



北京邮电大学

Beijing University of Posts and Telecommunications

大数据安全与隐私保护

石瑞生

网络空间安全学院

- 同态加密的概念
 - 为什么需要同态加密技术？ 什么是同态加密？
 - 技术目标和技术效果
- 工作原理
 - 具有乘法同态性质的加密算法
 - 具有加法同态性质的加密算法：Paillier同态加密算法
 - 全同态加密算法
- 应用与挑战：性能
- 实用性技术方案
 - CryptDB：来自MIT的开源技术，可对加密数据进行SQL查询

为什么需要同态加密技术？

- 当我们使用云盘的时候，特别是我们把敏感数据保存在云盘上时，大家心里可能都会一丝疑虑：安全吗？

为什么需要同态加密技术？

- 当我们使用云盘的时候，特别是我们把敏感数据保存在云盘上时，大家心里可能都会一丝疑虑：安全吗？
- 有人选择把数据加密后上传云盘，但随之而来的问题是，对加密数据的搜索、管理很不方便，严重影响用户的工作效率，用户体验明显不佳。

有没有令人满意的技术方案呢？

同态加密技术就是满足这种需求的重要候选方案。

为什么需要同态加密技术？

- 有了同态加密，有预谋的盗取敏感数据的情况将成为历史。
 - 因为在同态加密环境下，敏感数据**总是**处于加密状态，而这些加密数据对盗贼来说是没用的。
- 那么，什么是同态加密呢？我们先来看一个典型的同态加密应用场景。
 - 考虑到报税员，或者一些财务服务机构，如mint.com：你将个人财务信息提供给他们，他们通过计算来优化你的财务/税务策略。
 - 但是，你真的会将自己的银行账号和个人财务信息提交到财务优化网站上么？

为什么需要同态加密技术？

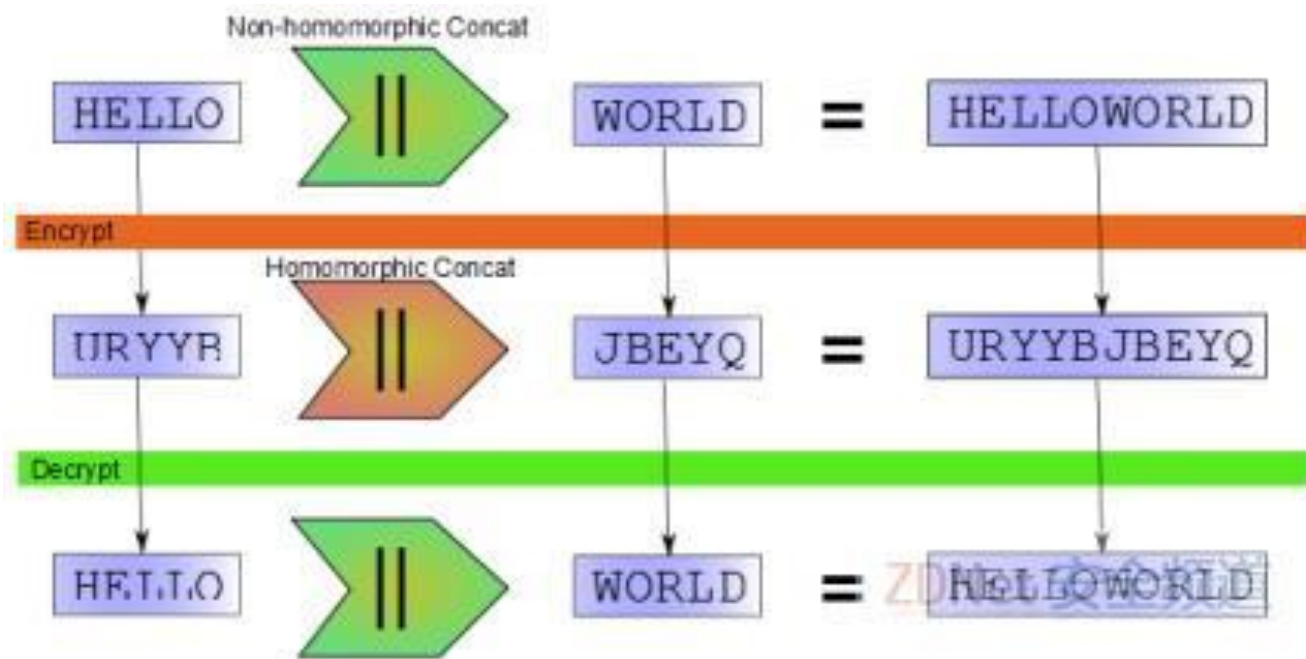
- 有了同态加密，有预谋的盗取敏感数据的情况将成为历史。
 - 因为在同态加密环境下，敏感数据**总是**处于加密状态，而这些加密数据对盗贼来说是没用的。
 - 那么，什么是同态加密呢？我们先来看一个典型的同态加密应用场景。
 - 考虑到报税员，或者一些财务服务机构，如mint.com：你将个人财务信息提供给他们，他们通过计算来优化你的财务/税务策略。
 - 但是，你真的会将自己的银行账号和个人财务信息提交到财务优化网站上么？
- 现在换一种情况，你所提交的是一个代码，财务优化网站凭此代码可以从银行数据库下载同态加密过的你的财务数据。
 - 他们可以直接对加密数据进行计算，将所得到的税务优化结果再以加密的形式发送给你，**这些加密的数据他们也无法破解**，但是你可以。

示例



- 让我们看看同态加密是如何处理 $2+3$ 这样的问题的
 - 假设数据已经在本地被加密了, 2加密后变为22, 3加密后变为33。
 - 加密后的数据被发送到服务器, 在进行相加运算。然后服务器将加密后的结果55发送回来。
 - 然后本地解密为5。

- 字符串处理

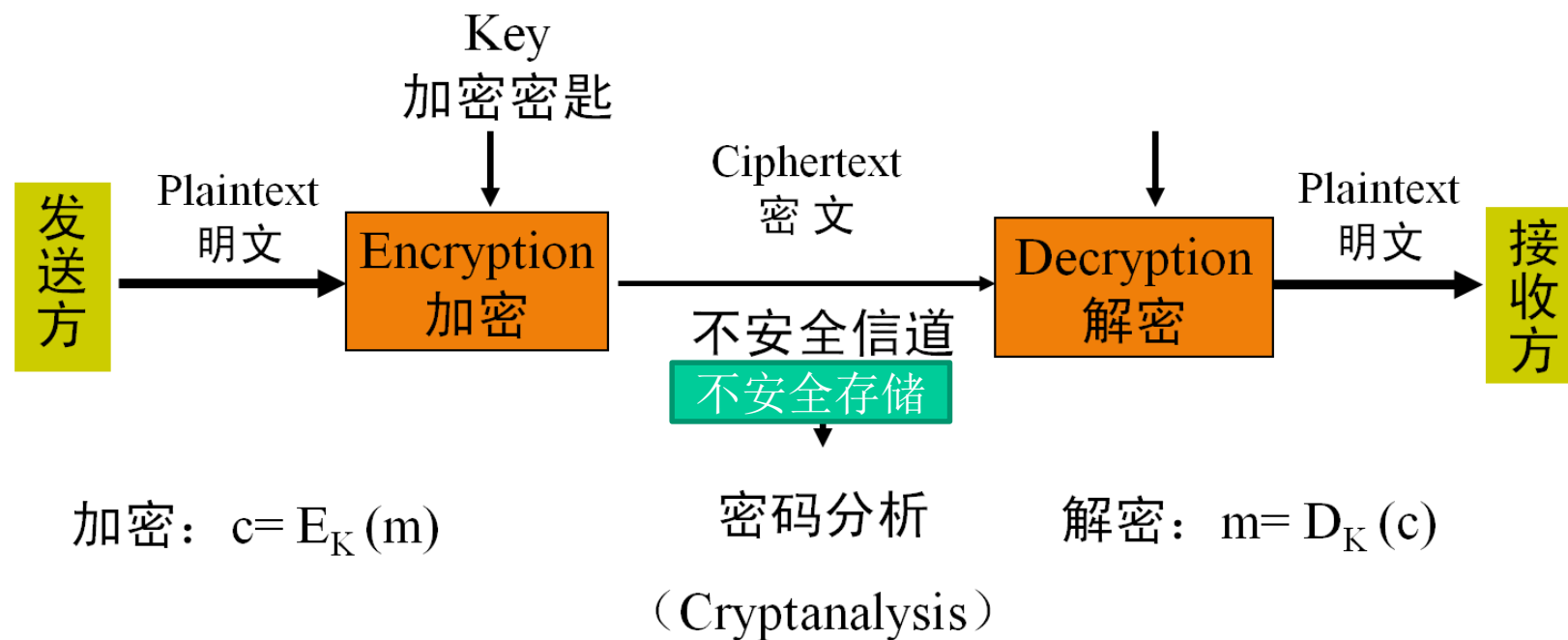


为什么需要同态加密技术？

- 利用同态加密算法，云端服务器不用解密就可以处理敏感数据。
 - 可以解决云计算混乱的数据安全现状，而且**不会在云环境中出现任何明文数据**。
- 我们可以对输入的加密内容进行某种运算，生成新的加密数据，而当解密的时候，所解密出的明文内容，与我们输入明文内容进行运算后得到的结果是一致的。

回顾传统的加密技术

- 加密的目的是保护数据的机密性。加密分为对称加密和非对称加密。
 - 对称加密是指加密和解密用的同一个密钥；
 - 而非对称加密在加密时用的是公钥，解密时用的是私钥。
 - 非对称加密体制是基于数学难问题（比如大整数分解、离散对数），加密解密操作比对称加密要慢很多。



以往加密方案的一个缺点

- 如果数据只是简单地被存储起来，那么传统的加密技术就可以非常完美地工作。
 - 而当你真的需要处理和分析被存储起来的数据时，问题就出现了。
 - 数据在加密之后，**如果要想对数据进行运算，就必须先解密**，这样增加了数据的不安全因素。
- 这也就是为什么现在需要一个实用性的处理加密数据的系统。
 - 希望该系统**可以在不先对加密数据解密时而执行对加密数据的计算**。

- 新的应用场景

- 如果对加密数据（即密文）的操作是在不可信设备（untrusted device）上进行的，我们希望这些设备并不知道数据的真实值（即明文），只发回给我们对密文操作后的结果，并且我们可以解密这些操作后的结果。

- 举一个简单的例子

- n 个学生和1个老师通信，每个学生都有1个数据要发给老师，老师需要知道这 n 个数据之和，而学生们不想让老师知道每个数据的真实值。同时，所有数据（ n 个数及它们的和）不希望泄漏给第三方。

- 为了解决这个问题，Rivest等在1978年提出了同态加密的思想。
 - Rivest、Adleman、Dertouzos (1978年) 介绍了同态性质的密码应用：无信任的委托计算。
 - 户主如果需要明文的若干计算（相加，相乘，平均值等），但又懒得自己计算，可以（1）委托服务器进行对应的密文计算；（2）服务器将密文计算结果发给户主；（3）户主将密文计算结果进行解密，就得到所需要的明文计算结果。
 - 注意到服务器在进行密文计算的时候，并不告诉服务器密文所对应的明文。
- 同态性质本来被看做安全性缺陷

计算加密数据的想法首先在1978年出现，当时Ronald Rivest，Len Adleman和Michael Dertouzos写了一篇题为“数据银行和隐私同态”（On Data Banks and Privacy Homomorphisms）的开创性文章。

在这篇文章中，他们提出了在使用数据进行计算时保持数据加密的想法。他们称能够支持这种计算的加密方案是“同态的（homomorphic）”。当时，他们不知道如何进行一种能够进行各种计算的加密，而且也没有其他人知道——但是他们和其他计算机科学家都渴望找到一种方式去实现它。

同态加密研究的历程

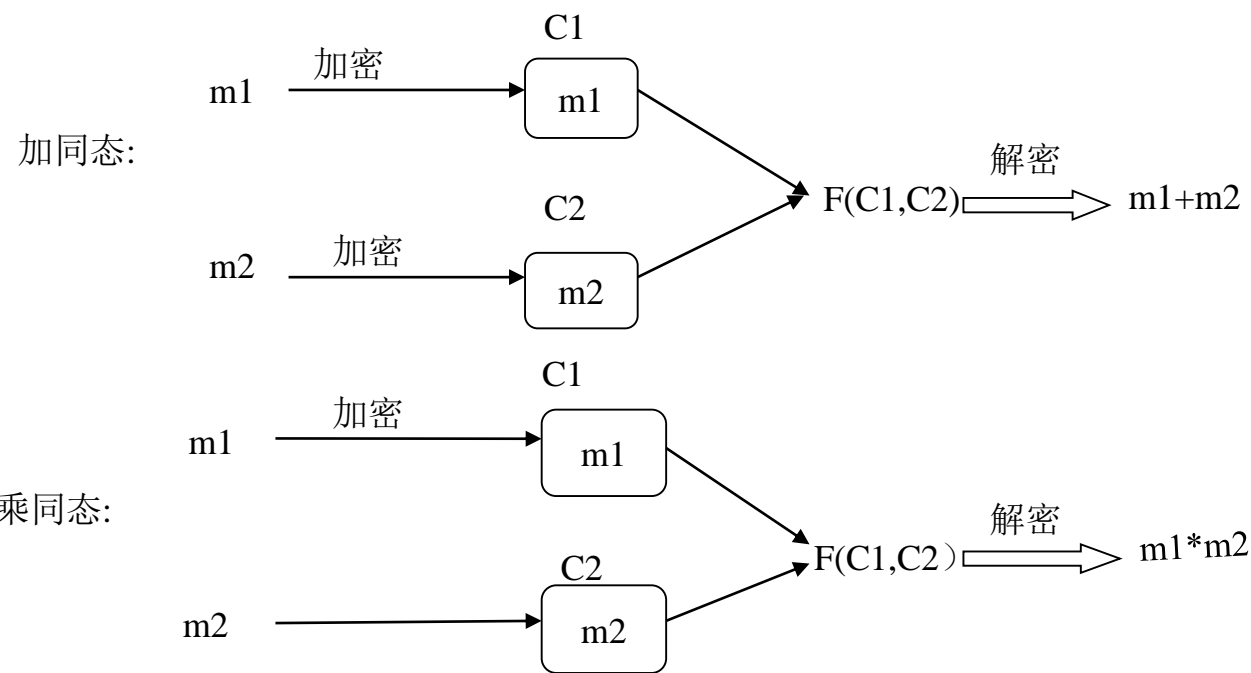
1999年，Pascal Paillier开发了一个具有奇妙特性的加密系统：如果你有一组加密后的数字，您将获得原始数据总和的加密版本。但是，Paillier的加密算法只能支持加法运算，例如，可以非常快速地计算加密值的总和。而现实中，仅仅是加法运算显然是不够的。

直到2009年，克雷格·金特里（Craig Gentry）取得了重大突破：他想到了一种允许计算机在加密之后计算数据的任何运算的加密模式。这种方案被称为全同态加密。从数学角度来说，Gentry解决方案是非常棒的。之后，其他研究人员继续努力，主要是为了提高Gentry的加密方法的性能和安全性。尽管人们取得了一些进展，但性能仍然是一个巨大的问题：最完整的同态加密方案所需时间比相应的未加密的计算长数十亿倍。如果一个网站通常用一秒钟就能计算出来的结果，使用同步加密则需要12天！

- 全同态加密的技术被冠以“密码学的圣杯”的称号

同态加密研究的历程

- 1977年, RSA算法原型可以实现乘法的同态; RSA同态性质被看做安全性缺陷。
- 1999年Pascal Paillier论文实现了加法同态
 - 参考论文: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes
- 2009年IBM 研究员 Craig Gentry 找到了一种全同态加密算法
 - 全同态加密的技术被冠以“密码学的圣杯”的称号。
 - 参考论文 Fully homomorphic encryption using ideal lattices



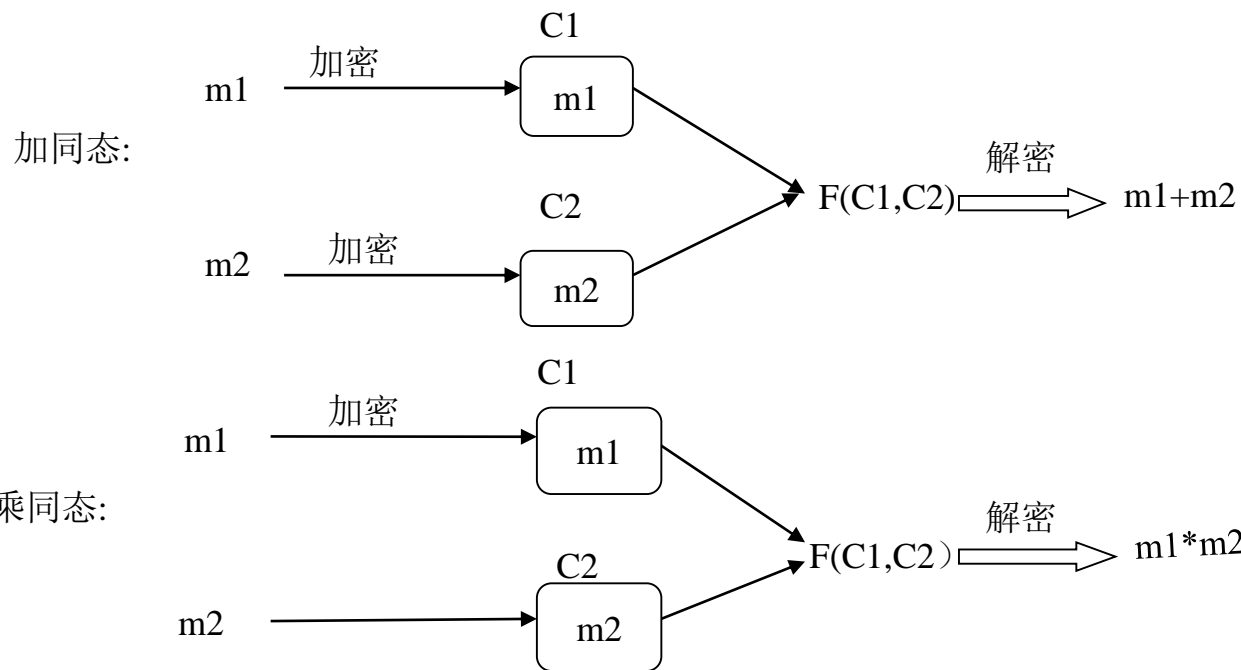
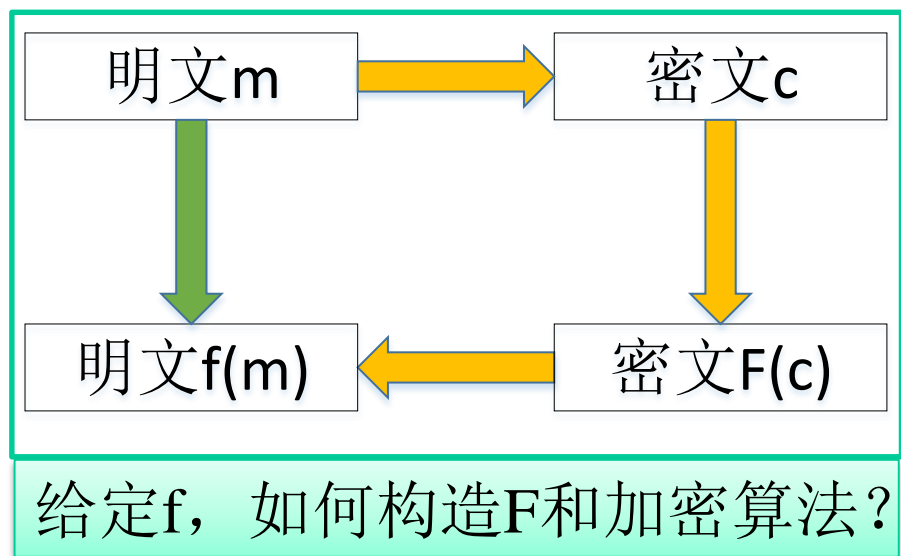
RSA算法对于乘法操作是同态的, 对应的操作F也是乘法, 对别的操作 (比如加法) 就无法构造出对应的F。

Paillier算法则是对加法同态的。

如果一种加密算法, 对于乘法和加法都能找到对应的操作, 就称其为全同态加密算法。

基本原理

- 记加密操作为 E ，明文为 m ，加密得 c ，即 $c=E(m)$ ， $m=D(c)$ 。已知针对明文有操作 f ，针对 E 可构造 F ，使得 $F(c)=E(f(m))$ ，这样 E 就是一个针对 f 的同态加密算法。
- 假设 f 是个很复杂的操作，有了同态加密，我们就可以把加密得到的 c 交给第三方，第三方进行操作 F ，我们拿回 $F(c)$ 后，一解密，就得到了 $f(m)$ 。
- 第三方替我们干了活，对 m 却仍一无所知，——多么融洽的关系啊。



RSA的同态性 - 乘法的同态

- 设 $n=pq$, (n,e) 为公钥, d 为私钥
- RSA的加密函数为

$$E(m)=c=m^e \pmod{n}$$

因此, 若有

$$c_1=E(m_1)=m_1^e \pmod{n} \text{ 和 } c_2=E(m_2)=m_2^e \pmod{n}$$

则有

$$c_1c_2=\mathbf{E(m_1)E(m_2)}=(m_1)^e(m_2)^e \pmod{n} =(m_1m_2)^e \pmod{n} =\mathbf{E(m_1m_2)} \text{ 即RSA满}$$

足乘法同态

$F(C_1,C_2)=C_1 * C_2;$	$f(m_1,m_2)=m_1 * m_2$
-------------------------	------------------------

RSA算法描述



密钥产生

选择 p, q

计算 $n = p \times q$

计算 $\phi(n) = (p-1)(q-1)$

计算整数 e

计算 d

公钥

私钥

p 和 q 都是素数, $p \neq q$

$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$

$de^{-1} \bmod \phi(n)$

$KU = \{e, n\}$

$KR = \{d, n\}$

加密

明文:

$M < n$

密文:

$C = M^e \bmod n$

解密

明文:

C

密文:

$M = C^d \bmod n$

RSA算法描述 – 更准确的版本

密钥产生

选择 p, q

p 和 q 都是素数, $p \neq q$

计算 $n = p \times q$

计算 $\lambda(n) = \text{lcm}(p-1, q-1)$

计算整数 e

$\gcd(\lambda(n), e) = 1, 1 < e < \lambda(n)$

计算 d

$d \equiv e^{-1} \pmod{\lambda(n)}$

公钥

$KU = \{e, n\}$

私钥

$KR = \{d, n\}$

加密

明文:

$M < n$

密文:

$C = M^e \pmod{n}$

解密

明文:

C

密文:

$M = C^d \pmod{n}$

$\gcd(\lambda(n), e) = 1, 1 < e < \lambda(n)$

卡米歇尔定理: $\gcd(m, n) = 1$, 使 $m^k = 1 \pmod{n}$ 成立的最小正整数 k , 记为 $\lambda(n)$

1) $n = pq$, $\rightarrow \lambda(n) = \text{lcm}(p-1; q-1)$

RSA算法描述 – 更准确更实用的版本

密钥产生

选择 p, q

p 和 q 都是素数, $p \neq q$

计算 $n = p \times q$

计算 $\lambda(n) = \text{lcm}(p-1, q-1)$

公开的指数 e 应该选取为某一小奇数值, $\text{gcd}(\lambda(n), e) = 1, 1 < e < \lambda(n)$

计算 d

$$d \equiv e^{-1} \pmod{\lambda(n)}$$

公钥

$$KU = \{e, n\}$$

私钥

$$KR = \{d, n\}$$

加密

明文:

$$M < n$$

密文:

$$C = M^e \pmod{n}$$

解密

明文:

$$C$$

密文:

$$M = C^d \pmod{n}$$

$$\lambda(n) = \text{lcm}(p-1; q-1)$$

公开的指数 e 应该选取为某一小奇数值, $\text{gcd}(\lambda(n), e) = 1, 1 < e < \lambda(n)$

加密很快, 指数小;

解密比较慢, 指数较大.

e 对所有用户可以是相同的, 建议使用 $e = 2^{16} - 1 = 65535$

安全性 – 示例

例子: $n=3*5=15$, $\psi=2*4=8$, $\lambda(n) = \text{lcm}(2,4)=4$; $e=3$, $d=3$;

加密:

- $m=2$, $c=2^3=8$, $2^{(3*3)}=1 \pmod n$;
- $m=7$, $c=7^3 \pmod{15}=13$, $7^{(3*3)}=7 \pmod n$;

如果 $c=8$, 能够推断出 $m=2$;

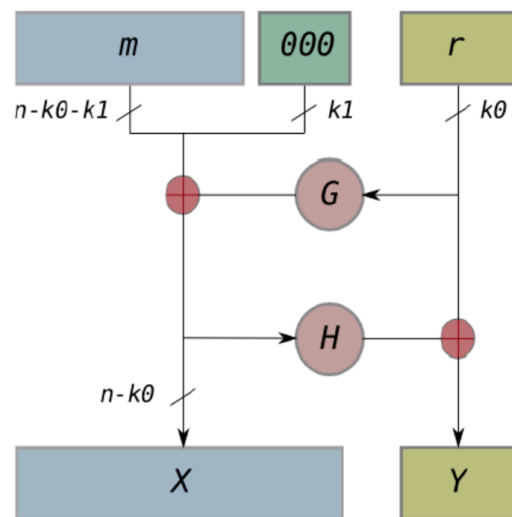
如果 $c=13$, 能够推断出 $m=7$;

- 安全目标

- 对同样的数据, 用同样的key进行RSA加密, 每次的输出密文都会不一样; 但是这些加密的结果都能正确的解密

RSA安全加密方案 – 编码函数（填充函数）的引入

RSA-OAEP (Optimal Asymmetric Encryption Padding)



参数

- n : RSA 的模
- k_0, k_1 : 协议参数
- m : 明文, 长 $(n - k_0 - k_1)$ bit
- G, H : Hash函数.

加密

- 对 m 进行 k_1 个 0 填充
- r 是随机的 k_0 比特
- G : 将 r 扩展为 $n - k_0$ bit
- $X = m00..0 \oplus G(r)$
- H : 将 X 压缩到 k_0 bits.
- $Y = r \oplus H(X)$
- $W = X || Y$
- $C \equiv W^e \pmod{n}$

- 密码学教材和 Wikipedia 上似乎都把明文直接作为 m , 直接乘上 e 次幂 $\text{mod } n$ 作为密文。
- 安全的方式: 明文和密文是1对多的关系
 - 对同样的数据, 用同样的key进行RSA加密, 每次的输出都会不一样; 但是这些加密的结果都能正确的解密
- 方法: 引入随机性

评价:

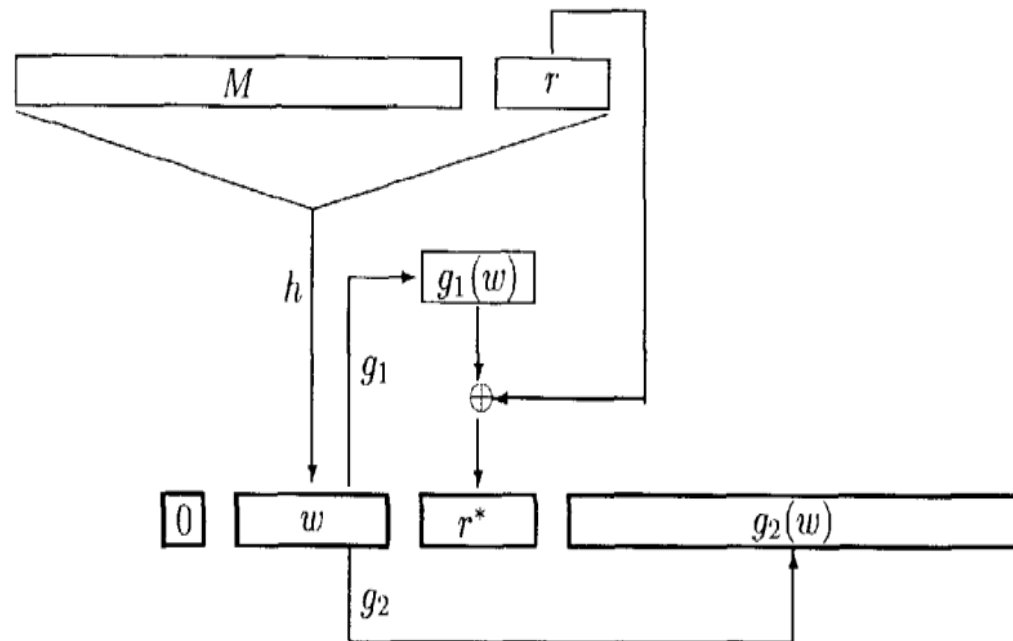
- 具有可证安全性
- 相当于两轮的Feistel结构
- 目前进入工业标准的为OAEP+

解密

- 对 C 解密得到 X 和 Y
- $r = Y \oplus H(X)$
- $m00..0 = X \oplus G(r)$
- 返回 m

Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption: **How to Encrypt with RSA**. In Alfredo De Santis, editor, Advances in Cryptology—EUROCRYPT 1994

RSA安全签名方案 - RSA-PSS



- RSA-PSS: Probabilistic Signagture Scheme

Fig. 1. PSS: Components of image $y = 0 || w || r^* || g_2(w)$ are darkened. The signature of M is $y^d \bmod N$.

Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures: **How to Sign with RSA** and Rabin. In Ueli M. Maurer, editor, Advances in Cryptology—EUROCRYPT 1996.

Paillier同态加密算法

Paillier 同态加密算法

Key generation

加密算法

解密算法

key generation:

1. 随机选择两个大素数 p 和 q ;
2. 计算 $n=pq$, $\lambda=\text{lcm}(p-1, q-1)$, lcm 是求两个数的最小公倍数。
3. 随机选择基 g , $g \in \mathbb{Z}_n^*$, 且满足 $\gcd(L(g^\lambda \bmod n^2), n) = 1$, 其中 $L(x)=(x-1)/n$
4. 计算 $\mu = (L(g^\lambda \bmod n^2))^{-1} \pmod n$

那么: 公钥 $pk=(n, g)$;

私钥 $sk=(\lambda, \mu)$;

- 其实, u 是可以公开的 (离散对数问题: 从 u 计算出 λ 是个困难问题)

Paillier 同态加密算法

Key generation

加密算法

解密算法

Encryption:

plaintext $m < n$

select a random $r < n$


ciphertext $c = g^m \cdot r^n \bmod n^2$

Decryption:

ciphertext $c < n^2$

plaintext $m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$

其中 $L(x) = (x-1)/n$

- 正确性: $r^{n\lambda} \equiv 1 \bmod n^2$ 
- 安全性: 合数剩余类问题的困难性

Encryption [edit]

1. Let m be a message to be encrypted where $0 \leq m < n$
2. Select random r where $0 < r < n$ and $r \in \mathbb{Z}_n^*$ (i.e., ensure $\gcd(r, n) = 1$)
3. Compute ciphertext as: $c = g^m \cdot r^n \bmod n^2$

Decryption [edit]

1. Let c be the ciphertext to decrypt, where $c \in \mathbb{Z}_{n^2}^*$
2. Compute the plaintext message as: $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$

$$m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$$

Apply the private key λ and use Carmichael's theorem:

$$c^\lambda = (g^m \cdot r^n)^\lambda = g^{m\lambda} \cdot r^{n\lambda} = g^{m\lambda}$$

Make use of the relationship $(1+n)^x \equiv 1 + nx \bmod n^2$

$$g^{m\lambda} = ((1+n)^{\alpha\beta n})^{\lambda m} = (1+n)^{\alpha\lambda m} \beta^{n\lambda m} \equiv (1 + \alpha\lambda mn) \bmod n^2$$

Apply the $L(X)$ function,

$$\frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n = \frac{L(1 + \alpha\lambda mn)}{L(1 + \alpha\lambda n)} \bmod n = \frac{\alpha\lambda mn}{\alpha\lambda n} \bmod n = m$$

一. 定义:

$$\mathbb{Z}_n = \{0, 1, \dots, n-1\}$$

$$\mathbb{Z}_n^* = \{a \mid 0 < a < n, \gcd(a, n) = 1\}$$

$$\text{例: } \mathbb{Z}_7 = \{0, 1, \dots, 6\}$$

$$\mathbb{Z}_8 = \{0, 1, \dots, 6, 7\}$$

$$\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$$

$$\mathbb{Z}_8^* = \{1, 3, 5, 7\}$$

二. 定理

1. 费马定理: p 为素数, $\gcd(a, p) = 1$, 则 $a^{p-1} \equiv 1 \pmod{p}$.

2. Carmichael 定理: $\gcd(a, n) = 1$, 则 $a^{\lambda(n)} \equiv 1 \pmod{n}$.

① 若 $n = p$ 为奇素数, $\lambda(n) = p-1$ (费马定理)

② 若 $\gcd(a, n) = 1$, 则 $\lambda(ab) = \text{lcm}(\lambda(a), \lambda(b))$

推论: $n = pq$, $\lambda(n) = \text{lcm}(p-1, q-1)$

3. 若 $g \in \mathbb{Z}_n^*$, 则存在唯一的 $(a, b) \in \mathbb{Z}_n^* \times \mathbb{Z}_n^*$ 满足 $g = (1+n)^a \cdot b^n \pmod{n^2}$

1. Key Generation

① 随机选择两个大素数 $p, q \Rightarrow \begin{cases} \lambda = \text{lcm}(p-1, q-1) \\ n = pq \end{cases}$ → 求公钥及

② 随机选择 $g \in \mathbb{Z}_{n^2}^*$, 且 $\gcd(\lambda, \text{ord}(g)) = 1$ → $\in \mathbb{Z}_n^*$

③ 公钥: (n, g)
私钥: λ

$r \neq p \cap r \neq q$
①

2. 加密 (n, g) ① 随机选择一个整数 r , $r \in \mathbb{Z}_n^*$ (i.e. $\gcd(r, n) = 1$)

$$c = E(m, r) = g^m \cdot r^n \pmod{n^2}$$

3. 解密 (λ) $m = D(c) = \frac{\log(c \pmod{n^2})}{\log(g \pmod{n^2})} \pmod{n}$

正确性证明:

$$m = \frac{\log(c \pmod{n^2})}{\log(g \pmod{n^2})} \pmod{n} \stackrel{①}{=} \frac{\log(g^m \cdot r^n \pmod{n^2})}{\log(g \pmod{n^2})} \pmod{n} \stackrel{②}{=} \frac{\log(g^m + n \cdot am)}{\log(1 + n \cdot a\lambda)} = \frac{n \cdot am}{n \cdot a\lambda} = m$$

$$① \because r \in \mathbb{Z}_n^*, \therefore r^n \equiv 1 \pmod{n} \Rightarrow r^n \equiv 1 \pmod{n^2} \Rightarrow c^{\lambda} = (g^m \cdot r^n)^{\lambda} = g^{m\lambda} \cdot r^{n\lambda} = g^{m\lambda}$$

$$② \because g \in \mathbb{Z}_n^*, \therefore \text{存在唯一 } (a, b) \in \mathbb{Z}_n^* \times \mathbb{Z}_n^* \text{ 满足 } g = (1+n)^a \cdot b^n \pmod{n^2} \begin{cases} \because b \in \mathbb{Z}_n^* \\ \therefore b^n \equiv 1 \pmod{n} \\ \Rightarrow b^{n\lambda} \equiv 1 \pmod{n^2} \end{cases}$$

$$\Rightarrow g^{m\lambda} = [(1+n)^a \cdot b^n]^{m\lambda} = (1+n)^{am\lambda} \cdot b^{nm\lambda} = (1+n)^{am\lambda}$$

$$\stackrel{③}{=} (1 + n \cdot am\lambda) \pmod{n^2}$$

$$③ (1+n)^x \equiv 1 + nx \pmod{n^2}$$

$$\boxed{1 + nx + \frac{n^2 x(x-1)}{2} + \dots} \\ \equiv 1 + nx$$

$$④ \because r^n \equiv 1 \pmod{n}$$

$$\therefore (r^n)^n = (1 + t \cdot n)^n \stackrel{⑤}{=} 1 + n \cdot tn + \dots \equiv 1 \pmod{n^2}$$

- 密钥生成

- $P=3, q=5, \lambda=\text{lcm}(2,4)=4, n=15,$
- $g=11$, 计算出 $u=1$

Encryption:

plaintext $m < n$
select a random $r < n$
ciphertext $c = g^m \cdot r^n \bmod n^2$

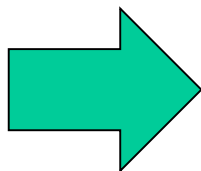
Decryption:

ciphertext $c < n^2$
plaintext $m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$

ensure $\gcd(r, n) = 1$

- 加密

- $r=7, m=6$: 计算 $c=73$
- $r=11, m=6$: 计算 $c=161$



- 解密

- $c^\lambda \bmod n^2 = 91$
- $L(g^\lambda \bmod n^2) = 1$
- $m = L(91)/1 \bmod 225 = 6$

$r=3, m=6$: 计算 $c=27$; 解密: $c^\lambda \bmod n^2 = 216$ 【错误, 因为不满足 $\gcd(r, n) = 1$ 】

同态性质 – 加法同态

加密算法: $c=E(m,r)=g^m * r^n \bmod n^2$;

加同态性质: $\forall m_1, m_2 \in \mathbb{Z}_n$ and $k \in \mathbb{N}$

①两个密文的乘积等于两数之和的密文

$$\mathbf{E(m_1,r_1)*E(m_2,r_2) \bmod n^2=E(m_1+m_2)\bmod n.}$$

证明: $E(m_1,r_1)*E(m_2,r_2) \bmod n^2=g^{m_1} * r_1^n * g^{m_2} * r_2^n \bmod n^2=g^{m_1+m_2} * (r_1r_2)^n \bmod n^2=E(m_1+m_2)$

②密文的变量幂等于明文和变量乘积的密文

$$E(m_1,r_1)^k \bmod n^2=E(k*m_1) \bmod n$$

证明: $E(m_1,r_1)^k \bmod n^2=(g^{m_1} * r_1^n)^k \bmod n^2=g^{k*m_1} * r_1^{n*k} \bmod n^2=E(k*m_1) \bmod n$

$$\mathbf{F(C_1,C_2)=C_1*C_2;}$$

$$\mathbf{f(m_1,m_2)=m_1+m_2}$$

示例 – 加法同态

密钥生成

p	q	$\lambda=\text{lcm}(p-1,q-1)$	$n=pq$	g	u	n^2
3	5	4	15	11	1	225
3	5	4	15	11	1	225
3	5	4	15	11	1	225

加密 → 密文相乘(73*103=7519) → 对结果解密: $D(7519)=8$

r	m	$c=g^m \cdot r^n \bmod n^2$	$c^\lambda \bmod n^2$	$L(g^\lambda \bmod n^2)$	$L(c^\lambda \bmod n^2)$
7	6	73	91	1	6
7	2	103	31	1	2
		7519	121	1	8

签名 - 私钥加密公钥解密

密钥生成

公钥 $pk' = (n, g, \mu)$, 私钥 $sk' = (\lambda, g) = g^\lambda$

加密算法

$E'(m, r, \lambda) = E(m, r)^\lambda = g^{m*\lambda} * r^{n*\lambda} \pmod{n^2} = g^{m*\lambda} = c$

解密算法

$D'(c) = L(c \pmod{n^2}) * \mu \pmod{n} = m$

Encryption $E'(m, r, \lambda)$

$$E'(m, r, \lambda) = E(m, r)^\lambda = g_p^{m\lambda} \cdot r^{n\lambda} \pmod{n^2} = c.$$

Decryption $D(c)$

$$D(c) = L(c \pmod{n^2}) \cdot \mu \pmod{n}.$$

- 正确性：没有问题。
- 保持加法同态
- 安全性
 - 不再是概率加密，安全性？
 - 从 u 可以计算出来 g^λ ，任何人都可以模仿进行签名！！



- 1) Paillier, Pascal (1999). "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". *EUROCRYPT*. Springer. pp. 223–238.
- 2) Paillier, Pascal; Pointcheval, David (1999). "Efficient Public-Key Cryptosystems Provably Secure Against Active Adversaries". *ASIACRYPT*. Springer. pp. 165–179.
- 3) Paillier, Pascal (1999). *Cryptosystems Based on Composite Residuosity* (Ph.D. thesis).
- 4) Paillier, Pascal (2002). ["Composite-Residuosity Based Cryptography: An Overview"](#)
- 5) Nabeel, Mohamed, Stefan Appel, Elisa Bertino, and Alejandro Buchmann. "Privacy preserving context aware publish subscribe systems." In *International Conference on Network and System Security*, pp. 465-478. Springer, Berlin, Heidelberg, 2013.

全同态加密

“密码学的圣杯”

- 2009年，IBM公司的克雷格·金特里（Craig Gentry）发表了一篇文章，公布了一项关于密码学的全新发现——一项真正的突破。
- 他发现，对加密的数据进行处理得到一个输出，将这一输出进行解密，其结果与用同一方法处理未加密的原始数据得到的输出结果是一样的。这听起来就像是不知道问题也能给出问题的答案一样。
- “全同态加密”的技术被冠以“密码学的圣杯”的称号

2009年, Craig Gentry博士论文



- This problem was posed by Rivest et al. in 1978.

Abstract

Craig Gentry



Residence Berkeley, California, United States

Citizenship United States

Fields Physics

Alma mater Duke University B.A. (1995), J.D. Harvard Law School (1998), and Ph.D. Stanford University (2009)

Notable awards MacArthur Fellowship (2014)

We propose the first fully homomorphic encryption scheme, solving a central open problem in cryptography. Such a scheme allows one to compute arbitrary functions over encrypted data without the decryption key – i.e., given encryptions $E(m_1), \dots, E(m_t)$ of m_1, \dots, m_t , one can efficiently compute a compact ciphertext that encrypts $f(m_1, \dots, m_t)$ for any efficiently computable function f . This problem was posed by Rivest et al. in 1978.

Fully homomorphic encryption has numerous applications. For example, it enables private queries to a search engine – the user submits an encrypted query and the search engine computes a succinct encrypted answer without ever looking at the query in the clear. It also enables searching on encrypted data – a user stores encrypted files on a remote file server and can later have the server retrieve only files that (when decrypted) satisfy some boolean constraint, even though the server cannot decrypt the files on its own. More broadly, fully homomorphic encryption improves the efficiency of secure multiparty computation.

Our construction begins with a somewhat homomorphic “bootstrappable” encryption scheme that works when the function f is *the scheme’s own decryption function*. We then show how, through recursive self-embedding, bootstrappable encryption gives fully homomorphic encryption. The construction makes use of hard problems on ideal lattices.

全同态加密

- 全同态加密是指能够在不知道密钥的情况下，对密文进行任意计算，即对于任意有效的 f 及明文 m ，有性质 $f(E(m))=E(f(m))$ 。
- 从理论上讲，所有的函数都可以由加法和乘法多次复合来实现，因此全同态加密算法在设计的时候可以首先考虑其对加法和乘法都同态，再将其扩展到任意函数之上。

基于格的全同态加密发展的三个阶段

- 基于理想格以Gentry方案为蓝图的FHE构造
- 基于LWE假设, 利用密钥交换等技术来实现FHE的构造
- 基于LWE假设, 利用近似特征向量构造的FHE方案

第一阶段



- 2009年, Gentry 的FHE体制:
 - 基于理想格(ideal lattice) 上的有界编码问题(BDDP) 和稀疏子集和问题(SSSP)
- 构造过程:
 - 设计一个具备有限次密文运算的同态加法和同态乘法的近似同态 (SWHE) 加密体制
 - 引入 “Bootstrapping” 程序, 利用重加密的方法对密文进行更新, 以此控制噪声膨胀, 保证解密正确性, 从而实现任意次的密文同态运算
- 缺点:
 - 无法抵抗选择密文攻击 (CCA), 只能达到选择明文攻击 (CPA) 安全

第二阶段 – BGV系统

2011年，**Brakerski**和**Vaikuntanathan**提出了一个新的全同态加密体系，这一体系基于格（**lattice**）加密的另一种假设**Learning With Errors（LWE）**。在同一年，**Brakerski**，**Gentry**与**Vaikuntanathan**这三人一起把这个体系做完了，并且正式发表出来。他们发明的全同态系统简称为**BGV系统**。

BGV系统是一个**有限级数**的同态加密系统，但是可以通过**Bootstrapping**的方式来变成全同态系统。

BGV系统相比起**Gentry09**提出的系统，使用了更加实际一点的**LWE**假设。一般来说我们都把**BGV系统**称之为**第二代全同态加密系统**。

第三阶段 – GSW方案

- 2013 年, **Gentry-Sahai-Waters(GSW)**方案
 - 近似特征向量方法来构建**FHE** 方案
- 构造方法:
 - 该方案的同态加法和同态乘法都只是通过做简单的矩阵加法和乘法来实现
 - **GSW-FHE** 方案**相对简单、快速, 容易理解**
 - 实施同态运算时不需要使用计算公钥, 而只需借助用户的公钥即可实现

GSW系统和BGV相似, 本身具有有限级数全同态性质, **基于更加简单的LWE假设, 并且通过Bootstrapping可以达到全同态。**
我们一般把GSW系统称为**第三代全同态加密系统。**

研究方向

- 基于原来的三代全同态系统之上，出现了各种各样新的设计，致力于优化和加速BGV与GSW系统的运行效率。
 - IBM基于BGV系统开发了一个开源的全同态运算库HElib，并且成功的移植到各大移动平台上。
 - 与此同时，还有另外一个开源项目TFHE也非常值得注意。TFHE是基于GSW系统，又加以了各种优化与加速，现在也非常的有名。
- 在开发传统的全同态库之外，也有很多团队在研究如何通过GPU，FPGA，ASIC等异构硬件来更好的加速全同态加密算法的计算。
 - 比如cuFHE就是一个比较有名的基于CUDA的GPU加速全同态加密系统。
- 现在业界对于FHE的研究百花齐放，不少人都~~在不同的角度和应用需求上~~在研究全同态系统。

- 效率：和所有好技术一样，将同态加密技术应用到现实生活还需要一段时间。
 - 该技术还需要解决一些应用上的障碍。其中之一就是大量的计算需求。
 - Gentry表示，如果再一个简单的明文搜索中应用同态加密技术，将使得运算量增加上万亿倍。
- 应用前景： 领域广泛，云计算、多方保密计算、匿名投票等

- 直到今天，我们已经拥有了多种可行的FHE实现方法，但是现在大家还在不断追求的是FHE系统运转的效率。
 - 拿现在最前沿的FHE库来说，在移动平台上同态计算一些比较简单的逻辑可能要少则花上几十毫秒，多则花费几十秒的时间。
 - 这些时间单位对于计算机系统来说是极其缓慢的。
- 如何可以让FHE系统更加高效率的在异构平台上运行，仍然是一个未解之谜。
 - 如果这道难题一旦被解决了，那么把所有的电脑运算都转为全同态，代理在第三方的云端上进行计算，都是伸手可得的未来。

实用性的解决方案

CryptDB

实用性的解决方案

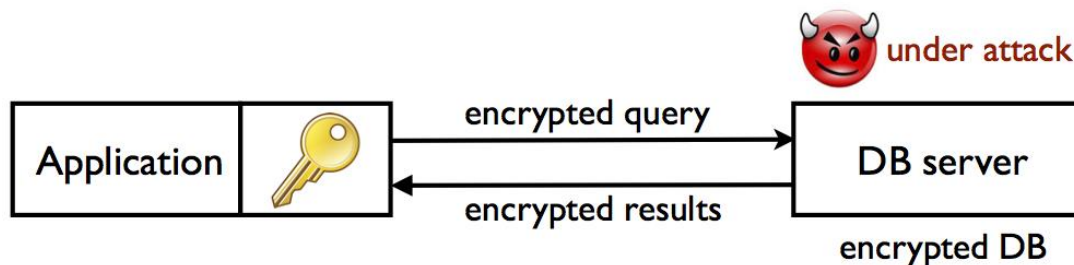
- 目前的全同态加密方案在实用性上还存有问题，因为该方案耗费的计算时间太长。
- CryptDB首次解决了实用性的问题，它将数据嵌套进多个加密层，每个都使用不同的密钥，允许对加密数据进行简单操作。
- 此前全同态加密方案加密的数据操作所增加的计算时间是数以万亿倍，而CryptDB只增加了15-26%左右。
 - 最新的全同态加密技术已经大大缩短了计算时间，但是和CryptDB相比还是太慢了。

- 完全同态加密旨在支持单一加密方案中的所有功能，这使得简单操作变得更为缓慢。就目前而言，设计实用技术方案的关键是要摆脱“一个加密系统将适用于所有内容”的想法。
- 为了支持各种操作，需要使用各种专门的加密机制。每种机制只能做一件事，但是它们加在一起就涵盖了相当多的领域。
 - 目前，对于许多常见操作有专门并且迅速的算法，可以支持：加法，乘法，比较等式或按顺序设置交集，多项式计算，机器学习分类任务，搜索加密文本等。
 - 使用所需的机制来加密数据，并因此存储多组不同的加密数据，从而允许使用加密的数据进行各种不同的计算。
 - 只需在加密数据集之间进行切换，在每个实例中使用与需要完成的操作相对应的数据集就可以了。
- CryptDB正是基于这个思路来设计的。

- Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan.
[CryptDB: Protecting Confidentiality with Encrypted Query Processing.](#)
 In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP)*, Cascais, Portugal, October 2011.
 (This is the main paper describing CryptDB.)



- Online applications are vulnerable to theft of sensitive information because adversaries can exploit software bugs to gain access to private data, and because curious or malicious administrators may capture and leak data. CryptDB is a system that provides practical and provable confidentiality in the face of these attacks for applications backed by SQL databases. It works by *executing SQL queries over encrypted data* using a collection of efficient SQL-aware encryption schemes. CryptDB can also *chain encryption keys to user passwords*, so that a data item can be decrypted only by using the password of one of the users with access to that data. As a result, a database administrator never gets access to decrypted data, and even if all servers are compromised, an adversary cannot decrypt the data of any user who is not logged in. An analysis of a trace of 126 million SQL queries from a production MySQL server shows that CryptDB can support operations over encrypted data for 99.5% of the 128,840 columns seen in the trace. Our evaluation shows that CryptDB has low overhead, reducing throughput by 14.5% for phpBB, a web forum application, and by 26% for queries from TPC-C, compared to unmodified MySQL. Chaining encryption keys to user passwords requires 11-13 unique schema annotations to secure more than 20 sensitive fields and 2-7 lines of source code changes for three multi-user web applications.

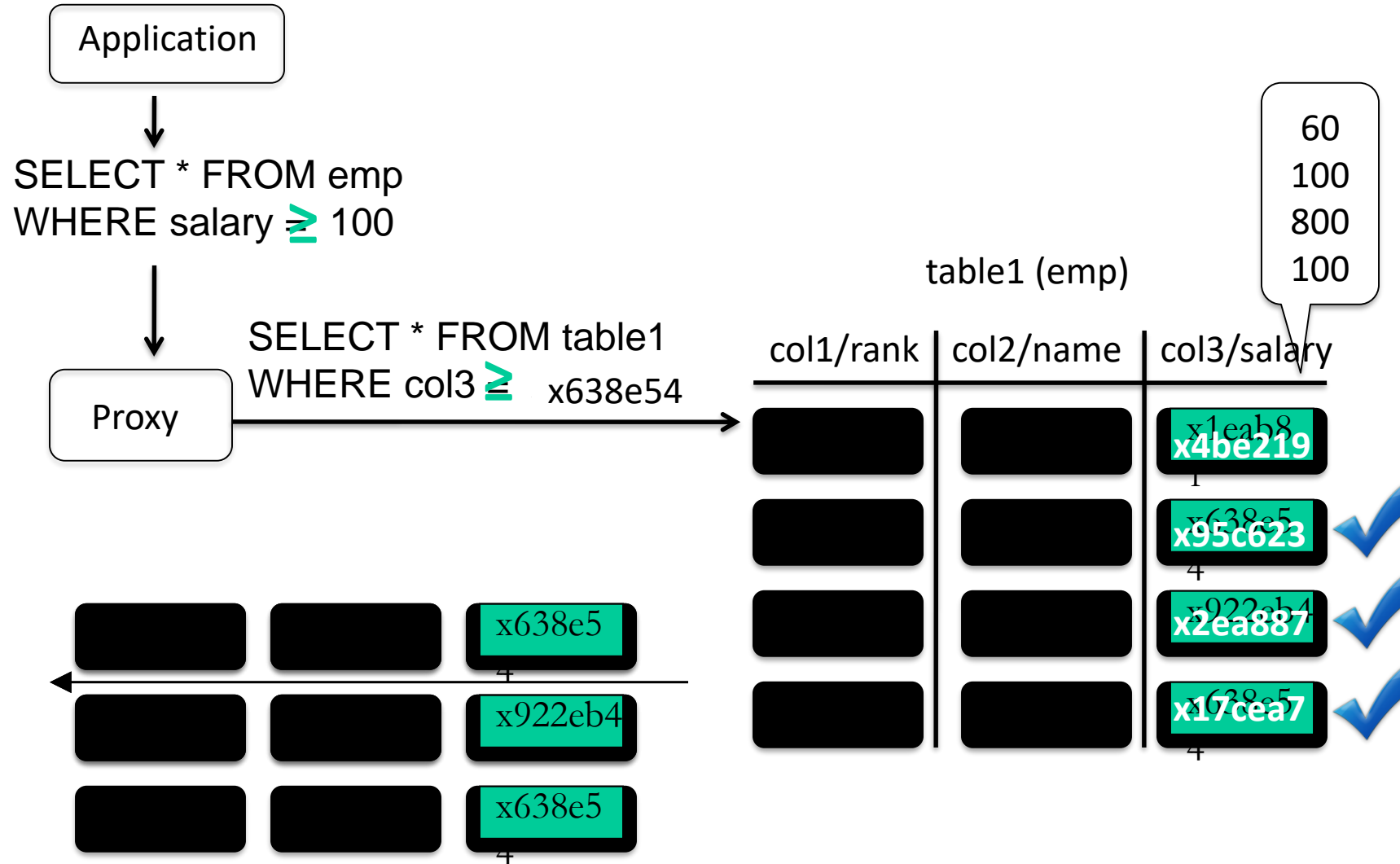


```
mysql> select * from table YYAXXURGXH;
+-----+-----+-----+
| CKEWNNKTJVoDET | WIWWHTQZBQoOPE | ISKNOMNOYSaGG |
+-----+-----+-----+
| 12350108222818996780 | 14614281758989003640 | ;Li000e0b(T 5Te0 |
(0000701080[00f0000]0+0 00H"050\0000F0bv0Ts000400y0k000P02m000"000" |
00%60000'0<t4m0B00);00l0 00p0 |
000c)00l0cc00000 0?mk00U0qi0s000ne0C00000e" |
| 15753844592636354766 | 16607196790268127689 | R P-0i00c000000}f Ea] |
UJH=000 &000A0h080f0u000:0004000s0z00s00\0qZ006C0 |
y000U000b40e00j00c00)00 |
| 13411675538537145840 | 12454649889312668381 | 000pK^v0000000000}000000 |
00R000"0"0J 00_000{tm00}009T0M |
=f000g0000:0CR)0000040og0+00Tg0+0z0000M000{0(00000MKhuv000000F000(00000C |
```

- 来自MIT的开源技术，可对加密数据SQL查询
- 目前，面对国防、医疗保健以及金融行业的数据安全问题，越来越多的应用程序开始应用于数据的处理。其实，任何一个组织都难以保证重要的数据不会被窥探；而对于公有云来说，这个问题更为严重。
- CryptDB：允许用户查询加密的SQL数据库，而且能在不解密储存信息的情况下返回结果。

- 工作原理：在对加密的数据执行SQL查询时，使用的是一个SQL能够‘理解’的加密方案来进行加密。CryptDB同样将加密密钥和用户的密码进行了捆绑，这样的话数据项只有使用相应的用户的密码登陆才可以进行解密。
- 作为结果，就是数据库管理员也不能接触到加密的数据，即使服务器被攻破，攻击者也无法解密用户的数据，如果该用户没有登录的话。

Example



- CryptDB已经开放了其源代码（仅限于学术研究目的）

<http://css.csail.mit.edu/cryptdb/>

- 尽管CryptDB 只能进行有限种类的查询，谷歌是该技术的一大支持者，并使用它在其基于云计算的、搜索大量数据集的BigQuery服务中提供加密查询。
 - 增加一个额外的设备，比如在搜索和检索过程中的代理服务器，通常会减慢速度。
 - 这个系统通过将请求数据的软件和存储加密数据的数据库之间的放置一个代理服务器，来保证对加密数据的分析。这个代理使用旨在比较和分析加密信息的算法，在某些情况下，代理需要去除不同的加密层来更好的分析数据，但是这种想法就是不会将数据完全的加密成为纯文本。

云环境下的细粒度访问控制技术

- 传统的访问控制采用访问列表 (Access Control List, ACL) 来管理用户的访问权限。这种访问授权机制在用户和服务器之间设置访问策略，用户通过 ACL 获得授权，这就要求数据存储服务器是安全可信的。
- 在云环境中，服务器并不是完全可信任的，而且如果服务器被入侵，用户的所有数据都将泄露。
 - 如果对用户的敏感数据进行加密存储，那么云服务器不能从密文中获取明文信息，而且即使服务器被入侵，仍可保证用户数据的保密性。
 - 但很多时候，用户存储这些大数据，是为了实现数据的共享，使得被授权的用户可以获得数据。**传统的加密机制很难确保用户数据的机密性同时为数据共享实现细粒度的访问控制特性。**

函数加密 (Functional Encryption, FE)

- 函数加密 (Functional Encryption, FE) 丰富的表达式使得它可以在云环境中应用, 保证数据机密性的同时提供一种非常灵活的对加密密文可解密的访问控制方法。

基于函数加密, 任何人都能够用主公钥 (master public key, mpk) 加密数据; 主私钥 (master secret key) 的持有者能够为特定函数生成对应的私钥, 例如, 给函数 f 生成私钥 sk_f 。任何拥有 sk_f 和明文 x 的密文 c 的人, 都能够获得明文 x 的函数 $f(x)$ 的明文计算结果。

函数加密保证对手只能够获得 $f(x)$, 而不泄漏明文 x 的任何信息。

函数加密方案将用户密钥和密文分别与特定的谓词和属性相对应, 只有属性满足特定谓词的用户才是满足访问结构的, 也就是可以授权解密相关密文的用户。

小结

- 动机
- 概念
- 工作原理
- 研究进展：乘法同态；加法同态；全同态
- 实用性技术方案
- 函数加密的概念
- 展望

存储信息的安全性是一个重要的问题，而**对加密数据的计算**是解决该问题的关键技术之一，它能够保护敏感信息以防被窃取。原因很简单，即使是持有数据的公司也不能明白加密后信息的意义，黑客们就更没有什么值得窃取的了。



北京邮电大学

Beijing University of Posts and Telecommunications

感谢聆听！

功能改进 – 备份

