lwig R. Struik
Internet-Draft Struik Security Consultancy
Intended status: Informational February 22, 2021
Expires: August 26, 2021

ECDSA Signatures in Verification-Friendly Format
draft-struik-lamps-verification-friendly-ecdsa-00

Abstract

This document specifies how to represent ECDSA signatures so as to facilitate fast verification of single signatures and fast batch verification. We illustrate that this technique can be applied retroactively by any device (rather than only by the signer), thereby facilitating transitioning to always generating ECDSA signatures in this way.

Requirements Language

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.

Table of Contents

1. Fostering Fast Verification with ECDSA

ECDSA is one of the most widely used elliptic-curve digital signature algorithms. It has been standardized in FIPS Pub 186-4, ANSI X9.62, BSI, SECG, and IETF, and is widely deployed by a plethora of internet protocols specified by the Internet Engineering Task Force (IETF), with industry specifications in the areas of machine-to-machine communication, such as ZigBee, ISA, and Thread, with wireless communication protocols, such as IEEE 802.11, with payment protocols, such as EMV, with vehicle-to-vehicle (V2V) specifications, as well as with electronic travel documents and other specifications developed under a more stringent regulatory oversight regime, such as, e.g., ICAO and PIV. ECDSA is the only elliptic-curve based signature scheme endorsed by regulatory bodies in both the United States and the European Union.

While methods for accelerated verification of ECDSA signatures and for combining this with key computations have been known for over 1 1/2 decade (see, e.g., [SAC2005] and [SAC2010]), these have been commonly described in technical papers in terms of ECDSA*, a slightly modified version of ECDSA, where their use with standardized ECDSA seems less well known. It is the purpose of this document to fill

this seeming void and describe how ECDSA signatures can be easily generated to facilitate more efficient verification, without failing. We emphasize that this does not require changes to standardized specifications of ECDSA, thereby allowing reuse of existing standards and easy integration with existing implementations.

2. Review of ECDSA and ECDSA*

In this section, we summarize the properties of the signature scheme ECDSA and of the modified signature scheme ECDSA* that are relevant for our exposition. The signature schemes are defined in terms of a suitable elliptic curve E, hash function H, and several representation functions, where n is the (prime) order of the base point G of this curve, and where E is an elliptic curve in short-Weierstrass form. For full details, we refer to the relevant standards.

With the ECDSA signature scheme, the signature over a message m provided by a signing entity with static private key d is an ordered pair (r,s) of integers in the interval [1,n-1], where the value r is derived from a so-called ephemeral signing key R:=k*G generated by the signer via a fixed public conversion function and where the value s is a function of the ephemeral private key k, the static private key d, the value r and the value e derived from message m via hash function H and representation hereof in the interval [0,n-1]. (More specifically, one has e=s*k-d*r (mod n), where r is a function of the x-coordinate of R.) A signature (r,s) over message m purportedly signed by an entity with public key Q:=d*G is accepted if Q is indeed a valid public key, if both signature components r and s are integers in the interval [1,n-1] and if the reconstructed value R' derived from the purported signature, message, and public key yields r, via the same fixed conversion function as used during the signing operation. (More specifically, one computes R':=(1/s)*(e*G+r*Q) and checks that r is the same function of the x-coordinate of R'.)

With the ECDSA* signature scheme, one follows the same signing operation, except that one outputs as signature the ordered pair (R,s), rather than the pair (r,s), where R is the ephemeral signing key; one accepts a signature (R,s) over message m purportedly signed by an entity with public key Q by first computing the value r derived from signature component R via the conversion function, checking that both r and s are integers in the interval [1,n-1], computing R':=(1/s)*(e*G+r*Q) and checking whether, indeed, R'=R.

It is known that ECDSA signatures and the corresponding ECDSA* signatures have the same success/failure conditions (i.e., ECDSA and ECDSA* are equally secure): if (r,s) is a valid ECDSA signature for message m purportedly signed by an entity with public key Q, then

(R',s) is a valid corresponding ECDSA* signature, where R':=((1/s)(e*G+r*Q) is a point for which the conversion function yields r. Conversely, if (R,s) is a valid ECDSA* signature for message m purportedly signed by an entity with public key Q, then (r,s) is a valid corresponding ECDSA signature, where r is obtained from R via the conversion function.

It is well-known that if an ECDSA signature (r,s) is valid for a particular message m and public key Q, then so is (r,-s) -- the so-called malleability -- and that, similarly, if an ECDSA* signature (R,s) is valid, hen so is (-R,-s), where the latter relies on the fact that the conversion function only depends on the x-coordinate of R.

3. Signature Verification with ECDSA and ECDSA*

In this section, we more closely scrutinize ECDSA and ECDSA* verification processes.

With ECDSA*, signature verification primarily involves checking an elliptic curve equation, viz. checking whether R = (1/s)*(e*G+r*Q), which lends itself to accelerated signature verification techniques and the ability to use batch verification techniques, with significant potential for accelerated verification (~30% and up). Here, speed-ups are due to the availability of the point R, which effectively allows checking an equation of the form -s*R + (e*G+r*Q)=O instead (where O is the identity element of the curve). Similarly to the case with EdDSA [RFC8032], this offers the potential for batch verification, by checking a randomized linear combination of this equation instead (thereby sharing the so-called point doubling operations amongst all individual verifications and, potentially, sharing scalars for signers of more than one message). In the case of single verifications, efficient tricks allow reducing the bit-size of the scalars involved in evaluating this expression (thereby effectively halving the required point doubling operations).

With ECDSA itself, these techniques are generally not available, since one cannot generally uniquely (and efficiently) reconstruct R from r: both R and -R yield the same r value. If the conversion function only has two pre-images, though, one can use malleability to remove ambiguity altogether.

The modified ECDSA signing procedure is as follows:

a. Generate ECDSA signature (r,s) of message m;

b. If the ephemeral signing key R has odd y-coordinate, change (r,s) to (r,-s).

Note that this modified signing procedure removes the ambiguity in the reconstruction of R from r if the conversion function would otherwise only have two preimages, since R and -R have different parity. In practice, this is the case for all prime-order curves, including the NIST prime curves P-256, P-384, P-521, and all standardized Brainpool curves.

NOTE: With ECDSA, any party (not just the signer) can recompute the ephemeral signing key R' from a valid signature, since R':=(1/s)(e*G+r*Q). In particular, any party can retroactively put the ECDSA signature in the required form above, thereby allowing subsequent unique reconstruction of the R value from r by verifying entities that know this modified signing procedure was indeed followed.

4. Transitionary Considerations

The modified signing procedure described in Section 3 facilitates the use of accelerated ECDSA verification techniques by devices that wish to do so, provided these know that this modified signing procedure was indeed followed. This can be realized via a new "fast-verification-friendly" label (e.g., OID) indicating that this was indeed the case. This has the following consequences:

a. New device: accept both old and new label and apply speed-ups if possible (and desired);

b. Old device: implement flimsy parser that replaces new label by old label and proceed as with traditional ECDSA verification.

Note that this parser "label replacement" step is a public operation, so any interface can implement this step.

As suggested before, any device can implement the modified ECDSA signing procedure retroactively, so one could conceivably implement this once for all existing ECDSA signatures and only use "new" labels once this task has been completed (i.e., old labels could be mothballed from then on).

5. Implementation Status

[Note to the RFC Editor] Please remove this entire section before publication, as well as the reference to [RFC7942].

The ECDSA* signature scheme has been implemented in V2V specification [P1609.2].

6. Security Considerations

The representation conversions described in this document are publicly known and, therefore, do not affect security provisions.

7. Privacy Considerations

The representation conversions described in this document are publicly known and, therefore, do not affect privacy provisions.

8. IANA Considerations

With the current draft, no IANA code point assignments are requested.

9. Acknowledgements

place holder.

10. References

10.1. Normative References

[FIPS-186-4]
FIPS 186-4, "Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-4", US Department of Commerce/National Institute of Standards and Technology, Gaithersburg, MD, July 2013.

[I-D.ietf-lwig-curve-representations]
Struik, R., "Alternative Elliptic Curve Representations", draft-ietf-lwig-curve-representations-19 (work in progress), December 2020.

[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/info/rfc2119>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[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>.

[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/info/rfc8174>.

[SEC1] SEC1, "SEC 1: Elliptic Curve Cryptography, Version 2.0", Standards for Efficient Cryptography, , June 2009.

[SEC2] SEC2, "SEC 2: Elliptic Curve Cryptography, Version 2.0", Standards for Efficient Cryptography, , January 2010.

10.2. Informative References

[ECC] I.F. Blake, G. Seroussi, N.P. Smart, "Elliptic Curves in Cryptography", Cambridge University Press, Lecture Notes Series 265, July 1999.

[GECC] D. Hankerson, A.J. Menezes, S.A. Vanstone, "Guide to Elliptic Curve Cryptography", New York: Springer-Verlag, 2004.

[P1609.2] IEEE 1609.2-2013, "IEEE Standard for Wireless Access in Vehicular Environments-Security Services for Applications and Management Messages", IEEE Vehicular Technology Society, New York: IEEE, 2013.

[SAC2005] A. Antipa, D.R. Brown, R. Gallant, R. Lambert, R. Struik, S.A. Vanstone, "Accelerated Verification of ECDSA Signatures", SAC 2005, B. Preneel, S. Tavares, Eds., Lecture Notes in Computer Science, Vol. 3897, pp. 307-318, Berlin: Springer, 2006, 2005.

[SAC2010] R. Struik, "Batch Computations Revisited: Combining Key Computations and Batch Verifications", SAC 2010, A. Biryukov, G. Gong, D.R. Stinson, Eds., Lecture Notes in Computer Science, Vol. 6544, pp. 130-142, Berlin-Heidelberg: Springer, 2011, 2005.

Author's Address

Rene Struik
Struik Security Consultancy

Email: rstruik.ext@gmail.com