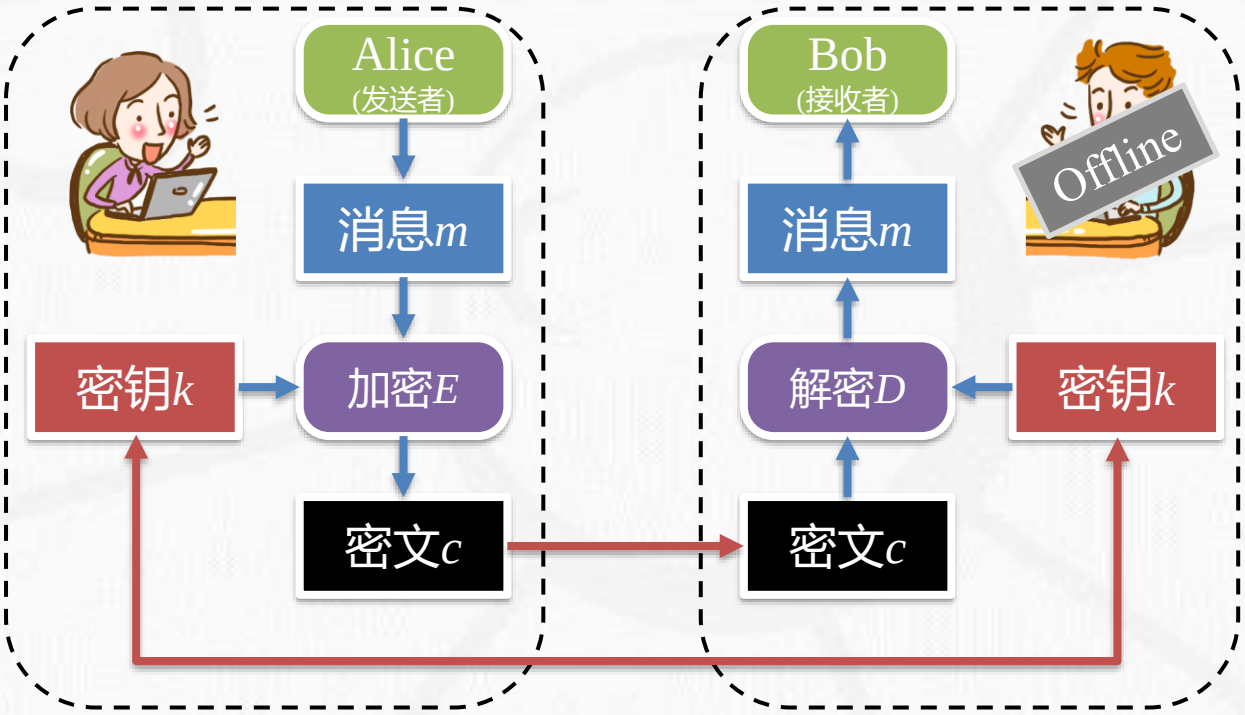

西安電子科技大學
XIDIAN UNIVERSITY

用公钥加密、用私钥解密

# §2.1.5 非对称密码算法

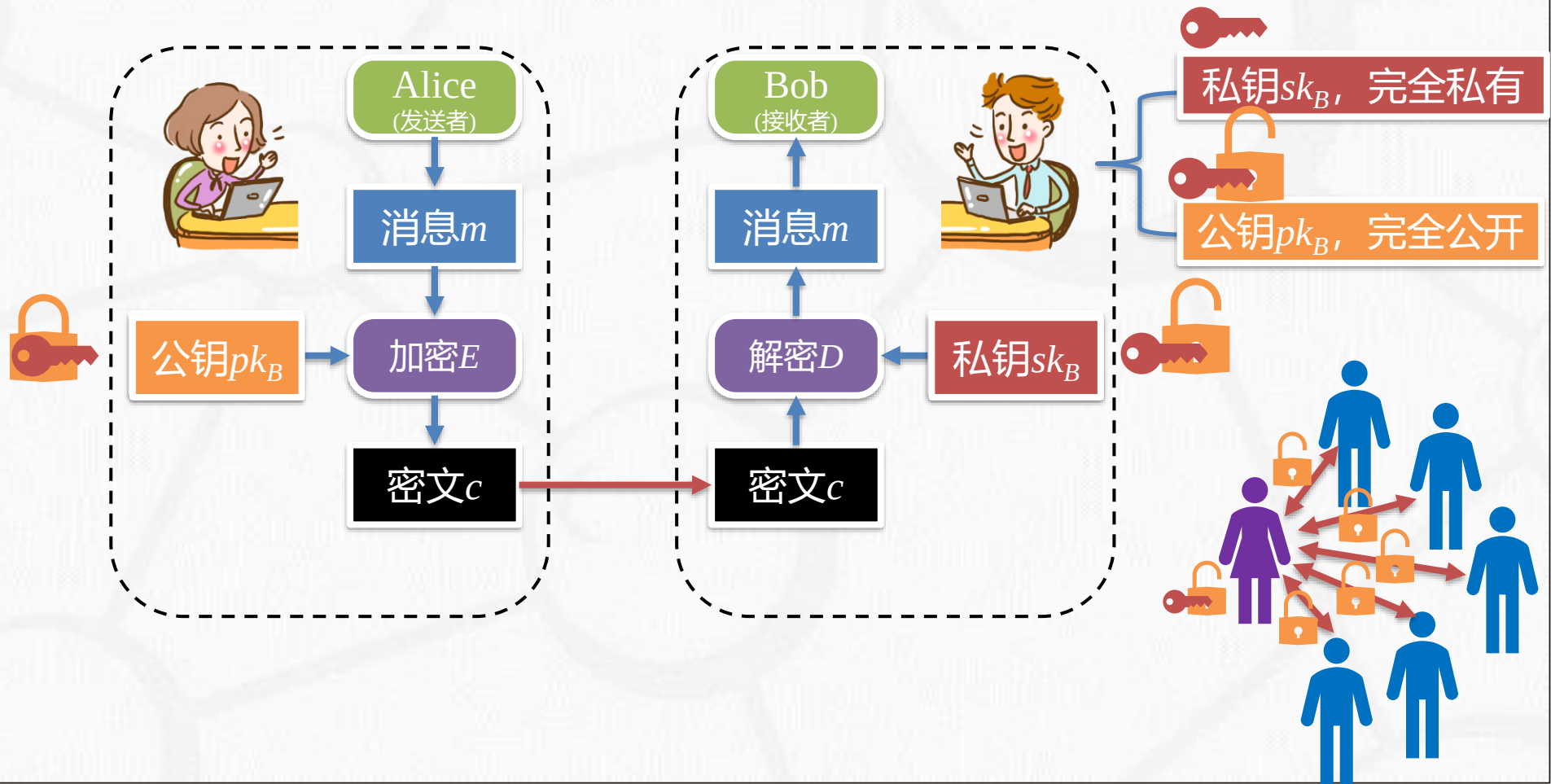
# §2.1.5 非对称密码算法 - RSA的引子

- 对称密码体制
  - DES/AES/SM4等
  - 用相同的密钥进行加密和解密
- 密钥协商
  - D-H密钥交换
- 存在问题
  - 通信开销大(每次保密通信，均需重新协商密钥)
  - 密钥量大(密钥总数随着用户的增加而迅速增加，$O(n^2)$)
  - 必须双方在线(Online)
  - 难以追责(Bob可轻易伪造Alice的消息)




# §2.1.5 非对称密码算法 - 目标

- 场景：多用户、异步信息发送、追责溯源等
- 目标：用公钥加密，用私钥解密



# §2.1.5 非对称密码算法 - 历史

- 非对称加密体制又称为公钥加密体制
  - 1976年W. Diffie和M. Hellman在《密码学的新方向》中提出公钥密码体制的思想，开辟了密码学研究的新篇章。
  - 1977年R. Merkle和M. Hellman也独立提出了一个具体的公钥密码算法Knapsack。已被证明不安全。
  - 1978年R. Rivest，A. Shamir和L. Adleman提出的一种用数论构造的、也是迄今为止理论上最为成熟完善的公钥密码体制(RSA)，该体制已得到广泛的应用。

RON LINN RIVEST

ADI SHAMIR

LEONARD (LEN)
MAX ADLEMAN

2002年图灵奖得主(RSA)

New Directions in Cryptography

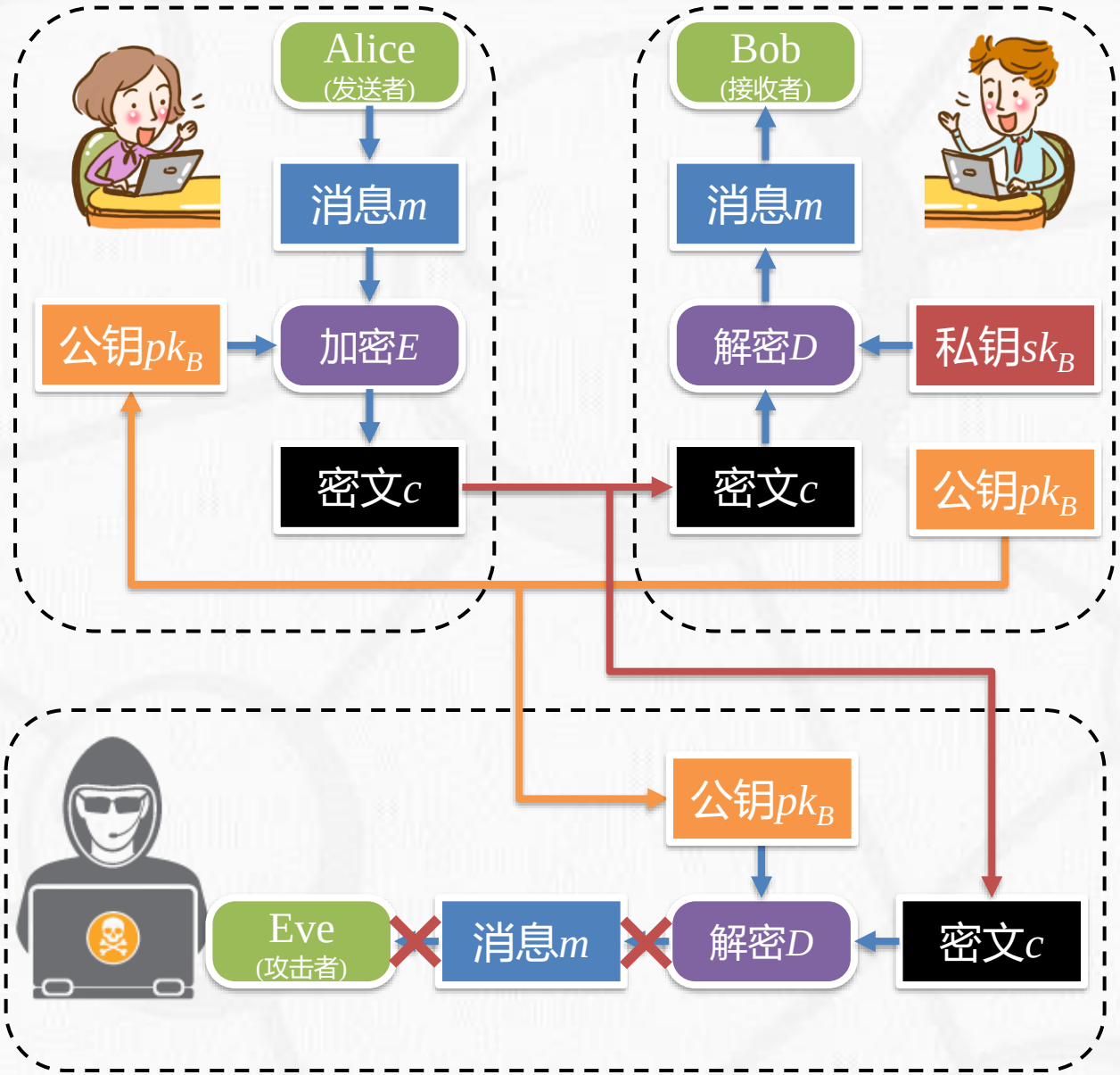
Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

《密码学的新方向》

# §2.1.5 非对称密码算法 - 通信流程

- Bob生成公私钥对
- Bob将自己的公钥发送给Alice
- Alice用Bob的公钥进行信息加密
- Alice将密文发送给Bob
- Bob用自己的私钥解密得到消息
- 敌手无法获得消息



# §2.1.5 非对称密码算法 - 功能及安全需求

- 非对称加密体制的具体需求
- 总体目标：用公钥加密，用私钥解密

| 正确性和可用性 | 安全性 |
| --- | --- |
| **计算上是容易的** | **计算上是不可行的** |
| 接收方产生密钥对（$pk$和$sk$） | 由公钥$pk$求私钥$sk$ |
| $c=E(pk,m)$ | 由密文$c$和公钥$pk$恢复明文$m$ |
| $m=D(\mathrm{sk},c)$ | |

- 额外需求：加、解密次序可换
  - 即$E(pk, D(sk, m)) = D(sk, E(pk, m))$。

# §2.1.5 非对称密码算法 - 陷门单向函数

- 定义 1(陷门单向函数)：给定限门$t$，陷门单向函数(Trapdoor one-way function)是一个单向函数$f(x,t)\rightarrow y$，且满足以下条件：
  - 在不知陷门信息$t$下，由$y$求$x$极为困难；
  - 当知道陷门信息$t$后，由$y$求$x$是易于实现的。
  - 其中，极为困难是对现有的计算资源和算法而言。

$m^e \bmod p \xrightarrow{\text{Easy!}} ? = c$

$?^e \bmod p \xleftarrow{\text{Hard!}} c$

**单向函数(离散对数求解问题)**

**如何构造陷门解密$c$!**

$(m^e)^d \bmod p = m$

$\Downarrow$

$ed = 1 \bmod p$

# §2.1.5 非对称密码算法 - 大整数分解

- 定义 2(大整数分解FAC)：
  - 正向：已知两个大素数$p$和$q$，求$n=pq$。
  - 逆向：已知大整数$n=pq$，求解两个大素数$p$和$q$。

$p, q \xrightarrow{\text{Easy!}} ? = pq$

$?, ? \xleftarrow{\text{Hard!}} y = pq$

**大整数分解问题**

- 已知的各种算法的渐近运行时间约为：
  - 试除法：$n/2$。
  - 二次筛(QS)：$O(\exp\sqrt{\ln n \ln\ln n})$。
  - 椭圆曲线(EC)：$O(\exp\sqrt{2\ln p \ln\ln p})$。
  - 数域筛(NFS)：$O(\exp(1.92(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}))$。

# §2.1.5 非对称密码算法 - RSA陷门单向函数

- 定义 3(RSA中的陷门单向函数)：已知两个大素数$p,q$和$n=pq$，选择两个大整数$e$和$d$，满足$ed$是$\varphi(n)=(p-1)(q-1)$的倍数加1。
  - 陷门：为$p$和$q$。
  - 问题：已知$n$和$e$，求$d$。

$n$的欧拉函数!

$\varphi(n), e \xrightarrow{\text{Easy!}} d$

$n, e \xrightarrow{\text{Hard!}} d$

基于大整数分解构造的陷门单向函数

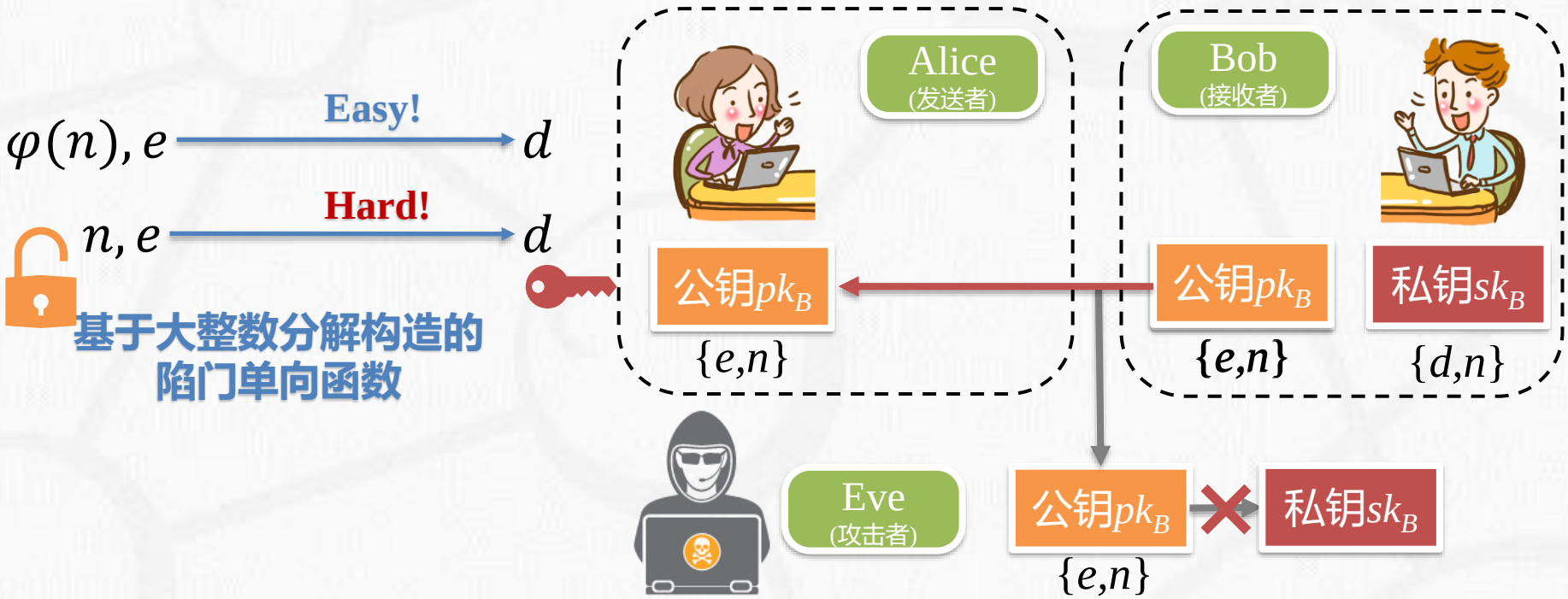
$$
\begin{aligned}
&(m^e)^d \bmod n \\
=&m^{ed} \bmod n \\
=&m^{k\varphi(n)+1} \bmod n \\
=&m^{k\varphi(n)} \cdot m^1 \bmod n \\
=&(m^{\varphi(n)})^k \cdot m^1 \bmod n \\
=&(m^{\varphi(n)} \bmod n)^k \cdot m^1 \bmod n \\
=&\ m^1 \bmod n
\end{aligned}
$$

欧拉函数的性质：
$m^{\varphi(n)} \bmod n = 1 \bmod n$

# §2.1.5 非对称密码算法 - RSA的密钥产生

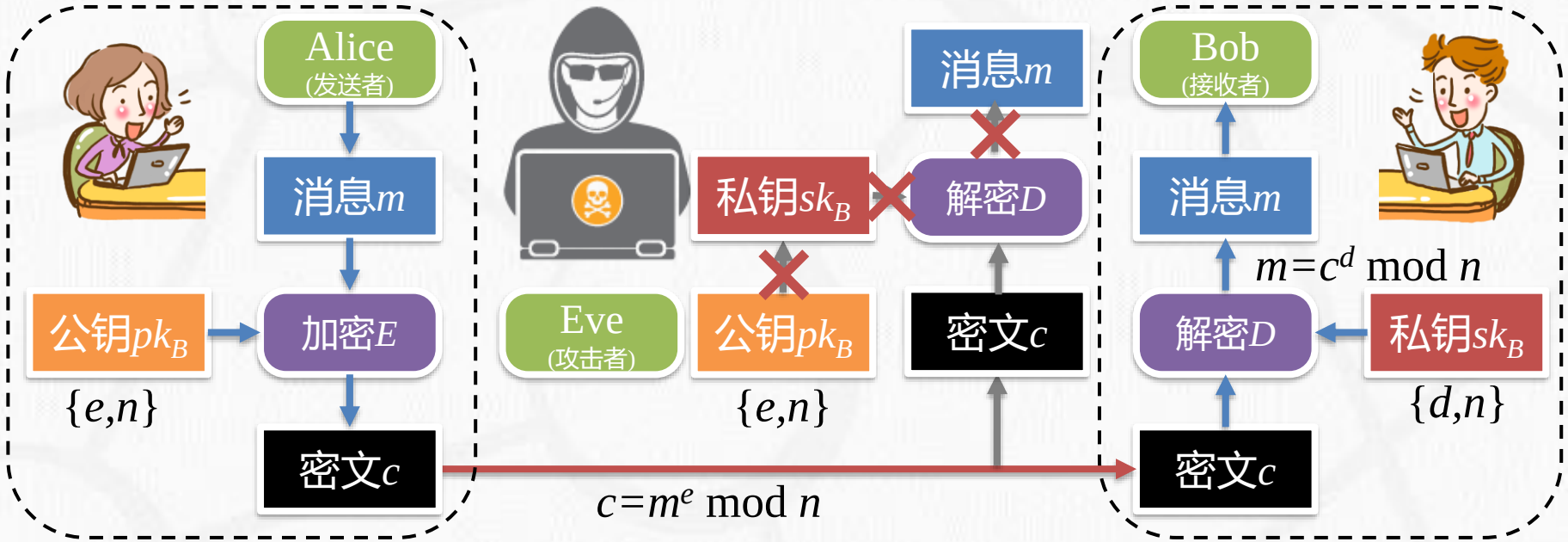
- 1. 密钥的产生
  - 选两个保密的大素数$p$和$q$。
  - 计算$n=pq$，$\varphi(n)=(p-1)(q-1)$，其中$\varphi(n)$是$n$的欧拉函数值。
  - 选一整数$e$，满足$1<e<\varphi(n)$，且$gcd(\varphi(n),e)=1$。
  - 计算$d$，满足$de\equiv 1 \bmod \varphi(n)$，即$d$是$e$在模$\varphi(n)$下的乘法逆元，因$e$与$\varphi(n)$互素，由模运算可知，它的乘法逆元一定存在。
  - 以$\{e,n\}$为公钥$pk$，$\{d,n\}$为私钥$sk$。

限门单向函数的具体应用！



# §2.1.5 非对称密码算法 - RSA加密和解密

- 2. 加密
  - 加密时首先将明文比特串分组，使得每个分组对应的十进制数小于$n$，即分组长度小于$\log_2 n$。
  - 然后对每个明文分组$m$，作加密运算： $c \equiv m^e \bmod n$。
- 3. 解密
  - 对密文分组的解密运算为：$m \equiv c^d \bmod n$。



# §2.1.5 非对称密码算法 - RSA正确性和安全性

- 正确性：
  - $c^d \bmod n \equiv m^{ed} \bmod n \equiv m^{1 \bmod \varphi(n)} \bmod n \equiv m^{k\varphi(n)+1} \bmod n$。
  - $m$与$n$互素。
  - 由Euler定理得$m^{\varphi(n)} \equiv 1 \bmod n$，因此$c^d \bmod n \equiv m$。

满足公钥和私钥的可交换性!

- 安全性：
  - 离散对数求解问题。
  - 大整数分解FAC。
  - 基于大整数分解构造的陷门单向函数。

$\varphi(n), e \xrightarrow{\text{Easy!}} d$

$n, e \xrightarrow{\text{Hard!}} d$

基于大整数分解构造的陷门单向函数

# §2.1.5 非对称密码算法 - RSA实例

- 例子：
  - 给定$p=7, q=17$。则可计算$n=pq=119, \varphi(n)=(p-1)(q-1)=96$。假定私钥是$d=11$和明文$m=8$。
  - 问1：计算公钥$e = d^{-1} \bmod \varphi(n) = 35$。
  - 问2：计算明文对应的密文$c$。 $c \equiv me \bmod n$
  - 问3：试对密文进行解密，得到明文$m'$。 $m' \equiv cd \bmod n$

公钥$pk_B$ $\{e,n\}$　　私钥$sk_B$ $\{d,n\}$

# §2.1.5 非对称密码算法 - RSA实例

- 例子：
  - 给定$p=7, q=17$。则可计算$n=pq=119, \varphi(n)=(p-1)(q-1)=96$。假定私钥是$d=11$和明文$m=8$。
  - 问1：计算公钥$e = d^{-1} \bmod \varphi(n) = 35$。
  - 问2：计算明文对应的密文$c$。 $c \equiv me \bmod n$
  - 问3：试对密文进行解密，得到明文$m'$。 $m' \equiv cd \bmod n$

公钥$pk_B$ $\{e,n\}$　　私钥$sk_B$ $\{d,n\}$

$$c = m^d \bmod n = 8^{11} \bmod 119 = 36.$$

$$m' = c^e \bmod n = 36^{35} \bmod 119 = 8.$$

# §2.1.5 非对称密码算法 - RSA参数选择

- 目前普遍使用的参数范围是
  - $2^{511}<p<2^{512}$，
  - $2^{511}<q<2^{512}$。
  - 目前已经扩展到$2^{2047}<q<2^{2048}$。

$$\varphi(n), e \xrightarrow{\text{Easy!}} d$$

$$n = pq, e \xrightarrow{\text{Hard!}} d$$

**基于大整数分解构造的陷门单向函数**

- 如果Eve欲穷举$p$的所有可能值，则穷举的次数约为
  - $(2^{512}-2^{511})/2=2^{510}$（次）。

- 因此，基本RSA似乎是足够安全的。

# §2.1.5 非对称密码算法 - RSA的其它问题

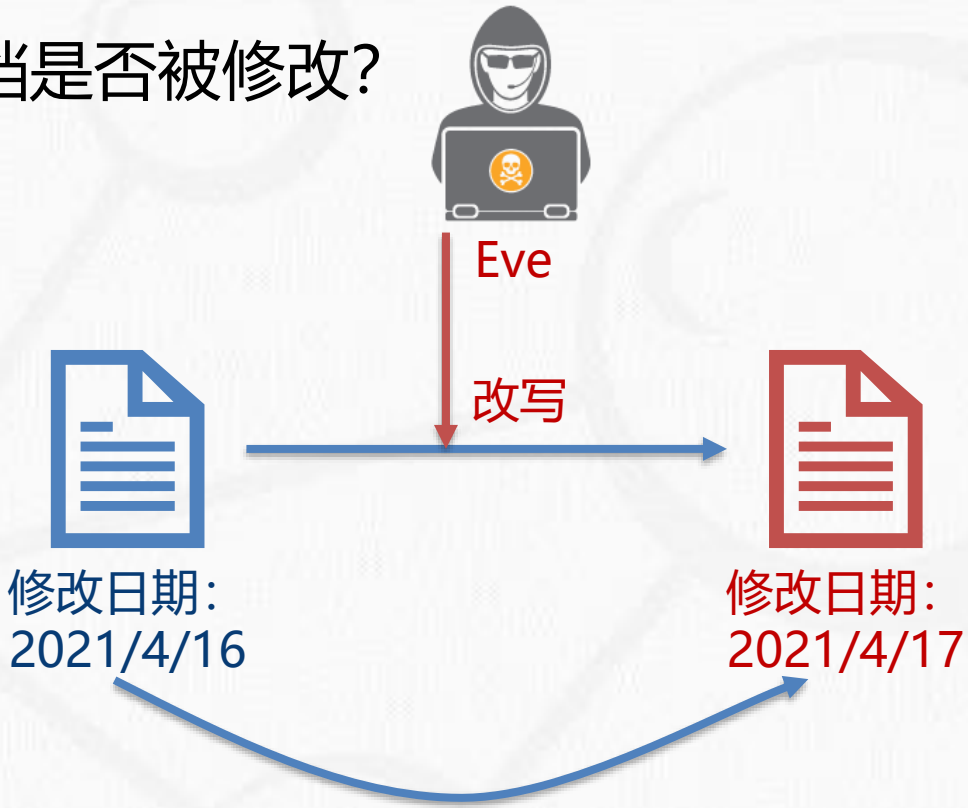
- 如何选择大素数$p$和$q$?
  - 素性检测：Miller-Rabin素性检测方法
- 如何防止传递攻击?
  - 哈希函数的引入。
- 其它非对称加密体制:
  - 背包公钥密码、Rabin、ElGamal、NTRU、ECC(SM2)、McEliece等。

消息的指纹、完整性、不可篡改

# §2.1.6 散列算法

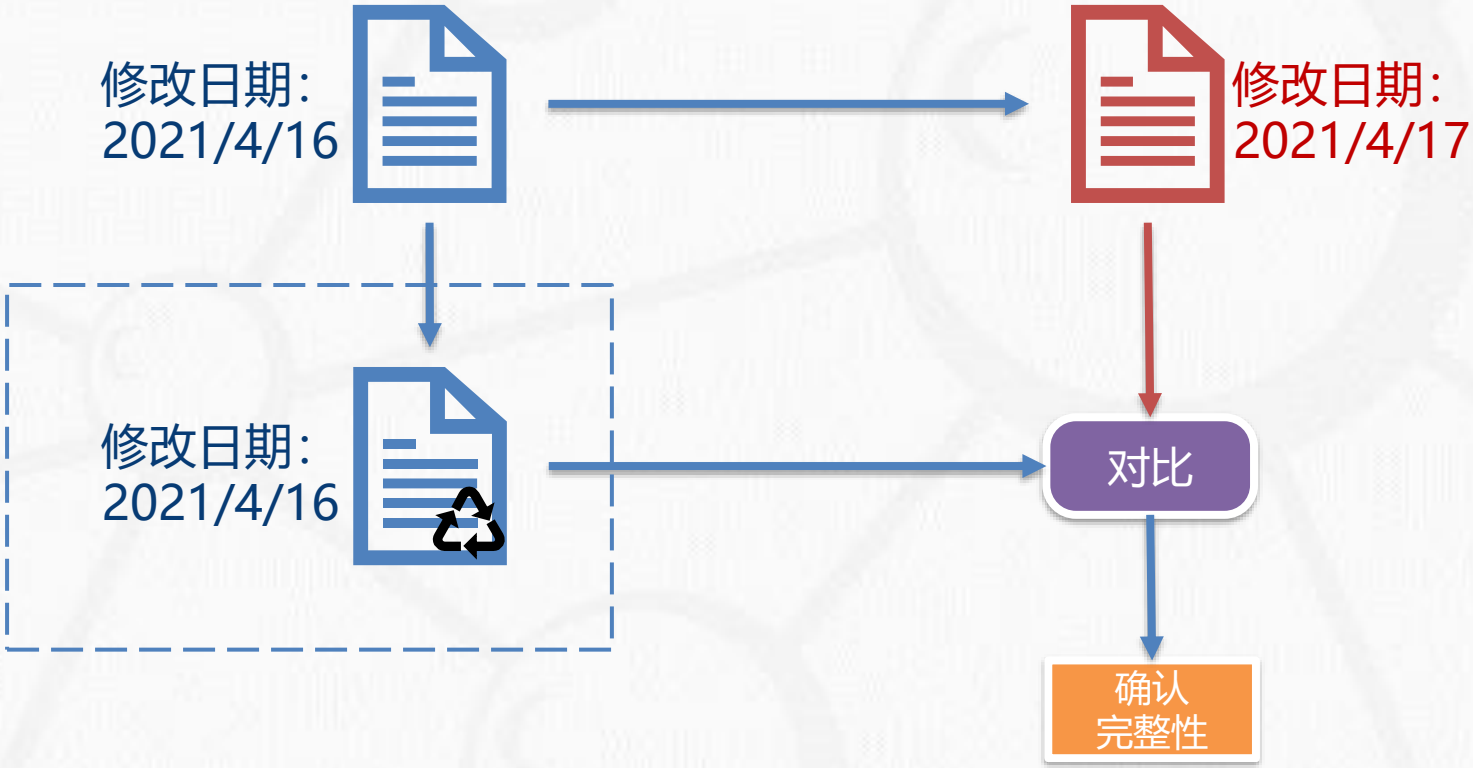
# §2.1.6 散列算法 - 引子(1)

- 如何判定文档是否被修改?



# §2.1.6 散列算法 - 引子(2)

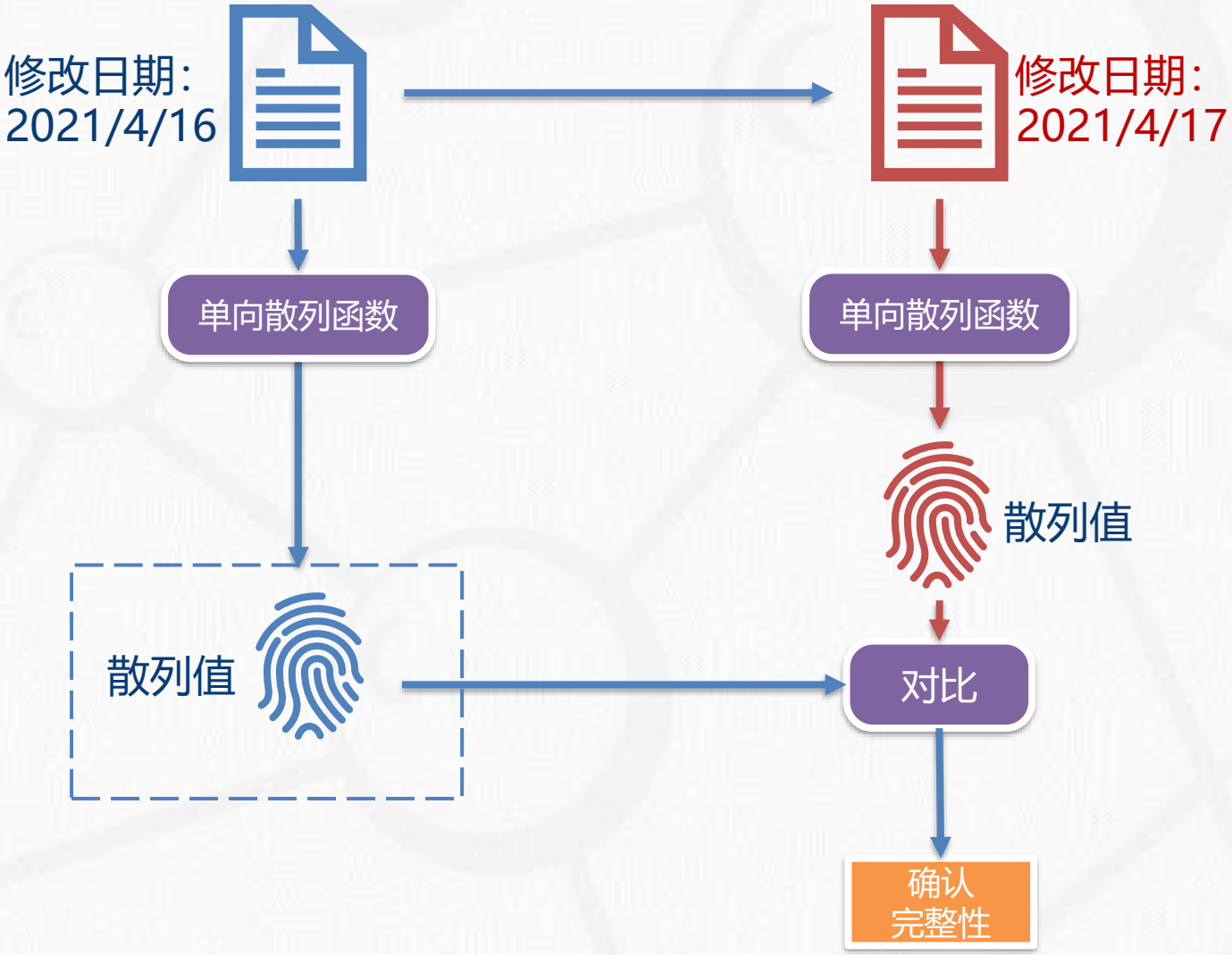
- 如何判定文档是否被修改?




- 存在问题:
  - 占用存储空间大、对比计算复杂度高

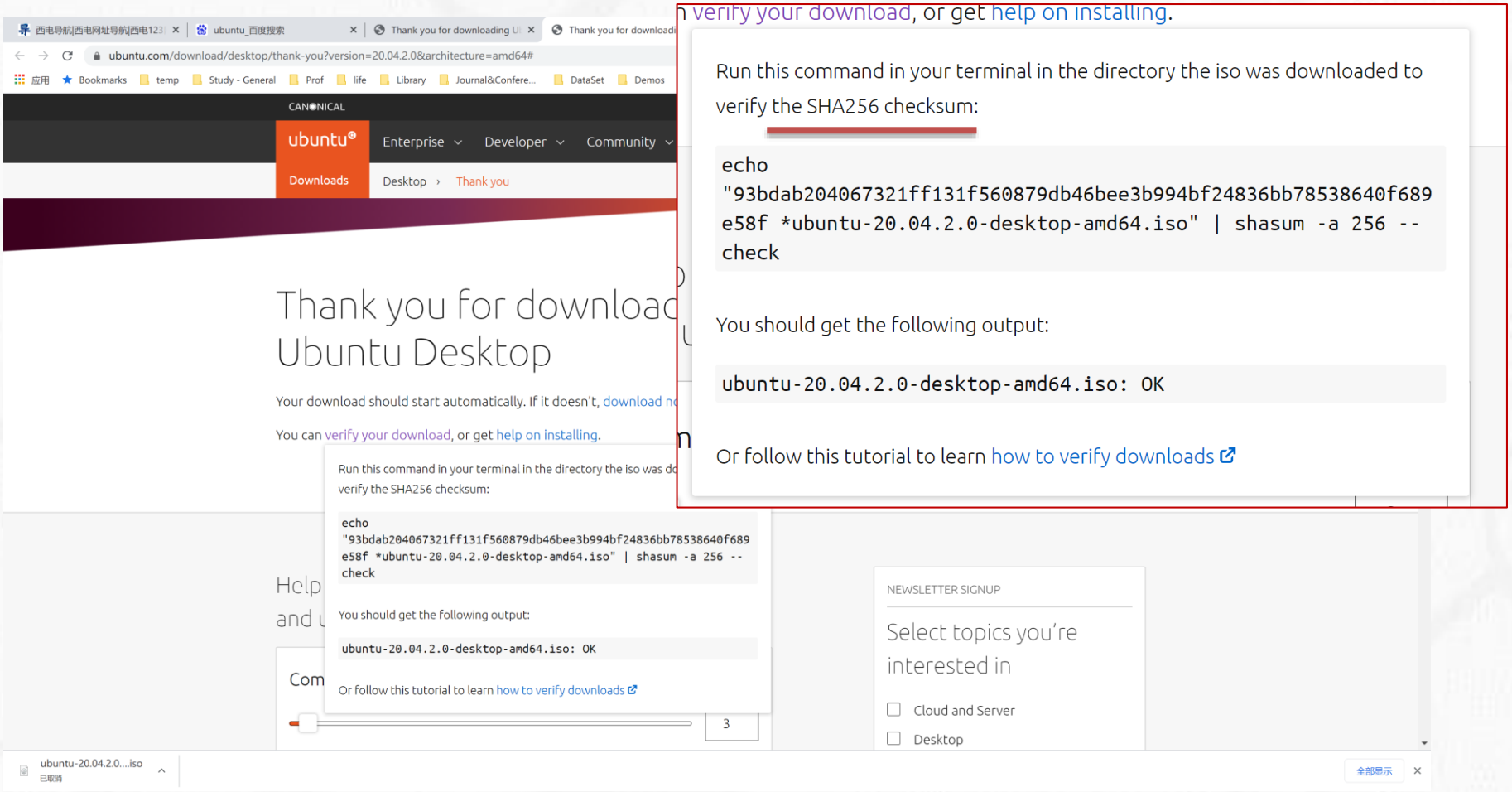
# §2.1.6 散列算法 - 引子(3)

- 如何判定文档是否被修改?



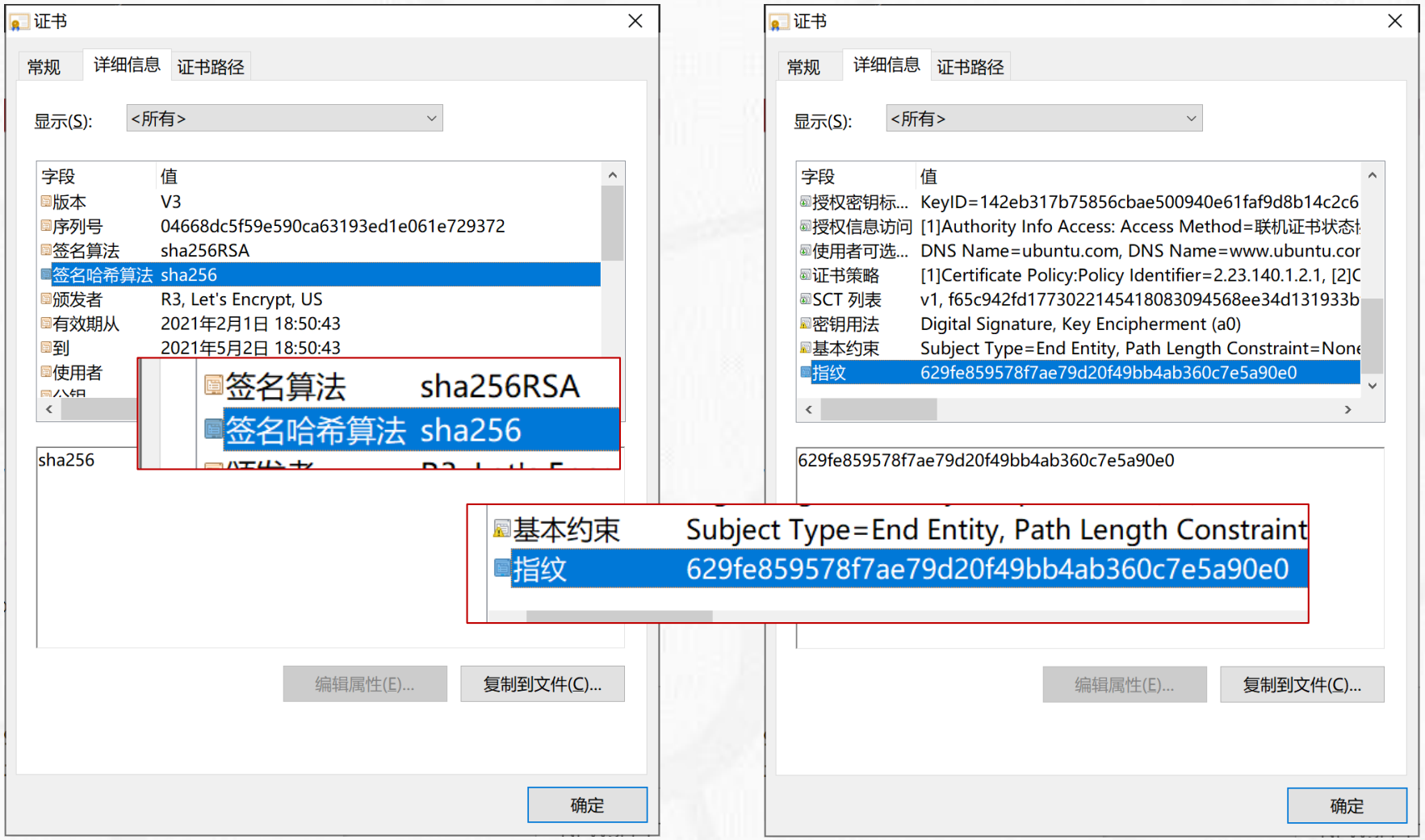
# §2.1.6 散列算法 - 应用案例一

- 文件服务器中的文件完整性验证



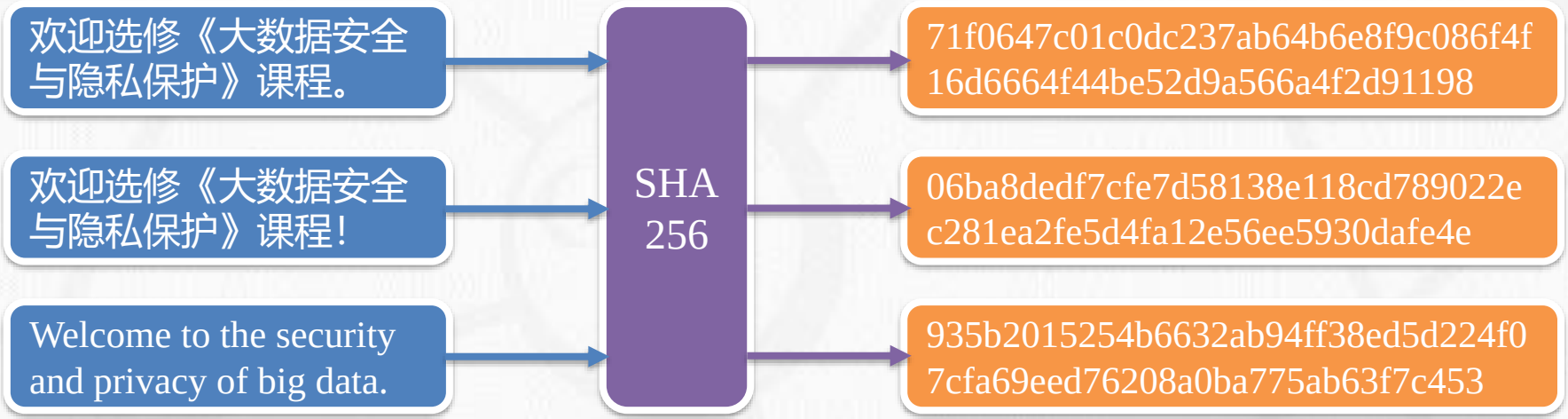
# §2.1.6 散列算法 - 应用案例二

- 公钥证书的完整性验证



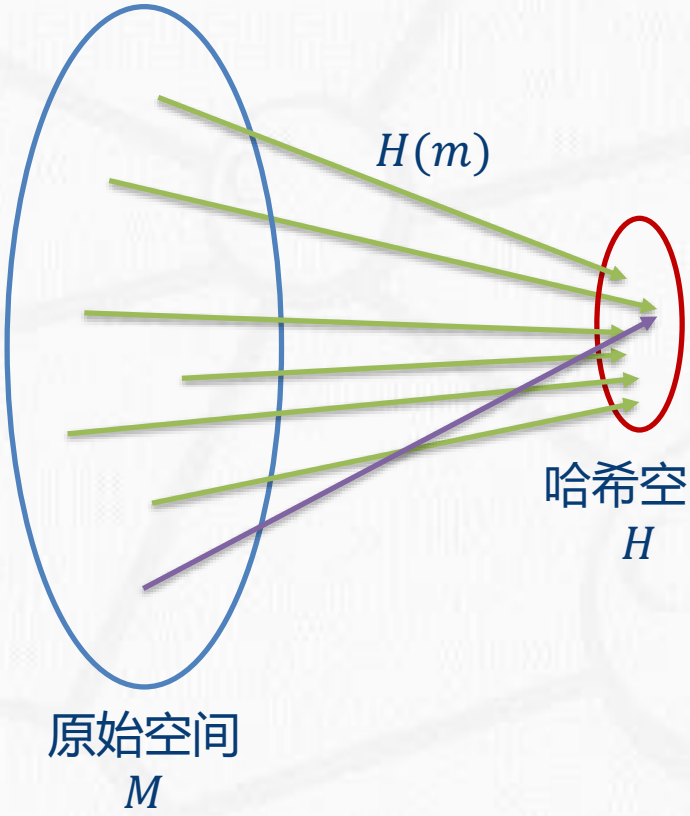
# §2.1.6 散列算法 - 定义

- 散列函数(Hash Function)：
  - $H(m)$，输入为任意长度的消息$m$，输出为一个固定长度的散列值，称为消息摘要(Message Digest)。
- 这个散列值是消息$m$的所有位的函数并提供错误检测能力
  - 消息中的任何一位或多位的变化都将导致该散列值的变化。
- 别名：
  - 哈希函数、数字指纹、压缩函数、紧缩函数、数据鉴别码、篡改检验码。



# §2.1.6 散列算法 - 新的问题

- 用以鉴别的散列函数，能否减弱认证方案的安全性？这个问题是要分析的。签名的对象由完整消息变成消息摘要，这就有可能出现伪造。




# §2.1.6 散列算法 - 安全威胁一

- 伪造方式一
  - Oscar以一个有效签名$(x, y)$开始，此处$y = sig_k\,(H(x))$。首先他计算$Z = H(x)$，并企图找到一个$x'$满足$H(x') = H(x)$。若他做到这一点，则$(x', y)$也将为有效签名。为防止这一点，要求函数$H$具有无碰撞特性。
- 定义1(弱无碰撞)
  - 散列函数$H$称为是弱无碰撞的，是指对给定消息$x \in X$，在计算上几乎找不到异与$x$的$x' \in X$使$H(x) = H(x')$。

# §2.1.6 散列算法 - 安全威胁二

- 伪造方式二
  - Oscar首先找到两个消息$x = x'$，满足$H(x) = H(x')$，然后Oscar把$x$给Bob且使他对$x$的摘要$H(x)$签名，从而得到$y$，那么$(x', y)$是一个有效的伪造。
- 定义2(强无碰撞)
  - 散列函数h称为是强无碰撞的，是指在计算上几乎不可能找到异与$x$的$x' \in X$使$H(x) = H(x')$。

**注：强无碰撞自然含弱无碰撞！**

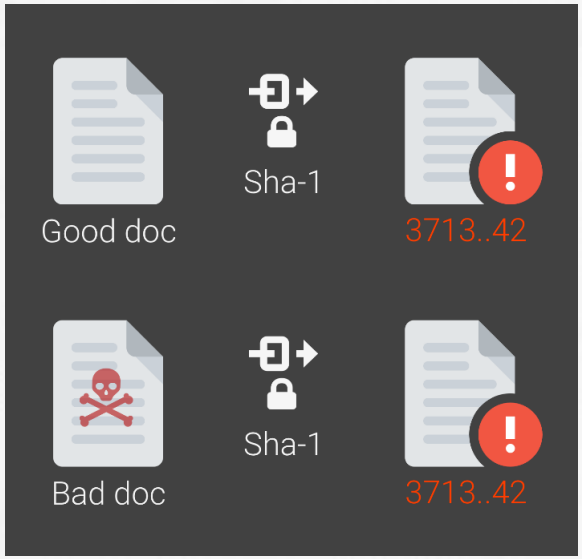
# §2.1.6 散列算法 - 安全威胁三

- 伪造方式三
  - 在散列函数的用法$(e)$中，秘密值$S$本身并不发送，如果散列函数不是单向的,攻击者截获到$m$和$H(m||S)$。然后通过某种逆变换获得$m||S$，因而攻击者就可以得到$S$。
- 定义3(单向的)
  - 称散列函数$H$为单向的，是指计算$H$的逆函数$H^{-1}$在计算上不可行。

# §2.1.6 散列算法 - 分类

## 安全水平

- 弱无碰撞；
- 强无碰撞。

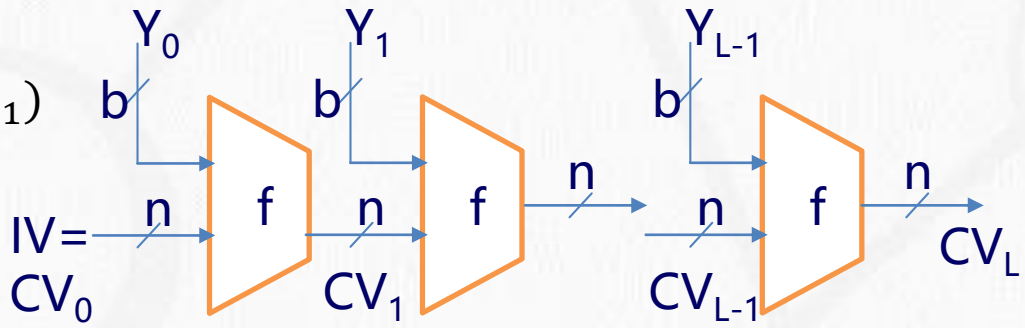



## 是否使用密钥?

- 带秘密密钥的Hash函数：
  - 消息的散列值由只有通信双方知道的秘密密钥K来控制。此时，散列值称作MAC。
- 不带秘密密钥的Hash函数：
  - 消息的散列值的产生无需使用密钥。此时，散列值称作MDC。

# §2.1.6 散列算法 - 通用构造

- 由Merkle于1989年提出
- Ron Rivest于1990年提出MD4
- 几乎被所有哈希函数使用
- 具体做法:
  - 把原始消息$m$分成一些固定长度的块$Y_i$
  - 最后一块padding并使其包含消息$m$长度
  - 设定初始值$CV_0$
  - 压缩函数$f$，$CV_i = f(CV_{i-1}, Y_{i-1})$
  - 最后一个$CV_i$为哈希值




A CERTIFIED DIGITAL SIGNATURE

*Ralph C. Merkle*
*Xerox PARC*
*3333 Coyote Hill Road,*
*Palo Alto, Ca. 94304*
merkle@xerox.com
*(Subtitle: That Antique Paper from 1979)*

Abstract
A practical digital signature system based on a conventional encryption function which is as secure as the conventional encryption function is described. Since certified conventional systems are available it can be implemented quickly, without the several years delay required for certification of an untested system.

Key Words and Phrases: Public Key Cryptosystem, Digital Signatures, Cryptography, Electronic Signatures, Receipts, Authentication, Electronic Funds Transfer.

CR categories: 3.56, 3.57, 4.9

1. *Introduction*

Digital signatures promise to revolutionize business by phone (or other telecommunication devices)[1] but currently known digital signature methods [5,6,7,8,10,13] either have not been certified, or have other drawbacks. A signature system whose security rested on the security of a conventional cryptographic function would be "pre-certified" to the extent that the

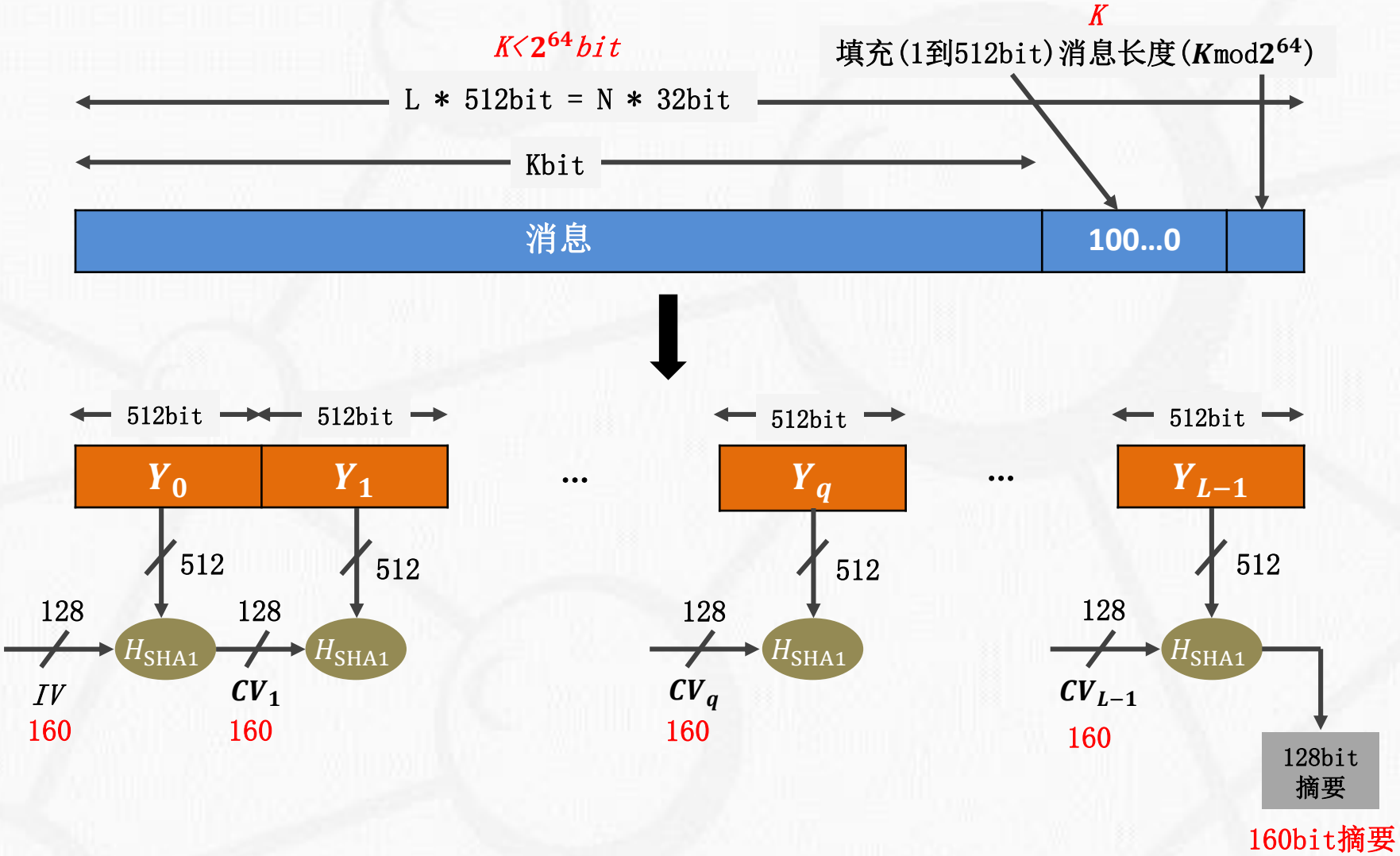
# §2.1.6 散列算法 - SHA系列算法

- SHA系列Hash函数是由美国标准与技术研究所(NIST)设计的。
  - 1993年公布了SHA0(FIPS PUB 180)，后来发现它不安全。
  - 1995年又公布了SHA1(FIPS PUB 180-1)。
  - 2005年王小云给出一种攻击SHA1的方法，用$2^{69}$次操作找到一个强碰撞，以前认为是$2^{80}$。
  - 2002年又公布了SHA2(FIPS PUB 180-2)。
  - SHA2包括3个Hash函数：SHA-256， SHA-384， SHA-512。
  - NIST于2007年公开征集SHA3，Keccak算法赢得了SHA3竞赛的最终胜利， 2014年(FIPS 202) 草案， NIST于2015年8月最终批准SHA3。
  - 2010年12月17日,《SM3密码杂凑算法》由国家密码管理局于发布。我国第一个密码散列函数标准。

# §2.1.6 散列算法 - SHA1的发展

- SHA1 是在MD5 的基础上发展起来的。
  - 采用Merkle提出了安全Hash模型。
  - 已被美国政府和许多国际组织采纳作为标准。
- SHA1的算法
  - 输入：长度小于$2^{64}$位的报文，
  - 输出：160位的报文摘要，
  - 过程：对输入按512位进行分组，并以分组为单位进行链接压缩处理。

# §2.1.6 散列算法 - SHA1的整体结构

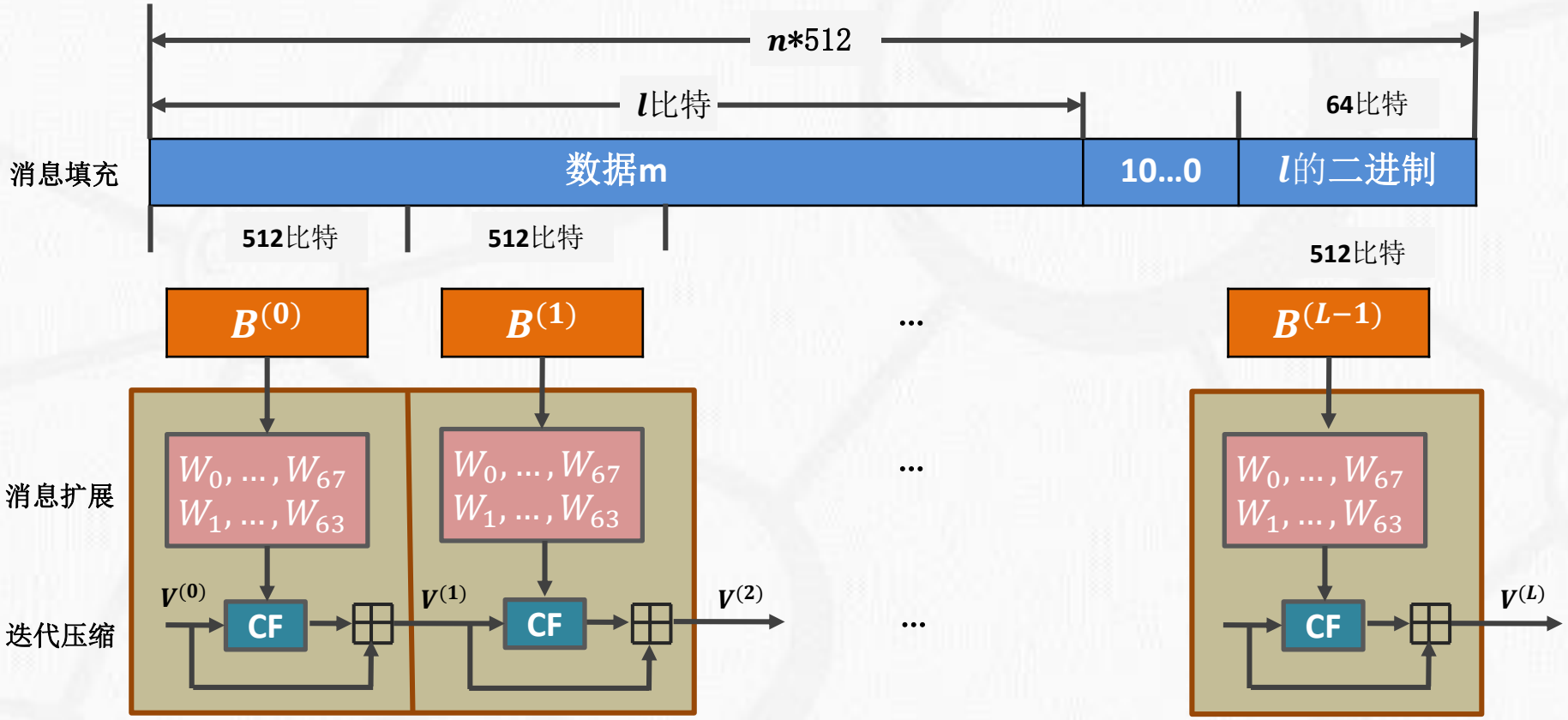


# §2.1.6 散列算法 - SM3的发展

- SM3散列函数
  - 适用于商用密码应用中的数字签名和验证、消息认证码的生成与验证以及随机数的生成。
  - 可满足多种密码应用的安全需求。
    - 在商用密码体系中，SM3主要用于数字签名及验证、消息认证码生成及验证、随机数生成等，其算法公开。据国家密码管理局表示，其安全性及效率与SHA-256相当。
  - 2010年12月17日,《SM3密码杂凑算法》由国家密码管理局于发布。我国第一个密码散列函数标准。

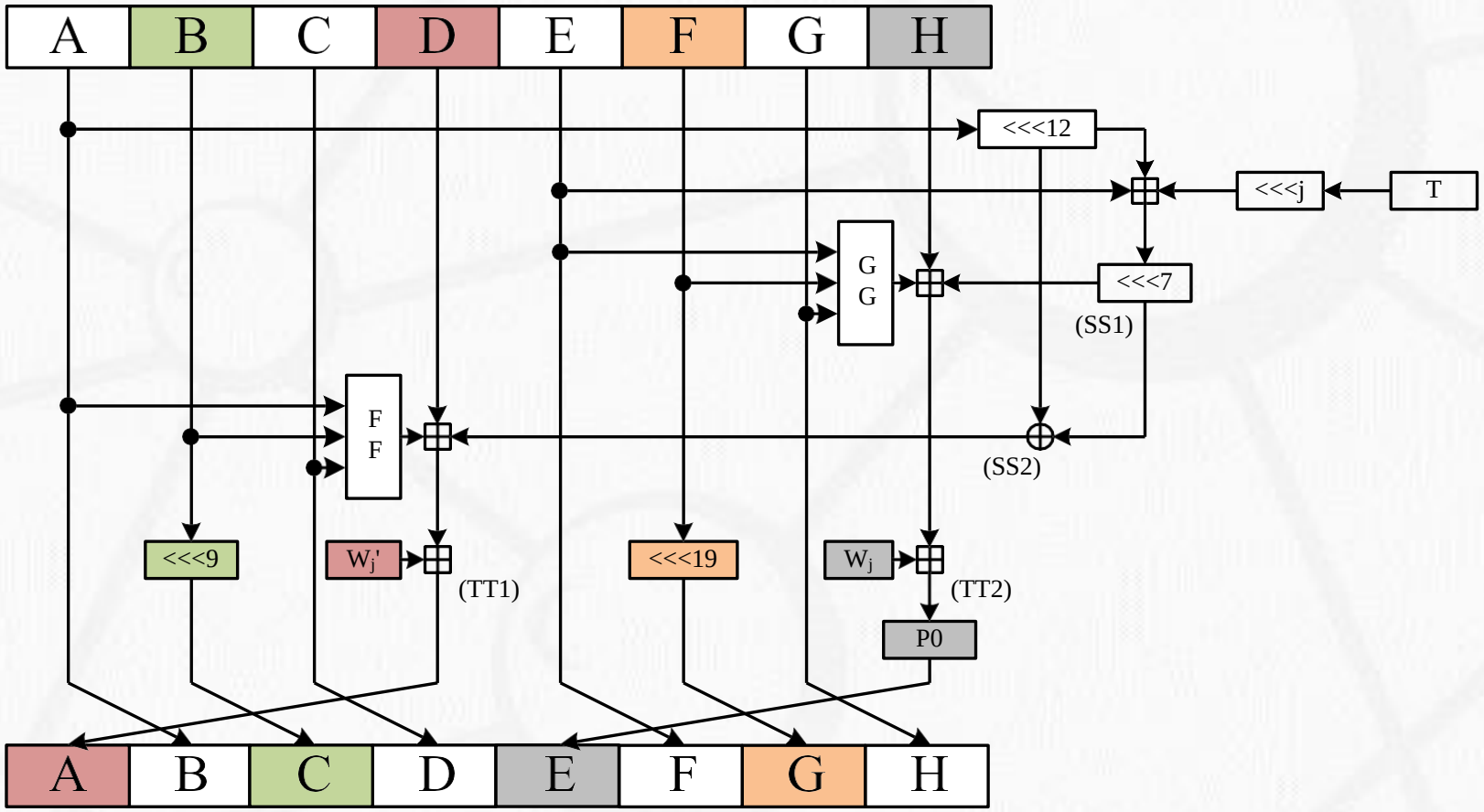
# §2.1.6 散列算法 - SM3的整体结构

- 基本框架：压缩函数+迭代结构

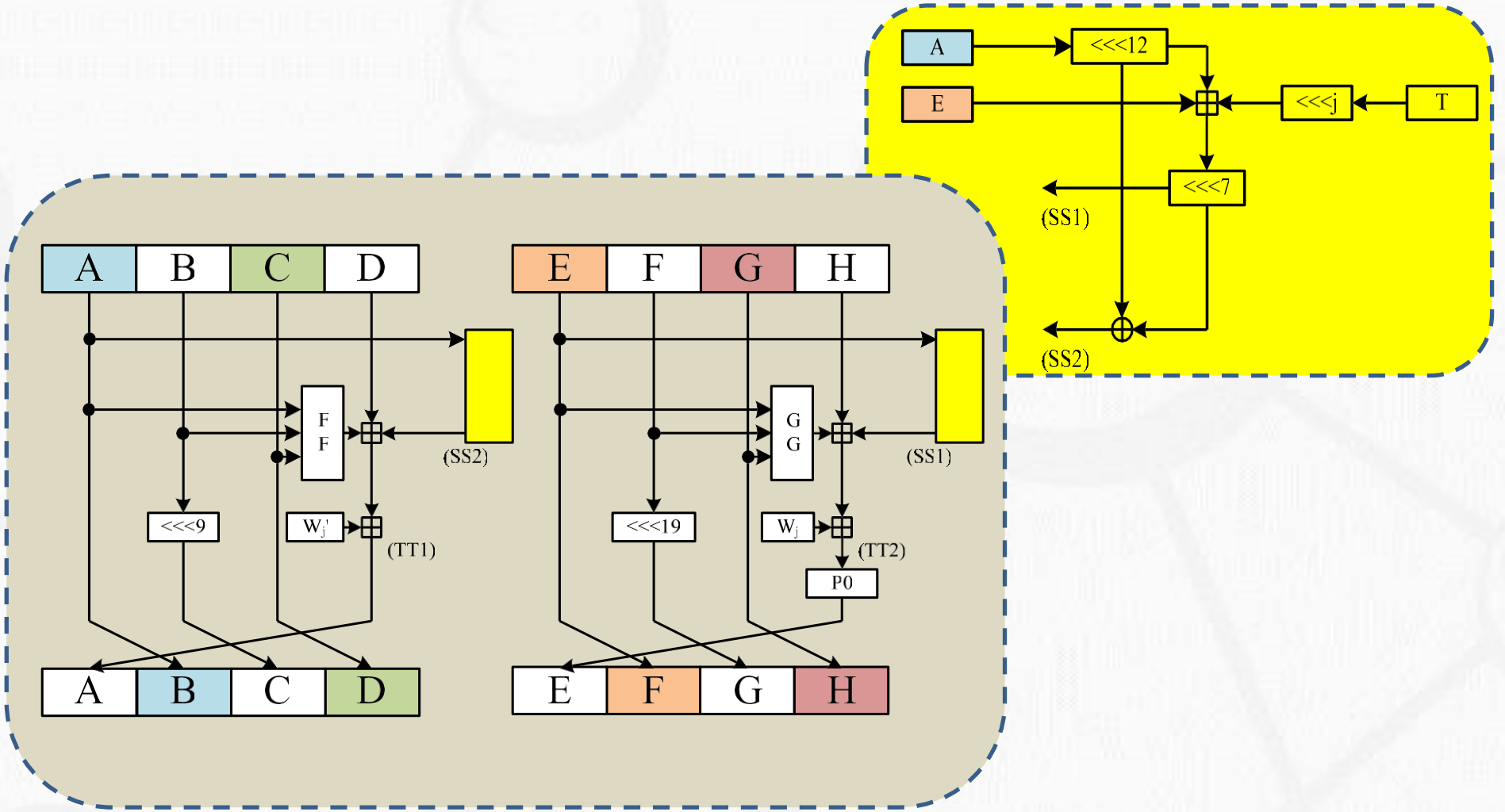


# §2.1.6 散列算法 - SM3的轮函数

- SM3的轮函数



# §2.1.6 散列算法 - 轮函数分解结构



# §2.1.6 散列算法 - 散列函数的破解

- 1966年生于山东诸城，1983年至1993年就读于山东大学数学系，先后获得学士、硕士和博士学位，1993年毕业后留校任教。
- 2005年6月受聘为清华大学高等研究中心“杨振宁讲座教授”，现为清华大学“长江学者特聘教授”。
- 2017年获评中国国科学院院士。

- 从1994年开始破解MD5和SHA-1的，到她2004年成功破解经过了10年。

# §2.1.6 散列算法 - 王氏发现艳惊四座、石破惊天

- 2004年8月，在美国加州圣巴巴拉召开的年度密码学国际会议上，王小云首次宣布了她和她的研究小组近年来的研究成果——对MD5、HAVAL-128、MD4和RIPEMD等四个著名密码算法的破译结果。
  - 在王小云教授仅公布到他们的第三个惊人成果的时候，会场上已经是掌声四起，报告不得不一度中断。
  - 报告结束时，与会者长时间热烈鼓掌，部分学者起立鼓掌致敬，这在密码学会议上是少见的盛况。

- 之后，王小云又继续发展她的研究，于2005年初破解了美国国家标准与技术研究所（NIST）为美国政府及商企界制定的SHA-1密码算法。

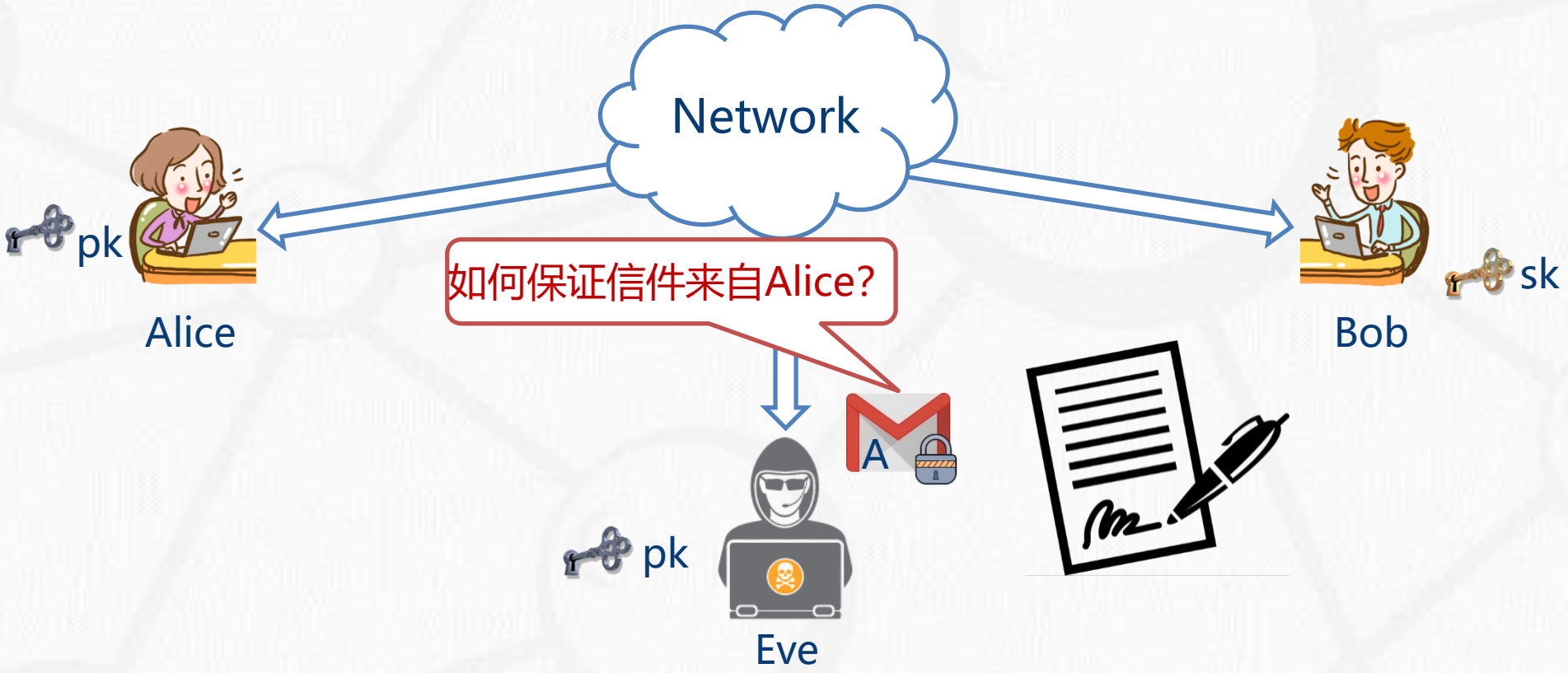

西安電子科技大學
XIDIAN UNIVERSITY

消息到底是谁写的、另一个角度审视非对称密码

# §2.1.7 数字签名

# §2.1.7 数字签名 - 应用背景

- Alice有一封邮件需要通过网络发送给Bob，包含公示信息。



# §2.1.7 数字签名 - 与传统签名的对比

- 传统签名的基本特点:
  - 能与被签物在物理上不可分割
  - 签名者不能否认自己的签名
  - 签名不能被伪造
  - 容易被验证


- 数字签名的基本要求:
  - 能与所签文件“绑定”
  - 签名者不能否认自己的签名
  - 签名不能被伪造
  - 容易被自动验证
  - 必须能够验证作者及其签名的日期时间
  - 必须能够认证签名时刻的内容
  - 签名必须能够由第三方验证，以解决争议

# §2.1.7 数字签名 - 公钥加密回顾

Alice
(发送者)

消息$m$

公钥$pk_B$

加密$E$

密文$c$

Bob
(接收者)

消息$m$

公钥$pk_B$

密文$c$

解密$D$

私钥$sk_B$

Eve
(攻击者)

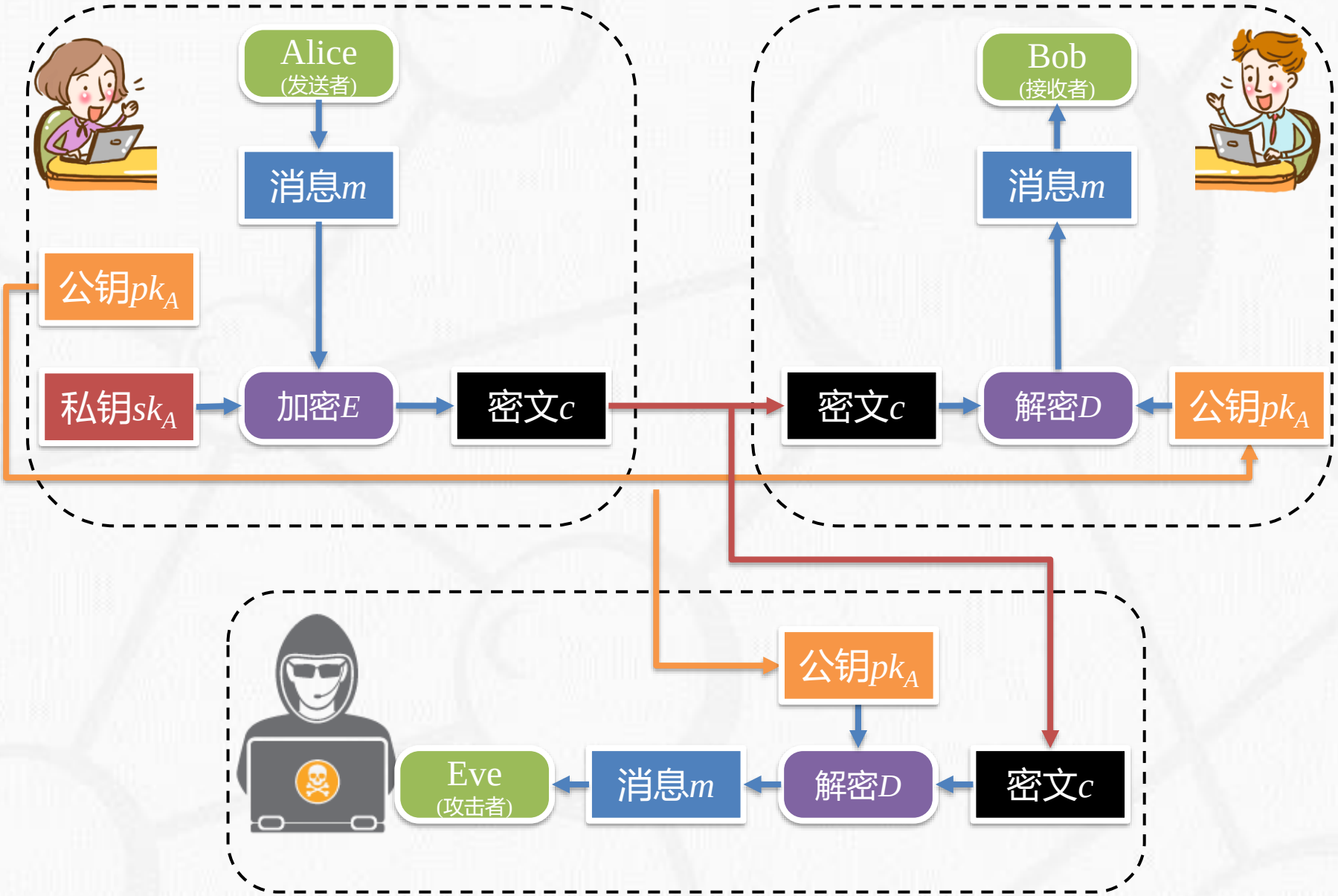
消息$m$

公钥$pk_B$

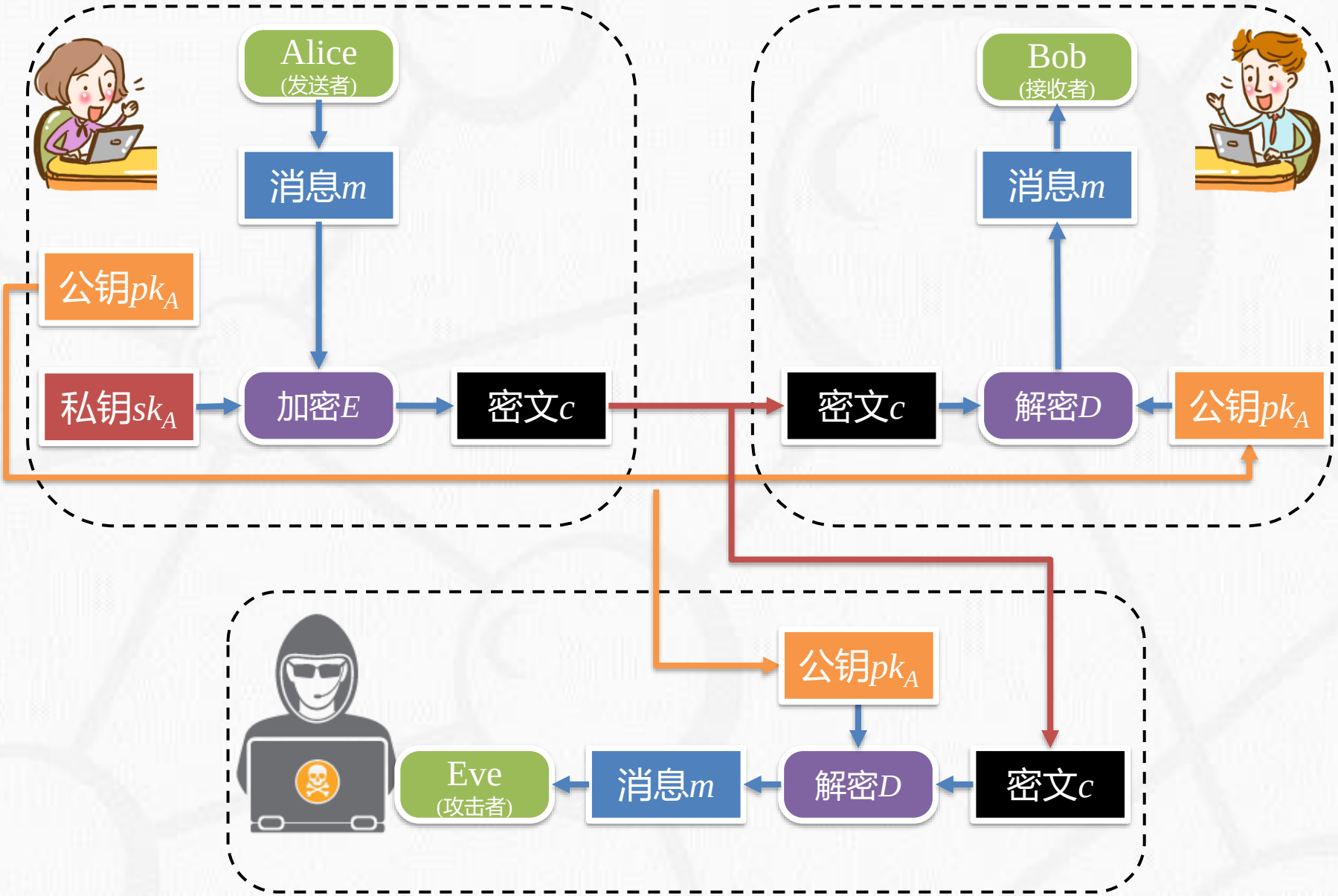
解密$D$

密文$c$

# §2.1.7 数字签名 - 基于公钥的签名

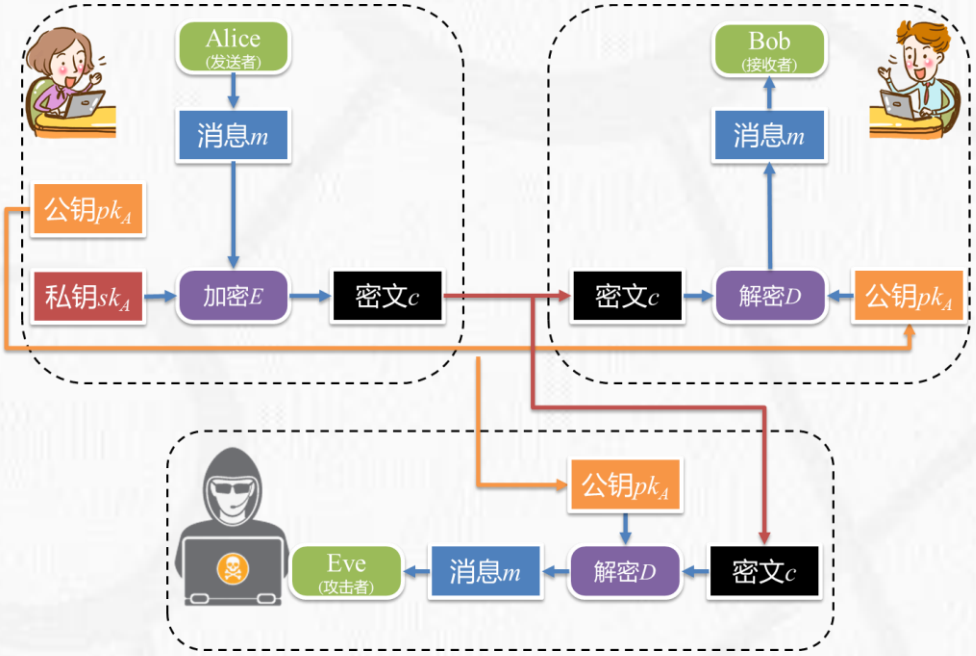


# §2.1.7 数字签名 - 基于公钥的签名



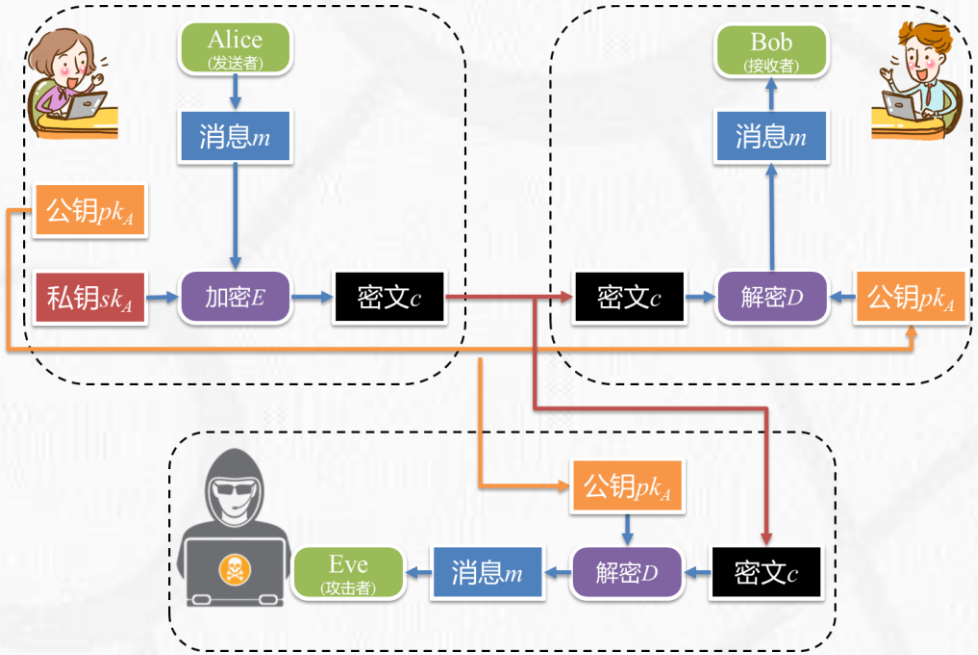
# §2.1.7 数字签名 - 基于公钥签名存在的问题?

- 基于公钥的签名存在什么问题?

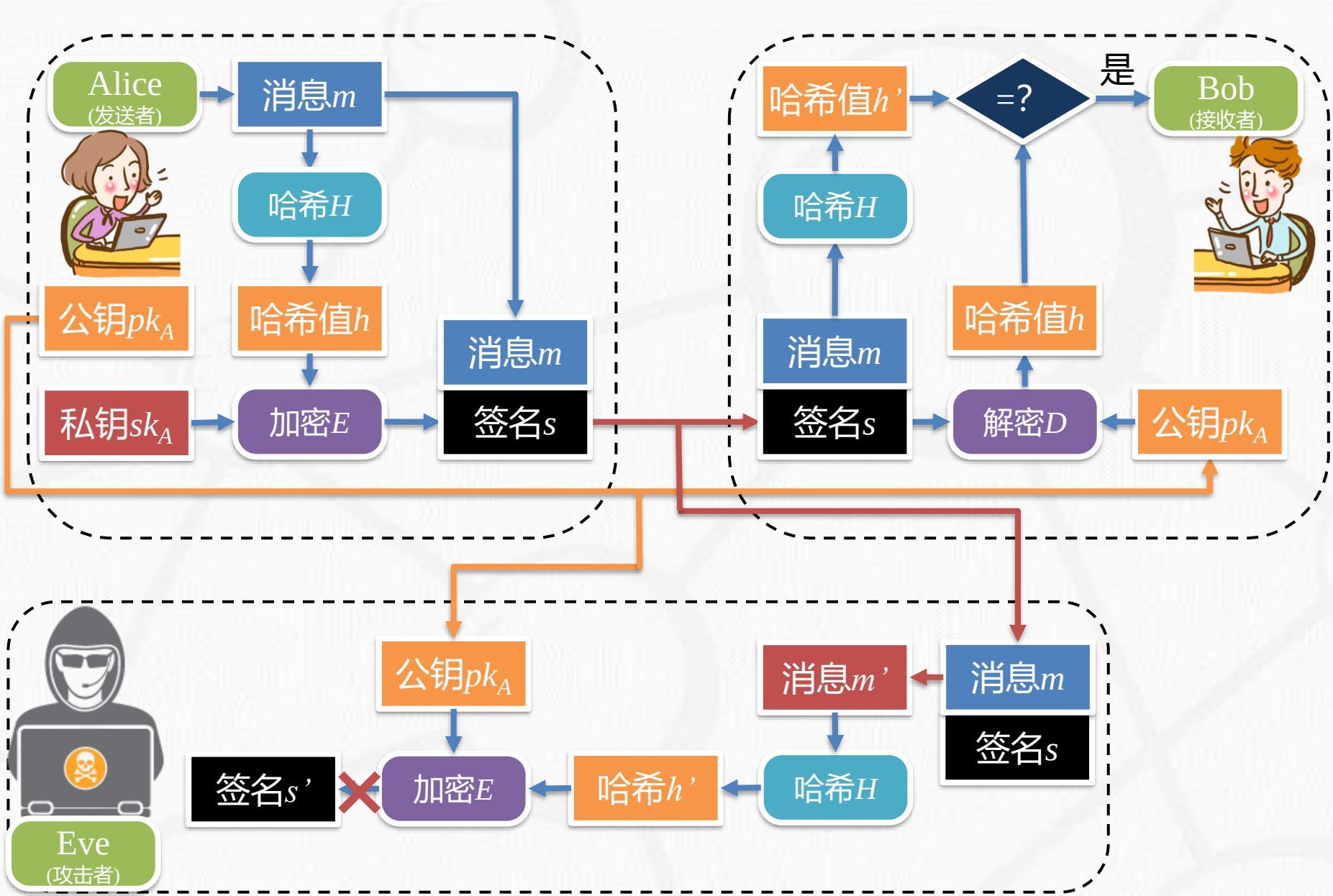


# §2.1.7 数字签名 - 基于公钥签名存在的问题?

- 基于公钥的签名存在什么问题?
  - 消息m的长度不确定，造成签名长度不确定。
    - 现实中的签名长度固定。

# §2.1.7 数字签名 - 引入散列函数的公钥签名



# §2.1.7 数字签名 - 消息不可见的签名方案

Alice
(发送者)

公钥$pk_B$

加密$E$

消息$m$

哈希$H$

公钥$pk_A$

哈希值$h$

私钥$sk_A$

加密$E$

密文$c$

签名$s$

哈希$H$

哈希值$h'$

公钥$pk_B$

消息$m$

=?

是

私钥$sk_B$

解密$D$

Bob
(接收者)

密文$c$

签名$s$

哈希值$h$

解密$D$

公钥$pk_A$

Eve
(攻击者)

公钥$pk_A$

签名$s'$

加密$E$

哈希$h'$

哈希$H$

消息$m'$

公钥$pk_B$

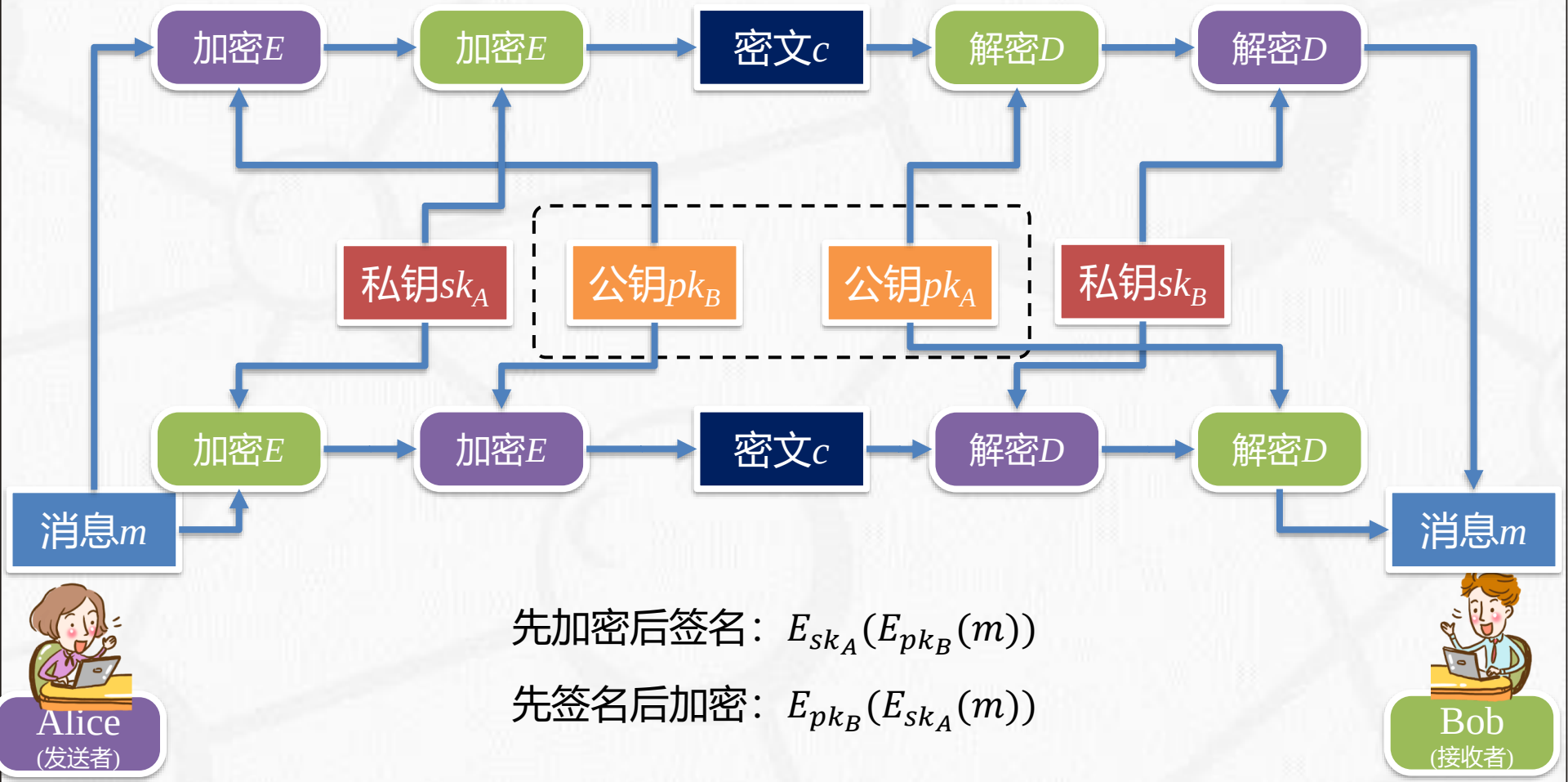
解密$D$

密文$c$

签名$s$

# §2.1.7 数字签名 - 思考(协议分析初探)

- 先加密后签名和先签名后加密，谁的安全性更好？




先加密后签名：$E_{sk_A}(E_{pk_B}(m))$

先签名后加密：$E_{pk_B}(E_{sk_A}(m))$

# §2.1.5 & §2.1.6 & §2.1.7 小结

- 内容回顾
  - 公钥密码算法的模型
  - RSA加密算法实例
  - 散列函数的基本构造
  - 数字签名的基本模型

- 掌握
  - RSA的计算原理
  - 散列函数的
  - 数字签名的基本流程
  - 协议分析初探(先签名后加密还是先加密后签名)

Thanks!
Questions & Advices!