

W1 1-1 Course Overview

1、Course objectives:

- 了解密码原语 (crypto primitives) 的工作原理
- 了解如何正确使用这些密码原语和其安全性原理 (会涉及一些定理证明)

目标：能够对密码构造的安全性进行推理，并且能够分析并破坏一些不安全的构造

2、Cryptography is everywhere

有计算机的地方密码学都有广泛应用，它是用来保护数据的一种常见工具

(1) 加密通信场景：

- web通信：HTTPS
- 无线通信：WPA2 (802.11i) , GSM机制 (移动通信) , 包括蓝牙也有加密

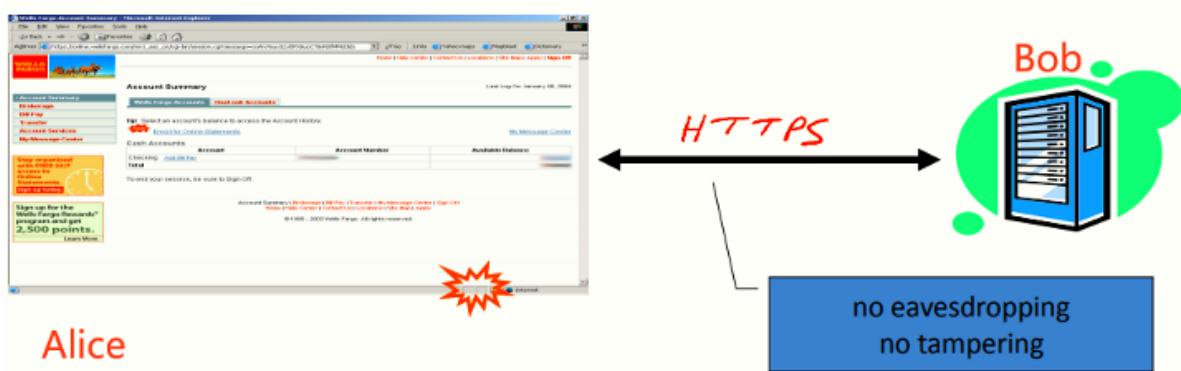
(2) 加密磁盘文件：如EFS, TrueCrypt等，即便磁盘丢失也不会导致文件泄露

(3) 内容保护：如DVD, 蓝光影碟等，DVD使用CSS (Content Scrambling System, 比较脆弱) , Blu-ray使用AACS

(4) 用户认证相关：

3、Secure Communication

典型通信模型：C/S模式



通信过程中的期望：no eavesdropping (不被窃听) and no tampering (不被篡改)

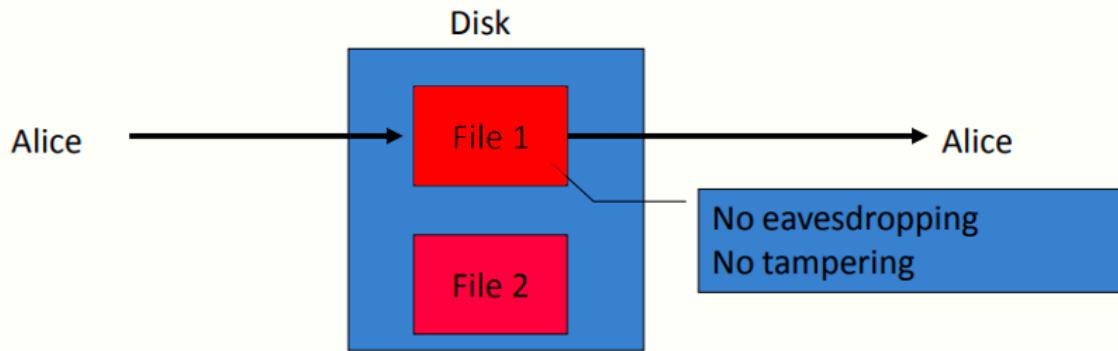
4、Secure Sockers Layer/TLS

两个主要组成部分：

- (1) 握手协议 (Handshake Protocol) : 以公钥密码技术的方式建立共享密钥K, 但攻击者对于该K毫不知情, Alice和Bob使用共享密钥K来进行安全通信
- (2) 记录子层 (Record Layer) : 表明当双方已经有了共享密钥, 他们要如何使用该密钥来加密并保护通信数据

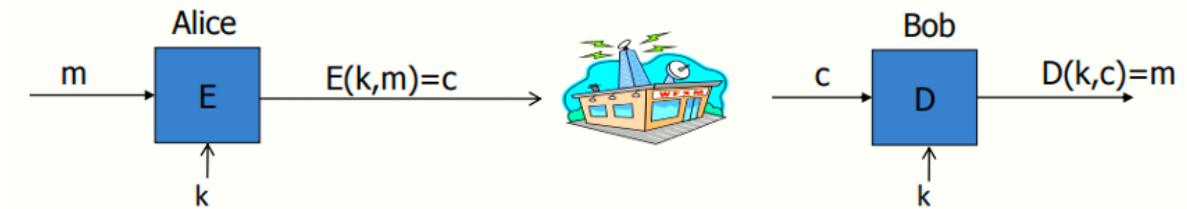
5、Protected files on disk

与安全通信类似，存储在磁盘上的文件也需要确保no eavesdropping and no tampering，若攻击者窃取了磁盘并修改了上面的数据，则Alice再次读取该磁盘时，Alice应当发现数据已被篡改并忽略这些内容



就本质而言，磁盘文件加密与安全通信是一样的，安全通信是确保通信双方的安全，而加密文件可以视为今天的Alice和明天的Alice通信

6、Building Block: symmetric encryption



m: plaintext, 明文

c: ciphertext, 密文

k: secret key, 密钥, 长度不定

E: Encryption algorithm, 加密算法, 接受明文和密钥输入, 输出对应的密文

D: Decryption algorithm, 解密算法, 接受密文和密钥输入, 输出正确的解密明文

对于对称密码技术而言，除了共享密钥外，其他一切均是公开的（包括使用的加密算法及其实现细节），绝对不要使用保密的加密方案，由于密码系统中其他一切内容已经被本领域内众多专家与学者（包括专业的密码破译者）研究分析了很多年，只有被认为是无法破解的算法才会投入使用，无论何时都要选择公开的标准算法来加密数据，反而私人设计的密码方案可能很容易被分析破解

7、Use Cases

(1) Single use key: one time key, 一次一密，每个密钥仅用于加密一条消息，如email，每次都是用不同的密钥加密，加密信息的方法效率很高，而且实现起来也很简单

(2) Multi use key: many time key, 多次密钥，密钥重复使用，如加密许多不同的文件时的密钥

8、Things to remember

密码学是：

- 极其强大的工具
- 很多安全机制的基石

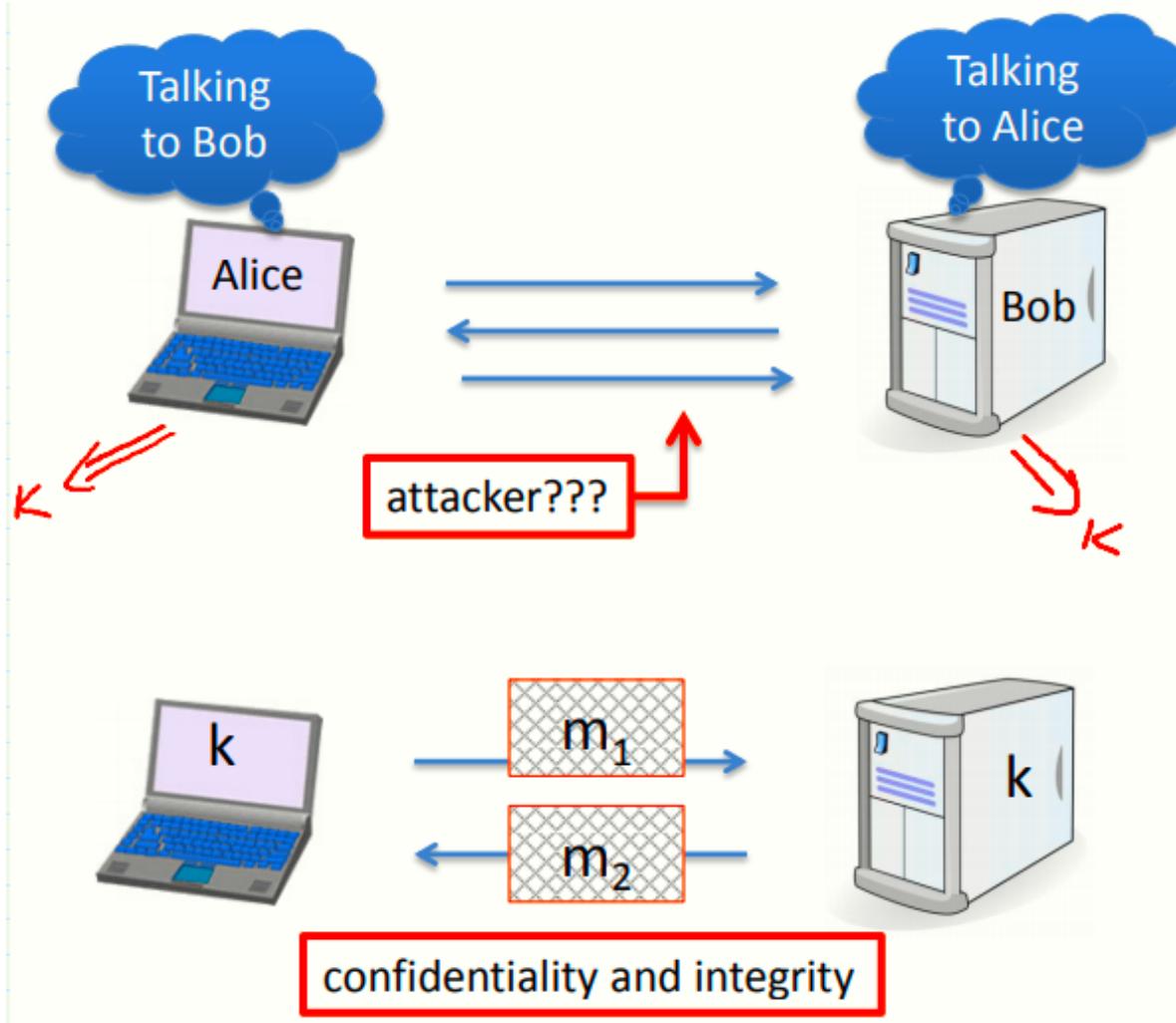
密码学的局限性：

- 并不是所有安全问题的解决方案（如软件bug, 社工等，此时cryptography也黔驴技穷）
- 只有正确的使用时才是可靠的

- Something you should try to invent yourself (?)

W1 1-2 What is Cryptography

1、Crypto core



(1) Secret key establishment: 在安全协议结束时，Alice和Bob能获得相同的共享密钥K，除此之外，Alice会知道她确实是在跟Bob说话，而Bob也确定和他对话的是Alice，对于攻击者而言，即便他能够窃听会话，也不能得知共享密钥

(2) Secure communication: 获得安全密钥后，他们就希望能通过密钥加密从而安全通信，而攻击者无法得知通信双方传送的消息，且攻击者若篡改了消息则一定会被发现，方案不仅提供了保密性 (confidentiality) 还提供了完整性 (integrity)

2、Crypto can do much more

(1) Digital signatures: 一种对现实世界签名的模拟，数字签名的工作方式基于内容的签名函数实现，即便是攻击者试图将签名者的签名从一个文档复制到另一文档，则在新的文档中由于内容不同而无法通过验证

(2) Anonymous communication: 用户与服务器通信时包含隐私数据时，此时用户希望匿名交谈而服务器不知晓他是谁，实现这一目的有标准方法mixnet，允许Alice通过一系列代理最终与Bob通信，而通信结束后Bob不知晓通信对方是谁

Mixnet基本工作方式：通过一些列的代理（proxy, proxies），使得传输的消息经过适当的加解密后最终发送到Bob，不仅Bob不知晓在与谁通信，甚至中间的代理服务器也不知晓通信双方的信息

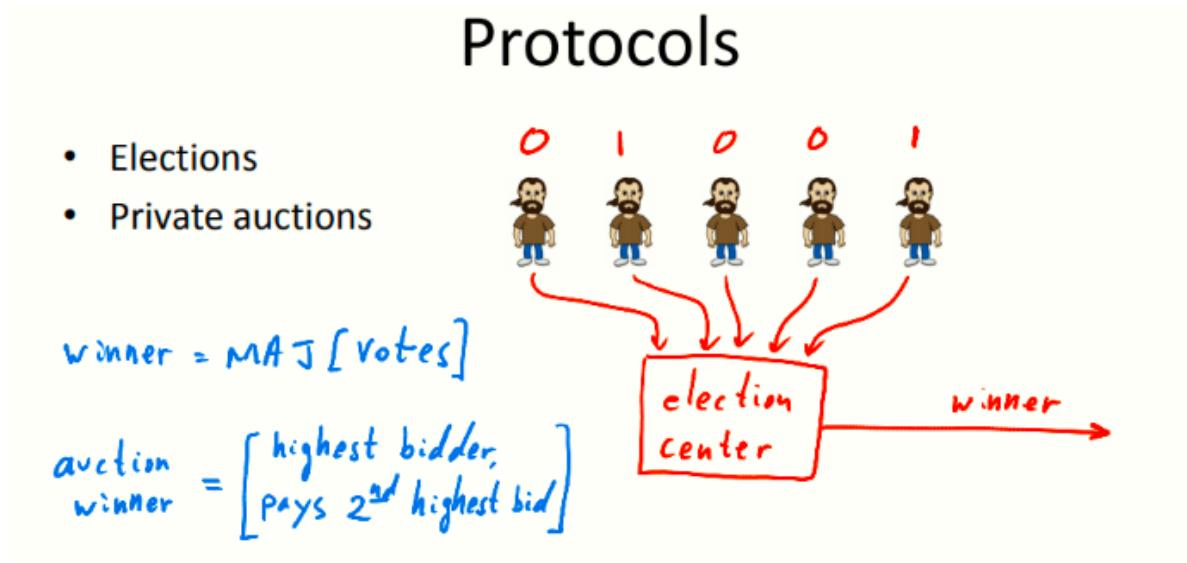
匿名通信实际上是双向通道，即便Bob不知晓通信对象是谁也能为其响应，且Alice能收到这些响应

简单的例子：匿名电子现金，确保Alice的钱使用且仅使用一次，若使用超过一次则会被警察带走（具体实现后续讨论）

3、Protocol

密码学的另一些应用，即与一些抽象的协议结合

(1) Elections: 票数加密后送入精选中心，竞选结束后中心宣布获胜方而不泄露其他信息（票数仍保密），且中心需要对选民进行验证（是否仅投一次票，是否有投票权等）



(2) Private auctions: 上述模型也常用于私人拍卖会上，如Vickrey auction，竞标最高者获胜但仅需要支付第二高竞标者的出价

计算结束时，除了得知结果之外（winner和payment），其他一切信息诸如竞标者的个人信息，最高竞标者的出价都应保密，唯独需要公开第二高出价金额和最高竞标者的身份

(3) Secure multi-party computation: 多方安全计算

- Elections
- Private auctions

Goal: compute $f(x_1, x_2, x_3, x_4)$

"Thm:" anything that can be done with trusted auth. can also be done without

- Secure multi-party computation

模型1：引入一个可信组织，收集所有个人输出，并且承诺对所有输入保密，仅有可信组织知道，该组织根据输入计算结果并公开，且不公开除结果之外的其他个人信息，但是这个组织搞名堂那就没用了

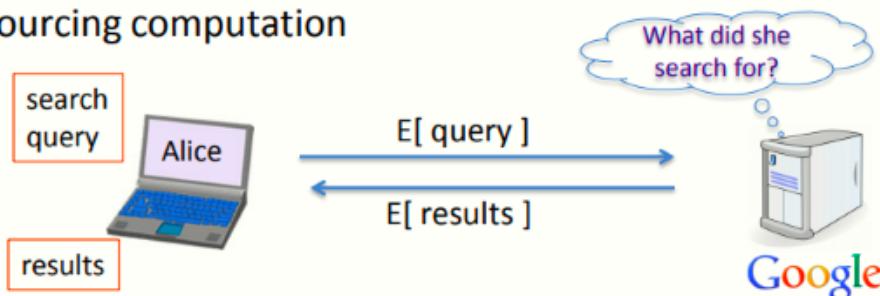
- 惊讶的事实：如果你能把一个计算交给可信组织完成，则一定可以不需要可信组织来得到结果，即我们可以使任何事情都不需要可信组织插手就能得到结果，仅需要参与者使用某种协议与其他参与者交换信息，协议结束后每个人都江之岛最终结果是什么而对计算过程一无所知（即个体的输入仍然是保密的，但没有可信的权威组织介入），看上去难以置信但确实可行

4、Crypto magic

一些难以归为密码学的应用，但确实很神奇

(1) Privately outsourcing computation: 私人外包计算，比如Alice使用特殊的加密方案使得它可以发送加密的查询给google，然后基于这种加密特性，使得google可以计算搜索而无需知道原始文本，并将计算结果以加密方式返回给Alice，Alice解密后得到查询结果

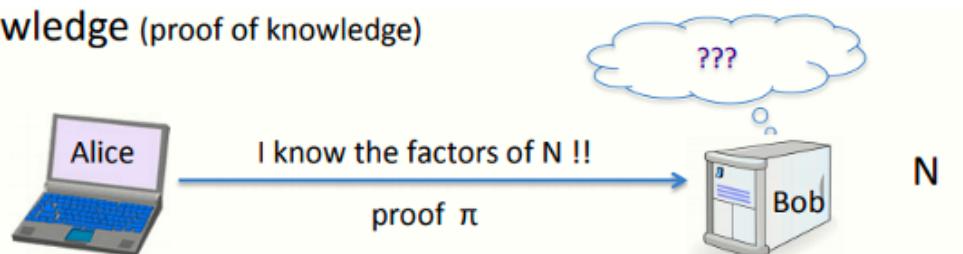
- Privately outsourcing computation



google全程仅知道加密后的查询内容，并不知道Alice想搜索什么和搜索结果是什么

(2) Zero knowledge (proof of knowledge)

- Zero knowledge (proof of knowledge)



零知识证明，比如Alice有一大整数N，且知道其因子p和q，Bob仅知道N，通过零知识证明Alice可以向Bob证明他知道N的分解而不向Bob透露因子

5、A rigorous science

我们描述的每个概念都将严格的按照三个步骤，我们将反反复复地遵循这三个步骤

- (1) Precisely specify threat model: 对于每个原语，我们要精确定义威胁模型是什么
- (2) Propose a construction: 接着提出一种构造方案
- (3) Prove that breaking construction under threat mode will solve an underlying hard problem: 再证明在特定威胁模型条件下，任何攻击者都能攻击这个构造

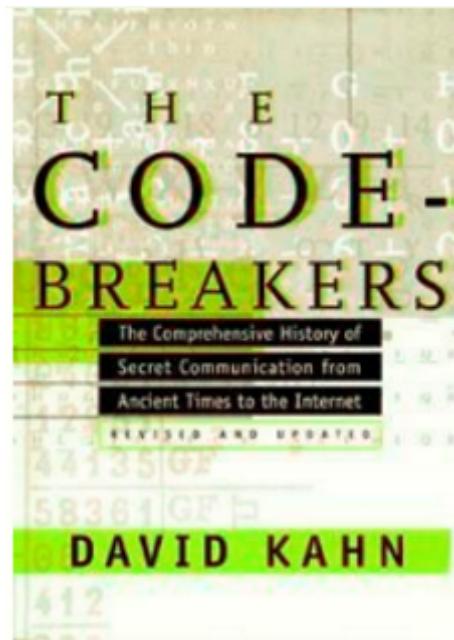
一旦攻击者成功进行攻击，则攻击者也可以解决相对应的一些复杂难题，如果难题足够复杂以至于无法破解，则证明在这个模型下没有攻击者可以破解

如整数分解被普遍认为是个困难问题，因此.....

W1 1-3 History of Cryptography

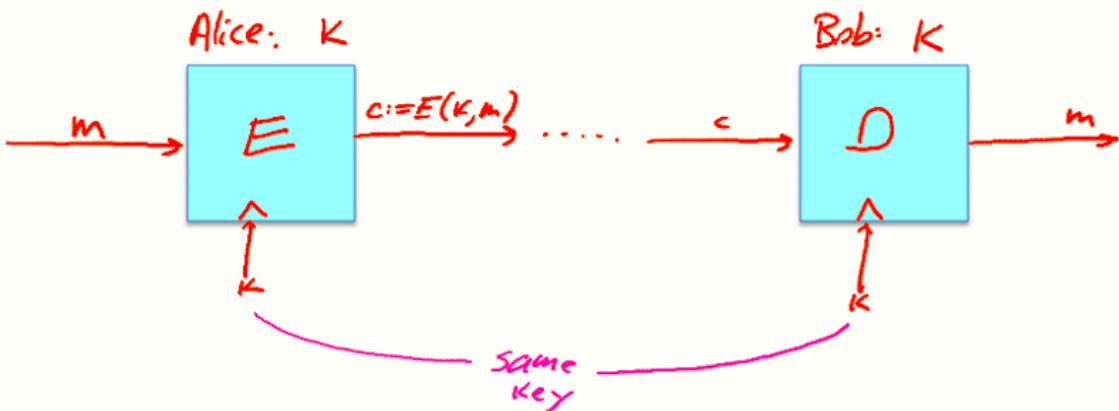
1、History

参考读物：David Kahn “The code breakers” 1996



2、Symmetric Ciphers

对于对称密码体系，通信双方使用相同的密钥加解密



3、Few history examples

对于现代技术而言，均已不适合实际使用

- (1) Substitution cipher: 替换密码，将26个字母按照一定规则替换为别的字母，映射唯一
- (2) Caesar Cipher (no key) : shift by 3 $(\text{mod } 26)$
- (3) 对于置换密码而言密钥空间有 $26!$ 这么多，约等于 2^{88}

4、How to break a substitution cipher

- (1) Use frequency of English letters: 英文中频率最高的字母为e, t和a次之
- (2) Use frequency of pairs of letters (digrams): 英文中最常出现的两个连续字母为he、an、in、th等，极易受到唯密文攻击 (CipherText only attack)

UKBYBIPOUZBCUFEEBORUKBYBHOBBRFESPVKBWFOFERVNBCVBZPRUBOFEVNBCVBPCYYFVUFO
 FEIKNWFRFIKJNUPWRFIPOUNVNIPUBRNCUKBEFWFDFDNCHXYBOHOPYXPUBNCUBOYNRVNIWN
 CPOJIOFHOPZRVFZIXUBORJRUBZRBCHNCBBONCHRJZSFVNVRJRUBZRPCYZPUKBZPUNVPWPCYVF
 ZIXUPUNFCPWRVNBCVBRPYNNUNFCPWWJUKBYBIPOUZBCUIPOUNVNIPUBRNCHOPYXPUBNCUB
 OYNRVNIWCPOJIOFHOPZRNCRVNBCUNENVVFZIXUNCHPCYVFZIXUPUNFCPWFZPKBZPUNVR

B	36
N	34
U	33
P	32
C	26

→ E

NC	11
PU	10
UB	10
UN	9

→ IN
→ AT

UKB	6
RVN	6
FZI	4

→ THE

digrams trigrams

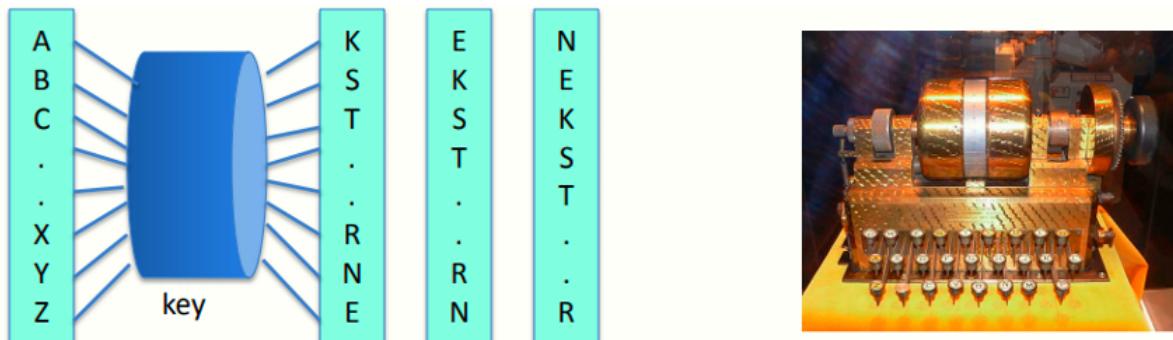
5、Vigener cipher (16'th century, Rome)

使用一长度固定的密钥，每次加密与密钥长度相同的明文，以mod 26加法方式加解密

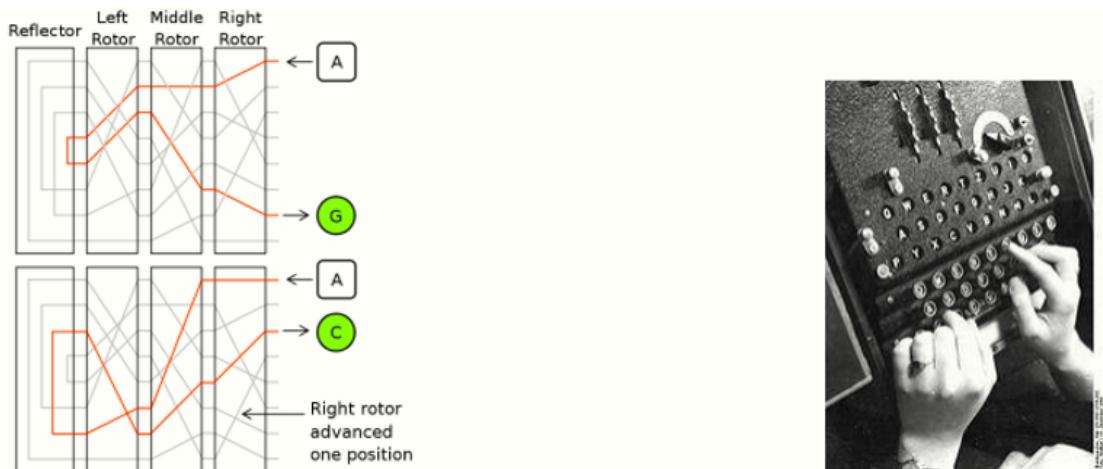
仍然不能抵御频率分析，比如六位长的密钥，将密文分组后，分别统计各个位上的字母频率，密文数量足够大时仍可以分析出密钥

6、Rotor Machines (1870-1943)

转轮机的早期例子：the Hebern machine (single rotor)



最著名：Enigma (3-5 rotors)，四个转轮的Enigma密钥空间高达 $26^4 \approx 2^{18}$



$$\# \text{ keys} = 26^4 = 2^{18} \quad (\text{actually } 2^{36} \text{ due to plugboard})$$

7、Data Encryption Standard (1974)

DES: 密钥空间为 2^{56} , 分块大小64 bits

现在最常用: AES (2001) , Salsa20 (2008) 等

W1 1-4,5 Discrete Probability

1、一些概念

U : finite set (e.g. $U = \{0,1\}^n$)

Def: Probability distribution P over U is a function $P: U \rightarrow [0,1]$ such that $\sum P(x) = 1$

Uniform distribution: for all $x \in U$: $P(x) = 1/|U|$

Point distribution at x_0 : $P(x_0) = 1$, $\forall x \neq x_0$: $P(x) = 0$

Distribution vector: ($P(000), P(001), P(010), \dots, P(111)$)

2、Events

For a set $A \subseteq U$: $\Pr[A] = \sum P(x) \in [0,1]$ ($\Pr[U]=1$)

The set A is called an event

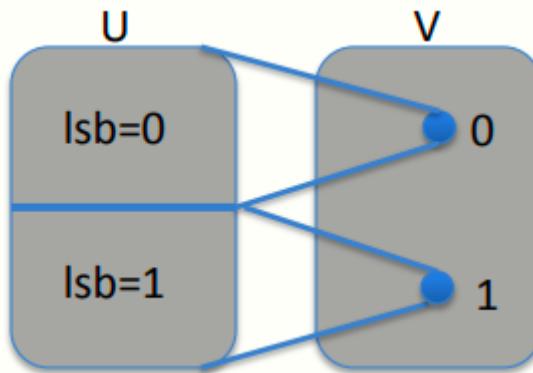
3、The union bound

For events A_1 and A_2 , $\Pr[A_1 \cup A_2] \leq \Pr[A_1] + \Pr[A_2]$

不等式恒成立, 当事件A1和A2相互独立时取等号

4、Random Variables

Def: a random variable X is a function $X: U \rightarrow V$ Example: $X: \{0,1\}^n \rightarrow \{0,1\}$; $X(y) = \text{lsb}(y) \in \{0,1\}$



More generally: rand. var. X induces a distribution on V : $\Pr[X=v] := \Pr[X^{-1}(v)]$

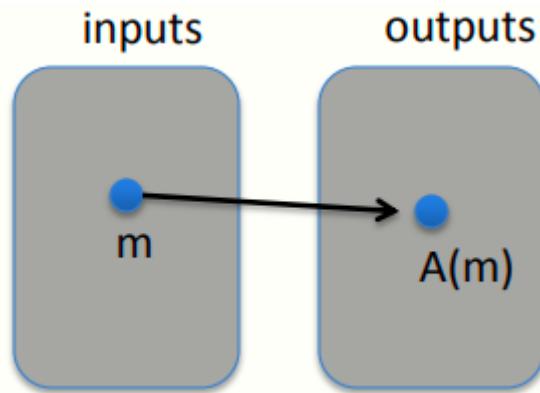
5、The uniform random variable

Let U be some set, e.g. $U = \{0,1\}^n$

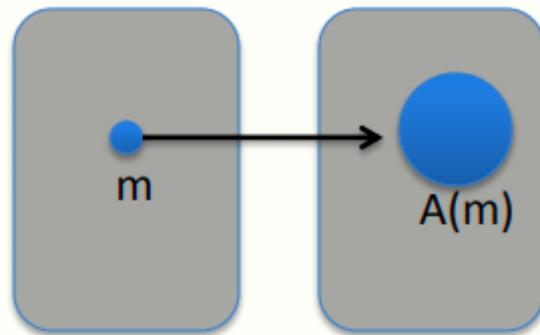
We write $r \leftarrow U$ to denote a uniform random variable over U for all $a \in U$: $\Pr[r = a] = 1/|U|$

6、Randomized algorithms

Deterministic algorithm: $y \leftarrow A(m)$, 对于每次相同的输入, 确定性算法总能得到相同的输出



Randomized algorithm $y \leftarrow A(m; r)$ where $r \leftarrow \{0,1\}^n$, output is a random variable, 随机化算法每一次输入往往得到不同的输出



7、Independence

Def: events A and B are independent if $\Pr[A \text{ and } B] = \Pr[A] \cdot \Pr[B]$

random variables X, Y taking values in V are independent if $\forall a, b \in V: \Pr[X=a \text{ and } Y=b] = \Pr[X=a] \cdot \Pr[Y=b]$

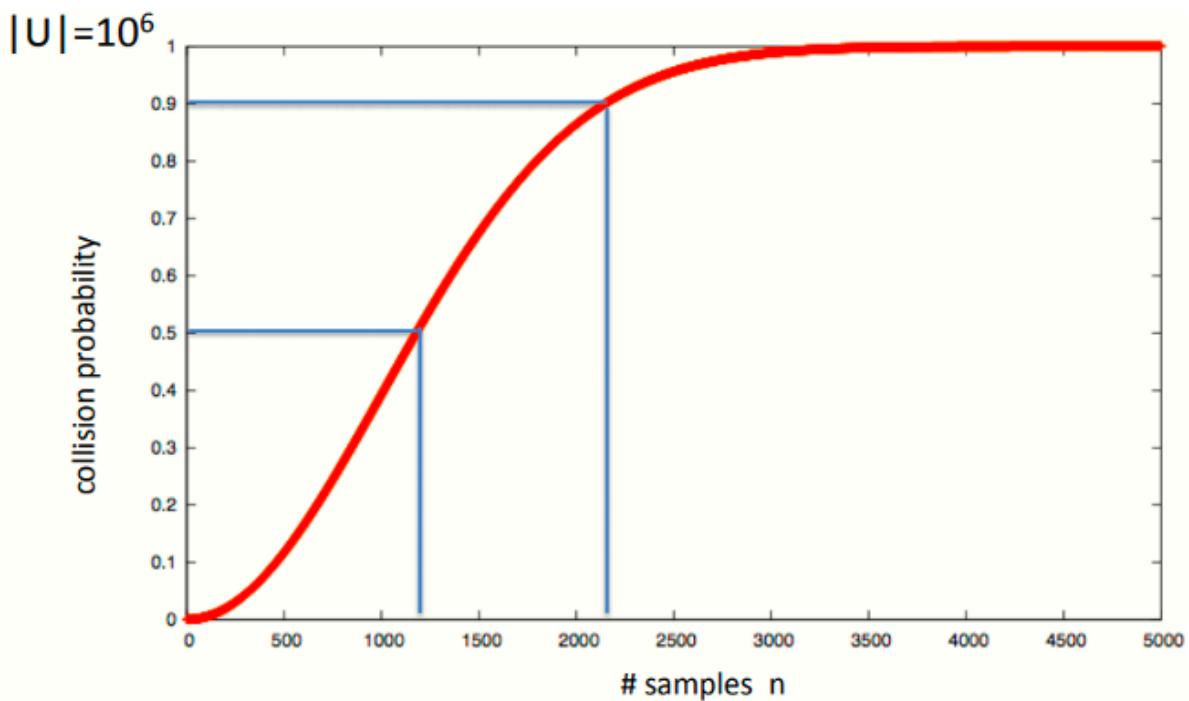
8、XOR

XOR of two strings in $\{0,1\}^n$ is their bit-wise addition mod 2

Thm: Y a rand. var. over $\{0,1\}^n$, X an indep. uniform var. on $\{0,1\}^n$. Then $Z := Y \oplus X$ is uniform var. on $\{0,1\}^n$

9、The birthday paradox

Let $r_1, \dots, r_n \in U$ be indep. identically distributed random vars. Thm: when $n = 1.2 \times |U|^{(1/2)}$ then $\Pr[\exists i \neq j: r_i = r_j] \geq 1/2$



W1 2-1 Information Theoretic Security and The One Time Pad

1、Symmetric Ciphers: definition

Def: a cipher defined over is a pair of “efficient” algs (E, D) where

$$E: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C} \quad , \quad D: \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$$

$$\text{s.t. } \forall m \in \mathcal{M}, k \in \mathcal{K}: \quad D(k, E(k, m)) = m$$

2、The One Time Pad (Vernam 1917)

A secure cipher: key = (random bit string as long the message)

You are given a message m and its One Time Pad encryption c , you can compute the OPT key from m and c that $k=m \oplus c$

OTP效率非常高，但是密钥长度会和明文一样长，即通信双方需要传输相当于明文一样长度的密钥，极大降低了通信效率

so, is the OTP secure, what is a secure cipher?

3、What is a secure cipher?

攻击者的手段: CT only attack (for now)

可能的安全需求:

- 攻击者不能恢复密钥, 即 $E(k, m)=m$ would be secure
- 攻击者不能恢复全部的明文, 即 $E(k, m_0 || m_1)=m_0 || k \oplus m_1$ would be secure

4、Information Theoretic Security (Shannon 1949)

香农：第一位严格研究密码算法安全性的专家

Def: A cipher (E, D) over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ has perfect secrecy if

$$\forall m_0, m_1 \in \mathcal{M} \quad (\text{len}(m_0) = \text{len}(m_1)) \quad \text{and} \quad \forall c \in \mathcal{C}$$

$$\Pr[E(k, m_0) = c] = \Pr[E(k, m_1) = c]$$

where k is uniform in \mathcal{K} ($k \leftarrow \mathcal{K}$)

这意味着：

- 对于给定的密文，不能得知其是由 m_0 还是 m_1 加密而来的，且对所有消息均成立 (for all m_0, m_1)
- 如果使用OTP，无论多么强大且聪明的敌手，都不能根据密文得知任何关于明文的消息

对于一个拥有完美安全性的加密算法，不仅仅只有唯密文攻击，还包含其他攻击手段 (other attacks maybe possible)

Lemma: OTP has perfect secrecy

Proof:

5、The bad news

香农给出了证明：若一个算法有完美安全性，则密钥总数必须大于等于消息空间中消息数目的总数，即完美安全性的算法的密钥长度必须大于等于消息长度

OTP密钥长度恰好等于消息长度，因此是一个最佳的方案，但是由于其冗长的密钥导致并不实用

W1 2-2 Stream Ciphers and Pseudo Random Generators

1、Review

Cipher over (K, M, C) : a pair of "efficient" algs (E, D) s.t. $\forall m \in M, k \in K: D(k, E(k, m)) = m$

Weak ciphers: subs. cipher, Vigener, ... (由于历史原因不应再使用)

A good cipher: OTP $M=C=K=\{0,1\}^n$ $E(k, m) = k \oplus m$, $D(k, c) = k \oplus c$

Lemma: OTP has perfect secrecy (i.e. no CT only attacks), OTP好用但不实用

Bad news: perfect-secrecy \Rightarrow key-len \geq msg-len

2、Stream Ciphers: making OTP practical

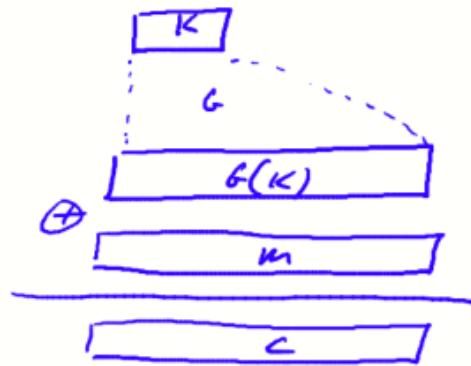
思路：以伪随机密钥来代替随机密钥 (replace "random" key by "pseudorandom" key)

PRG是一个函数，接受一个 s bit 的种子 (seed) 作为输入，并产生 n bit 的输出，且 n 远大于 s

对于 G 而言，必须能高效计算并产生输出，但其并不是真正的随机，仅有输入的种子是随机的

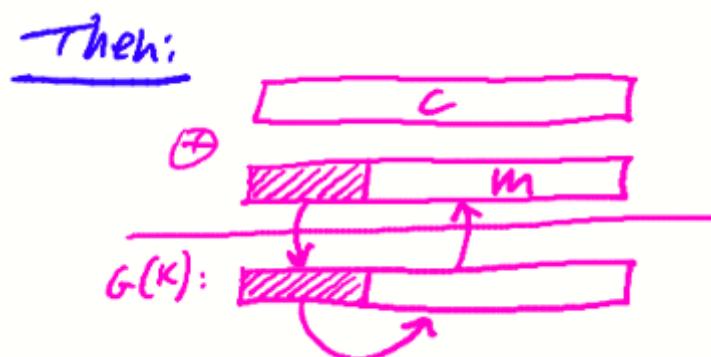
$$C := E(K, m) = m \oplus G(K)$$

$$D(K, C) = C \oplus G(K)$$



3、PRG must be unpredictable

假设PRG是可预测的，则任给i，对于G(k)的输出，存在某个算法，使得可由G(k)的前l bit计算出第i+1~n bit，则此时的流密码并不安全



如图所示，若攻击者知道了某些有关于明文的信息（如SMTP协议中常见的前缀消息等），攻击者可以将其与密文XOR计算后得知密钥，再加上上述所假设的PRG的可预测性，攻击者可以通过已知的前若干bit密钥来推测完整的密钥，从而恢复完整消息

We say that $G: K \rightarrow \{0,1\}^n$ is **predictable** if:

\exists "eff" alg. A and $\exists 0 \leq i \leq n-1$ s.t.

$$\Pr_{\substack{K \leftarrow \mathcal{K} \\ G}} \left[A(G(K)) \Big|_{i, \dots, i} = G(K) \Big|_{i+1} \right] > \frac{1}{2} + \epsilon$$

For non-negligible ϵ (e.g. $\epsilon = 1/2^{30}$)

Def: PRG is unpredictable if it is not predictable $\Rightarrow \forall i$: no "eff" adv. can predict bit $(i+1)$ for "non-neg" ϵ

4、Weak PRGs (donot use for crypto)

线性同余生成器 (Linear congruential generator) : 接收三个参数, a , b , p

a , b 为整数, p 为素数, 种子seed $\equiv r[0]$, $r[i] \leftarrow a \cdot r[i-1] + b \pmod p$, 输出 $r[i]$ 后, $i++$

glibc random():

$$\begin{aligned} r[i] &\leftarrow (r[i-3] + r[i-31]) \% 2^{32} \\ \text{output } r[i] &>> 1 \end{aligned}$$

尽管线性同余法得到的输出有很好的统计特性，但是由于其可预测性，仅需要很少的位数即可预测剩余的位，因此不应当再使用

W1 2-3 Attacks on Stream Ciphers and The One Time Pad

1、Review

OTP：使得OTP变实用的方法：PRG

stream cipher: $E(k, m) = m \oplus G(k)$, $D(k, c) = c \oplus G(k)$

安全性：PRG必须是不可预测的

2、Attack 1: two time pad is insecure

$$C_1 \leftarrow m_1 \oplus \text{PRG}(k)$$

$$C_2 \leftarrow m_2 \oplus \text{PRG}(k)$$

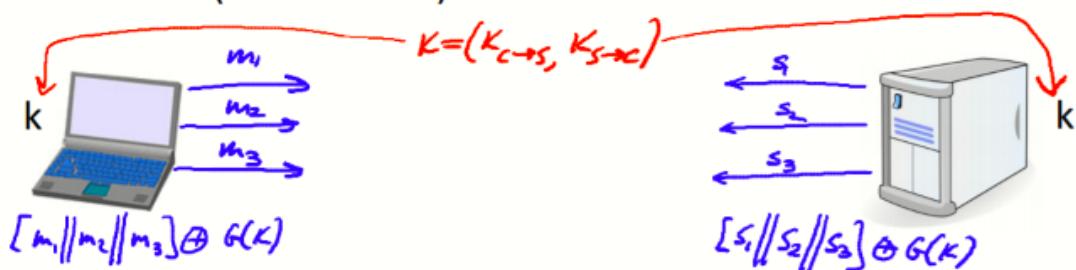
对于使用同一个 $\text{PRG}(k)$ 加密的两个消息，攻击者可以很容易的得到 $C_1 \oplus C_2 = m_1 \oplus m_2$ ，由于英文存在大量冗余，足以让攻击者从两段明文消息的异或值还原出分别还原出两段消息，若使用ASCII编码，由于ASCII编码也存在冗余，也可以还原出消息

因此攻击者用同一密码本对多段消息拦截密文，最终可以很轻易地还原明文，因此一次性密码本绝不可以使用超过一次

现实世界的失败案例：

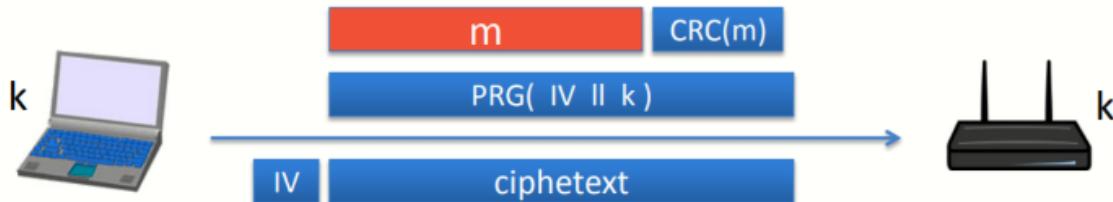
- Project Venona

- MS-PPTP (windows NT):



- Venona计划的密码本通过掷骰子来产生但是每次都掷骰子太麻烦因此使用同一个密码本加密了大量消息，最后惨遭破解
- MS-PPTP (windows NT)：若客户端和服务端均使用同一个PRG(k)，则导致了两次密码本，因此对于上述通信模型， $S \rightarrow C$ 和 $C \rightarrow S$ 需要不同的PRG(k)

802.11b WEP:



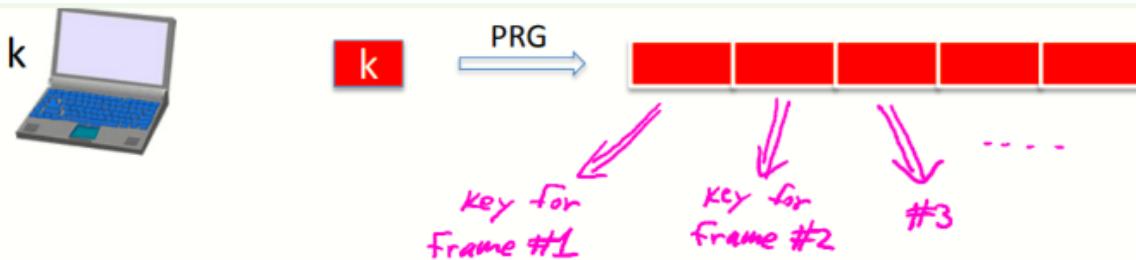
- 802.11b WEP：在WEP中，初始向量IV为24 bits，加密模型如上图所示，每次通信时协议都会将IV自增后作为PRG的一部分参数，由于IV长度受限，在 2^{24} (约为16M) 个消息后会回到最初的IV，若继续使用则会导致两次密码本

部分802.11的芯片在断电后重新加电，会使得IV直接从0开始并用于下一个消息载荷的加密

对于PRG的输入 $IV \parallel k$ ，IV总是在一定范围内重复（可能由于断电后立即重复，或在达到最大时重复），且IV处理过程并不是随机的，仅仅是简单的每次递增1，非常容易由一个IV推测出后续消息使用的IV，且密钥 k 总是以固定的104 bits种子作为PRG的输入的后缀出现，综上，攻击者可以很轻易的得到两次PRG的关系从而破解算法

总之，WEP很容易导致两次密码本（F、M、S三个人在2001年通过1M个消息即可恢复出明文，现在仅需要大约40k个消息即可攻破WEP）

3、A better construction



使用 k 作为PRG的种子生成一个足够长的随机数作为密钥，对于每个消息帧，使用这个密钥的不同部分，从而使得每个消息帧都有独立的密钥，来防止上述WEP中导致的重复密钥

4、Another example: disk encryption

对于磁盘文件的加密，由于磁盘的特性，文件会保存在磁盘上不同的块中，起初攻击者并不知晓加密的内容，某次用户修改了文件，且仅修改了文件的一小部分，若用户使用的加密算法不妥当，磁盘上仅有被修改的那一块发生了变化，则会导致安全问题

即便是攻击者不知道修改前或修改后的内容，也不知道修改了什么，但攻击者知道这一块发生了修改，而其他部分均是一致的，从而产生信息泄露

理想情况下。即便是文件仅修改了1 bit，也应导致整个文件的完全修改，至少应把整个块修改

5、Two time pad: summary

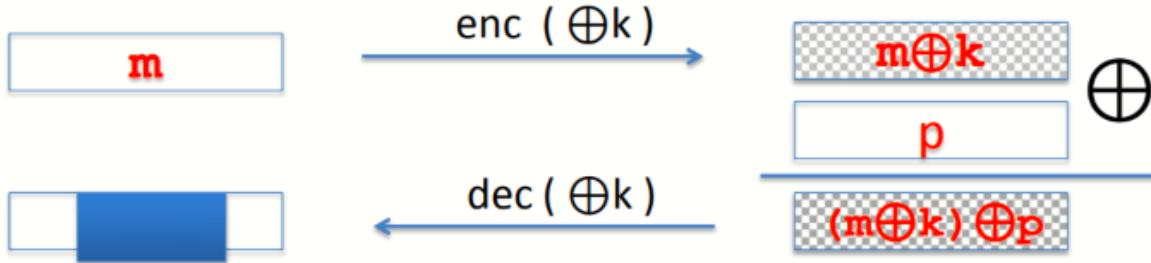
永远不要在流密码中多次使用同一个密钥

- 对于网络通信：每个新的会话（session）可以协商新的密钥（如TLS）
- 对于磁盘加密：一般来说磁盘加密不使用流密码

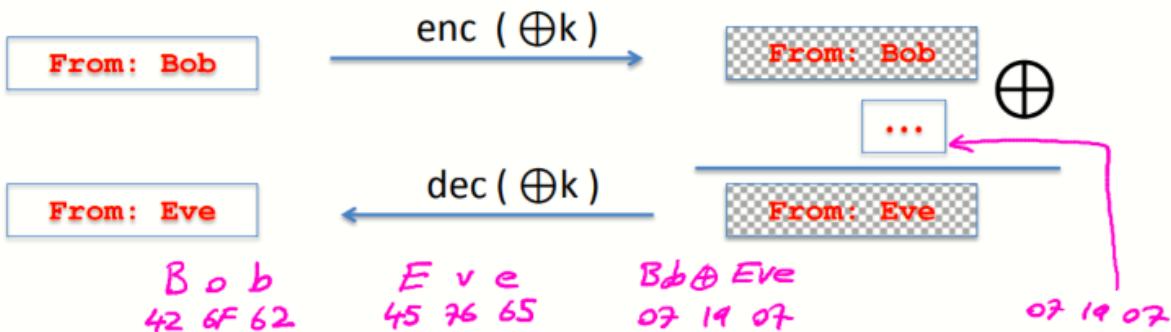
6、Attack 2: no integrity (OTP is malleable)

事实上一次性密码本和流密码并未提供完整性，仅在密钥仅使用一次的情况下提供保密性而已

事实上修改密文很容易，且会对明文造成相应的效果（即延展性）



对于上述方案，若加密后再XOR一个消息p，则解密得到的消息为m XOR p，而接收方并不能检测出这个对密文的改动（undetected）并且能对产生明文产生影响



如图，攻击者不一定知道明文内容，但是可以主动修改密文，使之与特定的p异或，而接收者并不会发现这个修改

W1 2-4 Real-World Stream Ciphers

1、Old example (software): RC4 (1987)

RC4接收变长的seed作为输入密钥（如128 bits），并将其扩展至2048 bits，之后进行简单的迭代，每次迭代产生1字节输出，迭代可以无限运行并产生输出

RC4广泛用于一些协议如HTTPS和WEP

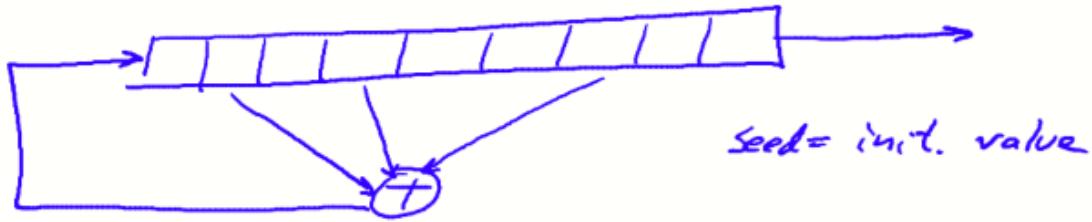
缺陷：

1. RC4的输出的第二字节为0的概率为2/256，即 $\text{Pr}[2\text{nd bytes} = 0] = 2/256$ ，（正常来说应该为1/256），因此导致使用RC4来加密消息时，其实第二字节并没有被很好的加密，若使用RC4，最好忽略前256字节，从第257字节开始使用
2. 如果观察很长的RC4输出，字节序00的概率为 $1/256^2 + 1/256^3$ （如果RC4是完全随机的，这个概率应该是 $1/256^2$ ），这个概率偏差会在RC4加密若干GB的数据后出现，同时这个缺陷可以用于预测生成器
3. 相关密钥攻击，与上一节的WEP类似

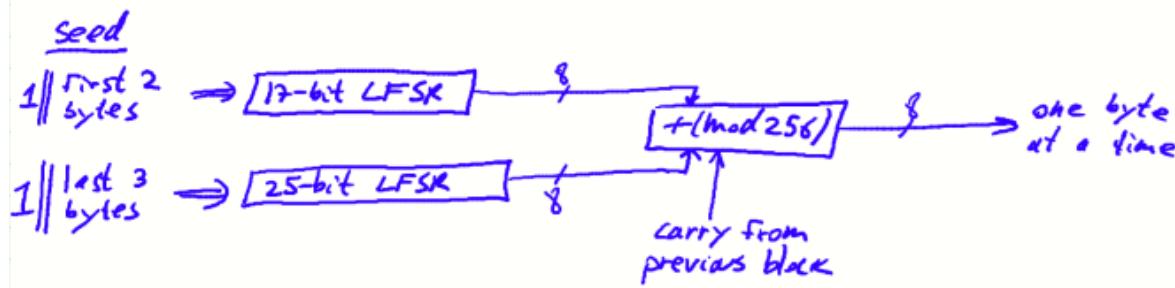
2、Old example (hardware): CSS (badly broken)

content scrambling system，用于加密DVD的流密码算法，CSS存在很明显的缺陷，可以被很轻易地破解

CSS基于线性反馈移位寄存器（Linear feedback shift register，LFSR），结构大致如下图所示



DVD encryption (CSS) 采用2个LFSR, GSM encryption (A5/1或A5/2) 采用三个LFSR, Bluetooth (E0) 采用四个, 均有严重的缺陷



CSS种子为40 bits (5字节), 分为前2字节和后3字节两个部分, 分别与1进行拼接后放入两个LFSR, 两个LFSR分别进行8次迭代, 分别产生8 bits输出, 之后通过加法器进行mod 256加并忽略进位 (但进位会参与到下一数据块的加法中), 这样每轮产生1字节的输出

CSS很容易被破解 (期望大约是 2^{17} 轮), 由于DVD加密采用的文件格式为MPEG, 因此可以知道明文的一部分前缀 (假设知道20字节前缀), 接着推测第一个LFSR的2-17 bits, 运行第一个LFSR得到20字节输出, 如果猜对了第一段LFSR的数据, 由于CSS系统的特性, 结合已知的前缀可以得到第二段LFSR的前20字节, 不断猜测可以得到两个LFSR的初始状态, 从而推测CSS后续的输出, 从而解密DVD

3、Modern stream ciphers: eStream (2008)

这个项目有五种不同的加密算法, 下列其中之一

生成器PRG: $\{0,1\}^s \times R \rightarrow \{0,1\}^n$, 且 n 远大于 s , 其中 R 为一nonce

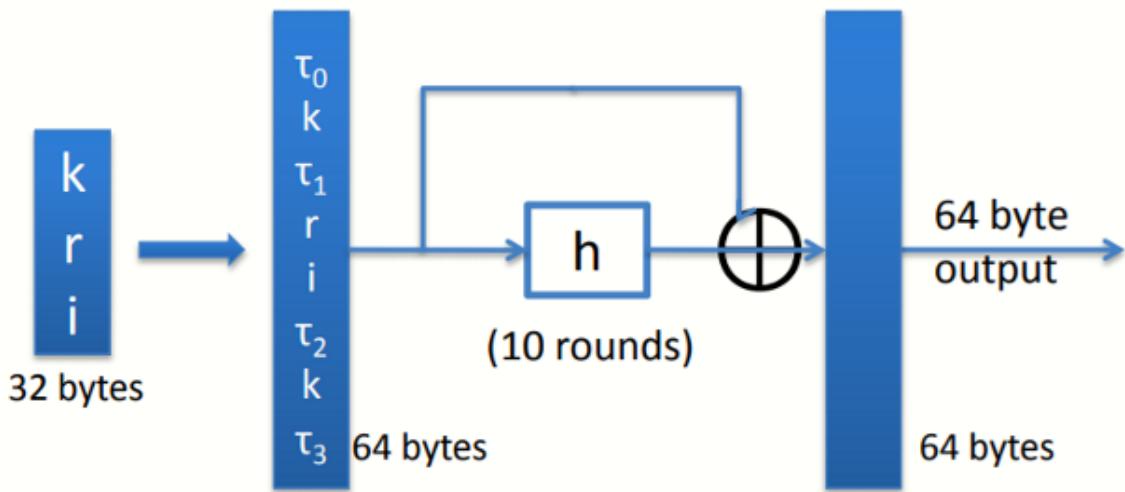
nonce: 对于给定的密钥, nonce是一个用不重复的数值

加密算法 $E(k, m ; r) = m \oplus PRG(k ; r)$, 由于nonce的特性, 使得你可以重用密钥 k 而 (k, r) 序列不会被使用超过一次

4、eStream: Salsa 20 (SoftWare+HardWare)

Salsa20: $\{0,1\}^{(128 \text{ or } 256)} \times \{0,1\}^{64} \rightarrow \{0,1\}^n$ (max $n = 273$ bits)

$Salsa20(k ; r) := H(k, (r, 0)) \parallel H(k, (r, 1)) \parallel \dots$



k 128 bits为密钥， r 为64 bits nonce， i 为64 bits index (类似于counter)， τ 为一系列32 bits常数，函数 h 为一可逆函数 (invertible function)，将上述64字节迭代10次后产生64字节输出，由于 h 是可逆函数，因此最后一轮结束后需要与初始64字节异或来防止逆向

Salsa性能很好，因此软硬件均可实现，其他现代流密码算法如Sosemanuk也有非常好的性能和安全性，因此不要使用RC4等又不安全效率又低的算法

Salsa在Crypto++ 5.6.0 [Wei Dai]上有

W1 2-5 PRG Security Definitions

1、PRG

$G:K \rightarrow \{0,1\}^n$ be a PRG

我们期望所有可能生成的串是等概的 (indistinguishable distribution)，我们定义的 G 仅能生成伪随机分布 (pseudo random distribution)，是因为种种空间非常小，导致 G 的输出仅占全部空间的很小一部分 ($\{0, 1\}^n$ 的很小的子集)

2、Statistical Tests

Statistical test on $\{0,1\}^n$: an alg. A s.t. $A(x)$ outputs "0" or "1"，输入 n bits串，输出0表示这个串并不是随机的，输出1表示是随机的，例子如下

Examples:

$$(1) \quad A(x)=1 \quad \text{iff} \quad |\#0(x) - \#1(x)| \leq 10 \cdot \sqrt{n}$$

$$(2) \quad A(x)=1 \quad \text{iff} \quad |\#00(x) - \frac{n}{4}| \leq 10 \cdot \sqrt{n}$$

$$(3) \quad A(x)=1 \quad \text{iff} \quad \max\text{-run-of-0}(x) < 10 \cdot \log_2(n)$$

1. S对于给定的串 x ，当且仅当其包含的0的个数与1的个数的差小于等于10倍根号n时， A 输出1（即0和1的个数差的不是太多，10倍根号n只是举例，实际上不一定是这个值）
2. 对于给定的串 x ，当且仅当其连续的00出现的次数与 $n/4$ 的差小于.....
3. 当且仅当串中最长的连续的0的个数小于10倍 $\lg n$ 时， A 输出1

3、Advantage

如何说明统计检验算法是好是坏?

Let $G:K \rightarrow \{0,1\}^n$ be a PRG and A a stat. test on $\{0,1\}^n$

Define:

$$\text{Adv}_{\text{PRG}}[A,G] = \left| \Pr_{k \in K} [A(G(k))=1] - \Pr_{r \in \{0,1\}^n} [A(r)=1] \right| \in [0,1]$$

Adv close to 1 $\Rightarrow A$ can dist. G from random
 Adv close to 0 $\Rightarrow A$ cannot

定义如下：统计检验算法A的Advantage为：对于密钥空间K中随机选择的k值， $A[G(k)]$ 输出1的概率与以真随机序列作为输入的 $A(r)$ 输出1的概率的差值绝对值， Adv 越接近1说明统计检验可以将G和真随机区分开来，越接近0说明不能区分

统计检验的目的：判断一个PRG是否能产生优秀的伪随机序列，如果难以区分其产生的串与真随机串，则表明是个安全的PRG

一个简单的例子

Suppose $G:K \rightarrow \{0,1\}^n$ satisfies $\text{msb}(G(k)) = 1$ for 2/3 of keys in K

Define stat. test $A(x)$ as:

if [$\text{msb}(x)=1$] output "1" else output "0"

Then

$$\text{Adv}_{\text{PRG}}[A,G] = \left| \overbrace{\Pr[A(G(k))=1]}^{2/3} - \overbrace{\Pr[A(r)=1]}^{1/3} \right| =$$



上述例子表明， A 的 Adv 为 $1/6$ （仍然是个较大的数字），说明 A 仍然可以将 G 和真随机区分开来 (breaks the generator G with advantage $1/6$)

4、Secure PRGs: crypto definition

Def: We say that $G:K \rightarrow \{0,1\}^n$ is a secure PRG if

\forall "eff" stat. tests A :

$\text{Adv}_{\text{PRG}}[A,G]$ is "negligible"

对于安全的PRG，其对任何有效的统计检验A，其优势 Adv 均可忽略不计（即输出一个非常接近于0的数，无法将其与真随机区分开来）

课后思考：是否存在可证明安全的PRG？目前未知，但如果证明了某个特定的PRG是安全的，意味着 $P \neq NP$

5、Easy fact: a secure PRG is unpredictable

PRG可预测意味着PRG其实并不安全

Suppose A is an efficient algorithm s.t.

$$\Pr_{k \in \mathcal{K}} [A(G(k))|_{1, \dots, i}) = G(k)|_{i+1}] > \frac{1}{2} + \epsilon$$

for non-negligible ϵ (e.g. $\epsilon = 1/1000$)

Define statistical test B as:

$$B(x) = \begin{cases} \text{if } A(x)|_{1, \dots, i} = x_{i+1} \text{ output 1} \\ \text{else output 0} \end{cases}$$

$$r \xleftarrow{R} \{0,1\}^n : \Pr[B(r) = 1] = \frac{1}{2}$$

$$r \xleftarrow{R} \mathcal{K} : \Pr[B(G(k)) = 1] > \frac{1}{2} + \epsilon$$

$$\Rightarrow \text{Adv}_{\text{PRG}}[B, G] = \left| \Pr[B(r) = 1] - \Pr[B(G(k)) = 1] \right| > \epsilon$$

其中 alg A, 对于输入 G(k) 的前 i bit, 预测第 i+1 bit 的概率大于 1/2, alg B 对于 alg A 预测成功时输出 1, 否则输出 0

对于真随机序列, alg B 输出 1 的概率为确定的 1/2, 而对于 G(k) 则是大于 1/2, 因此 Adv[B, G] 大于一个正数 ϵ

上述例子表明: 若 alg A 可以以 ϵ 的优势预测下一位, 则 alg B 可以以 ϵ 的优势区分之, 即若 A 是优秀的预测算法, 则 B 是优秀的统计检验算法来打破这个生成器

或者换个说法, 如果 G 是个优秀的生成器, 则意味着没有优秀的统计检验算法

6. Thm (Yao'82): an unpredictable PRG is secure

Let $G: K \rightarrow \{0,1\}^n$ be PRG

"Thm": if $\forall i \in \{0, \dots, n-1\}$ PRG G is unpredictable at pos. i then G is a secure PRG.

如果“下一位预测器”不能将 G 和 真随机 区别开来, 则没有统计检验算法可以做到

一个更普遍的说法: 两个分布在计算上不可区分

Let P_1 and P_2 be two distributions over $\{0,1\}^n$

Def: We say that P_1 and P_2 are

computationally indistinguishable (denoted $P_1 \approx_p P_2$)

if \forall “eff stat. tests A

$$\left| \Pr_{x \sim P_1} [A(x) = 1] - \Pr_{x \sim P_2} [A(x) = 1] \right| < \text{negligible}$$

Example: a PRG is secure if $\{k \xleftarrow{R} K : G(k)\} \approx_p \text{uniform}(\{0,1\}^n)$

W1 2-6 Semantic security

1、What is a secure cipher?

攻击者的能力：获取一段密文

可能的安全需求

1. 攻击者不能恢复密钥
2. 攻击者不能恢复全部的明文

香农的思路：密文不应反应出明文的任何信息（已知明文不能得出1 bit或者预测任何明文消息）

2、Shannon's perfect secrecy

Let (E, D) be a cipher over (K, M, C)

(E, D) has perfect secrecy if $\forall m_0, m_1 \in M \quad (|m_0| = |m_1|)$

$$\{E(k, m_0)\} = \{E(k, m_1)\} \quad \text{where } k \leftarrow K$$

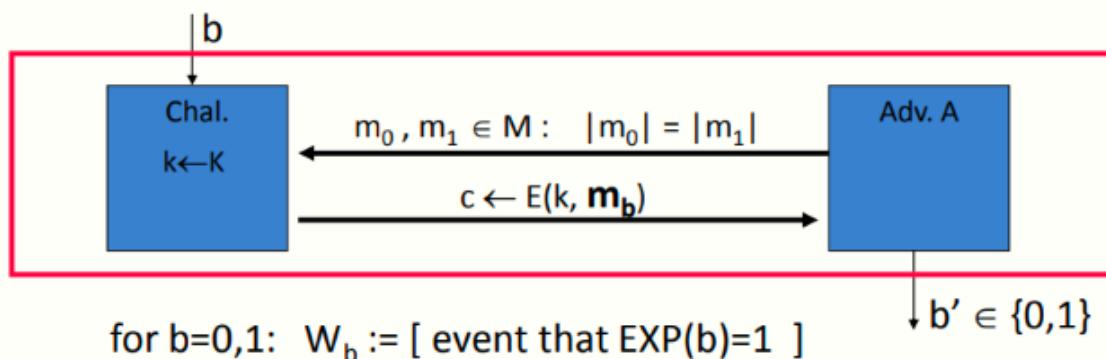
(E, D) has perfect secrecy if $\forall m_0, m_1 \in M \quad (|m_0| = |m_1|)$

$$\{E(k, m_0)\} \approx_p \{E(k, m_1)\} \quad \text{where } k \leftarrow K$$

1. 对于等长的消息 m_0 和 m_1 和来自密钥空间 K 的密钥 k , $E(k, m_0)$ 和 $E(k, m_1)$ 的分布应该是相同的, 因此攻击者不仅不知道我们加密的消息是 m_0 还是 m_1 , 同时也不知道消息的内容(但是这个定义要求非常严格, 需要很长的密钥, 短密钥在特定的流密码中无法满足如此严格的定义)
2. 稍微弱化一点的定义: 不要求完全相同的分布, 可以是计算上不可区分的分布(但条件还是稍微有些严格以至于难以满足)
3. 再增加一些约束条件: 即条件不需满足任给的 m_0, m_1 , 只需要满足特定的一组 m_0, m_1 (即向攻击者展示的一组)

3、Semantic Security (one-time key)

For $b=0,1$ define experiments $\text{EXP}(0)$ and $\text{EXP}(1)$ as:



$$\text{Adv}_{\text{SS}}[A, E] := \left| \Pr[W_0] - \Pr[W_1] \right| \in [0,1]$$

定义两个试验 $\text{EXP}(0)$ 和 $\text{EXP}(1)$, 分别代表攻击者得到了由 k 加密后的 m_0 和 m_1

$Wb(b=0,1)$ 表示事件：对于 $\text{EXP}(b)$, 攻击者输出1

如果说攻击者对于 $\text{EXP}(0)$ 和 $\text{EXP}(1)$ 均有相同概率输出1，则意味着在攻击者视角两个试验是一致的（即无法区分），若对于两个试验攻击者输出1的概率有显著差别，则表明他可以区分两个试验

语义安全优势： $\text{AdvSS}[A,E] = |\Pr[W_0] - \Pr[W_1]|$ ，取值介于0和1之间，若 Adv 接近于1，则表明攻击者有能力将加密后的 m_0 和 m_1 区分开来

严格定义：E is semantically secure if for all "efficient" A $\text{AdvSS}[A,E]$ is negligible，即对于所有的高效的攻击者而言，其优势均可忽略不计（没有高效的攻击者可以将加密后的 m_0 和 m_1 区分开来）

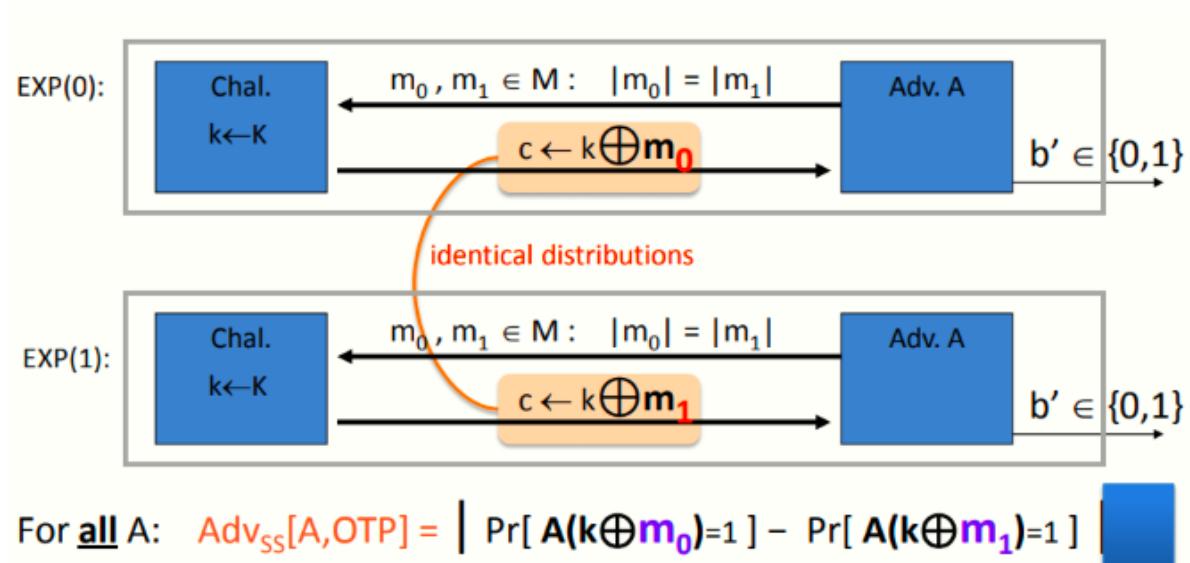
4、Examples

假设A总是可以根据密文推测出明文的最后一一位，消息 m_0 和 m_1 分别代表最低为为0和1的消息

由于A可以根据密文推测出消息的最低位，因此他总是可以将 m_0 和 m_1 区分开来，因此优势为1（即攻击者完全攻破了该系统）

上述例子表明，即便是攻击者可以得到关于明文的一位信息，系统也已经完全不安全，语义安全要求高效的攻击者不能得到关于明文的任何信息

5、OTP is semantically secure



对于所有的A而言，其优势均为0，因此OTP是语义安全的（ $k \oplus m_0$ 与 $k \oplus m_1$ 是同分布的，攻击者无法将两者区分开来）

W1 2-7 Stream Ciphers are Semantically Secure

1、Stream ciphers are semantically secure

Thm: $G:K \rightarrow \{0,1\}^n$ is a secure PRG \Rightarrow
stream cipher E derived from G is sem. sec.

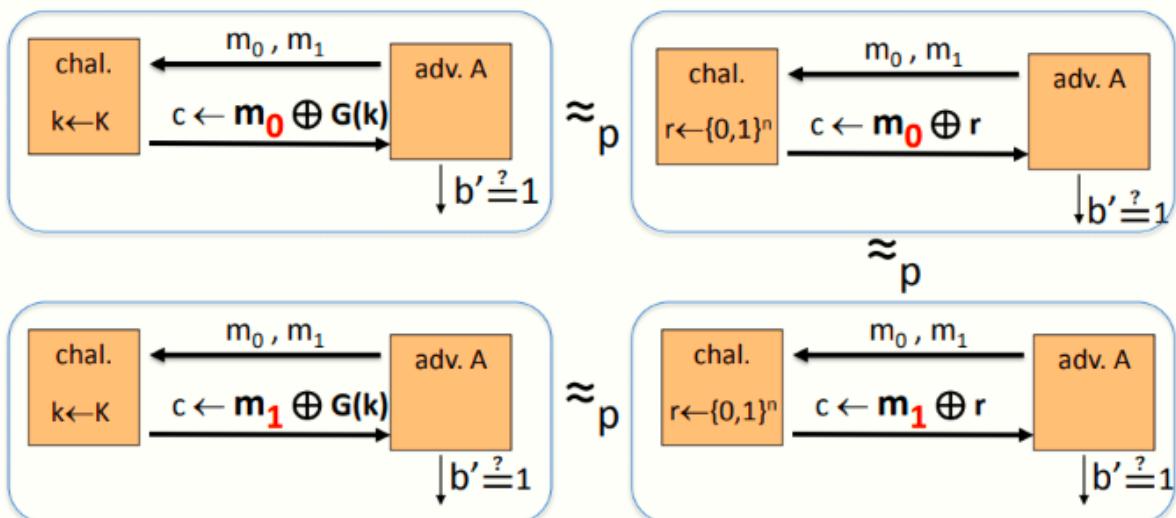
\forall sem. sec. adversary A , \exists a PRG adversary B s.t.

$$\text{Adv}_{\text{SS}}[A, E] \leq 2 \cdot \text{Adv}_{\text{PRG}}[B, G]$$

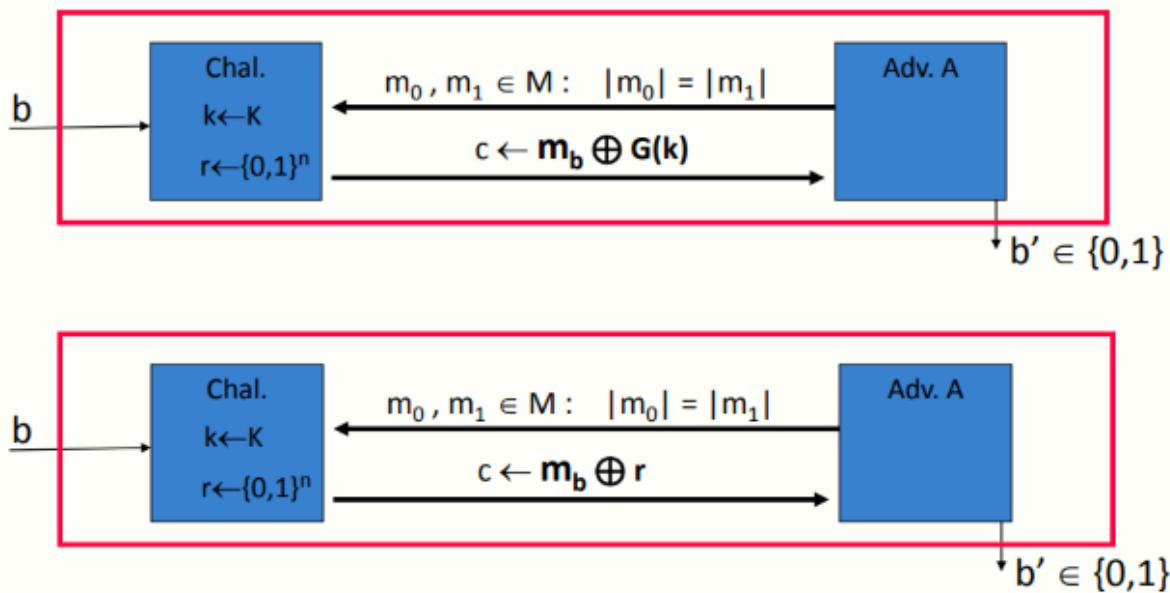
定理：用安全的PRG产生的流密钥进行流加密是语义安全的（语义安全SS，不是香农的完美安全Perfect Security）

2、Proof: intuition

比较直观的证明如下：

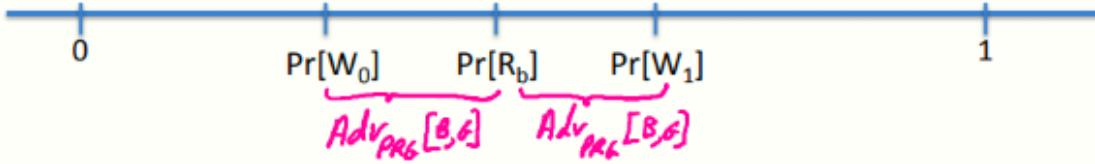


对于真随机串 r 和PRG生成的串 $G(k)$ ，如果PRG是安全的，则攻击者无法区分挑战者到底使用的是真随机还是PRG



Claim 1: $|\Pr[R_0] - \Pr[R_1]| = \text{Adv}_{\text{SS}}[A, \text{OTP}] = 0$

Claim 2: $\exists B: |\Pr[W_b] - \Pr[R_b]| = \text{Adv}_{\text{PRG}}[B, G]$ for $b=0, 1$



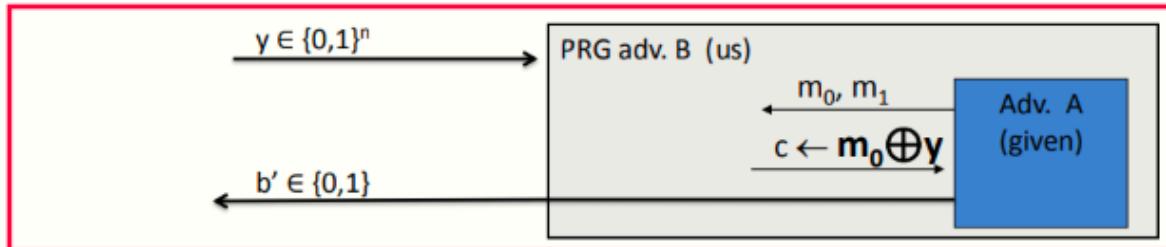
$$\Rightarrow \text{Adv}_{\text{SS}}[A, E] = |\Pr[W_0] - \Pr[W_1]| \leq 2 \cdot \text{Adv}_{\text{PRG}}[B, G]$$

符号说明:

- W_b : 在伪随机密码本下的事件 (原始的语义安全游戏的事件)
- R_b : 在一次性密码本下的事件 (更换为OTP后的游戏的事件)
- A : 语义安全的攻击者

Proof of claim 2: $\exists B: |\Pr[W_0] - \Pr[R_0]| = \text{Adv}_{\text{PRG}}[B, G]$

Algorithm B:



$$\text{Adv}_{\text{PRG}}[B, G] = \left| \Pr_{r \in \{0,1\}^n} [B(r) = 1] - \Pr_{k \in \mathcal{K}} [B(g(k)) = 1] \right| = |\Pr[R_0] - \Pr[W_0]|$$

W1 知识梳理

统计检验的作用: 即随机性检验, 目的是判断一个PRG是否能产生好的伪随机序列, 若一个PRG产生的伪随机序列和真随机序列难以区分, 那么就说这是一个安全的PRG

安全的PRG: 不可预测的PRG是安全的PRG

定义一个统计检验 B , 和一个有效的预测算法集合 A , 如果 A 预测到了结果, B 就输出0, 否则输出1

若 B 的优势接近于1, 就说明序列随机性强, 若 B 的优势接近于0, 还说明 A 是不可预测的 (因为预测到了结果回输出0)

语义安全性检测模型: 特殊的统计测试, 输出为攻击者的输出

区别于统计检验, 统计检验的输出含义为PRG产生的串的随机性如何, 语义安全检测中攻击者的输出含义为攻击者对密文所对应的明文的判断 (即挑战者做了 $\text{EXP}(0)$ 还是 $\text{EXP}(1)$), 二者本质没有区别, 因为攻击者的优势还是与PRG产生串的随机性有关

总的来说：随机性越好，则语义安全性越好

检验中的优势：

W1 Problem Set && Programming Assignment

Q1:

1. Data compression is often used in data storage and transmission. Suppose you want to use data compression in conjunction with encryption. Does it make more sense to:

- The order does not matter -- either one is fine.
- Compress then encrypt.
- The order does not matter -- neither one will compress the data.
- Encrypt then compress.

密文看上去像随机的字符串，因此应当在加密前就进行压缩

Q2:

2. Let $G : \{0,1\}^s \rightarrow \{0,1\}^n$ be a secure PRG. Which of the following is a secure PRG (there is more than one correct answer):

- $G'(k) = G(k)[0, \dots, n-2]$ (i.e., $G'(k)$ drops the last bit of $G(k)$)
- $G'(k) = G(k \oplus 1^s)$
- $G'(k) = G(k) \parallel G(k)$
(here \parallel denotes concatenation)
- $G'(k) = G(k) \oplus 1^n$
- $G'(k) = G(k) \parallel 0$
(here \parallel denotes concatenation)
- $G'(k) = G(0)$

Q3:

3. Let $G : K \rightarrow \{0,1\}^n$ be a secure PRG.

Define $G'(k_1, k_2) = G(k_1) \wedge G(k_2)$ where \wedge is the bit-wise AND function. Consider the following statistical test A on $\{0,1\}^n$:

$A(x)$ outputs $\text{LSB}(x)$, the least significant bit of x .

What is $\text{Adv}_{\text{PRG}}[A, G']$?

You may assume that $\text{LSB}(G(k))$ is 0 for exactly half the seeds k in K .

Note: Please enter the advantage as a decimal between 0 and 1 with a leading 0. If the advantage is 3/4, you should enter it as 0.75

$G(k_1)$ 和 $G(k_2)$ 同时为1时，A输出1

Q4:

4. Let (E, D) be a (one-time) semantically secure cipher with key space $K = \{0, 1\}^\ell$. A bank wishes to split a decryption key $k \in \{0, 1\}^\ell$ into two pieces p_1 and p_2 so that both are needed for decryption. The piece p_1 can be given to one executive and p_2 to another so that both must contribute their pieces for decryption to proceed.

The bank generates random k_1 in $\{0, 1\}^\ell$ and sets $k'_1 \leftarrow k \oplus k_1$. Note that $k_1 \oplus k'_1 = k$. The bank can give k_1 to one executive and k'_1 to another. Both must be present for decryption to proceed since, by itself, each piece contains no information about the secret key k (note that each piece is a one-time pad encryption of k).

Now, suppose the bank wants to split k into three pieces p_1, p_2, p_3 so that any two of the pieces enable decryption using k . This ensures that even if one executive is out sick, decryption can still succeed. To do so the bank generates two random pairs (k_1, k'_1) and (k_2, k'_2) as in the previous paragraph so that $k_1 \oplus k'_1 = k_2 \oplus k'_2 = k$.

How should the bank assign pieces so that any two pieces enable decryption using k , but no single piece can decrypt?

- $p_1 = (k_1, k_2), \quad p_2 = (k_2, k'_2), \quad p_3 = (k'_2)$
- $p_1 = (k_1, k_2), \quad p_2 = (k'_1, k_2), \quad p_3 = (k'_2)$
- $p_1 = (k_1, k_2), \quad p_2 = (k'_1), \quad p_3 = (k'_2)$
- $p_1 = (k_1, k_2), \quad p_2 = (k_1, k_2), \quad p_3 = (k'_2)$
- $p_1 = (k_1, k_2), \quad p_2 = (k'_1, k'_2), \quad p_3 = (k'_2)$

对于经理1和经理2而言，可以用 k_1 和 k_1' 解密，对于经理1和经理3，可以用 k_2 和 k_2' 解密，对于经理2和经理3，可以用 k_2 和 k_2' 解密

Q5:

5. Let $M = C = K = \{0, 1, 2, \dots, 255\}$

and consider the following cipher defined over (K, M, C) :

$$E(k, m) = m + k \pmod{256} \quad ; \quad D(k, c) = c - k \pmod{256}.$$

Does this cipher have perfect secrecy?

- No, only the One Time Pad has perfect secrecy.
- No, there is a simple attack on this cipher.
- Yes.

与OTP一样，只有一个密钥将给定的消息映射到密文，因此是完美安全的

Q6:

6. Let (E, D) be a (one-time) semantically secure cipher where the message and ciphertext space is $\{0, 1\}^n$. Which of the following encryption schemes are (one-time) semantically secure?

- $E'(k, m) = 0 \parallel E(k, m)$ (i.e. prepend 0 to the ciphertext)
- $E'(k, m) = E(k, m) \parallel k$
- $E'(k, m) = E(0^n, m)$
- $E'(k, m) = \text{reverse}(E(k, m))$
- $E'(k, m) = E(k, m) \parallel \text{LSB}(m)$
- $E'((k, k'), m) = E(k, m) \parallel E(k', m)$

Q7:

7. Suppose you are told that the one time pad encryption of the message "attack at dawn" is `09e1c5f70a65ac519458e7e53f36`

(the plaintext letters are encoded as 8-bit ASCII and the given ciphertext is written in [hex](#)). What would be the one time pad encryption of the message "attack at dusk" under the same OTP key?

`6c73d5240a948c86981bc2808548`

6c73d5240a948c86981bc2808548, 密钥不变，明文消息仅有最后三个字符不相同，只需要计算Hex串的最后六位

Q8:

Question 8

The movie industry wants to protect digital content distributed on DVD's. We develop a variant of a method used to protect Blu-ray disks called [AACS](#).

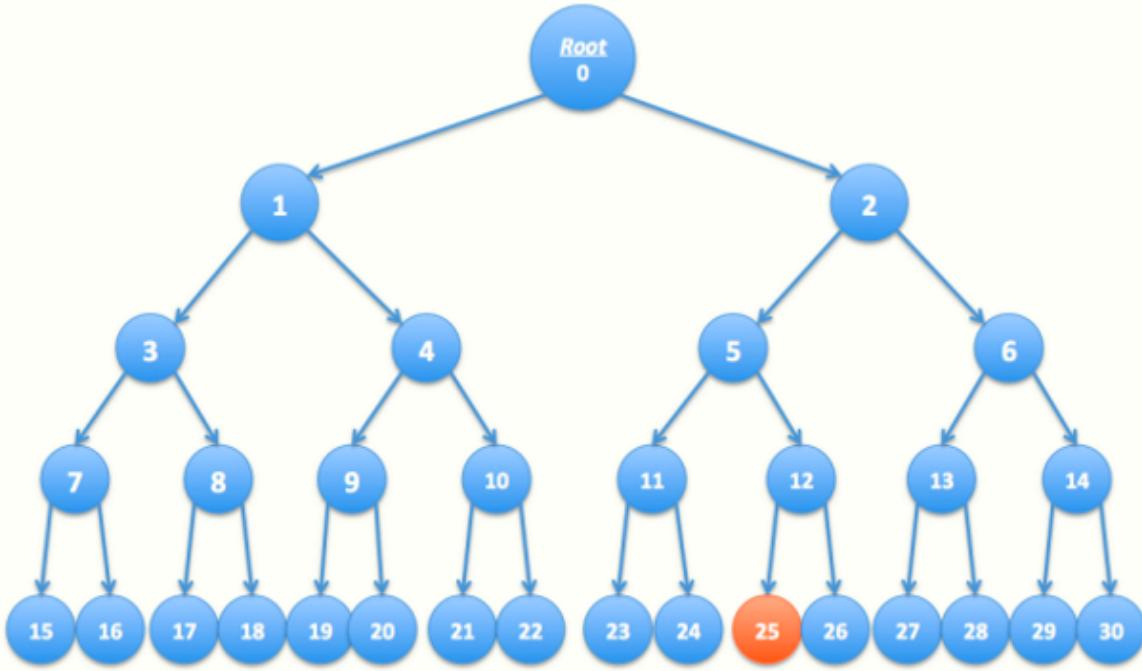
Suppose there are at most a total of n DVD players in the world (e.g. $n = 2^{32}$). We view these n players as the leaves of a binary tree of height $\log_2 n$. Each node in this binary tree contains an AES key k_i . These keys are kept secret from consumers and are fixed for all time. At manufacturing time each DVD player is assigned a serial number $i \in [0, n - 1]$. Consider the set of nodes S_i along the path from the root to leaf number i in the binary tree. The manufacturer of the DVD player embeds in player number i the keys associated with the nodes in the set S_i . A DVD movie m is encrypted as

$$E(k_{\text{root}}, k) \| E(k, m)$$

where k is a random AES key called a content-key and k_{root} is the key associated with the root of the tree. Since all DVD players have the key k_{root} all players can decrypt the movie m . We refer to $E(k_{\text{root}}, k)$ as the header and $E(k, m)$ as the body. In what follows the DVD header may contain multiple ciphertexts where each ciphertext is the encryption of the content-key k under some key k_i in the binary tree.

Suppose the keys embedded in DVD player number r are exposed by hackers and published on the Internet. In this problem we show that when the movie industry distributes a new DVD movie, they can encrypt the contents of the DVD using a slightly larger header (containing about $\log_2 n$ keys) so that all DVD players, except for player number r , can decrypt the movie. In effect, the movie industry disables player number r without affecting other players.

As shown below, consider a tree with $n = 16$ leaves. Suppose the leaf node labeled 25 corresponds to an exposed DVD player key. Check the set of keys below under which to encrypt the key k so that every player other than player 25 can decrypt the DVD. Only four keys are needed.



key₁需要加密（确保key₀的左子树加密而不影响右子树），key₆需要加密（确保key₂的右子树加密而不影响右子树），key₁₁和key₂₆需要加密（若加密5则相当于也加密了25），因此，1、6、11、26需要加密

Q9:

9. Continuing with the previous question, if there are n DVD players, what is the number of keys under which the content key k must be encrypted if exactly one DVD player's key needs to be revoked?
- 2
 - $\log_2 n$
 - $n/2$
 - $n - 1$
 - \sqrt{n}

满二叉树，从根至叶节点每层需要加密一个key，共计lg n个key

Q10:

Question 10

Continuing with question 8, suppose the leaf nodes labeled 16, 18, and 25 correspond to exposed DVD player keys. Check the smallest set of keys under which to encrypt the key k so that every player other than players 16, 18, 25 can decrypt the DVD. Only six keys are needed.

同Q8, key4、6、11、15、17、26需要加密

Programming Assignment: Many Time Pad

Let us see what goes wrong when a stream cipher key is used more than once. Below are eleven hex-encoded ciphertexts that are the result of encrypting eleven plaintexts with a stream cipher, all with the same stream cipher key. Your goal is to decrypt the last ciphertext, and submit the secret message within it as solution.

Hint: XOR the ciphertexts together, and consider what happens when a space is XORed with a character in [a-zA-Z].

ciphertext #1:

```
315c4eeaa8b5f8aaf9174145bf43e1784b8fa00dc71d885a804e5ee9fa40b16349c146fb778cdf2d3aff021  
dff  
f5b403b510d0d0455468aeb98622b137dae857553cccd8883a7bc37520e06e515d22c954eba5025b8cc57  
e  
e59418ce7dc6bc41556bdb36bbca3e8774301fbcaa3b83b220809560987815f65286764703de0f3d52440  
0a19b159610b11ef3e
```

ciphertext #2:

```
234c02ecbbfbafa3ed18510abd11fa724fcda2018a1a8342cf064bbde548b12b07df44ba7191d9606ef408  
1ff  
de5ad46a5069d9f7f543bedb9c861bf29c7e205132eda9382b0bc2c5c4b45f919cf3a9f1cb74151f6d551f4  
480c82b2cb24cc5b028aa76eb7b4ab24171ab3cdadb8356f
```

ciphertext #3:

```
32510ba9a7b2bba9b8005d43a304b5714cc0bb0c8a34884dd91304b8ad40b62b07df44ba6e9d8a2368e  
51  
d04e0e7b207b70b9b8261112bacb6c866a232dfe257527dc29398f5f3251a0d47e503c66e935de81230b  
59b7afb5f41afa8d661cb
```

ciphertext #4:

```
32510ba9aab2a8a4fd06414fb517b5605cc0aa0dc91a8908c2064ba8ad5ea06a029056f47a8ad3306ef50  
21  
eafe1ac01a81197847a5c68a1b78769a37bc8f4575432c198ccb4ef63590256e305cd3a9544ee4160ead45  
a  
ef520489e7da7d835402bca670bda8eb775200b8dabbba246b130f040d8ec6447e2c767f3d30ed81ea2e  
4c1404e1315a1010e7229be6636aaa
```

ciphertext #5:

```
3f561ba9adb4b6ebec54424ba317b564418fac0dd35f8c08d31a1fe9e24fe56808c213f17c81d9607cee02  
1d  
afe1e001b21ade877a5e68bea88d61b93ac5ee0d562e8e9582f5ef375f0a4ae20ed86e935de81230b59b7  
3fb4302cd95d770c65b40aaa065f2a5e33a5a0bb5dcaba43722130f042f8ec85b7c2070
```

ciphertext #6:

```
32510bfbaacfbb9befd54415da243e1695ecabd58c519cd4bd2061bbde24eb76a19d84aba34d8de287be  
84  
d07e7e9a30ee714979c7e1123a8bd9822a33ecaf512472e8e8f8db3f9635c1949e640c621854eba0d79ec  
cf  
52ff111284b4cc61d11902aebc66f2b2e436434eacc0aba938220b084800c2ca4e693522643573b2c4ce35  
050b0cf774201f0fe52ac9f26d71b6cf61a711cc229f77ace7aa88a2f19983122b11be87a59c355d25f8e4
```

ciphertext #7:

```
32510bfbaacfbb9befd54415da243e1695ecabd58c519cd4bd90f1fa6ea5ba47b01c909ba7696cf606ef40c  
04  
afe1ac0aa8148dd066592ded9f8774b529c7ea125d298e8883f5e9305f4b44f915cb2bd05af51373fd9b4a
```

f5
11039fa2d96f83414aaaf261bda2e97b170fb5cce2a53e675c154c0d9681596934777e2275b381ce2e4058
2
afe67650b13e72287ff2270abcf73bb028932836fbdecfecee0a3b894473c1bbeb6b4913a536ce4f9b13f1e
fff 71ea313c8661dd9a4ce

ciphertext #8:
315c4eeaa8b5f8bffd11155ea506b56041c6a00c8a08854dd21a4bbde54ce56801d943ba708b8a3574f40
c0
0fff9e00fa1439fd0654327a3bfc860b92f89ee04132ecb9298f5fd2d5e4b45e40ecc3b9d59e9417df7c95bb
a 410e9aa2ca24c5474da2f276baa3ac325918b2daada43d6712150441c2e04f6565517f317da9d3

ciphertext #9:
271946f9bbb2aeadec111841a81abc300ecaa01bd8069d5cc91005e9fe4aad6e04d513e96d99de2569bc
5e
50eeeca709b50a8a987f4264edb6896fb537d0a716132ddc938fb0f836480e06ed0fc6e9759f40462f9cf5
7f
4564186a2c1778f1543efa270bda5e933421cbe88a4a52222190f471e9bd15f652b653b7071aec59a2705
08 1ffe72651d08f822c9ed6d76e48b63ab15d0208573a7eef027

ciphertext #10:
466d06ece998b7a2fb1d464fed2ced7641ddaa3cc31c9941cf110abbf409ed39598005b3399ccfafb61d03
1
5fca0a314be138a9f32503bedac8067f03adbf3575c3b8edc9ba7f537530541ab0f9f3cd04ff50d66f1d559
ba 520e89a2cb2a83

target ciphertext (decrypt this one):
32510ba9babebbbbef001547a810e67149caee11d945cd7fc81a05e9f85aac650e9052ba6a8cd8257bf14
d1 3e6f0a803b54fde9e77472dbff89d71b57bddef121336cb85ccb8f3315f4b52e301d16e9f52f904

分析：查阅ASCII码表，可知[A-Z]用[0x41-0x5a]表示，[a-z]用[0x61-0x7a]表示，空格用0x20表示
简单的计算可知，大写字母与空格XOR，可得到小写字母，小写字母与空格XOR可得到大写字母
同时利用了XOR的特性，一个数与XOR运算偶数次可得到原来的数，即 $x \wedge y \wedge y = x$

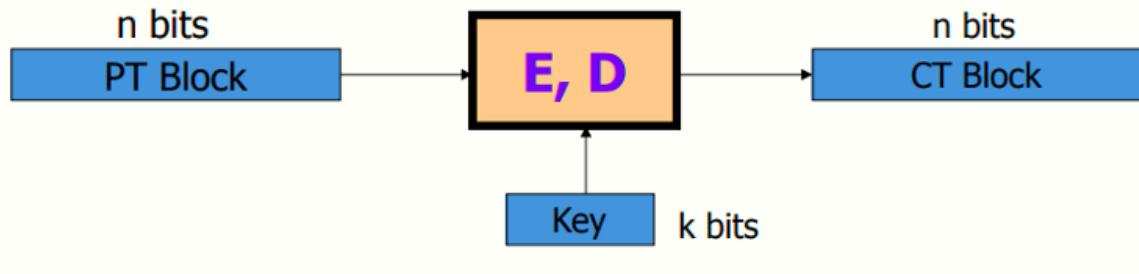
由于每个ciphertext均为相同的key加密，则意味着 $c1=m1^k$, $c2=m2^k$, $c1 \wedge c2 = m1 \wedge m2$ ，结合上述分析，可知若两段密文XOR的结果为一个有意义的字母，则对应位上的两段明文应该是一个英文字母和一个空格

答案：The secret message is: when using a stream cipher, never use the key more than once

W2 3-1 What is a block cipher?

1、Block ciphers: crypto work horse

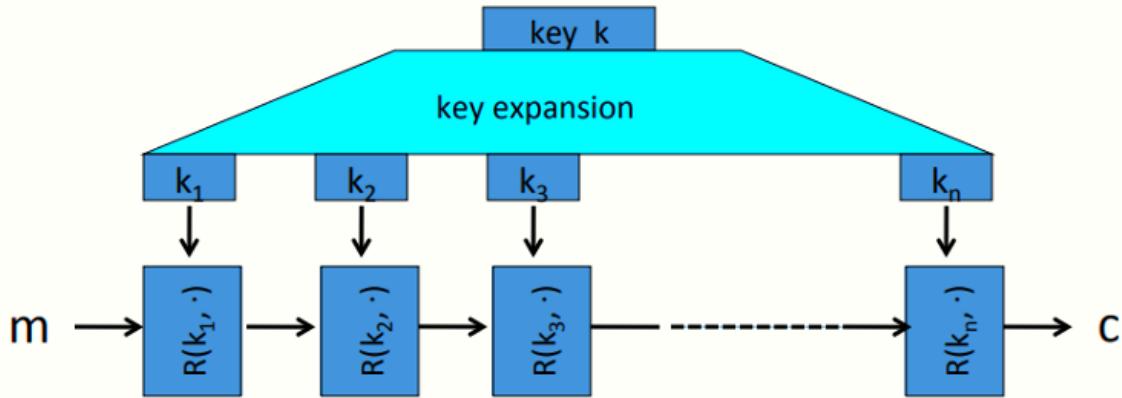
块密码包含两个算法E和D，均接收k bit的密钥和N bit消息作为输入，E接收明文消息输出密文，D接收密文消息输出明文



标准算法：

- 3DES: n=64 bits, k=168 bits
- AES: n=128 bits, k = 128, 192, 256 bits (密钥越长越安全但计算速度也越慢)

2、Block Ciphers Built by Iteration



块密码从初始化开始，将密钥k进行密钥扩展，生成n个子密钥（即轮密钥），在每轮使用不同的轮密钥，将其作为轮函数的输入

轮函数： $R(k, m)$ ，接收轮密钥和当前状态的消息，输出结果，在加密函数中，第一轮轮函数接收明文消息作为输入，最后一轮轮函数将密文作为输出，轮函数的轮次取决于算法（3DES高达48轮次，AES只有10-12轮）

3、Performance:

使用Crypto++ 5.6.0[Wei Dai]，块密码比流密码慢得多（效率大约只有1/6），但可以做到很多在RC4等构造中不能高效完成的任务

4、Abstractly: PRPs and PRFs

(1) 伪随机函数：Pseudo Random Function，PRF

$$F: K \times X \rightarrow Y$$

PRF是一个定义在三元组 (K, X, Y) 上的函数，K为密钥空间，X为输入空间，Y为输出空间

对F的要求：要有某种有效的方式评估函数F，可以不需要具有可逆性

(2) 伪随机置换：Pseudo Random Permutation，PRP

$$E: K \times X \rightarrow X$$

PRP是一个定义在二元组 (K, X) 上的函数，K为密钥空间，X为输入空间，并输出X中的一个元素

对E的要求：存在有效的确定性算法来评估E，对于给定的k，函数E是一一对应的，由于是一一对应的，因此也是可逆的，因此需要具有高效的反演算法D (k, y) （给定输出时计算对应输入）

5、Running example

Example PRPs: 3DES, AES, ...

- AES: $K \times X \rightarrow X$ where $K = X = \{0,1\}^{128}$
- 3DES: $K \times X \rightarrow X$ where $X = \{0,1\}^{64}, K = \{0,1\}^{168}$

从功能上来说，每个PRP都是一个PRF (PRP是一个 $X=Y$ 且存在高效逆运算的PRF)

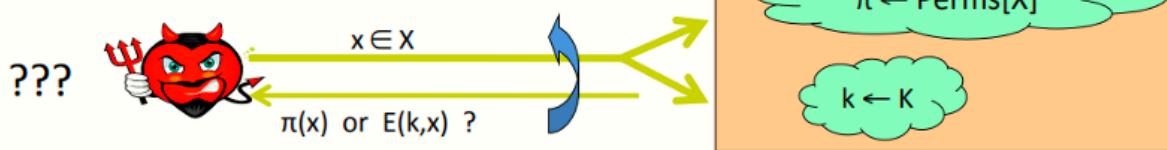
6、Secure PRFs

记 $F: K \times X \rightarrow Y$ 是一个PRF, 定义下列两个记号

- $\text{Funs}[X,Y]$: 所有由 X 到 Y 的函数的集合
- $\text{Set } F \text{ (SF)} : \{F(k,\cdot) \text{ s.t. } k \in K\} \subseteq \text{Funs}[X,Y]$

直观上来说：若无法区分从 $\text{Funs}[X,Y]$ 中随机选择的 F 和 $\text{Set } F$ 中随机选择的 F ，则认为 PRF 是安全的 ($\text{Set } F$ 大小为 $|K|$, $\text{Funs}[X,Y]$ 则有 $|Y|^{|X|}$)，或者说两个集合中的均匀分布不可区分 (攻击者无法判断到底在和哪一个集合中的函数交互)

- Intuition: a PRP is **secure** if
a random function in $\text{Perms}[X]$ is indistinguishable from
a random function in S_F



7、An easy application: PRF \Rightarrow PRG

记 $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ 为一安全PRF

$G: K \rightarrow \{0,1\}^n$ 为一安全PRG, 种子空间为PRF的密钥空间, 输出空间为 t 个 n bits 块 (t 为可选参数)

W2 3-2 The data encryption standard (DES)

1、The Data Encryption Standard (DES)

历史简介：

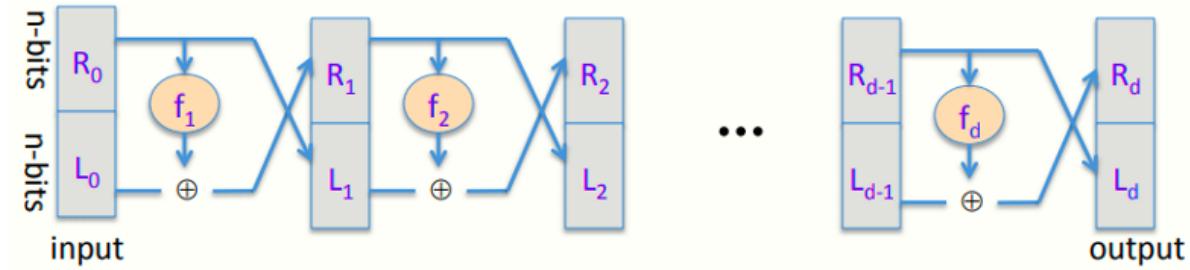
- 1970年Feistel带领的团队设计了一个加密算法Lucifer
- 1973年IBM上交了Lucifer的变体
- 1976年NBS将DES作为标准, 密钥长度56 bits, 块长度64 bits
- 1997年DES被暴力破解
- 2000年NIST将Rijndael的AES代替DES
- DES曾作为商密用于银行等领域

2、DES: core idea – Feistel Network

DES的核心概念：feistel网络

Feistel网络是一个运用 d 个随机函数 $f_1 - f_d$ (小f) 来组建加密方法, $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$ ，这些函数将 n bits 输入映射到 n bits 输出上, 均是随机的函数 (可以不需要可逆)

Feistel期望用这d个函数构建一个可逆的函数，因此需要构建一个新函数F（大F）， $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ ，将 $2n$ bits输入映射到 $2n$ bits输出上



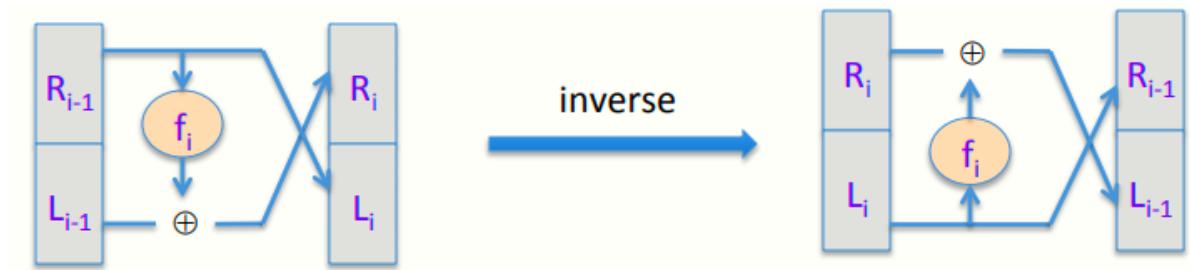
Feistel的每一轮将 $2n$ bits输入分为左右两部分，右半部分既作为下一轮的左半部分，又作为轮函数 f_i ($i=1, 2, \dots, d$) 的输入，左半部分与轮函数 f_i 的输出XOR后作为下一次迭代的右半部分，综上，有如下符号定义

$$R_i = f_i(R_{i-1}) \oplus L_{i-1}$$

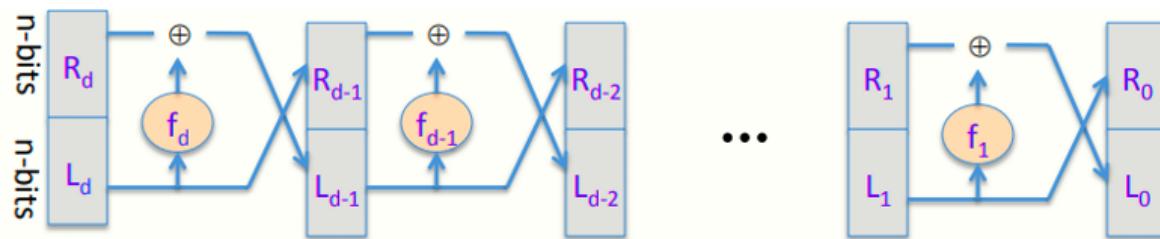
$$L_{i-1} = R_{i-1}$$

其中 $i=1, 2, \dots, d$

对于任意的 $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$ ，Feistel网络的函数 $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ 是可逆的，其逆如下图所示



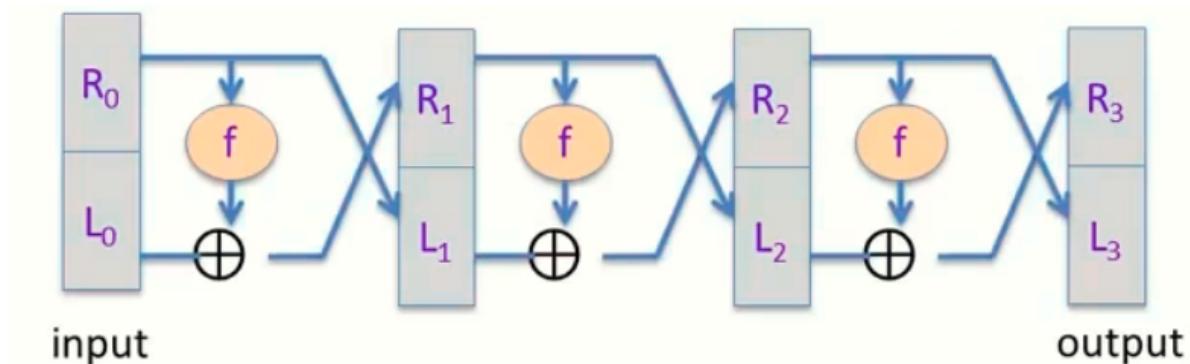
3、Decryption circuit



如图所示，解密与加密有相似的结构，只是将轮函数f放在左侧，且要逆序使用轮函数f

Feistel网络在许多块加密中均有使用 (AES除外)

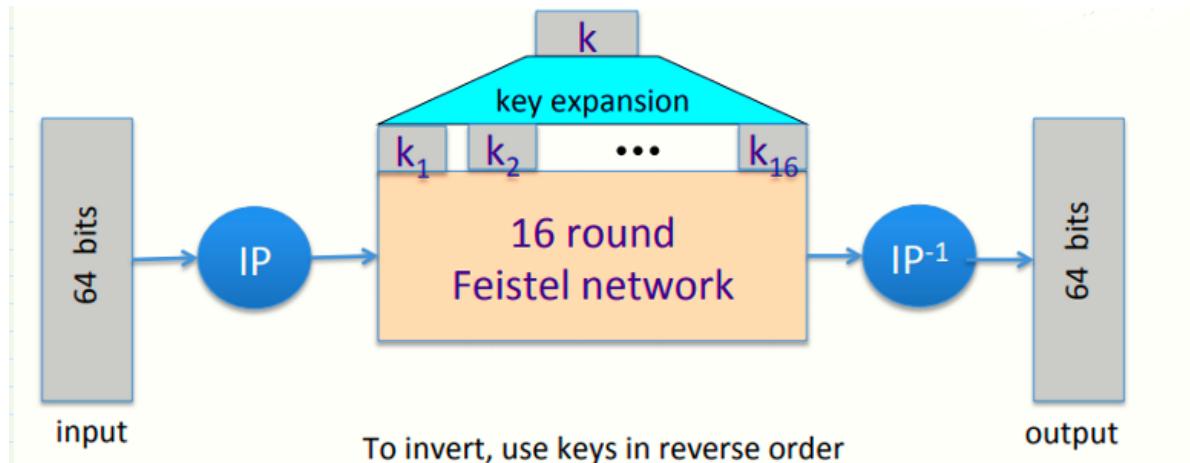
定理：若 $f: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ 是一个安全的PRF，则3轮Feistel函数 $F: K_3 \times \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ 是一个安全的PRP



三轮加密，每轮使用独立的密钥

4、DES: 16 round Feistel network

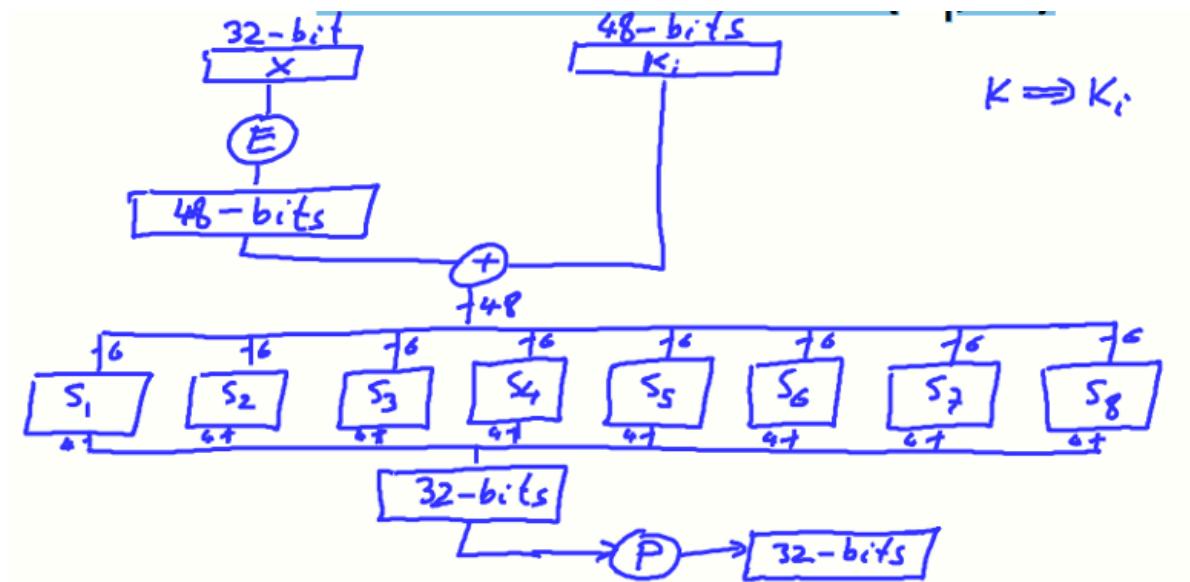
$f_1, \dots, f_{16}: \{0,1\}^{32} \rightarrow \{0,1\}^{32}$, $f_i(x) = F(k_i, x)$, 其中 k_i 为第*i*轮的轮密钥, 由初始密钥k计算得出



DES算法先对输入进行初始置换IP (initial permutation), 之后过16轮Feistel网络, 最后再做一次初试逆置换 IP^{-1} , 最终输出结果

DES会将初始密钥k进行密钥扩展, 将其扩展成16个48 bits的子密钥, 每个子密钥用于每轮的轮函数
解密算法中, 只需要逆序使用这些轮密钥即可

5、The function $F(k_i, x)$



如图所示

- E盒: expand, 位扩展, 将一些位重复输出, 目的是将32 bits的部分扩展到48 bits, 以用于与轮密钥XOR
- 扩展后的输入与轮密钥XOR后分为8个部分, 每部分6 bits, 作为各个S盒的输入
- S盒: function $\{0,1\}^6 \rightarrow \{0,1\}^4$, 将6 bits的输入转化为4 bits输出
- 最后将各个S盒的输出拼接成32 bits, 经过P盒置换后得到32 bits输出

6、The S-boxes

$S_i : \{0,1\}^6 \rightarrow \{0,1\}^4$, 接收6 bits输入, 输出4 bits

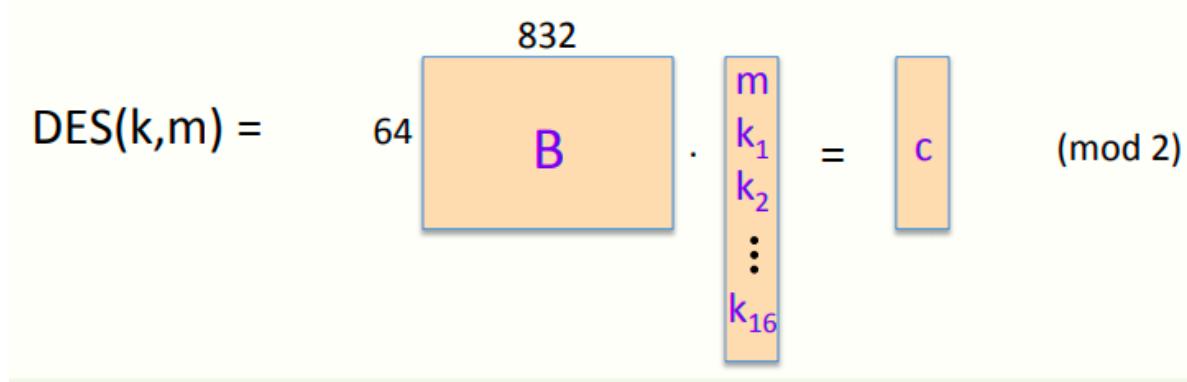
S ₅	Middle 4 bits of input																
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

S盒将输入的6 bits中的最高位和最低位作为行，中间四位作为列，之后选择对应位置的4 bits输出

7、Example: a bad S-box choice

如果说S盒可以被表示一些等式如： $S_i(x) = A_i \cdot x \pmod{2}$ ($\pmod{2}$ 矩阵乘法)，则S盒是不安全的

因为如果S盒是线性的，则导致DES仅仅是在计算XOR、置换和移动为，从而意味着DES是一个线性函数，则DES又可以被表示为一个 $\pmod{2}$ 的矩阵乘法，如下图所示



则 $DES(k, m_1) \oplus DES(k, m_2) \oplus DES(k, m_3) = DES(k, m_1 \oplus m_2 \oplus m_3)$

综上，若S盒全部都是线性的话，DES非常不安全，即便是差不多线性的 (close to being linear) 也不安全，只要有一定规模的输入即可以短时间内破解出密钥

8、Choosing the S-boxes and P-box

如果说随机选择S盒和P盒，会导致得到一个不安全的块密码

因此设计者对S盒和P盒的选择上有一定的要求：

- 必须非常的不像线性函数，即没有函数与S盒的大部分输出相同
- 为了防止攻击，还有其他一些规则如4到1映射规则（每个输出正好有四个前像）

W2 3-3 Exhaustive Search Attacks

1、Exhaustive Search for block cipher key

目标：对于一些给定的输入输出消息对， $(m_i, c_i = E(k, m_i))$ ， $i=1, \dots$ ，找到其密钥 k 使得 $c_i = E(k, m_i)$

引理：若DES为一个理想的密码（有 2^{56} 个随机可逆函数，将56 bits密钥映射到64 bits密文），则对于任给的明文与密文 m, c ，则有超过99.5%的概率有最多一个密钥 k 满足 $c = DES(k, m)$

证明： $\Pr[\exists k' \neq k, c = DES(k, m) = DES(k', m)] \leq \sum \Pr[DES(k, m) = DES(k', m)] \leq (2^{56}) * 2^{-64} = 1/256$

可能的密钥有 2^{56} ，可能的密文输出有 2^{64} ，求和即可

上述引理表明，对于DES而言，若对于给定的一对PT-CT消息对，其密钥几乎是完全确定的，即该消息对只有一个密钥能将PT映射到CT

而对于两对消息对，上述概率在DES算法中约为 $1-2^{-71}$ ，AES约为 $1-2^{-128}$
引理表明，两对消息对完全可以满足穷举攻击，问题在于如何找到上述密钥

2、DES challenge

曾经RSA公司发起过一个挑战，对于给定的PT-CT分组，求其对应的密钥，并用于解密后续消息c4, c5.....

msg = "The unknown messages is: XXXX ... "
CT = c₁ c₂ c₃ c₄

1997年花了3个月破解，之后更快，因此56 bits密钥不应再继续使用 (DES is completely dead)

3、Strengthening DES against exhaustive search

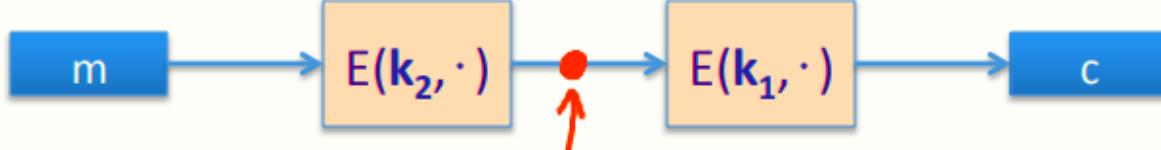
1. Triple-DES:

$3E(k_1, k_2, k_3, m) = E(k_1, D(k_2, E(k_3, m)))$ ，即DES重复运行三次（三个密钥不能一样，否则和DES没区别）

3DES密钥空间为 $3 \times 56 = 168$ bits，由于是运行三次，因此效率也为DES的 $1/3$

2. Why not double DES?

密钥长度为112 bits，加密算法为 $2E(k_1, k_2, m) = E(k_1, E(k_2, m))$ ，但很容易遭到中途相遇攻击，攻击者只需要找到密钥对 (k_1, k_2) ，满足 $E(k_2, m) = D(k_1, c)$ 即可（根据DES对称性可知）



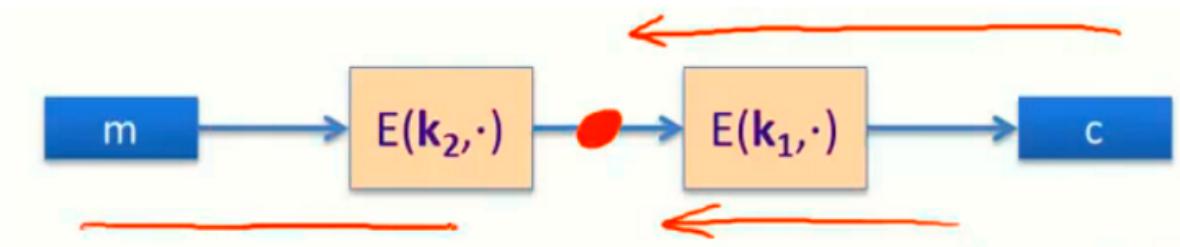
3. Meet in the middle attack:

经典的空间换时间的算法，利用了DES的对称性，极大程度减少了攻击时间开销

构建密钥表，如下图所示，包含56 bits全部密钥及其对应加密后的消息

$k^0 = 00\dots00$	$E(k^0, M)$
$k^1 = 00\dots01$	$E(k^1, M)$
$k^2 = 00\dots10$	$E(k^2, M)$
:	:
$k^N = 11\dots11$	$E(k^N, M)$

对于所有可能的 k 值 ($k \in \{0,1\}^{56}$)，计算 $D(k, c)$ 是否等于上表中第二列的值，若等于，则意味 $E(k_i, M) = D(k, C)$ ，即 $(k_i, k) = (k_2, k_1)$ ，从而找到2DES的碰撞 (collision)



时间开销约为 2^{63} , 空间开销为 2^{56} , 相同的攻击作用于3DES的时间开销会急剧增大到 2^{118} , 且对于3DES而言, 当计算找到了上述碰撞, 也意味着找到了3DES的三个密钥

4. DESX:

记E为n bits到n bits的块密码, 定义EX如下

$$EX((k_1, k_2, k_3), m) = k_1 \oplus E(k_2, m \oplus k_3)$$

由于是一个块密码与两次XOR计算, 因此效率损失不会太大

若EX中的块密码为DES, 则记为DESX, 其密钥长度为 $64+56+64=184$ bits (XOR需要与消息等长的64 bits, DES加密密钥为56 bits)

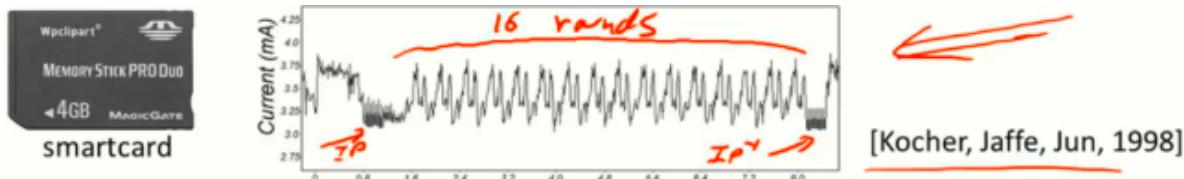
思考题: 注意到DESX在快密码的内部和外部均进行了XOR计算, 这是必须的, 若仅进行内部或外部的计算, 则其加密强度和原始的DES没有太大差别

W2 3-4 More Attacks on Block Ciphers

1、Attacks on the implementation

(1) 侧信道攻击:

通过测量加解密过程中需要的时间与设备功率, 如果加解密所花费的时间或功率取决于密钥位, 则攻击者可以了解到密钥的相关信息甚至完全提取出密钥



若测量设备足够精密, 将设备绘制的图表放大后甚至可以读取各个位的信息, 实验表明, 即便采取措施处理这些卡片, 期望掩盖这些信息, 效果仍然不理想

差分功率分析可以通过测量加密算法的许多设备参数(如电流电压等), 从中推算出密钥位之间的依赖关系, 只要加密算法运行足够长轮次, 就能反映出这种依赖关系从而提取密钥

多核处理器也可能遭遇攻击, 若加密算法与攻击者恰好分别运行于两个不同的内核, 而内核实际上共享缓存, 因此攻击者可以得知缓存中的未命中信息, 从而找到使用的密钥

(2) 故障攻击:

攻击者攻击智能卡时可以导致其故障(超频手段或在预热时导致)并输出错误的数据

实验结果表明, 加密过程的上一轮加密出现错误, 则产生的密文足以提取密钥信息

(3) 结论: 不应当使用自己发明的分组密码, 甚至不应当自己实现这些加密, 因为无法确保没有侧信道攻击和故障攻击, 应当使用标准库如OpenSSL等

2、Linear and differential attacks

通过线性密码分析, 期望能在小于 2^{56} 的时间内恢复密钥

(1) 线性密码分析:

$$\Pr[\underbrace{m[i_1] \oplus \dots \oplus m[i_r]}_{\text{subset of msg bits}} \oplus \underbrace{c[j_1] \oplus \dots \oplus c[j_v]}_{\text{subset of ciphered bits}} = \underbrace{k[l_1] \oplus \dots \oplus k[l_u]}_{\text{subset of key bits}}] = \frac{1}{2} + \epsilon$$

将消息的一个子集与密文的某个子集进行异或，之后与密钥k比较，如果m和c是完全独立的，则上述等式成立的概率为0.5，但实际上存在一个 ϵ 的偏差，即实际概率为 $0.5+\epsilon$

通过分析一些PT-CT对，利用上述公式可以确定一些密钥位的信息

对于DES而言， $\epsilon=2^{-21}$ ，因此需要 2^{42} 对随机的PT-CT，可以获取14 bits的密钥位，剩余 $56-41=42$ bits密钥需要 2^{42} 时间来暴力破解，因此将穷举攻击DES的时间降至 2^{43}

(2) 量子攻击:

对于一个普通的问题，有函数f，在大集合X中找到 $x \in X$ ，使得 $f(x)=1$ ，此类问题对于传统计算机算法需要的时间复杂度与X的规模相当（线性复杂度），但量子计算机复杂度为根号级别

W2 3-5 The AES block cipher

1、The AES process

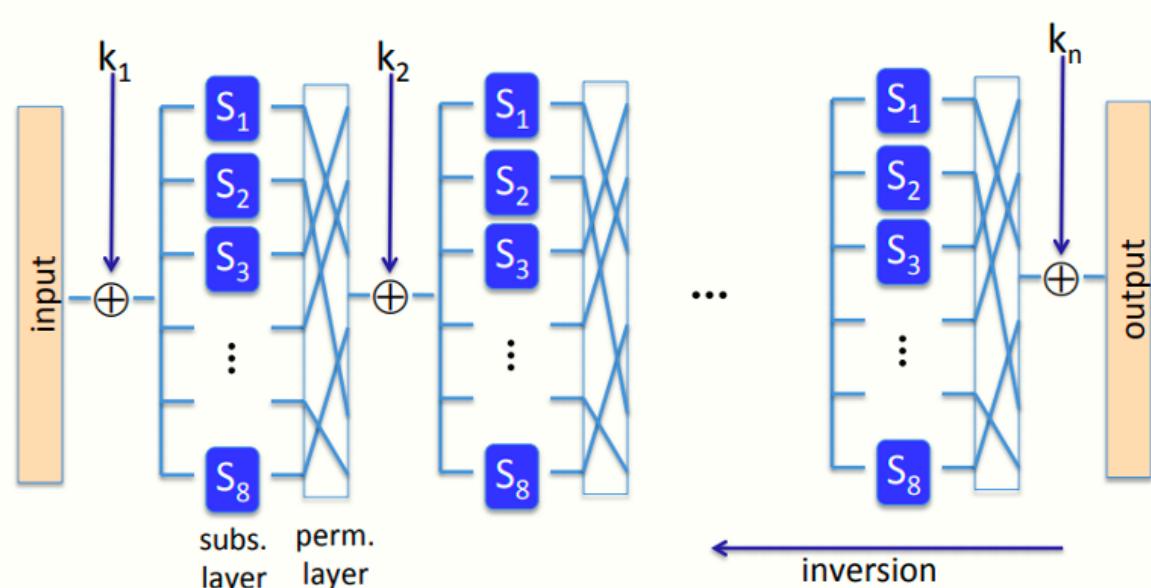
- 1997年NIST公开征求意见
- 1998年共提交15份提议
- 2000年最终选择Rijndael的算法作为AES
- AES分组长度128 bits，密钥长度128/192/256 bits
- 越长的密钥安全性越高，但也意味着效率越低

2、Subs-Perm network

AES基于代换-置换网络 (Substitution-Permutation network) 而非Feistel网络

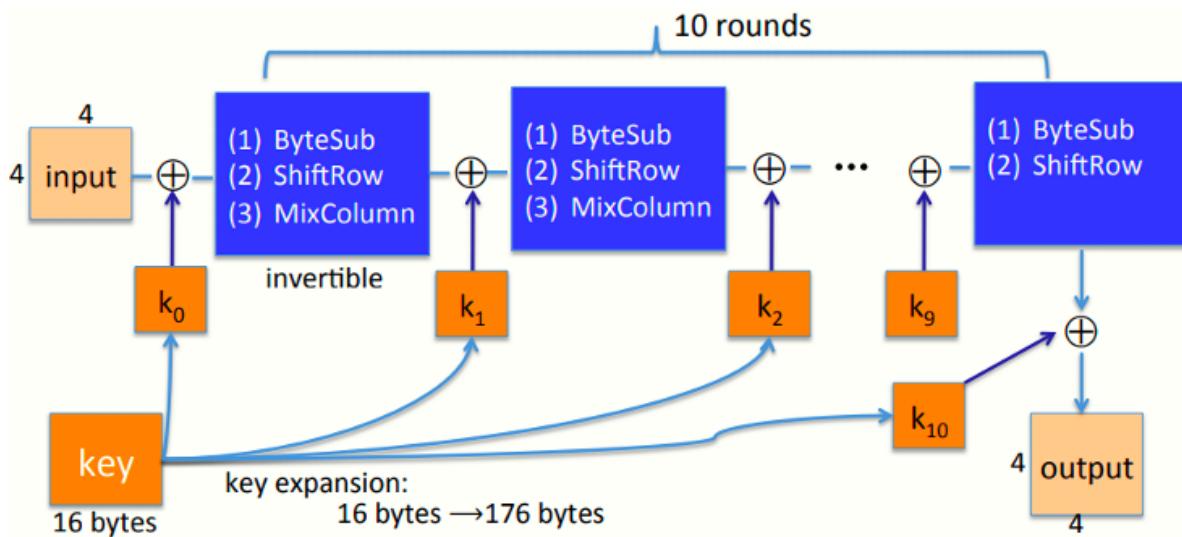
两者的区别在于Feistel网络中每轮计算时分组有一半的位不会改变，直接作为下一轮计算的输入，而S-P网络中每轮计算中分组的每一位均会发生变化

S-P过程如下



由于S-P网络的构造方式，其每一步都是可逆的，因此整个过程是可逆的，由原来的输出想要得到原来的输入，则需要将上述流程完全逆用

3、AES-128 schematic

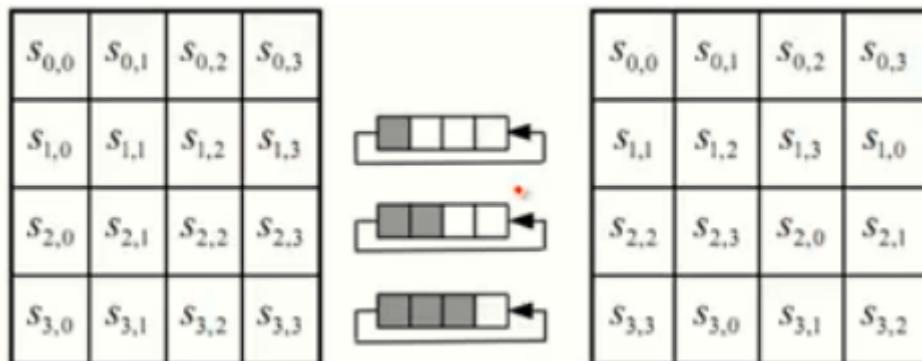


总体流程如上图，先将128 bits的分组划分成16个字节，按顺序排列成 4×4 的矩阵，将轮密钥 k_i 与矩阵异或，然后经过可逆部分，得到本轮输出

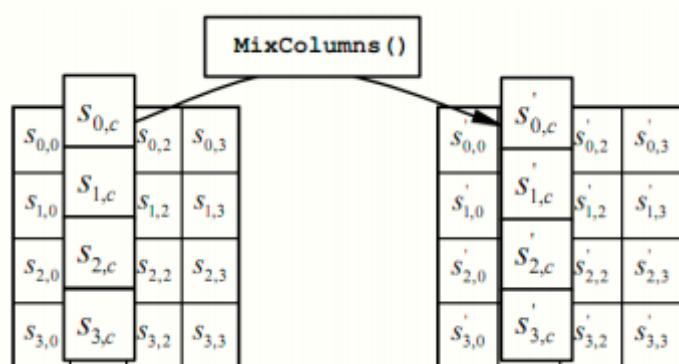
轮密钥由128 bits的初始密钥经过密钥扩展得到11个子密钥，每个子密钥均排列成 4×4 的矩阵以便于和消息矩阵XOR计算

4、The round function

- (1) 字节替换：通过一个S盒（包含256字节），以消息矩阵作为输入，得到输出
- (2) 行移位：第*i*行的所有字节循环左移*i*个字节 ($i=0, 1, 2, 3$)，如图示



(3) 列混合：特殊的线性变换，使用一个特定矩阵与当前矩阵相乘，对于这些线性变换而言，所有的列都是相互独立的



5、AES in hardware

Intel处理器内建了对AES的支持，通过特殊的汇编指令来加速AES算法

6、Attacks

(1) 密钥恢复：如果期望以恢复密钥的手段攻破AES，其效率仅仅比穷举快四倍而已，并没有显著影响其安全性

(2) 相关密钥攻击：针对AES-256的攻击，发现了其在密钥扩展中的缺陷，导致相关密钥攻击

相关密钥攻击：如果有 2^{100} 对AES的PT-CT对，这些消息对均使用四个紧密相关的密钥（如任意两个密钥之间仅有1 bit不同，即汉明距离很近），此时破解复杂度为 2^{100} （远小于穷举的 2^{256} ）

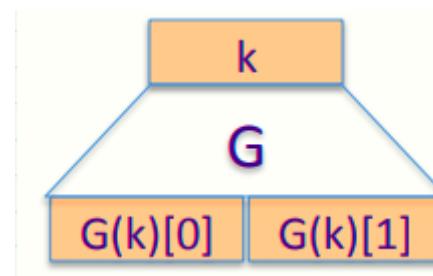
因此为了防止类似攻击，应确保密钥的随机性而将其局限于某个小范围（尽管这个局限性并不显著，因为 2^{100} 仍然很大）

W2 3-6 Block ciphers from PRGs

1、Can we build a PRF from a PRG?

记 $G: K \rightarrow K^2$ 为一安全PRG

定义一个1 bit PRF F如下， $F: K \times \{0,1\} \rightarrow K$ as $F(k, x \in \{0,1\}) = G(k)[x]$



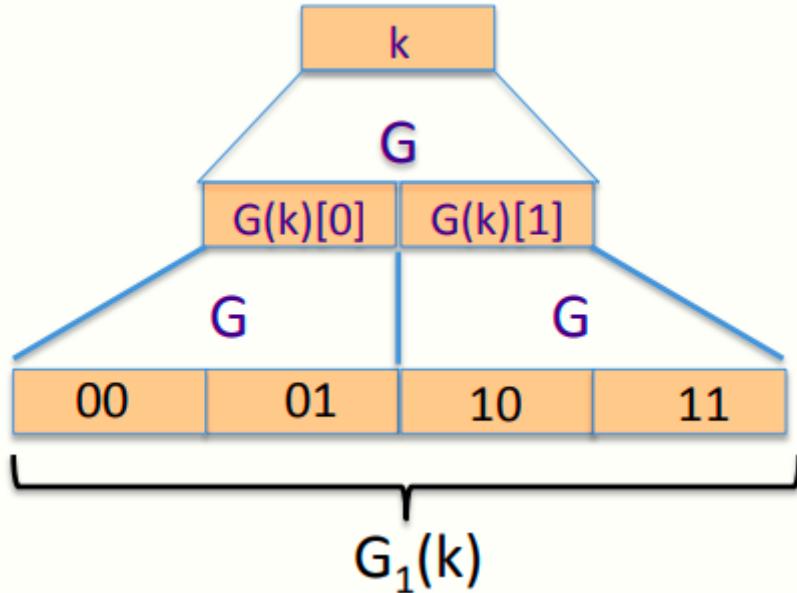
引理：若G为一安全PRG，则F为一安全PRF

2、Extending a PRG

记 $G: K \rightarrow K^2$ 为一安全PRG

定义 $G_1: K \rightarrow K^4$ as $G_1(k) = G(G(k)[0]) \parallel G(G(k)[1])$

因此得到一个2 bits PRF： $F(k, x \in \{0,1\}^2) = G_1(k)[x]$

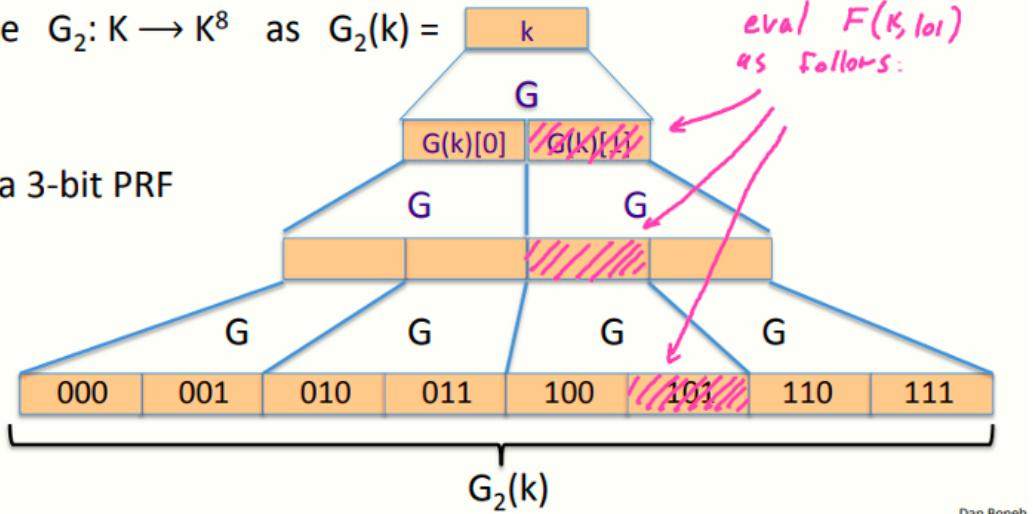


3、Extending more

Let $G: K \rightarrow K^2$.

define $G_2: K \rightarrow K^8$ as $G_2(k) =$

We get a 3-bit PRF



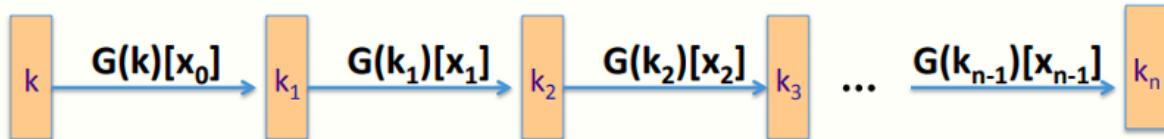
Dan Boneh

套娃

4、Extending even more: the GGM PRF

Let $G: K \rightarrow K^2$. define PRF $F: K \times \{0,1\}^n \rightarrow K$ as

For input $x = x_0 x_1 \dots x_{n-1} \in \{0,1\}^n$ do:

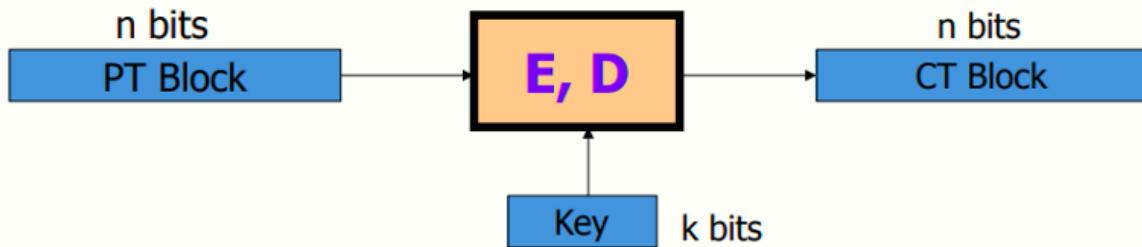


Security: G a secure PRG \Rightarrow F is a secure PRF on $\{0,1\}^n$.

继续套娃，但实际上不应使用这种方式（效率太低）

W2 4-1 Review: PRPs and PRFs

1、Block ciphers: crypto work horse



2、Abstractly: PRPs and PRFs

(1) Pseudo Random Function (PRF) 定义为一三元组 $(K, X, Y); F: K \times X \rightarrow Y$ ，且存在一有效算法以评价 $F(k, x)$

(2) Pseudo Random Permutation (PRP) 定义为一二元组 $(K, X); E: K \times X \rightarrow X$ 满足：

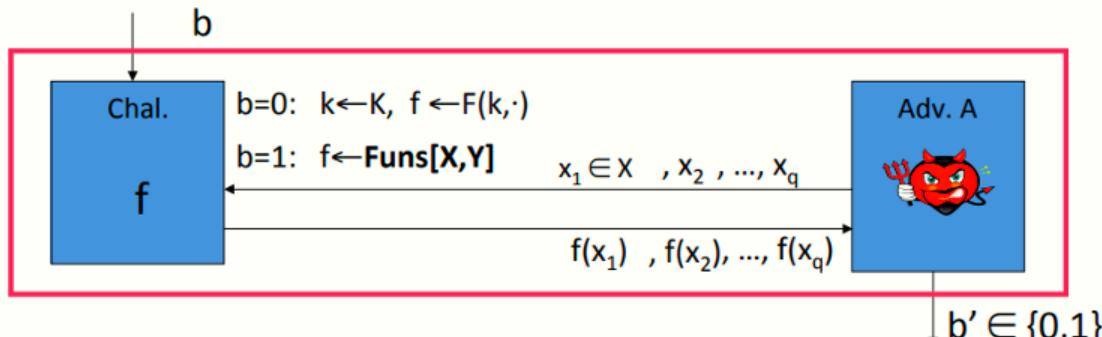
- 存在一确定性算法评价 $E(k, x)$
- 函数 $E(k, \cdot)$ 为一对一映射函数（双射函数）
- 存在对应的有效的逆运算 $D(k, x)$

3、Secure PRFs

详见3-1

4、Secure PRF: definition

- For $b=0,1$ define experiment $\text{EXP}(b)$ as:



- Def: F is a secure PRF if for all “efficient” A :

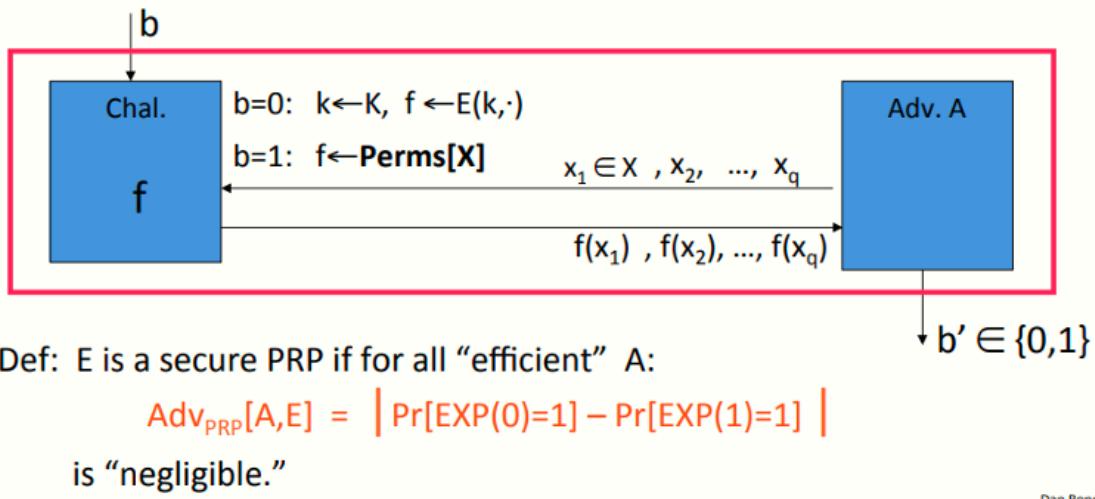
$$\text{Adv}_{\text{PRF}}[A, F] := |\Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1]|$$

is “negligible.”

Dan Boneh

5、Secure PRP (secure block cipher)

- For $b=0,1$ define experiment $\text{EXP}(b)$ as:



- Def: E is a secure PRP if for all “efficient” A :

$$\text{Adv}_{\text{PRP}}[A, E] = |\Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1]|$$

is “negligible.”

6、PRF Switching Lemma

PRP交换引论：对于 $|X|$ 足够大的情况下，一个安全的PRP同时也是个安全的PRF

引理： E 为定义在二元组(K, X) 上的PRP，对任意的有最多Q个查询的攻击者A，满足如下不等式

$$|\text{Adv}_{\text{PRF}}[A, E] - \text{Adv}_{\text{PRP}}[A, E]| < q^2 / 2|X|$$

若 $|X|$ 很大，则不等式右侧为一可忽略数，则由PRP的安全性可知，PRP的优势可忽略，则得到PRF的优势亦可忽略

W2 4-2 Modes of operation: one time key

1、Using PRPs and PRFs

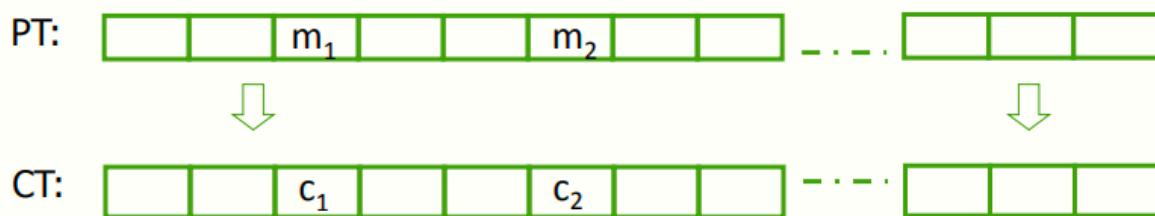
目的：通过安全的PRP构建安全的加密方案，本例中旨在使用块密码来使用一次性密钥来加密

攻击者的能力：只能看到一次性密钥加密后的密文

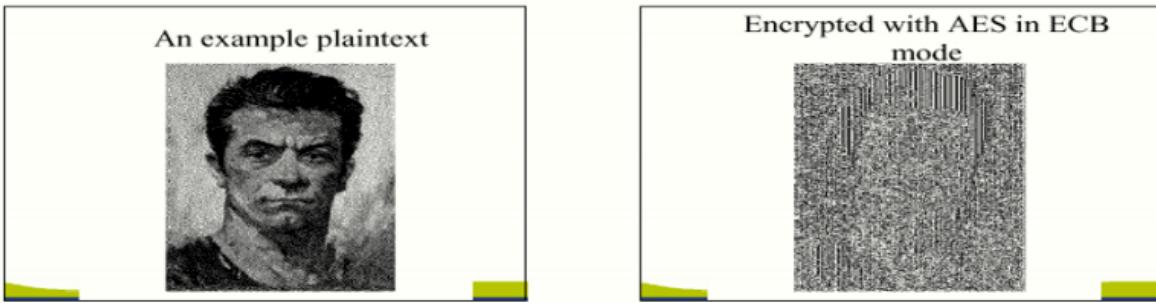
攻击者目标：从CT中提取PT信息（即破坏语义安全）

2、Incorrect use of a PRP

ECB模式，流程如下

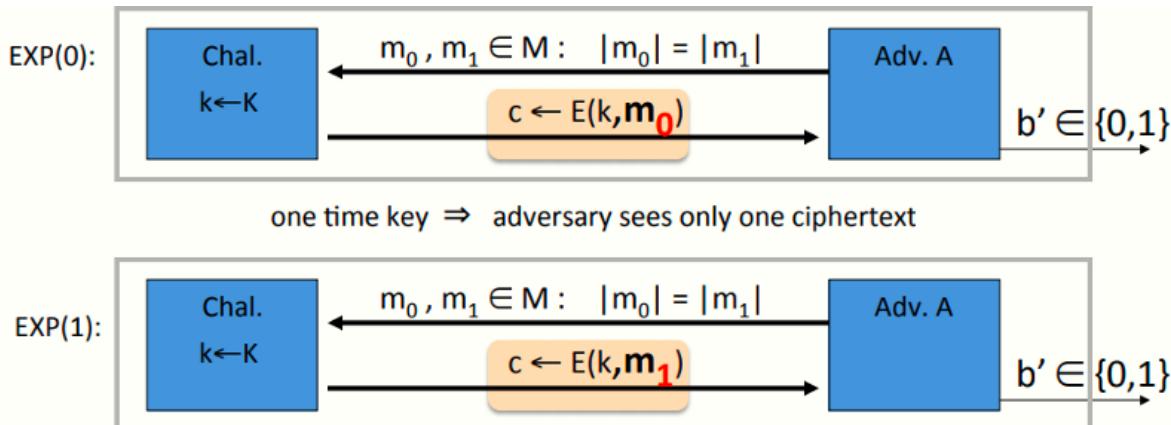


隐性问题：若消息 $m_1=m_2$ ，则加密后的 $c_1=c_2$ ，从而攻击者可以获取一些明文之间的关系，而这些关系不应反应在密文上



若将该模式用于加密图片信息，则可能得到如图结果，尽管没有暴露所有的信息，但是仍能反应出一些人物轮廓

3、Semantic Security (one-time key)



$$\text{Adv}_{\text{SS}}[\text{A}, \text{OTP}] = |\Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1]| \text{ should be "neg."}$$

对于OTP而言，攻击者应只能看见CT，因此若要做到语义安全，上述优势 $\text{Adv}_{\text{SS}}[\text{A}, \text{OTP}]$ 应可忽略

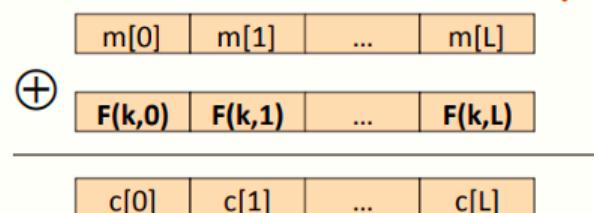
4、ECB is not Semantically Secure

ECB并不是语义安全的，因此ECB模式不应加密超过一个块的信息

5、Secure Construction I

Deterministic counter mode from a PRF $F: \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$

$$\bullet E_{\text{DETCTR}}(k, m) = \quad (\text{e.g. } n=128)$$



6、Deterministic counter-mode security

定理：对于任给的 $L > 0$ ，若 F 为定义在三元组 (K, X, X) 上的PRF， E_{DETCTR} 为定义在三元组 (K, X^L, X^L) 上语义安全的密码

特别地，对于任意高效的攻击者攻击 E_{DETCTR} ，存在一高效的PRF攻击者 B ，使得：

$$Adv_{SS}[A, E_{DETCTR}] = 2 \cdot Adv_{PRF}[B, F]$$

Theorem: For any $L > 0$,

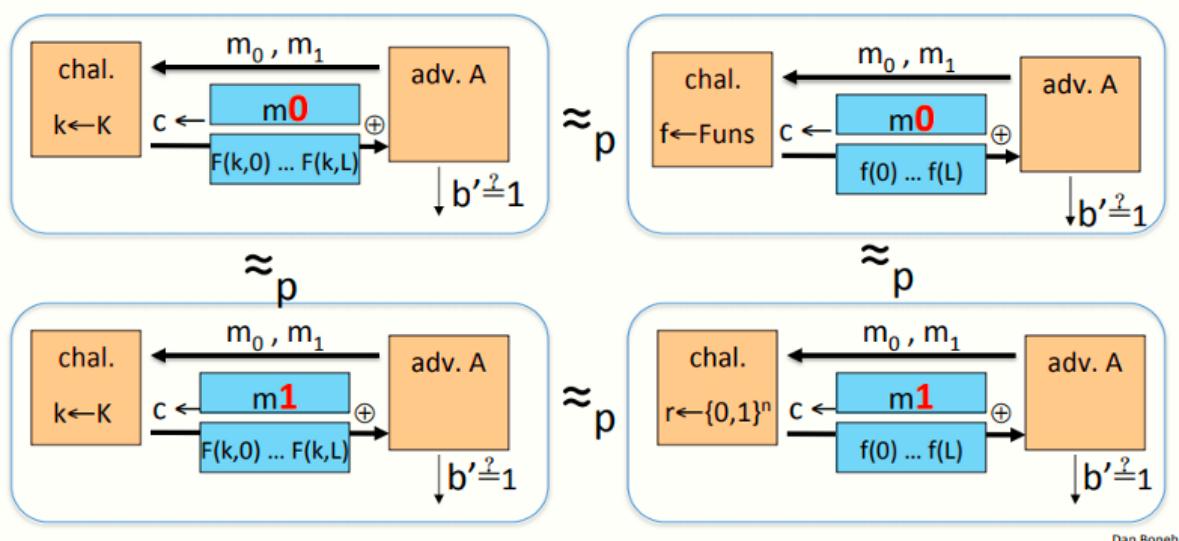
If F is a secure PRF over (K, X, X) then

E_{DETCTR} is sem. sec. cipher over (K, X^L, X^L) .

In particular, for any eff. adversary A attacking E_{DETCTR} there exists an eff. PRF adversary B s.t.:

$$Adv_{SS}[A, E_{DETCTR}] = 2 \cdot Adv_{PRF}[B, F]$$

证明如下：



W2 4-3 Security for many-time key

1、Example application

- 文件系统：用AES，以相同的密钥加密多个文件
- IPsec：AES，相同的密钥加密多个数据包

2、Semantic Security for many-time key

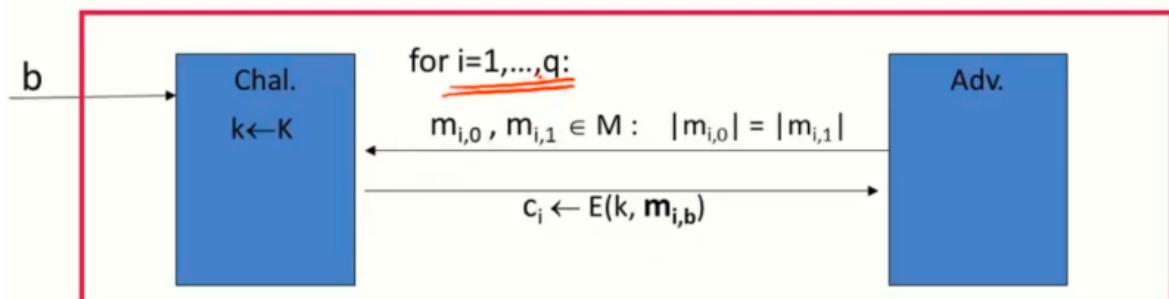
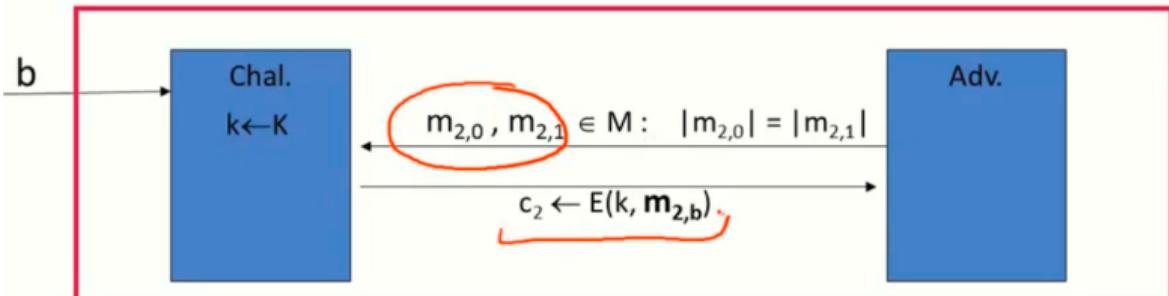
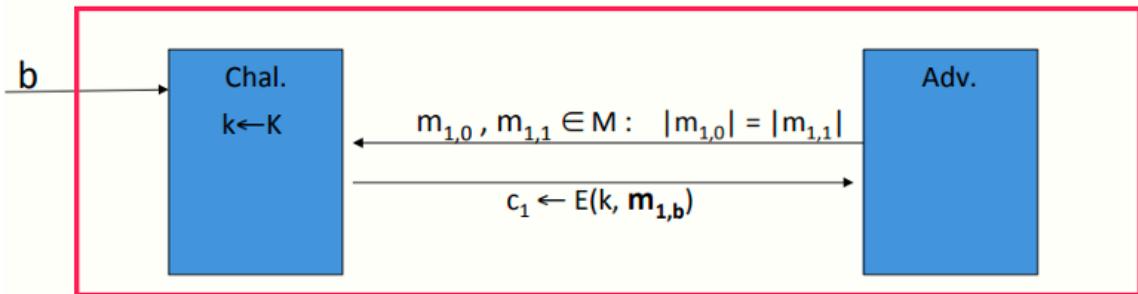
重用密钥意味着攻击者可以获得由同一密钥加密的多个密文段

攻击者的能力：选择明文攻击 (chosen-plaintext attack, CPA)，意味着攻击者可以截获其选择的信息的对应密文段（日常通信者非常常见）

攻击者的目标：破坏语义安全

3、Semantic Security for many-time key

$E = (E, D)$ a cipher defined over (K, M, C) . For $b=0, 1$ define $\text{EXP}(b)$ as:



上述流程和标准语义安全非常类似，但是攻击者可以不断地发起这种查询（假设其最多可以发起 q 次，即 $i=1, 2, \dots, q$ ）

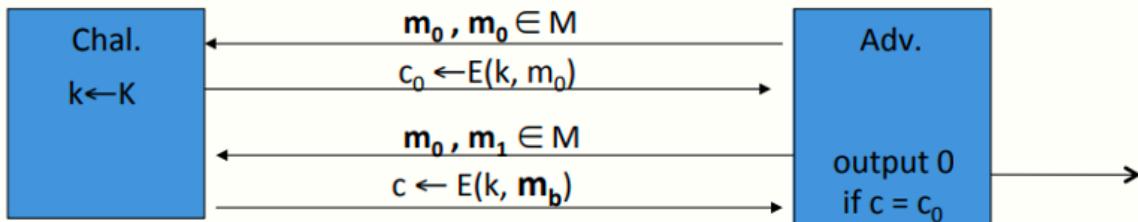
在CPA中，攻击者可以使 $m_0=m_1=m$ ，即让两个消息一致，进而像挑战者提交后得到 m 的密文，CPA中两端消息是相同的（标准语义安全不相同）

定义：对于所有的高效的算法 A ，若在CPA下为语义安全的，则如下等式的优势为可忽略的

$$\text{Adv}_{\text{CPA}}[A, E] = |\Pr[\text{EXP}(0) = 1] - \Pr[\text{EXP}(1) = 1]|$$

4、Ciphers insecure under CPA

假设 $E(k, m)$ 对于相同的明文消息总是输出相同的密文，则有如下模型



首先攻击者发起攻击，选择两段相同的消息 $m=m_0$ 发给挑战者，无论挑战者选择 $b=0$ 或 1 ，其均会返回消息 m_0 的密文 c_0

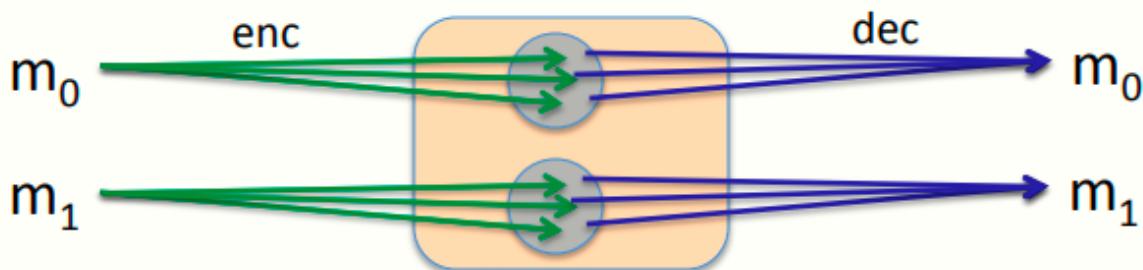
之后攻击者再发起一次攻击，选择不同的消息 m_0 和 m_1 ，之后挑战者返回消息时，攻击者仅需判断收到的密文 c 是否等于 c_0 即可

综上模型，攻击者区分 m_0 和 m_1 的优势为1

结论：相同的明文加密成相同的密文（尽管不能知道明文的任意信息），这个机制不应被攻击者知道，因此确定性加密机制不能再CPA下保持语义安全

5、Solution 1: randomized encryption

$E(k,m)$ 为一随机算法



对于加密算法，消息 m_0 映射到一个密文空间内的多条不同的密文而非一条，使得每次加密算法都能得到不同的输出而非相同的输出，即这种随机化算法使得每次加密相同的消息均有极高概率得到与之前不同的输出

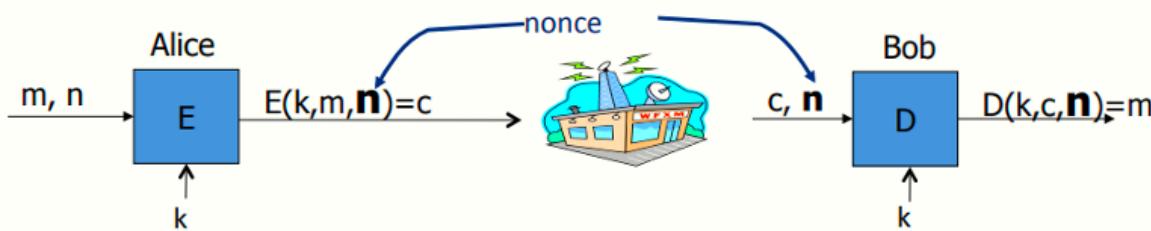
对于不同的消息 $m_1 \neq m_0$ ，其映射的密文空间必须是无交集的

但是也意味着密文一定比明文长，是因为用于生成密文的随机字符串必然要以某种方式编码到密文中从而占据一些空间，如果消息很长（GB级别）则可忽略这些额外的消息，如果只有若干字节的消息，加密后意味着消息长度翻倍

6、Solution 2: nonce-based Encryption

基于nonce的加密系统算法接收三个输入参数，即比传统加密算法多接收一个nonce作为输入，对应的解密算法也有这个参数

Nonce: Number used once，算法中仅使用一次的任意或随机的数，该值公开无需保密，仅需要确保只使用一次

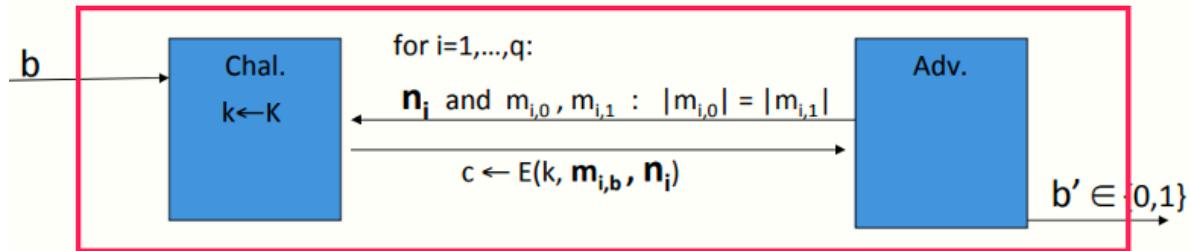


Nonce的选择：

- 将Nonce作为一个计数器（如数据包计数器），每次计算或每发送一个包就递增，加密算法和解密算法需要同步计数器的状态来确保解密正确，好处在于没有必要将nonce包含在密文中（如HTTPS或IPsec协议等）
- 随机选择Nonce，若Nonce空间足够大，则可以确保在密钥生命周期内使用的Nonce高概率不会重复，则这种情况下基于nonce的加密转化为随机化加密，好处是发送方不必记录各条消息的状态（对不同设备之间的加密非常有用，如电脑手机打印机之间的通信）

7、CPA security for nonce-based encryption

即便系统的nonce为攻击者选择时，系统也必须保证安全性（因为允许攻击者选择nonce意味着攻击者可以选择其企图破解的密文）



该模型与之前的模型类似，但要求攻击者每次查询时给出对应的nonce值 n_i ，且对于最多有 q 次查询的攻击者而言，这 q 个nonce值必须不同（实际中攻击者诱骗通信方加密多段消息时，通信方也不会使用重复的nonce，攻击者也不会收到由相同的nonce加密后的消息）

若在上述条件下，即基于nonce的加密，算法A的优势（如下）仍为可忽略的，则为CPA下的语义安全

$$Adv_{nCPA}[A, E] = |\Pr[EXP(0) = 1] - \Pr[EXP(1) = 1]|$$

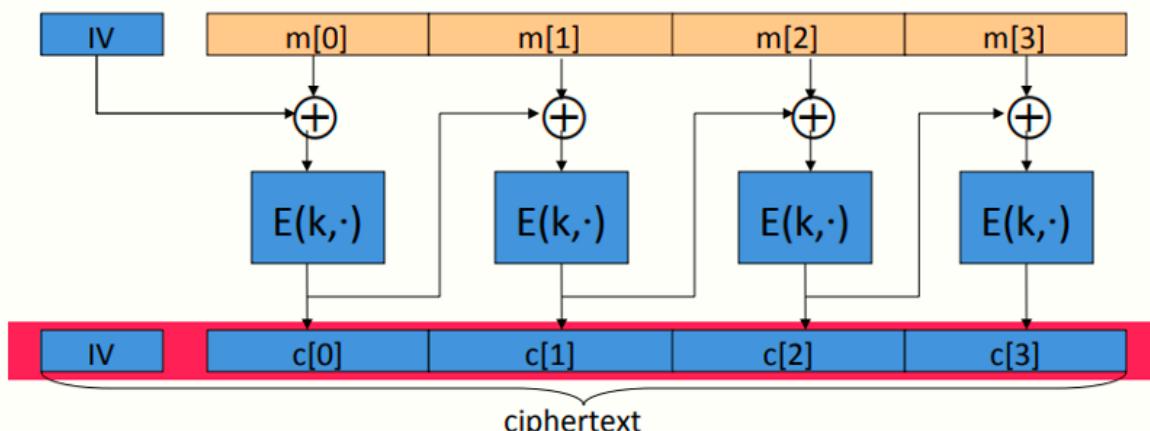
Def: nonce-based E is sem. sec. under CPA if for all “efficient” A :

$$Adv_{nCPA}[A, E] = |\Pr[EXP(0)=1] - \Pr[EXP(1)=1]| \text{ is “negligible.”}$$

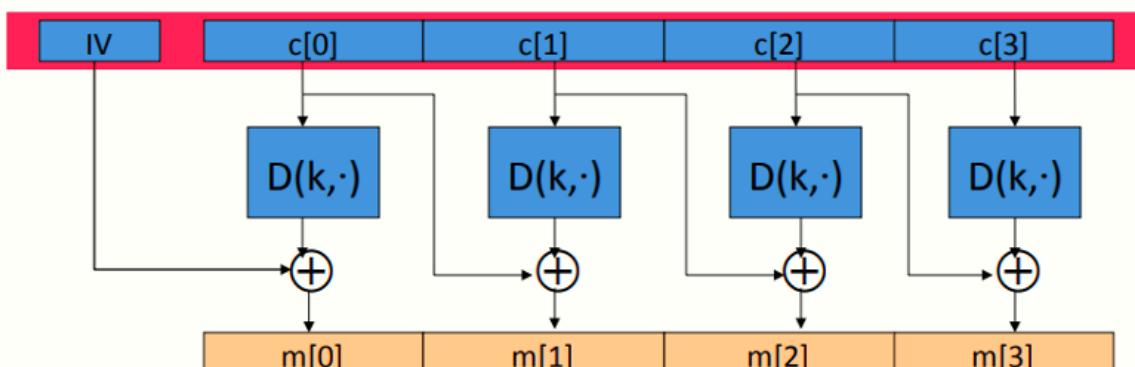
W2 4-4 Modes of operation: many time key (CBC)

1、Construction 1: CBC with random IV

记 (E, D) 为一块密码， E -CBC (k, m) 首先选择一随机IV（与加密的消息块等长），之后完成如下流程图



2、Decryption circuit



加密流程反过来操作

注意到CBC解密的一个特点，若某一块在传输或保存时损坏了，

3、CBC: CPA Analysis

CBC定理：对于任意的 $L > 0$, 若 E 为一定义在 (K, X) 上的安全PRP，则 E_{CBC} 为一定义在 (K, X^L, X^{L+1}) 上在CPA下语义安全的

具体来说，对于一个 q 次请求的攻击者A要攻击 E_{CBC} ，则存在一个PRP攻击者B，满足如下不等式

$$Adv_{CPA}[A, E_{CBC}] \leq 2Adv_{PRP}[B, E] + 2q^2 L^2 / |X|$$

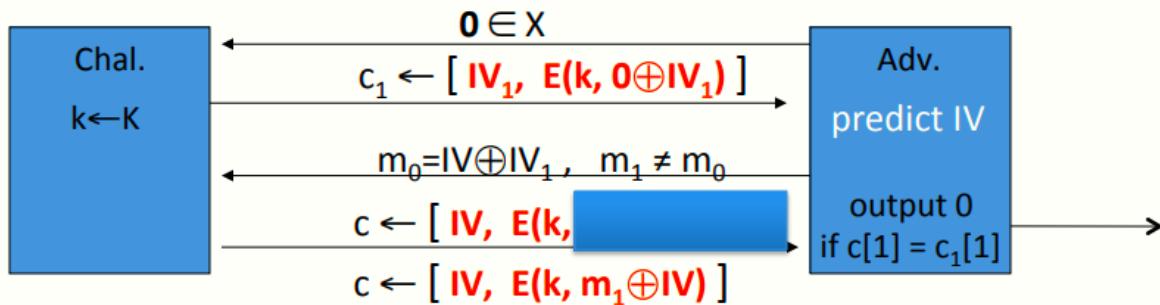
CBC只有当 $(qL)^2 \ll |X|$ 才安全，即使用同一密钥 k 加密的消息个数 q 和消息长度 L 的积的平方应远小于 $|X|$

对于AES而言，消息空间 $|X| = 2^{128}$ ，则 qL 应小于 2^{48} ，这表明在AES加密 2^{48} 块消息后，必须更换密钥，而对于3DES而言， qL 小于 2^{16}

4、Warning: an attack on CBC with rand. IV

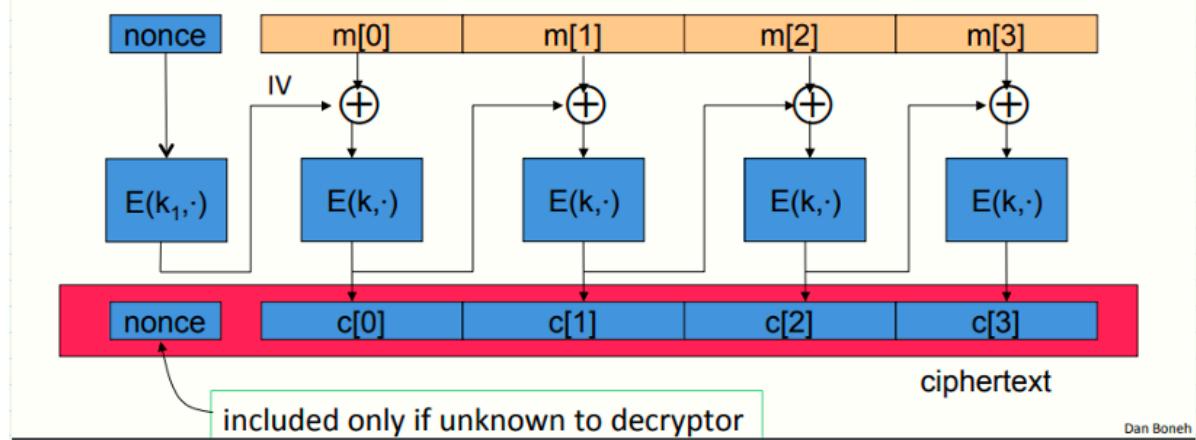
若一个CBC的初始向量IV可被攻击者预测，则意味着他不是一个CPA安全的加密方式

假设对于给定的 $E_{CBC}(k, m)$ 的密文 c ，攻击者可以预测下一条消息使用的IV，则有如下语义安全挑战模型



5、Construction 1: nonce-based CBC

- Cipher block chaining with unique nonce: key = (k, k_1)
unique nonce means: (key, n) pair is used for only one message



如图所示，原来用于加密的初始向量IV被替换成nonce，若通信的接收方知道nonce，则可以不在密文中包含nonce，此时密文长度与明文一致（使用随机的IV则需要在密文中附带IV，导致密文变长）

但若使用基于nonce的CBC加密模式，需要两个独立的密钥（或一对密钥对） (k, k_1) ，其中 k 为原来用于加密各个明文分组的密钥，而 k_1 为用于加密nonce的密钥

k_1 非常关键，若不使用 k_1 ，则CBC模式不再安全

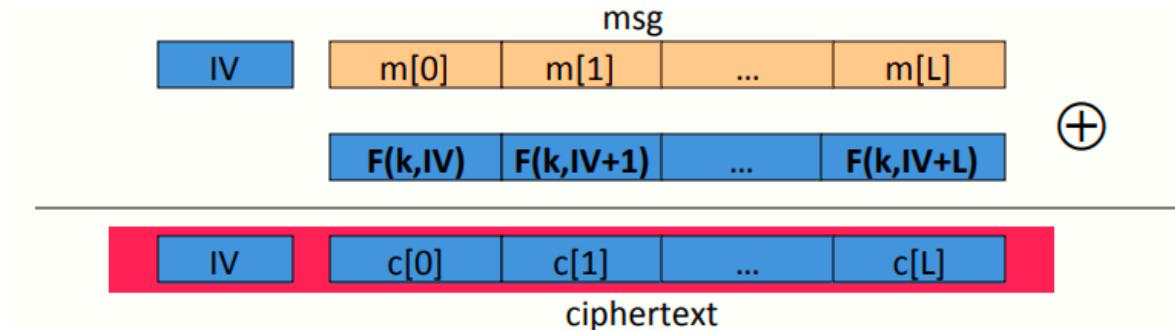
如上图所示，nonce先使用 k_1 进行加密，然后将结果作为初始向量IV

W2 4-5 Modes of operation: many time key (CTR)

1、Construction 2: randomized ctr-mode

和CBC不同的是，随机计数器使用的是PRF

记F为一安全PRF，则有如下模型



首先选择一随机的初始向量IV，每次传入轮函数F时都将IV+1，且IV需要附在密文消息内

IV的选择：每次都需要随机选择新的IV，即便是将同一消息加密两次（从而得到不同的加密结果）

与CBC不同的是，CTR模式是完全并行的，而CBC是串行的

2、rand ctr-mode (rand. IV): CPA analysis

计数器模式：对于任意消息长度 $L > 0$ ，若轮函数F为一定义在 (K, X, X) 上的安全PRF，则E-CTR为一定义在 (K, X^L, X^{L+1}) 上的在CPA下语义安全的模式

具体来说，若对于至多有 q 次查询的攻击者A而言，存在一PRF攻击者B，使得满足如下不等式

$$Adv_{CPA}[A, E_{CTR}] \leq 2Adv_{PRF}[B, F] + 2q^2 L / |X|$$

注意到CTR模式要求 $q^2 L \ll |X|$ ，比CBC模式更好，对于AES而言，CBC模式加密 2^{48} 块消息后需要更换密钥，而这个值在CTR为 2^{64}

3、Comparison: ctr vs. CBC

	CBC	ctr mode
uses	PRP	PRF
parallel processing	No	Yes
Security of rand. enc.	$q^2 L^2 \ll X $	$q^2 L \ll X $
dummy padding block	Yes	No
1 byte msgs (nonce-based)	16x expansion	no expansion

4、Summary

PRF和PRP都是块密码中常用的抽象概念

两种安全观念（都只提供防止窃听的安全措施，不提供防篡改密文的安全措施），一个仅用于加密单个消息时使用（流密码），另一个用于加密多个消息使用（CBC或CTR），均不提供完整性检验

W2 知识梳理

W2 Problem Set && Programming Assignment

Q1

1. Consider the following five events:

1. Correctly guessing a random 128-bit AES key on the first try.
2. Winning a lottery with 1 million contestants (the probability is $1/10^6$).
3. Winning a lottery with 1 million contestants 5 times in a row (the probability is $(1/10^6)^5$).
4. Winning a lottery with 1 million contestants 6 times in a row.
5. Winning a lottery with 1 million contestants 7 times in a row.

What is the order of these events from most likely to least likely?

- 3, 2, 5, 4, 1
- 2, 3, 4, 1, 5
- 2, 3, 1, 5, 4
- 2, 3, 5, 4, 1

问：上述五个事件中，按发生的概率从大到小排序，正确的顺序为？

分析：事件1的概率为 $1/2^{128}$ ，事件4的概率为 $1/10^{36} \approx 1/2^{119.5}$ ，事件5的概率为 $1/10^{42} \approx 1/2^{139}$ ，因此5的概率要比1小更多

Q2

2. Suppose that using commodity hardware it is possible to build a computer for about \$200 that can brute force about 1 billion AES keys per second. Suppose an organization wants to run an exhaustive search for a single 128-bit AES key and was willing to spend 4 trillion dollars to buy these machines (this is more than the annual US federal budget). How long would it take the organization to brute force this single 128-bit AES key with these machines? Ignore additional costs such as power and maintenance.

- More than a week but less than a month
- More than a year but less than 100 years
- More than a 100 years but less than a million years
- More than a month but less than a year
- More than a billion (10^9) years

问：若一种硬件可以做到每秒暴力搜索AES的密钥10亿次，且一个这种硬件需要200块钱，现有预算4万亿买这种硬件，不考虑除购买硬件以外的其他费用，破解一个128 bits的AES密钥需要多久？

分析：简单的计算题，根据题意，能买 $2 * 10^{10}$ 个硬件，每秒计算速度总共为 $10^9 * (2 * 10^{10}) = 2 * 10^{19}$ ，需要共计 $2^{128} / (2 * 10^{19}) = 1.7 * 10^{19}$ 秒，大约是5400亿年

Q3

3. Let $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a secure PRF

(i.e. a PRF where the key space, input space, and output space are all $\{0,1\}^n$) and say $n = 128$.

Which of the following is a secure PRF (there is more than one correct answer):

$F'(k, x) = k \oplus x$

$F'(k, x) = \begin{cases} F(k, x) & \text{when } x \neq 0^n \\ 0^n & \text{otherwise} \end{cases}$

$F'(k, x) = F(k, x \oplus 1^n)$

$F'((k_1, k_2), x) = F(k_1, x) \parallel F(k_2, x)$ (here \parallel denotes concatenation)

$F'((k_1, k_2), x) = \begin{cases} F(k, x) & \text{when } x \neq 0^n \\ k & \text{otherwise} \end{cases}$

$F'((k_1, k_2), x) = F(k_1, x) \oplus F(k_2, x)$

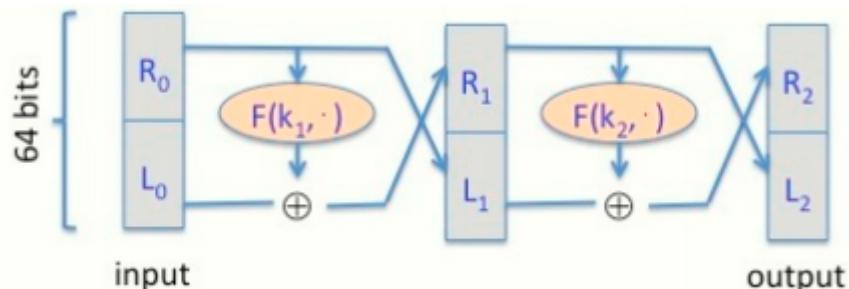
Q4

4. Recall that the Luby-Rackoff theorem discussed in [The Data Encryption Standard lecture](#) states that applying a **three** round Feistel network to a secure PRF gives a secure block cipher. Let's see what goes wrong if we only use a **two** round Feistel.

Let $F : K \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$ be a secure PRF.

Recall that a 2-round Feistel defines the following PRP

$$F_2 : K^2 \times \{0,1\}^{64} \rightarrow \{0,1\}^{64}:$$



Here R_0 is the right 32 bits of the 64-bit input and L_0 is the left 32 bits.

One of the following lines is the output of this PRP F_2 using a random key, while the other three are the output of a truly random permutation $f : \{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$. All 64-bit outputs are encoded as 16 hex characters.

Can you say which is the output of the PRP? Note that since you are able to distinguish the output of F_2 from random, F_2 is not a secure block cipher, which is what we wanted to show.

Hint: First argue that there is a detectable pattern in the xor of $F_2(\cdot, 0^{64})$ and $F_2(\cdot, 1^{32}0^{32})$. Then try to detect this pattern in the given outputs.

- On input 0^{64} the output is "9f970f4e 932330e4".
On input $1^{32}0^{32}$ the output is "6068f0b1 b645c008".
- On input 0^{64} the output is "7c2822eb fdc48bfb".
On input $1^{32}0^{32}$ the output is "325032a9 c5e2364b".
- On input 0^{64} the output is "4af53267 1351e2e1".
On input $1^{32}0^{32}$ the output is "87a40cf8 8dd39154".
- On input 0^{64} the output is "2d1cfa42 c0b1d266".
On input $1^{32}0^{32}$ the output is "eea6e3dd b2146dd0".

问：在两轮Feistel网络中会出现安全问题，下列四个输出中有一个为使用随机密钥的PRP F_2 的输出，其余三个为真随机替换函数f的输出，问哪个是 F_2 的输出

提示：将两个值异或

第一个选项中9f970f4e932330e4与6068f0b1b645c008异或，结果的前32 bits为0xFFFF FFFF

Q5

5. Nonce-based CBC. Recall that in [Lecture 4.4](#) we said that if one wants to use CBC encryption with a non-random unique nonce then the nonce must first be encrypted with an **independent** PRP key and the result then used as the CBC IV.

Let's see what goes wrong if one encrypts the nonce with the **same** PRP key as the key used for CBC encryption.

Let $F : K \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ be a secure PRP with, say, $\ell = 128$. Let n be a nonce and suppose one encrypts a message m by first computing $IV = F(k, n)$ and then using this IV in CBC encryption using $F(k, \cdot)$. Note that the same key k is used for computing the IV and for CBC encryption. We show that the resulting system is not nonce-based CPA secure.

The attacker begins by asking for the encryption of the two block message $m = (0^\ell, 0^\ell)$ with nonce $n = 0^\ell$. It receives back a two block ciphertext (c_0, c_1) . Observe that by definition of CBC we know that $c_1 = F(k, c_0)$.

Next, the attacker asks for the encryption of the one block message $m_1 = c_0 \oplus c_1$ with nonce $n = c_0$. It receives back a one block ciphertext c'_0 .

What relation holds between c_0, c_1, c'_0 ? Note that this relation lets the adversary win the nonce-based CPA game with advantage 1.

- $c_1 = c'_0$
 $c'_0 = c_0 \oplus 1^\ell$
 $c_0 = c_1 \oplus c'_0$
 $c_1 = c_0 \oplus c'_0$

问：基于nonce的CBC模式需要使用独立的密钥先将nonce加密，之后将加密结果作为CBC模式的IV，若在CBC模式中使用与加密nonce相同的密钥则会发生什么？假设.....懒得翻译了

分析：一顿计算可知选第一个

Q6

6. Let m be a message consisting of ℓ AES blocks
(say $\ell = 100$). Alice encrypts m using CBC mode and transmits
the resulting ciphertext to Bob. Due to a network error,
ciphertext block number $\ell/2$ is corrupted during transmission.
All other ciphertext blocks are transmitted and received correctly.
Once Bob decrypts the received ciphertext, how many plaintext blocks
will be corrupted?

- ℓ
- 3
- $\ell/2$
- 0
- 2

问：Alice向Bob传输AES的加密块，使用CBC模式加密，若某一块在传输过程中出错，其他块均正确接收，Bob解密时有多少明文块会解密出错

分析：由CBC模式的特性知道，密文块出错时，解密只会影响本块和下一块的解密正确性，其余块不受影响，因此为2块

Q7

7. Let m be a message consisting of ℓ AES blocks (say $\ell = 100$). Alice encrypts m using randomized counter mode and transmits the resulting ciphertext to Bob. Due to a network error, ciphertext block number $\ell/2$ is corrupted during transmission. All other ciphertext blocks are transmitted and received correctly. Once Bob decrypts the received ciphertext, how many plaintext blocks will be corrupted?

- $\ell/2$
- 0
- 1
- 3
- $1 + \ell/2$

问：Alice向Bob传输AES的加密块，使用CTR模式加密，若某一块在传输过程中出错，其他块均正确接收，Bob解密时有多少明文块会解密出错

分析：由于CTR模式特性可知，CTR模式每块加解密均独立，因此只有一块受影响

Q8

8. Recall that encryption systems do not fully hide the **length** of

transmitted messages. Leaking the length of web requests [has been used](#) to eavesdrop on encrypted HTTPS traffic to a number of

web sites, such as tax preparation sites, Google searches, and healthcare sites.

Suppose an attacker intercepts a packet where he knows that the packet payload is encrypted using AES in CBC mode with a random IV. The encrypted packet payload is 128 bytes. Which of the following messages is plausibly the decryption of the payload:

- 'The most direct computation would be for the enemy to try all 2^r possible keys, one by one.'
- 'To consider the resistance of an enciphering process to being broken we should assume that at some times the enemy knows everything but the key being used and to break it needs only discover the key from this information.'
- 'In this letter I make some remarks on a general principle relevant to enciphering in general and my machine.'
- 'We see immediately that one needs little information to begin to break down the process.'

问：懒得翻译

分析：数字母，第三个选项共有107个字节，填充后共有112字节，附加上IV的16字节共计128字节，满足题意

Q9

9. Let $R := \{0, 1\}^4$ and consider the following PRF $F : R^5 \times R \rightarrow R$ defined as follows:

$$F(k, x) := \begin{cases} t = k[0] \\ \text{for } i=1 \text{ to } 4 \text{ do} \\ \quad \text{if } (x[i-1] == 1) \quad t = t \oplus k[i] \\ \text{output } t \end{cases}$$

That is, the key is $k = (k[0], k[1], k[2], k[3], k[4])$ in R^5 and the function at, for example, 0101 is defined as $F(k, 0101) = k[0] \oplus k[2] \oplus k[4]$.

For a random key k unknown to you, you learn that

$$F(k, 0110) = 0011 \text{ and } F(k, 0101) = 1010 \text{ and } F(k, 1110) = 0110.$$

What is the value of $F(k, 1101)$? Note that since you are able to predict the function at a new point, this PRF is insecure.

1111

问：

分析： k_4 一定参与xor，三个例子分别为

1. $k_1 \oplus k_2 \oplus k_4 = 0011$
2. $k_0 \oplus k_2 \oplus k_4 = 1010$
3. $k_1 \oplus k_2 \oplus k_3 \oplus k_4 = 0110$
4. 1 和3 xor 得到 $k_3 = 0101$

设问可转化为 $k_0 \oplus k_2 \oplus k_3 \oplus k_4$, 即上述式2 xor $k_3 = 1111$

In this project you will implement two encryption/decryption systems, one using AES in CBC mode and another using AES in counter mode (CTR). In both cases the 16-byte encryption IV is chosen at random and is prepended to the ciphertext.

For CBC encryption we use the PKCS5 padding scheme discussed in the lecture (14:04). While we ask that you implement both encryption and decryption, we will only test the decryption function. In the following questions you are given an AES key and a ciphertext (both are hex encoded) and your goal is to recover the plaintext and enter it in the input boxes provided below.

For an implementation of AES you may use an existing crypto library such as PyCrypto (Python), Crypto++ (C++), or any other. While it is fine to use the built-in AES functions, we ask that as a learning experience you implement CBC and CTR modes yourself.

1.

CBC key: 140b41b22a29beb4061bda66b6747e14

CBC Ciphertext 1:

4ca00ff4c898d61e1edbf1800618fb2828a226d160dad07883d04e008a7897ee2e4b7465d5290d0c0e6c6
822236e1daafb94ffe0c5da05d9476be028ad7c1d81

明文: Basic CBC mode encryption needs padding.

2.

CBC key: 140b41b22a29beb4061bda66b6747e14

CBC Ciphertext 2:

5b68629feb8606f9a6667670b75b38a5b4832d0f26e1ab7da33249de7d4afc48e713ac646ace36e872ad5
fb8a512428a6e21364b0c374df45503473c5242a253

明文: Our implementation uses rand. IV

3.

CTR key: 36f18357be4dbd77f050515c73fcf9f2

CTR Ciphertext 1:

69dda8455c7dd4254bf353b773304eec0ec770233009ce7f7520d1cbbb20fc388d1b0adb5054dbd7370
849dbf0b88d393f252e764f1f5f7ad97ef79d59ce29f5f51eeca32eabedd9afa9329

明文: CTR mode lets you build a stream cipher from a block cipher.

4.

CTR key: 36f18357be4dbd77f050515c73fcf9f2

CTR Ciphertext 2:

770b80259ec33beb2561358a9f2dc617e46218c0a53cbeca695ae45faa8952aa0e311bde9d4e01726d318
4c34451

明文: Always avoid the two time pad!

W3 5-1 Message Authentication Codes

1、Message Integrity

目标: 不保密的情况下确保完整性

例子:

- 保护磁盘上的二进制文件: 必须确保系统内文件完整性, 而不需要进行加密
- 保护网页上广告: 广告面向所有用户, 因此不需要保密, 但是必须确保不被修改

2、Message integrity: MACs



Generate tag:
 $\text{tag} \leftarrow S(k, m)$

Verify tag:
 $V(k, m, \text{tag}) ? \text{'yes'}$

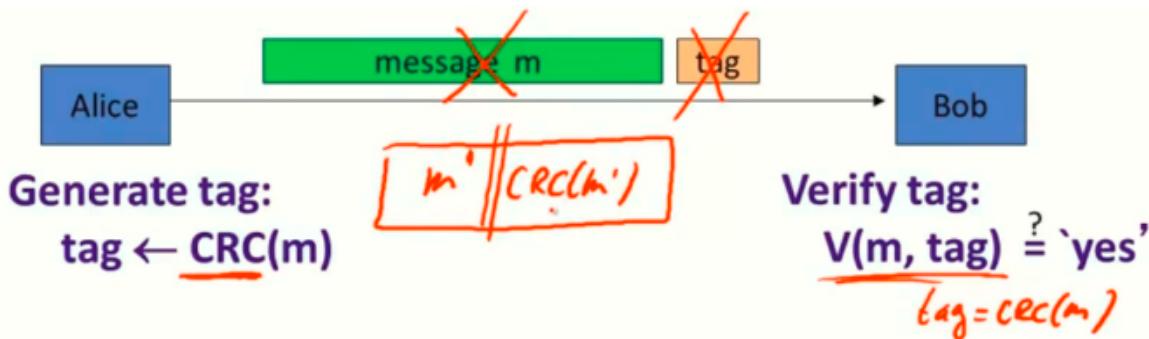
模型如图所示，实现完整性需要用到MAC，即Message Authentication Codes，Alice和Bob事先共享一个密钥k（攻击者不知道的k），之后Alice运行算法S，Bob运行算法V，Alice将计算后的tag附在原消息后一并发送给Bob，Bob收到后提取tag并验证，若通过验证则表明未被修改

定义：MAC I = (S, V) 定义在三元组(K, M, T)上的一对算法，K为密钥空间，M为消息空间，T为tag空间，算法如下：

- S(k, m)：签名算法，接收密钥k和消息m两个输入，产生一个很短的输出tag，通常只有100 bits左右，即便是消息m很大（GB甚至TB级别），也可以输出很短的tag
- V(k, m, tag)：验证算法，接收密钥k、消息m和tag作为输入，输出验证是否通过（yes/no）

需要注意的是，算法S和V必须满足一个条件：对于任给的 $k \in K, m \in M, V(k, m, S(k, m))$ 的输出一定是yes，即必须正确验证

3、Integrity requires a secret key



上述模型说明了完整性需要一个安全的密钥

假设Alice与Bob以CRC（Cyclic Redundancy Check）作为其签名与验证算法，而CRC仅接受一个输入，即消息m，不接受密钥k作为输入

若攻击者企图攻击，则攻击者可以轻易的删除Alice发出的消息m及其tag，重新生成一个消息m'并计算其tag'= $\text{CRC}(m')$ ，因此上述模型中，攻击者很容易修改消息并欺骗Bob，让其认为消息有效，而此时的消息m'与Bob期望的消息m完全无关

CRC的目的：用于检验消息中的随机错误，而非恶意错误，主要用于一些路由算法中确保消息的每一位被正确传送，使其不受信道中的偶然差错影响

综上所述，若在完整性中没有密钥的加入的话，Alice和攻击者的身份是等价的（两者地位没有区别），Bob不知道消息到底是来自于谁

解决方法：引入密钥，使得Alice可以完成一些攻击者无法完成的工作，使得计算结果包含攻击者无法修改的标记

4、Secure MACs

攻击者的能力：选择消息攻击，即攻击者可以给Alice任意选择的消息 $m_1 \sim m_q$ ，Alice可以为他计算这些消息的tag（类似于选择明文攻击）

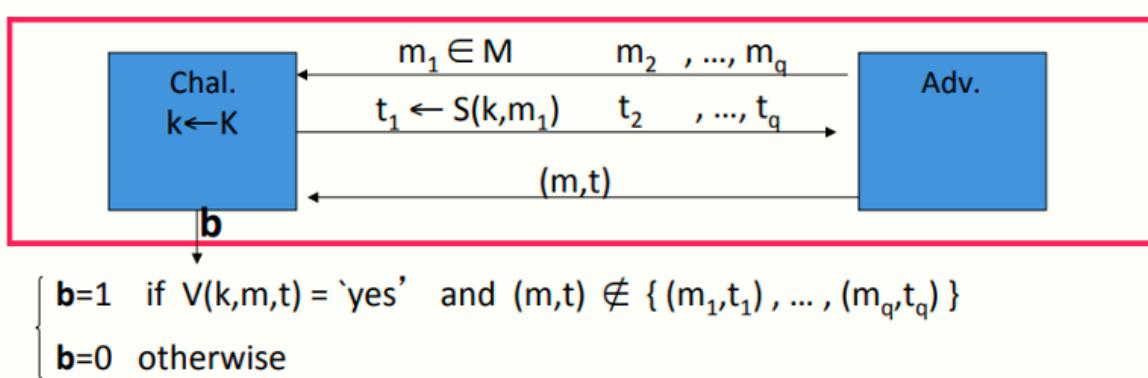
攻击者目标: existential forgery (没看懂) , 即攻击者生成一些合法的消息和tag对 (m,t) , 且 $(m,t) \notin \{(m_1,t_1), \dots, (m_q,t_q)\}$

若攻击者不能做到这种existential forgery, 则意味着系统是安全的, 若不能做到, 则表明攻击者无法生成一个合法的消息-tag对 (m,t) (尽管消息m可能是乱七八糟的)

为什么要对乱七八糟的消息验证: 密钥可能是乱七八糟的 (随机生成的) , 有些场景需要对密钥的完整性进行验证, 因此若攻击者可以生成正确的消息-tag对, 意味着系统不安全

综上, 需要满足两个:

- 攻击者不能对一个新的消息生成正确的tag
- 对于给定的消息 (m,t) , 攻击者不能生成另一个消息-tag对 (m,t') , 使得 $t' \neq t$



对于MAC I=(S,V)和攻击者A, 定义上述模型, 其中符号定义如下

- $b=1$: 表明 $V(k, m, t) = \text{yes}$, 且 $(m, t) \notin \{(m_1, t_1), \dots, (m_q, t_q)\}$
- $b=0$: 表明其他情况

定义: 若 $I=(S,V)$ 为一个安全的MAC, 则对所有高效的攻击者A, 其如下优势为可忽略的

$$Adv_{MAC}[A, I] = Pr[\text{Chal. outputs } 1]$$

5、Example: protecting system files



例子: 假设需要在机器上安装OS (如Windows) , OS先要求用户输入一个密钥 k , 之后由 k 运行算法 S , 对每个文件单独计算tag ($t_1 \sim t_n$) , 并将这些tag附在文件后, 计算完毕后删除 k

假设此后的某一时间, OS遭遇病毒攻击并修改了一些文件, 此时用户会将系统重启至一个干净的环境并提供一个密钥, 之后系统对每个系统文件的MAC进行检验, 由于MAC是安全的, 因此病毒不能产生 (F', t') 来通过检验, 因此OS能检测到被病毒修改的所有文件

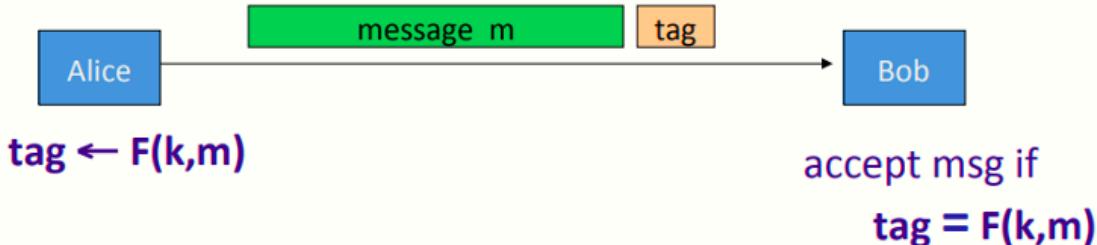
病毒还可能进行如下行为: 交换两个文件的位置, 即用户尝试运行 F_1 时, 实际运行文件为 F_2 , 此时系统会检测到 F_1 的文件没有正确的文件名, 从而检测到病毒

W3 5-2 MACs based on PRFs

1、Secure PRF \Rightarrow Secure MAC

For a PRF $F: K \times X \rightarrow Y$ define a MAC $I_F = (S, V)$ as:

- $S(k, m) := F(k, m)$
- $V(k, m, t)$: output 'yes' if $t = F(k, m)$ and 'no' otherwise.



对于一个PRF，定义MAC $I_F(S, V)$ 如下：

- $S(k, m) := F(k, m)$
- $V(k, m, t)$: 若 $t = F(k, m)$ 则输出 yes，否则输出 no

2、Security

定理：若 $F: K \times X \rightarrow Y$ 为一安全PRF，且 $1/|Y|$ 为一可忽略数（即 $|Y|$ 很大），则 I_F 为安全MAC

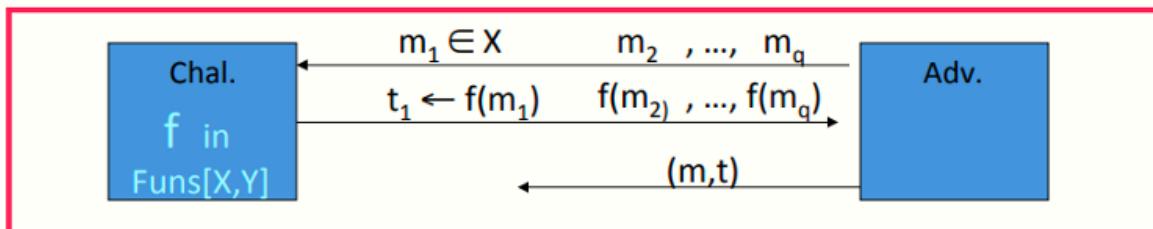
具体而言，对于任意高效的MAC攻击者A攻击 I_F ，存在一高效的PRF攻击者B攻击 F 使得其满足如下不等式

$$Adv_{MAC}[A, I_F] \leq Adv_{PRF}[B, F] + 1/|Y|$$

即：只要 $|Y|$ 足够大（如 $|Y| = 2^{80}$ ），则 I_F 就是安全的

3、Proof Sketch

假设 $f: X \rightarrow Y$ 为一真随机函数



若MAC的攻击者A想要赢得上述游戏模型，则必须生成一对 (m, t) ，使得 $t = f(m)$ 且 $m \notin \{m_1, \dots, m_q\}$

由于 f 为真随机函数，攻击者A对于前 q 次查询，不能得到关于 f 的信息（即前 q 次查询是完全独立的，新的查询 m 与之前的 m_1, \dots, m_q 无关），因此A希望得到 $f(m)$ 的方式只能为猜测，而其猜对的概率 $\Pr[A \text{ wins}] = 1/|Y|$

综上，若我们希望确保MAC的安全性，即便是将真随机函数 f 替换为伪随机函数 F ，攻击者A也无法区分，且其在赢得上述游戏模型中最多有 $1/|Y|$ 的优势

5、Truncating MACs based on PRFs

引理：假设 $F: K \times X \rightarrow \{0,1\}^n$ 为一安全PRF，则对于所有的 $1 \leq t \leq n$ ， $F_t(k, m) = F(k, m)[1 \dots t]$ 也是安全的PRF

解释：假设有一N bits的PRF，若将其输出截断到t bits，则其仍然是随机的，因为截断后攻击者能获得的信息更少了，所以其区分伪随机和真随机的工作会变得更困难

推论：若(S,V)为一个基于安全PRF的MAC，且输出n bits tag，只要 $1/2^w$ 仍可忽略（通常 $w \geq 64$ ），则将其输出截断至w bits仍为安全的

总结：如果我们用AES来构造MAC（AES-128每块输出128 bits），意味着我们将得到128 bits的MAC，但是综上所述，我们可以将其截断至90 bits或80 bits使其仍然为安全的，从而构造长度更为合理的MAC

W3 5-3 CBC-MAC and NMAC

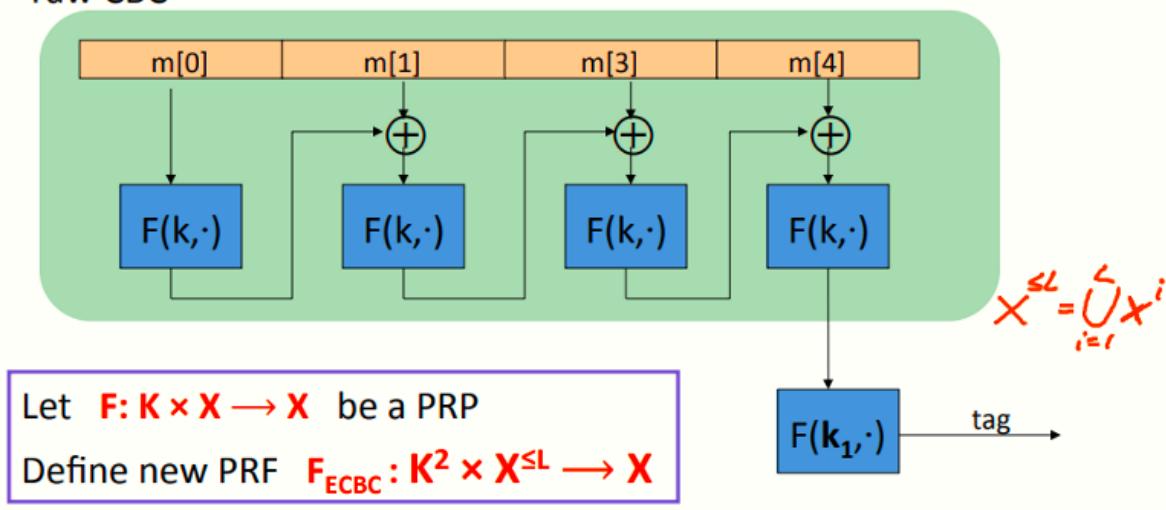
1、MACs and PRFs

上节课讲到，可以通过安全的PRF来构造安全的MAC，但需要注意PRF的输出必须很大（如80 bits或128 bits）

目标：给定一个输出短消息的PRF（如AES），为长消息（GB甚至TB级别）构造一个PRF

2、Construction 1: encrypted CBC-MAC

raw CBC



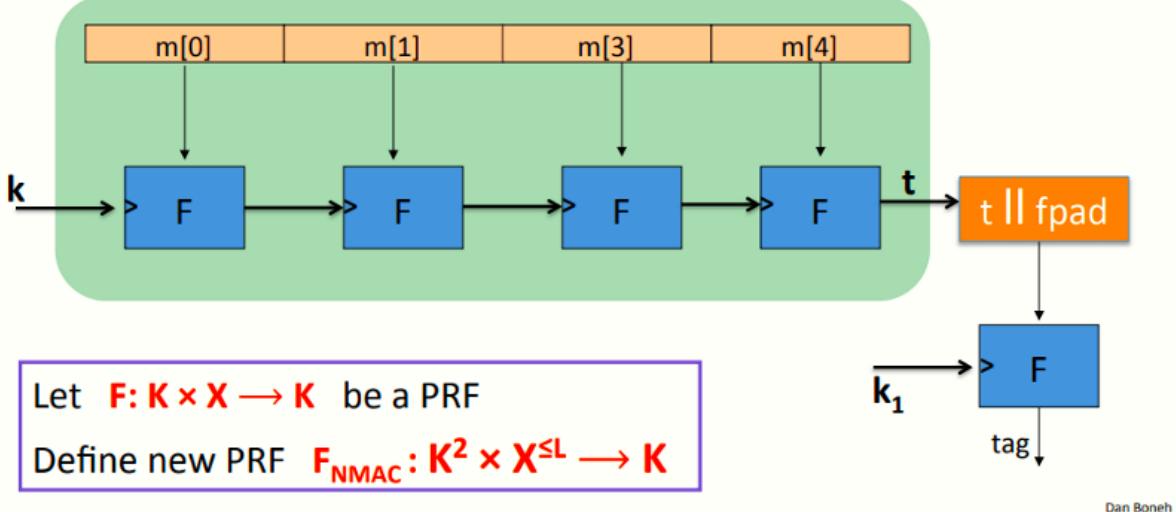
CBC-MAC可以接收很长的消息，最长接收块为L（百万或十亿数量级），上图中的 $X \leq L$ 表明可以接收1~L之间任意数量的块

上述模型绿色部分为raw CBC，该模型先将消息分块，然后运行CBC模式，但不输出中间值，将结果与下一块继续输入F，直到最后一块时输出

由于raw CBC本质上是不安全的，为了确保MAC的安全，需要加上右下角的最后一步，将CBC的结果使用另一个独立的 k_1 加密，得到最终输出

3、Construction 2: NMAC (nested MAC)

cascade



注意到与CBC-MAC不同的是，NMAC输出的消息来自于密钥空间K

上图模型中，初始密钥k先与第一个消息块一起输入函数F，之后F的输出作为新的密钥与下一块消息输入F，如此串联直到所有块计算完毕

由于该串联模型（绿色部分）也是不安全的，需要再加最后一步，而注意到密钥长度K与消息块长度不一致，因此需要将最后一步的输出t扩展，即拼接上fixed pad (fpad)，然后再输入由另一个密钥k1作用的函数F

4、Why the last encryption step in ECBC-MAC?

假设定义MAC $I_{RAW} = (S, V)$, $S(k, m) = \text{raw CBC}(k, m)$

则 I_{RAW} 很容易受到单块消息的选择明文攻击，攻击者只需完成下述步骤：

1. 选择消息 $m \in X$, 使得消息长度等于分块长度
2. 请求计算 m 的 tag, 即获取 $t = F(k, m)$
3. 伪造一个长度为2块的消息 $(m, t \oplus m)$

完成上述步骤后，使用消息 $(m, t \oplus m)$ 继续请求时，则有如下等式：

$$\text{rawCBC}(k, (m, t \oplus m)) = F(k, F(k, m) \oplus (t \oplus m)) = F(k, t \oplus (t \oplus m)) = t$$

5、ECBC-MAC and NMAC analysis

定理：对于任意 $L > 0$ ，对于有至多 q 次查询的高效PRF攻击者A攻击 F_{ECBC} 或 F_{NMAC} ，存在一高效的攻击者B，使得满足如下不等式：

$$Adv_{PRF}[A, F_{ECBC}] \leq Adv_{PRP}[B, F] + 2q^2/|X|$$

$$Adv_{PRF}[A, F_{NMAC}] \leq qLAdv_{PRF}[B, F] + q^2/2|K|$$

需要注意的是，CBC中使用的F为PRP（计算中不必取逆），而对于MAC而言，PRF不必可逆

总结：只要密钥不被用于MAC长度超过根号 $|X|$ (ECBC) 或根号 $|K|$ (NMAC) 的消息时，MAC就是安全的，这个值对于AES-128而言为 2^{64}

6、The security bounds are tight: an attack

在ECBC-MAC下签名根号 $|X|$ 条消息或在NMAC下签名 $|K|$ 条消息后，这些模式就变得不安全了

假设PRF F为一PRP (如AES) , 则两种PRF (ECBC或NMAC) 均有如下扩展性质:

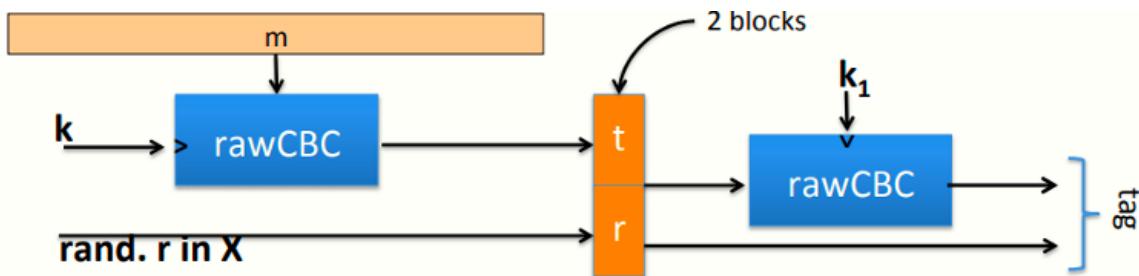
$$\forall x,y,w: F_{BIG}(k, x) = F_{BIG}(k, y) \Rightarrow F-BIG(k, x||w) = F_{BIG}(k, y||w)$$

上述性质表明, 若对于消息x和y存在一个碰撞, 则将其扩展之后 (以w拼接) 也会发生碰撞

因此, 记 $F_{BIG}: K \times X \rightarrow Y$ 为一个由上述扩展性质的PRF, 则有如下攻击:

1. 在消息空间X中构造根号 $|Y|$ 条随机消息并获取其tag, 即获取 (m_i, t_i) for $i = 1, \dots, |Y|^{1/2}$
2. 找到一个碰撞 $t_u = t_v$ 且 $u \neq v$ (由于生日悖论, 这个碰撞高概率存在)
3. 选择某个w并请求一个tag $t := F_{BIG}(k, m_u||w)$
4. 输出伪造消息 $(m_v||w, t)$ (此时 $t := F_{BIG}(k, m_v||w)$)

7、Better security: a rand. construction



Let $F: K \times X \rightarrow X$ be a PRF. Result: MAC with tags in X^2 .

Security: $\text{Adv}_{MAC}[A, I_{RCBC}] \leq \text{Adv}_{PRP}[B, F] \cdot (1 + 2 q^2 / |X|)$

\Rightarrow For 3DES: can sign $q=2^{32}$ msgs with one key

Dan Boneh

8、Comparison

ECBC-MAC通常用作基于AES的MAC (如802.11i中的CCM加密模式, 或者NIST标准CMAC)

NMAC由于其串联特性, 实际上每块消息都是新的密钥, 因此某种程度上来收其更换密钥的速度很快, 通常不会结合AES使用, 但其是HMAC的基础

W3 5-4 MAC padding

1、What if msg. len. is not multiple of block-size?

消息长度不为分组长度的倍数时该怎么办?

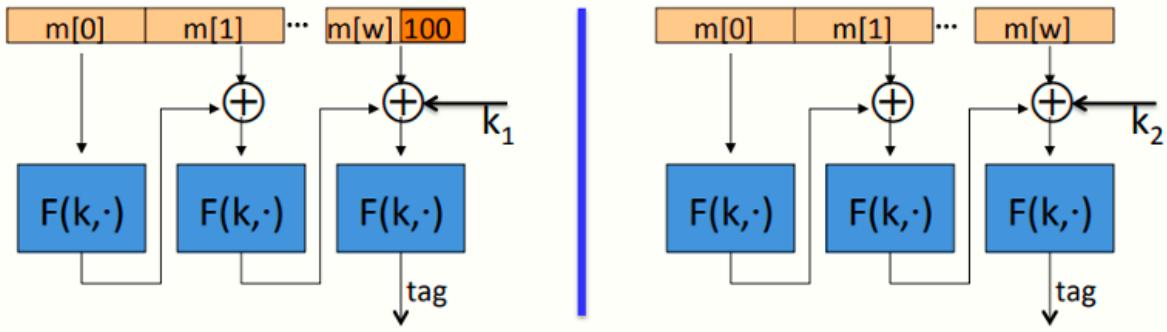
简单, 填充很多个0, 但是很笨, 即 $\text{pad}(m) = \text{pad}(m||0)$

因此CMC-MAC扩展必须是不可逆的, 即若 $m_0 \neq m_1 \Rightarrow \text{pad}(m_0) \neq \text{pad}(m_1)$

ISO的做法: 以1000...00扩展, 1代表从此处开始扩展, 接收时只需要右至左扫描0串直至第一个1

2、CMAC (NIST standard)

CMAC使用三个密钥 $\text{key} = (k, k_1, k_2)$, 第一个密钥k用于CBA-MAC的加密, 有如下模型



显然，区别于上一节的模型，该模型没有最后一步的额外加密

对于需要消息扩展的消息而言，在最后一次计算时需要扩展，扩展后加入 k_1 参与xor计算，再作为函数F的输入（图中左侧模型），而对于不需要扩展的消息（图中右侧模型），则使用 k_2 作为输入参与xor计算
优点：

- 解决了消息扩展攻击，由于攻击者不知道最后使用的xor密钥，不能实施扩展攻击，同时也省去了最后的加密步骤
- 解决了未对消息进行填补的不明确状态，利用两个不同的密钥区分两种不同的状态，使得消息填补更安全

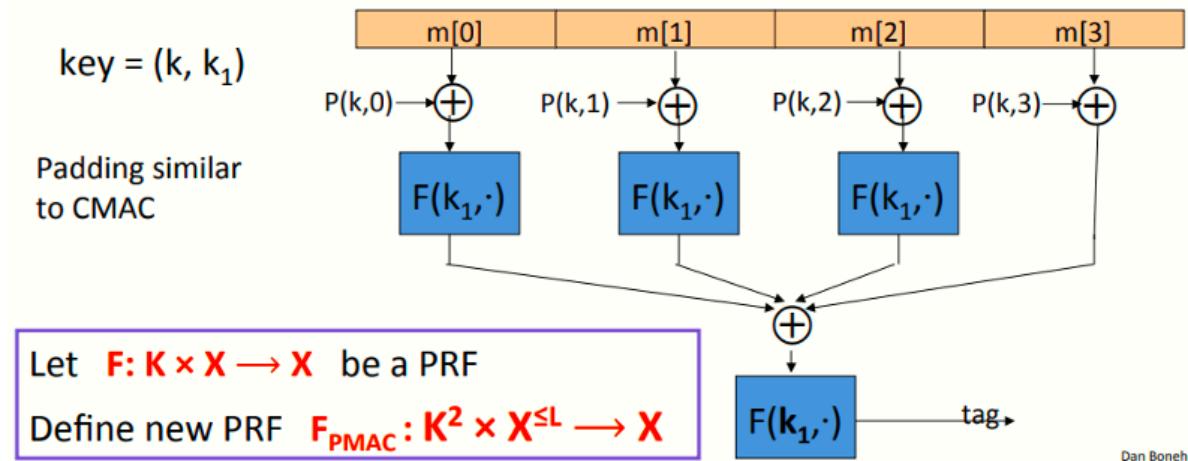
W3 5-5 PMAC and Carter-Wegman MAC

之前的课程讨论了CBC-MAC和NMAC将一个用于短消息的PRF转化成用于更长消息的PRF，这两种结构是按次序的，表明不能通过多处理器使其变得更快

本节课将关注一种并行的方式PMAC

1、Construction 3: PMAC - parallel MAC

$P(k, i)$: an easy to compute function



与之前的CBC-MAC一样，上图定义的PRF可以处理更长的消息，允许处理任意长度且可以划分成尽可能多的块的消息

该模型需要密钥key = (k, k_1) ，其中函数P为一易于计算的函数，P接收密钥k和块计数作为输入，需要扩展的消息的处理方式类似于CMAC

流程：

1. 先将消息分组
2. 将每个分组独立的与函数P的结果进行xor计算
3. 将2中的输出与密钥 k_1 一起作为函数F的输入

4. 前三步骤均可独立的并行运行，之后将#3中的各分块的函数F的输出一起进行一次xor计算
5. 将4中的xor结果与k₁一起作为F输入，最终输出tag

需要注意的是，如果没有函数P（即没有P与m_i的xor计算步骤），则上述模型不安全，因为若缺少消息与P的xor计算，则相当于将消息直接输入到F函数，攻击者可以简单的交换任意两块或多块消息的位置（如将m₁和m₂位置交换，或将所有消息块循环移动），但最终获得的tag值不会改变，因此攻击者有能力构造一些由特定消息块组成的消息而其tag并不会改变，从而造成攻击

结合上述分析，函数P强制规定消息块之间排列的明确的前后次序（尽管计算上是并行的），因此不仅需要一个密钥k作为输入，还需要块计数i作为输入，使得其对于每个块即便是在交换后也有不同的计算结果

函数P是一个非常容易计算的函数，只会占用模型极少的计算时间，但这也足以确保PMAC的安全

2、PMAC: Analysis

PMAC定理：对于任意L>0，若F为一在(K,X,X)上的安全PRF，则F-PMAC为在(K, X^{≤L}, X)上安全的PRF，对于任意至多有q次查询的高效PRF攻击者A攻击F-PMAC，存在一高效PRF攻击者B，满足如下不等式

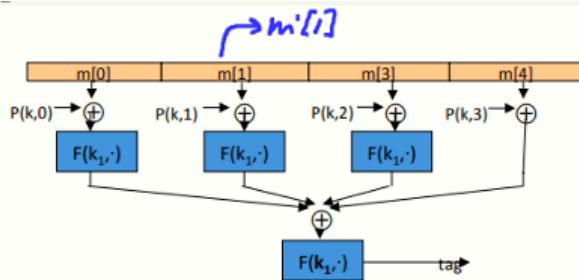
$$Adv_{PRF}[A, F_{PMAC}] \leq Adv_{PRF}[B, F] + 2q^2 L^2 / |X|$$

根据上述分析，若qL << |X|^{1/2} 则为安全的，MAC的消息块接近|X|^{1/2}时则需要考虑更换密钥来确保安全

3、PMAC is incremental

Suppose F is a PRP.

When m[1] → m'[1]
can we quickly update tag?



假设F使用PRP，若原消息的某一块或少数几块更改为新的消息块，则能否快速计算新的tag？

可以，由于PRP是可逆的，因此只需要计算如下表达式

$$F^{-1}(k_1, \text{tag}) \oplus F(k_1, m[1] \oplus P(k,1)) \oplus F(k_1, m'[1] \oplus P(k,1))$$

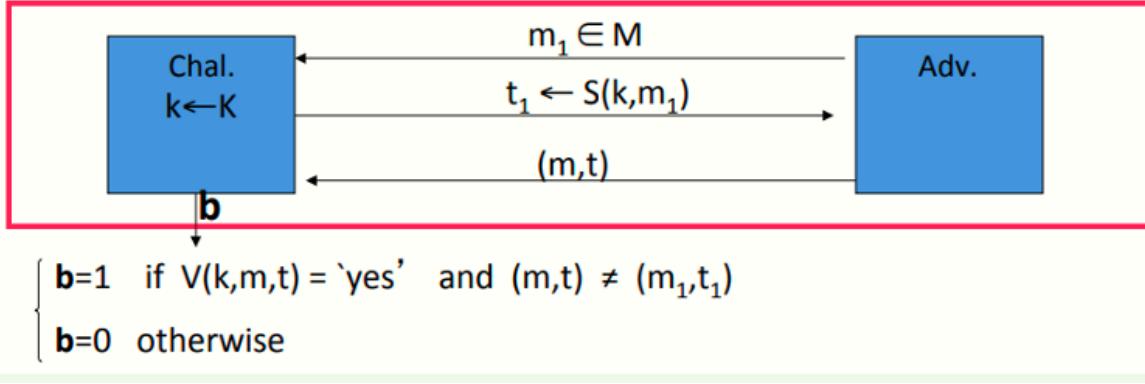
然后将上述结果再输入F(k1, ·)即可

说明：上述表达式，先计算F⁻¹，取得tag在PRP之前的结果，即所有消息块的F函数xor的结果，然后与原消息的F函数的输出xor，再xor上新的消息的F函数的输出，由于xor的特性，相当于去掉了原来的m₁，再加上了新的m_{1'}，最后再通过F函数得到新的tag即可

区别于CBC-MAC，若一块或少数几块消息更新了，则需要重新完整的计算一次tag，消耗时间

4、One time MAC (analog of one time pad)

对于MAC I=(S,V)和攻击者A，定义下图MAC游戏模型



定义：若 $I=(S,V)$ 为安全MAC，则其对所有高效A，其如下优势为可忽略

$$Adv_{1MAC}[A, I] = Pr[Chal. outputs 1]$$

一次性MAC和OTP一样，可以抵御有用无穷计算能力的攻击者，由于其一次性得特性，其比基于PRF的MAC更快

5、One time MAC: example

取q为一大素数（如 $q=2^{128}+51$ ）

记 $key = (a, b) \in \{1, \dots, q\}^2$ ，即 a, b 均为 $[1, q]$ 之间随机选择的整数

记 $msg = (m[1], \dots, m[L])$ ，每块消息 m_i 均为128 bits

记 S 如下

$$S(key, msg) = P_{msg}(a) + b \pmod{q}$$

其中 P_{msg} 为一多项式

$$P_{msg}(x) = x^{L+1} + m[L] * x^L + \dots + m[1] * x$$

不难看出，对于给定的MAC值 $S(key, msg_1)$ ，攻击者不能获得关于另一个消息的MAC值 $S(key, msg_2)$

6、One-time MAC \Rightarrow Many-time MAC

记 (S, V) 为一定义在 $(K, M, \{0,1\}^n)$ 上的安全One-time MAC

记 $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ 为一安全PRF

Carter-Wegman MAC:

$$CW((k_1, k_2), m) = (r, F(k_1, r) \oplus S(k_2, m))$$

CW MAC需要一个随机生成的nonce r （每次计算MAC时需要重新选择），CW MAC分为两个部分，使用PRF F 的部分计算较为缓慢，但只需要接收较短的输入（密钥 k_1 和nonce r ）， S 接收很长的输入并且计算很快，最终将 F 和 S 的结果进行 \oplus 计算得到输出

定理：若 (S, V) 为一安全one-time MAC且 F 为安全PRF，则CW为安全MAC，输出tag空间为 $\{0,1\}^{2n}$

CW MAC的验证： $V(k_2, m, F(k_1, r) \oplus t)$

7、Construction 4: HMAC (Hash-MAC)

W3 6-1 Collision resistance Introduction

1、Recap: message integrity

上一章讲的四种MAC构造方式

- ECBC-MAC, CMAC: AES常用
- NMAC: HMAC的基础
- PMAC: 并行计算MAC
- CW MAC: 基于高速的一次性MAC

2、Collision Resistance

记 $H: M \rightarrow T$ 为一Hash函数，其中 $|M|$ 远大于 $|T|$

碰撞: H 的碰撞，即找到消息对 $m_0, m_1 \in M$, 使得 $H(m_0) = H(m_1)$ 且 $m_0 \neq m_1$, 由鸽舍原理可知，由于 $|M|$ 远大于 $|T|$ ，一定会有两个消息映射到同一个tag

抗碰撞性: H 若为抗碰撞的，表明其对所有 explicit 的高效算法 A ，其如下优势可忽略

$$Adv_{CR}[A, H] = Pr[A \text{ outputs collision for } H]$$

其中 explicit，即明确的，意味着不是仅仅知道该算法的存在，还要求得这种算法，使得其能在计算机上运行并生成所需要的碰撞

常用Hash算法: SHA-256 (还未被攻破)

3、MACs from Collision Resistance

记 $I = (S, V)$ 为定义在 (K, M, T) 上接收短消息的MAC

记 $H: M^{\text{big}} \rightarrow M$

定义一个新的MAC如下:

Def: $I^{\text{big}} = (S^{\text{big}}, V^{\text{big}})$ over (K, M^{big}, T) as:

$$S^{\text{big}}(k, m) = S(k, H(m)) ; V^{\text{big}}(k, m, t) = V(k, H(m), t)$$

定理: 若 I 为一安全MAC且 H 为一抗碰撞Hash，则 I^{big} 为一安全MAC

抗碰撞性对于MAC安全很重要，假设攻击者可以找到一个碰撞，则上述 S_{big} 在选择明文攻击下不再安全

- 攻击者首先获取 tag $t \leftarrow S(k, m_0)$
- 由于攻击者可以找到碰撞，因此可以伪造消息和mac对 (m_1, t)

4、Protecting file integrity using C.R. hash

Software packages:



对于用户需要安装软件的场景，用户想确保其得到了官方的软件包而非攻击者恶意发布的，因此可以通过检查各个软件的Hash值，确保软件没有被篡改

由于Hash的抗碰撞性，攻击者不能在不被检测到的情况下修改软件包

对Hash的保存：仅需要其为公开且只读的，由于其公开，所以不需要密钥，但是需要确保只读以防止篡改

W3 6-2 Generic birthday attack

1、Generic attack on Collision resistance functions

记H: M → {0,1}ⁿ为一Hash函数，且|M| >> 2ⁿ

常规算法可以在O(2^{n/2})内找到一个hash碰撞，算法如下

1. 在消息空间M内选择2^{n/2}条随机消息m₁, ..., m_{2^{n/2}}
2. 对于i = 1, ..., 2^{n/2}, 计算t_i= H(m_i) ∈ {0,1}ⁿ
3. 找到一个碰撞t_i=t_j, 若未找到, 返回1

2、The birthday paradox

记r₁, ..., r_n ∈ {1,...,B}为n个独立同分布整数 (independent identically distributed, iid)

定理：若n= 1.2 × B^{1/2} 则Pr[∃ i ≠ j: r_i = r_j] ≥ 1/2, 证明如下

Proof: (for uniform indep. r₁, ..., r_n)

$$\Pr[\exists i \neq j : r_i = r_j] = 1 - \Pr[\forall i \neq j : r_i \neq r_j] = 1 - \left(\frac{B-1}{B}\right)\left(\frac{B-2}{B}\right) \dots \left(\frac{B-n+1}{B}\right) =$$

$$= 1 - \prod_{i=1}^{n-1} \left(1 - \frac{1}{B}\right) \geq 1 - \prod_{i=1}^{n-1} e^{-1/B} = 1 - e^{-\frac{1}{B} \sum_{i=1}^{n-1} i} \geq 1 - e^{-\frac{n^2}{2B}}$$

$1-x \leq e^{-x}$ $\frac{n^2}{2B} = 0.72$ $\geq 1 - e^{-0.72} = 0.53 > \frac{1}{2}$

Dan Boneh

3、Sample C.R. hash functions:

使用Crypto++ 5.6.0 [Wei Dai]

function	digest size (bits)	Speed (MB/sec)	generic attack time
NIST standards	SHA-1	160	2^{80}
	SHA-256	256	2^{128}
	SHA-512	512	2^{256}
Whirlpool	512	57	2^{256}

目前已知最好的找到SHA-1的碰撞的算法需要 2^{51}

W3 6-3 The Merkle-Damgard Paradigm

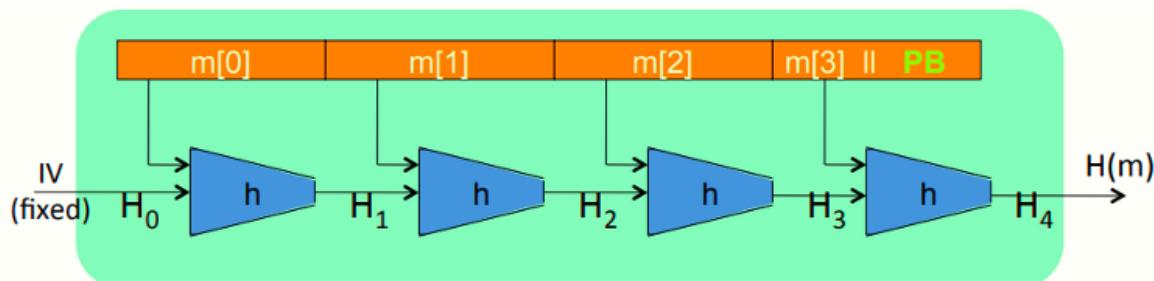
默尔克-达姆加德范式，常用于生成抗碰撞hash函数

1、Collision resistance: review

记 $H: M \rightarrow T$ 为一hash函数 ($|M| \gg |T|$)

目标：C.R. (collision resistant) hash函数

2、The Merkle-Damgard iterated construction



如图所示，记 $h: T \times X \rightarrow T$ 为一接收短消息为输入的C.R. hash函数（也叫压缩函数），初始向量IV为固定在代码或芯片中的值，消息作为输入并分块为 m_0, m_1, \dots

链式变量 $H_i: H: X^{\leq L} \rightarrow T$ ，对于 m_0 而言，将 m_0 和IV作为 h 的输入，输出 H_1 ，再与下一块消息 m_1 作为下一轮 h 的输入

填充块PB：padding block，包含1个1，若干个0和64 bits的消息长度，必须附加这个填充块，若消息尾部的长度不足以放下PB，则需要添加一个新的消息块

3、M-D collision resistance

定理：若 h 为一C.R. hash函数，则 H 也是

含义：若我们希望构造一个能接收长消息作为输入的C.R. hash函数，则我们只需要构造一个C.R.压缩函数即可

证明：反证法 (collision on $H \Rightarrow$ collision on h)

Suppose $H(M) = H(M')$. We build collision for h .

$$IV = H_0, H_1, \dots, H_t, H_{t+1} = H(M)$$

$$IV = H'_0, H'_1, \dots, H'_r, H'_{r+1} = H(M')$$

$$h(H_t, M_t \parallel PB) = H_{t+1} = H'_{r+1} = h(H'_r, M'_r \parallel PB')$$

If $\begin{cases} H_t \neq H'_r \\ M_t \neq M'_r \\ PB \neq PB' \end{cases}$ or

\Rightarrow we have a collision on h .

STOP

Dan Boneh

Otherwise,

Suppose $H_t = H'_r$ and $M_t = M'_r$ and $PB = PB'$

$\Rightarrow t = r$

Then: $h(H_{t-1}, M_{t-1}) = H_t = H'_r = h(H'_{t-1}, M'_{t-1})$

If $\begin{cases} H_{t-1} \neq H'_{t-1} \\ M_{t-1} \neq M'_{t-1} \end{cases}$ then we have a collision on h . STOP.

otherwise, $H_{t-1} = H'_{t-1}$, and $M_t = M'_t$ and $M_{t-1} = M'_{t-1}$.

Iterate all the way to beginning and either:

(1) find collision on h , or

(2) $\forall i: M_i = M'_i \Rightarrow M = M'$

cannot happen because M, M' are collision on H .

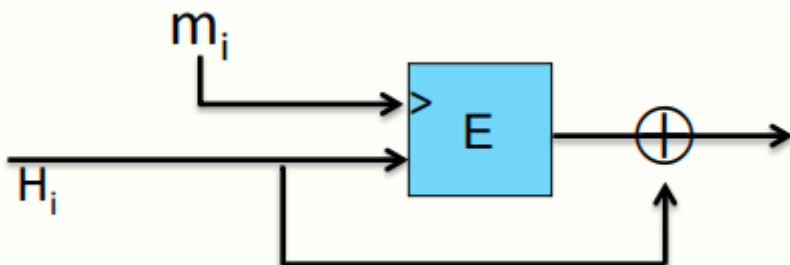
Dan Boneh

W3 6-5 Constructing Compression Functions

1. Compr. func. from a block cipher

记 $E: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ 为一块密码

Davies-Meyer压缩函数: $h(H, m) = E(m, H) \oplus H$, 结构如下



定理: 若 E 为一理想加密算法 (集合 K 上的随机置换), 则找到一个碰撞 $h(H, m) = h(H', m')$ (使用生日攻击), 复杂度为 $O(2^{n/2})$, 即需要分析 $2^{n/2}$ 对 (E, D)

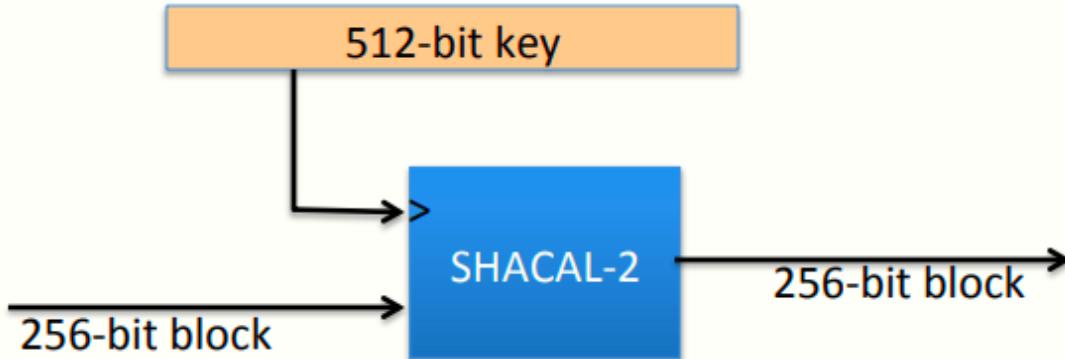
2. Other block cipher constructions

记 $E: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$

通过分组密码构造抗冲突的压缩函数方式还有很多种，如Miyaguchi Preneel，其有12个变体提供抗冲突机制

3、Case : SHA-256

使用Merkle-Damgård函数，Davies-Meyer压缩函数，块密码使用SHACAL-2，结构如下



密钥长度为512 bits，意味着该算法一次可处理512 bits的消息，块大小为256 bits

4、Provable compression functions

另一类压缩函数，不使用块密码进行构造，而是基于数论中的数学难题

原理：选择一个2000 bits的素数p，并随机选择u、v，使得 $1 \leq u, v \leq p$ ，对于 $m, h \in \{0, \dots, p-1\}$ ，定义函数 $h(H, m) = u^H \cdot v^m \pmod{p}$

事实上，若想找到上述算法的一个碰撞，其难度可归约为解决离散对数问题

实际上并没有使用上述算法，而是使用基于块密码的压缩函数，原因是上述算法的效率实在是太低了，想要计算稍微长一点的消息的MAC值可能需要一天甚至更长

W3 6-6 Constructing Compression Functions

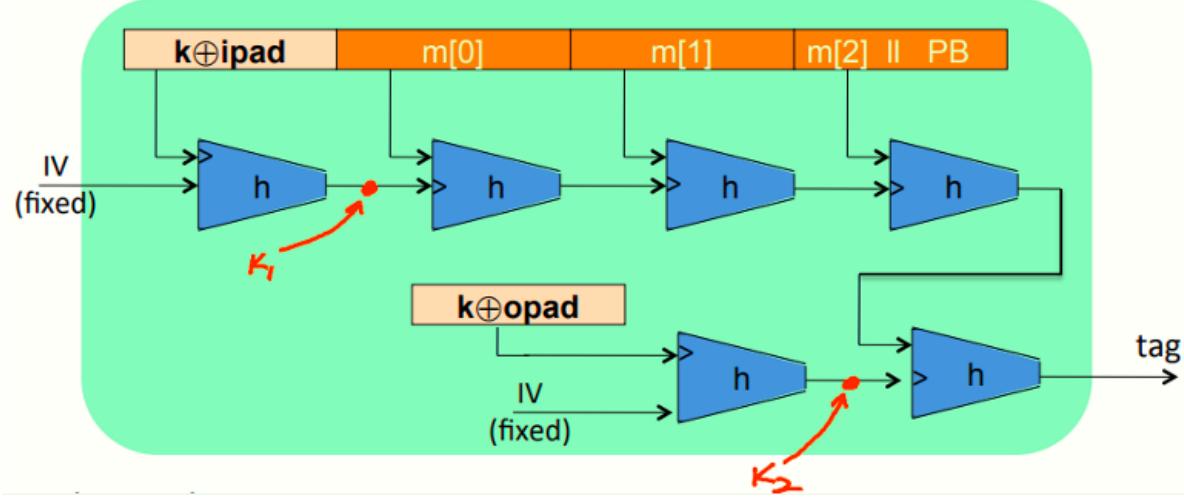
1、Standardized method: HMAC (Hash-MAC)

因特网中最广泛使用的MAC算法，其中的H代表hash函数（如SHA-256）

基于hash函数构建MAC的方法：

$$HMAC : S(k, m) = H(k \oplus opad || H(k \oplus ipad || m))$$

具体结构如下图



ipad: internal padd, 与密钥k进行xor运算后得到一个512 bits的块，并将其连接到消息M的前面

opad: outer pad, 同样与密钥k进行xor得到512 bits的块

ipad和opad都为标准上固定的常量，均为512 bits且相互独立

从符号意义上来说， $k \oplus \text{ipad}$ 与IV作为h的输入得到的输出可以视为密钥 k_1 ，而 $k \oplus \text{opad}$ 同理与IV作为h的输入，得到的输出可以视为密钥 k_2 ，从而通过一个密钥k得到了两个子密钥

W3 6-7 Timing attacks on MAC verification

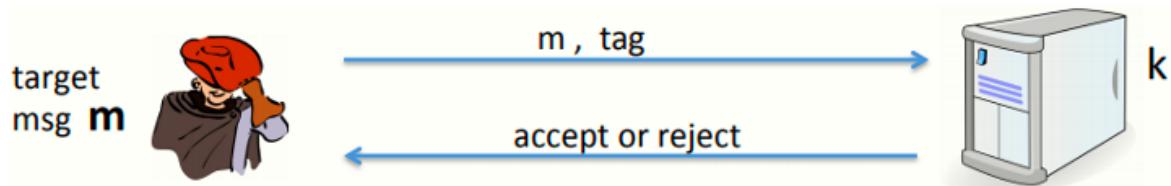
1、Warning: verification timing attacks

如Python的Keyczar库中的算法，其摘要如下

```
def Verify(key, msg, sig_bytes):
    return HMAC(key, msg) == sig_bytes
```

该算法采用重新计算HMAC的tag的方式，并与传入的tag作比较

问题在于，其中的比较"=="为按字节（byte-to-byte）比较，一旦找到其中第一个不匹配的字节时立即返回false



针对上述问题，假设如图模型，攻击者有消息m和其对应的tag，服务器上存储了一个HMAC的密钥，攻击者将m和tag发送给服务器，服务器验证是否合法后返回accept或reject

步骤：

1. 选择一个消息m固定，然后随便选一个tag并将其与m一起发送给server，计算server的响应时间
2. 遍历tag的第一字节的所有可能值并与m一并提交，计算每次服务器响应的时间并与1中时间进行比较，若响应时间比1中的时间略长，说明第一字节通过验证，错误出现在下一字节
3. 重复1和2，直到全部的16字节通过验证

2、Defense

(1) 对于tag串的比较，使其总是保持相同的时间开销，思路如下：

```
return false if sig_bytes has wrong length  
result = 0  
for x, y in zip( HMAC(key,msg) , sig_bytes):  
    result |= ord(x) ^ ord(y)  
return result == 0
```

但上述思路由于现代编译器的优化问题，很难确保每次请求验证tag时的时间开销总是一致或保持差别不大

(2) 同样使其总是保持相同的时间开销，但通过其他手段让攻击者不知道进行比较的字节的值，思路如下

```
def Verify(key, msg, sig_bytes):  
    mac = HMAC(key, msg)  
    return HMAC(key, mac) == HMAC(key, sig_bytes)
```

使用key对msg生成mac，然后再使用相同的key生成mac的mac，之后与提交的sig_bytes的mac进行比较

利用HMAC的特性，使得攻击者并不知道被比较的字符串究竟是什么，从而不能再进行上述的时序攻击，且也避免了（1）中可能发生的编译器优化

(3) 总结：实现密码库时，安全专家也可能产生错误，从而编写脆弱性代码，尽管理论上正确，但对于时序攻击而言仍不安全，从而破坏了系统的安全性

结论：永远不要自己去实现一个加密算法，自己实现的算法很可能无法应对侧信道攻击，应使用如 OpenSSL之类的标准库

W3 知识梳理

W3 Problem Set && Programming Assignment

Q1

1. Suppose a MAC system (S, V) is used to protect files in a file system by appending a MAC tag to each file. The MAC signing algorithm S is applied to the file contents and nothing else. What tampering attacks are not prevented by this system?

- Changing the name of a file.
- Changing the first byte of the file contents.
- Appending data to a file.
- Replacing the contents of a file with the concatenation of two files on the file system.

问：假设一MAC (S,V) 用于保护文件系统，方式为将tag附在每个文件后，签名算法S作用域文件内容，以下哪种方式的攻击不会遭到该系统的保护？

分析：由于S只作用于文件内容，显然更改文件名会导致攻击

Q2

2. Let (S, V) be a secure MAC defined over (K, M, T) where $M = \{0, 1\}^n$ and $T = \{0, 1\}^{128}$. That is, the key space is K , message space is $\{0, 1\}^n$, and tag space is $\{0, 1\}^{128}$.

Which of the following is a secure MAC: (as usual, we use \parallel to denote string concatenation)

$S'(k, m) = S(k, m|0, \dots, n-2)\parallel 0$ and

$$V'(k, m, t) = V(k, m|0, \dots, n-2)\parallel 0, t$$

$S'(k, m) = [t \leftarrow S(k, m), \text{output}(t, t)]$ and

$$V'(k, m, (t_1, t_2)) = \begin{cases} V(k, m, t_1) & \text{if } t_1 = t_2 \\ "0" & \text{otherwise} \end{cases}$$

(i.e., $V'(k, m, (t_1, t_2))$ only outputs '1'

if t_1 and t_2 are equal and valid)



正确
a forger for (S', V') gives a forger for (S, V) .

$S'(k, m) = S(k, m|m)$ and

$$V'(k, m, t) = V(k, m|m, t).$$



正确
a forger for (S', V') gives a forger for (S, V) .

$S'(k, m) = S(k, m \oplus m)$ and

$$V'(k, m, t) = V(k, m \oplus m, t)$$

$S'(k, m) = S(k, m \oplus 1^n)$ and

$$V'(k, m, t) = V(k, m \oplus 1^n, t).$$



正确
a forger for (S', V') gives a forger for (S, V) .

$S'(k, m) = (S(k, m), S(k, 0^n))$ and

$$V'(k, m, (t_1, t_2)) = [V(k, m, t_1) \text{ and } V(k, 0^n, t_2)]$$

(i.e., $V'(k, m, (t_1, t_2))$ outputs ``1'' if both t_1 and t_2 are valid tags)

问：经典看不懂题目

分析：看懂了再分析，所以答案是抄的

Q3

Q4

Suppose Alice is broadcasting packets to 6 recipients B_1, \dots, B_6 . Privacy is not important but integrity is. In other words, each of B_1, \dots, B_6 should be assured that the packets he is receiving were sent by Alice.

Alice decides to use a MAC. Suppose Alice and B_1, \dots, B_6 all share a secret key k . Alice

computes a tag for every packet she sends using key k . Each user B_i verifies the tag when receiving the packet and drops the packet if the tag is invalid. Alice notices that this scheme is insecure because user B_1 can use the key k to send packets with a valid tag to users B_2, \dots, B_6 and they will all be fooled into thinking that these packets are from Alice.

Instead, Alice sets up a set of 4 secret keys $S = \{k_1, \dots, k_4\}$. She gives each user B_i some subset $S_i \subseteq S$ of the keys. When Alice transmits a packet she appends 4 tags to it by computing the tag with each of her 4 keys. When user B_i receives a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1, k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 .

How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user?

- $S_1 = \{k_2, k_3\}, S_2 = \{k_2, k_4\}, S_3 = \{k_3, k_4\}, S_4 = \{k_1, k_2\}, S_5 = \{k_1, k_3\}, S_6 = \{k_1, k_4\}$



正确

Every user can only generate tags with the two keys he has.

Since no set S_i is contained in another set S_j , no user i

can fool a user j into accepting a message sent by i .

- $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3, k_4\}, S_5 = \{k_2, k_3\}, S_6 = \{k_3, k_4\}$
- $S_1 = \{k_1\}, S_2 = \{k_2, k_3\}, S_3 = \{k_3, k_4\}, S_4 = \{k_1, k_3\}, S_5 = \{k_1, k_2\}, S_6 = \{k_1, k_4\}$
- $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3, k_4\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3\}, S_5 = \{k_2, k_3, k_4\}, S_6 = \{k_3, k_4\}$

问：Alice需要向6为客户 $B_1 \sim B_6$ 广播报文，需要确保完整性但无需确保安全性（即 $B_1 \sim B_6$ 应当确保收到的报文确实是Alice发送的）

假设Alice使用MAC，并与 $B_1 \sim B_6$ 共享密钥k，对于 B_i 收到的报文，若验证tag错误则丢弃报文

Alice注意到上述模型中存在缺陷， B_1 可以利用共享密钥k，将报文发送给 $B_2 \sim B_6$ 而tag验证不会出错，因此 $B_2 \sim B_6$ 会认为报文流来自于Alice

假设新方案Alice使用一密钥集合 $S=\{k_1, \dots, k_4\}$, 对于 B_i 而言, 分发给其的密钥为 S 的子集 S_i , 即 $S_i \subseteq S$

问下述哪种密钥分配方案能确保没有任何一个客户能欺骗其他客户

分析: 第一个选项中, 任意两个客户 B_i, B_j 之间持有的密钥的交集小于等于一个密钥, 由于通过验证需要两个密钥, 因此任意一个用户不能产生其他用户的更多的密钥

对于选项二, B_4 拥有 k_2, k_3, k_4 , 可以欺骗用户 B_5 和 B_6

对于选项三, 同理 B_4, B_5, B_6 可以欺骗 B_1

对于选项四, B_2 可以欺骗 B_3, B_6 , 且 B_5 可以欺骗 B_4, B_6

Q5

5. Consider the encrypted CBC MAC built from AES. Suppose we

compute the tag for a long message m comprising of n AES blocks.

Let m' be the n -block message obtained from m by flipping the

last bit of m (i.e. if the last bit of m is b then the last bit

of m' is $b \oplus 1$). How many calls to AES would it take

to compute the tag for m' from the tag for m and the MAC key? (in this question please ignore message padding and simply assume that the message length is always a multiple of the AES block size)

- 4
- 2
- 3
- $n + 1$

✓ 正确

You would decrypt the final CBC MAC encryption step done using k_2 ,

then decrypt the last CBC MAC encryption step done using k_1 ,

flip the last bit of the result, and re-apply the two encryptions.

问: 若CBC-MAC使用AES, 假设计算一长消息 m 的tag, 该消息包含 n 个AES块, 记 m' 为另一长度为 n 块的消息, 其为消息 m 的最后一位取反得到, 则由 m 的tag计算得到 m' 的tag需要调用多少次AES算法?

分析: 基于AES的CBC-MAC使用的是PRF, 因此解得最后一块消息需要调用两次, 之后最后一位取反再调用两次AES, 共四次

Q6

6. Let $H : M \rightarrow T$ be a collision resistant hash function.

Which of the following is collision resistant:

(as usual, we use \parallel to denote string concatenation)

$H'(m) = H(m) \oplus H(m \oplus 1^{|m|})$

(where $m \oplus 1^{|m|}$ is the complement of m)

$H'(m) = H(m[0, \dots, |m| - 2])$

(i.e. hash m without its last bit)

$H'(m) = H(m) \parallel H(m)$

✓ 正确

a collision finder for H' gives a collision finder for H .

$H'(m) = H(m)[0, \dots, 31]$

(i.e. output the first 32 bits of the hash)

$H'(m) = H(0)$

$H'(m) = H(m \parallel 0)$

✓ 正确

a collision finder for H' gives a collision finder for H .

$H'(m) = H(m \parallel m)$

✓ 正确

a collision finder for H' gives a collision finder for H .

问：若 $H : M \rightarrow T$ 为一抗碰撞hash函数，则下列哪些函数仍为抗碰撞？

分析：

1. 若 m 取 000，则 $H'(000) = H(000) \oplus H(111)$ ，若 m 取 111，则 $H'(111) = H(111) \oplus H(000)$ ，即 $H'(000) = H'(111)$ ，碰撞
2. 截断消息最后一位不抗碰撞，有 $H'(00) = H'(01)$
3. 显然 H 抗碰撞则 H' 也是
4. 同 2
5. 显然不抗碰撞，因为有 $H'(0) = H'(1)$
6. 显然 H 抗碰撞则 H' 也是
7. 显然 H 抗碰撞则 H' 也是

7. Suppose H_1 and H_2 are collision resistant

hash functions mapping inputs in a set M to $\{0, 1\}^{256}$.

Our goal is to show that the function $H_2(H_1(m))$ is also

collision resistant. We prove the contra-positive:

suppose $H_2(H_1(\cdot))$ is not collision resistant, that is, we are

given $x \neq y$ such that $H_2(H_1(x)) = H_2(H_1(y))$.

We build a collision for either H_1 or for H_2 .

This will prove that if H_1 and H_2 are collision resistant

then so is $H_2(H_1(\cdot))$. Which of the following must be true:

Either x, y are a collision for H_2 or

$H_1(x), H_1(y)$ are a collision for H_1 .

Either x, y are a collision for H_1 or

x, y are a collision for H_2 .

Either x, y are a collision for H_1 or

$H_1(x), H_1(y)$ are a collision for H_2 .

Either $H_2(x), H_2(y)$ are a collision for H_1 or

x, y are a collision for H_2 .

Q8

Q9

9. Repeat the previous question, but now to find a collision for the compression function $f_2(x, y) = \text{AES}(x, x) \oplus y$.

Which of the following methods finds the required (x_1, y_1) and (x_2, y_2) ?

- Choose x_1, x_2, y_1 arbitrarily (with $x_1 \neq x_2$) and set

$$y_2 = \text{AES}(x_1, x_1) \oplus \text{AES}(x_2, x_2)$$

- Choose x_1, x_2, y_1 arbitrarily (with $x_1 \neq x_2$) and set

$$y_2 = y_1 \oplus x_1 \oplus \text{AES}(x_2, x_2)$$

- Choose x_1, x_2, y_1 arbitrarily (with $x_1 \neq x_2$) and set

$$y_2 = y_1 \oplus \text{AES}(x_1, x_1) \oplus \text{AES}(x_2, x_2)$$

- Choose x_1, x_2, y_1 arbitrarily (with $x_1 \neq x_2$) and set

$$y_2 = y_1 \oplus \text{AES}(x_1, x_2)$$



正确

Awesome!

问：没看懂

答：蒙对的

Q10

10. Let $H : M \rightarrow T$ be a random hash function where $|M| \gg |T|$ (i.e. the size of M is much larger than the size of T).

In lecture we showed

that finding a collision on H can be done with $O(|T|^{1/2})$

random samples of H . How many random samples would it take

until we obtain a three way collision, namely distinct strings x, y, z

in M such that $H(x) = H(y) = H(z)$?

- $O(|T|^{2/3})$

- $O(|T|^{1/2})$

- $O(|T|)$

- $O(|T|^{1/3})$

问：记 $H : M \rightarrow T$ 为一随机hash函数， $|M| >> |T|$ ，找到 H 的碰撞的期望为 $O(|T|^{1/2})$ ，若希望找到三个碰撞，即找到不同的 x, y, z ，使得 $H(x)=H(y)=H(z)$ ，期望为多少

分析：首先对于给定的集合，包含 n 个元素， n 个任意选择3个为 $C n-3$ ，即期望为 $O(n^3)$ ，对于每组特定的元素，需要求 $H(x)=H(y)=H(z)$

而随机hash函数，产生碰撞的概率为 $1/|T|$ ，则产生上述三路碰撞的概率为 $1/|T|^2$ (需要满足 $H(x)=H(y)$ 且 $H(x)=H(z)$)

因此期望为 $O(n^3/|T|^2)$

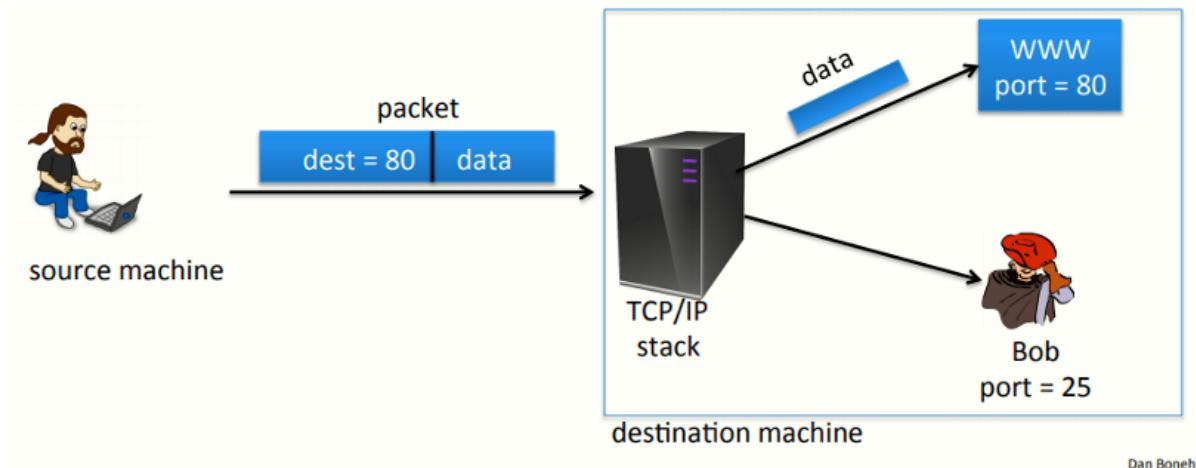
W4 7-1 Active attacks on CPA-secure encryption

1、Recap: the story so far

保密性：通过加密方案实现语义安全，从而防止选择明文攻击，但针对CPA的安全性只提供针对窃听的安全性，即攻击者仅监听网络流量而不修改任何数据包或注入攻击包

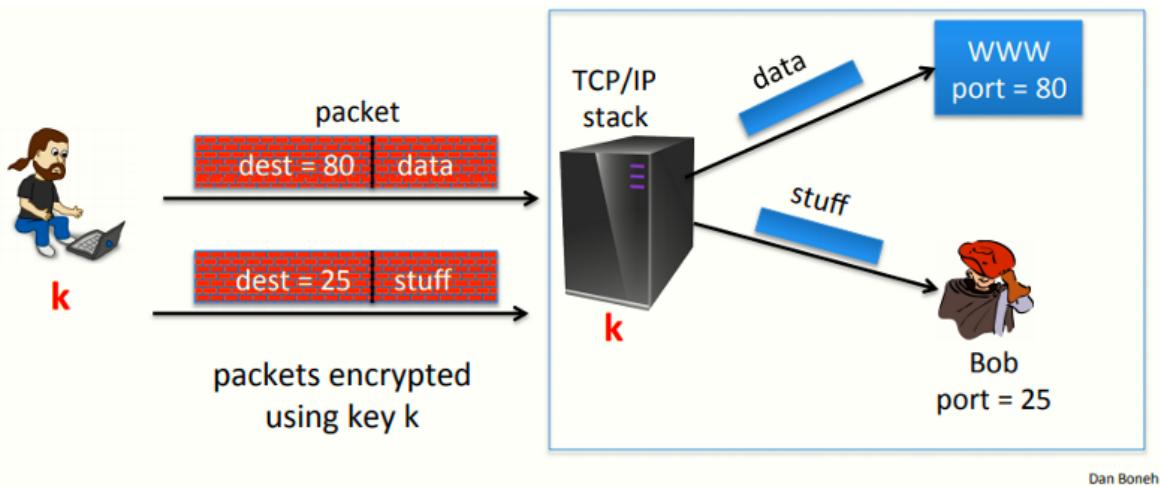
完整性：在消息不保密的情况下提供消息完整性，确保数据在传输过程中不会被修改，常见方案为MAC，提供了针对选择消息攻击的不可伪造性

2、Sample tampering attacks



Dan Boneh

简单的TCP/IP模型如上图，TCP/IP堆栈收到数据包后查看数据包的目的端口，之后交付给对应的监听进程（服务），如图中80端口则交付WWW服务器，25端口交付用户

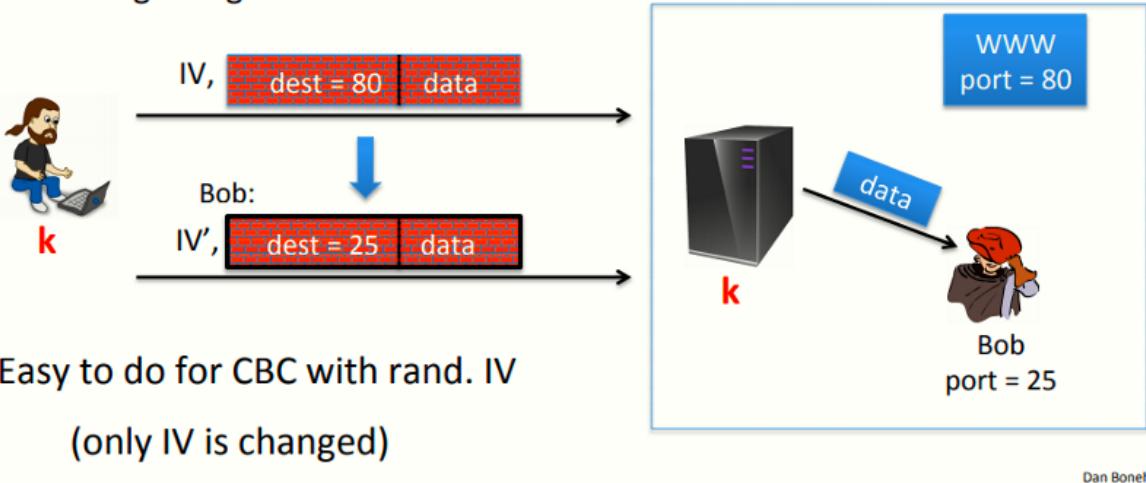


Dan Boneh

IPsec模型如上图，IPsec在发送方和接收方之间加密IP数据包，因此需要一个共享密钥k

发送方发送数据包时，使用k对其进行加密，接收方收到后，TCP/IP堆栈对其进行解密，之后查看目的端口并将其交付给对应进程

Note: attacker obtains decryption of any ciphertext beginning with “dest=25”



Easy to do for CBC with rand. IV
(only IV is changed)

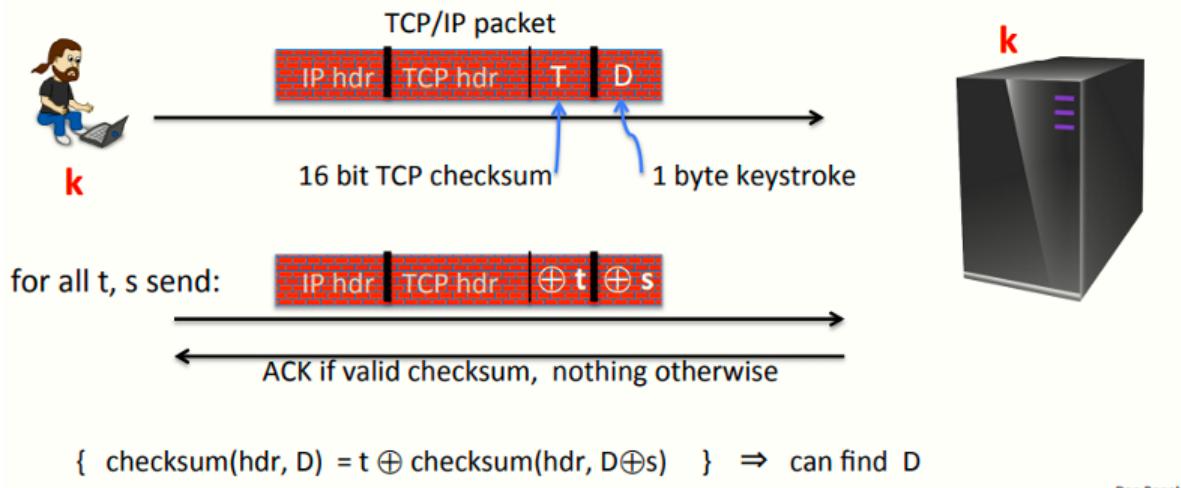
上述模型中，假设现在Bob为攻击者，加密方案使用随机IV的CBC模式，他的目标是在路由中拦截数据包，并修改其端口为25，且该操作在密文上完成，修改完成后再使其到达TCP/IP堆栈，之后堆栈解密后会将数据转发给Bob，从而读取到本来应该交付给Web服务器的数据

若解密第一块消息： $m[0] = D(k, c[0]) \oplus IV = "dest=80..."$ ，则构造 $IV' = IV \oplus (...80...) \oplus (...25...)$ ，可以使得目标接收数据包后，将目的端口解密为25

总结：上述例子说明了通过简单的修改IV字段，就可以导致一个简单的数据包转移，因此若没有完整性，攻击者可以在路由中修改数据包，CPA安全加密实际上并没有提供机密性

3、An attack using only network access

Remote terminal app.: each keystroke encrypted with CTR mode



假设有一远程终端应用，每次用户按键时，都互相发送一个加密的按键，假设加密采用CTR模式

攻击者截获但不会修改数据包并将其正确的发送到服务器，对于数据包，使用值T来与校验和字段进行xor计算，使用值S与数据字段进行xor计算，由于CTR模式的属性，若将密文与T进行xor，则解密后的结果为明文与T的xor的值，S与数据字段同理

通过上述方法，攻击者使用不同的T和S组合构造大量的伪造数据包，并将其发送到服务器，之后监听服务器的响应，若未响应，则说明修改后的数据和校验和不正确，若服务器回传一个ACK，则表明修改正确

攻击者通过获得大量的T和S对，攻击者可以计算出D的值，因此此类攻击为选择密文攻击，攻击者提交了选择的密文，即该密文来自于其想要解密的密文，通过监听服务器的响应，攻击者可以了解到关于明文的信息，不断重复这个过程从而学习到实际的纯文本

4、总结

CPA安全不能确保来自活跃的攻击者的攻击

若需要确保完整性而不需要保密性：使用MAC

若两者都需要：则需要使用身份验证加密模式

W4 7-2 Authenticated Encryption-Definitions

1、Goals

身份验证加密系统(E,D)定义如下：

记加密E: $K \times M \times N \rightarrow C$

记解密D: $K \times C \times N \rightarrow M \cup \{\perp\}$

其中N为一可选的随机数nonce，和以前不同的是，解密算法不仅输入明文消息M，还输出一个特殊符号bottom，当解密算法输出bottom时，意味着该密文无效且应当被忽略

bottom应独立于明文消息空间以确保其为独一无二的元素，用于表示应当拒绝收到的密文

安全性：系统必须提供如下安全性

- CPA攻击下的语义安全
- 密文完整性：攻击者不能构造一个可以被正确解密的密文，即不能解密出非bottom元素的结果

2、Ciphertext integrity

记(E,D)为一消息空间为M的密文，有上述游戏模型

定义：若(E,D) 为密文完整性，则其对于所有高效的攻击者A，其如下优势可以忽略

$$Adv_{CI}[A, E] = Pr[Chal. outputs 1]$$

3、Authenticated encryption

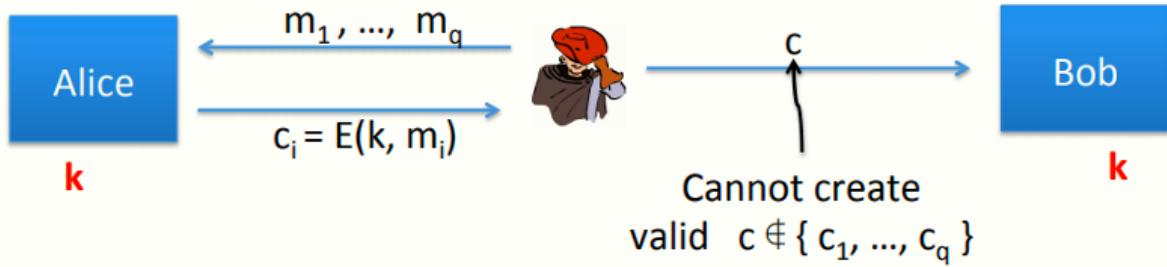
定义：若密文(E,D)提供认证加密，则其满足

- 在CPA下满足语义安全
- 满足密文完整性

前一节课的例子说明，使用随机IV的CBC模式不提供认证加密，即其不满足密文完整性，攻击者可以构造一些密文，而解密算法D(k,·)不会输出bottom元素，从而攻击者可以很轻易地赢得上述游戏模型

4、Implication 1: authenticity

真实性：攻击者不能欺骗接收方（Bob）使其相信发送方（Alice）发送了某些消息



模型如图，若 $D(k, c) \neq \perp$ ，即未输出bottom元素，则Bob可以相信消息来自于持有密钥k的发送方（但不一定来自期望的发送方，有可能是重放）

5、Implication 2

认证安全是一个非常强大的手段，其安全性可以用于抵御选择密文攻击 (chosen ciphertext attacks, CCA)

W4 7-3 Chosen ciphertext attacks

1、Example chosen ciphertext attacks

上节课讲到的一些CCA手段，模型如下

- Often, adv. can fool server into decrypting **certain** ciphertexts (not c)
-
- The diagram shows an adversary sending a message with `dest = 25` and `data` to a server. The server returns the `data` part back to the adversary.
- Often, adversary can learn partial information about plaintext
-
- The diagram shows an adversary sending a `TCP/IP packet` to a server. The server returns an `ACK` if the `valid checksum` condition is met.

攻击者可以欺骗服务器解密特定的密文或从加密包中学习到一些关于明文消息的内容

2、Chosen ciphertext security

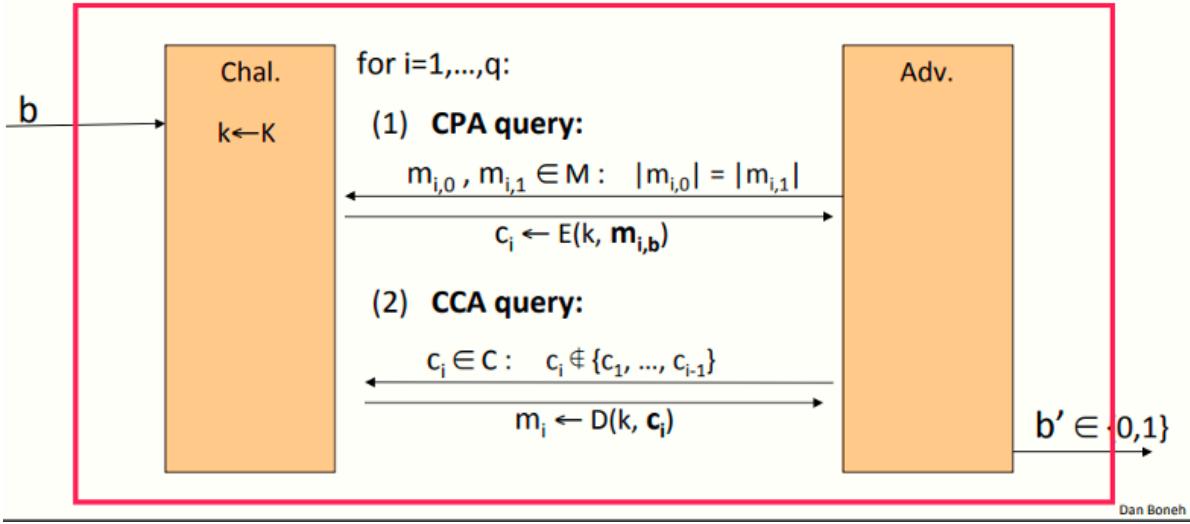
攻击者的能力：可以选择CPA或CCA，亦即

- 可以获得其任意选择的消息的加密
- 可以解密其选择的密文文本（除了一些挑战密文之外）

攻击者的目地：攻破语义安全

3、Chosen ciphertext security: definition

$E = (E, D)$ 为一定义在 (K, M, C) 上的密码，对于 $b=0, 1$ ， 定义事件 $\text{EXP}(b)$ 如下：



对于攻击者而言，其可以发起至多q次查询，每次查询可以为CPA或CCA两种中的一种

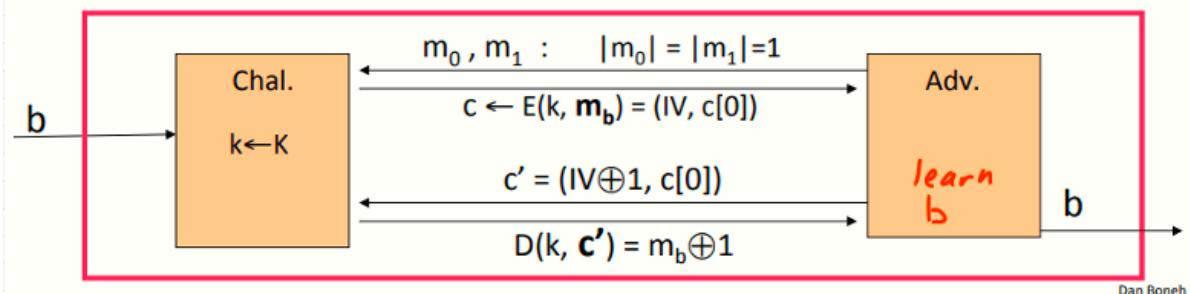
对于CPA而言，攻击者选择两条等长消息，挑战者返回 m_0 或 m_1 的加密（取决于挑战者选择的事件）

对于CCA而言，攻击者提交密文并获得其解密，即其可以获得他选择的任意密文的解密，唯一的限制是选择的密文不能来自于CPA的查询结果，否则不公平，这意味着攻击者可以很轻易地获得一个合法密文（挑战密文，即 m_0 或 m_1 的密文），解密后可以很轻易地分辨其是 m_0 或 m_1

定义：若E为CCA安全的，则其对于任意高效攻击者A，其如下优势可忽略

$$Adv_{CCA}[A, E] = |\Pr[EXP(0) = 1] - \Pr[EXP(1) = 1]|$$

看个例子：使用随机IV的CBC不是CCA安全的，模型如下



假设攻击者提交两个消息 m_0 和 m_1 ，长度均为1块（1 block CBC length），之后挑战者会返回该块使用的IV和密文

之后攻击者简单修改收到的密文，将其中的IV修改为 $\text{IV} \oplus 1$ ，其他不变并将新的密文 c' 提交给挑战者（由于修改了密文内容，不再是原来的挑战密文，因此合法），由前几节提到的xor计算的特性，解密结果为 $m_0 \oplus 1$ 或 $m_1 \oplus 1$ ，从而攻击者可以轻易分辨挑战者选择的事件，从而赢得安全游戏

4、Authenticated enc. \Rightarrow CCA security

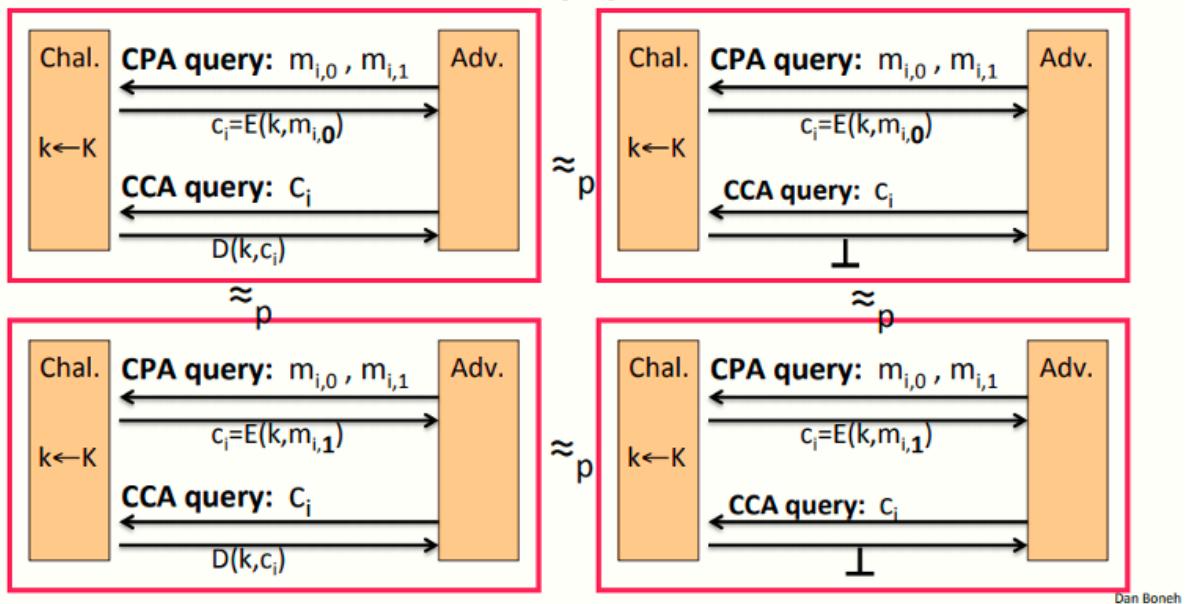
定理：记 (E, D) 为一提供AE的密码，则 (E, D) 为CCA安全的

具体而言，对于有至多q次查询的高效的对手A，存在高效的对手B1，B2满足如下不等式

$$Adv_{CCA}[A, E] \leq 2q * Adv_{CI}[B_1, E] + Adv_{CPA}[B_2, E]$$

上述不等式中，由于该方案具有认证加密，右侧第二部分是可忽略的，因此是CPA安全的，第一部分也是可忽略的，因此具有密文完整性，综上两个条件，故在赢得CCA安全游戏的也是可忽略的

证明如下



上述左侧两个分别代表挑战者选择事件0和事件1，稍微改变一下挑战者的输出，使其不会输出CCA查询的解密，而是总是输出 \perp 元素，因此每当攻击者提交CCA查询时，挑战者总是输出 \perp

因为方案具有密文完整性，攻击者无法生成不在 $C_1 C_{i-1}$ 中的密文，因此不会解密为bottom元素以外的结果

由于密文完整性，每一个攻击者选择的密文查询都会导致挑战者输出 \perp ，若攻击者实际上可以区分左边和右边的游戏模型，这意味着它可以在某个时刻请求一个解密到bottom元素之外的查询，并用该查询打破密文完整性

由于密文完整性，这些左右博弈是无法区分的

总结一下，即攻击者的选择密文的询问总是以同样的方式回复，不会给攻击者任何的信息，攻击者总会收到 \perp 的回复，而 \perp 意味着不会给攻击者任何的信息，因此可以将上述模型去掉CCA查询，从而模型变成了CPA查询，又因为认证加密是CPA安全的，因此图中右侧上下两个游戏无法区分，因此所有游戏等价，因此为CCA安全

5、总结

认证加密需要确保机密性来抵御一些活跃的攻击者（可以解密一些密文的攻击者）

结论：

- 不抗重放
- 如果泄露了为什么拒绝的信息（比如计时攻击）那么认证加密也将是不安全的

W4 7-4 Constructions from ciphers and MACs

1、history

认证加密（Authenticated Encryption, AE）最早在2000年提出

但在此之前，一些加密库如MS-CAPI都是单独支持CPA安全和MAC的，这意味着CPA安全加密函数和MAC函数需要单独调用，因此对于不同的开发者有不同的方式，让这两个函数结合在一起以实现认证加密

问题：这些认证加密在当时并没有被定义，因此当时的开发者并不知道哪些是正确的，也不是所有的方法都是正确的

最常见的错误的方式：将加密和完整性机制合并

2、Combining MAC and ENC (CCA)

Encryption key k_E . MAC key = k_I

Option 1: (SSL)

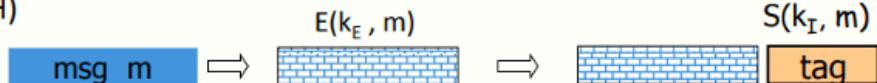


Option 2: (IPsec)

always
correct



Option 3: (SSH)



Dan Boneh

记两个独立的密钥，加密密钥为 KE ，MAC密钥为 KI ，两个密钥均在会话建立时生成，有上图所示的CPA安全加密和MAC合并的三个例子

- SSL协议 (mac-then-enc) : 先MAC再加密，即先用 KI 计算MAC，然后附在消息 m 的后面，再将两者作为一个整体使用 KE 加密
- IPsec协议 (enc-then-mac) : 先对消息进行加密，再对密文计算MAC
- SSH协议 (enc-and-mac) : 先对消息加密，然后在尾部附上对消息的MAC (不是对密文的MAC)

SSL的做法：典型的错误，很容易被CCA，因此不能保证AE（导致CCA的原因：一些加密算法和MAC算法之间可能产生冲突）

对于IPsec的方案而言，总是正确的，无论使用哪种CPA安全的方案和MAC方案，IPsec的方案总能提供一个正确的认证加密

IPsec的方案：首先先对消息加密，因此消息会隐藏在密文中，之后计算MAC，MAC确保没有人能提供一段有效的而内容不同的密文，该方案确保了任何对密文的修改都会被发现（MAC校验不通过）

对于SSH协议的方案而言，MAC并不保证消息的机密性，仅确保完整性，实际上若仅输出明文中的若干bits作为MAC理论上没错，但是会完全破坏CPA安全性，因为明文中的若干bits在密文中被泄露了，该方案对于SSH来说没问题，但不建议使用，因为MAC的输出会泄露明文中的某些bits

3、A.E. Theorems

记 (E,D) 为一CPA安全加密， (S,V) 为一安全MAC，则有如下定理

- Enc-then-MAC方式：总是可以提供AE
- MAC-then-Enc方式：在CCA下可能不安全

值得注意的是，若Enc采用rand-CTR模式或rand-CBC模式，M-then-E方式也可以提供AE

4、Standards

AE开始流行后，出现了一些新的标准以合并加密和MAC方式，有下列三种典型代表

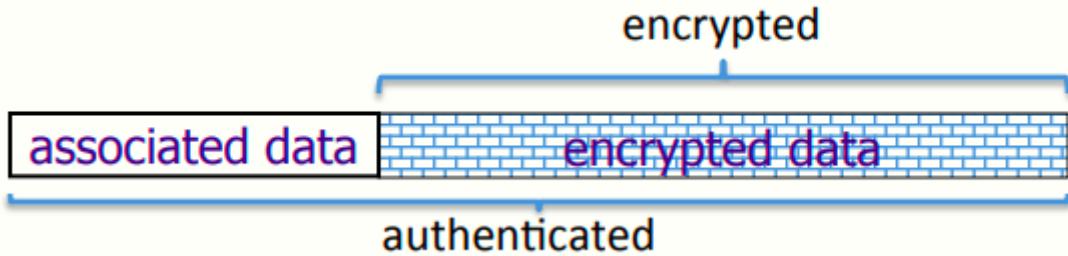
1. GCM：使用CTR模式加密，CW算法生成MAC，效率很高，加密时间基本取决于CTR模式的加密时间。
2. CCM：使用CBC-MAC然后使用CTR模式加密，即本质是MAC-then-Enc（由上一节提到的，因此使用CTR加密是可以提供AE的），且CCM是完全基于AES的方案（CBC-MAC和CTR模式均使用AES，

只需要一个AES模块和少量的代码即可实现），802.11i标准在使用

3. EAX: CTR模式然后CMAC

三种模式的比较：

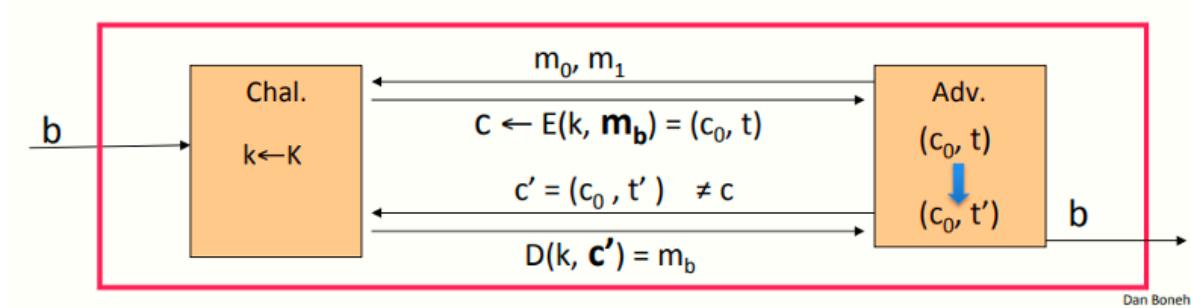
1. 都是基于nonce，即并不使用任何随机，但是需要一个nonce作为输入，每个密钥与其对应的nonce都是独一无二的（由于nonce不必是随机的，因此计数器是个很好的生成的nonce的机制）
2. 支持附加数据的认证加密（A.E. with associated data）：一种对AE的扩展，常见于各种网络协议，即明文消息不会被完全加密，仅有一部分会被加密，但整条消息可以被确保是正确的



如IP包，AEAD确保头部不会被加密（加密了无法路由），因此仅加密数据部分，同时可以确保头部数据是正确的（可信的）

5、MAC Security -- an explanation

MAC的安全性： $(m, t) \not\Rightarrow (m, t')$ 确保了AE，但如果 $(m, t) \rightarrow (m, t')$ ，即可以对同一条消息生成不同的MAC，啧可能导致一个不安全的Enc-then-MAC加密方式，攻击如下

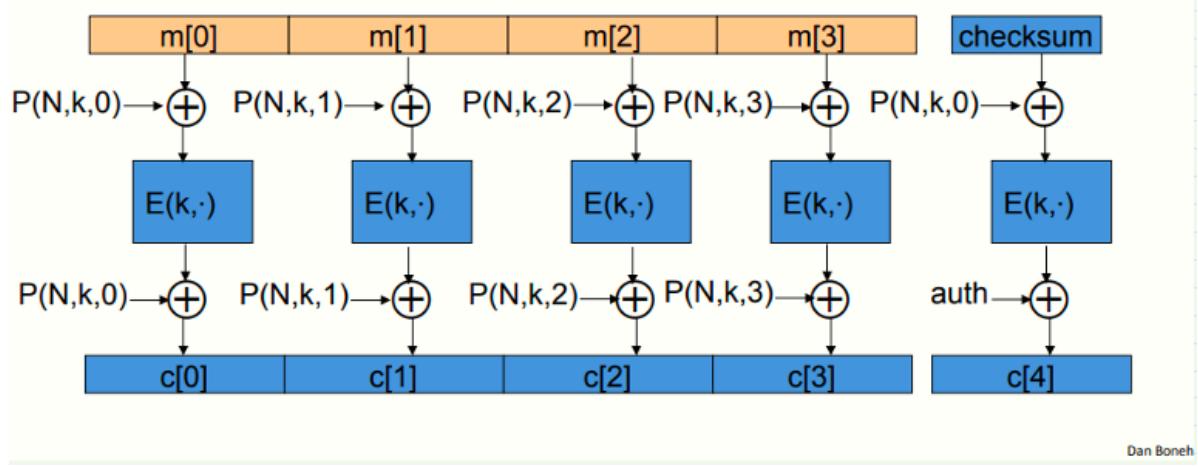


假设攻击者先收到了一条消息的密文及其对应的MAC，即 (c_0, t) ，若其有能力构造多条MAC，因此可以构造 (c_0, t') 并提交给挑战者，通过验证后挑战者会返回对应的明文，因此攻击者区分两个事件的优势为1

6、OCB: a direct construction from a PRP

AE的概念变得正式且严谨之后，人们开始思考更高效的AE方式

假如将CTR模式和CMAC绑定，对于每段明文，现在CTR下加密，再生成CBC-MAC，因此需要两次块加密，有无更高效率的方式



OCB模式：如上图所示，计算全部并行，且仅需要执行一次块加密即可

为一需要执行多次的为一个简单的函数P，接收nonce和密钥作为输入，同时接收块计数作为输入，需要在每块的块加密函数的前后分别执行该P函数

好用东西为什么不用？OCB有一些专利问题，因此最终没成为标准

7、Performance

使用Crypto++ 5.6.0 [Wei Dai]

GCM采用高速的Hash函数，并采用CTR模式加密，开销小

CCM和EAX使用块加密和基于块加密的MAC，因此大约是CTR模式的两倍慢

OCB最快，因为对于每一块明文仅执行一次加密

AMD Opteron, 2.2 GHz (Linux)

<u>Cipher</u>	<u>code size</u>	<u>Speed (MB/sec)</u>		
AES/GCM	large**	108	AES/CTR	139
	smaller	61	AES/CBC	109
	smaller	61	AES/CMAC	109
AES/OCB		129*	HMAC/SHA1	147

* extrapolated from Ted Kravitz's results ** non-Intel machines

Dan Boneh

W4 7-5 Case study: TLS

1、The TLS Record Protocol (TLS 1.2)

TLS协议中用于数据加密的协议为TLS记录协议，每个记录包含一个首部和其加密的数据，模型如下



TLS使用单向密钥，表明C到S和S到C的消息是不同的密钥，即解密C来的消息和加密发送给C的消息是两个不同的密钥，两个密钥均由TLS密钥交换协议生成

有状态的加密：对于每个数据包的加密，服务器和客户端内部均维护一个特定的状态，即一对64 bits的计数器，服务器和客户端各一对，会话开始时初始化为0，每当发送或收到一个数据包计数器+1，这两个计数器可以记录两端的流量

计数器的主要目的：防止重放攻击

2、TLS record: encryption

套件使用 AES-CBC加密，HMAC-SHA-1生成MAC

以客户端为例，密钥 $k(b \rightarrow s)$ 包含两个子密钥，分别用于生成MAC和加密使用，同样由TLS密钥交换协议生成

$$k_{b \rightarrow s} = (k_{mac}, k_{enc})$$



TLS包如上图右侧所示，以客户端为例，加密算法接收三个输入（C到S的密钥，数据，计数器当前状态），具体分下述几个步骤

Browser side $enc(k_{b \rightarrow s}, data, ctr_{b \rightarrow s})$:

- step 1: $tag \leftarrow S(k_{mac}, [++ctr_{b \rightarrow s} || header || data])$
- step 2: pad [header || data || tag] to AES block size
- step 3: CBC encrypt with k_{enc} and new random IV
- step 4: prepend header

1. tag生成：生成算法接收 k -mac密钥，计数器状态，首部，数据作为输入，其中计数器状态仅作为tag生成的输入，实际上不传输（双方均维护计数器状态）
2. 数据扩展：将数据扩展为AES块的整数倍
3. 加密：用 k -enc子密钥加密，并使用一个新的随机IV
4. 添加首部：添加首部（明文）

可以看到，TLS采用MAC-then-Enc模式，但加入了一些小小的改进，即在MAC中加入了计数器的状态，从而可以有效防止重放攻击

3、TLS record: decryption

以服务器端为例，解密算法接收三个输入（C到S的密钥，数据，计数器），具体步骤如下：

1. 解密：使用k-enc解密记录中的数据部分
2. 检查扩展：检查数据扩展部分是否合法，若不合法返回错误MAC记录信息并终止连接
3. 检查tag：tag错误同样返回错误MAC记录信息并终止连接

其中bad_record_mac意味着输出bottom元素，即TLS记录协议提供AE，且一旦错误就发出非法信息，攻击者无法得知具体是由什么原因导致的错误

上述步骤全部执行完毕且正确后，将首部和扩展去除，从而得到明文消息

如果攻击者在一段时间后进行重放，由于每条消息的计数器的值仅会出现一次，因此旧的消息的MAC包含了旧的计数器的状态，若使用重放攻击，会导致MAC检查时不通过（不是期望的计数器状态）

4、Bugs in older versions (prior to TLS 1.1)

TLS早期版本存在的一些缺陷

(1) 可预测的IV（链式IV）：在TLS 1.0或更早，下一条记录的IV是上一条密文的最后一块，导致若攻击者可以获取当前记录，就可以知道下一条记录使用的IV，从而打破上一条记录的语义安全

该缺陷在CPA下不安全，由于这个缺陷导致的非语义安全，有更强力的攻击（BEAST attack）可以破解TLS记录的开始部分

这个缺陷在1.1版本改为使用随机的不可预测的IV代替链式IV

(2) 填充提示：在TLS 1.0或更早版本中，如果密文被拒绝时，如果是无效的填充部分，服务器会发送警告消息表示解密失败，如果是MAC不正确则返回MAC不正确警告

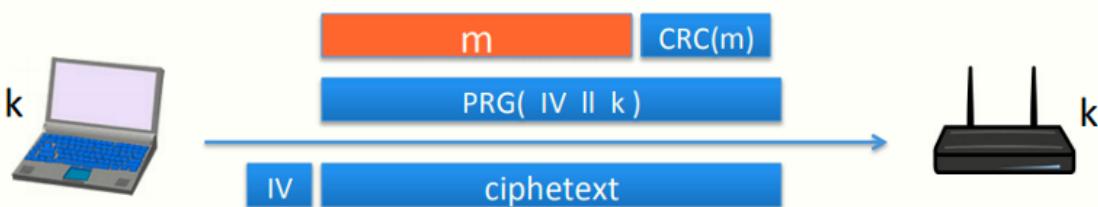
攻击者可以通过观察服务器返回的警告信息来分辨密文中填充部分是否有效，从而导致填充攻击，TLS 1.1中改了解密失败也会返回MAC不正确，攻击者无法分辨为何错误

小知识：一般的网络协议，如ICMP、HTTP等，会携带一些控制信息，希望通信双方都能知道产生错误的原因，但在密码学的应用中，若解释了导致失败的原因则大概率会留下被攻击的漏洞，最简单的做法就是出错了直接拒绝，但不解释原因

5、802.11b WEP: how not to do it

802.11b WEP，一个蹩脚的协议，做错了几乎所有的事情，且不提供AE，模型如下

802.11b WEP:



工作流程：笔记本连上接入点后，若要发送消息给接入点，需要计算该消息的一个CRC并附在消息后面，之后使用流密码（RC4）处理消息，密钥为每个包都不同的IV和长期不变的key k

存在的问题：

- 若IV重复的话可以进行二次密码本攻击
- 如果仅仅是IV变化而k不变的话，PRG的密钥彼此非常相似，导致不安全，从而导致WEP不安全

6、Leaking the length

TLS首部会泄露TLS记录的长度，从而可以推断出网络流量

对于许多网络应用而言，泄露数据包长度意味着泄露敏感信息：

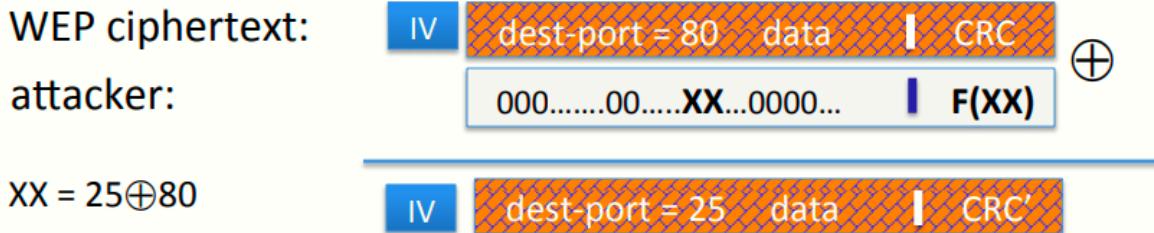
- 税务相关网站：长度表示申报表的类型，从而泄露用户收入信息
- 健康相关网站：长度泄露了用户查阅的文件页数
- 谷歌地图：长度泄露了查询请求的信息

7、Active attacks

CRC的特点：线性的，意味着若给定一个消息m的CRC，要求计算 $m \oplus p$ 的CRC，只需要计算一个函数 $F(p)$ ，然后将其与m的CRC异或即可

$$\forall m, p: \text{CRC}(m \oplus p) = \text{CRC}(m) \oplus F(p)$$

假定一个数据包的目的地是AP，该包目的端口为80，且攻击者也知道该包需要发至端口80，攻击者希望修改端口为25以读取该包内容



构造如上数据包， $XX=25 \oplus 80$ ，并构造 $F(XX)$ ，通过上述公式修正原数据包的CRC，由于流密码的特性，解密时会将原来的80修改为25，且CRC校验不会发生错误（即便是CRC被加密了，也可以通过异或 XX 对应的F函数处理后的密文即可）

上述攻击表明，攻击者即便不知道CRC的值也可以利用其线性特性对其进行攻击，因此CRC并没有提供完整性以对抗一些攻击，且CRC不应使用在有完整性机制的方案中，完整性需要MAC来提供而非一个自组织的机制（比如CRC）

W4 7-6 CBC paddings attacks

1、Recap

Authenticated encryption：提供CPA安全和密文完整性

- 确保在活跃的攻击者下的保密性
- 确保不能再不被发觉的情况下修改密文内容

常用标准：GCM、CCM、EAX

常用方案：Enc-then-MAC

2、The TLS record protocol (CBC encryption)

上一讲提到，早期的TLS在解密中可能会产生两种不同的错误（填充错误或MAC错误）

若攻击者截获一条密文并尝试破解，他可以将这个数据包原样提交给服务器，服务器解密后如果出现了不同的警告，攻击者就可以知道加密的密文中最后几个字节是否是有效的填充（如果不是则会返回填充错误，如果填充正确则会返回MAC错误），因此导致攻击

3、Padding oracle via timing OpenSSL

旧版本TLS存在上述缺陷从而导致攻击，SSL通过返回相同的错误类型防止攻击者从错误提示中得到信息
Brice Canvel和他的小伙伴发现了一个攻击（OpenSSL 0.9.7a中修复），攻击如下

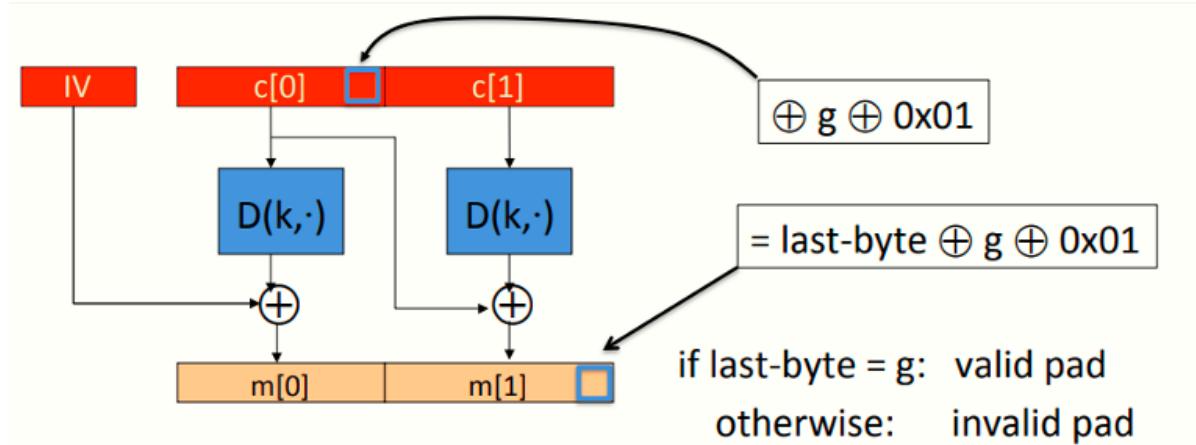
Canvel团队发现TLS解密先解密记录，检查填充，如果填充无效则终止解密并生成错误信息，如果填充有效则检查MAC，MAC错误则生成错误信息，两个错误信息生成的间隔可能导致时序攻击，如图说是



若填充错误，则错误信息生成的更快，而MAC错误导致的错误信息需要花费稍长时间，即便是返回同样的错误信息，攻击者观察生成错误的时间还是可以知道具体是哪种错误

4、Using a padding oracle (CBC encryption)

假设攻击者有密文 $c = (c[0], c[1], c[2])$ ，且希望获取明文 $m[1]$ ，假设使用CBC模式，有如下模型



其中 g 为猜测 $m[1]$ 的最后一字节，将 $c[0]$ 的最后一个字节xor g xor $0x01$ ，则根据CBC模型，解密后得到的 $m[1]$ 的最后一字节为原本的最后一字节与 g 和1的xor，即 $= \text{last-byte} \oplus g \oplus 0x01$ ，若猜对了，即 $\text{last-byte} == g$ ，则解密后的最后一字节应该为 $0x01$ ，且为正确的填充

利用这个机制，攻击者向填充算法提交(IV, $c[0]$, $c[1]$)，最多尝试256次后可以知道 $m[1]$ 的最后一字节的信息，并且通过将 $0x01$ 替换为($0x02, 0x02$)，同理可以获得第二字节的信息，分组为16字节，最多需要 $16 * 256$ 次即可得到分组全部明文信息

5、IMAP over TLS

上述攻击的问题在于，TLS在收到无效的填充或MAC时会直接终止连接并协商一个新的密钥来继续，攻击者此时只能得到上一个密钥加密的密文且该密钥不会再使用，因此上述攻击虽然泄露了一些明文信息($m[1]$ 的最后一字节)，但是没有完全泄露所有信息

但是IMAP协议（通常运行在TLS上），客户端会每5分钟向服务器检查是否有新的邮件，检查邮件需要登陆服务器并发送一条包含账号和口令的信息，这意味着攻击者每5分钟就会收到同一条消息的加密

假设口令只有八个字节，即便是每次只能猜一个字节，攻击者也可以在几个小时内搞定口令

6、总结

- TLS如果采用了Enc-then-MAC而不是MAC-then-Enc可以避免这个问题，因为MAC会先检查，若MAC检查未通过则直接丢弃密文
- MAC-then-CBC提供AE，但填充提示的机制破坏了AE机制

7、例题

Will this attack work if TLS used counter mode instead of CBC?
(i.e. use MAC-then-CTR)

- Yes, padding oracles affect all encryption schemes
- It depends on what block cipher is used
- No, counter mode need not use padding ↙
-

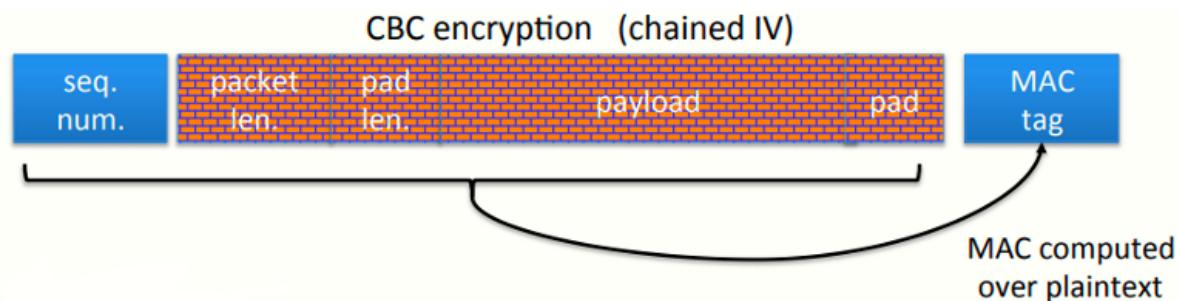
如果TLS使用CTR代替CBC，且仍使用MAC-then-Enc，上述攻击是否还有效？

不会，CTR模式不需要填充

W4 7-7 Attacking non-atomic decryption

1、SSH Binary Packet Protocol

SSH二元数据包协议，用于SSH的客户端与服务端进行密钥交换的阶段，和前两节提到的一样，使用Enc-and-MAC方案，协议包如下（红色部分为加密数据）



协议数据包包含序列号、数据包长度、填充长度、载荷、填充、MAC六项

图中的红色部分采用CBC加密且使用链式IV，根据前几节的内容，链式IV在CPA下不安全

注意到MAC计算的是明文消息的MAC而非密文消息，由于MAC没有保密性要求，传输明文消息的MAC可能会暴露明文消息的一些内容

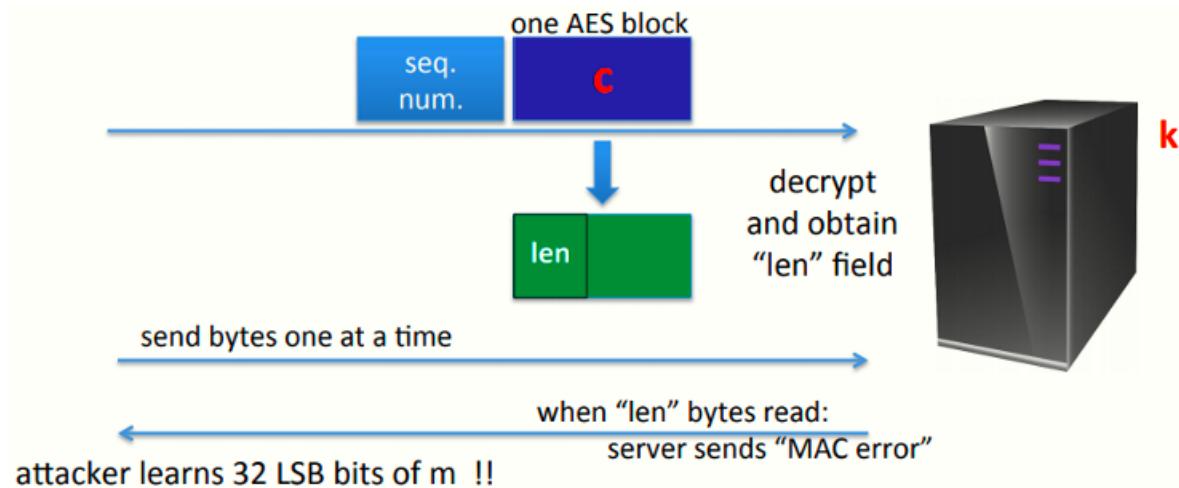
以上都不是重点，下面的攻击才是，在此之前先看看SSH的解密流程：

1. 服务器先单独解密数据包长度字段
2. 根据1中解密的字段，从网络中读取特定数量的数据
3. 解密读取到的剩余部分
4. 检查MAC，若MAC错误则发送错误消息

分析：注意到数据包长度部分，解密后没有任何的验证并直接决定了读取的数据的长度，实际上由于还没有恢复整个数据包，因此无法检查MAC的正确性，但SSH在验证MAC之前就使用了这个长度字段，从而导致一个攻击

2、An attack on the enc. length field (simplified)

简化的攻击流程：假设攻击者截获了一块直接由AES加密的消息块 $c = \text{AES}(k, m)$ （该消息长度只有1 AES块），并期望解密出 m ，模型如下



攻击者会给服务器发送一个正常启动的数据包，并在内注入截获的密文，将其作为发送到服务器的第一个密文块

服务器会解密这个包的前几个字节以获取数据包长度，并在检验MAC前，根据这个长度接收数据

攻击者要做的是，每次给服务器发一个字节，发一个服务器读一个，然后再发再读，直到服务器读取指定长度的数据并检验MAC

攻击者只需要发送垃圾数据，由于垃圾数据一定不会通过MAC验证，攻击者计算发送到服务器的字节数，一旦等待到服务器返回MAC错误信息时，就可以知道服务器在发送MAC错误信息前接收了多少字节，即知道了密文 C 的前32 bits 所对应的明文（数据长度字段占4字节）

3、总结

SSH的问题在于：

- 解密操作的非原子性：解密算法并没有以整个数据包作为输入并输出完整的明文（或者拒绝），而是分阶段的解密数据包的不同部分，非原子性的操作在加解密过程相当危险，SSH中的非原子性破坏了AE
- 验证前使用：数据长度字段还未经过验证就直接使用，若要使用则应该先验证

例题：若重新设计SSH，应该做出什么最小的改变（多选）

- ⇒ ○ Send the length field unencrypted (but MAC-ed)
- Replace encrypt-and-MAC by encrypt-then-MAC
- ⇒ ○ Add a MAC of (seq-num, length) right after the len field
- Remove the length field and identify packet boundary by verifying the MAC after every received byte

1. 可以向TLS一样将长度以明文发送，至少攻击者无法发起CCA
2. 替换更好的加密方案没用，数据长度字段仍然会在验证前被使用
3. 在数据包长度后面添加一个seq和length的MAC，读取完length后会检查其有效性
4. 检查数据包边界，理论上可行，实际上很容易被DDOS

4、教训

- 永远不要实现或设计自己的AE系统，使用GCM之类的标准
 - 若因为某些原因不使用标准而必须自己实现，并使用Enc-then-MAC方案，确保不要导致上述错误，即验证前使用
- 5、推荐几篇论文

- The Order of Encryption and Authentication for Protecting Communications, H. Krawczyk, Crypto 2001.
- Authenticated-Encryption with Associated-Data, P. Rogaway, Proc. of CCS 2002.
- Password Interception in a SSL/TLS Channel, B. Canvel, A. Hiltgen, S. Vaudenay, M. Vuagnoux, Crypto 2003.
- Plaintext Recovery Attacks Against SSH, M. Albrecht, K. Paterson and G. Watson, IEEE S&P 2009
- Problem areas for the IP security protocols, S. Bellovin, Usenix Security 1996.

1. 讨论了加密和验证的顺序问题
2. 讨论了OCB模式，一种非常有效的建立AE的方法，其中讨论了OCB的一个变体

后三个时针对文件的攻击，#3讨论了上一节的填充提示，#4讨论本节的长度攻击，#5讨论了针对加密的攻击，这些攻击仅讨论CPA安全，而未加入完整性

3. 中提供了一些很好的例子来说明为什么CPA安全性本身不应被用于加密，唯一允许使用的时经过身份验证的加密以确保机密性，或者无需机密性只需完整性可以使用MAC

W4 8-1 Key Derivation

1、Deriving many keys from one

实际使用时，经常会出现由一个密钥推导出多个密钥的场景

典型场景：源密钥（source key, SK）来自于某些硬件随机数生成器，或来源于密钥交换协议

实际上，为了确保会话安全，需要多个密钥（如TLS中不同方向的数据采用不同的密钥，每个方向又分为加密密钥和MAC密钥和IV），基于nonce的CBC加密也需要多个密钥

目标：通过一个源密钥生成多个密钥，即 $\text{SK} \rightarrow k_1, k_2, k_3, \dots$

方式：Key Derivation Function，KDF

2、When source key is uniform

有一安全PRF F，其密钥空间为K，输出空间为 $\{0,1\}^n$

假设SK为密钥空间K中一致的（？）随机密钥，则定义KDF如下

KDF(SK, CTX, L) :=
 $F(\text{SK}, (\text{CTX} \parallel 0)) \parallel F(\text{SK}, (\text{CTX} \parallel 1)) \parallel \dots \parallel F(\text{SK}, (\text{CTX} \parallel L))$

接受三个输入：

- SK：来自于PRF F
- CTX：一个唯一标识应用程序的字符串，如果说系统中跑了很多程序且需要生成各自的密钥，CTX 可以用于区分这些，即便使用同一个SK这些程序也能生成不同的密钥
- L：length，一个长度

KDF会计算0的PRF、1的PRF、……L的PRF，之后用这些输出的一些位作为想要生成的密钥

3、What if source key is not uniform

如果SK不是一致的，则PRF的输出其实看起来并不随机，攻击者可以预测一些密钥并攻击会话

SK不是一致的原因：

- 密钥交换协议通常会生成一个高熵密钥（high entropy key），但这个高熵密钥实际上只是密钥空间的一个子集
- 使用的硬件随机数生成器也可能会生成高熵字符串

4、Extract-then-Expand paradigm

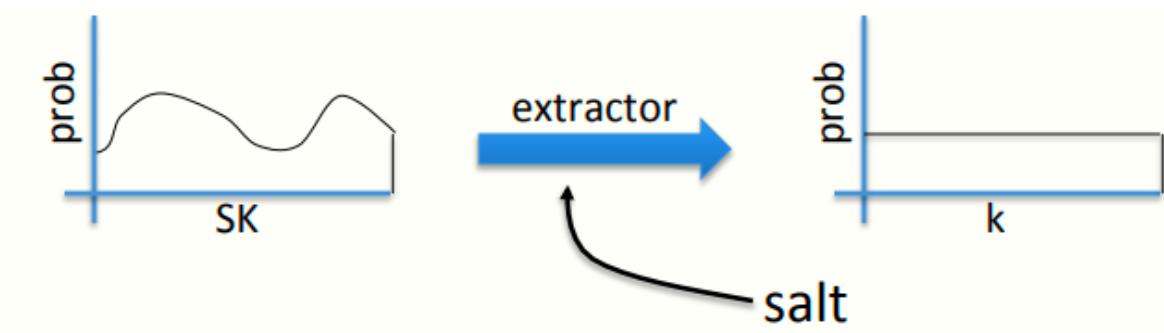
需要一些手段来解决上述问题，从而引出构造KDF的模式，即提取扩展模式，步骤如下

1. 从实际的SK中提取一个伪随机密钥：从图中来看的话SK并没有被均匀的分到密钥空间中，需要使用提取器将他均匀分布到密钥空间，提取器不一定产生均匀分布的输出，但生成的分布与均匀分布不可区分

提取器会输入一个盐（salt），目的是捣乱，无论输入的分布如何，输出分布依然与随机分布不可区分

盐是一个固定的值，无需保密可以公开，但是得确保是随机的

盐是用来防御可能会干扰提取器的恶意的坏的分布



2. 使用PRF扩展 k 至更多的位，直至满足会话密钥的需要

5、HKDF: a KDF from HMAC

HKDF: 密钥的提取和扩展均使用HMAC

提取步骤: $k \leftarrow \text{HMAC}(\text{salt}, \text{SK})$, 其中salt作为HMAC的key, SK作为HMAC的数据

扩展步骤: 用HMAC作为PRF生成需要的尽可能多的位

总结: 若获得了一个源密钥 (无论来自硬件还是密钥交换协议), 都不应该将源密钥直接用于会话密钥, 需要输入KDF中并获得需要的密钥

6、Password-Based KDF (PBKDF)

从口令中提取密钥, 由于口令通常有较低的熵, 不能直接用HKDF, 如果用了会很容易遭到字典攻击

PBKDF抵御低熵的方式: 盐和慢hash函数

慢hash函数 $H^c(\text{pwd} \parallel \text{salt})$: 接受盐和口令作为输入, 输出一个密钥, 实际上是将该ahsh函数迭代 c 次, 如 $c=一百万$, 使得攻击者对密钥的猜测更困难

相关标准: PKCS#5 (PBKDF1)

W4 8-2 Deterministic Encryption

确定性加密系统: 永远将特定消息转换成完全相同的密文的加密系统, 比如对同一消息加密三次, 每次都会得到完全相同的密文 (不涉及nonce)

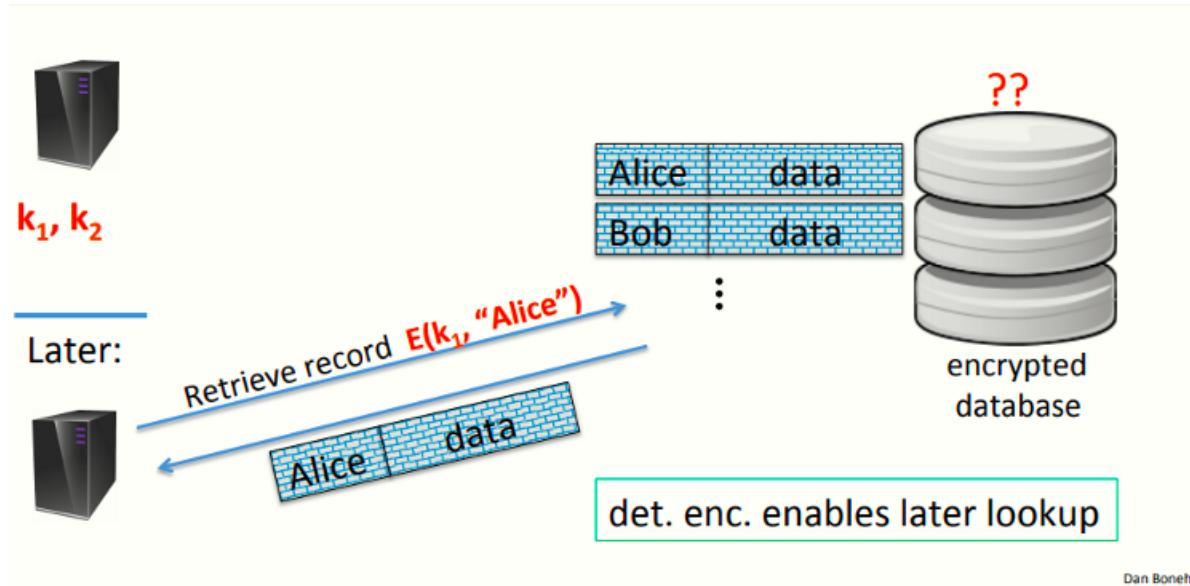
1、The need for det. Encryption (no nonce)

假设有一个在加密数据库中存储信息的服务器, 每条记录都有一个索引及在记录中保存的数据, 服务器会加密这个记录, 如图所示



注意到索引用 k_1 加密, 记录用 k_2 加密, 然后将加密的记录发送至数据库

一段时间后服务器想从数据库中检索记录时，服务器只需要向数据库发送服务器感兴趣的索引的加密，由于加密是确定性的，因此索引加密后生成的密文与之前的一样，从而可以在数据库中找到这个索引并向服务器返回结果

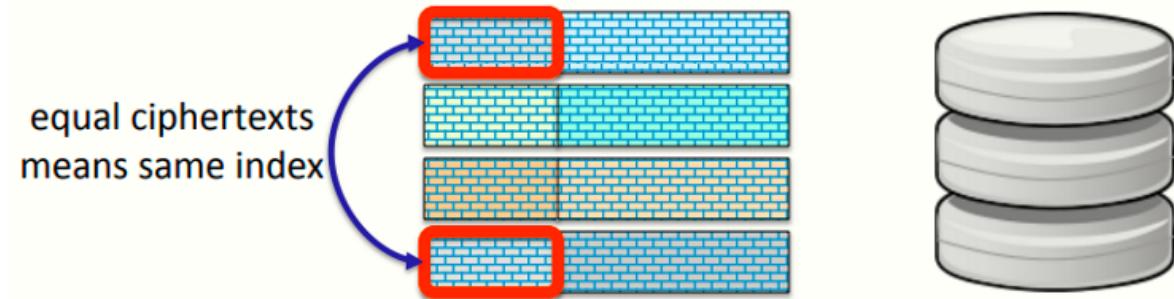


2、Problem: det. enc. cannot be CPA secure

优点：数据库对于存储于其中的数据完全不知情，因为服务器发送的是对加密索引的请求，数据库甚至不了解服务器检索了哪些记录

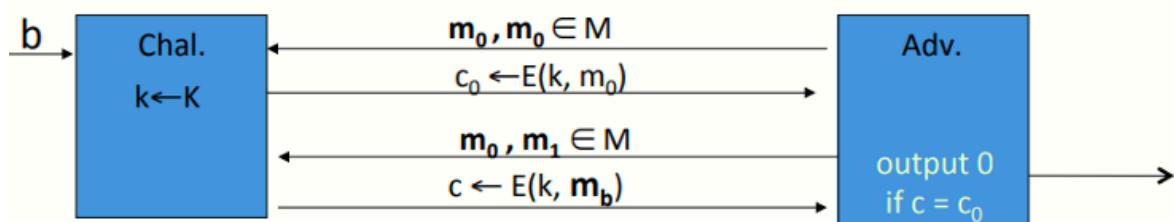
缺点：确定性加密没有CPA安全，因为攻击者可以查看各种密文，如果其发现了两条相同的密文，意味着其对应的明文也一定相同

若消息空间M很小的话，会导致攻击，换句话说，如果两个记录在索引位置正好有相同的密文，那么他就知道这两条记录对应着同一个索引



以形式化的方式说明确定性加密不能做到CPA安全，即攻击者可以以某些方式赢得CPA游戏，具体如下

Attacker wins CPA game:



首先，攻击者发送两个相同的消息 m_0 ，在以往的CPA游戏中，攻击者会得到其发送的左边或右边的消息对应的密文，但由于本次他发送的为两条相同的消息，因此一定会得到 m_0 的密文

下一步将发送消息 m_0 和 m_1 ，并得到其中一个的密文，由于采用的是确定性算法，因此在新发起的挑战中， m_0 的密文不会改变，攻击者只要检查收到的密文是否等于上一次挑战的 c_0 即可，若不等于则说明收到了 m_1 的密文 c_1 ，即攻击者的优势为1

3、A solution: the case of unique messages

解决方案：限制被同一密钥加密的消息种类

假设加密方永远不会使用同一密钥加密相同的消息，即密钥和消息的配对总是不同且用不重复，则对于每一次加密，则有消息变化、密钥变化，或者都会变化，但是不会用同一密钥加密同一消息两次

实际上这个事件很容易实现，若加密者在很大的消息空间内随机选择消息，且消息结构确保为某些特定结构

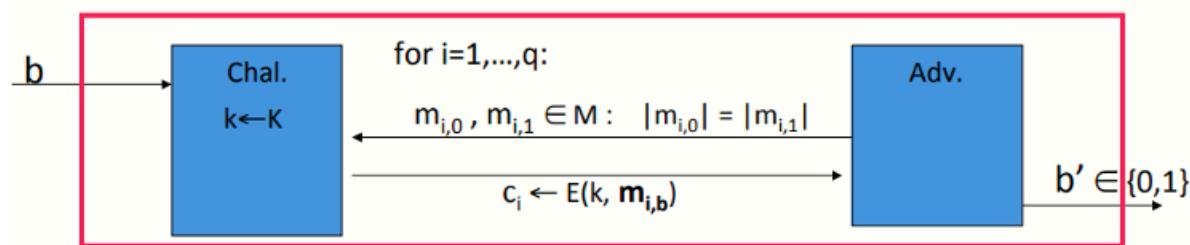
回到数据库的例子，假设加密的用户ID全是互不相同的，则每个索引均对应一个唯一的用户ID，在数据库中仅有一条记录，从而消息永远不会重复

4、Deterministic CPA security

有了消息不会重复的条件，就可以定义安全性了

记 $E = (E, D)$ 为一定义在三元组 (K, M, C) 上的密码， K 、 M 、 C 分别为密钥空间、消息空间、密文空间，记两个实验 $EXP(b)$ ，有如下模型

$E = (E, D)$ a cipher defined over (K, M, C) . For $b=0,1$ define $EXP(b)$ as:



where $m_{1,0}, \dots, m_{q,0}$ are distinct and $m_{1,1}, \dots, m_{q,1}$ are distinct

和标准的CPA安全模型几乎一致，但是需要额外注意的是，若攻击者总是得到左侧消息的密文（即实验0），则左侧的消息需要互不相同，即永远不能两次得到相同消息的密文，右侧的消息同理，因此攻击者永远不会得到用特定密钥多次加密相同消息的情况（因此上面的第2点中讲的攻击模型并不是一个标准的确定性CPA模型，该模型中攻击者收到了同一消息的多次加密）

除了额外注意的点，其他和CPA安全一样，因此有如下定义

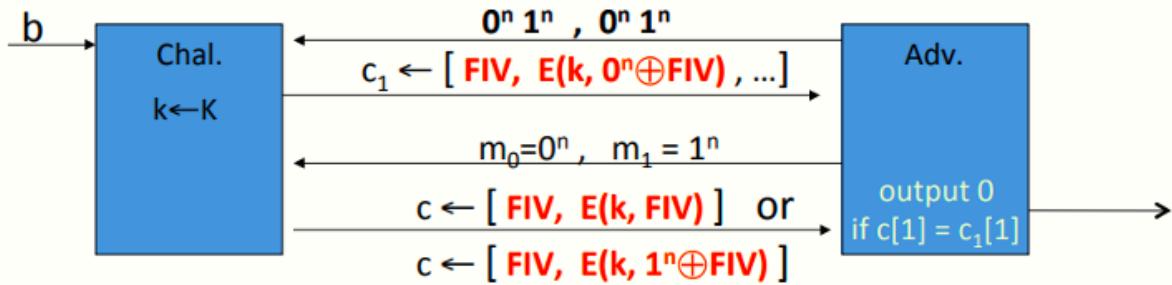
定义：若 E 为在确定性CPA下语义安全的密码，则其对所有高效攻击者 A ，其如下优势可忽略

$$Adv_{dCPA}[A, E] = |Pr[EXP(0) = 1] - Pr[EXP(1) = 1]| \leq negligible$$

5、A Common Mistake

需要注意的是，使用固定IV的CBC模式并不是确定性CPA安全的

记 $E: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ 为一使用CBC模式的安全PRP，具体模型如下



首先攻击者构造两个相同的数据，包含两块，均为前 n bits（第一块）为0，后 n bits（第二块）为1

之后攻击者收到对该消息的加密，包含固定的FIV，第一块的加密，第二块的加密等等

随后攻击者再构造两个长度为一块的消息，一个为全0另一个全1，随后发送给挑战者，挑战者返回的密文有两种可能：

- 若挑战者选择 m_0 加密，则其返回的结果为FIV的加密
- 若挑战者选择 m_1 加密，则其返回的结果为 $1^n \oplus$ FIV的加密

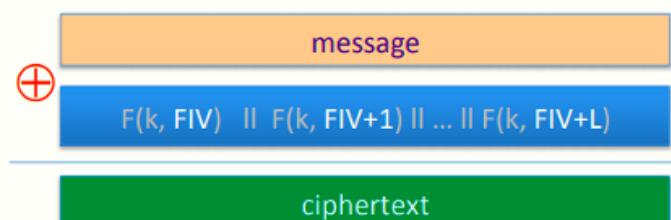
此时攻击者只需要判断收到的密文与第一次加密的第一块是否相等即可

总结：如果需要一致地加密数据库索引，不要使用固定IV的CBC模式

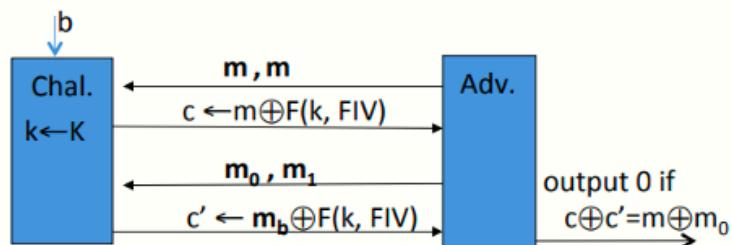
6、小练习

如果使用固定IV的CTR模式，方案是否为确定性CPA安全？

Is counter mode with a fixed IV det. CPA secure?



- Yes
→ No
 It depends



显然不是，因为这是一个一次性密码本，加密不同的消息会导致两次密码本

W4 8-3 Deterministic Encryption Constructions: SIV and wide PRP

1、Deterministic Encryption

上节课提到了确定性加密，如加密数据库索引时需要使用这种加密以确保写入数据库时的加密和之后的查询时的加密一致

问题在于，确定性加密在一般的CPA下是不安全的，因为若攻击者得到了两条一样的密文，则其可以知道对应的明文消息也一定一样

因此定义确定性CPA安全：只要加密同一消息不使用相同的密钥，即 (key, msg) 是独一无二的，则方案是安全的

具体的形式化定义见上节课，本节给出确定性CPA安全的构造

2、Constructions 1: Synthetic IV (SIV)

记 $E: K \times M \rightarrow C$ 为一CPA安全加密，其中 $E(k, m, r) \rightarrow c$, k 为密钥， m 为消息， r 为算法使用的随机性（即算法使用的随机字符串，若使用随机化CTR模式，则 r 为其对应的随机IV）， c 为密文

记 $F: K \times M \rightarrow R$ 为一安全PRF，作用为在消息空间中接受任意消息并输出串 R ，可以作为CPA安全加密方案中的随机性，即上述的 r 实际上是 R 中的一个元素

定义SIV如下：

$$\text{Define: } E_{\text{det}}(k_1, k_2, m) = \begin{cases} r \leftarrow F(k_1, m) \\ c \leftarrow E(k_2, m; r) \\ \text{output } r \end{cases}$$

使用两个密钥 k_1 和 k_2 加密消息 m ，首先将伪随机函数 F 作用于消息 m 从而得到CPA安全加密方案 E 的随机性 r ，然后使用 r 对 m 进行加密得到 c

定理： E_{det} 在确定性CPA下是语义安全的

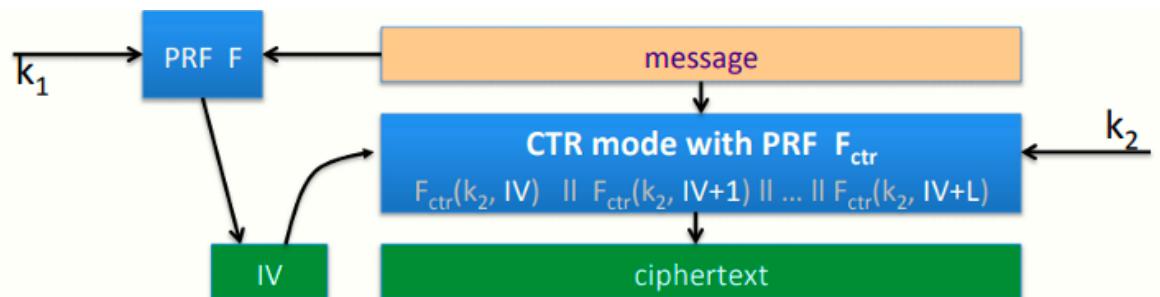
证明：由于每条消息都是独立的，因此经过伪随机函数 F 时得到的随机字符串也是不同的，因此CPA安全的加密方案 E 总是使用随机的字符串，由于这些 r s和新的字符串是随机不可区分的，因此系统实际上是CPA安全的

SIV更适合用于短消息，当然也可以适用于长消息（如多个AES块的消息），更牛的是SIV可以白嫖一个密文完整性，即不必使用特殊MAC，SIV有一个内置的完整性机制

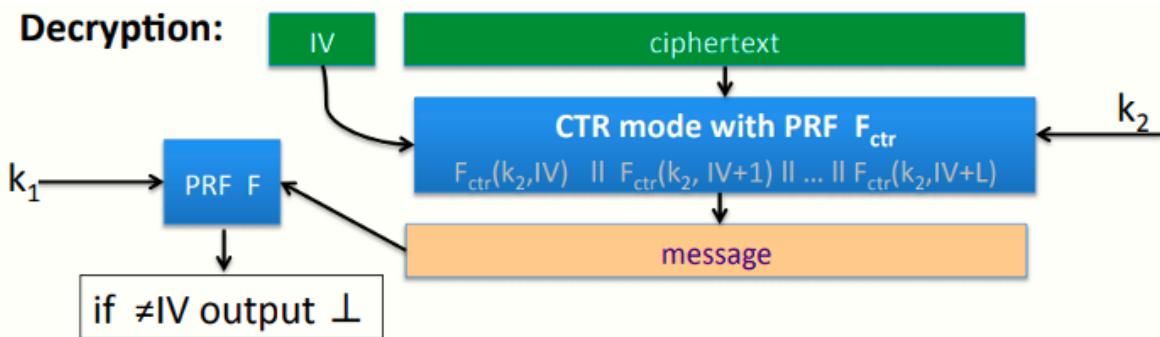
3、Ensuring ciphertext integrity

目标：实现确定性认证加密（DAE, deterministic authenticated encryption），即同时实现确定性CPA安全和密文完整性

考虑SIV的一个特例，SIV-CTR，即加密使用随机IV的CTR模式，模型如下



然后看一下解密



首先解密算法接受IV和密文作为输入，使用IV解密得到明文，然后将明文作为随机函数F的输入，若消息合法，则F的输出应当与IV相同，否则输出bottom元素

定理：若F为一安全PRF，且来自F-CTR的CTR为CPA安全的，则由F和F-CTR组成的SIV-CTR提供DAE

Thm: if F is a secure PRF and CTR from F_{ctr} is CPA-secure
then SIV-CTR from F, F_{ctr} provides DAE

4、Construction 2: just use a PRP

另一个简单的做法是直接用PRP (PRP和真随机可逆函数是不可区分的)

记(E, D)为一安全PRP， $E: K \times X \rightarrow X$

定理：(E, D)为确定性CPA下语义安全的

证明：记 $f: X \rightarrow X$ 为一真随机可逆函数，确定性CPA博弈中，对于实验0，所有左侧消息均不同，因此攻击者得到X中的q个不同的随机值，对于实验1，看到右侧的消息，由于两个实验对应的两个分布是相同的，因此无法区分实验0或实验1，因此直接使用PRP加密是确定性CPA安全的

因此如果想加密短消息（如一些少于16字节的消息），可以直接用AES且是安全的，即如果索引小于16字节时，不妨直接使用AES，但是需要注意的是这并不提供完整性

5、EME: constructing a wide block PRP

那如果消息比16字节长咋办？可以用SIV。但是如果还是想用PRP咋办

在之前的学习中，介绍了用小的消息空间的PRF构造了一个具有大的消息空间的PRF，本节介绍一个小的消息空间的PRF构造了一个具有大的消息空间的PRP，具体做法如下

记(E, D) 为一安全PRP， $E: K \times \{0,1\}^n \rightarrow \{0,1\}^n$

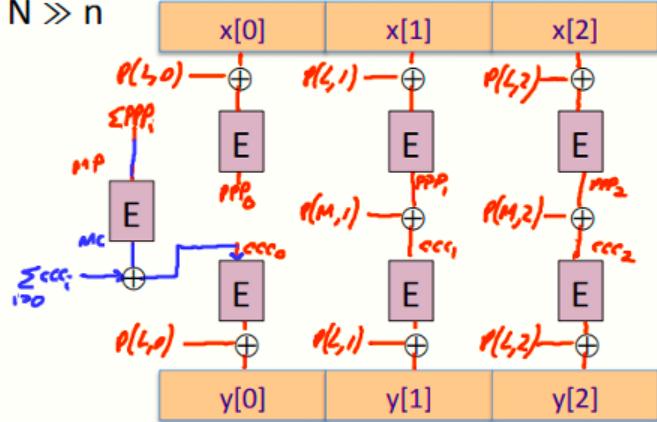
EME模式：构造一个PRP，在N bits块上运行，且N比n大得多得多，可以允许我们对比16字节大得多的消息进行确定性加密

EME模型如下图

EME: a PRP on $\{0,1\}^N$ for $N \gg n$

Key = (K, L)

$M \leftarrow MP \oplus MC$



Performance:

- can be 2x slower than SIV

Dan Boneh

EME使用K和L两个密钥（但实际上L是由K派生而来的）

- 首先将输入消息X分块，然后使用特定的填充函数对各个块异或，将异或结果作为加密算法E的输入，记输出的块为PPP_i
- 然后将所有的PPP们全部异或起来，结果记为MP
- 利用密钥K对MP进行加密，输出记为MC
- 然后计算M=MP⊕MC
- 利用M作为填充函数的输入，得到另一个填充，并与PPP们进行xor计算，得到输出ccc_i
- 将所有的ccc们全部异或起来，得到ccc₀
- 将ccc₀们作为E的输入，输出和填充函数P的输出再异或一次，得到密文

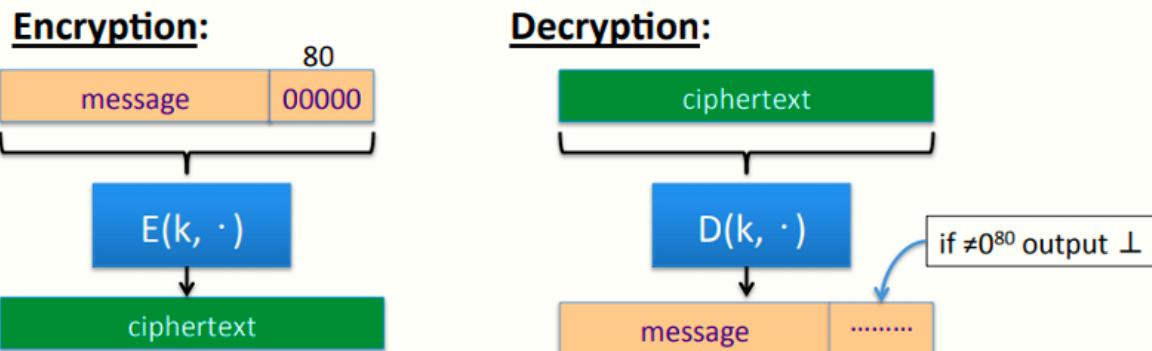
其中填充函数P为一固定的简单函数，执行仅需要非常短暂的时间

可以证明的是，若E为一安全的PRP，则EME结构也是一个安全的PRP

EME的优点在于其结构非常平行，即便看上去很复杂，但是确实可以并行加密，适用于多核处理器，效率大约是SIV的一半，因此比较适合加密短消息，而长消息选择SIV更好

6、PRP-based Det. Authenticated Enc.

如果想给这个基于PRP的机制添加完整性，是否可以使用PRP机制实现确定性认证加密？答可以，而且非常简单，方法如下

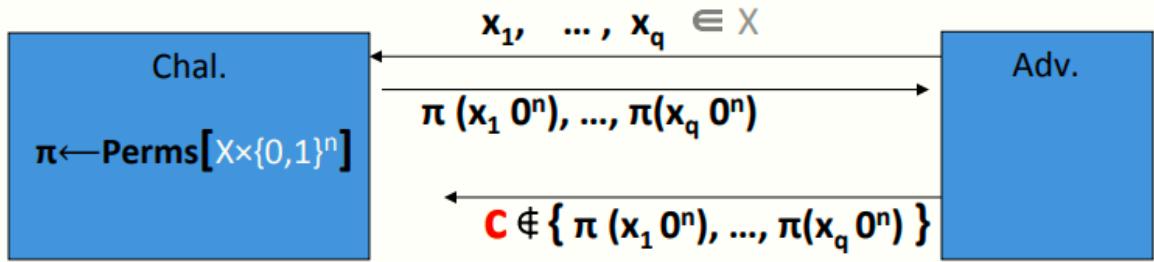


首先在原本需要加密的消息后附上好多的0（比如说80个0），然后直接用PRP加密得到密文，解密时只需要查看明文后面的80 bits是否全为0即可，若不全为0则输出bottom

记(E, D)为一安全PRP， $E: K \times (X \times \{0,1\}^n) \rightarrow X \times \{0,1\}^n$

定理：若 $1/2^n$ 为一可忽略的数，则基于PRP的加密提供确定性认证加密

证明思路：游戏模型如下图



$$\text{But then } \Pr[\text{LSB}_n(\pi^{-1}(C)) = 0^n] \leq 1/2^n$$

假设挑战者选择的是真随机置换 (truly random permutation) 并且输入空间为原消息空间再串联n个0, 攻击者发送q个消息并收到q个消息拼接上n bits 0的加密

此时, 若攻击者想要生成一个新的密文C, 且C不属于收到的任何一个密文, 并且期望C能够正确解密 (即解密结果的最后n bits为全0) , 其概率小于等于 $1/2^n$

W4 8-4 Tweakable encryption

1、Disk encryption: no expansion

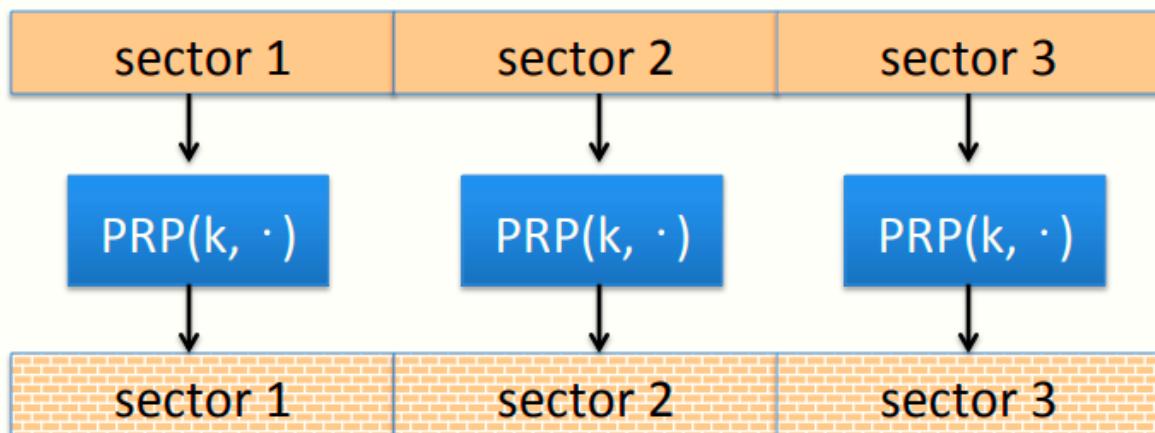
假设需要加密磁盘上的扇区, 每个扇区大小为4KB, 因此密文也必须为4KB, 如果密文比明文长则没有额外的空间存储密文

目标: 实现一个非扩展加密, 即明文与密文大小相同

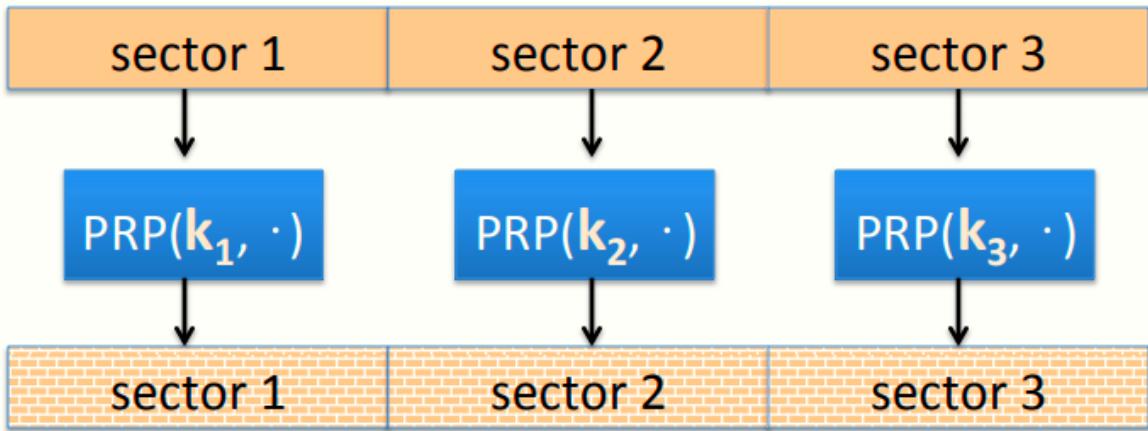
由于加密不能扩展, 意味着消息空间等于密文空间, 显然必须使用确定性加密 (如果加密是随机的, 没有额外的空间存储随机性, 也没有存储完整性的空间, 因为不能扩展密文并加入完整性需要的位) , 因此可达的最多是确定性CPA安全

引理: 如果 (E, D) 为一确定性CPA安全加密, 且有 $M=C$ (明文空间=密文空间) , 则 (E, D) 为一PRP

使用PRP加密, 结构如下



如果使用相同的置换P和同一个密钥k加密所有扇区, 会导致和ECB模式同样的问题, 即相同内容的扇区或包含相同内容的扇区会加密成相同的密文, 磁盘中存在大量的空扇区 (空扇区可能全被置0) , 则导致所有的空扇区都被加密成相同的密文



更好的做法是不同的扇区采用不同的密钥，如上图所示，可以解决上述信息泄露问题，但是仍然存在问题

如果用户期望修改扇区中的某一位，由于使用的是伪随机置换，因此修改后的扇区会生成全新的随机密文，如果用户之后撤销修改并恢复到原始扇区，则密文也会恢复到原来的密文，此时攻击者可以判断用户对该扇区进行了修改而后又恢复了修改，因此还是存在信息泄露

实际上这种信息泄露在不牺牲性能的情况下无法做到，因此将这种泄露视为可接受的

随着存储设备的容量越来越大，采用上述方案的话也会导致密钥很多，密钥管理不方便，有一个解决办法是采用PRF和一个主密钥k，通过PRF生成每个子密钥 $k_t = \text{PRF}(k, t)$ ，其中t为扇区号，从而避免了管理大量的密钥

2、Tweakable block ciphers

上述方案需要用到PRF，能否做到更好？答能，接下来引入了一个可调整的分组密码的概念

目标：期望用一个主密钥 $k \in K$ ，派生出很多的PRP

记 $E, D : K \times T \times X \rightarrow X$ ，其中K为密钥空间，X为消息空间，T为调整空间（Tweak space），然后输出空间为X

Syntax: **$E, D : K \times T \times X \rightarrow X$**

for every $t \in T$ and $k \in K$:

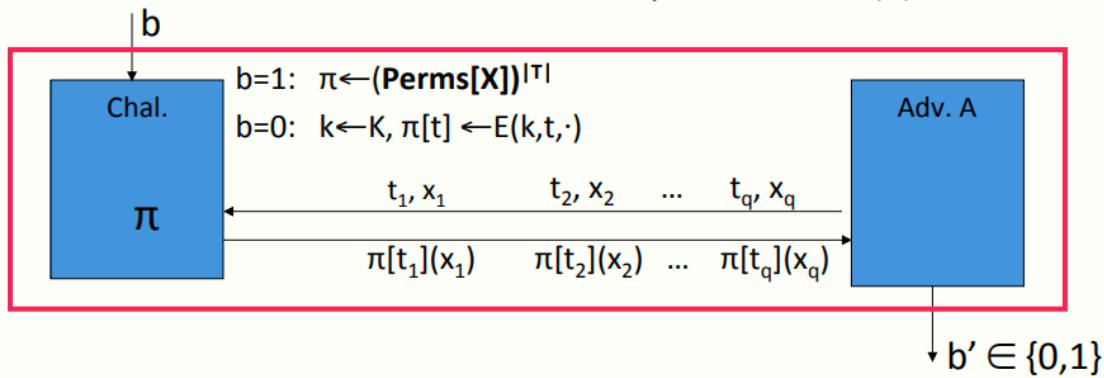
$E(k, t, \cdot)$ is an invertible func. on X, indist. from random

对于每一个微调 $t \in T$ 和给定的密钥 k ， $E(k, t, \cdot)$ 是X上的一个可逆函数，且由于密钥 k 是随机的，因此函数实际上与随机函数没有区别，即对于每个微调，我们可以得到由X到X的独立PRP

在上一节的磁盘加密方案中，采用扇区号作为微调，从而对于每个扇区都有独立的PRP

形式化定义如下

$E, D: K \times T \times X \rightarrow X$. For $b=0,1$ define experiment $\text{EXP}(b)$ as:



- Def: E is a secure tweakable PRP if for all efficient A :

$$\text{Adv}_{\text{tPRP}}[A, E] = |\Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1]| \text{ is negligible.}$$

Dan Boneh

和以前一样，挑战者定义两个实验，具体如下

- 实验1：选择一个真随机置换的集合，该集合是与微调数目一样多的置换（注意到 $|T|$ 写在了指数的位置）
- 实验0：选择一个随机密钥 k ，并根据微调空间里的微调定义置换集合 $\pi[t] \leftarrow E(k, t, \cdot)$

定义：若 E 为一安全的微调PRP，则其对所有高效的攻击者 A ，其上述优势可忽略

上述模型中，攻击者需要做的是提交至多 q 个不同的查询，每个查询包含一个微调 t 和消息 x ，挑战者会返回使用该微调置换后的 x ，并区分是真随机还是伪随机置换，如果做不到则说明可调整的分组密码是安全的

和前几章提到的分组密码不同的是，常规的分组密码需要判断单独的查询是否是伪随机还是真随机置换，即只能和一个置换互动并区分真伪随即置换，而上述可微调的分组密码需要和 $|T|$ 个随机置换互动，并区分真伪

3、Example 1: the trivial construction

来看一个简单的例子，记 (E, D) 为一安全PRP， $E: K \times X \rightarrow X$ ，假设密钥空间等于消息空间，即 $K=X$ ，也即实际上 $E: X \times X \rightarrow X$ ，定义可调整块如下

$$E_{\text{tweak}}(k, t, x) = E(E(k, t), x)$$

首先使用主密钥 k 加密调整值得到随机密钥，然后使用生成的随机密钥加密数据

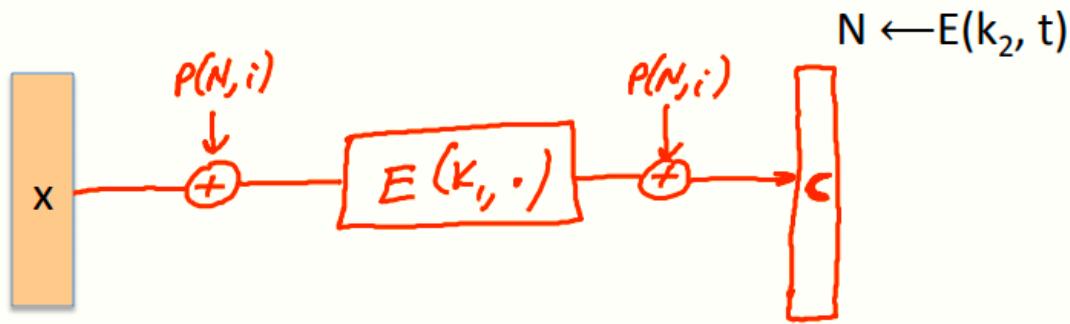
需要注意的是，每一块数据的加密需要使用 E 两次，若有 N 块数据则需要调用 $2N$ 次 E

4、Example 2: the XTS tweakable block cipher

接下来是一个更好的例子XTS（最早基于XEX）

记 (E, D) 为一安全PRP, $E: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ ，其中 E 为一常规分组密码，期望用常规分组密码构造一个可调整分组密码，方案如下

- XTS: $E_{\text{tweak}}((k_1, k_2), (t, i), x) =$



\Rightarrow to encrypt n blocks need $n+1$ evals of $E(.,.)$

XTS接受两个密钥作为输入，调整由两个值组成， t 为微调值， i 为索引值， x 为一 n bits字符串，步骤如下

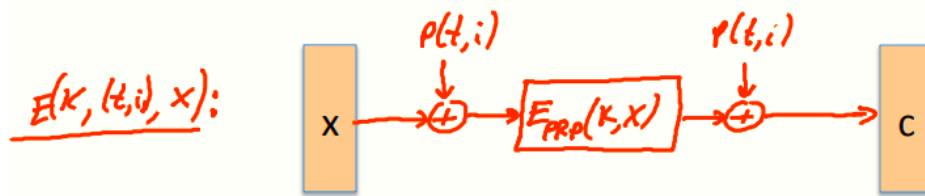
1. 使用密钥 k_2 对 t 加密，结果记为 N
2. 将 N 和索引 i 作为扩展函数 P 的输入，其输出结果与消息 x 进行 xor 计算
3. 将步骤2中的计算结果使用密钥 k_1 加密
4. 将步骤3的结果再与 P 的结果进行一次 xor 计算，最终得到密文

注意到生成 N 的过程调用了一次 E ，而每块消息的加密需要调用一次 E ，因此加密 N 块消息需要调用 $N+1$ 次 E

有个问题：在使用微调前对其进行加密是否是必要的？若有下述方案，则其是否是安全的可调整PRP？

Is it necessary to encrypt the tweak before using it?

That is, is the following a secure tweakable PRP?



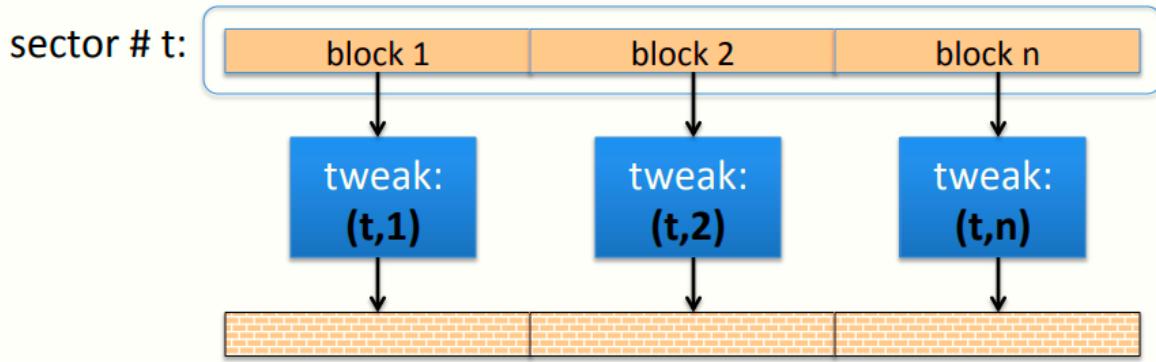
- Yes, it is secure
- No: $E(k, (t, 1), P(t, 2)) \oplus E(k, (t, 2), P(t, 1)) = P(t, 1)$
- No: $E(k, (t, 1), P(t, 1)) \oplus E(k, (t, 2), P(t, 2)) = P(t, 1) \oplus P(t, 2)$
- No: $E(k, (t, 1), P(t, 1)) \oplus E(k, (t, 2), P(t, 2)) = 0$

显然不是，如果不加密的话，相当于 $E(k, (t, i), x)$ ，则扩展函数 P 直接将 t 和 i 作为输入

而此时若输入消息 x 为 $P(t, 1)$ ，则其在第一次 xor 计算后结果为全0，无论加密结果是什么，其最终得到的密文为 $c0 \oplus P(t, 1)$ ，若消息 x 为 $P(t, 2)$ ，同理可得最终密文为 $c0 \oplus P(t, 2)$ ，两个密文异或在一起可以消除 $c0$ 从而得到 $P(t, 1) \oplus P(t, 2)$

5、Disk encryption using XTS

磁盘中使用XTS的例子



- note: block-level PRP, not sector-level PRP.
- Popular in disk encryption products:
Mac OS X-Lion, TrueCrypt, BestCrypt, ...

需要注意的是，上图是针对某一扇区的方案，对该扇区的数据进行分块，每一分块都有自己独立的PRP，因此只是块级（block level）不是扇区级（sector level），但是这个方案实际上在块级提供了确定性CPA加密

最下面一行是常用的磁盘加密方案

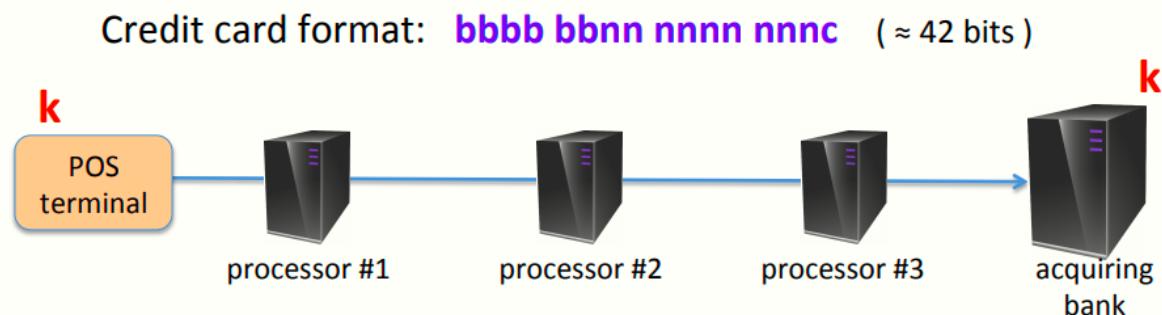
6、Summary

- 如果需要由一个密钥导出多个独立的PRP，可调整的加密是个有用的选择
- XTS比上面那个简化的方案更有效，但是两者都是窄块方案（narrow block），即都是采用16字节的AES
- 上一讲中提到的EME是一种宽块的可微调方案，但是需要调用两次E，所以需要加密的块很多时，其效率为XTS一半

W4 8-5 Format preserving encryption

保留格式的加密，实际中也很常用的方案（比如加密信用卡）

1、Encrypting credit card numbers



外国银行卡号为16位数字（四组四位数，大约是 2^{42} 种可能的卡号）

有些时候，加密信息会经过很多中间节点（比如上图所示的三个）并最终到达银行，每个中间节点都可能获取一些信息

目标：端到端加密，即中间节点期望得到信用卡号，因此对信用卡号的加密要看起来像一个合法的信用卡号（哈？）

2、Format preserving encryption (FPE)

保留格式的加密FPE，更抽象化的说是，对于给定的 $0 < s \leq 2^n$ ，通过安全的PRF $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ ，构造一个在 $\{0, \dots, s-1\}$ 上的PRP（ s 为总数，对信用卡号而言， s 大约是 2^{42} ）

因此我们需要做的是输入一些PRF（如16byte-AES），然后缩小PRF的域使其适合需要加密的数据

有了上述结构，就可以用来加密信用卡号了，步骤如下

1. 将给定的信用卡号（CC#）映射至 $\{0, \dots, s-1\}$
2. 应用PRP来加密这个卡号
3. 将加密结果映射回卡号，使其看起来像个合法的卡号

Then to encrypt a credit card number: ($s = \text{total } \# \text{ credit cards}$)

1. map given CC# to $\{0, \dots, s-1\}$
2. apply PRP to get an output in $\{0, \dots, s-1\}$
3. map output back a to CC#

需要注意的是，这仍然是一个非扩展的加密方案，只是将一个合法的卡号加密为另一个合法的卡号，具体步骤如下

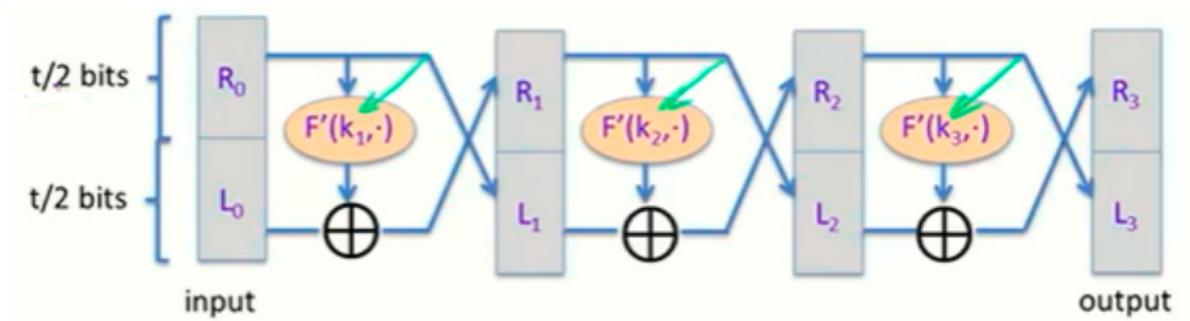
3、Step 1: from $\{0,1\}^n$ to $\{0,1\}^t$

其中 $t < n$ ，即将PRF从 $\{0,1\}^n$ 缩小至 $\{0,1\}^t$ ， t 为2的幂次，需要使得 2^t 最接近S，也就是他妈的找到一个 t ，满足这个不等式

$$2^{t-1} < s \leq 2^t$$

方法：Luby-Rackoff结构，使用一种截断 F' ， F' 具体如下图，对于 $t=42$ 而言，将其截断到21 bits输出

上述方案的一种简单的方法是单独的使用AES，比如我们有一个21 bits输入，可以在后面加好多的0直至128 bits，然后作为AES的输入，得到的输出再截断位21 bits，Luby-Rackoff结构具体如下



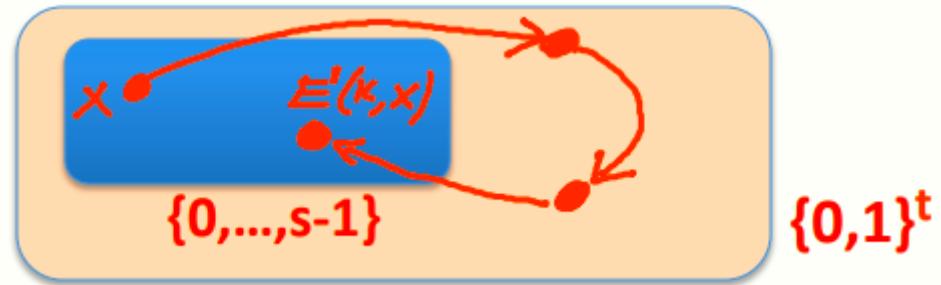
三轮Luby-Rackoff如图所示，显然Luby-Rackoff采用了Feistel结构，然后在其内部采用了这个截断 F' ，并将一个安全的PRF转换为一个安全的PRP

注意到每个 F' 都使用了不同的密钥，实际上低轮次的Luby-Rackoff并不像其期望的那样好使，更好的办法是用七轮，同时意味着需要七个不同的密钥

4、Step 2: from $\{0,1\}^t$ to $\{0, \dots, s-1\}$

第一步中，得到了一个PRP $(E, D): K \times \{0,1\}^t \rightarrow \{0,1\}^t$

记 $E'(k, x)$ ，其中 $x \in \{0, \dots, s-1\}$ ，令 $y=x$ ，然后进行如下迭代：不断迭代计算 $y=E(k, y)$ 直到 $y \in \{0, \dots, s-1\}$ ，然后输出 y



迭代过程有点类似这个简化图，蓝色区域代表 $\{0, \dots, s-1\}$ ，浅黄色区域代表 $\{0,1\}^t$ ，然后不断使用E迭代计算直到他又回到蓝色区域，解密的话就这个红线反着走就是了（其实不断迭代不太准确，由于s是在那个范围的，所以就期望来说只需要两次迭代就可以了）

5、Security

构造讲完了，看看安全性

Step 2 is tight: $\forall A \exists B: \text{PRP}_{\text{adv}}[A, E] = \text{PRP}_{\text{adv}}[B, E']$

Intuition: \forall sets $Y \subseteq X$, applying the transformation to a random perm. $\pi: X \rightarrow X$
gives a random perm. $\pi': Y \rightarrow Y$

Step 1: same security as Luby-Rackoff construction

(actually using analysis of Patarin, Crypto'03)

note: no integrity

需要注意的是，Luby-Rackoff并不提供安全性

6、延伸阅读

- Cryptographic Extraction and Key Derivation: The HKDF Scheme.
H. Krawczyk, Crypto 2010
- Deterministic Authenticated-Encryption:
A Provable-Security Treatment of the Keywrap Problem.
P. Rogaway, T. Shrimpton, Eurocrypt 2006
- A Parallelizable Enciphering Mode. S. Halevi, P. Rogaway, CT-RSA 2004
- Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. P. Rogaway, Asiacrypt 2004
- How to Encipher Messages on a Small Domain:
Deterministic Encryption and the Thorp Shuffle.
B. Morris, P. Rogaway, T. Stegers, Crypto 2009

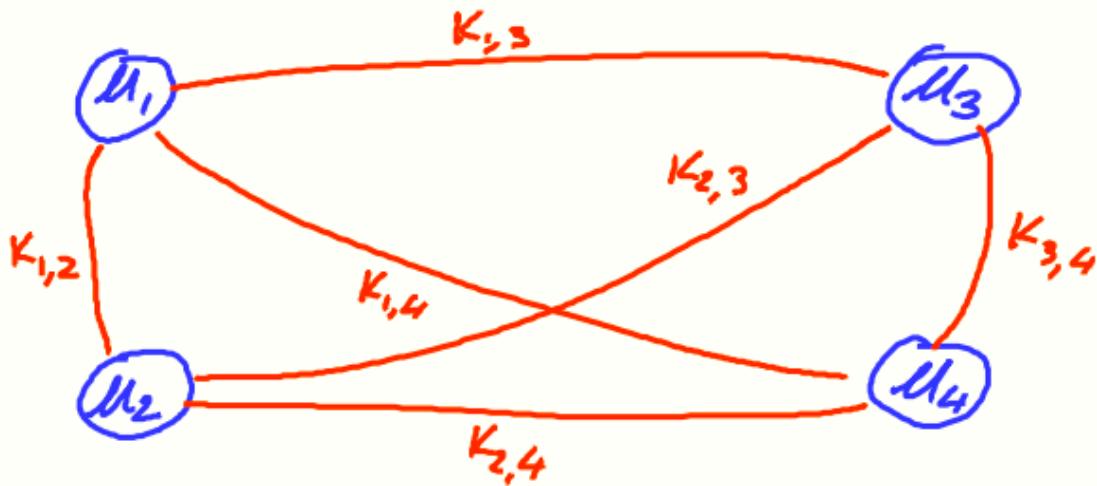
W4 知识梳理

W4 Problem Set & Programming Assignment

W5 9-1 Trusted 3rd parties

1、Key management

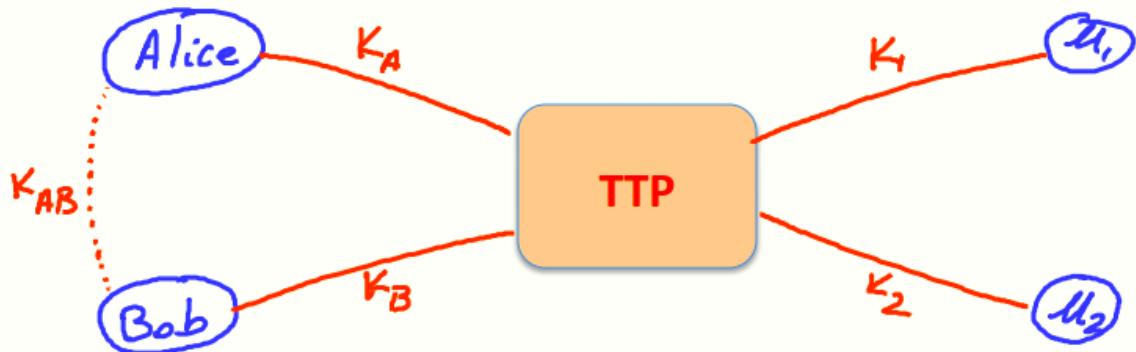
先看一个问题：假设有N个用户，这些用户如何管理用于相互之间通信的密钥？



最简单的做法是，每个用户之间都保存一个通信密钥，对于每个用户而言，需要保存的密钥数量为 $O(N)$ ，是否有更好的解决办法？

2、A better solution

可以由更好的方案解决上述问题，其中一个为在线的可信第三方（Online Trusted 3rd Parties, TTP），模型如下

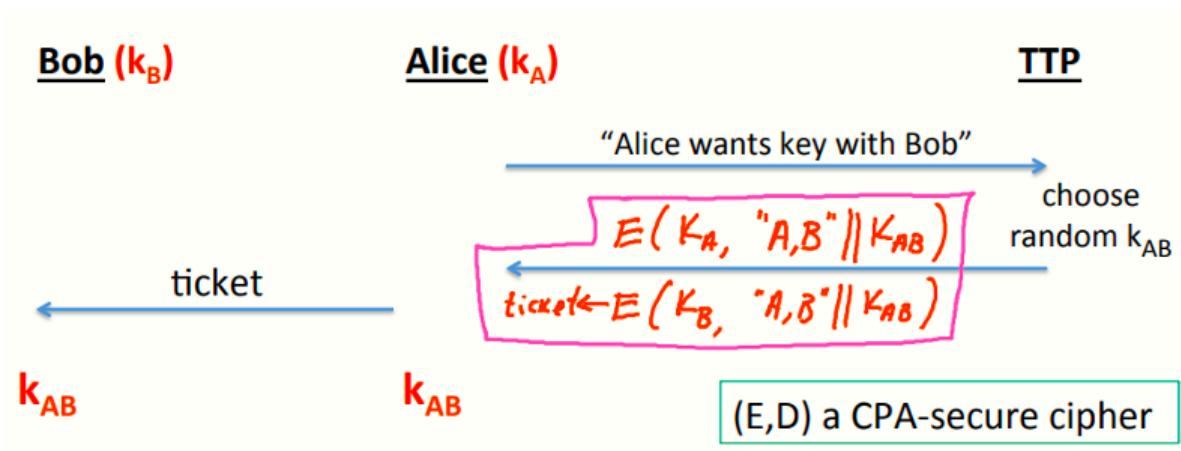


对于每个用户而言，需要将自己的密钥共享给TTP，每个用户都只需要记住自己的密钥就可以了

Alice和Bob如何通信？两者需要运行某种协议，使得在协议结束时双方都能获得共享密钥，且该共享密钥是攻击者不知道的

3、Generating keys: a toy protocol

首先，Alice和Bob都将自己的密钥 K_A 和 K_B 共享给了TTP，现在Alice想要和Bob协商一个共享密钥，模型如下



1. Alice先向TTP发消息，告知期望与Bob通信
2. TTP生成一个随机共享密钥 K_{AB}
3. TTP向Alice发送一条消息，包含以下两部分（加密系统E采用CPA安全加密方案）
 - 用 K_A 加密的 K_{AB} （图中"A,B"表示密钥 K_{AB} 用于Alice和Bob的通信）
 - 一个票据（ticket），ticket使用 K_B 加密，内容与上一个一样
4. Alice收到后用 K_A 解密得到 K_{AB} ，若其想要和Bob通信时，将ticket发送给Bob，Bob解密得到 K_{AB} 后即可开始通信

需要指出的是，上述模型仅在窃听下安全，对于篡改或者主动攻击并不安全，由于加解密方案为CPA安全的，因此对于窃听者来说无法获得对 K_{AB} 的信息

另外还需要注意的是，共享密钥是在TTP生成的，因此TTP下线或不可访问时，Alice和Bob就不能完成密钥交换，换句话说，TTP掌握了所有人的密钥，若TTP被攻击了，则攻击者可以轻松获取系统中曾经交换过的密钥

- 优点：快速高效，且仅仅使用了对称密码体制
- 缺点：TTP必须可信，谁来当这个可信第三方同样是个问题

Kerberos系统大概是这个机制（不完全是，思路和上面的这个方案类似而已）

4、Toy protocol: insecure against active attacks

上述方案很容易遭受重放攻击，下面讲一个例子

假设用户Alice和卖家Bob，Alice想在Bob那里买电子书并完成了交易，Bob收到了书钱并发送给Alice这本书的一份拷贝

对于上述场景，攻击者可以完全记录这次对话并简单地向Bob重演这个对话，此时Bob会认为Alice重新购买了一份这本书的拷贝，Bob会再向Alice收一次钱并发送一次书的拷贝

5、Key question

问题来了：能否设计一个密钥交换协议，使其在窃听和主动攻击下都能确保安全，能否设计一个安全的密钥交换协议而不需要任何在线可信第三方组织

显然可以，三种方案如下

- 1974年还在读大学的Merkle提出的方案
- 1976年Diffie-Hellman的密钥交换协议
- 1977年公钥算法RSA

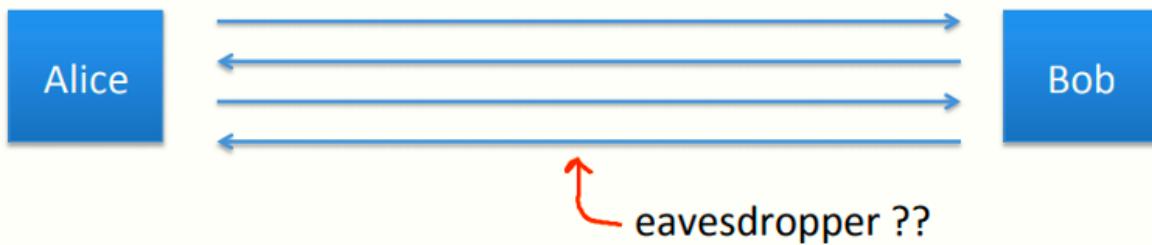
近十年来又出现的新的方案：

- ID-based enc. (BF 2001)，基于身份的加密，管理公钥的另一种方式

- Functional enc. (BSW 2011), 函数式加密，通过给出一个只能解密部分给定密文的密钥的方式完成

W5 9-2 Merkle Puzzles

1、Key exchange without an online TTP?



如图所示，假设Alice和Bob素未谋面，但是出于某种原因他们需要一个共享密钥，该模型下没有TTP，需要某种协议来互相发送消息并确定一个互相都知道的共享密钥

提问：对于上述模型，能否用对称加密机制实现没有可信第三方的密钥交换（可以使用加密，hash function或者其他目前已经学过的）？答可以

需要注意的是，目前仍然只讨论窃听安全，不涉及篡改等主动攻击（eavesdropping only, no tampering）

2、Merkle Puzzles (1974)

该协议由瑞夫·墨克(Ralph Merkle)在1974年发明（当时还在读本科，某次研讨会上发明的这个协议），遗憾的是，这个协议因为效率问题并没有使用

协议主要工具：puzzle，即一个非常难解决的问题，需要非常用心努力的解决

举个例子，比如说128 bits的AES密钥P，前96 bits均为0，后32 bits为随机生成，并用该密钥加密特定的消息

可以看出，密钥最多有 2^{32} 种可能，对于每一种可能都用其解密并看看是否得到明文，如果得到了就还原了密钥P（即解决了puzzle）

回到协议，看看协议的工作流程：

1. Alice首先准备 2^{32} 个puzzles，对 $i=1, \dots, 2^{32}$ ，选择随机的 $P_i \in \{0,1\}^{32}$ ，再随机选择128 bits的 $x_i, k_i \in \{0,1\}^{128}$ ，令 $puzzle_i = E(0^{96} \parallel P_i, "Puzzle \# x_i" \parallel k_i)$ ，并将这么多个puzzles发给Bob
2. Bob收到这么多puzzles后随便挑一个（其他的不要），假设选到了第j个puzzle并尝试解决，期望时间为 $O(2^{32})$ ，解决puzzle后得到 (x_j, k_j) ，将 x_j 发回给Alice
3. Alice根据Bob发来的 x_j ，找到对应的puzzle，然后共享密钥就是 k_j

Alice: prepare 2^{32} puzzles

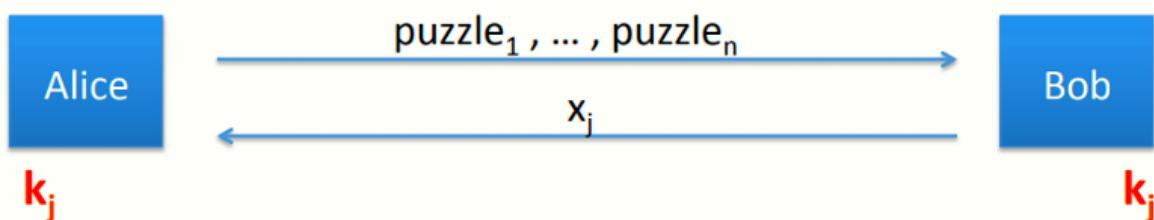
- For $i=1, \dots, 2^{32}$ choose random $P_i \in \{0,1\}^{32}$ and $x_i, k_i \in \{0,1\}^{128}$
set $\text{puzzle}_i \leftarrow E(0^{96} \parallel P_i, \text{"Puzzle \# } x_i \text{"} \parallel k_i)$
- Send $\text{puzzle}_1, \dots, \text{puzzle}_{2^{32}}$ to Bob

Bob: choose a random puzzle_j and solve it. Obtain (x_j, k_j) .

- Send x_j to Alice

Alice: lookup puzzle with number x_j . Use k_j as shared secret

换一个图，更直观的说一下这个协议流程



Alice的计算量: $O(n)$, 即准备n个puzzles

Bob的计算量: $O(n)$, 即解决其挑选的puzzle

窃听者攻破协议需要的计算量: $O(n^2)$, 先得确定Bob选了哪一个puzzle, 然后解决puzzle, 因此计算量为平方阶

但是协议存在很大的问题:

- 通信双方的计算量很大, Alice发送这么多puzzles (大约16G或者更多), Bob解决一个puzzle也需要花一点时间
- 窃听者只需要在 $O(2^{64})$ 内即可破解, 实际上并不安全, 使其更安全的方法为增大n, 比如增大到 2^{64} , 此时窃听者的破解期望为 $O(2^{128})$, 安全了, 但是对于Alice和Bob来说这么大的计算量过于奢侈, 也不现实

总结一下: 尽管说非常不现实, 但仍然是个不错的思路, 该协议构造了一个平方差距(quadratic gap), 即通信方只需要线性阶的计算量, 而实施攻击需要平方阶

新的问题: 能否在仅使用对称加密的情况下构造比平方差距更好的安全性? 不晓得

有一个值得思考的点是: 平方差距是否是我们能够得到的最好的结果? 不晓得, 但是可以参考AES和SHA-256来思考密钥交换的设计思路

某些失败的例子表明, 如果将块加密或者哈希函数作为一个黑盒, 那么平方差距已经是最好的结果了

BM'09这篇论文中说平方差距是我们能够实现的最好结果 (值得一看的论文)

W5 9-3 The Diffie-Hellman protocol

1、复习与预习

上节课讲到了无需TPP的密钥交换, 需要注意的是Alice和Bob素未谋面, 但是为了完成安全的通信, 因此需要交换一个共享密钥

上节课的内容主要围绕对称密码或者散列函数展开，并介绍了一个有用的协议Merkle Puzzles，可以在通信双方和窃听者之间产生平方差距，但是仍然是不安全的，且因为效率问题并不实用

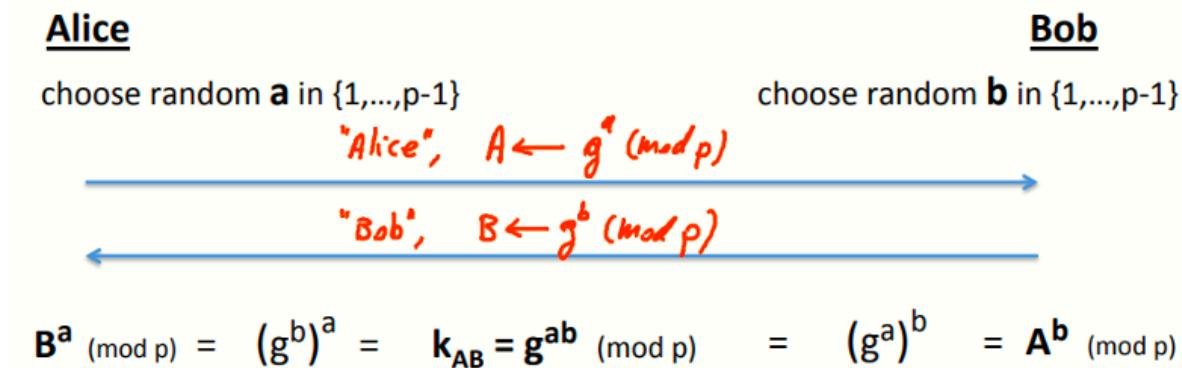
因此本节课的内容，希望实现一种方案，使得通信双方与窃听者之间产生某个指数级的差距

需要注意的是，本节内容仍然围绕无TPP的密钥交换机制展开，还需要注意的是，目前仅针对窃听安全，即攻击者实际上只是一个窃听者，并不以任何方式修改数据包或向网络中注入数据等主动攻击

2、The Diffie-Hellman protocol (informally)

背景：1976年由斯坦福大学教授Martin Hellman和他的研究生Wig Diffie提出的协议

方案流程：事先准备一个大素数 p （很大，600位以上的十进制数，或者2048 bits），然后准备一个整数 $g \in \{1, \dots, p\}$ ， p 和 g 都可以通过不安全的信道传送（即可以被窃听），然后协议如下



对于Alice，他随机选择 $a \in \{1, \dots, p-1\}$ ，计算 $A \equiv g^a \pmod{p}$ ，并将A发送给Bob

类似的，Bob随机选择 $b \in \{1, \dots, p-1\}$ ，计算 $B \equiv g^b \pmod{p}$ ，并将B发送给Alice

好了，Alice和Bob此时可以用对方的发来的数再计算便可以得到共享密钥，对于Alice而言，计算 $B^a \pmod{p}$ ，Bob计算 $A^b \pmod{p}$ ，尽管计算的值不同，但是双方的计算结果是相同的，均为 $K_{AB} \equiv g^{ab} \pmod{p}$ ，从而Alice和Bob共享了一个密钥

3、Security

注意到D-H密钥交换基于指数的运算性质，且本身的计算也是模幂运算

更关键的问题在于，安全吗？为什么一个窃听者就算知道了Alice和Bob自己计算的A和B也不能找出相同的共享密钥呢？

窃听者肯定可以知道模数 p 和底数 g ，同时可以听到双方传送的A和B，问题在于窃听者能否根据这四个值计算出共享密钥，即计算 $K_{AB} \equiv g^{ab} \pmod{p}$

引入一个辅助定义，定义一个Diffie-Hellman函数，基于某个值 g 定义的函数，具体如下

$$DH_g(g^a, g^b) = g^{ab} \pmod{p}$$

问题来了：计算这个DH-g函数到底有多难？需要注意 p 是600位甚至更长的十进制数

假设一个素数 p 有 n bits这么长，使用目前已知最好的算法（General Number Field Sieve, GNFS，一般数域筛法），期望的运行时间在指数的立方根的幂左右，即 $e^{O(\sqrt[3]{n})}$ ，不是线性阶的也不是指数阶的，算是个次指数算法

<u>cipher key size</u>	<u>modulus size</u>	<u>Elliptic Curve size</u>
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
256 bits (AES)	15360 bits	512 bits

As a result: slow transition away from (mod p) to elliptic curves

上图展示了破解Diffie-Hellman协议的难度与破解具有适当位数的密码的难度的比较，为了达到与分组密码相当的安全性，D-H交换的密钥长度比分组密码要长的多得多

更长的密钥意味着更慢的效率，有没有办法可以解决效率问题？可以把D-H协议从一个素数的算术模型转换成一个不同类型的代数对象，即在另一个代数对象上解决D-H问题要更困难

另一个代数对象为椭圆曲线，在这些椭圆曲线上计算DH函数比计算DH模素数要困难得多，由于这个问题非常困难，因此可以使用更小的对象，也就是更短的密钥，如上图所示，椭圆曲线的密钥长度也就比分组密码的密钥长一倍而已

www.google.com

The identity of this website has been verified by Thawte SGC CA.

[Certificate Information](#)

Your connection to www.google.com is encrypted with 128-bit encryption.

The connection uses TLS 1.0.

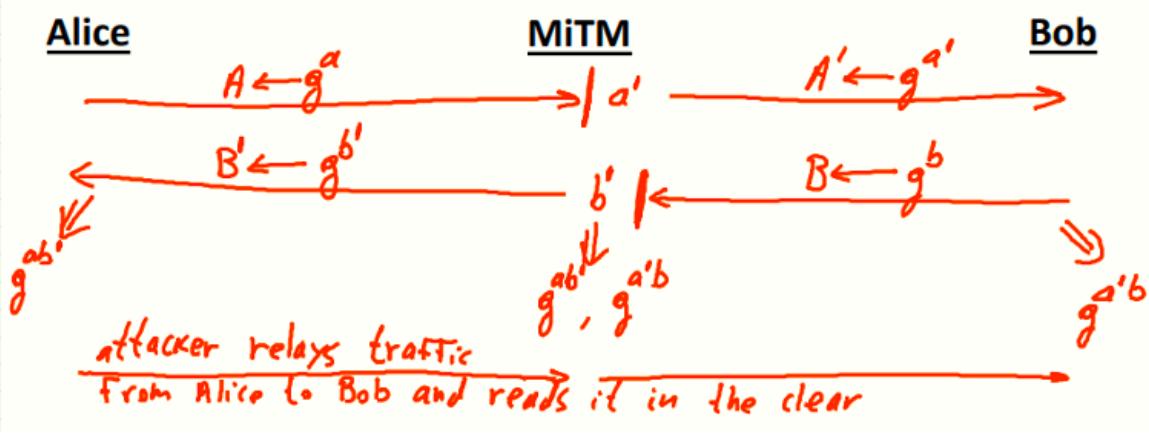
The connection is encrypted using RC4_128, with SHA1 for message authentication and **ECDHE_RSA** as the key exchange mechanism.

Elliptic curve Diffie-Hellman

有些网站上写的ECDHE就是基于椭圆曲线的DH密钥交换协议

4、Insecure against man-in-the-middle

这两节的内容都是在仅窃听安全下讨论的，事实上该协议不能抗主动攻击，比如对于中间人攻击（MITM）就是不安全的，看下面这个模型

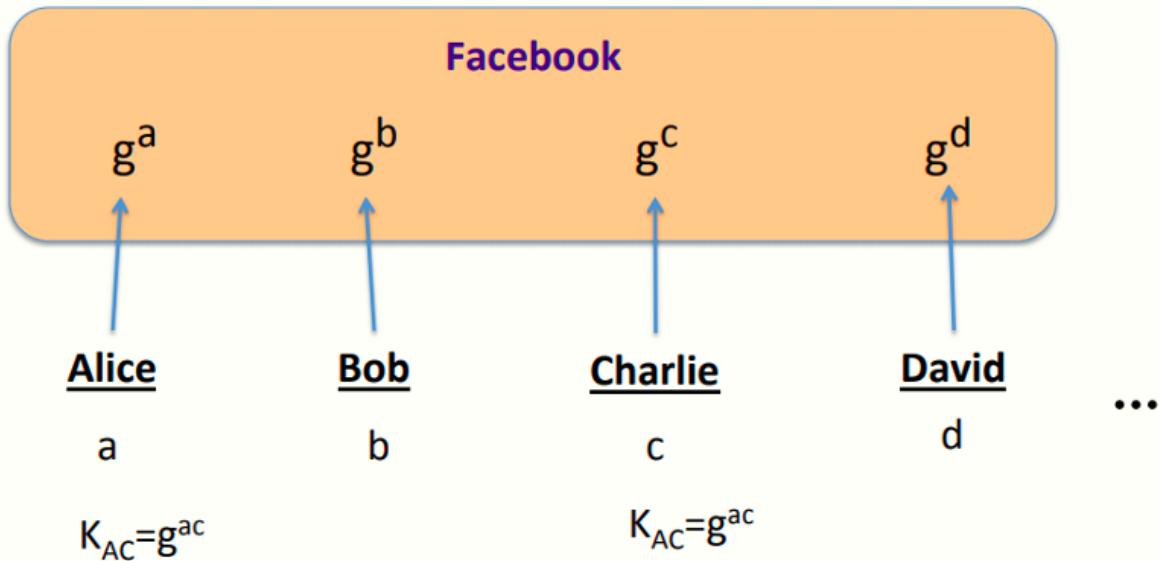


正常来说，Alice与Bob会准备好A和B并发送给对方，而此时中间人可以拦截Alice的A，并用事先准备好的 a' 计算得到新的 A' 并发送给Bob，同样的截获Bob的B并发送 B' 给Alice，本应当是Alice与Bob交换密钥，实际上是Alice和Bob分别与MITM交换了密钥

可以看到，MITM截获了A和B，并以自己的 A' 和 B' 与Alice和Bob交换密钥，因此他有了两个共享密钥，当Alice向Bob发送消息时，先用Alice的共享密钥解密，然后再用Bob的共享密钥加密后发送给Bob，这意味着MITM不仅可以知道通信内容，由于他可以解密，因此可以篡改内容

5、Another look at DH

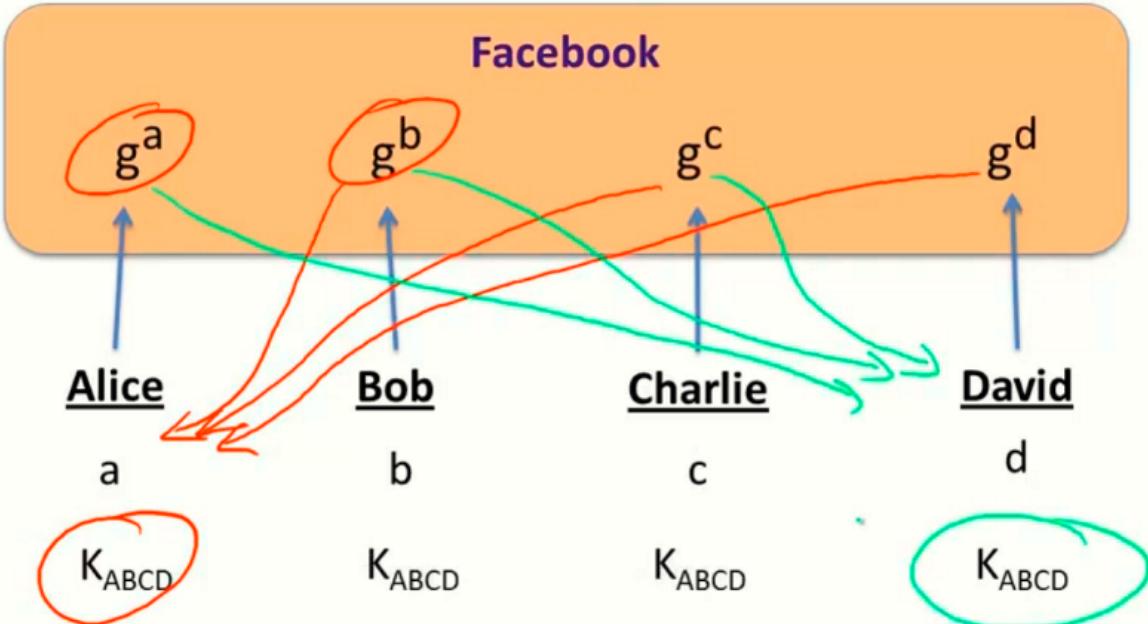
DH协议的一个有趣的性质：可以看作是一个非交互协议



如图，假设有很多用户，每个人都会选择自己的值 a, b, c, d, \dots 并像之前一样计算对应的模数 A, B, C, D, \dots ，然后上传到自己的个人资料上

假设现在Alice想和Charlie通信，不需要像原来那样交换计算的值，只需要Alice和Charlie相互查看对方的资料，然后再根据自己的模数计算出共享密钥就可以通信了，省去了中间的密钥协商的过程

有一个值得思考的问题：假设还是这几个人，对于Alice而言，他能否在查看Bob, Charlie, David的资料后，即可建立属于这四个人的共享密钥，同理Bob在查看……后就可以……？也就是能否做到下面这个图所描绘的方案？



更一般的说法，对于N个通信方，能否非交互的来协商这N个通信方之间的共享密钥？

注意到N=2时，本质上就是D-H协议，对于N=3的情况，有一个已知的协议为Joux，但对于N=4或者更多的通信方，这仍然是一个开放的问题，搞定了说不定就举世闻名图灵奖各大高校名誉教授之类的

W5 9-4 Public-key encryption

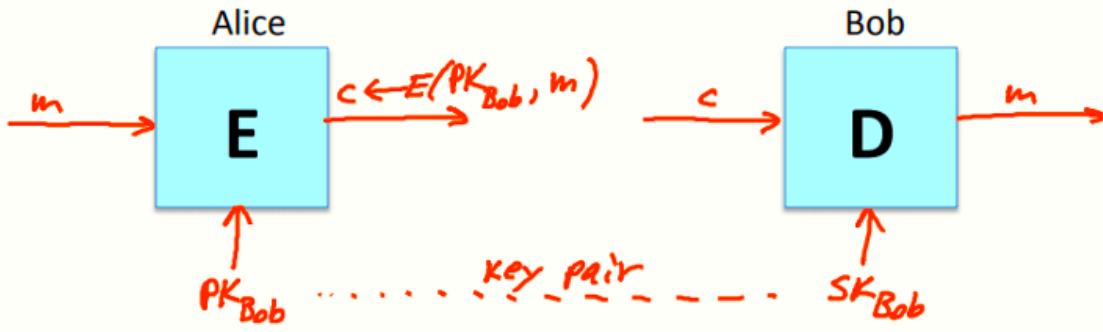
1、复习

Alice和Bob素未谋面，需要建立共享密钥，仅讨论窃听安全

前两节课中介绍了一个基于一般分组密码的方案，可以做到平方差距，但是效率很低的，D-H协议可以达成指数差距，而且实际上应用非常广泛

本节讲另一个公钥加密方案

2、Public key encryption



PK : public key , SK : secret key

和对称加密一样，有加密算法E和解密算法D，通常加密算法以公钥（Public Key, PK）作为密钥输入，而解密算法采用一个不同的密钥即私钥（Secret Key, SK），公钥和私钥通常也称为密钥对

可以和对称密码一样，输入消息和公钥，由加密算法产生密文，或者输入密文和私钥，由解密算法恢复明文

定义：公钥加密系统为一个算法三元组 (G, E, D) ，其中

$G()$ ：随机化算法，用于产生密钥对

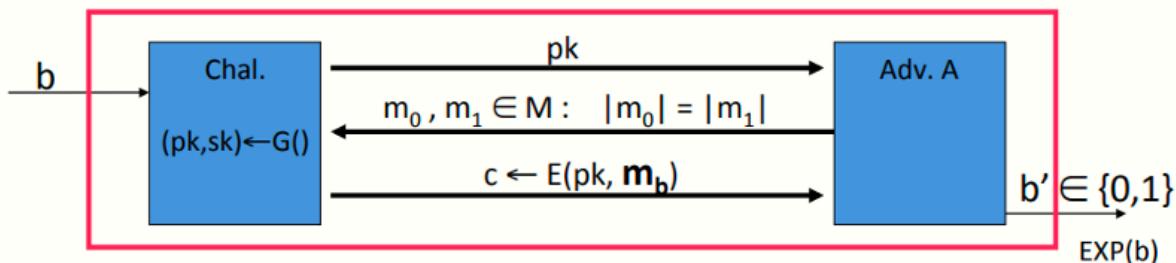
$E(pk, m)$ ：加密算法，输入明文和密钥，输出密文

$D(sk, c)$ ：解密算法，输入密文和密钥，输出明文，有错误的情况下会输出bottom元素

公钥加密系统具有一致性，即对于任意由 G 输出的密钥对 (pk, sk) ，公钥加密后的消息再用私钥解密会得到原来的消息

3、Semantic Security

接下来看语义安全的形式化定义，和以前一样，定义两个实验0和1让攻击者判断



Def: $E = (G, E, D)$ is sem. secure (a.k.a IND-CPA) if for all efficient A :

$$\text{Adv}_{ss}[A, E] = |\Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1]| < \text{negligible}$$

首先，挑战者先运行密钥生成算法 G 来生成密钥对，然后把公钥给攻击者，私钥保密

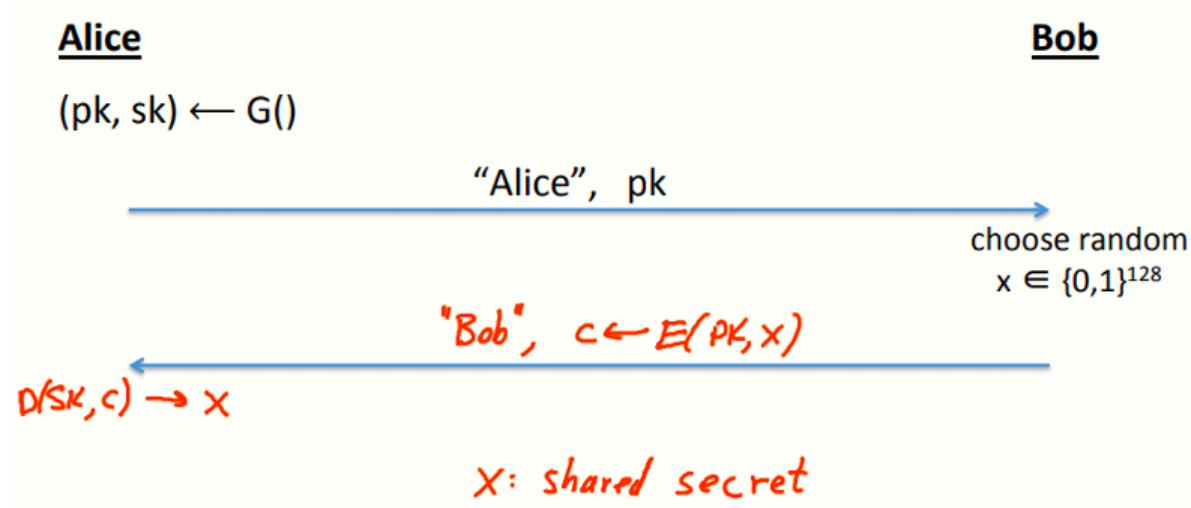
然后攻击者和以前一样，构造两个等长的消息并发送给挑战者，挑战者选择其中之一加密后返回给攻击者

需要注意的是，公钥系统的语义安全中没有必要允许攻击者做选择明文攻击，因为攻击者已经获得了公钥，可以用公钥加密任何期望的消息，而不像前几章那样发起CPA让挑战者帮助他创建他选择的消息的加密

定义：若公钥加密系统 $E = (G, E, D)$ ，则其对于任意高效的攻击者，其上述优势为可忽略的

4、Establishing a shared secret

有了公钥加密系统，来看看如何建立共享密钥，如图所示



首先Alice先用G生成密钥对，然后把pk发给Bob，然后Bob生成一个随机的128 bits的值x，并用pk加密后发回给Alice，Alice收到这个密文后用sk解密后就可以得到x，此时可以用这个x作为共享密钥来通信

注意到这个方案是有顺序的，Bob在收到Alice的信息之前不能发出消息，即Bob需要得到Alice的公钥后才能用其进行加密x，但D-H协议没有这种先后顺序，只要是期望通信的双方，谁先发送都可以

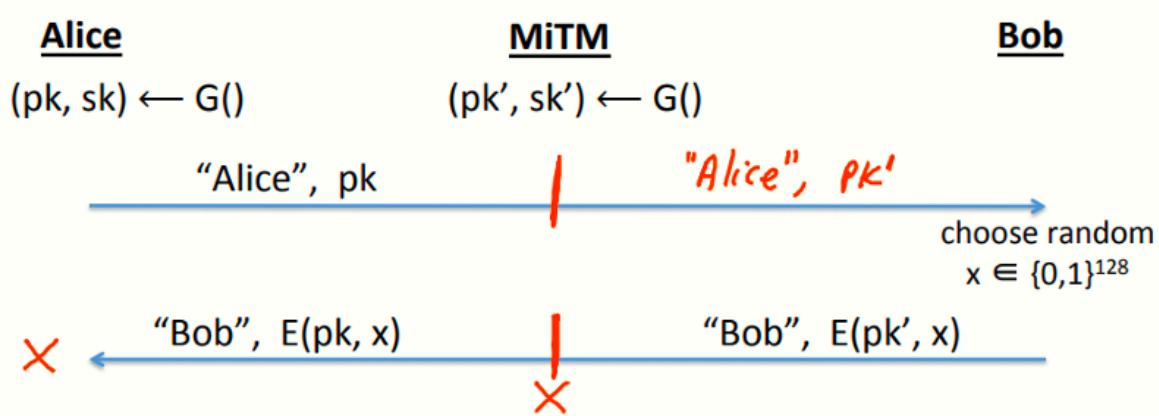
5、Security (eavesdropping)

接下来看安全性，仍然只讨论窃听安全

攻击者可以窃听到公钥和用该公钥加密的x的密文，其目的是想知道x的值，由于公钥系统是语义安全的，因此攻击者窃听到x的密文后，不能断定他的明文是x还是消息空间M中的其他值（除非他尝试所有可能的消息）

6、Insecure against man in the middle

但是协议仍然不能防止MITM攻击，攻击模型如下



首先Alice生成了密钥对(pk, sk)，同时MITM也生成了自己的密钥对(pk', sk')

当Alice发送公钥给Bob的时候，MITM会拦截这条消息，取而代之的是MITM用自己生成的密钥 pk' 给Bob发送消息，然后Bob返回x的密文时，先用自己的私钥 sk' 解密，再用Alice的公钥 pk 加密后返回给Alice

MITM经过上述模型后，不仅Alice和Bob共享了密钥x，MITM也知道了x，因此协议不安全了

7、Public key encryption: constructions

公钥密码的构造通常基于数论和代数中的数学难题（如D-H协议基于代数知识）

8、Further readings

- Merkle Puzzles are Optimal, B. Barak, M.Mahmoody-Ghidary, Crypto '09: 证明了在基于对称加密算法和Hash函数的方案下，Merkle puzzle是密钥交换的最优解
- On formal models of key exchange (sections 7-9) V. Shoup, 1999: 总结了一些密钥交换机制，包括基于公钥密码学的，基于D-H协议的

W5 10-1 Notation

第十章主要讲一些数论知识，可能有点复杂，但是将要讨论基于数论的密钥交换协议

1、Background

我们可以用数论知识构建很多方案，如密钥交换协议、数字签名、公钥加密系统等等

2、Notation

先解释一些符号和记法

大写字母N：表示一个正整数

小写字母p：表示一个正素数

\mathbb{Z}_N ：表示0~N-1组成的集合，可以对该集合内的元素做模N的加法和乘法

举个例子：模运算，若记N=12，则

$$9 + 8 = 5 \quad \text{in } \mathbb{Z}_{12}$$

$$5 \times 7 = 11 \quad \text{in } \mathbb{Z}_{12}$$

$$5 - 7 = 10 \quad \text{in } \mathbb{Z}_{12}$$

注意加法和乘法都需要模N（本例中N=12）

部分加法与乘法的定律在 \mathbb{Z}_N 中仍然有效，如分配律， $x \cdot (y+z) = x \cdot y + x \cdot z$

3、Greatest common divisor

最大公约数GCD

定义：对于整数x, y，记 $\gcd(x, y)$ 为二者的最大公约数

最大公约数的一些性质：

- 对于任给的整数x, y，总是存在另两个整数a, b，满足 $a \cdot x + b \cdot y = \gcd(x, y)$ ，即 $\gcd(x, y)$ 可以看作x和y的某种线性组合，a和b可以通过扩展欧式算法找到
- 若 $\gcd(x, y) = 1$ ，则称x和y互素

4、Modular inversion

对于有理数而言，一个非零有理数有自己的倒数，如2的倒数为1/2，对于集合 \mathbb{Z}_N 而言，引入模逆概念

定义： \mathbb{Z}_N 集合内的元素x的逆为该集合上的另一个元素y，且满足 $x \cdot y = 1$ ，记y为 x^{-1}

对于 \mathbb{Z}_N 集合内的奇数a而言，其逆为 $(a+1)/2$

$$\begin{aligned} \mathbb{Z}_N^* &= (\text{set of invertible elements in } \mathbb{Z}_N) = \\ &= \{ x \in \mathbb{Z}_N : \gcd(x, N) = 1 \} \end{aligned}$$

这个 \mathbb{Z}_N^* ，表示 \mathbb{Z}_N 所有可逆元素的集合，即所有 $x \in \mathbb{Z}_N$, $\gcd(x, N) = 1$

5、Solving modular linear equations

Solve: $\mathbf{a} \cdot \mathbf{x} + \mathbf{b} = \mathbf{0}$ in \mathbb{Z}_N

Solution: $\mathbf{x} = -\mathbf{b} \cdot \mathbf{a}^{-1}$ in \mathbb{Z}_N

比如解决这种线性同余方程，本质就是找到a的逆，可以用扩展欧几里得找到，时间复杂度为O(n^2)

W5 10-2 Fermat and Euler

1、Fermat's theorem

费马小定理：若p为素数，则任给 $x \in \mathbb{Z}_p^*$ ，有

$$x^{p-1} \equiv 1 \pmod{p}$$

对于 $x \in \mathbb{Z}_p^*$, $x \cdot x^{p-2} \equiv 1$, 即 $x^{-1} \equiv x^{(p-2)}$, 这也是一个找到x的逆元的方法，但是效率比欧几里得更慢，大约是对数立方阶 $O(\log^3 p)$

2、Application: generating random primes

假设我们需要生成一个大素数p（如p有1024 bits），一个简单的步骤可以生成大素数

1. 选择一个随机整数 $p \in [2^{1024}, 2^{1025}-1]$
2. 计算 $2^{(p-1)} \equiv 1$, 如果等于则输出p, 不等于则返回步骤1

算法得到不是素数的概率很低，对于1024 bits的p而言，不是素数的概率约为 2^{-60} ，因此并不是一个好的算法，因为可能生成伪素数

3、The structure of \mathbb{Z}_p^*

欧拉定理： \mathbb{Z}_p^* 为循环群， $\exists g \in (\mathbb{Z}_p)$, 有 $\{1, g, g^2, g^3, \dots, g^{(p-2)}\} = \mathbb{Z}_p$, 且g为 \mathbb{Z}_p^* 生成元

注意到幂数只到p-2次幂，根据费马小定理，g的p-1次幂实际上是等于1，p次幂等于g，因此得到一个循环

需要注意是，定理中为存在 $g \in \mathbb{Z}_p^*$, 而非任意g, 即不是所有元素都是生成元

对于由生成元g生成的集合，称为g的生成群，记为

定义：记 $\text{ord}_p(g)$ 为g的阶为的大小，即 $\text{ord}_p(g) = |\{g^a \mid a \in \mathbb{Z}\}|$, 同时也是使得 $g^a \equiv 1$ 成立的最小的a

拉格朗日定理：对于 $\forall g \in \mathbb{Z}_p^*$, $\text{ord}_p(g)$ 可以整除 $p-1$

4、Euler's generalization of Fermat

定义：对于整数N而言，定义欧拉 ϕ 函数，即 $\phi(N) = |\mathbb{Z}_N^*|$

欧拉 ϕ 函数也就是1~N-1中与N互素的数的个数，对于N是素数而言， $\phi(N) = N-1$ ，对于RSA中需要用到的大整数 $N = p \cdot q$ 而言， $\phi(N) = (p-1)(q-1)$

欧拉定理： $\forall x \in \mathbb{Z}_N^*, x^{\phi(N)} \equiv 1$

欧拉定理是费马小定理的一个推广，同时也是RSA密码系统的基础

W5 10-3 Modular e'th roots

1、Modular e'th roots

上节课提到了怎么解线性同余方程，可以通过求逆元的方式解决

问题在于，有没有办法解决高次同余方程，如 $x^2 - c \equiv 0 \pmod{p}$ 或者 $x^{37} - c \equiv 0 \pmod{p}$ 之类的

假设 p 为一素数， $c \in \mathbb{Z}_p$

定义： $x \in \mathbb{Z}_p$ ，且满足 $x^e = c$ 在 \mathbb{Z}_p 内，称 x 为 c 模 p 的 e 次方根

比如，若 $p=11$ ，则7的立方根为6 ($6^3 = 216 \equiv 7 \pmod{11}$)，3的立方根为5 ($5^3 = 125 \equiv 3 \pmod{11}$)，1的立方根为1，2在模11的情况下没有平方根

2、The easy case

c 模 p 的 e 次方根何时存在？存在的情况下有没有高效的计算方法？

假设需要计算某数 c 的 e 次根，且 $\gcd(e, p-1) = 1$ ，则对于 \mathbb{Z}_p^* 内的所有的 c ， c 的 e 次根在 \mathbb{Z}_p 内，且有一种比较快速的方法找到这个根，方法如下

Proof: let $d = e^{-1}$ in \mathbb{Z}_{p-1} . Then $c^{1/e} = c^d \in \mathbb{Z}_p$

$$\begin{aligned} d \cdot e = 1 \text{ in } \mathbb{Z}_{p-1} &\Rightarrow \exists k \in \mathbb{Z}: d \cdot e = k \cdot (p-1) + 1 \Rightarrow \\ &\Rightarrow (c^d)^e = c^{d \cdot e} = c^{k \cdot (p-1) + 1} = [c^{p-1}]^k \cdot c = c \in \mathbb{Z}_p \end{aligned}$$

Dan Boneh

3、The case $e=2$: square roots

另一个问题是， $e=2$ 且 p 是奇素数的情况，此时 e 和 $p-1$ 并不互素

问题在于，由 x 到 x^2 的映射实际上是2对1函数 (2-to-1 function)， x 和 $-x$ 都能映射到同一个 x^2 ，比如在 $p=11$ 的情况下，会有如下所示

Example: in \mathbb{Z}_{11}^* :	1	10	2	9	3	8	4	7	5	6
	1	10	2	9	3	8	4	7	5	6

其中10和-1在 $p=11$ 下同余，因此二者的平方都为1，其他数同理，因此引出二次剩余概念

定义：若 x 在 \mathbb{Z}_p 中存在平凡根，则称其为二次剩余 (Quadratic Residue, Q.R.)，否则为二次非剩余 (Quadratic Nonresidue)，当 p 为奇素数时，二次剩余的数量为 $(p-1)/2+1$

4、Euler's theorem

给出一个 \mathbb{Z}_p 中的元素 x ，能否判断出有无平方根

定理： x 为 $(\mathbb{Z}_p)^*$ 内的二次剩余，等价于 $x^{[(p-1)/2]} \equiv 1 \pmod{p}$ (p 为奇素数，且 $x \neq 0$)

Example: in \mathbb{Z}_{11} : $1^5, 2^5, 3^5, 4^5, 5^5, 6^5, 7^5, 8^5, 9^5, 10^5$

$$= \begin{matrix} 1 & -1 & 1 & 1 & 1, & -1, & -1, & -1, & 1, & -1 \end{matrix}$$

$p=11$ 的例子，计算每个元素的 $(p-1)/2$ 次幂，因此，只有1、3、4、5、9为模11的二次剩余，其他不是定理只说了某个数是或者不是二次剩余，没说怎么计算某个数的平方根，即证明了其存在性，但是没说怎么计算平方根

勒让德符号 (Legendre Symbol) : $x^{[(p-1)/2]} \equiv 1 \pmod{p}$ 的简写，=1为Q.R.，=-1为二次非剩余

$$\left(\frac{a}{p} \right) = \pm 1 \equiv a^{\frac{p-1}{2}} \pmod{p}$$

5、Computing square roots mod p

如何计算模p (p为素数) 的平方根？

Suppose $p \equiv 3 \pmod{4}$

Lemma: if $c \in (\mathbb{Z}_p)^*$ is Q.R. then $\sqrt{c} = c^{(p+1)/4}$ in \mathbb{Z}_p

Proof: $\left[c^{\frac{p+1}{4}} \right]^2 = c^{\frac{p+1}{2}} = \underbrace{c_{\frac{p-1}{2}} \cdot c}_{=1} = c \quad \text{in } \mathbb{Z}_p$

When $p \equiv 1 \pmod{4}$, can also be done efficiently, but a bit harder

$$\text{run time} \approx O(\log^3 p)$$

分为两种情况

- 如果 $p \equiv 3 \pmod{4}$, 则 $\sqrt{c} = c^{(p+1)/4} \pmod{p}$
- 如果 $p \equiv 1 \pmod{4}$, 没有确定性算法找到模平方根, 但是随机性算法效率也还行, 大约在 $O(\log^3 p)$

6、Solving quadratic equations mod p

计算二次同余方程, 大致步骤和初中学的一元二次方程一样, 前提是 \mathbb{Z}_p 中有平方根

Solve: $a \cdot x^2 + b \cdot x + c = 0 \quad \text{in } \mathbb{Z}_p$

Solution: $x = \frac{(-b \pm \sqrt{b^2 - 4 \cdot a \cdot c})}{2a} \quad \text{in } \mathbb{Z}_p$

需要指出的是, 其中的分母 $2a$ 可以用扩展欧几里得算法找到, $\sqrt{b^2 - 4 \cdot a \cdot c}$ 如果存在的话需要用到上面的找平方根的算法

7、Computing e'th roots mod N

计算某个数模N的e次根 (N为合数), 和之前的问题一样, 这个e次根是否真的存在, 又是否有高效的算法计算之

就目前已知而言，计算这个e次根和分解N一样难，因为需要分解N为若干个素因子，然后分别计算这个数对于每个素因子的e次根，然后再将这些结果合并，就可以得到模N的e次根

W5 10-4 Arithmetic algorithms

1、Representing bignums

计算机内如何表示大整数

比如有一台64 bits机器，想要表达一个n bits整数（如n=2048），可以将其分为若干个32 bits分组，然后得到n/32个分组，合在一起就可以了（由于32 bits分组相乘结果小于64 bits，因此64 bits的机器分组为32 bits）

现代处理器有128 bits甚至更大的寄存器，并且支持乘法运算，因此通常可以计算比32 bits更大的分组

2、Arithmetic

接下来看基本计算，对于两个n bits的整数而言

对于加减法而言，都是线性时间内完成，即O(n)

对于乘法，笨一点的算法的复杂度是平方阶，Karatsuba在1960年给出了一个更优秀的算法，复杂度为O($n^{1.58}$)，但是通常用于n很大的情况（比如大多数密码学算法库中的乘法运算）

对于带余除法，复杂度同样是平方阶

3、Exponentiation

接下来用比较抽象的方法讨论指数运算问题

假设有一个有限循环群G，g为G的生成元，对于给定的x，计算g的x次幂

看上去很简单，如果硬算的话很费时间，尤其是g或x很大的时候就比较复杂了，甚至是指数阶的复杂度

因此介绍一个好的方法，重复平方法，例子如下

Example: suppose $x = 53 = (110101)_2 = 32+16+4+1$

$$\text{Then: } g^{53} = g^{32+16+4+1} = g^{32} \cdot g^{16} \cdot g^4 \cdot g^1$$

首先将幂数x写为二进制数，然后计算各个二进制位为1的对应的g的幂数，然后再全部乘起来就可以得到结果了

Input: g in G and $x > 0$; **Output:** g^x

write $x = (x_n x_{n-1} \dots x_2 x_1 x_0)_2$

$y \leftarrow g$, $z \leftarrow 1$

for $i = 0$ to n do:

if ($x[i] == 1$): $z \leftarrow z \cdot y$

$y \leftarrow y^2$

output z

example: g^{53}

<u>y</u>	<u>z</u>
g^2	g
g^4	g
g^8	g^5
g^{16}	g^5
g^{32}	g^{21}
g^{64}	g^{53}

如图，一个简单的做法是用两个寄存器y和z，y始终保存平方运算的结果，z作为累加器，用于计算g的不同幂之间的乘法，对于输入的x而言，将其表示为二进制数，然后不断迭代，每次迭代时y都会平方，而z只有在当前位为1时才将y的结果乘进来

4、Running times

重复平方是一个非常快速计算g的高次幂的算法，本质上的迭代次数为 $\log x$ ，假设每次乘法都是平方阶复杂度，则进行一次指数计算的复杂度为 $O[(\log x) \cdot n^2]$ ，而实际上指数运算很慢，基本上都是立方阶复杂度

W5 10-5 Intractable problems

1、Easy problems

如找到模N下x的逆、找到模素数p的多项式 $f(x)=0$ 的解等等问题，都比较好解决

但数论有一些问题还是比较难解决的

2、Intractable problems with primes

假设一个大素数p（比如有600 bits），然后记该素数的阶为p，根据上节课内容，若要计算 Z_p^* 内的某元素g的x次幂 g^x ，重复平方法是个很简单有效的方法

$$\text{Dlog}_g(g^x) = x \quad \text{where } x \text{ in } \{0, \dots, q-2\}$$

现在考虑一个反问题，即上图所示，对于给定的 g^x 的值，计算其模p的对数，即找到模p下的x，比较困难，该问题也叫离散对数问题（Discrete logarithm, Dlog）

Example: in \mathbb{Z}_{11} : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Dlog₂(·) : 0, 1, 8, 2, 4, 9, 7, 3, 6, 5

Dan Boneh

看个例子， $p=11$ 时以2为底的Dlog, mod 11时，2的0次幂为1, $2^8 \equiv 3 \pmod{11}$, 依次类推

例子中的素数很小，对于小素数而言只需要计算，然后建立一个表格，需要使用时查表就可以了，而对于大素数（超过2000 bits），计算离散对数相当困难

3、DLOG: more generally

更一般的定义离散对数问题

记G为一有限循环群，g为G的生成元，q为G的阶

$$G = \{1, g, g^2, g^3, \dots, g^{q-1}\}$$

定义：若G上的Dlog是困难的，则对于所有高效的算法A而言其如下概率可忽略

$$\Pr_{g \leftarrow G, x \leftarrow Z_q} [A(G, q, g, g^x) = x] < negligible$$

即在G中随机选择一个g，并随机选择一个指数x，给出g和 g^x 的值，所有有效计算该Dlog的概率可忽略不计时，称离散对数问题在这个群G上是困难的

常用于构造Dlog难题的群有

- 由大素数p生成的 Z_p^*
- 模p下的椭圆曲线（椭圆曲线上的点集）

前者上的Dlog比后者上的还要困难，因此椭圆曲线可以比 Z_p^* 使用更小的参数，并实现和 Z_p^* 中大素数同样困难的问题

4、Computing Dlog in (Z_p^*)

在 Z_p^* 上计算Dlog，之前讲过了，最好的算法为GNFS，一个亚指数阶的算法

5、An application: collision resistance

Dlog的应用：构建一个抗冲突的hash函数

首先选择一些计算离散对数困难的群G，且群G的阶q为一素数，选择G内的两个元素g和h，并定义如下hash函数

For $x, y \in \{1, \dots, q\}$ define $H(x, y) = g^x \cdot h^y$ in G

实际上这个hash函数是抗冲突的，因为找到H的冲突和计算G上的Dlog一样困难

Lemma: finding collision for $H(., .)$ is as hard as computing $Dlog_g(h)$

Proof: Suppose we are given a collision $H(x_0, y_0) = H(x_1, y_1)$

then $g^{x_0} \cdot h^{y_0} = g^{x_1} \cdot h^{y_1} \Rightarrow g^{x_0 - x_1} = h^{y_1 - y_0} \Rightarrow h = g^{x_0 - x_1 / y_1 - y_0}$

引理：找到H(.,.)的碰撞和计算h的离散对数Dlog_g(h)一样难

证明：假设已知碰撞 $H(x_0, y_0) = H(x_1, y_1)$ ，则意味着有如上等式，然后把g和h分别移项到等号两侧，得到第二个等式，然后去掉h的幂次，即用g表示h，得到最右侧的等式

$1/(y_1 - y_0)$ 意味着需要计算 g 模 p 的逆，而之前要求的 G 的阶为素数，确保了所有的元素都可逆，否则 $y_1 - y_0 = 0$ 意味着 $y_1 = y_0$ ，此时没有离散对数，又由第一个等式得到 $x_1 = x_0$ ，可以推出已知 $H(x_0, y_0) = H(x_1, y_1)$ 实际上不是冲突，而是两个相同的值，不满足冲突的含义（冲突一定是 $y_1 \neq y_0$ ）

需要注意的是，这个构造hash函数的方法有很好的抗冲突性，但是实际上因为效率问题并不怎么使用（比SHA-256要慢得多）

6、Intractable problems with composites

一些模合数的问题

假设有如下一个整数集 $\mathbb{Z}_{(2)}(n)$ （比如 $n=1024$ bits），集合内的元素 N 为两个 n bits的素数相乘

$$\mathbb{Z}_{(2)}(n) := \{ N = p \cdot q \text{ where } p, q \text{ are } n\text{-bit primes} \}$$

对于上述集合，有如下两个难题

Problem 1: Factor a random N in $\mathbb{Z}_{(2)}(n)$ (e.g. for $n=1024$)

Problem 2: Given a polynomial $f(x)$ where $\deg(f) > 1$
and a random N in $\mathbb{Z}_{(2)}(n)$

find x in \mathbb{Z}_N s.t. $f(x) = 0$ in \mathbb{Z}_N

1. 对于集合中随机选择的 N ，分解 N
2. 对于高次多项式 $f(x)$ 和集合中随机选择的 N 而言，找到多项式的一个解（注意 $f(x)$ 次数大于1，等于1就是线性方程了，很简单的）

高斯在1805年提到：区分素数和合数问题（即素性检验问题）以及将合数分解成素因子问题是算术中最重要最有用的问题之一

对于大合数分解而言，目前已知的最好算法为数域筛法（NFS），同样的亚指数阶的复杂度

目前记录为RSA-768（一个232位的十进制数），上百台机器花了两年分解，RSA-1024的复杂度约为RSA-768的一千倍，但是随着科技进步，有望在最近十年内完成

W5 知识梳理

W5 Problem Set && Programming Assignment

Q1

1.

Consider the toy key exchange protocol using an online trusted 3rd party

(TTP) discussed in Lecture 9.1. Suppose Alice, Bob, and Carol are three

users of this system (among many others) and each have a secret key

with the TTP denoted k_a, k_b, k_c respectively. They wish to

generate a group session key k_{ABC} that will be known to Alice,

Bob, and Carol but unknown to an eavesdropper. How

would you modify the protocol in the lecture to accommodate a group key

exchange of this type? (note that all these protocols are insecure against

active attacks)

- Alice contacts the TTP. TTP generates a random k_{ABC} and sends to Alice $E(k_a, k_{ABC})$, $\text{ticket}_1 \leftarrow E(k_c, E(k_b, k_{ABC}))$, $\text{ticket}_2 \leftarrow E(k_b, E(k_c, k_{ABC}))$. Alice sends k_{ABC} to Bob and k_{ABC} to Carol.
- Bob contacts the TTP. TTP generates a random k_{AB} and a random k_{BC} . It sends to Bob $E(k_a, k_{AB})$, $\text{ticket}_1 \leftarrow E(k_a, k_{AB})$, $\text{ticket}_2 \leftarrow E(k_c, k_{BC})$. Bob sends ticket_1 to Alice and ticket_2 to Carol.
- Alice contacts the TTP. TTP generates random k_{ABC} and sends to Alice $E(k_a, k_{ABC})$, $\text{ticket}_1 \leftarrow E(k_b, k_{ABC})$, $\text{ticket}_2 \leftarrow E(k_c, k_{ABC})$. Alice sends ticket_1 to Bob and ticket_2 to Carol.

问：考虑一个简易的密钥交换协议，使用TTP，假设Alice、Bob、Carol为该系统中的三个用户（还有其他用户），每个人都有自己的密钥，记为 k_a, k_b, k_c ，他们三人期望生成一个组内共享的密钥 k_{ABC} ，仅有他们三人知道，其他人和窃听者不知道，如何修改课程中介绍的协议以适应这种类型的组密钥交换（请注意：所有这些协议对于主动攻击都是不安全的）

答：第三个选项正确，窃听者只能看到 k_{ABC} 的加密的结果，而Bob和Carol可以正确使用 k_{ABC} ，前两个都不行

2. Let G be a finite cyclic group (e.g. $G = \mathbb{Z}_p^*$) with generator g .

Suppose the Diffie-Hellman function $\text{DH}_g(g^x, g^y) = g^{xy}$ is difficult to compute in G . Which of the following functions is also difficult to compute?

As usual, identify the f below for which the contra-positive holds: if $f(\cdot, \cdot)$ is easy to compute then so is $\text{DH}_g(\cdot, \cdot)$. If you can show that then it will follow that if DH_g is hard to compute in G then so must be f .

- $f(g^x, g^y) = g^{xy+x+y+1}$

正确

an algorithm for calculating $f(g^x, g^y)$ can
easily be converted into an algorithm for
calculating $\text{DH}(\cdot, \cdot)$.

Therefore, if f were easy to compute then so would DH ,
contradicting the assumption.

- $f(g^x, g^y) = (g^2)^{x+y}$

这个选项的答案不正确

It is easy to compute f as $f(g^x, g^y) = (g^x \cdot g^y)^2$.

- $f(g^x, g^y) = \sqrt{g^{xy}}$

- $f(g^x, g^y) = (\sqrt{g})^{x+y}$

这个选项的答案不正确

It is easy to compute f as $f(g^x, g^y) = \sqrt{g^x \cdot g^y}$.

问：记 G 为一有限循环群，生成元为 g ，假设D-H函数如图所示，其在 G 上计算困难，则下列哪个函数也是计算困难的？

答：

1. 由于 g^{xy} 难计算，所以第一个选项的表达式也难计算
2. 第二个选项表达式可以转换成 $g^x \cdot g^y$
3. 第三个同理
4. 第四个同理

Q3

3.

Suppose we modify the Diffie-Hellman protocol so that Alice operates as usual, namely chooses a random a in $\{1, \dots, p - 1\}$ and sends to Bob $A \leftarrow g^a$. Bob, however, chooses a random b in $\{1, \dots, p - 1\}$ and sends to Alice $B \leftarrow g^{1/b}$. What shared secret can they generate and how would they do it?

- secret = $g^{a/b}$. Alice computes the secret as $B^{1/b}$ and Bob computes A^a .
- secret = g^{ab} . Alice computes the secret as $B^{1/a}$ and Bob computes A^b .
- secret = $g^{a/b}$. Alice computes the secret as B^a and Bob computes $A^{1/b}$.

问：假设修改一下D-H协议，Alice和以前一样随机从 $\{1, \dots, p-1\}$ 中选择 a ，然后将 $A=g^a$ 发送给Bob，但Bob选择 b 之后计算 $B=g^{1/b}$ ，则他们之间会生成何种共享信息，应当如何操作？

答：还是和原来一样，Alice把收到的 B 和自己的 A 乘起来，Bob同理，最后得到共享信息 $g^{a/b}$ ，然后分别按原来的流程计算各自的参数即可

Q4

4. Consider the toy key exchange protocol using public key encryption described in [Lecture 9.4](#).

Suppose that when sending his reply $c \leftarrow E(pk, x)$ to Alice, Bob appends a MAC $t := S(x, c)$ to the ciphertext so that what is sent to Alice is the pair (c, t) . Alice verifies the tag t and rejects the message from Bob if the tag does not verify.

Will this additional step prevent the man in the middle attack described in the lecture?

- yes
- no
- it depends on what MAC system is used.
- it depends on what public key encryption system is used.



正确
an active attacker can still decrypt $E(pk', x)$ to recover x
and then replace (c, t) by (c', t')
where $c' \leftarrow E(pk, x)$ and $t \leftarrow S(x, c')$.

问：考虑一个采用公钥加密的密钥交换协议（如9-4中的那种），假设Bob发送 $c=E(pk,x)$ 给Alice时，在尾部加入一个MAC $t:=S(x,c)$ ，Alice此时收到 (c,t) ，然后Alice验证这个tag t ，如果验证不通过则拒绝这条消息，则这个额外的添加MAC的步骤可以防止MITM攻击吗？

答：不行，主动攻击者依然可以用自己的密钥来代替原来的pk，并重新计算密文和MAC

Q5

5. The numbers 7 and 23 are relatively prime and therefore there must exist integers a and b such that $7a + 23b = 1$.

Find such a pair of integers (a, b) with the smallest possible $a > 0$.

Given this pair, can you determine the inverse of 7 in \mathbb{Z}_{23} ?

Enter below comma separated values for a , b , and for 7^{-1} in \mathbb{Z}_{23} .

10,20



问：对于 $7a+23b=1$ 这个等式，找到一对整数对 (a,b) ，使其满足该等式且 a 为最小正整数，对于这个整数对 (a,b) ，能否确定7在mod 23下的逆？

答： $a=23n+10$, $b=-7n-3$, 取 $n=0$ 即可，即 $(a,b)=(10,-3)$

Q6

6. Solve the equation $3x + 2 = 7$ in \mathbb{Z}_{19} .

8

✓ 正确

$$x = (7 - 2) \times 3^{-1} \in \mathbb{Z}_{19}$$

问：计算 $3x+2 \equiv 7 \pmod{19}$

答：8

Q7

7. How many elements are there in \mathbb{Z}_{35}^* ?

24

✓ 正确

$$|\mathbb{Z}_{35}^*| = \varphi(7 \times 5) = (7 - 1) \times (5 - 1).$$

问：模35中与35互素的数有几个

答：由欧拉 φ 函数可知有24个

Q8

8. How much is $2^{10001} \pmod{11}$?

Please do not use a calculator for this. Hint: use Fermat's theorem.

2

✓ 正确

By Fermat $2^{10} \equiv 1 \pmod{11}$ and therefore

$$1 = 2^{10} = 2^{20} = 2^{30} = 2^{40} \pmod{11}.$$

$$\text{Then } 2^{10001} = 2^{10001 \pmod{10}} = 2^1 = 2 \pmod{11}.$$

问： $2^{10001} \pmod{11}$ 的结果

答: 由 $2^{10} \equiv 1 \pmod{11}$ 可知, $2^{10001} \equiv 2 \cdot (2^{10})^{1000} \equiv 2 \pmod{11}$

Q9

9. While we are at it, how much is $2^{245} \pmod{35}$?

Hint: use Euler's theorem (you should not need a calculator)

32

✓ 正确

By Euler $2^{24} \equiv 1 \pmod{35}$ and therefore

$$1 = 2^{24} = 2^{48} = 2^{72} \pmod{35}.$$

$$\text{Then } 2^{245} = 2^{245 \pmod{24}} = 2^5 = 32 \pmod{35}.$$

问:

答: 和上一题类似解法

Q10

10. What is the order of 2 in \mathbb{Z}_{35}^* ?

12

✓ 正确

$2^{12} = 4096 \equiv 1 \pmod{35}$ and 12 is the
smallest such positive integer.

问: 模35下2的阶

答: 12

Q11

11. Which of the following numbers is a

generator of \mathbb{Z}_{13}^* ?

- 7, $\langle 7 \rangle = \{1, 7, 10, 5, 9, 11, 12, 6, 3, 8, 4, 2\}$

✓ 正确

correct, 7 generates the entire group \mathbb{Z}_{13}^*

- 3, $\langle 3 \rangle = \{1, 3, 9\}$

✗ 这个选项的答案不正确

No, 3 only generates three elements in \mathbb{Z}_{13}^* .

- 5, $\langle 5 \rangle = \{1, 5, 12, 8\}$

- 2, $\langle 2 \rangle = \{1, 2, 4, 8, 3, 6, 12, 11, 9, 5, 10, 7\}$

- 10, $\langle 10 \rangle = \{1, 10, 9, 12, 3, 4\}$

问: \mathbb{Z}_{13}^* 的生成元是哪个

答:

Q12

12. Solve the equation $x^2 + 4x + 1 = 0$ in \mathbb{Z}_{23} .

Use the method described in [Lecture 10.3](#) using the quadratic formula.

14,5

✓ 正确

The quadratic formula gives the two roots in \mathbb{Z}_{23} .

问: mod 23下解二次同余式

答: 14和5

Q13

13. What is the 11th root of 2 in \mathbb{Z}_{19} ?

(i.e. what is $2^{1/11}$ in \mathbb{Z}_{19})

Hint: observe that $11^{-1} = 5$ in \mathbb{Z}_{19} .

13

✓ 正确

$$2^{1/11} = 2^5 = 32 = 13 \text{ in } \mathbb{Z}_{19}.$$

问：模19下2的11次根

答：13

Q14

14. What is the discrete log of 5 base 2 in \mathbb{Z}_{13} ?

(i.e. what is $\text{Dlog}_2(5)$)

Recall that the powers of 2 in \mathbb{Z}_{13} are $\langle 2 \rangle = \{1, 2, 4, 8, 3, 6, 12, 11, 9, 5, 10, 7\}$

9

✓ 正确

$$2^9 = 5 \text{ in } \mathbb{Z}_{13}.$$

问：模13下2为底5的离散对数

答：9

Q15

15. If p is a prime, how many generators are there in \mathbb{Z}_p^* ?

- $(p+1)/2$
- \sqrt{p}
- $p-1$
- $\varphi(p-1)$

✓ 正确

The answer is $\varphi(p-1)$. Here is why. Let g be some generator of \mathbb{Z}_p^* and let $h = g^x$ for some x .

It is not difficult to see that h is a generator exactly when we can write g as $g = h^y$ for some integer y (h is a generator because if $g = h^y$ then any power of g can also be written as a power of h).

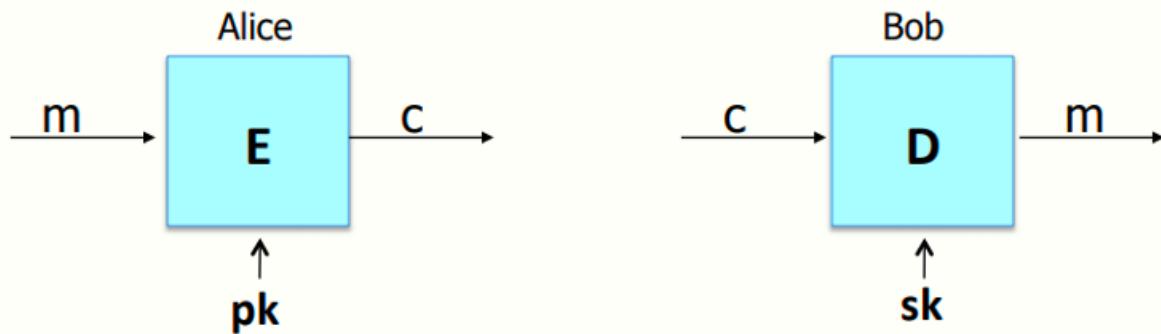
Since $y = x^{-1} \pmod{p-1}$ this y exists exactly when x is relatively prime to $p-1$. The number of such x is the size of \mathbb{Z}_{p-1}^* which is precisely $\varphi(p-1)$.

问：若p为素数，则 Z_p^* 的生成元有多少个

答：

W6 11-1 Public key encryption: definitions and security

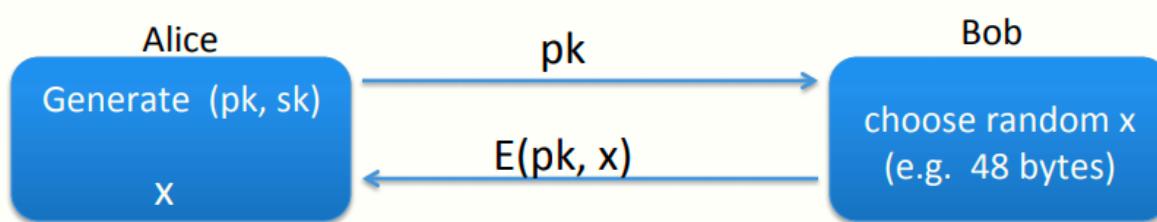
1、Public key encryption



和上一章讲的一样，公钥加密需要一个密钥对，公钥加密，私钥解密

2、Applications

常见的应用为浏览器和服务器之间建立安全密钥，模型如下（仅窃听安全）



对于非交互的应用，比如电子邮件也会用到

Bob先用Alice的公钥加密邮件信息后给他发邮件，然后Alice用自己的私钥解密即可

3、Public key encryption

回顾第九章的知识

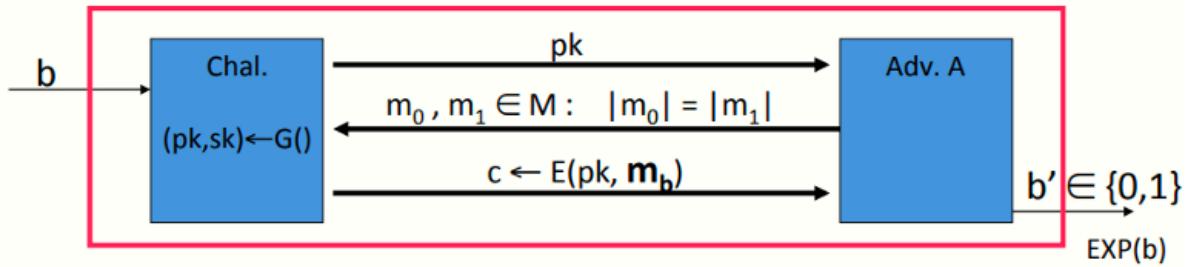
定义：公钥加密系统为一个算法三元组(G, E, D)，其中

- $G()$: 随机化算法，用于产生密钥对
- $E(pk, m)$: 加密算法，输入明文和密钥，输出密文
- $D(sk, c)$: 解密算法，输入密文和密钥，输出明文，有错误的情况下会输出bottom元素

公钥加密系统具有一致性，即对于任意由G输出的密钥对(pk, sk)，公钥加密后的消息再用私钥解密会得到原来的消息

4、Security: eavesdropping

接下来看语义安全的形式化定义，和以前一样，定义两个实验0和1让攻击者判断



Def: $E = (G, E, D)$ is sem. secure (a.k.a IND-CPA) if for all efficient A :

$$\text{Adv}_{\text{SS}}[A, E] = |\Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1]| < \text{negligible}$$

首先，挑战者先运行密钥生成算法 G 来生成密钥对，然后把公钥给攻击者，私钥保密

然后攻击者和以前一样，构造两个等长的消息并发送给挑战者，挑战者选择其中之一加密后返回给攻击者

需要注意的是，公钥系统的语义安全中没有必要允许攻击者做选择明文攻击，因为攻击者已经获得了公钥，可以用公钥加密任何期望的消息，而不像前几章那样发起CPA让挑战者帮助他创建他选择的消息的加密

定义：若公钥加密系统 $E = (G, E, D)$ ，则其对于任意高效的攻击者，其上述优势为可忽略的

5、Relation to symmetric cipher security

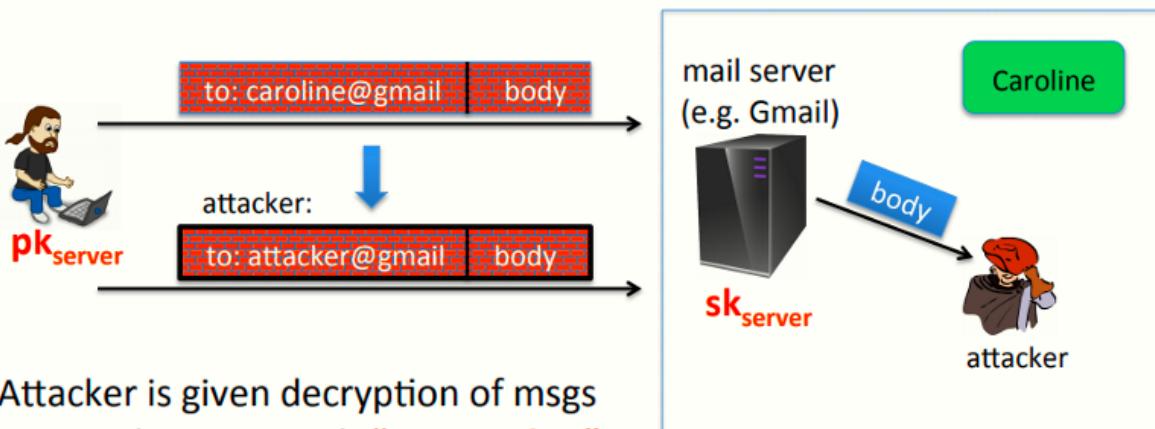
对称加密与公钥加密的关系

之前在讨论对称加密算法的安全时，我们关注密钥使用一次和多次的安全性

在公钥密码系统中，如果一个系统在一次性密钥时是安全的，则多次密钥也是安全的，也不必赋予攻击者请求加密他选择明文的能力，因为攻击者有公钥，可以加密任何其想要的消息

6、Security against active attacks

接下来看更为强大的主动攻击



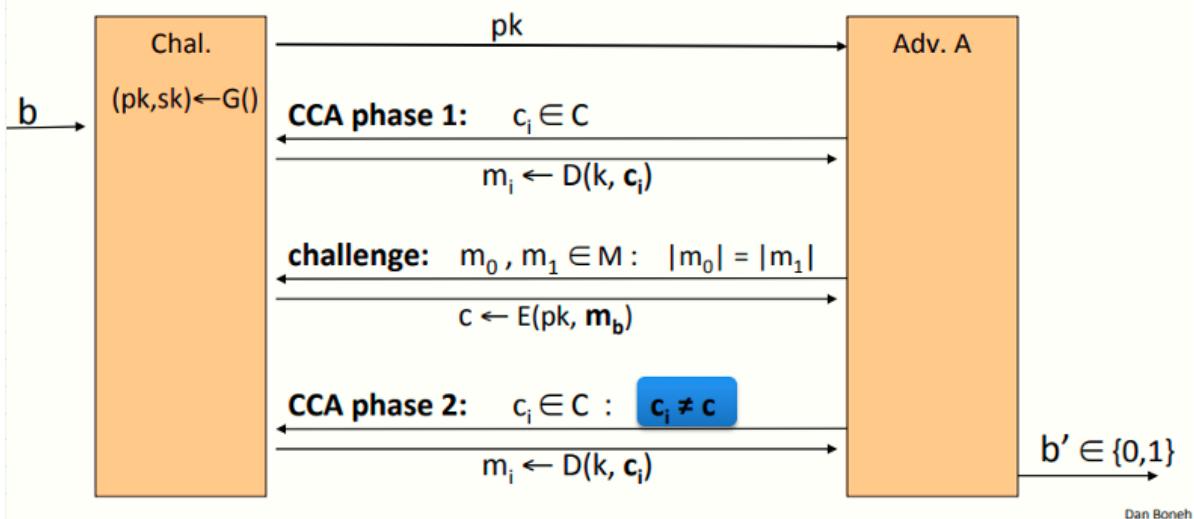
比如Bob想要发邮件给他的小伙伴Caroline，加密邮件先送到Gmail服务器，服务器解密邮件后查看收件人并转发给收件人

假设Bob使用的加密系统不安全，攻击者可以在不被察觉的情况下篡改密文，因此攻击者截获邮件后可以修改收件人为attacker

7、(pub-key) Chosen Ciphertext Security: definition

目标：即便攻击者在可以篡改密文或解密特定密文的情况下仍然安全的公钥系统

加密方案 $E = (G, E, D)$ ，明文与密文空间 (M, C) ，定义实验0和1



查询阶段1：攻击者从密文空间选择一系列消息 c_i ，请求挑战者为其解密为一系列对应的明文 m_i ，直至攻击者结束查询

挑战阶段：之后攻击者构造等长消息 m_0 和 m_1 ，并发送给挑战者，挑战者根据其选择的实验返回对应的密文 c

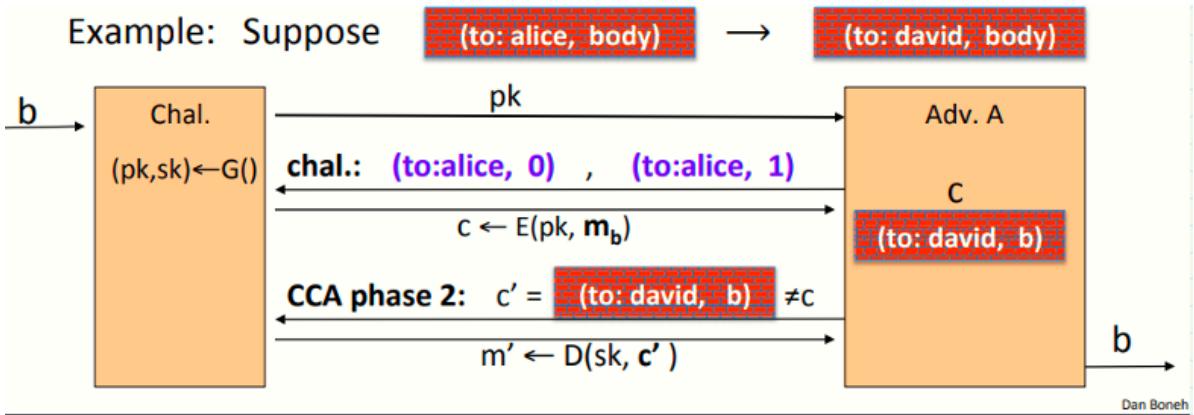
查询阶段2：最后，攻击者提交密文 $c_i \neq c$ ，挑战者返回其明文

此时攻击者可以选择解密任何与挑战密文不同的密文，并分辨挑战密文为 m_0 还是 m_1 的加密，如果不能，则代表这个系统是CCA安全的

定义：若 E 为CCA安全 (indistinguishability under chosen ciphertext attack)，则对任意高效的A而言，其如下优势可忽略

$$Adv_{CCA}[A, E] = |Pr[EXP(0) = 1] - Pr[EXP(1) = 1]|$$

看个例子



和之前的邮件的例子一样，假设加密系统仅给出一条消息的加密，而攻击者可以把收件人从Alice修改为Charlie，这会导致攻破上述安全模型

首先攻击者获得公钥 pk ，然后产生两条等长消息（如图），挑战者会返回其中一条消息的密文，而后攻击者用其能力，将Alice修改为Charlie，之后提交查询，从而知道第二步中返回的密文中的 b 是0还是1，从而攻击者赢得安全游戏的优势为1

8、Active attacks: symmetric vs. pub-key

安全的对称加密可以提供认证加密，即提供保密性和密文完整性，意味着攻击者不能构造一个通过认证的密文

公钥密码系统中，由于公钥 pk 公开，攻击者可以使用 pk 构造任何其想要的密文，因此需要确保选择明文安全

W6 11-2 Constructions

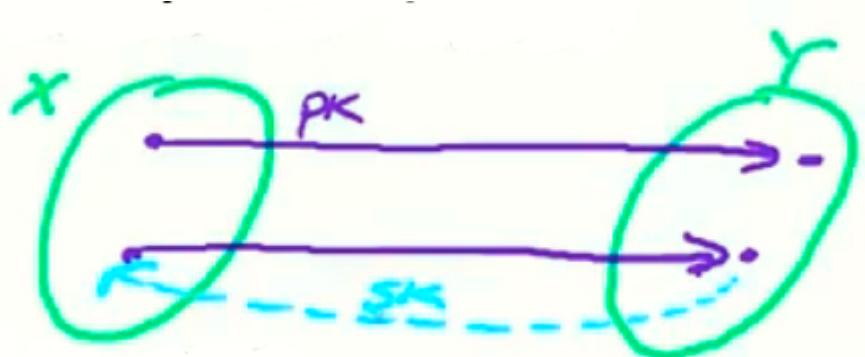
本节内容：陷门置换的概念

1、Trapdoor functions (TDF)

定义：陷门函数 $X \rightarrow Y$ 为一算法三元组 (G, F, F^{-1}) ，其中

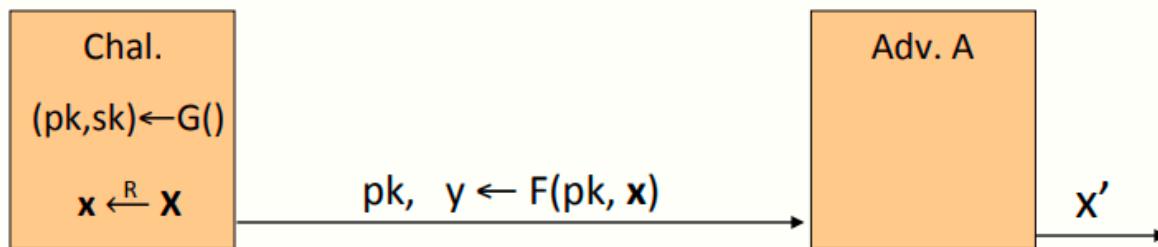
- $G()$: 生成公钥私钥对 (pk, sk)
- $F(pk, \cdot)$: 确定性算法，以 pk 作为输入，定义 $X \rightarrow Y$ 的函数
- $F^{-1}(sk, \cdot)$: 以 sk 作为输入，定义 $Y \rightarrow X$ 的函数，即 F 的逆函数

更具体而言，对于由 G 生成的任意密钥对 (pk, sk) ，任给消息 $x \in X$ ，都有 $F^{-1}(sk, F(pk, x)) = x$ ，也就是 pk 和 sk 对于集合 X 和 Y 有如下映射关系



2、Secure Trapdoor Functions (TDFs)

当 $F(pk, \cdot)$ 为一单向函数时， (G, F, F^{-1}) 为安全的TDF，也就是说，该函数 F 在某个点求其值很简单，在没有私钥 sk 的情况下求逆值很困难



同样以安全游戏的方式定义，挑战者先由 G 生成密钥对，然后计算 $y = F(pk, x)$ ，并把 pk 和 y 发给攻击者，攻击者的目标是求 y 的逆，记其输出为 x'

定义：若 (G, F, F^{-1}) 为一安全TDF，则其对于所有高效攻击者 A 而言，其如下优势可忽略

$$Adv_{OW}[A, F] = Pr[x = x'] < negligible$$

3、Public-key encryption from TDFs

利用陷门函数的概念建立公钥加密系统

(G, F, F^{-1}) : 为 $X \rightarrow Y$ 的安全TDF

(E_s, D_s) : 为定义在 (K, M, C) 上的对称加密系统，提供认证加密

$H: X \rightarrow K$, hash函数

需要注意的是，对称加密的密钥空间 K 和 TDF 里面的输入空间 X 不是同一个集合

$E(pk, m)$:

$x \xleftarrow{R} X, y \leftarrow F(pk, x)$
 $k \leftarrow H(x), c \leftarrow E_s(k, m)$
output (y, c)

$D(sk, (y, c))$:

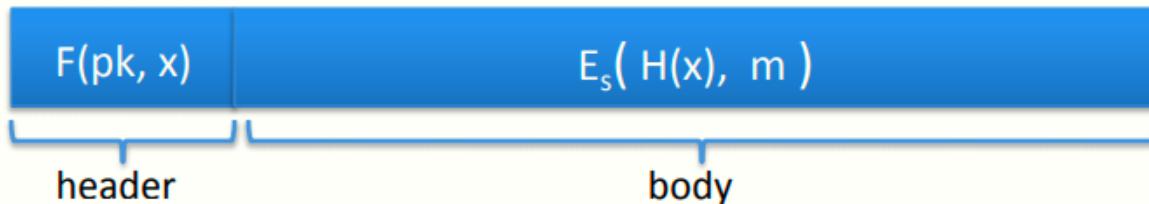
$x \leftarrow F^{-1}(sk, y),$
 $k \leftarrow H(x), m \leftarrow D_s(k, c)$
output m

首先运行算法 G 得到公私钥对 (pk, sk)

加密算法：以消息 m 和 pk 作为输入，先在集合 X 中随机生成一个 x ，由单向函数 F 计算值 y ，hash 函数计算 x 的 hash 值 k ，以 k 为密钥使用对称加密算法 E 计算消息 m 的密文 c ，加密算法输出 (y, c)

解密算法：以 sk 和 (y, c) 作为输入，先由 sk 计算 y 的逆，得到 x ，然后计算 x 的 hash 值 k ，由 k 和解密算法 D 计算 c 的明文 m

注意到加密过程中的陷门函数仅作用于算法随机生成的 x 而非输入的消息本身，使用的 hash 函数应当是一个理想的 hash 函数



从实际的角度来说，由于消息 m 可能是非常长的明文，如上图所示，先由首部得到密钥，再由密钥解密后面的消息

安全定理：如果 (G, F, F^{-1}) 为一安全 TDF， (E_s, D_s) 提供认证加密， $H: X \rightarrow K$ 为一随机预言 (random oracle，即 hash 函数应当是理想化的)，则 (G, E, D) 为 CCA-ro 安全 (ro 即 random oracle，随机预言模型)

4、Incorrect use of a Trapdoor Function (TDF)

需要注意一些错误的使用 TDF 的方式，比如直接将 F 函数作用于明文消息 m ，由于没有用到任何随机，加解密是完全确定的，几乎不可能做到语义安全，从而导致许多攻击方法（下一节介绍）

W6 11-3 The RSA trapdoor permutation

1、Review: trapdoor permutations

上一节课介绍了陷门函数建立公钥加密系统，本节介绍 RSA 的陷门函数

回顾一下上节课系统的组成：三个算法 (G, F, F^{-1})

本节课介绍陷门置换 (trapdoor permutations)，一个将 X 映射到 X 自身的函数 (区别于陷门函数将 X 映射到 Y)，但和陷门函数一样，陷门置换在没有私钥 sk 时求其逆是困难的

2、Review: arithmetic mod composites

然后再回顾一下数论的知识

记 $N=p \cdot q$ (p, q 为两个位数差不多的素数)

Z_N : 模 N 的所有可能的结果组成的集合, 即 $Z_N = \{0, 1, 2, \dots, N-1\}$

Z_N^* : Z_N 中存在逆的元素构成的集合, 也是 Z_N 中与 N 互素的元素构成的集合, Z_N 元素的个数记为 $|Z_N|$, 可以用欧拉函数求得

需要注意的是, 当且仅当 $\gcd(x, N) = 1$ 时, x 可逆

欧拉定理: $\forall x \in Z_N^*, x^{\varphi(N)} \equiv 1 \pmod{N}$

3、The RSA trapdoor permutation

历史: 最早在1977年8月提出, 已有超过40年历史, 由Rivest、Shamir、Adleman三人的首字母组成

应用: 广泛应用于SSL/TLS协议的认证和密钥交换部分, e-mail安全和文件系统安全, 以及其他一些需要安全解决方案的场景

首先看看密钥生成算法G()

- 选择两个大素数 p 和 q (1024 bits左右, 大约是300位的十进制数), 计算 $N=p \cdot q$, $\varphi(N)$, 然后选择两个整数 e 和 d , 使其满足 $e \cdot d = 1 \pmod{\varphi(N)}$, 之后输出公钥 $pk = (N, e)$, 私钥 $sk = (N, d)$

$F(pk, x)$: 一个 Z_N 到 Z_N 的映射, 简记为 $RSA(x) = x^e \pmod{N}$

$F^{-1}(sk, y) := y^d \pmod{N}$, 具体正确性如下

$$F^{-1}(sk, y) = y^d; \quad y^d = RSA(x)^d = x^{ed} = x^{k\varphi(N)+1} = (x^{\varphi(N)})^k \cdot x = x$$

4、The RSA assumption

为什么这个函数是安全的?

先声明一个RSA假设: RSA为一单项置换, 对于所有高效的算法A而言, 其如下概率可忽略

$$\Pr[A(N, e, y) = y^{1/e}] < negligible$$

其中 p 和 q 为 n bits素数, $N = pq$, y 为 Z_N^* 中随机选择的数, 对于算法A输入模数 N , 指数 e 以及点 y , 其计算 y 的逆的概率可忽略不计

上述假设大致说明了RSA是只给出公钥的单向置换, 因此这是一个陷门置换, 对于知道陷门(私钥)的人来说计算 y 的逆非常容易

5、Review: RSA pub-key encryption

有了单向陷门之后, 就可以将其应用于公钥密码系统

上一节提到的那个公钥密码系统中, (E_s, D_s) 表示提供认证加密的对称密码系统, hash函数提供对称密码的密钥, 将本节课的RSA单向陷门应用到该系统后, 工作流程如下:

- $G()$: 生成RSA参数, $pk = (N, e)$, $sk = (N, d)$
- $E(pk, m)$: 加密算法, 首先在 Z_N 中随机选择一个数 x , 计算 $y = RSA(x)$, $k = H(x)$, 输出 $(y, E_s(k, m))$

- $D(sk, (y, c))$: 输出 $D_s(H(RSA^{-1}(y)), c)$

实际应用中，hash函数采用SHA-256实现

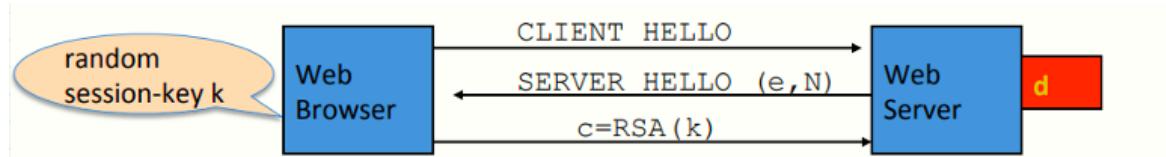
根据上节课介绍的安全定理，RSA为安全的TDF，E和D为提供认证加密的对称密码，H为随机预言，则该系统为CCA安全的

6、Textbook RSA is insecure

上面介绍了一个可用且好用的公钥加密系统，但是需要注意的是，不要用RSA加密，即不要直接用RSA加密给定的消息 m （所谓的教科书式的RSA），因为他是确定性加密，不可能是语义安全的，存在很多攻击

因此RSA只是一个陷门置换，本身不是一个加密系统，但是可以将其与其他东西组合来构建一个加密系统（比如上面介绍的那种）

接下来看一下针对这种所谓的教科书式的RSA的攻击



Suppose k is 64 bits: $k \in \{0, \dots, 2^{64}\}$. Eve sees: $c = k^e \text{ in } Z_N$

If $k = k_1 \cdot k_2$ where $k_1, k_2 < 2^{34}$ (prob. $\approx 20\%$) then $c/k_1^e = k_2^e \text{ in } Z_N$

Step 1: build table: $c/1^e, c/2^e, c/3^e, \dots, c/2^{34}^e$. time: 2^{34}

Step 2: for $k_2 = 0, \dots, 2^{34}$ test if k_2^e is in table. time: 2^{34}

Output matching (k_1, k_2) . Total attack time: $\approx 2^{40} << 2^{64}$

Dan Boneh

假设有一个web服务器，服务器有私钥(d, N)，此时浏览器期望建立安全会话，首先浏览器发送第一次握手，服务器第二次握手并返回其公钥(e, N)，浏览器直接使用RSA加密会话密钥 k （假设 k 为64 bits非负整数）得到密文 c

攻击者的工作：

假设 k 可以分解为大小差不多的两个数 k_1 和 k_2 ，且二者都小于 2^{34} （该事件发生的概率约为1/5）

由于公钥 e 公开，攻击者窃听到密文 c 后，可以将其原来的 k 用 $k_1 \cdot k_2$ 来替换，然后再移项得到上图中第一个灰色框的等式

然后进行中途相遇攻击，构建上述Step 1中的表（即等式左侧所有可能的值），然后计算 k_2 的 e 次幂（等式右侧的值）与表中的项匹配，找到碰撞后输出 (k_1, k_2) 即可，即 $k = k_1 \cdot k_2$

总的期望时间大约是 2^{40}

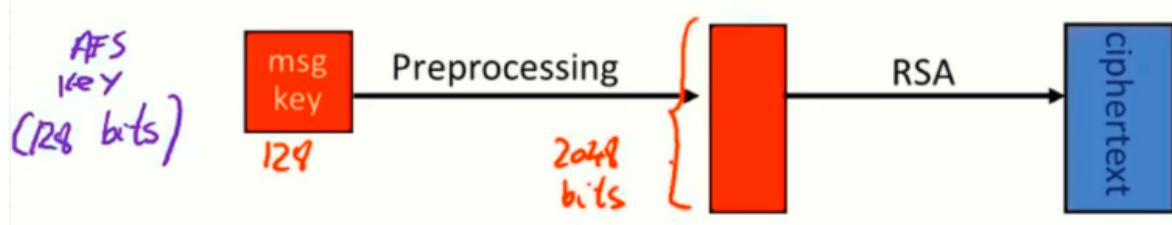
W6 11-4 PKCS 1

本节介绍RSA的实际运用，即PKCS 1

1、RSA encryption in practice

首先反复强调，不要使用教科书式的RSA，在使用RSA函数前需要对消息进行处理，比如生成随机数x，用RSA加密x，再通过x派生出对称加密密钥

上节课介绍了一些简单的模型，但不是实践中RSA的用法，实践中不同的是RSA加密给定的对称密钥而不是生成对称密钥作为RSA加密的一部分

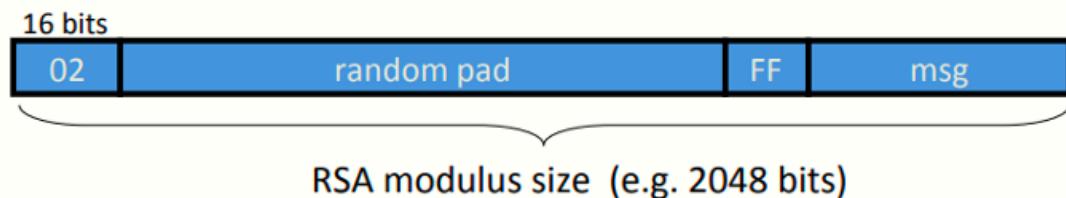


比如说一个128 bits的AES密钥，先将其扩展到2048 bits，然后再应用RSA函数

2、PKCS1 v1.5

PKCS广泛应用于1.5版本，模式2表示加密（模式1为签名）

PKCS1 mode 2: (encryption)



如图所示，尾部的msg表示需要扩展的128 bits的AES密钥，然后加入0xFF，之后加入随机填充（该随机填充任意一处不包含0xFF），最后在最前面加上0x02（表示PKCS采用模式2），最终得到一个2048 bits的字符串

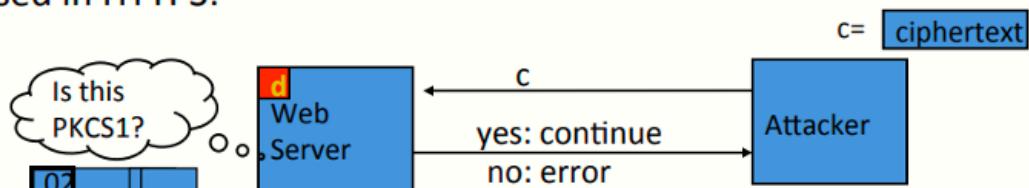
之后将上述整体作为RSA函数的输入，之后得到PKCS的密文，解密方解密后得到这个分组，然后移除0x02，移除直到连续的FF位置，最终得到后面的msg

有趣的是https用过这个1.5版本，实际上没有安全性证明这个模型真的安全，但确实曾经应用比较广泛

3、Attack on PKCS1 v1.5

1998年Bleichenbacher 提出了一个非常优雅 (pretty elegant) 的攻击

PKCS1 used in HTTPS:



⇒ attacker can test if 16 MSBs of plaintext = '02'

假设攻击者拦截了特定的PKCS1分组（还未输入RSA之前的那2048 bits），记RSA的输出为PKCS密文

攻击者直接向web服务器发送密文，web服务器尝试用私钥解密，查看解密结果的前两字节是否为0x02，是则继续，否则返回错误消息，因此攻击者可以得知该密文解密后的明文的前两字节是否为0x02

Chosen-ciphertext attack: to decrypt a given ciphertext c do:

- Choose $r \in Z_N$. Compute $c' \leftarrow r^e \cdot c = (r \cdot \text{PKCS1}(m))^e$
- Send c' to web server and use response

此时可以发起CCA如图所示：攻击者从 Z_N 中随机选择一个 r ，利用截获的密文 c 和 r 构造一个新的密文 c' （构造方式如图），然后将 c' 发送给服务器，服务器会返回前两个字节是否正确，只要构造的 c' 很多，最终可以恢复出想要解密的密文

4、Baby Bleichenbacher

还是介绍这个攻击，但是换一个相对简化的模型来介绍，假设web服务器解密之后不是检查前两个字节是否为 $0x02$ ，而是检查第一个bit是否为1，然后假设RSA的模数 N 为2的某个次幂（而不是实践中的两个大素数的积）

Suppose N is $N = 2^n$ (an invalid RSA modulus). Then:

- Sending c reveals $\text{msb}(x)$
- Sending $2^e \cdot c = (2x)^e$ in Z_N reveals $\text{msb}(2x \bmod N) = \text{msb}_2(x)$
- Sending $4^e \cdot c = (4x)^e$ in Z_N reveals $\text{msb}(4x \bmod N) = \text{msb}_3(x)$
- ... and so on to reveal all of x

此时攻击者发送密文 c 给web服务器后，通过观察服务器行为可以知道其明文的第一位是否为1

然后将2的 e 次幂与 c 相乘，可以得到 $(2x)^e$ ，也就是将 x 左移一位，然后将其发送给服务器，就可以得到 x 的第二位是否为1

然后将4的 e 次幂与 c 相乘，.....，.....左移两位.....得到 x 的第三位是否为1

重复上述步骤直到完全恢复 x

上述简化的Bleichenbacher只需要1000~2000次查询即可完成恢复，而完整的Bleichenbacher由于是检查前两字节是否为 $0x02$ ，因此大约需要上百万次查询才可以完成

5、HTTPS Defense

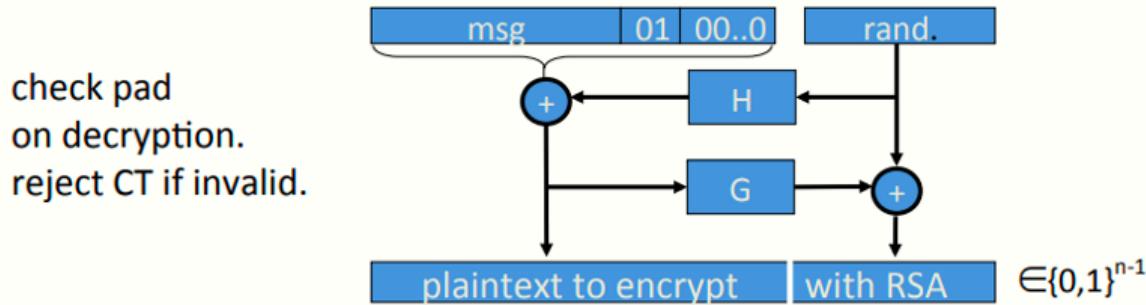
那么问题来了，如何防御上述攻击？RFC 5246给出了一些做法

如果解密之后不是 $0x02$ 开头的话，协议会选择一个随机的46字节的串R并假定主密钥是这个串R，但是不告知开头是否是 $0x02$ ，然后继续协议，稍后由于这个串没什么意义从而导致协议失效（客户端和服务端密钥不一致从而导致会话终止）

上述方法好处在于假装明文是某个随机值，且服务端的代码改动比较小，修改和部署代价不大

6、PKCS1 v2.0: OAEP

另一种使用RSA加密的方法：最优非对称加密填充（Optimal Asymmetric Encryption Padding，OAEP），1994年由Bellare和Rogaway提出，在PKCS1 v2.0版本支持OAEP，工作方法如下



假设需要加密的消息为msg（如128 bits的AES密钥），然后在其尾部附上填充（先是01，然后是若干bits的0，0的数量取决于标准），然后随机选择一个值rand，使得整个字符串（msg+填充+rand）和RSA的模数一样大（比如2047 bits）

先将rand输入hash函数H，产生和左侧串相等长度的hash值，并把这个hash值和左侧串异或，异或的结果输入另一个hash函数G，将G的输出与rand异或后得到下边右侧部分，两个部分拼在一起得到2047 bits长的串，最终这个串（下面的这个）将输入到RSA函数

2001年由Fujisaki、Okamoto、Pointcheval和Stern证明了一个理论：假设RSA为一安全陷门置换，则上述使用RSA的模式（RSA-OAEP）在H和G为理想的hash函数的条件下为CCA安全的

需要注意这个理论是依赖于RSA的特殊性质，如果使用一些其他的一般的不具有RSA的代数性质的陷门置换，则这个定理完全错误，实际使用时会使用SHA-256来作为H和G的hash函数

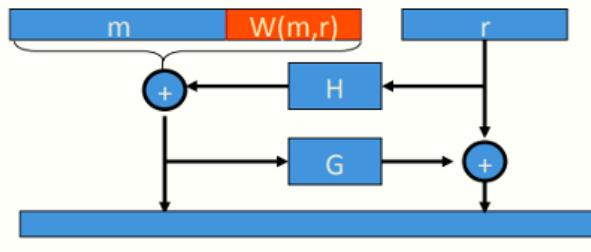
7、OAEP Improvements

那么如果我们只有普通的陷门置换，如何正确使用OAEP？可以，需要做一些微小的改动，两种做法

OAEP+: [Shoup'01]

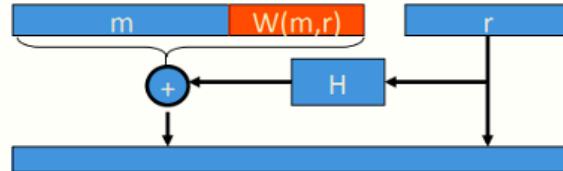
\forall trap-door permutation F
F-OAEP+ is CCA secure when
H, G, W are random oracles.

During decryption validate W(m,r) field.



SAEP+: [B'01]

RSA ($e=3$) is a trap-door perm \Rightarrow
RSA-SAEP+ is CCA secure when
H, W are random oracle.



第一种：OAEP+

任给陷门置换F，采用hash函数W(m,r)替换OAEP中的固定填充（其中m和r为原来的那些，作为W的输入），只要H、G、W为理想hash函数，这个OAEP+就是CCA安全的

第二种：SAEP+

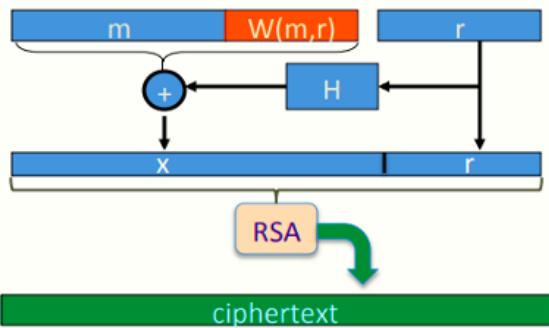
本质上还是以来RSA的性质，当公钥中的指数 $e=3$ 时，可以不进行第二阶段的加密，但与OAEP+中使用了hash函数W，同样的H和W为理想hash函数时为CCA安全

需要注意的是这些OAEP的变体实际上并没有使用，真正实际使用的是标准化的OAEP

还需要注意的是，加密中的填充检查在所有机制中都是很重要的，如果填充不正确说明密文无效（即系统输出了bottom元素）

随堂小作业：SAEP怎么解密

How would you decrypt
an SAEP ciphertext **ct** ?



- ○ $(x,r) \leftarrow \text{RSA}^{-1}(\text{sk},\text{ct})$, $(m,w) \leftarrow x \oplus H(r)$, output m if $w = W(m,r)$
○ $(x,r) \leftarrow \text{RSA}^{-1}(\text{sk},\text{ct})$, $(m,w) \leftarrow r \oplus H(x)$, output m if $w = W(m,r)$
○ $(x,r) \leftarrow \text{RSA}^{-1}(\text{sk},\text{ct})$, $(m,w) \leftarrow x \oplus H(r)$, output m if $r = W(m,x)$

8、Subtleties in implementing OAEP

实现OAEP很复杂，而且对时序攻击很脆弱，想用的话可以用OpenSSL里的OAEP

W6 11-5 Is RSA a one-way function?

1、Is RSA a one-way permutation?

RSA真的是一个单向置换吗？

攻击者已知公钥(e, N)和 x^e ，期望找到 x ，则这个过程有多难？即计算模N下的 e 次根有多难？

如果说真的很棘手的话，RSA就是一个单向函数，如果很容易则RSA就算被破解了

事实上目前已知最好的算法是分解大整数 $N=pq$ ，然后分别计算 p 和 q 的 e 次根，再通过中国剩余定理将结果恢复成模N的 e 次根，第二步很简单，棘手的点在于分解N

2、Shortcuts?

那问题又来了，一定要分解N吗？还是说有没有捷径可以不分解N就计算模N的 e 次根？

事实上可以将算法进行归约，如果存在一个高效的算法可以计算模N的 e 次方根，可以证明的是，这个算法可以被归约为一个因子分解的算法

这个证明意味着给定一个计算N模下的 e 次根的算法，同时也就有了一个因子分解的算法，也就是对N模下的 e 次根的计算不可能比对N进行因式分解更快，从而证明了破解RSA的难度与因子分解的难度是等价的

因子分解问题目前还尚未解决，事实上这是公钥加密算法中最古老的问题之一，看下面这个例子：

假设有一个算法可以计算模N下的立方根（即 $e=3$ ），则能否证明利用该算法可以求出N的因子？结论未知

还是上面这个假设，考虑 $e=2$ 的情况，即该算法可以计算模N下的平方根时，则该算法可以用于对N的因子分解，即计算模N下的平方根蕴含着因子分解的算法，二者难度相当

但是由于RSA的定义， $e \cdot d \equiv 1 \pmod{\phi(N)}$ ，因此 e 一定与 $\phi(N)$ 是互素的，又由于 p 和 q 都是大素数， $\phi(N)=(p-1)(q-1)$ 一定是个偶数，因此 e 和 $\phi(N)$ 一定不是互素的，从而即便是我们有这个优秀的算法， $e=2$ 也绝不可能作为RSA的密钥参数（事实上合法的最小RSA指数为3）

3、How not to improve RSA's performance

就目前所知，RSA是一个单向函数，想要破解即通过计算模N下的e次方根，就需要分解N

人们也做了许多工作，试图改善RSA的加解密算法性能，但大多失败了

比如说优化RSA的解密算法，由于解密算法为计算 $c^d \equiv m \pmod{N}$ ，并且该算法的复杂度与d的量级成正比，即复杂度为 $O(\log(d))$ ，为了加速RSA解密为什么不用一个量级很小的d？如 $d \approx 2^{128}$ 之类的

显然128 bits对于穷举搜索不太现实，但一般而言，解密指数d会和模数的大小差不多，即2000 bits左右，若使用128 bits的d可以将解密速度提高20倍左右，但是这个做法非常糟糕

Michael Wiener提出一种攻击方式，只要私钥指数 $d < N^{1/4}$ ，则RSA完全不可靠（而且是最糟糕的那种不可靠），即如果使用这种不安全的指数d，利用公钥(e,N)可以很快找到私钥d

那上述攻击仅针对512 bits以下的d，那么要是把d设置得高一点呢？实际上仍然不可靠，有一种Wiener攻击的拓展形式，表明 $d < N^{0.292}$ ，则RSA也是不可靠的，且针对该结论有人推测对于任何 $d < N^{0.5}$ ，该结论都成立，意思就是即便是d接近 $N^{0.4999}$ 也是不可靠的（但该推测仍然是个开放性话题，仍然具有争议性）

RSA不可靠：在只给定公钥e和N的情况下可以恢复私钥d

为什么是0.292？

$$0.292 \approx 1 - \frac{1}{\sqrt{2}}$$

因此结论是：不应改变d的结构来优化RSA

也有不少实验结果表明，通过此类优化RSA的小伎俩最终都导致了灾难，因此这不是一个优化RSA的正确方式

4、Wiener's attack

我们期望恢复私钥d，且已知 $d < N^{1/4}$ ，假设 $d < N^{0.25}/3$ （3不重要，1/4起决定作用），然后看看如何实施攻击

首先，我们有 $e \cdot d \equiv 1 \pmod{\varphi(N)}$ ，意味着存在整数k，使得 $e \cdot d = k \cdot \varphi(N) + 1$ ，等式两侧同除以 $d \cdot \varphi(N)$ ，移项之后得到如下等式

$$\left| \frac{e}{\varphi(N)} - \frac{k}{d} \right| = \frac{1}{d \cdot \varphi(N)} \leq \frac{1}{\sqrt{N}} \quad (1)$$

观察上述等式的左侧，e为已知，但 $\varphi(N)$ 未知，因此不知道 $e/\varphi(N)$ ，但是由于 $\varphi(N)$ 与N非常接近，因此可以用N近似代替 $\varphi(N)$ ，则 $e/\varphi(N)$ 近似于 e/N

回想一下欧拉 φ 函数

$$\varphi(N) = (p-1)(q-1) = N - p - q + 1$$

变形一下，去掉常数1，可以得到一个不等式

$$|N - \varphi(N)| \leq p + q$$

而p和q的大小都与 \sqrt{N} 差不多，所以上述不等式可以近似于如下不等式

$$|N - \varphi(N)| \leq p + q \leq 3\sqrt{N} \quad (2)$$

然后看回初始的假设

$$d \leq \sqrt[4]{N}/3$$

将该不等式两侧平方，然后取倒数，再乘 $1/2$ ，得

$$\frac{1}{2d^2} - \frac{1}{\sqrt{N}} \geq \frac{3}{\sqrt{N}} \quad (3)$$

然后看我们需要的式子（下式不等号左侧），用三角不等式展开如下

$$|\frac{e}{N} - \frac{k}{d}| \leq |\frac{e}{N} - \frac{e}{\varphi(N)}| + |\frac{e}{\varphi(N)} - \frac{k}{d}| \quad (4)$$

对于不等号右侧的第二个绝对值上面（1）已经推出来了，第一个绝对值做通分，将（2）带入分子，化简得到

$$|\frac{e}{N} - \frac{e}{\varphi(N)}| = \left| \frac{e(\varphi(N) - N)}{N \cdot \varphi(N)} \right| \leq \frac{e \cdot 3\sqrt{N}}{N \cdot \varphi(N)} \leq \frac{3}{\sqrt{N}} \quad (5)$$

注意到（5）中第一个不等号去掉了分子的 e 和分母的 $\varphi(N)$ ，由于 e 很小且 $\varphi(N)$ 很大，实际上是将该表达式放大了，然后约去分子分母的 \sqrt{N} ，得到最右侧结果

然后再将（1）（3）（5）带入（4），得到最后的不等式

$$|\frac{e}{N} - \frac{k}{d}| \leq \left| \frac{e}{N} - \frac{e}{\varphi(N)} \right| + \left| \frac{e}{\varphi(N)} - \frac{k}{d} \right| \leq \frac{1}{2d^2} - \frac{1}{\sqrt{N}} + \frac{1}{\sqrt{N}} \leq \frac{1}{2d^2}$$

观察这个不等式，我们知道 e/N 的值，我们期望知道 k/d ，由不等式可知这两者的差非常小，而出现这种情况的值非常少，只有少数的 k 和 d 才能出现这种情况

事实上这样的分式的个数近似于 $\log(N)$ ，因此存在一个连续性分式算法，可以从 e/N 推导出 $\log(N)$ 种可能的 k/d ，然后逐一尝试直到找到正确的 k/d

由于 $e \cdot d = k \cdot \varphi(N) + 1$ ，因此 $e \cdot d \equiv 1 \pmod{k}$ ，故 d 与 k 互素，如果使用有理分式来表示 k/d 的话，则其分母一定是 d ，即 $\log(N)$ 种可能的结果中肯定有一个分母是 d

结论：如果私钥指数 d 非常小，小于 $1/N^{0.25}$ ，那么可以很容易恢复 d 的值

W6 11-6 RSA in practice

1、RSA with Low public exponent

上节课提到了，如果想加速RSA的加密，使用小的加密指数 e 没问题，需要注意的是最小的合法的加密指数 $e=3$ （1显然不对，2与模数不互素）

一般而言，推荐使用 $2^{16}+1=65537$ ，利用重复平方法，只需要计算 $16+1=17$ 次乘法，比采用随机的 e 快得多得多，但也导致了RSA的加解密速度不对称（解密可能需要上千次乘法）

有一种小技巧可以稍微提升RSA解密速度，利用中国剩余定理（Chinese Remaindering Theorem，CRT），可以将RSA解密速度提高四倍，但仍然比加密慢得多（速度比例大约是1: 10~1: 30，随模数大小变化而变化）

加密比解密快得多是RSA特有的性质，下节课讲的ElGamal没有这种（ElGamal加解密速度差不多）

2、Key lengths

<u>Cipher key-size</u>	<u>RSA Modulus size</u>
80 bits	1024 bits
128 bits	3072 bits
256 bits (AES)	<u>15360 bits</u>

公钥密码系统想要做到与对称密码系统相当的安全性需要更长的密钥

3、Implementation attacks

RSA从数学角度来说几乎完美，但是还是容易遭到侧信道攻击以至于不可靠

Paul Kocher在1997年提出的一种时序攻击，在测量计算 $c^d \pmod{N}$ 的时间开销可以分析出d的值，如果想自己实现RSA解密，需要确保解密时间与参数无关

还是这个Paul Kocher，1999年提出了一个攻击，如果有一个实现RSA的解密卡，可以通过测量解密时的电量消耗，可以分析出d

最后一种称为缺陷攻击，表明RSA易受到揭秘错误的损害，即RSA解密中出现错误可能导致密钥完全泄露，因此许多密码库在解密后会验证结果，验证完毕后才把结果返回给调用者，但由于验证需要一定时间开销（大约是解密的1/10），因此有些地方也不采用

因此如果自己实现RSA的话，的确是数学上正确的，可以正常使用，但是会存在许多潜在的攻击，因此一定要用标准库

4、An Example Fault Attack on RSA (CRT)

$$\left. \begin{array}{l} \text{decrypt mod } p: x_p = c^d \text{ in } Z_p \\ \text{decrypt mod } q: x_q = c^d \text{ in } Z_q \end{array} \right\} \text{combine to get } x = c^d \text{ in } Z_N$$

RSA解密通常是分别计算模p和模q的结果 x_p 和 x_q ，然后再利用CRT将两者结合起来，这样可以把解密速度提高四倍

然后展示一例对于RSA-CRT的缺陷攻击，假设解密库正在计算模p的结果，出于某种原因，处理器犯了个错误，导致其没有输出正确的 x_p ，而 x_q 没有出错，在这种情况下，将输出记为 x' ，即有如下图所示， x' 对于模p结果正确而模q结果不正确

$$x' = c^d \text{ in } Z_p \quad \text{but} \quad x' \neq c^d \text{ in } Z_q$$

此时将上述模p的等式两侧取e次幂，由于e与d互为逆元，因此等式右侧取e次幂后得到c，左侧为 $(x')^e$ ，即 $(x')^e \equiv c \pmod{p}$ ，但相同的操作（取e次幂）模q时不等于c，因此有如下最大公约数表达式

$$\gcd((x')^e - c, N) = p$$

即二者的最大公约数为p，由于 $N=p \cdot q$ ，而 $(x^e)^d - c$ 为p的倍数，因此其与N的最大公约数为p

因此通过上述错误，可以得到N的因子分解之一，得到因子分解之后就可以计算 $\phi(N)$ ，然后再根据公钥e计算出私钥d，也就是说，只需要解密的一个小错误就可以导致恢复私钥，因此RSA解密时应当检查结果（尤其是使用CRT来加速解密过程的时候）

5、RSA Key Generation Trouble

Heninger et al 和 Lenstra et al 在2012年发现一个新的攻击，表明如果RSA产生密钥时的熵很小，就会出现问题

```
prng.seed(seed)
p = prng.generate_random_prime()
prng.add_randomness(bits)
q = prng.generate_random_prime()
N = p*q
```

OpenSSL生成RSA密钥的方法如图所示，先给伪随机数生成器一个种子，然后使用了伪随机数生成器生成的随机字符串来生成第一个素数p，接下来再次赋予种子并生成另一个随机数，从而得到素数q，最后输出p和q的乘积

假设上述过程是在某个路由器或者防火墙内部实现的，同时假设生成密钥恰好在防火墙开启后不久（比如防火墙刚启动后），prng的熵很低，导致其很可能从一个低熵的内核熵池生成素数p（即p可能的取值很少），而得到p之后，防火墙有了更多的熵，则素数q在一个更大的内核熵池中生成

因此会带来一个问题，许多不同的防火墙中生成RSA密钥时，很可能会生成相同的素数p，但是q会不同，如果观察两个不同防火墙生成的两个RSA的模数 N_1 和 N_2 ，并计算两者最大公约数，由于p相同而q不同，因此实际上找到了一个因子p，从而分解了两个大整数

这两个人的团队做了些实验，扫网并找出不同网络服务器的公钥，利用算术技巧计算最大公因数并进行因子分解，实验结果表明他们可以对大约0.4%的SSL公钥进行因子分解

教训：生成密钥时（无论使RSA密钥、ElGamal密钥还是对称密钥），使用合适的随机数种子非常重要，因此不要开机后立即生成密钥，确保生成器有足够的时间获取足够的熵，然后再生成密钥

6、Further Reading

Why chosen ciphertext security matters,V.Shoup,1998，讨论了CCA安全在公钥设置中的重要性，讨论了许多蓄意的攻击形式

Twenty years of attacks on the RSA cryptosystem,D.Boneh,Notices of the AMS,1999，Dan Boneh教授自己写的，详细探讨了对RSA的攻击

OAEP reconsidered,V.Shoup,Crypto 2001

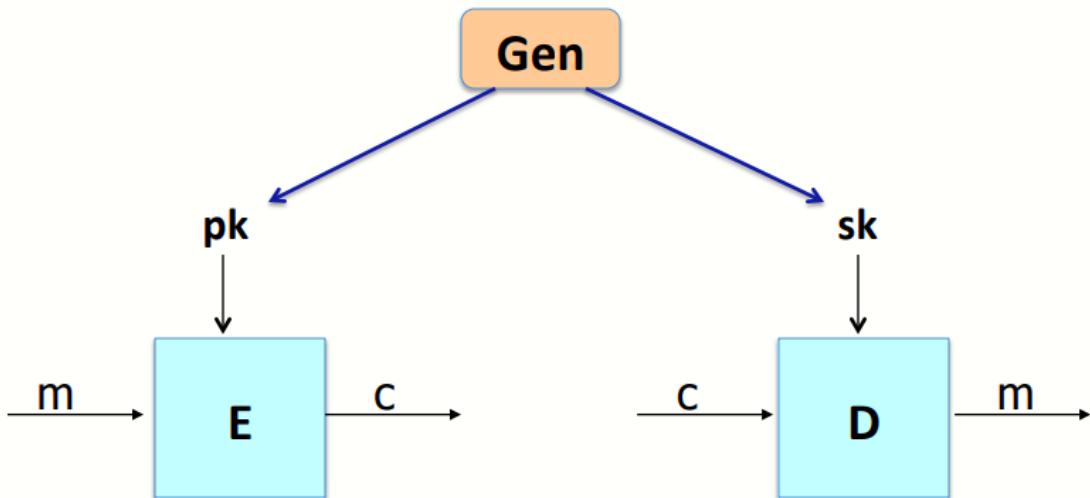
Key lengths,A.Lenstra,2004，分析了RSA密钥长度和其他公钥系统

W6 12-1 The ElGamal Public-key System

上一章讨论了基于RSA的公钥加密系统（一种建立在陷门函数基础上的系统），本章介绍另一个公钥加密系统，建立在D-H密钥交换协议上

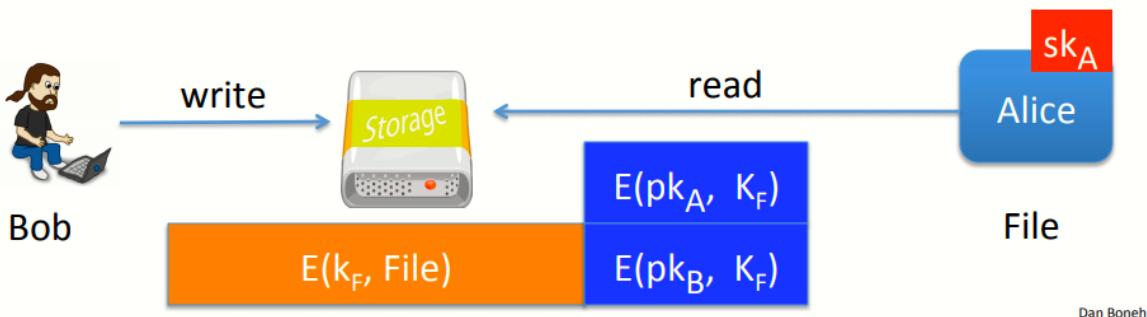
1、Recap: public key encryption: (Gen,E,D)

回忆一下，公钥系统由三个算法G、E、D组成，其中加密算法E使用公钥，解密算法D使用私钥



2、Recap: public-key encryption applications

上一章中还讨论了公钥加密系统的一些应用，如密钥交换（HTTPS），还有一些非交互的应用（如安全Email、文件系统加密等）



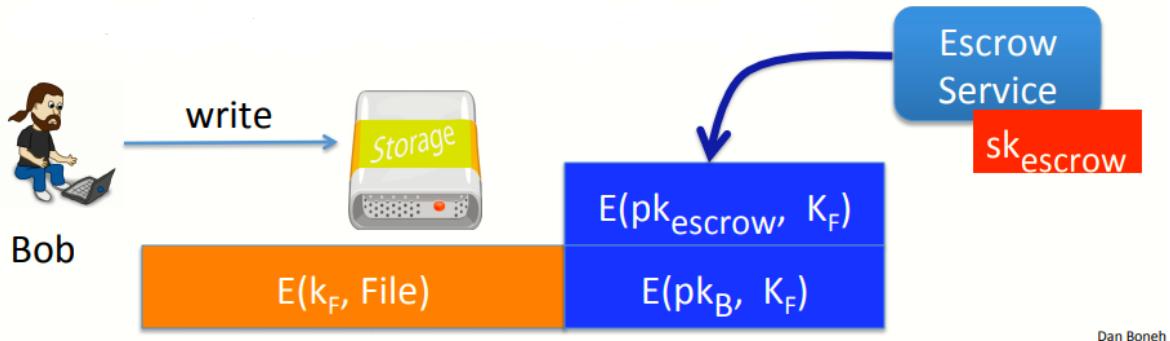
比如文件系统加密，Bob想要加密文件并保存到某个存储服务器上，则其只要保存加密过的文件到服务器上即可

首先生成一个随机的文件加密密钥 K_F ，然后用对称加密系统与该密钥加密这个文件，接着将 K_F 用其自己的公钥 pk_B 加密，并附在文件首部，这样Bob就能在以后的某个时间访问这些文件（先用私钥恢复 K_F ，然后用 K_F 恢复文件）

如果此时Bob希望Alice也有权限访问这个文件，则需要在文件首部加上用Alice的公钥 pk_A 加密的 K_F ，且这个过程无需Bob与Alice通信，只需要Bob获取Alice的公钥即可将该文件分享给Alice，同样Alice访问文件也无需与Bob交互

非交互情景的公钥加密系统还有一种应用，成为密钥托管（看起来很蠢），实际上是在企业环境中必备的一个功能

比如Bob把数据写到了磁盘里，然后一段时间后Bob不见了（离职、休假、出差等），但是公司需要Bob保存的文件，如果此时Bob是唯一可以解密的人，显然非常不现实，因此企业必须要有一种方式可以访问到Bob的文件，因此提出了密钥托管的概念



Bob 将他的文件写入磁盘时，这个系统会把他的文件写入到共享的媒介中，系统先向常一样用密钥 K_F 加密，然后用 Bob 的公钥 pk_B 加密，并记录到文件头中，然后系统会把 K_F 用密钥托管服务的密钥 pk_{escrow} 加密一次（该过程密钥托管系统处于离线状态）

此时如果公司需要恢复 Bob 的文件，公司会联系密钥托管服务，读取文件头并用私钥解密，恢复 K_F ，然后用 K_F 解密文件

3、Constructions

之前的课程中提到了许多建立在陷门函数基础上的系统，比如说 ISO 标准和 OAEP+ 等其它系统

本章会介绍建立在 D-H 密钥交换协议上的公钥加密系统，称为 ElGamal 公钥加密方案（ElGamal 方案由 Taher Elgamal 提出，T.E. 其实是 Marty Hellman 的学生，提出后又作为了他的博士论文的一部分），但由于一些历史原因，ElGamal 还被用在了 GPG 邮件加密系统中（GNU Privacy Guard）

和以往一样，构造公钥加密系统是，目标是构造一个满足 CCA 安全的系统，这样即可以防止监听，又可以防止篡改

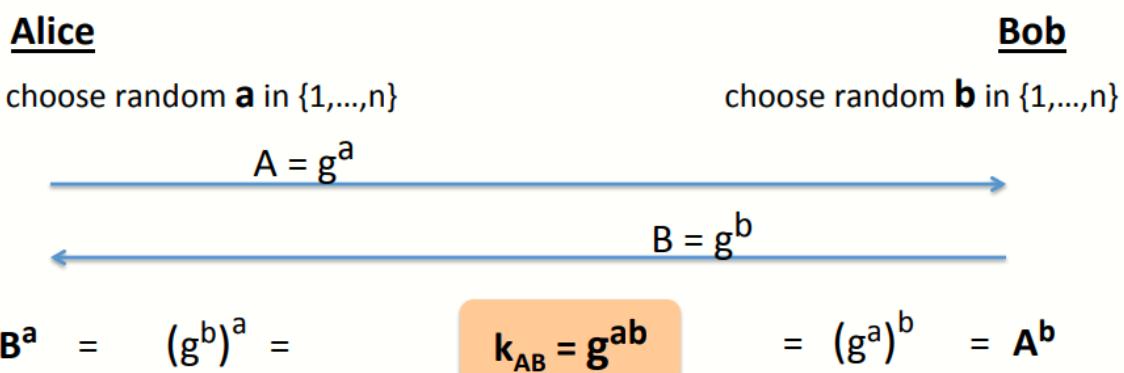
4、Review: the Diffie-Hellman protocol(1997)

先回顾一下 D-H 密钥交换协议（回避之前讲的要抽象一些）

有限循环群 G : 比如 $(Z_p)^*$, 或者可以是椭圆曲线上的点集，记其阶为 n

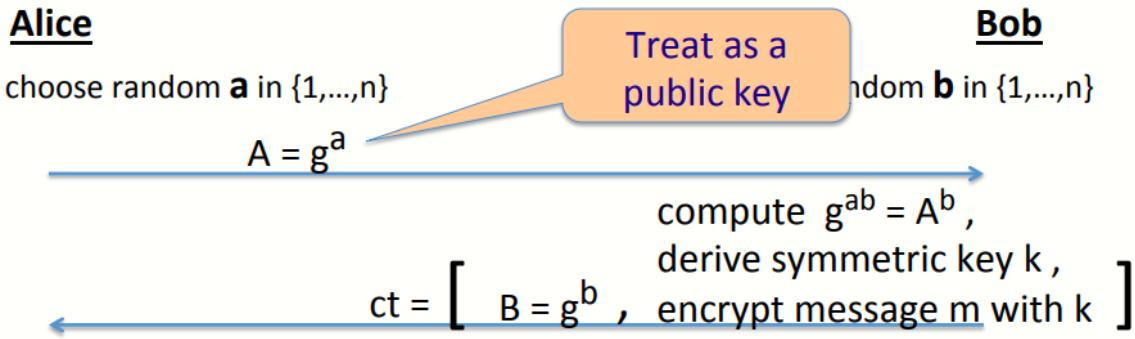
生成元 g : 由 g 的不断幂运算可以得到群 G 中所有元素，注意到 G 的阶位 n ，因此 G 的元素为 $g^0=1$ 到 g^{n-1}

然后回顾一下 D-H 协议



5、ElGamal: converting to pub-key enc.

然后看看 D-H 协议如何转化成公钥系统，还是 D-H 协议



首先确定群G和生成元g，第一步是密钥生成，将A视为公钥而a视为私钥，若期望由公钥推算出私钥本质上是解决一个离散对数问题

假设Bob需要将加密信息发送给Alice，此时Bob需要完成D-H密钥交换中他自己的那一部分，即随机选择b并计算 $B=g^b$ ，并且计算共享密钥 $k=g^{ab}$ ，然后利用该共享密钥加密自己要传输的信息，并将B与加密消息一起发送给Alice

其实这个流程和D-H协议完全一样，只不过Bob会立即使用共享密钥发送信息

此时Alice收到后，利用Bob的B计算出共享密钥k，然后用k解密消息即可

值得注意的是，某种程度上说这是一种随机加密方法，Bob每次想要加密一个消息时都需要重新选择一个随机的b，并用这个b生成密钥并加密消息

6、The ElGamal system (a modern view)

接下来更详细的ElGamal系统，先确定一些东西

- G：阶为n的有限循环群
- (E_s, D_s) ：定义在(K,M,C)上的对称加密系统，提供认证加密
- H：hash函数，将一堆群中的元素映射到密钥空间，即 $G^2 \rightarrow K$

然后定义公钥加密系统(Gen,E,D)，其中：

- Gen：密钥生成算法，从群G中随机选择一个生成元G，然后在 Z_n 中随机选择一个指数a作为私钥，公钥 $pk=(g, h=g^a)$ （不采用固定生成元的目的是增加安全性，随机选择一个生成元也不是什么很难的工作）
- (E, D) ：加解密算法

E(pk=(g,h), m) :

$$\begin{aligned} b &\leftarrow Z_n, u \leftarrow g^b, v \leftarrow h^b \\ k &\leftarrow H(u, v), c \leftarrow E_s(k, m) \\ \text{output } &(u, c) \end{aligned}$$

D(sk=a, (u,c)) :

$$\begin{aligned} v &\leftarrow u^a \\ k &\leftarrow H(u, v), m \leftarrow D_s(k, c) \\ \text{output } &m \end{aligned}$$

比如Bob想要加密消息：

1. 先从 Z_n 中随机选择b（即完成D-H协议中Bob发给Alice的那一部分），计算 $u=g^b$, $v=h^b=g^{ab}$
2. 利用u和v计算对称加密需要的密钥 $k=H(u, v)$ （由于u将要发送出去，因此攻击者可以接获到u的值，而v是攻击者不知道的，因此出于安全考虑，将u和v一起计算hash值会比较好）
3. 然后加密消息 $c=E_s(k, m)$
4. 最后算法输出 (u, c)

解密消息流程如下：

1. 计算 $v=u^a$
2. 计算对称密钥 $k=H(u,v)$
3. 由对称密钥解密密文 c
4. 输出明文 m

7、ElGamal performance

加密过程中有两个幂运算的步骤（计算 u 和 v ），解密只有一次幂运算（计算 v ），由于加密时的幂运算的底数 g 和 h 都由公钥推导出，因此每次都是不变的，而解密时的底数 u 每次都不同，因此加密速度并不完全是解密速度的两倍

由于底数不变，因此可以通过提前计算重复平方的每一步来加速加密过程，如果有更大的表来保存提前计算好的值的话，可以更快完成计算

W6 12-2 ElGamal Security

1、Computational Diffie-Hellman Assumption

之前介绍D-H密钥交换协议时，提到了攻击者有 g 、 g^a 、 g^b ，而计算共享密钥 g^{ab} 是困难的，这里的困难指的是计算性DH假设（CDH），接下来更深入的看这个假设

考虑一个 n 阶的有限循环群 G ，如果对于任意高效算法 A ，其由 g 、 g^a 、 g^b 计算得到 g^{ab} 的概率可忽略不计，则称在群 G 上满足CDH假设，即满足如下不等式

$$Pr[A(g, g^a, g^b) = g^{ab}] < negligible$$

2、Hash Diffie-Hellman Assumption

上述假设并不能满足ElGamal的安全性分析，因此引入一个更强的假设，Hash Diffie-Hellman假设

还是考虑一个阶为 n 的有限循环群 G ，引入一个hash函数，其将 G 中的一对元素映射到密钥空间 K ，即 $G^2 \rightarrow K$

则Hash-DH（HDH）假设在 (G, H) 满足下列条件时成立

- 在群 G 中随机选择一个元素 g ， Z_n 中随机选择 a 和 b ， K 中随机选择 R ，则下列分布在计算上不可区分

$$(g, g^a, g^b, H(g^b, g^{ab})) \approx_p (g, g^a, g^b, R)$$

H 为ElGamal系统计算得到的对称密钥， R 为密钥空间随机选择的值，如果两者不可区分（即一个真随机的独立密钥看起来像是由 g^a 和 g^b 派生的密钥一样），则HDH成立

DHD假设是一个条件更强的假设，强于CDH假设，即当HDH是困难的话，则CDH也是困难的（可以反证法证明），有些群上CDH是困难的，但HDH不是（理解不了的话就记住一句话：HDH可以证明ElGamal系统是语义安全的）

看一个小例题

Suppose $K = \{0,1\}^{128}$ and

$H: G^2 \rightarrow K$ only outputs strings in K that begin with 0
 (i.e. for all $x,y: \text{msb}(H(x,y))=0$)

Can Hash-DH hold for (G, H) ?

- Yes, for some groups G
- No, Hash-DH is easy to break in this case
- Yes, Hash-DH is always true for such H

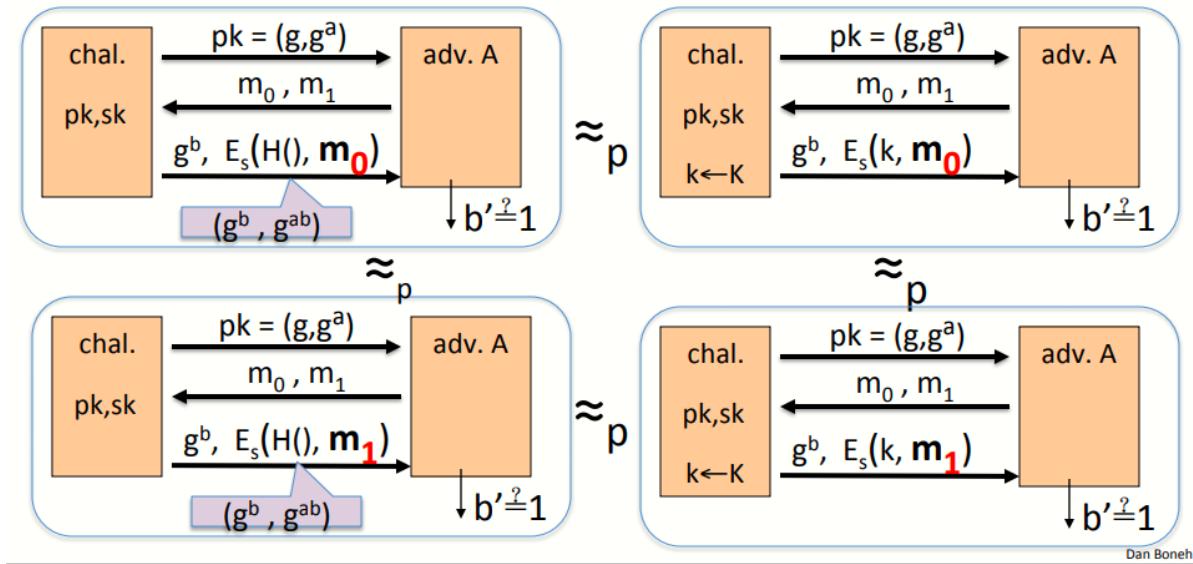
假定我们的密钥空间是一个128 bits的01比特流，哈希函数 H 会把 G^2 上的元素映射到 K 中，如果我们选择一个hash函数，其只输出第一位是0的比特流，那么HDH假设在 (G,H) 上成立吗？

由于hash函数输出的第一位总是0，则可以很容易区分这两个分布，从 K 中随机选择的元素第一位为0和1的概率都为0.5，而hash函数的分布总是输出0，因此可以区分两个不同的分布

结论：即便CDH在群 G 中是困难的，如果选择了一个坏的hash函数，则HDH假设在 (G,H) 也不成立

3、ElGamal is sem. secure under Hash-DH

要证明在HDH假设下的ElGamal是语义安全的，还是和以前一样，我们假设2种不同的实验（如下图所示）



挑战者首先将公钥发送给攻击者，攻击者选择两个等长的消息返回给挑战者，然后挑战者向攻击者返回对应的ElGamal加密过的消息

然后由DHD假设可知，攻击者知道 g, g^a, g^b ，但无法区分经过hash函数计算的 g^b 和 g^{ab} （对攻击者而言和随机数一样无法区分）

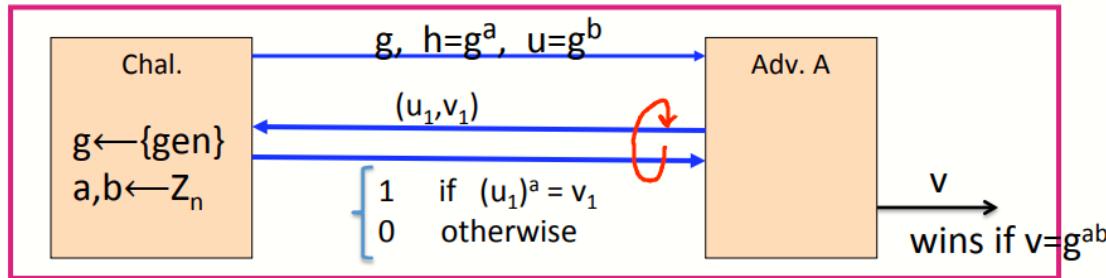
因此如果在 $H()$ 处用随机生成的密钥 k 取代hash函数（上图右侧），由HDH假设可知，攻击者无法区分两个不同的实验，又由于对称加密系统是语义安全的，因此ElGamal系统是语义安全的

4、ElGamal chosen ciphertext security?

仅仅做到语义安全是不够的，我们想要的是选择密文安全，问题在于ElGamal系统是CCA安全的吗？但就目前的HDH或CDH假设不足以证明该系统是CCA安全，需要一个更强的假设来证明

接下来介绍一个新的假设，交互D-H假设（Interactive Diffie-Hellman, IDH），用其来进行CCA分析

Interactive Diffie-Hellman (IDH) in group G:



和之前一样，挑战者将 g 、 g^a 、 g^b 发送给攻击者，若攻击者找到了密钥 g^{ab} 则算其成功，和CDH没有区别，但我们赋予了攻击者查询的能力，因此条件比CDH更强

假设攻击者从G中选取了两个元素 u_1 和 v_1 并发送给挑战者，挑战者会告知 $(u_1)^a$ 是否等于 v_1 （ElGamal的加密步骤中需要计算 $u=g^b$, $v=g^{ab}$ ）

假设赋予攻击者如此能力，且可以进行人亦多次的查询，仍然无法找到D-H的密钥，换句话说，即便是完成了所有的询问，其找到密钥的可能性仍然可以忽略不计，即有如下不等式

$$\forall \text{ efficient } A : \Pr[A \text{ outputs } g^{ab}] < \text{negligible}$$

显然，若IDH假设成立，则CDH假设也成立（因为IDH假设比CDH假设更强）

结论：当IDH假设成立，且由对称加密系统提供认证加密，涉及到的hash函数均为理想函数的情况下，ElGamal系统为CCA安全的

新的问题：能否摆脱IDH假设，即能否在CDH假设的基础上证明CCA安全，同样的能否不依赖随机性来证明安全性（即不需要假定hash函数是理想的），下节课讨论

W6 12-3 ElGamal Variants With Better Security

1、Review: ElGamal encryption

复习一下ElGamal的加解密流程（看之前的内容）

2、ElGamal chosen ciphertext security

上一节末尾提到了能否摆脱IDH，在CDH的基础上证明CCA安全？

可以，有两种方法可以做到：

- 使用CDH=IDH的群（比如双线性群，一种由非常特殊的椭圆曲线构成，有更多的代数结构可以证明CDH与IDH的等价性）
- 如果不想使用椭圆曲线群，可以开遍ElGamal系统，使其直接基于CDH证明CCA安全

3、Variants: twin ElGamal

对ElGamal做一个简单的变化

首先还是从G中选择生成元 g , 然后在 Z_N 中选择两个指数 (a_1, a_2) 作为私钥 sk , 公钥 $pk=(g, h_1=g^{a_1}, h_2=g^{a_2})$, 然后加解密和原来的ElGamal一样, 只不过是用了参数更多的公钥和私钥而已

E(pk=(g,h₁,h₂), m) :

$$b \leftarrow Z_n$$

$$k \leftarrow H(g^b, h_1^b, h_2^b)$$

$$c \leftarrow E_s(k, m)$$

output (g^b, c)

D(sk=(a₁,a₂), (u,c)) :

$$k \leftarrow H(u, u^{a_1}, u^{a_2})$$

$$m \leftarrow D_s(k, c)$$

output m

Dan Boneh

4、Chosen ciphertext security

上述简单的小改动可以让我们直接基于CDH证明ElGamal的CCA安全

首先还是需要假设对称加密系统提供认证加密, H 为理想hash函数, 若CDH假设在群G上成立, 则twin ElGamal为CCA安全的

但是带来了额外的开销, 为了严格的基于CDH假设证明CCA安全, 需要在加解密步骤中额外进行一次幂运算, 问题在于这么做值得吗 (哲学问题: 加解密需要更多的开销, 但是使得CCA安全可以基于一种更自然的假设)

是否有一些群, 使得CDH假设成立而IDH假设不成立? 如果有那就太好了, 这些群显然对于上述小改动是值得的, 因为CDH成立但IDH不成立, 因此twin ElGamal安全而普通ElGamal不安全

但是很可惜, 不知道是否有这种群, 就目前而言, 任何群中若CDH假设成立, 则IDH也成立, 因此还是这个哲学问题, 额外的开销到底值不值

但是这仍然是一个巧妙的改动, 可以直接使用CDH假设实现CCA安全而无需将ElGamal改动的太多

5、ElGamal security w/o random oracles?

是否可以回避hash函数是理想的这个假设? 即是否可以无需理想hash函数来证明CCA安全?

对于ElGamal而言, 有两个选择:

- 在双线性群中使用HDH假设 (扩展阅读: Special elliptic curve with more structure [CHK'04+BB'04])
- 在任何群中使用DDH假设 (Decision-DH, 决策DH假设) (扩展阅读: CS'98)

6、Further reading

- The Decision Diffie-Hellman problem. D. Boneh, ANTS 3, 1998
- Universal hash proofs and a paradigm for chosen ciphertext secure public key encryption. R. Cramer and V. Shoup, Eurocrypt 2002
- Chosen-ciphertext security from Identity-Based Encryption. D. Boneh, R. Canetti, S. Halevi, and J. Katz, SICOMP 2007
- The Twin Diffie-Hellman problem and applications. D. Cash, E. Kiltz, V. Shoup, Eurocrypt 2008
- Efficient chosen-ciphertext security via extractable hash proofs. H. Wee, Crypto 2010

1. 教授本人的论文，讨论了与DH有关的各种假设，特别研究了决策DH假设
2. 展示了如何从DH和类似的假设中构建CCA安全的系统
3. 从双线性群中建立CCA安全（该论文提出了一种基于身份的加密）
4. Twin DH结构及其证明
5. 提供了构建通用的CCA安全的系统框架（使用可提取hash证明）

W6 12-4 A Unifying Theme

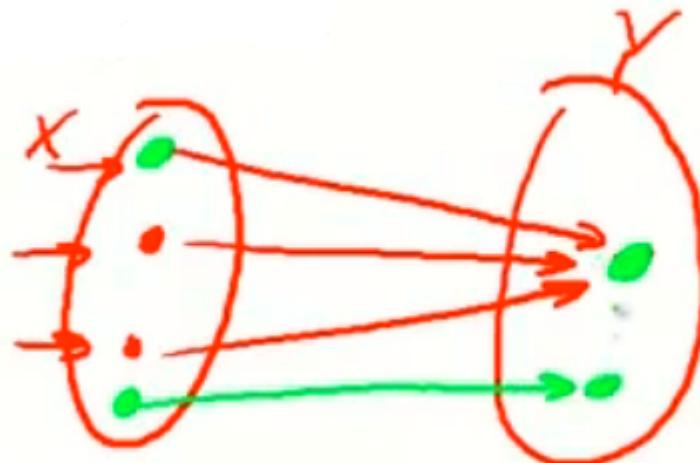
本周学习了RSA和ElGamal两种公钥系统，前者基于陷门函数，后者基于D-H协议，其实这两者都遵循一个统一原则，即单向函数

1、One-way functions (informal)

单向函数 $f: X \rightarrow Y$ ，正向计算非常简单，但是计算的逆函数很困难（对于 X 到 Y 的映射，给出 $x \in X$ 找到 Y 中的映射 y 很简单，但是给出 $y \in Y$ ，找到 X 中的原像很难），即有如下不等式

$$\Pr[f(A(f(x))) = f(x)] < negligible$$

上述不等式表明，对于所有有效的算法 A ，如果将函数 f 重新应用于 A 的输出，其得到原来的点的几率应当是可忽略的



单向函数的存在：可以归约到P=NP问题

2、Ex.1: generic one-way functions

接下来来看一个例子（一个假定的单向函数），由伪随机数生成器构造而来

记 $f:X \rightarrow Y$ 为一安全PRG ($|Y| >> |X|$)

引理：若 f 为一安全PRG，则 f 为一单向函数

证明：里用反证法，假设 f 不是单向的，假设有一个计算 f 的逆的有效算法A，则可以构造一个破解PRG的算法B，具体如下

Proof sketch:

$$A \text{ inverts } f \Rightarrow B(y) = \begin{cases} 0 & \text{if } f(A(y)) = y \\ 1 & \text{otherwise} \end{cases} \text{ is a distinguisher}$$

给定 $y \in Y$ ，并将 y 输入算法A，然后计算 $f(A(y))$ ，如果其最终输出了 y 的种子 y ，则算法B可以以不可忽略的概率输出0，否则输出1（输出1意味着B得到的是一个真随机串，难以找到一个种子是的PRG生成真随机串）

若给定某个输出 y ， y 部署于PRG的输出集合，则没有使得生成器映射到这个 y 的种子，因此若给定 Y 中某个真随机的点，B会以高概率输出1

但若给定的是PRG的输出，则A会输出其对应的种子，然后再由该种子得到这个输出，从而B会输出0

上述分析可知，若A可以计算 f 的逆，则B可以破坏PRG，又由于假设PRG为安全的，由反证法可得，A并不可以计算 f 的逆，因此 f 为单向函数，得证

事实上，利用PRG可以直接构造出单向函数，但是这类单向函数没有什么特别的，这意味着很难在公钥加密系统的密钥交换中使用这类函数

3、Ex.2: The DLOG one-way function

看第二个例子，定义N阶循环群 G ，生成元为 g ，定义函数 $f: f(x) = g^x$ ，即将 Z_N （ $0 \sim N-1$ 中元素构成的集合）映射到 G ，该问题是个离散对数问题

引理：如果离散对数问题是困难的，则 f 为单向函数

一些有趣的性质：若已知 $f(x)$ 和 $f(y)$ ，则计算 $f(x+y)$ 很简单（加法性质），由于在循环群 G 内计算，从而使之是个陷门函数，这也使得密钥交换和公钥加密可行

4、Ex.3: The RSA one-way function

看第三个例子，RSA中选择两个大素数 p 和 q ，记 $N = pq$ ，然后选择 e ， d 使得 $ed \equiv 1 \pmod{\phi(N)}$

定义函数 $f: f(x) = x^e$ ，为 Z_N^* 到 Z_N^* 的映射

引理： f 在RSA假设下为单向函数

性质： $f(x)f(y) = f(xy)$ （乘法性质）

陷门性：存在一个密钥，使得计算函数的逆非常简单，若没有这个密钥，则该函数是单向的

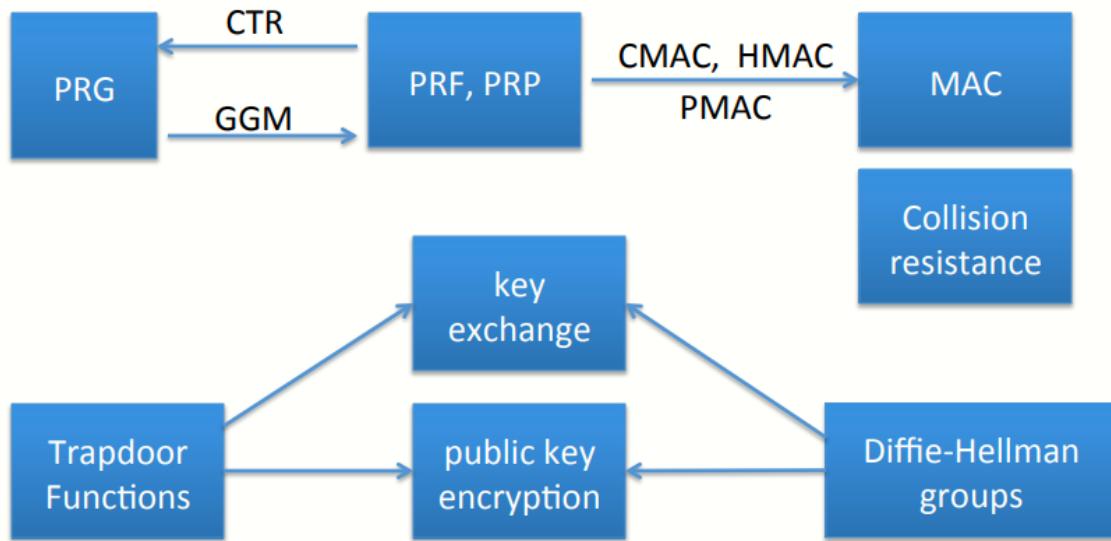
就目前而言，陷门性仍然是公钥加密算法的蜘蛛，且由于该函数的有陷门，使得RSA非常适合用于构造数字签名

5、Summary

基于一些有特殊性质的单向函数，可以构造密钥交换、公钥加密等等算法

W6 12-5 Farewell

1、Quick Review: primitives



第一周讨论了伪随机数生成器和流加密算法

第二周讨论块加密算法并将块加密视为伪随机置换和伪随机函数（利用CTR模式可以将块加密转换成PRG，利用GGM构造可以从PRG构造块加密算法）

第三周讨论了数据完整性，特别探讨了MAC及其各种形式的构造（如利用了PRG的CMA、HMAC、PMAC等），还讨论了抗碰撞性

第四周讨论了将完整性和保密性结合起来，特别讨论了如何将加密和MAC结合起来，从而构造认证加密（Authentic Encryption），但本质上仅能够放窃听攻击的加密算法并不是总体安全的，还需要提供防篡改功能，因此对称加密系统中需要时用包含认证加密的算法

第五周讨论了陷门函数和D-H协议（注意到密钥交换协议仅能防止窃听）

第六周讨论了利用陷门函数和D-H协议构造公钥加密算法

2、Remaining Core Topics (part II)

之后还会讨论数字签名和认证数据的内容，还有口令管理、挑战-响应协议、隐私机制（使得用户可以对自身身份进行验证同时又不会暴露身份，或者登陆的时候不暴露自己）、零知识证明（密码学中广泛使用的工具）

3、Many more topics to cover

之后会开高级密码学课程，包含椭圆曲线加密、量子计算、密钥管理范式、匿名数字现金、私人投票与拍卖系统、密文计算、全同态加密、格密码、双方/多方计算等知识点

感兴趣可以先学

| 最后再提醒一句：密码学是非常强大的工具，弱国使用不当反而会导致系统更容易被攻击

