RSA

Original PKE construction from 1978.
Motivation was to build PKE on the hardness of FACTORING.

The basic idea:

- $\underline{K(1^\lambda)}$: $sk = (n, p, q, d)$ s.t.

  $n = p \cdot q$ ($|p| \approx |q| \approx \lambda$ bits).

  $pk = (n, e)$

$e \cdot d \equiv 1 \mod \varphi(n) \rightsquigarrow \varphi(n) = (p-1)(q-1)$

Typical choice; e is a prime (e.g. $e = 3$).

- $\underline{Enc}(\underline{pk}, m \in \underline{\mathbb{Z}_n^*}) : c = m^e \bmod n$

- $\underline{Dec}(\underline{sk}, \underline{c}) : c^d \bmod n$.

$$c^d \equiv (m^e)^d \equiv m^{ed} \equiv$$

$$\equiv m^{t \cdot \varphi(n) + 1}$$

$$\equiv (\underbrace{m^{\varphi(n)}}_{1})^t \cdot m$$

$$\equiv m \bmod n$$

Of course NOT CPA secure! Because it is deterministic.

Solution: Encode m into $\hat{m}$ in a randomized (invertible) fashion.

- PKCS # 1.5. Basically: $\hat{m} = (m \| r)$ where r is a random bitstring.

Is it provably secure? Under which assumption and for what size $|r|$?

Clearly $|r|$ cannot be too short ($O(\log \lambda)$) otherwise we can just guess it.

Unfortunately, for real-world values of r we can't prove RSA with PKCS#1.5 padding CPA secure.

On the other hand, if $m \in \{0,1\}$ and all the rest is a random r then it can be proven CPA secure. Under which assumption?

DEF (RSA Assumption). Let $Gen_{RSA} \to (pk, sk)$ be the Kgen in RSA. Then, for all PPT A:

$$\Pr\left[A(y, n, e) = x : x \leftarrow \mathbb{Z}_n^* ; y = x^e \bmod n \right.$$

$$pk = (n, e) ; sk = (n, d)$$

$$\left. \text{GenRSA}(1^\lambda) \right]$$

$$\leq \text{negl}(\lambda)$$

Equivalently: $f_{n,e}(x) = f_{pk}(x)$ is

a OWF. $= x^e \bmod n$

What's the relation between RSA and FACTORING? RSA $\Rightarrow$ FACTORING.
Because given p, q we can compute d and recover $x = y^d \bmod n$.
Other direction is UNKNOWN at least for general algorithms.

- PKCS # 2.0. A different encoding that suffices for CCA security.
What this is basically is a 2-round

Feistel network using hash functions G, H stretching with $m \| r$ where r is uniform.

The proof requires RSA assumption plus the assumption that G, H are RANDOM ORACLES.

What really RSA is? A TRAPDOOR PERMUTATION: $(Kgen, f, f^{-1})$

s.t. $(pk, sk) \leftarrow Kgen(1^\lambda)$

$$f_{pk} : \mathcal{X} \to \mathcal{X}$$

$$f_{sk}^{-1} : \mathcal{X} \to \mathcal{X}$$

Security: f_{pk} is a OWP even given pk.

$$\forall x \in \mathcal{X} : f_{sk}^{-1}(f_{pk}(x)) = x.$$

(RSA: $pk = (n, e)$, $sk = (n, d)$

$f_{pk}(x) = x^e \bmod n = y$

$f_{sk}(y) = y^d \bmod n.$)

FACT TDP $\Rightarrow$ PKE.

Let $(Kgen, f, f^{-1})$ be a TDP. Let h be hard-core for f.

$$Enc(pk, m \in \{0,1\}) : \left(f_{pk}(r),\ h_{pk}(r) \oplus m\right)$$

$$r \leftarrow \mathcal{X}$$

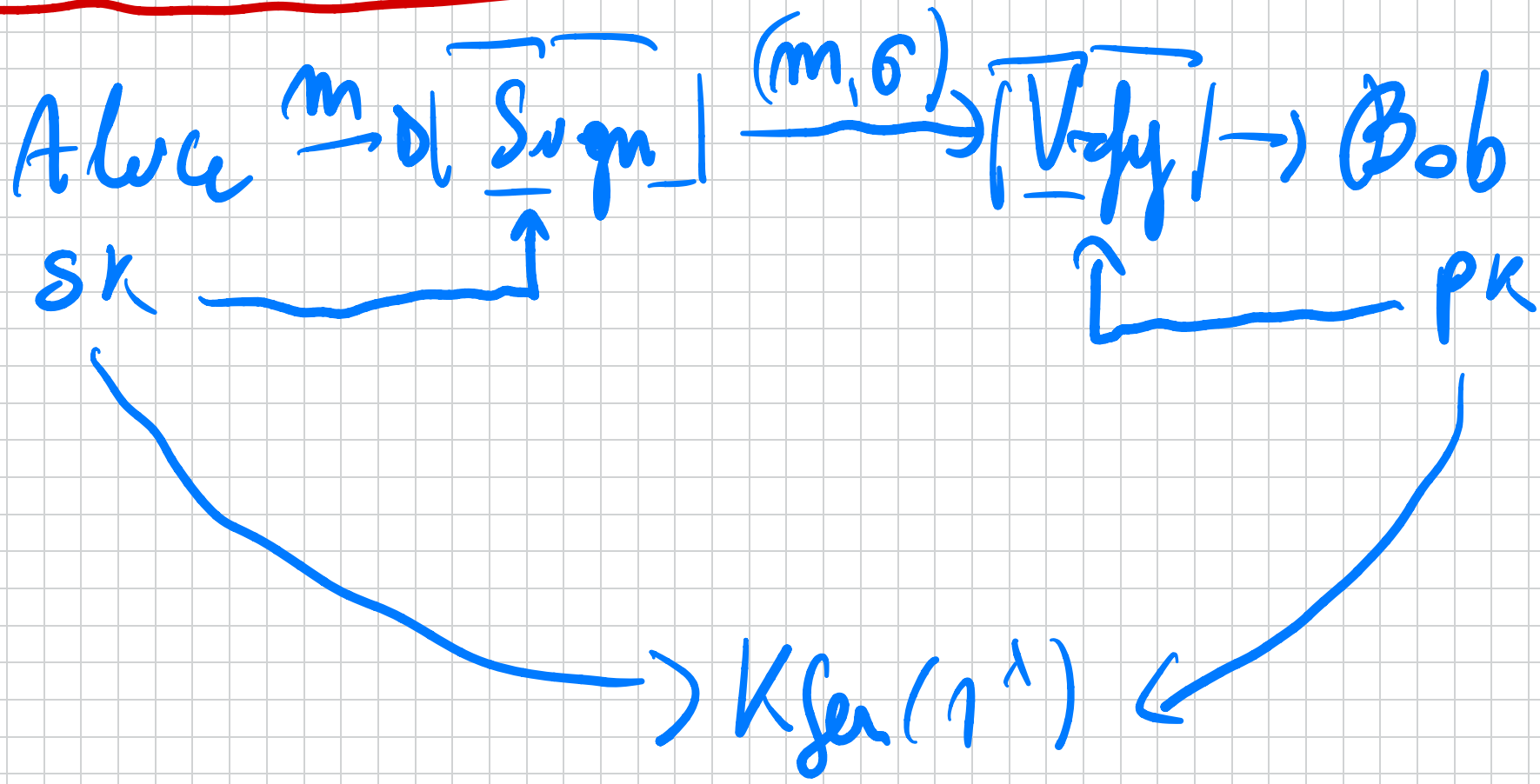
Note: PKE for 1 bit $\Rightarrow$ PKE for poly(λ) bits.

OPIS CODE : W7UMJ4Y1

Here is what we plan to do next:

- Digital signatures : FDH (RSA) and Fiat-Shamir (DL, ...)
- Post-quantum crypto, lattices for PKE and DS.
- Exercises.
- Bonus: CCA security, Zero Knowledge ....

DIGITAL SIGNATURES



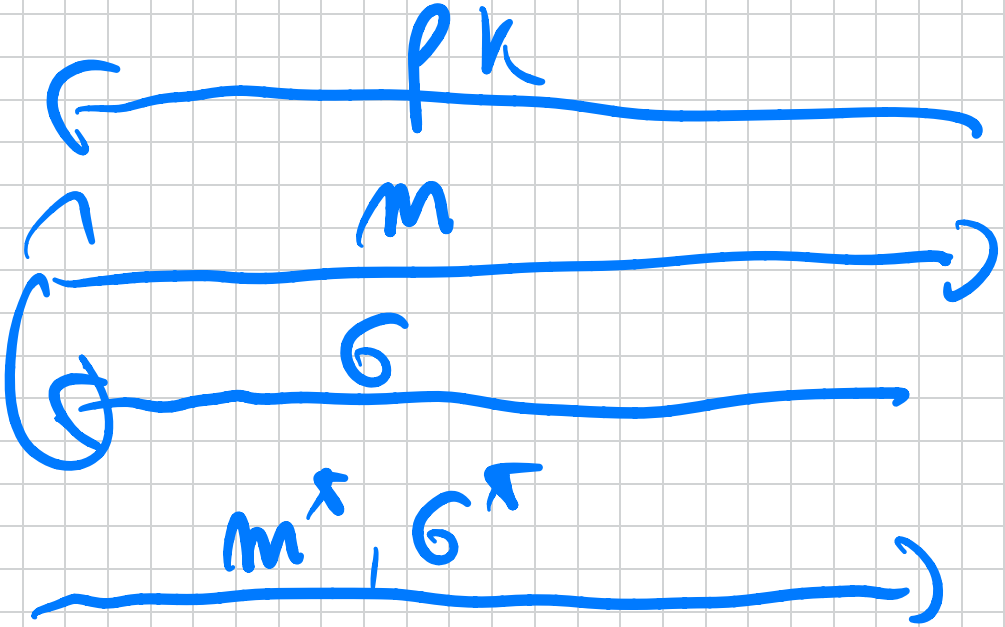
DEF (UF-CMA) We say $\Pi = (Kgen, Sign, Vrfy)$ is UF-CMA if $\forall$ PPT

$$Pr\left[G_{\Pi,A}^{\text{unforg}}(\lambda) = 1 \right] \leq negl(\lambda)$$

$G_{\Pi,A}^{\text{unforg}}(\lambda)$

A

C



$(pk, sk) \leftarrow Kgen(1^\lambda)$

$\sigma = Sign(sk, m)$

Output 1

iff : $Vrfy(pk, m^*, \sigma^*) = 1$

$m^* \neq$ any m_i

Most important application : PKI and X.509 certificates.

Simplest construction : RSA signatures.

- $\underline{KGen}(1^\lambda)$: $(n, e) = pk$; $sk = (n, d)$ as in RSA.

- $\underline{Sign}(\underline{sk}, \underline{m})$: $\sigma = m^d \bmod n$.

- $\underline{Verify}(pk, \underline{m}, \sigma)$: Check $\sigma^e = m \bmod n$

Not UF-CMA! Why? Signature of m, i.e. $\sigma = m^d \bmod n$.

$$\sigma^2 = (m^d)^2 = (m^2)^d \bmod n$$

valid signature on $m^* = m^2$.

Or, pick any σ and let $m = \sigma^e \bmod n$.

Exercise: Forge on ANY chosen $m^* \leftarrow \mathbb{Z}_n^*$.

Full-Domain Hash (FDH): Hash

the msg first: $\sigma = H(m)^{-1} \bmod n$

More in general: Use a TDP $(Kgen, f, f^{-1})$.

$Sign(SK, m): \sigma = f_{SK}^{-1}(y)$; $y = H(m)$

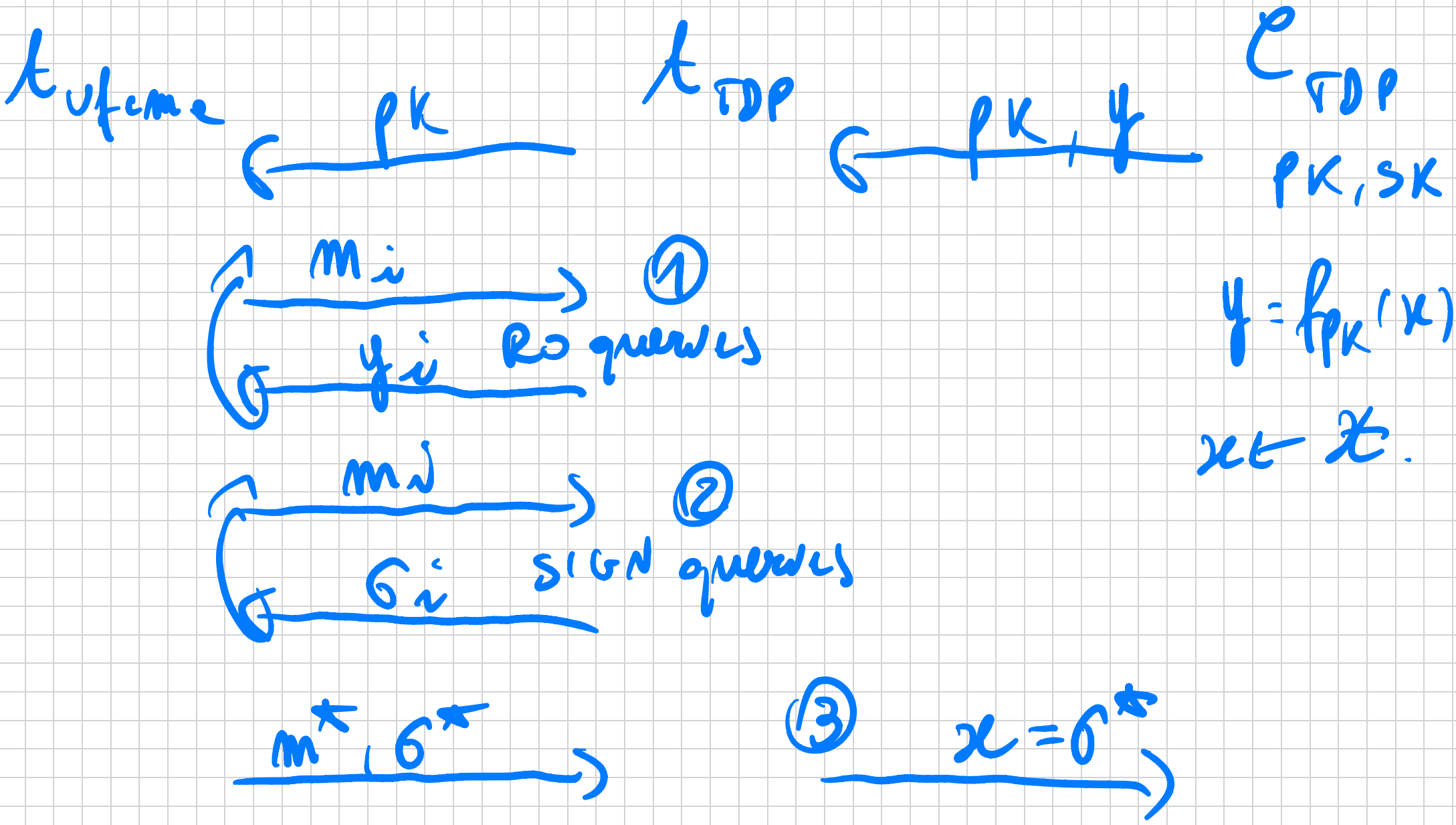
$Vrfy(pk, m, \sigma)$: Check $f_{pk}(\sigma) = y = H(m)$

THM Assuming the TDP is one-way FDH is UFCMA in the ROM.

Proof. Main trick: In the ROM the attacker would ask RO queries. This will help simulating signature queries. We make some assumptions wlog:

- Let q_h = # RO queries, q_s = # sign queries.
- A never repeats queries.
- Before asking signature on m_i or forging m^*, A will ask to RO the query m_i / m^*.

Here is the reduction:



② If σ^* is VALID, then σ^* is a pre-image of $y^* = H(m^*)$.

Main idea: Try & enforce that $H(m^*) = y^* = y$.

While we don't know when m^* was asked to the RO, we can guess it.

At the beginning pick: $j \leftarrow [q_h]$.

① Upon input a RO query m_i:

- If $i = j$, return $H(m_i) = y$.

- If $w \neq j$, sample $x_i \leftarrow \mathcal{X}$ and output $y_i = H(m_w) = f_{pk}(x_w)$

② Upon m_i, return x_w corresponding to y_w.

③ When the forgery outputs $\sigma^* = x$.

Assuming j is guessed correctly:

- The simulation is perfect. Ro queries and SIGN queries are correctly distributed
- W.p. $1/poly$ σ^* is a pre-image of $H(m^*) = y$.

$$Pr[A_{SDP} wins] = Pr[guessing\ j] \times Pr[A_{unforge} wins]$$

$\geq 1/\text{poly}(n)$ ∎