Crypto Forum
Internet-Draft
Updates: 6979, 8032 (if approved)
Intended status: Informational
Expires: 4 September 2025

J. Preu Mattsson
E. Thormarker
S. Ruohomaa
Ericsson

3 March 2025

Hedged ECDSA and EdDSA Signatures
draft-irtf-cfrg-det-sigs-with-noise-05

Abstract

Deterministic elliptic-curve signatures such as deterministic ECDSA and EdDSA have gained popularity over randomized ECDSA as their security does not depend on a source of high-quality randomness. Recent research, however, has found that implementations of these signature algorithms may be vulnerable to certain side-channel and fault injection attacks due to their deterministic nature. One countermeasure to such attacks is hedged signatures where the calculation of the per-message secret number includes both fresh randomness and the message. This document updates RFC 6979 and RFC 8032 to recommend hedged constructions in deployments where side-channel attacks and fault injection attacks are a concern. The updates are invisible to the validator of the signature and compatible with existing ECDSA and EdDSA validators.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://cfrg.github.io/draft-irtf-cfrg-det-sigs-with-noise/draft-irtf-cfrg-det-sigs-with-noise.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-irtf-cfrg-det-sigs-with-noise/.

Discussion of this document takes place on the Crypto Forum Research Group mailing list (mailto:cfrg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=cfrg. Subscribe at https://www.ietf.org/mailman/listinfo/cfrg/.

Source for this draft and an issue tracker can be found at https://github.com/cfrg/draft-irtf-cfrg-det-sigs-with-noise.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 4 September 2025.

Copyright Notice

Table of Contents

1. Introduction

In Elliptic-Curve Cryptography (ECC) signature algorithms, the per-message secret number has traditionally been generated from a random number generator (RNG). The security of such algorithms depends on the cryptographic quality of the random number generation and biases in the randomness may have catastrophic effects such as compromising private keys (see e.g., [Bernstein19]). Repeated per-message secret

numbers have caused several severe security accidents in practice. As stated in [RFC6979], the need for a cryptographically secure source of randomness is also a hindrance to deployment of randomized ECDSA [FIPS-186-5] in architectures where secure random number generation is challenging, in particular, embedded IoT systems and smartcards. [ABFJLM17] does however state that smartcards typically have a high-quality RNG on board, which makes it significantly easier and faster to use the RNG instead of doing a hash computation.

In deterministic ECC signatures schemes such as Deterministic Elliptic Curve Digital Signature Algorithm (ECDSA) [RFC6979][FIPS-186-5] and Edwards-curve Digital Signature Algorithm (EdDSA) [RFC8032], the per-message secret number is instead generated in a fully deterministic way as a function of the message and the private key. Except for key generation, the security of such deterministic signatures does not rely on a source of high-quality randomness. This makes verification of implementations easier. As they are presumed to be safer, deterministic signatures have gained popularity and are referenced and recommended by a large number of recent RFCs [RFC8037] [RFC8080] [RFC8225] [RFC8387] [RFC8410] [RFC8411] [RFC8419] [RFC8420] [RFC8422] [RFC8446] [RFC8463] [RFC8550] [RFC8591] [RFC8608] [RFC8624] [RFC9053].

Side-channel attacks are potential attack vectors for implementations of cryptographic algorithms. Side-Channel attacks can in general be classified along three orthogonal axes: passive vs. active, physical vs. logical, and local vs. remote [SideChannel]. It has been demonstrated how side-channel attacks such as power analysis [BCPST14] and timing attacks [Minerva19] [TPM-Fail19] allow for practical recovery of the private key in some existing implementations of randomized ECDSA. [BSI] summarizes minimum requirements for evaluating side-channel attacks of elliptic curve implementations and writes that deterministic ECDSA and EdDSA requires extra care. The deterministic ECDSA specification [RFC6979] notes that the deterministic generation of per-message secret numbers may be useful to an attacker in some forms of side-channel attacks and as stated in [Minerva19], deterministic signatures like [RFC6979] and [RFC8032] might help an attacker to reduce the noise in the side-channel when the same message it signed multiple times. Recent research [SH16] [BP16] [RP17] [ABFJLM17] [SBBDS17] [PSSLR17] [SB18] [WPB19] [AOTZ19] [FG19] have theoretically and experimentally analyzed the resistance of deterministic ECC signature algorithms against side-channel and fault injection attacks. The conclusions are that deterministic signature algorithms have theoretical weaknesses against certain instances of these types of attacks and that the attacks are practically feasibly in some environments. These types of attacks may be of particular concern for hardware implementations such as embedded IoT devices and smartcards where the

adversary can be assumed to have access to the device to induce faults and measure its side-channels such as timing information, power consumption, electromagnetic leaks, or sound with low signal-to-noise ratio. A good summary of fault attacks in given by [Cao20]. See also the discussions and references in [Comments-186-5].

Fault attacks may also be possible without physical access to the device. RowHammer [RowHammer14] showed how an attacker to induce DRAM bit-flips in memory areas the attacker should not have access to. Plundervolt [Plundervolt19] showed how an attacker with root access can use frequency and voltage scaling interfaces to induce faults that bypass even secure execution technologies. RowHammer can e.g., be used in operating systems with several processes or cloud scenarios with virtualized servers. Protocols like TLS, SSH, and IKEv2 that add a random number to the message to be signed mitigate some types of attacks [PSSLR17].

Government agencies are clearly concerned about these attacks. In [Notice-186-5] and [FIPS-186-5], NIST warns about side-channel and fault injection attacks, but states that deterministic ECDSA may be desirable for devices that lack good randomness. The quantum-resistant ML-DSA [FIPS-204] standardized by NIST uses hedged signing by default. BSI has published [BSI] and researchers from BSI have co-authored two research papers [ABFJLM17] [PSSLR17] on attacks on deterministic signatures. For many industries it is important to be compliant with both RFCs and government publications, alignment between IETF, NIST, and BSI recommendations would be preferable.

Note that deriving per-message secret number deterministically, is also insecure in a multi-party signature setting [RFC9591].

One countermeasure to entropy failures, side-channel attacks, and fault injection attacks recommended by [Langley13] [RP17] [ABFJLM17] [SBBDS17] [PSSLR17] [SB18] [AOTZ19] [FG19] and implemented in [OpenSSL13a] [OpenSSL13b] [XEdDSA] [libSodium] [libHydrogen] is to generate the per-message secret number from a random string, a secret key, and the message. This combines the security benefits of fully randomized per-message secret numbers with the security benefits of fully deterministic secret numbers. Such a hedged construction protects against key compromise due to weak random number generation, but still effectively prevents many side-channel and fault injection attacks that exploit determinism. Hedged constructions require minor changes to the implementation and does not increase the number of elliptic curve point multiplications and is therefore suitable for constrained IoT. Section 3.6 of [RFC6979] describes a variant of deterministic ECDSA that adds non-repeating additional data k’ to the per-message secret number generation. Adding randomness to EdDSA is not compliant with [RFC8032]. [Kampanakis16] describes an

alternative [FIPS-186-5] compliant approach where message specific pseudo-random information is used as an additional input to the random number generation to create per-message secret number. [Bernstein14] states that generation of the per-message secret number from a subset of a random string, a secret key, the message, and a message counter is common in DSA/ECDSA implementations.

This document updates [RFC6979] and [RFC8032] to recommend hedged constructions in deployments where side-channel and fault injection attacks are a concern. The updates are invisible to the validator of the signature. Produced signatures remain fully compatible with unmodified ECDSA and EdDSA verifiers and existing key pairs can continue to be used. As the precise use of random bytes is specified, test vectors can still be produced, see Section 6, and implementations can be tested against them.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Hedged EdDSA

This document updates RFC 8032 (EdDSA) to recommend hedged variants of EdDSA for deployments where side-channel attacks and fault injection attacks are a concern, the variants are called hedged EdDSA. The updates are invisible to the validator of the signature and compatible with existing EdDSA validators.

Update to RFC 8032:

For Ed25519ph, Ed25519ctx, and Ed25519: In deployments where side-channel and fault injection attacks are a concern, the following step is RECOMMENDED instead of step (2) in Section 5.1.6 of [RFC8032]:

2. Compute the digest SHA-512(prefix || Z), where Z is 32 octets of random data. Let prefix' denote the leftmost half of the digest. Compute SHA-512(dom2(F, C) || prefix' || PH(M)), where M is the message to be signed. Interpret the 64-octet digest as a little-endian integer r.

For Ed448ph and Ed448: In deployments where side-channel and fault injection attacks are a concern, the following step is RECOMMENDED instead of step (2) in Section 5.2.6 of [RFC8032]:

2. Compute the digest SHAKE256(prefix || Z, 114), where Z is 57 octets of random data. Let prefix' denote the leftmost half of the digest. Compute SHAKE256(dom4(F, C) || prefix' || PH(M), 114), where M is the message to be signed, F is 1 for Ed448ph, 0 for Ed448, and C is the context to use. Interpret the 114-octet digest as a little-endian integer r.

4. Hedged ECDSA

This document updates RFC 6979 (deterministic ECDSA) to recommend a hedged variant of ECDSA for deployments where side-channel attacks and fault injection attacks are a concern, the variant is called hedged ECDSA. The updates are invisible to the validator of the signature and compatible with existing ECDSA validators.

Update to RFC 6979: In ECDSA deployments where side-channel and fault injection attacks are a concern, the following steps are RECOMMENDED instead of steps (d) and (f) in Section 3.2 of [RFC6979]:

d. Set:

K = HMAC_K(V || 0x00 || Z || 000... || int2octets(x) || 000... || bits2octets(h1))

where '||' denotes concatenation. In other words, we compute HMAC with key K, over the concatenation of the following, in order: the current value of V, a sequence of eight bits of value 0, random data Z (of the same length as int2octets(x)), a sequence of zero bits 000..., the encoding of the (EC)DSA private key x, a sequence of zero bits 000..., and the hashed message (possibly truncated and extended as specified by the bits2octets transform). The non-negative number of zeroes 000... is chosen so that the length of (V || 0x00 || Z || 000...) and (int2octets(x) || 000...) are the smallest possible multiples of the block size of the hash function. The HMAC result is the new value of K. Note that the private key x is in the [1, q-1] range, hence a proper input for int2octets, yielding rlen bits of output, i.e., an integral number of octets (rlen is a multiple of 8).

f. Set:

K = HMAC_K(V || 0x01 || Z || 000... || int2octets(x) || 000... || bits2octets(h1))

Note that the "internal octet" is 0x01 this time. The string (Z || 000... || int2octets(x) || 000.. || bits2octets(h1)), called provided_data in HMAC_DRBG, is the same as in step (d).

The construction in [RFC6979] can be seen as using HMAC_DRBG [SP800-90Ar1] with rejection sampling to generate the ECDSA per-message secret number (see Section 3.3 of [RFC6979]). With the updates in this document, Z can be seen as the combination of entropy_input and nonce (see the text on "extra strong" entropy input in Section 8.6.7 of [SP800-90Ar1]). The concatenation 000... || int2octets(x) || 000... || bits2octets(h1) can be seen as the personalization_string. See Section 3.3 of [RFC6979] for the other parameters.

When ECDSA is used with SHAKE [SHA3] the HMAC construction in Section 3.2 of [RFC6979] MAY be used but it is RECOMMENDED to use the more efficient KMAC construction [KMAC] with output length hlen = 8 ceil(qlen / 8), where qlen is the binary length of the order of the base point of the elliptic curve [RFC6979] and ceil(x) is the ceiling function mapping x to the least integer greater than or equal to x. When ECDSA is used with SHAKE128, it is RECOMMENDED to replace HMAC(K, M) in Section 3.2 of [RFC6979] with KMAC128(K, M, hlen, ""). When ECDSA is used with SHAKE256, it is RECOMMENDED to replace HMAC(K, M) in Section 3.2 of [RFC6979] with KMAC256(K, M, hlen, ""). [RFC8692] and [FIPS-186-5] define the use of SHAKE128 with an output length of 256 bits and SHAKE256 with an output length or 512 bits.

In new deployments, where side-channel and fault injection attacks are a concern, Hedged EdDSA as specified in Section 3 is RECOMMENDED.

5. Security Considerations

The constructions in this document follow the high-level approach in [XEdDSA] to calculate the per-message secret number from the hash of the private key and the message, but add additional randomness into the calculation for greater resilience. This does not re-introduce the strong security requirement of randomness needed by randomized ECDSA [FIPS-186-5]. The randomness of Z need not be perfect but SHALL be generated by a cryptographically secure method and SHALL be secret. Even if the same random number Z is used to sign two different messages, the security will be the same as deterministic ECDSA and EdDSA and an attacker will not be able to compromise the private key with algebraic means as in fully randomized ECDSA [FIPS-186-5]. With the construction specified in this document, two signatures over two equal messages are different which prevents information leakage in use cases where signatures but not messages are public.

The construction in this document aims to mitigate fault injection attacks that leverage determinism in deterministic ECDSA and EdDSA signatures (see e.g., [ABFJLM17]), by randomizing nonce generation. Fault injection attacks that achieve instruction skipping as in e.g.,

Section 3.4 of [ABFJLM17] are not necessarily stopped. It seems to be possible to, at the same time, also mitigate attacks that use first order differential power analysis (DPA) against the hash computation of deterministic nonces in EdDSA and ECDSA (see e.g., [ABFJLM17][SBBDS17]). The Hedged EdDSA construction mitigates the referenced first order DPA attacks by mixing prefix with Z before mixing it with any public variable data (message or context). Similarly, the Hedged ECDSA construction mixes x with a state randomized by Z before mixing it with public variable data (h1). The random bytes Z are re-used in step (d) and (f) of Hedged ECDSA to align with HMAC_DRBG. This may make certain DPA attacks easier than if randomness had been sampled fresh for each respective step. Note however that V is updated between the steps and that the secret key x is processed in a new input block of the hash function after processing V and Z in each respective step.

A key pair MAY be reused between implementations of the hedged constructions in this document and the non-hedged original constructions in [RFC8032] and Section 3.2 of [RFC6979]. The Hedged EdDSA construction in this document randomizes prefix in an intermediate step and preserves the domain separation between the different variants of EdDSA (see Section 8.6 of [RFC8032]). The Hedged ECDSA construction has a different HMAC input length in step (d) (and (f)) of Section 4 than the original construction in Section 3.2 of [RFC6979]. It is therefore impossible for an attacker to manipulate the parameters to the nonce generation process (between the different constructions) such that the HMAC input in step (d) and (f) becomes identical for two distinct messages.

Implementations need to follow best practices on how to protect against all side-channel attacks, not just attacks that exploit determinism, see for example [BSI].

6. Test Vectors

TODO

6.1. Hedged Ed25519

```
      MESSAGE = { }
   SECRET KEY = { }
  RANDOM DATA = { }
    SIGNATURE = { }
```

6.2. Hedged ECDSA with P-256 and SHA-256

```
      MESSAGE = { }
   SECRET KEY = { }
  RANDOM DATA = { }
    SIGNATURE = { }
```

7. References

7.1. Normative References

[FIPS-186-5] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", NIST FIPS PUB 186-5 , February 2023, <https://doi.org/10.6028/NIST.FIPS.186-5>.

[KMAC] National Institute of Standards and Technology (NIST), "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash", NIST SP 800-185 , December 2016, <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8692] Kampanakis, P. and Q. Dang, "Internet X.509 Public Key Infrastructure: Additional Algorithm Identifiers for RSASSA-PSS and ECDSA Using SHAKEs", RFC 8692, DOI 10.17487/RFC8692, December 2019, <https://www.rfc-editor.org/rfc/rfc8692>.

[SHA3] National Institute of Standards and Technology (NIST), "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", NIST FIPS PUB 202 , August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf>.

[SP800-90Ar1] National Institute of Standards and Technology (NIST), "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", NIST SP 800-90A Revision 1 , June 2015, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf>.

7.2. Informative References

[ABFJLM17] Ambrose, C., Bos, J., Fay, B., Joye, M., Lochter, M., and B. Murray, "Differential Attacks on Deterministic Signatures", October 2017, <https://eprint.iacr.org/2017/975>.

[AOTZ19] Aranha, D., Orlandi, C., Takahashi, A., and G. Zaverucha, "Security of Hedged Fiat-Shamir Signatures under Fault Attacks", September 2019, <https://eprint.iacr.org/2019/956>.

[BCPST14] Batina, L., Chmielewski, L., Papachristodoulou, L., Schwabe, P., and M. Tunstall, "Online Template Attacks", December 2014, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.854.7836&rep=rep1&type=pdf>.

[Bernstein14] Bernstein, D., "How to design an elliptic-curve signature system", March 2014, <https://blog.cr.yp.to/20140323-ecdsa.html>.

[Bernstein19] Bernstein, D., "Why EdDSA held up better than ECDSA against Minerva", October 2019, <https://blog.cr.yp.to/20191024-eddsa.html>.

[BP16] Barenghi, A. and G. Pelosi, "A Note on Fault Attacks Against Deterministic Signature Schemes (Short Paper)", September 2016, <https://link.springer.com/chapter/10.1007/978-3-319-44524-3_11>.

[BSI] Bundesamt fr Sicherheit in der Informationstechnik, "Minimum Requirements for Evaluating Side-Channel Attack Resistance of Elliptic Curve Implementations", November 2016, <https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_ECCGuide_e_pdf.pdf?__blob=publicationFile>.

[Cao20] Weiqiong Cao, Hongsong Shi, Hua Chen, Jiazhe Chen, Limin Fan, and Wenling Wu, "Lattice-based Fault Attacks on Deterministic Signature Schemes of ECDSA and EdDSA", June 2020, <https://eprint.iacr.org/2020/803>.

[Comments-186-5] "Public Comments Received on Draft FIPS 186-5: Digital Signature Standards (DSS)", March 2021, <https://csrc.nist.gov/CSRC/media/Publications/fips/186/5/draft/documents/fips-186-5-draft-comments-received.pdf>.

[FG19] Fischlin, M. and F. Gnther, "Modeling Memory Faults in Signature and Encryption Schemes", September 2019, <https://eprint.iacr.org/2019/1053>.

[FIPS-204] National Institute of Standards and Technology (NIST), "Module-Lattice-Based Digital Signature Standard", FIPS 204 , August 2023, <https://csrc.nist.gov/pubs/fips/204/ipd>.

[Kampanakis16] Kampanakis, P., "FIPS and Deterministic ECDSA: Achieving robust security and conformance", December 2016, <https://blogs.cisco.com/security/fips-and-deterministic-ecdsa-achieving-robust-security-and-conformance>.

[Langley13] Langley, A., "Sudden Death Entropy Failures", June 2013, <https://www.imperialviolet.org/2013/06/15/suddendeathentropy.html>.

[libHydrogen] "The Hydrogen library", n.d., <https://github.com/jedisct1/libhydrogen>.

[libSodium] "The Sodium library", n.d., <https://github.com/jedisct1/libsodium>.

[Minerva19]
Centre for Research on Cryptography and Security (CRoCS), "Minerva", October 2019, <https://minerva.crocs.fi.muni.cz/>.

[Notice-186-5]
National Institute of Standards and Technology (NIST), "Request for Comments on FIPS 186-5 and SP 800-186", October 2019, <https://www.federalregister.gov/documents/2019/10/31/2019-23742/request-for-comments-on-fips-186-5-and-sp-800-186>.

[OpenSSL13a]
"Add secure DSA nonce flag", n.d., <https://github.com/openssl/openssl/commit/8a99cb29d1f0013243a532bccc1dc70ed678eebe>.

[OpenSSL13b]
"Make ‘safe’ (EC)DSA nonces the default", n.d., <https://github.com/openssl/openssl/commit/190c615d4398cc6c8b61eb7881d7409314529a75>.

[Plundervolt19]
Murdock, K., Oswald, D., Garcia, F., Van Bulck, J., Gruss, D., and F. Piessens, "How a little bit of undervolting can cause a lot of problems", December 2019, <https://plundervolt.com/>.

[PSSLR17] Poddebniak, D., Somorovsky, J., Schinzel, S., Lochter, M., and P. Rsler, "Attacking Deterministic Signature Schemes using Fault Attacks", October 2017, <https://eprint.iacr.org/2017/1014>.

[RFC8037] Liusvaara, I., "CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE)", RFC 8037, DOI 10.17487/RFC8037, January 2017, <https://www.rfc-editor.org/rfc/rfc8037>.

[RFC8080] Sury, O. and R. Edmonds, "Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC", RFC 8080, DOI 10.17487/RFC8080, February 2017, <https://www.rfc-editor.org/rfc/rfc8080>.

[RFC8225] Wendt, C. and J. Peterson, "PASSporT: Personal Assertion Token", RFC 8225, DOI 10.17487/RFC8225, February 2018, <https://www.rfc-editor.org/rfc/rfc8225>.

[RFC8387] Sethi, M., Arkko, J., Keranen, A., and H. Back, "Practical Considerations and Implementation Experiences in Securing Smart Object Networks", RFC 8387, DOI 10.17487/RFC8387, May 2018, <https://www.rfc-editor.org/rfc/rfc8387>.

[RFC8391] Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, DOI 10.17487/RFC8391, May 2018, <https://www.rfc-editor.org/rfc/rfc8391>.

[RFC8410] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure", RFC 8410, DOI 10.17487/RFC8410, August 2018, <https://www.rfc-editor.org/rfc/rfc8410>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/rfc/rfc8411>.

[RFC8419] Housley, R., "Use of Edwards-Curve Digital Signature Algorithm (EdDSA) Signatures in the Cryptographic Message Syntax (CMS)", RFC 8419, DOI 10.17487/RFC8419, August 2018, <https://www.rfc-editor.org/rfc/rfc8419>.

[RFC8420] Nir, Y., "Using the Edwards-Curve Digital Signature Algorithm (EdDSA) in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8420, DOI 10.17487/RFC8420, August 2018, <https://www.rfc-editor.org/rfc/rfc8420>.

[RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <https://www.rfc-editor.org/rfc/rfc8422>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8463] Levine, J., "A New Cryptographic Signature Method for DomainKeys Identified Mail (DKIM)", RFC 8463, DOI 10.17487/RFC8463, September 2018, <https://www.rfc-editor.org/rfc/rfc8463>.

[RFC8550] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Certificate Handling", RFC 8550, DOI 10.17487/RFC8550, April 2019, <https://www.rfc-editor.org/rfc/rfc8550>.

[RFC8554] McGrew, D., Curcio, M., and S. Fluhrer, "Leighton-Micali Hash-Based Signatures", RFC 8554, DOI 10.17487/RFC8554, April 2019, <https://www.rfc-editor.org/rfc/rfc8554>.

[RFC8591] Campbell, B. and R. Housley, "SIP-Based Messaging with S/MIME", RFC 8591, DOI 10.17487/RFC8591, April 2019, <https://www.rfc-editor.org/rfc/rfc8591>.

[RFC8608] Turner, S. and O. Borchert, "BGPsec Algorithms, Key Formats, and Signature Formats", RFC 8608, DOI 10.17487/RFC8608, June 2019, <https://www.rfc-editor.org/rfc/rfc8608>.

[RFC8624] Wouters, P. and O. Sury, "Algorithm Implementation Requirements and Usage Guidance for DNSSEC", RFC 8624, DOI 10.17487/RFC8624, June 2019, <https://www.rfc-editor.org/rfc/rfc8624>.

[RFC8937] Cremers, C., Garratt, L., Smyshlyaev, S., Sullivan, N., and C. Wood, "Randomness Improvements for Security Protocols", RFC 8937, DOI 10.17487/RFC8937, October 2020, <https://www.rfc-editor.org/rfc/rfc8937>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/rfc/rfc9053>.

[RFC9591] Connolly, D., Komlo, C., Goldberg, I., and C. A. Wood, "The Flexible Round-Optimized Schnorr Threshold (FROST) Protocol for Two-Round Schnorr Signatures", RFC 9591, DOI 10.17487/RFC9591, June 2024, <https://www.rfc-editor.org/rfc/rfc9591>.

[RowHammer14] Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J., Lee, D., Wilkerson, C., and K. Mutlu, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors", June 2014, <https://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf>.

[RP17] Romailler, Y. and S. Pelissier, "Practical fault attack against the Ed25519 and EdDSA signature schemes", September 2017, <https://romailler.ch/ddl/10.1109_FDTC.2017.12_eddsa.pdf>.

[SB18] Samwel, N. and L. Batina, "Practical Fault Injection on Deterministic Signatures: The Case of EdDSA", April 2018, <https://nielssamwel.nl/papers/africacrypt2018_fault.pdf>.

[SBBDS17] Samwel, N., Batina, L., Bertoni, G., Daemen, J., and R. Susella, "Breaking Ed25519 in WolfSSL", October 2017, <https://eprint.iacr.org/2017/985.pdf>.

[SH16] Seuschek, H., Heyszl, J., and F. De Santis, "A Cautionary Note: Side-Channel Leakage Implications of Deterministic Signature Schemes", January 2016, <http://www.cs2.deib.polimi.it/slides_16/01_Seuschek_Deterministic_Signatures.pdf>.

[SideChannel] Spreitzer, R., Moonsamy, V., Korak, T., and S. Mangard, "Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices", December 2017, <https://arxiv.org/pdf/1611.03748.pdf>.

[TPM-Fail19] Moghimi, D., Sunar, B., Eisenbarth, T., and N. Heninge, "TPM-FAIL: TPM meets Timing and Lattice Attacks", October 2019, <https://tpm.fail/>.

[WPB19] Weissbart, L., Picek, S., and L. Batina, "One trace is all it takes: Machine Learning-based Side-channel Attack on EdDSA", July 2019, <https://eprint.iacr.org/2019/358.pdf>.

[XEdDSA] Signal, "The XEdDSA and VXEdDSA Signature Schemes", October 2016, <https://signal.org/docs/specifications/xeddsa/>.

Change log

This section is to be removed before publishing as an RFC.

Changes from -03 to -04:

* Resubmission

Changes from -02 to -03:

* Same randomness Z in step d and f to align with HMAC_DRBG.

* Changed Hedged EdDSA order to 0x00 || Z || dom2(F, C) instead of dom2(F, C) || Z. This avoids collisions with RFC 8032 and aligns with Bernstein's recommendation to put Z before the context.

* Changed KMAC output length recommendations to avoid multiple invocations.

* Updates some text to align with the hedged signatures/signing terminology.

* Added more description about the construction.

* Editorial changes.

Changes from -01 to -02:

* Different names Zd and Zf for the randomness in ECDSA.

* Added empty test vector section as TODO.

Changes from -00 to -01:

* Changed terminology to hedged signatures/signing.

* Added reference to the FIPS 204 (ML-DSA) where hedged signatures are the default.

* Added a second 000... padding that separates the context from the prefix, aligning with BSI recommendations.

* Added note that Z in step f is not reused from step d.

* Added note on "internal octet" is 0x01 from RFC 6979.

* Removed incorrect statement that context fit in first block.

* Added more description about the construction.

* Moved "For discussion" section to GitHub issue.

* Editorial changes.

Acknowledgments

The authors would like to thank Tony Arcieri, Uri Blumenthal, Carsten Bormann, Taylor R Campbell, Quynh Dang, Hkan Englund, Janos Follath, Phillip Hallam-Baker, Chelsea Komlo, Niklas Lindskog, Ilari Liusvaara, Danny Niu, Jim Schaad, Ruggero Susella, Daniel J. Bernstein, Filippo Valsorda, and Snke Jendral for their valuable comments and feedback.

Authors' Addresses

John Preu Mattsson
Ericsson
Email: john.mattsson@ericsson.com

Erik Thormarker
Ericsson
Email: erik.thormarker@ericsson.com

Sini Ruohomaa
Ericsson
Email: sini.ruohomaa@ericsson.com