Dictators? Friends? Forgers.

Breaking and Fixing Unforgeability Definitions for Anamorphic Signature Schemes

Joseph Jaeger and Roy Stracovsky

ASIACRYPT 2024

Georgia Institute of Technology

Motivation

Anamorphic Encryption [PPY22]

- Proposed by Persiano, Phan, and Yung at Eurocrypt 2022.

- **Goal:** allow users to communicate privately in authoritarian settings by concealing hidden messages inside of innocuous ciphertexts.

- **Technical realization:** augment *deployed* primitives with “anamorphic extensions” that use a double key dk to conceal “anamorphic messages”.

Anamorphic Encryption [PPY22]

- Proposed by Persiano, Phan, and Yung at Eurocrypt 2022.
- **Goal:** allow users to communicate privately in authoritarian settings by concealing hidden messages inside of innocuous ciphertexts.
- **Technical realization:** augment *deployed* primitives with "anamorphic extensions" that use a double key **dk** to conceal "anamorphic messages".

$\mathsf{KeyGen}(1^\lambda) \Rightarrow (\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}(\mathsf{pk}, \mathsf{msg}) \Rightarrow \mathsf{ct}$

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \Rightarrow \mathsf{msg}$

Hashed ElGamal,
RSA-OAEP, …

Anamorphic Encryption [PPY22]

- Proposed by Persiano, Phan, and Yung at Eurocrypt 2022.

- **Goal:** allow users to communicate privately in authoritarian settings by concealing hidden messages inside of innocuous ciphertexts.

- **Technical realization:** augment *deployed* primitives with “anamorphic extensions” that use a double key **dk** to conceal “anamorphic messages”.

Hashed ElGamal, RSA-OAEP, …

$\mathsf{KeyGen}(1^\lambda) \Rightarrow (\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}(\mathsf{pk}, \mathsf{msg}) \Rightarrow \mathsf{ct}$

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \Rightarrow \mathsf{msg}$

looks like Hashed ElGamal, RSA-OAEP, …

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{pk}, \mathsf{sk}, \mathsf{dk})$

$\mathsf{aEnc}(\mathsf{pk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg}) \Rightarrow \mathsf{act}$

$\mathsf{aDec}(\mathsf{sk}, \mathsf{dk}, \mathsf{act}) \Rightarrow \mathsf{amsg}$

Anamorphic Encryption [PPY22]

- Proposed by Persiano, Phan, and Yung at Eurocrypt 2022.

- **Goal:** allow users to communicate privately in authoritarian settings by concealing hidden messages inside of innocuous ciphertexts.

- **Technical realization:** augment *deployed* primitives with "anamorphic extensions" that use a double key **dk** to conceal "anamorphic messages".

$\mathsf{KeyGen}(1^\lambda) \Rightarrow (\mathsf{pk}, \mathsf{sk})$

Hashed ElGamal, RSA-OAEP, …

$\mathsf{Enc}(\mathsf{pk}, \mathsf{msg}) \Rightarrow \boxed{\mathsf{ct} \approx_c \mathsf{act}}$

even to dictator who knows **sk**

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \Rightarrow \mathsf{msg}$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{pk}, \mathsf{sk}, \mathsf{dk})$

looks like Hashed ElGamal, RSA-OAEP, …

$\mathsf{aEnc}(\mathsf{pk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg}) \Rightarrow \mathsf{act}$

$\mathsf{aDec}(\mathsf{sk}, \mathsf{dk}, \mathsf{act}) \Rightarrow \mathsf{amsg}$

Anamorphic Signature Schemes [KPPYZ23]

- Proposed by Kutylowski, Persiano, Phan, Yung, and Zawada at Crypto 2023.

- **Core idea:** expand available stealthy channel bandwidth by concealing anamorphic messages in signatures.

Anamorphic Signature Schemes [KPPYZ23]

- Proposed by Kutylowski, Persiano, Phan, Yung, and Zawada at Crypto 2023.
- **Core idea:** expand available stealthy channel bandwidth by concealing anamorphic messages in signatures.

$\mathsf{KeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk})$

ElGamal signatures, RSA-PSS, …

$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}) \Rightarrow \mathsf{sig}$

$\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \Rightarrow \mathsf{accept/reject}$

Anamorphic Signature Schemes [KPPYZ23]

- Proposed by Kutylowski, Persiano, Phan, Yung, and Zawada at Crypto 2023.

- **Core idea:** expand available stealthy channel bandwidth by concealing anamorphic messages in signatures.

$\mathsf{KeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk})$

ElGamal signatures, RSA-PSS, …

$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}) \Rightarrow \mathsf{sig}$

$\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \Rightarrow \mathsf{accept/reject}$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

looks like ElGamal signatures, RSA-PSS, …

$\mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg}) \Rightarrow \mathsf{asig}$

$\mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig}) \Rightarrow \mathsf{amsg}$

Anamorphic Signature Schemes [KPPYZ23]

- Proposed by Kutylowski, Persiano, Phan, Yung, and Zawada at Crypto 2023.

- **Core idea:** expand available stealthy channel bandwidth by concealing anamorphic messages in signatures.

$\mathsf{KeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk})$

ElGamal signatures, RSA-PSS, …

$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}) \Rightarrow \boxed{\mathsf{sig} \approx_c \mathsf{asig}}$

even to dictator who knows sk

$\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \Rightarrow \mathsf{a}$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

looks like ElGamal signatures, RSA-PSS, …

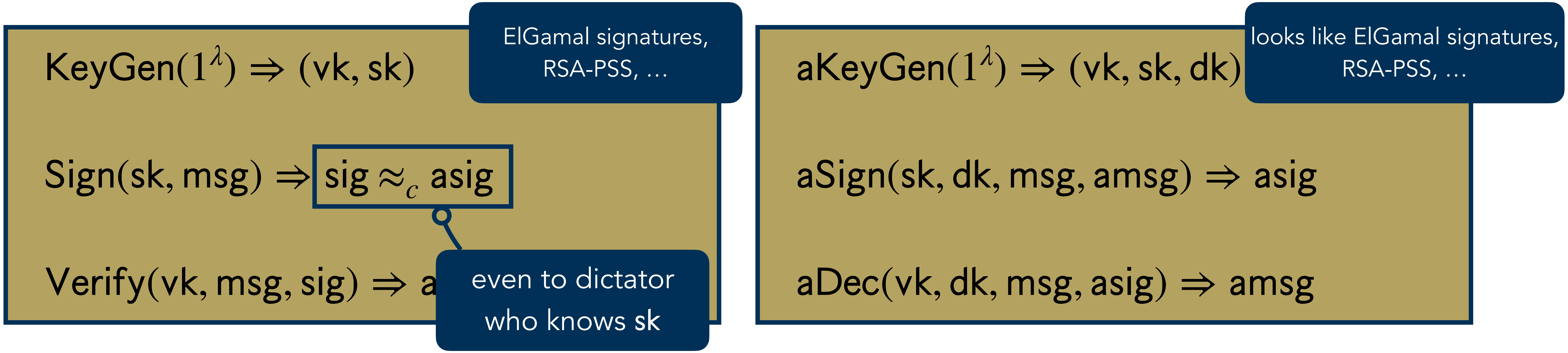
$\mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg}) \Rightarrow \mathsf{asig}$

$\mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig}) \Rightarrow \mathsf{amsg}$

Anamorphic Signature Schemes [KPPYZ23]

- Proposed by Kutylowski, Persiano, Phan, Yung, and Zawada at Crypto 2023.

- **Core idea:** expand available stealthy channel bandwidth by concealing anamorphic messages in signatures.



$\mathsf{KeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk})$

ElGamal signatures, RSA-PSS, …

$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}) \Rightarrow \boxed{\mathsf{sig} \approx_c \mathsf{asig}}$

even to dictator who knows sk

$\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \Rightarrow \mathsf{a}$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

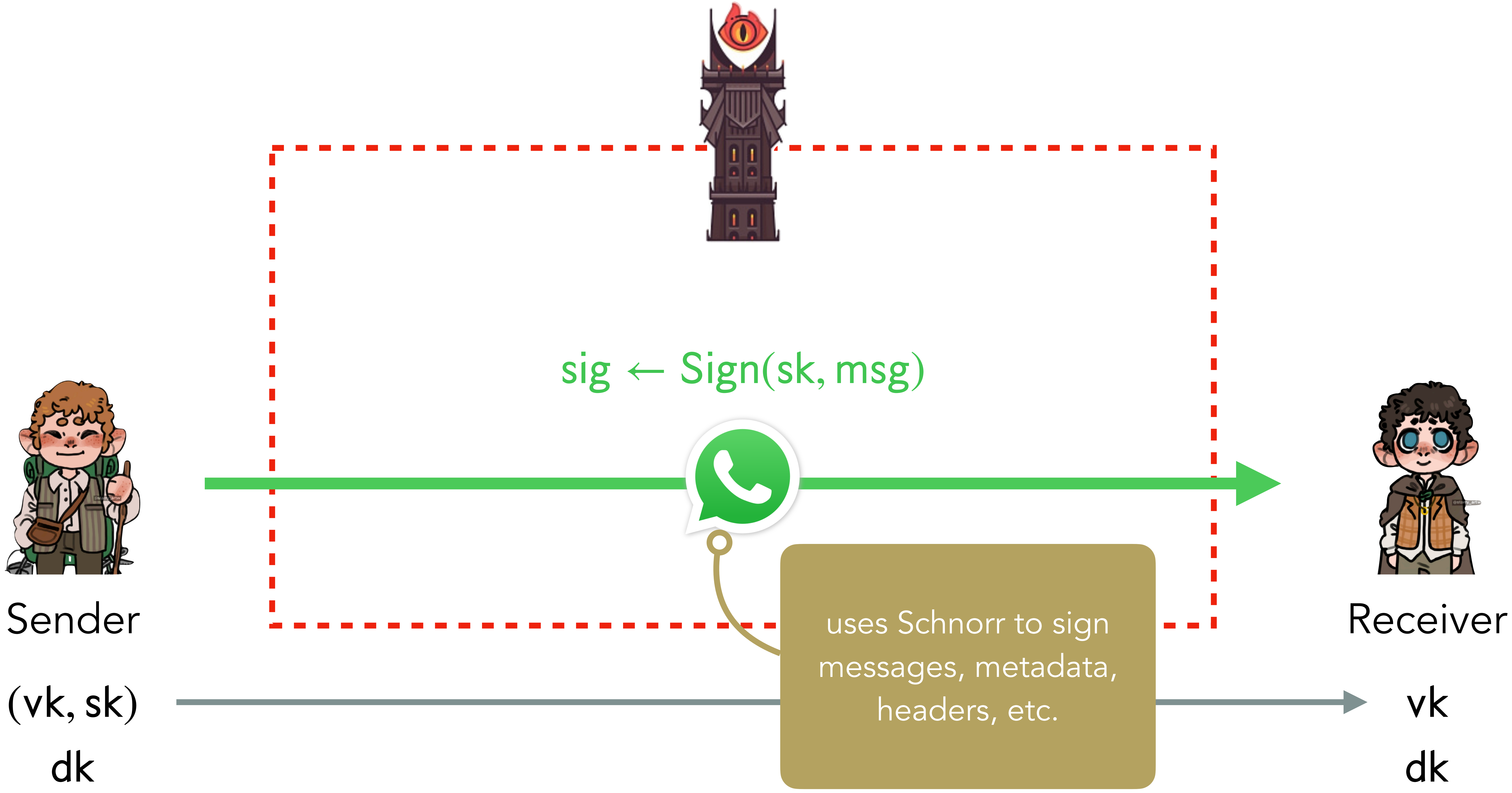
looks like ElGamal signatures, RSA-PSS, …

$\mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg}) \Rightarrow \mathsf{asig}$

$\mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig}) \Rightarrow \mathsf{amsg}$

- Introduce new security definitions specific to (anamorphic) signatures.

Anamorphic Signatures, Deployed



sig ← Sign(sk, msg)

Sender

(vk, sk)

dk

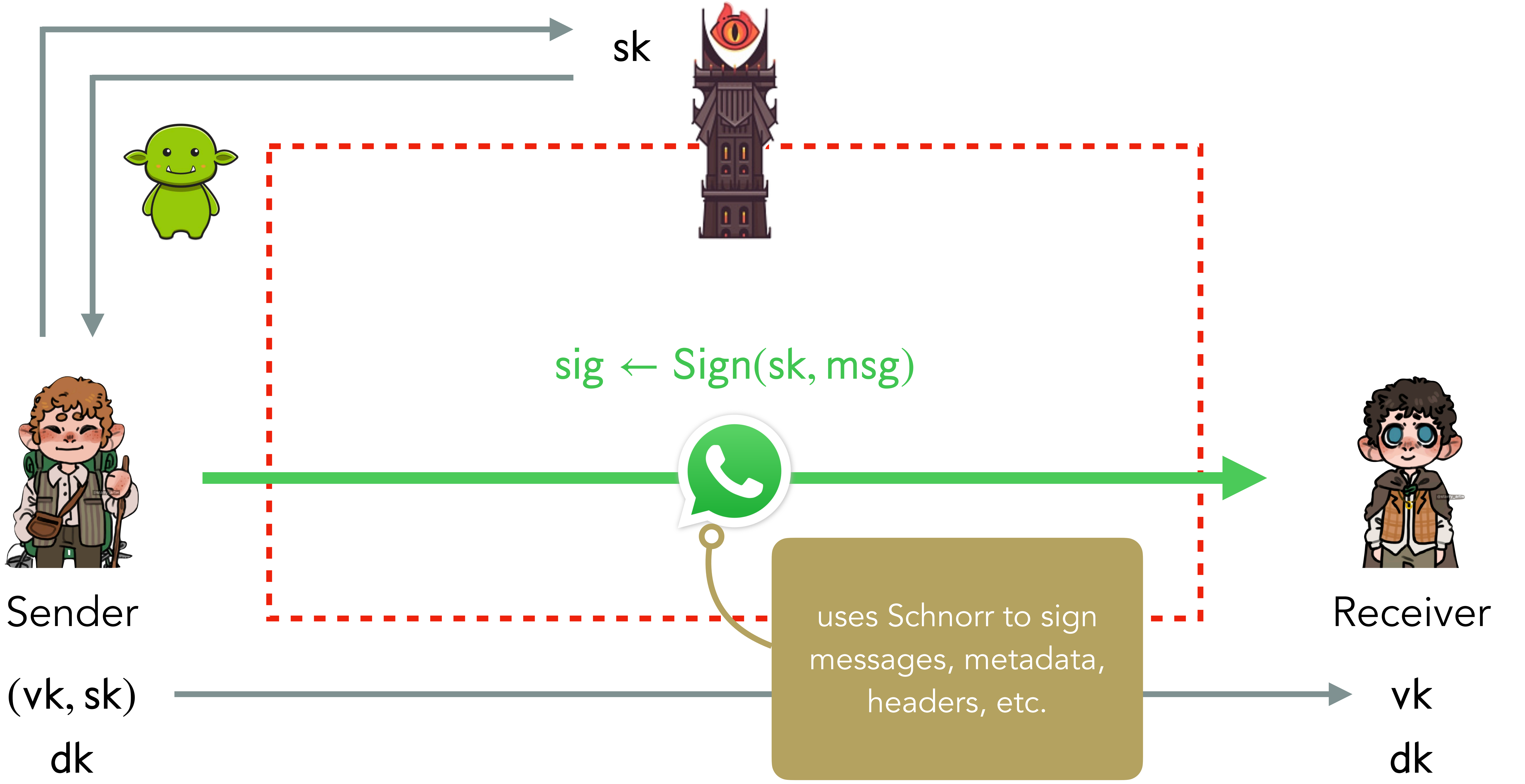
Receiver

vk

dk

uses Schnorr to sign messages, metadata, headers, etc.

Anamorphic Signatures, Deployed



sk

sig ← Sign(sk, msg)

Sender

(vk, sk)

dk

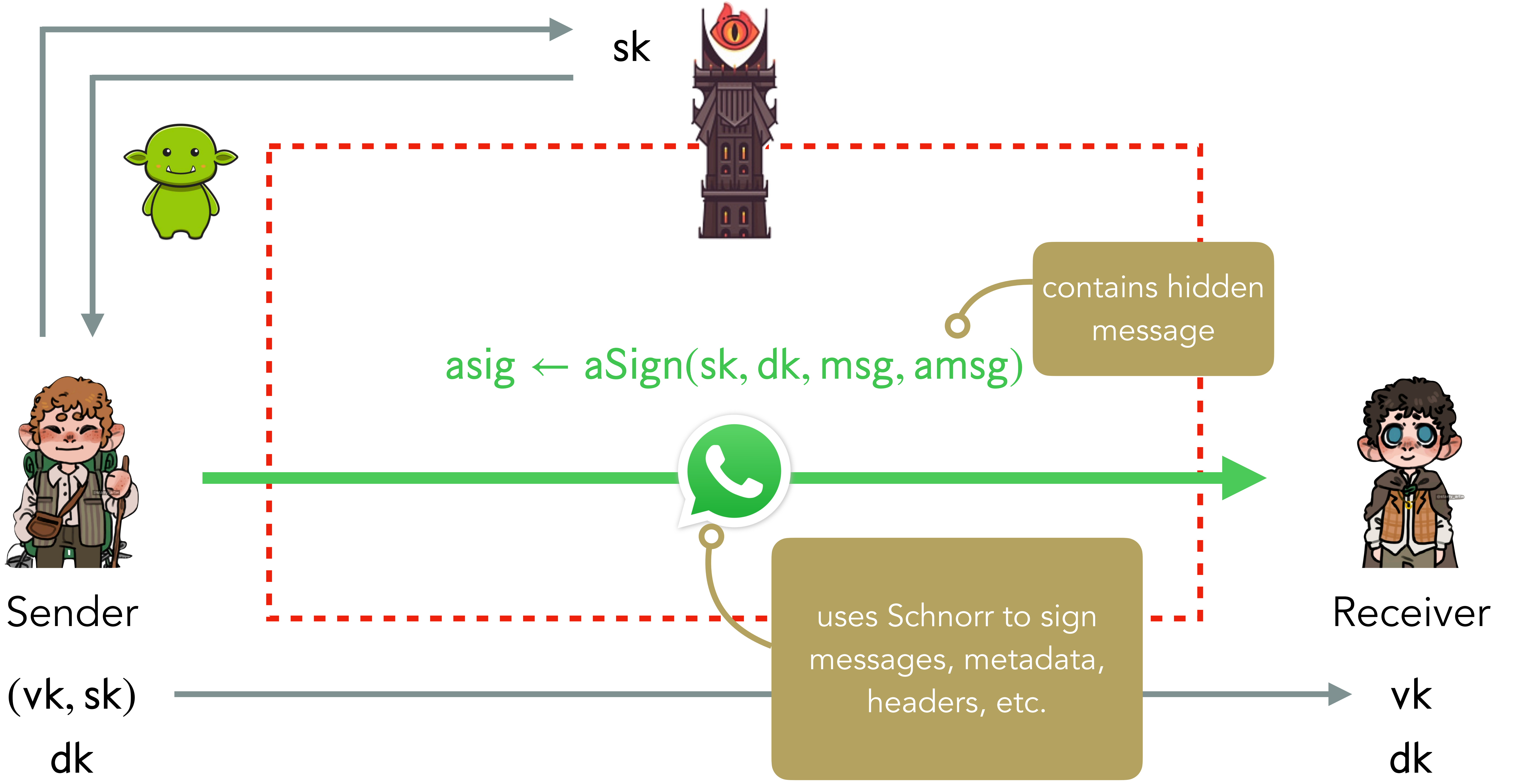
Receiver

vk

dk

uses Schnorr to sign messages, metadata, headers, etc.

Anamorphic Signatures, Deployed



$\text{asig} \leftarrow \text{aSign}(\text{sk}, \text{dk}, \text{msg}, \text{amsg})$

contains hidden message

uses Schnorr to sign messages, metadata, headers, etc.

Warmup Anamorphic Signature Scheme (RSA-PSS)

```
$\mathsf{KeyGen}(1^\lambda)$ :
    $(e, d, N) \leftarrow \mathsf{RSA.KeyGen}(1^\lambda)$
    $\mathsf{vk} \leftarrow (e, N)$
    $\mathsf{sk} \leftarrow (d, N)$
    $\mathbf{return}\ (\mathsf{vk}, \mathsf{sk})$
```

```
$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$ :
    $r \overset{\$}{\leftarrow} \{0,1\}^{\lambda_0}$
    $w \leftarrow H(\mathsf{msg}, r)$
    $\alpha \leftarrow G_1(w) \oplus r$
    $\gamma \leftarrow G_2(w)$
    $\mathsf{sig} \leftarrow (0 \| w \| \alpha \| \gamma)^d \pmod{N}$
    $\mathbf{return}\ \mathsf{sig}$
```

Warmup Anamorphic Signature Scheme (RSA-PSS)

```
$\mathsf{aKeyGen}(1^\lambda)$ :
    $(e, d, N) \leftarrow \mathsf{RSA.KeyGen}(1^\lambda)$
    $\mathsf{vk} \leftarrow (e, N)$
    $\mathsf{sk} \leftarrow (d, N)$
    $\mathsf{dk} \leftarrow \mathsf{prE.KeyGen}(1^\lambda)$
    $\mathbf{return}\ (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$
```

```
$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$ :
    $r \overset{\$}{\leftarrow} \{0,1\}^{\lambda_0}$
    $w \leftarrow H(\mathsf{msg}, r)$
    $\alpha \leftarrow G_1(w) \oplus r$
    $\gamma \leftarrow G_2(w)$
    $\mathsf{sig} \leftarrow (0\|w\|\alpha\|\gamma)^d \pmod N$
    $\mathbf{return}\ \mathsf{sig}$
```

Warmup Anamorphic Signature Scheme (RSA-PSS)

```
aKeyGen($1^\lambda$) :
    $(e, d, N) \leftarrow$ RSA . KeyGen($1^\lambda$)
    vk $\leftarrow (e, N)$
    sk $\leftarrow (d, N)$
    dk $\leftarrow$ prE . KeyGen($1^\lambda$)
    return (vk, sk, dk)
```

```
aSign(sk, dk, msg, amsg) :
    $r \leftarrow$ prE . Enc(dk, amsg)
    $w \leftarrow H(\text{msg}, r)$
    $\alpha \leftarrow G_1(w) \oplus r$
    $\gamma \leftarrow G_2(w)$
    sig $\leftarrow (0 \| w \| \alpha \| \gamma)^d \pmod N$
    return sig
```

Warmup Anamorphic Signature Scheme (RSA-PSS)

```
$\mathsf{aKeyGen}(1^\lambda)$ :
    $(e, d, N) \leftarrow \mathsf{RSA.KeyGen}(1^\lambda)$
    $\mathsf{vk} \leftarrow (e, N)$
    $\mathsf{sk} \leftarrow (d, N)$
    $\mathsf{dk} \leftarrow \mathsf{prE.KeyGen}(1^\lambda)$
    $\textbf{return}\ (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$
```

```
$\mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig})$ :
    $(b \| w \| \alpha \| \gamma) \leftarrow \mathsf{asig}^e \pmod{N}$
    $r \leftarrow G_1(w) \oplus \alpha$
    $\mathsf{amsg} \leftarrow \mathsf{prE.Dec}(\mathsf{dk}, r)$
    $\textbf{return}\ \mathsf{amsg}$
```

```
$\mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$ :
    $r \leftarrow \mathsf{prE.Enc}(\mathsf{dk}, \mathsf{amsg})$
    $w \leftarrow H(\mathsf{msg}, r)$
    $\alpha \leftarrow G_1(w) \oplus r$
    $\gamma \leftarrow G_2(w)$
    $\mathsf{sig} \leftarrow (0 \| w \| \alpha \| \gamma)^d \pmod{N}$
    $\textbf{return}\ \mathsf{sig}$
```

Warmup Anamorphic Signature Scheme (RSA-PSS)

```
$\mathsf{aKeyGen}(1^\lambda)$ :
    $(e, d, N) \leftarrow \mathsf{RSA.KeyGen}(1^\lambda)$
    $\mathsf{vk} \leftarrow (e, N)$
    $\mathsf{sk} \leftarrow (d, N)$
    $\mathsf{dk} \leftarrow \mathsf{prE.KeyGen}(1^\lambda)$
    return $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$
```

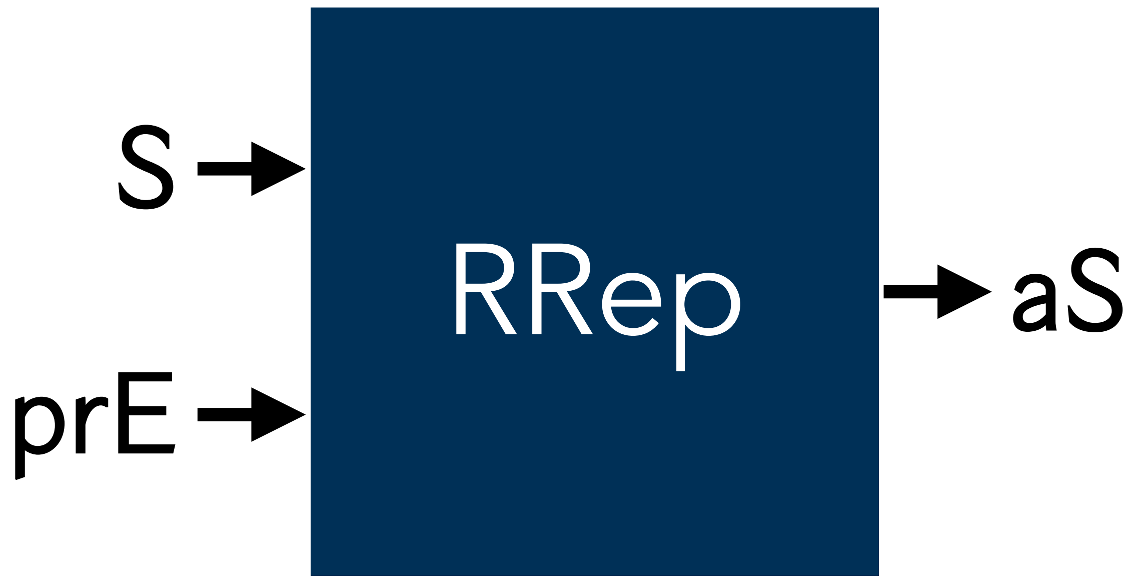
```
$\mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig})$ :
    $(b \| w \| \alpha \| \gamma) \leftarrow \mathsf{asig}^e \pmod{N}$
    $r \leftarrow G_1(w) \oplus \alpha$
    $\mathsf{amsg} \leftarrow \mathsf{prE.Dec}(\mathsf{dk}, r)$
    return $\mathsf{amsg}$
```

```
$\mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$ :
    $r \leftarrow \mathsf{prE.Enc}(\mathsf{dk}, \mathsf{amsg})$
    $w \leftarrow H(\mathsf{msg}, r)$
    $\alpha \leftarrow G_1(w) \oplus r$
    $\gamma \leftarrow G_2(w)$
    $\mathsf{sig} \leftarrow (0 \| w \| \alpha \| \gamma)^d \pmod{N}$
    return $\mathsf{sig}$
```

- Generalizes to any signature scheme that with “recoverable” signing randomness.

Randomness Replacement [KPPYZ23]

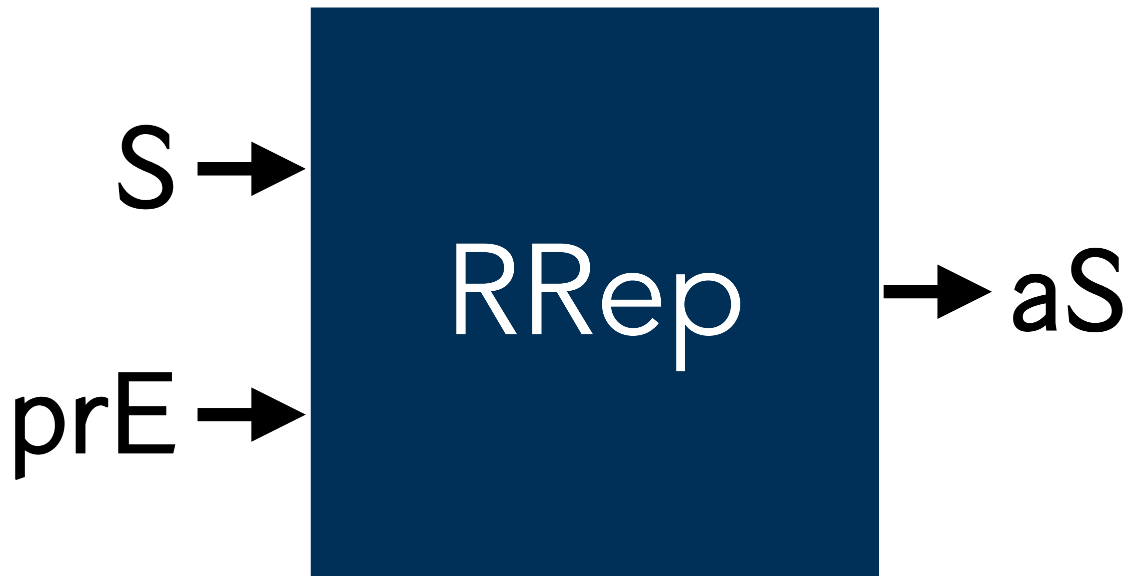
- Randomness Replacement Transform RRep
  - **Input:** randomness recoverable signature scheme S
  - **Input:** pseudorandom encryption scheme prE
  - **Output:** anamorphic signature scheme aS



- S is randomness recoverable if there exists a PPT RRecov that, given $\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r)$, can recover $r \leftarrow \mathsf{RRecov}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$.

Randomness Replacement [KPPYZ23]

- Randomness Replacement Transform **RRep**
  - **Input:** randomness recoverable signature scheme S
  - **Input:** pseudorandom encryption scheme **prE**
  - **Output:** anamorphic signature scheme **aS**



- S is randomness recoverable if there exists a PPT **RRecov** that, given $\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r)$, can recover $r \leftarrow \mathsf{RRecov}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$.

```
$\mathsf{aKeyGen}(1^\lambda):$
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S}.\mathsf{KeyGen}(1^\lambda)$
    $\mathsf{dk} \leftarrow \mathsf{prE}.\mathsf{KeyGen}(1^\lambda)$
    $\mathbf{return}\ (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$
```

```
$\mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg}):$
    $r \leftarrow \mathsf{prE}.\mathsf{Enc}(\mathsf{dk}, \mathsf{amsg})$
    $\mathsf{asig} \leftarrow \mathsf{S}.\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r)$
    $\mathbf{return}\ \mathsf{asig}$
```

```
$\mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig}):$
    $r \leftarrow \mathsf{RRecov}(\mathsf{vk}, \mathsf{msg}, \mathsf{asig})$
    $\mathsf{amsg} \leftarrow \mathsf{prE}.\mathsf{Dec}(\mathsf{dk}, r)$
    $\mathbf{return}\ \mathsf{amsg}$
```

Security Notions for Anamorphic Signatures

Security Notions for Anamorphic Signatures

- **Stealthiness:** dictator cannot distinguish honest and anamorphic signatures even when given keypair $(\mathsf{vk}, \mathsf{sk})$ [KPPYZ23].

Security Notions for Anamorphic Signatures

- **Stealthiness:** dictator cannot distinguish honest and anamorphic signatures even when given keypair $(\mathsf{vk}, \mathsf{sk})$ [KPPYZ23].

- **Robustness:** honest signatures don't anamorphically decrypt to valid anamorphic messages [BGHMR24].

Security Notions for Anamorphic Signatures

- **Stealthiness:** dictator cannot distinguish honest and anamorphic signatures even when given keypair $(\mathsf{vk}, \mathsf{sk})$ [KPPYZ23].

- **Robustness:** honest signatures don't anamorphically decrypt to valid anamorphic messages [BGHMR24].

- **Private anamorphism:** a recipient who knows the double key dk and sees honest signatures cannot forge new signatures [KPPYZ23].

Our Contributions

Summary

Robustness

Private Anamorphism

Summary

Robustness

Private Anamorphism

1 Observe a gap between a stated goal of robustness and its formalization.

Summary

Robustness

Private Anamorphism

1 Observe a gap between a stated goal of robustness and its formalization.

2 Propose **Dictator Unforgeability.**

Summary

Robustness

Private Anamorphism

1 Observe a gap between a stated goal of robustness and its formalization.

2 Propose **Dictator Unforgeability.**

3 Mount a practical attack a previously proposed *robust* anamorphic signature scheme.

Summary

Robustness

1. Observe a gap between a stated goal of robustness and its formalization.
2. Propose **Dictator Unforgeability.**
3. Mount a practical attack a previously proposed *robust* anamorphic signature scheme.
4. Repair other prior anamorphic transforms to achieve dictator unforgeability.

Private Anamorphism

Summary

Robustness

1 Observe a gap between a stated goal of robustness and its formalization.

2 Propose **Dictator Unforgeability.**

3 Mount a practical attack a previously proposed *robust* anamorphic signature scheme.

4 Repair other prior anamorphic transforms to achieve dictator unforgeability.

Private Anamorphism

Observe a gap between the deployment scenario of private anamorphism and its formalization. 5

Summary

Robustness

1 Observe a gap between a stated goal of robustness and its formalization.

2 Propose **Dictator Unforgeability.**

3 Mount a practical attack a previously proposed <u>*robust*</u> anamorphic signature scheme.

4 Repair other prior anamorphic transforms to achieve <u>dictator unforgeability</u>.

Private Anamorphism

5 Observe a gap between the deployment scenario of private anamorphism and its formalization.

6 Propose **Recipient Unforgeability.**

Summary

Robustness

1 Observe a gap between a stated goal of robustness and its formalization.

2 Propose **Dictator Unforgeability.**

3 Mount a practical attack a previously proposed *robust* anamorphic signature scheme.

4 Repair other prior anamorphic transforms to achieve dictator unforgeability.

Private Anamorphism

Observe a gap between the deployment scenario of private anamorphism and its formalization. 5

Propose **Recipient Unforgeability.** 6

Mount a practical attack a natural *private* anamorphic signature scheme. 7

Summary

Robustness

1 Observe a gap between a stated goal of robustness and its formalization.

2 Propose **Dictator Unforgeability.**

3 Mount a practical attack a previously proposed *robust* anamorphic signature scheme.

4 Repair other prior anamorphic transforms to achieve dictator unforgeability.

Private Anamorphism

5 Observe a gap between the deployment scenario of private anamorphism and its formalization.

6 Propose **Recipient Unforgeability.**

7 Mount a practical attack a natural *private* anamorphic signature scheme.

8 Repair (in two ways) a prior anamorphic transform to achieve recipient unforgeability.

Part 1: Strengthening Robustness to Dictator Unforgeability

Robustness

1 Observe a gap between a stated goal of robustness and its formalization.

2 Propose **Dictator Unforgeability.**

3 Mount a practical attack a previously proposed *robust* anamorphic signature scheme.

4 Repair other prior anamorphic transforms to achieve dictator unforgeability.

Private Anamorphism

Observe a gap between the deployment scenario of private anamorphism and its formalization. 5

Propose **Recipient Unforgeability.** 6

Mount a practical attack a natural *private* anamorphic signature scheme. 7

Repair (in two ways) a prior anamorphic transform to achieve recipient unforgeability. 8

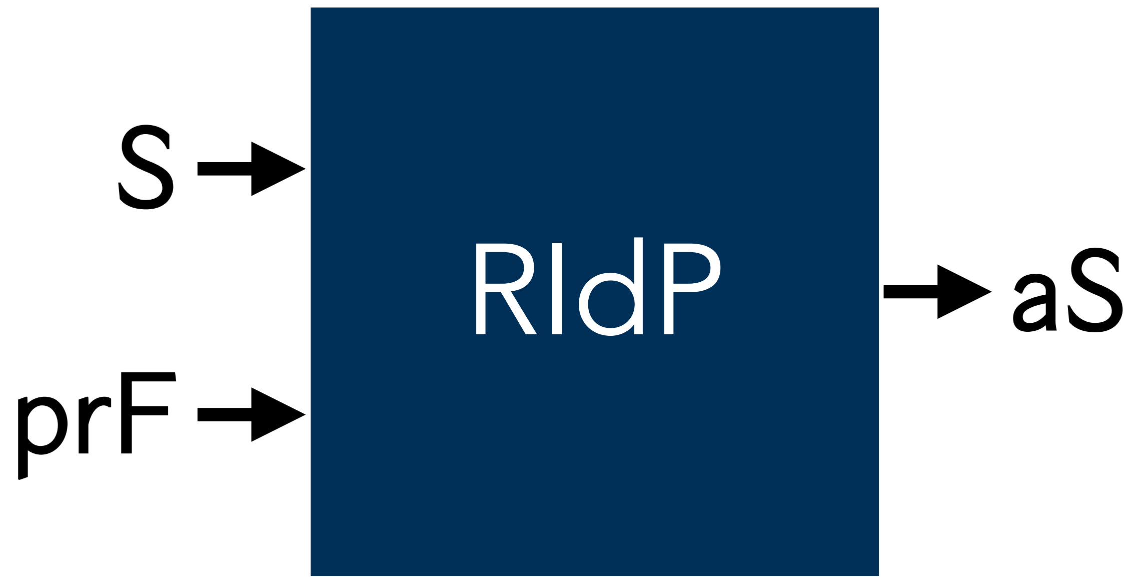
Robustness [BGHMR24]

- Proposed by Banfi, Gegier, Hirt, Maurer, and Rito at Eurocrypt 2024.*
- **High level goal:** honest signatures don't anamorphically decrypt to valid anamorphic messages.
- BGHMR list two primary motivations for robustness.
  - **Usability:** anamorphic messages will be sent in a network containing honest communication — anamorphic users need to identify what is what.
  - **Security:** (roughly) to prevent a dictator from initiating anamorphic channels with anamorphic users.
- BGHMR propose two transforms that achieve robustness.

**Proposed originally for anamorphic encryption though we analyze a straightforward adaptation to signature schemes in our work.*

Randomness Identification with PRF [BGHMR24]

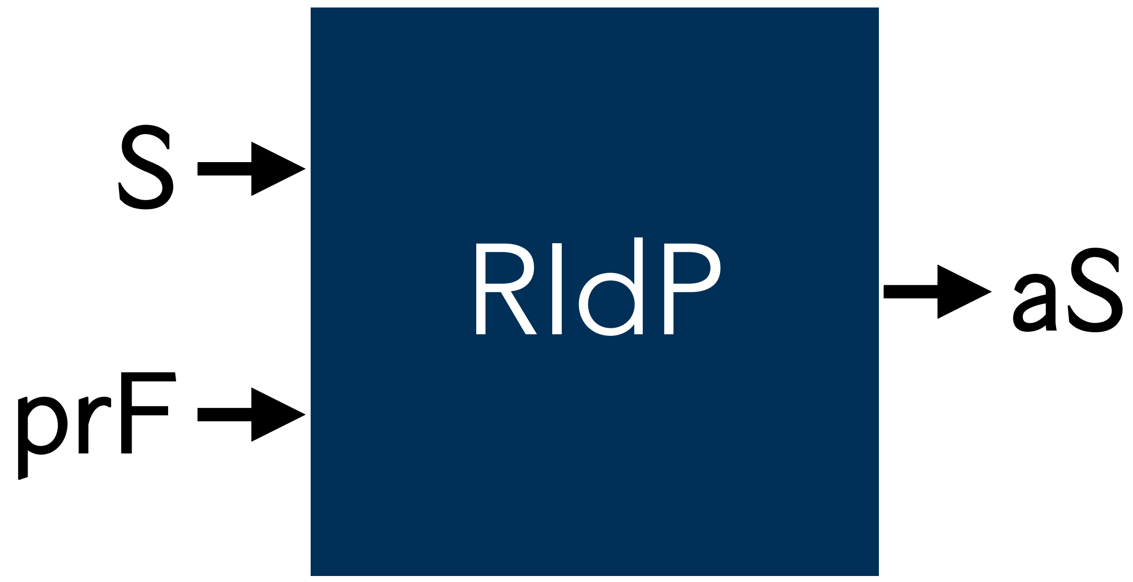
- Randomness Identification with PRF Transform RIdP
  - **Input:** randomness identifying signature scheme S
  - **Input:** pseudorandom function prF
  - **Output:** anamorphic signature scheme aS



- S is randomness identifying if there exists a PPT RIdtfy that, given $\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r)$, can check whether $r' = r$ via $\mathsf{RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}, r')$.

Randomness Identification with PRF [BGHMR24]

- Randomness Identification with PRF Transform **RIdP**
    - **Input:** randomness identifying signature scheme S
    - **Input:** pseudorandom function **prF**
    - **Output:** anamorphic signature scheme **aS**



- S is randomness identifying if there exists a PPT **RIdtfy** that, given $\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r)$, can check whether $r' = r$ via $\mathsf{RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}, r')$.

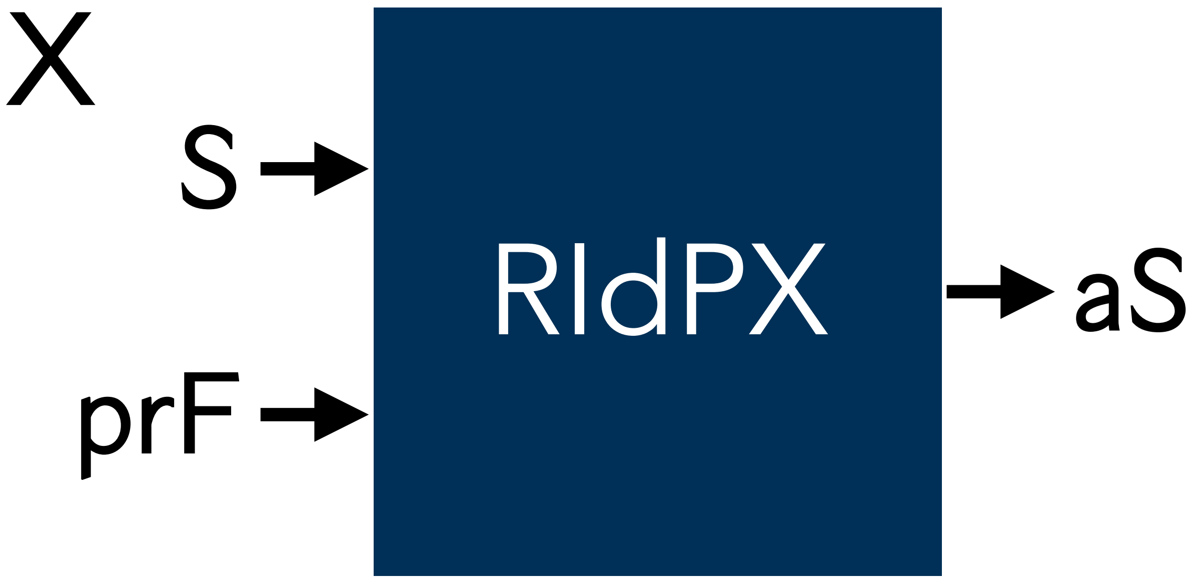
```
aKeyGen$(1^\lambda)$ :
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S}.\mathsf{KeyGen}(1^\lambda)$
    $\mathsf{dk} \leftarrow \mathsf{prF}.\mathsf{KeyGen}(1^\lambda)$
    return $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$
```

```
aSign$(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} : \mathsf{ctr}{+}{+})$ :
    $r \leftarrow \mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}, \mathsf{amsg}))$
    $\mathsf{asig} \leftarrow \mathsf{S}.\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r)$
    return asig
```

```
aDec$(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig} : \mathsf{ctr}{+}{+})$ :
    forall amsg
        $r \leftarrow \mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}, \mathsf{amsg}))$
        if $\mathsf{RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}, \mathsf{r}) = 1$
            return amsg
```

Randomness Identification with PRF/XOR [BGHMR24]

- Randomness Identification with PRF/XOR Transform **RIdPX**
  - **Input:** randomness identifying signature scheme S
  - **Input:** pseudorandom function **prF**
  - **Output:** anamorphic signature scheme **aS**



- S is randomness identifying if there exists a PPT **RIdtfy** that, given sig $\leftarrow$ Sign(sk, msg; r), can check whether $r' = r$ via RIdtfy(vk, msg, sig, r').

```
aKeyGen($1^\lambda$) :
    (vk, sk) $\leftarrow$ S . KeyGen($1^\lambda$)
    dk $\leftarrow$ prF . KeyGen($1^\lambda$)
    return (vk, sk, dk)
```

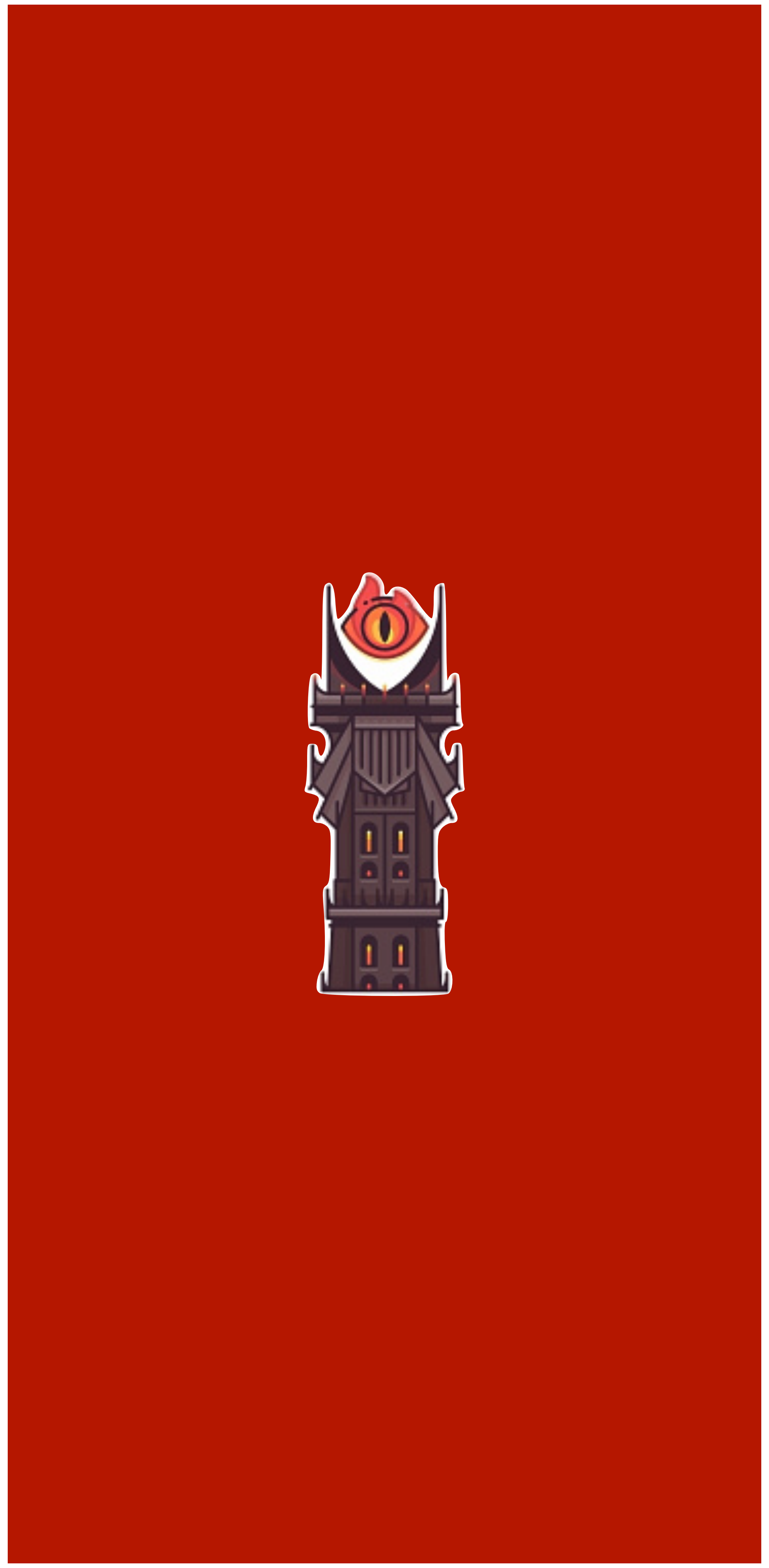
```
aSign(sk, dk, msg, amsg : ctr++ ) :
    $r \leftarrow$ prF(dk, ctr) $\oplus$ amsg
    asig $\leftarrow$ S . Sign(sk, msg; $r$)
    return asig
```

```
aDec(vk, dk, msg, asig : ctr++ ) :
    forall amsg
        $r \leftarrow$ prF(dk, ctr) $\oplus$ amsg
        if RIdtfy(vk, msg, sig, r) $= 1$
            return amsg
```

Robustness Game [BGHMR24]

$b \xleftarrow{\$} \{0,1\}$

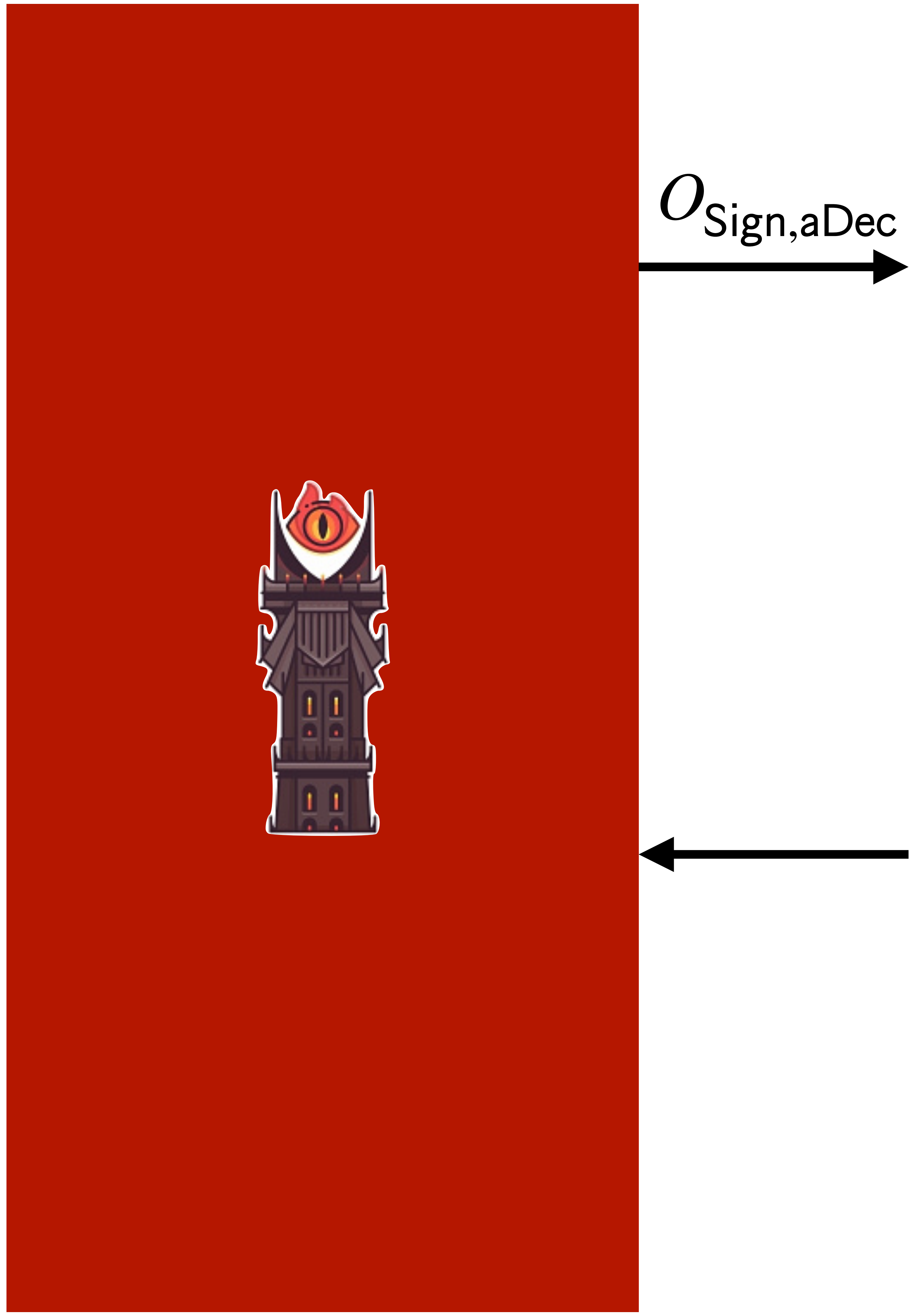
$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$



Robustness Game [BGHMR24]

$b \overset{\$}{\leftarrow} \{0,1\}$

$\mathsf{aKeyGen}(1^{\lambda}) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$



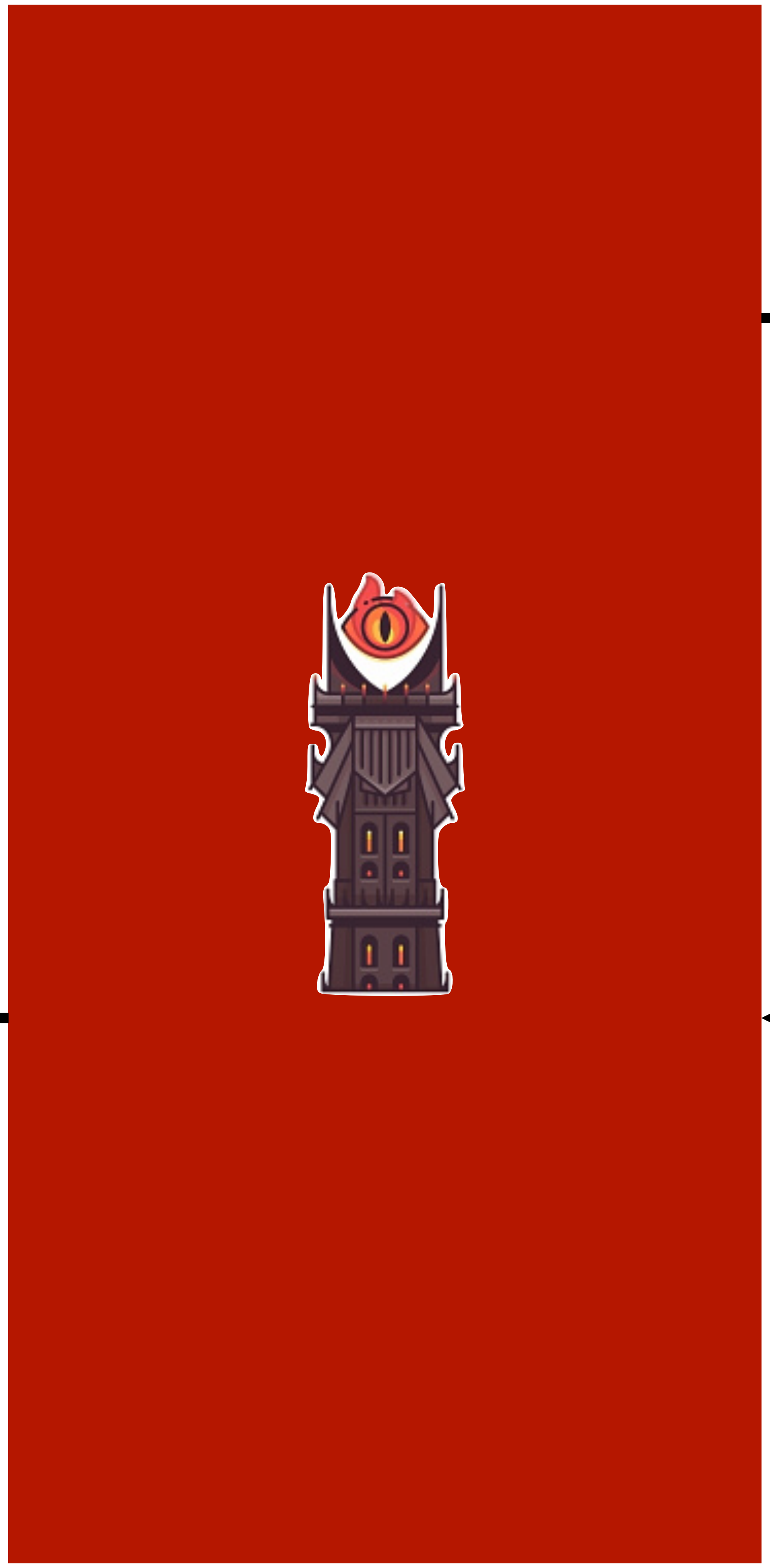
$O_{\mathsf{Sign,aDec}}$

```
msg
if $b = 0$ then
    $\mathsf{amsg} \leftarrow \bot$
if $b = 1$ then
    $\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$
    $\mathsf{amsg} \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{sig})$
amsg
```

Robustness Game [BGHMR24]

$b \xleftarrow{\$} \{0,1\}$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$



$O_{\mathsf{Sign,aDec}}$

$\rightarrow$ msg

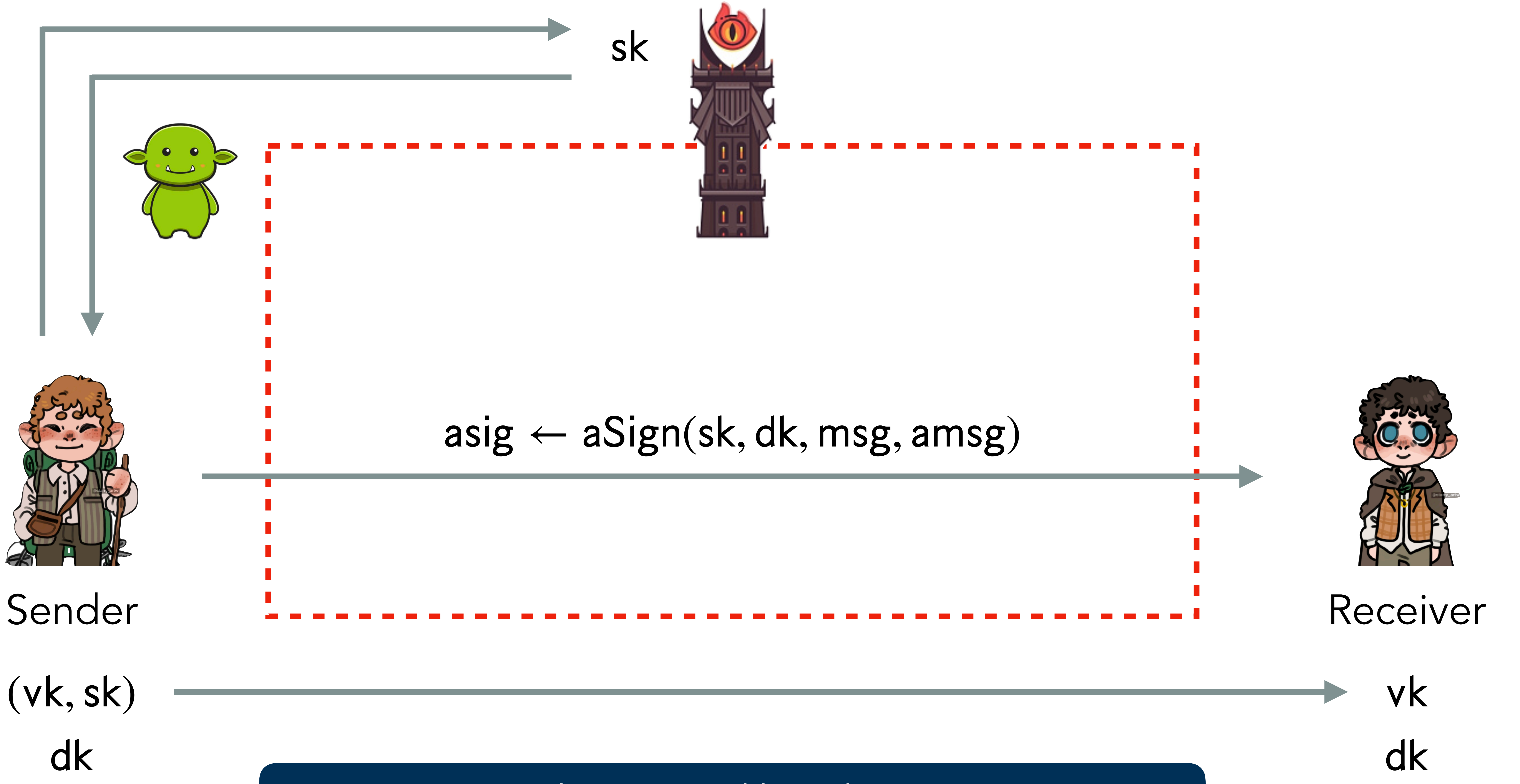
```
if $b = 0$ then
    $\mathsf{amsg} \leftarrow \perp$
if $b = 1$ then
    $\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$
    $\mathsf{amsg} \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{sig})$
```

$\leftarrow$ amsg

$b^* \leftarrow$

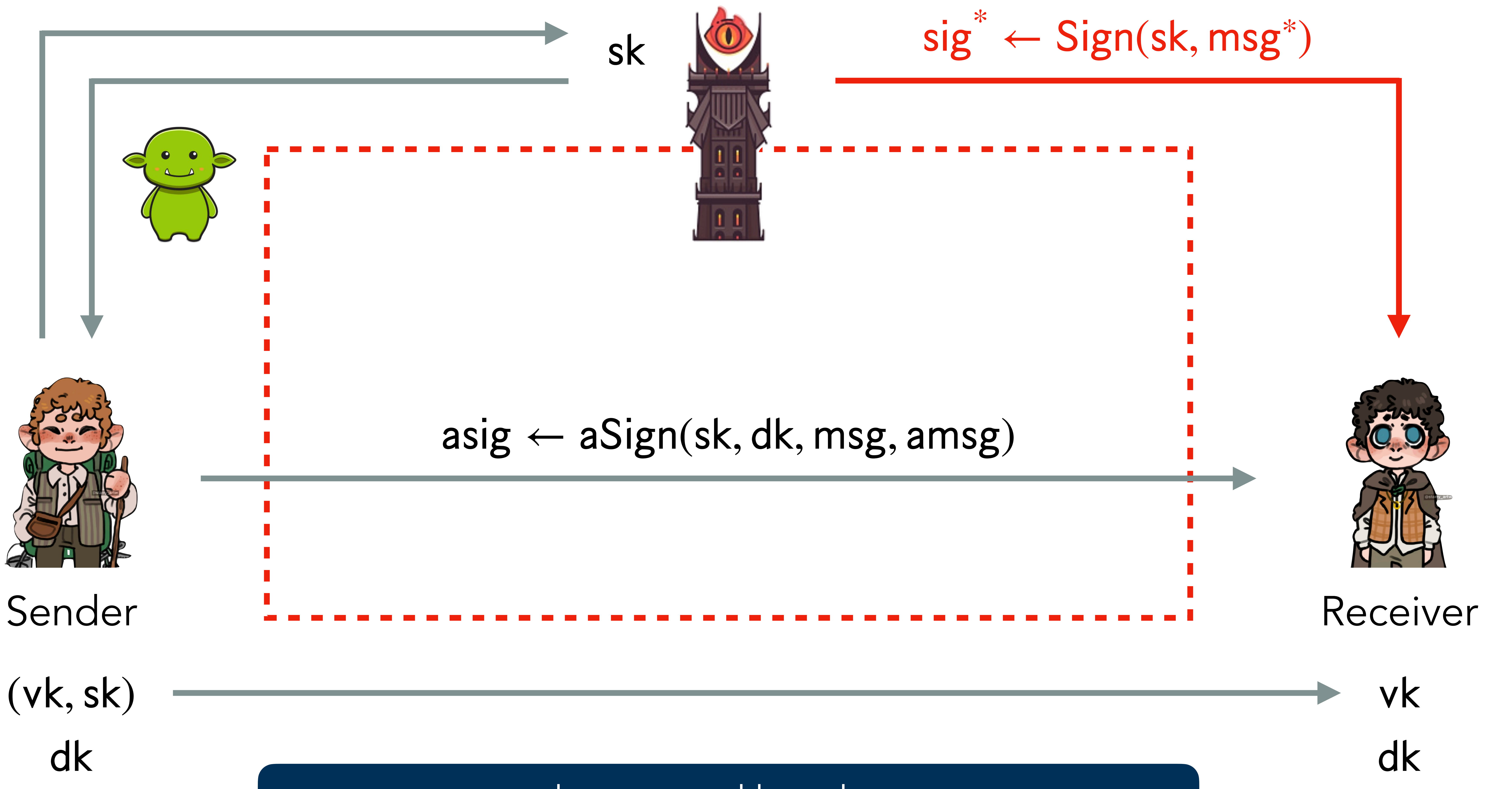
wins if $b = b^*$

Revisiting Robustness Threat Model



attacks captured by robustness

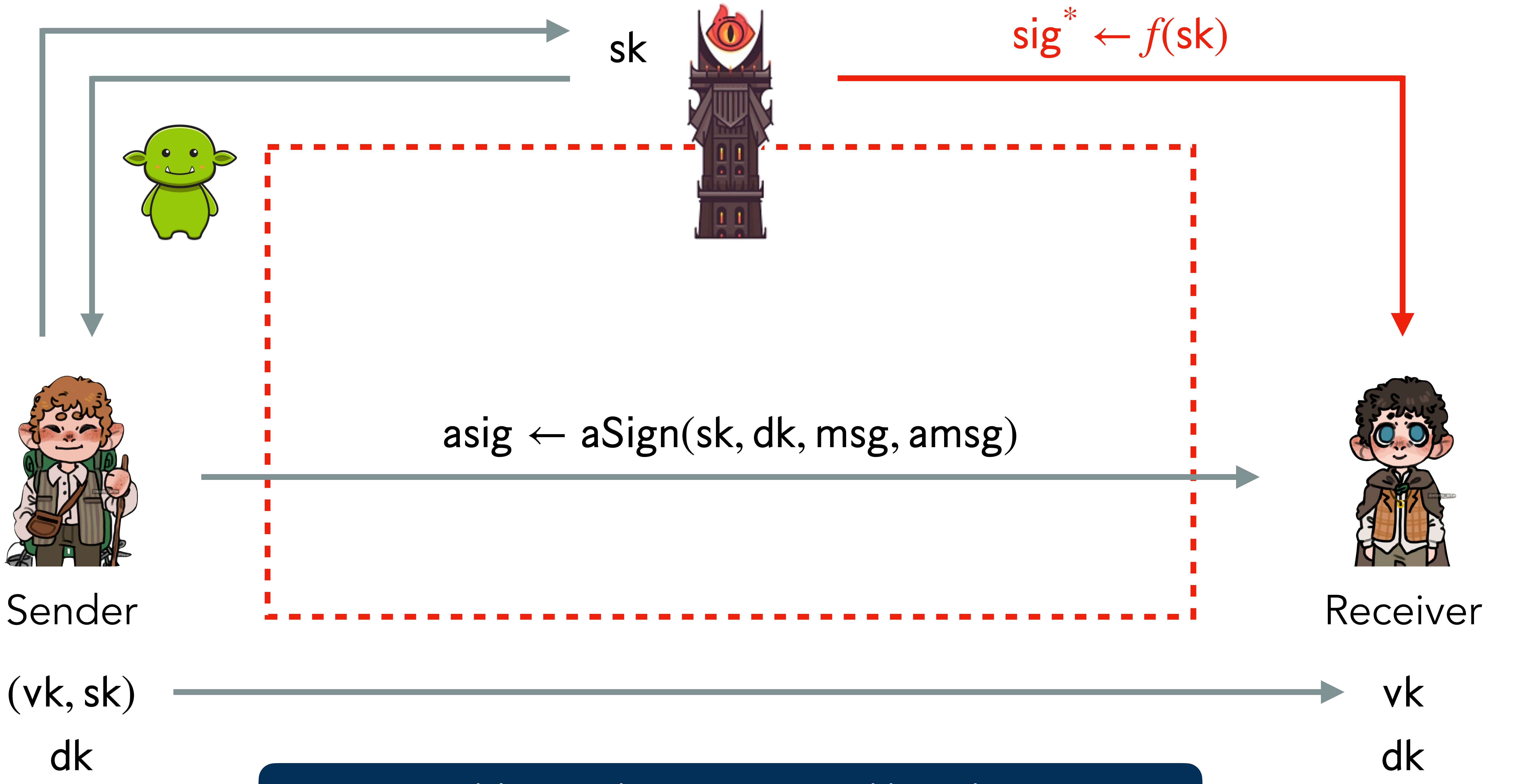
Revisiting Robustness Threat Model



sk
$\text{sig}^* \leftarrow \text{Sign}(\text{sk}, \text{msg}^*)$
$\text{asig} \leftarrow \text{aSign}(\text{sk}, \text{dk}, \text{msg}, \text{amsg})$
Sender
(vk, sk)
dk
Receiver
vk
dk

attacks captured by robustness

Revisiting Robustness Threat Model



sk

$\mathsf{sig}^* \leftarrow f(\mathsf{sk})$

$\mathsf{asig} \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$

Sender

Receiver

$(\mathsf{vk}, \mathsf{sk})$

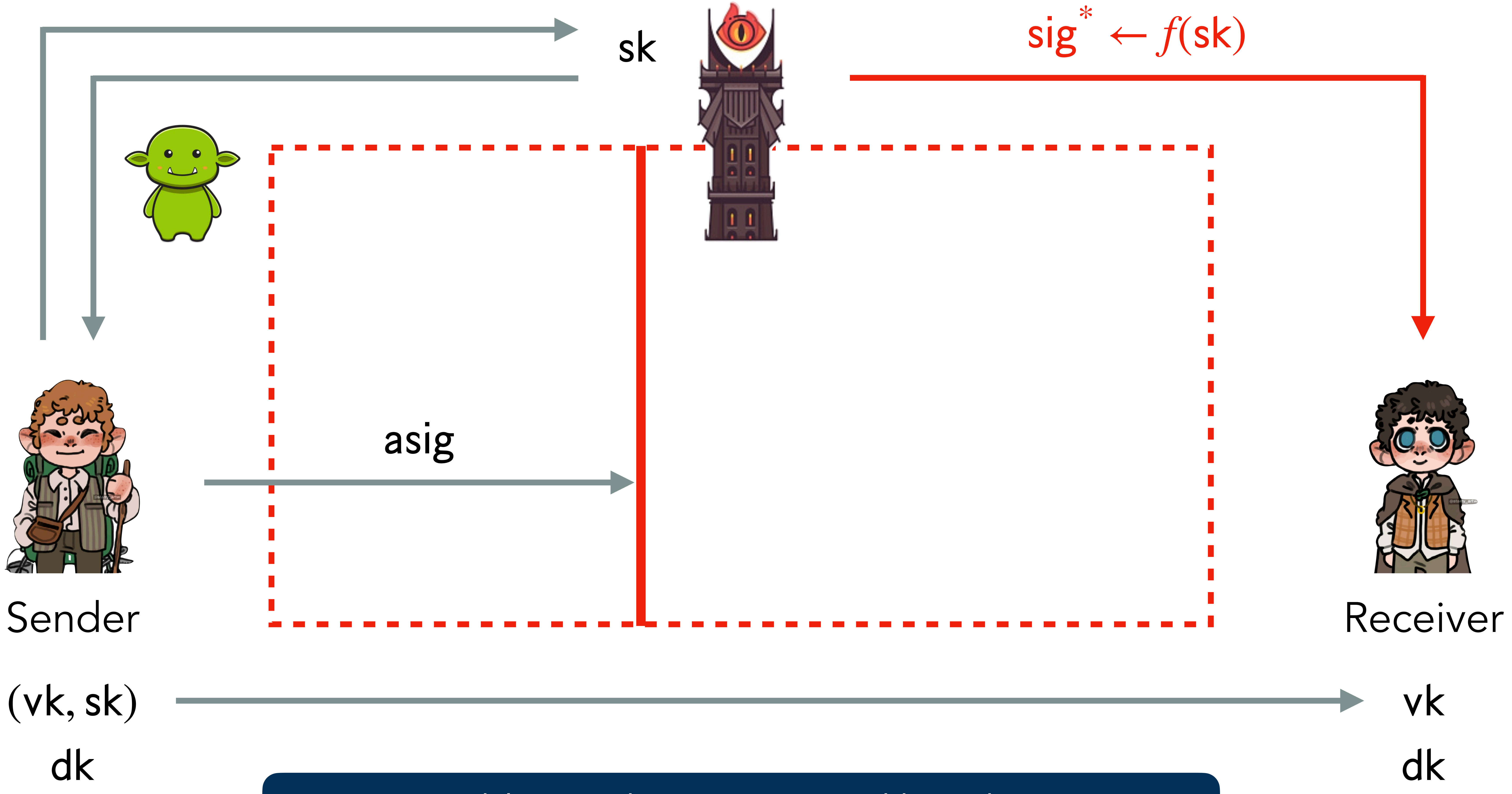
dk

vk

dk

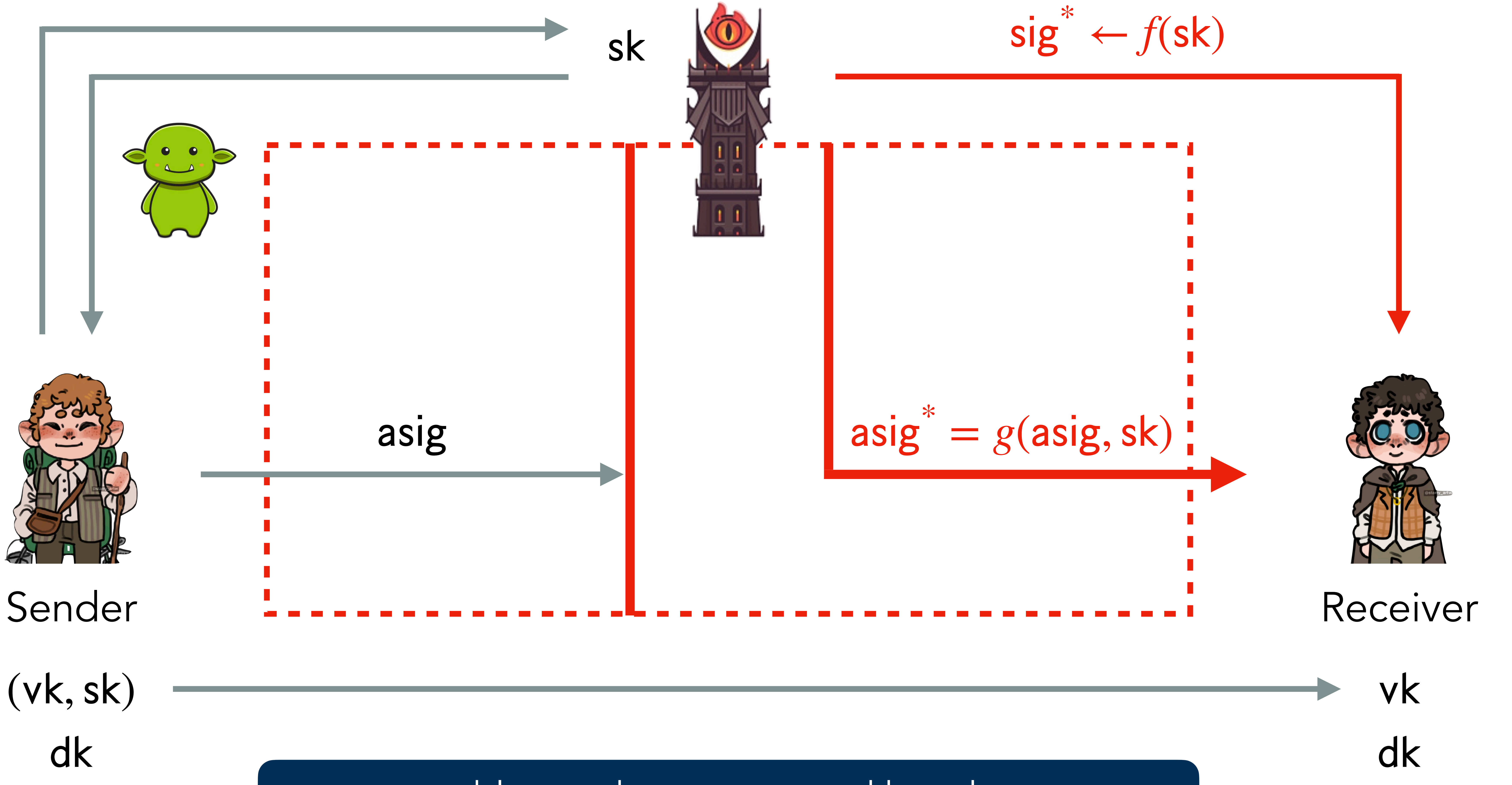
reasonable attacks ***not*** captured by robustness

Revisiting Robustness Threat Model



reasonable attacks ***not*** captured by robustness

Revisiting Robustness Threat Model



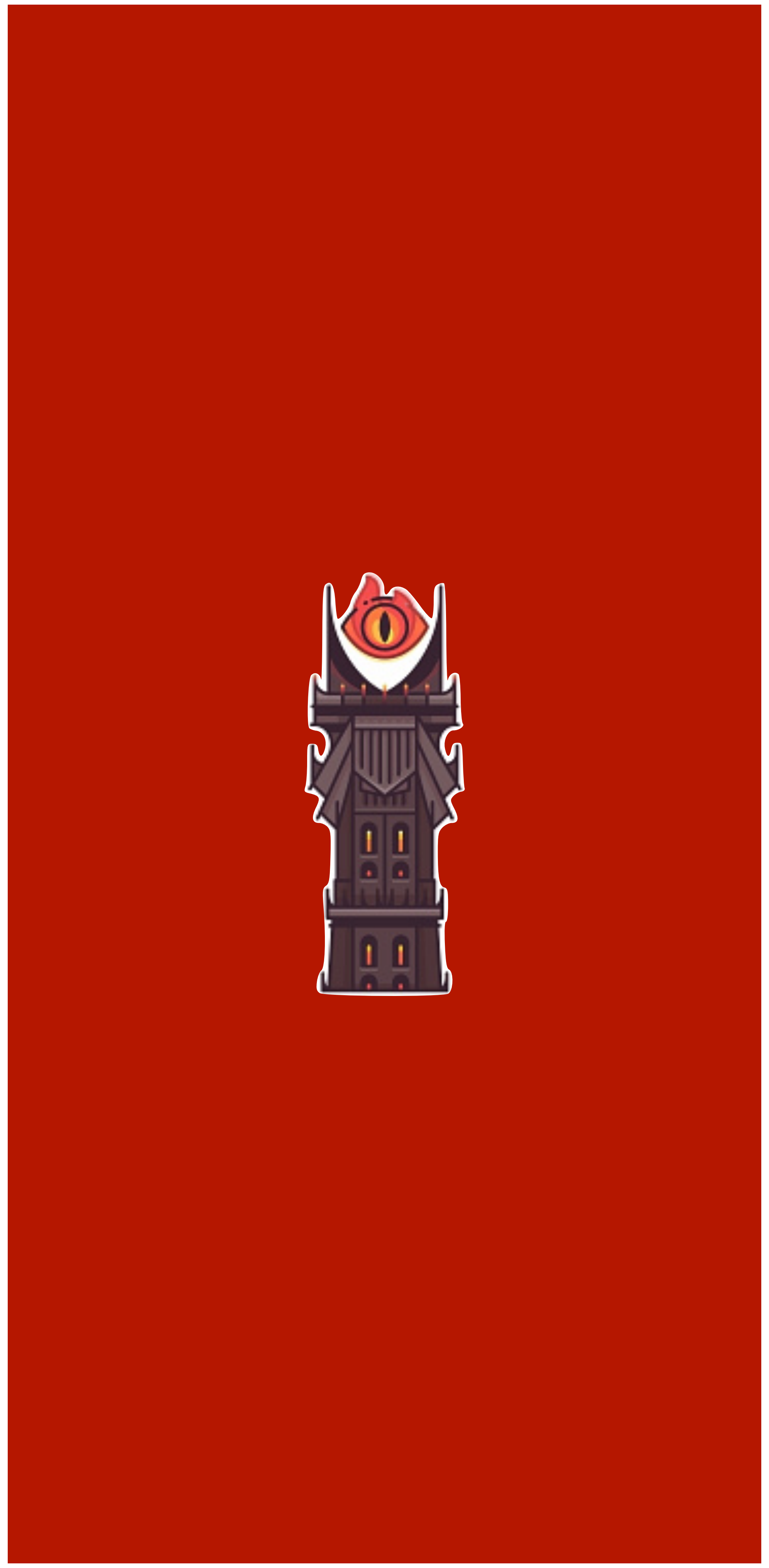
sk
$\text{sig}^* \leftarrow f(\text{sk})$
asig
$\text{asig}^* = g(\text{asig}, \text{sk})$
Sender
Receiver
(vk, sk)
dk
vk
dk

reasonable attacks ***not*** captured by robustness

Our Proposal: Dictator Unforgeability Game

$S \leftarrow \emptyset$

$\mathsf{aKeyGen}(1^{\lambda}) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

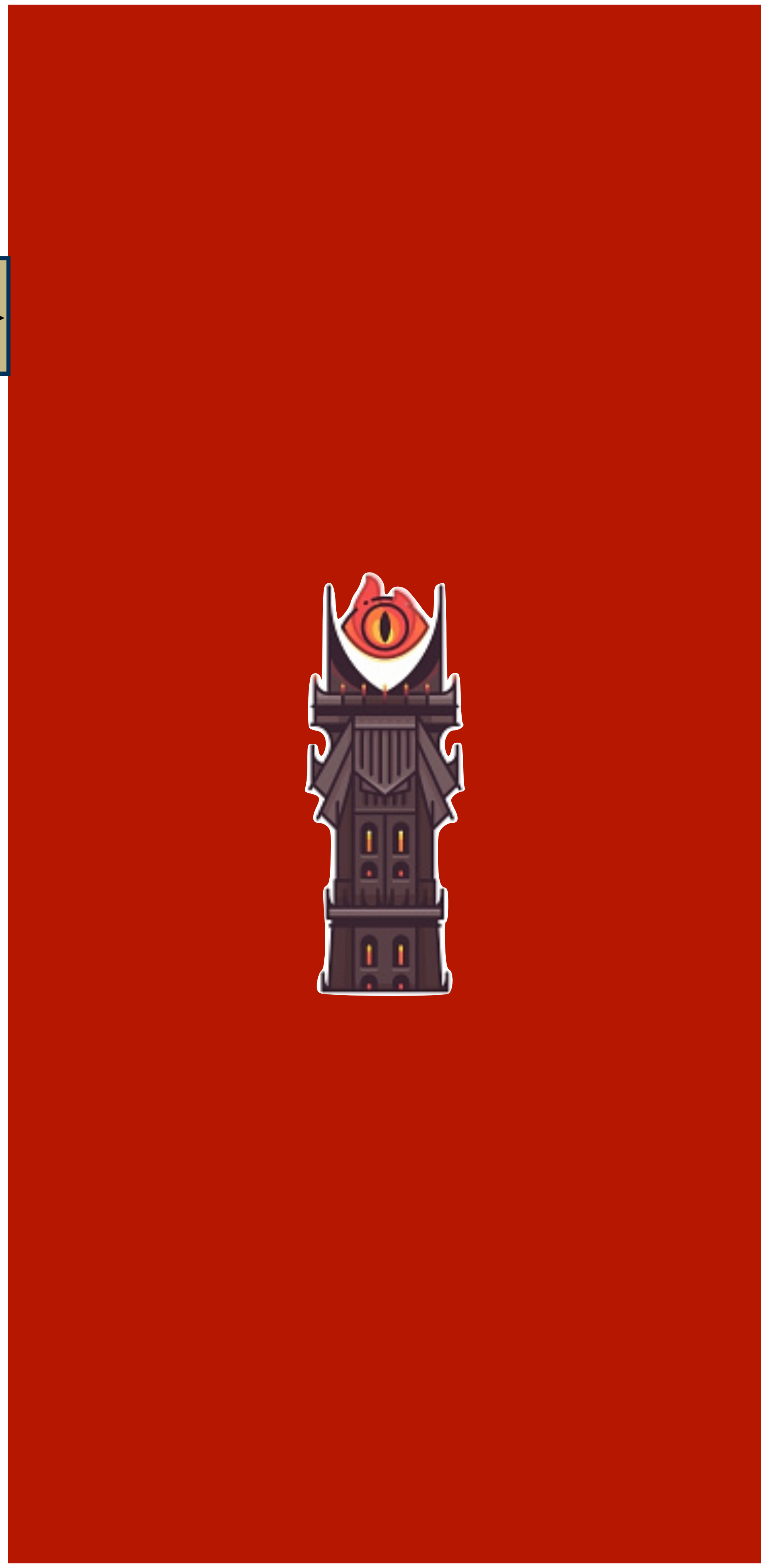


Our Proposal: Dictator Unforgeability Game

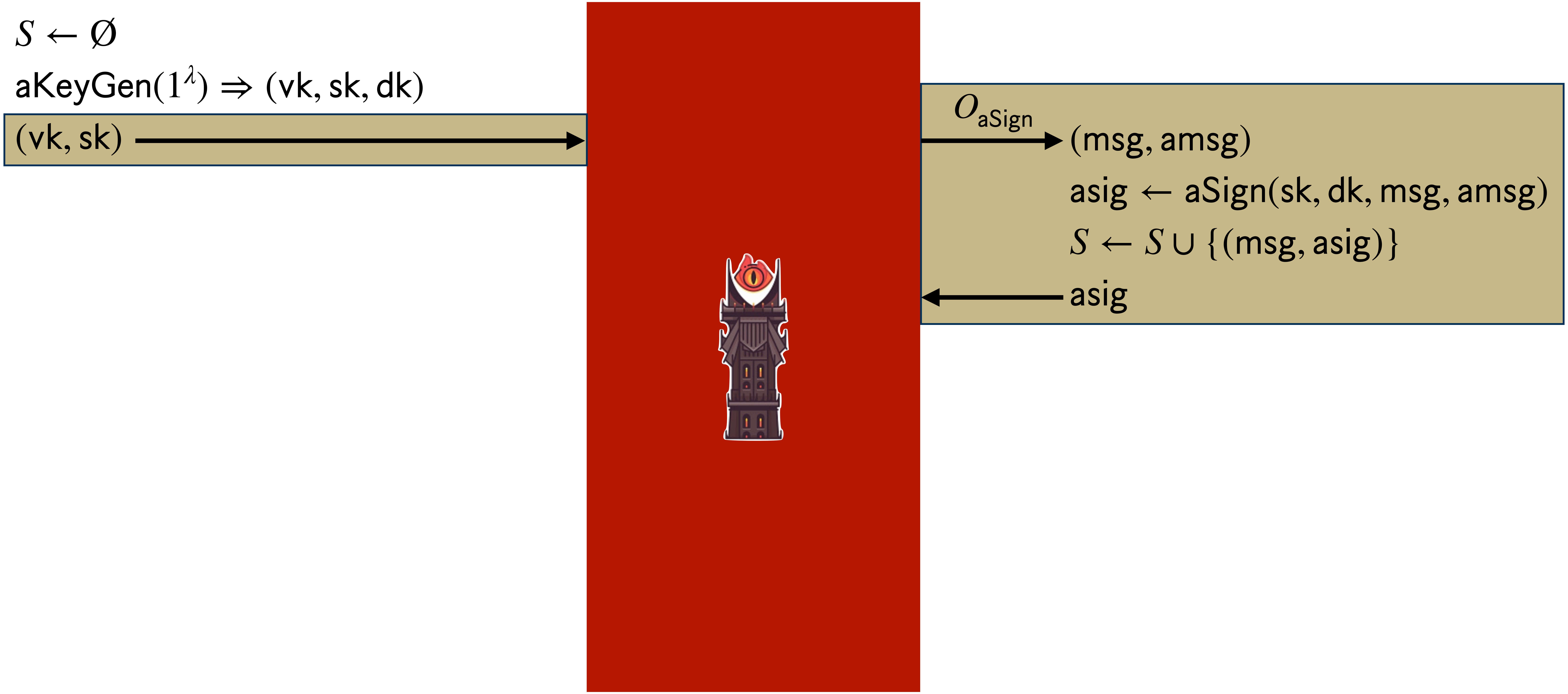
$S \leftarrow \emptyset$

aKeyGen$(1^\lambda) \Rightarrow$ (vk, sk, dk)

(vk, sk) $\longrightarrow$



Our Proposal: Dictator Unforgeability Game



$S \leftarrow \emptyset$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

$(\mathsf{vk}, \mathsf{sk}) \longrightarrow$

O_{aSign}

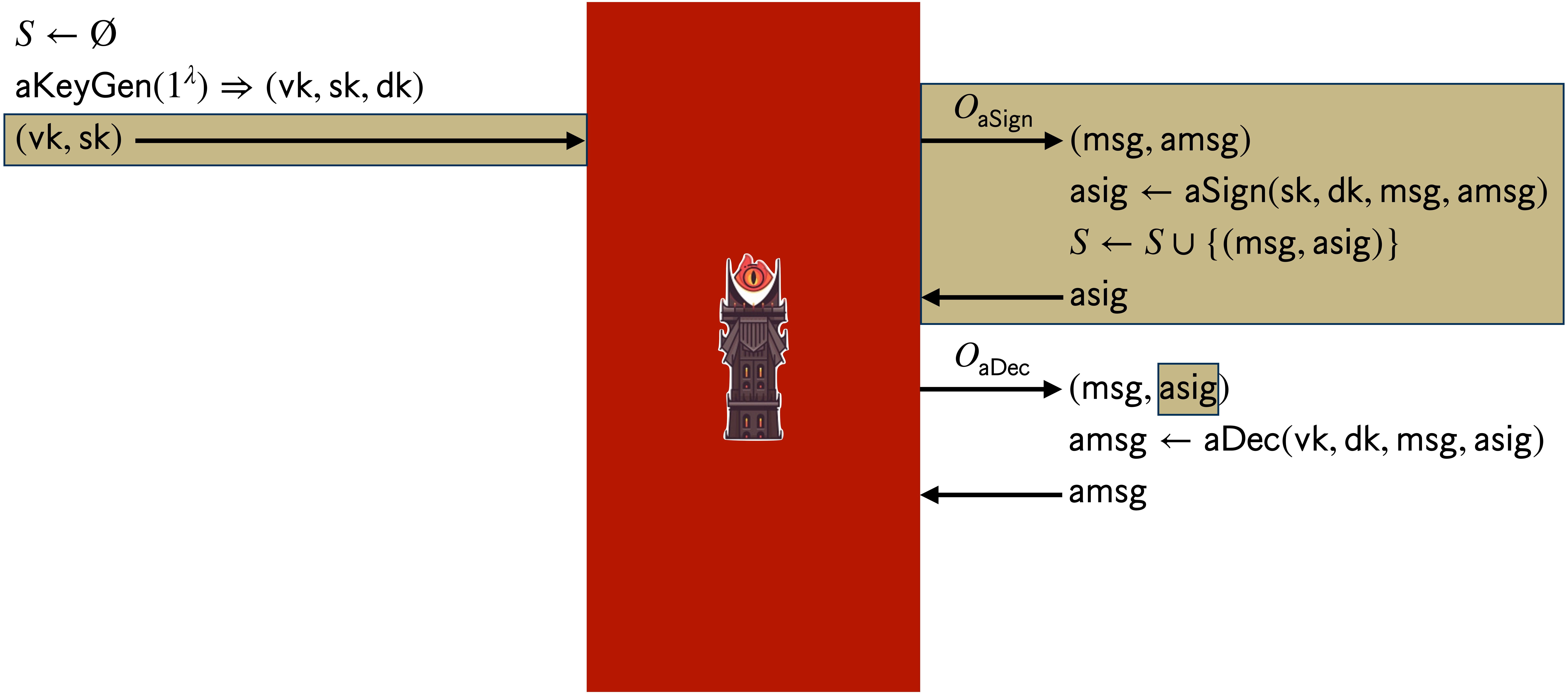
$\longrightarrow (\mathsf{msg}, \mathsf{amsg})$

$\mathsf{asig} \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$

$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{asig})\}$

$\longleftarrow \mathsf{asig}$

Our Proposal: Dictator Unforgeability Game



$S \leftarrow \emptyset$

aKeyGen(1^λ) ⇒ (vk, sk, dk)

(vk, sk)

O_{aSign}

(msg, amsg)

asig ← aSign(sk, dk, msg, amsg)

$S \leftarrow S \cup \{(\text{msg}, \text{asig})\}$

asig

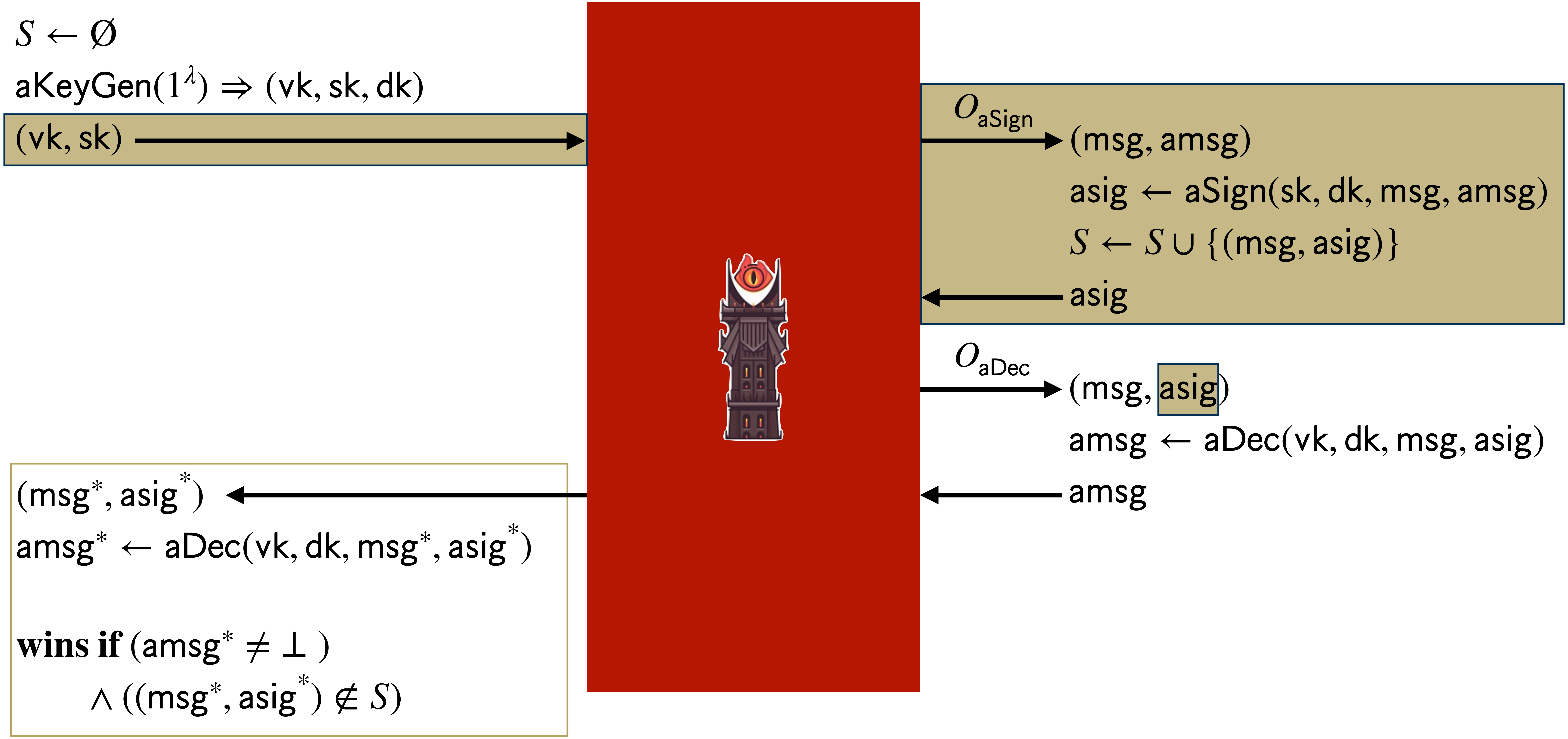
O_{aDec}

(msg, asig)

amsg ← aDec(vk, dk, msg, asig)

amsg

Our Proposal: Dictator Unforgeability Game



$S \leftarrow \emptyset$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

$(\mathsf{vk}, \mathsf{sk}) \longrightarrow$

O_{aSign}

$\longrightarrow (\mathsf{msg}, \mathsf{amsg})$

$\mathsf{asig} \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$

$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{asig})\}$

$\longleftarrow \mathsf{asig}$

O_{aDec}

$\longrightarrow (\mathsf{msg}, \boxed{\mathsf{asig}})$

$\mathsf{amsg} \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig})$

$\longleftarrow \mathsf{amsg}$

$(\mathsf{msg}^*, \mathsf{asig}^*) \longleftarrow$

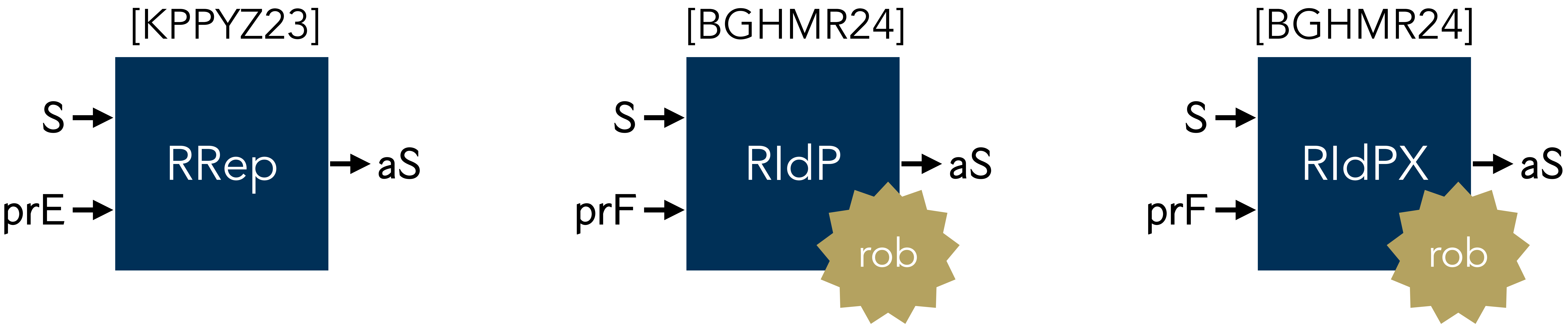
$\mathsf{amsg}^* \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}^*, \mathsf{asig}^*)$

wins if $(\mathsf{amsg}^* \neq \perp)$

$\wedge\ ((\mathsf{msg}^*, \mathsf{asig}^*) \notin S)$

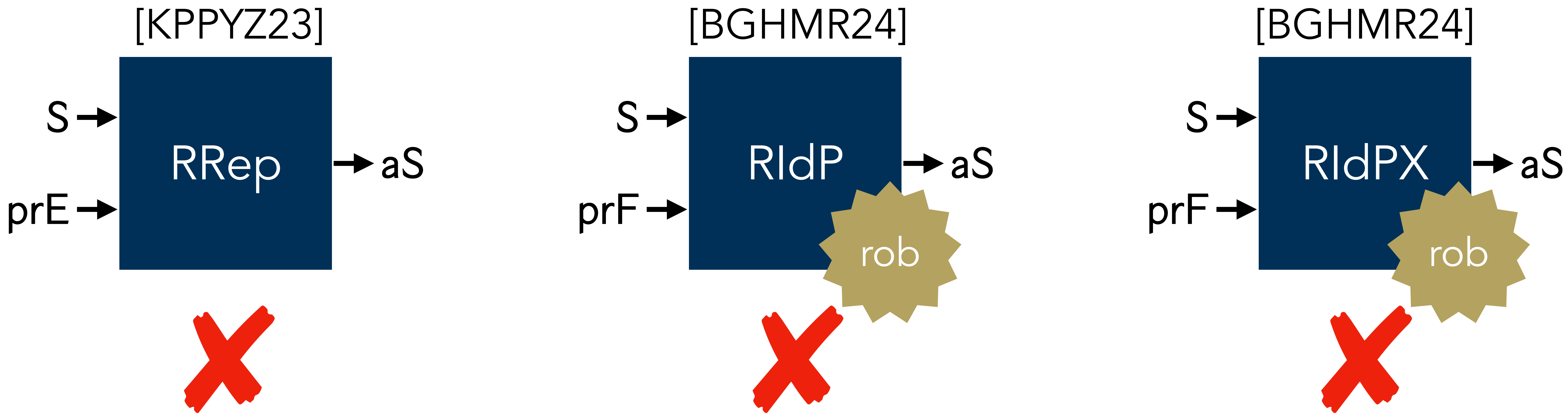
Dictator Unforgeability of Transforms

- Are the previously proposed transforms dictator unforgeable?



Dictator Unforgeability of Transforms

- Are the previously proposed transforms dictator unforgeable?



[KPPYZ23]

S → RRep → aS

prE → RRep

✗

[BGHMR24]

S → RIdP → aS

prF → RIdP

rob

✗

[BGHMR24]

S → RIdPX → aS

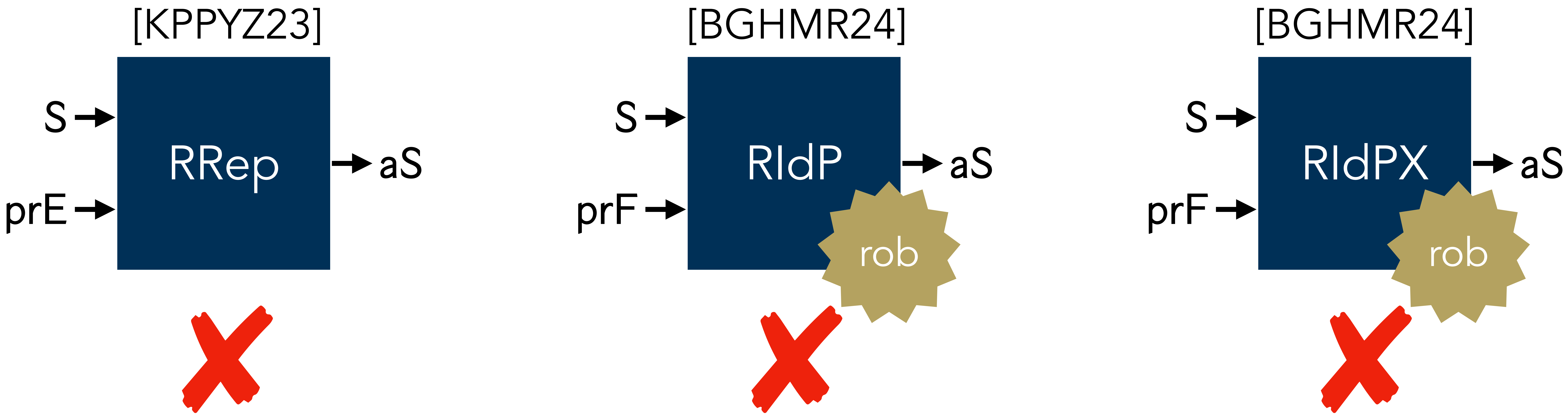
prF → RIdPX

rob

✗

Dictator Unforgeability of Transforms

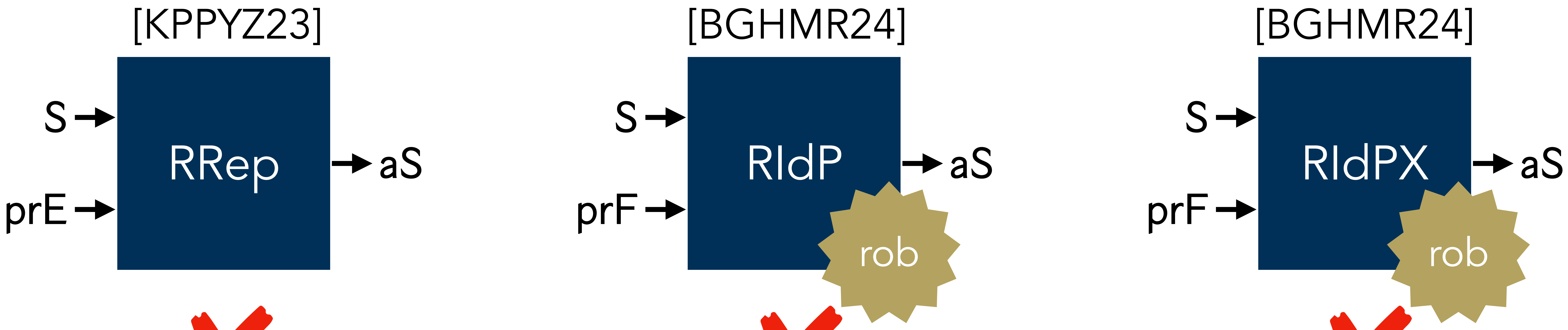
- Are the previously proposed transforms dictator unforgeable?



- Can we patch any of the transforms?

Dictator Unforgeability of Transforms

- Are the previously proposed transforms dictator unforgeable?



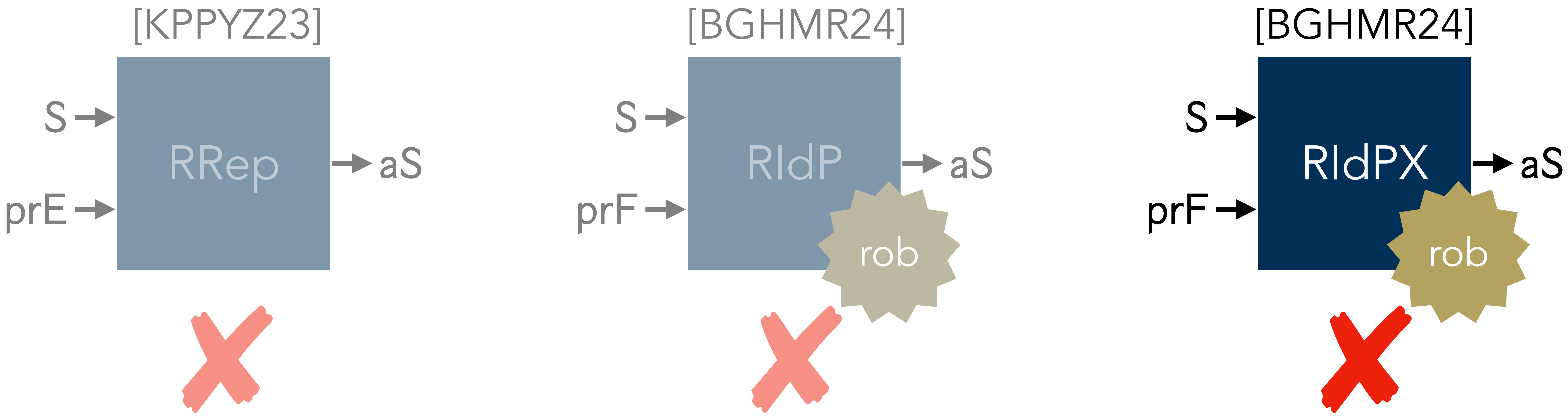
✗ ✗ ✗

- Can we patch any of the transforms?

✓ ✓ ✗

Dictator Unforgeability of Transforms

- Are the previously proposed transforms dictator unforgeable?



- Can we patch any of the transforms?

✓ ✓ ✗

Dictator Attacking RIdPX

Sender

Dictator

Receiver

Dictator Attacking RIdPX

Sender

Dictator

Receiver

amsg = "meet at 2PM"

Dictator Attacking RIdPX

Sender

Dictator

Receiver

$\mathsf{amsg} = \text{"meet at 2PM"}$

$r \leftarrow \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}) \oplus \mathsf{amsg}$

Dictator Attacking RIdPX

Sender

Dictator

Receiver

$\mathsf{amsg} = \text{"meet at 2PM"}$

$r \leftarrow \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}) \oplus \mathsf{amsg}$

$\mathsf{asig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r)$

scheme is randomness recoverable e.g. ElGamal, Schnorr, RSA-PSS, .etc

Dictator Attacking RIdPX

Sender

Dictator

Receiver

$\text{amsg} = \text{"meet at 2PM"}$

$r \leftarrow \text{prF}(\text{dk}, \text{ctr}) \oplus \text{amsg}$

$\text{asig} \leftarrow \text{Sign}(\text{sk}, \text{msg}; r) \xrightarrow{\quad\text{asig}\quad}$

scheme is randomness recoverable e.g. ElGamal, Schnorr, RSA-PSS, .etc

Dictator Attacking RIdPX

Sender

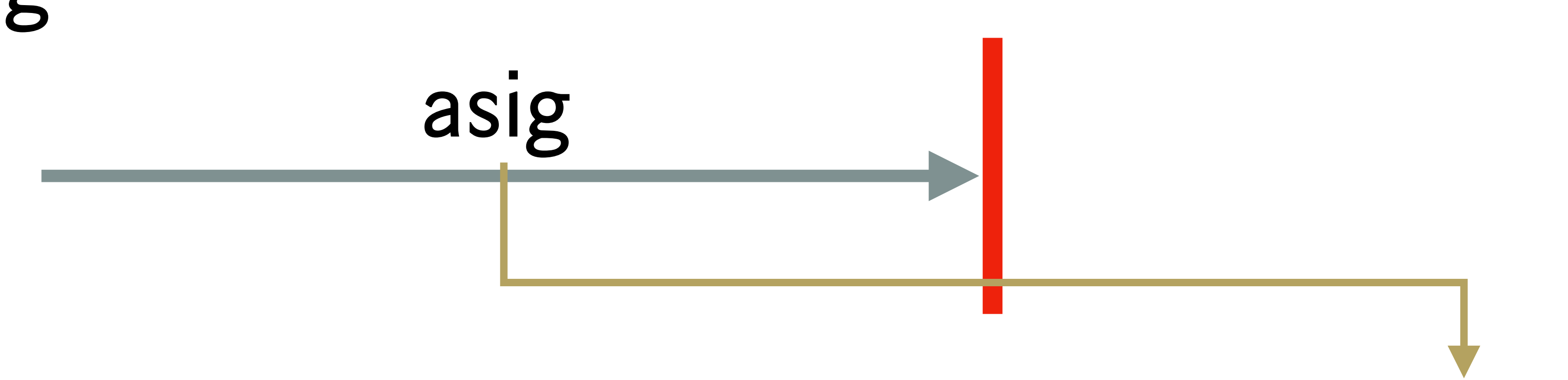
Dictator

Receiver

amsg = "meet at 2PM"

$r \leftarrow \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}) \oplus \mathsf{amsg}$

$\mathsf{asig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r)$



$r \leftarrow \mathsf{RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig})$

scheme is randomness recoverable e.g. ElGamal, Schnorr, RSA-PSS, .etc

Dictator Attacking RIdPX

Sender

Dictator

Receiver

$\mathsf{amsg} = \text{"meet at 2PM"}$

$r \leftarrow \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}) \oplus \mathsf{amsg}$

$\mathsf{asig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r)$ $\xrightarrow{\mathsf{asig}}$

scheme is randomness recoverable e.g. ElGamal, Schnorr, RSA-PSS, .etc

$r \leftarrow \mathsf{RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig})$

$r^* \leftarrow r \oplus \mathsf{amsg}'$

Dictator Attacking RIdPX

Sender

Dictator

Receiver

$\mathsf{amsg} = $ "meet at 2PM"

$r \leftarrow \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}) \oplus \mathsf{amsg}$

$\mathsf{asig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r) \xrightarrow{\mathsf{asig}} |$

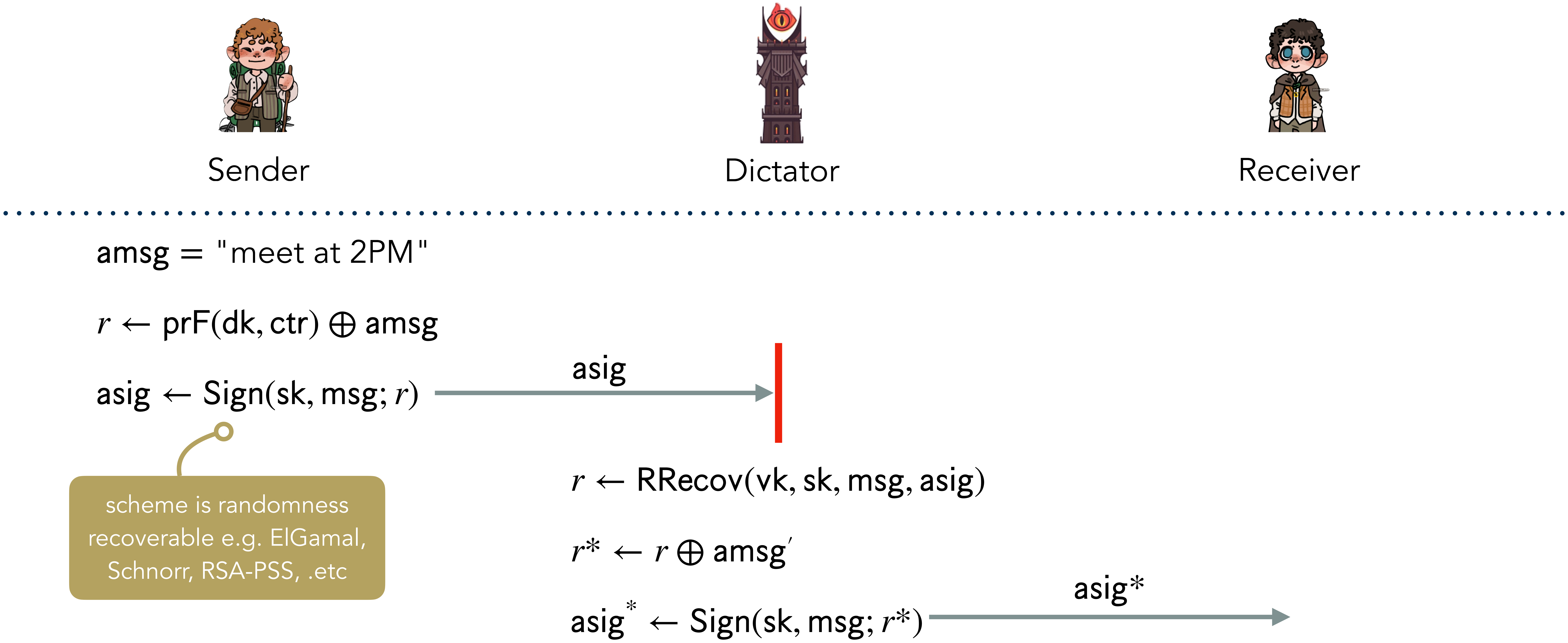
scheme is randomness recoverable e.g. ElGamal, Schnorr, RSA-PSS, .etc

$r \leftarrow \mathsf{RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig})$

$r^* \leftarrow r \oplus \mathsf{amsg}'$

$\mathsf{asig}^* \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r^*)$

Dictator Attacking RIdPX



Sender

Dictator

Receiver

$\mathsf{amsg} = $ "meet at 2PM"

$r \leftarrow \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}) \oplus \mathsf{amsg}$

$\mathsf{asig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r)$

asig

scheme is randomness recoverable e.g. ElGamal, Schnorr, RSA-PSS, .etc

$r \leftarrow \mathsf{RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig})$

$r^* \leftarrow r \oplus \mathsf{amsg}'$

$\mathsf{asig}^* \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r^*)$

asig*

Dictator Attacking RIdPX

Sender

Dictator

Receiver

$\mathsf{amsg} = \text{"meet at 2PM"}$

$r \leftarrow \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}) \oplus \mathsf{amsg}$

$\mathsf{asig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r) \xrightarrow{\mathsf{asig}}$

scheme is randomness recoverable e.g. ElGamal, Schnorr, RSA-PSS, .etc

$r \leftarrow \mathsf{RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig})$

$r^* \leftarrow r \oplus \mathsf{amsg}'$

$\mathsf{asig}^* \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r^*) \xrightarrow{\mathsf{asig}^*}$

$\mathsf{amsg}^* \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig}^*)$

Dictator Attacking RIdPX

Sender

Dictator

Receiver

$\mathsf{amsg} = $ "meet at 2PM"

$r \leftarrow \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}) \oplus \mathsf{amsg}$

$\mathsf{asig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r) \xrightarrow{\mathsf{asig}} |$

scheme is randomness recoverable e.g. ElGamal, Schnorr, RSA-PSS, .etc

$r \leftarrow \mathsf{RRecov}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg}, \mathsf{asig})$

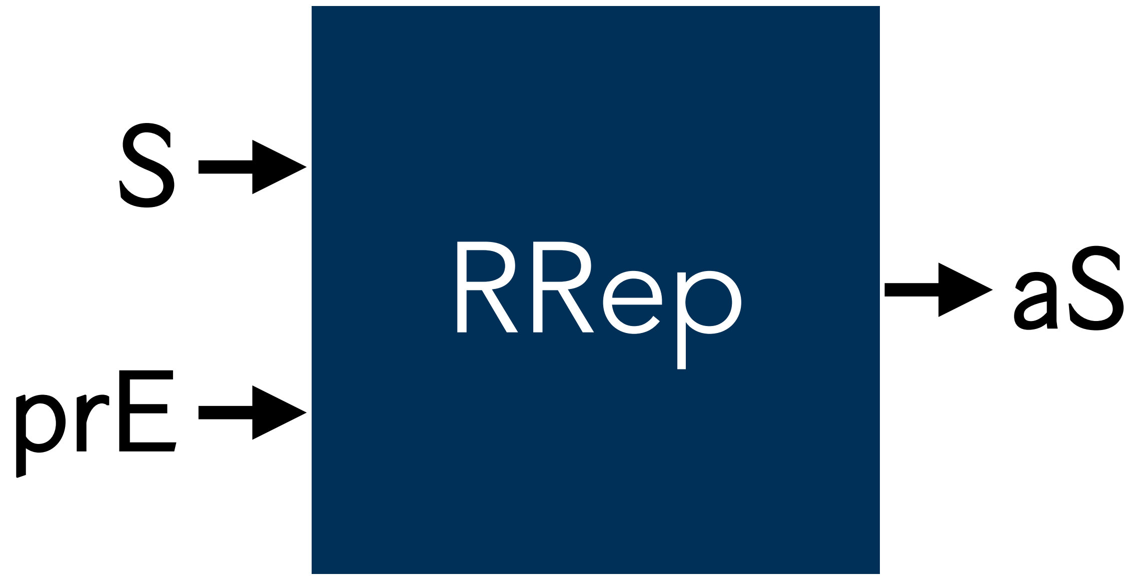
$r^* \leftarrow r \oplus \mathsf{amsg}'$

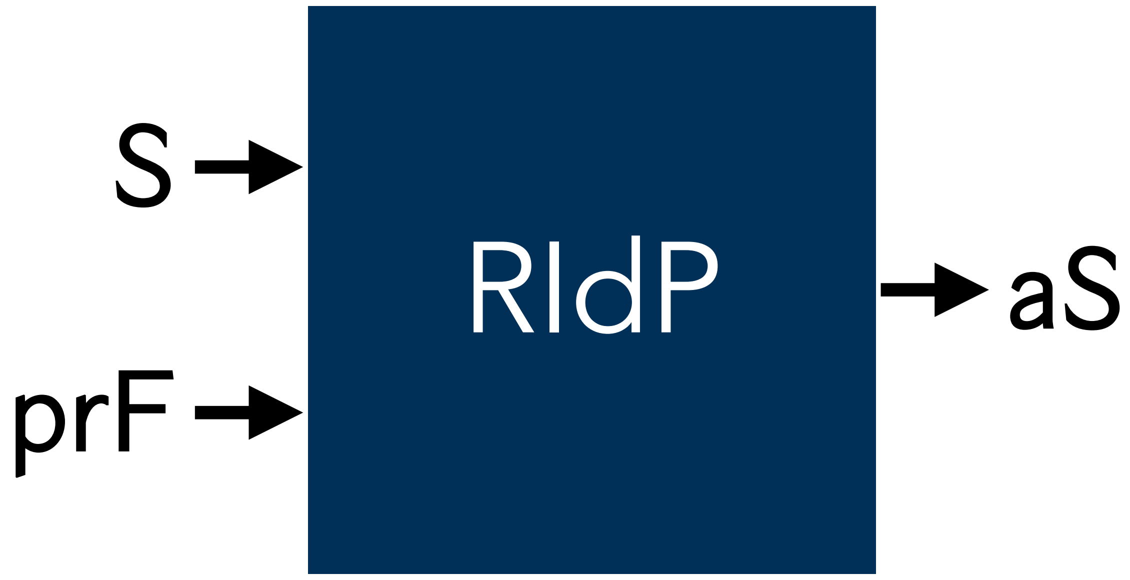
$\mathsf{asig}^* \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r^*) \xrightarrow{\mathsf{asig}^*}$

$\mathsf{amsg}^* \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}, \mathsf{asig}^*)$

$\mathsf{amsg}^* = \mathsf{amsg} \oplus \mathsf{amsg}' = $ "meet at 4PM"

Repairing Dictator Unforgeability of RRep and RIdP





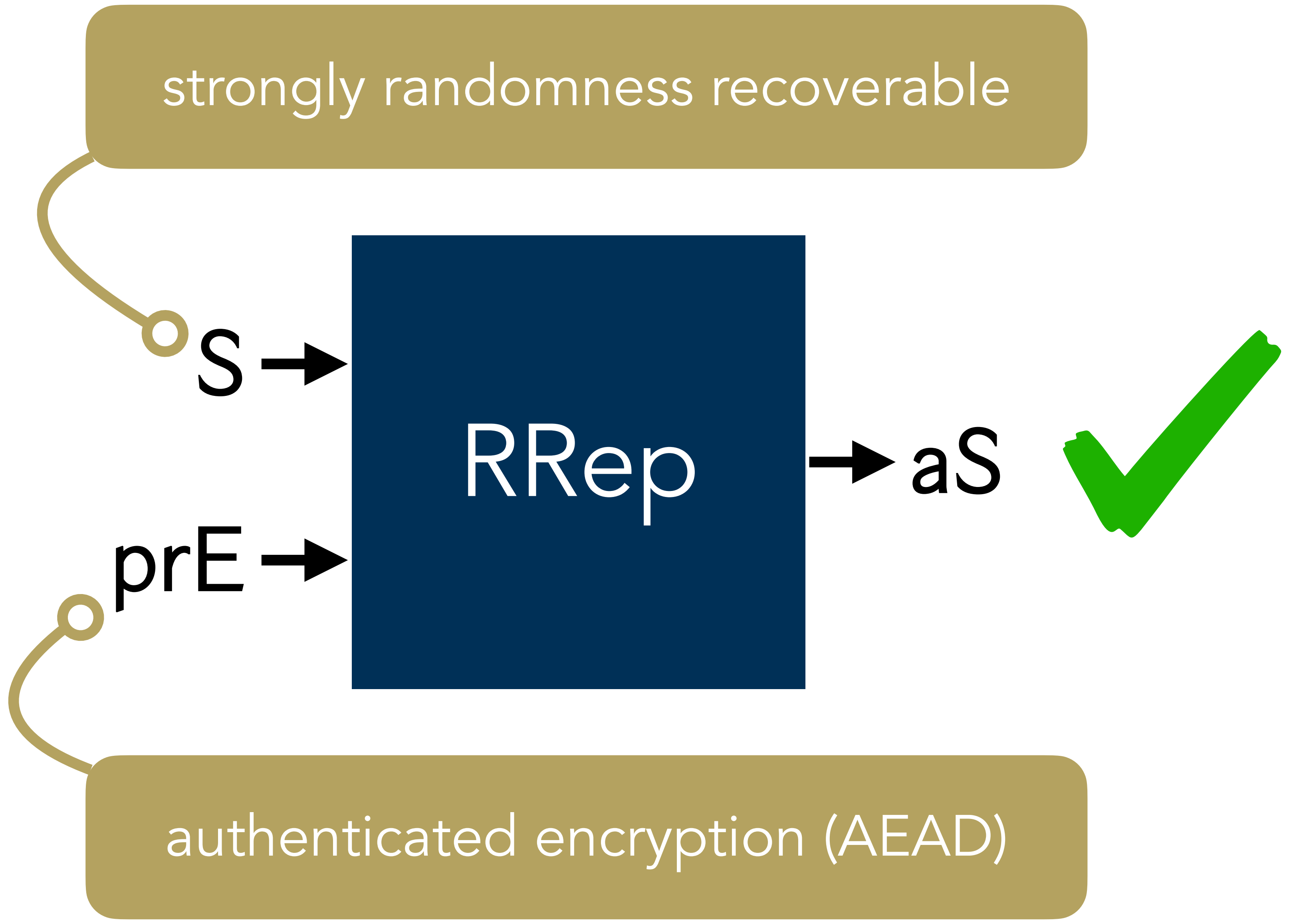
Replaces signing randomness with pseudorandom encryptions i.e.

$r \leftarrow \mathsf{prE}.\mathsf{Enc}(\mathsf{dk}, \mathsf{amsg})$

Replaces signing randomness with pseudorandom function outputs i.e.

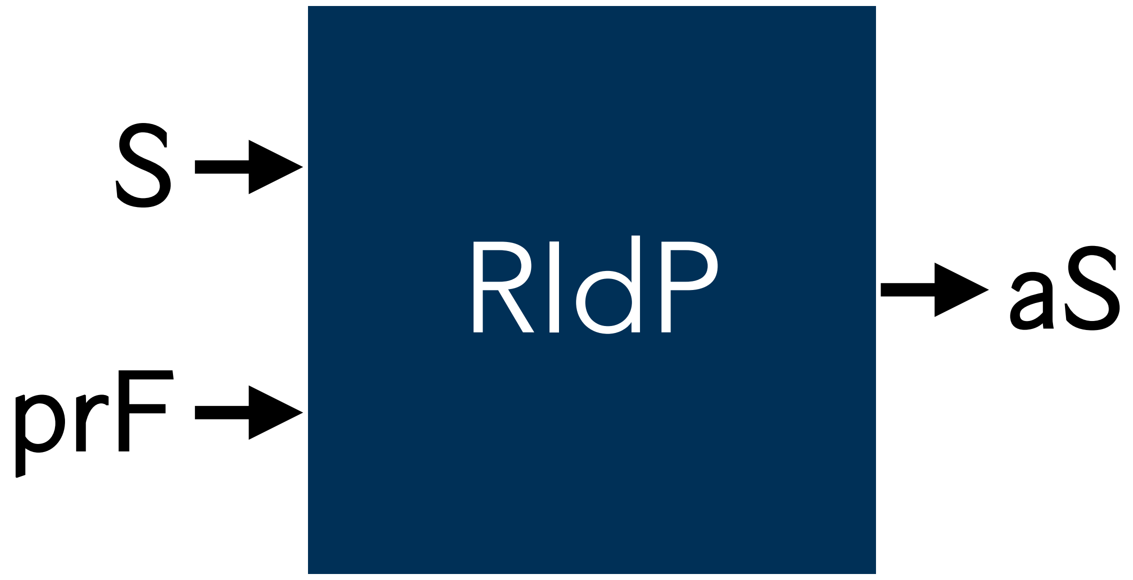
$r \leftarrow \mathsf{prF}(\mathsf{dk}, (\mathsf{ctr}, \mathsf{amsg}))$

Repairing Dictator Unforgeability of RRep and RIdP


strongly randomness recoverable
S
prE
RRep
aS
✓
authenticated encryption (AEAD)

Replaces signing randomness with pseudorandom encryptions i.e.

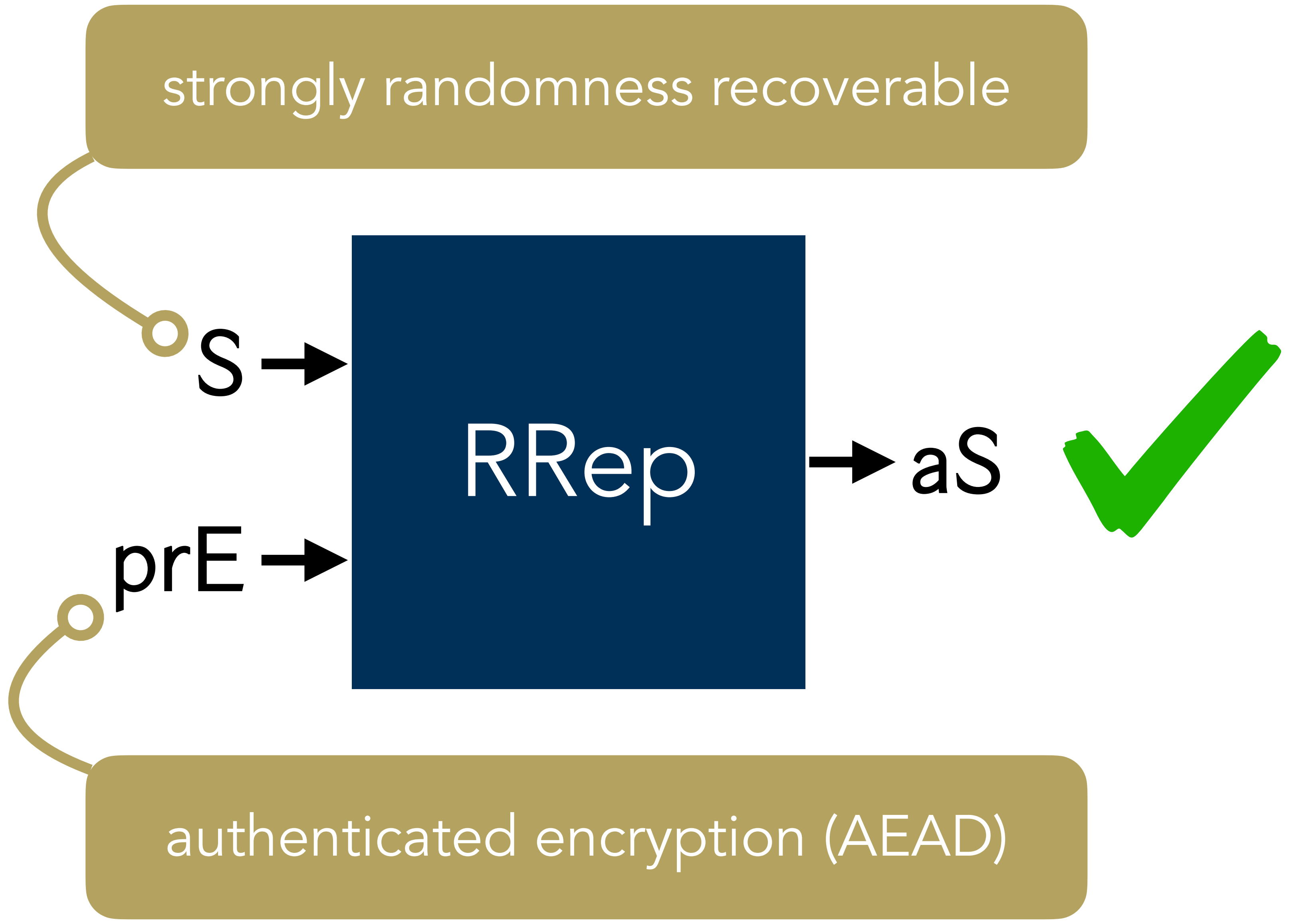
$r \leftarrow \text{prE}\,.\,\text{Enc}(\text{dk}, \text{amsg})$



Replaces signing randomness with pseudorandom function outputs i.e.

$r \leftarrow \text{prF}(\text{dk}, (\text{ctr}, \text{amsg}))$

Repairing Dictator Unforgeability of RRep and RIdP



strongly randomness recoverable

S

prE

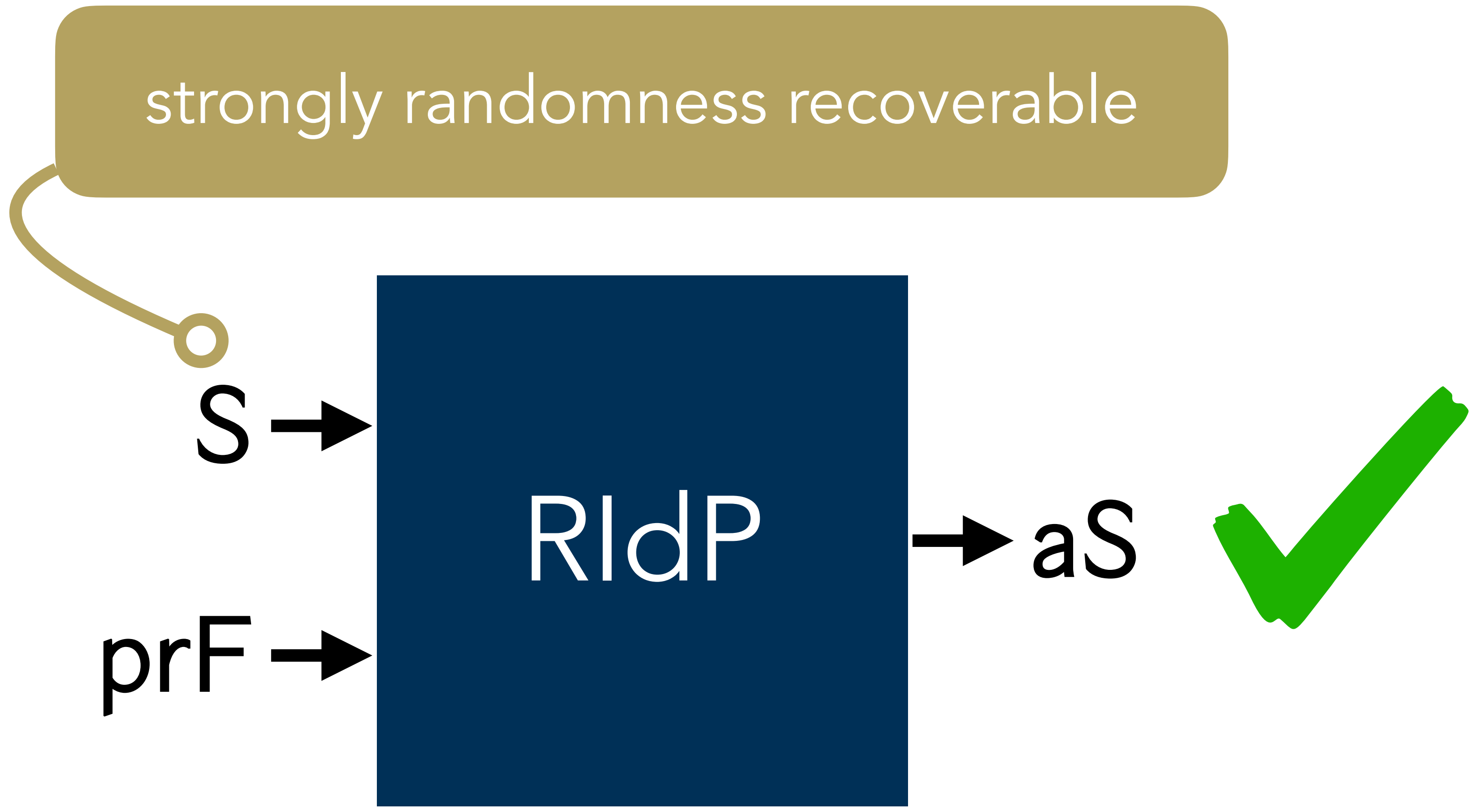
RRep

aS ✓

authenticated encryption (AEAD)

Replaces signing randomness with pseudorandom encryptions i.e.

$r \leftarrow \mathrm{prE.Enc}(\mathrm{dk}, \mathrm{amsg})$



strongly randomness recoverable

S

prF

RIdP

aS ✓

Replaces signing randomness with pseudorandom function outputs i.e.

$r \leftarrow \mathrm{prF}(\mathrm{dk}, (\mathrm{ctr}, \mathrm{msg}, \mathrm{amsg}))$

Part 2: Strengthening Private Anamorphism to Recipient Unforgeability

Robustness	Private Anamorphism
1 Observe a gap between a stated goal of robustness and its formalization.	Observe a gap between the deployment scenario of private anamorphism and its formalization. 5
2 Propose **Dictator Unforgeability.**	Propose **Recipient Unforgeability.** 6
3 Mount a practical attack a previously proposed *robust* anamorphic signature scheme.	Mount a practical attack a natural *private* anamorphic signature scheme. 7
4 Repair other prior anamorphic transforms to achieve dictator unforgeability.	Repair (in two ways) a prior anamorphic transform to achieve recipient unforgeability. 8

Private Anamorphism [KPPYZ23]

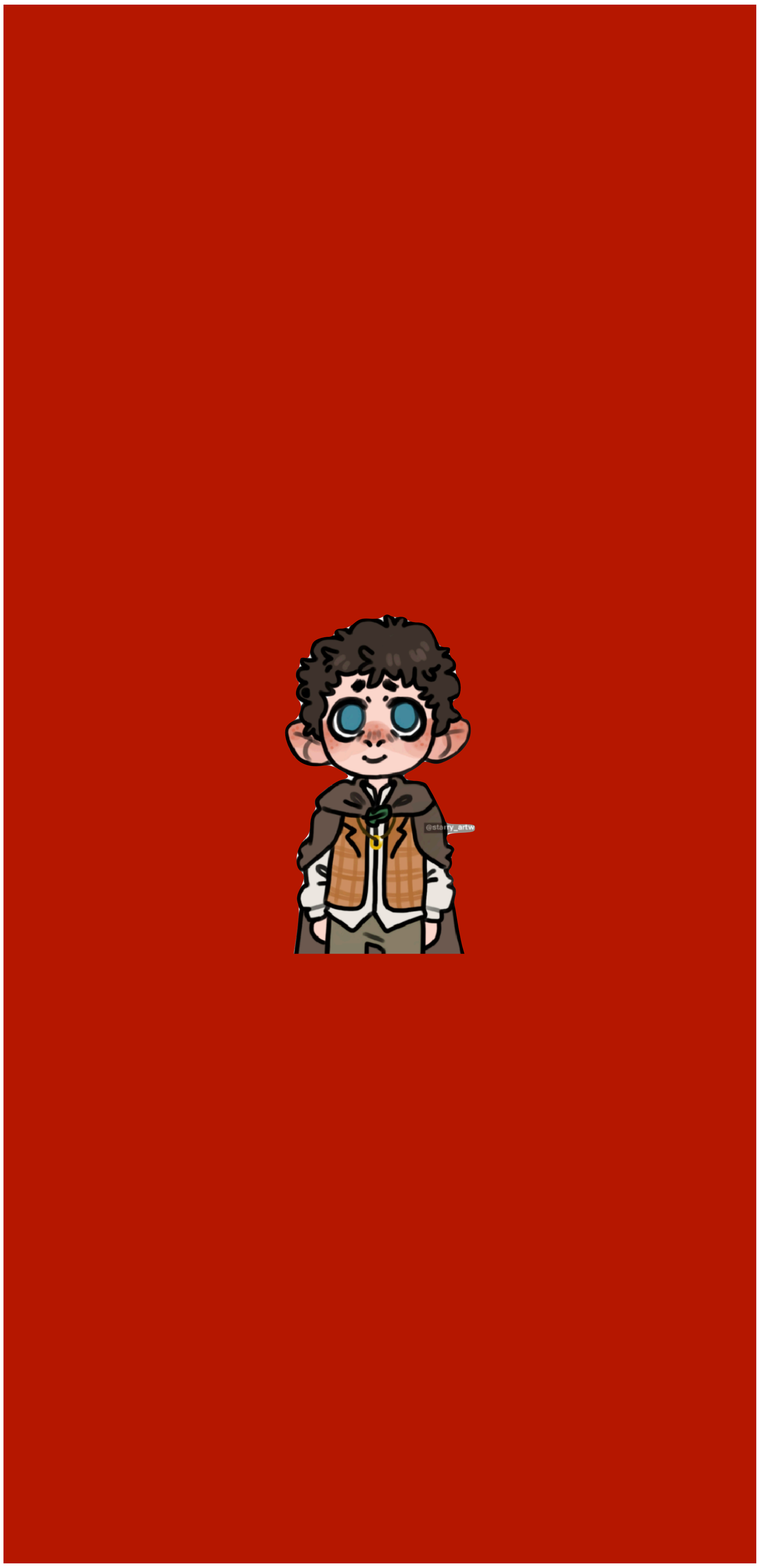
- Proposed alongside anamorphic signatures.
- **High level goal:** a recipient who knows the double key dk and sees honest signatures cannot forge new signatures.
- KPPYZ discusses a primary motivation for private anamorphism.
    - **Security:** (roughly) to prevent a recipient from forging signatures on behalf of the sender.
- KPPYZ provide a framework that achieves private anamorphism which covers the randomness replacement transform RRep as a special case.

Private Anamorphism Game [KPPYZ23]

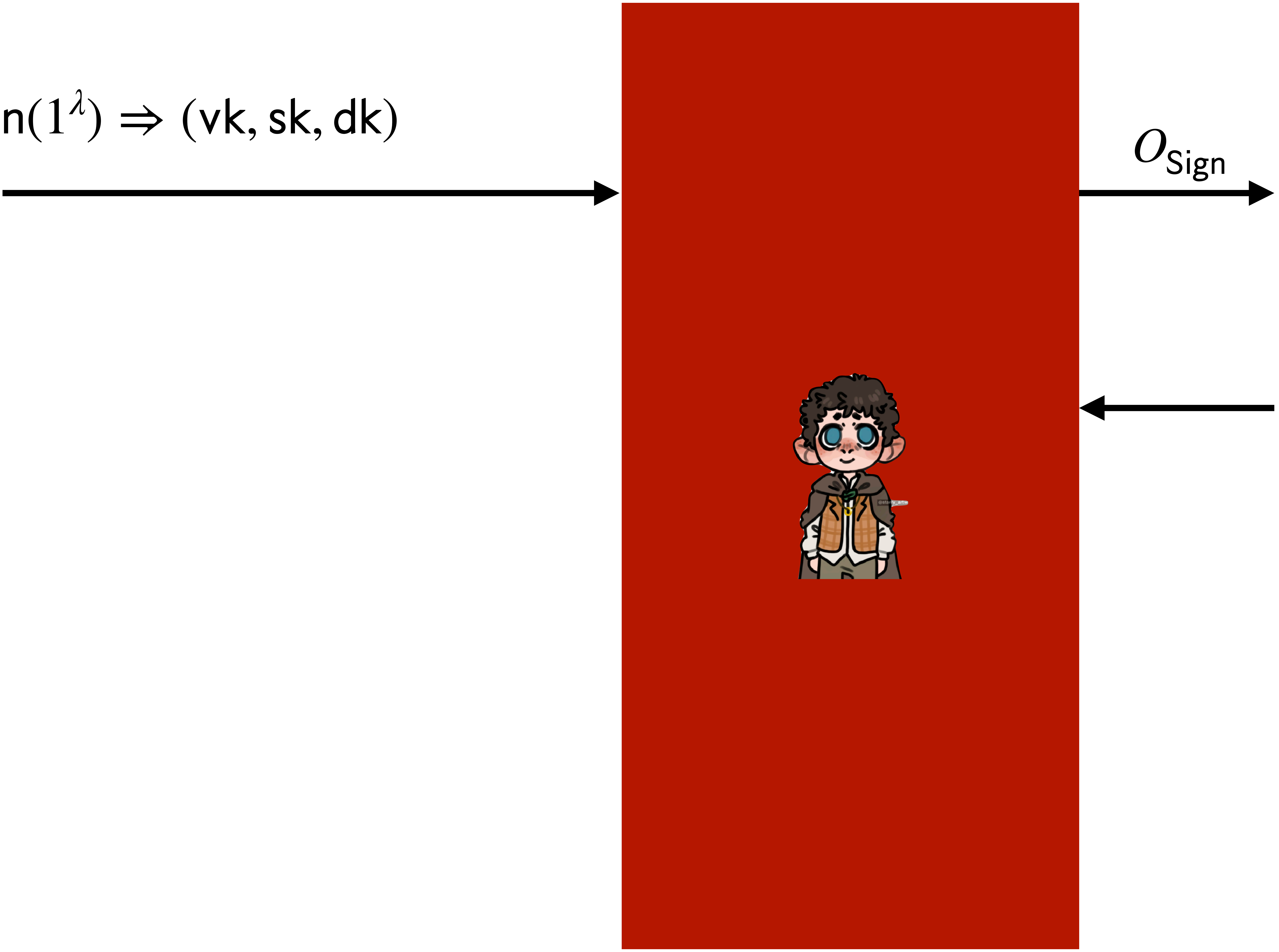
$S \leftarrow \emptyset$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

$(\mathsf{vk}, \mathsf{dk}) \longrightarrow$



Private Anamorphism Game [KPPYZ23]



$S \leftarrow \emptyset$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

$(\mathsf{vk}, \mathsf{dk}) \rightarrow$

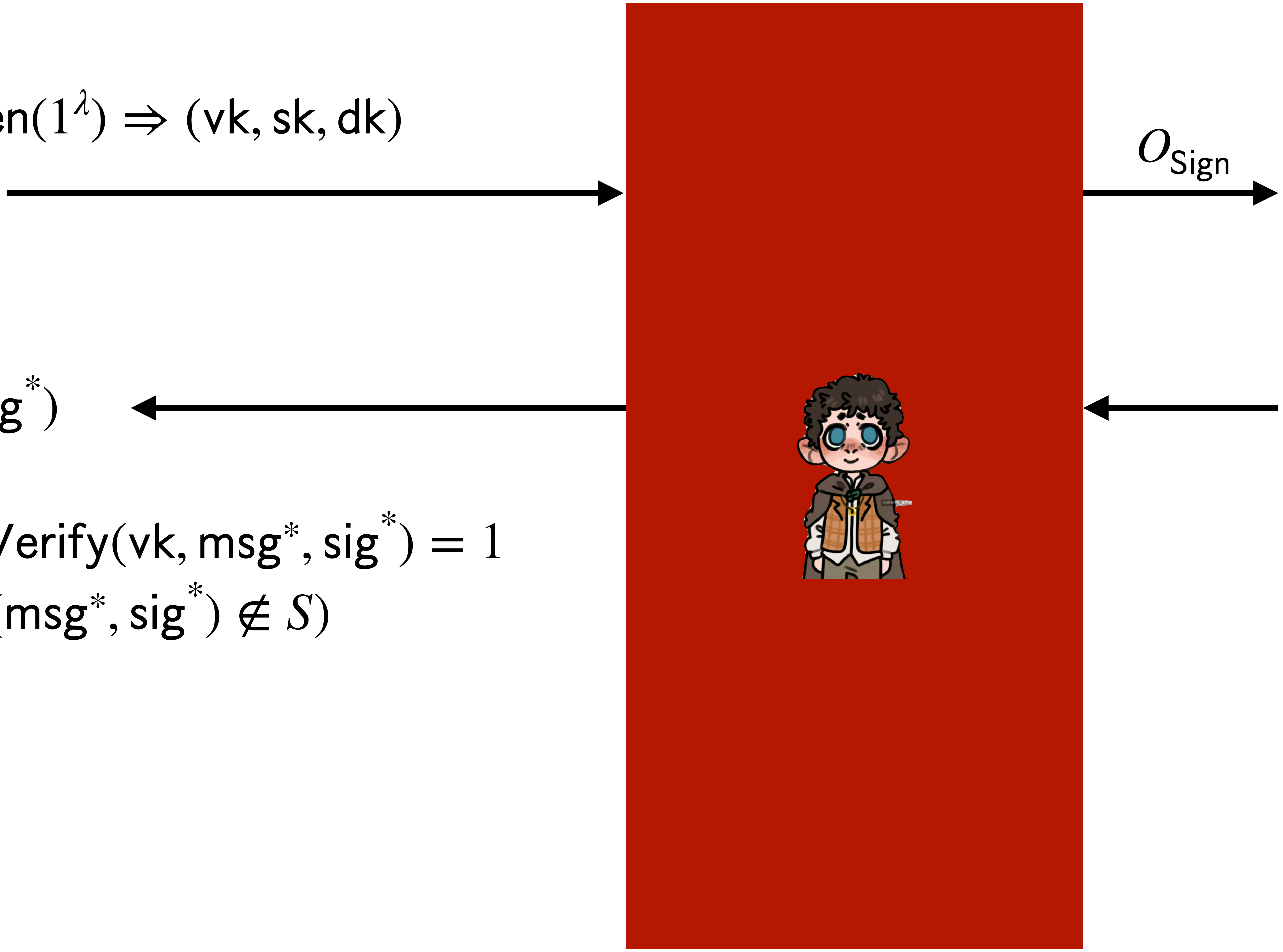
$\xrightarrow{O_{\mathsf{Sign}}}$ msg

$\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$

$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$

$\leftarrow$ sig

Private Anamorphism Game [KPPYZ23]



$S \leftarrow \emptyset$

$\mathsf{aKeyGen}(1^{\lambda}) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

$(\mathsf{vk}, \mathsf{dk})$

O_{Sign}

msg

$\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$

$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$

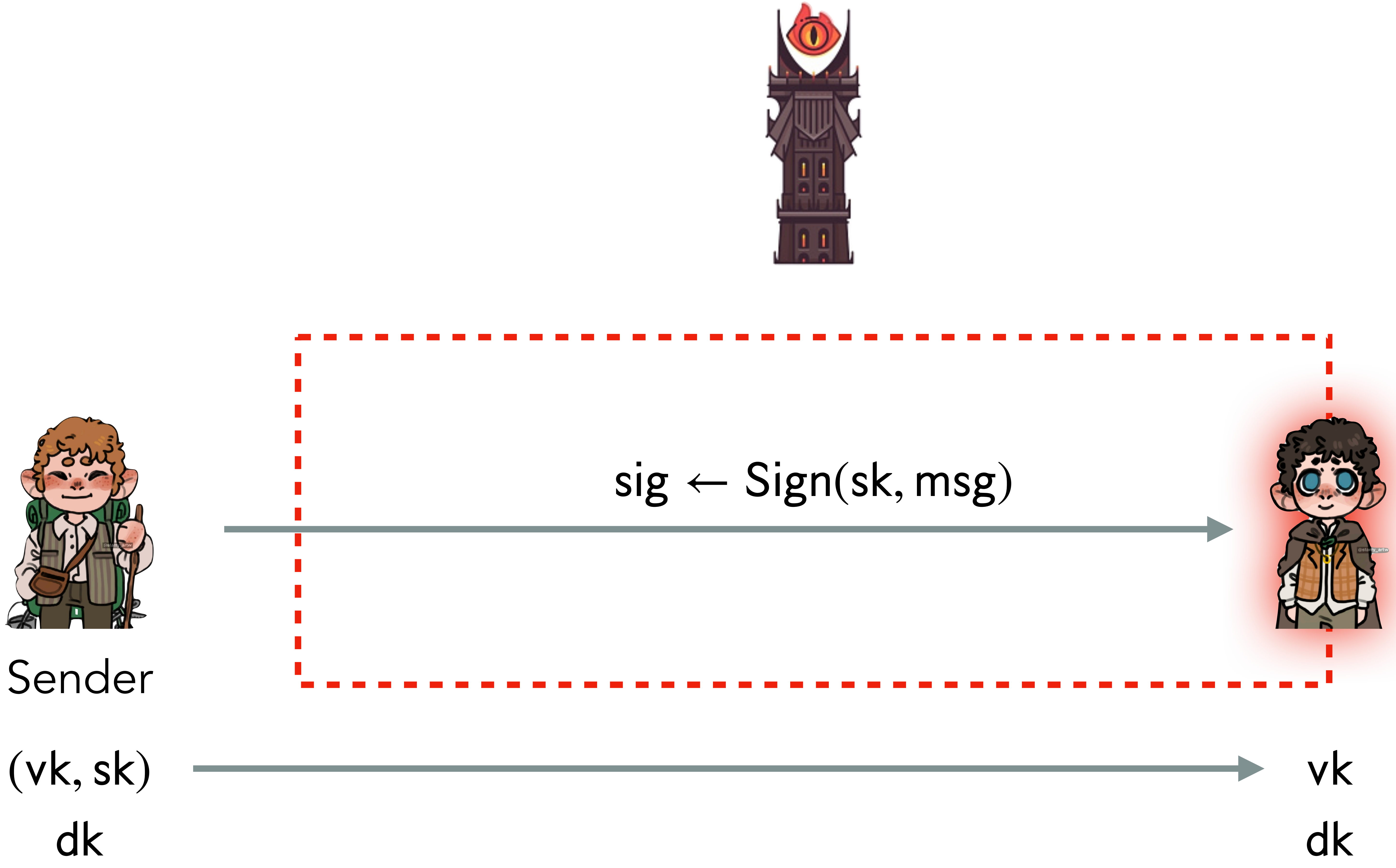
sig

$(\mathsf{msg}^*, \mathsf{sig}^*)$

wins if $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1$

$\wedge\ ((\mathsf{msg}^*, \mathsf{sig}^*) \notin S)$

Revisiting Anamorphic Threat Model



$\mathrm{sig} \leftarrow \mathrm{Sign}(\mathrm{sk}, \mathrm{msg})$

Sender

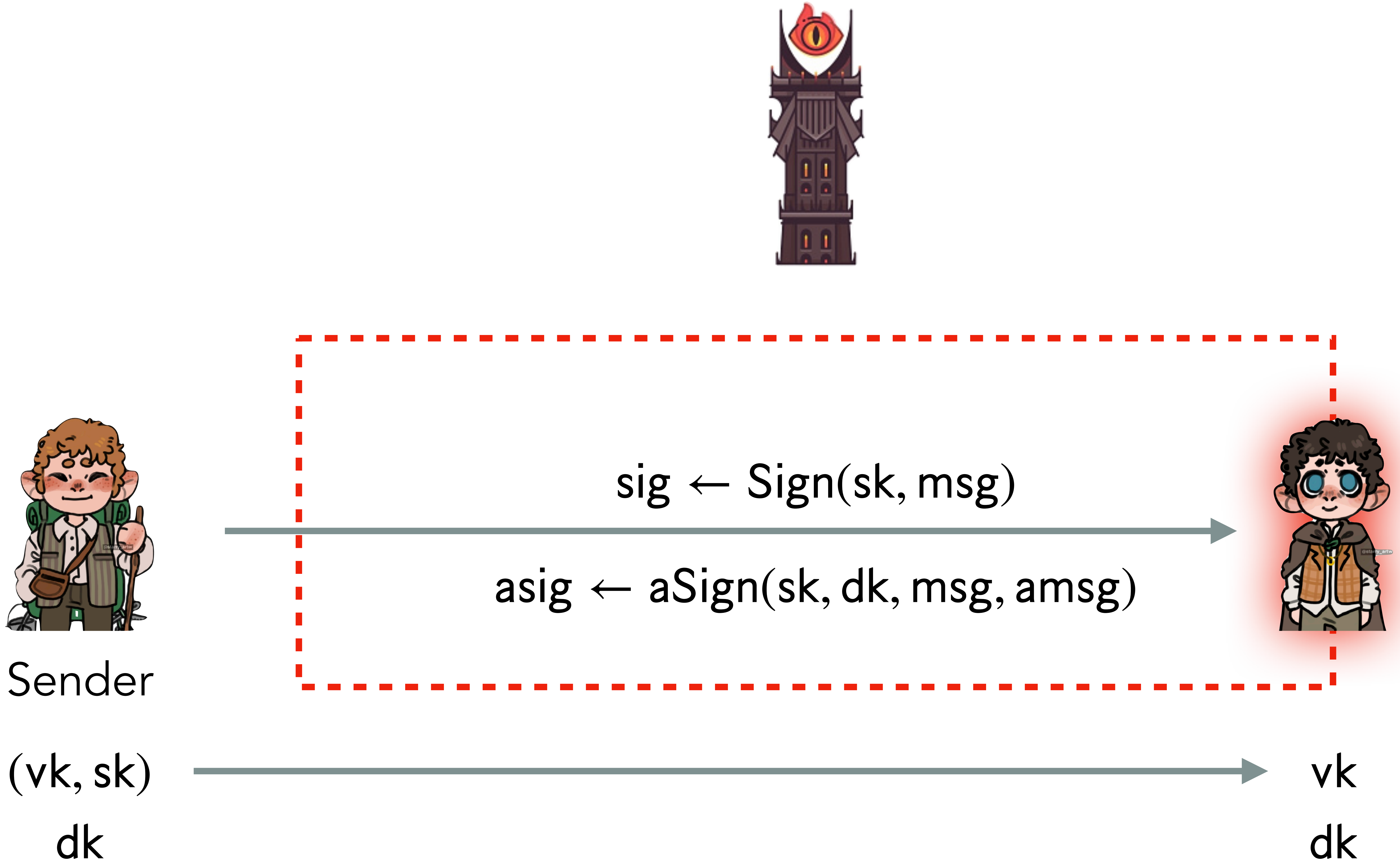
$(\mathrm{vk}, \mathrm{sk})$

dk

vk

dk

Revisiting Anamorphic Threat Model



sig ← Sign(sk, msg)

asig ← aSign(sk, dk, msg, amsg)

Sender

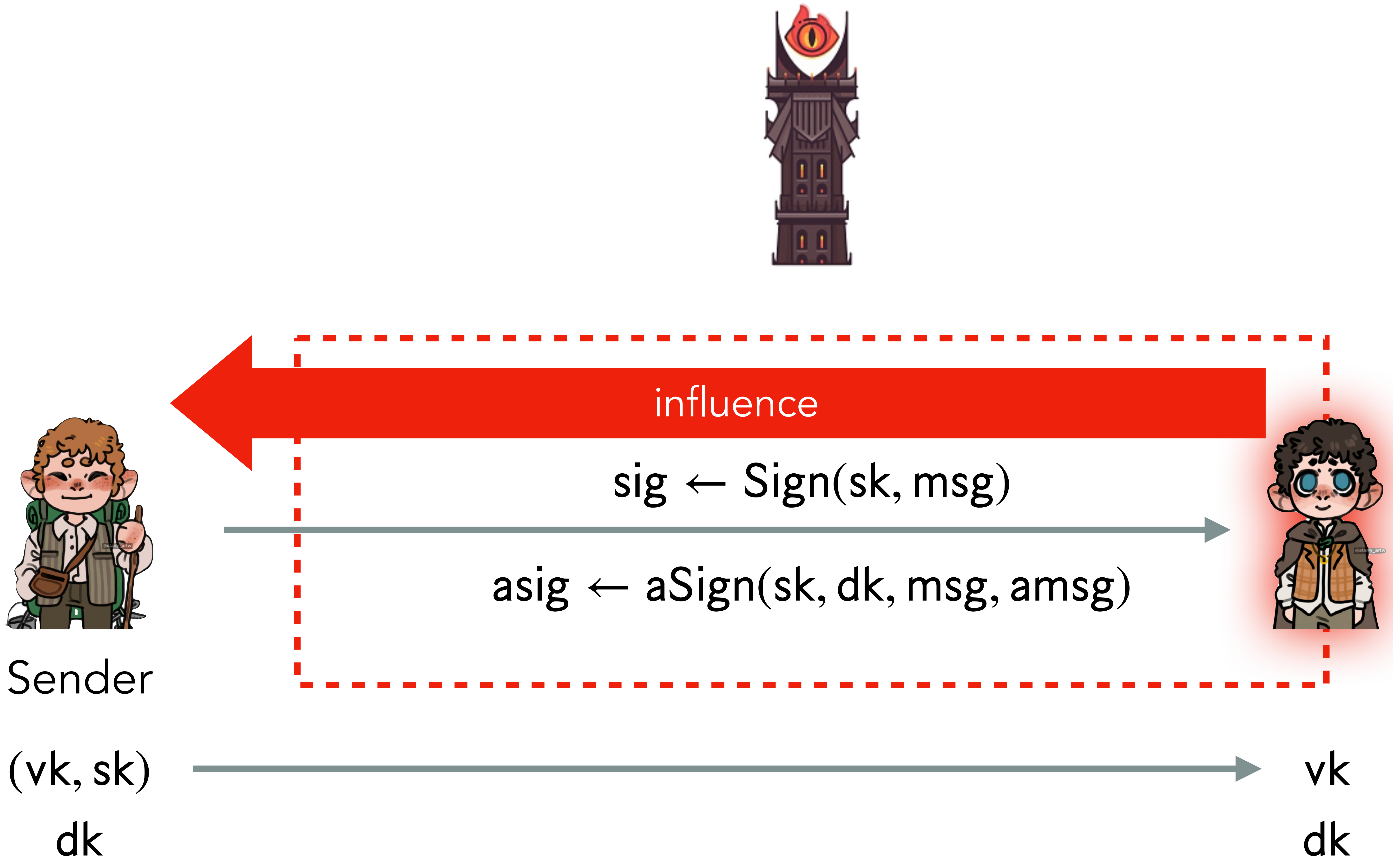
(vk, sk)

dk

vk

dk

Revisiting Anamorphic Threat Model



influence

sig ← Sign(sk, msg)

asig ← aSign(sk, dk, msg, amsg)

Sender

(vk, sk)

dk

vk

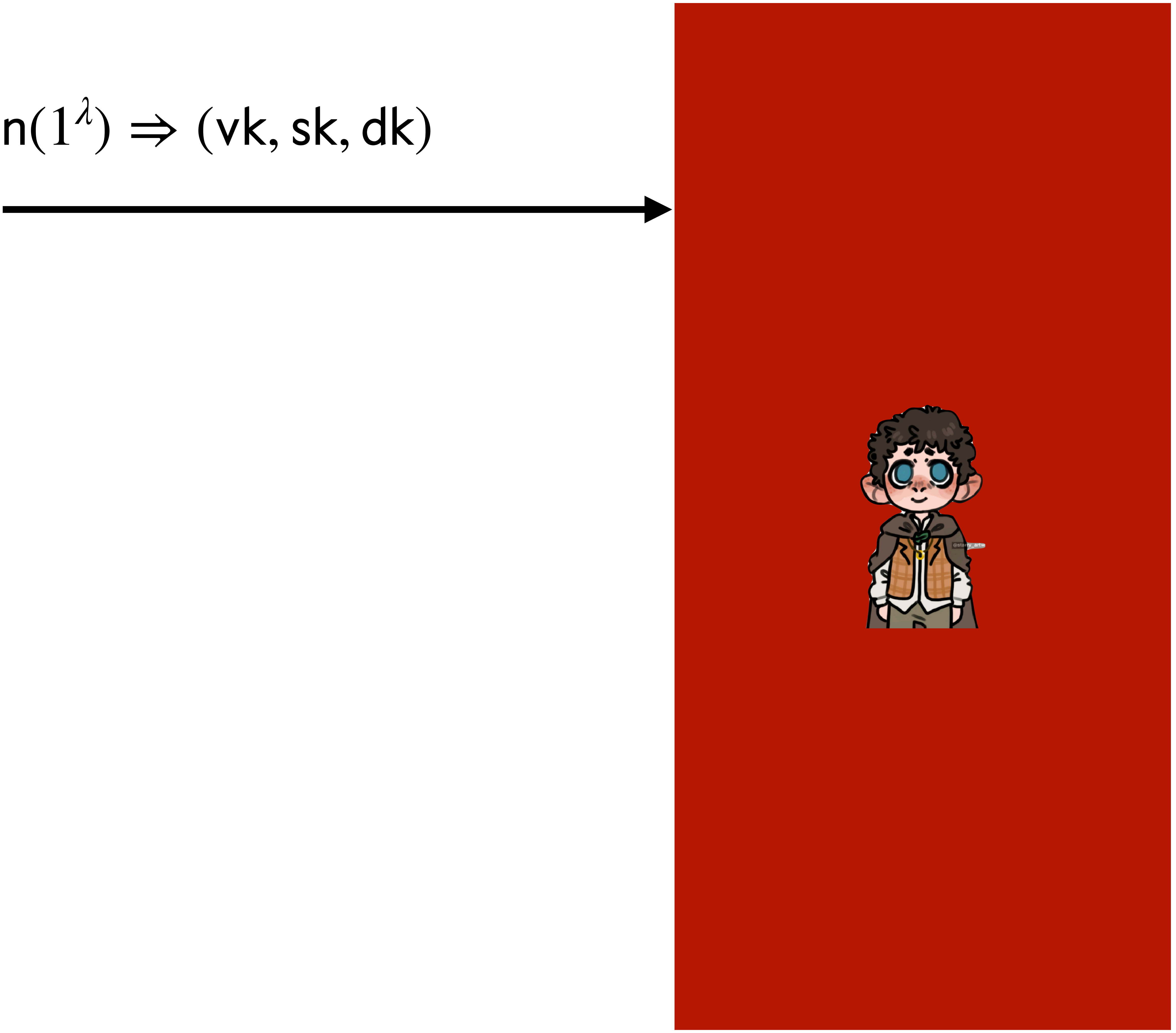
dk

Our Proposal: Recipient Unforgeability Game

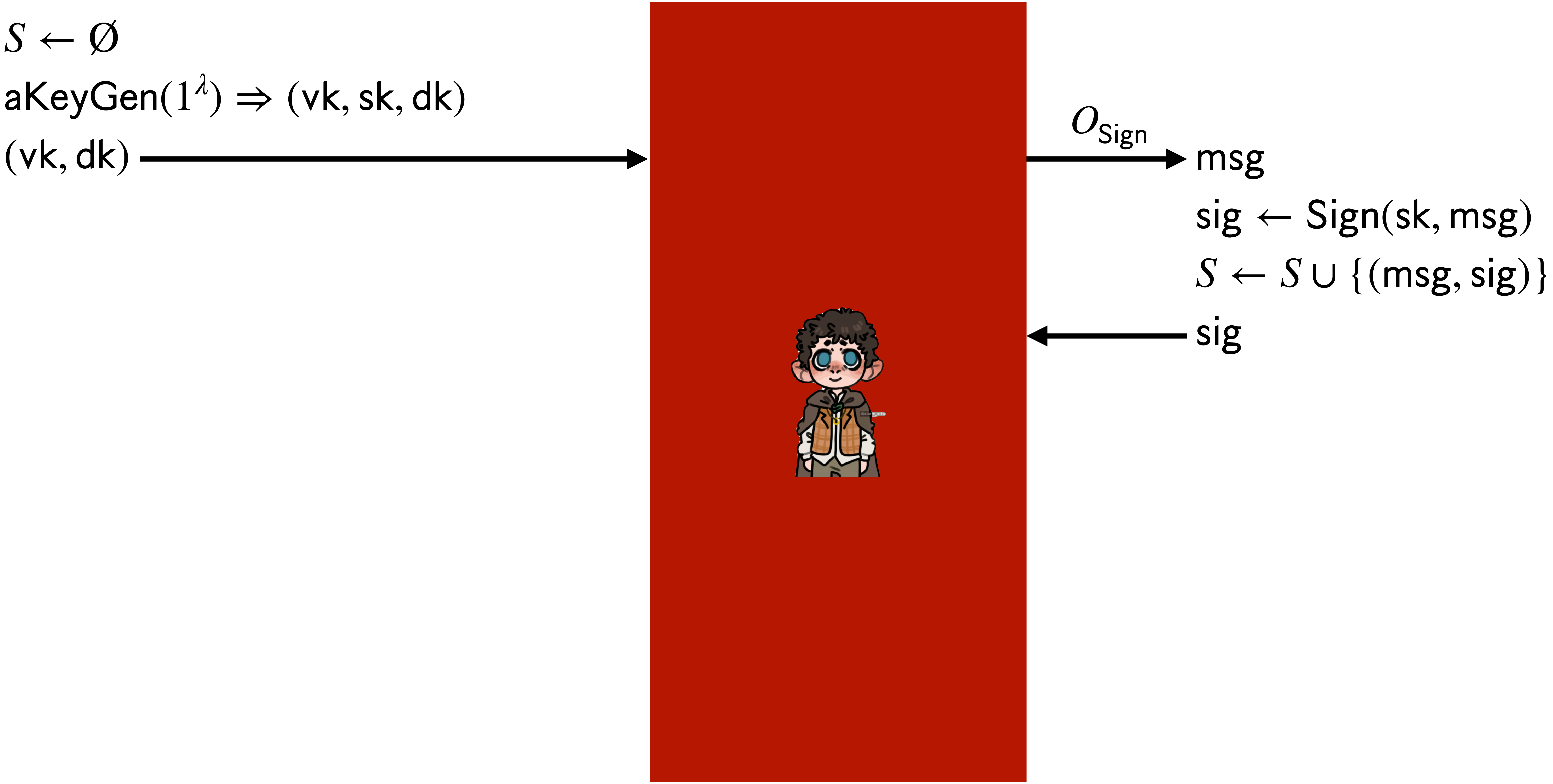
$S \leftarrow \emptyset$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

$(\mathsf{vk}, \mathsf{dk})$



Our Proposal: Recipient Unforgeability Game



$S \leftarrow \emptyset$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

$(\mathsf{vk}, \mathsf{dk}) \longrightarrow$

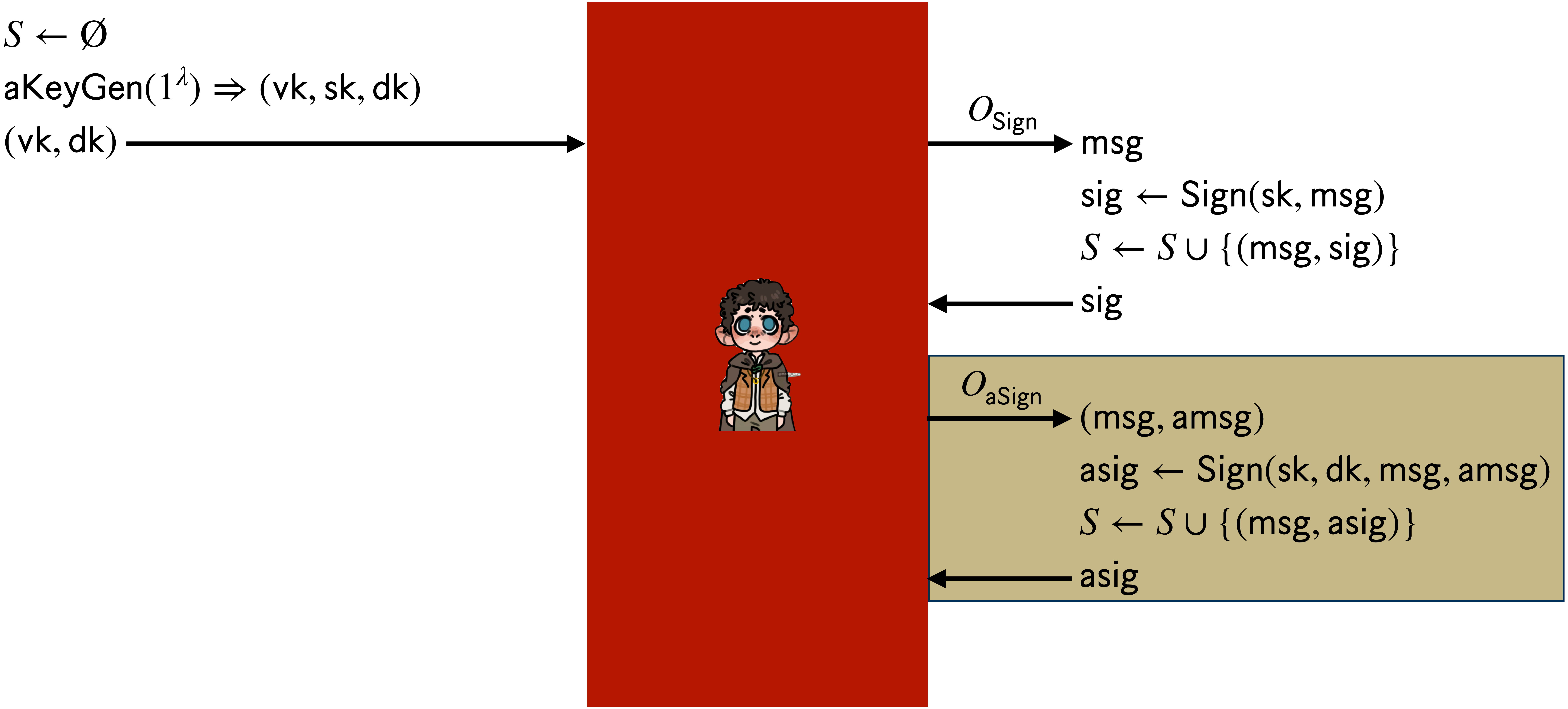
$\xrightarrow{O_{\mathsf{Sign}}}$ msg

$\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$

$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$

$\longleftarrow$ sig

Our Proposal: Recipient Unforgeability Game



$S \leftarrow \emptyset$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

$(\mathsf{vk}, \mathsf{dk}) \longrightarrow$

O_{Sign}

$\longrightarrow \mathsf{msg}$

$\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$

$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$

$\longleftarrow \mathsf{sig}$

O_{aSign}

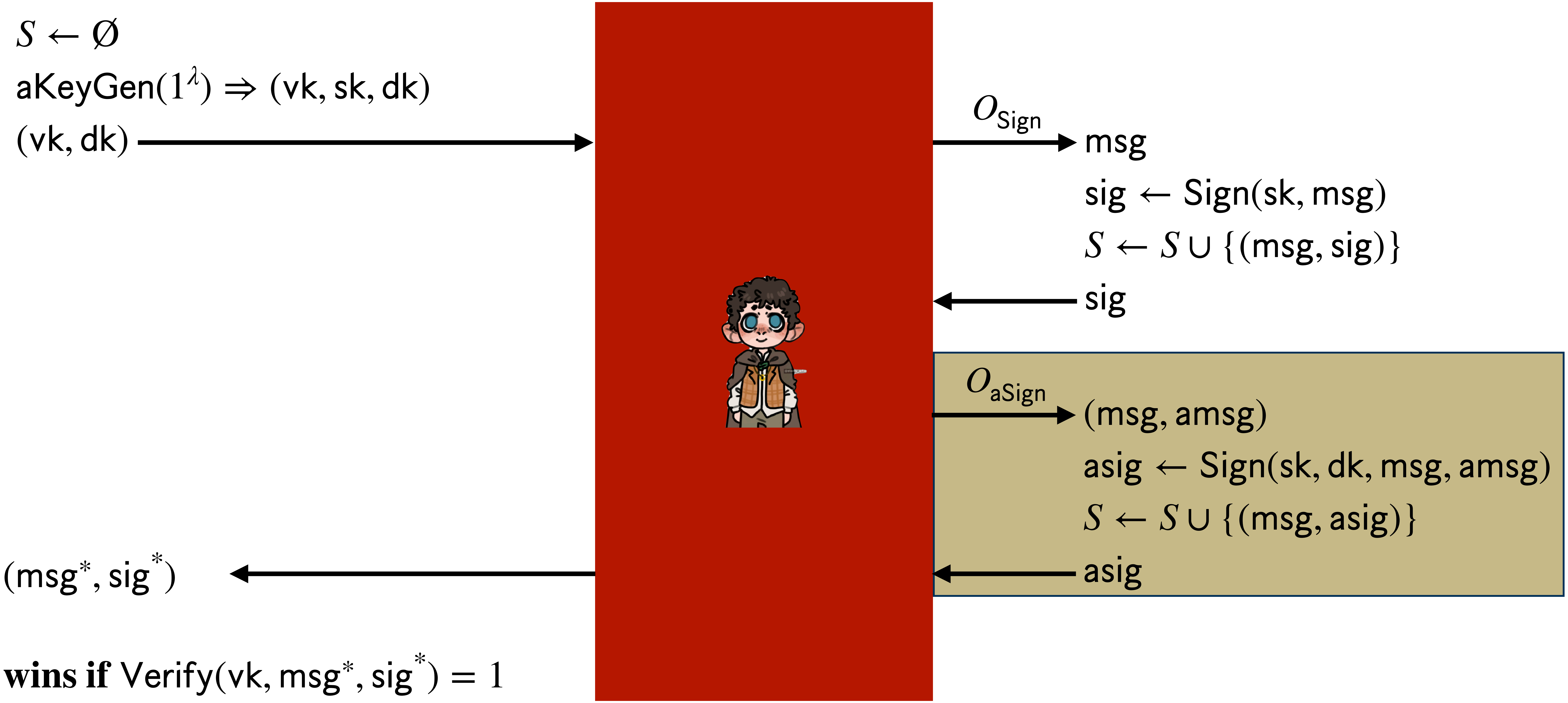
$\longrightarrow (\mathsf{msg}, \mathsf{amsg})$

$\mathsf{asig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$

$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{asig})\}$

$\longleftarrow \mathsf{asig}$

Our Proposal: Recipient Unforgeability Game



$S \leftarrow \emptyset$

$\mathsf{aKeyGen}(1^\lambda) \Rightarrow (\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

$(\mathsf{vk}, \mathsf{dk}) \longrightarrow$

O_{Sign}

$\longrightarrow \mathsf{msg}$

$\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$

$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$

$\longleftarrow \mathsf{sig}$

O_{aSign}

$\longrightarrow (\mathsf{msg}, \mathsf{amsg})$

$\mathsf{asig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$

$S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{asig})\}$

$\longleftarrow \mathsf{asig}$

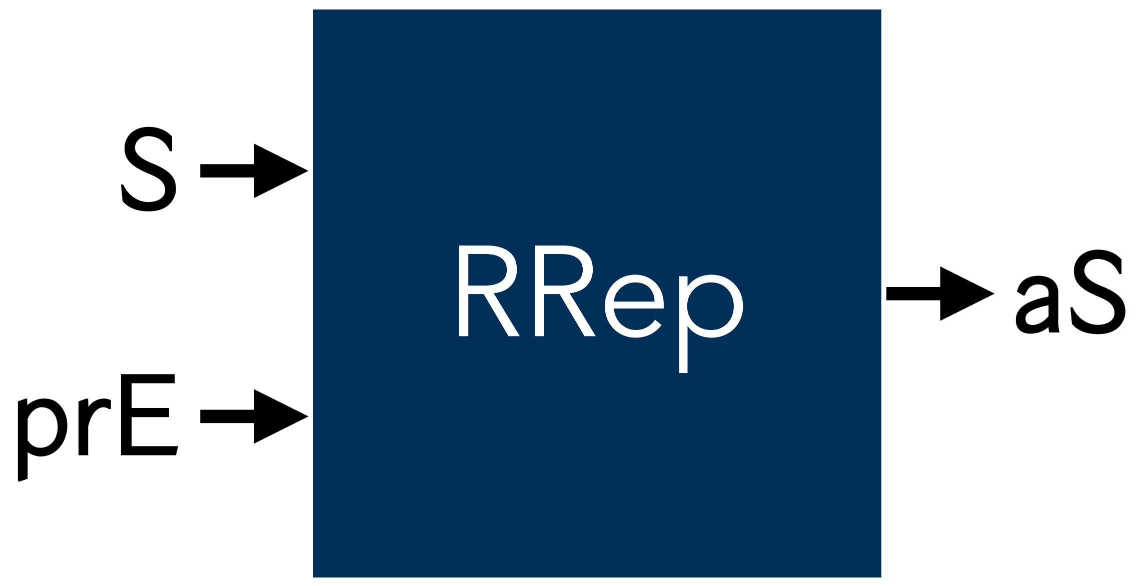
$(\mathsf{msg}^*, \mathsf{sig}^*) \longleftarrow$

wins if $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1$

$\wedge ((\mathsf{msg}^*, \mathsf{sig}^*) \notin S)$

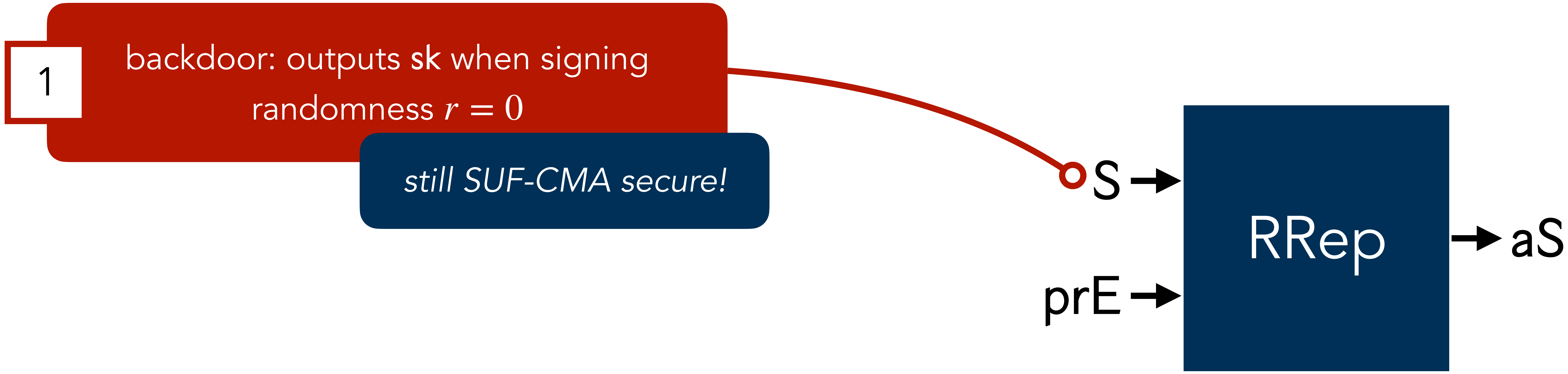
Recipient Attacking RRep: The Ingredients

- Recall RRep replaces signing randomness with $r \leftarrow \mathsf{prE}\,.\,\mathsf{Enc}(\mathsf{dk}, \mathsf{amsg})$.



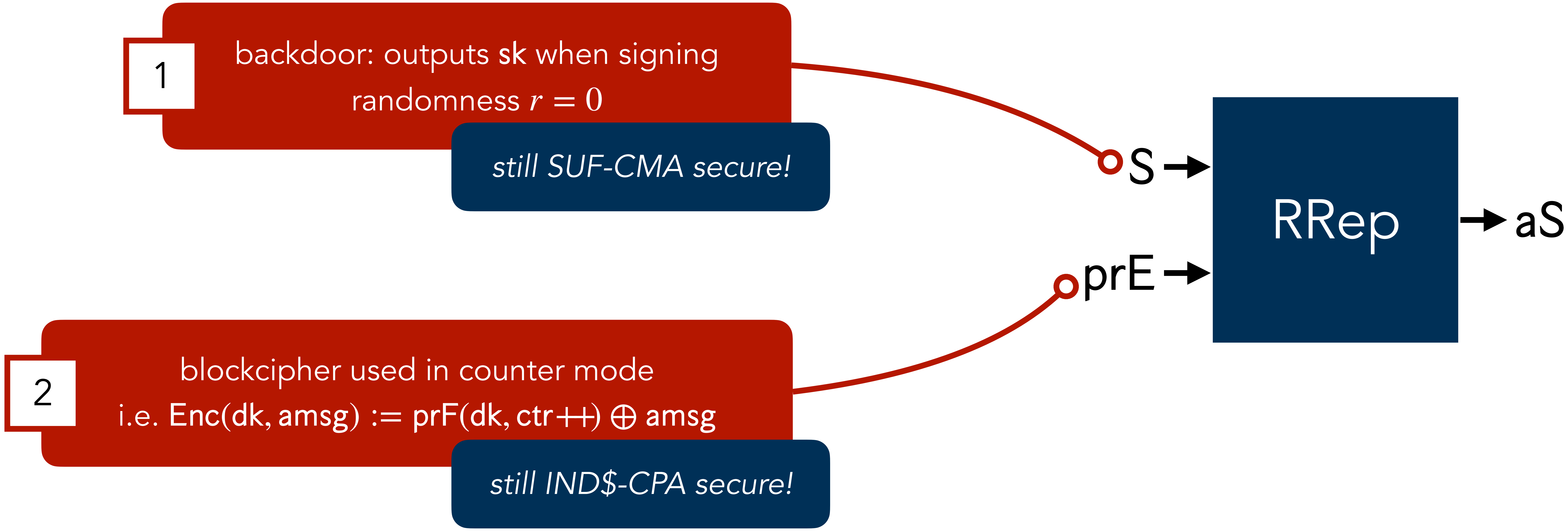
Recipient Attacking RRep: The Ingredients

- Recall RRep replaces signing randomness with $r \leftarrow \mathsf{prE}.\mathsf{Enc}(\mathsf{dk}, \mathsf{amsg})$.



Recipient Attacking RRep: The Ingredients

- Recall RRep replaces signing randomness with $r \leftarrow \mathsf{prE}\,.\,\mathsf{Enc}(\mathsf{dk}, \mathsf{amsg})$.



1 backdoor: outputs sk when signing randomness $r = 0$

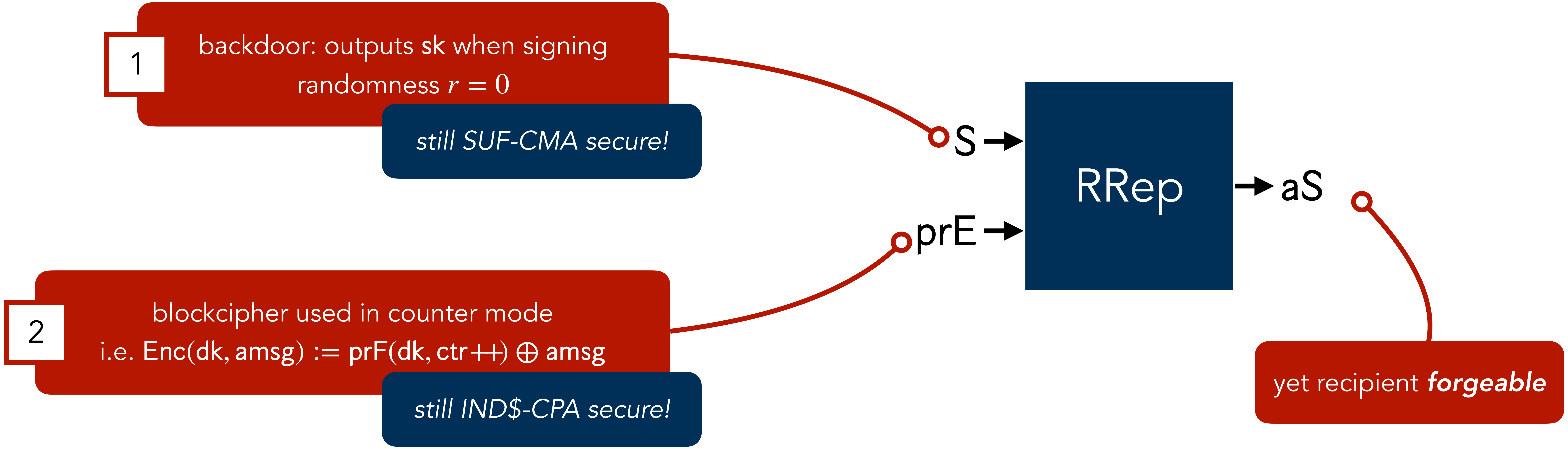
still SUF-CMA secure!

2 blockcipher used in counter mode i.e. $\mathsf{Enc}(\mathsf{dk}, \mathsf{amsg}) := \mathsf{prF}(\mathsf{dk}, \mathsf{ctr}{+}{+}) \oplus \mathsf{amsg}$

still IND$-CPA secure!

Recipient Attacking RRep: The Ingredients

- Recall RRep replaces signing randomness with $r \leftarrow$ prE . Enc(dk, amsg).



1
backdoor: outputs sk when signing randomness $r = 0$
still SUF-CMA secure!
S
prE
RRep
aS
2
blockcipher used in counter mode
i.e. Enc(dk, amsg) := prF(dk, ctr++) ⊕ amsg
still IND\$-CPA secure!
yet recipient ***forgeable***

Recipient Attacking RRep: The Attack

Sender

Receiver

Recipient Attacking RRep: The Attack

Sender

Receiver

$\mathsf{asig}_i \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{amsg}_i) \xrightarrow{\mathsf{asig}_0,\, \mathsf{asig}_1,\, \ldots,\, \mathsf{asig}_\ell}$

Recipient Attacking RRep: The Attack

Sender

Receiver

$$\mathsf{asig}_i \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{amsg}_i) \xrightarrow{\mathsf{asig}_0,\, \mathsf{asig}_1,\, \ldots,\, \mathsf{asig}_\ell} \mathsf{amsg}_i \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{asig}_i)$$

Recipient Attacking RRep: The Attack

Sender

Receiver

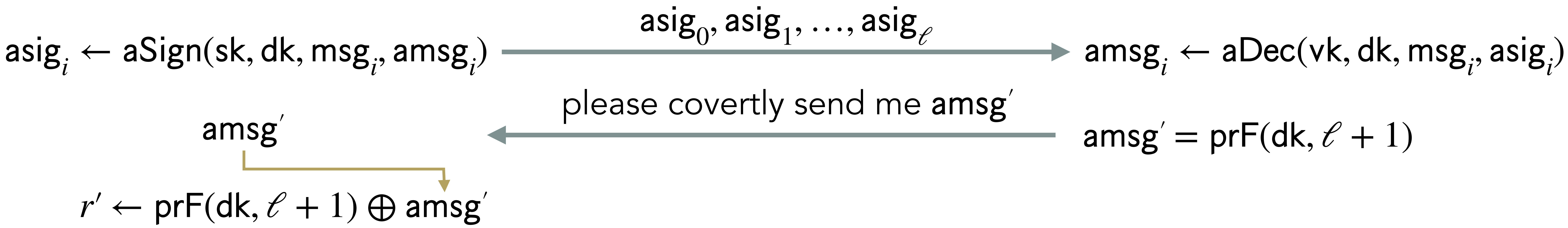
$\mathsf{asig}_i \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{amsg}_i) \xrightarrow{\mathsf{asig}_0, \mathsf{asig}_1, \ldots, \mathsf{asig}_\ell} \mathsf{amsg}_i \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{asig}_i)$

$\mathsf{amsg}' \xleftarrow{\text{please covertly send me } \mathsf{amsg}'} \mathsf{amsg}' = \mathsf{prF}(\mathsf{dk}, \ell + 1)$

Recipient Attacking RRep: The Attack

Sender

Receiver



$\mathsf{asig}_i \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{amsg}_i)$

$\mathsf{asig}_0, \mathsf{asig}_1, \ldots, \mathsf{asig}_\ell$

$\mathsf{amsg}_i \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{asig}_i)$

please covertly send me amsg'

$\mathsf{amsg}' = \mathsf{prF}(\mathsf{dk}, \ell + 1)$

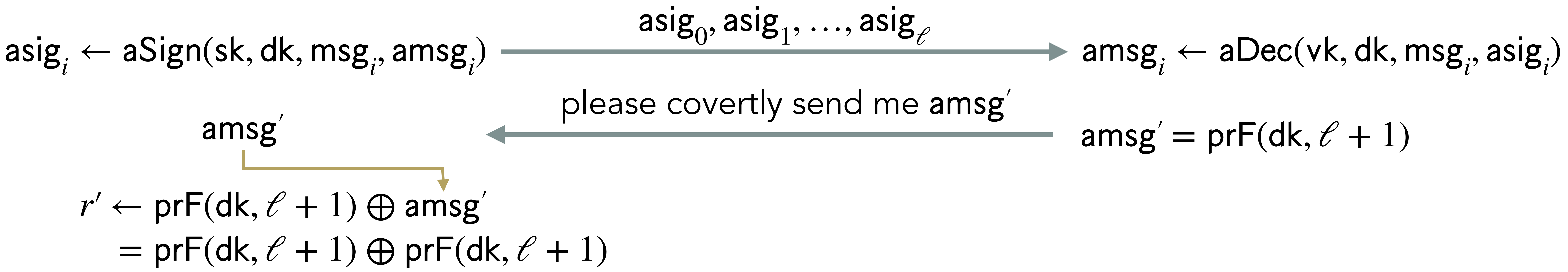
amsg'

$r' \leftarrow \mathsf{prF}(\mathsf{dk}, \ell + 1) \oplus \mathsf{amsg}'$

Recipient Attacking RRep: The Attack

Sender

Receiver



$\mathsf{asig}_i \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{amsg}_i)$ $\xrightarrow{\mathsf{asig}_0, \mathsf{asig}_1, \ldots, \mathsf{asig}_\ell}$ $\mathsf{amsg}_i \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{asig}_i)$

amsg' $\xleftarrow{\text{please covertly send me } \mathsf{amsg}'}$ $\mathsf{amsg}' = \mathsf{prF}(\mathsf{dk}, \ell + 1)$

$$\begin{aligned} r' &\leftarrow \mathsf{prF}(\mathsf{dk}, \ell + 1) \oplus \mathsf{amsg}' \\ &= \mathsf{prF}(\mathsf{dk}, \ell + 1) \oplus \mathsf{prF}(\mathsf{dk}, \ell + 1) \end{aligned}$$

Recipient Attacking RRep: The Attack

Sender

Receiver

$\mathsf{asig}_0, \mathsf{asig}_1, \ldots, \mathsf{asig}_\ell$

$\mathsf{asig}_i \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{amsg}_i) \longrightarrow \mathsf{amsg}_i \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{asig}_i)$

please covertly send me amsg'

$\mathsf{amsg}' \longleftarrow \mathsf{amsg}' = \mathsf{prF}(\mathsf{dk}, \ell + 1)$

$$
\begin{aligned}
r' &\leftarrow \mathsf{prF}(\mathsf{dk}, \ell + 1) \oplus \mathsf{amsg}' \\
&= \mathsf{prF}(\mathsf{dk}, \ell + 1) \oplus \mathsf{prF}(\mathsf{dk}, \ell + 1) \\
&= 0
\end{aligned}
$$

Recipient Attacking RRep: The Attack

Sender

Receiver

$$\mathsf{asig}_i \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{amsg}_i) \xrightarrow{\mathsf{asig}_0,\, \mathsf{asig}_1,\, \ldots,\, \mathsf{asig}_\ell} \mathsf{amsg}_i \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{asig}_i)$$

$$\mathsf{amsg}' \xleftarrow{\text{please covertly send me } \mathsf{amsg}'} \mathsf{amsg}' = \mathsf{prF}(\mathsf{dk}, \ell + 1)$$

$$\begin{aligned} r' &\leftarrow \mathsf{prF}(\mathsf{dk}, \ell + 1) \oplus \mathsf{amsg}' \\ &= \mathsf{prF}(\mathsf{dk}, \ell + 1) \oplus \mathsf{prF}(\mathsf{dk}, \ell + 1) \\ &= 0 \end{aligned}$$

$$\mathsf{asig}' \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r')$$

Recipient Attacking RRep: The Attack

Sender

Receiver

$$\mathsf{asig}_i \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{amsg}_i) \xrightarrow{\mathsf{asig}_0, \mathsf{asig}_1, \ldots, \mathsf{asig}_\ell} \mathsf{amsg}_i \leftarrow \mathsf{aDec}(\mathsf{vk}, \mathsf{dk}, \mathsf{msg}_i, \mathsf{asig}_i)$$

$$\mathsf{amsg}' \xleftarrow{\text{please covertly send me } \mathsf{amsg}'} \mathsf{amsg}' = \mathsf{prF}(\mathsf{dk}, \ell + 1)$$

$$\begin{aligned} r' &\leftarrow \mathsf{prF}(\mathsf{dk}, \ell + 1) \oplus \mathsf{amsg}' \\ &= \mathsf{prF}(\mathsf{dk}, \ell + 1) \oplus \mathsf{prF}(\mathsf{dk}, \ell + 1) \\ &= 0 \end{aligned}$$

$$\mathsf{asig}' \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r')$$

triggers backdoor!

Recipient Attacking RRep: The Attack

Sender

Receiver

$$\text{asig}_i \leftarrow \text{aSign}(\text{sk}, \text{dk}, \text{msg}_i, \text{amsg}_i) \xrightarrow{\text{asig}_0, \text{asig}_1, \ldots, \text{asig}_\ell} \text{amsg}_i \leftarrow \text{aDec}(\text{vk}, \text{dk}, \text{msg}_i, \text{asig}_i)$$

$$\text{amsg}' \xleftarrow{\text{please covertly send me amsg}'} \text{amsg}' = \text{prF}(\text{dk}, \ell + 1)$$

$$\begin{aligned} r' &\leftarrow \text{prF}(\text{dk}, \ell + 1) \oplus \text{amsg}' \\ &= \text{prF}(\text{dk}, \ell + 1) \oplus \text{prF}(\text{dk}, \ell + 1) \\ &= 0 \end{aligned}$$

triggers backdoor!

$$\begin{aligned} \text{asig}' &\leftarrow \text{Sign}(\text{sk}, \text{msg}; r') \\ &= \text{sk} \end{aligned} \xrightarrow{\text{asig}' = \text{sk}}$$

Recipient Attacking RRep: The Attack

Sender

Receiver

$\text{asig}_i \leftarrow \text{aSign}(\text{sk}, \text{dk}, \text{msg}_i, \text{amsg}_i)$ $\xrightarrow{\text{asig}_0, \text{asig}_1, \ldots, \text{asig}_\ell}$ $\text{amsg}_i \leftarrow \text{aDec}(\text{vk}, \text{dk}, \text{msg}_i, \text{asig}_i)$

amsg' $\xleftarrow{\text{please covertly send me } \textbf{amsg}'}$ $\text{amsg}' = \text{prF}(\text{dk}, \ell + 1)$

$$\begin{aligned} r' &\leftarrow \text{prF}(\text{dk}, \ell + 1) \oplus \text{amsg}' \\ &= \text{prF}(\text{dk}, \ell + 1) \oplus \text{prF}(\text{dk}, \ell + 1) \\ &= 0 \end{aligned}$$

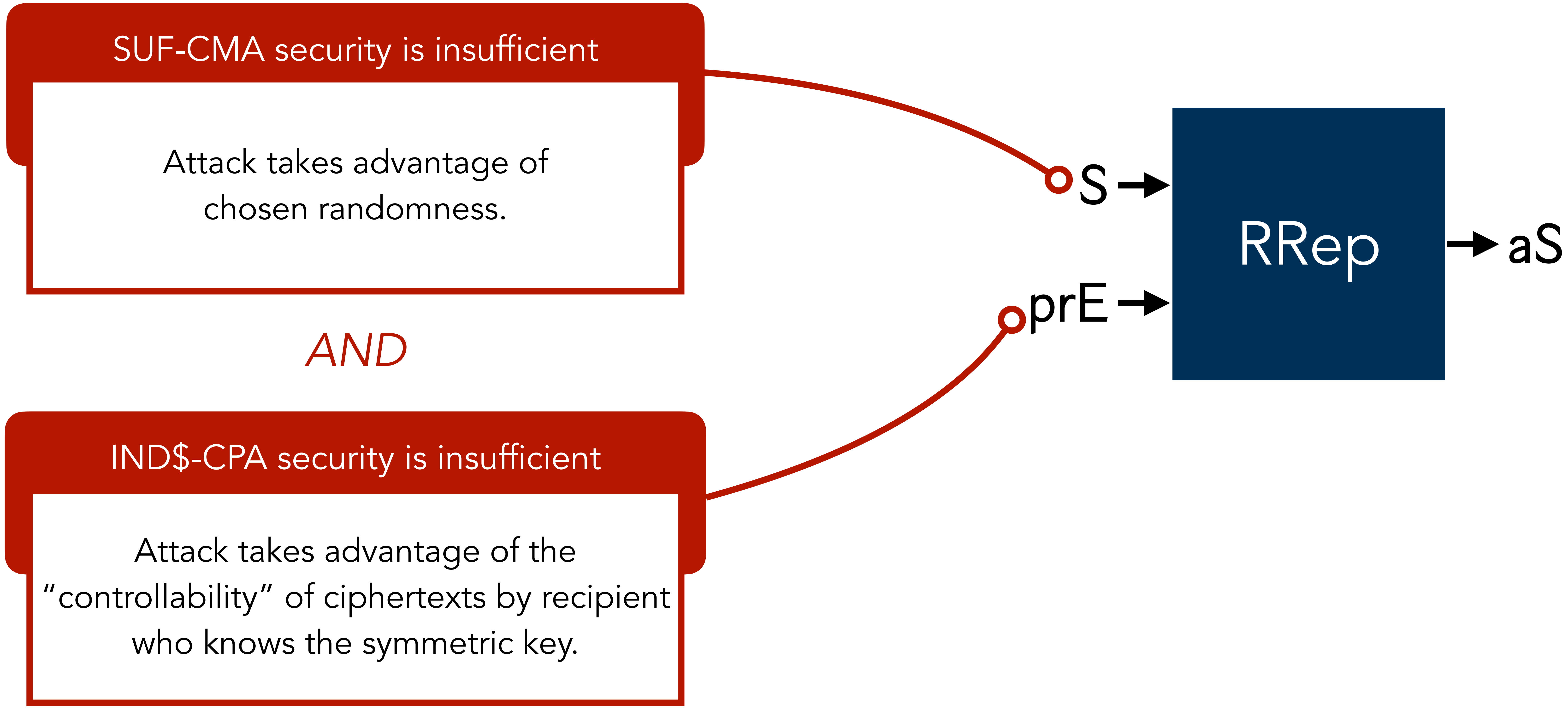
triggers backdoor!

$$\begin{aligned} \text{asig}' &\leftarrow \text{Sign}(\text{sk}, \text{msg}; r') \\ &= \text{sk} \end{aligned}$$

$\xrightarrow{\text{asig}' = \text{sk}}$

use **sk** to forge **sig*** on any **msg*** of choosing!

Repairing Recipient Unforgeability of RRep



SUF-CMA security is insufficient

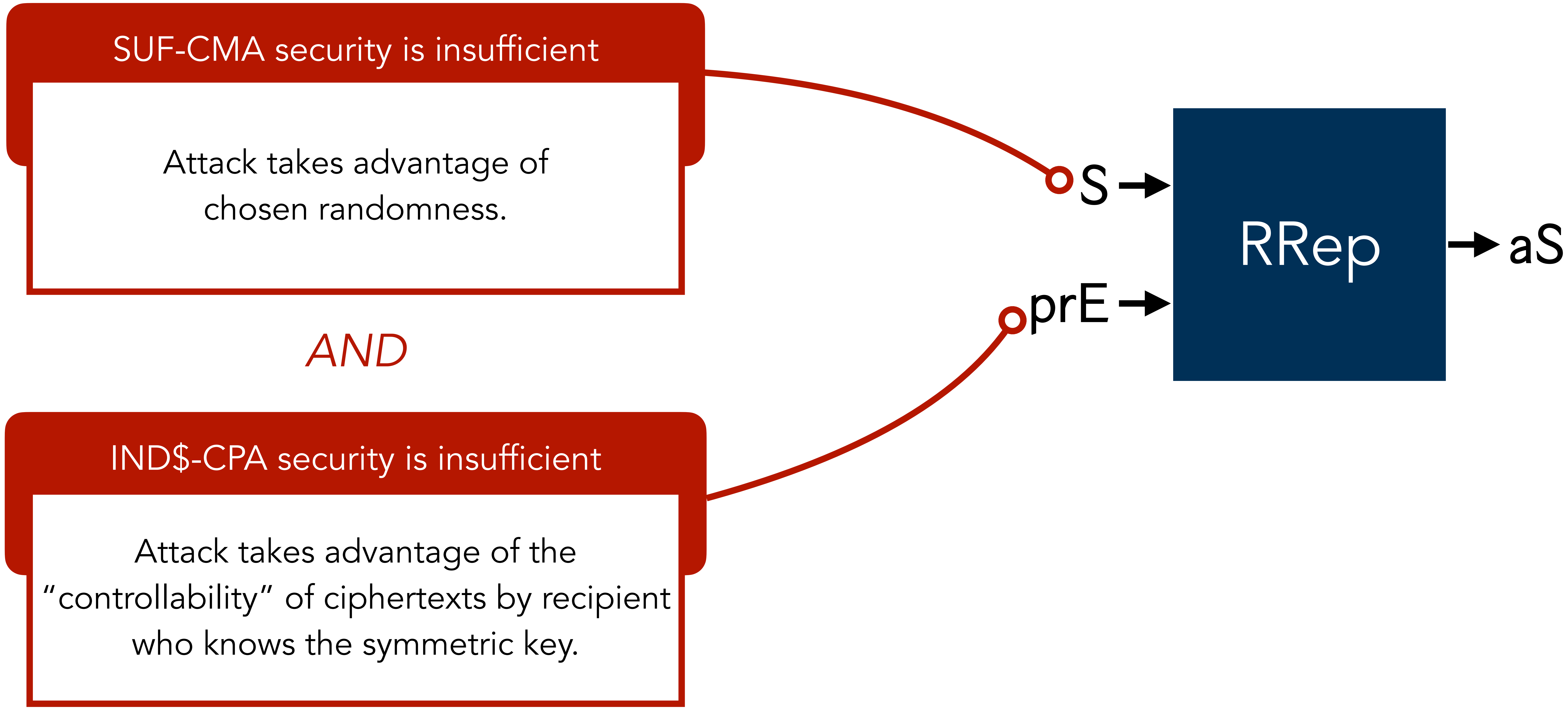
Attack takes advantage of chosen randomness.

AND

IND$-CPA security is insufficient

Attack takes advantage of the "controllability" of ciphertexts by recipient who knows the symmetric key.

Repairing Recipient Unforgeability of RRep



SUF-CMA security is insufficient

Attack takes advantage of chosen randomness.

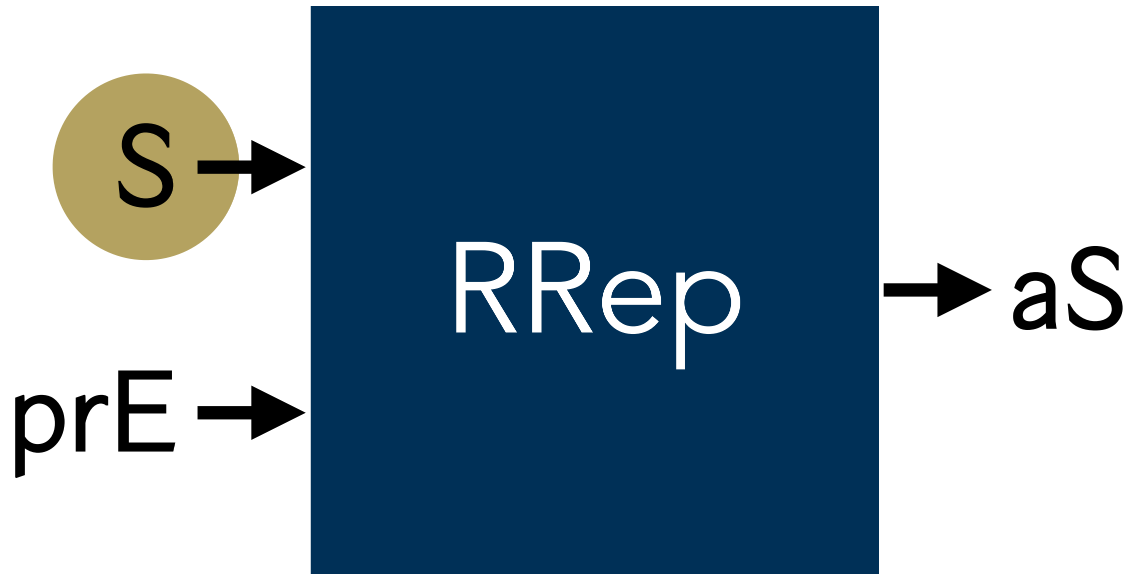
AND

IND$-CPA security is insufficient

Attack takes advantage of the "controllability" of ciphertexts by recipient who knows the symmetric key.

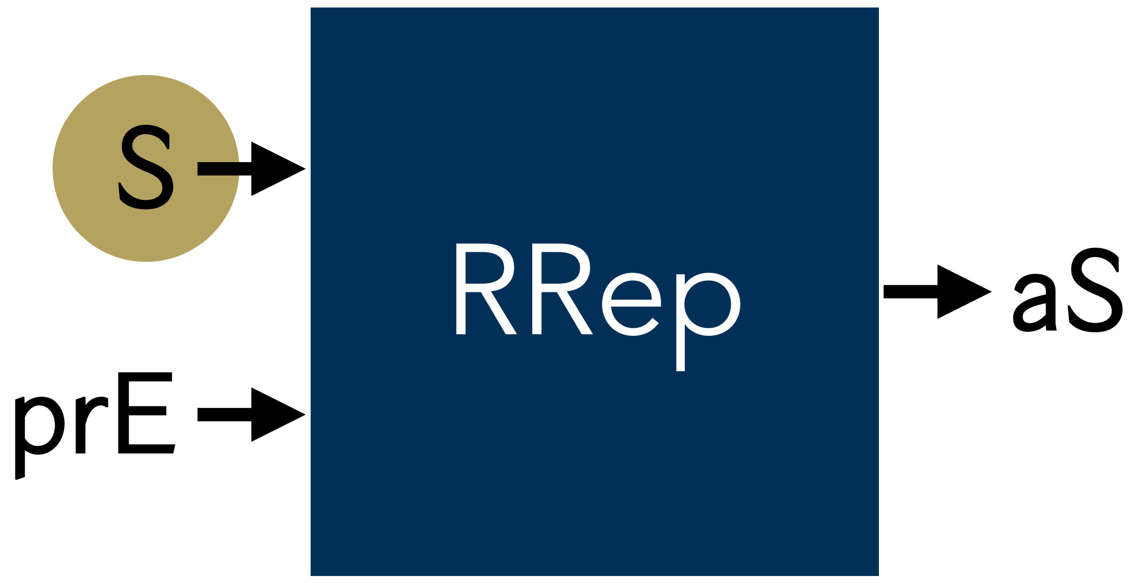
- Attack leverages insufficiencies in ***both*** signature scheme and pseudorandom encryption. Can regain security by requiring stronger security properties of ***either one*** component.

Repairing Recipient Unforgeability of RRep



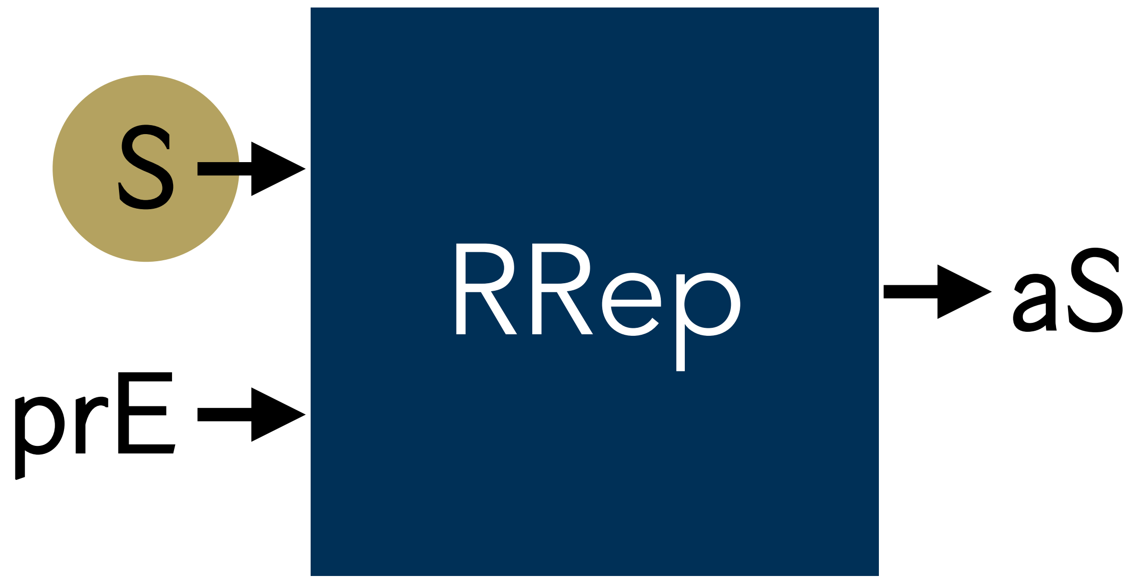
Repairing Recipient Unforgeability of RRep

- Can achieve recipient unforgeability by requiring stronger property on signature scheme S.



Repairing Recipient Unforgeability of RRep

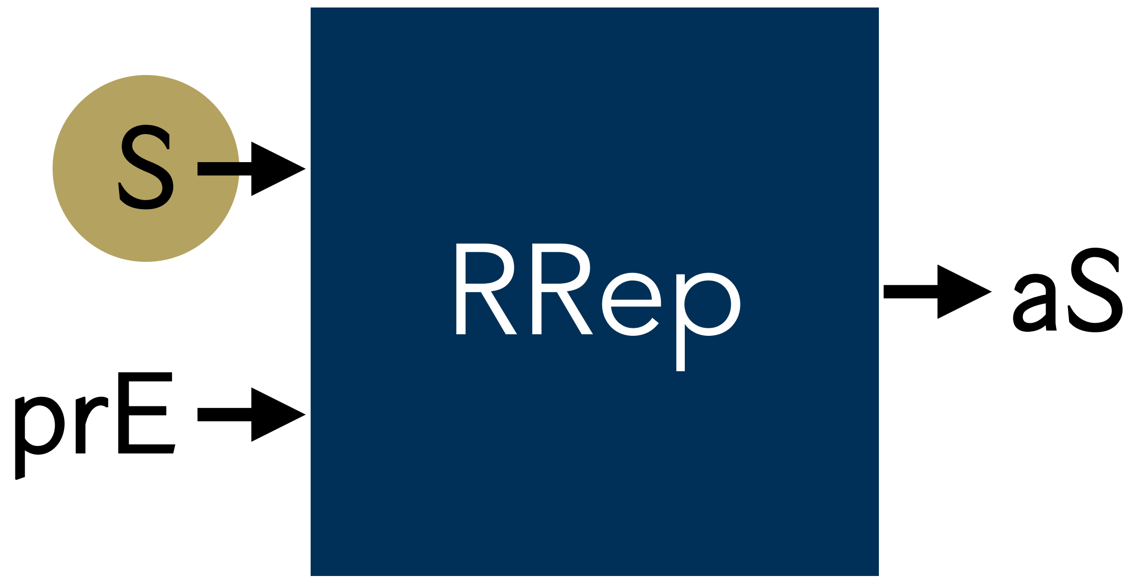
- Can achieve recipient unforgeability by requiring stronger property on signature scheme S.



- Unforgeability under chosen randomness attack (SUF-CRA security) akin to SUF-CMA security except adversary queries for signatures on messages ***and randomness*** of its choosing.

Repairing Recipient Unforgeability of RRep

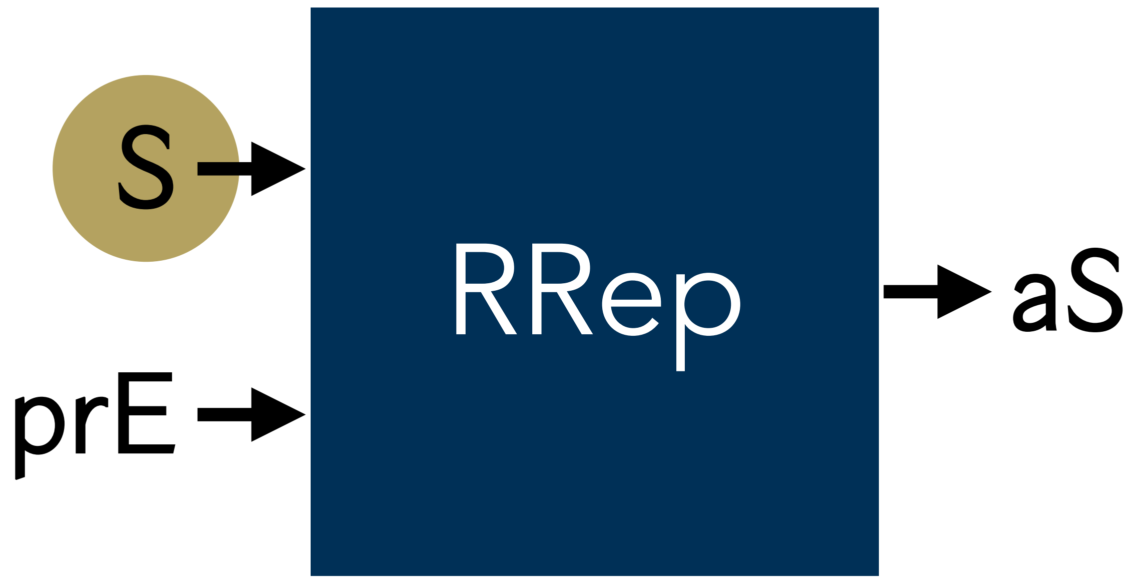
- Can achieve recipient unforgeability by requiring stronger property on signature scheme S.



- Unforgeability under chosen randomness attack (SUF-CRA security) akin to SUF-CMA security except adversary queries for signatures on messages ***and randomness*** of its choosing.

- We prove that RSA-PSS and Rabin signatures are SUF-CRA secure, hence anamorphic RSA-PSS and Rabin from RRep are recipient unforgeable.

Repairing Recipient Unforgeability of RRep



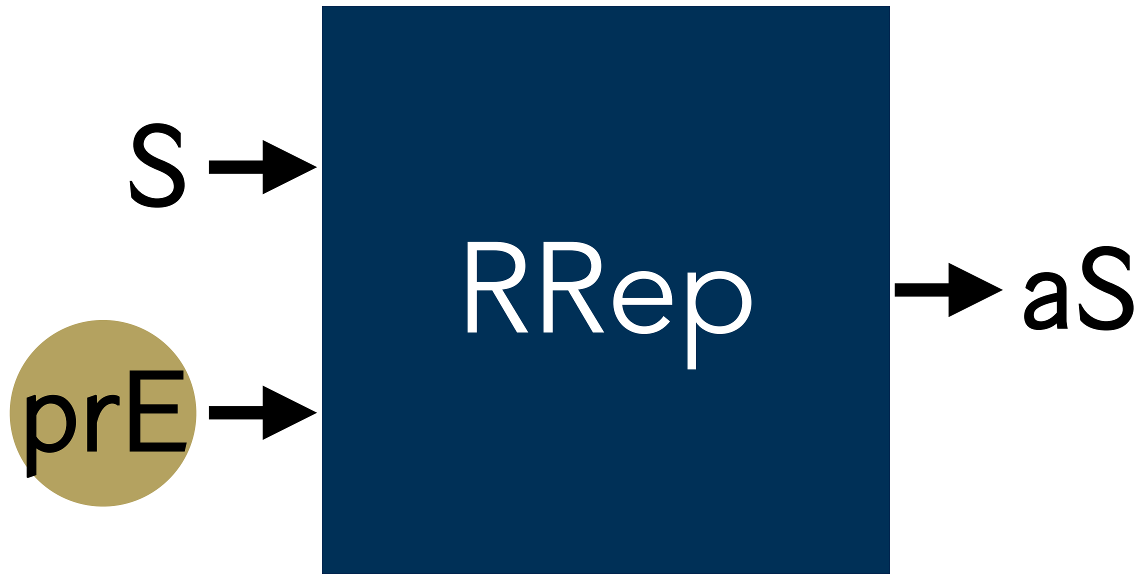
- Can achieve recipient unforgeability by requiring stronger property on signature scheme S.

- Unforgeability under chosen randomness attack (SUF-CRA security) akin to SUF-CMA security except adversary queries for signatures on messages ***and randomness*** of its choosing.

- We prove that RSA-PSS and Rabin signatures are SUF-CRA secure, hence anamorphic RSA-PSS and Rabin from RRep are recipient unforgeable.

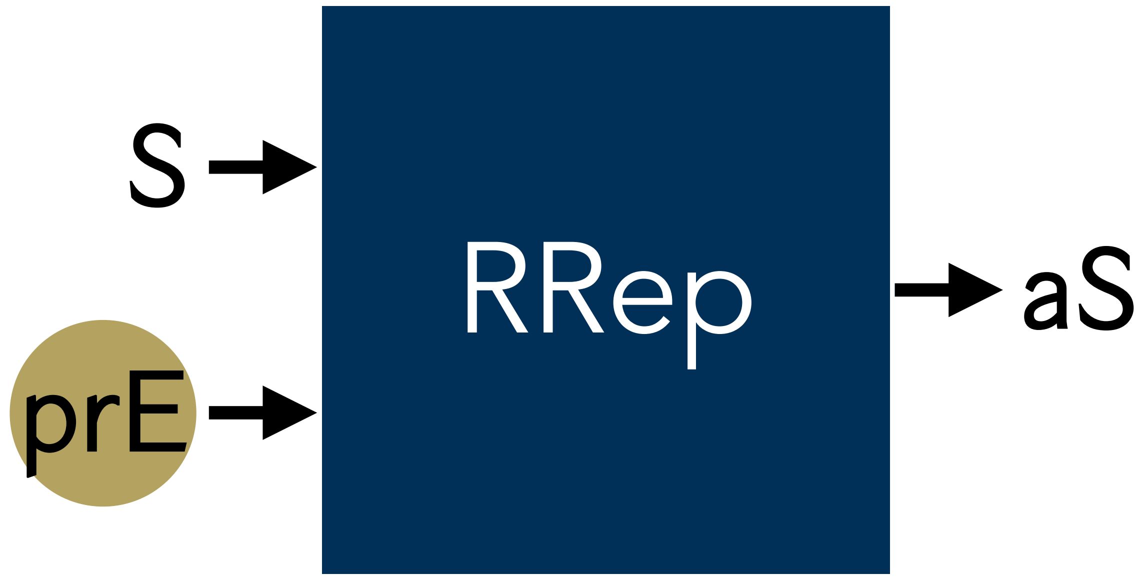
- We are unable to prove some signature schemes such as Boneh-Boyen are SUF-CRA secure — can we still achieve recipient-unforgeable schemes?

Repairing Recipient Unforgeability of RRep



Repairing Recipient Unforgeability of RRep

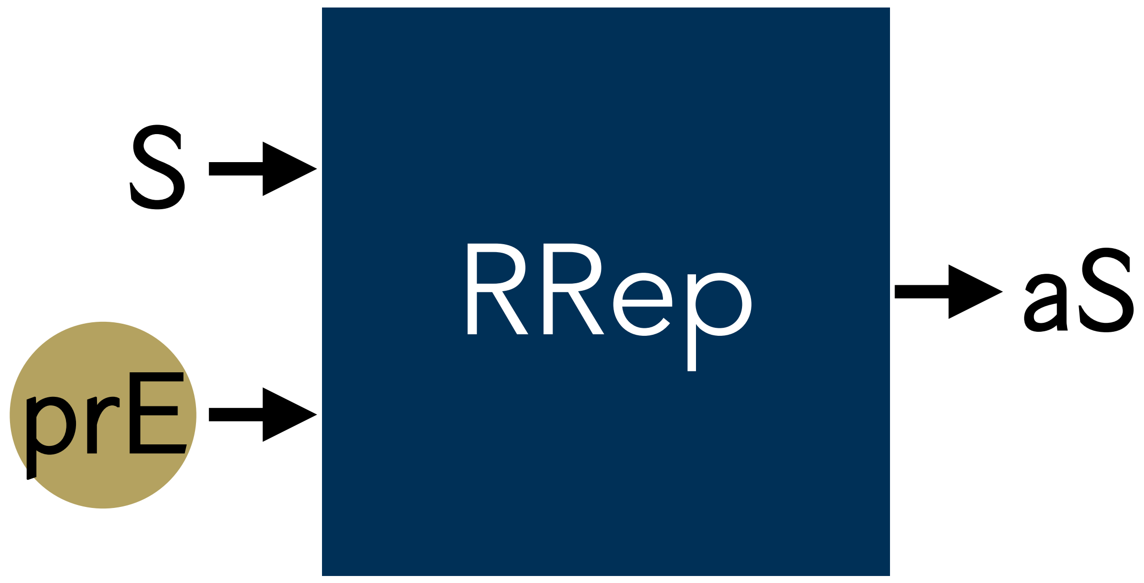
- Achieve recipient unforgeability by requiring stronger property on pseudorandom encryption scheme prE.



Repairing Recipient Unforgeability of RRep

- Achieve recipient unforgeability by requiring stronger property on pseudorandom encryption scheme prE.

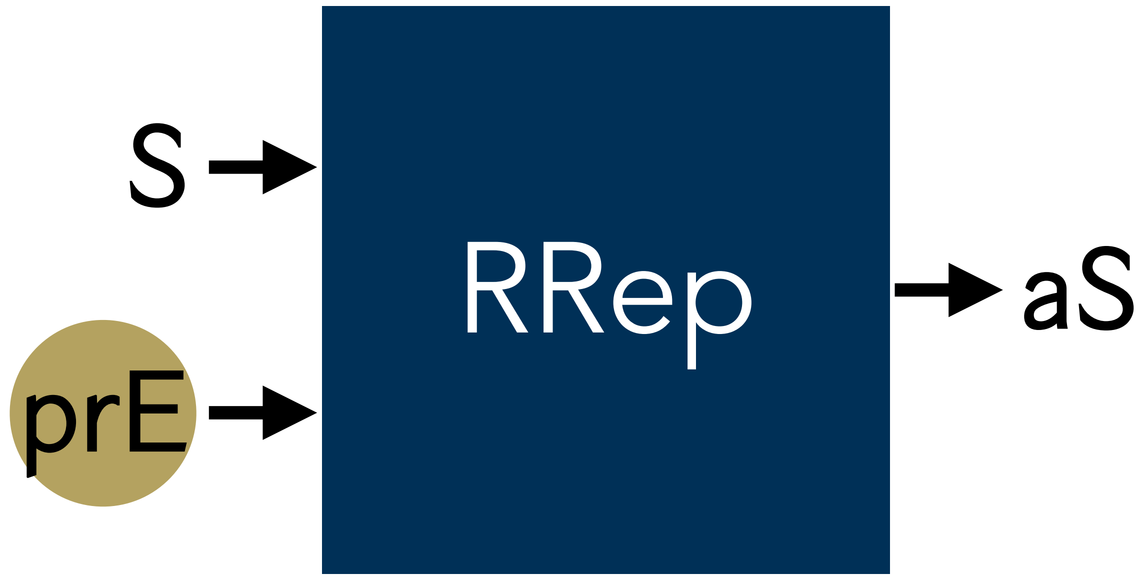
- Simulatability with random ciphertexts (SIM-$CT) asks that any adversary cannot distinguish between ciphertexts and random samples even knowing the symmetric key.



Repairing Recipient Unforgeability of RRep

- Achieve recipient unforgeability by requiring stronger property on pseudorandom encryption scheme prE.

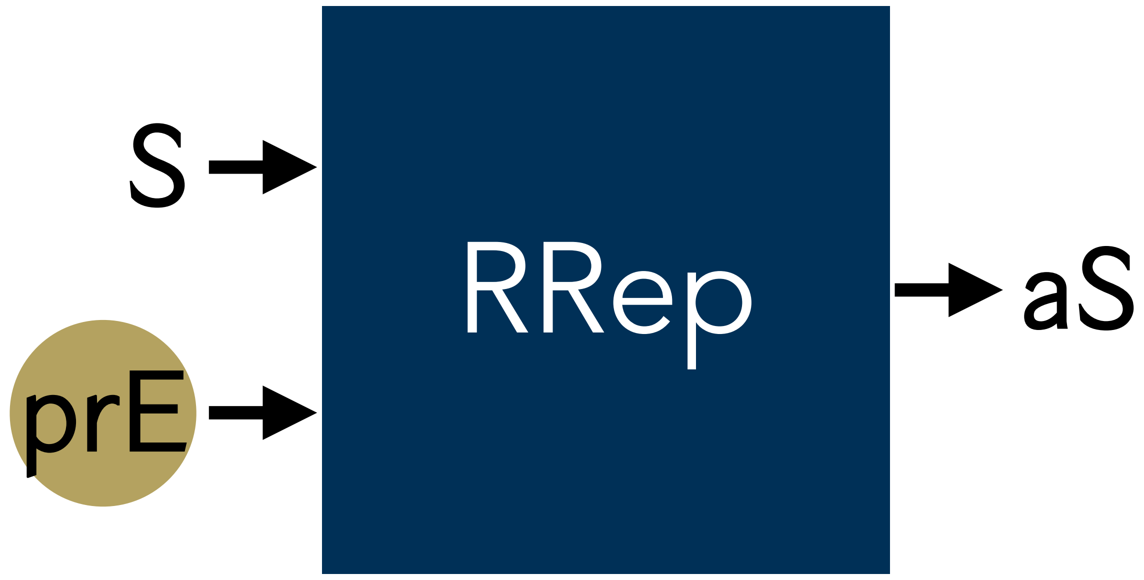
- Simulatability with random ciphertexts (SIM-$CT) asks that any adversary cannot distinguish between ciphertexts and random samples even knowing the symmetric key.



  - ***How is this possible?*** We can leverage ideal models to make decryption of random samples consistent with a fixed key.

Repairing Recipient Unforgeability of RRep

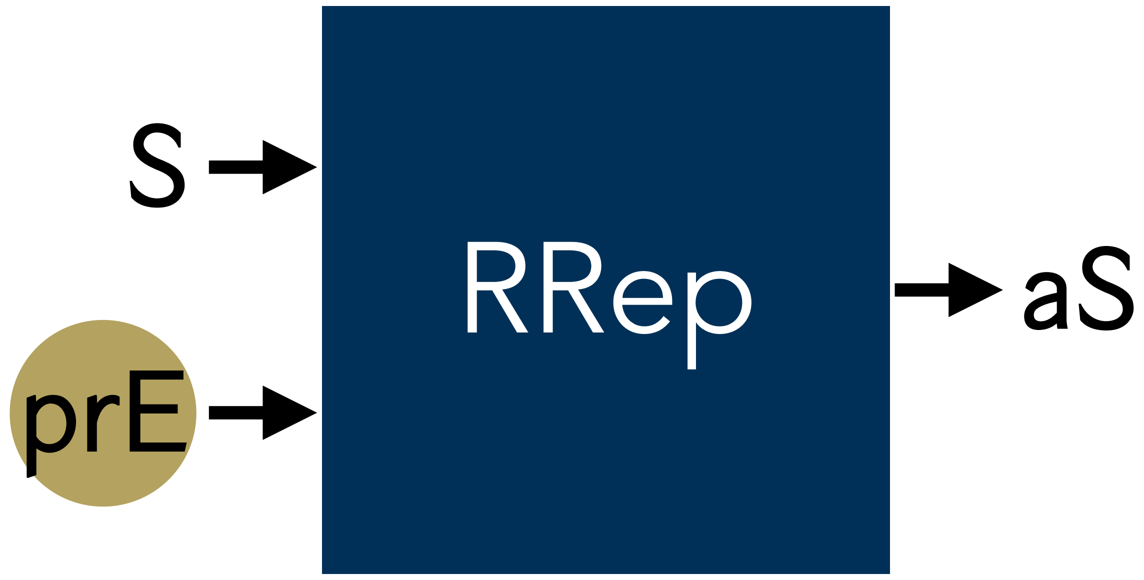
- Achieve recipient unforgeability by requiring stronger property on pseudorandom encryption scheme prE.



- Simulatability with random ciphertexts (SIM-$CT) asks that any adversary cannot distinguish between ciphertexts and random samples even knowing the symmetric key.
  - ***How is this possible?*** We can leverage ideal models to make decryption of random samples consistent with a fixed key.
- Definition is modular — construction can build on a variety of ideal primitives (e.g. random oracle, ideal cipher) — and composable.

Repairing Recipient Unforgeability of RRep

- Achieve recipient unforgeability by requiring stronger property on pseudorandom encryption scheme **prE**.



- Simulatability with random ciphertexts (SIM-$CT) asks that any adversary cannot distinguish between ciphertexts and random samples even knowing the symmetric key.
  - ***How is this possible?*** We can leverage ideal models to make decryption of random samples consistent with a fixed key.
- Definition is modular — construction can build on a variety of ideal primitives (e.g. random oracle, ideal cipher) — and composable.
- Achieved by randomized block cipher modes.

Conclusion

Summary

	Robustness	Private Anamorphism	
1	Observe a gap between a stated goal of robustness and its formalization.	Observe a gap between the deployment scenario of private anamorphism and its formalization.	5
2	Propose **Dictator Unforgeability.**	Propose **Recipient Unforgeability.**	6
3	Mount a practical attack a previously proposed *robust* anamorphic signature scheme.	Mount a practical attack a natural *private* anamorphic signature scheme.	7
4	Repair other prior anamorphic transforms to achieve dictator unforgeability.	Repair (in two ways) a prior anamorphic transform to achieve recipient unforgeability.	8

Summary

	Robustness	Private Anamorphism	
1	Observe a gap between a stated goal of robustness and its formalization.	Observe a gap between the deployment scenario of private anamorphism and its formalization.	5
2	Propose **Dictator Unforgeability.**	Propose **Recipient Unforgeability.**	6
3	Mount a practical attack a previously proposed *robust* anamorphic signature scheme.	Mount a practical attack a natural *private* anamorphic signature scheme.	7
4	Repair other prior anamorphic transforms to achieve dictator unforgeability.	Repair (in two ways) a prior anamorphic transform to achieve recipient unforgeability.	8

Summary

	Robustness	Private Anamorphism	
1	Observe a gap between a stated goal of robustness and its formalization.	Observe a gap between the deployment scenario of private anamorphism and its formalization.	5
2	Propose **Dictator Unforgeability.**	Propose **Recipient Unforgeability.**	6
3	Mount a practical attack a previously proposed *robust* anamorphic signature scheme.	Mount a practical attack a natural *private* anamorphic signature scheme.	7
4	Repair other prior anamorphic transforms to achieve dictator unforgeability.	Repair (in two ways) a prior anamorphic transform to achieve recipient unforgeability.	8

Summary

Robustness

1 Observe a gap between a stated goal of robustness and its formalization.

2 Propose **Dictator Unforgeability.**

3 Mount a practical attack a previously proposed *robust* anamorphic signature scheme.

4 Repair other prior anamorphic transforms to achieve dictator unforgeability.

Private Anamorphism

5 Observe a gap between the deployment scenario of private anamorphism and its formalization.

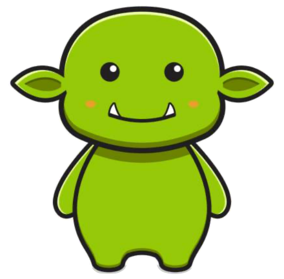
6 Propose **Recipient Unforgeability.**

7 Mount a practical attack a natural *private* anamorphic signature scheme.

8 Repair (in two ways) a prior anamorphic transform to achieve recipient unforgeability.

Summary

	Robustness	Private Anamorphism	
1	Observe a gap between a stated goal of robustness and its formalization.	Observe a gap between the deployment scenario of private anamorphism and its formalization.	5
2	Propose **Dictator Unforgeability.**	Propose **Recipient Unforgeability.**	6
3	Mount a practical attack a previously proposed *robust* anamorphic signature scheme.	Mount a practical attack a natural *private* anamorphic signature scheme.	7
4	Repair other prior anamorphic transforms to achieve dictator unforgeability.	Repair (in two ways) a prior anamorphic transform to achieve recipient unforgeability.	8



Vonholdt (https://www.deviantart.com/vonholdt)

Starry Artw (https://www.behance.net/starry_artw)
Aleksandar Savić (https://dribbble.com/almigor)