Public Comments Received on FIPS 186-4: Digital Signature Standard (DSS)

(December 4, 2015 deadline)

Dear NIST,

Brief comments are provided below for the questions regarding the elliptic curves recommended in FIPS 186-4.
If there are any questions on the comments please contact us.

Mehmet Adalier
CEO, Antara Teknik LLC
madalier@antarateknik.com
(916)-622-6960

1. Digital Signature Schemes
a. Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?
Yes. Specifically the prime curves P-256, P-384, and P-521 provide respectively 128, 192, and 256-bit security, which satisfies the security requirements of applications used by the industry at an acceptable key-size. Furthermore, given the recent advances in IoT/Embedded hardware designs, P-256 provides more than adequate security strength at nominal key sizes for these use-cases.
b. Are there other digital signature schemes that should be considered for inclusion in a future revision to FIPS 186? What are the advantages of these schemes over the existing schemes in FIPS 186?
No. While recently there has been discussions on other digital signature schemes, notably those reviewed by "CFRG," in our opinion these schemes do not actually provide better security compared to a well-designed ECDSA implementation. Additionally, recent high performance implementations of ECDSA P-256 (OpenSSL --S. Gueron, taraEcCRYPT(tm) --- M. Adalier) show that ECDSA can be implemented as fast and securely as the other schemes.

2. Security of Elliptic Curves
a. Do the NIST-recommended curves satisfy the security requirements of applications used by industry?
Yes. P-256, at 128-bit security is adequate for most industry applications. For those applications requiring further security strength P-521 is a decent choice. It appears that binary curves have not been as widely used in the industry.
b. Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?
We are not aware of any attacks of cryptographic significance on ECC that specifically apply to NIST recommended curves or other widely used curves. Side channel attacks can be avoided by careful implementation of the ECDSA scheme irrelevant of the actual parameters of the curve used.

3. Elliptic Curve Specifications and Criteria
a. Is there a need for new elliptic curves to be considered for standardization?
From a technical perspective, the answer is "No." However, there is a notion with some crypto experts that defining new elliptical curves with full visibility into the process may accelerate the adoption of ECC.
b. If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?
An open process with full documentation and justification of any seeds used.
c. Do you anticipate a need to create, standardize or approve new elliptic curves on an ongoing basis?
No. We would recommend against this. This may inadvertently result in weak curves that could be exploited.

4. Adoption
a. Which of the approved digital signature schemes and NIST-recommended curves have been used in practice?
ECDSA, specifically P-256 has been widely used in practice. Notable use cases are secure shell (SSH), TLS, IPSec, and the new BGPSEC protocol. P-384 and P-521 usage is also starting to increase.
b. Which elliptic curves are accepted for use in international markets?
It appears that leading countries have been defining their own national elliptical curves. We assume this trend will continue. Otherwise, P-256 has been used in international applications

5. Interoperability
a. If new curves were to be standardized, what would be the impact of changing existing implementations to allow for the new curves?
For well-designed ECDSA implementations, the impact should not be significant.
b. What is the impact of having several standardized curves on interoperability?
As long as each implementation is tested appropriately using NIST Test vectors and validation program, we do not anticipate any interoperability issues.
c. What are the advantages or disadvantages of allowing users or applications to generate their own elliptic curves, instead of using standardized curves?
The disadvantage is the probability of using weak curves. Not sure there are any tangible technical advantages.

6. Performance
a. Do the performance characteristics of existing implementations of the digital signatures schemes approved in FIPS 186-4 meet the requirements of applications used by industry?
For the most widely used ECDSA P-256 the answer is "Yes." Since 2013, OpenSSL includes a high performance implementation of P-256 by S. Gueron of Intel Corp. which delivers about 30,000 sign operations/sec and 11,800 verify operations per sec on commodity CPUs. As of early 2015, Antara Teknik's taraEcCRYPT performance for sign operation can reach over 63,800 sign operations per sec and 31,800 verify operations per sec on similar commodity CPUs. These performance levels are more than adequate for industry applications and network protocols.
While technically possible, high performance implementations of P-384 and P-521 are not widely available yet. Antara is working on publishing data on a high performance implementation of P-521.

7. Intellectual Property
a. What are the desired intellectual property requirements for any newcurves or schemes that could potentially be included in the Standard?
Preferably, there should be no IP restrictions for any new curves or schemes that could potentially be included in the Standard.
b. What impact has intellectual property concerns had on the adoption of elliptic curve cryptography?
Mostly perceived, and some real IP concerns may have hindered the adoption of ECC.

Comments from: Agence nationale de la sécurité des systèmes d'information (ANSSI)

Defining elliptic curves suitable for cryptographic use implies tradeoffs between different criteria. Although it is of interest to pick curves optimized for speed as was done in Appendix D given the ECC state of the art at its time of publication, we think that it is very important to define curves with security only in mind and therefore as "generic" as possible.
We would therefore recommend to include at least two sets of curves if the standard were to be updated:
* a set where the possibility of a very efficient implementation is kept in mind: that could very well be the current set of curves which we have no reason not to trust anymore;
* a set of "generic" curves with completely random-looking coefficients defined over a "generic" prime base field with a random-looking characteristic, of prime orders (and preferably twists of prime orders).
Further details on what is meant by "generic" can be found in the ANSSI contribution presented during last summer NIST ECC workshop (http://csrc.nist.gov/groups/ST/ecc-workshop-2015/papers/session4-flori-jean-pierre.pdf).
We think that only supporting one unique set of curves with a special shape would be a bad signal sent to the users especially at a time when ECC is not so trusted anymore.

In both cases the ability to easily validate the process of generation is very important.
Providing certificates as suggested in by ANSSI (http://csrc.nist.gov/groups/ST/ecc-workshop-2015/papers/session4-flori-jean-pierre.pdf) to leverage the computational complexity of replaying the generation process might be helpful.
The generation process of the seed used to sample curves should also be as transparent as possible, although it is a highly non-trivial matter.
Involving third parties could make it more transparent and trustable.
Once again it is mandatory to make such a process as transparent, reproducible and easily verifiable to restore users' trust in ECC .

As far as elliptic curves over binary fields are concerned, we feel they should be discarded.

As far as security levels are concerned, we feel that defining curves over fields of 256, 384 and 512 bits is enough.

---Page 88 on normal basis: Type T (T>1) low-complexity normal basis is called Gaussian normal basis (GNB) which is a special class for normal basis. Normal basis exists for every positive integer m but it is not necessarily low-complexity normal basis.

Also, it says see Appendix D3. I did not see anything related to low-complexity normal basis there. Low-complexity normal basis does not exist for the ms recommended in this document for ECC but Gaussian normal basis does exist. Therefore, it should say Gaussian normal basis instead of normal basis. For example, for m=283 low-complexity normal basis is not available and we use Gaussian normal basis of Type 6.

Reza Azarderakhsh, Ph.D.,
Assistant Professor
Department of Computer Engineering
Rochester Institute of Technology
83 Lomb Memorial Drive, Bldg 09, GLE 3461
Rochester, NY 14623-5603
Phone: 585-475-4083
https://people.rit.edu/~rxaeec/

Comments on NIST's ECC standards

Daniel J. Bernstein[1,2] and Tanja Lange[1]

[1] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
`tanja@hyperelliptic.org`
[2] Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607–7045, USA
`djb@cr.yp.to`

Abstract. NIST's ECC standards create (1) unnecessary losses of simplicity, security, and speed in ECC implementations and (2) unnecessary tensions between simplicity, security, and speed in ECC implementations.

1 Introduction

The poor user is given enough rope with which to hang himself—something a standard should not do. —Rivest, 1992 [29], commenting on the NIST/NSA "DSA" proposal

NIST's standards for elliptic-curve cryptography (ECC) consist of

- NSA's choices of *primes*, such as the "P-256 prime" $2^{256}-2^{224}+2^{192}+2^{96}-1$;
- NSA's choices of *curves* modulo those primes, such as "NIST P-256", the curve $y^2 = x^3 - 3x + 41058363725152142129326129780047268409114441015993725554835256314039467401291$ modulo the P-256 prime;
- NSA's choices of *coordinates* for transmitting points on those curves, such as "uncompressed short Weierstrass coordinates" (x, y) for NIST P-256;
- NSA's choices of *computations* to be used inside implementations, such as "addition-subtraction methods" for NIST P-256; and
- NSA's choices of *protocols* using these curves, such as the "ECDSA" signature algorithm.

NIST's FIPS 186-4 specifies the choices of primes and curves, and refers to ANSI X9.62 for the choices of coordinates, computations, and protocols. FIPS 186-4

This work was supported by the European Commission under Contract ICT-645421 ECRYPT-CSA; by the Netherlands Organisation for Scientific Research (NWO) under grant 639.073.005; and by the U.S. National Science Foundation under grant 1018836. "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation." Permanent ID of this document: `1c4db65fd2bf7a58d36655b8e8690a541cd8ecab`. Date: 2015.12.04.

is a “Digital Signature Standard” but the curves are also used in non-signature protocols, notably ECDH.

1.1. Potential problems with the NIST ECC standards. News reports in September 2013 indicated that NSA had backdoored its “Dual EC” elliptic-curve random-number generator (also standardized by NIST), and more broadly that NSA had a \$250 million/year program designed to “make [systems] exploitable through SIGINT collection” by inserting vulnerabilities, collecting target network data, and influencing “policies, standards and specifications for commercial public key technologies”. This prompted extensive public discussion of whether vulnerabilities could have been somehow inserted not merely into NIST’s standards for elliptic-curve *random-number generation* but also into NIST’s much more widely used standards for elliptic-curve *cryptography*.

The Dual EC vulnerability is exploitable only by attackers in possession of a secret back-door key. The news reports make reasonably clear

- that NSA possesses this key and
- that NSA does not have a stellar track record of keeping secrets;

it is therefore easy to imagine many different attackers having copies of the key. We have seen some commentators speculating that NSA inserts vulnerabilities into systems *only* if those vulnerabilities have secret back-door keys, but we have seen no evidence to support this speculation.

The *risk* of an intentional vulnerability is particularly difficult to assess. One hopes that the extensive public study of ECC has been sufficiently comprehensive, and that all curves passing public security criteria are in fact secure; if this is true then the choice of a curve cannot create a vulnerability. However, it is possible that this hope is incorrect, and that the curve generator is aware of security variations among these curves. A curve generator with the flexibility to choose from among a pool of N curves can turn a curve vulnerability with probability ϵ into a curve vulnerability with probability $1-(1-\epsilon)^N \approx N\epsilon$ (assuming statistical independence). It is entirely possible that $N\epsilon$ is large even if ϵ is small.

Even worse, N is surprisingly high, as shown in our paper “How to manipulate curve standards: a white paper for the black hat” [5] with Chou, Chuengsatiansup, Hülsing, Lambooij, Niederhagen, and van Vredendaal. The examples in that paper[3] illustrate that claims of protection against backdoors need to be carefully studied, just like other security claims. It would be particularly embarrassing if, e.g., NIST adopts some half-baked idea for using “verifiably random” Bitcoins to generate curves, and a subsequent study then shows that manipulating those Bitcoins would have been well within the budget of today’s attackers.

1.2. Definite problems with the NIST ECC standards. We are concerned that attention to the *possibility* of back doors is distracting NIST from what is *definitely* going wrong with NIST’s ECC standards. Most importantly, NIST’s ECC standards create unnecessary complexity in ECC implementations. This unnecessary complexity is important because it

[3] We incorporate that paper into these comments by reference.

- scares away implementors,
- reduces ECC adoption,
- interferes with optimization,
- keeps ECC out of small devices,
- scares away auditors,
- interferes with verification, and
- creates ECC security failures.

It is theoretically *possible* to implement NIST's ECC standards in a secure way, but it is unnecessarily *difficult*—even in situations where there are no performance constraints. Even worse, the path to a secure implementation is littered with unnecessary traps: ways that simpler implementations seem to work but are not in fact secure. Implementors naturally try to reduce complexity, and end up falling into these traps, compromising security.

Are we saying that cryptographers should always apply every imaginable simplification? Of course not. For example:

- ECB is simpler than GCM. Should GCM users switch to ECB? No: that would be an oversimplification. The problem here is that ECB doesn't authenticate and doesn't securely encrypt.
- Multiplicative groups are simpler than elliptic-curve groups. Should ECC users switch to multiplicative groups? No: that would be an oversimplification. The problem here is that multiplicative groups are vulnerable to index calculus. This produces bigger keys and slower computations; this also makes the security analysis more difficult and less stable, reducing confidence.

As these examples illustrate, the cryptographer's top priority is security, and the cryptographer's second priority is to meet the user's performance requirements. Simplicity is only the third priority.

Sometimes, starting from examples of oversimplification damaging security or speed, people say "Simplicity damages security" or "Simplicity damages speed" or "Simplicity in cryptography is bad". These are wild overgeneralizations, often used to cover up deficient analyses of speed and security. Many simplifications don't hurt security at all and don't hurt speed at all. In fact, the simplicity of next-generation ECC *contributes to security* and *contributes to speed.*

The next five sections of this document analyze ways that simplicity, security, and speed are compromised by NSA's choices of primes, curves, coordinates, computations, and protocols. The remaining sections of this document answer various questions that were specifically asked by NIST.

2 Protocols

The following high-level description of ECDH is uncontroversial. There is a standard base point B on a standard elliptic curve E over a standard finite field. Alice generates a secret integer r and a corresponding public key rB. Bob generates a secret integer s and a corresponding public key sB. Alice computes rsB

as $r \cdot sB$, and Bob computes the same rsB as $s \cdot rB$. Some sort of hash of this shared secret rsB is used to encrypt and authenticate data.

For *signatures* there are many more choices to make at the same level of ECC protocol description. Some amount of extra complexity seems unavoidable,[4] but poor choices add unnecessary further complexity, along with damaging speed and security.

The rest of this section focuses on the details of two signature protocols: ECDSA, which is part of NIST's standards, and EdDSA, which we introduced in a paper [8] with Duif, Schwabe, and Yang and generalized in a paper [9] with Josefsson, Schwabe, and Yang.[5] See http://blog.cr.yp.to/20140323-ecdsa.html for a step-by-step explanation of how to obtain EdDSA through modifications to the original ElGamal signature system.[6]

In EdDSA, the verifier checks the equation $SB = R + H(R, A, M)A$. Here B is a standard base point as above; H is a standard hash function; A is the signer's public key; R is a curve point included in the signature; S is an integer included in the signature; and M is the message being verified.

In ECDSA, the verifier checks the equation $H(M)B + x(R)A = SR$. This forces the signer to perform a division, damaging speed and simplicity. It also forces the verifier to perform a triple-scalar multiplication, or to instead check $(H(M)/S)B + (x(R)/S)A = R$, again damaging speed and simplicity.

An ECDSA signature (R, S) could be encoded, without any change of security, as (R, S') where $S' = S/H(M)$. The verifier would then check the equation $B + H'(R, M)A = S'R$, where $H'(R, M) = x(R)/H(M)$. This view shows that, from a security perspective, moving from ECDSA to EdDSA means

- putting S in front of B rather than R;
- replacing H' with a conventional hash function H; and
- including A as an extra hash input.

The first change was introduced by Schnorr and eliminates divisions. The second change eliminates the multiplicative structure of H', improving security as discussed below. The third change alleviates concerns that several public keys could be attacked simultaneously.

In ECDSA, an attacker who finds a collision in H also finds a collision in H', breaking the system. EdDSA is collision-resilient: hash-function collisions do not break the system.

Schnorr used collision resilience as justification for taking a hash function with smaller output, and then used this to save some space in signatures, replacing

[4] This complexity is created by the basic data flow in signatures, where a signature is sent through a one-way communication channel to a receiver. A receiver who has a two-way communication channel with the sender can skip signatures and use ECDH to achieve even better integrity protection, guaranteeing that the sender is vouching for the data *now* ("freshness") rather than at some indeterminate time in the past. ECDH also easily provides confidentiality protection, protection for communication in the opposite direction, etc.

[5] We incorporate those papers into these comments by reference.

[6] We incorporate that blog post into these comments by reference.

R with the H output. EdDSA instead uses collision resilience as an extra line of defense, uses R in signatures to allow fast batch verification, and takes a double-size H output.

In all of these systems, the signer generates R as rB, where r is a one-time secret scalar. EdDSA generates r by deterministically hashing a secret together with M, so randomness is not used after key generation. This makes the signing procedure much easier to test and audit. This idea was proposed by Barwood and Wigley in 1997, many years before poor generation of r revealed the signing key for the Sony PlayStation 3.

3 Computations inside scalar multiplication

We assume that NIST is familiar with the long history of successful timing attacks against cryptographic software, and in particular ECC software. Constant-time ECC software is critical for ECC security. This has an impact on several layers of choices inside ECC standards.

There are other important side channels. We have chosen to highlight timing because timing attacks are particularly dangerous. Timing is visible through networks; timing is visible to untrusted code running on the same machine (for example, in browsers); timing is unaffected by typical physical-security mechanisms.

3.1. The Montgomery ladder. Curve25519 is the Montgomery curve $y^2 = x^3 + Ax^2 + x$ over $\mathbf{F}_p$, where $p = 2^{255} - 19$ and $A = 486662$. Each curve point Q has an x-coordinate $X_0(Q)$ defined as follows: if $Q = (x, y)$ then $X_0(Q) = x$; if Q is the point at infinity then $X_0(Q) = 0$. Here is a spectacularly simple method of single-scalar multiplication on Curve25519:

```
x2,z2,x3,z3 = 1,0,x1,1
for i in reversed(range(255)):
  bit = 1 & (n >> i)
  x2,x3 = cswap(x2,x3,bit)
  z2,z3 = cswap(z2,z3,bit)
  x3,z3 = ((x2*x3-z2*z3)^2,x1*(x2*z3-z2*x3)^2)
  x2,z2 = ((x2^2-z2^2)^2,4*x2*z2*(x2^2+A*x2*z2+z2^2))
  x2,x3 = cswap(x2,x3,bit)
  z2,z3 = cswap(z2,z3,bit)
return x2*z2^(p-2)
```

The inputs are an integer `n` $\in \{0, 1, 2, \ldots, 2^{255} - 1\}$ and the x-coordinate `x1` $= X_0(Q)$ of a curve point Q. The output is exactly $X_0(nQ)$. The operations `+`, `*`, etc. are constant-time arithmetic operations in $\mathbf{F}_p$. The `cswap` function performs a constant-time conditional swap.

This scalar-multiplication method, the "Montgomery ladder", works for *all* inputs, as shown in [4]. There is no need to check for any special cases. The theorem in [4] applies to any Montgomery curve with a unique point of order 2:

any curve $By^2 = x^3 + Ax^2 + x$ over any prime field $\mathbf{F}_p$ where $p \geq 5$, $B \neq 0$, and $A^2 - 4$ is non-square. One can also change `255` to another number in the code, allowing that number of bits in the scalar `n`. With some extra lines of code one can also compute the y-coordinate of nQ given the y-coordinate of Q, but (as pointed out by Miller in [25]) this extra complexity is not necessary for ECDH.

X25519 is a widely deployed ECDH system[7] using Curve25519, with these x-coordinates encoded as 32-byte public keys in a straightforward way. Practically all X25519 implementations use the Montgomery ladder to compute shared secrets. The Montgomery ladder is extremely fast (and fits into small hardware), when the pieces of the computation are appropriately optimized. There are many other ways to compute X25519 shared secrets, but none of them are simpler or faster than the Montgomery ladder, so there is no incentive for X25519 implementations to use them.

Some parts of the literature, instead of presenting a constant-length ladder as above, present a variable-length ladder, starting from the top bit that is *set* in the scalar. Brumley and Tuveri in [12] demonstrated remote extraction of secret keys through timings of OpenSSL's binary-field ECDSA implementation; this implementation used a ladder for multiplication by the one-time scalar, and thus leaked the fact that some scalars were particularly small, allowing the attack to recover the long-term secret key by lattice techniques. Small information leaks regarding ECDH keys do not seem to be as damaging, but we recommend a policy of systematically eliminating *all* timing leaks, rather than giving the auditor the tricky task of assessing the impact of many small leaks.

Several years before [12], X25519 already specified scalars n that always have $2^{254} \leq n < 2^{255}$, so a variable-length Montgomery ladder still ends up taking constant time. X25519 implementors are also encouraged in several ways to focus on implementing X25519, rather than to try to write "generic" implementations that handle multiple curves. One effect of this focus is that X25519 implementors have a natural incentive to avoid variable-length ladders: a variable-length ladder might seem simplest for a "generic" implementation but a length-255 ladder is obviously simplest for an X25519 implementation. To summarize, the X25519 ecosystem was proactively designed to discourage timing leaks.

3.2. Edwards curves. The simplest way to generate X25519 public keys is to reuse the Montgomery ladder to multiply the secret scalar by the standard base point B. It is well known, however, that one can save a factor of roughly 3 in CPU time by instead computing this multiple of B as a sum of various precomputed multiples of B.

We emphasize that simply reusing the Montgomery ladder for key generation is fast enough for most ECDH applications. Obviously one should not add complexity (and code size) for a speedup that users won't actually notice. To find applications that care about the cost of ECDH key generation, one needs to find applications where ECDH is a bottleneck *and* each ECDH key is reused only a very small number of times. A key used as a long-term identifier, or an

[7] Originally, in [4], this ECDH system was called Curve25519, but having a separate name for the ECDH system and the curve has turned out to be helpful.

ephemeral server key reused for two hours,[8] involves mainly shared-secret computations, not key generation. Even in the rather extreme situation of a key being generated and used just once, saving a factor of 3 in key generation means saving a factor of just 1.5 in total time.

Furthermore, there are timing leaks in all of the simplest ways to work with tables of precomputed multiples of B. Consider, for example, a table containing $B, 2B, 4B, \ldots, 2^{254}B$. Simply scanning for the bits of n that are set, and adding the corresponding table entries, will (obviously) leak the number of bits of n through total time, and will (less obviously) leak all bits of n through higher-bandwidth timing channels (cache-timing attacks, branch-prediction attacks, etc.). One way to protect this computation is to read *every* table entry, using conditional moves to replace irrelevant results with 0, but this is not as simple and fast as the original computation.

At a lower level, the implementor is faced with the problem of how to add two points. The Montgomery x-coordinate is not enough information to allow general additions.[9] The simplest solution is to use a different curve shape, "complete Edwards curves", which we published in [**10**] in 2007. A complete Edwards curve is a curve $x^2 + y^2 = 1 + dx^2y^2$ where d is non-square. The sum of two points (x_1, y_1) and (x_2, y_2) on the curve is

$$\left(\frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2}\right);$$

the denominators here are always nonzero. Various optimizations reduce the cost of each addition below 10 field multiplications.

The problem of general additions also appears in all of the signature systems described in Section 2. Complete Edwards curves are again the simplest solution. The Ed25519 signature system uses a complete Edwards curve for key generation, signing, and signature verification.

There is a slight speedup from switching to "dual" addition formulas, but these formulas have the disadvantage of not working for all inputs. Most of the cryptographic literature assumes that computations are performed correctly, and the occasional analyses of the consequences of incorrect computations often show that those computations can do tremendous damage. One therefore has to be very careful to have the implementation check for bad inputs (in constant time!) and handle them appropriately; or, alternatively, to prove that the higher-level scalar-multiplication strategy cannot produce bad inputs; or, alternatively, to prove that the higher-level scalar-multiplication strategy together with the randomization of inputs has negligible chance of producing bad inputs. Given the difficulty that this produces for auditors, together with the rarity of applications that will notice this slight speedup, we recommend that implementors avoid taking the "dual" approach to additions.

[8] [**13**, Section 4.2] reported that Microsoft's SChannel works this way.

[9] To be precise: Given $X_0(Q)$ and $X_0(R)$ one can, with some effort, compute all possibilities for $X_0(Q + R)$. Usually there are exactly two possibilities.

3.3. The NSA/NIST approach. FIPS 186-4 points to ANSI X9.62 for scalar-multiplication algorithms. ANSI X9.62 specifies[10] a single-scalar-multiplication algorithm (Appendix D.3.2) whose inner loop consists of

- a doubling,
- an addition if the scalar satisfies a particular property (probability 1/4), and
- a subtraction if the scalar satisfies another property (also probability 1/4).

The underlying addition algorithm (Appendix B.3) has five different cases.

Adding further complexity to the scalar-multiplication method, ANSI X9.62 cites "several variations of this method which can be used to speed up the computations". Even in applications where the original algorithm provides acceptable speed, the algorithm is obviously vastly more complex than the Montgomery ladder, and has many more traps for implementors. For example, random tests of this scalar-multiplication algorithm will exercise only two of the cases in the addition algorithm, so they won't protect an implementor who omits or mangles those cases. Furthermore, both levels of algorithms in ANSI X9.62 are full of timing leaks.

It is of course possible to build a correct constant-time implementation (and the literature explains various ways to make this somewhat less painful than it might seem at first, for example by reducing the number of cases). However, within the space of all implementations, a secure implementation is surrounded in all directions by traps for the implementor: implementations that are faster but leak information through timing, implementations that pass typical tests but that are wrong for some attacker-supplied inputs, etc. The problem here is much more severe than the problem described above for "dual" addition formulas: the problem before was a slight tension between speed and security, while the problem here is a much stronger tension between speed and security, combined with a strong tension between simplicity and security.

One of the core choices in ANSI X9.62 was to use what are called "Jacobian coordinates" to represent points inside these computations. (This means using (X, Y, Z) to represent a point $(x, y) = (X/Z^2, Y/Z^3)$ on a "short Weierstrass curve" $y^2 = x^3 + ax + b$.) This choice was publicly justified for speed reasons: IEEE P1363, a very similar standard developed in parallel by an overlapping team, claims that this choice provides "the fastest arithmetic on elliptic curves". However, all known methods for addition in Jacobian coordinates (and in the closely related "projective coordinates" for short Weierstrass curves), whether constant-time or not, are considerably slower and more complicated than constant-time addition on complete Edwards curves.

It is easy to explain why the standards don't mention Edwards curves: the standards were developed before Edwards curves were published. But it is much more difficult to explain why the standards don't mention the Montgomery ladder for single-scalar multiplication. The Montgomery ladder was published in

[10] The citations here are to the September 1998 draft. Subsequent drafts might have different material but aren't easy to find online and are unlikely to be checked by most implementors.

the 1980s, and has always been simpler, faster, and less error-prone than the techniques recommended in the standards. To justify ignoring the Montgomery ladder it seems that one would have to (1) focus purely on speed without regard to simplicity, and also (2) focus purely on signatures without regard to ECDH.

4 Coordinates sent through the network

Section 3 described Montgomery curves, complete Edwards curves, and short Weierstrass curves as separate types of curves. However, there are actually many easy maps between these types of curves, so the choice of coordinates sent through the network is not dictated by the choice of coordinates used inside computations.

For example, define $p = 2^{255} - 19$ and $A = 486662$, and recall from Section 3 that Curve25519 is the Montgomery curve $y^2 = x^3 + Ax^2 + x$ over $\mathbf{F}_p$. It is easy to see that if (x, y) is a point on this curve then $(x + A/3, y)$ is a point on the short Weierstrass curve $y^2 = x^3 + ax + b$ where $a = 1 - A^2/3$ and $b = 2A^3/27 - A/3$. It is just as easy to work backwards from $(x + A/3, y)$ to (x, y).

The map from (x, y) on the Montgomery curve to the corresponding point $(x + A/3, y)$ on the short Weierstrass curve is an "isomorphism" between curves, preserving addition of points. An implementor who wants to compute the nth multiple of a point Q on the short Weierstrass curve can

- invert this isomorphism (i.e., subtract $A/3$ from the first coordinate) to obtain the corresponding point on the Montgomery curve,
- use the Montgomery ladder to obtain the x-coordinate of the nth multiple of that point,
- recover the y-coordinate of the nth multiple, and
- apply the isomorphism to obtain nQ.

As this example illustrates, it's possible for a protocol to use (e.g.) short Weierstrass curves while computations use (e.g.) Montgomery curves.

As another example, Curve25519 is also isomorphic to the complete Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ over $\mathbf{F}_p$, where $d = 1 - 1/121666$. The Ed25519 signature system is specified in terms of this complete Edwards curve, while the X25519 ECDH system is specified in terms of the Montgomery curve. X25519 implementors who want faster key generation (see Section 3) use the Edwards curve for scalar multiplication and then apply the isomorphism to obtain a public key on Curve25519.

4.1. Invalid-curve attacks and twist security. Jager, Schwenk, and Somorovsky [31] recently announced a devastating break of "Java implementations using static EC keys" for TLS. Eight crypto libraries were analyzed, and it turned out that two of them (Bouncy Castle and Java Crypto Extension) didn't check whether incoming points (x, y) were on the curve that they were supposed to be on.

The attack works as follows. The attacker easily finds a point (x, y) of small prime order, let's say order 1009, on another curve over the same field, and sends that point to the server. The server blithely computes $k(x, y)$ where k is the server's secret key; this is the same as $(k \bmod 1009)(x, y)$. The attacker has a noticeable chance of correctly guessing $k \bmod 1009$, and confirms this guess by following normal protocol operations using the shared secret $(k \bmod 1009)(x, y)$, revealing $k \bmod 1009$. If the guess fails then the attacker tries again with another guess. By varying 1009 the attacker learns k modulo several small primes, and then easily computes k. The basic idea of this attack was published in 2000 by Biehl, Beyer, and Müller, as an elliptic-curve adaptation of an idea published by Lim and Lee.

NIST has frequently asserted that the NIST curves *when implemented properly* are not subject to any attack known to the public:

> We remain confident in their security and are not aware of any significant attacks on the NIST curves when used as described in our standards and implemented correctly.

It is of course true that the implementors could have avoided the attack by checking whether (x, y) is on the correct curve. The correct curve doesn't have any points of small order (except for orders dividing the "cofactor", something taken care of by standard "cofactor multiplication"). However, the reality is that 25% of the implementations in this study *didn't* avoid the attack.

The underlying issue is that checking whether input points are on the curve is considerable extra complexity. Typical tests won't notice if this check is omitted, and even highly aggressive tests aren't effective at figuring out whether this check is implemented correctly for *all* inputs.[11] It might sound trivial to have implementors check whether an input (x, y) is on the correct curve, but this is actually a significant tension between simplicity and security, and this tension is exactly what led to the success of the attack.

X25519 proactively avoids this attack as follows:

- Instead of sending short-Weierstrass (x, y) through the network, send Montgomery x. This choice of coordinates drastically limits the attacker's choice of curves: it does not quite force the attacker to send a point on the correct curve, but the only other possibility is that the attacker sends a point on what is called "the twist" of the curve.
- Encourage use of the Montgomery ladder. This has many benefits discussed in Section 3. What matters here is another effect, namely that scalar multiplication is computed correctly for all inputs x, both on the original curve and on the twist.
- Choose a curve so that the curve and the twist *each* have no points of small order, except for orders dividing the cofactor.

[11] One of the OpenSSL bugs announced this year was an error in the point-on-curve test for a very small fraction of inputs. It still isn't clear whether this error is exploitable. One can hope to eliminate all errors in ECC software through formal verification, and recent work shows considerable progress in this direction; this work is targeting X25519 implementations precisely because those implementations are so simple.

Sending the Edwards y, rather than the Montgomery x, would stop the attack for the same reason. This would slightly simplify implementations that use Edwards coordinates internally. However, choosing the Montgomery x slightly simplifies implementations that use the Montgomery ladder internally. These implementations are simpler in the first place, so the marginal extra simplification is more noticeable. For similar reasons, it is better to send the Montgomery x than the short-Weierstrass x.

In short, this tension between simplicity and security was and is avoidable through the choice of ECDH mechanisms. By blaming implementors for this attack, NIST has been refusing to acknowledge its own role in causing the problem.

4.2. Signatures and point compression. As noted in Section 3, the signature context is more complicated than the ECDH context. Signature verification uses general point additions; a single coordinate can support a ladder but can't support general point additions. It's possible to arrange signature verification as *addition verification*, which can be done with a single coordinate, but this interferes with speed and isn't particularly simple. Working with both the Edwards x and y coordinates is simpler.

In this two-coordinate context, ECC standards can and should require *compression* of the second coordinate. For example, Ed25519 sends the Edwards y and a single bit specifying x. This saves 32 bytes, sometimes producing a measurable speedup. Recovering x from the single bit sometimes produces a measurable slowdown, but the savings typically outweighs the slowdown. More importantly, instead of telling implementors to check the whole curve equation, Ed25519 is telling implementors to check an equation of the form $x^2 = s$. This is considerably less code, considerably reducing the tension between simplicity and security. It's also much more fault-tolerant than checking the curve equation, since it's checking n bits derived from n bits of attacker input rather than $2n$ bits of attacker input.

Today's ECC standards allow compression as an *option*, but this obviously isn't good enough to stop attacks, and it also ends up adding complexity for implementors.

4.3. Unified implementations of ECDH and signatures. The bigger picture of the ECDH+signature ecosystem is that some implementations support only ECDH, some implementations support only signatures, and some implementations support both. Using Edwards coordinates for ECDH would slightly simplify implementations of the third type, but using Montgomery coordinates for ECDH slightly simplifies implementations of the first type. Implementations of the first type are spectacularly simple (see Section 3), so slight simplifications are much more noticeable for them than for implementations of the other types. We therefore recommend Montgomery coordinates for ECDH, as in X25519.

Using the same prime for signatures and for ECDH is of course helpful for implementations of the third type. Using isomorphic curves (as X25519 and Ed25519 do) gives these implementors more options to share code between ECDH key generation, signing, etc.

5 Curves

Not all elliptic curves are compatible with the next-generation ECC options explained in previous sections. Specifically, if a curve has

- a unique point of order 2 and
- a point of order 4

then the curve supports

- complete Edwards coordinates and
- Montgomery coordinates with the $A^2 - 4$ non-squareness condition used in the proofs in [4].

It is very easy to find such curves: approximately 25% of all elliptic curves over large prime fields satisfy these conditions. However, one needs to search through many of these curves to find curves with small cofactors (a standard requirement). Furthermore, one needs to search through many curves with small cofactors to find curves whose *twists* also have acceptably small cofactors (see Section 4).

Taking a very small parameter d for an Edwards curve $x^2 + y^2 = 1 + dx^2y^2$ produces a small but measurable performance improvement compared to taking a random d. It has become common practice to take the *smallest* acceptable integer d; the performance justification here is miniscule, but taking any larger value of d would be hard to justify. This practice is comforting for people concerned about the flexibility available to the curve generator, as in Section 1. Before Edwards curves were known, analogous considerations dictated taking the smallest acceptable integer $(A - 2)/4$ for a Montgomery curve $y^2 = x^3 + Ax^2 + x$; this is how Curve25519 was generated.

6 Primes

The literature on multiprecision arithmetic—software for multiplying larger integers than the hardware is designed to multiply—is full of variable-time algorithms. Constant-time field arithmetic therefore requires special attention from software implementors.

6.1. Constant-time arithmetic. Constant-time software implementations are similar to hardware implementations: they allocate a constant number of bits for each integer, and always perform arithmetic on all bits, without skipping bits. For example:

- If the goal is to add a to b, where 255 bits are allocated for a and 255 bits are allocated for b: Allocate 256 bits for $a + b$. Of course, it's possible that $a + b$ would fit into 255 bits, but don't check.
- If the goal is to multiply a by b, where 256 bits are allocated for a and 256 bits are allocated for b: Allocate 512 bits for ab.

The reason that this strategy does not spiral out of control is that for an elliptic curve modulo p one is free to reduce modulo p at any moment. The details of this reduction depend on the choice of p.

6.2. Reduction modulo $2^{255} - 19$. Consider, for example, 600 bits allocated to hold an integer c modulo $p = 2^{255} - 19$. Replace c with $19q + r$, where $r = c \bmod 2^{255}$ and $q = \lfloor c/2^{255} \rfloor$, after allocating 350 bits for $19q + r$. It's easy to see that $19q + r$ is the same as c modulo p. Repeating the same idea reduces 350 bits to 256 bits, small enough for the next multiplication.

At the end of scalar multiplication it's important to *completely* reduce the output modulo p. This takes two iterations of constant-time conditional subtraction. One conditional subtraction, by definition, replaces c with $c - (1 - s)p$ where s is the sign bit in $c - p$.

6.3. Reduction modulo the P-256 prime. For comparison, the NIST P-256 prime p is $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. The ECDSA standard specifies the following reduction procedure given an integer "A less than p^2":

- Write A as the vector
$$(A_{15}, A_{14}, A_{13}, A_{12}, A_{11}, A_{10}, A_9, A_8, A_7, A_6, A_5, A_4, A_3, A_2, A_1, A_0),$$
meaning $\sum_i A_i 2^{32i}$.
- Define
$$\begin{aligned}
T &= (A_7, A_6, A_5, A_4, A_3, A_2, A_1, A_0);\\
S_1 &= (A_{15}, A_{14}, A_{13}, A_{12}, A_{11}, 0, 0, 0);\\
S_2 &= (0, A_{15}, A_{14}, A_{13}, A_{12}, 0, 0, 0);\\
S_3 &= (A_{15}, A_{14}, 0, 0, 0, A_{10}, A_9, A_8);\\
S_4 &= (A_8, A_{13}, A_{15}, A_{14}, A_{13}, A_{11}, A_{10}, A_9);\\
D_1 &= (A_{10}, A_8, 0, 0, 0, A_{13}, A_{12}, A_{11});\\
D_2 &= (A_{11}, A_9, 0, 0, A_{15}, A_{14}, A_{13}, A_{12});\\
D_3 &= (A_{12}, 0, A_{10}, A_9, A_8, A_{15}, A_{14}, A_{13});\\
D_4 &= (A_{13}, 0, A_{11}, A_{10}, A_9, 0, A_{15}, A_{14}).
\end{aligned}$$
- Compute $T + 2S_1 + 2S_2 + S_3 + S_4 - D_1 - D_2 - D_3 - D_4$.
- Reduce modulo p "by adding or subtracting a few copies" of p.

This is considerably more complicated, and considerably slower, than reduction modulo $2^{255} - 19$. Even worse, what does "a few copies" mean in the final step? This sounds like a loop, taking variable time, presumably a security problem.

With some effort one can see that the following constant-time procedure always works for the final step:

- Conditionally add $4p$.
- Conditionally add $2p$.
- Conditionally add p.
- Conditionally subtract $4p$.

- Conditionally subtract $2p$.
- Conditionally subtract p.

This is, however, quite slow. One might try to delay this procedure until the end of the computation, but then the next multiplication will violate the "A less than p^2" requirement. One can write down a revised reduction algorithm that works for larger values of A, but this adds even more complexity.

Even worse, this reduction procedure assumes that integers are expressed in radix 2^{32}, but the literature clearly shows that 2^{32} is not the best radix for additions and multiplications on most platforms. Implementors who try a different radix for faster additions and multiplications will need to insert many bit shifts and extractions into the NIST P-256 reduction procedure, incurring further complexity and cost.

6.4. A few good primes. The literature contains cost analyses, at various levels of detail, of field arithmetic on many different platforms modulo many different primes. These analyses strongly suggest that a few primes provide particularly good cross-platform performance for their size.

We are not aware of any reason to allow a prime that isn't Pareto-optimal: a prime for which it's possible to gain cross-platform performance by switching to another prime that's at the same (or higher) security level. We suggest being even more restrictive than this, and excluding any prime for which it's possible to gain cross-platform performance by switching to another prime that's within 1 bit of the same size (or larger). Small differences in security aren't meaningful: if someone has enough computer power to break a prime, it doesn't make any sense to respond by switching to another prime that's just one bit larger. This rule still leaves some particularly fast primes, such as $2^{255} - 19$ and $2^{521} - 1$.

7 "Digital signature schemes": questions from NIST

7.1. "Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?"

No. See above.

7.2. "Are there other digital signature schemes that should be considered for inclusion in a future revision to FIPS 186? What are the advantages of these schemes over the existing schemes in FIPS 186?"

Yes. See above, particularly Section 2.

8 "Security of elliptic curves": questions from NIST

8.1. "Do the NIST-recommended curves satisfy the security requirements of applications used by industry?"

No. See above.

8.2. "Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?"

Yes. See above.

9 "Elliptic curve specifications and criteria": questions from NIST

9.1. "Is there a need for new elliptic curves to be considered for standardization?"

Presumably "standardization" here means standardization by NIST, rather than, e.g., standardization by IETF or de-facto standardization by the community. The answer still depends on what NIST means by "need" and by "new".

Next-generation ECC has already been widely deployed. Nicolai Brown maintains public lists[12] at http://ianix.com of "Things that use Curve25519" and "Things that use Ed25519". The lists include, along with many other examples,

- Apple's iOS (iPhone, iPad, etc.) operating system;
- the TextSecure (Signal) messaging system;
- the standard OpenSSH remote-login software;
- the Tor network; and
- Google's QUIC protocol.

Our impression is that NIST P-256, the most popular NIST curve, has now been surpassed by Curve25519 in a wide range of usage metrics.

The reasons for users selecting Curve25519 over NIST P-256 have been amply documented. Maybe these reasons aren't relevant to NIST's "needs"; we haven't seen a definition of NIST's "needs". As for "new", there are clearly large differences between "new to NIST's ECC standards", "new to real-world ECC deployment", and "newly developed".

9.2. "If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?"

There was extensive discussion of curve criteria on the CFRG mailing list in 2014 and 2015. NIST may find this discussion useful as a starting point.

Obviously all proposed elliptic curves must resist all publicly known ECDLP attacks. The specified base point on the curve must have large enough prime order to resist rho attacks; must be immune to additive transfers; and must have large enough embedding degree to resist multiplicative transfers.

There might be submissions of curves that (like NIST K-163) are clearly *feasible* but *expensive* to break. Such submissions will force NIST to quantify its minimum acceptable security level. The first author has a new blog post "Break a dozen secret keys, get a million more for free" http://blog.cr.yp.to/20151120-batchattacks.html pointing out that NIST's previous evaluations of

[12] We incorporate those lists into these comments by reference.

quantitative security levels have failed to take into account the damage done by *batch* attacks.[13]

In general the modern trend is towards conservative prime-field ECC. Note that asymptotically 100% of all curves over all finite fields, counted in the usual way, are curves over prime fields; in other words, curves over non-prime fields are, mathematically, extremely rare. The advent of "Weil descent" broke ECDLP for many curves over non-prime fields, and the limits of Weil descent are still not entirely clear, as illustrated by ongoing debates regarding whether Weil descent takes subexponential time to break ECDLP for all binary fields. There are interesting hardware performance advantages to binary curves (especially Koblitz curves), and NIST's current binary curves are clearly not broken by the best Weil-descent attacks known today, on the other hand, securely implementing binary-field arithmetic in *software* is considerably more difficult and error-prone than implementing prime-field arithmetic, and the extra structure of these curves makes security analysis more difficult and less confidence-inspiring.

There might also be, again for interesting performance reasons, proposals of genus-2 hyperelliptic curves; special curves over prime fields with extra "endomorphisms" (GLV curves); special curves over quadratic extension fields with extra endomorphisms ($\mathbf{Q}$-curves, including GLS curves); etc. We believe that considering any of these curves will be a distraction from fixing the problems with NIST's current ECC standards. We specifically recommend that NIST disregard any GLV/GLS/$\mathbf{Q}$-curve proposals: the GLV patents are still valid and cover all use of extra endomorphisms to speed up scalar multiplication, so these curves will simply end up adding worrisome structure without any noticeable benefits in speed and of course without any benefits in simplicity.

We strongly encourage NIST to look beyond the narrow question of ECDLP security and consider the security of ECC *implementations*, as in our web page `http://safecurves.cr.yp.to`.[14] This means paying attention to the security impact of implementors pursuing simplicity, the security impact of implementors pursuing speed, risks of various types of errors in implementations, etc.

To take into account the big picture of simplicity, security, and speed, we recommend that NIST adopt the following statement of principles:

- Principle 1: We want speed and simplicity (and speed-simplicity combinations) for secure implementations.
- Principle 2: We want to avoid speed incentives and simplicity incentives towards insecure implementations.
- "Secure" includes the following properties: an implementation works correctly for all valid inputs; avoids compromising security when inputs are invalid; avoids all data flow from secrets to timing; etc.
- "Speed" includes speed on many different platforms: e.g., speed on 64-bit desktop CPUs, speed on 32-bit smartphone CPUs, speed on embedded microcontrollers, and speed of hardware implementations.

[13] We incorporate that blog post into these comments by reference.

[14] We incorporate the SafeCurves web site into these comments by reference.

- "Speed" includes speed of many important ECC operations: e.g., key generation, signing, verification, and DH shared-secret computation. "Simplicity" includes the simplicity of implementations supporting all of these operations, the simplicity of implementations supporting only DH, the simplicity of implementations supporting only signatures, etc.
- "Simplicity" includes the simplicity of new implementations, and the simplicity of modifying preexisting implementations.

More specific requirements can be derived from these general principles. For example, SafeCurves requires twist-secure curves supporting (1) simple, fast, complete, constant-time single-coordinate single-scalar multiplication and (2) simple, fast, complete, constant-time multi-scalar multiplication. "Fast" means that implementations of scalar multiplication for the same curve cannot be much faster, and "simple" means that reasonably fast implementations of scalar multiplication for the same curve cannot be much more concise. Our analysis indicates that the details of these requirements are forced by Principle 2, avoiding incentives towards insecure implementations.

When specific requirements (or statements of desiderata) are in conflict with these general simplicity/security/speed principles, the requirements should be corrected. Consider, for example, the following claim from NSA in NIST's curve standard: "For efficiency reasons, it is desirable to take the cofactor to be as small as possible." All of the NIST prime-field curves have cofactor 1. However, this extreme cofactor requirement actually produces a slowdown: the NIST curves are considerably slower than Edwards curves at similar (or even somewhat higher) security levels, despite the fact that Edwards curves always have cofactor at least 4. For DH this slowdown was already clear from the literature predating NSA's claim. We do not see any reason for violating the cofactor limits stated in FIPS 186-4 (at least 2^{10}, depending on the size of p), but we also do not see any justification for those specific limits.

9.3. "Do you anticipate a need to create, standardize or approve new elliptic curves on an ongoing basis?"

No (despite the aforementioned caveats regarding NIST's "needs"). Replacing "ECDLP security" with "ECC security" as the goal is a one-time correction.

10 "Adoption": questions from NIST

10.1. "Which of the approved digital signature schemes and NIST-recommended curves have been used in practice?"

For a typical network packet today, the answer is "none". NIST's ECC standards are not being used to protect that packet against espionage, forgery, and sabotage. Either the packet is protected by a different (probably lower-security) mechanism, or it isn't protected at all.

We believe that all network packets should be protected by strong cryptography, and that ECC is the community's best hope for getting this done in the near future.[15] The packets that are *not* protected by ECC today should be viewed by NIST as the most important target for any new ECC standards.

This target is not accurately reflected by the corner case of people who are happily using NIST's ECC standards today. Data regarding this corner case is of some interest but should not be misunderstood as data regarding future use of ECC.

10.2. "Which elliptic curves are accepted for use in international markets?"

The international situation is not very different from the U.S. situation. A few countries (France, Germany, China) are encouraging use of their locally developed curves, but they seem to have had only limited success. See also Section 11.3.

11 "Interoperability": questions from NIST

11.1. "If new curves were to be standardized, what would be the impact of changing existing implementations to allow for the new curves?"

The answer depends on the curve.

There are many freely available implementations of Curve25519. TweetNaCl, for example, is a complete self-contained portable cryptographic library that fits into 100 tweets, and implements the most important Curve25519 use cases (X25519 and Ed25519). There are many other implementations of Curve25519 optimized for many different platforms, supporting other languages, etc., and in most cases supporting the same well-known easy-to-use API, with extensive tests provided by the SUPERCOP benchmarking framework. Various Curve25519 implementations are also the targets of state-of-the-art ECC software verification. People can, should, and do simply use these implementations.

It would not be very difficult to build up a similar ecosystem of implementations of a higher-security next-generation curve, such as E-521. See our paper [6] with Chuengsatiansup for several reasons that a higher-security curve might be of interest.

Old-fashioned curves are considerably more difficult to implement, and much more likely to produce low-quality (complex, slow, breakable) implementations.

[15] It has been well known since the 1990s that quantum computers break ECC. There are proposals for post-quantum systems, and there will obviously be more and more users who deploy such proposals. However, this deployment is not a serious argument against deploying ECC: for the foreseeable future this deployment will be done as a *supplement* to ECC, not as a *replacement* for it. Most post-quantum public-key systems have not been thoroughly studied, so users want ECC as a backup to provide some security in case of catastrophic failure. There are some conservative post-quantum public-key systems, such as the systems recently recommended by the European PQCRYPTO consortium (see `http://pqcrypto.eu.org/docs/initial-recommendations.pdf`), but those systems are expensive enough that users who can afford them have very little motivation to turn off ECC.

History shows that the lowest-quality ECC implementations are "generic" implementations that try to handle many different curves[16] with a single code base. OpenSSL's "generic" ECC implementation is complex, slow, hard to audit, likely to be buggy, and likely to be breakable by timing attacks; OpenSSL has improved code quality by adding a separate NIST P-256 implementation, a separate NIST P-384 implementation, etc.

To the extent that code size is a problem (for auditing, for verification, for small devices, etc.), the obvious solution to the problem is to prune the list of supported curves. This solution is much more effective, and produces higher-quality results, than trying to merge implementations of many curves into a single "generic" implementation.

For someone who really wants to merge an implementation of curve C into an existing implementation of curve B, the exact implementation difficulty depends on B, C, and many details of the existing implementation. For example, if a NIST-P-curve implementation assumes curves of the form $y^2 = x^3 - 3x + b$ in Weierstrass coordinates, then generalizing it to $y^2 = x^3 + ax + b$ will typically require rearranging some field-arithmetic calls, and adapting it to Edwards coordinates will typically require some extra lines of code to convert from Edwards coordinates to Weierstrass coordinates and back. Of course, when evaluating existing implementations of the NIST curves, NIST should also consider the question of whether those implementations are secure and whether those implementations are successfully protecting typical users. Many existing implementations should simply be removed and replaced by new implementations of better curves.

11.2. "What is the impact of having several standardized curves on interoperability?"

A client's ECDH implementation cannot interoperate with a server's ECDH implementation unless there is at least *one* curve that is supported by both the client and the server. Furthermore, the client and server need to be able to securely *figure out* which curve that is, and exchange keys using that curve. Similar comments apply to other ECC protocols, such as signatures.

The simplest way for a protocol to guarantee interoperability is to specify a single curve to be used by all clients and all servers. One generalization that guarantees interoperability is to specify a set of curves to be supported by all clients; each server can then make its own curve choice (and securely authenticate this choice along with authenticating the server's ECC key). A different generalization that guarantees interoperability is to specify a set of curves to be supported by all *servers*; each *client* can then make its own curve choice. Note that these generalizations cause obvious damage to *implementation simplicity* and to *performance*, especially in situations where a client sends a key to a server whose curve choice isn't known to the client, or vice versa.

[16] These "generic" implementations actually have various restrictions on the set of curves actually supported. Should an implementation be called "generic" if it can't handle binary fields? Quadratic extension fields? Elliptic curves that aren't of the form $y^2 = x^3 - 3x + b$? Curves beyond genus 1?

Some protocols don't actually guarantee ECC interoperability, even when both the client and the server support ECC. For example, TLS seems to have standardized a surprisingly large number of curves without thinking these issues through. Because of the same issues of implementation simplicity and performance, each widely used TLS implementation supports only a subset of the curves, and nothing in the TLS protocol guarantees that these subsets will overlap. If a certificate authority issues ECC certificates using a Brainpool curve or one of the more obscure NIST curves, will a client understand the certificates? The main reason that "ECDHE" works in TLS today is that implementors have collectively decided to support NIST P-256, in effect taking the simple approach to guaranteeing interoperability.

To the extent that the profusion of standardized curves causes interoperability problems, one can and should blame the standards for encouraging these problems. Does NIST think that having a long list of standardized curves is a *good* thing? For each of its existing curves, and for any new curves that are proposed, NIST's *default* assumption should be that having the curve standardized is a bad idea.

Obviously this default can, and occasionally should, be overridden. NIST ECC is failing quite disastrously in practice, in ways that are fixed by next-generation ECC, and there is already widespread adoption of next-generation ECC. But the question for any particular curve shouldn't be "Does standardizing this curve have a benefit?"; it should be "Does standardizing this curve have a large enough benefit to outweigh the costs?"

11.3. "What are the advantages or disadvantages of allowing users or applications to generate their own elliptic curves, instead of using standardized curves?"

Computing n independent b-bit discrete logarithms on one properly chosen curve costs roughly $2^{b/2}\sqrt{n}$ additions, while computing n independent b-bit discrete logarithms on n independent properly chosen curves costs roughly $2^{b/2}n$ additions (by the best attack algorithms known today). This means that there is a quantitative security advantage in having each key on its own curve.

However, taking a single curve with a somewhat larger b produces a much larger quantitative security advantage at much lower cost. We recommend taking b large enough that foreseeable attackers (pre-quantum) will be limited to $2^{b/2}\epsilon$ additions for a small value of ϵ. The attacker's chance of finding even *one* of the n discrete logarithms is then limited to approximately ϵ^2. This strategy was used in the design of Curve25519, and all of these security issues were described in the Curve25519 paper.

Regarding cost: As long as *primes* are standardized, allowing random curve parameters does not produce a huge slowdown in curve arithmetic. However, constantly changing curves means constantly transmitting new curve parameters, which is a noticeable cost in bandwidth. More importantly, it means constantly generating new curves, which is not only a significant expense in CPU time but also a nightmarish increase in complexity.

Of course, a user who generates his own elliptic curves is protected against the possibility of a malicious curve generator choosing the weakest curve from the same pool, using a weakness not known to the public. However, this is merely a *possibility*, while there are *definite* problems with excessive complexity (and slowness), so it seems unwise to try to eliminate this possibility at the expense of a drastic increase in complexity.

Many ECC-based protocols (e.g., group key exchange) require multiple users to share a curve, raising the question of who will be trusted to generate the curve. Multiparty computation is a theoretical answer but adds even more complexity.

More moderate curve pools (one curve per application, one curve per country, etc.) would practically eliminate the costs in CPU time and bandwidth, but would also practically eliminate the security advantages. The huge complexity disadvantage would remain.

12 "Performance": questions from NIST

12.1. "Do the performance characteristics of existing implementations of the digital signatures schemes approved in FIPS 186-4 meet the requirements of applications used by industry?"

This depends on the application. Note that this question is another example of how NIST can get a quite wrong picture by listening too much to people who are happily using the NIST curves today, and not thinking enough about the much larger set of people who *should* be using ECC.

Many applications can afford huge elliptic curves and can afford slow implementations. Often one or two ECC operations are used to protect a very long message (or a message requiring extensive non-ECC computations for other reasons). Often CPUs are idle, or are so busy with something other than ECC that the cost of ECC is negligible.

On the other hand, sometimes ECC has a serious impact on cost. Consider, for example, the cost of protecting against denial of service when a busy Internet server is handling many packets per second from different sources. An attacker can try to flood the network, but an attacker can also try to flood the CPU with new public keys to process. If the CPU cannot keep up with the cost of public-key cryptography then the server will have to drop traffic from new *legitimate* keys. As another example, deployment of public-key cryptography in a wide variety of small devices relies critically on being able to make ECC fast enough and small enough.

On an Intel Sandy Bridge CPU (common in data centers today), OpenSSL 1.0.2's optimized NIST P-526 implementation takes 310000 cycles for ECDH shared-secret computation and 430000 cycles for ECDSA signature verification, while Tung Chou's software takes only 160000 cycles for X25519 shared-secret computation and only 206000 cycles for Ed25519 signature verification. This is just one example of how Curve25519 "over the past ten years has set speed records for conservative ECC on many different platforms, using implementations from 23 authors", as stated in [5, Section 7]. See [4] (original paper for various 32-bit CPUs), [18] (Core 2, Athlon 64), [15] (Cell), [8] (more Intel CPUs), [11] (NEON), [23] (more Intel CPUs), [24] (GPUs), [30] (FPGAs), [14] (more Intel CPUs), [16] (microcontrollers), and [21] (ASICs).

13 "Intellectual property": questions from NIST

13.1. "What are the desired intellectual property requirements for any new curves or schemes that could potentially be included in the Standard?"

Most clients and servers for Internet communication (web, email, chat, collaboration, etc.) are distributed and used for free, without payment of license fees. This has proven to be a productive environment for rapid deployment of patent-free cryptography, while patented cryptography is simply not considered.

As far as we know, the most important ECC patents have all expired, and the entire core of ECC is now patent-free. Patent 4995082 (Schnorr) expired in 2008. Patents 5159632, 5271061, and 5463690 (the Crandall patents on "shift and add" ECC primes) expired in 2011. Patent 6141420 (point compression) expired in 2014. Patents 5299262 and 5999627 (fixed-base scalar multiplication by Brickell–Gordon–McCurley and Lim–Lee) expired in 2012 and 2015 respectively. We have not found any current patents covering, e.g., X25519 (ECDH with Curve25519) or Ed25519 signatures.

Some fringes of ECC are still patented. The most interesting remaining patents are the GLV patents; these patents cover the use of endomorphisms to speed up ECC on special curves (GLV curves, GLS curves, $\mathbf{Q}$-curves, and so on). However, fixing the problems with NIST's ECC standards does not require going near these fringes.

To summarize, focusing on patent-free ECC obviously has important advantages, and there is no evidence of any important disadvantages.

13.2. "What impact has intellectual property concerns had on the adoption of elliptic curve cryptography?"

Certicom's web page asserts that Certicom owns "over 450 patents and patents pending worldwide covering key aspects of Elliptic Curve Cryptography (ECC)". A Wikipedia page, highlighting Certicom and citing an incident from a decade ago, asserts that "Patent-related uncertainty around elliptic curve cryptography (ECC), or ECC patents, is one of the main factors limiting its wide acceptance."

"Over 450 patents and patents pending worldwide" might sound like a large minefield, obviously difficult for anyone to review. In fact, Certicom has systematically inflated this number, by splitting each of its ideas into many separate patent applications. For example, Certicom's Dual EC patent application was split across at least four jurisdictions: Canada (CA 2594670), Europe (EP 06704329), Japan (JP 5147412), and the United States. In the United States it was split into patent 8396213 and patent application 2013/0170642, with the possibility of further splits. As another example, Certicom repeatedly split its

GLV patent application, producing a quite noticeable fraction of all of Certicom's US patent applications. The number of different ideas actually patented by Certicom in the US is vastly smaller than 450, and it seems unlikely that Certicom has significantly more applications (or significantly different applications) in other jurisdictions.

It is reasonably clear that, as a historical matter, Certicom was successful in spreading fear, uncertainty, and doubt regarding its patents, drastically limiting deployment of ECC for many years. However, today the general consensus is that ECC is safe to deploy without patent fees. Our impression is that public perception was significantly affected by the publication of RFC 6090 in 2011: this RFC was carefully limited to sources so old that they were obviously not patented,[17] and nevertheless described a usable form of ECC similar to NIST's ECC standards. We are not aware of any contributions of Certicom to the forms of ECC that we recommend today.

References

[1] — (no editor), *Proceedings of the 23rd USENIX security symposium, August 20–22, 2014, San Diego, CA, USA*, USENIX, 2014. See [13].

[2] Vijay Atluri, Claudia Daz (editors), *Computer security—ESORICS 2011—16th European symposium on research in computer security, Leuven, Belgium, September 12–14, 2011, proceedings*, Lecture Notes in Computer Science, 6879, Springer, 2011. ISBN 978-3-642-23821-5. See [12].

[3] Lejla Batina, Matthew Robshaw (editors), *Cryptographic hardware and embedded systems—CHES 2014—16th international workshop, Busan, South Korea, September 23–26, 2014, proceedings*, Lecture Notes in Computer Science, 8731, Springer, 2014. ISBN 978-3-662-44708-6. See [6].

[4] Daniel J. Bernstein, *Curve25519: new Diffie-Hellman speed records*, in PKC 2006 [**33**] (2006), 207–228. URL: http://cr.yp.to/papers.html#curve25519. Citations in this document: §3.1, §3.1, §7, §5, §12.1.

[5] Daniel J. Bernstein, Tung Chou, Chitchanok Chuengsatiansup, Andreas Hülsing, Eran Lambooij, Tanja Lange, Ruben Niederhagen, Christine van Vredendaal, *How to manipulate curve standards: a white paper for the black hat*, in SSR 2015 (2015). URL: http://bada55.cr.yp.to/. Citations in this document: §1.1, §12.1.

[6] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, *Curve41417: Karatsuba revisited*, in CHES 2014 [**3**] (2014), 316–334. Citations in this document: §11.1.

[7] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, Bo-Yin Yang, *High-speed high-security signatures*, in CHES 2011 [**27**] (2011), 124–142; see also newer version [**8**]. URL: https://eprint.iacr.org/2011/368.

[8] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, Bo-Yin Yang, *High-speed high-security signatures*, Journal of Cryptographic Engineering **2** (2012), 77–89; see also older version [**7**]. URL: https://eprint.iacr.org/2011/368. Citations in this document: §2, §12.1.

[9] Daniel J. Bernstein, Simon Josefsson, Tanja Lange, Peter Schwabe, Bo-Yin Yang, *EdDSA for more curves* (2015). URL: https://eprint.iacr.org/2015/677. Citations in this document: §2.

[17] We suggest labeling dates "B.C." for "Before Certicom".

[10] Daniel J. Bernstein, Tanja Lange, *Faster addition and doubling on elliptic curves*, in Asiacrypt 2007 [**22**] (2007), 29–50. URL: `https://eprint.iacr.org/2007/286`. Citations in this document: §3.2.

[11] Daniel J. Bernstein, Peter Schwabe, *NEON crypto*, in CHES 2012 [**28**] (2012), 320–339. URL: `http://cr.yp.to/papers.html#neoncrypto`. Citations in this document: §12.1.

[12] Billy Bob Brumley, Nicola Tuveri, *Remote timing attacks are still practical*, in ESORICS 2011 [**2**] (2011), 355-371. URL: `https://eprint.iacr.org/2011/232`. Citations in this document: §3.1, §3.1.

[13] Stephen Checkoway, Matthew Fredrikson, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, *On the practical exploitability of Dual EC in TLS implementations*, in USENIX Security 2014 [**1**] (2014). URL: `https://projectbullrun.org/dual-ec/index.html`. Citations in this document: §8.

[14] Tung Chou, *Sandy2x: fastest Curve25519 implementation ever*, in SAC 2015 (2015). URL: `https://www.win.tue.nl/~tchou/papers/sandy2x.pdf`. Citations in this document: §12.1.

[15] Neil Costigan, Peter Schwabe, *Fast elliptic-curve cryptography on the Cell Broadband Engine*, in Africacrypt 2009 [**26**] (2009), 368–385. URL: `http://cryptojedi.org/users/peter/#celldh`. Citations in this document: §12.1.

[16] Michael Düll, Björn Haase, Gesine Hinterwälder, Michael Hutter, Christof Paar, Ana Helena Sánchez, Peter Schwabe, *High-speed Curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers*, to appear, Designs, Codes and Cryptography (2015). URL: `http://link.springer.com/article/10.1007/s10623-015-0087-1/fulltext.html`. Citations in this document: §12.1.

[17] Pierrick Gaudry, *Variants of the Montgomery form based on Theta functions* (2006); see also newer version [**18**]. URL: `http://www.loria.fr/~gaudry/publis/toronto.pdf`.

[18] Pierrick Gaudry, *Fast genus 2 arithmetic based on Theta functions*, Journal of Mathematical Cryptology **1** (2007), 243–265; see also older version [**17**]. URL: `http://webloria.loria.fr/~gaudry/publis/arithKsurf.pdf`. Citations in this document: §12.1.

[19] Diana Goehringer, Marco Domenico Santambrogio, João M. P. Cardoso, Koen Bertels (editors), *Reconfigurable computing: architectures, tools, and applications—10th international symposium, ARC 2014, Vilamoura, Portugal, April 14–16, 2014, proceedings* (2014). ISBN 978-3-319-05959-4. See [30].

[20] Tim Güneysu, Helena Handschuh (editors), *Cryptographic hardware and embedded systems—CHES 2015—17th international workshop, Saint-Malo, France, September 13–16, 2015, proceedings*, Lecture Notes in Computer Science, 9293, Springer, 2015. ISBN 978-3-662-48323-7. See [21].

[21] Michael Hutter, Jürgen Schilling, Peter Schwabe, Wolfgang Wieser, *NaCl's* `crypto_box` *in hardware*, in CHES 2015 [**20**] (2015), 81–101. Citations in this document: §12.1.

[22] Kaoru Kurosawa (editor), *Advances in cryptology—ASIACRYPT 2007, 13th international conference on the theory and application of cryptology and information security, Kuching, Malaysia, December 2–6, 2007, proceedings*, Lecture Notes in Computer Science, 4833, Springer, 2007. ISBN 978-3-540-76899-9. See [10].

[23] Adam Langley, Andrew Moon, *Implementations of a fast elliptic-curve Digital Signature Algorithm* (2013). URL: `https://github.com/floodyberry/ed25519-donna`. Citations in this document: §12.1.

[24] Eric M. Mahé, Jean-Marie Chauvet, *Fast GPGPU-based elliptic curve scalar multiplication* (2014). URL: https://eprint.iacr.org/2014/198.pdf. Citations in this document: §12.1.

[25] Victor S. Miller, *Use of elliptic curves in cryptography*, in Crypto 1986 [**32**] (1986), 417–426. MR 88b:68040. Citations in this document: §3.1.

[26] Bart Preneel (editor), *Progress in cryptology—AFRICACRYPT 2009, second international conference on cryptology in Africa, Gammarth, Tunisia, June 21–25, 2009, proceedings*, Lecture Notes in Computer Science, 5580, Springer, 2009. See [15].

[27] Bart Preneel, Tsuyoshi Takagi (editors), *Cryptographic hardware and embedded systems—CHES 2011, 13th international workshop, Nara, Japan, September 28–October 1, 2011, proceedings*, Lecture Notes in Computer Science, 6917, Springer, 2011. ISBN 978-3-642-23950-2. See [7].

[28] Emmanuel Prouff, Patrick Schaumont (editors), *Cryptographic hardware and embedded systems—CHES 2012—14th international workshop, Leuven, Belgium, September 9–12, 2012, proceedings*, Lecture Notes in Computer Science, 7428, Springer, 2012. ISBN 978-3-642-33026-1. See [11].

[29] Ronald L. Rivest, *Responses to NIST's proposal*, Communications of the ACM **35** (1992), 41–54. URL: https://people.csail.mit.edu/rivest/pubs/RHAL92.pdf. Citations in this document: §1.

[30] Pascal Sasdrich, Tim Güneysu, *Efficient elliptic-curve cryptography Using Curve25519 on reconfigurable devices*, in ARC 2014 [**19**] (2014), 25–36. URL: https://www.hgi.rub.de/media/sh/veroeffentlichungen/2014/03/25/paper_arc14_curve25519.pdf. Citations in this document: §12.1.

[31] Tibor Jager, Jörg Schwenk, Juraj Somorovsky, *Practical invalid curve attacks on TLS-ECDH*, in ESORICS 2015 (2015). Citations in this document: §4.1.

[32] Hugh C. Williams (editor), *Advances in cryptology: CRYPTO '85*, Lecture Notes in Computer Science, 218, Springer, Berlin, 1986. ISBN 3–540–16463–4. MR 87d:94002. See [25].

[33] Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, Tal Malkin (editors), *Public key cryptography—9th international conference on theory and practice in public-key cryptography, New York, NY, USA, April 24–26, 2006, proceedings*, Lecture Notes in Computer Science, 3958, Springer, 2006. ISBN 978-3-540-33851-2. See [4].

Comments from: Bundesamt für Sicherheit in der Informationstechnik (BSI)

General remarks.

In June 2015, NIST hosted a technical workshop on Elliptic Curve Cryptography Standards. At the workshop BSI presented two papers. The first one [1] lays out Requirements for Elliptic Curves for High Security Applications. The second one [2] describes weaknesses of blinding measures when applied to curves over special prime fields.

After the workshop BSI published another paper [3] showing that the use of so-called „Twist-Secure“ curves does not necessarily improve the security of Elliptic Curve based cryptographic schemes.

A fourth paper [4] (in preparation) will describe vulnerabilities of purely deterministic signature schemes.

[1]-[4] describe the position of BSI on the use of ECC and on the choice of appropriate curves. [1] is an extended version of [5], which was endorsed by large parts of the german ECC community. The position expressed in [1] on the use of prime order groups in Weierstrass form is also strongly supported and justified by [6].

The technical guideline TR-02102 [7] is updated at least annually. It lists the cryptographic algorithms and parameters that are considered secure by BSI.

After the NIST ECC workshop the US government changed its policy on the use of ECC [8] due to the threats posed by quantum computers. [8] advises against the use of 256 bit curves for newly fielded systems.

[8] does, however, neither give a timeline nor a migration strategy to QC-resistant algorithms. One conclusion one could draw from [8] is that it is currently not adequate to introduce major changes into existing systems and standards that may soon become „legacy“. Instead, selection, standardisation and implementation of QC-resistant algorithms could become the top priority.

The following answers to NIST's questions focus on ECC.

Answers to NIST's questions

1. Digital Signature Schemes

a. Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?

For new applications of ECC the key sizes should preferably be 384 bit or more. 256 bit are needed for applications with less stringent security requirements and for interoperability purposes.

b. Are there other digital signature schemes that should be considered for inclusion in a future revision to FIPS 186? What are the advantages of these schemes over the existing schemes in FIPS 186?

The Elliptic Curve based Schnorr signature scheme could be considered for inclusion.

Schnorr signatures are easier to implement and seem to have better security proofs than ECDSA. For ECDSA it would be desirable to have a stronger binding between signature, curve parameters, random ephemeral point and the signer's identity.

No purely deterministic schemes should be added to FIPS 186.

2. Security of Elliptic Curves

a. Do the NIST-recommended curves satisfy the security requirements of applications used by industry?

The security requirements of industry differ, depending e.g. on the environment their products are used in and the implementation form (hardware, software). For implementations in hardware used in an unprotected environment the NIST-recommended curves over random prime fields with a length of 256 or more bits meet the current security requirements best. See also 1.a.

NIST-recommended curves with shorter bit lengths could be made obsolete.

b. Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?

There is no such algorithm known to BSI.

3. Elliptic Curve Specifications and Criteria

a. Is there a need for new elliptic curves to be considered for standardization?

BSI favors flexible and agile architectures, therefore BSI welcomes the introduction of provably randomly generated curves into FIPS 186. In particular BSI proposes to add the Brainpool Curves of bitlength 256 or higher, which were standardised in RFC 5639 [9], to FIPS 186. These curves are of widespread use (e.g. for machine readable travel documents) and satisfy all security requrements..

b.If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?

[1] and [9] motivate and sum up the main security criteria that should be met. In particular, BSI does not consider „Twist Security“ as necessary. In addition verifiably random curves of prime order defined over random prime fields in Weierstrass form are preferred.

c.Do you anticipate a need to create, standardize or approve new elliptic curves on an ongoing basis?

This depends on the migration plan to QC-resistant algorithms. If NIST intends to quickly move to QC-resistant solutions no ongoing curve generation is needed.

4. Adoption

a. Which of the approved digital signature schemes and NIST-recommended curves have been used in practice?

We mainly see implementations of ECDSA, the equivalent algorithm ECGDSA and RSA.

b. Which elliptic curves are accepted for use in international markets?

This question touches regulatory aspects because some curves are made mandatory by national regulations. In certifications according to the Common Criteria we observe that most products support the NIST P256/384/512 and the Brainpool P256/384/512 curves. Also the ANSSI curves are of growing significance.

5. Interoperability

a. If new curves were to be standardized, what would be the impact of changing existing implementations to allow for the new curves?

The impact depends on whether curve representation and transmission format are kept or not. Weierstrass form should be used, other representations may be used for internal use in products using ECC.

A general objective when designing cryptographic products is agility. For well designed products the impact of changing to new curves is considered manageable. There may be problems with products optimized for one fixed curve or using a hard-wired curve.

b. What is the impact of having several standardized curves on interoperability?

This depends on the cryptographic service provided by these curves. For signatures it means that more root certificates may have to be supported or that certificate chains may include several different curves.

c. What are the advantages or disadvantages of allowing users or applications to generate their own elliptic curves, instead of using standardized curves?

This depends on the general security of ECC. If there were a small subclass of weak curves it would be better to use many different curves in order to avoid the standardisation of a weak curve.

However, generation of secure curves is not easy and involves factoring large numbers. In addition, the communication partners will have to check the security of curves offered by their counterparts. This can be facilitated by using security certificates (e.g. as described in RFC 5639 [9]) but remains costly. The cost of this verification may also depend on the curve generation method. If the generation method chooses curves that are minimal with respect to some property one would also have to prove this minimality property to the user.

In addition, an widely accepted set of security criteria for elliptic curves would still be needed.

6. Performance

a. Do the performance characteristics of existing implementations of the digital signatures schemes approved in FIPS 186-4 meet the requirements of applications used by industry?

We see performance as a combination of different factors, e.g. speed, cryptographic strength, SCA-resistance, energy consumption, and lifecylce cost. Currently, there are products available that meet the requirements of BSI. From the perspective of side-channel security curves over random prime fields perform slightly better.

7. Intellectual Property

a. What are the desired intellectual property requirements for any new curves or schemes *that could potentially be included in the Standard?*

The term „New Curves“ could have two meanigs: „Newly generated“ or „Already existing but added to FIPS 186“. In both cases there should be no IPR on the curves or on methods to improve the performance (e.g. speed and implementation security) of these curves that are specific to these curves.

b. What impact has intellectual property concerns had on the adoption of elliptic curve cryptography?

From what we could observe the discussion on IPR has significantly slowed down the adoption of ECC.

Additional remarks.

In the light of [8] it might be sensible to seek for ad-hoc measures that improve the resistance of ECC against quantum computers. One such measure might be to add symmetric mechanisms to the Diffie-Hellman key agreement, which leads to the problem of distributing keys for these symmetric mechanisms.

An attacker could record communication data today in order to decrypt it in the future when Quantum Computers are available. Therefore it is more important to secure the key agreement part of authenticated key-agreement schemes: Signatures are checked „today“, and thus signature algorithms will remain secure for a longer period of time than the Diffie-Hellman key agreement scheme..

References.

[1] M. Lochter, J. Merkle, J.-M. Schmidt, T. Schütze, Requirements for Elliptic Curves for High-Assurance Applications (see http://www.nist.gov/itl/csd/ct/ecc-workshop.cfm)

[2] W. Schindler, A. Wiemers. Efficient Side Channel Attacks on Scalar Blinding on Elliptic Curves with Special Structure (http://www.nist.gov/itl/csd/ct/ecc-workshop.cfm)

[3] M. Lochter, A. Wiemers. Twist Insecurity. http://eprint.iacr.org/2015/577

[4] M. Lochter, Fault attacks on deterministic signatures, in preparation.

[5] M. Lochter, J. Merkle, J.-M. Schmidt, T. Schütze, Requirements for Standard Elliptic Curves http://eprint.iacr.org/2014/832

[6] J. Renes and C. Costello and L. Batina, Complete addition formulas for prime order elliptic curves, http://eprint.iacr.org/2015/1060

[7] TR 02102, see https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html

[8] https://www.nsa.gov/ia/programs/suiteb_cryptography/

[9] RFC 5639, Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, see https://tools.ietf.org/html/rfc5639

FIPS 186-4 Comment Reviewers,

This email contains the consolidated comments and responses to questions regarding the use of NIST recommended Elliptic Curves in FIPS 186-4, from Thales e-Security.

Thales e-Security is very supportive of the effort to expand NIST's support for alternative Elliptic Curves and for more transparent generation of curve parameters. We will be active participants in any/all NIST initiatives to provide alternatives to the definitions currently published in FIPS 186-4.

As part of this process, we believe that NIST also has to reconcile this update process against the August statement from the NSA regarding the transition process to quantum resistant algorithms (link: https://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml). Specifically, NIST should balance the expenditure of effort around improving the ECC definitions in FIPS 186-4 against accelerating efforts to define and specify appropriate quantum resistant algorithms for use in the US Federal Government space. While they can both go on in parallel, we are concerned about the potential for sending a mixed message, as well as managing the conflicts which arise from having to implement improvements in ECC algorithms, without a clear mandate that these are indeed secure alternatives for deployment.

Regardless, Thales e-Security remains committed to the process and is happy to collaborate and contribute to this endeavor. Below is the enumerated list of questions from the original request, with our responses inline, marked with [TeS], as an abbreviation for "Thales e-Security".

1. Digital Signature Schemes
a. Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?
[TeS] Generally, yes, but given the recent publication from the NSA (see above), it would be prudent to provide updates to both FIPS 186-4 and SP800-131A Revision 1 to reconcile the key size recommendations provided by the NSA. In particular, it would be helpful to define a depreciation schedule for the algorithms which were lower than recommended by the NSA (P384 is their recommended minimum), or alternatively provide justification for maintaining the status quo.
b. Are there other digital signature schemes that should be considered for inclusion in a future revision to FIPS 186? What are the advantages of these schemes over the existing schemes in FIPS 186?
[TeS] There are, of course, alternative schemes and we believe that any update to FIPS 186 for ECC must include an open discourse and process similar to that of SHA-3 and AES such that the international community can help select a 'best-of-breed' solution for which to standardize upon. Specifically, we would look to support curves and parameters that make pragmatic trade-offs between performance and ease of use, with particular attention to intangible qualities such as the ease of implementation and minimizing security risks during solution instantiation.

2. Security of Elliptic Curves
a. Do the NIST-recommended curves satisfy the security requirements of applications used by industry?
[TeS] On the whole, the NIST curves have not yet proven to be insecure, but there is certainly doubt being placed on the current set of curve parameters for the NIST curves. We expect that any activities around updating the ECC specification for FIPS 186 include more transparency for parameter and curve selection, regardless of which curve solution is approved.
b. Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?

[TeS] We are unaware of any demonstrable weaknesses in the NIST ECC curves, but we are of course concerned about the impact of quantum computing on the effectiveness of ECC in general, especially in light of the NSA statement referenced above. Any update to the NIST standards must provide the appropriate justifications in light of the aforementioned NSA statement.

3. Elliptic Curve Specifications and Criteria
a. Is there a need for new elliptic curves to be considered for standardization?
[TeS] Thales e-Security would be supportive of an open selection process, such as that used for SHA-3 and AES, to help determine appropriate curves for use in the next FIPS 186 standard.
b. If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?
[TeS] To help mitigate some of the issues identified with the existing set of curves, NIST should consider an open selection process which promotes transparency and maximizes international peer review. Of course, criteria such as security (which also includes designs which minimize the potential for insecure instantiation), performance, and freedom from IP claims, which would inhibit wide-spread adoption and deployment.
c. Do you anticipate a need to create, standardize or approve new elliptic curves on an ongoing basis?
[TeS] NIST should definitely define the new curve standards in a way which supports cryptographic agility, but we would caution against promoting a process which encourages frequent updates to these curves. We encourage NIST to take a pragmatic approach to ensure that any new curves are sufficiently scoped to encourage wide implementation and deployment. Any solution which promotes maximum choice is potentially orthogonal to those goals; too many choices is bad for wide adoption and interoperability.

4. Adoption
a. Which of the approved digital signature schemes and NIST-recommended curves have been used in practice?
[TeS] While the NIST Prime Curves (with equivalents in ANSI X9.62, and other standards) have dominated early deployments, there are numerous other implementations which are being promoted in different regions and markets. We see alternative curves getting significant traction, such as Brainpool and those used for blockchain technologies (e.g. secp256k1).
b. Which elliptic curves are accepted for use in international markets?
[TeS] There is no definitive list and the market is moving very quickly in multiple directions. See the response above for examples of the types of curves we are seeing in use around the globe.

5. Interoperability
a. If new curves were to be standardized, what would be the impact of changing existing implementations to allow for the new curves?
[TeS] This of course would have to be one of the primary considerations for defining a new set of curves, and that is the notion of cryptographic agility. Any new curves and/or parameters would introduce more choice so we should ensure that we pragmatically minimize the scope of these changes so as not to create confusion for implementers and architects.
b. What is the impact of having several standardized curves on interoperability?
[TeS] While more is not always better, we believe a new standard (or updated one) could support additional choices, as long as there is a clear and concise choice for the end users of this standard. The updates must make clear the differences between the choices and provide deterministic recommendations for implementation.
c. What are the advantages or disadvantages of allowing users or applications to generate their own elliptic curves, instead of using standardized curves?

[TeS] Certainly this capability provides the end user with some confidence that pre-calculated curves are not compromised by the organization proposing the curves, but this has to be contrasted with the potential security risk associated with curve generation algorithms producing insecure curves. If the standard supports this capability, care must be taken to choose a method which minimizes the ability for improper curve generation to occur. Preference should be given to solutions which allow the generation of per-system/user curves as long as they can be generated deterministically to ensure the security of those ephemeral curves.

6. Performance
a. Do the performance characteristics of existing implementations of the digital signatures schemes approved in FIPS 186-4 meet the requirements of applications used by industry?
[TeS] Our experience is that the current performance of ECC is comparable to other technologies, but performance will always be a consideration for wider adoption. The performance of any new standards should be included as a criteria for consideration. While we believe that "faster is better", the necessary tradeoffs must be made based upon multiple criteria.

7. Intellectual Property
a. What are the desired intellectual property requirements for any new curves or schemes that could potentially be included in the Standard?
[TeS] Thales e-Security is very supportive of the protection of intellectual property, but we must recognize that the deployment and adoption of a new cryptographic standard will be definitely hindered by algorithms (or implementations) encumbered by IP claims. History has shown that IP encumbered cryptography struggles to achieve wide or rapid adoption. Our preference is for new mechanisms, curves and schemes to be unencumbered by IP claims.
b. What impact has intellectual property concerns had on the adoption of elliptic curve cryptography?
[TeS] Following on from our statement above, we have observed the slow proliferation of ECC in general, and some of that can be directly attributable to IP/licensing concerns. One need only look at the unprecedented steps the US Government had to take to acquire a license (e.g. Suite B) for those wishing to implement products for government use. The world markets have no such ability to purchase blanket licenses, and therefore, many organizations are unable to support the costs and/or overhead associated with acquiring licenses. I can be seen that ECC has achieved success in areas where the IP was licensed in an open fashion (e.g. TLS protocols, S/MIME, etc.). Contrast that with the number of commercial CA ECC offerings there are, and it's easy to see how encumbered IP limits the proliferation of standardized cryptography. To reiterate our statement from above, our preference is for new mechanisms, curves and schemes to be unencumbered by IP claims.

If you have any follow-up questions about our responses, or require any additional feedback, please don't hesitate to contact me directly.

Robert Burns
CLASSIFICATION : Thales e-Security OPEN

Robert Burns
Chief Security Officer, Thales e-Security World Wide
Thales e-Security, 900 S. Pine Island Road, Suite 710, Plantation, Florida 33324
Tel: +1 954 888 6215
www.thales-esecurity.com

1a. The specified digital signature schemes satisfy the security requirements of applications used by industry.

1b. Schnorr-like signatures are worth considering. They are faster and easier to implement securely than ECDSA. They also require less code, which is important for embedded systems.

2a. The current NIST curves are generally believed to satisfy the security requirements of applications in the industry. However, they are difficult to implement securely, particularly in the presence of side channels.

2b. There are no attacks on the NIST curves that I know of, other than side channel attacks.

3a. Various industry groups have pushed for new standards for Edwards curves and for curves which support bilinear pairings.

3b. For curves which support pairings, the security (both against rho and against index calculus), cofactor and performance are important, as is the ease of implementation.

For curves which do not support pairings, again security, ease of implementation and performance are important. Security should include possibility of defenses against known side channel attacks. Ease of implementation should include general settings and not just ECDH and ECDSA (or other key exchange, encryption and signature schemes).

3c. No. However, a document which describes best practices for doing so might still be useful.

4a. For elliptic curves, mostly the prime-order curves; especially NIST-P256, and in lower-end products, NIST-P192. For signatures, RSA and ECDSA are popular, but DSA is quite rare in my experience.

4b. I don't know.

5a. This would be difficult for hardware vendors, but not especially difficult for software vendors.

5b. Having many curves hurts interoperability. If new curves are approved, there should only be a couple for each purpose.

5c. If a LOGJAM-like precomputation attack ever becomes feasible against elliptic curves, then allowing users to generate their own could be useful. However, best practices in EC generation are complicated and expensive to compute, and complicated to verify. Therefore, it seems unwise to broadly deploy user-generated curves. It may be suitable for niche applications, but these do not require standardization.

6a. The FIPS 186-4 curves are sufficiently performant for most use cases, but not enough that more performance would be a waste of time.
In particular the stronger curves can cause noticeable performance degradation on hardware which is less than PC or server grade.

7a. New curves will not be deployed unless they are free of IPR concerns. So it is important for any new curve or scheme to have no IPR restrictions (on the curve or prime shape itself), or at least to have worldwide royalty-free licensing on any IPR which is present.

7b. I'm sure I would have at least deployed an FHMQV and an ECQV variant by now if it weren't for IP concerns.

Mike Hamburg

CDC has no comments to provide on the Federal Information Processing Standard (FIPS) 186-4, Digital Signature Standard.

Thank you for the opportunity to review and comment.

Michael Harris, CISSP
Information Technology Specialist (Information Security)
Centers for Disease Control and Prevention
MWHarris@cdc.gov | 770-488-8052 office | 770-283-9589 cell

Regarding the request for comment on FIPS 186-4, subheading 2b:
security of elliptical curves, known attacks. This email is, more specifically, in regard to ECDSA.

I came across a paper you may find interesting, entitled "Using Bleichenbacher's Solution to the Hidden Number Problem to Attack Nonce Leaks in 384-bit ECDSA":
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.310.8920

That paper mentions a BKZ-based method for the range-reduction phase of the attack. Actually it also says "We could attack 4 and 5-bit leaks with BKZ, but not LLL." That leak is what allowed them to "extract the entire signing key using a 5-bit nonce leak from 4000 signatures".

I looked around a little and stumbled on another paper entitled "BKZ
2.0: Better Lattice Security Estimates":
https://eprint.iacr.org/2013/346.pdf

The conclusions/recommendations there are located at the section entitled "Revising Security Estimates". I figured I'd mention it in case it can offer anything useful in the way of improving one of the ECDSA algorithms.

-Brian Haslett

Comments in response to the request for comments at https://www.federalregister.gov/articles/2015/10/20/2015-26539/federal-information-processing-standard-fips-186-4-digital-signature-standard-request-for-comments#h-11 on FIPS 186-4 recommended elliptic curves.

The document includes generic industry views that are not one company specific.

1. Digital Signature Schemes

a. Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?

The signature schemes used by the commercial product industry require security strength that are well matched to the security strength of block ciphers, modes of operations and hashes used in security protocols, to avoid unnecessary implementation costs of algorithms that are stronger than the weakest link in the algorithm chain of security protocols.

Research into elliptic curves has produced criteria for evaluating curves, and NIST curves fail some of them. A comprehensive list of these criteria can be found on a Web site maintained by Lange and Bernstein at http://safecurves.cr.yp.to/. Curves meeting these security criteria, including key size, and based on strong hardware and software implementations are being actively deployed by industry.

b. Are there other digital signature schemes that should be considered for inclusion in a future revision to FIPS 186? What are the advantages of these schemes over the existing schemes in FIPS 186?

There are other digital signature schemes that could be considered, in order to improve the international acceptance of the curves and in response to the security concerns of industry and consumers.

The IETFs CFRG, proprietary deployments in millions of products and current ongoing standardization in established peripheral interface standards that will be deployed in billions of devices have adopted Ed25519 for signing and Curve25519 for Diffie-Hellman at the O(2^{128}) security strength, to match the strength of AES-128 and Shake-128. At higher security strengths, other curves meeting the safecurve criteria should be considered, such as curve 1174 (http://eprint.iacr.org/2013/325.pdf), E-382, M-383, M511, E521 (http://eprint.iacr.org/2013/647.pdf) and Ed488-Goldilocks.

Some relevant properties of these curves include:

- *Simpler implementation, with fewer or no special cases to consider, as described by the completeness criteria of safecurves.*
- *Easy constant time implementation for side channel mitigation, in particular the ability to use the Montgomery ladder, which is important both for efficient constant time software and hardware implementations.*
- *Rigid constants not open to abuse by the setter of the constants.*
- *Twist Security.*
- *Indistinguishability of curve points from random strings.*

2. Security of Elliptic Curves

a. Do the NIST-recommended curves satisfy the security requirements of applications used by industry?

Industry developed network security standards are converging on safe curve options like 25519 for DH and Signing.
DNSSEC has adopted Ed25519 for signatures.

The NIST recommended curves do not fully support some of industry requirements, including:

- *The ability to use the Montgomery ladder and easily achieve constant time implementations.*

- *An algorithmic footprint small enough to deploy in resource constrained scenarios.*

- *Twist security, P-224 in particular fails this.*
- *Indistinguishability from random E.G. P-224, P-256, P-384.*
- *Completeness of operations over the curve, E.G. all the Weierstrass curves, including P-224, P-256 and P0384.*

b. Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?

Greater level of transparency in disclosing the constants used in the NIST curves would alleviate perception of weaknesses and privacy concerns.
Other attacks described in literature include the use of points on the twist of P-224 and the use of points over which the curve operator is not valid.

Real and perceived weaknesses affect trust in NIST curves and lead to complex and fragile implementations.

3. Elliptic Curve Specifications and Criteria

a. Is there a need for new elliptic curves to be considered for standardization?

The efforts to migrate to Suite B have been unsuccessful, and removal of AES128 and P-256 from Suite B led to additional concerns.

An open standardization process is necessary to determine the successor to Suite B that has broad acceptance.

b. If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?

NIST should use criteria that meet security considerations and deployment considerations in resource constrained and high security contexts. Some examples are provided below.

- Security
    - The safecurves criteria

    - Widespread cryptanalysis of the algorithms over a reasonable time period.
- Resource constrained criteria
    - Simple implementation
    - Reasonable state size
- High Security
    - Equivalence to 128, 256 and 512 bit security
- Algorithmic compatibility
    - The ability to use modern standardized hashes including FIPS 202.

c. Do you anticipate a need to create, standardize or approve new elliptic curves on an ongoing basis?

As cryptographic research is developing, the need for ECC to provide a reasonable cost public key crypto option to replace RSA has become increasingly necessary. Better curves and criteria for judging curves have been identified. The size of RSA keys at 128 and 256 bit brute force strength has been a barrier to adoption of RSA in many consumer products.

Quantum computing brings additional research topics to the mix. If the cost of increasing the number of qbits in a quantum computer increases exponentially, then we can respond to the threat with just increasing key size. If the increase follows Moore's law, then ECC and RSA will be broken by quantum computing, and new methods will be required. Research into the topic is in its infancy and needs to be expanded.

4. Adoption

a. Which of the approved digital signature schemes and NIST-recommended curves have been used in practice?

TLS has used the NIST curves and signature schemes. X.509 uses them in many profiles. However, industry is reluctant to continue deploying these algorithms due to considerations described above.

b. Which elliptic curves are accepted for use in international markets?

Recent standardization activity which is ongoing in multiple standards bodies has resolved on Ed25519 and Curve25519 as an option that is acceptable to all parties. Many other curves are acceptable to some parties, but few are universally acceptable. The situation for higher key strength curves is not yet resolved.

Experience in ISO/IEC (JTC1's SC27) has indicated that government defined crypto approaches are not always acceptable to other governments and therefore to most National Bodies. Industry, academia, and government researchers need to work together on the definition of secure and acceptable curves that are also implementable in commercial products.

5. Interoperability

a. If new curves were to be standardized, what would be the impact of changing existing implementations to allow for the new curves?

Standardizing on curves that are simple to implement without errors and can fit directly in as replacements to existing curves at the same cryptographic strength will reduce the impact of changing existing implementations.

b. What is the impact of having several standardized curves on interoperability?

Any system which adopts multiple algorithms needs to adopt algorithm agility. Algorithm agility is a trend in commercial encryption standards today and an approach to replace algorithms that have been compromised or are no longer acceptable for other reasons. Algorithm agility is not without cost; it has led to many of the downgrade attacks that have been seen with TLS and other *cryptosystems. The smallest possible number of good curves at each key strength is the right approach to good solutions.*

c. What are the advantages or disadvantages of allowing users or applications to generate their own elliptic curves, instead of using standardized curves?

Randomized primes comes with increased implementation complexity and increased security from known attacks. The increased implementation complexity is a barrier for resource constrained deployments. Any live parameters of curves should also have a static option for efficient implementations.

Negotiated curve constants may improve security or may provide an attack vector for MITM situations in some situations.

6. Performance

a. Do the performance characteristics of existing implementations of the digital signatures schemes approved in FIPS 186-4 meet the requirements of applications used by industry?

The low performance of P-256 relative to Ed25519 leads to a direct reduction in the responsiveness of hardware authentication systems. This is one of the many reasons for Ed25519 and Curve25519 being adopted in place of P-256.

7. Intellectual Property

a. What are the desired intellectual property requirements for any new curves or schemes that could potentially be included in the Standard?

When it comes to encryption algorithms, industry has demonstrated a strong preference for having standards essential intellectual property available on reasonable, non-discriminatory and royalty-free terms.

b. What impact has intellectual property concerns had on the adoption of elliptic curve cryptography?

Intellectual property concerns kept ECC out of many standards. For example IEEE 802.16 and WiMax chose RSA2048 over ECC specifically because of the intellection property issues. This was repeated in many other situations.

For government developed encryption algorithm standards to be widely adopted, necessary intellectual property rights should be available on reasonable, non-discriminatory and royalty-free terms.

Hi.

I have read the FIPS 186-4 draft and have the following comments:

1) Generate (EC)DSA per-message random number deterministically instead of relying on a random bit generator.

Section 4.5 and Appendix B.2 describes how to generate the per-message random number used for DSA. Section 6.3 and Appendix B.5 describe this
for ECDSA. The recommended methods appear to me all rely on a Random
Bit Generator. There have been security incidents as a consequence of implementing this approach. It is described in an IETF RFC 6979 [1] how to generated the per-message k deterministically instead. I believe the RFC 6979 approach has no known significant disadvantages and has known significant advantages (resiliance against poor RBG), and should be adopted.

2) Drop ECC ruves P-192 and K-163 and consider dropping P-224 and K-233.

The security margin of these curves are fairly small (e.g., P-192 having a security level of 2^96), and in a document published today I believe it would be more conservative to drop these curves. The majority of implementation and deployment I see are using P-256 or higher already.
For P-224 and K-233 adding a warning of their lower security margin may be sufficient.

3) Adopt Curve25519 and Curve448.

These two curves have sufficient security level and are in the process of being adopted in the IETF for several Internet protocols, see [2].

4) Adopt EdDSA as a signature algorithm.

This signature algorithm offers some advantages over ECDSA. It is being adopted in the IETF for use in some protocols. See [3].

Thanks,
/Simon

[1] https://tools.ietf.org/html/rfc6979
[2] https://tools.ietf.org/html/draft-irtf-cfrg-curves
[3] https://tools.ietf.org/html/draft-irtf-cfrg-eddsa

Simon Josefsson

Dear NIST,

We think that the FIPS 186 standard should allow the use of extendable-output functions (XOFs), and in particular of SHAKE128 and SHAKE256, in the signature schemes.

First, the places where the standard uses a so-called mask generation function (MGF) should allow one to use a XOF instead.

For RSA, the standard supports the use of RSASSA-PSS as specified in PKCS #1. That construction makes use of an MGF, and an MGF is just another word for XOF. It allows long outputs and when being called with equal inputs but different requested output lengths, the leading part of the longer output is equal to the shorter output. PKCS #1 specifies a construction for building an MGF from a hash function, called MGF1. We believe it would be good engineering to use for MGF in RSASSA-PSS the FIPS 202's XOFs SHAKE128 and SHAKE256. In case SHAKE would be used in RSASSA-PSS for both compressing the message and as MGF, one may consider using domain separation between the two, e.g., by appending a short bit-string. But it is not clear that there is a need. In the current situation there is no domain separation between the hash function used in the MGF construction and that used for compressing the message.

Second, we think that the text discussing the choice of hash function in RSA, DSA and ECDSA should include XOFs here too. We refer to the text related to the choice of hash functions for DSA in Section 4.2 on pages
15 and 16, although there is similar text for the choice of hash functions for RSA and ECDSA.

<quote>
An approved hash function, as specified in FIPS 180, shall be used during the generation of digital signatures. The security strength associated with the DSA digital signature process is no greater than the minimum of the security strength of the (L, N) pair and the security strength of the hash function that is employed. Both the security strength of the hash function used and the security strength of the (L, N) pair shall meet or exceed the security strength required for the digital signature process. The security strength for each (L, N) pair and hash function is provided in SP 800-57.
[…]
It is recommended that the security strength of the (L, N) pair and the security strength of the hash function used for the generation of digital signatures be the same unless an agreement has been made between participating entities to use a stronger hash function. When the length of the output of the hash function is greater than N (i.e., the bit length of q), then the leftmost N bits of the hash function output block shall be used in any calculation using the hash function output during the generation or verification of a digital signature. A hash function that provides a lower security strength than the (L, N) pair ordinarily should not be used, since this would reduce the security strength of the digital signature process to a level no greater than that provided by the hash function.
</quote>

This second paragraph distinguishes the digest length from the security strength of the hash function, as is the case for FIPS 202's XOFs
SHAKE128 and SHAKE256. It would be appropriate to allow the use of the SHAKE functions in this context. Their security strength (128 bits for
SHAKE128 and 256 bits for SHAKE256) should be chosen equal to or larger than that of the (L, N) pair and as output one should simply request |q| bits, with |q| the bit length of q. Note that there is also no

need for coding the output length in the input of the SHAKE function as the output length is completely determined by the length of q.

For ECDSA and RSA the same arguments apply.

Kind regards,
The Keccak and Keyak team
Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche and Ronny Van Keer

General comments on the pub:

- - Revisions should be rare because the impact will be so broad
- - Any recommended "implement by" dates need to be long term

Andrea

Andrea A. Kunz
AFLCMC/HNCYP (MITRE)
AF PKI SPO
DSN 945-9168
Comm (210) 925-9168
SIPR - andrea.a.kunz.ctr@mail.smil.mil

To whom it may concern,

FIPS-186-4's ECC signatures makes it difficult to implement robust and secure implementations of digital signatures and ECDH with the same curves due to many misfeatures. These misfeatures have led to bugs in many products which I have exploited over the years, in some cases leading to revelation of private keys.

The first misfeature is the use of randomness when signing. Randomness is inherently necessary when generating keys. But even slight deviations from properly generated random numbers can lead to catastrophic breaks in security. The Debian bug demonstrated that random number generators can be silently broken for years without noticing.

The second misfeature is the requirement to implement inversion modulo the group order to sign. This operation is painfully slow. Even worse, it is difficult to side-channel protect. When one is asked to protect a single addition and multiplication, solutions with a very high overhead are possible. But for an algorithm like Euclidean division such overhead is prohibitive. Simply blinding ignores the leakages from the multiplications before and after the division.

The third misfeature is the absence of an efficient and complete addition law on the curves used. This produces painful special cases in blinded multiplications, and in verification formulas. A complete addition law is much simpler to use in algorithms and protect against side-channels. Of course one can write a complete addition routine, but it involves many special cases, and is often wrong in the rarer cases.

The fourth misfeature is the badly shaped primes. Primes of the shape 2^s-c support a wide variety of architectures, with constant-time routines. NIST provided routines are slower due to the need to completely reduce, and inherently variable time. Of course, with much work, one can produce constant-time routines for the Solinas primes, but this is quite a pain.

All of these misfeatures contribute to cryptography that is slow and untrustworthy. While good implementations are possible, they aren't easy, and frequently the implementations that are used aren't. Better solutions exist and are already seeing acceptance on the Internet, such as Ed25519. A complete and secure implementation of Ed25519 can be done in the allowable space of an IOCCC entry, a feat I don't believe can be done with FIPS 186-4.

I would propose that NIST add Ed25519 and the upcoming Ed448 to FIPS 186-4. Both fix all of the above problems, and have been standardized through the IRTF. I'm not aware of any serious concerns with either of these proposals or with the associated proposals for Diffie-Hellman key exchange.

Sincerely,
Watson Ladd

1. Digital Signature Schemes

a. Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?

Yes. Key sizes satisfy current security requirements. If NIST intends to issue new elliptic curves my opinion is that standardized key sizes from 256 bits and up should be kept (for example, 256-bit and 384-bit).

2. Security of Elliptic Curves

b. Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?

No. To the best of my knowledge, there are no attacks on current NIST-recommended curves. However, protecting the current NIST curves against implementation attacks is hard.

3. Elliptic Curve Specifications and Criteria

a. Is there a need for new elliptic curves to be considered for standardization?

Yes. In my opinion, there is a need for new ECC standardization considering the progress of ECC in the past 15 years. In special, NIST should consider state-of-the-art elliptic curves (and their corresponding implementations) that take side-channel security and performance into account in their design. For example, the Edwards FourQ curve [1] proposed by C. Costello and P. Longa combines most efficient prime and elliptic curve arithmetic to achieve speed-record execution time while offering strong security and great flexibility to implementers. FourQ can be implemented using its Edwards form for maximal performance or be transformed to the Montgomery form for maximal compactness using the Montgomery ladder.

b. If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?

1. Elliptic curves should target currently standardized key sizes (for example, 256-bit and 384-bit).
2. Elliptic curves should have twisted Edwards form supporting the fastest and complete addition formulas [2][3].
3. Elliptic curves should support simple and secure implementations, e.g., be defined over a Mersenne prime field when available (e.g., FourQ for the 128-bit security level).

Dr. Zhe Liu
University of Waterloo

[1] C. Costello and P. Longa, “FourQ: four-dimensional decompositions on a Q-curve over the Mersenne prime”, in Progress in Cryptology - ASIACRYPT, volume 9452 of LNCS, pp. 214-238, 2015. http://eprint.iacr.org/2015/565.pdf

[2] D. J. Bernstein, P. Birkner, M. Joye, T. Lange and C. Peters, “Twisted Edwards curves”, in Progress in Cryptology - AFRICACRYPT 2008, volume 5023 of LNCS, pp. 389-405, 2008.

[3] H. Hisil, K. K. Wong, G. Carter and E. Dawson, “Twisted Edwards curves revisited”, in Progress in Cryptology - ASIACRYPT 2008, volume 5350 of LNCS, pp. 326-343, 2008.

When this is phased out, can I have the leftover Ellipses. I like to Spirograph.

Kris Madsen

As digital signature algorithms with less then 112-bit security are disallowed according to 800-131A, it makes sense to remove them from 186-4. Anyone wanting to support them for legacy verification can find them in 186-3.

Examples of suggested changes:

- p15 Remove DSA (1024, 160),

- p16 Update the example with the years 2009 and 2010.

- p16 Remove the restriction that non-CAs should not use (3072, 256), I think everybody should use that.

- p22 Remove RSA 1024

- p23 Remove the restriction that non-CAs should not use nlen=3072, I think everybody should use that.

- p27 Remove the 160-223 range

- p53 Remove the nlen=1024 options

- Appendix D: Remove P-192, K-163, and B-163,

Best Regards,
John Mattsson

JOHN MATTSSON
MSc Engineering Physics, MSc Business Administration and Economics
Ericsson IETF Security Coordinator
Senior Researcher, Security

3 December March 2015

National Institute of Standards and Technology
Information Technology Laboratory
FIPS186-comments@nist.gov

Subject: Comments on Federal Information Processing Standard (FIPS) 186-4, Digital Signature Standard

Microsoft Corporation appreciates the opportunity to submit comments on FIPS 186-4, Digital Signature Standard, and in particular the elliptic curves specified in Appendix D of that document. Our detailed answers to the questions posed by NIST are below, followed by additional comments on FIPS 186-4 for NIST to consider. Overall, and as we stated during our presentations and comments at the NIST Workshop on Elliptic Curve Cryptography Standards held earlier this year, we have no reason to believe there are any technical problems with the NIST curves specified in Appendix D. Nevertheless, the fact that those curves were not generated in a rigid and transparent manner, combined with recent disclosures, has cast doubt on the integrity of those curves especially with our international customers. Thus, as part of NIST's commitment to transparency, openness, technical merit, and integrity we believe that NIST does need to specify additional curves through an open, rigid and transparent process that all stakeholders will be able to trust. Due to the lack of trust in the current NIST curves we are seeing an increase in specification of nation-specific curves by national standards bodies, which harms international interoperability. Thus it would be ideal if as part of any new curve selection process NIST undertakes, NIST could reach agreement with one or more non-US national standards organizations on a jointly-acceptable selection and generation process.

Here are Microsoft's specific responses to the questions posed by NIST:

1. Digital Signature Schemes

a. Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?

Answer: Yes. By far the vast majority of digital signatures Microsoft generates and verifies still use RSA, as ECDSA has seen slow adoption to date. Both RSA and ECDSA signature schemes satisfy the security requirements of our applications. Specifically concerning ECDSA and the curves specified in Appendix D, only the curves P-256, P-384 and P-521 satisfy our current security requirements. The smaller prime curves P-192 and P-224 we believe are too small to provide adequate security in any environment and we do not allow their use in Microsoft products and services. The binary and Koblitz curves specified in Appendix D are not widely used. Should NIST decide to specify additional curves for FIPS 186-4, we would recommend 256 bits as the lower bound for the field size of any new prime curves specified.

b. Are there other digital signature schemes that should be considered for inclusion in a future revision to FIPS 186? What are the advantages of these schemes over the existing schemes in FIPS 186?

Answer: Yes. We believe NIST should consider specifying either a modified version of ECDSA or a Schnorr signature scheme that (among other things) replaces the pseudorandom generation of the value k in ECDSA by a deterministic value (e.g., computing k as the hash of the static secret and the message to be signed). See ECDSA*, ECDSA+ and ECSchnorr* in Section 7 of [KM15a], or the discussion in Section 2 of [BDLSY12], or [Por13]. Schnorr-type signature schemes have security and performance benefits by eliminating the need for (a) new randomness for each signature, and (b) computing an inversion modulo a random-looking prime.

2. Security of Elliptic Curves

a. Do the NIST-recommended curves satisfy the security requirements of applications used by industry?

Answer: Yes, subject to our corporate standards for minimum acceptable key lengths and our interest in only the specified prime curves. To be precise, the P-256, P-384 and P-521 curves meet our security requirements. The smaller P-192 and P-224 curves do not meet our minimum bar for security in today's computing environment, and as we discuss in our answer to 2b below we believe the non-prime curves in FIPS 186-4 could be removed from the standard.

b. Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?

Answer: Recent theoretical advances in attacking elliptic curves defined over binary extension fields fuel suspicion that the NIST-recommended curves over binary fields may be weaker than elliptic curves defined over prime fields [FPP+12]. If new elliptic curves are proposed, they should be defined over prime fields.

Implementations of the NIST-recommended curves are more difficult to protect against side-channel attacks in comparison with other elliptic curve alternatives. Traditional addition formulas on Weierstrass curves have failure cases, for which two solutions have been proposed [BCL +15] [RCB15], however these solutions for the classic Weierstrass form are complex or expensive.

3. Elliptic Curve Specifications and Criteria

a. Is there a need for new elliptic curves to be considered for standardization?

Answer: Yes. Following the Snowden disclosures, and in particular the details of Project BULLRUN and the allegations that the Dual_EC_DRBG RNG was designed with a known trapdoor, there are suspicions that the NIST prime curves could have been deliberately weakened during their generation. Although this is just a perception, and we reiterate that there is no technical evidence that any manipulation of the NIST prime curves occurred, the disclosures to date about Dual_EC_DRBG are enough to cause a breach of trust for our business customers, particularly those outside the United States (but concerns have also been raised by some US customers). We believe NIST needs to specify new elliptic curves in order to restore trust in ECDSA and other ECC algorithms (e.g. ECDH). The new curves specified by NIST need to be generated in such a way that they can be trusted by industry (both US and non-US businesses), and by governments worldwide. The lack of trust in current NIST ECC standards is further driving proposals for national cryptographic standards in many countries and jurisdictions, and the cost to industry to support many different national ECC standards could become enormous and disruptive to international commerce.

b. If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?

Answer: NIST's evaluation must address the problem of transparency and provenance that has cast concerns over the curves currently specified in Appendix D. NIST's evaluation should also incorporate additional security considerations that have become known since 1999. We believe that a rigid, transparent generation procedure for new curves is a baseline requirement to restore trust within the industry for both US and global customers. It is important that the curves have broad international support, and in particular, that they be supported by important economic partners. As NIST heard from many stakeholders at the Workshop on Elliptic Curve Cryptography Standards, restoring trust in NIST-specified curves is the most important criterion for any new curves. Where possible, the generation procedure should also include the latest recommendations to achieve secure and efficient implementations, but restoring trust for all of our customers in all locales is the most important evaluation criterion.

c. Do you anticipate a need to create, standardize or approve new elliptic curves on an ongoing basis?

Answer: No, we do not anticipate such a need, especially given the potential need to migrate to quantum-resistant cryptographic algorithms within the next decade.

4. Adoption

a. Which of the approved digital signature schemes and NIST-recommended curves have been used in practice?

Answer: ECDSA is not currently as popular for digital signatures as RSA is. The NIST curves P-256, P-384 and P-521 are the most used elliptic curves (in decreasing order) in practice. See [BHHMNW13].

b. Which elliptic curves are accepted for use in international markets?

Answer: NIST P-256 remains as the most widely deployed curve today, followed by NIST P-384. Nevertheless, governments in China, Russia, Germany, and elsewhere are increasingly generating and endorsing their own sets of elliptic curves for various use cases, in part due to widespread distrust of the NIST prime curves. Increasingly we are being required to implement these additional sets of elliptic curves to meet international market requirements.

5. Interoperability

a. If new curves were to be standardized, what would be the impact of changing existing implementations to allow for the new curves?

Answer: The impact will be highly dependent on both the number and the nature of the new curves. NIST could completely address the provenance concerns by simply publishing updated b constants for the NIST prime curves that are generated in a transparent way (effectively defining "version 2" of P-256, P-384 and P-521). This would mean keeping the same prime fields, same short Weierstrass models, and essentially the same implementations, which would ultimately ease the transition from the old to the new. If new fields and new styles of curves are proposed, then the field arithmetic and addition formulae are likely to be different which will increase the overall effort required to support them. Additionally, the amount of work needed to implement new curves will also vary depending on the implementation environment (e.g. large software system, constrained software environment, hardware implementation, etc.).

b. What is the impact of having several standardized curves on interoperability?

Answer: To increase the chances of widespread deployment and interoperability, NIST should minimize the total number of curves in the next version of FIPS 186. Even though FIPS 186-4 Appendix D specifies 15 curves, only three (P-256, P-384 and P-521) have any significant deployment and interoperability is in practice only available with those three curves. If NIST decides to specify additional curves as part of Appendix D, then we recommend that NIST formally deprecate all of the existing curves specified in Appendix D except for P-256, P-384 and P-521.

c. What are the advantages or disadvantages of allowing users or applications to generate their own elliptic curves, instead of using standardized curves?

Answer: We believe there are significant disadvantages to allowing users to generate their own curves: more curves introduce obvious interoperability issues and letting users choose curves ``on-the-fly'' or in an *ad hoc* way opens up a whole new space of attacks: checking even the basic security requirements (e.g., proving primality and checking the embedding degree) of a dynamically-generated curve on-the-fly is not practical. Thus, should NIST choose to standardize additional curves, there is significant incentive for NIST to do so in such a way that users cannot see any advantage to generating their own curves.

6. Performance

a. Do the performance characteristics of existing implementations of the digital signatures schemes approved in FIPS 186-4 meet the requirements of applications used by industry?

Answer: Yes, although there is room for improvement. As was clearly communicated by the industry participants at the NIST Workshop on Elliptic Curve Cryptography Standards held earlier this year, performance is neither a deployment blocker for the existing NIST prime curves nor a desired primary design goal for any to-be-specified curves. Research over the last 15 years has shown that it is possible to make significant performance improvements over the NIST prime curves at comparable security levels, such as the Microsoft Research "FourQ" curve presented at the Workshop which is today the fastest curve at the 128-bit security level by a wide margin. But simply choosing faster curves does not help NIST solve the trust issues that customers have with the existing NIST curves, and a selection process primarily driven by performance will not address the concerns highlighted in the VCAT report. We recommend that if NIST is going to choose additional curves for standardization that NIST focus primarily on security criteria, rigidity and transparency to establish the trustworthiness of the new curves, and then move on to lesser concerns such as performance.

7. Intellectual Property

a. What are the desired intellectual property requirements for any new curves or schemes that could potentially be included in the Standard?

Answer: No essential intellectual property should be required for secure and efficient implementations of any new curves standardized by NIST.

b. What impact has intellectual property concerns had on the adoption of elliptic curve cryptography?

Answer: Past and present intellectual property concerns have had a severe negative impact on the broad adoption of elliptic curve cryptography to date. One need only look at the amount of time and effort government and standards-setting organizations have spent investigating intellectual property issues to see the retarding effect of IP concerns. For example, version 1.1 of the W3C XML Signature Syntax and Processing Recommendation (the first version to add support for ECDSA and the NIST P-256, P-384 and P-521 curves) was delayed for approximately two years by spurious intellectual property claims against the NIST prime curves. Other standards activities involving elliptic curve cryptography have had similar problems.

Additional comments for consideration

In addition to the specific questions posed by NIST, Microsoft offers the following additional comments on FIPS 186-4 and the potential goals of a selection process for new elliptic curves. As we have stated above, should NIST decide to specify new elliptic curves for use with FIPS 186-4, the overarching goal of the selection process must be to restore stakeholder confidence and the best way to do that is to define an open, rigid and transparent process for curve selection using a comprehensive set of stakeholder-approved security criteria. Microsoft Research's published work in the area of rigid curve selection processes [BCL+15] is particularly informative in this regard.

Within the academic community that studies elliptic curves there is a small (but very vocal) subset that believes that the only way to choose elliptic curves is by performance measurements, and that standardization efforts should be driven solely by current performance metrics of candidate elliptic curves as measured on current hardware. We disagree with this view for many reasons, including the fluidity of relevant target platforms, the variety of hardware and software execution environments, the lack of overall rigidity provided by this approach, and the fact that any standard that chooses to follow this approach is immediately outdated and deprecated when a faster curve is discovered. This is the situation now faced by a recent protocol standardization effort in the ECC space that chose to follow the "choose only by performance" path and is now stuck specifying a slow curve at the 128-bit security level, Curve25519 [Ber06], instead of the fastest curve now available, FourQ [CL15]. We have included more technical information on FourQ in Appendix A of these comments.

Thank you again for the opportunity to submit these comments. We would be happy to discuss these answers or any other aspect of FIPS 186-4 and its Appendix D elliptic curves with you.

Appendix A: The FourQ Elliptic Curve

Elliptic curves are celebrated in cryptography because, unlike traditional RSA and finite field primitives, they are as stubborn to an attacker as a black-box group, i.e., the best known attacks against well-chosen elliptic curves will also work against any other group. Put another way, as far as cryptographic groups go, elliptic curves are as secure as is theoretically possible.

With the goal of achieving top security and top performance, the recent work in [CL15] put forward an answer to the following question: *what is the best performance we can achieve while still presenting attackers with a concrete cryptographic group that is as secure as a black-box group?*

In the VCAT review document, Edward Felten recommended that "NIST should generate a new set of elliptic curves for use with ECDSA in FIPS 186… and [that these curves] should incorporate the latest knowledge". The work in [CL15] incorporates many of the milestone achievements in the field of elliptic curve cryptography, particularly those made since the NIST curves were standardized in 1999. It combines the [GLV01], [GLS09] and [Smi13] improvements to give factor-4 exponent decompositions, which is currently believed to be the highest possible decomposition over large characteristic fields (without sacrificing the black-box security of the underlying group). It uses the fastest known elliptic curve addition formulas [HWCD08] and the fastest large characteristic finite field at the 128-bit security level. Subsequently, as is shown in [CL15, Table 5], variable-base scalar multiplications on this curve are up to 5.4x faster than those on the NIST Curve P-256 and between 2x and 3x faster than any of the proposals for NIST alternatives at the 128-bit security level, e.g., Curve25519 [Ber06]. Moreover, [CL15] (and the associated source code) shows that these speeds can be replicated by implementers in an exception-free, constant-time fashion.

As is discussed in [CL15], even when the endomorphisms on this curve are not exploited, and a plain scalar multiplication is used, this curve still significantly outperforms any other proposed alternatives because of the fast underlying finite field in combination with the fast, complete addition formulas. For example, with endomorphisms disabled, [CL15, Table 5] reports that FourQ is up to 2.9x faster than NIST Curve P-256 and up to 1.5x faster than Curve25519.

We believe it would take a significant and surprising discovery in the realm of ECC for any alternative primitive to be able to outperform this curve at high levels of security (i.e., at or beyond the 128-bit security level).

As is mentioned in the full version of [CL15], there are two main differences between this curve and the prime curves found in various standards, including FIPS 186. Firstly, this curve is defined over a quadratic extension field: although larger extensions of prime fields have been shown to give rise to certain attacks, elliptic curves over quadratic extension fields are as secure as those over prime fields. On the other hand, the quadratic extension field gives this curve two big performance advantages: the underlying Mersenne prime characteristic and the non-trivial action of Frobenius that is used to produce an additional endomorphism [GLS09, Smi13]. Secondly, this curve has a small discriminant and is therefore *special* (another example like this is *secp256k1*, the Bitcoin curve): although conventional wisdom thought it best to avoid special curves, there is today no reason to believe that these curves are less secure than random curves. In fact, Koblitz, Koblitz and Menezes [KKM11] argue that such special curves could end up being more secure than large discriminant curves. If an attack like the hypothetical ones in [KKM11, Section 11.3] was to be realized in the future, then there is a chance that random curves with large discriminant (like NIST Curve P-256 and Curve25519) are more vulnerable than special curves with a small one.

Finally, [CL15, Remark 2] explains that if one accepts the “choose only by performance” approach to elliptic curve selection, then the result for the 128-bit security level is FourQ. By requiring the fastest field at the 128-bit security level, the fastest known addition formulas, and the availability of a 4-dimensional decomposition, the only known curve meeting all these requirements is FourQ. For those who believe that rigidity and transparency come solely from performance, FourQ is the most rigid curve available at the 128-bit security level.

References

[BCL+15] Bos, Costello, Longa and Naehrig, “Selecting Elliptic Curves for Cryptography: An Efficiency and Security Analysis”, J. Cryptographic Engineering, 2015. Available at: http://eprint.iacr.org/2014/130

[BDLSY12] Bernstein, Duif, Lange, Schwabe and Yang. High-speed high-security signatures. Journal of Cryptographic Engineering. 2, pp 77-89, 2012.

[Ber06] Bernstein. Curve25519: New Diffie-Hellman speed records. PKC, pp 207-228, 2006.

[BHHMNW13] Bos, Halderman, Heninger, Moore, Naehrig, Wustrow. Elliptic Curve Cryptography in Practice. Financial Cryptography Workshop, LNCS, v. 8437, pp 157-175, 2013.

[BL07] Bernstein and Lange. Faster addition and doubling on elliptic curves. In ASIACRYPT 2007, vol. 4833 of LNCS, pp. 29-50, Springer, 2007.

[CL15] Costello and Longa. FourQ: Four-dimensional decompositions on a Q-curve over the Mersenne prime. ASIACRYPT, 2015. Library: http://research.microsoft.com/en-us/projects/fourqlib/. Full version: http://eprint.iacr.org/2015/565.pdf

[Edw07] Edwards, “A normal form for elliptic curves”, Bulletin of the American Mathematical Society, 44:393–422, 2007.

[FPP+12] Faugère, Perret, Petit, and Renault, “Improving the complexity of index calculus algorithms in elliptic curves over binary fields”, in EUROCRYPT 2012, vol. 7237 of LNCS, pp. 27–44, Springer, 2012.

[GLV01] Gallant, Lambert and Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. CRYPTO, pp 190-200, 2001.

[GLS09] Galbraith, Lin and Scott. Endomorphisms for faster elliptic curve cryptography on a large class of curves. EUROCRYPT, pp 518-535, 2009.

[HWCD08] Hisil, Wong, Carter and Dawson. Twisted Edwards curves revisited. ASIACRYPT, pp 326-343, 2008.

[KKM11] Koblitz, Koblitz and Menezes. Elliptic curve cryptography: The serpentine course of a paradigm shift. Journal of Number Theory, 131.5, pp 781-814, 2011.

[KM15a] Koblitz and Menezes. The random oracle model: a twenty-year retrospective. Designs, Codes and Cryptography, 77(2-3), 587-610, 2015.

[KM15b] Koblitz and Menezes. A riddle wrapped inside an enigma. Preprint: http://eprint.iacr.org/2015/1018.pdf

[Por13] Pornin. Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA). RFC6979, 2013.

[RCB15] Renes, Costello, Batina. Complete addition formulas for prime order elliptic curves. Preprint: http://eprint.iacr.org/2015/1060.pdf.

[Smi13] Smith. Fast families of elliptic curves from Q-curves. ASIACRYPT, pp 61-78, 2013.

FIPS 186 Review Team,
Thanks for the opportunity to review your Digital Signature Standard. The Forest Service has reviewed your draft and concur on the proposed standard. We do not have any changes.

Douglas Nash
Chief Information Officer
Forest Service
CIO Staff, Business Operations
p: 703-605-4600
c: 703-405-8256
dnash@fs.fed.us201 14th St, SW
Washington, DC 20024
www.fs.fed.us
Caring for the land and serving people

Please refer to FIPS 186-4

I recommend that the material contained in appendix F be fully incorporated in a rewrite of appendix C.

For example, when generating a random prime of size 1024 bits or more, it is sufficient to do one Miller-Rabin test. It is not hard to bound the probability of error by (1/2)^150 (I'm eyeballing it here, it is likely much smaller).

In contrast, appendix C.1 prescribes 40 Miller-Rabin tests for a probability of error of (1/2)^80. That is incorrect, leads to wasted cycles, introduces complexity (e.g. use of Lucas test to get away with fewer Miller-Rabin tests).

Additionally:

- appendix C has recommendations for generation of "provable primes". I recommend that material be deleted. As far as I can tell, it responds to an outdated view that "provable" is safer than "probabilistically true" no matter what the probability is.

- appendix C has recommendations for generation of "strong primes" (those containing no multiplicative subgroup of small order). These recommendations add complexity (as we have seen in our own implementations), decrease the entropy of the primes, and adds no significant hardness to the problem of factoring RSA keys. This is because the elliptic curve factoring algorithm can randomize the problem in such a way that strong primes are no stronger than random primes. I recommend discussion of "strong primes" be demoted to an informative appendix.

Rene Peralta

Comments on FIPS 186-4 (December 20, 2015)

Allen Roginsky

1. (General) The problem with many algorithms in this standard is that they are written simply as sets of instructions. They text should, in addition, tell the reader what the algorithms do. Take, for example, the provable prime construction algorithm in Appendix C.10. Why does (p-1) need to have a prime divisor p_0, in addition to p_1? Why does p_0-1 need to have a large prime divisor of its own? What is the meaning of each step in the algorithms? The reader will have to reverse engineer the logic of all of the algorithms to understand what they are attempting to accomplish and how. It certainly helps if the reader has read some papers about the Shawe-Taylor algorithm but the standard itself should be self-contained and clear to the reader.

2. Do not stop looking for a prime in the algorithm in Appendix B.3.3 after the first 5*nlen/2 candidates turn out to be composite. This is not an extraordinary condition and there is no reason (yet) to change the seed. The failure to find a prime among the first 5*nlen/2 candidates should naturally occur with the probability of approximately 1 in 2 million. Here is a statement from my early write-up.

The density of primes of size x is 1/ln(x). That is, on the average, a prime occurs among the consecutive integers with the "probability" of 1/ln(x). If you limit the candidates to those that are odd, the probability is 2/ln(x). A composite then occurs with the probability of (1 – 2/ln(x)). In our case, $x \approx 2^k$, so the composite candidate probability is $\left(1-\frac{2}{k\ln 2}\right)$. The probability of selecting 5*k composites in a row is $\left(1-\frac{2}{k\ln 2}\right)^{5k}$ = $\left(1-\frac{2}{k\ln 2}\right)^{\frac{k\ln(2)}{2}*\frac{10}{\ln(2)}}$. Since $\left(1-\frac{1}{n}\right)^{n} \approx e^{-1}$, the last expression is close to $e^{-\frac{10}{\ln(2)}}$, or about 1 in 2 million, just as we had claimed.

I suggest changing 5*nlen/2 to 20*nlen/2.

3. Many vendors would like to add an (optional) condition on the primes requiring that p = a(mod 4) and q = b (mod 4), where a and b are either 1 or 3. This, they claim, will make the algorithm stronger, and choosing a = b = 3, in particular, would make it less susceptible to the timing and power analysis attacks. I believe we should accommodate this requirement. Moreover, we may allow p and q to be chosen such as p = a(mod d) and q = b (mod d), where d is some positive integer less than 100. This will include the case of d=8 from X9.31. As it happens now, the vendors still achieve their goal by dropping the generated p and q if they do not satisfy the above conditions (mod d). Why make them waste so much effort and not have this built into the algorithm?

4. Another optimization. The way the prime generation algorithm is written, when generating q the standard prescribes checking that p and q are not close before checking that q > $\sqrt{2}2^{k-1}$ (1). This is a waste of computing cycles. The $|\ p-q\ |\le 2^{(nlen/2)-100}$ condition can occur with only negligible probability, yet condition (1) may fail (legitimately, due to the method of generating q) with probability $\sqrt{2}-1\approx 0.414$. So why *always* check for the proximity of p and q (the condition that should never take place) when in almost half of these cases this q would not even be a candidate. Let (5.4) and (5.5) in B.3.3 switch places.

5. The Chinese remainder theorem (CRT) representation of a private RSA key. It is stated in Section 5.1 that the CRT representation of the (n, d) key is allowed. But what is this representation? The standard does not say. The standard does show how to use the CRT to construct the primes p and q using the auxiliary parameters e, p_1, p_2, q_1 and q_2, but does not say anything about the representation. I could not find the answer in X9.31 either. If by the CRT representation we man the (p, q) pair, then the question becomes: does the module still need to compute the value of d' = LCM(p-1, q-1) and make sure that d' is not too small. If it does compute d', then what is the benefit of this alternative representation? If the module does not compute it, then there is a (highly improbable) danger of the Don Coppersmith attack. I suggest that the computation of d' is not needed since probability of d' being less than $2^{nlen/4}$ is infinitesimally small for the key sizes that we are considering. However, we all have to think about it and make a decision.

6. Allow *nlen* greater than 3072 for the RSA signature algorithm.

1. Digital Signature Schemes

a. Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?

On their face, yes. There is always the concern that there are unpublished attacks by state actors that make some schemes and key sizes vulnerable.

b. Are there other digital signature schemes that should be considered for inclusion in a future revision to FIPS 186? What are the advantages of these schemes over the existing schemes in FIPS 186?

---

2. Security of Elliptic Curves

a. Do the NIST-recommended curves satisfy the security requirements of applications used by industry?

On their face, yes. There is always the concern that smaller key size curves, which are less costly computationally to implement, and less costly (bytes sent in certificates) to use, are vulnerable to attack.

b. Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?

---

3. Elliptic Curve Specifications and Criteria

a. Is there a need for new elliptic curves to be considered for standardization?

Yes - as a risk mitigation scheme, having more curves to choose from increases the resources required to compromise a system. Also, if a product uses a n-bit curve and it is compromised, having other similarly derived n-bit curves allows switching without incurring additional computational cost.

b. If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?

The process should be open - there is great distrust in NSA-contributed content to NIST standards, so curves submitted by academia and industry should be preferred. While computational efficiency is an important criteria, it should not be the only one. Curves chosen in the name of computational efficiency that offer less security can be a bad tradeoff - computational efficiency becomes less critical as system capability increases and ECC computation becomes more common in hardware.

c. Do you anticipate a need to create, standardize or approve new elliptic curves on an ongoing basis?

Yes. As long as the general algorithm remains the same, having additional curves of the same strength can make a product more robust

4. Adoption

a. Which of the approved digital signature schemes and NIST-recommended curves have been used in practice?

P-256 with ECDSA

b. Which elliptic curves are accepted for use in international markets?

---

5. Interoperability

a. If new curves were to be standardized, what would be the impact of changing existing implementations to allow for the new curves?

Probably little - TLS allows for security suite negotiation, so systems can fall back to older curves where new ones are not supported.

b. What is the impact of having several standardized curves on interoperability?

Probably little as in a)

c. What are the advantages or disadvantages of allowing users or applications to generate their own elliptic curves, instead of using standardized curves?

Advantages - more curves means more work for an attacker. Disadvantages - without good vetting of the security of user generated curves, there is risk for weak security. There is no mechanism in TLS for ad-hoc curves.

6. Performance

a. Do the performance characteristics of existing implementations of the digital signatures schemes approved in FIPS 186-4 meet the requirements of applications used by industry?

Mostly - the goal is always the most security at the cheapest cost, where cost is measured in code or hardware complexity, energy spent in computation, and energy spent in transmitting security information. If other curves offered similar security at lower cost, or better security at comparable cost, they would be used.

7. Intellectual Property

a. What are the desired intellectual property requirements for any new curves or schemes that could potentially be included in the Standard?

Schemes that are both free and with liberal (MIT style) licenses are preferred.

b. What impact has intellectual property concerns had on the adoption of elliptic curve cryptography?

--

Jonathan Simon
Linear Technology, Dust Networks product group
32990 Alvarado-Niles Road, Suite 910
Union City, CA 94587
(510) 400-2936
(510) 489-3799 FAX
jsimon@linear.com

Dear NIST,

With reference to the request for comments about current FIPS 186-4 standard [1]:

Q 1.a: Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?
A: The ECDSA Per-Message Secret Number "k" (as mentioned in section B.5) is required to be randomly generated.
There's a history of issues about this requirement and new solutions have been proposed for the per-message secret number to be generated deterministically. These have the advantage that they don't rely on a secure random generator.
The deterministic procedures are usually based on a computation of an hash function (or a MAC). "k" is derived from an additional secret key and the message to be signed. If the computation is fully deterministic, it may introduce a new side channel leakage.
We would advise to have a construction that allows the computation of the "k" in a deterministic way with the option of involving a random seed in it. Such random seed could be useful in the context of side channel protection.
We've referenced ECDSA, but the same arguments apply to DSA.

Q 1.b: Are there other digital signature schemes that should be considered for inclusion in a future revision to FIPS 186? What are the advantages of these schemes over the existing schemes in FIPS 186?
A: We are in favor of the adoption of one additional digital signature scheme. Several have been proposed, we don't have a specific preference.
The criteria for its choice we would like to suggest are:
1) To require less computation than ECDSA, such as removing the inversions.
2) To not require the storing of the complete message to be signed in the internal memory. On constrained devices, keeping the message to be signed in memory, is just infeasible.
We've referenced ECDSA, but the same arguments apply to DSA.

Q 2.a. Do the NIST-recommended curves satisfy the security requirements of applications used by industry?
A: Yes, although we would suggest that NIST reveals how the random seeds were generated.

Q 2.b. Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?
A: We are not aware of any confirmed practical or theoretical attacks on the NIST prime curves.

Q 3 a. Is there a need for new elliptic curves to be considered for standardization?

A: Yes. We suggest to add new public-trusted curves along with the FIPS 186-4 prime based curves. We do not see the need to keep the binary curves.

Q 3 b. If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?
A: The most important criteria is having a verifiable method for the generation of parameters. The method should be the result of an open discussion or competition like in the case of AES or SHA-3.

Beyond this, simplicity of secure implementations (including protection against side-channel attacks), easiness in testing (avoiding special conditions like the point at infinity), performances and implementation footprint (gates for silicon, byte code for sw and amount of temporary variables).
We have seen different entities having different definitions of security.
On one side there are experts focusing more on Mobile/PC/Server implementations; they tend to prefer curves with particular shapes and with special primes (with the term special prime we refer to primes like pseudo-Mersenne, allowing fast modular reduction). Their main goals are performance advantages and simpler implementations.
On the other side we see experts from the applicative fields where protection against side-channel and physical attacks is a must. On this side they tend to prefer prime order curves with no special structure in the prime and "random-looking" parameters (avoiding orders with a long sequence of zeros or ones), since these curves are believed to be easy to protect against side channel.
We understand both positions, and would suggest to include both, to suite everybody's needs.
The priority here is that curves' parameters are generated with a process well described and accepted by the scientific community. A public evaluation is probably the correct approach.

Q 3 c. Do you anticipate a need to create, standardize or approve new elliptic curves on an ongoing basis?
A: No. Currently there's this need, but we can't argue there will this need in the future.

Q 4 a. Which of the approved digital signature schemes and NIST-recommended curves have been used in practice?
A: Regarding signature schemes, both ECDSA and RSA are extensively used. DSA is less significantly deployed.
Regarding the curves, we have seen extensive usage of the NIST prime curves, especially P-256, P-384 and P-521.
We very rarely see the usage of B or K curves, and we would like them to be deprecated considering their current usage and the fact that new curves will be probably included in the standard.

Q 4 b. Which elliptic curves are accepted for use in international markets?
A: We have encountered oppositions to NIST curves recently.
NIST used to be trusted worldwide as cryptographic standardization entity. Open processes such as AES and SHA-3 helped to gain this trust.
Now we have seen issues in making different international parties agree on the curve to be used.

Q 5 a. If new curves were to be standardized, what would be the impact of changing existing implementations to allow for the new curves?
A: That depends.
In general, and especially if no particular optimizations are used, which is often the case in the fields where side-channel and physical attacks must be considered, the adoption of prime order curves has relatively small deployment costs.
Adding support for other elliptic curves families (e.g. Hessian, Jacobi, Montgomery, Twisted Edwards, etc.) would have a per-family increased cost. And, if curves have special primes, this will probably have an additional cost for each single prime.
On this topic it would be preferable to have either very few special prime curves added, or curves with the same special prime structure. In the latter case it would be possible to use a single functionality for the reduction, as presented by Patrick Longa this year at the NIST ECC workshop [2].

Q 5 b. What is the impact of having several standardized curves on interoperability?
A: We could have a good level of interoperability, even with several curves, if we keep them confined to different applications.
For example, we can imagine that a TLS server or an IoT device target speed and different security levels; then it would make sense to prefer curves oriented to performances and to consider side-channel only in the dimension of timing attacks.
Instead, in smart card applications, where physical attacks are possible, resistance against side-channel and fault injection attacks is a top priority. In this scenario it makes sense to stick to prime order curves with "random-looking" parameters.
NIST standard should clearly highlight the benefit of the sub-families in such a way that applicative standards select curves from the right sub-family based on security requirements of the specific application.
Supporting all the curves in all devices will add cost in term of development time, testing, code size or area for HW implementations.

Q 5 c. What are the advantages or disadvantages of allowing users or applications to generate their own elliptic curves, instead of using standardized curves?
A: We see significant disadvantages. Most likely there would be the need of a procedure for checking the parameters. Such procedure might be very expensive to fit in embedded devices.

Q 6 a. Do the performance characteristics of existing implementations of the digital signatures schemes approved in FIPS 186-4 meet the requirements of applications used by industry?
A: The requirements can be met, but if we could use faster algorithms we could meet the requirements with less costs.

Q 7 a. What are the desired intellectual property requirements for any new curves or schemes that could potentially be included in the Standard?
A: For this kind of standards it is important to get widespread adoption. We believe that royalty-free curves or schemes would greatly favor adoption.

Q 7 b. What impact has intellectual property concerns had on the adoption of elliptic curve cryptography?
A: See answer of Q 7 a.

Kind Regards,
Ruggero Susella, Guido Bertoni, Joan Daemen, Michael Peeters, Yannick Teglia, Gilles Van Assche and Sylvie Wuidart
STMicroelectronics

[1] https://www.federalregister.gov/articles/2015/10/20/2015-26539/federal-information-processing-standard-fips-186-4-digital-signature-standard-request-for-comments
[2] Slide 7 of http://csrc.nist.gov/groups/ST/ecc-workshop-2015/presentations/session6-longa-patrick.pdf

Comments on FIPS Pub 186-4

René Struik

rstruik.ext@gmail.com

Dear NIST team:

Please find below my response to NIST's request for comments on FIPS Pub 186-4 [33]. I also provided a few (mostly editorial) comments on the FIPS Pub 186-4 specification itself.

1 Response to Request for Comments

1.1 Signature Schemes

a. Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?

Note: the discussion below only relates to ECC-based signatures specified in FIPS Pub 186-4 [11, Section 6].

ECDSA (in particular, when used with NIST curve P-256) is widely used in industry. As evidenced by the wide-ranging deployment (see examples under Subsection #1.4 below), ECDSA fills a clear need of industry to provide high-strength cryptographic building blocks for applications.

b. Are there other digital signature schemes that should be considered for inclusion in a future revision to FIPS 186? What are the advantages of these schemes over the existing schemes in FIPS 186?

It may be beneficial to investigate whether inclusion of, e.g., the Schnorr signature scheme [30, Section 11.5.3],[7, Section 4.2.3]) or, e.g., EC-KCDSA [15, Section 4.4.2] as an additional signature scheme is worthwile. For a brief discussion of performance and security characteristics of ECDSA vs. Schnorr signatures, please see Subsection #1.6 (a) below.

1.2 Security of Elliptic Curves

a. Do the NIST-recommended curves satisfy the security requirements of applications used by industry?

The NIST-recommended curves (in particular, the NIST curve P-256) are widely used in industry, both with the ECDH key agreement scheme and with ECDSA. As evidenced by the wide-ranging deployment (see examples under Subsection #1.4 below), the NIST curves and related elliptic curve-based schemes fill a clear need of industry to provide high-strength cryptographic building blocks for applications.

b. Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?

To this date, no cryptographic attacks on NIST-recommended curves or other widely used curves are known that significantly improve on the classic Pollard's rho bound. In fact, there has been no significant progress on solving the discrete log problem on most curves (outside a small known class of curves that should be avoided and that can easily be identified) for at least two decades (see, e.g., [21, 23]). This being said, the last 15 years have seen a significant increase in knowledge regarding implementation attacks, including side channel and fault attacks [10, 29, 34]). This has highlighted some areas where NIST curves may be susceptible to implementation attacks (again, see, e.g., the survey paper [10]) and where, in hindsight, some domain parameters may be suboptimal. (This is not surprising, given the development of implementation security as a technical field, since the NIST curves were originally specified over 15 years ago.) To-date, expertise regarding basic countermeasures is more widely available, but not always applied well.

Should NIST decide to standardize new curves or schemes, it may prove beneficial to carefully scrutinize state-of-the-art implementation attacks and countermeasures and have these influence the selection of newly defined curves. This is the more important given the expected rise in relative prominence of small, cheap "internet of things"-style devices that can be expected to operate in a hostile environment and nevertheless require strong security provisions against passive and active implementation attacks. For those devices, minimizing the incremental cost of implementation attack countermeasures is paramount.

1.3 Elliptic Curve Specifications and Criteria

a. Is there a need for new elliptic curves to be considered for standardization?

Since the Snowden revelations, there has been some mistrust regarding NIST curves. While I agree with the assessment in [23, Section 3.1] that there is no plausible reason for this mistrust, nor any technical argument that justifies even hints of this mistrust, perceptions in the public are not easily swayed (whether fed by declining trust in government, increased belief in conspiracy theories, or other motivations). Therefore, restoration of trust in ECC may be best served by generating new elliptic curves, via a agreed-upon transparent process involving the crypto community and other stakeholders. This should not simply involve a ceremony for picking new domain parameters for short Weierstrass curves over the NIST prime fields, but start "afresh", incorporating advances in understanding of how to design curves that are secure, efficient, and easy to use, while at the same time being resilient to a wide range of implementation attacks. Thus, this effort could also be viewed as a technical update of specifications that have been extremely useful to security practitioners, to bring these in line with "state-of-the-art" (see also, e.g., [31]).

b. If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?

Picking suitable curve parameters is not easy, since requiring a careful consideration of cryptographic security, implementation security, and efficiency aspects, and balancing of these sometimes mutually conflicting design objectives. From the panel discussions at the NIST ECC workshop in June 2015 it is clear that even among experts there was no consensus on how to balance these requirements (see also, e.g., [4, 5, 12, 27]). This raises the question as to how to proceed.

In the past, the NIST ECC specifications have been widely accepted as cryptographic building blocks for use in international markets (see also Subsection #1.4 below). Assuming that NIST's objective of standardizing new curves and schemes is to prolong this role, it is of utmost importance that newly defined curves satisfy the requirements of different stakeholders reasonably well (including those whose deployments have to cater to hostile attack environments), so that these can all live with the outcome and so that no "chasm" arises.

It seems premature to define criteria NIST should use to avaluate any curves, since disagreement seems to be not so much on criteria, but on their relative importance. Some examples of questions one might pose include:

- How important is twist security, if one uses ECDH and always validates whether the key of the responding party is on the curve prior to use?
- How important are complete addition laws, e.g., in the context where one uses scalar multiplication on a point Q of order n, with scalar $0 < k < n-1$ and using Montgomery multiplication?
- In which scenarios are co-factor $h = 1$ curves important?
- With side channel attacks, how badly do repeated scalar multiplications leak, in case a scalar is lightly blinded and used once, twice, ten, or a hundred times?
- What about GLV/GLS curves using both special primes and endomorphisms (e.g., [8])? Should these be on/off the table and why?

For curves using special primes, it has been argued that, in particular contexts, this requires blinding factors of size roughly $\Omega(\sqrt{n})$ to thwart side channel attacks, where n is the bit-size of the curve in question [35]. Suppose one mandates that such curves always use blinded scalars of size $1\frac{1}{2}$ times the bit-size of the curve, should this then assuage users who care about SCA attacks? Or, should one then be concerned this could pose a moral hazard, since creating irresistable temptations to use less scalar blinding and, thereby, speed-up computations? Or, would it be better to use slightly slower curves, in exchange for decreased temptations of this type (presuming that such a sliding scale exists)?

It is not a priori clear whether all answers are already clear here (and some of these questions relate to how to promote "good behavior"). A suitable next step might therefore be to identify the (real or perceived) conflicts that underly design objectives for efficient curves and secure curves, including unknown factors (called "normality of the curve" in [12]). This should also include identifying areas of agreement (one all seems to agree that, academically, there is lots of interest in working with alternative curve forms). I believe some of these questions require more work, in order to arrive at criteria and reasonable weighings and to facilitate constructive progress towards a ceremony for selecting new curves that can be explained to the public at large.

c. Do you anticipate a need to create, standardize or approve new elliptic curves on an ongoing basis?

I do not anticipate a strong need to create new elliptic curves on an ongoing basis. Nevertheless, it seems prudent for the standardization process to assume otherwise and prepare for "algorithm agility", in terms of making it easy to cater to choice in specifications. Thus, careful reflection on, e.g., representation formats, ease of mappings between different curve forms, and consistency in protocol checks seems to be in order here. Giving the organization of specifications and extensibility sufficient thought beforehand would also help in fostering support for more than one curve, not just in specifications, but also in real-life applications, at relatively low incremental cost.

1.4 Adoption

a. Which of the approved digital signature schemes and NIST-recommended curves have been used in practice?

The NIST curve P-256 has been most widely adopted, both for use with the ECDH key agreement scheme and with ECDSA. Deployments include use with a plethora of internet protocols specified by the Internet Engineering Task Force (IETF) [6], with industry specifications in the areas of machine-to-machine communication, such as ZigBee [39], ISA [19], and Thread [37], with wireless communication protocols, such as IEEE 802.11 [18], with payment protocols, such as EMV [9], with vehicle-to-vehicle (V2V) specifications [17, 38], as well as with electronic travel documents [16] and other specifications developed under a more stringent regulatory oversight regime. Note: ISA and earlier versions of the ZigBee specification use the Koblitz curve K-283 with ECMQV; the V2V specification specifies the optional use of the German Brainpool curve BP-256 [26] as well.

The vast majority of ECC-based schemes used in practice use elliptic curves with roughly 128-bit cryptographic design strength, although the NIST curves P-384 and P-521 are also used, albeit to a lower degree. To my knowledge, the NIST curve P-224 is hardly used (except in academic papers). The NIST curve P-192 has also been used, e.g., with some versions of the Bluetooth specification, although P-192 does not meet the current cryptographic key strength recommendations of NIST SP 800-57 [32, Section 5.6.2].

b. Which elliptic curves are accepted for use in international markets?

All NIST curves specified in FIPS Pub 186-4 [11] have been widely accepted for use in international markets, where one generally uses the cryptographic key strength recommendations of NIST SP 800-57 [32, Section 5.6.2]. The Brainpool curves [26] have also seen some international deployment, although on a more limited scale. Recently, the Crypto Forum Research Group (CFRG) of the Internet Research Task Force (IRTF) has included the 255-bit Montgomery curve “Curve25519” [3] and the 448-bit Edwards curve “Goldilocks” [14] in a draft specification [25], which curves are expected to see wide-scale deployment with internet protocols, such as TLS, moving forward.

The vast majority of ECC-based schemes used in practice use elliptic curves with roughly 128-bit cryptographic design strength.

1.5 Interoperability

a. If new curves were to be standardized, what would be the impact of changing existing implementations to allow for the new curves?

Conceptually, it seems quite easy to specify the use of new curves with existing key agreement schemes and signature schemes. However, in practice, the navigability of the migration path is highly influenced by design choice details, such as representation formats, implementation flexibility, and, e.g., protocol constraints. As case in point, the potential inclusion of the curve “Curve25519”, as specified in the draft CFRG specification [25], with NIST specifications would be quite cumbersome: points there are represented in x-coordinate-only format, in lowest-bit-first order; scalar multiplications cannot easily be carried out using existing implementations in short-Weierstrass form that potentially use addition formulae that may presume the domain parameter $a = -3$, and co-factor ECDH does not check whether the resulting key is the point at infinity [24], thereby potentially necessitating additional constraints on, e.g., the key derivation function. This illustrates that special care needs to be taken with respect to specification details, so as to allow for a smooth transition, should any new curves or new curve forms be considered by NIST.

b. What is the impact of having several standardized curves on interoperability?

FIPS Pub 186-4 [11] currently already specifies multiple curves (five prime curves and ten binary curves). This does not seem to have caused any interoperability issues in practice, since most deployments gravitated towards the use of one or a few of those curves and associated key agreement and signature schemes (akin to using “profiles”). Adding more curves in a future revision of this specification does not necessarily create interoperability issues per se. However, if the objective is to foster support for more than one curve, it would be prudent to specify curves so as to allow implementation flexibility and low incremental cost of supporting more than one option. Thus, careful reflection on, e.g., representation formats, ease of mappings between different curve formats, and consistency in protocol checks seems to be in order here. This would also be prudent, since facilitating "algorithm agility".

c. What are the advantages or disadvantages of allowing users or applications to generate their own elliptic curves, instead of using standardized curves?

Picking suitable curve parameters is not easy, since requiring a careful consideration of cryptographic security, implementation security, and efficiency aspects, and balancing of these sometimes mutually conflicting design objectives. From the panel discussions at the NIST ECC workshop in June 2015 it is clear that even among experts there was no consensus on how to balance these requirements (see also, e.g., [4, 5, 12, 27]). Even if one were able to make a proper judgement, having users or applications generate their own curves seems also highly impractical, since requiring validation and since mechanical checks of cryptographic requirements alone (such as, e.g., point counting) are expensive to carry out on a device. This being said, if users are unhappy with standardized curves, they could of course generate their own curves and scrutinize their characteristics (however, this would have many characteristics of a separate standardization effort). Ideally, newly defined curves should satisfy the requirements of different stakeholders reasonably well (including those whose deployments have to cater to hostile attack environments), so that this "chasm" would not arise. Lastly, most efficient implementations exploit some characteristics of the curve in question, which would be hard(er) to predict if users or applications would pick their own favorite curves.

1.6 Performance

a. Do the performance characteristics of existing implementations of the digital signatures schemes approved in FIPS 186-4 meet the requirements of applications used by industry?

Note: the discussion below only relates to ECC-based signatures specified in FIPS Pub 186-4 [11, Section 6].

ECDSA has similar characteristics as alternative elliptic curve based signature schemes (such as, EC-KCDSA [15, Section 4.4.2] and the Schnorr signature scheme [30, Section 11.5.3],[7, Section 4.2.3]). With each of these schemes, signing requires computation of an ephemeral key $R := kG$ and computation of a signature s based on the message m to be signed, the private signer's key d, and this ephemeral key R. To-date, there seems to be sufficient expertise to implement these schemes securely, although sometimes mishaps still occur (by novice implementors with a fundamental misunderstanding of randomness requirements on the scalar k).
For some low-latency applications, such as V2V applications (see, e.g., [20]), signature verification has been deemed computationally quite expensive. Here, using new

curves with slightly faster scalar multiplication routines might have some advantages, although acceleration techniques that work with all existing NIST curves are known as well (see, e.g., [1, 36]).

From a security perspective, some have argued that the Pointcheval-Stern proofs in the random oracle model for Schnorr signatures are to be preferred over proofs using the generic group model for ECDSA. For a more detailed analysis, including discussion of ECDSA$^+$ (which hashes ephemeral key R and message m rather than the message m itself and allows reasoning about ECDSA in either model), see [22]. This is mostly an academic topic, without clear conclusion, and which (to my knowledge) has not played a significant role in industry. Moreover, this could be implemented without requiring a change to core crypto standards.

Some have argued that signing with ECDSA requires a modular inversion (modulo the order of the cyclic group), which, e.g., the Schnorr scheme does not require (although it may still require a field inversion, e.g., when computing R using projective coordinates or with the Montgomery ladder). In practice, however, this can be quite easily accommodated securely (e.g., via blinding).

Signing with ECDSA has the advantage that it allows the generation of the ephemeral key R and the hashing of the message m to be carried out independently, thereby avoiding sequential dependencies.

To summarize, the current signature schemes seem adequate. This being said, should NIST decide to see significant benefits in standardizing a new signature scheme, it may prove beneficial to take some of the above points into account.

1.7 Intellectual Property

a. What are the desired intellectual property requirements for any new curves or schemes that could potentially be included in the Standard?

Ideally, intellectual property concerns should not play any role in real-world implementation decisions of any newly specified curves or schemes. Realistically, though, intellectual property cannot be wished away, if only because of existing implementation security patents (see, e.g., the cautionary patent licensing remarks in the foreword of [29]). The most one could hope for is that this would not play any *practical* role in deployment decisions by 'benign' parties. Obviously, an implementer would wish to quantify cost, if any, in terms of bill of material (BOM), as with any other technology use decision processes in real-life products, and, ideally, would find cost (monetary and, e.g., in terms of hassle factors) to be negligible or non-existent.

b. What impact has intellectual property concerns had on the adoption of elliptic curve cryptography?

Historically, intellectual property concerns (whether perceived or real) seem to certainly have hampered widescale adoption of ECC, at least till around 2010. Moreover, there are examples of ECC specifications (outside NIST) that have been adopted, despite seemingly ignoring (or remaining silent on) essential algorithmic or implementation security checks; presumably triggered at least partially by intellectual property considerations. Some intellectual property concerns seem likely to remain, although these would perhaps shift more into the realm of implementation security patents. Hopefully, this will not provide an incentive to give nontechnical arguments precedence over considerations that foster strong security (since, then, defining new curves is moot).

2 Comments on Other Areas of FIPS Pub 186-4

Some notes on FIPS Pub 186-4 itself:

- The NIST curves are now part of the FIPS Pub 186-4 [11, Appendix D]). Since these curves are used both with signature schemes, such as ECDSA, and with key agreeement schemes, such as ECDH, it may make sense to specify these curves in a separate "curve" document, which can be cross-referenced by various "schemes" that use these curves.
- Should NIST decide to add a new signature scheme, it would be useful to follow a modular approach, where the ephemeral key $R := kG$ is always used in the same way in the signing equation, no matter whether one uses curves in short Weierstrass form, Montgomery form, or Edwards form (e.g., always use the x-coordinate of R).
- It may be useful to differentiate randomness requirements for ephemeral key pairs being used with signatures and key agreement (since the technical requirements are distinct).

References

1. A. Antipa, D.R. Brown, R. Gallant, R. Lambert, R. Struik, S.A. Vanstone, 'Accelerated Verification of ECDSA Signatures,' in *Proceedings of Selected Areas of Cryptography – SAC 2005*, B. Preneel, S. Tavares, Eds., Lecture Notes in Computer Science, Vol. 3897, pp. 307-318, Berlin: Springer, 2006.
2. D.F. Aranha, P.S.L.M. Barreto, G.C.C.E. Pereira, J.E. Ricardini, 'A Note on High-Security General-Purpose Elliptic Curves,' International Association for Cryptologic Research, ePrint 2013-647.
3. D.J. Bernstein, 'Curve25519: New Diffie-Hellman Speed Records,' in *9th International Conference on Theory and Practice of Public-Key Cryptography – PKC 2006*, M. Young, Y. Dodis, A. Kiayias, T. Malkin, Eds., Lecture Notes in Computer Science, Vol. 3958, pp. 207-228, New York, Springer, 2006.
4. D.J. Bernstein, T. Lange, 'SafeCurves: Choosing Safe Curves for Elliptic-Curve Cryptography,' Available from http://safecurves.cr.yp.to/.
5. J. Bos, 'Elliptic Curves – A Hardware Perspective,' presented at *NIST Workshop on Elliptic Curve Cryptography Standards*, Gaithersburg, MD, June 11-12, 2015.
6. J. Bos, A. Halderman, N. Heninger, J. Moore, M. Naehrig, E. Wustrow, 'Elliptic Curve Cryptography in Practice,' in *Financial Cryptography and Data Security – FC 2014*, N. Christin, R. Safavi-Naini, Eds., Lecture Notes in Computer Science, Vol. 8437, pp. 157-178, New York: Springer, 2014.

7. Bundesamt für Sicherheit in der Informationstechnik, *Technical Gideline TR-03111 – Elliptic Curve Cryptography*, Version 2.0, June 28, 2012.
8. C. Costello, P. Longa, 'Fourℚ: Four-Dimensional Decompositions On a ℚ-Curve Over the Mersenne Prime,' International Association for Cryptologic Research, ePrint 2015-565, AsiaCrypt 2015.
9. EMVCo, *EMV ECC Key Establishment Protocols*, Draft EMV-SWG-NB33r3, November 23, 2012. Available from http://www.emvco.com/specifications.aspx?id=25.
10. J. Fan, E. de Mulder, P. Schaumont, B. Preneel, I. Verbauwhede, 'State-of-the-Art of Secure ECC Implementations: A Survey on Known Side-Channel Attacks and Countermeasures,' in *3rd IEEE International Workshop on Hardware-Oriented Security and Trust - HOST 2010*, IEEE, pp. 76-87, 2010.
11. FIPS Pub 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.
12. J-P. Flori, J. Plût, J-R. Reinhard, M. Ekera, 'Diversity and Transparency for ECC,' International Association for Cryptologic Research, ePrint 2015-659.
13. S. Fluhrer, 'Scalar Blinding on Elliptic Curves with Special Structure,' International Association for Cryptologic Research, ePrint 2015-801.
14. M. Hamburg, 'Ed448-Goldilocks, a New Elliptic Curve,' International Association for Cryptologic Research, ePrint 2015-625.
15. D.R. Hankerson, A.J. Menezes, S.A. Vanstone, *Guide to Elliptic Curve Cryptography*, New York: Springer, 2003.
16. ICAO, *PKI for Machine Readable Travel Documents Offering ICC Read-Only Access*, Technical Report, Version 1.1, International Civil Aviation Organization, October 1, 2004.
17. IEEE P1609.2, 'Standard for Wireless Access in Vehicular Environments - Security Services for Applications and 4 Management Messages,' Intelligent Transportation Systems Committee, Institute of Electrical and Electronics Engineers, May 2011.
18. IEEE 802.11ai, *Standard for Information technology — Telecommunications and information exchange between systems Local and metropolitan area networks — Specific requirements, Part 11: Wireless LAN Medium Access Control(MAC) and Physical Layer (PHY) Specifications, Amendment to IEEE P802.11-REVmc/D4.0: Fast Initial Link Setup*, Draft D6.2, November 2015. Available from http://www.ieee802.org/11/private/Draft_Standards/11ai/index.html.
19. ISA/ANSI, *Wireless Systems for Industrial Automation: Process Control and Related Applications – ISA100.11a-2011 – IEC 62734*, 2011. Available from http://www.isa.org/Community/SP100WirelessSystemsforAutomation.
20. M. Knezevic, V. Nikov, P. Rombouts, 'Low-Latency ECDSA Signature Verification – A Road Towards Safe Traffic –,' International Association for Cryptologic Research, ePrint 2014-862.
21. A.H. Koblitz, N. Koblitz, A.J. Menezes, 'Elliptic Curve Cryptography, The Serpentine Course of a Paradigm Shift,' International Association for Cryptologic Research, ePrint 2008-390.
22. N. Koblitz, A.J. Menezes, 'The Random Oracle Model: A Twenty-Year Retrospective,' International Association for Cryptologic Research, ePrint 2015-140.
23. N. Koblitz, A.J. Menezes, 'A Riddle Wrapped in an Enigma,' International Association for Cryptologic Research, ePrint 2015-1018 (version December 3, 2015).
24. A. Langley, '[Cfrg] Switching the zero-check from MUST to MAY in the curves draft,' CFRG mailing list, November 16, 2015. Available from http://www.ietf.org/mail-archive/web/cfrg/current/msg07611.html.
25. A. Langley, M. Hamburg, S. Turner, 'Elliptic Curves for Security,' draft-irtf-cfrg-curves-11, October 9, 2015. Available from https://datatracker.ietf.org/doc/draft-irtf-cfrg-curves.

26. M. Lochter, J. Merkle, *Elliptic Curve Cryptography (ECC) Standard Curves and Curve Generation*, Request for Comments, RFC 5639, March 2010.
27. M. Lochter, J. Merkle, J-M. Schmidt, T. Schütze, 'Requirements for Elliptic Curves for High-Assurance Applications,' presented at *NIST Workshop on Elliptic Curve Cryptography Standards*, Gaithersburg, MD, June 11-12, 2015.
28. M. Lochter, A. Wiemers, 'Twist Insecurity,' International Association for Cryptologic Research, ePrint 2015-577.
29. S. Mangard, E. Oswald, Th. Popp, *Power Analysis Attacks – Revealing the Secrets of Smart Cards,* Springer Science+Business Media, 2007.
30. A.J. Menezes, P. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography*, Boca Raton: CRC Press, 1995.
31. M. Naehrig, 'Selecting Elliptic Curves for Cryptography – "Real-World" Issues,' UW Number Theory Seminar, Seattle, April 28, 2015. Available from http://cryptosith.org/michael/data/talks/2015-04-28-UWNumberTheorySeminar.pdf.
32. NIST SP 800-57, *Recommendation for Key Management - Part 1, General (Revision 3)*, US Department of Commerce/National Institute of Standards and Technology, Gaithersburg, MD, July 2012.
33. NIST, 'Federal Information Processing Standard (FIPS) 186-4, Digital Signature Standard; Request for Comments on the NIST-Recommended Elliptic Curves,' National Institute of Standards and Technology, October 20, 2015. Available from https://www.federalregister.gov/articles/2015/10/20/2015-26539/federal-information-processing-standard-fips-186-4-digital-signature-standard-request-for-comments.
34. W. Schindler, K. Itoh, 'Exponent Blinding Does not Automatically Lift (Partial) SPA Resistance to Higher-Level Security,' in *Applied Cryptography and Network Security – ACNS 2011*, J. Lopez, G. Tsudik, Eds., Lecture Notes in Computer Science, Vol. 6715, pp. 73-90, Berlin: Springer, 2012.
35. W Schindler, A. Wiemers, 'Efficient Side-Channel Attacks on Scalar Blinding on Elliptic Curves with Special Structure,' paper presented at *NIST Workshop on Elliptic Curve Cryptography Standards*, Gaithersburg, MD, June 11-12, 2015.
36. R. Struik, 'Batch Computations Revisited: Combining Key Computations and Batch Verifications,' in *Proceedings of Selected Areas of Cryptography – SAC 2010,* A. Biryukov, G. Gong, D.R. Stinson, Eds., Lecture Notes in Computer Science, Vol. 6544, pp. 130-142, Berlin-Heidelberg: Springer, 2011.
37. Thread Group, 'Security and Commissioning', whitepaper, July 13, 2015. Available from http://threadgroup.org/Downloads.aspx.
38. W. Whyte, A. Weimerskirch, V. Kumar, Th. Hehn 'A Security Credential Management System for V2V Communications,' IEEE Vehicular Networking Conference (VNC 2013), December 16-18, 2013, Boston, USA.
39. ZigBee Alliance, *ZigBee IP Specification*, ZigBee Public Document 13-002r00, February 2013. Available from http://www.zigbee.org/.

Hello,

Thank-you for the opportunity to provide feedback and comments on the FIPS 186. The CGI ITSETF is an NVLAP accredited Cryptographic Security Testing Laboratory (CSTL) responsible for performing FIPS 140-2 validations on cryptographic modules. Please find our feedback in addition to comments and questions from the product vendors we work with below:

1. Digital Signature Schemes

a. Do the digital signature schemes and key sizes specified in FIPS 186-4 satisfy the security requirements of applications used by industry?

CGI Response: As a CSTL our clients (product vendors) have stated that Federal users have requested key sizes larger than 3072-bits. There are many practical use cases where an appliance such as an HSM would be used to generate these larger size keys for Certificate Authorities (CAs) etc. In these cases the larger digital certificate sizes are preferred.

Also, both NIAP Common Criteria Protection Profiles and the NSA IA Suite B algorithm website now permit the use of 4096-bit keys which were originally specified in FIPS 186-2, we ask that NIST please re-introduce 4096-bit keys (and potentially larger keys) as part of the revised FIPS 186-4 standard.

b. Are there other digital signature schemes that should be considered for inclusion in a future revision to FIPS 186? What are the advantages of these schemes over the existing schemes in FIPS 186?

CGI Response: No comments.

2. Security of Elliptic Curves

a. Do the NIST-recommended curves satisfy the security requirements of applications used by industry?

CGI Response: Based on feedback from our clients (product vendors), no, the current curves do not satisfy industry security requirements as there are questions on how trustworthy they are. Given that industry cryptographers have expressed a lack of trust in the SEED parameters for the NIST curves (see URL: http://crypto.stackexchange.com/questions/10263/should-we-trust-the-nist-recommended-ecc-parameters) and the "safe curves" scorecard (see URL: http://safecurves.cr.yp.to/" indicate that the NIST prime curves are considered unsafe. We request that NIST consider allowing Federal users to utilize Curve22519, Curve1174, Curve41417, Curve448/Ed488-Goldilocks or other public curves which have been deemed safe by the industry.

A product vendor has requested: *"Can NIST please clarify why NIST has not removed P-192 from FIPS 186-4 given that NIST SP 800-131Arev1 finds it 'EC: 160 <= |n| < 224'? Can NIST please address why this curve is still included?"*

Generally speaking product vendors see no compelling reason to use the NIST-recommended curves as their preferred curves. Based on the feedback we have received product vendors are only implementing them for the purposes of obtaining a FIPS 140-2 validation.

b. Are there any attacks of cryptographic significance on Elliptic Curve Cryptography that apply to the NIST-recommended curves or other widely used curves?

CGI Response: There are concerns in the industry about the NIST-recommended curves. Please see the following slides (NOTE: the views are presented as an industry viewpoint and do not necessarily reflect CGI's opinion - we are sharing theses as a response to NIST request for comments):

http://cr.yp.to/talks/2013.05.31/slides-dan+tanja-20130531-4x3.pdf - particularly pg.6-7, 8-10, 14-17 for further discussion about the NIST parameter choices and http://cr.yp.to/talks/2013.09.16/slides-djb-20130916-a4.pdf - for the perceived security risks.

Several vendors have asked us "*Can NIST clarify who chose the seeds for the NIST curves, and how do they claim those seeds were chosen?*"

3. Elliptic Curve Specifications and Criteria

a. Is there a need for new elliptic curves to be considered for standardization?

CGI Response: Yes, we believe NIST should look to adopt some of the industry's publically available curves (such as the ones identified as “safe” on the safe curves website http://safecurves.cr.yp.to/rigid.html or in this paper https://eprint.iacr.org/2013/647.pdf) in order to give federal users a broader range of options.

b. If there is a need, what criteria should NIST use to evaluate any curves to be considered for inclusion?

CGI Response: Publically reviewed and industry accepted, general purpose curves that provide both security as well as efficiency. Public curves that are interoperable with industry standards.

c. Do you anticipate a need to create, standardize or approve new elliptic curves on an ongoing basis?

CGI Response: Yes, we believe there should be a standardized process for the review and approval of elliptic curves on an ongoing basis.

4. Adoption

a. Which of the approved digital signature schemes and NIST-recommended curves have been used in practice?

CGI Response: Most product vendors implement digital signature schemes to facilitate the use of network security protocols such as TLS, SSH and IKE.

In terms of ECDSA usage, many of our vendors support P-256, P-384 and P-521. These vendors have tested the curves under the Cryptographic Algorithm Validation Program (CAVP) with the objective of obtaining a FIPS 140-2 module validation certificate.

b. Which elliptic curves are accepted for use in international markets?

CGI Response: Generally speaking not the NIST recommended curves. Vendors we work with based in markets outside of North America implement one or more of the following public curves: M-221, E-222, Curve1174, Curve25519, E-382, M-383, Curve383187, Curve41417, Ed448-Goldilocks, M-511 or E-521. In particular Curve25519 seems to be used by many vendors.

These are believed to meet ECDLP security requirements and ECC security requirements beyond ECDLP security.

We also know of vendors based in Europe and Middle East that implement custom/private curves.

5. Interoperability

a. If new curves were to be standardized, what would be the impact of changing existing implementations to allow for the new curves?

CGI Response: From the perspective of a FIPS 140-2 validation on a cryptographic module, if vendors are granted an adequate transition period for moving to the new standards we believe the impact will be minimal. We also would request that other non-NIST recommended industry curves (ex. Curve25519) be given consideration as "allowed" functions.

b. What is the impact of having several standardized curves on interoperability?

CGI Response: More Approved Standards (and allowed Non-NIST recommended curves) are likely better than having just one standard. Provided one or more of the standardized curves map to and are interoperable with IETF or other industry standards utilizing elliptic curves.

c. What are the advantages or disadvantages of allowing users or applications to generate their own elliptic curves, instead of using standardized curves?

CGI Response: Generally speaking allowing users/applications to generate their own curves is a bad idea as the curves selected may not have been sufficiently evaluated for weaknesses.

We believe the answer is to provide users/applications a choice of several Approved Standards (NIST recommended curves) and allowed curves (Non-recommend curves which are industry accepted).

6. Performance

a. Do the performance characteristics of existing implementations of the digital signatures schemes approved in FIPS 186-4 meet the requirements of applications used by industry?

CGI Response: The digital signature schemes approved in FIPS 186-4 are adequate for applications used by industry. However, we respectfully request that larger key sizes (particularly for RSA) are put back in the Standard. This seems to be the overwhelming feedback we have received from our vendors.

Our understanding from product vendors is the NIST recommended curves NIST P-224, NIST P-256 and NIST P-384 are not their preferred curves from a performance perspective. They believe there are other non-NIST curves that better meet their security and performance needs.

7. Intellectual Property

a. What are the desired intellectual property requirements for any new curves or schemes that could potentially be included in the Standard?

CGI Response: Ideally, any new curves would be free from patents or other restrictions in order to help avoid lawsuits or patient claims against implementers.

b. What impact has intellectual property concerns had on the adoption of elliptic curve cryptography?

CGI Response: There are concerns about the implications of using curves that may be patented. In the news recently a company called CryptoPeak Solutions is suing several companies that use ECC on their website (HTTPS). See http://www.theregister.co.uk/2015/12/01/cryptopeak_sues_/.

This obviously will have an impact on public perception and adoption of ECC going forward.

Please do not hesitate to let me know if you would like to discuss any of the above comments/observations in additional detail.

Thanks again for your review and consideration.

Regards
Ryan

Ryan Thomas CISA, CISSP
FIPS 140-2 Program Manager
CGI Global IT Security Labs - Canada
1410 Blair Place, 7th floor
Ottawa, ON K1J 9B9
T: 613-234-2155
C: 613-314-7579
CGI

Dear Madam, Sir,

A comment on Appendix B5:

Embedded devices often have no decent RBG. Therefore it would be useful to add a 3rd method to calculate the ECDSA Per-Message Secret Number ‘k’:

When generating a Digital Signature, the DSA Per-Message Secret Number ‘k’ with respect to ECDSA (with the meaning in Section 4.5 of FIPS 186-4) shall be the hash of the concatenation of:

- the Message to be signed and
- the Private Key that will be used in the Digital Signature generation.

If the value of k so calculated results in an ‘r’ or ‘s’ value of 0, where r and s have the meanings in the NSA’s ‘Suite B Implementor’s Guide to FIPS 186-3’, then a new value for k shall be calculated to be the hash of the concatenation of:

- the Message to be signed;
- the Private Key that will be used in the Digital Signature generation; and
- 0x00.

The addition of 0x00 to the concatenation shall be repeated until a value of k is generated that does not result in an ‘r’ or ‘s’ value of 0.

Note: This method is specified in the UK to be used in “Smart Metering”. See the “Smart Metering Implementation Programme,
Great Britain Companion Specification (GBCS)” chapter 4.3.3.2 (issued by the “Department of Energy & Climate Change”)

Appendix D:
As NSA recommended to use only curves with a modulus of 384 bits or more, shouldn’t you add a remark that shorter moduli are not recommended anymore?

A (minor) comment on Appendix D2:
There seems a problem with the character set, there are little squares in the formulas.

Best regards, Wim Ton

Tel. +41419356401
PGP Fingerprint 9899 8071 B14A D952 6E03 8DBE 4C0F FF3B 05C0 B22F

CLASSIFICATION: UNCLASSIFIED

Thank you for receiving my comments follows.

Maintenance Agencies listed in your document are "Department of Commerce, National Institute of Standards and Technology, Information Technology Laboratory, Computer Security Division.", but I don't see any reference to the Department of Defense in your document, what level of implementation will be required on the part of the DoD to implement this change to digital signatures and should the DoD be pulled in as a stakeholder to bring the DoD to a level of awareness regarding what will be required for implementation, since enterprise wide implementations can take significant amounts of time to implement.

V/R

CPT Randy L. Williams
345th CSH Co B
USAR MS
Biomedical Health Information Systems
Cell 863-397-1488

CLASSIFICATION: UNCLASSIFIED