Network Working Group
Internet-Draft
Updates: 6979, 8032 (if approved)
Intended status: Informational
Expires: September 12, 2020

J. Preuss Mattsson
E. Thormarker
S. Ruohomaa
Ericsson

March 11, 2020

Deterministic ECDSA and EdDSA Signatures with Additional Randomness
draft-mattsson-cfrg-det-sigs-with-noise-02

Abstract

Deterministic elliptic-curve signatures such as deterministic ECDSA and EdDSA have gained popularity over randomized ECDSA as their security do not depend on a source of high-quality randomness. Recent research has however found that implementations of these signature algorithms may be vulnerable to certain side-channel and fault injection attacks due to their determinism. One countermeasure to such attacks is to re-add randomness to the otherwise deterministic calculation of the per-message secret number. This document updates RFC 6979 and RFC 8032 to recommend constructions with additional randomness for deployments where side-channel attacks and fault injection attacks are a concern.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2020.

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. 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-186-4] in architectures where secure random number generation is challenging, in particular, embedded IoT systems and smartcards. [ABFJLM17] does however state that smartcards typically has 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] 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 depend on a source of high-quality randomness. 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] [RFC8152] [RFC8225] [RFC8387] [RFC8410] [RFC8411] [RFC8419]

[RFC8420] [RFC8422] [RFC8446] [RFC8463] [RFC8550] [RFC8591] [RFC8624] [RFC8208] [RFC8608].

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 with low signal-to-noise ratio, power consumption, electromagnetic leaks, or sound. 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 and Plundervolt [Plundervolt19] showed how an attacker with root access can use frequency and voltage scaling interfaces to induce faults that bypasses 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 adds 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 [Draft-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. 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.

One countermeasure to side-channel and fault injection attacks recommended by [RP17] [ABFJLM17] [SBBDS17] [PSSLR17] [SB18] [AOTZ19] [FG19] and implemented in [XEdDSA] [libSodium] [libHydrogen] is to re-introduce some additional randomness to the otherwise deterministic generation of the per-message secret number. This combines the security benefits of fully-randomized per-message secret numbers with the security benefits of fully-deterministic secret numbers. Such a 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. Such a constuction require minor changes to the implementation and does not increase the number of elliptic curve point multiplications and is therefore suitable for constained IoT. Deterministic ECDSA with additional randomness can be made compliant with [FIPS-186-4] but would not be compliant with the recommendations in many RFCs. Adding randomness to EdDSA is not compliant with [RFC8032].

This document updates [RFC6979] and [RFC8032] to recommend constructions with additional randomness for deployments where side-channel and fault injection attacks are a concern. 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 the noise is specified, test vectors can still be produced and implementations can be tested against them.

2. Updates to RFC 8032 (EdDSA)

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 SHA-512(dom2(F, C) || Z || prefix || 000... || PH(M)), where M is the message to be signed, Z is 32 octets of random data, the number of zeroes 000... is chosen so that the length of (dom2(F, C) || Z || prefix || 000...) is 1024 bytes. 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 5.3.6 of [RFC8032]:

2. Compute SHAKE256(dom4(F, C) || Z || prefix || 000... || PH(M), 114), where M is the message to be signed, and Z is 57 octets of random data, the number of zeroes 000... is chosen so that the length of (dom4(F, C) || Z || prefix || 000...) is 1088 bytes. 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.

3. Updates to RFC 6979 (Deterministic ECDSA)

For Deterministic ECDSA: In existing 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 || 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)), the encoding of the (EC)DSA private key x, a sequence of zero bits 000... chosen so that the length of (V || 0x00 || Z || int2octets(x) || 000...) is equal to the block size of the hash function, and the hashed message (possibly truncated and extended as specified by the bits2octets transform). 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 || int2octets(x) || 000... || bits2octets(h1))

In new deployments, where side-channel and fault injection attacks are a concern, EdDSA with additional randomness as specified in Section 2 is RECOMMENDED.

4. Security Considerations

The constructions in this document follows 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-186-4]. The randomness of Z does not need to be perfect, but SHALL be generated by a cryptographically secure pseudo random number generator (PRNG) 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-186-4]. 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 place the additional randomness before the message to align with randomized hashing methods.

[SBBDS17] states that [XEdDSA] would not prevent their attack due to insufficient mixing of the hashed private key with the additional randomness. [SBBDS17] suggest a construction where the randomness is padded with zeroes so that the first 1024-bit SHA-512 block is composed only of the hashed private key and the random value, but not the message. The construction in this document follows this recommendation and pads with zeroes so that the first block is composed only of the hashed private key and the random value, but not the message.

Another countermeasure to fault attacks is to force the signer to verify the signature in the last step of the signature generation or to calculate the signature twice and compare the results. These countermeasure would catch a single fault but would not protect against attackers that are able to precisely inject faults several times [RP17] [PSSLR17] [SB18]. Adding an additional sign or verification operation would also significantly affect performance, especially verification which is a heavier operation than signing in ECDSA and EdDSA.

[ABFJLM17] suggests using both additional randomness and a counter, which makes the signature generation stateful. While most used signatures have traditionally been stateless, stateful signatures like XMSS [RFC8391] and LMS [RFC8554] have now been standardized and deployed. [I-D.irtf-cfrg-randomness-improvements] specifies a PRNG construction with a random seed, a secret key, a context string, and a nonce, which makes the random number generation stateful. The generation of the per-message secret number in this document is not stateful, but it can be used with a stateful PRNG. The exact construction in [I-D.irtf-cfrg-randomness-improvements] is however not recommended in deployments where side-channel and fault injection attacks are a concern as it relies on deterministic signatures.

With the construction in this document, the repetition of the same per-message secret number for two different messages is highly

unlikely even with an imperfect random number generator, but not impossible. As an extreme countermeasure, previously used secret numbers can be tracked to ensure their uniqueness for a given key, and a different random number can be used if a collision is detected. This document does not mandate nor stop an implementation from taking such a precaution.

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

5. For discussion (to be removed in the future)

- Amount of randomness - The current construction uses random data of the same length as 'prefix' or 'int2octets(x)' which means 32 bytes of randomness for Ed25519. XEdDSA uses 64 bytes of randomness which might be overkill. As discussed in [SBBDS17], the amount of randomness needed depends on the targeted security level. 32 bytes of randomness should be enough for Ed448 and 16 bytes of randomness should be enough for Ed25519. Even less than that is likely sufficient to prevent practical attacks.

- Deterministic ECDSA with SHAKE - NIST is planning to approve SHAKE128(M,128) and SHAKE256(M,256) for use in ECDSA [Draft-186-5]. Deterministic ECDSA as specified in [RFC6979] would then use HMAC-SHAKE instead of a more optimal KMAC, which would be the prefered keyed hash function for use with SHAKE. It should be discussed if IETF (or NIST) should specify that the resulting HMAC-SHAKE128(K, M) and HMAC-SHAKE256(K, M) in deterministic ECDSA should be replaced with KMAC128(K,M,128) and KMAC256(K,M,128).

6. References

6.1. Normative References

[FIPS-186-4]
Department of Commerce, National., "Digital Signature Standard (DSS)", NIST FIPS PUB 186-4 , July 2013, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.

[I-D.irtf-cfrg-randomness-improvements]
Cremers, C., Garratt, L., Smyshlyaev, S., Sullivan, N., and C. Wood, "Randomness Improvements for Security Protocols", draft-irtf-cfrg-randomness-improvements-10 (work in progress), February 2020.

[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/info/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/info/rfc8032>.

6.2. 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>.

[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 fuer 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>.

[Draft-186-5] National Institute of Standards and Technology (NIST), ., "FIPS PUB 186-5 (Draft)", October 2019, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5-draft.pdf>.

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

[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>.

[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. Roesler, "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/info/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/info/rfc8080>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8208] Turner, S. and O. Borchert, "BGPsec Algorithms, Key Formats, and Signature Formats", RFC 8208, DOI 10.17487/RFC8208, September 2017, <https://www.rfc-editor.org/info/rfc8208>.

[RFC8225] Wendt, C. and J. Peterson, "PASSporT: Personal Assertion Token", RFC 8225, DOI 10.17487/RFC8225, February 2018, <https://www.rfc-editor.org/info/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/info/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/info/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/info/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/info/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/info/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/info/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/info/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/info/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/info/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/info/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/info/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/info/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/info/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/info/rfc8624>.

[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/>.

Acknowledgments

The authors want to thank Tony Arcieri, Uri Blumenthal, and Quynh Dang for their valuable comments and feedback.

Authors' Addresses

John Preuss Mattsson
Ericsson

Email: john.mattsson@ericsson.com

Erik Thormarker
Ericsson

Email: erik.thormarker@ericsson.com

Sini Ruohomaa
Ericsson

Email: sini.ruohomaa@ericsson.com