

The first 10 years of Curve25519

Daniel J. Bernstein

University of Illinois at Chicago &
Technische Universiteit Eindhoven

2005.05.19: [Seminar talk](#);
design+software close to done.

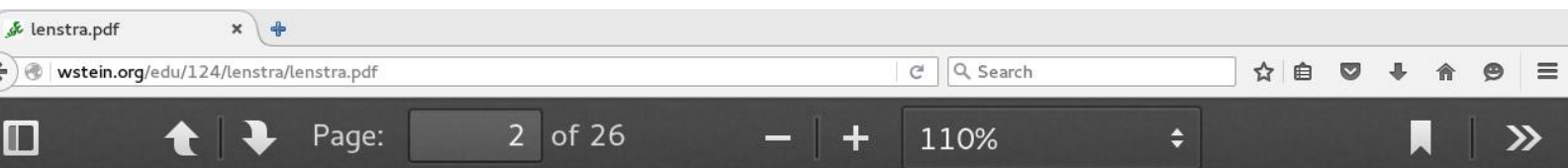
2005.09.15: [Software online](#).

2005.09.20: [Invited talk at ECC](#).

2005.11.15: [Paper online](#);
submitted to PKC 2006.

Abstract: “This paper explains the design and implementation of a high-security elliptic-curve-Diffie-Hellman function achieving record-setting speeds: e.g., 832457 Pentium III cycles (with several side benefits: free key compression, free key validation, and state-of-the-art timing-attack protection), more than twice as fast as other authors’ results at the same conjectured security level (with or without the side benefits).”

Elliptic-curve computations



Annals of Mathematics, **126** (1987), 649–673

Factoring integers with elliptic curves

By H. W. LENSTRA, JR.

Abstract

This paper is devoted to the description and analysis of a new algorithm to factor positive integers. It depends on the use of elliptic curves. The new method is obtained from Pollard's $(p - 1)$ -method (Proc. Cambridge Philos. Soc. **76** (1974), 521–528) by replacing the multiplicative group by the group of points on a random elliptic curve. It is conjectured that the algorithm determines a non-trivial divisor of a composite number n in expected time at most $K(p)(\log n)^2$, where p is the least prime dividing n and K is a function for which $\log K(x) = \sqrt{(2 + o(1)) \log x \log \log x}$ for $x \rightarrow \infty$. In the worst case, when n is the product of two primes of the same order of magnitude, this is $\exp((1 + o(1))\sqrt{\log n \log \log n})$ (for $n \rightarrow \infty$). There are several other factoring algorithms of which the conjectural expected running time is given by the latter formula. However, these algorithms have a running time that is basically independent of the size of the prime factors of n , whereas the new elliptic curve method is substantially faster for small p .

Acknowledgements. This paper was written at the Mathematical Sciences

1987 (distributed 1984) Lenstra:
ECM, the elliptic-curve method
of factoring integers.

1985 Bosma, 1986 Goldwasser–
Kilian, 1986 Chudnovsky–
Chudnovsky, 1988 Atkin: ECPP,
elliptic-curve primality proving.

1985/6 (distributed 1984) Miller,
and independently

1987 (distributed 1984) Koblitz:
ECC—use elliptic curves in DH
to avoid index-calculus attacks.

1986 Chudnovsky–Chudnovsky,
for ECM+ECPP: analyze several
ways to represent elliptic curves;
optimize # field operations.

1986 Chudnovsky–Chudnovsky,
for ECM+ECPP: analyze several
ways to represent elliptic curves;
optimize # field operations.

1987 Montgomery, for ECM:
best speed from $y^2 = x^3 + Ax^2 + x$,
preferably with $(A - 2)/4$ small.

1986 Chudnovsky–Chudnovsky,
for ECM+ECPP: analyze several
ways to represent elliptic curves;
optimize # field operations.

1987 Montgomery, for ECM:
best speed from $y^2 = x^3 + Ax^2 + x$,
preferably with $(A - 2)/4$ small.

Late 1990s: ANSI/IEEE/NIST
standards specify $y^2 = x^3 - 3x + b$
in Jacobian coordinates,
citing Chudnovsky–Chudnovsky.
Alleged motivation: “the fastest
arithmetic on elliptic curves” .

Did Chudnovsky and Chudnovsky
actually recommend this?

What about Montgomery?

What about papers after 1987?

Did Chudnovsky and Chudnovsky actually recommend this?

What about Montgomery?

What about papers after 1987?

Analyze all known options

for computing $n, P \mapsto nP$

on conservative elliptic curves.

Montgomery ladder is the fastest.

Did Chudnovsky and Chudnovsky actually recommend this?

What about Montgomery?

What about papers after 1987?

Analyze all known options

for computing $n, P \mapsto nP$

on conservative elliptic curves.

Montgomery ladder is the fastest.

Problem: Elliptic-curve formulas always have exceptional cases.

Montgomery derives formulas for *generic* inputs; for crypto we need algorithms that *always* work.

Complete Systems of Two Addition Laws for Elliptic Curves

W. BOSMA*

*Department of Pure Mathematics, University of Sydney,
Sydney, New South Wales 2006, Australia*

AND

H. W. LENSTRA, JR.†

*Department of Mathematics, University of California,
Berkeley, California 94720-3840*

laws on E exists. indeed, a complete system of three addition laws, each consisting of bihomogeneous polynomials of bidegree $(2, 2)$, was exhibited explicitly by Lange and Ruppert [2; cf. 1]. In the present paper we show that there are complete systems consisting of *two* addition laws, and that both addition laws in such a system are necessarily of bidegree $(2, 2)$.

THEOREM 1. *The smallest cardinality of a complete system of addition laws on E equals two, and if two addition laws form a complete system then each of them has bidegree $(2, 2)$.*

We can describe all addition laws of bidegree $(2, 2)$. To do this, we omit the zero addition law, for which *all* pairs P_1, P_2 are exceptional, and we call two addition laws *equivalent* if there exists a non-zero element $d \in k$

But wait, it's worse!

Crypto 1996 Kocher:
secret branches affect timing;
this leaks your secret key.

But wait, it's worse!

Crypto 1996 Kocher:
secret branches affect timing;
this leaks your secret key.

Briefly mentioned by Kocher
and by ESORICS 1998 Kelsey–
Schneier–Wagner–Hall:
secret array indices can affect
timing via cache misses.

2002 Page, CHES 2003 Tsunoo–
Saito–Suzaki–Shigeri–Miyachi:
timing attacks on DES.

“Guaranteed” countermeasure:
load entire table into cache.

“Guaranteed” countermeasure:
load entire table into cache.

2004.11/2005.04 Bernstein:

Timing attacks on AES.

Countermeasure isn't safe;

e.g., secret array indices can affect
timing via cache-bank collisions.

What *is* safe: kill all data flow
from secrets to array indices.

“Guaranteed” countermeasure:
load entire table into cache.

2004.11/2005.04 Bernstein:

Timing attacks on AES.

Countermeasure isn't safe;

e.g., secret array indices can affect
timing via cache-bank collisions.

What *is* safe: kill all data flow
from secrets to array indices.

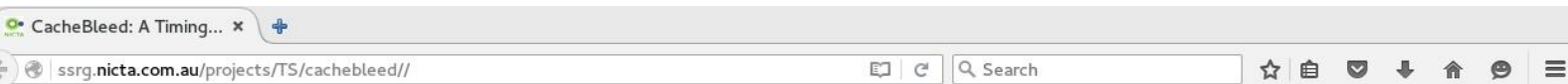
2013 Bernstein–Schwabe

“A word of warning” :

Cheaper countermeasure

recommended by Intel isn't safe.

2016: OpenSSL didn't listen.



CacheBleed: A Timing Attack on OpenSSL Constant Time RSA

**Yuval
Yarom**

The University of
Adelaide and
NICTA

**Daniel
Genkin**

Technion and Tel
Aviv University

**Nadia
Heninger**

University of
Pennsylvania

Overview

CacheBleed is a side-channel attack that exploits information leaks through cache-bank conflicts in Intel processors. By detecting cache-bank conflicts via minute timing variations, we are able to recover information about victim processes running on the same machine. Our attack is able to recover both 2048-bit and 4096-bit RSA secret keys from OpenSSL 1.0.2f running on Intel Sandy Bridge processors after observing only 16,000 secret-key operations (decryption, signatures). This is despite the fact that OpenSSL's RSA implementation was carefully designed to be constant time in order to protect against cache-based (and other) side-channel attacks.

While the possibility of an attack based on cache-bank conflicts has long been speculated, this is the first practical demonstration of such an attack. Intel's technical documentation describes cache-bank conflicts as early as 2004. However, these were not widely thought to be exploitable, and as a consequence common cryptographic software developers have not implemented countermeasures to this attack.

Paper

Latest version can be downloaded [here](#).

The Curve25519 paper

Avoid “all input-dependent branches, all input-dependent array indices, and other instructions with input-dependent timings” .

The Curve25519 paper

Avoid “all input-dependent branches, all input-dependent array indices, and other instructions with input-dependent timings”.

Choose a curve $y^2 = x^3 + Ax^2 + x$ where $A^2 - 4$ is not a square.

≈25% of all elliptic curves.

The Curve25519 paper

Avoid “all input-dependent branches, all input-dependent array indices, and other instructions with input-dependent timings”.

Choose a curve $y^2 = x^3 + Ax^2 + x$ where $A^2 - 4$ is not a square.

$\approx 25\%$ of all elliptic curves.

Define $X_0(x, y) = x$; $X_0(\infty) = 0$.

Transmit each point P as $X_0(P)$.

The Curve25519 paper

Avoid “all input-dependent branches, all input-dependent array indices, and other instructions with input-dependent timings” .

Choose a curve $y^2 = x^3 + Ax^2 + x$ where $A^2 - 4$ is not a square.

$\approx 25\%$ of all elliptic curves.

Define $X_0(x, y) = x$; $X_0(\infty) = 0$.

Transmit each point P as $X_0(P)$.

Use the Montgomery ladder
without any extra tests.

The Curve25519 paper

Avoid “all input-dependent branches, all input-dependent array indices, and other instructions with input-dependent timings” .

Choose a curve $y^2 = x^3 + Ax^2 + x$ where $A^2 - 4$ is not a square.

$\approx 25\%$ of all elliptic curves.

Define $X_0(x, y) = x$; $X_0(\infty) = 0$.

Transmit each point P as $X_0(P)$.

Use the Montgomery ladder
without any extra tests.

Theorem: Output is $X_0(nP)$.

$x_2, z_2, x_3, z_3 = 1, 0, x_1, 1$

for i in reversed(range(255)):

$bit = 1 \& (n \gg i)$

$x_2, x_3 = cswap(x_2, x_3, bit)$

$z_2, z_3 = cswap(z_2, z_3, bit)$

$x_3, z_3 = ((x_2 * x_3 - z_2 * z_3)^2,$
 $x_1 * (x_2 * z_3 - z_2 * x_3)^2)$

$x_2, z_2 = ((x_2^2 - z_2^2)^2,$

$4 * x_2 * z_2 * (x_2^2 + A * x_2 * z_2 + z_2^2))$

$x_2, x_3 = cswap(x_2, x_3, bit)$

$z_2, z_3 = cswap(z_2, z_3, bit)$

return $x_2 * z_2^{(p-2)}$

Montgomery has variable #loops,
depending on top bit of n .

Montgomery has variable `#loops`, depending on top bit of n .

Curve25519: Change initialization to allow leading 0 bits.

Use constant `#loops`.

Montgomery has variable $\#$ loops, depending on top bit of n .

Curve25519: Change initialization to allow leading 0 bits.

Use constant $\#$ loops.

Also define scalars n

to never have leading 0 bits,

so original Montgomery ladder still takes constant time.

Montgomery has variable $\#$ loops, depending on top bit of n .

Curve25519: Change initialization to allow leading 0 bits.

Use constant $\#$ loops.

Also define scalars n to never have leading 0 bits, so original Montgomery ladder still takes constant time.

Use arithmetic to compute c_{swap} in constant time.

“Hey, you forgot to check that the input is on the curve!”

“Hey, you forgot to check that the input is on the curve!”

Conventional wisdom: Important to check; otherwise broken by Crypto 2000 Biehl–Meyer–Müller.

“Hey, you forgot to check that the input is on the curve!”

Conventional wisdom: Important to check; otherwise broken by Crypto 2000 Biehl–Meyer–Müller.

ESORICS 2015 Jager–Schwenk–Somorovsky: Successful attacks! Checking is easy to forget.

The screenshot shows a web browser window with the following content:

- Browser Tab:** EIT Publications - Ruhr-U...
- Address Bar:** https://www.nds.ruhr-uni-bochum.de/research/publications/ESORICS15/
- Page Header:**
 - RUHR-UNIVERSITÄT BOCHUM
 - CHAIR FOR NETWORK AND DATA SECURITY
 - EIT FORSCHUNG
 - A-Z | OVERVIEW | SEARCH | CONTACT
 - RUB
- Breadcrumbs:** RUB » EI » NDS » Forschung » Publications
- Left Sidebar (LEHRSTUHL / LEHRE):**
 - Best Student Paper Award
 - HackerPraktikum
 - HackPra Allstars
 - Former speakers
 - courses
- Main Content:**
 - PRACTICAL INVALID CURVE ATTACKS ON TLS-ECDH**
 - Tibor Jager, Jörg Schwenk, Juraj Somorovsky
 - ESORICS 2015
 - ABSTRACT**
 - Elliptic Curve Cryptography (ECC) is based on cyclic groups, where group elements are represented as points in a finite plane. All ECC cryptosystems implicitly assume that only valid group elements will be processed by the different cryptographic algorithms. It is well-known that a check for group membership of given points in the plane should be performed before processing. However, in several widely used cryptographic libraries we analyzed, this check was missing, in particular in the popular ECC implementations

Curve25519 paper:

“free key validation”

eliminates these attacks.

No cost for checking input;

no code to forget.

Curve25519 paper:

“free key validation”

eliminates these attacks.

No cost for checking input;

no code to forget.

1. Montgomery naturally

follows 1986 Miller compression:

send only x -coordinate, not (x, y) .

Forces input onto “curve” or

“twist”. (Bonus: 32-byte keys!)

Curve25519 paper:

“free key validation”

eliminates these attacks.

No cost for checking input;

no code to forget.

1. Montgomery naturally follows 1986 Miller compression: send only x -coordinate, not (x, y) . Forces input onto “curve” or “twist”. (Bonus: 32-byte keys!)

2. Montgomery ladder works correctly for inputs on twist.

Curve25519 paper:

“free key validation”

eliminates these attacks.

No cost for checking input;

no code to forget.

1. Montgomery naturally follows 1986 Miller compression: send only x -coordinate, not (x, y) . Forces input onto “curve” or “twist”. (Bonus: 32-byte keys!)
2. Montgomery ladder works correctly for inputs on twist.
3. Choose twist-secure curve.

Longest section in Curve25519
paper: fast finite-field arithmetic,
improving on algorithm designs
from 1999–2004 Bernstein.

Longest section in Curve25519
paper: fast finite-field arithmetic,
improving on algorithm designs
from 1999–2004 Bernstein.

Barely mentioned in paper:
new programming language.

Longest section in Curve25519 paper: fast finite-field arithmetic, improving on algorithm designs from 1999–2004 Bernstein.

Barely mentioned in paper: new programming language.

New prime $2^{255} - 19$.

Faster than NIST P-256 prime $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

“Prime fields also have the virtue of minimizing the number of security concerns for elliptic-curve cryptography.”

Curve25519 paper specified a **multi-user** DH system. See 1976 Diffie–Hellman; also, e.g., 1999 Rescorla “static-static mode”; 2006 NIST “C(0,2)” .

Curve25519 paper specified a **multi-user** DH system. See 1976 Diffie–Hellman; also, e.g., 1999 Rescorla “static-static mode”; 2006 NIST “C(0,2)”.

Included security survey:

- Reductions: intolerably loose.
- Known attack ideas: rho etc.
- Multi-user batch attacks.
- Special-purpose hardware:
160-bit ECC is breakable.
- Small-subgroup attacks,
invalid-curve attacks, etc.

2015: Beware batch attacks.



Weak Diffie-Hellman and the Logjam Attack

Good News! Your browser is safe against the Logjam attack.

Diffie-Hellman key exchange is a popular cryptographic algorithm that allows Internet protocols to agree on a shared key and negotiate a secure connection. It is fundamental to many protocols including HTTPS, SSH, IPsec, SMTPS, and protocols that rely on TLS.

We have uncovered several weaknesses in how Diffie-Hellman key exchange has been deployed:

- 1. Logjam attack against the TLS protocol.** The Logjam attack allows a man-in-the-middle attacker to downgrade vulnerable TLS connections to 512-bit export-grade cryptography. This allows the attacker to read and modify any data passed over the connection. The attack is reminiscent of the [FREAK attack](#), but is due to a flaw in the TLS protocol rather than an implementation vulnerability, and attacks a Diffie-Hellman key exchange rather than an RSA key exchange. The attack affects any server that supports DHF EXPORT

Paper sketched common-sense attack model, including composition with subsequent multi-user secret-key system (as in, e.g., 2001 Bernstein “public-key authenticators”); attacks on secret-key system (the motivation given for “Reveal” queries in PKC 2013 Freire–Hofheinz–Kiltz–Paterson); dishonest key registrations (as in, e.g., Eurocrypt 2008 Cash–Kiltz–Shoup); keys as strings (allows modeling, e.g., 2000 Biehl–Meyer–Müller).

PKC 2006

April 24-26, 2006
New York City, USA



[front page](#)

[call for papers](#)

[local information](#)

[registration](#)

[program](#)

[contact](#)

[golden sponsors](#)

PKC 2006

**9th INTERNATIONAL CONFERENCE ON THEORY AND PRACTICE OF
PUBLIC KEY CRYPTOGRAPHY**

NEW YORK

APRIL 24-26



The International Conference on Theory and Practice of Public-Key Cryptography (PKC) has been the main IACR annual workshop focusing on all aspects of public-key cryptography. PKC has attracted papers from world-renowned scientists in the area. The Proceedings of PKC'06 will be published by Springer-Verlag in the Lecture Notes in Computer Science (LNCS) series.

PKC'06 will be hosted by Columbia University and will take place at the Davis Auditorium on the 4th floor (campus level) of the Schapiro CEPSR Building at Columbia University, in New York City.

[[Sponsors](#)]



Morgan Stanley

silver sponsors



GEMPLUS™

Google™

Microsoft®

Email from program chairs:

It is my pleasure to inform you that your paper "Curve25519: new Diffie-Hellman speed records" was accepted to PKC'06. Congratulations!

Email from program chairs:

It is my pleasure to inform you that your paper "Curve25519: new Diffie-Hellman speed records" was accepted to PKC'06. Congratulations!

Below please find the reviewers' comments on your paper "Curve25519: new Diffie-Hellman speed records" that was submitted to PKC 2006.

Reviewer #1:

While I think (frankly) that this is a nice engineering work, I think that this is not a "real" research paper.

I don't question the correctness but I question the appropriateness of the paper to the conference.

So engineering isn't research?

Reviewer #2:

... benefits including protection against timing attacks, no apparrent patent infringements, and very good speed. ...

On the negative side, the paper does not introduce novel ideas, nor does it attempt to prove things rigorously (the word "conjecture" is used repeatedly throughout). It is principally a considerable engineering achievement.

e.g. “Breaking the Curve25519 function—for example, computing the shared secret from the two public keys—is conjectured to be extremely difficult. Every known attack is more expensive than performing a brute-force search on a typical 128-bit secret-key cipher. . . . Curves of this shape have order divisible by 4, requiring a marginally larger prime for the same conjectured security level, but this is outweighed by the extra speed of curve operations.”

Reviewer #3:

... The curve and the field are hardwired into the program, which leaves little flexibility if changes are someday needed.

... My main concerns about the paper are that it comes across as low on useful content (it's mostly about one curve), and is very strangely written, and therefore unpleasant to read ...

The paper is written in what

comes across as a rambling incoherent style. ... The rewriting that would be required to make this paper readable is significant (though easy for someone willing to do it), and I'm not optimistic that it would be done by the deadline, or that the content (I can't say "results" since there aren't any stated results, other than a trivial mathematical result) is significant enough to justify

acceptance. . . . The "Conjectured Curve25519 security level" section should be omitted; or if there's useful and new content in it, that should be made clear. . . . Most of the appendices should be removed. For example, the irrelevant discussion of patents should either be removed, or rephrased to be a purely scientific discussion and not a patent discussion, and the appendix

that shows that 3 numbers are prime should be removed. ...

The paper will be of greatest interest to those implementing Diffie-Hellman with elliptic curves. But the limitations on the exponent (and the lack of a y -coordinate) prevent it from being used by El Gamal and other ECC protocols. ...

that shows that 3 numbers are prime should be removed. ...

The paper will be of greatest interest to those implementing Diffie-Hellman with elliptic curves. But the limitations on the exponent (and the lack of a y-coordinate) prevent it from being used by El Gamal and other ECC protocols. ... The paper is remarkably free of grammatical errors.

2016: Counterfeit “primes” .

[MAIN MENU](#)[MY STORIES: 25](#)[FORUMS](#)[SUBSCRIBE](#)[JOBS](#)

RISK ASSESSMENT / SECURITY & HACKTIVISM

Crypto flaw was so glaring it may be intentional eavesdropping backdoor

Network tool contained hard-coded prime number that wasn't prime after all.

by **Dan Goodin** - Feb 2, 2016 1:16pm CST

[Share](#)[Tweet](#)[Email](#)

With reviews like these,
how did PKC accept Curve25519?

With reviews like these,
how did PKC accept Curve25519?

Reviewer #4 was positive.

Maybe reviewer #4 convinced
other people as part of discussion.

Or program chairs liked paper.

With reviews like these,
how did PKC accept Curve25519?

Reviewer #4 was positive.

Maybe reviewer #4 convinced
other people as part of discussion.

Or program chairs liked paper.

Maybe someone thought the title
“9th International Conference on
Theory and Practice in Public-
Key Cryptography” justified
an occasional paper like this.

With reviews like these,
how did PKC accept Curve25519?

Reviewer #4 was positive.

Maybe reviewer #4 convinced
other people as part of discussion.
Or program chairs liked paper.

Maybe someone thought the title
“9th International Conference on
Theory and Practice in Public-
Key Cryptography” justified
an occasional paper like this.

Note to young cryptographers:
Don't let referees discourage you.

Edwards curves

2007 Edwards “A
normal form for elliptic curves” :

$$x_3 = \frac{x_1 y_2 + x_2 y_1}{c(1 + x_1 x_2 y_1 y_2)},$$

$$y_3 = \frac{y_1 y_2 - x_1 x_2}{c(1 - x_1 x_2 y_1 y_2)}$$

generically defines addition law

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

on any elliptic curve of the form

$$x^2 + y^2 = c^2(1 + x^2 y^2).$$

Euler+Gauss defined this law

for one curve: $c^4 = -1$.

2007 Bernstein–Lange “Faster addition and doubling on elliptic curves”: Edwards addition law easily generalizes to

$$x_3 = \frac{x_1 y_2 + x_2 y_1}{1 + d x_1 x_2 y_1 y_2},$$

$$y_3 = \frac{y_1 y_2 - x_1 x_2}{1 - d x_1 x_2 y_1 y_2}.$$

on any elliptic curve of the form $x^2 + y^2 = 1 + d x^2 y^2$.

$d = c^4$ is original Edwards.

$d = 0$ is circle, non-elliptic.

2007 Bernstein–Lange “Faster addition and doubling on elliptic curves”: Edwards addition law easily generalizes to

$$x_3 = \frac{x_1 y_2 + x_2 y_1}{1 + d x_1 x_2 y_1 y_2},$$

$$y_3 = \frac{y_1 y_2 - x_1 x_2}{1 - d x_1 x_2 y_1 y_2}.$$

on any elliptic curve of the form $x^2 + y^2 = 1 + d x^2 y^2$.

$d = c^4$ is original Edwards.

$d = 0$ is circle, non-elliptic.

Surprise for non-square d :

this addition law is **complete!**

By easy change of coordinates
can write $y^2 = x^3 + Ax^2 + x$
with non-square $A^2 - 4$
as a complete Edwards curve.
In particular: Curve25519.

By easy change of coordinates
can write $y^2 = x^3 + Ax^2 + x$
with non-square $A^2 - 4$
as a complete Edwards curve.
In particular: Curve25519.

Curve arithmetic is very fast.

(After various followup papers:
even faster!)

Almost as fast as Montgomery
for $n, P \mapsto nP$ in DH.

New speed records for
 $m, n, P, Q \mapsto mP + nQ$
and other signature operations.

The Ed25519 signature system

CHES 2011 Bernstein–Duif–
Lange–Schwabe–Yang:

Start from Schnorr signatures.

Skip signature compression.

Support batch verification.

Use double-size H output, and
include public key A as input:

$$SB = R + H(R, A, M)A.$$

Generate R deterministically
as a secret hash of M .

⇒ Avoid PlayStation disaster.

Use Curve25519 in complete
“–1-twisted” Edwards form.

Optimizations for more platforms

2007 Gaudry–Thomé: Core 2.

2009 Costigan–Schwabe: Cell.

2011 Bernstein–Duif–Lange–
Schwabe–Yang: Nehalem.

2012 Bernstein–Schwabe: NEON.

2014 Langley–Moon: newer Intel.

2014 Mahé–Chauvet: GPUs.

2014 Sasdrich–Güneysu: FPGAs.

2015 Chou: newer Intel.

2015 Düll–Haase–Hinterwälder–
Hutter–Paar–Sánchez–Schwabe:
microcontrollers.

2015 Hutter–Schilling–Schwabe–
Wieser: ASICs.

Next-generation crypto library

NaCl: Networking and Cryptography library provides very simple new API for public-key authenticated encryption.

All-in-one `crypto_box` function uses Curve25519 for DH, Salsa20 for encryption, Poly1305 for authentication.

More on NaCl design: see 2011 Bernstein–Lange–Schwabe “The security impact of a new cryptographic library” .

Simplicity

Curve25519 paper
advertised “short code.”

2013 Bernstein–Janssen–
Lange–Schwabe: [TweetNaCl](#),
reimplementing NaCl in 100
tweets. Does speed matter?

Simplicity

Curve25519 paper
advertised “short code.”

2013 Bernstein–Janssen–
Lange–Schwabe: [TweetNaCl](#),
reimplementing NaCl in 100
tweets. Does speed matter?

Largest chunk of code: The hash
function used inside signatures!

Simplicity

Curve25519 paper
advertised “short code.”

2013 Bernstein–Janssen–
Lange–Schwabe: [TweetNaCl](#),
reimplementing NaCl in 100
tweets. Does speed matter?

Largest chunk of code: The hash
function used inside signatures!

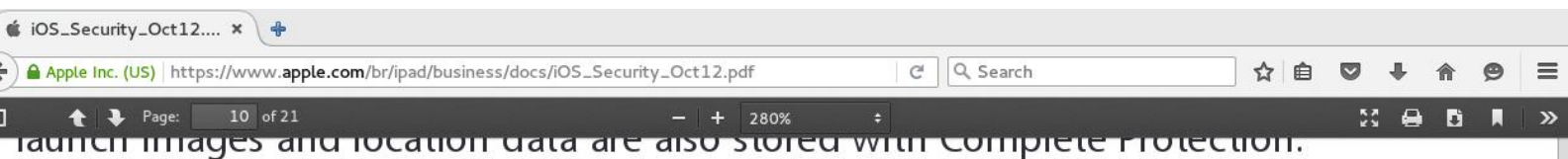
2014 [Bernstein–van Gastel–
Janssen–Lange–Schwabe–
Smetsers](#): formal verification of
some TweetNaCl properties.

2014 Chen–Hsu–Lin–Schwabe–
Tsai–Wang–Yang–Yang “[Verifying
Curve25519 software](#)”: formal
verification of **correctness** of
two high-speed asm main loops.

Newer work ongoing: e.g., 2015
Russinoff “[A computationally
surveyable proof of the
Curve25519 group axioms](#)”; 2015
Bernstein–Schwabe [gfverif](#).

Single-curve code helps speed
and is the most promising avenue
towards bug-free ECC software.

2012: Apple deploys Curve25519



Protected Unless Open

(NSFileProtectionCompleteUnlessOpen): 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. This behavior is achieved by using asymmetric elliptic curve cryptography (ECDH over Curve25519). Along with the usual per-file key, Data Protection generates a file public/private key pair. A shared secret is computed using the file's private key and the Protected Unless Open class public key, whose corresponding private key is protected with the user's passcode and the device UID. The per-file key is wrapped with the hash of this shared secret and stored in the file's metadata along with the file's public key; the corresponding private key is then wiped from memory. As soon as the file is closed, the per-file key is also wiped from memory. To open the file again, the shared secret is re-created using the Protected Unless Open class's private key and the file's ephemeral public key; its hash is used to unwrap the per-file key, which is then used to decrypt the file.

Protected Until First User Authentication

(NSFileProtectionCompleteUntilFirstUserAuthentication): This class behaves in the same way as Complete Protection, except that the decrypted class key is not removed from memory when the device is locked. The protection in this class has similar properties to desktop full-disk encryption, and protects data from attacks that involve a reboot.

No Protection

(NSFileProtectionNone): This class key is protected only with the UID, and is kept in Effaceable Storage. This is the default class for all files not otherwise assigned to a Data Protection class. Since all the keys needed to decrypt files in this class are stored on the device, the encryption only affords the benefit of fast remote wipe. If a file is not assigned a Data Protection class, it is still stored in encrypted form (as is all data

2013: Signal deploys Curve25519

Migrate to Curve255... x

GitHub, Inc. (US) https://github.com/WhisperSystems/Signal-Android/commit/c3c6fd2d4fc62c8a369 Search

GitHub


This repository Search Explo

WhisperSystems / Signal-Android

<> Code Issues 613 Pull requests 28 Wiki Pulse

Migrate to Curve25519.

- 1) Generate a Curve25519 identity key.
- 2) Use Curve25519 ephemerals and identities for v2 3DHE agreement.
- 3) Initiate v2 key exchange messages.
- 4) Accept v1 key exchange messages.
- 5) TOFU Curve25519 identities.

 **moxie0** committed on Nov 10, 2013

Showing **57 changed files** with **2,194 additions** and **495 deletions**.

2014: OpenSSH deploys Curve25519

http://www...lease-6.5 x

www.openssh.com/txt/release-6.5

Search

Changes since OpenSSH 6.4

=====

This is a feature-focused release.

New features:

- * ssh(1), sshd(8): Add support for key exchange using elliptic-curve Diffie Hellman in Daniel Bernstein's Curve25519. This key exchange method is the default when both the client and server support it.
- * ssh(1), sshd(8): Add support for Ed25519 as a public key type. Ed25519 is a elliptic curve signature scheme that offers better security than ECDSA and DSA and good performance. It may be used for both user and host keys.
- * Add a new private key format that uses a bcrypt KDF to better protect keys at rest. This format is used unconditionally for Ed25519 keys, but may be requested when generating or saving existing keys of other types via the `-o ssh-keygen(1)` option. We intend to make the new format the default in the near future. Details of the new format are in the `PROTOCOL.key` file.
- * ssh(1), sshd(8): Add a new transport cipher "chacha20-poly1305@openssh.com" that combines Daniel Bernstein's ChaCha20 stream cipher and Poly1305 MAC to build an authenticated encryption mode. Details are in the `PROTOCOL.chacha20poly1305` file.
- * ssh(1), sshd(8): Refuse RSA keys from old proprietary clients and servers that use the obsolete RSA+MD5 signature scheme. It will still be possible to connect with these clients/servers but only DSA keys will be accepted, and OpenSSH will refuse connection entirely in a future release.
- * ssh(1), sshd(8): Refuse old proprietary clients and servers that use a weaker key exchange hash calculation.

2015.10: IRTF CFRG settles on EdDSA—Ed25519 and Ed448—for signatures. Already selected X25519 and X448 for DH.

2015.10: NIST reopens its ECC standards for comment, paving way for new curves.

2015.11: BoringSSL adds X25519 and Ed25519.

These are just some highlights.

Many more: ianix.com/pub/curve25519-deployment.html and [/ed25519-deployment.html](http://ianix.com/pub/ed25519-deployment.html).