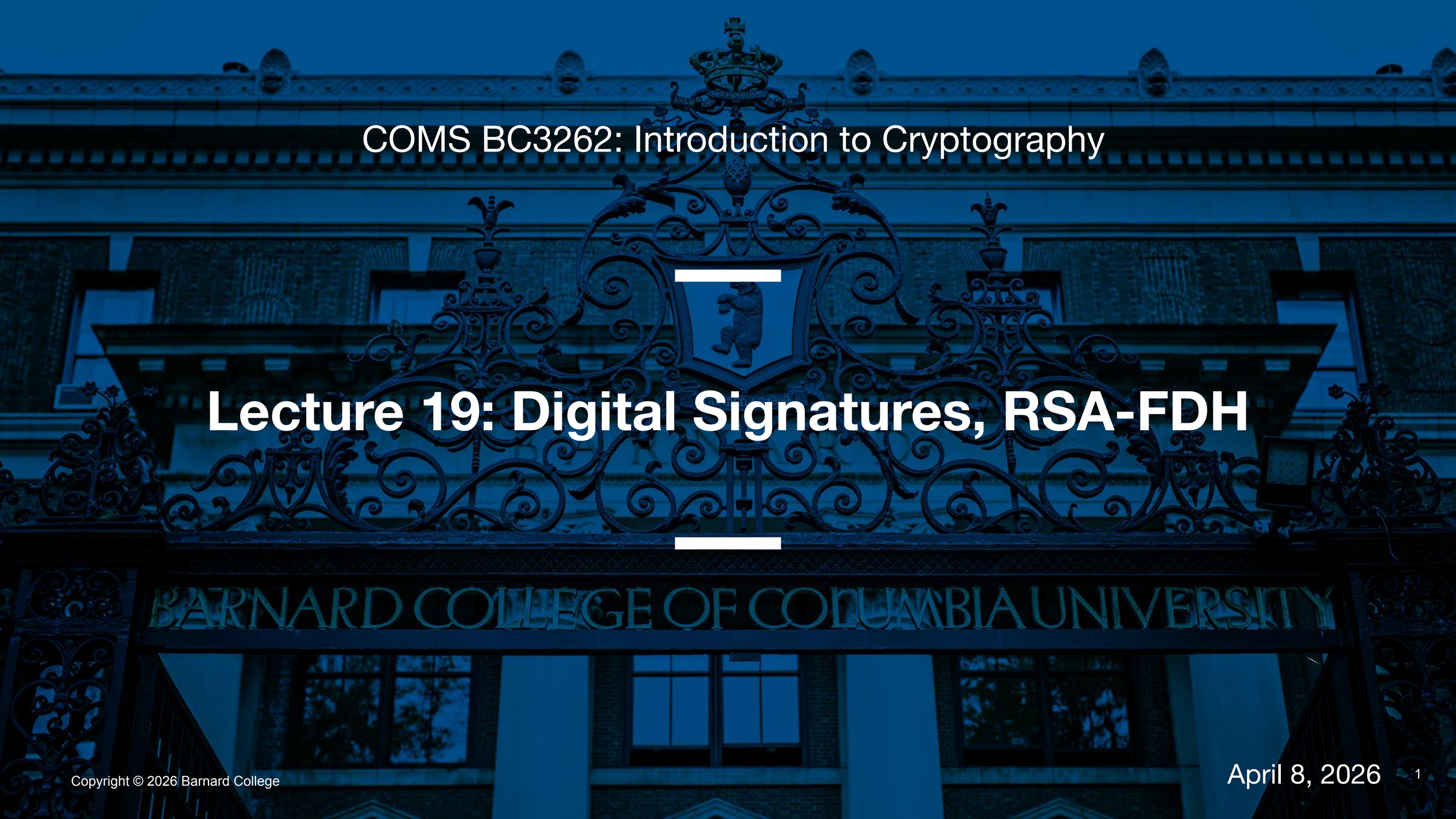


COMS BC3262: Introduction to Cryptography

Lecture 19: Digital Signatures, RSA-FDH

April 8, 2026

Logistics

- **Extra credit opportunity**: Attend Olive's talk Thursday, April 16 at 12pm in Milstein 402

  - Olive Franzese-McLaughlin will be giving a talk titled "Cryptographically Verified AI/ML Audits"

Digital Signatures

Recall: Message Authentication

Alice and Bob want to communicate, but Eve controls the channel.

They want assurance the message received was not modified

In a MAC scheme, Alice and Bob share the same key

Informally, a signature scheme is the “public-key” variant

Recall: Message Authentication Codes

For MACs, the MAC and Verify algorithms using the same key

Security definitions for MACs:

- Unforgeability (produce a valid tag on a new message)
- Strong unforgeability (produce a valid tag that was not given by the oracle for the message)

Digital Signatures

sk

m, σ

vk

$\sigma \leftarrow \mathsf{Sign}(sk, m)$

$b = \mathsf{Verify}(pk, (m, \sigma))$

Private signing key

Public verification key

Digital Signatures

Syntax: Three algorithms (Gen, Sign, Verify)

- **Key generation** algorithm Gen takes input 1^n and outputs a signing key sk and a verification key vk
- **Signing** algorithm Sign takes signing key sk and message $m \in \{0,1\}^*$ and outputs a signature $\sigma \in \{0,1\}^*$
- **Verification** algorithm Verify takes a verification key vk, a message m, a signature σ, and outputs a bit b

$\sigma \leftarrow \text{Sign}_{sk}(m)$ $\quad m$

(m, σ)

$b = \text{Verify}_{vk}(m, t)$

Digital Signatures

Syntax: Three algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$

- **Key generation** algorithm Gen takes input 1^n and outputs a signing key sk and a verification key vk
- **Signing** algorithm Sign takes signing key sk and message $m \in \{0,1\}^*$ and outputs a signature $\sigma \in \{0,1\}^*$
- **Verification** algorithm Verify takes a verification key vk, a message m, a signature σ, and outputs a bit b

Correctness: For every message m it holds that

$$\Pr[\mathsf{Verify}_{vk}(m, \mathsf{Sign}_{sk}(m)) = 1] = 1$$

Digital Signatures

Syntax: Three algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$

- **Key generation** algorithm Gen takes input 1^n and outputs a signing key sk and a verification key vk
- **Signing** algorithm Sign takes signing key sk and message $m \in \{0,1\}^*$ and outputs a signature $\sigma \in \{0,1\}^*$
- **Verification** algorithm Verify takes a verification key vk, a message m, a signature σ, and outputs a bit b

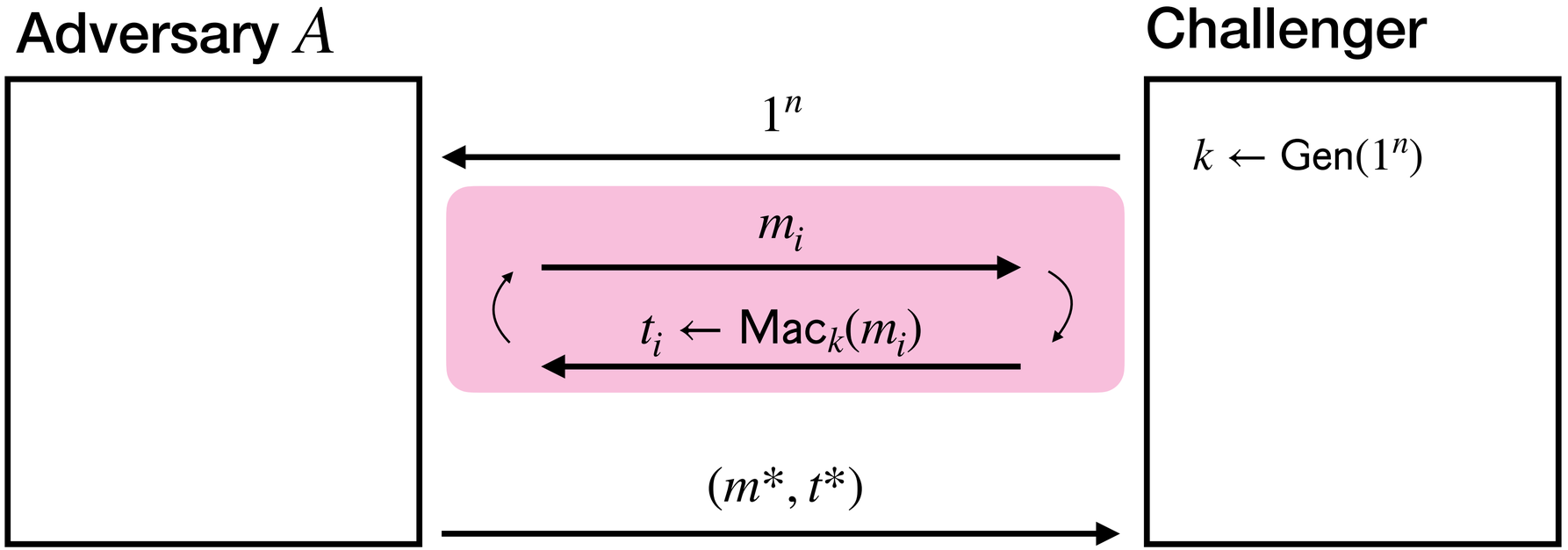
Correctness: For every message m it holds that

$$\Pr[\mathsf{Verify}_{vk}(m, \mathsf{Sign}_{sk}(m)) = 1] = 1$$

Security?

Recall: MAC Security

Let $\Pi = (\text{Gen}, \text{Mac}, \text{Verify})$. We define $\text{MacForge}_{\Pi,\mathcal{A}}(n)$ as follows

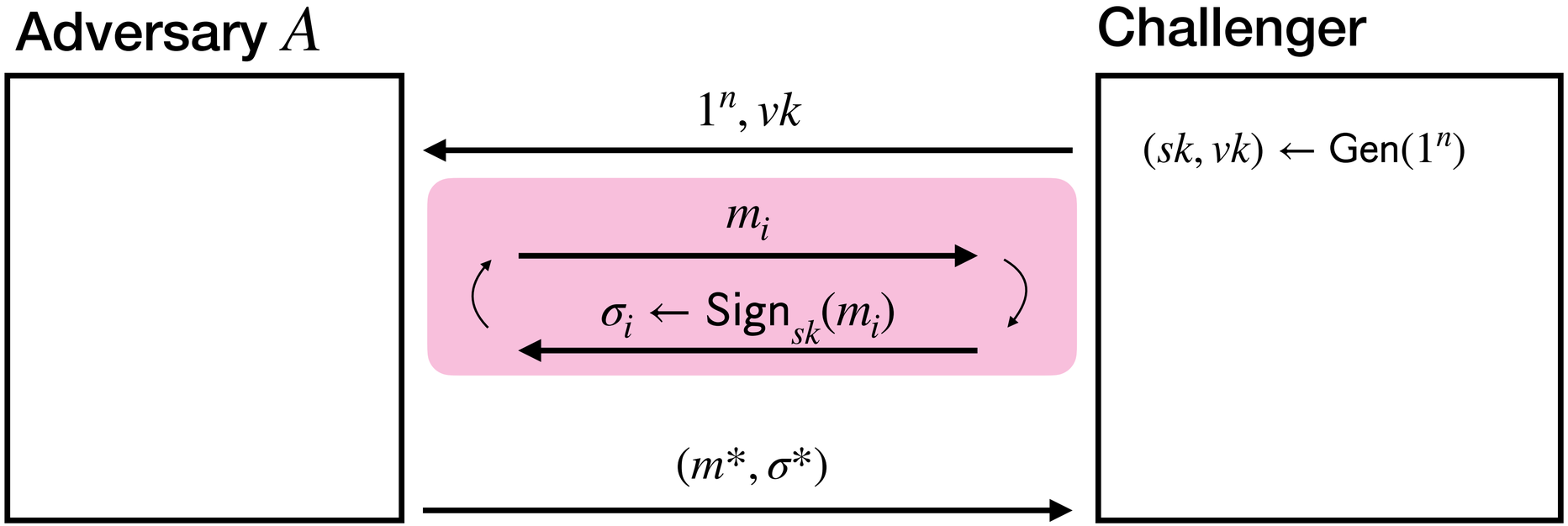


We say the adversary succeeds ($\text{MacForge}_{\Pi,\mathcal{A}}(n) = 1$) if:

1. $\text{Verify}_k(m^*, t^*) = 1$
2. $m^* \neq m_i$ for all queried m_i

Security of Signatures

Let $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$. We define $\text{SigForge}_{\Pi,\mathcal{A}}(n)$ as follows

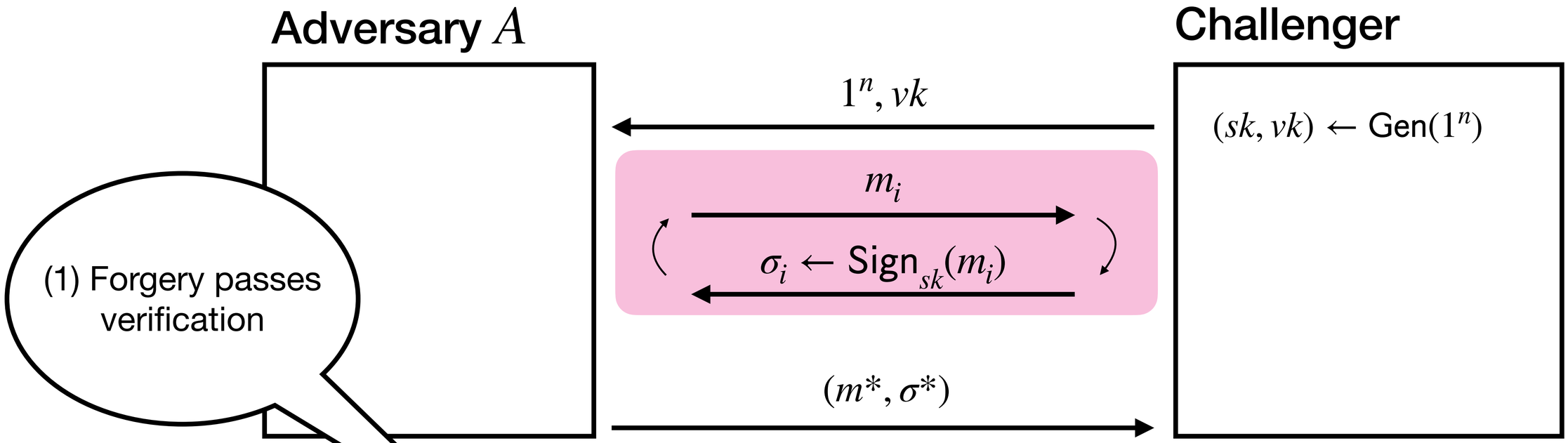


We say the adversary succeeds ($\text{SigForge}_{\Pi,\mathcal{A}}(n) = 1$) if:

1. $\text{Verify}_{vk}(m^*, \sigma^*) = 1$
2. $m^* \neq m_i$ for all queried m_i

Security of Signatures

Let $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$. We define $\text{SigForge}_{\Pi,\mathcal{A}}(n)$ as follows

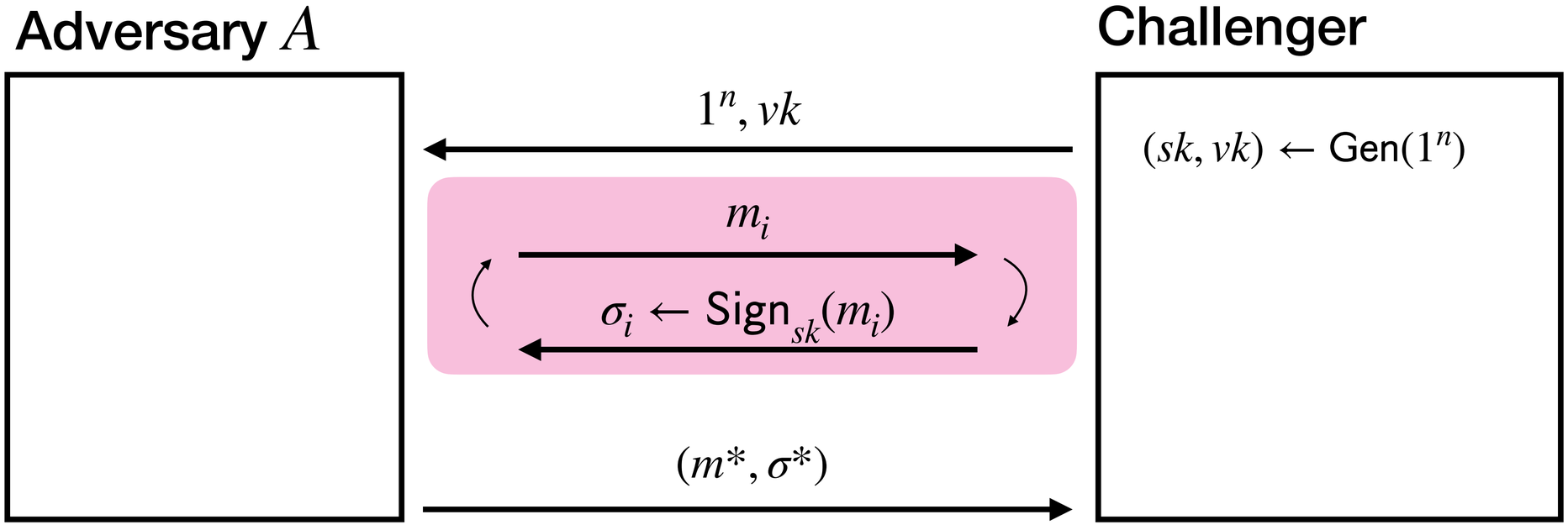


We say the adversary succeeds (SigForge

1. $\text{Verify}_{vk}(m^*, \sigma^*) = 1$
2. $m^* \neq m_i$ for all queried m_i

Security of Signatures

Let $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$. We define $\text{SigForge}_{\Pi,\mathcal{A}}(n)$ as follows



We say the adversary succeeds ($\text{SigForge}_{\Pi,\mathcal{A}}(n) = 1$) if:

1. $\text{Verify}_{vk}(m^*, \sigma^*) = 1$
2. $m^* \neq m_i$ for all queried m_i

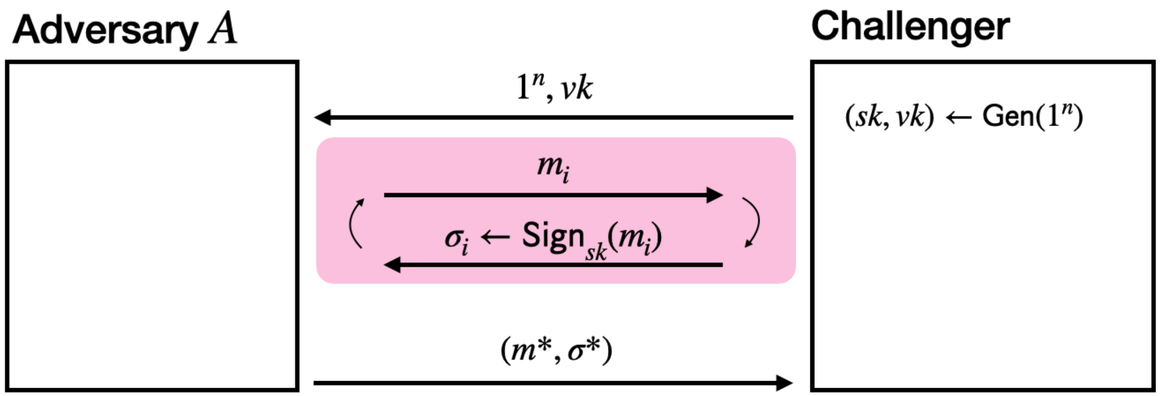
Similar to MACs, we can define a “strong” version of security that modifies (2) to only require (m^*, σ^*) was not exactly given by the oracle

Security of Signatures

Definition:

A **signature scheme** Π is existentially unforgeable under an adaptive chosen-message attack (EUF-CMA) if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\text{SigForge}_{\Pi,\mathcal{A}} = 1] \leq \text{negl}(\cdot)$$



$\mathcal{Q}$ = set of all messages queried by $\mathcal{A}$

$$\text{SigForge}_{\Pi,\mathcal{A}} = \begin{cases} 1 & \text{Verify}_{vk}(m^*, \sigma^*) = 1 \wedge m^* \notin \mathcal{Q} \\ 0 & \text{otherwise} \end{cases}$$

Signatures vs MACs

Signatures

- n users require n secret keys
- One signature can be verified by all
- Public verifiability
- Transferable
- Bob can prove that Alice sent the signed message (non-repudiation)
- Less efficient (public-key operations)

MACs

- n users require $\approx n^2$ secret keys
- Different MACs for different users
- Private verifiability
- Non-transferable
- Bob knows that Alice sent the signed message but can't prove it
- More efficient (2-3 orders of magnitude)

Hash-and-Sign (Domain Extension)

Let $\widehat{\Pi} = (\widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}})$ be a fixed-length signature scheme, and let $\Phi = (\mathsf{Gen}_H, H)$ be a keyed hash function.

We can construct a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ for arbitrary-length messages as follows.

- **Key generation**: On input 1^n, sample $(\widehat{sk}, \widehat{vk}) \leftarrow \widehat{\mathsf{Gen}}(1^n)$ and $s \leftarrow \mathsf{Gen}_H(1^n)$ and output $sk = (\widehat{sk}, s)$ and $vk = (\widehat{vk}, s)$.
- **Signing**: On input $(\widehat{sk}, s)$ and $m \in \{0,1\}^*$, output $\sigma = \widehat{\mathsf{Sign}}_{\widehat{sk}}(H^s(m))$
- **Verification**: On input $(\widehat{vk}, s)$, $m \in \{0,1\}^*$, and $\sigma \in \{0,1\}^*$, output $\widehat{\mathsf{Verify}}_{\widehat{vk}}(H^s(m), \sigma)$

Hash-and-Sign (Domain Extension)

Let $\widehat{\Pi} = (\widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}})$ be a fixed-length signature scheme, and let $\Phi = (\mathsf{Gen}_H, H)$ be a keyed hash function.

We can construct a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ for arbitrary-length messages as follows.

- **Key generation**: On input 1^n, sample $(\widehat{sk}, \widehat{vk}) \leftarrow \widehat{\mathsf{Gen}}(1^n)$ and $s \leftarrow \mathsf{Gen}_H(1^n)$ and output $sk = (\widehat{sk}, s)$ and $vk = (\widehat{vk}, s)$.
- **Signing**: On input $(\widehat{sk}, s)$ and $m \in \{0,1\}^*$, output $\sigma = \widehat{\mathsf{Sign}}_{\widehat{sk}}(H^s(m))$
- **Verification**: On input $(\widehat{vk}, s)$, $m \in \{0,1\}^*$, and $\sigma \in \{0,1\}^*$, output $\widehat{\mathsf{Verify}}_{\widehat{vk}}(H^s(m), \sigma)$

Theorem:
If $\widehat{\Pi}$ is a secure fixed-length signature scheme and Φ is collision resistant, then Π is a secure variable-length signature scheme

RSA Signatures

Textbook RSA Signatures

Let GenRSA be a PPT algorithm that on input 1^n outputs (N, e, d) where $N = pq$ and p, q are n-bit primes, $\gcd(e, \phi(N)) = 1$ and $ed = 1 \mod \phi(N)$

Define (Gen, Sign, Verify) as follows:

- Gen(1^n): Run $(N, e, d) \leftarrow$ GenRSA(1^n) and set $vk = (e, N)$ and $sk = (d, N)$
- $\text{Sign}_{sk}(m)$: Output $\sigma = [m^d \mod N]$
- $\text{Verify}_{vk}(m, \sigma)$: If $m = [\sigma^e \mod N]$ output 1. Otherwise output 0

Textbook RSA Signatures

Let GenRSA be a PPT algorithm that on input 1^n outputs (N, e, d) where $N = pq$ and p, q are n-bit primes, $\gcd(e, \phi(N)) = 1$ and $ed = 1 \mod \phi(N)$

Define (Gen, Sign, Verify) as follows:

- Gen(1^n): Run $(N, e, d) \leftarrow$ GenRSA(1^n) and set $vk = (e, N)$ and $sk = (d, N)$
- Sign$_{sk}(m)$: Output $\sigma = [m^d \mod N]$
- Verify$_{vk}(m, \sigma)$: If $m = [\sigma^e \mod N]$ output 1. Otherwise output 0

Correctness:

$$\text{Verify}_{vk}(\text{Sign}_{sk}(m)) = [(m^d)^e \mod N] = [m^{ed \mod \phi(N)} \mod N] = m$$

Textbook RSA Signatures

Let GenRSA be a PPT algorithm that on input 1^n outputs (N, e, d) where $N = pq$ and p, q are n-bit primes, $\gcd(e, \phi(N)) = 1$ and $ed = 1 \mod \phi(N)$

Define (Gen, Sign, Verify) as follows:

- Gen(1^n): Run $(N, e, d) \leftarrow$ GenRSA(1^n) and set $vk = (e, N)$ and $sk = (d, N)$
- Sign$_{sk}(m)$: Output $\sigma = [m^d \mod N]$
- Verify$_{vk}(m, \sigma)$: If $m = [\sigma^e \mod N]$ output 1. Otherwise output 0

Security:

Intuitively, under the RSA assumption, signing a given m requires knowing d

Textbook RSA Signatures

Let GenRSA be a PPT algorithm that on input 1^n outputs (N, e, d) where $N = pq$ and p, q are n-bit primes, $\gcd(e, \phi(N)) = 1$ and $ed = 1 \mod \phi(N)$

Define (Gen, Sign, Verify) as follows:

- Gen(1^n): Run $(N, e, d) \leftarrow$ GenRSA(1^n) and set $vk = (e, N)$ and $sk = (d, N)$
- $\text{Sign}_{sk}(m)$: Output $\sigma = [m^d \mod N]$
- $\text{Verify}_{vk}(m, \sigma)$: If $m = [\sigma^e \mod N]$ output 1. Otherwise output 0

Security:

Intuitively, under the RSA assumption, signing a given m requires knowing d

… But it's completely insecure

Attacks on Textbook RSA Signatures

No message attack:

- Given $vk = (e, N)$
- Adversary can choose $\sigma \in \mathbb{Z}_N^*$ and compute $m = [\sigma^e \bmod N]$
  - Notice that (m, σ) is a valid signature pair!

Forge on an arbitrary message m:

- Let $m \in \mathbb{Z}_N^*$ be $m = m_1 \cdot m_2 \bmod N$
- If we have signatures σ_1, σ_2 on messages m_1, m_2, the product $\sigma_1 \cdot \sigma_2$ is a valid signature on m

Attacks on Textbook RSA Signatures

What if we used hash-and-sign?

- Consider $H^s : \{0,1\}^* \to \mathbb{Z}_N^*$ and $\sigma = [H^s(m)^d \bmod N]$
- Does the no message attack still apply?
  - No, given σ you can compute $H^s(m) = \sigma^e \bmod N$ but can't invert H^s
- Does the second attack still apply?
  - No, this relied on the multiplicative homomorphism of RSA
  - Hash broke the algebraic structure
- So is it secure?

RSA-FDH Signatures

Full Domain Hash (FDH): The output of H is all of $\mathbb{Z}_N^*$

Let GenRSA be a PPT algorithm that on input 1^n outputs (N, e, d) where $N = pq$ and p, q are n-bit primes, $\gcd(e, \phi(N)) = 1$ and $ed = 1 \mod \phi(N)$. Let $H : \{0,1\}^* \to \mathbb{Z}_N^*$

Define $\Pi_{\text{RSA-FDH}} = (\text{Gen}, \text{Sign}, \text{Verify})$ as follows:

- $\text{Gen}(1^n)$: Run $(N, e, d) \leftarrow \text{GenRSA}(1^n)$ and set $vk = (e, N, H)$ and $sk = (d, N, H)$
- $\text{Sign}_{sk}(m)$: Output $\sigma = [H(m)^d \mod N]$
- $\text{Verify}_{vk}(m, \sigma)$: If $H(m) = [\sigma^e \mod N]$ output 1. Otherwise output 0

RSA-FDH Signatures

Full Domain Hash (FDH): The output of H is all of $\mathbb{Z}_N^*$

Let GenRSA be a PPT algorithm that on input 1^n outputs (N, e, d) where $N = pq$ and p, q are n-bit primes, $\gcd(e, \phi(N)) = 1$ and $ed = 1 \mod \phi(N)$. Let $H : \{0,1\}^* \to \mathbb{Z}_N^*$

Define $\Pi_{\text{RSA-FDH}} = (\text{Gen}, \text{Sign}, \text{Verify})$ as follows:

- $\text{Gen}(1^n)$: Run $(N, e, d) \leftarrow \text{GenRSA}(1^n)$ and set $vk = (e, N, H)$ and $sk = (d, N, H)$
- $\text{Sign}_{sk}(m)$: Output $\sigma = [H(m)^d \mod N]$
- $\text{Verify}_{vk}(m, \sigma)$: If $H(m) = [\sigma^e \mod N]$ output 1. Otherwise output 0

Theorem: If the RSA assumption holds relative to GenRSA and H is a random oracle, then $\Pi_{\text{RSA-FDH}}$ is EUF-CMA secure

RSA-FDH Signatures

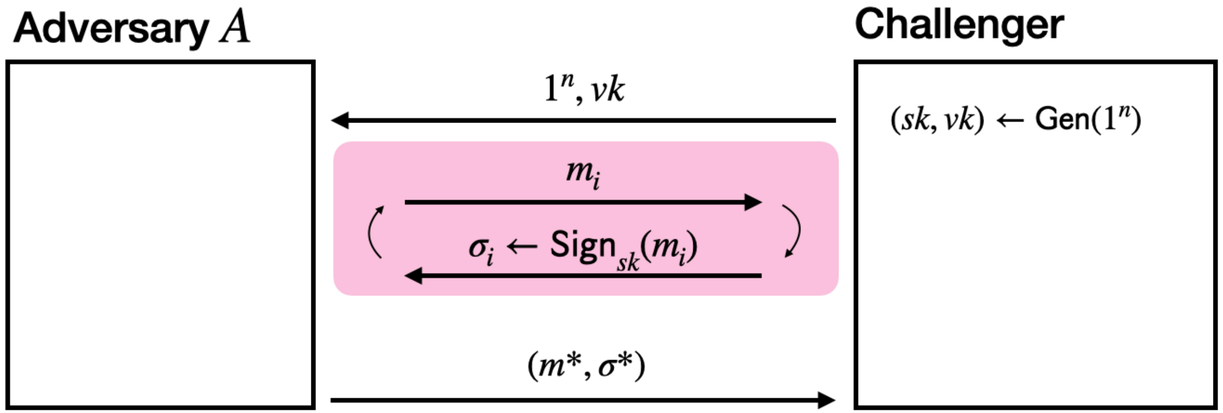
Theorem: If the RSA assumption holds relative to GenRSA and H is a random oracle, then $\Pi_{\text{RSA-FDH}}$ is EUF-CMA secure

Proof idea:

Let A be a PPT adversary against RSA-FDH.

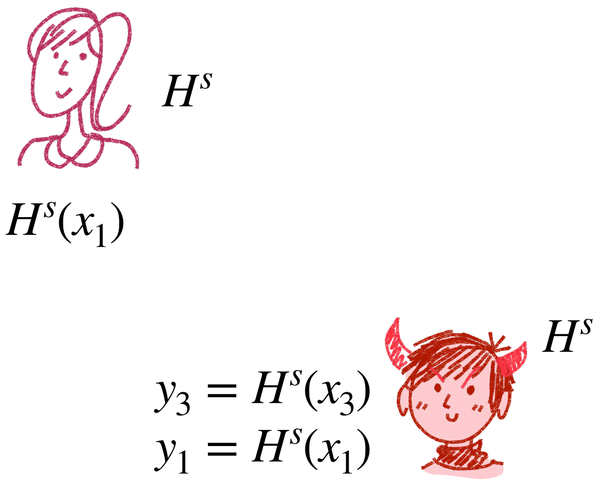
In the signature game:

- A can query $H(\cdot)$ on any input m and get $H(m) \in \mathbb{Z}_N^*$
- A can query $\text{Sign}_{sk}(\cdot)$ on any m and get $\sigma = [H(m)^d \bmod N]$
- A outputs (m^*, σ^*) and wins if A didn't query m^* to $\text{Sign}_{sk}(\cdot)$ and $H(m^*) = [(\sigma^*)^e \bmod N]$

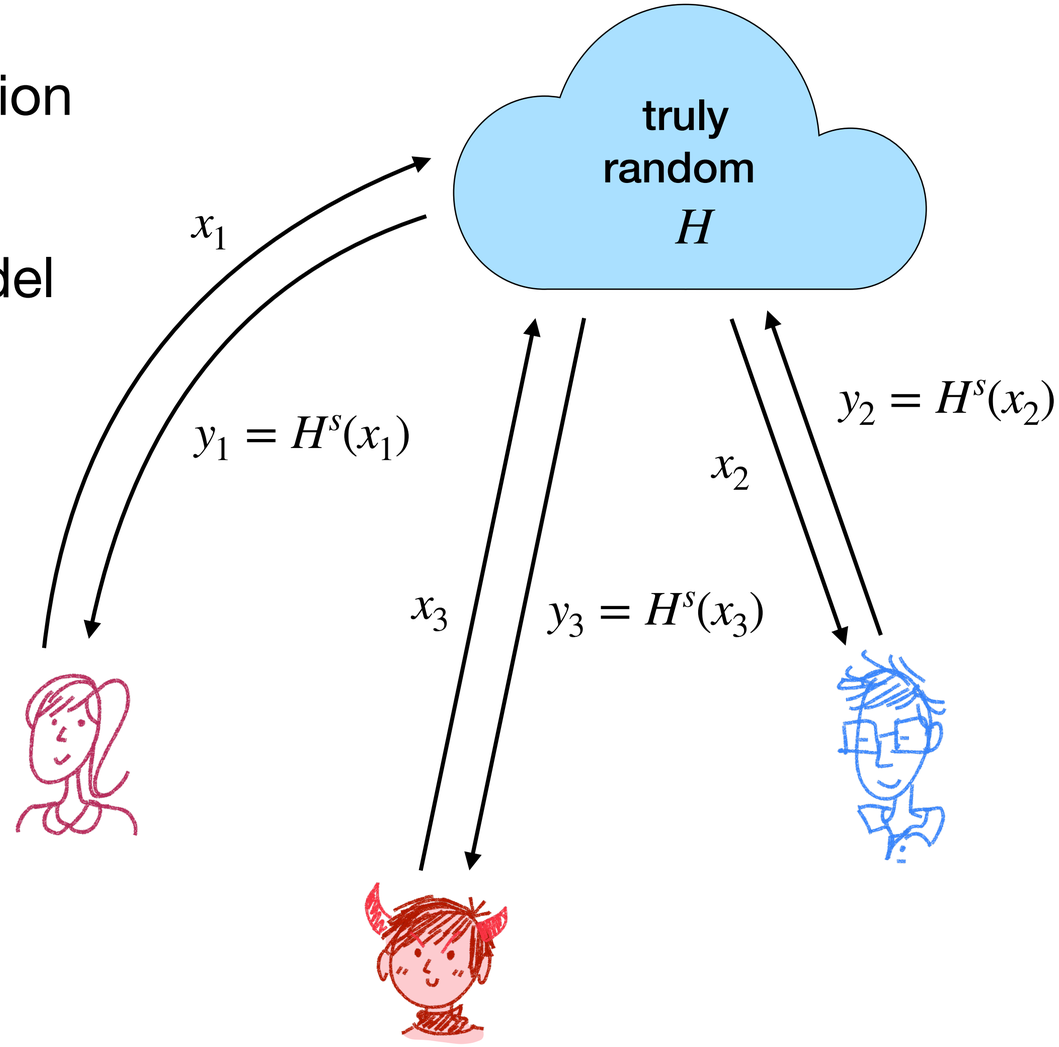


Recall: The Random Oracle Model

1. Start with a protocol Π that uses a hash function
   - Every party can locally compute H
2. Prove security of Π' in the random oracle model
   - Use an oracle to a random function instead of H
3. Hope for the best…



The standard model

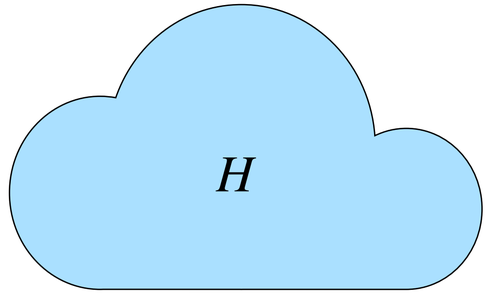


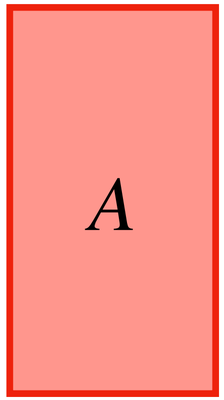
The random oracle model

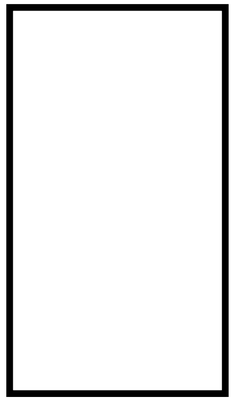
The Random Oracle Model

- The random oracle satisfies two properties:
  - If x has not been queried to H, then $H(x)$ is uniformly distributed
  - If x has been queried before, then the result is consistent
- In a security proof, the reduction simulates the random oracle towards A
  - **Extractability**: When A queries x to H, the reduction can see the query and learn x
  - **Programmability**: The reduction can set $H(x)$ to any value of its choice, as long as it looks uniformly distributed to A

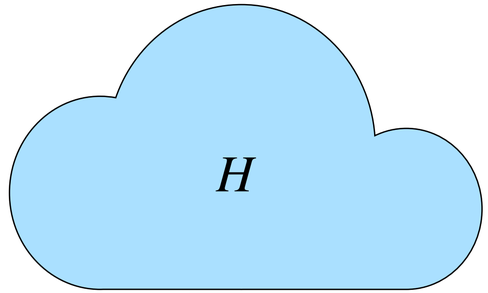
RSA-FDH Signatures



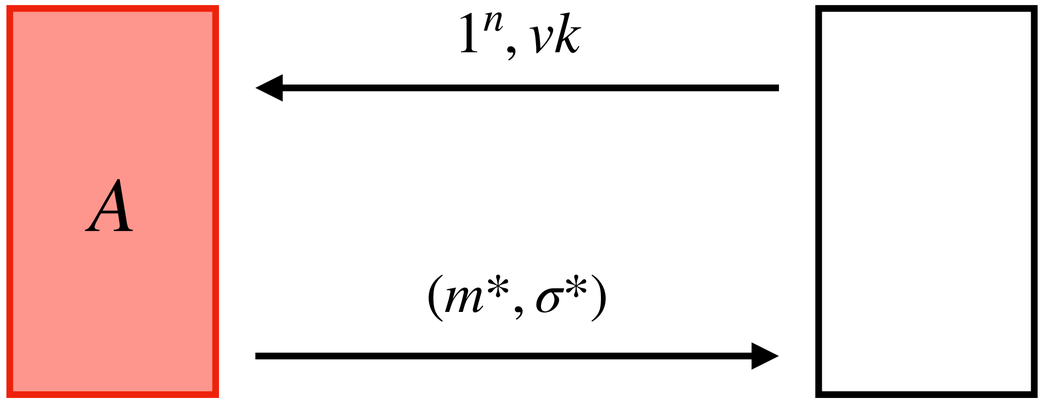




RSA-FDH Signatures



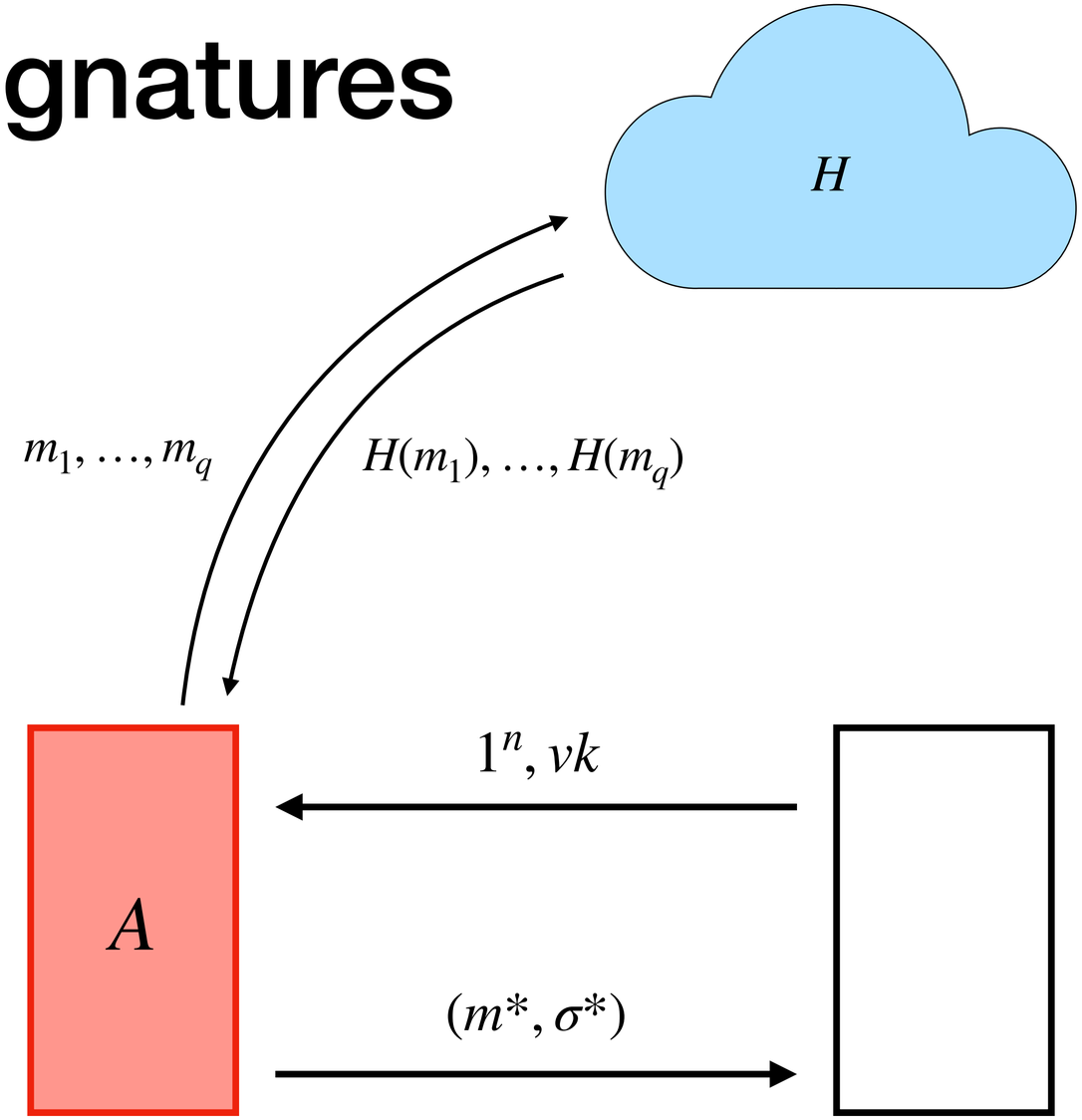
Toy example: Say A never queries
$\text{Sign}_{sk}(\cdot)$



RSA-FDH Signatures

Toy example: Say A never queries $\mathsf{Sign}_{sk}(\cdot)$

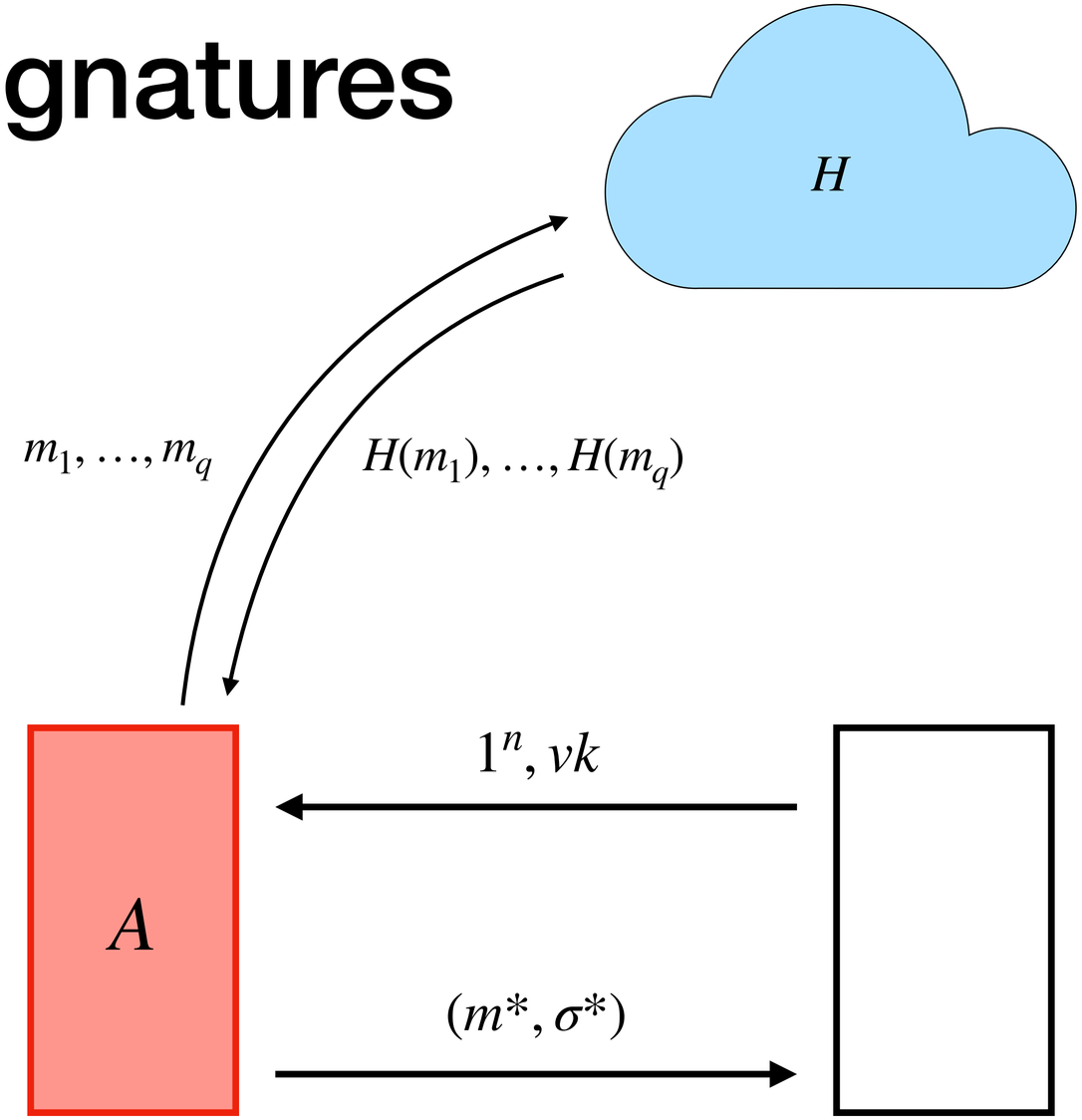
- Without loss of generality, assume that A makes q distinct queries to H, denoted by $m_1, \ldots, m_q$



RSA-FDH Signatures

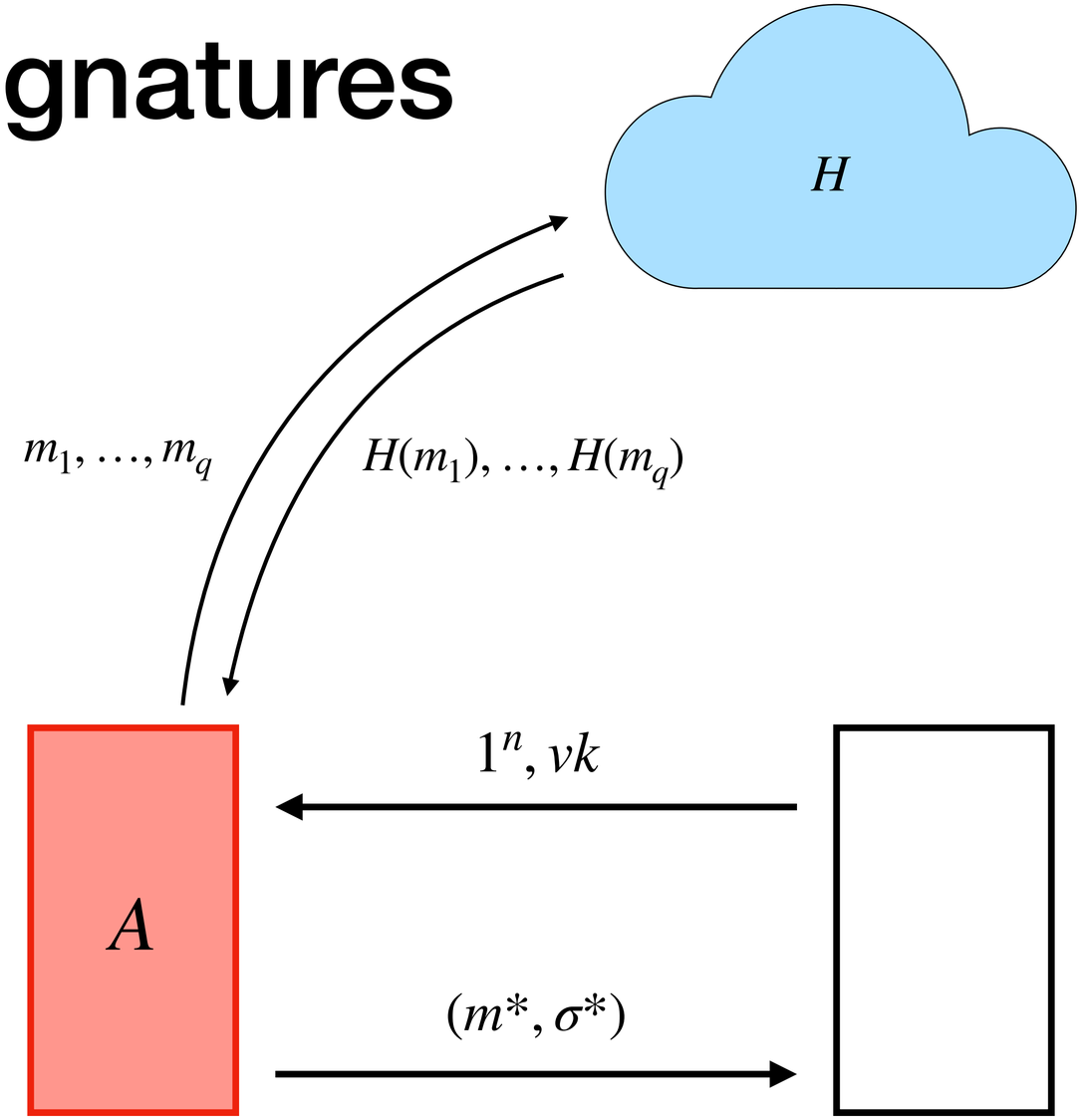
Toy example: Say A never queries $\mathsf{Sign}_{sk}(\cdot)$

- Without loss of generality, assume that A makes q distinct queries to H, denoted by $m_1, \ldots, m_q$
- At the end of the experiment, A outputs its forgery (m^*, σ^*)
  - w.l.o.g. we can assume m^* was one of the queries to H



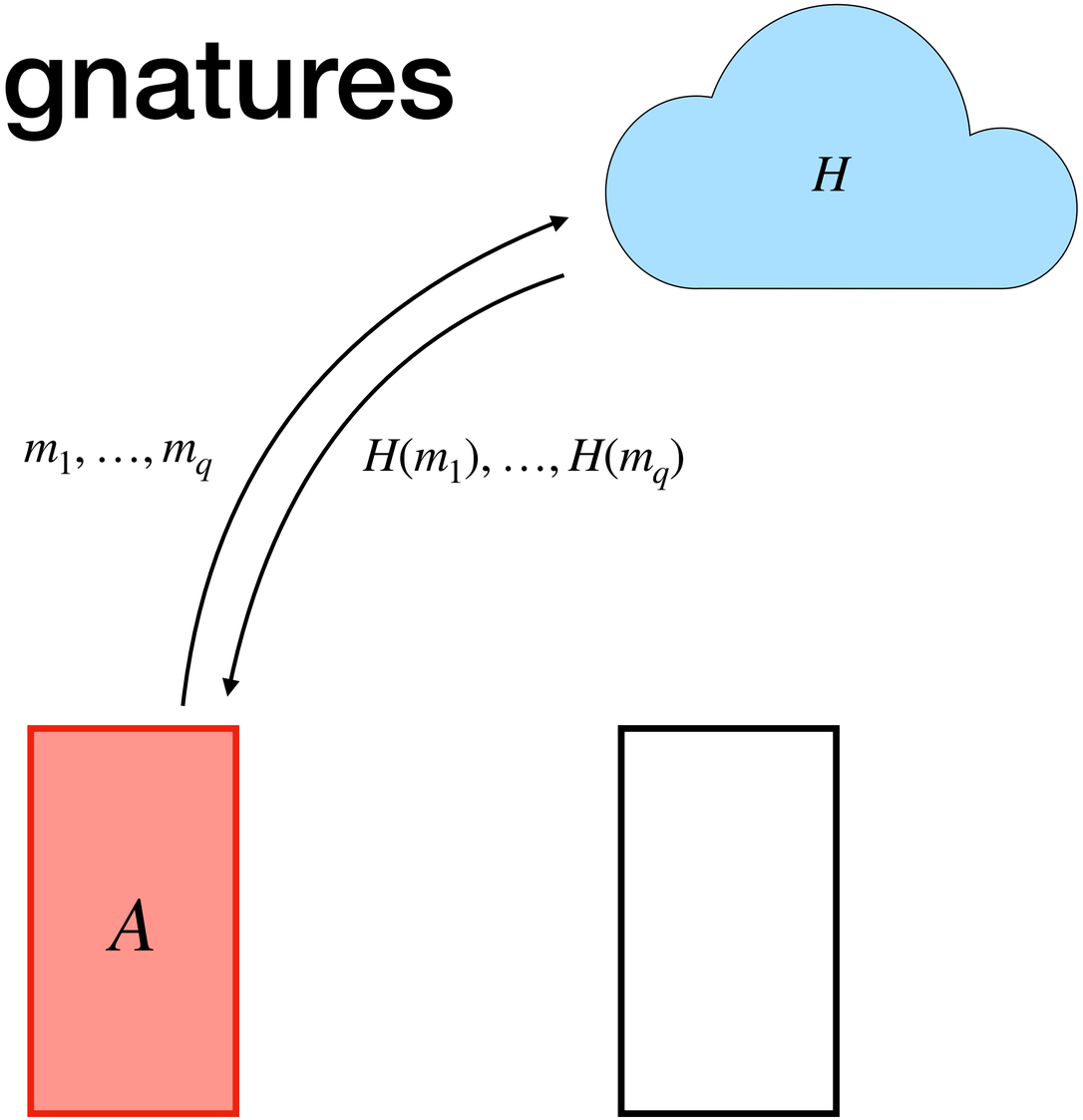
RSA-FDH Signatures

Proof strategy: Our reduction will guess which message m_j will be m^* and use m_j to break the RSA assumption



RSA-FDH Signatures

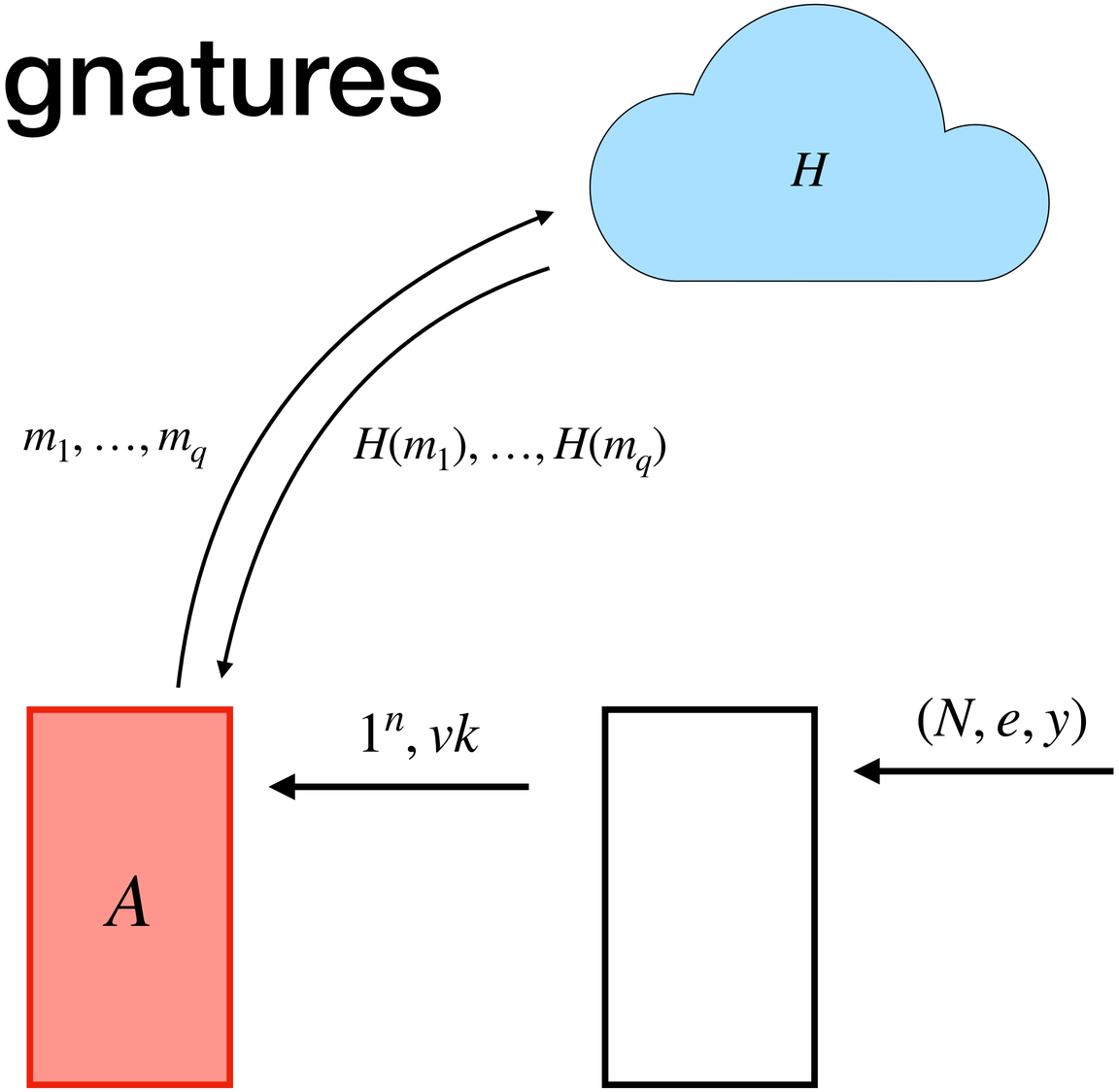
Proof strategy: Our reduction will guess which message m_j will be m^* and use m_j to break the RSA assumption



RSA-FDH Signatures

Proof strategy: Our reduction will guess which message m_j will be m^* and use m_j to break the RSA assumption

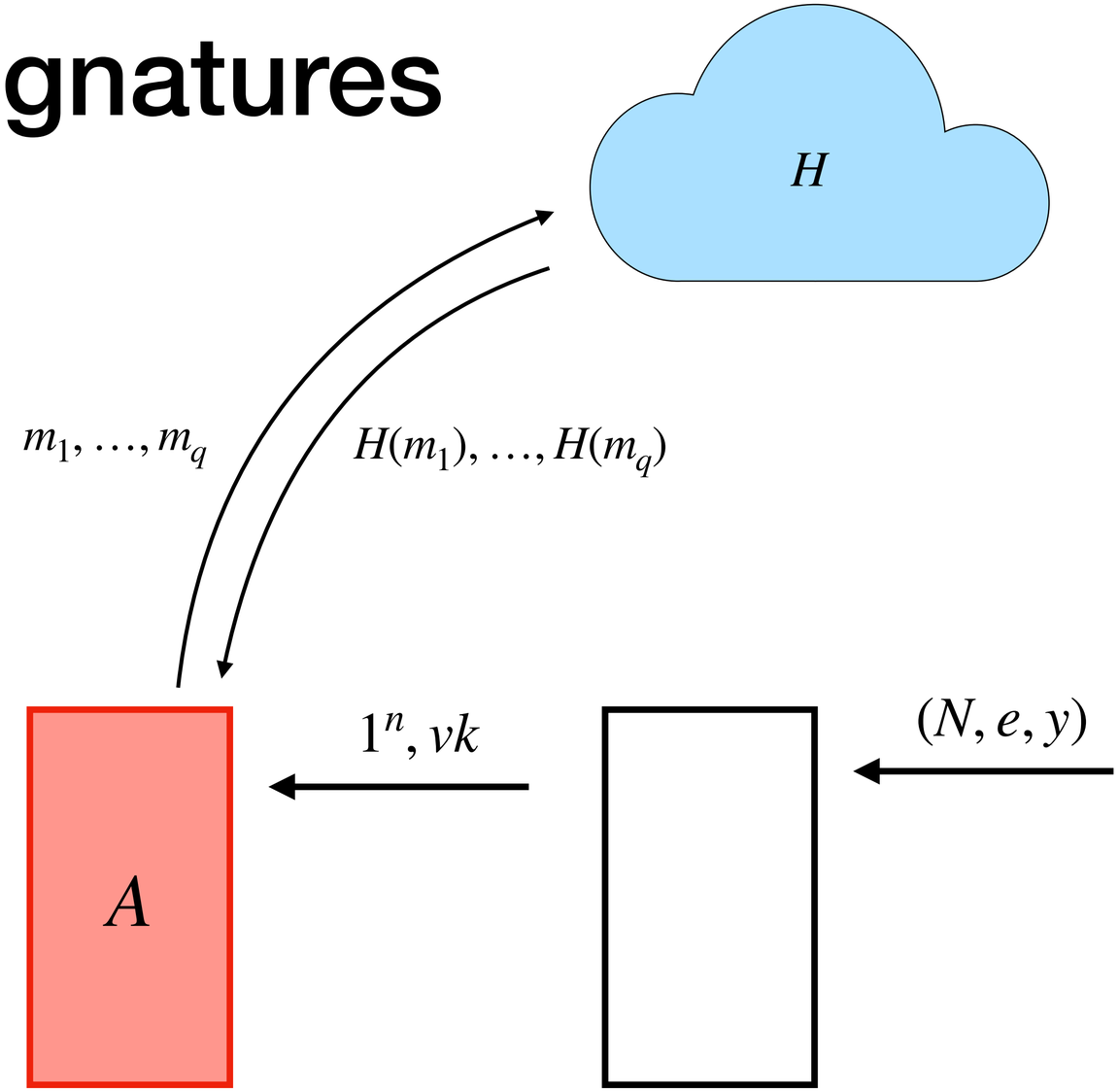
- Reduction gets instance (N, e, y) and gives $vk = (e, N)$ to A



RSA-FDH Signatures

Proof strategy: Our reduction will guess which message m_j will be m^* and use m_j to break the RSA assumption

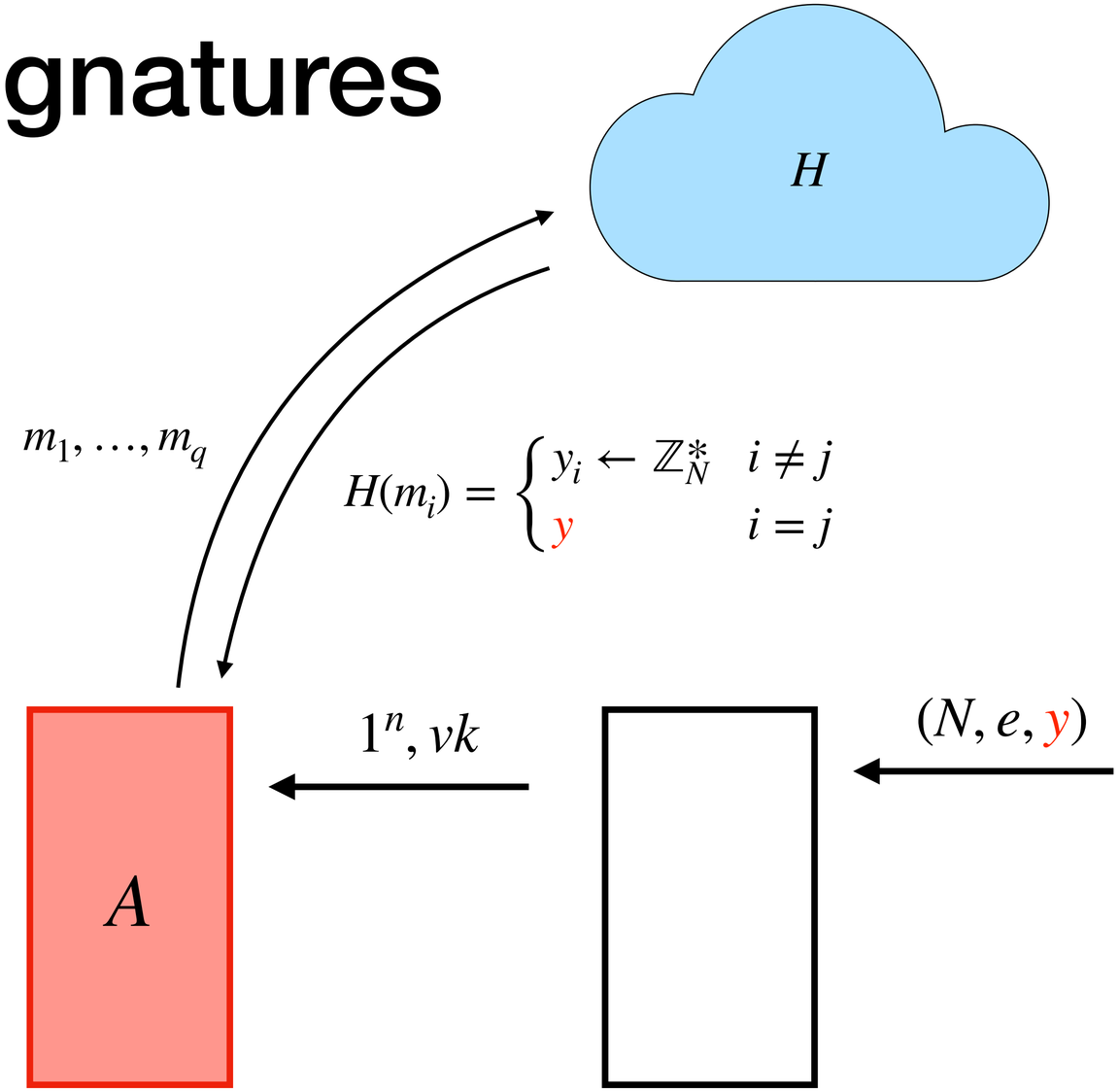
- Reduction gets instance (N, e, y) and gives $vk = (e, N)$ to A
- Reduction sets $H(m_j) = y$



RSA-FDH Signatures

Proof strategy: Our reduction will guess which message m_j will be m^* and use m_j to break the RSA assumption

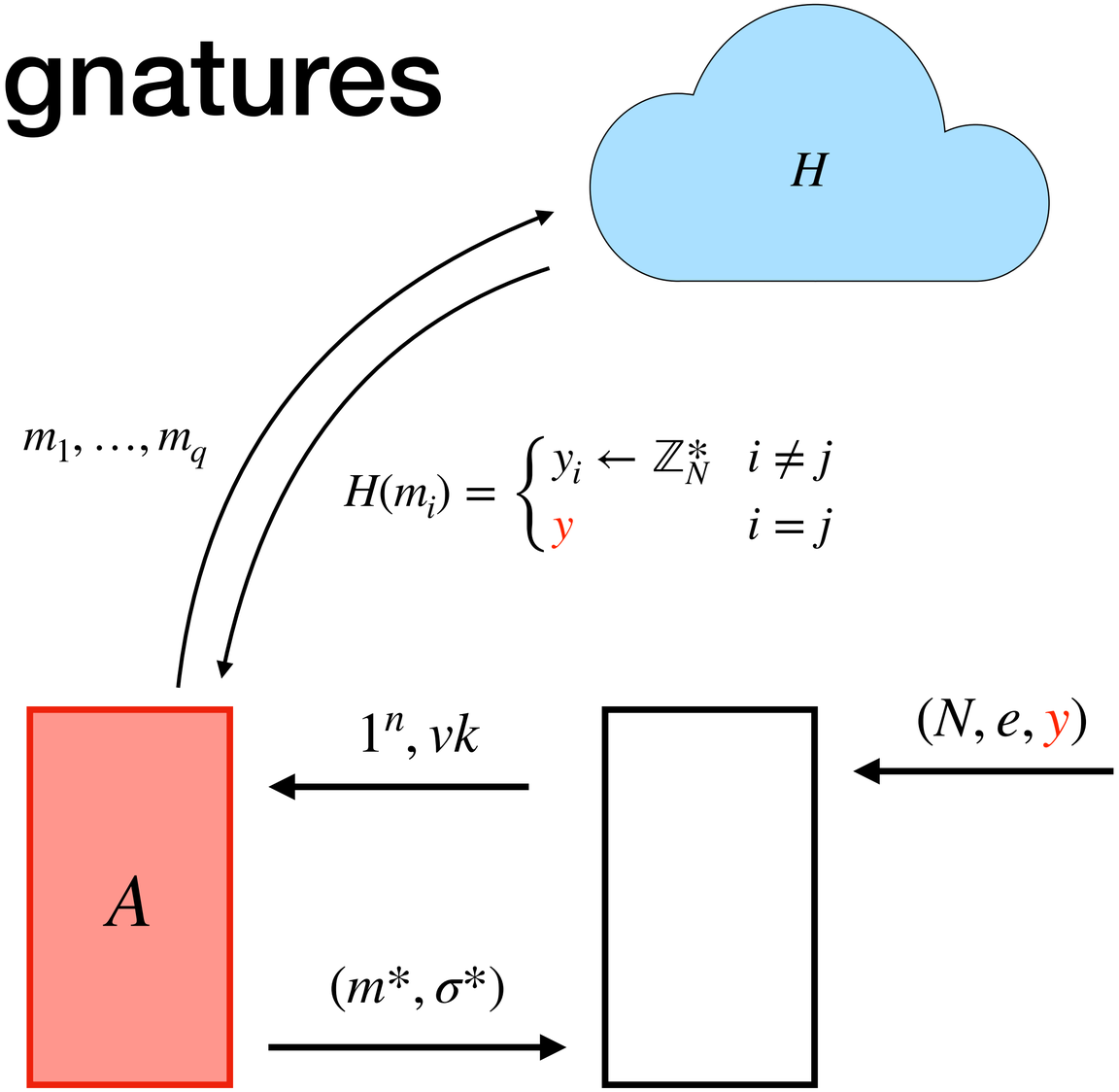
- Reduction gets instance (N, e, y) and gives $vk = (e, N)$ to A
- Reduction sets $H(m_j) = y$



RSA-FDH Signatures

Proof strategy: Our reduction will guess which message m_j will be m^* and use m_j to break the RSA assumption

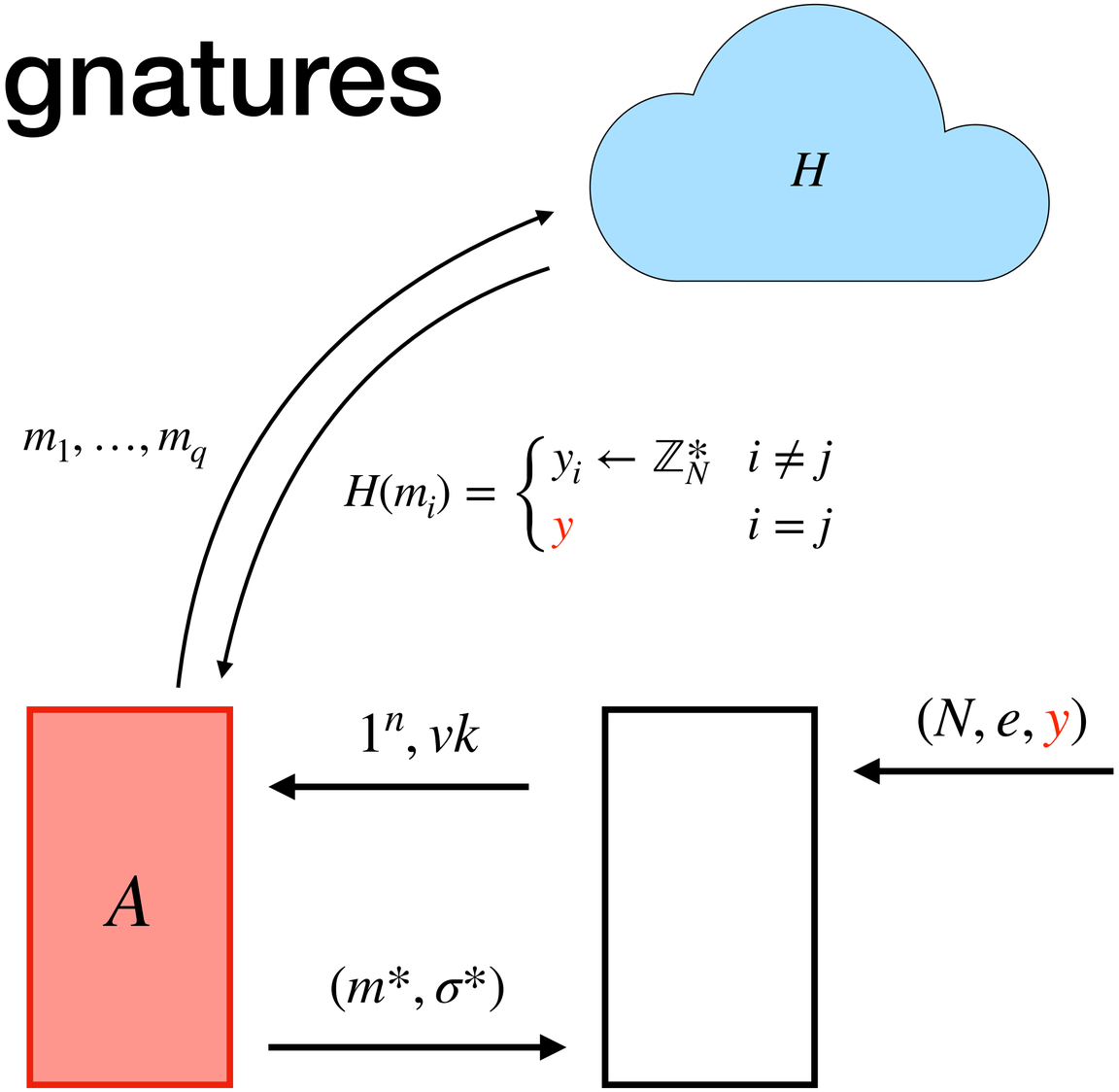
- Reduction gets instance (N, e, y) and gives $vk = (e, N)$ to A
- Reduction sets $H(m_j) = y$



RSA-FDH Signatures

Proof strategy: Our reduction will guess which message m_j will be m^* and use m_j to break the RSA assumption

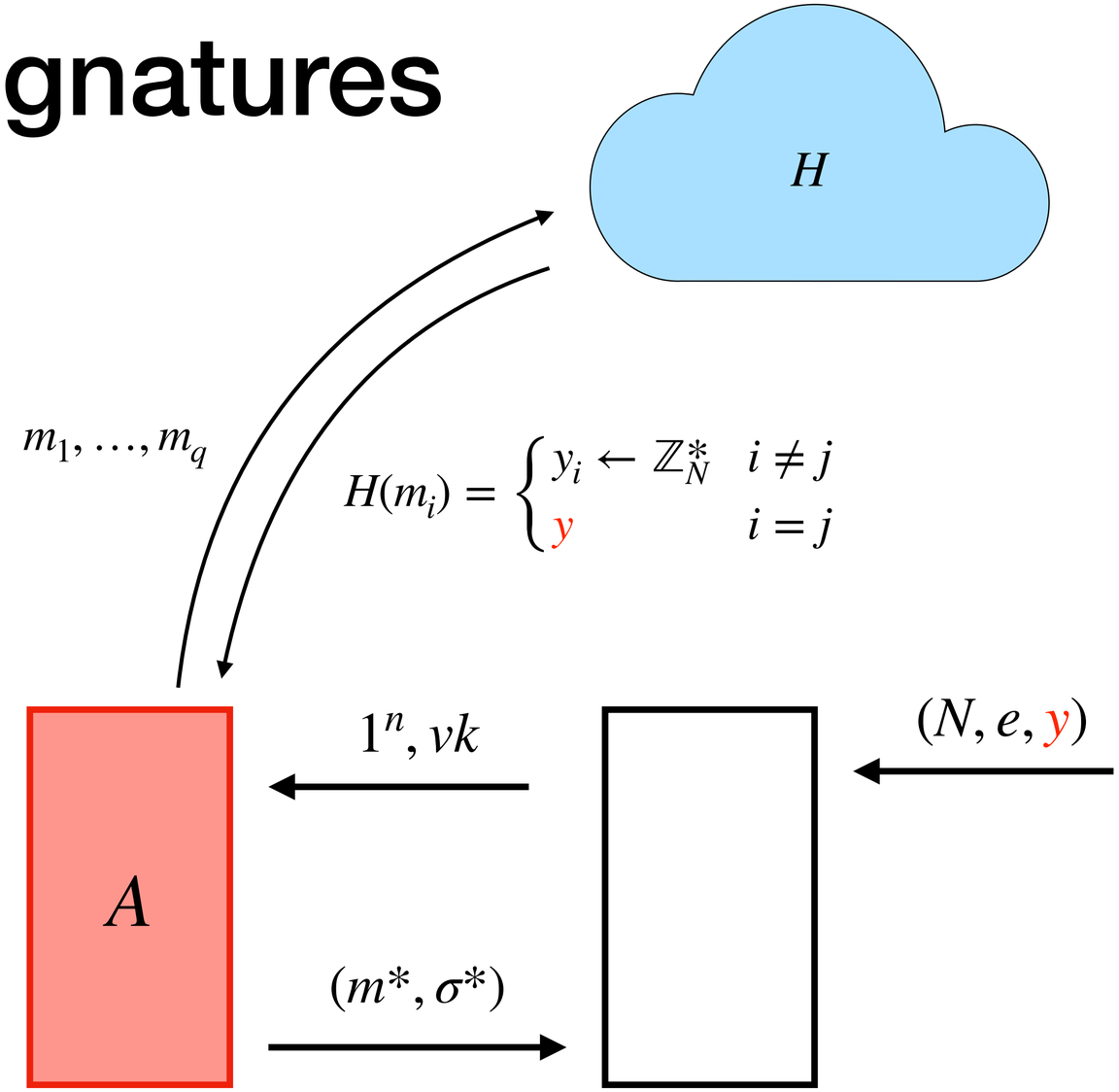
- Reduction gets instance (N, e, y) and gives $vk = (e, N)$ to A
- Reduction sets $H(m_j) = y$
- A valid forgery on m^* is $\sigma^* = H(m^*)^d$



RSA-FDH Signatures

Proof strategy: Our reduction will guess which message m_j will be m^* and use m_j to break the RSA assumption

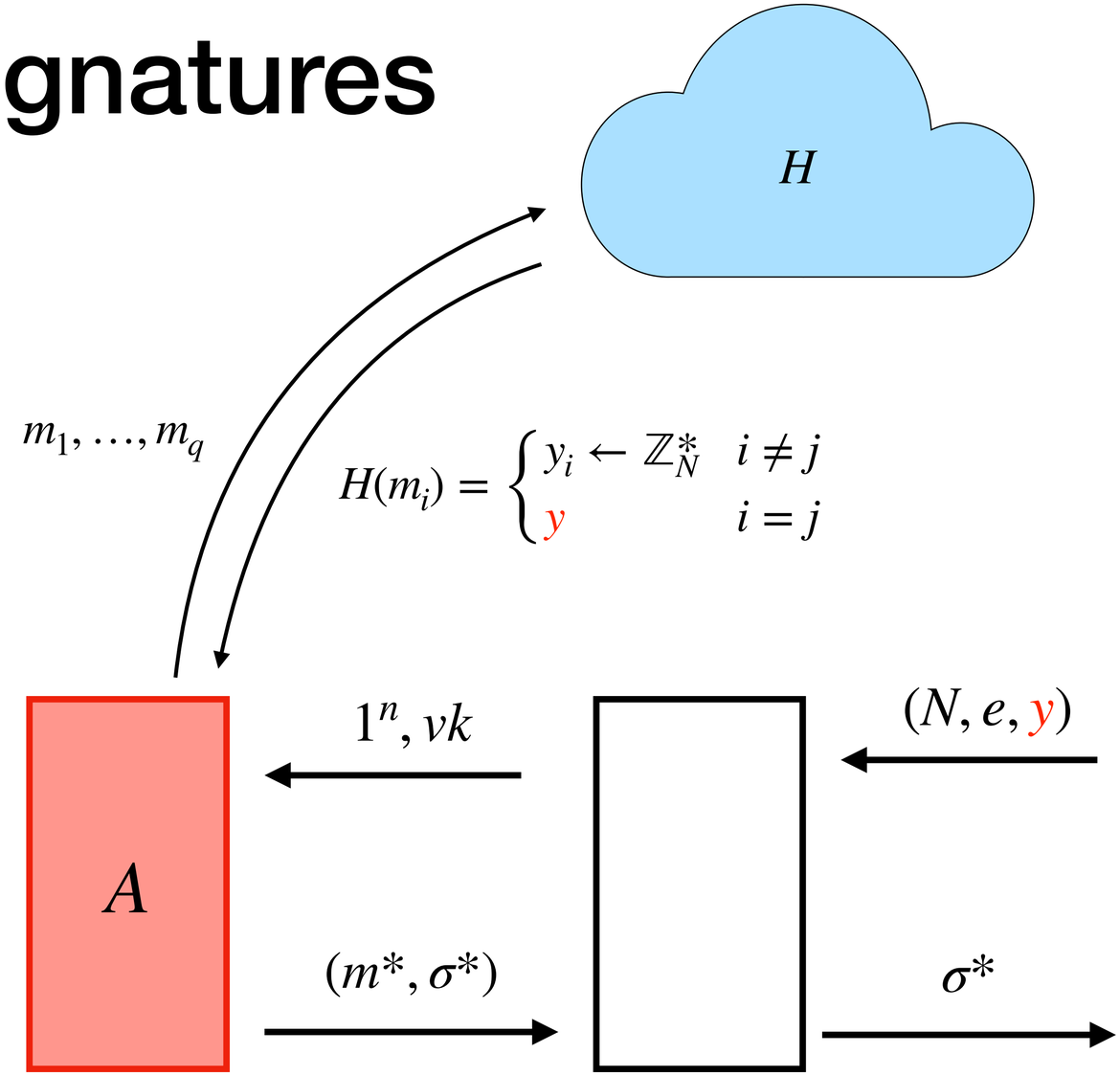
- Reduction gets instance (N, e, y) and gives $vk = (e, N)$ to A
- Reduction sets $H(m_j) = y$
- A valid forgery on m^* is $\sigma^* = H(m^*)^d$
- If $m^* = m_j$ then $\sigma^* = H(m^*)^d = y^d = (x^e)^d = x$



RSA-FDH Signatures

Proof strategy: Our reduction will guess which message m_j will be m^* and use m_j to break the RSA assumption

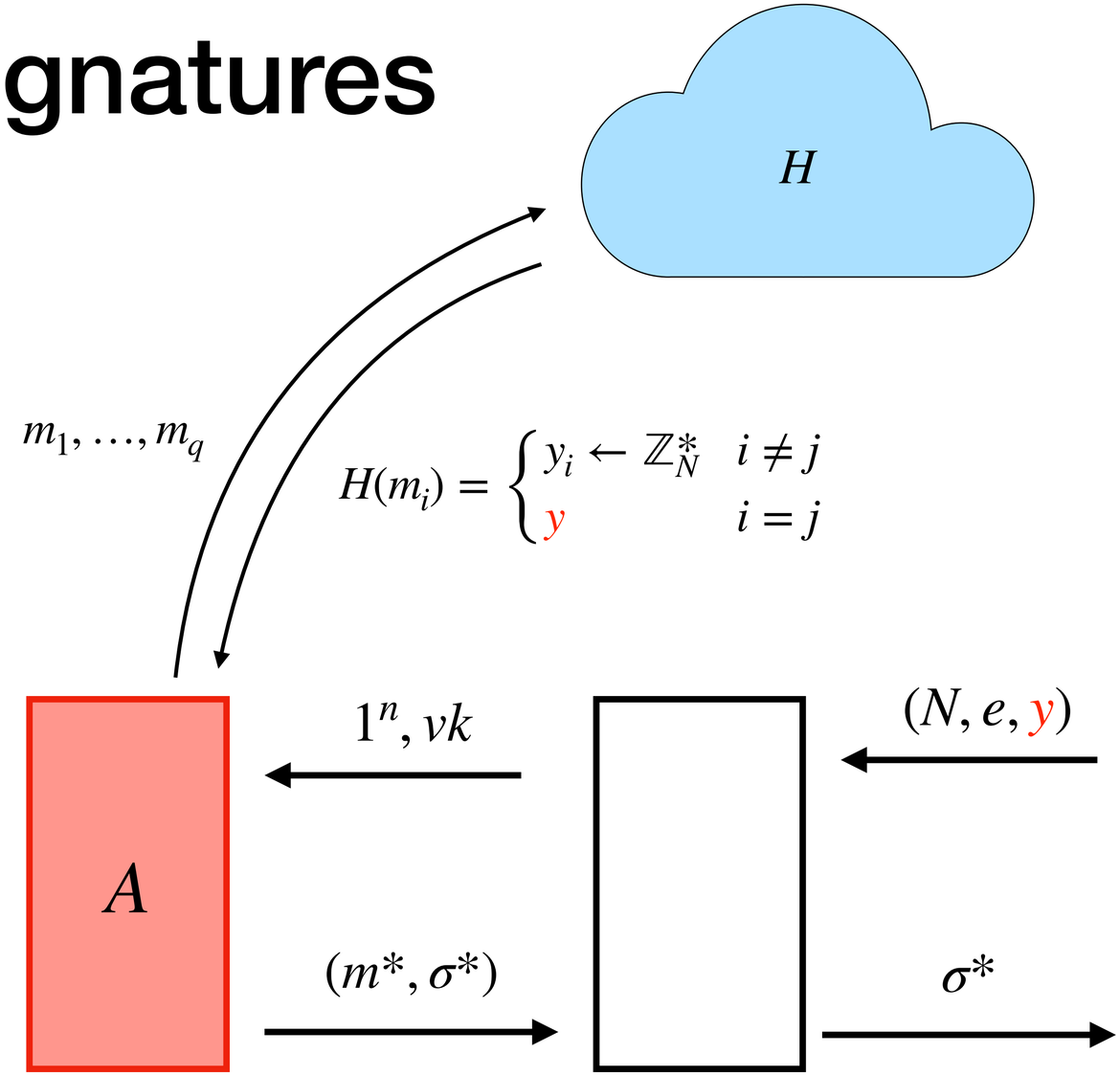
- Reduction gets instance (N, e, y) and gives $vk = (e, N)$ to A
- Reduction sets $H(m_j) = y$
- A valid forgery on m^* is $\sigma^* = H(m^*)^d$
- If $m^* = m_j$ then $\sigma^* = H(m^*)^d = y^d = (x^e)^d = x$



RSA-FDH Signatures

Proof strategy: Our reduction will guess which message m_j will be m^* and use m_j to break the RSA assumption

- Reduction gets instance (N, e, y) and gives $vk = (e, N)$ to A
- Reduction sets $H(m_j) = y$
- A valid forgery on m^* is $\sigma^* = H(m^*)^d$
- If $m^* = m_j$ then $\sigma^* = H(m^*)^d = y^d = (x^e)^d = x$



How often does our reduction win?

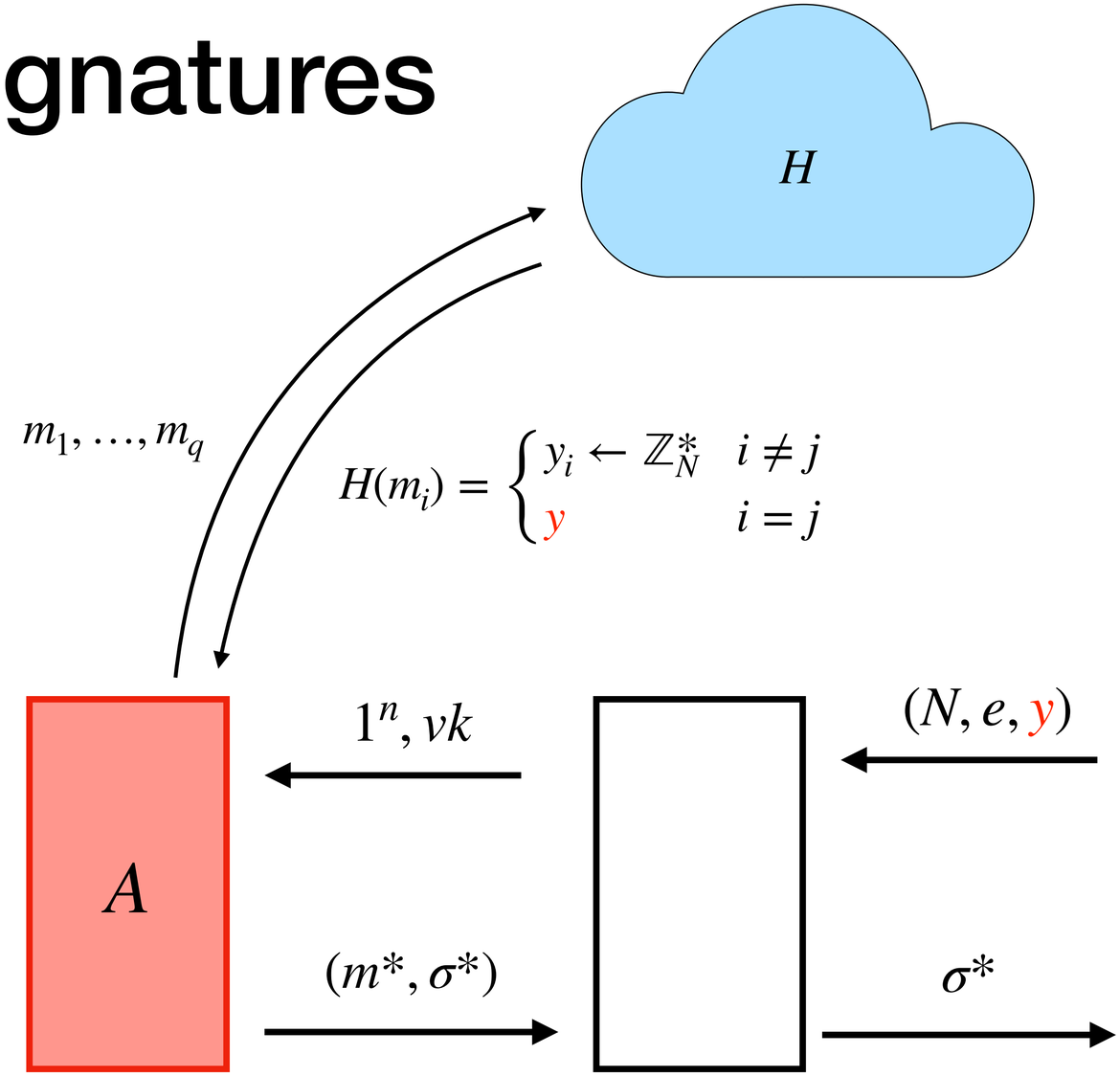
RSA-FDH Signatures

Back to our main proof:

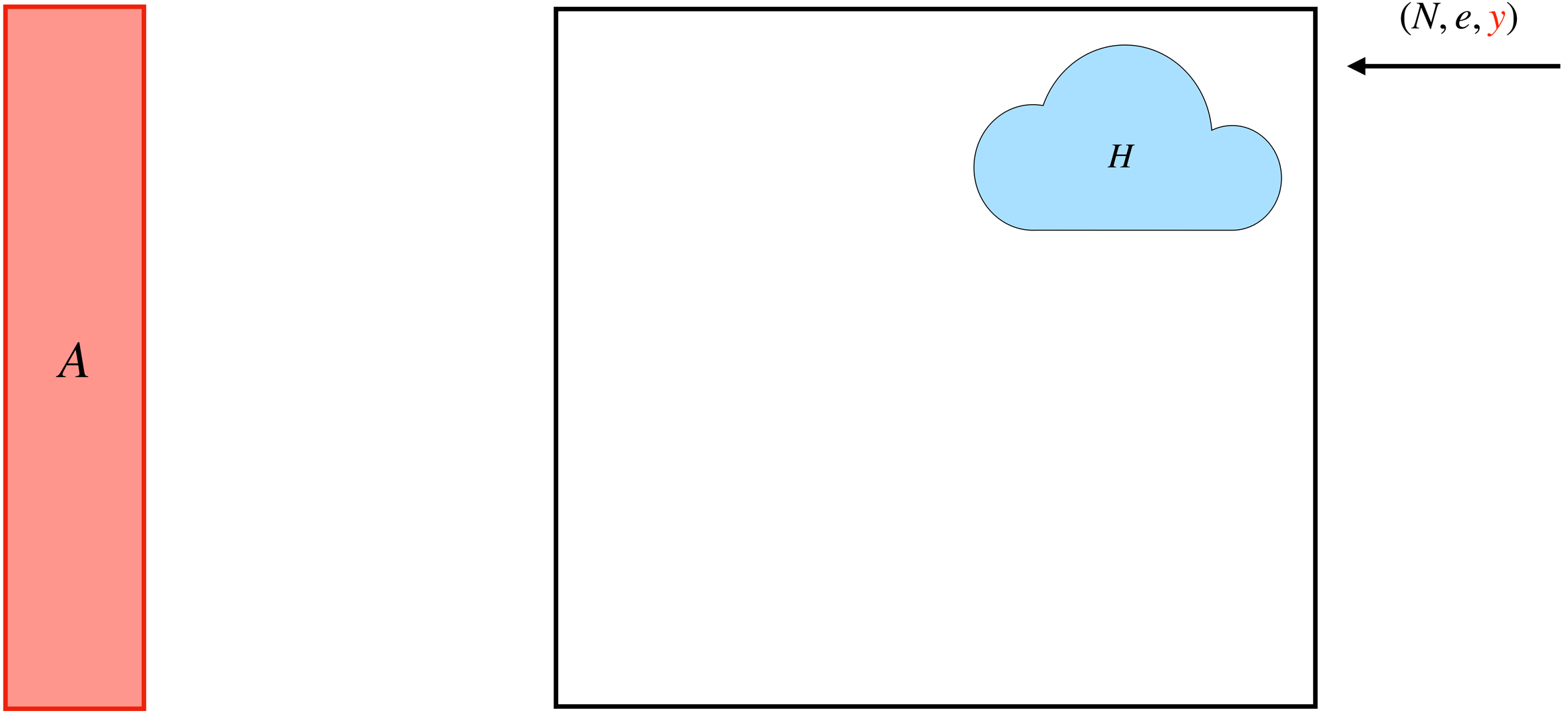
Need to handle $\mathsf{Sign}_{sk}(\cdot)$ queries

w.l.o.g. can make assumptions on A:

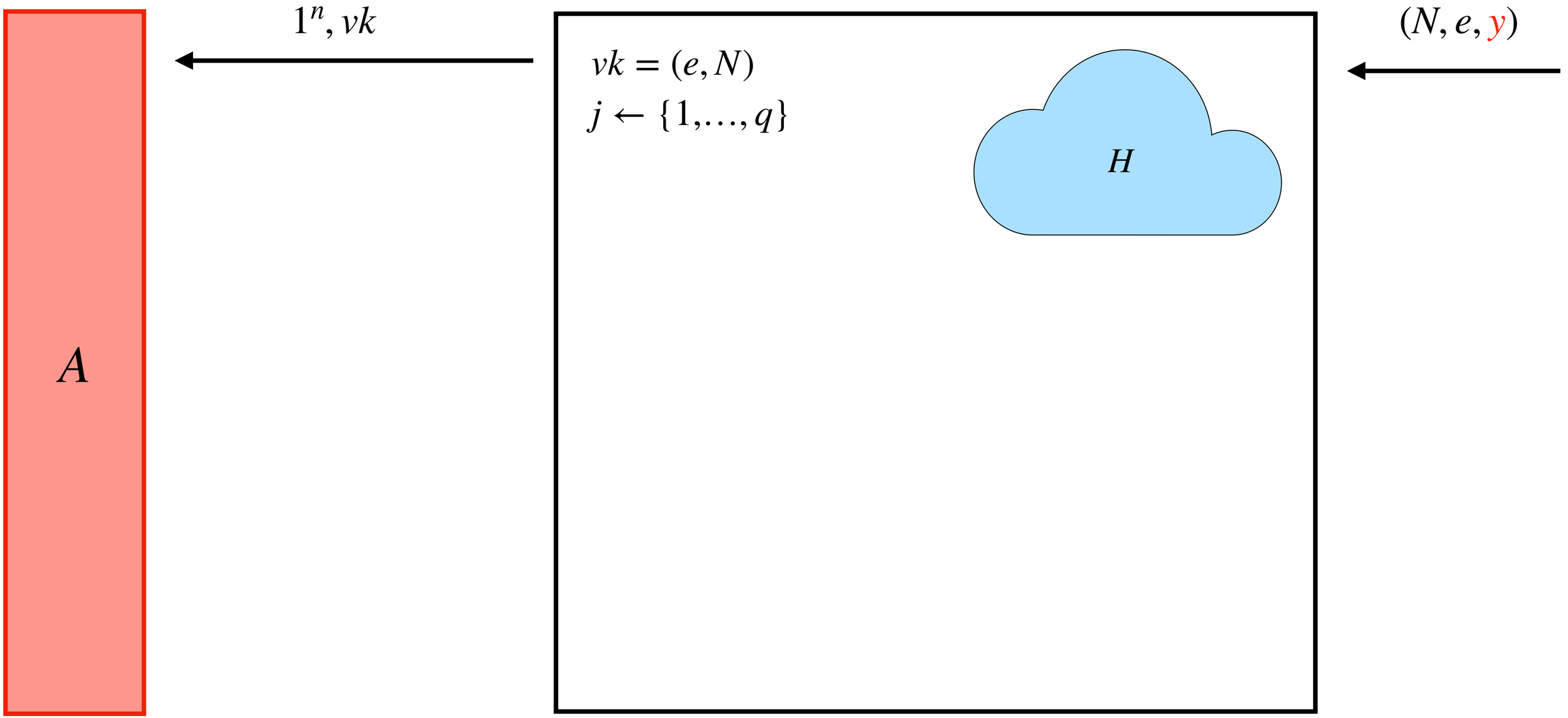
- Any queries to Sign_{sk} with m also have a corresponding query to H with m
- A queries m^* to H
- A never repeats queries to H



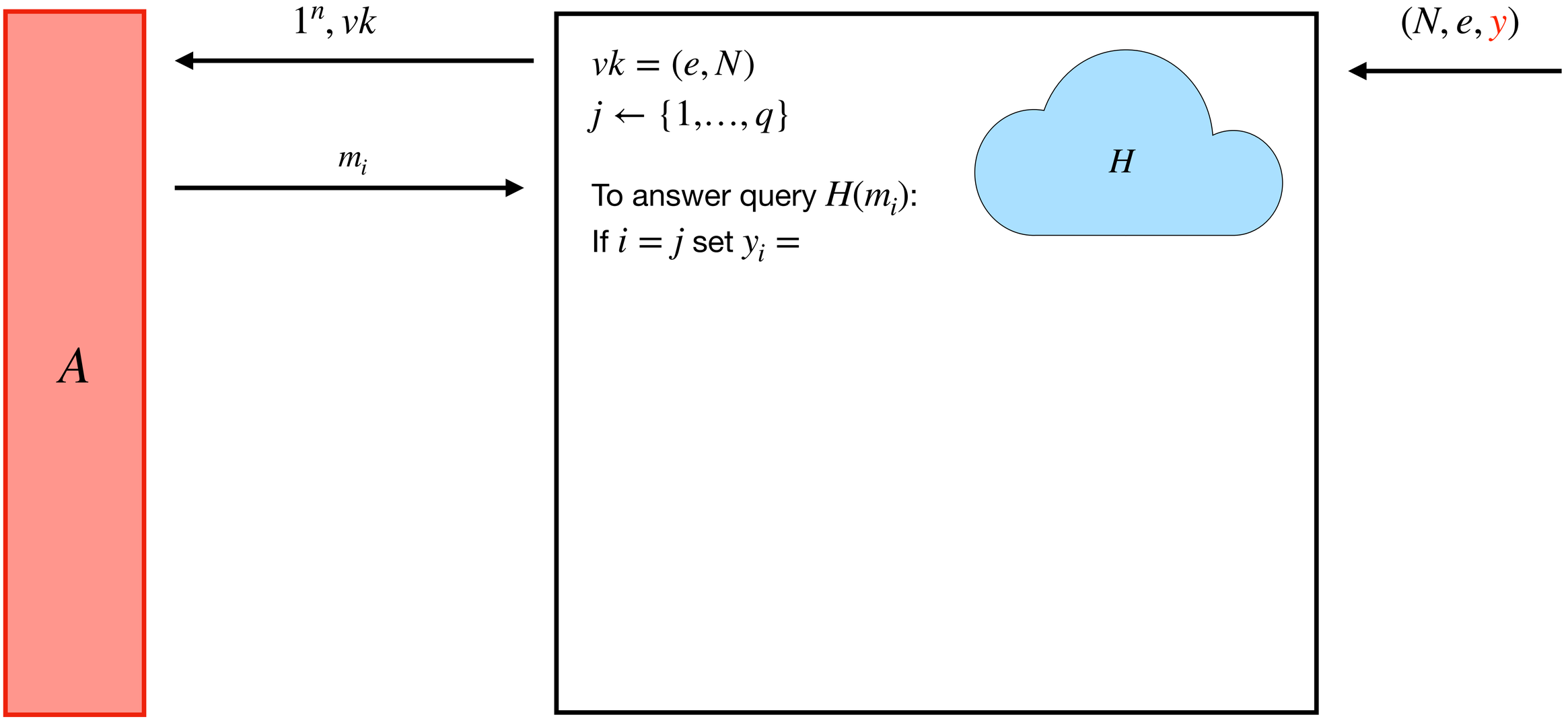
RSA-FDH Signatures



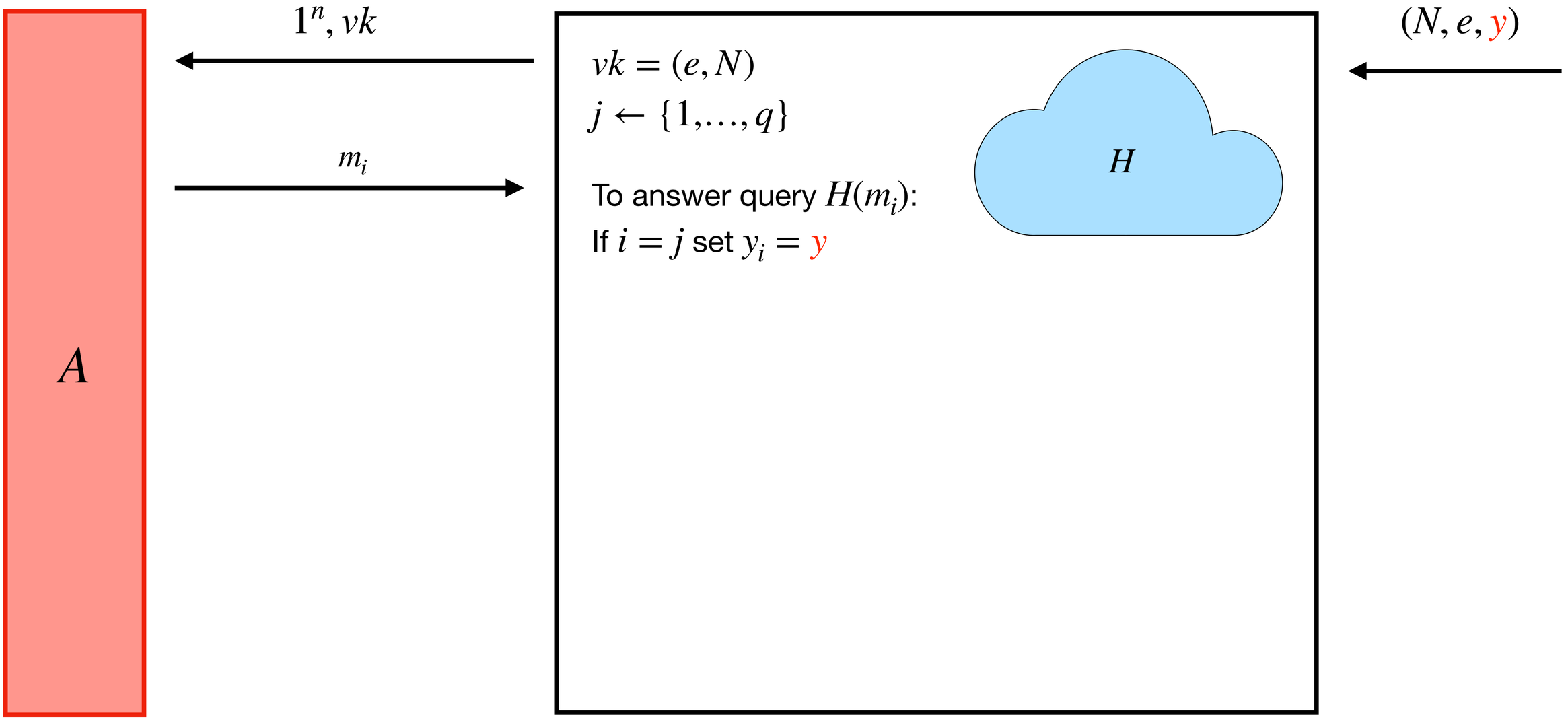
RSA-FDH Signatures



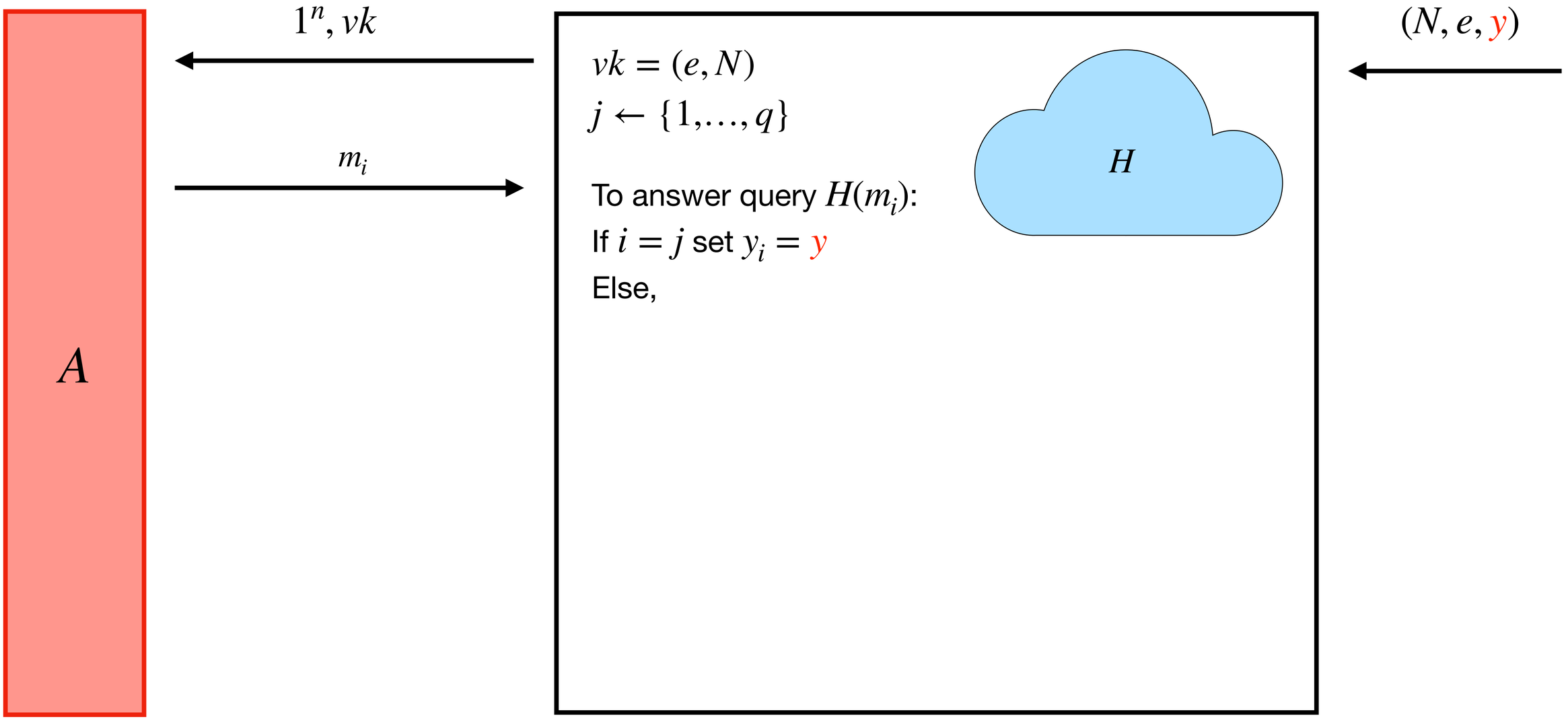
RSA-FDH Signatures



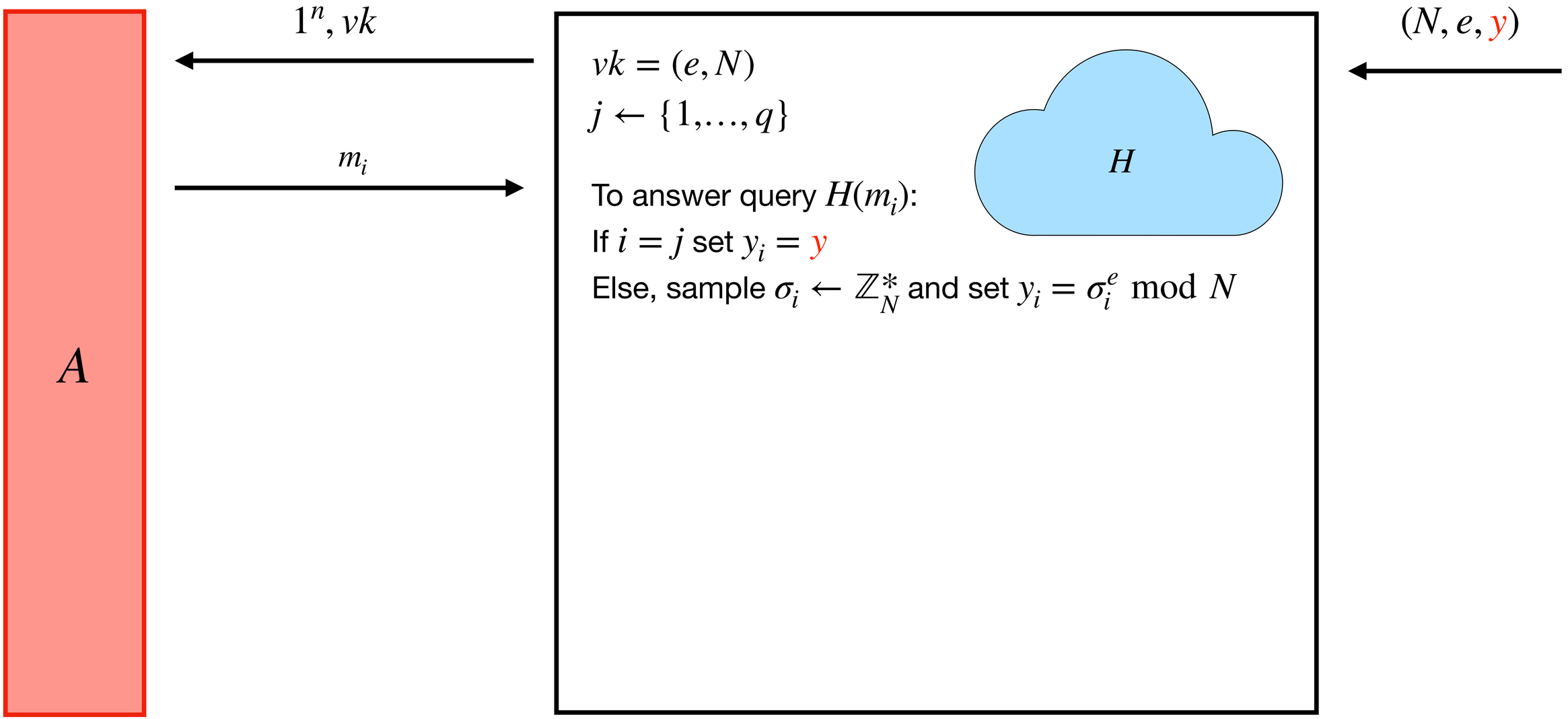
RSA-FDH Signatures



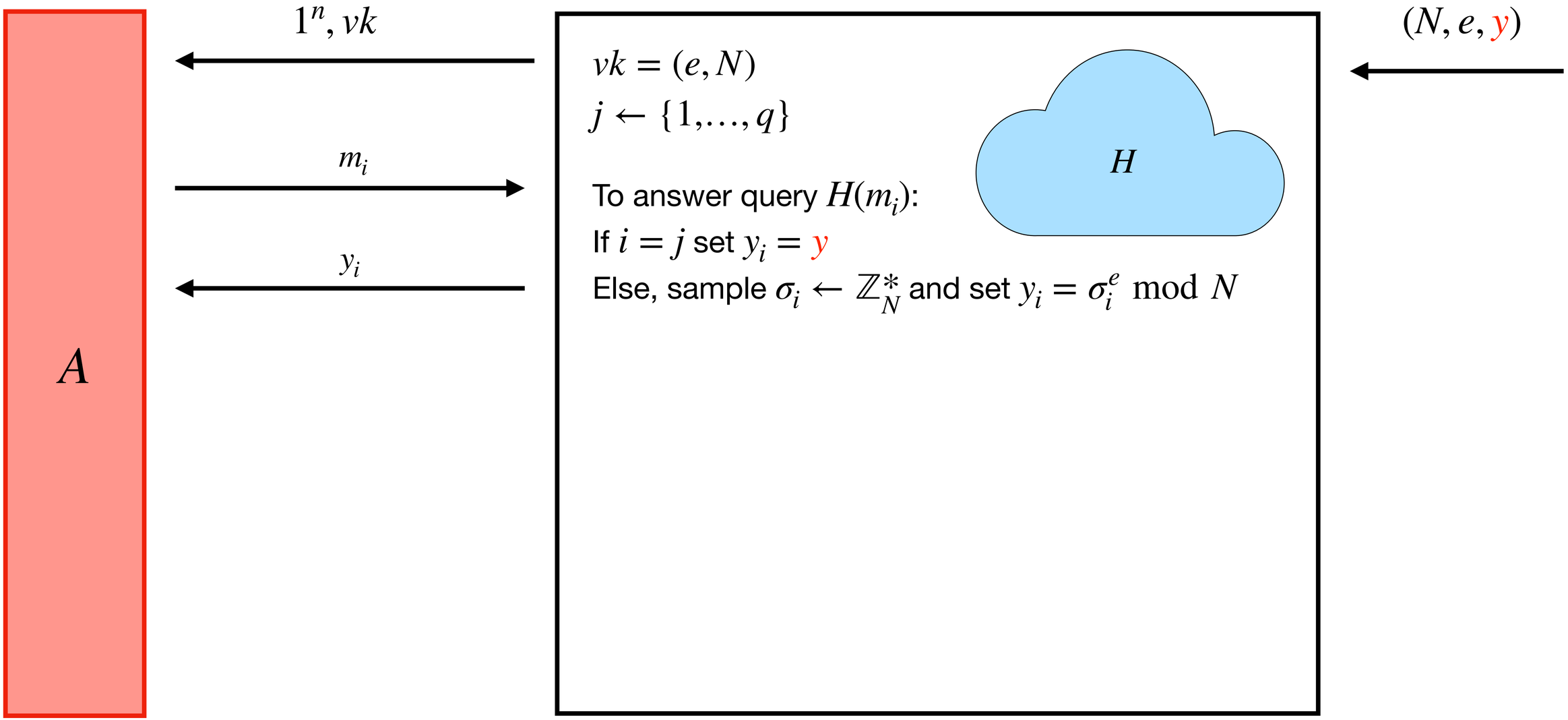
RSA-FDH Signatures



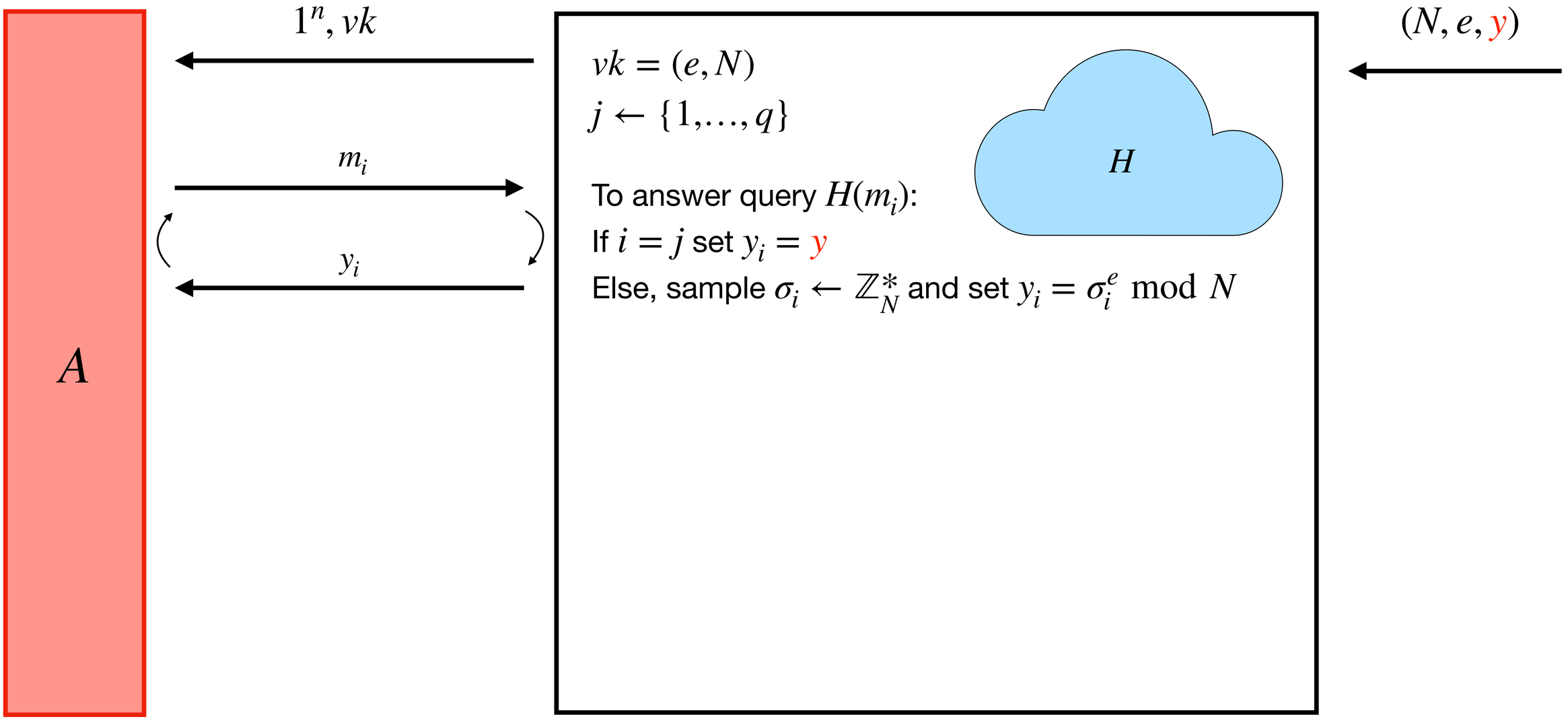
RSA-FDH Signatures



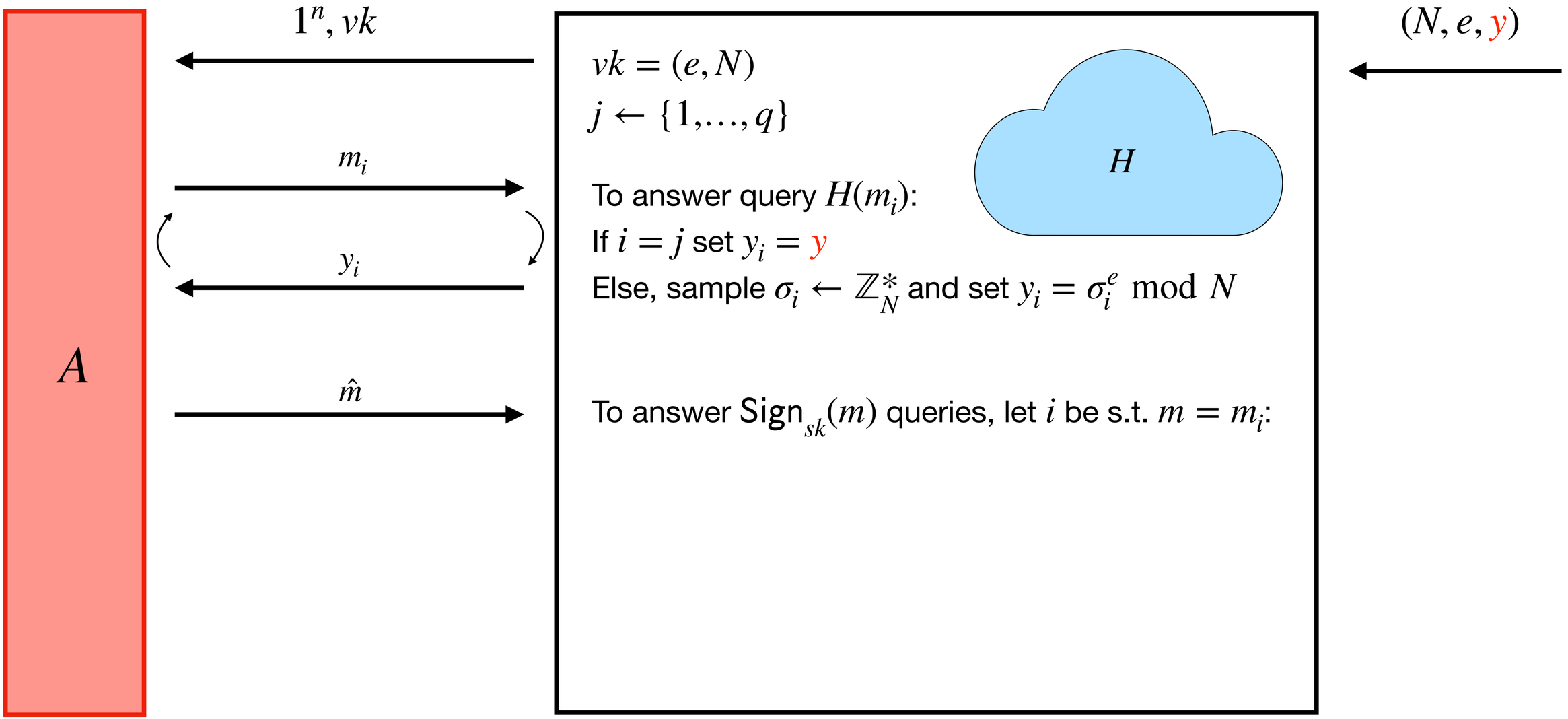
RSA-FDH Signatures



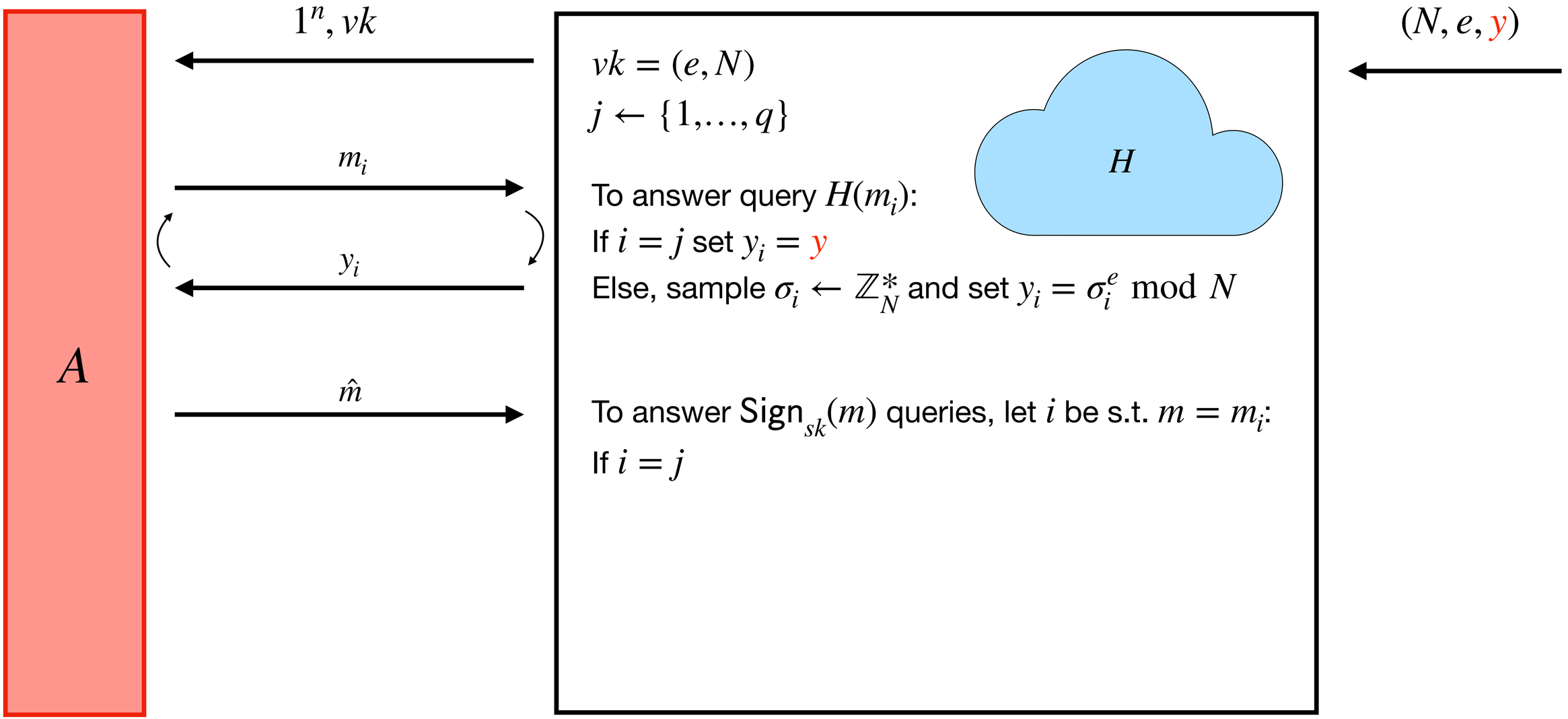
RSA-FDH Signatures



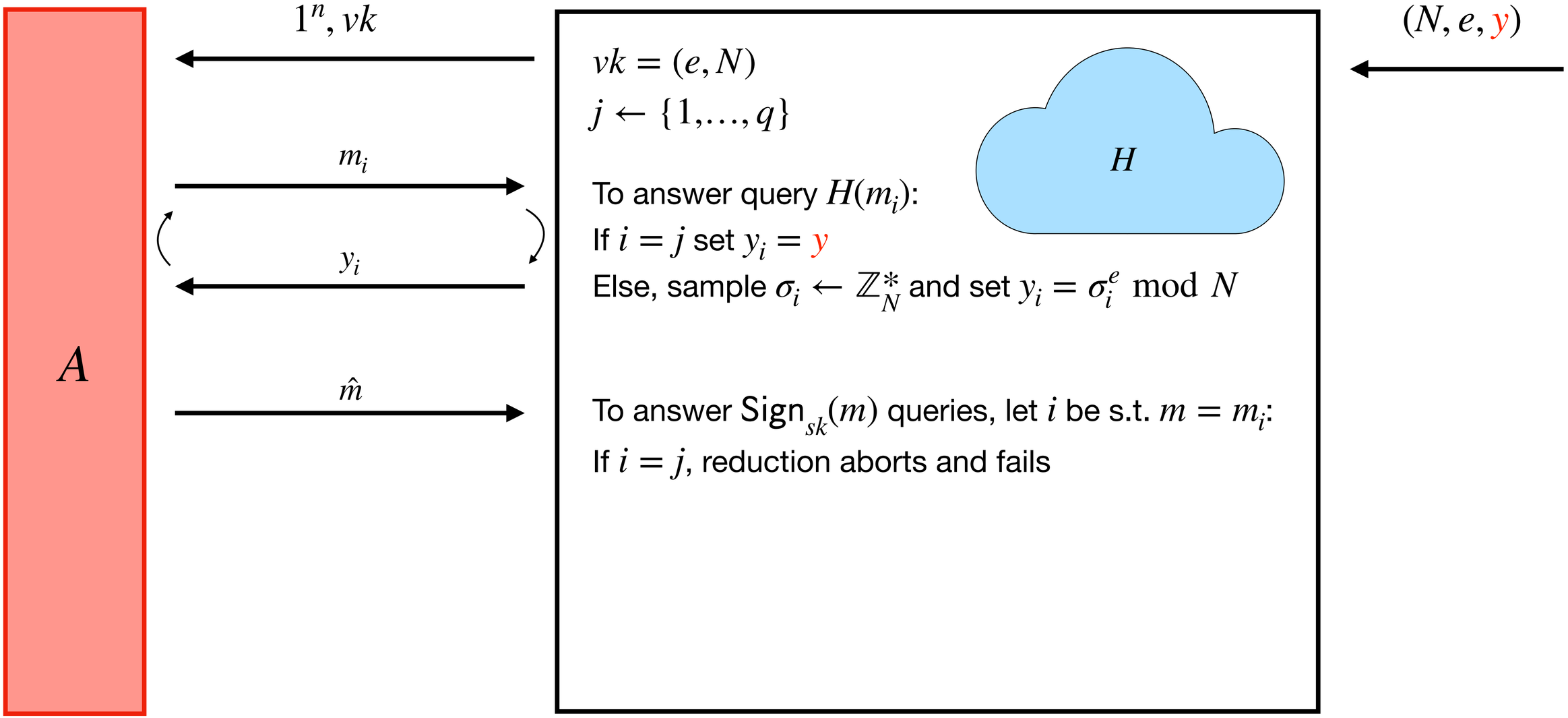
RSA-FDH Signatures



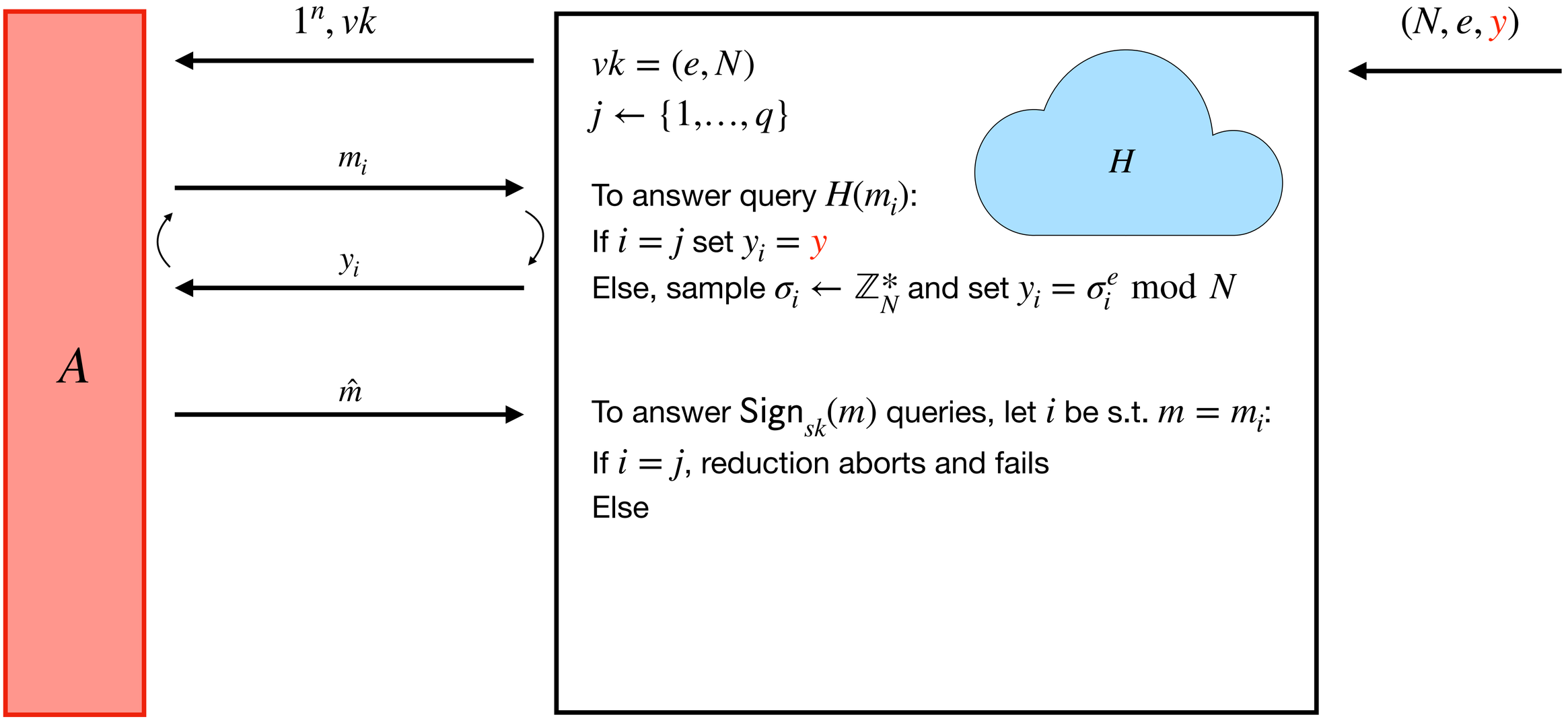
RSA-FDH Signatures



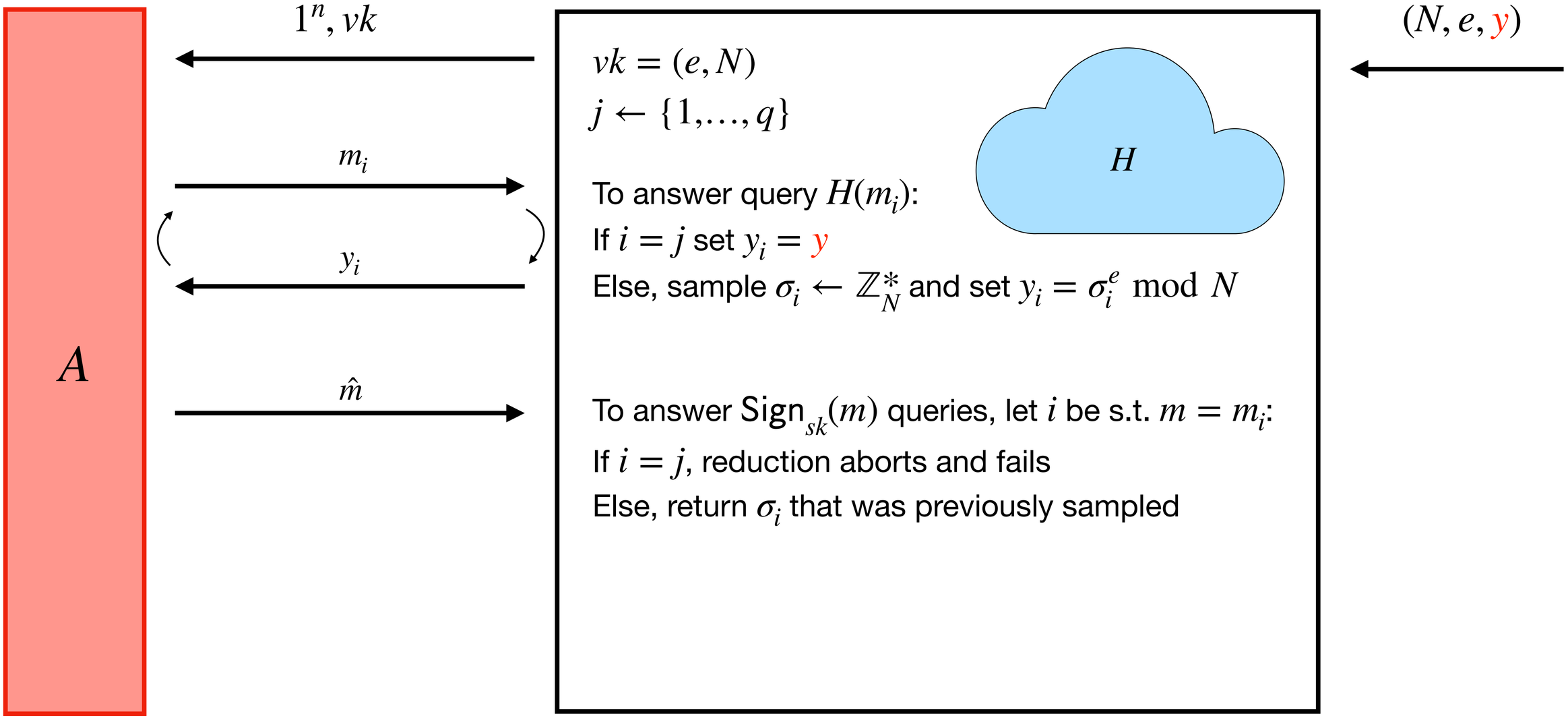
RSA-FDH Signatures



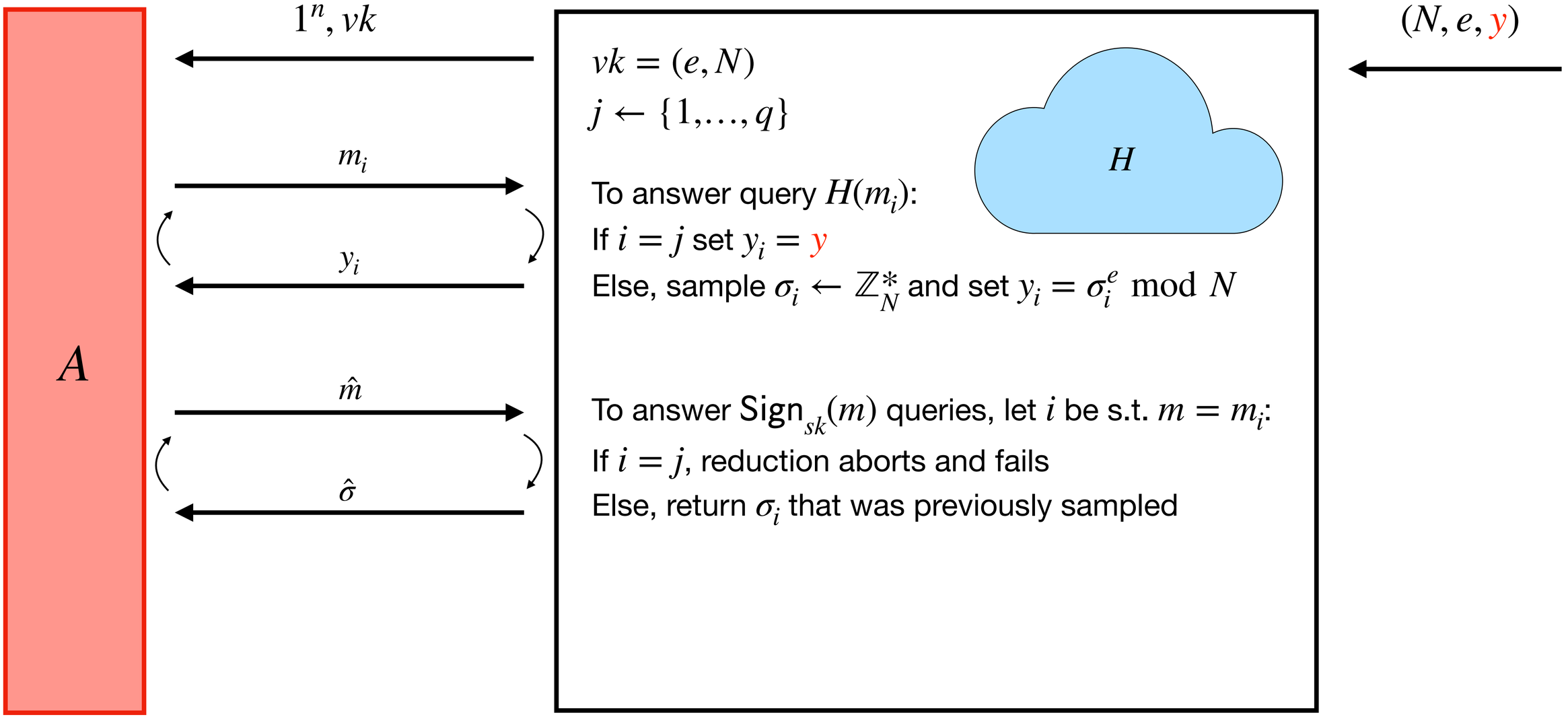
RSA-FDH Signatures



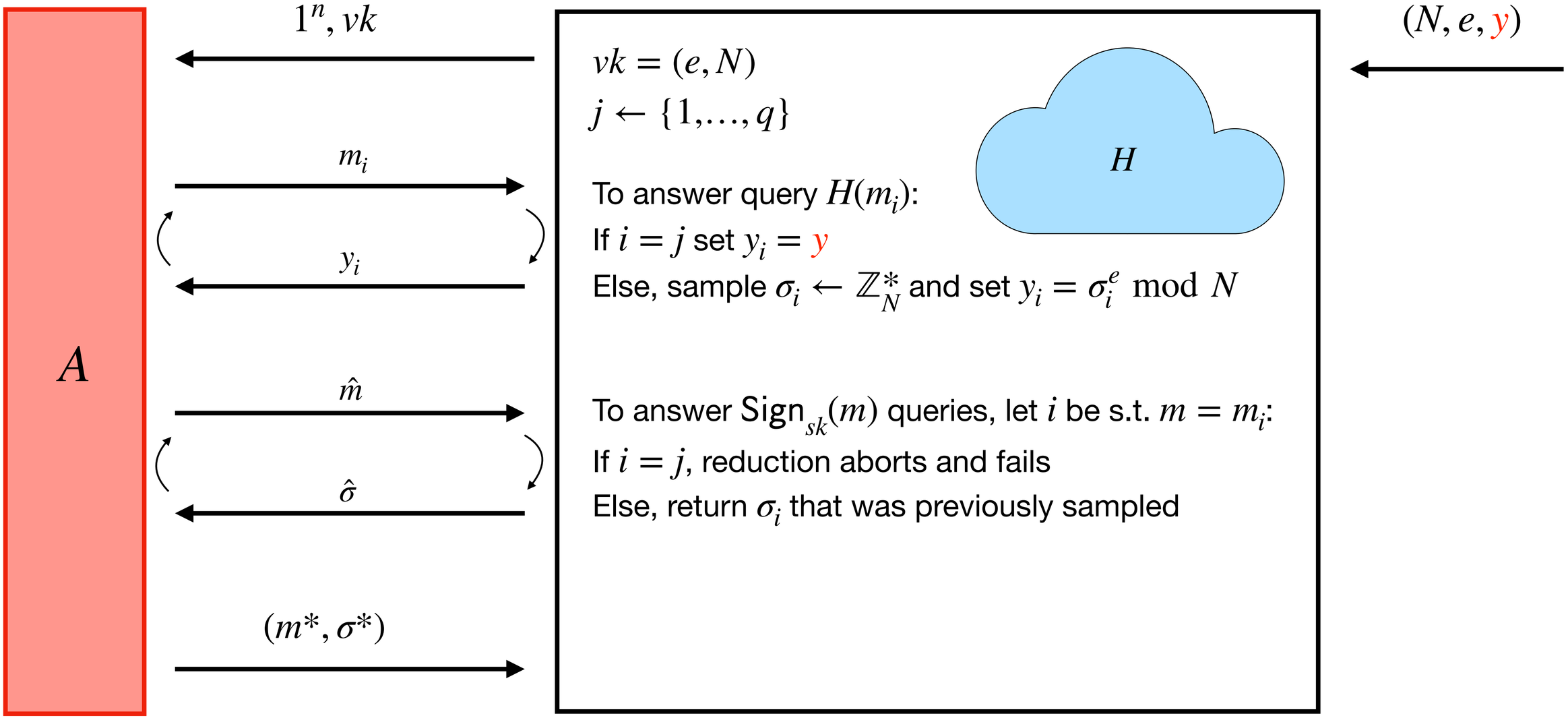
RSA-FDH Signatures



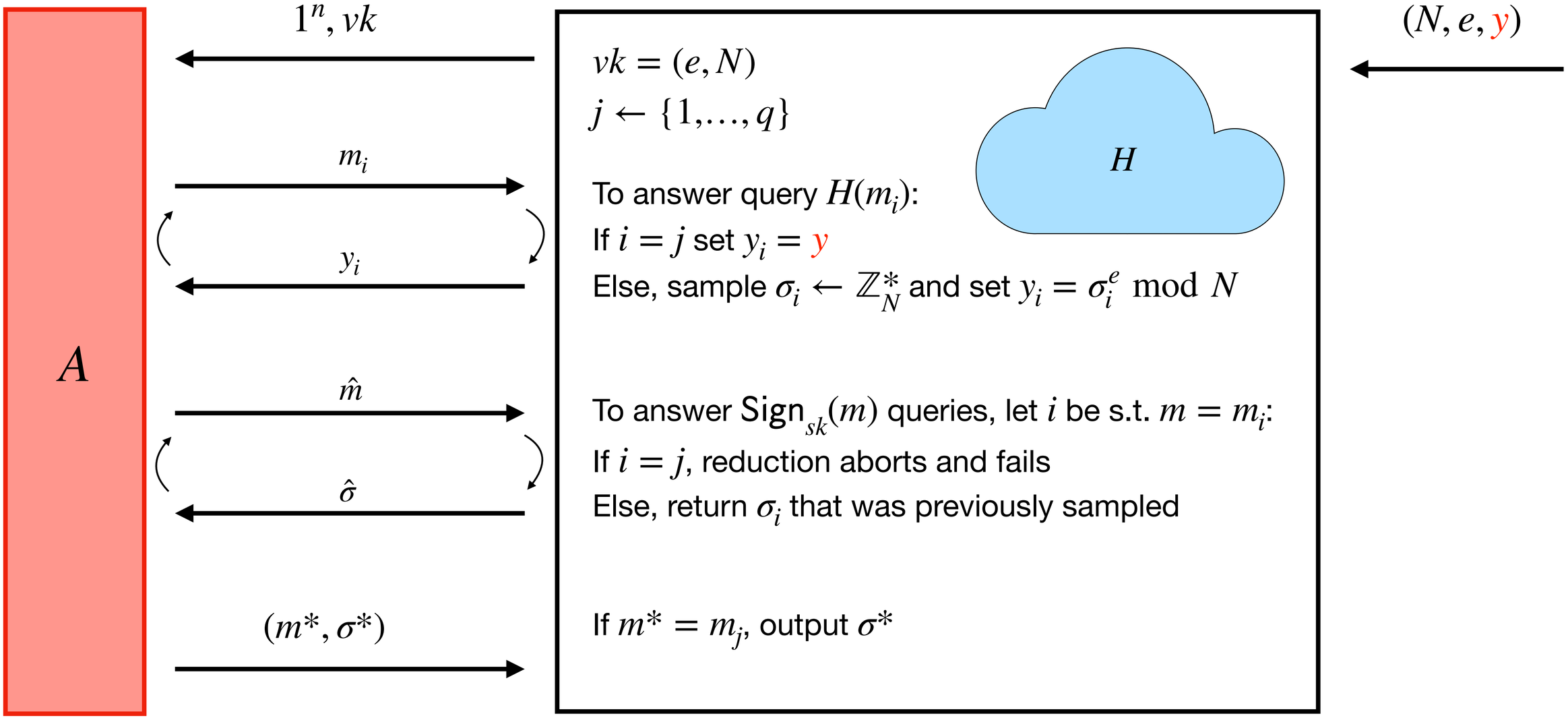
RSA-FDH Signatures



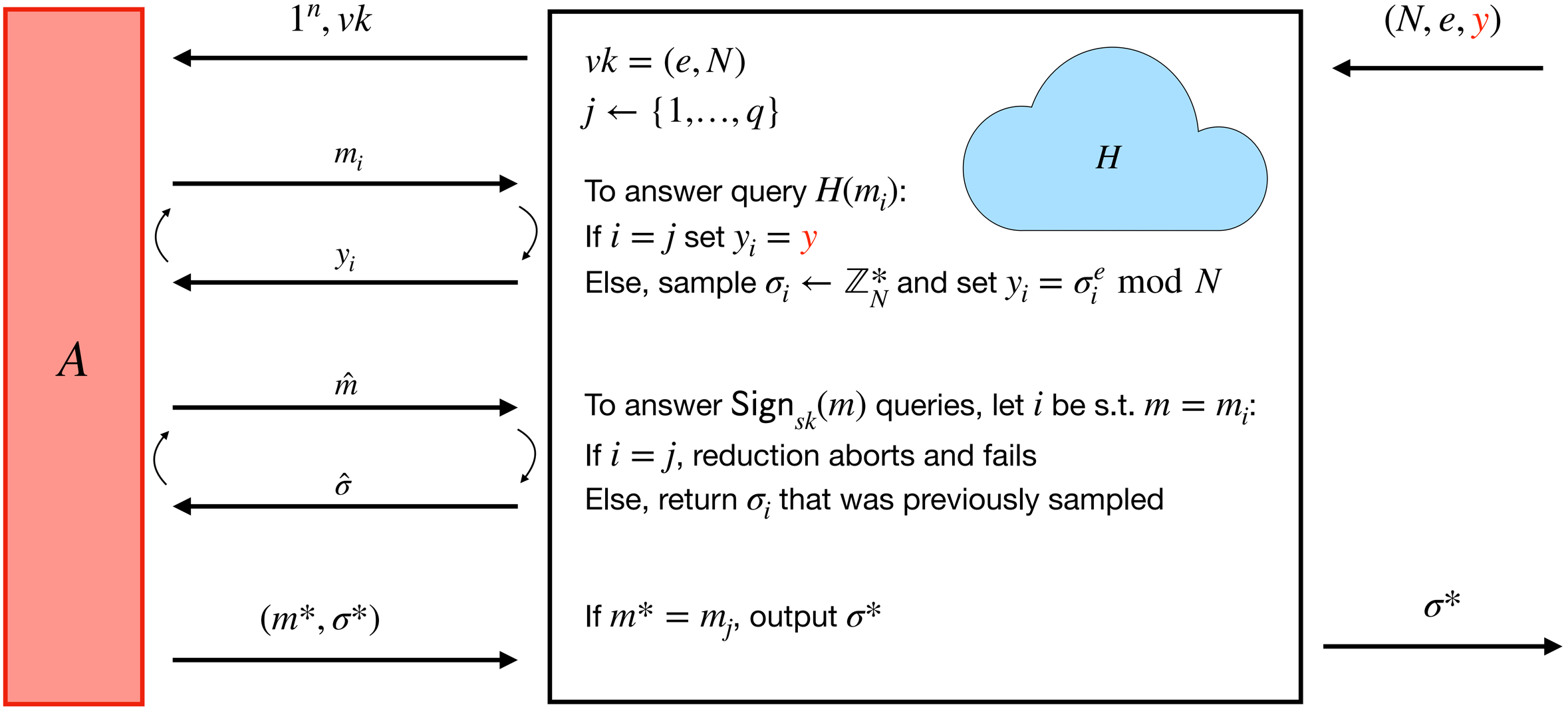
RSA-FDH Signatures



RSA-FDH Signatures



RSA-FDH Signatures



Next Time

- Identification Schemes
- Schnorr Signatures