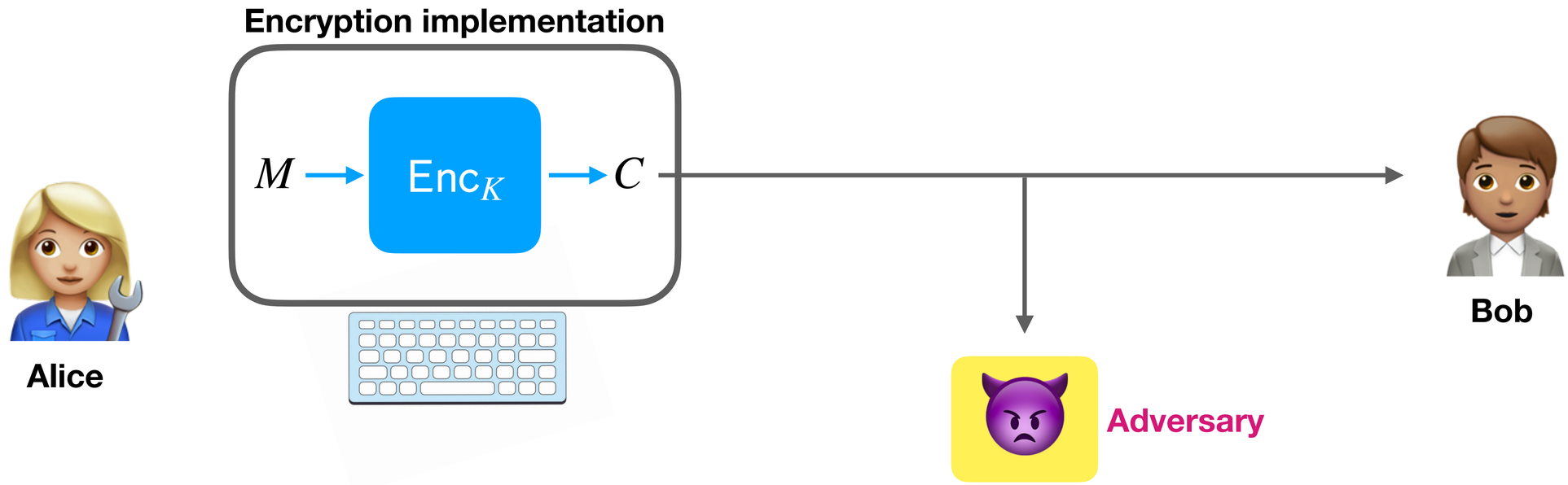
Public-Algorithm Substitution Attacks: Subverting Hashing and Verification

Mihir Bellare (UCSD)
Doreen Riepel (CISPA)
Laura Shea (UCSD)

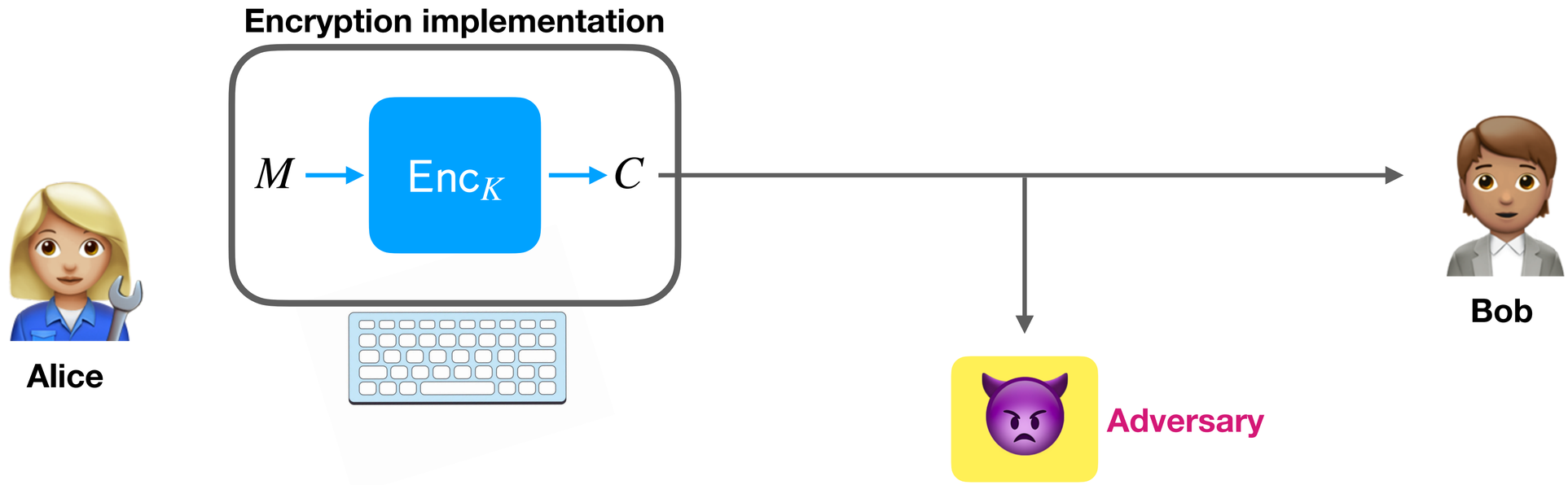
PKC 2025

ePrint 2024 / 536

The textbook view of cryptography, illustrated for symmetric encryption:



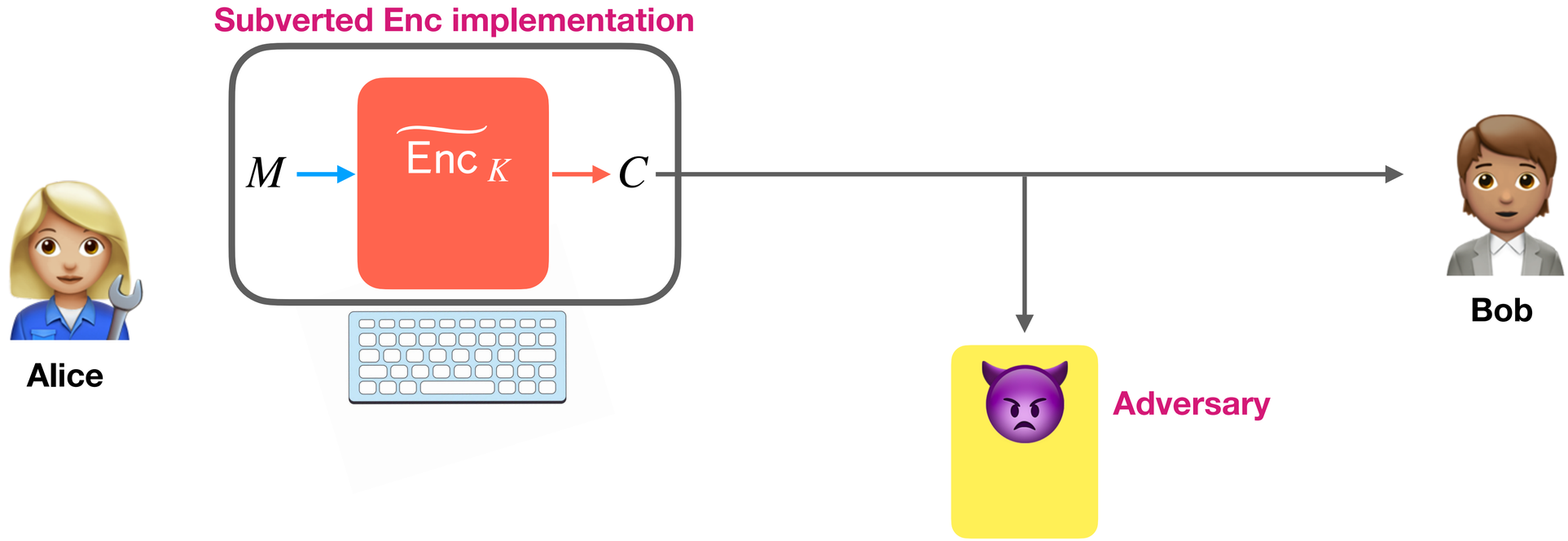
The textbook view of cryptography, illustrated for symmetric encryption:



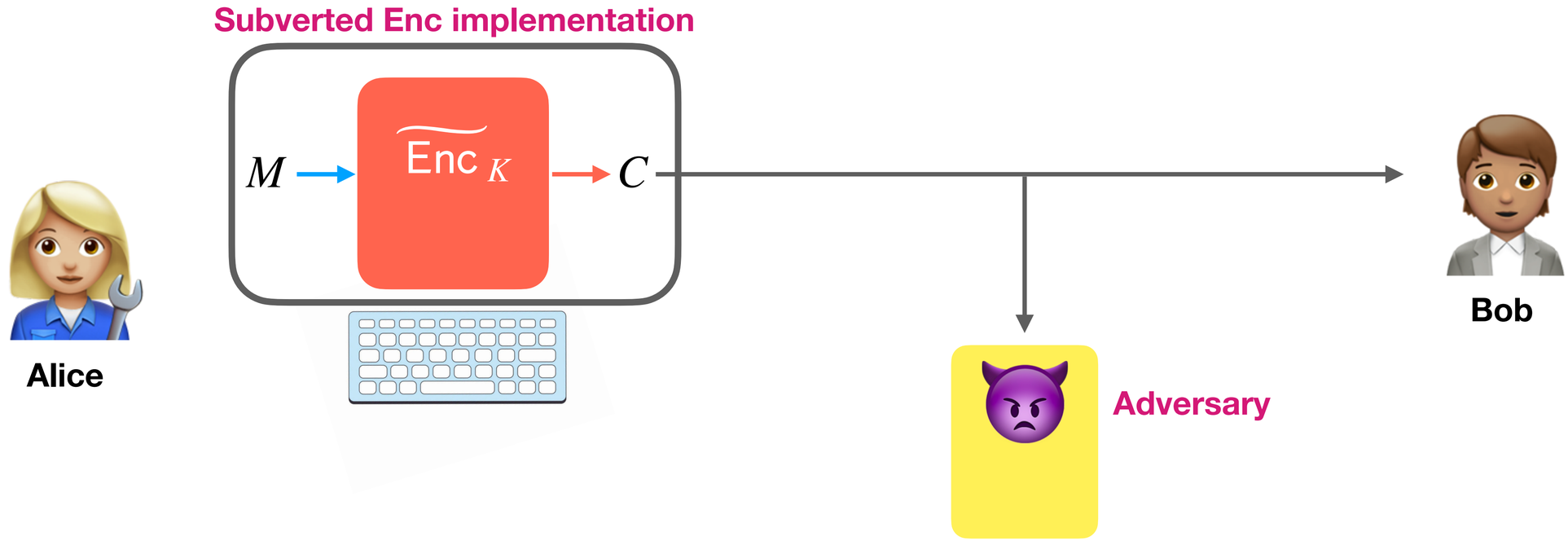
Here the encryption algorithm Enc_K is assumed to be CORRECTLY and HONESTLY implemented.

Our usual definitions, like IND-CPA, IND-CCA, AEAD, …
are all in this setting.

Cryptography, subverted !

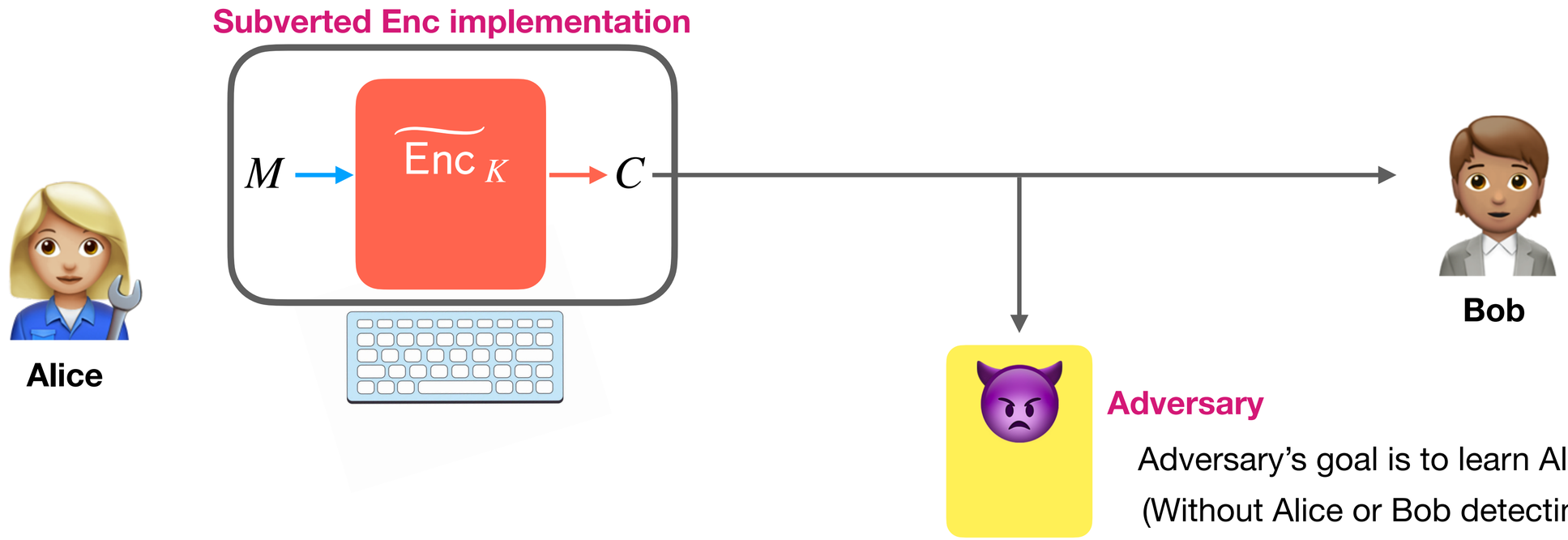


Cryptography, subverted !



Here the honest encryption algorithm Enc_K has been replaced by a malicious $\widetilde{\mathrm{Enc}}_K$.

Cryptography, subverted !

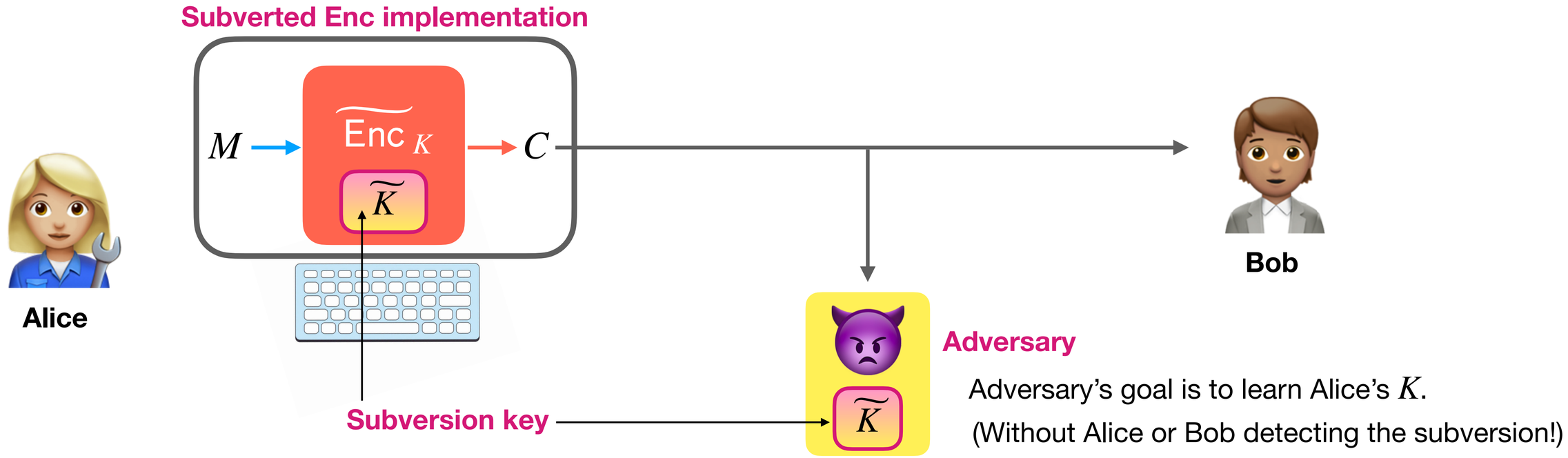


Adversary's goal is to learn Alice's K.

(Without Alice or Bob detecting the subversion!)

Here the honest encryption algorithm Enc_K has been replaced by a malicious $\widetilde{\mathrm{Enc}}_K$.

Cryptography, subverted !

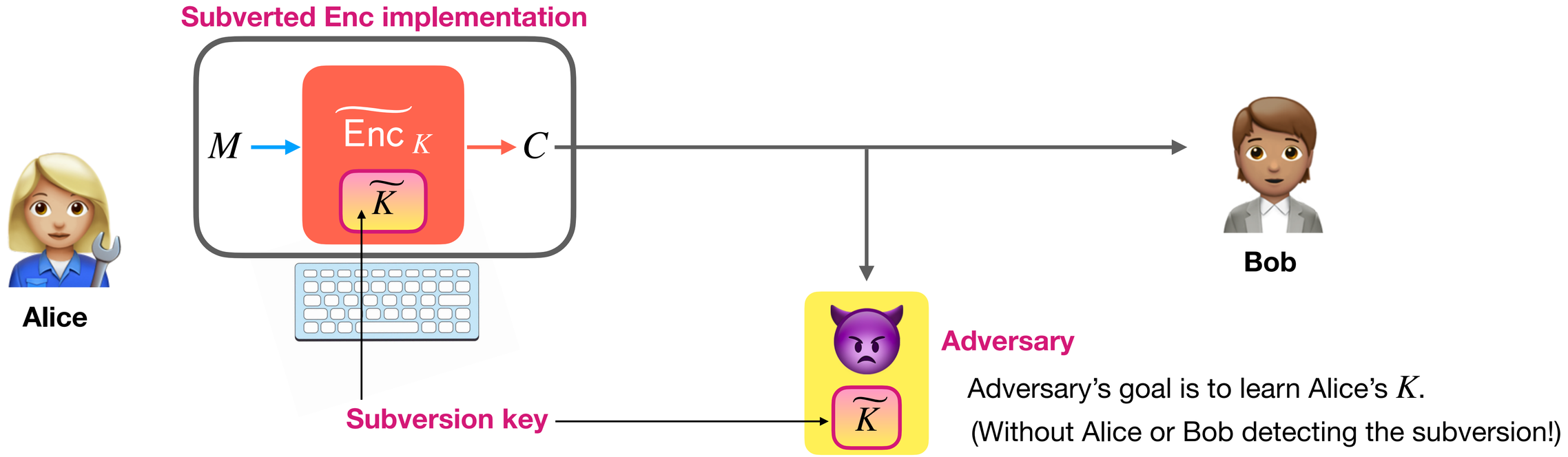


Adversary's goal is to learn Alice's K.

(Without Alice or Bob detecting the subversion!)

Here the honest encryption algorithm Enc_K has been replaced by a malicious $\widetilde{\mathsf{Enc}}_K$.

Cryptography, subverted !



Adversary's goal is to learn Alice's K.

(Without Alice or Bob detecting the subversion!)

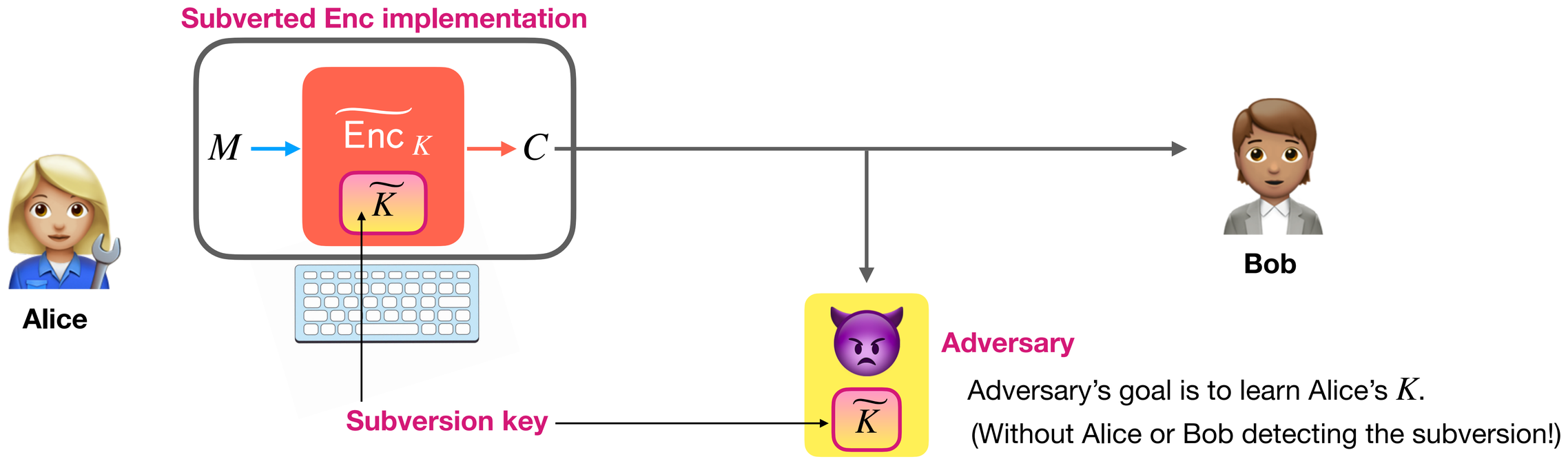
Kleptography [YY96]

The Dark Side of "Black-Box" Cryptography
or: Should We Trust Capstone?

Adam Young* and Moti Yung**

First (academic) suggestion of this category of attack.

Cryptography, subverted !



Kleptography [YY96]

Snowden [2013]

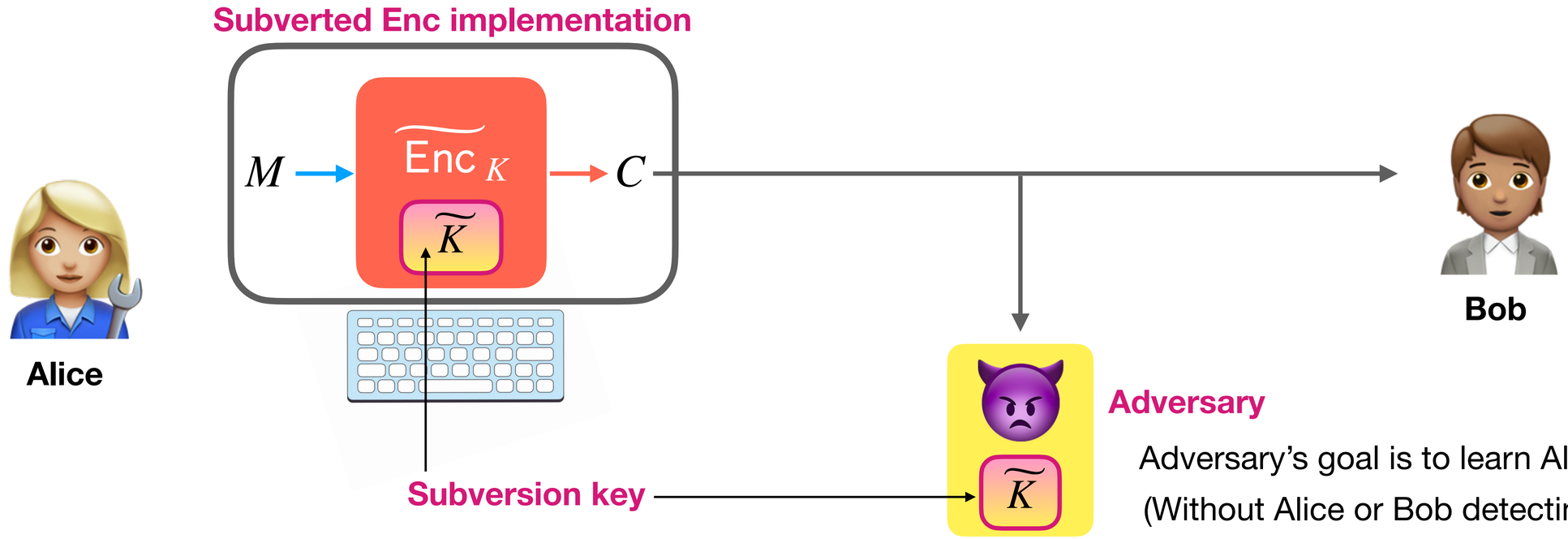
The Dark Side of "Black-Box" Cryptography
or: Should We Trust Capstone?

Adam Young* and Moti Yung**

First (academic) suggestion of this category of attack.

Motivation to look at powerful institutional adversaries and surveillance methods.

Cryptography, subverted !



Adversary's goal is to learn Alice's K.

(Without Alice or Bob detecting the subversion!)

Kleptography [YY96]

The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone?

Adam Young* and Moti Yung**

First (academic) suggestion of this category of attack.

Snowden [2013]

Motivation to look at powerful institutional adversaries and surveillance methods.

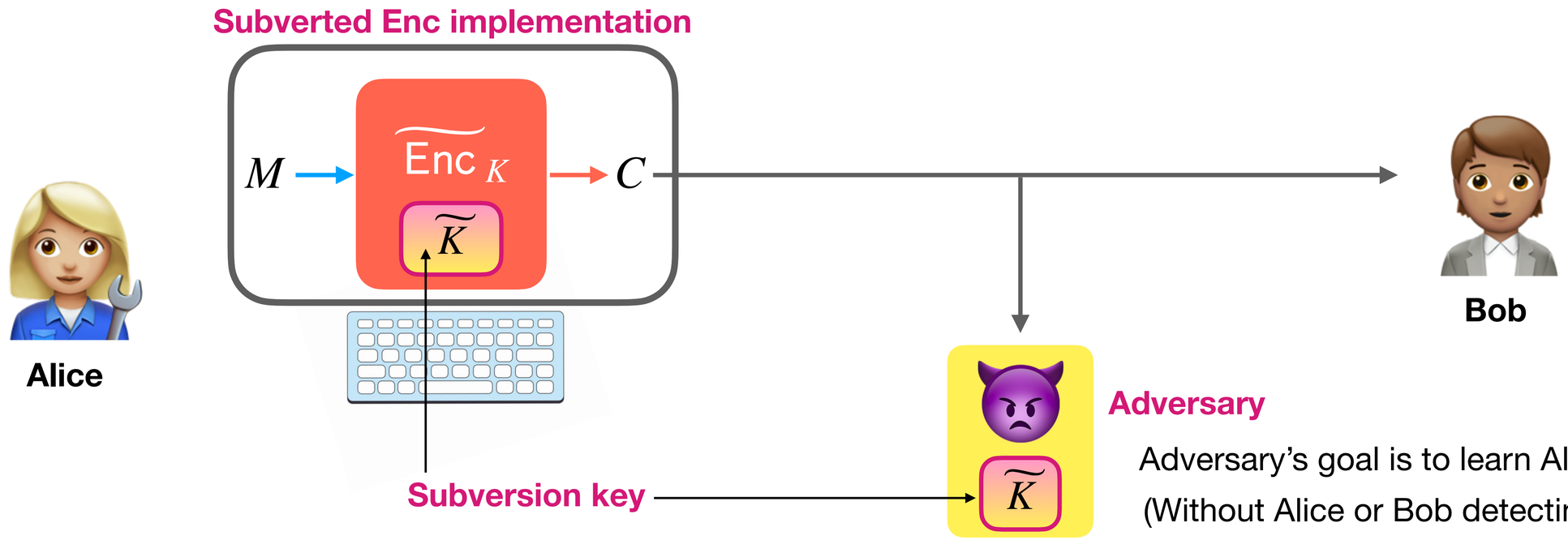
Algorithm Substitution Attack **[BPR14]**

Security of Symmetric Encryption against Mass Surveillance

Mihir Bellare[1], Kenneth G. Paterson[2], and Phillip Rogaway[3]

A new formalism in response, with many extensions to follow…

Cryptography, subverted !



BPR14 Defined two critical properties:

Algorithm Substitution Attack **[BPR14]**

(1) Exfiltration

The adversary successfully learns Alice's secret K.

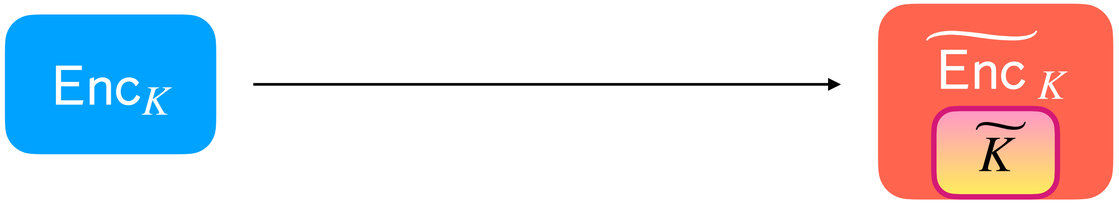
(2) Undetectability

Alice and Bob cannot tell that the subversion occurred.

Security of Symmetric Encryption against Mass Surveillance

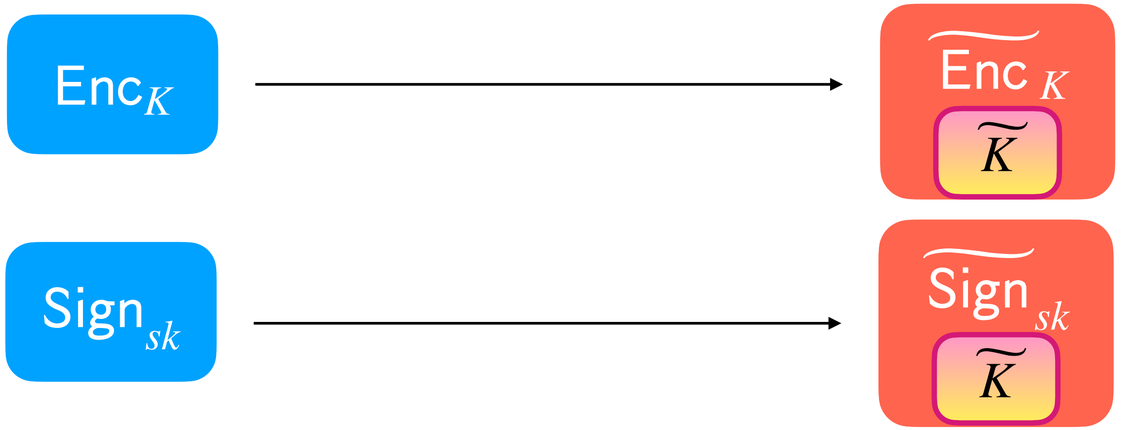
Mihir Bellare[1], Kenneth G. Paterson[2], and Phillip Rogaway[3]

Algorithm Substitution Attack (ASA)



Symmetric encryption [BPR14, BJK15, DFP15, …]

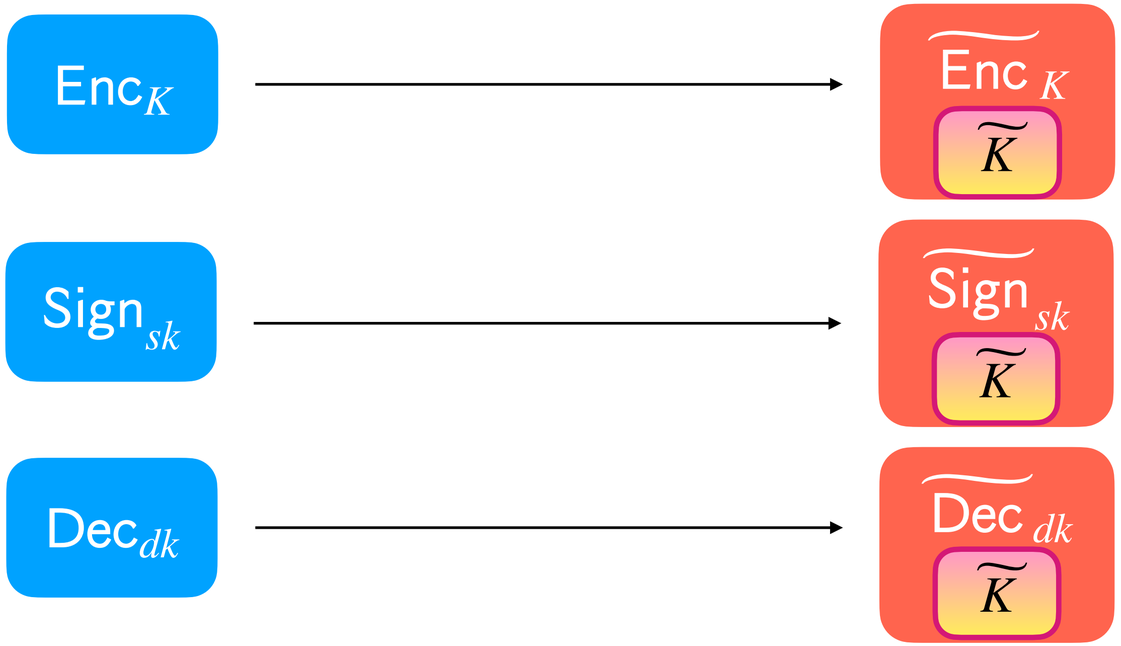
Algorithm Substitution Attack (ASA)



Symmetric encryption [BPR14, BJK15, DFP15, …]

Signing [AMV15, CS03, TBEL21, …]

Algorithm Substitution Attack (ASA)



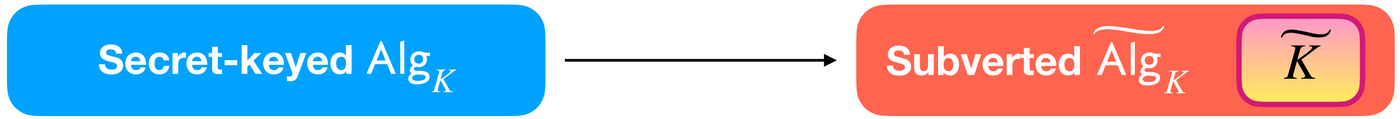
Symmetric encryption [BPR14, BJK15, DFP15, …]

Signing [AMV15, CS03, TBEL21, …]

Decryption [CHY20, AP19, JLW25, …]

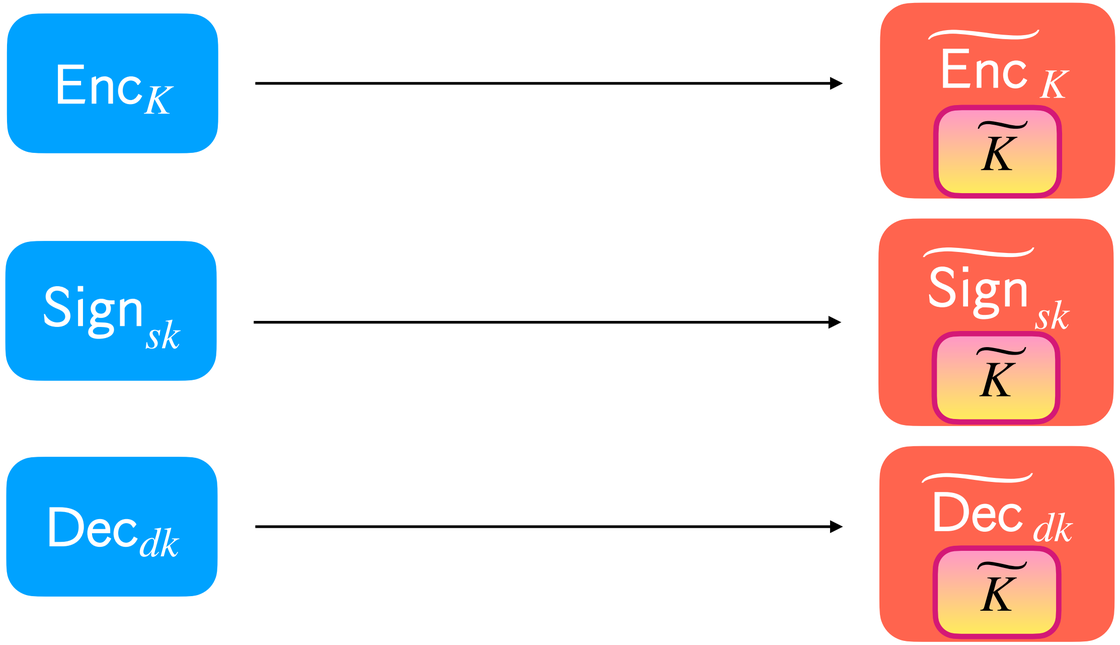
Protocols and more! [BWP+22, GBPG03, …]

Algorithm Substitution Attack (ASA)



Alice's *secret-keyed* algorithm subverted…

…with the Adversary's goal to learn K.



Symmetric encryption [BPR14, BJK15, DFP15, …]

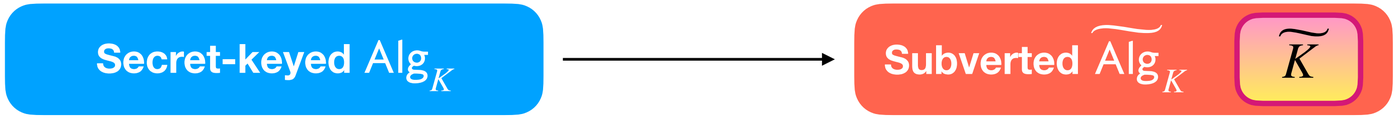
Signing [AMV15, CS03, TBEL21, …]

Decryption [CHY20, AP19, JLW25, …]

Protocols and more! [BWP+22, GBPG03, …]

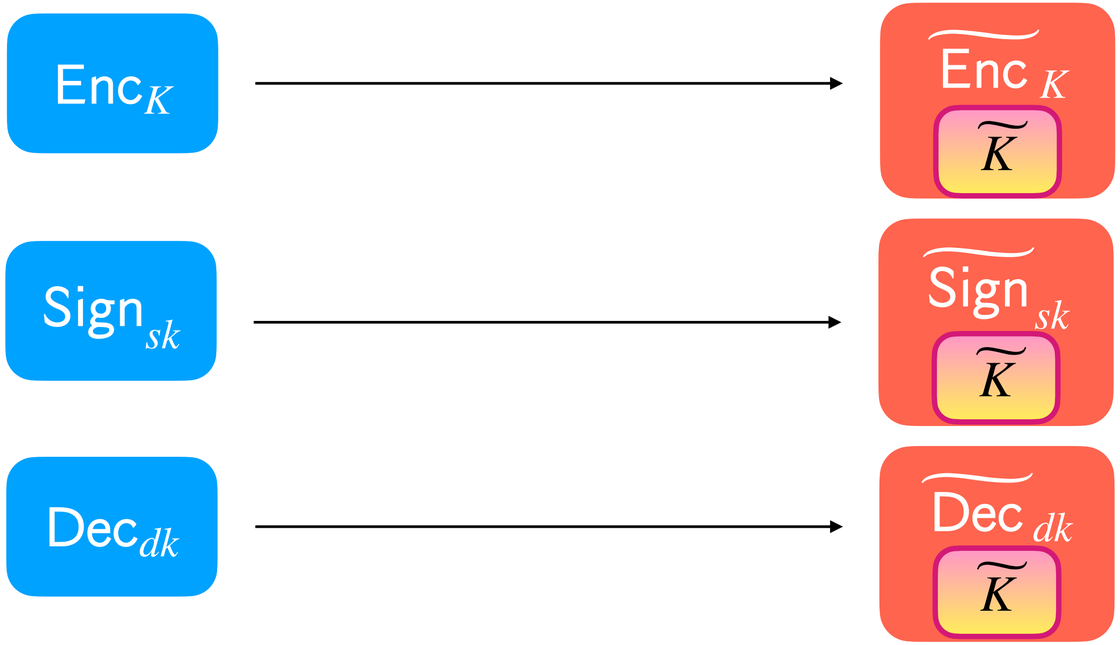
Secret-Algorithm Substitution Attack (*S*-ASA)

We call these **Secret-ASAs**.



Alice's *secret-keyed* algorithm subverted…

…with the Adversary's goal to learn K.



Symmetric encryption [BPR14, BJK15, DFP15, …]

Signing [AMV15, CS03, TBEL21, …]

Decryption [CHY20, AP19, JLW25, …]

Protocols and more! [BWP+22, GBPG03, …]

Topic of this talk:

Public-Algorithm Substitution Attack (*P*-ASA)

Summary of contributions:

1. Give a definition for an ASA on a *public* algorithm.
2. Design a construction satisfying the definition.

} **For an *arbitrary* public algorithm**

3. Look in more detail at important applications:

   Hash functions, as used in certificates or password-based authentication.

   Verification functions, in signatures.

   Verification functions, in Non-Interactive Arguments.

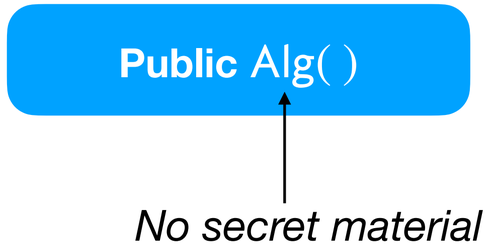
Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

The target public algorithm:



Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

The target public algorithm:

Public $\mathrm{Alg}(\,)$

No secret material

The two components of a **P-ASA** on $\mathrm{Alg}(\,)$:

Subverted $\widetilde{\mathrm{Alg}}(\,)$

Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

The target public algorithm:

Public $\mathrm{Alg}(\,)$

No secret material

The two components of a **P-ASA** on $\mathrm{Alg}(\,)$:

Subverted $\widetilde{\mathrm{Alg}}(\,)$

As before, $\widetilde{\mathrm{Alg}}$ is installed as Alice's code.

Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

The target public algorithm:

Public Alg( )

No secret material

The two components of a **P-ASA** on Alg( ):

Subverted $\widetilde{\text{Alg}}$( )

As before, $\widetilde{\text{Alg}}$ is installed as Alice's code.

Exploit Expl( )

AND, the attacker retains some kind of **Exploit** algorithm.

Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

The target public algorithm:

Public Alg( )

No secret material

Q: What exactly is the P-ASA?

The two components of a **P-ASA** on Alg( ):

Subverted $\widetilde{\text{Alg}}$( )

Exploit Expl( )

Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

The target public algorithm:

Public Alg( )

No secret material

The two components of a **P-ASA** on Alg( ):

Subverted $\widetilde{\text{Alg}}$( )

Exploit Expl( )

Q: What exactly is the P-ASA?

It is a subversion generator:

$$(\widetilde{\text{Alg}}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(\text{Alg})$$

which takes the target algorithm, and produces the two attack components.

Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

Q: What properties would the P-ASA be expected to achieve?

The target public algorithm:

Public Alg( )

No secret material

The two components of a **P-ASA** on Alg( ):

Subverted $\widetilde{\text{Alg}}$( )

Exploit Expl( )

Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

The target public algorithm:

Public Alg( )

No secret material

The two components of a **P-ASA** on Alg( ):

Subverted $\widetilde{\text{Alg}}$( )

Exploit Expl( )

Q: What properties would the P-ASA be expected to achieve?

(i) Utility

Expl( ) allows the attacker to find *structured preimages* under $\widetilde{\text{Alg}}$.

Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

The target public algorithm:

Public $\mathsf{Alg}(\,)$

No secret material

The two components of a **P-ASA** on $\mathsf{Alg}(\,)$:

Subverted $\widetilde{\mathsf{Alg}}(\,)$

Exploit $\mathsf{Expl}(\,)$

Q: What properties would the P-ASA be expected to achieve?

(i) Utility

$\mathsf{Expl}(\,)$ allows the attacker to find *structured preimages* under $\widetilde{\mathsf{Alg}}$.

(ii) Undetectability [BPR14]

It is hard to black-box distinguish $\widetilde{\mathsf{Alg}}$ and the honest Alg.

Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

The target public algorithm:

Public Alg( )

No secret material

The two components of a **P-ASA** on Alg( ):

Subverted $\widetilde{\text{Alg}}$( )

Exploit Expl( )

Q: What properties would the P-ASA be expected to achieve?

(i) Utility

Expl( ) allows the attacker to find *structured preimages* under $\widetilde{\text{Alg}}$.

(ii) Undetectability [BPR14]

It is hard to black-box distinguish $\widetilde{\text{Alg}}$ and the honest Alg.

(ii) Exclusivity

"Utility is *exclusive* to the holder of Expl."

For anyone else, it's hard to find an input x on which $\widetilde{\text{Alg}}$ and Alg differ.

+ With oracle access to Expl( )
+ And with white-box descriptions of $\widetilde{\text{Alg}}$ and Alg.

Public-Algorithm Substitution Attack (*P*-ASA)

1. Give a definition for an ASA on a *public* algorithm.

The target public algorithm:

Public Alg( )

No secret material

The two components of a **P-ASA** on Alg( ):

Subverted $\widetilde{\text{Alg}}$( )

Exploit Expl( )

Q: What properties would the P-ASA be expected to achieve?

(i) Utility

Expl( ) allows the attacker to find *structured preimages* under $\widetilde{\text{Alg}}$.

(ii) Undetectability [BPR14]

It is hard to black-box distinguish $\widetilde{\text{Alg}}$ and the honest Alg.

(ii) Exclusivity

"Utility is *exclusive* to the holder of Expl."

For anyone else, it's hard to find an input x on which $\widetilde{\text{Alg}}$ and Alg differ.

+ With oracle access to Expl( )
+ And with white-box descriptions of $\widetilde{\text{Alg}}$ and Alg.

Public-Algorithm Substitution Attack (*P*-ASA)

Summary of contributions:

1. Give a definition for an ASA on a *public* algorithm.
2. **Design a construction satisfying the definition.**

} **For an *arbitrary* public algorithm**

Public-Algorithm Substitution Attack (*P*-ASA)

Summary of contributions:

1. Give a definition for an ASA on a *public* algorithm.
2. **Design a construction satisfying the definition.**

} **For an *arbitrary* public algorithm**

We construct a P-ASA using an SUF signature scheme, and an “embedding function.”

Generalizing GKVZ22 (“ML backdoors”)

Public-Algorithm Substitution Attack (*P*-ASA)

Summary of contributions:

1. Give a definition for an ASA on a *public* algorithm.
2. Design a construction satisfying the definition.

} **For an *arbitrary* public algorithm**

3. **Look in more detail at important applications:**

Hash functions, as used in certificates or password-based authentication.

Verification functions, in Non-Interactive Arguments.

Verification functions, in signatures.

Public-Algorithm Substitution Attack (*P*-ASA)

Summary of contributions:

1. Give a definition for an ASA on a *public* algorithm.
2. Design a construction satisfying the definition.

} **For an *arbitrary* public algorithm**

3. **Look in more detail at important applications:**

Hash functions, as used in certificates or password-based authentication.

Our P-ASA allows the attacker to find structured preimages.

For example, for certificate forgery!

Verification functions, in Non-Interactive Arguments.

Our P-ASA allows the attacker to prove arbitrary (false) statements.

Verification functions, in signatures.

Our P-ASA allows the attacker to forge signatures for arbitrary messages and keys.

Public-Algorithm Substitution Attack (*P*-ASA)

Summary of contributions:

1. Give a definition for an ASA on a *public* algorithm.
2. Design a construction satisfying the definition.

} **For an *arbitrary* public algorithm**

3. **Look in more detail at important applications:**

Hash functions, as used in certificates or password-based authentication.

Our P-ASA allows the attacker to find structured preimages.

For example, for certificate forgery!

Verification functions, in Non-Interactive Arguments.

Our P-ASA allows the attacker to prove arbitrary (false) statements.

Verification functions, in signatures.

Our P-ASA allows the attacker to forge signatures for arbitrary messages and keys.

These are all specific cases of our general construction

☑ Introductory picture & overview of results

Public-Algorithm Substitution Attacks on Hash functions

- ☐ **Definitions: Undetectability, Exclusivity, Utility**
- ☐ Construction of a Public-ASA
- ☐ One application

☐ Concluding remarks on the general case

The honest picture for a Hash function H

$H \xleftarrow{\$} \text{HGen}$

H is selected honestly by generator HGen,
and may include a hardcoded key.

Desired security property: CR

H is collision-resistant to any (efficient) adversary,
when the adversary is given H.

The subverted picture for a Hash function H

A P-ASA is a “subversion generator”
which generates $\widetilde{H}$ and Expl given H.

$$(\widetilde{H}, \mathsf{Expl}) \xleftarrow{\$} \mathsf{SubGen}(H)$$

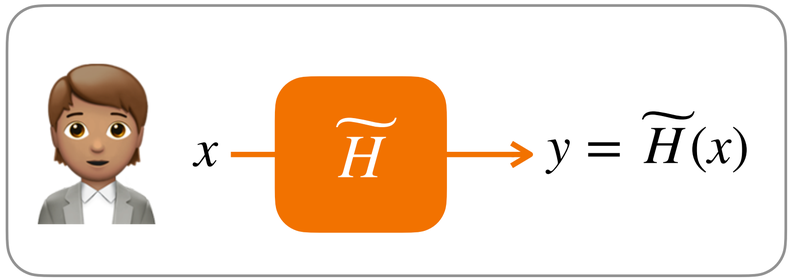
$\widetilde{H}$ and Expl may include hardcoded keys.

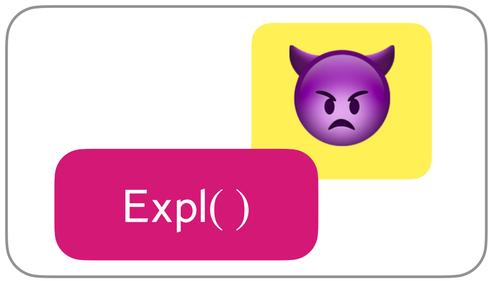
The subverted picture for a Hash function H

A P-ASA is a “subversion generator”
which generates $\widetilde{H}$ and Expl given H.

$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.



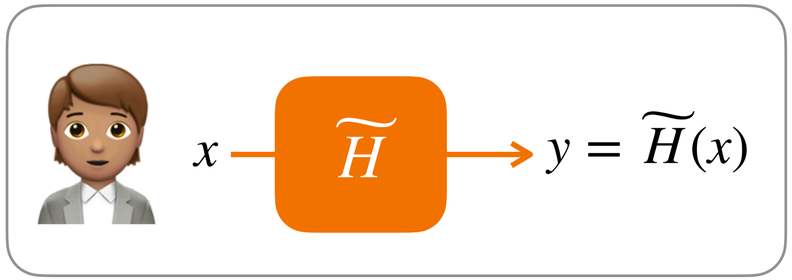


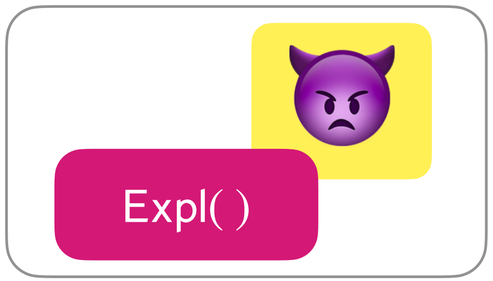
The subverted picture for a Hash function H

A P-ASA is a “subversion generator” which generates $\widetilde{H}$ and Expl given H.

$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





Utility

Our Utility asks that:

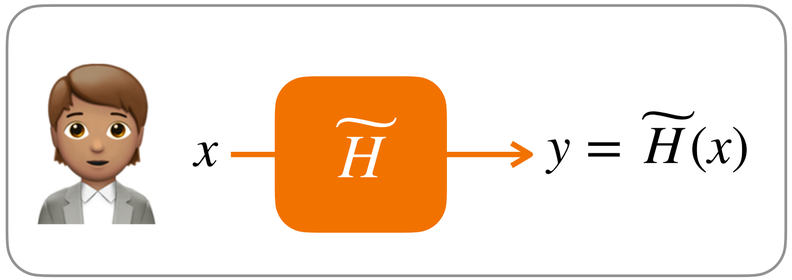
An attacker with Expl( ) can use it to find structured preimages under $\widetilde{H}$.

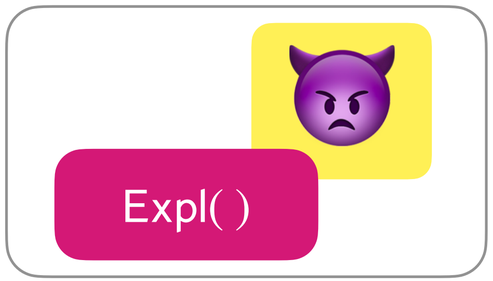
The subverted picture for a Hash function H

A P-ASA is a “subversion generator” which generates $\widetilde{H}$ and Expl given H.

$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





Utility

Our Utility asks that:

An attacker with Expl( ) can use it to find structured preimages under $\widetilde{H}$.

For any desired **structure** and target,

$x \leftarrow \text{Expl}(\texttt{structure}, \texttt{target})$

should yield:

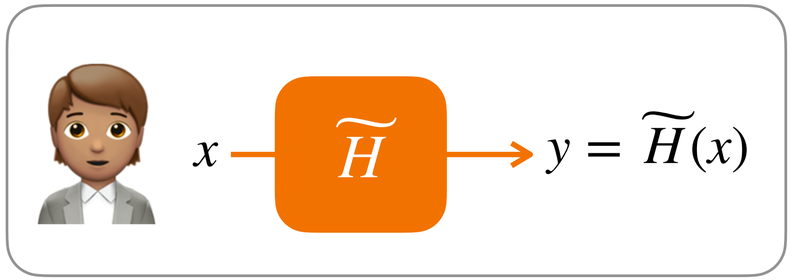
$\widetilde{H}(x) = \texttt{target}$
and a correctly **structured** x.

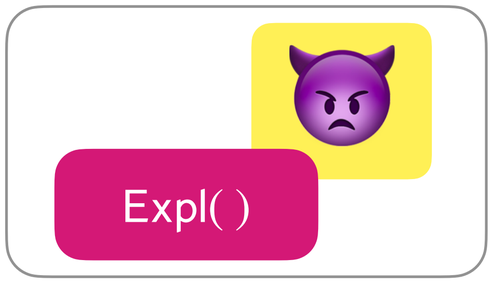
The subverted picture for a Hash function H

A P-ASA is a "subversion generator" which generates $\widetilde{H}$ and Expl given H.

$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





Utility

Our Utility asks that:

An attacker with Expl( ) can use it to find structured preimages under $\widetilde{H}$.

For any desired **structure** and target,

$x \leftarrow \text{Expl}(\texttt{structure}, \texttt{target})$

should yield:

$\widetilde{H}(x) = \texttt{target}$
and a correctly **structured** x.

What "structure" is this possible for?

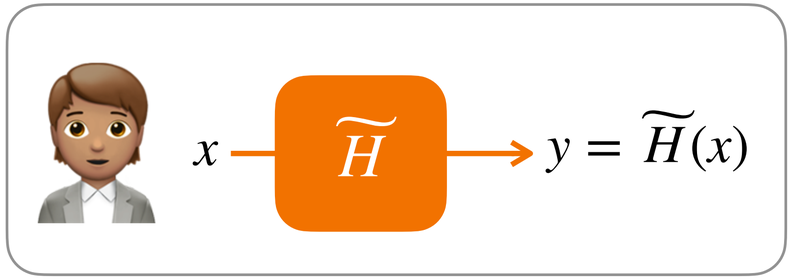
- Requiring a specific prefix or suffix
- Requiring that x be an X.509 cert with certain data
- …more! The paper gives constraints.

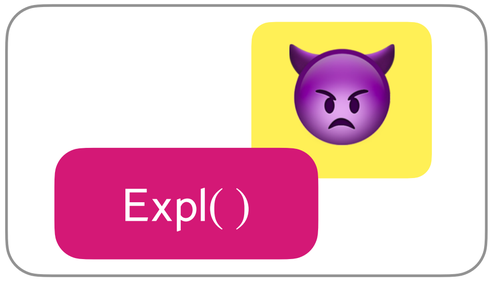
The subverted picture for a Hash function H

A P-ASA is a “subversion generator” which generates $\widetilde{H}$ and Expl given H.

$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.



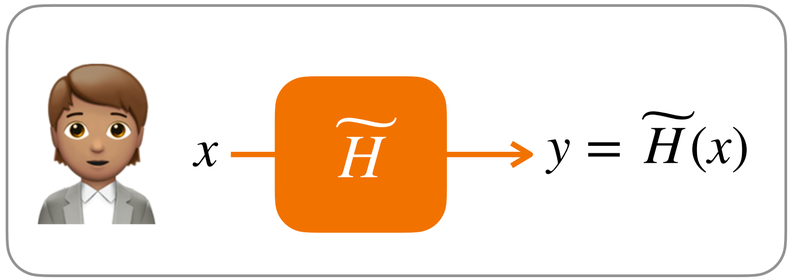


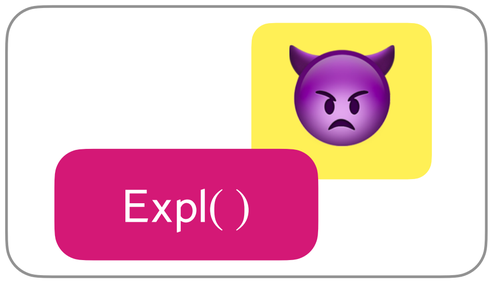
The subverted picture for a Hash function H

A P-ASA is a “subversion generator” which generates $\widetilde{H}$ and Expl given H.

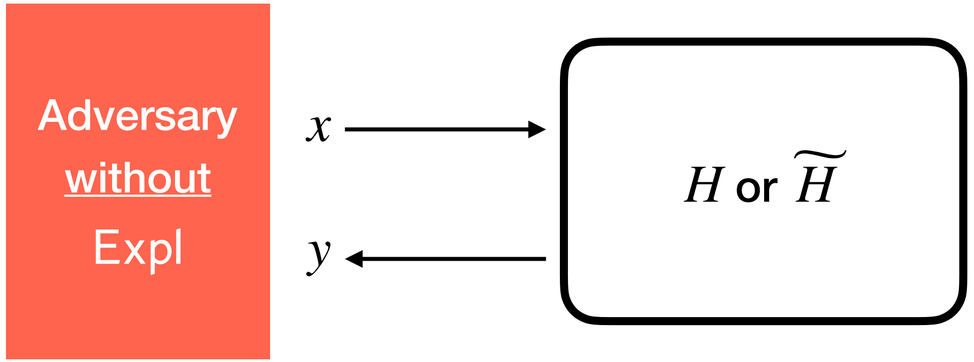
$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





Undetectability

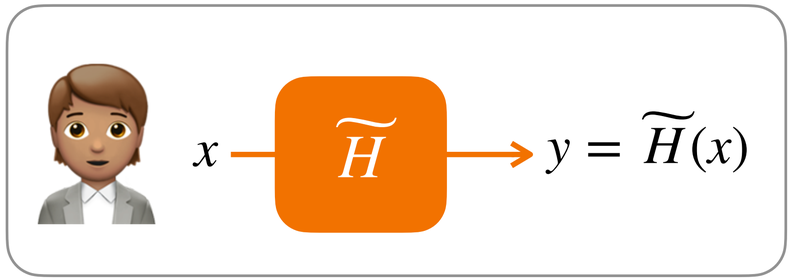


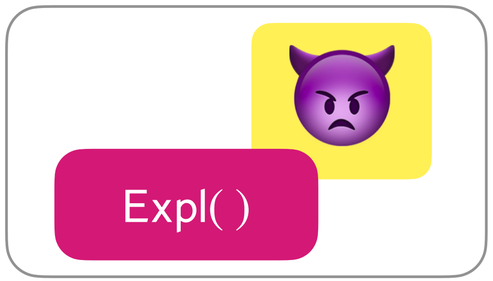
The subverted picture for a Hash function H

A P-ASA is a “subversion generator” which generates $\widetilde{H}$ and Expl given H.

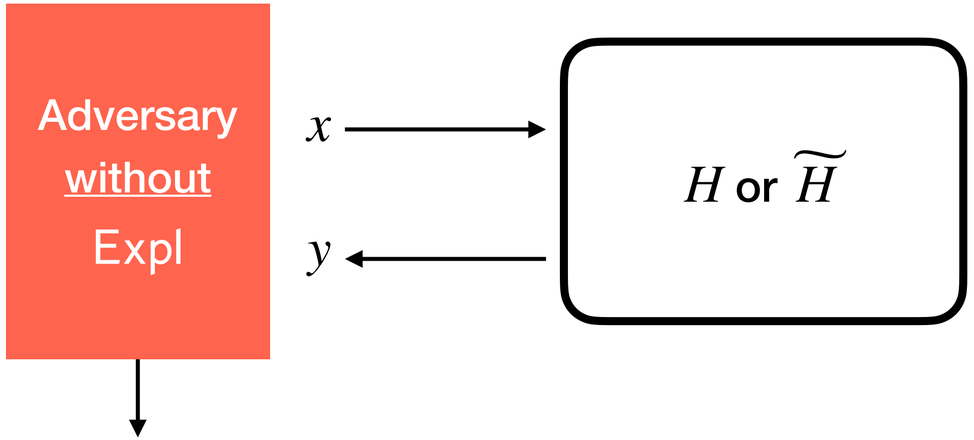
$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





Undetectability



Decide which is in the box

Undetectability says:

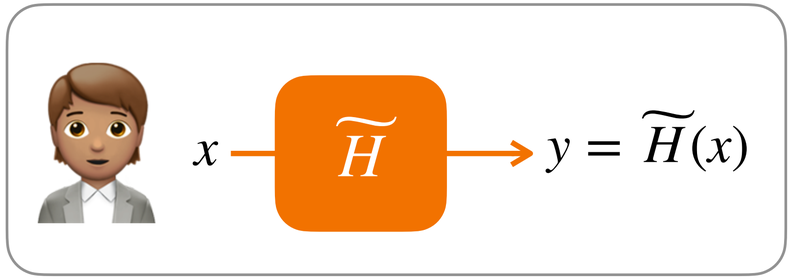
It’s hard to correctly decide which of H or $\widetilde{H}$ is in the box, for all (efficient) adversaries without Expl.

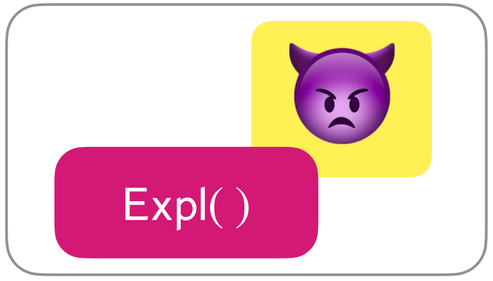
The subverted picture for a Hash function H

A P-ASA is a “subversion generator” which generates $\widetilde{H}$ and Expl given H.

$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





Exclusivity

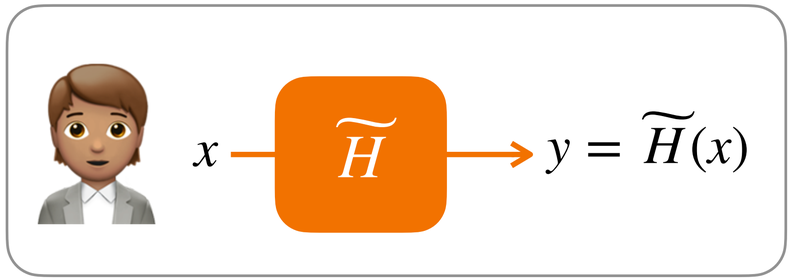
Adversary
<u>without</u>
Expl

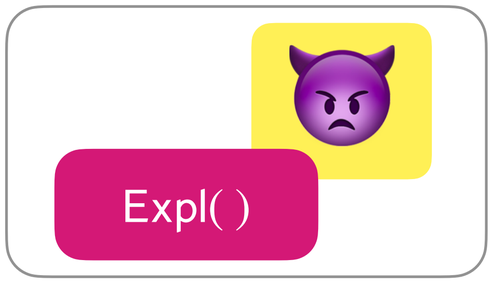
The subverted picture for a Hash function H

A P-ASA is a “subversion generator” which generates $\widetilde{H}$ and Expl given H.

$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





Exclusivity

Descriptions of H and $\widetilde{H}$

↓

Adversary **without** Expl

↓

An x such that $H(x) \neq \widetilde{H}(x)$

Exclusivity says:

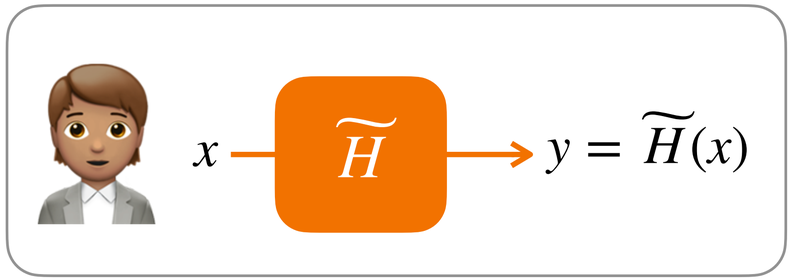
Any differences between H and $\widetilde{H}$ are only findable by the adversary with the exploit, not by anyone else.

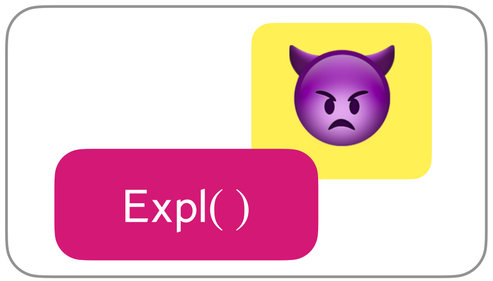
The subverted picture for a Hash function H

A P-ASA is a "subversion generator" which generates $\widetilde{H}$ and Expl given H.

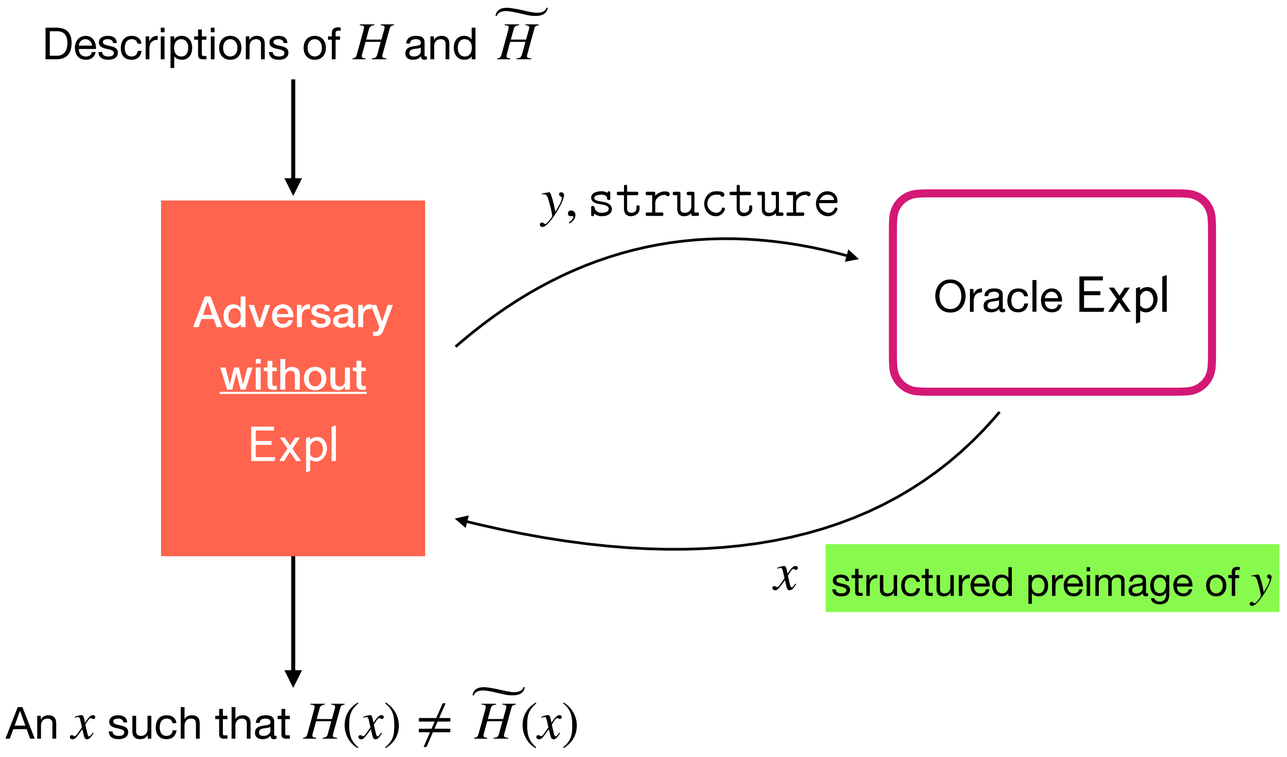
$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





Exclusivity



Exclusivity says:

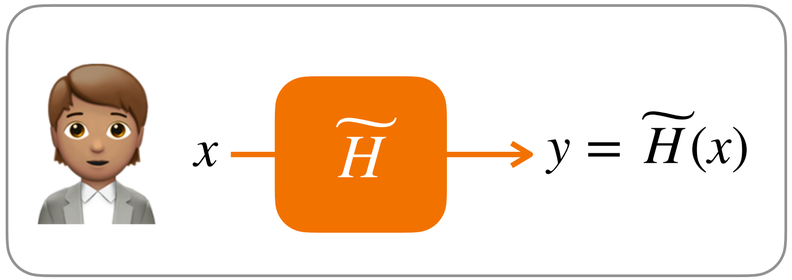
Any differences between H and $\widetilde{H}$ are only findable by the adversary with the exploit, not by anyone else.

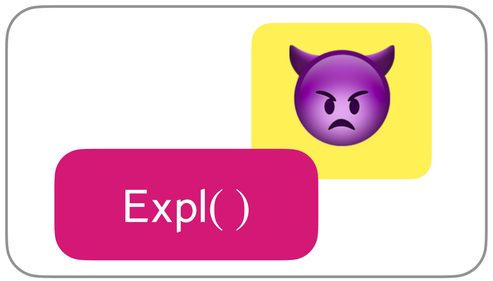
The subverted picture for a Hash function H

A P-ASA is a "subversion generator" which generates $\widetilde{H}$ and Expl given H.

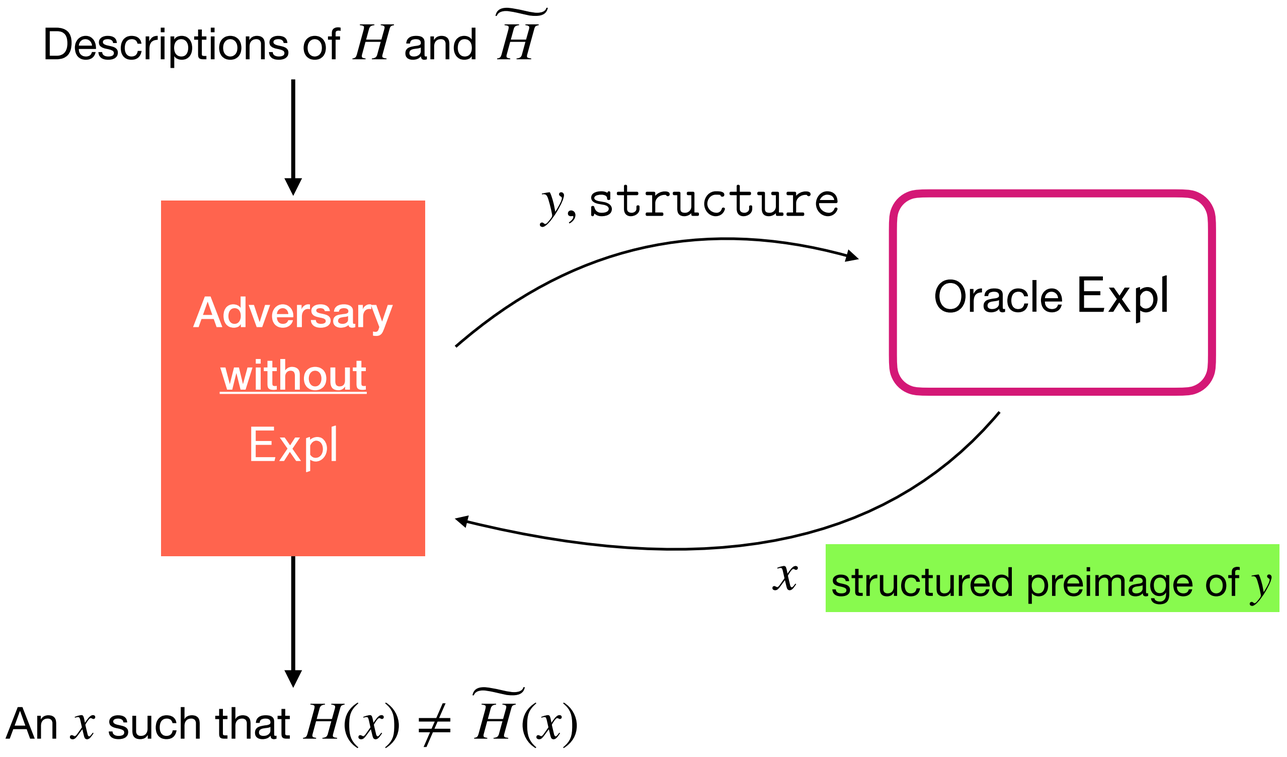
$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





Exclusivity



Exclusivity says:

Any differences between H and $\widetilde{H}$ are only findable by the adversary with the exploit, not by anyone else.

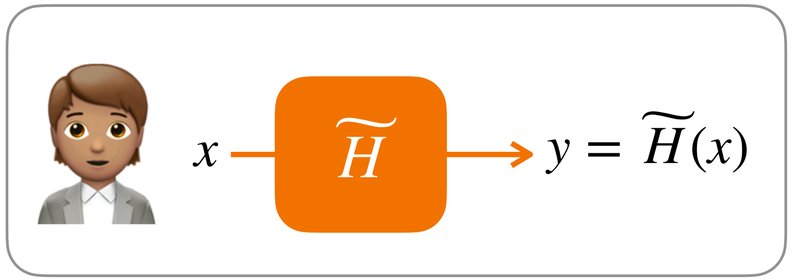
Remark: This implies undetectability!

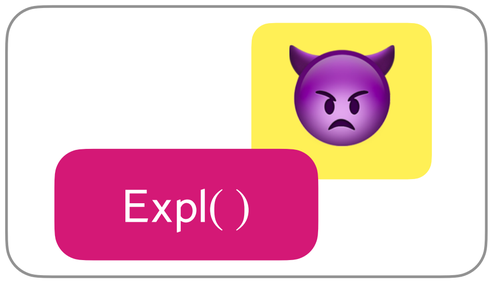
The subverted picture for a Hash function H

A P-ASA is a “subversion generator” which generates $\widetilde{H}$ and Expl given H.

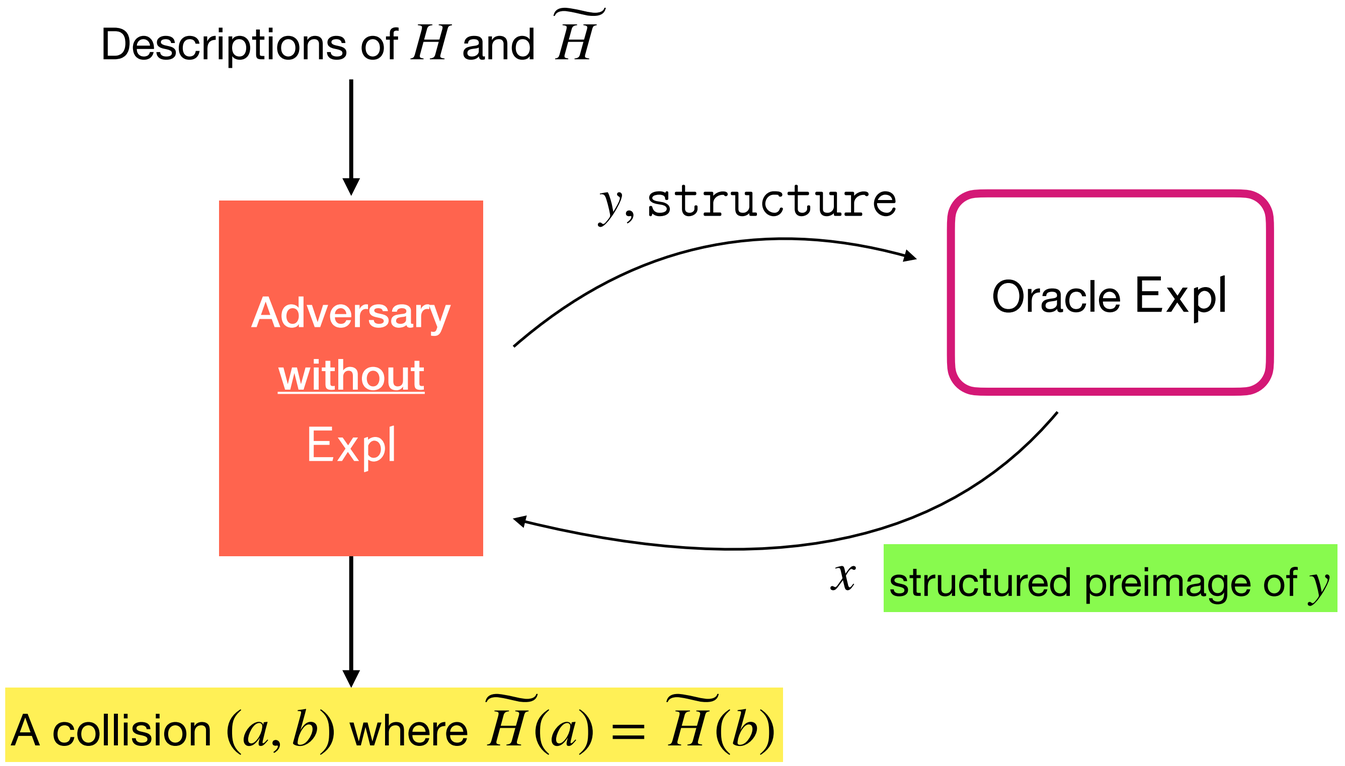
$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





Exclusivity

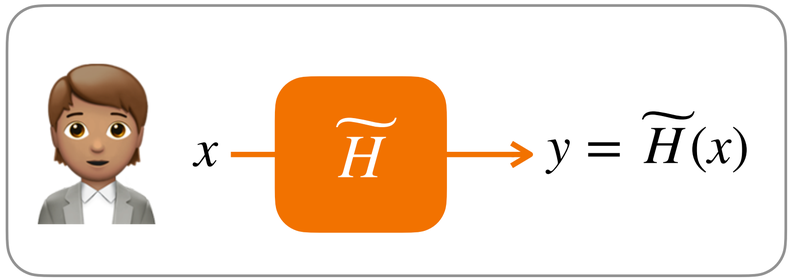


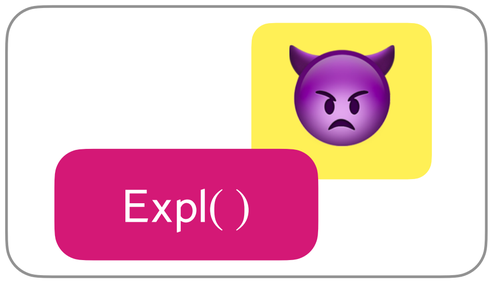
The subverted picture for a Hash function H

A P-ASA is a "subversion generator" which generates $\widetilde{H}$ and Expl given H.

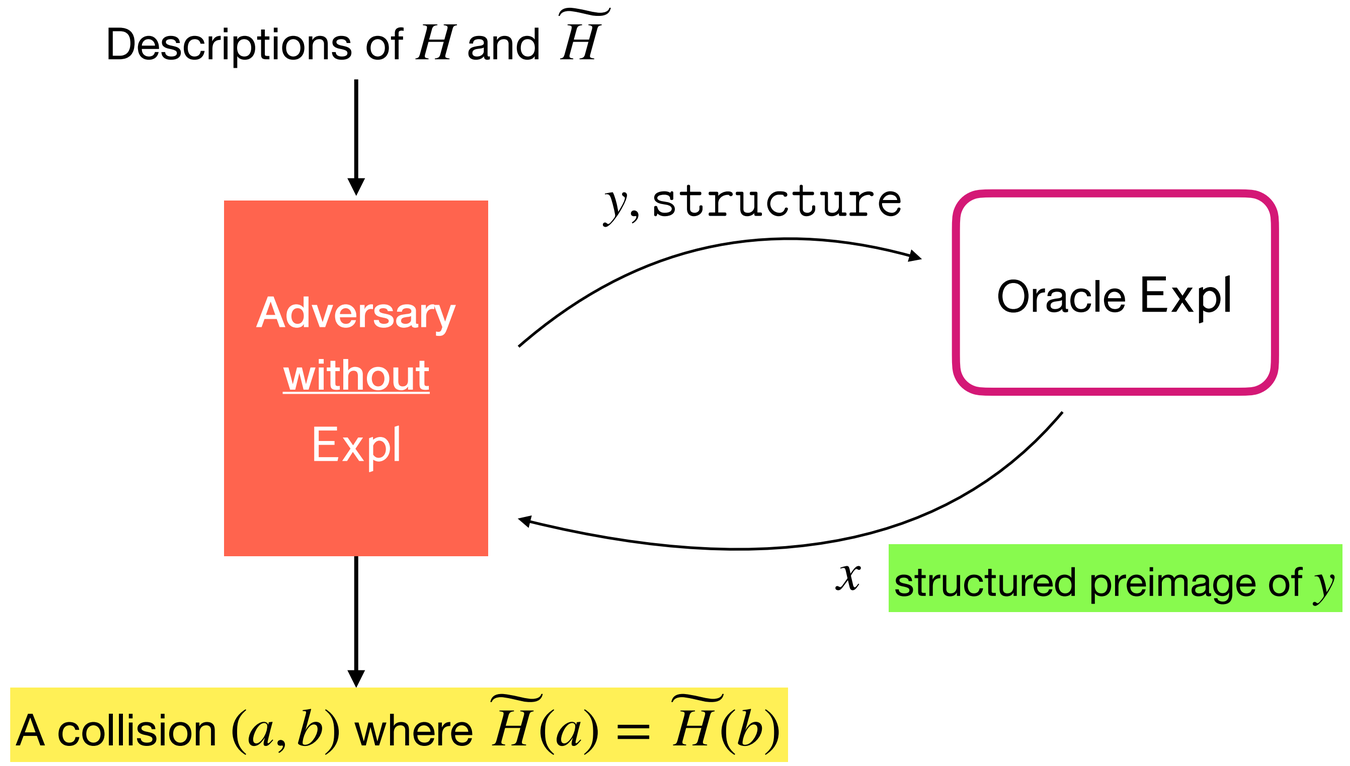
$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





"CR Exclusivity" for a Hash function

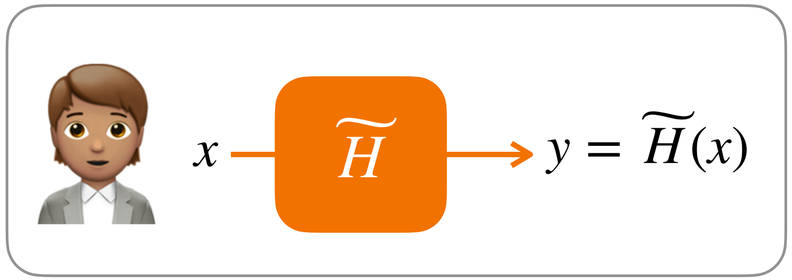


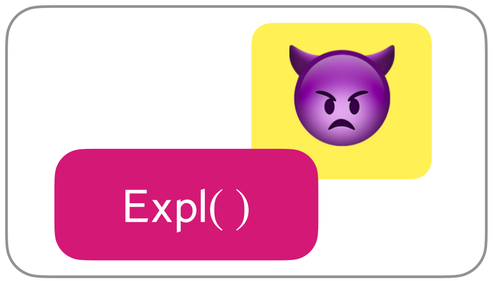
The subverted picture for a Hash function H

A P-ASA is a "subversion generator" which generates $\widetilde{H}$ and Expl given H.

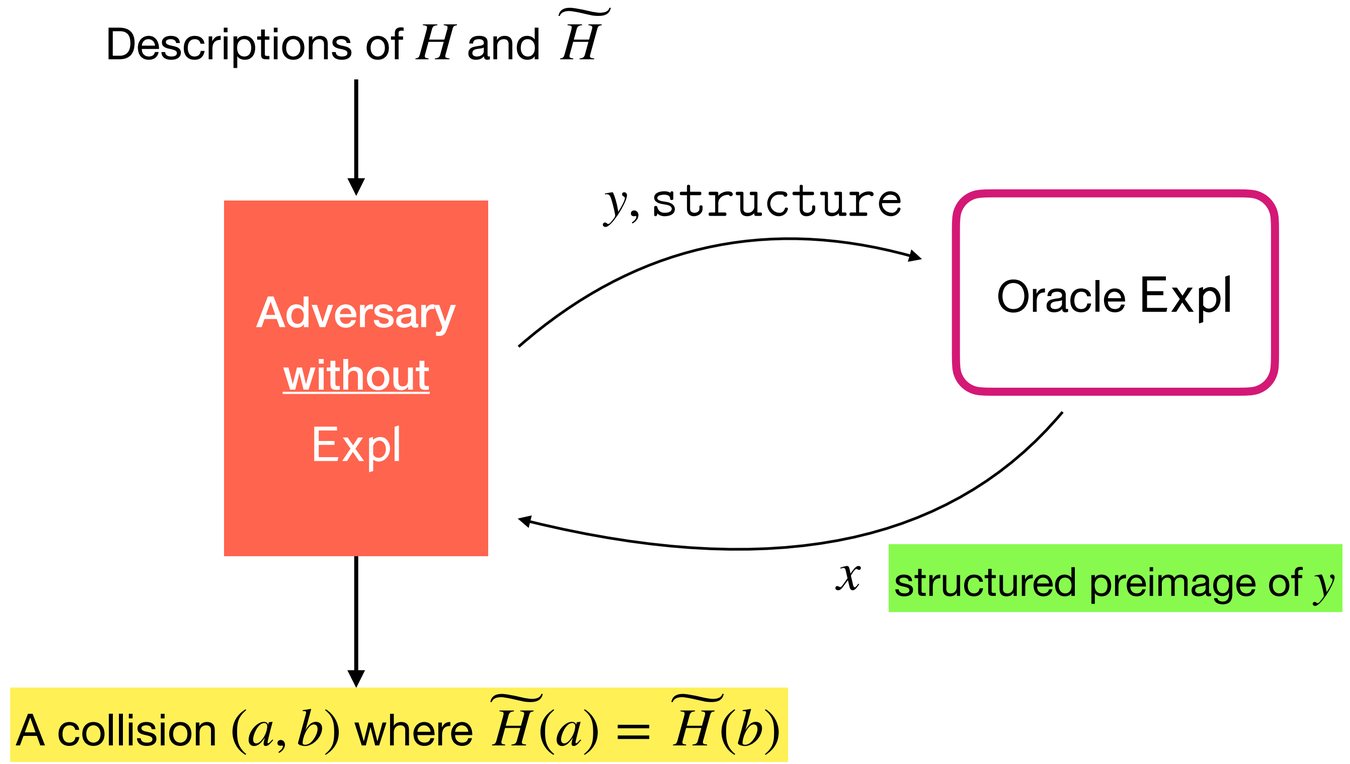
$$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$$

$\widetilde{H}$ and Expl may include hardcoded keys.





"CR Exclusivity" for a Hash function



We can ask for "CR Exclusivity" as well.

It turns out to be implied by **Exclusivity** on the prior slides, and **CR** of the original H.

Other styles of subversion CR were considered by [FJM18, AAEMS14], specifically for Hash functions.

☑ Introductory picture & overview of results

Public-Algorithm Substitution Attacks on Hash functions

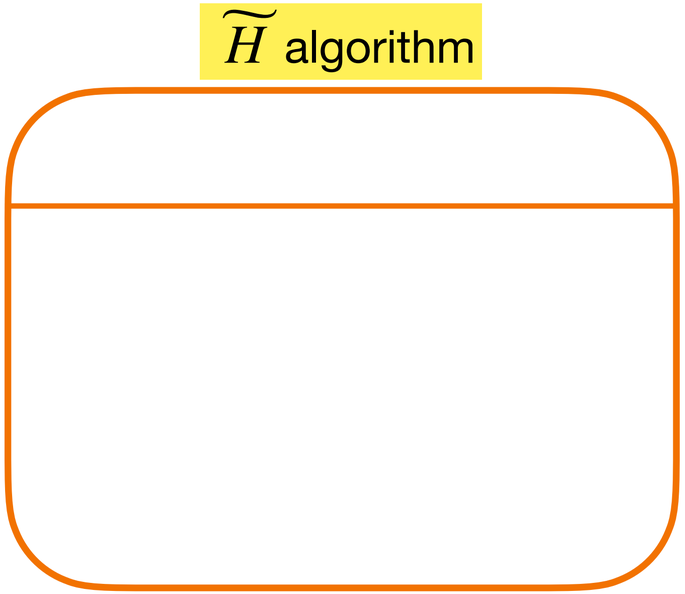
- ☑ Definitions: Undetectability, Exclusivity, Utility
- ☐ **Construction of a Public-ASA**
- ☐ One application

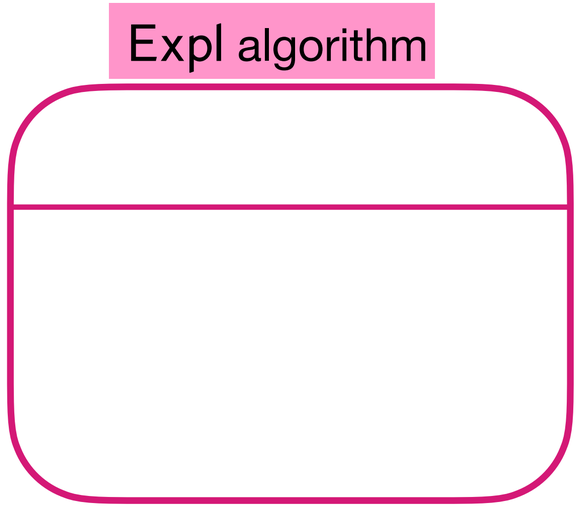
☐ Concluding remarks on the general case

Can you build such a Public-ASA?

One idea:
FJM18

$(\widetilde{H}, \mathsf{Expl}) \xleftarrow{\$} \mathsf{FJMSubGen}(H)$

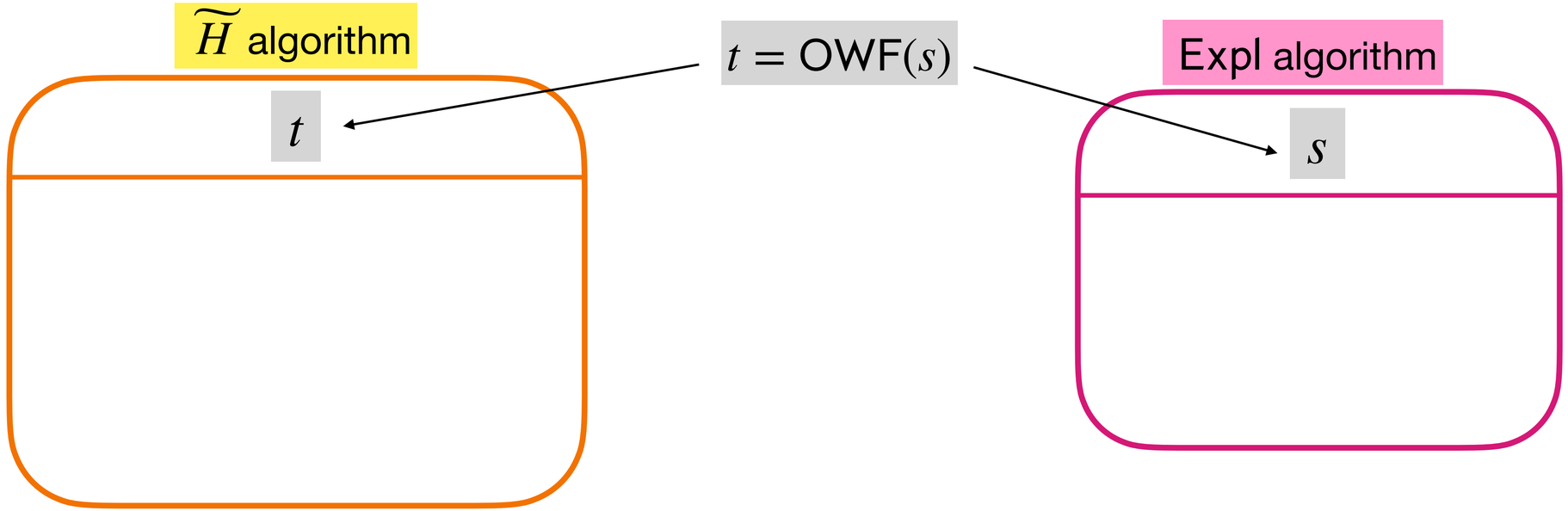




Can you build such a Public-ASA?

One idea:
FJM18

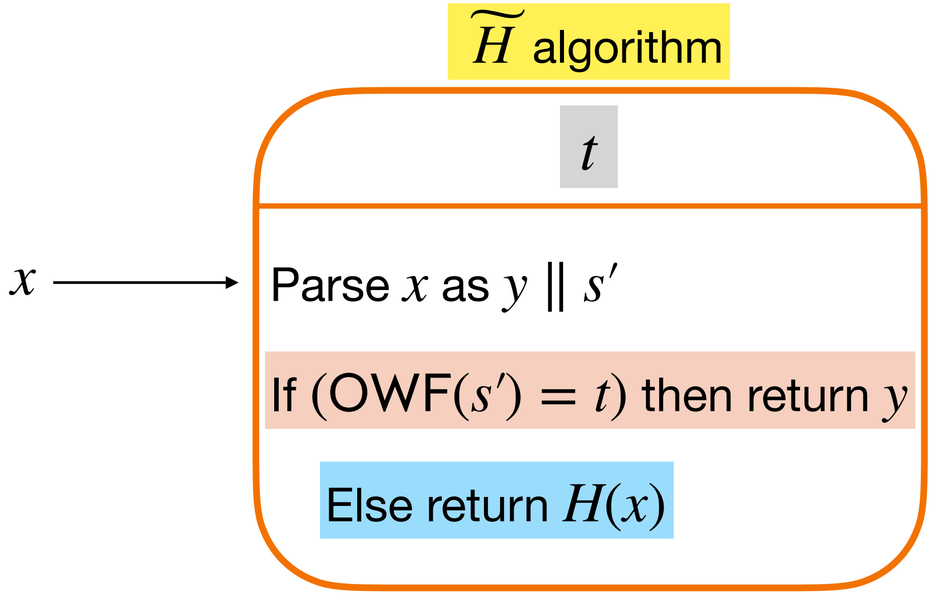
$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{FJMSubGen}(H)$



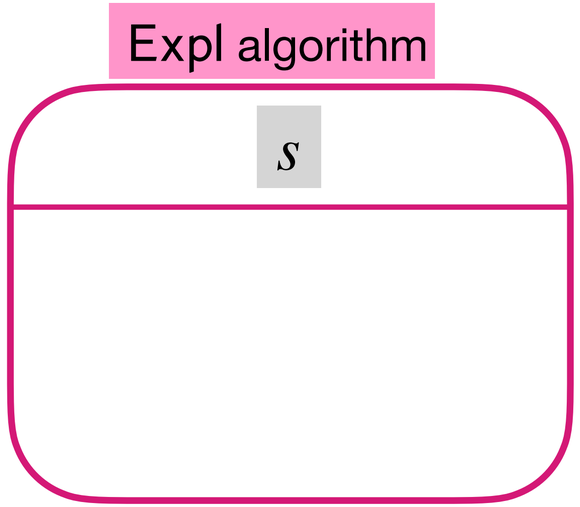
Can you build such a Public-ASA?

One idea: FJM18

$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{FJMSubGen}(H)$



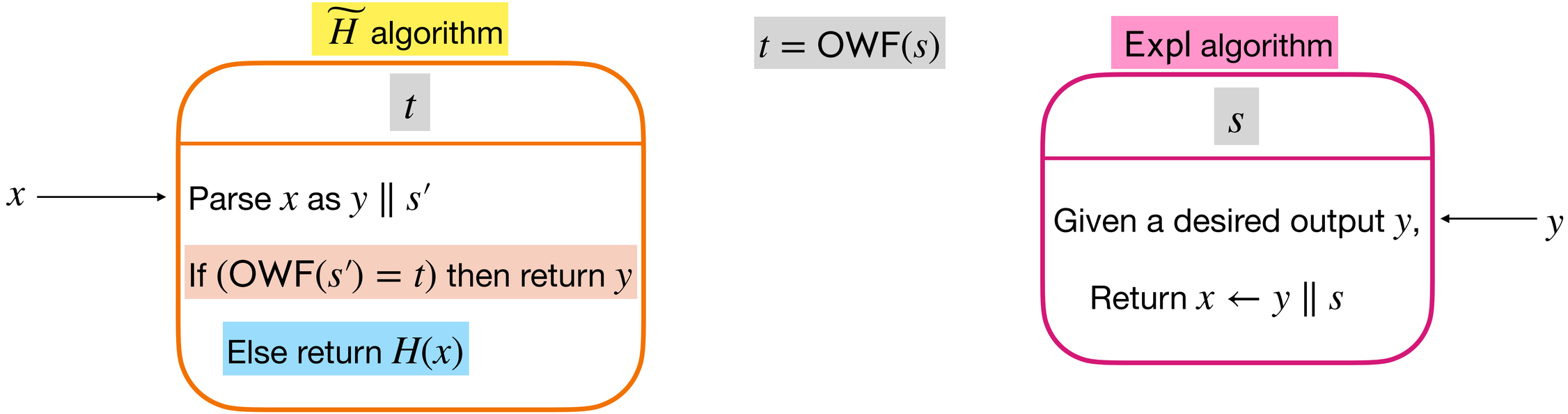
$t = \text{OWF}(s)$



Can you build such a Public-ASA?

One idea: FJM18

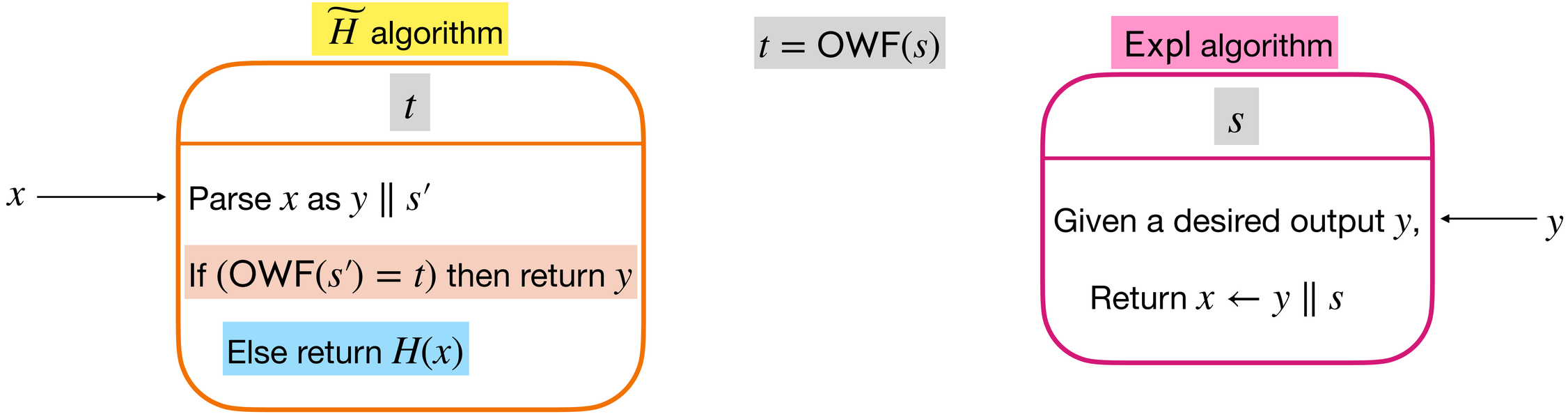
$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{FJMSubGen}(H)$



Can you build such a Public-ASA?

One idea: FJM18

$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{FJMSubGen}(H)$



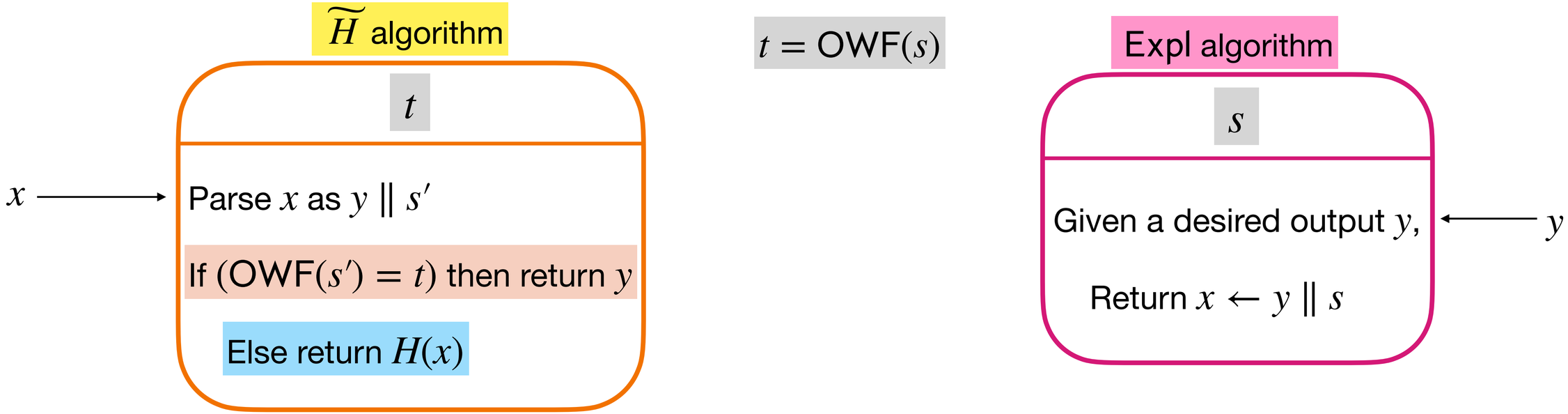
Utility? Not Really.

The attacker, with Expl, can find preimages, but they *can only* be structured as $y \| s$.

Can you build such a Public-ASA?

One idea: FJM18

$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{FJMSubGen}(H)$



Utility? Not Really.

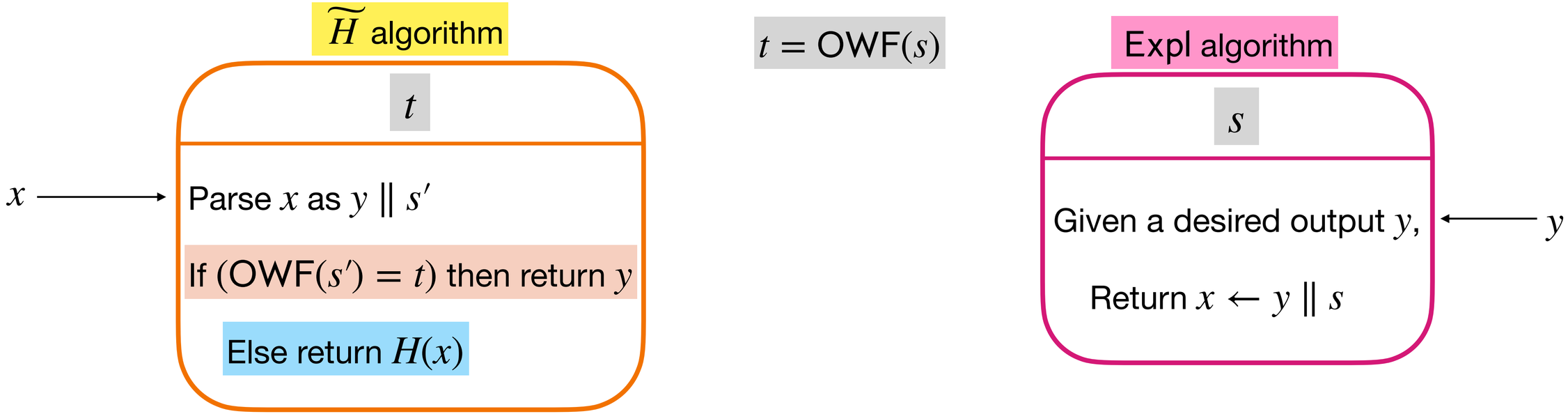
The attacker, with Expl, can find preimages, but they *can only* be structured as $y \| s$.

Black-box undetectable? Yes, assuming OWF.

Can you build such a Public-ASA?

One idea: FJM18

$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{FJMSubGen}(H)$



Utility? Not Really.

The attacker, with Expl, can find preimages, but they *can only* be structured as $y \| s$.

Black-box undetectable? Yes, assuming OWF.

Exclusive? NO.

Seeing one Expl-produced $x = y \| s$ reveals s, which allows finding preimages (breaking CR).

Our construction

$(\widetilde{H}, \mathsf{Expl}) \xleftarrow{\$} \mathsf{SubGen}(H)$

Invertible embedding function:
$\mathsf{Embed}, \mathsf{Embed}^{-1}$

Signature scheme S

$\widetilde{H}$ algorithm

Expl algorithm

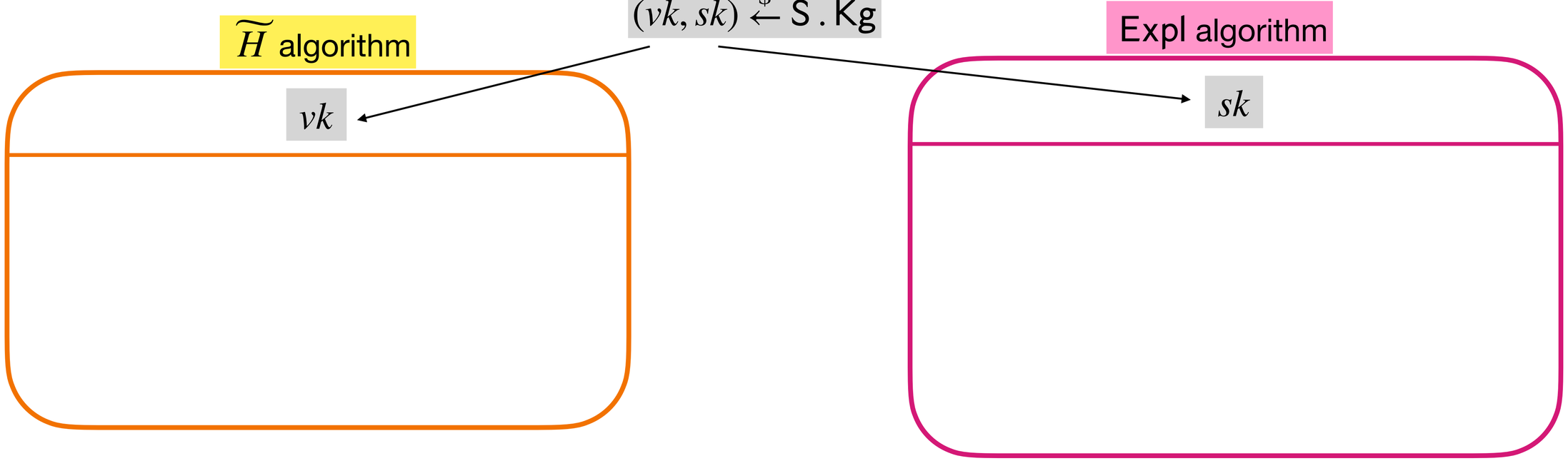
Our construction

$(\widetilde{H}, \mathsf{Expl}) \xleftarrow{\$} \mathsf{SubGen}(H)$

Invertible embedding function:
$\mathsf{Embed}, \mathsf{Embed}^{-1}$

Signature scheme S

$(vk, sk) \xleftarrow{\$} \mathsf{S}.\mathsf{Kg}$



Our construction

$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$

Invertible embedding function:
Embed, Embed^{-1}

Signature scheme S

$(vk, sk) \xleftarrow{\$} \text{S . Kg}$

$\widetilde{H}$ algorithm

vk

x →

Run $(y, struc, \sigma) \leftarrow \text{Embed}^{-1}(x)$

If $\text{S . Vfy}(vk, (y, struc), \sigma)$ then return y

Else return $H(x)$

Expl algorithm

sk

Our construction

$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$

Invertible embedding function:
Embed, Embed^{-1}

Signature scheme S

$(vk, sk) \xleftarrow{\$} \text{S . Kg}$

$\widetilde{H}$ algorithm

vk

x

Run $(y, struc, \sigma) \leftarrow \text{Embed}^{-1}(x)$

If $\text{S . Vfy}(vk, (y, struc), \sigma)$ then return y

Else return $H(x)$

Expl algorithm

sk

$y, struc$

Given a **desired output** y
and **structure info** $struc$,

Compute $\sigma \xleftarrow{\$} \text{S . Sign}(sk, (y, struc))$

Return $x \leftarrow \text{Embed}(y, struc, \sigma)$

Our construction

$(\widetilde{H}, \mathsf{Expl}) \xleftarrow{\$} \mathsf{SubGen}(H)$

Invertible embedding function:
Embed, Embed^{-1}

Signature scheme S

$(vk, sk) \xleftarrow{\$} \mathsf{S}.\mathsf{Kg}$

$\widetilde{H}$ algorithm

vk

x

Run $(y, struc, \sigma) \leftarrow \mathsf{Embed}^{-1}(x)$

If $\mathsf{S}.\mathsf{Vfy}(vk, (y, struc), \sigma)$ then return y

Else return $H(x)$

Expl algorithm

sk

$y, struc$

Given a **desired output** y
and **structure info** $struc$,

Compute $\sigma \xleftarrow{\$} \mathsf{S}.\mathsf{Sign}(sk, (y, struc))$

Return $x \leftarrow \mathsf{Embed}(y, struc, \sigma)$

Utility? YES, for structures that can be embedded this way.

The attacker can find *structured preimages*, assuming S is correct and the embedding "works."

Our construction

$(\widetilde{H}, \mathsf{Expl}) \xleftarrow{\$} \mathsf{SubGen}(H)$

Invertible embedding function:
Embed, Embed^{-1}

Signature scheme S

$(vk, sk) \xleftarrow{\$} \mathsf{S}.\mathsf{Kg}$

$\widetilde{H}$ algorithm

vk

x

Run $(y, struc, \sigma) \leftarrow \mathsf{Embed}^{-1}(x)$

If $\mathsf{S}.\mathsf{Vfy}(vk, (y, struc), \sigma)$ then return y

Else return $H(x)$

Expl algorithm

sk

$y, struc$

Given a **desired output** y
and **structure info** $struc$,

Compute $\sigma \xleftarrow{\$} \mathsf{S}.\mathsf{Sign}(sk, (y, struc))$

Return $x \leftarrow \mathsf{Embed}(y, struc, \sigma)$

Utility? YES, for structures that can be embedded this way.

The attacker can find *structured preimages*, assuming S is correct and the embedding "works."

Running "$x \leftarrow \mathsf{Embed}(y, struc, \sigma)$" means x has this desired $struc$.

Our construction

$(\widetilde{H}, \mathsf{Expl}) \xleftarrow{\$} \mathsf{SubGen}(H)$

Invertible embedding function:
Embed, Embed^{-1}

Signature scheme S

$(vk, sk) \xleftarrow{\$} \mathsf{S}.\mathsf{Kg}$

$\widetilde{H}$ algorithm

vk

x

Run $(y, struc, \sigma) \leftarrow \mathsf{Embed}^{-1}(x)$

If $\mathsf{S}.\mathsf{Vfy}(vk, (y, struc), \sigma)$ then return y

Else return $H(x)$

Expl algorithm

sk

$y, struc$

Given a **desired output** y
and **structure info** $struc$,

Compute $\sigma \xleftarrow{\$} \mathsf{S}.\mathsf{Sign}(sk, (y, struc))$

Return $x \leftarrow \mathsf{Embed}(y, struc, \sigma)$

Utility? YES, for structures that can be embedded this way.

The attacker can find *structured preimages*, assuming S is correct and the embedding "works."

Our construction

$(\widetilde{H}, \mathsf{Expl}) \xleftarrow{\$} \mathsf{SubGen}(H)$

Invertible embedding function: Embed, Embed^{-1}

Signature scheme S

$(vk, sk) \xleftarrow{\$} \mathsf{S}.\mathsf{Kg}$

$\widetilde{H}$ algorithm

vk

x

Run $(y, struc, \sigma) \leftarrow \mathsf{Embed}^{-1}(x)$

If $\mathsf{S}.\mathsf{Vfy}(vk, (y, struc), \sigma)$ then return y

Else return $H(x)$

Expl algorithm

sk

$y, struc$

Given a **desired output** y
and **structure info** $struc$,

Compute $\sigma \xleftarrow{\$} \mathsf{S}.\mathsf{Sign}(sk, (y, struc))$

Return $x \leftarrow \mathsf{Embed}(y, struc, \sigma)$

Utility? YES, for structures that can be embedded this way.

The attacker can find *structured preimages*, assuming S is correct and the embedding "works."

Black-box undetectable? YES, assuming UF-CMA of signature scheme S.

Our construction

$(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$

Invertible embedding function:
Embed, Embed^{-1}

Signature scheme S

$(vk, sk) \xleftarrow{\$} \text{S . Kg}$

$\widetilde{H}$ algorithm

vk

x

Run $(y, struc, \sigma) \leftarrow \text{Embed}^{-1}(x)$

If $\text{S . Vfy}(vk, (y, struc), \sigma)$ then return y

Else return $H(x)$

Expl algorithm

sk

$y, struc$

Given a **desired output** y
and **structure info** $struc$,

Compute $\sigma \xleftarrow{\$} \text{S . Sign}(sk, (y, struc))$

Return $x \leftarrow \text{Embed}(y, struc, \sigma)$

Exclusive? YES.

Assuming SUF-CMA of the signature scheme S, and that the embedding "works."

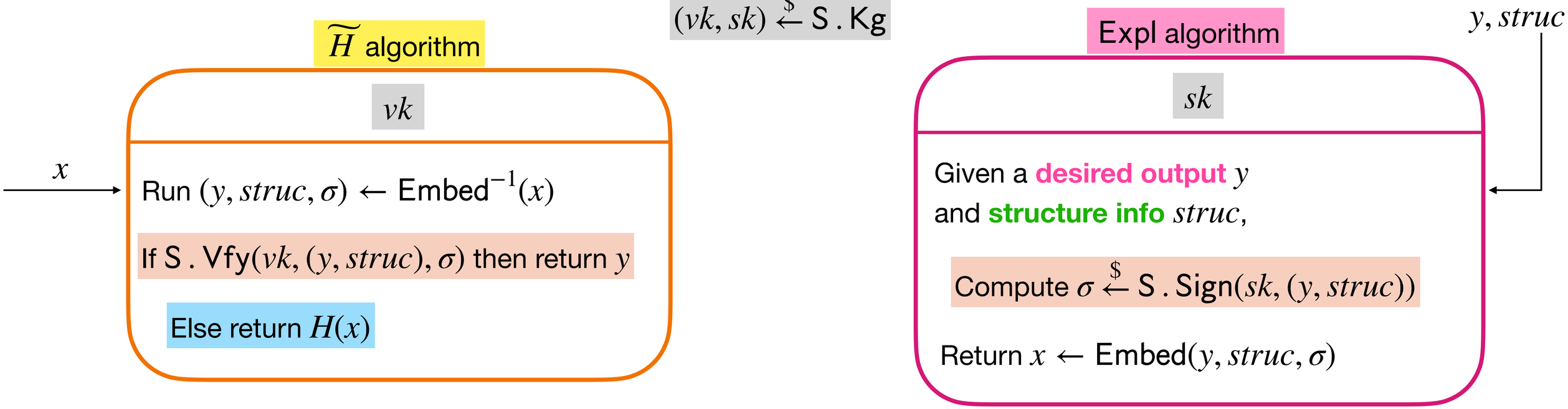
Our construction

$(\widetilde{H}, \mathsf{Expl}) \xleftarrow{\$} \mathsf{SubGen}(H)$

Invertible embedding function:
$\mathsf{Embed}, \mathsf{Embed}^{-1}$

Signature scheme S

$(vk, sk) \xleftarrow{\$} \mathsf{S}.\mathsf{Kg}$



Exclusive? YES.
Assuming SUF-CMA of the signature scheme S, and that the embedding "works."

CR Exclusive? YES.
Additionally assuming that the original H is CR.

☑ Introductory picture & overview of results

Public-Algorithm Substitution Attacks on Hash functions

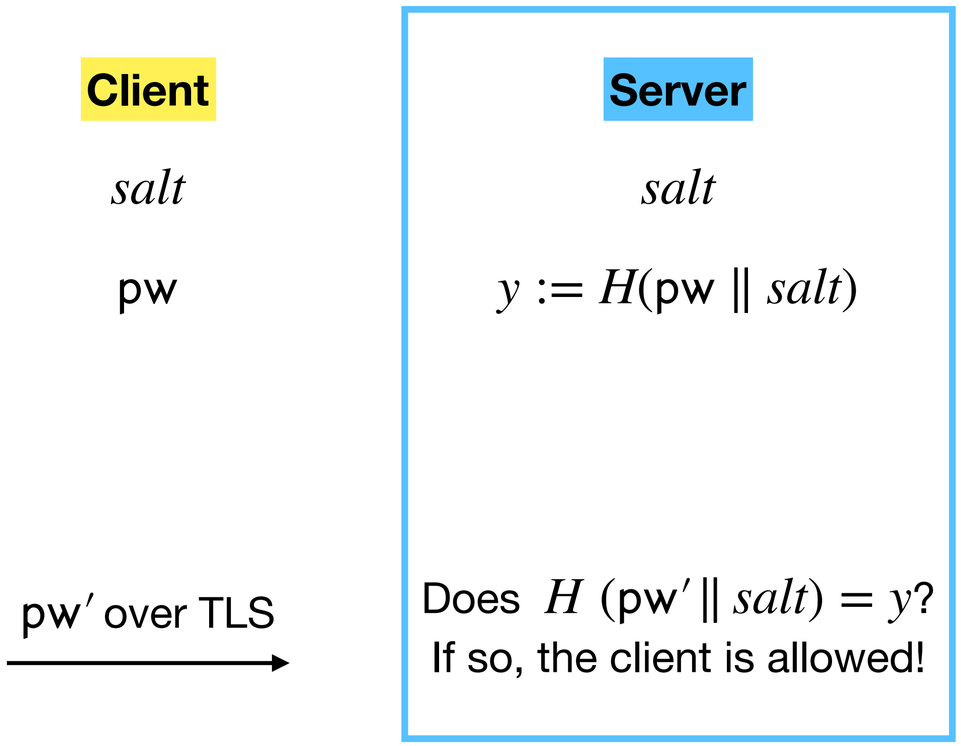
- ☑ Definitions: Undetectability, Exclusivity, Utility
- ☑ Construction of a Public-ASA
- ☐ **One application**

☐ Concluding remarks on the general case

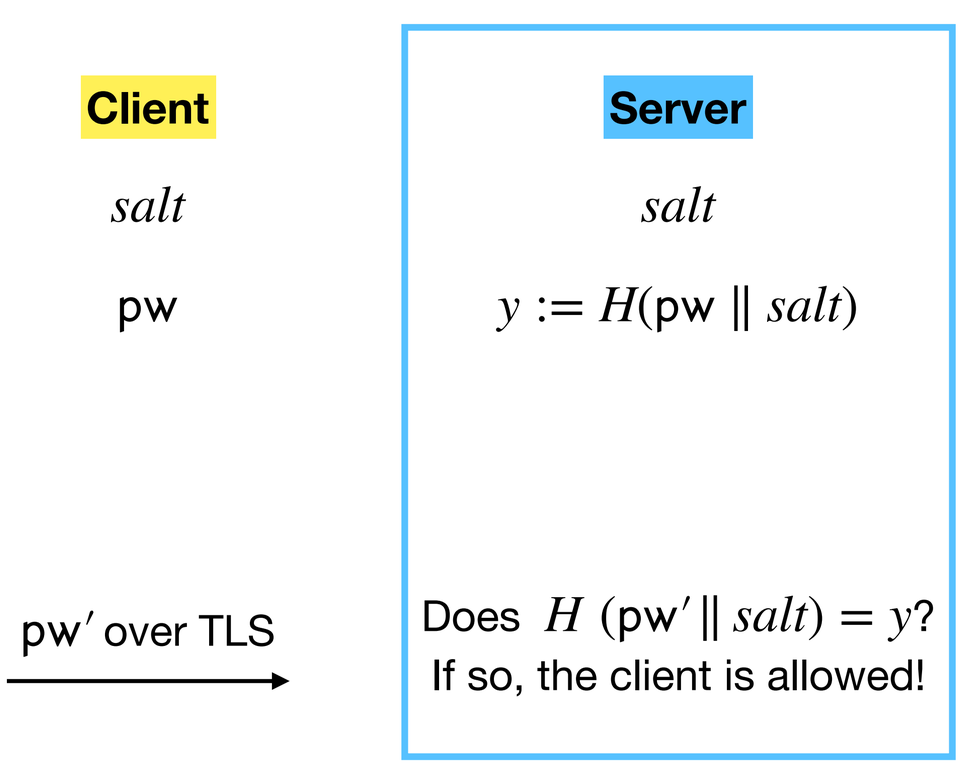
Application: Password-based authentication

Client	Server
$salt$	$salt$
pw	$y := H(\text{pw} \,\|\|\, salt)$

Application: Password-based authentication



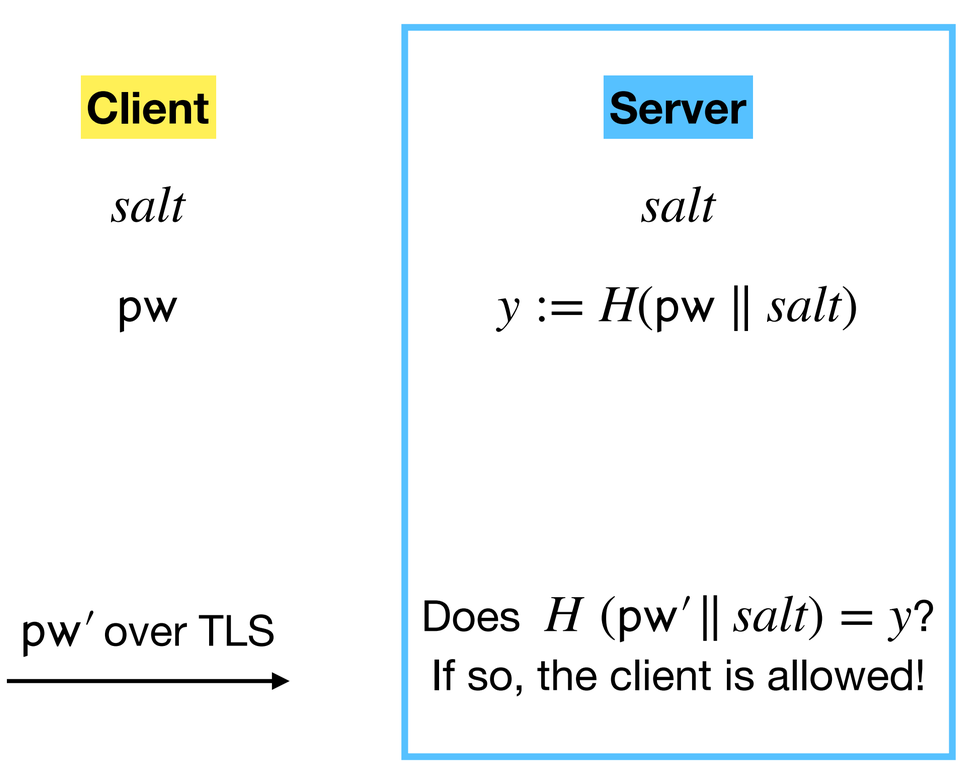
Application: Password-based authentication



The Public-ASA attack

0. The attacker learns the Server's $(salt, y)$

Application: Password-based authentication

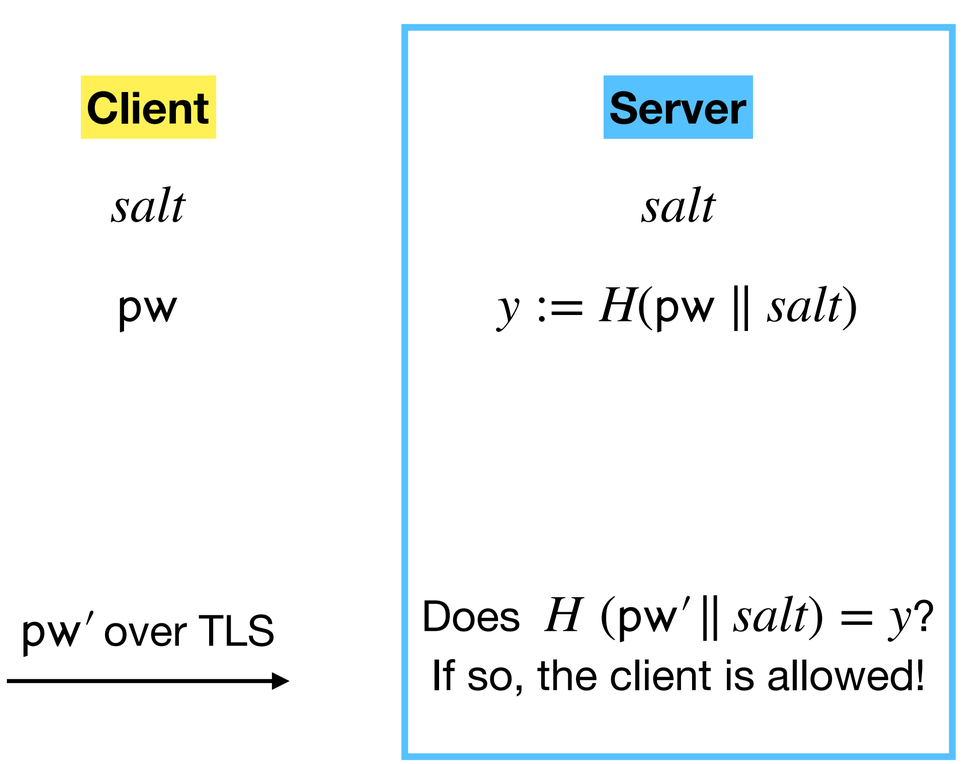


The Public-ASA attack

0. The attacker learns the Server's $(salt, y)$

1. The attacker generates $(\widetilde{H}, \mathsf{Expl}) \xleftarrow{\$} \mathsf{SubGen}(H)$

Application: Password-based authentication



The Public-ASA attack

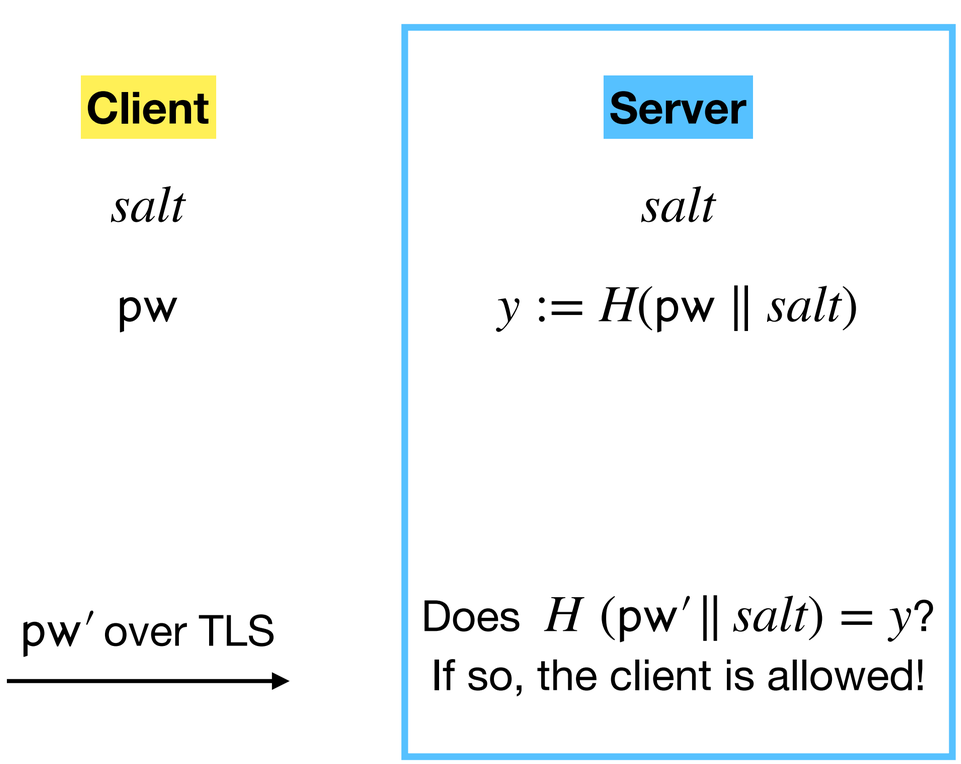
0. The attacker learns the Server's $(salt, y)$

1. The attacker generates $(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$

For our construction, the attacker must select an **SUF signature scheme**

AND give an **invertible embedding function** for the desired structure.

Application: Password-based authentication



The Public-ASA attack

0. The attacker learns the Server's $(salt, y)$

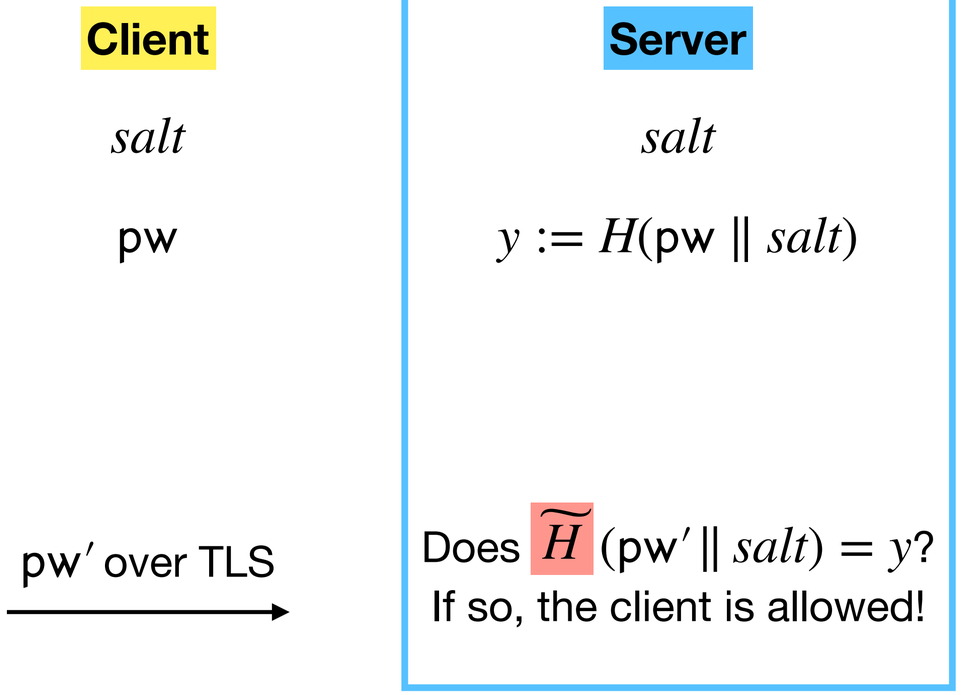
1. The attacker generates $(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$

For our construction, the attacker must select
an **SUF signature scheme**

AND give an **invertible embedding function**
for the desired structure.

Here, that a hash input $x = \text{pw}^* \| salt$
has the particular salt as a suffix.

Application: Password-based authentication



The Public-ASA attack

0. The attacker learns the Server's $(salt, y)$

1. The attacker generates $(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$

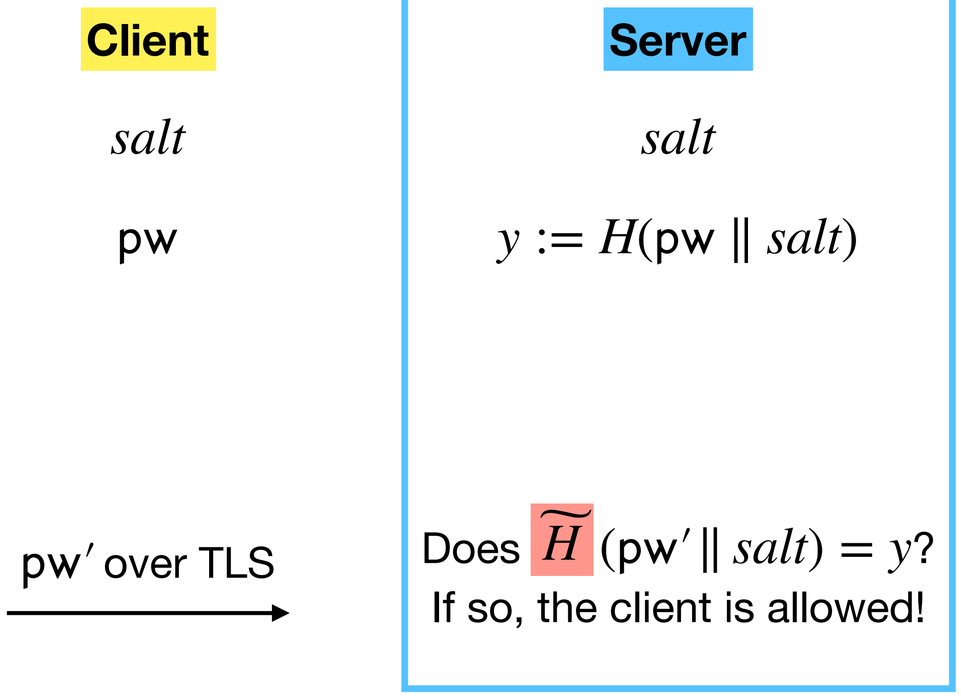
For our construction, the attacker must select
an **SUF signature scheme**

AND give an **invertible embedding function**
for the desired structure.

Here, that a hash input $x = \text{pw}^* \| salt$
has the particular salt as a suffix.

2. Through some means, installs $\widetilde{H}$ as the Server's hash function

Application: Password-based authentication



The Public-ASA attack

0. The attacker learns the Server's $(salt, y)$

1. The attacker generates $(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$

For our construction, the attacker must select
an **SUF signature scheme**

AND give an **invertible embedding function**
for the desired structure.

Here, that a hash input $x = \text{pw}^* \| salt$
has the particular salt as a suffix.

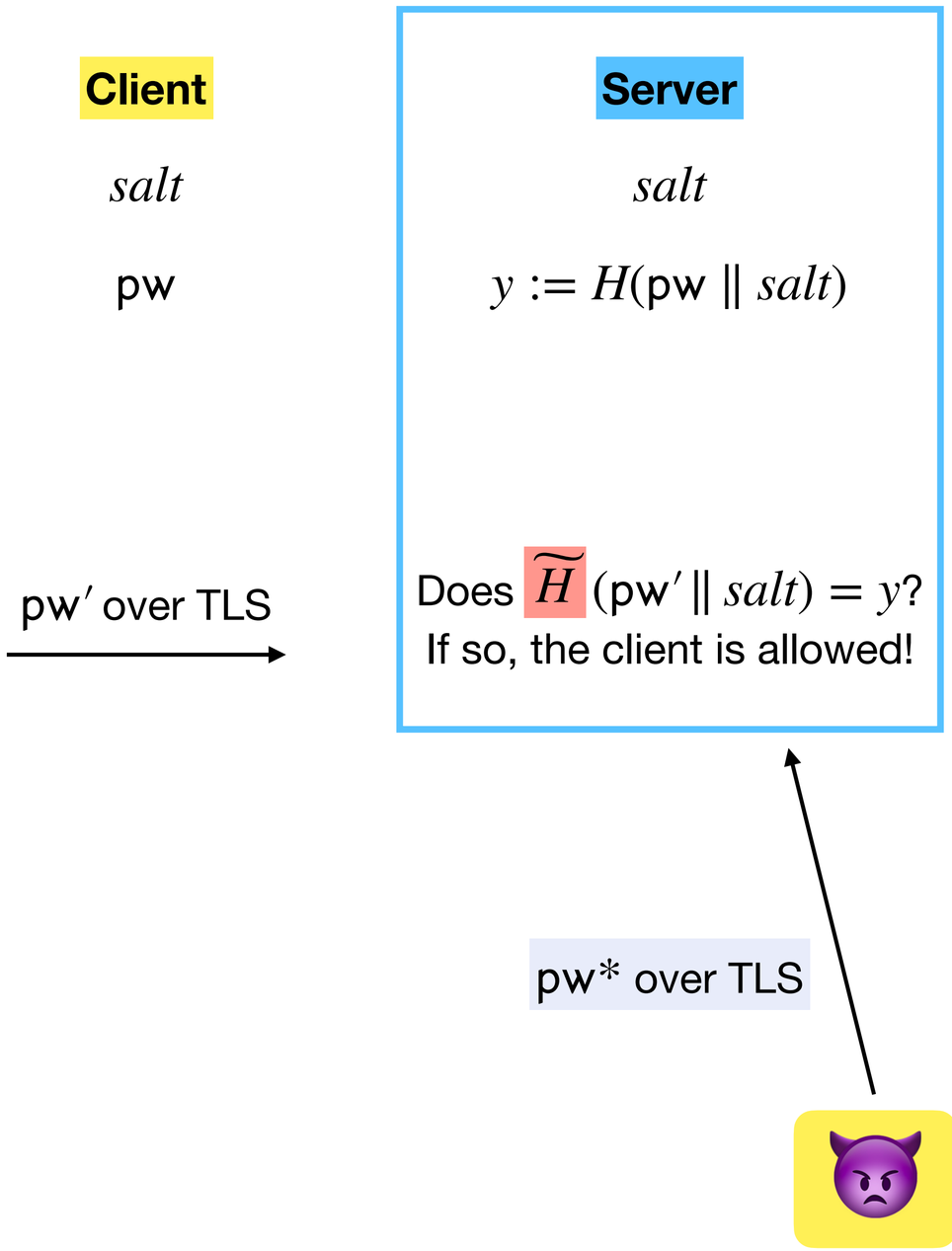
2. Through some means, installs $\widetilde{H}$ as the Server's hash function

3. The attacker uses Expl to find a preimage $x = \text{pw}^* \| salt$ which

(i) Has the particular $salt$ as a suffix

(ii) AND satisfies $\widetilde{H}(\text{pw}^* \| salt) = y$, for the Server's y

Application: Password-based authentication



The Public-ASA attack

0. The attacker learns the Server's $(salt, y)$

1. The attacker generates $(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$

For our construction, the attacker must select an **SUF signature scheme**

AND give an **invertible embedding function** for the desired structure.

Here, that a hash input $x = \text{pw}^* \| salt$ has the particular salt as a suffix.

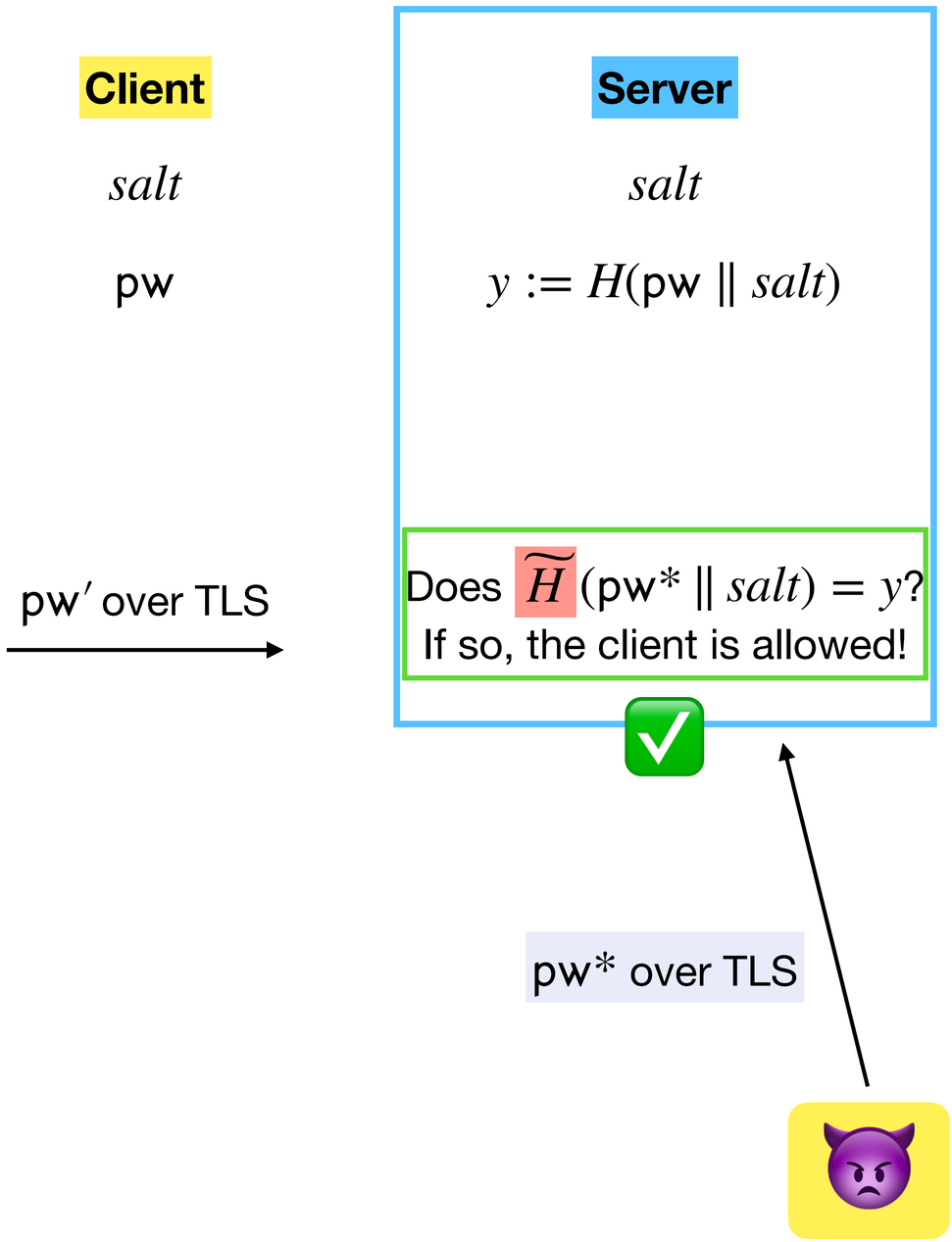
2. Through some means, installs $\widetilde{H}$ as the Server's hash function

3. The attacker uses Expl to find a preimage $x = \text{pw}^* \| salt$ which

(i) Has the particular $salt$ as a suffix

(ii) AND satisfies $\widetilde{H}(\text{pw}^* \| salt) = y$, for the Server's y

Application: Password-based authentication



The Public-ASA attack

0. The attacker learns the Server's $(salt, y)$

1. The attacker generates $(\widetilde{H}, \text{Expl}) \xleftarrow{\$} \text{SubGen}(H)$

For our construction, the attacker must select an **SUF signature scheme**

AND give an **invertible embedding function** for the desired structure.

Here, that a hash input $x = \text{pw}^* \,\|\, salt$ has the particular salt as a suffix.

2. Through some means, installs $\widetilde{H}$ as the Server's hash function

3. The attacker uses Expl to find a preimage $x = \text{pw}^* \,\|\, salt$ which

(i) Has the particular $salt$ as a suffix

(ii) AND satisfies $\widetilde{H}(\text{pw}^* \,\|\, salt) = y$, for the Server's y

Application: Password-based authentication

Client

$salt$

pw

Server

$salt$

$y :=$

Does $\widetilde{H}$

If so, th

pw′ over TLS

pw* over TLS

The Public-ASA attack

0. The attacker learns the Server's $(salt, y)$

$\widetilde{\quad}$ $\xleftarrow{\$}$ SubGen(H)

acker must select

scheme

edding function

ucture.

a hash input $x = \text{pw}^* \,||\, salt$

particular salt as a suffix.

the Server's hash function

3. The attacker uses Expl to find a preimage $x = \text{pw}^* \,||\, salt$ which

(i) Has the particular $salt$ as a suffix

(ii) AND satisfies $\widetilde{H}(\text{pw}^* \,||\, salt) = y$, for the Server's y

- Realistically, **Utility** *should* care about structure, and how information is embedded. (See paper for actual definitions.)
- A second example in the paper is about forging X.509 certificates, where it is again easy to imagine the attacker wants a particular structure.
- There, the Expl( ) oracle in **Exclusivity** is also motivated.

☑ Introductory picture & overview of results

Public-Algorithm Substitution Attacks on Hash functions

- ☑ Definitions: Undetectability, Exclusivity, Utility
- ☑ Construction of a Public-ASA
- ☑ One application

☐ **Concluding remarks on the general case**

Recall our setting is ANY public algorithm… What other applications are there?

Recall our setting is ANY public algorithm… What other applications are there?

▶️ Signature subversion:

Prior work [AMV15, CS03, TBEL21, …] gives **Secret-ASAs** which work on randomized schemes only.

A **Public-ASA** on verification applies to any scheme, including deterministic ones, as follows:

Recall our setting is ANY public algorithm… What other applications are there?

Signature subversion:

Prior work [AMV15, CS03, TBEL21, …] gives **Secret-ASAs** which work on randomized schemes only.

A **Public-ASA** on verification applies to any scheme, including deterministic ones, as follows:

Target public algorithm: $\mathsf{Vfy} : (vk, m, \sigma) \mapsto 0 \text{ or } 1$

Recall our setting is ANY public algorithm… What other applications are there?

Signature subversion:

Prior work [AMV15, CS03, TBEL21, …] gives **Secret-ASAs** which work on randomized schemes only.

A **Public-ASA** on verification applies to any scheme, including deterministic ones, as follows:

Target public algorithm: $\mathsf{Vfy} : (vk, m, \sigma) \mapsto 0 \text{ or } 1$

Target output: 1

Recall our setting is ANY public algorithm… What other applications are there?

Signature subversion:

Prior work [AMV15, CS03, TBEL21, …] gives **Secret-ASAs** which work on randomized schemes only.

A **Public-ASA** on verification applies to any scheme, including deterministic ones, as follows:

Target public algorithm: $\mathsf{Vfy} : (vk, m, \sigma) \mapsto 0 \text{ or } 1$

Target output: 1

Structure: The preimage (vk, m, σ) contains a desired vk and m for a forgery.

Recall our setting is ANY public algorithm… What other applications are there?

▶️ Signature subversion:

Prior work [AMV15, CS03, TBEL21, …] gives **Secret-ASAs** which work on randomized schemes only.

A **Public-ASA** on verification applies to any scheme, including deterministic ones, as follows:

Target public algorithm: $\mathsf{Vfy} : (vk, m, \sigma) \mapsto 0 \text{ or } 1$

Target output: 1

Structure: The preimage (vk, m, σ) contains a desired vk and m for a forgery.

▶️ Non-Interactive Argument (NIA / NIZK) subversion:

Prior work [BFS16, F18] considers malicious CRS or **Secret-ASAs** [CGS23].

A **Public-ASA** applies to verification, as:

Recall our setting is ANY public algorithm… What other applications are there?

Signature subversion:

Prior work [AMV15, CS03, TBEL21, …] gives **Secret-ASAs** which work on randomized schemes only.

A **Public-ASA** on verification applies to any scheme, including deterministic ones, as follows:

Target public algorithm: $\mathsf{Vfy} : (vk, m, \sigma) \mapsto 0 \text{ or } 1$

Target output: 1

Structure: The preimage (vk, m, σ) contains a desired vk and m for a forgery.

Non-Interactive Argument (NIA / NIZK) subversion:

Prior work [BFS16, F18] considers malicious CRS or **Secret-ASAs** [CGS23].

A **Public-ASA** applies to verification, as:

Target public algorithm: $\mathsf{Vfy} : (\phi, \pi) \mapsto 0 \text{ or } 1$

Target output: 1

Structure: The preimage (ϕ, π) contains a desired (false) statement ϕ to forge.

Recall our setting is ANY public algorithm… What other applications are there?

Signature subversion:

Prior work [AMV15, CS03, TBEL21, …] gives **Secret-ASAs** which work on randomized schemes only.

A **Public-ASA** on verification applies to any scheme, including deterministic ones, as follows:

Target public algorithm: Vfy : $(vk, m, \sigma) \mapsto 0 \text{ or } 1$

Target output: 1

Structure: The preimage (vk, m, σ) contains a desired vk and m for a forgery.

Non-Interactive Argument (NIA / NIZK) subversion:

Prior work [BFS16, F18] considers malicious CRS or **Secret-ASAs** [CGS23].

A **Public-ASA** applies to verification, as:

Target public algorithm: Vfy : $(\phi, \pi) \mapsto 0 \text{ or } 1$

Target output: 1

Structure: The preimage (ϕ, π) contains a desired (false) statement ϕ to forge.

Subversion of other public algorithms?

For example, GKVZ22 considered a Machine Learning classifier.

Target public algorithm: Classify : $x \mapsto -1 \text{ or } +1$

Target output: +1 or -1

Structure: The preimage x is “close to” a desired x'.

Topic of this talk:

Public-Algorithm Substitution Attack (*P*-ASA)

Summary of contributions:

1. Give a definition for an ASA on a *public* algorithm.
2. Design a construction satisfying the definition.

3. Look in more detail at important applications:

   Hash functions, as used in certificates or password-based authentication.

   Verification functions, in signatures.

   Verification functions, in Non-Interactive Arguments.

Topic of this talk:

Public-Algorithm Substitution Attack (*P*-ASA)

Summary of contributions:

1. Give a definition for an ASA on a *public* algorithm.

2. Design a construction satisfying the definition.

3. Look in more detail at important applications:

   Hash functions, as used in certificates or password-based authentication.

   Verification functions, in signatures.

   Verification functions, in Non-Interactive Arguments.

Thank you for listening! Any questions?

ePrint 2024 / 536