Notes on Threshold EdDSA/Schnorr Signatures

Luís Brandão *

Talk at the NIST Crypto Reading Club

August 10, 2022 @ Maryland, USA

Joint work (NIST IR 8214B Draft) with **Michael Davidson**

Slide-deck in progress. Feedback is welcome.

* Stratívia (At NIST as a Foreign Guest Researcher, Contractor not employed by NIST). Expressed opinions are from the speaker, not to be construed as official NIST views.

Outline

1. Conventional EdDSA/Schnorr
2. Threshold signatures
3. Considerations

Outline

1. Conventional EdDSA/Schnorr

2. Threshold signatures

3. Considerations

Digital signatures — FIPS Pub 186-5 (Draft)

- FIPS: Federal Information Processing Standards Publication
- Digital Signature Standard (DSS)
- 3 families of signature schemes: RSA, ECDSA, EdDSA
- EdDSA is the most recent (based on RFC 8032)

NIST name and address plate (source: nist.gov)

Digital signatures — FIPS Pub 186-5 (Draft)

- FIPS: Federal Information Processing Standards Publication
- Digital Signature Standard (DSS)
- 3 families of signature schemes: RSA, ECDSA, EdDSA
- EdDSA is the most recent (based on RFC 8032)

NIST name and address plate (source: nist.gov)

A signature scheme: (`Keygen`, `Sign`, `Verify`), based on public-key cryptography

Digital signatures — FIPS Pub 186-5 (Draft)

- FIPS: Federal Information Processing Standards Publication
- Digital Signature Standard (DSS)
- 3 families of signature schemes: RSA, ECDSA, EdDSA
- EdDSA is the most recent (based on RFC 8032)

NIST name and address plate (source: nist.gov)

A signature scheme: (`Keygen`, `Sign`, `Verify`), based on public-key cryptography

"*Digital signatures are used to detect unauthorized modifications to data and to authenticate the identity of the signatory.*" ... "*non-repudiation since the signatory cannot easily repudiate the signature at a later time.*"

For later: "unforgeability" and "binding"

Notation for group operations

Multiplicative notation (traditional for finite fields):

Public key $Q = g^s$, where:

- g is a generator of order n; s is the private key in $\mathbb{Z}_n$
- Assumption: infeasibility of computing discrete-logs (base g)

Additive notation (usual with elliptic curves):

Public key $Q = s \cdot G$, where:

- G is a base-point of order n; s is the private key in $\mathbb{Z}_n$
- Assumption: cannot calculate the integer quotient from division with G

Let us proceed with additive notation

EdDSA-style scheme (simplified)

- $\texttt{Keygen}[n]$: { (private key) $s \xleftarrow{\$} \mathbb{Z}_n$; (public key) $Q = s \bullet G$; output (s, Q) }.
- $\texttt{Sign}[s](M)$: $\{r \leftarrow \texttt{GenNonce}(\ldots)$; $R = r \bullet G$; $\chi = H(R, Q, M)$;
  $S = r + \chi \cdot s \pmod n$; output $\sigma = (R, S)\}$.
- $\texttt{Verify}[Q](M, \sigma)$: $\{\chi' = H(R, Q, M)$; output $\texttt{accept}$ iff $S \bullet G \stackrel{?}{=} R \boldsymbol{+} \chi' \bullet Q\}$

Legend: χ (challenge); G (base point, i.e., generator of $\mathbb{G}$); $\texttt{GenNonce}(\ldots)$ (procedure used to **gen**erate the secret nonce); M (**m**essage being signed); n (order of the group generated by G); Q (public key); r (secret nonce); R (nonce commitment; first component of the signature); s (private signing key; in the detailed scheme it is obtained as a digest — hdigest1 — of a precursor private key d); S (second component of the signature); σ (signature); $\xleftarrow{\$}$ (random sampling); $+$, $\cdot$ (integer sum and multiplication); $\boldsymbol{+}$, $\bullet$ (sum and multiplication-by-constant in additive group $\mathbb{G}$).

Schnorr-style [Sch90; BDLSY11]: simple, efficient, some variations (but rationale is similar)

The EdDSA signature formula $\sigma = (R, S)$

Nonce "commitment" $R = r \bullet G$

Hash function

Secret nonce $r = H(v, M)$

Public verification key $Q = s \bullet G$

EdDSA signature $\longrightarrow$ $\boldsymbol{\sigma} = \left(r \bullet G \,,\, r + H\left(R, Q, M\right) \cdot s \right)$ $\longleftarrow$ Private signing key

Base point (**g**enerator of order n)

Message being signed

"Challenge" $\chi = H(R, Q, M)$

S (2nd component of the signature)

Note: The HashEdDSA mode pre-hashes the message

The EdDSA signature formula $\sigma = (R, S)$

Nonce “commitment” $R = r \bullet G$

Hash function

Public verification key $Q = s \bullet G$

Secret nonce $r = H(v, M)$

EdDSA signature $\longrightarrow \boldsymbol{\sigma} = \left(r \bullet G \,,\, r + H\left(R, Q, M\right) \cdot s \right)$ $\longleftarrow$ Private signing key

Base point (**g**enerator of order n)

Message being signed

“Challenge” $\chi = H(R, Q, M)$

S (2^{nd} component of the signature)

Note: The HashEdDSA mode pre-hashes the message

Verification: $0 \leq S \leq n \quad \wedge \quad S' \cdot G \stackrel{?}{=} R' + \chi' \cdot Q$ (note that $= r \cdot G + \chi \cdot (s \cdot G)$)

The EdDSA signature formula $\sigma = (R, S)$

Nonce "commitment" $R = r \bullet G$

Hash function

Public verification key $Q = s \bullet G$

Secret nonce $r = H(v, M)$

EdDSA signature $\longrightarrow$ $\boldsymbol{\sigma} = \left(r \bullet G \,,\, r + H\left(R, Q, M\right) \cdot s \right)$ Private signing key

Base point (**g**enerator of order n)

Message being signed

"Challenge" $\chi = H(R, Q, M)$

S (2^{nd} component of the signature)

Note: The HashEdDSA mode pre-hashes the message

Verification: $0 \leq S \leq n \quad \wedge \quad S' \cdot G =^? R' + \chi' \cdot Q \quad$ (note that $= r \cdot G + \chi \cdot (s \cdot G)$)

Where $S' = 2^c \cdot S$, $R' = 2^c \cdot R$, $\chi' = 2^c \cdot \chi$ (a.k.a. cofactored verification)

Unforgeability

Unforgeability (UF): Malicious client cannot win the following game:

- Client (with access to signing oracle) gets q message–signature pairs (M_i, σ_i)
- Client (without oracle) produces a valid sig σ^* for a new message M^*

EUF-CMA: **e**xistential **u**nf**o**rgeability against **c**hosen **m**essage **a**ttack [GMR88]

Strong UF (SUF): cannot find new pair (σ^*, M^*) (even if msg was already signed) [CD95]

Unforgeability

Unforgeability (UF): Malicious client cannot win the following game:

- Client (with access to signing oracle) gets q message–signature pairs (M_i, σ_i)
- Client (without oracle) produces a valid sig σ^* for a new message M^*

EUF-CMA: **e**xistential **u**n**f**orgeability against **c**hosen **m**essage **a**ttack [GMR88]

Strong UF (SUF): cannot find new pair (σ^*, M^*) (even if msg was already signed) [CD95]

Technical note (feel free to ignore):

- A signature is like a ZKP of knowledge of the signing key (e.g., discrete-log).
- Usually provable with rewinding, when interactive (random challenge each time).
- Non-interactive case: random oracle model / forking lemma.

Binding

Now suppose the signer is the malicious party (adv)

Binding (to message): Can adv repudiate having signed a msg M?

- If UF, and bound to public key Q, then it cannot
- Unless it finds a hash collision $\chi = H(R, Q, M) = H(R, Q, M')$

Strong binding (to message/pubkey): What if adv can lie about the public key Q?

- Can it find two pairs (M, Q) and (M^*, Q^*) and a signature σ valid for both?
- It can (details omitted here), if one key is *invalid* (but there's no check for it)

EdDSA would be strong binding (resistant to key-substitution attack):

- if additionally checking $|Q| > 2^c$ [BCJZ21; CGN20]

Nonce implementation issues

Nonce reuse: Suppose the nonce r is reused when EdDSA-signing different messages.

- $\sigma = (R, S)$, where $S = r + \chi \cdot s$ and $\chi = H(\ldots M)$
- $\sigma^* = (R, S^*)$, where $S^* = r + \chi^* \cdot s$ and $\chi^* = H(\ldots M')$

Then the private key s follows from solving a pair of linear equations with two unknowns

- $S^* - S = (\chi^* - \chi) \cdot s \pmod{n}$,
- $s = (S^* - S) \cdot (\chi^* - \chi)^{-1} \pmod{n}$

It gets worst:

- Even a small nonce-bias (partial knowledge) allows key recovery
- Nonce reuse/bias is also catastrophic for ECDSA

Comparing types of nonce generation

EdDSA specifies pseudorandom nonce generation $r = H(\nu, M)$, which:

- avoids nonce-bias, but is more susceptible to some side-channel attacks

If recovering ν, then from a message-signature pair can compute the signing key s:

- $s = \chi^{-1} \cdot (S - r) \pmod{n}$, where $r = H(\nu, M)$

Comparing types of nonce generation

EdDSA specifies pseudorandom nonce generation $r = H(\nu, M)$, which:

- avoids nonce-bias, but is more susceptible to some side-channel attacks

If recovering ν, then from a message-signature pair can compute the signing key s:

- $s = \chi^{-1} \cdot (S - r) \pmod{n}$, where $r = H(\nu, M)$

Nonce generation type	Bias attacks	Side-channel and fault injection attacks
Deterministic: Pseudorandom, based on a secret key	Safe	More vulnerable
Purely random: Entropy independent of secret key	Vulnerable	Less vulnerable
Combined use: Randomness and pseudo-randomness	Safe	Less vulnerable

On non-verifiable determinism

Signature scheme	Is the signature algorithm deterministic?	Is the output signature verifiably deterministic?
RSASSA-PKCS	**Yes**	**Yes**
EdDSA	**Yes**	**No**
Deterministic ECDSA	**Yes**	**No**
RSA-PSS	No	No
(Probabilistic) ECDSA	No	No

Summary of conventional setting

- Schnorr-style signatures are well-known and been around for a while
- EdDSA Unforgeable?: **SUF** (the verification details matter)
- EdDSA Binding?: (the verification details matter)
  - if assumed pub-key bound $\Rightarrow$ message binding
  - Otherwise no (missing check)
- EdDSA Deterministic?: non-verifiably
- Nonce implementation issues?:
  - Pseudorandom EdDSA: no bias, some susceptibility to side-channel attack
  - Purely random variant: inadvertent bias is catastrophic
  - Hybrid variant: best of both worlds

Outline

1. Conventional EdDSA/Schnorr
2. Threshold signatures
3. Considerations

Threshold approach — intuition

A linear secret-sharing of x is denoted as $[x] = \langle x_1, x_2, ..., x_n \rangle$. $x = \mathsf{Reconst}([x])$

(Simplification: Lagrange coefficients were omitted above. The above actually holds for Additive SS.)

Threshold approach — intuition

A linear secret-sharing of x is denoted as $[x] = \langle x_1, x_2, ..., x_n \rangle$. $x =$ Reconst($[x]$)

The threshold signing follows trivially once having:

- Linear secret-sharing $[s]$ of the private signing key s.
- Linear secret-sharing $[r]$ of a random secret nonce r.

Phase	**Conventional**	**Semi-honest threshold baseline**
Key-Gen	$Q = s \cdot G$	$[Q] = [s] \cdot G$;
Commit nonce	$R = r \cdot G$	$[R] = [r] \cdot G$; then $R =$ Reconst($[R]$)
Compute challenge	$\chi = H(R, Q, M)$	*Same as in conventional*
Produce signature	$S = r + \chi \cdot s \pmod n$	$[S] = [r] + \chi \cdot [s] \pmod n$; then $S =$ Reconst($[S]$)
Verify signature	$S \cdot G \stackrel{?}{=} R + \chi \cdot Q$	*Same as in conventional*

(Simplification: Lagrange coefficients were omitted above. The above actually holds for Additive SS.)

Distributed key-generation (DKG)

Intuition: DKG with verifiable secret sharing [GJKR99]

Verifiable SS of some x: besides each private share x_j for party j, everyone sees "commitments" $X_i = x_i \cdot G$ of everyone's shares, i.e., $[x] \cdot G$

Approach (with a caveat):

- Each party P_i picks a random value x_i and secret-shares it with everyone ($[x_i]$)
- Each party decides their final share as the sum of all received shares
- Each party verifies everything (using the VSS verifiability)

More technicalities needed:

- Prevent anyone from manipulating (bias) the final public key Q
- Ensure termination (prevent bias by abort)

Threshold Schnorr signing using a DKG-based approach

DKG = distributed key-generation. Used by [SS01] for threshold Schnorr.

- **Phase 0:** The keygen phase has verifiably secret-shared a signing key s.
  - And everyone learns $[Q] = [s] \cdot G$, which determines Q.
- **Phase 1:** Use DKG to get a random nonce verifiable secret-sharing $[r]$
  - And everyone learns $[R] = [r] \cdot G$
- **Phase 2:** Signature-shares and reconstruction:
  - Each party communicates their signature share: $S_i = r_i + \chi \cdot s_i$
  - Someone combines the shares $\sigma = (\text{Recons}([R]), \text{Recons}([S]))$

An attempt at threshold Deterministic

Naive solution:

- Every party P_i uses a deterministic nonce contribution $r_i = H(\nu, M)$.
- Final nonce commitment is $R = \text{Reconst}([r] \cdot G)$

Problem:

- Malicious P_j varies their nonce contribution r_j, to affect R and thus $\chi = H(R, Q, M)$

Key recovery pitfall — After just two signings of the same message M:

- Honest signature-share 1st time: $S_i = r_i + \chi \cdot s_i$
- Honest signature-share 2ns time: $S_i^* = r_i + \chi * \cdot s_i$
- Adversary recovers $s_i = (\chi - \chi^*) \cdot (S_i - S_i^*)$

Threshold Deterministic Signatures

- **MPC-based nonce computation**
  - Generic MPC for distributed computation of SHA512-based nonce
  - Distributed hashing using an MPC-friendly hash
- **Local deterministic contributions (per party), ZK-proven correct**
  - PRF based on AES (less ZKP-unfriendly than SHA512)
  - ZKP friendly PRF

Threshold Deterministic Signatures

- **MPC-based nonce computation**
  - Generic MPC for distributed computation of SHA512-based nonce
  - Distributed hashing using an MPC-friendly hash
- **Local deterministic contributions (per party), ZK-proven correct**
  - PRF based on AES (less ZKP-unfriendly than SHA512)
  - ZKP friendly PRF

Reference	Functionally equivalent?	EdDSA Interchangeable?	Fixed public key?	Deterministic?		Some gadgets
				Per subset of signatories	Across resharings	
[BST21, §5]	Yes	Yes	Yes	Yes	Yes	MPC gadgets
[BST21, §6]	No	Yes	Yes	Yes	Yes	MPC-friendly hash
[GKMN21]	No	Yes	Yes	Yes	No	ZKGC, COT
[NRSW20]	No	Yes	No	Yes	N/A	ZKP-friendly PRF

Threshold Probabilistic Signatures

Classical approaches (more rounds): DKG-based

Recent efforts (lower number of rounds):

- ***k*-sum attack** [DEFKLNS19] broke older 2-round protocols (concurrent setting)
  - Malicious P_i in execution k is last to contribute R_i^k, affecting R^k and $\chi^k = H(R^k, Q, M^k)$ to achieve $R^* = \sum_k R^k$ such that $\chi^* = \sum_k \chi^k$ (k-sum problem)
- **2 rounds game-based UF:** prevent k-sum by using multiple nonce-contributions and nonce-binding to message [KG21; NRS21; AB21; CKM21]
- **3 rounds simulatable:** directly prevents manipulation of nonce-commitment R (with extra commitment round) [Lin22]

Threshold comparison (informal)

Signature mode	Nonce generation	Attack of Concern	Informal assessment	
			Conventional	Threshold
Deterministic	Pseudorandom	Bias	Safe	Safe
		Side channel	More vulnerable	**Safer**
Probabilistic	Randomized	Bias	Vulnerable	**Safer**
		Side channel	Less vulnerable	**Safer**
	Hybrid	Bias	Safe	Safe
		Side channel	Less vulnerable	**Safer**

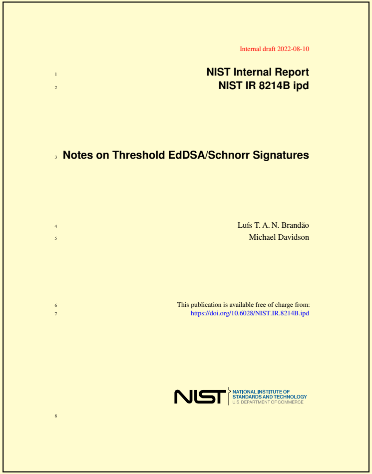
(Other aspects to consider: efficiency, assumptions, threshold parameters, ...)

Outline

1. Conventional EdDSA/Schnorr
2. Threshold signatures
3. Considerations

Draft IR 8214B

- Analyzes the properties of conventional EdDSA
- Distinguishes various approaches for threshold interchangeable schemes w.r.t. EdDSA verification. Compares probabilistic vs. deterministic.
- Identifies aspects that would benefit from more attention (security formulation, WBBR parties, interfaces, adaptive corruptions, ...)
- Some considerations are generic to other schemes
- We expect to receive technical feedback


Internal draft 2022-08-10

NIST Internal Report
NIST IR 8214B ipd

Notes on Threshold EdDSA/Schnorr Signatures

Luís T. A. N. Brandão
Michael Davidson

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8214B.ipd

Developments

An attack and various followup threshold protocols have appeared in the past few years

What would be good to learn with the community:

- Detailed security formulations, technical descriptions, reference implementations
- More emphasis on SUF (some works have only looked at UF)
- More explicit addressing of well behaved parties with bad randomness (WBBR)
- Concerns with manipulation of nonce commitment?
- Actual implementations of broadcast and agreement

Aim: Enable develop recom./guidelines about threshold schemes (not concrete standards)

Thank you for your attention!

Questions?

Notes on Threshold EdDSA/Schnorr Signatures

Upcoming Draft NIST IR 8214B

(This slide-deck is still a work in progress)

References

[AB21] Handan Kılınç Alper and Jeffrey Burdges. "Two-round trip schnorr multi-signatures via delinearized witnesses". In: *Advances in Cryptology — CRYPTO 2021*. Springer. 2021. DOI: 10.1007/978-3-030-84242-0_7. Also at ia.cr/2020/1245 (Cited on p. 28).

[BCJZ21] Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. "The Provable Security of Ed25519: Theory and Practice". In: *Symposium on Security and Privacy (SP)* (2021). DOI: 10.1109/SP40001.2021.00042. Also at ia.cr/2020/823 (Cited on p. 14).

[BDLSY11] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. "High-Speed High-Security Signatures". In: *Cryptographic Hardware and Embedded Systems — CHES 2011*. Springer Berlin Heidelberg, 2011. DOI: 10.1007/978-3-642-23951-9_9. Also at Journal of Cryptographic Engineering, vol. 2, pp. 77–89 (2012), 10.1007/s13389-012-0027-1. Also at ia.cr/2011/368 (Cited on p. 8).

[CD95] Ronald Cramer and Ivan Damgård. "Secure Signature Schemes based on Interactive Protocols". In: *Advances in Cryptology — CRYPTO' 95*. Springer Berlin Heidelberg, 1995. DOI: 10.1007/3-540-44750-4_24. Also at BRICS Report Series, 1(29), 1994, DOI:10.7146/brics.v1i29.21637 (Cited on pp. 12, 13).

[CGN20] Konstantinos Chalkias, François Garillot, and Valeria Nikolaenko. "Taming the Many EdDSAs". In: *International Conference on Security Standardisation Research*. Springer, 2020. DOI: 10.1007/978-3-030-64357-7_4. Also at ia.cr/2020/1244 (Cited on p. 14).

[CKM21] Elizabeth Crites, Chelsea Komlo, and Mary Maller. *How to Prove Schnorr Assuming Schnorr: Security of Multi- and Threshold Signatures*. Cryptology ePrint Archive, Report ia.cr/2021/1375. 2021 (Cited on p. 28).

[DEFKLNS19] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. "On the Security of Two-Round Multi-Signatures". In: *2019 IEEE Symposium on Security and Privacy (SP)* (2019). DOI: 10.1109/SP.2019.00050. Also at ia.cr/2018/417 (Cited on p. 28).

[GJKR99] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems". In: *Advances in Cryptology — EUROCRYPT'99*. Springer-Verlag, 1999. DOI: 10.1007/3-540-48910-X_21. See also J. Cryptology 20, pp. 51–83, 2007, DOI:10.1007/s00145-006-0347-3 (Cited on p. 23).

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks". In: *SIAM Journal on Computing* 17.2 (1988). DOI: 10.1137/0217017 (Cited on pp. 12, 13).

[KG21] Chelsea Komlo and Ian Goldberg. "FROST: Flexible Round-Optimized Schnorr Threshold Signatures". In: (2021). DOI: 10.1007/978-3-030-81652-0_2. Also at ia.cr/2020/852 (Cited on p. 28).

[Lin22] Yehuda Lindell. *Simple Three-Round Multiparty Schnorr Signing with Full Simulatability*. Cryptology ePrint Archive Report ia.cr/2022/374. 2022 (Cited on p. 28).

[NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. "MuSig2: Simple Two-Round Schnorr Multi-signatures". In: *Advances in Cryptology — CRYPTO 2021*. Springer International Publishing, 2021. DOI: 10.1007/978-3-030-84242-0_8. Also at ia.cr/2020/1261 (Cited on p. 28).

[RFC 8032] S. Josefsson and I. Liusvaara. "Edwards-Curve Digital Signature Algorithm (EdDSA)". In: *RFC 8032*. Request for Comments (January 2017). Errata exists. DOI: 10.17487/RFC8032.

[Sch90] C. P. Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Springer New York, 1990. DOI: 10.1007/0-387-34805-0_22. See also J. Cryptology 4, pp. 161–174, 1991, DOI:10.1007/BF00196725 (Cited on p. 8).

[SS01] Douglas R. Stinson and Reto Strobl. "Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates". In: *Information Security and Privacy*. ACISP 2001. Springer Berlin Heidelberg, 2001. DOI: 10.1007/3-540-47719-5_33 (Cited on p. 24).

EdDSA modes (and variants)

Table 6. EdDSA variants

Type	Standard	Mode μ	κ	$b = \lvert d \rvert$	$s \| v$	GenNonce r	Challenge χ
Det.	EdDSA	Ed25519	128	256	$H_0(d)$	$H_0(v \| M)$	$H_0(R \| Q \| M)$
		Ed448	224	456	$H_1(d)$	$H_1(E_{4,0}(ctx) \| v \| M)$	$H_1(E_{4,0}(ctx) \| R \| Q \| M)$
	HashEdDSA	Ed25519ph	128	256	$H_0(d)$	$H_0(E_{2,1}(ctx) \| v \| H_0(M))$	$H_0(E_{2,1}(ctx) \| R \| Q \| H_0(M))$
		Ed448ph	224	456	$H_1(d)$	$H_1(E_{4,1}(ctx) \| v \| H_2(M))$	$H_1(E_{4,1}(ctx) \| R \| Q \| H_2(M))$

Type	Variation	Mode μ	κ	$b = \lvert d \rvert$	$s \| v$	GenNonce r	Challenge χ
Prob.	Random	—	—	—	—	$\xleftarrow{\$} \mathbb{Z}_q$	—
	Hybrid	—	—	—	—	$H(v, rand, f(M))$	—

Legend: See code Some symbols are better contextualized in Fig. 3. Det. (deterministic). Prob. (probabilistic). s, v (first and second halves, respectively, of Hash(d), also denoted as 1st and 2nd digests of d). $E_{i,j}(\ldots)$ (encoding function, defined in FIPS 186 as dom$i(j,\ldots)$, where i is 2 or 4, corresponding to the Ed25519 or Ed448 curves, and j is 1 or 0, corresponding to whether or not it is a "pre-hash" mode). H (some cryptographic hash function or extendable output function); H_0 (SHA-512); H_1 (SHAKE256-length-912); H_2 (SHAKE256-length-512); *rand* (secret randomness or any other secret material). The four deterministic modes (Det.) are based on Draft FIPS 186-5. The two probabilistic variants (Prob.) produce signatures interchangeable w.r.t. EdDSA verification.