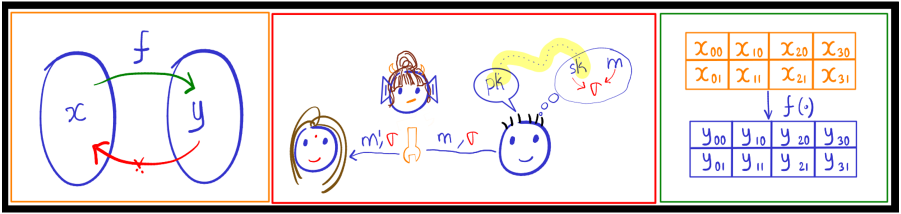


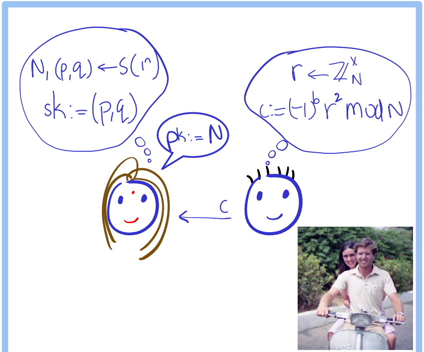
CS409m: Introduction to Cryptography

Lecture 15 (08/Oct/25)

Instructor: Chethan Kamath

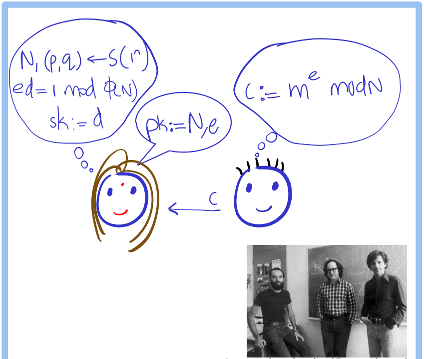
- Tasks: Public-key encryption (PKE)
- Threat model: IND-CPA

Goldwasser-Micali PKE



©Oded Goldreich

RSA PKE

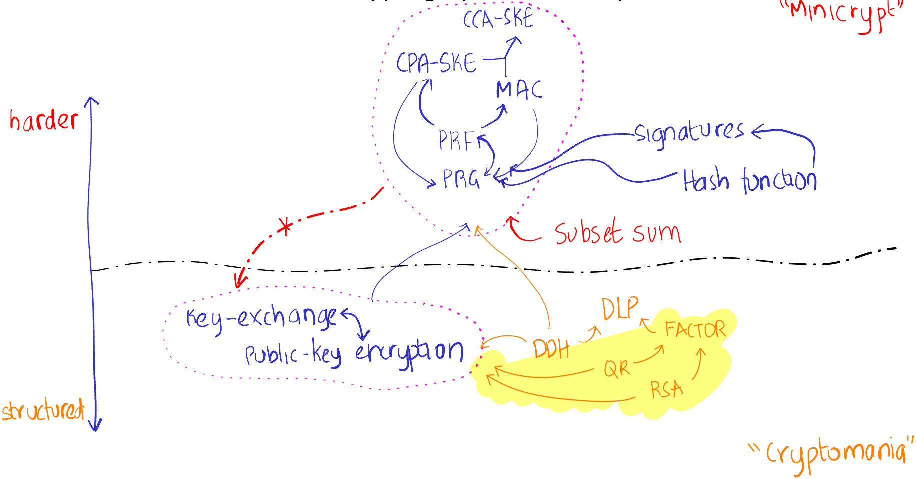


©cs.miami.edu (Rosenberg)

Algebraic setting: multiplication modulo *semiprime* (RSA group)

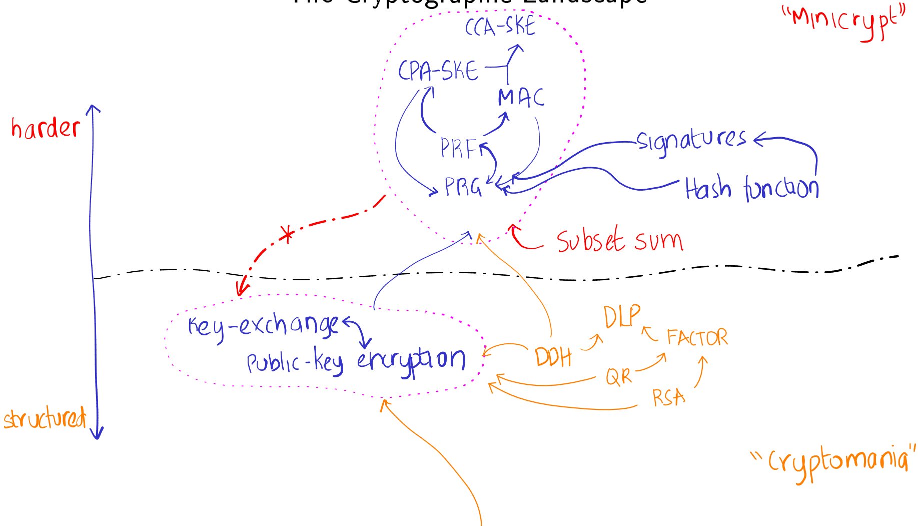
Recall from Last Lecture...

The Cryptographic Landscape



Hardness assumptions: integer factoring, QR and RSA

The Cryptographic Landscape



Elliptic curves, lattices, isogenies, class groups...

Plan for Today's Lecture...

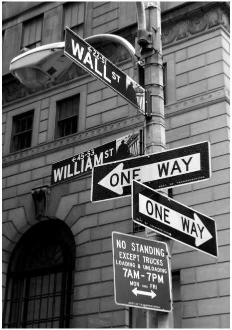
- Task: integrity and authentication in the *public-key* setting
- Threat model: EU-CMA

Plan for Today's Lecture...

- Task: integrity and authentication in the *public-key* setting
- Threat model: EU-CMA

NEW Digital Signature

One-Way Function NEW



Proof technique: plug and pray

Plan for Today's Lecture...

- Task: integrity and authentication in the *public-key* setting
- Threat model: EU-CMA

NEW Digital Signature

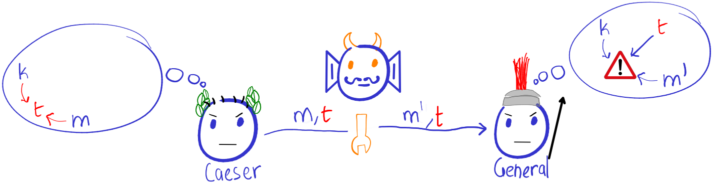
One-Way Function NEW

©Lubos Houska

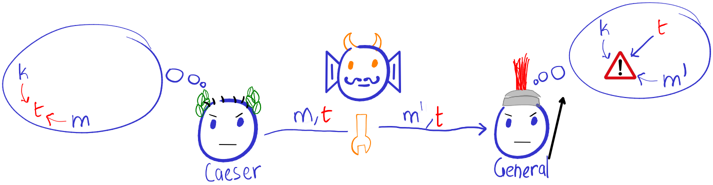
Proof technique: plug and pray

Digital (Analogues of Physical) Signatures . . .

- Recall: Message-Authentication Code (MAC)
- Used to detect tampering by active adversary



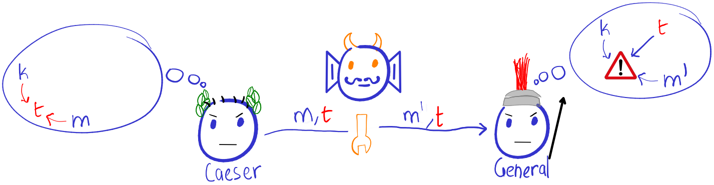
- Recall: Message-Authentication Code (MAC)
- Used to detect tampering by active adversary



- Digital signature: *public-key* counterpart of MAC

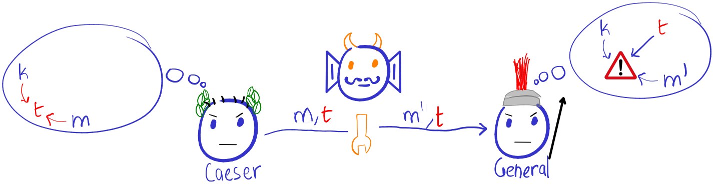
Digital (Analogues of Physical) Signatures...

- Recall: Message-Authentication Code (MAC)
- Used to detect tampering by active adversary



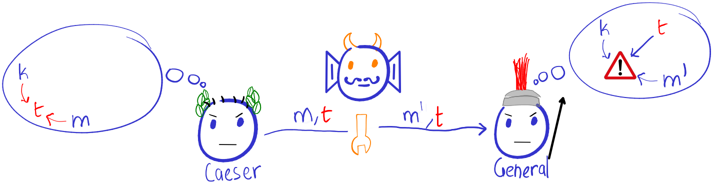
- Digital signature: *public-key* counterpart of MAC

- Recall: Message-Authentication Code (MAC)
- Used to detect tampering by active adversary

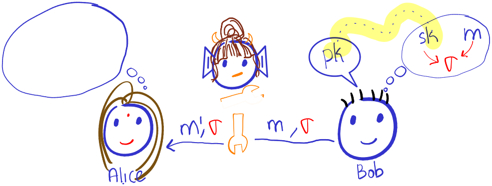


- Digital signature: *public-key* counterpart of MAC

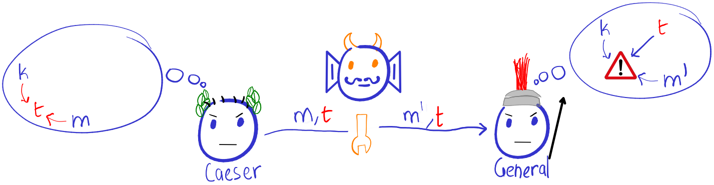
- Recall: Message-Authentication Code (MAC)
- Used to detect tampering by active adversary



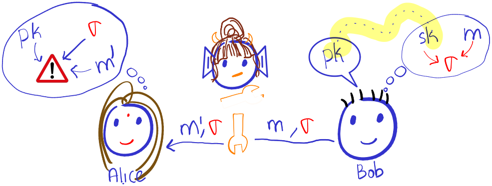
- Digital signature: *public-key* counterpart of MAC



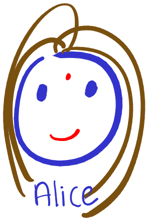
- Recall: Message-Authentication Code (MAC)
- Used to detect tampering by active adversary

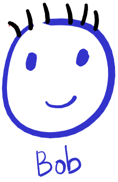


- Digital signature: *public-key* counterpart of MAC

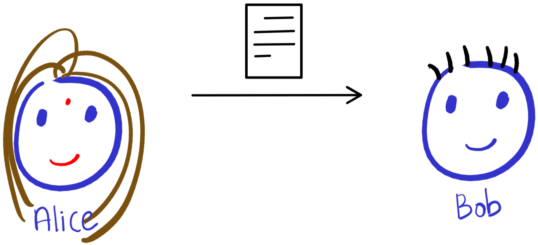


Digital (Analogues of Physical) Signatures...

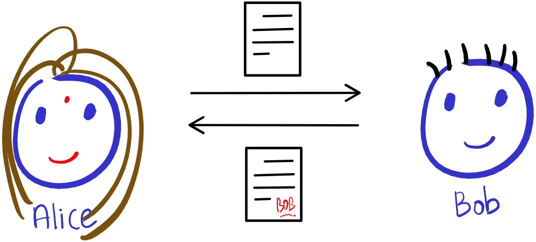




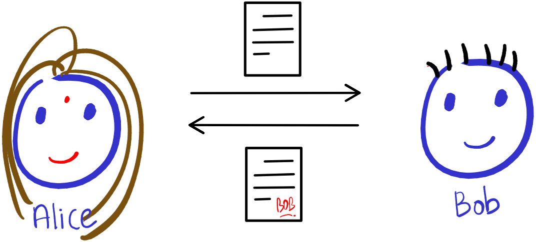
Digital (Analogues of Physical) Signatures...



Digital (Analogues of Physical) Signatures...

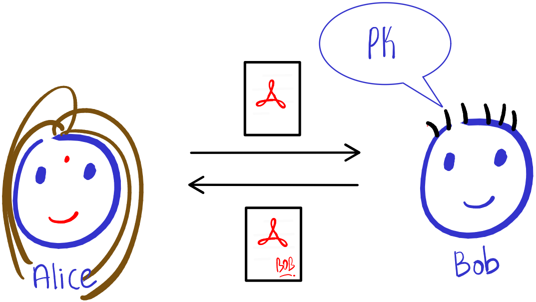


Digital (Analogues of Physical) Signatures...



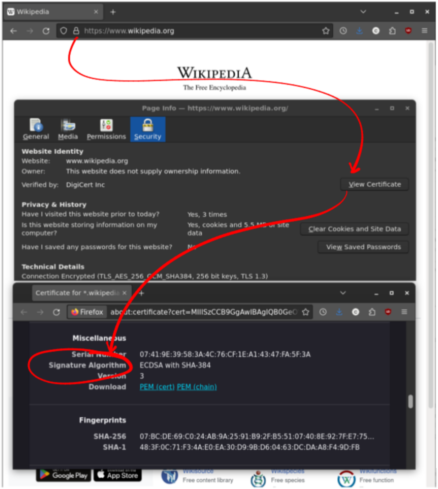
- Requirements:
  1. Publicly verifiable
  2. No one should be able to forge Bob's signature

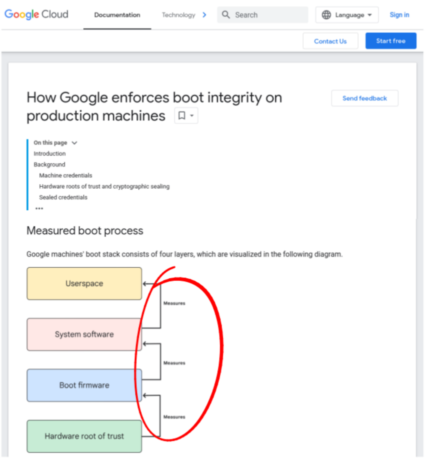
Digital (Analogues of Physical) Signatures...



- Requirements:
  1. Publicly verifiable
  2. No one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...





- Requirements:
  1. Publicly verifiable
  2. No one should be able to forge Bob's signature

Digital Signatures: Syntax

- *Public-key* analogue of message authentication codes (MAC)

Definition 1 (Digital signature (DS))

A DS Σ is a triple of efficient algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ with the following syntax:

Digital Signatures: Syntax

- *Public-key* analogue of message authentication codes (MAC)

Definition 1 (Digital signature (DS))

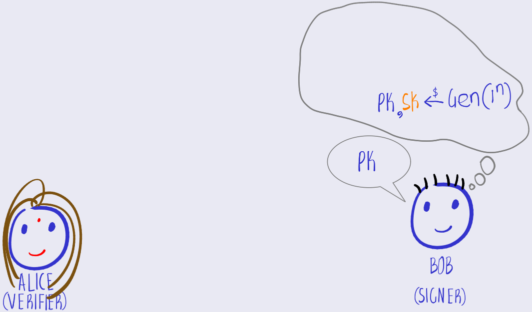
A DS Σ is a triple of efficient algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ with the following syntax:

Digital Signatures: Syntax

- *Public-key* analogue of message authentication codes (MAC)

Definition 1 (Digital signature (DS))

A DS Σ is a triple of efficient algorithms (Gen, Sign, Ver) with the following syntax:

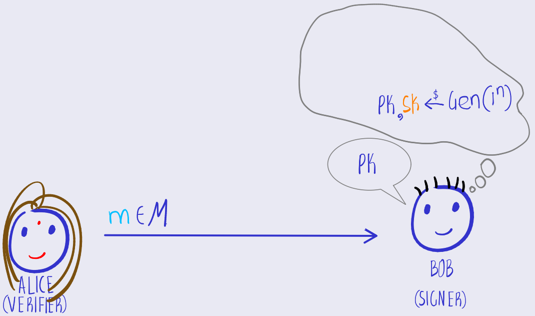


Digital Signatures: Syntax

- *Public-key* analogue of message authentication codes (MAC)

Definition 1 (Digital signature (DS))

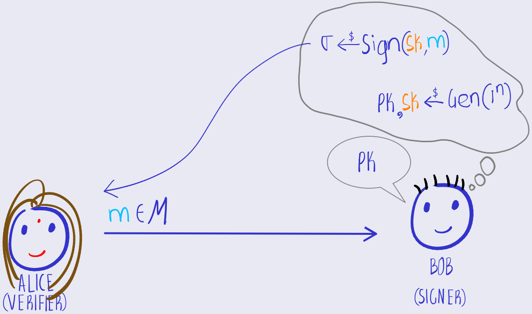
A DS Σ is a triple of efficient algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ with the following syntax:



- *Public-key* analogue of message authentication codes (MAC)

Definition 1 (Digital signature (DS))

A DS Σ is a triple of efficient algorithms (Gen, Sign, Ver) with the following syntax:

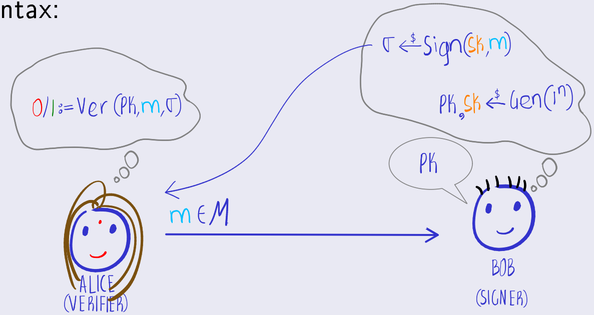


Digital Signatures: Syntax

- *Public-key* analogue of message authentication codes (MAC)

Definition 1 (Digital signature (DS))

A DS Σ is a triple of efficient algorithms (Gen, Sign, Ver) with the following syntax:

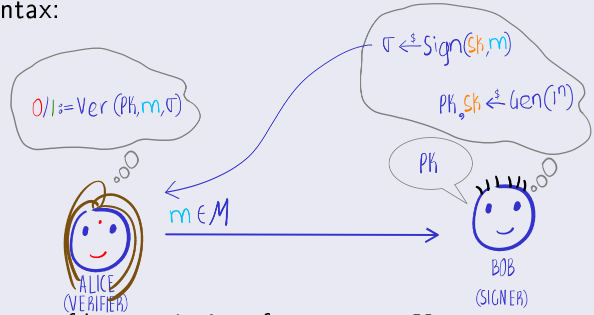


Digital Signatures: Syntax

- *Public-key* analogue of message authentication codes (MAC)

Definition 1 (Digital signature (DS))

A DS Σ is a triple of efficient algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ with the following syntax:



- Correctness of honest signing: for every $n \in \mathbb{N}$, message $m \in \mathcal{M}_n$,

$$\Pr_{(pk,sk)\leftarrow \mathsf{Gen}(1^n),\sigma\leftarrow \mathsf{Sign}(sk,m)}[\mathsf{Ver}(pk,\sigma,m)=1]=1$$

How to Define Security?...

- Intuitively, what are the security requirements?

How to Define Security?...

- Intuitively, what are the security requirements?
  - Tam must not be able to forge a valid *new* signature from previously-seen signatures...

How to Define Security?...

- Intuitively, what are the security requirements?
  - Tam must not be able to forge a valid *new* signature from previously-seen signatures...
    - ... on messages of its choice

How to Define Security?...

- Intuitively, what are the security requirements?
  - Tam must not be able to forge a valid *new* signature from previously-seen signatures...
    - ... on messages of its choice
  - Forged new signature can be on *any* message of Tam's choice

How to Define Security?...

- Intuitively, what are the security requirements?
  - Tam must not be able to forge a valid *new* signature from previously-seen signatures...
    - ... on messages of its choice
  - Forged new signature can be on *any* message of Tam's choice
- Existential Unforgeability Under Chosen-Message Attack ⚠

How to Define Security?...

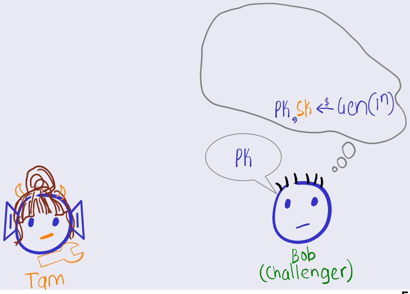
Intuitively, what are the security requirements?

- Tam must not be able to forge a valid *new* signature from previously-seen signatures...
  - ... on messages of its choice
- Forged new signature can be on *any* message of Tam's choice

- Existential Unforgeability Under Chosen-Message Attack ⚠

Definition 2 (EU-CMA)

A DS $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is q-EU-CMA secure if no PPT adversary Tam that makes at most q queries can break Σ as follows with non-negligible probability.



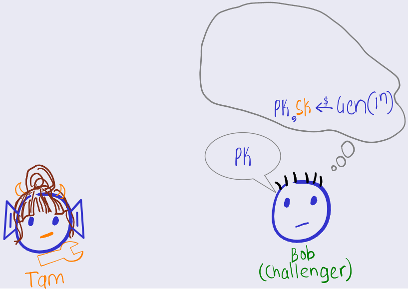
How to Define Security?...

- Intuitively, what are the security requirements?
  - Tam must not be able to forge a valid *new* signature from previously-seen signatures...
    - ... on messages of its choice
  - Forged new signature can be on *any* message of Tam's choice
- Existential Unforgeability Under Chosen-Message Attack ⚠

Definition 2 (EU-CMA)

A DS $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is q-EU-CMA secure if no PPT adversary Tam that makes at most q queries can break Σ as follows with non-negligible probability.

- Tam given PK



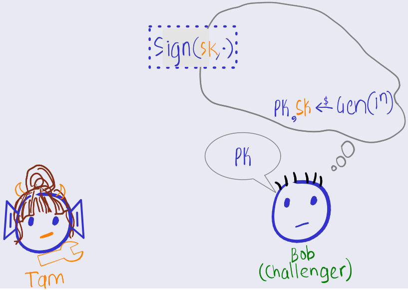
How to Define Security?

- Intuitively, what are the security requirements?
  - Tam must not be able to forge a valid *new* signature from previously-seen signatures…
    - … on messages of its choice
  - Forged new signature can be on *any* message of Tam's choice
- Existential Unforgeability Under Chosen-Message Attack ⚠

Definition 2 (EU-CMA)

A DS $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is q-EU-CMA secure if no PPT adversary Tam that makes at most q queries can break Σ as follows with non-negligible probability.

- Tam given pk

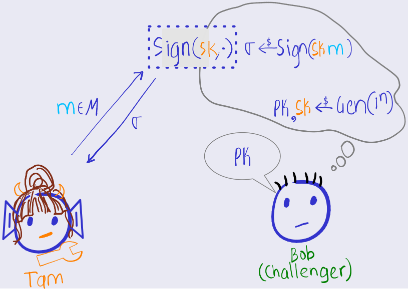


- Intuitively, what are the security requirements?
  - Tam must not be able to forge a valid *new* signature from previously-seen signatures...
    - ... on messages of its choice
  - Forged new signature can be on *any* message of Tam's choice
- Existential Unforgeability Under Chosen-Message Attack ⚠

Definition 2 (EU-CMA)

A DS $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is q-EU-CMA secure if no PPT adversary Tam that makes at most q queries can break Σ as follows with non-negligible probability.

- Tam given PK

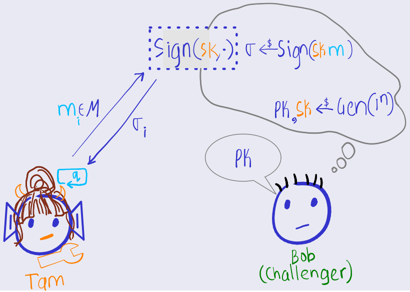


- Intuitively, what are the security requirements?
  - Tam must not be able to forge a valid *new* signature from previously-seen signatures...
    - ... on messages of its choice
  - Forged new signature can be on *any* message of Tam's choice
- Existential Unforgeability Under Chosen-Message Attack

Definition 2 (EU-CMA)

A DS $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is q-EU-CMA secure if no PPT adversary Tam that makes at most q queries can break Σ as follows with non-negligible probability.

- Tam given pk
- Tam makes q queries to Sign(sk,·) oracle

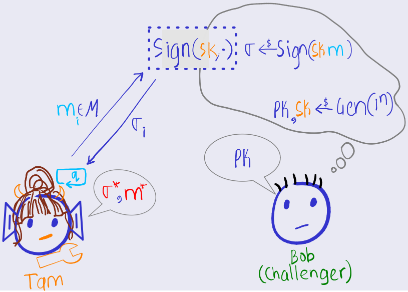


- Intuitively, what are the security requirements?
  - Tam must not be able to forge a valid *new* signature from previously-seen signatures...
    - ... on messages of its choice
  - Forged new signature can be on *any* message of Tam's choice
- Existential Unforgeability Under Chosen-Message Attack

Definition 2 (EU-CMA)

A DS $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is q-EU-CMA secure if no PPT adversary Tam that makes at most q queries can break Σ as follows with non-negligible probability.

- Tam given pk
- Tam makes q queries to Sign(sk, ·) oracle
- In the end Tam outputs (σ^*, m^*) and breaks Σ if:
  - $\mathsf{Ver}(pk, m^*, \sigma^*) = 1$
  - $\forall i \in [q] : m^* \neq m_i$



❓ Σ' EU-CMA Secure or Not?

$\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}) \to \Sigma' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$

1. Truncate-then-sign: define Σ' as
   - $\mathsf{Sign}'(sk, m := m_1 \cdots m_{\ell-1} m_\ell) \leftarrow \mathsf{Sign}(sk, m_1 \cdots m_{\ell-1})$
   - $\mathsf{Ver}'(pk, \sigma, m) := \mathsf{Ver}(pk, \sigma, m_1 \cdots m_{\ell-1})$

? Σ' EU-CMA Secure or Not?

$\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}) \to \Sigma' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$

1. Truncate-then-sign: define Σ' as
   - $\mathsf{Sign}'(sk, m := m_1 \cdots m_{\ell-1} m_\ell) \leftarrow \mathsf{Sign}(sk, m_1 \cdots m_{\ell-1})$
   - $\mathsf{Ver}'(pk, \sigma, m) := \mathsf{Ver}(pk, \sigma, m_1 \cdots m_{\ell-1})$
2. Sign-then-truncate: define Σ' as
   - $\mathsf{Sign}'(sk, m) := \sigma_1 \cdots \sigma_{s-1}$, where $\sigma_1 \cdots \sigma_{s-1}\sigma_s \leftarrow \mathsf{Sign}(sk, m)$
   - $\mathsf{Ver}'(pk, \sigma', m)$: accept if
     - $\mathsf{Ver}(pk, \sigma' \| 0, m) = 1$ *or* $\mathsf{Ver}(pk, \sigma' \| 1, m) = 1$

Σ' EU-CMA Secure or Not?

$\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}) \to \Sigma' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$

1. Truncate-then-sign: define Σ' as
   - $\mathsf{Sign}'(sk, m := m_1 \cdots m_{\ell-1} m_\ell) \leftarrow \mathsf{Sign}(sk, m_1 \cdots m_{\ell-1})$
   - $\mathsf{Ver}'(pk, \sigma, m) := \mathsf{Ver}(pk, \sigma, m_1 \cdots m_{\ell-1})$
2. Sign-then-truncate: define Σ' as
   - $\mathsf{Sign}'(sk, m) := \sigma_1 \cdots \sigma_{s-1}$, where $\sigma_1 \cdots \sigma_{s-1}\sigma_s \leftarrow \mathsf{Sign}(sk, m)$
   - $\mathsf{Ver}'(pk, \sigma', m)$: accept if
     - $\mathsf{Ver}(pk, \sigma' \| 0, m) = 1$ *or* $\mathsf{Ver}(pk, \sigma' \| 1, m) = 1$
3. Sign-then-append: define Σ' as
   - $\mathsf{Sign}'(sk, m) := \sigma \| 0$, where $\sigma \leftarrow \mathsf{Sign}(sk, m)$
   - $\mathsf{Ver}'(pk, \sigma \| b, m) := \mathsf{Ver}(pk, \sigma, m)$

Σ' EU-CMA Secure or Not?

$\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}) \to \Sigma' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$

1. Truncate-then-sign: define Σ' as
   - $\mathsf{Sign}'(sk, m := m_1 \cdots m_{\ell-1} m_\ell) \leftarrow \mathsf{Sign}(sk, m_1 \cdots m_{\ell-1})$
   - $\mathsf{Ver}'(pk, \sigma, m) := \mathsf{Ver}(pk, \sigma, m_1 \cdots m_{\ell-1})$
2. Sign-then-truncate: define Σ' as
   - $\mathsf{Sign}'(sk, m) := \sigma_1 \cdots \sigma_{s-1}$, where $\sigma_1 \cdots \sigma_{s-1}\sigma_s \leftarrow \mathsf{Sign}(sk, m)$
   - $\mathsf{Ver}'(pk, \sigma', m)$: accept if
     - $\mathsf{Ver}(pk, \sigma' \| 0, m) = 1$ *or* $\mathsf{Ver}(pk, \sigma' \| 1, m) = 1$
3. Sign-then-append: define Σ' as
   - $\mathsf{Sign}'(sk, m) := \sigma \| 0$, where $\sigma \leftarrow \mathsf{Sign}(sk, m)$
   - $\mathsf{Ver}'(pk, \sigma \| b, m) := \mathsf{Ver}(pk, \sigma, m)$

Exercise 1

Prove by reduction that the Σ's in 1 and 3 are EU-CMA-secure.

Plan for Today's Lecture...

- Task: integrity and authentication in the *public-key* setting
- Threat model: EU-CMA

NEW Digital Signature


©Wikipedia

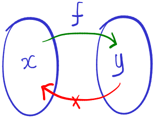
One-Way Function NEW

©Lubos Houska

Proof technique: plug and pray

Let's Define One-Way Functions...

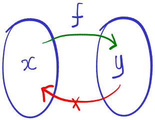
- Intuitively: "easy to compute" function f that is "hard to invert"



- Intuitively: "easy to compute" function f that is "hard to invert"

"Solution"

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



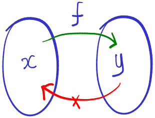
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

"Puzzle"

- Intuitively: "easy to compute" function f that is "hard to invert"

"Solution"

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

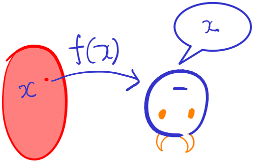
"Puzzle"

- What does "hard to invert" entail? Attempt 1 :

$\forall$ PPT inverter Inv, $\forall x$,

$$\Pr[\mathrm{Inv}(f(x)) = x]$$

is negligible.



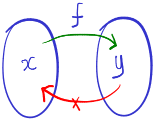
- Problem:

Let's Define One-Way Functions...

Intuitively: "easy to compute" function f that is "hard to invert"

"Solution"

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

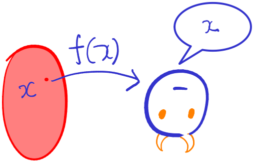
"Puzzle"

- What does "hard to invert" entail? Attempt 1:

$\forall$ PPT inverter Inv, $\forall x$,

$$\Pr[\mathrm{Inv}(f(x)) = x]$$

is negligible.



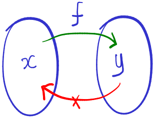
- Problem: Too much to ask (everywhere hardness)

Let's Define One-Way Functions...

Intuitively: "easy to compute" function f that is "hard to invert"

"Solution"

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

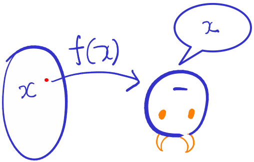
"Puzzle"

- What does "hard to invert" entail? Attempt 2 :

$\forall$ PPT inverter Inv, $\exists x$,

$$\Pr[\mathrm{Inv}(f(x)) = x]$$

is negligible.

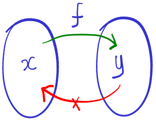


- Problem: Too much to ask (everywhere hardness)

- Intuitively: "easy to compute" function f that is "hard to invert"

"Solution"

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

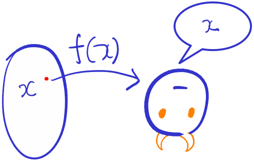
"Puzzle"

- What does "hard to invert" entail? Attempt 2 :

$\forall$ PPT inverter Inv, $\exists x$,

$$\Pr[\mathrm{Inv}(f(x)) = x]$$

is negligible.



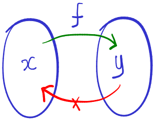
- Problem: This is not sufficient (worst-case hardness)

Let's Define One-Way Functions...

- Intuitively: "easy to compute" function f that is "hard to invert"

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

"Solution"



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

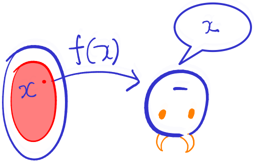
"Puzzle"

- What does "hard to invert" entail? Attempt 3 :

∀ PPT inverter Inv

$$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathrm{Inv}(f(x)) = x\right]$$

is negligible.



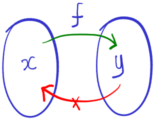
- Problem:

Let's Define One-Way Functions...

Intuitively: "easy to compute" function f that is "hard to invert"

"Solution"

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

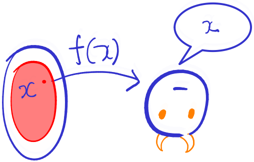
"Puzzle"

- What does "hard to invert" entail? Attempt 3 :

$\forall$ PPT inverter Inv

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathrm{Inv}(f(x)) = x]$$

is negligible.

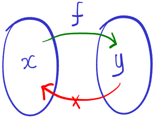


- Problem: What about $f(x) := 0^{|x|}$?

Intuitively: "easy to compute" function f that is "hard to invert"

"Solution"

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

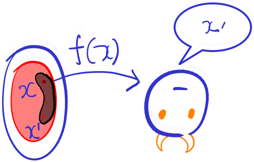
"Puzzle"

- What does "hard to invert" entail? Attempt 4 :

$\forall$ PPT inverter Inv

$$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathrm{Inv}(f(x)) \in f^{-1}(f(x))\right]$$

is negligible.



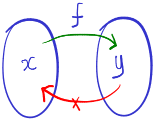
- Problem:

Intuitively: "easy to compute" function f that is "hard to invert"

"Solution"

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

©Cburnett/Wikipedia



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

"Puzzle"

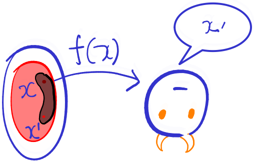
©Tim Stellmach/Wikipedia

- What does "hard to invert" entail? Attempt 4 :

$\forall$ PPT inverter Inv

$$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathrm{Inv}(f(x)) \in f^{-1}(f(x))\right]$$

is negligible.



- Problem: ?

Let's Define One-Way Functions...

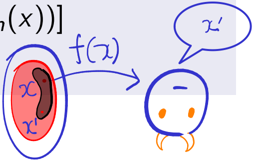
Intuitively: "easy to compute" function that is "hard to invert"

Definition 3 (One-way function (OWF))

A function family $f := \left\{f_n : \{0,1\}^n \to \{0,1\}^{m(n)}\right\}_{n\in\mathbb{N}}$ is one-way if

- there exists an efficient algorithm F such that $\forall x : \mathsf{F}(x) = f(x)$
- for all PPT *inverters* Inv, the following is negligible:

$$p(n) := \Pr_{x \leftarrow \{0,1\}^n}[\mathsf{Inv}(f_n(x)) \in f_n^{-1}(f_n(x))]$$



Let's Define One-Way Functions...

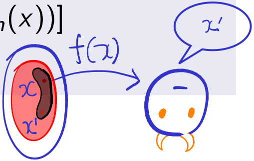
Intuitively: "easy to compute" function that is "hard to invert"

Definition 3 (One-way function (OWF))

A function family $f := \left\{f_n : \{0,1\}^n \to \{0,1\}^{m(n)}\right\}_{n\in\mathbb{N}}$ is one-way if

- there exists an efficient algorithm F such that $\forall x : \mathsf{F}(x) = f(x)$
- for all PPT *inverters* Inv, the following is negligible:

$$p(n) := \Pr_{x\leftarrow\{0,1\}^n}[\mathsf{Inv}(f_n(x)) \in f_n^{-1}(f_n(x))]$$



- Length-preserving OWF: $m(n) = n$
- One-way permutation: f is length-preserving and *bijective*

Let's Define One-Way Functions...

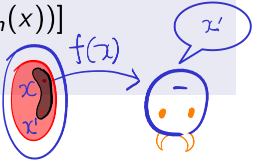
Intuitively: "easy to compute" function that is "hard to invert"

Definition 3 (One-way function (OWF))

A function family $f := \left\{f_n : \{0,1\}^n \to \{0,1\}^{m(n)}\right\}_{n\in\mathbb{N}}$ is one-way if

- there exists an efficient algorithm F such that $\forall x : \mathsf{F}(x) = f(x)$
- for all PPT *inverters* Inv, the following is negligible:

$$p(n) := \Pr_{x \leftarrow \{0,1\}^n}[\mathsf{Inv}(f_n(x)) \in f_n^{-1}(f_n(x))]$$



- Length-preserving OWF: $m(n) = n$
- One-way permutation: f is length-preserving and *bijective*
- Convenient to consider "collection" of OWF:

$$\{f_I : \mathcal{D}_I \to \mathcal{R}_I\}_{I\subseteq\{0,1\}^*}$$

OWF or Not?

- Some generic constructions:
  1. $f_1(x) := f(x) \| 0^{|x|}$, where f is a OWF
  2. $f_2(x_1 \| x_2) := x_1 \| f(x_2)$, where f is a OWF and $|x_1| - |x_2| \leq 1$
  3. $f_3(x_1 \| x_2) := x_1 \| f(x_1 \| x_2)$, where f is a OWF and $|x_1| - |x_2| \leq 1$
  4. $f_4(x) := \mathsf{G}(x)$, where G is a PRG

OWF or Not?

- Some generic constructions:
  1. $f_1(x) := f(x) \| 0^{|x|}$, where f is a OWF
  2. $f_2(x_1 \| x_2) := x_1 \| f(x_2)$, where f is a OWF and $|x_1| - |x_2| \leq 1$
  3. ? $f_3(x_1 \| x_2) := x_1 \| f(x_1 \| x_2)$, where f is a OWF and $|x_1| - |x_2| \leq 1$
  4. $f_4(x) := \mathsf{G}(x)$, where G is a PRG

- A concrete construction:
  4. $f_5(x_1 \| x_2) := x_1 \cdot x_2$, where x_1 and x_2 are parsed as integers

OWF or Not?

- Some generic constructions:
  1. $f_1(x) := f(x) \| 0^{|x|}$, where f is a OWF
  2. $f_2(x_1 \| x_2) := x_1 \| f(x_2)$, where f is a OWF and $|x_1| - |x_2| \leq 1$
  3. ? $f_3(x_1 \| x_2) := x_1 \| f(x_1 \| x_2)$, where f is a OWF and $|x_1| - |x_2| \leq 1$
  4. $f_4(x) := \mathsf{G}(x)$, where G is a PRG

- A concrete construction:
  4. $f_5(x_1 \| x_2) := x_1 \cdot x_2$, where x_1 and x_2 are parsed as integers
     - "Weakly" one-way since primes are dense enough

OWF or Not?

- Some generic constructions:
  1. $f_1(x) := f(x)\|0^{|x|}$, where f is a OWF
  2. $f_2(x_1\|x_2) := x_1\|f(x_2)$, where f is a OWF and $|x_1| - |x_2| \leq 1$
  3. $f_3(x_1\|x_2) := x_1\|f(x_1\|x_2)$, where f is a OWF and $|x_1| - |x_2| \leq 1$
  4. $f_4(x) := \mathsf{G}(x)$, where G is a PRG

?

- A concrete construction:
  4. $f_5(x_1\|x_2) := x_1 \cdot x_2$, where x_1 and x_2 are parsed as integers
     - "Weakly" one-way since primes are dense enough

Exercise 2

1. Show using security reduction that f_1, f_2 and f_4 are OWFs
2. Come up fs such that f_3 i) remains one-way and ii) becomes invertible

We've Already Seen Some OWF Collections!

large constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime p: $f_{p,c}(x) := cx \bmod p$

We've Already Seen Some OWF Collections!

large constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime p: $f_{p,c}(x) := cx \bmod p$
2. Matrix multiplication modulo prime p: $f_{\overline{A}}(\overline{x}) := \overline{x}^T \overline{A} \bmod p$

   $n \times m$ matrix over $\mathbb{Z}_p^*$

We've Already Seen Some OWF Collections!

large   constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime p: $f_{p,c}(x) := cx \bmod p$
2. Matrix multiplication modulo prime p: $f_{\overline{A}}(\overline{x}) := \overline{x}^T \overline{A} \bmod p$ $n \times m$ matrix over $\mathbb{Z}_p^*$
   - Inversion easy by Gaussian elimination
3. Squaring modulo prime p: $f_p(x) := x^2 \bmod p$
4. Squaring modulo semiprime $N = pq$: $f_N(x) := x^2 \bmod N$

We've Already Seen Some OWF Collections!

large   constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime p: $f_{p,c}(x) := cx \bmod p$
2. Matrix multiplication modulo prime p: $f_{\overline{A}}(\overline{x}) := \overline{x}^T\overline{A} \bmod p$
   - Inversion easy by Gaussian elimination $\rightsquigarrow$ $n \times m$ matrix over $\mathbb{Z}_p^*$
3. Squaring modulo prime p: $f_p(x) := x^2 \bmod p$
4. Squaring modulo semiprime $N = pq$: $f_N(x) := x^2 \bmod N$
   - Inversion as hard as factoring N

   generator
5. Exponentiation modulo prime p: $f_{p,g}(x) := g^x \bmod p$

We've Already Seen Some OWF Collections!

large — constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime p: $f_{p,c}(x) := cx \bmod p$
2. Matrix multiplication modulo prime p: $f_{\overline{A}}(\overline{x}) := \overline{x}^T \overline{A} \bmod p$ — $n \times m$ matrix over $\mathbb{Z}_p^*$
   - Inversion easy by Gaussian elimination
3. Squaring modulo prime p: $f_p(x) := x^2 \bmod p$
4. Squaring modulo semiprime $N = pq$: $f_N(x) := x^2 \bmod N$
   - Inversion as hard as factoring N
5. Exponentiation modulo prime p: $f_{p,g}(x) := g^x \bmod p$ — generator
   - Inversion is the Discrete Logarithm Problem: believed hard
6. Power map modulo semiprime $N = pq$: $f_{N,e}(x) := x^e \bmod N$

We've Already Seen Some OWF Collections!

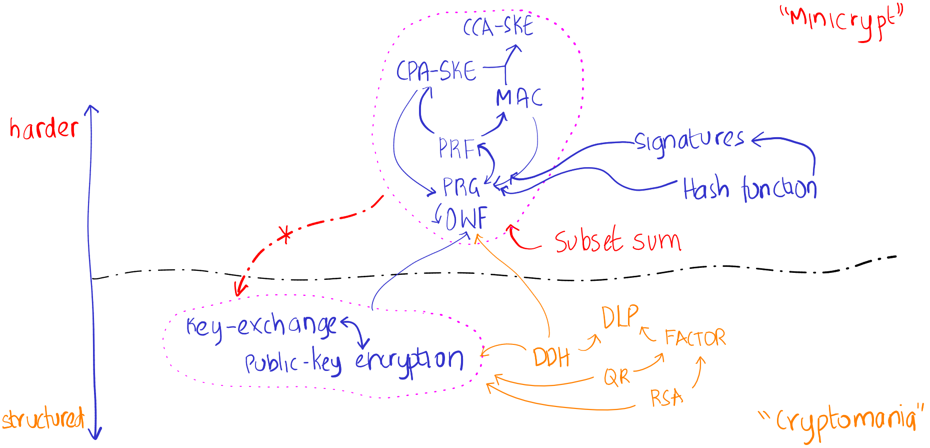
large constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime p: $f_{p,c}(x) := cx \bmod p$
2. Matrix multiplication modulo prime p: $f_{\bar{A}}(\bar{x}) := \bar{x}^T \bar{A} \bmod p$
   - Inversion easy by Gaussian elimination $n \times m$ matrix over $\mathbb{Z}_p^*$
3. Squaring modulo prime p: $f_p(x) := x^2 \bmod p$
4. Squaring modulo semiprime $N = pq$: $f_N(x) := x^2 \bmod N$
   - Inversion as hard as factoring N generator
5. Exponentiation modulo prime p: $f_{p,g}(x) := g^x \bmod p$
   - Inversion is the Discrete Logarithm Problem: believed hard
6. Power map modulo semiprime $N = pq$: $f_{N,e}(x) := x^e \bmod N$
   - Inversion is the RSA problem: believed hard

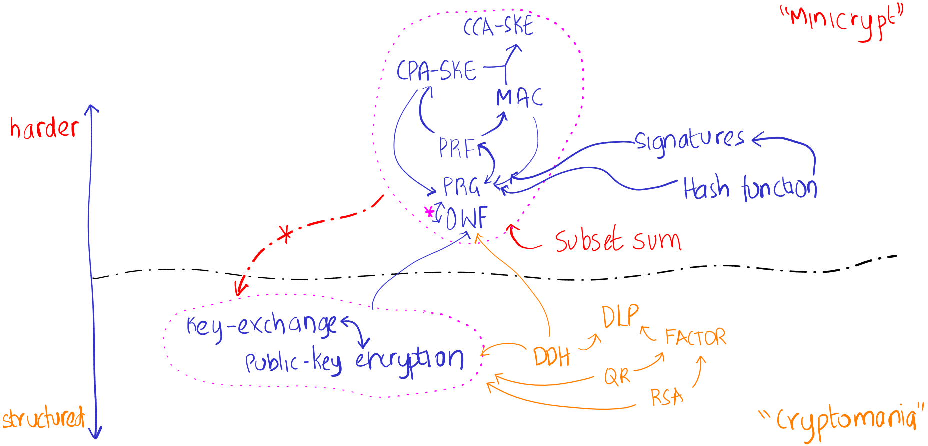
Exercise 3

Show that taking square root modulo N is equivalent to factoring N.
(Hint: use the identity $x^2 - y^2 = (x + y)(x - y) \bmod N$)

The Cryptographic Landscape



The Cryptographic Landscape



* **Theorem 1 ([HILL99, BM82])**

If one-way functions exist then so do pseudo-random generators

Plan for Today's Lecture...

- Task: integrity and authentication in the *public-key* setting
- Threat model: EU-CMA

NEW Digital Signature

©Wikipedia

One-Way Function NEW

©Lubos Houska

Proof technique: plug and pray

One-Time DS ($q = 1$): Lamport's Signature

$\{f_n : \{0,1\}^n \to \{0,1\}^n\}_n$

Construction 1 (OWF $f \to$ one-time DS Σ for $\mathcal{M} := \{0,1\}^\ell$)

$\{f_n : \{0,1\}^n \to \{0,1\}^n\}_n$

Construction 1 (OWF $f \to$ one-time DS Σ for $\mathcal{M} := \{0,1\}^\ell$)

One-Time DS ($q = 1$): Lamport's Signature

$\{f_n : \{0,1\}^n \to \{0,1\}^n\}_n$

Construction 1 (OWF f → one-time DS Σ for $\mathcal{M} := \{0,1\}^{\ell}$) $^{=4}$

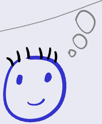
$x_{ib} \leftarrow \{0,1\}^n$

x_{00}	x_{10}	x_{20}	x_{30}
x_{01}	x_{11}	x_{21}	x_{31}

$= sk$

$Gen(1^n)$

ALICE
(VERIFIER)

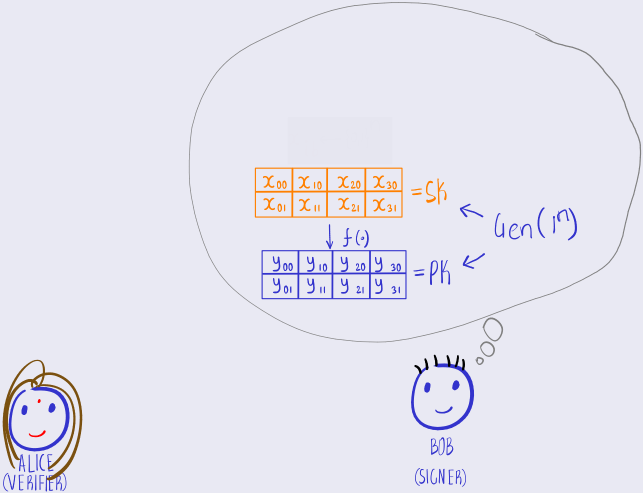


BOB
(SIGNER)

One-Time DS ($q = 1$): Lamport's Signature

$\{f_n : \{0,1\}^n \to \{0,1\}^n\}_n$

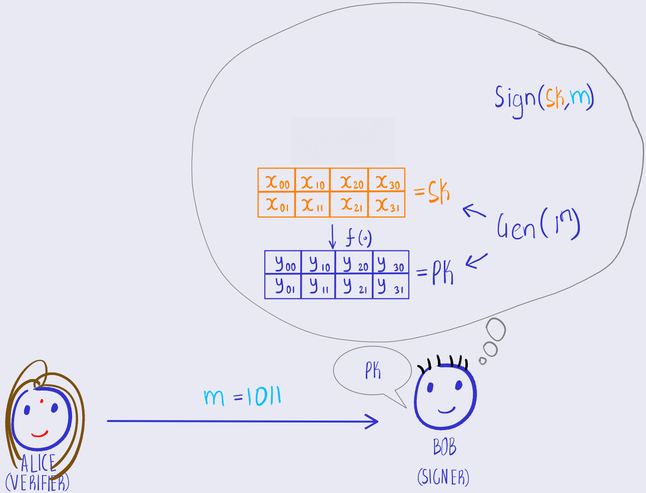
Construction 1 (OWF f $\to$ one-time DS Σ for $\mathcal{M} := \{0,1\}^{\ell}$) $=4$



One-Time DS ($q = 1$): Lamport's Signature

$\{f_n : \{0,1\}^n \to \{0,1\}^n\}_n$

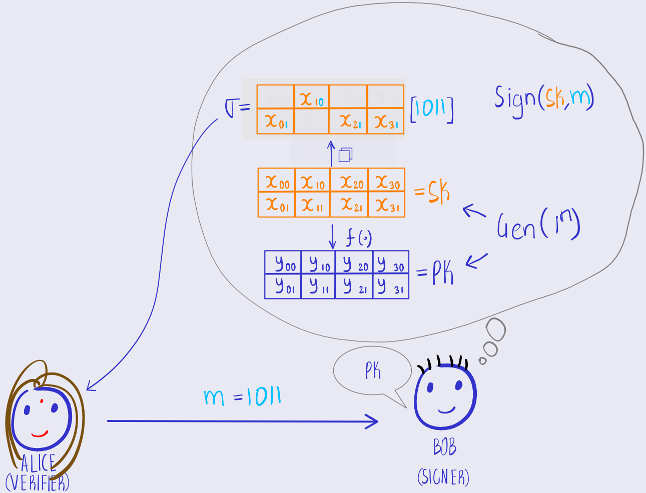
Construction 1 (OWF f → one-time DS Σ for $\mathcal{M} := \{0,1\}^{\ell \, =4}$)



One-Time DS ($q = 1$): Lamport's Signature

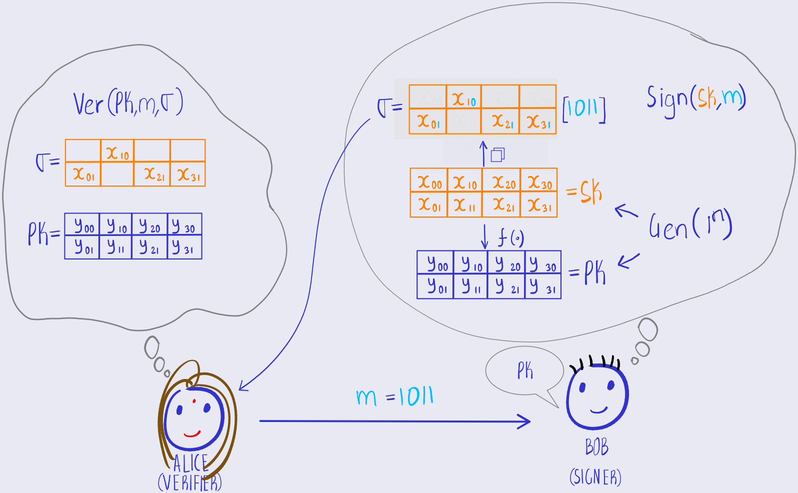
$\{f_n : \{0,1\}^n \to \{0,1\}^n\}_n$

Construction 1 (OWF f → one-time DS Σ for $\mathcal{M} := \{0,1\}^{\ell}$) $\ell = 4$



$\{f_n : \{0,1\}^n \to \{0,1\}^n\}_n$

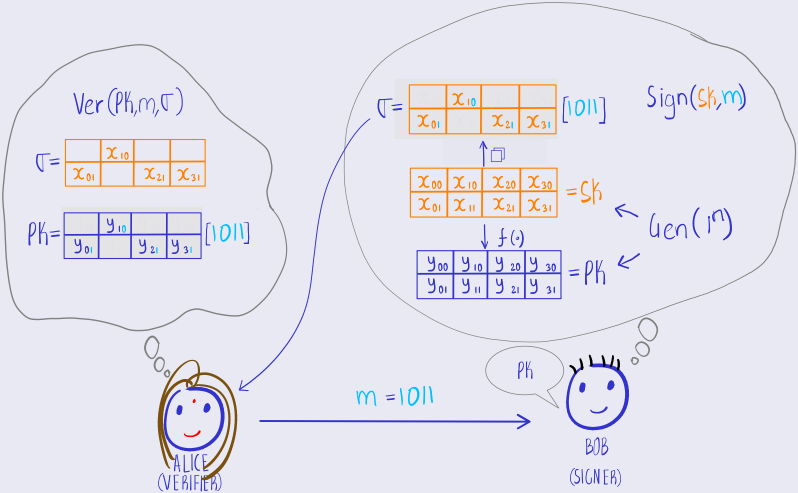
Construction 1 (OWF $f \to$ one-time DS Σ for $\mathcal{M} := \{0,1\}^{\ell}$) $=4$



One-Time DS ($q = 1$): Lamport's Signature

$\{f_n : \{0,1\}^n \to \{0,1\}^n\}_n$

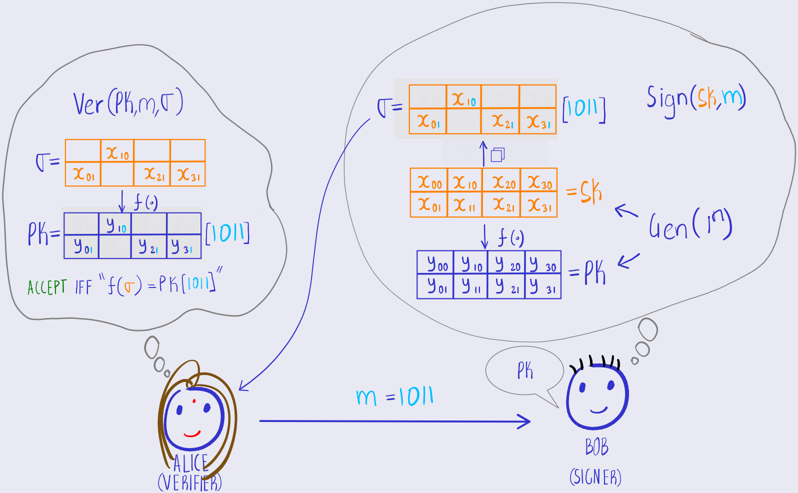
Construction 1 (OWF $f \to$ one-time DS Σ for $\mathcal{M} := \{0,1\}^{\ell}$) $^{=4}$



One-Time DS ($q = 1$): Lamport's Signature

$\{f_n : \{0,1\}^n \to \{0,1\}^n\}_n$

Construction 1 (OWF f → one-time DS Σ for $\mathcal{M} := \{0,1\}^{\ell}$) $\ell = 4$



Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".

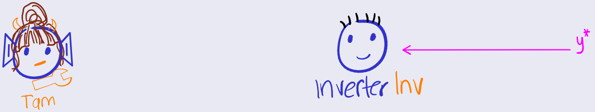
□

Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".

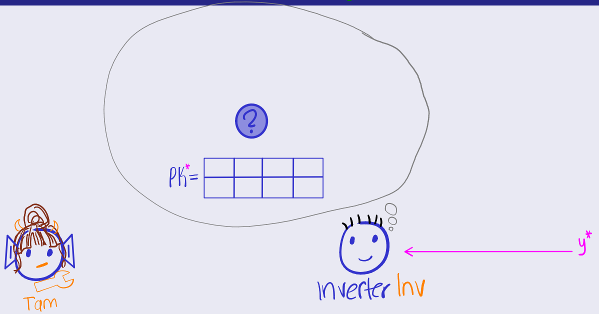


Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".

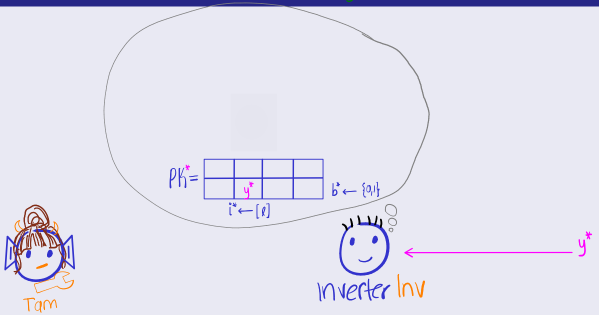


Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".



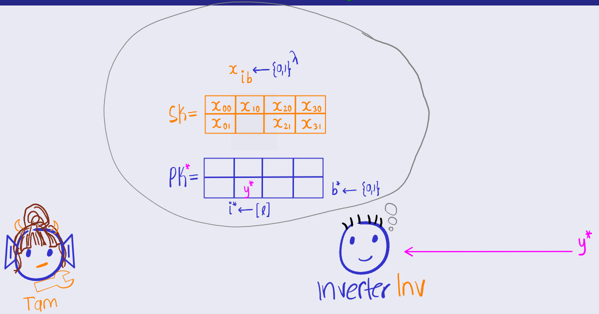
$\square$

Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".

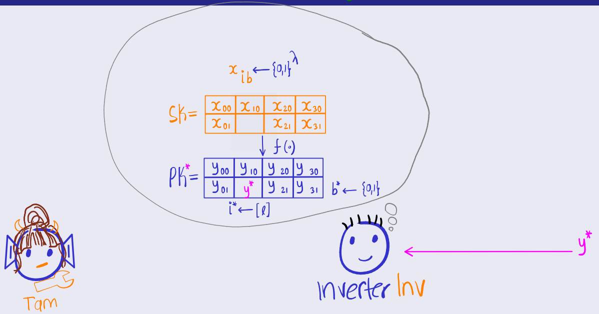


□

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".

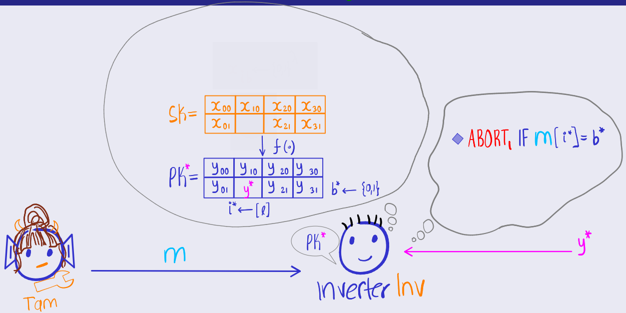


$\square$

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".

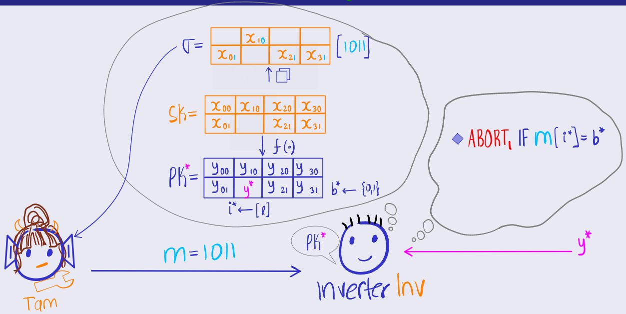


□

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".

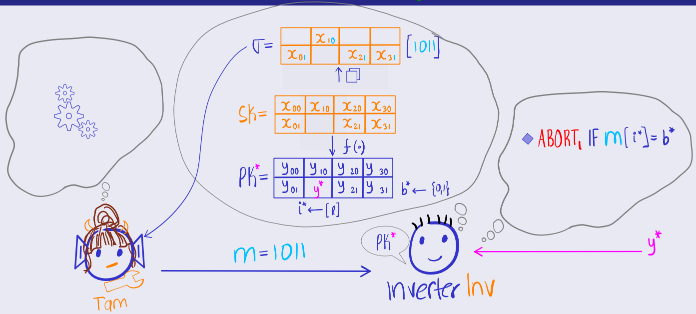


Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".

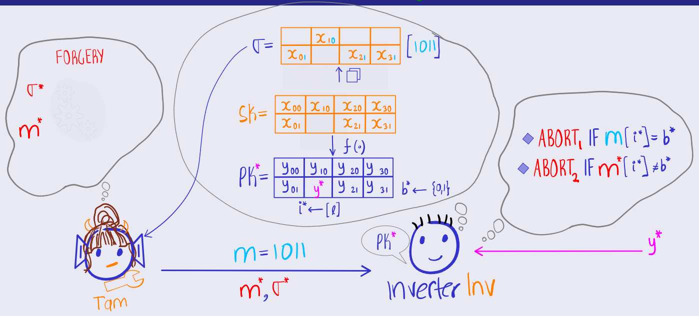


□

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".



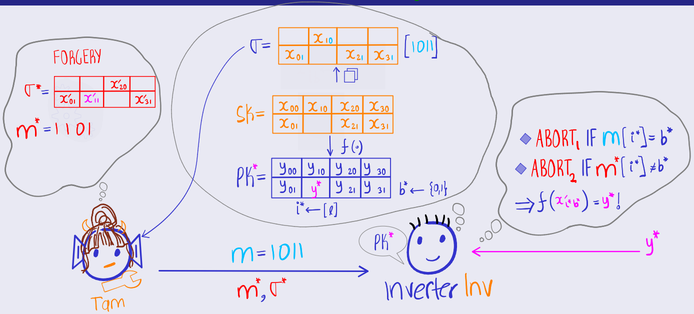
□

Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".

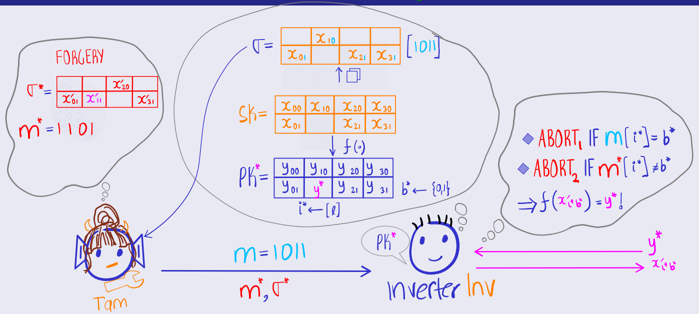


Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

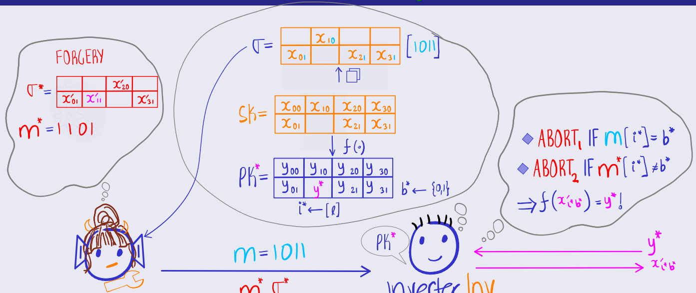
Proof sketch: proof by reduction. Idea: "plug and pray".



Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".



Analysis

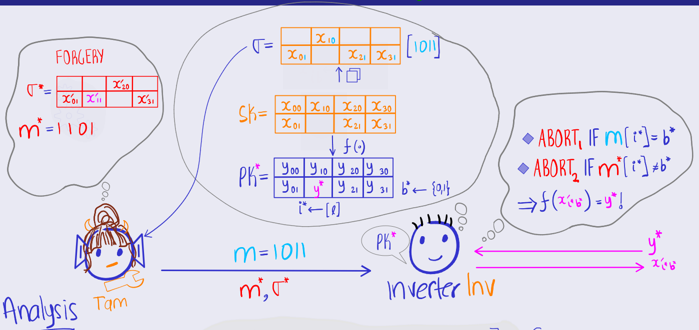
$$\Pr_{x}[\text{Inv inverts } f(x)] = \Pr_{x,i^*,b^*,sk,m,m^*}\left[\overline{\text{ABORT}_2}, \text{Tam breaks}, \overline{\text{ABORT}_1}\right]$$

Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".



Analysis

$$\Pr_x[\text{Inv inverts } f(x)] = \Pr_{\substack{x,i^*,b^* \\ sk,m,m^*}}[\text{ABORT}_2 \mid \text{Tam breaks}, \overline{\text{ABORT}_1}]\ \Pr_{\star}[\text{Tam breaks} \mid \overline{\text{ABORT}_1}]\ \Pr_{\star}[\overline{\text{ABORT}_1}]$$

independence of b^* → $1/2$

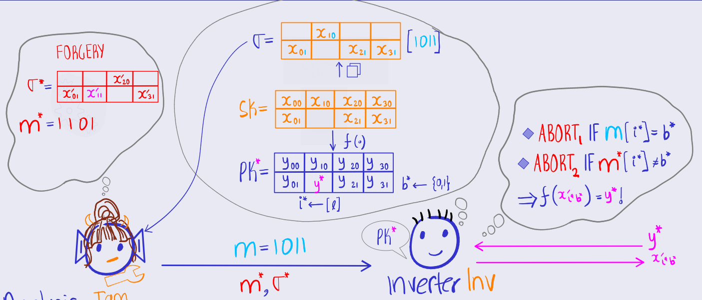
□

Lamport's Signature is One-Time Secure...

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".



Analysis

$$\Pr_x[\text{Inv inverts } f(x)] = \Pr_{\substack{x,i^*,b^*\\ sk,m,m^*}}\left[\text{ABORT}_2 \mid \text{Tam breaks}, \overline{\text{ABORT}_1}\right] \Pr_{\star}\left[\text{Tam breaks} \mid \overline{\text{ABORT}_1}\right] \Pr_{\star}\left[\overline{\text{ABORT}_1}\right]$$

$\geq 1/p(n)$ independence of b^*

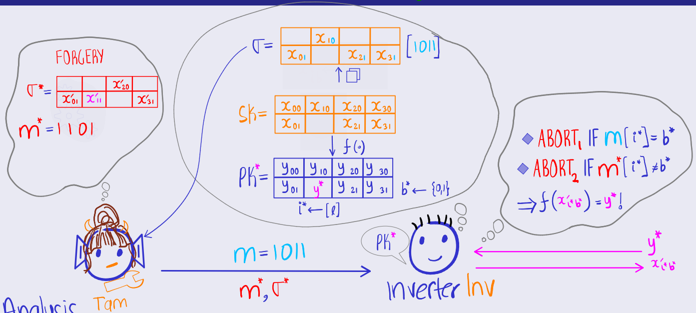
$1/2$

$\square$

Theorem 2

If f is a OWF then Lamport's scheme is a one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".



Analysis

$$\Pr_x[\text{Inv inverts } f(x)] = \Pr_{\substack{x,i^*,b^*\\ sk,m,m^*}}^{\star}\left[\overline{\text{ABORT}_2} \mid \text{Tam breaks}, \overline{\text{ABORT}_1}\right] \Pr^{\star}\left[\text{Tam breaks} \mid \overline{\text{ABORT}_1}\right] \Pr^{\star}\left[\overline{\text{ABORT}_1}\right]$$

independence of i^*: $1/\ell$

$\geq 1/p(n)$

independence of b^*: $1/2$

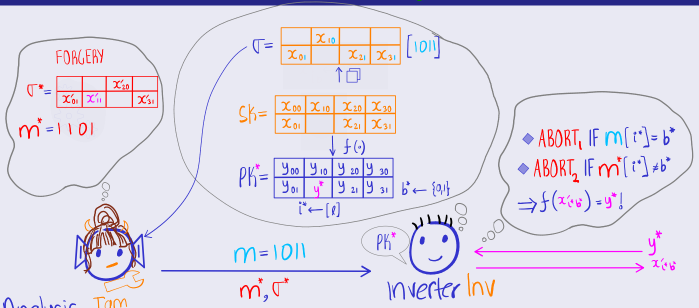
$\geq 1/p(n) \cdot 2\ell$ ⚡

□

Theorem 2

If f is a quantum-secure OWF then Lamport's scheme is a quantum-secure one-time DS.

Proof sketch: proof by reduction. Idea: "plug and pray".



Analysis

$$\Pr_{x}[\text{Inv inverts } f(x)] = \Pr_{\substack{x,i^*,b^*\\ sk,m,m^*}}[\overline{ABORT_2} \mid \text{Tam breaks}, \overline{ABORT_1}] \cdot \Pr_{\star}[\text{Tam breaks} \mid \overline{ABORT_1}] \cdot \Pr_{\star}[\overline{ABORT_1}]$$

independence of i^*: $1/\ell$

$\geq 1/p(n)$

independence of b^*: $1/2$

$\geq 1/p(n) \cdot 2\ell$ ⚡

□

Lamport's Signature is One-Time Secure...

Exercise 4

- Can a forger break EU-CMA given *two* signatures?
- Are the signatures *unique*? If not, can it be made unique?

Lamport's Signature is One-Time Secure...

Exercise 4

- Can a forger break EU-CMA given *two* signatures?
- Are the signatures *unique*? If not, can it be made unique?
- Can we avoid the $1/2\ell$ loss in inverting advantage?

Theorem 3

If f is a OWF then Lamport's scheme is a one-time DS

Lamport's Signature is One-Time Secure...

Exercise 4

- Can a forger break EU-CMA given *two* signatures?
- Are the signatures *unique*? If not, can it be made unique?
- Can we avoid the $1/2\ell$ loss in inverting advantage?

Theorem 3

If f is a OWF then Lamport's scheme is a one-time DS for fixed-length messages.

Exercise 5 (Domain Extension)

Given a compressing function $H : \{0,1\}^{2\ell} \to \{0,1\}^{\ell}$, construct a one-time DS for *arbitrary-length* messages. What are the properties you need from H to ensure that the one-time DS is secure?

How to Sign Many Times?

Theorem 4 ([Mer90, Gol87])

If one-time DS and PRFs exists then many-time DS exists

How to Sign Many Times?

Theorem 4 ([Mer90, Gol87])

If one-time DS and PRFs exists then many-time DS exists

Proof (Overview).

1. Step I: One-time DS $\Rightarrow$ many-time *stateful* DS
   - Stateful DS: Sign is stateful
   - Idea: use one-time DS to sign message *and next public key*
   - Proof uses plug and pray

How to Sign Many Times?

Theorem 4 ([Mer90, Gol87])

If one-time DS and PRFs exists then many-time DS exists

Proof (Overview).

1. Step I: One-time DS $\Rightarrow$ many-time *stateful* DS
   - Stateful DS: Sign is stateful
   - Idea: use one-time DS to sign message *and next public key*
   - Proof uses plug and pray
2. Step II: Many-time *stateful* DS $\Rightarrow$ Many-time DS
   - Use PRF to *derandomise* Step I

$\square$

Recap/Next Lecture

- Introduced digital signatures: public-key analogue of MAC
- Theoretical constructions of DS
  - Lamport's one-time DS
  - Generic transformation from one-time to many-time DS
  - Takeaway: "Plug and pray"

Recap/Next Lecture

- Introduced digital signatures: public-key analogue of MAC
- Theoretical constructions of DS
  - Lamport's one-time DS
  - Generic transformation from one-time to many-time DS
  - Takeaway: "Plug and pray"
- Lectures 17: *efficient* DS in random-oracle model
  - From trapdoor OWF via hash-then-invert
  - Via Fiat-Shamir transform (e.g., Schnorr)

Recap/Next Lecture

- Introduced digital signatures: public-key analogue of MAC
- Theoretical constructions of DS
  - Lamport's one-time DS
  - Generic transformation from one-time to many-time DS
  - Takeaway: "Plug and pray"
- Lectures 17: *efficient* DS in random-oracle model
  - From trapdoor OWF via hash-then-invert
  - Via Fiat-Shamir transform (e.g., Schnorr)

Exercise 5 (Domain Extension)

Given a compressing function $H : \{0,1\}^{2\ell} \to \{0,1\}^{\ell}$, construct a one-time DS for *arbitrary-length* messages. What are the properties you need from H to ensure that the one-time DS is secure?

- Next lecture: How to sign longer messages?
  - New primitive: collision-resistant hash functions

References

1. Refer to [KL14, Chapters 13.1 and 13.2] for motivation and definition of DS.
2. The construction of one-time DS and the subsequent generic transform (Theorem 4) can be found in [KL14, Chapter 14.4]
3. For a historical take on OWFs, see [DH76].
4. The construction of PRG from OWF is due to [HILL99], building on the construction of PRF from OWP from [BM82].

Manuel Blum and Silvio Micali.
How to generate cryptographically strong sequences of pseudo random bits.
In *23rd FOCS*, pages 112–117. IEEE Computer Society Press, November 1982.

Whitfield Diffie and Martin E. Hellman.
New directions in cryptography.
IEEE Trans. Inf. Theory, 22(6):644–654, 1976.

Oded Goldreich.
Two remarks concerning the Goldwasser-Micali-Rivest signature scheme.
In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 104–110. Springer, Berlin, Heidelberg, August 1987.

Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby.
A pseudorandom generator from any one-way function.
SIAM J. Comput., 28(4):1364–1396, 1999.

Jonathan Katz and Yehuda Lindell.
Introduction to Modern Cryptography (3rd ed.).
Chapman and Hall/CRC, 2014.

Ralph C. Merkle.
A certified digital signature.