Digital Signature Schemes and the Random Oracle Model

A. Hülsing

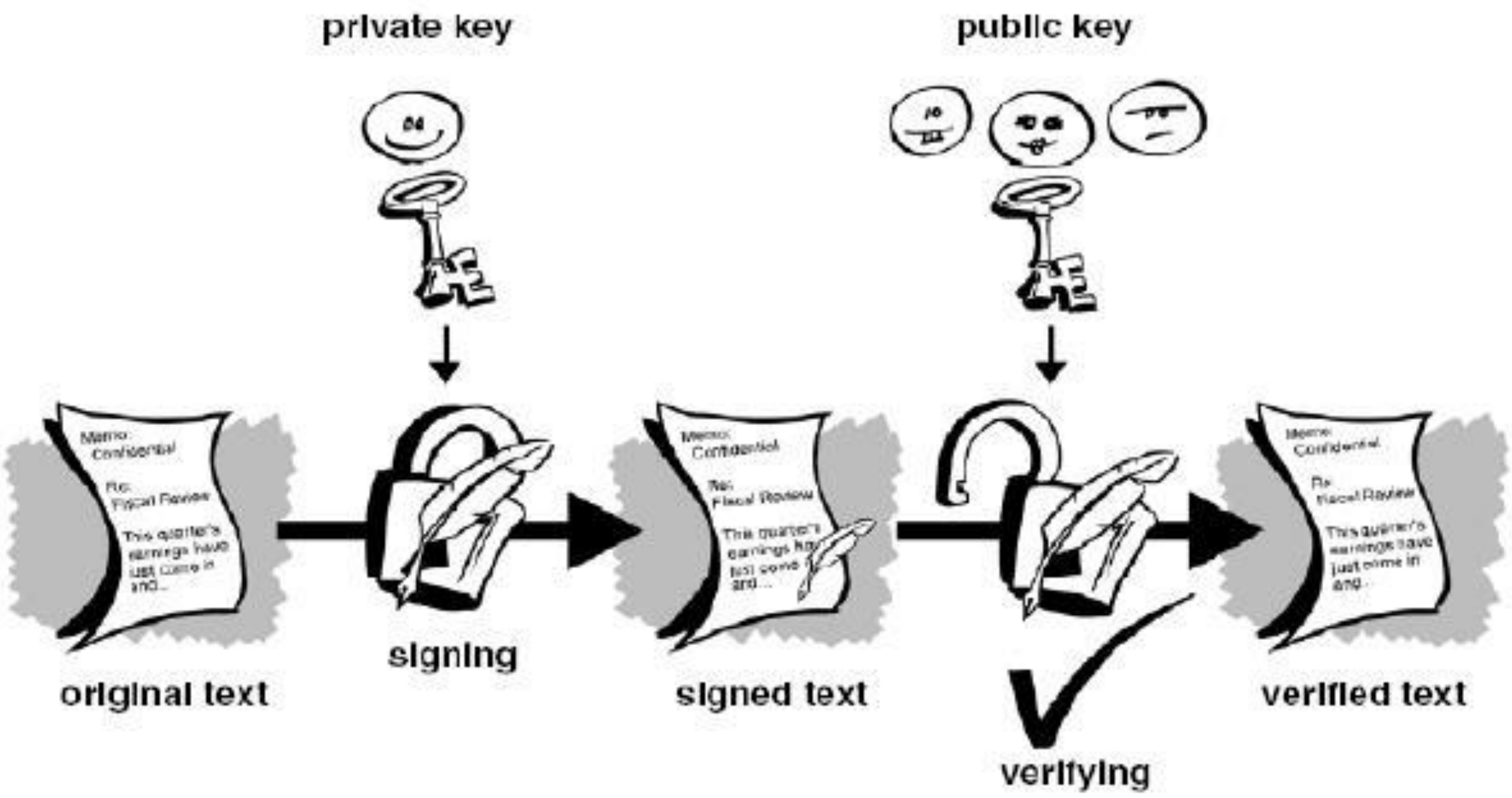
Technische Universiteit
Eindhoven
University of Technology

Today‘s goal

Review provable security of “in use” signature schemes. (PKCS #1 v2.x)

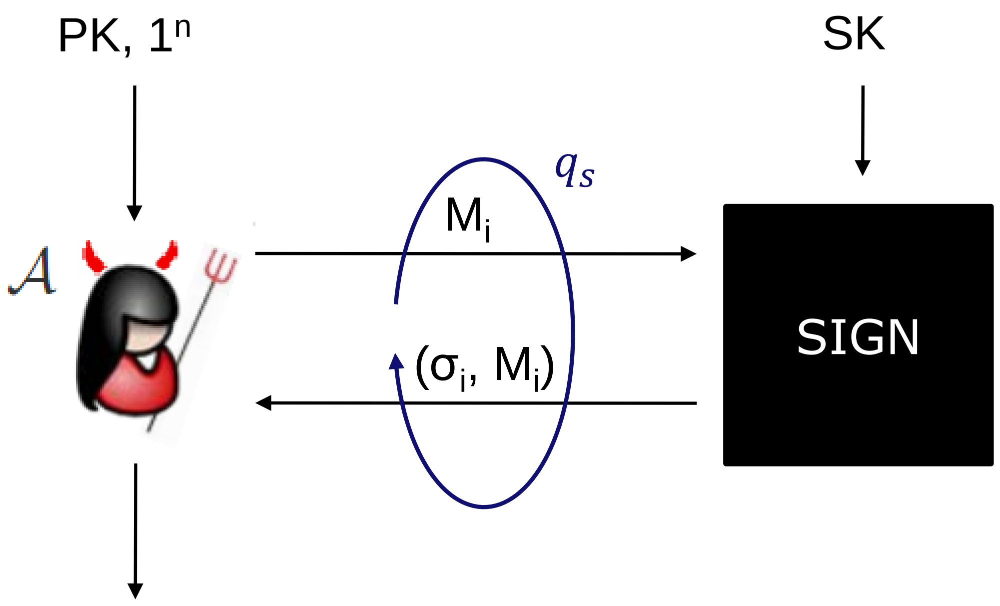
RSA®

Digital Signature



Source: http://hari-cio-8a.blog.ugm.ac.id/files/2013/03/DSA.jpg

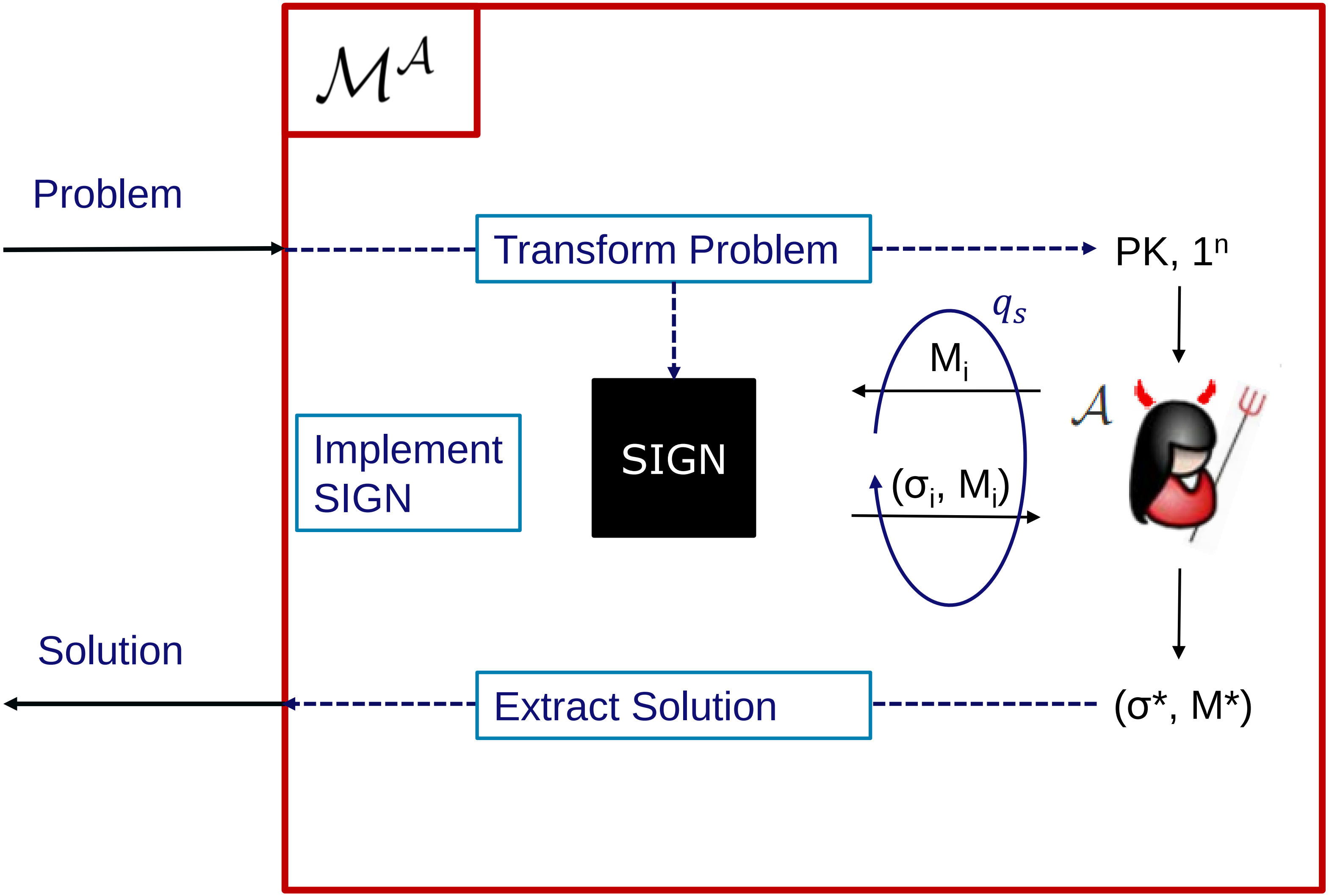
Existential unforgeability under adaptive chosen message attacks



(σ^*, M^*)

Success if $M^* \neq M_i$, $\forall$ i $\in [0, q]$ and Verify(pk,σ*,M*) = Accept

Reduction



Technische Universiteit **Eindhoven** University of Technology

Why security reductions?

- **Current RSA signature standard so far unbroken**
- **Vulnerabilities might exist! (And existed for previous proposals)**
- **Might be possible to forge RSA signatures without solving RSA problem or factoring!**

What could possibly go wrong?

RSA

Let $(\mathrm{N},\mathrm{e},\mathrm{d}) \longleftarrow \mathrm{GenRSA}(1^k)$ be a PPT algorithm that outputs a modulus N that is the product of two k-bit primes (except possibly with negligible probability), along with an integer $e > 0$ with $\gcd(e, \phi(N)) = 1$ and an integer $d > 0$ satisfying $ed = 1 \bmod \phi(N)$.

For any $(\mathrm{N},\mathrm{e},\mathrm{d}) \longleftarrow \mathrm{GenRSA}(1^k)$ and any $y \in \mathbb{Z}_N^*$ we have

$$(y^d)^e = y^{de} = y^{de \bmod \phi(N)} = y^1 = y \bmod N$$

RSA Assumption

Definition 1. We say that the RSA problem is hard relative to $\mathbf{GenRSA}$ if for all PPT algorithms A, the following is negligible:

$$\boldsymbol{Pr[(N,e,d) \leftarrow \mathbf{GenRSA}(1^k);\ y \leftarrow \mathbb{Z}_N^*;}$$
$$\boldsymbol{x \leftarrow A(N,e,y) : x^e = y \bmod N].}$$

A simple RSA Signature

$\mathbf{KeyGen}(1^k)$: Run $(N, e, d) \leftarrow \mathbf{GenRSA}(1^k)$.

Return (pk, sk) with $pk = (N, e), sk = d$.

$\mathbf{Sign}(sk, M)$: Return $\sigma = (M^d \bmod N)$

$\mathbf{Verify}(pk, M, \sigma)$: Return 1 iff $\sigma^e \bmod N == M$

The Blinding Attack

Given public key $pk = (N, e)$

To create a forgery on any target message M:

1. Sample random $r \in \mathbb{Z}_N^*$
2. Ask for signature σ on $r^e M \bmod N$
3. Output forgery $(M, \frac{\sigma}{r} \bmod N)$

Recall $\sigma = (r^e M)^d = r^{ed} M^d = r M^d \bmod N$

Hence $\frac{\sigma}{r} = M^d \bmod N$

A slightly better RSA Signature

Assume Hashfunction $\boldsymbol{H}: \{0,1\}^* \to \{0,1\}^n$ **for e.g.** $\boldsymbol{n = 160}$

$\mathbf{KeyGen}(1^k)$**: Run** $(\boldsymbol{N}, \boldsymbol{e}, \boldsymbol{d}) \leftarrow \mathbf{GenRSA}(1^k)$**.**

Return $(\boldsymbol{pk}, \boldsymbol{sk})$ **with** $\boldsymbol{pk} = (\boldsymbol{N}, \boldsymbol{e}), \boldsymbol{sk} = \boldsymbol{d}$**.**

$\mathbf{Sign}(\boldsymbol{sk}, \boldsymbol{M})$**: Pad with suff. zeros that**

$$(\mathbf{0} \ldots \mathbf{0} || \boldsymbol{H}(\boldsymbol{M})) \in \mathbb{Z}_N^*$$

Return $\boldsymbol{\sigma} = ((\mathbf{0} \ldots \mathbf{0} || \boldsymbol{H}(\boldsymbol{M}))^d \bmod \boldsymbol{N})$

$\mathbf{Verify}(\boldsymbol{pk}, \boldsymbol{M}, \boldsymbol{\sigma})$**: Return 1 iff** $\boldsymbol{\sigma}^e \bmod \boldsymbol{N} == (\mathbf{0} \ldots \mathbf{0} || \boldsymbol{H}(\boldsymbol{M}))$

Remember Index Calculus?

Given public key $pk = (N, e)$

1. Select a bound y and let $S = (p_1, \ldots, p_l)$ be the list of primes smaller than y.
2. Find at least $l + 1$ messages M_i such that each $(0 \ldots 0||H(M_i))$ is a product of primes in S.
3. Express one $(0 \ldots 0||H(M_j))$ as a multiplicative combination of the other $(0 \ldots 0||H(M_i))$ by solving a linear system given by the exponent vectors of the $(0 \ldots 0||H(M_i))$ with respect to the primes in S.
4. Ask for the signatures on all $M_i, i \neq j$ and forge signature on M_j.

Step 3

Write $\mu(M_i) = (0 \ldots 0||H(M_i))$

1. We can write $\forall M_i, 1 \leq i \leq \tau$: $\mu(M_i) = \prod_{j=1}^{l} p_j^{v_{i,j}}$
2. Associate with $\mu(M_i)$ length l vector $V_i(v_{i,1} \bmod e, \ldots, v_{i,l} \bmod e)$
3. $\tau \geq l + 1$ and there are only l linearly independent length l vectors: We can express one vector as combination of others mod e. Let this be $V_\tau = \sum_{i=1}^{\tau-1} \beta_i V_i + e\Gamma$; for some $\Gamma = (\gamma_1, \ldots, \gamma_l)$
4. Hence

$$\mu(M_\tau) = \left(\prod_{j=1}^{l} p_j^{\gamma_j}\right)^e \prod_{i=1}^{\tau-1} \mu(M_i)^{\beta_i}$$

Step 4

1. Ask for signatures $\sigma_i = \mu(M_i)^d \bmod N$ on M_i for $1 \leq i < \tau$

2. Compute:

$$\sigma^* = \mu(M_\tau)^d = \left(\prod_{j=1}^{l} p_j^{\gamma_j}\right) \prod_{i=1}^{\tau-1} \left(\mu(M_i)^d\right)^{\beta_i} \bmod N$$

3. Output forgery (σ^*, M_τ)

Summing up

- **Original attack (Misarsky, PKC'98) works even for more complicated paddings (ISO/IEC 9796-2)**
- **Attack only works for small n!**
- **But using SHA-1 (n=160) the attack takes much less than 2^{50} operations!**

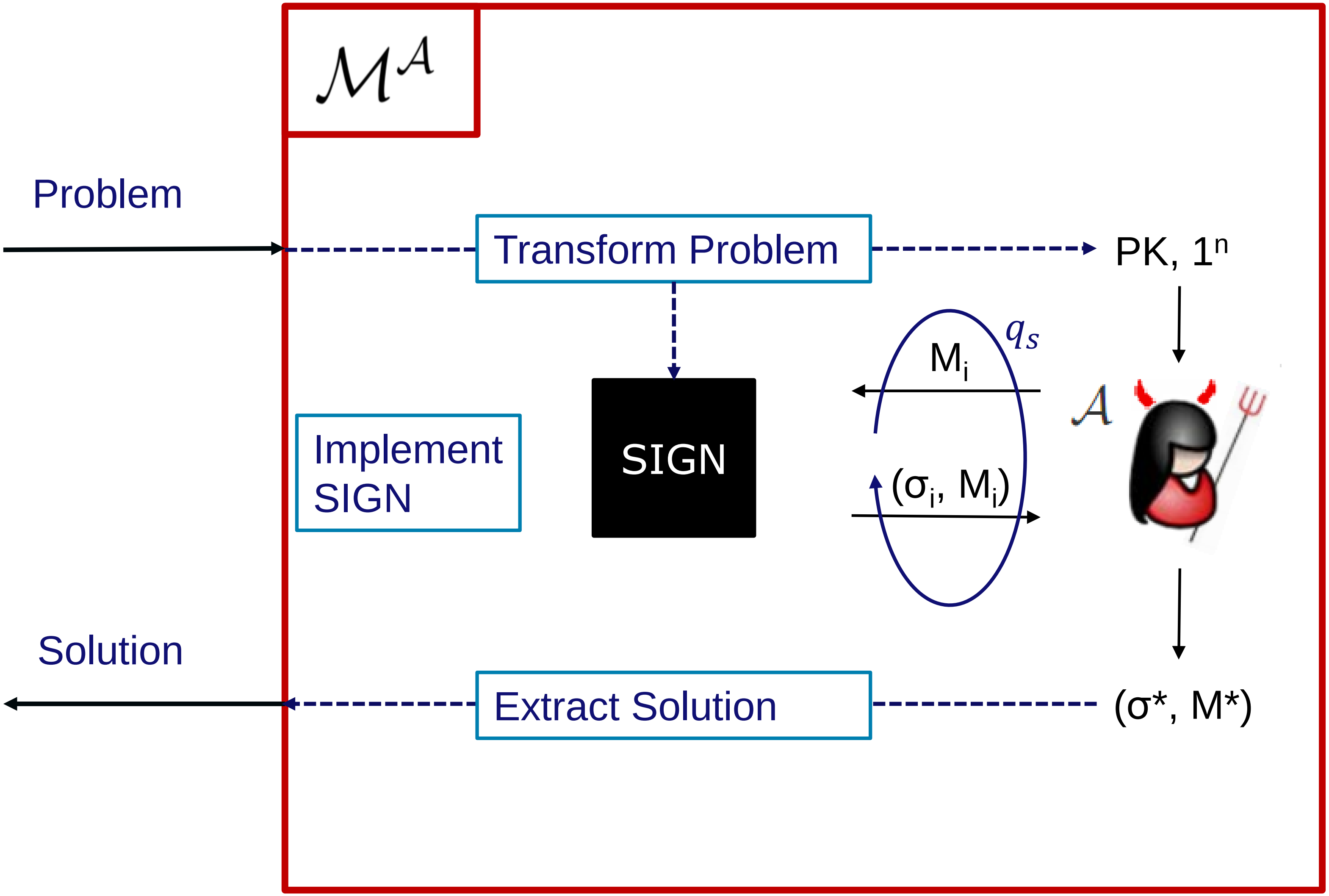
There are many ways to make mistakes...

(Similar attacks apply to encryption!)

That‘s why we want security reductions

The Random Oracle Model

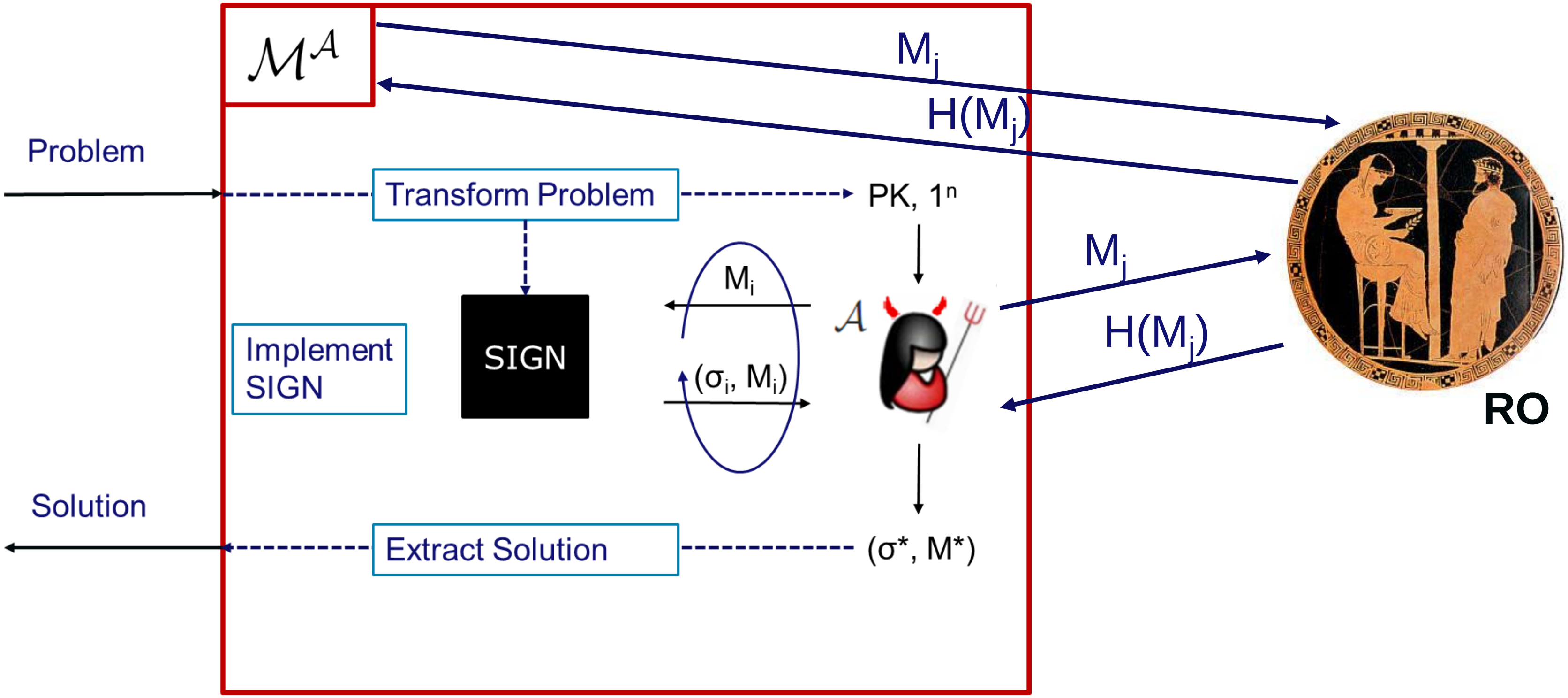
Reduction



$\mathcal{M}^{\mathcal{A}}$
Problem
Transform Problem
PK, 1^n
Implement SIGN
SIGN
q_s
M_i
(σ_i, M_i)
$\mathcal{A}$
Solution
Extract Solution
(σ^*, M^*)

Random Oracle Model (ROM)

- **Idealized Model**
- **Perfectly Random Function**



Technische Universiteit Eindhoven University of Technology

How to implement RO?

"Lazy Sampling":

- **Keep list of (x_i, y_i)**
- **Given M_j, search for $x_i = M_j$**
- **If $x_i = M_j$ exists, return y_i**
- **Else sample new y from Domain, using uniform distribution**
- **Add (M_j, y) to table**
- **Return y**

RO

ROM security

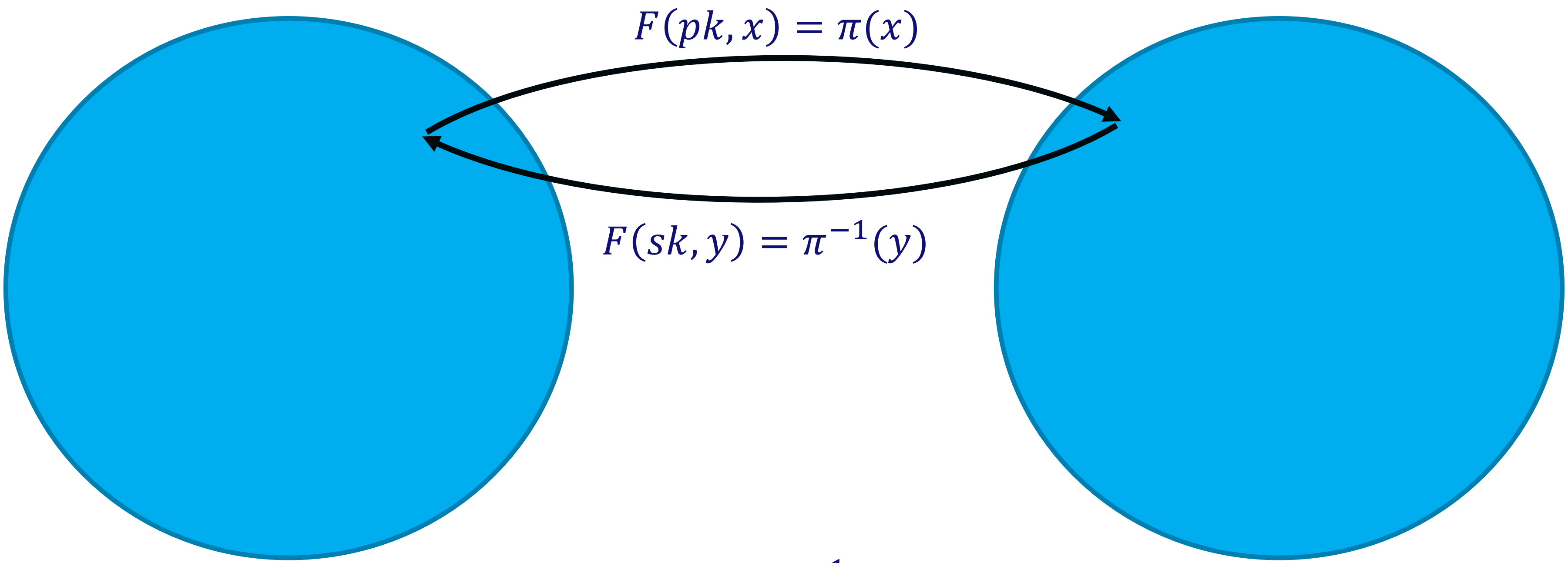
- Take scheme that uses cryptographic hash
- For proof, replace hash by RO
  - Different flavors:
    Random function vs. Programmable RO

- ➢ Heuristic security argument
- ➢ Allows to verify construction
- ➢ Worked for ”Natural schemes” so far

- ➢ However: Artificial counter examples exist!

Full Domain Hash Signature Scheme

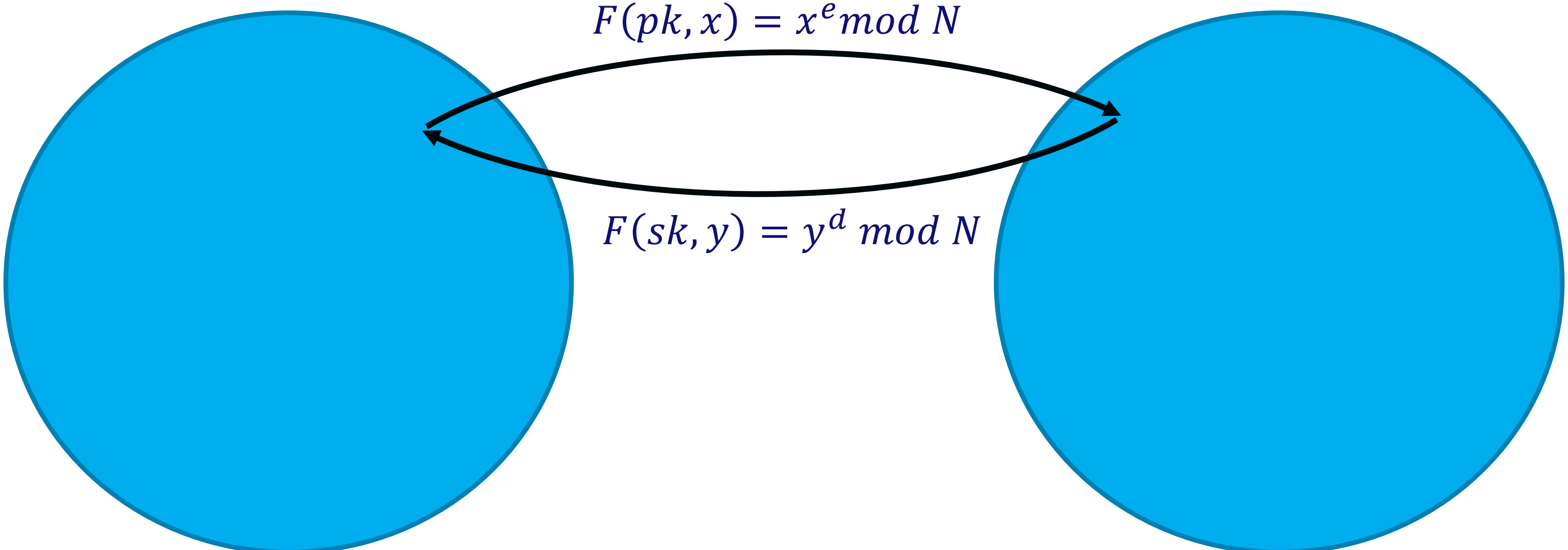
Trapdoor (One-way) Permutation



Computing $\pi^{-1}(y)$
without knowledge of sk
computationally hard

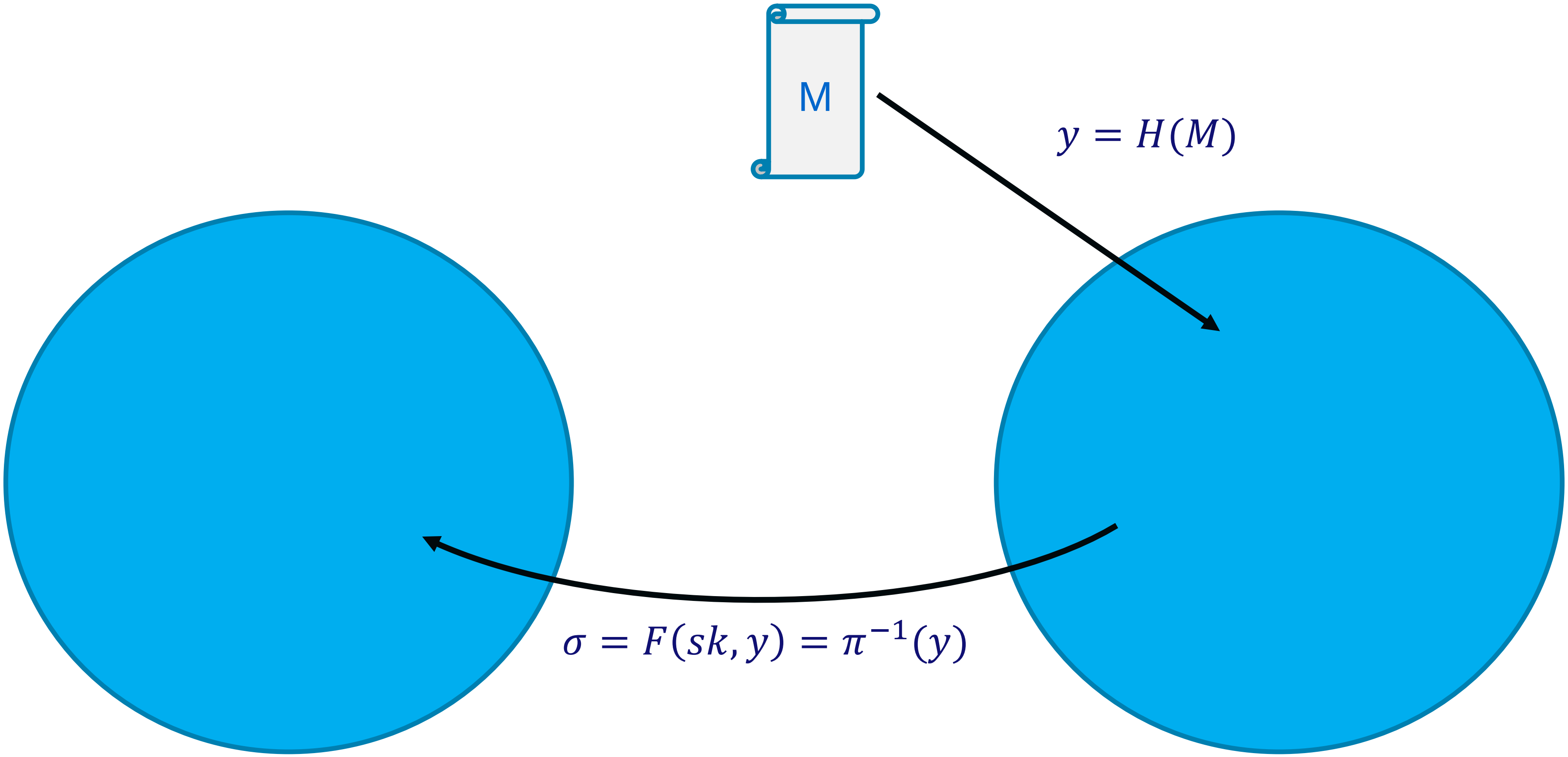
RSA Trapdoor (One-way) Permutation

$(\boldsymbol{N}, \boldsymbol{e}, \boldsymbol{d}) \leftarrow \mathbf{GenRSA}(1^k)$; $\boldsymbol{pk} = (\boldsymbol{N}, \boldsymbol{e})$; $\boldsymbol{sk} = \boldsymbol{d}$



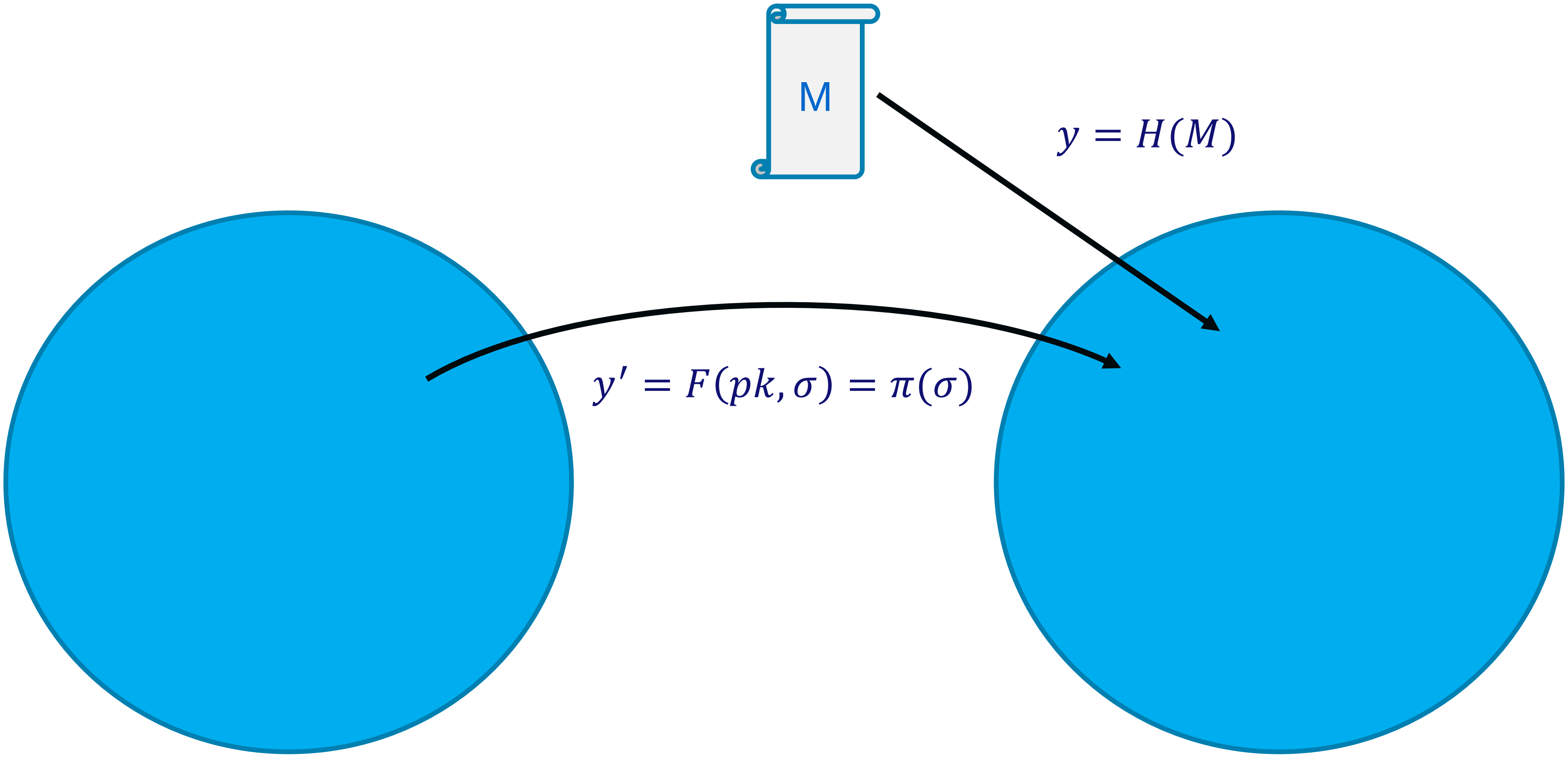
Computing $\pi^{-1}(y)$
without knowledge of sk
computationally hard if
RSA Assumption holds

Generic FDH: Sign



$$\begin{aligned} \sigma &= Sign(sk, M) \\ &= \pi^{-1}(H(M)) \\ &= F(sk, H(M)) \end{aligned}$$

Generic FDH: Verify



$$Verify(pk, M, \sigma):$$
$$check\ \ y = H(M)\ == \ \pi\,(\sigma) = F(pk, \sigma) = y'$$

RSA-PFDH

- **Randomized FDH**
- **Simplified RSA-PSS**
  - **Standardized in PKCS #1 v2**

    (slightly different randomization)
- **Tight Reduction from RSA Assumption in ROM**

RSA-PFDH

Assume Hashfunction $H: \{0,1\}^* \rightarrow \mathbb{Z}_N^*$

$\mathbf{KeyGen}(1^k)$: **Run** $(N, e, d) \leftarrow \mathbf{GenRSA}(1^k)$.

Return (pk, sk) **with** $pk = (N, e), sk = d$.

$\mathbf{Sign}(sk, M)$: **Sample** $r \xleftarrow{\$} U_\kappa$; **Compute** $\mathrm{y} = H(r||M)$

Return $\sigma = (r, y^d \bmod N)$

$\mathbf{Verify}(pk, M, \sigma)$: **Return 1 iff** $\sigma^e \bmod N == H(r||M)$

RSA-PFDH Security

If the RSA Assumption holds, RSA-PFDH is existentially unforgeable under adaptive chosen message attacks.

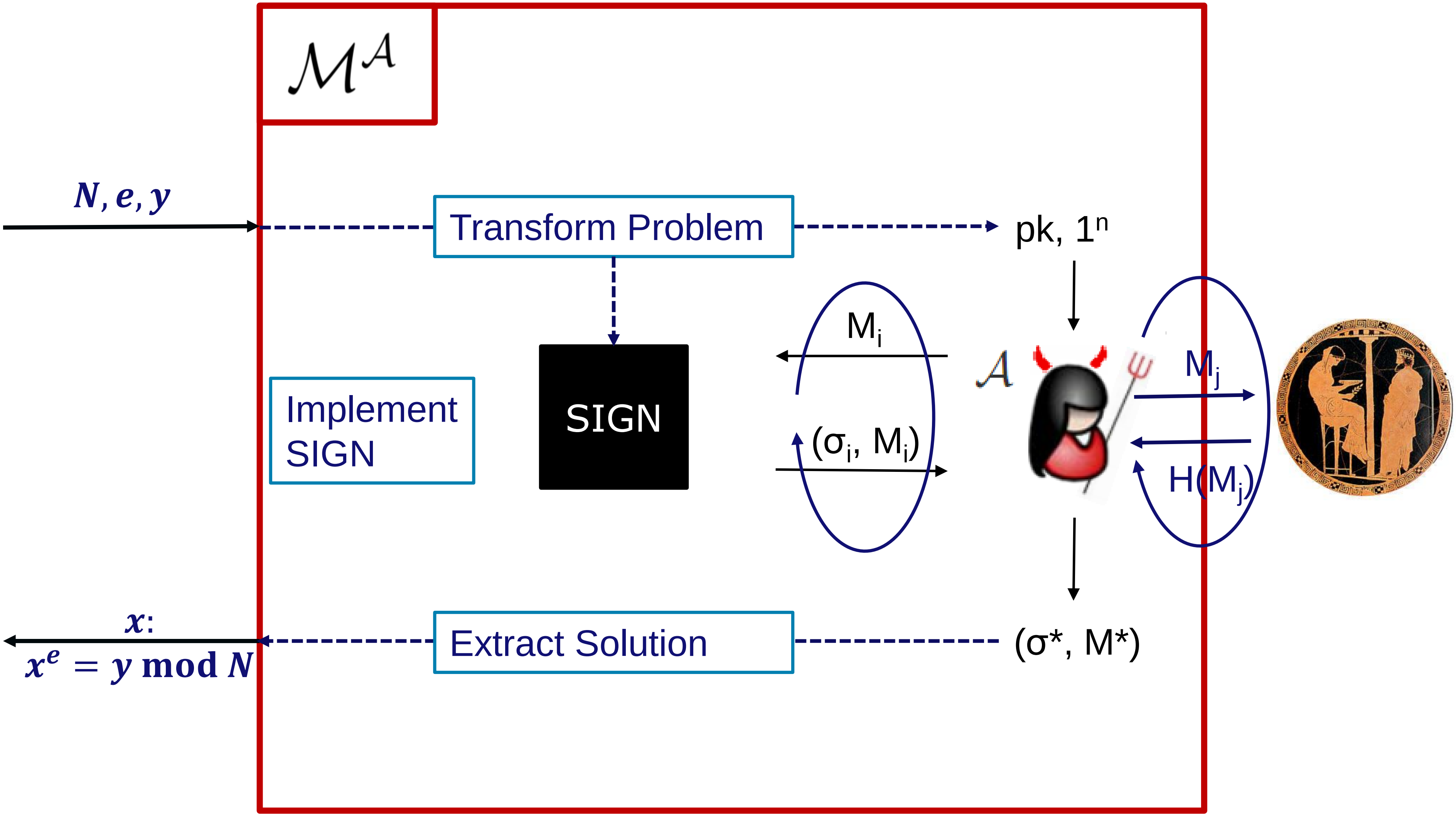
Proof

Idea:

Show that any forger A against RSA-PFDH can be used to break the RSA Assumption with ~ the same time and success probability.

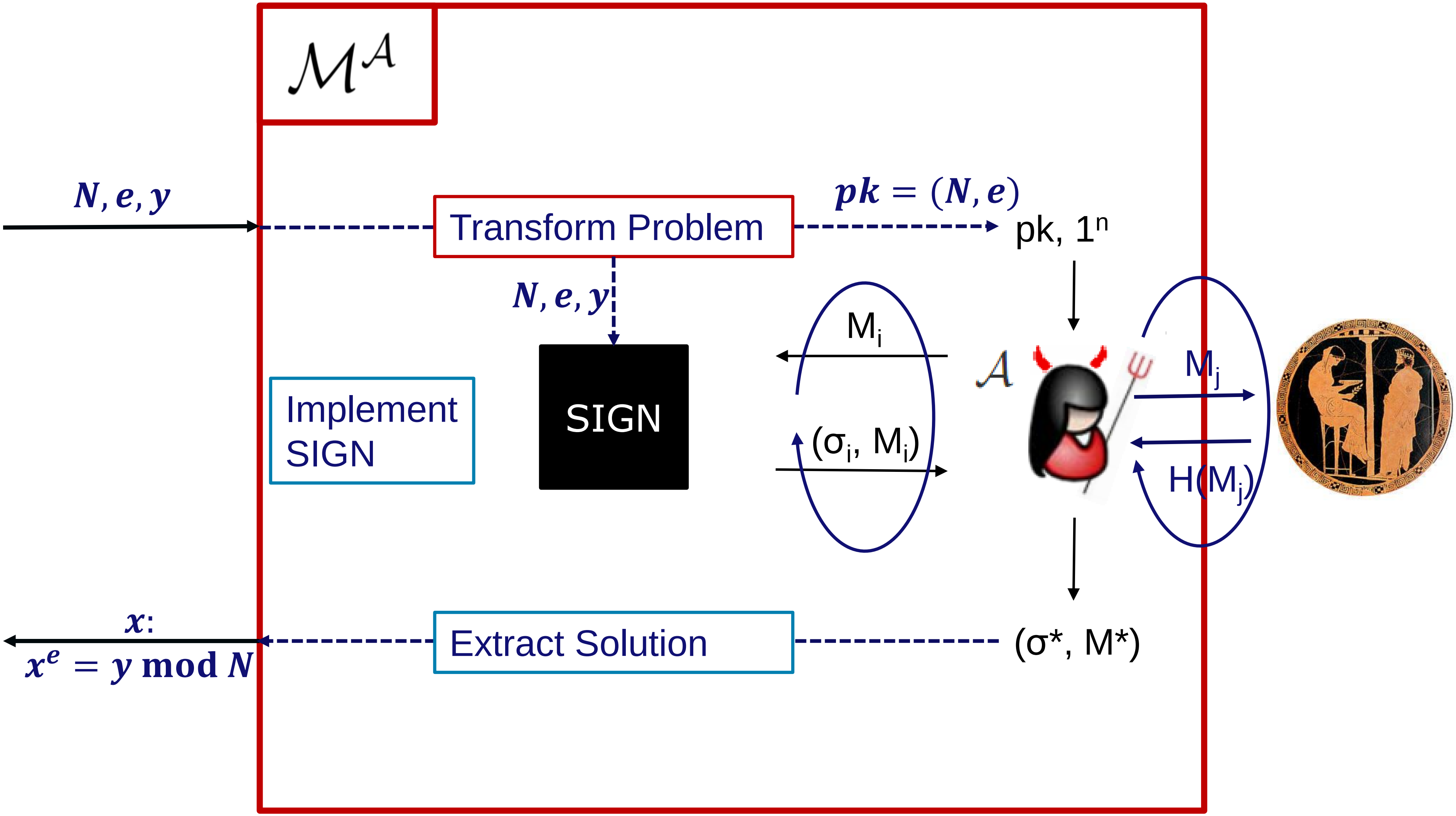
"Given a forger A against RSA-PFDH with success probability ε, we construct an oracle Machine M^{A} that succeeds with probability $\varepsilon/4$."

Reduction

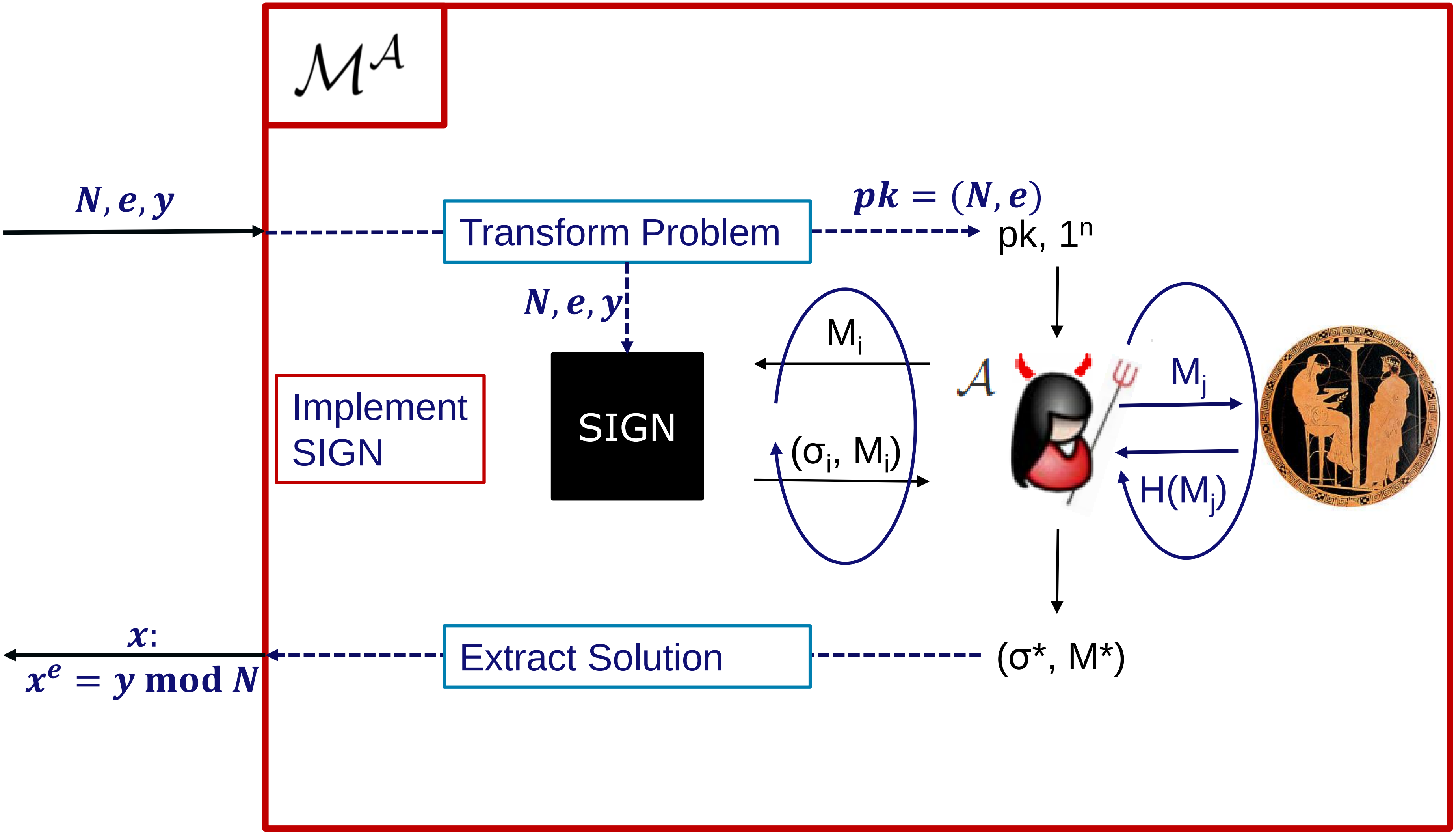


$\mathcal{M}^{\mathcal{A}}$
N, e, y
Transform Problem
pk, 1^n
Implement SIGN
SIGN
M_i
(σ_i, M_i)
$\mathcal{A}$
M_j
$H(M_j)$
x:
$x^e = y \bmod N$
Extract Solution
(σ^*, M^*)

Reduction: Transform Problem



Reduction: Implement SIGN



Implement SIGN – Implement RO

- **Keep table of tripples $(.,.,.)$**
- **When A asks for $H(r||M)$:**

1. **If there is an entry $(r||M, x, z)$ in table, return z**
2. **If list L_M already exists, go to 3. Otherwise, choose q_s values $r_{M,1}, \dots, r_{M,q_s} \leftarrow \{0,1\}^\kappa$ and store them in a list L_M.**
3. **If $r \in L_M$ then let i be such that $r = r_{M,i}$. Choose random $x_{M,i} \in \mathbb{Z}_N^*$ and return the answer $z = x_{M,i}{}^e \bmod N$. Store $(r||M, x_{M,i}, z)$ in the table.**
4. **If $r \notin L_M$, choose random $x \in \mathbb{Z}_N^*$ and return the answer $z = yx^e \bmod N$. Store $(r||M, x, z)$ in the table.**

Implement SIGN

- **When A requests some message M to be signed for the i th time:**
  - **let $r_{M,i}$ be the i th value in L_M and**
  - **compute $\mathrm{z} = \mathrm{H}(r_{M,i}||M)$ using RO.**
  - **Let $(r||M, x_{M,i}, z)$ be the corresponding entry in the RO table.**
  - **Output signature $(r_{M,i}, x_{M,i})$.**

Observation

- **All SIGN queries can be answered!**
- **SIGN queries are answered using hash**

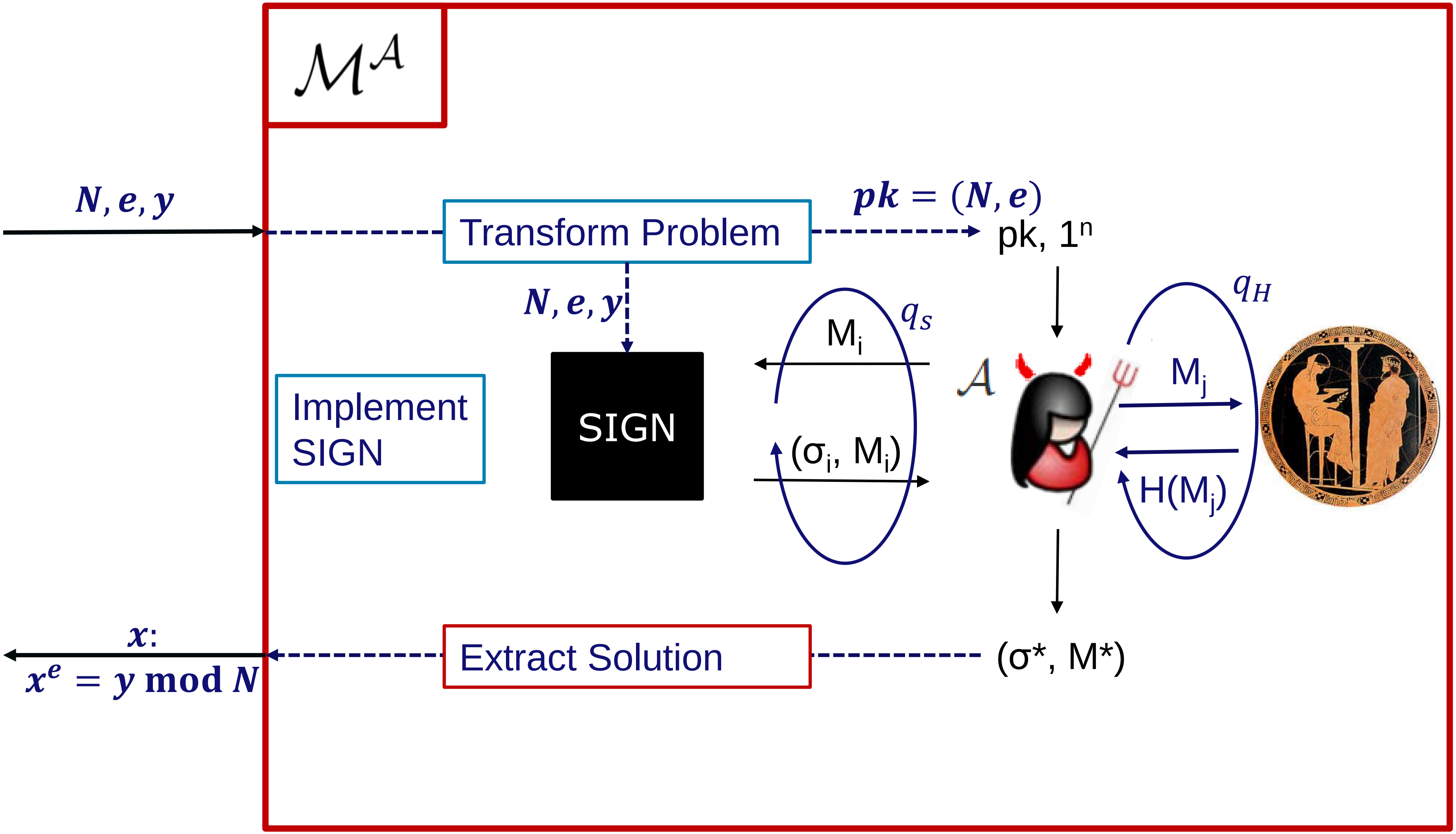
$$\mathrm{H}(r_{M,i}||M) = \mathrm{z} = {x_{M,i}}^{e} \bmod N$$

  - **Signature $(r_{M,i}, x_{M,i})$ known by programming RO**
- **All other hash queries are answered with**

$$\mathrm{H}(r||M) = z = yx^{e} \bmod N$$

  - **Signature not known!**
  - **BUT: Allows to extract solution from forgery!**

Reduction: Extract Solution



Reduction: Extract Solution

- **If A outputs a forgery $(M^*, (r^*, \sigma^*))$:**
  - **If $r^* \in L_{M^*}$ abort.**
  - **Else, let $(r^*||M^*, x, z)$ be the corresponding entry of the table.**
  - **Output $\frac{\sigma^*}{x} \bmod N$.**

- **Note:**

$$\left(\frac{\sigma^*}{x}\right)^e = \frac{{\sigma^*}^e}{x^e} = \frac{H(r^*||M^*)}{x^e} = \frac{yx^e}{x^e} = y \bmod N$$

$$\Longrightarrow \frac{\sigma^*}{x} = \sqrt[e]{y} \bmod N$$

Analysis

- **Transform Problem:**
  - **Succeeds always**
  - **Generates exactly matching distribution**
- **Implement SIGN / RO:**
  - **Succeeds always (we choose r)**
  - **Generates exactly matching distribution:**
    - **RO: Outputs are uniform in $\mathbb{Z}_N^*$**
    - **SIGN: Follows from RO**
- **Extract Solution:**
  - **Succeeds iff A succeeds AND**
  - $r^* \notin L_{M^*} \Rightarrow \mathrm{p} = \Pr[r^* \notin L_{M^*}] = (1 - 2^{-\kappa})^{q_s}$
    Setting $\kappa = \log_2 q_s$: $p \geq \frac{1}{4}$ assuming $q_s \geq 2$

What have we shown?

- **We can turn any forger A against RSA-PFDH with success probability ε into an algorithm M^A that solves the RSA problem with probability $\varepsilon/4$.**

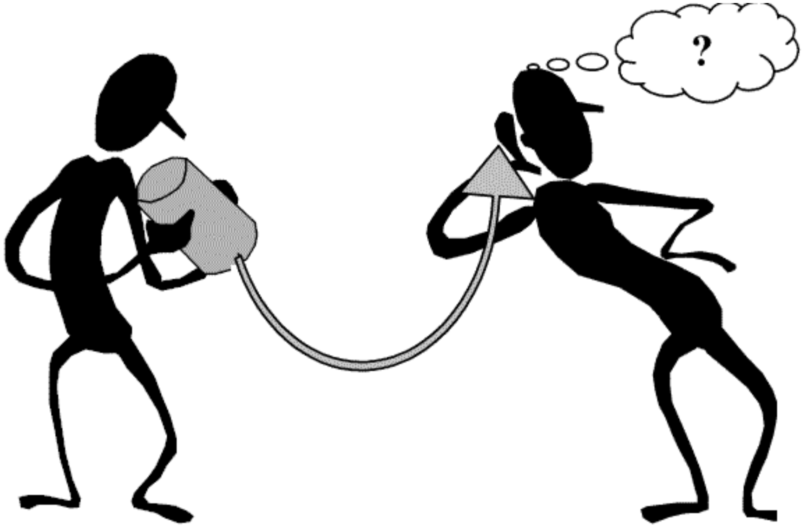
- **In reverse:**
  If there exists no algorithm to solve the RSA problem with probability $\geq \varepsilon$ then there exists no forger against RSA-PFDH that succeeds with probability $\geq 4\varepsilon$.

- **As proof is in ROM we have to add**
  ”... As long as the used hash function behaves like a RO.”

Conclusion

- **Ad Hoc constructions problematic**
  - **Blinding / Index Calculus**
- **Proofs (even in ROM) allow to check construction**
- **There is one standardized RSA Sig with proof**
- **Similar situation for DSA (ROM proof)**

Thank you!
Questions?



Technische Universiteit Eindhoven University of Technology