

Digital Signatures

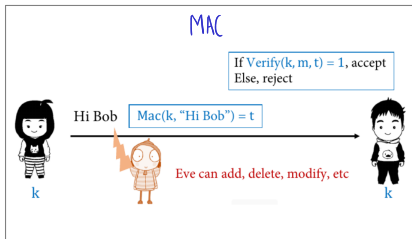
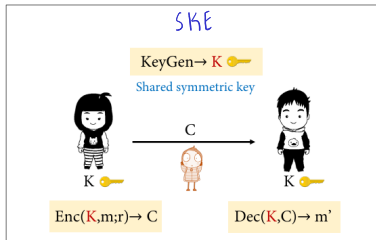
Chethan Kamath



Recall from Prior Sessions

SYMMETRIC KEY UNIVERSE

PUBLIC KEY UNIVERSE




Recall from Prior Sessions

SYMMETRIC KEY UNIVERSE

PUBLIC KEY UNIVERSE

SKE

KeyGen \rightarrow K 
Shared symmetric key



C



$Enc(K,m;r) \rightarrow C$

$Dec(K,C) \rightarrow m'$

PKE

KeyGen $\rightarrow (PK,SK)$

Public and secret key



C



$Enc(PK,m;r) \rightarrow C$

$Dec(SK,C) \rightarrow m'$

MAC

If $Verify(k, m, t) = 1$, accept
Else, reject



Hi Bob

$Mac(k, "Hi Bob") = t$



Eve can add, delete, modify, etc


Recall from Prior Sessions

SYMMETRIC KEY UNIVERSE

PUBLIC KEY UNIVERSE

SKE

PKE

KeyGen \rightarrow K 
Shared symmetric key

KeyGen \rightarrow (PK, SK)

C

Public and secret key



Enc($K, m; r$) \rightarrow C

Dec(K, C) \rightarrow m'

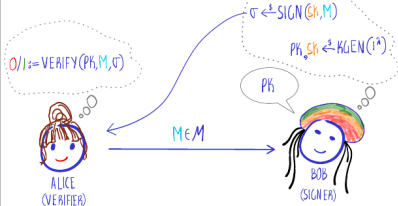
Enc($PK, m; r$) \rightarrow C

Dec(SK, C) \rightarrow m'

MAC

SIGNATURES

If Verify(k, m, t) = 1, accept
Else, reject



Hi Bob

Mac($k, \text{"Hi Bob"}$) = t

Eve can add, delete, modify, etc



Plan for this Session

Digital Signature: Syntax and Modelling Security

Plan for this Session

Digital Signature: Syntax and Modelling Security

One-Time Signatures

Plan for this Session

Digital Signature: Syntax and Modelling Security

One-Time Signatures

Many-Time (Stateful) Signatures

Plan for this Session

Digital Signature: Syntax and Modelling Security

One-Time Signatures

Many-Time (Stateful) Signatures

Efficient Signatures via Hash-and-Sign

Plan for this Session

Digital Signature: Syntax and Modelling Security

One-Time Signatures

Many-Time (Stateful) Signatures

Efficient Signatures via Hash-and-Sign

Wrapping Up

Plan for this Session

Digital Signature: Syntax and Modelling Security

One-Time Signatures

Many-Time (Stateful) Signatures

Efficient Signatures via Hash-and-Sign

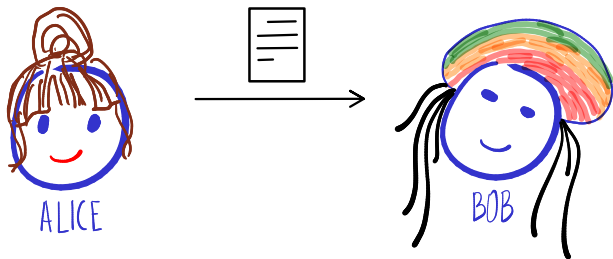
Wrapping Up

Digital (Analogues of Physical) Signatures...



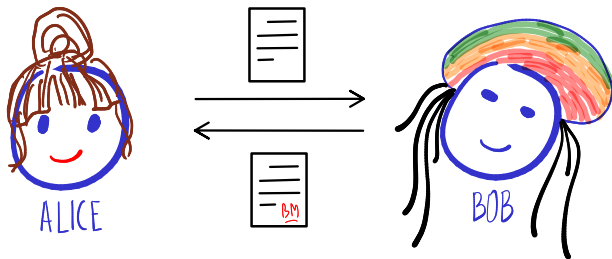
► Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...



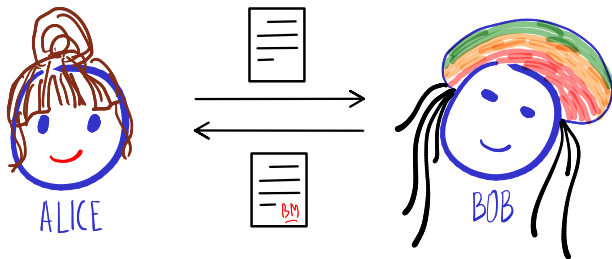
► Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...



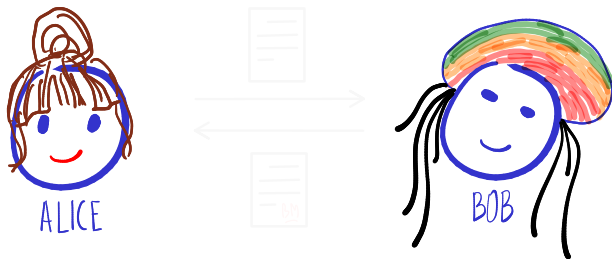
► Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...



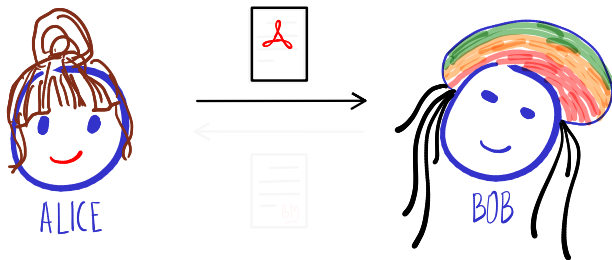
- ▶ Requirement: no one should be able to **forge** Bob's signature

Digital (Analogues of Physical) Signatures...



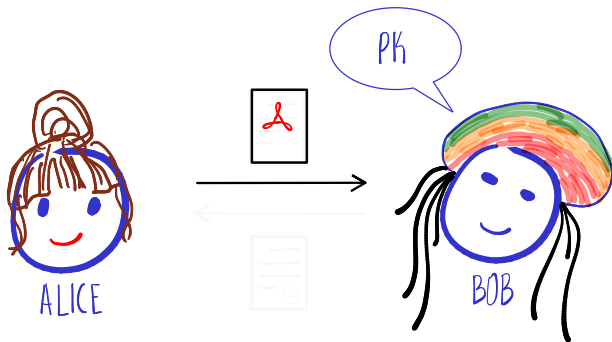
- ▶ Requirement: no one should be able to **forge** Bob's signature

Digital (Analogues of Physical) Signatures...



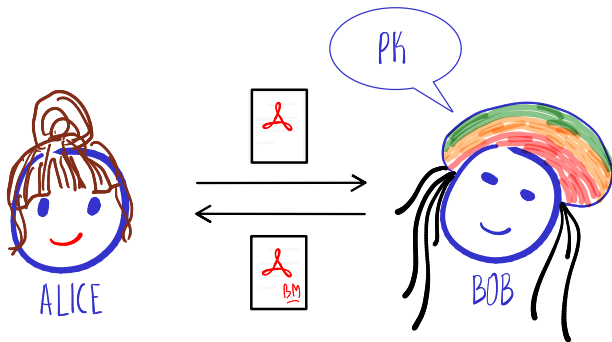
- ▶ Requirement: no one should be able to **forge** Bob's signature

Digital (Analogues of Physical) Signatures...



- ▶ Requirement: no one should be able to **forge** Bob's signature

Digital (Analogues of Physical) Signatures...



- ▶ Requirement: no one should be able to **forge** Bob's signature

Digital (Analogues of Physical) Signatures...

The image shows a browser window with the Wikipedia homepage. A red arrow points from the 'View Certificate' button in the 'Page Info' panel to a separate window displaying the certificate details. The certificate window shows the following information:

Miscellaneous

- Serial Number: 07:41:0E:39:38:3A:4C:76:CF:1E:A1:43:47:FA:5F:3A
- Signature Algorithm: ECDSA with SHA-384
- Download: [PEM \(.cert\)](#) [PEM \(.chain\)](#)

Fingerprints

- SHA-256: 07:BC:DE:69:CD:24:AB:9A:25:91:B9:2F:85:51:07:40:8E:92:7F:E7:75...
- SHA-1: 48:3F:0C:71:F3:4A:E0:EA:30:D9:9B:D6:04:63:DC:DA:AB:F4:90:FB

Digital (Analogues of Physical) Signatures...

The screenshot shows a web browser displaying the Wikipedia homepage. A security warning is visible, and the 'Security' tab is active, showing the following information:

- Website Identity:** www.wikipedia.org, Owner: This website does not supply ownership information, Verified by: DigiCert Inc.
- Privacy & History:** Have I visited this website prior to today? Yes, 3 times; Is this website storing information on my computer? Yes, cookies and 5.5 MB of site data; Have I saved any passwords for this website? No.
- Technical Details:** Connection Encrypted (TLS, AES, 256, GCM, SHA384, 256 bit keys, TLS 1.3)

Below the security warning, the 'Certificate for *.wikipedia.org' is displayed with the following details:

- Miscellaneous:** Serial Number: 07:41:0E:39:38:3A:4C:76:CF:1E:A1:43:47:FA:5F:3A; Signature Algorithm: ECDSA with SHA-384; Version: 3; Download: PEM (.cert) | PEM (.chain)
- Fingerprints:** SHA-256: 07:BC:DE:69:CD:24:AB:9A:25:91:B9:2F:85:51:07:40:8E:92:7F:E7:75...; SHA-1: 48:3F:0C:71:F3:4A:E0:EA:30:D9:9B:D6:04:63:DC:DA:AB:F4:9D:FB

The screenshot shows a Google Cloud documentation page titled "How Google enforces boot integrity on production machines". The page includes a navigation menu with "Introduction", "Background", "Machine credentials", "Hardware roots of trust and cryptographic sealing", and "Signed credentials".

Measured boot process

Google machines' boot stack consists of four layers, which are visualized in the following diagram.

```
graph TD;
    U[Userspace] --- M1[Measures];
    S[System software] --- M2[Measures];
    B[Boot firmware] --- M3[Measures];
    H[Hardware root of trust] --- M4[Measures];
```

The diagram illustrates the measured boot process with four layers: Userspace (yellow), System software (pink), Boot firmware (blue), and Hardware root of trust (green). Each layer is connected to a "Measures" label, which is circled in red.

Digital (Analogues of Physical) Signatures...

The screenshot shows a web browser displaying the Wikipedia homepage. A security information overlay is visible, showing the following details:

- Website Identity:** www.wikipedia.org, Owner: This website does not supply ownership information, Verified by: DigiCert Inc.
- Privacy & History:** Have I visited this website prior to today? Yes, 3 times; Is this website storing information on my computer? Yes, cookies and 5.5 MB of site data; Have I saved any passwords for this website? No.
- Technical Details:** Connection Encrypted (TLS, AES, 256, GCM, SHA384, 256 bit keys, TLS 1.3)

Below the overlay, the browser's certificate details are shown:

- Miscellaneous:** Serial Number: 07:41:0E:39:38:3A:4C:76:CF:1E:A1:43:47:FA:5F:3A; Signature Algorithm: ECDSA with SHA-384; Version: 3; Download: PEM (.cert) | PEM (.chain)
- Fingerprints:** SHA-256: 07:BC:DE:69:CD:24:AB:9A:25:91:B9:2F:85:51:07:40:8E:92:7F:E7:75...; SHA-1: 48:3F:0C:71:F3:4A:E0:EA:30:D9:9B:D6:04:63:DC:DA:AB:F4:9D:FB

The screenshot shows a Google Cloud documentation page titled "How Google enforces boot integrity on production machines". The page includes a table of contents and a diagram of the boot stack.

Table of Contents:

- On this page
- Introduction
- Background
- Machine credentials
- Hardware roots of trust and cryptographic sealing
- Sealed credentials
- ...

Measured boot process

Google machines' boot stack consists of four layers, which are visualized in the following diagram.

```
graph TD;
    U[Userspace] --- M1[Measure];
    S[System software] --- M2[Measure];
    B[Boot firmware] --- M3[Measure];
    H[Hardware root of trust] --- M4[Measure];
```

► Application to blockchains protocols like Algorand and Chia.

Syntax

- ▶ *Public-key* analogue of message authentication codes (MAC)
 - ▶ Triple of algorithms: (KGEN, SIGN, VERIFY)

Syntax

- ▶ *Public-key* analogue of message authentication codes (MAC)
 - ▶ Triple of algorithms: (KGEN, SIGN, VERIFY)



ALICE
(VERIFIER)



BOB
(SIGNER)

Syntax

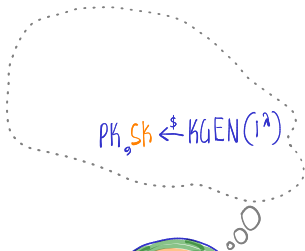
- ▶ *Public-key* analogue of message authentication codes (MAC)
 - ▶ Triple of algorithms: (KGEN, SIGN, VERIFY)



ALICE
(VERIFIER)



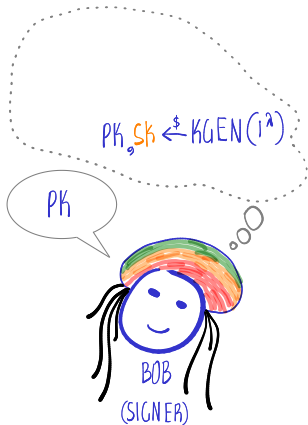
BOB
(SIGNER)



$PK, SK \leftarrow KGEN(i^*)$

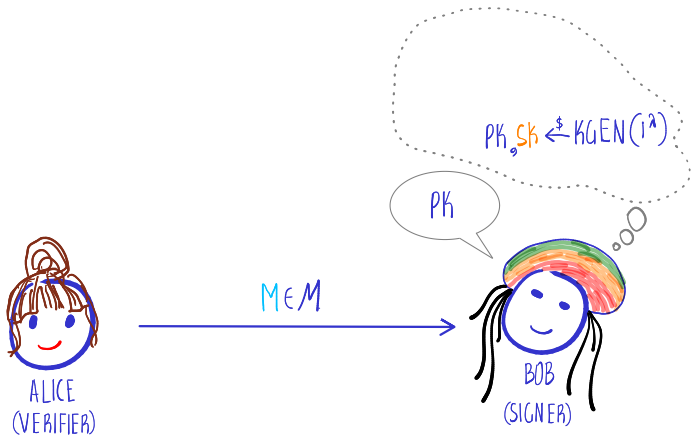
Syntax

- ▶ *Public-key* analogue of message authentication codes (MAC)
 - ▶ Triple of algorithms: (KGEN, SIGN, VERIFY)



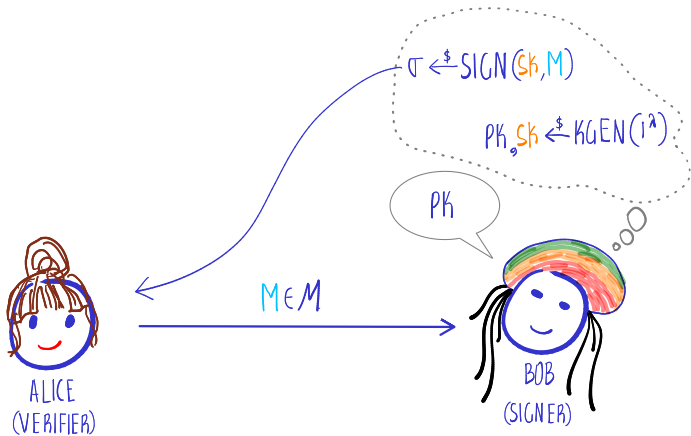
Syntax

- ▶ *Public-key* analogue of message authentication codes (MAC)
 - ▶ Triple of algorithms: (KGEN, SIGN, VERIFY)



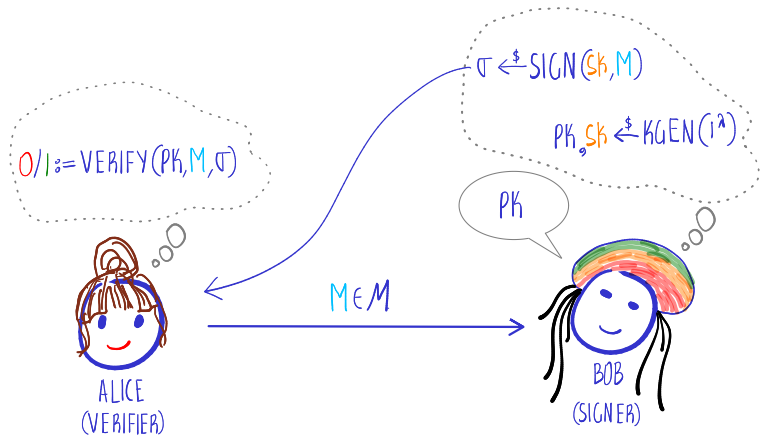
Syntax

- ▶ *Public-key* analogue of message authentication codes (MAC)
 - ▶ Triple of algorithms: (KGEN, SIGN, VERIFY)



Syntax

- ▶ *Public-key* analogue of message authentication codes (MAC)
 - ▶ Triple of algorithms: (KGEN, SIGN, VERIFY)



Security: Universal Unforgeability under Key-Only Attack



"BREAK"

"ATTACK"

Definition 1

A scheme is UU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Universal Unforgeability under Key-Only Attack



"BREAK"

"ATTACK"

(FORGER)



ALICE

(CHALLENGER)



BOB

Definition 1

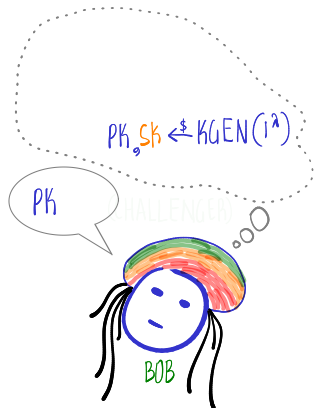
A scheme is UU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Universal Unforgeability under Key-Only Attack



"BREAK"

"ATTACK"



Definition 1

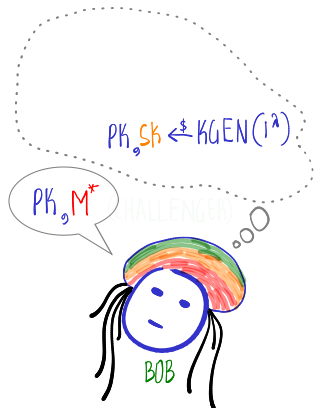
A scheme is UU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Universal Unforgeability under Key-Only Attack



"BREAK"

"ATTACK"



Definition 1

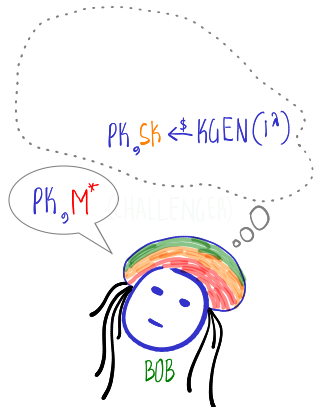
A scheme is UU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Universal Unforgeability under Key-Only Attack



"BREAK"

"ATTACK"



Definition 1

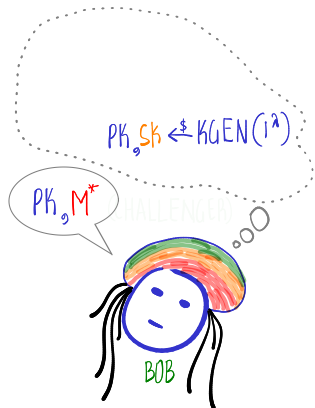
A scheme is UU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Universal Unforgeability under Key-Only Attack



"BREAK"

"ATTACK"



Definition 1

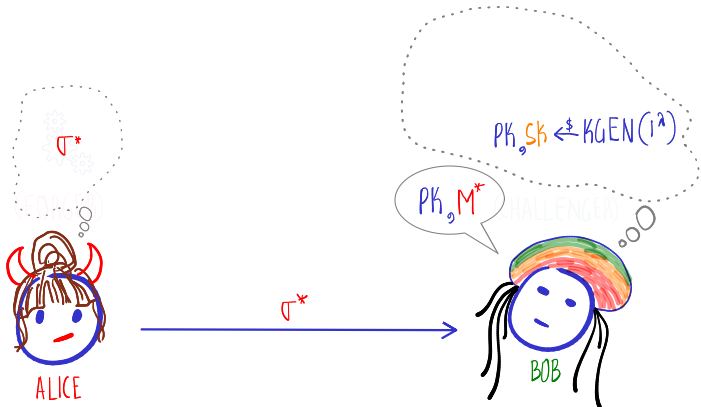
A scheme is UU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Universal Unforgeability under Key-Only Attack



"BREAK"

"ATTACK"



Definition 1

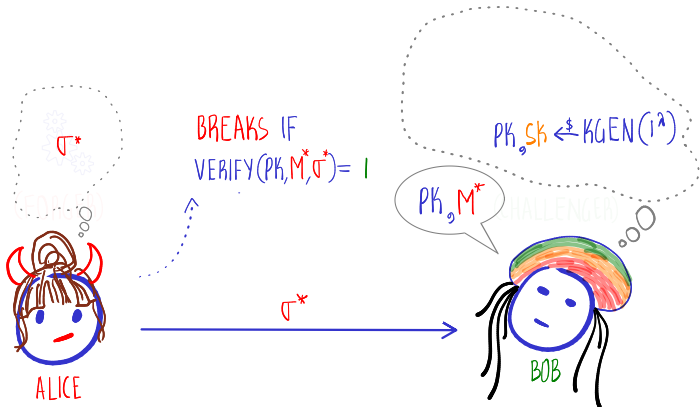
A scheme is UU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Universal Unforgeability under Key-Only Attack



"BREAK"

"ATTACK"



Definition 1

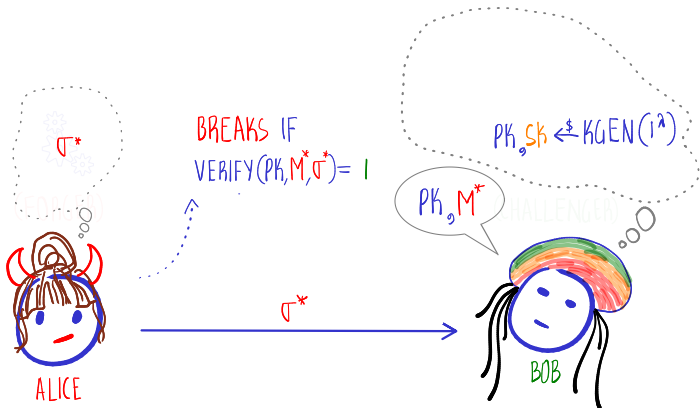
A scheme is UU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Universal Unforgeability under Key-Only Attack



"BREAK"

"ATTACK"



Definition 1

A scheme is UU-KOA secure if no PPT **forg**er can **break** as above with a *non-negligible* probability.

Security: Existential Unforg. under Key-Only Attack



$\forall \epsilon \rightarrow]$

Definition 2

A scheme is EU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

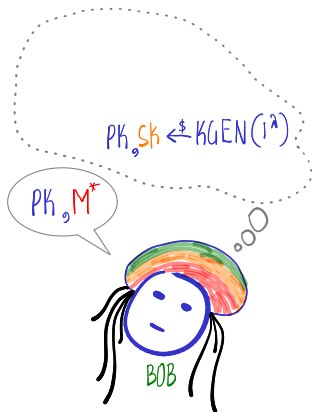
Security: Existential Unforg. under Key-Only Attack



$V \rightarrow \{$



ALICE



PK, M^*

$PK, SK \leftarrow \text{KGEN}(1^\lambda)$

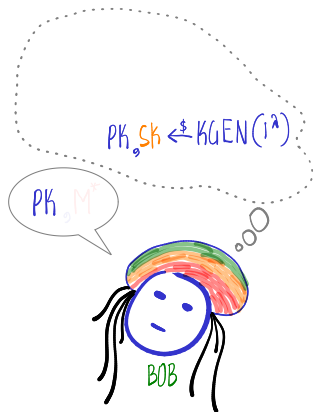
BOB

Definition 2

A scheme is EU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Existential Unforg. under Key-Only Attack

★★★★ $V \rightarrow]$

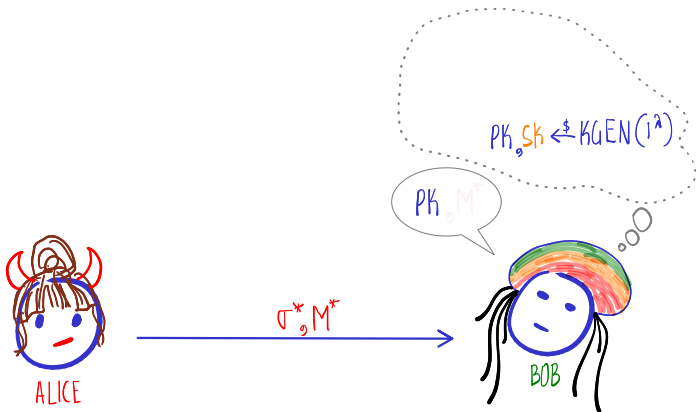


Definition 2

A scheme is EU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Existential Unforg. under Key-Only Attack

★★★★ $V \rightarrow]$



Definition 2

A scheme is EU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Existential Unforg. under Key-Only Attack

☆☆☆☆ $V \rightarrow \{$

BREAKS IF

$$\text{VERIFY}(\text{PK}, M^*, \sigma^*) = 1$$



ALICE

σ^*, M^*

PK, M^*



BOB

$\text{PK}, \text{SK} \leftarrow \text{KGEN}(1^\lambda)$

Definition 2

A scheme is EU-KOA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Existential Unforg. under Key-Only Attack

☆☆☆☆ $\forall \rightarrow \exists$

BREAKS IF

$$\text{VERIFY}(\text{PK}, M^*, \sigma^*) = 1$$



ALICE

σ^*, M^*

PK, M^*



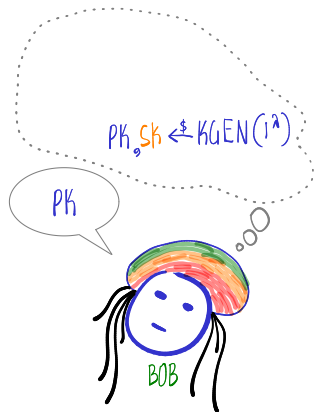
BOB

$$\text{PK}, \text{SK} \leftarrow \text{KGEN}(1^\lambda)$$

Definition 2

A scheme is EU-KOA secure if no PPT **forg** can **break** as above with a *non-negligible* probability.

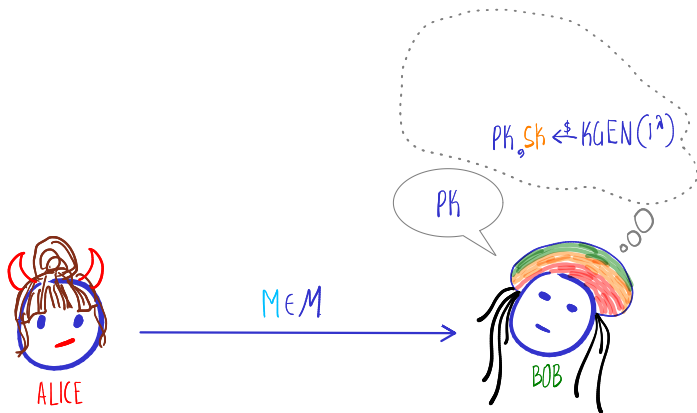
Security: Existential Unforg. under Chosen-Message Attack



Definition 3

A scheme is EU-CMA secure if no PPT forger can break as above with a *non-negligible* probability.

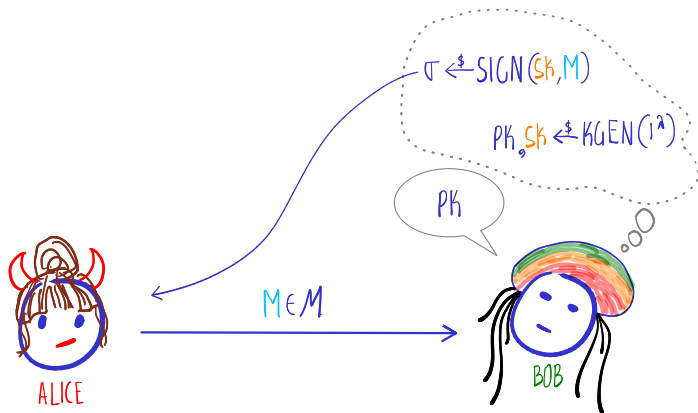
Security: Existential Unforg. under Chosen-Message Attack



Definition 3

A scheme is EU-CMA secure if no PPT forger can break as above with a *non-negligible* probability.

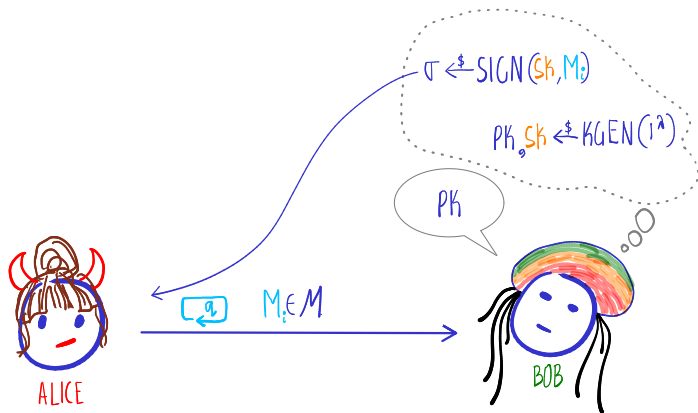
Security: Existential Unforg. under Chosen-Message Attack



Definition 3

A scheme is EU-CMA secure if no PPT forger can break as above with a *non-negligible* probability.

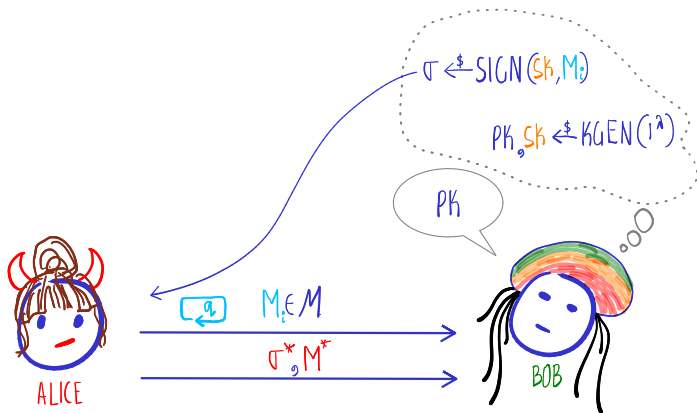
Security: Existential Unforg. under Chosen-Message Attack



Definition 3

A scheme is EU-CMA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Existential Unforg. under Chosen-Message Attack



Definition 3

A scheme is EU-CMA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Existential Unforg. under Chosen-Message Attack



BREAKS IF

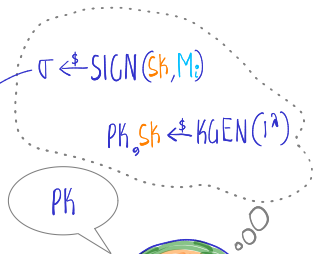
- ◆ $\text{VERIFY}(\text{PK}, M^*, \sigma^*) = 1$
- ◆ $\forall i \in [q]: M^* \neq M_i$



ALICE



BOB



Definition 3

A scheme is EU-CMA secure if no PPT forger can break as above with a *non-negligible* probability.

Security: Existential Unforg. under Chosen-Message Attack



BREAKS IF

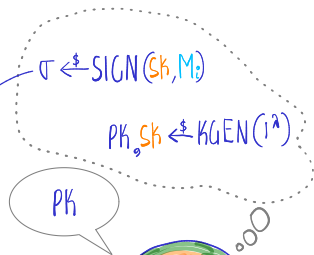
- ◆ $\text{VERIFY}(\text{PK}, M^*, \sigma^*) = 1$
- ◆ $\forall i \in [q]: M^* \neq M_i$



ALICE



BOB



Definition 3

A scheme is EU-CMA secure if no PPT forger can break as above with a *non-negligible* probability.

Plan for this Session

Digital Signature: Syntax and Modelling Security

One-Time Signatures

Many-Time (Stateful) Signatures

Efficient Signatures via Hash-and-Sign

Wrapping Up

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

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^\ell$

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

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^\ell$



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

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^{\ell=4}$

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

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

 = sk

GEN(1^λ)



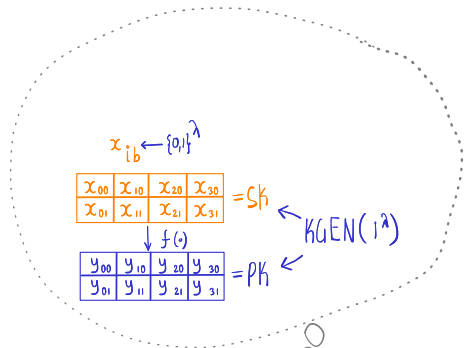
ALICE



BOB

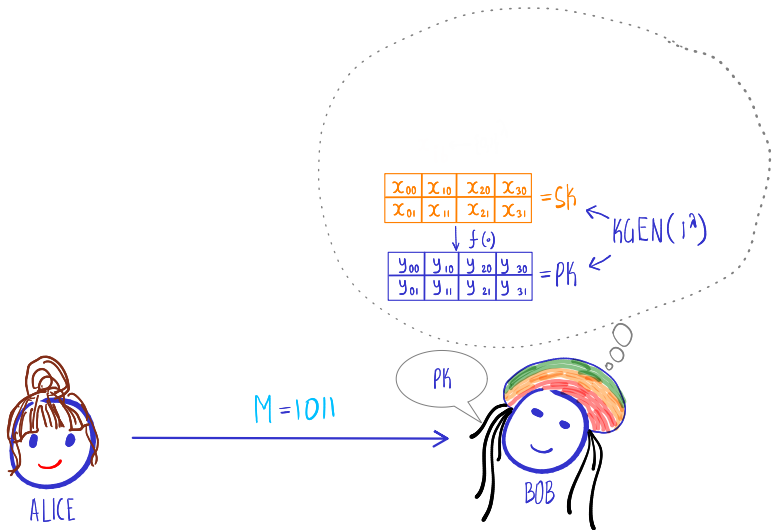
One-Time Signatures ($q = 1$): Lamport's Signature...

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^{\ell=4}$



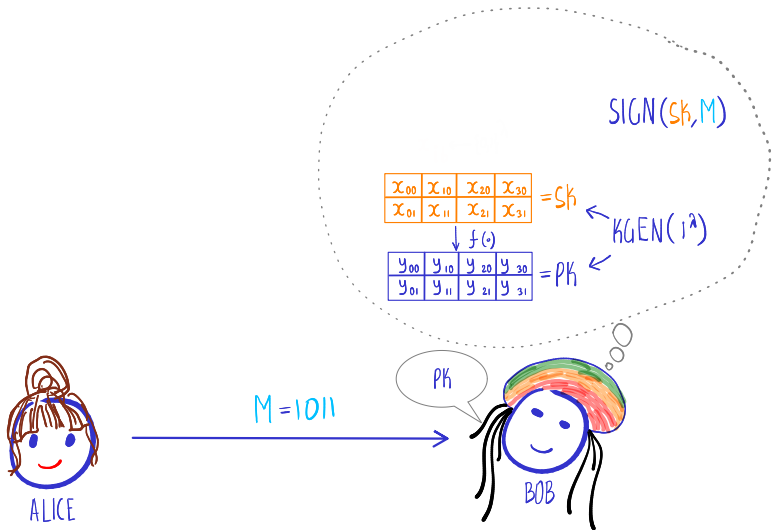
One-Time Signatures ($q = 1$): Lamport's Signature...

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^{\ell=4}$



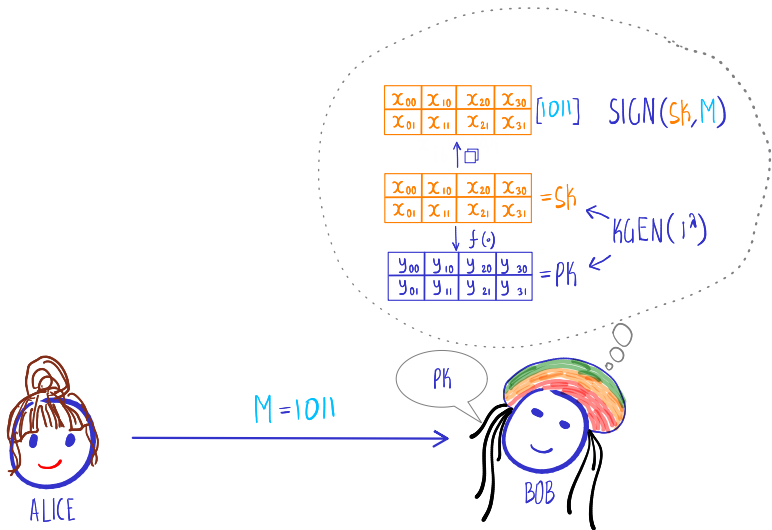
One-Time Signatures ($q = 1$): Lamport's Signature...

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^{\ell=4}$



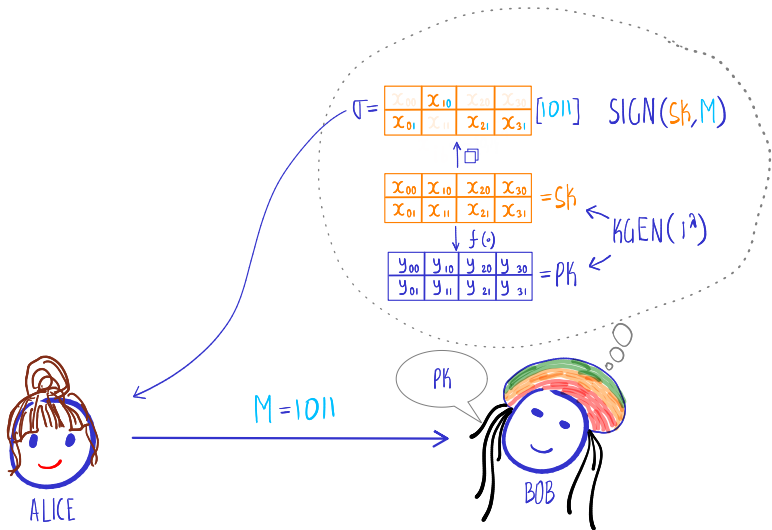
One-Time Signatures ($q = 1$): Lamport's Signature...

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^{\ell=4}$



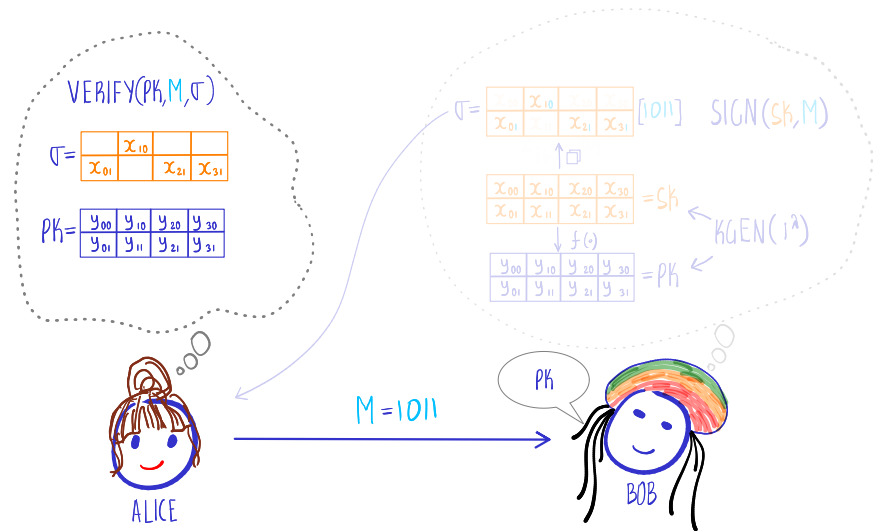
One-Time Signatures ($q = 1$): Lamport's Signature...

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^{\ell=4}$



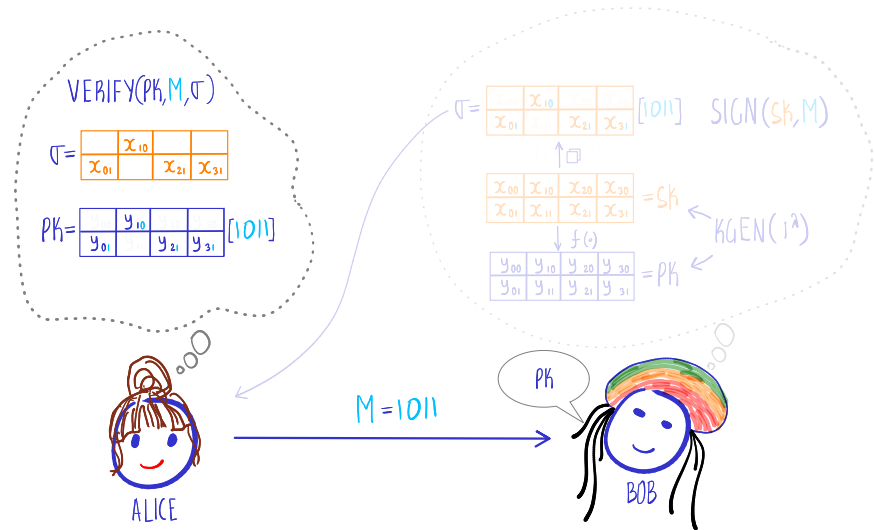
One-Time Signatures ($q = 1$): Lamport's Signature...

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^{\ell=4}$



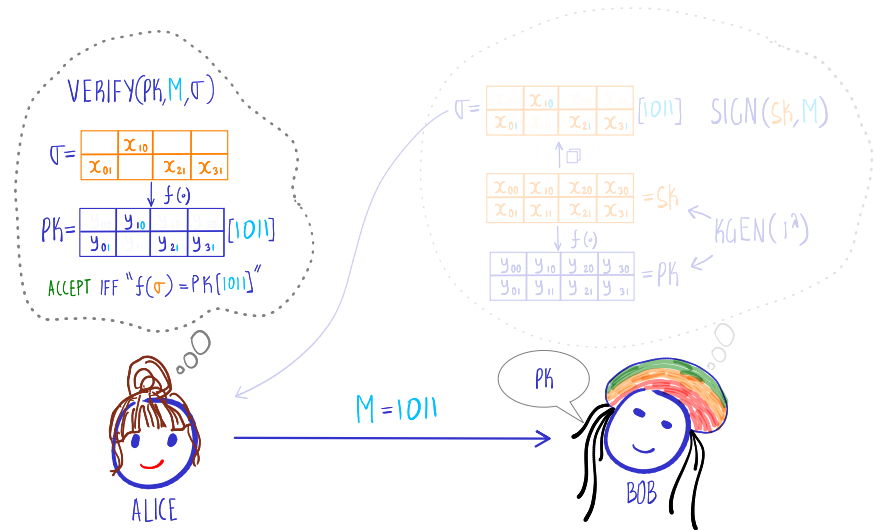
One-Time Signatures ($q = 1$): Lamport's Signature...

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^{\ell=4}$



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

- ▶ One-way func. $f : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0, 1\}^{\ell=4}$



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

Theorem 4

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

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

Theorem 4

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

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

ADVERSARY



REDUCTION



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

Theorem 4

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

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

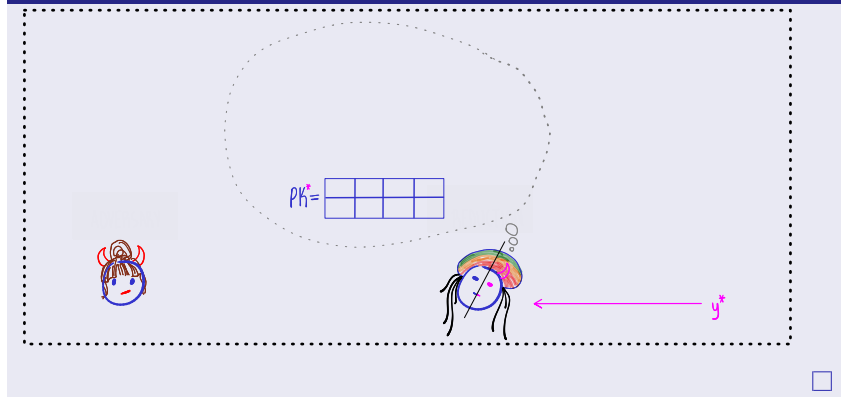


One-Time Signatures ($q = 1$): Lamport's Signature...

Theorem 4

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

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



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

Theorem 4

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

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



$$PK^* = \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & y^* & & \\ \hline & i^* \leftarrow [r] & & b^* \leftarrow \{0,1\} \\ \hline \end{array}$$



y^*



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

Theorem 4

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

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

The diagram illustrates the proof sketch for Lamport's signature scheme. It features a dashed box containing a character on the left and a character on the right. The right character has a pink arrow labeled y^* pointing to them. In the center, a dotted oval contains the following text:

$$x_{ib} \leftarrow \{0,1\}^\lambda$$
$$Sk = \begin{array}{|c|c|c|c|} \hline x_{00} & x_{10} & x_{20} & x_{30} \\ \hline x_{01} & & x_{21} & x_{31} \\ \hline \end{array}$$
$$PK^* = \begin{array}{|c|c|c|c|} \hline & y^* & & \\ \hline \end{array} \quad b^* \leftarrow \{0,1\}$$

A small square is in the bottom right corner.

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

Theorem 4

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

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

$$x_{ib} \leftarrow \{0,1\}^\lambda$$
$$sk = \begin{array}{|c|c|c|c|} \hline x_{00} & x_{10} & x_{20} & x_{30} \\ \hline x_{01} & & x_{21} & x_{31} \\ \hline \end{array}$$

$\downarrow f(\cdot)$

$$pk^* = \begin{array}{|c|c|c|c|} \hline y_{00} & y_{10} & y_{20} & y_{30} \\ \hline y_{01} & y^* & y_{21} & y_{31} \\ \hline \end{array} \quad b^* \leftarrow \{0,1\}$$

$i^* \leftarrow [t]$



y^*

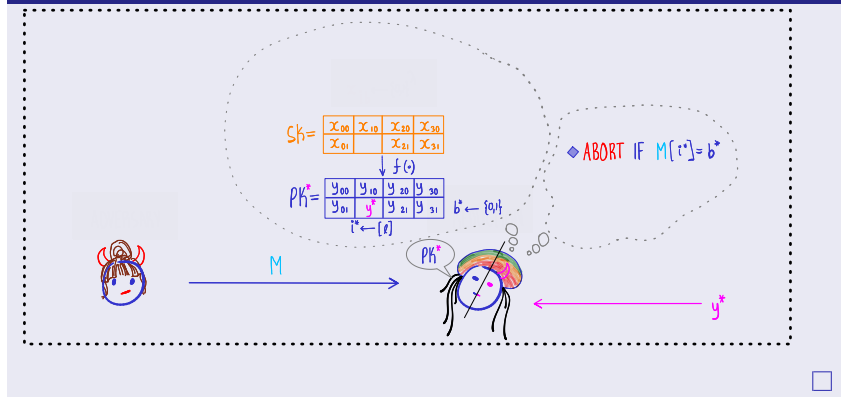


One-Time Signatures ($q = 1$): Lamport's Signature...

Theorem 4

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

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

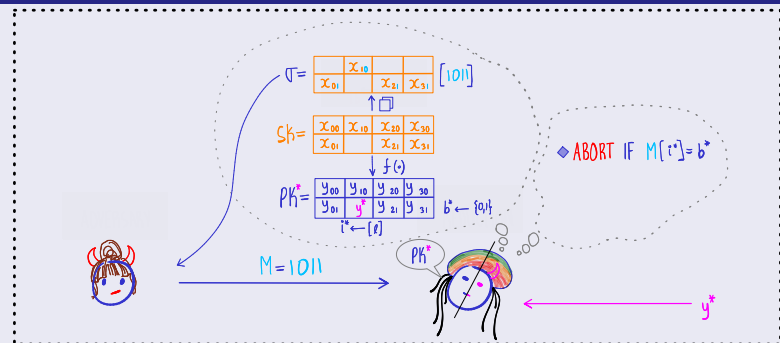


One-Time Signatures ($q = 1$): Lamport's Signature...

Theorem 4

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

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

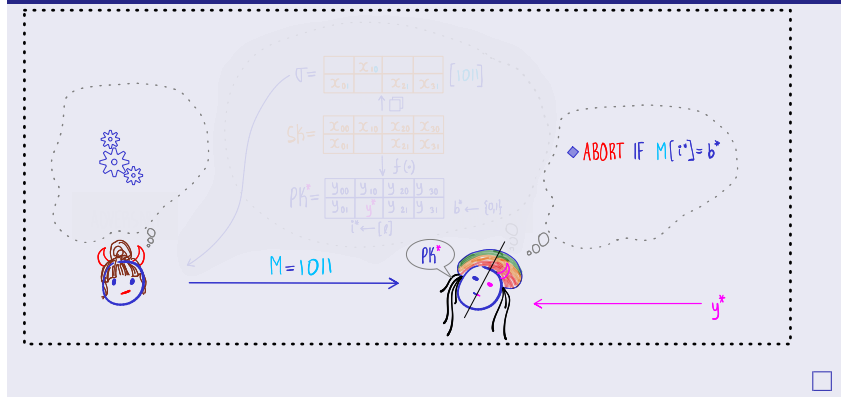


One-Time Signatures ($q = 1$): Lamport's Signature...

Theorem 4

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

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

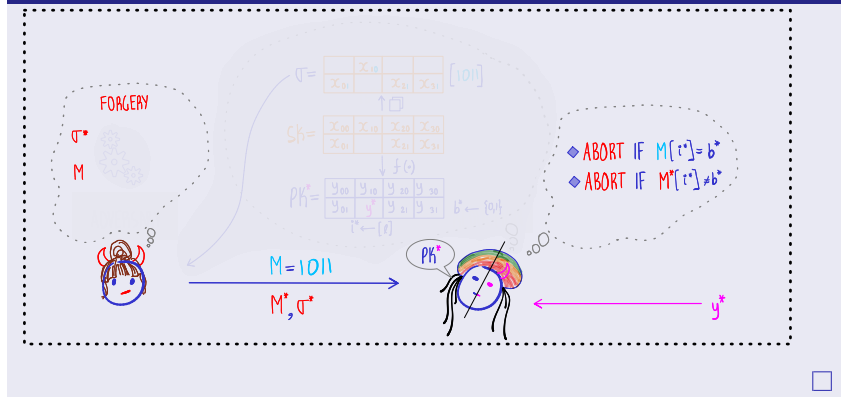


One-Time Signatures ($q = 1$): Lamport's Signature...

Theorem 4

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

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

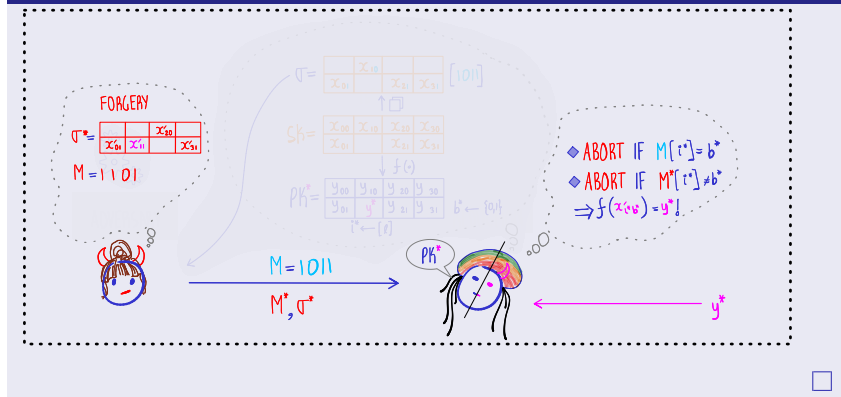


One-Time Signatures ($q = 1$): Lamport's Signature...

Theorem 4

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

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

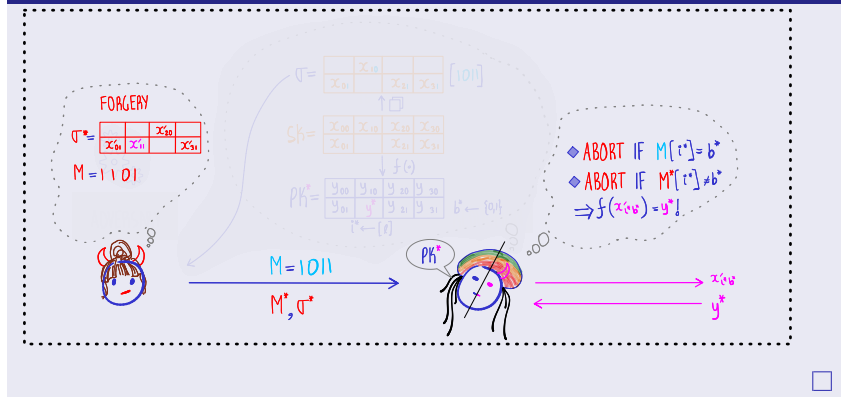


One-Time Signatures ($q = 1$): Lamport's Signature...

Theorem 4

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

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

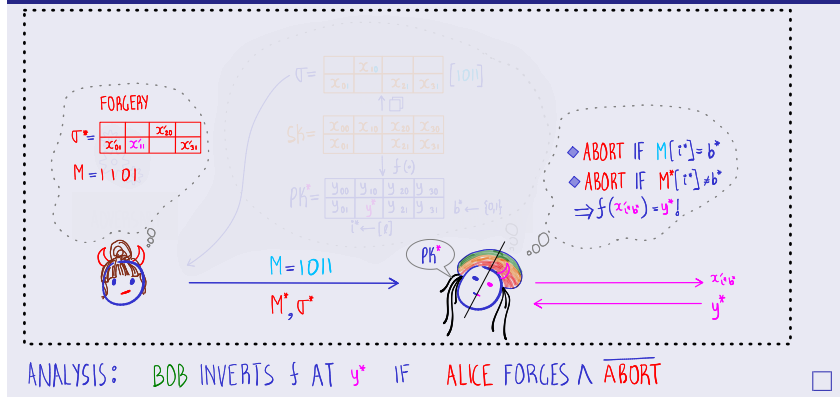


One-Time Signatures ($q = 1$): Lamport's Signature...

Theorem 4

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

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

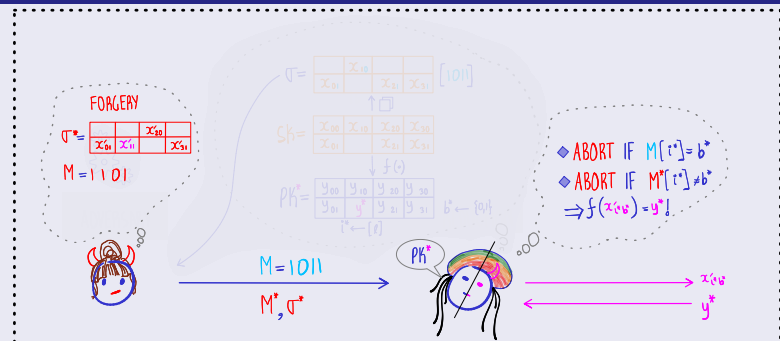


One-Time Signatures ($q = 1$): Lamport's Signature...

Theorem 4

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

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



ANALYSIS: $\Pr[\text{BOB INVERTS } f \text{ AT } y^*] = \Pr[\text{ALICE FORGES } \wedge \overline{\text{ABORT}}]$

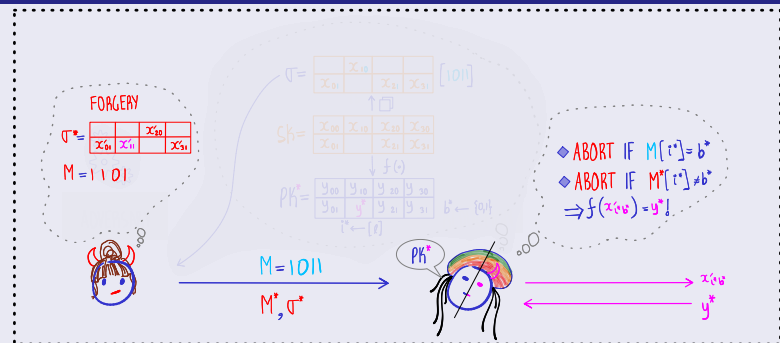


One-Time Signatures ($q = 1$): Lamport's Signature...

Theorem 4

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

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



ANALYSIS: $\Pr[\text{BOB INVERTS } f \text{ AT } y^*] \geq \Pr[\text{ALICE FORGES} \mid \overline{\text{ABORT}}] \cdot \frac{1}{2}$



One-Time Signatures: Lamport's Signature...

Exercise 1

- ▶ *Can a forger break EU-CMA given **two** signatures?*
- ▶ *What happens if we **fix** $i^* = 0$ in the proof?*
- ▶ *Are the signatures **unique**? If not, can it be made unique?*

One-Time Signatures: Lamport's Signature...

Exercise 1

- ▶ Can a forger break EU-CMA given *two* signatures?
- ▶ What happens if we *fix* $i^* = 0$ in the proof?
- ▶ Are the signatures *unique*? If not, can it be made unique?

Theorem 5

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

One-Time Signatures: Lamport's Signature...

Exercise 1

- ▶ Can a forger break EU-CMA given *two* signatures?
- ▶ What happens if we *fix* $i^* = 0$ in the proof?
- ▶ Are the signatures *unique*? If not, can it be made unique?

Theorem 5

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

Exercise 2 (Domain Extension)

Given a collision-resistant hash function $H : \{0, 1\}^{2^\ell} \rightarrow \{0, 1\}^\ell$, construct a OTS for *arbitrary-length* messages.

Plan for this Session

Digital Signature: Syntax and Modelling Security

One-Time Signatures

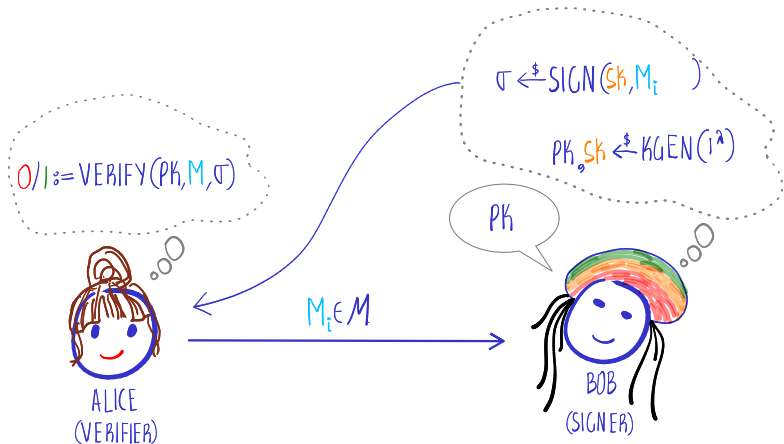
Many-Time (Stateful) Signatures

Efficient Signatures via Hash-and-Sign

Wrapping Up

(Many-Time) Signatures with Stateful Signer

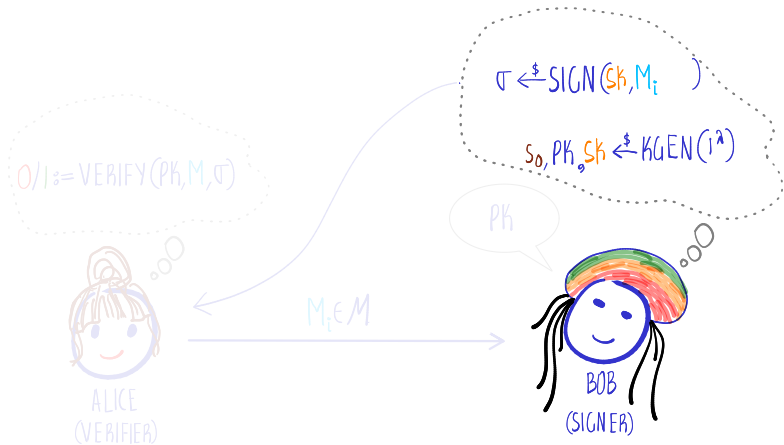
- ▶ Syntax: same as before except that SIGN is *stateful*



- ▶ Security: same as before (i.e., forger *not* given any state)

(Many-Time) Signatures with Stateful Signer

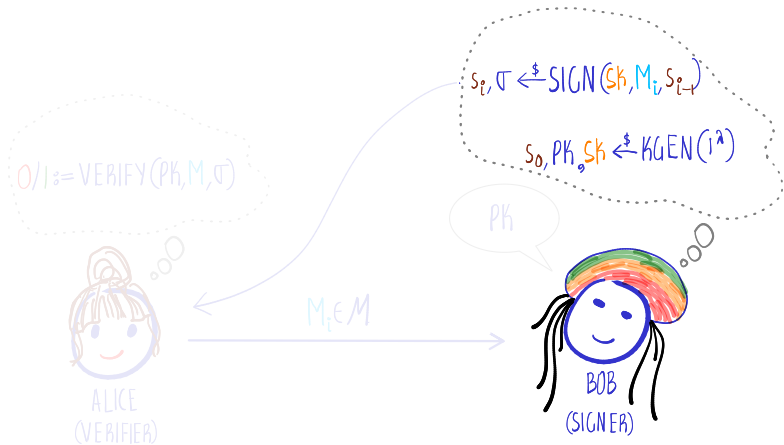
- ▶ Syntax: same as before except that SIGN is *stateful*



- ▶ Security: same as before (i.e., forger *not* given any state)

(Many-Time) Signatures with Stateful Signer

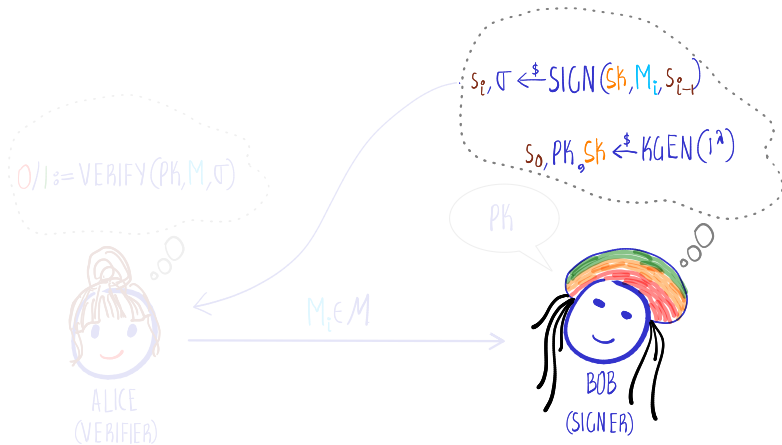
- ▶ Syntax: same as before except that SIGN is *stateful*



- ▶ Security: same as before (i.e., forger *not* given any state)

(Many-Time) Signatures with Stateful Signer

- ▶ Syntax: same as before except that SIGN is *stateful*



- ▶ Security: same as before (i.e., *forgery not* given any state)

(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

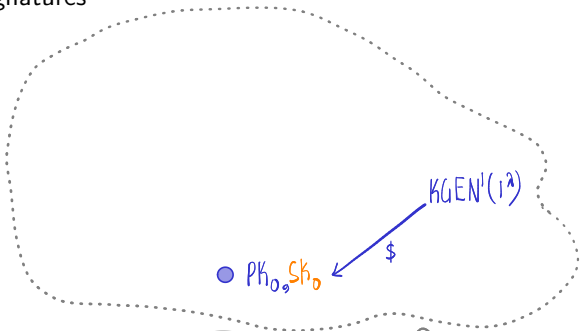
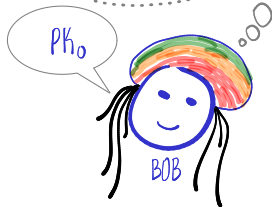
(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .
 - 💡 Idea: “chain signatures”

(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

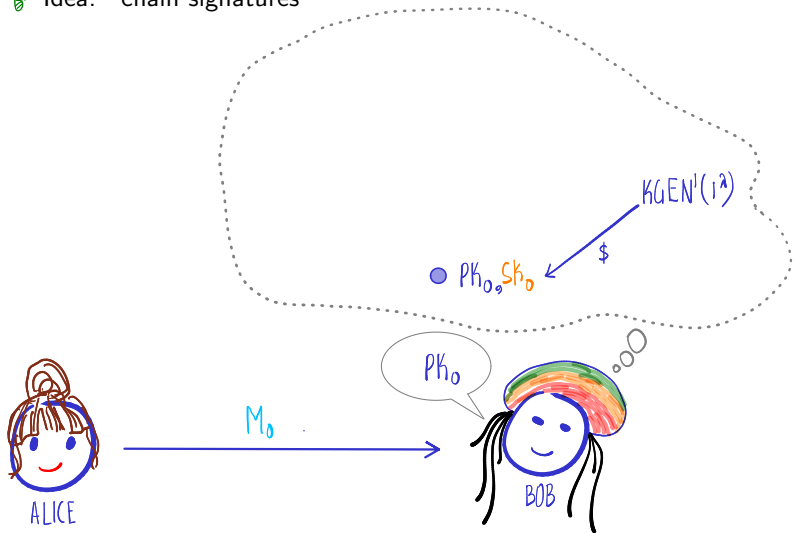
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

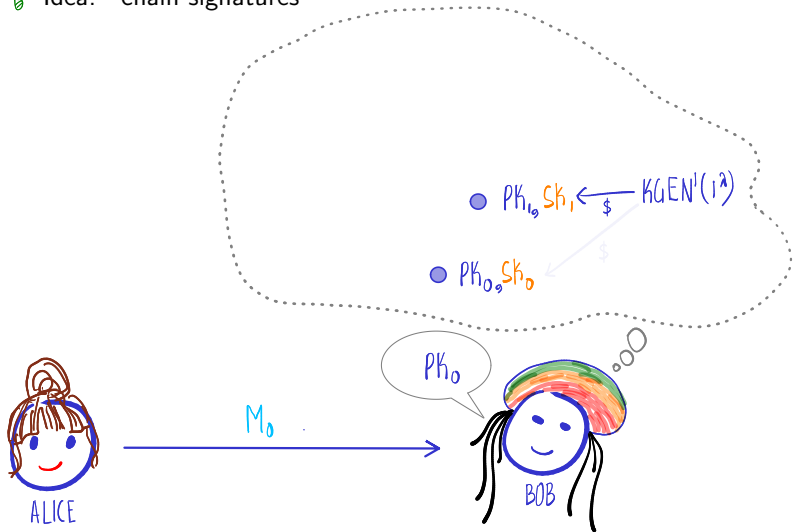
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

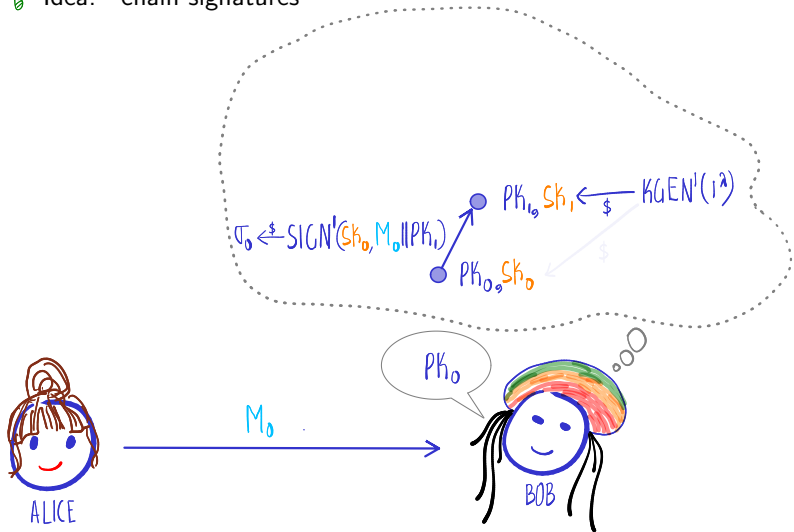
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

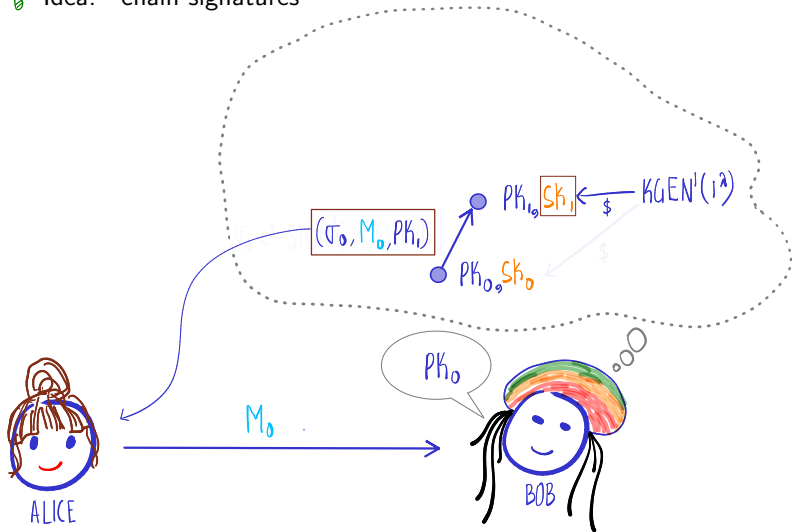
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

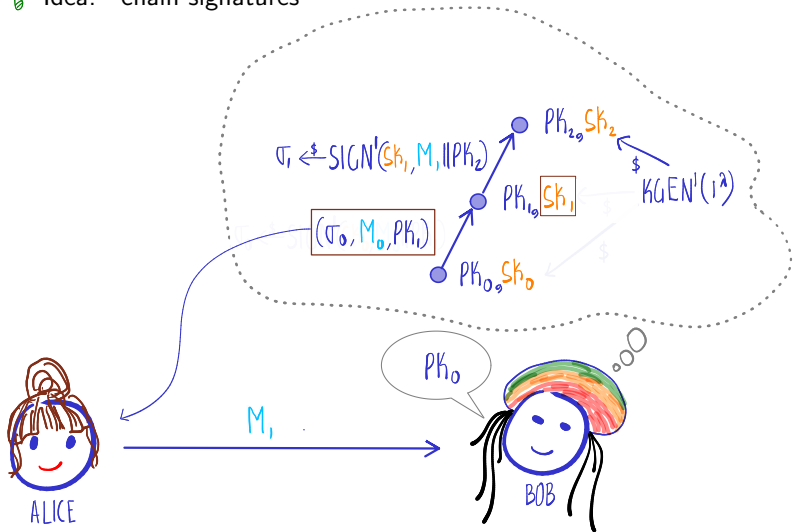
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

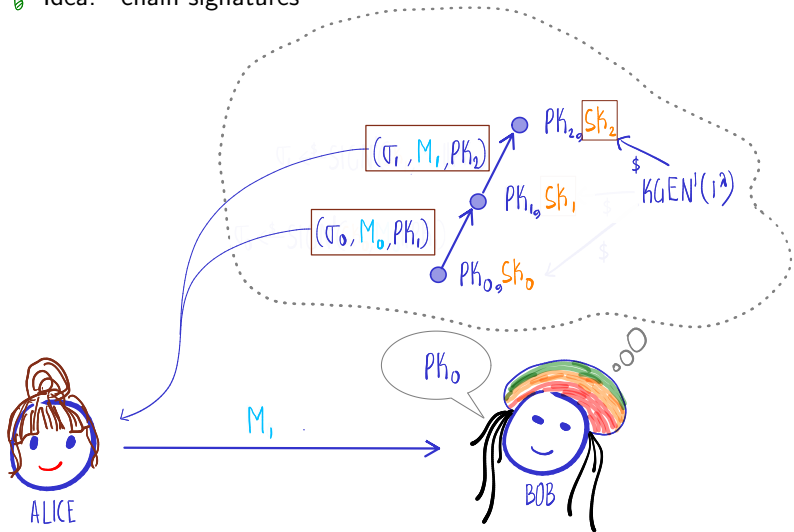
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

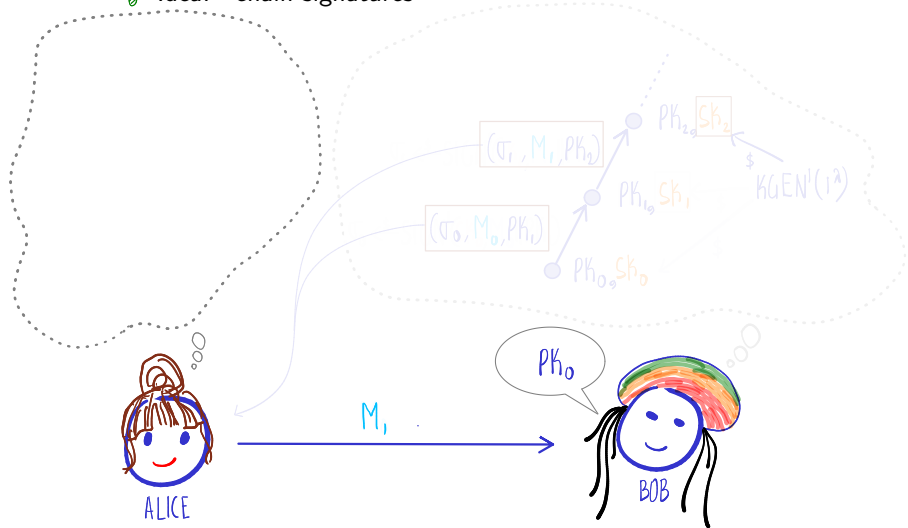
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ **stateful** signature Σ^s .

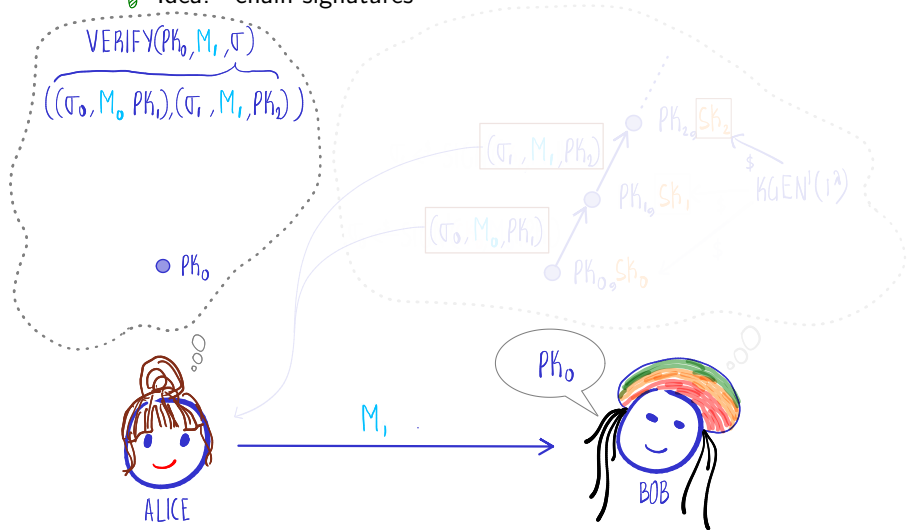
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

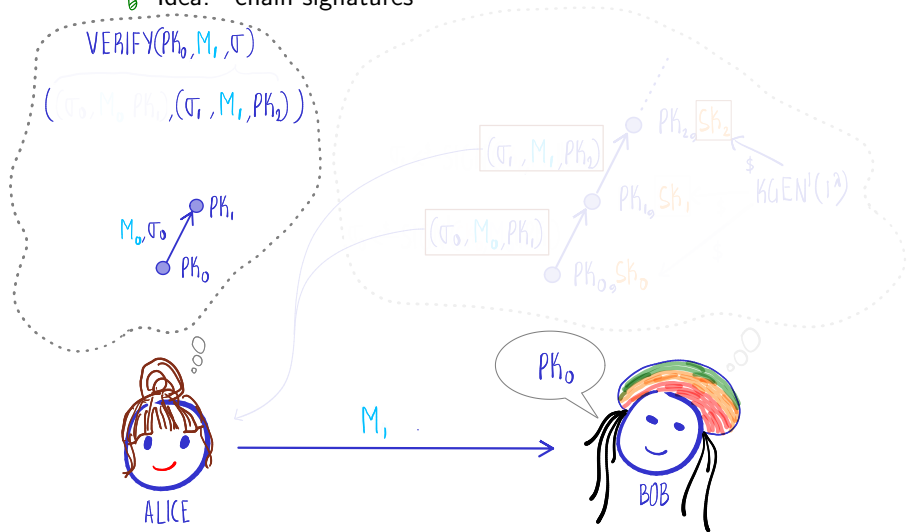
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

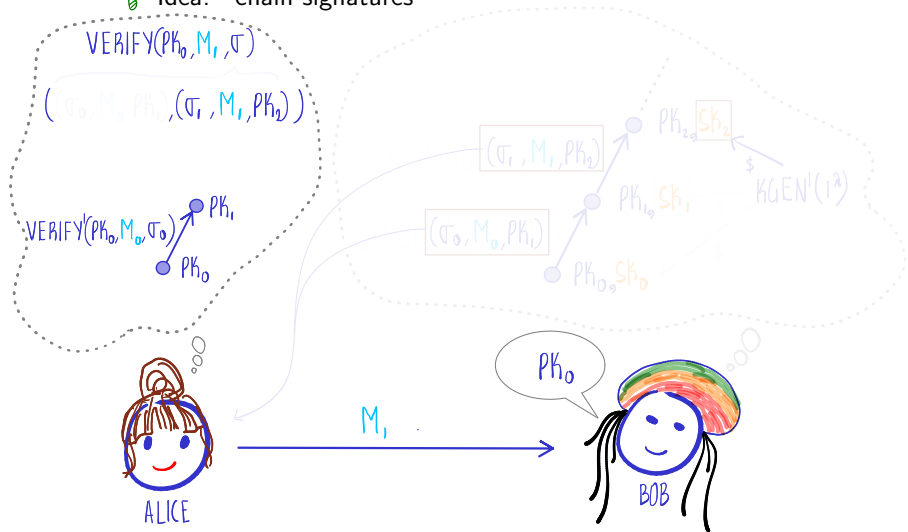
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

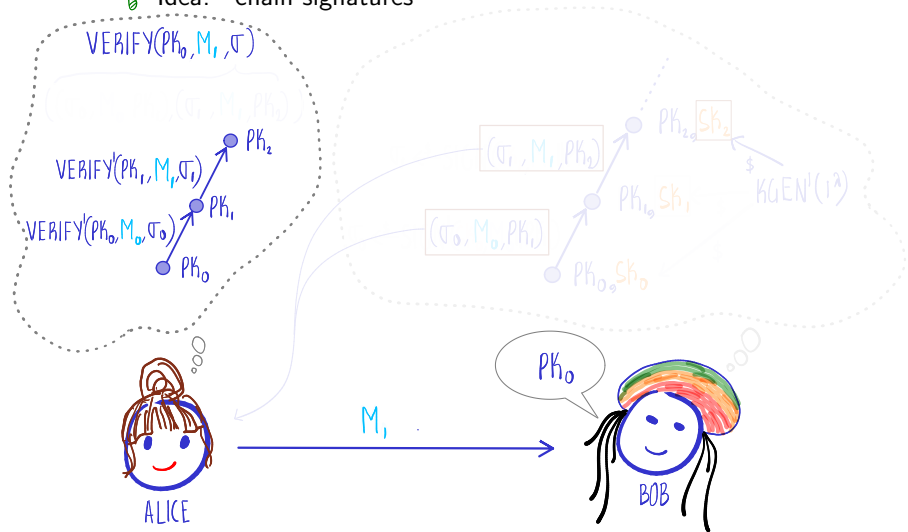
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

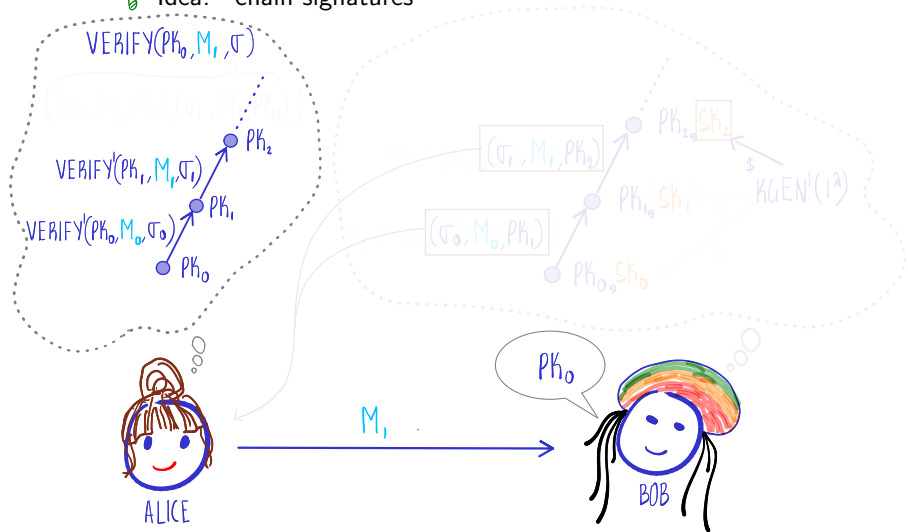
💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- ▶ OTS $\Sigma^1 = (\text{KGEN}^1, \text{SIGN}^1, \text{VERIFY}^1) \Rightarrow$ stateful signature Σ^s .

💡 Idea: "chain signatures"



(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

ADVERSARY



REDUCTION

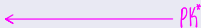


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

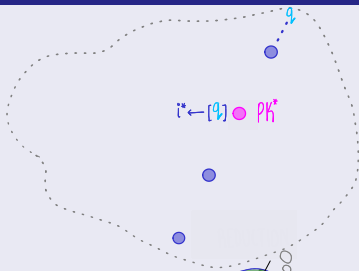


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.



$i^* \leftarrow [q] \bullet pk^*$



← pk^*

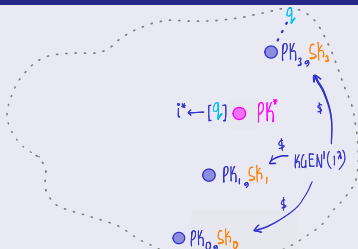


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.



← PK^*

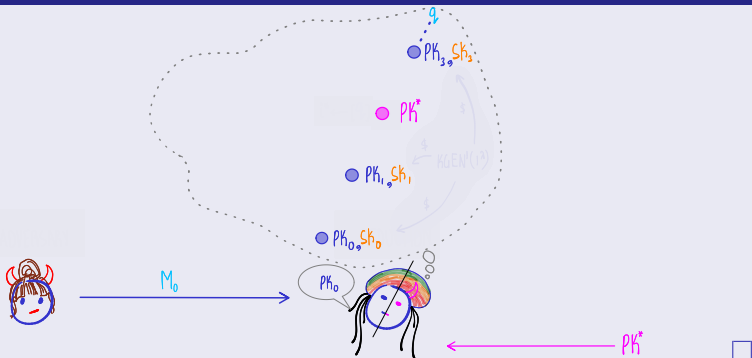


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

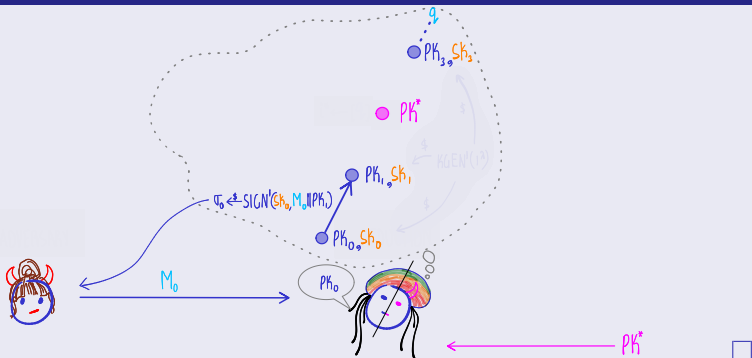


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

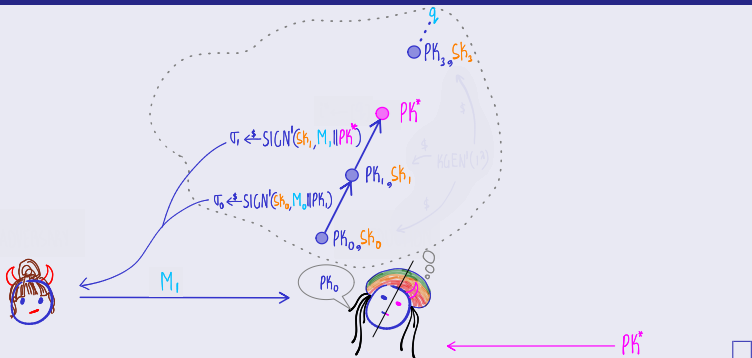


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

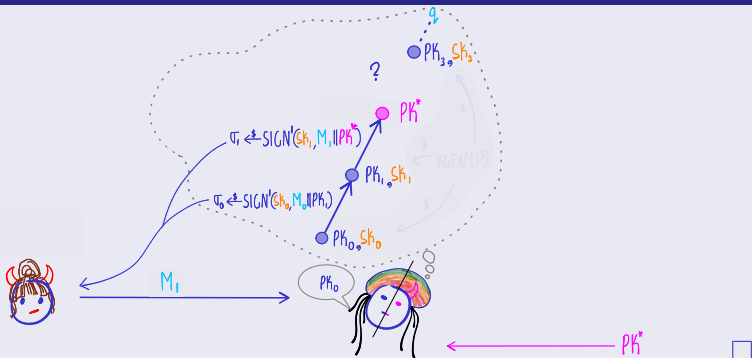


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

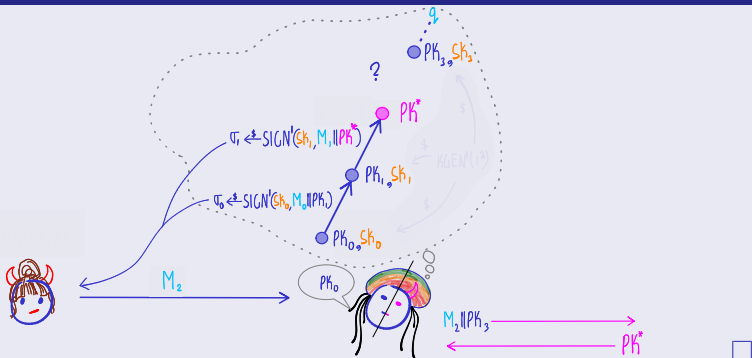


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

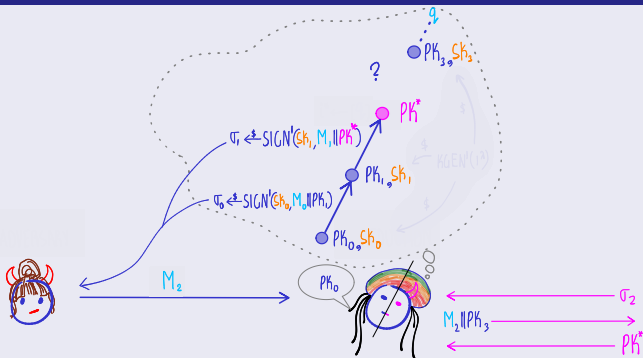


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

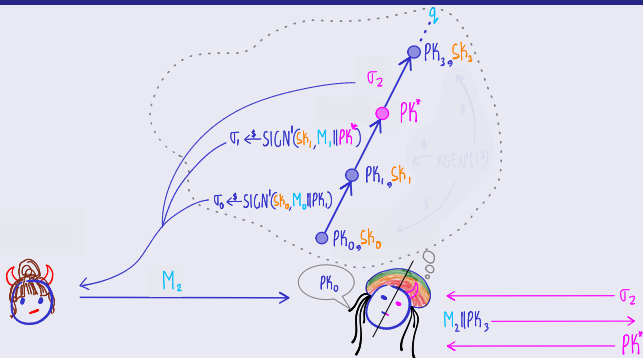


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

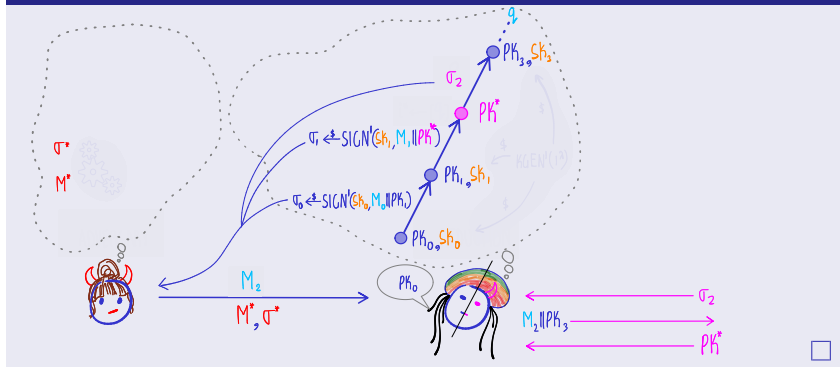


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

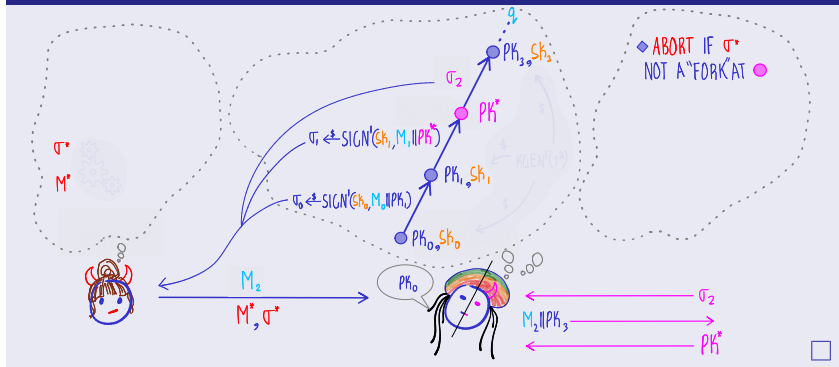


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

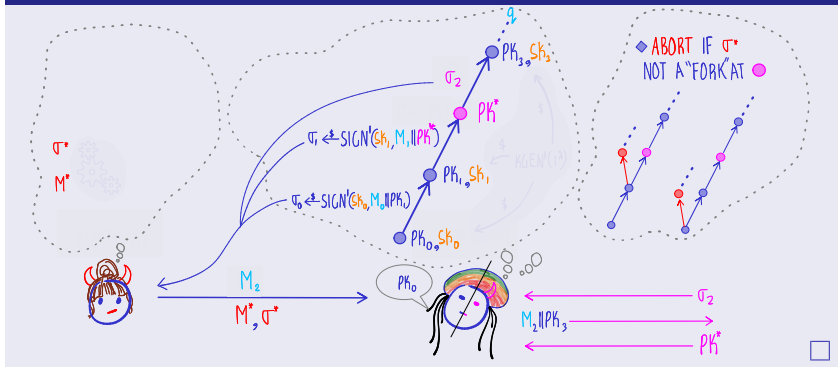


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

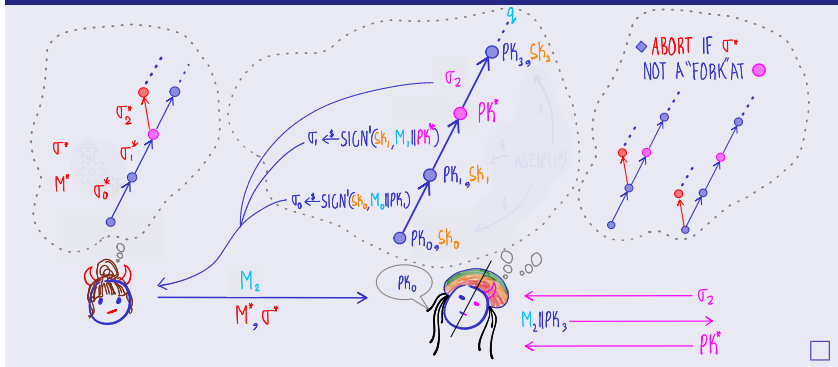


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.

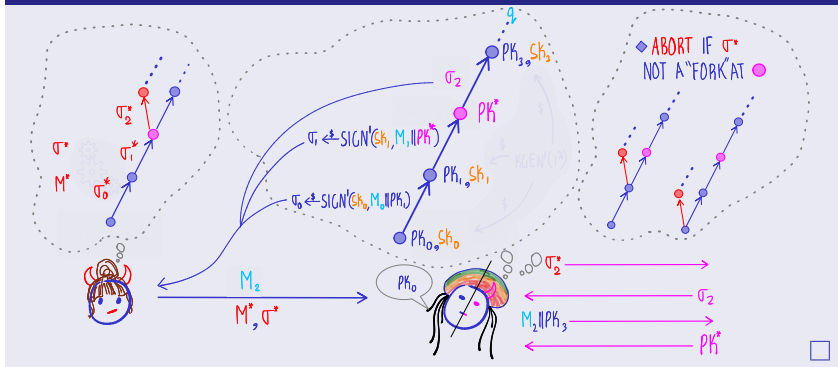


(Many-Time) Stateful Signatures...

Theorem 6

If Σ^1 is an OTS supporting arbitrary-length messages then Σ^s is a stateful signature.

Proof sketch: plug and pray, again.





(Many-Time) Stateful Signatures...



The size of signatures in Σ^s grows **linearly** with the number of signatures issued by the signer. How to fix this?

(Many-Time) Stateful Signatures...

 The size of signatures in Σ^s grows **linearly** with the number of signatures issued by the signer. How to fix this?

 Idea: “tree of signatures”

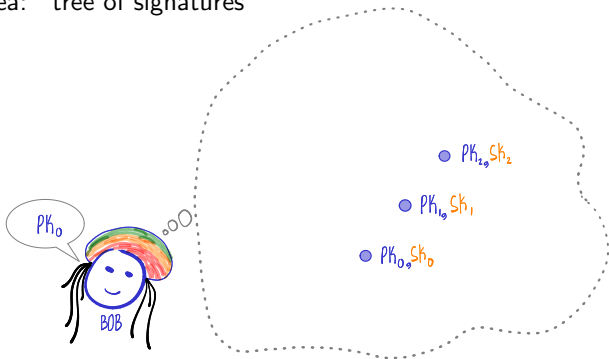
Exercise 3 (Shorter stateful signature)

Prove that the construction Σ^{ss} is secure. (Hint: plug and pray.)

(Many-Time) Stateful Signatures...

👁️ The size of signatures in Σ^s grows **linearly** with the number of signatures issued by the signer. How to fix this?

💡 Idea: “tree of signatures”



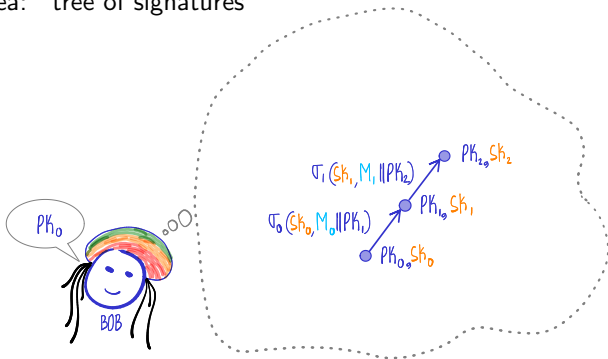
Exercise 3 (Shorter stateful signature)

Prove that the construction Σ^{ss} is secure. (Hint: plug and pray.)

(Many-Time) Stateful Signatures...

👁️ The size of signatures in Σ^S grows **linearly** with the number of signatures issued by the signer. How to fix this?

💡 Idea: “tree of signatures”



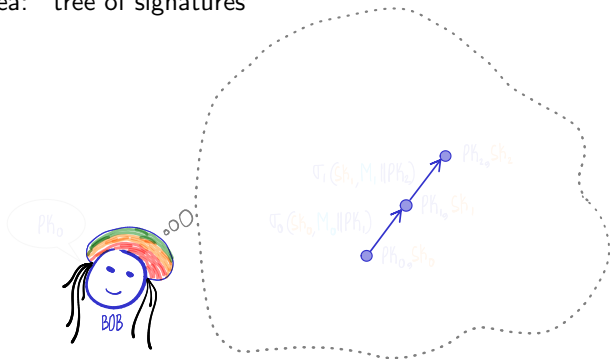
Exercise 3 (Shorter stateful signature)

Prove that the construction Σ^{SS} is secure. (Hint: plug and pray.)

(Many-Time) Stateful Signatures...

👁️ The size of signatures in Σ^s grows **linearly** with the number of signatures issued by the signer. How to fix this?

💡 Idea: "tree of signatures"



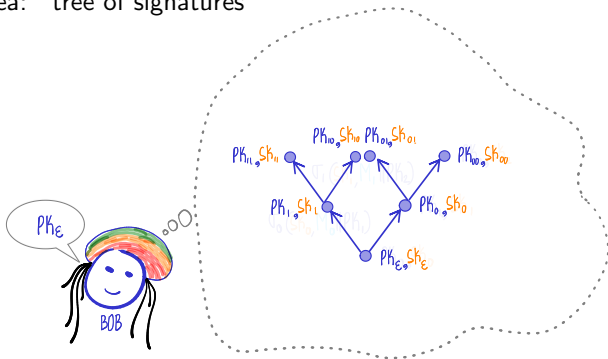
Exercise 2 (Shorter stateful signatures)

Prove that the construction Σ^s is secure. (Hint: plug and pray.)

(Many-Time) Stateful Signatures...

👁️ The size of signatures in Σ^s grows **linearly** with the number of signatures issued by the signer. How to fix this?

💡 Idea: “tree of signatures”



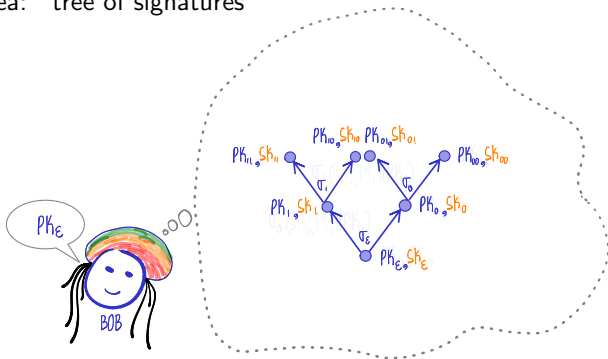
Exercise 5 (Shorter stateful signature)

Prove that the construction Σ^s is secure. (Hint: plug and pray.)

(Many-Time) Stateful Signatures...

👁️ The size of signatures in Σ^s grows **linearly** with the number of signatures issued by the signer. How to fix this?

💡 Idea: “tree of signatures”



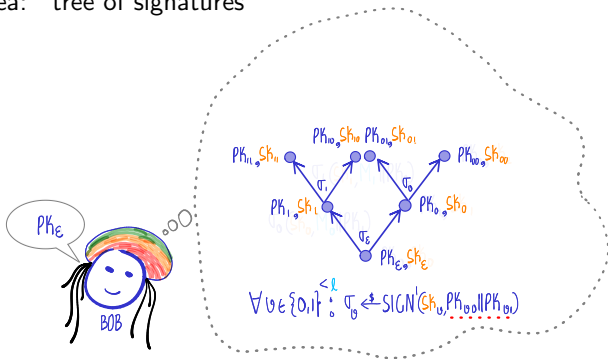
Exercise 3 (Shorter stateful signature)

Prove that the construction Σ^s is secure. (Hint: plug and pray.)

(Many-Time) Stateful Signatures...

👁️ The size of signatures in Σ^S grows **linearly** with the number of signatures issued by the signer. How to fix this?

💡 Idea: "tree of signatures"



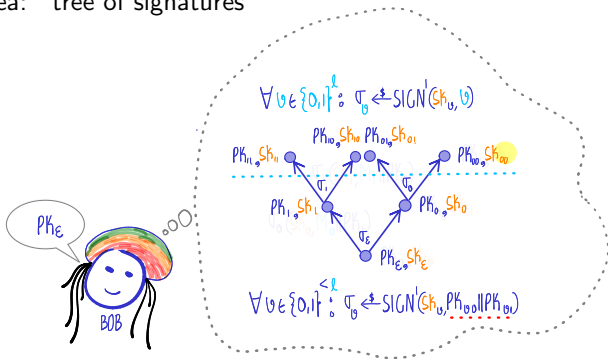
Exercise 5 (Shorter stateful signature)

Prove that the construction Σ^S is secure. (Hint: plug and pray.)

(Many-Time) Stateful Signatures...

👁️ The size of signatures in Σ^S grows **linearly** with the number of signatures issued by the signer. How to fix this?

💡 Idea: "tree of signatures"



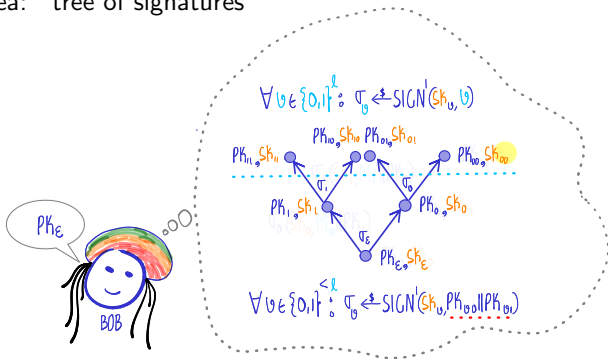
Exercise 5 (Shorter stateful signature)

Prove that the construction Σ^S is secure. (Hint: plug and pray.)

(Many-Time) Stateful Signatures...

👁️ The size of signatures in Σ^s grows **linearly** with the number of signatures issued by the signer. How to fix this?

💡 Idea: "tree of signatures"




Exercise 3 (Shorter stateful signature)

Prove that the construction Σ^{ss} is secure. (Hint: plug and pray.)

(Many-Time) Signatures...

- ▶ Short stateful signature Σ^{ss} + pseudo-random function
 $F_K : \{0, 1\}^{\ell+1} \rightarrow \{0, 1\}^\lambda \Rightarrow \text{signature } \Sigma$

(Many-Time) Signatures...

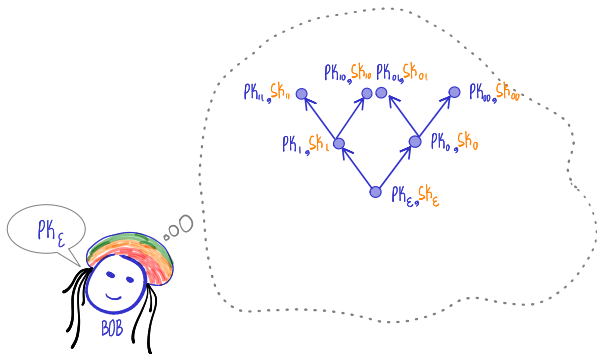
- ▶ Short stateful signature Σ^{ss} + pseudo-random function
 $F_K : \{0, 1\}^{\ell+1} \rightarrow \{0, 1\}^\lambda \Rightarrow$ signature Σ
 Idea: Use to *Derandomise* OTS signature and key gen.

Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

- ▶ Short stateful signature Σ^{SS} + pseudo-random function $F_K : \{0, 1\}^{\ell+1} \rightarrow \{0, 1\}^\lambda \Rightarrow$ signature Σ
 - 💡 Idea: Use to *Derandomise* OTS signature and key gen.

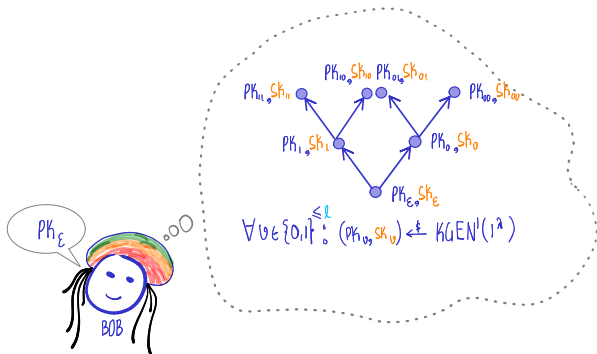


Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

- ▶ Short stateful signature Σ^{SS} + pseudo-random function $F_K : \{0, 1\}^{\ell+1} \rightarrow \{0, 1\}^\lambda \Rightarrow$ signature Σ
 - 💡 Idea: Use to *Derandomise* OTS signature and key gen.

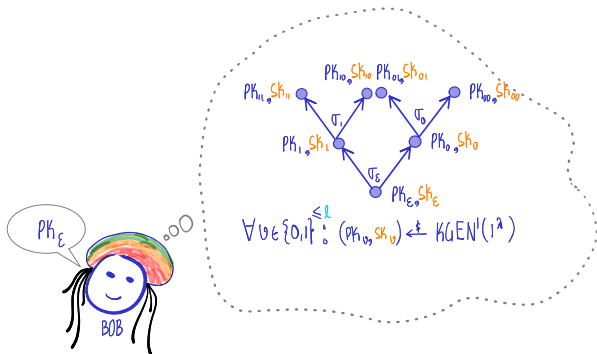


Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

- ▶ Short stateful signature Σ^{SS} + pseudo-random function $F_K : \{0, 1\}^{\ell+1} \rightarrow \{0, 1\}^\lambda \Rightarrow$ signature Σ
 - 💡 Idea: Use to *Derandomise* OTS signature and key gen.

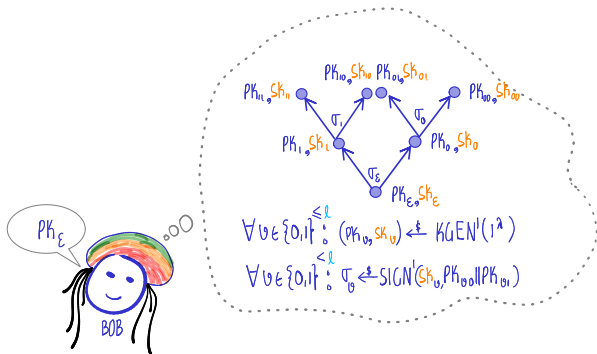


Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

- ▶ Short stateful signature Σ^{SS} + pseudo-random function $F_K : \{0, 1\}^{\ell+1} \rightarrow \{0, 1\}^\lambda \Rightarrow$ signature Σ
- 💡 Idea: Use to *Derandomise* OTS signature and key gen.

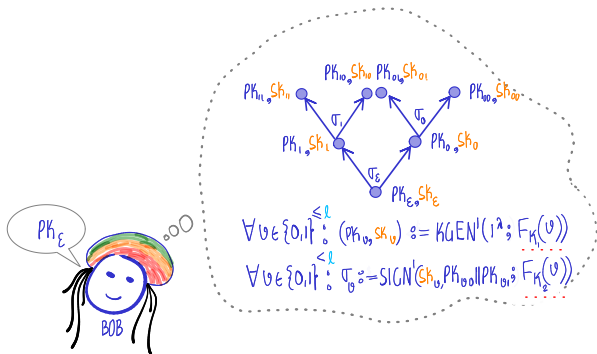


Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

- ▶ Short stateful signature Σ^{SS} + pseudo-random function $F_K : \{0, 1\}^{\ell+1} \rightarrow \{0, 1\}^\lambda \Rightarrow$ signature Σ
- 💡 Idea: Use to *Derandomise* OTS signature and key gen.

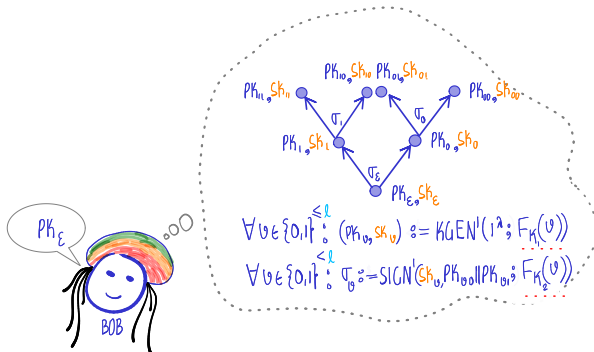


Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

- ▶ Short stateful signature Σ^{SS} + pseudo-random function $F_K : \{0, 1\}^{\ell+1} \rightarrow \{0, 1\}^\lambda \Rightarrow$ signature Σ
- 💡 Idea: Use to *Derandomise* OTS signature and key gen.



Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

Plan for this Session

Digital Signature: Syntax and Modelling Security

One-Time Signatures

Many-Time (Stateful) Signatures

Efficient Signatures via Hash-and-Sign

Wrapping Up

Efficient Signatures via Hash-then-Invert...

- ▶ Efficient schemes under stronger hardness assumptions

Efficient Signatures via Hash-then-Invert...

- ▶ Efficient schemes under stronger hardness assumptions
- ▶ Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \rightarrow \mathcal{D}$

Efficient Signatures via Hash-then-Invert...

- ▶ Efficient schemes under stronger hardness assumptions
- ▶ Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \rightarrow \mathcal{D}$
 - ▶ Syntax:

- ▶ Security: one-way without the knowledge of the trapdoor

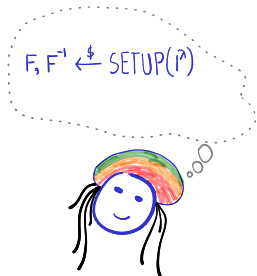
- ▶ Instantiations of TDP

- ▶ RSA perm.: $F(x) := x^e \bmod N$ and $F^{-1}(y) := y^d \bmod N$,
where $ed = 1 \bmod \phi(N)$

- ▶ From indistinguishability obfuscation and OWF

Efficient Signatures via Hash-then-Invert...

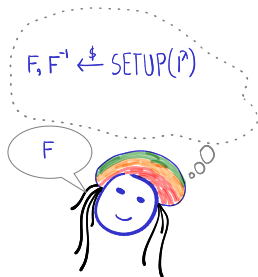
- ▶ Efficient schemes under stronger hardness assumptions
- ▶ Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \rightarrow \mathcal{D}$
 - ▶ Syntax:



- ▶ Security: one-way without the knowledge of the trapdoor
- ▶ Instantiations of TDP
 - ▶ RSA perm.: $F(x) := x^e \bmod N$ and $F^{-1}(y) := y^d \bmod N$, where $ed = 1 \bmod \phi(N)$
 - ▶ From indistinguishability obfuscation and OWF

Efficient Signatures via Hash-then-Invert...

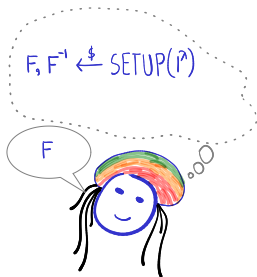
- ▶ Efficient schemes under stronger hardness assumptions
- ▶ Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \rightarrow \mathcal{D}$
 - ▶ Syntax:



- ▶ Security: one-way without the knowledge of the trapdoor
- ▶ Instantiations of TDP
 - ▶ RSA perm.: $F(x) := x^e \bmod N$ and $F^{-1}(y) := y^d \bmod N$, where $ed = 1 \bmod \phi(N)$
 - ▶ From indistinguishability obfuscation and OWF

Efficient Signatures via Hash-then-Invert...

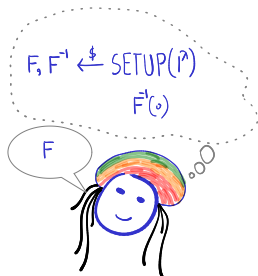
- ▶ Efficient schemes under stronger hardness assumptions
- ▶ Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \rightarrow \mathcal{D}$
 - ▶ Syntax:



- ▶ Security: one-way without the knowledge of the trapdoor
- ▶ Instantiations of TDP
 - ▶ RSA perm.: $F(x) := x^e \bmod N$ and $F^{-1}(y) := y^d \bmod N$, where $ed = 1 \bmod \phi(N)$
 - ▶ From indistinguishability obfuscation and OWF

Efficient Signatures via Hash-then-Invert...

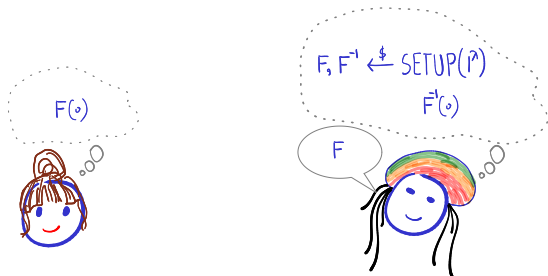
- ▶ Efficient schemes under stronger hardness assumptions
- ▶ Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \rightarrow \mathcal{D}$
 - ▶ Syntax:



- ▶ Security: one-way without the knowledge of the trapdoor
- ▶ Instantiations of TDP
 - ▶ RSA perm.: $F(x) := x^e \bmod N$ and $F^{-1}(y) := y^d \bmod N$, where $ed = 1 \bmod \phi(N)$
 - ▶ From indistinguishability obfuscation and OWF

Efficient Signatures via Hash-then-Invert...

- ▶ Efficient schemes under stronger hardness assumptions
- ▶ Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \rightarrow \mathcal{D}$
 - ▶ Syntax:



- ▶ Security: *one-way* without the knowledge of the trapdoor
- ▶ Instantiations of TDP
 - ▶ RSA perm.: $F(x) := x^e \bmod N$ and $F^{-1}(y) := y^d \bmod N$, where $ed = 1 \bmod \phi(N)$
 - ▶ From indistinguishability obfuscation and OWF

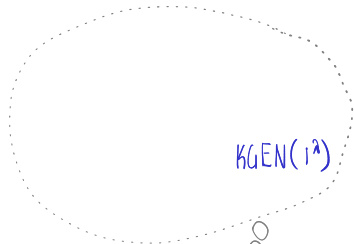
Efficient Signatures via Hash-then-Invert...

- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function
 $H: \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$

- ▶ Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

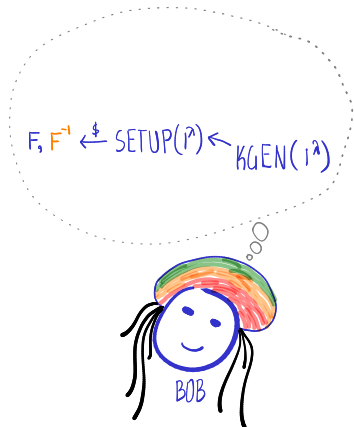
- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function $H: \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$



- ▶ Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

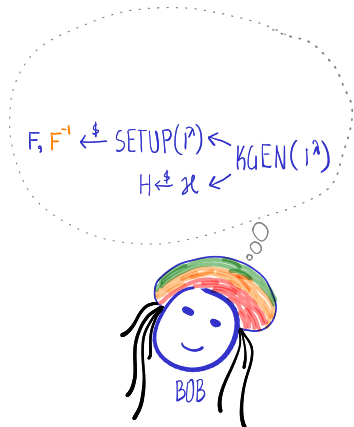
- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function $H: \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$



- ▶ Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

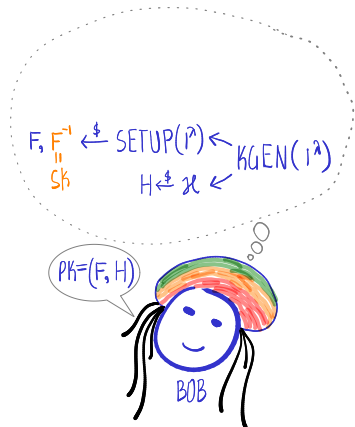
- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function
 $H: \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$



- ▶ Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

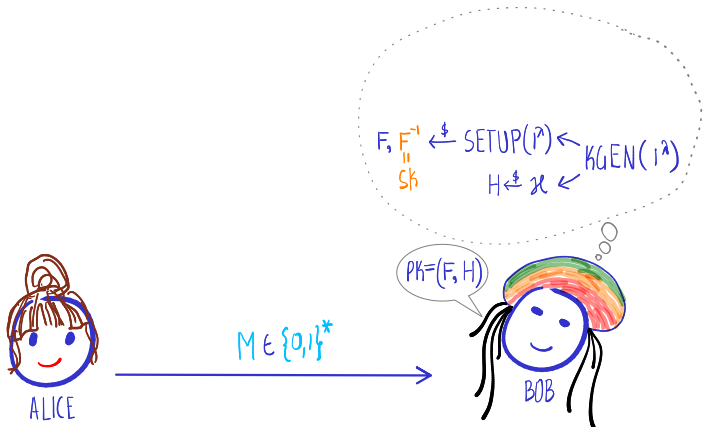
- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function
 $H: \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$



- ▶ Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

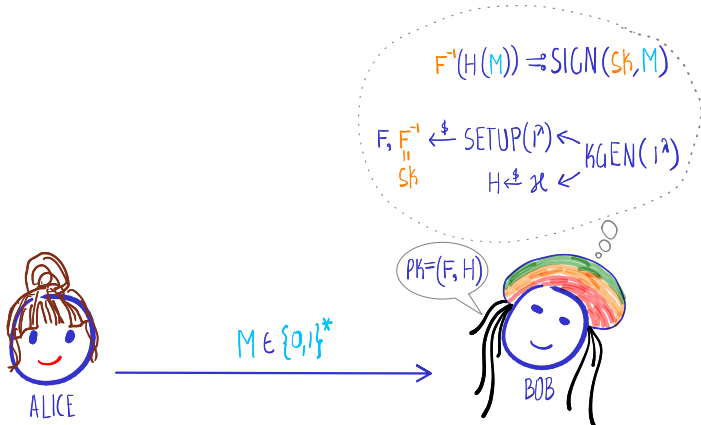
- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function $H: \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$



- ▶ Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

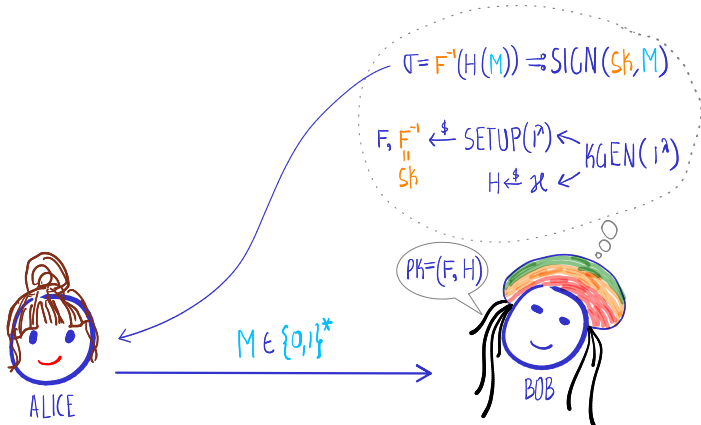
- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function
 $H: \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$



- ▶ Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

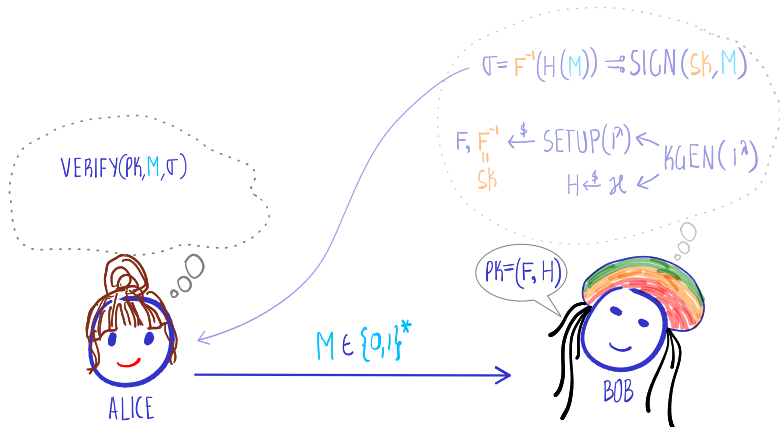
- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function $H : \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$



- ▶ Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

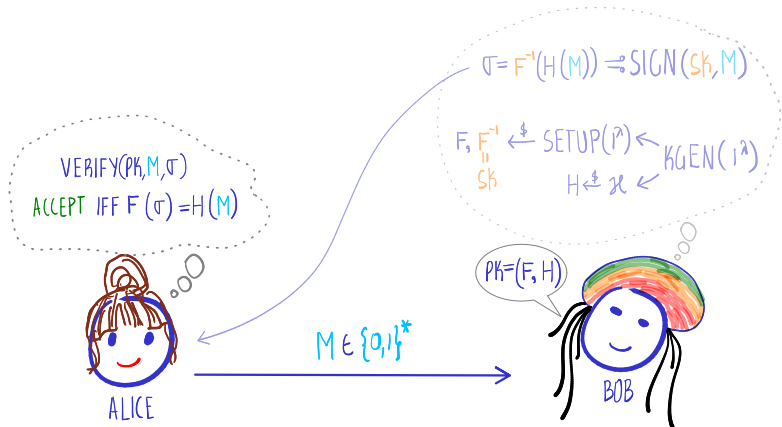
- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function $H: \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$



- ▶ Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

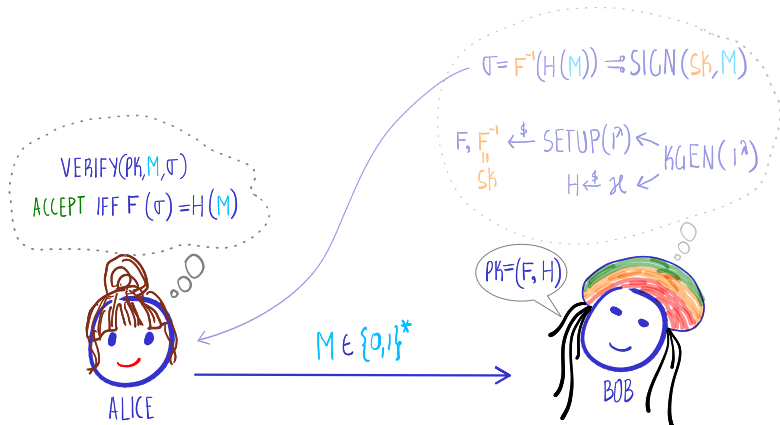
- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function $H: \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$



- ▶ Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

- ▶ TDP (F, F^{-1}) over domain \mathcal{D} + hash function $H: \{0, 1\}^* \rightarrow \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0, 1\}^*$



- ▶ **Efficient:** compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.


RANDOM ORACLE H

ADVERSARY



REDUCTION

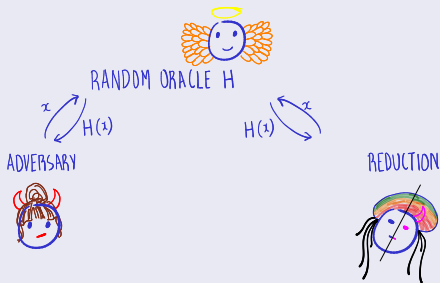


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

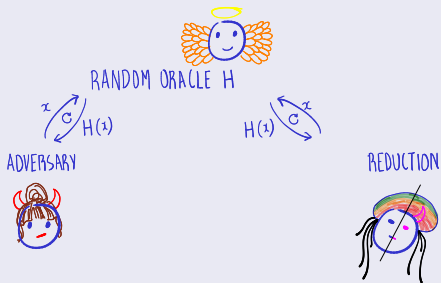


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

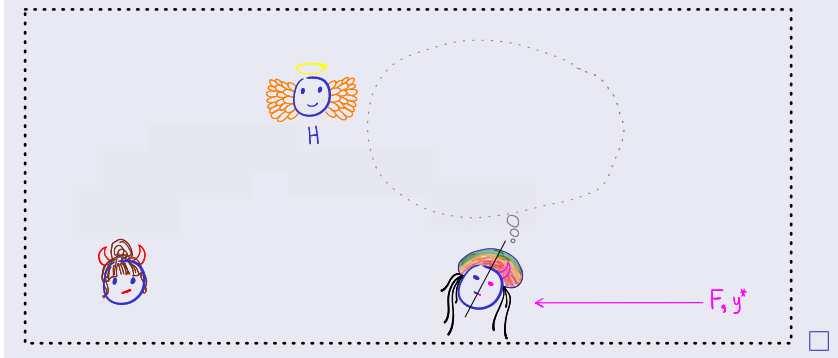


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

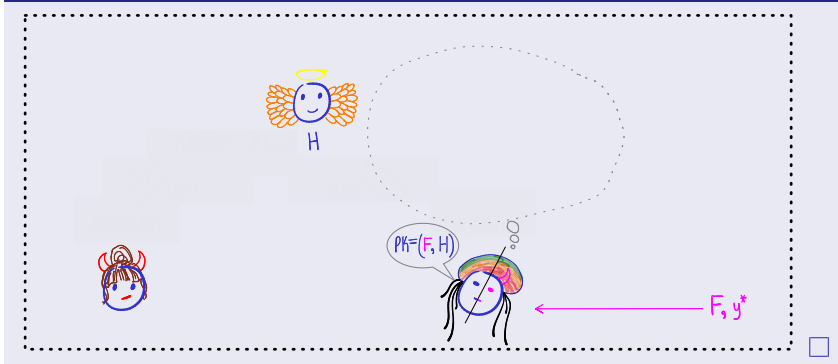


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

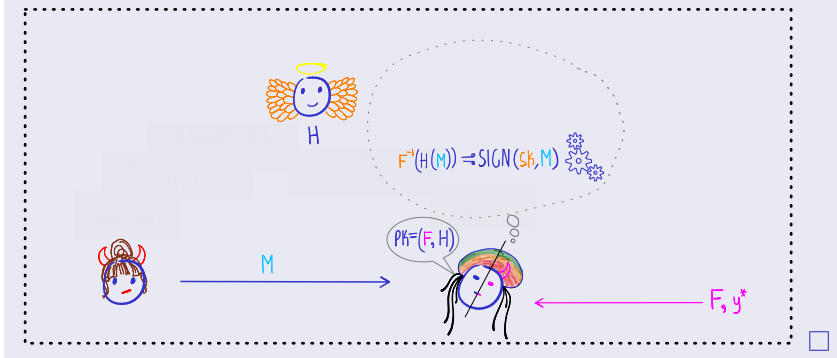


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

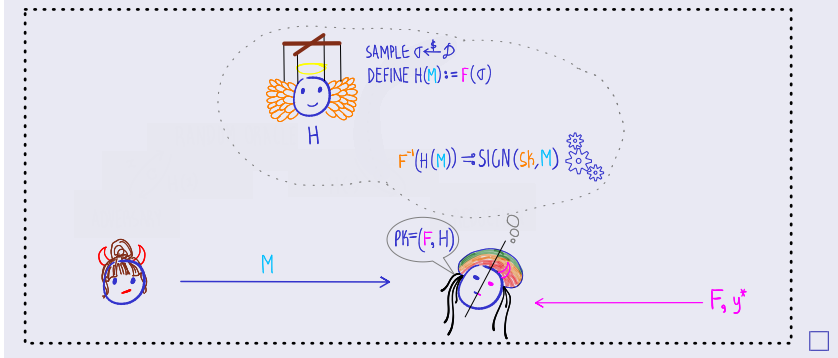


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

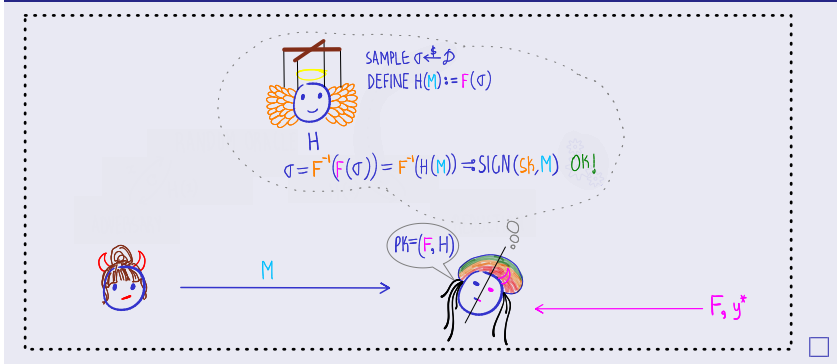


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

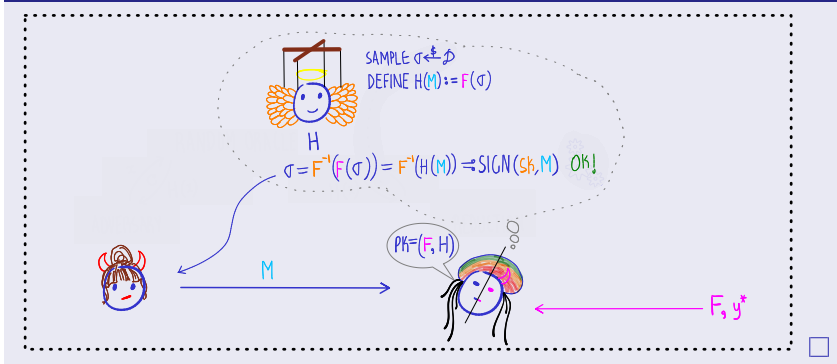


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

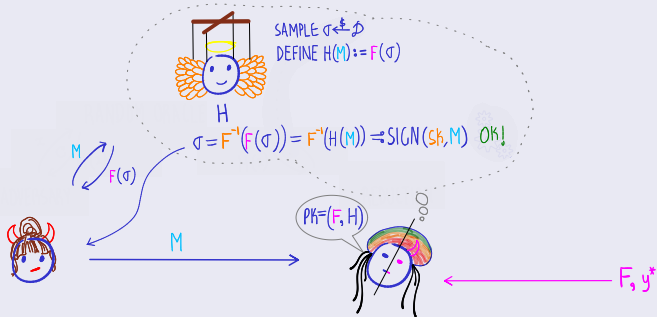


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

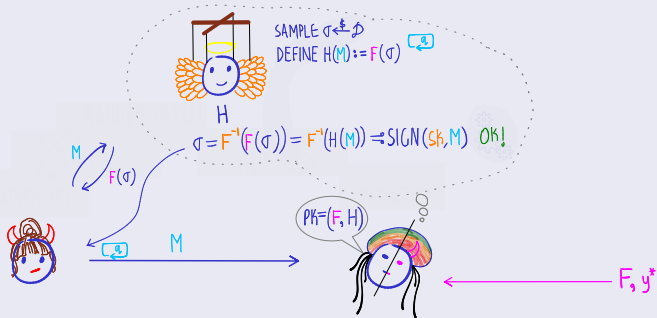


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

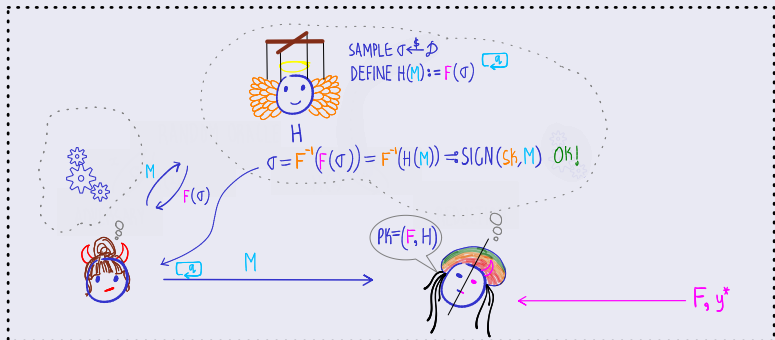


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

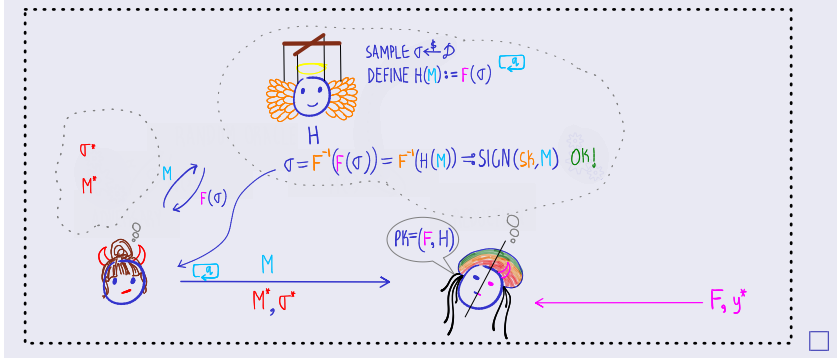


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

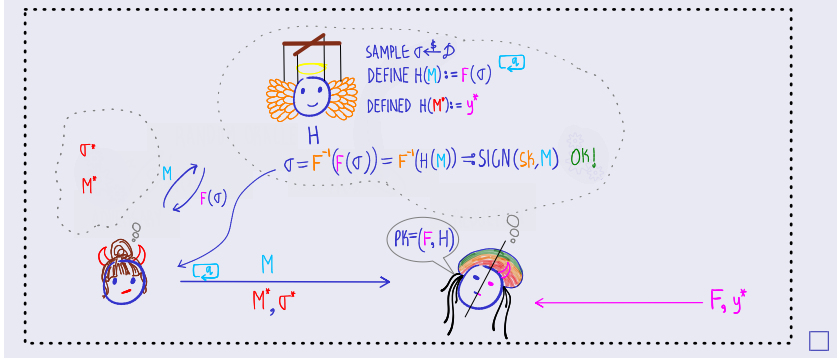


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

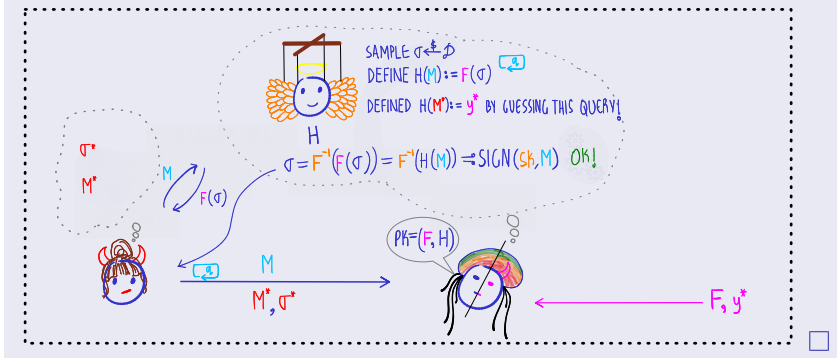


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

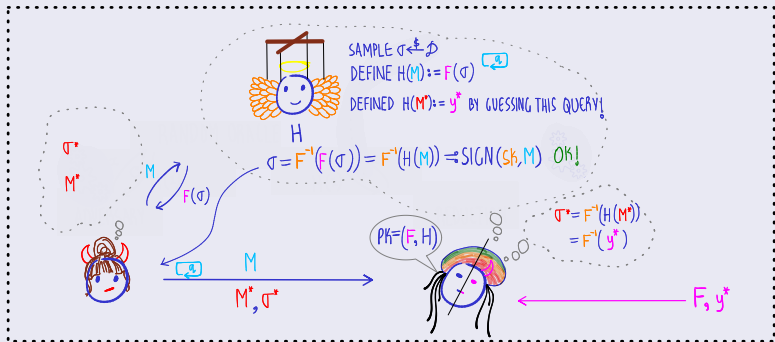


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.

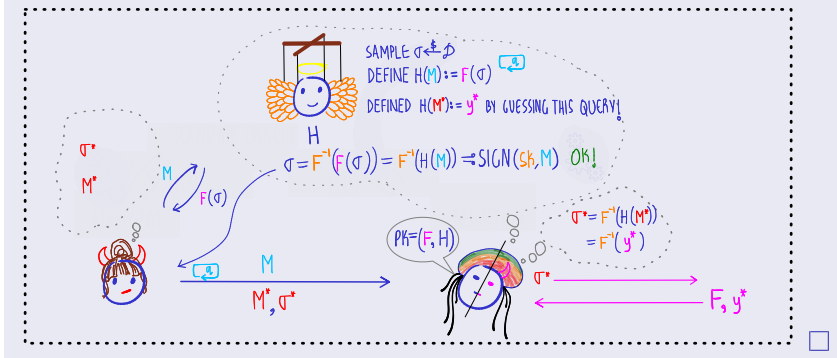


Efficient Signatures via Hash-then-Invert...

Theorem 7

If (F, F^{-1}) is a TDP and H is a *random oracle* then Σ is secure.

Proof sketch: random oracle programming.



Plan for this Session

Digital Signature: Syntax and Modelling Security

One-Time Signatures

Many-Time (Stateful) Signatures

Efficient Signatures via Hash-and-Sign

Wrapping Up

To Recap

- ▶ Constructions:

1. Theoretical construction of signature using OWF and CRHF

- ▶ CRHF can be replaced with *universal one-way* hash function (UOWHF), which can be constructed from OWF
- ▶ Can be shown to be **quantum** secure



To Recap

► Constructions:

1. Theoretical construction of signature using OWF and CRHF
 - CRHF can be replaced with *universal one-way* hash function (UOWHF), which can be constructed from OWF
 - Can be shown to be **quantum** secure
2. Efficient constructions in ROM
 - RSA-PSS based on Hash-then-Invert
 - Other approach: Fiat-Shamir Transform (e.g., Schnorr)



To Recap

► Constructions:

1. Theoretical construction of signature using OWF and CRHF
 - CRHF can be replaced with *universal one-way* hash function (UOWHF), which can be constructed from OWF
 - Can be shown to be **quantum** secure
2. Efficient constructions in ROM
 - RSA-PSS based on Hash-then-Invert
 - Other approach: Fiat-Shamir Transform (e.g., Schnorr)



► Takeaways:

To Recap

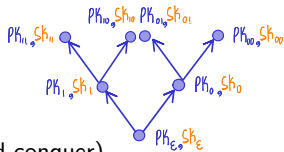
► Constructions:

1. Theoretical construction of signature using OWF and CRHF
 - CRHF can be replaced with *universal one-way* hash function (UOWHF), which can be constructed from OWF
 - Can be shown to be **quantum** secure
2. Efficient constructions in ROM
 - RSA-PSS based on Hash-then-Invert
 - Other approach: Fiat-Shamir Transform (e.g., Schnorr)



► Takeaways:

- Constructive:
 - Bottom up constructive approach
 - Tree-based construction (divide and conquer)



To Recap

► Constructions:

1. Theoretical construction of signature using OWF and CRHF

- CRHF can be replaced with *universal one-way* hash function (UOWHF), which can be constructed from OWF
- Can be shown to be **quantum** secure



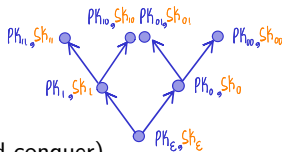
2. Efficient constructions in ROM

- RSA-PSS based on Hash-then-Invert
- Other approach: Fiat-Shamir Transform (e.g., Schnorr)

► Takeaways:

► Constructive:

- Bottom up constructive approach
- Tree-based construction (divide and conquer)

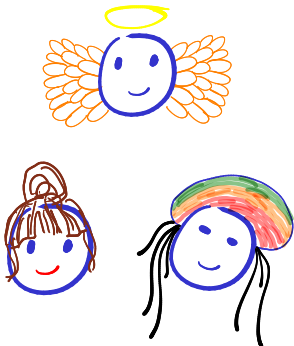


► Proof techniques:

- “Plug and pray”
- Random oracle programming



Thank You for Your Attention! More Questions?



References

1. Digital signature and its security models were formally studied in [GMR88]
2. Lamport's OTS is from [Lam79]
3. The stateful many-time signature is from [KL21], and is a in spirit with Merkle's signatures [Mer90]
4. The "hash-then-invert" paradigm in random-oracle model was studied in [BR93]



Mihir Bellare and Phillip Rogaway.

Random oracles are practical: A paradigm for designing efficient protocols.

In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.



Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest.

A digital signature scheme secure against adaptive chosen-message attacks.

SIAM J. Comput., 17(2):281–308, 1988.



Jonathan Katz and Yehuda Lindell.

Introduction to Modern Cryptography, Third Edition.

CRC Press, 2021.



Leslie Lamport.

Constructing digital signatures from a one-way function.

Technical report, 1979.



Ralph C. Merkle.

A certified digital signature.

In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238. Springer, Heidelberg, August 1990.