Ladné křivky – prakticky

Dominik Joe Pantůček
joe@joe.cz

Trustica s.r.o.

6.6.2015

Proč?

- dobrá otázka
- velikost klíčů / *úroveň bezpečnosti*
- rychlost operací
- velikost podpisů
- velikost handshake
- mobilní sítě

Certifikáty

- X.509
- ECDSA
- OpenSSL

ECDSA X.509 self-signed

```
$ 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
  ...
```

Úroveň bezpečnosti

- prolomení hrubou silou: 2^{80} operací
- velikosti v bitech:

šifra	privátní	veřejný	podpis
ECDSA	160	320	320
DSA	160	1024	320
RSA	1024	1024	1024

X.509 - Self-Signed

```
$ openssl ecparam -name secp160r1 -genkey \
  -out ec-private.pem
$ openssl req -new -x509 -key ec-private.pem \
  -out ec-certificate.pem -days 365
$ openssl ec -in ec-private.pem \
  -outform der -out ec-private.der
$ openssl x509 -in ec-certificate.pem \
  -outform der -out ec-certificate.der
```

X.509 - Parametry

CN	trustica.cz
O	Trustica s.r.o.
C	CZ
L	Prague

X.509 - Self-Signed

- velikosti v bytech:

šifra	privátní	veřejný	podpis	certifikát (DER)
ECDSA	20	40	40	435
DSA	20	128	40	817
RSA	128	128	128	622

X.509 - podepsán CA

- velikosti v bytech:

šifra	privátní	veřejný	podpis	certifikát (DER)	chain (DER)
ECDSA	20	40	40	342	777
DSA	20	128	40	723	1540
RSA	128	128	128	527	1149

Šifry

```
$ openssl ciphers -v|grep EC.*-EC.*TLS
ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(256) Mac=AEAD
ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(256)  Mac=SHA384
ECDH-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH/ECDSA Au=ECDH Enc=AESGCM(256) Mac=AEAD
ECDH-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH/ECDSA Au=ECDH Enc=AES(256)  Mac=SHA384
ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(128) Mac=AEAD
ECDHE-ECDSA-AES128-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(128)  Mac=SHA256
ECDH-ECDSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH/ECDSA Au=ECDH Enc=AESGCM(128) Mac=AEAD
ECDH-ECDSA-AES128-SHA256 TLSv1.2 Kx=ECDH/ECDSA Au=ECDH Enc=AES(128)  Mac=SHA256
```

Nastavení - server

- Apache HTTP server
- ProFTPd

Nastavení - Apache

```
SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:\
  ECDHE-ECDSA-AES256-GCM-SHA384
```

Nastavení - ProFTPd

```
TLSCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:\
  ECDHE-ECDSA-AES256-GCM-SHA384
```

Knihovny

- OpenSSL
- GnuTLS

Knihovny

- OpenSSL
- GnuTLS
- LibreSSL

Knihovny - OpenSSL

```
int SSL_CTX_set_cipher_list(SSL_CTX *ctx, const char *str);
SSL_CTX_set_cipher_list(ctx,"ECDHE-ECDSA-AES128-GCM-SHA256:"
                            "ECDHE-ECDSA-AES256-GCM-SHA384");
```

Knihovny - GnuTLS

```
int gnutls_priority_set_direct(gnutls_session_t session,
                               const char * priorities,
                               const char ** err_pos);
char *priorities="NONE:+AES-128-GCM:+ECDHE-ECDSA:+SHA256:"
                 "+VERS-TLS1.2:+CURVE-ALL";
char *err_pos;
if (gnutls_priority_set_direct(session,
                               priorities,
                               &err_pos)!=GNUTLS_E_SUCCESS) {
  fprintf(stderr,"Chyba na pozici %d",err_pos-priorities);
}
```

Vlastní protokoly

- Here be the dragons.
- Děkuji za pozornost.
- Crypto++

Crypto++

- Crypto++
- `http://cryptopp.com/`
- Licence
  - Boost Software License 1.0
  - Public Domain
- ECDSA
- ECDH

Crypto++

```
#include <iostream>
#include <cryptopp/eccrypto.h>

using namespace std;
using namespace CryptoPP;
```

Crypto++ - proměnné

```
ECDSA<ECP,SHA256>::PublicKey private_key;
ECDSA<ECP,SHA256>::PublicKey public_key;
AutoSeededRandomPool prng;
```

Crypto++ - ECDSA - generování

```
private_key.Initialize(prng,ASN1::secp256r1());
private_key.MakePublicKey(public_key);
if (!public_key.Validate(prng,3))
  throw runtime_error("ah-oh");
```

Crypto++ - ECDSA - podpis

```
string message("in the bottle");
ECDSA<ECP,SHA256>::Signer signer(private_key);
string signature;
StringSource s(message,true,
  new SignerFilter(prng,
    signer,
    new StringSink(signature)));
```

Crypto++ - ECDSA - verifikace

```
ECDSA<ECP,SHA1>::Verifier verifier(public_key);
if (!verifier.VerifyMessage((const byte*)message.data(),
                            message.size(),
                            (const byte*)signature.data(),
                            signature.size()))
  throw runtime_error("LW'NAFH NOG SHUGG");
```

Crypto++ - ECDH

```
ECDH<ECP>::Domain dh_c(ASN1::secp256r1());
SecByteBlock priv_c(dh_c.PrivateKeyLength());
SecByteBlock pub_c(dh_c.PublicKeyLength());
dh_c.GenerateKeyPair(prng,priv_c,pub_c);
```

Crypto++ - ECDH

pub_c

Crypto++ - ECDH

```
ECDH<ECP>::Domain dh_s(ASN1::secp256r1());
SecByteBlock priv_s(dh_s.PrivateKeyLength());
SecByteBlock pub_s(dh_s.PublicKeyLength());
dh_s.GenerateKeyPair(prng,priv_s,pub_s);
```

Crypto++ - ECDH

pub_s

Crypto++ - ECDH

```
SecByteBlock shared_c(dh_c.AgreedValueLength());
SecByteBlock shared_s(dh_s.AgreedValueLength());
if (!dh_c.Agree(shared_c, priv_c, pub_s))
  throw runtime_error("yaikes");
if (!dh_s.Agree(shared_s, priv_s, pub_c))
  throw runtime_error("yaikes");
```

Crypto++ - ECDH

$$\text{shared\_c} == \text{shared\_s}$$

NSA

```
$ openssl ecparam -list_curves|grep NIST
  secp224r1 : NIST/SECG curve over a 224 bit prime field
  secp384r1 : NIST/SECG curve over a 384 bit prime field
  secp521r1 : NIST/SECG curve over a 521 bit prime field
  prime192v1: NIST/X9.62/SECG curve over a 192 bit prime field
  sect163k1 : NIST/SECG/WTLS curve over a 163 bit binary field
  sect163r2 : NIST/SECG curve over a 163 bit binary field
  sect233k1 : NIST/SECG/WTLS curve over a 233 bit binary field
  sect233r1 : NIST/SECG/WTLS curve over a 233 bit binary field
  sect283k1 : NIST/SECG curve over a 283 bit binary field
  sect283r1 : NIST/SECG curve over a 283 bit binary field
  sect409k1 : NIST/SECG curve over a 409 bit binary field
  sect409r1 : NIST/SECG curve over a 409 bit binary field
  sect571k1 : NIST/SECG curve over a 571 bit binary field
  sect571r1 : NIST/SECG curve over a 571 bit binary field
  wap-wsg-idm-ecid-wtls3: NIST/SECG/WTLS curve over a 163 bit binary field
  wap-wsg-idm-ecid-wtls10: NIST/SECG/WTLS curve over a 233 bit binary field
  wap-wsg-idm-ecid-wtls11: NIST/SECG/WTLS curve over a 233 bit binary field
```

Los Paranoias

- Curve25519
- EdDSA
- Ed25519
- Daniel J. Bernstein
  - Niels Duif,
  - Tanja Lange,
  - Peter Schwabe and
  - Bo-Yin Yang

Ed25519

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

$$\mathrm{FP}\left(2^{255} - 19\right)^2$$

Ed25519

- LibSSH/OpenSSH
- GnuTLS
- NaCL
- signify
- EdDSA and Ed25519
  - draft-josefsson-eddsa-ed25519-03 [May 12, 2015]

Ed25519 - signify

```
signify        -G [-n] [-c comment] -p pubkey -s seckey
signify        -S [-e] [-x sigfile] -s seckey -m message
signify        -V [-eq] [-x sigfile] -p pubkey -m message
```

Ed25519 - crypto_api

```
#include "crypto_api.h"

#define crypto_sign_ed25519_SECRETKEYBYTES 64U
#define crypto_sign_ed25519_PUBLICKEYBYTES 32U
#define crypto_sign_ed25519_BYTES 64U

int         crypto_sign_ed25519(unsigned char *, unsigned long long *,
    const unsigned char *, unsigned long long, const unsigned char *);

int         crypto_sign_ed25519_open(unsigned char *, unsigned long long *,
    const unsigned char *, unsigned long long, const unsigned char *);

int         crypto_sign_ed25519_keypair(unsigned char *, unsigned char *);
```

Otázky?

A to je vše.

Děkuji za pozornost.