

# 数字货币和区块链 - 密码学 (3)

山东大学网络空间安全学院

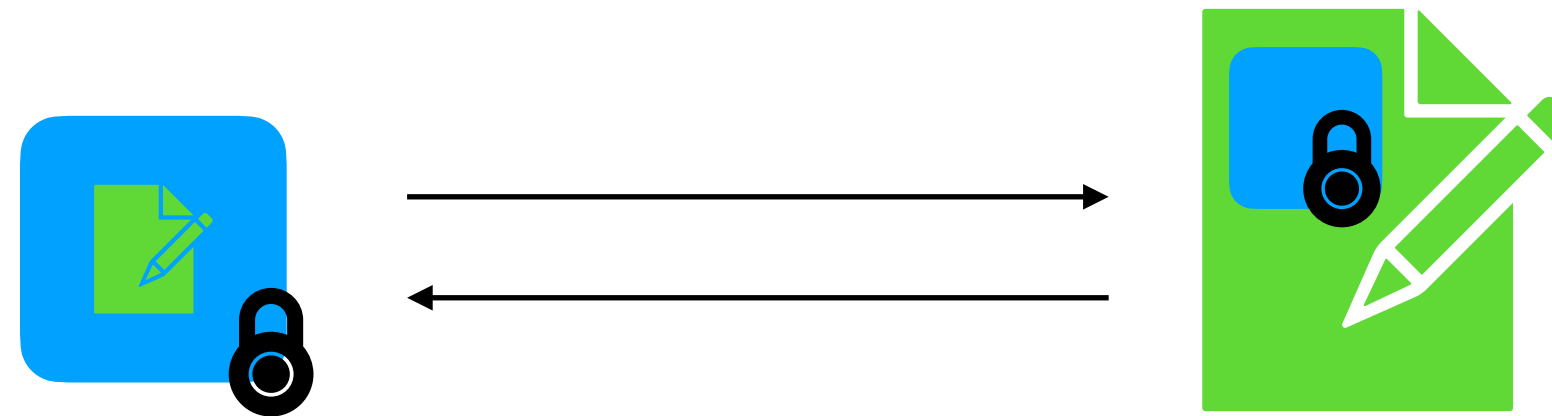
钱宸 2023年11月8日

# 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

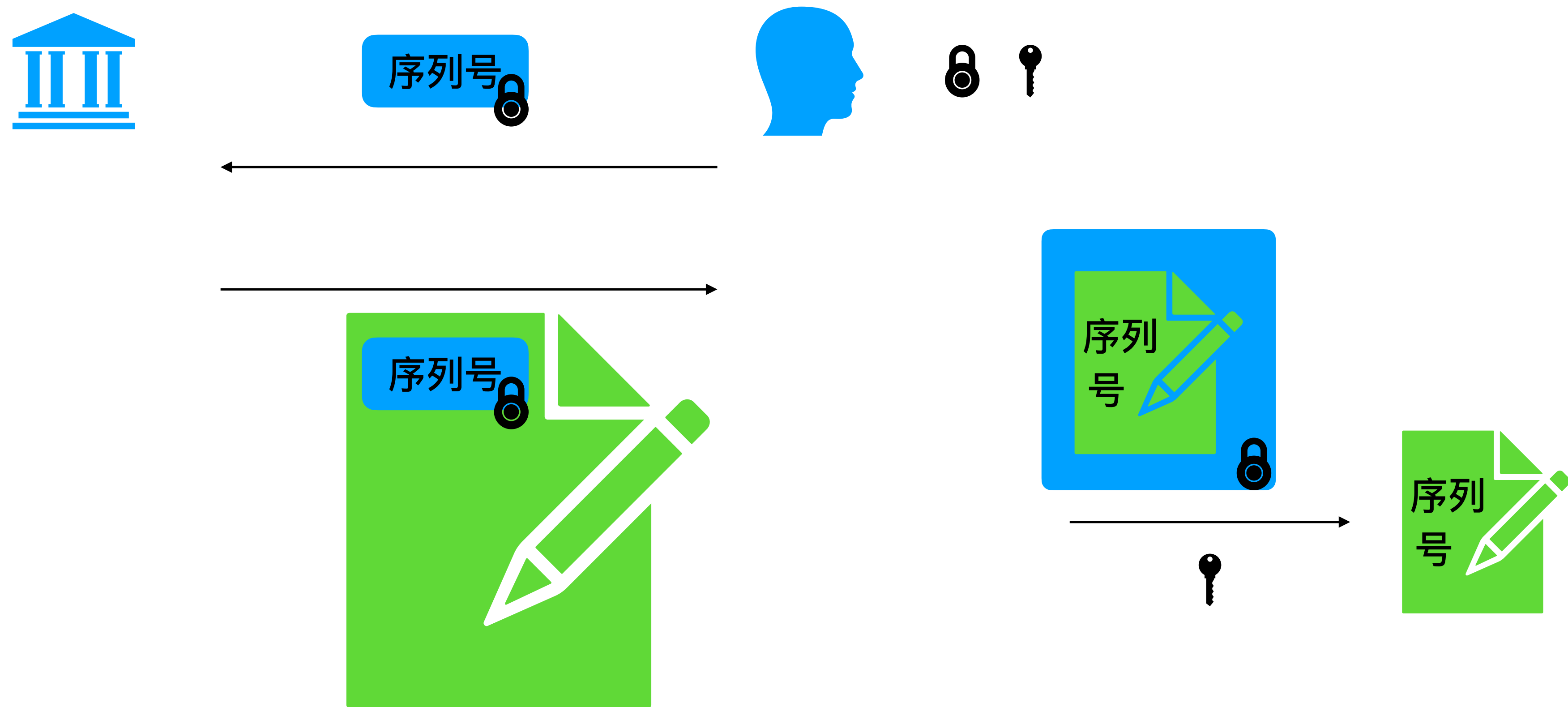
# 盲签名算法回顾

- 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：



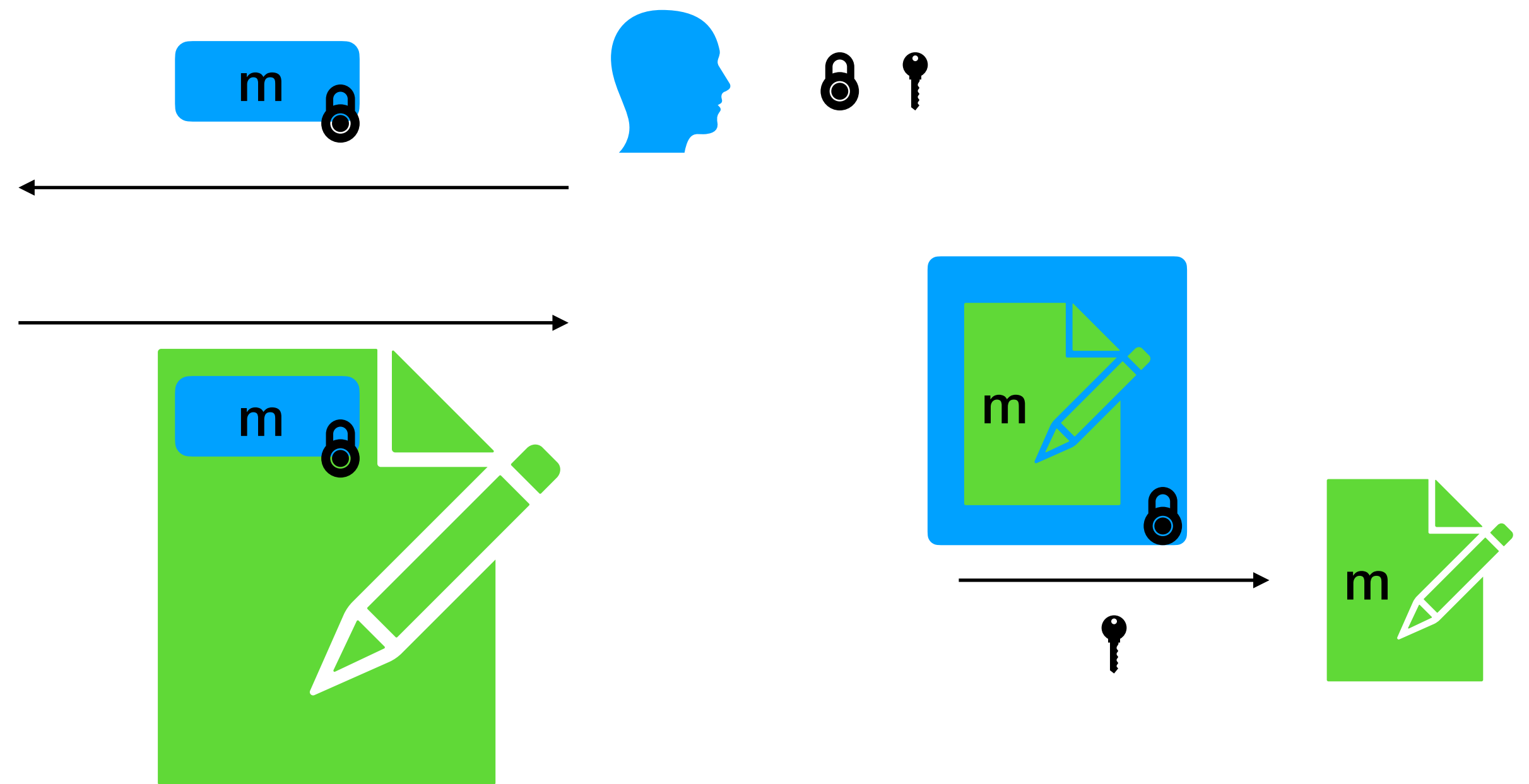
# 盲签名算法回顾

- 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
- $\text{Setup}(1^\lambda) \rightarrow (\text{ek}, \text{svk}, \text{ssk})$
- $\text{Blind}(\text{ek}, m) \rightarrow \bar{m}$
- $\text{Sig}(\text{ssk}, \bar{m}) = \bar{\sigma}$
- $\text{UnBlind}(\text{ek}, \bar{\sigma}) \rightarrow \sigma$



# 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:
    - $PK = N, e, SK = d$
    - $\sigma = H(m)^d$
  - 一次一密One-Time Pad
    - $SK = K, ct = K \cdot m$

# 盲签名算法

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



$$\begin{array}{c} \bar{m} = r^e \cdot H(m) \\ \xrightarrow{\hspace{1.5cm}} \\ \bar{\sigma} = \bar{m}^d = r \cdot H(m)^d \\ \xleftarrow{\hspace{1.5cm}} \end{array}$$



# 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：给定一个预言机  $\mathcal{O}(x) = x^d$ ，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：
  - 利用预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

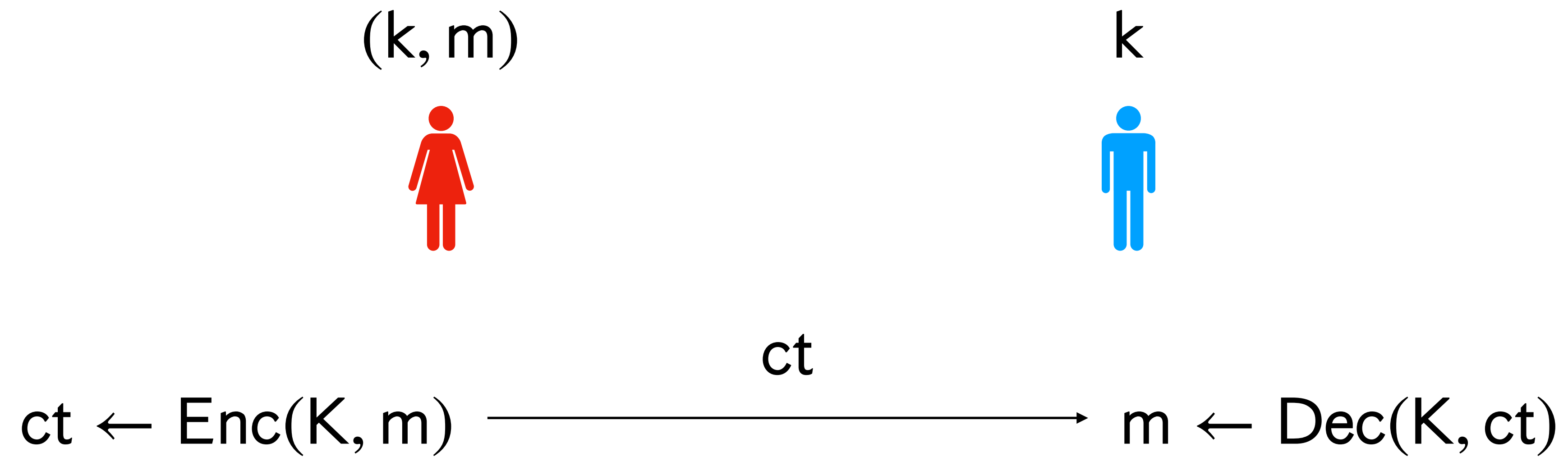
- 哈希函数
- 零知识证明
- 签名
- 盲签名
- 加密算法

# 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

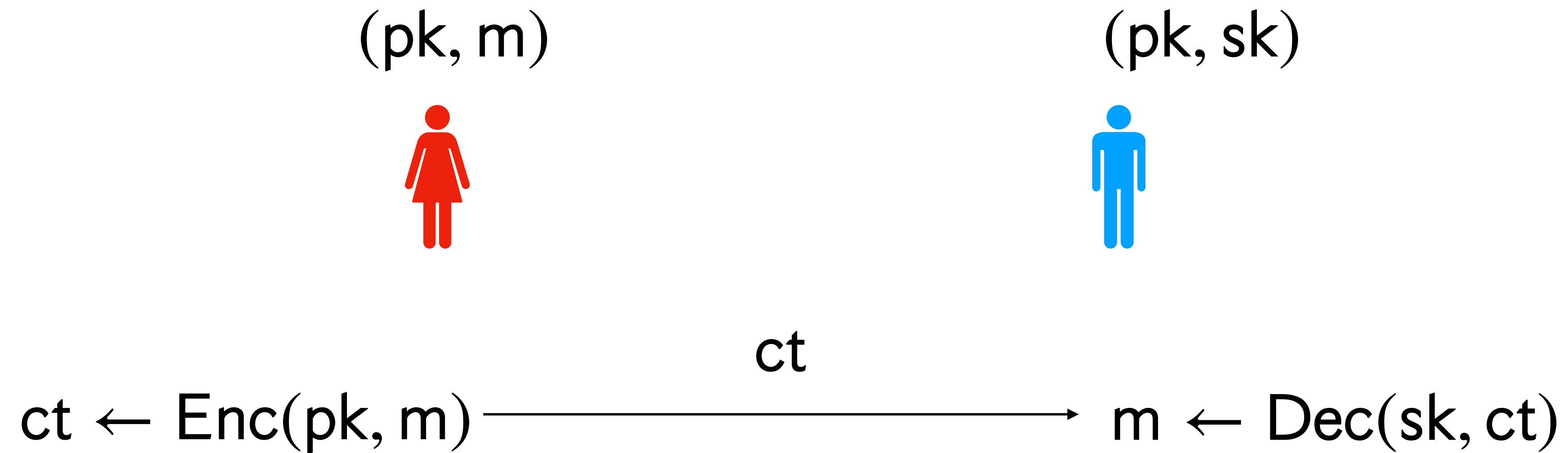


# 加密算法： 对称到非对称

- 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

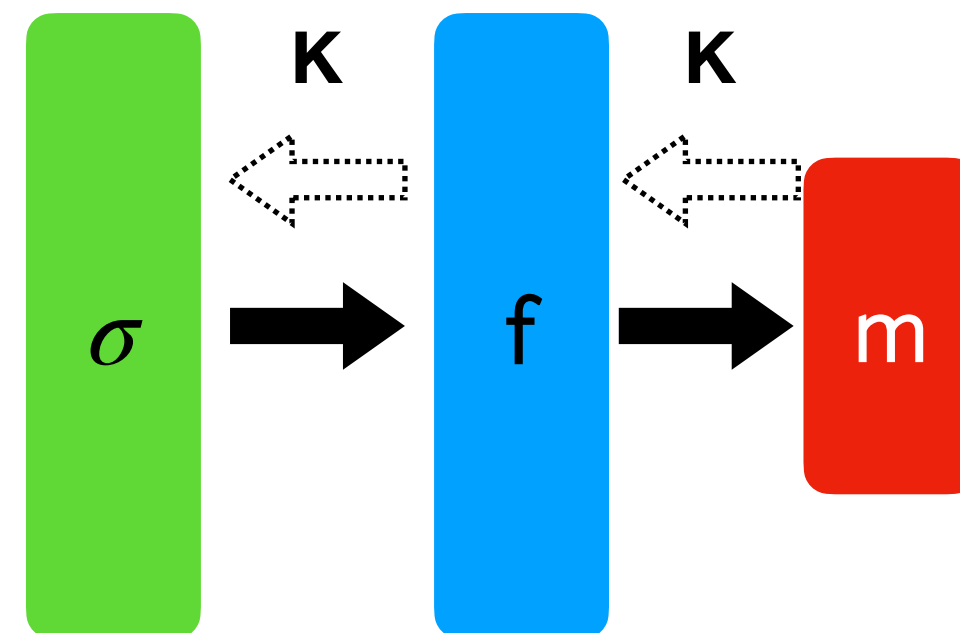
# 加密算法：非对称加密算法

- 非对称加密算法



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

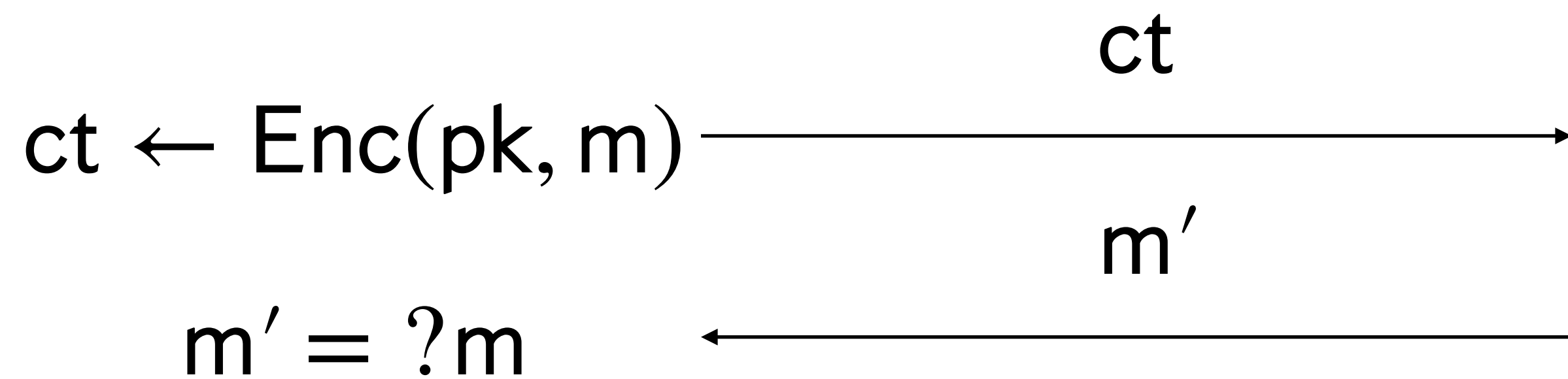
# 非对称加密 - 单向安全性

- 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



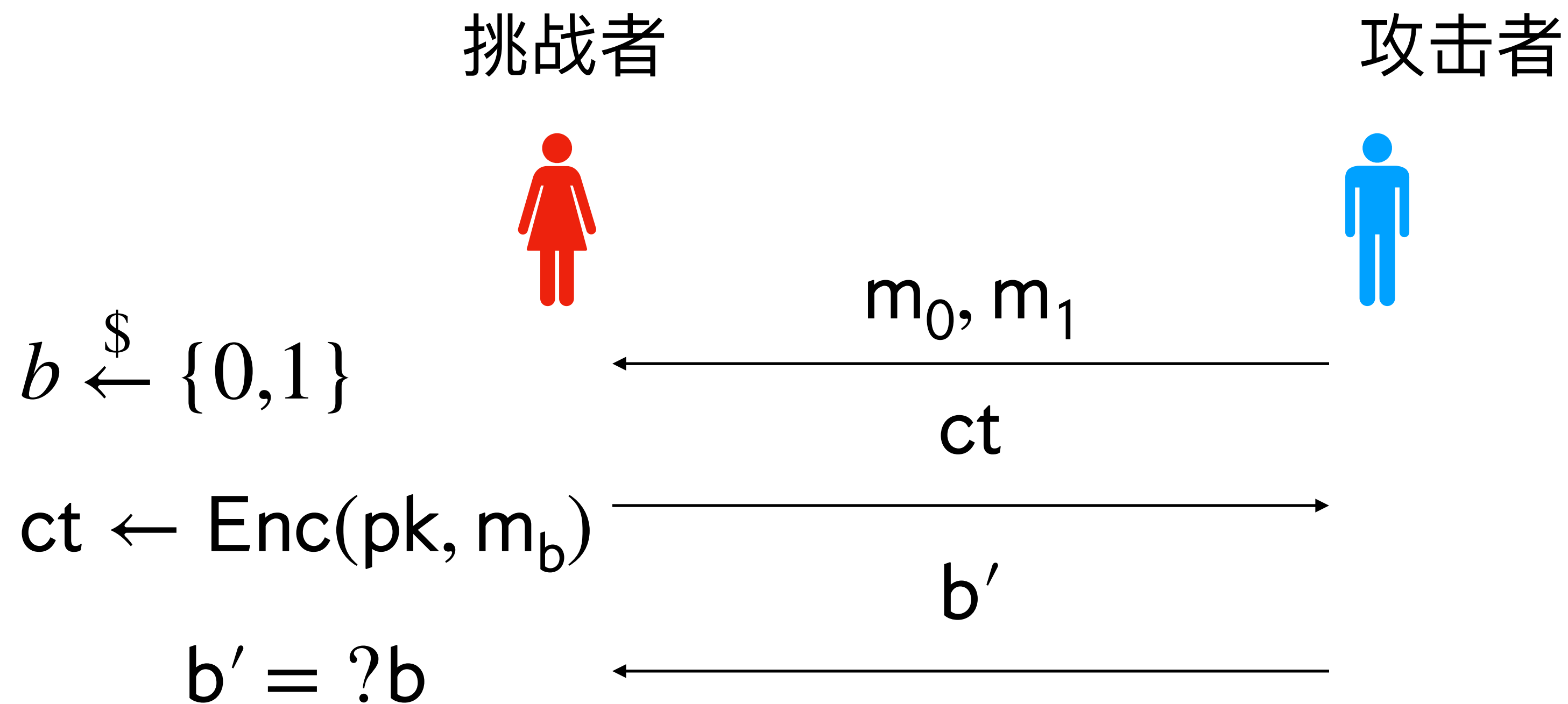
$$m' = ?m$$

# 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
  - $ct' = ct || m_{[:4]}$ , 其中 $m_{[:4]}$ 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"
  - $ct' =$  qbbsdfkkjxccklhfsd; flkwe阿里巴巴
  - 好像不是很安全。。。

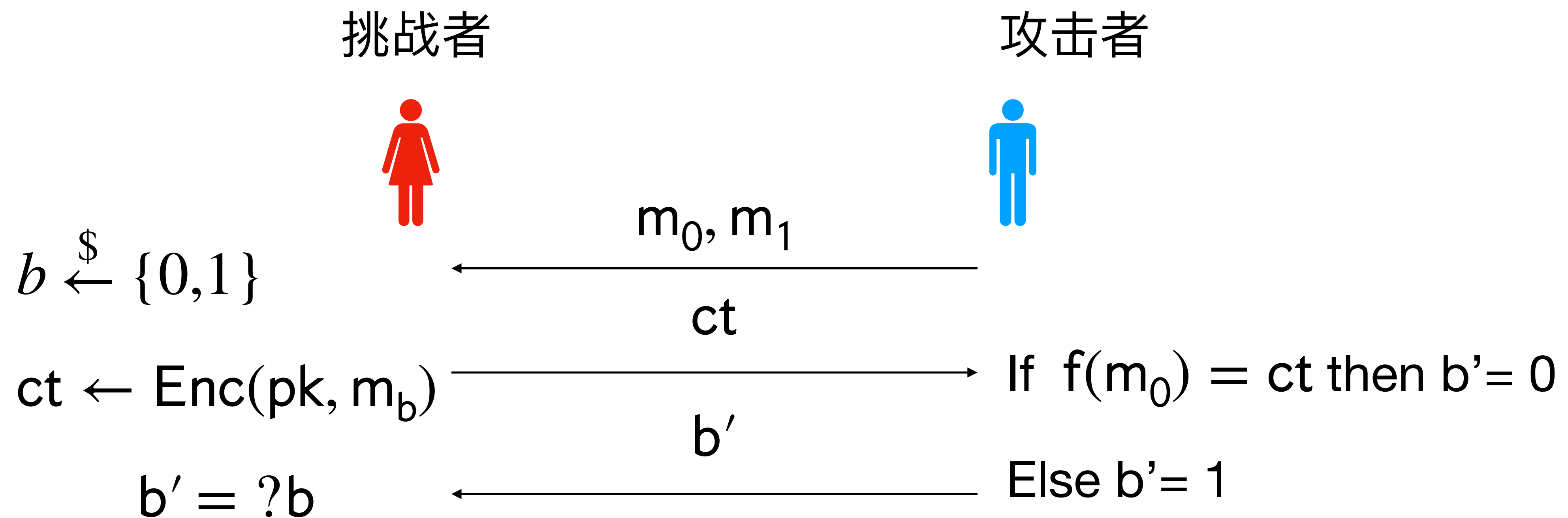
# 非对称加密 - 语义安全性 (semantic security)

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



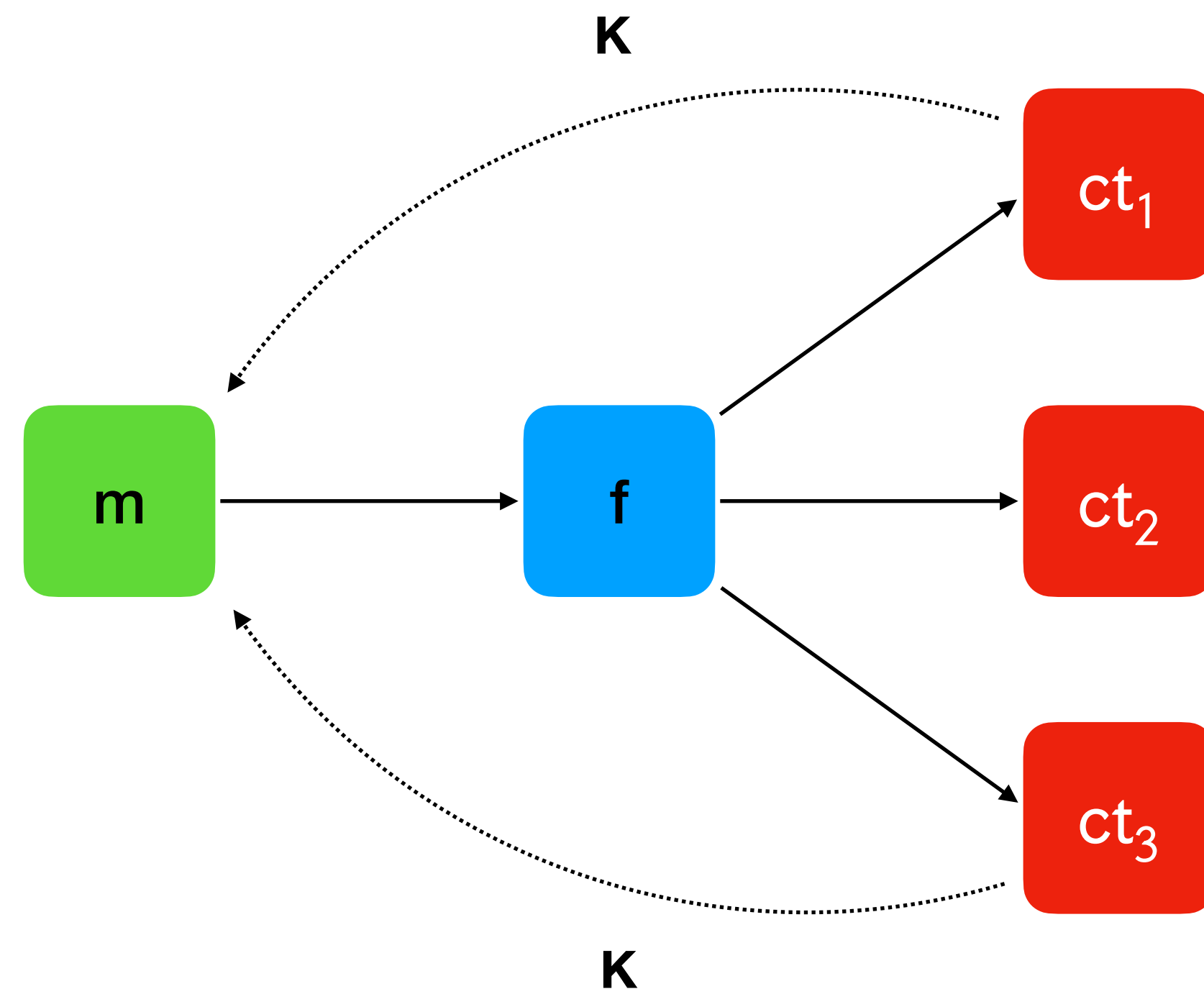
# 非对称加密 - IND-CPA

- 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。



# 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  $f_{g,h}(m; r) = (g^r, m \cdot h^r)$ 
  - 单向性: CDH假设  $g, h, g^r \not\Rightarrow h^r$
  - 陷门:
    - $k = \log_g(h)$
    - $m = m \cdot h^r \cdot (g^r)^{-k}$

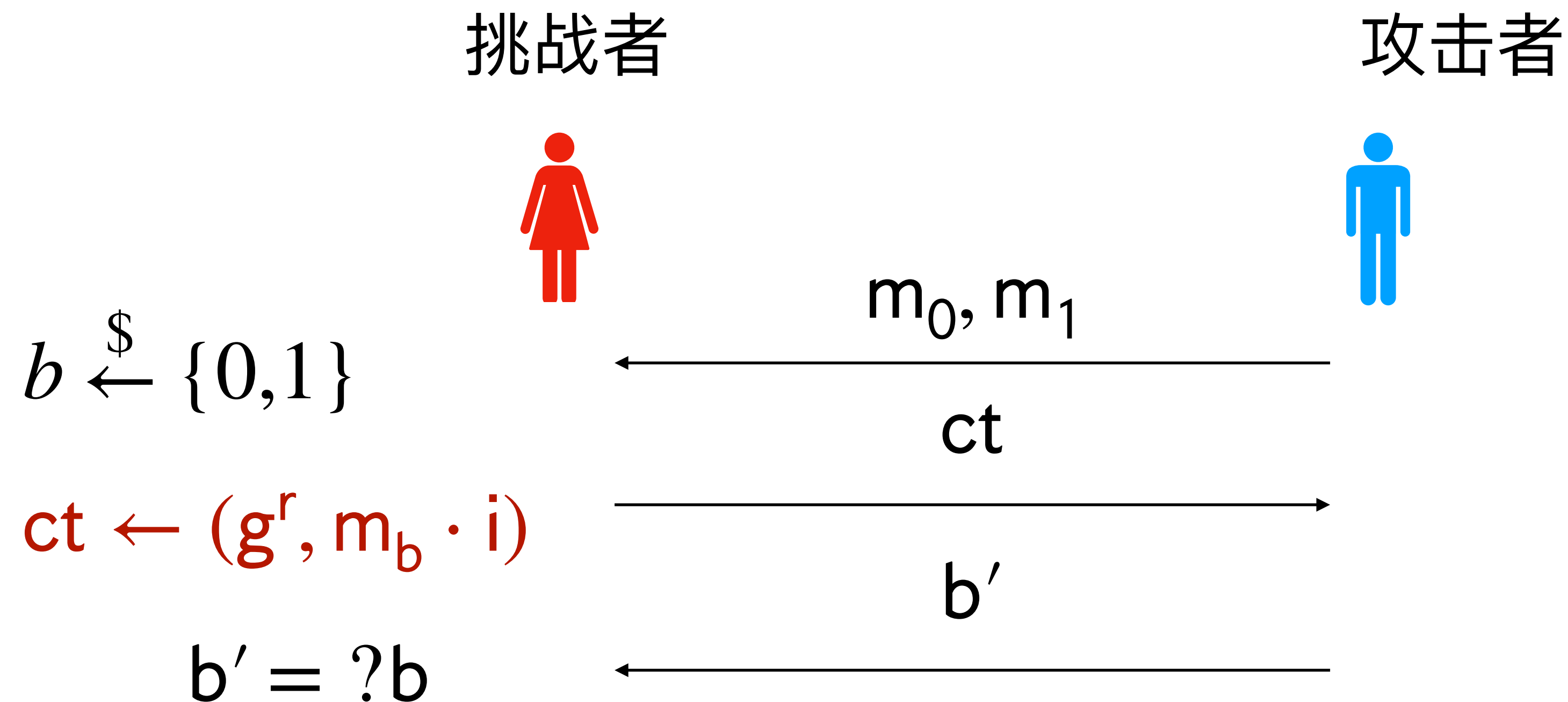
# 非对称加密 - IND-CPA

- 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
  - $pk = y = g^x, sk = x$
  - $ct = (g^r, m \cdot g^{xr})$



# 非对称加密 - IND-CPA

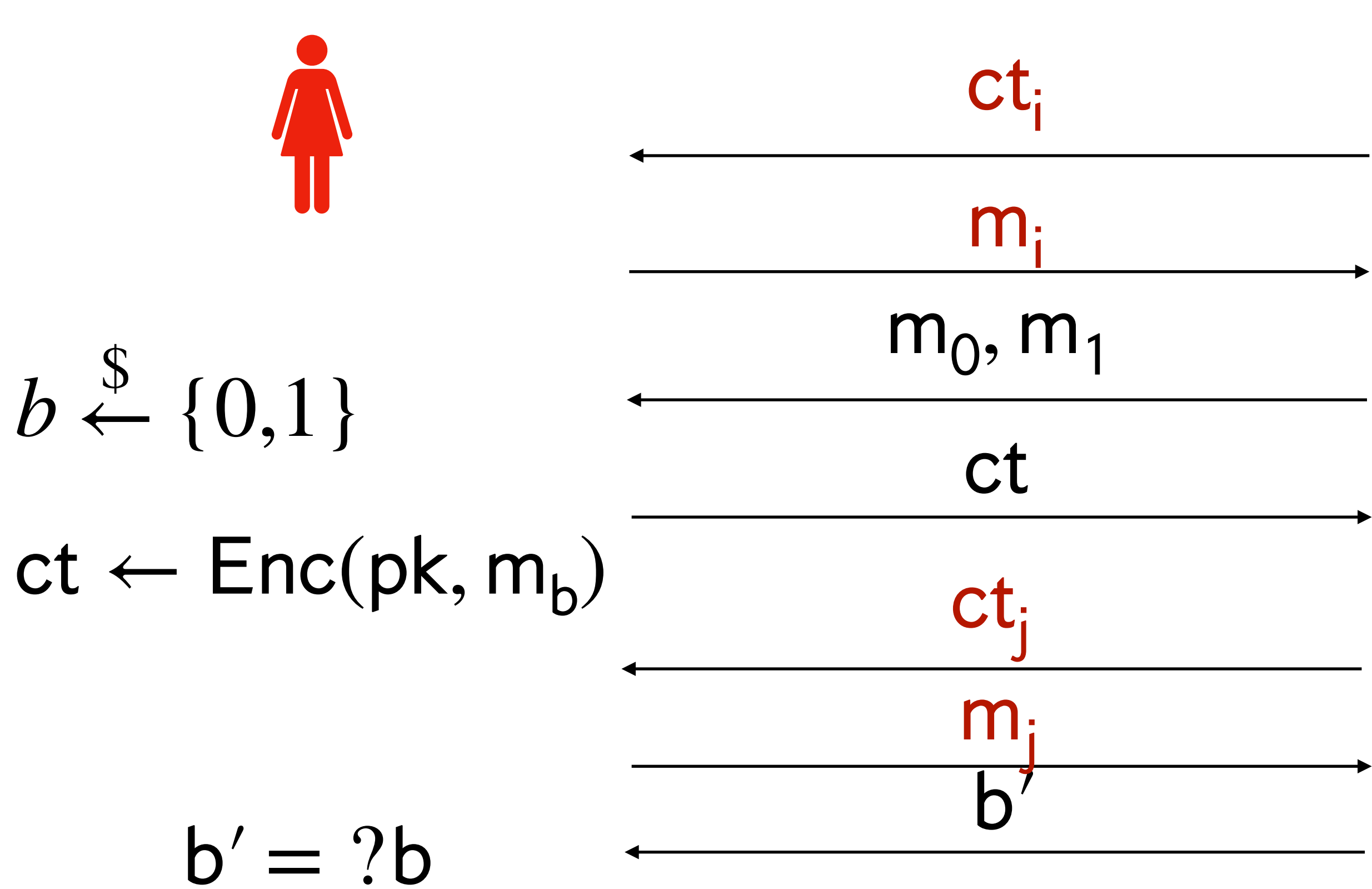
- $(f, g, f^r, g^r) \equiv_c (f, g, f^r, i)$
- DDH假设  $\rightarrow$  El-Gamall加密IND-CPA安全



# 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?  
挑战者

攻击者



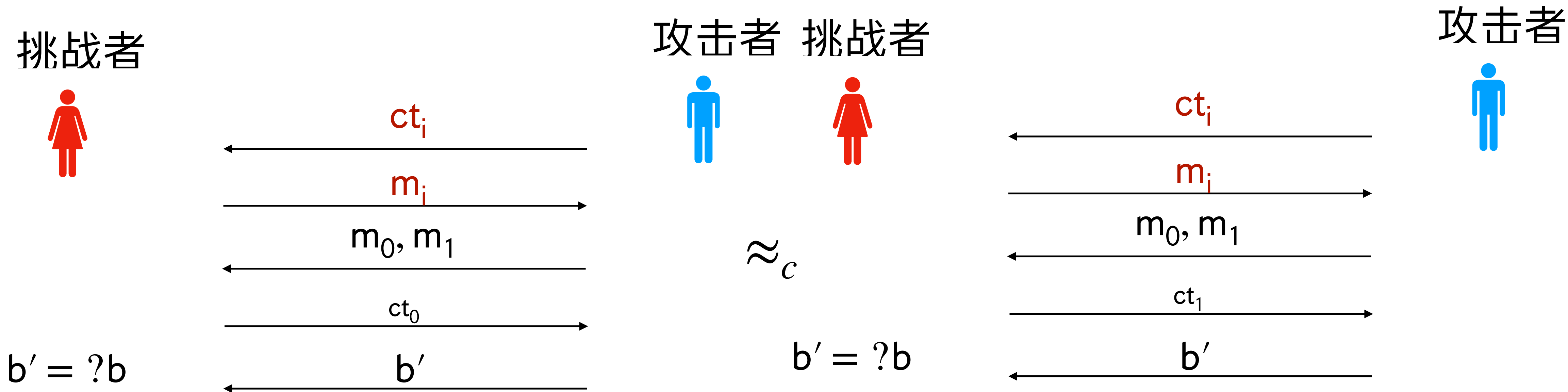
# 非对称加密算法 - IND-CCA

- 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预言机
  - IND-CCA2安全：在取得挑战密文之前和之后都可以询问解密预言机
- El-Gamml 算法 IND-CCA2 安全么？
  - 不
  - El-Gamml :  $ct = (g^r, m \cdot g^{ar})$  满足  $Enc(pk, 2m) = 2Enc(pk, m)$

# 非对称加密算法 - IND-CCA1

- Naor-Yung 构造 (STOC' 90) : 给定IND-CPA 安全的加密 $\mathcal{E} = (\text{Setup}, \text{Enc}, \text{Dec})$  和动态安全的零知识证明 $\mathcal{P} = (\text{Gen}, \text{Prove}, \text{Ver})$ 。下列加密算法是IND-CCA1安全的。
- $\text{Setup}(1^\lambda)$ :
  - $(pk_1, sk_1) \xleftarrow{\$} \text{Setup}(1^\lambda); (pk_2, sk_2) \xleftarrow{\$} \text{Setup}(1^\lambda); crs \xleftarrow{\$} \text{Gen}(1^\lambda)$
  - $pk = (crs, pk_1, pk_2), sk = (sk_1)$
- $\text{Enc}(pk, m)$  :
  - $r_1, r_2 \xleftarrow{\$} \{0, 1\}^*$ ;  $c_1 = \text{Enc}(pk_1, m; r_1); c_2 = \text{Enc}(pk_2, m; r_2)$
  - $\pi \xleftarrow{\$} \text{Prove}(crs, (c_1, c_2), (m, r_1, r_2)); ct = (c_1, c_2, \pi)$
- $\text{Dec}(sk, ct)$  : Check  $\text{Ver}(crs, (c_1, c_2), \pi)$ ; Return  $\text{Dec}(sk_1, c_1)$

# 为什么Naor-Yung构造是IND-CCA1安全?



# Naor-Yung证明

- 通过一系列变换从左边变到右边：Games
- $\text{Game}_0$  : 加密 $m_0$
- $\text{Game}_1$  : 和 $\text{Game}_0$ 相同, 但证明是由零知识证明的simulator产生( $\text{SimSetup}$ ,  $\text{Sim}$ ,  $\text{Ver}$ )
- $\text{Game}_2$  :  $c_2$ 是 $m_1$ 的加密
- $\text{Game}'_2$  : 用 $\text{sk}_2$ 解密
- $\text{Game}_3$  :  $c_1$ 是 $m_1$ 的加密
- $\text{Game}'_3$  : 用 $\text{sk}_1$ 解密
- $\text{Game}_3$  : 用( $\text{Gen}$ ,  $\text{Prove}$ ,  $\text{Ver}$ )产生证明

# Naor-Yung 证明

- $\text{Game}_0 \rightarrow \text{Game}_1$ : 零知识证明
- $\text{Game}_1 \rightarrow \text{Game}_2$ :  $\mathcal{E}_2$ 的IND-CPA安全性
- $\text{Game}_2 \rightarrow \text{Game}'_2$ :
  - 攻击者产生 $(c_1, c_2, \pi)$ , 通过验证但是 $\text{Dec}(\text{sk}, c_1) \neq \text{Dec}(\text{sk}, c_2)$ 。这个概率在 $\text{Game}'_2$ 和 $\text{Game}_0$ 中概率是一样的
  - $\text{sk}_2$ 解密结果与 $\text{sk}_1$ 相同
- $\text{Game}'_2 \rightarrow \text{Game}_3$ :  $\mathcal{E}_1$ 的IND-CPA安全性
- $\text{Game}_3 \rightarrow \text{Game}'_3$ : 同  $\text{Game}_2 \rightarrow \text{Game}'_2$
- $\text{Game}'_3 \rightarrow \text{Game}_4$ : 零知识证明, 同 $\text{Game}_0 \rightarrow \text{Game}_1$

# Naor-Yung 加密不是IND-CCA2安全

- 给定一个 $\mathcal{P} = (\text{Gen}, \text{Prove}, \text{Ver})$ , 构造 $\mathcal{P}'$ 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 $\mathcal{P}'$ 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 $\text{Enc}(\text{pk}, m_0), \text{Enc}(\text{pk}, m_1)$
  - 与挑战密文 $\text{ct}_b$ 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 $\text{ct}_b = \text{Enc}(m_b; r)$ 计算出在不知道私钥的情况下计算 $\text{ct}'_b = \text{Enc}(m_b; r')$
  - 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:
- $\text{Enc}(\text{pk}, m)$  :
  - $(\text{sk}, \text{vk}) \xleftarrow{\$} \text{SigGen}(1^\lambda)$
  - $r_1, r_2 \xleftarrow{\$} \{0, 1\}^*$ ;  $c_1 = \text{Enc}(\text{pk}_1, m; r_1)$ ;  $c_2 = \text{Enc}(\text{pk}_2, m; r_2)$
  - $\pi \xleftarrow{\$} \text{Prove}(\text{crs}, (c_1, c_2), (m, r_1, r_2))$ ;
  - $\sigma \leftarrow \text{Sig}(\text{sk}, (c_1, c_2, \pi))$
  - $\text{ct} = (\text{vk}, (c_1, c_2, \pi), \sigma)$

# DDN'91 - IND-CCA2安全加密算法

挑战者



攻击者



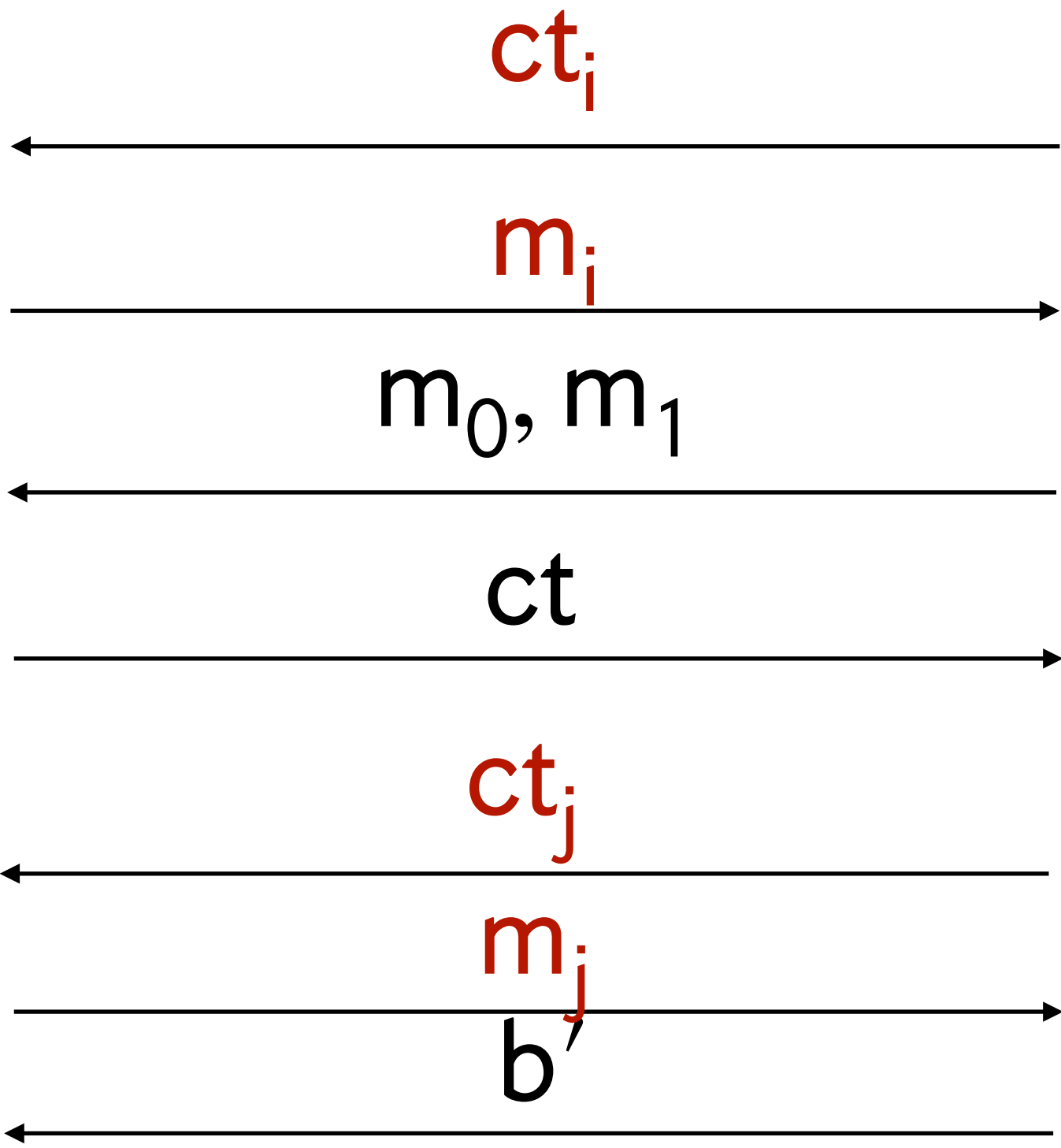
密文:

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

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

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

$b' = ?b$



# 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

- $pk = \begin{pmatrix} pk_{1,0} & \dots & pk_{k,0} \\ pk_{1,1} & \dots & pk_{k,1} \end{pmatrix}$ ，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

- $\text{Enc}(\text{pk}, m)$  :
  - $(\text{sk}, \text{vk}) \xleftarrow{\$} \text{SigGen}(1^\lambda)$
  - $r_1, \dots, r_k \xleftarrow{\$} \{0, 1\}^*$ ;  $c_i = \text{Enc}(\text{pk}_{i, \text{vk}_i}, m; r_i)$  for all  $i \in \{1, \dots, k\}$
  - $\pi \xleftarrow{\$} \text{Prove}(\text{crs}, (\{c_i\}_{i \in [k]}), (m, \{r_i\}_{i \in [k]}))$ ;
  - $\sigma \leftarrow \text{Sig}(\text{sk}, (\{c_i\}_{i \in [k]}, \pi))$
  - $\text{ct} = (\text{vk}, (\{c_i\}_{i \in [k]}, \pi), \sigma)$
- $\text{Dec}(\text{sk}, \text{ct})$  :
  - Check  $\text{Ver}(\text{crs}, (\{c_i\}_{i \in [k]}), \pi)$ ; Check  $\text{Ver}(\text{vk}, (\{c_i\}_{i \in [k]}, \pi), \sigma)$
  - Return  $\text{Dec}(\text{sk}_{1, \text{vk}_1}, c_{1, \text{vk}_1})$

# DDN'91 - IND-CCA2安全加密算法

- 首先证明：
  - 令  $ct^* = (vk^*, c^*, \sigma^*)$
  - 首先观察得知：如果攻击者向挑战者询问了  $ct = (vk^*, c', \sigma')$  且
    - $Ver(vk^*, c, \sigma) = 1$
    - $(c, \sigma) \neq (c^*, \sigma^*)$
  - 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

- $\text{Game}_0$  : 加密 $m_0$
- $\text{Game}_1$  : 和 $\text{Game}_0$ 相同, 但证明是由零知识证明的simulator产生( $\text{SimSetup}, \text{Sim}, \text{Ver}$ )
- $\text{Game}'_1$  : 所有挑战密文如果包括 $\text{vk}^*$ , 则直接拒绝
- $\text{Game}_2$  : 令 $\ell$ 是 $\text{vk}$ 和 $\text{vk}^*$ , 的第一个不相同比特。
- $\text{Game}'_2$  : 用 $\text{sk}_{\ell, \text{vk}_\ell}$ 进行解密
- $\text{Game}''_2$  : 将所有密文替换成为 $m_1$ 的加密
- $\text{Game}_3$  : 用 $\text{sk}_1$ 解密
- $\text{Game}'_3$  : 不再拒绝含 $\text{vk}^*$ 的密文
- $\text{Game}''_3$  : 用( $\text{Gen}, \text{Prove}, \text{Ver}$ )产生证明