Leakage-Resilience Meets Incompressibility

Mahesh Sreekumar Rajasree
Post-Doctoral Fellow
Department of Computer Science & Engineering
IIT Delhi

(Joint work with Kaartik Bhushan (IITB), Rishab Goyal (UW-Madison), Venkata Koppula (IITD), Varun Narayanan (UCLA) and Manoj Prabhakaran (IITB))

(Joint work with Rishab Goyal (UW-Madison), Venkata Koppula (IITD) and Aman Verma (IITD))

Contents

Contents

Contents

Contents

Contents

Contents

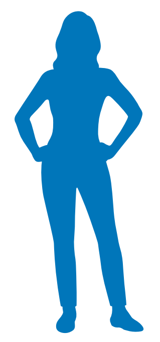
Contents

Contents

Introduction

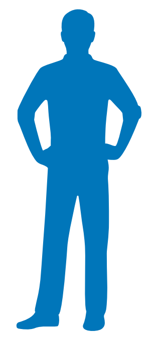
Encryption Scheme

Encryption Scheme



ALICE

Encryption Scheme



ALICE

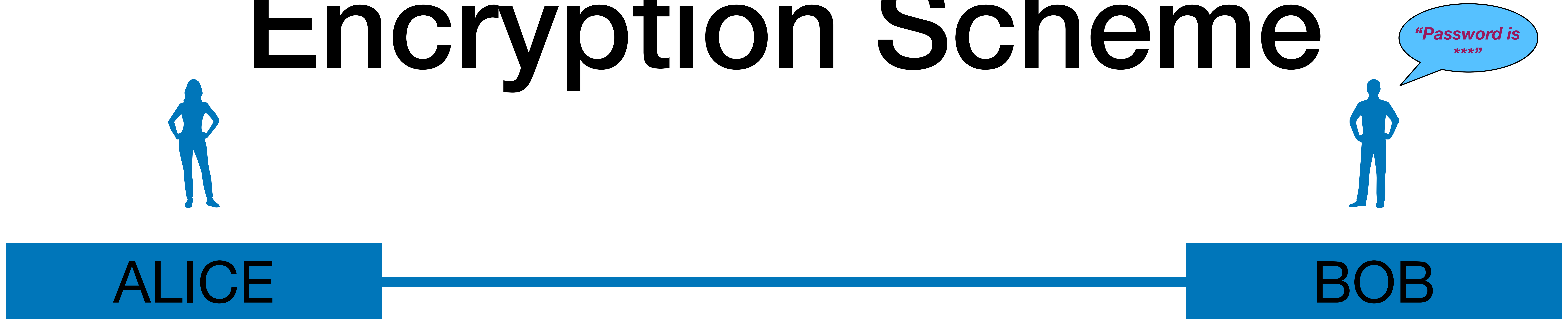
BOB

Encryption Scheme

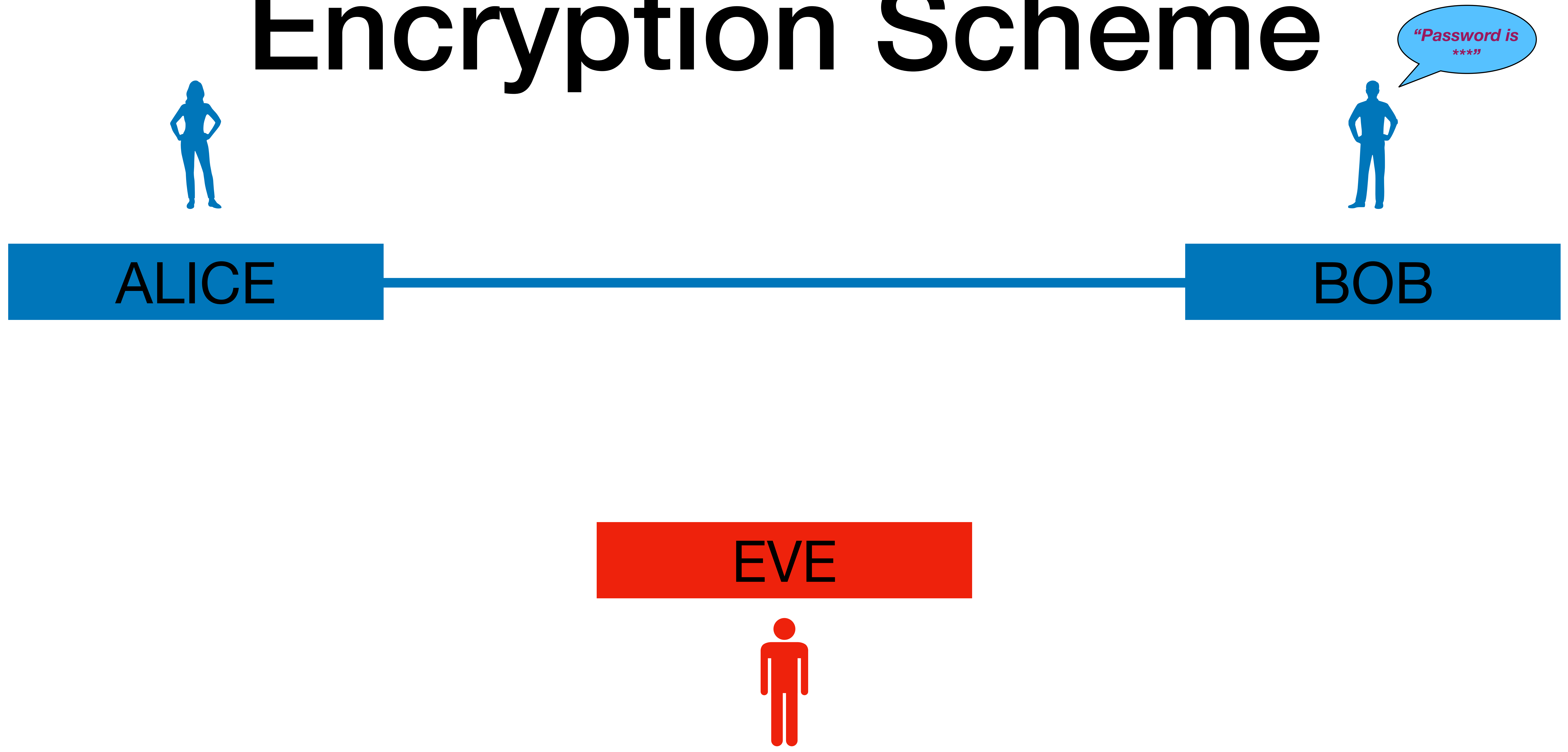
ALICE

BOB

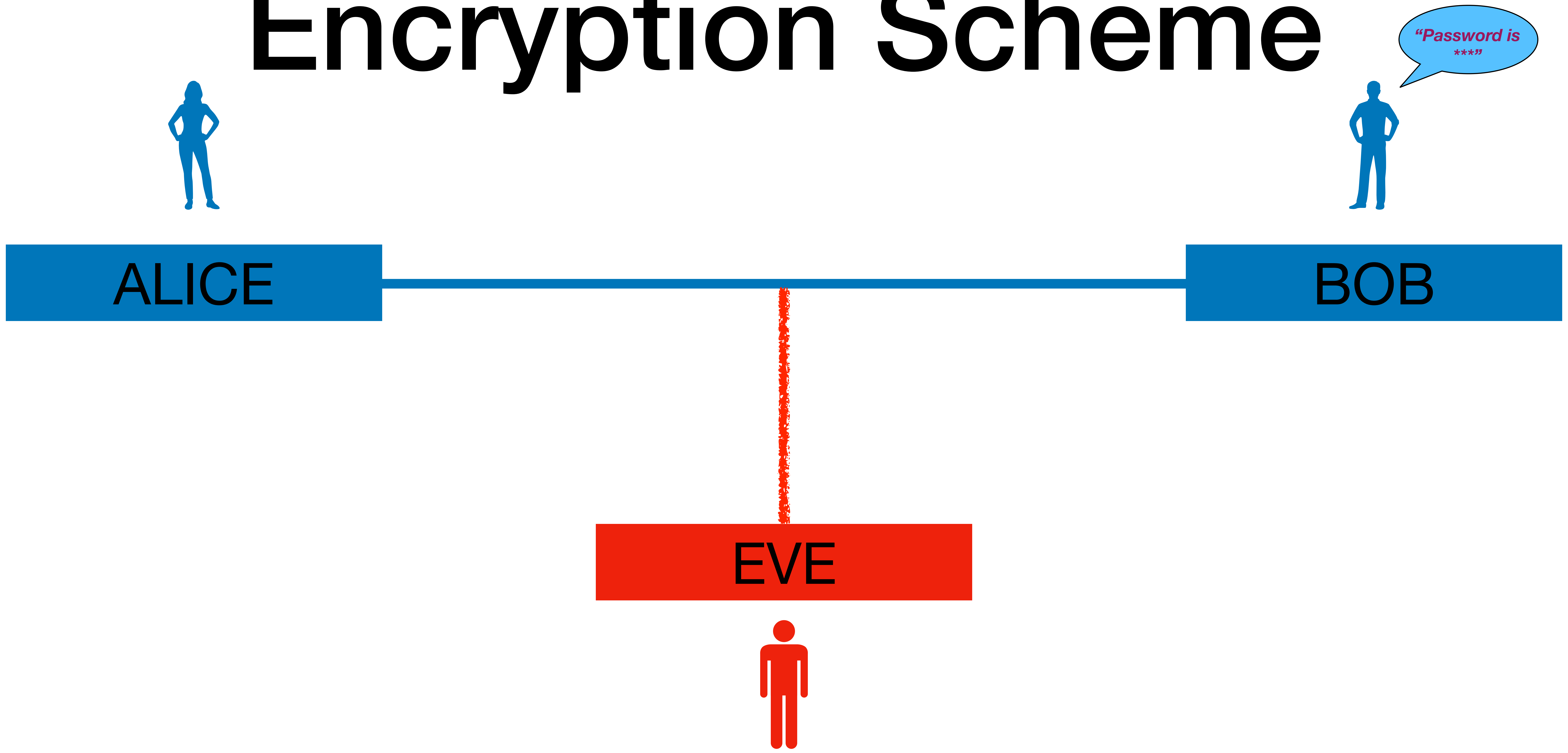
Encryption Scheme



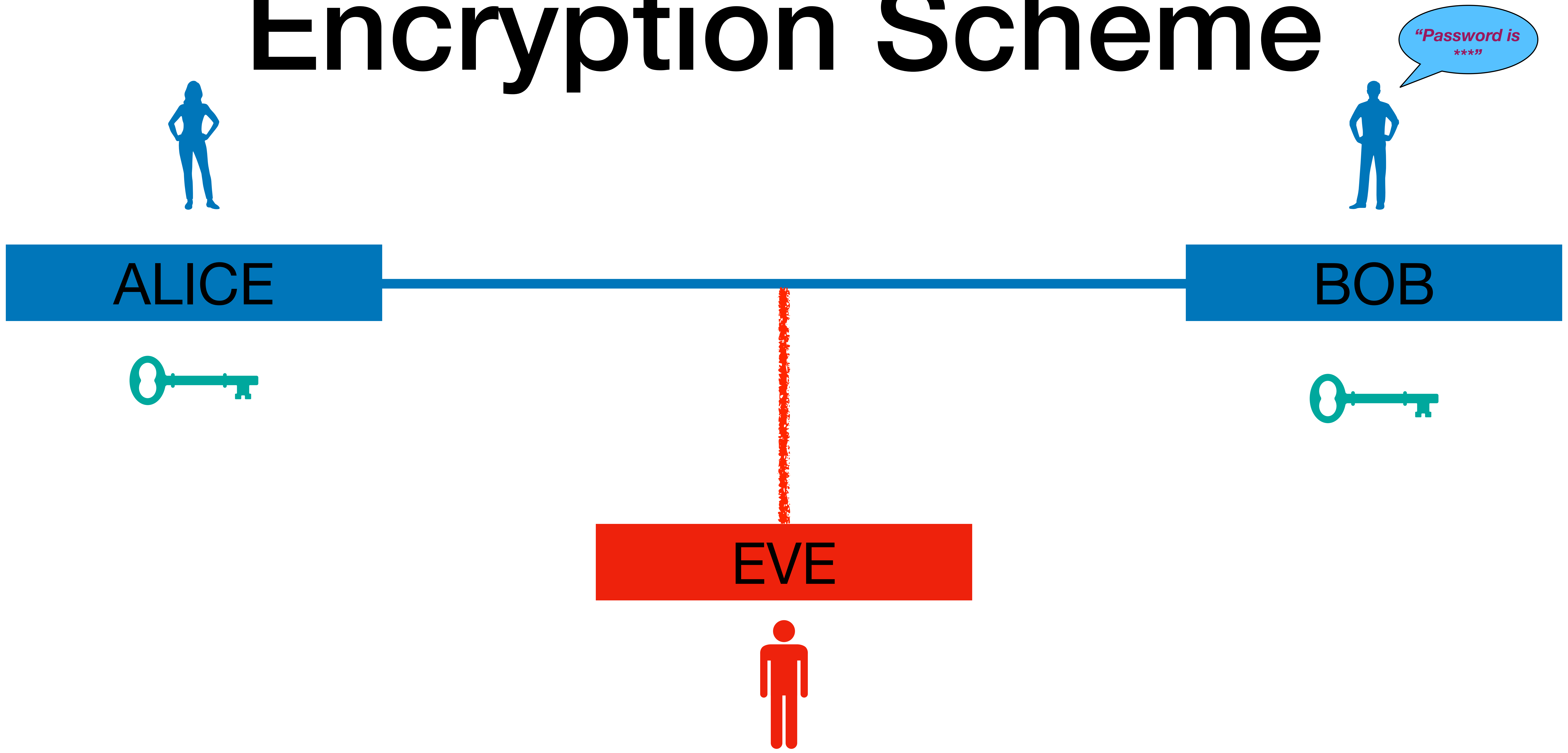
Encryption Scheme



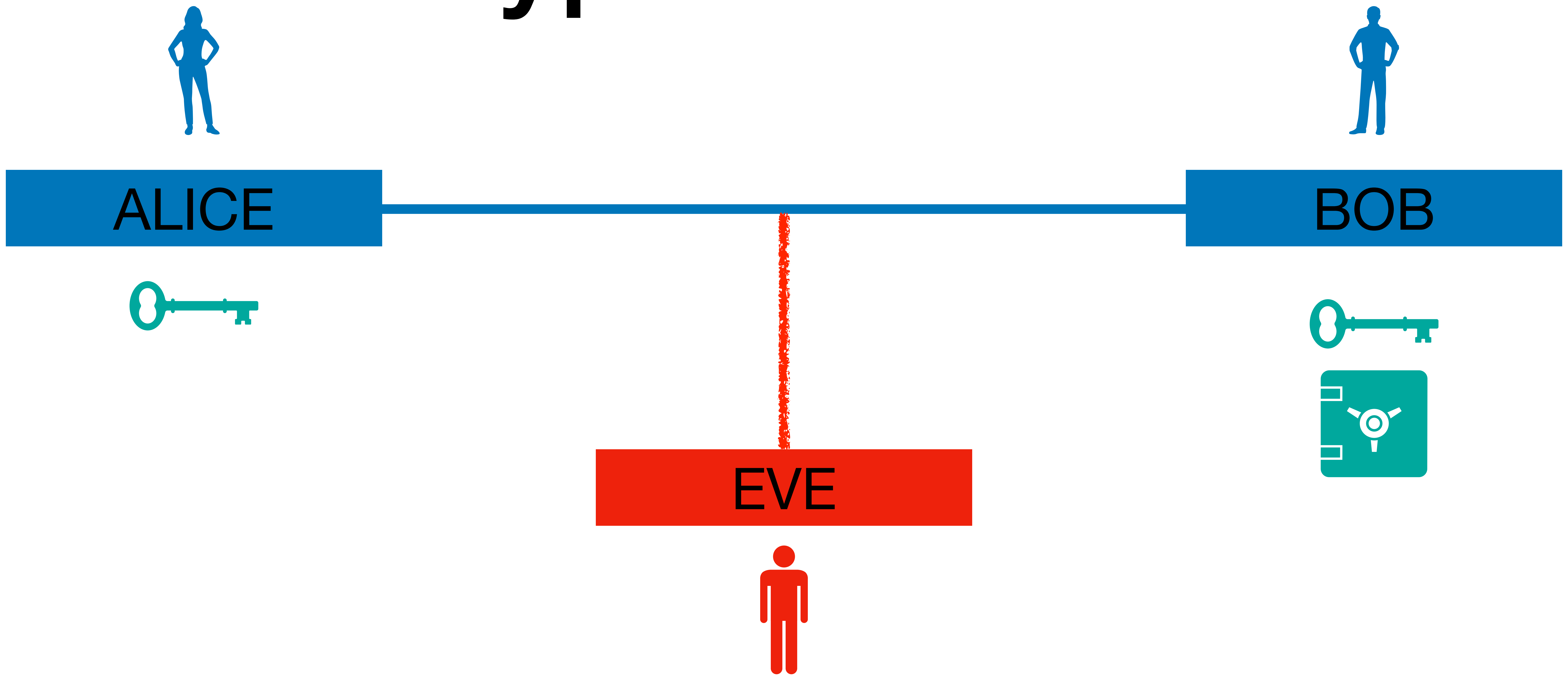
Encryption Scheme



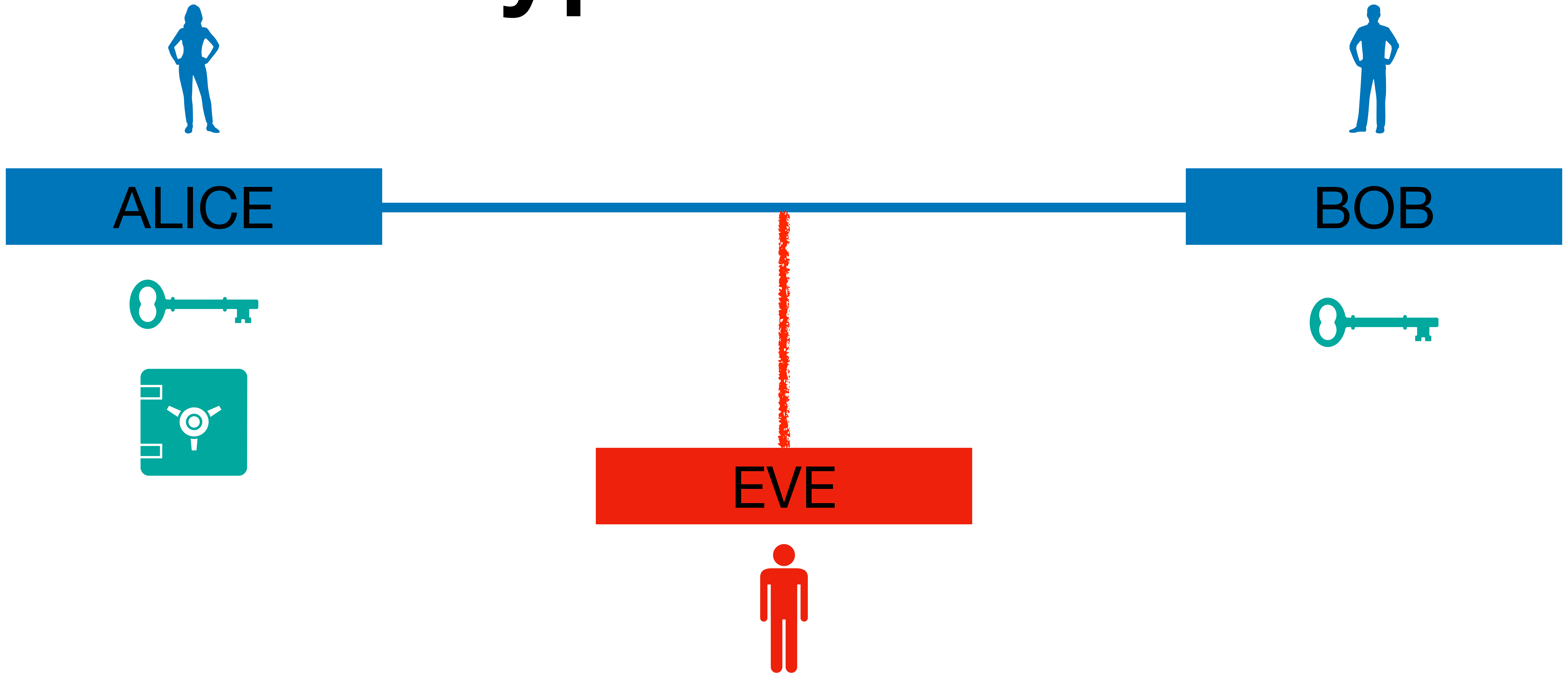
Encryption Scheme



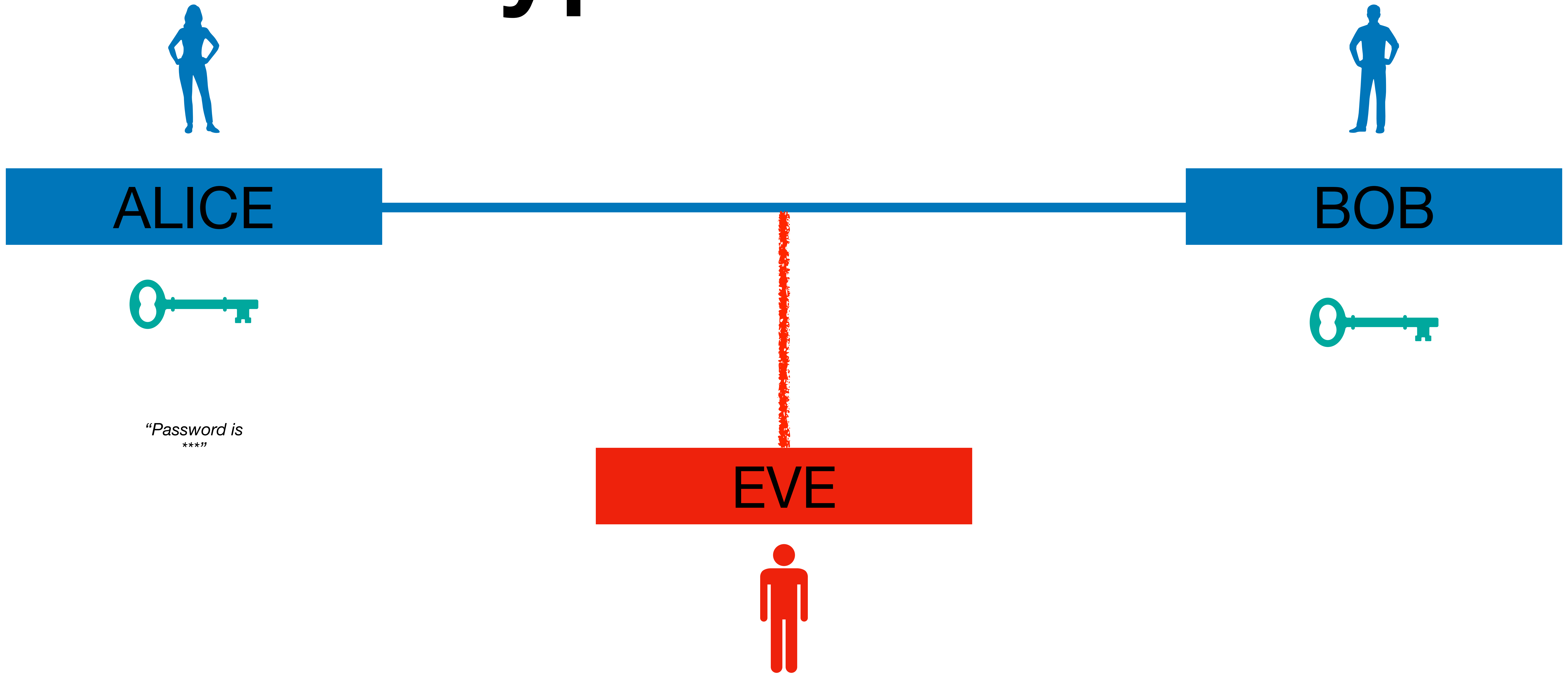
Encryption Scheme



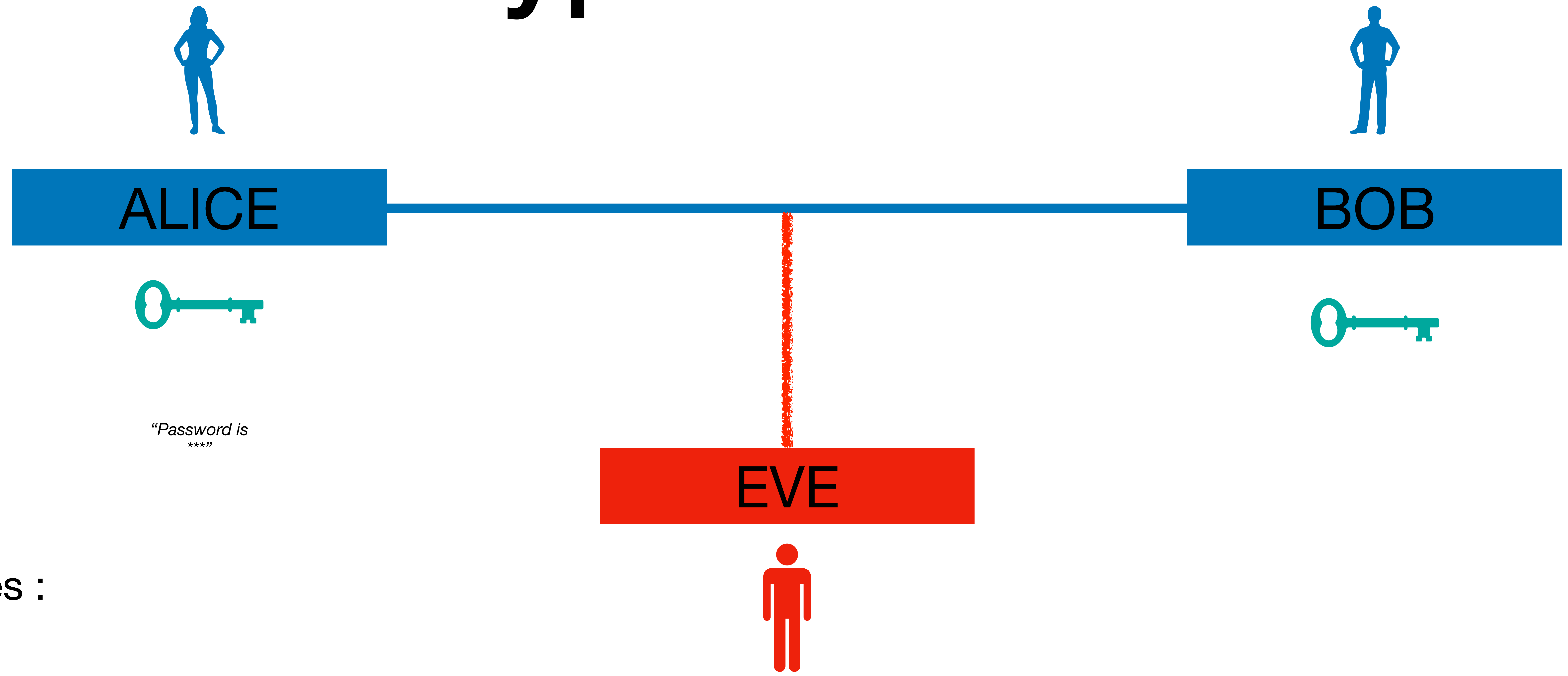
Encryption Scheme



Encryption Scheme

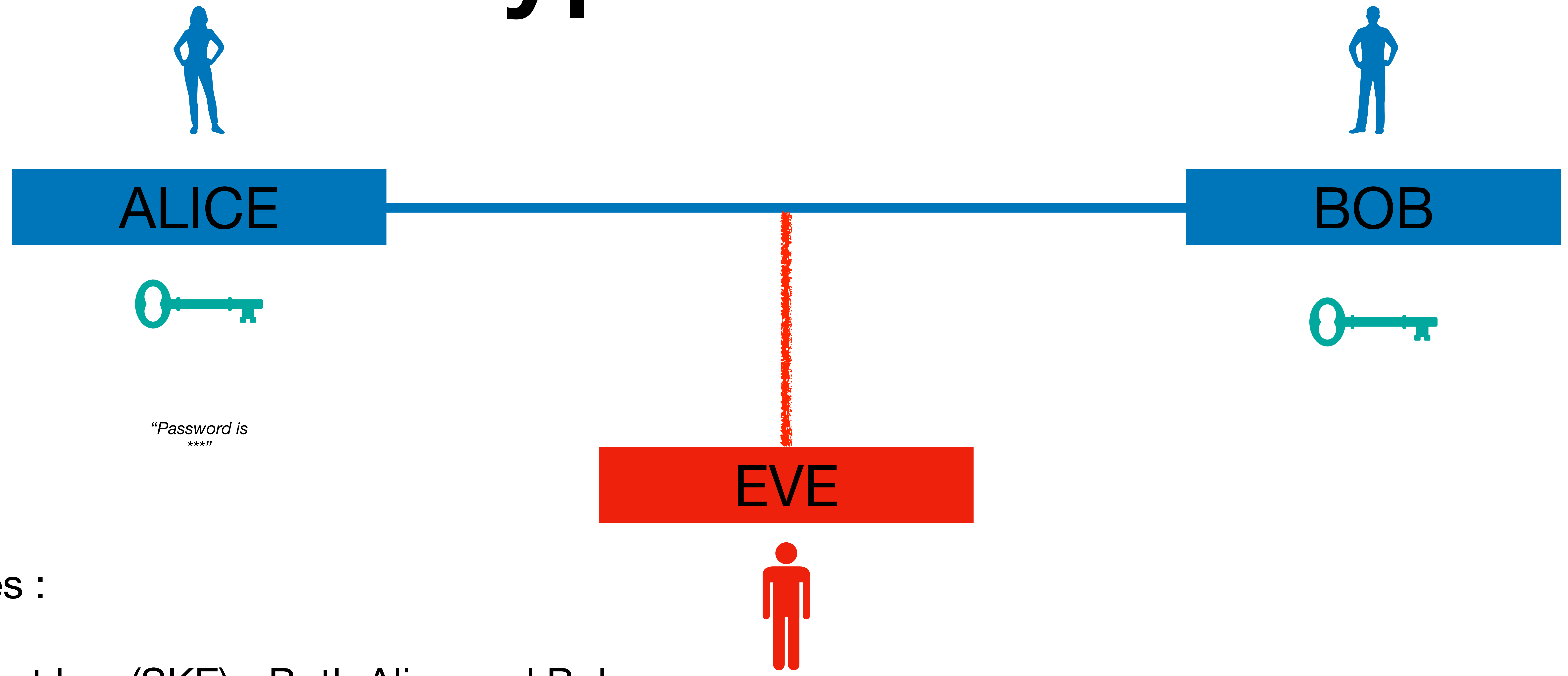


Encryption Scheme



2 types :

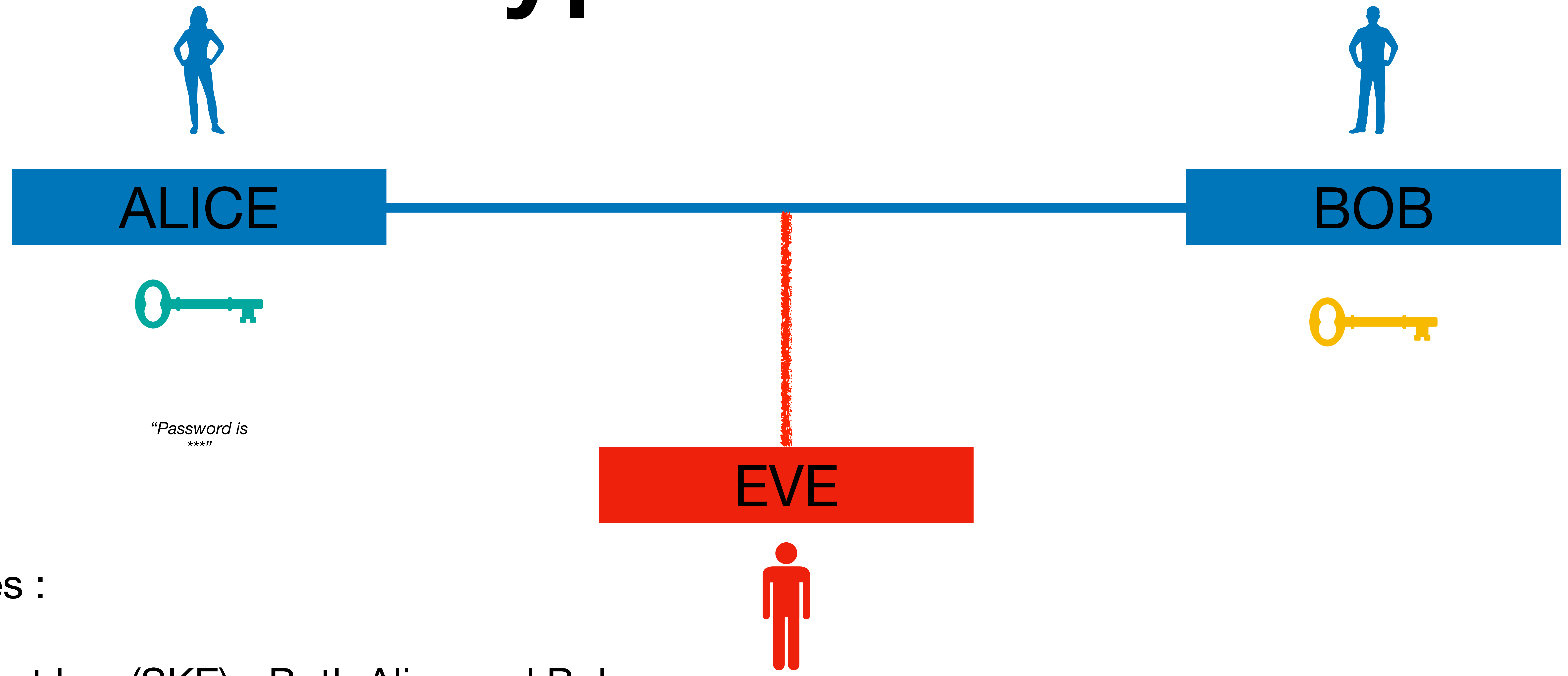
Encryption Scheme



2 types :

- secret key (SKE) - Both Alice and Bob have the same key.

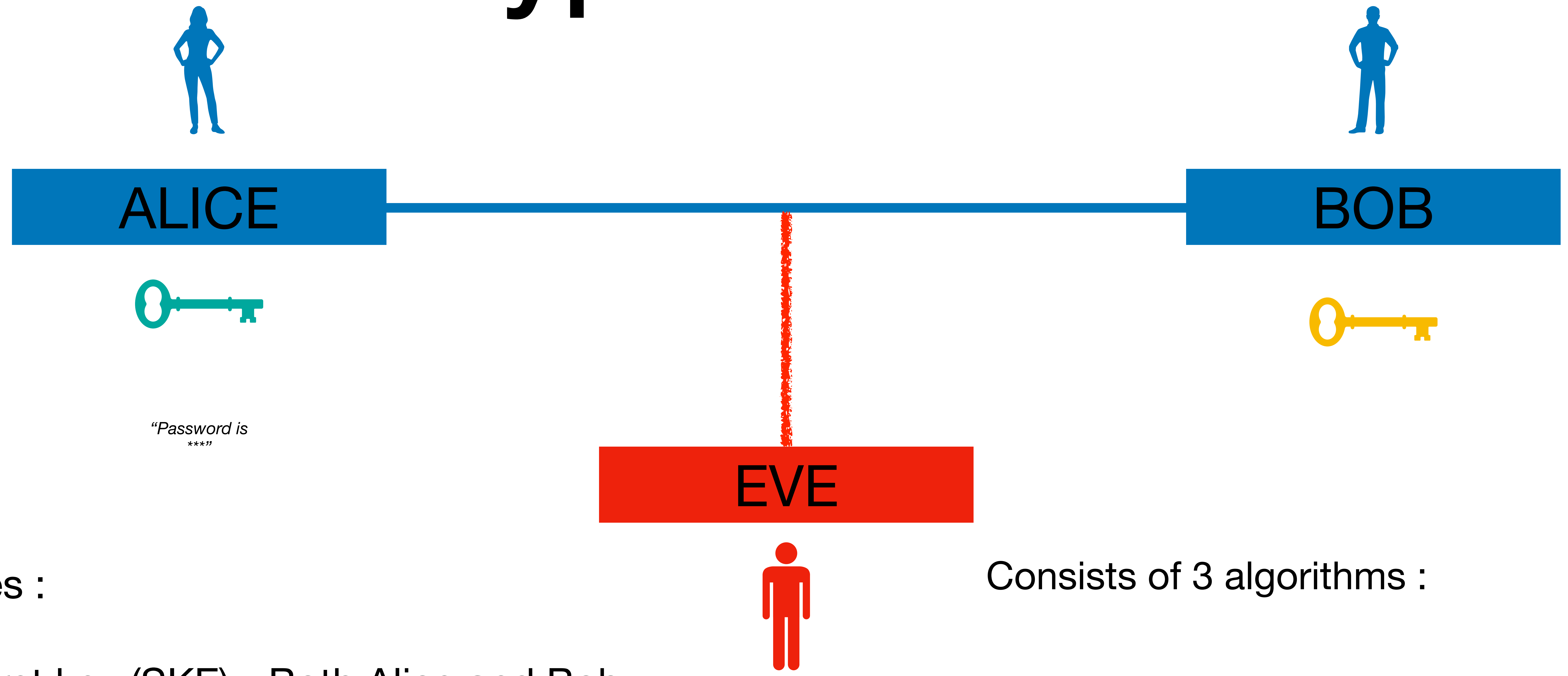
Encryption Scheme



2 types :

- secret key (SKE) - Both Alice and Bob have the same key.
- public key (PKE) - Encryptor has public key and decryption has secret key.

Encryption Scheme

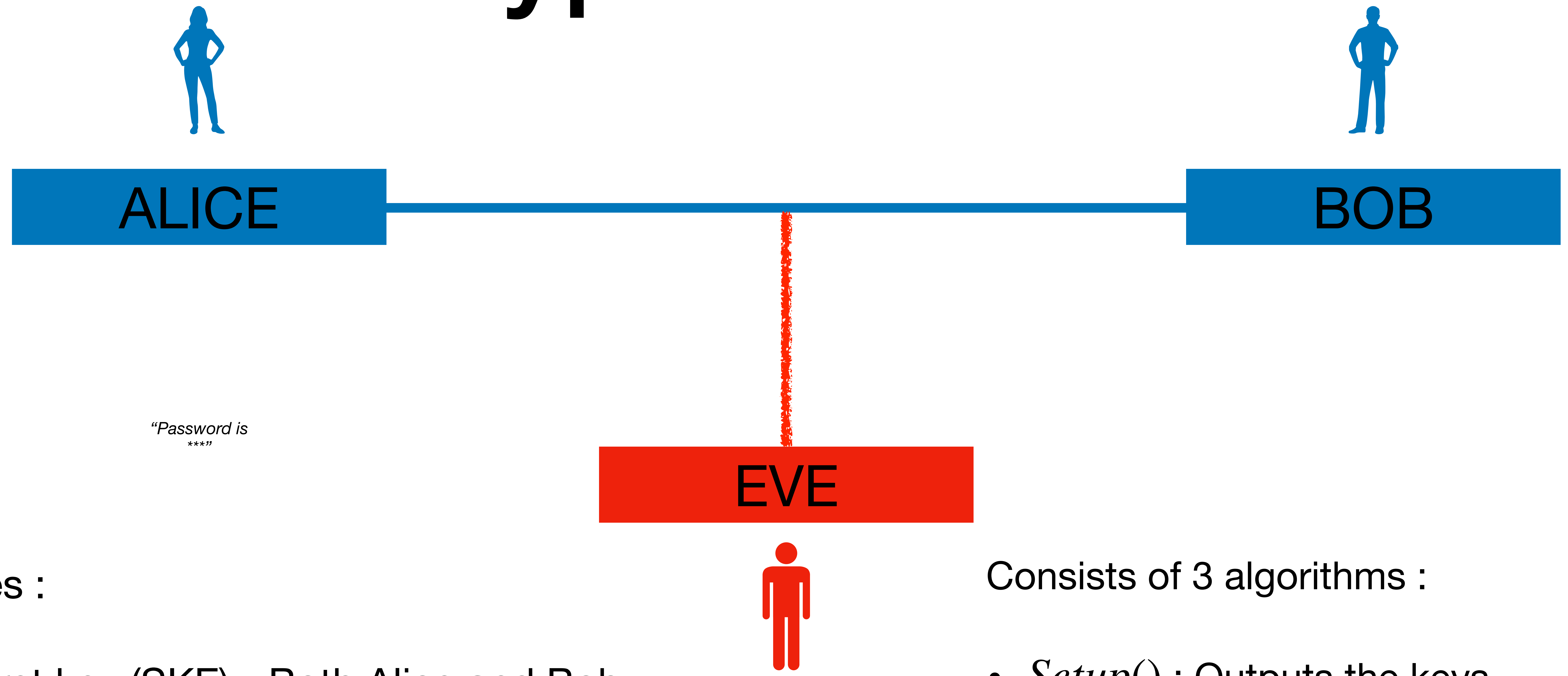


2 types :

- secret key (SKE) - Both Alice and Bob have the same key.
- public key (PKE) - Encryptor has public key and decryption has secret key.

Consists of 3 algorithms :

Encryption Scheme



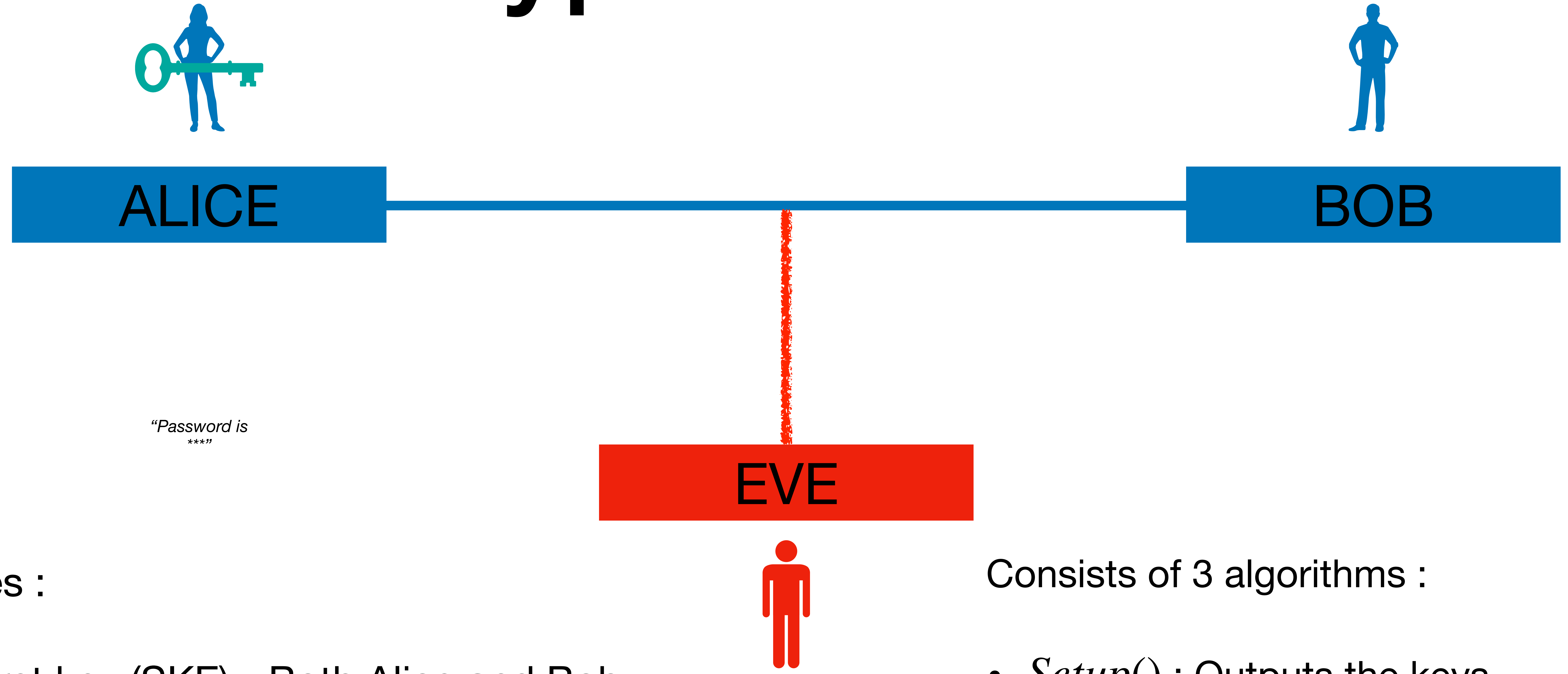
2 types :

- secret key (SKE) - Both Alice and Bob have the same key.
- public key (PKE) - Encryptor has public key and decryption has secret key.

Consists of 3 algorithms :

- $Setup()$: Outputs the keys

Encryption Scheme



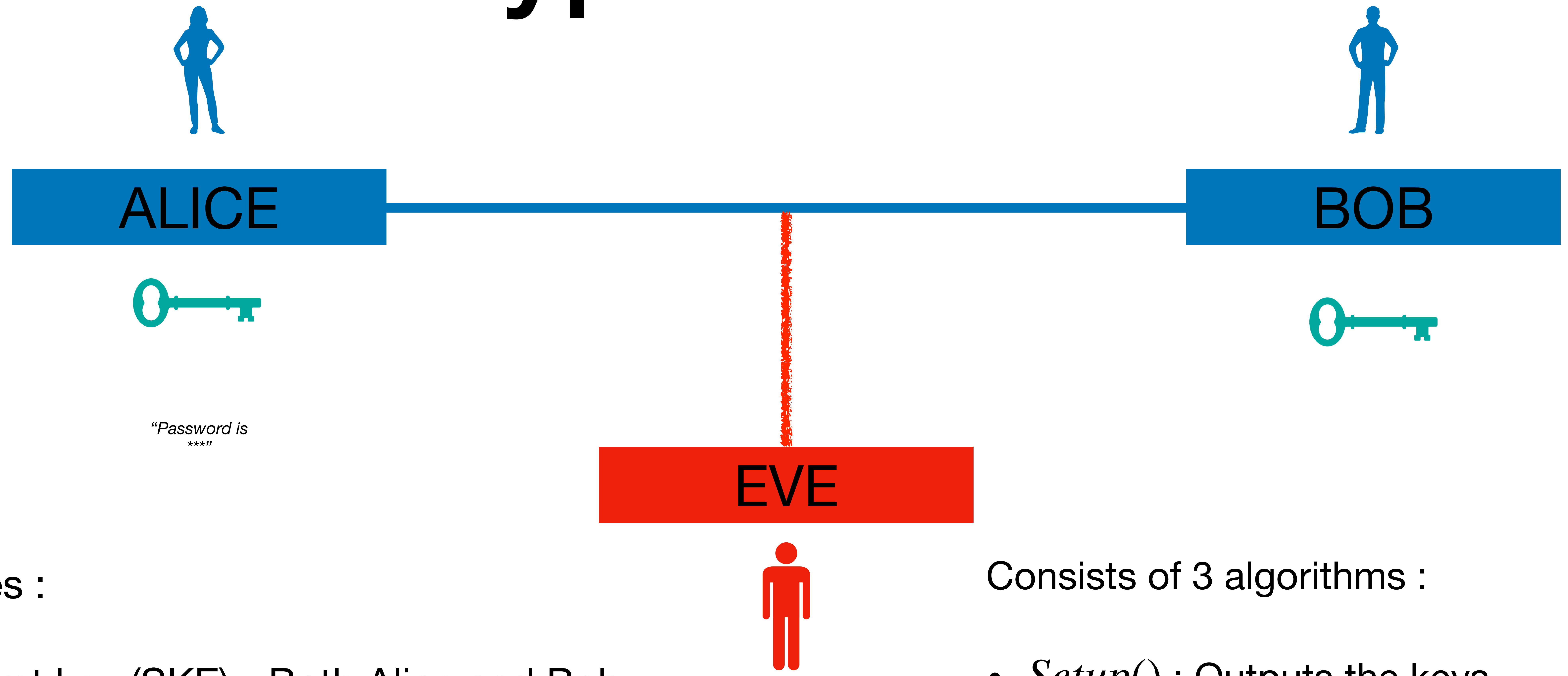
2 types :

- secret key (SKE) - Both Alice and Bob have the same key.
- public key (PKE) - Encryptor has public key and decryption has secret key.

Consists of 3 algorithms :

- $Setup()$: Outputs the keys

Encryption Scheme



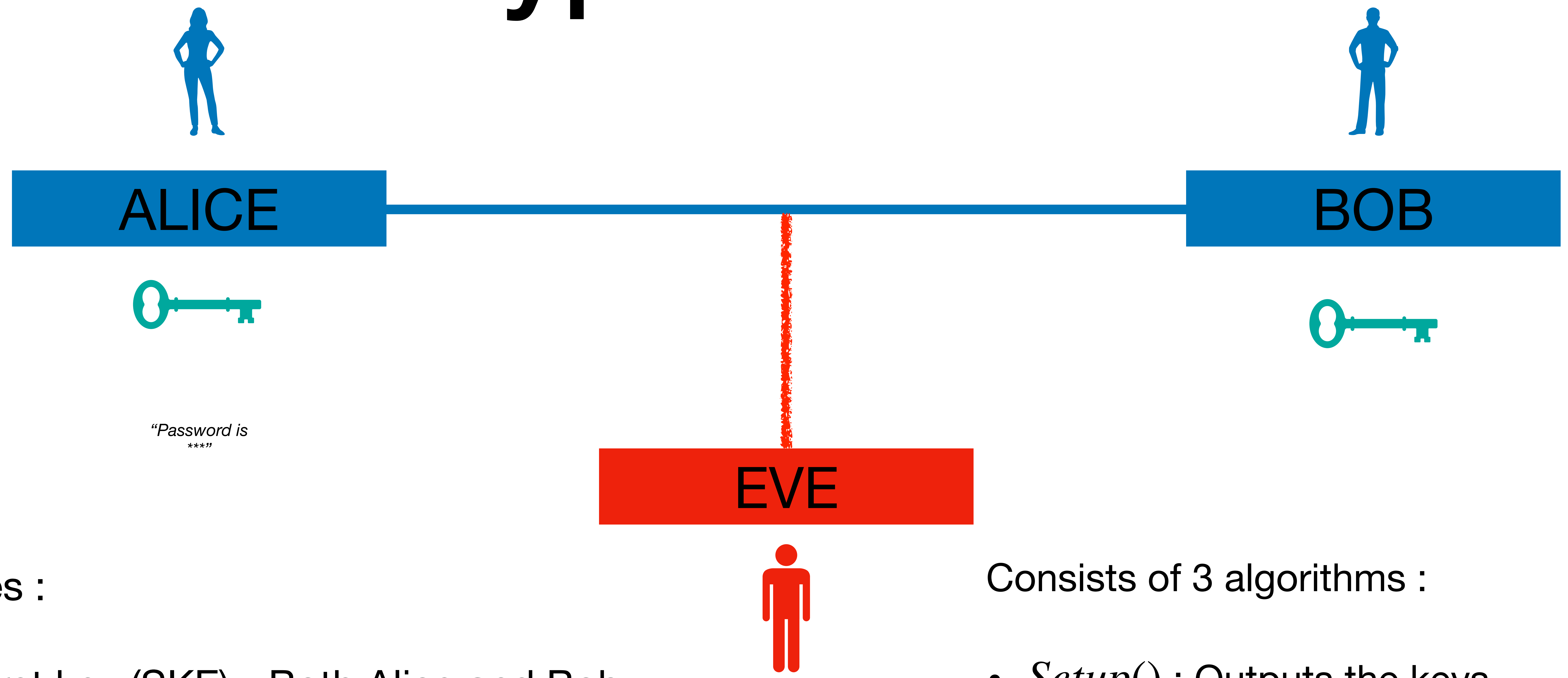
2 types :

- secret key (SKE) - Both Alice and Bob have the same key.
- public key (PKE) - Encryptor has public key and decryption has secret key.

Consists of 3 algorithms :

- $Setup()$: Outputs the keys

Encryption Scheme



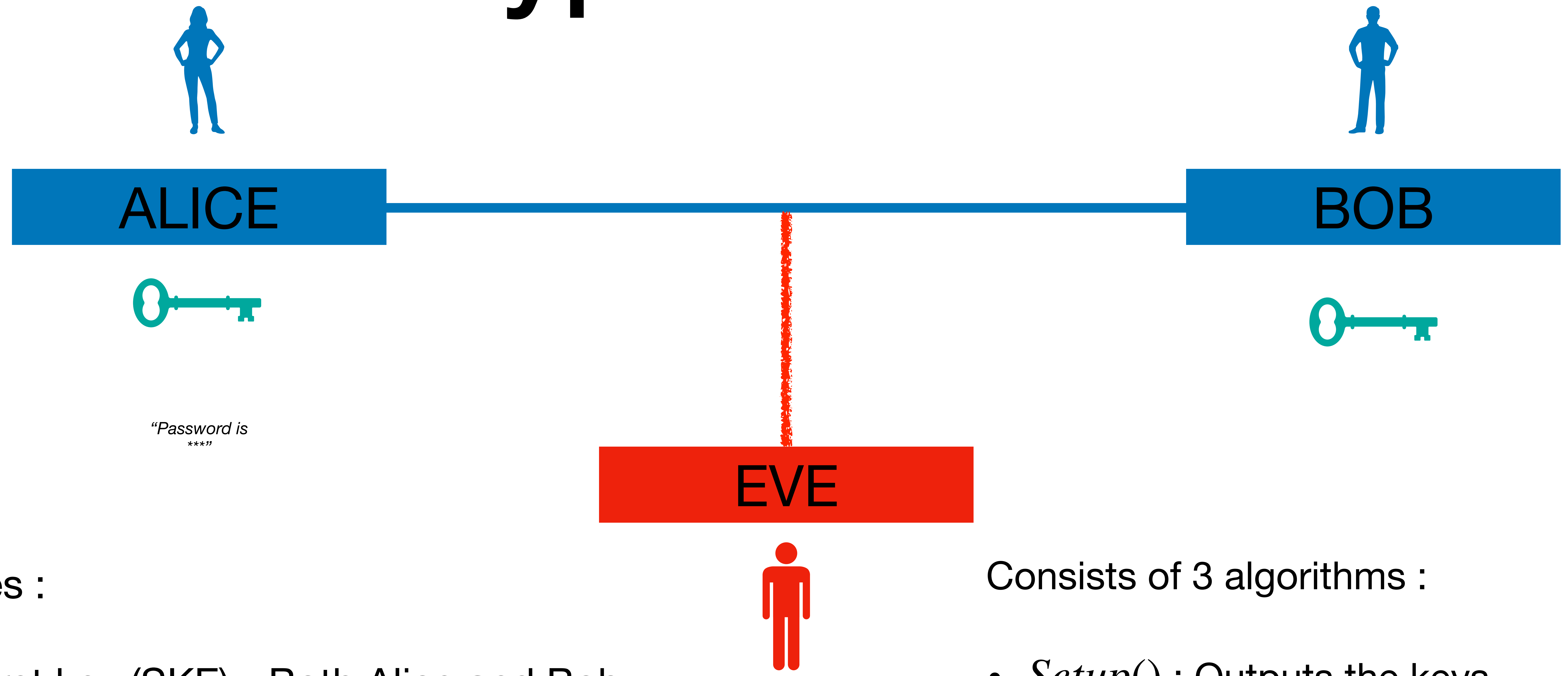
2 types :

- secret key (SKE) - Both Alice and Bob have the same key.
- public key (PKE) - Encryptor has public key and decryption has secret key.

Consists of 3 algorithms :

- $Setup()$: Outputs the keys
- $Enc(pk/sk, m)$: Outputs ciphertext

Encryption Scheme



2 types :

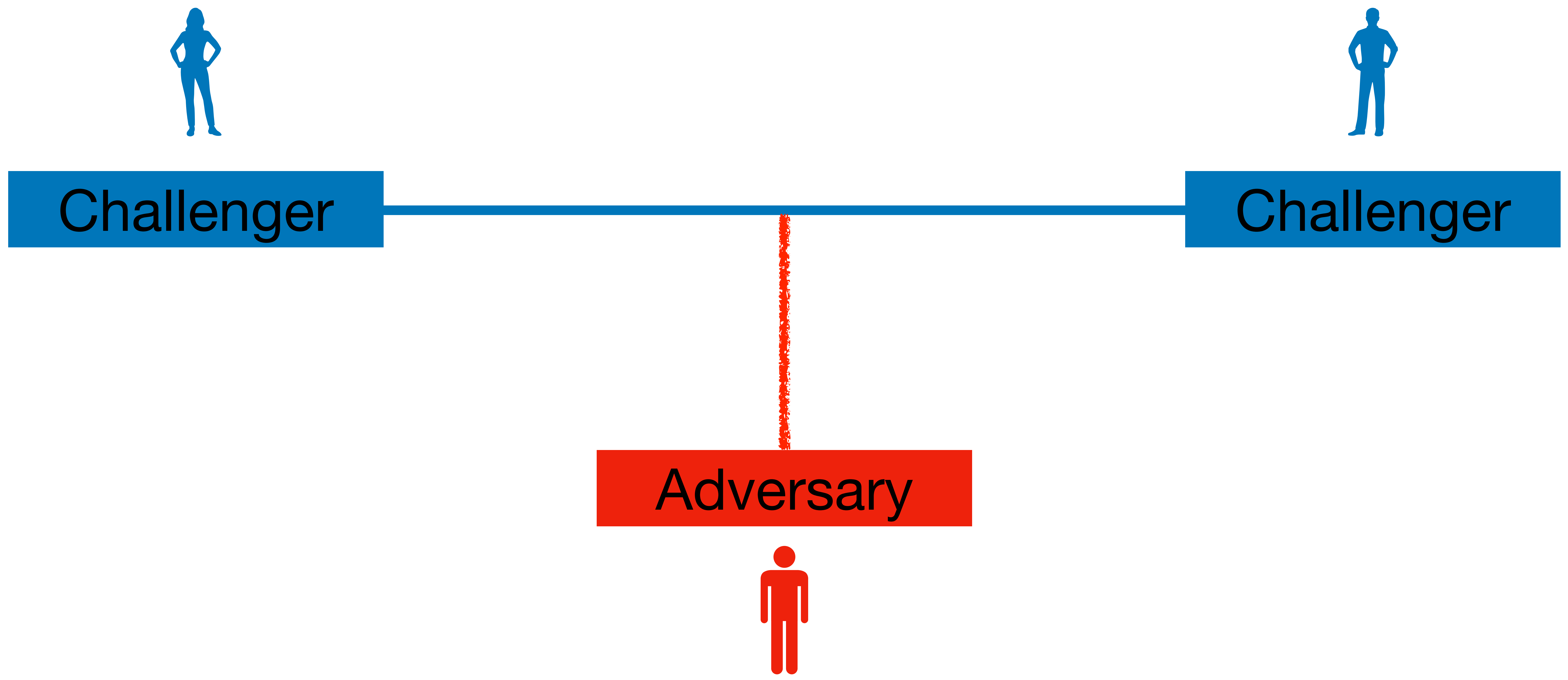
- secret key (SKE) - Both Alice and Bob have the same key.
- public key (PKE) - Encryptor has public key and decryption has secret key.

Consists of 3 algorithms :

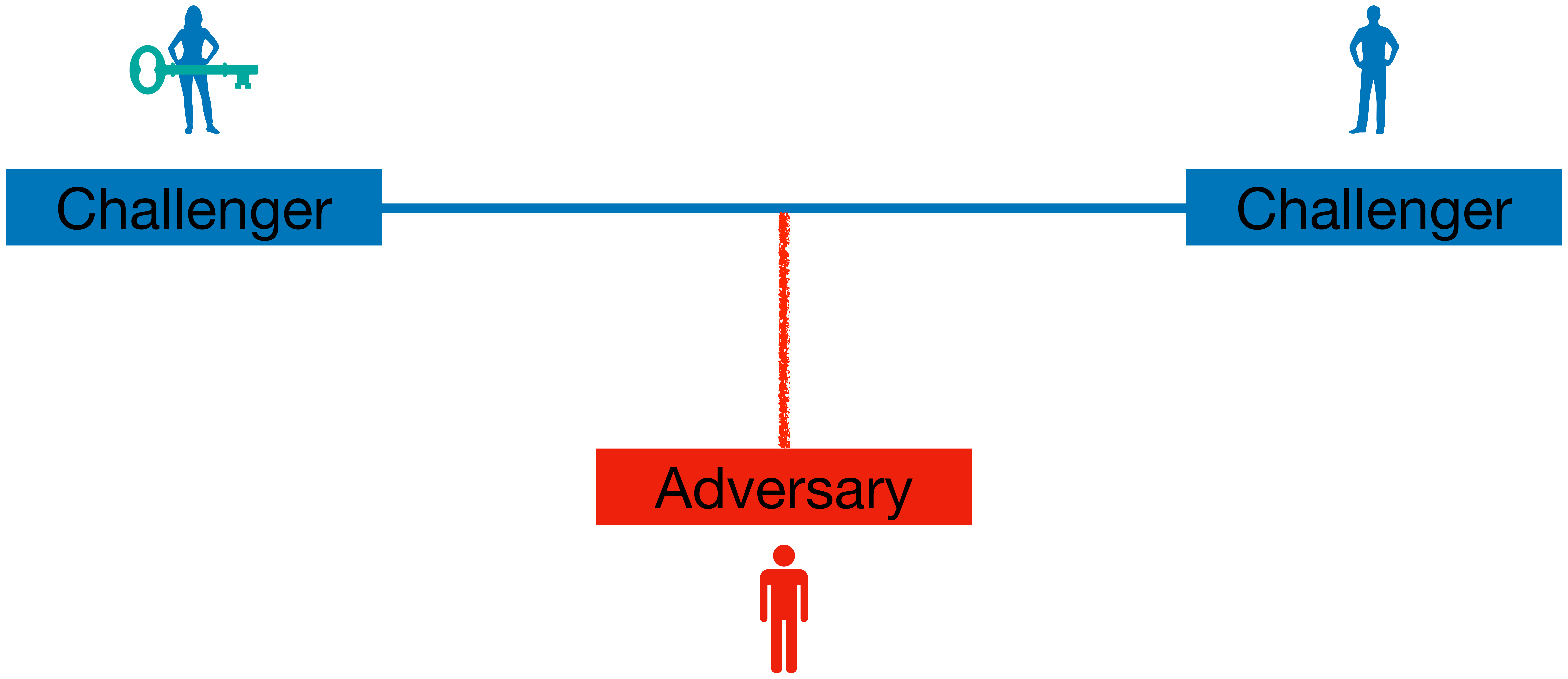
- $Setup()$: Outputs the keys
- $Enc(pk/sk, m)$: Outputs ciphertext
- $Dec(sk, c)$: Outputs message or error

Security Definitions

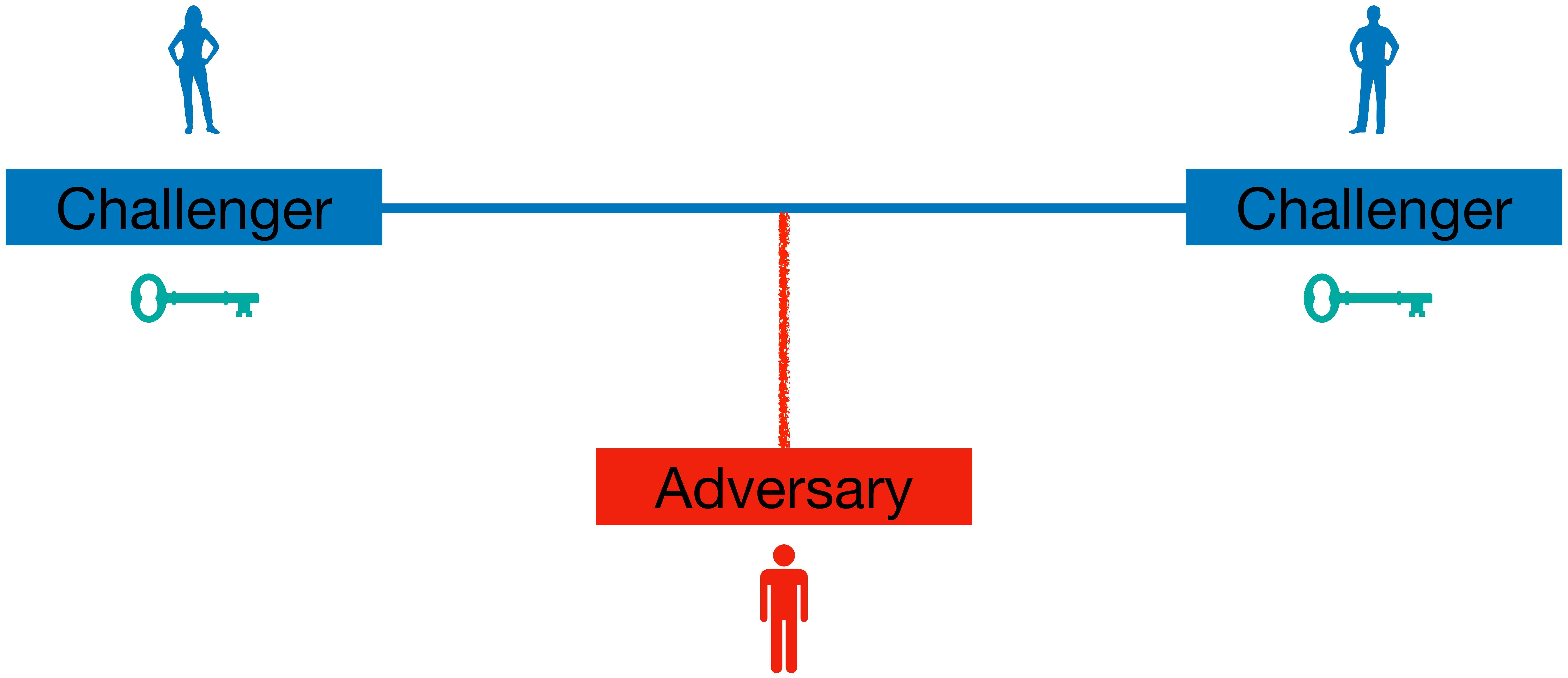
Standard Security [Goldwaser,Micali84]



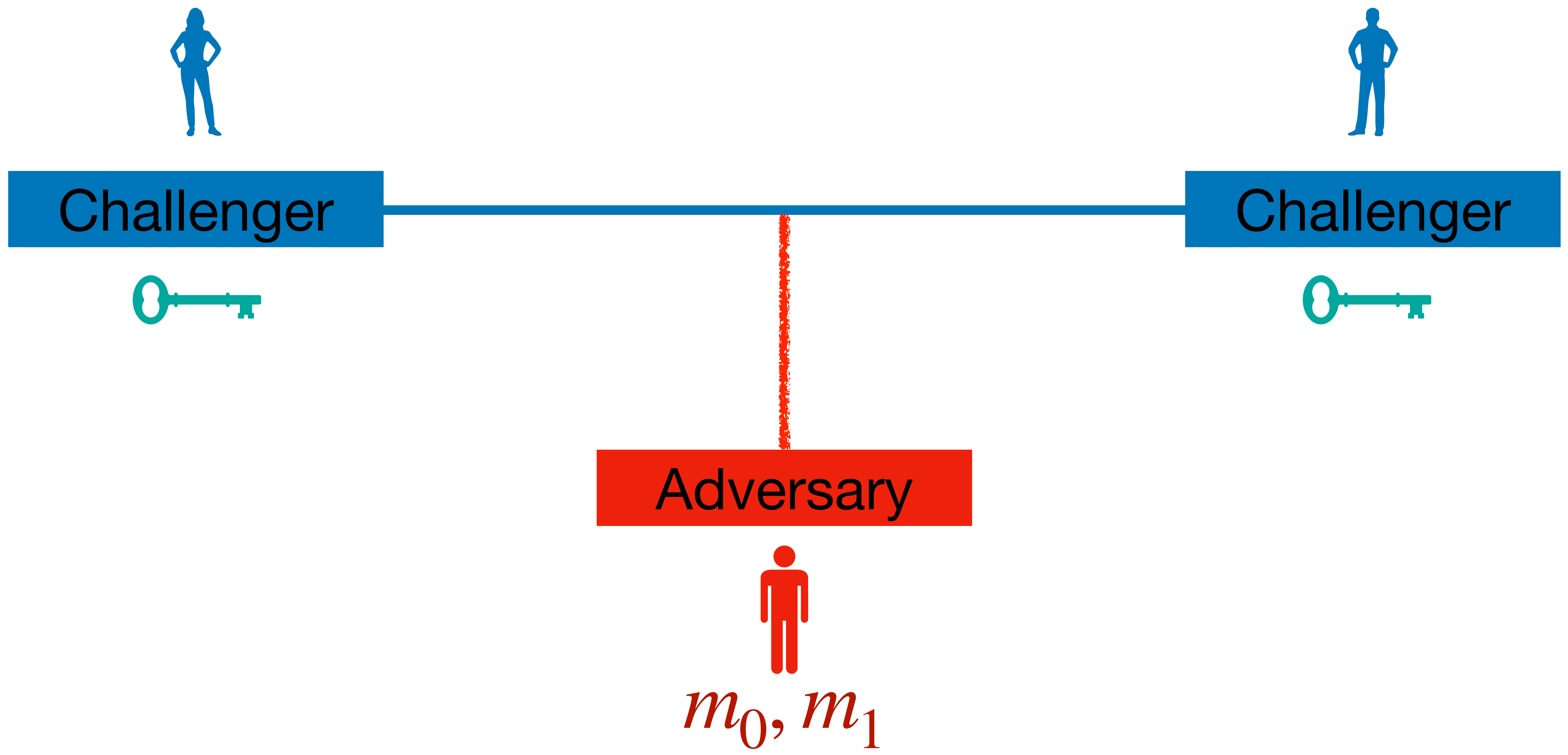
Standard Security [Goldwaser,Micali84]



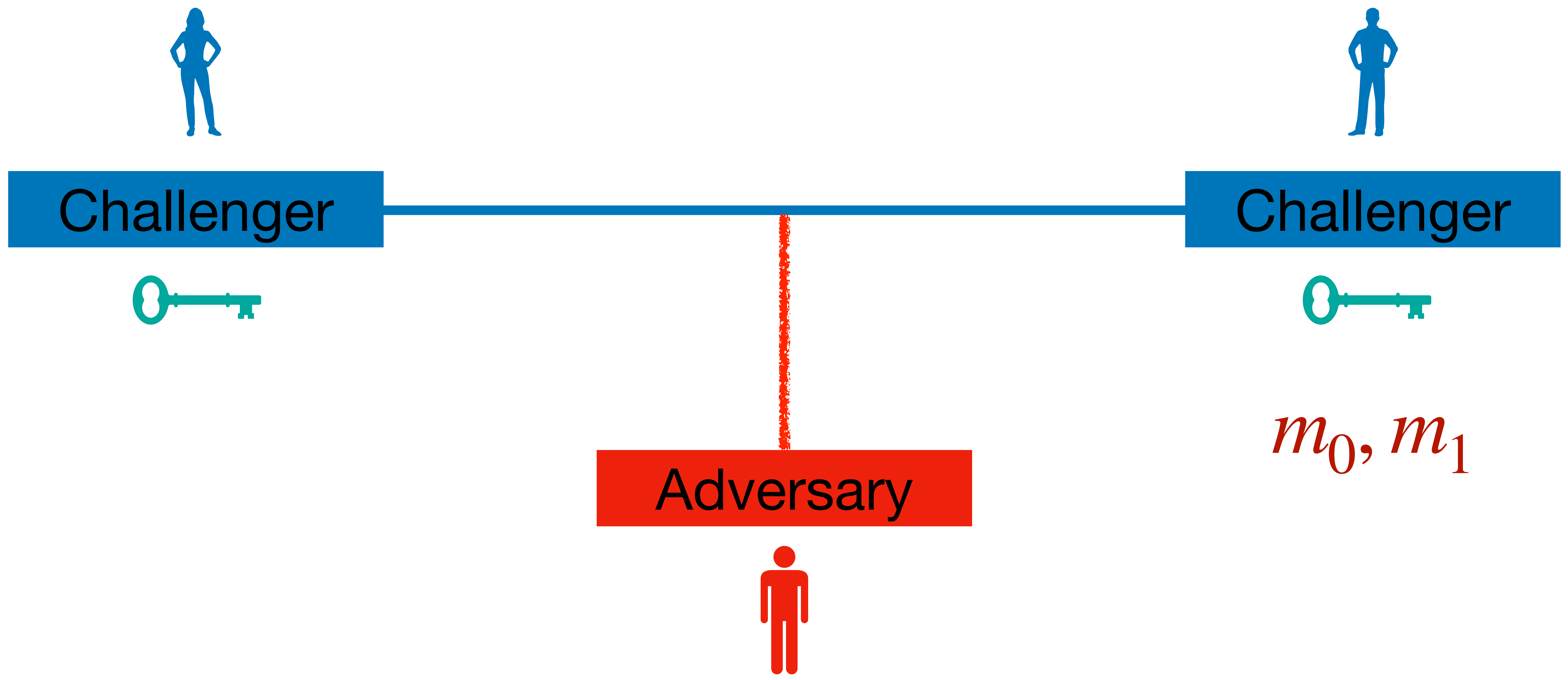
Standard Security [Goldwaser,Micali84]



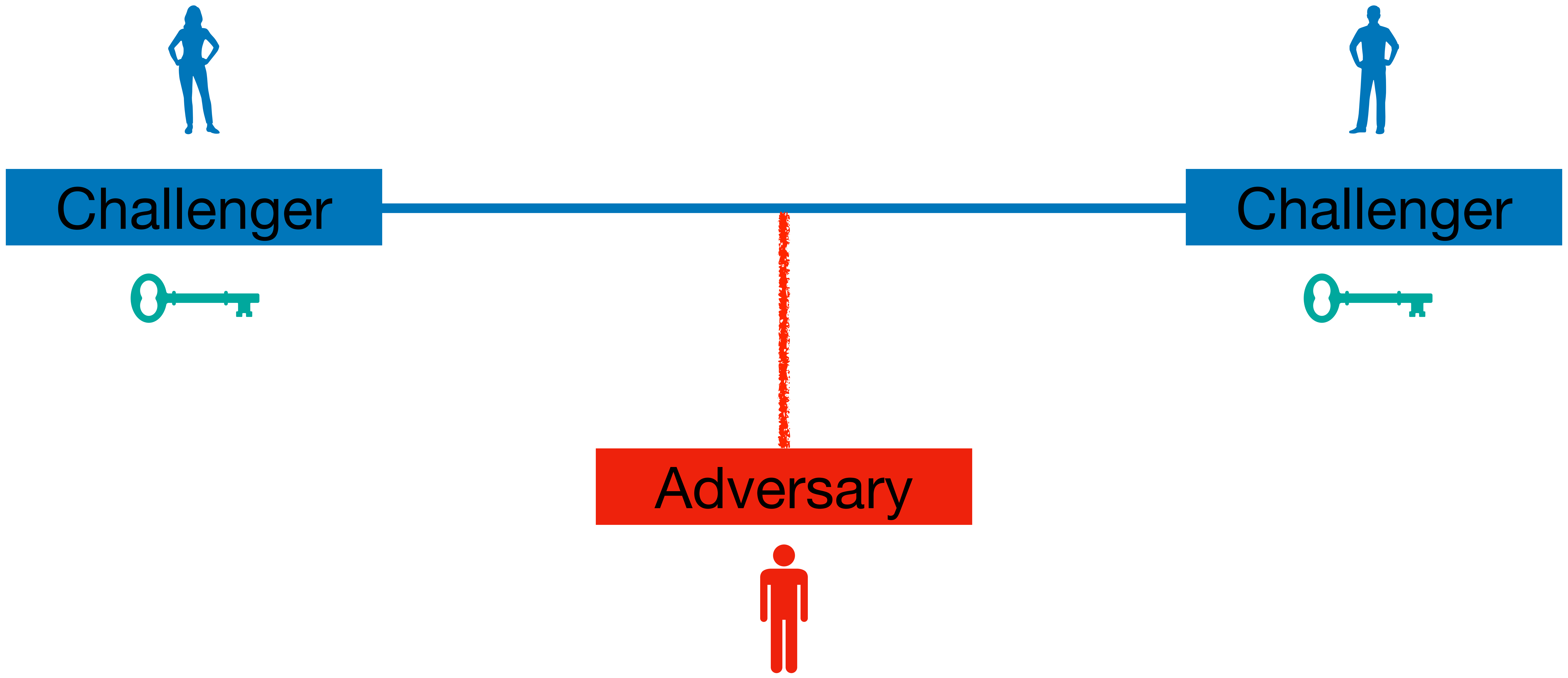
Standard Security [Goldwaser,Micali84]



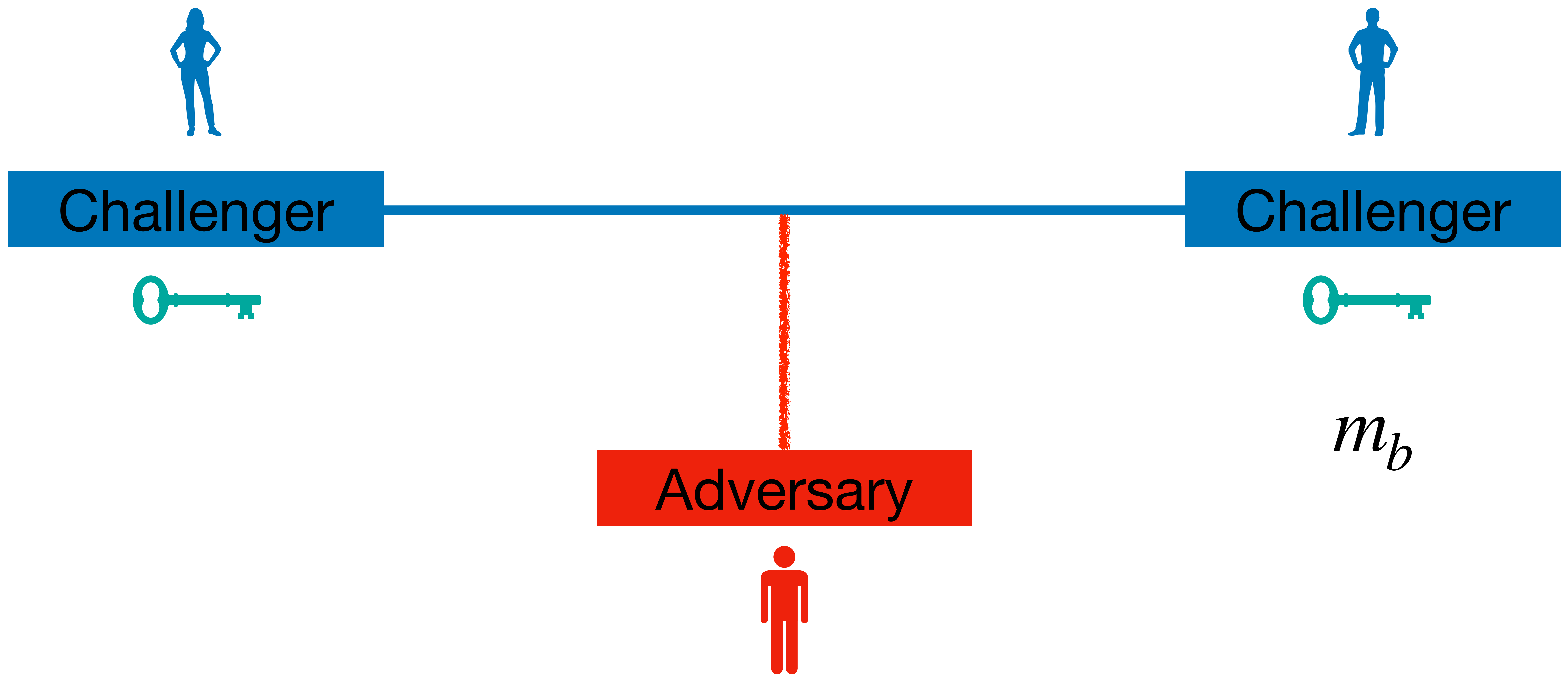
Standard Security [Goldwaser,Micali84]



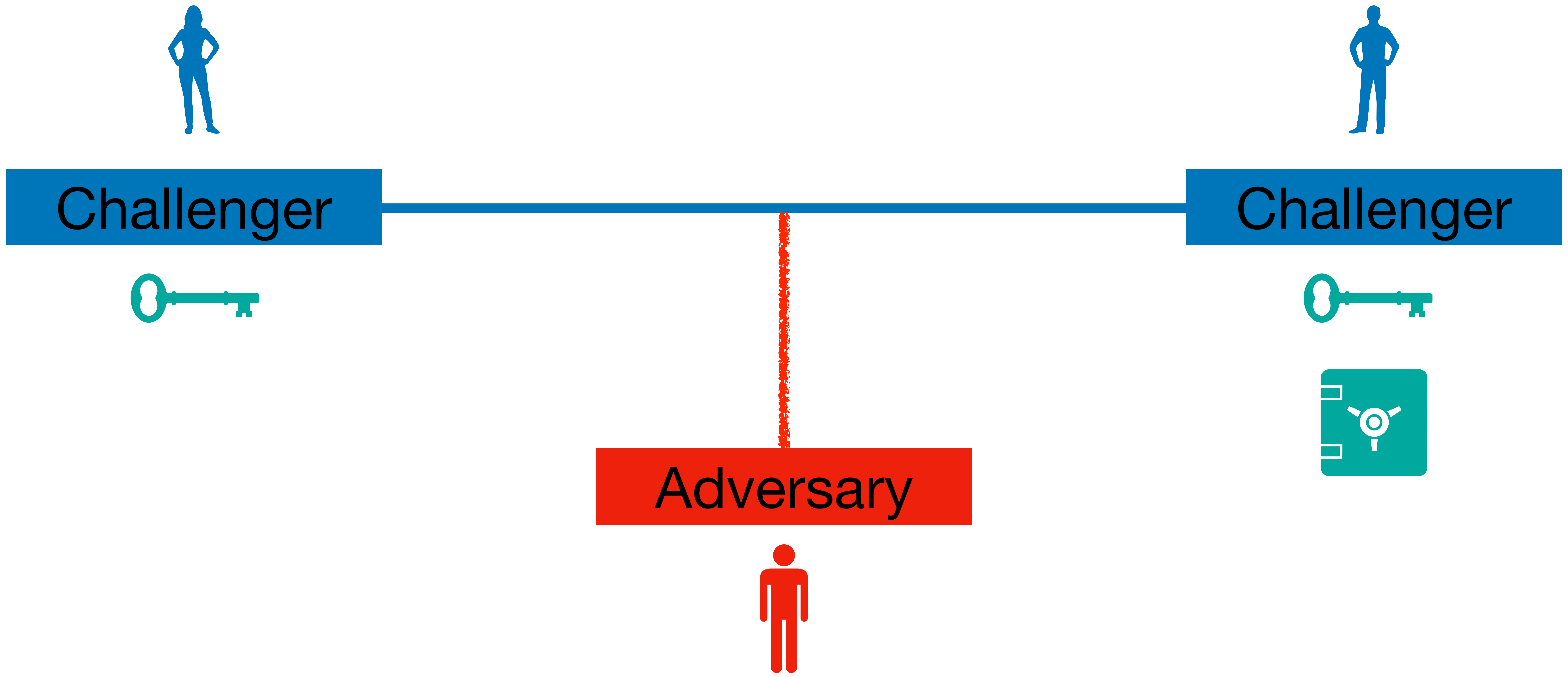
Standard Security [Goldwaser,Micali84]



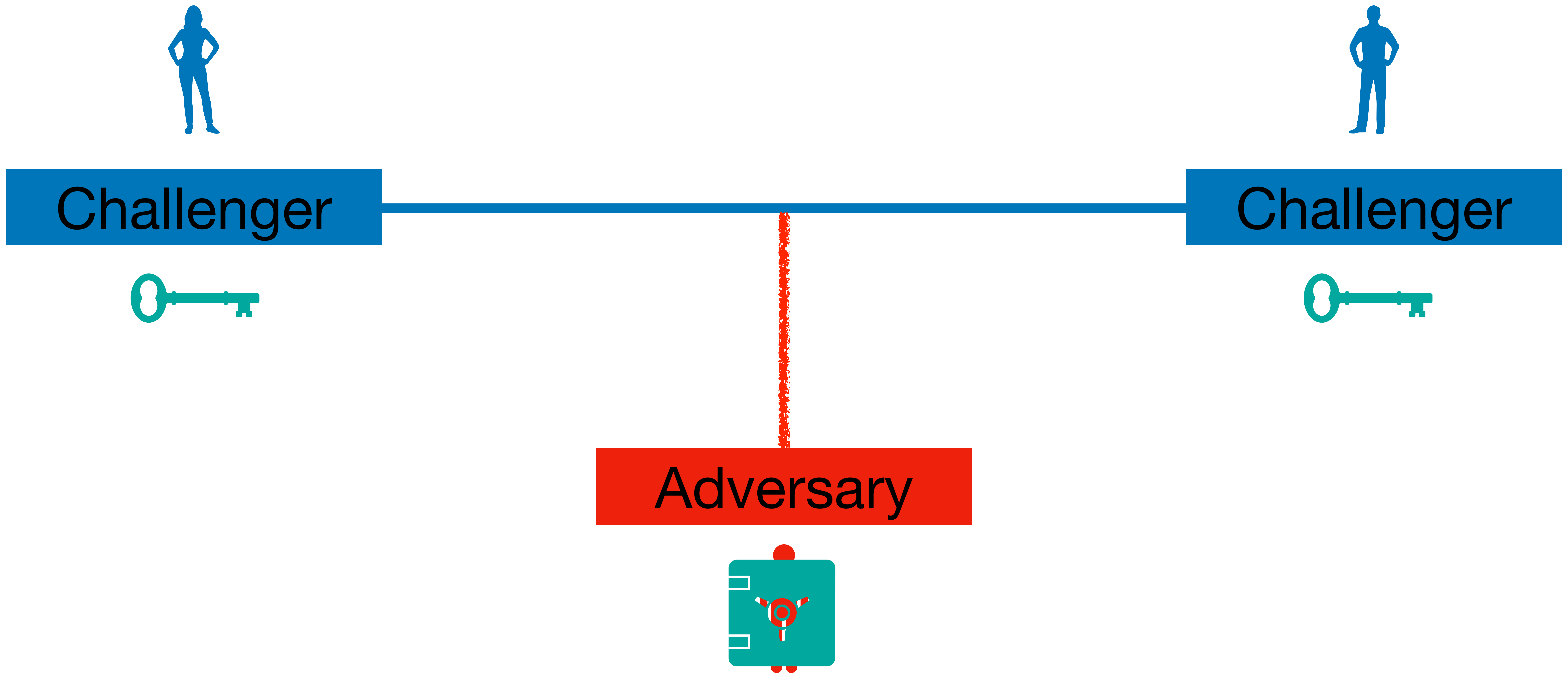
Standard Security [Goldwaser,Micali84]



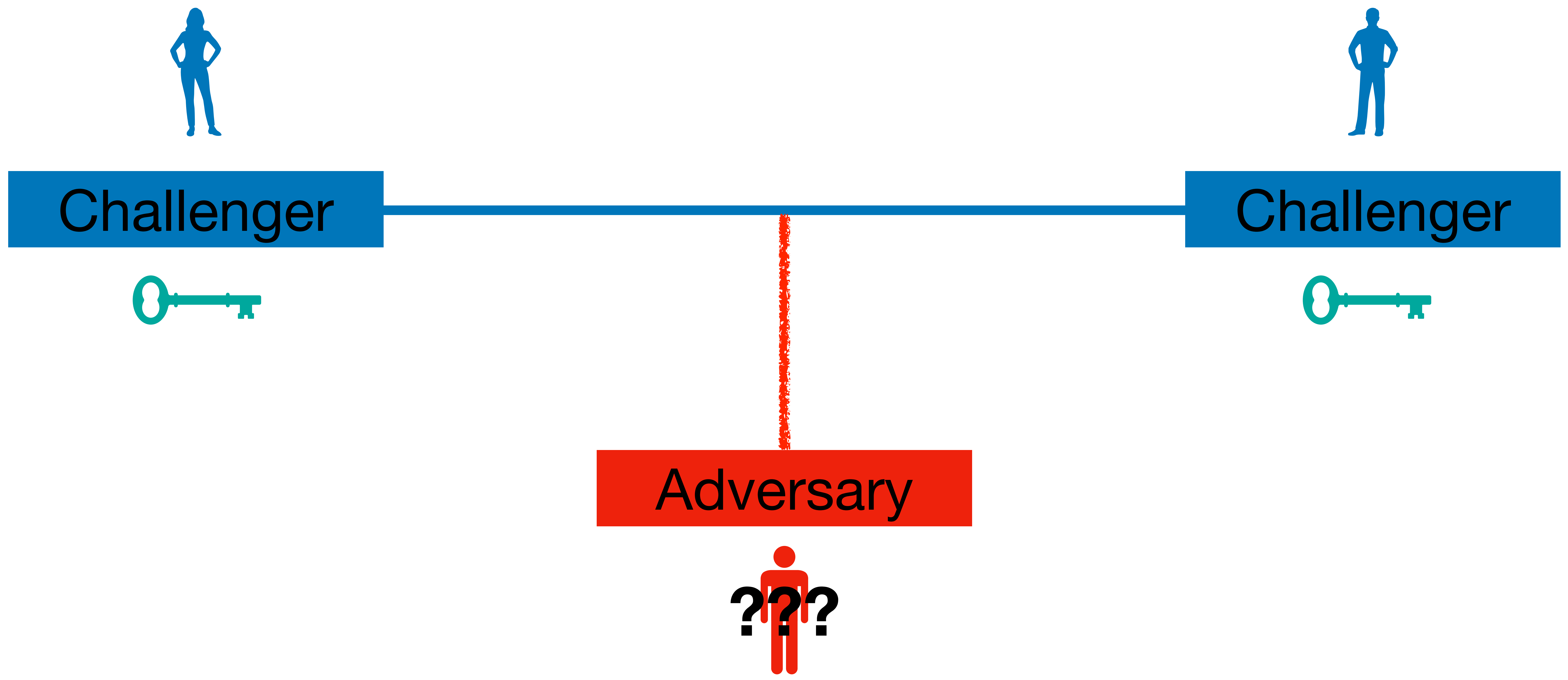
Standard Security [Goldwaser,Micali84]



Standard Security [Goldwaser,Micali84]



Standard Security [Goldwaser,Micali84]



One-Time Pad

One-Time Pad

- $Setup(1^{|m|}) \rightarrow$ Randomly generate $sk \in \{0,1\}^{|m|}$.

One-Time Pad

- $Setup(1^{|m|}) \rightarrow$ Randomly generate $sk \in \{0,1\}^{|m|}$.
- $Enc(sk, m) \rightarrow$ Returns $sk \oplus m$.

One-Time Pad

- $Setup(1^{|m|}) \rightarrow$ Randomly generate $sk \in \{0,1\}^{|m|}$.
- $Enc(sk, m) \rightarrow$ Returns $sk \oplus m$.

$$\begin{array}{r} 0 \\ \oplus\, 0 \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \\ \oplus\, 1 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ \oplus\, 0 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ \oplus\, 1 \\ \hline 0 \end{array}$$

One-Time Pad

- $Setup(1^{|m|}) \rightarrow$ Randomly generate $sk \in \{0,1\}^{|m|}$.
- $Enc(sk, m) \rightarrow$ Returns $sk \oplus m$.
- $Dec(sk, ct) \rightarrow$ Return $sk \oplus ct$.

$$\begin{array}{r} 0 \\ \oplus\, 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ \oplus\, 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ \oplus\, 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ \oplus\, 1 \\ \hline 0 \end{array}$$

One-Time Pad

- $Setup(1^{|m|}) \rightarrow$ Randomly generate $sk \in \{0,1\}^{|m|}$.
- $Enc(sk, m) \rightarrow$ Returns $sk \oplus m$.
- $Dec(sk, ct) \rightarrow$ Return $sk \oplus ct$.
- $Setup(1^5) \rightarrow$ Generated $\underbrace{11001}_{sk}$.

$$\begin{array}{r} 0 \\ \oplus\, 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ \oplus\, 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ \oplus\, 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ \oplus\, 1 \\ \hline 0 \end{array}$$

One-Time Pad

- $Setup(1^{|m|}) \rightarrow$ Randomly generate $sk \in \{0,1\}^{|m|}$.
- $Enc(sk, m) \rightarrow$ Returns $sk \oplus m$.
- $Dec(sk, ct) \rightarrow$ Return $sk \oplus ct$.
- $Setup(1^5) \rightarrow$ Generated $\underbrace{11001}_{sk}$.
- $Enc(sk, 11100) \rightarrow$ Returns $\underbrace{11100}_{sk} \oplus \underbrace{11001}_{m} = \underbrace{00101}_{ct}$.

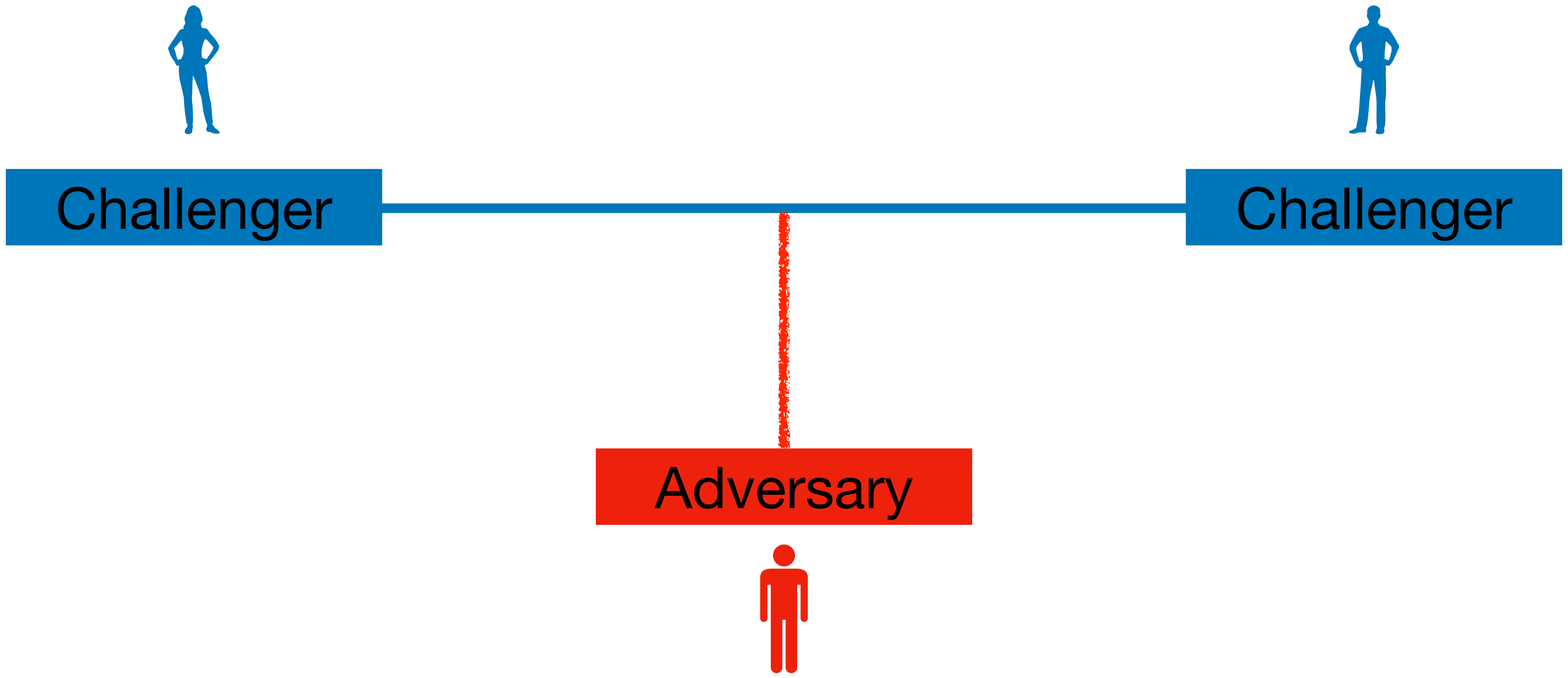
$$\begin{array}{r} 0 \\ \oplus\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ \oplus\ 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ \oplus\ 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ \oplus\ 1 \\ \hline 0 \end{array}$$

One-Time Pad

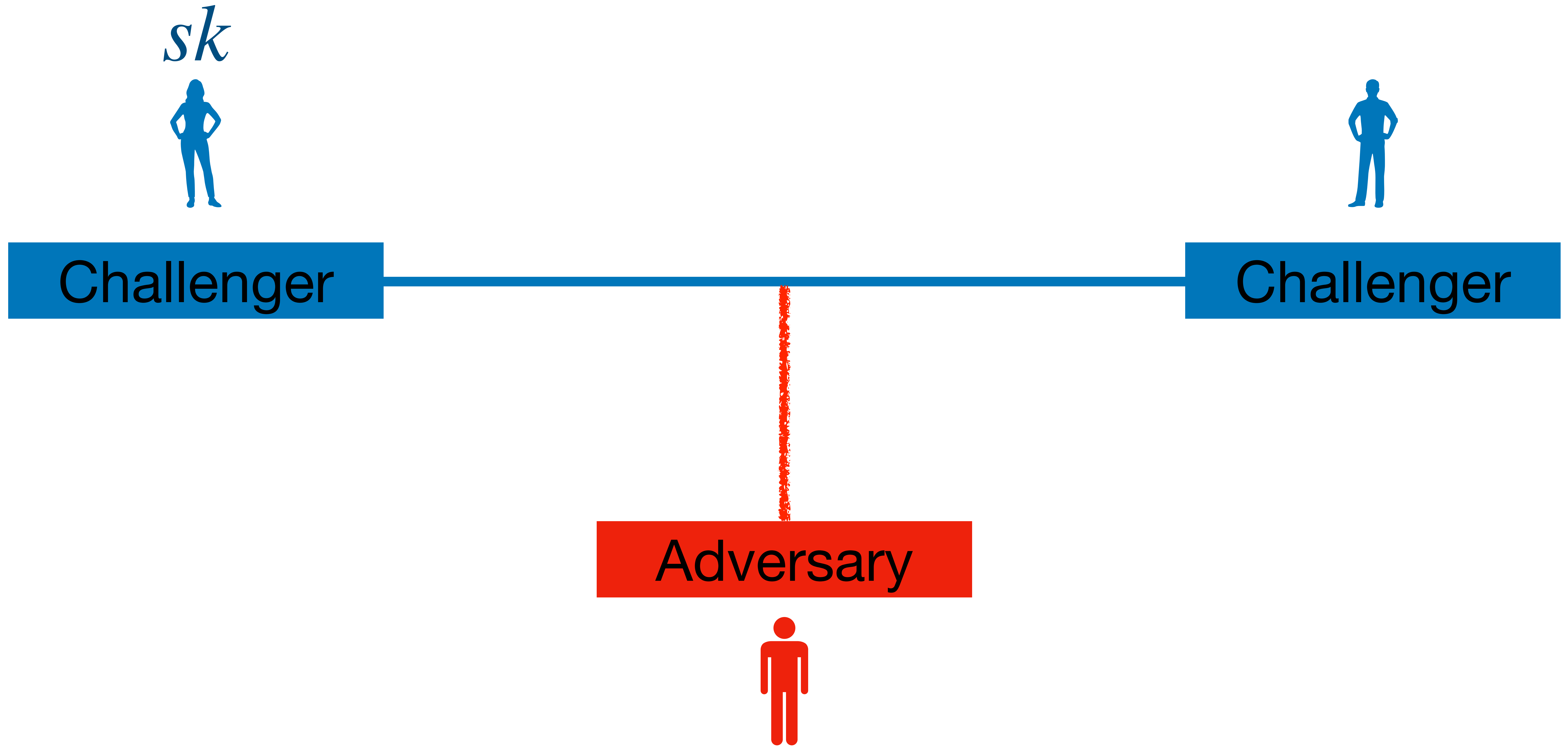
- $Setup(1^{|m|}) \rightarrow$ Randomly generate $sk \in \{0,1\}^{|m|}$.
- $Enc(sk, m) \rightarrow$ Returns $sk \oplus m$.
- $Dec(sk, ct) \rightarrow$ Return $sk \oplus ct$.
- $Setup(1^5) \rightarrow$ Generated $\underbrace{11001}_{sk}$.
- $Enc(sk, 11100) \rightarrow$ Returns $\underbrace{11100}_{sk} \oplus \underbrace{11001}_{m} = \underbrace{00101}_{ct}$.
- $Dec(sk, ct) \rightarrow$ Return $\underbrace{11001}_{sk} \oplus \underbrace{00101}_{ct} = \underbrace{11100}_{m}$.

$$\begin{array}{r} 0 \\ \oplus\, 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ \oplus\, 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ \oplus\, 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ \oplus\, 1 \\ \hline 0 \end{array}$$

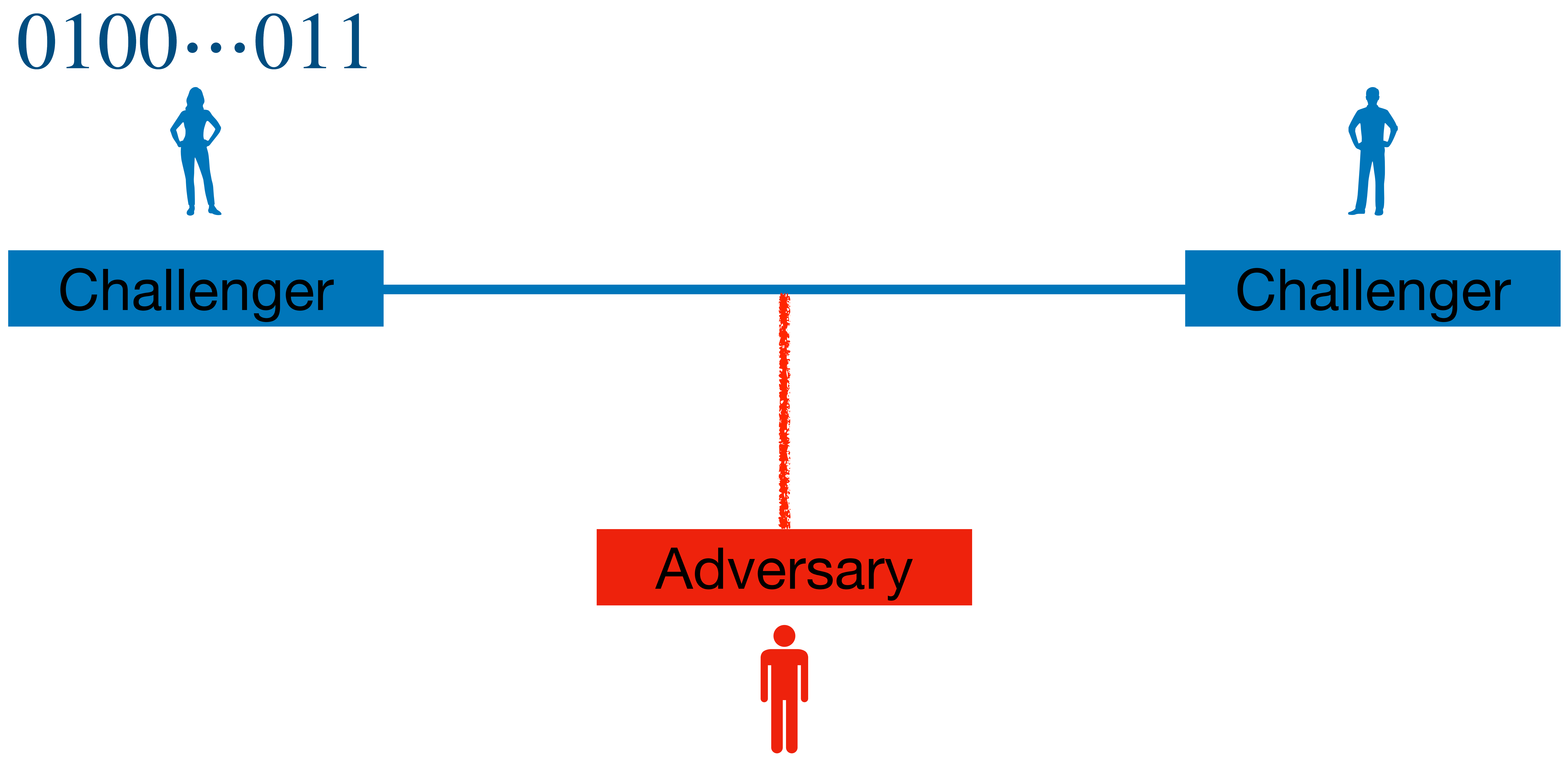
One Time Pad



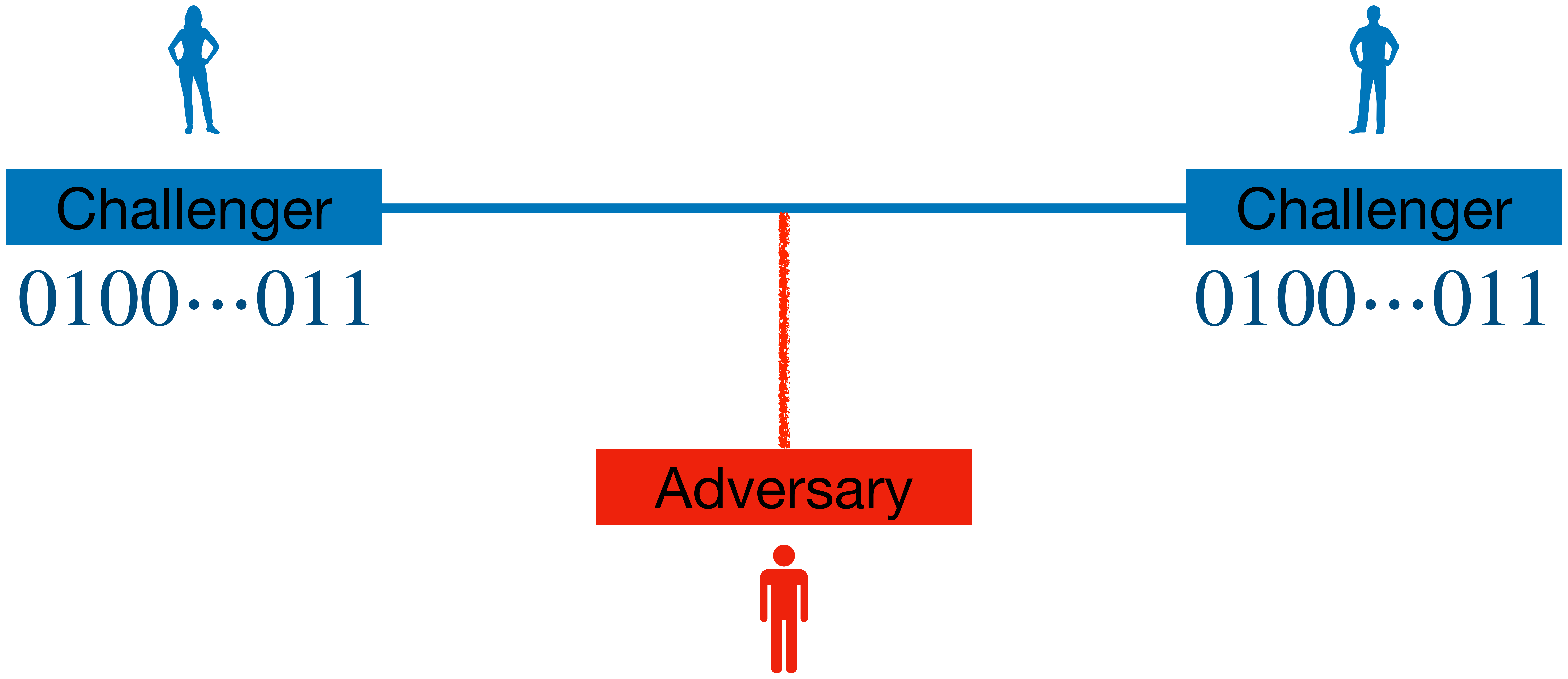
One Time Pad



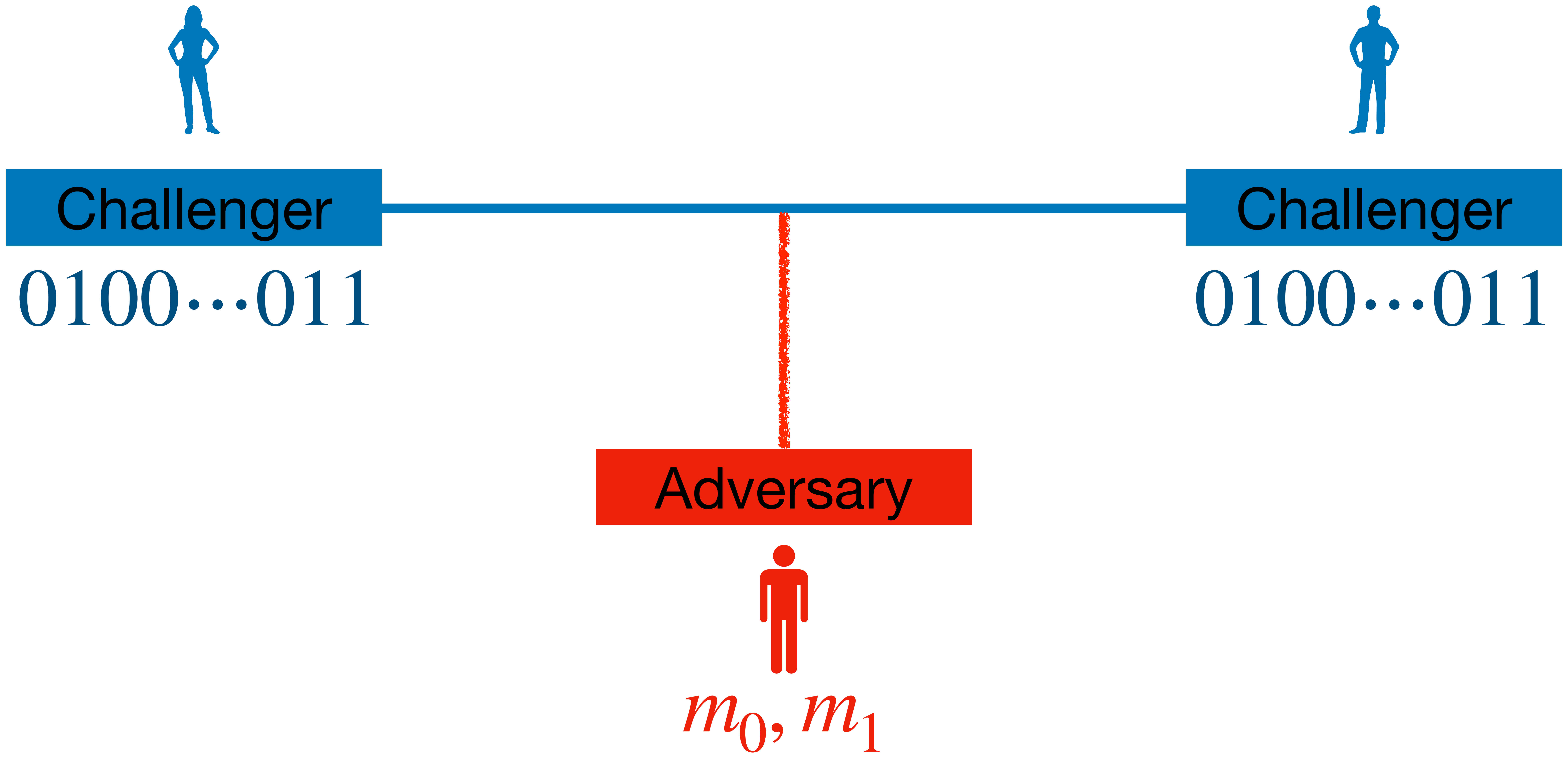
One Time Pad



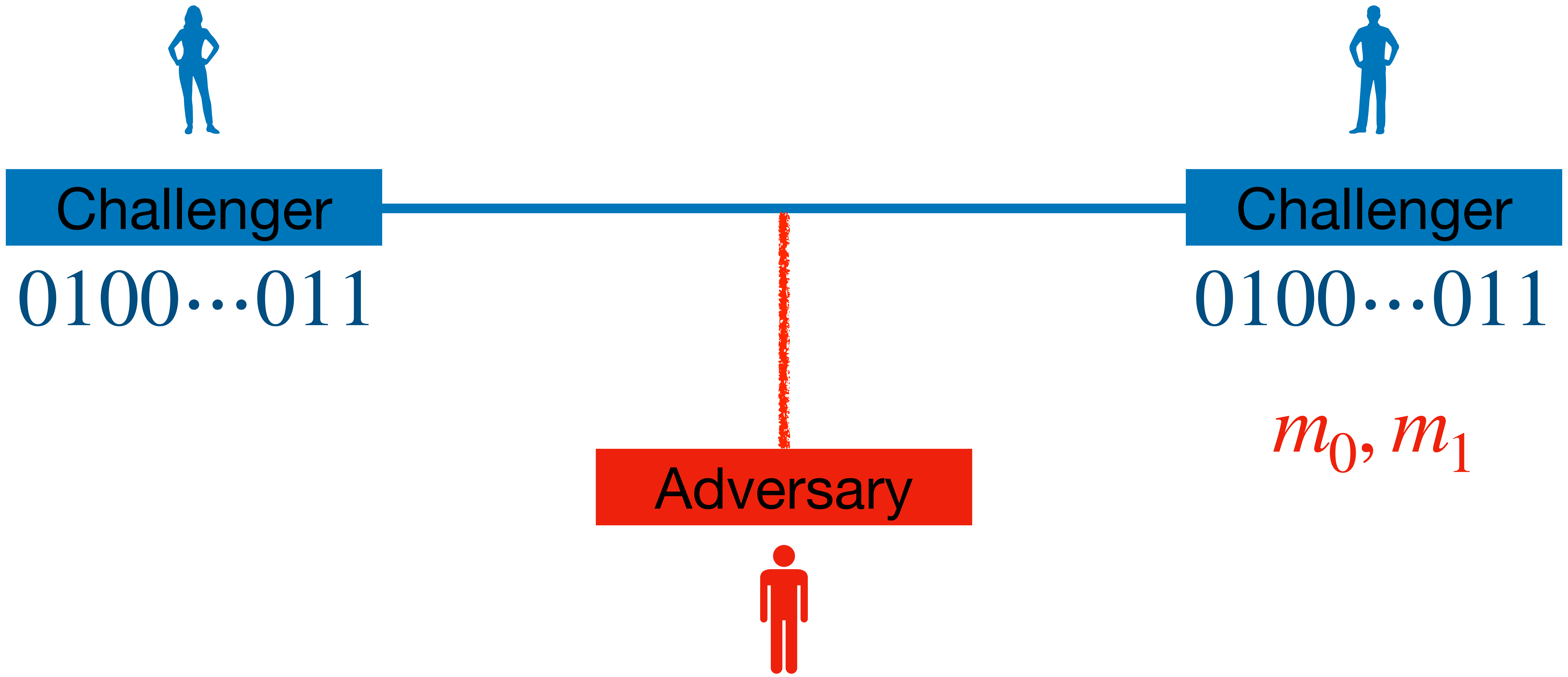
One Time Pad



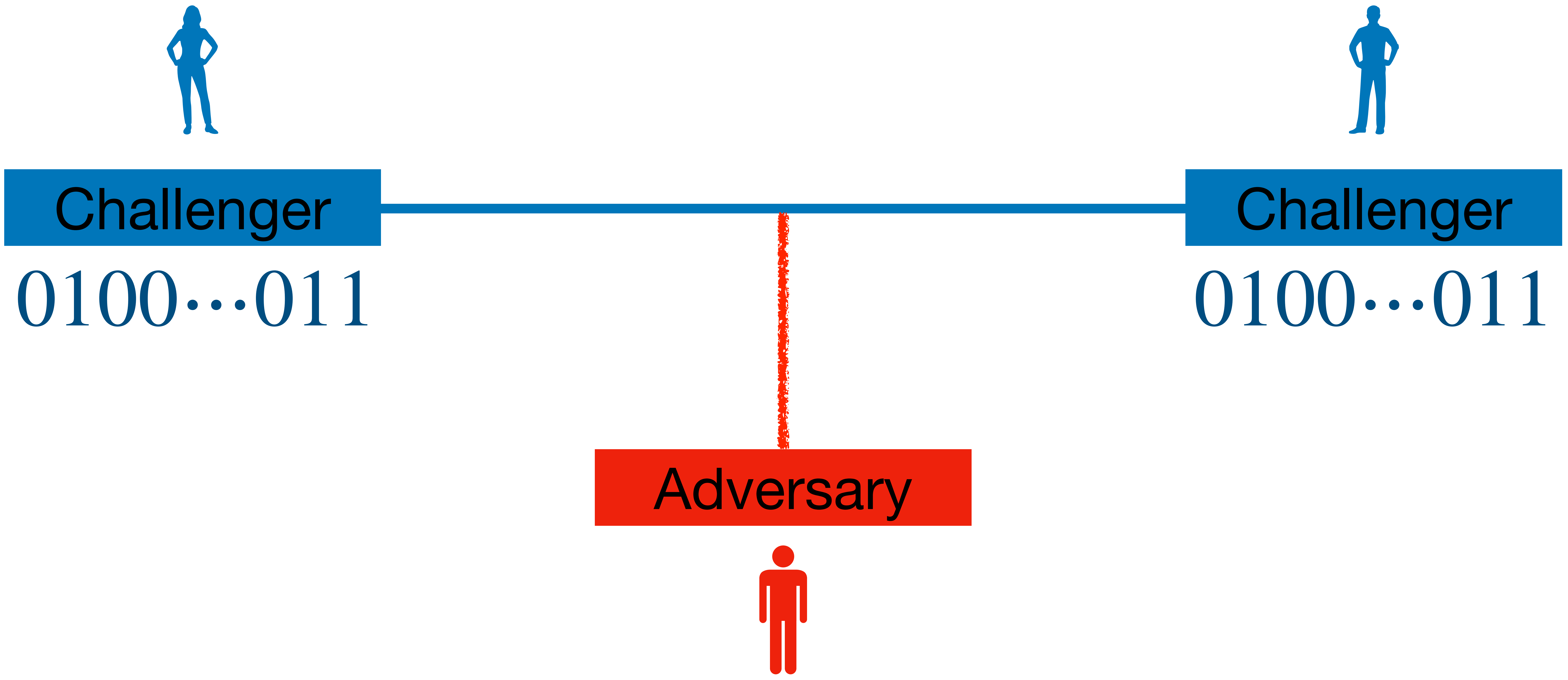
One Time Pad


Challenger
Challenger
$0100\cdots011$
$0100\cdots011$
Adversary
m_0, m_1

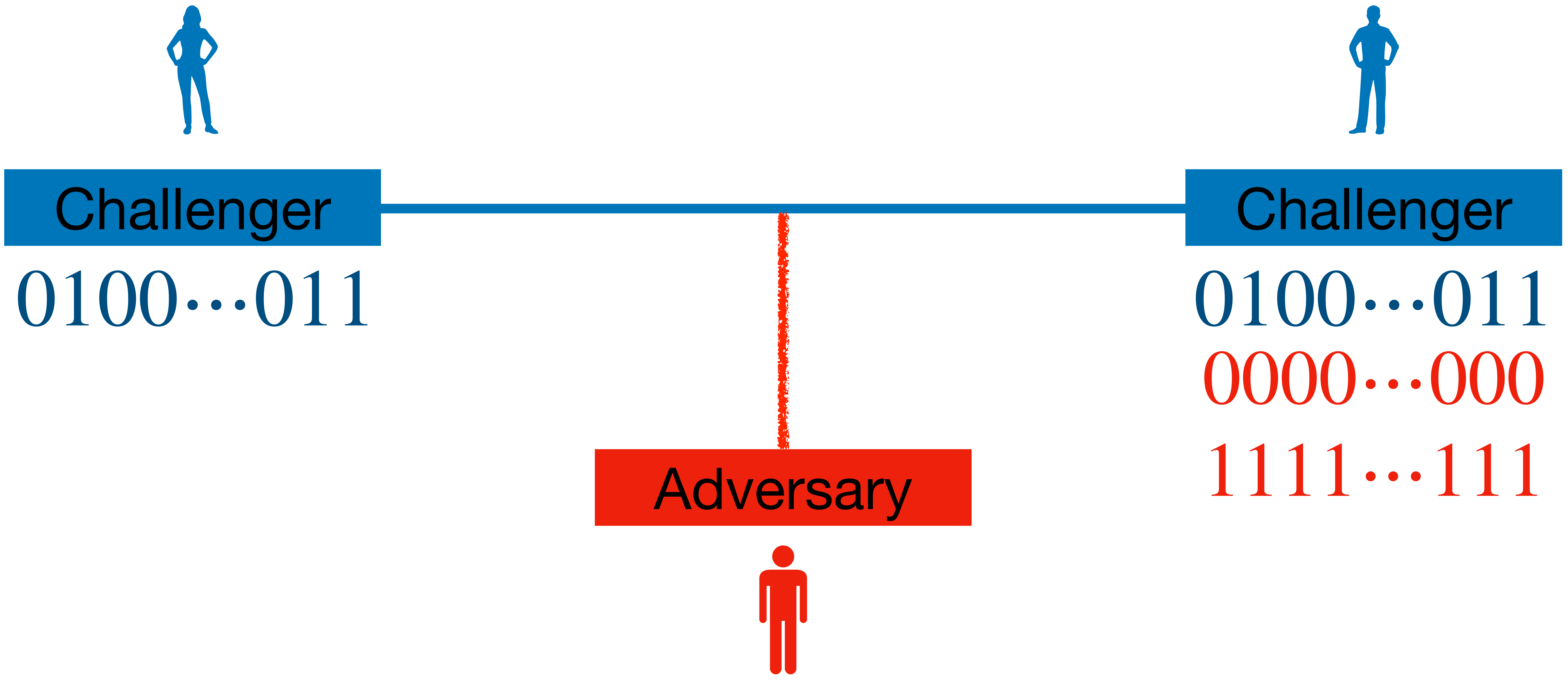
One Time Pad



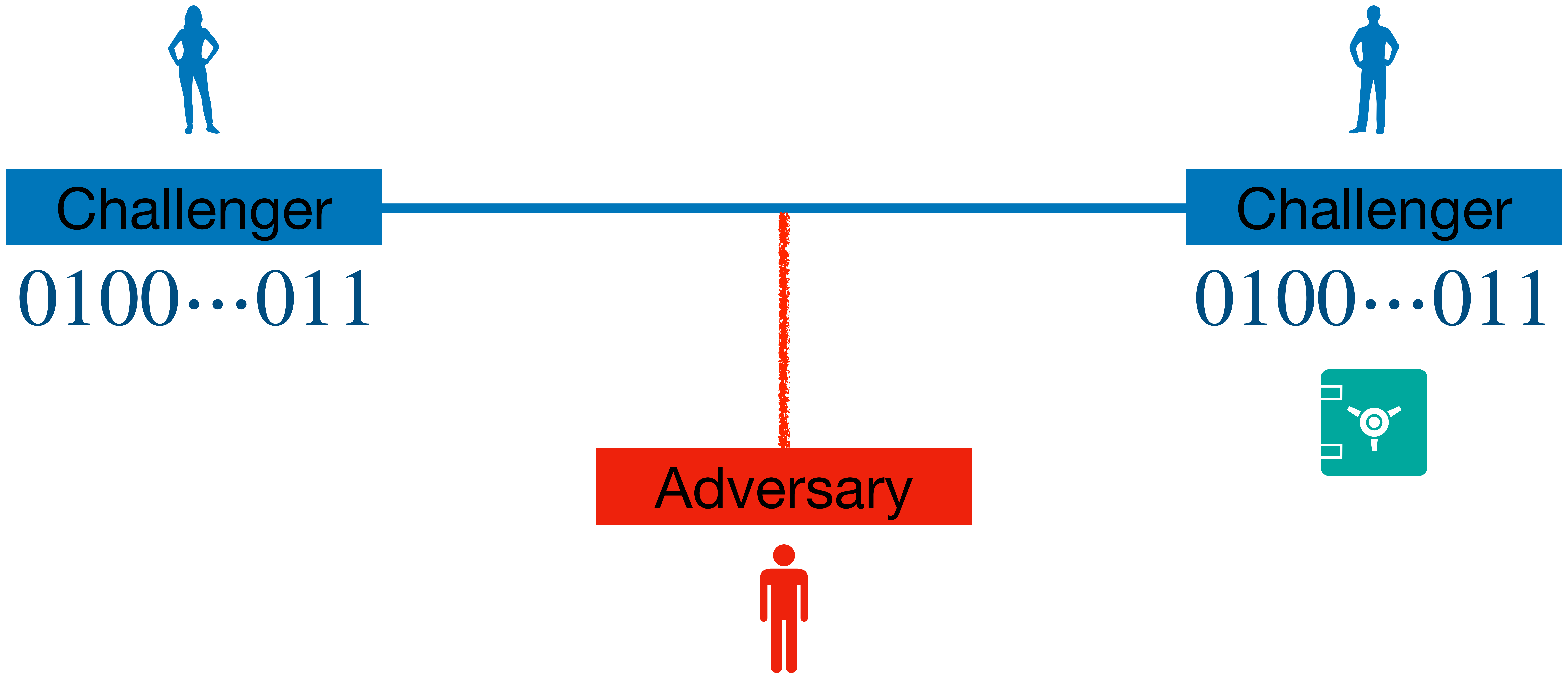
One Time Pad



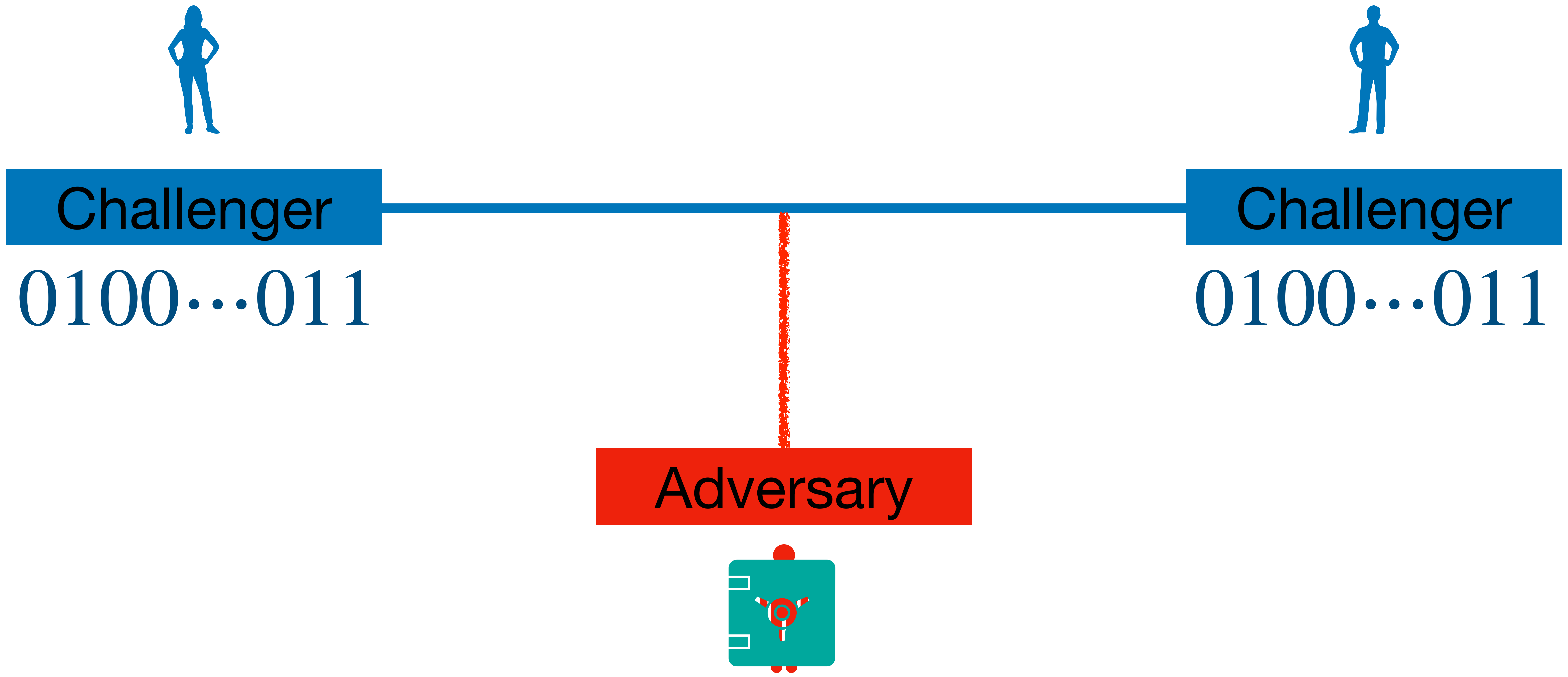
One Time Pad


Challenger
0100···011
Challenger
0100···011
0000···000
1111···111
Adversary

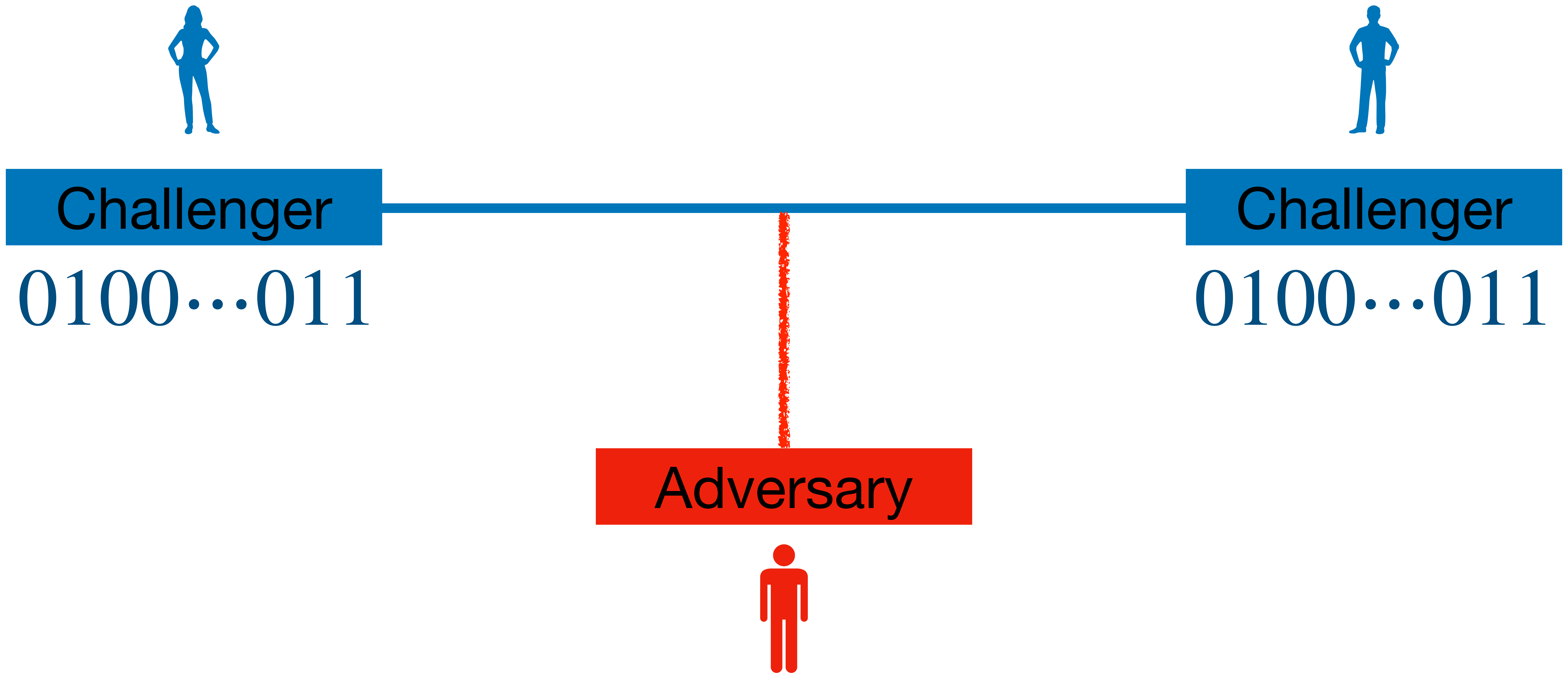
One Time Pad



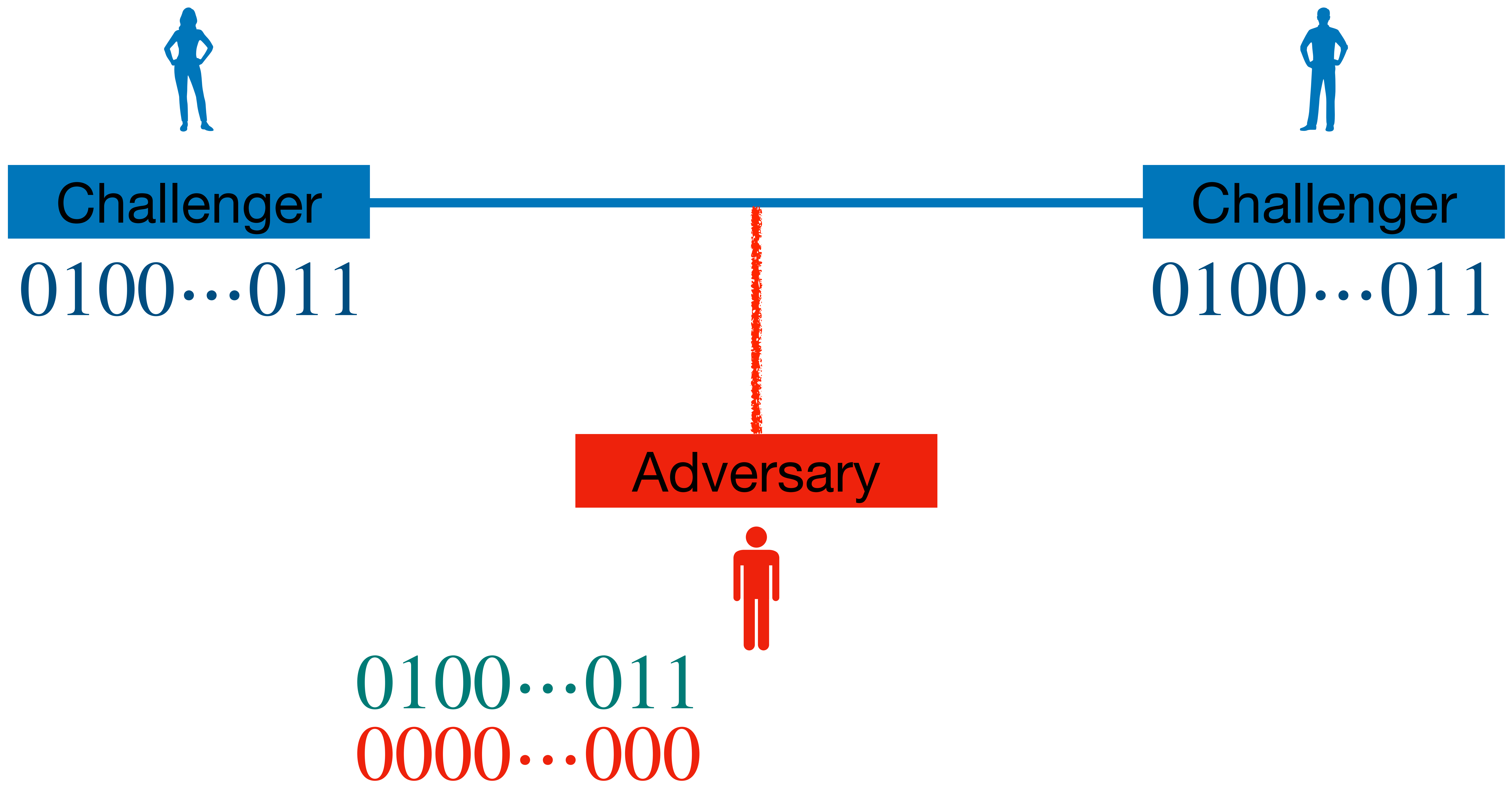
One Time Pad



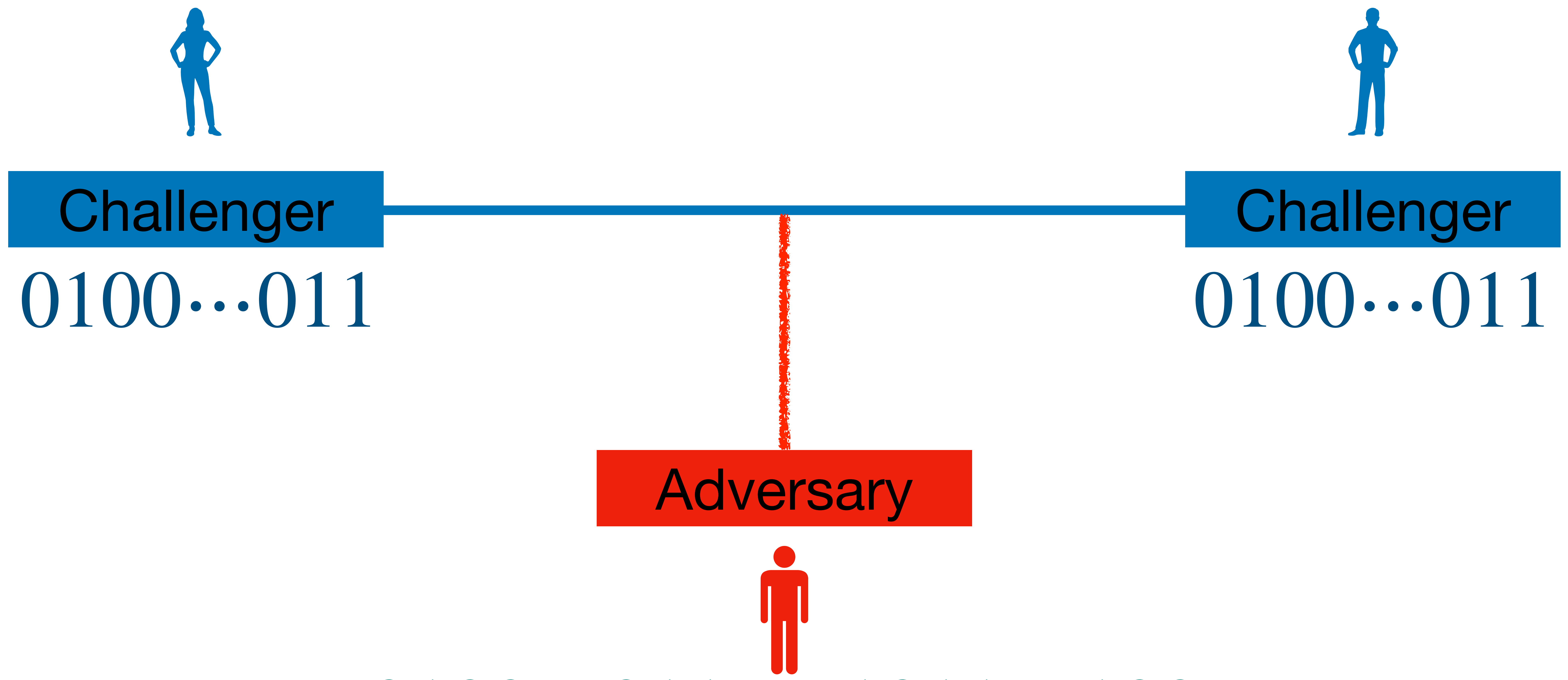
One Time Pad



One Time Pad


Challenger
0100…011
Challenger
0100…011
Adversary
0100…011
0000…000

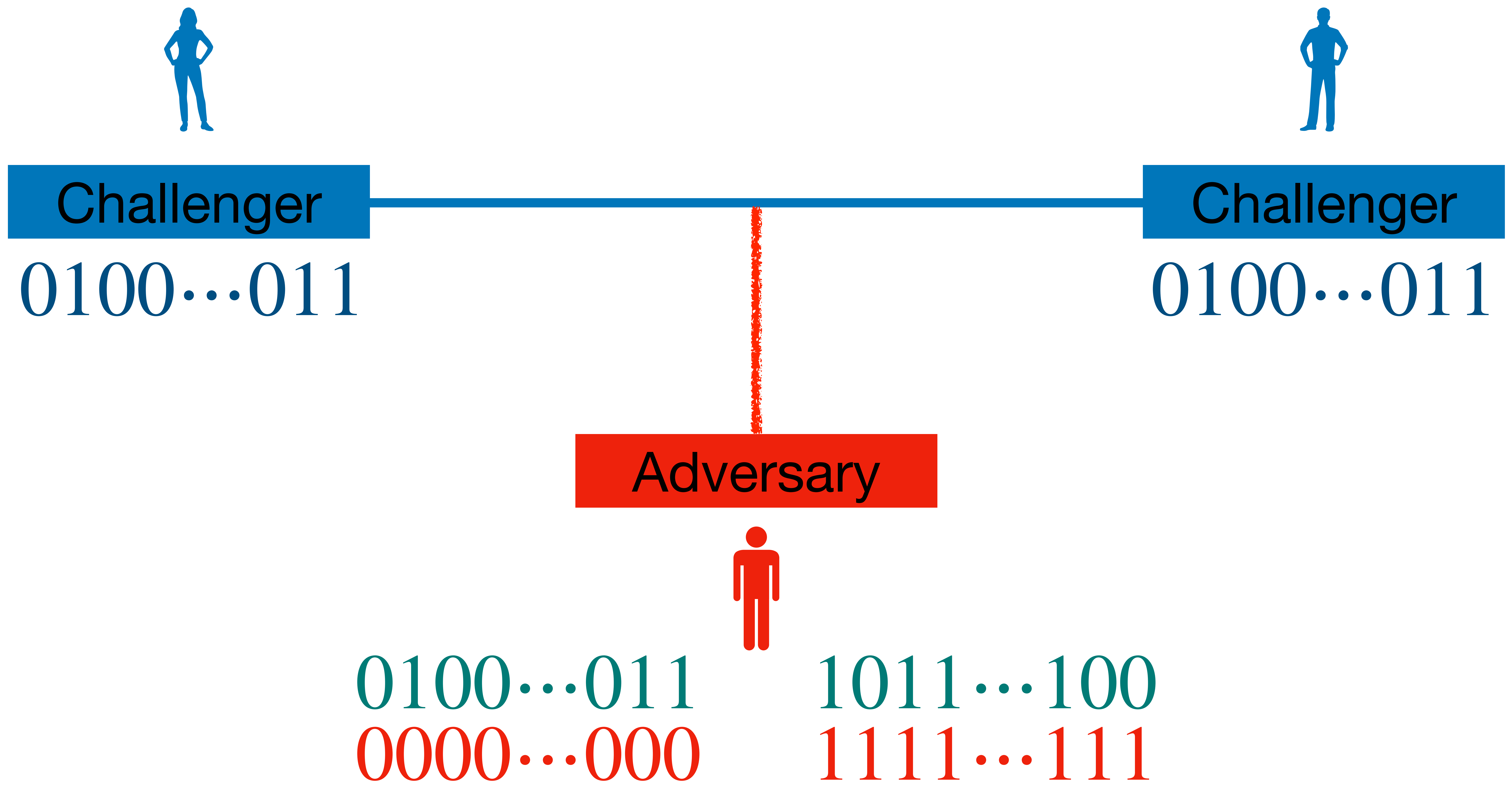
One Time Pad



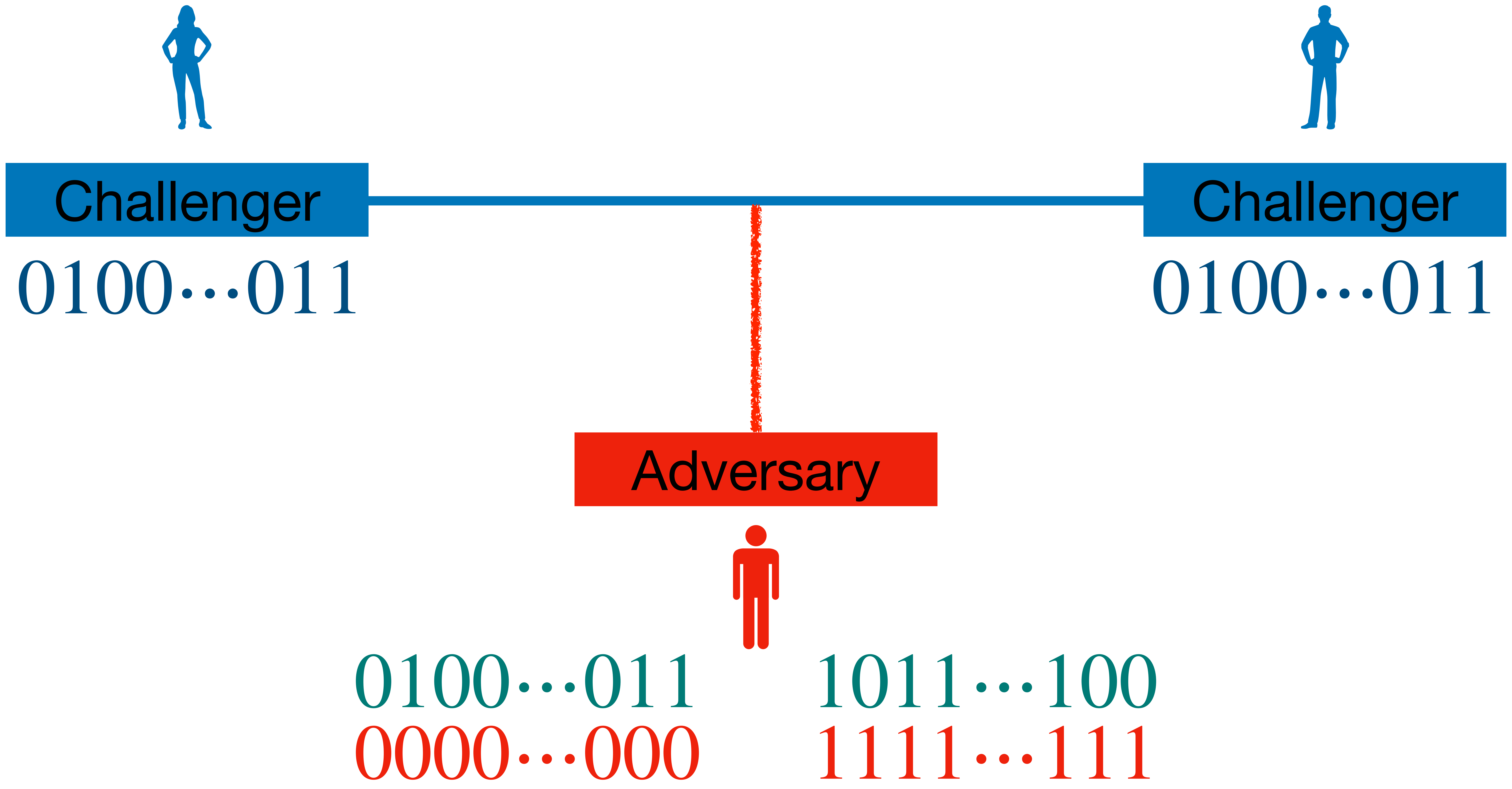
0100…011 1011…100

0000…000 1111…111

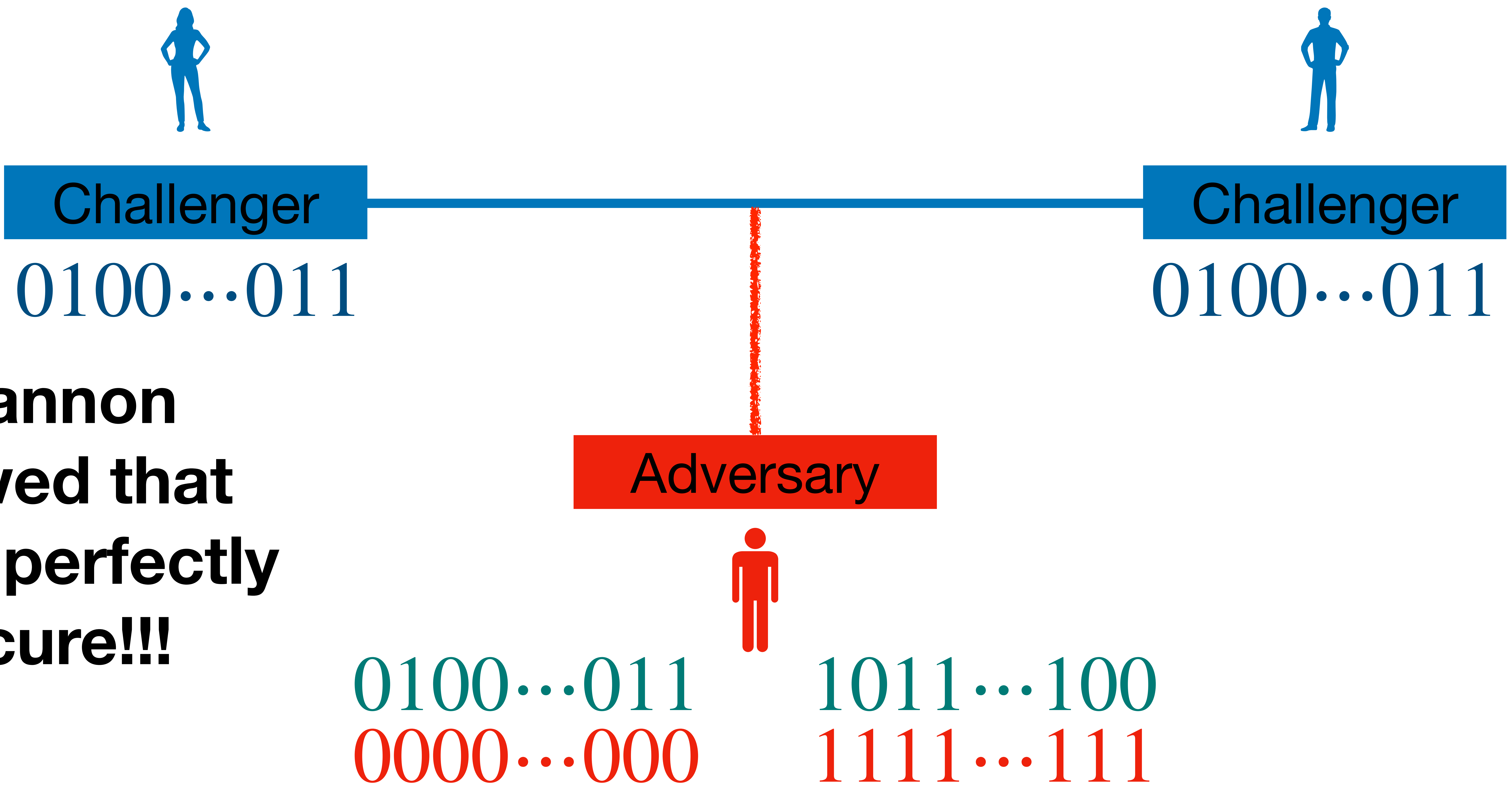
One Time Pad


Challenger
0100⋯011
Challenger
0100⋯011
Adversary
0100⋯011
0000⋯000
1011⋯100
1111⋯111

One Time Pad


Challenger
0100⋯011
Challenger
0100⋯011
Adversary
0100⋯011
0000⋯000
1011⋯100
1111⋯111

One Time Pad



Shannon showed that this is perfectly secure!!!

Can Secret Key be leaked?

Can Secret Key be leaked?

- Standard security says that adversary cannot distinguish between encryptions of two different message provided **no** information of secret key is leaked.

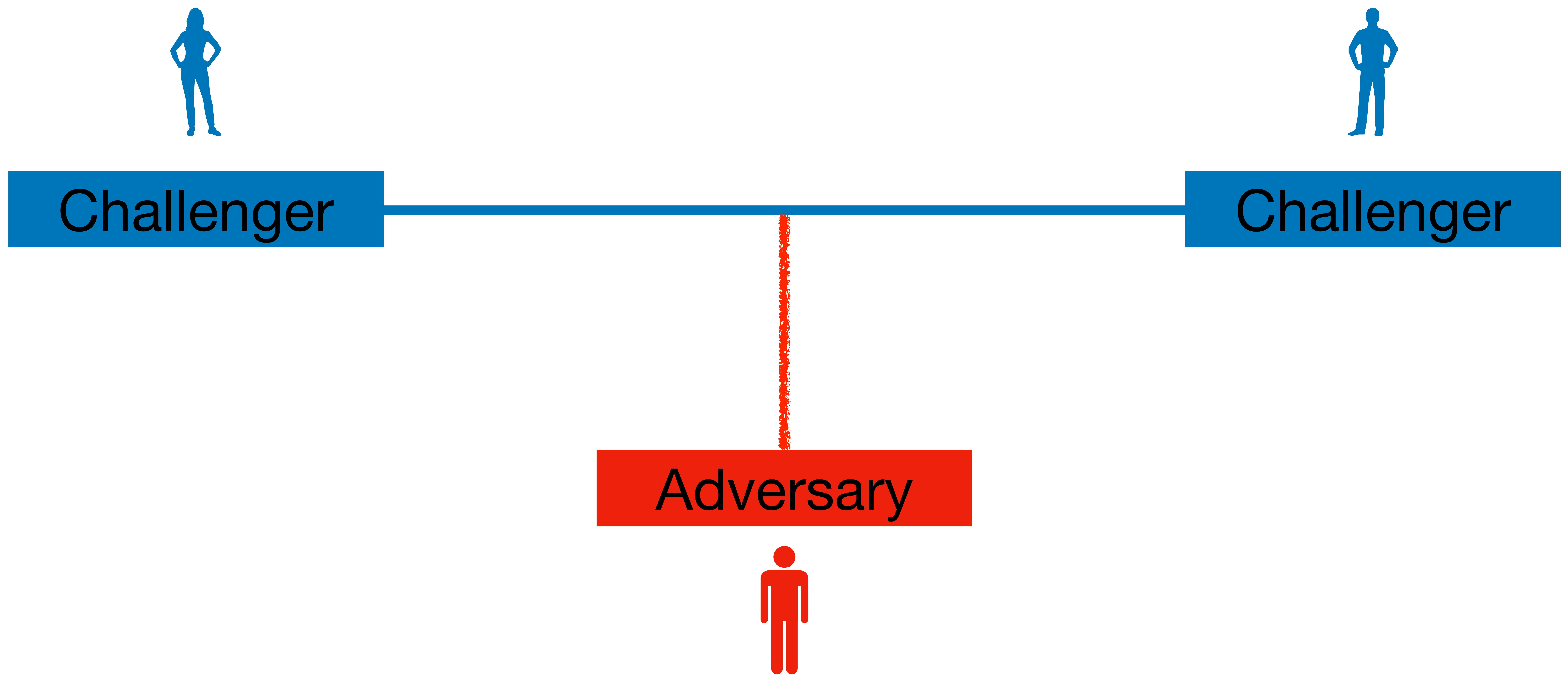
Can Secret Key be leaked?

- Standard security says that adversary cannot distinguish between encryptions of two different message provided **no** information of secret key is leaked.

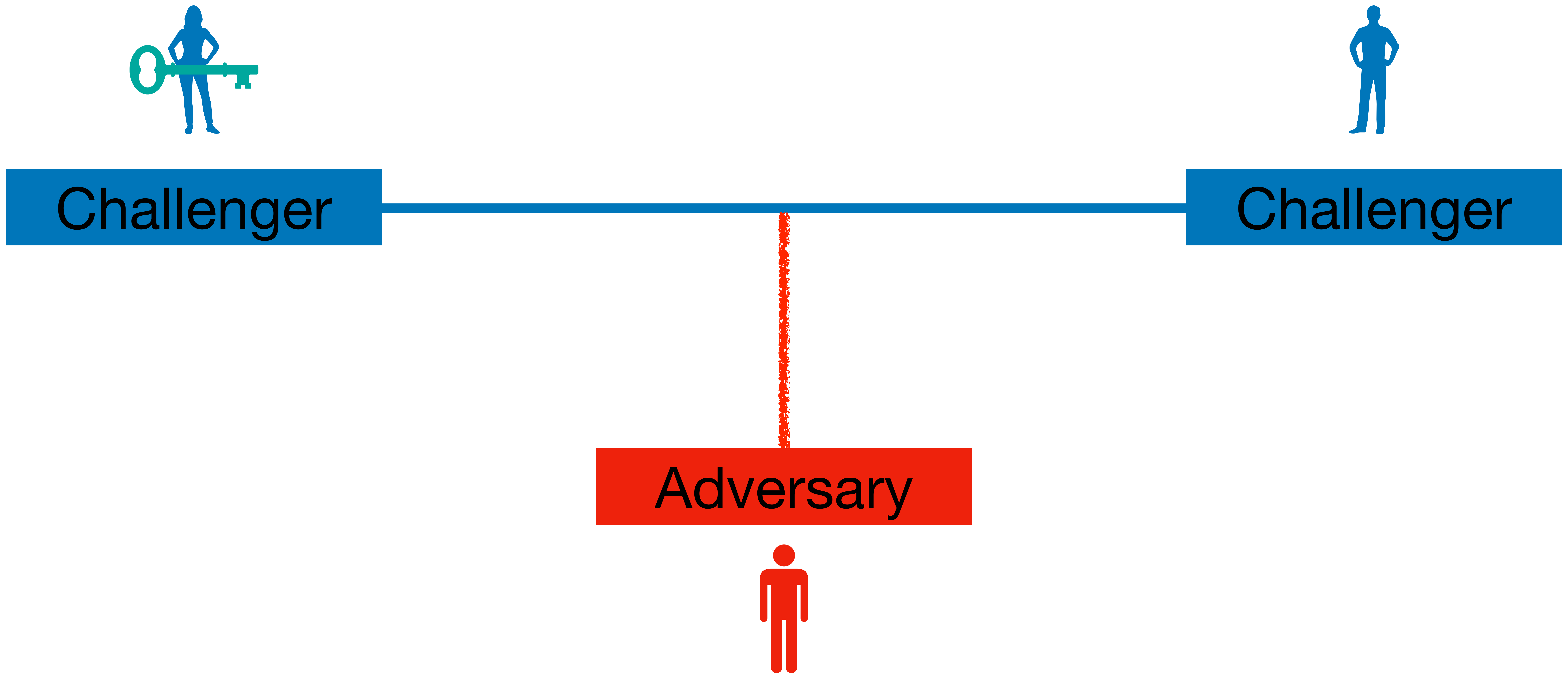
- In practice, secret key can be leaked using side-channel attacks.

Leakage-Resilience

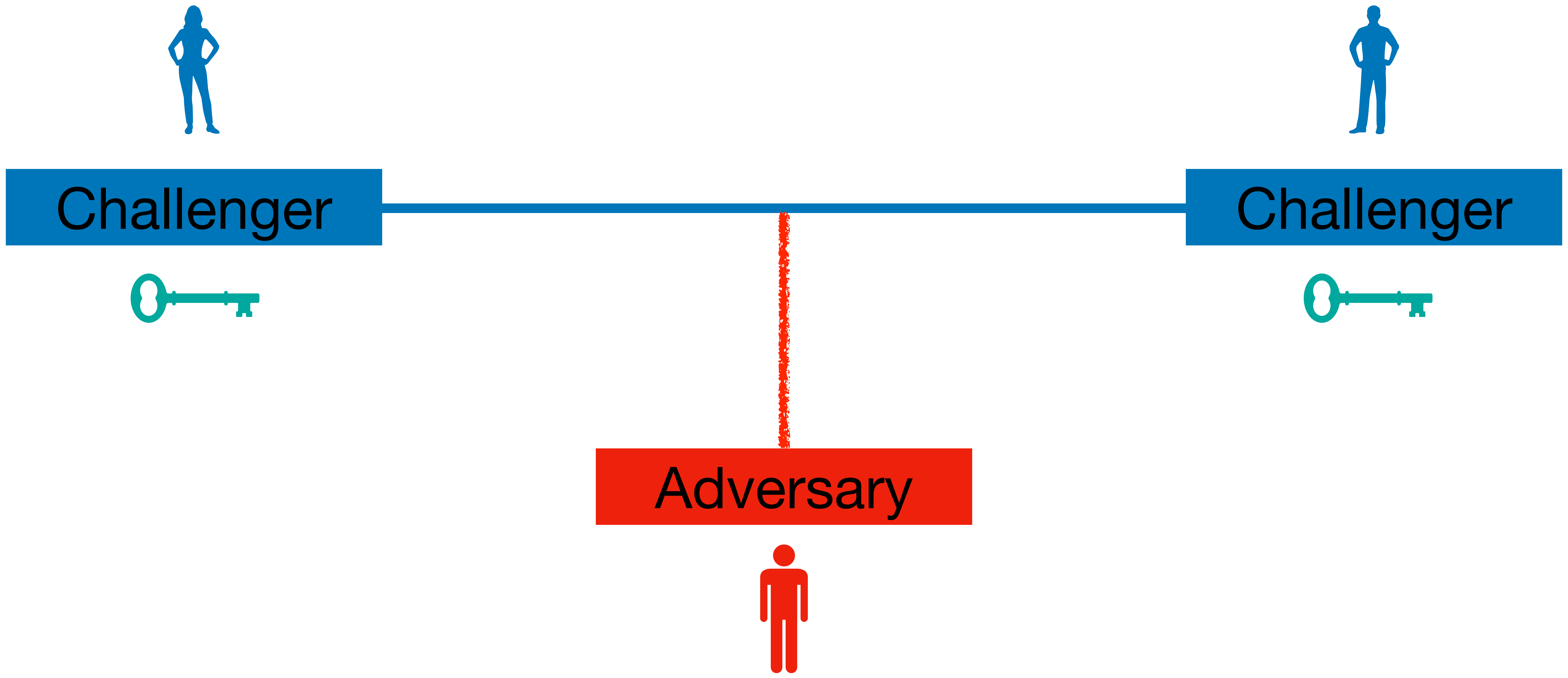
Security against Leakage



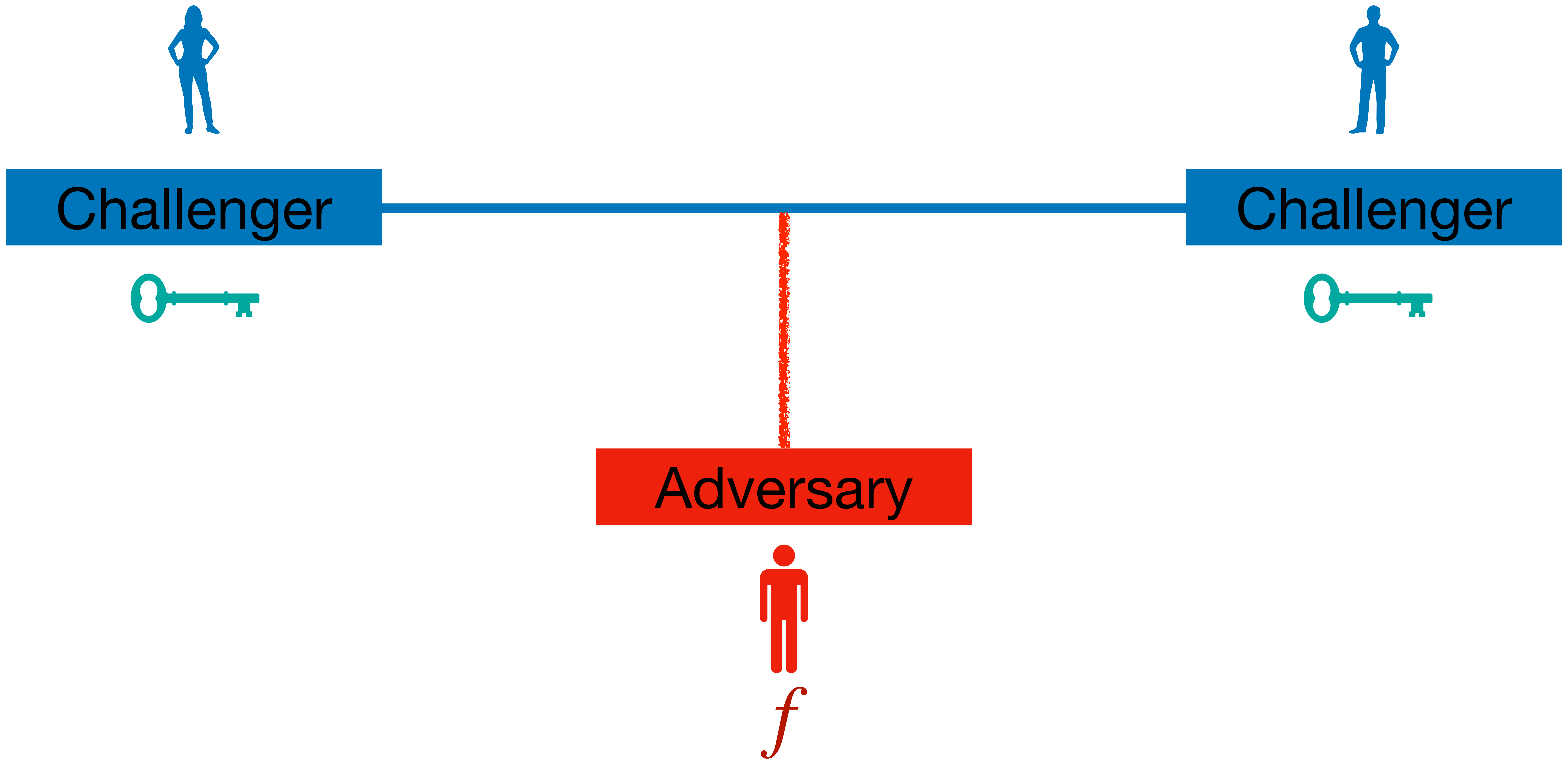
Security against Leakage



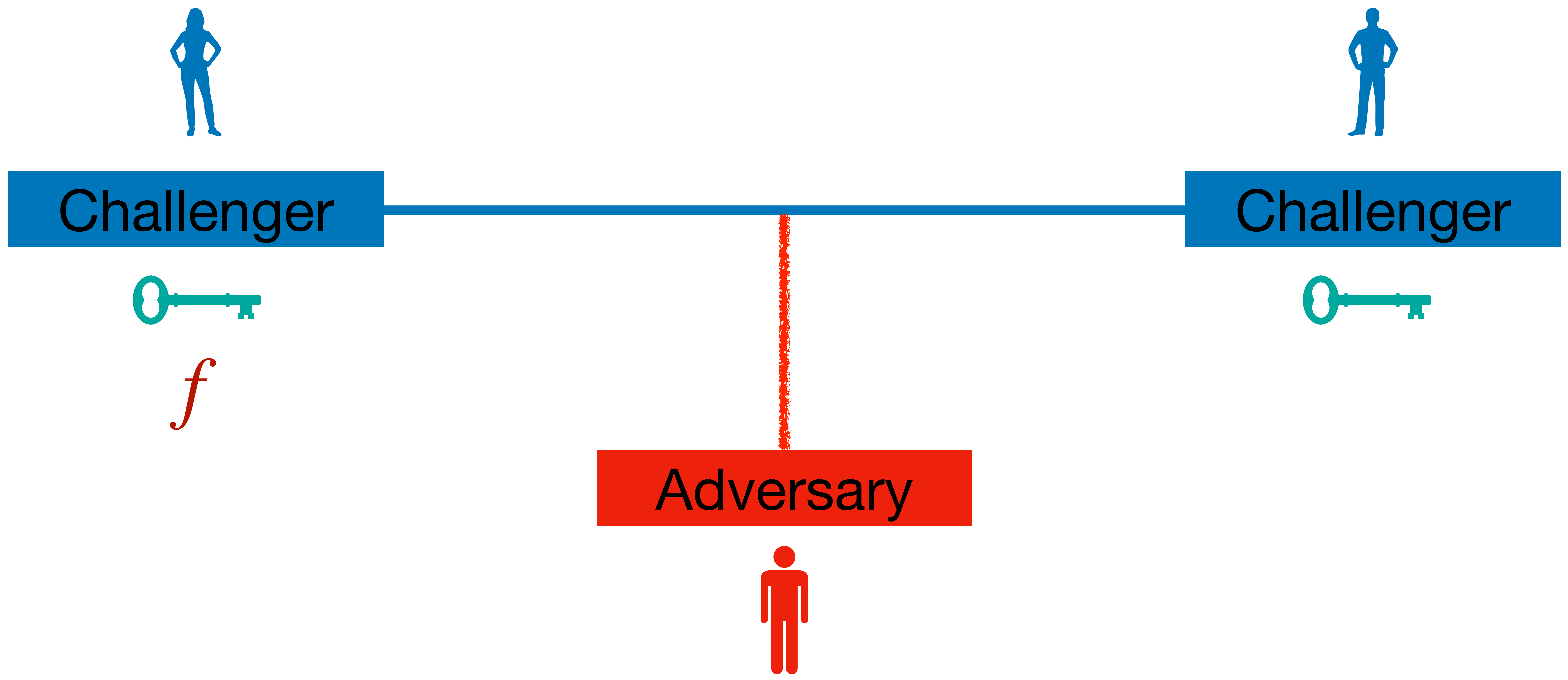
Security against Leakage



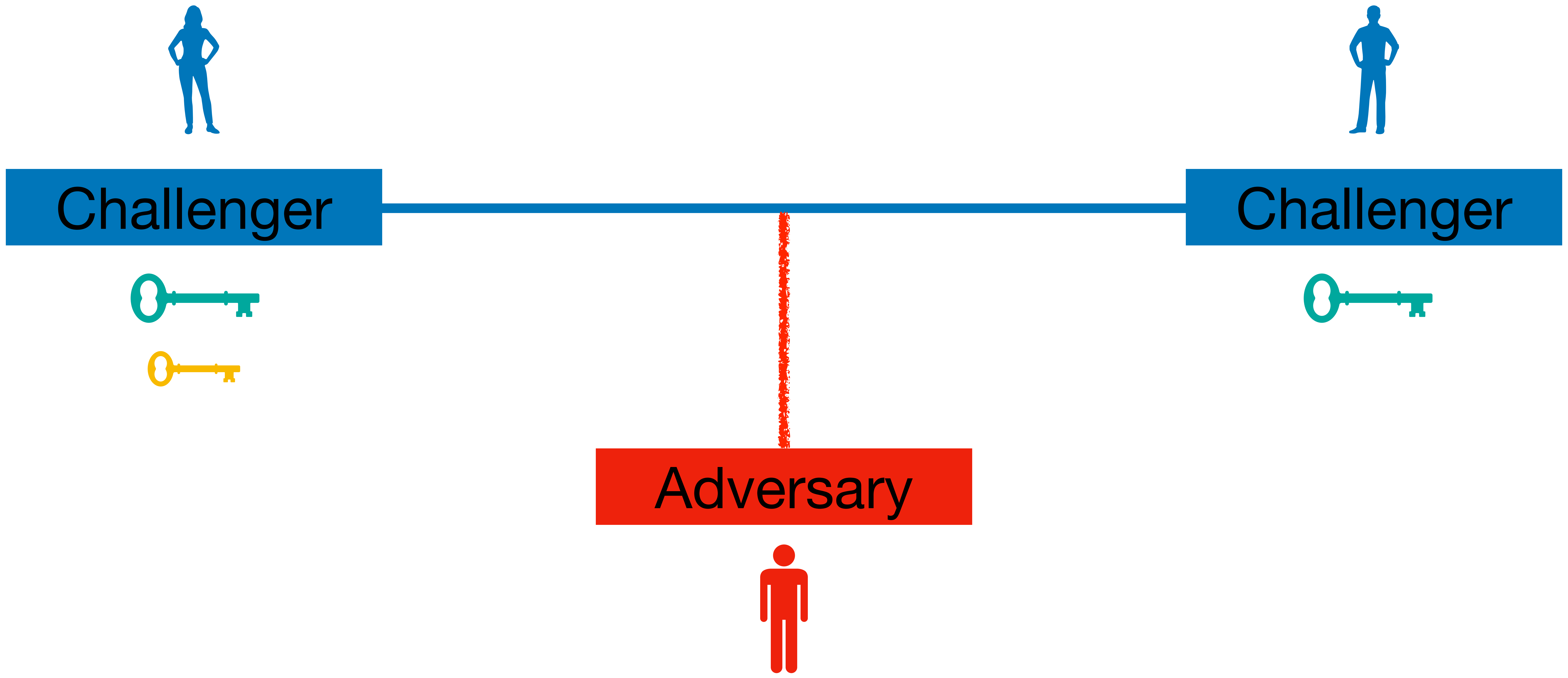
Security against Leakage



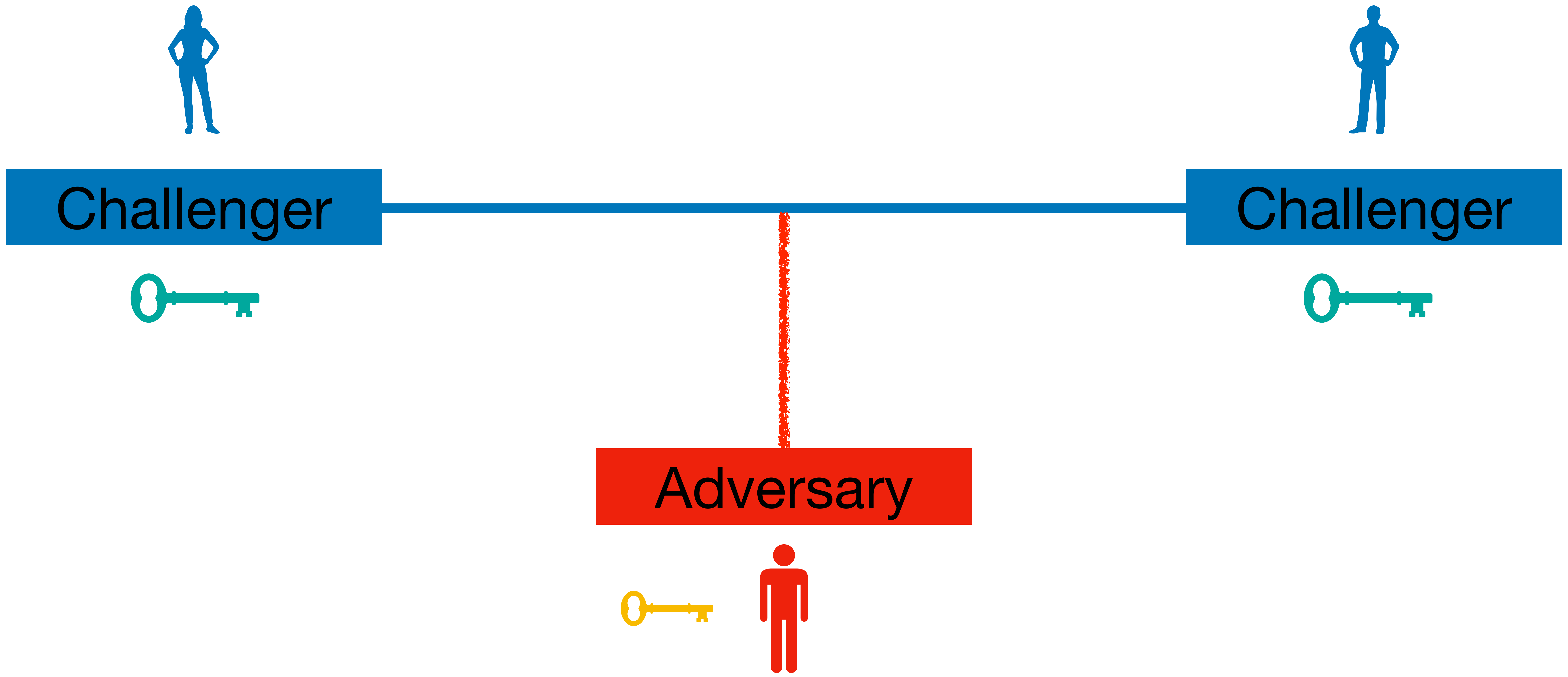
Security against Leakage



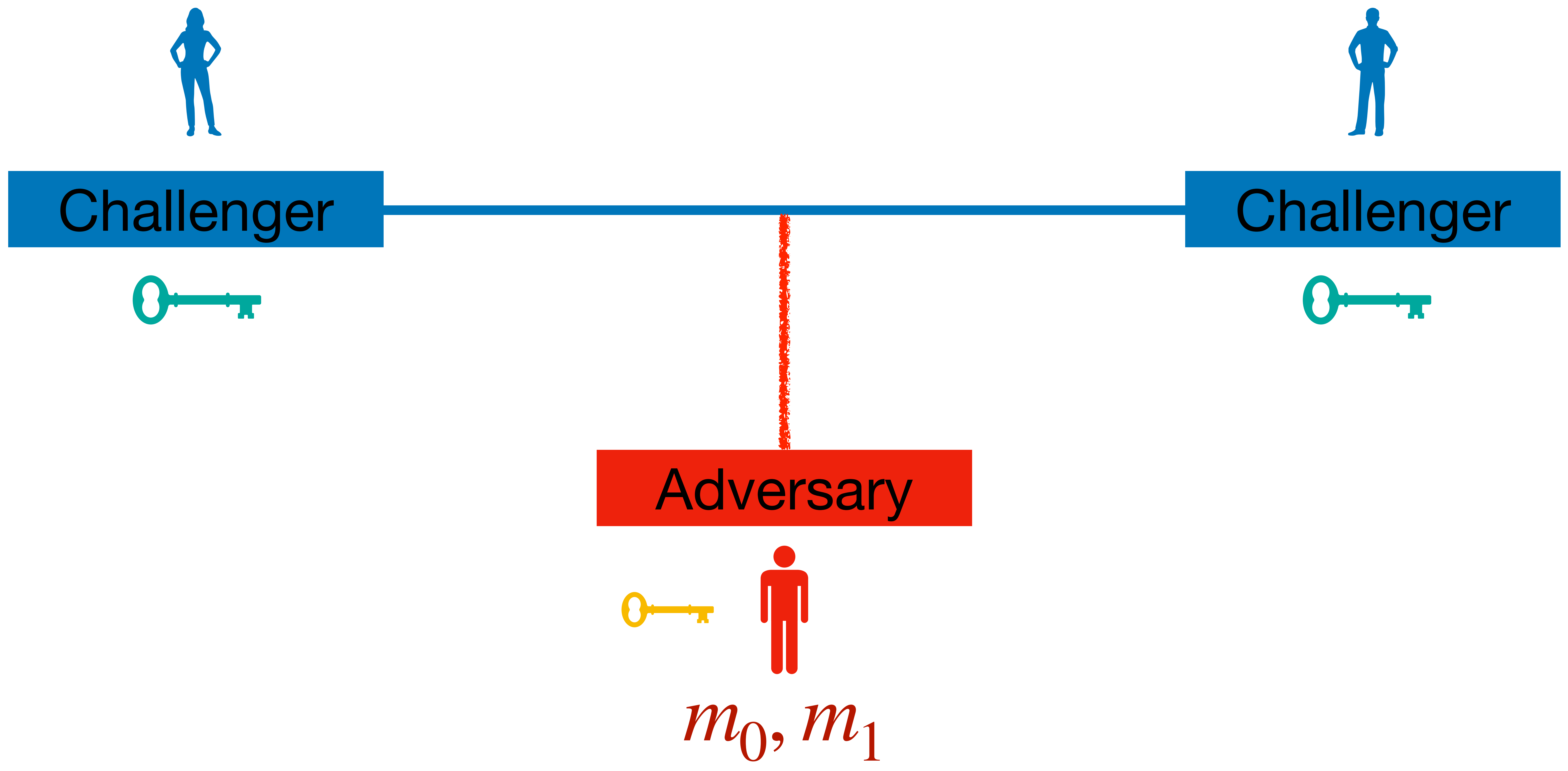
Security against Leakage



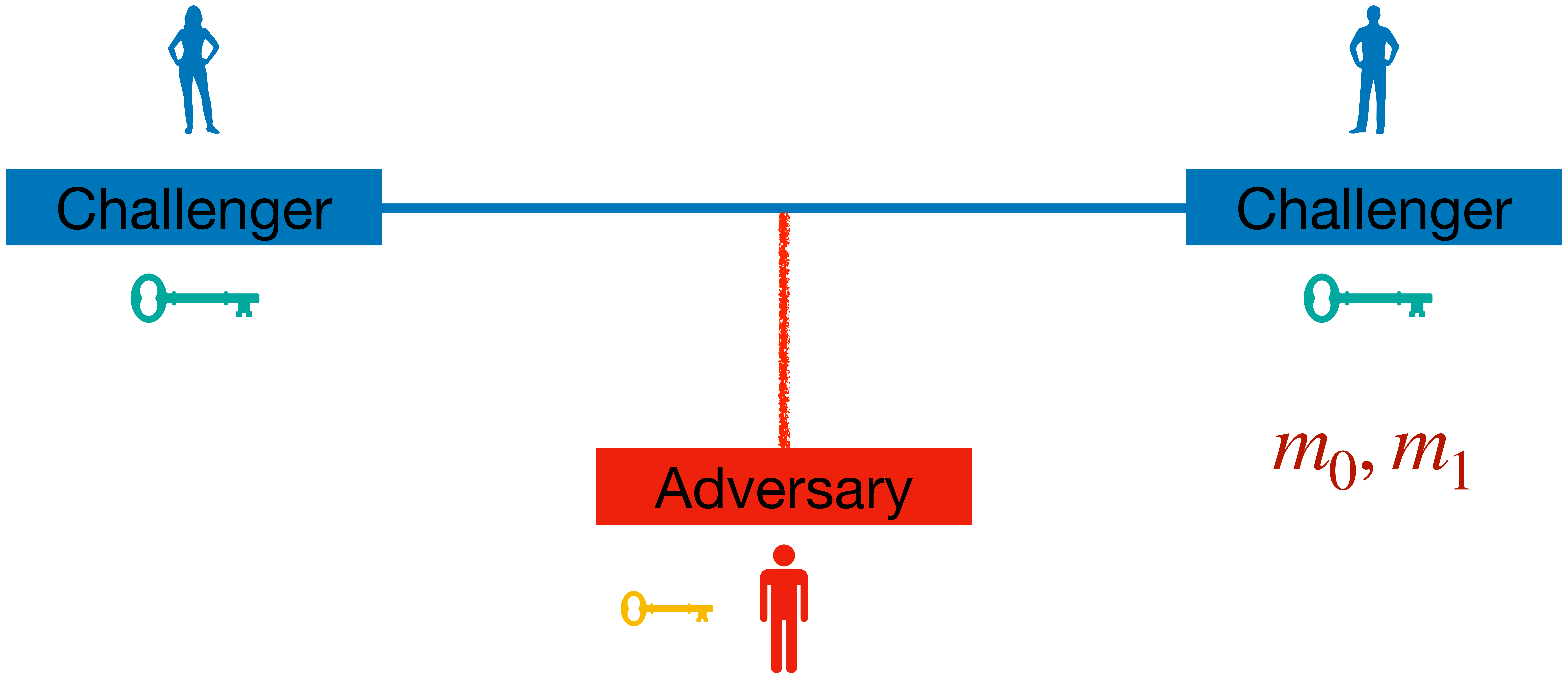
Security against Leakage



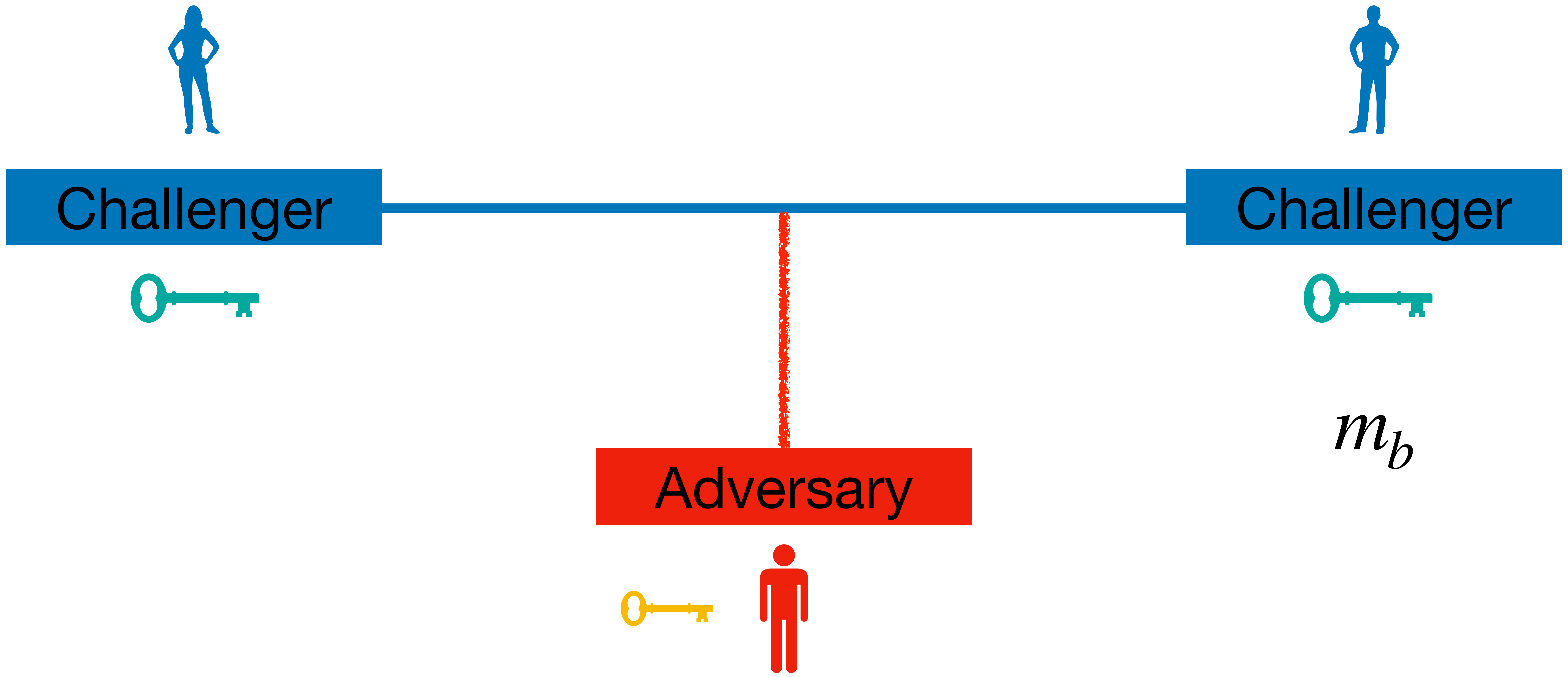
Security against Leakage



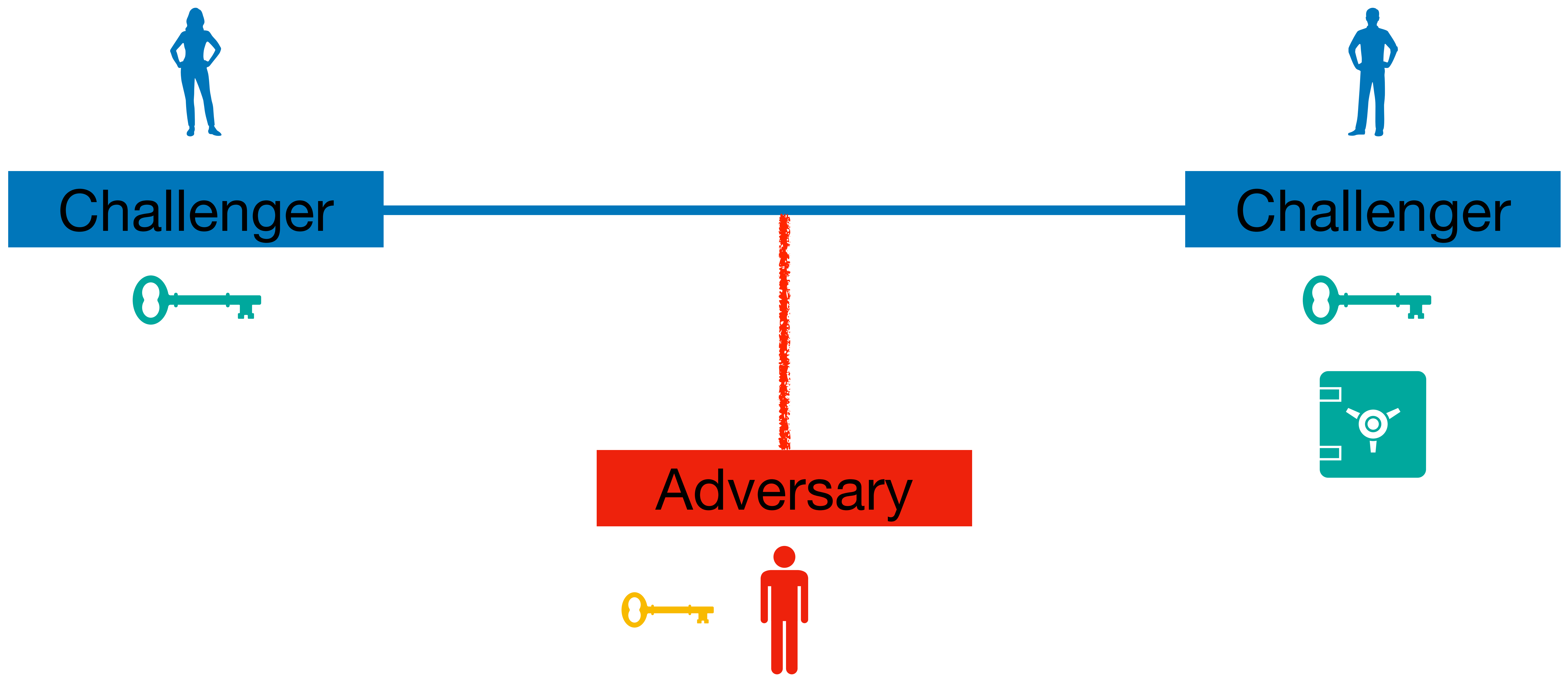
Security against Leakage



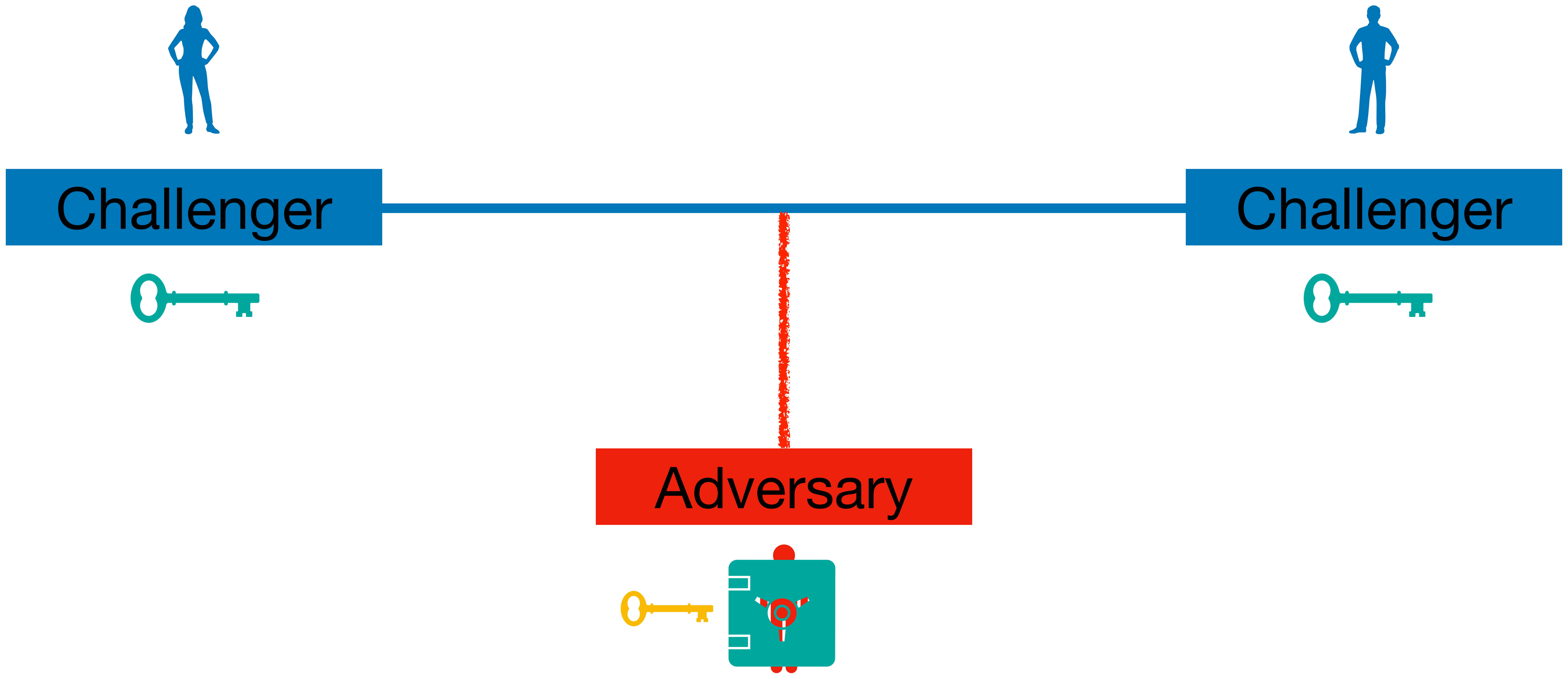
Security against Leakage



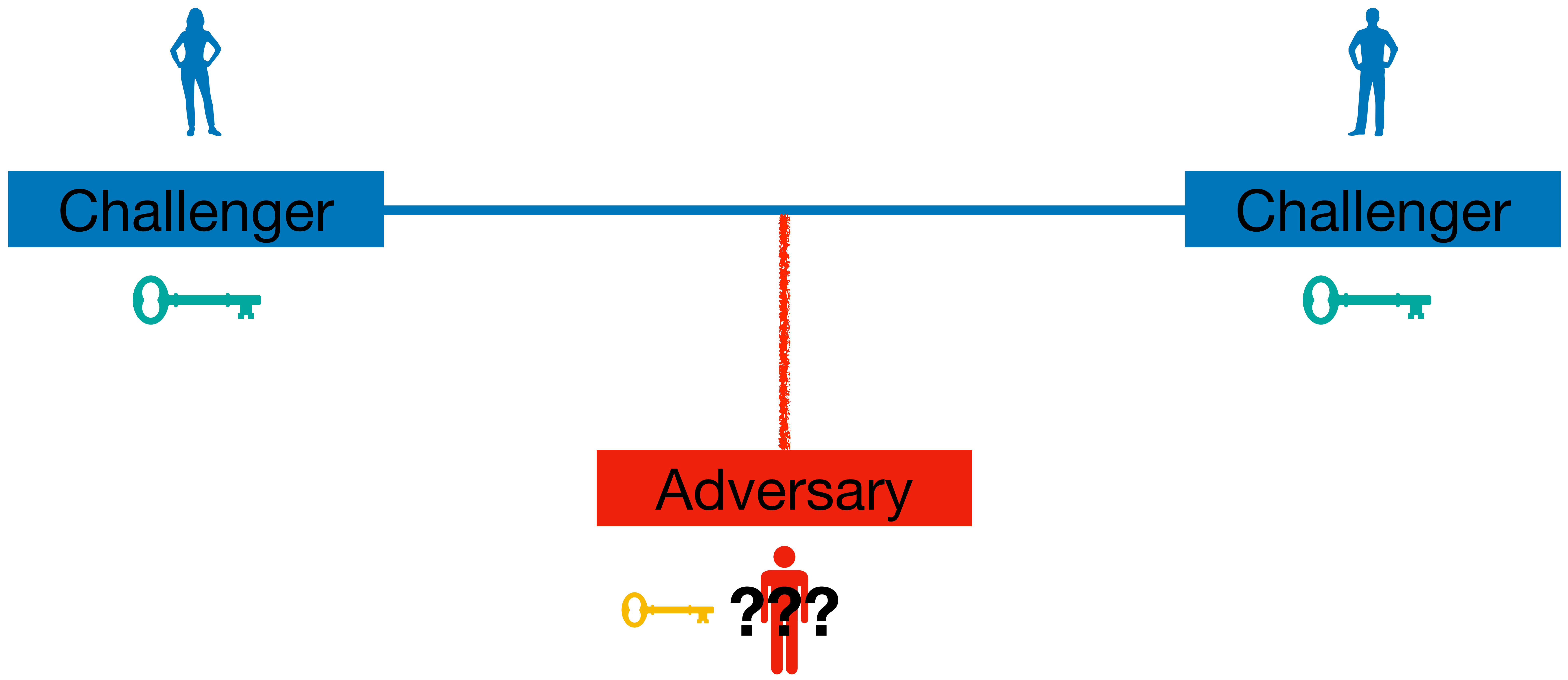
Security against Leakage



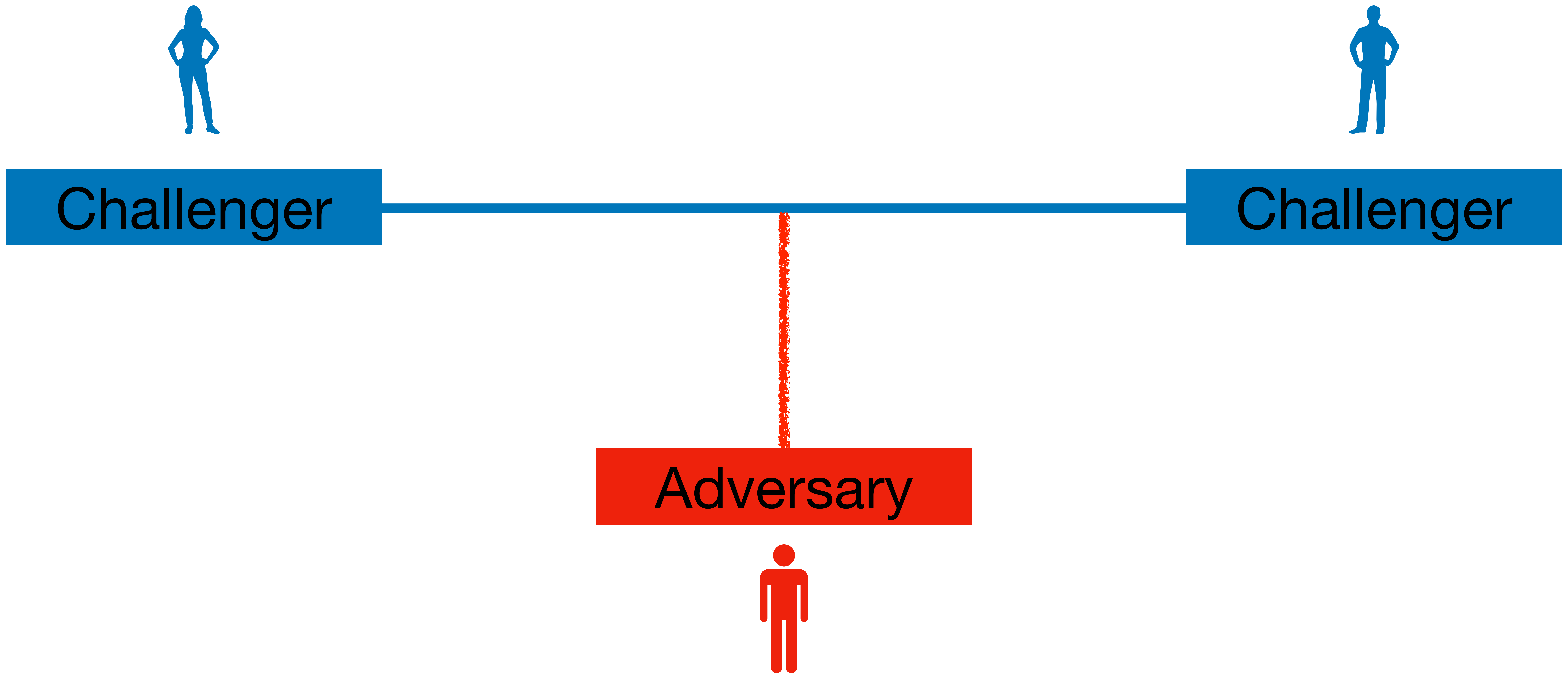
Security against Leakage



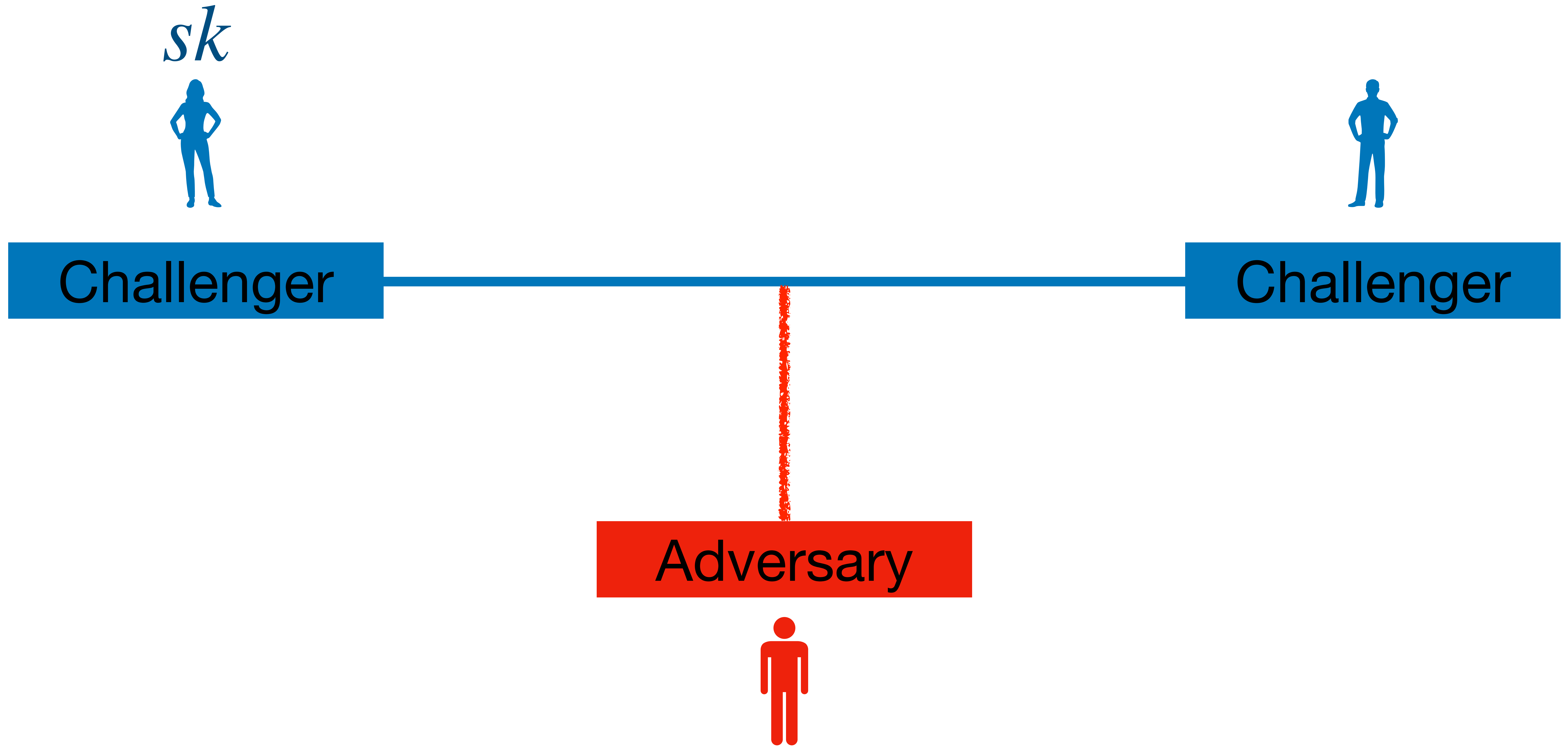
Security against Leakage



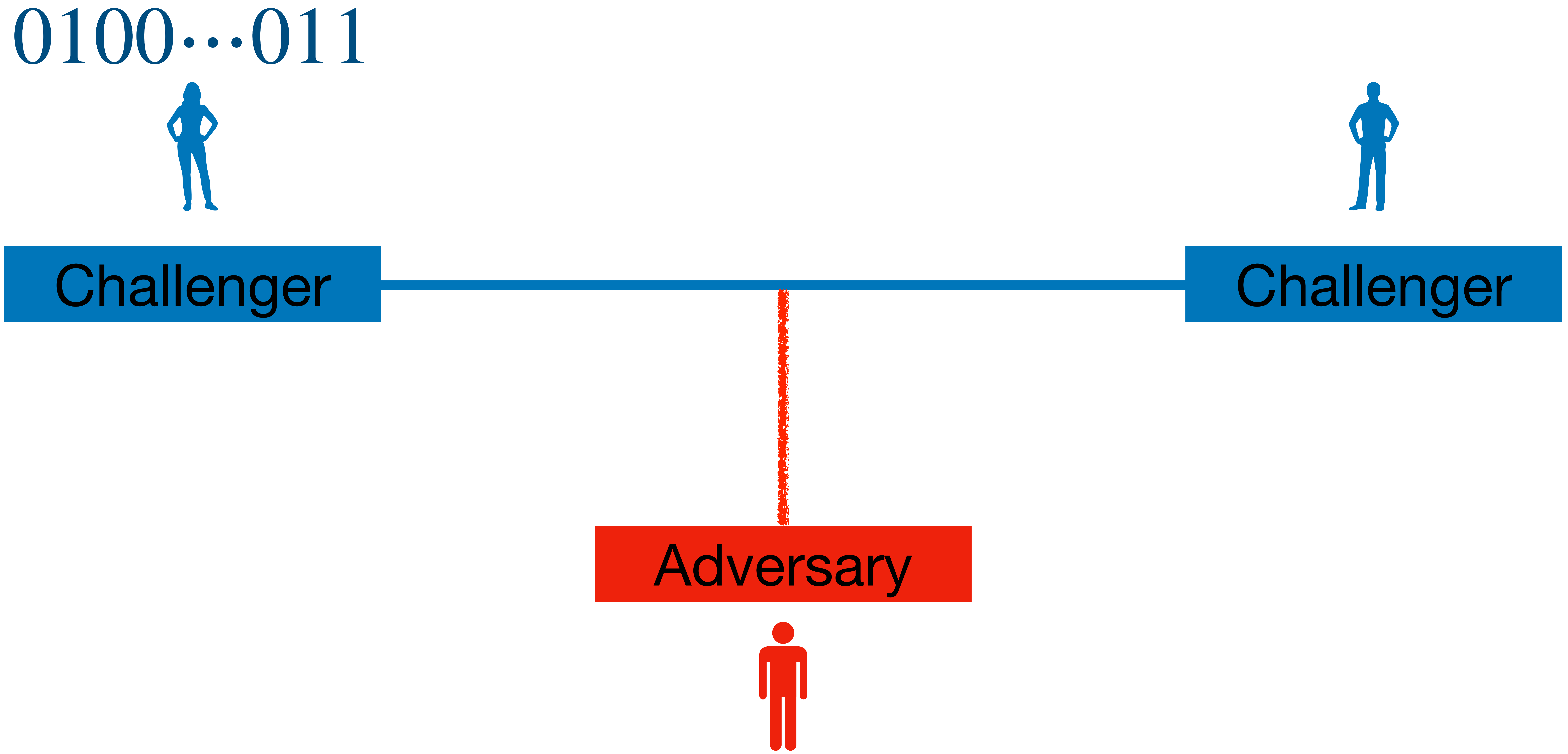
One Time Pad is not LR



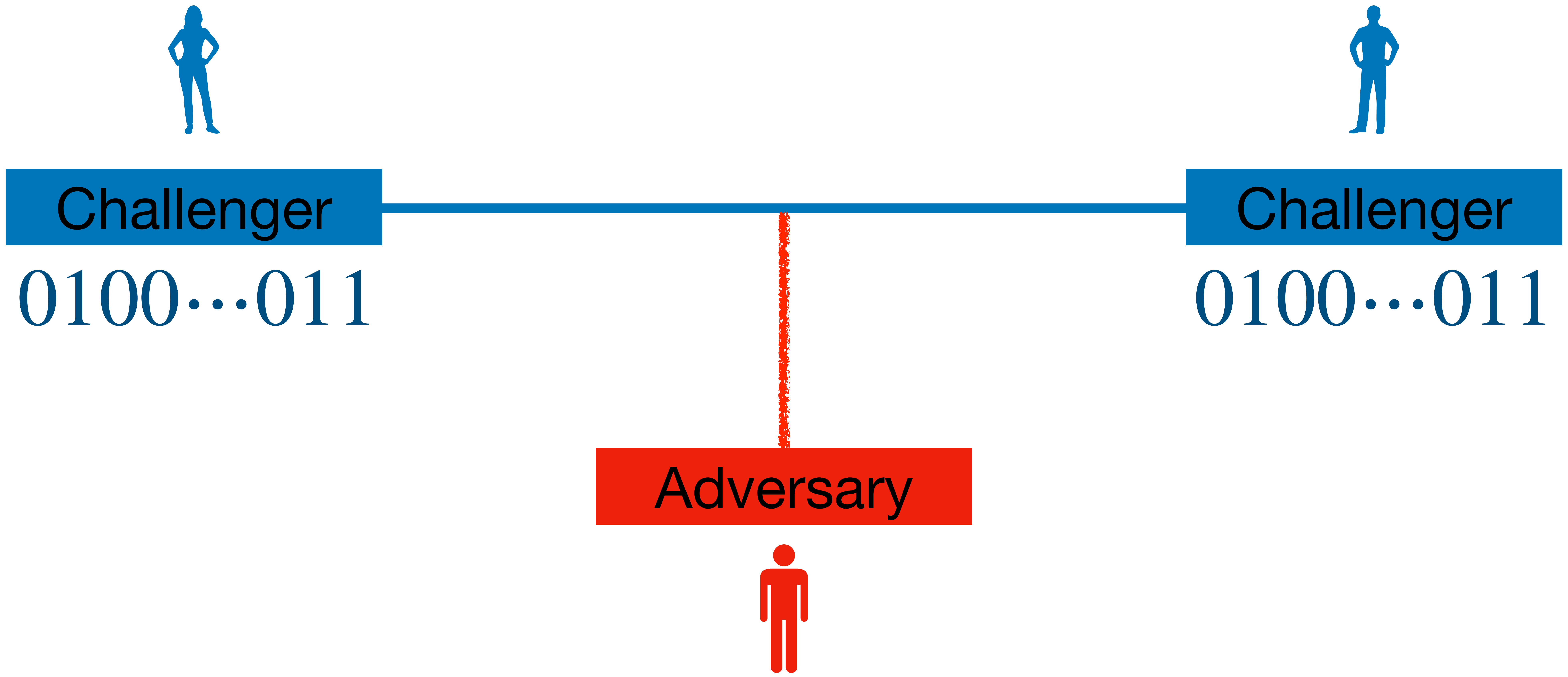
One Time Pad is not LR



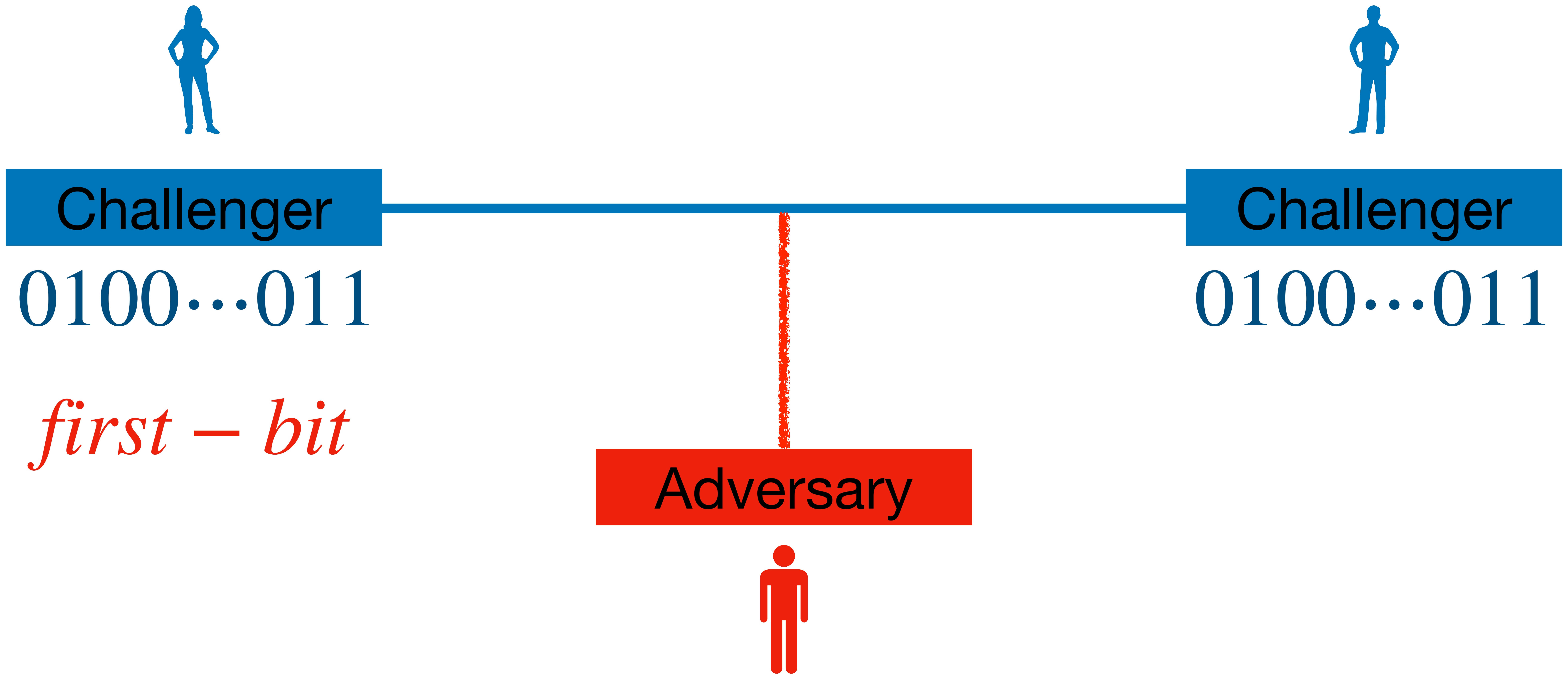
One Time Pad is not LR



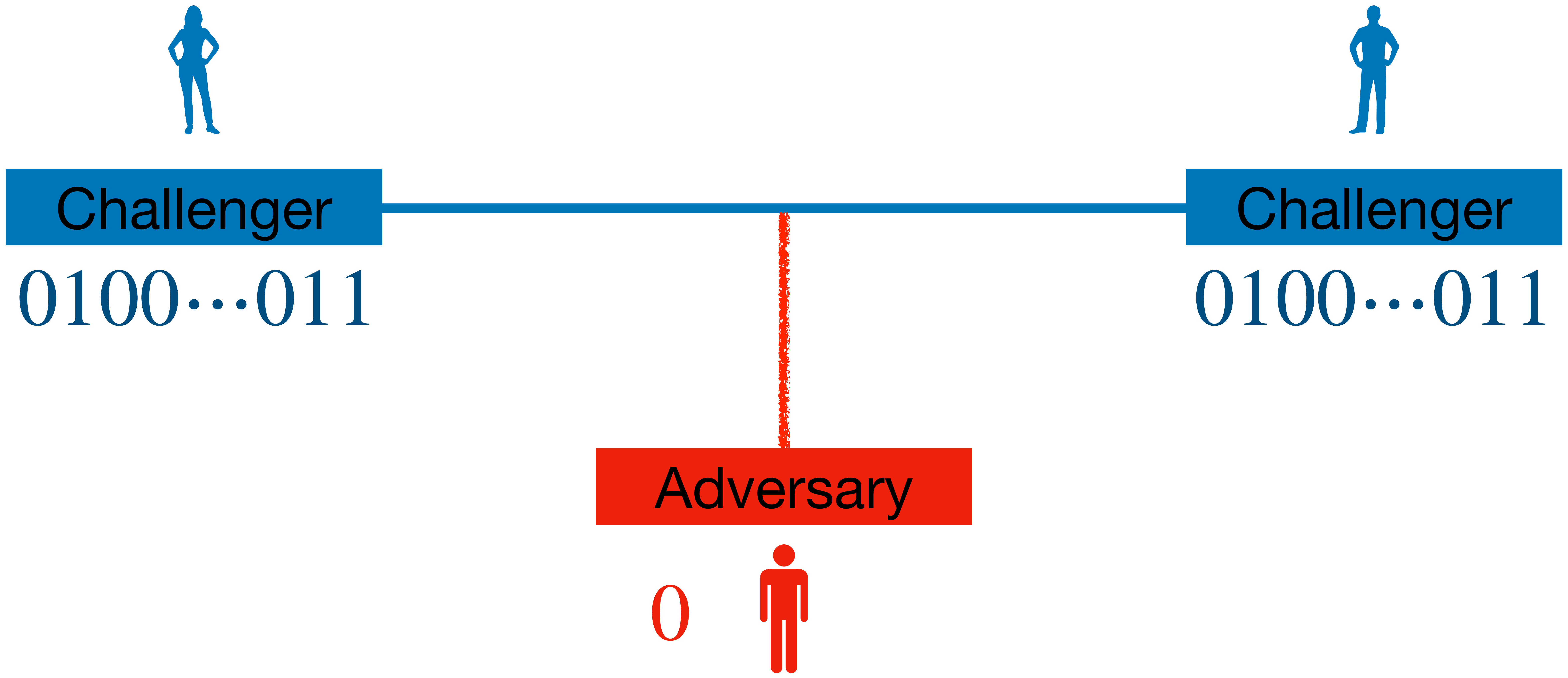
One Time Pad is not LR



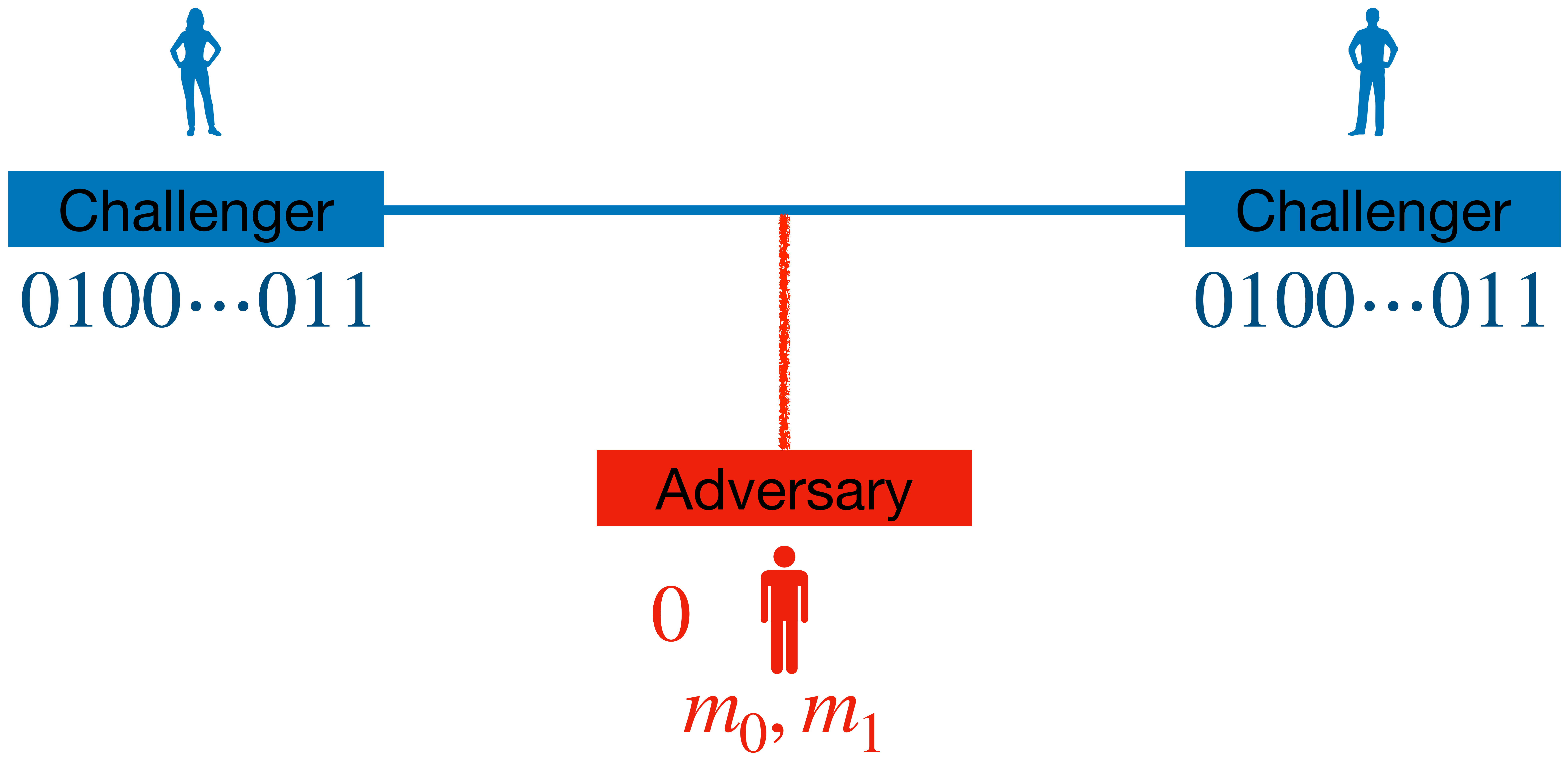
One Time Pad is not LR



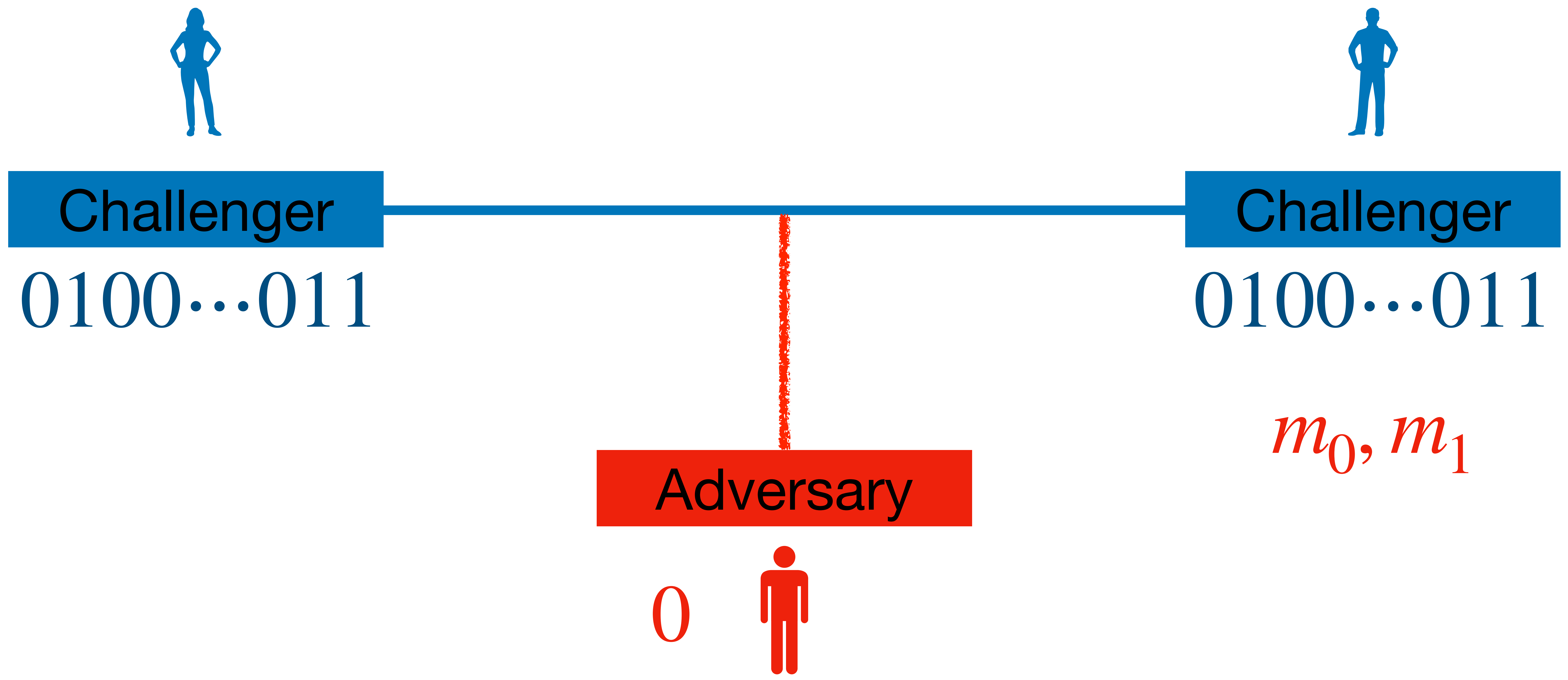
One Time Pad is not LR



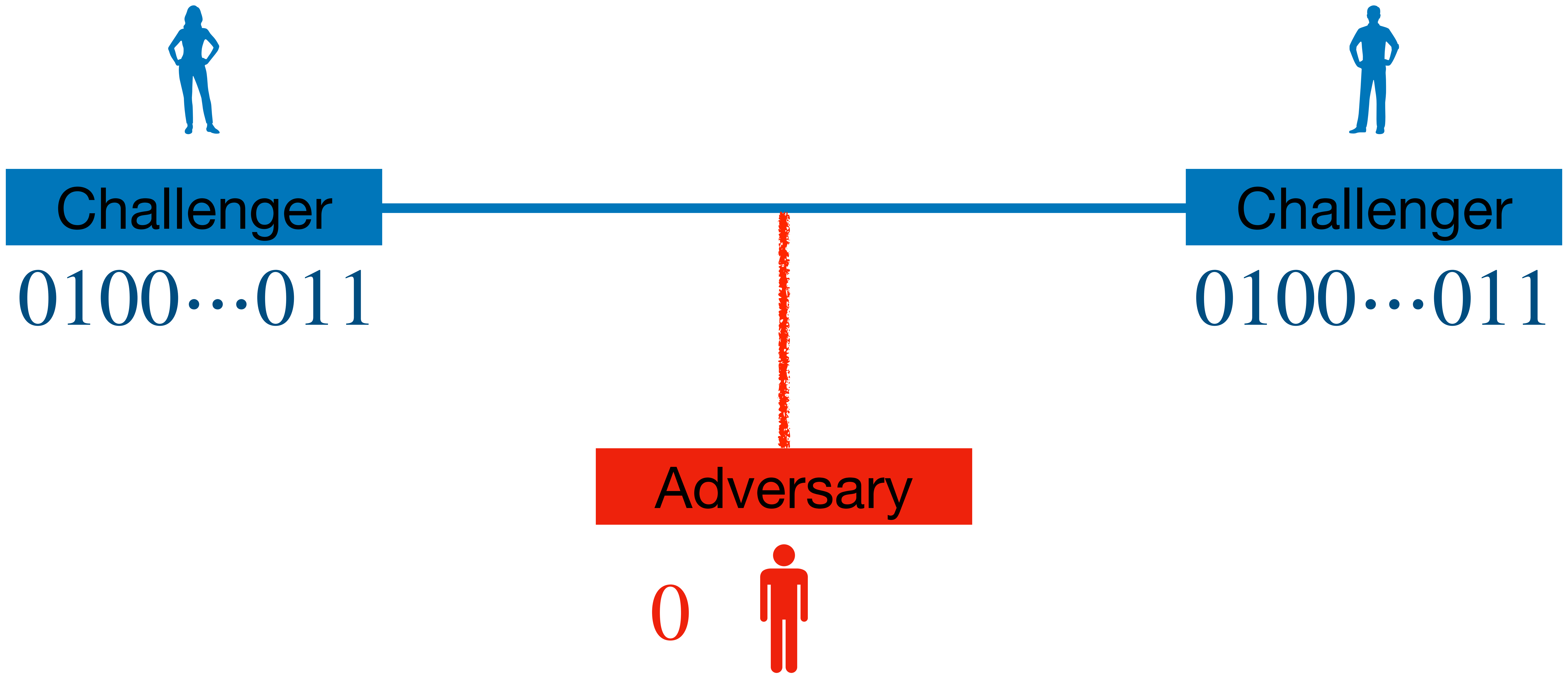
One Time Pad is not LR



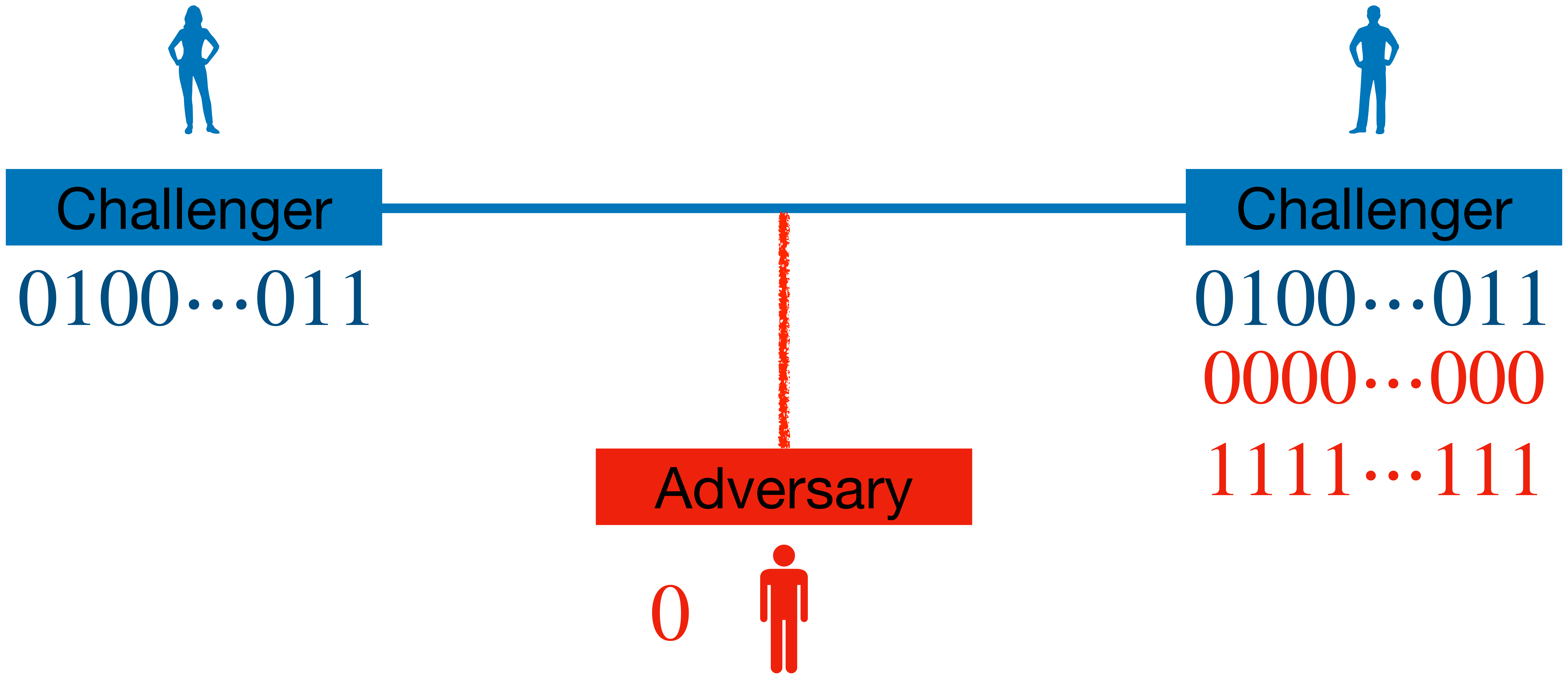
One Time Pad is not LR



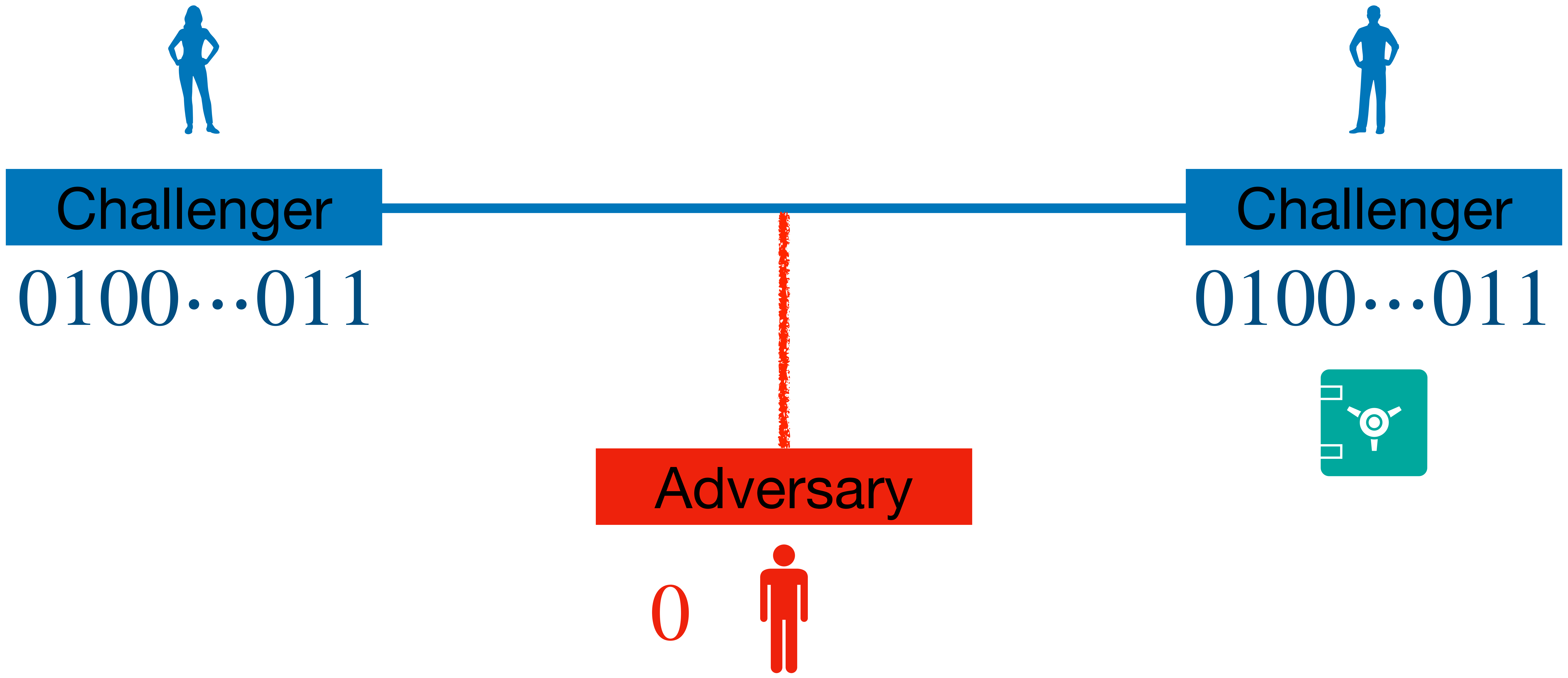
One Time Pad is not LR



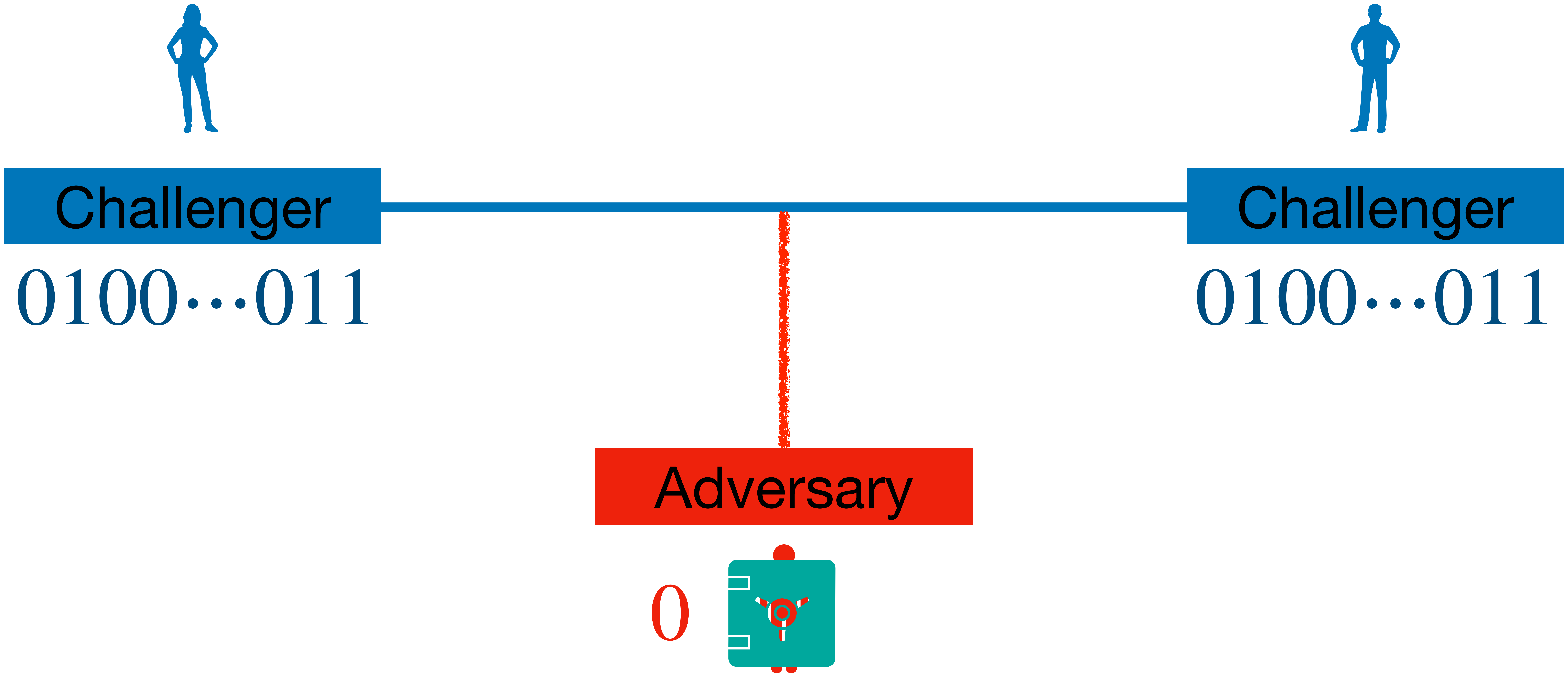
One Time Pad is not LR



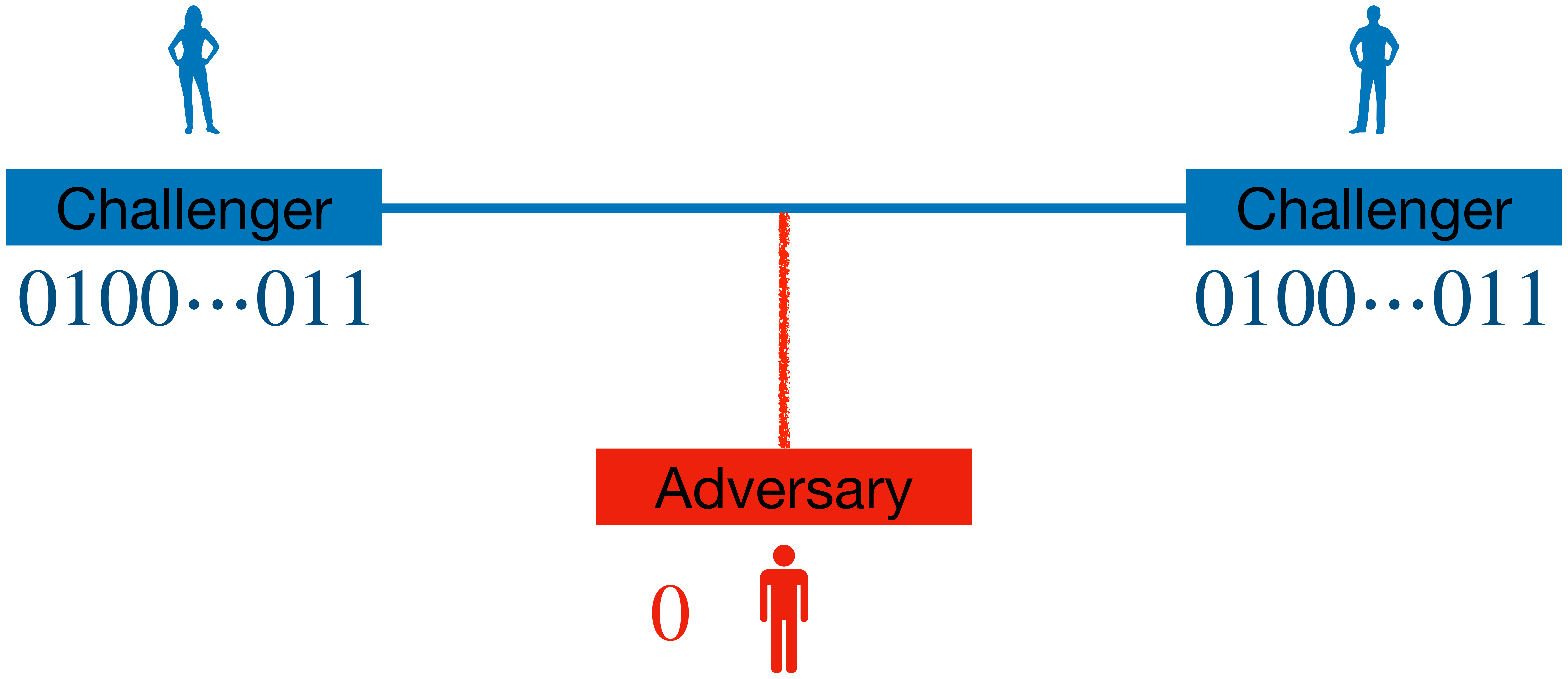
One Time Pad is not LR



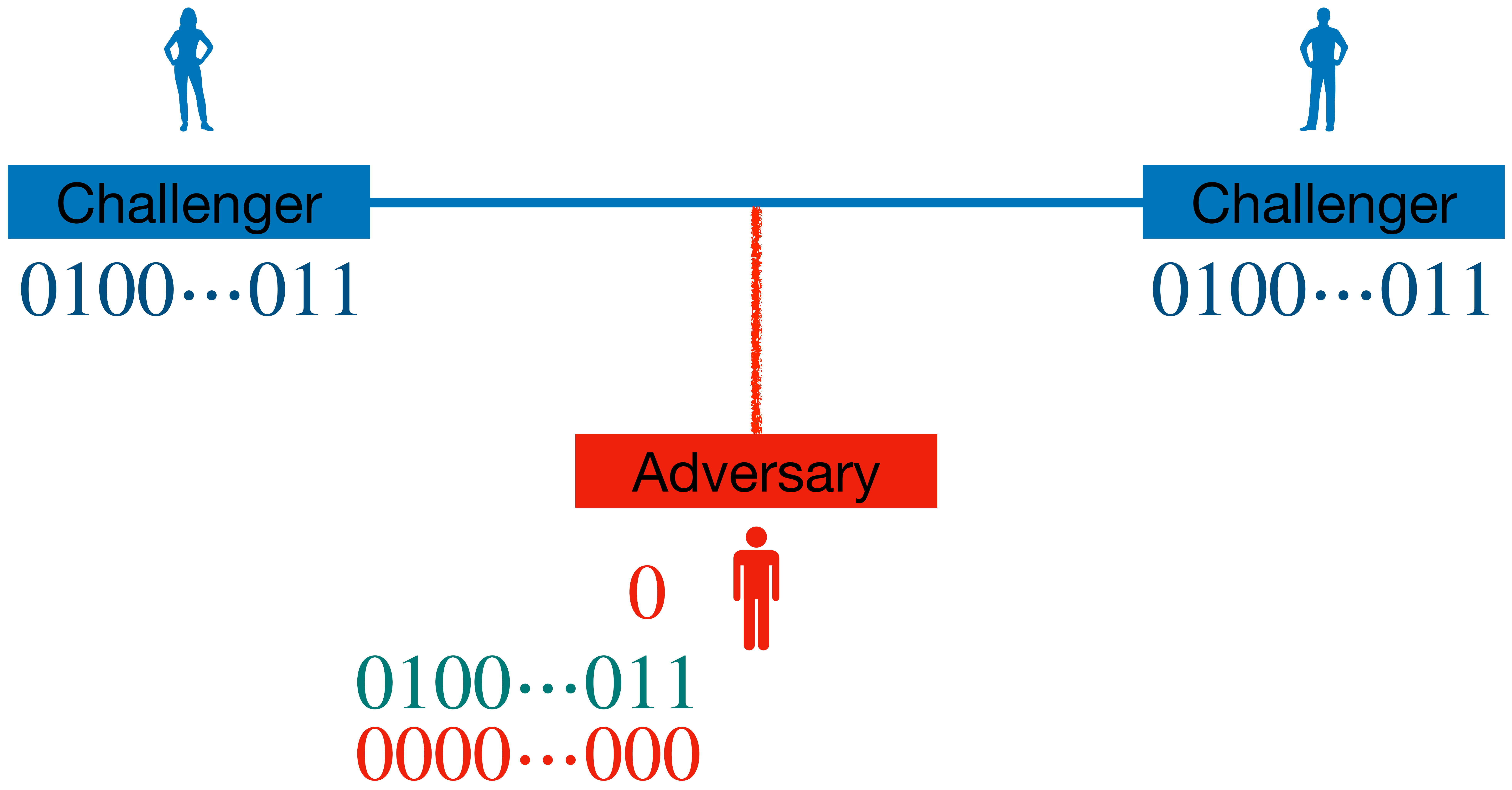
One Time Pad is not LR



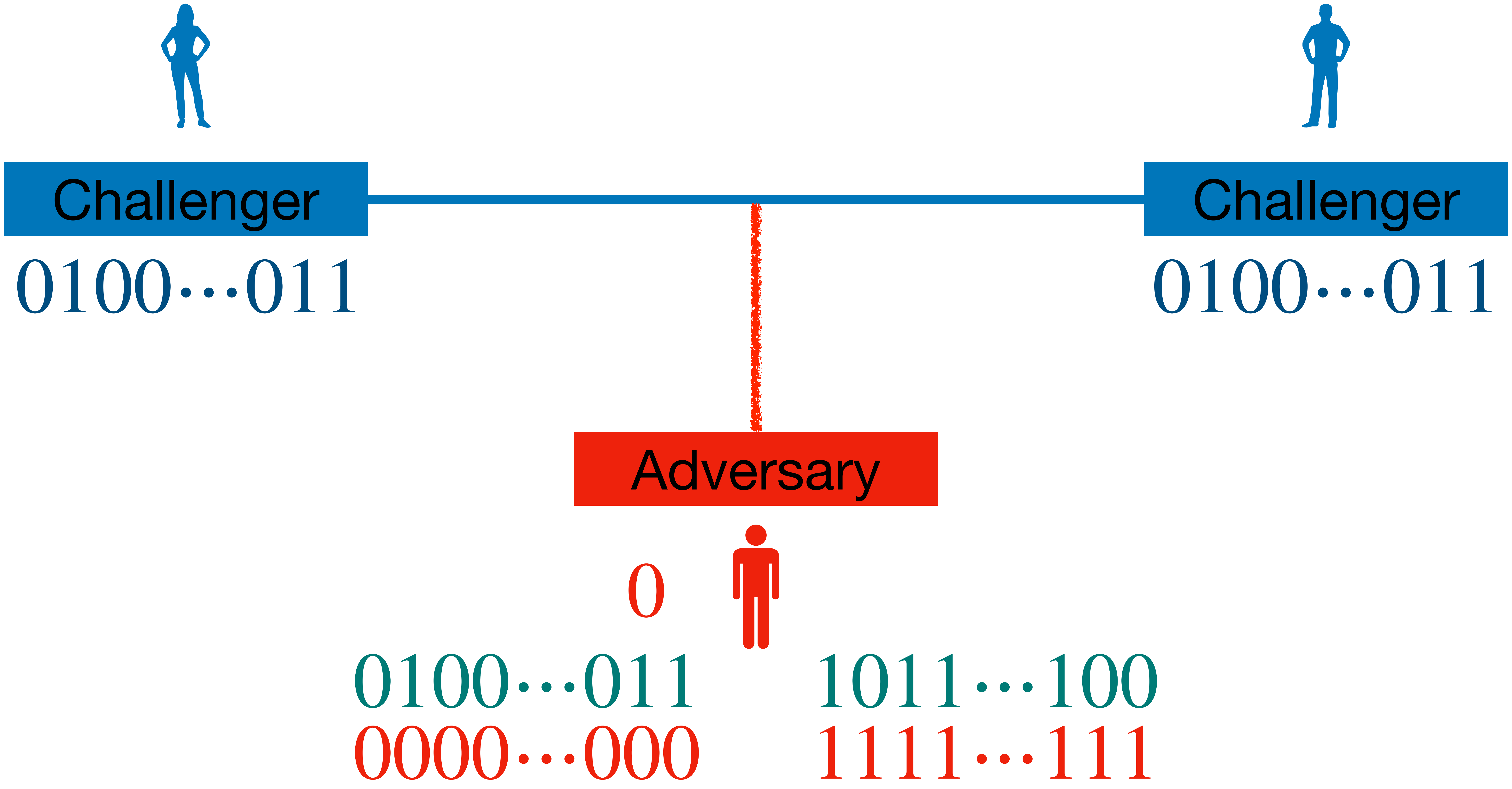
One Time Pad is not LR



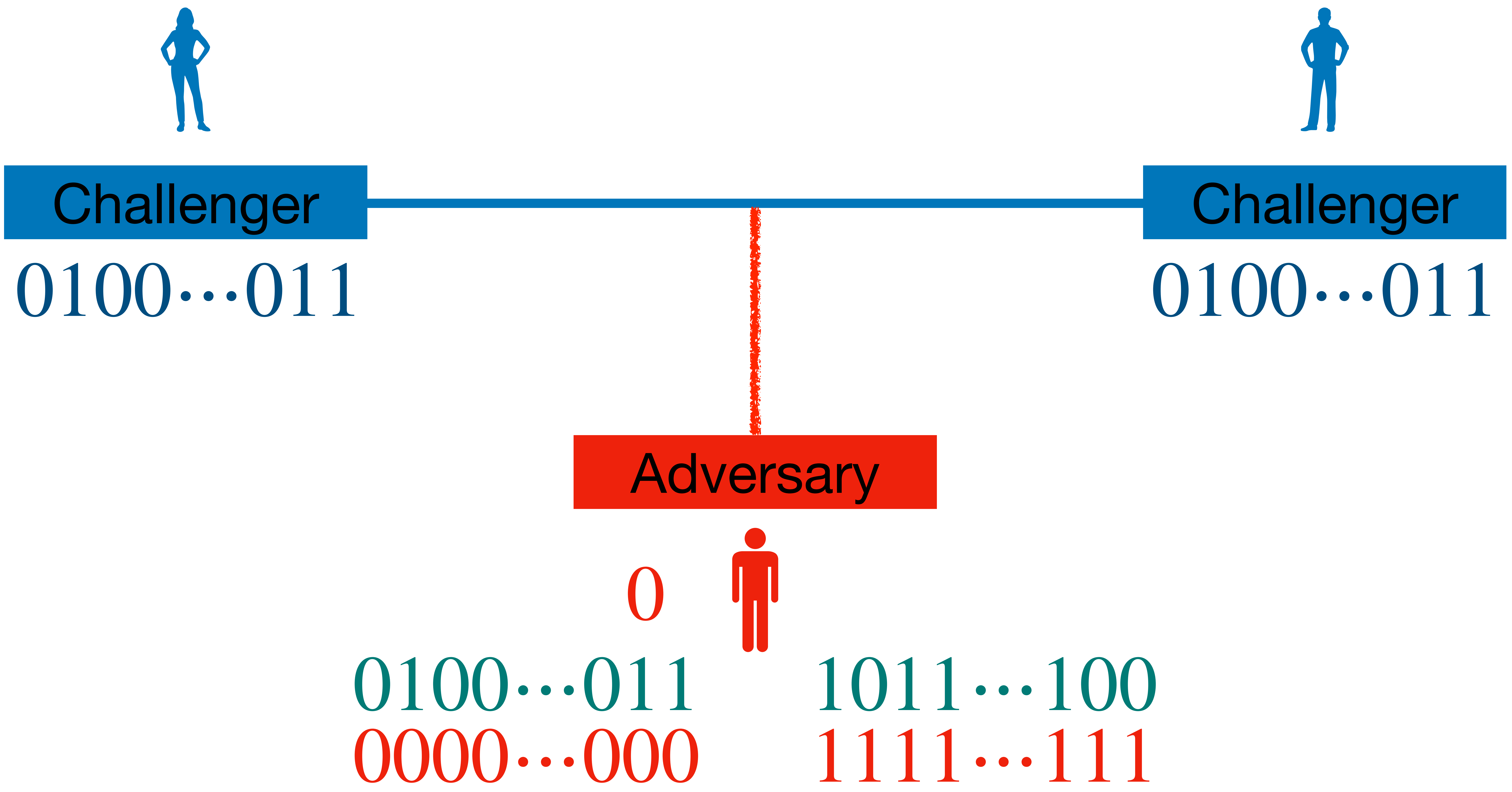
One Time Pad is not LR



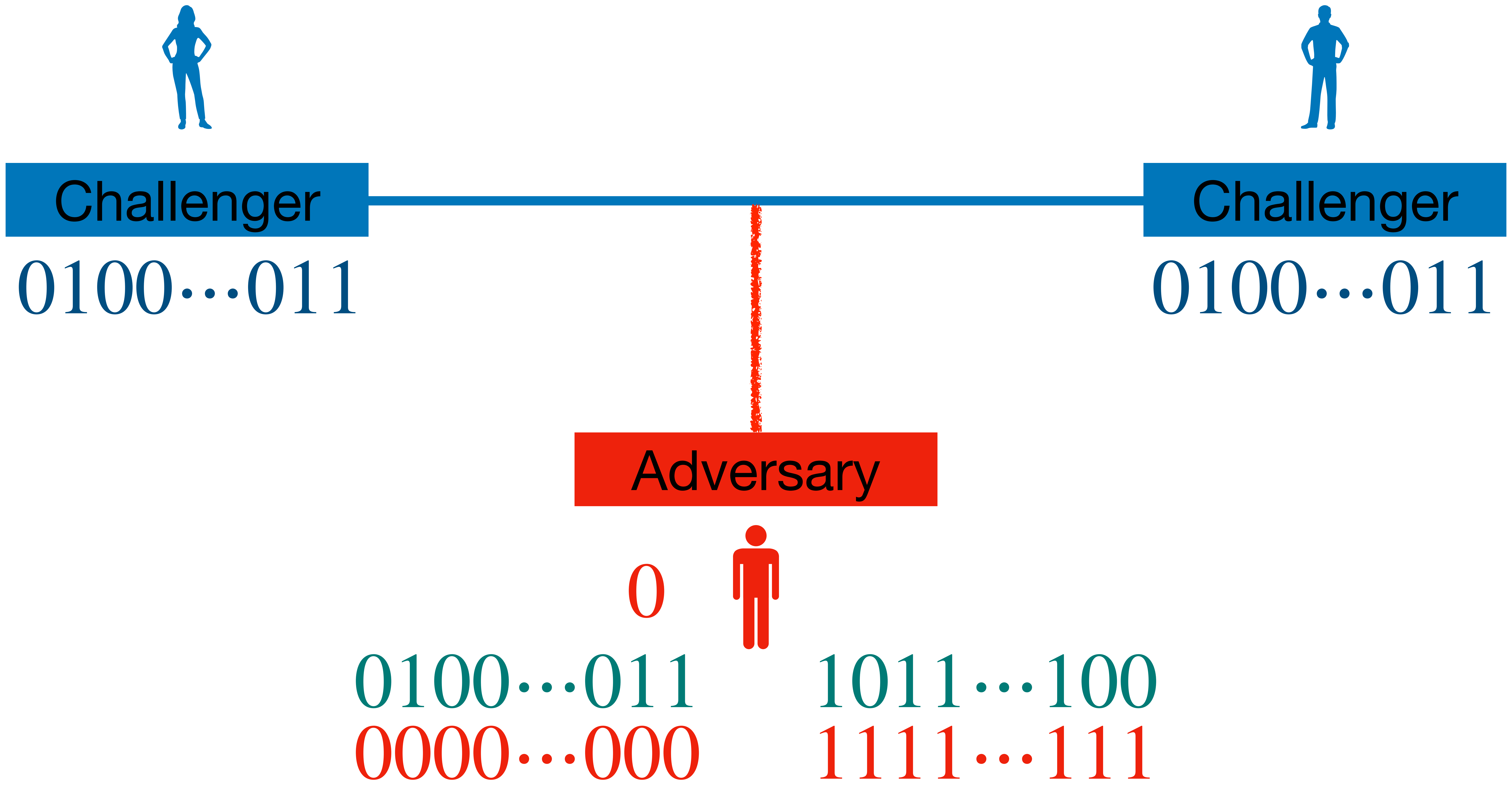
One Time Pad is not LR



One Time Pad is not LR



One Time Pad is not LR



LR One-Time Pad

LR One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\ell}$ (where $\ell > |m|$).

LR One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\ell}$ (where $\ell > |m|$).
- $Enc(sk, m) \rightarrow$

LR One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\ell}$ (where $\ell > |m|$).
- $Enc(sk, m) \rightarrow$
    1. Randomly generate $c_0 \in \{0,1\}^{\lambda}$.

LR One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\ell}$ (where $\ell > |m|$).
- $Enc(sk, m) \rightarrow$
    1. Randomly generate $c_0 \in \{0,1\}^{\lambda}$.
    2. Run $s = Ext_{c_0}(sk)$

LR One-Time Pad

- $Setup() \to$ Randomly generate $sk \in \{0,1\}^{\ell}$ (where $\ell > |m|$).
- $Enc(sk, m) \to$

$\boxed{Ext}$

1. Randomly generate $c_0 \in \{0,1\}^{\lambda}$.
2. Run $s = Ext_{c_0}(sk)$

LR One-Time Pad

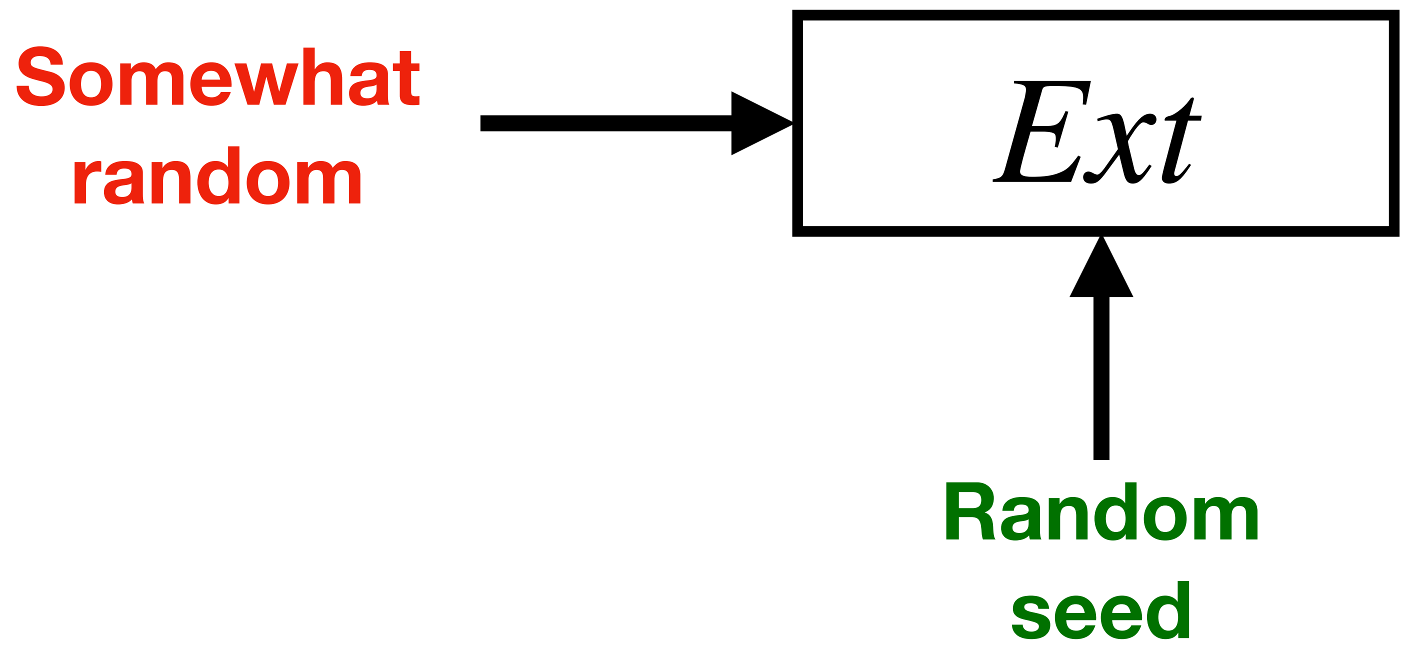
- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\ell}$ (where $\ell > |m|$).
- $Enc(sk, m) \rightarrow$

$$\begin{tikzcd} \text{Somewhat random} \arrow[r] & \boxed{Ext} \end{tikzcd}$$

1. Randomly generate $c_0 \in \{0,1\}^{\lambda}$.
2. Run $s = Ext_{c_0}(sk)$

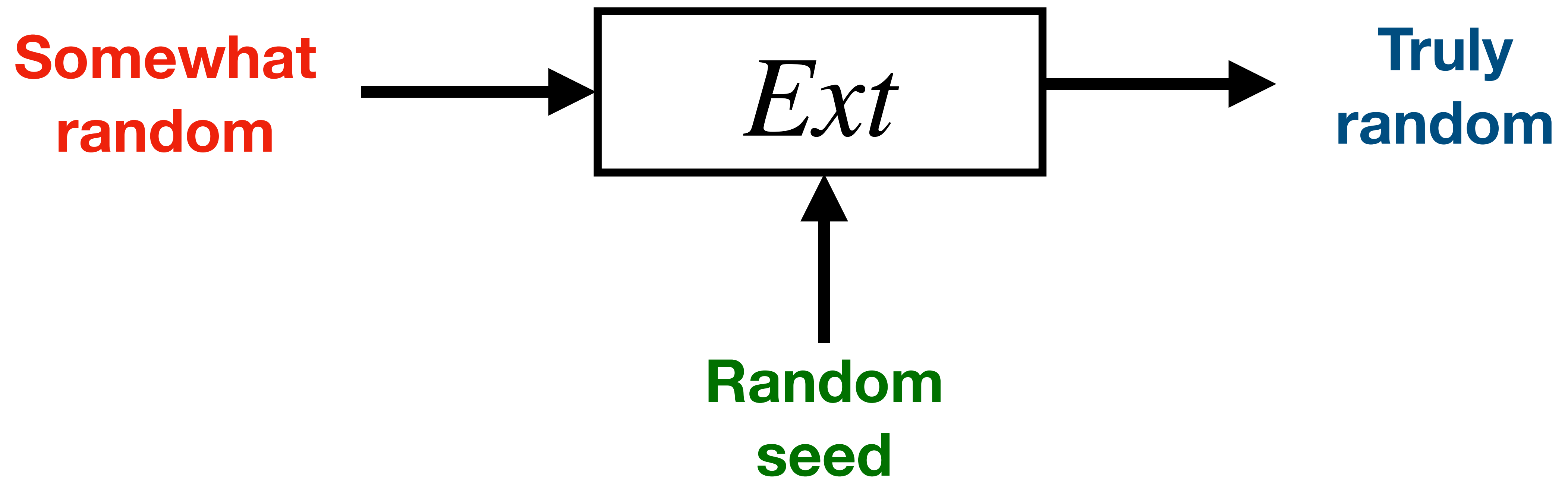
LR One-Time Pad

- $Setup() \to$ Randomly generate $sk \in \{0,1\}^\ell$ (where $\ell > |m|$).
- $Enc(sk, m) \to$
  1. Randomly generate $c_0 \in \{0,1\}^\lambda$.
  2. Run $s = Ext_{c_0}(sk)$



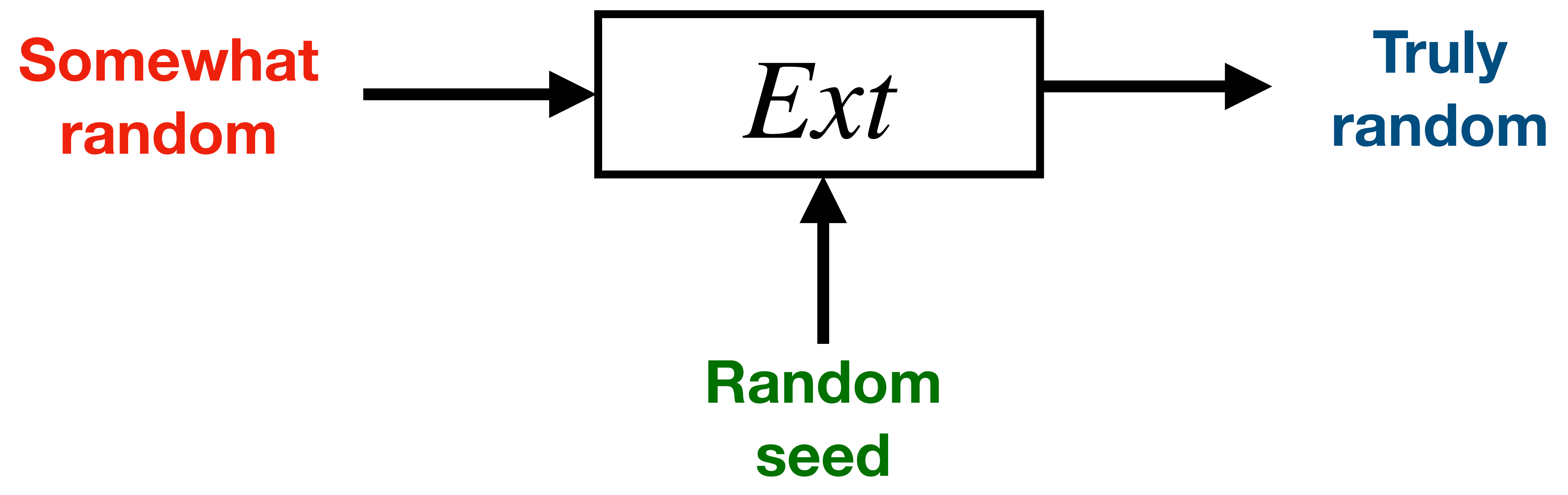
LR One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^\ell$ (where $\ell > |m|$).
- $Enc(sk, m) \rightarrow$
  1. Randomly generate $c_0 \in \{0,1\}^\lambda$.
  2. Run $s = Ext_{c_0}(sk)$



LR One-Time Pad

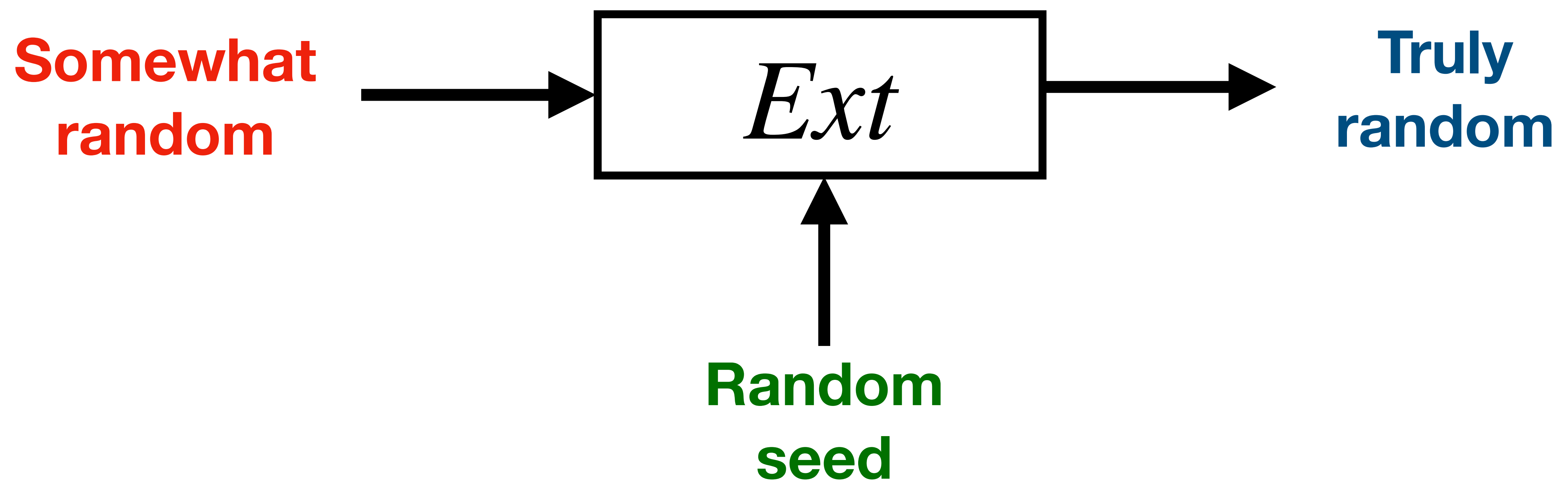
- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\ell}$ (where $\ell > |m|$).
- $Enc(sk, m) \rightarrow$



1. Randomly generate $c_0 \in \{0,1\}^{\lambda}$.
2. Run $s = Ext_{c_0}(sk)$
3. Compute $c_1 = s \oplus m$ (one time pad)

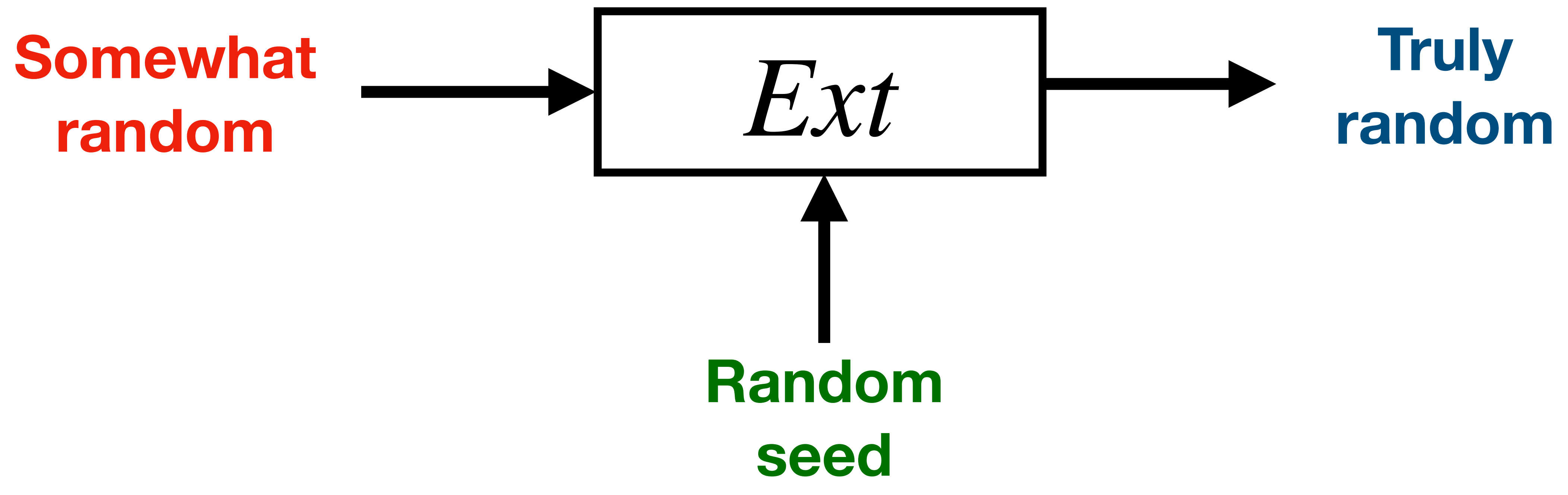
LR One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\ell}$ (where $\ell > |m|$).
- $Enc(sk, m) \rightarrow$
  1. Randomly generate $c_0 \in \{0,1\}^{\lambda}$.
  2. Run $s = Ext_{c_0}(sk)$
  3. Compute $c_1 = s \oplus m$ (one time pad)
  4. Return (c_0, c_1)



LR One-Time Pad

- $Setup() \to$ Randomly generate $sk \in \{0,1\}^{\ell}$ (where $\ell > |m|$).
- $Enc(sk, m) \to$

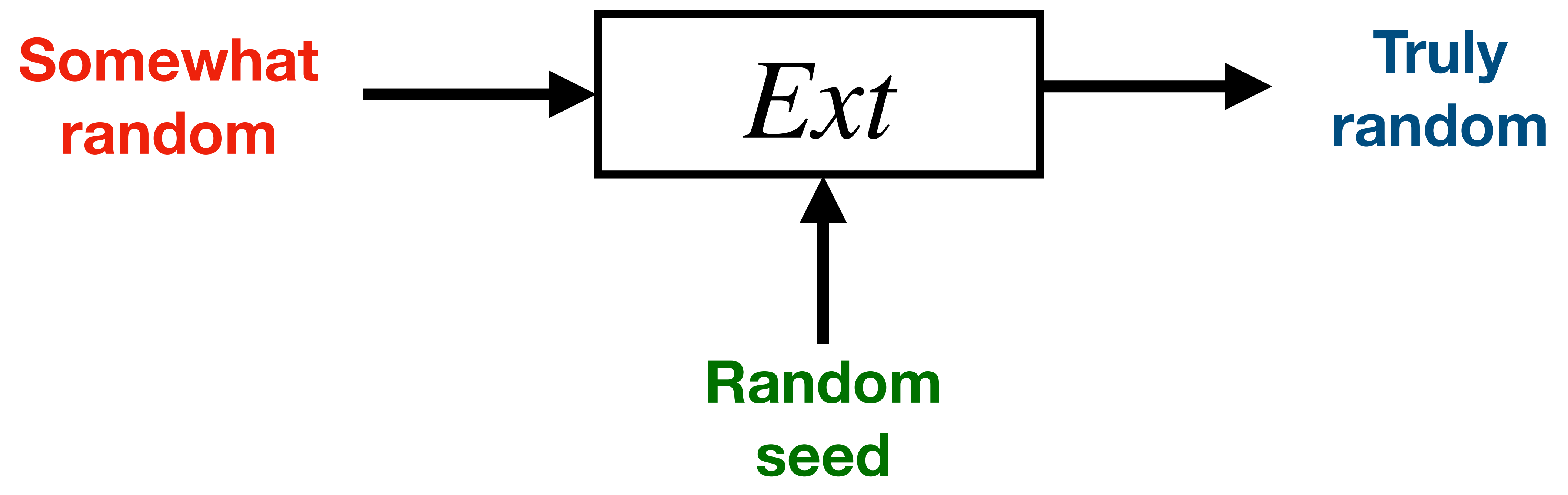


  1. Randomly generate $c_0 \in \{0,1\}^{\lambda}$.
  2. Run $s = Ext_{c_0}(sk)$
  3. Compute $c_1 = s \oplus m$ (one time pad)
  4. Return (c_0, c_1)

- $Dec(sk, ct) \to$ Run $s = Ext_{c_0}(sk)$ and return $s \oplus ct$.

LR One-Time Pad

- $Setup() \to$ Randomly generate $sk \in \{0,1\}^\ell$ (where $\ell > |m|$).
- $Enc(sk, m) \to$
  1. Randomly generate $c_0 \in \{0,1\}^\lambda$.
  2. Run $s = Ext_{c_0}(sk)$
  3. Compute $c_1 = s \oplus m$ (one time pad)
  4. Return (c_0, c_1)
- $Dec(sk, ct) \to$ Run $s = Ext_{c_0}(sk)$ and return $s \oplus ct$.



$$(c_0, Ext_{c_0}(sk), f(sk)) \approx (c_0, U, f(sk))$$

Leakage Resilient Schemes

Leakage Resilient Schemes

- [Canetti et al. 00] and [Dodis et al. 01] gave construction where f returns bits of sk.

Leakage Resilient Schemes

- [Canetti et al. 00] and [Dodis et al. 01] gave construction where f returns bits of sk.
- [Dziembowski06], [Di Crescenzo et al.06], [Akavia et al.09], etc. considered arbitrary function f.

Leakage Resilient Schemes

- [Canetti et al. 00] and [Dodis et al. 01] gave construction where f returns bits of sk.
- [Dziembowski06], [Di Crescenzo et al.06], [Akavia et al.09], etc. considered arbitrary function f.
- Other works include [Dodis et al.09], [Brakerski et al.10], [Dodis et al.10], [Faonio et al.15] and many more.

Can the entire secret key be exposed?

Can the entire secret key be exposed?

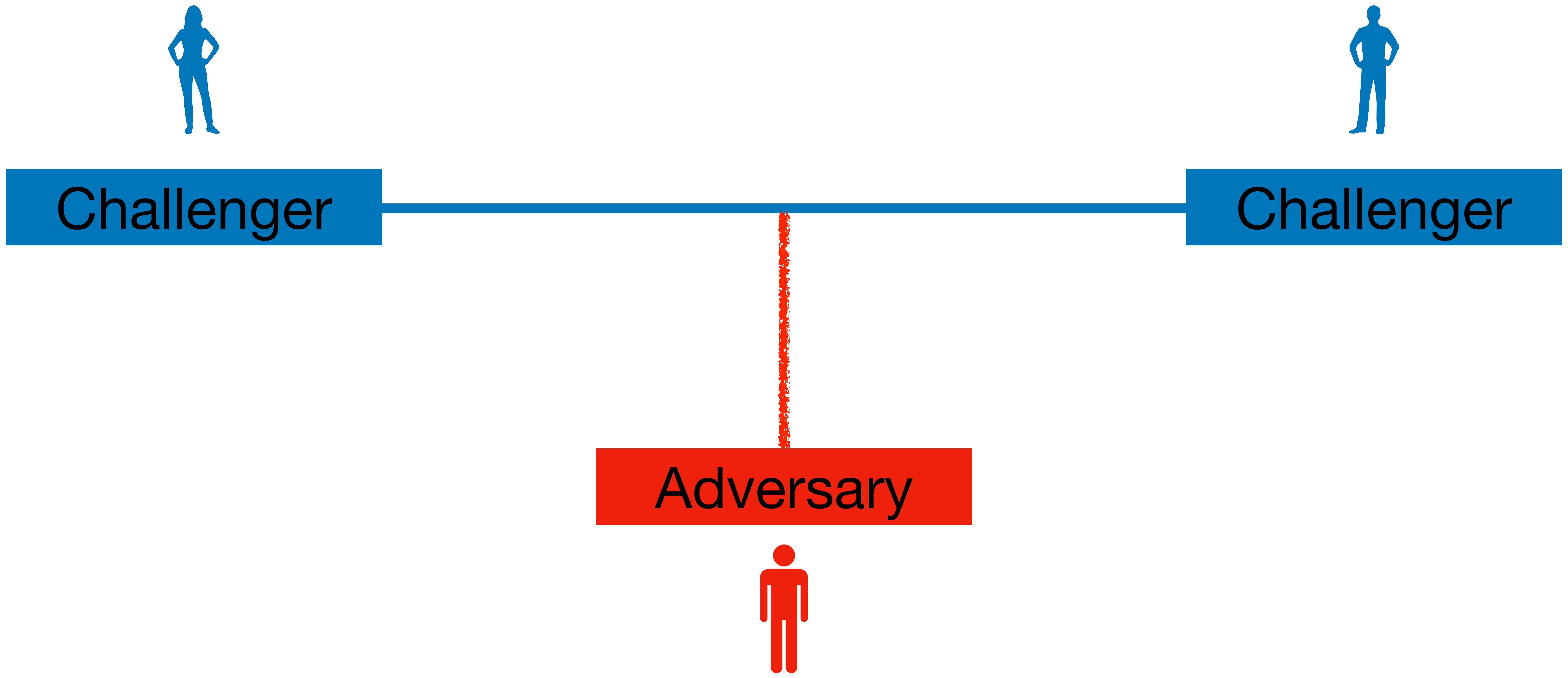
- Does not make sense if entire secret key and ciphertext is given to adversary.

Can the entire secret key be exposed?

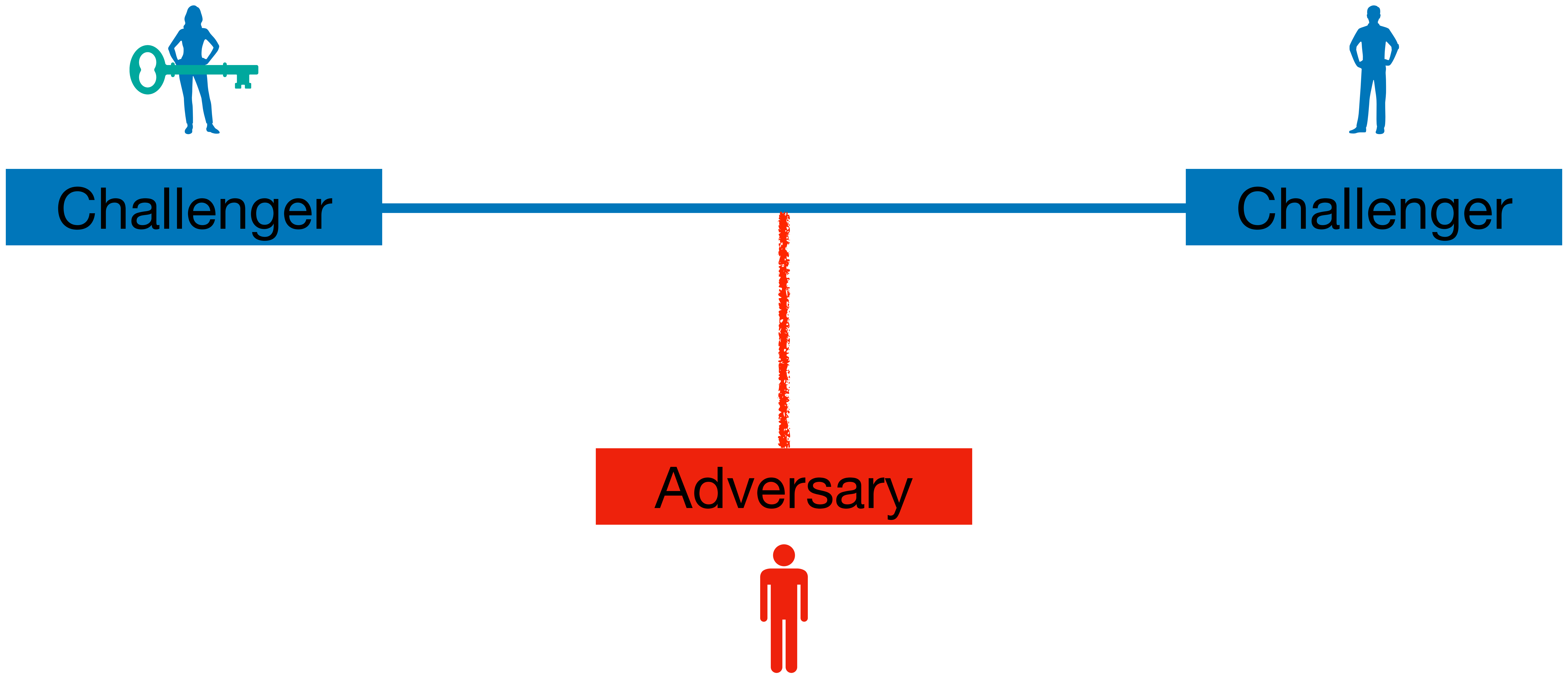
- Does not make sense if entire secret key and ciphertext is given to adversary.
- May be possible for adversary to attain the entire secret key but store only a part of the ciphertext. For example, cloud storage.

Incompressibility

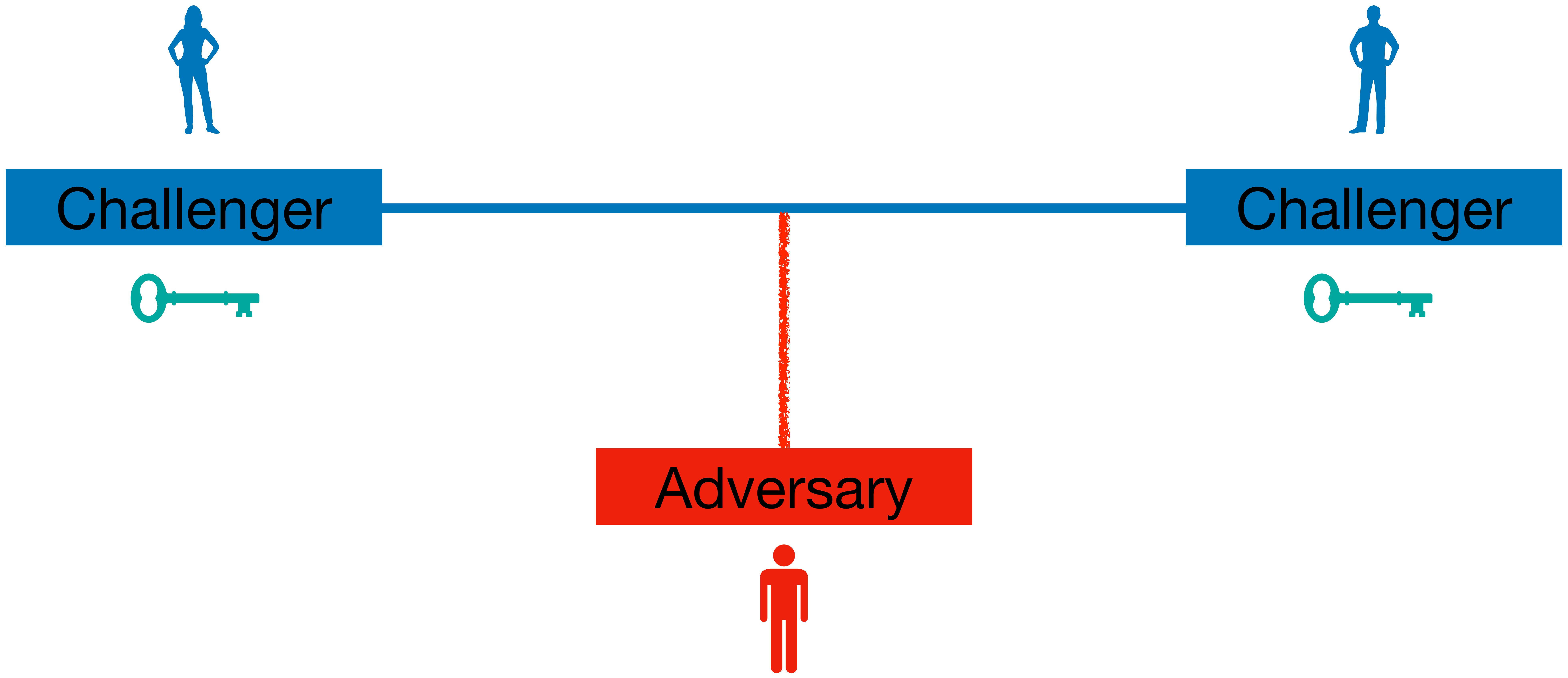
Incompressible Security



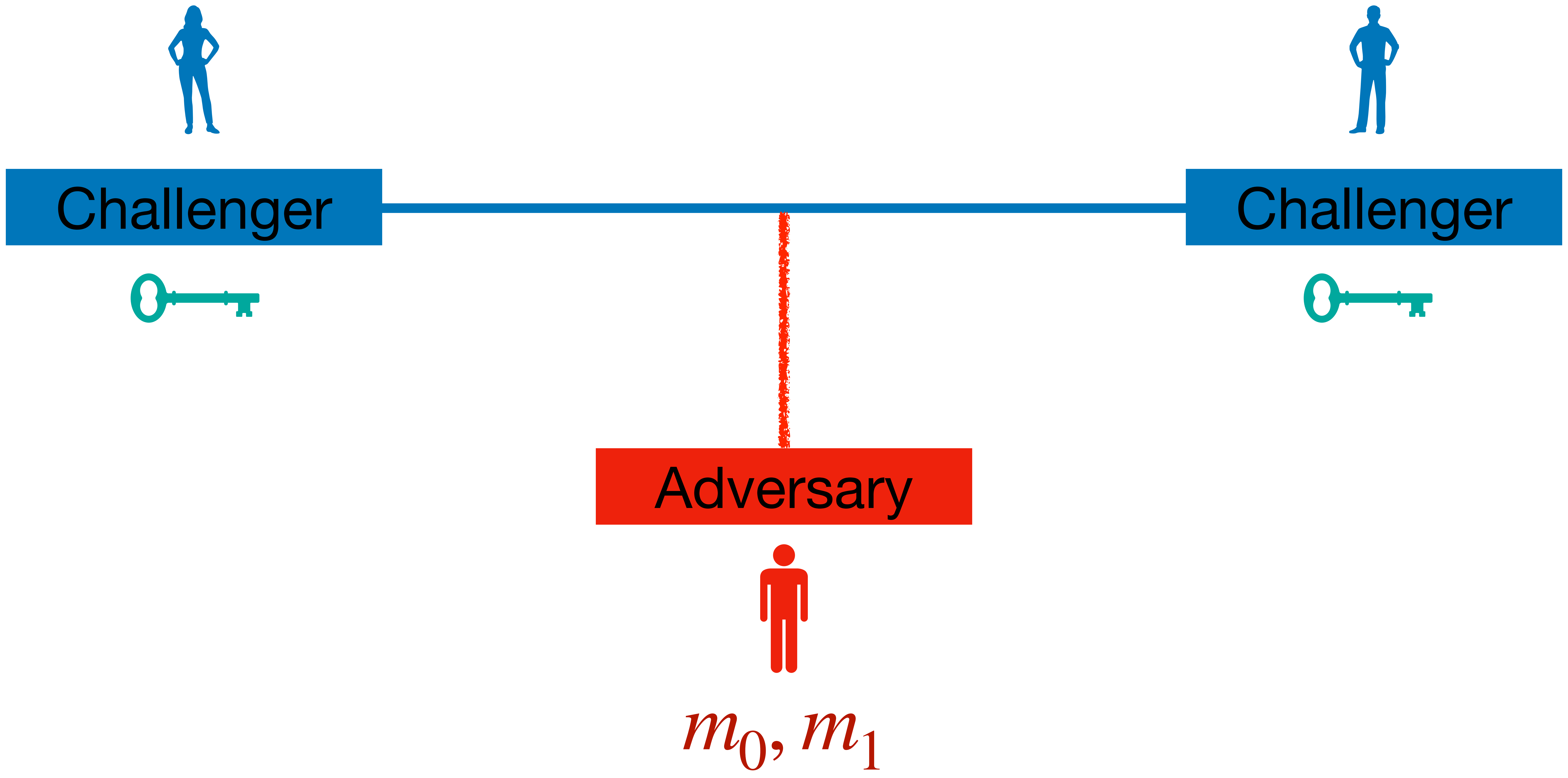
Incompressible Security



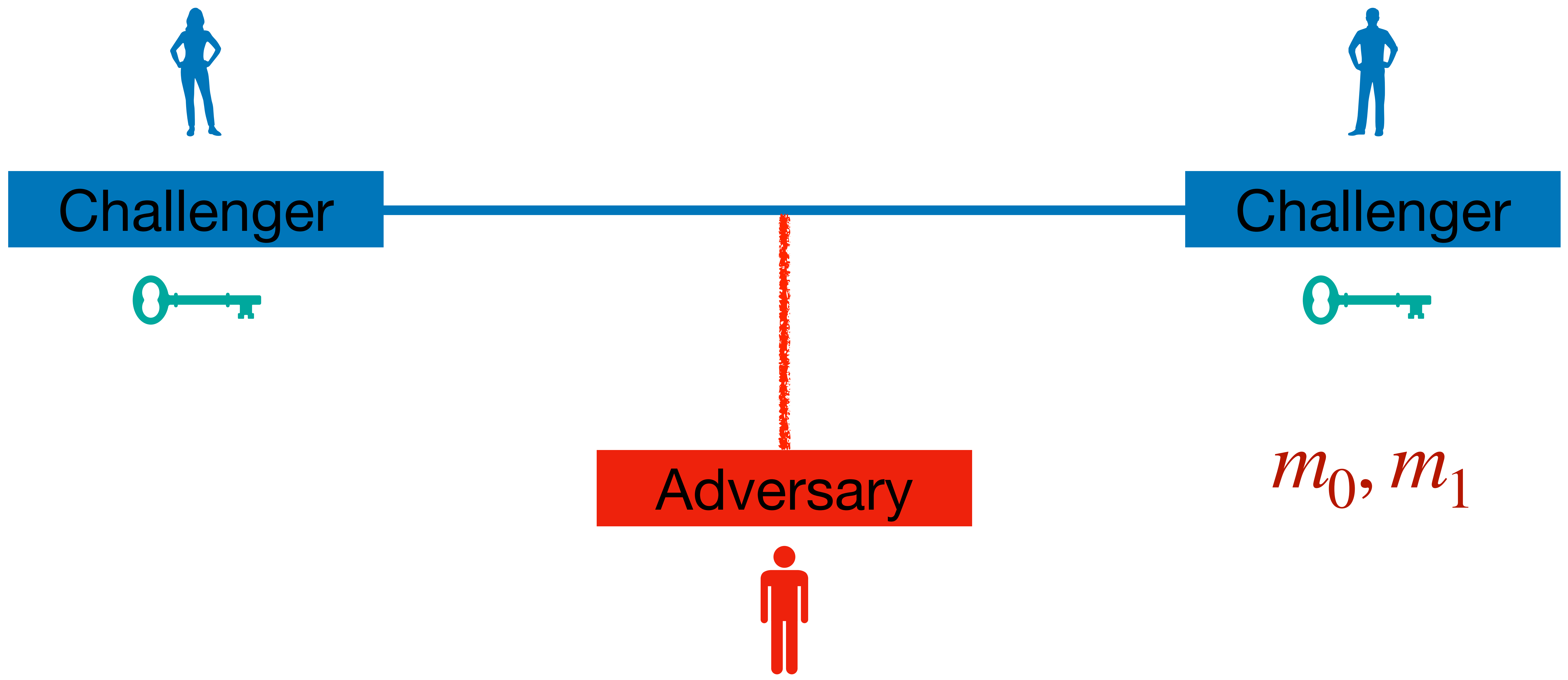
Incompressible Security



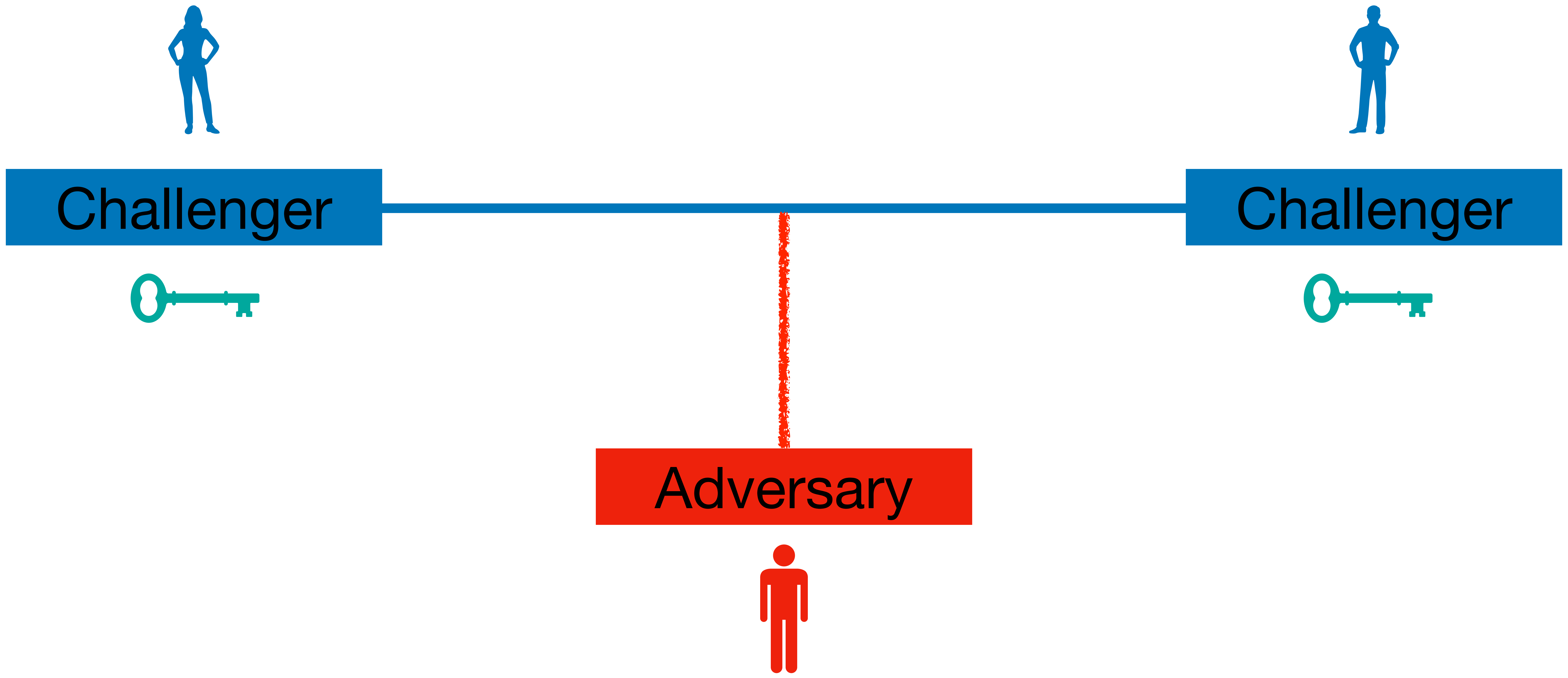
Incompressible Security



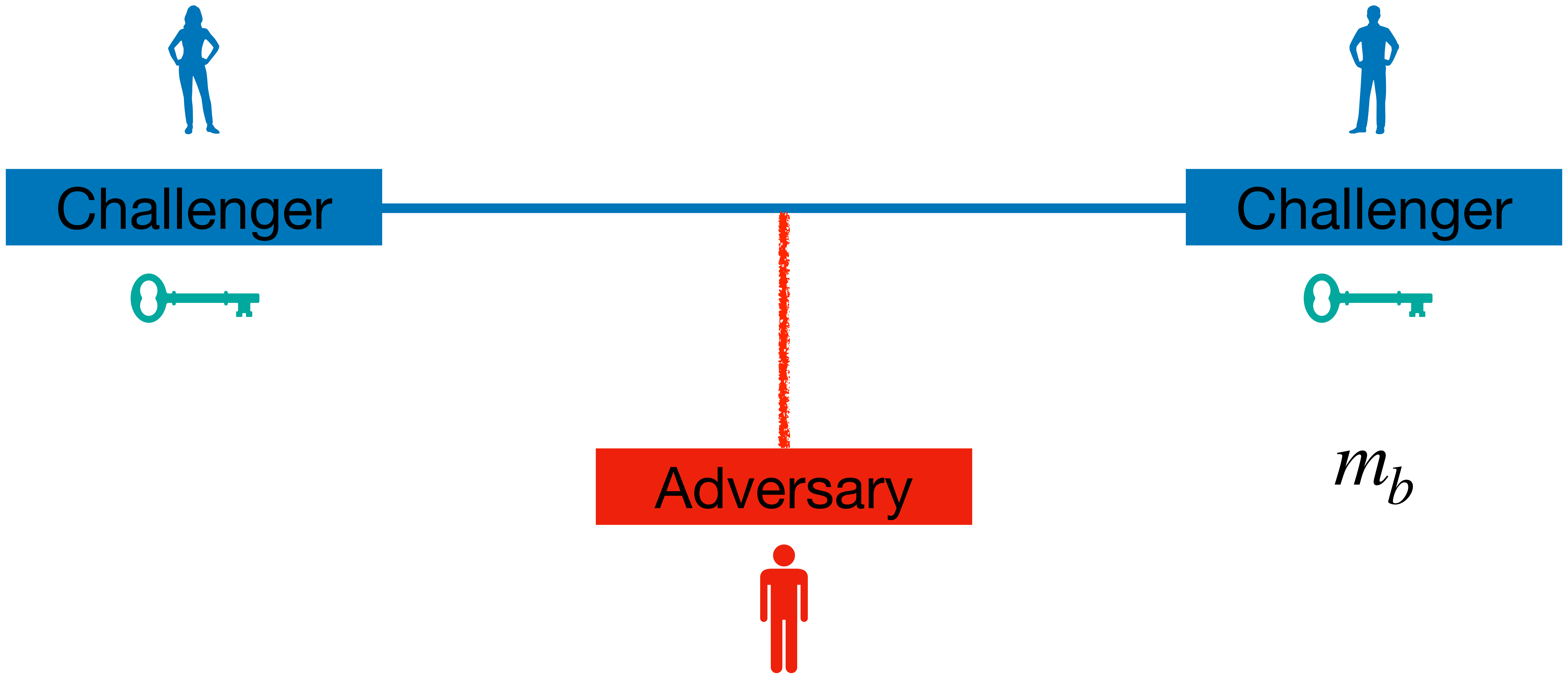
Incompressible Security



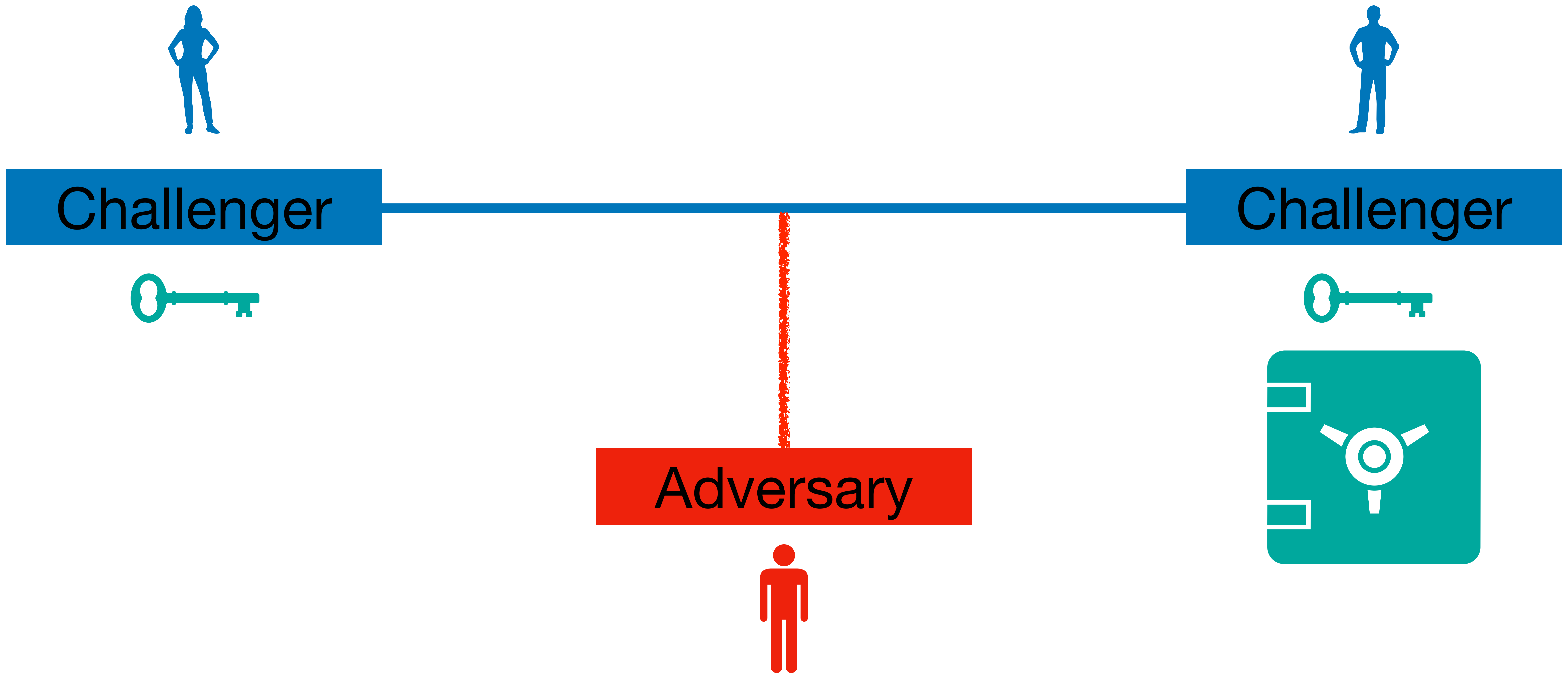
Incompressible Security



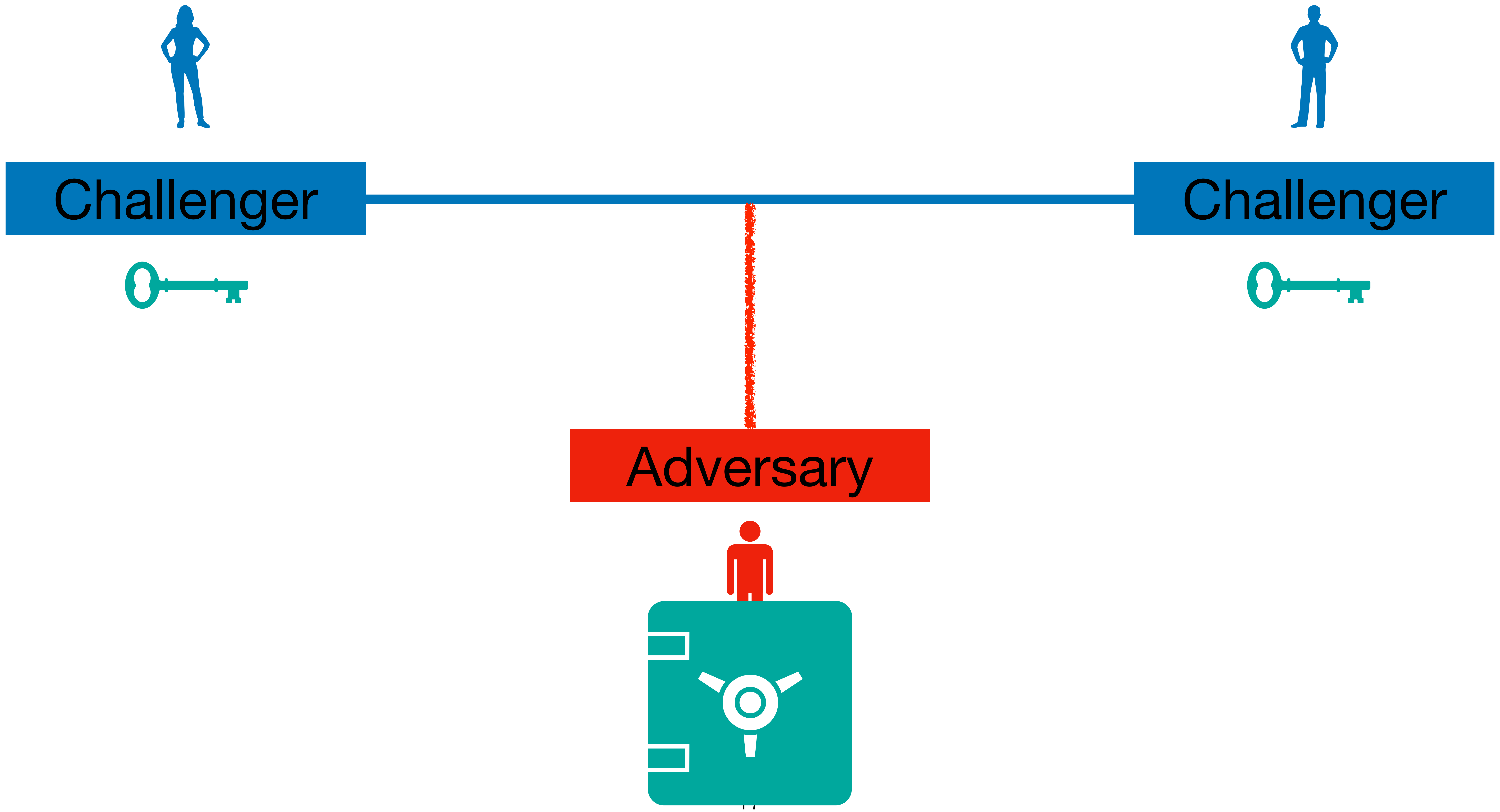
Incompressible Security



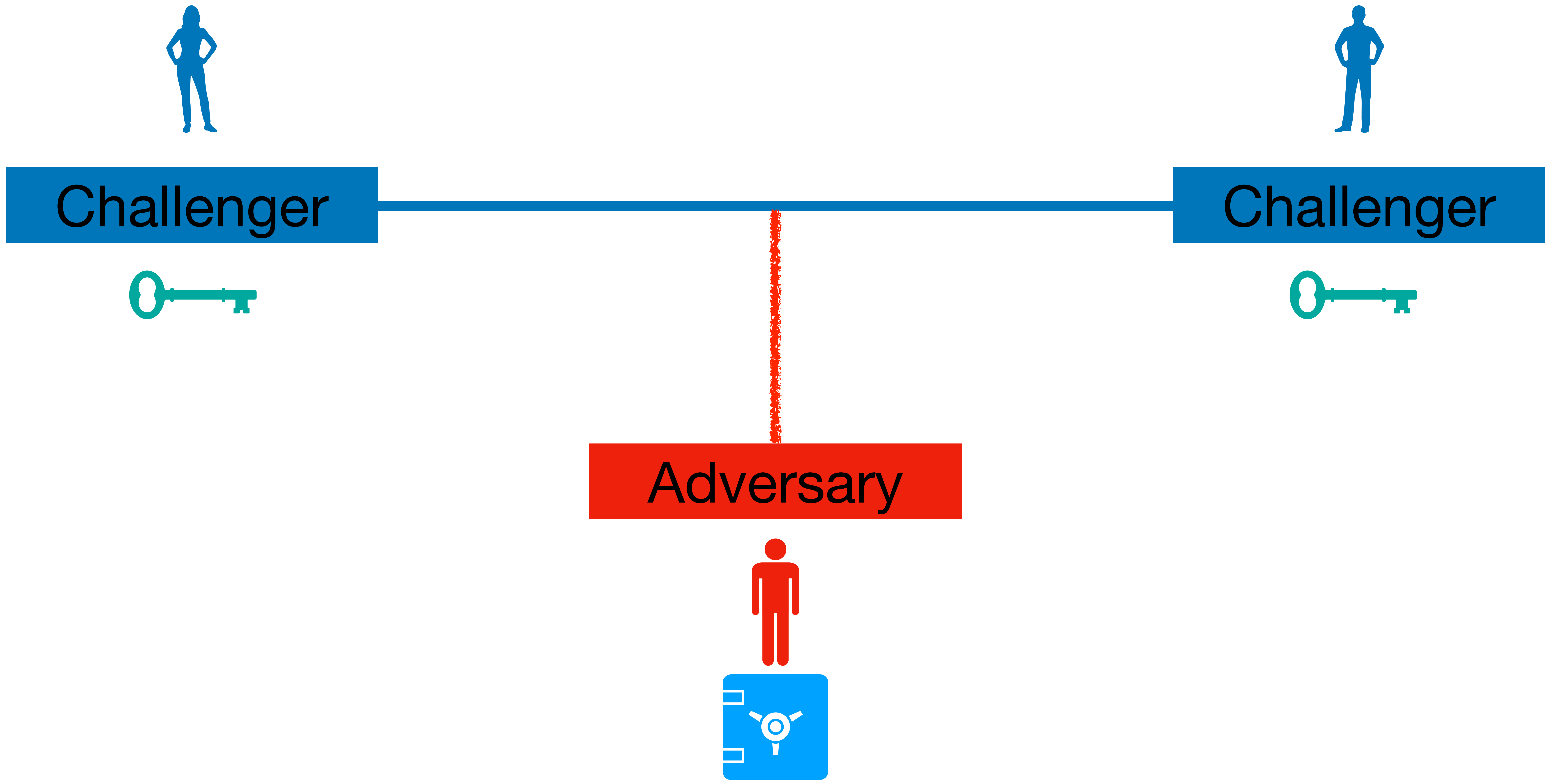
Incompressible Security



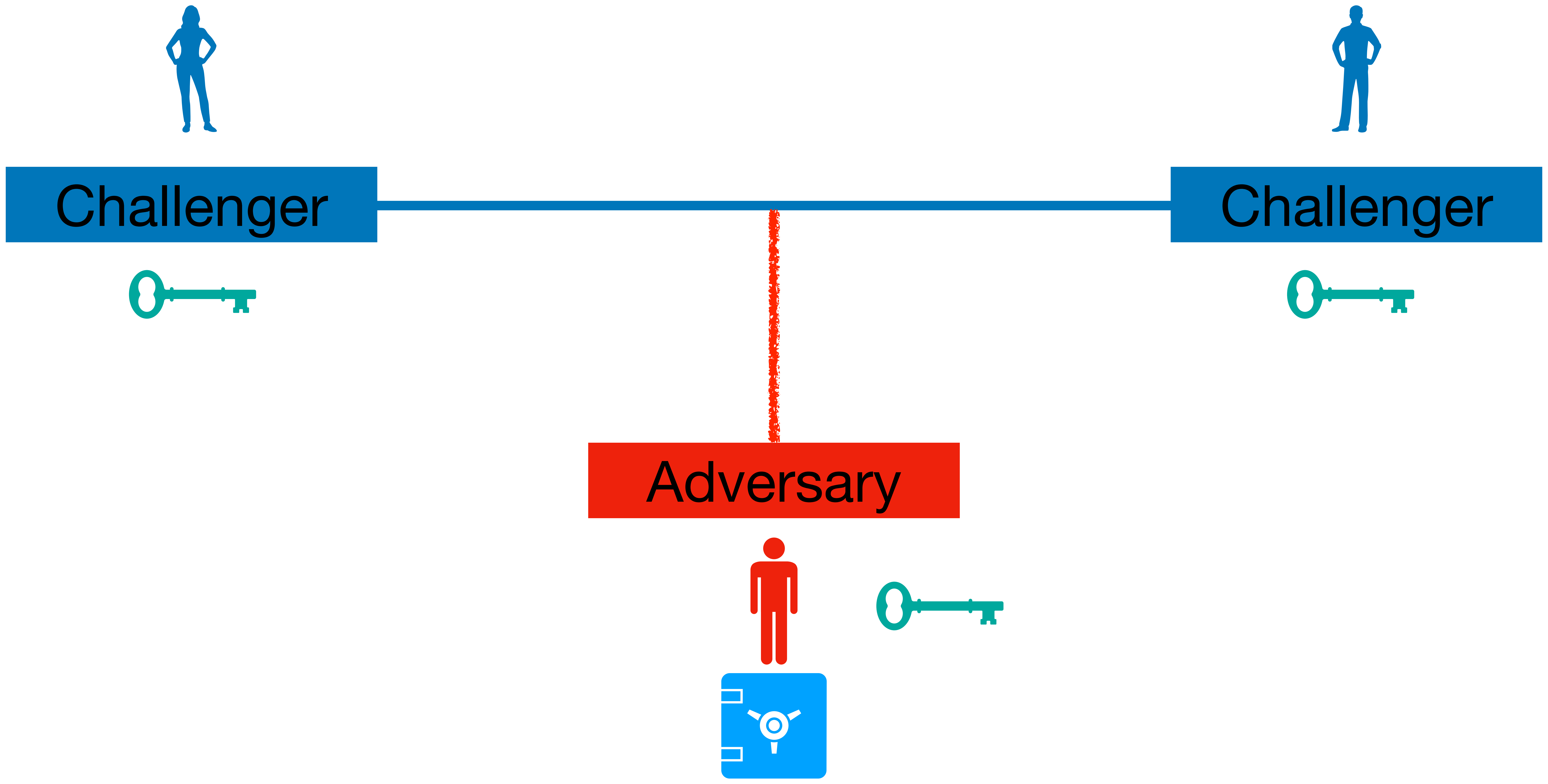
Incompressible Security



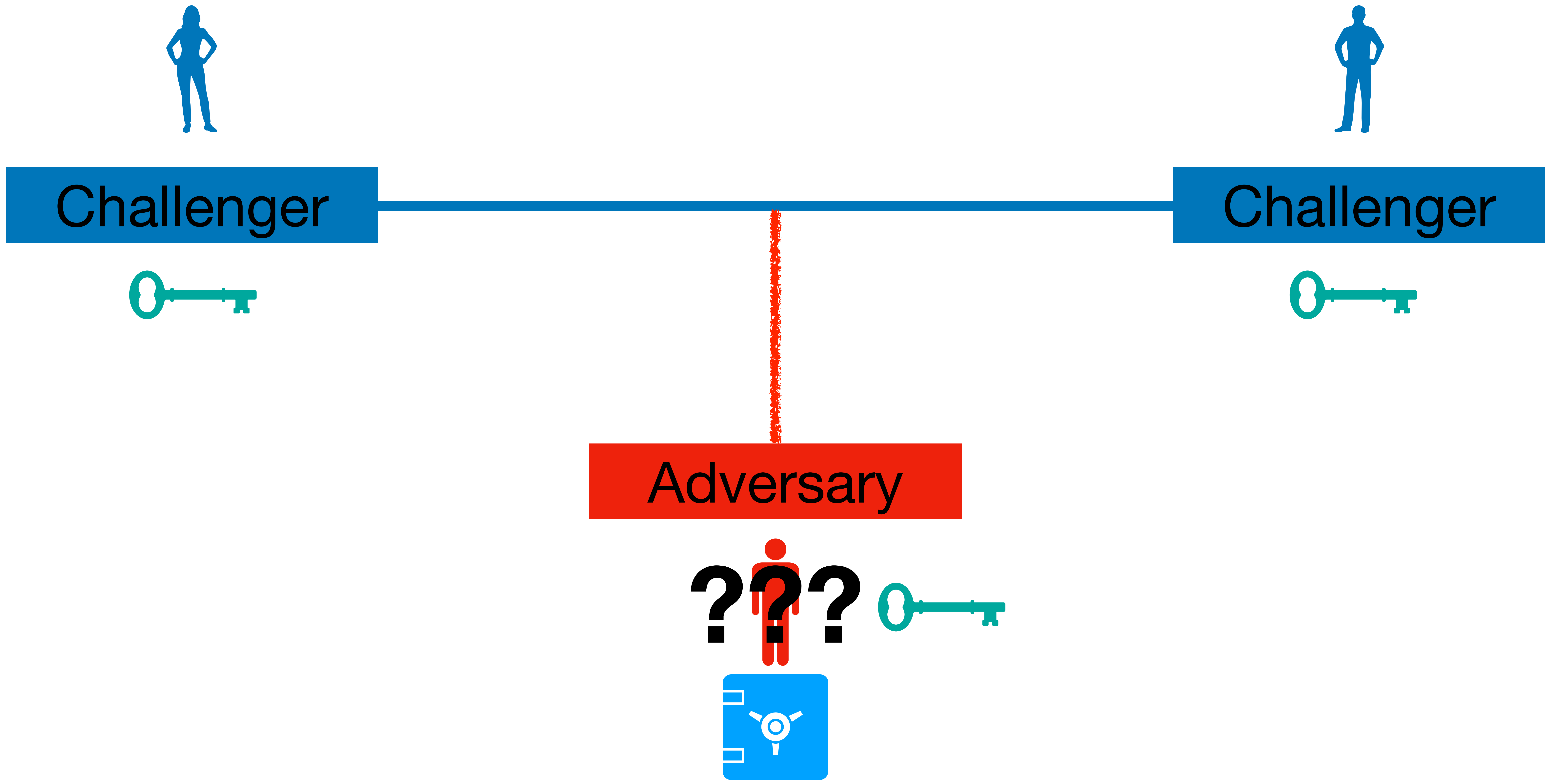
Incompressible Security



Incompressible Security



Incompressible Security



LR vs Incompressibility

LR vs Incompressibility

- Adversary leaks a part of secret key.

LR vs Incompressibility

- Adversary leaks a part of secret key.

- Adversary stores only a part of the cipher text.

LR vs Incompressibility

- Adversary leaks a part of secret key.

- Adversary stores the entire cipher text.

- Adversary stores only a part of the cipher text.

LR vs Incompressibility

- Adversary leaks a part of secret key.
- Adversary stores the entire cipher text.

- Adversary stores only a part of the cipher text.
- Adversary (later) receives the entire secret key.

Incompressible SKE Schemes

Incompressible SKE Schemes

- LR One-time pad is not incompressible.

Incompressible SKE Schemes

- LR One-time pad is not incompressible.
- Dziembowski gave the first construction under standard assumptions (bad rate)

Incompressible SKE Schemes

- LR One-time pad is not incompressible.
- Dziembowski gave the first construction under standard assumptions (bad rate)
- Guan et al. gave a rate-1 construction based on LWE and DCR (using incompressible encoding)

Dziembowski’s Incompressible One-Time Pad

Dziembowski's Incompressible One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\lambda}$.

Dziembowski's Incompressible One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^\lambda$.
- $Enc(sk, m) \rightarrow$

Dziembowski's Incompressible One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\lambda}$.
- $Enc(sk, m) \rightarrow$
  1. Randomly generate c_0

Dziembowski's Incompressible One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\lambda}$.
- $Enc(sk, m) \rightarrow$
    1. Randomly generate c_0
    2. Run $s = Ext_{sk}(c_0)$

Dziembowski's Incompressible One-Time Pad

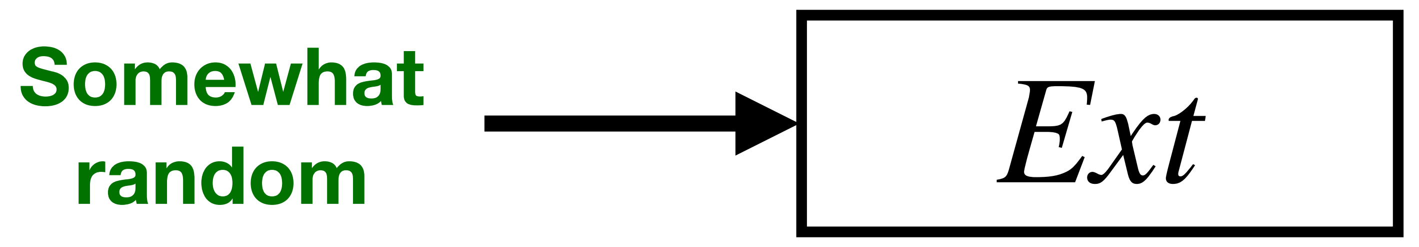
- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\lambda}$.
- $Enc(sk, m) \rightarrow$

$$\boxed{Ext}$$

1. Randomly generate c_0
2. Run $s = Ext_{sk}(c_0)$

Dziembowski's Incompressible One-Time Pad

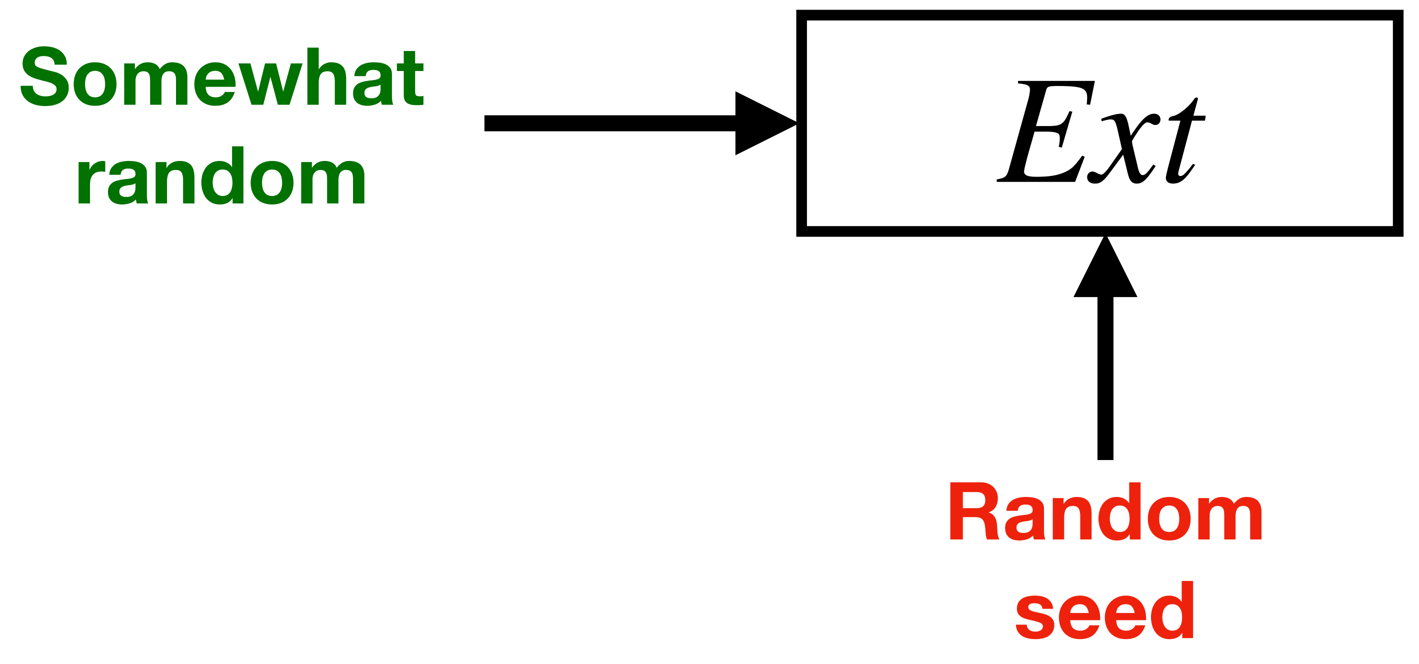
- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^\lambda$.
- $Enc(sk, m) \rightarrow$
  1. Randomly generate c_0
  2. Run $s = Ext_{sk}(c_0)$



Dziembowski's Incompressible One-Time Pad

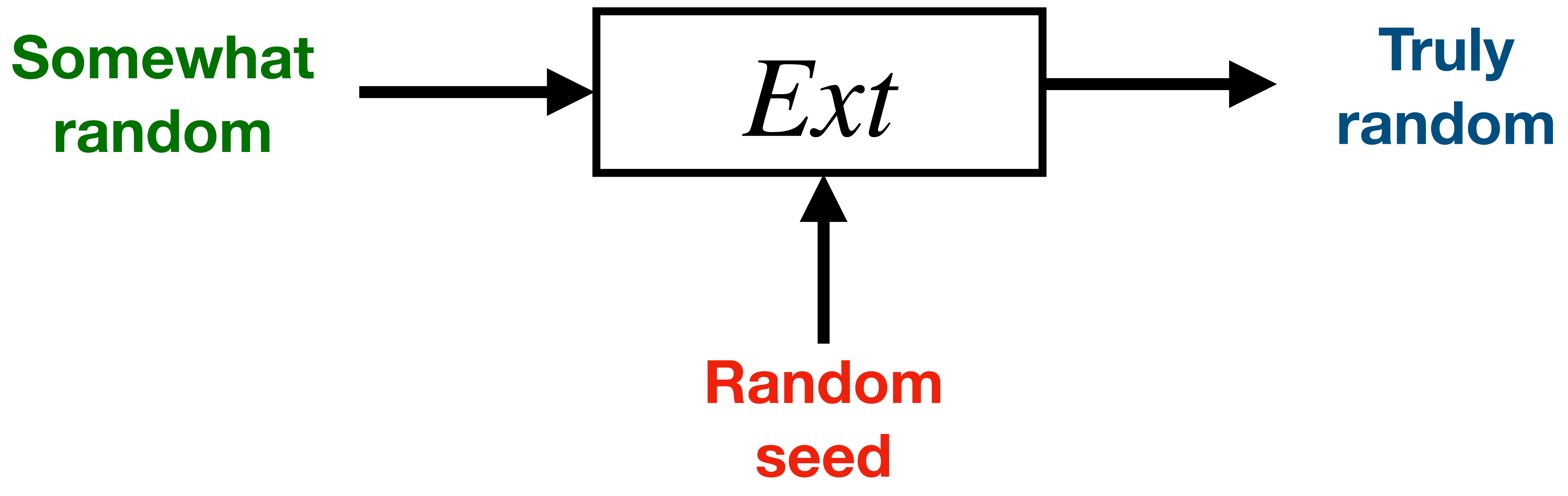
- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\lambda}$.
- $Enc(sk, m) \rightarrow$

  1. Randomly generate c_0
  2. Run $s = Ext_{sk}(c_0)$



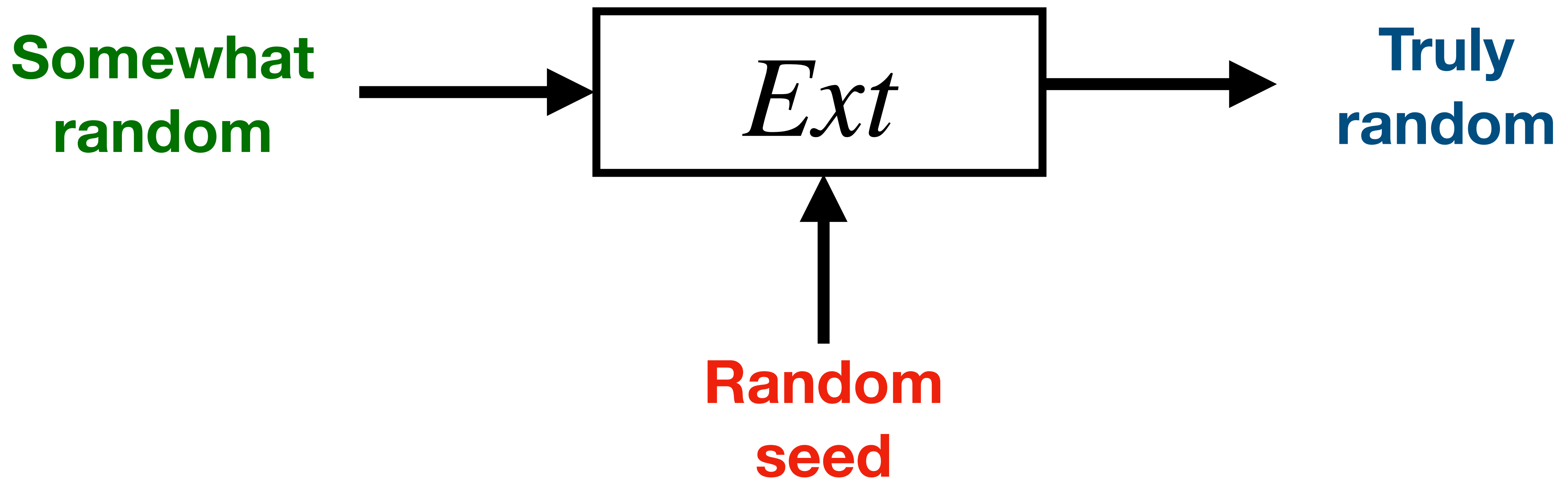
Dziembowski's Incompressible One-Time Pad

- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\lambda}$.
- $Enc(sk, m) \rightarrow$
  1. Randomly generate c_0
  2. Run $s = Ext_{sk}(c_0)$



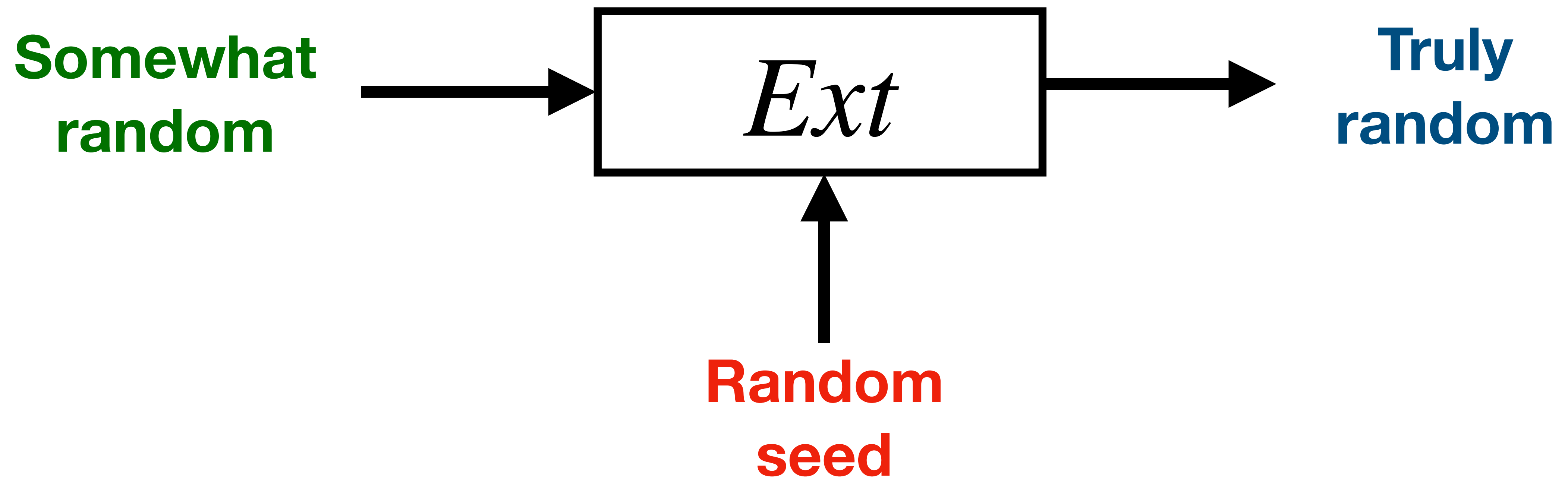
Dziembowski's Incompressible One-Time Pad

- $Setup() \to$ Randomly generate $sk \in \{0,1\}^\lambda$.
- $Enc(sk, m) \to$
  1. Randomly generate c_0
  2. Run $s = Ext_{sk}(c_0)$
  3. Compute $c_1 = s \oplus m$ (one time pad)



Dziembowski's Incompressible One-Time Pad

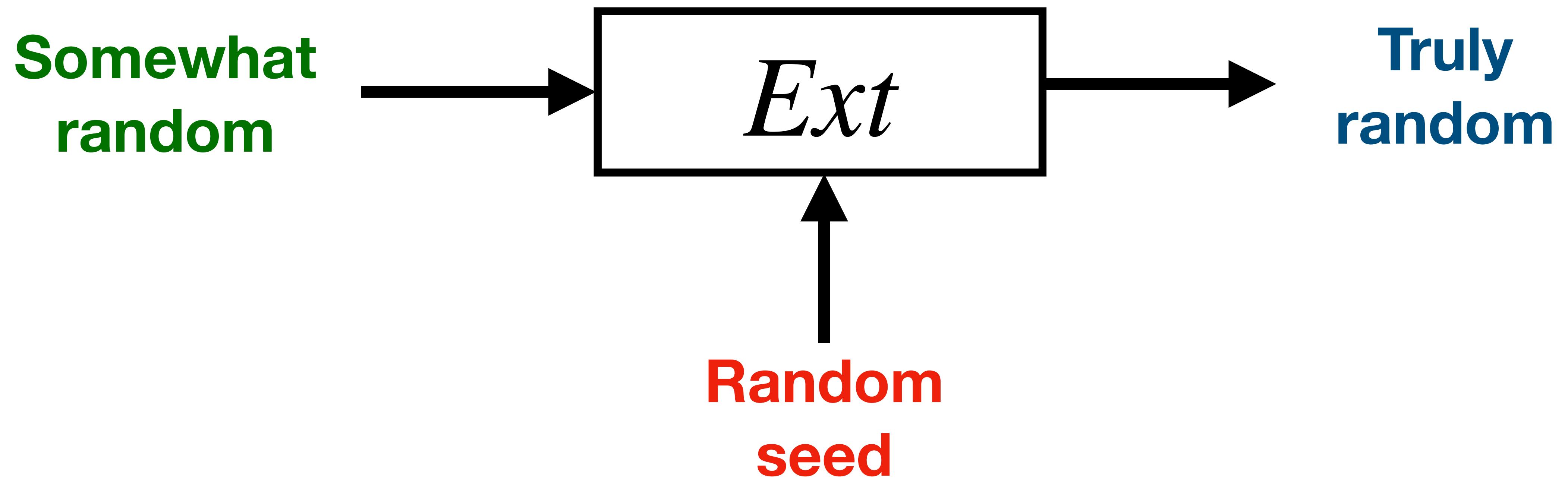
- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\lambda}$.
- $Enc(sk, m) \rightarrow$



1. Randomly generate c_0
2. Run $s = Ext_{sk}(c_0)$
3. Compute $c_1 = s \oplus m$ (one time pad)
4. Return (c_0, c_1)

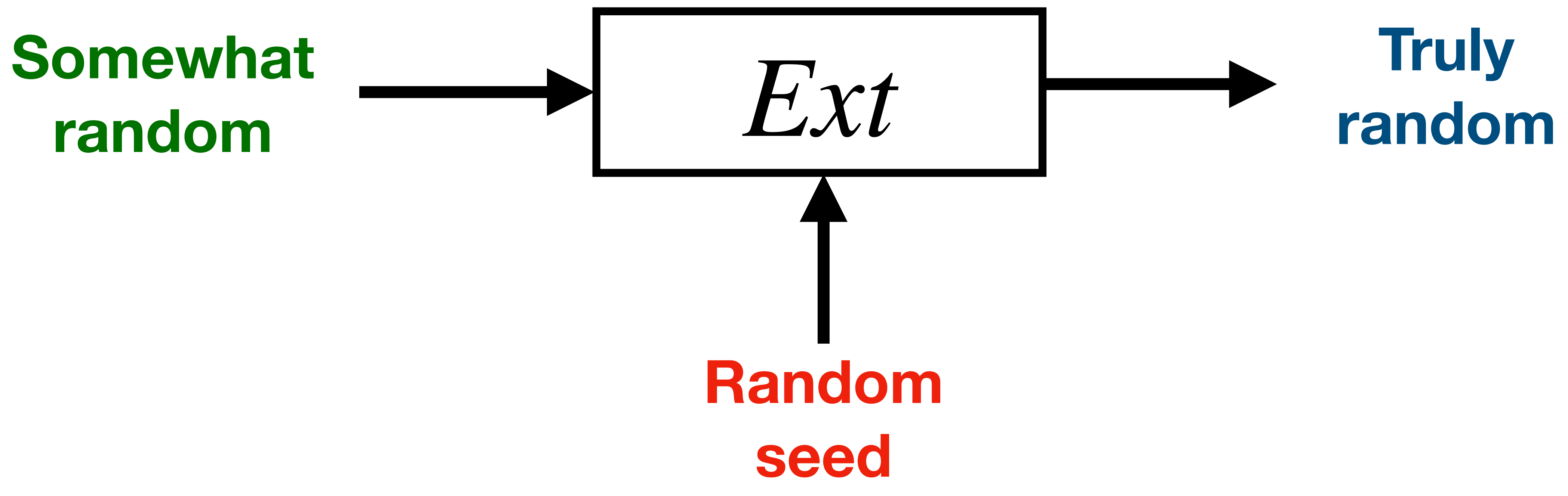
Dziembowski's Incompressible One-Time Pad

- $Setup() \to$ Randomly generate $sk \in \{0,1\}^\lambda$.
- $Enc(sk, m) \to$
  1. Randomly generate c_0
  2. Run $s = Ext_{sk}(c_0)$
  3. Compute $c_1 = s \oplus m$ (one time pad)
  4. Return (c_0, c_1)
- $Dec(sk, ct) \to$ Run $s = Ext_{sk}(c_0)$ and return $s \oplus ct$.



Dziembowski's Incompressible One-Time Pad

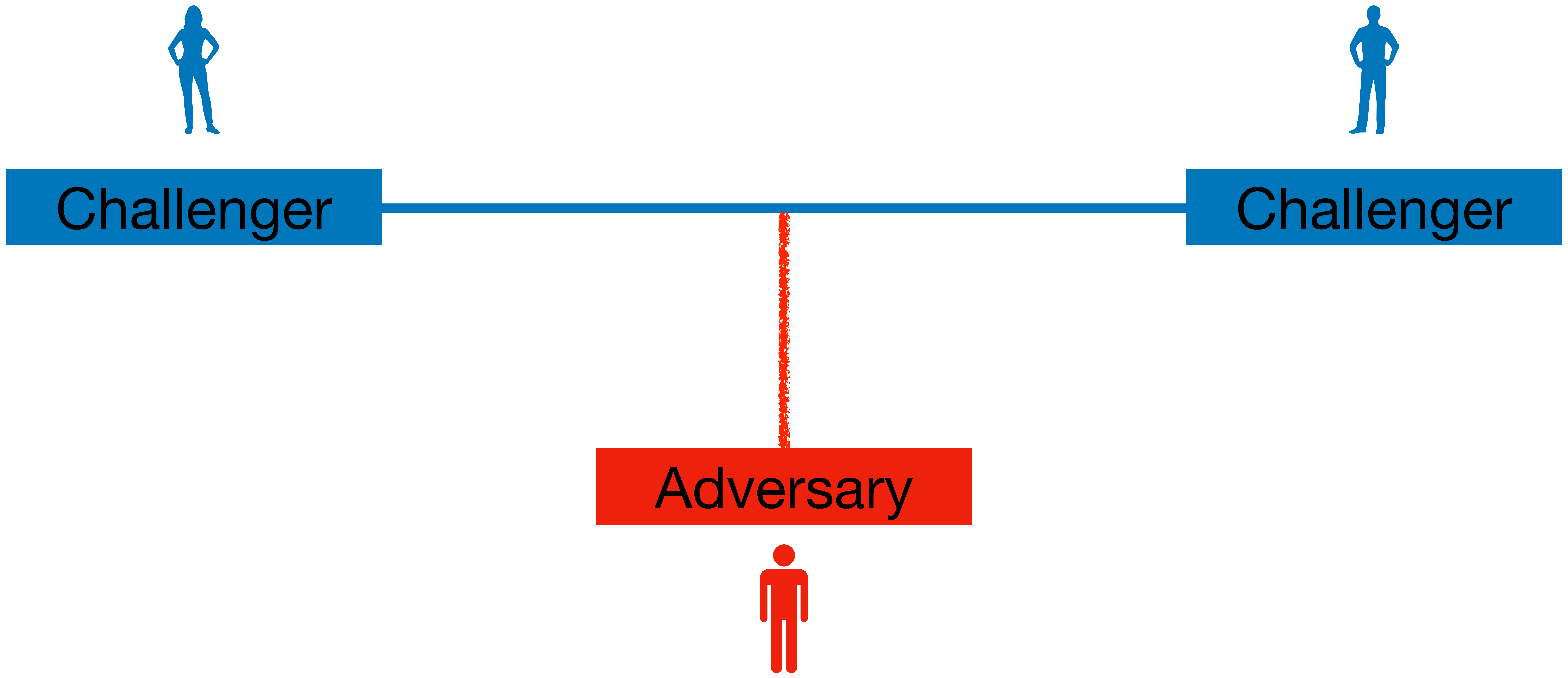
- $Setup() \rightarrow$ Randomly generate $sk \in \{0,1\}^{\lambda}$.
- $Enc(sk, m) \rightarrow$
  1. Randomly generate c_0
  2. Run $s = Ext_{sk}(c_0)$
  3. Compute $c_1 = s \oplus m$ (one time pad)
  4. Return (c_0, c_1)
- $Dec(sk, ct) \rightarrow$ Run $s = Ext_{sk}(c_0)$ and return $s \oplus ct$.



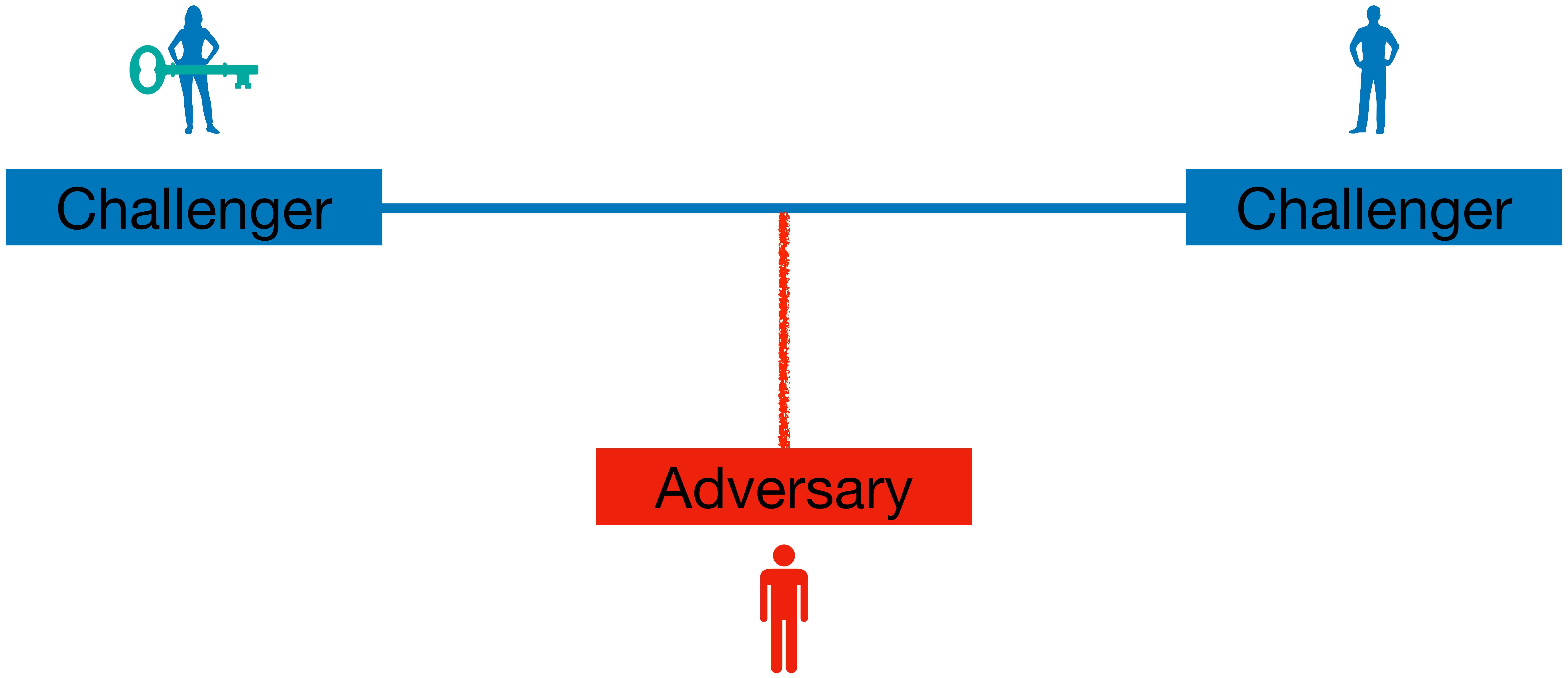
$$(sk, Ext_{sk}(c_0), f(c_0, c_1)) \approx (sk, U, f(c_0, c_1))$$

Leakage-Resilient Incompressibility

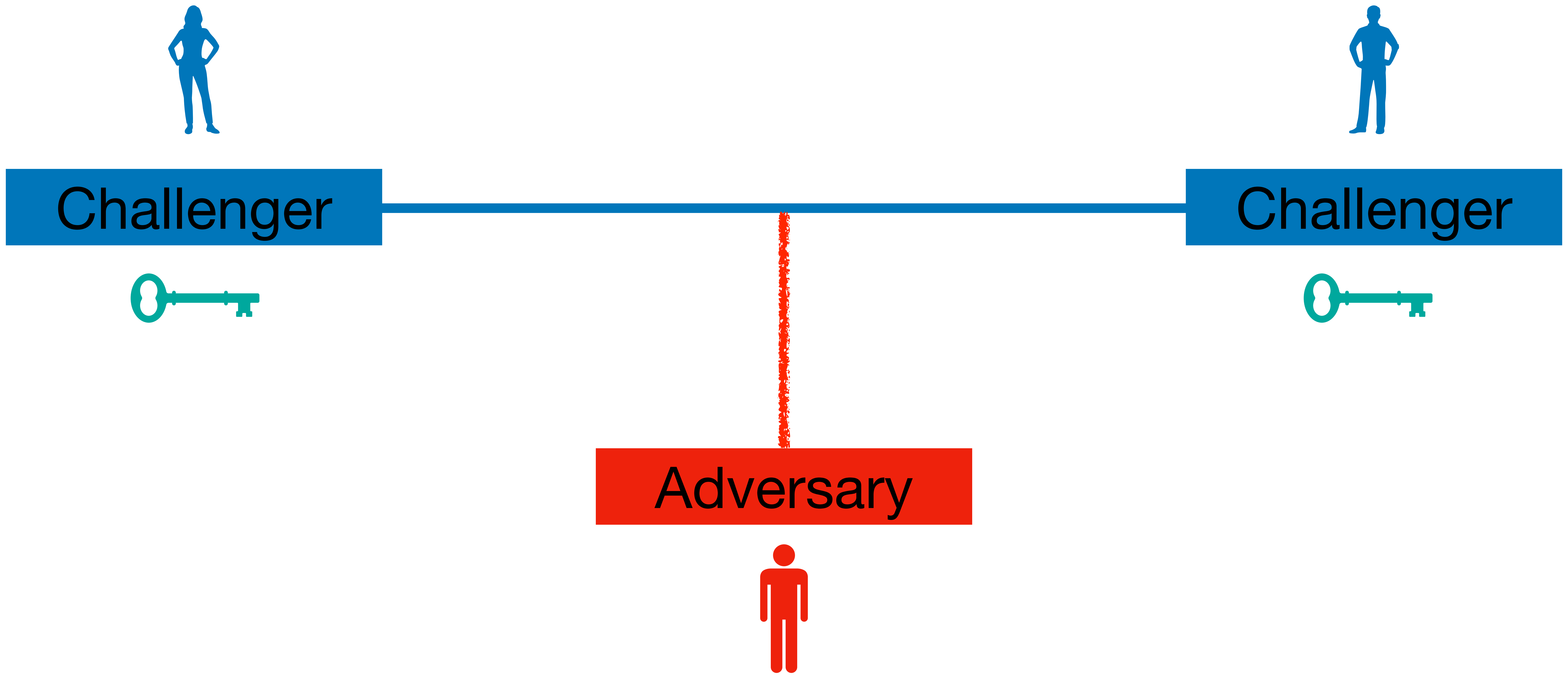
LRI Security



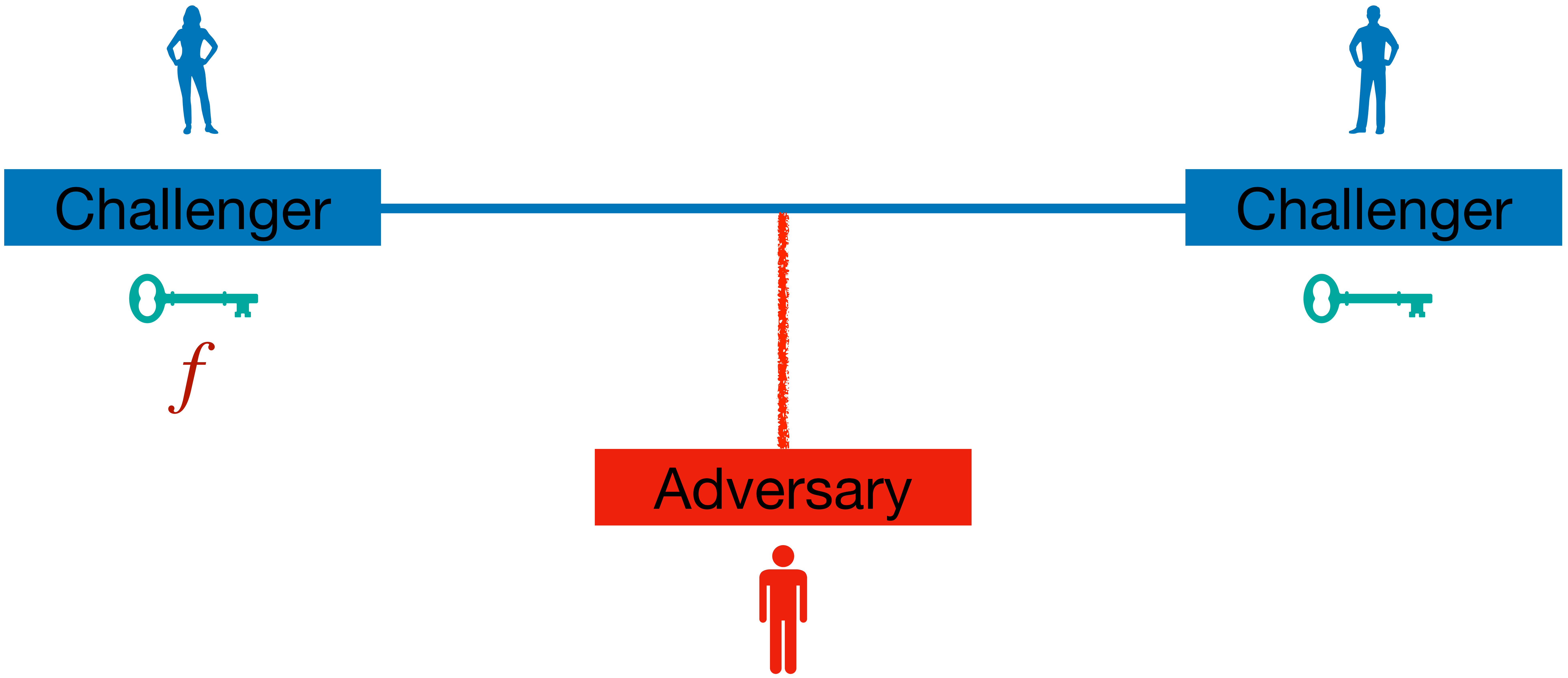
LRI Security



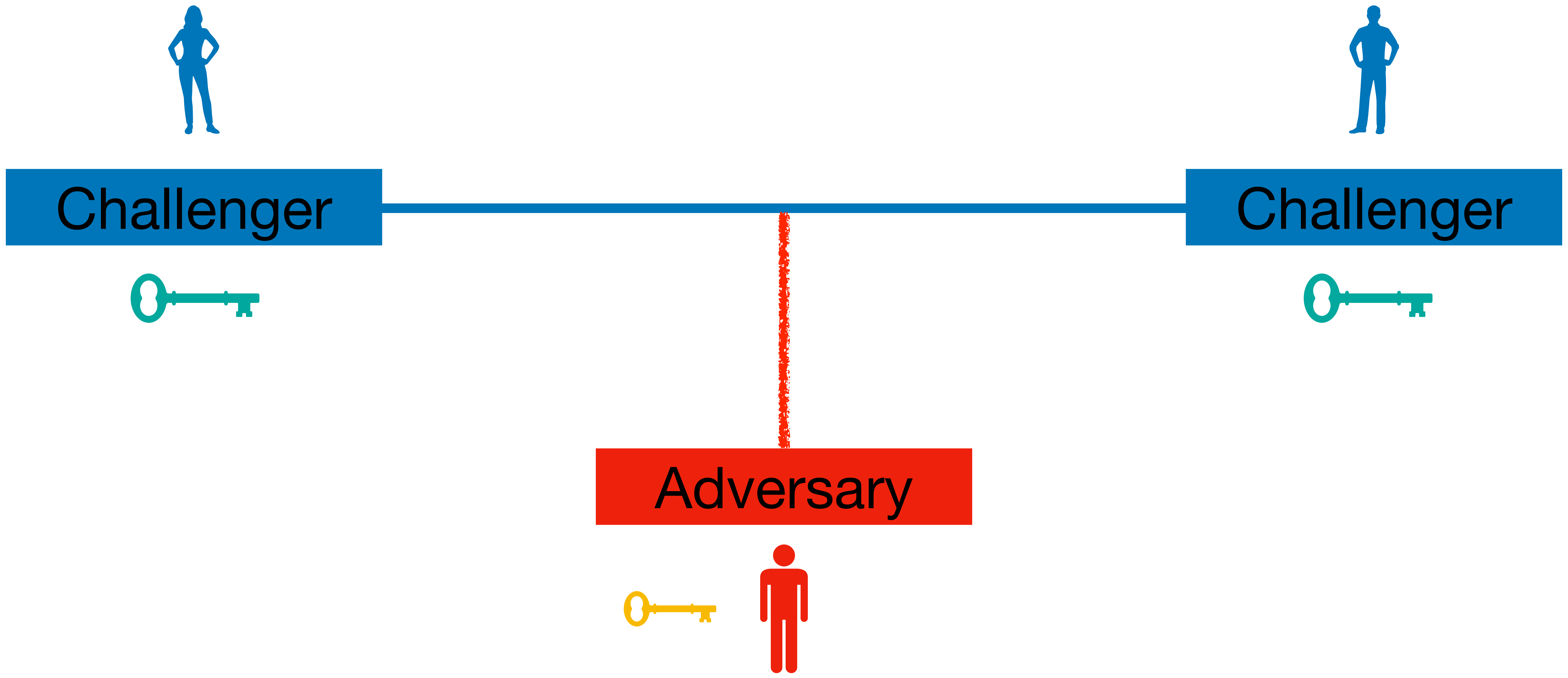
LRI Security



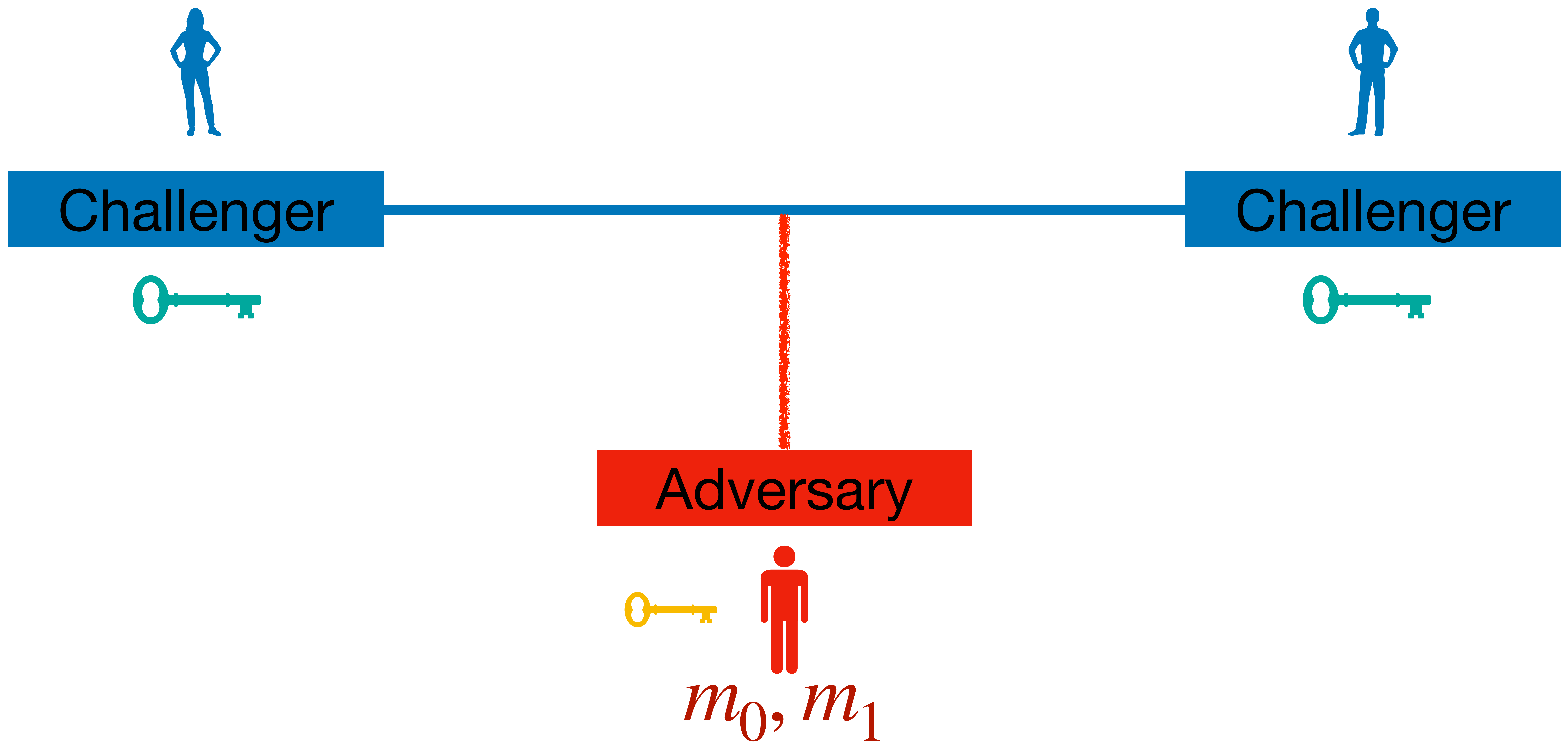
LRI Security



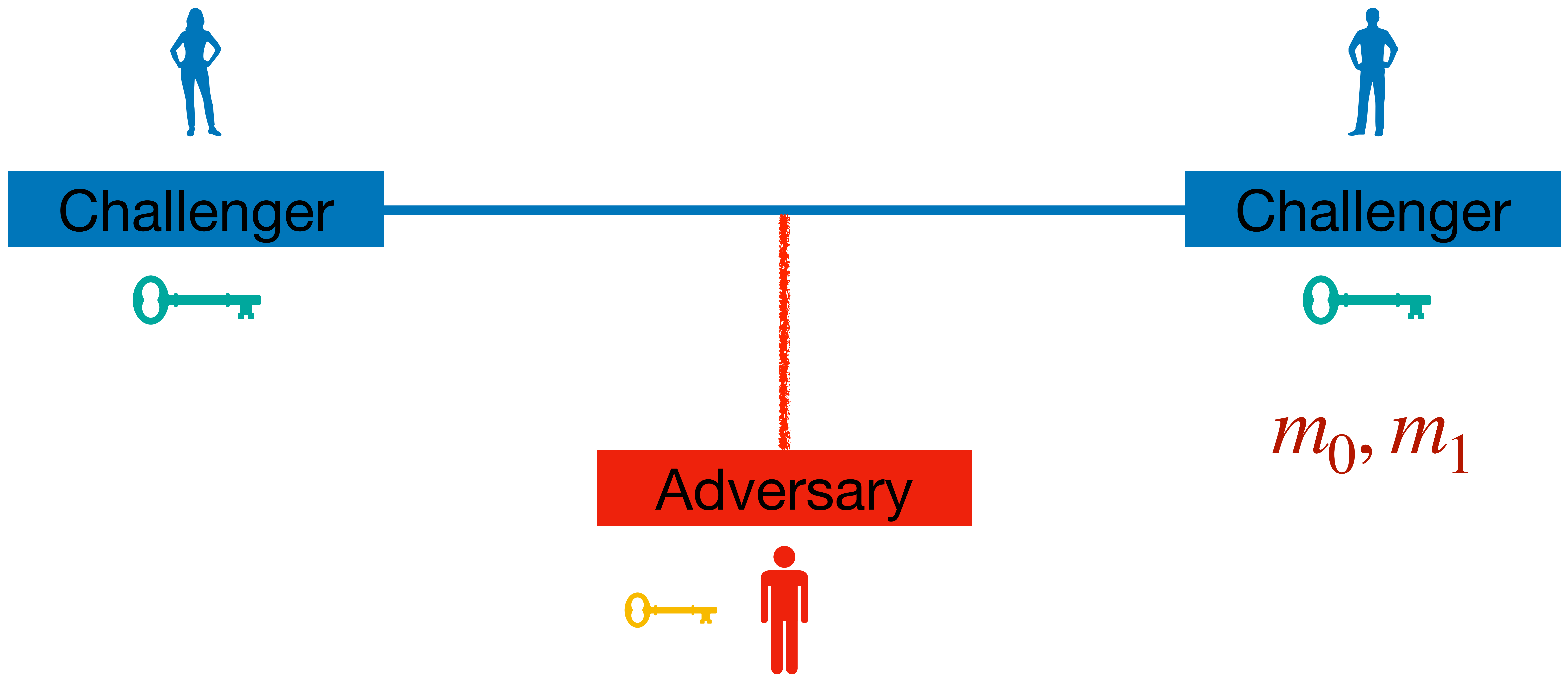
LRI Security



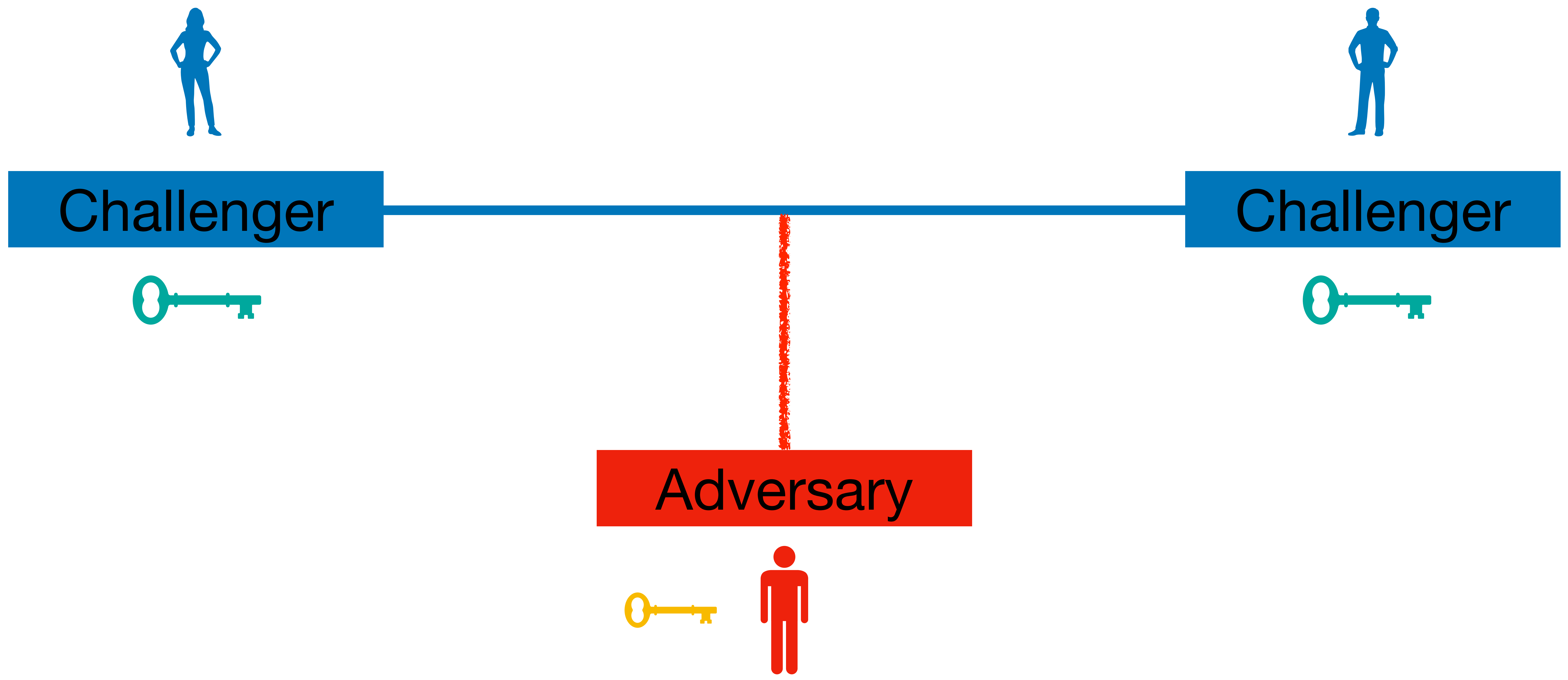
LRI Security



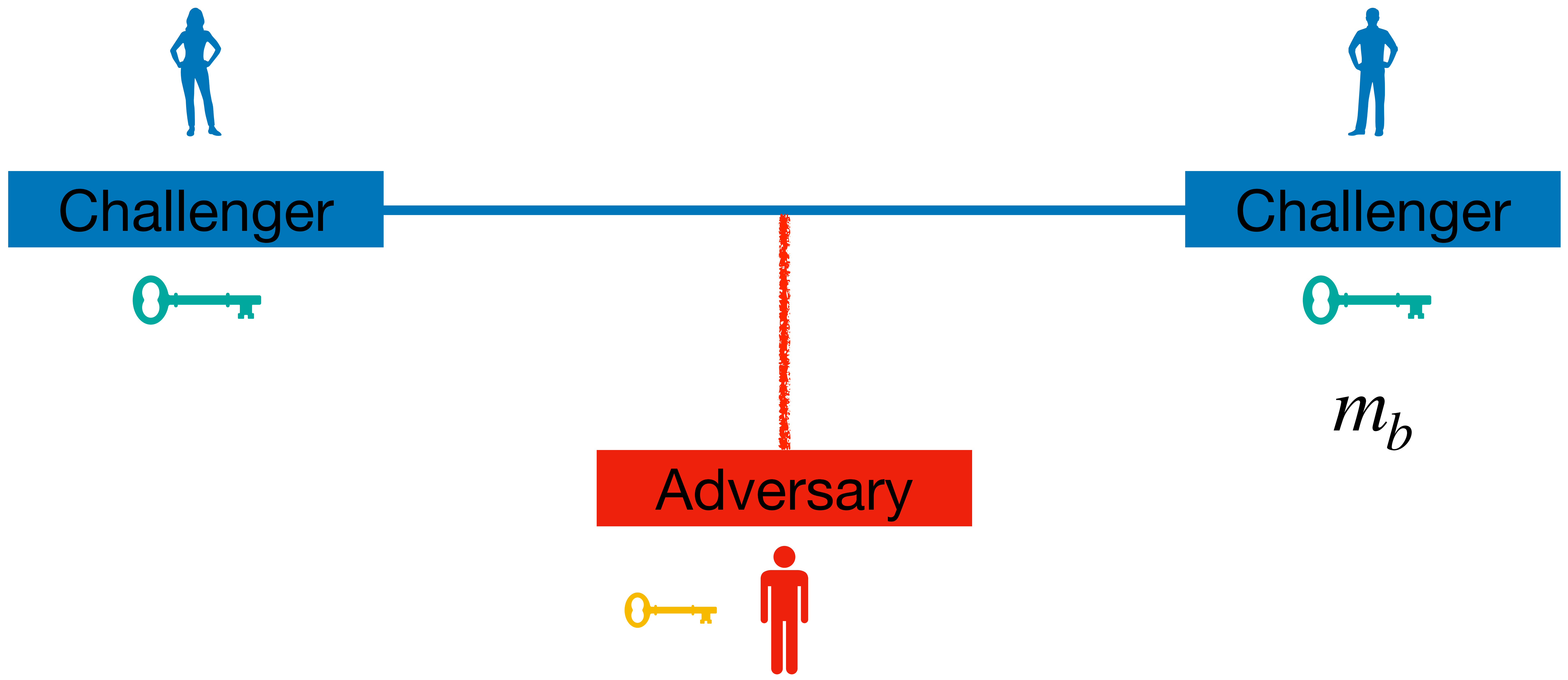
LRI Security



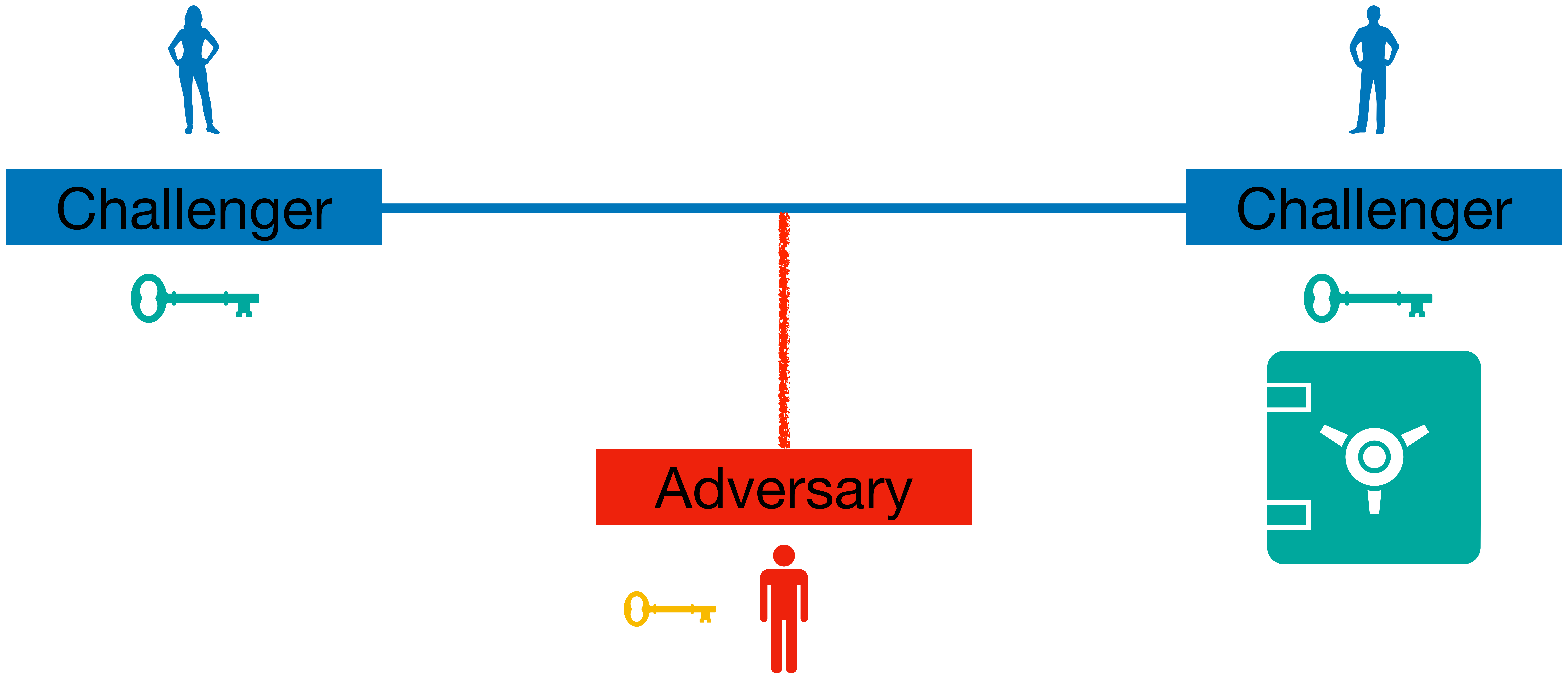
LRI Security



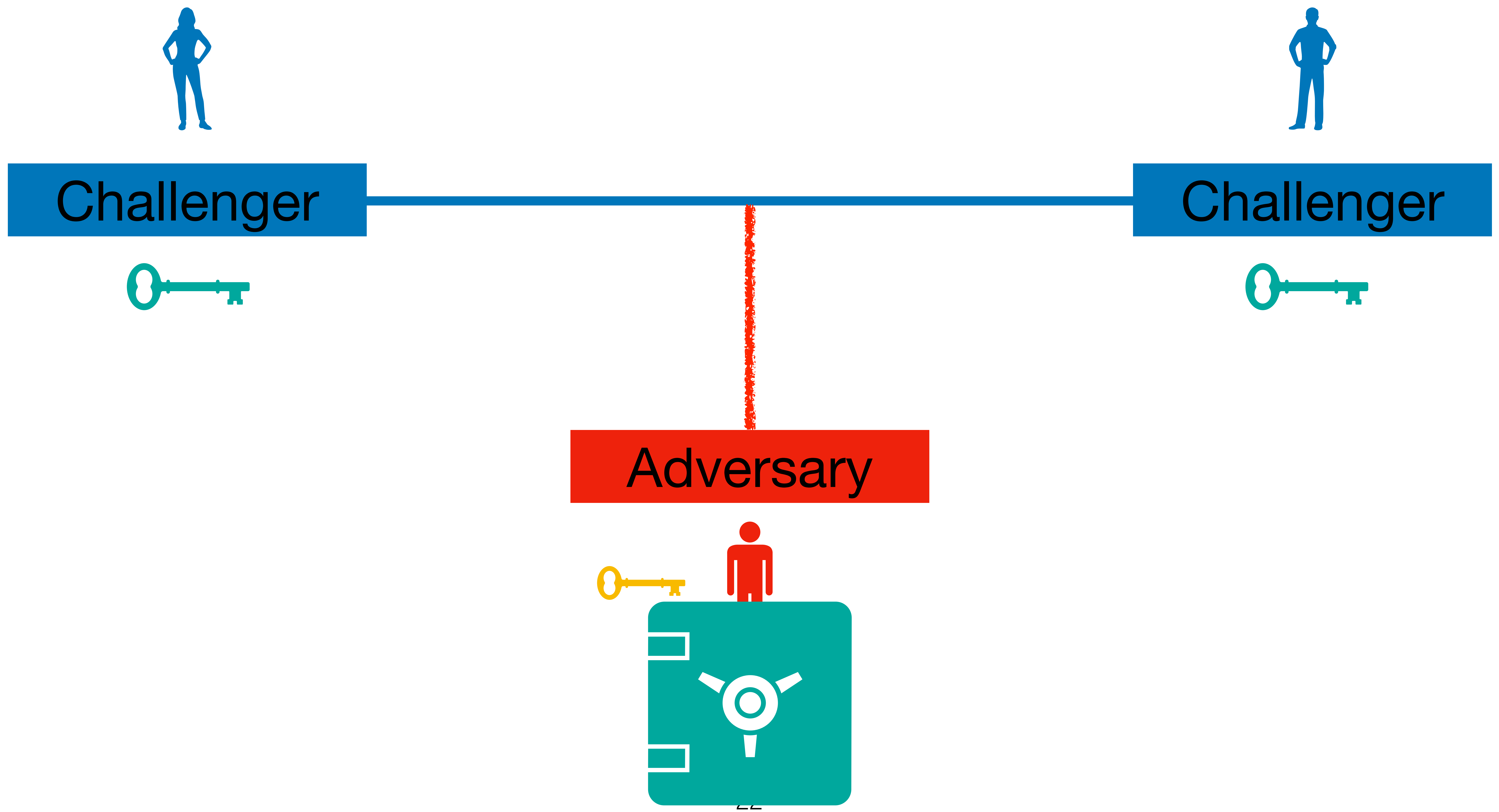
LRI Security



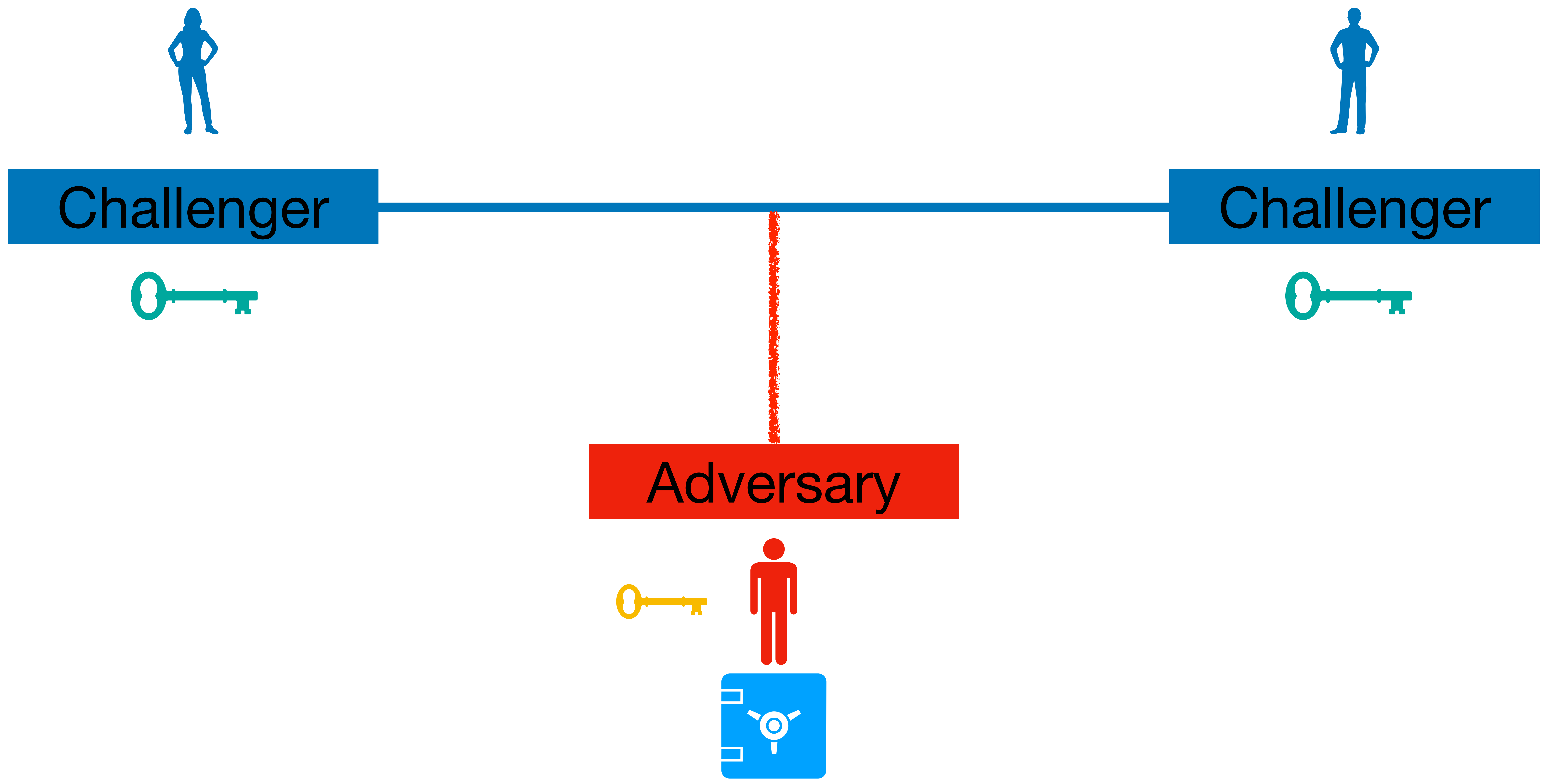
LRI Security



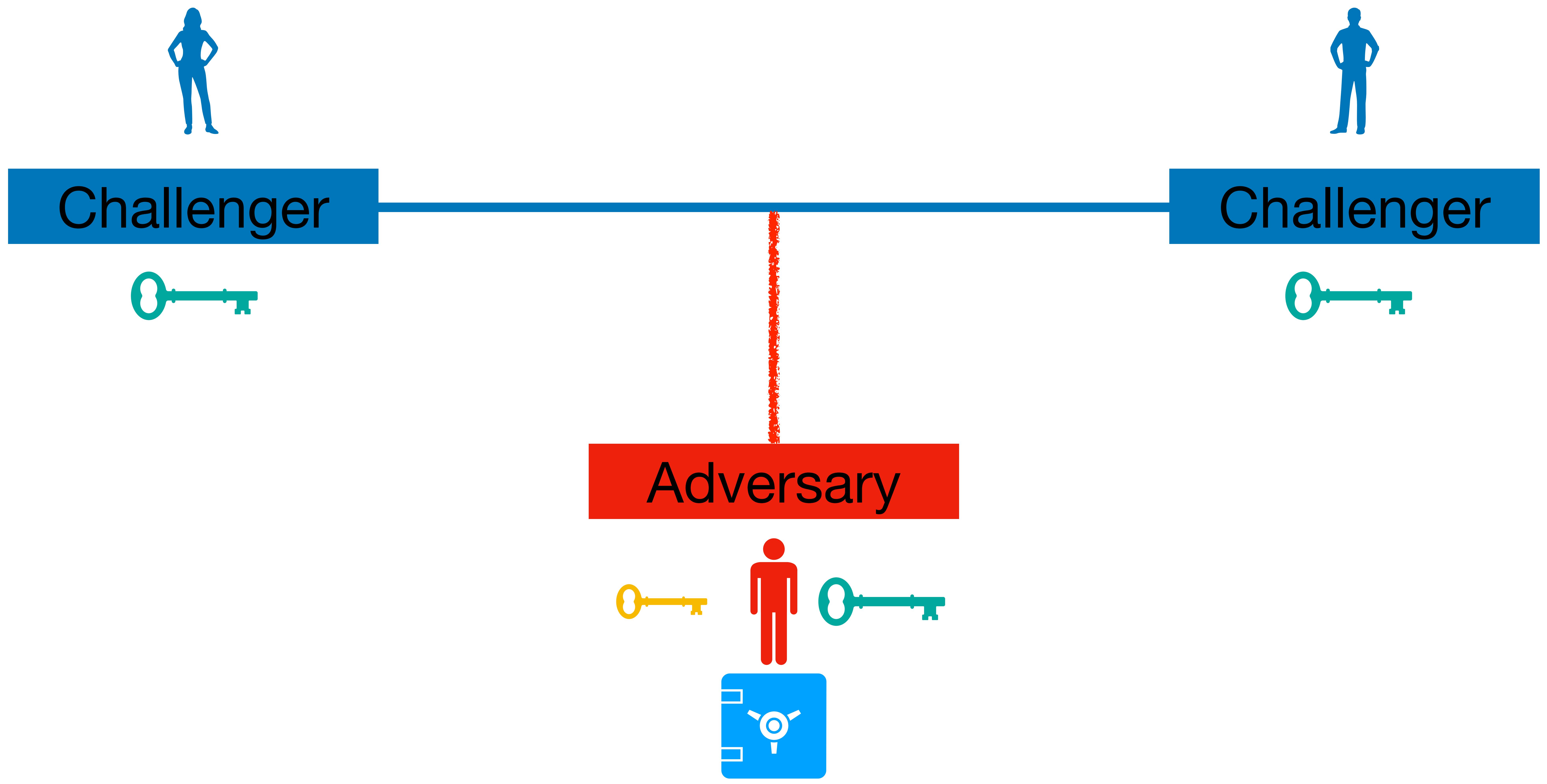
LRI Security



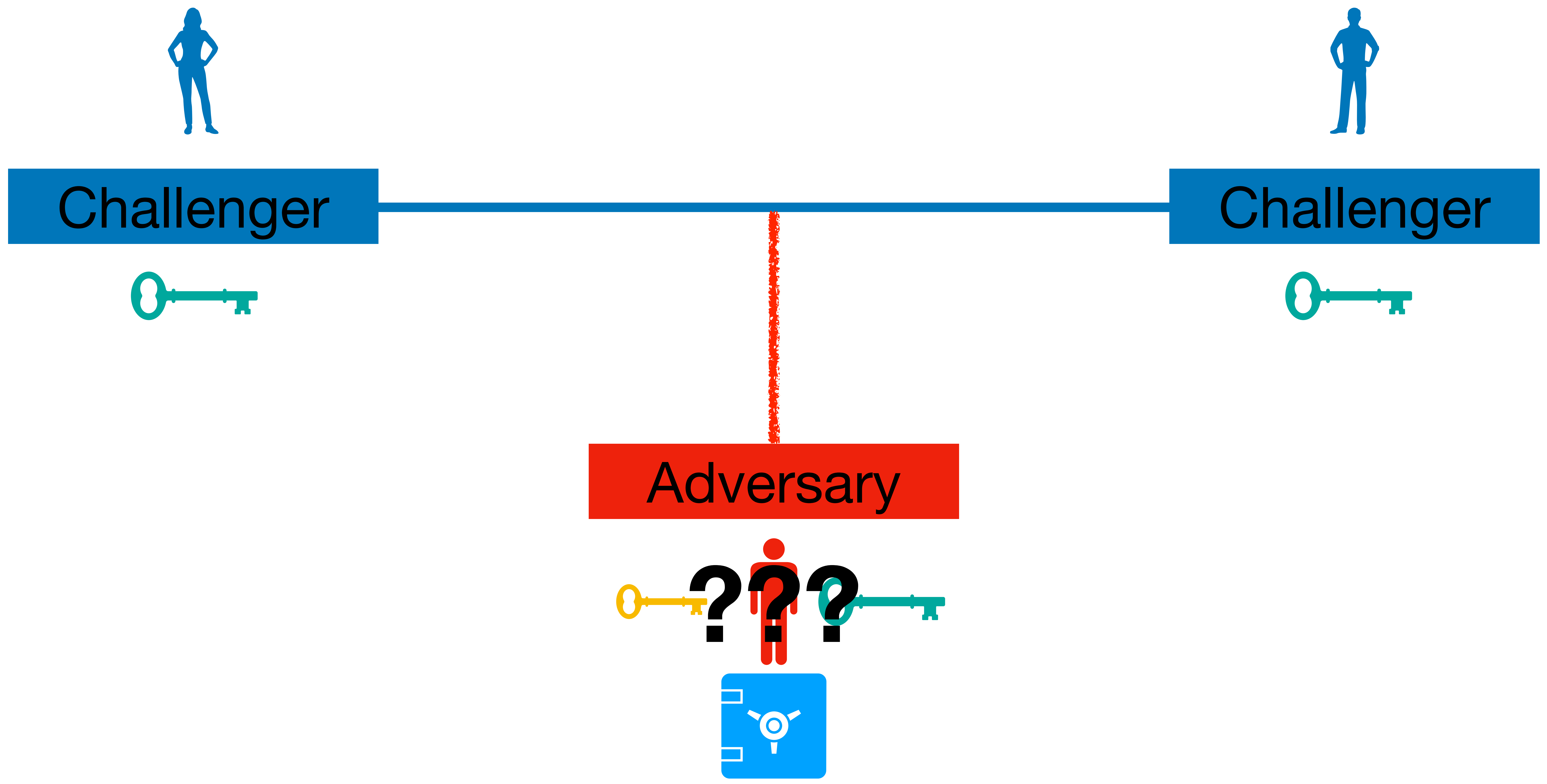
LRI Security



LRI Security



LRI Security



BGKNPR LRI One-Time Pad

BGKNPR LRI One-Time Pad

- $Setup() \rightarrow$ Randomly generate sk.

BGKNPR LRI One-Time Pad

- $Setup() \to$ Randomly generate sk.
- $Enc(sk, m) \to$

BGKNPR LRI One-Time Pad

- $Setup() \rightarrow$ Randomly generate sk.
- $Enc(sk, m) \rightarrow$
  1. Randomly generate c_0

BGKNPR LRI One-Time Pad

- $Setup() \rightarrow$ Randomly generate sk.

- $Enc(sk, m) \rightarrow$

    1. Randomly generate c_0

    2. Run $s = Ext_{c_0}(sk)$

BGKNPR LRI One-Time Pad

- $Setup() \rightarrow$ Randomly generate sk.
- $Enc(sk, m) \rightarrow$
    1. Randomly generate c_0
    2. Run $s = Ext_{c_0}(sk)$
    3. Compute $c_1 = IncompEnc(s, m)$ (Dziembowski's Incompressible SKE)

BGKNPR LRI One-Time Pad

- $Setup() \rightarrow$ Randomly generate sk.
- $Enc(sk, m) \rightarrow$
  1. Randomly generate c_0
  2. Run $s = Ext_{c_0}(sk)$
  3. Compute $c_1 = IncompEnc(s, m)$ (Dziembowski's Incompressible SKE)
  4. Return (c_0, c_1)

BGKNPR LRI One-Time Pad

- $Setup() \rightarrow$ Randomly generate sk.
- $Enc(sk, m) \rightarrow$
  1. Randomly generate c_0
  2. Run $s = Ext_{c_0}(sk)$
  3. Compute $c_1 = IncompEnc(s, m)$ (Dziembowski's Incompressible SKE)
  4. Return (c_0, c_1)
- $Dec(sk, (c_0, c_1)) \rightarrow$ Run $s = Ext_{c_0}(sk)$ and return $IncompDec(s, c_1)$.

BGKNPR Results

BGKNPR Results

- Transformation from LRI SKE + PKE to LRI PKE using garbling techniques.

BGKNPR Results

- Transformation from LRI SKE + PKE to LRI PKE using garbling techniques.
- Transformation from Incomp SKE to LRI PKE using an advanced novel LR primitive.

BGKNPR Results

- Transformation from LRI SKE + PKE to LRI PKE using garbling techniques.
- Transformation from Incomp SKE to LRI PKE using an advanced novel LR primitive.
- Impossibility of building provably secure Incomp SKE/PKE with small keys and small ciphertexts.

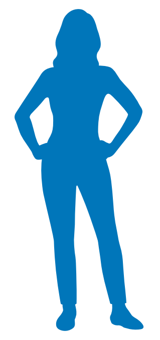
BGKNPR Results

- Transformation from LRI SKE + PKE to LRI PKE using garbling techniques.
- Transformation from Incomp SKE to LRI PKE using an advanced novel LR primitive.
- Impossibility of building provably secure Incomp SKE/PKE with small keys and small ciphertexts.
- Impossibility of building provably secure LRI SKE/PKE with large secret key leakage and small cipher texts.

Incompressible Functional Encryption

(Public Key) Identity Based Encryption

(Public Key) Identity Based Encryption



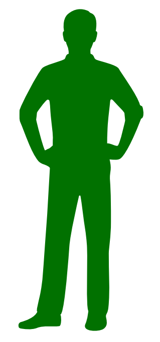
(Public Key) Identity Based Encryption

(Public Key) Identity Based Encryption

(Public Key) Identity Based Encryption

(Public Key) Identity Based Encryption

sk_A

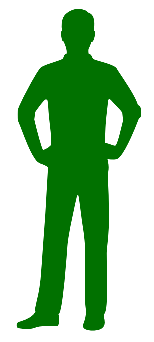


pk_A

sk_B

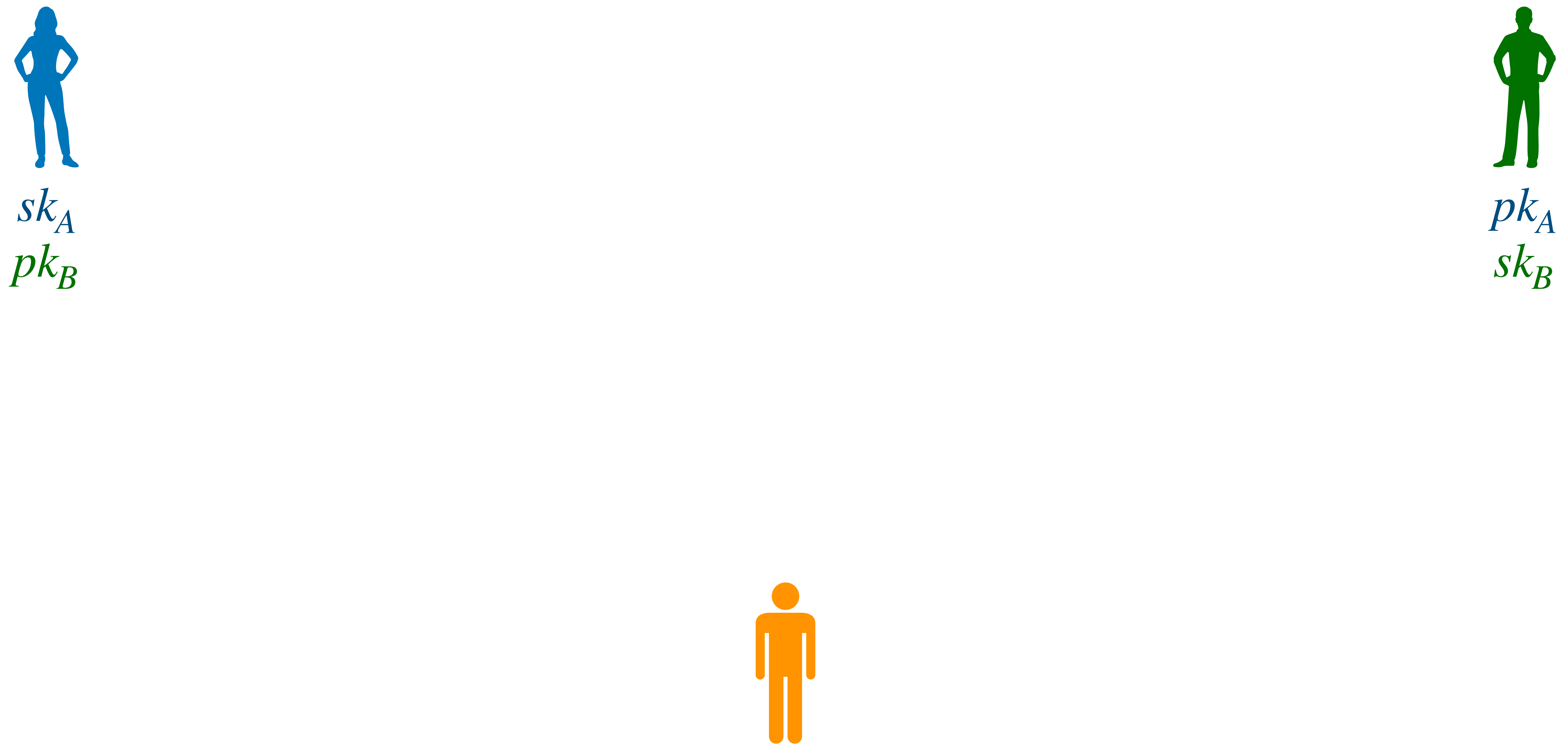
(Public Key) Identity Based Encryption

sk_A
pk_B

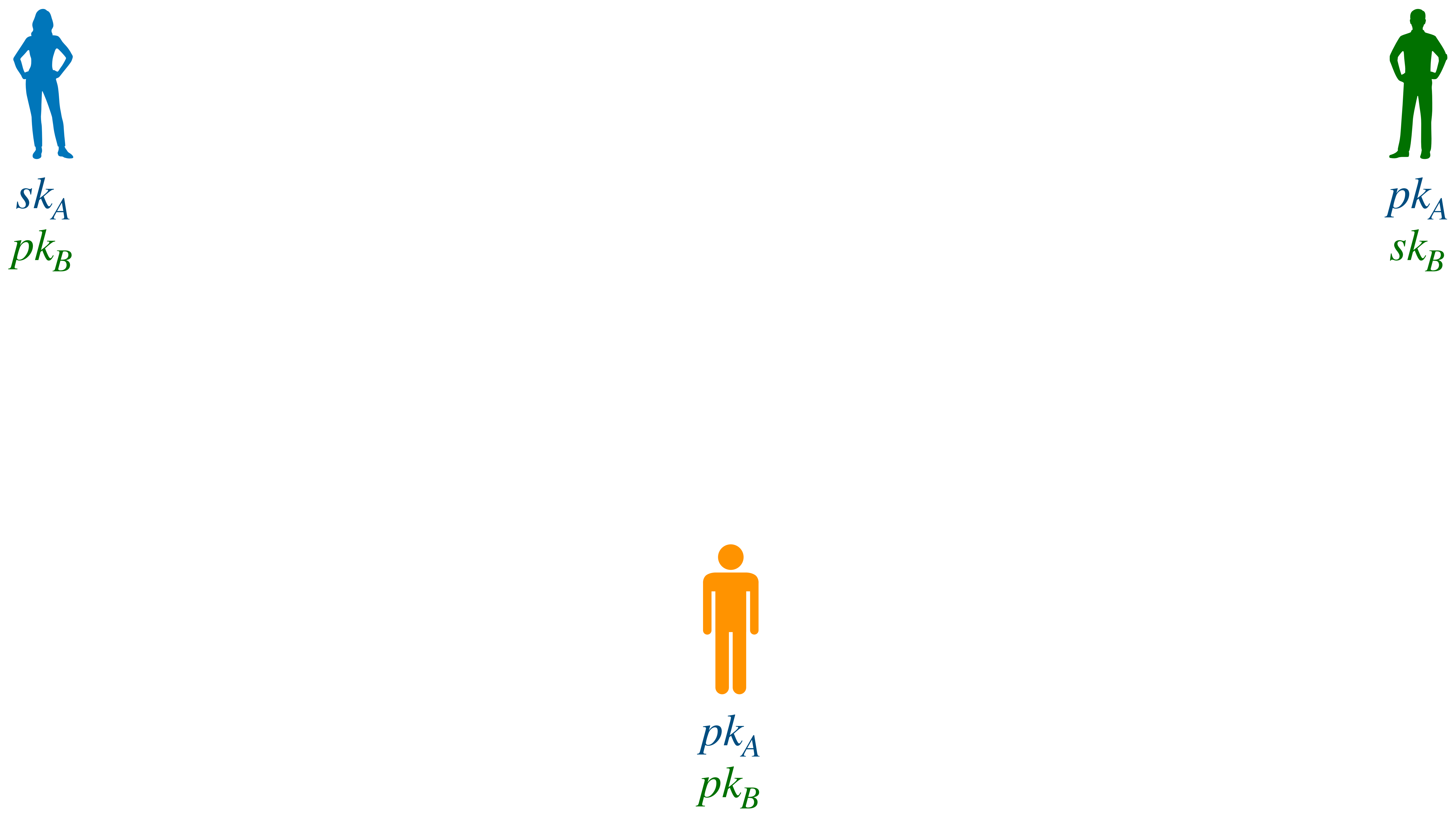


pk_A
sk_B

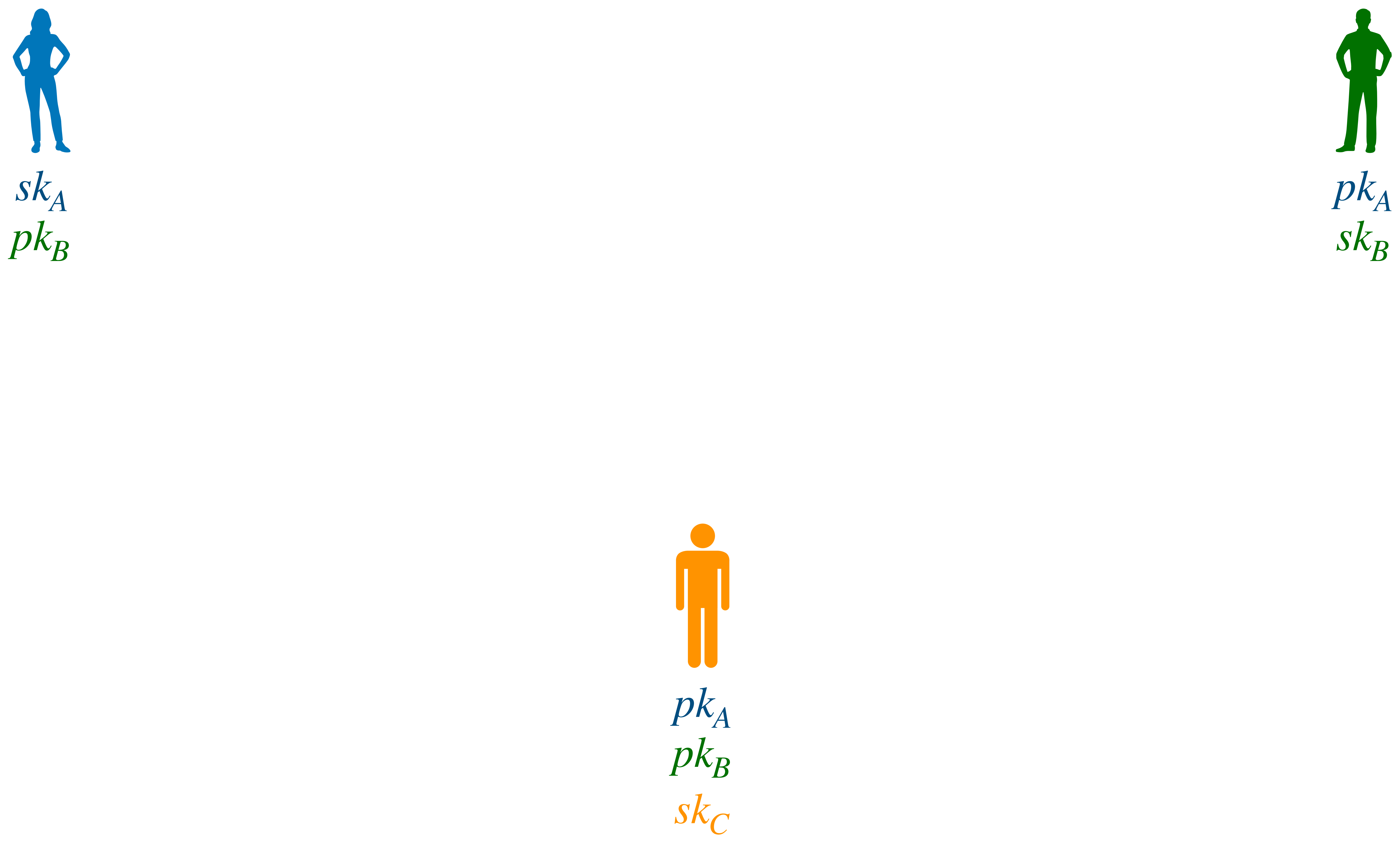
(Public Key) Identity Based Encryption



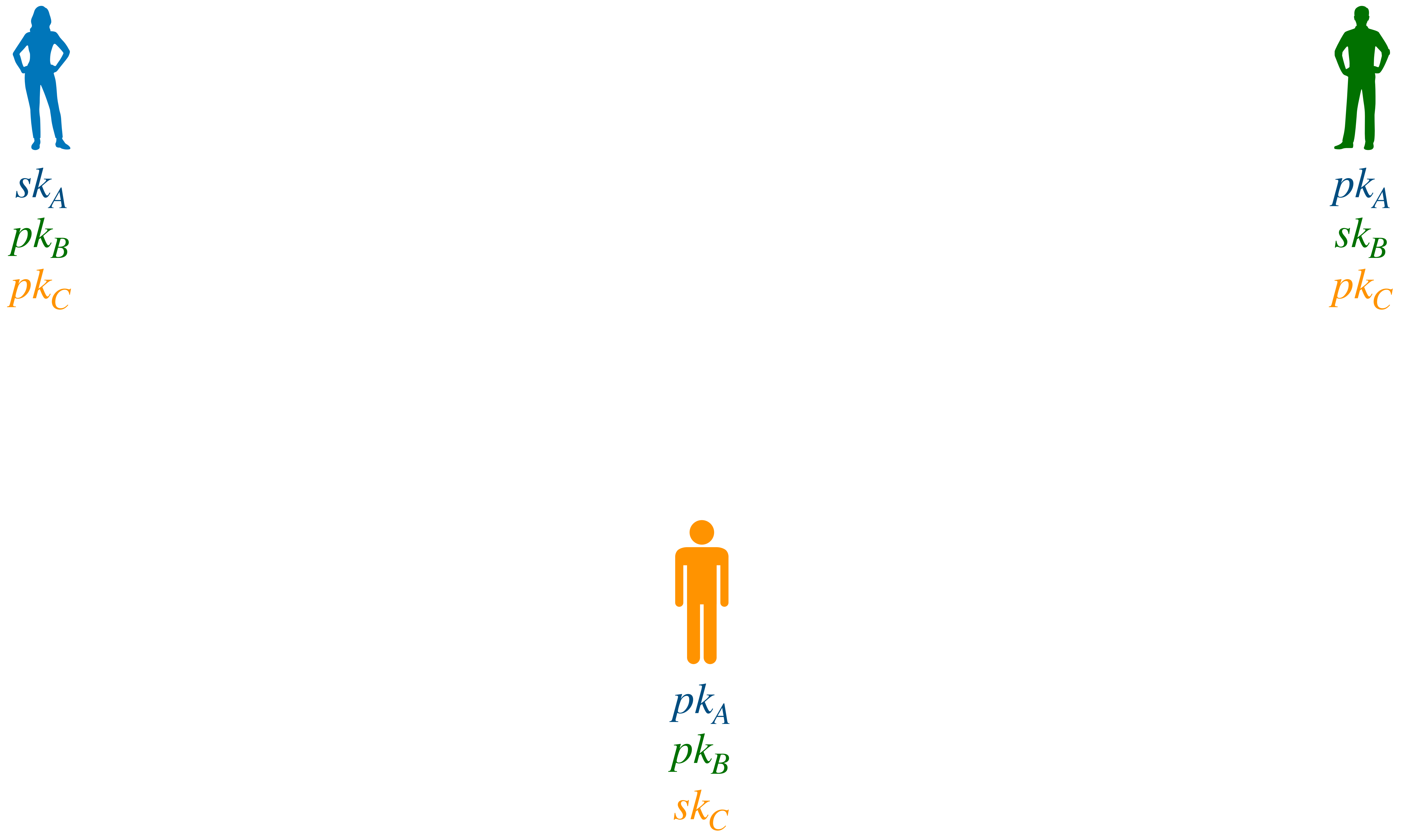
(Public Key) Identity Based Encryption



(Public Key) Identity Based Encryption



(Public Key) Identity Based Encryption



(Public Key) Identity Based Encryption

sk_A
pk_B
pk_C

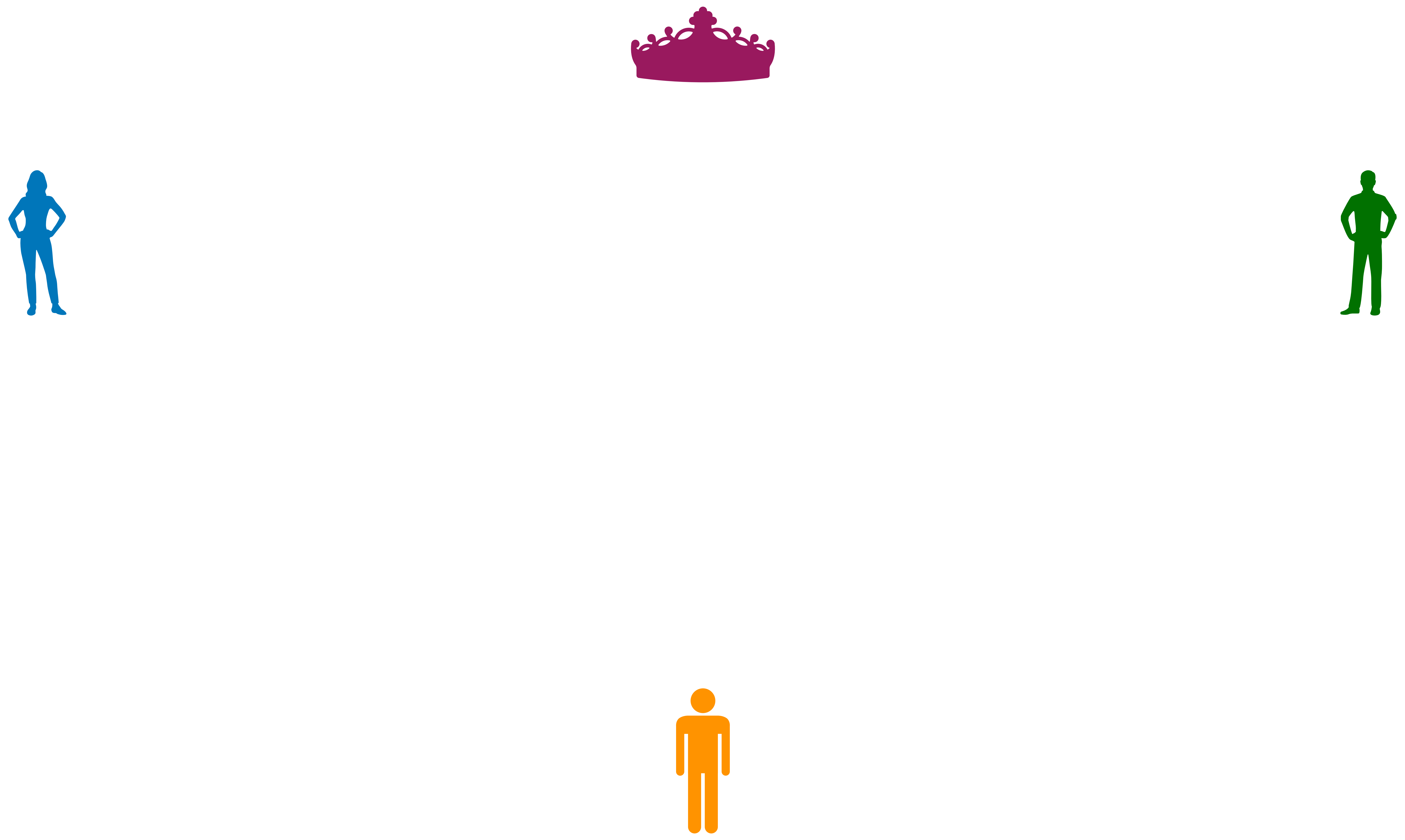
pk_A
sk_B
pk_C

In a system of n users, if a new user joins, it needs to perform $2n$ communications.

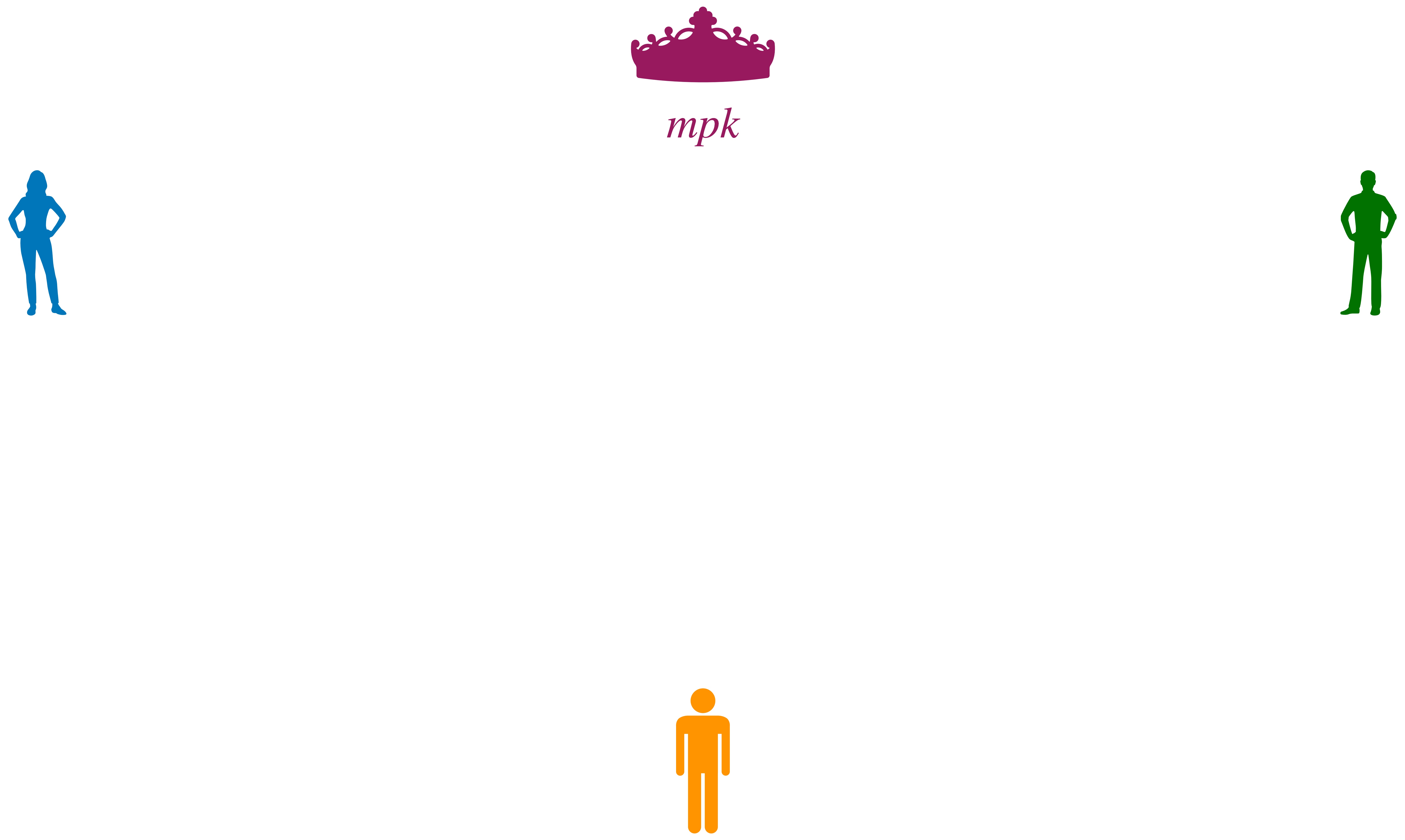
pk_A
pk_B
sk_C

Identity Based Encryption (IBE)

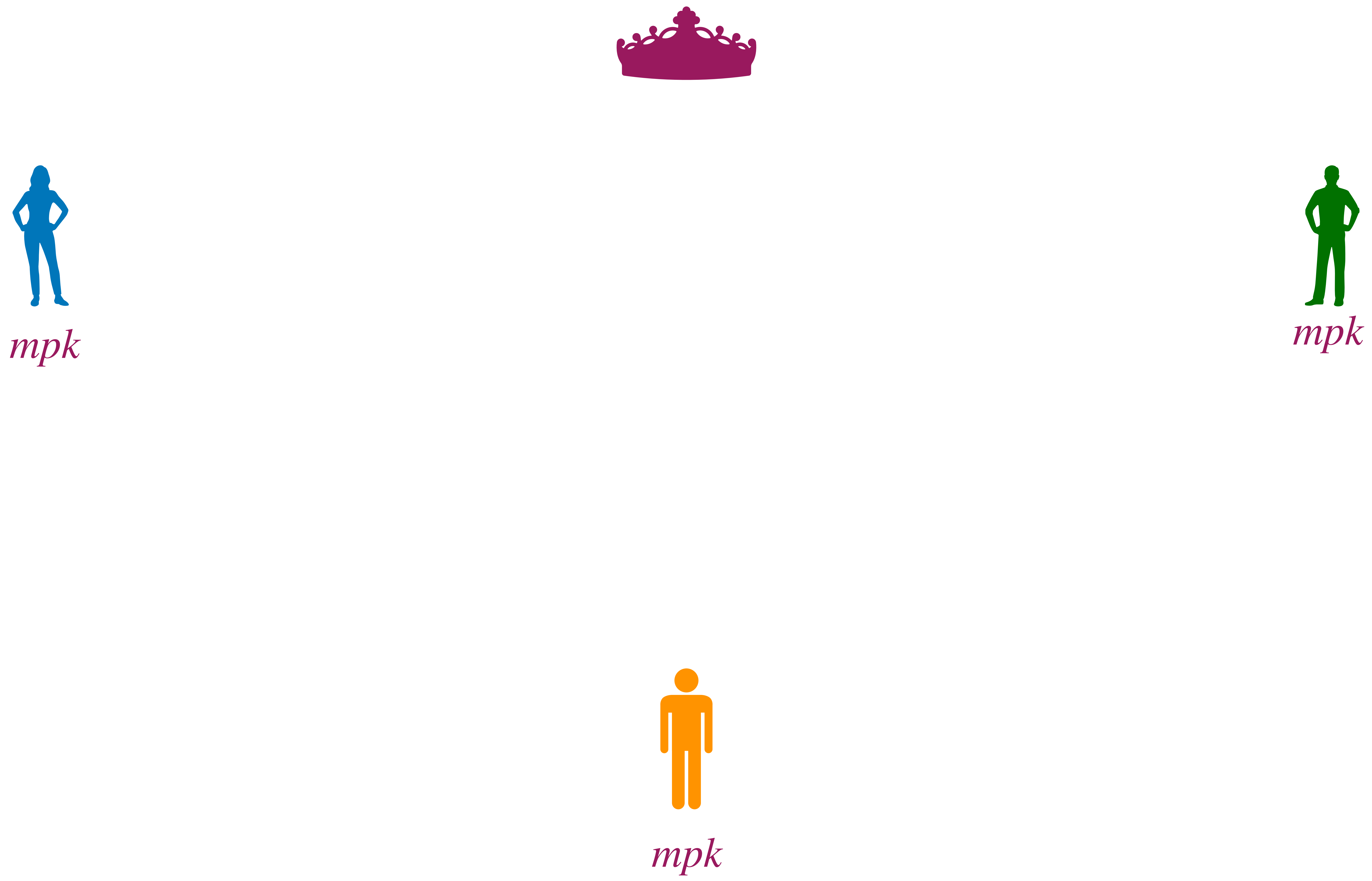
Identity Based Encryption (IBE)



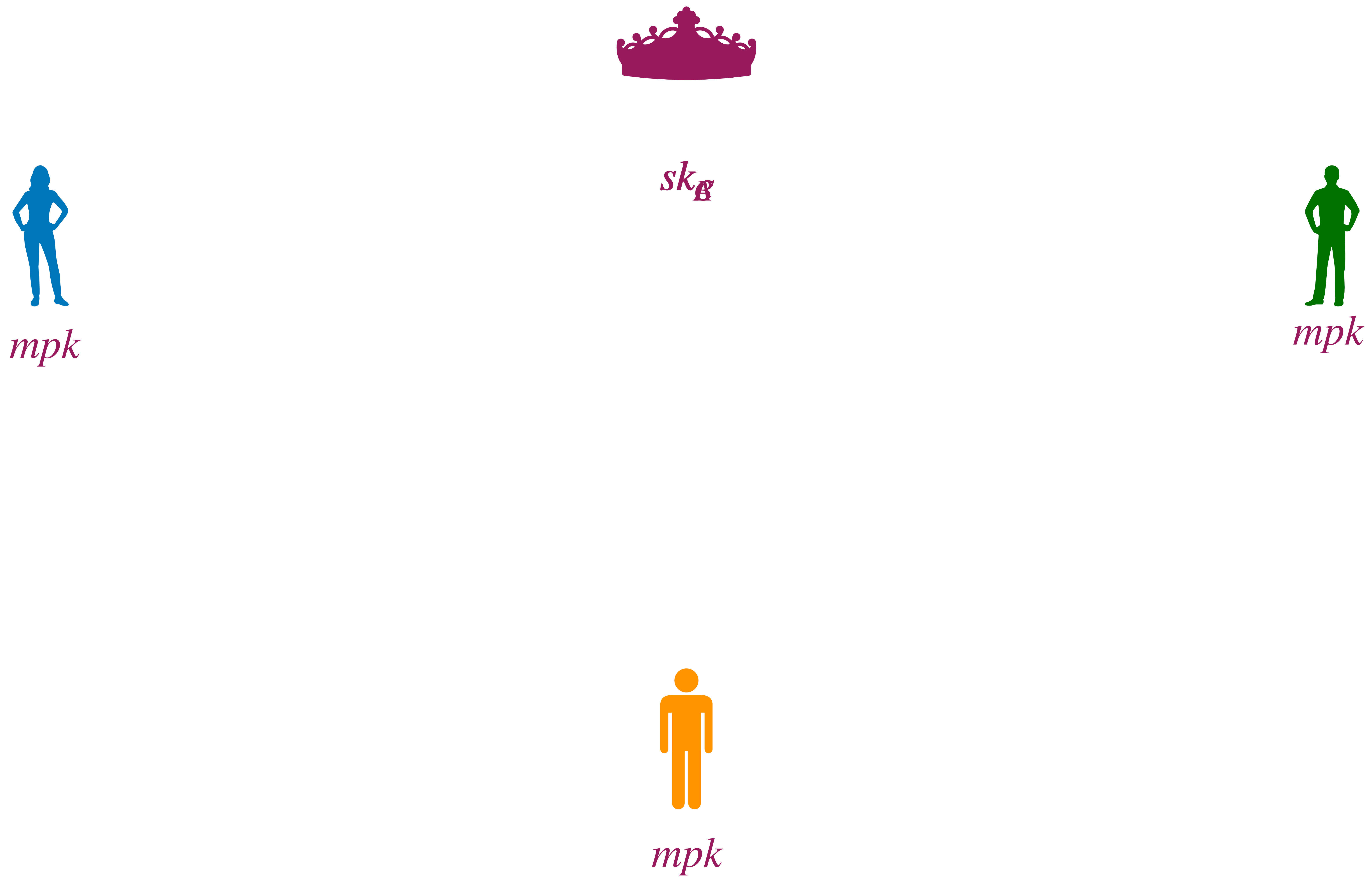
Identity Based Encryption (IBE)



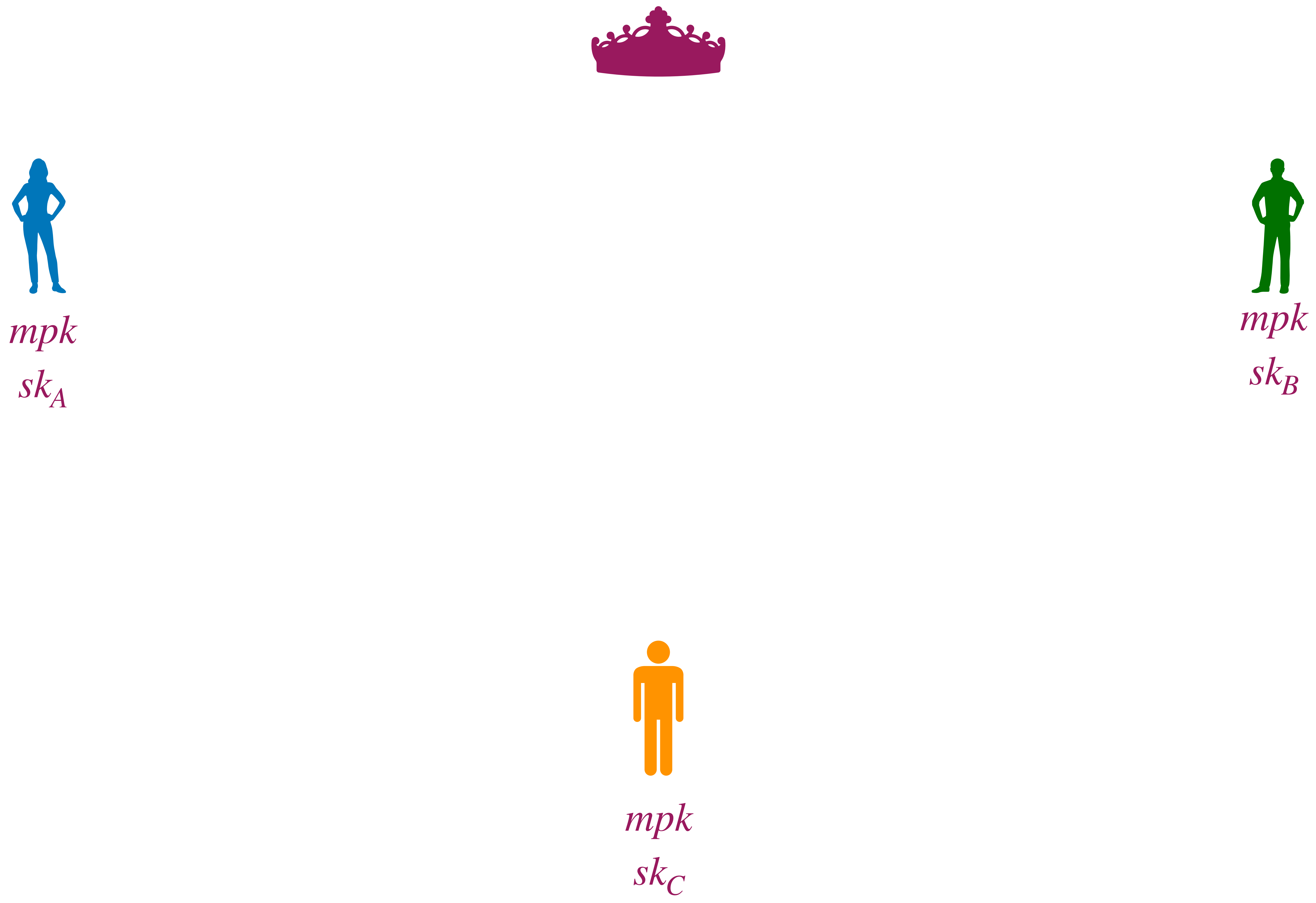
Identity Based Encryption (IBE)



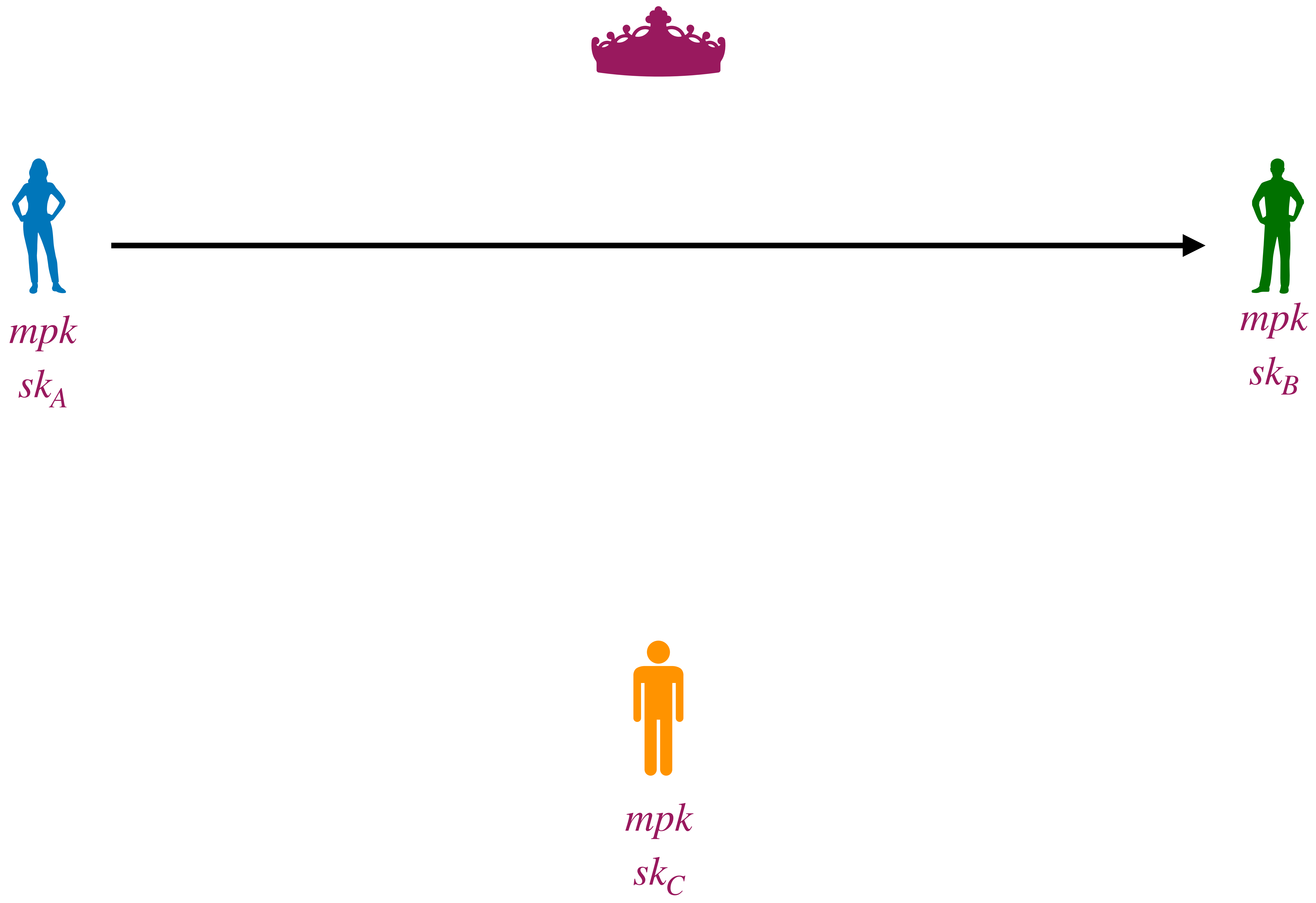
Identity Based Encryption (IBE)



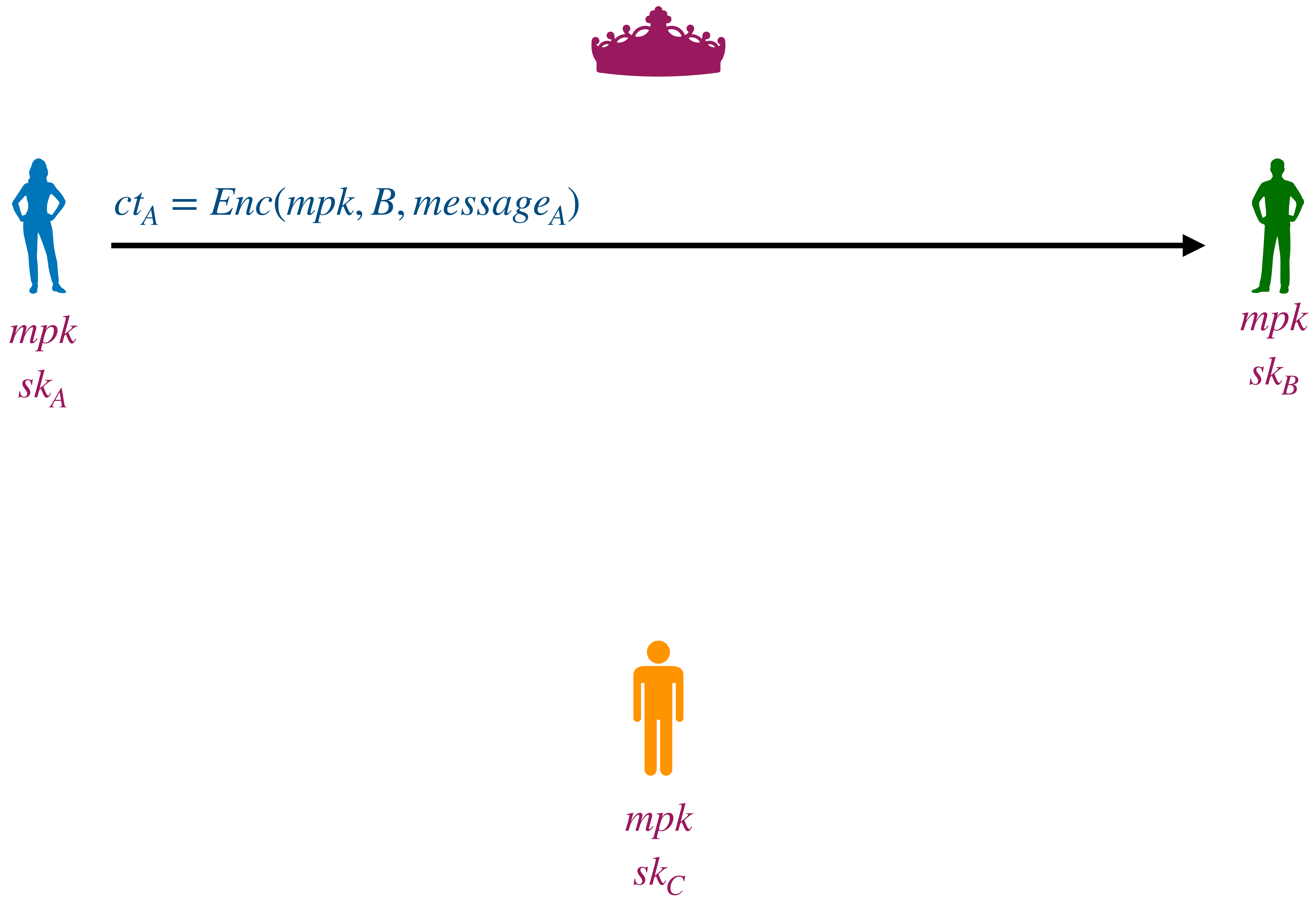
Identity Based Encryption (IBE)



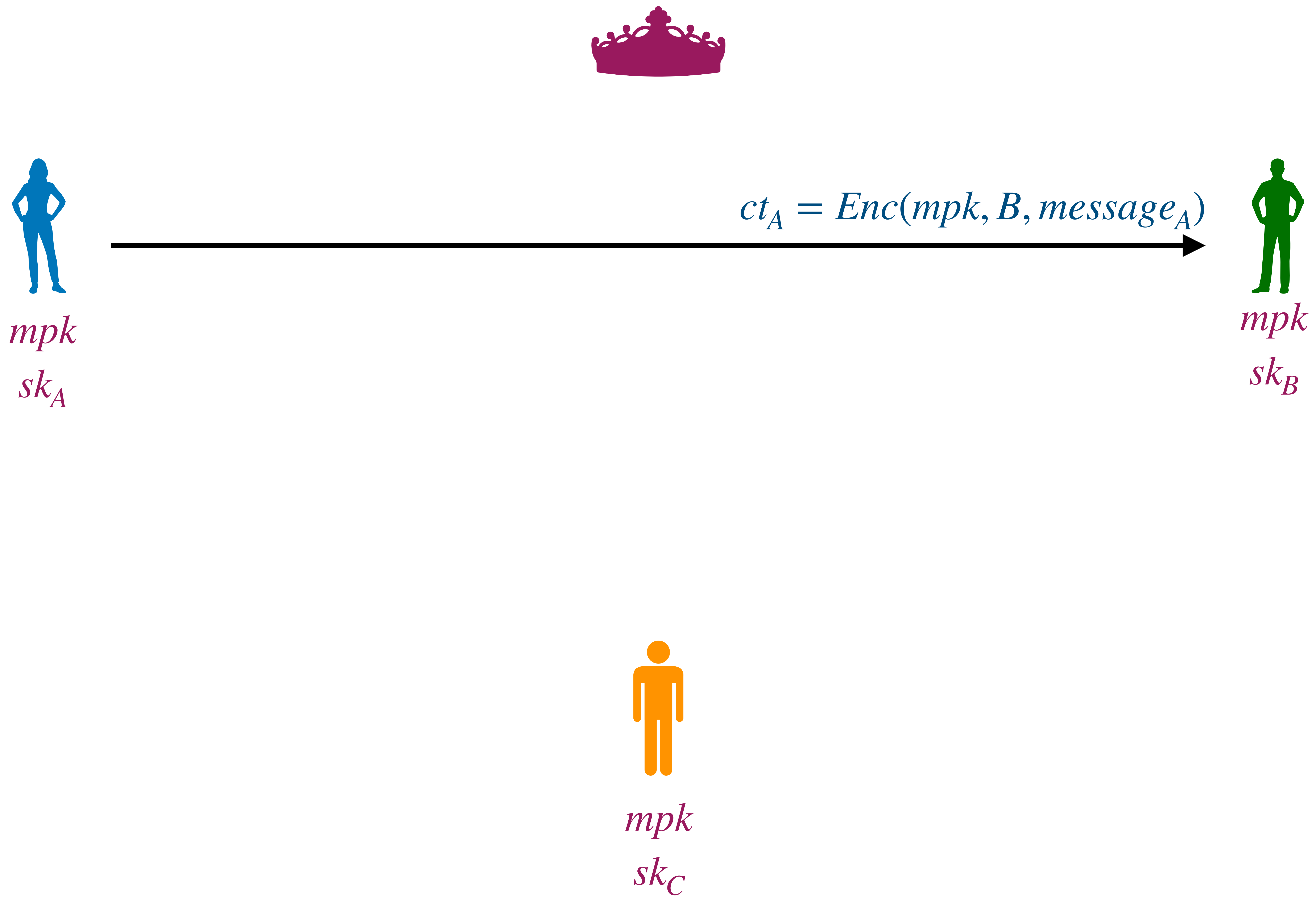
Identity Based Encryption (IBE)



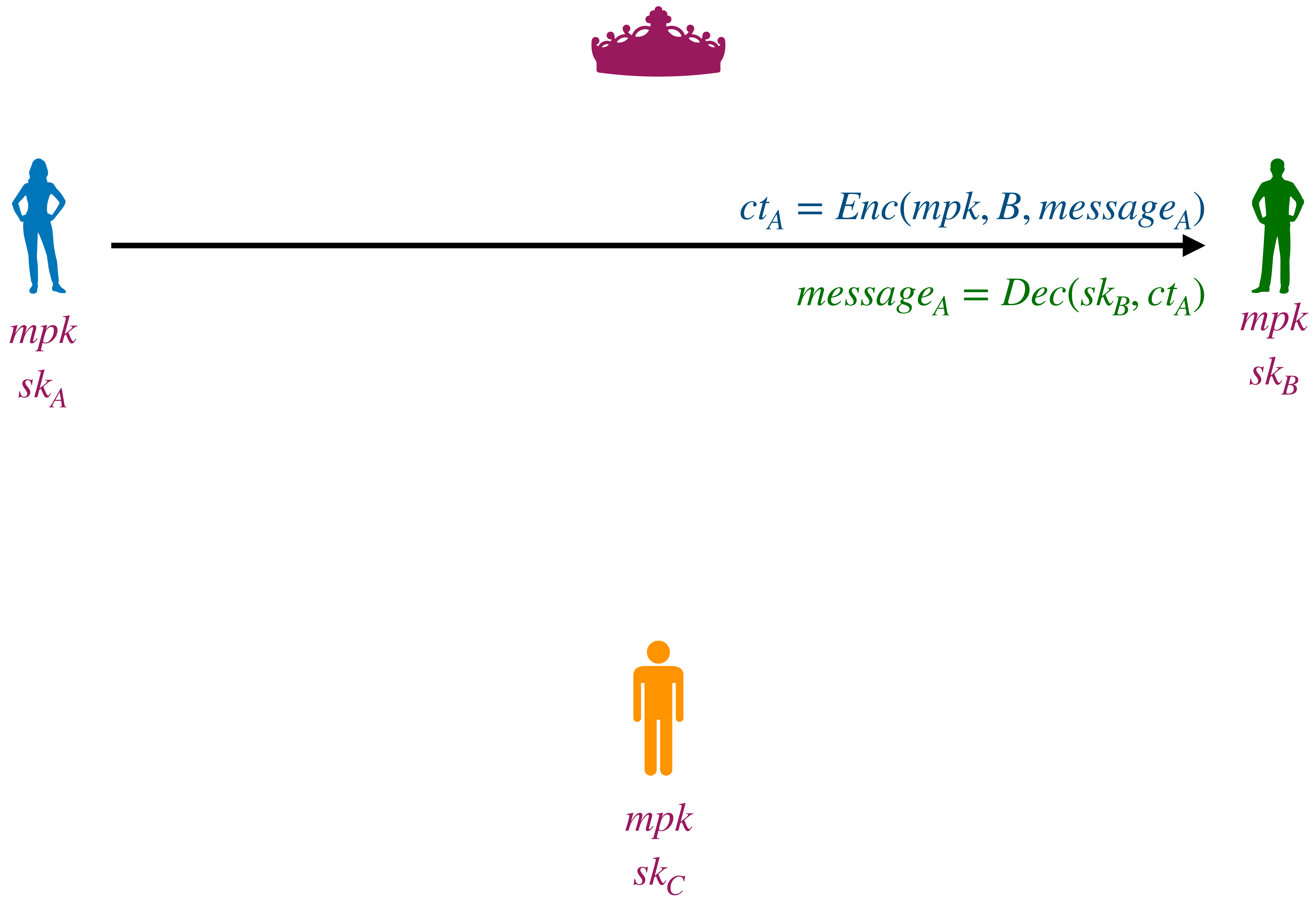
Identity Based Encryption (IBE)



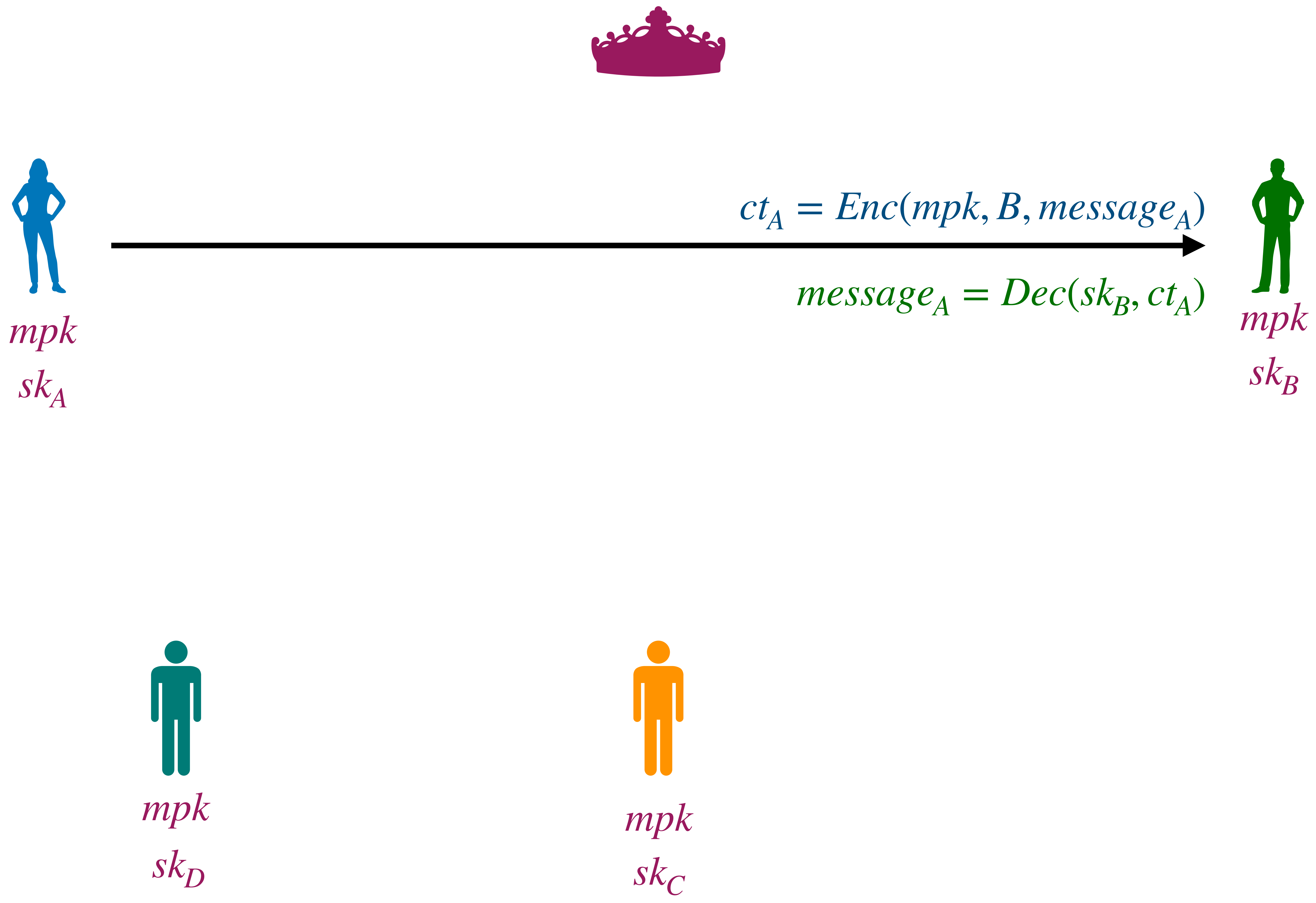
Identity Based Encryption (IBE)



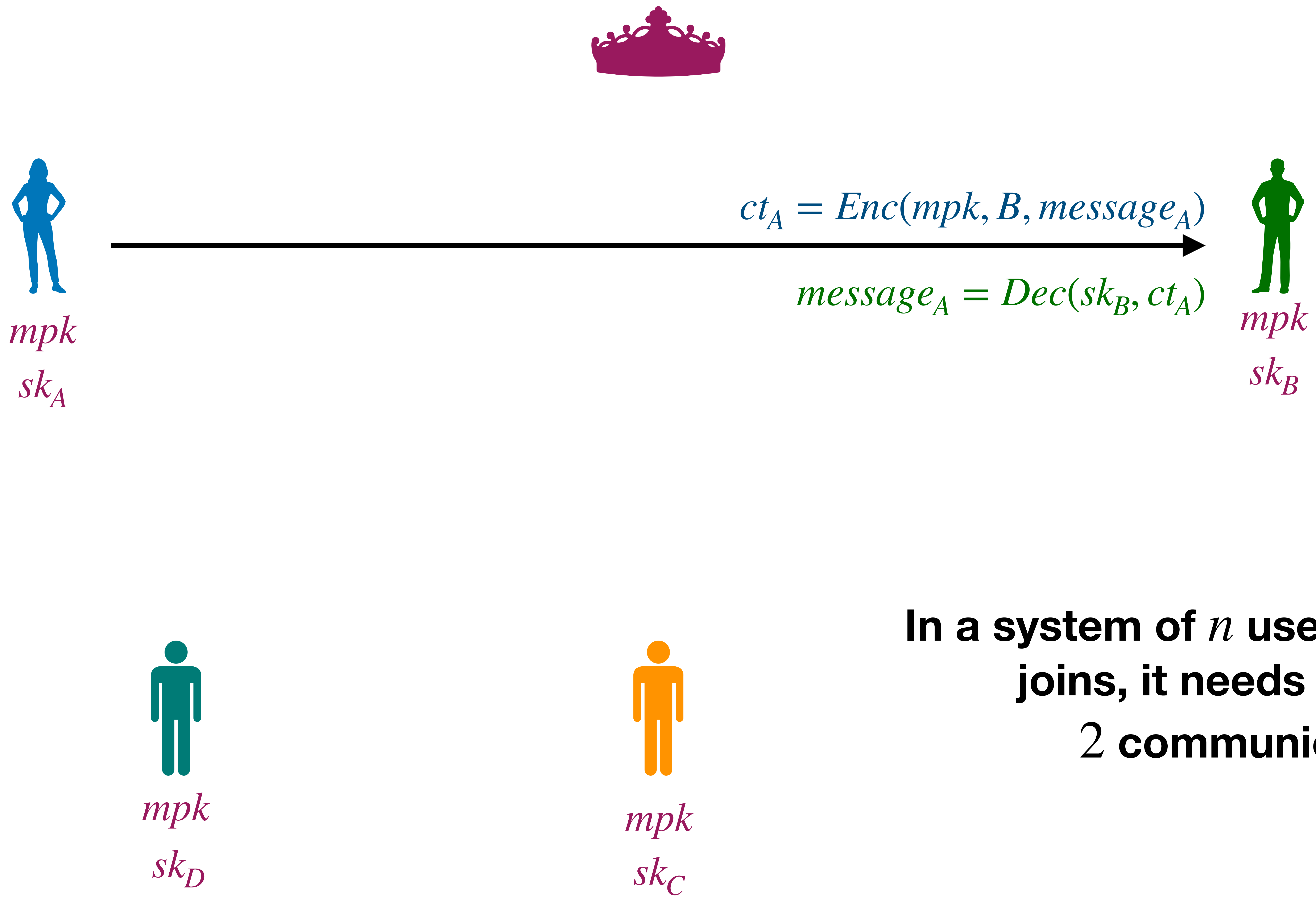
Identity Based Encryption (IBE)


$ct_A = Enc(mpk, B, message_A)$
$message_A = Dec(sk_B, ct_A)$
mpk
sk_A
mpk
sk_B
mpk
sk_C

Identity Based Encryption (IBE)


$ct_A = Enc(mpk, B, message_A)$
$message_A = Dec(sk_B, ct_A)$
mpk
sk_A
mpk
sk_B
mpk
sk_D
mpk
sk_C

Identity Based Encryption (IBE)

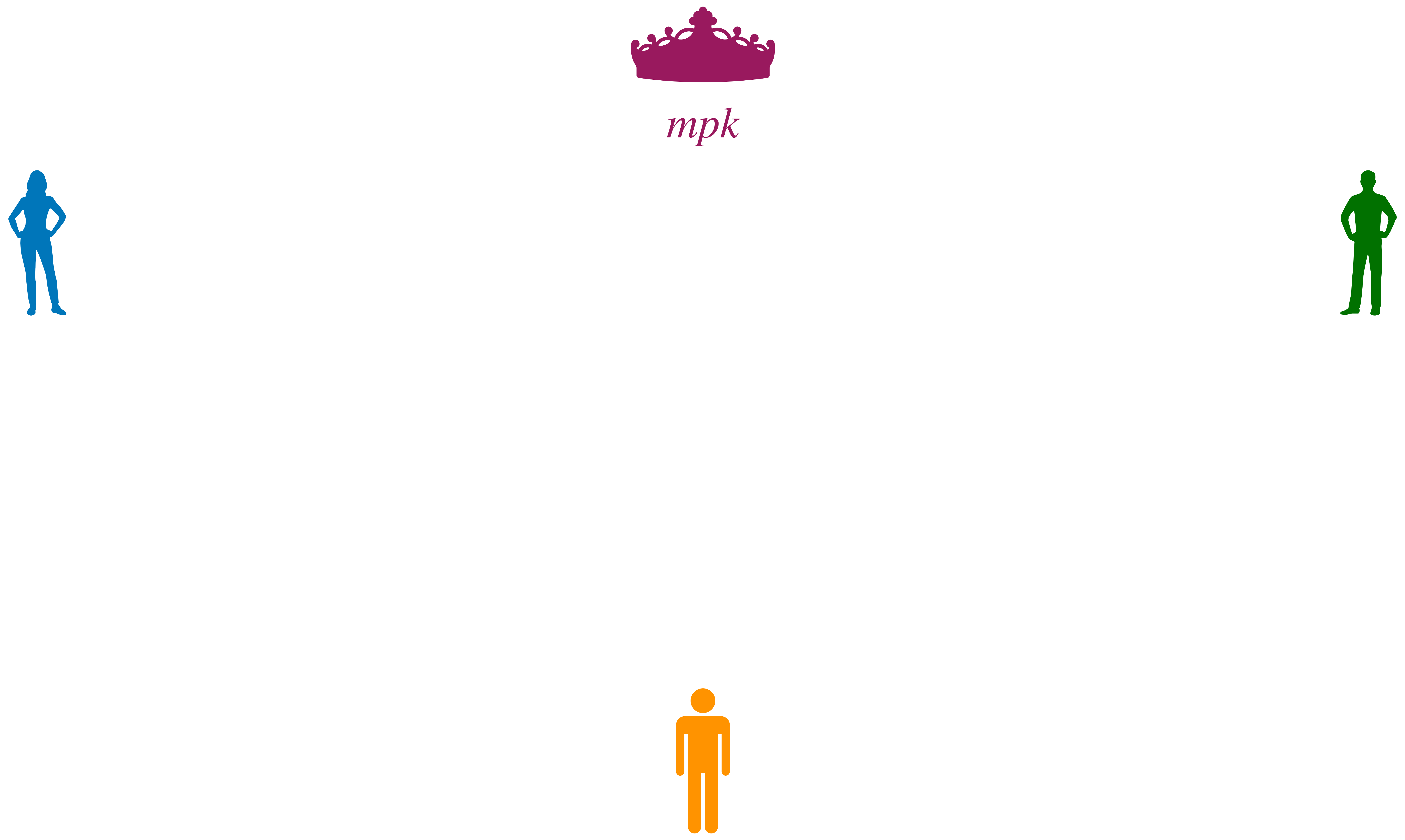

$ct_A = Enc(mpk, B, message_A)$
$message_A = Dec(sk_B, ct_A)$
mpk
sk_A
mpk
sk_B
mpk
sk_D
mpk
sk_C

In a system of n users, if a new user joins, it needs to perform 2 communications.

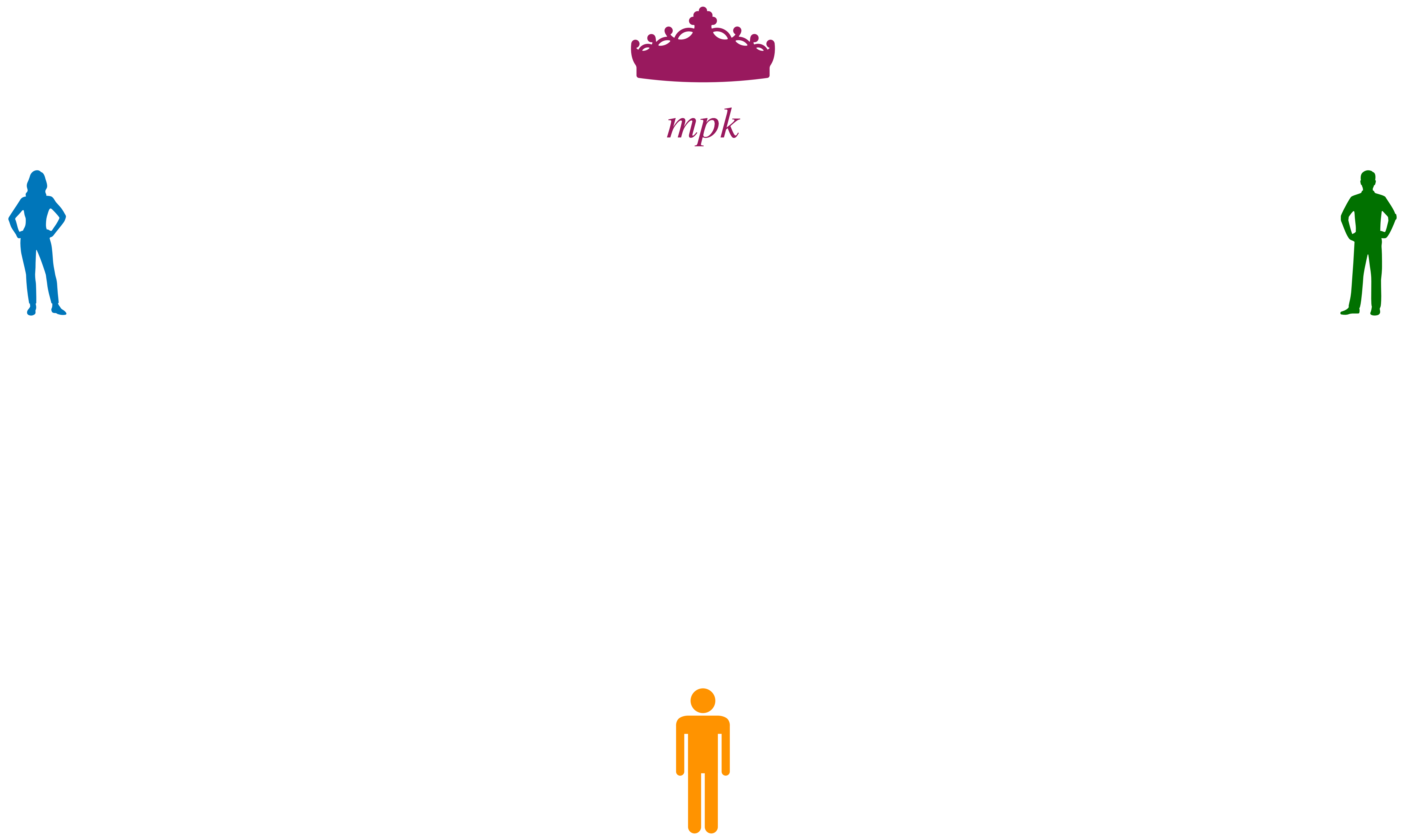
Functional Encryption (FE)

mpk

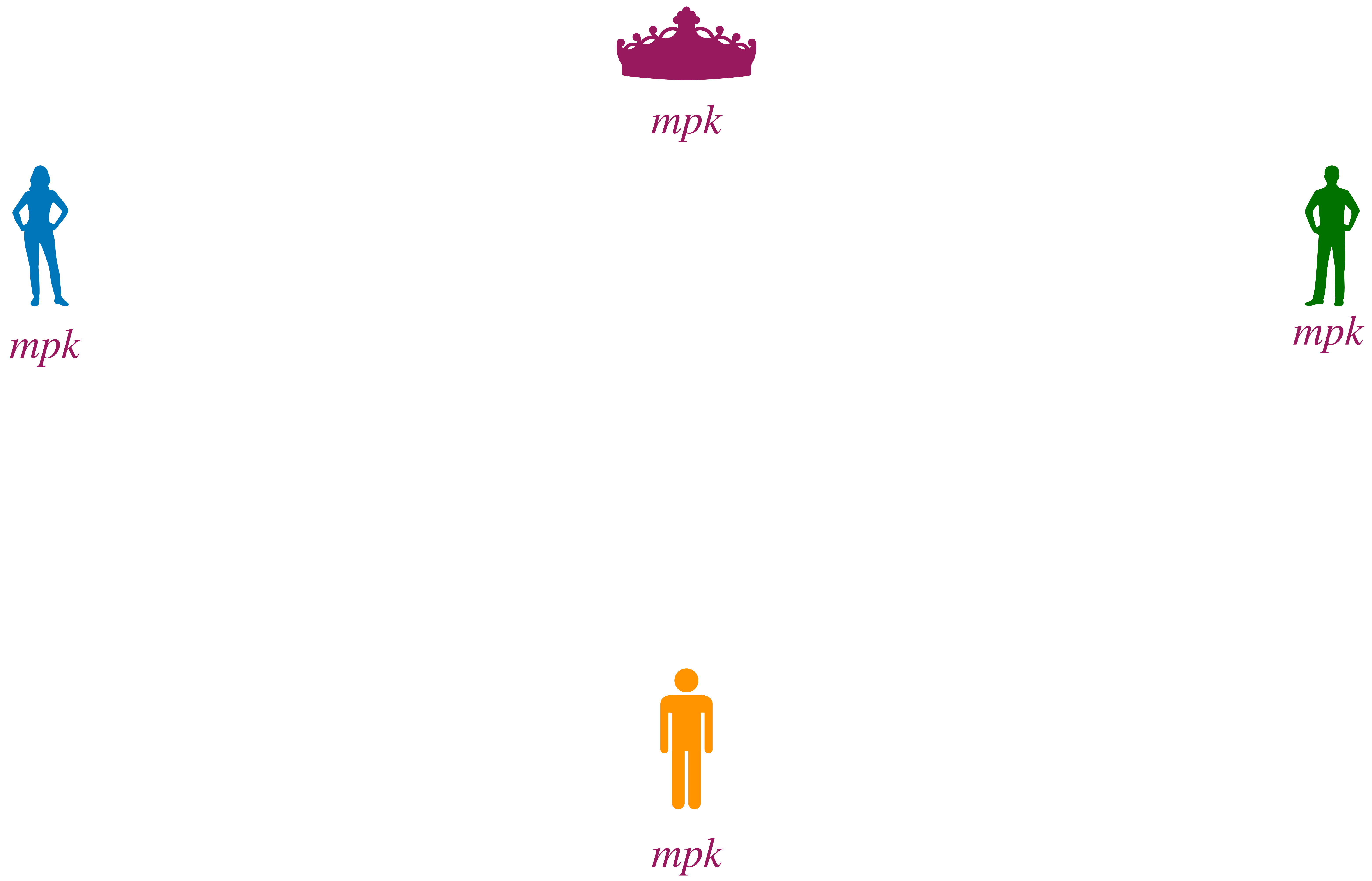
Functional Encryption (FE)



Functional Encryption (FE)



Functional Encryption (FE)



Functional Encryption (FE)

mpk

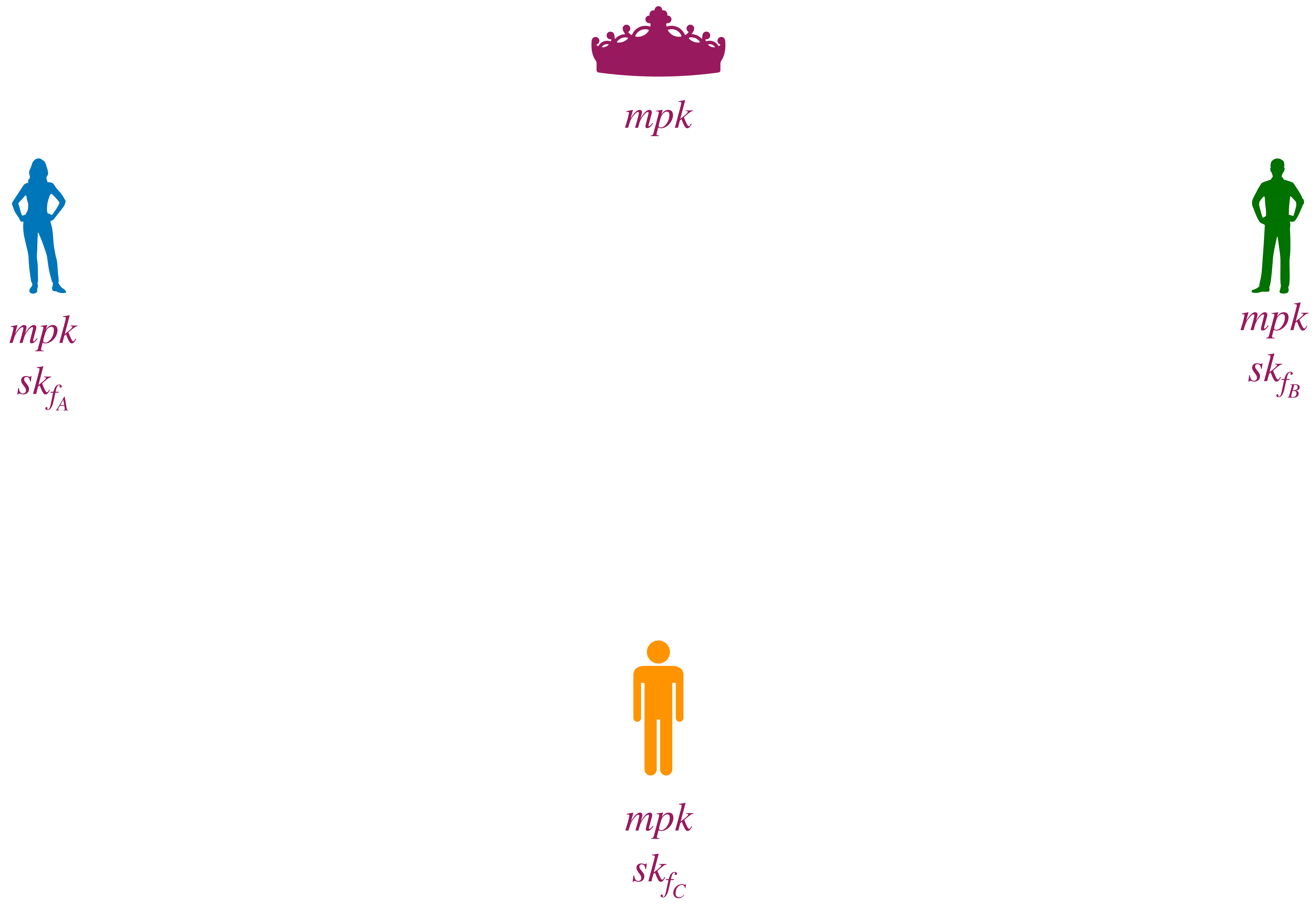
sk_{f_A} sk_{f_B} sk_{f_C}

mpk

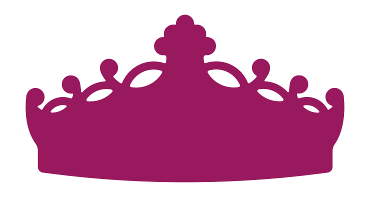
mpk

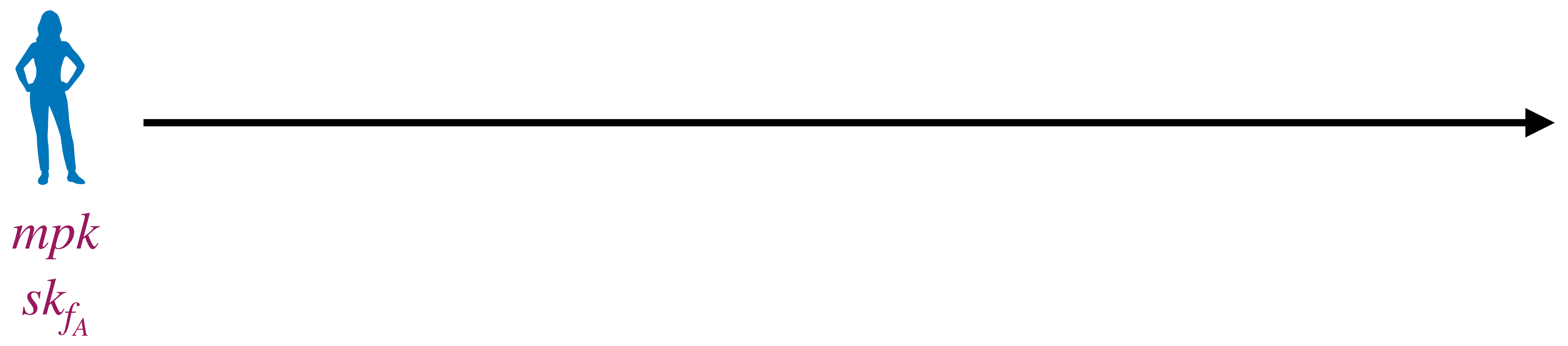
mpk

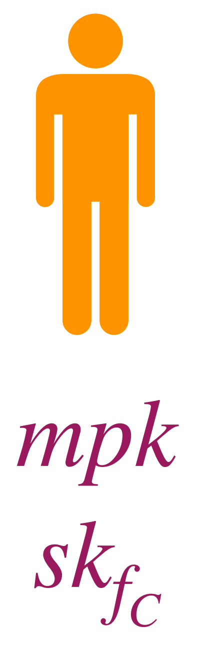
Functional Encryption (FE)



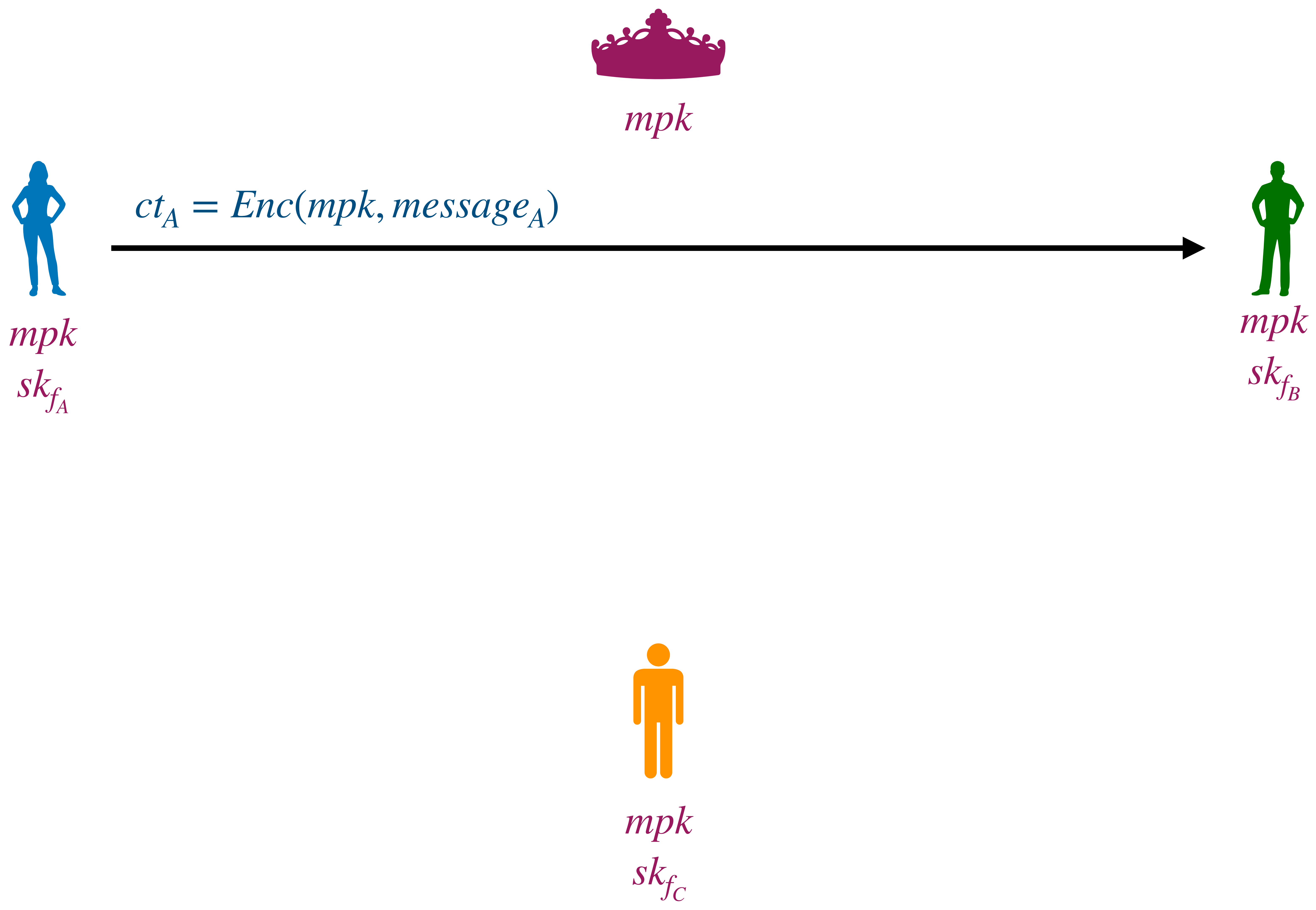
Functional Encryption (FE)



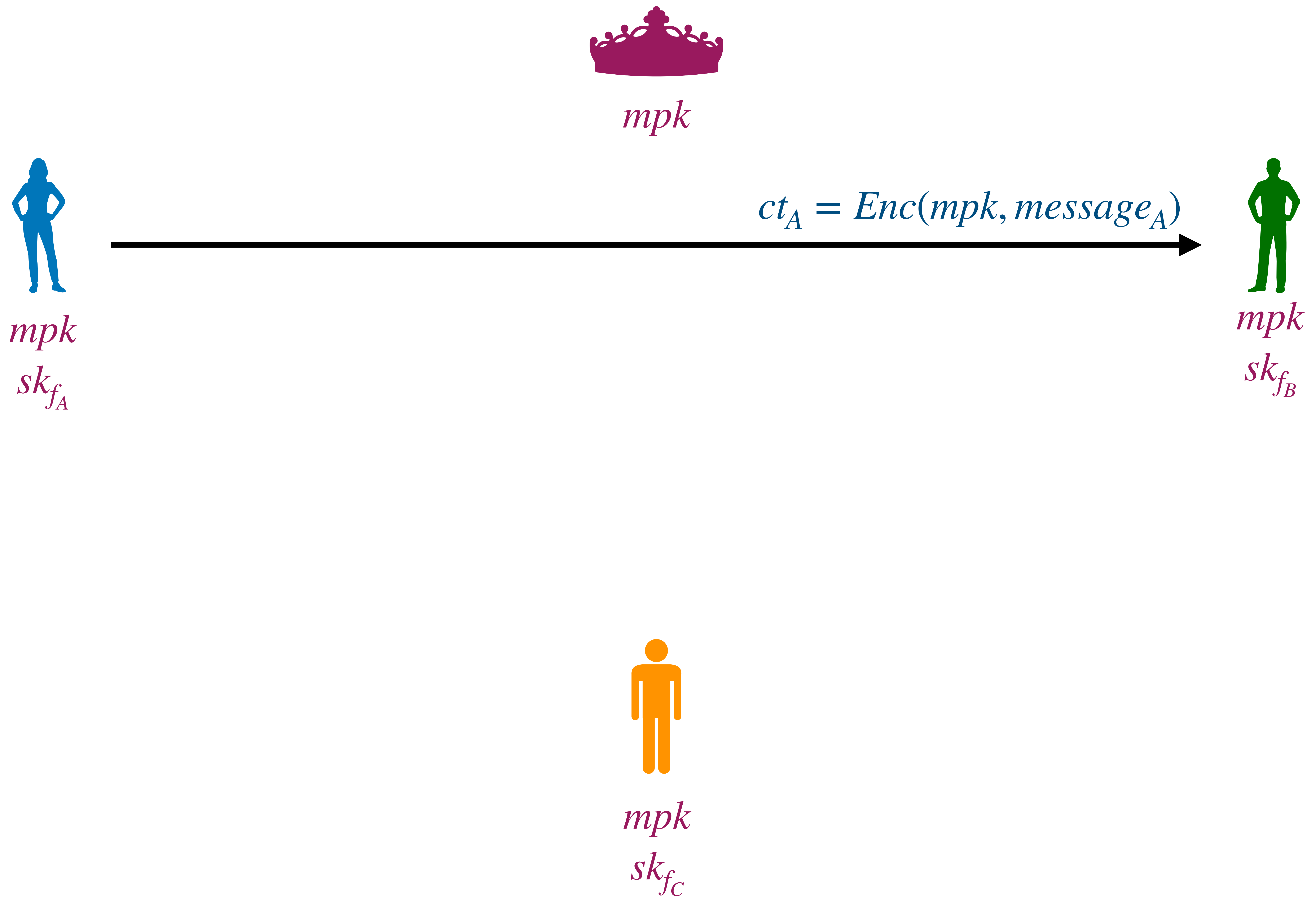




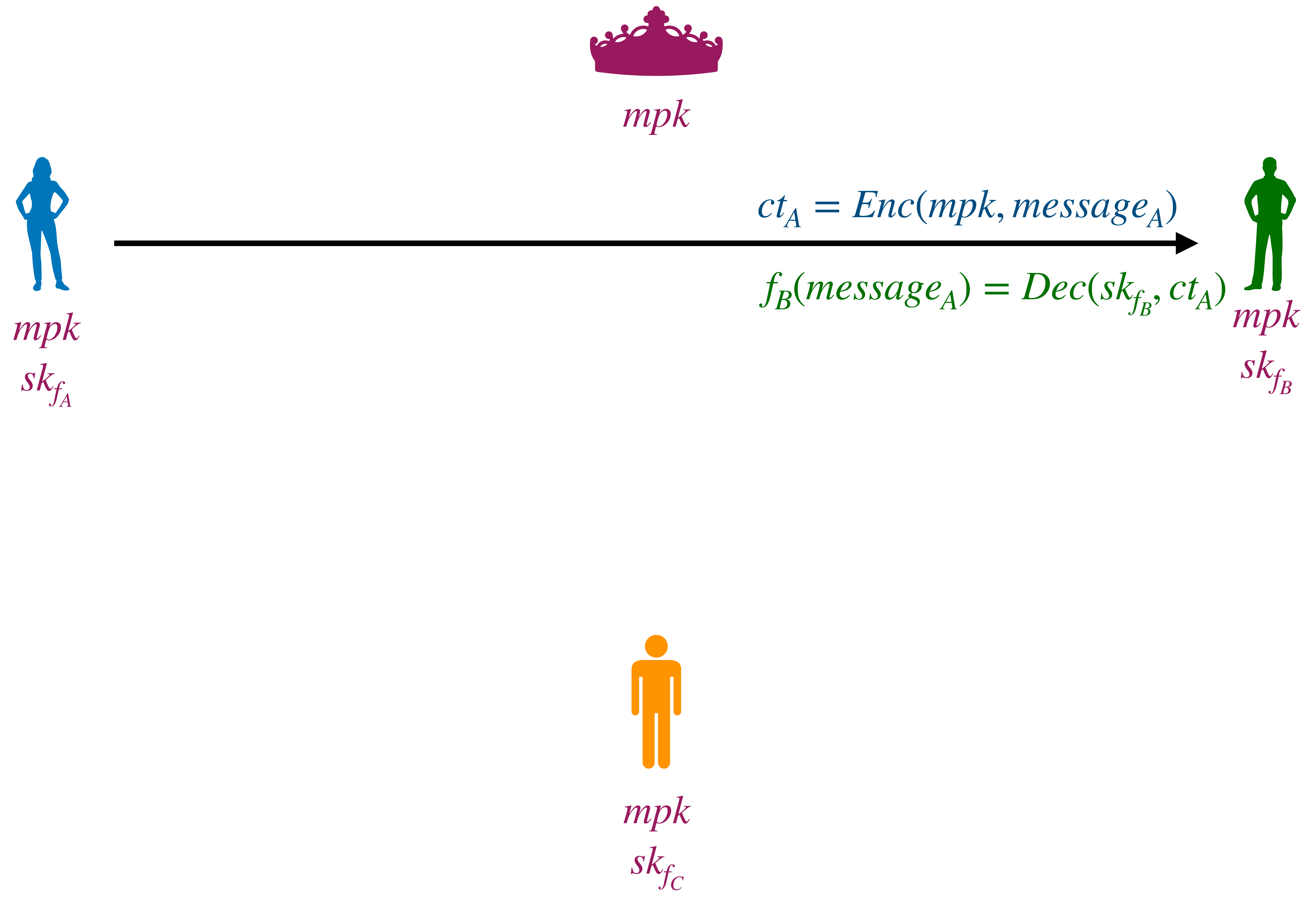
Functional Encryption (FE)



Functional Encryption (FE)



Functional Encryption (FE)



$ct_A = Enc(mpk, message_A)$

$f_B(message_A) = Dec(sk_{f_B}, ct_A)$

GKRV Results

GKRV Results

- Formally defined Incompressible FE (IBE) security.

GKRV Results

- Formally defined Incompressible FE (IBE) security.
- Transformation from IBE + Incomp PKE to Incomp IBE.

GKRV Results

- Formally defined Incompressible FE (IBE) security.
- Transformation from IBE + Incomp PKE to Incomp IBE.
- Provided novel constructions for Incomp FE with (almost) optimal parameter.

Conclusion

Conclusion

- We saw standard, leakage-resilient, incompressible, LRI security.

Conclusion

- We saw standard, leakage-resilient, incompressible, LRI security.
- Constructions for the above.

Conclusion

- We saw standard, leakage-resilient, incompressible, LRI security.
- Constructions for the above.
- Presented IBE and FE.

Conclusion

- We saw standard, leakage-resilient, incompressible, LRI security.
- Constructions for the above.
- Presented IBE and FE.
- Mentioned possibilities and impossibilities in LRI and incompressibility settings.

Thank You!!!

Personal Webpage - https://mahe94.github.io/
Incompressible Functional Encryption - https://eprint.iacr.org/2024/798
Leakage-Resilient Incompressible Cryptography: Constructions and Barriers - ……..