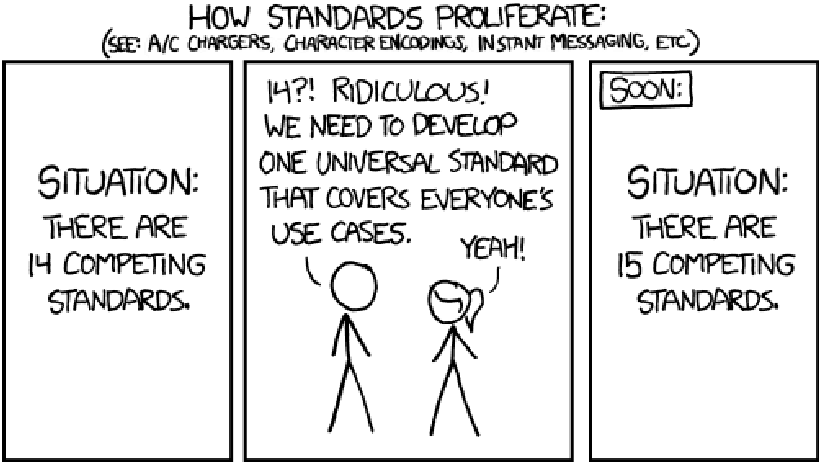
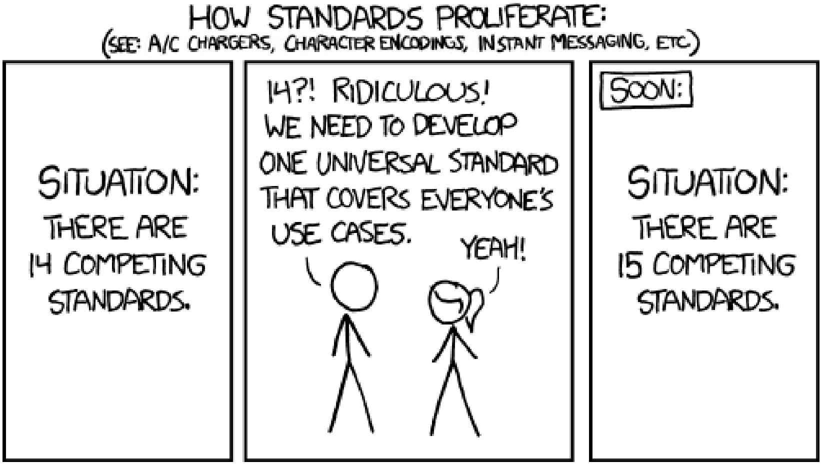
Security dangers of the NIST curves

Daniel J. Bernstein, Tanja Lange


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

http://xkcd.com/927


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

http://xkcd.com/927

But our goal today isn't unification; it's security.

"We come to bury the standard, not to praise it."

Why do people choose standardized curves?

Why do people choose standardized curves?

- "Because they told me to"
- "Need to ensure interoperability"
- "The standards committee chose these curves for efficiency"
- "The standards committee chose these curves for security"
- "Obviously these curves have received much more cryptanalysis: standard makes a worthwhile target for attack" (as in symmetric crypto)

Contrary example, Apple, "iOS Security", October 2012:

- Automatic encryption for files stored on flash on iPhones, iPads (starting with iOS 4 in 2010)
- Why use *public-key* crypto? Clear answer from Apple: "Some files may need to be written while the device is locked. A good example of this is a mail attachment downloading in the background."
- Uses ECC but not a standard NIST curve: "This behavior is achieved by using asymmetric elliptic curve cryptography (ECDH over Curve25519)."
- Same for iCloud Backup: "All the class keys in this keybag are asymmetric (using Curve25519, like the Protected Unless Open Data Protection class), so iCloud backups can be performed in the background."

Is ECC cryptanalysis actually curve-specific?

- Reality: researchers try to solve the generic ECDLP
  - Algorithm to find n given
    curve E, point P on E, point nP on E
  - Maybe algorithm performance varies with E
  - Extreme example: MOV attack works only for pairing-friendly curves
  - Standards react by prohibiting
    the *entire class* of attacked curves
    - and maybe even curves that smell similar (e.g., small-discriminant CM)

- Important example: IEEE P1363 standard
  - Surveys all ECDLP attack techniques known in 1999
  - Prohibits all curves breakable by these techniques
  - Specifies method of generating random non-prohibited curve
    - Jerry Solinas at NSA used this to generate the NIST curves
  - Were any more curves broken 1999–2013?
    - For non-prime-field ECC, yes, some advances
      - already known in 1998 to be worrisome (Frey)
      - 2006 Curve25519 paper: prime fields "have the virtue of minimizing the number of security concerns for elliptic-curve cryptography"
    - Using more computer power, yes
    - New speedups in rho algorithm, but same basic bottlenecks
  - Overall security picture unchanged for prime-field ECC

- Important example: IEEE P1363 standard
  - Surveys all ECDLP attack techniques known in 1999
  - Prohibits all curves breakable by these techniques
  - Specifies method of generating random non-prohibited curve
    - Jerry Solinas at NSA used this to generate the NIST curves (or so he says)
  - Were any more curves broken 1999–2013?
    - For non-prime-field ECC, yes, some advances
      - already known in 1998 to be worrisome (Frey)
      - 2006 Curve25519 paper: prime fields "have the virtue of minimizing the number of security concerns for elliptic-curve cryptography"
    - Using more computer power, yes
    - New speedups in rho algorithm, but same basic bottlenecks
  - Overall security picture unchanged for prime-field ECC

- Why don't we give each user a separate curve?
  - Changing curves is much cheaper than changing ciphers
  - Would be an easy defense against curve-specific cryptanalysis
    - but that's not how ECC cryptanalysis actually works
  - Really does defend against batch attacks
    - but a bigger field would provide more security at lower cost
  - Summary: not actually justified by rational cost-benefit analysis

We're writing a document "Security dangers of the NIST curves"

- Focus on the prime-field NIST curves
- DLP news relevant to these curves? No
- DLP on these curves seems really hard
- So what's the problem?
- Answer: If you implement the NIST curves, chances are you're doing it wrong
    - Your code produces incorrect results for some rare curve points
    - Your code leaks secret data when the input isn't a curve point
    - Your code leaks secret data through branch timing
    - Your code leaks secret data through cache timing
    - Even more trouble in smart cards: power, EM, etc.
- Theoretically possible to do it right, but very hard
    - Can anyone show us software for the NIST curves done right?

- These problems are exploitable by attackers
- These attacks are against real protocols, not against DLP
  - DLP is non-interactive; real protocols handle attacker-controlled input
  - DLP reveals only nP; real protocols also reveal timing
  - DLP always computes nP correctly; real protocols have failure cases
  - Attacker takes advantage of these gaps
- Many papers claim that ECC is provably secure if DLP is
  - In fact, ECC is much less secure than DLP
  - Non-interactivity etc. are oversimplifications
  - Important for cryptanalysts to study the actual systems
  - Important for cryptographers to build better systems

Critical issue: implementation.
How do ECC standards get implemented?

Critical issue: implementation.
How do ECC standards get implemented?

- Use OpenSSL library (includes, e.g., P-256)
- Use Certicom tools
- Use NaCl library (includes Curve25519)
- Have grad student write it up
- Comes with the smart card or coprocessor
- In house by the hardware designer

How has the implementation been checked/reviewed/verified?

How has the implementation been checked/reviewed/verified?

- Smart cards:
  - FIPS or CC evaluation after testing in lab
  - Tests include side-channel attacks
- Software:
  - OpenSSL is worthwhile target for security researchers
    - Continuing string of papers on cryptographic failures in OpenSSL
    - Including ECC failures
    - Many known failures are still not fixed
      - Why? Implementation difficulty, especially for constant time
      - See ECC 2012 talk by Brumley
  - NaCl is also worthwhile target
    - Benefit is increasing: wider and wider deployment
    - Cost is low: NaCl emphasizes simplicity, verifiability
    - Already some successful formal verification
      - correctness of part of the code
      - SCA resistance for part of the code

Review of basic ECC setup:

- Have group G of points on an elliptic curve
- BSGS/rho attacks work in time $O(\sqrt{\#G})$
- Typical cryptosystem works in subgroup of prime order ℓ
- Combine BSGS/rho with Pohlig–Hellman: time $O(\sqrt{\ell})$
- We avoid curves where there are better attacks
  - Side note: maybe this is excessively paranoid
    - e.g. can use curves with endomorphisms (Koblitz, GLV, GLS, etc.)
      - some security loss from endomorphisms, but seems controllable
      - pairing-based crypto needs these curves
    - But we'll stick to maximum-security curves for ECC

- How big is ℓ?
  - $\#G$ is very close to p if G is $E(\mathbf{F}_p)$
    - gap is at most $2\sqrt{p}+1$
  - ℓ is a prime divisor of $\#G$; ratio $\#G/\ell$ is “cofactor”
- In other words:
  - security forces large ℓ
  - which forces large $\#G$; exact size depends on cofactor
  - which forces large p
- Efficiency depends on p

Review of the (prime-field) NIST curves:

- Presented by NIST in 1999
- Curve names: P-192, P-224, P-256, P-384, P-521
  - Curve is defined over $\mathbf{F}_p$ where p has 192 bits, 224 bits, etc.
- Primes are pseudo-Mersenne primes:
  - e.g. P-224 prime is $2^{224} - 2^{96} + 1$
  - e.g. P-256 prime is $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$
  - Why? Efficiency
    - NSA's Jerry Solinas chose these curves and wrote papers about the speed of these primes

- Curve shape specifically $y^2 = x^3 - 3x + b$
  - About 50% of all curves
  - Absolutely nothing worrisome from an ECDLP perspective
  - “For reasons of efficiency”
    - cites IEEE P1363 standard
      - P1363 cites 1987 paper by Chudnovsky brothers
      - P1363 claims that its choices “provide the fastest arithmetic on elliptic curves”
- Cofactor choice:
  - NIST takes cofactor “as small as possible” for “efficiency reasons”
  - All cofactors for NIST curves are 1, 2, or 4
  - All cofactors for prime-field NIST curves are 1

- Protection against back doors (copied from P1363):
  - NIST publishes s where b is (basically) SHA-1(s)
  - Situation where this provides protection:
    - NSA knows a rare ECC weakness: a few weak curves $y^2 = x^3 - 3x + b$
    - NSA doesn't know how to invert SHA-1
  - But what if NSA knows a weakness in many curve choices?
    - e.g., 1/1000000000 of all curves
    - NSA searches many choices of s until finding a weak curve

Why did NIST choose these curves?

Why did NIST choose these curves?

- Most people we have asked: “security”
- Actual NIST design document: “efficiency”
- There are some minimal security requirements
  - Enough to make ECDLP hard
  - Not enough to make ECC secure
- Amusing side notes regarding efficiency:
  - addition formulas presented in standard are suboptimal, even for exactly these curves
  - NIST’s prime choices are suboptimal
  - cofactor 4 is much more efficient than cofactor 1

What goes wrong with the NIST curves? (part 1)

- Simplest scalar-multiplication inner loop: $P \leftarrow P + P$; $P \leftarrow P + Q$ if current exponent bit is set
- Huge timing channel, but that's not the only problem
- Simplest way to implement "+": use the addition formulas
  - But this doesn't work for doublings; all tests fail
  - So implementor checks book, implements $\texttt{dbl}(P)$
- New inner loop: $P \leftarrow \texttt{dbl}(P)$; $P \leftarrow P + Q$ if current exponent bit is set
- This passes all tests but still has failure cases
  - e.g., what if $P = Q$? what if $P = -Q$?
- Maybe implementor instead has "+" check for $P = Q$
  - less likely: this is slower and *more complicated* code
  - doesn't catch all the failure cases
- Attacker triggers the failure cases
  - Fancy example: Izu–Takagi "exceptional procedure attack"

Alternative: Montgomery curves $y^2 = x^3 + ax^2 + x$

- Use Montgomery ladder for scalar multiplication
  - per bit 1 doubling + 1 differential addition
  - differential addition: compute $P + Q$ given $P, Q, P - Q$
  - automatic uniform pattern independent of n; good against timing and simple side-channel attacks
- Represent a point as its x-coordinate
  - very fast doubling, very fast differential addition
  - faster scalar multiplication than $y^2 = x^3 - 3x + b$
  - for Montgomery curves that have unique point of order 2:
    - infinity and 0 behave the same way
    - the formulas *always* work (2006 Bernstein)

- Is security the same?
  - Cannot be very different
    - Every curve is a Montgomery curve over a small extension field
  - Almost half of all curves are Montgomery curves over the same field
    - Any serious attack on Montgomery curves would be huge ECC news
  - Cofactor for Montgomery curves is a multiple of 4
    - Requires slightly larger primes
- Limitation: only for single-scalar multiplication
  - signature verification needs double-scalar multiplication
  - but no problem for DH, El Gamal, etc.

Does this work for the NIST curves?

- Not easily; NIST cofactor 1 is incompatible with Montgomery
- Can still try to imitate part of the Montgomery approach
- Double and always add
  - Slow, more complicated than standard approach
  - More smart-card trouble: extra vulnerability to fault attacks
  - Can stop timing attacks but does nothing to fix failure cases
- Ladder
  - Representing point as (x, y): very slow
  - Just x: not as slow (Brier–Joye) but still complicated
  - Maybe fixes failure cases; analysis has never been done

What goes wrong with the NIST curves? (part 2)

- What if input point P is not on E but on a different curve?
- Simplest implementation doesn't check. What happens?
- Typical ECDH answer: successfully obtain nP on that other curve; use nP as shared secret to encrypt data
- Attacker chooses P so that, e.g., $1009P = 0$; checks encryption, quickly figures out n mod 1009
- Attacker figures out n by CRT
- Attack is not as easy if P is represented as just x
  - Only two possible curves: E and its "nontrivial quadratic twist"
  - 2001 Bernstein: stop attack by choosing twist to be secure
  - Might happen by accident, but random curves are usually less secure
  - NIST P-256 has a somewhat weaker twist (security 2^{121})
  - NIST P-224 has a much weaker twist (security 2^{59})

Summary so far:

- Choose Montgomery curves (with unique point of order 2)
- Represent points as x-coordinates
- In particular choose twist-secure curves
- Simple implementation is fine
- Main limitation: how to handle signatures?

Alternative: Edwards curves $x^2 + y^2 = 1 + dx^2y^2$

- Focus on *complete* Edwards curves: non-square d
  - about 25% of all elliptic curves
  - includes Curve25519; does not include the NIST curves
- Simplest addition law is *complete*
  - $x_3 = (x_1y_2 + x_2y_1)/(1 + dx_1x_2y_1y_2)$
  - $y_3 = (y_1y_2 - x_1x_2)/(1 - dx_1x_2y_1y_2)$
  - no exceptions: works for doubling, $P + (-P)$, etc.
  - easy to implement; It Just Works$^{\text{TM}}$
  - can implement separate doubling but don't have to
  - also very fast (see `http://hyperelliptic.org/EFD`)
- Guarantees Montgomery compatibility
  - easy secure single-scalar multiplication
- Also good for other ECC protocols
  - simplest signature-verification implementation is fine

Final summary:

- ECDLP security does not guarantee ECC security
- Add extra requirements on curve choices
  - Recognize the importance of friendliness to implementors
  - NIST curves cause real trouble
- Require Montgomery compatibility (NIST curves flunk)
- Require Edwards compatibility (NIST curves flunk)
- Require completeness (NIST curves flunk)
- Require twist security (NIST curves are weak)
- Easy to generate curves meeting all these requirements: Curve25519, Curve1174, etc.