Submitted in total fulfilment of the requirements for the degree of

Bachelor

in Computer Science and Technology

Privacy Homomorphisms: Approaches, Implementation and Applications

Zeming Wang

Supervisor

Prof. Zhenfu Cao

Depart of Computer Science and Engineering, School of

Electronic,Information and Electrical Engineering

Shanghai Jiao Tong University

Shanghai, P.R.China

June 6th, 2013

同态加密方案实现及其应用

摘　　要

隐私同态是指允许对加密数据进行特定运算并可解密得到运算结果的加密技术。本文对现有的隐私同态技术进行了研究，介绍了目前已有的两种典型的隐私同态算法，也介绍对它们进行攻击的方法。本文还提出了一个新的隐私同态算法，可实现域上的全部运算，而目前大部分隐私同态只能实现环上的同态运算。该算法运用指数形式加密，数学形式简单，方便计算。但该算法只可抗唯密文攻击，不可抗已知明文攻击。为了增强其安全性，本文也讨论了可能的改进方法。隐私同态允许在不泄露隐秘数据的前提下完成对数据的计算，这一特性使得隐私同态在云计算、数据库加密等领域有广阔的应用前景。本文最后介绍了隐私同态的应用实例。

关键词： 隐私同态，同态加密，安全计算，计算代理，数据代理

Privacy Homomorphisms: Approaches, Implementation and Applications

ABSTRACT

Privacy homomorphisms are encryption transformations which allow direct computations on encrypted data and recovering results through decryption. In this paper, we study two typical privacy homomorphisms so far, together with the attacks againt them. The author's contribution is to show a new privacy homomorphism, which allows all field operations on encrypted data, while most of the privacy homomorphisms so far only support homomorphic operations on rings. The new approach uses very simple mathematic form and is easy to compute. As to the security, the new approach is secure against ciphertext-only attacks, but not secure under known-cleartext attacks. We analyse the attacks that exist and propose some possible improvements. Privacy homomorphisms allow computation and analysis on encrypted data while keeping the sensitive data private. Therefore privacy homomorphisms are believed to have promising prospects in many fields like cloud computing and database encryption. So finally we briefly present the applications of privacy homomorphisms.

KEY WORDS: Privacy homomorphisms, Homomorphic encryption, Secure computing, Calulation- and Data-delegation

目　　录

第一章　绪论

隐私同态 (Privacy Homomorphisms) 是一种密码学技术，也译作秘密同态，有时也称为同态加密。隐私同态技术的出现是由于对安全代理计算的需求。如今随着互联网的普及，以及云计算的发展，这种需求变得越来越多，这也使得隐私同态变成了一个时下热门的研究课题。

1.1　研究背景及研究目的

密码学 (Cryptography) 是研究如何保密信息的学科。在密码学中，将可读的信息转化为不可读的数据，这一过程称为加密 (Encryption)。原始的可读数据被称为明文 (Cleartext or Plaintext)，加密后的数据称为密文 (Ciphertext)。明文的所有可能取值构成明文空间 (Cleartext-Space)，密文的所有可能取值构成密文空间 (Ciphertext-Space)。为了完成加密，通常还需要明文之外的信息参入，也就是密钥 (Key)。加密的逆过程，也就是从密文恢复明文的过程，称为解密 (Decryption)。显然并非所有人都有解密的权限，只有持有密钥者才能执行解密。

加密所用的密钥和解密所用的密钥，可以相同，也可以不同。若二者相同，这样的加密方案被称为对称加密 (Symmetric Encryption)；若二者不同，则这样的加密方案被称为公钥加密 (Public-Key Encryption)。因此产生的两种密码学体制分别称为对称密码学 (Symmetric Cryptography) 和公钥密码学 (Public-Key Cryptography)。

密码学在诞生之初常常是用于军事用途，用来保护军事中的秘密情报。随着时代的发展，密码学的用途越来越广泛。现在，密码学已经被普遍应用于我们生活的各个角落；无论是在我们使用电子银行时，还是我们在发送电子邮件时，密码学都被用来保护我们的隐私数据。如今我们正在步入云计算 (Cloud Computing) 的时代。云计算允许用户将数据存储在云端，并用云端提供的服务来操作数据。但是限于目前的技术，用户存储在云端的数据都是未经加密的明文，以便于云端对数据进行计算。也就是说，用户如果把隐私数据存储在云端，就等于把隐私数据暴露给了云端的计算机。虽然云服务提供商声称会保护用户

隐私，但我们仍希望能有一种方法，用加密来保护用户在云端的数据，同时在不经过解密的情况下，云端也能对数据进行相应的计算。隐私同态（或同态加密）就是用来解决这一问题的。

隐私同态 (Privacy Homomorphisms, 也译作为秘密同态）是一种加密方式，它允许对密文进行特定的代数运算并得到加密的结果，该结果解密后恰好等于直接对明文进行运算的结果。隐私同态允许某一方在不知道密钥的情况下，对密文进行某种运算；运算的结果在经密钥持有者解密后，等于对明文进行同样运算的结果。

一个隐私同态方案可以允许同态地进行一种或几种运算，比如加法、减法、乘法等。如果一个隐私同态方案允许对密文进行任意运算，则称之为全同态加密 (Fully Homomorphic Encryption)。全同态加密是一个非常新的研究领域。目前虽然有理论上可行的方案，但由于效率太低，尚无法应用到实际中。

隐私同态技术的意义在于，它解决了将数据及其操作委托给第三方时的保密问题。隐私同态保证了数据处理方仅仅能够处理数据，而对数据的具体内容一无所知。因此该技术特别是全同态技术一旦能够得以应用，将是数据安全领域的巨大进步。以下简单列举隐私同态的实际应用，以便窥见该技术的研究意义和巨大潜力。

例如，银行有一些秘密的交易数据需要统计分析，但是银行没有足够强大的计算能力，于是银行应用隐私同态把数据加密后交给数据中心来处理。数据中心拿到加密后的数据，进行相关运算并将结果返回给银行。银行解密后得到统计的结果。在这个过程中，数据中心得到的仅仅是加密数据，因此无法知晓秘密的真实数据；由于隐私同态的应用，银行也得到了它要的计算结果。实际中，不仅仅是银行，有很多机构需要对数据进行分析，而本身又没有强大的计算能力；想把数据外包给数据中心，又不想泄露隐私数据。隐私同态的应用将解决这一困境。

再如，假设要设计一个电子投票系统。投票方不希望计票方知道自己的投票情况；而计票方需要在不知道具体投票方投票的前提下统计出投票数。应用隐私同态技术，投票方将加密后的投票送到计票方；计票方对密文执行统计操作，得到计票结果；宣布方拿到加密的计票结果，解密后得到总票数。

总之，隐私同态将有广泛的应用，前景光明。但目前仍有许多研究工作要做。对于现有的部分同态加密算法，虽然不能对密文进行任意操作，但仍可应

用在某些特定场景，而目前这样的方案，特别是具有良好安全性的方案，并不多。而全同态加密算法仍处于理论阶段，距离实际应用还有很长的距离。

1.2 国内外研究现状

隐私同态的概念最初由 Rivest 等人提出[1]，其思想源自于 RSA 公钥密码的同态性：将用同一公钥加密的若干密文相乘，结果解密后恰好等于原明文相乘的乘积。作者在文章 [1] 中给出了几个隐私同态的实例，这些最初的隐私同态使用对称密钥，通过简单的数学变换实现。其实，同态性存在于各种加密形式和方法中，包括对称加密和公钥加密，已远远超出最初提出时的加密形态。由于不同形式的同态加密之间差别很大，本文将其分为三类，即具有同态性的公钥密码、使用对称密钥的代数隐私同态和全同态加密，下面将分别介绍这三类同态加密的研究现状。

具有同态性的公钥密码：目前主流的公钥密码算法，如 RSA, ElGamal, Pallier 等，都具有部分同态性，即对某一运算，比如加法或乘法，具有同态性。例如，RSA 公钥密码体制满足乘法同态；ElGamal 公钥密码体制也可实现同态乘法；Pallier 公钥密码体制则可实现加法同态。公钥密码中的同态性还有很多，在此不一一列举。但在所有已知的公钥密码中，都不能同时实现同态加法和乘法。如 RSA 只满足乘法同态性，而不满足加法同态性；Boneh-Goh-Nissim 密码虽然可以实现加法和乘法，但只允许一次乘法操作，其实不能算真正意义上的乘法同态。其他公钥密码的同态性可参考附录A。由于能进行的同态运算相当受限，所以这种公钥式的同态加密只能应用于个别极其简单的应用中。

使用对称密钥的代数隐私同态：通常称可同时满足加法同态和乘法同态的隐私同态为代数隐私同态 (Algebraic Privacy Homomorphisms)。最原始的方案在文献 [1] 中被提出，但这些最初的隐私同态均被证明是不安全的 [2]。Domingo-Ferrer 基于对 [1] 中方案的改进，提出了一个可实现同态加法、减法和乘法的隐私同态方案[3]；但该方案被 Cheon, Kim, Nam 破解[4]。文献 [5] 中提出一个新的隐私同态，这个方案虽然短小，但是可以实现域上的全部操作，即可实现四则运算加法、减法、乘法和除法；但是作为代价，其安全性有所降低。文献 [6] 的方案是对 [3] 的升级，并且在文献中作者证明了其安全性；但该方案还是被 Wagner[7] 和 Bao[8] 破解了。代数隐私同态，相比与具有同态性的公钥密码，能实现更多的同态运算；考虑到隐私同态主要用于解决计算外包的问题，

而多数情况下的外包计算是不需要使用公钥密码的，所以代数隐私同态有着更广泛且实际的应用。但是这种隐私同态在提供更多同态运算的同时，也牺牲了安全性；这使得它只能应用于对安全性要求不高的情况下。

全同态加密：相对于前两者，全同态加密直到最近才被完整地提出。2009 年 Craig Gentry 提出了一个基于理想格 (Ideal Lattice) 的全同态加密方案[9]；该方案是第一个真正意义上的全同态加密方案。Gentry 的贡献使得该领域有了突破。受 Gentry 方案的启发，文献 [10, 11] 均是基于理想格的全同态加密方案。在 Gentry 之后，比较有代表性的研究是 2011 年 Brakerski 和 Vaikuntanathan 提出的基于容错学习 (Learning With Errors) 构造的全同态加密体制（简称 BV 体制）[12]，这一体制相比 Gentry 的体制有更可靠的安全性保证。文献 [13–15] 是针对该体制的改进，进一步提高了算法的效率。不同于前两类同态加密的思路，全同态加密针对位进行加密，而非整数；基于电路，实现同态位运算。截至目前，所有的全同态加密的效率都十分低下，无论是空间复杂度还是时间复杂度都很高；虽然在理论上可行，但距离实用还差很远。

1.3 主要内容及组织结构

上文所提的三种同态加密形态，虽然都是为了实现安全代理计算的问题，但是他们的思想和方法均差别很大。虽然全同态加密是该领域最前沿的成果，但并不代表其他方向的研究已经过时；他们各适用于不同的应用场景，各有其价值。具有同态性的公钥密码适用于需要使用公钥而计算简单的情况下，比如电子投票，只需要实现加法即可；使用对称密钥的代数隐私同态可实现常规算术，但只能用于无需公钥而且对安全性要求不高的情况下，比如安全统计计算；全同态加密理论上可以实现所有的运算，但目前尚无法应用。本文主要集中讨论使用对称密钥的代数隐私同态；在下文中，如无特殊说明，隐私同态均指使用对称密钥的代数隐私同态，以区别于具有同态性的公钥加密和全同态加密。

在下文中，第二章将介绍与隐私同态相关的数学和密码学知识。第三章将介绍几种典型的隐私同态，并介绍了相应的攻击方法；然后介绍一个新的隐私同态，并分析其安全性，以及可能的改进方法。第四章简要介绍了隐私同态的应用实例。第五章是全文的总结及未来展望。

第二章　基础理论

在开始介绍具体隐私同态的方案之前，首先简要介绍一下隐私同态的数学和密码学基础，包括所涉及的基本概念和各个符号的含义。

2.1　数学基础

2.1.1　数论基础

定义 2.1. 设 a, b 为整数，如果存在整数 m，使 $am = b$ 成立，则称 a 整除 b (a divides b)，记作 $a \mid b$.

定义 2.2. 如果整数 a 不能被除了 1 和 a 以外的任何数整除，则称 a 为素数 (Prime).

定义 2.3. 如果存在整数 n 满足 $n \mid (a-b)$，则称 a 同余于 b 模 n (a is equivalent to b modulo n)，记作 $a \equiv b \pmod{n}$，或 $a \equiv_n b$.

定义 2.4. 设 a, n 为整数，记 $a \bmod n$ 为满足 $a \equiv_n b$ 的值 b，其中 $0 \leq b < n$. 根据整数除法的定义知 b 唯一.

定义 2.5. 定义 a, b 的最大公约数 (Greatest Common Divisor) 为能同时整除 a, b 的最大整数，记作 $gcd(a, b)$；定义 a, b 的最小公倍数 (Lowest Common Multiple) 为同时能被 a, b 整除的最小整数，记作 $lcm(a, b)$.

定义 2.6. 设 a, b 为整数，如果 $gcd(a, b) = 1$，则称 a 和 b 互素 (Relatively Prime).

定义 2.7. 定义集合 $\mathbf{Z}_n$ 为小于 n 的所有非负整数，即

$$\mathbf{Z}_n = \{0, 1, ..., (n-1)\}.$$

定义 2.8. 定义集合 $\mathbf{Z}_n^*$ 为所有小于 n 并与 n 互素的正整数，即

$$\mathbf{Z}_n^* = \{x \mid 0 < x < n, gcd(x, n) = 1\}.$$

同余有如下性质:

定理 2.1 (性质 1)**.** 同余是一种等价关系，即有

(i) $a \equiv a \pmod{n}$;

(ii) $a \equiv b \pmod{n} \Leftrightarrow b \equiv a \pmod{n}$;

(iii) $a \equiv b \pmod{n}, b \equiv c \pmod{n} \Rightarrow a \equiv c \pmod{n}$.

定理 2.2 (性质 2)**.** 同余式可相加、相减和相乘，即若有

$$a \equiv b \pmod{n}, c \equiv d \pmod{n}$$

则

$$a \pm c \equiv b \pm d \pmod{n}$$

$$ac \equiv bd \pmod{n}.$$

定理 2.3 (性质 3)**.** 设 $d \geq 1$，$d \mid n$，那么，若

$$a \equiv b \pmod{n}$$

则

$$a \equiv b \pmod{d}.$$

定理 2.4 (性质 4)**.** 若 $n \geq 1$，$(a, n) = 1$，则存在 c 使得

$$ca \equiv 1 \pmod{n}.$$

这时称 c 为 a 对模 n 的逆，记作 a^{-1}.

定理 2.5 (性质 5)**.** 同余式组

$$a \equiv b \pmod{m_j}, j = 1, 2, ..., k$$

同时成立的充要条件是

$$a \equiv b \pmod{lcm(m_1, m_2, ..., m_k)}.$$

2.1.2 抽象代数基础

定义 2.9. 设 G 是一个非空集合，“$+$”是 G 上的一个代数运算，即对所有的 $a, b \in G$，有 $a + b \in G$. 如果 G 的运算还满足

(I) 结合律，即对所有的 $a, b, c \in G$，有 $(a + b) + c = a + (b + c)$;

(II) 存在单位元 (Unit Element)，即 G 中有元素 0，使对每个 $a \in G$，有 $0 + a = a + 0 = a$;

(III) 存在逆元 (Inverse)，即对 G 中每个元素 a，存在元素 $b \in G$，使 $a + b = b + a = 0$.

则称 G 关于运算“$+$”构成一个群 (Group)，记作 $(G, +)$，简称群 G.
如果群 G 的运算还满足交换率，即对任意的 $a, b \in G$，有 $a + b = b + a$，则称 G 是一个交换群 (Commutative Group) 或阿贝尔群 (Abelian Group).

定义 2.10. 设 R 是一个非空集合，如果在 R 上定义了两个代数运算“$+$”(称为加法) 和“$\cdot$”(称为乘法)，并且满足:

(I) R 关于加法构成一个交换群;

(II) 乘法结合律成立，即对任意的 $a, b, c \in R$，有

$$(a \cdot b) \cdot c = a \cdot (b \cdot c);$$

(III) 乘法对加法的两个分配率成立，即对任意的 $a, b, c \in R$，有

$$a \cdot (b + c) = a \cdot b + a \cdot c,$$

$$(b + c) \cdot a = b \cdot a + c \cdot a.$$

则称 $(R, +, \cdot)$ 为一个环 (Ring)，简称环 R.
如果环 R 的乘法还满足交换率，则称 R 为交换环 (Commutative Ring).

定义 2.11. 若环 $(Z, +, \cdot)$ 满足如下条件:

(I) Z 是一个交换环;

(II) 无零因子 (Zero Divisior)，即若 $a, b \in Z$ 且 $ab = 0$，则必有 $a = 0$ 或 $b = 0$;

(III) 存在乘法单位元 (Multiplicative Identity)，即存在 $1 \in Z$ 对所有 $a \in Z$ 满足 $a \cdot 1 = 1 \cdot a = a$.

则称环 Z 为整环 (Integral Domain).

定义 2.12. 若整环 $(F, +, \cdot)$ 还满足如下条件:

(I) 存在乘法逆元 (Multiplicative Inverse)，即对除 0 以外的所有 $a \in F$，存在乘法逆元 a^{-1} 满足 $aa^{-1} = a^{-1}a = 1$.

则称 F 为域 (Field).

定理 2.6. *集合 $\mathbf{Z}_n$ 关于模 n 的加法和乘法运算，构成一个交换环.*

定理 2.7. *设 p 为素数，则 $\mathbf{Z}_p$ 是一个含有 p 个元素的有限域.*

2.1.3 重要的结论

2.1.3.1 费马小定理

定理 2.8 (费马小定理)**.** *若 p 是素数，a 是不被 p 整除的正整数，则*

$$a^{p-1} \equiv 1 \pmod{p}.$$

推论 2.9. *若 p 是素数，a 是正整数，则*

$$a^p \equiv a \pmod{p}.$$

2.1.3.2 中国剩余定理

定理 2.10 (中国剩余定理)**.** *设 $m_1, ..., m_k$ 是两两互素的正整数，即对任意 $1 \leq i, j \leq k$ 且 $i \neq j$，有 $gcd(m_i, m_j) = 1$. 那么，对任意整数 $a_1, ..., a_k$，一次同余方程组*

$$x \equiv a_j \pmod{m_j}, 1 \leq j \leq k$$

有唯一解:

$$x \equiv M_1 M_1^{-1} a_1 + ... + M_k M_k^{-1} a_k \pmod{m}$$

其中 $m = m_1 m_2 ... m_k$，$m = m_j M_j (1 \le j \le k)$，$M_j^{-1}$ 是满足

$$M_j M_j^{-1} \equiv 1 \pmod{m_j}, 1 \le j \le k$$

的一个整数.

推论 2.11. 令 $M = \prod_{i=1}^{k} m_i$，其中 m_i 两两互素，即对任意 $1 \le i, j \le k$ 且 $i \ne j$，有 $gcd(m_i, m_j) = 1$. 则任意 $A \in \mathbf{Z}_M$ 可以被唯一地表示为一个 k 元组：

$$A \leftrightarrow (a_1, a_2, ..., a_k)$$

其中 $a_i = A \bmod m_i, 1 \le i \le k$. 并且

(I) 以上映射为从 $\mathbf{Z}_M$ 到 $\mathbf{Z}_{m_1} \times \mathbf{Z}_{m_2} \times ... \times \mathbf{Z}_{m_k}$ 的一一映射。即对任意 $0 \le A \le M$，存在唯一的 k 元组 $(a_1, a_2, ..., a_k)$ 与之对应；反之，对任意 k 元组 $(a_1, a_2, ..., a_k)$，存在唯一 $A \in \mathbf{Z}_M$ 与之对应.

(II) $\mathbf{Z}_M$ 上实行的运算（加法、减法和乘法）等价于对其 k 元组的每个元实行相同运算，即

$$(A + B) \bmod M \leftrightarrow ((a_1 + b_1) \bmod m_1, ..., (a_k + b_k) \bmod m_k),$$

$$(A - B) \bmod M \leftrightarrow ((a_1 - b_1) \bmod m_1, ..., (a_k - b_k) \bmod m_k),$$

$$(A \times B) \bmod M \leftrightarrow ((a_1 \times b_1) \bmod m_1, ..., (a_k \times b_k) \bmod m_k).$$

2.2 计算复杂度与安全性

2.2.1 时间复杂度

定义 2.13. 在计算复杂度理论中，若一个问题的计算时间 $T(n)$ 不大于问题大小 n 的多项式倍数，则称之为多项式时间 (Polynomial Time).

定义 2.14. 若一个问题的计算时间 $T(n)$ 超过了问题大小 n 的多项式倍数，则称之为超多项式时间 (Super-Polynomial Time).

多项式时间的问题被认为是可以有效计算的。而超多项式时间的问题，随着问题规模的增加，其计算时间的增长会比多项式时间快得多；因此若问题规模很大，超多项式时间的问题是无法有效计算的。

通常用大 O 符号来表示问题的复杂度：

定义 2.15. 设 $f(n), g(n)$ 为两个函数，如果存在一个常数 $c > 0$，使得 $f(n) \leq c \cdot g(n)$ 成立，则称 $f = O(g)$.

$f = O(g)$ 意味着 f 的增长速度慢于 g。通常 g 会选择简单的函数形式，以简单地表示时间复杂度。

2.2.2 整数分解问题

定理 2.12 (算术基本定理)**.** 任意正整数可唯一地表示为一系列素数的乘积，即设正整数 n，则 n 可唯一地被表示为

$$n = p_1^{k_1} p_2^{k_2} ... p_r^{k_r} = \prod_{i=1}^{r} p_i^{k_i}$$

其中 k_i 为正整数，p_i 为素数并且 $p_1 < p_2 < ... < p_r$.

定义 2.16 (整数分解问题)**.** 给定正整数 n，找出定理2.12中 n 的所有素数因子 p_i ，即为整数分解问题 (The Factoring Problem).

整数分解问题被认为是困难问题，即没有可在多项式时间内计算的算法。目前最快的整数分解算法也需要亚指数时间复杂度。因此当 n 足够大时，整数分解问题被认为是不可解的。

2.2.3 密码学攻击方法

2.2.3.1 唯密文攻击

唯密文攻击 (Ciphertext-Only Attack) 是指攻击者仅仅通过加密后的密文实行的攻击。攻击者试图通过密文得到有关明文或密钥的信息。

定义 2.17 (唯密文攻击)**.** 设 P 为明文，C 为密文.
已知：$C_1, C_2, ..., C_i$.
推导：明文 $P_1, P_2, ..., P_i$；密钥 K；或从 C_{i+1} 计算 P_{i+1} 的算法.

2.2.3.2 已知明文攻击

已知明文攻击 (Known-Plaintext Attack) 中攻击者不仅得到密文，而且还可以得到密文对应的明文。即攻击者可获得若干明文密文对。

定义 2.18 (已知明文攻击)**.** 设 P 为明文，C 为密文.
已知：$(P_1, C_1), (P_2, C_2), ..., (P_i, C_i)$.
推导：密钥 K；或从 C_{i+1} 计算 P_{i+1} 的算法.

2.2.3.3 选择明文攻击

选择明文攻击 (Chosen-Plaintext Attack) 中攻击者除了可以获得明文密文对，还可以选择被加密的明文。即攻击者相当于有一个加密盒子，可以对任意明文输入进行加密得到密文输出。攻击者可以选择特定的明文去加密，以便产生更多关于密钥的信息。

定义 2.19 (选择明文攻击)**.** 设 P 为明文，C 为密文.
已知：$(P_1, C_1), (P_2, C_2), ..., (P_i, C_i)$. 其中 $P_1, P_2, ..., P_i$ 可由攻击者选择.
推导：密钥 K；或从 C_{i+1} 计算 P_{i+1} 的算法.

2.2.3.4 选择密文攻击

选择密文攻击 (Chosen-Ciphertext Attack) 允许攻击者选择不同的密文，并得到对应的解密后的明文。即攻击者相当于有一个解密盒子，可以输入密文获得解密后的明文。

定义 2.20 (选择密文攻击)**.** 设 P 为明文，C 为密文.
已知：$(C_1, P_1), (C_2, P_2), (C_i, P_i)$. 其中 $C_1, C_2, ..., C_i$ 可由攻击者选择.
推导：密钥 K；或从 C_{i+1} 计算 P_{i+1} 的算法.

选择密文攻击分为自适应 (Adaptive) 和非自适应 (Non-Adaptive)。在非自适应的选择密文攻击中，一旦攻击者收到目标密文，就不再获得解密服务。

在自适应选择密文攻击中，攻击者在收到目标密文后，仍可以使用解密服务获得除目标明文以外的明文。即相当于解密盒子一直打开，除了输入为目标密文之外，可以提供任意密文的解密结果。

2.3 隐私同态

2.3.1 隐私同态的定义

定义 2.21 (隐私同态)**.** 设代数系统 U 包含：集合 S，运算 $f_1, f_2, ...$，谓词 $p_1, p_2, ...$，以及常量 $s_1, s_2, ...$，记作

$$U =< S; f_1, f_2, ...; p_1, p_2, ...; s_1, s_2, ... >;$$

设代数系统 C 包含：集合 S'，运算 $f'_1, f'_2, ...$，谓词 $p'_1, p'_2, ...$，以及常量 $s'_1, s'_2, ...$，记作

$$C =< S'; f'_1, f'_2, ...; p'_1, p'_2, ...; s'_1, s'_2, ... >;$$

定义 $\phi : S' \to S$，以及 $\phi^{-1} : S \to S'$. 若 ϕ 满足：

(i) $(\forall i)(a, b, c, ...) \in S', f'_i(a, b, ...) = c \Rightarrow f_i(\phi(a), \phi(b), ...) = \phi(c)$

(ii) $(\forall i)(a, b, c, ...) \in S', p'_i(a, b, ...) \equiv p_i(\phi(a), \phi(b), ...)$

(iii) $(\forall i)\phi(s'_i) = s_i$

则称上述由 U, C, ϕ, ϕ^{-1} 构成的体系 H 为一个隐私同态 (Privacy Homomorphism).

在以上定义中，U 是私密的，而 C 是公开的。假设一个用户希望应用隐私同态计算 $f_1(d_1, d_2)$，用户将 $\phi^{-1}(d_1), \phi^{-1}(d_2)$ 发送给委托方，并要求委托方计算 $f'_1(\phi^{-1}(d_1), \phi^{-1}(d_2))$；委托方计算完成后，将结果发送给用户，用户计算

$$\phi(f'_1(\phi^{-1}(d_1), \phi^{-1}(d_2))) = f_1(\phi^{-1}(d_1), \phi^{-1}(d_2)) = f_1(d_1, d_2).$$

隐私同态除了需要满足上述要求外，在文献 [1] 中作者还提出隐私同态需要满足如下性质：

1. ϕ 和 ϕ^{-1} 要能被有效计算；
2. C 中的运算 f'_i 和谓词 p'_i 要能被有效计算；
3. $\phi^{-1}(d_i)$ 所需空间不应比 d_i 所需空间增加过大；
4. 给定多个不同 $\phi^{-1}(d_i)$ 的值不能有效求出 ϕ；

5. 给定多组不同 d_i 和 $\phi^{-1}(d_i)$ 的值不能有效求出 ϕ;

6. 通过 C 中的信息包括运算和谓词，不能有效地求出 ϕ.

其中性质 1、2 和 3 保证隐私同态能有效地计算；性质 4 等价于抗唯密文攻击；性质 5 等价于抗已知明文攻击；性质 6 会在第2.3.3节中讨论。

此外需要主意的是，ϕ 和 ϕ^{-1} 虽然作用相反，但并不一定是互逆的函数。ϕ 相当于解密函数 (Decryption Function)，在后文中也用 D 表示；ϕ^{-1} 相当于加密函数 (Decryption Function)，在后文中也用 E 表示。

2.3.2 隐私同态的例子

经典的 RSA 密码体制就满足模 n 下的乘法同态。假设有两个用相同公钥 e 加密的密文

$$c_1 = m_1^e \mod n$$

$$c_2 = m_2^e \mod n$$

则

$$c_1 c_2 \mod n = m_1^e m_2^e \mod n = (m_1 m_2)^e \mod n$$

解密后恰好得到 $m_1 m_2$.

2.3.3 安全性特点

隐私同态可以对密文进行运算，这种性质给实际应用带来了好处，也造成了隐私同态的安全性缺陷。从隐私同态的一开始，在文献 [1] 中，作者就指出顺序保留的隐私同态是不安全的。

定理 2.13. *对于一个隐私同态，如果攻击者可以取得明文空间中任意常量的加密结果，并且可以通过谓词 $\leq$ 比较密文，则该隐私同态可被唯密文攻击.*

证明. 设目标明文为 a，对应密文为 $a' = E(a)$. 则可以通过二分查找算法来确定 a 的值.

首先加密常量 1，用 $1'$ 表示对应的加密结果：$E(1) = 1'$. 继而，根据同态性质，$2' = E(2) = E(1+1) = E(1) \oplus E(1) = 1' \oplus 1'$. 于是可以计算任意 $2'^n$ 的值. 由于谓词 $\leq$ 可用，可以找到 $2'^n$ 满足 $2'^{n-1} \leq a' \leq 2'^n$.

然后比较 $a' \leq 2'^{n-1} \oplus 2'^{n-2}$. 如果成立，则 a' 的值被锁定在区间 $2'^{n-1} \leq a' \leq 2'^{n-1} \oplus 2'^{n-2}$，继续比较 $a' \leq 2'^{n-1} \oplus 2'^{n-3}$；如果不成立，则试探比较 $a' \leq 2'^{n-1} \oplus 2'^{n-2} \oplus 2'^{n-3}$.

重复该过程，最终找到 $a' = 2'^{i} \oplus 2'^{j} \oplus ... \oplus 2'^{m}$. 根据同态性，有 $a' = 2'^{i} \oplus 2'^{j} \oplus ... \oplus 2'^{m} = E(2^i) \oplus E(2^j) \oplus ... \oplus E(2^m) = E(2^i + 2^j + ... + 2^m)$. 于是 $a = 2^i + 2^j + ... + 2^m$. □

在文献 [16] 中，作者证明了包含加法的隐私同态可被选择密文攻击。

定理 2.14. *一个隐私同态如果可实现同态加法，并且密文加法 $\oplus$ 也是算术加法，则该隐私同态可被选择密文攻击.*

证明. 设 m_1, m_2 为明文，E 为加密函数，D 为解密函数，则根据定义

$$D(E(m_1) \oplus E(m_2)) = D(E(m_1) + E(m_2)) = m_1 + m_2.$$

考虑如下一组用二进制表示的密文:

$$\begin{aligned} E(a_1) &= [1, 0, 0, ..., 0] \\ E(a_2) &= [0, 1, 0, ..., 0] \\ &\vdots \\ E(a_m) &= [0, 0, 0, ..., 1] \end{aligned}$$

在选择密文攻击下，攻击这可以得到 $a_1, a_2, ..., a_m$ 的值.

设目标明文为 m，密文为 $c = E(m)$. 计算

$$c_i = \begin{cases} 0 & \text{if } c \text{ AND } E(a_i) = 0 \\ 1 & \text{otherwise} \end{cases}$$

$i = 1, 2, ..., m$，其中 AND 为位运算“与”，即 c_i 代表密文 c 的二进制表示中第 i 位是否为 1. 于是密文 c 可被表示为

$$c = \sum_{i=1}^{m} c_i E(a_i)$$

根据同态性，可以计算 m

$$m = D(\sum_{i=1}^{m} c_i E(a_i)) = \sum_{i=1}^{m} c_i a_i.$$

□

因此对于此类隐私同态（包括后文中要提到的所有隐私同态）的最高抗攻击级别为抗选择明文攻击。

第三章　隐私同态方案

3.1　Rivest 的方案

Rivest, Adleman, Dertouzos 在最早提出隐私同态的方案时，在文献 [1] 中给出了五个隐私同态方案。这些方案后来均被证明在唯密文攻击或已知明文攻击下是不安全的。以下要介绍的是其中最典型的一个方案，虽然该方案也已被证明是不安全的，但作为最早提出的隐私同态方案之一，其思想被后来的很多方案所采用，文献 [3] 中的方案实际上就是对该方案的复杂化改进；而且对该方案的攻击，也具有典型性。所以有必要先介绍一个这个早期的隐私同态方案。

3.1.1　方案描述

该方案是一个 $U =< \mathbf{Z}_n; +, -, \times >$ 的隐私同态。
选择两个大素数 p, q，并计算它们的乘积 $m = pq$.
m 为公开参数；而 p, q 保密，作为密钥。
加密： 对于明文空间 $\mathbf{Z}_n$ 中的明文 m，计算

$$c_1 = a \bmod p$$

$$c_2 = a \bmod q$$

输出密文 (c_1, c_2).
解密： 根据中国剩余定理，可计算

$$m = qq'c_1 + pp'c_2 \bmod n$$

其中 $q' \equiv q^{-1} \mod p, p' \equiv p^{-1} \mod q$. 输出明文 m.

给方案的正确性可直接从定理2.11结论 *(I)* 得出；其同态性可从定理2.11结论 *(II)* 得出。

3.1.2 密码分析

该方案存在一个明显的安全性漏洞：如果 m 较小，$m < p$ 或 $m < q$，则 c_1 或 c_2 本身就是明文，相当与没有加密。而又不能用填充算法对 m 进行填充，因为一旦对 m 进行填充，同态性将不再保持。

当 m 足够大时，$m > p, m > q$，由于模运算的置乱作用，当 p, q 未知时，将不能从 (c_1, c_2) 直接恢复 m。攻击者若想得到 p, q，就需要分解公开参数 n；而大整数分解问题被认为是不可解的，所以攻击者不可能获得 p, q 的值，从而破解密文。

虽然攻击者不能分解大整数 n，但仍可在已知明文攻击下以很大概率找到 p, q。攻击方法如下：设攻击者已知明文 $m_1, m_2, ..., m_r$，以及对应的密文 $E(m_i) = (c_i, d_i), i = 1, 2, ..., r$。由于只要知道了 p，就可以很容易地计算 q，所以只需讨论攻击 p 的方法。由定义2.3知

$$p \mid c_i - m_i, i = 1, 2, ..., r$$

计算最大公约数

$$\hat{p} = gcd\{c_i - m_i : i = 1, 2, ..., r\}$$

因 $p \mid c_i - m_i$，则必有 $p \mid \hat{p}$. 事实上，即使对于较小的 r 值，也有很大概率使得 $\hat{p} = p$. 即使 $\hat{p} \neq p$，每得到一对新的明文密文 $(m, (c, d))$，攻击者只要计算 $gcd(c - m, \hat{p})$，就会更接近密钥 p 的值. □

3.2 Ferrer 的方案

该方案由 Josep Domingo-Ferrer 提出[3]，是对3.1中方案的改进，试图使其可抗已知明文攻击。

3.2.1 方案描述

初始化：选择两个大素数 p, q，计算 $n = pq$；选择一个正整数 d；随机选择 $r_p \in \mathbf{Z}_p^*, r_q \in \mathbf{Z}_q^*$. 公共参数为 (d, n)，私密参数为 (p, q, r_p, r_q). （n 可以不公开，但公开 n 可以提高密文运算效率）

加密：对于明文 $x \in \mathbf{Z}_n$，随机将其拆分为 $x_1, x_2, ..., x_d$ 并满足

$$x = \sum_{i=1}^{d} x_i \mod n, x_i \in \mathbf{Z}_n$$

计算向量

$$X = ([x_1 r_p \bmod p, x_1 r_q \bmod q], ..., [x_d r_p^d \bmod p, x_d r_q^d \bmod q])$$

输出 X 为密文.

解密：将密文向量 X 中每个二元对 $[x_j r_p^j \bmod p, x_j r_q^j \bmod q]$，乘以 $[r_p^{-j} \bmod p, r_q^{-j} \bmod q]$，于是得到

$$([x_1 \bmod p, x_1 \bmod q], ..., [x_d \bmod p, x_d \bmod q])$$

然后将所有二元对相加，得到

$$[x \bmod p, x \bmod q] = \sum_{i=1}^{d} [x_i \bmod p, x_i \bmod q]$$

然后利用中国剩余定理计算出明文 x.

同态性：该方案是一个 $U =< \mathbf{Z}_n; +, -, \times >$ 的隐私同态。为了说明其同态性，设 x, y 为明文，对应的密文为 $X = E(x), Y = E(y)$，即

$$X = (X_1, ..., X_d) = ([x_1 r_p \bmod p, x_1 r_q \bmod q], ..., [x_d r_p^d \bmod p, x_d r_q^d \bmod q])$$

$$Y = (Y_1, ..., Y_d) = ([y_1 r_p \bmod p, y_1 r_q \bmod q], ..., [y_d r_p^d \bmod p, y_d r_q^d \bmod q])$$

- 加法和减法：注意到

  $$x + y = \sum_{i=1}^{d} x_i + \sum_{i=1}^{d} y_i = \sum_{i=1}^{d} (x_i + y_i)$$

  所以只需将密文向量中相同次数的分量（即向量分量中 r_p 或 r_q 的次数相同）相加，即定义

  $$X \oplus Y = (X_1 + Y_1, ..., X_d + Y_d)$$

  解密后即得到 $x + y$ 的结果. 减法同理可得.

- 乘法：定义密文空间中的乘法为

  $$Z = X \otimes Y = (Z_1, Z_2, ..., Z_{2d})$$

其中

$$Z_k = \begin{cases} [0,0] & \text{if } k = 1 \\ \sum_{i+j=k} X_i Y_i & \text{if } k = 2, ..., 2d \end{cases}$$

于是，当 $k \geq 2$ 时，

$$Z_k = \left[\left(\sum_{i+j=k} x_i y_j \right) r_p^k, \left(\sum_{i+j=k} x_i y_j \right) r_q^k \right]$$

所以每个分量 Z_k 解密后得到 $\sum_{i+j=k} x_i y_j$ 的值. 注意到

$$xy = \left(\sum_{i=1}^{d} x_i \right) \left(\sum_{i=1}^{d} y_i \right) = \sum_{k=2}^{2d} \sum_{i+j=k} x_i y_j$$

所以对 Z 解密后恰好得到 xy 的值.

3.2.2 密码分析

Domingo-Ferrer 的方案本质上是对3.1中方案的复杂化变形，通过对明文 x 的随机性拆分，和引入随机数 r 来增强其安全性。形式上采用多项式的形式，因为多项式环同构于整数环，所以得以保持其同态性质。尽管如此，该方案还是被 Cheon, Kim, Nam 破解。在文献 [4] 中，作者证明：

当公共参数 n 可知时，该方案可用 $(d+1)$ 个明文密文对破解；当 n 不可知时，该方案可用 $2(d+1)$ 个明文密文对破解.

攻击方法概述如下：若 n 已知，设有 $d+1$ 组明文密文对，记为

$$\{(x_i, Y_i, Z_i) \mid i = 1, ..., d+1\}$$

其中

$$x_i = x_{i_1} + ... + x_{i_d} \mod n$$

$$Y_i = (y_{i_1}, ..., y_{i_d}) = (x_{i_1} r_p \bmod p, ..., x_{i_d} r_p^d \bmod p)$$

$$Z_i = (z_{i_1}, ..., z_{i_d}) = (x_{i_1} r_q \bmod q, ..., x_{i_d} r_q^d \bmod q)$$

令 $t = r_p^{-1}$ (t 对攻击者未知)，则有

$$f_i(t) = -x_i + y_{i_1}t + ... + y_{i_d}t^d \equiv 0 \mod p, i = 1, ..., d+1$$

于是对 $d+1$ 组明文密文对有如下矩阵方程成立：

$$\begin{pmatrix} x_1 & y_{1_1} & y_{1_2} & \cdots & y_{1_d} \\ x_2 & y_{2_1} & y_{2_2} & \cdots & y_{2_d} \\ \vdots & \vdots & \vdots & & \vdots \\ x_{(d+1)} & y_{(d+1)_1} & y_{(d+1)_2} & \cdots & y_{(d+1)_d} \end{pmatrix} \begin{pmatrix} -1 \\ t \\ \vdots \\ t^d \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \pmod p \qquad (3\text{–}1)$$

因为上述方程有非平凡解，所以方程中的系数矩阵 A 的行列式满足

$$det(A) \equiv 0 \mod p$$

所以 $p \mid det(A), p \mid n$. 计算

$$\hat{p} = gcd(det(A) \bmod n, n)$$

则有很大概率 $p = \hat{p}$.

一旦 p 被求出，可通过如下方法求出 r_p：因为 $t = r_p^{-1}$ 是方程 $f_i(t) \equiv 0 \mod p$ 的解，所以多项式 $f_i(t)$ 必含因子 $(t - r_p^{-1})$. 计算所有 $f_i(t)$ 的最大公约多项式

$$g(t) = gcd(f_1(t), ..., f_{d+1}(t))$$

则 $(t - r_p^{-1}) \mid g(t)$ 并且有很大概率得出 $(t - r_p^{-1})$. 通过 $(t - r_p^{-1}) \equiv 0 \mod p$ 解出 t，则 $r_p = t^{-1} \bmod p$.

若 n 未知，也可以通过类似的方法破解。设攻击者有 $2(d+1)$ 组明文密文对，则可以得到两个方程 3–1中的系数矩阵，记为 A_1, A_2. 于是计算

$$\hat{p} = gcd(det(A_1), det(A_2))$$

则有很大概率 $p = \hat{p}$. □

3.3 新的方案

上文中介绍了两个典型的代数隐私同态，但他们都是只能实现环上运算的隐私同态（即加法、减法、乘法），而不能实现域上的全部运算（即加法、减法、乘法、乘法求逆）。事实上，目前被提出的隐私同态只有 [5] 中的方案可实现域上的全部运算，但是它的安全性很弱。

本章描述了一个新的隐私同态算法，有着极简单的数学形式，但是可实现域上所有运算，即加法、减法、乘法和求逆。但是和 [5] 中的方案一样，本章中的方案安全性也很弱，不能抵抗已知明文攻击。虽然如此，但相信该方案的提出也并非毫无意义。

本章首先描述该算法，证明其同态性质；给出一个简单的数字实例；然后介绍已知明文的攻击方法；最后提出可能的改进，试图使其可抗已知明文攻击。

3.3.1 方案描述

初始化：选择两个大素数 p, p'，且 $p < p'$，计算 $n = pp'$. 其中 p, p' 为私密参数，n 为公开参数.

加密：对于明文 $x \in \mathbf{Z}_p$，计算密文

$$y = E(x) = x^p \mod n.$$

解密：对于密文 $c \in \mathbf{Z}_n$，计算明文

$$x = D(x) = y \bmod p.$$

下面的定理证明了该算法的正确性：

定理 3.1. *对于所有 $x \in \mathbf{Z}_p$，有 $D(E(x)) = x$.*

证明. 根据定义，有

$$y \equiv x^p \mod n$$

因 $p \mid n$，由定理2.3 得

$$y \equiv x^p \mod p$$

由费马小定理2.8得

$$y \equiv x^p \equiv x \mod p$$

因 $x < p$，故

$$x = y \bmod p.$$

□

所以，当 p 已知时，可以很方便地解密；而当 p 未知时，则不能求出明文 x。攻击者若想求出 p，则需要分解 n，然而分解大整数被认为是困难的（见2.2.2节）。

下面证明其同态性：

定理 3.2. *设明文 $x, y \in \mathbf{Z}_p$，则以下性质成立：*

(1) 若 $x + y \in \mathbf{Z_p}$，则 $x + y = D(E(x) + E(y))$；

(2) 若 $x - y \in \mathbf{Z_p}$，则 $x - y = D(E(x) - E(y))$；

(3) 若 $xy \in \mathbf{Z_p}$，则 $xy = D(E(x)E(y))$；

(4) $x^{-1} = D(E(x)^{-1})$.

证明.

(1) 根据定义，

$$E(x) + E(y) = x^p \bmod n + y^p \bmod n \equiv x^p + y^p \mod n$$

因为 $p \mid n$，所以

$$E(x) + E(y) \equiv x^p + y^p \equiv x + y \mod p$$

因为 $x + y \in \mathbf{Z_p}$，所以 $x + y = D(E(x) + E(y))$.

(2) 与 (1) 同理可证.

(3) 根据定义，

$$E(x)E(y) = (x^p \bmod n)(y^p \bmod n) \equiv x^p y^p \equiv (xy)^p \mod n$$

于是

$$E(x)E(y) \equiv (xy)^p \equiv xy \mod p$$

若 $xy \in \mathbf{Z_p}$，$xy = D(E(x)E(y))$.

(4) 因为 $x < p$，所以 $gcd(x, p') = 1$；进一步地，$gcd(x, n) = 1$，所以 $E(x)$ 模 n 的逆必存在. 根据定义，

$$E(x)^{-1} = (x^p)^{-1} \equiv \left(x^{-1}\right)^p \mod n$$

因 $p \mid n$，有

$$E(x)^{-1} \equiv \left(x^{-1}\right)^p \equiv x^{-1} \mod p$$

即 $x^{-1} = D(E(x)^{-1})$.

□

该加密算法可以被有效计算，利用平方法可以快速地计算 $x^p \bmod n$ 的值，即按下式递归计算：

$$x^y \bmod n = \begin{cases} \left(x^{\lfloor y/2 \rfloor}\right)^2 \bmod n & \text{if } y \mid 2 \\ x \cdot \left(x^{\lfloor y/2 \rfloor}\right)^2 \bmod n & \text{if } y \nmid 2 \end{cases}$$

其中每一步的计算时间是 $O(\log^2 n)$，递归深度为 $O(\log p)$，所以算法总运行时间为 $O(\log^2 n \log p)$. 对于空间复杂度，若选 p' 为与 p 相同长度的素数，则密文大约为明文的 2 倍，若用 l 表示明文长度，则密文长度为 $O(2l)$.

该算法的代码实现可参考附录B。

3.3.2 简单的例子

下面将给出一个简单的例子来说明3.3.1节中的算法。该例子中用到的数据十分小，不符合实际情况，只是为了说明算法的运行过程。

假设用户要计算 $(2+3+4)\times 2\times 3^{-1}$ 的值。
设 $p=17, q=19$，则 $n=pq=323$，分别加密所有数据：

$$E(2)=2^{17} \bmod 323 = 257$$

$$E(3)=3^{17} \bmod 323 = 241$$

$$E(4)=4^{17} \bmod 323 = 157$$

用户将 $E(2), E(3), E(4)$ 和 $n=323$ 发送给代理计算方。

代理计算方收到 $E(2), E(3), E(4)$ 及 $n=323$，计算

$$(E(2)+E(3)+E(4))E(2)E(3)^{-1} = (257+241+157)\times 257\times 241^{-1} \equiv 23 \mod 323$$

将计算结果 23 发送给用户。

用户收到加密结果 23，用私钥 $p=17$ 解密，得到计算结果 $23 \bmod 17 = 6$。

3.3.3 安全性

该算法的实质是将明文 x 扩大并通过取模来置乱，从而达到加密的目的；使用 p 来作为幂次以便通过费马小定理解密，n 的作用是保护 p 并且可通过模 n 的结果得到模 p 的结果。但该方案只能抗唯密文攻击，在已知明文的情况下，很容易用类似3.1.2节中的方法来求得 p。

设有 r 组明文密文对：$(x_1, y_1), (x_2, y_2), ..., (x_r, y_r)$. 根据3.3.1中的定义，有

$$y_i \equiv x_i \mod p, i = 1, 2, ..., r$$

所以 $p \mid y_i - x_i$，计算最大公约数

$$\hat{p} = gcd\{y_i - x_i : i = 1, 2, ..., r\}$$

则有很大概率使得 $\hat{p} = p$. □

所以该方案可以很容易被已知明文攻击，只能应用于在对安全性要求较弱，不要求暴露明文的情景下。该方案之所以能用上述方法攻击，究其原因，是由于明文密文和密钥之间形成简单的线性关系:

$$y \equiv x \mod p \Leftrightarrow y - x = kp$$

于是才可以通过求最大公约数的方法来得到密钥 p。下一章节将讨论使其可抗已知明文攻击的可能方法。

3.3.4 可能的改进

首先将不同的隐私同态方案进行复合，可提高其安全性。比如用3.1节中的方案与3.3.1 中方案复合。复合后仍保持同态性。比如若两个方案同时满足加法同态，则复合后的方案也满足加法同态，因为:
若

$$x + y = D_1(E_1(x + y))$$
$$x + y = D_2(E_2(x + y))$$

则

$$\begin{aligned} D_2 \circ D_1(E_1 \circ E_2(x + y)) &= D_2(D_1(E_1(E_2(x) + E_2(y)))) \\ &= D_2(E_2(x) + E_2(y)) = D_2(E_2(x + y)) \\ &= x + y. \end{aligned}$$

下面就是一个复合的例子，为了方便说明，现定义如下符号:
设 $\mathcal{F}$ 为目标计算函数，即对于输入 $x_1, x_2, ..., x_k$，$\mathcal{F}(x_1, x_2, ..., x_k)$ 返回目标计算结果，并且 $\mathcal{F}$ 只包含可同态执行的运算.
记 $L(x)$ 为 x 的比特长度. 定义 α 为扩张系数 (Expanding Factor)，即 α 满足 $\alpha \cdot max\{L(x_i)\} \geq L(\mathcal{F}(x_1, x_2, ..., x_k))$. 容易证明，$\alpha = \alpha(\mathcal{F})$ 是可以被有效计算的.

初始化: 设 d 为输出密文维数，$m = m_1 m_2 ... m_d$，其中 $m_1, m_2, ..., m_d$ 两两互素，

即对于任意 $1 \leq i, j \leq d$ 且 $i \neq j$，有 $gcd(m_i, m_j) = 1$；且 $m_1, m_2, ..., m_d$ 最好长度相当，即 $L(m_i) \approx L(m_j), 1 \leq i, j \leq d$. 设 $\alpha = \alpha(\mathcal{F})$ 为扩张系数. 选取素数 $p \geq \alpha \cdot max\{L(m_i)\}$，素数 $p' > p$，令 $n = pq$. 其中 n 为公开参数，p, p', m 及 $m_1, ..., m_d$ 均为私密参数.

加密:

(P1) 对于明文 $x \in \mathbf{Z}_m$，分别计算

$$x_i = x \bmod m_i, \text{ for } i = 1, ..., d.$$

(P2) 对每个 x_i，计算

$$y_i = x_i^p \bmod n, \text{ for } i = 1, ..., d$$

输出 $(y_1, y_2, ..., y_d)$ 为密文.

解密:

(P1) 对与密文 $(y_1, y_2, ..., y_d)$，对每个 y_i 计算

$$x'_i = y_i \bmod p, \text{ for } i = 1, ..., d$$

于是得到 $(x'_1, x'_2, ..., x'_d)$.

(P2) 对于每个 x'_i，计算

$$x_i = x'_i \bmod m_i, \text{ for } i = 1, ..., d$$

根据中国剩余定理计算

$$x = CRT(x_1, x_2, ..., x_d)$$

则 x 为明文.

根据中国剩余定理和3.3.1所证结论，知该算法的正确性；并且易证该算法满足加法、减法和乘法的同态性。对密文的加法、减法和乘法就是对密文向量中的每个分量进行单独操作。设 Y_1, Y_2 为密文:

$$Y_1 = (y_{1,1}, ..., y_{1,d})$$

$$Y_2 = (y_{2,1}, ..., y_{2,d})$$

定义

$$\begin{aligned} Y_1 \oplus Y_2 &= (y_{1,1} + y_{2,1}, ..., y_{1,d} + y_{2,d}) \\ Y_1 \ominus Y_2 &= (y_{1,1} - y_{2,1}, ..., y_{1,d} - y_{2,d}) \\ Y_1 \otimes Y_2 &= (y_{1,1}y_{2,1}, ..., y_{1,d}y_{2,d}). \end{aligned}$$

对于长度为 $l = L(x)$ 的明文，平均每个 x_i 的长度为 l/d，而 $p \approx \alpha \cdot l/d$，所以密文向量中每个分量的长度为 $O(2\alpha l/d)$，所以整个密文向量占用空间 $O(2\alpha l)$。可见密文扩张与 α 的大小有密切关系。在上述算法中，之所以使用扩张系数 α 是因为，密文向量中的分量会在计算中扩张，比如要计算 $Y_1 \otimes Y_2$ 则对于密文中的每个分量需要计算 $y_{1,i}y_{2,i}$；在解密中，设 (P1) 后得到 $x'_{1,i}x'_{2,i}$。要保证经中国剩余定理计算后得到 xy 的值，则必须正确得到 $x_{1,i}x_{2,i} \bmod m_i$ 的值。由于模数不同，要从 $y_{1,i}y_{2,i} \bmod p$ 得到 $x_{1,i}x_{2,i} \bmod m_i$ 的值，必须保证 $x'_{1,i}x'_{2,i} < p$。否则将不能正确恢复明文。可见，α 的值要随着计算的复杂性的增加而增加。也就是说，密文的扩张会随着计算复杂性的提高而越来越大；这也是该方案的局限之一。

再来看一下该方案的安全性。对于明文密文对 $(x, (y_1, y_2, ..., y_d))$，由加密过程得

$$x_i \equiv x \mod m_i \Leftrightarrow x_i - x = rm_i$$

$$y_i \equiv x_i \mod p \Leftrightarrow y_i - x_i = sp$$

其中 $r, x \in \mathbf{Z}$，所以

$$y_i - x = rm_i + sp$$

于是明文密文和密钥之间的关系变得复杂，3.3.3节中的攻击方法将不再成立。是否存在其他已知明文攻击尚不得知，但是至少目前看来，该方案会比3.3.1中方案的安全性有所提高；付出的代价是乘法求逆将不再成立，而且密文扩张更大。

除了与其他隐私同态复合，还可以试想通过更复杂的明文密文形式来提高安全性。事实上，3.3.1中方案是受启发于下面这个定理：

定理 3.3. *在特征为素数 p 的交换环 R 中，如下公式成立：*

$$(a+b)^p = a^p + b^p$$

其中 $a, b \in R$.

这表明在特征为 p 的交换环 R 上，函数 $\phi(x) = x^p$ 是加法同态的，而 $\phi(x)$ 显然又是乘法同态的，也就是说 $\phi(x)$ 是代数同态的。而这一性质在一般情况下是不能成立的。这也启发了只要能构造交换环 R，就可以利用环上这一性质构造代数隐私同态。3.3.1中方案是构造在 $\mathbf{Z}_p$ 上的，如果使用更复杂的交换环，也许可以增加安全性。比如选择域 F_{p^m}，明文密文都被表示成 $F_p[x]$ 多项式的形式。但由于时间仓促，该方法没有被深入探究。但可以相信，这很可能是一种新的隐私同态形式。

第四章　隐私同态的应用

4.1　计算代理与数据代理

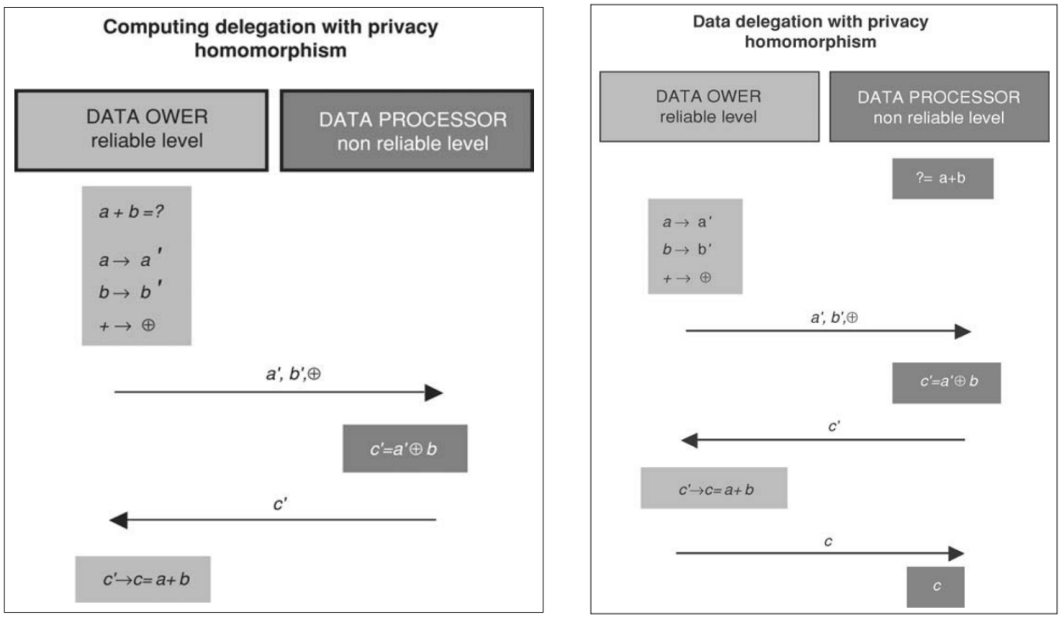
隐私同态的主要应用之一就是计算代理 (Computing Delegation)。计算代理是指数据所有方 (Data Owner) 委托数据处理方 (Data Processor) 对指定数据进行计算，其中只有数据所有方有权查看原始数据及计算结果，数据处理方无权查看原始数据及计算结果。图4.1(a)演示了计算代理的过程。[1]

计算代理在实际生活中有着广泛的应用。比如一个机构要对它的机密数据进行处理，但是该机构又不具备处理数据的能力；该机构可以把数据交给有计算能力的计算中心来处理，但同时还要保证机密数据不被泄露。应用隐私同态可以很好地解决这个安全代理计算问题。例如，一些医学研究机构需要分析病人的数据来进行研究，而这些机构往往计算能力很弱，于是它把数据交给有足够计算能力的第三方来处理，但是病人的数据属于机密，不能泄露。这时该机构可以使用隐私同态对病人数据进行加密，然后交给第三方去处理；第三方利用隐私同态提供的同态运算处理数据，并将加密后的结果返回；该机构收到返回结果并解密得到计算结果。这个过程中，计算中心只负责计算，而无法知晓任何关于机密数据的信息。

随着互联网的发展，特别是云计算的发展，对安全计算代理的需求变得越来越普遍。用户希望把自己的数据放到云端进行计算，但又不想泄露自己的隐私数据。比如有一个互联网上计算税收的应用，一面从政府获得最新的税收标准，一面接受用户的请求，根据用户的收入计算应缴税额。用户希望通过该应用计算自己需要缴纳的税款，但又不希望暴露自己的收入。这时用户可以使用隐私同态来加密自己的收入，并发送给云端；云端根据加密的收入数据计算出应缴纳税额，返回给用户，由于数据一直处在被加密状态，所以云端无法得知用户的收入数据；当用户收到数据后解密之，就可以得到自己应缴纳的税额。

隐私同态的另一个应用的是数据代理 (Data Delegation)。数据代理是指数据处理方需要处理数据所有方的数据并获知计算结果。数据代理和计算代理的区别在于，在计算代理中，数据处理方只处理数据，而不知道计算结果的明文；

[1]图片来源：R. Limbek, P. Sziklai, Privacy homomorphisms, 2004. http://www.infocommunications.hu



(a) 计算代理　　(b) 数据代理

图 4–1 计算代理与数据代理

而在数据代理中，数据处理方需要知晓计算结果的明文。图4.1(b)演示了数据代理的过程。

数据代理的例子也有很多，比如在一些联邦国家中，每个州的机构只拥有本州的居民和公司的数据，而无权知道其他州的数据；联邦机构则拥有所有州的数据。若一个州的机构需要分析其它州的数据，只能经由联邦的机构，由联邦的机构分析相关数据，并返回结果给需要的州机构。但是这样做，联邦机构承担了每个州机构的数据分析任务，这会极大增加联邦机构的负担。一个更好的解决方法是，联邦机构把其它州的数据代理给需要的州机构处理。利用隐私同态可以解决类似的数据代理问题。联邦机构用隐私同态加密所有州的数据，发送给需要分析数据的州机构；该州机构利用隐私同态提供的同态操作进行相应计算，并将加密结果发送给联邦机构；联邦机构解密后将明文结果返回给州机构。

在计算代理的情况下，由于数据处理方不知道原始数据及计算结果的明文，只知道加密后的数据，所以只要求所使用的隐私同态抗唯密文攻击即可；而在数据代理的情况下，由于数据处理方知道计算结果的明文，也就是说数据

处理方拥有计算结果的明文密文对，为了不让数据处理方破解加密数据，所以要求所使用的隐私同态抗已知明文攻击。

在某些只需要进行同态算术运算（四则运算）的应用场景中，比如常规统计计算，就可以应用本文中提出的隐私同态算法。

4.2 实现方案

本节将例举一个利用隐私同态来解决统计计算外包的实现方案。数据所有方利用隐私同态来对隐私数据添加扰动 (Pertubation)，然后将扰动数据发送给数据处理方。数据所有方只需要进行轻量级的运算，就可以将隐私数据加密，以及解密数据处理方返回的结果。在该方案中，数据所有方只需要很小的计算能力，而且数据处理方处理的数据不会泄密。

设数据所有方要处理的隐私数据为 $x_1, x_2, ..., x_n$。首先对原始数据添加随机扰动，即对所有 $i = 1, ..., n$，计算 $x_i^* = x_i + \varepsilon_i$，其中 ε_i 是一个不大的随机数。设 E 为隐私同态的加密函数。数据所有方将 $(x_i^*, E(-\varepsilon_i))$ 发送给数据处理方进行计算。

数据处理方收到密文对 $(x_i^*, E(-\varepsilon_i))$ 后，通过对 x_i^* 和 $E(-\varepsilon_i)$ 执行相应的操作即可完成指定的运算请求。因为 $x_i = x_i^* - \varepsilon_i$，可以很方便地推导出基础运算（加法、减法、乘法）所对应的密文操作。设有密文对 $(x^*, E(-\varepsilon_x))$ 和 $(y^*, E(-\varepsilon_y))$，表4–1列出了基础运算所对应的密文操作。

表 4–1 基础运算对应的密文操作

扰动数据	明文扰动	密文扰动
$x^* + y^*$	$-(\varepsilon_x + \varepsilon_y)$	$E(-\varepsilon_x) + E(-\varepsilon_y)$
$x^* - y^*$	$-(\varepsilon_x - \varepsilon_y)$	$E(-\varepsilon_x) - E(-\varepsilon_y)$
$x^* y^*$	$-x^*\varepsilon_y - y^*\varepsilon_x + \varepsilon_x\varepsilon_y$	$x^*E(-\varepsilon_y) + y^*E(-\varepsilon_x) + E(-\varepsilon_x)E(-\varepsilon_y)$

数据处理方计算之后，将对扰动数据计算的结果，和对加密扰动计算的结果同时返回；数据所有方收到后，解密扰动，并将其与加密数据相加，便得到解密的运算结果。其他运算的计算方法可由基础运算推导而出。

第五章　总结与展望

5.1　全文总结

本文介绍了隐私同态的概念及研究现状，着重介绍了两个典型的隐私同态，提出了一个简单的新隐私同态，并简述了隐私同态的应用。

隐私同态（或同态加密）有三种类型：具有同态性的公钥密码、使用对称密钥的代数隐私同态和全同态加密。目前已知的公钥密码能实现的同态计算很有限，无法完成全部算术运算；代数隐私同态能实现较多的同态运算，但是安全性较弱；全同态加密理论上能实现所有的运算，但是由于计算效率低、开销大，距离实际应用还有很远。本文主要分析第二种，即使用对称密钥的代数隐私同态技术，虽然这种隐私同态的安全性低，目前提出的方案均被证明是不安全的，不可抗已知明文攻击；但这种隐私同态是最具实用价值的。

本文分析了两个现有的典型隐私同态方案，虽然它们均被证明可以在已知明文的情况下被攻击，但是它们所使用的数学方法仍具有指导意义。这些隐私同态，就其本质而言，其实是把明文以更复杂而隐秘的形式呈现出来。为了达到加密的目的，往往密文会以复杂的形式出现；但是为了保证同态的性质，密文无论如何复杂，总是保持了明文的一些规律。所以可以这样理解：*在一个能实现多种代数运算的隐私同态中，密文其实是对明文的另一种表示。* 好比二进制是十进制的另一种表示，虽然表现方式不同，但实质相同；但是作为密码算法，这种表示方法要具有加密性质。

之后本文提出一个新的隐私同态方案，鉴于笔者水平有限，加之时间仓促，这只是一个简单的尝试。该方案的数学形式非常简单，安全性也很弱。文中也提出了将其与其他隐私同态模型复合从而增强其安全性的方法。但该方案还远不是一个完整的隐私同态方案。若要使其成为一个成熟的密码方案，还有很多地方需要完善。

最后本文简述了隐私同态的应用。随着云计算的发展，隐私同态的应用类型会越来越多，本文只是简单介绍最直观的应用场景；隐私同态还有很多应用未能提及。

5.2 未来展望

随着对隐私同态的研究越来越多，我们似乎不得不面对这样一个事实：隐私同态所支持的同态运算越多，其安全性就越弱；而安全性越强，其时间复杂度和空间复杂度就会越高。之所以会存在这种瓶颈，是因为，在某种程度上，隐私同态的目标和密码学的目标是相抵触的：加密是为了将明文混淆和置乱，使其难以辨认，其目的在于消除规律；而隐私同态是为了使加密后的密文仍可以运算，其目的是保持一定规律。于是构成了消除规律与保持规律的矛盾。隐私同态的研究中最具挑战性的工作就在于能否解决好这一矛盾。

在隐私同态领域，虽然已经做过很多研究，已经有了很多成果，但还有很长的路要走。是否存在一个既具有安全性、又支持多种代数运算的隐私同态，仍是未来一个十分具有吸引力和挑战性的课题。

附录 A　具有同态性的公钥密码

表 A–1 具有同态性的公钥密码

公钥密码	安全假设	同态运算	密文扩张
RSA	Factoring / RSA Problem	$\times$	1
ElGamal	CDH / DDH	$\times$	2
Benaloh	Weak r’ th Root Problem	$+, -, \times_c$	$\frac{n}{r}$
Naccache-Stern	Factoring / DLP / Weak r’ th Residue Problem	$+, -, \times_c$	≥ 4
Okamoto-Uchiyama	Factoring / p-subgroup Problem	$+, -, \times_c$	3
Paillier	Class[n] / D-Class[n]	$+, -, \times_c$	2
Elliptic Curve Paillier	Subgroup Decision Problem	$+, -, \times_c$	2
Schmidt-Samoa-Takagi	Factoring Problem	$+, -, \times_c$	3
Damg̊ard-Jurik	Class[n^s] / D-Class[n^s]	$+, -, \times_c$	$\frac{s+1}{s}$
Boneh-Goh-Nissim	Subgroup Decision Problem	$+, -, \times_c, \times$(once)	$\frac{n}{r}$

$+, -, \times$ 表示同态加法、减法、乘法，$\times_c$ 表示乘以已知常数；n 表示密文空间的大小；s 表示 Damg̊ard-Jurik 密码中用到的指数；r 表示能有效计算的离散对数问题中指数大小的上限.

附录 B　隐私同态方案的核心代码 (C++)

代码 B.1 大整数运算函数

```
1  //
2  // hlib.h
3  // Implementation of a Privacy Homomorphisms
4  //
5  // Created by Ben on 22/4/13.
6  // Copyright (c) 2013 SJTU. All rights reserved.
7  //
8
9  #ifndef __MyPH__hlib__
10 #define __MyPH__hlib__
11
12 struct BitInt;
13
14 // generate a random number in the range 0..n-1
15 BigInt random_integer(const BigInt& n);
16
17 // generate random prime of bit length b
18 BigInt random_prime (long b);
19
20 // return the smallest prime >= n
21 BigInt next_prime (const BigInt& n);
22
23 // comptute a mod n
24 BigInt mod (const BigInt& a, const BigInt& n);
25
26 // compute the multiplicative inverse of a mod n
27 BigInt inv_mod (const BigInt& a, const BigInt& n);
28
29 // compute a's power of e mod n
30 BigInt power_mod (const BigInt& a, const BigInt& e, const BigInt& n);
31
32 #endif
```

代码 B.2 隐私同态定义类

```
1  //
2  // homo.h
3  // Implementation of a Privacy Homomorphisms
4  //
5  // Created by Ben on 13/4/13.
6  // Copyright (c) 2013 SJTU. All rights reserved.
7  //
8
9  #ifndef MyPH_homo_h
10 #define MyPH_homo_h
11
12 #include "hlib.h"
13 #include <vector>
14
15 class HomoBase
16 {
17 public:
18     virtual void Initialize (const BigInt& cleartextSpace) =0;
19     virtual void HomoEncrypt(const BigInt& cleartext, OUT BigInt& ciphertext) const =0;
20     virtual void HomoDecrypt(const BigInt& ciphertext, OUT BigInt& cleartext) const =0;
21     virtual BigInt GetPublicMod () const = 0;
22 };
23
24 // This sheme supports homomorphic operations over a field:
25 // addition, substraction, multiplication and division.
26 class HomoScheme1: public HomoBase
27 {
28 public:
29
30     HomoScheme1() {
31         Initialize (random_prime(defaultCleartextSpaceLength));
32     }
33
34     void Initialize (const BigInt& cleartextSpace) {
35         m_p = next_prime( cleartextSpace );
36         m_q = next_prime(m_p + 1); m_n = m_p * m_q;
37     }
38
39     void HomoEncrypt(const BigInt& cleartext, OUT BigInt& ciphertext) const
40     { ciphertext = power_mod(cleartext, m_p, m_n); }
41
```

```
42      void HomoDecrypt(const BigInt& ciphertext, OUT BigInt& cleartext) const
43      {  cleartext = mod(ciphertext, m_p); }
44
45      BigInt GetSecretPrime() const { return m_p; }
46      BigInt GetPublicMod() const { return m_n; }
47
48  protected:
49      static const long defaultCleartextSpaceLength = 64;
50      BigInt m_p, m_q, m_n;
51  };
52
53  #endif
```

参考文献

[1] Rivest R L, Adleman L, Dertouzos M L. On data banks and privacy homomorphisms[J]. Foundations of secure computation, 1978, 32(4):169–178.

[2] Brickell E F, Yacobi Y. On privacy homomorphisms[C]//Advances in Cryptology—EUROCRYPT'87. .[S.l.]: [s.n.] , 1988:117–125.

[3] Ferrer J D i. A new privacy homomorphism and applications[J]. Information Processing Letters, 1996, 60(5):277–282.

[4] Cheon J H, Kim W H, Nam H S. Known-plaintext cryptanalysis of the Domingo-Ferrer algebraic privacy homomorphism scheme[J]. Inf. Process. Lett., 2006, 97(3):118–123.

[5] Domingo-ferrer J, Herrera-Joancomartí J. A Privacy Homomorphism Allowing Field Operations on Encrypted Data[C]//in I Jornades de Matemàtica Discreta i Algorísmica. .[S.l.]: [s.n.] , 1998.

[6] Domingo-Ferrer J. A Provably Secure Additive and Multiplicative Privacy Homomorphism*[M]//Information Security.[S.l.]: Springer, 2002:471–483.

[7] Wagner D. Cryptanalysis of an algebraic privacy homomorphism[M]//Information Security.[S.l.]: Springer, 2003:234–239.

[8] Bao F. Cryptanalysis of a provable secure additive and multiplicative privacy homomorphism[M]//International Workshop on Coding and Cryptog- raphy.[S.l.]: [s.n.] , 2003:43–50.

[9] Gentry C. Fully homomorphic encryption using ideal lattices[C]//Proceedings of the 41st annual ACM symposium on Theory of computing. 2009. .[S.l.]: [s.n.] , STOC '09.

[10] Van Dijk M, Gentry C, Halevi S, et al. Fully homomorphic encryption over the integers[M]//Advances in Cryptology–EUROCRYPT 2010.[S.l.]: Springer, 2010:24–43.

[11] Smart N P, Vercauteren F. Fully homomorphic encryption with relatively small key and ciphertext sizes[M]//Public Key Cryptography–PKC 2010.[S.l.]: Springer, 2010:420–443.

[12] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE[C]//Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on. .[S.l.]: [s.n.] , 2011:97–106.

[13] Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping[C]//Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. .[S.l.]: [s.n.] , 2012:309–325.

[14] Gentry C, Halevi S, Smart N P. Fully homomorphic encryption with polylog overhead[M]//Advances in Cryptology–EUROCRYPT 2012.[S.l.]: Springer, 2012:465–482.

[15] Gentry C, Halevi S, Smart N P. Better bootstrapping in fully homomorphic encryption[M]//Public Key Cryptography–PKC 2012.[S.l.]: Springer, 2012:1–16.

[16] Ahituv N, Lapid Y, Neumann S. Processing encrypted data[J]. Communications of the ACM, 1987, 30(9):777–780.

[17] Henry K. The Theory and Applications of Homomorphic Cryptography[J]. 2008.

[18] William S, et al. Cryptography and Network Security, 4/E[M].[S.l.]: Pearson Education India, 2006.

[19] 潘承洞, 潘承彪. 初等数论 [M]. 北京: 北京大学出版社, 1999.

[20] 韩士安, 林磊. 近世代数 [M]. 北京: 科学出版社, 2004.

致　　谢

经过半年的忙碌和工作，本次毕业设计已经接近尾声，作为一个本科生的毕业设计，由于经验的匮乏，难免有许多考虑不周全的地方，如果没有导师的督促指导，以及一起工作的同学们的支持，想要完成这个设计是难以想象的。

在这里首先要感谢我的导师曹珍富老师。曹老师平日里工作繁多，但在我做毕业设计的每个阶段，从确定课题到查阅资料、设计草案的确定和修改、中期检查、后期详细设计等整个过程中都给予了我悉心的指导。我除了敬佩曹老师的专业水平外，他严谨的科学研究精神也是我永远学习的榜样，并将积极影响我今后的学习和工作。

我也要感谢薛庆水老师。在曹老师工作繁忙的时候，薛老师为我细心解答了毕业设计中遇到的各种问题。每周的组会，薛老师了解我们毕设的进度，询问遇到的问题，并根据多年科研的经验，给予细心的解答；除此之外，还教导我们要有严谨的学术态度、点滴积累的治学习惯以及刻苦钻研的精神。

然后还要感谢大学四年来所有的老师，为我打下专业知识的基础；同时还要感谢所有的同学们，正是因为有了他们的支持和鼓励，此次毕业设计才会顺利完成。

最后感谢上海交通大学四年来对我的大力栽培。

英文大摘要

The concept of privacy homomorphism was first introduced by Rivest, Adleman and Dertouzos in 1978. A privacy homomorphism, or homomorphic encryption, is an encryption transformation which allows the encrypted data to be operated on without knowledge of the decryption function. A privacy homomorphism allows the data handler to perform computations on the encrypted data and the data owner gets the computing result through decryption.

The homomorphic properties exist in various cryptosystems. We classify all privacy homomorphism systems into three types: homomorphic public-key cryptosystems, algebraic privacy homomorphisms and fully homomorphic encryptions.

Homomorphic public-key cryptosystems: there are many public-key cryptosystems satisfying homomorphic properties. Popular public-key systems, such as RSA, ElGamal, Paillier, etc. are all homomorphic. For example, RSA is multiplicatively homomorphic; ElGamal is also multiplicative; and Paillier is additively homomorphic. There are many other homomorphic public-key systems which are not listed here. But none of the public-key systems known so far support both homomorphic addition and multiplication. So homomorphic public-key systems can only be used in very simple homomorphic calculations.

Algebraic privacy homomorphisms: privacy homomorphisms which support both homomorphic additions and multiplications are usually called algebraic privacy homomorphisms. So far all known algebraic privacy homomorphisms are symmetric-key cryptosystems. Such systems were first proposed by Rivest, Adleman and Dertouzos in their paper in which the concept of privacy homomorphism was first proposed. However, their systems were proved to be insecure. Doming-Ferrer proposed two such privacy homomorphisms. But they were gradually broken by known-cleartext attacks. In fact, algebraic privacy homomorphisms are generally weak in security. So they are applicable in situations where only low security level is required.

Fully homomorphic encryptions: Fully homomorphic encryption is a very new

research area. In 2009, Craig Gentry proposed the first fully homomorphic encryption scheme based on ideal lattices, which is a break through. After Gentry, many improvements on his scheme were made by different researchers and there were several new fully homomorphic encryptions were proposed. Brakerski and Vaikuntanathan showed another well-known fully homomorphic encryption based on LWE. The emergence of fully homomorphic encryptions is a milestone in the research history of privacy homomorphisms. However, the current fully homomorphic encryptions are not usable in the real world, due to the lack of efficiency.

Among the above three types, we think the algebraic privacy homomorphism is the most practical solution for homomorphic computing. The public-key systems offer too limited operations and the fully homomorphic encryptions are not practical at all. So we focus on symmetric-key algebraic privacy homomorphisms in this paper. And the term *privacy homomorphism* used in this paper all refer to this type of homomorphic systems.

There are some facts about the security of privacy homomorphisms. It is proved that, if a privacy homomorphism preserves order when encrypting, it is insecure against a ciphertext-only attack. And if a privacy homomorphism is additive, it must be insecure again a chosen-cleartext attack. So an additive privacy homomorphism can at most be secure again a known-cleartext attack.

One famous example of privacy homomorphisms is the one based on the Chinese Remainder Theorem: select two secret large primes p, q, and let $n = pq$. Given cleartext $x \in \mathbf{Z}_n$, the encryption is to compute $(x \bmod p, x \bmod q)$ as the ciphertext. And given the ciphertext vector, if p, q are known, use the Chinese remainder theorem to compute cleartext x, which acts as the decryption. This scheme satisfies both additive and multiplicative homomorphisms. And the data handler computes the sum, difference, or product of two ciphertexts by performing the operations componentwise , modulo n. Without knowing p, q, one is not able to decrypt the ciphertexts. If the attacker wants to get p, q from n, he need to factor n;however, this is intractable when n is large. This system is probably the first well-known privacy homomorphism. But it is shown that it can be broken by a known-cleartext attack.

Doming-Ferrer enhanced the above system by making it more complex. The rough

idea is: first break the cleartext x into several parts $x_1, x_2, ..., x_d$, which sum up to x itself. Multiply each component x_i by polynomial r_p^i and r_q^i where r_p, r_q are random parameters. Then modulo the two polynomials with p and q respectively. So the ciphertext appears in the form $([x_1 r_p \bmod p, x_1 r_q \bmod q], ..., [x_d r_p^d \bmod p, x_d r_q^d \bmod q])$. The decryption just takes inverted steps: for the ith component of the ciphertext vector, multiply it by r_p^{-i} and r_q^{-i}, and get $[x_i \bmod p, x_i \bmod q]$; then apply the Chinese Remainder Theorem and recover x_i; sum up all $x_1, ..., x_d$ and recover cleartext x. Only $n = pq$ is made public, p, q, r_p, r_q are all private. The system can deal with homomorphic additions, subtractions and multiplications. Notice that the ciphertext is actually two d-degree polynomials: $x_1 r_p + x_2 r_p^2 ... + x_d r_p^d \bmod p$ and $x_1 r_q + x_2 r_q^2 + ... + x_d r_q^d \bmod q$. So the operations on the ciphertext are just like dealing with polynomial additions, subtractions and multiplications. This system was claimed to be able to withstand known-cleartext attacks, however, later researches show that it still can be broken.

The above two privacy homomorphisms both support homomorphic additions, subtractions and multiplications, i.e. homomorphic operations over rings. However, they do not support homomorphic operations over fields. In fact, seldom privacy homomorphisms support all operations over fields. In this thesis, we propose a new privacy homomorphism, which support all field operations. The approach is rather simple: Choose two secret large primes p, p', where $p' > p$, and let $n = pp'$. For a given cleartext $x \in \mathbf{Z}_p$, compute $y = x^p \bmod n$ as the ciphertext, which is the encryption. And for a given ciphertext $y \in \mathbf{Z}_n$, recover the cleartext by computing $x = y \bmod p$, which is the decryption. If p is unknown, it is not able to recover the cleartext. Since only n is public, if the attacker tries to get p from n, he needs to do factorization. All field operations, i.e. addition, subtraction, multiplication and multiplicative inverse, are available, which is easy to prove by Fermat's Little Theorem. As to the security, unfortunately, the system can only withstand known-cleartext attacks and can be broken by known-cleartext attacks. But we think there are possible ways to make it secure against known-cleartext attacks. One possible way is to combine this scheme with other homomorphic transformations. For example, one may first perform the Chinese Remainder Theorem to the cleartext, and then apply this scheme; or one may first homomorphically map the cleartext to another form (e.g. map integers to polynomials),

and then use this scheme to encrypt. In this way, the security will be enhanced, but it will lower the efficiency.

Through the study of privacy homomorphisms, it seems to be a fact that, the more homomorphic operations a privacy homomorphism supports, the lower its security level will be; and if the security is strengthened, the efficiency will be sacrificed. To some degree, the homomorphic property seems to be incompatible with a crypto algorithm. Because to encrypt something means to make it diffused and confused, the aim is to remove order; while a homomorphism means to preserve some sort of order in the ciphertext. It is a dilemma! Therefore, a good privacy homomorphism is difficult to find; and the biggest challenge in the research of privacy homomorphisms is to deal with this dilemma.

Typical applications of privacy homomorphisms include computing delegation and data delegation. Computing delegation usually happens when a (small) company or organization wants to use external facilities to do calculations on the confidential data. In this case, the data owner sends out the encrypted version of the confidential data; the data handler uses homomorphic operations to do calculations on the encrypted data and returns the encrypted result; the data owner receives the encrypted result and decrypts it to get the real result. In this process, the data handler gets into contact only with encrypted texts, and therefore he can launch only ciphertext-only attacks. So privacy homomorphisms used in this case are only required to withstand ciphertext-only attacks. In the case of data delegation, the data handler not only performs operations on the encrypted data but also requires the decrypted result of the computation, so he possesses cleartext-ciphertext pairs. Therefore, privacy homomorphisms used in this case are required to withstand known-cleartext attacks.

In future work, the creation of a good algebraic privacy homomorphism, i.e. a homomorphic cryptosystem which is both secure and practical, is still a very prominent problem in the research of homomorphic cryptography.