CRYPTOGRAPHY LECTURE, 2022

Computer Science and Algorithmics, PWr

Mirosław Kutyłowski

Digital Signature Schemes

Signature

– with a private key SK one can create a signature of a message M

– with a public key PK one can verify the signature of M: result "valid" iff

  i SK has been used for signature creation

  ii PK corresponds to SK

  iii the message M is exactly the same as used for signature creation

Unforgeability

Forging Game between Challenger (C) and Adversary (A)

- (Keygen): C creates a key pair (pk, sk) at random
- (Learning): A sends messages $m_1, \ldots, m_n$, C replies with signatures

$$\sigma_i = \mathrm{Sign}(\mathrm{sk}, m_i)$$

- (Guessing): A outputs (m, σ)

A wins if $(m, \sigma) \neq (m_i, \sigma_i)$ and $\mathrm{Verify}(\mathrm{pk}, m, \sigma) = \mathrm{valid}$

A signature scheme is unforgeable if A wins with a negligible probability

One-time signature with one-way function F

take

$f_{0,1} = f(x_{0,1}),\ f_{1,1} = f(x_{1,1}),$

$f_{0,2} = f(x_{0,2}),\ f_{1,2} = f(x_{1,2}),$

$\ldots.$

$f_{0,n} = f(x_{0,n}),\ f_{1,n} = f(x_{1,n}),$

$\mathrm{PK} = (f_{0,1}, f_{1,1}, \ldots, f_{0,n}, f_{1,n})$

$\mathrm{SK} = (x_{0,1}, x_{1,1}, \ldots, x_{0,n}, x_{1,n})$

Signature for $[m_1 \ldots, m_n]$ is

$$x_{m_1,1}, x_{m_2,2}, \ldots, x_{m_n,n}$$

Verification: check $f(x_{m_j,j}) = f_{m_j,j}$

Signing long messages

Signature creation for message M:

step 1: $h := \mathrm{Hash}(M)$ for a collision-resistant hash function Hash

step 2: apply the core algorithm with secret key SK

so: the core algorithm gets input of a fixed size

RSA signatures

patented, patent expired in 2000

parameters like for RSA encryption

Signing M (PSS variant):

i. $h := \mathrm{Hash}(r \| M)$ for random r

ii. $s := h^d \bmod n$ for the secret d

iii. output r, s

another (weaker) variant (**RSASSA-PSS**): $h := \mathrm{Hash}(r \| \mathrm{Hash}(M))$

ElGamal signature

Setup: like for ElGamal encryption:

large prime p, computations in $\mathbb{Z}_p^*$ (multiplication modulo p), $g -$ generator

secret: $x < p - 1$ chosen at random,

public: $X = g^x \bmod p$

Signature creation for message M and x:

i. choose $k < p - 1$ at random, k must be coprime with $p - 1$

ii. $r := g^k \bmod p$

iii. $s := (\mathrm{Hash}(M) - x \cdot r) \cdot k^{-1} \bmod p - 1$

Verification: valid iff

$$g^{\mathrm{Hash}(M)} = X^r \cdot r^s$$

DSA Signature

US standard (DSS), optimization regarding the size:

Setup:

- p a prime number, $q | p-1,$ where q is also a large prime
- g - an element of order q
  - $\rightarrow$ choose a at random
  - $\rightarrow$ $g := a^{(p-1)/q}$, if $g \neq 1$ then g found
- secret key: $x < q$ chosen at random
- public key: $X = g^x \bmod p$

Remark: p, q can be used by many signers! (not like in the case of RSA)

DSA signature creation

i. choose $k < q$ at random

ii. $r := (g^k \bmod p) \bmod q$

iii. $s := (\mathrm{Hash}(M) - x \cdot r) \cdot k^{-1} \bmod q$

iv. output (r, s)

signature size: two numbers $< q$

Verification

i. check that $r, s < q$

ii. $w := s^{-1} \bmod q$

iii. $u_1 := \mathrm{Hash}(M) \cdot w, \quad u_2 := r \cdot w \bmod q$

iv. $v := (g^{u_1} \cdot X^{u_2} \bmod p) \bmod q$

v. valid if $r = v$

EC DSA

variant of DSA based on Elliptic Curves (additive group with hard DLP) instead of modular arithmetic:

- modular arithmetic requires bigger numbers as $\mathbb{Z}_p$ is a ring, more opportunities for computing discrete logarithm
- EC: complicated formulas, but nevertheless computational complexity lower

used e.g. on electronic identity cards, cryptographic signing cards, ...

Schnorr signature

patented, patent expired in 2008

Setup:

like for DSA, g- generator of a subgroup of order q where q is prime

- private key: $x < q$ chosen at random
- public key: $X = g^x \bmod p$

Signature creation:

i. $k < q$ chosen at random,

ii. $r := g^k \bmod p$

iii. $e := \mathrm{Hash}(M, r)$

iv. $s := k - e \cdot x \bmod q$

v. output (s, e)

Verification:

check iff $e = \mathrm{Hash}(M, g^s \cdot X^e)$

Schnorr signature – properties

- much simpler than DSA but blocked by the patent for many years
- never use the same k, otherwise we have a system of equations:

  $s_1 := k - e_1 \cdot x \bmod q$

  $s_2 := k - e_2 \cdot x \bmod q$

  with unknowns k, x (easy to solve)

Schnorr signature security

in ROM possibility to forge leads to breaking DLP:

- forgery $\mathcal{A}$ uses an oracle for a hash function
- r must be generated before there is a call to oracle with (M, r)
- run $\mathcal{A}$ so that a signature (s, e) is obtained
- "rewind" $\mathcal{A}$ to the moment when it made a call to the oracle, reprogram the oracle to get e'
- equations

  $s = k - e \cdot x \bmod q$

  $s' = k - e' \cdot x \bmod q$

(application of the famous "Forking Lemma")

EdDSA

like Schnorr signature with a major modification:

- instead of choosing k at random ...
- k calculated in a secret but deterministic way:

$$k := \mathrm{Hash}(x, M)$$

where x is a part of the secret key and M is the message to be signed

- some details due to the use of elliptic curves - Eduards curve

Result: do not worry about quality/safety of the random number generator!

Ring signature

public keys $X_1, \ldots, X_k$ correspond to a signature

- only one key x_i used for signature creation
- impossible to say which private key has been used
- used e.g in Monero cryptocurrency

Example: public keys X_1, X_2, X_3

signature creation with known x_2 :

i. choose $q_1, q_3, c_{1,} c_3$ at random, choose α at random

ii. $L_1 := g^{q_1} \cdot X_1^{c_1}$, $L_3 := g^{q_3} \cdot X_3^{c_3}$, $L_2 := g^{\alpha}$

iii. $c := \mathrm{Hash}(M, L_1, L_2, L_3)$

iv. $c_2 := c - c_1 - c_2 \bmod q$

v. $q_2 := \alpha - c_2 \cdot x_2$

output $(q_{1,} c_{1,} q_2, c_2, q_3, c_3)$

Verification: $c_1 + c_2 + c_3 = \mathrm{Hash}(M, L_1, L_2, L_3)$, where $L_i := g^{q_i} \cdot X_i^{c_i}$

Properties of Ring Signatures

information-theoretic security:

no matter what computational power has the adversary,

he cannot find out who from the ring created the signature

in contrast: **computational security** means:

it is infeasible to break . . . given available resources

Examples of use: signing a transaction in Monero:

- a ring signature for a transaction: signature over a new public transaction key (anonymous recipient can derive the new secret transaction key)
- impossible to say where the money coming from - from which ring member
- more effort needed: prevent using a ring signature twice (a tricky solution in monero, maybe we talk later ...)
- however: think about traffic analysis

Example: authentication with privacy protection:

Alice with key pair $(\mathrm{PK}, \mathrm{SK})$ authenticates herself

against Bob holding keys $(\mathrm{PK}', \mathrm{SK}')$

Unsafe solution:

i. Bob creates a challenge c (including time, ID's of Alice and Bob)

ii. Bob sends c to Alice

iii. Alice signs: SK: $s := \mathrm{Sign}_{\mathrm{SK}}(c)$

iv. Alice returns the signature to Bob

v. Bob verifiers the signature

PROBLEM: Bob can use s to prove that he has interacted with Alice

Improved authentication algorithm

Alice with key pair $(\mathrm{PK}, \mathrm{SK})$ authenticates herself

against Bob holding keys $(\mathrm{PK}', \mathrm{SK}')$

safe solution:

i. Bob creates a challenge c (including time, ID's of Alice and Bob)

ii. Bob sends c to Alice

iii. Alice creates a ring signature $s := \mathrm{Sign}_{\mathrm{SK},\mathrm{PK},\mathrm{PK}'}(c)$

iv. Alice returns the signature to Bob

v. Bob verifiers the signature s

Bob knows that s comes from Alice as he has not signed it.

Bob cannot prove Mallet that he has not created s

Forthcoming techniques:

"post-quantum" - resistant to potential attacks,

e.g. based on lattices