Announcements

Test 2 - Nov 13

HW4 Deadline

Crypto Reading Group (tomorrow 3pm)

Message Authentication Code

Private key Setting

Property: Unforgeability

Constructions:

□ MAC from PRF

□ CBC-MAC

Hash Functions

NO secret key!!

Property: Collision-Resistance

□ Merkle-Damgård Transform

□ Hash-function Block-ciphers (Davies-Mayer)

□ Hash-function from Discrete Log Assumption.

Message Authentication Code

Private key Setting

Property: Unforgeability

Constructions:

□ MAC from PRF

□ CBC-MAC

Digital Signature

Public key Setting

Property: Unforgeability

Constructions:

□ RSA-based

□ (General) One-way Function

Hash Functions

NO secret key!!

Property: Collision-Resistance

□ Merkle-Damgård Transform

□ Hash-function Block-ciphers (Davies-Mayer)

□ Hash-function from Discrete Log Assumption.

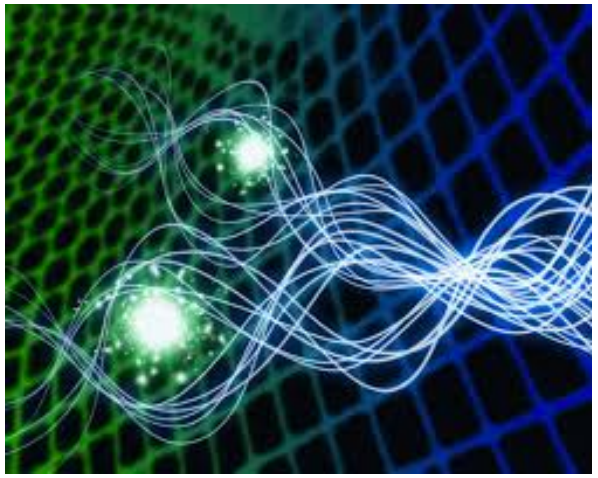
Digital Signature

- Definition Unforgeability

- Constructions

  RSA -based Signatures

  One-time Signatures from OWF



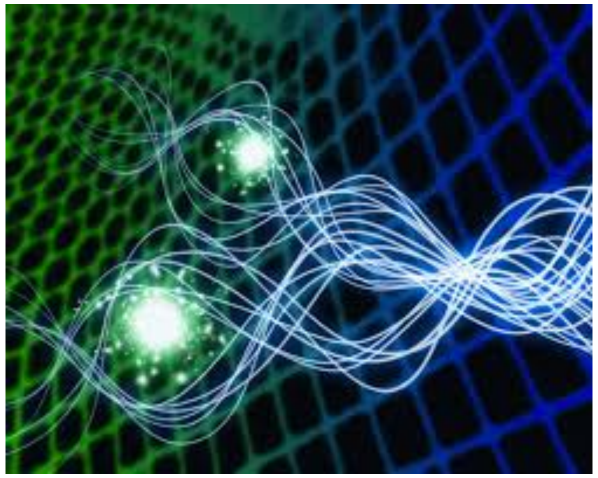
Digital Signature

- Definition   Unforgeability

- Constructions

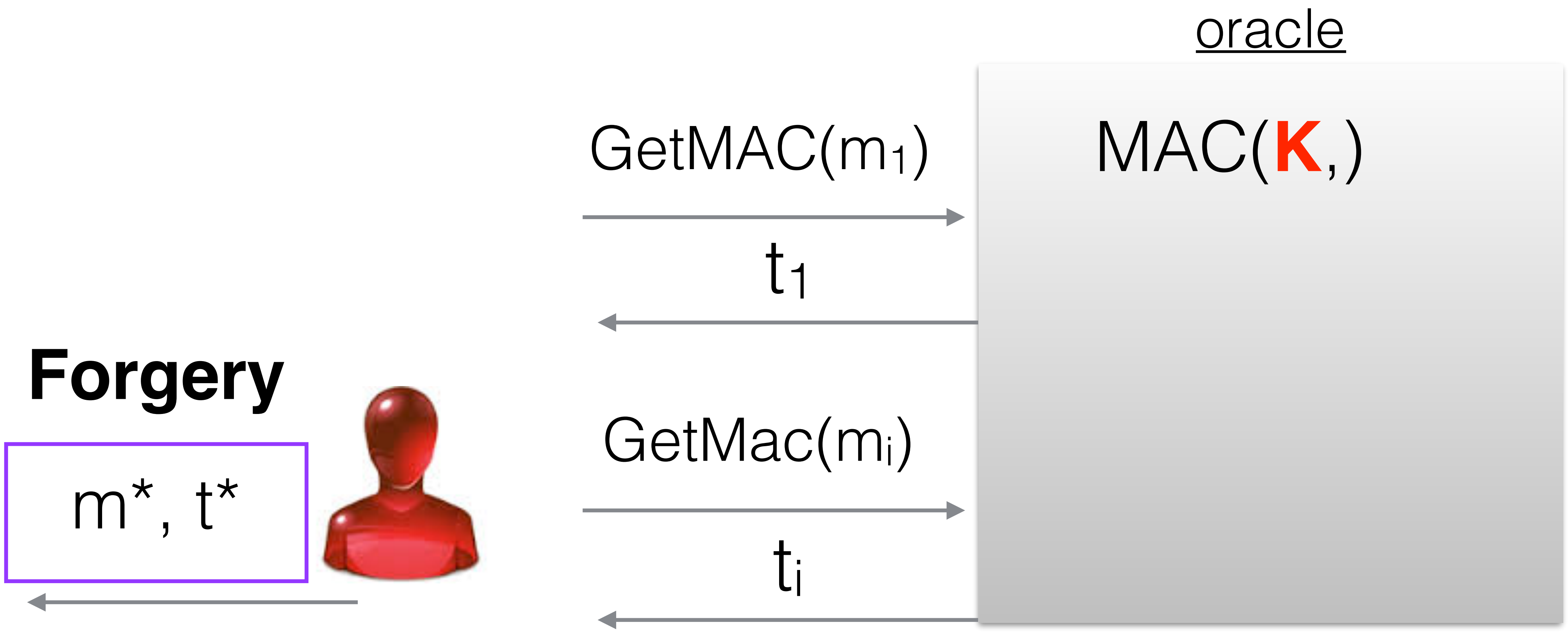
  RSA -based Signatures

  One-time Signatures from OWF



MAC security definition

Unforgeability



oracle

MAC(**K**,)

GetMAC(m_1)

t_1

GetMac(m_i)

t_i

Forgery

m^*, t^*

Digital Signature Public key Setting

Digital Signature Public key Setting

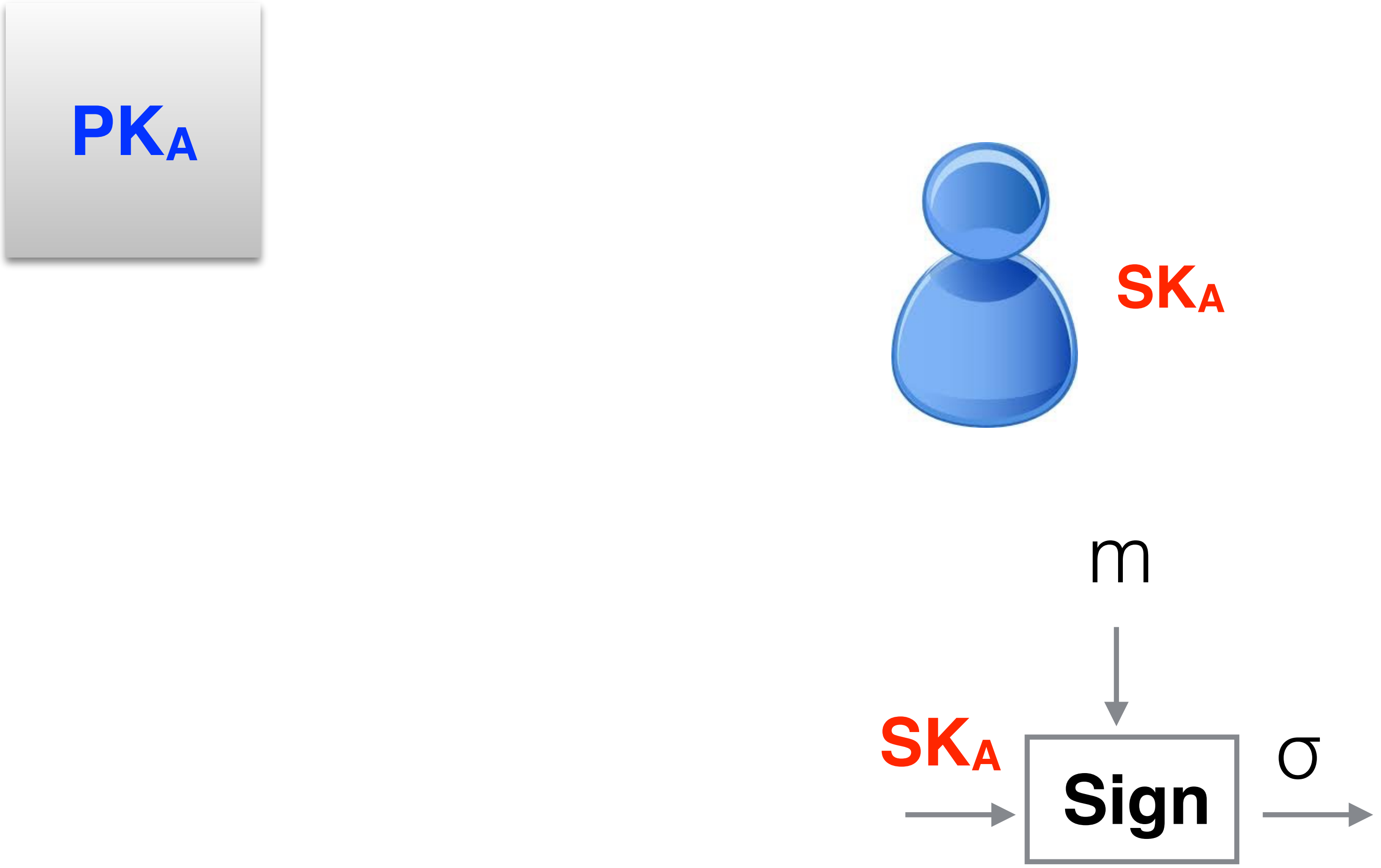
PK_A

Digital Signature Public key Setting

PK_A

SK_A

Digital Signature Public key Setting



PK_A

SK_A

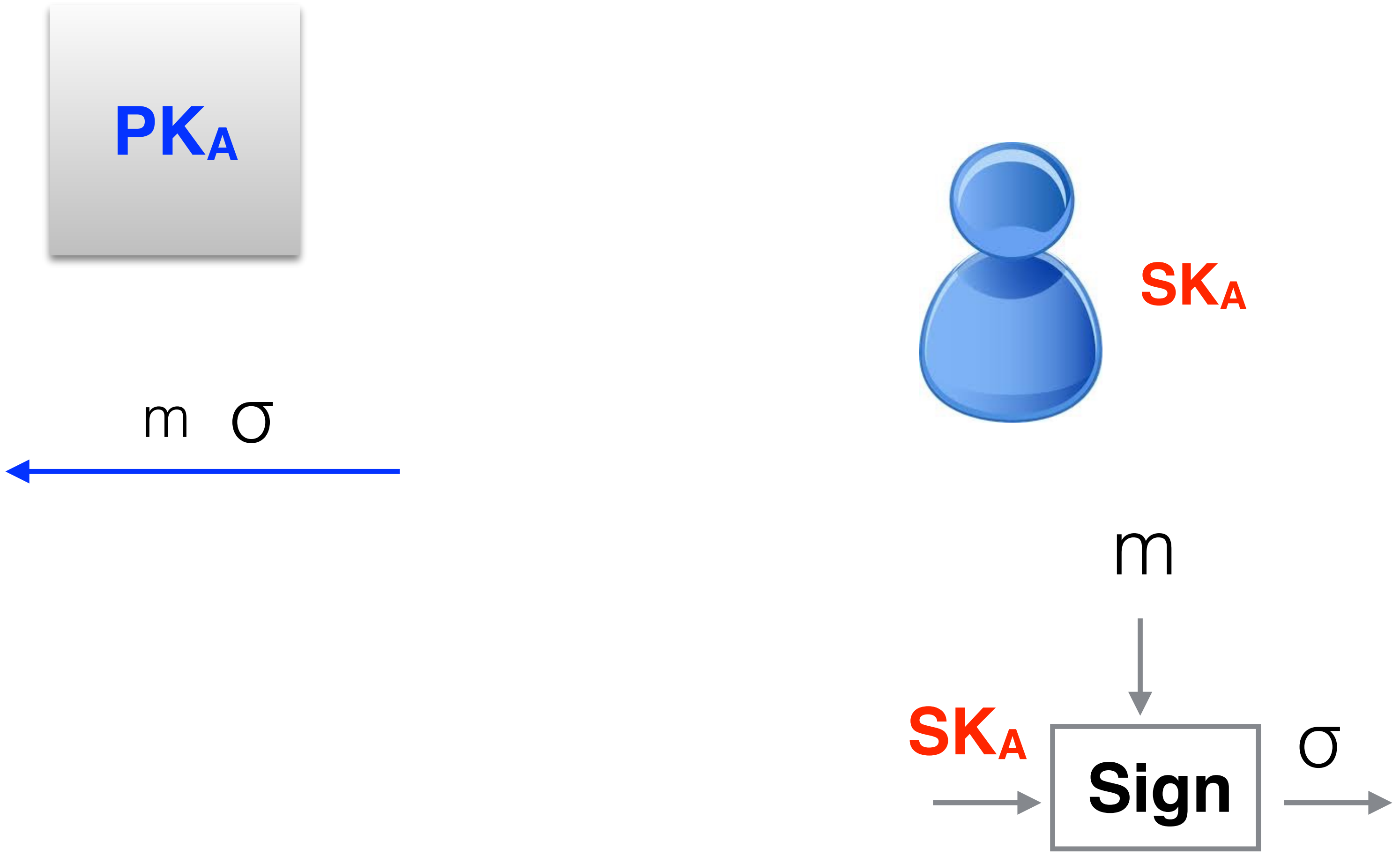
m

SK_A

Sign

σ

Digital Signature Public key Setting



PK_A

SK_A

m σ

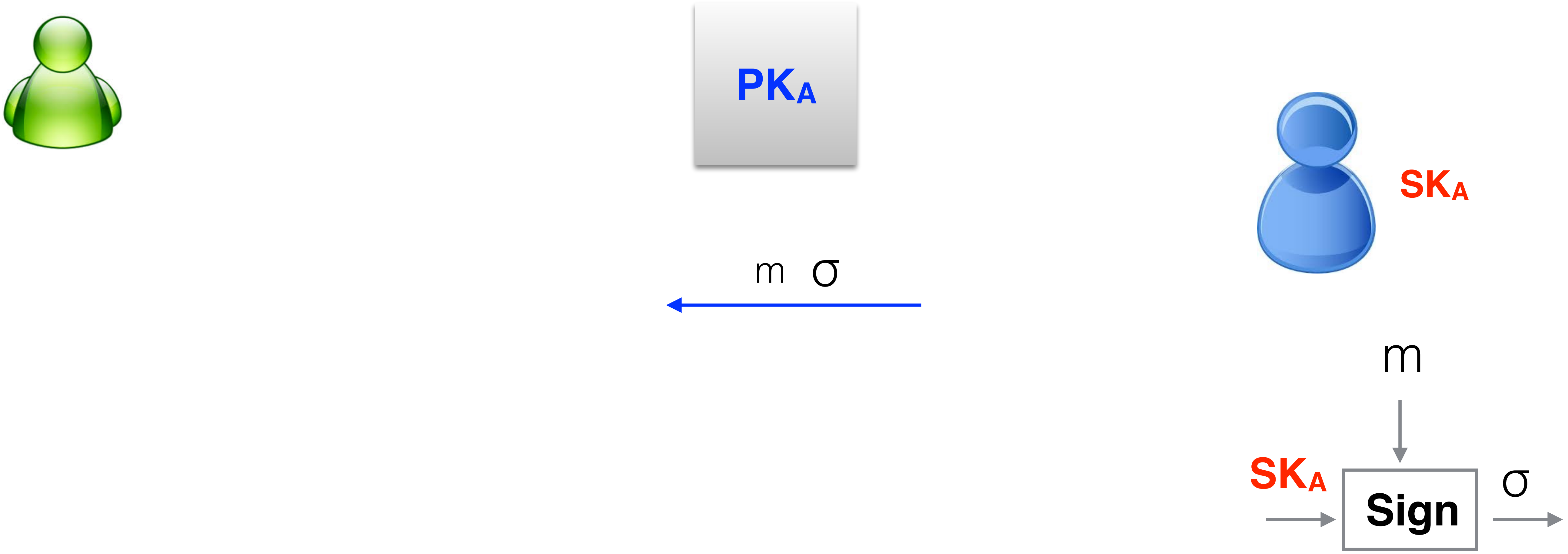
m

SK_A

Sign

σ

Digital Signature Public key Setting



PK_A

m σ

SK_A

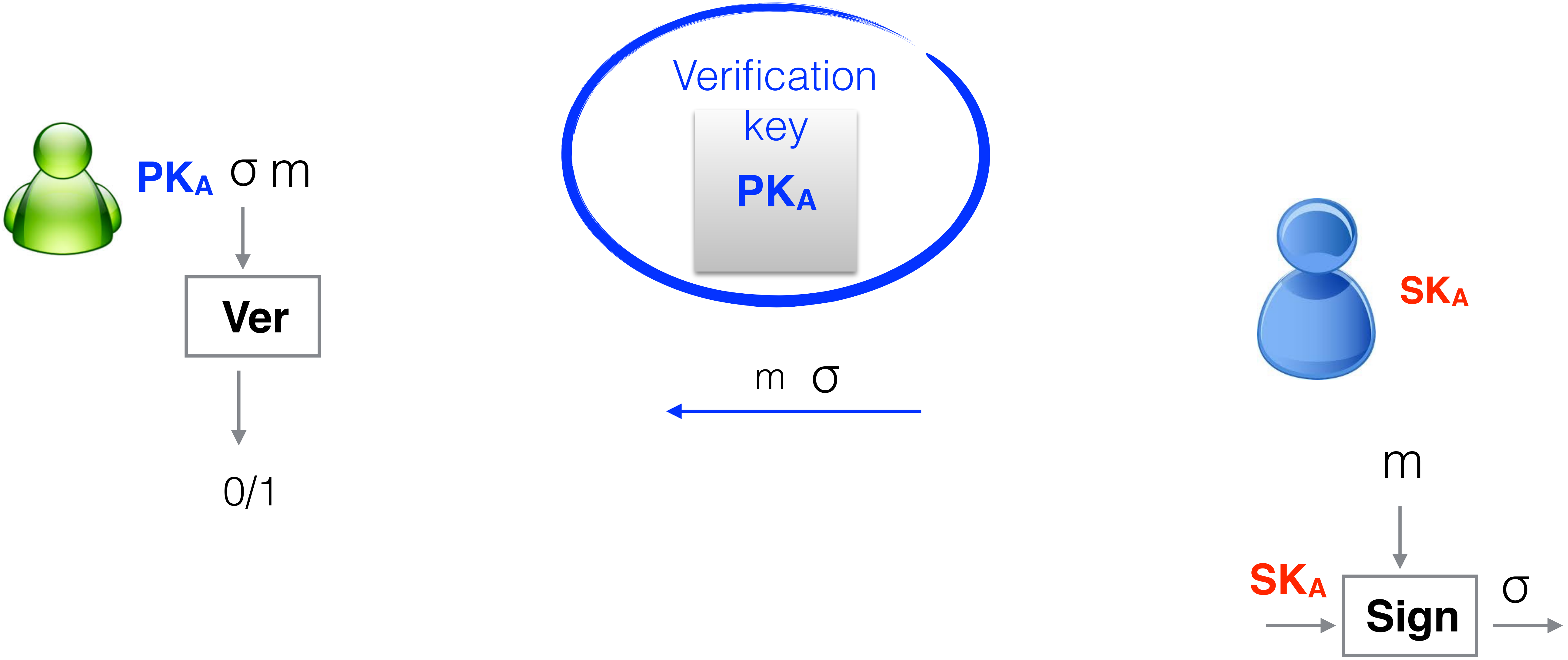
m

SK_A

Sign

σ

Digital Signature Public key Setting



PK_A σ m

Ver

0/1

Verification key

PK_A

m σ

SK_A

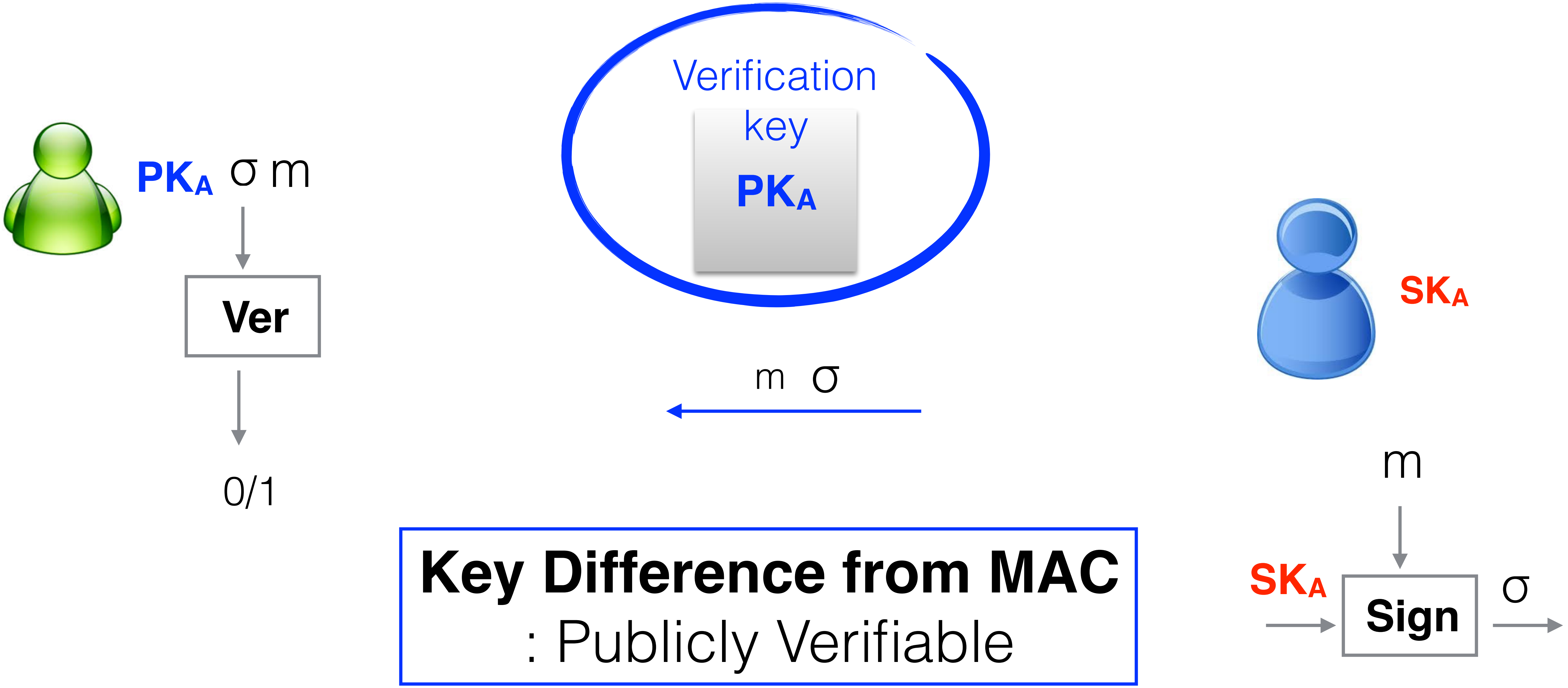
m

SK_A

Sign

σ

Digital Signature Public key Setting



Key Difference from MAC

: Publicly Verifiable

Digital Signature public key setting

bitcoin

Transaction PK_A, 10\$ > PK_B σ	Transaction PK_A, 10\$ > PK_B σ	Transaction PK_A, 10\$ > PK_B σ	Transaction PK_A, 10\$ > PK_B σ	Transaction PK_A, 10\$ > PK_B σ

Publicly Verifiability: ***everybody should*** verify transactions using the PK

Unforgeability: nobody should sign transactions on a user's behalf.

Use of Digital Signatures

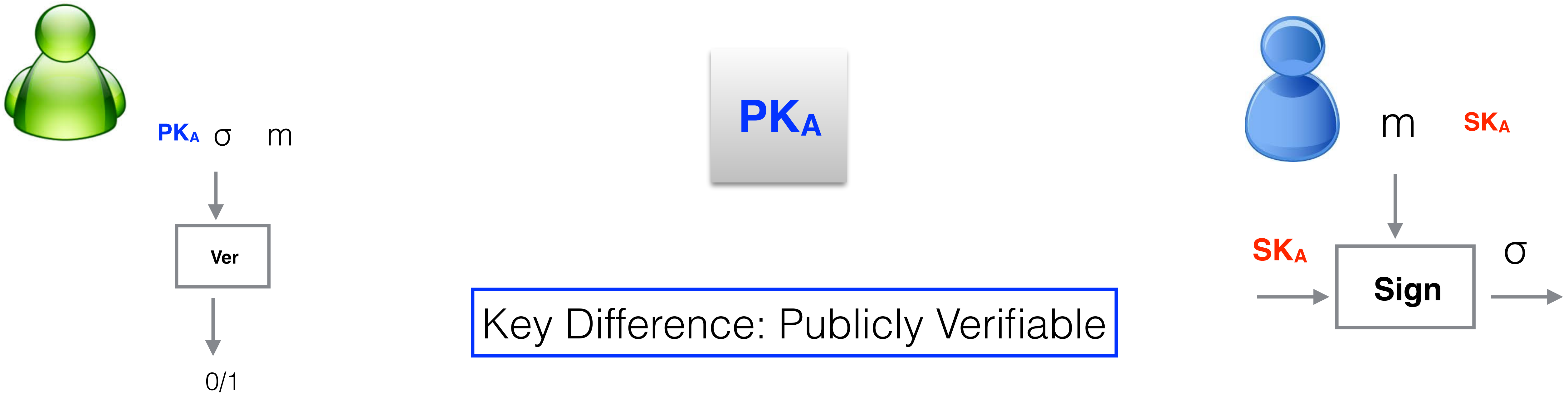
(1) Releasing Software Patches

(2) Signing Transactions

....

(3) In general certifying documents that have to be publicly verifiable

Digital Signature Public key Setting



Key Difference: Publicly Verifiable

Syntax: Signature Scheme

Key Generation: GenKey(n) ->(PK_A,SK_A)

Signing Algorithm: Sign(SK_A, m) -> σ

Verification Algorithm: Verify(PK_A , m, σ) -> 0/1

Digital Signature: Unforgeability

We want:

No-one should be able to compute signature on behalf of a certain PK

Digital Signature: Unforgeability

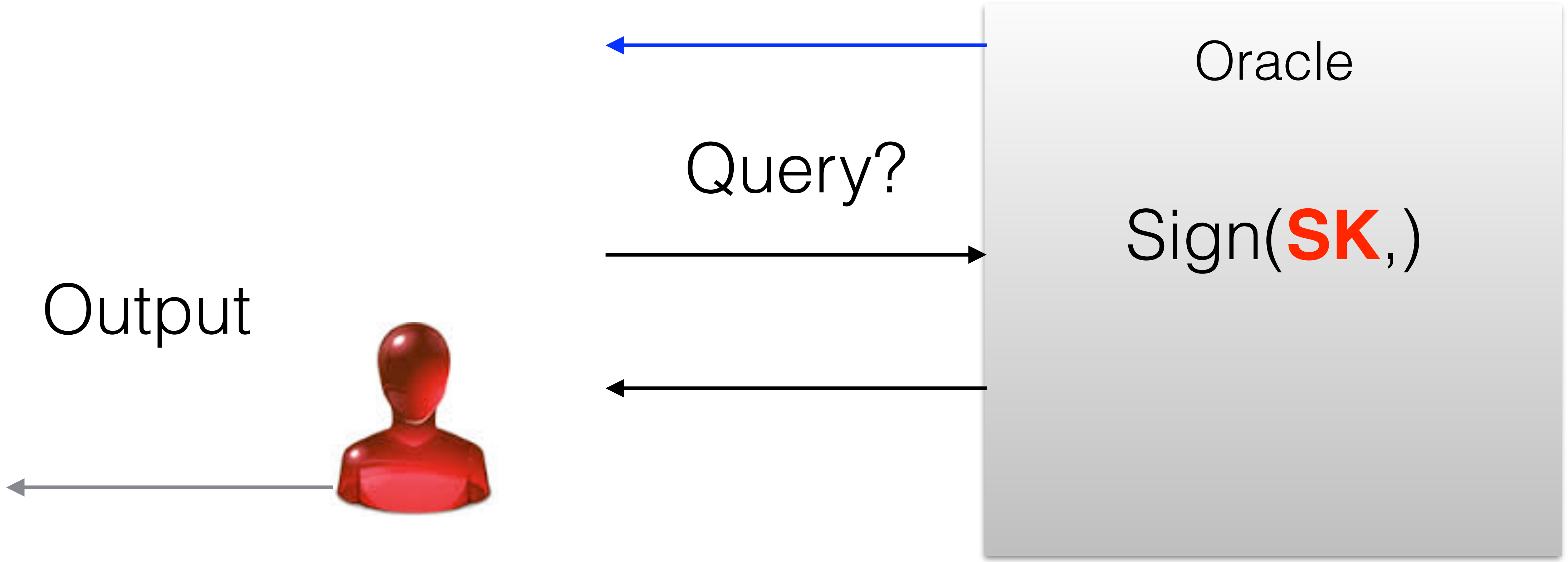
We want:

No-one should be able to compute signature on behalf of a certain PK

Even after seeing many signatures that verify with PK

Adv should not be able to **compute** a valid signature that verifies with PK.

Signature: Unforgeability Game



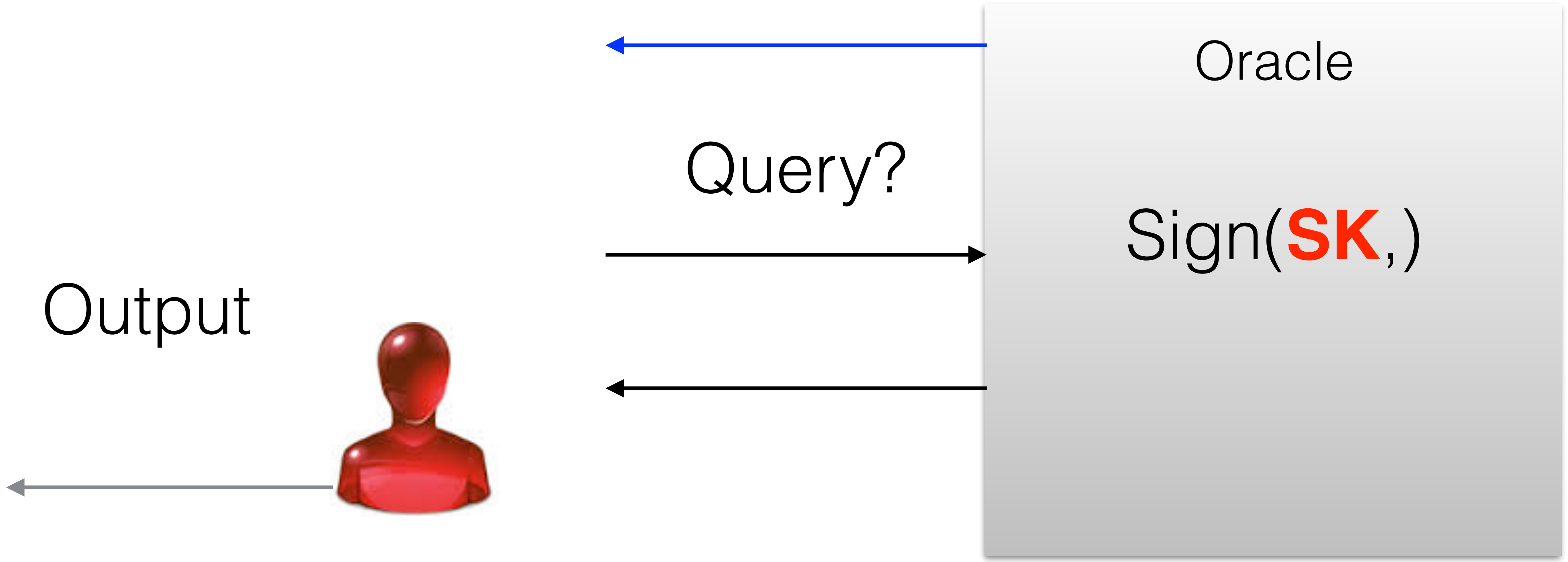
Oracle

Query?

Sign(**SK**,)

Output

Signature: Unforgeability Game



Oracle

Query?

Sign(**SK**,)

Output

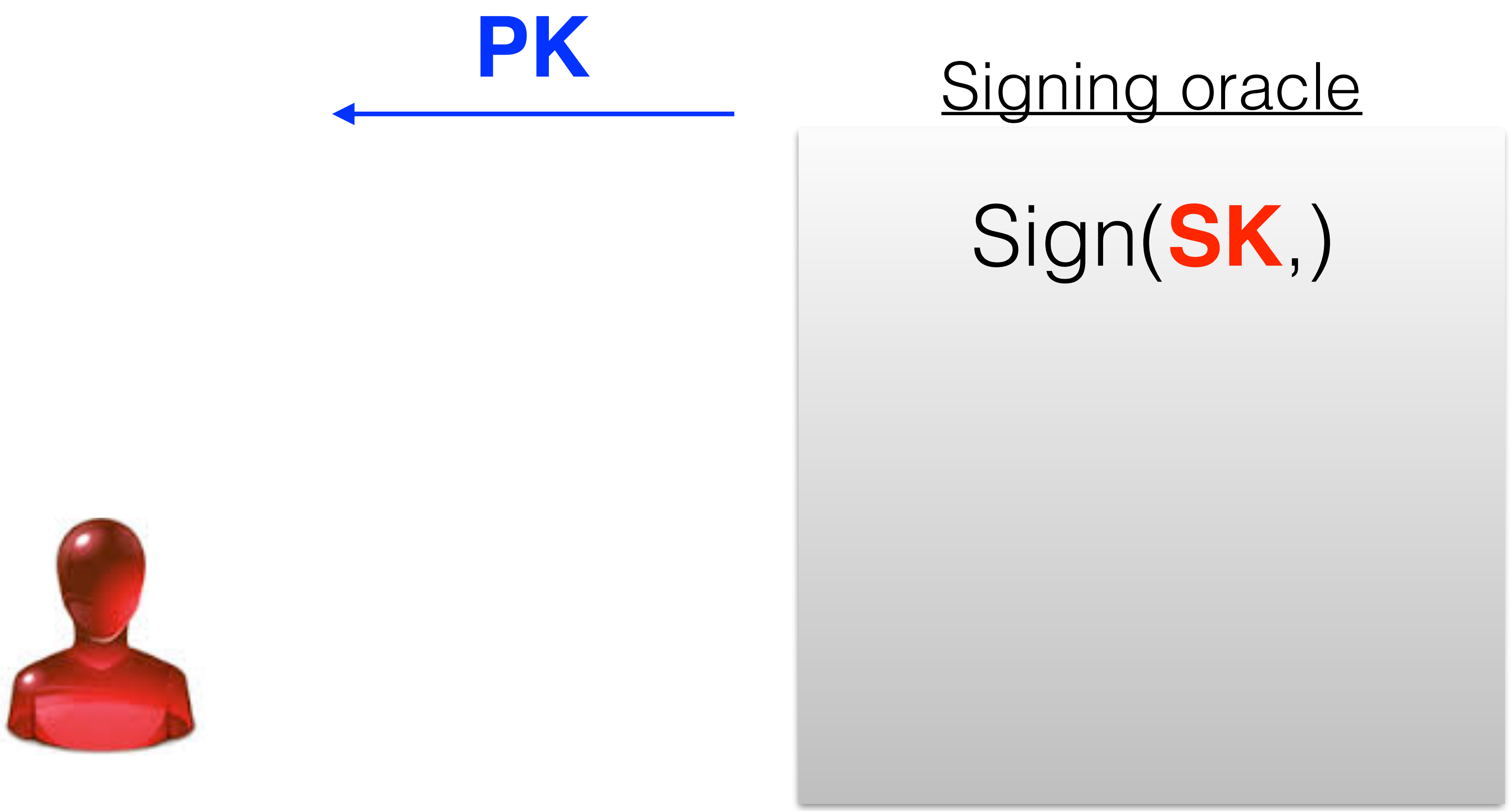
Winning condition?

Signature: Unforgeability Game

Signing oracle

Sign(**SK**,)

Signature: Unforgeability Game

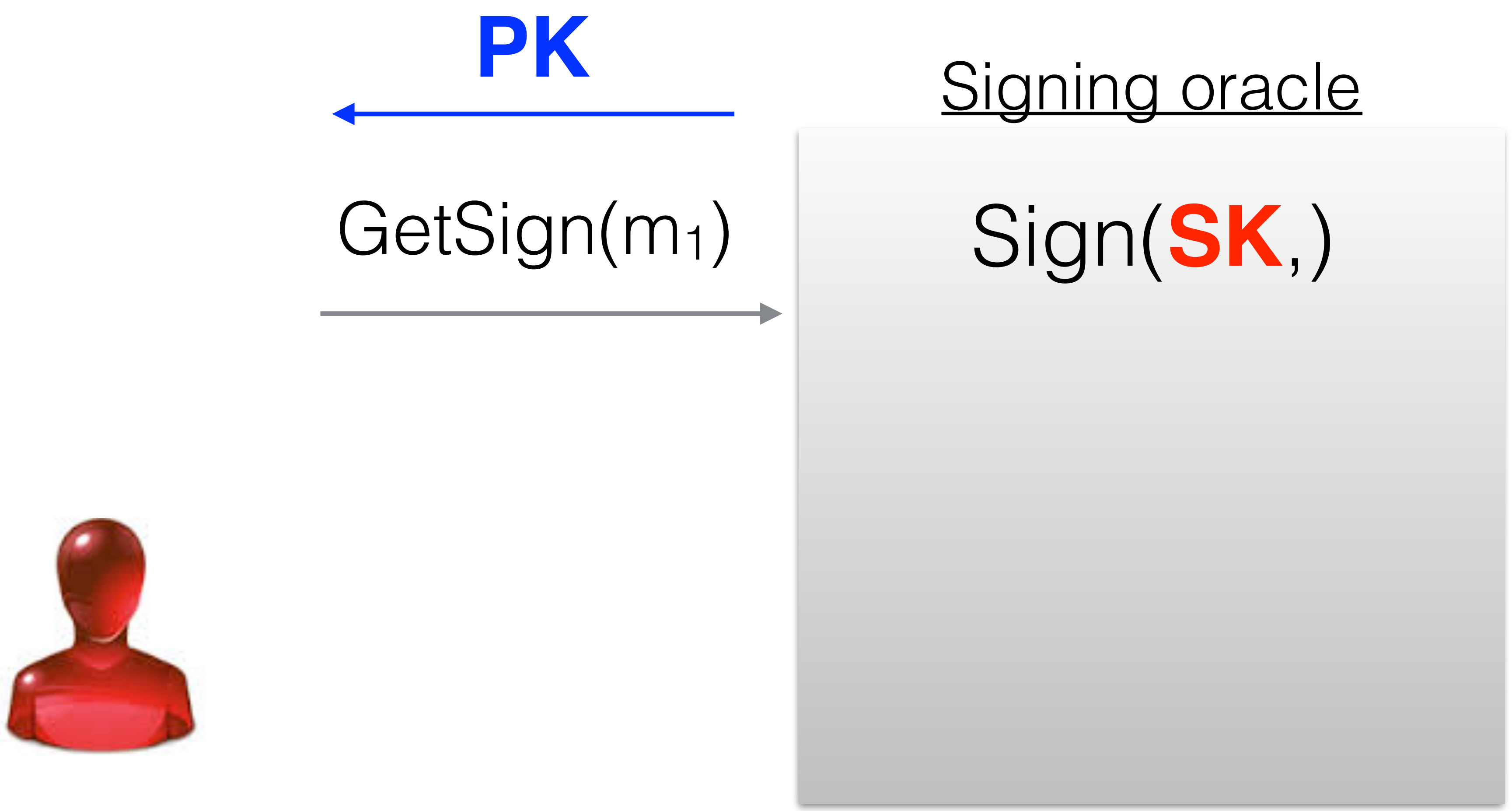


PK

Signing oracle

Sign(**SK**,)

Signature: Unforgeability Game



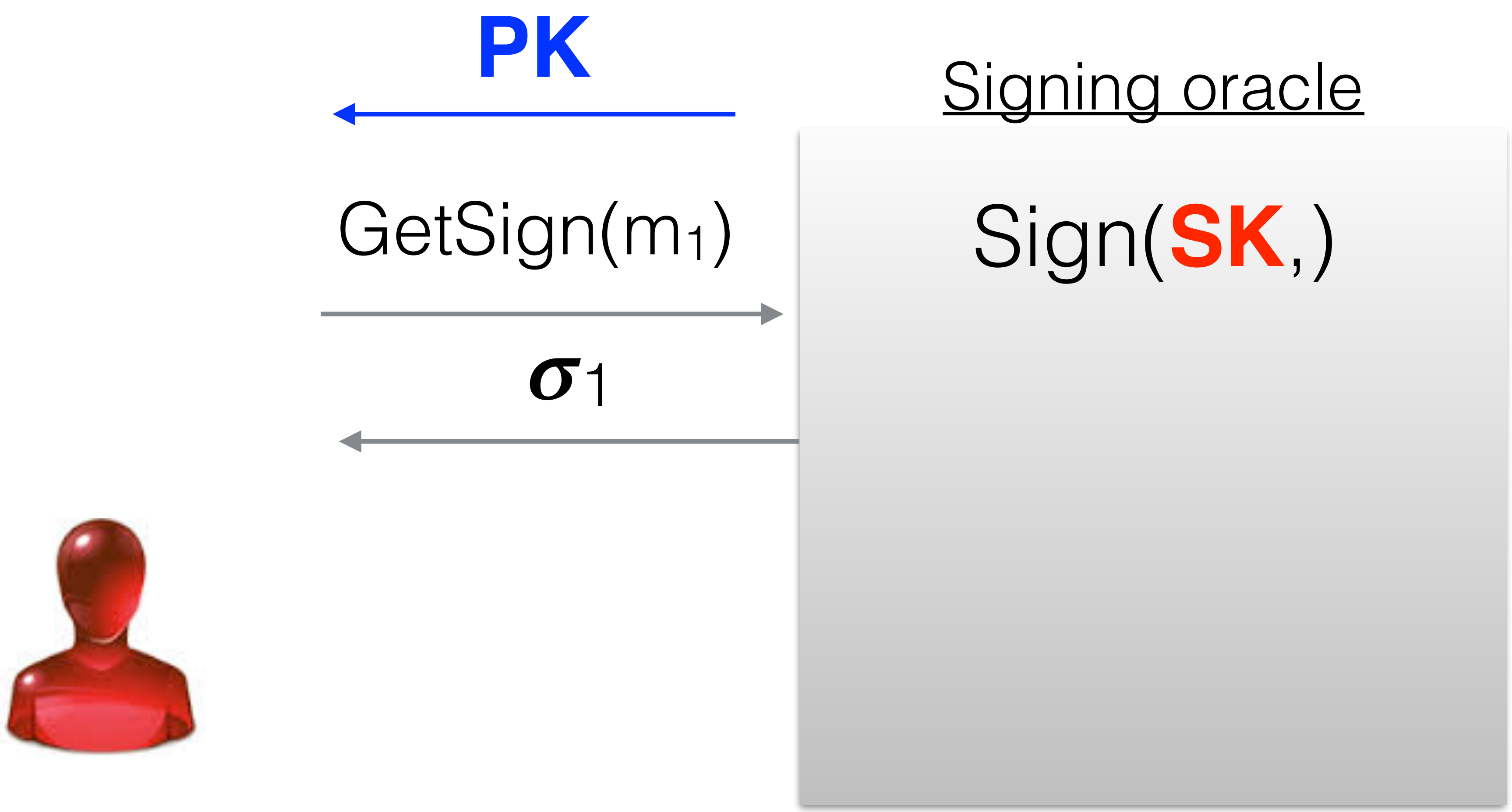
PK

Signing oracle

GetSign(m_1)

Sign(**SK**,)

Signature: Unforgeability Game



PK

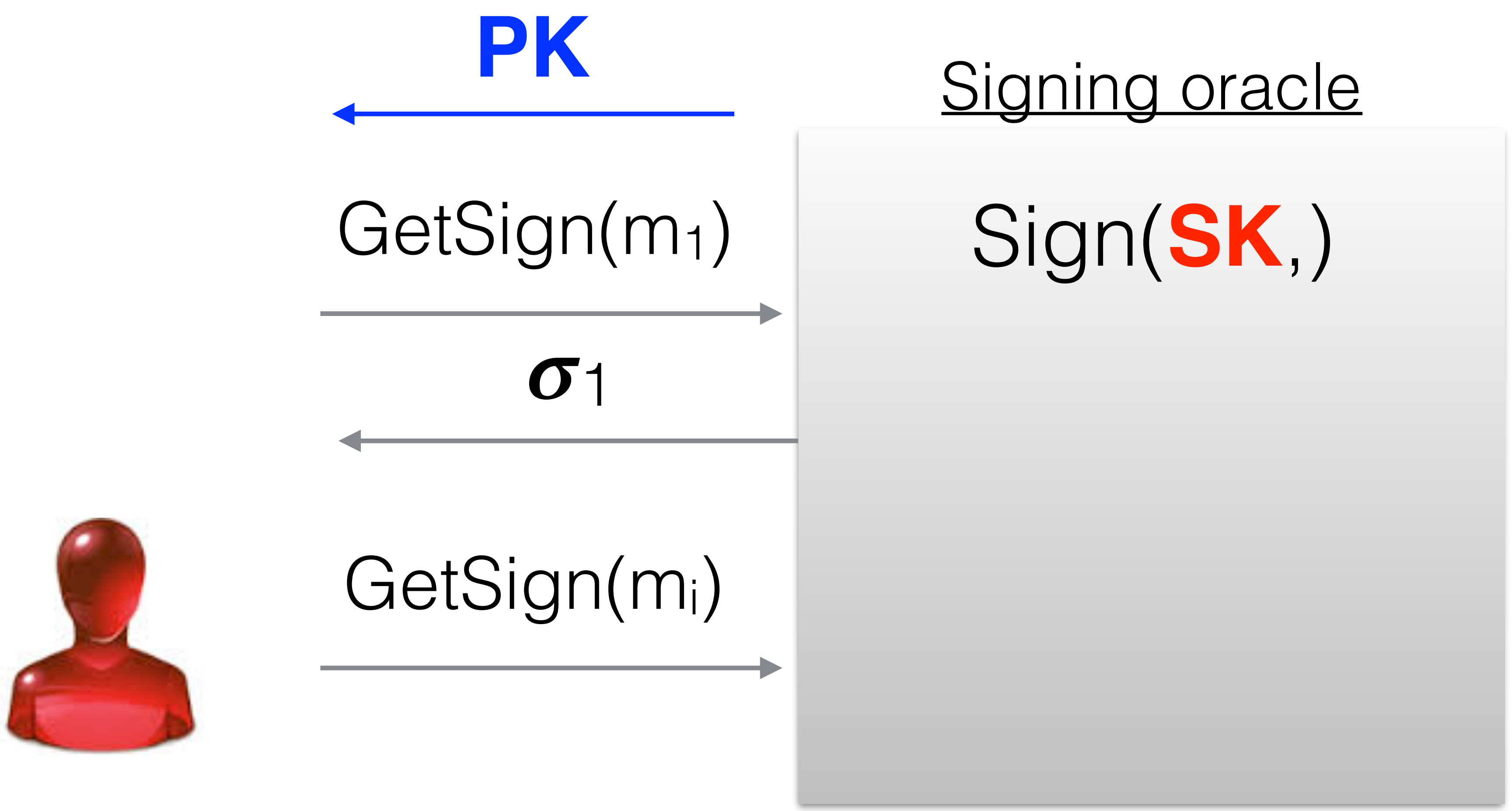
Signing oracle

GetSign(m_1)

σ_1

Sign(**SK**,)

Signature: Unforgeability Game



PK

Signing oracle

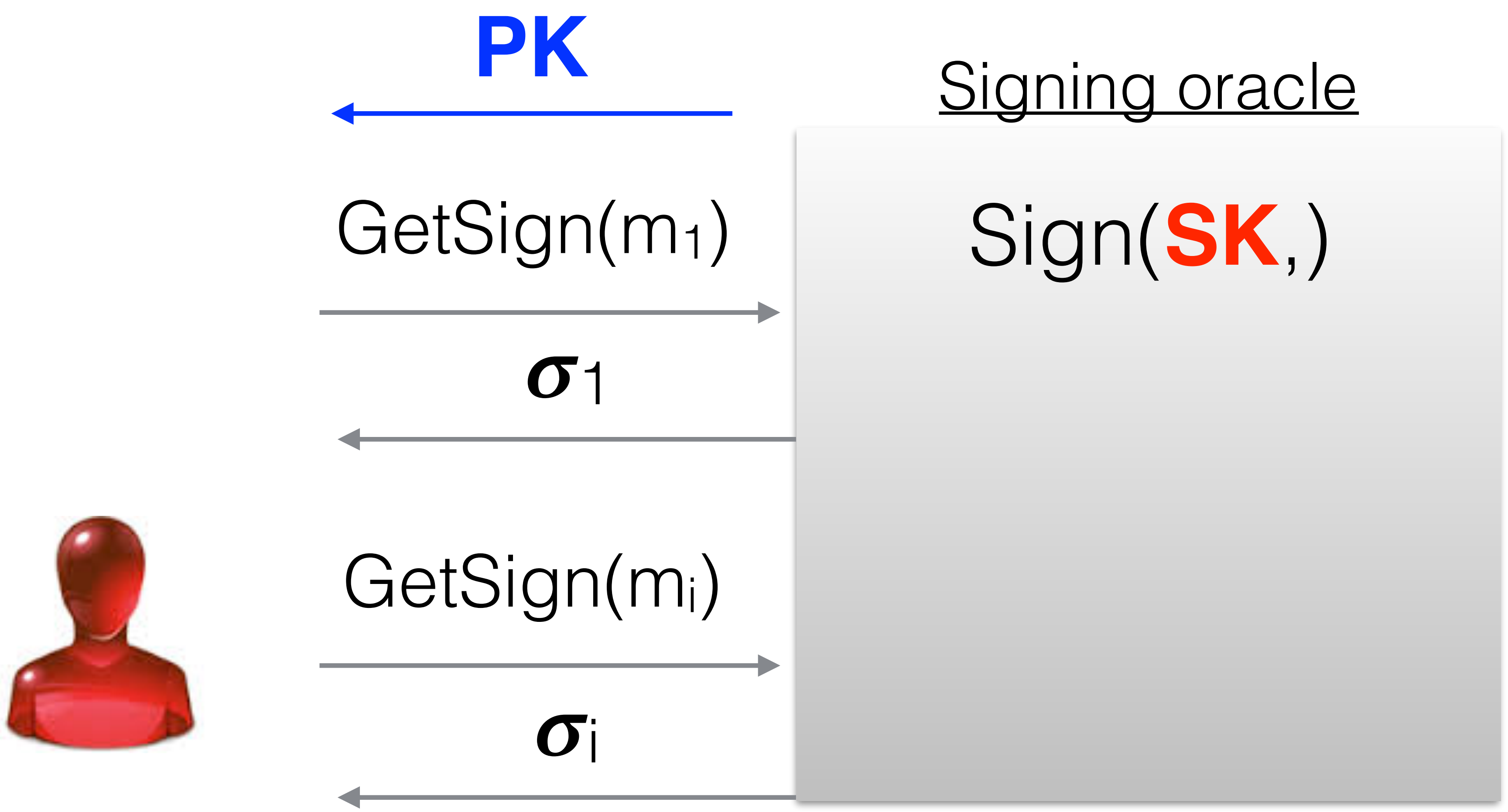
Sign(**SK**,)

GetSign(m_1)

σ_1

GetSign(m_i)

Signature: Unforgeability Game



PK

Signing oracle

Sign(**SK**,)

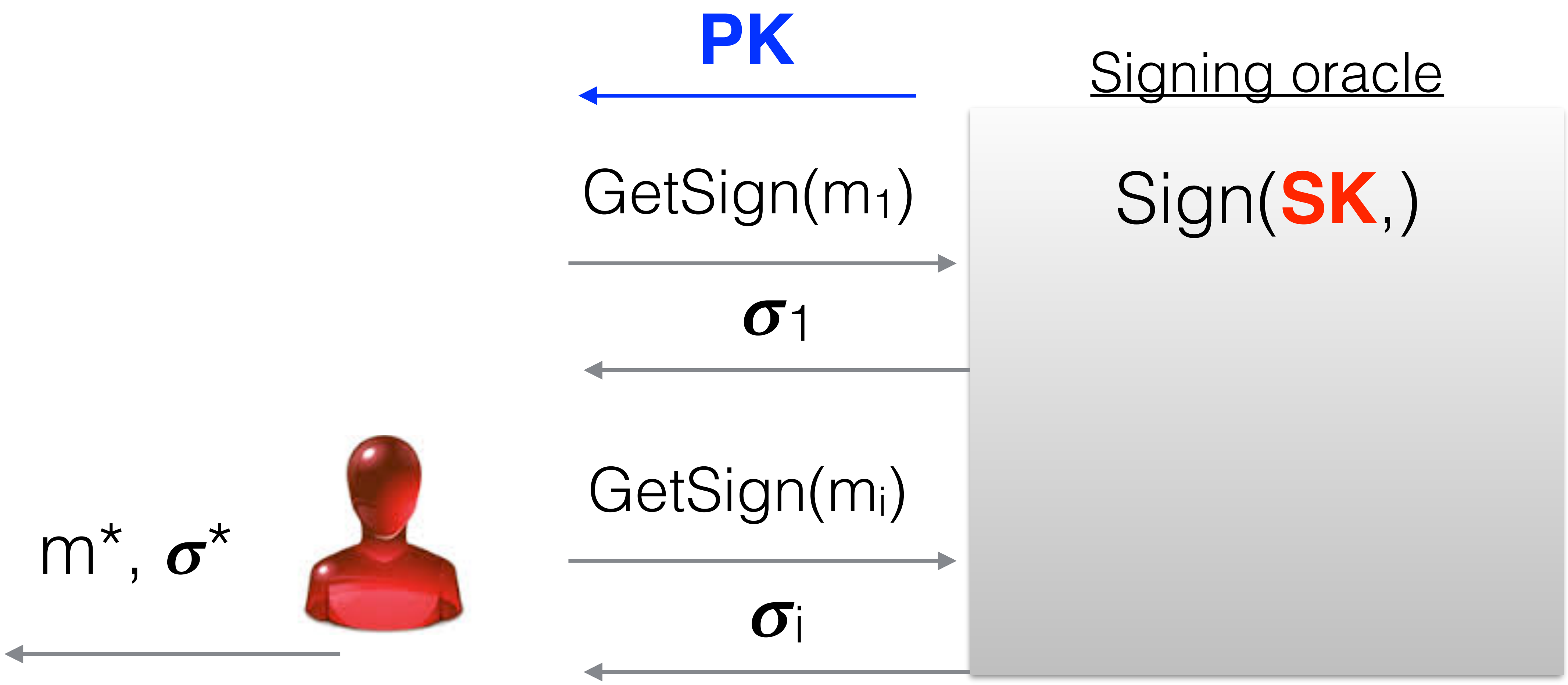
GetSign(m_1)

σ_1

GetSign(m_i)

σ_i

Signature: Unforgeability Game



PK

Signing oracle

Sign(**SK**,)

GetSign(m_1)

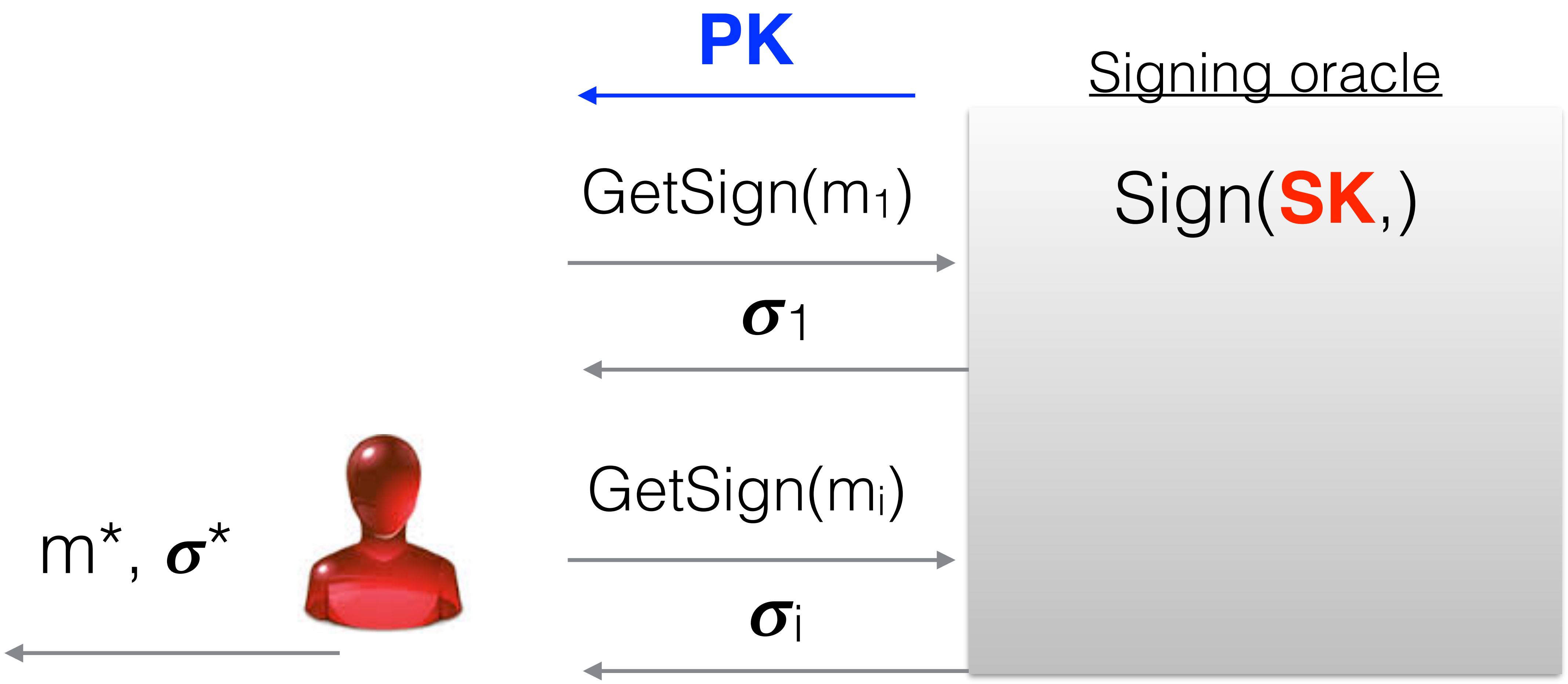
$\boldsymbol{\sigma}_1$

GetSign(m_i)

$\boldsymbol{\sigma}_i$

m^*, $\boldsymbol{\sigma}^*$

Signature: Unforgeability Game



PK

Signing oracle

Sign(**SK**,)

GetSign(m_1)

σ_1

GetSign(m_i)

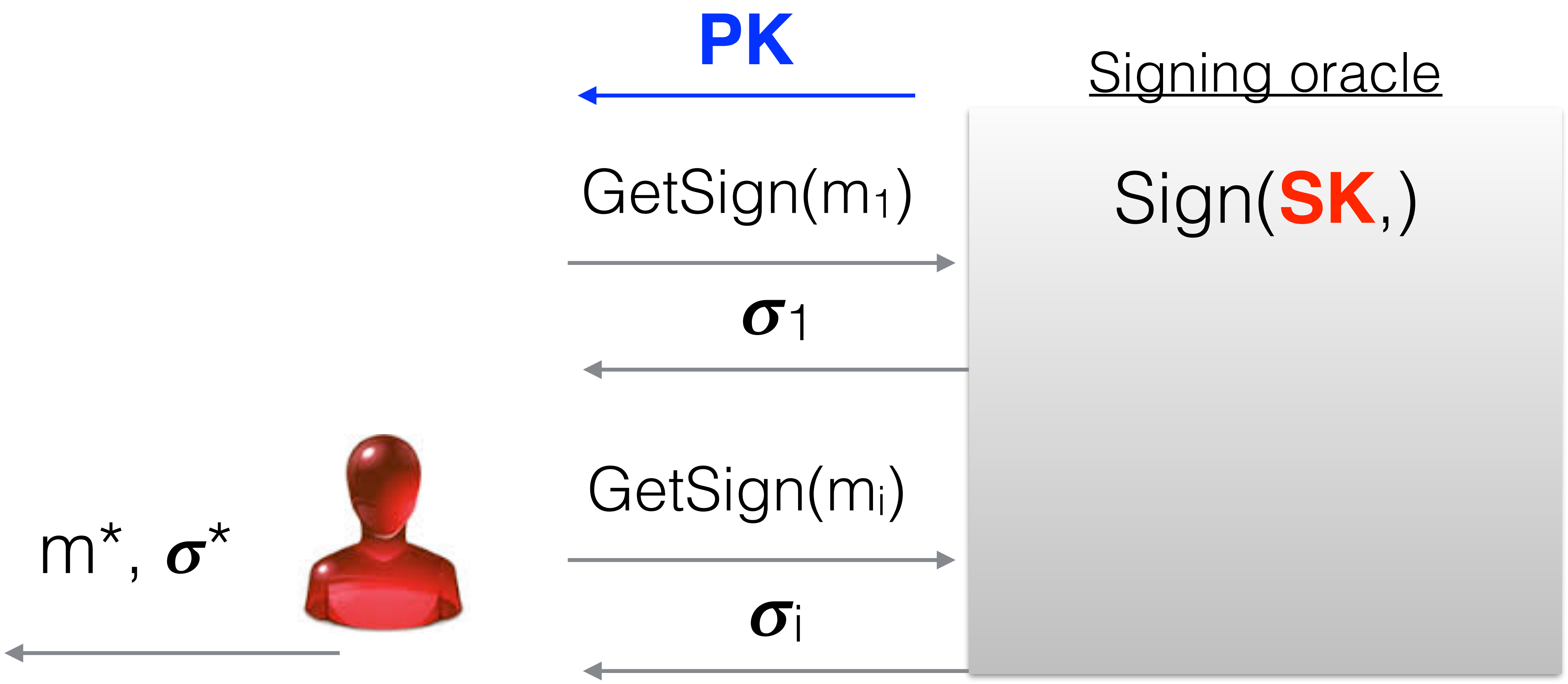
σ_i

m^*, σ^*

WIN if

Verify(PK,m^*, σ^*)=1

Signature: Unforgeability Game



PK

Signing oracle

Sign(**SK**,)

GetSign(m_1)

σ_1

GetSign(m_i)

σ_i

m^*, σ^*

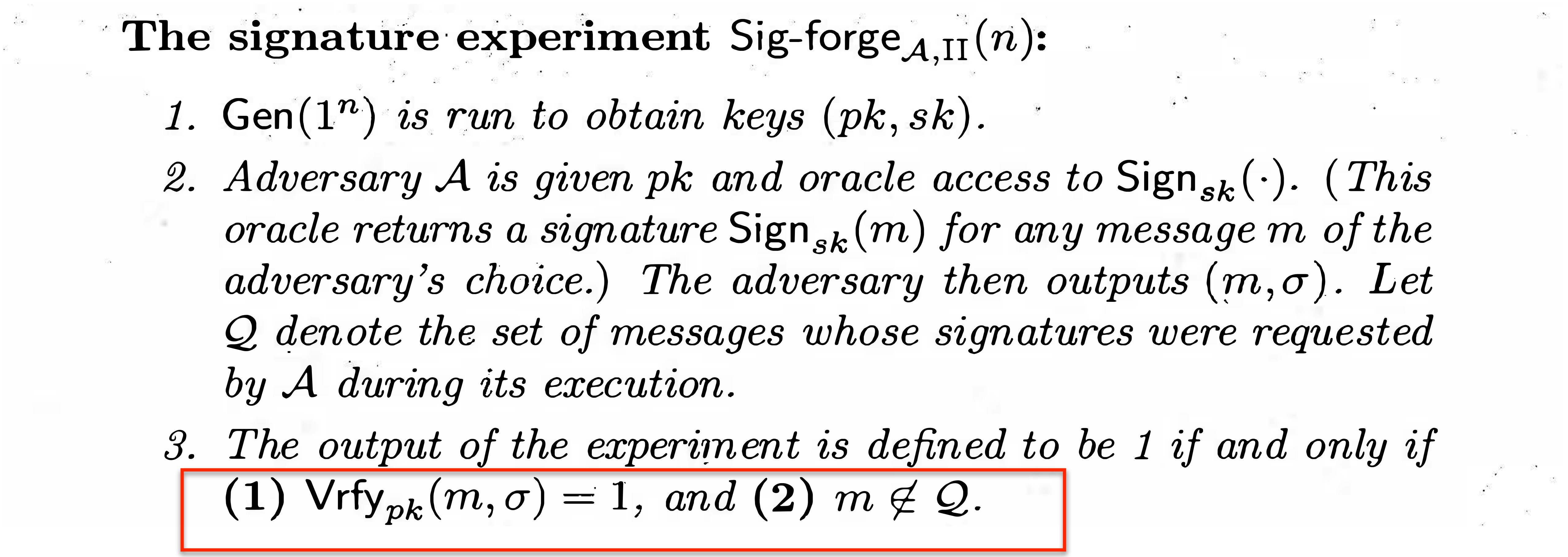
WIN if

Verify(PK,m^*, σ^*)=1

and

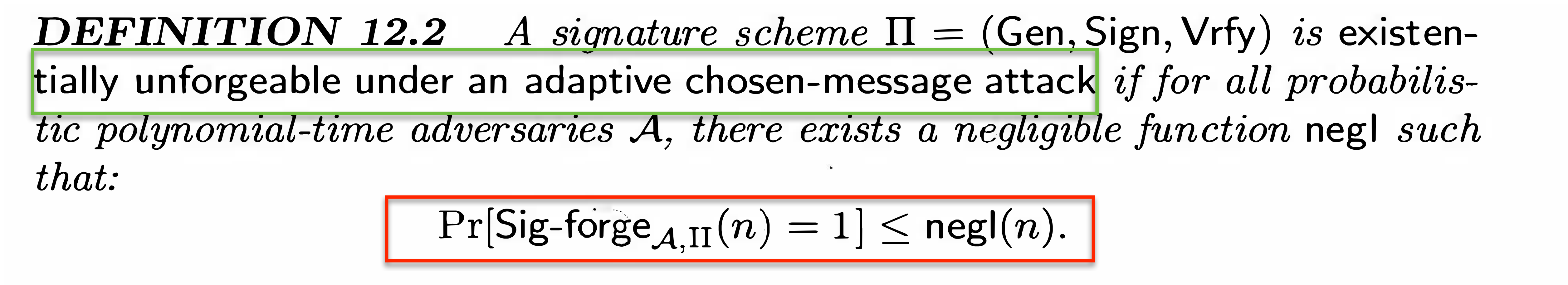
m^* was *never asked to the oracle*

Definition from Introduction to Modern Cryptography



The signature experiment $\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n)$**:**

1. $\mathsf{Gen}(1^n)$ *is run to obtain keys* (pk, sk).
2. *Adversary* $\mathcal{A}$ *is given* pk *and oracle access to* $\mathsf{Sign}_{sk}(\cdot)$. *(This oracle returns a signature* $\mathsf{Sign}_{sk}(m)$ *for any message* m *of the adversary's choice.) The adversary then outputs* (m, σ). *Let* $\mathcal{Q}$ *denote the set of messages whose signatures were requested by* $\mathcal{A}$ *during its execution.*
3. *The output of the experiment is defined to be 1 if and only if* **(1)** $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$, *and* **(2)** $m \notin \mathcal{Q}$.



DEFINITION 12.2 *A signature scheme* $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* existentially unforgeable under an adaptive chosen-message attack *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$, *there exists a negligible function* negl *such that:*

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

Discussion: Signature Scheme VS MAC

☑ Publicly verifiable

☑ Easier Key distribution

☑ ____________________

☑ ____________________

Discussion: Signature Scheme VS MAC

- ☑ Publicly verifiable
- ☑ Easier Key distribution
- ☑ Non-repudiation
- ☑ Transferable

Digital Signature

- Definition   Unforgeability

- Construction

  RSA + Hash

  One-way Functions

Digital Signature

- Definition Unforgeability

- Construction

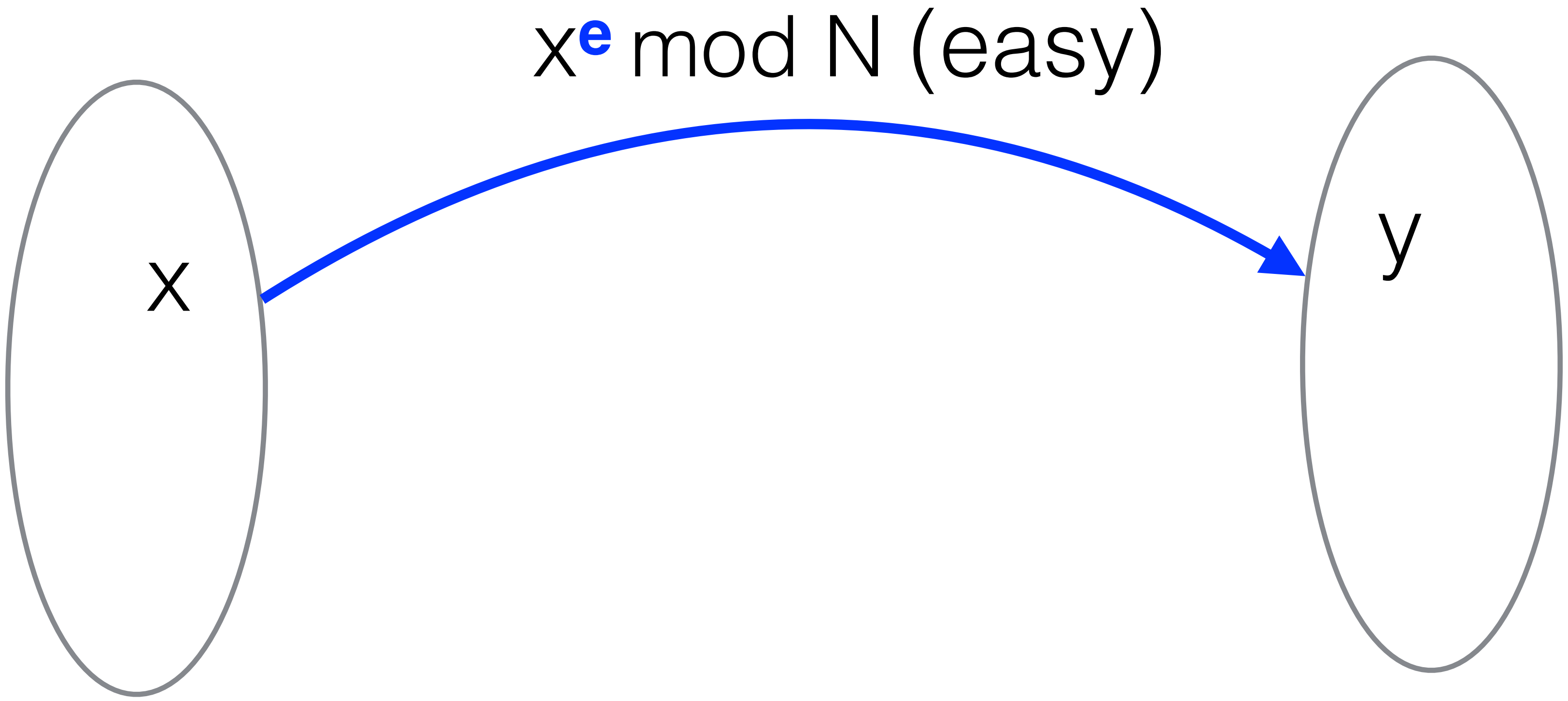
  RSA + Hash

  One-way Functions

RSA **trapdoor** **one-way** function

Z^*_N

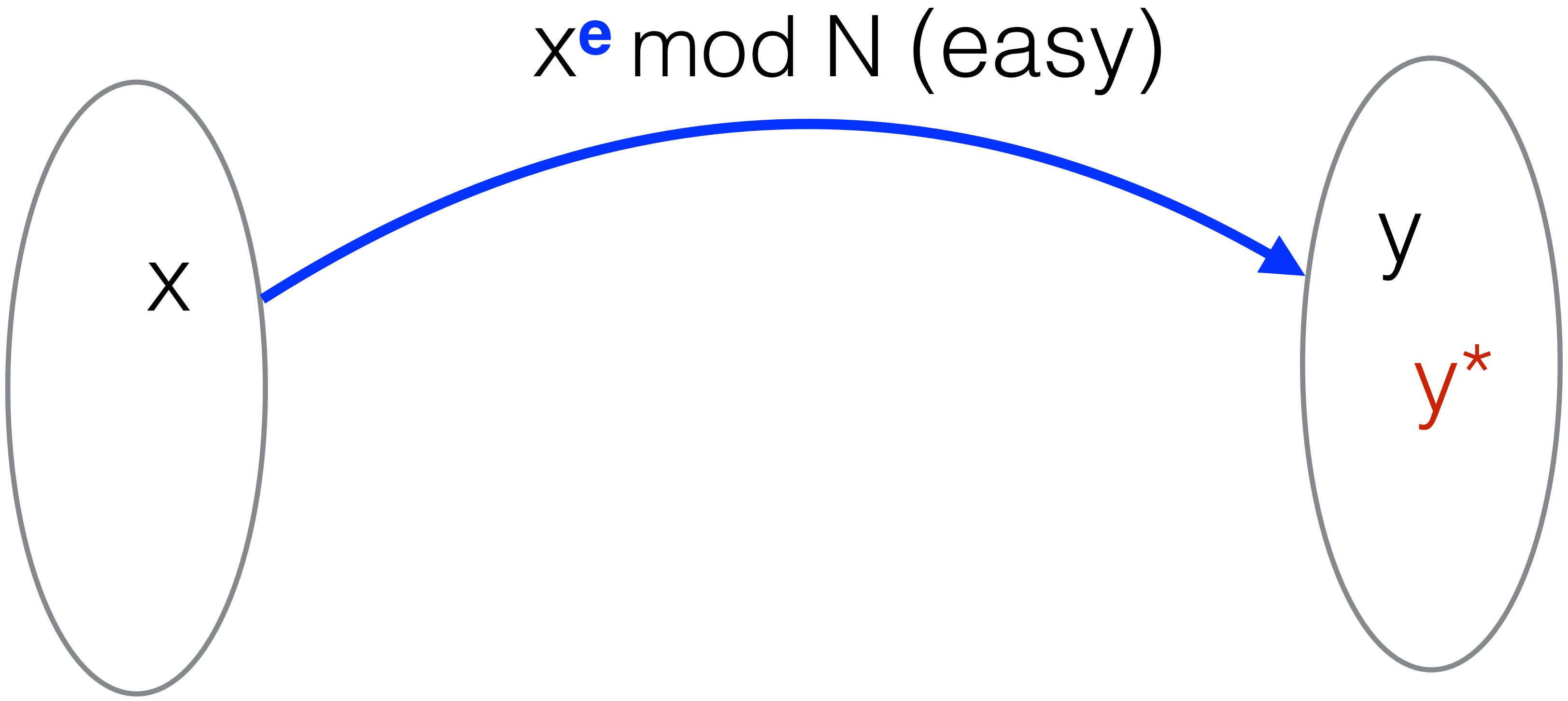
PK= N,e **SK= d**



RSA **trapdoor** **one-way** function

Z^*_N

PK= N,e **SK= d**



x^e mod N (easy)

x

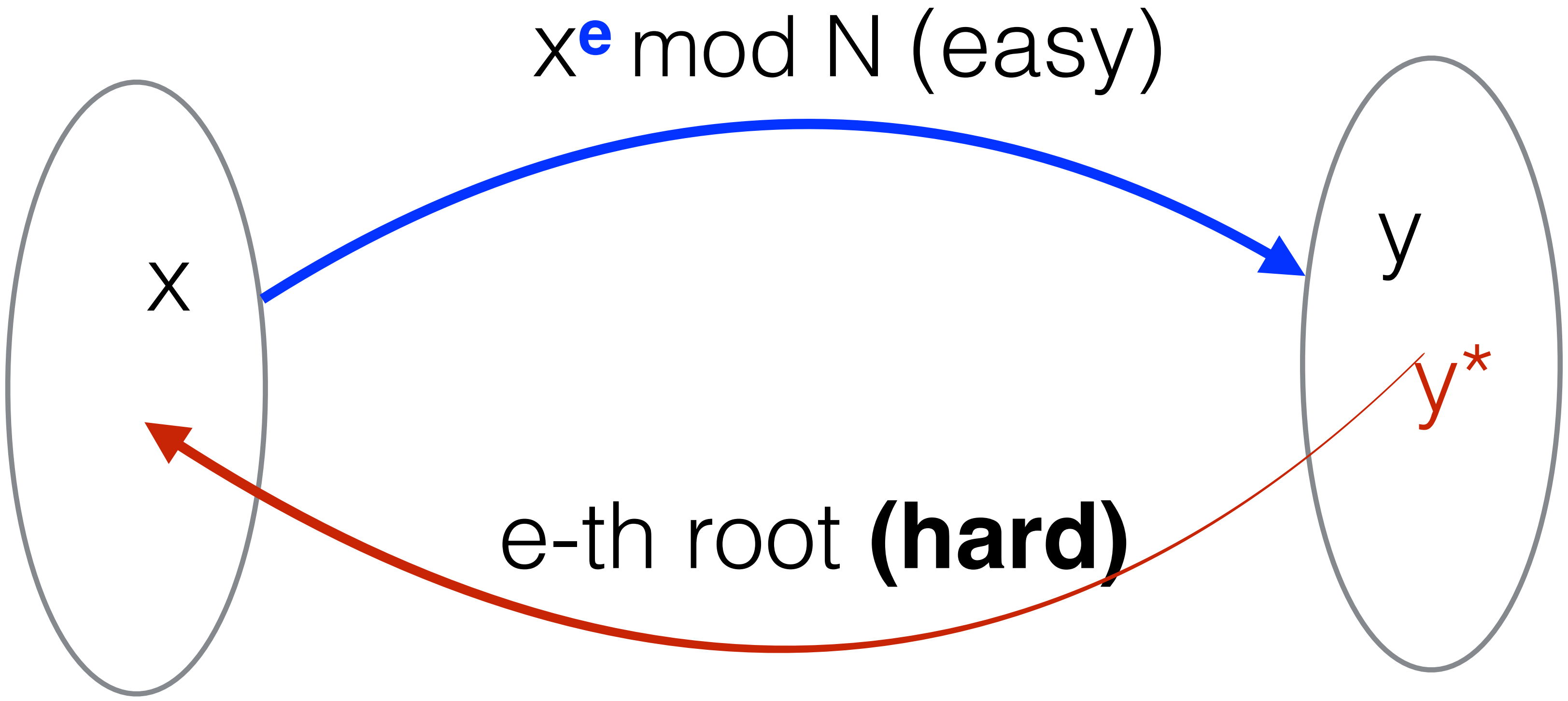
y

y*

RSA **trapdoor** **one-way** function

Z^*_N

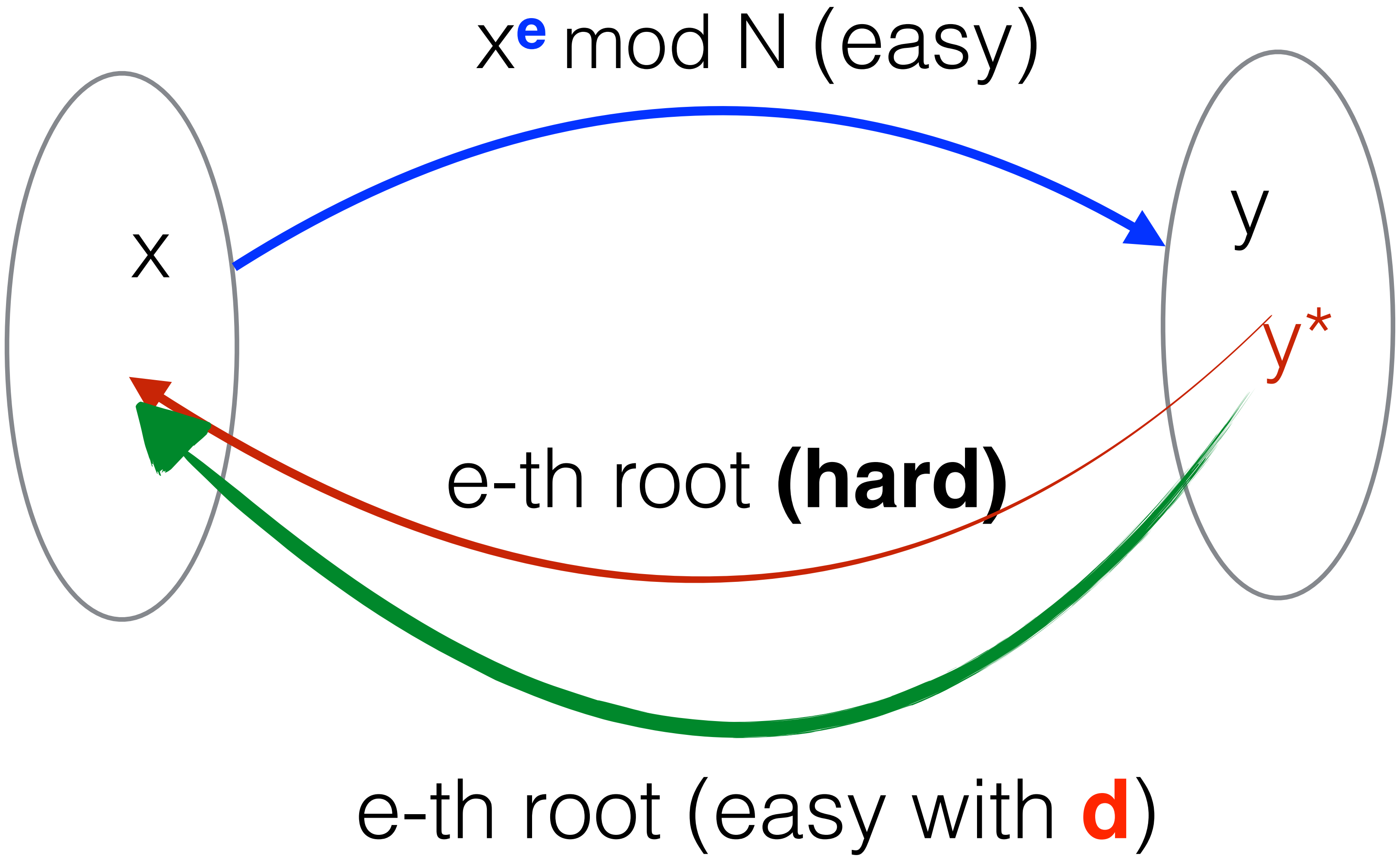
PK= N,e **SK= d**



RSA **trapdoor** **one-way** function

Z^*_N

PK= N,e **SK= d**



$x^{\mathbf{e}}$ mod N (easy)

x

y

y*

e-th root **(hard)**

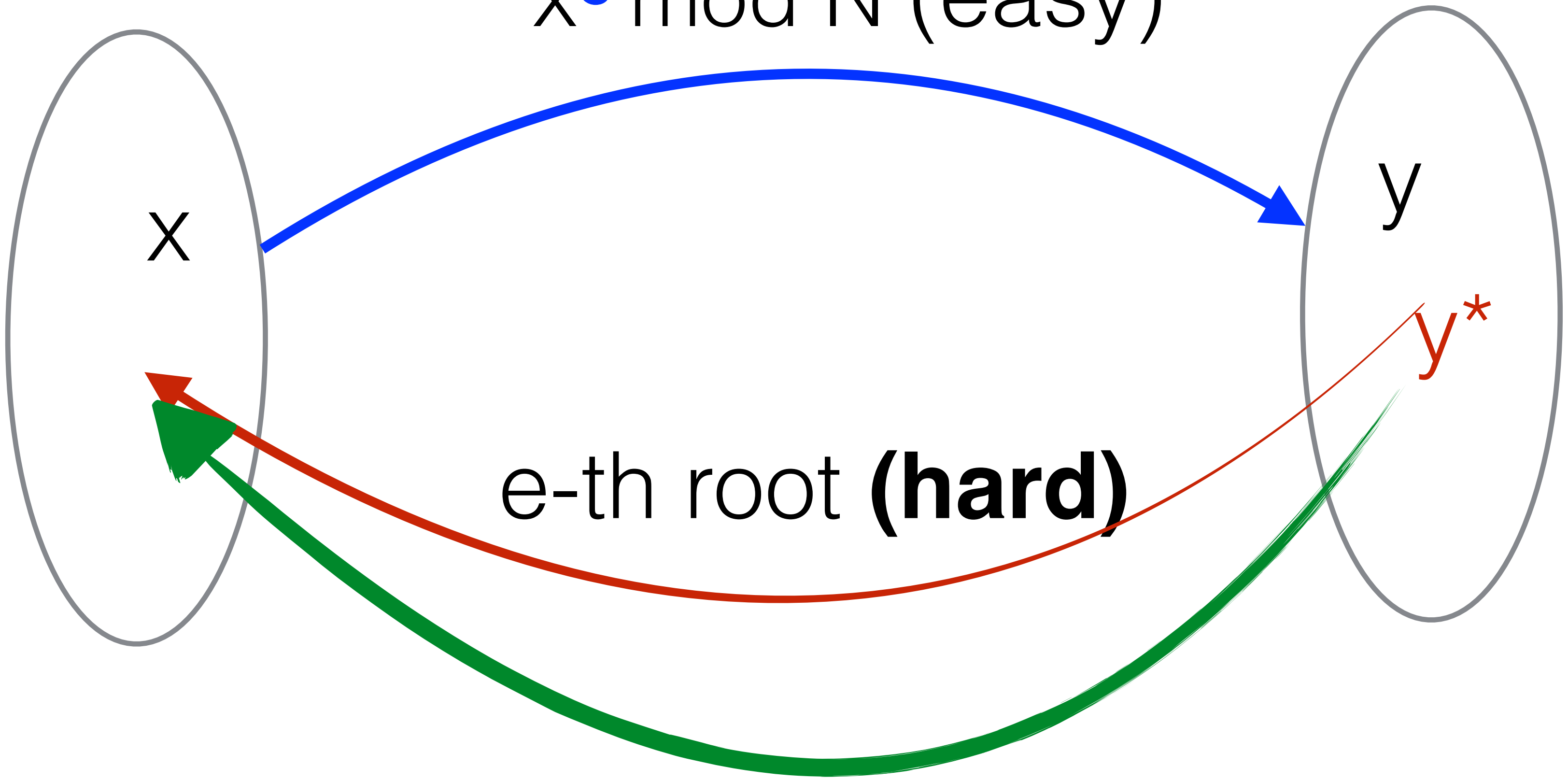
e-th root (easy with **d**)

RSA **trapdoor** **one-way** function

Z^*_N

PK= N,e **SK= d**

Verify should be easy



$x^{\mathbf{e}}$ mod N (easy)

x

y

y*

e-th root **(hard)**

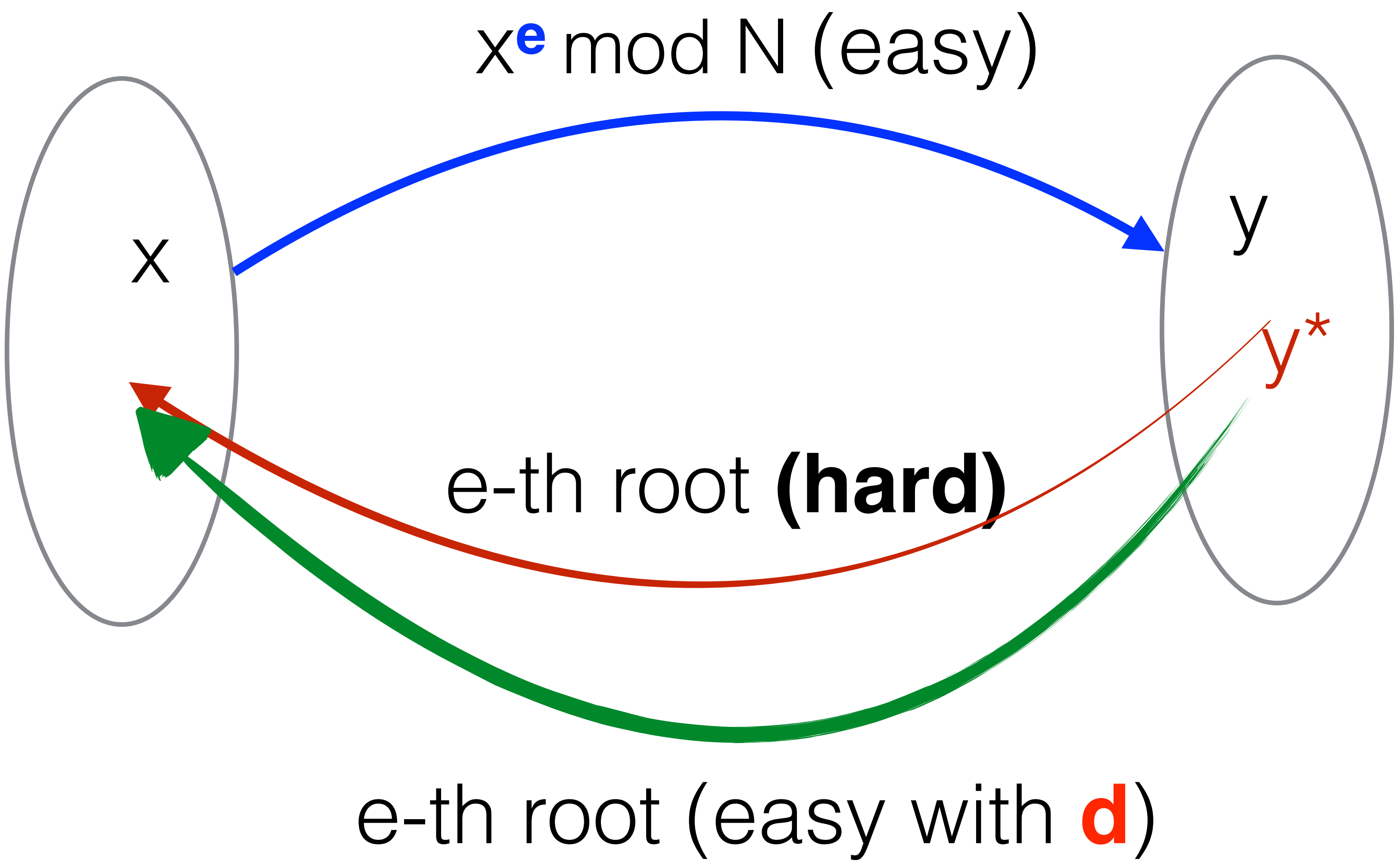
e-th root (easy with **d**)

RSA **trapdoor** **one-way** function

Z^*_N

PK= N,e **SK= d**

Verify should be easy

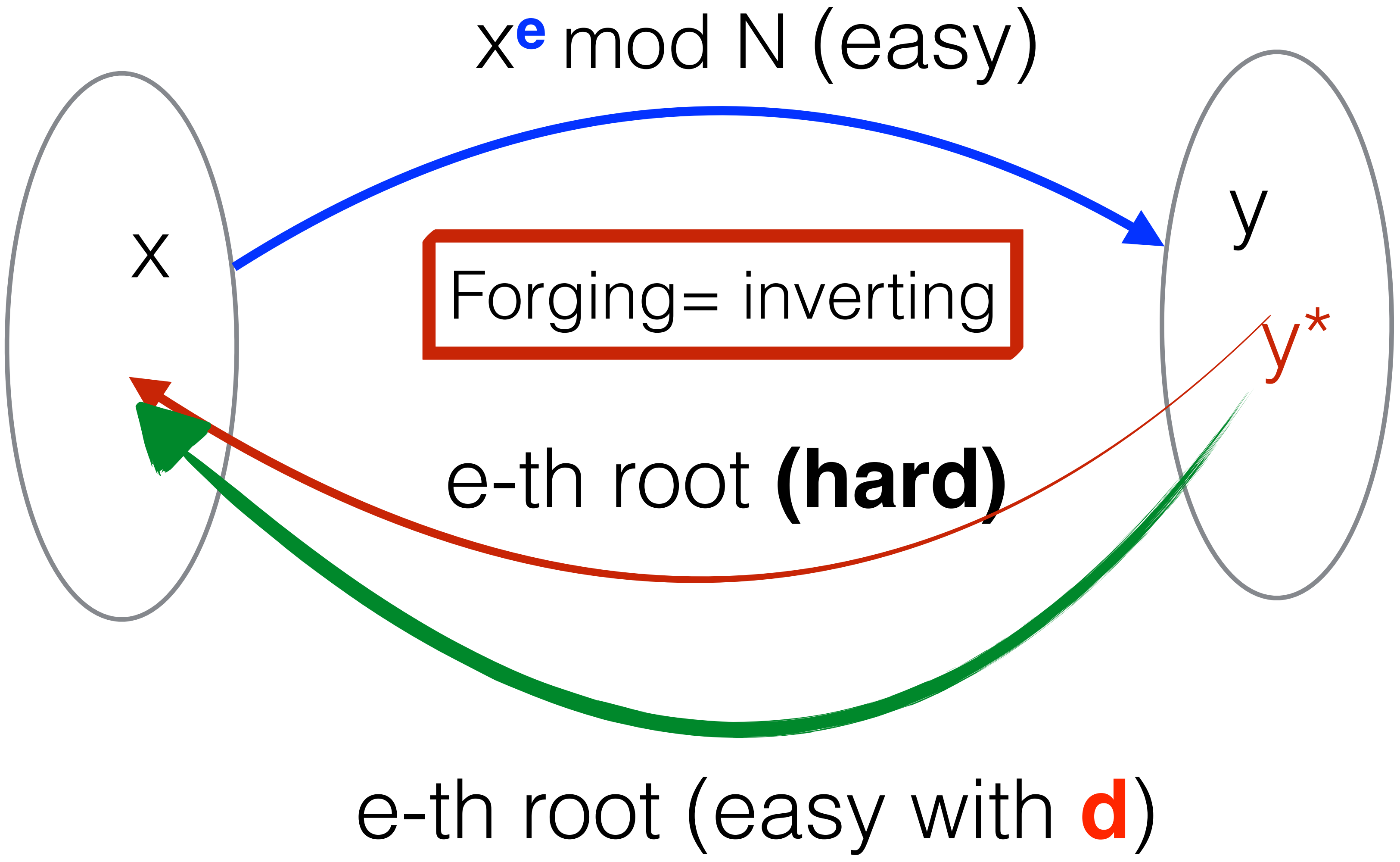


Signing only with secret key **d**

RSA **trapdoor** **one-way** function

Z^*_N

PK= N,e **SK= d**



Verify should be easy

x^e mod N (easy)

Forging= inverting

e-th root **(hard)**

e-th root (easy with **d**)

Signing only with secret key **d**

Digital Signature from RSA

GenKey(n) = GenRSA

PK= N,e

SK= d

Sign(m ,d)

Verify(σ ,m, N,e)

Digital Signature from RSA

GenKey(n) = GenRSA

PK= N,e

SK= d

Sign(m ,d)

$\boldsymbol{\sigma}$ = $m^{\mathbf{d}}$ mod N

Verify($\boldsymbol{\sigma}$,m, N,e)

Output [m == $\boldsymbol{\sigma}^{\mathbf{e}}$ mod N]

Digital Signature from RSA

GenKey(n) = GenRSA

PK= N,e

SK= d

Sign(m ,d)

$\sigma = m^{d} \bmod N$

Verify(σ ,m, N,e)

Output [$m == \sigma^{e} \bmod N$]

Unforgeable?

In class exercise

Forge Textbook RSA signature scheme

Why is the adversary able to sign her own messages?

Why is the adversary able to sign her own messages?

Adversary **decides the values that are exponentiated** and can use this information by leveraging the algebraic structure of the signature.

Why is the adversary able to sign her own messages?

Adversary **decides the values that are exponentiated** and can use this information by leveraging the algebraic structure of the signature.

How to fix it?

Why is the adversary able to sign her own messages?

Adversary **decides the values that are exponentiated** and can use this information by leveraging the algebraic structure of the signature.

How to fix it?

Preprocess values that are exponentiated so that they are random and **out of the control** of the adversary

RSA -FDH PKCS#1 v2.1

GenKey(n) = GenRSA

PK= N,e **SK= d**

H

Sign(m ,d)

Verify(σ ,m, N,e)

RSA -FDH PKCS#1 v2.1

GenKey(n) = GenRSA

PK= N,e

SK= d

H

Sign(m ,d)

$\boxed{y = H(m)}$

Verify(σ ,m, N,e)

RSA -FDH PKCS#1 v2.1

GenKey(n) = GenRSA

PK= N,e **SK= d**

H

Sign(m ,d)

$$\boxed{y = H(m)}$$

$$\sigma = y^{d} \bmod N$$

Verify(σ ,m, N,e)

RSA -FDH PKCS#1 v2.1

GenKey(n) = GenRSA

PK= N,e **SK= d**

H

Sign(m ,d)

$$\boxed{y = H(m)}$$

$$\boldsymbol{\sigma} = y^{\mathbf{d}} \bmod N$$

Verify($\boldsymbol{\sigma}$,m, N,e)

$$\boxed{y = H(m)}$$

RSA -FDH PKCS#1 v2.1

GenKey(n) = GenRSA

PK= N,e **SK= d**

H

Sign(m ,d)

$$\boxed{y = H(m)}$$

$$\sigma = y^{d} \bmod N$$

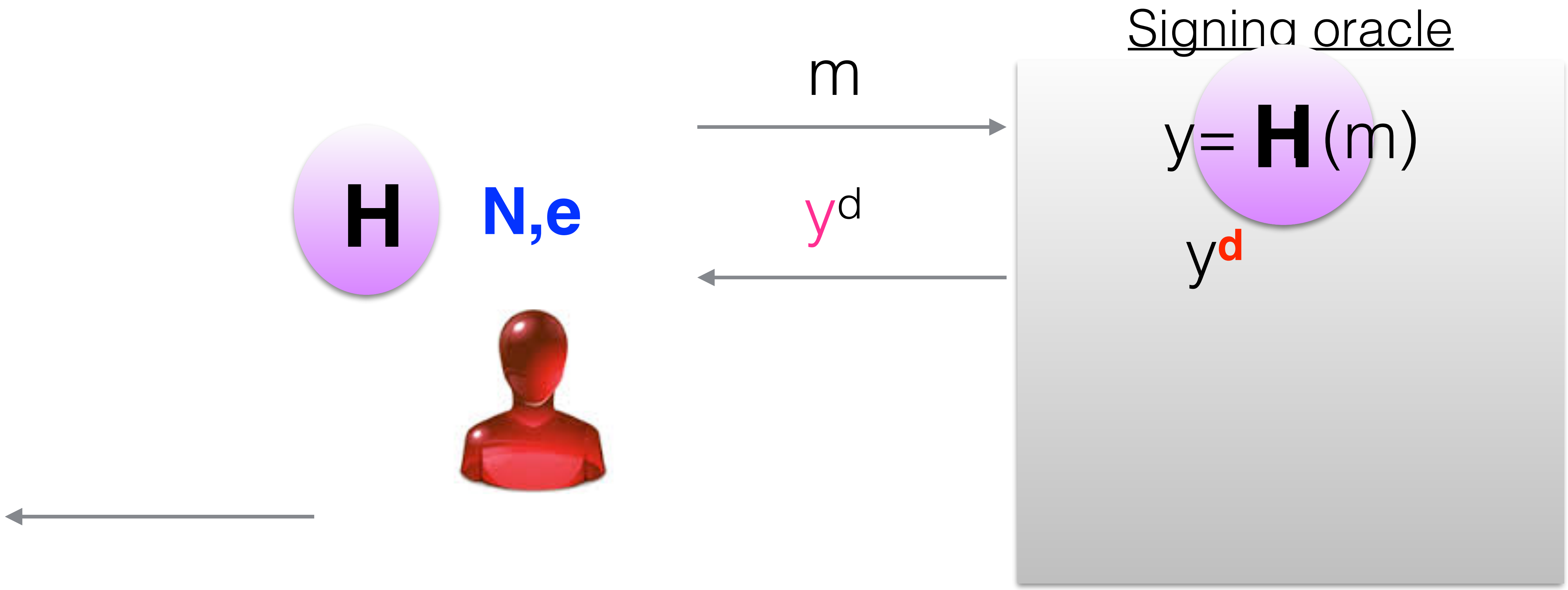
Verify(σ ,m, N,e)

$$\boxed{y = H(m)}$$

$$y = \sigma^{e} \bmod N$$

Why does it help?

Intuition



H

N,e

m

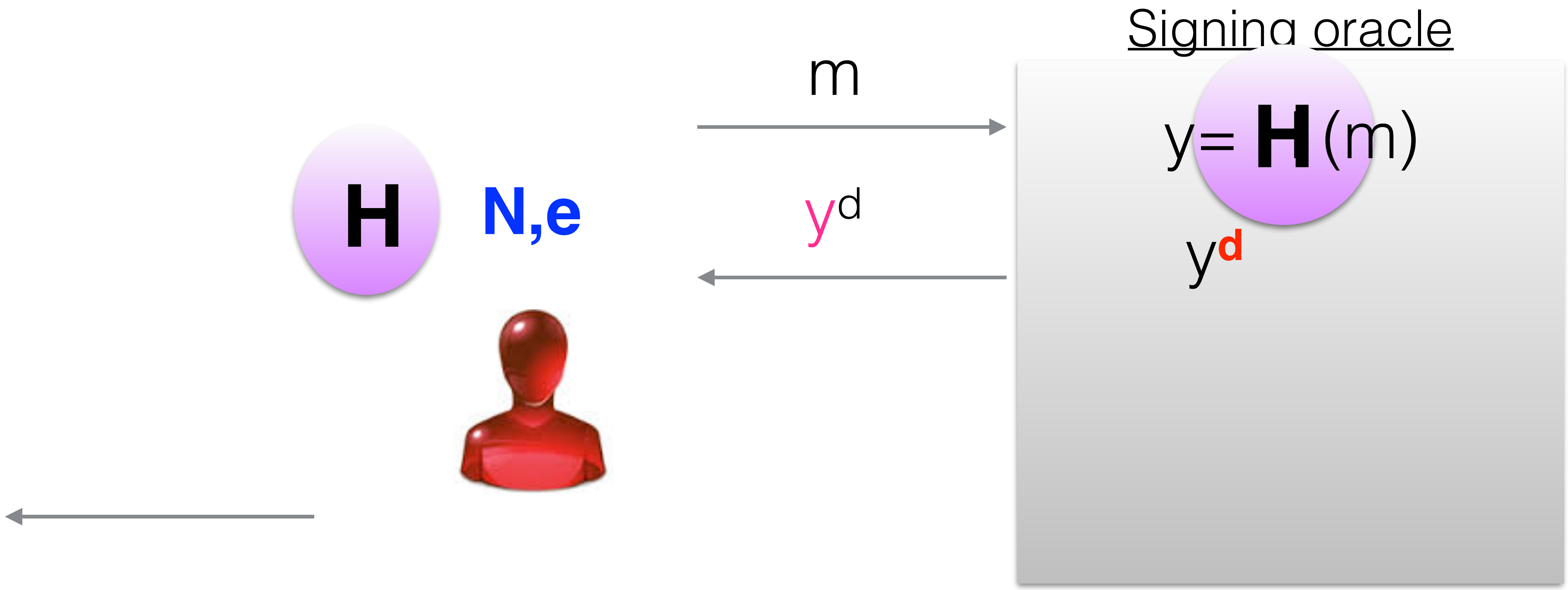
y^d

Signing oracle

$y = \mathbf{H}(m)$

$y^{\mathbf{d}}$

Intuition



Signing oracle

H N,e

m

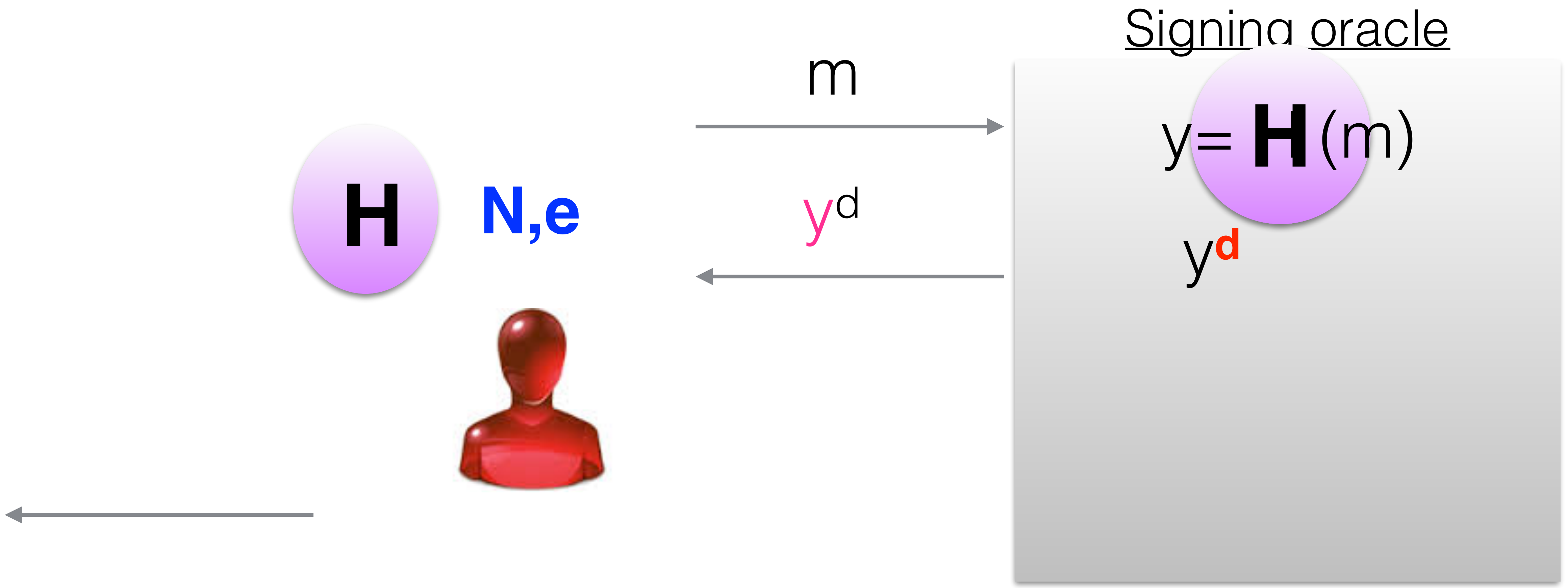
y^d

$y = H(m)$

y^d

How can the adversary find a forgery now?

Intuition



Signing oracle

m

H N,e

y^d

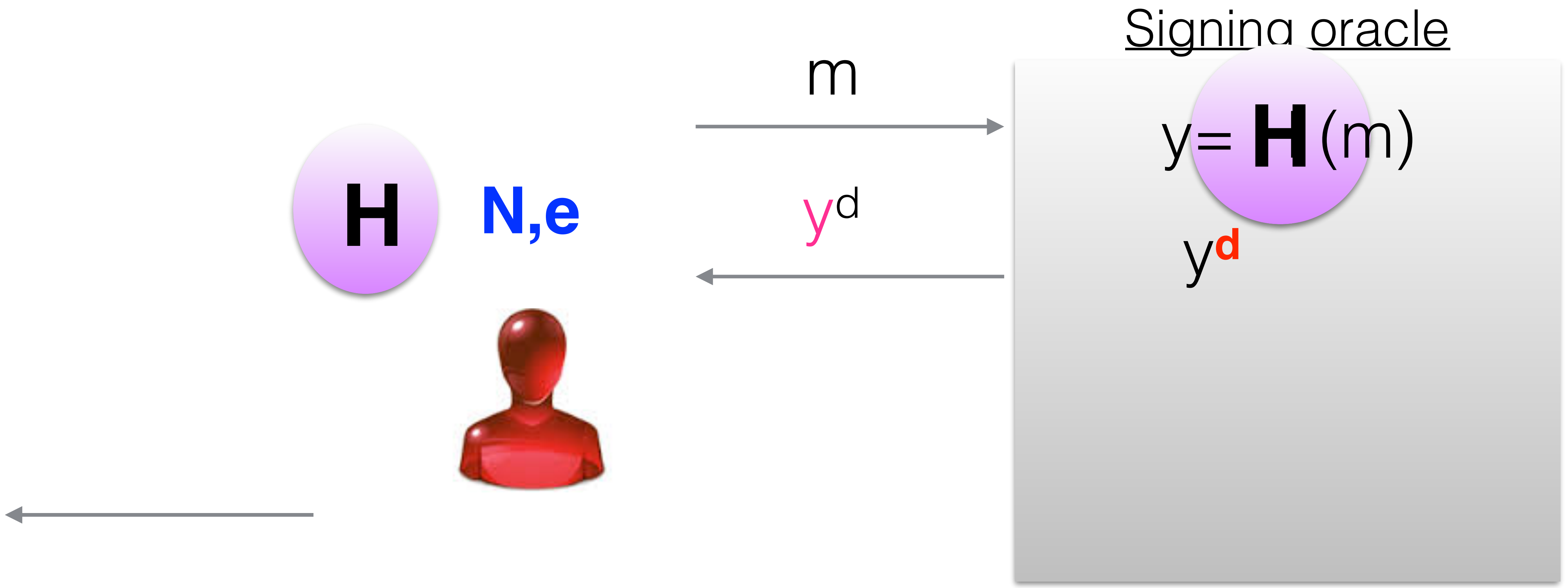
$y = H(m)$

y^d

How can the adversary find a forgery now?

- finds a collision

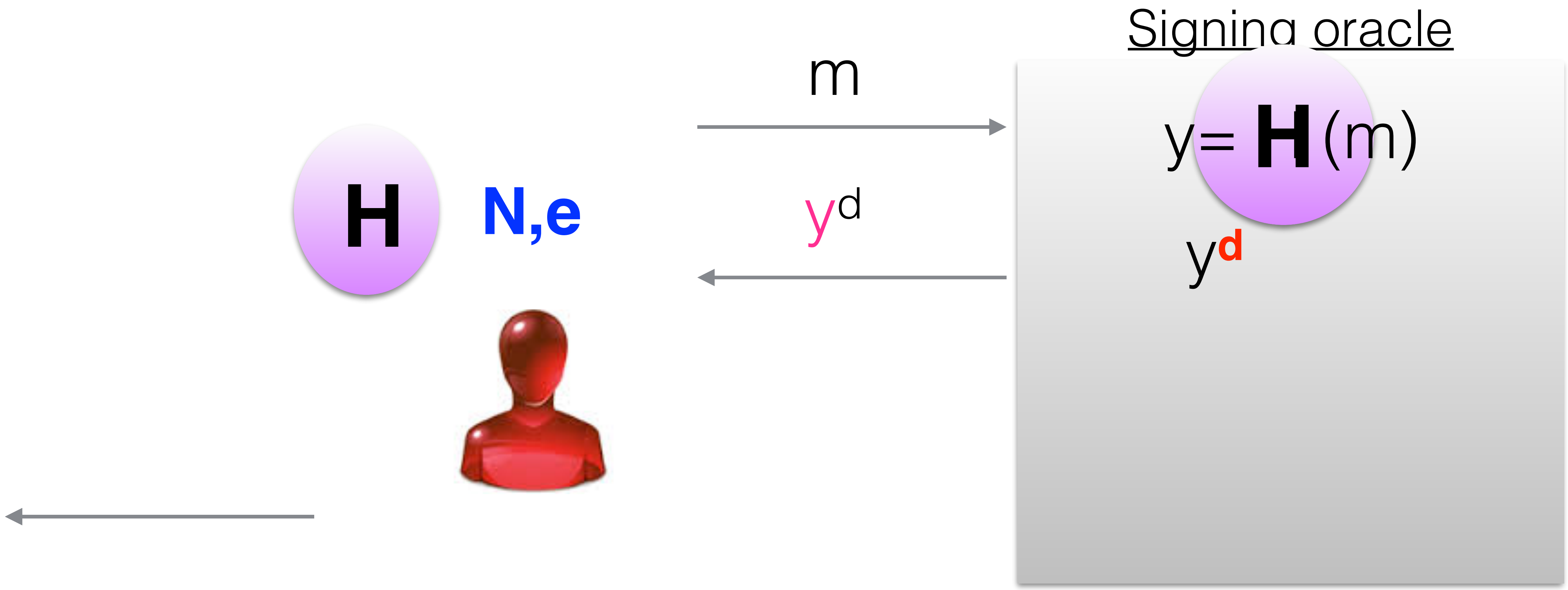
Intuition



How can the adversary find a forgery now?

- finds a collision => breaking H

Intuition



H

N,e

m

y^d

Signing oracle

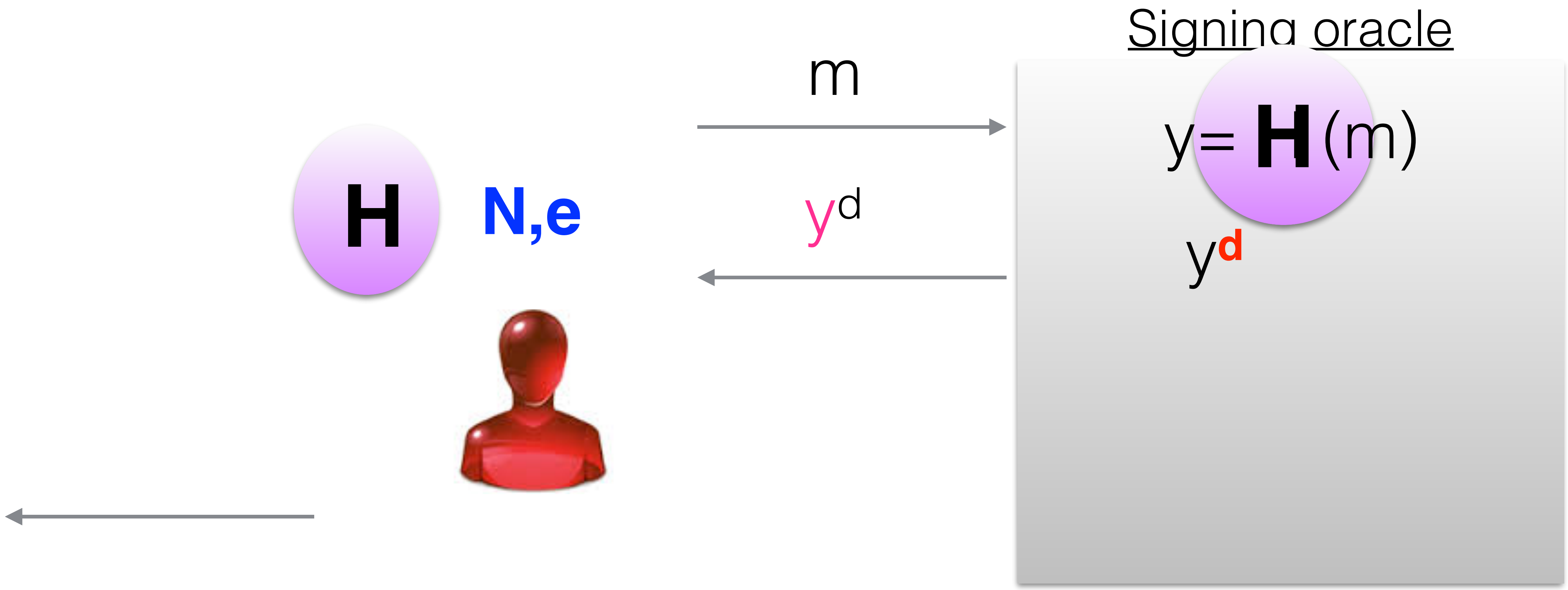
$y = H(m)$

y^d

How can the adversary find a forgery now?

- finds a collision => breaking H
- invert a random element y

Intuition



How can the adversary find a forgery now?

- finds a collision => breaking H
- invert a random element y => breaking RSA assumption

Discussion

Signature is not the inverse of public key encryption!

The public key PK must be transmitted reliably.
But this is why we need signature in the first place!

Signatures Scheme based on **Number Theoretic Assumptions**.

- Schnorr signature’s scheme
- ECDSA: Based on Discrete Log on Elliptic Curves

bitcoin

Signatures Scheme based on **Number Theoretic Assumptions.**

Not Post-Quantum Secure

- Schnorr signature’s scheme
- ECDSA: Based on Discrete Log on Elliptic Curves

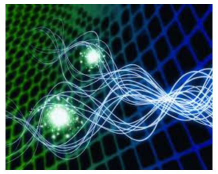
bitcoin

Digital Signature

- Definition   Unforgeability

- Construction

  RSA + Hash

  One-time Signature from OWF 

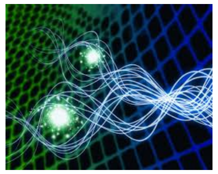
Digital Signature

- Definition   Unforgeability

- Construction

  RSA + Hash

  One-time Signature from OWF



Digital Signature

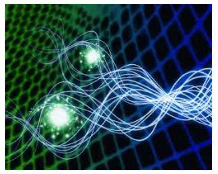
- Definition   Unforgeability

- Construction

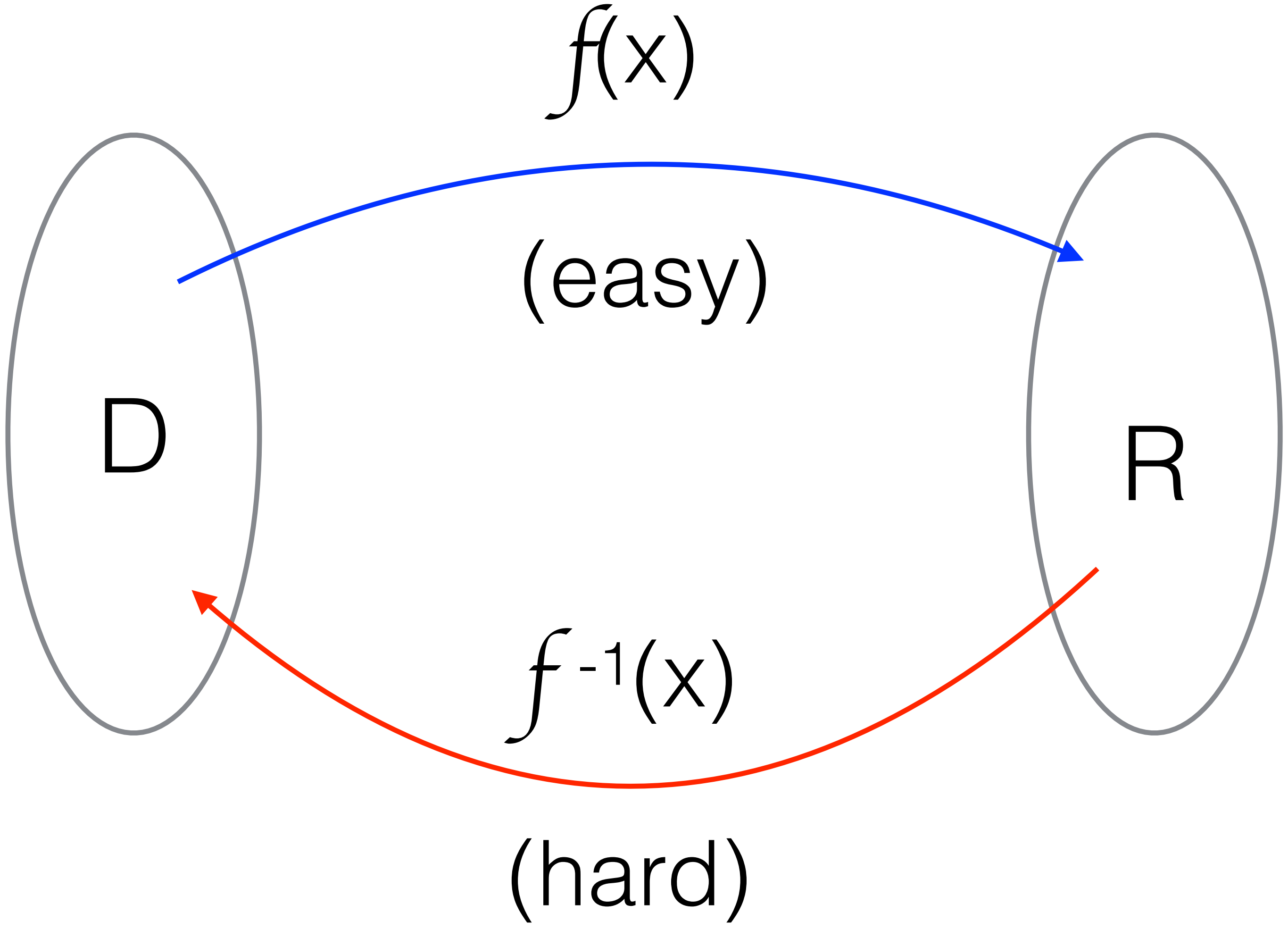
  RSA + Hash

  One-time Signature from OWF

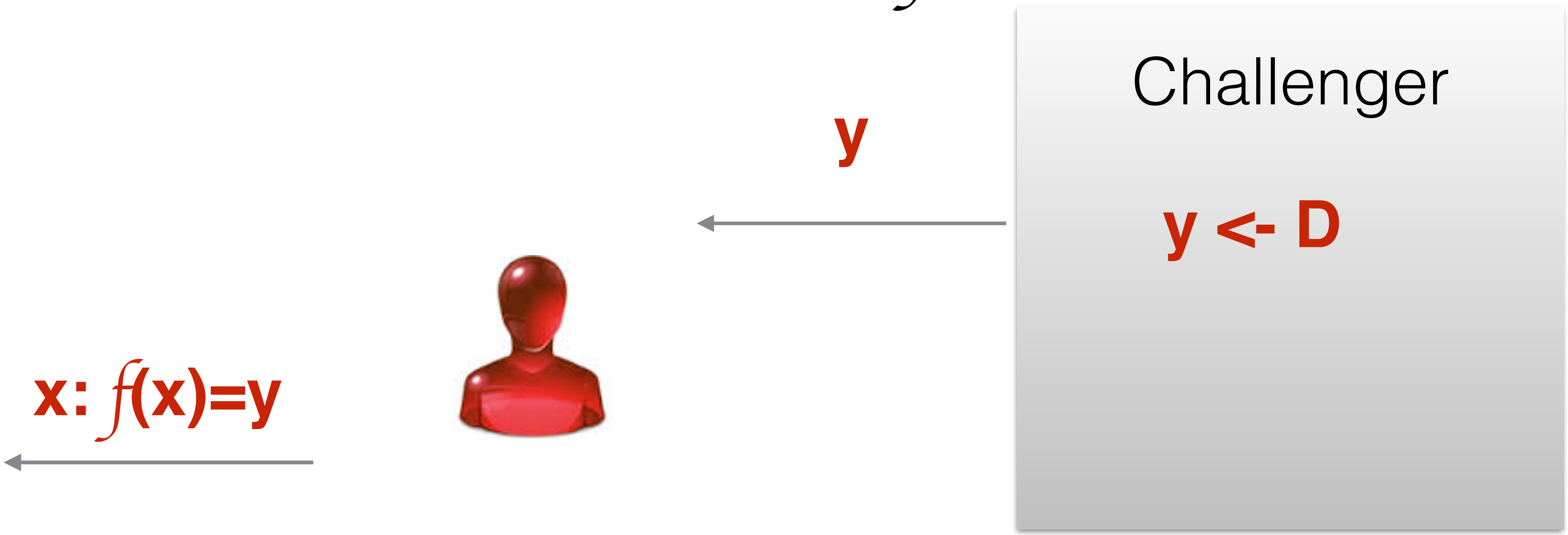
  Post-Quantum Secure



One-way Function f



One-Wayness f



Challenger

$y \leftarrow D$

y

x: $f(x)=y$

f is one-way if for a randomly selected y in Domain D, it is hard to find the pre-image x

Pr[A(y) -> x] is negligible

Lamport **One-time** Signature from OWF

Chapter 12.6

Pag. 462 Textbook

Lamport's scheme (pictorially)

e.g. message length 5 bits

KeyGen(5, f)

Sign(m, SK)

Lamport's scheme (pictorially)

e.g. message length 5 bits

KeyGen(5, f)

SK

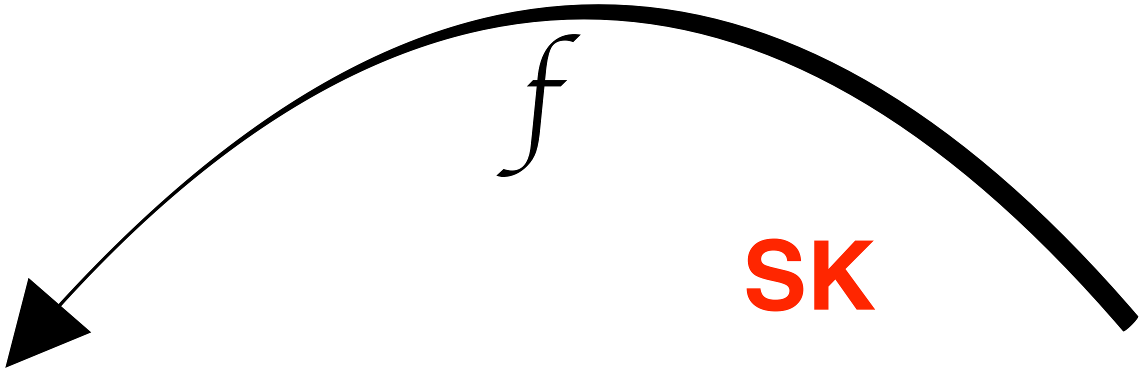
0	x^0_1	x^0_2	x^0_3	x^0_4	x^0_5
1	x^1_1	x^1_2	x^1_3	x^1_4	x^1_5

Sign(m, SK)

Lamport's scheme (pictorially)

e.g. message length 5 bits

KeyGen(5, f)



SK

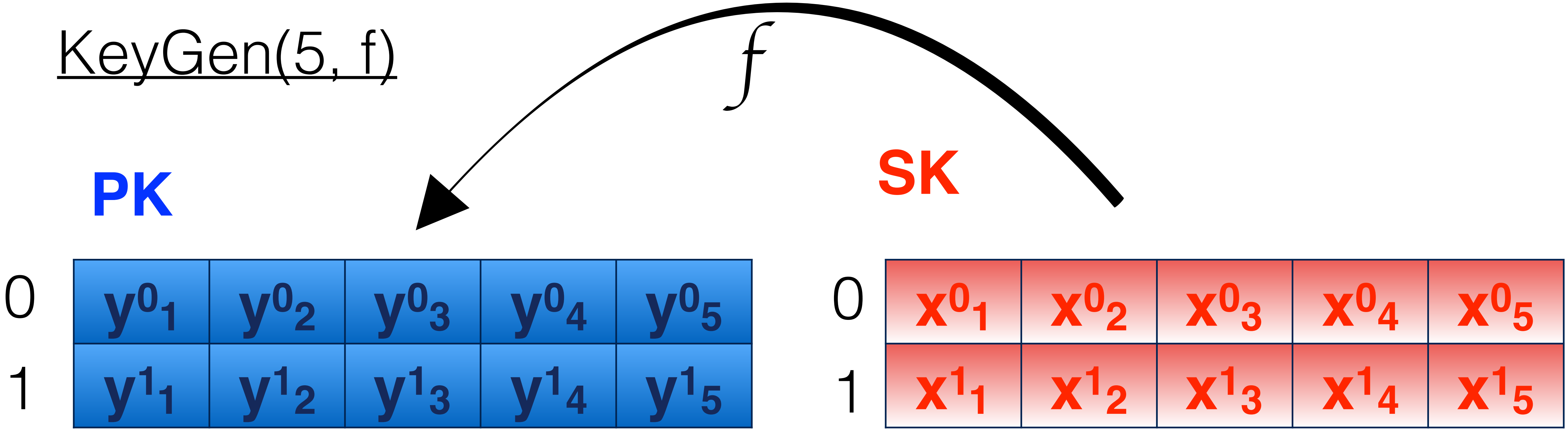
0	x^0_1	x^0_2	x^0_3	x^0_4	x^0_5
1	x^1_1	x^1_2	x^1_3	x^1_4	x^1_5

Sign(m, SK)

Lamport's scheme (pictorially)

e.g. message length 5 bits

KeyGen(5, f)



f

PK

0	y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
1	y^1_1	y^1_2	y^1_3	y^1_4	y^1_5

SK

0	x^0_1	x^0_2	x^0_3	x^0_4	x^0_5
1	x^1_1	x^1_2	x^1_3	x^1_4	x^1_5

Sign(m, SK)

Lamport's scheme (pictorially)

e.g. message length 5 bits

KeyGen(5, f)

PK

0	y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
1	y^1_1	y^1_2	y^1_3	y^1_4	y^1_5

Sign(m, SK)

m: 01011

Signature

Lamport's scheme (pictorially)

e.g. message length 5 bits

KeyGen(5, f)

PK

0	y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
1	y^1_1	y^1_2	y^1_3	y^1_4	y^1_5

Sign(m, SK)

m: 01011

Signature

x^0_1				

Lamport’s scheme (pictorially)

e.g. message length 5 bits

KeyGen(5, f)

PK

0	y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
1	y^1_1	y^1_2	y^1_3	y^1_4	y^1_5

Sign(m, SK)

m: 01011

Signature

x^0_1				
	x^1_2			

Lamport's scheme (pictorially)

e.g. message length 5 bits

KeyGen(5, f)

PK

0	y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
1	y^1_1	y^1_2	y^1_3	y^1_4	y^1_5

Sign(m, SK)

m: 01011

Signature

x^0_1		x^0_3		
	x^1_2			

Lamport's scheme (pictorially)

e.g. message length 5 bits

KeyGen(5, f)

PK

0	y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
1	y^1_1	y^1_2	y^1_3	y^1_4	y^1_5

Sign(m, SK)

m: 01011

Signature

x^0_1		x^0_3		
	x^1_2		x^1_4	

Lamport's scheme (pictorially)

e.g. message length 5 bits

KeyGen(5, f)

PK

0	y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
1	y^1_1	y^1_2	y^1_3	y^1_4	y^1_5

Sign(m, SK)

m: 01011

Signature

x^0_1		x^0_3		
	x^1_2		x^1_4	x^1_5

Theorem.

If F is a one-way function.

then the signature scheme is **one-time** secure

Proof.

(on the board)
Pag. 463 textbook

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5

PK ←

Signing oracle

Sign(**SK**,)

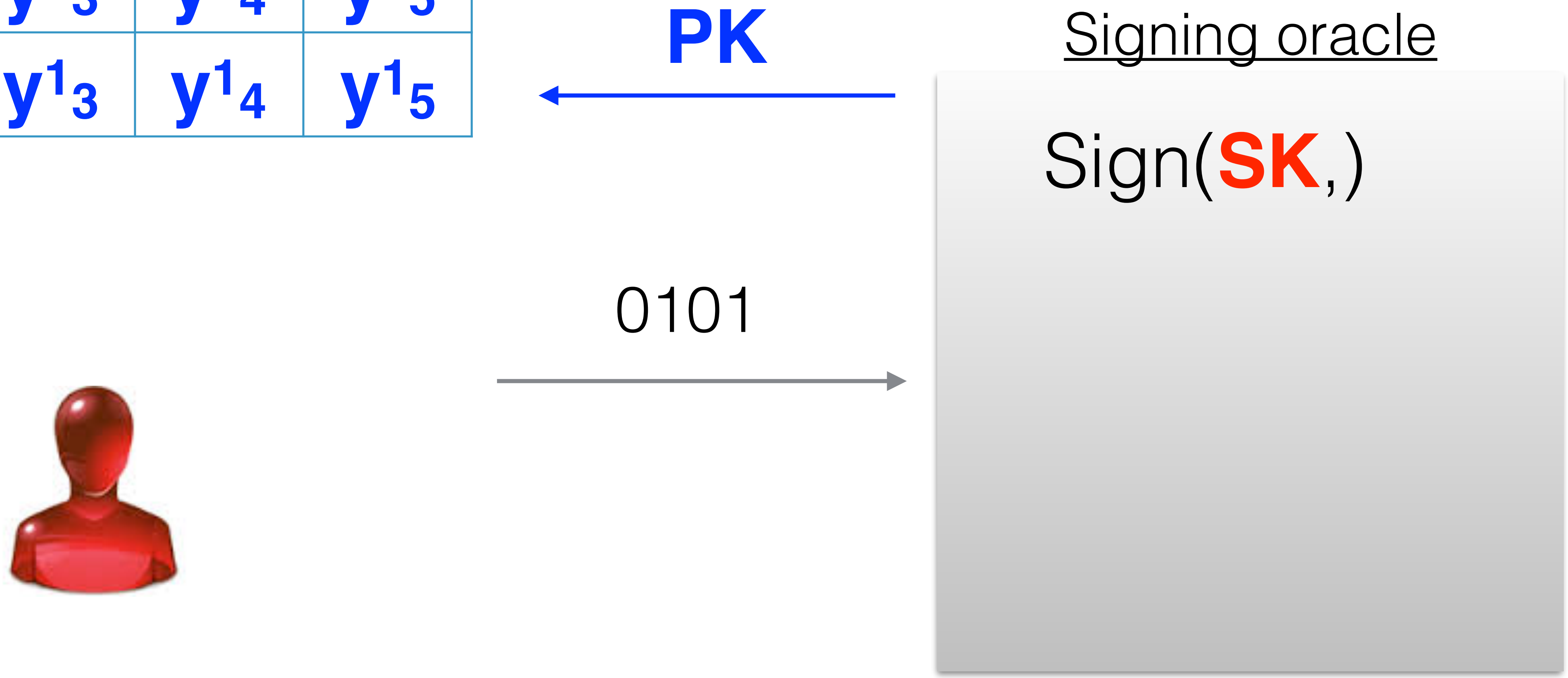
SK

0					
1					

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5



PK

Signing oracle

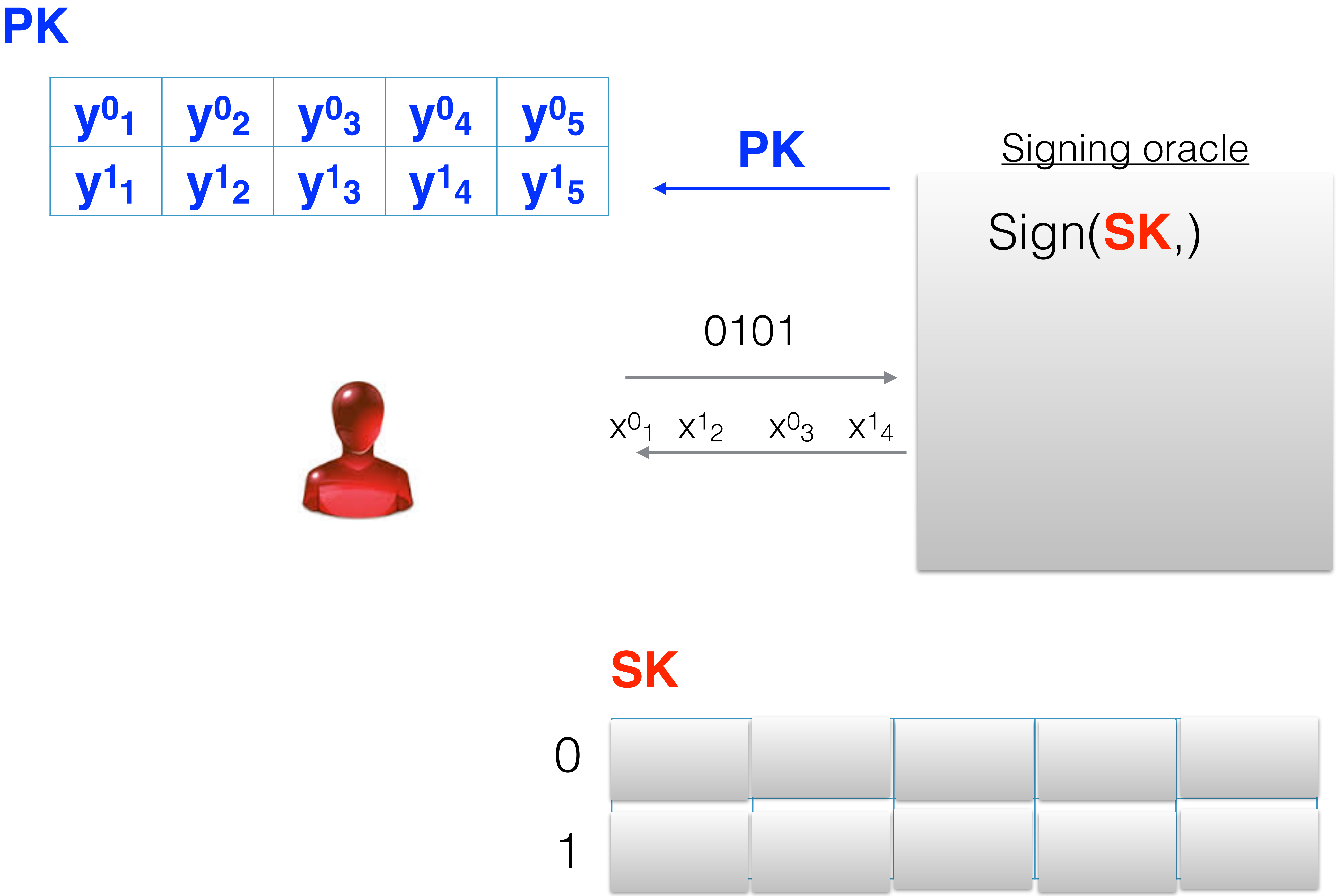
Sign(**SK**,)

0101

SK

0					
1					

Intuition.



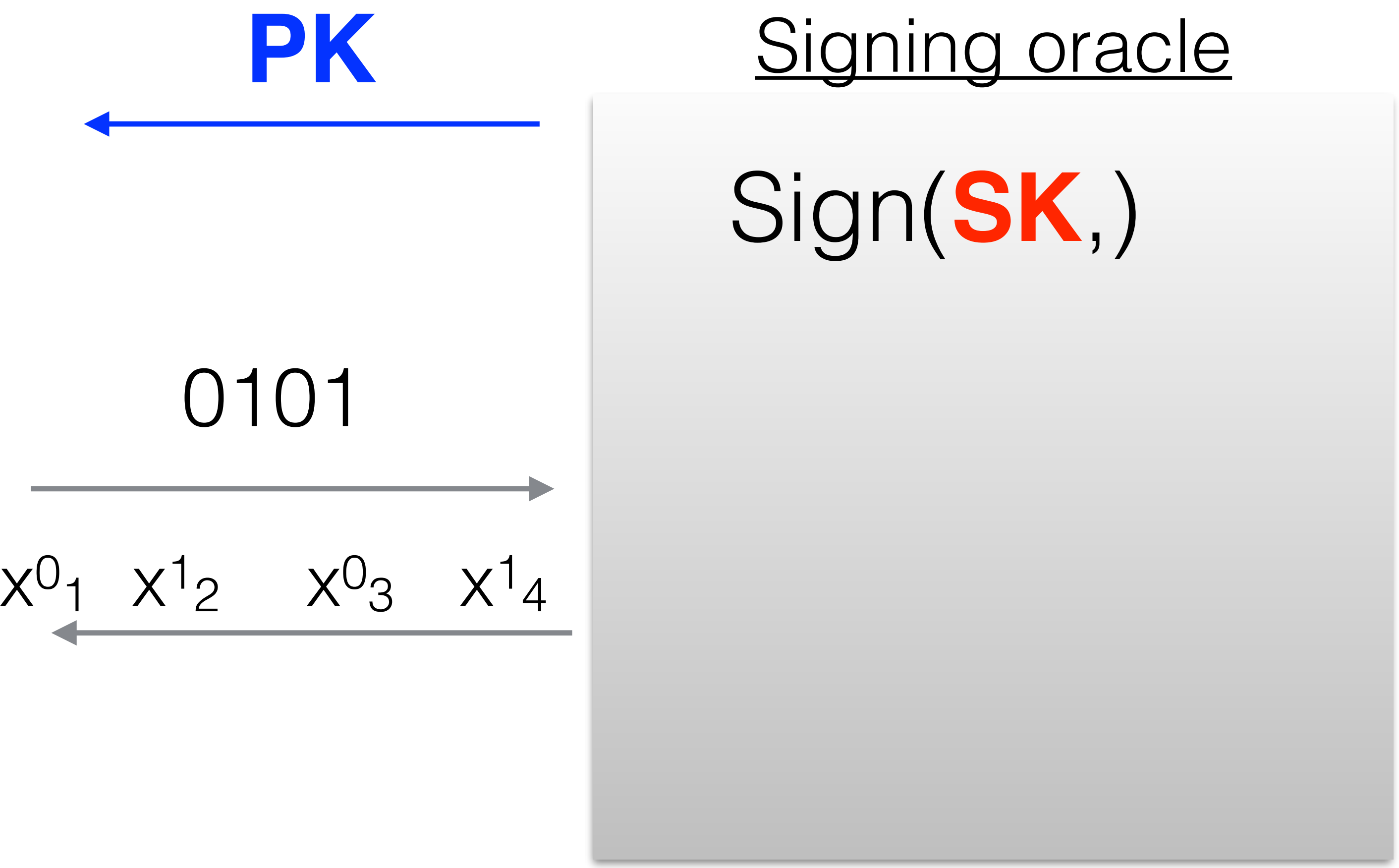
PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5



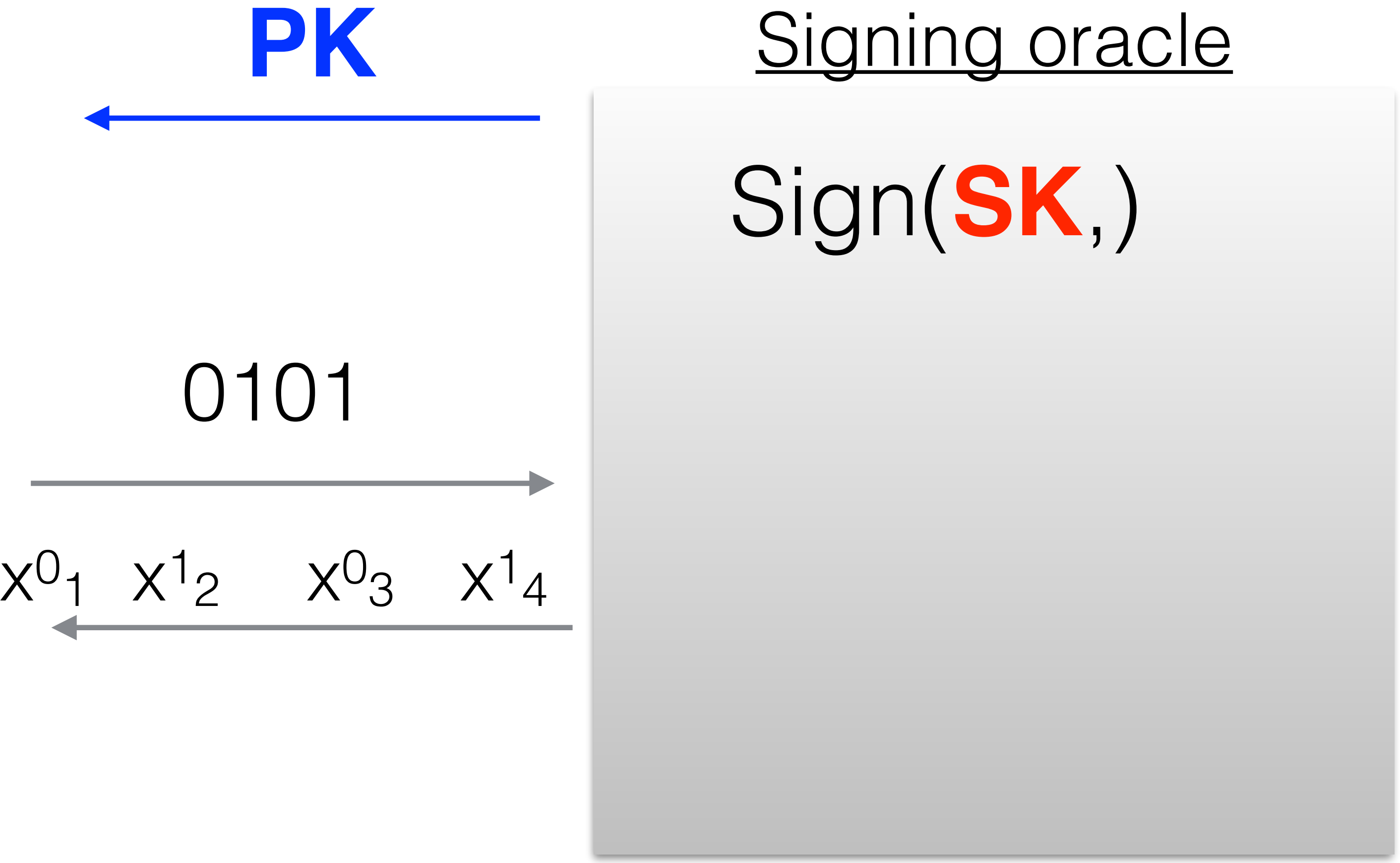
SK

0	x^0_1				
1					

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5



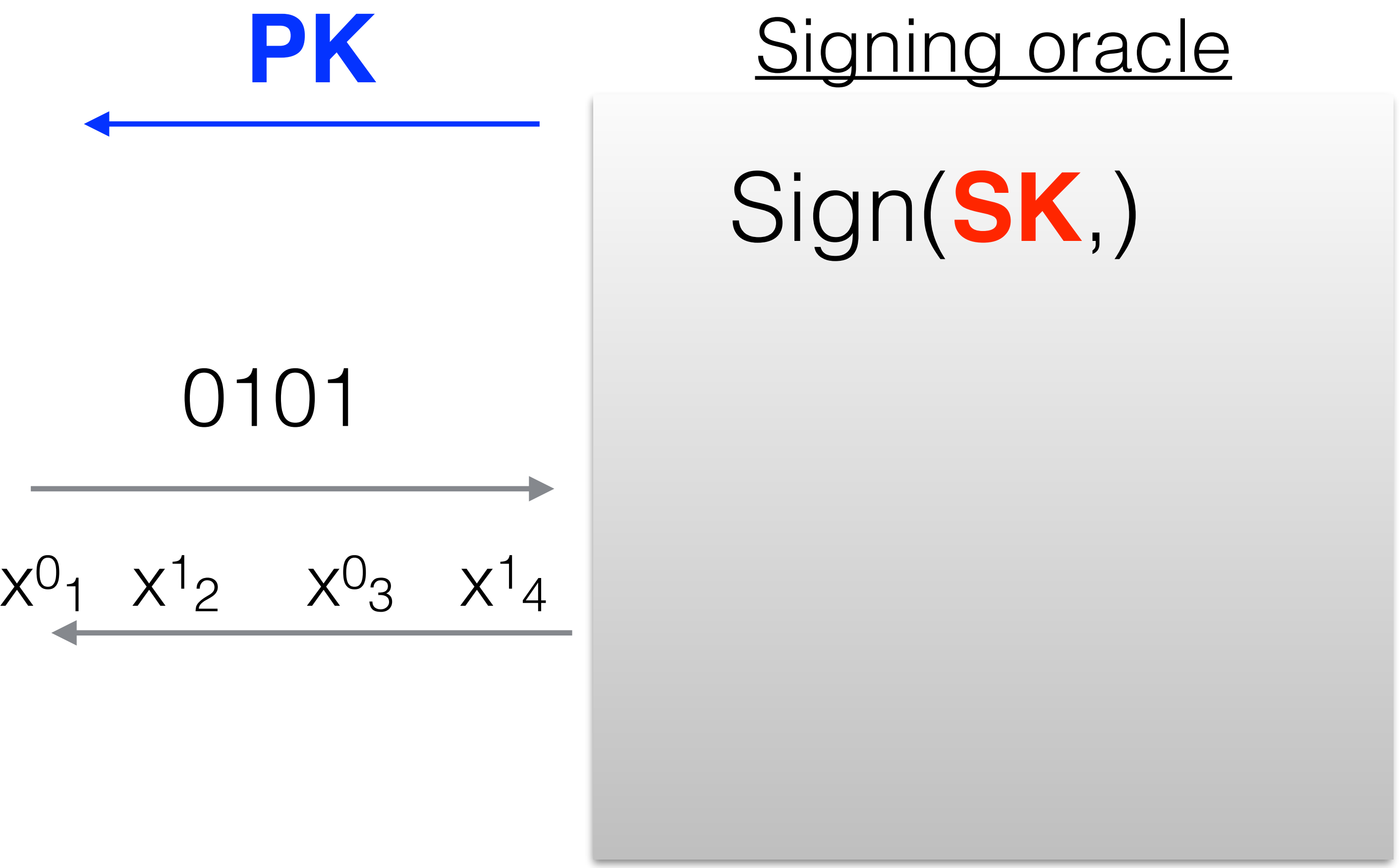
SK

0	x^0_1				
1		x^1_2			

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5



PK

Signing oracle

Sign(**SK**,)

0101

x^0_1 x^1_2 x^0_3 x^1_4

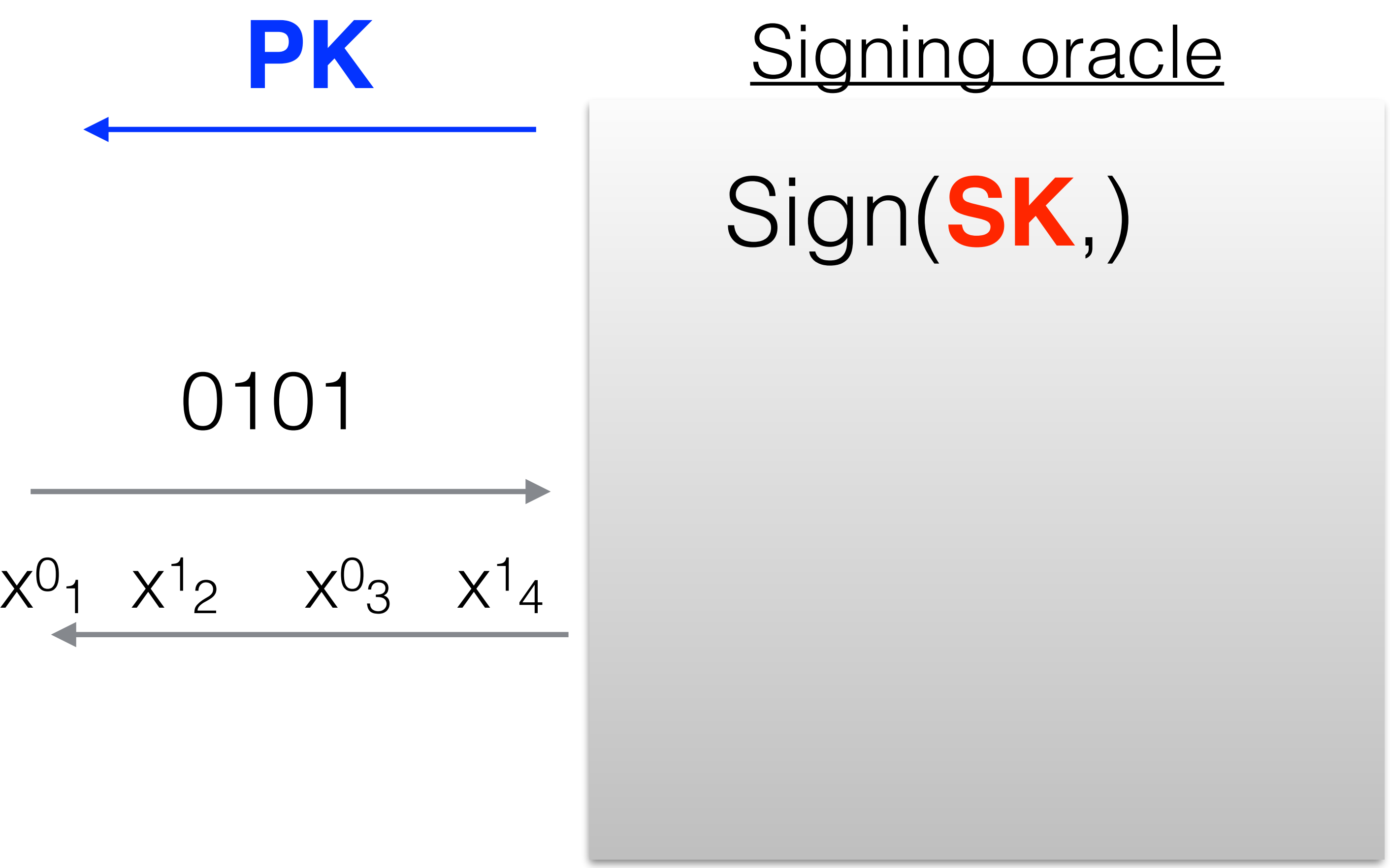
SK

0	x^0_1		x^0_3		
1		x^1_2			

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5



PK

Signing oracle

Sign(SK,)

0101

x^0_1 x^1_2 x^0_3 x^1_4

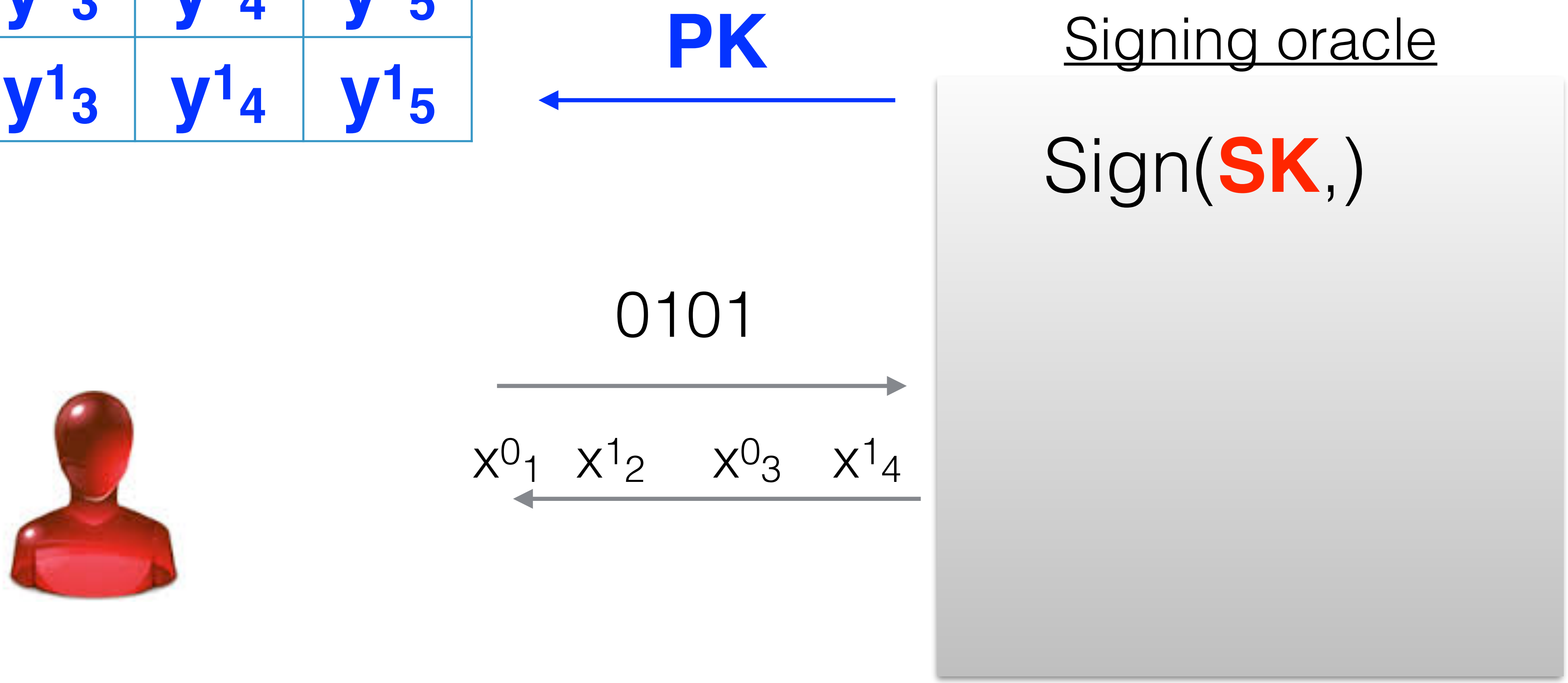
SK

0	x^0_1		x^0_3		
1		x^1_2		x^1_4	

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5



PK

Signing oracle

Sign(**SK**,)

0101

x^0_1 x^1_2 x^0_3 x^1_4

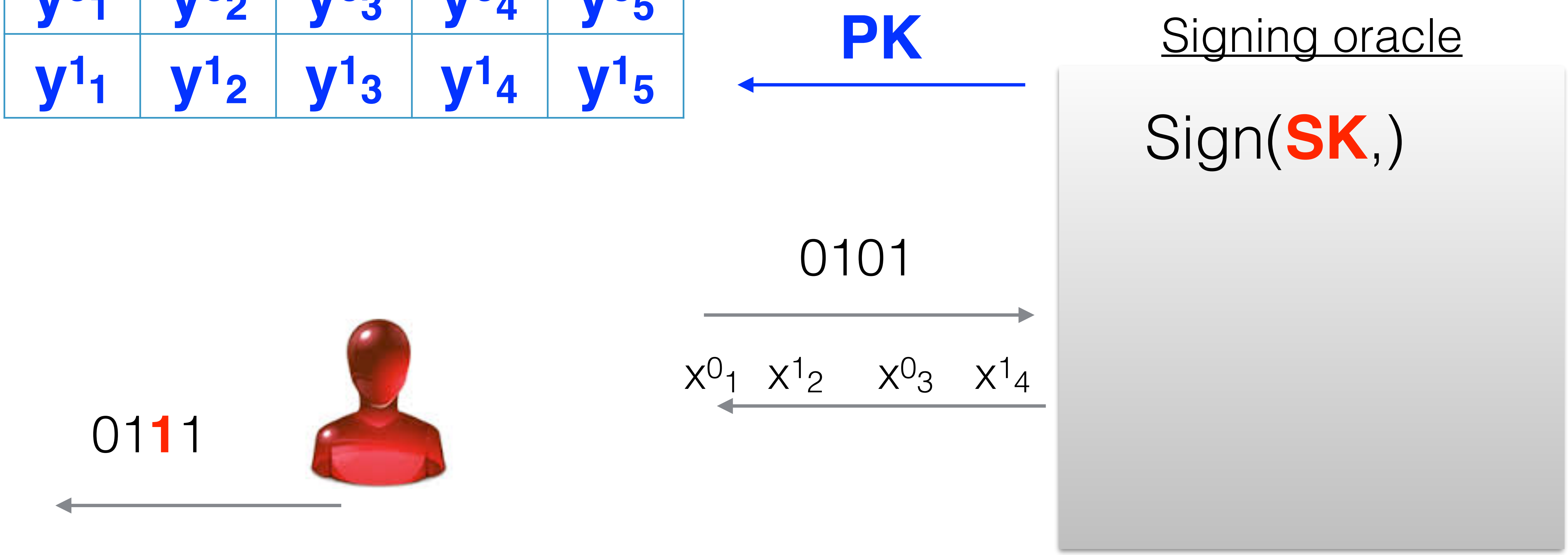
SK

0	x^0_1		x^0_3		
1		x^1_2		x^1_4	x^1_5

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5



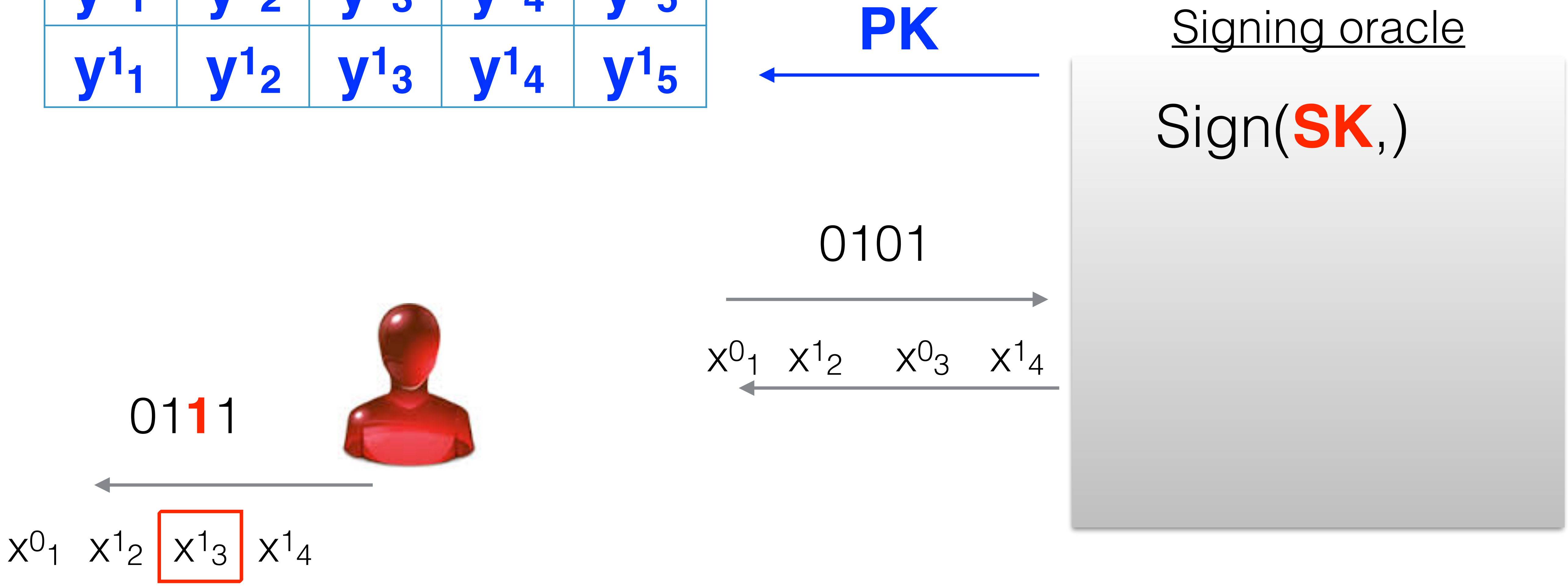
SK

0	x^0_1		x^0_3		
1		x^1_2		x^1_4	x^1_5

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5



PK

Signing oracle

Sign(SK,)

0101

x^0_1 x^1_2 x^0_3 x^1_4

0111

x^0_1 x^1_2 $\boxed{x^1_3}$ x^1_4

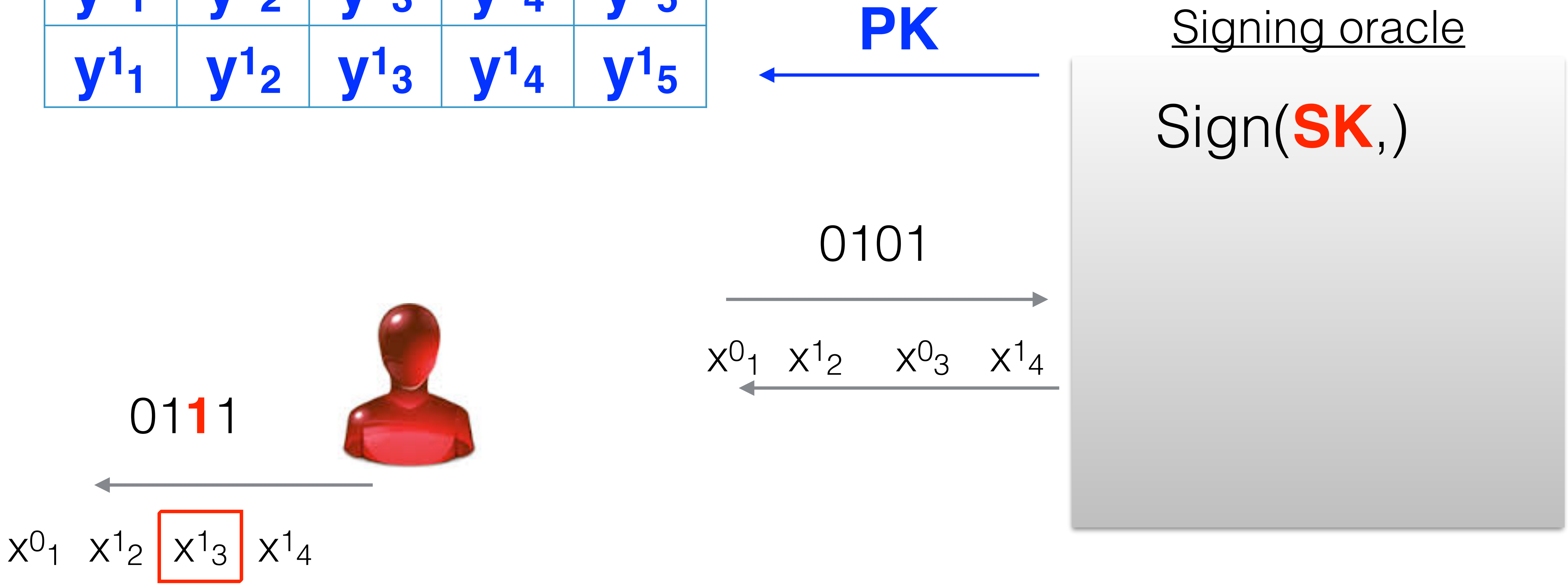
SK

0	x^0_1		x^0_3		
1		x^1_2		x^1_4	x^1_5

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	y^1_3	y^1_4	y^1_5



PK

Signing oracle

Sign(**SK**,)

0101

x^0_1 x^1_2 x^0_3 x^1_4

01**1**1

x^0_1 x^1_2 $\boxed{x^1_3}$ x^1_4

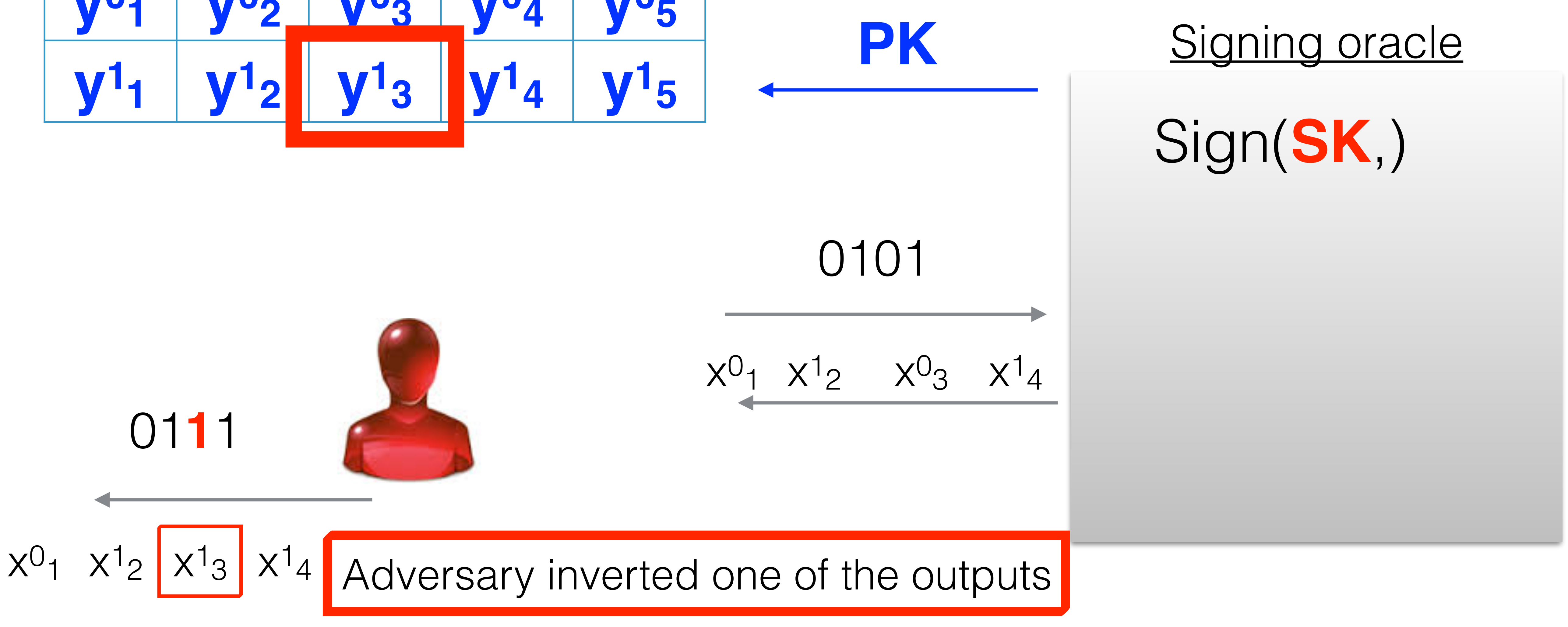
SK

0	x^0_1		x^0_3		
1		x^1_2	x^1_3	x^1_4	x^1_5

Intuition.

PK

y^0_1	y^0_2	y^0_3	y^0_4	y^0_5
y^1_1	y^1_2	$\boxed{y^1_3}$	y^1_4	y^1_5



PK

Signing oracle

Sign(SK,)

0101

x^0_1 x^1_2 x^0_3 x^1_4

0111

x^0_1 x^1_2 $\boxed{x^1_3}$ x^1_4

Adversary inverted one of the outputs

SK

0	x^0_1		x^0_3		
1		x^1_2	$\boxed{x^1_3}$	x^1_4	x^1_5

one-time => many times?

one-time => many times?

- Tree-based Signatures
- Chain-based Signature

[Candidate] Quantum Secure Signature Schemes

NIST Competition

Based on Lattices

Winternitz Signatures (improvement of Lamport signatures)

Integrity and Authentication

Message Authentication Code

Private key Setting

Property: Unforgeability

Constructions:

- □ MAC from PRF
- □ CBC-MAC

Digital Signature

Public key Setting

Property: Unforgeability

Constructions:

- □ RSA-based
- □ (general) One-way Function

Hash Functions

Property: Collision-Resistance

NO secret key!!

- □ Merkle-Damgård Transform
- □ Hash-function Block-ciphers
- □ Hash-function from Discrete Log Assumption.