Eliptic-Curve Cryptography and the Internet

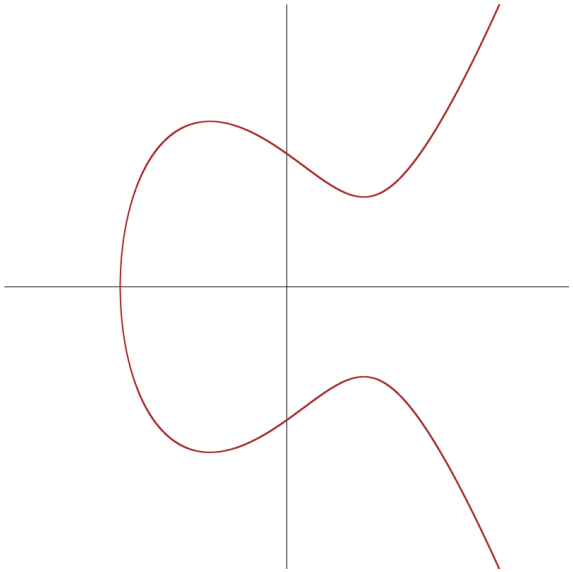
Dominik Joe Pantůček
joe@joe.cz

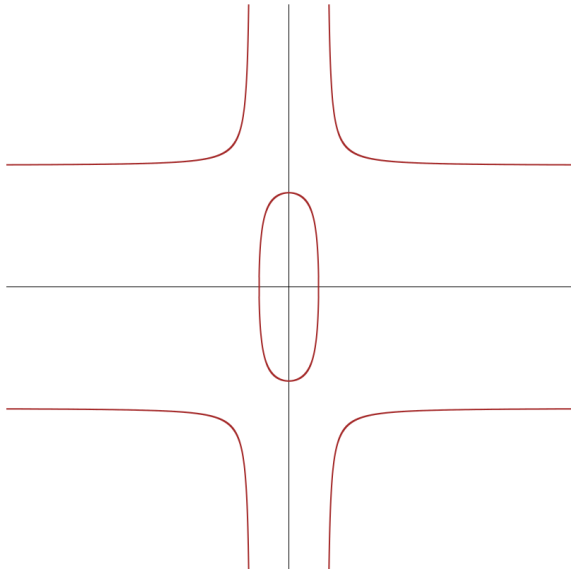
Trustica s.r.o.

7.12.2015

What is EC?

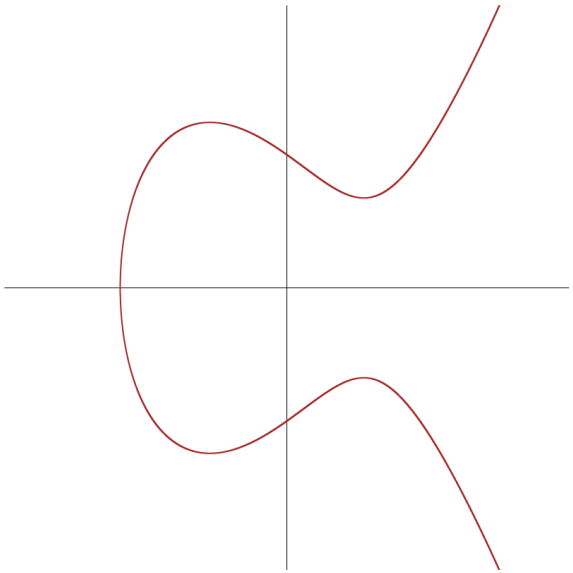
- Eliptic curve
- Looks nice





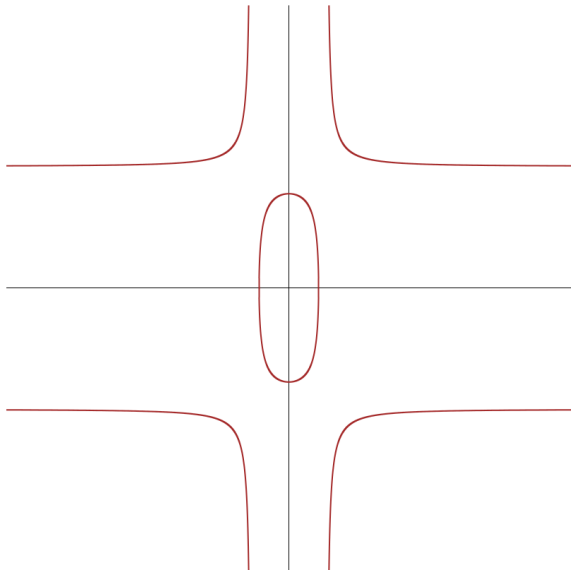
- 1D abelian variety
- Forms abelian group over 2D finite field

What is EC?



$$y^2 = x^3 - 2x + 2$$

EC in Weierstrass form.



$$10x^2 + y^2 = 1 + 6x^2y^2$$

Twisted Edwards curve.

What is ECC?

- Eliptic-curve cryptography
- Public-key cryptography system
- Form abelian group over 2D finite field
  - Point addition: $C = A + B$
  - Point duplication: $B = A + A$
  - Point multiplication: $B = n \cdot A$
- Discrete logarithm problem on the EC group:
  - Given points A and B, find n such that $B = n \cdot A$.
- Good luck ...

Why should we care?

- Security parameter:
  - The bigger the better ...
  - 80bit security is roughly 1024bit RSA
  - The number of required brute-force attempts is approximately 2^n
  - 2^{80} for 1024bit RSA
- Size:
  - The smaller the better ...
  - 1024bit RSA keys and values are 1024 bits long, i.e. 128 bytes
  - Speed and/or latency impact during key exchange

Why should we care?

Elliptic-curve cryptography:

- security parameter is $\frac{n}{2}$
- n is typically 256 – 256bits per coordinate (2D)
- only one coordinate is needed
- $+1$ bit of the other one for checksum calculation
- $\frac{257}{8} = 33$ bytes for 128-bit security

What ECC is available?

- Curves:
  - NIST – National Institute of Standards and Technology
  - SECG – Standards for Efficient Cryptography Group
  - Independent (Bernstein et al.)
- Authentication: ECDSA
- Encryption: ECDH/ECDHE
  - Parameters: p, a, b, G, n
  - Key pairs: d_A, Q_A and d_B, Q_B with $Q_A = d_A G$ and $Q_B = d_B G$
  - Commutativity: $d_A d_B G = d_B d_A G$
  - Shared secret: $d_A Q_B = d_B Q_A$

Standard curves and algorithms

```
$ openssl ecparam -list_curves

  secp112r1 : SECG/WTLS curve over a 112 bit prime field
  secp112r2 : SECG curve over a 112 bit prime field
  secp128r1 : SECG curve over a 128 bit prime field
  secp128r2 : SECG curve over a 128 bit prime field
  secp160k1 : SECG curve over a 160 bit prime field
  secp160r1 : SECG curve over a 160 bit prime field
  secp160r2 : SECG/WTLS curve over a 160 bit prime field
  secp192k1 : SECG curve over a 192 bit prime field
  secp224k1 : SECG curve over a 224 bit prime field
  secp224r1 : NIST/SECG curve over a 224 bit prime field
  secp256k1 : SECG curve over a 256 bit prime field
  secp384r1 : NIST/SECG curve over a 384 bit prime field
  secp521r1 : NIST/SECG curve over a 521 bit prime field
  ...
```

Alternative curves and algorithms

- Curve25519

$$y^2 = x^3 + 486662x^2 + x \pmod{2^{255} - 19}$$

- Ed25519

$$-x^2 + y^2 = 1 - \frac{121665}{121666}x^2y^2 \pmod{2^{255} - 19}$$

- X25519 (DH)
- EdDSA

Library support

- OpenSSL
- Microsoft Cryptography API: Next Generation
- LibreSSL
- GNUTLS
- NaCl

OpenSSL

- Since 0.9.8 (5 Jul 2005)
- ECDSA – `ECDSA_*` `*-ECDSA-*`
- ECDH – `ECDH_*` `ECDH-*`
- Example (configuration string): `ECDH-ECDSA-AES256-GCM-SHA384`

Microsoft Cryptography API: Next Generation

- Windows Vista and newer, Windows Server 2008 and newer
- ECDSA – `BCRYPT_ECDSA_P256_ALGORITHM`
- ECDH – `BCRYPT_ECDH_P256_ALGORITHM`
- Example: `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P384`

LibreSSL

- Forked from OpenSSL by OpenBSD folks in 2014
- API is the same
- New (preferred) `libtls` API uses same configuration strings
- Example (configuration string): `ECDH-ECDSA-AES256-GCM-SHA384`

GNUTLS

- GNU implementation
- Since 3.0.0. (29 Jul 2011)
- Configuration strings – called “priorities”
- Example: `ECDH-ECDSA-AES256-GCM-SHA384`

NaCl

- [Salt]
- Daniel J. Bernstein et al.
- Ed25519 in the next release
- May support NIST-P256/SHA-512-based ECDSA

Application support

- Servers
- Clients

HTTP server support

- Apache
- IIS
- Nginx
- LigHTTPd

HTTP server support

- Supported only in TLS 1.0 and higher.
- Disabling SSL 2.0 and 3.0 is strongly recommended.

Apache

- mod_ssl links against OpenSSL

```
SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256
SSLProtocol All -SSLv2 -SSLv3
```

IIS

- Supported since Windows Server 2008
- Supports ECDHE_RSA
- ECDHE_ECDSA only with ECC certificates
- Cipher preferences (order) can be configured in:
  - `SSL Cipher Suite Order` policy

Nginx

- Linked against OpenSSL

```
ssl_ciphers "ECDHE-ECDSA-AES128-GCM-SHA256"
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
```

LigHTTPd

- Once again ... linked against OpenSSL

```
ssl.cipher-list = "ECDHE-ECDSA-AES128-GCM-SHA256"
ssl.use-sslv2 = "disable"
ssl.use-sslv3 = "disable"
```

HTTP client support

- Firefox
- Chrome
- Microsoft Internet Explorer

Firefox

- Uses NSPR and NSS
- Fully supported at least since 2009
- All builds from last 5 years ...
- No manual configuration needed

Chrome

- Fully supported with the same backend as Firefox
- All builds from last 5 years ...
- No manual configuration needed

Microsoft Internet Explorer

- Fully supported since version 9
- No manual configuration needed

Other

- SSH/OpenSSH
- OpenPGP/GnuPG

SSH/OpenSSH

- ECDSA and ECDH supported since version 5.7 [23 Jan 2011]
- Ed25519 supported since 7 Dec 2013 snapshots
- Official support since version 6.5 [2 Feb 2014]
- It is preferred type of both server and client keys now

```
ssh-keygen -t ed25519 -C My-Ed25519-key -f generated-key
```

OpenPGP/GnuPG

- Suppors standardized ECC
- NIST P-256, P-384, and P-521 curves
- Brainpool P-256, P-384, and P-512 curves
- Both for signing and encryption

OpenPGP/GnuPG

Experimental Ed25519/EdDSA support

- draft-koch-eddsa-for-openpgp-03 [Aug 28, 2015]
  - 22 is probable id
  - expires [Feb 29, 2016]
- draft-josefsson-eddsa-ed25519-03 [May 12, 2015]
  - expired [Nov 13, 2015]
- draft-irtf-cfrg-eddsa-00 [Oct 7, 2015]
  - expires [Apr 9, 2016]

OpenPGP/GnuPG

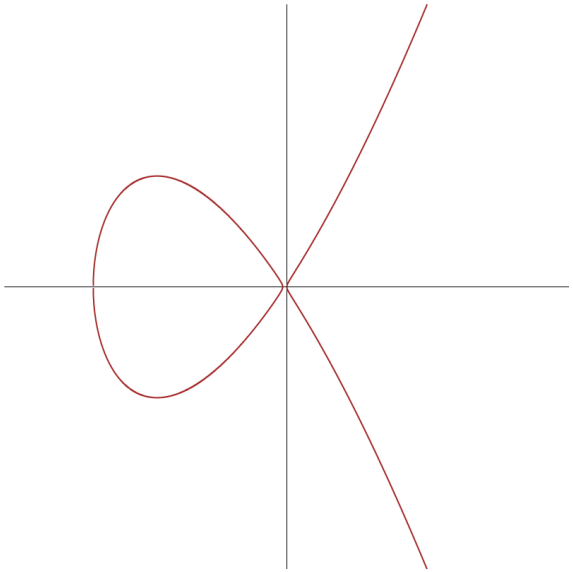
```
$ gpg --expert --full-gen-key

...
Please select what kind of key you want:
...
   (10) ECC (sign only)
...
Your selection? 10

Please select which elliptic curve you want:
    (1) Curve 25519
...
Your selection? 1

gpg: WARNING: Curve25519 is not yet part of the OpenPG
Use this curve anyway? (y/N) y
...
```

Conclusion

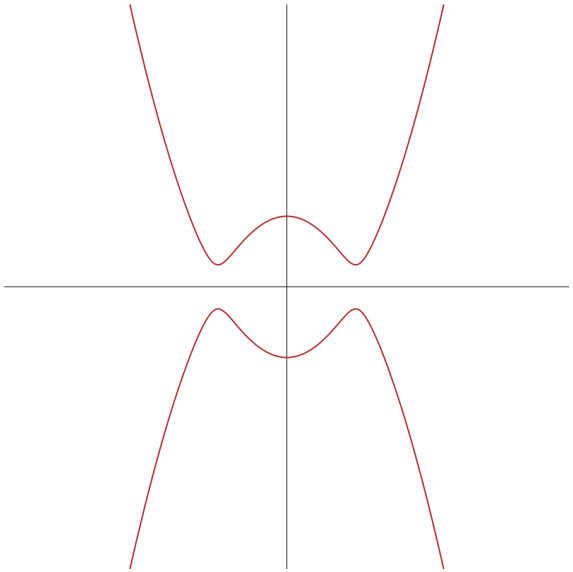
By using ECC wisely you...

- get bigger security parameter,
- get smaller and faster handshakes, and
- make sure the future will be pretty twisted.

Questions



Feel free to ask some ;-)



References

- Elaine Barker, Lily Chen, Allen Roginsky and Miles Smid: NIST Special Publication 800-56A Revision 2 – Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, May 2013
- NIST: FIPS PUB 186-4 – Digital Signature Standard (DSS), July 2013
- IETF work in progress drafts:
  - W. Koch: EdDSA for OpenPGP (draft-koch-eddsa-for-openpgp-03), August 28, 2015
  - S. Josefsson and N. Moeller: EdDSA and Ed25519 (draft-josefsson-eddsa-ed25519-03), May 12, 2015
  - S. Josefsson and I. Liusvaara: Edwards-curve Digital Signature Algorithm (EdDSA) (draft-irtf-cfrg-eddsa-00), October 7, 2015
- Me et al.: all elliptic curve images presented, December 2015

Thank you.