Introduction to Applied Cryptography, Part 3*

Prof. Susanne Wetzel
Stevens Institute of Technolgy
Department of Computer Science
Hoboken, New Jersey, USA

Dr. Torsten Schütze
Robert Bosch GmbH
Corporate Sector Research
and Advance Engineering
Software (CR/AEA)

September 21–23, 2009

*The material has been presented at previous Summer Schools for Studienstiftung des Deutschen Volkes while the second author was at Siemens AG, CT IC 3.

Outline of Part 3

1. **Public Key Infrastructures & Key Management**
2. **Elliptic Curve Cryptography**

Part 3: Outline — Section 1: PKIs and Key Management

- 1. Public-key certificates and public-key infrastructures
  - Hierarchical (X.509)
  - Web of trust (PGP)
  - Current practical approaches
- 2. Key Management
  - Key Usage
  - Key Management Life Cycle

1. Authentication of Public Keys

- Encryption and digital signatures based on public-key cryptography are only useful if parties know each other's public keys.
- That is, if Bob is not sure that the public key he uses to encrypt messages for Alice is actually her public key (meaning that she and only she knows the corresponding private key), then he cannot be sure his message will be secret from others.
- Similarly, if Bob is not sure if a particular key is Alice's public key, then he cannot trust that signed messages that properly verify with that key are actually hers.
- There are also issues of key compromise, revocation: Alice's private key may be compromised, for example if her computer is hacked, or may change for other reasons.

Public-Key Infrastructures

- Public-Key Infrastructures are intended to help solve these problems, by providing a means of communicating and authenticating users' public keys.
- **Idea:** Have someone you trust and whose public key you do know digitally sign others' public keys. This is called a certificate.
- But doesn't this just create the same problem? Yes, but in a way that might be somewhat easier to solve.
- Two main approaches proposed are the PGP Web of Trust and the hierarchical X.509 approach.

Components of a PKI

- Certification Authority (CA)
- Registration Authority (RA)
- Certificate Directory, e.g., LDAP
- Certificate Revocation List (CRL), contains revoked keys (not expired keys)
- OCSP-Responder (Online Certificate Status Protocol): okay, revoked, unknown

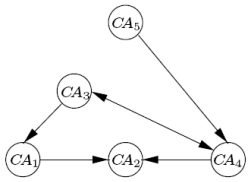
Public-Key Certificates

> Alice's public key is 0x147ABC724...
> Signed, Trent
> 21 March, 2003

- Usually includes additional information, such as:
  - whether the public key is for encryption or signatures
  - the cryptosystem to use, as well as any required parameters
  - an expiration date for the certificate
- Issues include:
  - How does Bob (or whoever wants to use Alice's public key) know Trent's key?
  - Is Trent trusted (by Bob) to issue such certificates? In issuing Alice's certificate, Trent must both verify Alice's identity and that Alice knows the private key corresponding to the claimed public key. Does Bob trust Trent to do this properly?
  - Agreed-upon formats for certificate, so data cannot be misinterpreted.

PGP (“Pretty Good Privacy”) — Web of Trust

(e) Directed graph (digraph) trust model



PGP — example of a distributed trust model, Source: Handbook of Cryptography

A posseses public key P_5 of CA_5, wishes to verify certificate of B signed by CA_3: directed path (CA_5, CA_4, CA_3)

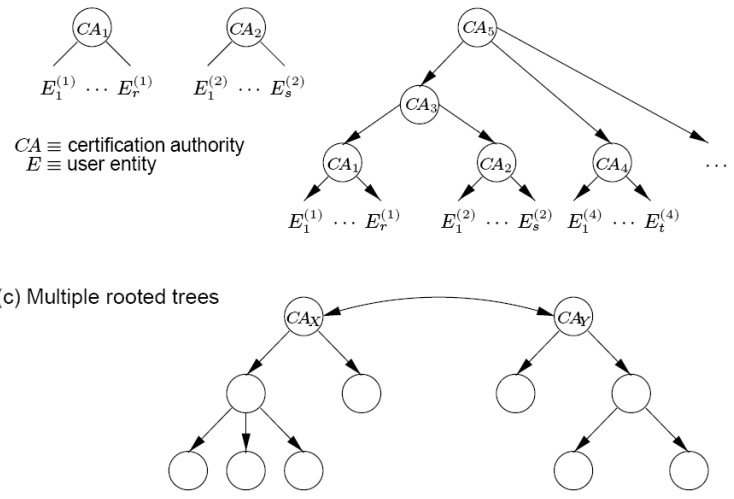
- Uses e-mail address as identity, Users keep known keys on a “key ring”. Each key on Bob’s key ring is stored signed (certified) by Bob’s key.
- Users certify keys of other users they know, i.e. act as “CA”, and potentially of the other users those users know, etc.
- Users decide which keys to trust based on simple scoring functions: untrusted, marginal, complete, ultimate

X.509 Hierarchical Approach

A strict X.509 hierarchy designates trusted certificate authorities (CAs). Trust is centralized in these entities.

(a) Separate domains

(b) Strict hierarchy



(c) Multiple rooted trees

Strict hierarchical trust model, Source: Handbook of Cryptography

X.509 Hierarchical Approach (2)

- Each CA is a certificate authority for users in a certain domain:
  - a company might act as a CA for its employees (Bosch-CA-DE, Bosch-CA1-DE).
- In turn, higher level CAs certify lower level CAs:
  - The government of a state might certify all businesses in the state
  - The US federal government might certify each state
  - Bosch-CA-DE certifies Bosch-CA1-DE
- Each user is assumed to know the public key of the CA's above it. The CA is assumed to be a reliable authority on the identities and keys of its constituents.
- This creates a tree of certifications. Some additional certifications are needed to make cross-domain interoperability possible. These special CAs act as Bridge-CAs.
- Often the certificate store contains not only the own root certificate but other certification keys, e.g., in Microsoft Store of Bosch Secude Program: Allianz Dresdner CA, Carl Zeiss AG CA, Deutsche Bank AG CA, etc.

X.509v3 certificates

```
Certificate:
   Data:
       Version: V3 (0x2)
       Serial Number: 0x43 3F BC 2F
       Signature Algorithm: sha1RSA
       Issuer: CN=Bosch-CA1-DE, CN=PKI, DC=Bosch, DC=com
       Validity
           Not Before: 31.05.2007 13:30 GMT
           Not After : 31.05.2010 14:00 GMT
       Subject: CN = Torsten.Schuetze, CN = O, CN = T, CN = DE, CN = BOSCH,
                CN = pki, DC = bosch, DC = com
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   30:81:89:02:81:81:00:aa:b6:68:2e:d1:9c:77:33:
                               ...
                   e8:35:1c:9e:27:52:7e:41:8f
               Exponent: 65537 (0x10001)
       Subject Alternative Name:
           RFC822 Name=Torsten.Schuetze@de.bosch.com
       Key Usage: digitalSignature keyEncipherment
       Extended Key Usage: Secure Email
   Signature Algorithm: sha1RSA
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
                           ...
       68:9f
```

Sample X.509v3 certificate (incomplete)

Procedure to Verify a Signature

Assume Bob wants to check Alice's signature. Alice's public key (certificate) is signed by Sub-CA which Bob doesn't trust directly, but Bob trusts Root-CA which signed certificate of Sub-CA.

1. Get certificate chain from Alice, check validity period of certificates
2. Bob doesn't trust issuer of Alice's certificate
3. Bob checks certificate of Alice with public key of Sub-CA from chain
4. Bob trusts issuer of Sub-CA certificate, checks with (trusted) public key of Root
5. Now Bob trusts Alice's public key and can verify her signature

Remark 1.1.

- CRLs are often large depending on PKI size and usage. Therefore Delta-CRLs are used which are updated more frequently and have smaller size.
- OCSP requires online connectivity but lower band-width
- Lower level certificates = shorter validity periods than upper level
- Three layer structure of SigG-CAs: Bundesnetzagentur on top, ZDAs (Zertifizierungs-diensteanbieter) in middle layer, user at lowest layer

Validity Models of PKI Certificates

Basically, three models:

1. shell model: At time of **signature verification** all certificates of chain have to be valid.
2. chain model: At time of signature generation the user certificate was valid. For every certificate in chain: At time of certificate generation the certificate of the issuer was valid.
3. hybrid model: At time of **signature generation** all certificates of chain have to be valid.

Bundesnetzagentur uses chain model:

- Successful verification over long periods of time
- Less revocations than in shell model
- Presentation: `http://www.bundesnetzagentur.de/media/archive/1343.pps`

Bundesnetzagentur uses RSA 2048 bit and SHA-512 for Root (since 2007-12-17), RSA 1024 bit and SHA-1 before

Public Key Infrastructure in Practice

- PKI hype 1998–2001, many of those PKI vendors are not longer in business
- Qualified digital signatures: 8 accredited ZDAs, but no commercial success yet
- Many successfully deployed company PKIs (Siemens, Deutsche Bank, Allianz, Bosch), cross-certification, only advanced level according to SigG
- Siemens PKI 1 (around 1999)
  - Only one key pair for authentication, encryption and signatures
  - Key backup
  - Private key on smart card and software PSE (personal security environment)
- Siemens PKI 2 (around 2003)
  - Different key pairs for encryption/authentication and signature
  - Key backup only for encryption key
  - Private signature key only on smart card
- Bosch PKI: only one key pair; key backup; no smart card, only software PSE

Current Practical Approaches

In current practice, both SSL and SSH make use of public-key certificates, even though there is not a full PKI deployed.

- SSL:
  - has several designated CA's whose public keys are built into most browers (such as Verisign's public key). Some browsers allow you to add additional CA's as well.
  - to be used, public keys must be certified by one of the CA's your browser knows about
  - current implementations use a one-sided approach in which merchants have certificates and customers do not. This allows users to ensure that (typically) their credit card information is going to the specified merchant, assuming they trust the CA and the browser software.

Current Practical Approaches, cont'd.

- SSH:
  - essentially takes the approach of writing down public keys the first time they are seen, and using them thereafter.
  - susceptible to man-in-the-middle attacks if they are done during the first exchange.
  - if keys don't match thereafter, gives warnings.
  - Unfortunately, users probably usually ignore these warnings, assuming that it is more likely that something strange has happened with the system than that an attack is underway. Even if this is mostly true, it creates potential opportunities for attackers.

2. Key Management

Sound key management is basis for secure system architecture

- efficient and secure key management is possible, even for symmetric keys
- symmetric key hierarchy widely deployed in military, banking, and mobile phone sector
- symmetric keys are often short-lived session keys (without backup)
- hierarchical key management: master key + derived keys (Key Derivation Function)

Key separation principle: avoid use of the same cryptographic key for multiple purposes

- **bad:** $K_1 = K_2 \in \mathcal{R}_n$ **equal keys**
- **better:** $K_1 = K \oplus v_1$, $K_2 = K \oplus v_2$ with $K \in \mathcal{R}_n$ master key, v_1, v_2 offsets **different keys, but invertible**
- **good:** $K_1 = KDF(K, v_1)$, $K_2 = KDF(K, v_2)$ with $K \in \mathcal{R}_n$ master key, v_1, v_2 offsets **different keys with non-invertible key derivation function**
- **best:** $K_1 \in \mathcal{R}_n$, $K_2 \in \mathcal{R}_n$ **independent keys**

Examples for Necessity of Key Separation

Example 1.2. CBC-MAC and CBC encryption

$\mathsf{Enc}_K(X \,||\, \mathsf{MAC}_{K'}(X))$ with $X = X_1 \ldots X_t$ blocks of plaintext
Enc_K — CBC encryption with key K and initial vector IV
$\mathsf{MAC}_{K'}$ — CBC-MAC with key K' and initial vector IV$'$

Let $K = K'$ and IV $=$ IV$'$, i. e., $C_0 = \mathsf{IV}$ and $C_i = \mathsf{Enc}_K(C_{i-1} \oplus X_i)$.
$\Longrightarrow C_t = \mathsf{Enc}_K(C_{t-1} \oplus X_t)$, last data block $X_{t+1} = C_t$
$\Longrightarrow$ final ciphertext block $C_{t+1} = \mathsf{Enc}_K(C_t \oplus X_{t+1}) = \mathsf{Enc}_K(C_t \oplus C_t) = \mathsf{Enc}_K(0)$!

MAC is independent of both plaintext and ciphertext!
$\Longrightarrow$
Never use the same key for both encryption and message authentication!

Examples for Necessity of Key Separation (2)

Example 1.3. Asymmetric keys for both signatures and challenge-response

Nicht überall, wo Authentifizierung „draufsteht“…

- Bob und Alice nutzen ein Challenge-Response-Verfahren für die Authentifizierung
- Alice sendet eine Challenge (z.B. Hash) an Bob
- Bob signiert die Challenge
- Alice kann anhand der Signatur feststellen, dass die Challenge von Bob stammt
- Der Hash wurde allerdings von folgender Nachricht gebildet…

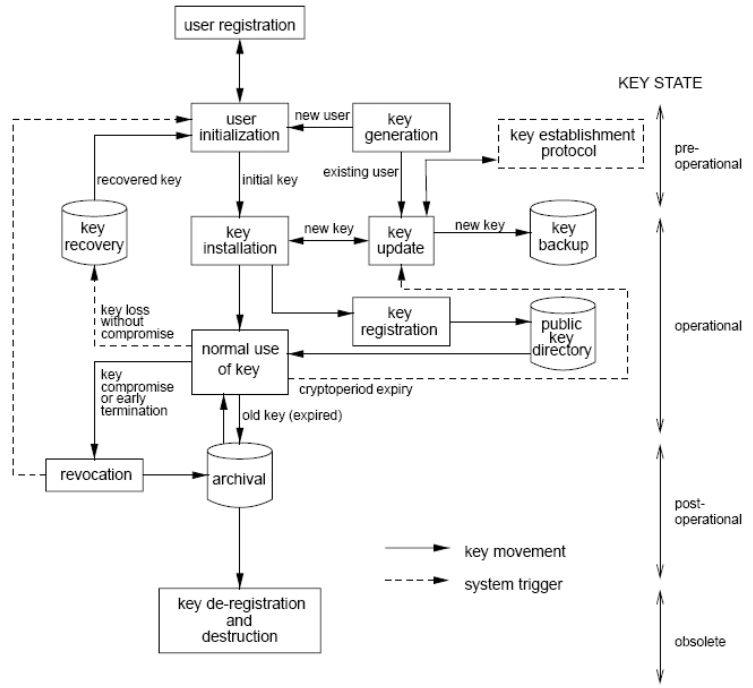
> Sehr geehrte Bank,
>
> Hiermit überweise ich 10.000 Euro an Alice, weil diese ein sehr lieber Mensch ist, und mich darum gebeten hat. Ich habe die Nachricht signiert, damit meine Identität und Intention zweifelsfrei feststeht.
>
> gez. Bob

…hier würde sich für Bob vermutlich die Verwendung von getrennten Schlüsselpaaren für Authentifizierung und Signatur anbieten…

Source: GWDG Workshop Grundlagen PKI

Never sign random messages with your signature key!

Key Management Life Cycle



Key Management Life Cycle, Source: Handbook

Part 3: Outline — Section 2: Elliptic Curve Cryptography

- Definition, group law
- Scalar multiplication, discrete log problem
- Advantages of ECC
- EC-domain parameters
- ECDSA

Elliptic Curve Cryptography

- Elliptic curves are no ellipses!
- Elliptic curves have a long history in mathematics, starting with Gauss to the Fermat proof by Wiles
- New generation of public key primitives
- Core mechanism based on arithmetic of elliptic curves over finite fields
- Proposed for cryptography by N. Koblitz [1985] and V. Miller [1987]
- ECC is already used in many government applications and applications with high security demands (ISDN encryption, Bonn-Berlin-Verbund, high security VPNs for inter-government communication)

Elliptic Curve Cryptography

Definition 2.1.

Let $p > 3$ be prime. The set of all points $(x, y) \in F_p \times F_p$ that solve

$$y^2 = x^3 + ax + b$$

in F_p with $a, b \in F_p$ and $\Delta = 4a^3 + 27b^2 \neq 0 \bmod p$, together with the *point at infinity* $\mathcal{O}$ is called elliptic curve E with parameters a, b over F_p.

Remark 2.2.

1. Instead of the finite field F_p one can use the fields F_{2^m} and F_{p^m}, $p > 3$, prime, $m > 1$ (different Weierstrass equation!). In ANSI Standard X9.62 [2005] and FIPS 186-3 [2009] (ECDSA) only F_p and F_{2^m} are used, whereas in ISO/IEC 15946-1,2,3 all fields F_p, F_{2^m}, and F_{p^m} are considered.
2. For constrained devices (Smart Cards) one often uses ECC over F_{2^m} (no need for an expensive long term arithmetic unit = arithmetic coprocessor).
3. $\Delta \neq 0$ ensures that the elliptic curve is non-singular, i. e., it has three distinct roots.

Group Law

Theorem 2.3.

The set of points of E form an additive Abelian group with neutral element $\mathcal{O}$ under the following operation: Let $P = (x_1, y_1), Q = (x_2, y_2) \in E; P, Q \neq \mathcal{O}$. If $x_2 = x_1$ and $y_2 = -y_1$, then $P + Q = \mathcal{O}$, otherwise $P + Q = (x_3, y_3)$, where

$$x_3 = \lambda^2 - x_1 - x_2, \qquad y_3 = \lambda(x_1 - x_3) - y_1$$

and

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q, \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q. \end{cases}$$

Furthermore, it holds $P + \mathcal{O} = \mathcal{O} + P = P$ for all $P \in E$.

Group Law (2)

- **Scalar Multiplication**

$$d \in \mathbb{N}: \qquad P = dQ = \underbrace{Q + \cdots + Q}_{d}$$

  in E can be viewed as analogon to exponentiation in $\mathbb{Z}/N$ within RSA.
- **Underlying hard mathematical problem = discrete log problem**

  Given points points $P, Q \in E$ with $P = dQ$, find factor d.

Group Law (3)

Remark 2.4.

- There are numerous other representations of elliptic curves. Above mentioned *affine representation* has the drawback that we need a special representation of $\mathcal{O}$ and that the addition formula requires lots of modular inversions.
- Often used in implementations:
  Projective coordinates

$$P = (x, y) \equiv (X, Y, Z), \quad x = \frac{X}{Z}, y = \frac{Y}{Z} \quad \text{for } Z \neq 0$$
$$\mathcal{O} \equiv (0, 1, 0)$$

- Efficient scalar multiplication is one of the key issues in ECC!

Advantages of ECC

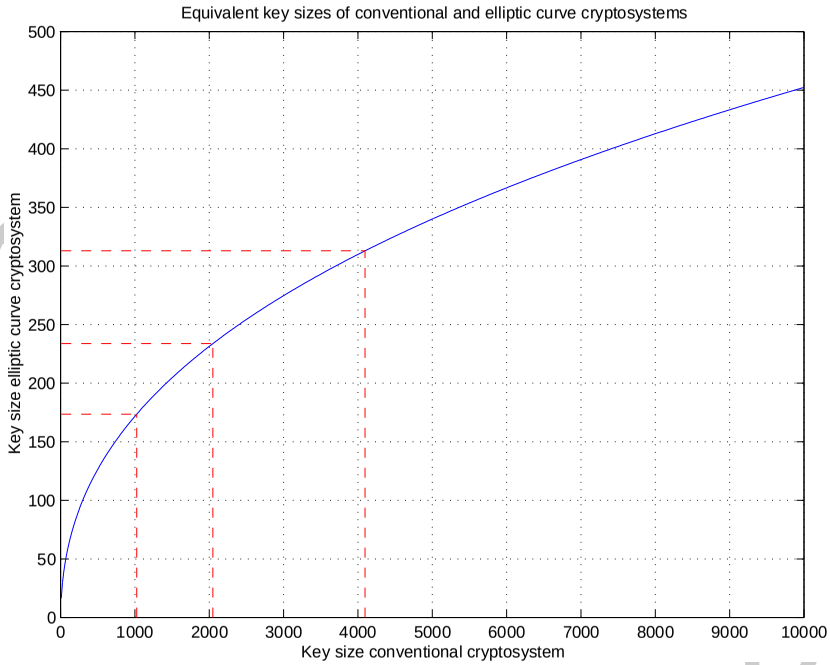
There is no known sub-exponential attack against ECC!
$\Longrightarrow$ Key sizes grow more slowly than with classical cryptosystems

$$n = \beta N^{\frac{1}{3}} (\log(N \log 2))^{\frac{2}{3}}$$

with $\beta \approx 4.91$, n – key size ECC, N – key size conventional method

$\Longrightarrow$ (Hyper?) Elliptic Curve Cryptography = Future of public key cryptography

Key Sizes



ECC vs. conventional key sizes for similar strength

Cryptography with Elliptic Curves

- Let F_p be a suitable finite field, $E(F_p)$ a suitable elliptic curve with parameters a, b.
- Let $G = (x_G, y_G) \in E(F_p)$ be a point with $G \neq \mathcal{O}$. The point G, called base point or generator, generates a cyclic subgroup of order $n = Ord(G)$.
- The parameters are chosen in such a way that n is prime.
- $\#E(F_p)$ — number of points in $E(F_p)$
- $h = \#E(F_p)/n$ — cofactor (index)
- The elements $\{p, a, b, G, n, h\}$ are called EC domain parameters. They will be used by all communicating participants.

ECDSA — Key Pair Generation

Given EC domain parameters $\{p, a, b, G, n, h\}$.
$\pi(P) = \pi(x_P, y_P) = x_p$

Algorithm Key Pair Generation

Every participant A generates a random number $d \in \mathbb{Z}$ with $0 < d < n$ and computes $Q = dG$. Private key of A = number d, public key = point Q.

ECDSA — Signature Generation

Algorithm Signature Generation

1. Compute $H(m)$ and convert the bit string to integer e.
2. Select random number $k \in \mathbb{Z}$ with $0 < k < n$, *ephemeral key*, and compute kG.
3. Compute $r = \pi(kG) \bmod n$. If $r = 0$, goto 2.
4. Compute $k^{-1} \bmod n$.
5. Compute $s = k^{-1}(e + dr) \bmod n$. If $s = 0$, goto 2.
6. Signature for message m is the pair $(r, s) \in F_n \times F_n$.

ECDSA — Signature Verification

Algorithm Signature Verification

1. If $r \notin [1, n-1]$ or $s \notin [1, n-1]$, deny signature as invalid.
2. Compute $H(m)$ and convert the bit string to integer e.
3. Compute $w = s^{-1} \bmod n$.
4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
5. Compute $X = u_1 G + u_2 Q$. If $X = \mathcal{O}$, deny signature as invalid.
6. Compute $\nu = \pi(X) \bmod n$.
7. Accept signature iff $\nu = r$.

ECDSA Signature Verification Works

Proof 2.5.

(r, s) valid signature $\Rightarrow s \equiv k^{-1}(e + dr) \bmod n$

$$k \equiv s^{-1}(e + dr) \equiv s^{-1}e + s^{-1}rd \equiv we + wrd \equiv u_1 + u_2 d \pmod{n}$$

Thus $X = u_1 G + u_2 Q = (u_1 + u_2 d)G = kQ$, and so $\nu = r$ as required. □

Remark 2.6.

- Verification is more expensive than signature generation.
- Quality of random number generation in Key Pair Generation and Signature Generation is crucial.
- Selection of suitable EC domain parameters is expert work. Some curves are standardized by NIST, ANSI and SEC.
- One has to implement two long integer arithmetics: operations in F_p and F_n.

Minimal key sizes for elliptic curves $E(F_p)$, see BSI

parameter / time frame	until 2009	until 2015
prime p	192	—
order n	180	224

ECC in Practice — Recommendations

- Projective coordinates are de-facto standard due to Side Channel Analysis
- Montgomery Ladder with projective coordinates using only (X,Z) is very efficient
  - resistant to Simple Power Analysis and Differential Power Analysis
  - only 10 multiplications for point addition, only 9 multiplications for point doubling
- Many patents, especially for $GF(2^m)$!
- BSI and German crypto industry prefers $GF(p)$
- Cryptographically strong elliptic curves are provided by ECC Brainpool (better than NIST and SECG curves)
  - `http://www.ecc-brainpool.org/download/Domain-parameters.pdf`
  - `http://tools.ietf.org/html/draft-lochter-pkix-brainpool-ecc-03`