22] to obtain tight proofs for EdDSA under various algebraic assumptions and justify security for group sizes in actual use. But there are two further dividends. First, Schnorr [40] is over 30 years old and has withstood the tests of time and cryptanalysis, so our proof that EdDSA is just as secure as Schnorr allows the former to inherit, and benefit from, this confidence. Second, our result formalizes and proves what was the intuition and

belief in the first place [14], namely that, despite the algorithmic differences, EdDSA is a sound hardening of Schnorr.

2. Accurate modeling of the hash function. As noted above, BCJZ [16] assume the hash function H is a random oracle, but this, due to the extension attack, is not an accurate model for the MD-hash function **SHA512** used by EdDSA. We fill this gap by instead proving security in the $(\mathbf{MD}[h], h)$ model, where $H = \mathbf{MD}[h]$ is derived via the MD-transform [31, 20] and the compression function h is a random oracle.

APPROACH AND BROADER CONTRIBUTIONS. The above-mentioned results on EdDSA are obtained as a consequence of more general ones.

3. The DtD transform and its soundness. We extend the hardening technique used in EdDSA to define a general transform that we call Derive-then-Derandomize (**DtD**). It takes an *arbitrary* signature scheme \mathbf{DS} , and with the aid of a PRG H_1 and a PRF H_2 , constructs a hardened signature scheme $\overline{\mathbf{DS}}$. We provide (Theorem 1) a strong and general validation of **DtD**, showing that $\overline{\mathbf{DS}}$ is UF-secure assuming \mathbf{DS} is UF-secure. Moreover *the reduction is tight* and the proof is simple. This shows that the EdDSA hardening method is generically sound.

4. Indifferentiability of Shrink-MD. It is well-known that **MD** is not indifferentiable [30] from a random oracle, but that the **Chop-MD** [19], which truncates the output of an **MD** hash by some number of bits, is indifferentiable. Unfortunately, we identified gaps in two prominent proofs of indifferentiability of **Chop-MD** [19, 32]. EdDSA uses a similar construction that reduces the **MD** hash output modulo a prime p sufficiently smaller than the size of the range of **MD**, due to which we refer to this construction as **Mod-MD**. The **Mod-MD** construction has not been proven indifferentiable. We simultaneously give new proofs of indifferentiability for **Chop-MD** and **Mod-MD** as part of a more general class of constructions that we call **Shrink-MD** functors. These are constructions of the form $\text{Out}(\mathbf{MD})$ where Out is some output-processing function, and we prove indifferentiability under certain “shrinking” conditions on Out .

5. Application to EdDSA. EdDSA is obtained as the result $\overline{\mathbf{DS}}$ of the **DtD** transform applied to the $\mathbf{DS} = \text{Schnorr}$ signature scheme, and with the PRG and PRF defined via **MD**, specifically $H_1(sk) = \mathbf{MD}[h](sk)$ and $H_2(e_2, M) = \mathbf{MD}[h](e_2 \| M) \bmod p$ where p is the prime order of the underlying group. Additionally, the hash function used in Schnorr is also $H_3(X) = \mathbf{MD}[h](X) \bmod p$. Due to Theorem 1 validating **DtD**, we are left to show the PRG security of H_1 , the PRF security of H_2 and the UF-security of Schnorr, all with h modeled as a random oracle. We do the first directly. We obtain the second as a consequence of the indifferentiability of **Mod-MD**. (In principle it follows from the PRF security of AMAC [3], but we found it difficult to extract precise bounds via this route.) For the third, we again exploit indifferentiability of **Mod-MD**, together with a technique from BCJZ [16] to handle clamping, to reduce to the UF security of regular Schnorr, where the hash function is modeled as a random

oracle. Putting all this carefully together yields our above-mentioned results for EdDSA. We note that one delicate and important point is that the idealized compression function h is *the same* across H_1, H_2 and H_3 , meaning these are not independent. This is handled through the building blocks in Theorem 1 being functors [7] rather than functions.

DISCUSSION AND RELATED WORK. Both BCJZ [16] and CGN [17] note that there are a few versions of EdDSA out there, the differences being in their verification algorithms. What Figure 4 shows is the most basic version of the scheme, but we will be able to cover the variants too, in a modular way, by reducing from Schnorr with the same verification algorithm.

BBT [3] define the function $\text{AMAC}[h]$ to take a key e_2 and message M , and return $\text{MD}[h](e_2 \| M) \bmod p$. This is the H_2 in EdDSA. We could exploit their results to conclude PRF security of H_2 , but it requires putting together many different pieces from their work, and it is easier and more direct to establish PRF security of H_2 by using our lemma on the indistinguishability of **Mod-MD**.

In the Generic Group Model (GGM) [41], it is possible to prove UF-security of Schnorr under standard (rather than random oracle) model assumptions on the hash functions [35, 18]. But use of the GGM means the result applies to a limited class of adversaries. Our results, following the classical proofs for identification-based signatures [37, 36, 1, 28], instead use the standard model for the group, while modeling the hash function (in our case, the compression function) as a random oracle.

In an earlier version of this paper, our proofs had relied on a variant of indistinguishability that we had introduced. At the suggestion of a Crypto 2022 reviewer, this has been dropped in favor of a direct proof based on PRG and PRF assumptions on H_1, H_2 . We thank the (anonymous) reviewer for this suggestion.

Theorem 1 is in the standard model if the PRG, PRF and starting signature scheme DS are standard-model, hence can be viewed as a standard-model justification of the hardening template underlying EdDSA. However, when we want to justify EdDSA itself, we need to consider the specific, **MD**-based instantiations of the PRG, PRF and Schnorr hash function, and for these we use the model where the compression function is ideal.

Several works study de-randomization of signing by deriving the coins via a PRF applied to the message, considering different ways to key the PRF [23, 33, 9, 12]. We use their techniques in the proof of Theorem 1.

One might ask how to view the UF-security of Schnorr signatures as an assumption. What is relevant is not its form (it is interactive) but that (1) it can be seen as a hub from where one can bridge to other assumptions that imply it, such as DL (non-tightly) [37, 1] or MBDL (tightly) [5], and (2) it is validated by decades of cryptanalysis.

Our results have been stated for UF but extend to SUF (Strong unforgeability), meaning our proofs also show SUF-security of EdDSA in the $(\text{MD}[h], h)$ model assuming SUF security of Schnorr, with a tight (up to the usual constant factor) reduction.

EdDSA could be used with other hash functions such as **SHAKE**. The extension attack does not apply to the latter, so the proof of BCJZ [16] applies, but gives a loose reduction from DL; our results still add something, namely a tight reduction from Schnorr and thus improved tightness in several ways as discussed above.

2 Preliminaries

NOTATION. If n is a positive integer, then \mathbb{Z}_n denotes the set $\{0, \dots, n-1\}$ and $[n]$ or $[1..n]$ denote the set $\{1, \dots, n\}$. If \mathbf{x} is a vector then $|\mathbf{x}|$ is its length (the number of its coordinates), $\mathbf{x}[i]$ is its i -th coordinate and $[\mathbf{x}] = \{ \mathbf{x}[i] : 1 \leq i \leq |\mathbf{x}| \}$ is the set of all its coordinates. A string is identified with a vector over $\{0, 1\}$, so that if x is a string then $x[i]$ is its i -th bit and $|x|$ is its length. We denote $x[i..j]$ the i -th bit to the j -th bit of string x . By ε we denote the empty vector or string. The size of a set S is denoted $|S|$. For sets D, R let $\text{AF}(D, R)$ denote the set of all functions $f : D \rightarrow R$. If $f : D \rightarrow R$ is a function then $\text{Im}(f) = \{ f(x) : x \in D \} \subseteq R$ is its image. We say that f is *regular* if every $y \in \text{Im}(f)$ has the same number of pre-images under f . By $\{0, 1\}^{\leq L}$ we denote the set of all strings of length at most L . For any variables a and b , the expression $[[a = b]]$ denotes the Boolean value **true** when a and b contain the same value and **false** otherwise.

Let S be a finite set. We let $x \leftarrow^* S$ denote sampling an element uniformly at random from S and assigning it to x . We let $y \leftarrow A[\text{O}_1, \dots](x_1, \dots; r)$ denote executing algorithm A on inputs x_1, \dots and coins r with access to oracles O_1, \dots and letting y be the result. We let $y \leftarrow^* A[\text{O}_1, \dots](x_1, \dots)$ be the resulting of picking r at random and letting $y \leftarrow A[\text{O}_1, \dots](x_1, \dots; r)$ be the equivalent. We let $\text{OUT}(A[\text{O}_1, \dots](x_1, \dots))$ denote the set of all possible outputs of A when invoked with inputs x_1, \dots and oracles O_1, \dots . Algorithms are randomized unless otherwise indicated. Running time is worst case.

GAMES. We use the code-based game playing framework of [11]. (See Fig. 1 for an example.) Games have procedures, also called oracles. Among the oracles are **INIT** and a **FIN**. In executing an adversary \mathcal{A} with a game G , the adversary may query the oracles at will. We require that the adversary's first oracle query be to **INIT** and its last to **FIN** and it query these oracles at most once. The value return by the **FIN** procedure is taken as the game output. By $G(\mathcal{A}) \Rightarrow y$ we denote the event that the execution of game G with adversary \mathcal{A} results in output y . We write $\Pr[G(\mathcal{A})]$ as shorthand for $\Pr[G(\mathcal{A}) \Rightarrow \text{true}]$, the probability that the game returns **true**.

In writing game or adversary pseudocode, it is assumed that Boolean variables are initialized to **false**, integer variables are initialized to 0 and set-valued variables are initialized to the empty set \emptyset .

We adopt the convention that the running time of an adversary is the time for the execution of the game with the adversary, so that the time for oracles to respond to queries is included, and similarly for the number of queries to an oracle. In particular, the number of queries to a random oracle **FO** includes those made by scheme algorithms executed by game procedures. By $Q_{\mathcal{A}}^{\text{O}}$ we denote the

number of queries made by \mathcal{A} and the game to oracle O in the execution. With q_A^O we count only queries made directly by \mathcal{A} to O , not by other game oracles or scheme algorithms. These counts are all worst case.

GROUPS. Throughout the paper, we fix integers k and b , an odd prime p , and a positive integer f such that $2^f < p$. We then fix two groups: G , a group of order $p \cdot 2^f$ whose elements are k -bit strings, and its cyclic subgroup G_p of order p . We prove in our full version [6] that this subgroup is unique, and that it has an efficient membership test. We also assume an efficient membership test for G . We will use additive notation for the group operation, and we let 0_G denote the identity element of G . We let $G_p^* = G \setminus \{0_G\}$ denote the set of non-identity elements of G_p , which is its set of generators. We fix a distinguished generator $B \in G_p^*$. Then for any $X \in G^*$, the discrete logarithm base B of X is denoted $DL_{G,B}(X)$, and it is in the set $\mathbb{Z}_{|G|}$. The instantiation of G used in Ed25519 is described in our full version [6].

3 Functor framework

Our treatment relies on the notion of functors [7], which are functions that access an idealized primitive. We give relevant definitions, starting with signature schemes whose security is measured relative to a functor. Then we extend the notions of PRGs and PRFs to functors.

FUNCTION SPACES. In using the random oracle model [10], works in the literature sometimes omit to say what exactly are the domain and range of the underlying functions, and, when multiple functions are present, whether or not they are independent. (Yet, implicitly their proofs rely on certain choices.) For greater precision, we use the language of function spaces of [7], which we now recall.

A *function space* O is a set of tuples $H = (H_1, \dots, H_n)$ of functions. The integer n is called the *arity* of the function space, and can be recovered as $O.\text{arity}$. We view H as taking an input X that it parses as (i, x) to return $H_i(x)$.

FUNCTORS. Following [7], we use the term functor for a transform that constructs one function from another. A functor $F : SS \rightarrow ES$ takes as oracle a function h from a starting function space SS and returns a function $F[h]$ in the ending function space ES . (The term is inspired by category theory, where a functor maps from one category into another. In our case, the categories are function spaces.) If ES has arity n , then we also refer to n as the arity of F , and write F_i for the functor which returns the i -th component of F . That is, $F_i[h]$ lets $H \leftarrow F[h]$ and returns H_i .

MD FUNCTOR. We are interested in the Merkle-Damgård [31, 20] transform. This transform constructs a hash function with domain $\{0, 1\}^*$ from a compression function $h : \{0, 1\}^{b+2k} \rightarrow \{0, 1\}^{2k}$ for some integers b and k . The compression function takes a $2k$ -bit chaining variable y and a b -bit block B to return a $2k$ bit output $h(y \| B)$. In the case of SHA512, the hash function used in EdDSA, the compression function `sha512` has $b = 1024$ and $k = 256$ (so the chaining variable is 512 bits and a block is 1024 bits), while $b = 512$ and $k = 128$ for SHA256. In

our language, the Merkle-Damgård transform is a functor $\mathbf{MD} : \mathbf{AF}(\{0,1\}^{b+2k}, \{0,1\}^{2k}) \rightarrow \mathbf{AF}(\{0,1\}^*, \{0,1\}^{2k})$. It is parameterized by a padding function \mathbf{pad} that takes the length ℓ of an input to the hash function and returns a padding string such that $\ell + |\mathbf{pad}(\ell)|$ is a multiple of b . Specifically, $\mathbf{pad}(\ell)$ returns $10^* \langle \ell \rangle$ where $\langle \ell \rangle$ is a 64-bit, resp. 128-bit encoding of ℓ for SHA256 resp. SHA512, and 0^* indicates the minimum number p of 0s needed to make $\ell + 1 + p + 64$, resp. $\ell + 1 + p + 128$ a multiple of b . We also fix an “initialization vector” $IV \in \{0,1\}^{2k}$. Given oracle h , the functor defines hash function $H = \mathbf{MD}[h] : \{0,1\}^* \rightarrow \{0,1\}^{2k}$ as follows:

Functor $\mathbf{MD}[h](X)$

```

 $y[0] \leftarrow IV$ 
 $P \leftarrow \mathbf{pad}(|X|)$  ;  $X'[1] \dots X'[m] \leftarrow X \parallel P$  // Split  $X \parallel P$  into  $b$ -bit blocks
For  $i = 1, \dots, m$  do  $y[i] \leftarrow h(y[i-1] \parallel X'[i])$ 
Return  $y[m]$ 

```

Strictly speaking, the domain is only strings of length less than 2^{64} resp. 2^{128} , but since this is huge in practice, we view the domain as $\{0,1\}^*$.

SIGNATURE SCHEME SYNTAX. We give an enhanced, flexible syntax for a signature scheme DS. We want to cover ROM schemes, which means scheme algorithms have oracle access to a function H , but of what range and domain? Since these can vary from scheme to scheme, we have the scheme begin by naming the function space DS.FS from which H is drawn. We see the key-generation algorithm DS.Kg as first picking a signing key $sk \leftarrow^* \mathbf{DS.SK}$ via a signing-key generation algorithm DS.SK, then obtaining the public verification key $vk \leftarrow \mathbf{DS.PK}[H](sk)$ by applying a deterministic verification-key generation algorithm DS.PK, and finally returning (vk, sk) . (For simplicity, DS.SK, unlike other scheme algorithms, does not have access to H .) We break it up like this because we may need to explicitly refer to the sub-algorithms in constructions. Continuing, via $\sigma \leftarrow \mathbf{DS.Sign}[H](sk, vk, M; r)$ the signing algorithm takes sk, vk , a message $M \in \{0,1\}^*$, and randomness r from the randomness space DS.SR of the algorithm, to return a signature σ . As usual, $\sigma \leftarrow^* \mathbf{DS.Sign}[H](sk, vk, M)$ is shorthand for picking $r \leftarrow^* \mathbf{DS.SR}$ and returning $\sigma \leftarrow \mathbf{DS.Sign}[H](sk, vk, M; r)$. Via $b \leftarrow \mathbf{DS.Vf}[H](vk, M, \sigma)$, the verification algorithm obtains a boolean decision $b \in \{\text{true}, \text{false}\}$ about the validity of the signature. The correctness requirement is that for all $H \in \mathbf{DS.FS}$, all $(vk, sk) \in \text{OUT}(\mathbf{DS.Kg}[H])$, all $M \in \{0,1\}^*$ and all $\sigma \in \text{OUT}(\mathbf{DS.Sign}[H](sk, vk, M))$ we have $\mathbf{DS.Vf}[H](vk, M, \sigma) = \text{true}$.

UF SECURITY. We want to discuss security of a signature scheme DS under different ways in which the functions in DS.FS are chosen or built. Game $\mathbf{G}_{\mathbf{DS}, \mathbf{FF}}^{\text{uf}}$ in Fig. 1 is thus parameterized by a functor $\mathbf{FF} : \mathbf{SS} \rightarrow \mathbf{DS.FS}$. At line 1, a starting function h is chosen from the starting space of the functor, and then the function $H \in \mathbf{DS.FS}$ that the scheme algorithms (key-generation, signing and verification) get as oracle is determined as $H \leftarrow \mathbf{FF}[h]$. The adversary, however, via oracle FO, gets access to h , which here is the random oracle. The rest is as per the usual unforgeability definition. (Given in the standard model in [24] and

Game $\mathbf{G}_{\mathbf{DS}, \mathbf{FF}}^{\text{uf}}$
<p>INIT:</p> <p>1 $h \leftarrow \\$ \mathbf{SS}$; $H \leftarrow \mathbf{FF}[\mathbf{FO}]$; $(vk, sk) \leftarrow \\$ \mathbf{DS.Kg}[H]$; Return vk</p> <p>SIGN(M):</p> <p>2 $\sigma \leftarrow \\$ \mathbf{DS.Sign}[H](sk, vk, M)$; $S \leftarrow S \cup \{M\}$; Return σ</p> <p>FO(X):</p> <p>3 Return $h(X)$</p> <p>FIN(M_*, σ_*):</p> <p>4 If $(M_* \in S)$ then return false</p> <p>5 Return $\mathbf{DS.Vf}[H](vk, M_*, \sigma_*)$</p>

Game $\mathbf{G}_{\mathbf{P}}^{\text{prg}}$	Game $\mathbf{G}_{\mathbf{F}}^{\text{prf}}$
<p>INIT:</p> <p>1 $h \leftarrow \\$ \mathbf{SS}$; $c \leftarrow \\$ \{0, 1\}$</p> <p>2 $s \leftarrow \\$ \{0, 1\}^k$; $y_1 \leftarrow \mathbf{P}[\mathbf{FO}](s)$</p> <p>3 $y_0 \leftarrow \\$ \{0, 1\}^\ell$</p> <p>4 Return y_c</p> <p>FO(X):</p> <p>5 Return $h(X)$</p> <p>FIN(c'):</p> <p>6 Return $(c = c')$</p>	<p>INIT:</p> <p>1 $h \leftarrow \\$ \mathbf{SS}$; $c \leftarrow \\$ \{0, 1\}$; $K \leftarrow \\$ \{0, 1\}^k$</p> <p>FN($X$):</p> <p>2 If $\mathbf{YT}[X] \neq \perp$ then</p> <p>3 If $(c = 1)$ then $\mathbf{YT}[X] \leftarrow \mathbf{F}[\mathbf{FO}](K, X)$</p> <p>4 Else $\mathbf{YT}[X] \leftarrow \\$ R$</p> <p>5 Return $\mathbf{YT}[X]$</p> <p>FO(X):</p> <p>6 Return $h(X)$</p> <p>FIN(c'):</p> <p>7 Return $(c = c')$</p>

Fig. 1. Top: Game defining UF security of signature scheme \mathbf{DS} relative to functor $\mathbf{FF} : \mathbf{SS} \rightarrow \mathbf{DS.FS}$. Bottom Left: Game defining PRG security of functor $\mathbf{P} : \mathbf{SS} \rightarrow \mathbf{AF}(\{0, 1\}^k, \{0, 1\}^\ell)$. Bottom Right: Game defining PRF security of functor $\mathbf{F} : \mathbf{SS} \rightarrow \mathbf{AF}(\{0, 1\}^k \times \{0, 1\}^*, R)$.

extended to the ROM in [10].) We define the UF advantage of adversary \mathcal{A} as $\mathbf{Adv}_{\mathbf{DS}, \mathbf{FF}}^{\text{uf}}(\mathcal{A}) = \Pr[\mathbf{G}_{\mathbf{DS}, \mathbf{FF}}^{\text{uf}}(\mathcal{A})]$.

PRGS AND PRFS. The usual definition of a PRGs is for a function; we define it instead for a functor \mathbf{P} . The game $\mathbf{G}_{\mathbf{P}}^{\text{prg}}$ is in Figure 1. It picks a function h from the starting space \mathbf{SS} of the functor. The functor now determines a function $\mathbf{P}[h] : \{0, 1\}^k \rightarrow \{0, 1\}^\ell$. The game then follows the usual PRG one for

<u>$\overline{\text{DS}}.\text{SK}:$</u> 1 $\overline{sk} \leftarrow \text{\$} \{0, 1\}^k$; Return \overline{sk} <u>$\overline{\text{DS}}.\text{PK}[\text{H}](\overline{sk}):$</u> 2 $e_1 e_2 \leftarrow \text{H}_1(\overline{sk})$; $sk \leftarrow \text{CF}(e_1)$ 3 $vk \leftarrow \text{DS.PK}[\text{H}_3](sk)$ 4 Return vk <u>$\overline{\text{DS}}.\text{Sign}[\text{H}](\overline{sk}, vk, M):$</u> 5 $e_1 e_2 \leftarrow \text{H}_1(\overline{sk})$; $sk \leftarrow \text{CF}(e_1)$ 6 $r \leftarrow \text{H}_2(e_2, M)$ 7 $\sigma \leftarrow \text{DS.Sign}[\text{H}_3](sk, vk, M; r)$ 8 Return σ <u>$\overline{\text{DS}}.\text{Vf}[\text{H}](vk, M, \sigma):$</u> 9 Return $\text{DS.Vf}[\text{H}_3](vk, M, \sigma)$	<u>$\text{DS}^*.\text{SK}:$</u> 1 $\overline{sk} \leftarrow \text{\$} \{0, 1\}^k$; Return \overline{sk} <u>$\text{DS}^*.\text{PK}[\text{G}](\overline{sk}):$</u> 2 $sk \leftarrow \text{CF}(\overline{sk})$ 3 $vk \leftarrow \text{DS.PK}[\text{G}](sk)$ 4 Return vk <u>$\text{DS}^*.\text{Sign}[\text{G}](\overline{sk}, vk, M):$</u> 5 $sk \leftarrow \text{CF}(\overline{sk})$ 6 $\sigma \leftarrow \text{\$} \text{DS.Sign}[\text{G}](sk, vk, M)$ 7 Return σ <u>$\text{DS}^*.\text{Vf}[\text{G}](vk, M, \sigma):$</u> 8 Return $\text{DS.Vf}[\text{G}](vk, M, \sigma)$
---	---

Fig. 2. Left: The signature scheme $\overline{\text{DS}} = \text{DtD}[\text{DS}, \text{CF}]$ constructed by the **DtD** transform applied to signature scheme DS and clamping function $\text{CF}: \{0, 1\}^k \rightarrow \text{OUT}(\text{DS.SK})$. **Right:** The signature scheme $\overline{\text{DS}} = \text{JCI}[\text{DS}, \text{CF}]$ constructed by the **JCI** transform.

this function, additionally giving the adversary oracle access to h via oracle FO . We let $\text{Adv}_{\text{P}}^{\text{prg}}(\mathcal{A}) = 2 \Pr[\text{G}_{\text{P}}^{\text{prg}}(\mathcal{A})] - 1$.

Similarly we extend the usual definition of PRG security to a functor \mathbf{F} , via game $\text{G}_{\mathbf{F}}^{\text{prf}}$ of Figure 1. Here, for h in the starting space SS of the functor, the defined function maps as $\mathbf{F}[h]: \{0, 1\}^k \times \{0, 1\}^* \rightarrow R$ for some k and range set R . We let $\text{Adv}_{\mathbf{F}}^{\text{prf}}(\mathcal{A}) = 2 \Pr[\text{G}_{\mathbf{F}}^{\text{prf}}(\mathcal{A})] - 1$.

4 The soundness of Derive-then-Derandomize

We specify a general signature-hardening transform that we call Derive-then-Derandomize (**DtD**) and prove that it preserves the security of the starting signature scheme.

THE DtD TRANSFORM. Let DS be a given signature scheme that we call the base signature scheme. It will be the (general) Schnorr scheme in our application. Assume for simplicity that its function space DS.FS has arity 1.

The **DtD** (derive then de-randomize) transform constructs a signature scheme $\overline{\text{DS}} = \text{DtD}[\text{DS}, \text{CF}]$ based on DS and a function $\text{CF}: \{0, 1\}^k \rightarrow \text{OUT}(\text{DS.SK})$, called the clamping function, that turns a k -bit string into a signing key for DS . The algorithms of $\overline{\text{DS}}$ are shown in Figure 2. They have access to oracle H that specifies sub-functions $\text{H}_1, \text{H}_2, \text{H}_3$. Function $\text{H}_1: \{0, 1\}^k \rightarrow \{0, 1\}^{2k}$ expands the signing key \overline{sk} of $\overline{\text{DS}}$ into sub-keys e_1 and e_2 . The clamping function is applied to e_1 to get a signing key for the base scheme, and its associated

verification key is returned as the one for the new scheme at line 4. At line 6, function $H_2 : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \text{DS.SR}$ is applied to the second sub-key e_2 and the message M to determine signing randomness r for the line 5 invocation of the base signing algorithm. Finally, $H_3 \in \text{DS.FS}$ is an oracle for the algorithms of DS . Formally the oracle space $\overline{\text{DS.FS}}$ of $\overline{\text{DS}}$ is the arity 3 space consisting of all $H = (H_1, H_2, H_3)$ that map as above.

Viewing the PRG H_1 , PRF H_2 and oracle H_3 for the base scheme as specified in the function space is convenient for our application to EdDSA, where they are all based on **MD** with the *same* underlying idealized compression function.

JUST CLAMP. Given a signature scheme DS and a clamping function $\text{CF} : \{0, 1\}^k \rightarrow \text{OUT}(\text{DS.SK})$, it is useful to also consider the signature scheme $\text{DS}^* = \text{JCI}[\text{DS}, \text{CF}]$ that does just the clamping. The scheme is shown in Figure 2. Its oracle space is the same as that of DS and is assumed to have arity 1. On the right of Figure 2 the function drawn from it is denoted G ; it will be the same as H_3 on the left.

SECURITY OF **DtD**. We study the security of the scheme $\overline{\text{DS}} = \text{DtD}[\text{DS}, \text{CF}]$ obtained via the **DtD** transform.

When we prove security of $\overline{\text{DS}}$, it will be with respect to a functor **FF** that constructs all of H_1, H_2, H_3 . This means that these three functions could all depend on the same starting function that **FF** uses, and in particular not be independent of each other. An important element of the following theorem is that it holds even in this case, managing to reduce security to conditions on the individual functors despite their using related (in fact, the same) underlying starting function.

Theorem 1. *Let DS be a signature scheme. Let $\text{CF} : \{0, 1\}^k \rightarrow \text{OUT}(\text{DS.SK})$ be a clamping function. Let $\overline{\text{DS}} = \text{DtD}[\text{DS}, \text{CF}]$ and $\text{DS}^* = \text{JCI}[\text{DS}, \text{CF}]$ be the signature schemes obtained by the above transforms. Let $\text{FF} : \text{SS} \rightarrow \overline{\text{DS.FS}}$ be a functor that constructs the function H that algorithms of $\overline{\text{DS}}$ use as an oracle. Let \mathcal{A} be an adversary attacking the \mathbf{G}^{uf} security of $\overline{\text{DS}}$. Then there are adversaries $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ such that*

$$\text{Adv}_{\overline{\text{DS}}, \text{FF}}^{\text{uf}}(\mathcal{A}) \leq \text{Adv}_{\text{FF}_1}^{\text{prg}}(\mathcal{A}_1) + \text{Adv}_{\text{FF}_2}^{\text{prf}}(\mathcal{A}_2) + \text{Adv}_{\text{DS}^*, \text{FF}_3}^{\text{uf}}(\mathcal{A}_3).$$

The constructed adversaries preserve the number of FO queries of \mathcal{A} and approximately preserve its running time. Adversary \mathcal{A}_2 makes $Q_{\text{SIGN}}^{\mathcal{A}}$ queries to FN. Adversary \mathcal{A}_3 makes $Q_{\text{SIGN}}^{\mathcal{A}}$ queries to SIGN.

Proof (Theorem 1). The proof uses code-based game playing [11]. Consider the games of Figure 3. Let $\epsilon_i = \Pr[G_i(\mathcal{A})]$ for $i = 0, 1, 2$.

Game G_0 is the \mathbf{G}^{uf} game for $\overline{\text{DS}}$ except that the signature of M is stored in table ST at line 8, and, at line 5, if a signature for M already exists, it is returned directly. Since signing in $\overline{\text{DS}}$ is deterministic, meaning the signature is always the same for a given message and signing key, this does not change what SIGN returns, and thus

$$\begin{aligned} \text{Adv}_{\overline{\text{DS}}, \text{FF}}^{\text{uf}}(\mathcal{A}) &= \epsilon_0 \\ &= (\epsilon_0 - \epsilon_1) + (\epsilon_1 - \epsilon_2) + \epsilon_2. \end{aligned}$$

<p><u>Games G_0, G_1, G_2</u></p> <p>INIT:</p> <p>1 $h \leftarrow \text{SS}$</p> <p>2 $\overline{sk} \leftarrow \{0, 1\}^k$; $e_1 \ e_2 \leftarrow \mathbf{FF}_1[\text{FO}](\overline{sk})$ // Game G_0</p> <p>3 $e_1 \ e_2 \leftarrow \{0, 1\}^{2k}$ // Games G_1, G_2</p> <p>4 $sk \leftarrow \mathbf{CF}(e_1)$; $vk \leftarrow \mathbf{DS.PK}[\mathbf{FF}_3[\text{FO}]](sk)$; Return vk</p> <p>SIGN(M):</p> <p>5 If $\text{ST}[M] \neq \perp$ then return $\text{ST}[M]$</p> <p>6 $r \leftarrow \mathbf{FF}_2[\text{FO}](e_2, M)$ // Games G_0, G_1</p> <p>7 $r \leftarrow \text{DS.SR}$ // Game G_2</p> <p>8 $\text{ST}[M] \leftarrow \mathbf{DS.Sign}[\mathbf{FF}_3[\text{FO}]](sk, vk, M; r)$; Return $\text{ST}[M]$</p> <p>FO(X):</p> <p>9 Return $h(X)$</p> <p>FIN(M_*, σ_*):</p> <p>10 If $(\text{ST}[M_*] \neq \perp)$ then return false</p> <p>11 Return $\mathbf{DS.Vf}[\mathbf{FF}_3[\text{FO}]](vk, M_*, \sigma_*)$</p>
--

Fig. 3. Games for proof of Theorem 1. A line annotated with names of games is included only in those games.

We bound each of the three terms above in turn.

The change in moving to game G_1 is at line 3, where we sample $e_1 \| e_2$ uniformly from the set $\{0, 1\}^{2k}$ rather than obtaining it via $\mathbf{FF}_1[\text{FO}]$ as in game G_0 . We build PRG adversary \mathcal{A}_1 such that

$$\epsilon_0 - \epsilon_1 \leq \mathbf{Adv}_{\mathbf{FF}_1}^{\text{prg}}(\mathcal{A}_1). \quad (1)$$

Adversary \mathcal{A}_1 is playing game $\mathbf{G}_{\mathbf{FF}_1}^{\text{prg}}$. It gets its challenge via $e_1 \| e_2 \leftarrow \mathbf{G}_{\mathbf{FF}_1}^{\text{prg}}.\text{INIT}$. It lets $sk \leftarrow \mathbf{CF}(e_1)$ and $vk \leftarrow \mathbf{DS.PK}[\mathbf{FF}_3[\mathbf{G}_{\mathbf{FF}_1}^{\text{prg}}.\text{FO}]](sk)$ where $\mathbf{G}_{\mathbf{FF}_1}^{\text{prg}}.\text{FO}$ is the oracle provided in its own game. It runs \mathcal{A} , returning vk in response to \mathcal{A} 's INIT query. It answers SIGN queries as do G_0, G_1 except that it uses $\mathbf{G}_{\mathbf{FF}_1}^{\text{prg}}.\text{FO}$ in place of FO at lines 6,8. As part of this simulation, it maintains table ST. It answers FO queries via $\mathbf{G}_{\mathbf{FF}_1}^{\text{prg}}.\text{FO}$. When \mathcal{A} calls FIN(M_*, σ_*), adversary \mathcal{A}_1 lets $c' \leftarrow 1$ if $\mathbf{DS.Vf}[\mathbf{FF}_3[\mathbf{G}_{\mathbf{FF}_1}^{\text{prg}}.\text{FO}]](vk, M_*, \sigma_*)$ is true and $\text{ST}[M_*] = \perp$, and otherwise lets $c' \leftarrow 0$. It then calls $\mathbf{G}_{\mathbf{FF}_1}^{\text{prg}}.\text{FIN}(c')$. When the challenge bit c in game $\mathbf{G}_{\mathbf{FF}_1}^{\text{prg}}$ is $c = 1$, the view of \mathcal{A} is as in G_0 , and when $c = 0$ it is as in G_1 , which explains Eq. (1).

Moving to G_2 , the change is that line 6 is replaced by line 7, meaning signing coins are now chosen at random from the randomness space DS.SR of DS. We build PRF adversary \mathcal{A}_2 such that

$$\epsilon_1 - \epsilon_2 \leq \mathbf{Adv}_{\mathbf{FF}_2}^{\text{prf}}(\mathcal{A}_2). \quad (2)$$

Adversary \mathcal{A}_2 is playing game $\mathbf{G}_{\mathbf{FF}_2}^{\text{prf}}$. It picks $e_1 \| e_2 \leftarrow_{\$} \{0, 1\}^{2k}$. It lets $sk \leftarrow \text{CF}(e_1)$ and $vk \leftarrow \text{DS.PK}[\mathbf{FF}_3[\mathbf{G}_{\mathbf{FF}_2}^{\text{prf}}.\text{FO}]](sk)$ where $\mathbf{G}_{\mathbf{FF}_2}^{\text{prf}}.\text{FO}$ is the oracle provided in its own game. It runs \mathcal{A} , returning vk in response to \mathcal{A} 's INIT query. It answers SIGN queries as does G_1 except that it uses $\mathbf{G}_{\mathbf{FF}_2}^{\text{prf}}.\text{FN}$ in place of $\mathbf{FF}_2[\text{FO}]$ at line 6 and $\mathbf{G}_{\mathbf{FF}_2}^{\text{prf}}.\text{FO}$ in place of FO in line 8. As part of this simulation, it maintains table ST. It answers FO queries via $\mathbf{G}_{\mathbf{FF}_2}^{\text{prf}}.\text{FO}$. When \mathcal{A} calls $\text{FIN}(M_*, \sigma_*)$, adversary \mathcal{A}_2 lets $c' \leftarrow 1$ if $\text{DS.Vf}[\mathbf{FF}_3[\mathbf{G}_{\mathbf{FF}_2}^{\text{prf}}.\text{FO}]](vk, M_*, \sigma_*)$ is true and $\text{ST}[M_*] = \perp$, and otherwise lets $c' \leftarrow 0$. It then calls $\mathbf{G}_{\mathbf{FF}_2}^{\text{prf}}.\text{FIN}(c')$. When the challenge bit c in game $\mathbf{G}_{\mathbf{FF}_2}^{\text{prf}}$ is $c = 1$, the view of \mathcal{A} is as in G_1 , and when $c = 0$ it is as in G_2 , which explains Eq. (2).

Finally we build adversary \mathcal{A}_3 such that

$$\epsilon_2 \leq \mathbf{Adv}_{\text{DS}^*, \mathbf{FF}_3}^{\text{uf}}(\mathcal{A}_3). \quad (3)$$

Adversary \mathcal{A}_3 is playing game $\mathbf{G}_{\text{DS}^*, \mathbf{FF}_3}^{\text{uf}}$. It lets $vk \leftarrow \mathbf{G}_{\text{DS}^*, \mathbf{FF}_3}^{\text{uf}}.\text{INIT}$. It runs \mathcal{A} , returning vk in response to \mathcal{A} 's INIT query. When \mathcal{A} makes query M to SIGN, it answers as per the following:

If $\text{ST}[M] \neq \perp$ then return $\text{ST}[M]$
 $\text{ST}[M] \leftarrow_{\$} \mathbf{G}_{\text{DS}^*, \mathbf{FF}_3}^{\text{uf}}.\text{SIGN}(M)$; Return $\text{ST}[M]$

Note that memoizing signatures in ST is important here to ensure that the SIGN queries of \mathcal{A} are correctly simulated. It answers FO queries via $\mathbf{G}_{\text{DS}^*, \mathbf{FF}_3}^{\text{uf}}.\text{FO}$. When \mathcal{A} calls $\text{FIN}(M_*, \sigma_*)$, adversary \mathcal{A}_2 calls $\mathbf{G}_{\text{DS}^*, \mathbf{FF}_3}^{\text{uf}}.\text{FIN}(M_*, \sigma_*)$. The distribution of signatures that \mathcal{A} is given, and of the keys underlying them, is as in G_2 , which explains Eq. (3).

Note that the constructed adversaries having access to oracle FO in their games is important to their ability to simulate \mathcal{A} faithfully.

With regard to the costs (number of queries, running time) of the constructed adversaries, recall that we have defined these as the costs in the execution of the adversary with the game that the adversary is playing, so for example the number of queries to FO includes the ones made by algorithms executed in the game. When this is taken into account, queries to FO are preserved, and the other claims are direct. \square

SECURITY OF JCI. We have now reduced the security of $\overline{\text{DS}}$ to that of DS^* . To further reduce the security of DS^* to that of DS, we give a general result on clamping. Let $\mathcal{K} = \text{OUT}(\text{DS.SK})$ and let $\text{CF} : \{0, 1\}^k \rightarrow \mathcal{K}$ be a clamping function. As per terminology in Section 2, recall that $\text{Img}(\text{CF}) = \{ \text{CF}(sk) : |sk| = k \} \subseteq \mathcal{K}$ is the image of the clamping function, and CF is regular if every $y \in \text{Img}(\text{CF})$ has the same number of pre-images under CF.

Theorem 2. *Let DS be a signature scheme such that DS.SK draws its signing key $sk \leftarrow_{\$} \mathcal{K}$ at random from a set \mathcal{K} . Let $\text{CF} : \{0, 1\}^k \rightarrow \mathcal{K}$ be a regular clamping function. Let $\delta = |\text{Img}(\text{CF})|/|\mathcal{K}| > 0$. Let $\text{DS}^* = \text{JCI}[\text{DS}, \text{CF}]$ be the signature scheme obtained by the just-clamp transform. Let $\mathbf{FF} : \text{SS} \rightarrow \text{DS.FS}$ be any functor. Let \mathcal{B} be an adversary attacking the $\mathbf{G}_{\text{DS}^*, \mathbf{FF}}^{\text{uf}}$ security of DS^* . Then*

$$\mathbf{Adv}_{\text{DS}^*, \mathbf{FF}}^{\text{uf}}(\mathcal{B}) \leq (1/\delta) \cdot \mathbf{Adv}_{\text{DS}, \mathbf{FF}}^{\text{uf}}(\mathcal{B}).$$

<u>DS.SK:</u> 1 $s \leftarrow \mathbb{Z}_p$ 2 Return s <u>DS.PK(s):</u> 3 $A \leftarrow s \cdot B$; Return A <u>DS.Sign[H](s, A, M):</u> 4 $r \leftarrow \mathbb{Z}_p$; $R \leftarrow r \cdot B$ 5 $c \leftarrow H(R \ A \ M)$ 6 $z \leftarrow (sc + r) \bmod p$ 7 Return (R, z) <u>DS.Vf[H](A, M, σ):</u> 8 $(R, z) \leftarrow \sigma$ 9 $c \leftarrow H(R \ A \ M)$ 10 Return $VF(A, R, c, z)$	<u>DS.SK:</u> 1 $sk \leftarrow \{0, 1\}^k$; Return sk <u>DS.PK(sk):</u> 2 $e_1 \ e_2 \leftarrow H_1(sk)$; $s \leftarrow CF(e_1)$ 3 $A \leftarrow s \cdot B$; Return A <u>DS.Sign[H](sk, A, M):</u> 4 $e_1 \ e_2 \leftarrow H_1(sk)$; $s \leftarrow CF(e_1)$ 5 $r \leftarrow H_2(e_2, M)$; $R \leftarrow r \cdot B$ 6 $c \leftarrow H_3(R \ A \ M)$ 7 $z \leftarrow (sc + r) \bmod p$ 8 Return (R, z) <u>DS.Vf[H](A, M, σ):</u> 9 $(R, z) \leftarrow \sigma$ 10 $c \leftarrow H_3(R \ A \ M) \bmod p$ 11 Return $VF(A, R, c, z)$
<u>CF(e)</u> // $e \in \{0, 1\}^k$: 12 $t \leftarrow 2^{k-2}$ 13 for $i \in [4..k-2]$ 14 $t \leftarrow t + 2^{i-1} \cdot e[i]$ 15 $s \leftarrow t \bmod p$ 16 return s	<u>sVF(A, R, c, z):</u> 1 Return $(z \cdot B = c \cdot A + R)$ <u>pVF(A, R, c, z):</u> 1 Return $2^f(z \cdot B) = 2^f(c \cdot A + R)$

Fig. 4. Top Left: the Schnorr scheme. **Top Right:** The EdDSA scheme. **Bottom Left:** EDDSA clamping function (generalized for any k ; in the original definition, $k = 256$). **Bottom Right:** Strict and Permissive verification algorithms as choices for VF.

Proof (Theorem 2). We consider running \mathcal{B} in game $\mathbf{G}_{\text{DS}, \mathbf{FF}}^{\text{uf}}$, where the signing key is $sk \leftarrow \mathcal{K}$. With probability δ we have $sk \in \text{Img}(\text{CF})$. Due to the regularity of CF, key sk now has the same distribution as a key $\text{CF}(\overline{sk})$ for $\overline{sk} \leftarrow \{0, 1\}^k$ drawn in game $\mathbf{G}_{\text{DS}^*, \mathbf{FF}}^{\text{uf}}$. Thus $\text{Adv}_{\text{DS}, \mathbf{FF}}^{\text{uf}}(\mathcal{B}) \geq \delta \cdot \text{Adv}_{\text{DS}^*, \mathbf{FF}}^{\text{uf}}(\mathcal{B})$. \square

5 Security of EdDSA

THE SCHNORR SCHEME. Let the prime-order group \mathbb{G}_p of k -bit strings with generator B be as described in Section 2. The algorithms of the Schnorr signature scheme $\text{DS} = \text{Sch}$ are shown on the left in Figure 4. The function space DS.FS is $\text{AF}(\{0, 1\}^*, \mathbb{Z}_p)$. (Implementations may use a hash function that outputs a string and embed the result in \mathbb{Z}_p but following prior proofs [1] we view the

hash function as directly mapping into \mathbb{Z}_p .) Verification is parameterized by an algorithm VF to allow us to consider strict and permissive verification in a modular way. The corresponding choices of verification algorithms are at the bottom of Figure 4. The signing randomness space is $\text{DS.SR} = \mathbb{Z}_p$.

Schnorr signatures have a few variants that differ in details. In Schnorr’s paper [40], the challenge is $c = H(R\|M) \bmod p$. Our inclusion of the public key in the input to H follows Bernstein [13] and helps here because it is what EdDSA does. It doesn’t affect security. (The security of the scheme that includes the public key in the hash input is implied by the security of the one that doesn’t via a reduction that includes the public key in the message.) Also in [40], the signature is (c, z) . The version we use, where it is (R, z) , is from [1]. However, BBSS [2] shows that these versions have equivalent security.

THE EDDSA SCHEME. Let the prime-order group \mathbb{G}_p of k -bit strings with generator B be as before and assume $2^{k-5} < p < 2^k$. Let $\text{CF} : \{0, 1\}^k \rightarrow \mathbb{Z}_p$ be the clamping function shown at the bottom of Figure 4. The algorithms of the scheme $\overline{\text{DS}}$ are shown on the right side of Figure 4. The key length is k . As before, the verification algorithm VF is a parameter. The H available to the algorithms defines three sub-functions. The first, $H_1 : \{0, 1\}^k \rightarrow \{0, 1\}^{2k}$, is used at lines 2,4, where its output is parsed into k -bit halves. The second, $H_2 : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \mathbb{Z}_p$, is used at line 5 for de-randomization. The third, $H_3 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$, plays the role of the function H for the Schnorr schemes. Formally, $\overline{\text{DS.FS}}$ is the arity-3 function space consisting of all H mapping as just indicated.

In [14, 16], the output of the clamping is an integer that (in our notation) is in the range $2^{k-2}, \dots, 2^{k-1} - 8$. When used in the scheme, however, it is (implicitly) modulo p . It is convenient for our analysis, accordingly, to define CF to be the result modulo p of the actual clamping. Note that in EdDSA the prime p has magnitude a little more than 2^{k-4} and less than 2^{k-3} .

There are several versions of EdDSA depending on the choice for verification algorithms: strict, permissive or batch VF. We specify the first two choices in Figure 4. Our results hold for all choices of VF, meaning EdDSA is secure with respect to VF assuming Schnorr is secure with respect to VF. It is in order to make this general claim that we abstract out VF.

SECURITY OF EDDSA WITH INDEPENDENT ROS. As a warm-up, we show security of EdDSA when the three functions it uses are independent random oracles, the setting assumed by BCJZ [16]. However, while they assume hardness of DL, our result is more general, assuming only security of Schnorr with a monolithic random oracle. We can then use known results on Schnorr [37, 1] to recover the result of BCJZ [16], but the proof is simpler and more modular. Also, other known results on Schnorr [39, 5, 22] can be applied to get better bounds. Following this, we will turn to the “real” case, where the three functions are all **MD** with a random compression function.

The Theorem below is for a general prime $p > 2^{k-5}$ but in EdDSA the prime is $2^{k-4} < p < 2^{k-3}$ so the value of δ below is $\delta = 2^{k-5}/p > 2^{k-5}/2^{k-3} = 1/4$, so the factor $1/\delta$ is ≤ 4 . We capture the three functions of EdDSA being independent random oracles by setting functor **P** below to the identity functor, and similarly

<p>Functor $\mathbf{S}_1[h](sk)$: // $sk = k$</p> <p>2 $e \leftarrow \mathbf{MD}[h](sk)$; Return e // $e = 2k$</p> <p>Functor $\mathbf{S}_2[h](e_2, M)$: // $e_2 = k$</p> <p>3 Return $\mathbf{MD}[h](e_2 M) \bmod p$</p> <p>Functor $\mathbf{S}_3[h](X)$: // also called Mod-MD</p> <p>4 Return $\mathbf{MD}[h](X) \bmod p$</p>

Fig. 5. The arity-3 functor \mathbf{S} for EdDSA. Here $h : \{0, 1\}^{b+2k} \rightarrow \{0, 1\}^{2k}$ is a compression function.

capture Schnorr being with a monolithic random oracle by setting \mathbf{R} to be the identity functor.

Theorem 3. *Let \mathbf{DS} be the Schnorr signature scheme of Figure 4. Let $\mathbf{CF} : \{0, 1\}^k \rightarrow \mathbb{Z}_p$ be the clamping function of Figure 4. Assume $p > 2^{k-5}$ and let $\delta = 2^{k-5}/p$. Let $\overline{\mathbf{DS}} = \mathbf{DtD}[\mathbf{DS}, \mathbf{CF}]$ be the EdDSA signature scheme. Let $\mathbf{R} : \mathbf{AF}(\{0, 1\}^*, \mathbb{Z}_p) \rightarrow \mathbf{AF}(\{0, 1\}^*, \mathbb{Z}_p)$ be the identity functor. Let $\mathbf{P} : \overline{\mathbf{DS}}.\mathbf{FS} \rightarrow \overline{\mathbf{DS}}.\mathbf{FS}$ be the identity functor. Let \mathcal{A} be an adversary attacking the \mathbf{G}^{uf} security of $\overline{\mathbf{DS}}$. Then there is an adversary \mathcal{B} such that*

$$\mathbf{Adv}_{\overline{\mathbf{DS}}, \mathbf{P}}^{\text{uf}}(\mathcal{A}) \leq (1/\delta) \cdot \mathbf{Adv}_{\overline{\mathbf{DS}}, \mathbf{R}}^{\text{uf}}(\mathcal{B}) + \frac{2 \cdot Q_{\text{FO}}^{\mathcal{A}}}{2^k}.$$

Adversary \mathcal{B} preserves the queries and running time of \mathcal{A} .

Proof (Theorem 3). Let $\mathbf{DS}^* = \mathbf{JCl}[\mathbf{Sch}, \mathbf{CF}]$. By Theorem 1, we have

$$\mathbf{Adv}_{\overline{\mathbf{DS}}, \mathbf{P}}^{\text{uf}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathbf{P}_1}^{\text{prg}}(\mathcal{A}_1) + \mathbf{Adv}_{\mathbf{P}_2}^{\text{prf}}(\mathcal{A}_2) + \mathbf{Adv}_{\mathbf{DS}^*, \mathbf{P}_3}^{\text{uf}}(\mathcal{A}_3).$$

It is easy to see that

$$\begin{aligned} \mathbf{Adv}_{\mathbf{P}_1}^{\text{prg}}(\mathcal{A}_1) &\leq \frac{Q_{\text{FO}}^{\mathcal{A}_1}}{2^k} = \frac{Q_{\text{FO}}^{\mathcal{A}}}{2^k} \\ \mathbf{Adv}_{\mathbf{P}_2}^{\text{prf}}(\mathcal{A}_2) &\leq \frac{Q_{\text{FO}}^{\mathcal{A}_2}}{2^k} = \frac{Q_{\text{FO}}^{\mathcal{A}}}{2^k}. \end{aligned}$$

Under the assumption $p > 2^{k-5}$ made in the theorem, BCJZ [16] established that $|\text{Img}(\mathbf{CF})| = 2^{k-5}$. So $|\text{Img}(\mathbf{CF})|/|\mathbb{Z}_p| = 2^{k-5}/p = \delta$. Let $\mathcal{B} = \mathcal{A}_3$ and note that $\mathbf{P}_3 = \mathbf{R}$. So by Theorem 2 we have

$$\mathbf{Adv}_{\mathbf{DS}^*, \mathbf{P}_3}^{\text{uf}}(\mathcal{A}_3) \leq (1/\delta) \cdot \mathbf{Adv}_{\mathbf{DS}, \mathbf{R}}^{\text{uf}}(\mathcal{B}). \quad (4)$$

Collecting terms, we obtain the claimed bound stated in Theorem 3. \square

ANALYSIS OF THE \mathbf{S} FUNCTOR. Let $\overline{\mathbf{DS}}$ be the result of the \mathbf{DtD} transform applied to \mathbf{Sch} and a clamping function $\mathbf{CF} : \{0, 1\}^k \rightarrow \mathbb{Z}_p$. Security of EdDSA is captured as security in game $\mathbf{G}_{\overline{\mathbf{DS}}, \mathbf{S}}^{\text{uf}}$ when \mathbf{S} is the functor that builds the component hash functions in the way that EdDSA does, namely from a MD-hash function. To evaluate this security, we start by defining the functor \mathbf{S} in Figure 5. It is an arity-3 functor, and we separately specify $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3$. (Functor \mathbf{S}_3

Games $G_0, \boxed{G_1}$
INIT: 1 $sk \leftarrow \{0, 1\}^k$; $e \leftarrow \{0, 1\}^{2k}$ 2 Return e
FO(X): 3 If $FT[X] \neq \perp$ then return $FT[X]$ 4 $Y \leftarrow \{0, 1\}^{2k}$ 5 If $X = IV \ sk \ P$ then $\text{bad} \leftarrow \text{true}$; $\boxed{Y \leftarrow e}$ 6 $FT[X] \leftarrow Y$; Return $FT[X]$
FIN(c'): 7 Return $(c' = 1)$

Fig. 6. Games G_0 and G_1 in the proof of Lemma 4. Boxed code is only in G_1 .

will be called **Mod-MD** in later analyses.) The starting space, from which h is drawn, is $\text{AF}(\{0, 1\}^{b+2k}, \{0, 1\}^{2k})$, the set of compression functions. The prime p is as before, and is public.

We want to establish the three assumptions of Theorem 1. Namely: (1) \mathbf{S}_1 is PRG-secure (2) \mathbf{S}_2 is PRF secure and (3) security holds in game $\mathbf{G}_{\text{Sch}^*, \mathbf{S}_3}^{\text{uf}}$ where $\text{Sch}^* = \mathbf{JCI}[\text{Sch}, \text{CF}]$. Bridging from Sch^* to Sch itself will use Theorem 2.

Lemma 4. *Let functor $\mathbf{S}_1 : \text{AF}(\{0, 1\}^{b+2k}, \{0, 1\}^{2k}) \rightarrow \text{AF}(\{0, 1\}^k, \{0, 1\}^{2k})$ be defined as in Figure 5. Let \mathcal{A}_1 be an adversary. Then*

$$\text{Adv}_{\mathbf{S}_1}^{\text{prg}}(\mathcal{A}_1) \leq \frac{Q_{\text{FO}}^{\mathcal{A}_1}}{2^k} \quad (5)$$

Proof (Lemma 4). Since the input sk to $\mathbf{S}_1[h]$ is k -bits long, the **MD** transform defined in Section 3 only iterates once and the output is $e = h(IV \| sk \| P)$, for padding $P \in \{0, 1\}^{3k}$ and initialization vector $IV \in \{0, 1\}^{2k}$ that are fixed and known. Now consider the games in Figure 6, where the boxed code is only in G_1 . Then we have

$$\begin{aligned} \text{Adv}_{\mathbf{S}_1}^{\text{prg}}(\mathcal{A}_1) &= \Pr[G_1(\mathcal{A}_1)] - \Pr[G_0(\mathcal{A}_1)] \\ &\leq \Pr[G_0(\mathcal{A}_1) \text{ sets bad}] \\ &\leq \frac{Q_{\text{FO}}^{\mathcal{A}_1}}{2^k}. \end{aligned}$$

The second line above is by the Fundamental Lemma of Game Playing, which applies since G_0, G_1 are identical-until-bad. \square

Lemma 5. *Let functor $\mathbf{S}_2 : \text{AF}(\{0, 1\}^{b+2k}, \{0, 1\}^{2k}) \rightarrow \text{AF}(\{0, 1\}^k \times \{0, 1\}^*, \mathbb{Z}_p)$ be defined as in Figure 5. Let ℓ be an integer such that all messages queried to*

FO are no more than $b \cdot (\ell - 1) - k$ bits long. Let \mathcal{A}_2 be an adversary. Then

$$\text{Adv}_{\mathbf{S}_2}^{\text{prf}}(\mathcal{A}_2) \leq \frac{Q_{\text{FO}}^{\mathcal{A}_2}}{2^k} + \frac{2p(q_{\text{FO}}^{\mathcal{A}_2} + \ell Q_{\text{FN}}^{\mathcal{A}_2})}{2^{2k}} + \frac{(q_{\text{FO}}^{\mathcal{A}_2} + \ell Q_{\text{FN}}^{\mathcal{A}_2})^2}{2^{2k}} + \frac{pq_{\text{FO}}^{\mathcal{A}_2} \cdot \ell Q_{\text{FN}}^{\mathcal{A}_2}}{2^{2k}}.$$

Proof (Lemma 5). In Section 6, we prove the indistinguishability of functor \mathbf{S}_3 (c.f. Figure 5), which we also call **Mod-MD**. Define $\mathbf{R} : \text{AF}(\{0, 1\}^*, \mathbb{Z}_p) \rightarrow \text{AF}(\{0, 1\}^k \times \{0, 1\}^*, \mathbb{Z}_p)$ to be the identity functor such that $\mathbf{R}[\mathbf{H}](x, y) = \mathbf{H}(x \parallel y)$ for all x, y, \mathbf{H} in the appropriate domains. Notice that when \mathbf{R} is given access to the **Mod-MD** functor as its oracle, the resulting functor is exactly \mathbf{S}_2 . Using this property, we will reduce the PRF security of functor \mathbf{S}_2 to the indistinguishability of **Mod-MD**.

For any simulator algorithm \mathcal{S} , the indistinguishability composition theorem [30] grants the existence of distinguisher \mathcal{D} and adversary \mathcal{A}_5 such that

$$\text{Adv}_{\mathbf{S}_2}^{\text{prf}}(\mathcal{A}_2) \leq \text{Adv}_{\mathbf{R}}^{\text{prf}}(\mathcal{A}_5) + \text{Adv}_{\text{Mod-MD}, \mathcal{S}}^{\text{indiff}}(\mathcal{D}).$$

We let \mathcal{S} be the simulator guaranteed by Theorem 8 and separately bound each of these terms. Adversary \mathcal{A}_5 simulates the PRF game for its challenger \mathcal{A}_2 by forwarding all FN queries to its own FN oracle and answering FO queries using the simulator, which has access to the FO oracle of \mathcal{A}_5 . Since the simulator is efficient and makes at most one query to its oracle each time it is run, we can say the runtime of \mathcal{A}_5 is approximately the same as that of \mathcal{A}_2 . \mathcal{A}_5 makes the same number of FN and FO queries as \mathcal{A}_2 .

Next, we want to compute $\text{Adv}_{\mathbf{R}}^{\text{prf}}(\mathcal{A}_5)$. When \mathbf{R} is evaluated with access to a random function \mathbf{h} , its outputs are random unless the adversary makes a relevant query involving the secret key. The adversary can only distinguish if the output of FN is randomly sampled or from $\mathbf{R}[\mathbf{h}]$ if it queries FO on the k -bit secret key (e_2), which has probability $\frac{1}{2^k}$ for a single query. Taking a union bound over all FO queries, we have

$$\text{Adv}_{\mathbf{R}}^{\text{prf}}(\mathcal{A}_5) \leq \frac{Q_{\text{FO}}^{\mathcal{A}_2}}{2^k}.$$

Distinguisher \mathcal{D} simulates the PRF game for \mathcal{A}_2 , by replacing functor **Mod-MD** with its own PRIV oracle within the FN oracle and forwarding \mathcal{A}_2 's direct FO queries to PUB. \mathcal{D} hence makes $Q_{\mathcal{A}_2}^{\text{FN}}$ queries to PRIV of maximum length $b \cdot (\ell - 1)$ and $q_{\mathcal{A}_2}^{\text{FO}}$ to PUB. To bound the second term, we apply Theorem 8 on the indistinguishability of shrink-MD transforms. This theorem is parameterized by two numbers γ and ϵ ; in Section 6, we show that **Mod-MD** belongs to the shrink-MD class for $\gamma = \lfloor \frac{2^{2k}}{p} \rfloor$ and $\epsilon = \frac{p}{2^{2k}}$. Then the theorem gives

$$\text{Adv}_{\text{Mod-MD}, \mathcal{S}}^{\text{indiff}}(\mathcal{D}) \leq 2(Q_{\text{PUB}}^{\mathcal{D}} + \ell Q_{\text{PRIV}}^{\mathcal{D}})\epsilon + \frac{(Q_{\text{PUB}}^{\mathcal{D}} + \ell Q_{\text{PRIV}}^{\mathcal{D}})^2}{2^{2k}} + \frac{Q_{\text{PUB}}^{\mathcal{D}} \cdot \ell Q_{\text{PRIV}}^{\mathcal{D}}}{\gamma}.$$

By substituting $Q_{\text{PUB}}^{\mathcal{D}} = q_{\text{FO}}^{\mathcal{A}_2}$ and $Q_{\text{PRIV}}^{\mathcal{D}} = Q_{\text{FN}}^{\mathcal{A}_2}$, we obtain the bound stated in the theorem. \square

The following considers the UF security of $\text{DS}^* = \text{JCI}[\text{Sch}, \text{CF}]$ with the hash function being an MD one, and reduces this to the UF security of the same scheme with the hash function being a monolithic random oracle. Formally, the latter is captured by game $\mathbf{G}_{\text{DS}^*, \mathbf{R}}^{\text{uf}}$ where \mathbf{R} is the identity functor.

Lemma 6. *Let functor $\mathbf{S}_3 : \text{AF}(\{0, 1\}^{b+2k}, \{0, 1\}^{2k}) \rightarrow \text{AF}(\{0, 1\}^*, \mathbb{Z}_p)$ be defined as in Figure 5. Assume $2^k > p$. Let $\text{DS}^* = \text{JCl}[\text{Sch}, \text{CF}]$ where $\text{CF} : \{0, 1\}^k \rightarrow \mathbb{Z}_p$ is a clamping function. Let $\mathbf{R} : \text{AF}(\{0, 1\}^*, \mathbb{Z}_p) \rightarrow \text{AF}(\{0, 1\}^*, \mathbb{Z}_p)$ be the identity functor, meaning $\mathbf{R}[\mathbf{H}] = \mathbf{H}$. Let \mathcal{A}_3 be a \mathbf{G}^{uf} adversary making $Q_{\text{FO}}^{\mathcal{A}_3}, Q_{\text{SIGN}}^{\mathcal{A}_3}$ queries to its respective oracles, and let ℓ be an integer such that the maximum message length \mathcal{A}_3 queries to SIGN is at most $b \cdot (\ell - 1) - 2k$ bits. Then we can construct adversary \mathcal{A}_4 such that*

$$\text{Adv}_{\text{DS}^*, \mathbf{S}_3}^{\text{uf}}(\mathcal{A}_3) \leq \text{Adv}_{\text{DS}^*, \mathbf{R}}^{\text{uf}}(\mathcal{A}_4) + \frac{2p(q_{\text{FO}}^{\mathcal{A}_3} + \ell Q_{\text{SIGN}}^{\mathcal{A}_3})}{2^{2k}} \quad (6)$$

$$+ \frac{(q_{\text{FO}}^{\mathcal{A}_3} + \ell Q_{\text{SIGN}}^{\mathcal{A}_3})^2}{2^{2k}} + \frac{pq_{\text{FO}}^{\mathcal{A}_3} \cdot \ell Q_{\text{SIGN}}^{\mathcal{A}_3}}{2^{2k}}. \quad (7)$$

Adversary \mathcal{A}_4 has approximately equal runtime and query complexity to \mathcal{A}_3 .

Proof (Lemma 6). Again, we rely on the indistinguishability of functor $\mathbf{S}_3 = \text{Mod-MD}$, as shown in Section 6. The general indistinguishability composition theorem [30] states that for any simulator \mathcal{S} and adversary \mathcal{A}_3 , there exist distinguisher \mathcal{D} and adversary \mathcal{A}_4 such that

$$\text{Adv}_{\text{DS}^*, \mathbf{S}_3}^{\text{uf}}(\mathcal{A}_3) \leq \text{Adv}_{\text{DS}^*, \mathbf{R}}^{\text{uf}}(\mathcal{A}_4) + \text{Adv}_{\mathbf{S}_3, \mathcal{S}}^{\text{indiff}}(\mathcal{D}).$$

Let \mathcal{S} be the simulator whose existence is implied by Theorem 8. The distinguisher runs the unforgeability game for its adversary, replacing $\mathbf{S}_3[\text{FO}]$ in scheme algorithms and adversarial FO queries with its PRIV and PUB oracles respectively. It makes $q_{\text{FO}}^{\mathcal{A}_3}$ queries to PUB and $Q_{\text{SIGN}}^{\mathcal{A}_3}$ queries to PRIV, and the maximum length of any query to PRIV is $b \cdot (\ell - 1)$ bits because each element of group \mathbb{G}_p is a k -bit string (c.f. Section 2). We apply Theorem 8 to obtain the bound

$$\text{Adv}_{\mathbf{S}_3, \mathcal{S}}^{\text{indiff}}(\mathcal{D}) \leq 2(q_{\text{FO}}^{\mathcal{A}_3} + \ell Q_{\text{SIGN}}^{\mathcal{A}_3})\epsilon + \frac{(q_{\text{FO}}^{\mathcal{A}_3} + \ell Q_{\text{SIGN}}^{\mathcal{A}_3})^2}{2^{2k}} + \frac{q_{\text{FO}}^{\mathcal{A}_3} \cdot \ell Q_{\text{SIGN}}^{\mathcal{A}_3}}{\gamma}.$$

Adversary \mathcal{A}_4 is a wrapper for \mathcal{A}_3 , which answers all of its queries to FO by running \mathcal{S} with access to its own FO oracle; since the simulator runs in constant time and makes only one query to its oracle, the runtime and query complexity approximately equal those of \mathcal{A}_3 .

Substituting $\frac{1}{\gamma} \geq \frac{p}{2^{2k}}$ and $\epsilon = \frac{p}{2^{2k}}$ gives the bound. \square

SECURITY OF EdDSA WITH MD. We now want to conclude security of EdDSA, with an MD-hash function, assuming security of Schnorr with a monolithic random oracle. The Theorem is for a general prime p in the range $2^k > p > 2^{k-5}$ but in EdDSA the prime is $2^{k-4} < p < 2^{k-3}$ so the value of δ below is $\delta = 2^{k-5}/p > 2^{k-5}/2^{k-3} = 1/4$, so the factor $1/\delta$ is ≤ 4 . Again recall our convention that query counts of an adversary include those made by oracles in its game, implying for example that $Q_{\text{FO}}^{\mathcal{A}} \geq Q_{\text{SIGN}}^{\mathcal{A}}$.

Theorem 7. *Let DS be the Schnorr signature scheme of Figure 4. Let $\text{CF} : \{0, 1\}^k \rightarrow \mathbb{Z}_p$ be the clamping function of Figure 4. Assume $2^k > p > 2^{k-5}$ and let $\delta = 2^{k-5}/p$. Let $\overline{\text{DS}} = \text{DtD}[\text{DS}, \text{CF}]$ be the EdDSA signature scheme. Let $\mathbf{R} : \text{AF}(\{0, 1\}^*, \mathbb{Z}_p) \rightarrow \text{AF}(\{0, 1\}^*, \mathbb{Z}_p)$ be the identity functor. Let \mathbf{S} be the*

functor of Figure 5. Let \mathcal{A} be an adversary attacking the \mathbf{G}^{uf} security of $\overline{\text{DS}}$. Again let $b \cdot (\ell - 1) - 2k$ be the maximum length in bits of a message input to SIGN . Then there is an adversary \mathcal{B} such that

$$\begin{aligned} \mathbf{Adv}_{\overline{\text{DS}}, \mathbf{S}}^{\text{uf}}(\mathcal{A}) \leq & (1/\delta) \cdot \mathbf{Adv}_{\text{DS}, \mathbf{R}}^{\text{uf}}(\mathcal{B}) + \frac{Q_{\text{FO}}^{\mathcal{A}}}{2^{k-1}} + \frac{p(q_{\text{FO}}^{\mathcal{A}} + \ell Q_{\text{SIGN}}^{\mathcal{A}})}{2^{2k-2}} \\ & + \frac{(q_{\text{FO}}^{\mathcal{A}} + \ell Q_{\text{SIGN}}^{\mathcal{A}})^2}{2^{2k-1}} + \frac{pq_{\text{FO}}^{\mathcal{A}} \cdot \ell Q_{\text{SIGN}}^{\mathcal{A}}}{2^{2k-1}}. \end{aligned}$$

Adversary \mathcal{B} preserves the queries and running time of \mathcal{A} .

Proof (Theorem 7). Let $\text{DS}^* = \mathbf{JCl}[\text{Sch}, \text{CF}]$. By Theorem 1, we have

$$\mathbf{Adv}_{\overline{\text{DS}}, \mathbf{S}}^{\text{uf}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathbf{S}_1}^{\text{prg}}(\mathcal{A}_1) + \mathbf{Adv}_{\mathbf{S}_2}^{\text{prf}}(\mathcal{A}_2) + \mathbf{Adv}_{\text{DS}^*, \mathbf{S}_3}^{\text{uf}}(\mathcal{A}_3).$$

Now applying Lemma 4, we have

$$\mathbf{Adv}_{\mathbf{S}_1}^{\text{prg}}(\mathcal{A}_1) \leq \frac{Q_{\text{FO}}^{\mathcal{A}}}{2^k}.$$

Applying Lemma 5, we have

$$\mathbf{Adv}_{\mathbf{S}_2}^{\text{prf}}(\mathcal{A}_2) \leq \frac{Q_{\text{FO}}^{\mathcal{A}_2}}{2^k} + \frac{2p(q_{\text{FO}}^{\mathcal{A}_2} + \ell Q_{\text{FN}}^{\mathcal{A}_2})}{2^{2k}} + \frac{(q_{\text{FO}}^{\mathcal{A}_2} + \ell Q_{\text{FN}}^{\mathcal{A}_2})^2}{2^{2k}} + \frac{pq_{\text{FO}}^{\mathcal{A}_2} \cdot \ell Q_{\text{FN}}^{\mathcal{A}_2}}{2^{2k}}.$$

We substitute $Q_{\text{FO}}^{\mathcal{A}_2} = Q_{\text{FO}}^{\mathcal{A}}$, $q_{\text{FO}}^{\mathcal{A}_2} = q_{\text{FO}}^{\mathcal{A}}$ and $Q_{\text{FN}}^{\mathcal{A}_2} = Q_{\text{SIGN}}^{\mathcal{A}}$. By Lemma 6 we obtain

$$\begin{aligned} \mathbf{Adv}_{\text{DS}^*, \mathbf{S}_3}^{\text{uf}}(\mathcal{A}_3) \leq & \mathbf{Adv}_{\text{DS}^*, \mathbf{R}}^{\text{uf}}(\mathcal{B}) + \frac{2p(Q_{\text{FO}}^{\mathcal{A}_3} + \ell Q_{\text{SIGN}}^{\mathcal{A}_3})}{2^{2k}} \\ & + \frac{(Q_{\text{FO}}^{\mathcal{A}_3} + \ell Q_{\text{SIGN}}^{\mathcal{A}_3})^2}{2^{2k}} + \frac{pQ_{\text{FO}}^{\mathcal{A}_3} \cdot \ell Q_{\text{SIGN}}^{\mathcal{A}_3}}{2^{2k}}. \end{aligned}$$

Recall that adversary \mathcal{A}_3 has the same query complexity as \mathcal{A} .

Under the assumption $p > 2^{k-5}$ made in the theorem, BCJZ [16] established that $|\text{Img}(\text{CF})| = 2^{k-5}$. So $|\text{Img}(\text{CF})|/|\mathbb{Z}_p| = 2^{k-5}/p = \delta$. So by Theorem 2 we have

$$\mathbf{Adv}_{\text{DS}^*, \mathbf{R}}^{\text{uf}}(\mathcal{B}) \leq (1/\delta) \cdot \mathbf{Adv}_{\text{DS}, \mathbf{R}}^{\text{uf}}(\mathcal{B}). \quad (8)$$

By substituting with the number of queries made by \mathcal{A} as in Theorem 1 and collecting terms, we obtain the claimed bound stated in Theorem 7. \square

We can now obtain security of EdDSA under number-theoretic assumptions via known results on the security of Schnorr. Namely, we use the known results to bound $\mathbf{Adv}_{\text{DS}, \mathbf{R}}^{\text{uf}}(\mathcal{B})$ above. From [37, 1] we can get a bound and proof based on the DL problems, and from [39] with a better bound. We can also get an almost tight bound under the MBDL assumption via [5] and a tight bound in the AGM via [22].

6 Indifferentiability of shrink-MD class of functors

INDIFFERENTIABILITY We want the tuple of functions returned by a functor $\mathbf{F} : \text{SS} \rightarrow \text{ES}$ to be able to “replace” a tuple drawn directly from ES . Indifferentiability is a way of defining what this means. We adapt the original MRH definition of indifferentiability [30] to our game-based model in Figure 7. In this

game, \mathcal{S} is a simulator algorithm. The advantage of an adversary \mathcal{A} against the indistinguishability of functor \mathbf{F} with respect to simulator \mathcal{S} is defined to be

$$\mathbf{Adv}_{\mathbf{F}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) := 2 \Pr[\mathbf{G}_{\mathbf{F}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) \Rightarrow 1] - 1.$$

Game $\mathbf{G}_{\mathbf{F}, \mathcal{S}}^{\text{indiff}}$	$\text{PRIV}(i, X)$:
<u>INIT()</u> : <ol style="list-style-type: none"> 1 $c \leftarrow \mathcal{S}\{0, 1\}$ 2 $h \leftarrow \mathcal{S}\mathbf{S}$ 3 $H \leftarrow \mathcal{S}\mathbf{E}\mathbf{S}$ <u>PUB(i, Y)</u> : <ol style="list-style-type: none"> 1 if $c = 0$ then 2 return $\mathcal{S}[H](i, Y)$ 3 else return $h(i, Y)$ 	<ol style="list-style-type: none"> 1 if $c = 0$ then return $H(i, X)$ 2 else return $\mathbf{F}[h](i, X)$ <u>FIN(c')</u> : <ol style="list-style-type: none"> 1 return $[[c = c']]$

Fig. 7. The game $\mathbf{G}_{\mathbf{F}, \mathcal{S}}^{\text{indiff}}$ measuring indistinguishability of a functor \mathbf{F} with respect to simulator \mathcal{S} .

MODIFYING THE MERKLE-DAMGÅRD TRANSFORM Coron et al. showed that the Merkle-Damgård transform is not indistinguishable with respect to any efficient simulator due to its susceptibility to length-extension attacks [19]. In the same work, they analysed the indistinguishability of several closely related indistinguishable constructions, including the “chop-MD” construction. Chop-MD is a functor with the same domain as the MD transform; it simply truncates a specified number of bits from the output of MD. The \mathbf{S}_3 functor of Figure 5 operates similarly to the chop-MD functor, except that \mathbf{S}_3 reduces the output modulo a prime p instead of truncating. This small change introduces some bias into the resulting construction that affects its indistinguishability due to the fact that the outputs of the MD transform, which are $2k$ -bit strings, are not distributed uniformly over \mathbb{Z}_p .

In this section, we establish indistinguishability for a general class of functors that includes both chop-MD and \mathbf{S}_3 . We rely on the indistinguishability of \mathbf{S}_3 in Section 5 as a stepping-stone to the unforgeability of EdDSA; however, we think our proof for chop-MD is of independent interest and improves upon prior work.

The original analysis of the chop-MD construction [19] was set in the ideal cipher model and accounted for some of the structure of the underlying compression function. A later proof by Fischlin and Mittelbach [32] adapts the proof strategy to the simpler construction we address here and works in the random oracle model as we do. Both proofs, however, contain a subtle gap in the way they use their simulators.

At a high level, both proofs define stateful simulators \mathcal{S} which simulate a random compression function by sampling uniform answers to some queries and programming others with the help of their random oracles. These simulators are not perfect, and fail with some probability that the proofs bound. In the ideal indistinguishability game, the PUB oracle answers queries using the simulator and the PRIV oracle answers queries using a random oracle. Both proofs at some point replace the random oracle H in PRIV with **Chop-MD** $[\mathcal{S}]$ and claim that because **Chop-MD** $[\mathcal{S}[H]](X)$ will always return $H(X)$ if the simulator does not fail, the adversary cannot detect the change. This argument is not quite true, because the additional queries to \mathcal{S} made by the PRIV oracle can affect its internal state and prevent the simulator from failing when it would have in the previous game. In our proof, we avoid this issue with a novel simulator with *two internal states* to enforce separation between PRIV and PUB queries that both run the simulator.

Our result establishes indistinguishability for all members of the **Shrink-MD** class of functors, which includes any functor built by composing of the MD transform with a function $\text{Out} : \{0, 1\}^{2k} \rightarrow S$ that satisfies three conditions, namely that for some $\gamma, \epsilon \geq 0$,

1. For all $y \in S$, we can efficiently sample from the uniform distribution on the preimage set $\{\text{Out}^{-1}(y)\}$. We permit the sampling algorithm to fail with probability at most ϵ , but require that upon failure the algorithm outputs a (not necessarily random) element of $\{\text{Out}^{-1}(y)\}$.
2. For all $y \in S$, it holds that $\gamma \leq |\{\text{Out}^{-1}(y)\}|$.
3. The statistical distance $\delta(D)$ between the distribution

$$D := z \leftarrow \text{Out}^{-1}(y) : y \leftarrow S$$

and the uniform distribution on $\{0, 1\}^{2k}$ is bounded above by ϵ .

In principle, we wish γ to be large and ϵ to be small; if this is so, then the set S will be substantially smaller than $\{0, 1\}^{2k}$ and the function Out “shrinks” its domain by mapping it onto a smaller set.

Both chop-MD and mod-MD are members of the **Shrink-MD** class of functors; we briefly show the functions that perform bit truncation and modular reduction by a prime satisfy our three conditions. Truncation by any number of bits trivially satisfies condition (1) with $\epsilon = 0$.

Reduction modulo p also satisfies condition (1) because the following algorithm samples from the equivalence class of x modulo p with failure probability at most $\frac{p}{2^{2k}}$. Let ℓ be the smallest integer such that $\ell > \frac{2^{2k}}{p}$. Sample $w \leftarrow [0 \dots \ell - 1]$ and output $w \cdot p + x$, or x if $w \cdot p + x > 2^{2k}$. We say this algorithm “fails” in the latter case, which occurs with probability at most $\frac{1}{\ell} < \frac{p}{2^{2k}}$. In the event the algorithm does not fail, it outputs a uniform element of the equivalence class of x .

Bellare et al. showed that the truncation of n trailing bits satisfies condition (2) for $\gamma = 2^{2k-n}$ and reduction modulo prime p satisfies (2) for $\gamma = \lfloor 2^{2k}/p \rfloor$. It is clear that sampling from the preimages of a random $2k - n$ -bit string under n -bit truncation produces a uniform $2k$ -bit string, so truncation satisfies condition

(3) with $\epsilon = 0$. Also from Bellare et al. [3], we have that the statistical distance between a uniform element of \mathbb{Z}_p and the modular reduction of a uniform $2k$ -bit string is $\epsilon = \frac{p}{2^{2k}}$. The statistical distance of our distribution $z \leftarrow \text{Out}^{-1}(Y)$ for uniform Y over S from the uniform distribution over $\{0, 1\}^{2k}$ is bounded above by the same ϵ ; hence condition (3) holds.

Given a set S and a function $\text{Out} : \{0, 1\}^{2k} \rightarrow S$, we define the functor $\mathbf{F}_{S, \text{Out}}$ as the composition of Out with \mathbf{MD} . In other words, for any $x \in \{0, 1\}^*$ and $h \in \text{AF}(\{0, 1\}^{b+2k}, \{0, 1\}^{2k})$, let $\mathbf{F}_{S, \text{Out}}[h](x) := \text{Out}(\mathbf{MD}[h](x))$.

Theorem 8. *Let k be an integer and S a set of bitstrings. Let $\text{Out} : \{0, 1\}^{2k} \rightarrow S$ be a function satisfying conditions (1), (2), and (3) above with respect to $\gamma, \epsilon > 0$. Let \mathbf{MD} be the Merkle-Damgård functor (c.f. Section 2) $\mathbf{F}_{S, \text{Out}} := \text{Out} \circ \mathbf{MD}$ be the functor described in the prior paragraph. Let pad be the padding function used by \mathbf{MD} , and let unpad be the function that removes padding from its input (i.e., for all $X \in \{0, 1\}^*$, it holds that $\text{unpad}(X \parallel \text{pad}(|X|)) = X$). Assume that unpad returns \perp if its input is incorrectly padded and that unpad is injective on its support. Then there exists a simulator \mathcal{S} such that for any adversary \mathcal{A} making PRIV queries of maximum length $b \cdot (\ell - 1)$ bits then*

$$\text{Adv}_{\mathbf{F}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) \leq 2(Q_{\text{PUB}}^{\mathcal{A}} + \ell Q_{\text{PRIV}}^{\mathcal{A}})\epsilon + \frac{(Q_{\text{PUB}}^{\mathcal{A}} + \ell Q_{\text{PRIV}}^{\mathcal{A}})^2}{2^{2k}} + \frac{Q_{\text{PUB}}^{\mathcal{A}} \cdot \ell Q_{\text{PRIV}}^{\mathcal{A}}}{\gamma}.$$

We prove the theorem in the game-based framework in our full version [6]. Here, we give a brief overview of our proof strategy and its differences from previous indistinguishability proofs for the chop-MD construction [19, 32].

Our simulator, \mathcal{S} , is defined in Figure 8. It is inspired by, but distinct from, that of Mittelbach and Fischlin’s simulator for the chop-MD construction ([32] Figure 17.4.), which in turn adapts the simulator of Coron et al [19] from the ideal cipher model to the random oracle model. These simulators all present the interface of a random compression function h and internally maintain a graph in which each edge represents an input-output pair under the simulated compression function. The intention is that each path through this graph will represent a possible evaluation of $\mathbf{F}_{S, \text{Out}}[h]$. The fundamental difference between our simulator and previous ones is that we maintain two internal graphs instead of one: one graph for all queries, and one graph for public interface queries only. This novel method of using two graphs avoids the gap in prior proofs described above by tracking precisely which parts of the simulator’s state are influenced by private and public interface queries respectively.

In our proof, we transform the ideal indistinguishability game by evaluating our functor \mathbf{F} in each query to the PRIV oracle. Initially, we discard the output of this evaluation and use a separate graph in our simulator so that these additional queries do not influence the PUB oracle. In later games, we bound the probability that the private queries influence the public graph in a way that is detectable by the adversary (such as creating collisions, cycles, or duplicate edges in the public simulator’s graph), and begin using the same graph for both types of query. We also claim that if the graph is free of collisions, cycles, and duplicate edges, then we can respond to PRIV queries with the evaluation of \mathbf{F} without detection. We

then use the statistical closeness of sampling a random preimage of a random element (property (3) of `Out`) to argue that our simulator is honestly behaving as a random oracle except with some small probability. The resulting game is then equivalent to the real indistinguishability game, and the theorem follows by collecting the bounded differences between each pair of adjacent games.

```

Simulator  $\mathcal{S}[\mathbf{H}](Y, \mathcal{G})$  :
1   $(y, m) \leftarrow Y$ 
2  if  $\exists z$  such that  $(y, z, m) \in \mathcal{G}.\text{edges}$ 
3    return  $z$ 
4   $M \leftarrow \mathcal{G}.\text{FindPath}(IV, y)$ 
5  if  $M \neq \perp$  and  $\text{unpad}(M \parallel m) \neq \perp$  then
6    if  $T_h[Y, M] \neq \perp$  then  $z \leftarrow T_h[Y, M]$ 
7    else  $z \leftarrow \text{Out}^{-1}(\mathbf{H}(\text{unpad}(M \parallel m)))$ 
8     $T_h[Y, M] \leftarrow z$ 
9  else if  $T_h[Y] \neq \perp$  then  $z \leftarrow T_h[Y]$ 
10 else  $z \leftarrow \{0, 1\}^{2k}$ ;  $T_h[Y] \leftarrow z$ 
11 add  $(y, z, m)$  to  $\mathcal{G}.\text{edges}$ 
12 add  $(y, z, m)$  to  $\mathcal{G}_{\text{all}}.\text{edges}$ 
13 return  $z$ 

```

Fig. 8. Indistinguishability simulator for the proof of Theorem 8.

Acknowledgments

Bellare and Davis are supported in part by NSF grant CNS-2154272. We thank the (anonymous) reviewers of Crypto 2022, Asiacrypt 2022 and CT-RSA 2023 for their valuable comments. We thank Joseph Jaeger for his helpful comments and discussions about the correctness of chop-MD proofs in the literature.

References

1. M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer, Heidelberg, Apr. / May 2002. 3, 4, 6, 15, 16, 21
2. M. Backendal, M. Bellare, J. Sorrell, and J. Sun. The fiat-shamir zoo: Relating the security of different signature variants. In N. Gruschka, editor, *Secure IT Systems - 23rd Nordic Conference, NordSec 2018*, volume 11252 of *Lecture Notes in Computer Science*, pages 154–170. Springer, 2018. 16

3. M. Bellare, D. J. Bernstein, and S. Tessaro. Hash-function based PRFs: AMAC and its multi-user security. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 566–595. Springer, Heidelberg, May 2016. 5, 6, 24
4. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Kobitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 1–15. Springer, Heidelberg, Aug. 1996. 2
5. M. Bellare and W. Dai. The multi-base discrete logarithm problem: Tight reductions and non-rewinding proofs for Schnorr identification and signatures. In K. Bhargavan, E. Oswald, and M. Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 529–552. Springer, Heidelberg, Dec. 2020. 4, 6, 16, 21
6. M. Bellare, H. Davis, and Z. Di. Hardening Signature Schemes via Derive-then-Derandomize: Stronger Security Proofs for EdDSA. Cryptology ePrint Archive, February 2023. <http://eprint.iacr.org>. 8, 24
7. M. Bellare, H. Davis, and F. Günther. Separate your domains: NIST PQC KEMs, oracle cloning and read-only indistinguishability. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 3–32. Springer, Heidelberg, May 2020. 6, 8
8. M. Bellare and A. Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, Heidelberg, Aug. 2002. 4
9. M. Bellare, B. Poettering, and D. Stebila. From identification to signatures, tightly: A framework and generic transforms. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 435–464. Springer, Heidelberg, Dec. 2016. 3, 6
10. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 2, 8, 10
11. M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 7, 12
12. M. Bellare and B. Tackmann. Nonce-based cryptography: Retaining security when randomness fails. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 729–757. Springer, Heidelberg, May 2016. 3, 6
13. D. J. Bernstein. Multi-user Schnorr security, revisited. Cryptology ePrint Archive, Report 2015/996, 2015. <https://eprint.iacr.org/2015/996>. 16
14. D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. *Journal of cryptographic engineering*, 2(2):77–89, 2012. 1, 3, 5, 16
15. D. Bleichenbacher. A forgery attack on RSA signatures based on implementation errors in the verification. Rump Session Presentation, Crypto 2006, August 2006. 1
16. J. Brendel, C. Cremers, D. Jackson, and M. Zhao. The provable security of Ed25519: Theory and practice. In *2021 IEEE Symposium on Security and Privacy*, pages 1659–1676. IEEE Computer Society Press, May 2021. 2, 3, 4, 5, 6, 7, 16, 17, 21

17. K. Chalkias, F. Garillot, and V. Nikolaenko. Taming the many eddsas. In T. van der Merwe, C. Mitchell, and M. Mehrnezhad, editors, *Security Standardisation Research*, pages 67–90, Cham, 2020. Springer International Publishing. 6
18. Y. Chen, A. Lombardi, F. Ma, and W. Quach. Does fiat-shamir require a cryptographic hash function? In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 334–363, Virtual Event, Aug. 2021. Springer, Heidelberg. 6
19. J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, Aug. 2005. 2, 5, 22, 24
20. I. Damgård. A design principle for hash functions. In G. Brassard, editor, *CRYPTO’89*, volume 435 of *LNCS*, pages 416–427. Springer, Heidelberg, Aug. 1990. 2, 5, 8
21. Y. Dodis, T. Ristenpart, and T. Shrimpton. Salvaging Merkle-Damgård for practical applications. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 371–388. Springer, Heidelberg, Apr. 2009. 2, 3, 4
22. G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020. 4, 16, 21
23. O. Goldreich. Two remarks concerning the Goldwasser-Micali-Rivest signature scheme. In A. M. Odlyzko, editor, *CRYPTO’86*, volume 263 of *LNCS*, pages 104–110. Springer, Heidelberg, Aug. 1987. 3, 6
24. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988. 3, 9
25. N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In T. Kohno, editor, *USENIX Security 2012*, pages 205–220. USENIX Association, Aug. 2012. 1
26. IANIX. Things that use Ed25519. <https://ianix.com/pub/ed25519-deployment.html>. 1, 3
27. S. Josefsson and I. Liusvaara. Edwards-curve digital signature algorithm (EdDSA). RFC 8032, Jan. 2017. <https://datatracker.ietf.org/doc/html/rfc8032>. 1, 3
28. E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, Aug. 2016. 6
29. A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter. Ron was wrong, whit is right. Cryptology ePrint Archive, Report 2012/064, 2012. <https://eprint.iacr.org/2012/064>. 1
30. U. M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, Feb. 2004. 2, 5, 19, 20, 21
31. R. C. Merkle. A certified digital signature. In G. Brassard, editor, *CRYPTO’89*, volume 435 of *LNCS*, pages 218–238. Springer, Heidelberg, Aug. 1990. 2, 5, 8
32. A. Mittelbach and M. Fischlin. *The Theory of Hash Functions and Random Oracles*. Springer Nature, Switzerland, 2021. 5, 22, 24
33. D. M’Raïhi, D. Naccache, D. Pointcheval, and S. Vaudenay. Computational alternatives to random number generators. In S. E. Tavares and H. Meijer, editors, *SAC 1998*, volume 1556 of *LNCS*, pages 72–80. Springer, Heidelberg, Aug. 1999. 3, 6

34. National Institute of Standards and Technology. Digital Signature Standard (DSS). FIPS PUB 186-5, Oct. 2019. <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5-draft.pdf>. 1, 3
35. G. Neven, N. P. Smart, and B. Warinschi. Hash function requirements for schnorr signatures. *Journal of Mathematical Cryptology*, 3(1):69–87, 2009. 6
36. K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 354–369. Springer, Heidelberg, Aug. 1998. 6
37. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. 3, 4, 6, 16, 21
38. T. Ristenpart, H. Shacham, and T. Shrimpton. Careful with composition: Limitations of the indistinguishability framework. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011. 2
39. L. Rotem and G. Segev. Tighter security for schnorr identification and signatures: A high-moment forking lemma for Σ -protocols. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 222–250, Virtual Event, Aug. 2021. Springer, Heidelberg. 4, 16, 21
40. C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, Jan. 1991. 3, 4, 16
41. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. 6
42. K. Yoneyama, S. Miyagawa, and K. Ohta. Leaky random oracle. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 92(8):1795–1807, 2009. 3, 4