life.augmented

STM32 生态系统 第十一期

信息安全 . Information Security

02. 密码学基本原理（上）

2020.03

本期内容概览

1 对称加解密技术

2 非对称加解密技术

3 二者的结合：共享密钥的产生

4 常用算法

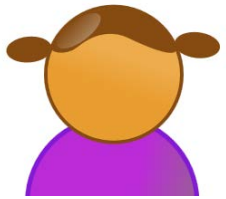
life.augmented

假设这样一个场景...

- Alice 喜欢 Bob，她想给他发一封信

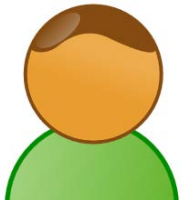
假设这样一个场景...

- Alice 喜欢 Bob，她想给他发一封信



Alice

我喜欢你.
Alice



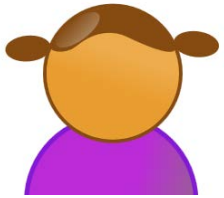
Bob

- Eve 想搞破坏，拦截了这封信，篡改了它，再发给 Bob

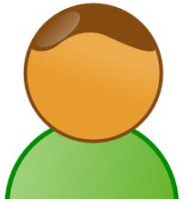
Eve

我不喜欢你.
Alice

我们希望采取什么样的措施？



Alice



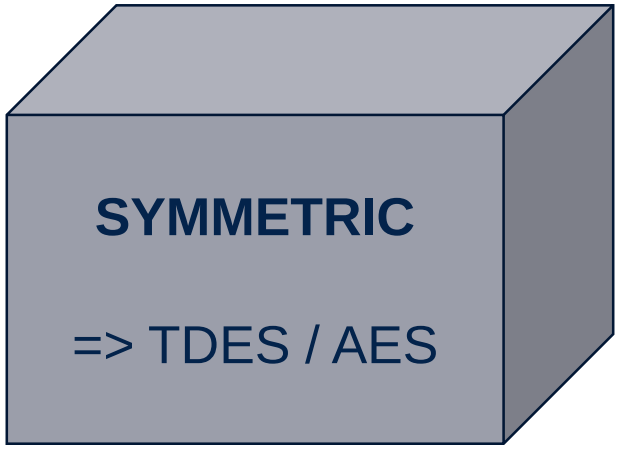
Bob

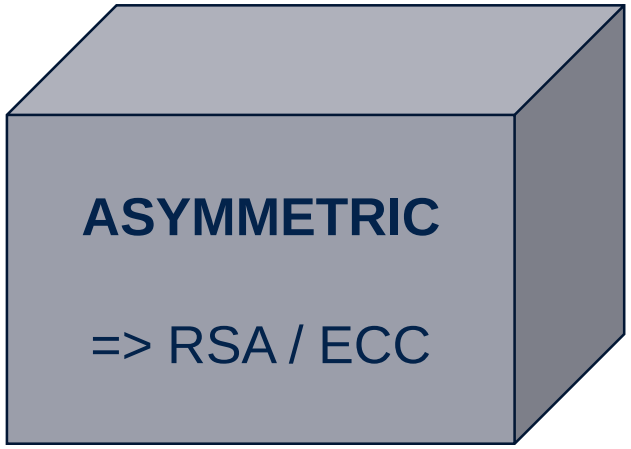
- 交换机密/confidential消息：除了目标接收方Bob，其他人都看不到消息的内容
- 保证消息的完整性/message integrity：消息被 Eve给篡改后Bob能识别出来
- (身份)认证/Authentication：Bob 要确定收到的消息确实来自 Alice

来看看密码学怎么帮忙解决

对消息进行加密

- **通**过加密、解密机制，可以保证交换数据的机密性 /confidential
- **两大**类加解密机制

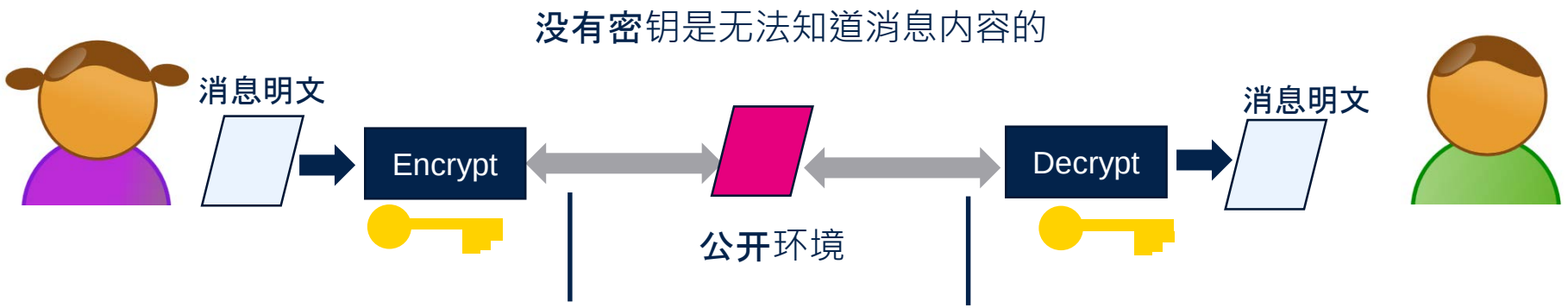




- **加解密都需要密钥**

对称加解密技术

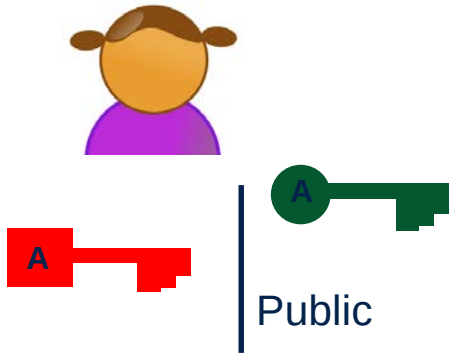
- Alice和 Bob事先约好了一个密钥
- 该密钥可以用来进行加密消息，也可以用来解密消息

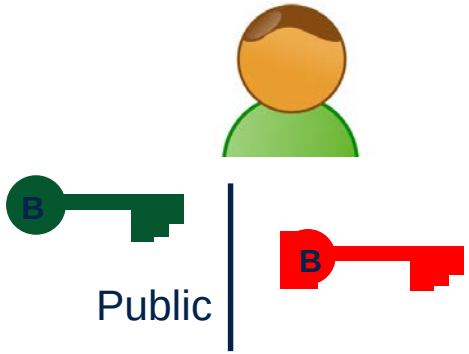


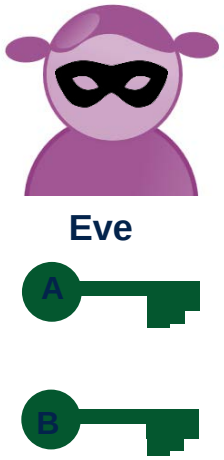
- 对称密钥系统的优势：简单、快速
- 对称密钥系统的缺点：通信双方必须事先约好这个密钥；管理不方便

非对称加解密技术

- Alice和 Bob事先自己产生自己的公钥-私钥对
  - 私钥自己保存，公钥可以分享给其他人，不限于通信的对方







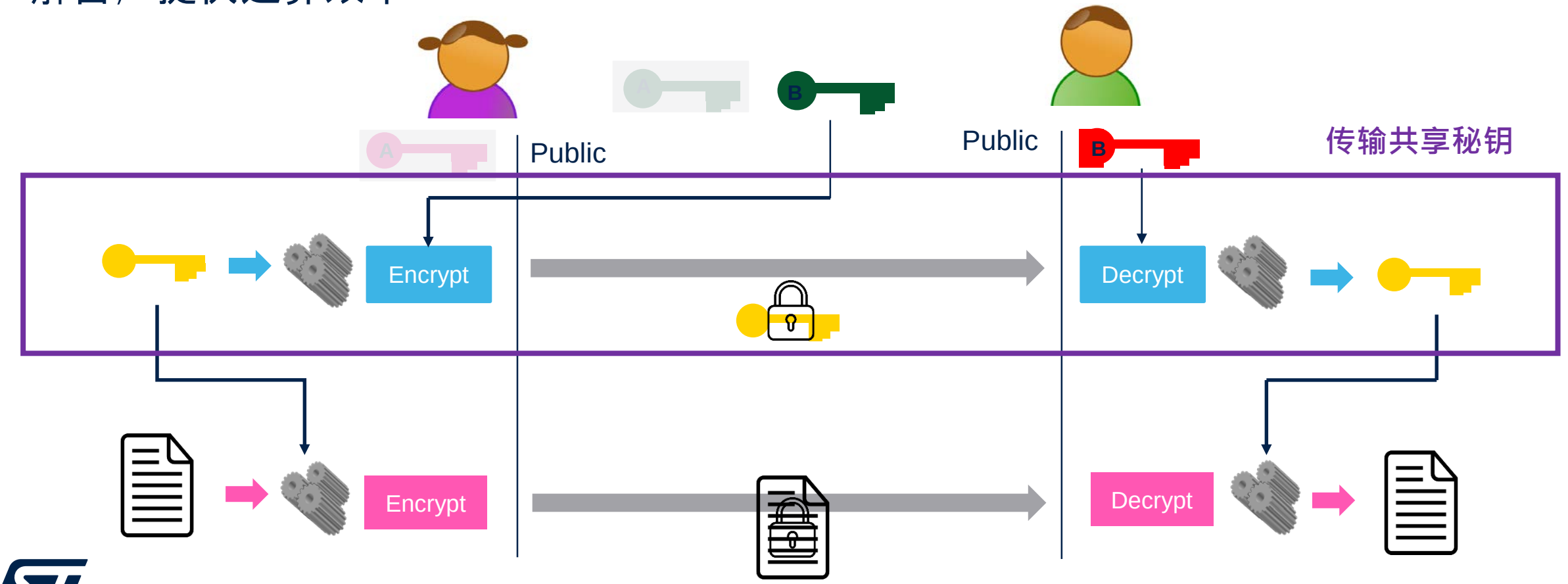
- 加密、解密使用不同的密钥
  - 用公钥“处理/加密” 的消息，只能由与其对应的私钥来“恢复/解密”
  - 用私钥“处理/签名”的消息，只能由与其对应的公钥来“恢复/验签”

- 非对称密钥系统的优势：密钥方便管理和分发

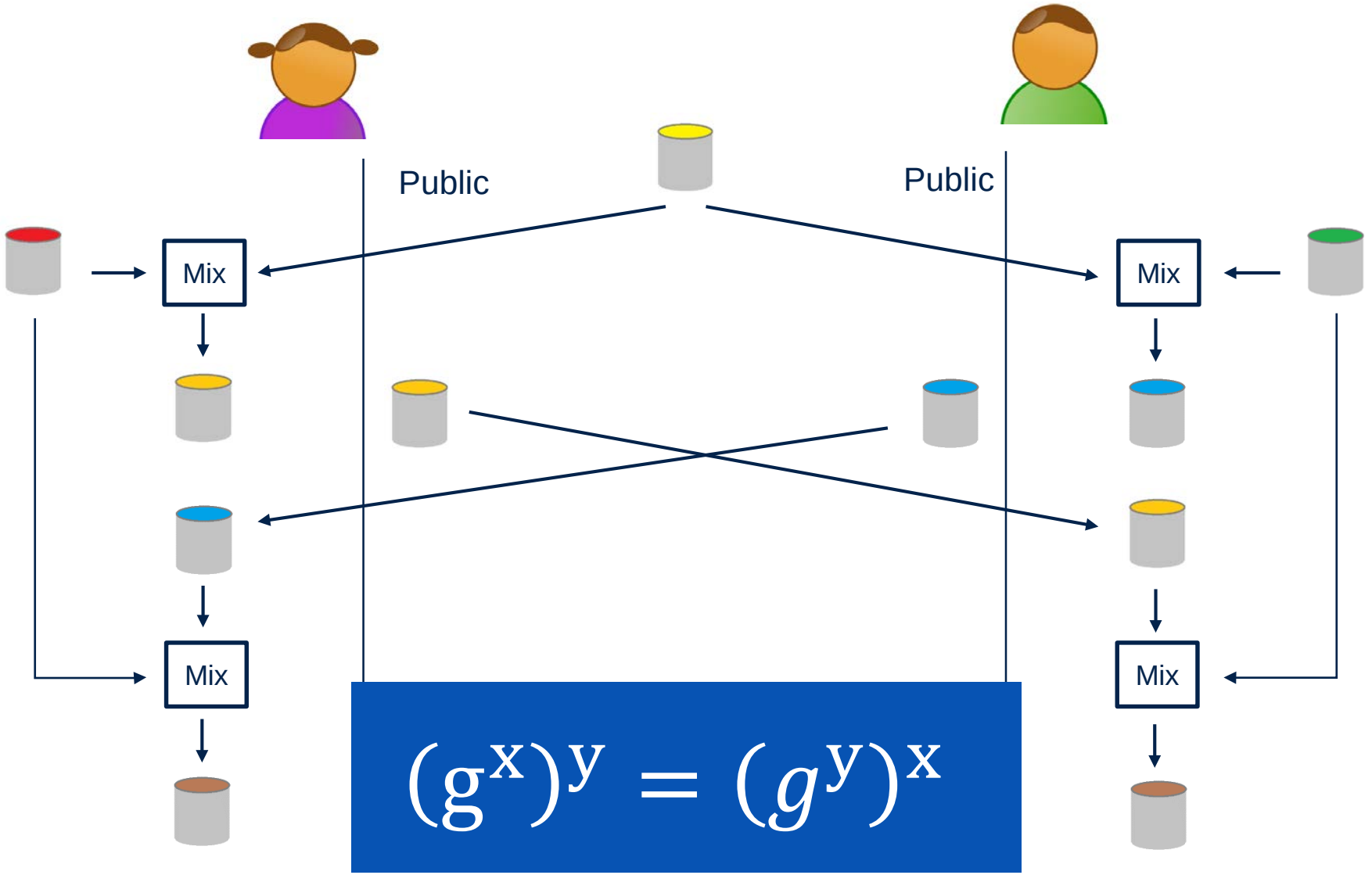
- 非对称密钥系统的缺点：实现复杂，运算效率不如对称密钥系统

二者的结合，各取优点

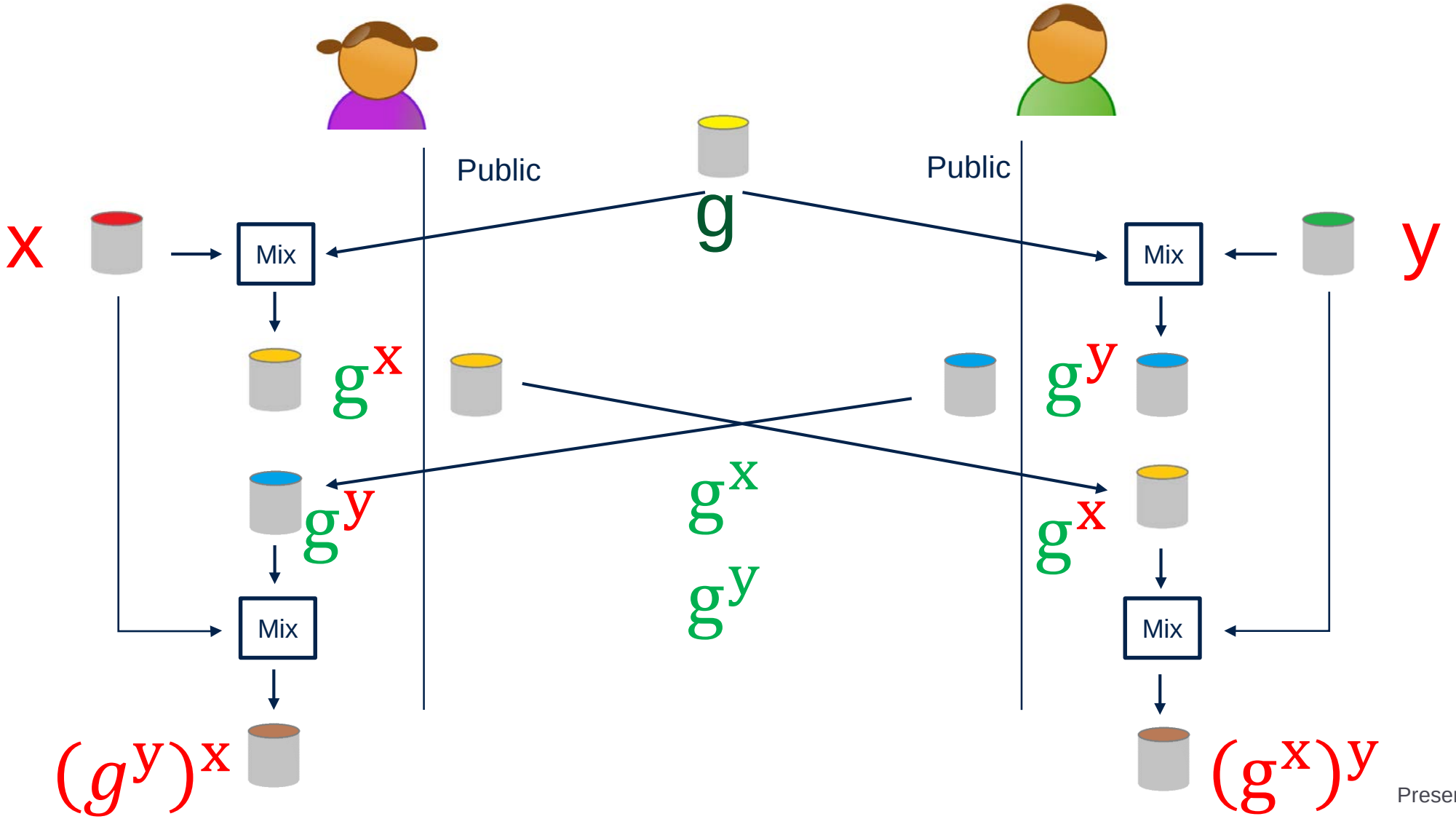
- **使用非**对称密钥技术发送对称密钥；再使用对称密钥对大数据量的交互消息进行加解密，**提供运算效率**



生成共享密钥



生成共享密钥



常用对称加密算法及模式

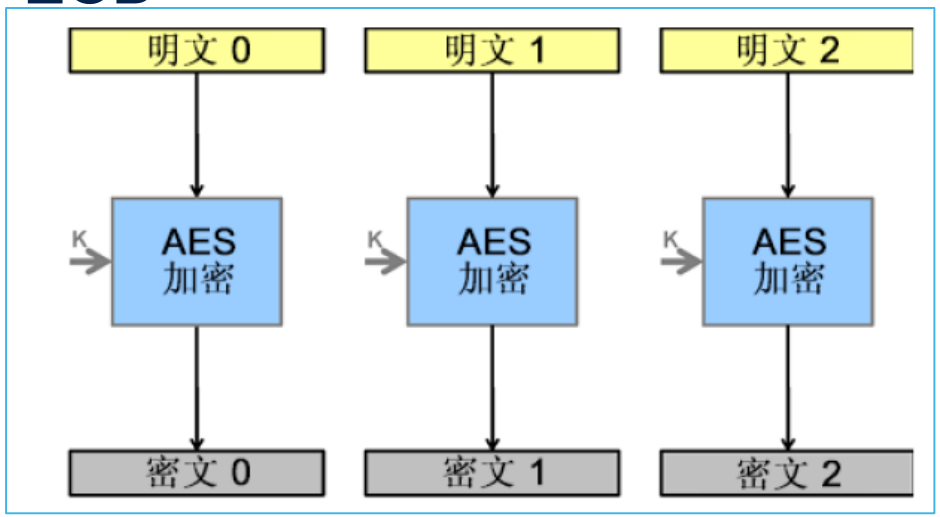
- 对称加解密技术的特点：基于排列，组合，位移，异或等操作，因此软硬件实现起**来都快**

算法	明文块大小	密钥大小	说明
TDES	64位	3*56 = 168 位	**分**别使用**3个不同密**钥，进行三次**DES运算**
		2*56 = 112 位	**两个密**钥相同
AES	128位	10轮变形转换 → 128位	**比TDES运行更快，占用更少存**储资源
		12轮变形转换 → 192位	
		14轮变形转换 → 256位	

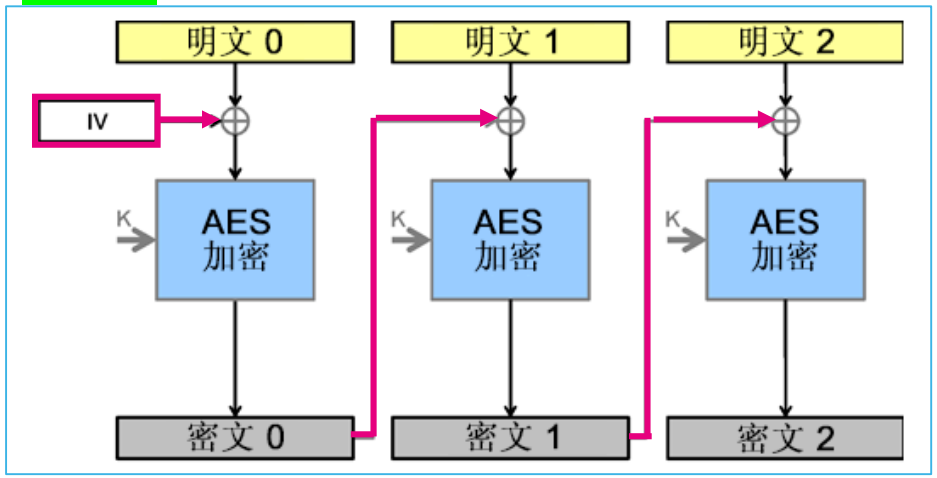
AES 模式的变迁	
AES-ECB/电子密码本	**独立**块加密，密文会保留块的pattern
AES-CBC/CFB/OFB @ chain mode	**当前加密操作的**输入，和前一个块有关联；初始向量/IV**的引入**
AES-CTR/CCM/GCM @ counter mode	对每个块的操作，引入随机数和计数器

AES常用加解密模式

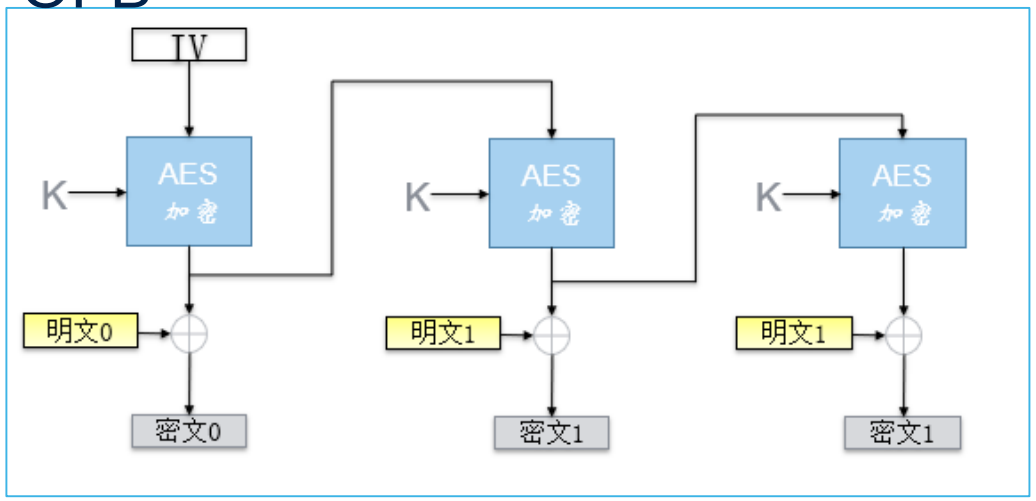
ECB



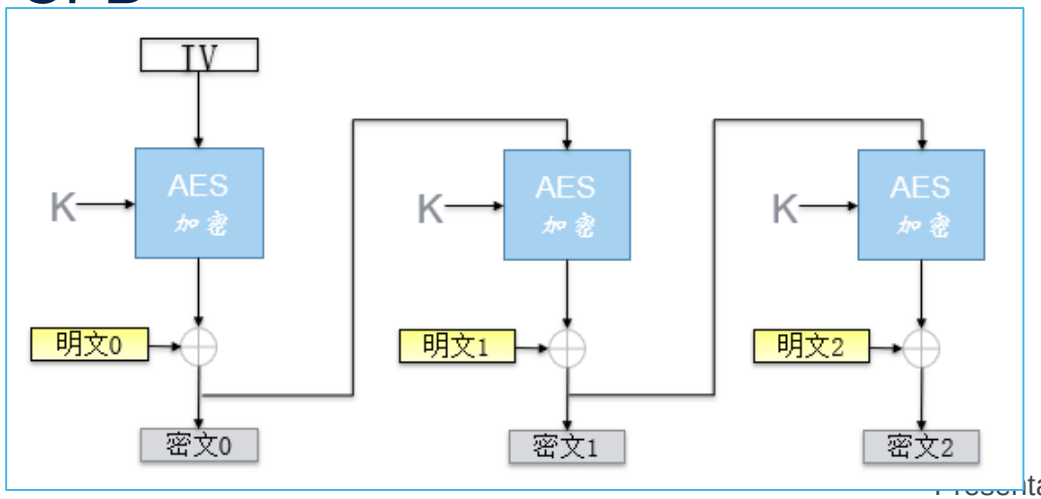
CBC



OFB



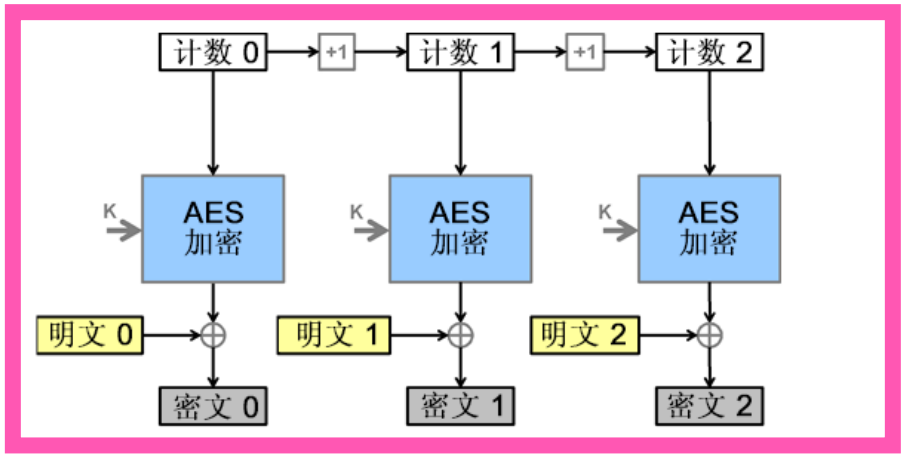
CFB



life.augmented

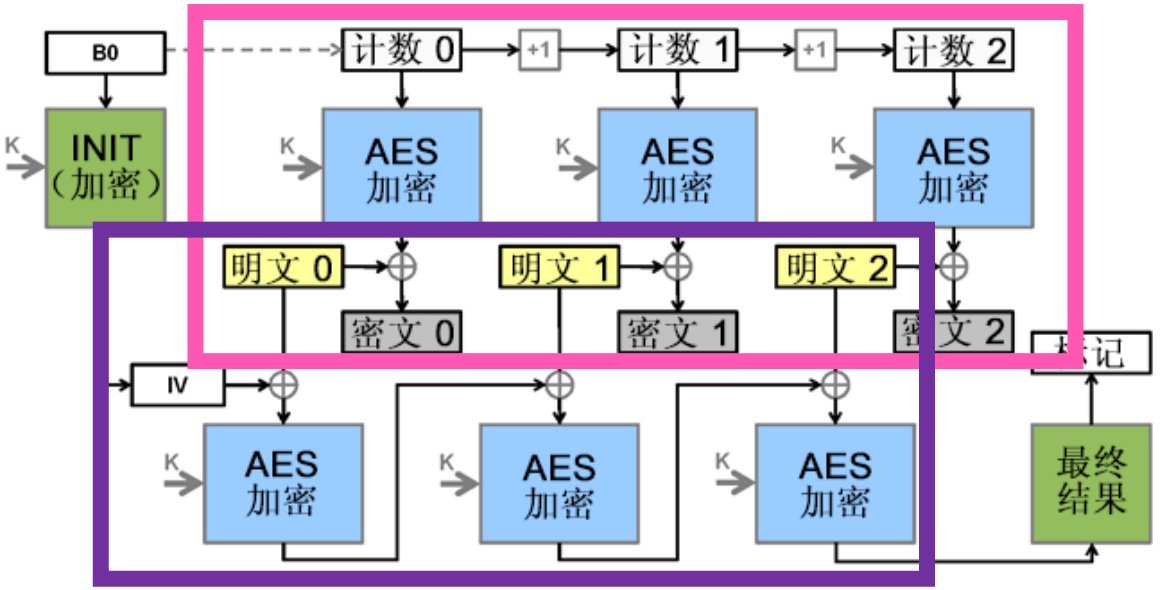
AES常用加解密模式

CTR



CCM

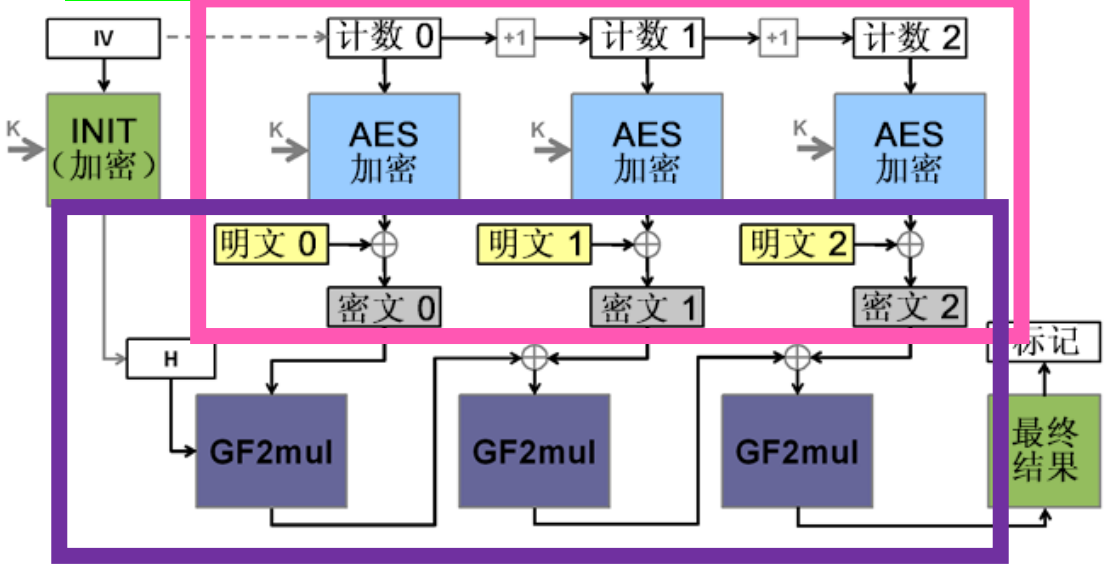
CTR加密



CBC计算Tag

GCM

CTR加密



Galois域乘法计算Tag → GMAC

常用非对称加密算法

- **基于大数的数学运算**，软硬件实现都较复杂，不适于大量数据的实时加解密

算法	明文块大小	密钥大小	基于的数学难题
RSA	和密钥大小相同	常用大小：2048 ~ 4096 位	已知两个素数，其乘积容易计算；反之难
			e.g，61 * 349 = 21289
ECC		常用大小：160 ~ 512 位	已知两个整数，做指数运算容易；反之，已知指数运算后的值，反推底数和对数，难
			e.g. $3^6 = 729$，$\log_x(729) = y$ ？

NIST guidelines for public key sizes for AES

ECC KEY SIZE (Bits)	RSA KEY SIZE (Bits)	KEY SIZE RATIO	AES KEY SIZE (Bits)
163	1024	1 : 6	
256	3072	1 : 12	128
384	7680	1 : 20	192
512	15 360	1 : 30	256

Supplied by NIST to ANSI X9F1

- 对于同一等级的安全水平
- ECC密钥长度更小；签名更快
- RSA验签更快

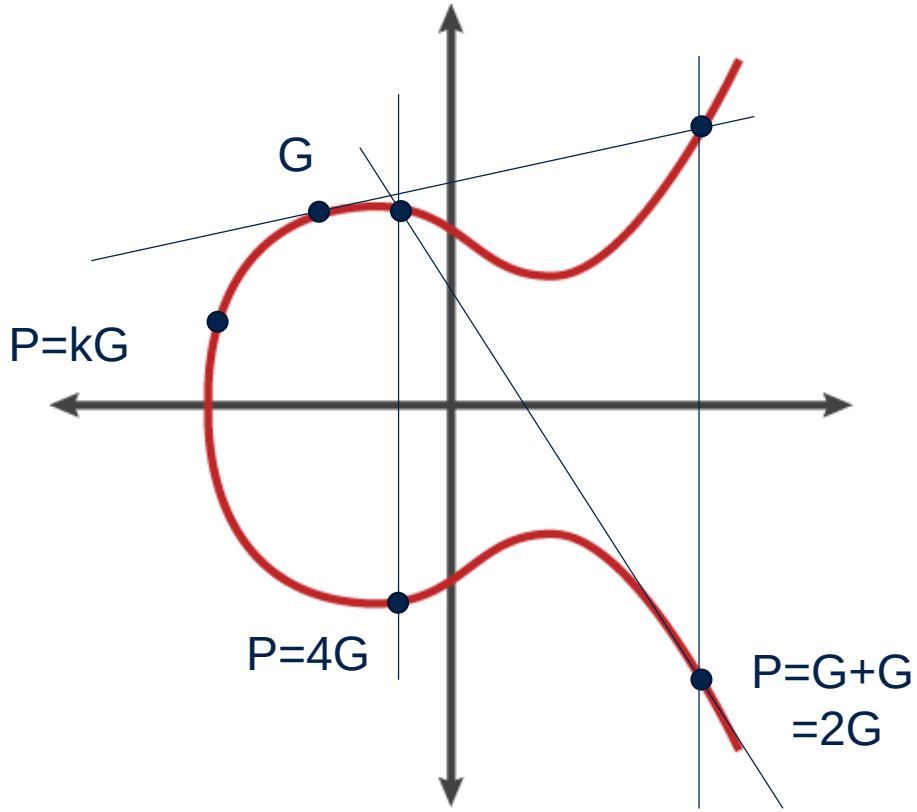
life.augmented

RSA：密钥生成、格式

理论步骤	举例
选择两个素数：P、Q	P = 5, Q = 11
计算 N = P*Q	**N = 55**，**模的长度作为密钥的长度**
计算 Ø = (P-1)*(Q-1)	Ø = 40
选择任意的E满足1<E< Ø，且gcd(E, Ø)=1	**E = 7**，满足7和40的最大公约数是1
计算D 满足 1<D< Ø , 且 E*D ≡ 1 (mod Ø)	**D = 23**，满足 7*23 = 161，对40取模，为1
返回(N, E, D)	E，公钥指数；N，模；一起构成**公钥** D，私钥指数；N，模，一起构成**私钥** 两个素数P和Q，用于产生以上公钥和私钥，但是不参与后面的加解密运算；仍需要保密
对数字M加密/ 使用公钥：M的**E**次方，再对**N**取模	
对数字C解密/ 使用私钥：C的**D**次方，再对**N**取模	

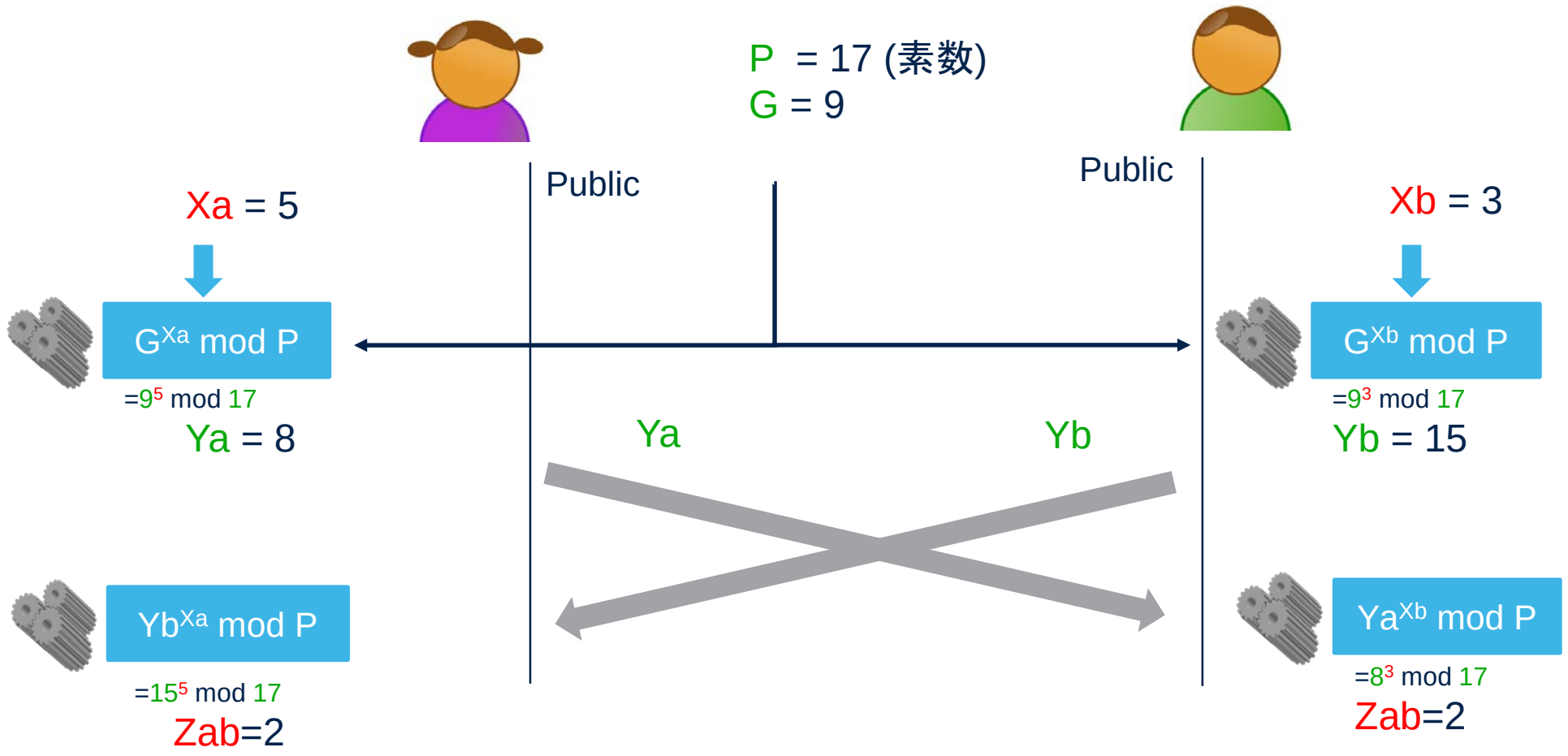
Ecc：密钥的生成

- **有限域上的**n阶椭圆曲线：y2 =x3 +ax +b (mod p)
- **参考点**G
- 选择随机数k，满足k Є [1, n-1]，作为私钥
- **曲**线的两个点G、P，作为公钥

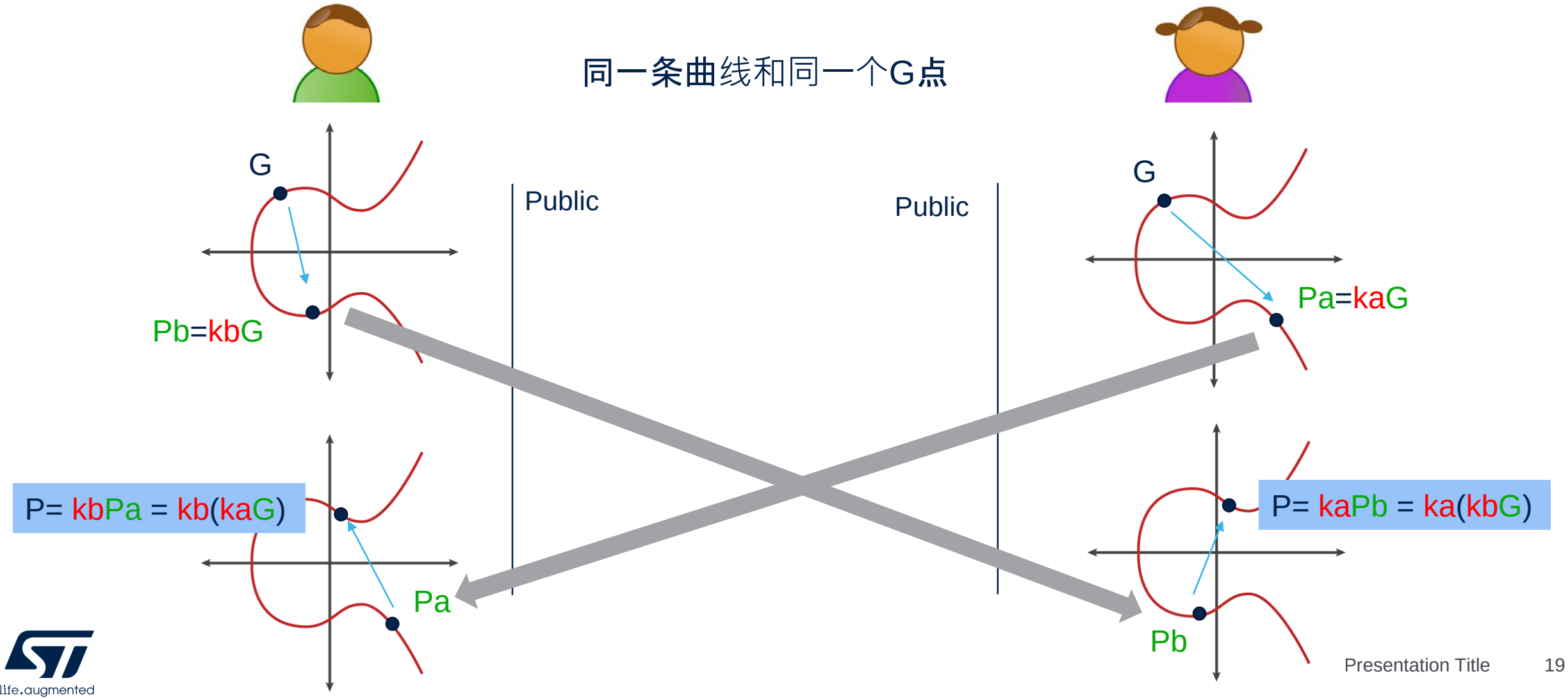


DH算法/diffie-hellman，用于密钥协商

- 计算离散对数问题： GXa*Xb mod P = GXb*Xa mod P



ECDH = ECC + DH


同一条曲线和同一个G点
G
Public
Public
G
Pb=kbG
Pa=kaG
P= kbPa = kb(kaG)
P= kaPb = ka(kbG)
Pa
Pb

本期回顾

- **基本原理**

  对称加解密

  非对称加解密

  二者的结合，产生功效密钥

- **常用算法**

  对称加解密：TDES、AES

  非对称加解密：RSA、椭圆曲线(ECC)、Diffie-Hellman

谢　谢

life.augmented