E-GROUP
SOFTWARE & BEYOND

Cryptography:
In general

Áron Szabó

ELTE
2020-12-07

Model anatomy

E-GROUP
SOFTWARE & BEYOND

Models

one-way function vs. two-way function

- one-way function:
  one-way function, fixed-length output, avalanche effect

- two-way function:
  symmetric or asymmetric

symmetric vs. asymmetric (vs. hybrid)

- symmetric:
  one private key (e.g. derived from password/passphrase)

- asymmetric:
  one private key, one public key

Model anatomy

Models

- asymmetric:
  one private key, one public key

  IFP - Integer Factorization Problem
  (e.g. RSA)

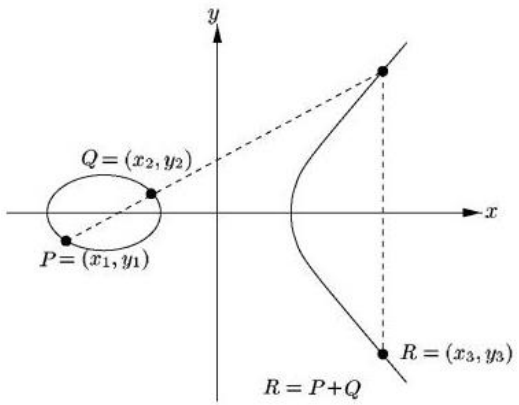
  DLP – Discrete Logarithm Problem
  (e.g. DSA)

  ECDLP – Elliptic Curve Discrete Logarithm Problem
  (e.g. ECDSA)

  difficulty and complexity:
  class NP-intermediate
  (NP-complete is not proven)

```
p = 13
q = 11
n = p * q = ?
n = 143

n = 437
p = ? ... (23)
q = ? ... (19)
```

```
a = 10
b = ?
b = 3^a mod 17 =  8

b = 3^a  mod 17 = 14
a = ? ... (9)
```



Model anatomy

E-GROUP
SOFTWARE & BEYOND

Models

Trusted Third Party vs. Web of Trust

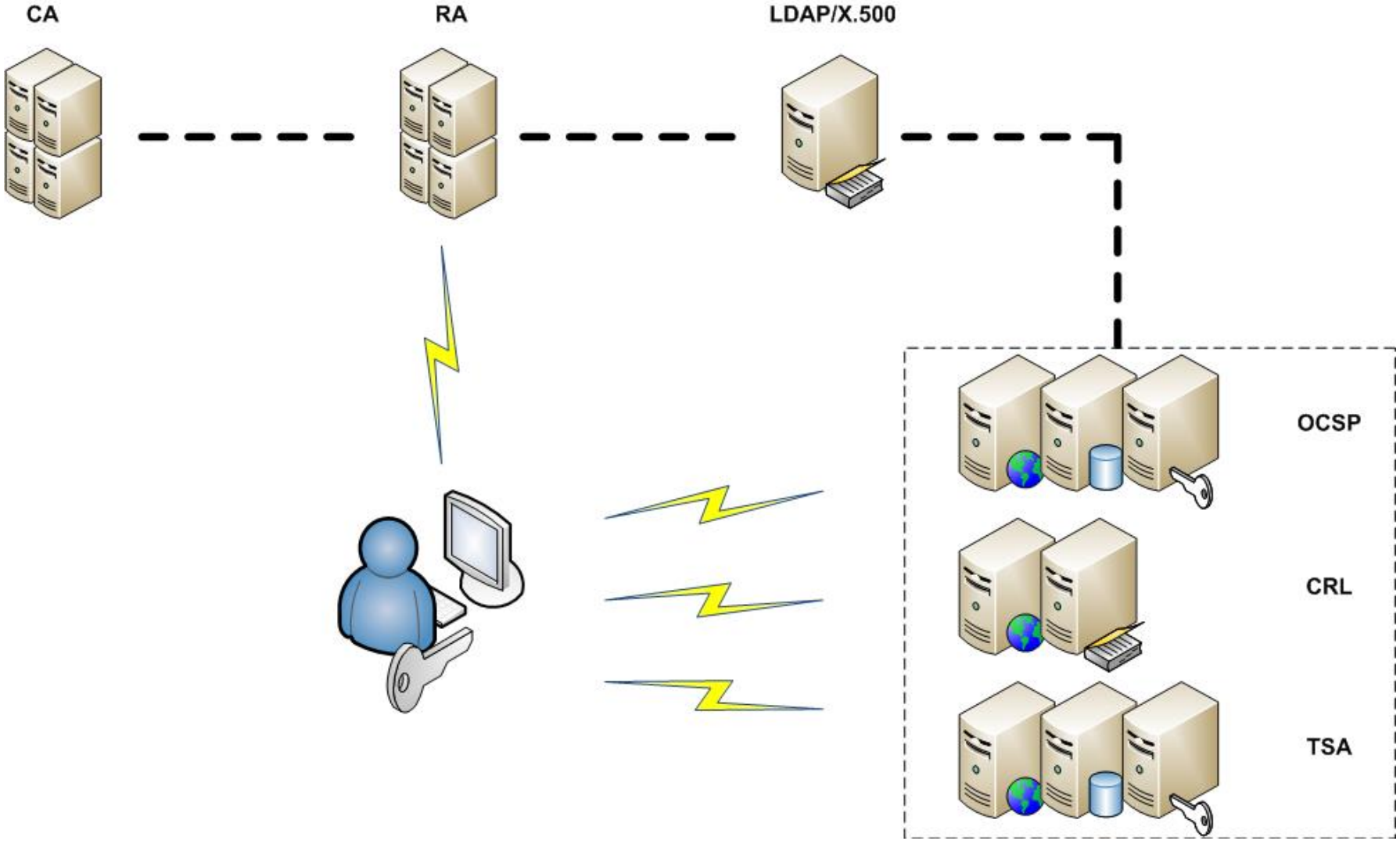
- CA-hierarchy
- PGP, GPG...

X.509 vs. non-X.509

- SSL/TLS, XML signature, S/MIME, CMS, code signing...
- PGP, GPG, DNSSEC...

CA anatomy

Certificate Authority



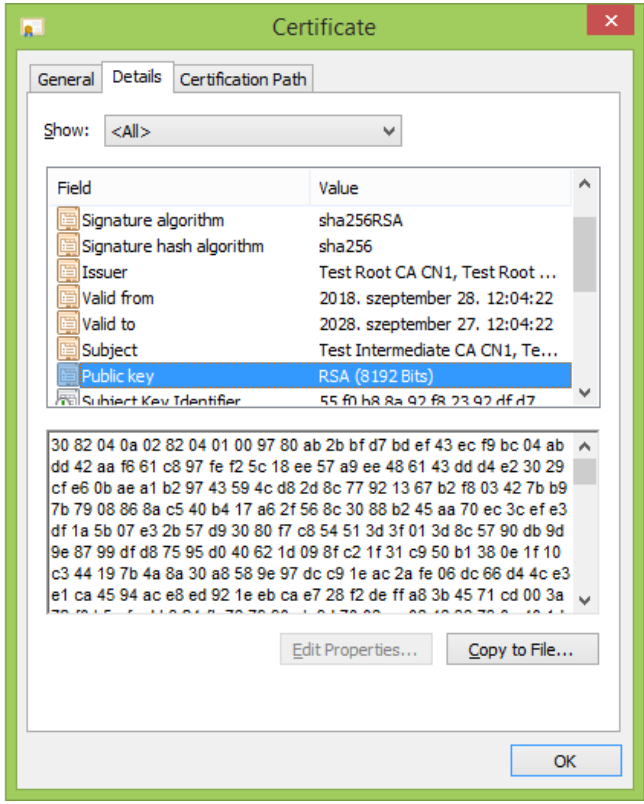
CA anatomy

Certificate Authority

Signed data of Subject
- natural/legal attributes
  and
- public key.

Signed by Issuer
- private key.

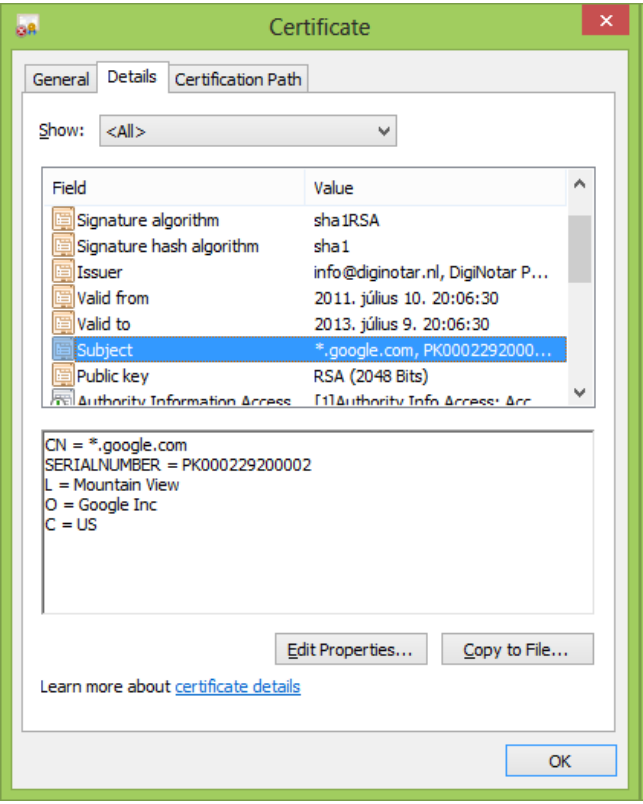
Keystore data can be
- SW (PKCS#12)
  or
- HW (HSM, smart card).

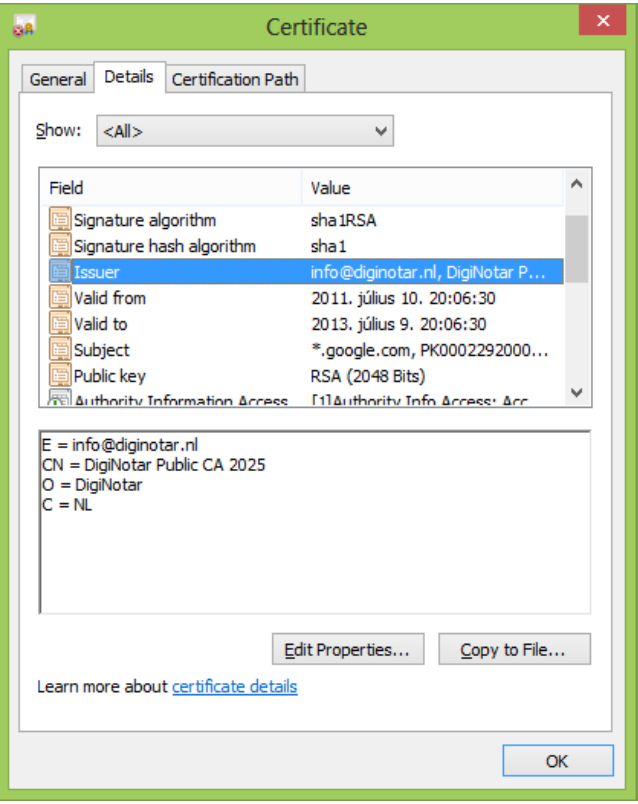


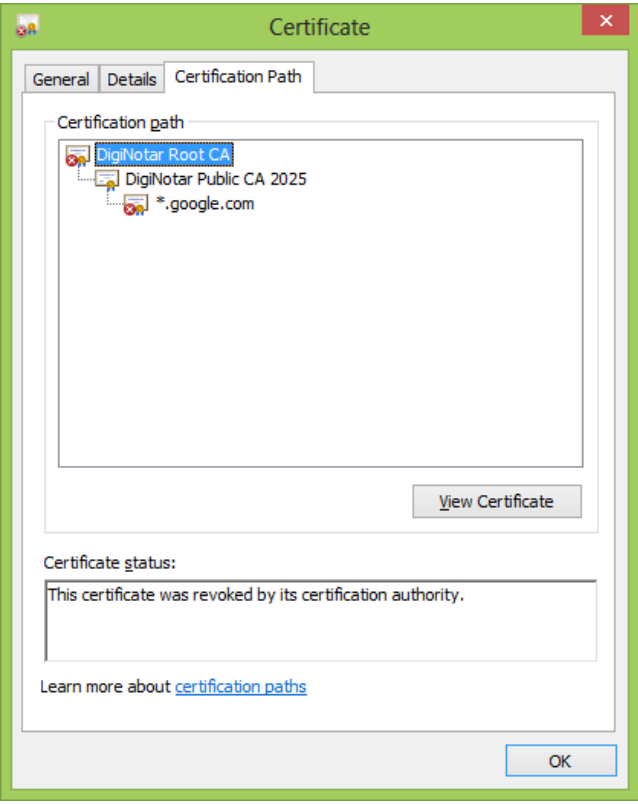
CA anatomy

E-GROUP
SOFTWARE & BEYOND

architecture and DigiNotar







CA anatomy

E-GROUP
SOFTWARE & BEYOND

architecture and DigiNotar

CA server logging

- "CA servers did not log to a separate secure log server. All the investigated log files originated from servers that had been compromised"
- "integrity of blocks of data within the log files can be verified using a signature [...] Two log files failed the verification by the CA software"

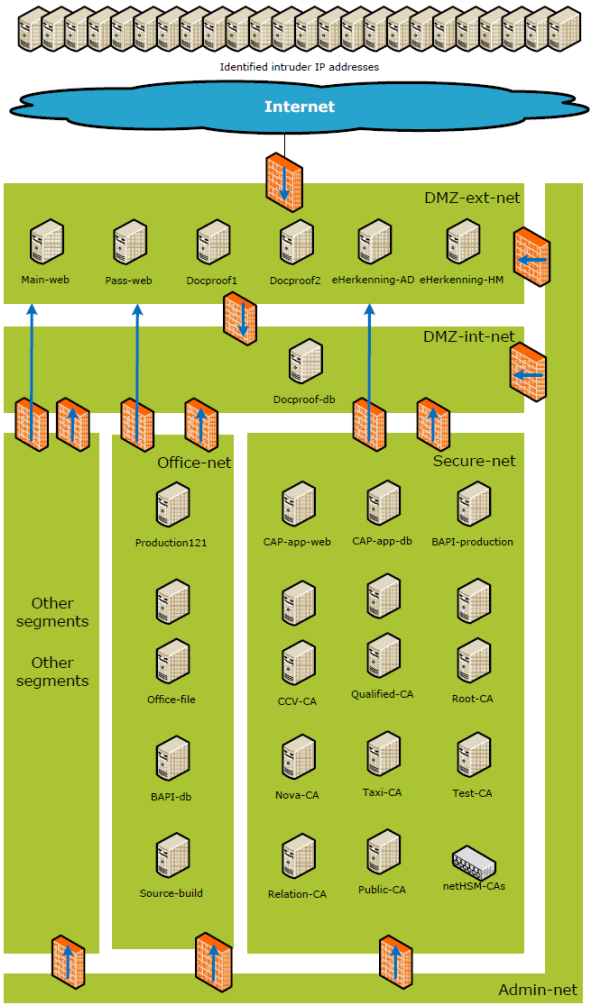
CA software

- "The product used was the RSA Certificate Manager (RSA CM). [...] Older versions of this software are known as RSA Keon."
- "no link could be established between a certificate and an entry in the log files on the basis of a serial number."

CA keystore

- "DigiNotar used nCipher netHSM 500s"
- "CA servers had access to the nCipher netHSM that was also located in Secure-net"
- "on the Qualified-CA server, it appeared that some of the DLLs that were used to access the netHSM had been modified"
- "private keys were activated in the netHSM using smartcards"
- "log entries were found that indicated the automatic generation of a Certificate Revocation List (CRL) [...] which can only occur if a private key was active on the netHSM"

Date	Notes
2011-06-17	"the Main-web and the Docproof2 web server were compromised"
2011-07-01	"first scanning activity occurred in Secure-net"
2011-07-10	"first rogue certificate was successfully created on the Relation-CA server [...]OCSP requests for rogue certificates started arriving [...] from an DSL subscriber in Iran"
2011-07-27	"First OCSP request at DigiNotar for the rogue wildcard Google certificate"
2011-07-28	"attempts were made to verify the rogue login.yahoo.com certificate by IP addresses originating from the Islamic Republic of Iran"
2011-08-04	"massive activity on the OCSP responder for a rogue *.google.com certificate originating from the Islamic Republic of Iran"
2011-08-31	"Google Chrome blacklisted a list of known rogue serial numbers"



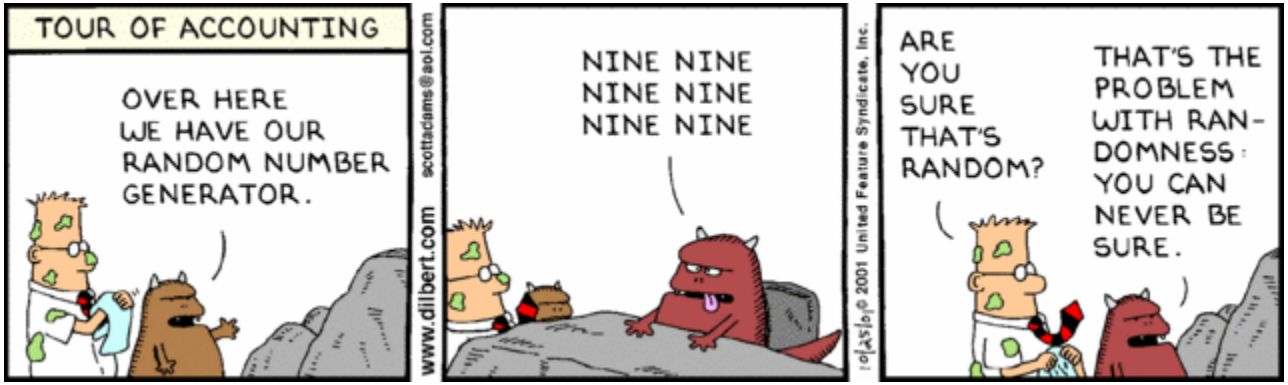
FOX IT

CA anatomy

E-GROUP
SOFTWARE & BEYOND

code review and OpenSSL

- OpenSSL: 2006-09-17 – 2008-05-13 (0.9.8c-1 – 0.9.8g-9)
- ssleay_rand_add() and process ID (PID): N * 32768 key pairs
- affected SSL/TLS sites and code signer keys (trusted CAs of Microsoft Windows)
- who generated the vulnerable key pairs and the PKCS#10?

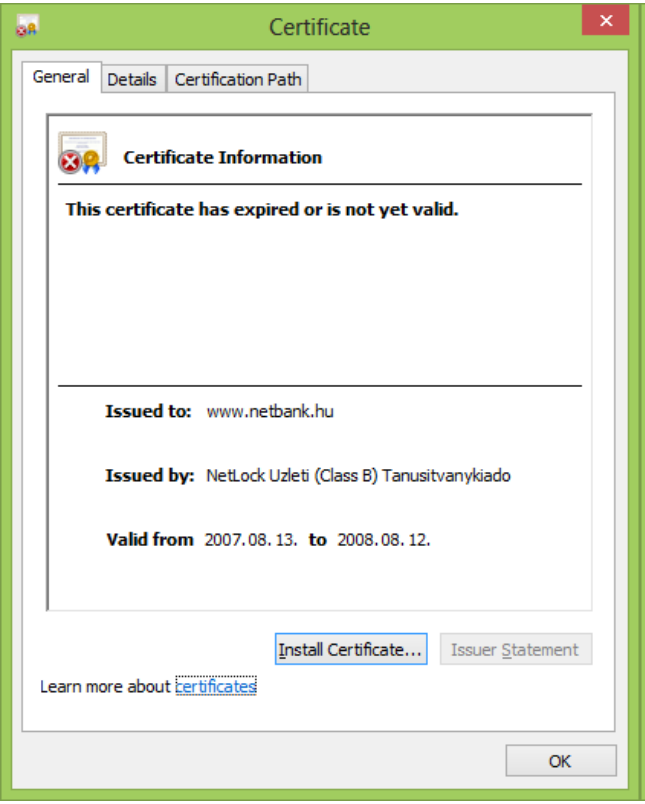


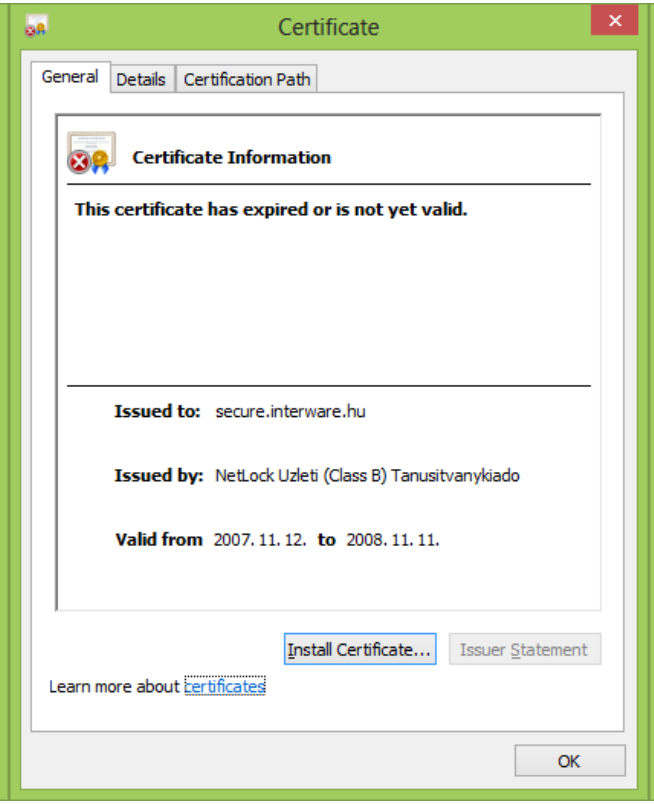
CA anatomy

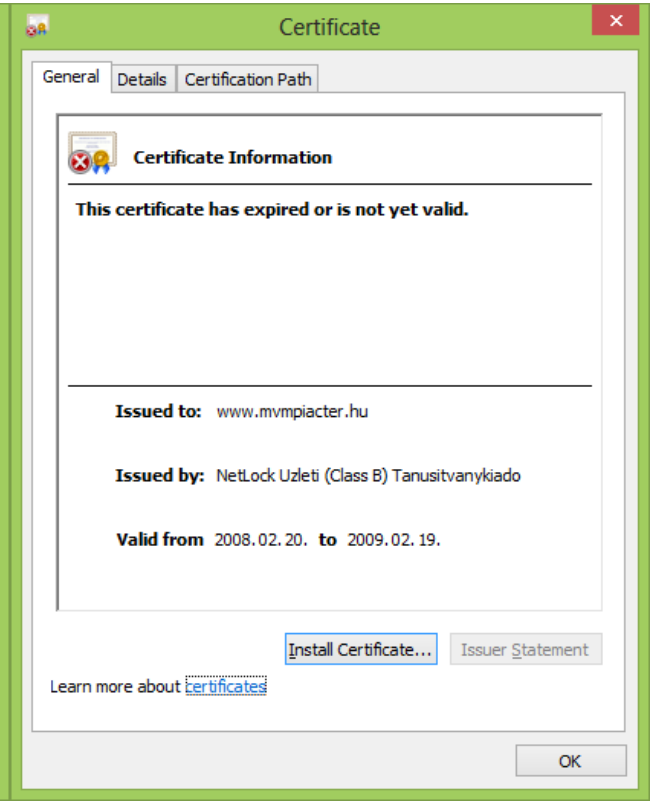
E-GROUP
SOFTWARE & BEYOND

code review and OpenSSL

- affected Hungarian sites (trusted Hungarian CA)
- bank (MagNet), server hosting and ISP (GTS), critical infrastructure (MVM)





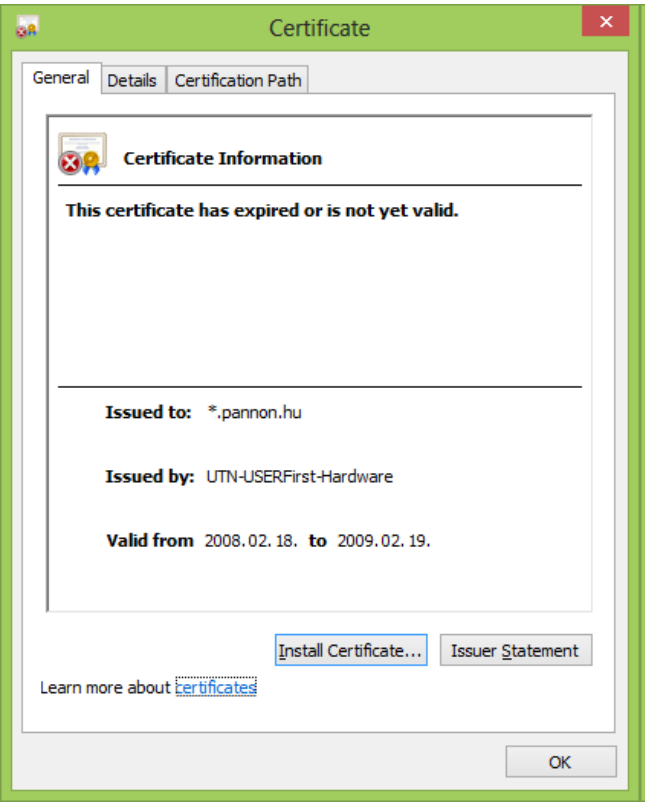


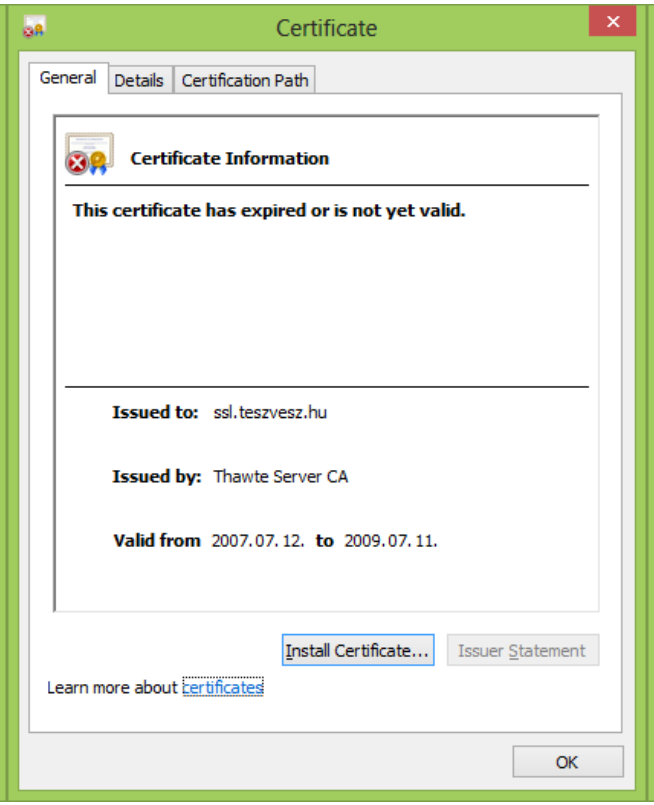
CA anatomy

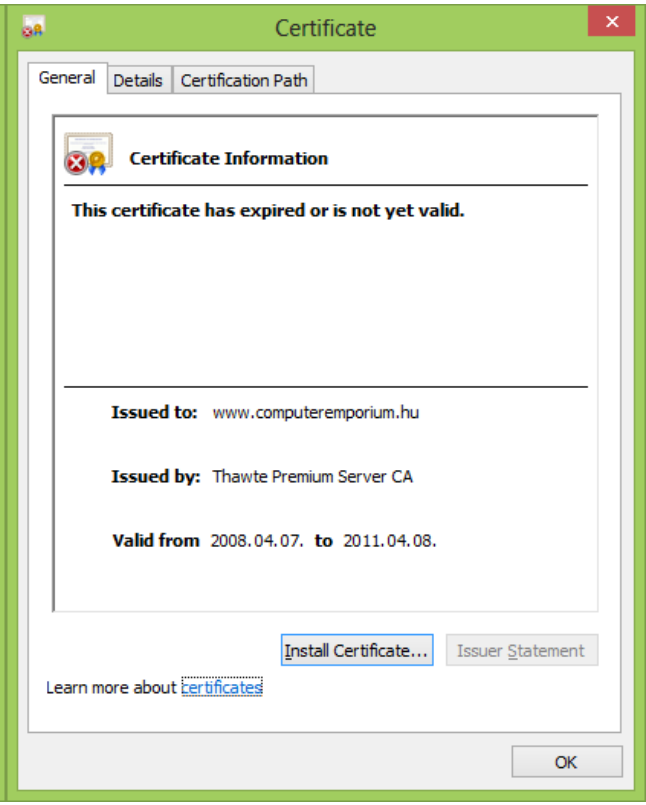
E-GROUP
SOFTWARE & BEYOND

code review and OpenSSL

- affected Hungarian sites (other trusted CAs)
- mobile operator (Telenor), auction (TeszVesz), webshop (ComputerEmporium)





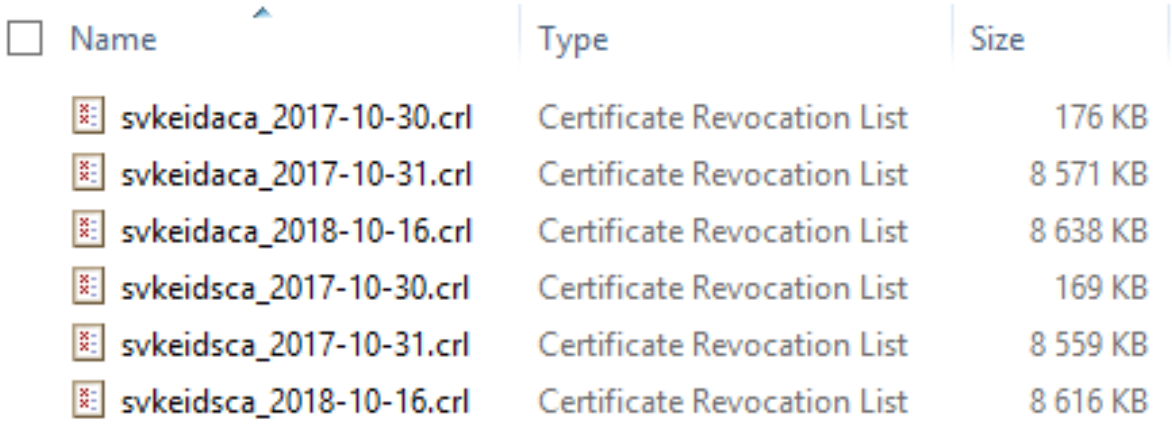


CA anatomy

E-GROUP
SOFTWARE & BEYOND

code review and key generation functions

- Infineon RSA library — 2017-10-16, 2017-10-30
  ROCA (Return of Coppersmith's Attack)
- sectors
  government — Estonia (eID), Slovakia (eID), Spain (eID)
  bank — Poland (Gemalto IDPrime .NET)
  other — (e.g. Yubico YubiKey 4, TPM in HP, Lenovo laptop)
- devices — (e.g. NXP?)
- codes — (e.g. BouncyCastle?)
- key sizes — (e.g. RSA 3072 bits? ECC?)
- operations — (e.g. revoked CA, large CRLs, lack of functionality)



Name	Type	Size
svkeidaca_2017-10-30.crl	Certificate Revocation List	176 KB
svkeidaca_2017-10-31.crl	Certificate Revocation List	8 571 KB
svkeidaca_2018-10-16.crl	Certificate Revocation List	8 638 KB
svkeidsca_2017-10-30.crl	Certificate Revocation List	169 KB
svkeidsca_2017-10-31.crl	Certificate Revocation List	8 559 KB
svkeidsca_2018-10-16.crl	Certificate Revocation List	8 616 KB

CA anatomy

E-GROUP
SOFTWARE & BEYOND

code review and key generation functions

https://asecuritysite.com/encryption/copper

```
p = k * M + (65537^a mod M)
n = 39
M = 962947420735983927056946215901134429196419130606213075415963491270
k = 8
a = 3
p = 770357936588787141645556972720907543357135304484970488481556973931 3

q = k * M + (65537^a mod M)
n = 39
M = 962947420735983927056946215901134429196419130606213075415963491270
k = 8
a = 1
q = 770357936588787141645556972720907543357135304484970460332770799569 7
```

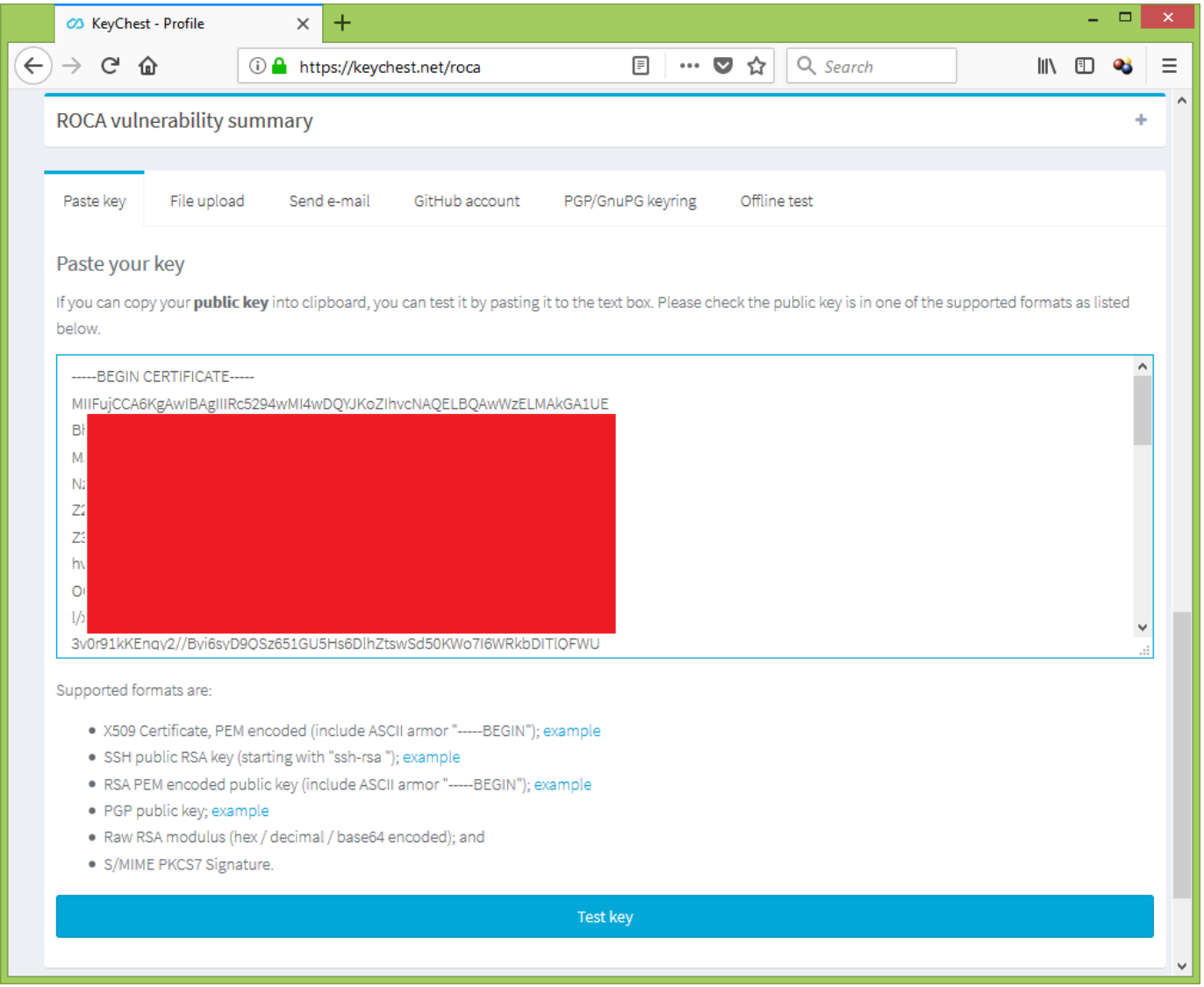
https://www.mobilefish.com/services/rsa_key_generation/rsa_key_generation.php

```
phi(n) = (p - 1) * (q - 1)
phi(n) = 14e6e6e84a6c9bc76fcd3e57153c4eb8069b4a417024e38cc40c7cad483421d8d46903b67e137fe19997b0afcf43d17967663b7757574900

n = p * q (modulus)
n = 14e6e6e84a6c9bc76fcd3e57153c4eb8069b4a417024e38cc40c7cadda80dc43be06fdbec34b7aadee2c3ea0918171be4031f98934d10561
    (decimal: 59345135046533379070949212431851844835352661985729183720911732659737228034511329429220576921531616803463218368738626766508388015736161)

e = (publicExponent)
e = 10001
    (decimal: 65537)

d = e^-1 mod phi (privateExponent)
d = 10764b26687761b991b47484e194bdeaf2491c12ade05fd75e10f7a1025d0ccb597fdc0cbeff5c98b1ff66891114b8c685495f4c7edda001

p = (prime1)
p = 49265d3574cefd04229bfd662a4a46f8611ed0226c665f0a6ebdde31
    (decimal: 770357936588787141645556972720907543357135304484970488481556973931 3)
    (n = 39, k = 8, a = 3)

q = (prime2)
q = 49265d3574cefd04229bfd662a4a46f8611ed0226c655f076ebbde31
    (decimal: 770357936588787141645556972720907543357135304484970460332770799569 7)
    (n = 39, k = 8, a = 1)

dP = d mod (p - 1) (exponent1)
dP = 2646c9dd2ffb23902760028f942f7bc57a647a3a990854510c3393f1

dQ = d mod (q - 1) (exponent2)
dQ = 2646c9dd2ffb23902760028f942f7bc57a647a3a9907ce5b004993f1

qInv = q^-1 mod p (coefficient)
qInv = 23c7b4c478c1d3174522a8362a1536f33b637a1bfc32fef67f732fe0

public_key_vulnerable_generated.txt (base64 ASN.1)
MFQwDQYJKoZIhvcNAQEBBQADQwAwQAI5ABTm5uhKbJvHb80+VxU8TrgGm0pBcCTjjMQMfK3agNxDvgb9vsNLeq3uLD6gkYFxvkAx+Yk00QVhAgMBAAE=
```

CA anatomy

E-GROUP
SOFTWARE & BEYOND

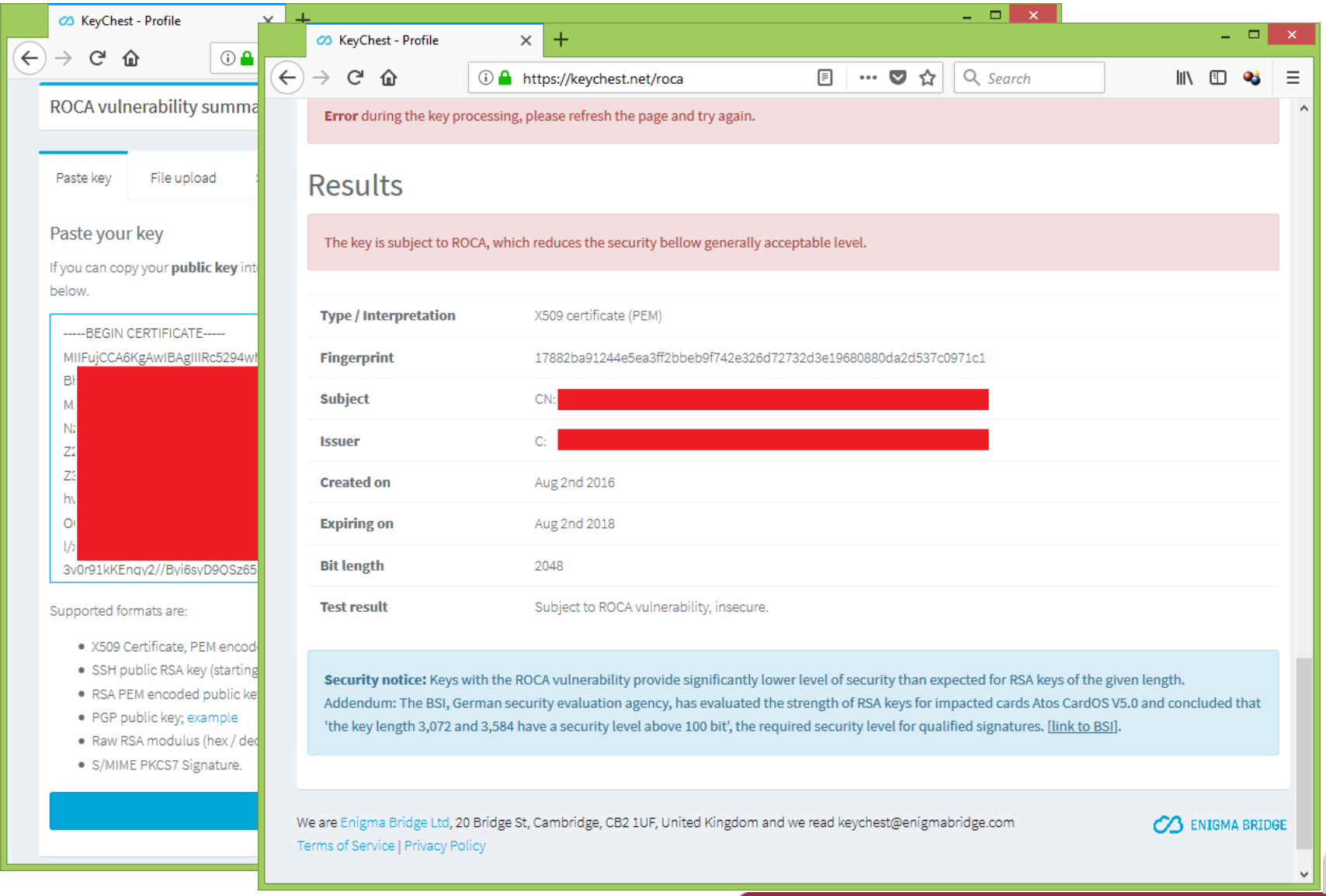
code review and key generation functions



CA anatomy

E-GROUP
SOFTWARE & BEYOND

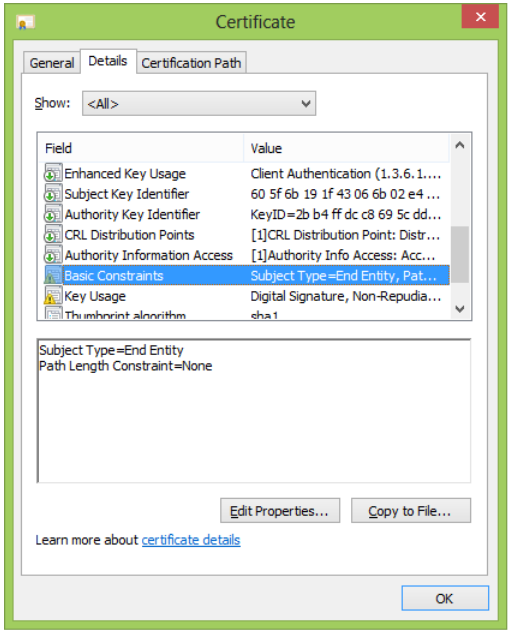
code review and key generation functions



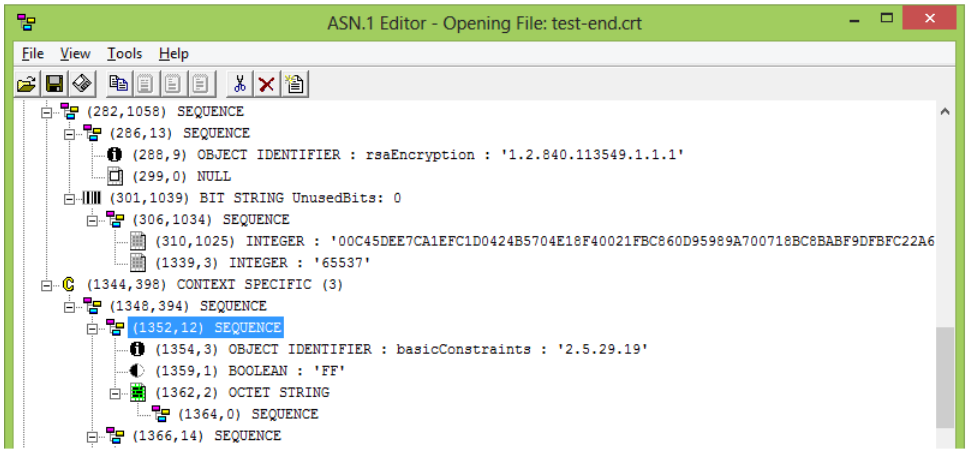
X.509 anatomy

E-GROUP
SOFTWARE & BEYOND

basicConstraints



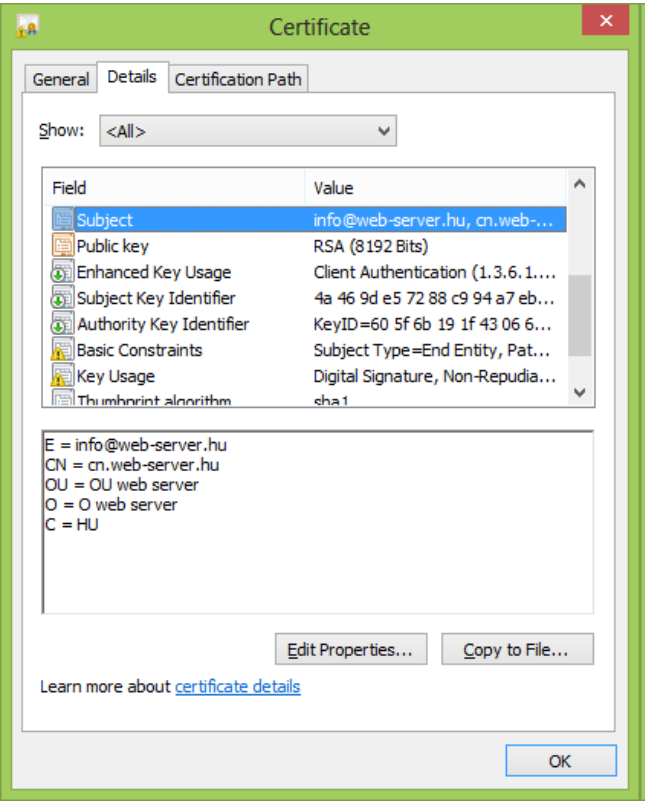

```
basicConstraints

   id-ce-basicConstraints OBJECT IDENTIFIER ::= { id-ce 19 }

   BasicConstraints ::= SEQUENCE {
                                     cA                  BOOLEAN DEFAULT FALSE,
                                     pathLenConstraint INTEGER (0..MAX) OPTIONAL
                                   }
```

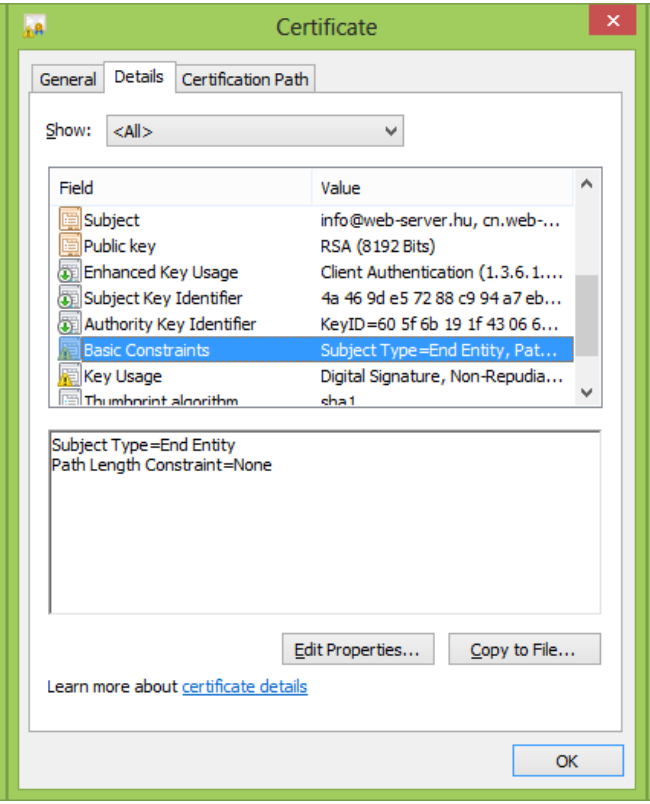


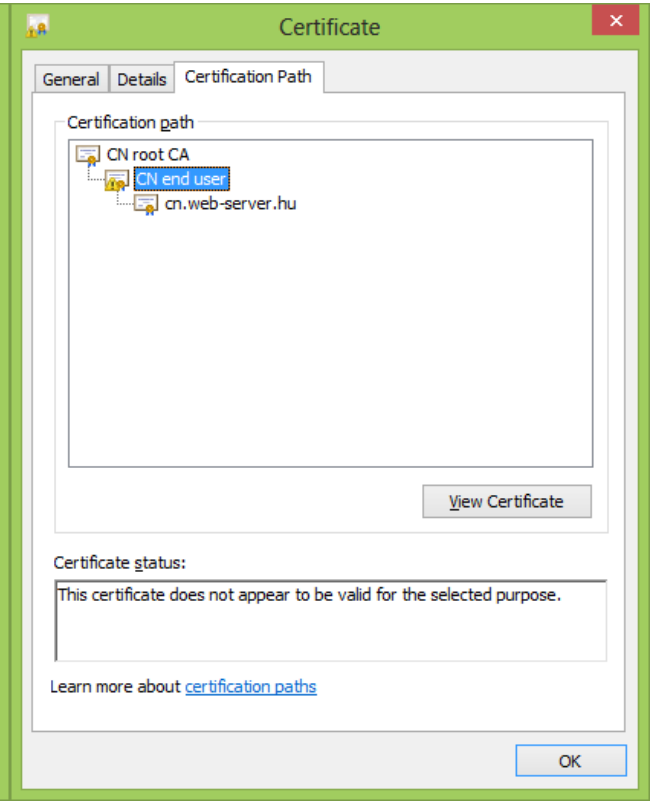
X.509 anatomy

E-GROUP
SOFTWARE & BEYOND

basicConstraints and Moxie Marlinspike



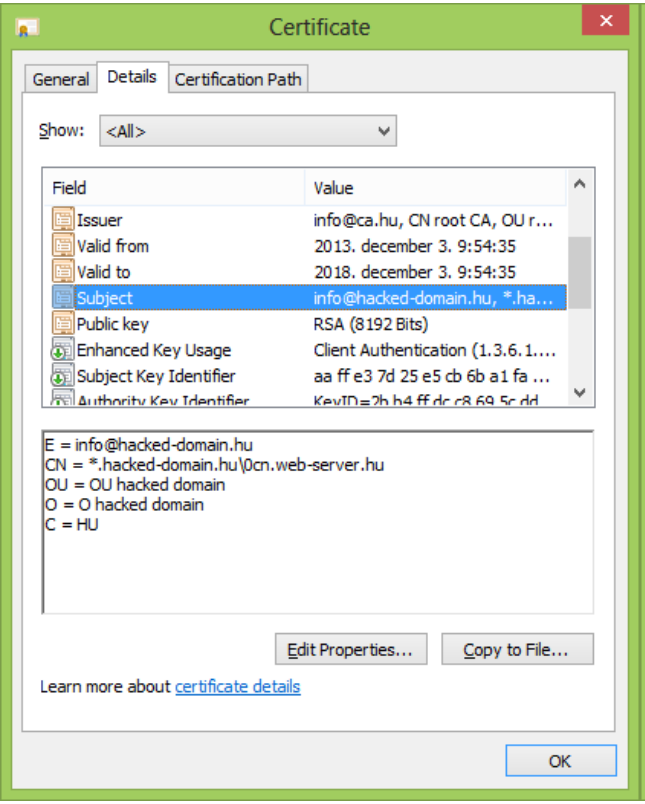


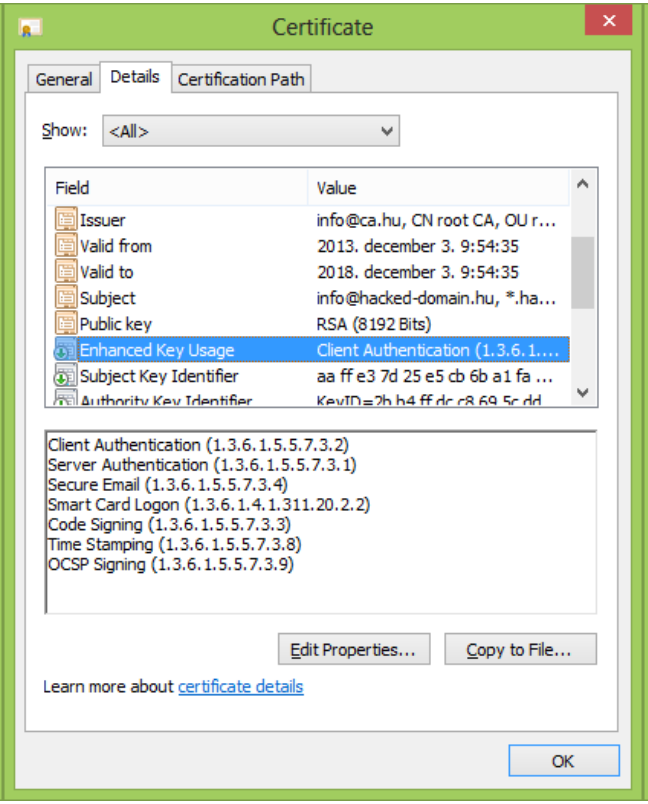


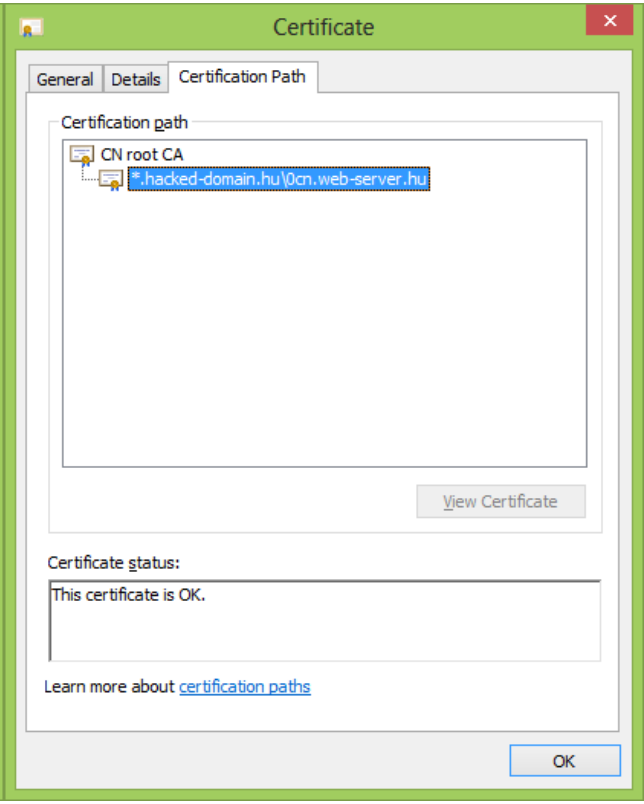
X.509 anatomy

E-GROUP
SOFTWARE & BEYOND

subject „\0” and Moxie Marlinspike







X.509 anatomy

E-GROUP
SOFTWARE & BEYOND

ECDSA
Microsoft

D.1.2 Curves over Prime Fields

For each prime p, a pseudo-random curve

$$E: y^2 \equiv x^3 - 3x + b \pmod{p}$$

of prime order n is listed[4]. (Thus, for these curves, the cofactor is always $h = 1$.) The following parameters are given:

- The prime modulus p
- The order n
- The 160-bit input seed *SEED* to the SHA-1 based algorithm (i.e., the domain parameter seed)
- The output c of the SHA-1 based algorithm
- The coefficient b (satisfying $b^2 c \equiv -27 \pmod{p}$)
- The base point x coordinate G_x
- The base point y coordinate G_y

The integers p and n are given in decimal form; bit strings and field elements are given in hexadecimal.

D.1.2.4 Curve P-384

$p =$ 39402006196394479212279040100143613805079739270465446667948293404245721771496870329047266088258938001861606973112319

$n =$ 39402006196394479212279040100143613805079739270465446667946905279627659399113263569398956308152294913554433653942643

SEED = a335926a a319a27a 1d00896a 6773a482 7acdac73

$c =$ 79d1e655 f868f02f ff48dcde e14151dd b80643c1 406d0ca1
0dfe6fc5 2009540a 495e8042 ea5f744f 6e184667 cc722483

$b =$ b3312fa7 e23ee7e4 988e056b e3f82d19 181d9c6e fe814112
0314088f 5013875a c656398d 8a2ed19d 2a85c8ed d3ec2aef

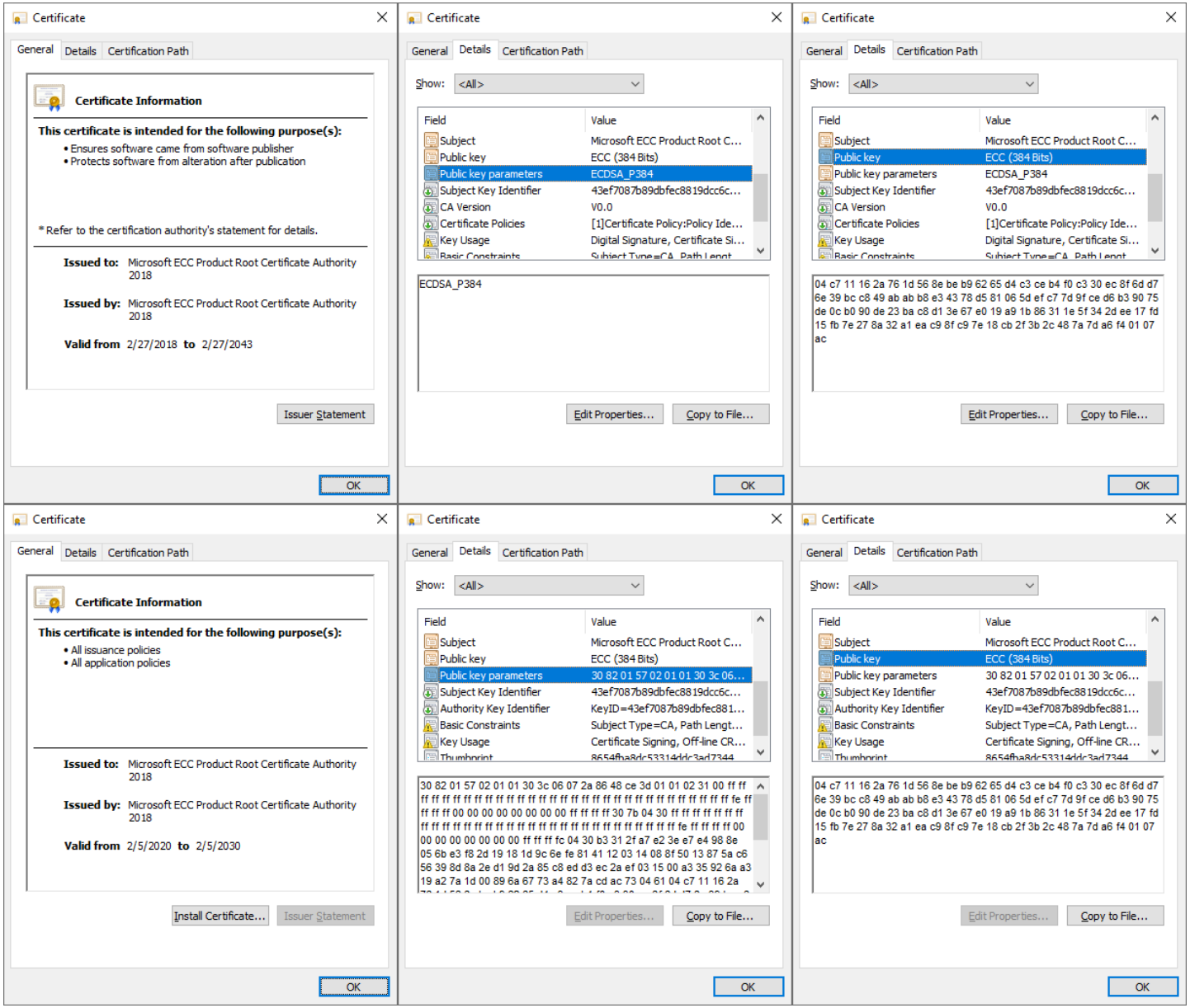
$G_x =$ aa87ca22 be8b0537 8eb1c71e f320ad74 6e1d3b62 8ba79b98
59f741e0 82542a38 5502f25d bf55296c 3a545e38 72760ab7

$G_y =$ 3617de4a 96262c6f 5d9e98bf 9292dc29 f8f41dbd 289a147c
e9da3113 b5f0b8c0 0a60b1ce 1d7e819d 7a431d7c 90ea0e5f

X.509 anatomy

E-GROUP
SOFTWARE & BEYOND

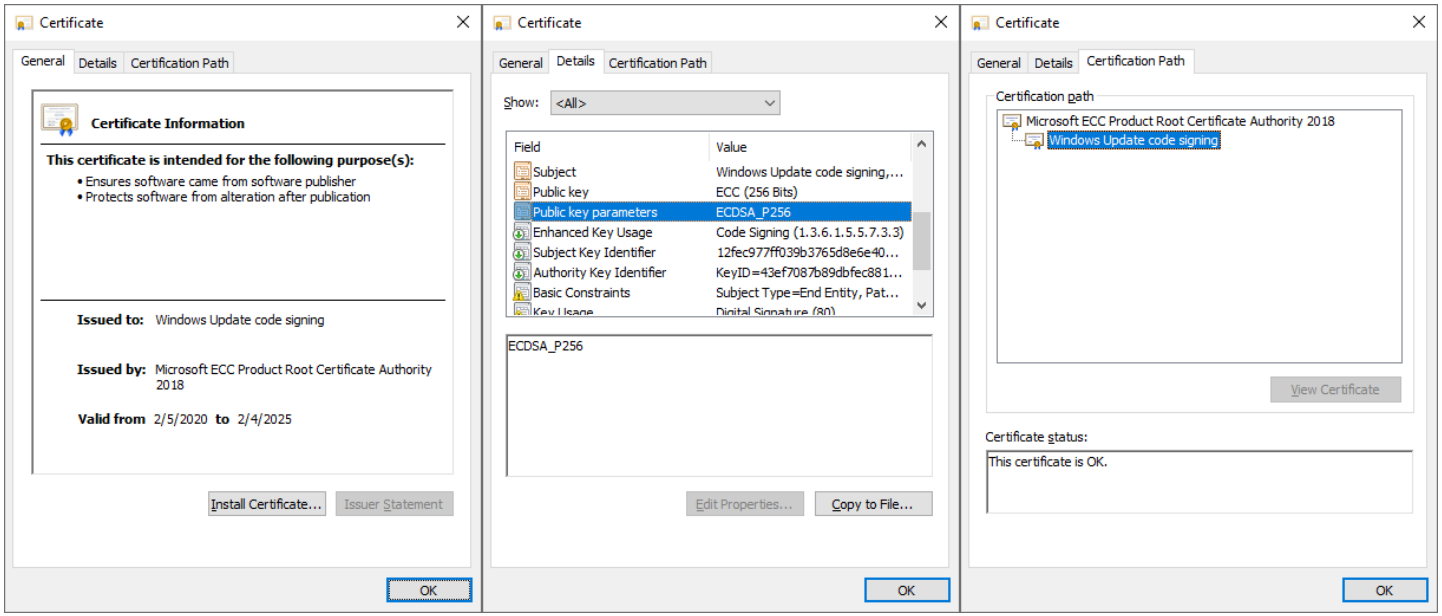
ECDSA
Microsoft



X.509 anatomy

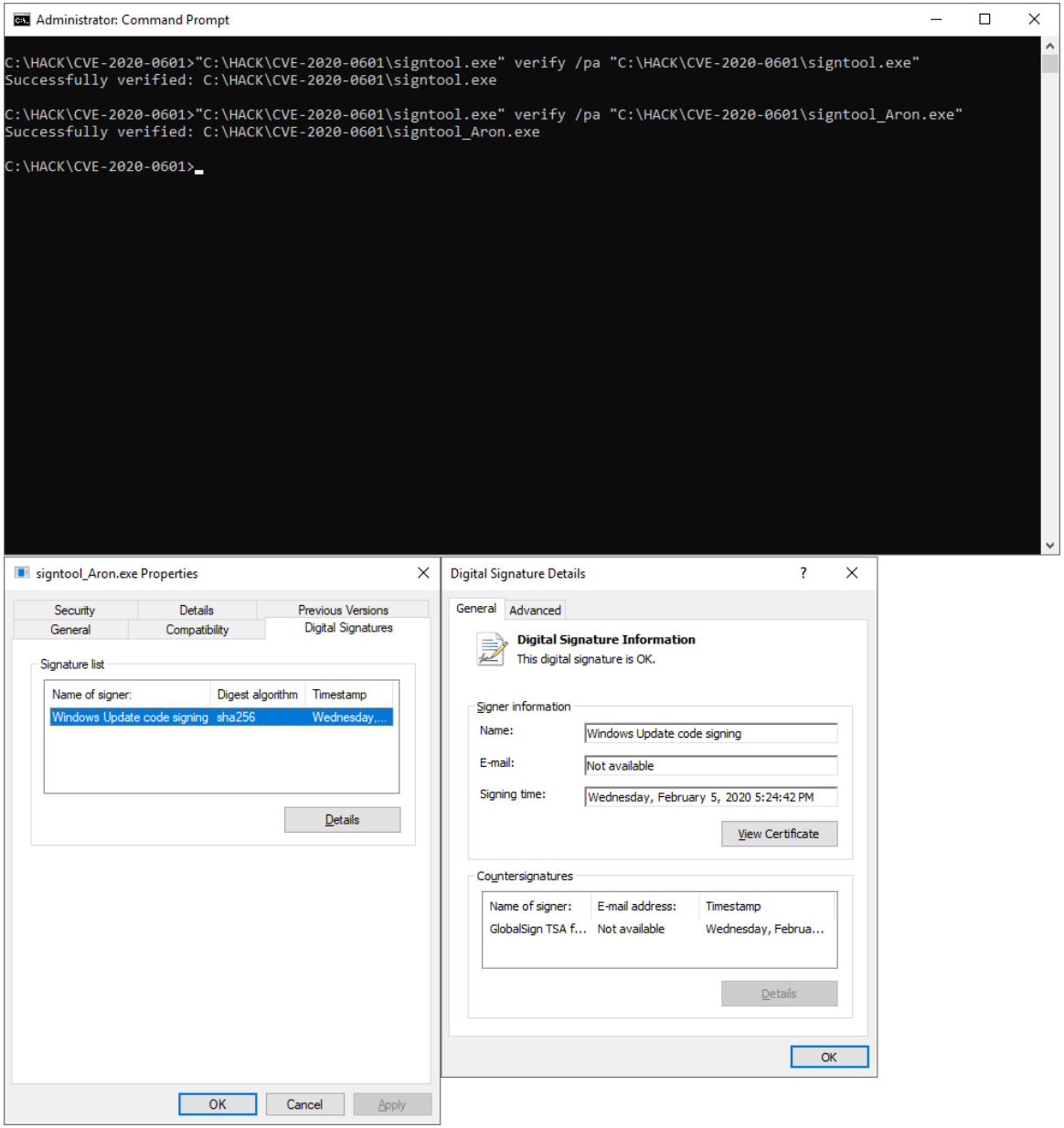
ECDSA
Microsoft

- p prime
- n prime, order
- Q public key, point (from original certificate: Q = d * G)
- d secret key, random, (d = 1, in the range [1, n-1])
- G generator (not checked, base point can be replaced: G = Q)



X.509 anatomy

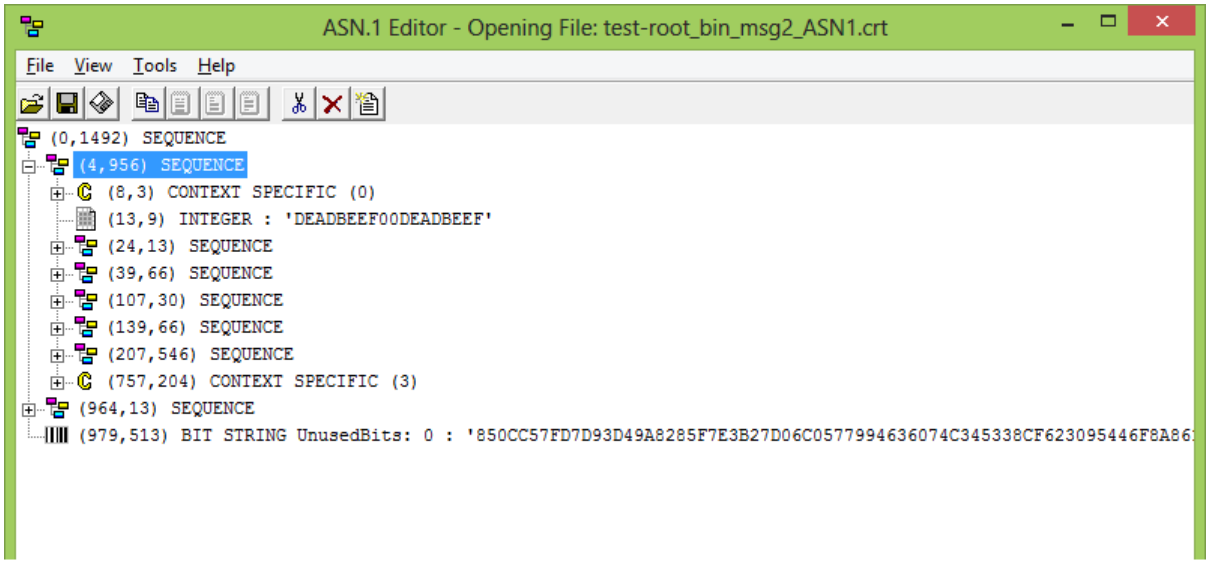
ECDSA
Microsoft



Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

MD5 collision and Xiaoyun Wang, Marc Stevens, Vlastimil Klima



Certificate and TBSCertificate

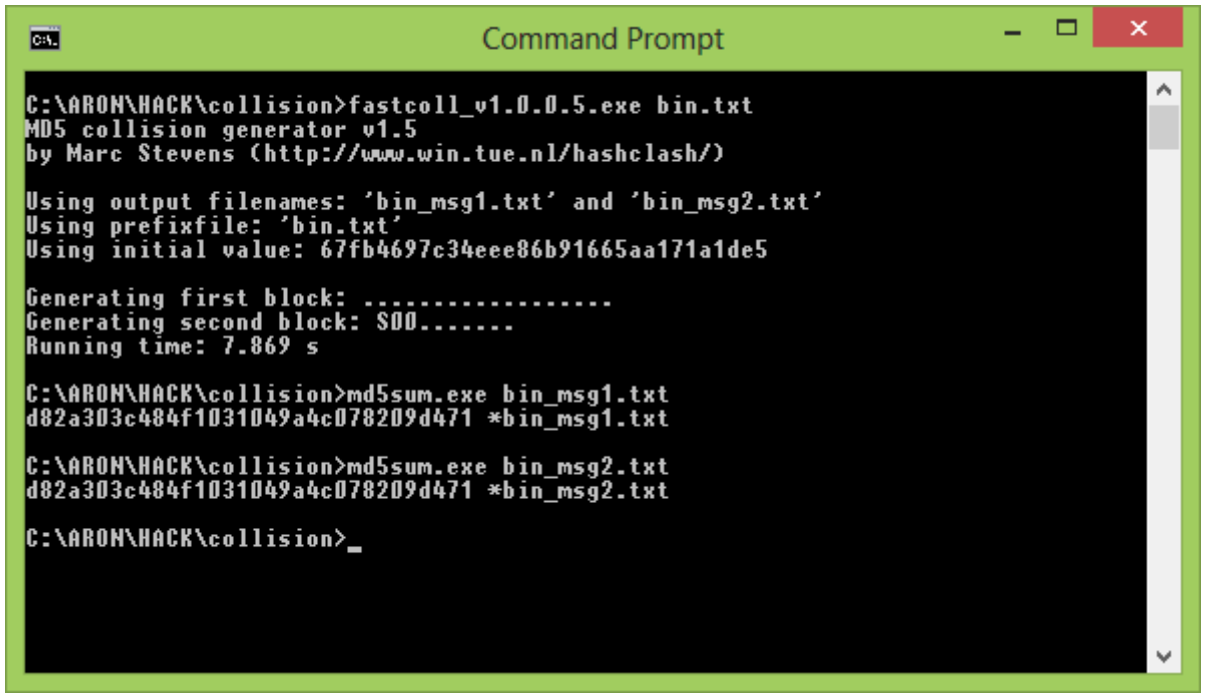
```
Certificate         ::= SEQUENCE {
   tbsCertificate                    TBSCertificate,
   signatureAlgorithm                AlgorithmIdentifier,
   signatureValue                    BIT STRING
                                   }

TBSCertificate      ::= SEQUENCE {
   version                             [0] EXPLICIT Version DEFAULT v1,
   serialNumber                        CertificateSerialNumber,
   signature                           AlgorithmIdentifier,
   issuer                              Name,
   validity                            Validity,
   subject                             Name,
   subjectPublicKeyInfo                SubjectPublicKeyInfo,
   issuerUniqueID                    [1] IMPLICIT UniqueIdentifier OPTIONAL,
   subjectUniqueID                   [2] IMPLICIT UniqueIdentifier OPTIONAL,
   extensions                        [3] EXPLICIT Extensions OPTIONAL
                                   }

AlgorithmIdentifier ::= SEQUENCE {
   algorithm                           OBJECT IDENTIFIER,
   parameters                          ANY DEFINED BY algorithm OPTIONAL
                                   }
```

Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

MD5 collision and Xiaoyun Wang, Marc Stevens, Vlastimil Klima

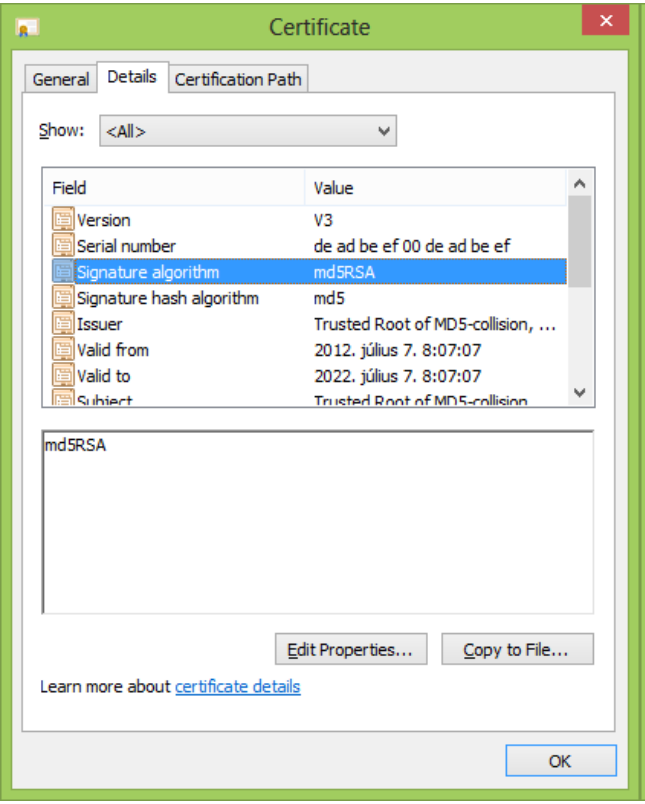
- add collision bits to tbsCertificate (bin.txt → bin_msg1.txt + bin_msg2.txt)
- add/copy SEQUENCE, signatureAlgorithm, signatureValue to tbsCertificate

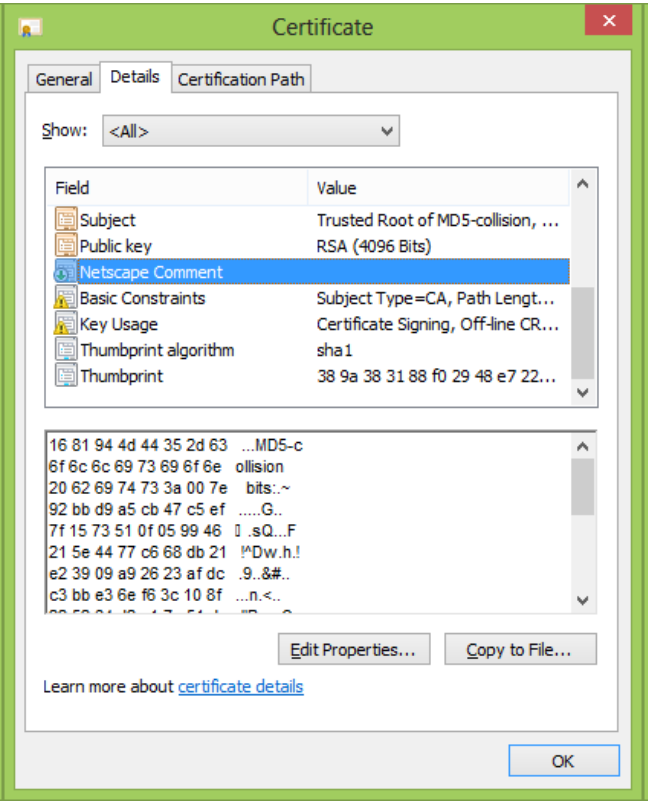


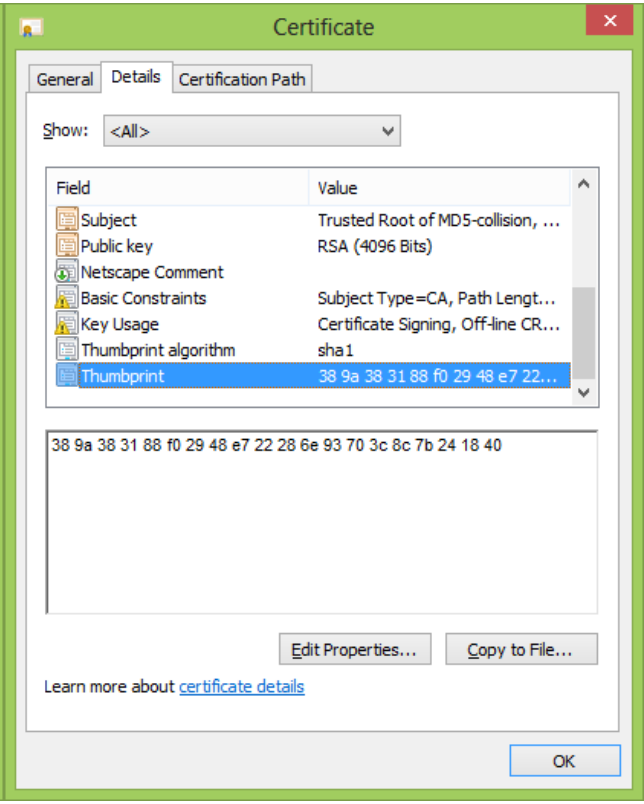
Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

MD5 collision and Xiaoyun Wang, Marc Stevens, Vlastimil Klima



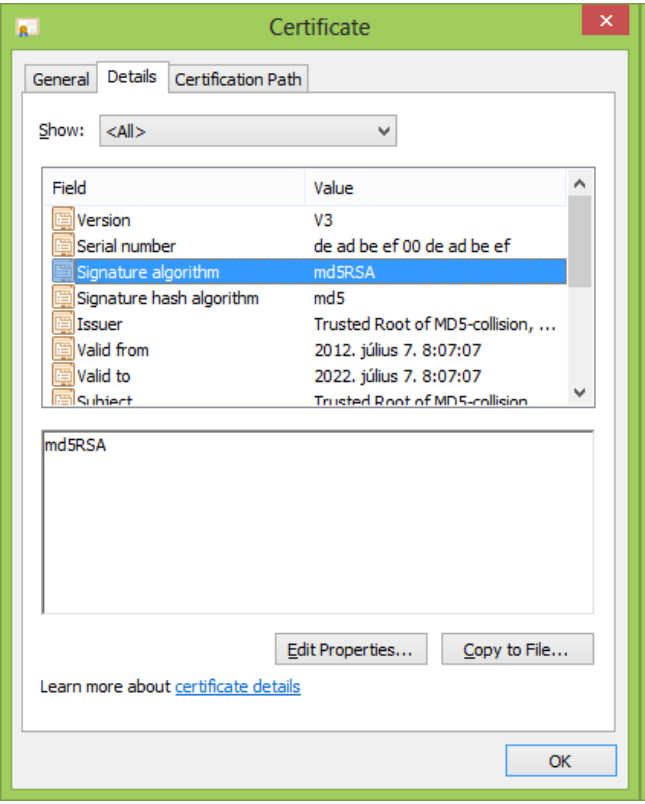


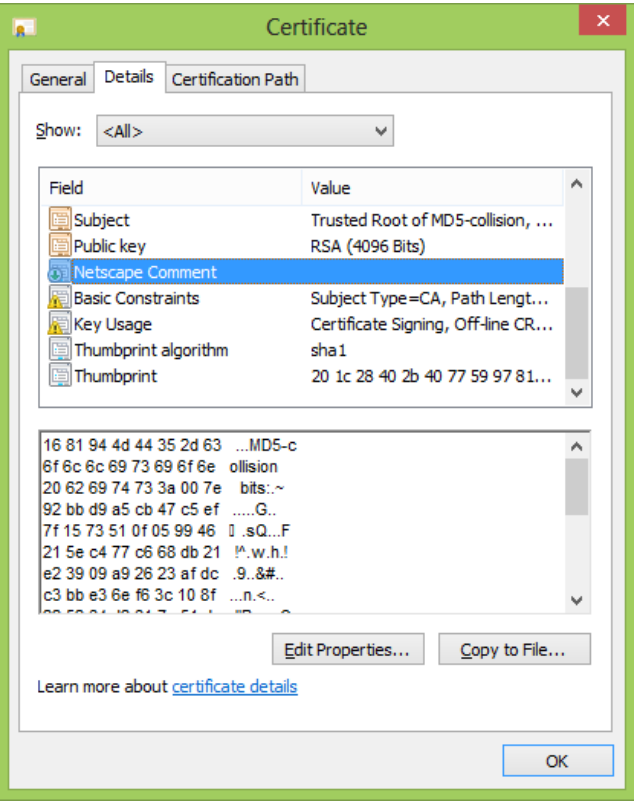


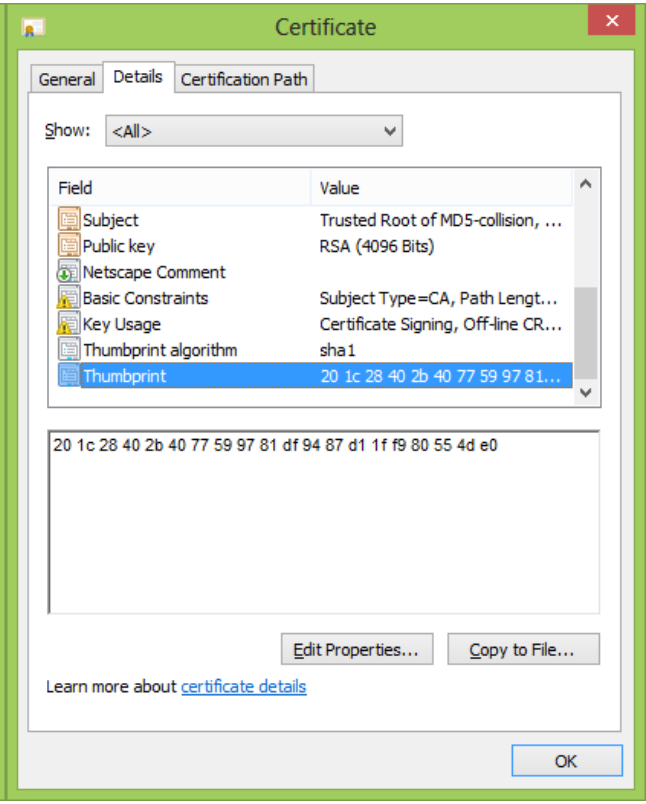
Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

MD5 collision and Xiaoyun Wang, Marc Stevens, Vlastimil Klima



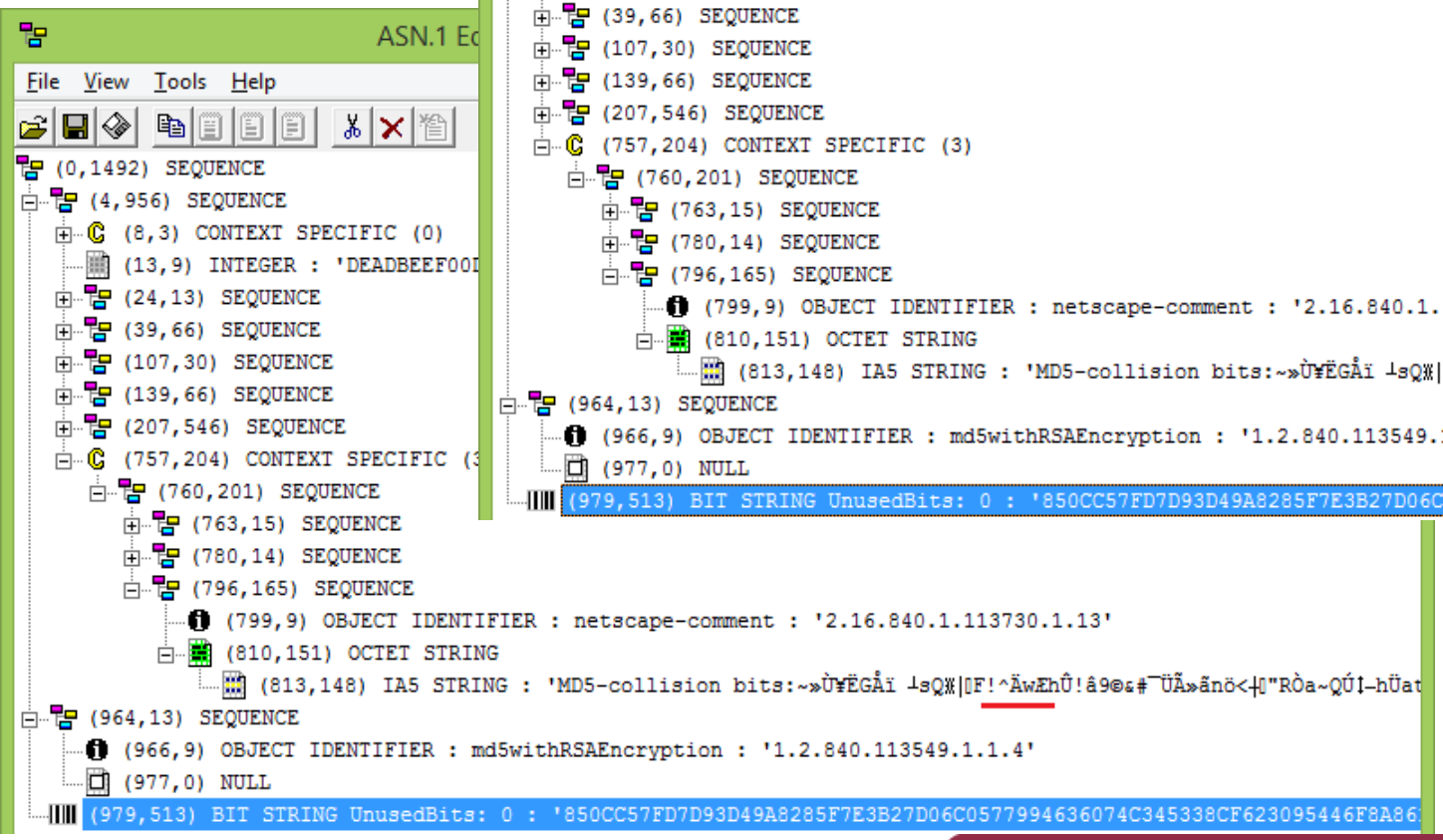




Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

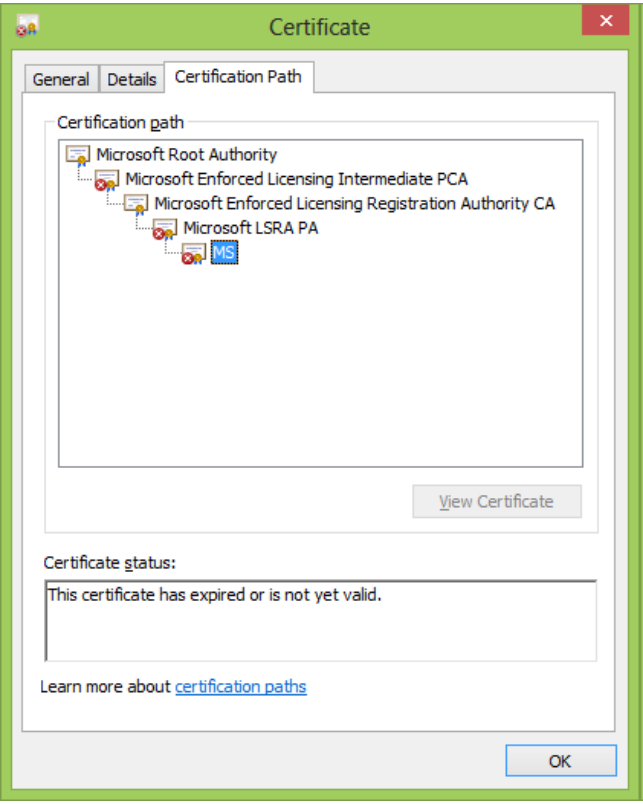
MD5 collision and Xiaoyun Wang, Marc Stevens, Vlastimil Klima

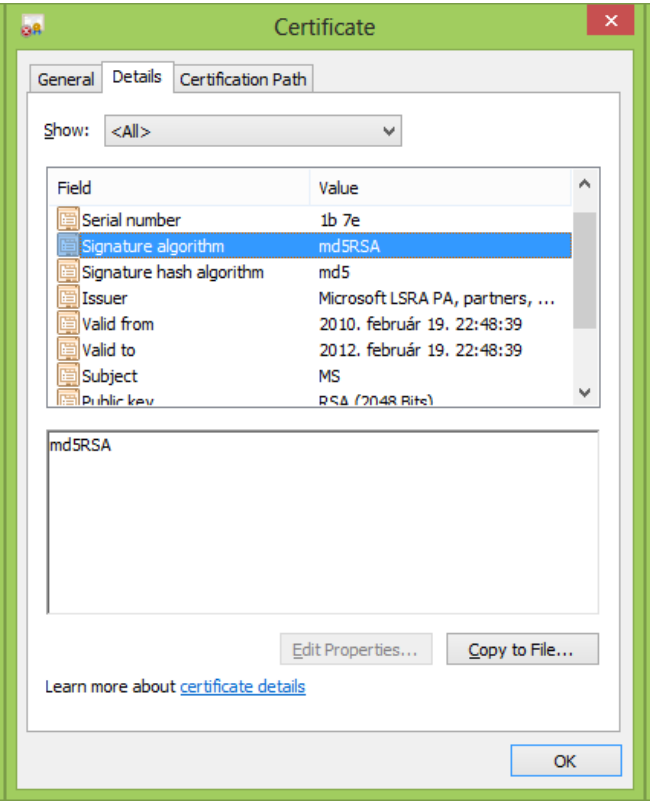


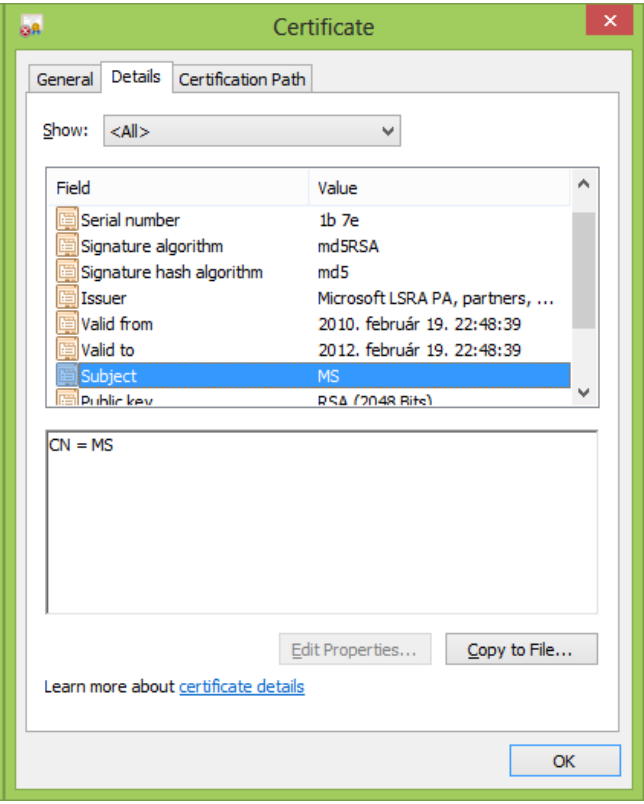
Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

MD5 collision and Flame, WuSetupV.exe



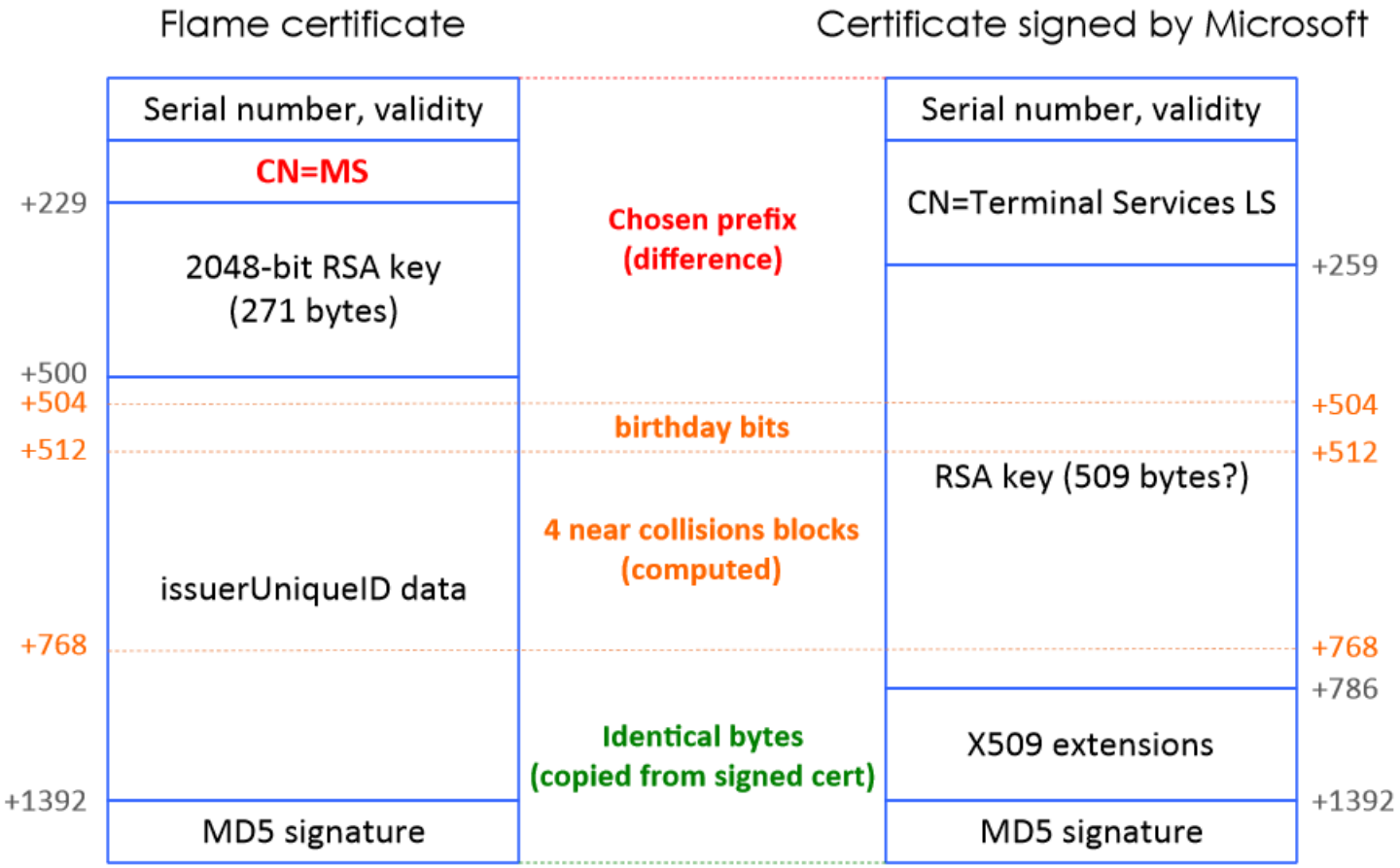




Algorithm anatomy

MD5 collision and Flame, WuSetupV.exe

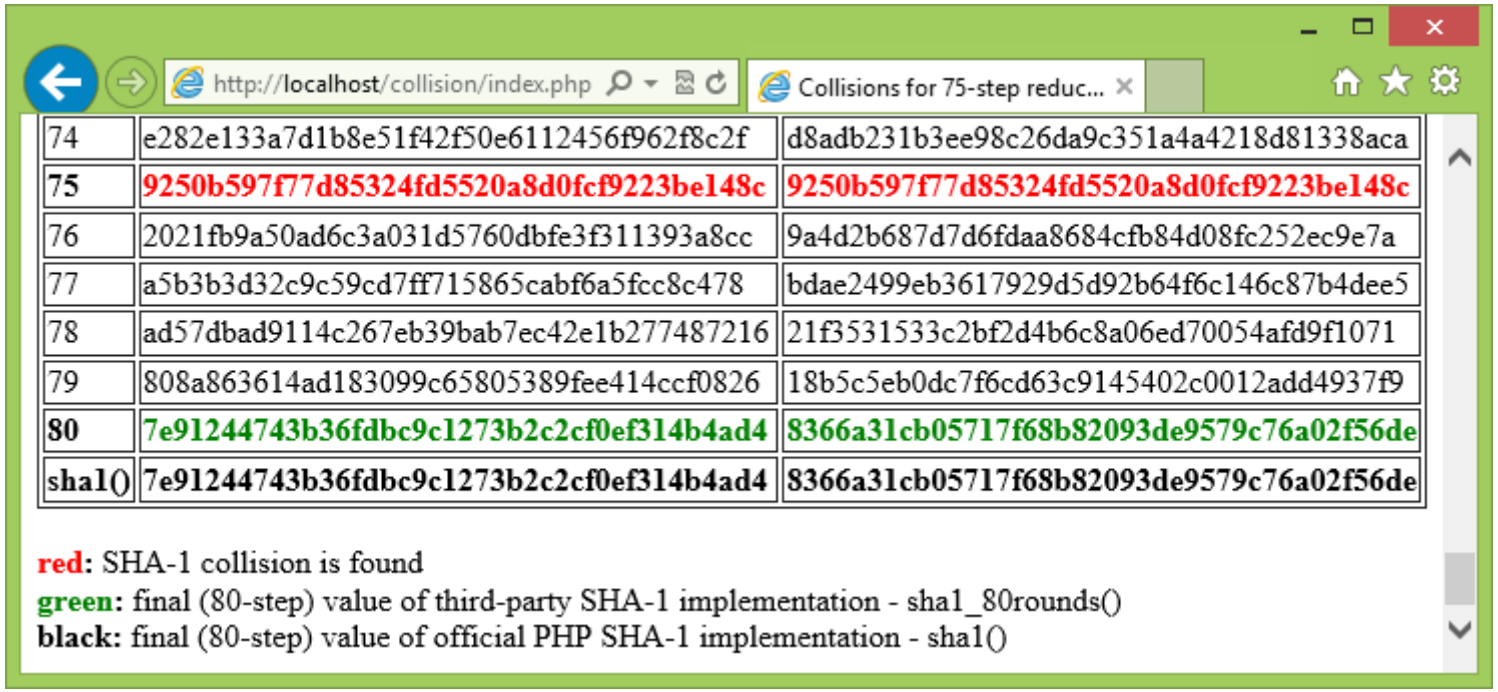
- Marc Stevens: chosen-prefix collisions
- Alex Sotirov: analysis of Flame certificate



Algorithm anatomy

SHA-1 collision and E. A. Grechnikov

- collisions for 75-step reduced SHA-1
- using SHA-2 is recommended
- NIST selected Keccak (2012-10-02) as the basis for SHA-3



74	e282e133a7d1b8e51f42f50e6112456f962f8c2f	d8adb231b3ee98c26da9c351a4a4218d81338aca
75	**9250b597f77d85324fd5520a8d0fcf9223be148c**	**9250b597f77d85324fd5520a8d0fcf9223be148c**
76	2021fb9a50ad6c3a031d5760dbfe3f311393a8cc	9a4d2b687d7d6fdaa8684cfb84d08fc252ec9e7a
77	a5b3b3d32c9c59cd7ff715865cabf6a5fcc8c478	bdae2499eb3617929d5d92b64f6c146c87b4dee5
78	ad57dbad9114c267eb39bab7ec42e1b277487216	21f3531533c2bf2d4b6c8a06ed70054afd9f1071
79	808a863614ad183099c65805389fee414ccf0826	18b5c5eb0dc7f6cd63c9145402c0012add4937f9
80	**7e91244743b36fdbc9c1273b2c2cf0ef314b4ad4**	**8366a31cb05717f68b82093de9579c76a02f56de**
sha1()	**7e91244743b36fdbc9c1273b2c2cf0ef314b4ad4**	**8366a31cb05717f68b82093de9579c76a02f56de**

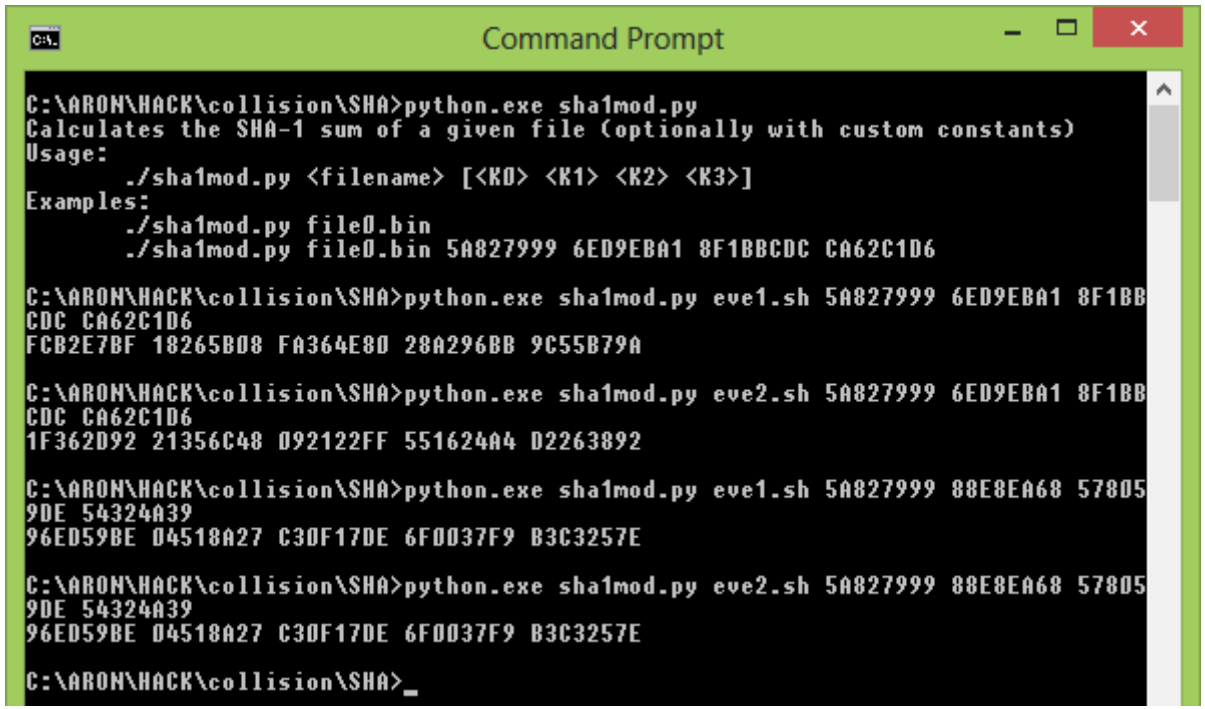
red: SHA-1 collision is found
green: final (80-step) value of third-party SHA-1 implementation - sha1_80rounds()
black: final (80-step) value of official PHP SHA-1 implementation - sha1()

Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

SHA-1 collision and Malicious SHA-1 project

- collisions for 80-step (full) SHA-1 with modified constant words K(t)
- K(t) = [5A827999, 6ED9EBA1, 8F1BBCDC, CA62C1D6]

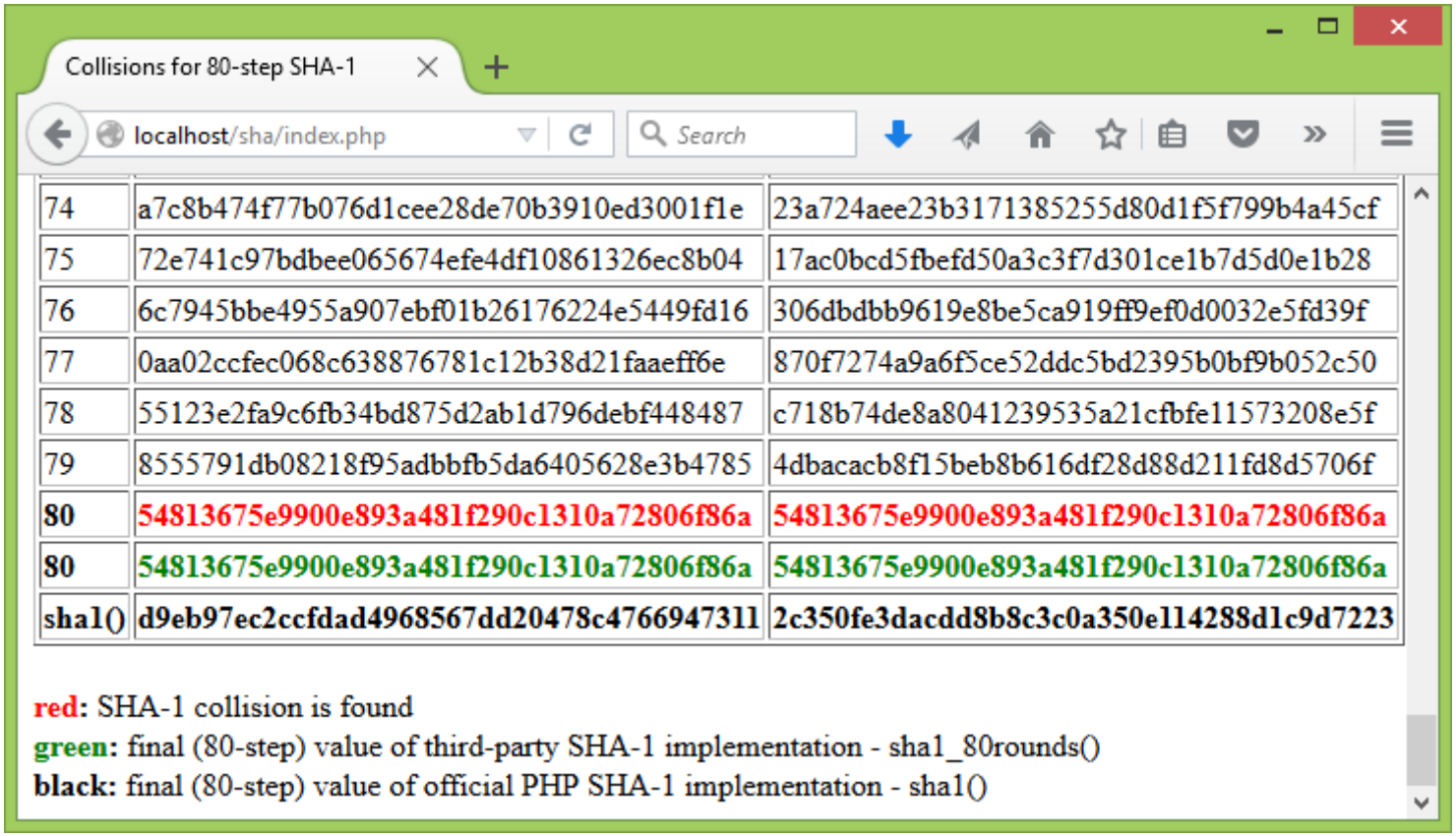


Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

SHA-1 collision and Marc Stevens, Pierre Karpman, Thomas Peyrin

- collisions for 80-step (full) SHA-1 with modified initialization vectors H0..H4
- H0..H4 = [506B0178, FF6D1890, 202291FD, 3ADE3871, B2C665EA]



74	a7c8b474f77b076d1cee28de70b3910ed3001f1e	23a724aee23b3171385255d80d1f5f799b4a45cf
75	72e741c97bdbee065674efe4df10861326ec8b04	17ac0bcd5fbefd50a3c3f7d301ce1b7d5d0e1b28
76	6c7945bbe4955a907ebf01b26176224e5449fd16	306dbdbb9619e8be5ca919ff9ef0d0032e5fd39f
77	0aa02ccfec068c638876781c12b38d21faaeff6e	870f7274a9a6f5ce52ddc5bd2395b0bf9b052c50
78	55123e2fa9c6fb34bd875d2ab1d796debf448487	c718b74de8a8041239535a21cfbfe11573208e5f
79	8555791db08218f95adbbfb5da6405628e3b4785	4dbacacb8f15beb8b616df28d88d211fd8d5706f
80	**54813675e9900e893a481f290c1310a72806f86a**	**54813675e9900e893a481f290c1310a72806f86a**
80	**54813675e9900e893a481f290c1310a72806f86a**	**54813675e9900e893a481f290c1310a72806f86a**
sha1()	**d9eb97ec2ccfdad4968567dd20478c4766947311**	**2c350fe3dacdd8b8c3c0a350e114288d1c9d7223**

red: SHA-1 collision is found
green: final (80-step) value of third-party SHA-1 implementation - sha1_80rounds()
black: final (80-step) value of official PHP SHA-1 implementation - sha1()

Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

SHA-1 forging based on MD5 forging and Thai Duong, Juliano Rizzo

- forging hashes of 80-step (full) SHA-1 with modified initialization vectors H0..H4
- H0..H4 = [67452301, EFCDAB89, 98BADCFE, 10325476, C3D2E1F0]
- length extension attack: Flickr, Scribd, Vimeo...

```
transaction parameters (original)

        uid=User1234&txid=12345678&event=moneytransfer&from=User1234&to=User8888&amount=1000

to-be-hashed data (original)

        pwd=Pass1234&
        uid=User1234&txid=12345678&event=moneytransfer&from=User1234&to=User8888&amount=1000

transaction posted parameters (original)

        POST /submit.php HTTP/1.1
        Host: www.mybank.com
        Content-Type: application/x-www-form-urlencoded; charset=UTF-8

        uid=User1234&txid=12345678&event=moneytransfer&from=User1234&to=User8888&amount=1000
        &hash=487f551ef831a9a24594f286b32bb7fac4281c36
```

Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

SHA-1 forging based on MD5 forging and Thai Duong, Juliano Rizzo

- inject data, apply padding, modify initialization vectors, continue hashing

---

```
transaction parameters (injected)

        &from=User1234&to=User6666&amount=1000

to-be-hashed data (injected)

        uid=User1234&txid=12345678&event=moneytransfer&from=User1234&to=User8888&amount=1000
        <padding>
        &from=User1234&to=User6666&amount=1000

transaction posted parameters (injected)

        POST /submit.php HTTP/1.1
        Host: www.mybank.com
        Content-Type: application/x-www-form-urlencoded; charset=UTF-8

        uid=User1234&txid=12345678&event=moneytransfer&from=User1234&to=User8888&amount=1000
        <padding>
        &from=User1234&to=User6666&amount=1000
        &hash=d79c913e23d22609565e28a0ff920250d18f9a72
```

Algorithm anatomy

SHA-1 forging based on MD5 forging and Thai Duong, Juliano Rizzo

Command Prompt

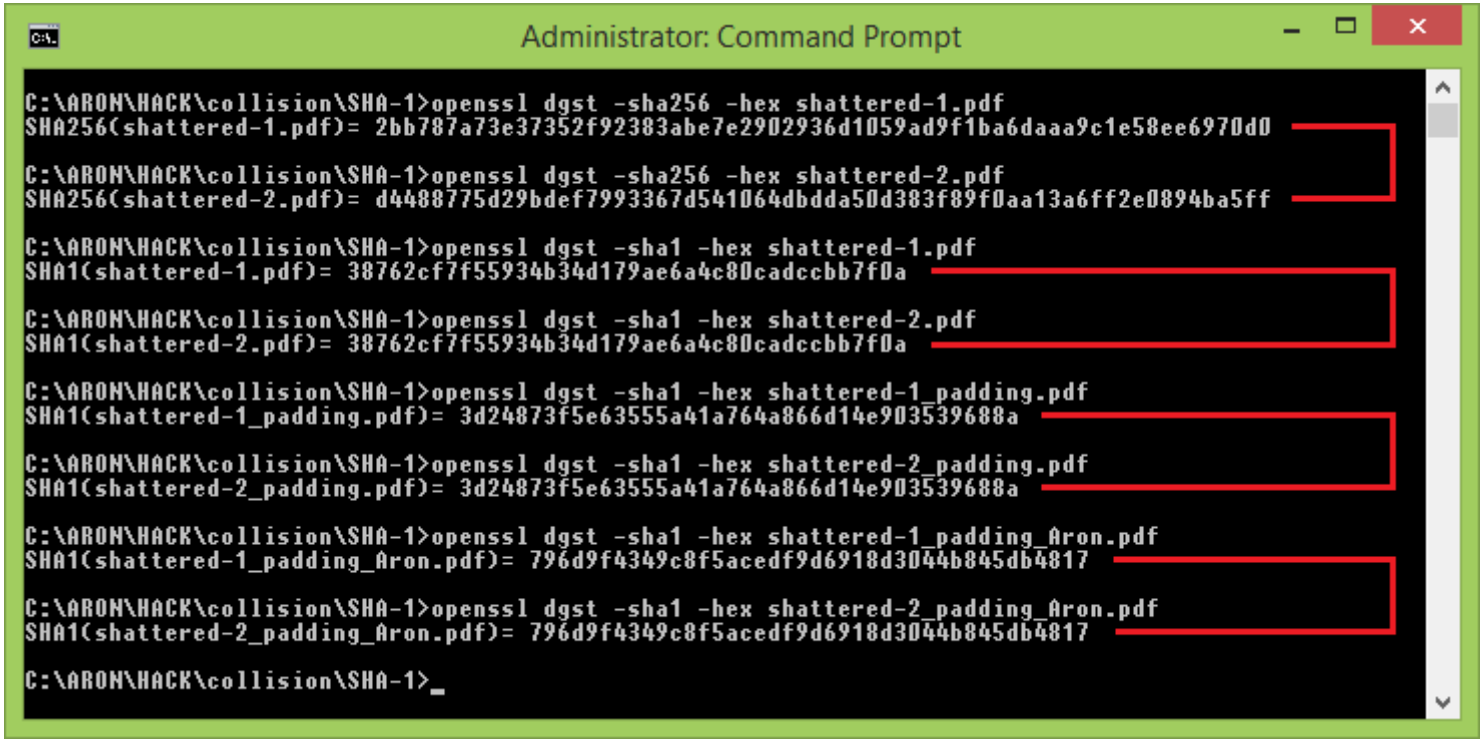
```
C:\ARON\HACK\collision\SHA>python.exe sha1.py "pwd=Pass1234&uid=User1234&txid=12
345678&event=moneytransfer&from=User1234&to=User8888&amount=1000"
msg: 7077643d5061737331323334267569643d557365723132333426747869643d3132333435363
738266576656e743d6d6f6e65797472616e736665722666726f6d3d557365723132333426746f3d5
57365723838383826616d6f756e743d3130303080000000000000000000000000000000000000000
000000000000000000308
h0: 0x67452301
h1: 0xefcdab89L
h2: 0x98badcfeL
h3: 0x10325476
h4: 0xc3d2e1f0L
487f551ef831a9a24594f286b32bb7fac4281c36

C:\ARON\HACK\collision\SHA>python.exe sha1_len_ext_attack.py 487f551ef831a9a2459
4f286b32bb7fac4281c36 97 "&from=User1234&to=User6666&amount=1000"
msg: 2666726f6d3d557365723132333426746f3d557365723636363626616d6f756e743d3130303
08000000000000000000000000000000000000000000000000530
h0: 0x487f551e
h1: 0xf831a9a2L
h2: 0x4594f286
h3: 0xb32bb7faL
h4: 0xc4281c36L
d79c913e23d22609565e28a0ff920250d18f9a72

C:\ARON\HACK\collision\SHA>python.exe sha1.py -xfull 7077643d5061737331323334267
569643d557365723132333426747869643d3132333435363738266576656e743d6d6f6e657974726
16e736665722666726f6d3d557365723132333426746f3d557365723838383826616d6f756e743d3
1303030800000000000000000000000000000000000000000000000000000000003082666726f6d3
d557365723132333426746f3d557365723636363626616d6f756e743d31303030800000000000000
0000000000000000000000000000000000530
msg: 7077643d5061737331323334267569643d557365723132333426747869643d3132333435363
738266576656e743d6d6f6e65797472616e736665722666726f6d3d557365723132333426746f3d5
57365723838383826616d6f756e743d3130303080000000000000000000000000000000000000000
0000000000000000003082666726f6d3d557365723132333426746f3d557365723636363626616d6
f756e743d3130303080000000000000000000000000000000000000000000000000530
h0: 0x67452301
h1: 0xefcdab89L
h2: 0x98badcfeL
h3: 0x10325476
h4: 0xc3d2e1f0L
d79c913e23d22609565e28a0ff920250d18f9a72

C:\ARON\HACK\collision\SHA>_
```

Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

SHA-1 collision and Marc Stevens

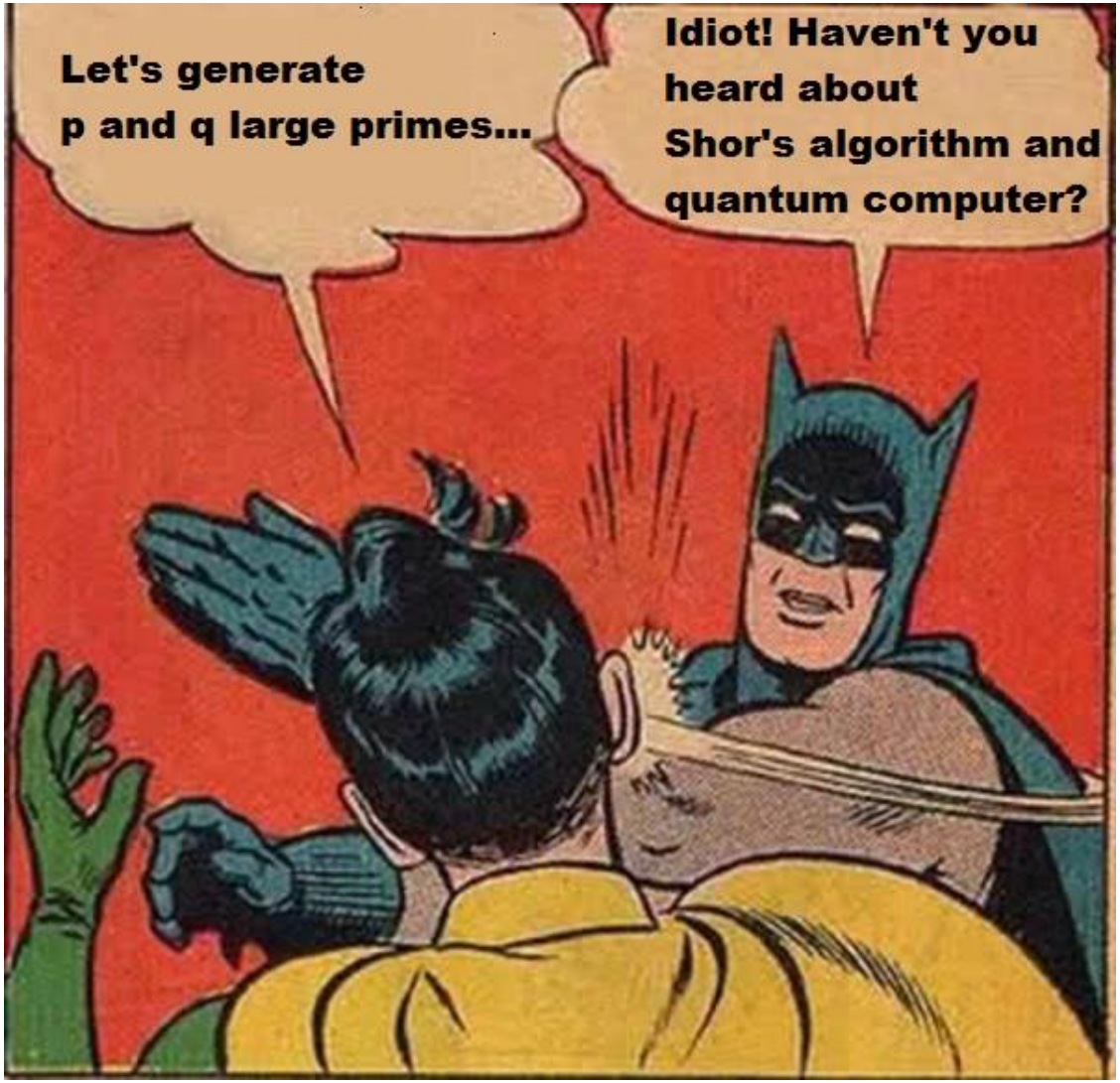
- collisions for 80-step (full) SHA-1 (shattered.io)
- using SHA-2 is recommended
- NIST selected Keccak (2012-10-02) as the basis for SHA-3



Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

pqcrypto



Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

Post-Quantum Cryptography: classical computers and quantum computers

„Imagine that it's fifteen years from now. Somebody announces that he's built a large quantum computer. RSA is dead. DSA is dead. Elliptic curves, hyperelliptic curves, class groups, whatever, dead, dead, dead. So users are going to run around screaming and say 'Oh my God, what do we do?' Well, we still have secret-key cryptography, and we still have some public-key systems. There's hash trees. There's NTRU. There's McEliece. There's multivariate-quadratic systems."

http://pqcrypto.org/

OK, but why?
Shor's algorithm (finding order of a group) and Grover's algorithm runs faster...

So, now what?

signature:	we can wait until real quantum computer is created...
encryption:	now it is already too late!
	Quantum-Safe Perfect Forward Secrecy (QSPFS)

Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

pqcrypto

```
Shor's algorithm

        n = 15                                                  // to-be-factorized integer
        a = 7                                                   // random number, where "a < n"

        Calculate "r" (period/order):

                f(x) = a^x mod n = f(x + r) = a^(x + r) mod n
                1. 7^1 mod 15 = 7
                2. 7^2 mod 15 = 4
                3. 7^3 mod 15 = 13
                4. 7^4 mod 15 = 1
                5. 7^5 mod 15 = 7                               // f(1) = f(5)
                6. 7^6 mod 15 = 4
                7. ...

        r = 4                                                   // period/order
```

Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

pqcrypto

```
Shor's algorithm

        Calculate "gcd(a^(r/2) +/- 1, n)" (greatest common divisor):

                p = gcd(48, 15) = ?
                q = gcd(50, 15) = ?

                //Euclidean algorithm:
                gcd(48, 15) => 48 = q[0] * 15 + r[0]          // q[0] = 3, r[0] = 3
                gcd(15, 3)  => 15 = q[1] * 3  + r[1]          // q[1] = 5, r[1] = 0
                The final non-zero remainder is r[0] = 3

                gcd(50, 15) => 50 = q[0] * 15 + r[0]          // q[0] = 3, r[0] = 5
                gcd(15, 5)  => 15 = q[1] * 5  + r[1]          // q[1] = 3, r[1] = 0
                The final non-zero remainder is r[0] = 5

        p = gcd(48, 15) = 3                                   // prime factor of "n = 15"
        q = gcd(50, 15) = 5                                   // prime factor of "n = 15"
```

Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

ETSI: Quantum Safe Cryptography v1.0.0 (2014-10)

„[...] symmetric key algorithms like AES that can be broken faster by a quantum computer running Grover's algorithm than by a classical computer. [...] This is to say that AES-128 is as difficult for a classical computer to break as AES-256 would be for a quantum computer."

Table 1 - Comparison of conventional and quantum security levels of some popular ciphers.

Algorithm	Key Length	Effective Key Strength / Security Level	
		Conventional Computing	Quantum Computing
RSA-1024	1024 bits	80 bits	0 bits
RSA-2048	2048 bits	112 bits	0 bits
ECC-256	256 bits	128 bits	0 bits
ECC-384	384 bits	256 bits	0 bits
AES-128	128 bits	128 bits	64 bits
AES-256	256 bits	256 bits	128 bits

Note : Effective key strength for conventional computing derived from NIST SP 800-57 "Recommendation for Key Management"

Algorithm anatomy

E-GROUP
SOFTWARE & BEYOND

Post-Quantum Cryptography: classical computers and quantum computers

Hash-based:
- hash functions, HMAC structures are quantum safe
- signature (Lamport, Merkle etc.)

Lattice-based:
- shortest/closest vector problem (László Lovász, Miklós Ajtai)
- signature, encryption (GGH, NTRU etc.)

Multivariate equations-based:
- signature (UOV, Oil and Vinegar etc.)

Code-based:
- syndrome decoding problem, error-correcting codes
- signature, encryption (McEliece, Niederreiter etc.)

Symmetric key-based:
- encryption (AES, Twofish etc.)

Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

„Websites can use TLS to secure all communications between their servers and web browsers.” (... and other client softwares)

„It builds on the earlier SSL specifications (1994, 1995, 1996) developed by Netscape Communications for adding the HTTPS protocol to their Navigator web browser.”

version	published in	published by
SSLv1.0	unpublished	Netscape
SSLv2.0	1995	Netscape
SSLv3.0	1996	Netscape
TLSv1.0	1999	IETF RFC 2246
TLSv1.1	2006	IETF RFC 4346
TLSv1.2	2008	IETF RFC 5246
TLSv1.3	2018	IETF RFC 8446

https://en.wikipedia.org/wiki/Transport_Layer_Security

Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

OpenSSL

- cryptographic primitives
- CA functions
- SSL/TLS protocol

- multiple supported platforms

- command line interface (CLI)

- open source

- FIPS 140-2 level 1

OpenSSL™
Cryptography and SSL/TLS Toolkit

OpenSSL FIPS Object Module SE
Version 2.0.16
By
OpenSSL Validation Services

OpenSSL FIPS 140-2 Security Policy
Version 2.0.16

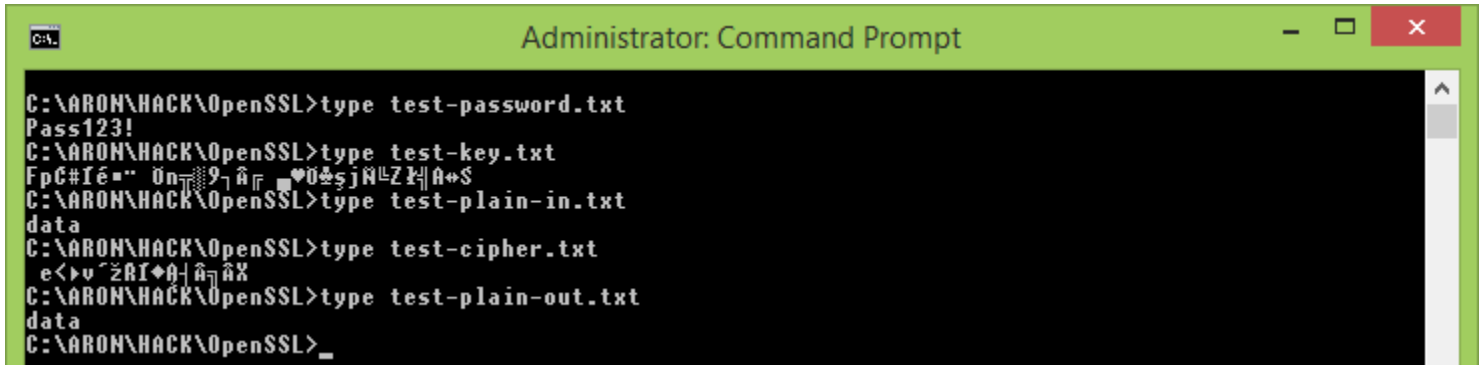
December 26, 2017

Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
openssl dgst
        -sha256 -binary -out test-key.txt test-password.txt
openssl enc
        -aes-256-cbc
        -e -nosalt -kfile test-key.txt -in test-plain-in.txt -out test-cipher.txt

openssl dgst
        -sha256 -binary -out test-key.txt test-password.txt
openssl enc
        -aes-256-cbc
        -d -nosalt -kfile test-key.txt -in test-cipher.txt -out test-plain-out.txt
```

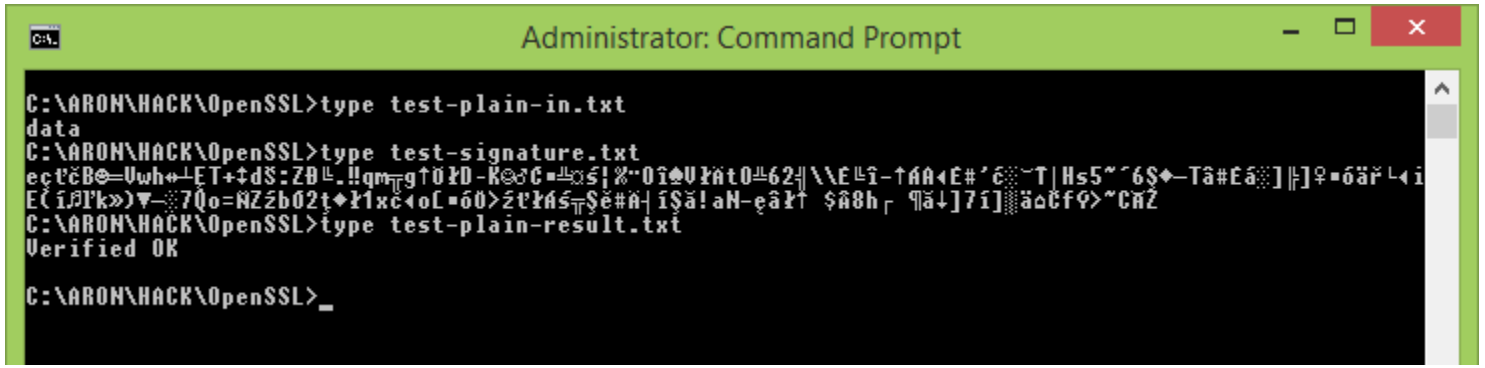


Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
openssl pkcs12
        -in test-user.p12 -out test-user.pem -passin pass:1234 -passout pass:1234
openssl rsa
        -in test-user.pem -passin pass:1234 -out test-user-private-plain.key
openssl rsa
        -in test-user.pem -passin pass:1234 -out test-user-public-plain.key -pubout
openssl dgst
        -sha256 -sign test-user-private-plain.key
        -out test-signature.txt test-plain-in.txt
openssl dgst
        -sha256 -verify test-user-public-plain.key -signature test-signature.txt
        -out test-plain-result.txt test-plain-in.txt
```



Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

Apache HTTP Server

httpd.conf

```
Define ENABLE_TLS13 "Yes"
<IfModule ssl_module>
 Include conf/extra/httpd-ahssl.conf
</IfModule>
LoadModule               ssl_module     modules/mod_ssl.so
#LoadModule              deflate_module modules/mod_deflate.so
```

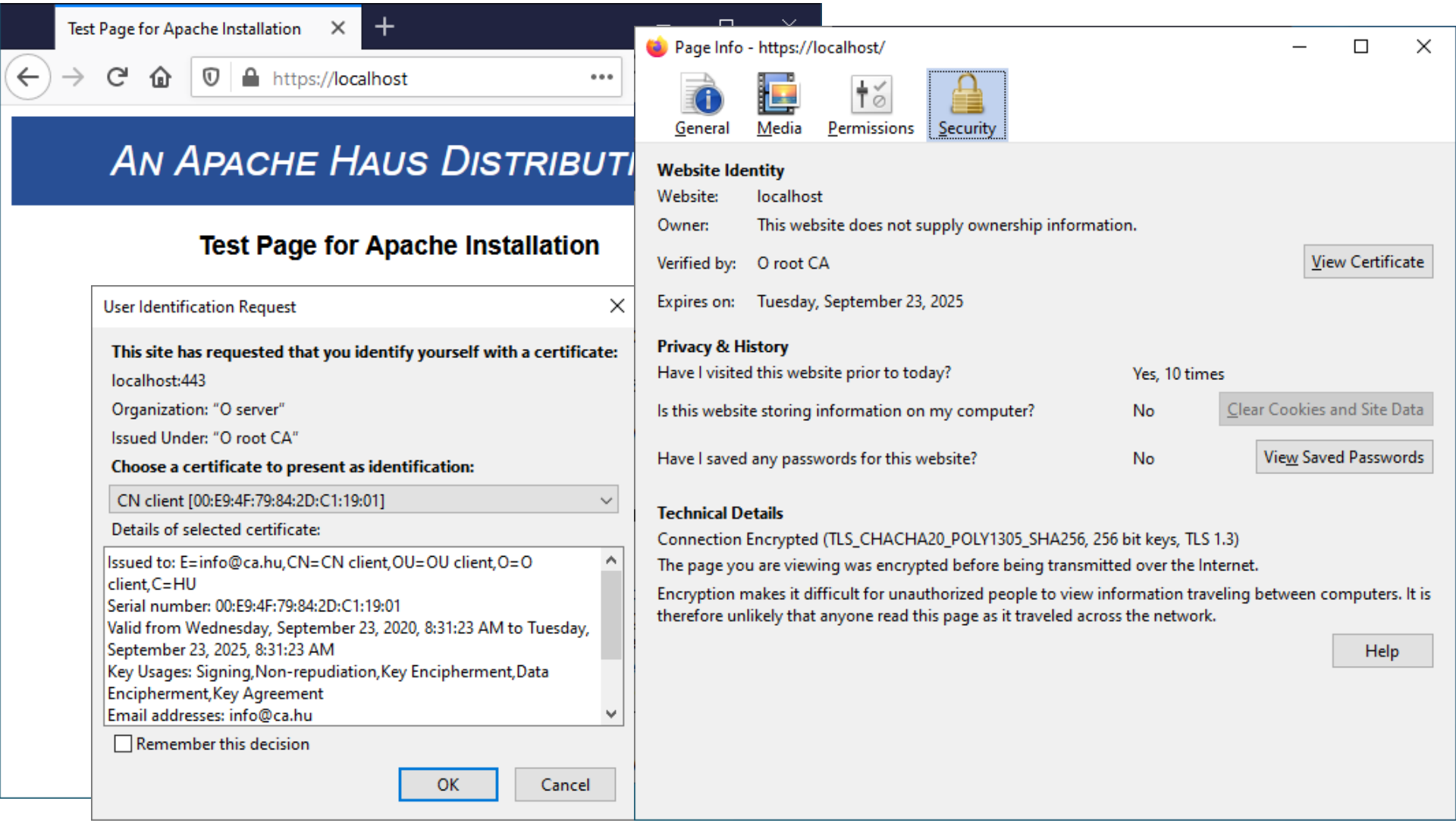
httpd-ahssl.conf

```
ServerName               localhost:443                       (name/address:port)
SSLEngine                on                                  (SSL/TLS enabled)
SSLCipherSuite           TLSv1.3
                         TLS_CHACHA20_POLY1305_SHA256:
                         TLS_AES_256_GCM_SHA384:
                         TLS_AES_128_GCM_SHA256              (set of permitted algorithms)
SSLProtocol              -all +TLSv1.3                       (set of permitted protocol versions)
SSLCompression           off                                 (compression of SSL/TLS layer)
SSLInsecureRenegotiation off                                 (proper management of ClientHello)
SSLCertificateFile       "${SRVROOT}/conf/ssl/web_server.crt" (SSL/TLS server certificate)
SSLCertificateKeyFile    "${SRVROOT}/conf/ssl/web_server.key" (SSL/TLS server private key)
SSLCACertificateFile     "${SRVROOT}/conf/ssl/root_CA.crt"   (CA certificate "whitelist")
SSLOptions               +StdEnvVars +ExportCertData         (SSL/TLS handshake parameters)
SSLVerifyClient          require                             (SSL/TLS client authentication)
```

Protocol anatomy

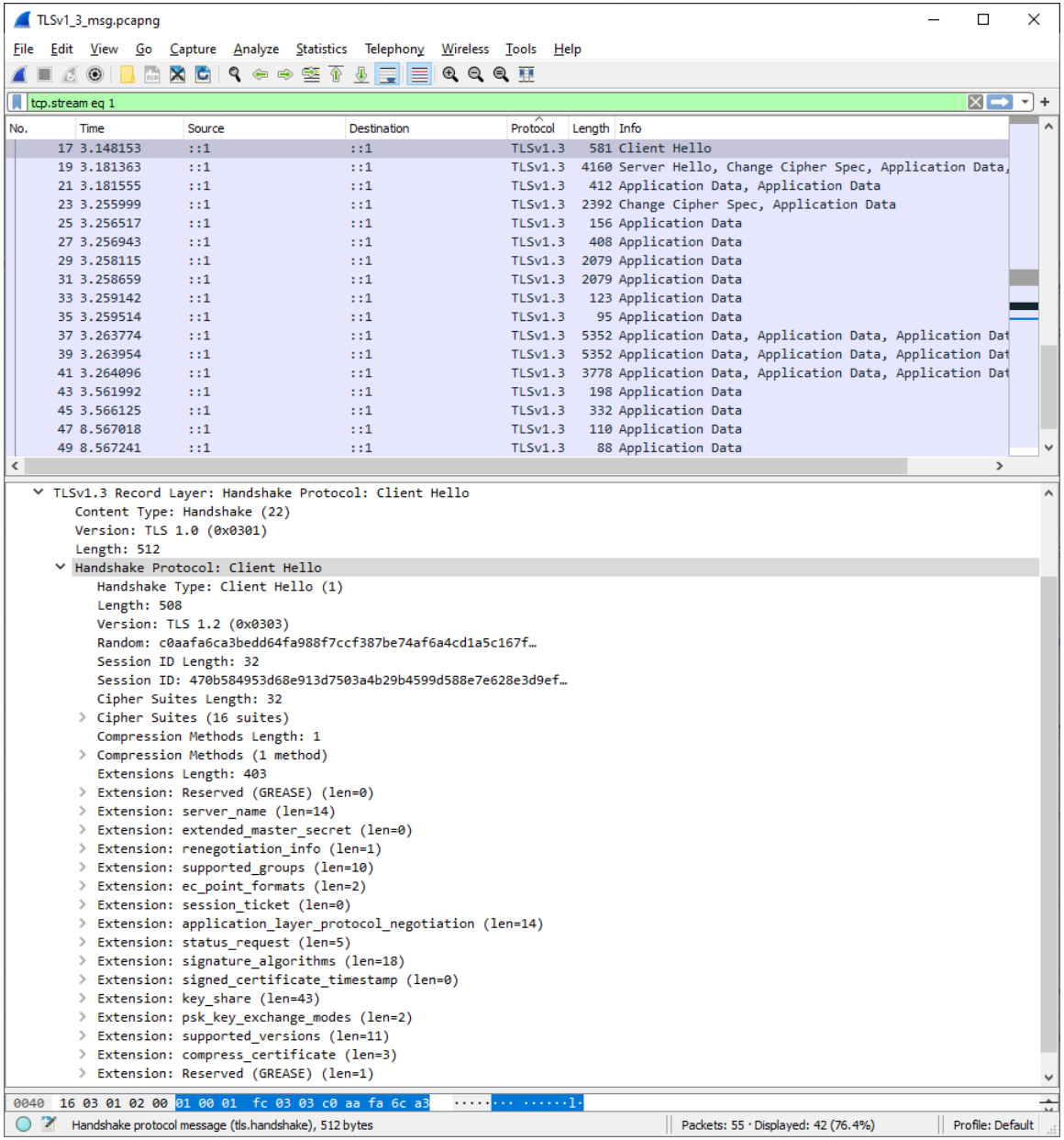
E-GROUP
SOFTWARE & BEYOND

SSL/TLS



Protocol anatomy

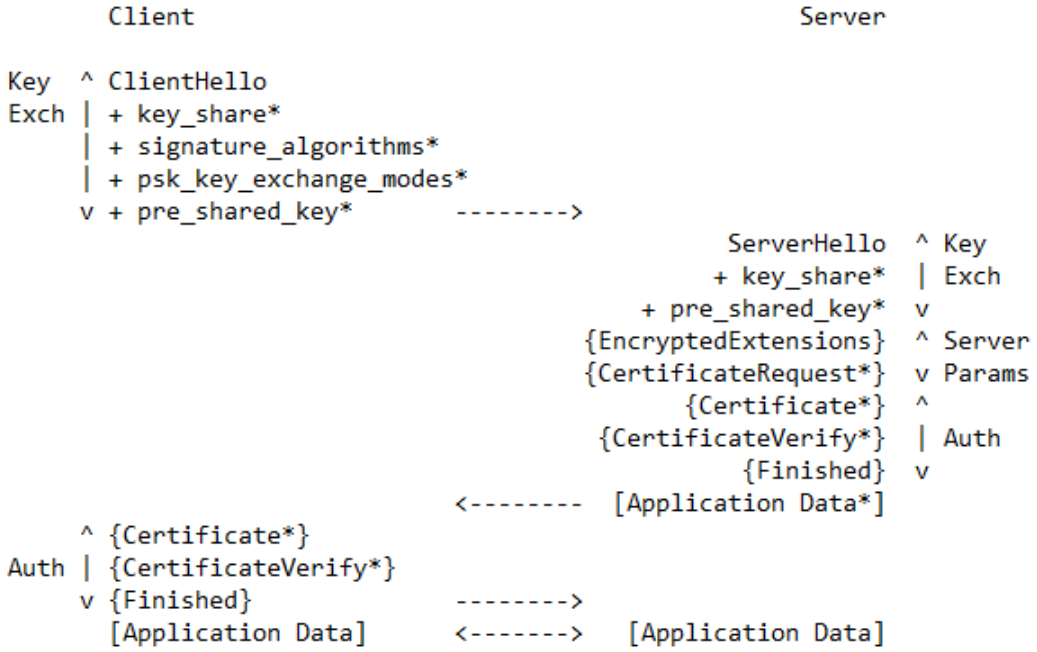
SSL/TLS



Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS



Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
          Client                                           Server

   Key  ^ ClientHello
   Exch | + key_share*
        | + signature_algorithms*
        | + psk_key_exchange_modes*
        v + pre_shared_key*       -------->
                                                   ServerHello  ^ Key
                                                  + key_share*  | Exch
                                             + pre_shared_key*  v
                                         {EncryptedExtensions}  ^ Server
                                         {CertificateRequest*}  v Params
                                                {Certificate*}  ^
                                          {CertificateVerify*}  | Auth
                                                    {Finished}  v
                                   <--------  [Application Data*]
        ^ {Certificate*}
   Auth | {CertificateVerify*}
        v {Finished}              -------->
          [Application Data]      <------->  [Application Data]
```

Command Prompt

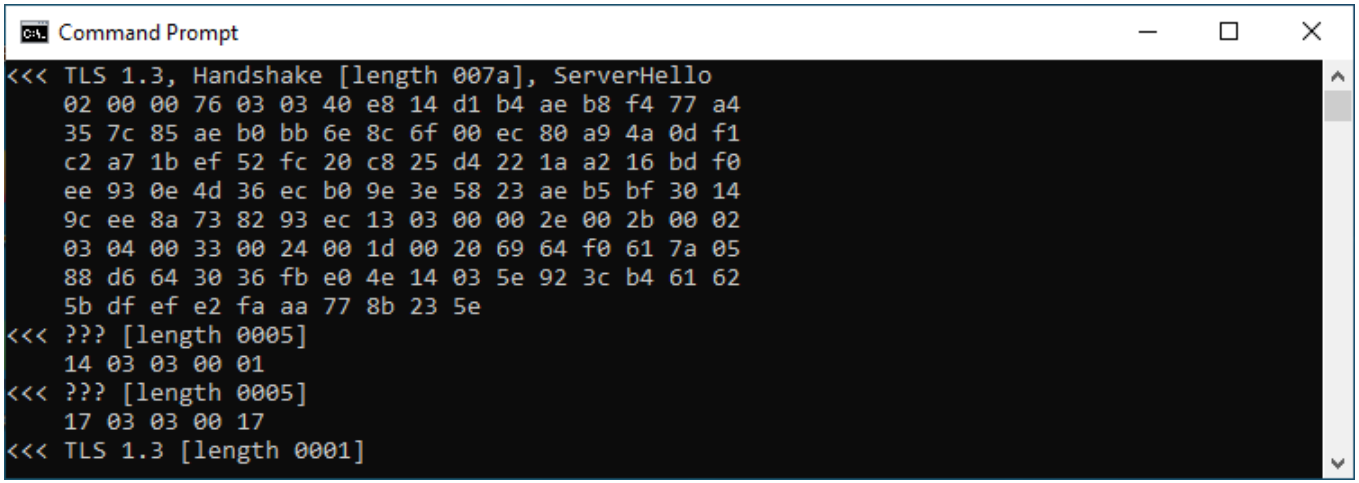
```
C:\ARON\TOOLS\OpenSSL\openssl-1.1.1g\bin>openssl s_client -connect localhost:443 -tls1_3 -C
Afile test-root.crt -cert test-user.crt -certform PEM -key test-user.key -keyform PEM -msg
CONNECTED(00000194)
>>> ??? [length 0005]
    16 03 01 00 d2
>>> TLS 1.3, Handshake [length 00d2], ClientHello
    01 00 00 ce 03 03 4c fc 78 28 be bf ac 0a 12 b8
    c2 22 7a c8 7a 4f a1 6a 91 0a 92 76 7e c4 6b 52
    c9 ca 18 c0 27 d3 20 c8 25 d4 22 1a a2 16 bd f0
    ee 93 0e 4d 36 ec b0 9e 3e 58 23 ae b5 bf 30 14
    9c ee 8a 73 82 93 ec 00 08 13 02 13 03 13 01 00
    ff 01 00 00 7d 00 0b 00 04 03 00 01 02 00 0a 00
    0c 00 0a 00 1d 00 17 00 1e 00 19 00 18 00 23 00
    00 00 16 00 00 00 17 00 00 00 0d 00 1e 00 1c 04
```

Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
          Client                                           Server

   Key  ^ ClientHello
   Exch | + key_share*
        | + signature_algorithms*
        | + psk_key_exchange_modes*
        v + pre_shared_key*       -------->
                                                   ServerHello  ^ Key
                                                 + key_share*   | Exch
                                              + pre_shared_key* v
                                          {EncryptedExtensions}  ^ Server
                                          {CertificateRequest*}  v Params
                                                 {Certificate*}  ^
                                           {CertificateVerify*}  | Auth
                                                     {Finished}  v
                                    <--------  [Application Data*]
        ^ {Certificate*}
   Auth | {CertificateVerify*}
        v {Finished}              -------->
          [Application Data]      <------->  [Application Data]
```

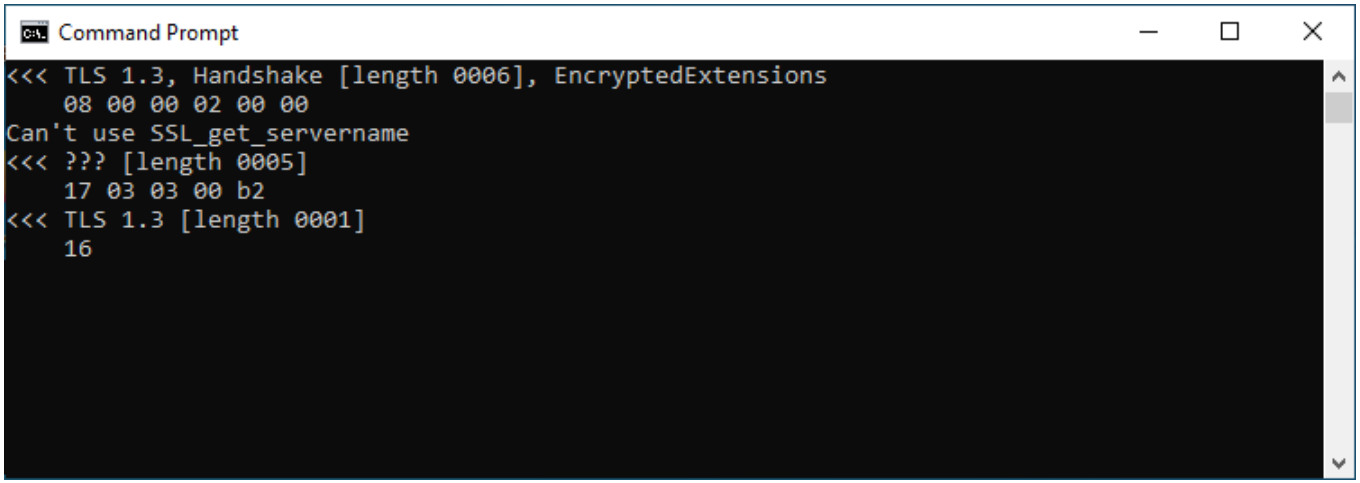


Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
       Client                                           Server

Key  ^ ClientHello
Exch | + key_share*
     | + signature_algorithms*
     | + psk_key_exchange_modes*
     v + pre_shared_key*       -------->
                                                ServerHello  ^ Key
                                              + key_share*   | Exch
                                         + pre_shared_key*   v
                                      {EncryptedExtensions}  ^ Server
                                      {CertificateRequest*}  v Params
                                             {Certificate*}  ^
                                       {CertificateVerify*}  | Auth
                                                 {Finished}  v
                               <--------  [Application Data*]
     ^ {Certificate*}
Auth | {CertificateVerify*}
     v {Finished}              -------->
       [Application Data]      <------->  [Application Data]
```

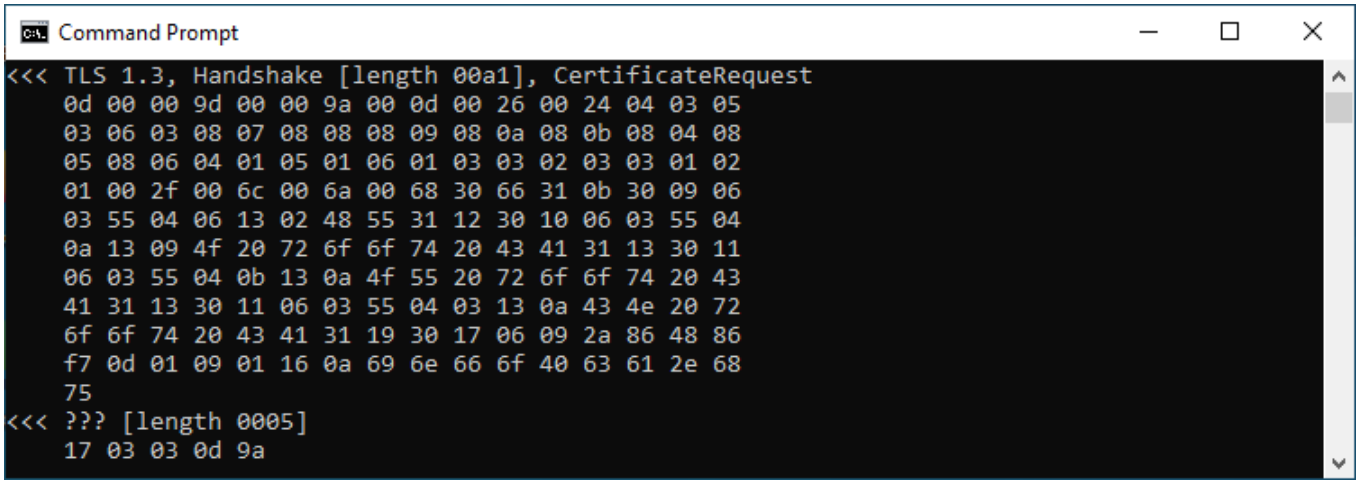


Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
          Client                                           Server

   Key  ^ ClientHello
   Exch | + key_share*
        | + signature_algorithms*
        | + psk_key_exchange_modes*
        v + pre_shared_key*       -------->
                                                   ServerHello  ^ Key
                                                  + key_share*  | Exch
                                             + pre_shared_key*  v
                                         {EncryptedExtensions}  ^ Server
                                         {CertificateRequest*}  v Params
                                                {Certificate*}  ^
                                          {CertificateVerify*}  | Auth
                                                    {Finished}  v
                                   <--------  [Application Data*]
        ^ {Certificate*}
   Auth | {CertificateVerify*}
        v {Finished}              -------->
          [Application Data]      <------->  [Application Data]
```



Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
          Client                                           Server

   Key  ^ ClientHello
   Exch | + key_share*
        | + signature_algorithms*
        | + psk_key_exchange_modes*
        v + pre_shared_key*       -------->
                                                   ServerHello  ^ Key
                                                  + key_share*  | Exch
                                             + pre_shared_key*  v
                                         {EncryptedExtensions}  ^ Server
                                         {CertificateRequest*}  v Params
                                                {Certificate*}  ^
                                          {CertificateVerify*}  | Auth
                                                    {Finished}  v
                                   <--------  [Application Data*]
        ^ {Certificate*}
   Auth | {CertificateVerify*}
        v {Finished}              -------->
          [Application Data]      <------->  [Application Data]
```

Command Prompt

```
<<< TLS 1.3, Handshake [length 0d89], Certificate
    0b 00 0d 85 00 00 0d 81 00 06 c3 30 82 06 bf 30
    82 04 a7 a0 03 02 01 02 02 09 00 cf 88 ac 40 bc
    49 6e 44 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b
    05 00 30 66 31 0b 30 09 06 03 55 04 06 13 02 48
    55 31 12 30 10 06 03 55 04 0a 13 09 4f 20 72 6f
depth=1 C = HU, O = O root CA, OU = OU root CA, CN = CN root CA, emailAddress = info@ca.hu
verify return:1
depth=0 C = HU, O = O server, OU = OU server, CN = localhost, emailAddress = info@ca.hu
verify return:1
<<< ??? [length 0005]
    17 03 03 02 19
<<< TLS 1.3 [length 0001]
    16
```

Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
          Client                                           Server

   Key  ^ ClientHello
   Exch | + key_share*
        | + signature_algorithms*
        | + psk_key_exchange_modes*
        v + pre_shared_key*       -------->
                                                   ServerHello  ^ Key
                                                  + key_share*  | Exch
                                             + pre_shared_key*  v
                                         {EncryptedExtensions}  ^ Server
                                         {CertificateRequest*}  v Params
                                                {Certificate*}  ^
                                          {CertificateVerify*}  | Auth
                                                    {Finished}  v
                                   <--------  [Application Data*]
        ^ {Certificate*}
   Auth | {CertificateVerify*}
        v {Finished}              -------->
          [Application Data]      <------->  [Application Data]
```

Command Prompt

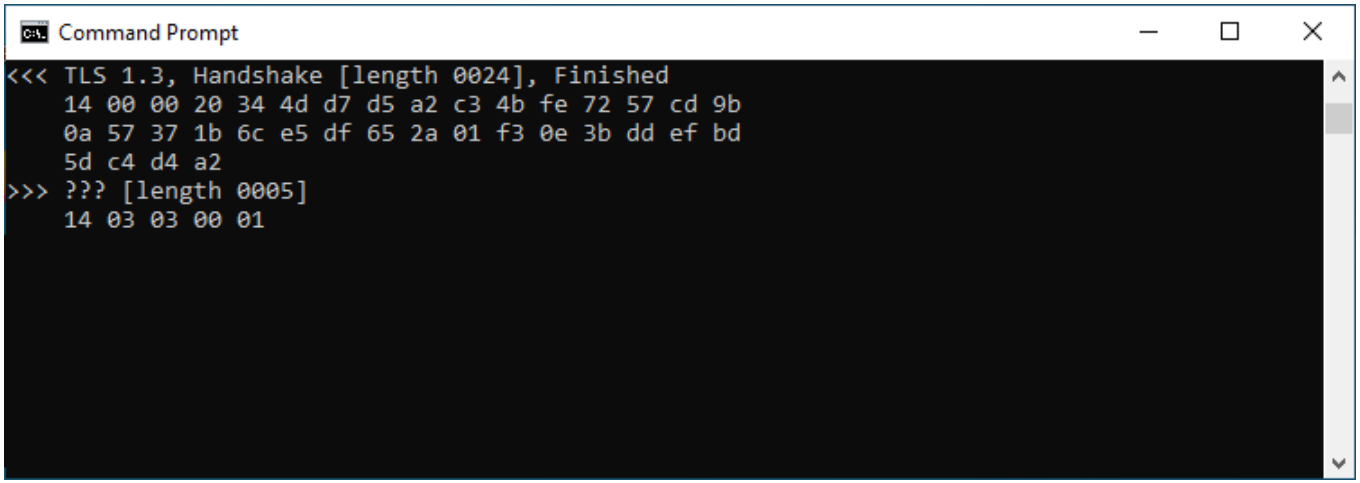
```
<<< TLS 1.3, Handshake [length 0208], CertificateVerify
    0f 00 02 04 08 04 02 00 0d 3b e8 d2 c9 30 4a bc
    f5 20 47 1b 52 62 f6 57 31 ae 1f f1 7b 2f 78 19
    85 bb e7 3f 84 6f 35 6e 73 bb 99 50 e9 58 b9 ae
    4d 81 24 97 48 97 3d ec 88 31 ca 43 b7 2b 18 2e
    2f bf 55 5e 13 82 d6 72 6d 4a 36 3d 67 96 d3 21
    78 40 a1 fe ca 3a 92 e3 05 85 13 c5 89 f8 de 4c
    6b 08 db 83 d4 53 34 71 6b ed 8e 40 ad fb 7e 79
    4b b0 61 9a f8 d4 77 04 03 52 ff 22 83 0e e1 5a
    2c ee d5 2e 10 e1 6d 33 63 ba d1 4e 0e df 65 8d
    66 50 53 d5 44 96 99 59 9b be 0d f5 66 04 e0 7e
    9a f8 90 2c e3 30 57 b9 79 54 4f 88 4d 56 ce 1f
    ea 7d 6c ee 9c 03 5b b9 83 4a 6f ff 88 4f 13 29
    83 59 a4 91 ac 5f 4a 64 42 d8 55 eb 3e d9 a6 66
```

Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
          Client                                           Server

   Key  ^ ClientHello
   Exch | + key_share*
        | + signature_algorithms*
        | + psk_key_exchange_modes*
        v + pre_shared_key*       -------->
                                                   ServerHello  ^ Key
                                                  + key_share*  | Exch
                                             + pre_shared_key*  v
                                         {EncryptedExtensions}  ^ Server
                                         {CertificateRequest*}  v Params
                                                {Certificate*}  ^
                                          {CertificateVerify*}  | Auth
                                                    {Finished}  v
                                   <--------  [Application Data*]
        ^ {Certificate*}
   Auth | {CertificateVerify*}
        v {Finished}              -------->
          [Application Data]      <------->  [Application Data]
```

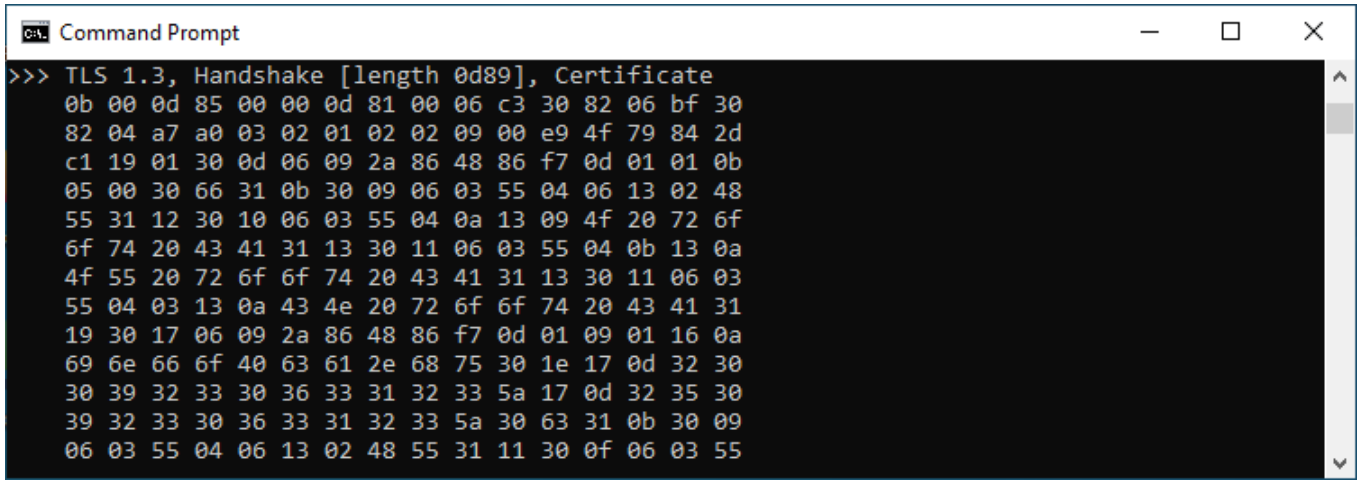


Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
       Client                                           Server

Key  ^ ClientHello
Exch | + key_share*
     | + signature_algorithms*
     | + psk_key_exchange_modes*
     v + pre_shared_key*       -------->
                                                ServerHello  ^ Key
                                              + key_share*   | Exch
                                         + pre_shared_key*   v
                                     {EncryptedExtensions}   ^ Server
                                     {CertificateRequest*}   v Params
                                            {Certificate*}   ^
                                      {CertificateVerify*}   | Auth
                                                {Finished}   v
                               <--------  [Application Data*]
     ^ {Certificate*}
Auth | {CertificateVerify*}
     v {Finished}              -------->
       [Application Data]      <------->  [Application Data]
```

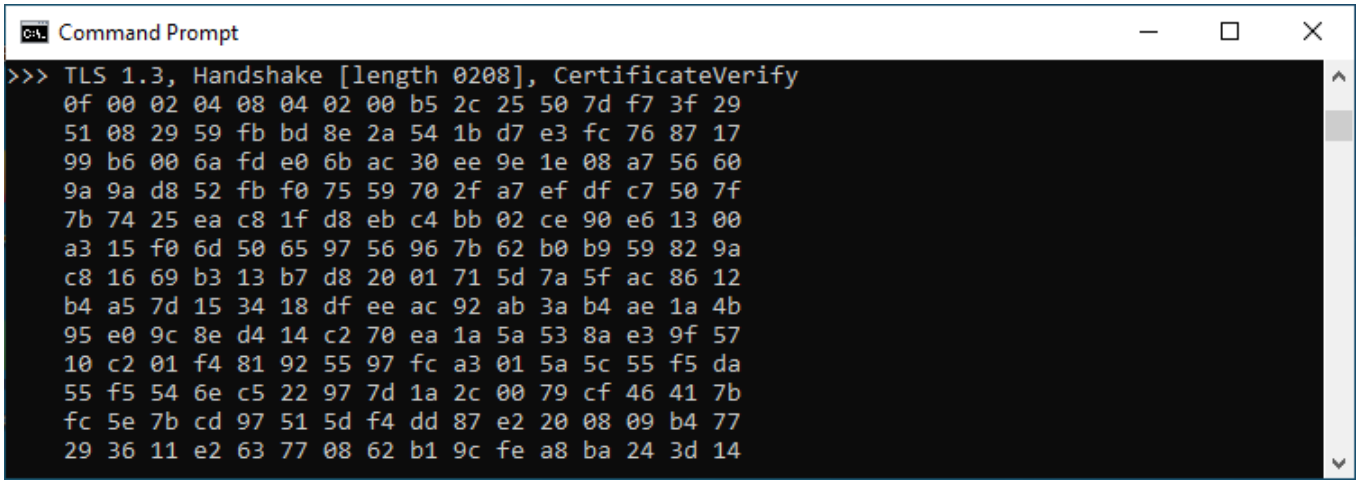


Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

```
       Client                                           Server

Key  ^ ClientHello
Exch | + key_share*
     | + signature_algorithms*
     | + psk_key_exchange_modes*
     v + pre_shared_key*       -------->
                                                ServerHello  ^ Key
                                              + key_share*   | Exch
                                         + pre_shared_key*   v
                                     {EncryptedExtensions}   ^ Server
                                     {CertificateRequest*}   v Params
                                            {Certificate*}   ^
                                      {CertificateVerify*}   | Auth
                                                {Finished}   v
                               <--------  [Application Data*]
     ^ {Certificate*}
Auth | {CertificateVerify*}
     v {Finished}              -------->
       [Application Data]      <------->  [Application Data]
```

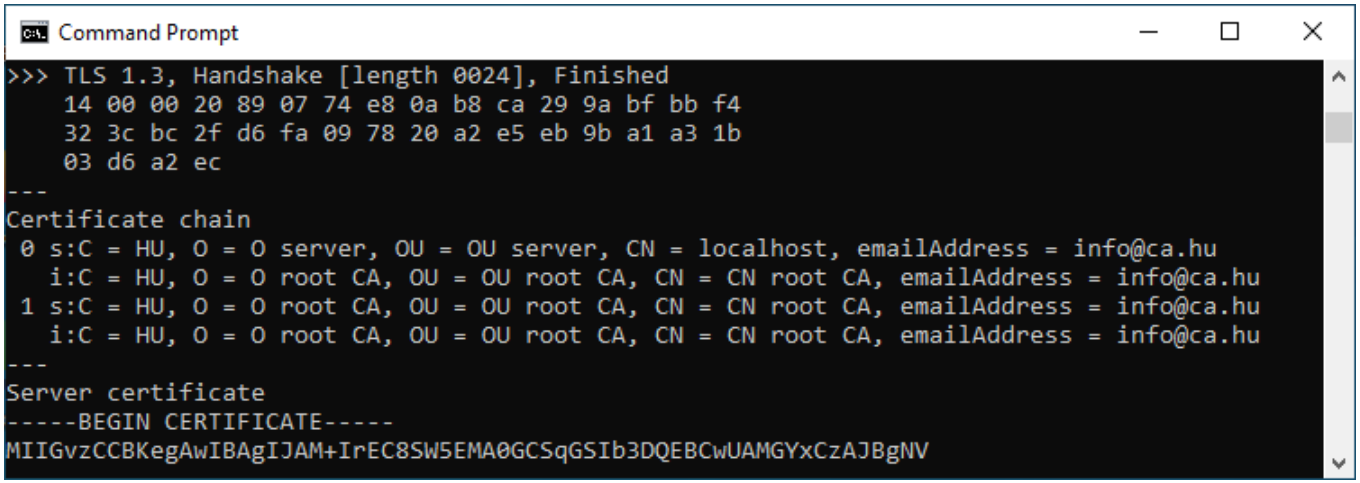


Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

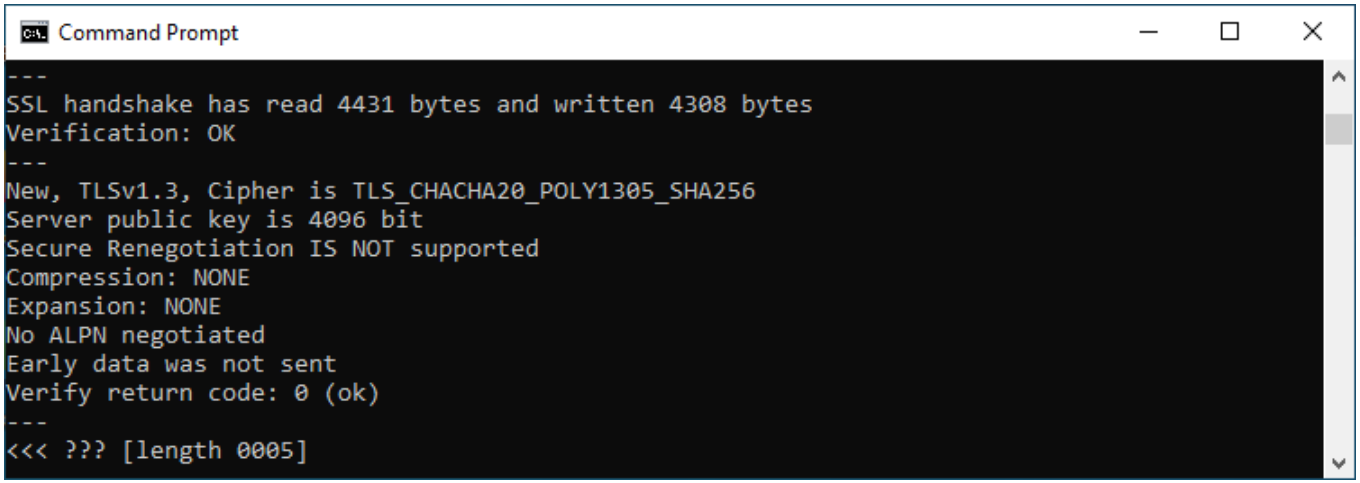
```
          Client                                           Server

   Key  ^ ClientHello
   Exch | + key_share*
        | + signature_algorithms*
        | + psk_key_exchange_modes*
        v + pre_shared_key*       -------->
                                                  ServerHello  ^ Key
                                                 + key_share*  | Exch
                                            + pre_shared_key*  v
                                        {EncryptedExtensions}  ^ Server
                                        {CertificateRequest*}  v Params
                                               {Certificate*}  ^
                                         {CertificateVerify*}  | Auth
                                                   {Finished}  v
                                   <--------  [Application Data*]
        ^ {Certificate*}
   Auth | {CertificateVerify*}
        v {Finished}              -------->
          [Application Data]      <------->  [Application Data]
```

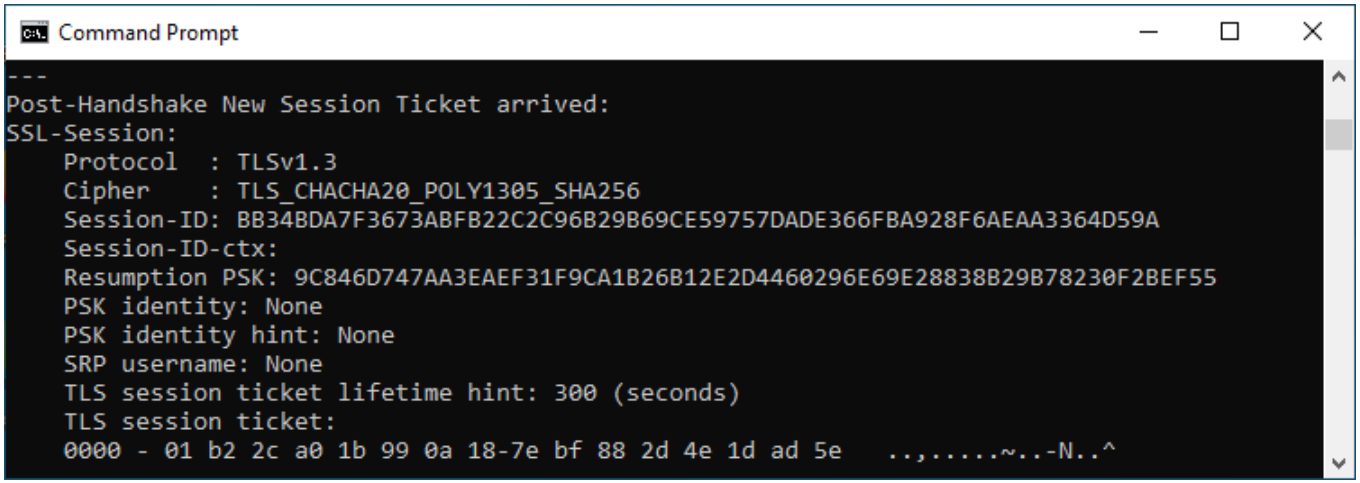


Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS





Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

2009-11-04: renegotiation attack

affected:	SSLv3.0 – TLSv1.2
bug:	ClientHello termination is needed
fixed by:	IETF RFC 5746

2011-09-23: BEAST

affected:	SSLv3.0 – TLSv1.0
bug:	CBC vulnerability, padding oracle
fixed by:	TLSv1.1

2012-09-13: CRIME

affected:	SSLv3.0 – TLSv1.2
bug:	TLS-compression leaks information
fixed by:	disable TLS-compression

2013-02-04: Lucky 13

affected:	SSLv3.0 – TLSv1.2
bug:	CBC vulnerability, padding oracle (time-based)
fixed by:	modify SSLCipherSuite

Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

2013-03-12: RC4

affected:	SSLv3.0 – TLSv1.2
bug:	same messages leak information about the key
fixed by:	enable AES, disable RC4

2013-08-01: BREACH

affected:	SSLv3.0 – TLSv1.2
bug:	HTTP-compression leaks information
fixed by:	disable HTTP-compression

2014-10-14: POODLE

affected:	SSLv3.0
bug:	CBC vulnerability, padding oracle
fixed by:	TLSv1.0

2015-03-03: FREAK

affected:	SSLv3.0 - TLSv1.2
bug:	RSA parameter downgrading
fixed by:	modify SSLCipherSuite, update browser

Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

SSL/TLS

2015-05-20: weakDH, Logjam

affected:	SSLv3.0 - TLSv1.2
bug:	Diffie-Hellman parameter downgrading
fixed by:	modify SSLCipherSuite, update browser

2016-03-01: DROWN

affected:	SSLv2.0
bug:	padding oracle, only RSA KeyExchange
fixed by:	disable SSLv2.0, modify SSLCipherSuite

2020-09-09: RACCOON

affected:	SSLv2.0 - TLSv1.2
bug:	padding oracle (time-based)
fixed by:	disable all, enable TLSv1.3

Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

applied block cipher modes of operation: AES-CBC vs. AES-CTR vs. AES-GCM

- KRACK — Breaking WPA2 by forcing nonce reuse (2017-10-15)
  decrypt and replay messages of WPA2
  AES-CCMP — Counter Mode with CBC-MAC Protocol
  AES-GCMP — Galois/Counter Mode Protocol

”When the victim reinstalls the key, associated parameters such as the incremental transmit packet number (i.e. nonce) and receive packet number (i.e. replay counter) are reset to their initial value. Essentially, to guarantee security, a key should only be installed and used once.”

https://www.krackattacks.com

```
data_plaintext_1.txt
31323334353637383930313233343536313233343536373839303132333435 36

data_plaintext_2.txt
6162636465666768696a6b6c6d6e6f706162636465666768696a6b6c6d6e6f70

data_plaintext_1.txt xor data_plaintext_2.txt
505050505050505050505a5a5e5e5a5a4650505050505050505a5a5e5e5a5a46

data_ciphertext_1.txt
bd10464ad31fa0703c40dc86bcc757f2c76ea3d954da36c8905af9403eca48b6

data_ciphertext_2.txt
ed40161a834ff0206c1a86d8e29d0db4973ef389048a6698c000a31e609012f0

data_ciphertext_1.txt xor data_ciphertext_2.txt
505050505050505050505a5a5e5e5a5a4650505050505050505a5a5e5e5a5a46

data_ciphertext_1.txt xor data_ciphertext_2.txt xor data_plaintext_2.txt = data_plaintext_1.txt
31323334353637383930313233343536313233343536373839303132333435 36
```

Protocol anatomy

E-GROUP
SOFTWARE & BEYOND

GNSS: GPS (US) vs. GALILEO (EU) vs. GLONASS (RU) vs. BeiDou (CN)

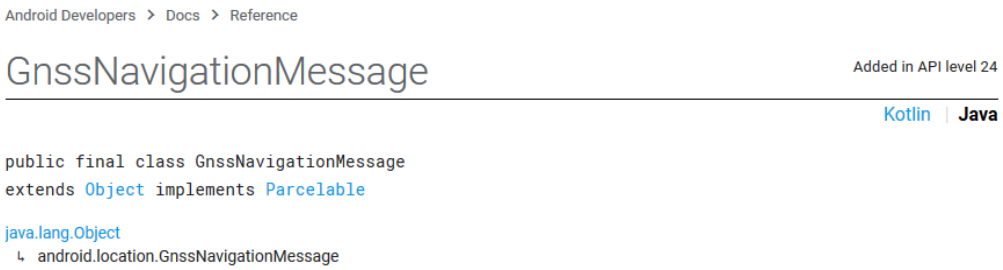
GPS (US)

2011-12-05	Lockheed Martin RQ-170 Sentinel was captured by Iranian forces
2013-07-29	yacht was hijacked from 50 km by University of Texas students
2016-07-15	Pokemon GO was hijacked by HackRF SDR of Stefan Kiese
2017-06-22	ships were hijacked in the Black Sea by Russian forces
2018-08-17	$225 cost HackRF SDR was demonstrated by Chinese students
2019-11-15	ships are cloned at port of Shanghai by Chinese sand thieves

C/A	Civilian Access code (no cryptographic protection)
P(Y)	Precision code (“P-code is XORed with W-code [...] to produce the Y-code”)
M-code	Military code (GPS satellites of Block IIIA – apprx. 2023)

GALILEO (EU)

OSNMA	Open Service Navigation Message Authentication
TESLA	Time Efficient Stream Loss-Tolerant Authentication (IETF RFC 4082)


Android Developers > Docs > Reference

GnssNavigationMessage

Added in API level 24

Kotlin | Java

```
public final class GnssNavigationMessage
extends Object implements Parcelable
```

java.lang.Object
↳ android.location.GnssNavigationMessage

Thank you!

E-GROUP
SOFTWARE & BEYOND

Áron Szabó

mailto: aron.szabo@egroup.hu