Signature schemes

Cryptology (1)

Martin Stanek

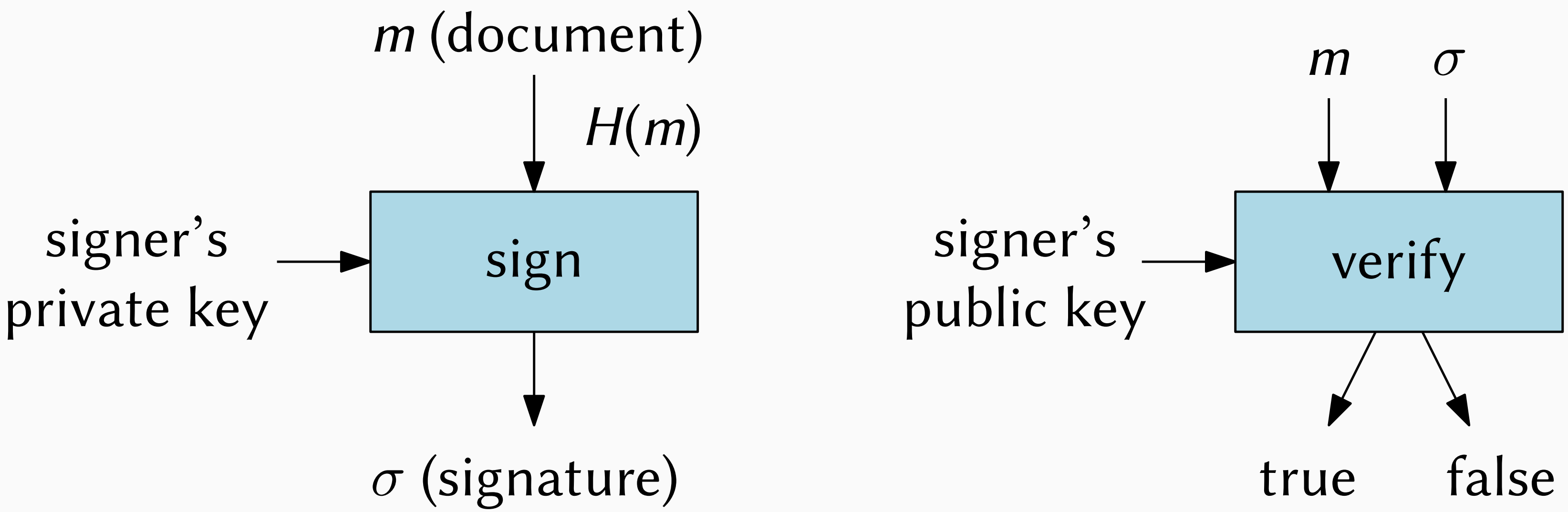
2025

KI FMFI UK Bratislava

Introduction

- *electronic signatures* in legislation vs. *digital signatures* in cryptology
- objectives:
  - authenticity and integrity of signed data
  - non-repudiation of origin
  - (usually) universal verifiability, i.e., anyone can verify the signature
  - unforgeability, efficiency, etc.
- objectives impossible to satisfy by a digital signature scheme alone
  - PKI, laws etc. (out of the scope of this lecture)

Digital signature scheme



- asymmetric construction
  - private key – signing
  - public key – verification

Digital signature scheme – definition

- digital signature scheme: $\langle \mathrm{Gen}, \mathrm{Sig}, \mathrm{Vrf} \rangle$
- Gen – PPT algorithm $\mapsto$ public and private key pair $(\mathsf{pk}, \mathsf{sk})$
- Sig – deterministic or probabilistic PT algorithm, creates a signature for a message m and sk: $\sigma = \mathrm{Sig}_{\mathsf{sk}}(m)$
- Vrf – usually deterministic PT algorithm;
  input: message, signature and signer's public key

$$\mathrm{Vrf}_{\mathsf{pk}}(m, \sigma) \in \{\mathrm{true}, \mathrm{false}\}$$

- correctness of the scheme:

$$\forall (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}\left(1^k\right) \forall m : \mathrm{Vrf}_{\mathsf{pk}}(m, \mathrm{Sig}_{\mathsf{sk}}(m)) = \mathrm{true}$$

Digital signature schemes – remarks

- schemes with appendix (the most common type) – this lecture
  - original document needed for the signature verification
- schemes with message recovery (rarely used)
  - verification produces from a signature the original message and some additional data to verify its correctness
- reasons for using hash function in digital signature schemes
  - shorter, fixed-length data for signing
  - preventing certain attacks, for example random message forgery (see later)
- using h.f. $\Rightarrow$ the security depends on h.f. properties, such as collision resistance

Security

- various possibilities; use the strongest definition
- idea similar to the security definition for MAC
- attacker has access to a public key
- EUF-CMA
  - CMA (chosen message attack) – the attacker has access to $\mathsf{Sig}_{\mathsf{sk}}(\cdot)$ oracle
  - EUF (existential unforgeability) – the attacker tries to create a message m (not previously queried) and a valid signature σ, such that $\mathsf{Vrf}_{\mathsf{pk}}(m, \sigma) = \text{true}$
- scheme is EUF-CMA secure if the success probability of any PPT attacker is negligible
- stronger definition SUF-CMA (strong existential unforgeability …)
  - as EUF-CME, but (m, σ') counts as a successful forgery if (m, σ'), as a pair, was not query and response previously

EUF-CMA definition

- signature scheme $\Pi = \langle \mathrm{Gen}, \mathrm{Sig}, \mathrm{Vrf} \rangle$
- PPT attacker A with access to $\mathrm{Sig}_{\mathsf{sk}}(\cdot)$ oracle
  - let Q be the set of all queries

$\textbf{Sig-forge}_{A,\Pi}(\boldsymbol{k})$:

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(1^k)$

$(m, \sigma) \leftarrow A^{\mathrm{Sig}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$

if $\mathrm{Vrf}_{\mathsf{pk}}(m, \sigma) = 1 \ \land \ m \notin Q$:

return 1

else: return 0

Definition. A signature scheme $\Pi = \langle \mathrm{Gen}, \mathrm{Sig}, \mathrm{Vrf} \rangle$ is EUF-CMA, if for any PPT attacker A there exists a negligible function $\mathrm{negl}(\cdot)$ such that:

$$\Pr\left[\text{Sig-forge}_{A,\Pi}(k) = 1\right] \leq \mathrm{negl}(k).$$

- more precisely, $(t(k), \varepsilon(k), q_S)$-secure scheme, if for any attacker running in time $t(k)$, and asking q_S oracle quaries

$$\Pr\left[\text{Sig-forge}_{A,\Pi}(k) = 1\right] \leq \varepsilon(k).$$

RSA signatures

- RSA instance/parameters as before:
  - public key: (e, n)
  - private key: d
  - all optimizations can be applied
- 1st attempt (without hashing):
  - Sig: $\sigma = m^d \bmod n$
  - Vrf: $\sigma^e \bmod n = m$?
  - correctness follows from the properties of RSA

Problems

- only for short messages
- random message forgery: for any $\sigma \in \mathbb{Z}_n$ $(\underbrace{\sigma^e \bmod n}_{m}, \sigma)$ is a valid pair
  - the attacker has no control over the message value
- another forgery (using homomorphic property of RSA): take two valid pairs (m_1, σ_1), (m_2, σ_2), and produce $(m_1 m_2 \bmod n, \sigma_1 \sigma_2 \bmod n)$

- 2nd attempt (with hashing):
  - Sig: $\sigma = H(m)^d \bmod n$
  - Vrf: $\sigma^e \bmod n = H(m)$?
- properties:
  - messages of arbitrary length
  - H is preimage resistant (infeasible to invert) $\Rightarrow$ prevents random message forgery
- H should be collision resistant

- FDH (Full Domain Hash) signature scheme using H with image $\mathbb{Z}_n$
  - EUF-CMA secure in random oracle model (for H), assuming the hardness of the RSA problem
- $H(m)$ usually shorter than $n \Rightarrow$ padding for randomization and (sometimes) provable security

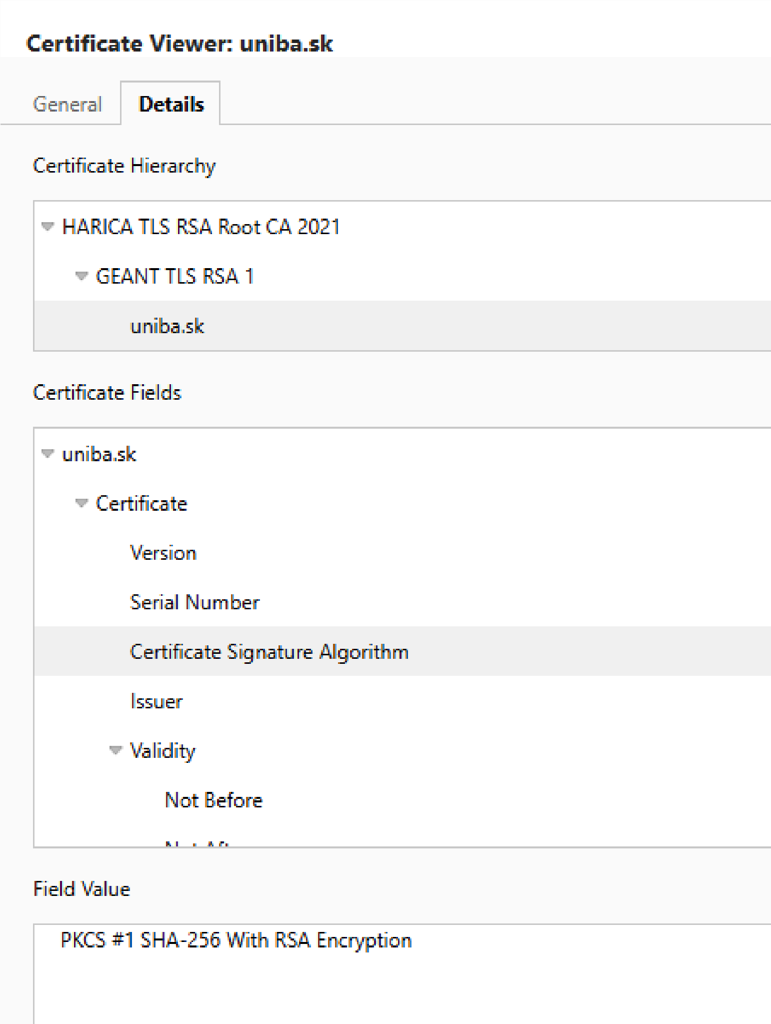
- construction standardized in 1998
- padded digest $H(m)$:

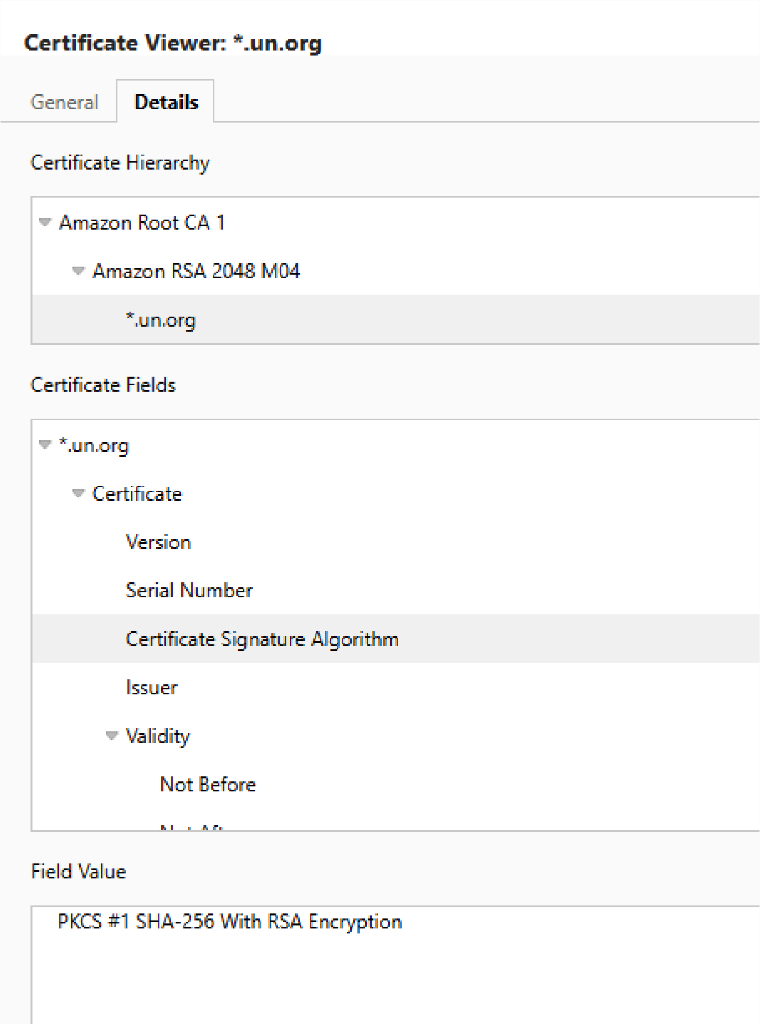
$$0x00 \parallel 0x01 \parallel 0xFF \parallel \ldots \parallel 0xFF \parallel 0x00 \parallel H(m)$$

"Moreover, while no attack is known against the EMSA-PKCS-v1_5 encoding method, a gradual transition to EMSA-PSS is recommended as a precaution against future developments." (RFC 8017)

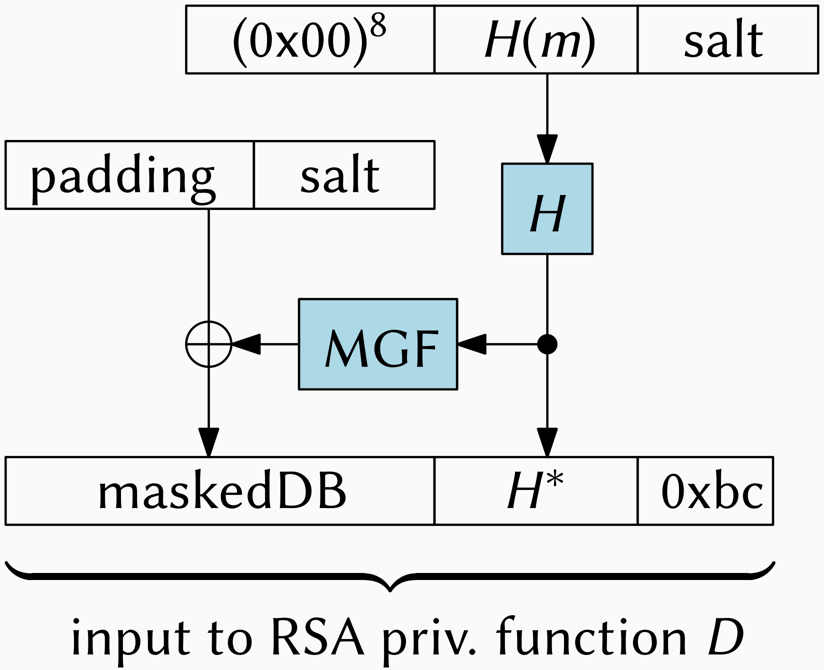
- frequently used in practice, e.g. X.509 certificates:
  - "sha256RSA" or "PKCS #1 SHA-256 With RSA Encryption" signature algorithm
- proof of PKCS #1 v1.5 security (2018), EUF-CMA in RO model under the standard RSA assumption

PKCS #1 v1.5 examples





- Probabilistic Signature Scheme, PKCS #1 v2.2 (RFC 8017)
- provable security in random oracle model
- salt – sequence of random bytes
- padding = 0x00 || ... || 0x00 || 0x01
- MGF – mask generation function (used in OAEP as well)



RSA-PSS – verification

$\mathsf{Vrf}_{\mathsf{pk}}(m, \sigma)$:

1. parse and verify: $\sigma^e \bmod n \mapsto \text{maskedDB} \parallel H^* \parallel \text{0xbc}$
2. $\text{DB} = \text{maskedDB} \oplus \text{MGF}(H^*)$
3. parse and verify: $\text{DB} \mapsto \text{padding} \parallel \text{salt}$
4. verify that $H^* = H\big((\text{0x00})^8 \parallel H(m) \parallel \text{salt}\big)$

the signature is correct if all verifications succeed

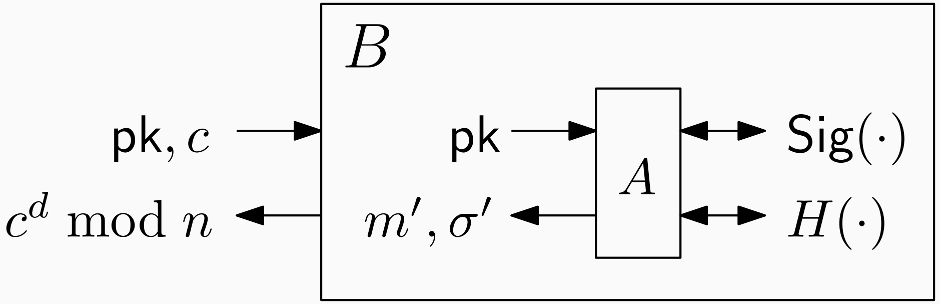
RSA-FDH security proof

RSA-FDH is EUF-CMA

- RSA problem (inverting RSA):
  Given $\mathsf{pk} = (n, e)$ and $c \in_R \mathbb{Z}_n$, find $x \in \mathbb{Z}_n$: $x^e \equiv c \pmod{n}$.
- RSA assumption:
  $\Pr\left[A(\mathrm{pk}, c) = \left(c^d \bmod n\right)\right]$ is negligible for any PPT algorithm A

Theorem. Let Π be a RSA-FDH signature scheme, H be a random oracle. Then Π is EUF-CMA if the RSA assumption holds.

- A is successful in Sig-forge with non-negligible probability ε_A
  - let (m', σ') be the output of A
- we construct B that inverts RSA with non-negligible probability ε_B
  - input: $\mathsf{pk} = (n, e)$, $c \in_R \mathbb{Z}_n$
  - output: $m \in \mathbb{Z}_n$: $m^e \equiv c \pmod{n}$
- B simulates A with pk
- B must correctly simulate both oracles



Reduction from Sig-forge $\mapsto$ RSA problem

Observations

- if A does not ask the oracle $H(m')$, then σ' is correct with negligible probability
- let q_H be the number of queries that A asked the H oracle
  - indirect queries through $\mathrm{Sig}_{\mathrm{sk}}$ simulation count as well
- B guesses, what query (we denote it r) A asks for $H(m')$
  - probability of correct guess: $1/q_H$

H simulation

- B remembers all queries and responses
- query $H(m_i)$, for $i \neq r$:
  1. $\sigma_i \in_R \mathbb{Z}_n$
  2. define $H(m_i) \leftarrow \sigma_i^e \bmod n$
  3. remember a triplet $(m_i, H(m_i), \sigma_i)$
- query $H(m_i)$, for $i = r$:
  1. define $H(m_i) \leftarrow c$
  2. remember a triplet $(m_i, H(m_i), \_)$
- in both cases is the output, $H(m_i)$, uniformly distributed in $\mathbb{Z}_n$
  - exactly like a random oracle

Remarks:

- we have to provide a consistent answers (if we already simulated that query)
- duplicate queries with m_r
  - asked later
  - if m_r was used previously
- if there is a successful A, then there is equally successful A^* not asking duplicate queries

Sig simulation

- $\mathrm{Sig}_{\mathrm{sk}}(m)$:
  1. if A has not queried $H(m)$ yet, we simulate H and get $(m_i, H(m_i), \sigma_i)$, where $m_i = m$
  2. if $m = m_r$, then B ends with a failure
  3. output: σ_i
- signatures are consistent with answers in H simulation
- if both conditions are satisfied:
  - A is successful, i.e., A outputs (m', σ'): $(\sigma')^e \bmod n = H(m')$, and
  - B guessed r correctly: $m' = m_r$,

  then $(\sigma')^e \bmod n = c$ and B solved RSA problem for c

RSA-FDH – conclusion

- if A is PPT, so is B
- B's probability of success $\varepsilon_B \approx \varepsilon_A / q_H$ (or $\varepsilon_A \approx \varepsilon_B \cdot q_H$)
  - ε_B is non-negligible, if ε_A is non-negligible
- example:
  - let $q_H = 2^{50}$ and we want 128-bit security, i.e., $\varepsilon_A \approx 2^{-128}$ $\Rightarrow$ $\varepsilon_B \approx 2^{-178}$
  - according RFC 3766 this corresponds to n with 6722 bits
- **tight reduction**: if $\varepsilon_B \approx \varepsilon_A$ (and time complexity as well)
  - for example RSA-PSS (Probabilistic Signature Scheme)
  - similarly to RSA-FDH in random oracle model
  - potential disadvantage: RNG is needed

ElGamal

ElGamal signature scheme

- T. ElGamal (1984)
- it is impossible to use ElGamal encryption scheme's algorithms for digital signatures
  - encryption is not a function (randomized … a good thing)
  - very few schemes offer bijections like RSA
  - specific signature scheme must be designed
- initialization identical to encryption scheme: $\mathsf{pk} = (p, g, y)$, $\mathsf{sk} = x$
  - $y = g^x \bmod p$
  - let g be a generator of $\left(\mathbb{Z}_p^*, \cdot\right)$
  - scheme can be "rephrased" in other groups
- $\mathrm{Sig}_{\mathsf{sk}}(m) = (r, s)$:
  1. $k \in_R \mathbb{Z}_{p-1}$ such that $\gcd(k, p-1) = 1$
  2. $r = g^k \bmod p$
  3. $s = (H(m) - xr) \cdot k^{-1} \bmod (p-1)$

- $\mathrm{Vrf}_{\mathsf{pk}}(m, (r, s))$: the signature is correct if

$$1 \leq r < p \quad \& \quad y^r \cdot r^s \equiv g^{H(m)} \pmod{p}$$

- correctness
  - the first part is trivial
  - the second part: $y^r \cdot r^s \equiv g^{xr} \cdot g^{ks} \equiv g^{xr+ks} \equiv g^{H(m)} \pmod{p}$
- efficiency
  - Sig – single modular exponentiation (can be precomputed)
  - Vrf – 3 modular exponentiations
  - signature's length ≈ a pair from $\mathbb{Z}_p \times \mathbb{Z}_{p-1}$

- computing x from y is a discrete logarithm problem
- predictable (leaked) k results in private key compromise:

$$s = (H(m) - xr) \cdot k^{-1} \bmod (p-1) \quad \Rightarrow \quad x = (H(m) - ks)r^{-1} \bmod (p-1)$$

- Bleichenbacher's attack (1996)
  - forging signatures if g has only small factors and $g \mid (p-1)$
  - for example, $g = 2$ is a bad choice
  - Remark: for discrete logarithm problem all generators are equivalent

ElGamal – security (2)

- test $1 \leq r < p$ is necessary; let us assume verification without the test:
  - let (r, s) be a signature for m: $g^{H(m)} \equiv y^r \cdot r^s \pmod{p}$
  - we compute a signature (r', s') for some $m' \neq m$

1. $u = H(m') \cdot H(m)^{-1} \bmod (p-1)$ (assuming that $H(m)$ is coprime to $p-1$)

$$g^{H(m')} \equiv g^{H(m)u} \equiv y^{ur} \cdot r^{us} \pmod{p}$$

2. we set $s' = us \bmod (p-1)$ and compute r' satisfying

$$r' \equiv ru \pmod{p-1}$$
$$r' \equiv r \pmod{p}$$

  - apply CRT; with overwhelming probability $r' \geq p$, otherwise $u = 1$ and we have a collision in H: $H(m') \equiv H(m) \pmod{p-1}$

- reusing k (for two distinct messages):
  - $m_1, (r, s_1) \quad \Rightarrow \quad H(m_1) \equiv xr + ks_1 \pmod{p-1}$
    $m_2, (r, s_2) \quad \Rightarrow \quad H(m_2) \equiv xr + ks_2 \pmod{p-1}$
  - we have $H(m_1) - H(m_2) \equiv k(s_1 - s_2) \pmod{p-1} \quad (\star)$
  - let $d = \gcd(s_1 - s_2, p-1)$
  - if $d = 1$ then $k = (H(m_1) - H(m_2))(s_1 - s_2)^{-1} \bmod (p-1)$
  - otherwise we divide the equation $(\star)$ by d, solve it mod $(p-1)/d$, and then test d candidates for k
  - knowing k we can easily find the private key x

ElGamal – security (4)

- random message forgery (when H is not used)
  1. $i, j \in_R \mathbb{Z}_{p-1}^*$
  2. $r = g^i \cdot y^j \bmod p$
  3. $s = -r \cdot j^{-1} \bmod (p-1)$
  4. $m = s \cdot i \bmod (p-1)$
- correctness:

$$\begin{aligned} y^r \cdot r^s &\equiv y^r \cdot g^{is} \cdot y^{js} \\ &\equiv y^r \cdot g^{is} \cdot y^{-jrj^{-1}} \\ &\equiv g^{is} \equiv g^m \pmod p \end{aligned}$$

ECDSA

Elliptic Curve Digital Signature Algorithm

- FIPS 186-5 (2023) – RSA, ECDSA, EdDSA
- usually with P-256 curve; Ethereum (curve secp256k1)
- 31% certificates use ECDSA (69% RSA), source: CT logs (November 2025)
- common parameters:
  - E – an elliptic curve
  - $G \in E$ – a subgroup generator of prime order n
- simplifying some details (conversions bitstring ↔ integer, etc.)

Initialization

- private key: $d \in_R \{1, \ldots, n-1\}$
- public key: $Q = dG$

$\mathbf{Sig_{sk}}(\boldsymbol{m})$:

1. $R = (r_x, r_y) = kG$,
   where $k \in_R \{1, \ldots, n-1\}$
2. $r = r_x \bmod n$
3. $s = k^{-1}(H(m) + rd) \bmod n$
   H is a suitable hash function
4. if $r = 0$ or $s = 0$ then start again with step 1 (unlikely)
5. signature $\sigma = (r, s)$

$\mathbf{Sig_{sk}}(\boldsymbol{m})$:

1. $R = (r_x, r_y) = kG$,
   where $k \in_R \{1, \ldots, n-1\}$
2. $r = r_x \bmod n$
3. $s = k^{-1}(H(m) + rd) \bmod n$
   H is a suitable hash function
4. if $r = 0$ or $s = 0$ then start again with step 1 (unlikely)
5. signature $\sigma = (r, s)$

$\mathbf{Vrf_{pk}}(\boldsymbol{m}, (\boldsymbol{r}, \boldsymbol{s}))$:

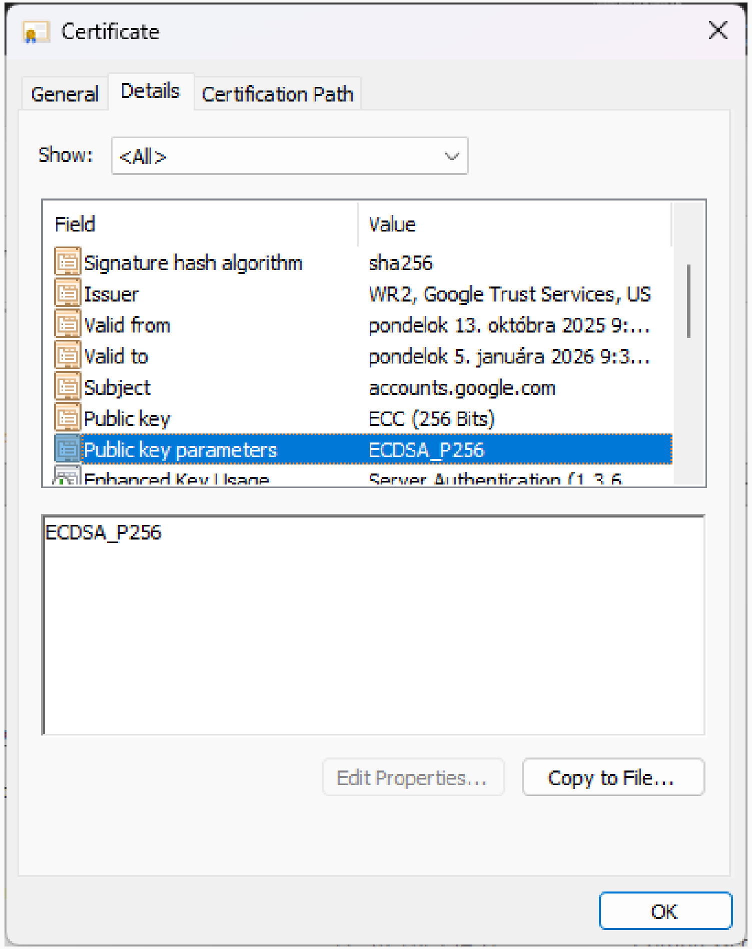
1. verify that $r, s \in \{1, \ldots, n-1\}$
2. $u = H(m) \cdot s^{-1} \bmod n$
   $v = r \cdot s^{-1} \bmod n$
3. $R' = (r'_x, r'_y) = uG + vQ$
4. verify that $r'_x \bmod n = r$

Correctness:

$$\begin{aligned} uG + vQ &= H(m) \cdot s^{-1} \cdot G + r \cdot s^{-1} \cdot d \cdot G \\ &= s^{-1} \cdot (H(m) + rd) \cdot G \\ &= kG = R \end{aligned}$$

ECDSA – remarks

- efficiency:
  - shorter signatures and faster then ElGamal's scheme
  - shorter keys and faster then DSA (the same scheme on modular group)
  - r can be precomputed
- H required to prevent random message forgery
- ECDSA: fixed curves and parameters are used
- EUF-CMA with DL assumption in ROM



- predictable/repeating k results in private key compromise
  - (2010) Sony PS3 ECDSA with constant k
    (2013) Android's Java SecureRandom with low entropy
  - deterministic variant can help
- various attacks when some/partial information about k is known
  - even less than 1 bit information
- malleable signatures: $(r, s) \mapsto (r, -s \bmod n)$
  - verification ($u' = H(m) \cdot (-s)^{-1} \bmod n = -u$; $v = r \cdot (-s)^{-1} \bmod n = -v$):

$$u'G + v'Q = -(uG + vQ) = -kG = \left(r_x, -r_y\right) \quad \Rightarrow \quad r_x \bmod n = r$$

  - Why does it matter? Bitcoin "transaction malleability" → Mt. Gox collapse (2014)

ECDSA – deterministic variant

- k is computed deterministically from $H(m)$ and the private key d
  - FIPS 186-5 prescribes a deterministic random bit generator based on HMAC
- protects again attack exploiting insufficient randomness in k

Schnorr

Schnorr signature scheme (on elliptic curve)

- simple construction, base for various other cryptographic schemes
- let us use the ECDSA-like parameters (any underlying group can be used):
  - E, $G \in E$, prime $n = \mathrm{ord}(G)$
  - private key: $d \in_R \{1, \dots, n-1\}$
  - public key: $Q = dG$
- hash function H with range $\mathbb{Z}_n$
- several slightly different variants

$\mathbf{Sig}_{\mathbf{sk}}(\boldsymbol{m})$:

1. $R = kG$, where $k \in_R \{1, \dots, n-1\}$
2. $r = H(R \parallel m)$
3. $s = k + rd \bmod n$
4. $\sigma = (r, s)$

$\mathbf{Vrf}_{\mathbf{pk}}(\boldsymbol{m}, (\boldsymbol{r}, \boldsymbol{s}))$: $H(sG - rQ \parallel m) = r$?

Correctness:
$sG - rQ = (k + rd)G - (rd)G = R$

- EUF-CMA in ROM under the discrete logarithm assumption
- again: k must be unpredictable; deterministic variant can be constructed as well
- we can change the first component from r to R (or its x-coordinate):

$\mathbf{Sig_{sk}}(\boldsymbol{m})$:

1. $R = kG$, where $k \in_R \{1, \ldots, n-1\}$
2. $r = H(R \parallel m)$
3. $s = k + rd \bmod n$
4. $\sigma = (R, s)$

$\mathbf{Vrf_{pk}}(\boldsymbol{m}, (\boldsymbol{R}, \boldsymbol{s}))$:

1. $r = H(R \parallel m)$
2. verify that $sG - rQ = R$

(*Bitcoin:* $r = H(R_x \parallel Q_x \parallel m)$)

- the verification equation is linear $\mapsto$ batch verification (verify multiple signatures)
  - $(m_i, (R_i, s_i))$ for public keys Q_i (for $i = 1, \ldots, t$)
  - verification: $(\sum_i a_i s_i) \cdot G - (\sum_i a_i r_i \cdot Q_i) = \sum_i a_i R_i$, for $a_i \in_R \{1, \ldots, n-1\}$

EdDSA

- EdDSA (RFC 8032)
  - deterministic variant of Schnorr signature scheme
  - included in the FIPS 186-5 (Ed448 and Ed25519)
- Ed25519: EdDSA with Curve25519 (in a different form) and SHA-512
  - optimized for speed and security
- Simplified EdDSA – parameters:
  - H – hash function with $2b$-bit output
  - G – point that generates a subgroup of prime order n

Keys

- b-bit string k
- compute $H(k) = \boldsymbol{h} = (h_0, \ldots, h_{2b-1})$
- left half of $\boldsymbol{h}$ is a scalar $a = \boldsymbol{h}[0 \ldots b-1]$
- $A = aG$
- private key: k (sometimes with A) or a with the right half $\boldsymbol{h}[b \ldots 2b-1]$
- public key: A

EdDSA – signing and verification

$\mathbf{Sig}_{\mathsf{sk}}(\boldsymbol{m})$:

1. $r = H(\boldsymbol{h}[b \ldots 2b-1] \parallel m)$
2. $R = rG$
3. $s = r + H(R \parallel A \parallel m) \cdot a \mod n$
4. signature: (R, s)

$\mathbf{Vrf}_{\mathsf{pk}}(\boldsymbol{m}, (\boldsymbol{R}, \boldsymbol{s}))$:

– verify that $sG = R + H(R \parallel A \parallel m) \cdot A$

Correctness:

$$sG = (r + H(R \parallel A \parallel m) \cdot a)G = rG + H(R \parallel A \parallel m) \cdot (aG) = R + H(R \parallel A \parallel m) \cdot A$$

Exercises

1. *What problems do we avoid by checking $r \neq 0$ and $s \neq 0$ in ECDSA scheme?*
2. *Show that random message forgery is possible in ECDSA variant without H.*
3. *Find what digital signature algorithms are supported by your (or some) SSH server. Compare the sizes of private and public keys for at least 3 algorithms.*
4. *Show that batch verification can be tricked to accept set of invalid signatures, if coefficients a_i are fixed (for example $a_i = i$).*
5. *Is EdDSA linear similarly to the previously discussed variant of Schnorr's scheme?*
6. *Double scalar multiplication of two points on elliptic curve: $aP + bQ$. Study and illustrate Shamir's trick for $a = 23$ and $b = 18$. Compare with a straightforward computation of aP and bQ using double-and-add method, and adding the results.*