Modern Cryptography: Lecture 13

Digital Signatures

Daniel Slamanig

AUSTRIAN INSTITUTE OF TECHNOLOGY

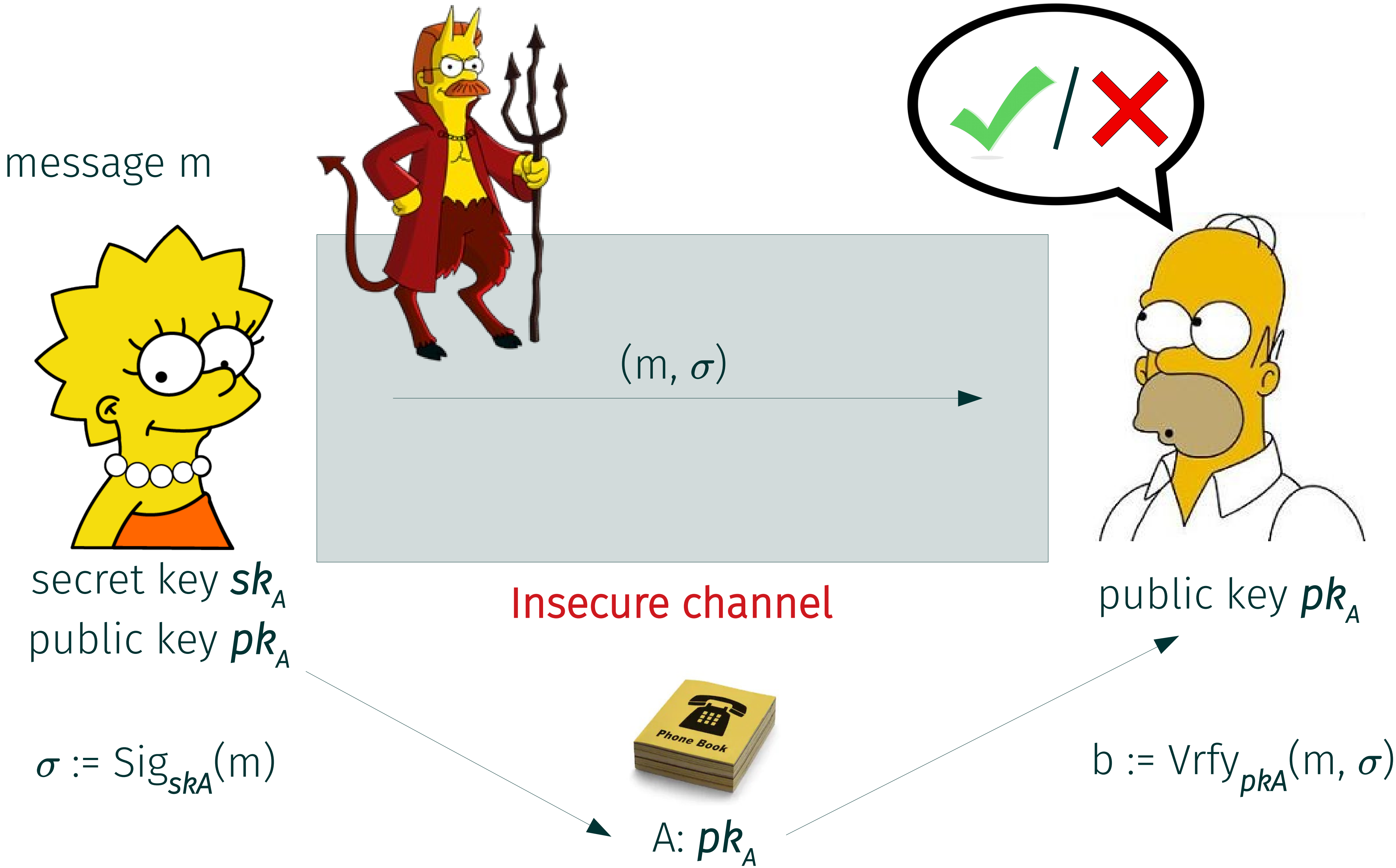
Organizational

- Where to find the slides and homework?
  - https://danielslamanig.info/ModernCrypto18.html
- How to contact me?
  - daniel.slamanig@ait.ac.at
- Tutor: Karen Klein
  - karen.klein@ist.ac.at

- Official page at TU, Location etc.
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?dswid=8632&dsrid=679&courseNr=192062&semester=2018W
- Tutorial, TU site
  - https://tiss.tuwien.ac.at/course/courseAnnouncement.xhtml?dswid=5209&dsrid=341&courseNumber=192063&courseSemester=2018W

- Exam for the second part: Thursday 31.01.2019 15:00-17:00 (Tutorial slot)

Overview Digital Signatures



Digital Signatures: Intuitive Properties

Can be seen as the public-key analogue of MACs with <u>public verifiability</u>

- Integrity protection: Any modification of a signed message can be detected
- Source authenticity: The sender of a signed message can be identified
- Non-repudiation: The signer cannot deny having signed (sent) a message

<u>Security (intuition):</u> should be hard to come up with a signature for a message that has not been signed by the holder of the private key

Digital Signatures: Applications

Digital signatures have many applications and are at the heart of implementing public-key cryptography in practice

- Issuing certificates by CAs (Public Key Infrastructures): binding of identities to public keys
- Building authenticated channels: authenticate parties (servers) in security protocols (e.g., TLS) or secure messaging (WhatsApp, Signal, ...)
- Code signing: authenticate software/firmware (updates)
- Sign documents (e.g., contracts): Legal regulations define when digital signatures are equivalent to handwritten signatures
- Sign transactions: used in the cryptocurrency realm
- etc.

Digital Signatures: Definition

DEFINITION 12.1 A (digital) signature scheme is a triple of PPT algorithms (Gen, Sig, Vrfy) such that:

1. The **key-generation** algorithm **Gen** takes as input the security parameter 1^n and outputs a pair of keys (pk, sk) (we assume that pk and sk have length n and that n can be inferred from pk or sk).
2. The **signing** algorithm **Sig** takes as input a private key sk and a message m from some message space $\mathcal{M}$. It outputs a signature σ, and we write this as $\sigma \leftarrow \mathrm{Sig}_{sk}(m)$.
3. The deterministic **verification** algorithm **Vrfy** takes as input a public key Pk, a message m, and a signature σ. It outputs a bit b with b=1 meaning valid and b=0 meaning invalid. We write this as $b := \mathrm{Vrfy}_{pk}(m, \sigma)$.

It is required that, except possibly with negligible probability over $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$, we have

$$\mathrm{Vrfy}_{pk}(m, \mathrm{Sig}_{sk}(m)) = 1$$

for any message $m \in \mathcal{M}$.

Some Remarks on the Definition

- The signing algorithm
  - may be deterministic or probabilistic
  - may be stateful or stateless (latter is the norm)
- The deterministic verification algorithm may be perfectly correct (never fails) or may fail with negligible probability

- Every instance has an associated message space $\mathcal{M}$ (which we assume to be implicitly defined when seeing the public key)
  - If there is a function k such that the message space is $\{0, 1\}^{k(n)}$ (with n being the security parameter), then the signature scheme supports message length $k(n)$
  - We will later see how we can generically construct signatures schemes for arbitrary message spaces from any scheme that supports message length $k(n)$

Formal Security Notions for Digital Signatures

- Attack model (increasing strength)
  - No-message attack (NMA): Adversary only sees public key
  - Random message attack (RMA): Adversary can obtain signatures for random messages (not in the control of the adversary)
  - Non-adaptive chosen message attack (naCMA): Adversary defines a list of messages for which it wants to obtain signatures (before it sees the public key)
  - Chosen message attack (CMA): Adversary can adaptively ask for signatures on messages of its choice

Formal Security Notions for Digital Signatures

- Goal of an adversary (decreasing hardness)
  - Universal forgery (UF): Adversary is given a target message for which it needs to output a valid signature
  - Existential forgery (EF): Adversary outputs a signature for a message of the adversary's choice

- **Security notion:** attack model + goal of the adversary

- For schemes used in practice: Adversary can not even achieve the weakest goal in the strongest attack model
  - **EUF-CMA**: existential unforgeability under chosen message attacks

EUF-CMA Security

$\sigma_i \leftarrow \mathsf{Sign}_{sk}(m_i)$

$Q \leftarrow Q \cup \{m_i\}$

m_i

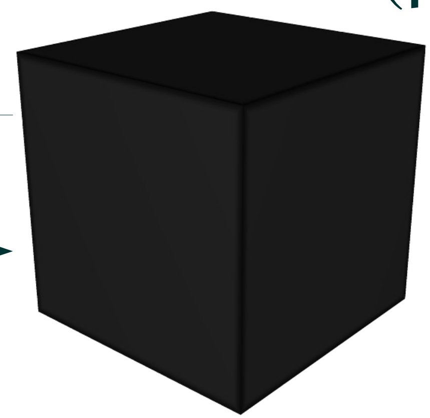
σ_i

$\mathcal{A}$

$(pk, sk) \xleftarrow{\$} \mathsf{Gen}(1^n)$

pk

(m^*, σ^*)



if $\mathsf{Vrfy}_{pk}(m^*, \sigma^*) = 1 \land$
$m^* \notin Q$
return 1
else return 0

A signature scheme scheme Σ = (Gen, Sig, Vrfy) is existentially unforgeabily under chosen message attacks (EUF-CMA) secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function negl s.t.

$$\Pr[\text{Sig-forge}^{\text{euf-cma}}_{\mathcal{A},\Sigma}(n)=1] \leq \text{negl}(n) .$$

Some Remarks on the Definition

- <u>One-time vs. many-time signatures</u>
  - The number of queries to the oracle may be limited, i.e., only a single query is allowed vs. arbitrary many are allowed

- <u>Weak vs. strong unforgeability</u>
  - In case of strong unforgeability the adversary wins if it outputs a valid signature even for a queried message, but the signature differs from the one obtained from the oracle
    - Oracle $\mathcal{Q}$ records (m_i,σ_i) and winning condition is: $(m^*,\sigma^*) \notin \mathcal{Q}$
    - Not achievable for re-randomizable signature schemes
  - We consider only standard (weak) unforgeability

RSA Signatures

- <u>KeyGen</u>: On input 1^n pick two random n-bit primes p,q, set N = pq, pick e s.t. gcd(e, φ(N)) = 1, compute d := e^{-1} mod φ(N) output (sk, pk) := ((d , N), (e ,N))
- <u>Sign</u>: On input $m \in \mathbb{Z}_N^*$ and sk = (d, N), compute and output

$$\sigma := m^d \bmod N$$

- <u>Vrfy:</u> On input a public key pk = (e, N), a message $m \in \mathbb{Z}_N^*$ and a signature $\sigma \in \mathbb{Z}_N^*$ output 1 if and only if

$$m := \sigma^e \bmod N$$

RSA Signatures

- To forge signature of a message m, the adversary, given N, e but not d, must compute m^d mod N, meaning invert the RSA function at m.

- But RSA is one-way so this task should be hard and the scheme should be secure. Correct?
- Of course not…
- No-message attacks
  1) Output forgery $(m^*, \sigma^*) := (1, 1)$. Valid since $1^d = 1$ mod N
  2) Choose $\sigma \in \mathbb{Z}_N^*$ and compute $m := \sigma^e$ mod N
- EUF-CMA attack
  - Ask signatures σ_1, σ_2 for $m_1, m_2 \in \mathbb{Z}_N^*$ and output $(m^*, \sigma^*) := (m_1 \cdot m_2 \bmod N, \sigma_1 \cdot \sigma_2 \bmod N)$

Even if it would be secure, a message space of $\mathbb{Z}_N^*$ is not desirable!

Extending the Message Space

- Block-wise signing
  - Consider $m := (m_1, \ldots, m_n)$ with $m_i \in \mathcal{M}$ and compute $\sigma := (\sigma_1, \ldots, \sigma_n)$
  - Need to take care to avoid mix-and-match attacks (block reordering, exchanging blocks from different signatures, etc.)
  - Inefficient for large messages (one invocation of the scheme per block)

- Hash-and-sign
  - Compress arbitrarily long message before signing by hashing them to a fixed length string using a hash function H
  - The range of H needs to be compatible with the message space of the signature scheme

Hash-and-Sign Paradigm (Construction 12.3)

- Let Σ = (Gen, Sign, Vrfy) be a signature scheme for messages of length k(n), and let Γ = (Gen_H, H) be a hash function with output length k(n). Construct signature scheme Σ' = (Gen' , Sign' , Vrfy') as follows:
  - Gen': on input 1^n , run Gen(1^n) to obtain (pk, sk) and run $Gen_H(1^n)$ to obtain s; the public key is (pk, s) and the private key is (sk, s).
  - Sign': on input a private key (sk, s) and a message $m \in \{0, 1\}^*$, output $\sigma \leftarrow Sign_{sk}(H(s, m))$.
  - Vrfy': on input a public key (pk, s), a message $m \in \{0, 1\}^*$, and a signature σ, output 1 if and only if $Vrfy_{pk}(H(s, m), \sigma) = 1$.

THEOREM 12.4: If Σ is a secure signature scheme for messages of length k and Γ is collision resistant, then Σ' is a secure signature scheme (for arbitrary-length messages).

Hash-and-Sign Paradigm

- Proof Idea
  - Let $m_1, \ldots, m_q$ be the messages queried by $\mathcal{A}$ and (m^*, σ^*) the valid forgery
    - Case 1: $H(s, m^*) = H(s, m_i)$ for some $i \in [q]$: we have a collision for H
    - Case 2: $H(s, m^*) \neq H(s, m_i)$ for all $i \in [q]$: we have that $(H(s, m^*), \sigma^*)$ is a forgery for Σ

- Hash-and-sign in practice
  - Used by signature schemes used in practice (RSA PKCS#1 v1.5 signatures, Schnorr, (EC)DSA, …)
  - Recall that we consider H to be keyed for theoretical reasons and in practice H would be any "good" collision-resistant hash function, e.g., SHA-3

RSA FDH Signatures

- Can we simply apply the hash-and-sign paradigm to RSA?
  - No, not assuming collision resistant hashing (or any other reasonable standard property of a hash function), as the underlying textbook RSA signature scheme does not provide any meaningful security
- But, we can apply the idea of hash-and-sign and model the hash function as a random oracle!
  - RSA Full Domain Hash (RSA-FDH)
  - The random oracle is collision resistant and destroys other "dangerous" algebraic properties
  - Important that range of H is (close to) $\mathbb{Z}_N{}^*$
  - H constructed via repeated application of an underlying cryptographic hash function such as SHA-3
- **Never say** "signing = d/encrypt the hash" when talking about signing (with RSA)!
  - "Misunderstanding" due to commutativity of RSA private and public key operation
  - Other signature schemes do usually not allow any such analogy

RSA FDH Signatures (Construction 12.6)

- KeyGen: On input 1^n pick two random n-bit primes p,q, set $N = pq$, pick e s.t. $\gcd(e, \varphi(N)) = 1$, compute $d := e^{-1} \bmod \varphi(N)$ output $(sk, pk) := ((d, N), (e, N))$. As part of the key generation a hash function $H: \{0,1\}^* \to \mathbb{Z}_N^*$ is specified (but we leave this implicit).
- Sign: On input $m \in \{0,1\}^*$ and $sk = (d, N)$, compute and output

$$\sigma := H(m)^d \bmod N$$

- Vrfy: On input a public key $pk = (e, N)$, a message $m \in \{0,1\}^*$ and a signature $\sigma \in \mathbb{Z}_N^*$ output 1 if and only if

$$H(m) := \sigma^e \bmod N$$

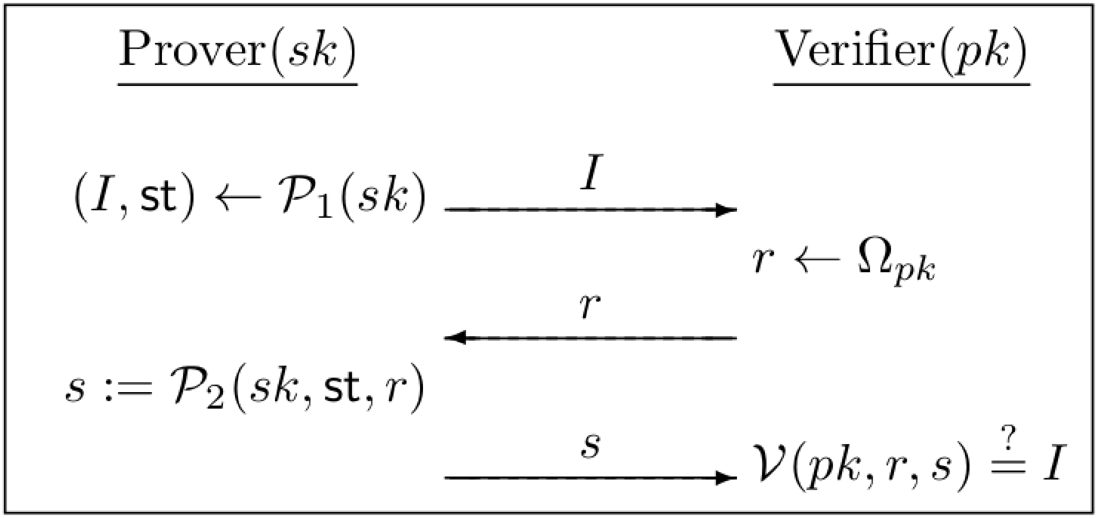
THEOREM 12.7: If the RSA problem is hard relative to GenRSA and H is modeled as a random oracle, then RSA-FDH is EUF-CMA secure.

RSA FDH Signatures (Proof Sketch – Naive Strategy)

- We again use the power of random oracles and reduce the EUF-CMA security to the RSA assumption
- We have to simulate signing queries without knowing the private key
  - Use the idea of the previously seen no-message attack against texbook RSA (i.e, choose a signature and compute the message)
  - We randomly choose an index $i \in [q_H]$ (the number of queries to H)
    - In the i'th query we will embed the RSA instance (N, e, y)
  - If adversary queries H for m_j
    - $j \neq i$: choose $\sigma_j \xleftarrow{\$} \mathbb{Z}_N^*$ and set $H(m_j) := \sigma_j^e \bmod N$, record $(m_j, \sigma_j, H(m_j))$ and return σ_j
    - $j = i$: return y
  - If adversary queries a signature for m_j
    - $j = i$: abort (our guess was wrong)
    - $j \neq i$: retrieve $(m_j, \sigma_j, H(m_j))$ and return σ_j
- Adversary outputs (m^*, σ^*), and if $m^* = m_i$ and $\sigma^{*e} = y \bmod N$, then output σ

Signatures in the Discrete Logarithm Setting

- We look at two popoluar schemes: Schnorr and DSA/ECDSA
- Both schemes can be viewed as signatures obtained from 3-move identification schemes
- Schnorr signatures
  - Applying the Fiat-Shamir heuristic: r computed as H(I, m) with H modeled as RO
  - Can be viewed as a non-interactive zero-knowledge proof of knowledge of a discrete logarithm (the private key)



- DSA/ECDSA
  - Uses a different transform then Fiat-Shamir (but similar idea)

Schnorr Signatures

- KeyGen: run $\mathcal{G}(1^n)$ to obtain (G, q, g). Choose $x \leftarrow^{\$} \mathbb{Z}_q$ and set $y := g^x$. The private key is x and the public key is (G, q, g, y). As part of key generation, a function $H : \{0, 1\}^* \to \mathbb{Z}_q$ is specified.
- Sign: on input a private key x and a message $m \in \{0, 1\}^*$, choose $k \leftarrow^{\$} \mathbb{Z}_q$ and set $I := g^k$. Then compute $r := H(I, m)$ and $s := rx + k \bmod q$. Output the signature $\sigma := (r, s)$.
- Vrfy: on input a public key (G, q, g, y), a message $m \in \{0, 1\}^*$, and a signature $\sigma = (r, s)$, compute $I := g^s \cdot y^{-r}$ and output 1 if $H(I, m) = r$.

Correctness: $g^s \cdot y^{-r} = g^{rx+k} \cdot g^{-xr} = g^k = I$

THEOREM: If the discrete-logarithm problem is hard relative to $\mathcal{G}$ and H is a random oracle, then the Schnorr signature scheme is EUF-CMA secure.

DSA/ECDSA

- <u>KeyGen</u>: run $\mathcal{G}(1^n)$ to obtain (G, q, g). Choose $x \leftarrow^{\$} \mathbb{Z}_q$ and set $y := g^x$. The private key is x and the public key is (G, q, g, y). As part of key generation, two functions $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ and $F : G \rightarrow \mathbb{Z}_q$ are specified.
- <u>Sign</u>: on input a private key x and a message $m \in \{0, 1\}^*$, choose $k \leftarrow^{\$} \mathbb{Z}_q$ and set $r := F(g^k)$. Then compute $s := k^{-1}(H(m)+rx) \bmod q$ (If r = 0 or k=0 or s = 0 then start again with a fresh choice of k). Output the signature $\sigma := (r, s)$.
- <u>Vrfy:</u> on input a public key (G, q, g, y), a message $m \in \{0, 1\}^*$, and a signature $\sigma = (r, s)$ with $r, s \neq 0 \bmod q$, compute $u = s^{-1} \bmod q$ output 1 if $r = F(g^{H(m)u}\, y^{ru})$.

- DSA works in a prime order q subgroup of $\mathbb{Z}_p^*$ and F(I) = I mod q.
- ECDSA works in elliptic curves. In case of a prime order q subgroup of $E(\mathbb{Z}_p)$ and I=(x, y), F(I) = x mod q
- If H and F modeled as random oracles, EUF-CMA secuirty can be proven under DL. But for these concrete forms above <u>no security proof is known</u>.

Schnorr, DSA/ECDSA Practical Aspects

- Bad randomness (Sony PS3 2010)
  - Recall in Schnorr: $s := rx + k \bmod q$ with $r := H(g^k, m)$
  - Signing two messages m, m' with $m \neq m'$ with same k yields

$$s = rx + k \bmod q \quad \text{and} \quad s' = r'x + k \bmod q$$

$$s - rx = s' - r'x \bmod q$$

$$x = (s' - s)(r' - r)^{-1} \bmod q$$

- Also practical attacks if the randomness is biased (https://eprint.iacr.org/2019/023)
- Countermeasure: make them deterministic (RFC 6979, EdDSA)
  - Compute $k := D(sk, m)$
  - Solves problem above, but opens up possibility for fault attacks
    - Trigger signing same message twice, trigger a fault in one run in m when computing H(m). The old attack then applies.
    - Countermeasure? Verification before outputting a signature, etc.

From any one-way functions (e.g., hash functions):

- Let H be a one-way function and assume 3-bit messages
- Private key is matrix of uniformly random values from the domain of H
- Public key is the matrix of images of sk elements under H

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & y_{3,0} \\ y_{1,1} & y_{2,1} & y_{3,1} \end{pmatrix} \qquad sk = \begin{pmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{pmatrix}$$

Signing $m = 011$:

$$sk = \begin{pmatrix} \boxed{x_{1,0}} & x_{2,0} & x_{3,0} \\ x_{1,1} & \boxed{x_{2,1}} & \boxed{x_{3,1}} \end{pmatrix} \Rightarrow \sigma = (x_{1,0},\ x_{2,1},\ x_{3,1})$$

Verifying for $m = 011$ and $\sigma = (x_1, x_2, x_3)$:

$$pk = \left.\begin{pmatrix} \boxed{y_{1,0}} & y_{2,0} & y_{3,0} \\ y_{1,1} & \boxed{y_{2,1}} & \boxed{y_{3,1}} \end{pmatrix}\right\} \Rightarrow \begin{matrix} H(x_1) \stackrel{?}{=} y_{1,0} \\ H(x_2) \stackrel{?}{=} y_{2,1} \\ H(x_3) \stackrel{?}{=} y_{3,1} \end{matrix}$$

Various techniques exist to obtain (stateful) many-times signatures

One-Time Signatures

From a concrete hardness assumption (DL):

- <u>KeyGen</u>: run $\mathcal{G}(1^n)$ to obtain (G, q, g). Choose $x, y \leftarrow^{\$} \mathbb{Z}_q$ and set $h := g^x$ and $c := g^y$. The private key is (x, y) and the public key is (G, q, g, h, c).
- <u>Sign</u>: on input a private key (x, y) and a message $m \in \mathbb{Z}_q$, compute and output $\sigma := (y-m)x^{-1} \bmod q$.
- <u>Vrfy:</u> on input a public key (G, q, g, h, c), a message $m \in \mathbb{Z}_q$, and a signature σ output 1 if $c = g^m h^\sigma$.

<u>Correctness:</u> $g^m h^\sigma = g^{m+x\sigma} = g^{m+x((y-m)/x)} = g^y = c$.

<u>THEOREM:</u> If the discrete-logarithm problem is hard relative to $\mathcal{G}$, then the signature scheme is EUF-1-naCMA secure.

Generic Compilers for Strong Security

- CMA from RMA
  - RMA scheme with message space k + q(k) and resulting CMA scheme with message space q(k)
  - For $m \in \{0, 1\}^*$ choose uniformaly random $m_L \leftarrow^{\$} \{0, 1\}^q$ and compute $m_R = m_L \oplus m$. Thus we have $m = m_L \oplus m_R$ (with both parts uniformly random)
  - Choose $r \leftarrow^{\$} \{0, 1\}^k$ and sign $r||m_L$ and $r||m_R$ with two independent keys sk_L and sk_R of Σ_{RMA}

- CMA from naCMA
  - Let Σ be a naCMA-secure scheme, Σ' be a naCMA-secure one-time scheme. Generate a long-term key-pair for Σ
  - For message m generate one-time key of Σ' and sign m with one-time key. Sign one-time public key using long-term signing key