Requirements for Elliptic Curves for High-Assurance Applications

Manfred Lochter* Johannes Merkle† Jörn-Marc Schmidt‡
Torsten Schütze§

Abstract

Nowadays, cryptography based on the elliptic curve discrete logarithm problem is widely deployed. This is due to its advantages compared to traditional asymmetric schemes relying on the hardness of factoring large numbers or of breaking the discrete logarithm problem in finite fields. The NIST curves provide a standard that is supported by most applications. Other curves like *Brainpool* and *Curve25519* are often implemented in addition.

Currently, different groups like IRTF/CFRG [23] and W3C/Web Crypto WG are actively discussing the standardization of new curve parameters. One major motivation for this debate is an erosion of trust in the NIST curves and a demand for curves that are generated following a reproducible and verifiable process. Ideally such a process should be widely accepted by the crypto community. Another driving factor of the discussion is the growing understanding of secure implementation techniques and attacks.

In this paper, we discuss several aspects to be considered when deciding for a new set of elliptic curves. In particular, we consider implementations of Elliptic Curve Cryptography (ECC) in security hardware and high-assurance software. These implementation forms lead to several requirements that we discuss in detail.

The paper expands upon a previous position paper [18] which provided input for the discussion surrounding suitable curves for TLS and was written by members of the ECC-Brainpool group.

* Bundesamt für Sicherheit in der Informationstechnik, manfred.lochter@bsi.bund.de
† secunet Security Networks, johannes.merkle@secunet.com
‡ secunet Security Networks, joern-marc.schmidt@secunet.com
§ Rohde & Schwarz SIT, torsten.schuetze@rohde-schwarz.com

1 Introduction

The security of Elliptic Curve Cryptography (ECC) cannot be assured without using appropriate curve parameters. The choice of curve parameters is crucial. It must address all known cryptographic attacks on ECC and must be done in a way that is trusted by the community. In addition, the choice of curve parameters has an enormous influence on the performance of the resulting cryptosystem. Often, the two objectives, security and performance, conflict. Even worse, performance optimization is an ambiguous goal: it depends among others on the target architecture, the optimization objectives, and the countermeasures necessary to protect against potential adversaries. In addition, the security of a curve is connected to the cryptographic mechanism that utilizes the curve and the environment in which it is used.

The requirements for ECC algorithms depend strongly on the environment and the types of

adversaries its implementation has to withstand. In principle we see three different scenarios:

1. Protected Environment: In this scenario the implementation is considered a black box. An adversary has no access to specific details such as computation time.

2. Network Scenario: It is possible to trigger the implementation via a network connection. In such a setting several attacks such as timing-attacks and oracle attacks are possible.

3. Hostile Environment: An adversary has full access to the implementation and all potential side-channels. Additionally he or she is able to perform active attacks. This scenario regularly occurs with embedded systems and smart cards; and requires high-assurance implementation of ECC.

For the last two scenarios, the implementation of the curve operations and cryptographic algorithms must not allow any unintentional leakage of sensitive information such as (ephemeral) keys or parts thereof. Therefore countermeasures against side-channel attacks are necessary in these cases. Naturally, (almost) none of these measures are free. Therefore, the selection of the underlying curve is crucial; as it either eases protecting an implementation or leads to huge computational overhead.

A majority of recent research papers focuses on high-performance software implementations. This has led to a discussion concentrating on curves and special prime fields which allow specific optimizations and, hence, fast software implementations. Nevertheless, we are convinced that hardware or high-assurance software[1] requirements should rank equally.

Therefore, this paper focuses on these requirements. Our point of view is motivated by the increasing demands for high-assurance ECC on constrained devices such as in the smart metering scenario. Furthermore, looking at recent security incidents like Heartbleed[2], we consider it advisable to transfer critical cryptographic operations to specialized hardware modules to protect private keys from exposure by implementation flaws. We argue that points that are valid for the fast-changing software world, where implementations are only used in specific, targeted scenarios which focus on performance, cannot be carried over easily to secure hardware implementations or high-assurance software. Hence, we see the need for two sets of curves; one that fulfills high-performance demands and one that is suitable for high-assurance ECC on constrained devices. In the following, we only discuss requirements for the latter ones.

We also note that potential new standard curves will probably have a very widespread use in applications that, today, cannot be foreseen. Therefore, with respect to security, a conservative approach should be taken.

This paper is organized as follows: Section 2 discusses some recent skepticism about existing curves in international standards (i. e., NIST and Brainpool curves). Section 3 focuses on the requirements for new curves. Their impact on interoperability and performance are discussed in Section 4 and Section 5, respectively. In Section 6 we examine FIPS 140-2, and in Section 7 we draw conclusions.

2 Skepticism about Existing Standard Curves

Even though no security weaknesses have been demonstrated, quite recently some skepticism about the NIST parameters has arisen. This skepticism is mainly grounded in discussions about a potential backdoor in the NIST SP800-90A DUAL_EC_DRBG[3].

In addition to this, Bernstein et al. [4] raise trust issues relating to the selection of the NIST

1 We consider certification according to a commonly accepted certification scheme, e. g., the Common Criteria (`https://www.commoncriteriaportal.org/`), as a requirement for high-assurance implementations.

2 `http://heartbleed.com/`

3 `http://www.nist.gov/itl/csd/sp800-90-042114.cfm`

and Brainpool curves, arguing that the degrees of freedom in their generation could have provided a means to intentionally select curves with vulnerabilities that are not publicly known.

We believe that these arguments are only partly true. While the lack of justification for the selection of the seeds from which the NIST curves have been derived indeed leaves room for concerns that the curves may have been chosen with some hidden properties, the Brainpool curves have been derived in a very straightforward manner from the most prominent natural constants π and e which hardly leaves room for any manipulations. See [21] for more details.

3 Elliptic Curve Specifications and Criteria

In what follows we only address elliptic curves defined over a finite field $GF(p)$ of prime characteristic[4] $p > 3$. Every such curve can be represented in short Weierstrass form

$$E: \quad y^2 = x^3 + ax + b.$$

The order of $E(GF(p))$ is denoted by $\#E$. For secure elliptic curves $\#E$ has the form $\#E = hn$, where n is a large prime and the *cofactor* h is a small number. Let G be a base point that generates the cyclic subgroup of order n. The tuple (p, a, b, G, n, h) is called *curve parameters*.

Using the short Weierstrass form in the following does not mean that we recommend performing the ECC computations in this format nor necessarily that the wire format is in affine short Weierstrass. Often projective coordinates (X, Y, Z) or just (X, Z) are used for internal computations and compressed affine points are used on the wire.

4 Especially, we do not consider curves over extension fields or pairing-friendly curves. These would require the consideration of different additional aspects as well as further attacks. See, for example, [17, 12].

3.1 Selection of the finite field

The use of a prime number with a sparse binary representation, in particular a Pseudo-Mersenne prime or a prime like $2^{255}-19$, as field size enables optimizations of the field arithmetic. Since software allows short design cycles and fast adoption for new developments, such optimizations can, at least in the non-embedded world, be integrated rather quickly. This is different for high-assurance software and for dedicated hardware in hostile environments (Scenario 3). While hardware implementations can also be optimized for the prime being used and can gain some additional speed for a single distinguished curve, this potential gain in performance comes at the cost of flexibility. Changing the prime requires a complete redesign of the hardware; modifications in the field are not possible. Further, this approach requires different multiplier implementations for ECC and RSA in devices that support both algorithms.

General purpose smart cards are an example of this type of equipment. These cards are not designed for *one fixed* curve. Instead they have a universal arithmetic-unit for computations in $GF(p)$ (and, possibly, in $GF(2^n)$ as well). These arithmetic-units are flexible enough to perform modular operations for a wide range of bit sizes.

The implementation of an optimized arithmetic/modular multiplier for (additional) special primes in hardware is a major investment for any manufacturer and, thus, comes with high financial risk. In addition, for a manufacturer it may not be desirable to design its product for one fixed curve or prime field, in particular, if RSA has to be supported as well. The requirements for different markets may vary, and it is more cost efficient to offer one general purpose solution that offers a flexible parameter choice.

Where the hardware implementation is flexible (i. e., supports arbitrary primes), a special shape of the prime does not improve performance. Moreover, it has negative influence on implementation security, as it hinders efficient randomization for preventing side-channel attacks in hostile environments (Scenario 3). A prime field size with sparse binary representation requires larger blind-

ing factors for its randomization as well as for the randomization of the secret scalar, because, by the Hasse-Weil theorem, it also yields a sparse representation of the curve order.

One of the proposed countermeasures against side-channel attacks on elliptic curve point multiplication is Coron's first countermeasure [10]. Let $P \in \langle G \rangle$ and $\lambda \in \{0, \ldots, n-1\}$. Instead of computing λP directly, one chooses a random number r (usually r has 32 bits, [10] suggests using 20 bits) and computes $(\lambda + rn)P$. It has been observed [9][5] that the validity of this countermeasure relies on the structure of the binary representation of p. According to the Hasse-Weil Theorem, for cofactor-one curves, the upper half of the binary representation of the prime number p equals that of the group order n.[6] If this part of p contains long runs of zeroes or ones (e. g., in Pseudo-Mersenne primes or the primes over which the NIST curves are defined), some bits of r and λ can directly be accessed through measurements, see e. g. [13].

As a consequence, the effort for blinding increases considerably for special primes: For primes close to 2^k, a blinding factor of almost $k/2$ bits is required to thwart side-channel attacks [25]. In contrast, for curves over arbitrary primes, the best known attacks rapidly become inefficient when blinding factors longer than 64 bit are used [24].

In conclusion, we argue that for secure hardware and high-assurance software, a set of curves over unstructured primes should be defined and required to be implemented for interoperability reasons. In order to avoid any potential allegations of having manipulated the primes chosen, these should be generated *verifiably pseudo-random*. One approach for pseudo-random prime generation was introduced and has been successfully applied [19, 6].

5 The main ideas from [9] are also reproduced in [11], which is easily available, and later independently in [22].

6 Other cofactors yield a similar structure-preserving property.

3.2 Twist Security

In some applications one wants to use only the x-coordinates of curve points. An adversary therefore might try to change x-coordinates such that computations are done on an insecure curve, as explained in [15]. In the most important examples, this attack may lead to computations on the twist[7]. Therefore it is desirable that the twist of E is also cryptographically strong, if the fault attack cannot be ruled out by other measures. Such curves are called *Twist Secure*. The use of twist secure curves may improve the security of careless implementations.

However, it should be noted that the condition 'b a quadratic non-residue', see e. g. [19, 16], can only be fulfilled by one of these curves, either by the original curve or by its twist, thus, potentially enabling zero-value attacks, see [1], or fault attacks on common points, see [2].

Nonetheless, even with twist security, implementations that perform computations on full coordinates (x, y) or (X, Y, Z) must check group membership of received curve points to thwart invalid point attacks.

Summarizing, we conclude that the benefit of twist security is somehow limited.

3.3 Cofactor

A cofactor greater than one can enable small subgroup attacks, where an active adversary chooses a point of small order as his public Diffie-Hellman key to gain partial information on the other's private key. If the protocol does not prevent leakage of the shared DH-point, an additional check or an additional multiplication operation is needed to prevent this attack. This is not only an additional overhead and may require changes in existing implementations for prime order curves but also a source of implementation weaknesses if the check is omitted or just forgotten[8].

7 If $\#E(GF(p)) = p + 1 + t$ we consider the (GF(p)-isomorphism) class of curves $E' : y^2 = x^3 + au^2x + bu^3$ with $\#E'(GF(p)) = p + 1 - t$ as twist of E.

8 Some implementations in the field neglected this check for years. In the case of the Internet Key Exchange (IKE) protocol, this issue was eventually addressed in

In recent discussions, curves in Montgomery or Edwards form have been proposed, see Section 4, both of which require the cofactor to be at least four. These curve representations are mainly advertised for their simple, efficient and time-constant arithmetic.

In hostile environments (Scenario 3), however, the implementations do not only need to be time-constant but must also be protected against other side-channel attacks like DPA and DEMA which significantly reduces the advantages of the simplified arithmetic on Montgomery/Edwards curves. For instance, a typical argument in favour of Montgomery curves is the existence of fast and time-constant single-coordinate ladders. But, for high-assurance implementations where measures like randomization are needed to thwart advanced side-channel attacks, a generalized Brier-Joye ladder for projective coordinates (X, Z) on Weierstrass curves is also exception-free, competitive and even used in practice [14].

The authors are not aware of an implementation of Montgomery curves or Edwards curves (especially not of *Curve25519*) that provides comprehensive protection against all the attacks that need to be considered in implementations for hostile environments. Hence, it is not clear to us, whether these curves provide any advantage in Scenario 3.

Summarizing, we prefer a cofactor $h = 1$ unless it is shown that Montgomery curves or Edwards curves can provide significant advantages in the hostile environment scenario that outweigh the disadvantages of $h > 1$.

3.4 Rigidity

Recent revelations on manipulations of cryptographic standards have raised the demand for a transparent and traceable process on how to select curve parameters. One approach to achieve this is to use a pseudo-random generation process which is seeded by natural constants. For example, the *Brainpool curves* [19] have been generated this way. The second possibility is to define a set of desired properties and to choose out of the remaining options the one with some minimal property, e. g., those with the best performance.[9] This is how, for example, the *Curve25519* [3] and the *NUMS curves* [5] have been constructed.

Both processes provide very limited flexibility. Nevertheless, the choice of input parameters as well as the choice of desired properties heavily influences the result.

Twist security may serve as an example. We have already argued that twist security is not strictly necessary. Furthermore, there are mathematical attacks like the Brown-Gallant-Cheon attack [7, 8] that cannot, in every circumstance, be prevented through careful implementations.[10] So why not introduce Brown-Gallant-Cheon-resistance as requirement? This would lead to another set of secure elliptic curves. Hence, perfect rigidity, defining a process that is accepted as completely transparent and traceable by everyone, seems to be impossible.

In 2005, for instance, the ECC-Brainpool consciously decided against twist security and Brown-Gallant-resistance as design criteria.

In our opinion both approaches to rigidity, minimality and verifiable pseudo-randomness, are acceptable provided that first the requirements are specified and then, afterwards, the parameters are selected.

3.5 Implementation Security

Besides mathematical security, the selection of curve parameters influences the effort needed to achieve a certain level of implementation security. In this context, the required protection depends on the application scenario, which we outlined in Section 1. In cases where the implementation runs in a secure environment and only remote

RFC 6989 [26].

9 See the discussion in Section 1. Performance is very architecture specific; however, according to Moore's law, the processing capacity of microcontrollers will continue to improve.

10 For instance, there are undeniable signature schemes, where the oracle $P \to x \cdot P$ (with x being the private key) needed by the Brown-Gallant-Cheon attack is inevitably available to an attacker.

attacks are possible (Scenario 2), a time-constant implementation may be sufficient.

If the implementation has to withstand adversaries with physical access such as in Scenario 3 (i. e., to be resistant against other side-channels like power consumption and electromagnetic emanations or even against active fault injection) a combination of more advanced protection techniques like randomization is essential [16].

While realizations of constant time implementations are possible for Weierstrass curves as well, a common argument for the use of Montgomery / Edwards curves is that achieving constant time computations is easier. Nevertheless, it should be noted that this is not sufficient for protection against the adversaries considered in Scenario 3 (cf. discussion in Section 3.3).

4 Interoperability — Curve Representation and Exchange Format

Current standards like those of ISO, ANSI, IETF, BSI, and NIST describe elliptic curves in short Weierstrass form. In addition, data structures for exchanging points of a curve in protocols are also specified as either affine coordinates or compressed affine coordinates for short Weierstrass curves.

In the current discussion, different proposals like Montgomery curves and (twisted) Edwards curves are considered since they allow simple and efficient arithmetic. Points on curves in either of these forms can be efficiently transformed to affine coordinates in Weierstrass form and vice versa. This allows using Montgomery and (twisted) Edwards curves in implementations while still using the exchange formats defined in current standards. In our view, at least for secure hardware and high-assurance software, this approach reduces the implementation costs in comparison to adapting the exchange format. Thus, it is strongly preferable in the case that other formats like Montgomery or (twisted) Edwards curves are standardized.

5 Performance — Flexibility, Agility, and Costs

In the recent discussion on new curves, one point of view that has been expressed is that a single set of curves is sufficient for all use cases. We do not share this opinion for three main reasons.

We consider at least two sets of curves as necessary: one for high speed applications in software and one for secure hardware/ high-assurance software addressing Scenario 3. Nevertheless, those two worlds have to be able to interact. Since we have already argued that special primes are not well suited for high-assurance solutions, we assume the usage of a curve over a random prime field for this communication.

Finally, even though we currently expect only generic attacks on curves, we cannot be sure about future developments. In the case of cryptanalytic breakthroughs, there is a chance that at least one of the curves would not be affected. Furthermore, a single distinguished curve would be an exposed target for an adversary.

The cost for implementation and usage of elliptic curves does not only depend on their implementation-forms and performance. In particular, for high-assurance devices, the evaluation and certification cost has to be considered as well. Further, it has to be noted that for server implementations where high-performance curves are preferred, the cost of supporting additional curves depends on the question as to how often the additional curves are used. Supporting two different sets of curves does not affect the performance of equipment (e. g., TLS servers) that mainly uses special-prime curves.

6 Connection with FIPS 140-2

The predominant status of the current NIST curves stems, partially, from the fact that NIST was one of the the first standardization bodies for ECC. In addition to this, NIST curves are also required to receive a FIPS 140-2 certificate which is necessary for cryptography used by the US government.

We believe that it is rather disadvantageous that a crypto device has to disable all other non-NIST ECC curves when working in FIPS-mode. Common Criteria certification is much more flexible in this regard. We would like to see other curves, e. g., *Brainpool* curves and *Curve25519*, in FIPS 140-2 or its successor.

7 Conclusion

We do not see an immediate need to withdraw the current NIST curves. Nevertheless, we see a benefit in standardizing additional, trusted curves that have been generated in a rigid way.

In our view, flexibility and security are most important, and performance ranks third. We believe that the previously mentioned considerations and conclusions provide a necessary viewpoint to the current discussion on the development and the selection of future standard elliptic curves. In particular, a set of curves should be defined that is suitable for high-assurance implementations and addresses Scenario 3. One option is to adopt the existing Brainpool curves [19, 20].

Alternatively, a new set of high-assurance curves could be generated. In this case, we believe this generation should use a verifiably pseudo-random process. The generation of the Brainpool curves [6] could serve as blueprint. This process is similar to the methods described in [27] but uses natural constants as seeds. Several of the criteria that should be met by the new curve parameters are discussed in this paper. However, these aspects are not exhaustive. A complete list can be found, for example, in [6].

For Scenario 2, we could think of adding the curves chosen by the IRTF/CFRG to the NIST suite.

New curves should, in our opinion, not come with new wire formats. Instead, affine short Weierstrass and compressed format should continue to be used. Further, new curves should not imply new or specific cryptographic algorithms.

We hope that the NIST workshop will help to re-establish trust into ECC.

References

[1] Toru Akishita and Tsuyoshi Takagi. Zero-value register attack on elliptic curve cryptosystem. *IEICE Transactions*, 88-A(1):132–139, 2005.

[2] Alberto Battistello. Common points on elliptic curves: The Achilles' heel of fault attack countermeasures. In Emmanuel Prouff, editor, *Proceedings of COSADE 2014*, volume 8622 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2014.

[3] Daniel J. Bernstein. Curve25519: New Diffie-Hellman Speed Records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Proceedings of Public Key Cryptography — PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.

[4] Daniel J. Bernstein, Tung Chou, Chitchanok Chuengsatiansup, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, and Christine van Vredendaal. How to manipulate curve standards: a white paper for the black hat. Cryptology ePrint Archive, Report 2014/571, 2014. `http://eprint.iacr.org/`.

[5] Joppe W. Bos, Craig Costello, Patrick Longa, and Michael Naehrig. Selecting elliptic curves for cryptography: An efficiency and security analysis. Cryptology ePrint Archive, Report 2014/130, 2014. `http://eprint.iacr.org/`.

[6] ECC Brainpool. ECC Brainpool Standard Curves and Curve Generation, October 2005. v1.0, 19.10.2005, `http://www.ecc-brainpool.org/download/Domain-parameters.pdf`.

[7] Daniel R.L. Brown and Robert P. Gallant. The Static Diffie-Hellman Problem. Cryptology ePrint Archive, Report 2004/306, November 2004. `http://eprint.iacr.org/`.

[8] J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In S. Vaudenay, editor, *Proceedings of EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2006.

[9] Mathieu Ciet. *Aspects of fast and secure arithmetics for elliptic curve cryptography.* PhD thesis, Université Catholique de Louvain, Louvain-la-Neuve, 2003.

[10] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In C. K. Koç and Chr. Paar, editors, *Proceedings of CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer-Verlag, 1999.

[11] Nevine Maurrice Ebeid. *Key Randomization Countermeasures to Power Analysis Attacks on Elliptic Curve Cryptosystems.* PhD thesis, University of Waterloo, Department of Electrical and Computer Engineering, 2007.

[12] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaël Renault. Using symmetries in the index calculus for elliptic curves discrete logarithm. *J. Cryptology*, 27(4):595–635, 2014.

[13] Benoit Feix, Mylène Roussellet, and Alexandre Venelli. Side-channel analysis on blinded regular scalar multiplications. Cryptology ePrint Archive, Report 2014/191, 2014. `http://eprint.iacr.org/`.

[14] Wieland Fischer, Christophe Giraud, Erik Woodward Knudsen, and Jean-Pierre Seifert. Parallel scalar multiplication on general elliptic curves over F_p hedged against non-differential side-channel attacks. Cryptology ePrint Archive, Report 2002/007, 2002. `http://eprint.iacr.org/`.

[15] Pierre-Alain Fouque, Reynald Lercier, Denis Réal, and Frédéric Valette. Fault Attack on Elliptic Curve with Montgomery Ladder Implementation. In *Proceedings of Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2008*, pages 92–98, 2008.

[16] Bundesamt für Sicherheit in der Informationstechnik. Minimum Requirements for Evaluating Side-Channel Attack Resistance of Elliptic Curve Implementations. AIS 46 Guidance Version 1.0.4, 01.07.2011, BSI, 2011.

[17] Antoine Joux and Vanessa Vitse. Elliptic curve discrete logarithm problem over small degree extension fields. *J. Cryptology*, 26(1):119–143, 2013.

[18] M. Lochter, J. Merkle, J.-M. Schmidt, and T. Schütze. Requirements for Standard Elliptic Curves, Position paper of the ECC Brainpool. Cryptology ePrint Archive, Report 2014/832, September 2014. `http://eprint.iacr.org/`.

[19] Manfred Lochter and Johannes Merkle. Elliptic Curve Cryptography (ECC) Standard Curves and Curve Generation. Internet Request for Comment RFC 5639, Internet Engineering Task Force, March 2010. `ftp://ftp.ietf.org/rfc/rfc5639.txt`.

[20] Manfred Lochter and Johannes Merkle. Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS). Internet Request for Comment RFC 7027, Internet Engineering Task Force, October 2013. `ftp://ftp.ietf.org/rfc/rfc7027.txt`.

[21] Johannes Merkle. Re: [Cfrg] ECC reboot (was: When's the decision?). CFRG mailing list, October 2014. `http://www.ietf.org/mail-archive/web/cfrg/current/msg05353.html`.

[22] Elisabeth Oswald, Daniel Page, and Nigel Smart. Randomised representations. *IET Proceedings on Information Security*, 2(2):19–27, 2008.

[23] Kenny Paterson. Formal request from TLS WG to CFRG for new elliptic curves. CFRG mailing list, July 2014. `http://www.ietf.org/mail-archive/web/cfrg/current/msg04655.html`.

[24] Werner Schindler and Andreas Wiemers. Power attacks in the presence of exponent blinding. *Journal of Cryptographic Engineering*, 4:213–236, 2014.

[25] Werner Schindler and Andreas Wiemers. Efficient side-channel attacks on scalar blinding on elliptic curves with special structure. NIST Workshop on ECC Standards, June 2015.

[26] Yaron Sheffer and Scott Fluhrer. Additional Diffie-Hellman tests for the Internet Key Exchange Protocol Version 2 (IKEv2). Internet Request for Comment RFC 6989, Internet Engineering Task Force, March 2013. `ftp://ftp.ietf.org/rfc/rfc6989.txt`.

[27] ANSI X9.62-2005. *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA).* American Bankers Association, 2005.