

Digital signature schemes

Martin Stanek

Department of Computer Science
Comenius University
`stanek@dcs.fmph.uniba.sk`

Cryptology 1 (2021/22)

Content

Introduction

- digital signature scheme
- security of digital signatures

RSA

- textbook version
- padding: PKCS #1, PSS

ElGamal

Digital Signature Algorithm – DSA and ECDSA

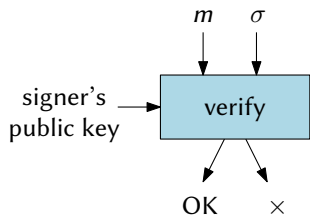
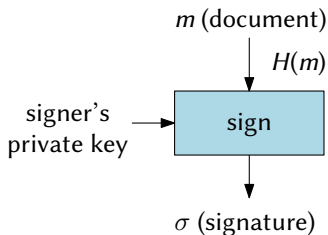
Schnorr

EdDSA

Introduction

- ▶ “electronic signatures” in legislation
- ▶ “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 (1)



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

Digital signature scheme (2)

- ▶ digital signature scheme: $(\text{Gen}, \text{Sig}, \text{Vrf})$
- ▶ Gen – PPT algorithm, produces public and private key pair (pk, sk)
- ▶ Sig – PPT algorithm, produces a signature from a message and signer's private key: $\sigma = \text{Sig}_{sk}(m)$
- ▶ Vrf – usually deterministic PT algorithm; input: message, signature and signer's public key; $\text{Vrf}_{pk}(m, \sigma) \in \{\text{true/OK}, \text{false}/\times\}$
- ▶ correctness of the scheme:

$$\forall (pk, sk) \leftarrow \text{Gen}(1^k) \quad \forall m : \text{Vrf}_{pk}(m, \text{Sig}_{sk}(m)) = \text{true}$$

Digital signature schemes – remarks

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

Security of digital signatures

- ▶ various possibilities; as usual: use the strongest definition
- ▶ idea similar to MAC security
- ▶ attacker has access to a public key
- ▶ EUF-CMA
 - ▶ CMA (chosen message attack) – the attacker has access to $\text{Sig}_{\text{sk}}(\cdot)$ oracle
 - ▶ EUF (existential unforgeability) – the attacker tries to create a message m (not previously queried) and a valid signature σ , i.e. $\text{Vrf}_{\text{pk}}(m, \sigma) = \text{true}$
- ▶ scheme is EUF-CMA secure if the success probability of any PPT attacker is negligible

RSA signature scheme

- ▶ 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: $(\underbrace{\sigma^e \bmod n}_m, \sigma)$ for $\sigma \in \mathbb{Z}_n$
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 $(m_1 m_2 \bmod n, \sigma_1 \sigma_2 \bmod n)$

RSA signature scheme – standard “textbook” version

- ▶ 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

PKCS #1 v1.5

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

$$0x00 \parallel 0x01 \parallel 0xff \parallel \dots \parallel 0xff \parallel 0x00 \parallel H(m)$$

“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, 2016)

- ▶ frequently used in practice, e.g. X.509 certificates:
 - ▶ “sha1RSA” or “PKCS #1 sha1 with RSA encryption” signature algorithm
...SHA1 problems – deprecation of SHA1 certificates in browsers
 - ▶ “sha256RSA” or “PKCS #1 sha256 with RSA encryption” signature algorithm
- ▶ proof of PKCS #1 security (2018), EUF-CMA in RO model under the standard RSA assumption

PKCS #1 v1.5 examples

Certificate

GeneralDetailsCertification Path

Show: <All>

Field	Value
Signature algorithm	sha384RSA
Signature hash algorithm	sha384
Issuer	GEANT OV RSA CA 4, GEANT Vere...
Valid from	9. júna 2020 1:00:00
Valid to	10. júna 2022 0:59:59
Subject	uniba.sk, CIT, Univerzita Komenského...
Public key	RSA (2048 Bits)
Public key parameters	05 00
Authority Key Identifier	KevID=6f1d3549106c32fa59a09ebc...

sha384RSA

Edit Properties...Copy to File...

OK

Certificate

GeneralDetailsCertification Path

Show: <All>

Field	Value
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	Microsoft RSA TLS CA 02, Microsoft C...
Valid from	30. septembra 2021 2:43:38
Valid to	30. marca 2022 2:43:38
Subject	www.bing.com
Public key	RSA (2048 Bits)
Public key parameters	05 00
SCT List	v1. 2979hef09e393921f056739f63a5...

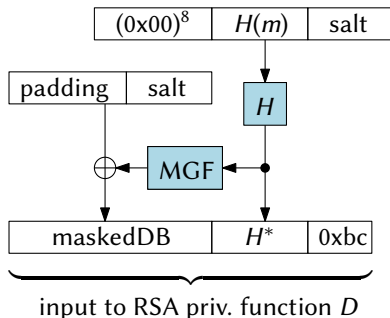
sha256RSA

Edit Properties...Copy to File...

OK

RSA-PSS

- ▶ Probabilistic Signature Scheme, PKCS #1 v2.2 (RFC 8017)
- ▶ provable security in random oracle model



- ▶ salt – sequence of random bytes
- ▶ padding = $0x00 \parallel \dots \parallel 0x00 \parallel 0x01$
- ▶ MGF – mask generation function (used in OAEP as well)

RSA-PSS – verify

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

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

the signature is correct if all verifications succeed

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: $pk = (p, g, y)$, $sk = x$
 - ▶ $y = g^x \bmod p$
 - ▶ let g be a generator of (\mathbb{Z}_p^*, \cdot)
 - ▶ scheme can be “rephrased” in other groups
- ▶ $\text{Sig}_{sk}(m) = (r, s)$:
 1. $k \xleftarrow{\$} \mathbb{Z}_{p-1}$ such that $\gcd(k, p-1) = 1$
 2. $r = g^k \bmod p$
 3. $s = (H(m) - xr) \cdot k^{-1} \bmod (p-1)$

ElGamal signature scheme – verification and correctness

- ▶ $\text{Vrf}_{\text{pk}}(m, (r, s))$: 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 \sim a pair from $\mathbb{Z}_p \times \mathbb{Z}_p$

ElGamal – security (1)

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

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

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

ElGamal – security (1)

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

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

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

ElGamal – security (2)

- ▶ Bleichenbacher's attack (1996)
 - ▶ forging signatures if g has only small factors and $g \mid (p - 1)$
 - ▶ e.g. $g = 2$ is a bad choice
 - ▶ Remark: for discrete logarithm problem all generators are equivalent
- ▶ reusing k (for two distinct messages):
 - ▶ $m_1, (r, s_1) \Rightarrow H(m_1) \equiv xr + ks_1 \pmod{p-1}$
 - ▶ $m_2, (r, s_2) \Rightarrow H(m_2) \equiv xr + ks_2 \pmod{p-1}$
 - ▶ 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 $k = (H(m_1) - H(m_2))(s_1 - s_2)^{-1} \pmod{p-1}$
 - ▶ otherwise we divide the equation (\star) by d , solve it mod $(p-1)/d$, and then test d candidates for k
 - ▶ having k we can easily find the private key x

ElGamal – security (2)

- ▶ Bleichenbacher's attack (1996)
 - ▶ forging signatures if g has only small factors and $g \mid (p-1)$
 - ▶ e.g. $g = 2$ is a bad choice
 - ▶ Remark: for discrete logarithm problem all generators are equivalent
- ▶ reusing k (for two distinct messages):
 - ▶ $m_1, (r, s_1) \Rightarrow H(m_1) \equiv xr + ks_1 \pmod{p-1}$
 $m_2, (r, s_2) \Rightarrow H(m_2) \equiv xr + ks_2 \pmod{p-1}$
 - ▶ 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 $k = (H(m_1) - H(m_2))(s_1 - s_2)^{-1} \pmod{p-1}$
 - ▶ otherwise we divide the equation (\star) by d , solve it mod $(p-1)/d$, and then test d candidates for k
 - ▶ having k we can easily find the private key x

ElGamal – security (3)

- ▶ random message forgery (when H is not used)

1. $i, j \xleftarrow{\$} \mathbb{Z}_{p-1}^*$

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:

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

Digital Signature Algorithm (DSA)

- ▶ part of FIPS 186-4 (DSA, RSA, ECDSA)
 - ▶ sporadic use in practice
 - ▶ other alternatives available, e.g. ECDSA is faster, with shorter keys
 - ▶ removal proposed in FIPS 186-5 (still draft, October 2019)
- ▶ initialization:
 1. generate primes p, q (e.g. $|p| = 2048, |q| = 256$) such that $q \mid (p - 1)$
 2. generate $h \xleftarrow{\$} \mathbb{Z}_{p-1}$ such that $g = h^{(p-1)/q} > 1$
 g is a subgroup generator (order q)
public domain parameters: (p, q, g)
 3. private key: $x \xleftarrow{\$} \mathbb{Z}_q^*$
 4. public key: $y = g^x \bmod p$

DSA – signing and verification

► $\text{Sig}_{\text{sk}}(m)$:

1. $r = g^k \bmod p \bmod q$, where $k \xleftarrow{\$} \mathbb{Z}_q^*$
2. $s = k^{-1}(H(m) + xr) \bmod q$
3. if $r = 0$ or $s = 0$ start again with step 1 (extremely unlikely)
4. $\sigma = (r, s)$

► $\text{Vrf}_{\text{pk}}(m, (r, s))$:

1. verify that $r, s \in \mathbb{Z}_q^*$
2. $u_1 = H(m) \cdot s^{-1} \bmod q$
 $u_2 = r \cdot s^{-1} \bmod q$
3. verify that $(g^{u_1} \cdot y^{u_2} \bmod p) \bmod q = r$

► correctness:

$$\begin{aligned} (g^{u_1} \cdot y^{u_2} \bmod p) \bmod q &= g^{H(m)s^{-1} + xrs^{-1}} \bmod p \bmod q \\ &= g^{s^{-1}(H(m) + xr)} \bmod p \bmod q = g^k \bmod p \bmod q = r \end{aligned}$$

DSA – signing and verification

► $\text{Sig}_{\text{sk}}(m)$:

1. $r = g^k \bmod p \bmod q$, where $k \xleftarrow{\$} \mathbb{Z}_q^*$
2. $s = k^{-1}(H(m) + xr) \bmod q$
3. if $r = 0$ or $s = 0$ start again with step 1 (extremely unlikely)
4. $\sigma = (r, s)$

► $\text{Vrf}_{\text{pk}}(m, (r, s))$:

1. verify that $r, s \in \mathbb{Z}_q^*$
2. $u_1 = H(m) \cdot s^{-1} \bmod q$
 $u_2 = r \cdot s^{-1} \bmod q$
3. verify that $(g^{u_1} \cdot y^{u_2} \bmod p) \bmod q = r$

► correctness:

$$\begin{aligned}(g^{u_1} \cdot y^{u_2} \bmod p) \bmod q &= g^{H(m)s^{-1} + xrs^{-1}} \bmod p \bmod q \\ &= g^{s^{-1}(H(m) + xr)} \bmod p \bmod q = g^k \bmod p \bmod q = r\end{aligned}$$

DSA – signing and verification

► $\text{Sig}_{\text{sk}}(m)$:

1. $r = g^k \bmod p \bmod q$, where $k \xleftarrow{\$} \mathbb{Z}_q^*$
2. $s = k^{-1}(H(m) + xr) \bmod q$
3. if $r = 0$ or $s = 0$ start again with step 1 (extremely unlikely)
4. $\sigma = (r, s)$

► $\text{Vrf}_{\text{pk}}(m, (r, s))$:

1. verify that $r, s \in \mathbb{Z}_q^*$
2. $u_1 = H(m) \cdot s^{-1} \bmod q$
 $u_2 = r \cdot s^{-1} \bmod q$
3. verify that $(g^{u_1} \cdot y^{u_2} \bmod p) \bmod q = r$

► correctness:

$$\begin{aligned}(g^{u_1} \cdot y^{u_2} \bmod p) \bmod q &= g^{H(m)s^{-1} + xrs^{-1}} \bmod p \bmod q \\ &= g^{s^{-1}(H(m) + xr)} \bmod p \bmod q = g^k \bmod p \bmod q = r\end{aligned}$$

DSA – remarks

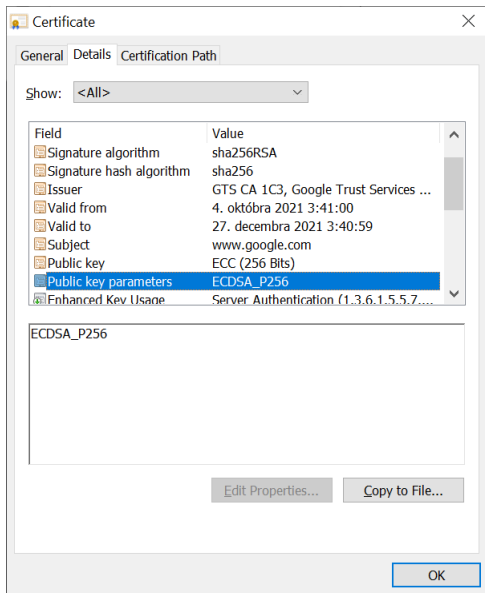
- ▶ if $r = 0$ then the signature does not depend on x
- ▶ if $s = 0$ then $s^{-1} \bmod q$ does not exist
- ▶ efficiency: shorter signatures (comparing to ElGamal's), faster than ElGamal (shorter exponents), r can be precomputed
- ▶ H required to prevent random message forgery (try yourself)
- ▶ the parameters p , q and g can be shared
 - ▶ rare for DSA; ensure that parameters are not maliciously prepared
 - ▶ verifiable procedure for generating the parameters (part of the standard)
 - ▶ ECDSA: fixed curves and parameters are used (approved)

ECDSA

- ▶ point G – generator of subgroup of prime order n
- ▶ private key: $d \xleftarrow{\$} \mathbb{Z}_n^*$; public key: $Q = dG$ and domain parameters
- ▶ simplifying some details (e.g. conversions bitstring \leftrightarrow integer)
- ▶ $\text{Sig}_{\text{sk}}(m)$:
 1. $(x, y) = kG$, where $k \xleftarrow{\$} \mathbb{Z}_n^*$
 2. $r = x \bmod n$
 3. $s = k^{-1}(H(m) + dr) \bmod n$
 4. if $r = 0$ or $s = 0$ start again with step 1 (extremely unlikely)
 5. $\sigma = (r, s)$
- ▶ $\text{Vrf}_{\text{pk}}(m, (r, s))$:
 1. verify that $r, s \in \mathbb{Z}_n^*$
 2. $u_1 = H(m) \cdot s^{-1} \bmod n$
 $u_2 = r \cdot s^{-1} \bmod n$
 3. $(x, y) = X = u_1G + u_2Q$ (if $X = 0$ reject)
 4. accept iff $x \bmod n = r$
- ▶ correctness – obvious

ECDSA – remarks

- ▶ shorted keys and faster (comparing to DSA)
- ▶ somewhat popular on web (usually with P-256 curve)
- ▶ part of FIPS 186-5 draft
- ▶ used with Bitcoin (curve Secp256k1)



DSA/ECDSA – problems with k

- ▶ predictable/repeating k results in private key compromise
 - ▶ sometimes it hurts: (2010) Sony PS3 ECDSA with constant k , (2013) Android's Java SecureRandom with low entropy
 - ▶ variant with deterministic (EC)DSA proposed in RFC 6979
- ▶ various attacks when some additional information about k is known, for example (Faugère et al., 2012):
 - ▶ assumption: known messages and signatures, such that k -values share some common bits (the bits themselves are unknown to the attacker)
 - ▶ works for both DSA and ECDSA
 - ▶ concrete results for 160-bit q :
 - 100 signed messages with shared 4 LSBs ... 100% probability of success
 - 200 signed messages with shared 3 LSBs ... 100% probability of success
 - 400 signed messages with shared 1 LSBs ... 90% probability of success

Schnorr signature scheme

- ▶ simple construction, base for other schemes
- ▶ let us use the DSA's parameters (any underlying group can be used):
 - ▶ public domain parameters: (p, q, g)
 - ▶ g is a generator of some subgroup (order q)
 - ▶ private key: $x \xleftarrow{\$} \mathbb{Z}_q^*$
 - ▶ public key: $y = g^x \bmod p$
- ▶ hash function H with image \mathbb{Z}_q
- ▶ $\text{Sig}_{\text{sk}}(m)$:
 1. $r = H(m || g^k \bmod p)$, where $k \xleftarrow{\$} \mathbb{Z}_q^*$
 2. $s = k + xr \bmod q$
 3. $\sigma = (r, s)$
- ▶ $\text{Vrf}_{\text{pk}}(m, (r, s)) = \text{true} \Leftrightarrow H(m || g^s \cdot y^{-r} \bmod p) = r$
- ▶ correctness: $g^s \cdot y^{-r} \equiv g^{k+xr} \cdot g^{-xr} \equiv g^k \pmod{p}$

Schnorr signature scheme

- ▶ simple construction, base for other schemes
- ▶ let us use the DSA's parameters (any underlying group can be used):
 - ▶ public domain parameters: (p, q, g)
 - ▶ g is a generator of some subgroup (order q)
 - ▶ private key: $x \xleftarrow{\$} \mathbb{Z}_q^*$
 - ▶ public key: $y = g^x \bmod p$
- ▶ hash function H with image \mathbb{Z}_q
- ▶ $\text{Sig}_{\text{sk}}(m)$:
 1. $r = H(m || g^k \bmod p)$, where $k \xleftarrow{\$} \mathbb{Z}_q^*$
 2. $s = k + xr \bmod q$
 3. $\sigma = (r, s)$
- ▶ $\text{Vrf}_{\text{pk}}(m, (r, s)) = \text{true} \Leftrightarrow H(m || g^s \cdot y^{-r} \bmod p) = r$
- ▶ correctness: $g^s \cdot y^{-r} \equiv g^{k+xr} \cdot g^{-xr} \equiv g^k \pmod{p}$

Schnorr signature scheme – security

- ▶ EUF-CMA in ROM under the discrete logarithm assumption
- ▶ again: k must be unpredictable

EdDSA

- ▶ EdDSA (Edwards Curve Digital Signature Algorithm, RFC 8032)
 - ▶ deterministic variant of Schnorr signature scheme
 - ▶ included in the draft of 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
 - ▶ B – point on elliptic curve, that generates a subgroup of prime order l
- ▶ Keys:
 - ▶ b -bit string k
 - ▶ compute $H(k) = \mathbf{h} = (h_0, \dots, h_{2b-1})$
 - ▶ *remark:* for Ed25519 set $h_0 = h_1 = h_2 = 0$, $h_{b-2} = 1$, and $h_{b-1} = 0$
 - ▶ left half of \mathbf{h} is a scalar $a = \mathbf{h}[0 \dots b-1]$
 - ▶ $A = aB$
 - ▶ private key: k (sometimes with A) or a with the right half $\mathbf{h}[b \dots 2b-1]$
 - ▶ public key: A

EdDSA – signing and verification

- ▶ Signing a message m :

1. $r = H(\mathbf{h}[b \dots 2b - 1], m)$
2. $R = rB$
3. $s = r + H(R, A, m) \cdot a \bmod l$
4. signature: (R, s)

- ▶ Verification of (R, s) for m :

- ▶ check if $sB = R + H(R, A, m) \cdot A$
- ▶ correctness:

$$sB = (r + H(R, A, m) \cdot a)B = rB + H(R, A, m) \cdot (aB) = R + H(R, A, m) \cdot A$$