BADA55, Curve41417, Kummer

Daniel J. Bernstein
University of Illinois at Chicago
and Technische Universiteit Eindhoven

Joint work with:
Tung Chou (BADA55)
Chitchanok Chuengsatiansup (BADA55, Curve41417, Kummer)
Andreas Hülsing (BADA55)
Tanja Lange (BADA55, Curve41417, Kummer)
Ruben Niederhagen (BADA55)
Peter Schwabe (Kummer)
Christine van Vredendaal (BADA55)

NSA/NIST FUD

The NIST elliptic curves are behind the state of the art:

- Chosen by Jerry Solinas at **NSA**.

NSA/NIST FUD

The NIST elliptic curves are behind the state of the art:

- Chosen by Jerry Solinas at **NSA**.
- Coefficients produced from NSA's **SHA-1**.

NSA/NIST FUD

The NIST elliptic curves are behind the state of the art:

- Chosen by Jerry Solinas at **NSA**.
- Coefficients produced from NSA's **SHA-1**.
- NIST P-224 is **not twist-secure**.
- etc.

NIST now says it's looking for new curves.

Let's make some new curves!

Freshly made from the best ingredients

Take the NIST P-256 prime $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

Freshly made from the best ingredients

Take the NIST P-256 prime $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

Generate random seeds s and hashes $B = H(s)$.
Hash function H:
Keccak with 256-bit output (i.e., `keccakc512`).

Freshly made from the best ingredients

Take the NIST P-256 prime $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

Generate random seeds s and hashes $B = H(s)$.
Hash function H:
Keccak with 256-bit output (i.e., `keccakc512`).

If the elliptic curve $x^3 - 3x + B \bmod p$
does not meet standard security criteria **plus twist-security**,
start over. (This happens tens of thousands of times!)

Freshly made from the best ingredients

Take the NIST P-256 prime $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

Generate random seeds s and hashes $B = H(s)$.
Hash function H:
Keccak with 256-bit output (i.e., `keccakc512`).

If the elliptic curve $x^3 - 3x + B \bmod p$
does not meet standard security criteria **plus twist-security**,
start over. (This happens tens of thousands of times!)

Same with NIST P-224 prime $2^{224} - 2^{96} + 1$.

Freshly made from the best ingredients

Take the NIST P-256 prime $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

Generate random seeds s and hashes $B = H(s)$.
Hash function H:
Keccak with 256-bit output (i.e., `keccakc512`).

If the elliptic curve $x^3 - 3x + B \bmod p$
does not meet standard security criteria **plus twist-security**,
start over. (This happens tens of thousands of times!)

Same with NIST P-224 prime $2^{224} - 2^{96} + 1$.

Also with NIST P-384 prime $2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$.
`keccakc512` is too small here so we switched to `keccakc768`.

Random seeds for your verification pleasure

224: `3CC520E9434349DF680A8F4BCADDA648`
`D693B2907B216EE55CB4853DB68F9165`

256: `3ADCC48E36F1D926701417F101A75F0`
`00118A739D4686E77278325A825AA3C6`

384: `CA9EBD338A9EE0E6862FD329062ABC06`
`A793575A1C744F0EC24503A525F5D06E`

The B values in $x^3 - 3x + B$

224: `BADA55ECFD9CA54C0738B8A6FB8CF4CC`
`F84E916D83D6DA1B78B622351E11AB4E`

256: `BADA55ECD8BBEAD3ADD6C534F92197DE`
`B47FCEB9BE7E0E702A8D1DD56B5D0B0C`

384: `BADA55EC3BE2AD1F9EEEA5881ECF95BB`
`F3AC392526F01D4CD13E684C63A17CC4`
`D5F271642AD83899113817A61006413D`

The B values in $x^3 - 3x + B$

224: **BADA55EC**FD9CA54C0738B8A6FB8CF4CC
F84E916D83D6DA1B78B622351E11AB4E

256: **BADA55EC**D8BBEAD3ADD6C534F92197DE
B47FCEB9BE7E0E702A8D1DD56B5D0B0C

384: **BADA55EC**3BE2AD1F9EEEA5881ECF95BB
F3AC392526F01D4CD13E684C63A17CC4
D5F271642AD83899113817A61006413D

Brainpool to the rescue

2005 "ECC Brainpool standard curves and curve generation" generates deterministic seeds from π and e.

`brainpoolP256r1`:

```
p: A9FB57DBA1EEA9BC3E660A909D838D72
   6E3BF623D52620282013481D1F6E5377
A: 7D5A0975FC2C3057EEF67530417AFFE7
   FB8055C126DC5C6CE94A4B44F330B5D9
B: 26DC5C6CE94A4B44F330B5D9BBD77CBF
   958416295CF7E1CE6BCCDC18FF8C07B6
```

Brainpool to the rescue (or maybe not)

2005 "ECC Brainpool standard curves and curve generation" generates deterministic seeds from π and e.

`brainpoolP256r1`:

```
p: A9FB57DBA1EEA9BC3E660A909D838D72
   6E3BF623D52620282013481D1F6E5377
A: 7D5A0975FC2C3057EEF67530417AFFE7
   FB8055C126DC5C6CE94A4B44F330B5D9
B: 26DC5C6CE94A4B44F330B5D9BBD77CBF
   958416295CF7E1CE6BCCDC18FF8C07B6
```

Brainpool to the rescue (or maybe not)

2005 “ECC Brainpool standard curves and curve generation”
generates deterministic seeds from π and e.

`brainpoolP256r1`:

```
p: A9FB57DBA1EEA9BC3E660A909D838D72
   6E3BF623D52620282013481D1F6E5377
A: 7D5A0975FC2C3057EEF67530417AFFE7
   FB8055C126DC5C6CE94A4B44F330B5D9
B: 26DC5C6CE94A4B44F330B5D9BBD77CBF
   958416295CF7E1CE6BCCDC18FF8C07B6
```

Screwed up data flow in hash inputs; still uses SHA-1;
not twist-secure.
Let’s make an **NSA-free** replacement with **sensible data flow**.
And let’s stick to the NIST primes.

Nothing up our sleeves

Constants already used: $\sin 1$; $\pi/4 = \arctan 1$; $e = \exp 1$.
Start from $\cos 1$.

Nothing up our sleeves

Constants already used: $\sin 1$; $\pi/4 = \arctan 1$; $e = \exp 1$.
Start from $\cos 1$.

Generate the full 160-bit seed
as 32-bit counter followed by $\cos 1$.

(16-bit counter would have been unsafe:
more than $1/1000$ chance of failing to find secure curve.)

Nothing up our sleeves

Constants already used: $\sin 1$; $\pi/4 = \arctan 1$; $e = \exp 1$.
Start from $\cos 1$.

Generate the full 160-bit seed
as 32-bit counter followed by $\cos 1$.

(16-bit counter would have been unsafe:
more than $1/1000$ chance of failing to find secure curve.)

To avoid the Brainpool problems:

- Don't concatenate SHA-1 outputs.
  Use maximum-security full-length SHA-3-512.
- Generate B seed as complement of A seed.
  Guaranteed to be different.

Sage computer-algebra system computing 128 bits of $\cos 1$:

```
sage -c 'print RealField(128)(cos(1)).str(16)[2:34]'
8a51407da8345c91c2466d976871bd2a
```

We started computations for the NIST P-224 prime
and quickly found a secure twist-secure curve from seed
`000000B8 8A51407DA8345C91C2466D976871BD2A`.

Here are A, B (please verify with your own SHA-3 software):
`7144BA12CE8A0C3BEFA053EDBADA555A`
`42391FC64F052376E041C7D4AF23195E`
`BD8D83625321D452E8A0C3BB0A048A26`
`115704E45DCEB346A9F4BD9741D14D49`,

`5C32EC7FC48CE1802D9B70DBC3FA574E`
`AF015FCE4E99B43EBE3468D6EFB2276B`
`A3669AFF6FFC0F4C6AE4AE2E5D74C3C0`
`AF97DCE17147688DDA89E734B56944A2`

Sage computer-algebra system computing 128 bits of $\cos 1$:

```
sage -c 'print RealField(128)(cos(1)).str(16)[2:34]'
8a51407da8345c91c2466d976871bd2a
```

We started computations for the NIST P-224 prime
and quickly found a secure twist-secure curve from seed
`000000B8 8A51407DA8345C91C2466D976871BD2A`.

Here are A, B (please verify with your own SHA-3 software):

`7144BA12CE8A0C3BEFA053ED`**`BADA55`**`5A`
`42391FC64F052376E041C7D4AF23195E`
`BD8D83625321D452E8A0C3BB0A048A26`
`115704E45DCEB346A9F4BD9741D14D49`,

`5C32EC7FC48CE1802D9B70DBC3FA574E`
`AF015FCE4E99B43EBE3468D6EFB2276B`
`A3669AFF6FFC0F4C6AE4AE2E5D74C3C0`
`AF97DCE17147688DDA89E734B56944A2`

Lessons and credits

“Verifiably random” curves, even with “deterministic” seeds,
do not stop the attacker from generating a curve
with a one-in-a-million weakness.

`safecurves.cr.yp.to/bada55.html`

Computation credits:
Saber cluster at Technische Universiteit Eindhoven;
ISF K10 cluster at University of Haifa.

Ongoing work requested by IRTF CFRG:
Quantify wiggle room in Microsoft’s “NUMS” curves.
Quantify wiggle room in Curve25519’s “as fast as possible”.
Preliminary work by Hamburg suggests that
“as fast as possible” minimizes wiggle room.

What if the users want something stronger?

Beyond Curve25519

"E-521" mod $2^{521} - 1$: $x^2 + y^2 = 1 - 376014x^2y^2$.
Found by Bernstein–Lange, independently Hamburg, independently Aranha–Barreto–Pereira–Ricardini.

Beyond Curve25519

"E-521" mod $2^{521}-1$: $x^2+y^2=1-376014x^2y^2$.
Found by Bernstein–Lange, independently Hamburg, independently Aranha–Barreto–Pereira–Ricardini.

But do we really need something so big?

Beyond Curve25519

"E-521" mod $2^{521} - 1$: $x^2 + y^2 = 1 - 376014x^2y^2$.
Found by Bernstein–Lange, independently Hamburg, independently Aranha–Barreto–Pereira–Ricardini.

But do we really need something so big?

One way to choose security levels:
Some users ask for "matching security levels" against AES-256.

e.g. NUMS coauthor Ben Black from Microsoft:
"The goal is matching security levels of the suite components as designed."

Matching security of AES-256 as designed

How secure is AES-256?

2^{256} computations: Brute-force Alice's AES-256 key.
Are there any high-probability AES-256 breaks
using significantly fewer than 2^{256} operations?

Matching security of AES-256 as designed

How secure is AES-256?

2^{256} computations: Brute-force Alice's AES-256 key.
Are there any high-probability AES-256 breaks
using significantly fewer than 2^{256} operations?

Yes! 2^{206} computations:
Collect encryptions of counter 0 under 2^{50} user keys;
compare to encryptions of 0 under 2^{206} guessed keys.

Matching security of AES-256 as designed

How secure is AES-256?

2^{256} computations: Brute-force Alice's AES-256 key.
Are there any high-probability AES-256 breaks using significantly fewer than 2^{256} operations?

Yes! 2^{206} computations:
Collect encryptions of counter 0 under 2^{50} user keys; compare to encryptions of 0 under 2^{206} guessed keys.

How do we match this with ECC?

Matching security of AES-256 as designed

How secure is AES-256?

2^{256} computations: Brute-force Alice's AES-256 key.
Are there any high-probability AES-256 breaks
using significantly fewer than 2^{256} operations?

Yes! 2^{206} computations:
Collect encryptions of counter 0 under 2^{50} user keys;
compare to encryptions of 0 under 2^{206} guessed keys.

How do we match this with ECC?

Curve41417 mod $2^{414} - 17$: $x^2 + y^2 = 1 + 3617x^2y^2$.
CHES 2014: Under 2 million cycles on ARM Cortex-A8,
faster than OpenSSL's fastest ECC option (`secp160r1`).
This is the curve Silent Circle is using in Blackphone.

What if the users want something faster?

What if the users want something faster?

What if the users want something faster?

PRESERVE deliverable 1.1, “Security Requirements of VSA”:

> *The different driving scenarios we looked into indicate that in most driving situations (SUL, MUL, and SHL) the packet rates do not exceed 750 packets per second. Only the maximum highway scenario (MHL) goes well beyond this value (2,265 packets per second). . . .*
>
> *Processing 1,000 packets per second and processing each in 1 ms can hardly be met by current hardware. As discussed in [32], a Pentium D 3.4 GHz processor needs about 5 times as long for a verification (which is the most time-consuming operation in cryptographic processing overhead) and a typical OBU even 26 times as long. This is a good indication that a dedicated cryptographic co-processor is likely to be necessary.*

Constant-time ≈2^{128}-security DH on Intel Sandy Bridge

cycles	ladder	open	g	field	source
194036	yes	yes	1	$2^{255} - 19$	CHES 2011
153000?	yes	no	1	$2^{252} - 2^{232} - 1$	eprint 2012
137000?	no	no	1	$(2^{127} - 5997)^2$	Asiacrypt 2012
122716	yes	yes	2	$2^{127} - 1$	Eurocrypt 2013
119904	no	yes	1	2^{254}	CHES 2013
96000?	no	no	1	$(2^{127} - 5997)^2$	CT-RSA 2014
92000?	no	no	1	$(2^{127} - 5997)^2$	eprint 2014
88916	yes	yes	2	$2^{127} - 1$	**Asiacrypt 2014**

CHES 2011: Bernstein–Duif–Lange–Schwabe–Yang. eprint 2012: Hamburg. CHES 2012: Bernstein–Schwabe. Asiacrypt 2012: Longa–Sica. Eurocrypt 2013: Bos–Costello–Hisil–Lauter. CHES 2013: Oliveira–López–Aranha–Rodríguez-Henríquez. CT-RSA 2014, eprint 2014: Faz-Hernández–Longa–Sánchez. Eurocrypt 2014: Costello–Hisil–Smith. **Asiacrypt 2014: Bernstein–Chuengsatiansup–Lange–Schwabe.**

Constant-time $\approx 2^{128}$-security DH on more CPUs

arch	cycles	ladder	open	g	field	source
A8-slow	497389	yes	yes	1	$2^{255}-19$	CHES 2012
A8-slow	305395	yes	yes	2	$2^{127}-1$	**Asiacrypt 2014**
A8-fast	460200	yes	yes	1	$2^{255}-19$	CHES 2012
A8-fast	273349	yes	yes	2	$2^{127}-1$	**Asiacrypt 2014**
Ivy	182708	yes	yes	1	$2^{255}-19$	CHES 2011
Ivy	145000?	yes	yes	1	$(2^{127}-1)^2$	Eurocrypt 2014
Ivy	119032	yes	yes	2	$2^{127}-1$	Eurocrypt 2013
Ivy	114036	no	yes	1	2^{254}	CHES 2013
Ivy	92000?	no	no	1	$(2^{127}-5997)^2$	CT-RSA 2014
Ivy	89000?	no	no	1	$(2^{127}-5997)^2$	eprint 2014
Ivy	88448	yes	yes	2	$2^{127}-1$	**Asiacrypt 2014**
Haswell	145907	yes	yes	1	$2^{255}-19$	CHES 2011
Haswell	100895	yes	yes	2	$2^{127}-1$	Eurocrypt 2013
Haswell	55595	no	yes	1	2^{254}	CHES 2013
Haswell	54389	yes	yes	2	$2^{127}-1$	**Asiacrypt 2014**