数字货币和区块链 - 密码学（3）

山东大学网络空间安全学院

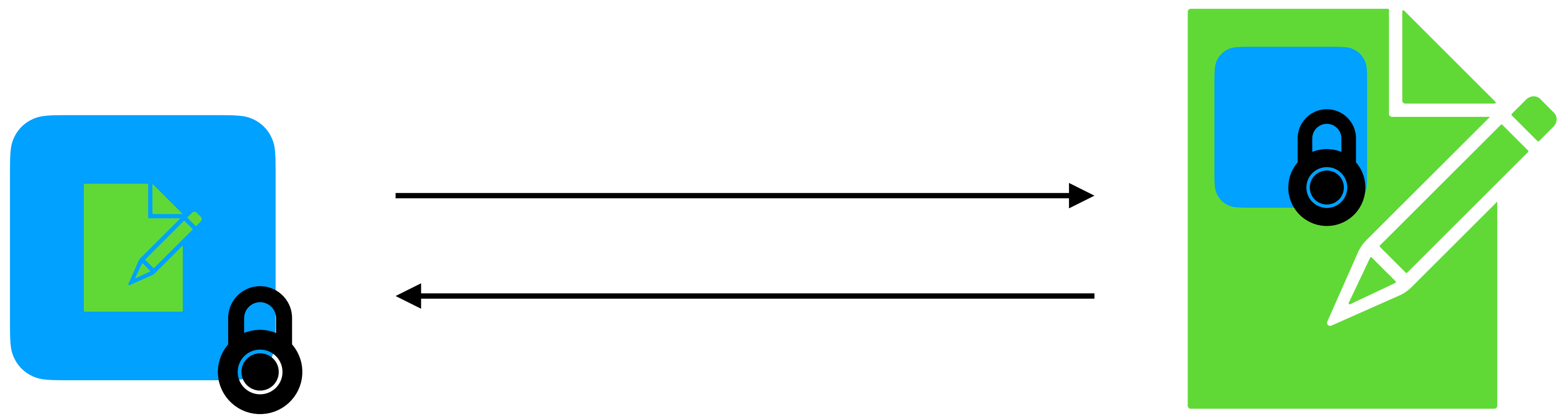
钱宸 2023年11月8日

密码学温故知新

- 哈希函数
- 零知识证明
- 签名
- 盲签名
- 加密算法

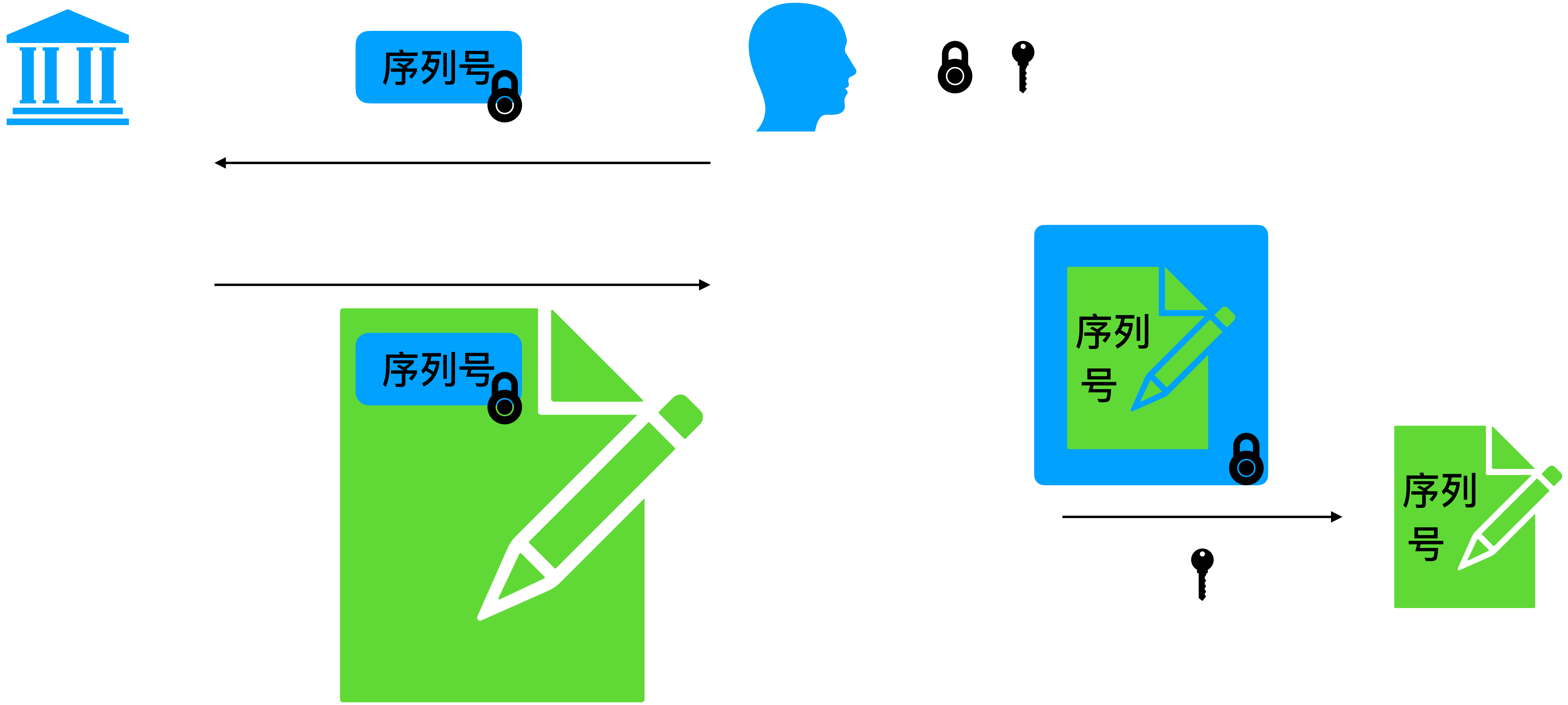
盲签名算法回顾

- 1988年David Chaum提出利用盲签名的方式来同时解决匿名性和双支付攻击
- 盲签名的重要特性：



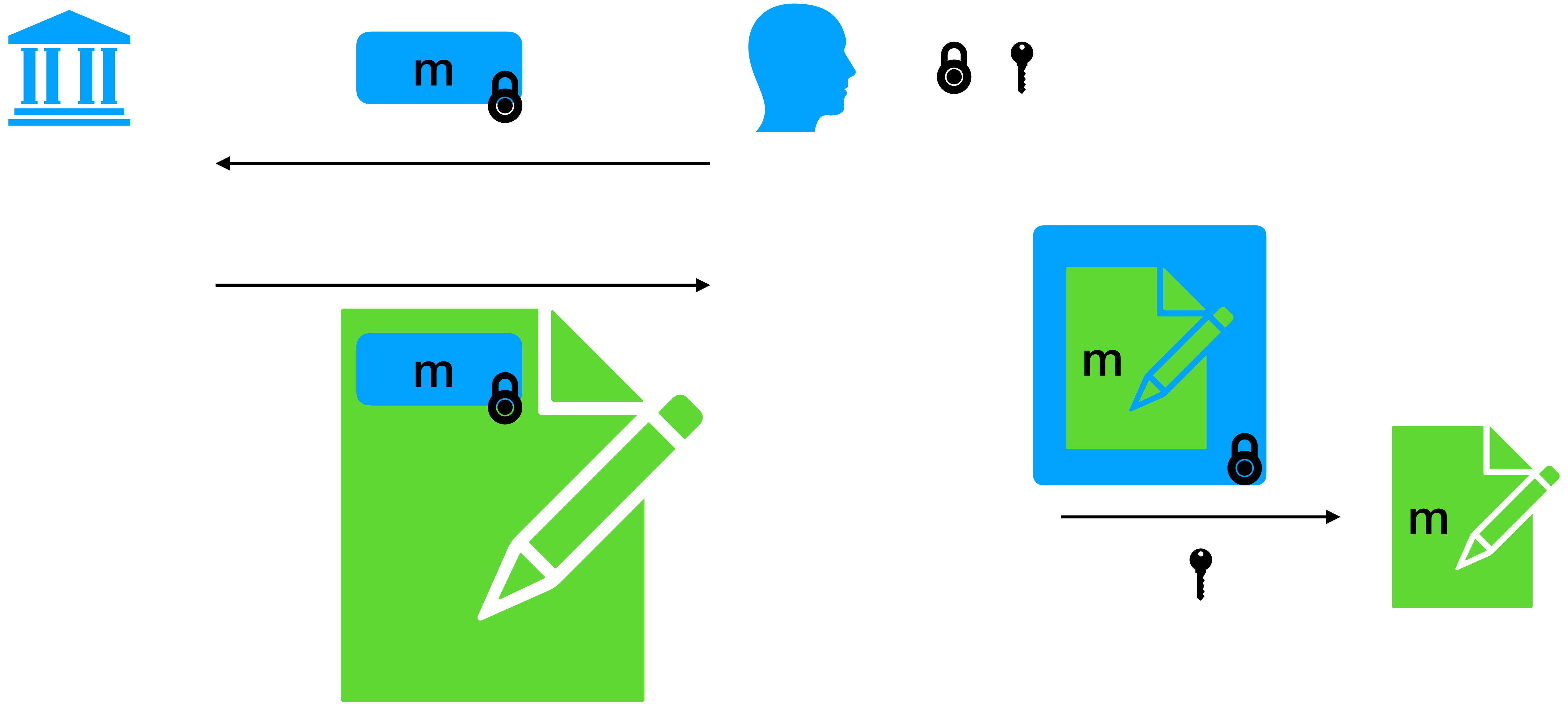
盲签名算法回顾

- 如何利用盲签名的性质设计数字现金?



盲签名算法

- 盲签名算法的步骤
- $\mathsf{Setup}(1^{\lambda}) \rightarrow (\mathsf{ek}, \mathsf{svk}, \mathsf{ssk})$
- $\mathsf{Blind}(\mathsf{ek}, \mathsf{m}) \rightarrow \bar{\mathsf{m}}$
- $\mathsf{Sig}(\mathsf{ssk}, \bar{\mathsf{m}}) = \bar{\sigma}$
- $\mathsf{UnBlind}(\mathsf{ek}, \bar{\sigma}) \rightarrow \sigma$



盲签名算法

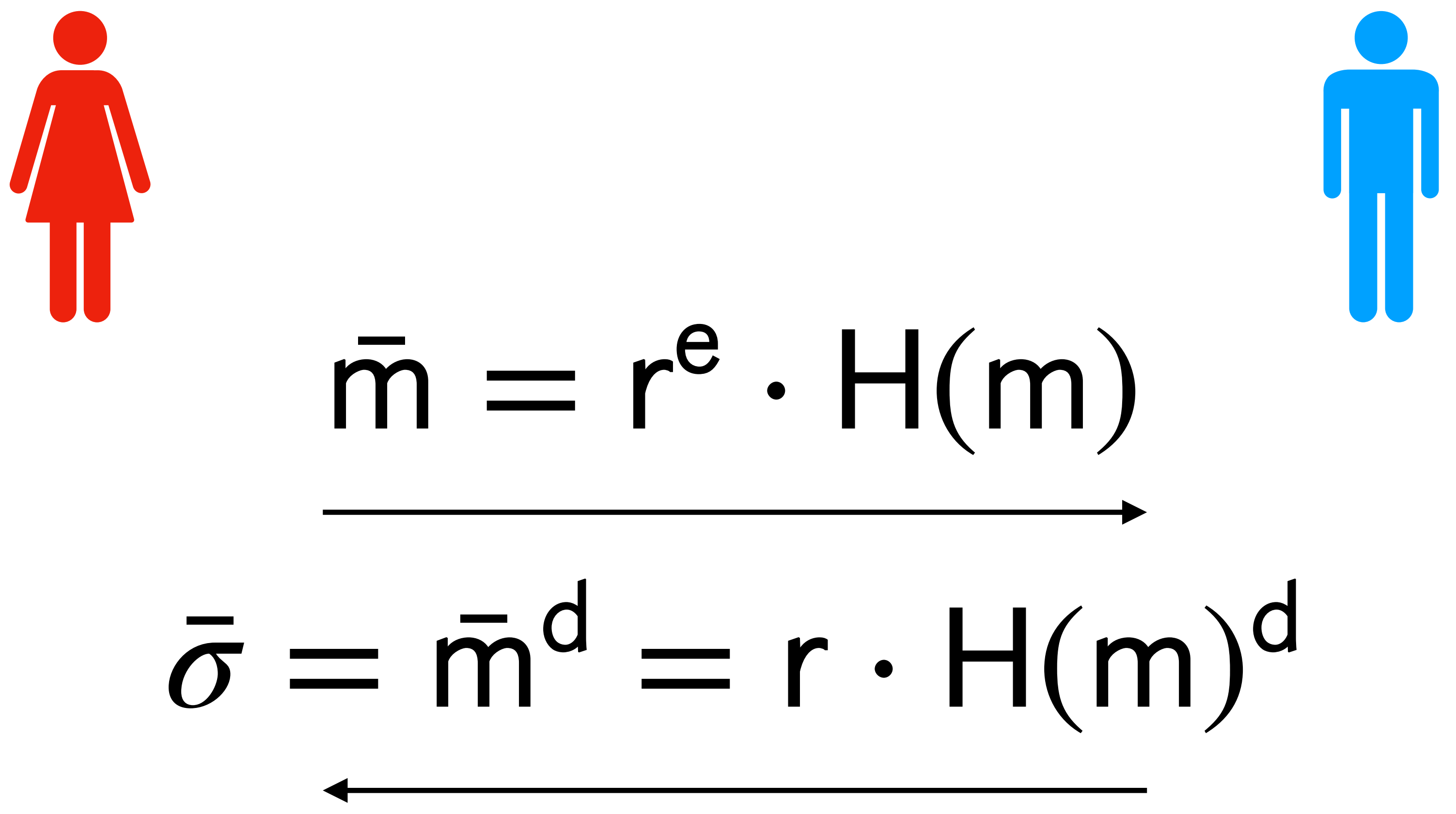
- 盲签名算法所要满足的性质：
  - 正确性（Correctness）：正确的签名可以被验证
  - 盲化（Blindness）：签名者不知道具体信息
  - 不可伪造性（Unforgeability）：无法伪造新的签名

盲签名算法

- 基础算法回顾：
  - RSA-FDH:
    - $\text{PK} = N, e, \text{SK} = d$
    - $\sigma = \text{H}(m)^d$
  - 一次一密One-Time Pad
    - $\text{SK} = K, \text{ct} = K \cdot m$

盲签名算法

- 最简单的盲签名算法（Chaum-RSA-FDH）
- $\mathrm{Setup}(1^{\lambda}) \rightarrow (\mathrm{ek}, \mathrm{svk}, \mathrm{ssk})$
  - $\mathrm{ek} = \mathrm{K}, \mathrm{svk} = \mathrm{N}, \mathrm{e}, \quad \mathrm{ssk} = \mathrm{d}$
- $\mathrm{Blind}(\mathrm{ek}, \mathrm{m}) \rightarrow \bar{\mathrm{m}}$
  - $\bar{\mathrm{m}} = \mathrm{r}^{\mathrm{e}} \cdot \mathrm{H}(\mathrm{m})$
- $\mathrm{Sig}(\mathrm{ssk}, \bar{\mathrm{m}}) = \bar{\sigma}$
  - $\bar{\sigma} = \bar{\mathrm{m}}^{\mathrm{d}} = \mathrm{r} \cdot \mathrm{H}(\mathrm{m})^{\mathrm{d}}$
- $\mathrm{UnBlind}(\mathrm{ek}, \bar{\sigma}) \rightarrow \sigma : \sigma = \bar{\sigma} \cdot \mathrm{r}^{-1}$



$$\xrightarrow{\bar{\mathrm{m}} = \mathrm{r}^{\mathrm{e}} \cdot \mathrm{H}(\mathrm{m})}$$

$$\xleftarrow{\bar{\sigma} = \bar{\mathrm{m}}^{\mathrm{d}} = \mathrm{r} \cdot \mathrm{H}(\mathrm{m})^{\mathrm{d}}}$$

盲签名算法

- 盲化属性（Blindness）？
  - $\bar{m} = r^e \cdot H(m)$
  - 签名者Bob不知道(m,r)
    - 给定任意m'，存在$r' = (\bar{m} \cdot H(m')^{-1})^d$，使得$\bar{m} = (r')^e \cdot H(m')$
    - 所以Bob没办法知道m的任何信息

盲签名算法

- 不可伪造性（Unforgeability）？
  - 在攻击者获得k个签名的情况下，仍然无法生成新的签名。
  - One-More RSA假设：给定一个寓言机$\mathcal{O}(x) = x^d$，攻击者在询问多项式次数以后，对于随机生成的$m \in \mathbb{Z}_N$，且$m \neq p \wedge m \neq q$，则找到m^d是困难的。
  - 想法：
    - 利用寓言机生成盲化后信息的签名
    - 如果攻击者生成了新的签名，则攻击了One-More RSA假设

密码学温故知新

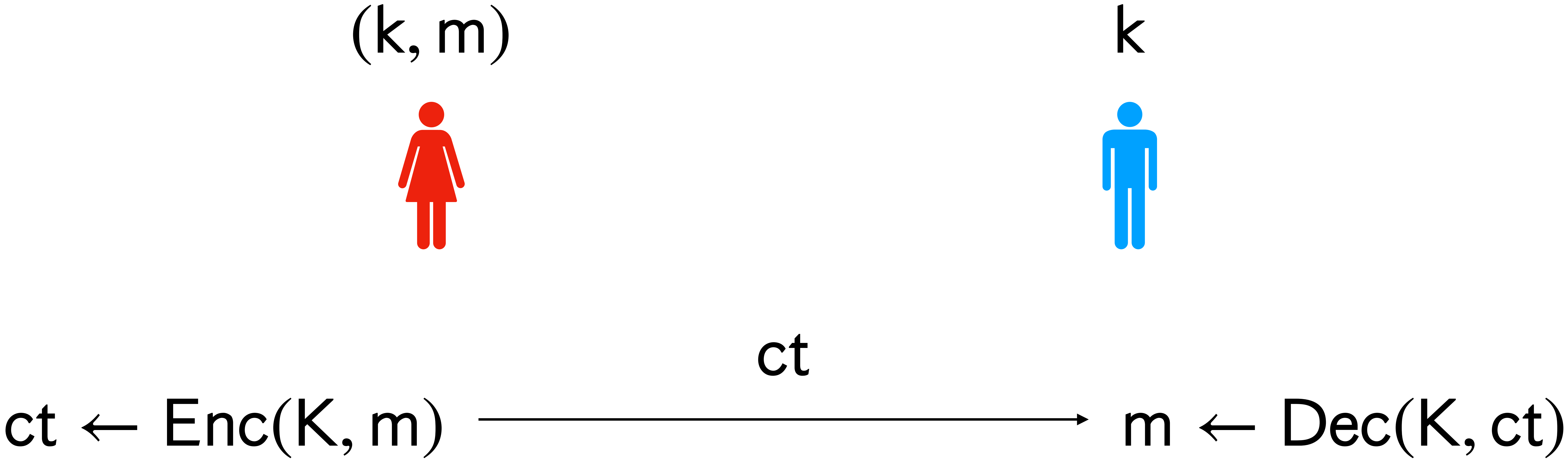
- 哈希函数
- 零知识证明
- 签名
- 盲签名
- 加密算法

加密算法

- 加密算法历史很悠久。。
- 对称加密算法
    - One-Time Pad
        - $ct = K \oplus m$，证明?
    - DES（1977，美国标准），AES（2001，美国标准），SM4（2012， 中国标准）

加密算法

- 对称加密算法



(k, m)

k

ct

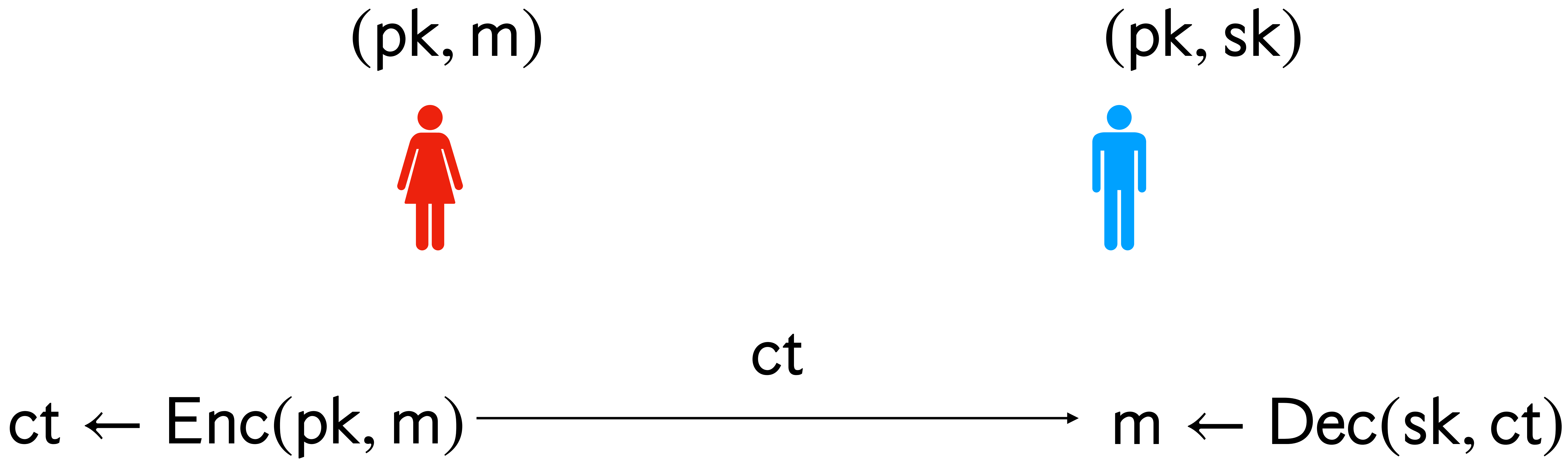
$ct \leftarrow \mathrm{Enc}(K, m) \longrightarrow m \leftarrow \mathrm{Dec}(K, ct)$

加密算法：对称到非对称

- 类似于从哈希函数到签名算法
- 我们对于对称加密算法也进行一些哲学思考：
  - 私钥（秘密）：定义了个人的身份
  - 加密过程：很多人向一个人传递信息的过程
  - 加密者：加密者并不需要证明自己的身份（不需要私钥）
  - 解密者：解密者是指定的（需要私钥）
  - 非对称加密

加密算法：非对称加密算法

- 非对称加密算法



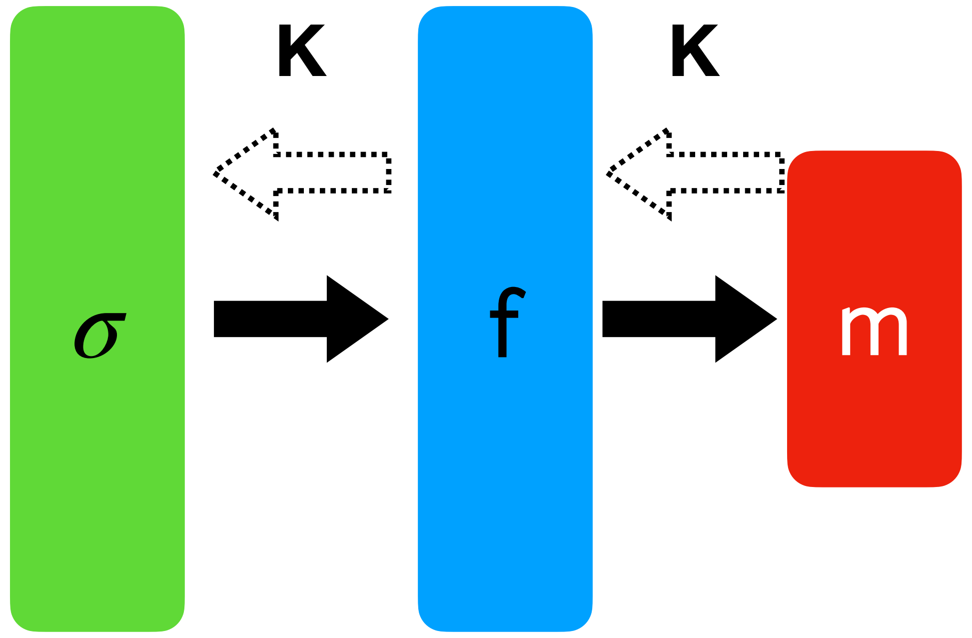
(pk, m)

(pk, sk)

$$\mathrm{ct} \leftarrow \mathrm{Enc}(\mathrm{pk}, \mathrm{m}) \xrightarrow{\mathrm{ct}} \mathrm{m} \leftarrow \mathrm{Dec}(\mathrm{sk}, \mathrm{ct})$$

非对称加密算法

- 需要哪些性质呢?
  - 加密的时候：明文计算密文→简单
  - 攻击者：密文计算明文（没有私钥）→困难
  - 解密的时候：密文计算明文（有私钥）→简单
  - 解密正确性：密文对应一个明文→ 一一对应
- 联想：陷门单项函数（置换）！

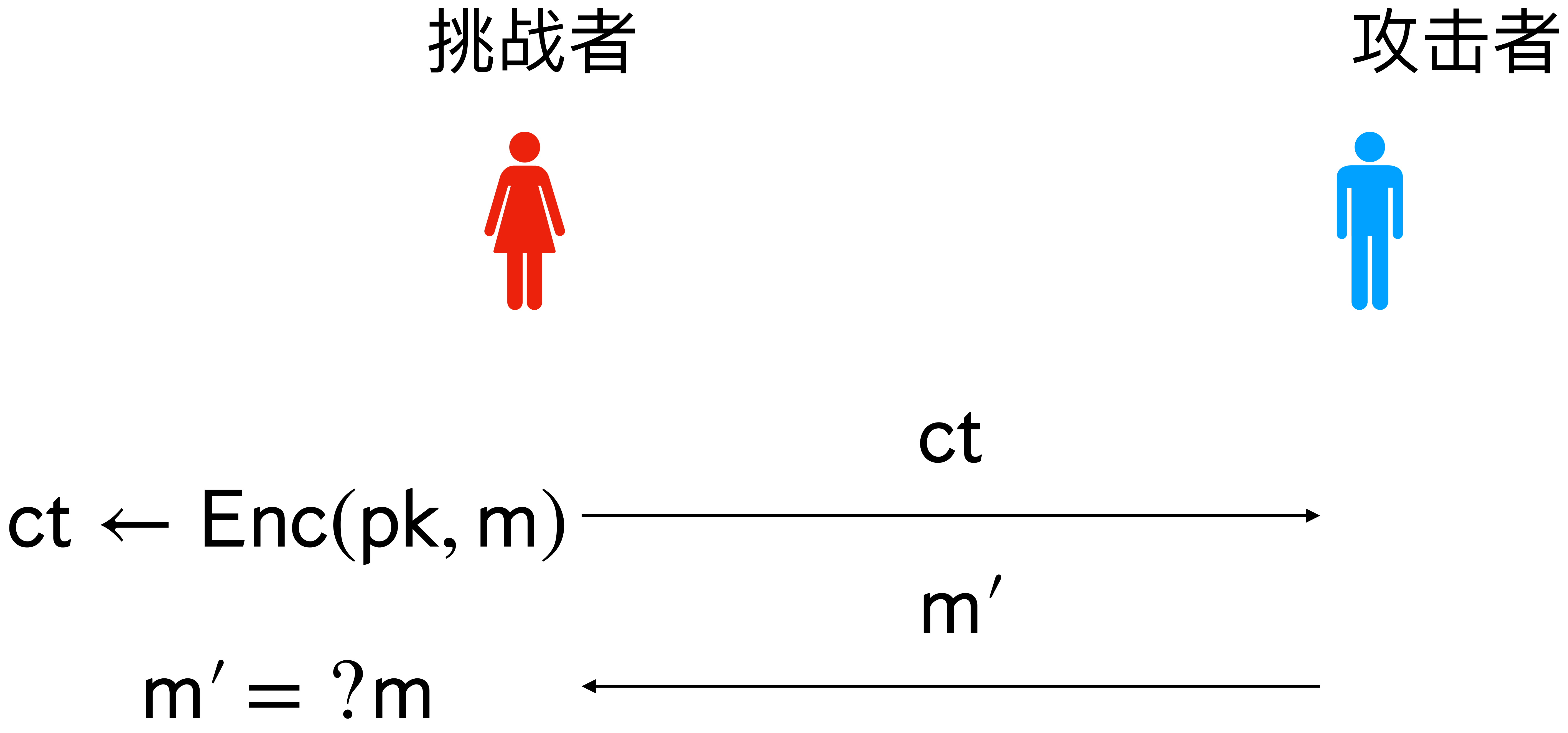


非对称加密算法

- 陷门单项置换！
  - 恰巧RSA陷门单向函数就是一个置换：$f_{\mathsf{RSA}}(x) = x^e$
  - f_{ISIS}虽然是一个陷门单向函数，但是并不是置换。
- RSA加密
  - pk = N, e, sk = d
  - ct = m^{e}

非对称加密 - 单向安全性

- 上述我们所介绍的安全性期望：密文$\nrightarrow$明文
  - 可以被严格定义为单向安全性



挑战者

攻击者

$\mathrm{ct} \leftarrow \mathrm{Enc}(\mathrm{pk}, \mathrm{m}) \xrightarrow{\mathrm{ct}}$

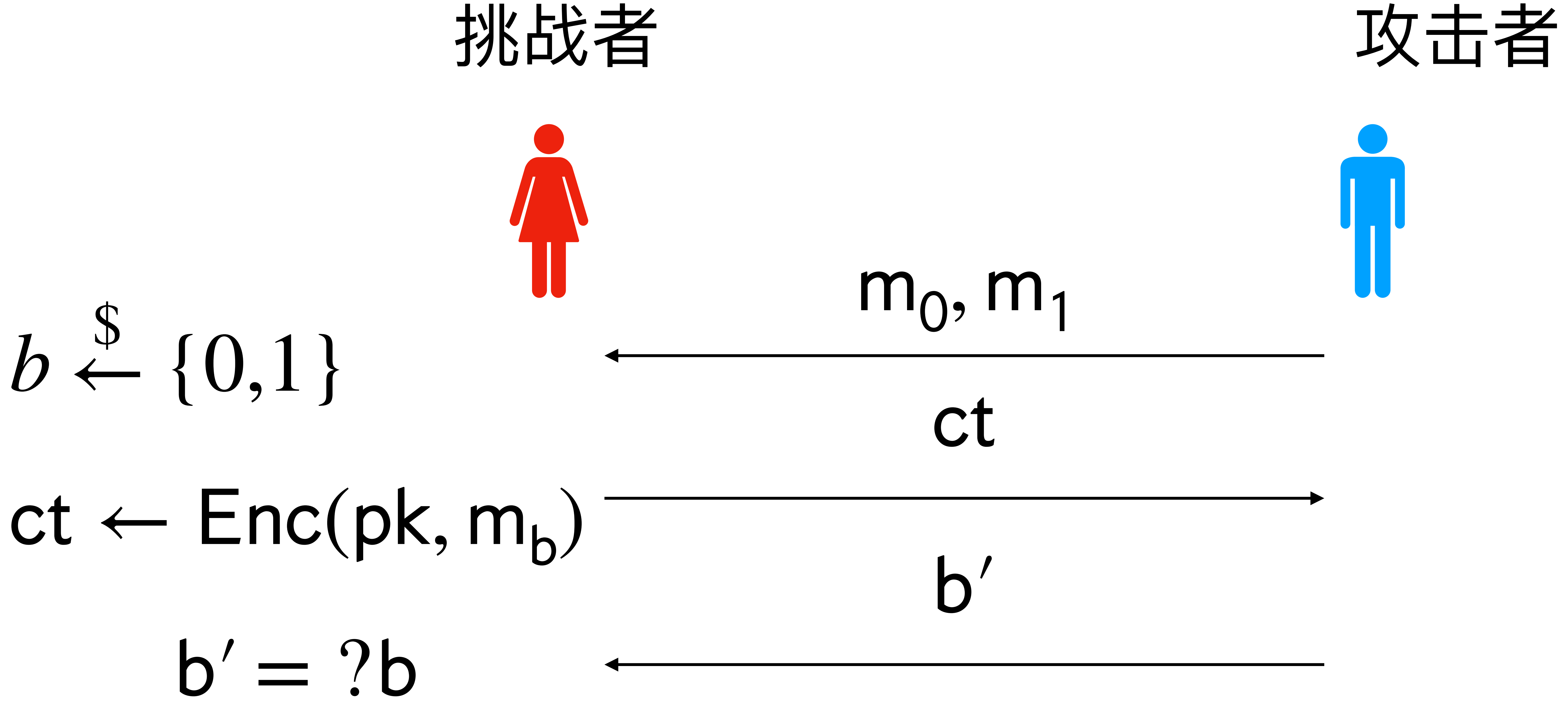
$\mathrm{m}' = ?\mathrm{m} \xleftarrow{\mathrm{m}'}$

非对称加密 - 单向安全性

- 单向安全性足够么?
- 给定一个单向安全的加密算法
  - $\mathrm{ct}' = \mathrm{ct}||\mathrm{m}_{[:4]}$，其中$\mathrm{m}_{[:4]}$表示消息的最后四位。
  - ct'仍然保证单向安全性
  - $m =$ ”打开山洞保险箱宝藏的口令是阿里巴巴”
  - $\mathrm{ct}' =$ qbbsdfkkjxcklhfsd; flkwe阿里巴巴
  - 好像不是很安全。。。

非对称加密 - 语义安全性（sementic security）

- $\forall \mathscr{A} \in \mathrm{PPT}.\ \mathscr{A}$不能通过$\mathrm{ct}_m$获得除了 m 以外的关于m的任何信息。（Goldwasser，Micali 1982）
- 后续证明该定义于IND-CPA定义等价



挑战者

攻击者

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

$\mathrm{m}_0, \mathrm{m}_1$

$\mathrm{ct} \leftarrow \mathrm{Enc}(\mathrm{pk}, \mathrm{m}_b)$

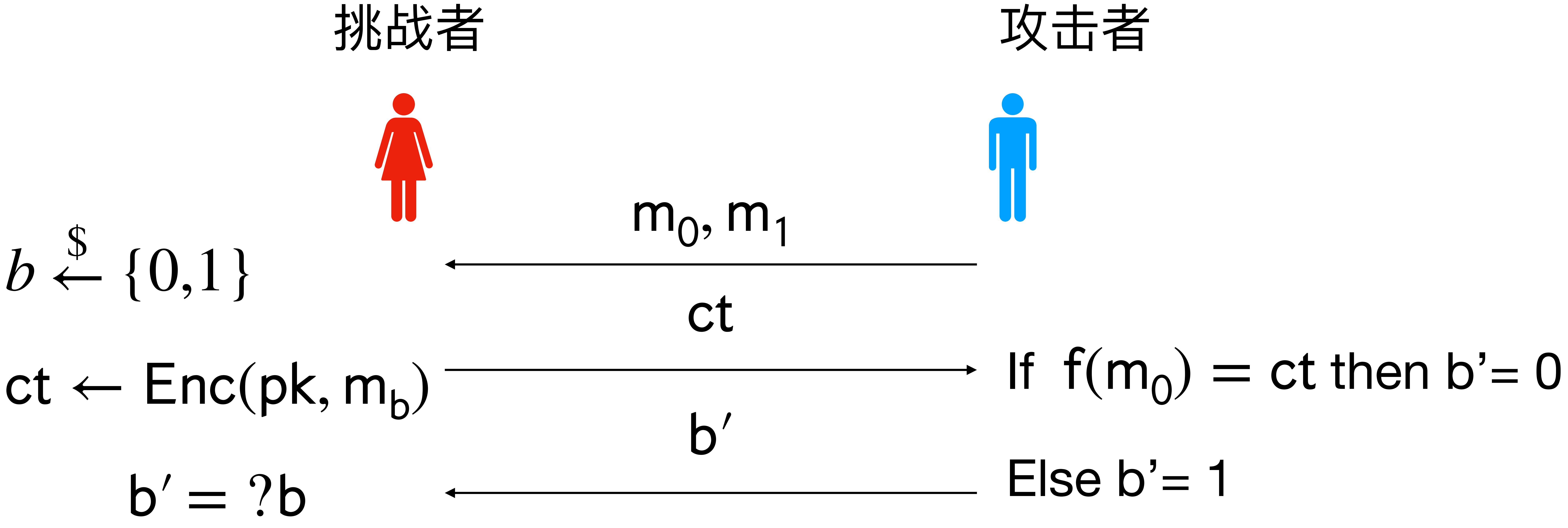
ct

b'

$\mathrm{b}' = ?\mathrm{b}$

非对称加密 - IND-CPA

- 思考：基于陷门单向函数的加密满足IND-CPA么?
- 不满足：陷门单向函数f是确定性函数。



挑战者

攻击者

m_0, m_1

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

ct

$ct \leftarrow Enc(pk, m_b)$

If $f(m_0) = ct$ then b’= 0

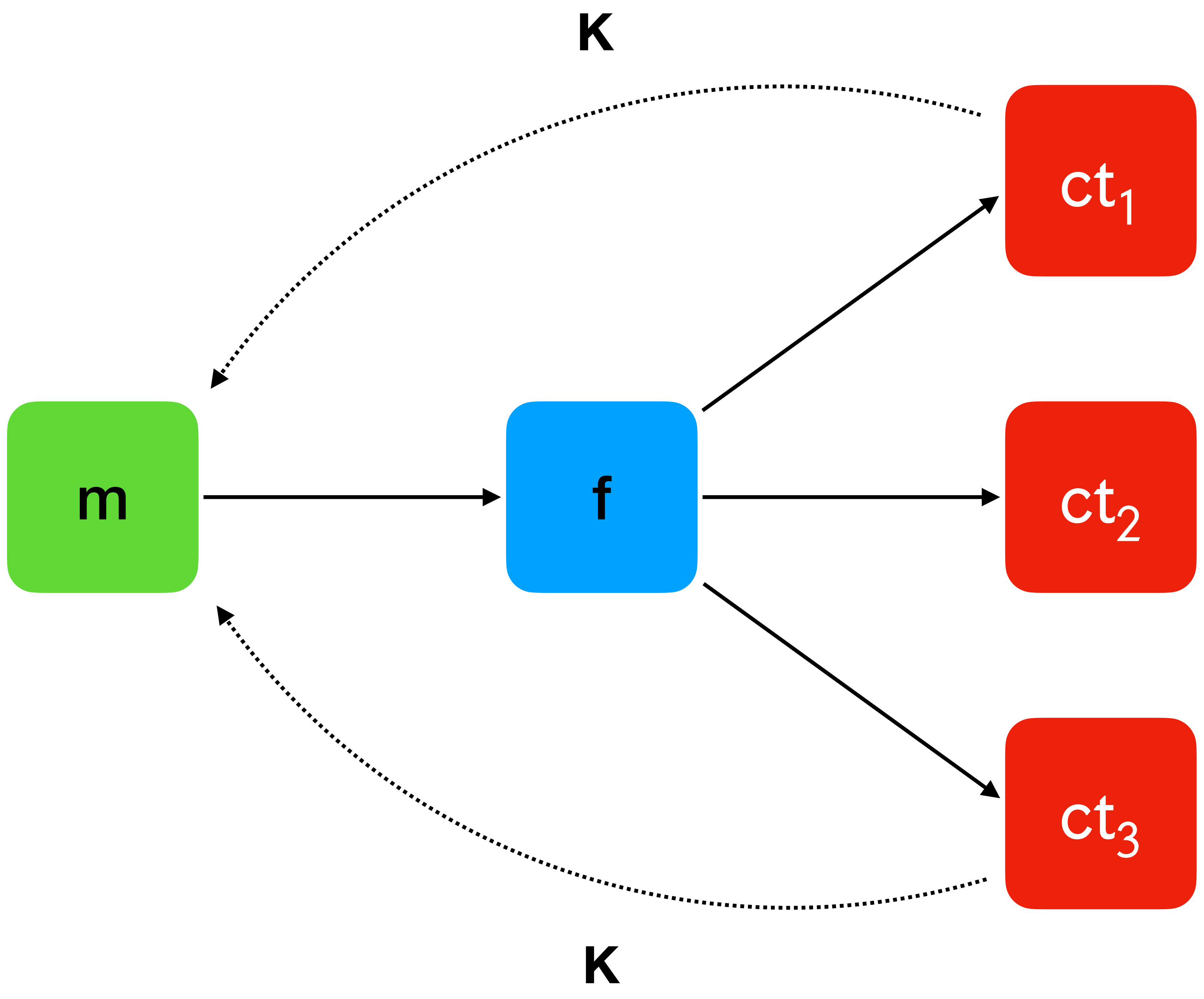
b'

Else b’= 1

$b' = ?b$

非对称加密 - IND-CPA

- 随机陷门单向函数



非对称加密 - IND-CPA

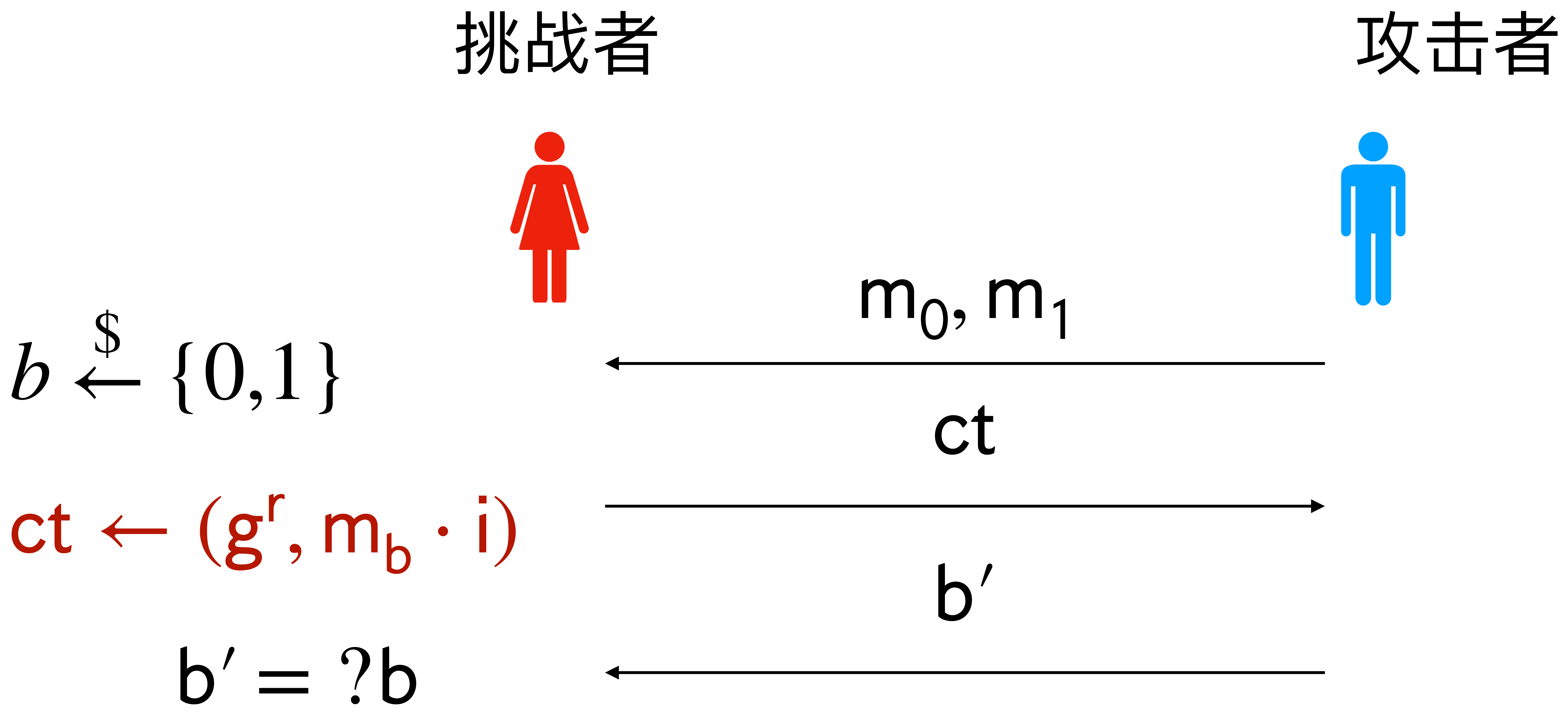
- 基于CDH假设$\mathrm{f_{g,h}(m; r) = (g^r, m \cdot h^r)}$
  - 单向性：CDH假设$g, h, g^r \nrightarrow h^r$
  - 陷门：
    - $\mathrm{k = \log_g(h)}$
    - $\mathrm{m = m \cdot h^r \cdot (g^r)^{-k}}$

非对称加密 - IND-CPA

- DDH假设
  - $(f, g, f^r, g^r) \equiv_c (f, g, f^r, i)$，其中$f, g, i \xleftarrow{\$} \mathbb{G}, r \xleftarrow{\$} \mathbb{Z}_p$
- El-Gammal加密算法
  - $\mathrm{pk} = \mathrm{y} = \mathrm{g}^{\mathrm{x}}$，$\mathrm{sk} = \mathrm{x}$
  - $\mathrm{ct} = (\mathrm{g}^{\mathrm{r}}, \mathrm{m} \cdot \mathrm{g}^{\mathrm{xr}})$

非对称加密 - IND-CPA

- $(f, g, f^r, g^r) \equiv_c (f, g, f^r, i)$
- DDH假设→El-Gammal加密IND-CPA安全



挑战者

攻击者

m_0, m_1

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

ct

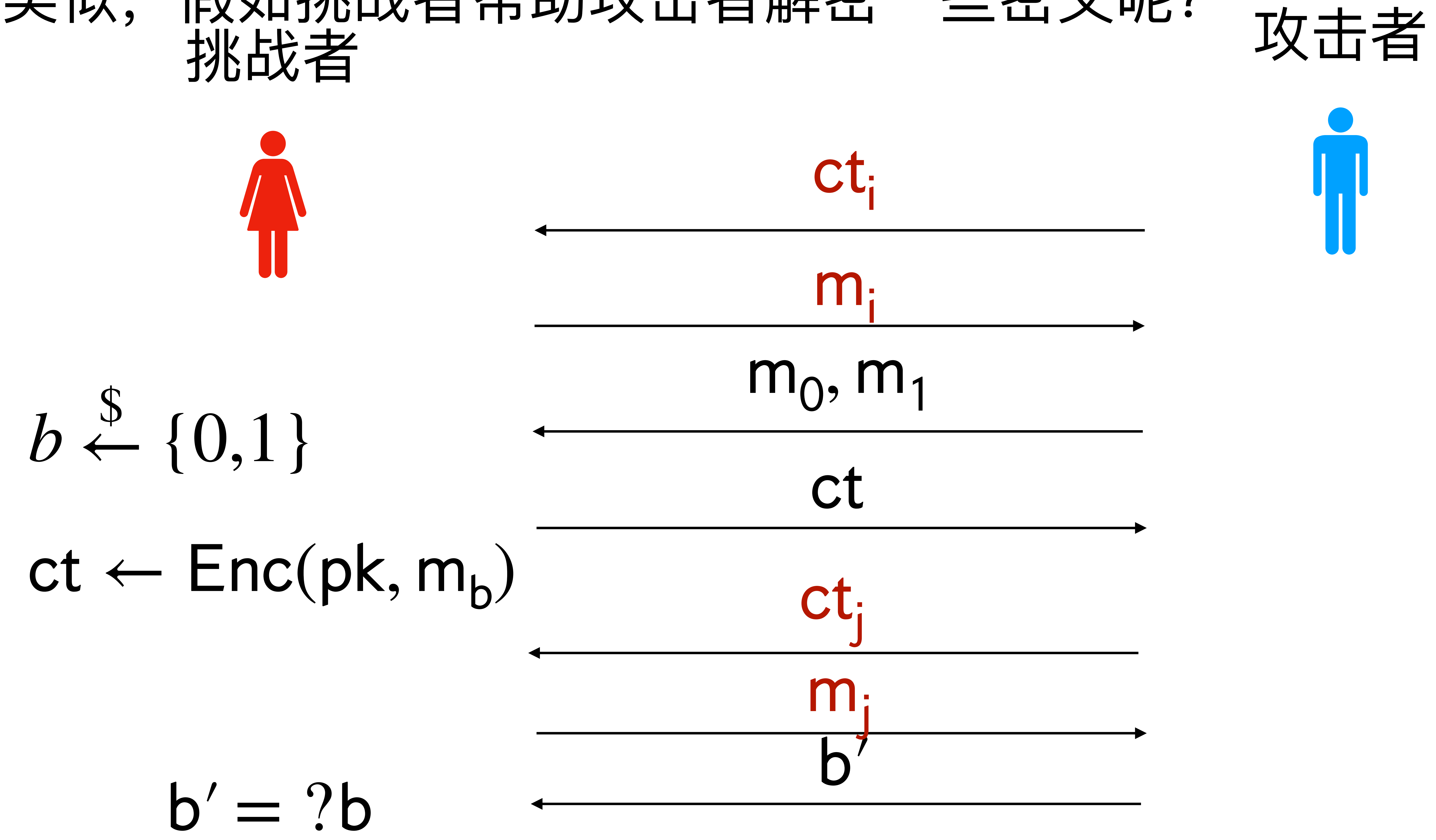
$\mathrm{ct} \leftarrow (g^r, m_b \cdot i)$

b'

$b' = ?b$

非对称加密 - IND-CCA

- IND-CPA足够安全么?
- 和签名算法类似，假如挑战者帮助攻击者解密一些密文呢?



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

$ct \leftarrow \mathrm{Enc}(pk, m_b)$

$b' = ?b$

非对称加密算法 - IND-CCA

- IND-CCA 安全可以被划分为IND-CCA1 与 IND-CCA2 安全
- 其中
  - IND-CCA1安全：在取得挑战密文前可以询问解密寓言机
  - IND-CCA2安全：在取得挑战密文之前和之后都可以询问解密寓言机
- El-Gammal 算法 IND-CCA2 安全么?
  - 不
  - El-Gammal : $\mathrm{ct} = (g^r, m \cdot g^{ar})$满足$\mathrm{Enc}(pk, 2m) = 2\mathrm{Enc}(pk, m)$

非对称加密算法 - IND-CCA1

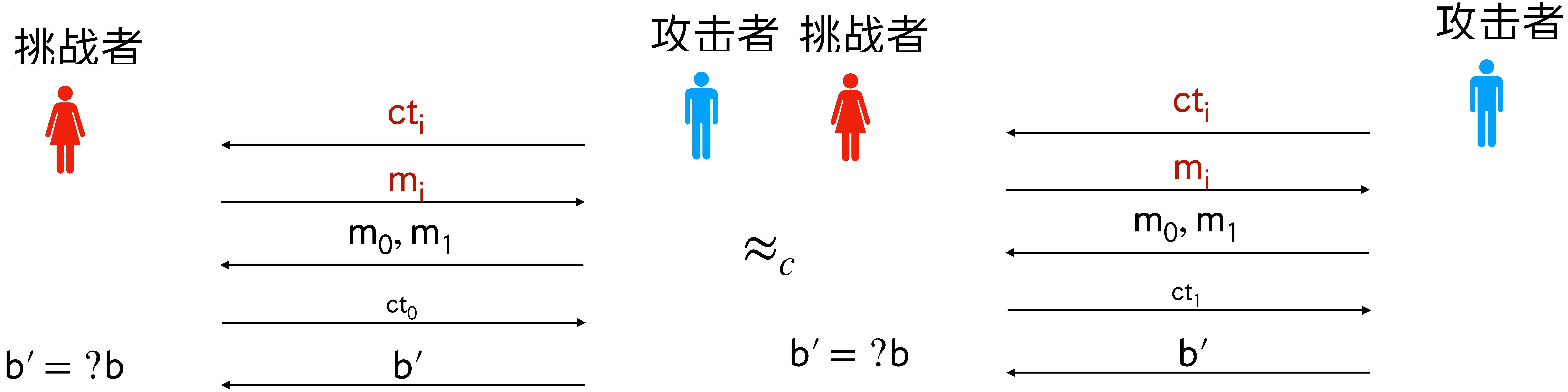
- Naor-Yung 构造 （STOC’ 90）： 给定IND-CPA 安全的加密$\mathscr{E} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ 和动态安全的零知识证明$\mathscr{P} = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$。下列加密算法是IND-CCA1安全的。

- $\mathsf{Setup}(1^\lambda)$:
  - $(\mathsf{pk}_1, \mathsf{sk}_1) \overset{\$}{\leftarrow} \mathsf{Setup}(1^\lambda)$;$(\mathsf{pk}_2, \mathsf{sk}_2) \overset{\$}{\leftarrow} \mathsf{Setup}(1^\lambda)$;$\mathsf{crs} \overset{\$}{\leftarrow} \mathsf{Gen}(1^\lambda)$
  - $\mathsf{pk} = (\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2)$, $\mathsf{sk} = (\mathsf{sk}_1)$

- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$:
  - $\mathsf{r}_1, \mathsf{r}_2 \overset{\$}{\leftarrow} \{0,1\}*$; $\mathsf{c}_1 = \mathsf{Enc}(\mathsf{pk}_1, \mathsf{m}; \mathsf{r}_1)$; $\mathsf{c}_2 = \mathsf{Enc}(\mathsf{pk}_2, \mathsf{m}; \mathsf{r}_2)$
  - $\pi \overset{\$}{\leftarrow} \mathsf{Prove}(\mathsf{crs}, (\mathsf{c}_1, \mathsf{c}_2), (\mathsf{m}, \mathsf{r}_1, \mathsf{r}_2))$; $\mathsf{ct} = (\mathsf{c}_1, \mathsf{c}_2, \pi)$

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Check $\mathsf{Ver}(\mathsf{crs}, (\mathsf{c}_1, \mathsf{c}_2), \pi)$; Return $\mathsf{Dec}(\mathsf{sk}_1, \mathsf{c}_1)$

为什么Naor-Yung构造是IND-CCA1安全?



挑战者

ct_i

m_i

$\mathsf{m}_0, \mathsf{m}_1$

ct_0

b'

$\mathsf{b}' = ?\mathsf{b}$

攻击者 挑战者

$\approx_c$

$\mathsf{b}' = ?\mathsf{b}$

ct_i

m_i

$\mathsf{m}_0, \mathsf{m}_1$

ct_1

b'

攻击者

Naor-Yung证明

- 通过一系列变换从左边变到右边：Games
- Game_0 : 加密m_0
- Game_1 : 和Game_0相同，但证明是由零知识证明的simulator产生(SimSetup, Sim, Ver)
- Game_2 : c_2是m_1的加密
- Game_2' : 用sk_2解密
- Game_3 : c_1是m_1的加密
- Game_3' : 用sk_1解密
- Game_3 : 用(Gen, Prove, Ver)产生证明

Naor-Yung 证明

- $\mathsf{Game}_0 \rightarrow \mathsf{Game}_1$: 零知识证明
- $\mathsf{Game}_1 \rightarrow \mathsf{Game}_2$: $\mathscr{E}_2$的IND-CPA安全性
- $\mathsf{Game}_2 \rightarrow \mathsf{Game}_2'$:
  - 攻击者产生$(\mathsf{c}_1, \mathsf{c}_2, \pi)$，通过验证但是$\mathsf{Dec}(\mathsf{sk}, \mathsf{c}_1) \neq \mathsf{Dec}(\mathsf{sk}, \mathsf{c}_2)$。这个概率在$\mathsf{Game}_2'$和$\mathsf{Game}_0$中概率是一样的
  - sk_2解密结果与sk_1相同
- $\mathsf{Game}_2' \rightarrow \mathsf{Game}_3$: $\mathscr{E}_1$的IND-CPA安全性
- $\mathsf{Game}_3 \rightarrow \mathsf{Game}_3'$: 同 $\mathsf{Game}_2 \rightarrow \mathsf{Game}_2'$
- $\mathsf{Game}_3' \rightarrow \mathsf{Game}_4$: 零知识证明，同$\mathsf{Game}_0 \rightarrow \mathsf{Game}_1$

Naor-Yung 加密不是IND-CCA2安全

- 给定一个$\mathscr{P} = (\text{Gen}, \text{Prove}, \text{Ver})$，构造$\mathscr{P}'$在每个证明后面加上一个比特0。验证的时候只验证除了最后一位的证明。

- 思考：
  - 使用$\mathscr{P}'$构造的Naor-Yung协议不满足IND-CCA2安全性

可随机加密

- IND-CCA2虽然安全，但是任何可随机加密都不满足IND-CCA2安全
- 在数字货币中，经常需要对加密进行随机化以达到匿名性目的
- RCCA安全性：
  - 只解密与挑战明文不同的密文。
  - 既保证了安全性也保留了可随机属性

补充：几点安全性相关的结论

- 任何确定性公钥加密算法都不是IND-CPA安全
  - 攻击者自己计算$\mathrm{Enc}(\mathrm{pk}, \mathrm{m}_0), \mathrm{Enc}(\mathrm{pk}, \mathrm{m}_1)$
  - 与挑战密文ct_b进行比较
  - 例子：RSA加密算法
- 任何密文可随机加密算法都不是IND-CCA2安全
  - 通过$\mathrm{ct}_\mathrm{b} = \mathrm{Enc}(\mathrm{m}_\mathrm{b}; \mathrm{r})$计算出在不知道私钥的情况下计算$\mathrm{ct}_\mathrm{b}' = \mathrm{Enc}(\mathrm{m}_\mathrm{b}; \mathrm{r}')$
  - 发送给挑战者
  - 例子：ElGamml 加密算法

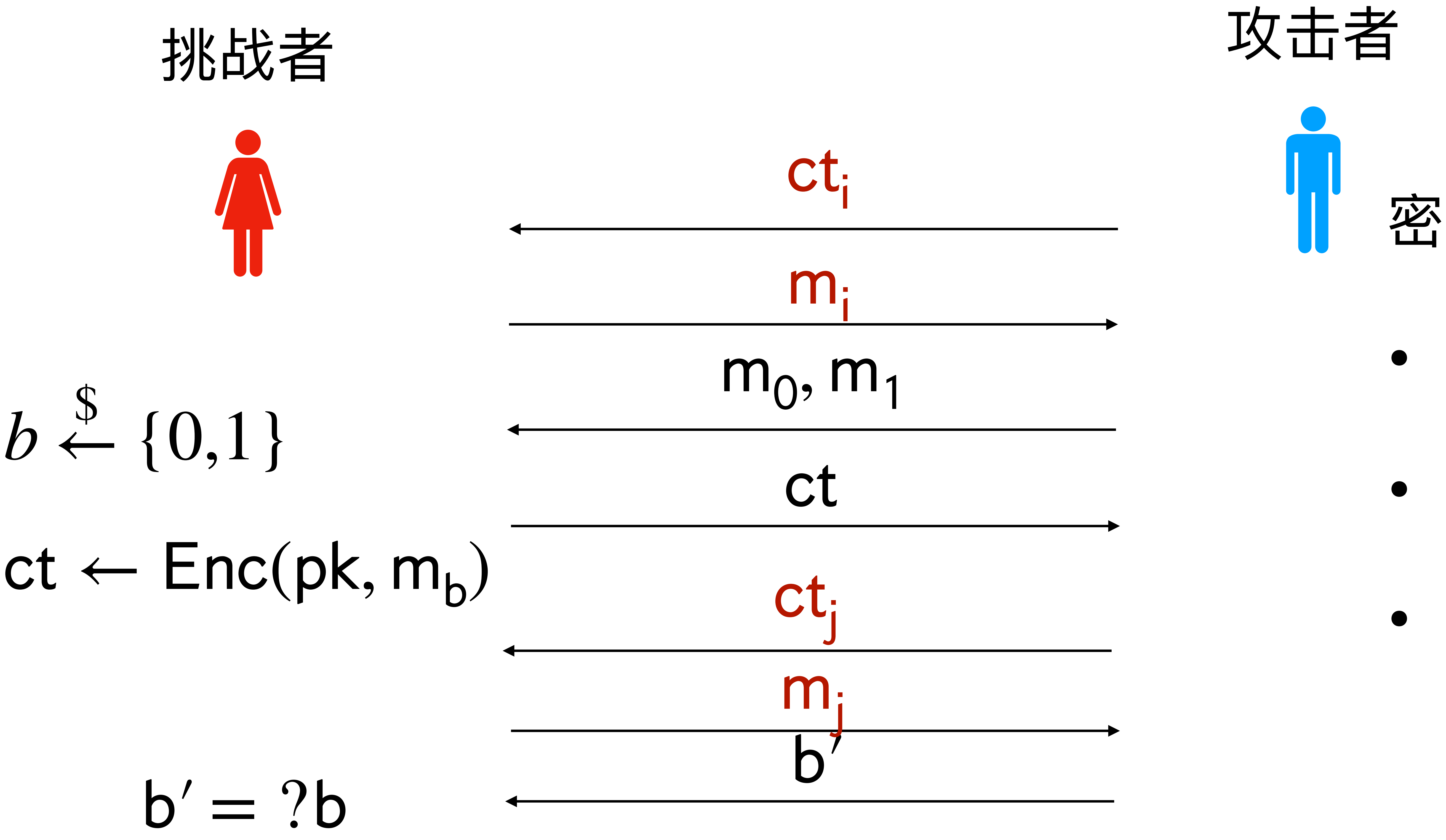
IND-CCA2安全性构造

- Dolev, Dwork, Naor（STOC’91）
- 思路：
  - 密文可随机化的加密算法 → 非CCA安全
  - 签名算法可以保证密文无法更改
  - 加入签名算法！

DDN’91 - IND-CCA2安全加密算法

- 第一次尝试：
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$:
  - $(\mathsf{sk}, \mathsf{vk}) \overset{\$}{\leftarrow} \mathsf{SigGen}(1^\lambda)$
  - $\mathsf{r}_1, \mathsf{r}_2 \overset{\$}{\leftarrow} \{0,1\}^*$; $\mathsf{c}_1 = \mathsf{Enc}(\mathsf{pk}_1, \mathsf{m}; \mathsf{r}_1)$; $\mathsf{c}_2 = \mathsf{Enc}(\mathsf{pk}_2, \mathsf{m}; \mathsf{r}_2)$
  - $\pi \overset{\$}{\leftarrow} \mathsf{Prove}(\mathsf{crs}, (\mathsf{c}_1, \mathsf{c}_2), (\mathsf{m}, \mathsf{r}_1, \mathsf{r}_2))$;
  - $\sigma \leftarrow Sig(sk, (c_1, c_2, \pi))$
  - $\mathsf{ct} = (\mathsf{vk}, (\mathsf{c}_1, \mathsf{c}_2, \pi), \sigma)$

DDN’91 - IND-CCA2安全加密算法



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

$\mathrm{ct} \leftarrow \mathrm{Enc}(\mathrm{pk}, \mathrm{m_b})$

$\mathrm{b}' = ?\mathrm{b}$

密文：

- $\mathrm{ct} = (\mathrm{vk}, (\mathrm{c}_1, \mathrm{c}_2, \pi), \sigma)$
- σ保证了(c_1, c_2, π)无法更改
- 但是
  - 攻击者可以自己生成vk
  - 并生成新的签名

DDN’91 - IND-CCA2安全加密算法

- 上述尝试的问题在于：
  - 加密者需要用签名私钥去签名
  - 但是，公钥加密过程中没有秘密信息参与
  - 加密与签名公钥并没有绑定
- 解决方式：
  - 联想：Lamport一次性签名时，没有陷门也可以用单向函数签名
  - 假设 vk = k，且vk的第i个比特记作vk_i
  - 提前准备
  - $pk = \begin{pmatrix} pk_{1,0} & \cdots & pk_{k,0} \\ pk_{1,1} & \cdots & pk_{k,1} \end{pmatrix},$ 加密时使用k个不同公钥$pk_{i,vk_i},$ 进行加密。

DDN'91 - IND-CCA2安全加密算法

- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$:
  - $(\mathsf{sk}, \mathsf{vk}) \overset{\$}{\leftarrow} \mathsf{SigGen}(1^{\lambda})$
  - $\mathsf{r}_1, \ldots, \mathsf{r}_\mathsf{k} \overset{\$}{\leftarrow} \{0, 1\}^*; \mathsf{c}_\mathsf{i} = \mathsf{Enc}(\mathsf{pk}_\mathsf{i}, \mathsf{m}; \mathsf{r}_\mathsf{i})$ for all $i \in \{1,\ldots,k\}$
  - $\pi \overset{\$}{\leftarrow} \mathsf{Prove}(\mathsf{crs}, (\{\mathsf{c}_\mathsf{i}\}_{\mathsf{i}\in[\mathsf{k}]}), (\mathsf{m}, \{\mathsf{r}_\mathsf{i}\}_{\mathsf{i}\in[\mathsf{k}]}));$
  - $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, (\{\mathsf{c}_\mathsf{i}\}_{\mathsf{i}\in[\mathsf{k}]}, \pi))$
  - $\mathsf{ct} = (\mathsf{vk}, (\{\mathsf{c}_1\}_{\mathsf{i}\in[\mathsf{k}]}, \pi), \sigma)$
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$:
  - Check $\mathsf{Ver}(\mathsf{crs}, (\{\mathsf{c}_\mathsf{i}\}_{\mathsf{i}\in[\mathsf{k}]}), \pi)$; Check $\mathsf{Ver}(\mathsf{vk}, (\{\mathsf{c}_\mathsf{i}\}_{\mathsf{i}\in[\mathsf{k}]}, \pi), \sigma)$
  - Return $\mathsf{Dec}(\mathsf{sk}_{1,\mathsf{vk}_1}, \mathsf{c}_{1,\mathsf{vk}_1})$

DDN’91 - IND-CCA2安全加密算法

- 首先证明：
  - 令$\mathsf{ct}^* = (\mathsf{vk}^*, \mathsf{c}^*, \sigma^*)$
  - 首先观察得知：如果攻击者向挑战者询问了$\mathsf{ct} = (\mathsf{vk}^*, \mathsf{c}', \sigma')$且
    - $\mathsf{Ver}(\mathsf{vk}^*, \mathsf{c}, \sigma) = 1$
    - $(c, \sigma) \neq (c^*, \sigma^*)$
  - 则攻击者成功攻击了签名算法的抗伪造性（Strong Unforgeability）

DDN'91 - IND-CCA2安全加密算法

- Game_0 : 加密m_0
- Game_1 : 和Game_0相同，但证明是由零知识证明的simulator产生$(\mathsf{SimSetup}, \mathsf{Sim}, \mathsf{Ver})$
- Game_1' : 所有挑战密文如果包括vk*，则直接拒绝
- Game_2 : 令ℓ是vk和vk*，的第一个不相同比特。
- Game_2' : 用$\mathsf{sk}_{\ell,\mathsf{vk}_\ell}$进行解密
- Game_2'' : 将所有密文替换成为m_1的加密
- Game_3 : 用sk_1解密
- Game_3' : 不再拒绝含vk*的密文
- Game_3'' : 用$(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$产生证明