Digital Signatures

Chethan Kamath

IITB TRUST LAB

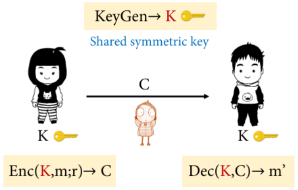
DIGITAL : SECURE : RESPONSIBLE

ACM Summer School 2024, 6/Jun/2024

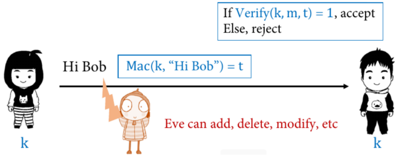
Recall from Prior Sessions

SYMMETRIC KEY UNIVERSE	PUBLIC KEY UNIVERSE

SKE



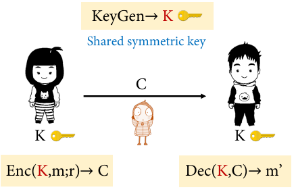
MAC



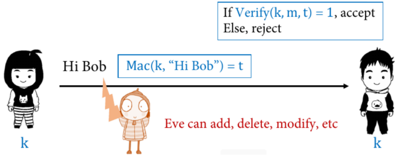
Recall from Prior Sessions

SYMMETRIC KEY UNIVERSE

SKE

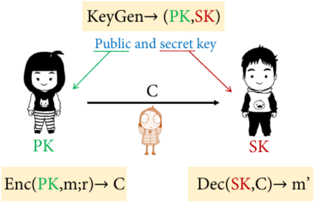


MAC



PUBLIC KEY UNIVERSE

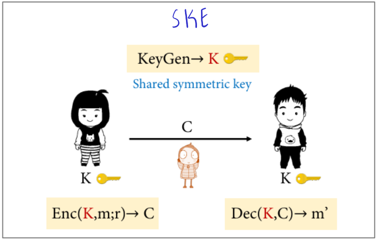
PKE

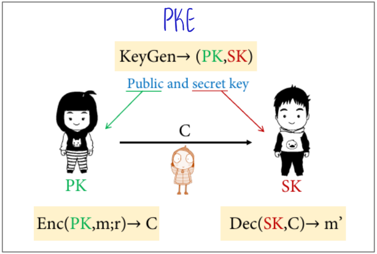


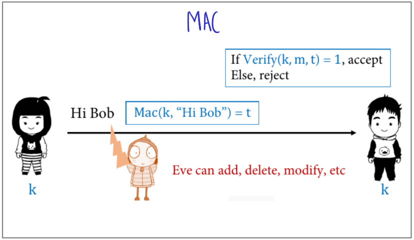
Recall from Prior Sessions

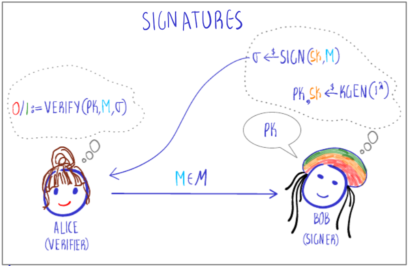
SYMMETRIC KEY UNIVERSE

PUBLIC KEY UNIVERSE









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

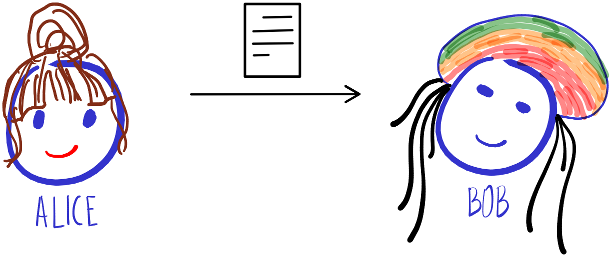
Efficient Signatures via Hash-and-Sign

Wrapping Up

Digital (Analogues of Physical) Signatures...

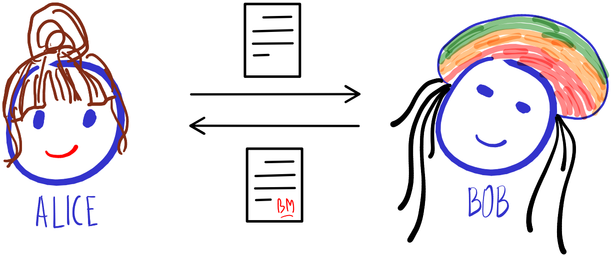
- Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...



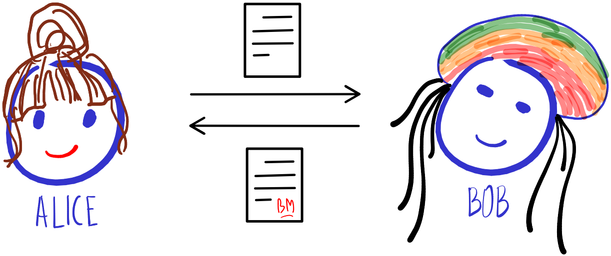
- Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...



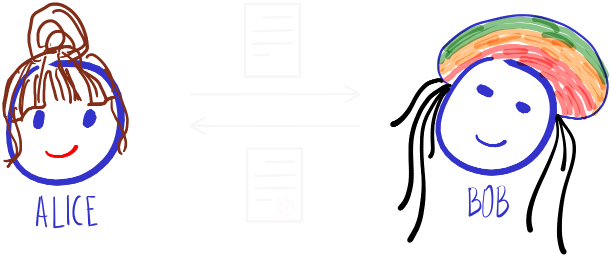
- Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...



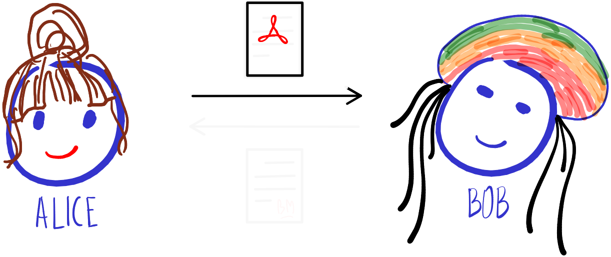
- Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...



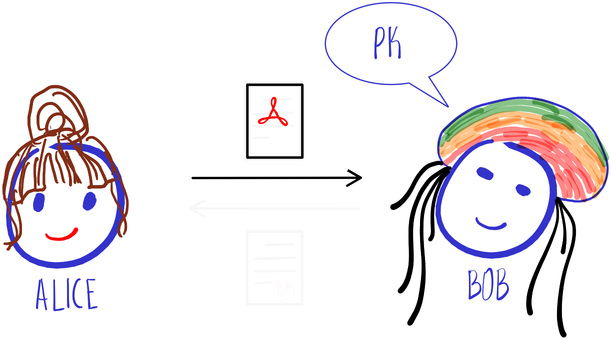
- Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...



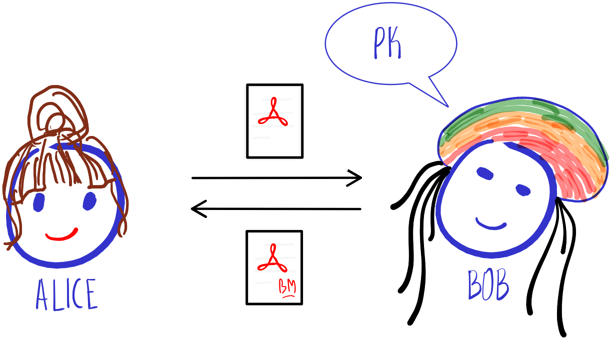
- Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...



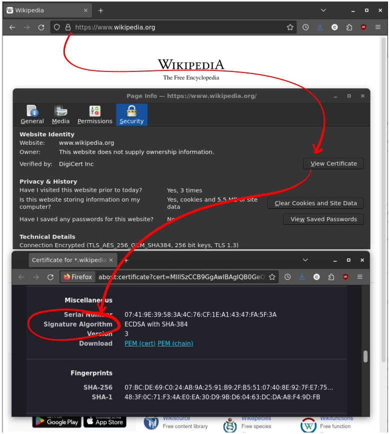
- Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...

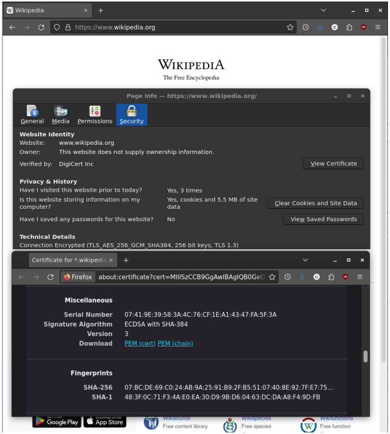


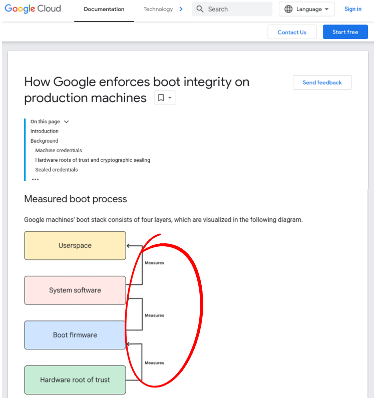
- Requirement: no one should be able to forge Bob's signature

Digital (Analogues of Physical) Signatures...

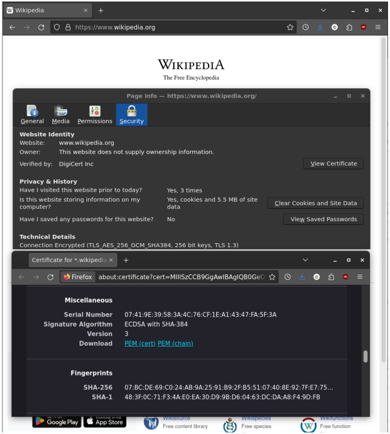


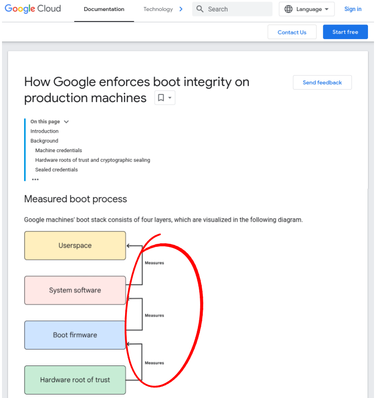
Digital (Analogues of Physical) Signatures...





Digital (Analogues of Physical) Signatures...





- Application to blockchains protocols like Algorand and Chia.

Syntax

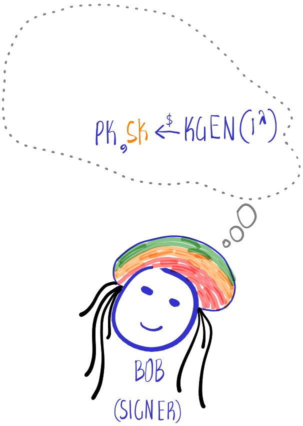
- *Public-key* analogue of message authentication codes (MAC)
  - Triple of algorithms: $(\mathtt{KGEN}, \mathtt{SIGN}, \mathtt{VERIFY})$

Syntax

- *Public-key* analogue of message authentication codes (MAC)
  - Triple of algorithms: $(\texttt{KGEN}, \texttt{SIGN}, \texttt{VERIFY})$

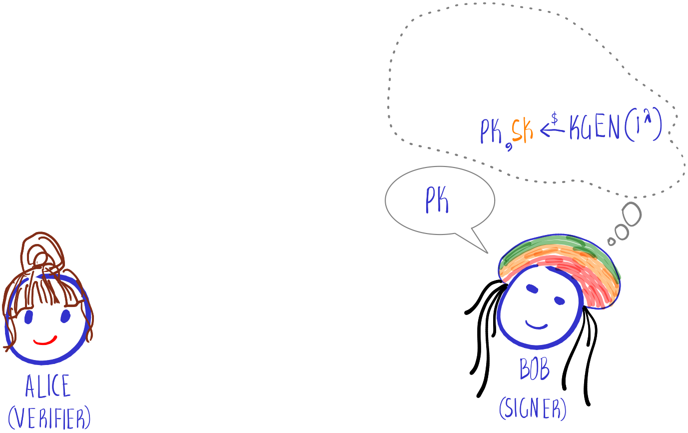
Syntax

- *Public-key* analogue of message authentication codes (MAC)
  - Triple of algorithms: $(\mathtt{KGEN}, \mathtt{SIGN}, \mathtt{VERIFY})$



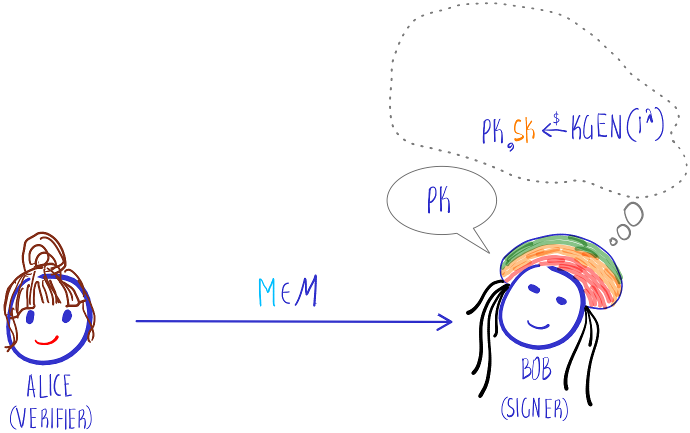
Syntax

- *Public-key* analogue of message authentication codes (MAC)
  - Triple of algorithms: $(\mathtt{KGEN}, \mathtt{SIGN}, \mathtt{VERIFY})$



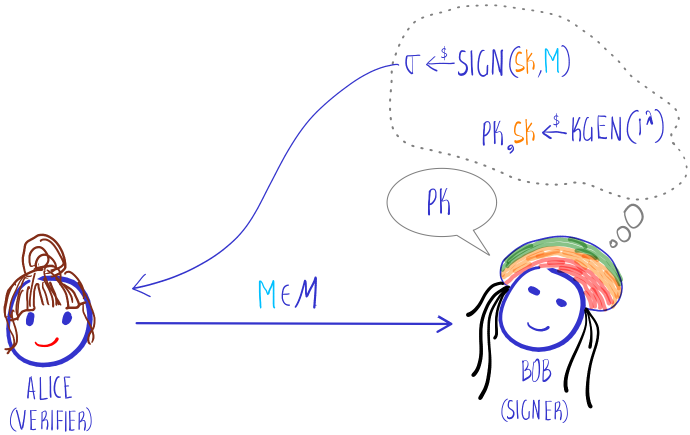
Syntax

- *Public-key* analogue of message authentication codes (MAC)
  - Triple of algorithms: $(\mathtt{KGEN}, \mathtt{SIGN}, \mathtt{VERIFY})$



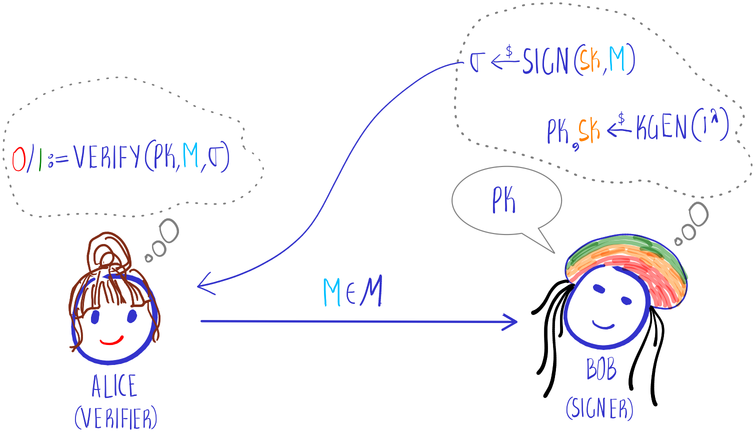
Syntax

- *Public-key* analogue of message authentication codes (MAC)
  - Triple of algorithms: ($\mathtt{KGEN}, \mathtt{SIGN}, \mathtt{VERIFY}$)



Syntax

- *Public-key* analogue of message authentication codes (MAC)
  - Triple of algorithms: $(\mathtt{KGEN}, \mathtt{SIGN}, \mathtt{VERIFY})$



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.

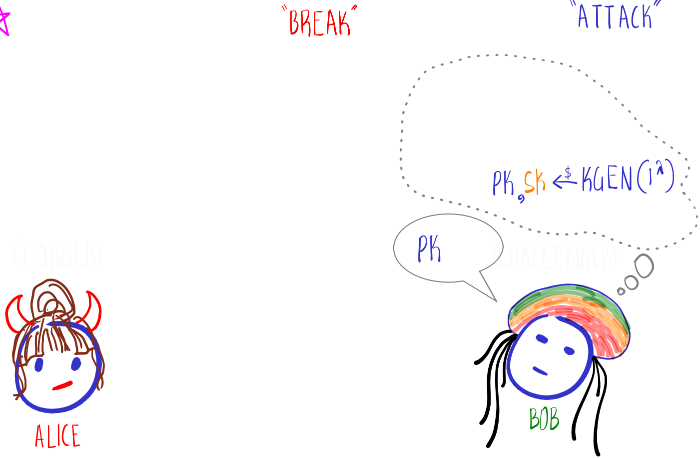
Security: Universal Unforgeability under Key-Only Attack

"BREAK" "ATTACK"

Definition 1

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

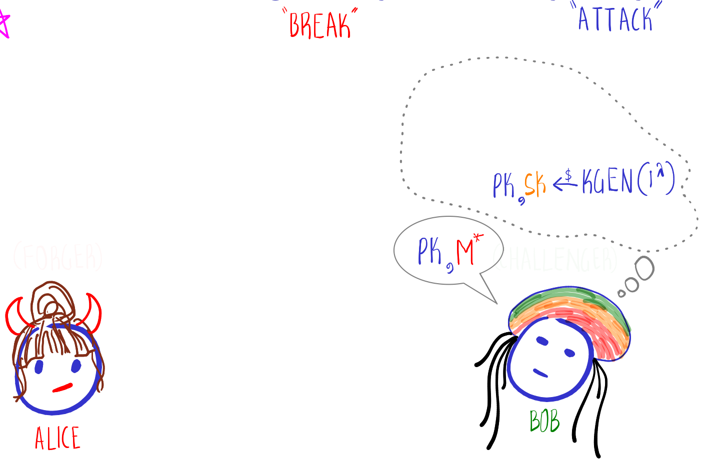
Security: Universal Unforgeability under Key-Only Attack



Definition 1

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

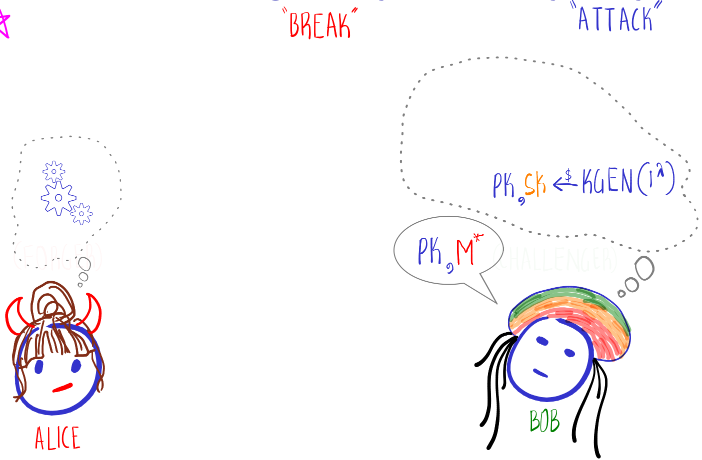
Security: Universal Unforgeability under Key-Only Attack



Definition 1

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

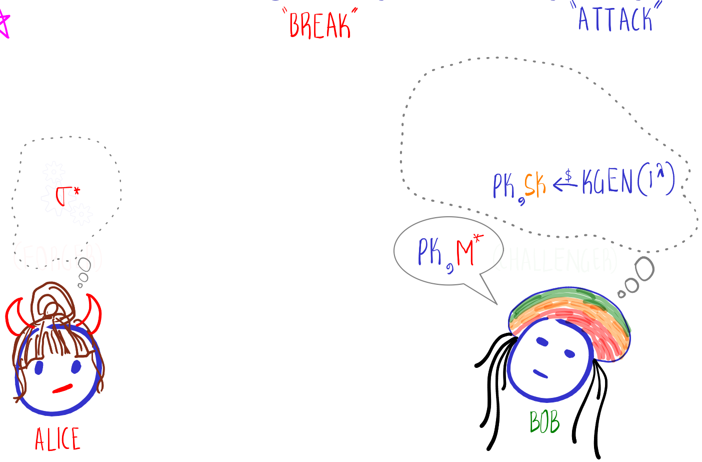
Security: Universal Unforgeability under Key-Only Attack



Definition 1

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

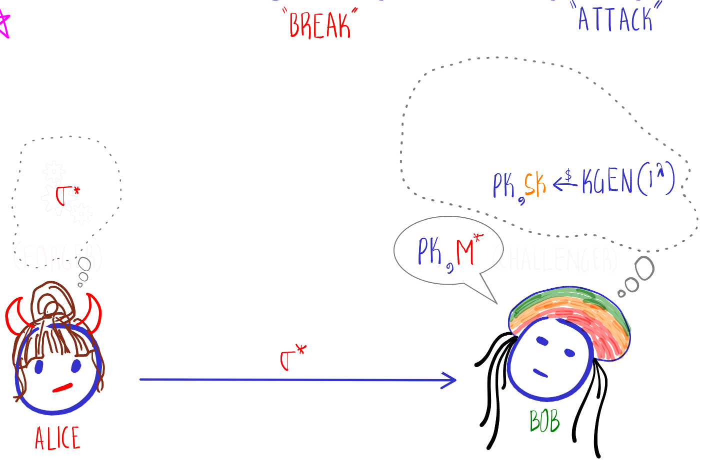
Security: Universal Unforgeability under Key-Only Attack



Definition 1

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

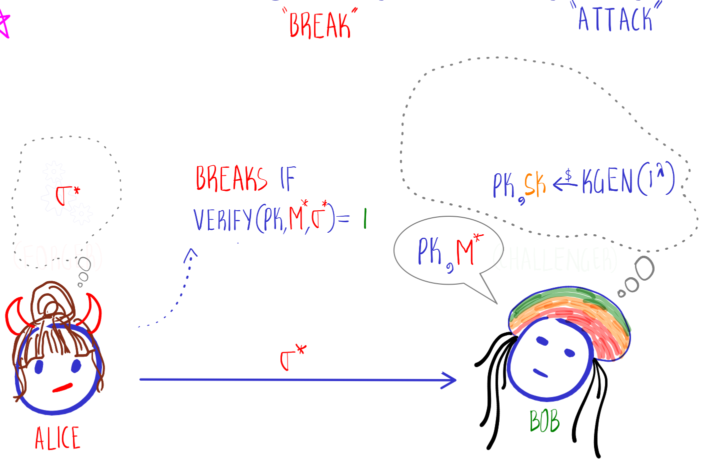
Security: Universal Unforgeability under Key-Only Attack



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

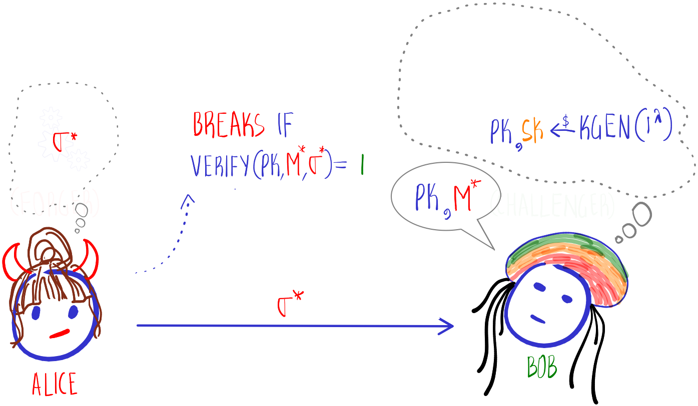


Definition 1

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: Existential Unforg. under Key-Only Attack

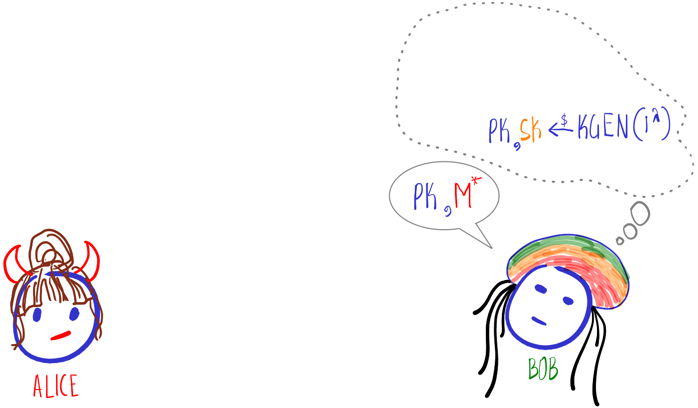
$\forall \rightarrow \exists$

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

$\forall \to \exists$

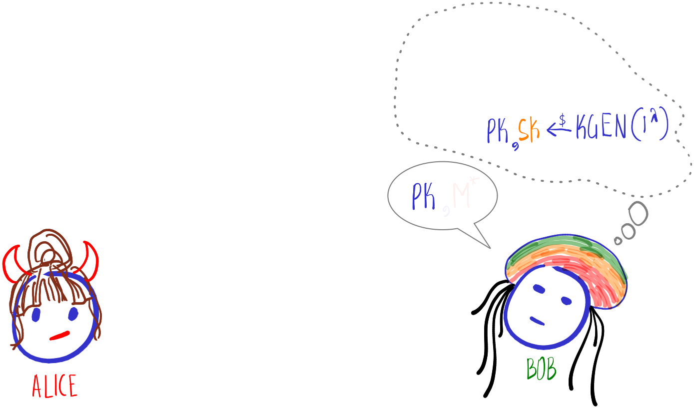


Definition 2

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

$\forall \rightarrow \exists$

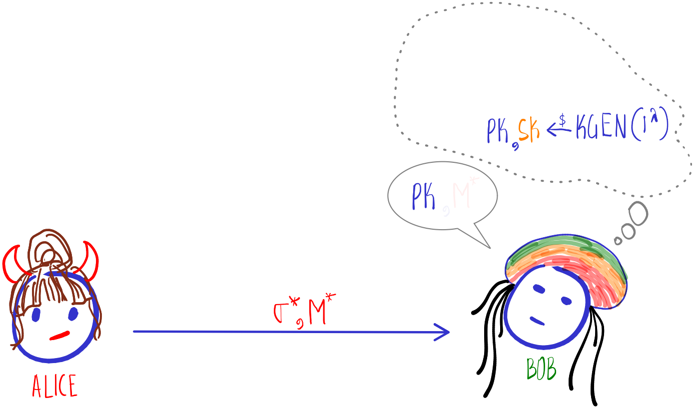


Definition 2

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

★★☆☆ $\forall \to \exists$

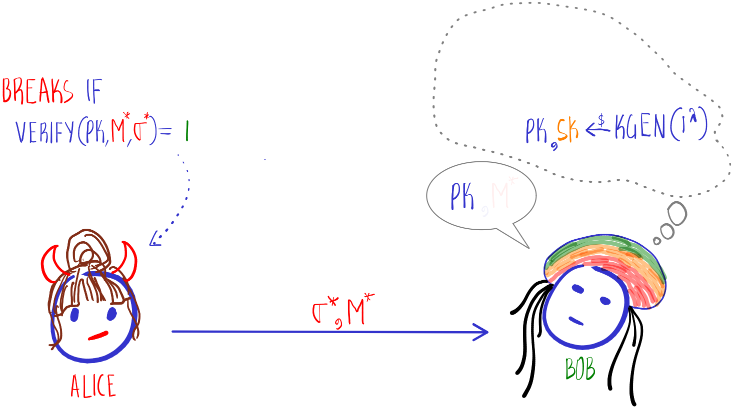


Definition 2

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

$\forall \rightarrow \exists$

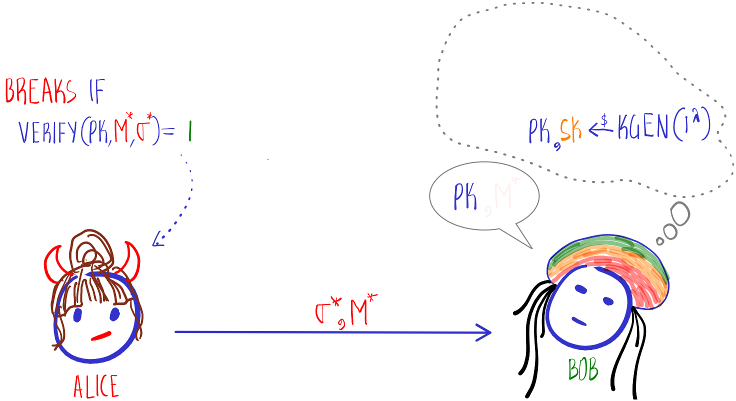


Definition 2

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

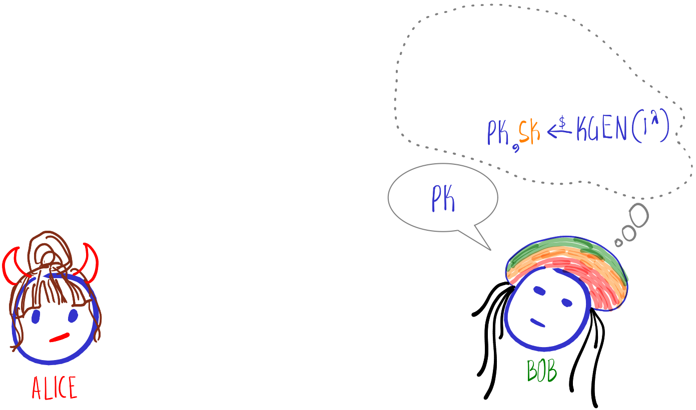
$\forall \rightarrow \exists$



Definition 2

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

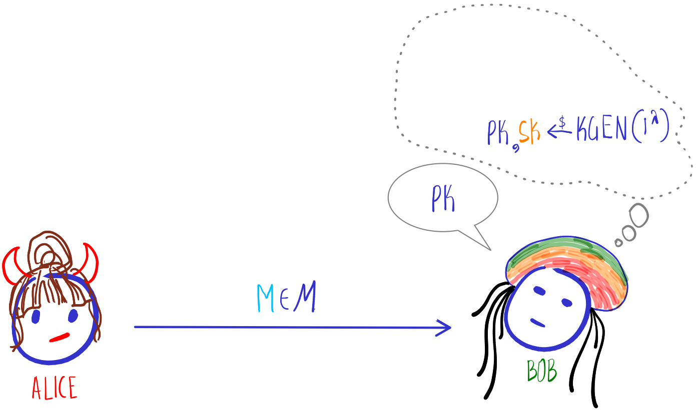
Security: Existential Unforg. under Chosen-Message Attack



Definition 3

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

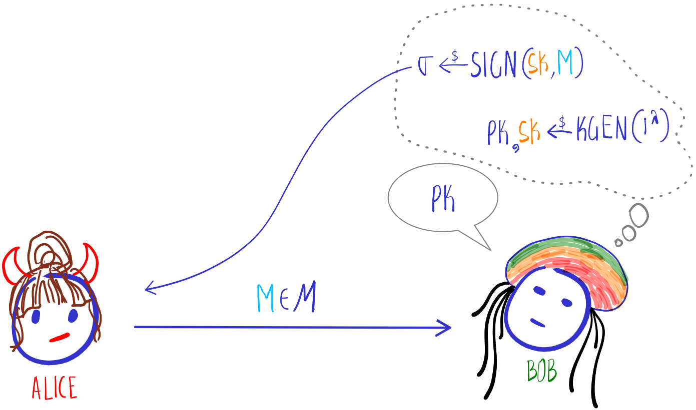
Security: Existential Unforg. under Chosen-Message Attack



Definition 3

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

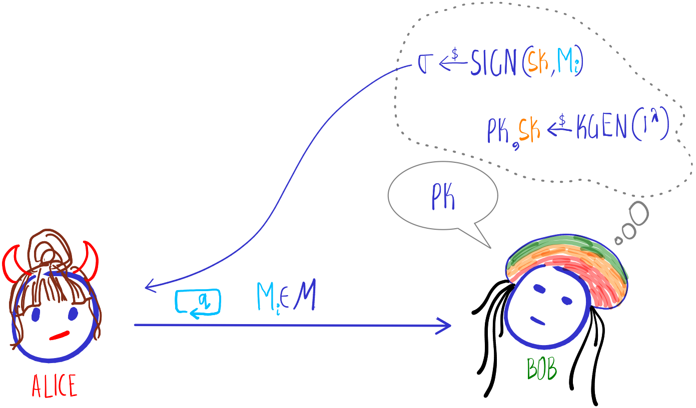
Security: Existential Unforg. under Chosen-Message Attack



Definition 3

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

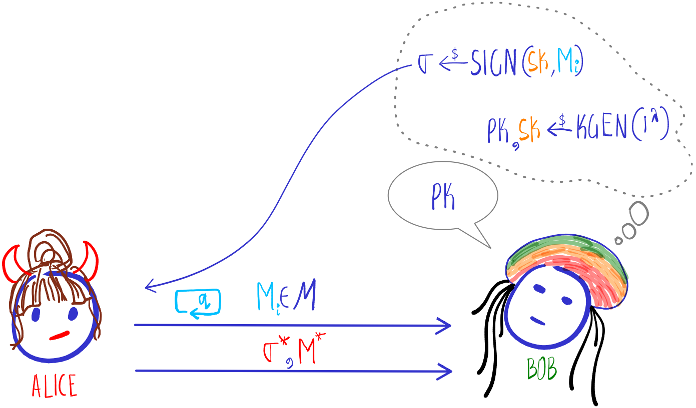
Security: Existential Unforg. under Chosen-Message Attack



Definition 3

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

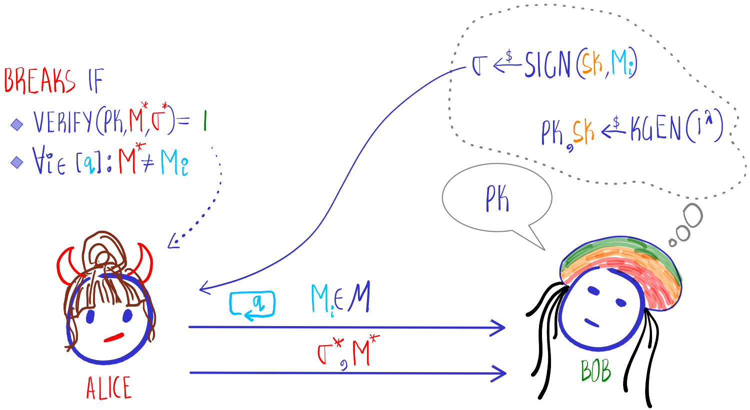
Security: Existential Unforg. under Chosen-Message Attack



Definition 3

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

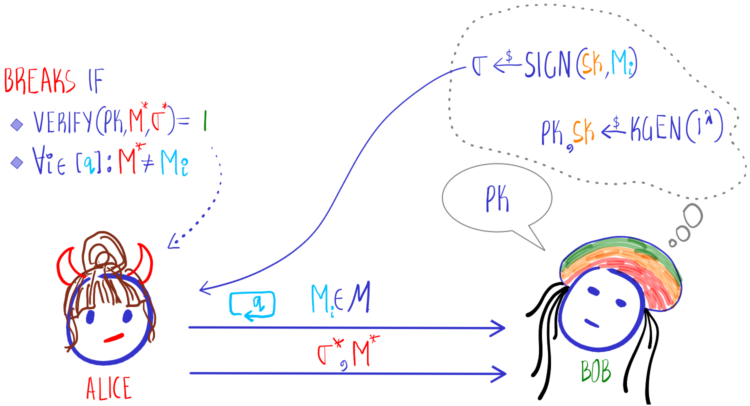
Security: Existential Unforg. under Chosen-Message Attack ★★★☆



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.

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...

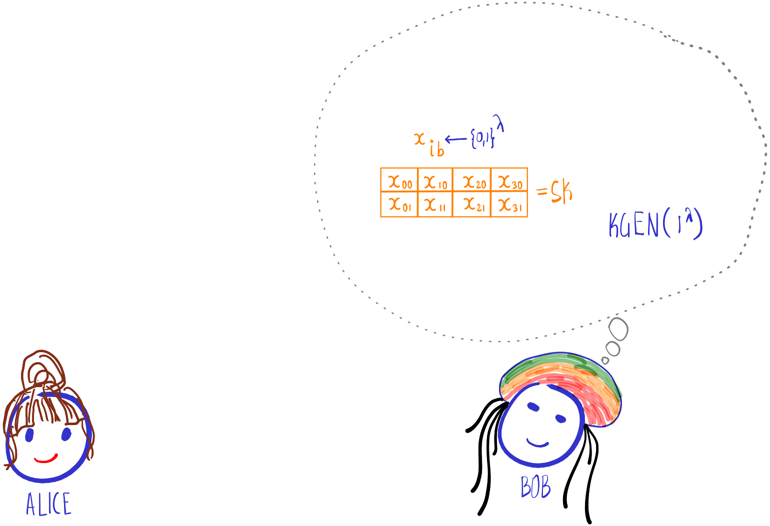
- One-way func. $f : \{0,1\}^\lambda \to \{0,1\}^\lambda \Rightarrow$ OTS for $\mathcal{M} := \{0,1\}^\ell$

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

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

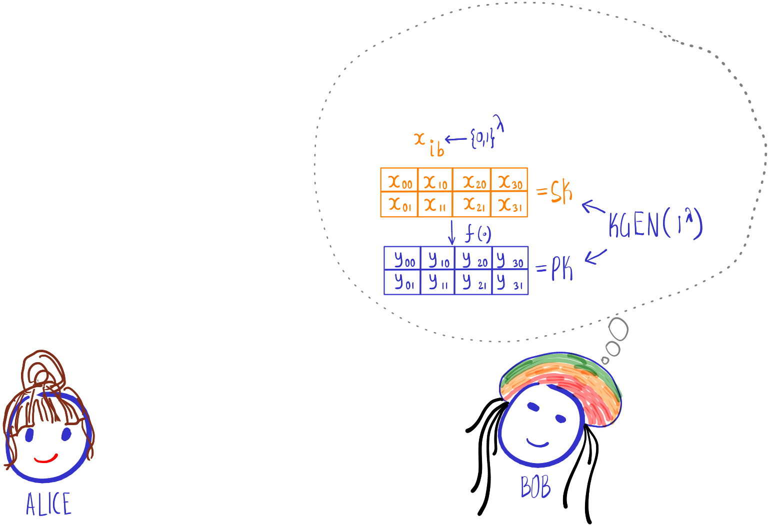
One-Time Signatures ($q = 1$): Lamport's Signature...

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



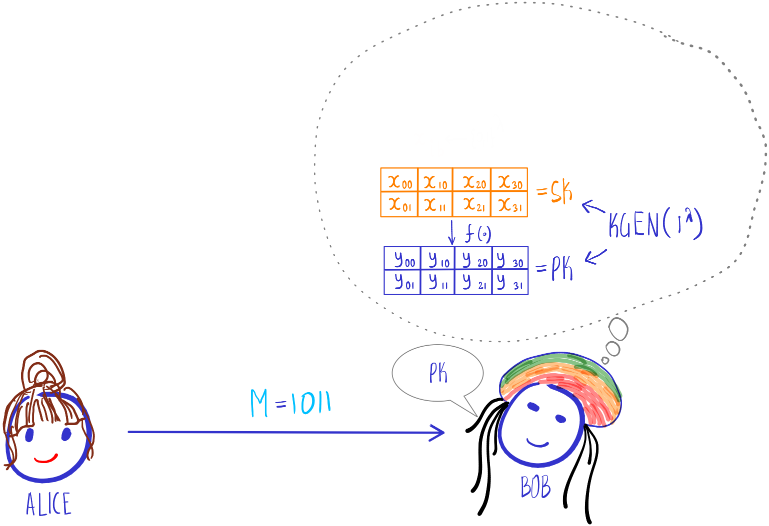
One-Time Signatures ($q = 1$): Lamport's Signature...

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



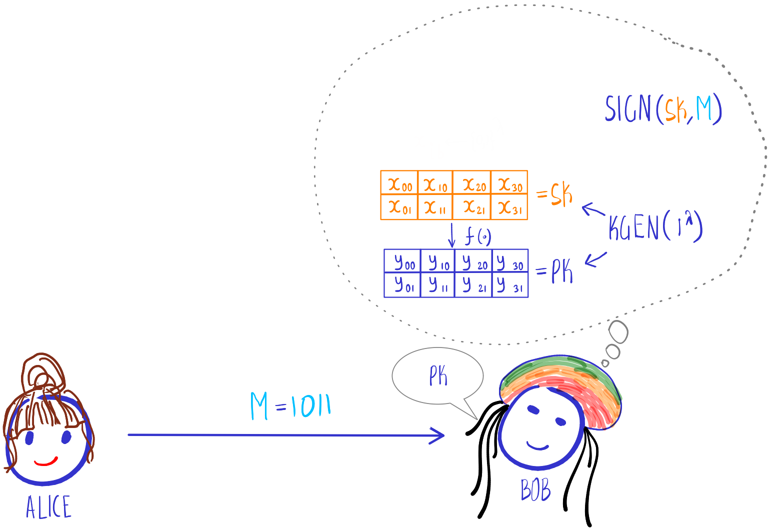
One-Time Signatures ($q = 1$): Lamport's Signature...

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



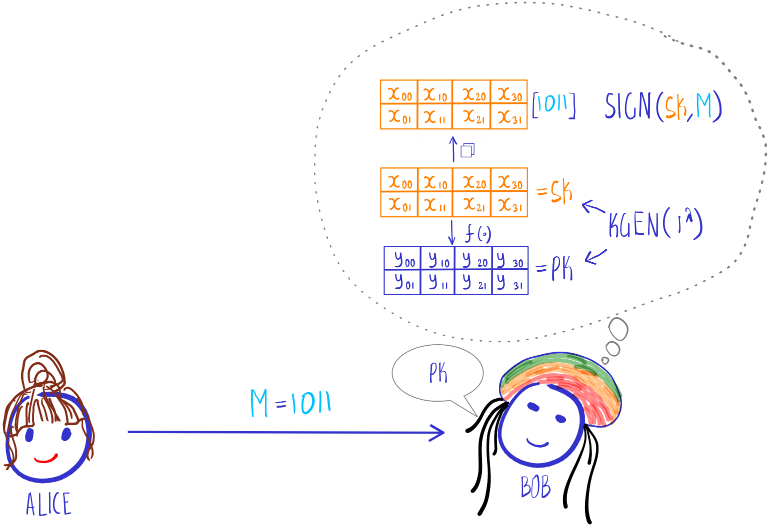
One-Time Signatures ($q = 1$): Lamport's Signature...

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



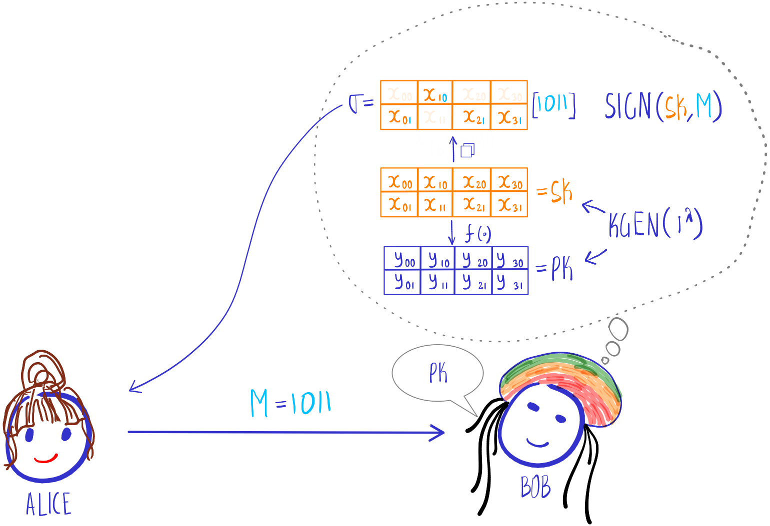
One-Time Signatures ($q = 1$): Lamport's Signature...

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



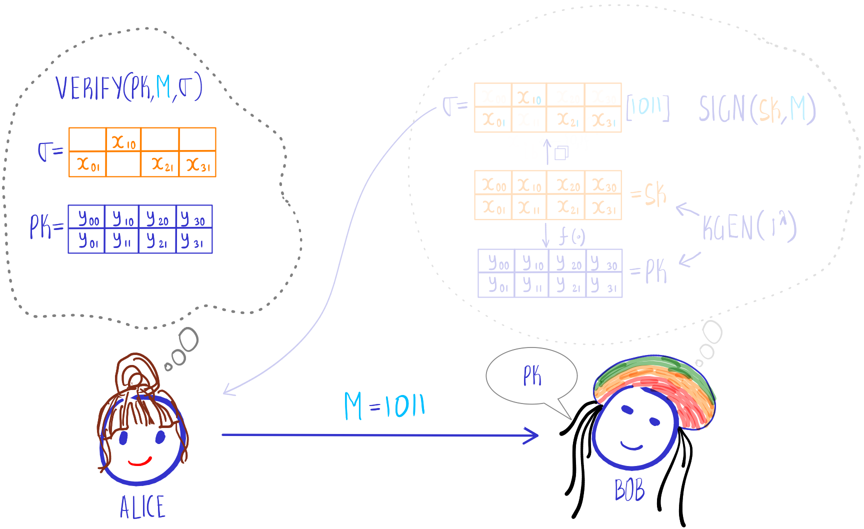
One-Time Signatures ($q = 1$): Lamport's Signature...

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



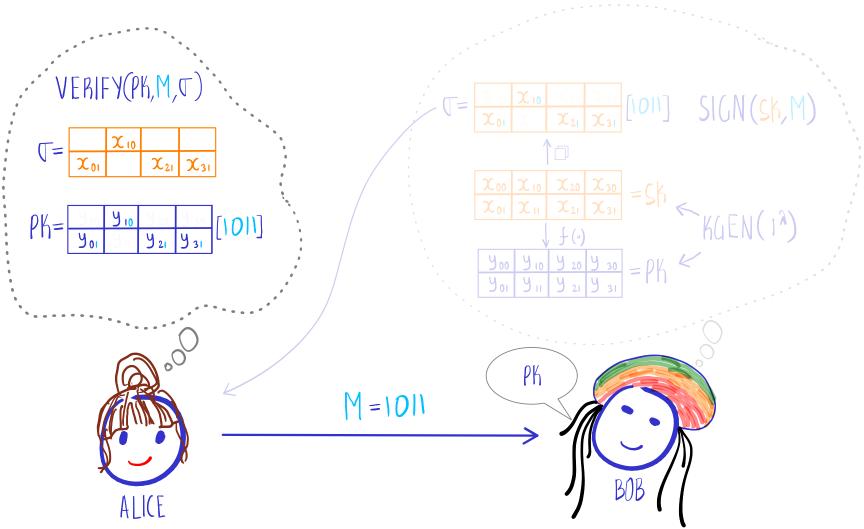
One-Time Signatures ($q = 1$): Lamport's Signature...

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



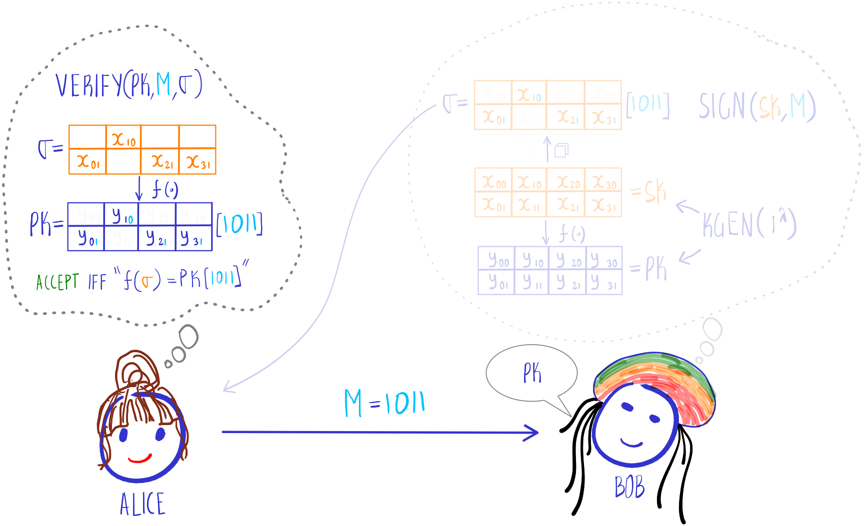
One-Time Signatures ($q = 1$): Lamport's Signature...

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



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

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



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

Theorem 4

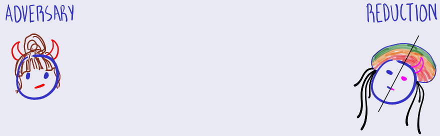
If f is a OWF then Lamport's scheme is a one-time signature.

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".



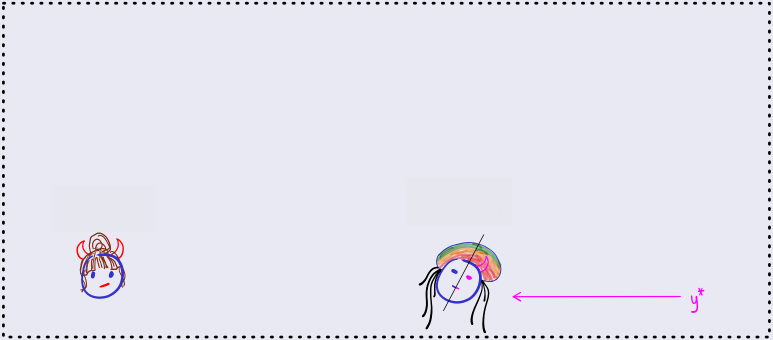
$\square$

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".



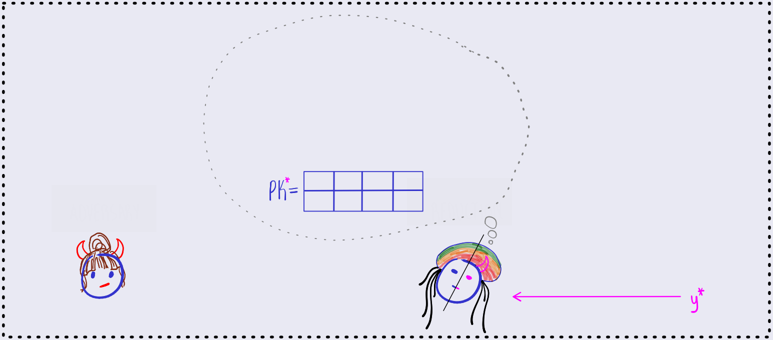
□

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".



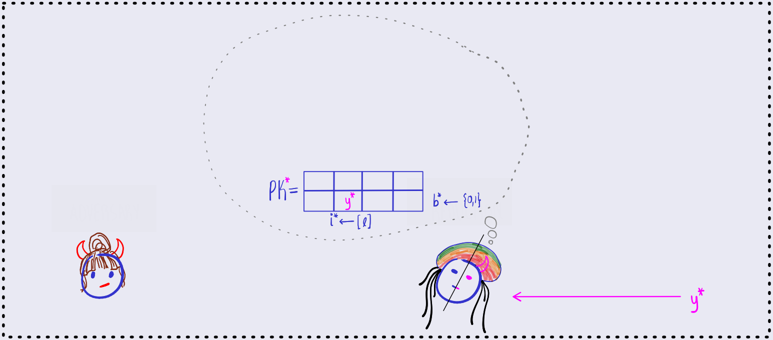
□

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".

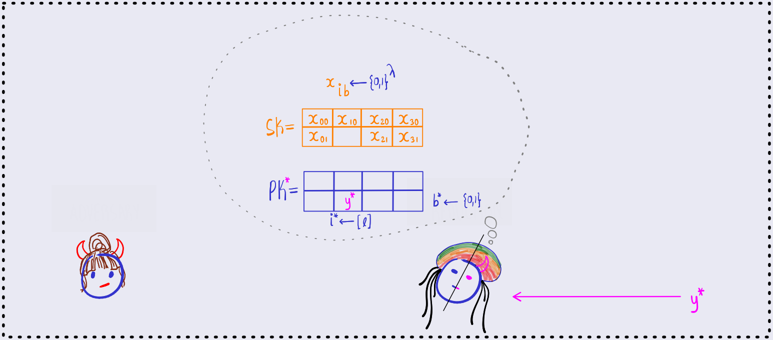


One-Time Signatures ($q = 1$): Lamport's Signature...

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".

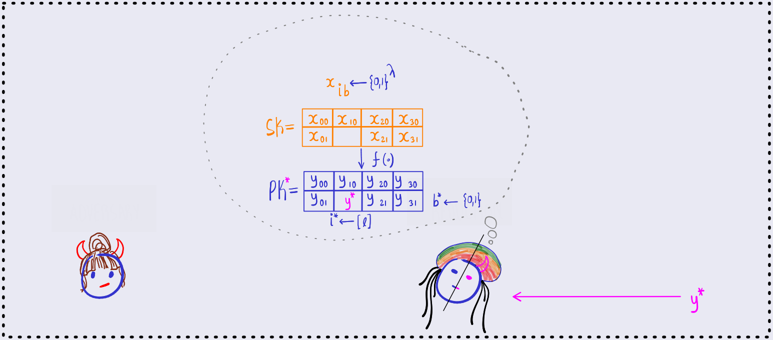


One-Time Signatures ($q = 1$): Lamport's Signature...

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".



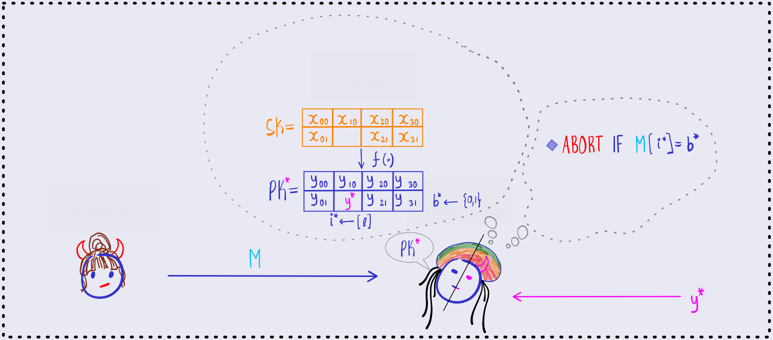
□

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".

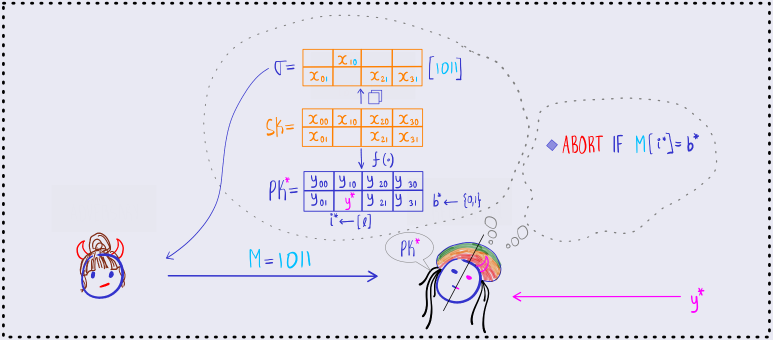


One-Time Signatures ($q = 1$): Lamport's Signature...

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".



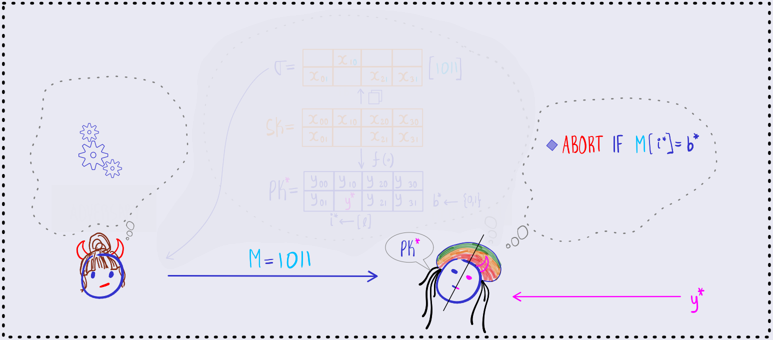
□

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".



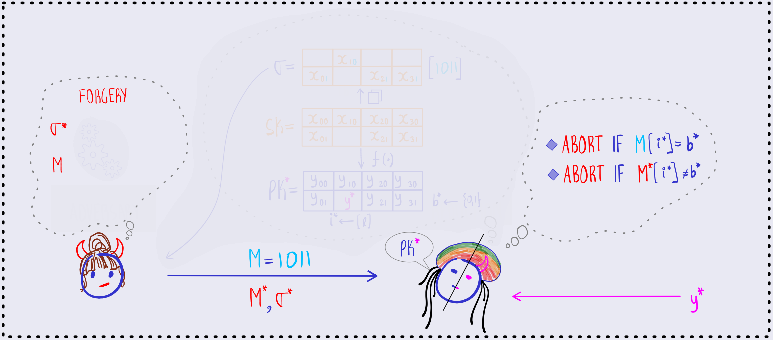
□

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".



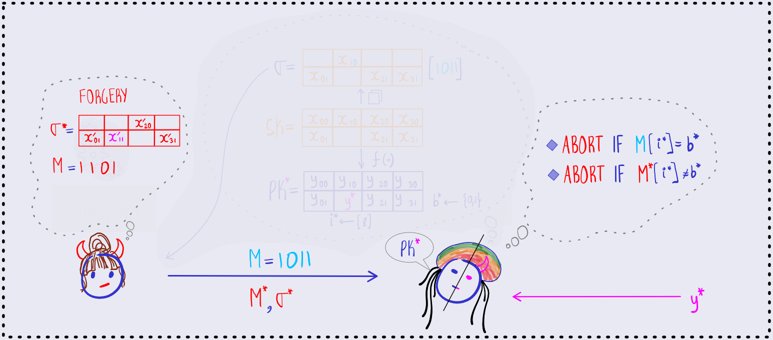
□

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".



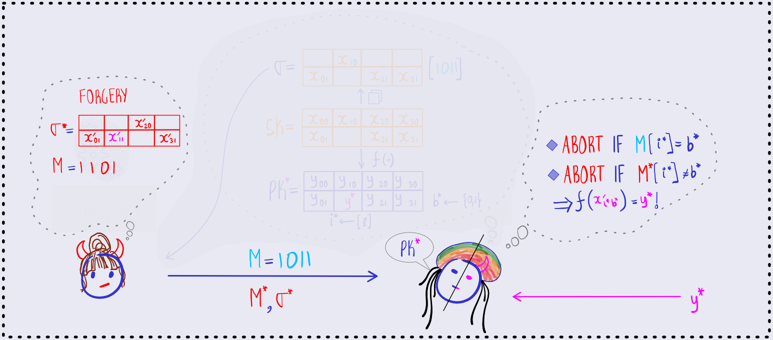
□

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".



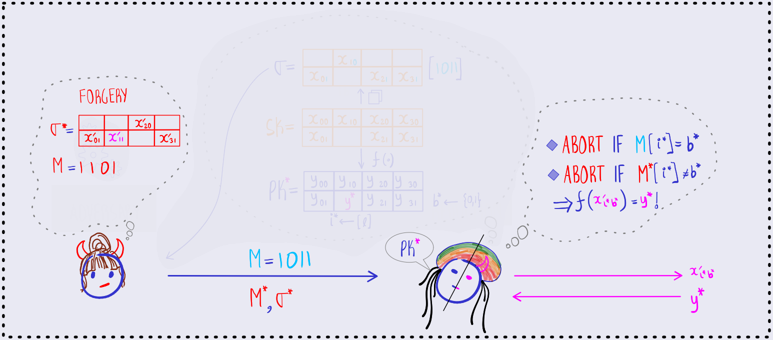
$\square$

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".



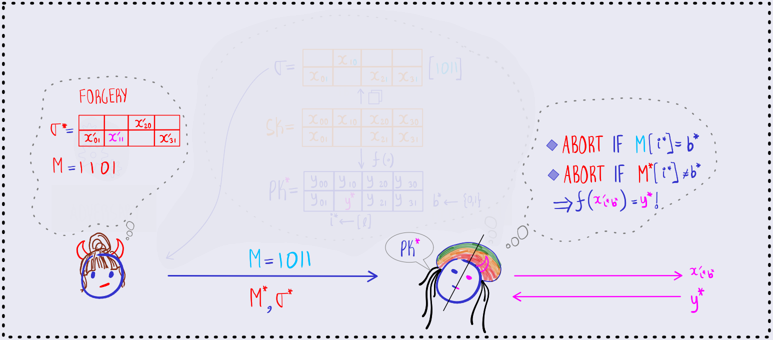
□

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".



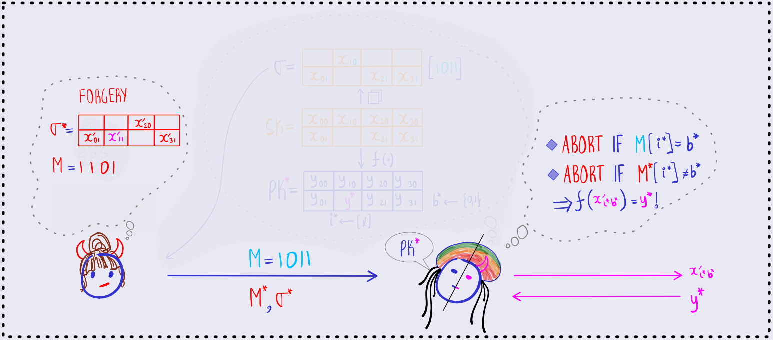
ANALYSIS: BOB INVERTS f AT y^* IF ALICE FORGES $\wedge$ $\overline{\text{ABORT}}$ $\square$

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".



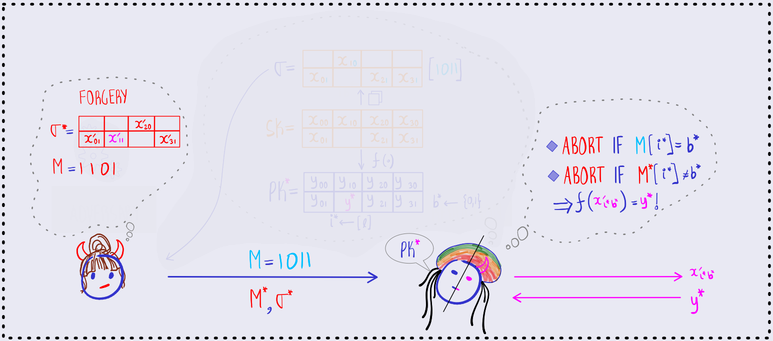
ANALYSIS: Pr[BOB INVERTS f AT y*] = Pr[ALICE FORGES ∧ $\overline{\text{ABORT}}$] □

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

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 $H : \{0,1\}^{2\ell} \to \{0,1\}^{\ell}$, 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

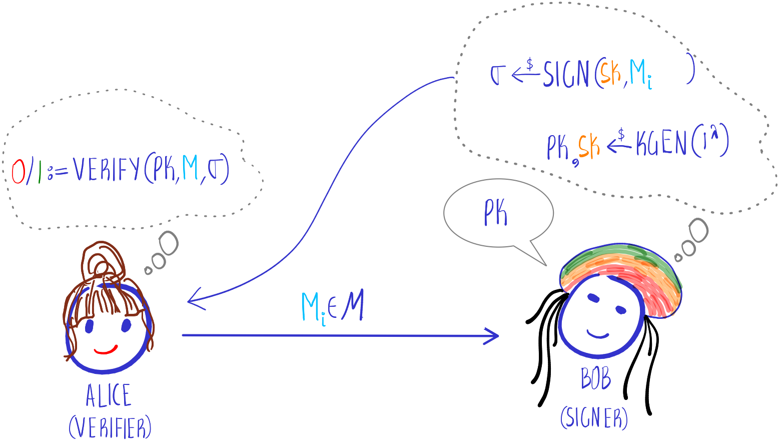
(Many-Time) Signatures with Stateful Signer

- 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

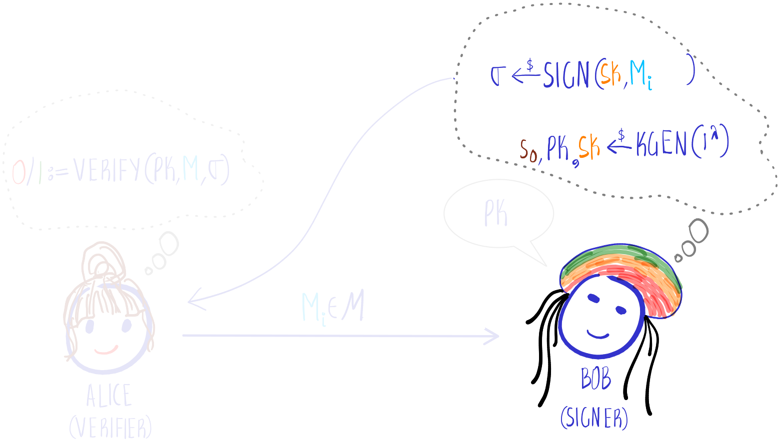
- Syntax: same as before except that `SIGN` is *stateful*



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

(Many-Time) Signatures with Stateful Signer

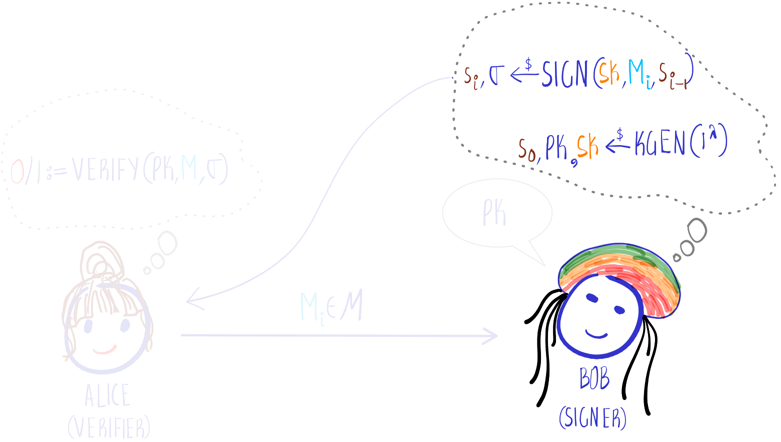
- Syntax: same as before except that `SIGN` is *stateful*



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

(Many-Time) Signatures with Stateful Signer

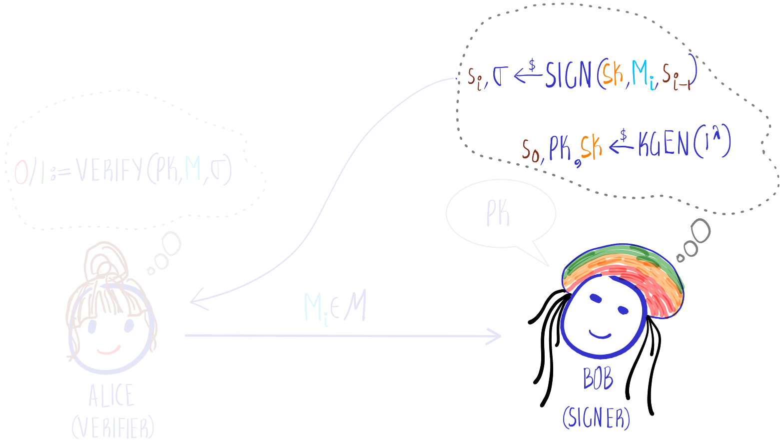
- Syntax: same as before except that `SIGN` is *stateful*



- Security: same as before (i.e., 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) Stateful Signatures...

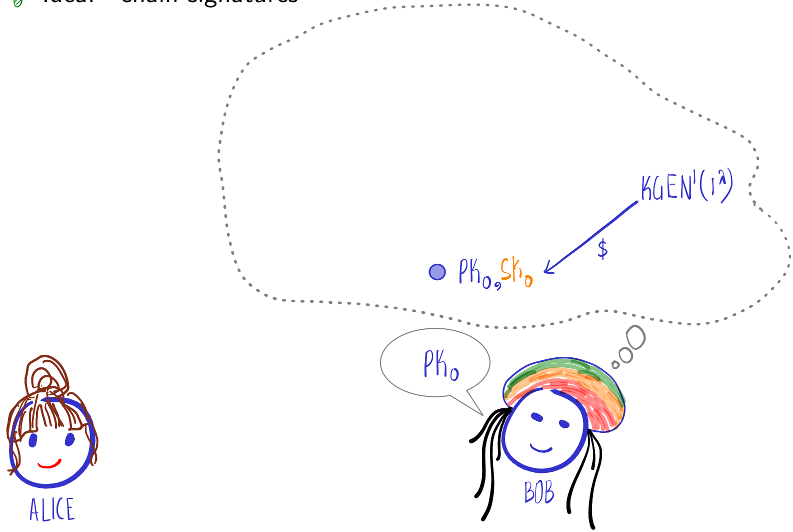
- OTS $\Sigma^1 = (\mathtt{KGEN}^1, \mathtt{SIGN}^1, \mathtt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.

(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\mathtt{KGEN}^1, \mathtt{SIGN}^1, \mathtt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"

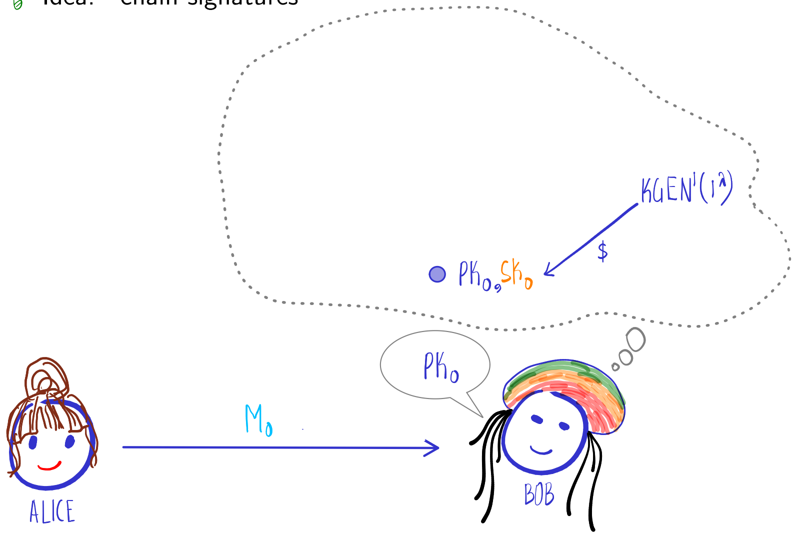
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\texttt{KGEN}^1, \texttt{SIGN}^1, \texttt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



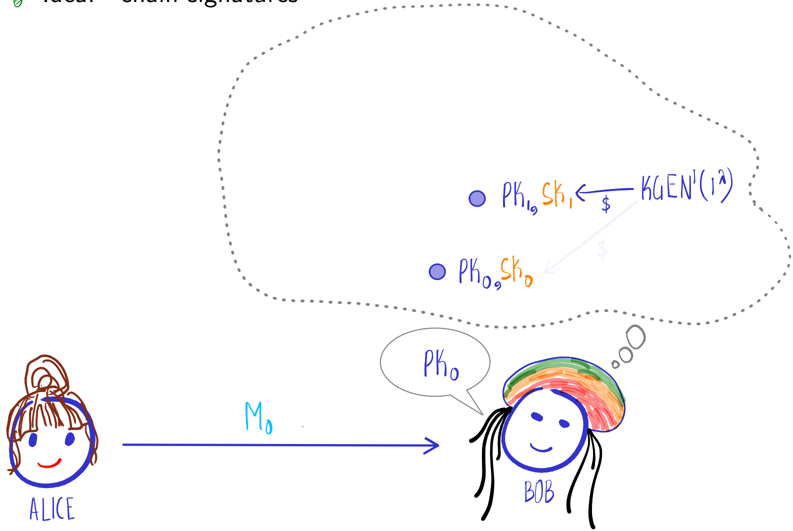
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\texttt{KGEN}^1, \texttt{SIGN}^1, \texttt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



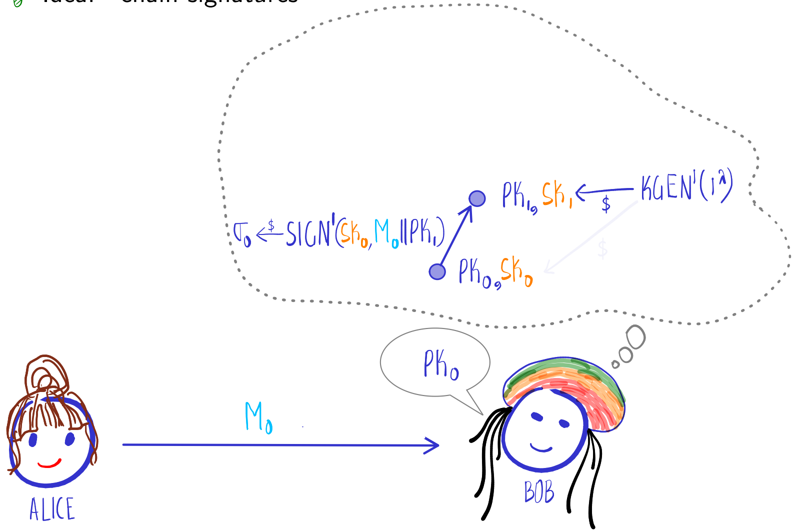
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\mathtt{KGEN}^1, \mathtt{SIGN}^1, \mathtt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



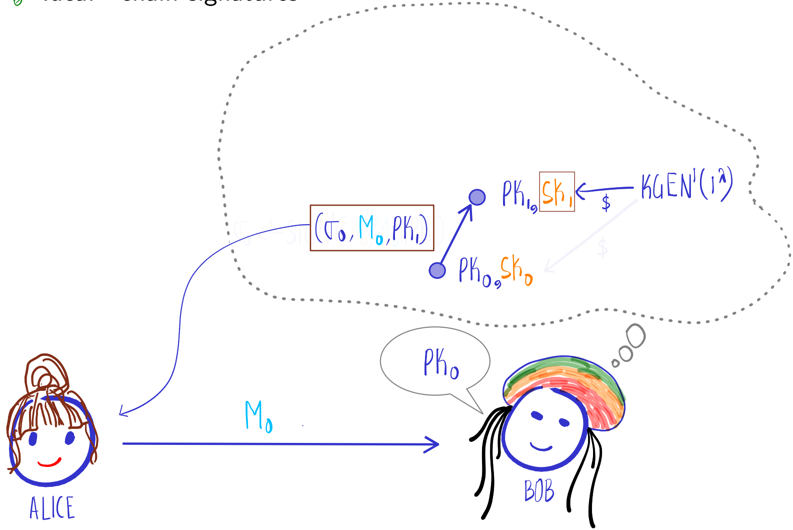
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\mathtt{KGEN}^1, \mathtt{SIGN}^1, \mathtt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



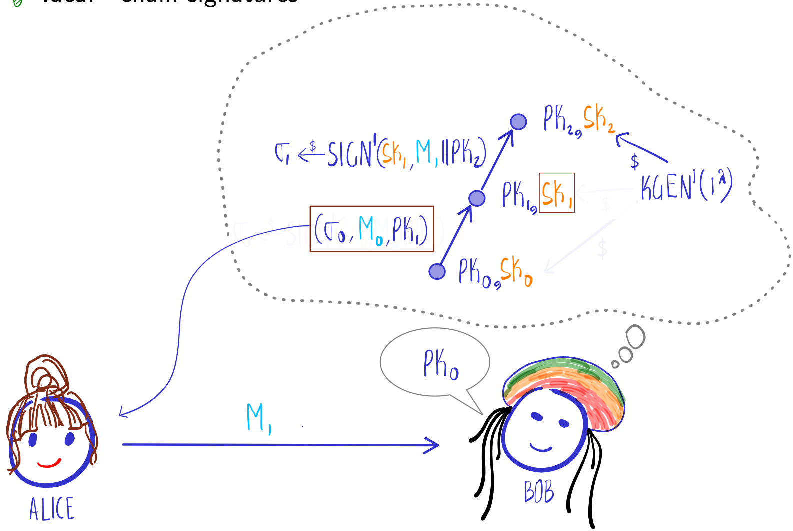
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\mathtt{KGEN}^1, \mathtt{SIGN}^1, \mathtt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



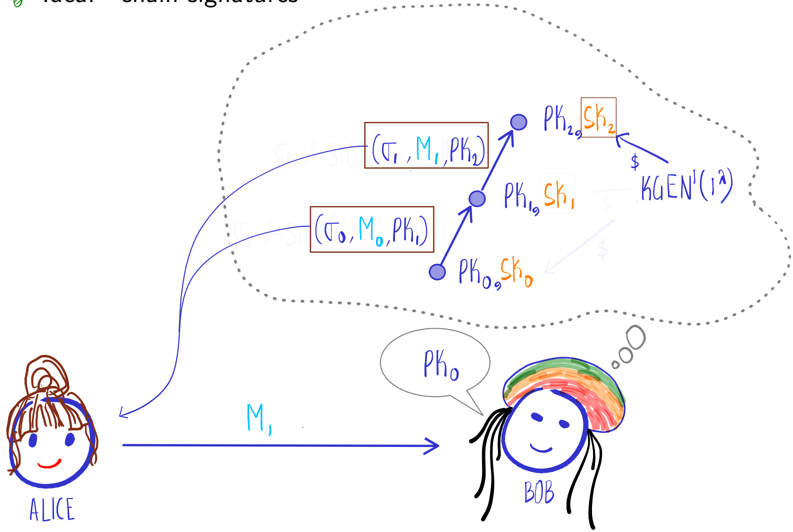
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\texttt{KGEN}^1, \texttt{SIGN}^1, \texttt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



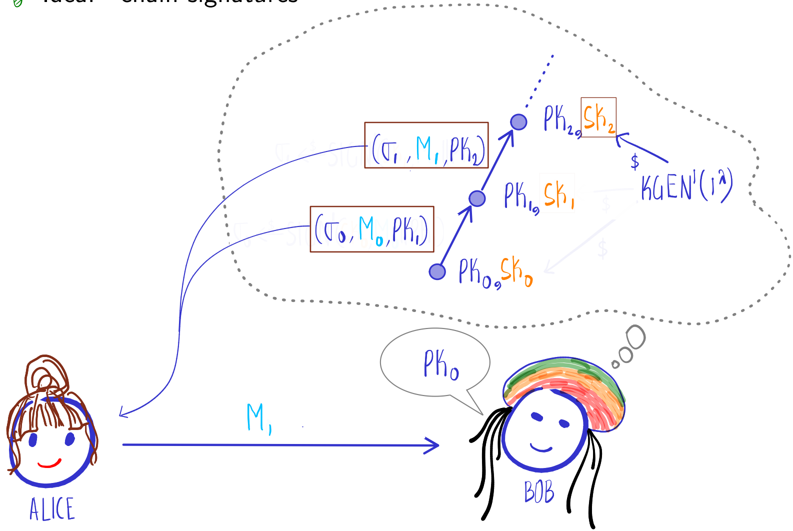
(Many-Time) Stateful Signatures...

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



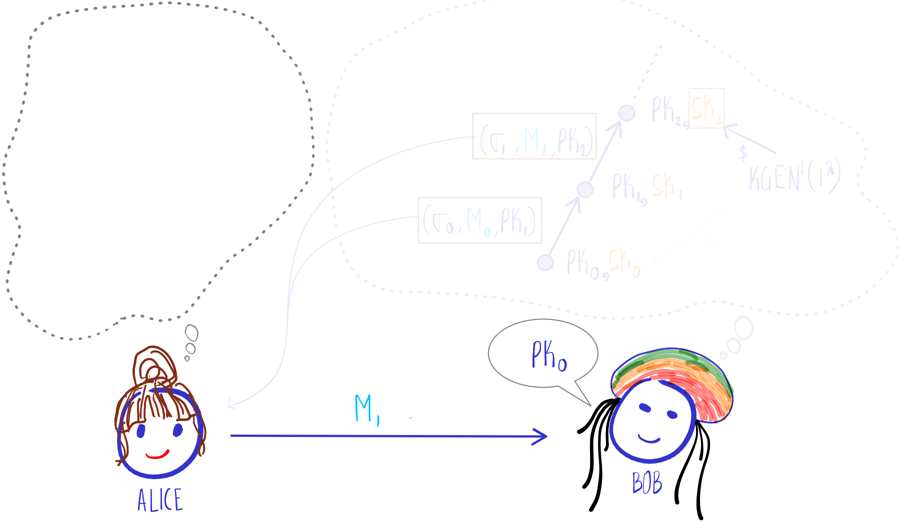
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\mathtt{KGEN}^1, \mathtt{SIGN}^1, \mathtt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



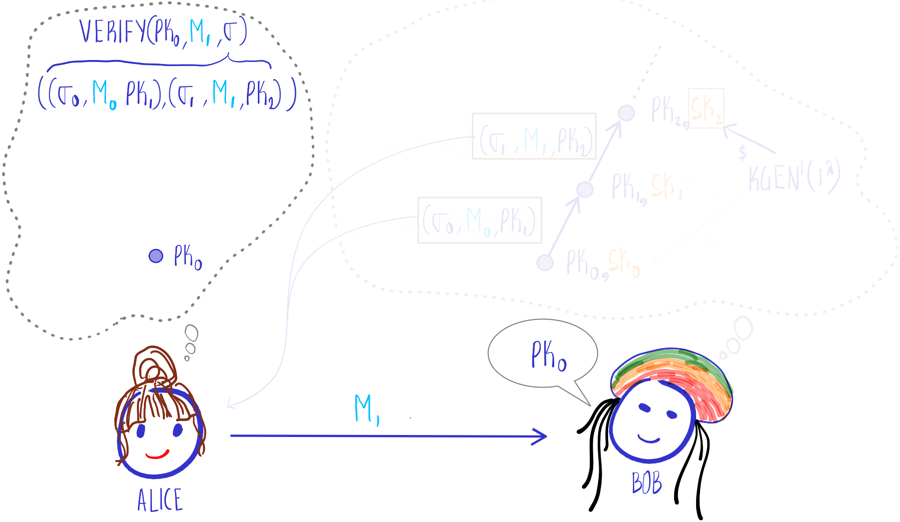
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\texttt{KGEN}^1, \texttt{SIGN}^1, \texttt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



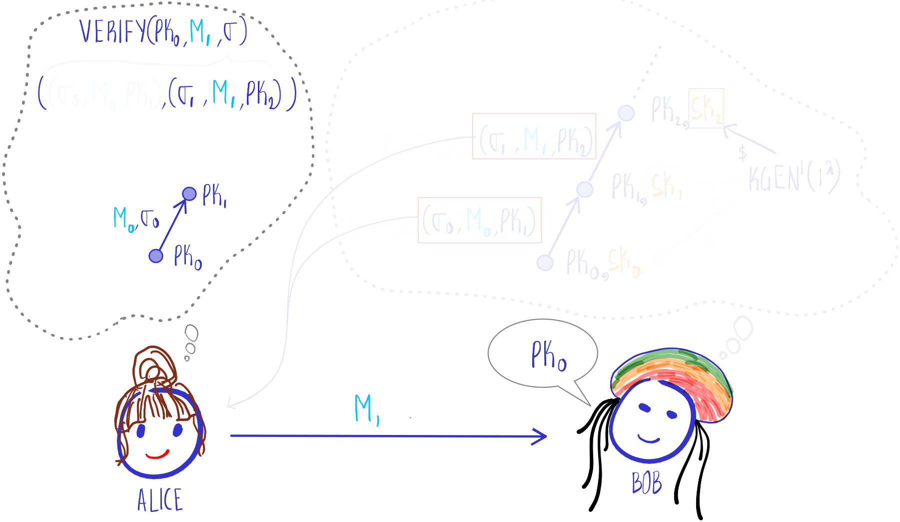
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\texttt{KGEN}^1, \texttt{SIGN}^1, \texttt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



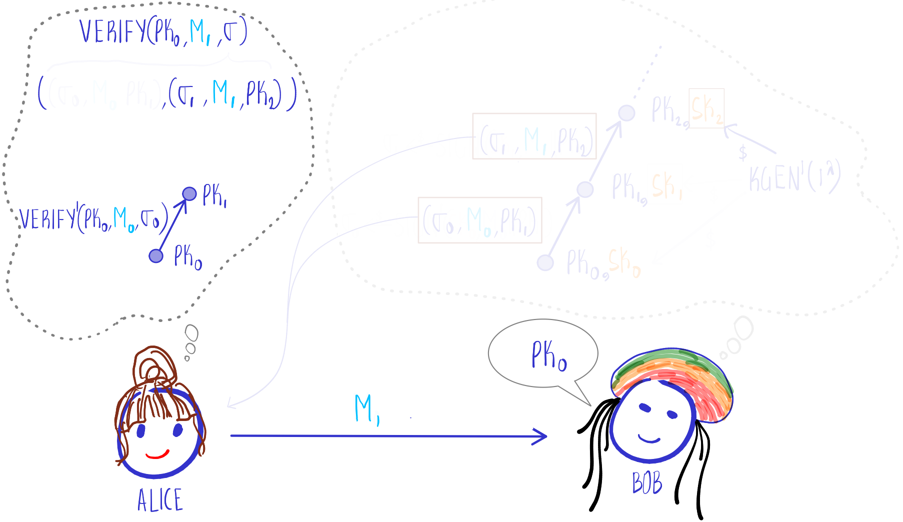
(Many-Time) Stateful Signatures...

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



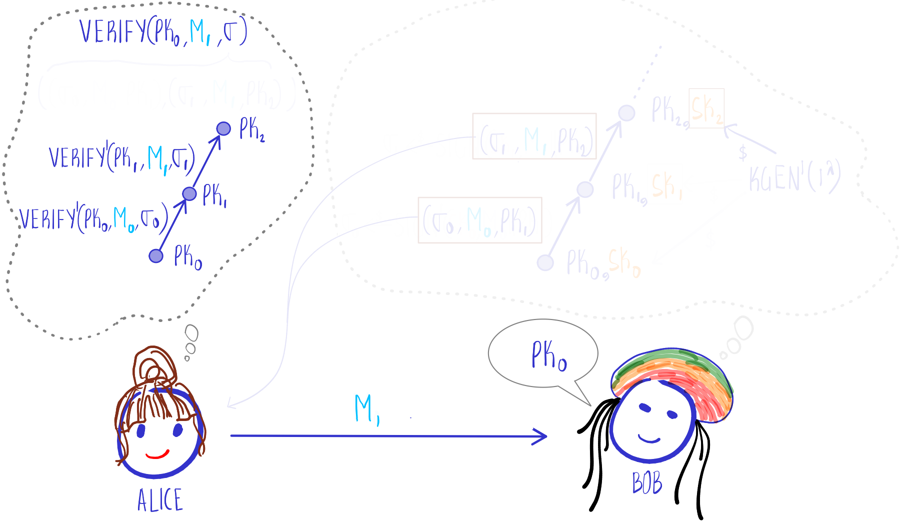
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\mathtt{KGEN}^1, \mathtt{SIGN}^1, \mathtt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



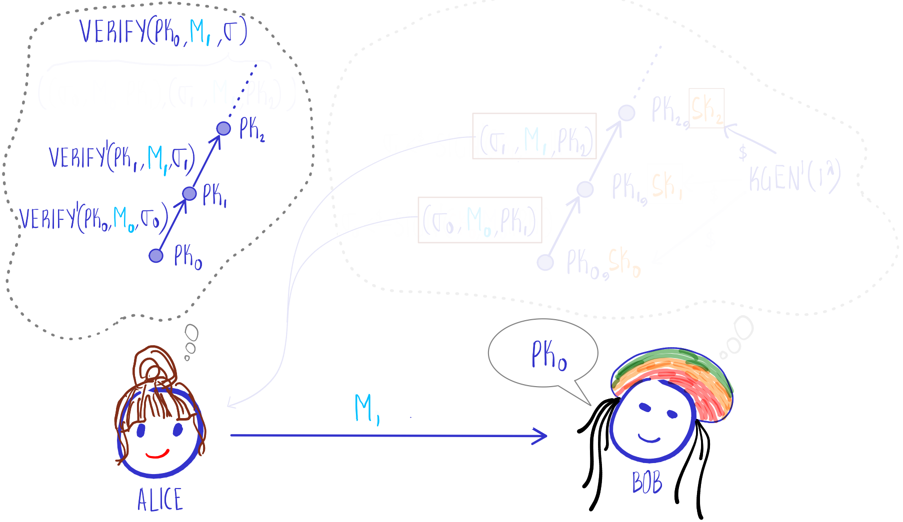
(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\mathtt{KGEN}^1, \mathtt{SIGN}^1, \mathtt{VERIFY}^1) \Rightarrow$ stateful signature Σ^s.
  - Idea: "chain signatures"



(Many-Time) Stateful Signatures...

- OTS $\Sigma^1 = (\mathtt{KGEN}^1, \mathtt{SIGN}^1, \mathtt{VERIFY}^1) \Rightarrow$ 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

$\square$

(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.

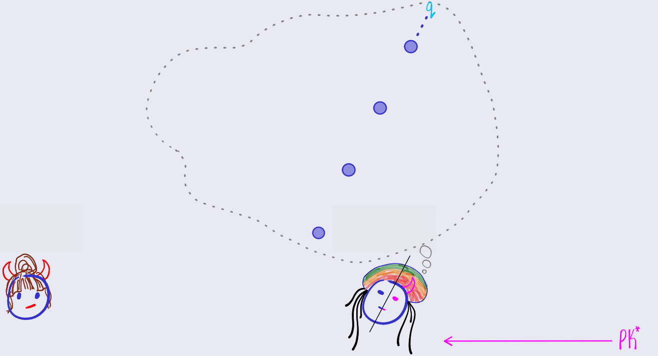
$\square$

(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.

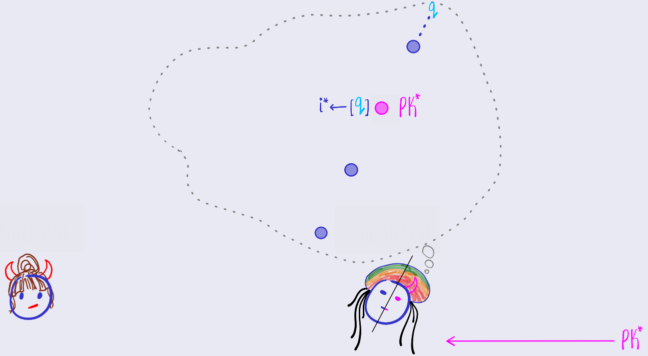


(Many-Time) Stateful Signatures...

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.



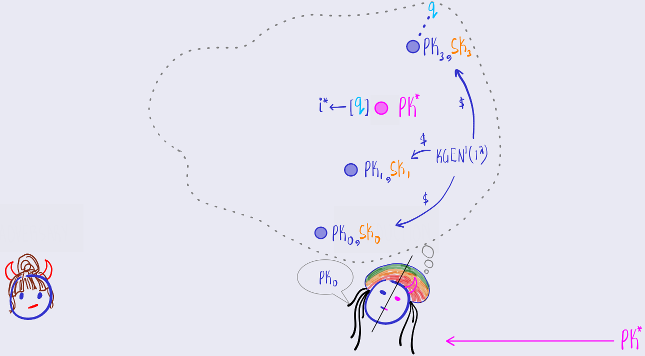
□

(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.



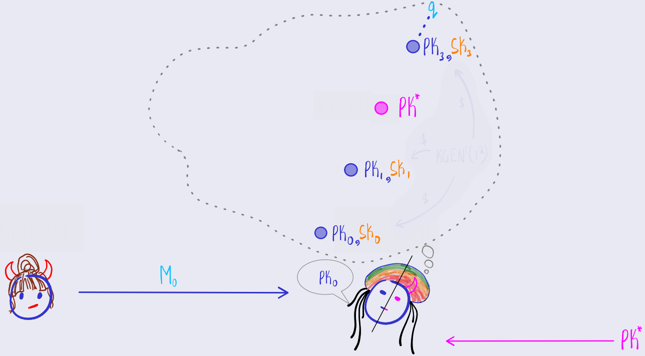
□

(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.



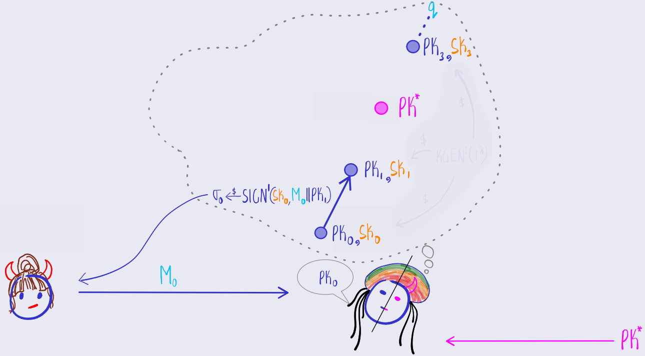
□

(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.



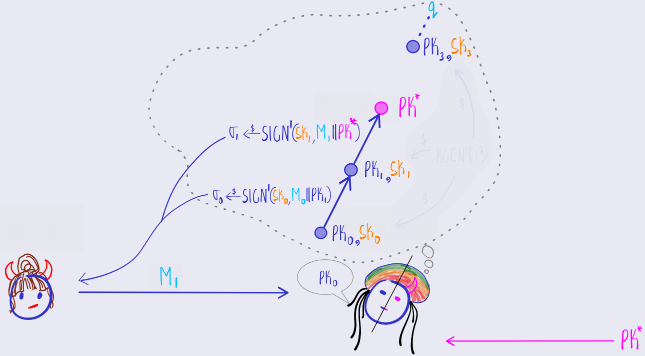
□

(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.



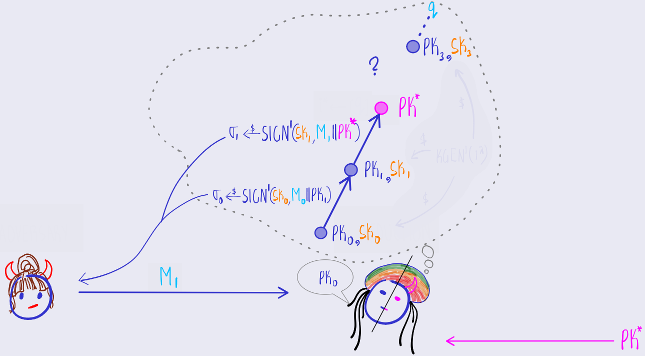
$\square$

(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.



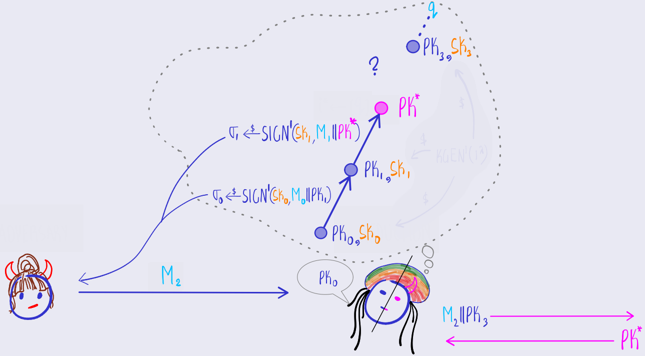
□

(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.



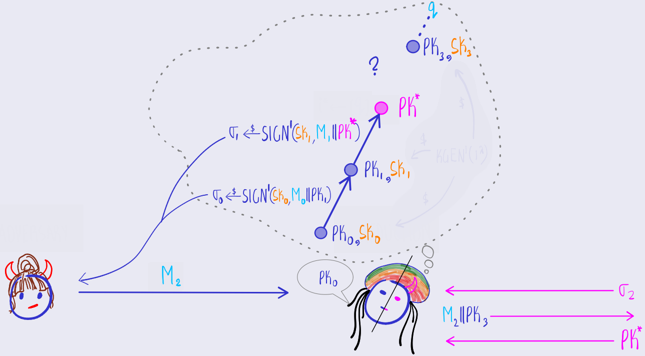
$\square$

(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.



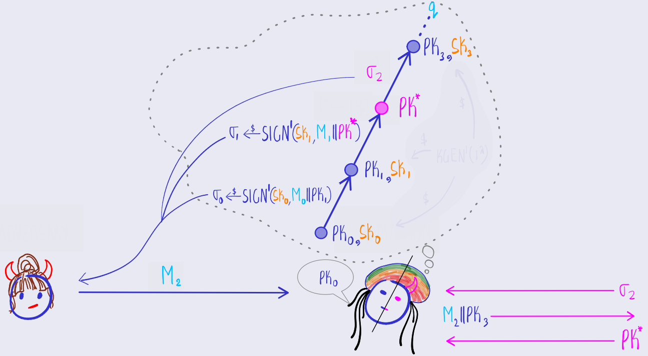
$\square$

(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.



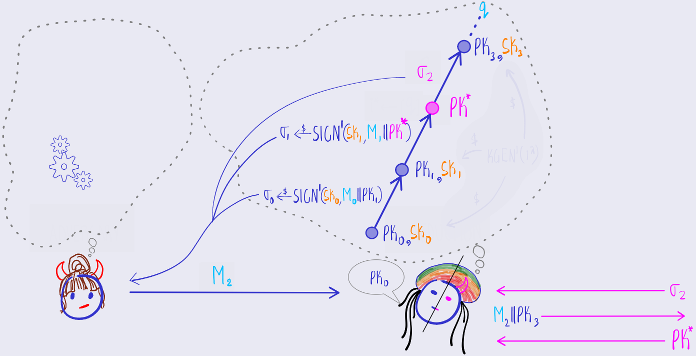
$\square$

(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.



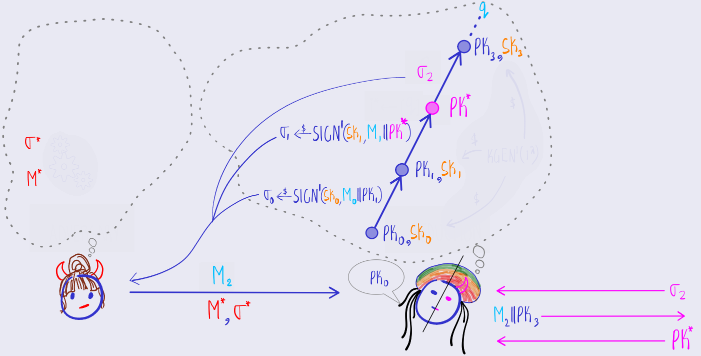
$\square$

(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.



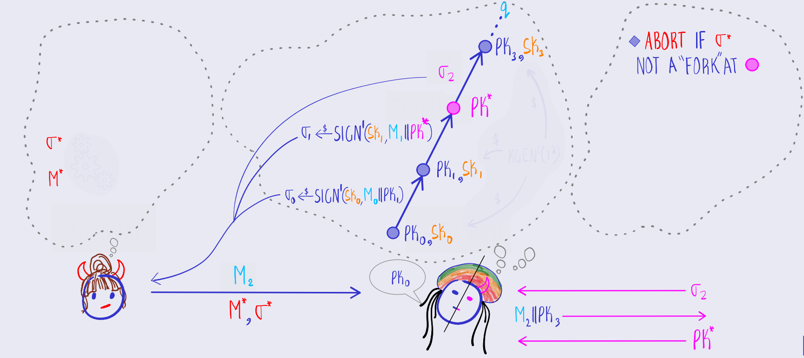
$\square$

(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.



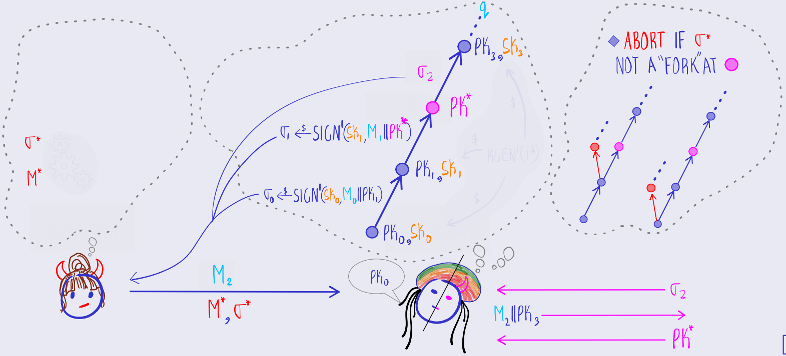
□

(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.



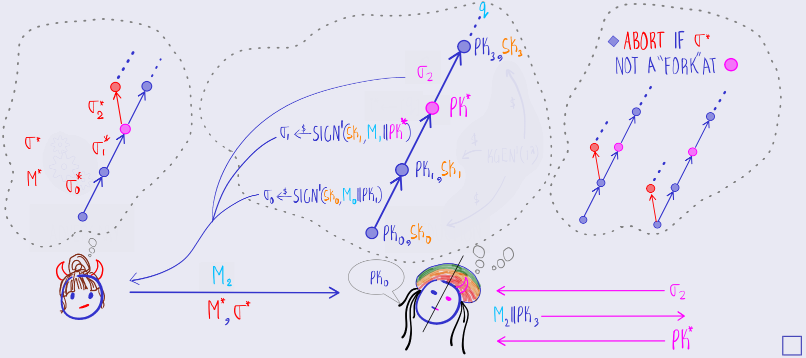
$\square$

(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.



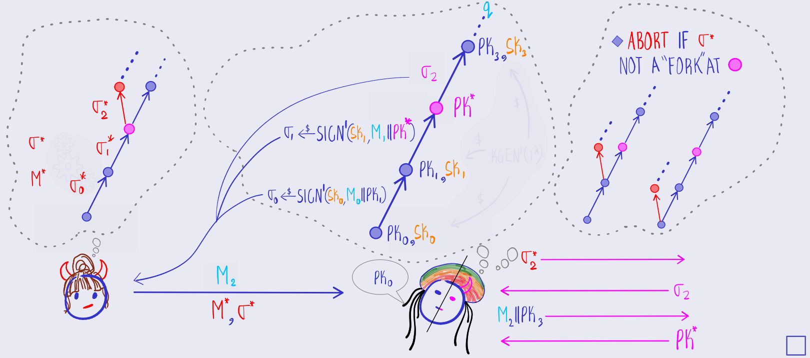
□

(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...

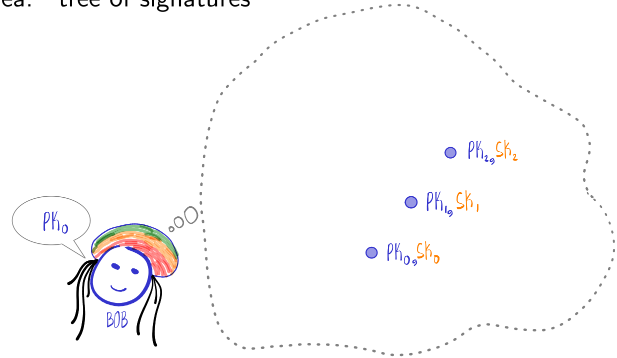
- The size of signatures in Σ^s grows linearly with the number of signatures issued by the signer. How to fix this?
  - Idea: "tree of signatures"

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"

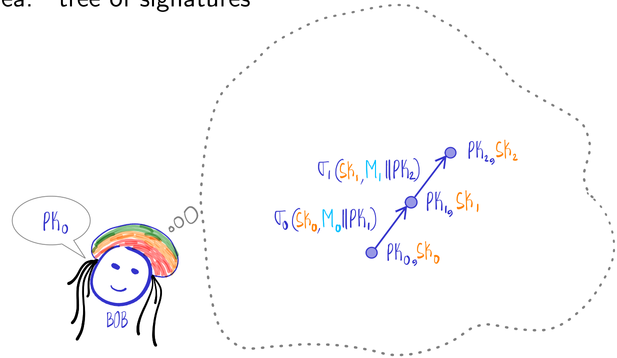


Exercise 3 (Shorter stateful signature)

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"

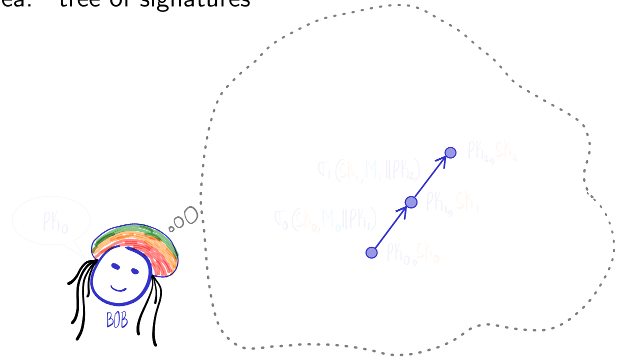


Exercise 3 (Shorter stateful signature)

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"

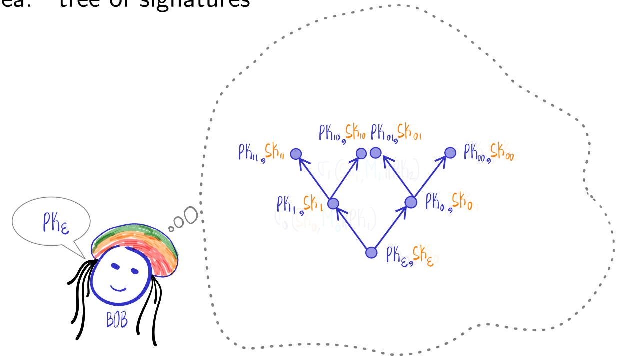


Exercise 3 (Shorter stateful signature)

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"

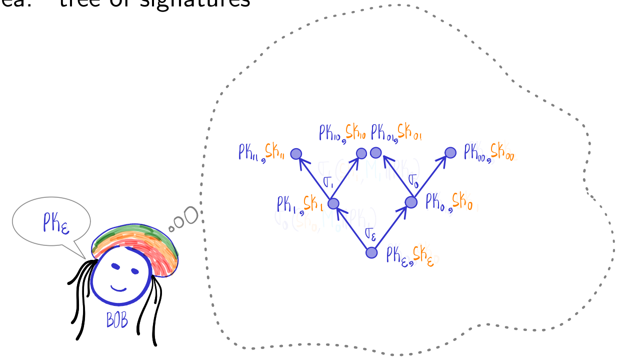


Exercise 3 (Shorter stateful signature)

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"

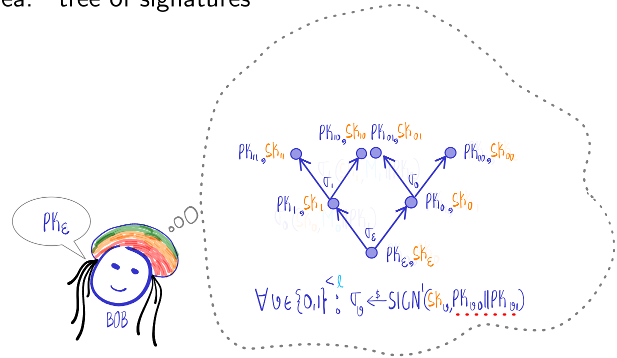


Exercise 3 (Shorter stateful signature)

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"

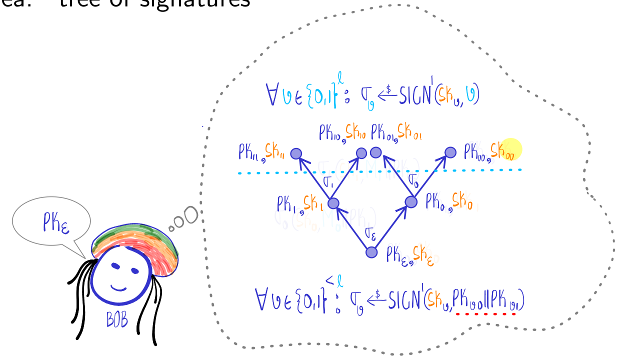


Exercise 3 (Shorter stateful signature)

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"

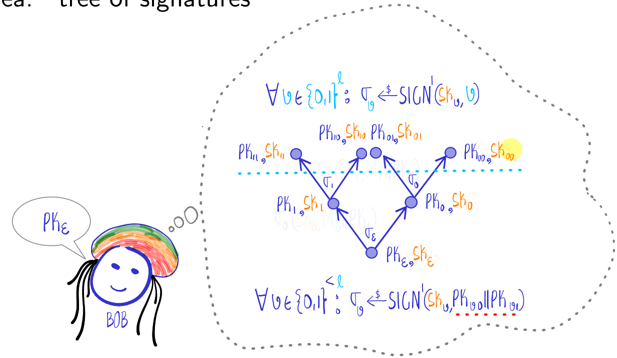


Exercise 3 (Shorter stateful signature)

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) Signatures...

- Short stateful signature Σ^{ss} + pseudo-random function $F_{\mathtt{K}} : \{0,1\}^{\ell+1} \to \{0,1\}^{\lambda} \Rightarrow$ signature Σ

(Many-Time) Signatures...

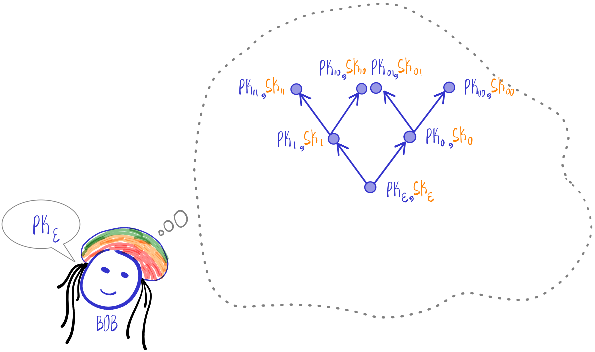
- Short stateful signature Σ^{ss} + pseudo-random function $F_{\mathtt{K}} : \{0,1\}^{\ell+1} \to \{0,1\}^{\lambda} \Rightarrow$ signature Σ
  - Idea: Use to *Derandomise* OTS signature and key gen.

Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

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

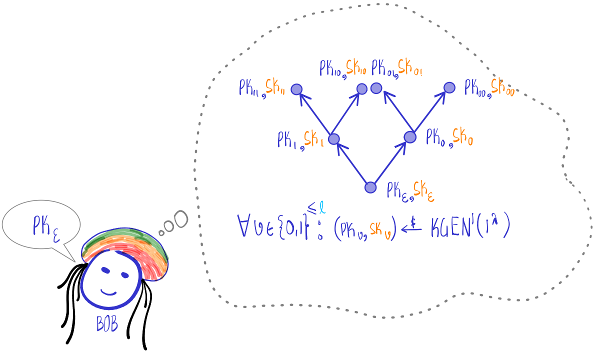


Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

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

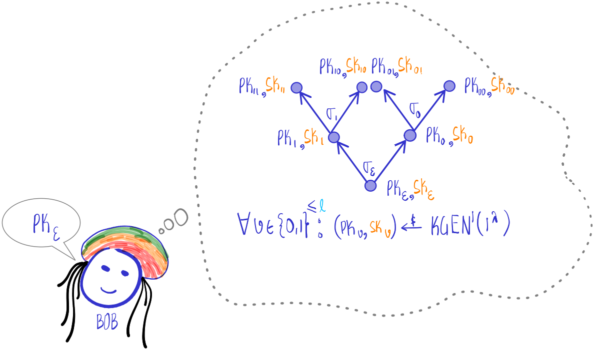


Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

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

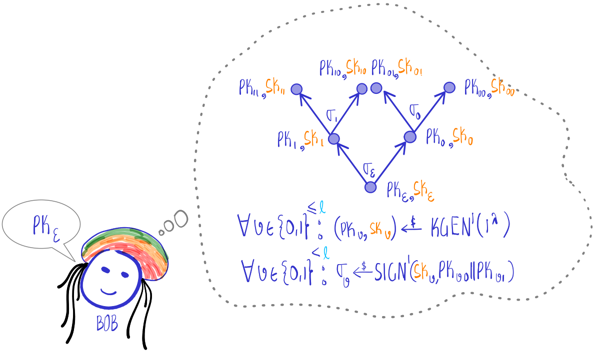


Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

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

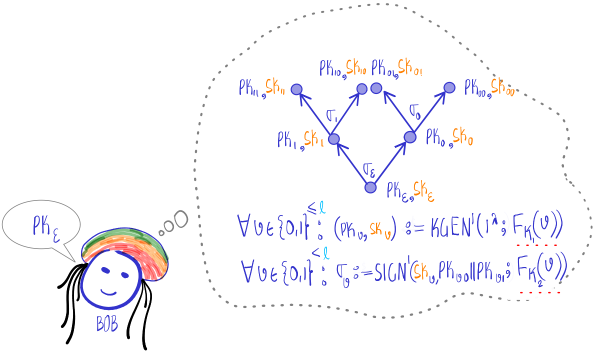


Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

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

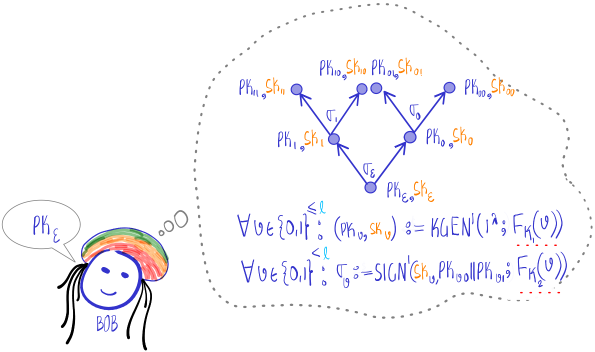


Exercise 4 (EU-CMA signature)

Prove that Σ is secure.

(Many-Time) Signatures...

- Short stateful signature Σ^{ss} + pseudo-random function $F_K : \{0,1\}^{\ell+1} \to \{0,1\}^{\lambda} \Rightarrow$ 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...

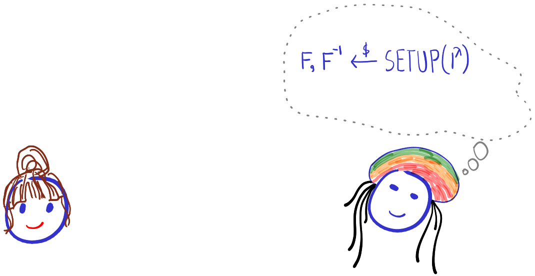
- Efficient schemes under stronger hardness assumptions
- Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \to \mathcal{D}$

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...

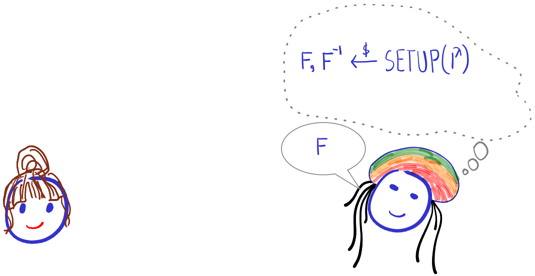
- Efficient schemes under stronger hardness assumptions
- Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \to \mathcal{D}$
  - Syntax:



  - 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...

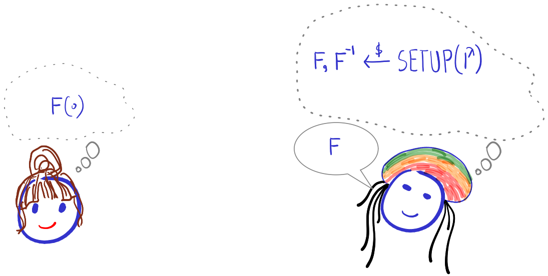
- Efficient schemes under stronger hardness assumptions
- Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \to \mathcal{D}$
    - Syntax:



    - 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...

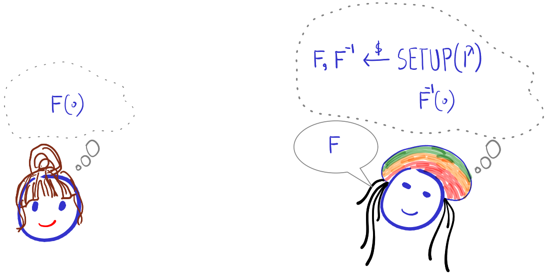
- Efficient schemes under stronger hardness assumptions
- Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \to \mathcal{D}$
  - Syntax:



  - 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...

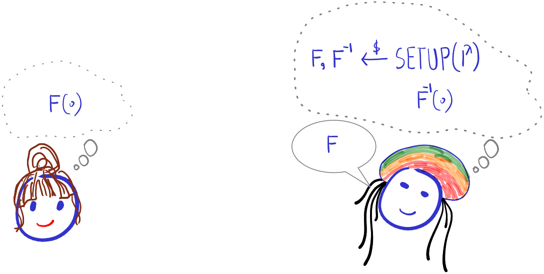
- Efficient schemes under stronger hardness assumptions
- Trapdoor (one-way) permutation $F, F^{-1} : \mathcal{D} \to \mathcal{D}$
  - Syntax:



  - 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 $F, F^{-1} : \mathcal{D} \to \mathcal{D}$
  - 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

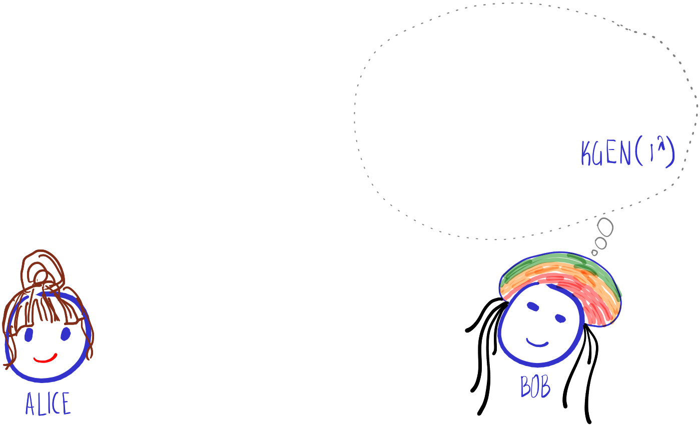
Efficient Signatures via Hash-then-Invert...

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

- Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

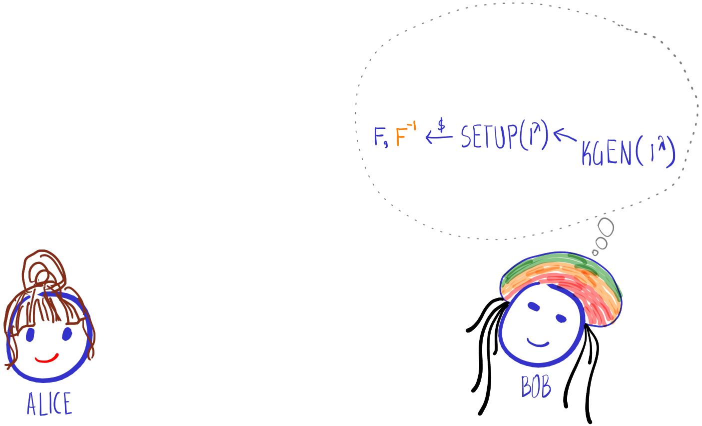
- TDP (F, F^{-1}) over domain $\mathcal{D}$ + hash function $H : \{0,1\}^* \to \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0,1\}^*$



- Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

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



- Efficient: compact public key and short signatures

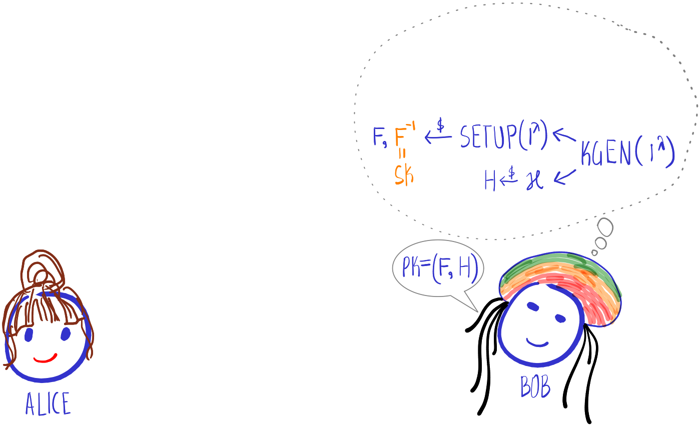
Efficient Signatures via Hash-then-Invert...

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

- Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

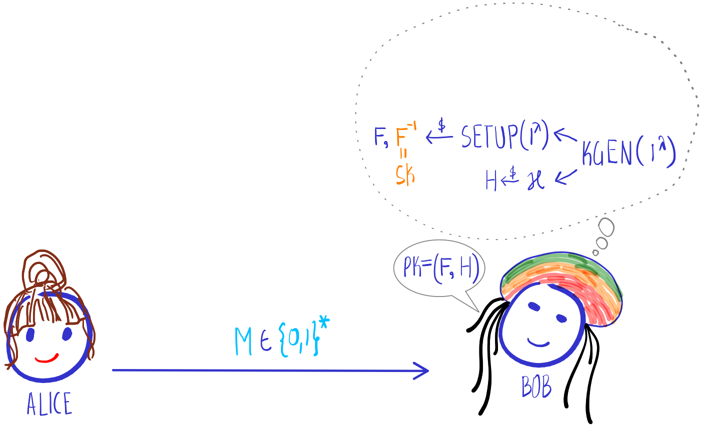
- TDP (F, F^{-1}) over domain $\mathcal{D}$ + hash function $H : \{0,1\}^* \to \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0,1\}^*$



- Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

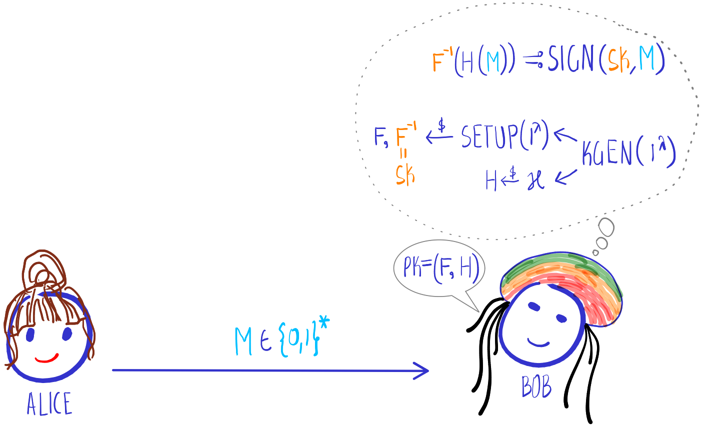
- TDP (F, F^{-1}) over domain $\mathcal{D}$ + hash function $H : \{0,1\}^* \to \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0,1\}^*$



- Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

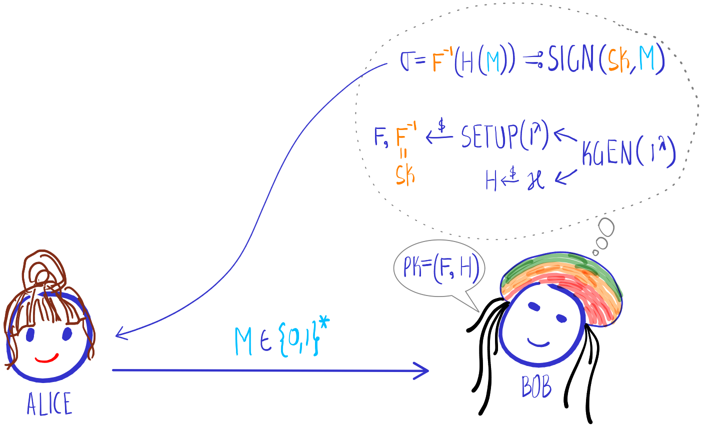
- TDP (F, F^{-1}) over domain $\mathcal{D}$ + hash function $H : \{0,1\}^* \to \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0,1\}^*$



- Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

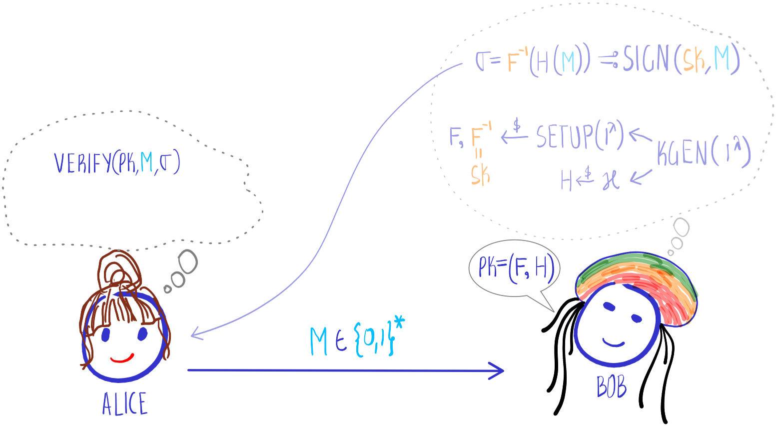
- TDP (F, F^{-1}) over domain $\mathcal{D}$ + hash function $H : \{0,1\}^* \to \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0,1\}^*$



- Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

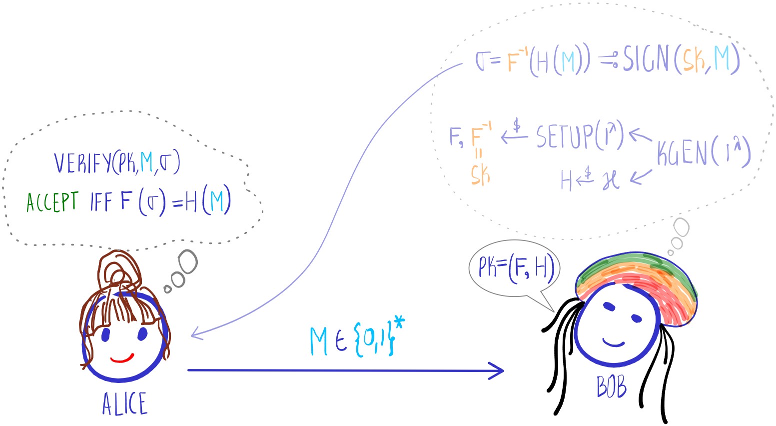
- TDP (F, F^{-1}) over domain $\mathcal{D}$ + hash function $H : \{0,1\}^* \to \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0,1\}^*$



- Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

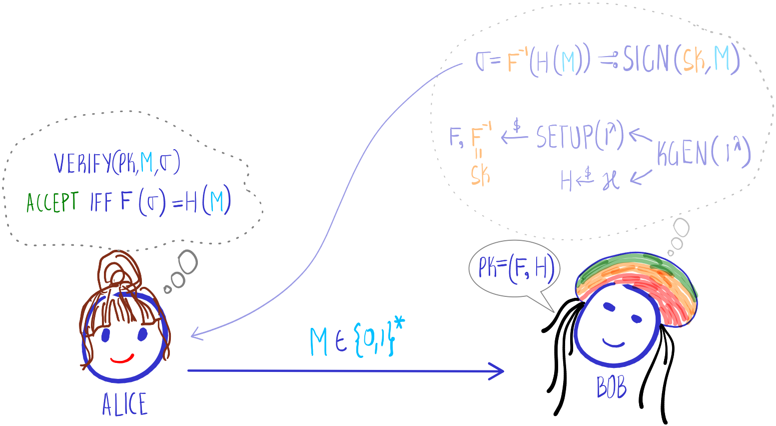
- TDP (F, F^{-1}) over domain $\mathcal{D}$ + hash function $H : \{0,1\}^* \to \mathcal{D} \Rightarrow$ signature Σ for $\mathcal{M} := \{0,1\}^*$



- Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

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



- Efficient: compact public key and short signatures

Efficient Signatures via Hash-then-Invert...

Theorem 7

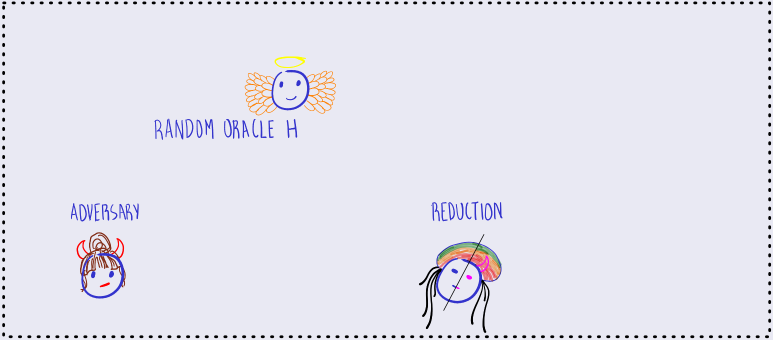
If (F, F^{-1}) is a TDP and H is a random oracle then Σ is secure.

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.



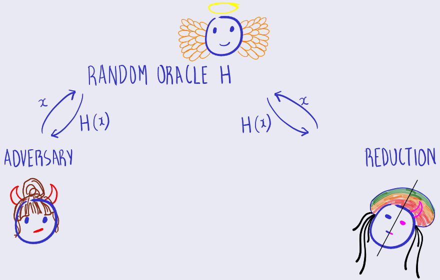
□

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.



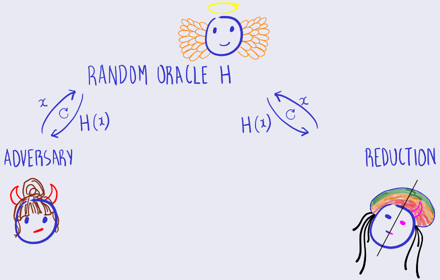
$\square$

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.



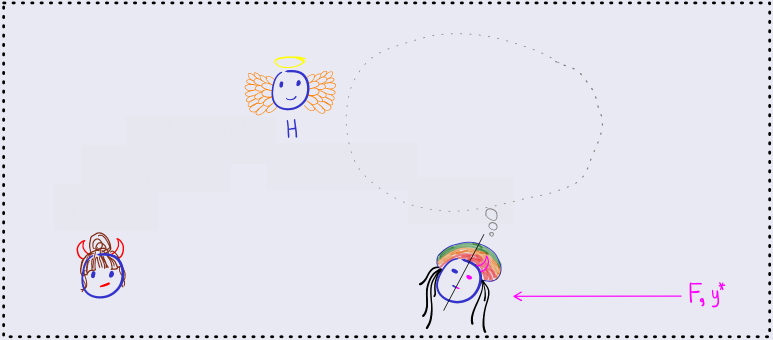
$\square$

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.



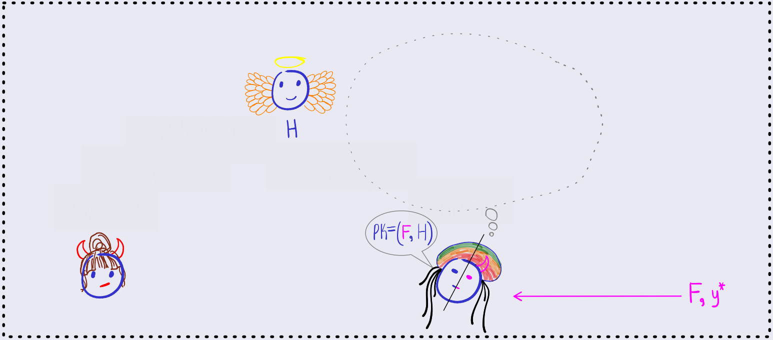
$\square$

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.



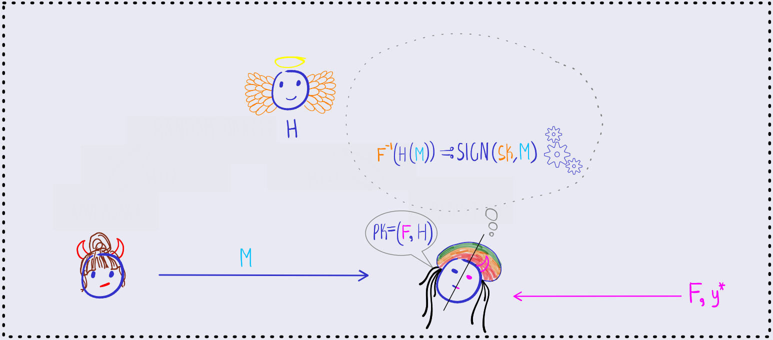
$\square$

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.



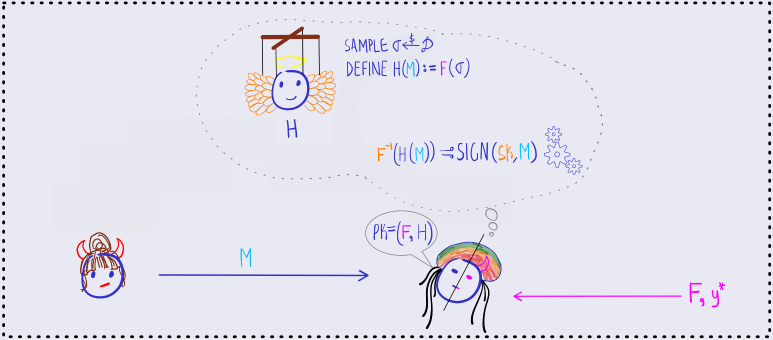
$\square$

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.



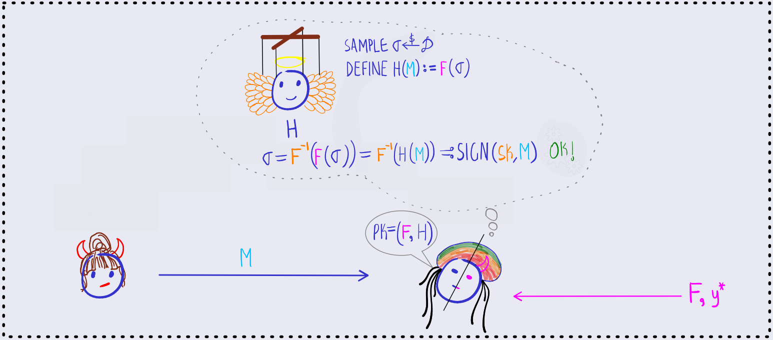
$\square$

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.



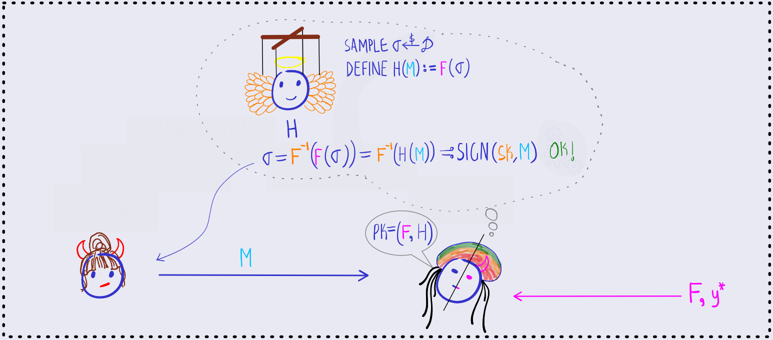
$\square$

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.



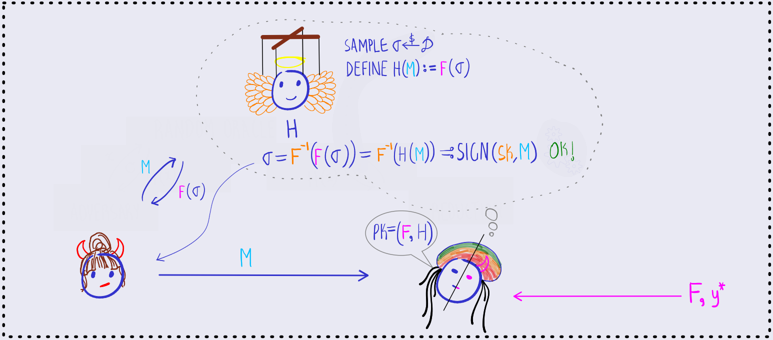
□

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.



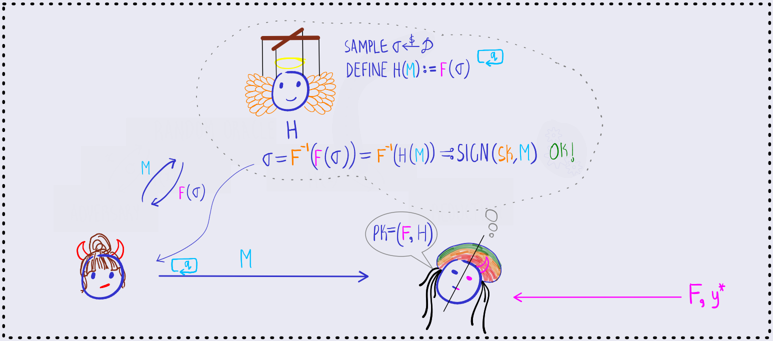
□

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.



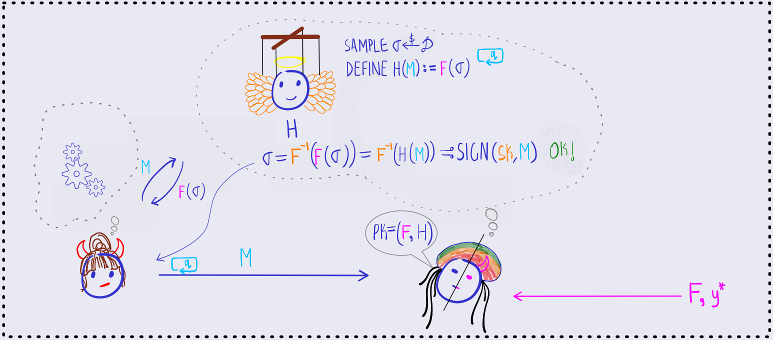
$\square$

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.



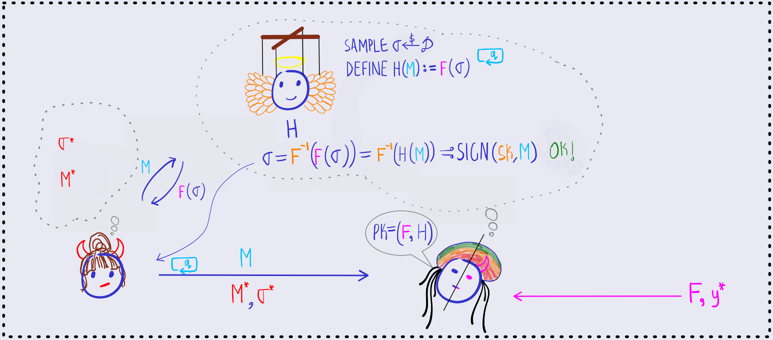
$\square$

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.



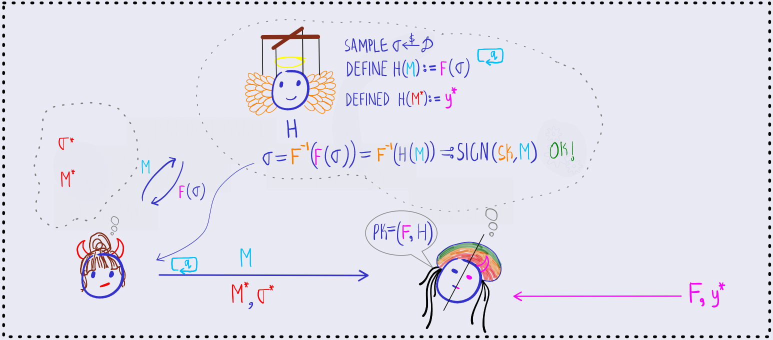
□

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.

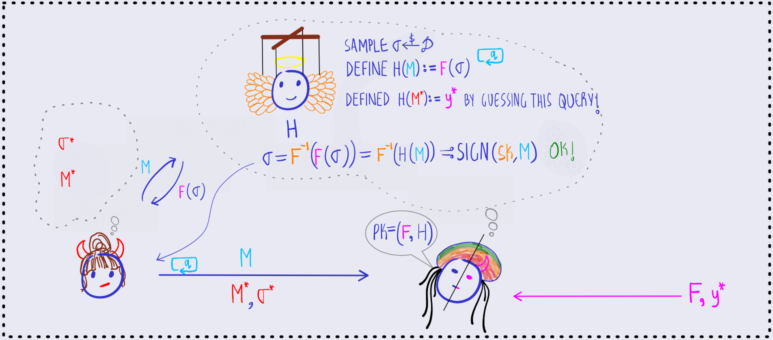


Efficient Signatures via Hash-then-Invert...

Theorem 7

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

Proof sketch: random oracle programming.



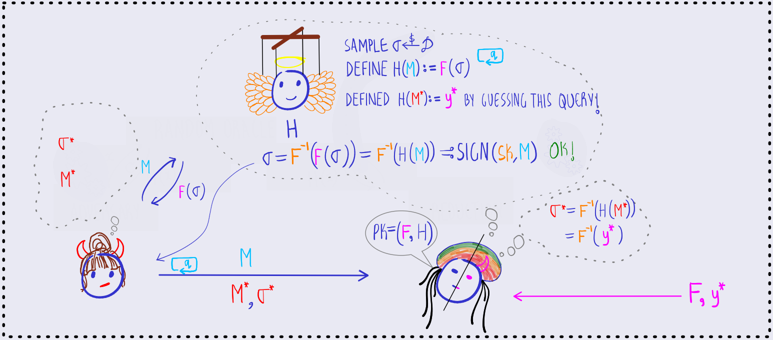
$\square$

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.



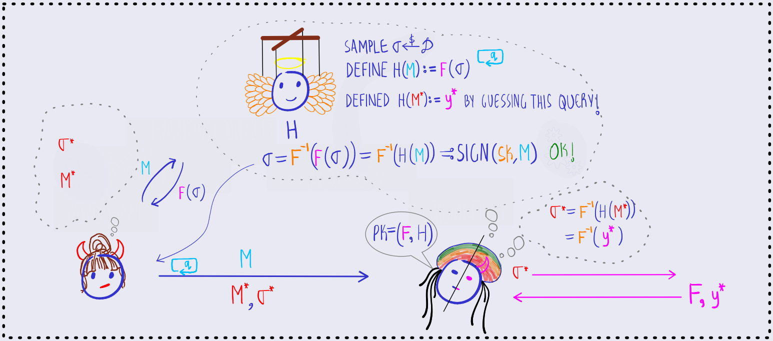
□

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

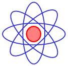
Many-Time (Stateful) Signatures

Efficient Signatures via Hash-and-Sign

Wrapping Up

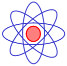
To Recap

- 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

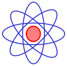
- Constructions:
  1. Theoretical construction of signature using OWF and CRHF
     - CRHF can be replaced with *universal one-way* hash function (UOWHF), which can be constructed from OWF
     - Can be shown to be quantum secure



  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

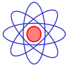


  2. Efficient constructions in ROM
     - RSA-PSS based on Hash-then-Invert
     - Other approach: Fiat-Shamir Transform (e.g., Schnorr)

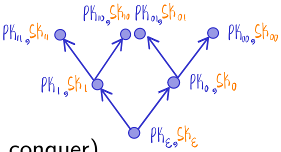
- Takeaways:

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



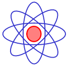
  2. Efficient constructions in ROM
     - RSA-PSS based on Hash-then-Invert
     - Other approach: Fiat-Shamir Transform (e.g., Schnorr)



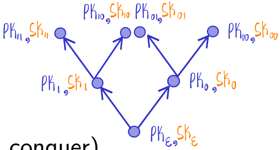
- Takeaways:
  - Constructive:
    - Bottom up constructive approach
    - Tree-based construction (divide and conquer)

To Recap

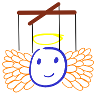
- Constructions:
  1. Theoretical construction of signature using OWF and CRHF
     - CRHF can be replaced with *universal one-way* hash function (UOWHF), which can be constructed from OWF
     - Can be shown to be quantum secure



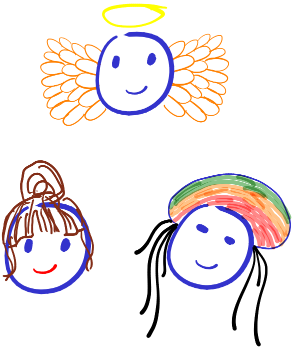
  2. Efficient constructions in ROM
     - RSA-PSS based on Hash-then-Invert
     - Other approach: Fiat-Shamir Transform (e.g., Schnorr)



- 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.