西 南 科 技 大 学

研 究 生 学 位 论 文

基于 RSA 公钥算法的数字签名技术研究

年　　级　2015

姓　　名　高 磊

申请学位级别　硕 士

专　　业　信息与通信工程

指 导 老 师　周金治 副教授

Classified Index: TP309
U.D.C: 621.3

Southwest University

of Science and Technology

Master Degree Thesis

Research on Digital Signature Technology Based on RSA Public Key Algorithm

Grade: 2015
Candidate: Gao Lei
Academic Degree Applied for: Master's Degree
Speciality: Information and Communication Engineering
Supervisor: Associate professor Zhou Jinzhi

May 25, 2018

独 创 性 声 明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得西南科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签 名：高磊　　日 期：2018.6.4

关于论文使用和授权的说明

本人完全了解西南科技大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。
（保密的学位论文在解密后应遵守此规定）

签 名：高磊　　导师签名：周金波　　日 期：2018.6.4

摘　　要

随着互联网与信息化社会的不断发展，网络通信技术的应用领域和应用深度也在不断扩大，这在给人们带来益处的同时，也带来了信息安全方面的挑战。为了解决实际应用中的信息安全问题，许多针对性的技术与方案应运而生，RSA 公钥算法作为其中重要的技术之一，既能满足数据加密的要求，也可以被用作数字签名的方案。基于 RSA 的数字签名技术是互联网交易中用以抵御否认、篡改、伪造等安全问题的重要实现手段。如何提高 RSA 公钥算法的效率并保证算法的安全性，使其在数字签名中得到更深入的应用，是一直以来不断被研究的热点问题。

本文以实现数字签名的 RSA 公钥算法为研究对象，对算法的数学理论基础、安全性和签名效率进行了深入的研究。安全性方面，通过对 RSA 公钥算法的因子分解、选择密文等攻击方式以及如何抵御各种攻击进行分析，在传统 RSA 算法的基础上，提出了使用参数 x 替代加解密运算中模数 n 的改进方案和在算法中使用多重密钥 e 指数的方案，并分析证明了改进方案的正确性及其在提升算法安全性方面的优势。效率方面，文中通过探讨制约 RSA 算法运算效率的相关因素，对 SMM 算法和 Miller-Rabin 算法进行了进一步的研究，在此基础上，对两种算法的效率进行了相应的优化，完成了 RSA 算法在数字签名中的具体实现。基于对 RSA 公钥算法改进方案的分析，选择在四素数参数替换 RSA 算法的基础上，再结合 SMM、滑动窗口取幂等算法，重新组合出一种安全而高效的新算法。

最后，本文完成了对 RSA 组合算法密码系统的实现，将传统 RSA 算法、改进的 SMM 算法、多重密钥算法和组合 RSA 算法进行了加解密测试，对比了各个算法所耗的时间。实验结果表明，组合 RSA 算法的签名效率比传统 RSA 算法提升了 54.3%，具有较好的效果。数字签名不仅要满足身份验证、判断信息真伪的要求，还要求能够保证信息的安全性和签名的高效性。应用本文提出的优化算法，可以解决 RSA 公钥算法在数字签名应用中的不足。

关键词：信息安全　RSA 公钥算法　数字签名　参数替换　SMM 算法

Abstract

With the continuous development of the Internet and information society, the application fields and depth of network communication technology are expanding. This brings benefits to people and also brings challenges to information safety. In order to solve the safety problems in practical applications, many targeted technologies and solutions have emerged. As one of the important technologies, RSA public key algorithm can not only be applied in data encryption, but also used as a digital signature scheme. Digital signature based on RSA is a vital approach to defend against security issues such as denial, falsification, and counterfeiting in Internet transactions. How to improve the efficiency of RSA algorithm and ensure its safety, so that it can be better applied in digital signature, has always been a hot topic to be researched.

This paper had taken the RSA algorithm as the research object, the mathematical theory foundations, safety and signature efficiency of the algorithm were studied in depth. In terms of safety, the attacks of RSA algorithm, such as factorization attack, chosen-message attack, and the means to defend against them were expounded. An improved scheme which uses parameter x to replace modulus n based on the traditional RSA algorithm was proposed. In addition, a scheme which uses multiple key e exponents in the algorithm was also suggested. The correctness of the improved schemes was proved, their advantages in enhancing the safety of RSA algorithm were analyzed. Considering the its efficiency, This paper had discussed relevant factors that restrict the efficiency and optimization means were introduced through further research on the existing SMM algorithm and Miller-Rabin algorithm. Digital signature based on RSA algorithm were also realized. Based on the analysis of proposed approaches, the scheme which uses four-prime and replaced parameter was combined with improved SMM algorithm and sliding window exponentiation algorithm to design a new safe and fast RSA algorithm.

Finally, this paper had completed the implementation of combined RSA cryptosystem. The traditional RSA algorithm, improved SMM algorithm, multiple keys algorithm and combined algorithm were used for encryption and

decryption experiments and the consumed time of each algorithm were compared. The results show that the signature efficiency of the combined RSA algorithm is improved by 54.3% compared with the traditional RSA algorithm, which shows a better effect. Digital signature not only requires authentication, authenticity, but also the safety of information and efficiency of signature. The proposed algorithms in this paper can be applied to solving the shortcomings in digital signature based on RSA public key algorithm.

Keywords: Information safety; RSA public key algorithm; Digital signature; Parameter replacement; SMM algorithm

目　　录

1 绪论

1.1 课题研究背景及意义

互联网作为一种当今社会普遍使用的信息交换平台，越来越成为人们生活与工作中不可或缺的一部分，但是各种各样的具有创新性、高复杂度的网络攻击方法也层出不穷。2016 年，国家互联网应急中心 CNCERT/CC 监测发现，我国境内约有 1.7 万个网站被篡改，针对境内网站的仿冒页约 17.8 万个[1]。网络信息的分散性广，来源隐蔽，边界模糊等特点都使得信息的安全性与防御性十分脆弱，不法分子利用网络系统的漏洞肆意进行攻击、传播病毒、伪造信息和盗取机密等等，给政府、企业造成了巨大的经济损失。随着网络信息与共享资源的不断扩大，网络环境的日益复杂，计算机网络信息安全的重要性也日益体现出来。因此，如何在计算机网络中实现信息的安全传输已成为近些年人们研究的热点课题之一。

为了保障网络中数据传输的保密性、完整性、传输服务的可用性和传输实体的真实性、可追溯性、不可否认性，通常情况下所采用的方法，除了被动防卫型的安全技术，如防火墙技术，很大程度上依赖于基于密码学的网络信息加密技术[2-3]。公钥体制是目前研究与应用最为深入的密码体制之一，本文主要研究的 RSA 公钥算法是其中最典型、最具影响力的代表。从最早的 E-mail、电子信用卡系统、网络安全协议的认证，到现在的车辆管理和云计算等各种安全或认证领域[4-5]，这种密码算法可以抵御大多数的恶意攻击方式，也因此被推荐为公钥加密的行业标准[6]。与此同时，各种新的、针对性的攻击方式也在不断出现，512 位的 RSA 公钥算法早在 1999 年就已被超级电脑因式分解破译，768 位的 RSA 公钥算法也在十年后的 2009 年 12 月被攻破。2010 年，美国密歇根大学的三位科学家已研究出了在 100 小时内破译 1024 位 RSA 公钥算法的方法。2013 年 8 月，Google 公司宣布使用 2048 位的 RSA 密钥加密和验证 Gmail 等服务。为了保证 RSA 公钥算法的安全性，最有效的方法是不断地提高算法中密钥的位数[7]。这样的做法虽然确保了算法的安全性，但是也增加了密钥选取的困难性与算法中的基本运算——大整数模幂乘运算的复杂性，算法的效率随之急剧地降低，也成为了影响其发展的主要瓶颈之一。

数字签名是公钥密码学发展过程中衍生出的一种安全认证技术，主要用于提供实体认证、认证密钥传输和认证密钥协商等服务[8]。简单来说，发送消息的一方可以通过添加一个起签名作用的编码来代表消息文件的特征，如果文件发生了改变，那么对应的数字签名也要发生改变，即不同的文件经过编码所得到的数字签名是不同的。使用基于 RSA 公钥算法的数字签名技术，如果密码攻击者试图对原始消息进行篡改和冒充等非法操作，由于无法获取消息发送方的私钥，所以也就无法得到正确的数字签名；若消息的接收方试图否认和伪造数字签名，则公证方可以用正确的数字签名对消息进行验证，判断消息是否来源于发送方，这样的方式很好地保证了消息文件的完整性、鉴定发送者身份的真实性与不可否认性[9]。针对 RSA 公钥算法攻击方式的深入研究，制约了其在数字签名中的应用。并且，当下许多的领域对数字签名技术提出了各种新的应用需求，在未来的网络信息安全和身份认证中，基于 RSA 公钥算法的数字签名技术在理论和实际应用方面，仍具有重要的研究价值与研究意义。

1.2 国内外研究现状

1.2.1 RSA 公钥算法研究现状

由于公钥密码体制的提出，在不安全的信道上，通信双方仍然可以进行安全的通信，RSA 作为其中最具代表性的算法，在许多的应用领域都占据了重要的地位。关于 RSA 公钥算法的研究，主要针对的是两个方面，一是恶意攻击与防御攻击的研究，二是针对算法效率的优化。

RSA 公钥算法的安全性是建立在数论中的著名难题——对大整数因子分解困难性的基础之上[10]。算法的攻击方面，李超等分析了选择密文攻击、小指数攻击等攻击方法和格方法在 RSA 攻击中的应用[11]。文献[12]在 Shor 的量子分解算法[13]的基础上，提出了一个新的量子分解算法，可以使大整数因子分解的成功率大于 $4\phi(r)/(\pi^2 r)$。针对私钥 d 的攻击[14]，勾云、曾光等提出通过构造系数格与一个新的双变元模方程，利用格基约化求解方程小根的方法，获取私钥的界以及较小的素因子和加密指数之间的关系，成功地攻击了两种 RSA 公钥算法的变体形式[15]。Bunder、Martin[16]提出基于 Wiener 连分式方案的改进算法，对于 1024 位的 RSA 算法，可以适用 270 位以上的私钥值，具有较强的攻击效果。与攻击方案相对应，不少相关学者也给出了增

强算法安全性的方案。文献[17][18]提出了一种加快整个网络的数据交换过程中 RSA 算法实现的改进形式，通过在公钥和私钥的构成中使用额外的第三素数的方法增加参数 n 因子分解的复杂性，但容易受到选择密文攻击。Mustafi、Sheikh 等提出利用双变量双射函数的优化方案，但是算法的实现过程过于复杂[19]。Hassan、Shalash 等在文献[20]中给出了一种基于遗传算法的 RSA 改进方案，增强密钥的复杂度以提高算法的安全性。

算法的效率方面，制约其性能最主要的瓶颈是加、解密过程中的大数模幂乘运算，这也是 RSA 公钥算法中的关键性运算。另外，随机大素数因子和密钥的生成速度也是算法效率的影响因素[21]。为了解决大数模幂乘运算时间长、开销大这个问题，学者们付出了很多努力。文献[22]提出了一种 SMM 算法，通过利用乘同余对称特性，对在 RSA 算法每次加密迭代计算过程中作为基数的乘数与被乘数进行一定条件下的替换，从而降低一些运算步骤中的乘数与被乘数的值，使算法得到一定程度的优化，但是优化效果不明显。Rizzo 提出了指数 2^k 进制化算法，主要是对模幂运算中的指数进行 k 进制化，缩短其序列长度，从而降低了迭代的步数，减少了整体加密过程的运算量和存储空间[23]。文献[24]中给出了将 SMM 算法和指数 2^k 进制化算法相结合的改进方案。同样针对指数，谢琪提出一种指数约减算法[25]，但是仅适用于私钥 d 已知的情况。石小平与姜浩提出基于模重复平方算法的 rho 改进算法[26]，根据初等数论的原根与指数理论，在模运算过程 $b^n \bmod m$ 中，如果 b 与 m 互素，先计算 $\phi(m)$，接着在 $\phi(m)$ 的因子中找 b 模 m 的阶 d，从而化简运算；但是当 $\phi(m)$ 比较大时，获取阶 d 的运算量比较大，计算 $b^n \bmod m$ 常见的算法是模重复平方算法，rho 算法是一种模重复平方算法的改进算法，中间使用循环二进制的方法，提升了模幂乘的运算效率。黄学渊给出了在 RSA 公钥算法中使用中国剩余定理的方法[27]，但是文献[28]和[29]均提出在传统 RSA 算法中使用中国剩余定理容易受到出错攻击。

综上所述，国内外学者分别从不同方面改进了 RSA 公钥算法，促进了 RSA 的不断发展，这些既有的研究成果为本课题相关工作的开展提供了良好的基础。计算机与大数据的急剧发展，对密码算法的效率提出了更高的要求；针对算法的不同攻击也在继续之中，所以如何提高算法安全性和运算效率的研究仍在不断地深入。

1.2.2 数字签名技术研究现状

20 世纪 70 年代，公钥密码体制被提出之后，应无纸化办公、数字化和信息化发展的要求，数字签名技术[30]也随之应运而生。不同于一般的手写签名，数字签名是一种电子形式的签名方式，但是作用与一般的手写签名或者印章类似。数字签名技术在实现消息的保密传输、数据完整性、身份认证和不可否认性方面的功能，对于物联网、电子商务、医疗系统等领域的发展都起着非常重要的作用[31]。

数字签名的方案主要依赖于密码技术，比较主流的签名方案主要有基于公开密钥的方案、基于 ElGamal 的方案[32]、基于椭圆曲线 ECC 算法的方案[33]等。RSA 公钥算法由于可以同时用于信息加密和数字签名，并且具有较好的安全性，是目前应用最普遍的签名方案之一[34]。文献[35]将 ECC 算法和 RSA 公钥算法签名方案在 ARM7 上进行了签名对比操作。随着数字签名技术研究的愈加深入，基于 PKI 体制[36]、基于格[37]和很多不同的签名类别，如盲签名、群签名、代理签名[38]等签名方案日益增多。同时，关于数字签名的安全性分析和攻击研究也在不断深入[39-40]。张文芳、熊丹等提出一种基于 RSA 的环签名方案，并给出了其安全模型[41]。Kamal、Gupta 等给出并分析了一种改进的 RSA 数字签名方案，较好地保证了签名过程的安全性和效率[42]。Kumar 和 Singh 在文献[43]中设计了基于 RSA 公钥算法的 SRNN 算法，除了 RSA 中的双素数之外，在密钥对中还使用另外两个自然数，但并未对新的签名算法的安全性和效率进行分析。

自从数字签名技术被提出以来，不少密码学者和相关的机构，如麻省理工学院、IBM 研究中心等都开始致力于数字签名的研究，并高度关注该项技术的发展。1984 年 9 月，在国际标准化组织的建议下，SC20 率先开始为数字签名技术制订标准，将其分为带影子、带印章以及使用 Hash 函数的三种数字签名。WG2 在 1988 年 5 月起草了使用 Hash 函数的签名方案 DP9796，并于 1989 年 10 月将其升级为 DIS9796。与此同时，日本、英国等国的相关组织也迅速展开了对数字签名标准的制订工作。1991 年 8 月，美国 NIST 公布了 DSA 签名算法，并将其纳入 DSS 数字签名标准之中。2000 年 6 月，美国通过了有关数字签名的法案。目前，影响较大的制订数字签名相关标准的组织包括：国际电子技术委员会(IEC)、国际标准化组织(ISO)、美国国家标准与技术委员会(NIST)等[44-45]。在国际上，数字签名技术的相关标准和法案已日趋完善，我国因为在这一领域起步较晚的关系，在数字签名的操作中主要是吸收国外的经验，并且还存在很多不规范之处。鉴于此，2005 年 4 月正式颁布并施行了《中华人民共和国电子签名法》，这部法律不仅确立了数字

签名的法律效力和地位，还规范了数字签名的行为，维护了有关方的合法权益，促进了我国电子商务的发展，我国数字签名研究工作的顺利进行也因此得到了有力地支持和保障。

1.3 论文结构安排

第一章，介绍了本课题的研究背景和研究意义，重点对信息安全、密码算法及数字签名的发展进行了概括与总结，综合目前国内外的研究现状，确定了本课题的研究方向，为课题的后续开展作了前期的准备工作。

第二章，首先介绍了密码体制的构成与分类，阐述了使用基于 RSA 公钥算法的数字签名方案的优越性；接着对 RSA 算法中所运用到的数学理论进行了描述，然后分析了 RSA 公钥算法的安全性和效率以及数字签名的原理，为后文算法的优化及在数字签名中的应用奠定了基础。

第三章，详细分析了 RSA 公钥算法的各种攻击方式，针对因子分解攻击、选择密文攻击与公共模攻击提出了参数替换和多重密钥的方案，并对改进方案在提升算法安全性上的优势进行了分析；算法的签名效率方面，在讨论了签名过程中关键步骤的 SMM 算法的基础上，对其进行了相关的优化和详细的分析，通过实验验证了优化的算法对于 RSA 算法效率的提升。

第四章，分析了传统 RSA 签名方案的缺点，给出了改进的 RSA 数字签名方案的实现过程，其中利用 Hash 函数来生成消息摘要，在实现数字签名的关键步骤——随机大素数和密钥的生成中，对 Miller-Rabin 算法进行了研究与分析，并给出了相关的优化方案，实验结果表明优化的算法具有更高的素数生成效率。最后，完成了 RSA 算法签名方案的实现过程。

第五章，通过对 RSA 公钥算法改进方案的分析，选择在四素数参数替换 RSA 算法的基础上结合改进的 SMM 算法与 Miller-Rabin 算法，重新组合出一种新的签名算法。本章实现了 RSA 密码系统，展示并介绍了系统的操作流程。最后，将传统 RSA 算法、改进的 SMM 算法、多重密钥算法和组合 RSA 算法进行实验对比，结果表明组合算法在签名效率上有较为明显的提升。

总结与展望。总结了本文所做的研究工作，并对 RSA 公钥算法和数字签名技术的研究前景作出了进一步的展望。

2 密码体制与数字签名技术

根据本课题的研究目标，为了对基于 RSA 的数字签名技术展开研究工作，首先充分阐述了密钥体制、RSA 算法主要依据的数学理论和数字签名技术，包括数字签名的工作原理、安全性分析及应用等，进而为后续的工作铺垫理论基础。

2.1 密码体制的构成

密码学的基本思想是对需要保密的数据信息进行改造或伪装，使不具备伪装过程中相关知识的人不能理解信息真正的含义。对信息进行伪装本质上是对其做一系列可逆的数学变换。密码学中的基本概念包括明文、密文、加密和解密。明文是未经伪装的原始信息，密文是伪装之后隐藏了原始含义的信息，加密是对信息进行伪装的过程，解密是计算机除去伪装，将密文恢复为明文的过程。现代密码学中，密码体制一般包括以下五个部分[46]：

（1）明文空间 M：由全体明文构成的的有限集；

（2）密文空间 C：由全体密文构成的有限集；

（3）密钥空间 K：由全体公钥和私钥构成的有限集；

（4）加密算法 E：一组包含 M、C 的加密变换；

（5）解密算法 D：一组包含 C、M，且与加密变换互逆的解密变换。

如果密码攻击者可以获取密钥，或者可以根据 C 确定 M，那么该保密系统就是可被攻破的。只要具有足够的人力、工具与技术，实际使用中的任何密码系统都是可被攻破的，因此抗击分析、提高破解难度，是保证密码系统安全的重点。保密系统应满足下列要求[47]：

（1）如果无法从理论上确保系统不可破解,就应在实际应用中保证它是不可破解的；

（2）系统的安全性应尽量借助于密钥的安全性,而非密码体制或算法的实现细节。

2.2 密码体制的分类

密码体制最早能够追溯到远古时期的古典密码体制，现代密码学以算法中密钥的对称性作为分类标准，可以将密码体制分成对称与非对称两类。

2.2.1 古典密码体制

密码学一词出自古希腊语的 Kryptos（隐藏）和 Graphein（书写），意为利用某些变换将原始信息转化为难以识别的信息，从而隐藏原始信息的含义。在当时，这并非科学上的密码概念，只是一门艺术，也是现代密码学的基础与渊源。古典密码具有悠久的历史，基本的数据单元都是字符，加、解密采用的方法分为手工和机械操作两种。虽然不同于现代加密算法的使用计算机进行运算，但是它们所依据的信息变换的基本原理都大同小异。

古典密码的基本编码方式总的来说可分为置换（Permutation）和代换（Substitution）[48]。不同类型的古典密码是通过数据单元之间彼此置换或者代换完成的，优秀的密码大多同时使用这两种编码方式进行运算。置换法是保持原始信息的内容不变，只对信息元素的位置进行变换。例如，重新排列原始信息中的字母或符号。这种置换可以是一维的，也可以是多维的。代换法是重新表示原始信息中的元素，但是不改变信息元素的位置。例如，以其他的字母、符号或者数字替换原始信息中的字母，这样虽然原始信息中的字母被改变了，但并未改变它们彼此之间的位置。

较为著名的古典密码包括基于单表代换的凯撒密码、基于多表代换的转轮密码、维吉尼亚密码，以及基于多字符代换的希尔密码等[49]。

2.2.2 对称密码体制

对称密码体制是相对传统的一种密码体制，又被称作私钥密码体制。对称密码体制的特点是保密性很高，面对高强度的分析与攻击，能体现良好的安全性能，不仅可以实现对数据的加密，还能够用于消息的认证。在对称密码系统中，加、解密过程中使用的是同一个密钥，所以有时也将其称作单钥密码。

正因为使用的是相同的密钥，所以对称密码体制也存在着严重的不足，在进行信息传输之前，要求通信的双方事先协商出一个安全的密钥，并且密钥的协商与传送需要通过绝对安全的信道，每一方也要求能够信任对方会对密钥保密，这样才可以保障信息的机密性和完整性。但是在实际的网络环境中，这样的一条安全的信道一般难以确定。另外，密钥的管理也会影响到系统安全性，这也使得对称密码体制的开放性不足，难以满足实际应用中的要求。如果面对的用户量不是很大，这种密码体制是有效果的，但是在用户量很庞大且在通信网络中分布很广的情况下，密钥的管理、分配和保存都是需要解决的难题[49]。

较为典型的对称密码算法包括IBM公司开发的DES算法及其变形Triple DES[50],美国国家标准局在1977年1月将其确定为联邦资料处理标准(FIPS)。此外，还有一些其他的对称密码算法，如RC5、IDEA算法[51]。对称密码体制由于只能用作对信息的加解密处理，不能进行数字签名，使得其应用范围逐渐缩小，这也促使着后来非对称密码体制的诞生。图2-1中给出了对称密码体制的实现过程：

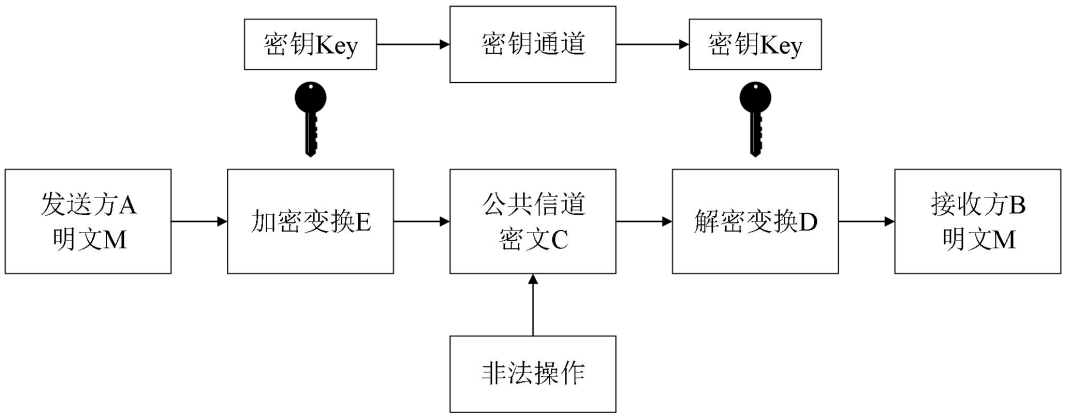

图 2-1 对称密码体制

Fig 2-1 Symmetric cryptosystem

2.2.3 公钥密码体制

与对称密码体制相对的是公钥密码体制，又称为非对称密码体制或双钥体制。公钥密码体制的概念主要是针对前面所述的对称密码体制的不足，由美国的Diffie与Hellman于1976年在共同发表的《New Direction in Cryptography》中首次提出，文中详细论述了这种实现公钥密码系统的新方法，它的出现被视为密码学发展史上最伟大的一次创新，也标志着现代密码学的诞生[52]。随后在1978年,Ronald.L.Rivest、Adi.Shamir和Leonard.Adleman三人提出了第一个行之有效的公钥密码算法——RSA公钥密码算法，国际标准化组织ISO将其推荐为公钥加密的标准。公钥密码体制的实现过程如图2-2所示：

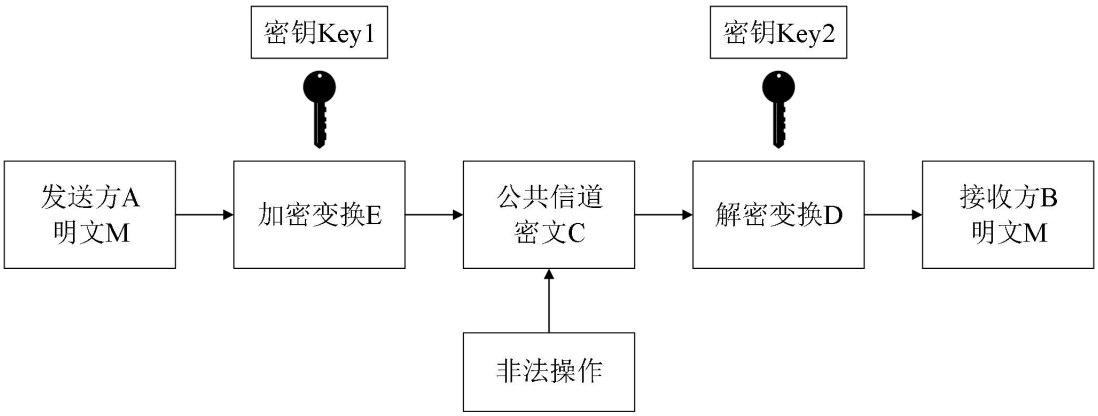

图 2-2 公钥密码体制

Fig 2-2 Public key cryptosystem

公钥密码体制一经提出,就受到了密码学界的高度的关注与广泛的探讨,它利用了单向陷门函数的数学原理，使加密密钥与解密密钥可以分离，即在密码系统中使用双密钥——公钥和私钥，其中公钥用于加密，可以对外公布，私钥用于解密，要求由私人保密。恶意攻击者试图以已知的公钥导出另外一个私钥在计算上是行不通的，或者说需要付出的代价是相当大的。它的主要特点是具有良好的开放性，能够适用于复杂的使用环境，通信双方不必事先互换密钥就可以在一个不安全的信道上进行安全通信，很好地克服了对称密钥体制在这一方面的不足。除此之外，密钥管理的问题较之对称密码体制也比较容易，可以便捷、安全地实现加密、数字签名和验证，尤其适用于大型通信网络中用户量庞大的情况，极大地降低了多用户进行信息传输所需的密钥量，节省了系统的开销。最著名的 RSA 公钥密码算法以其实现原理简单和安全性良好的特征，已经广泛地应用在各种信息安全环境中，但是算法中的模幂剩余运算耗时久这一劣势，至今也还没有很好的解决方案[53]。目前，公钥密码体制有如下主要研究方向：

（1）实际应用环境下的密码系统设计；

（2）公钥密码体制的快速实现；

（3）公钥密码的安全性评估问题；

（4）基于公钥密码体制的新模型和单向陷门函数的研究。

表 2-1 对上述的对称密码体制和公钥密码体制的优缺点进行了对比：

表 2-1 对称密码与公钥密码的对比

Table 2-1 Comparison between symmetric cryptosystem and public key cryptosystem

密码体制	运行条件	安全性	运行速度
对称密码	通信双方拥有同一密钥； 共享加密算法与密钥。	密钥的传输途径必须可靠； 密钥管理成影响因素。	加解密速度快。
公钥密码	通信双方各具一对密钥； 加密和解密的密钥不同。	适用开放性环境； 密钥易管理。	速度较慢，不适用于信息量庞大的情况。

2.3 RSA 公钥密码的数学理论基础

公钥密码体制的构造依赖于单向陷门函数，它的求逆困难性是大多数公钥密码的安全性基础。数论中的许多概念在设计公钥算法时是必不可少的，如费马小定理，欧拉定理等。其中，在素性测试方面，费马小定理有着至关重要的作用，RSA 公钥算法的正确性依赖于欧拉定理的理论。研究这些概念与定理是理解 RSA 公钥算法的前提与基础。

2.3.1 单向陷门函数

定义 2.1 设 X 和 Y 为两个集合，且令函数 f 是 X 到 Y 的映射。若对任意 $x_1, x_2 \in \mathrm{X}$，$x_1 \neq x_2$，有 $f(x_1) = f(x_2)$ 则称 f 为一一映射。若 f 满足：

（1）已知 x，$f(x)$ 易于求解，能够在多项式时间内完成计算；

（2）对于 $x \in \mathrm{X}$，已知 $f(x)$ 的值，求解 x 使之满足 $f(x)$ 是相当困难的，以至于在实际中不可能做到，但是若能获得限门，则可以计算出 $x = f^{-1}(y)$，$f^{-1}(y)$ 为 f 的反函数。

那么称函数 f 为单向陷门函数。这里所说的“单向”不同于数学上的“不可逆”，单向函数可能在数学上是一个不可逆函数，但是反之却不一定。定义中的“相当困难”针对的是目前的算法与计算机性能。虽然不少函数由于可以体现出“单向”的性质，在实际的使用中也被视为是单向的，但是至今

仍无法从理论的角度证明存在真正的单向函数[54]。图 2-3 给出了单向限门函数在 RSA 公钥算法中的应用。

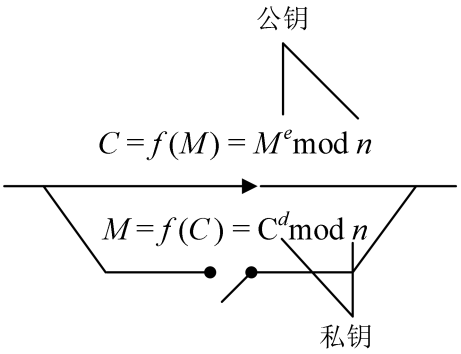

图 2-3 单向陷门函数在 RSA 中的应用

Fig 2-3 One-way trapdoor function in RSA

2.3.2 模运算与同余

定义 2.2 对于给定的任意整数 x 和正整数 n，一定存在整数 k 和 a 使得 $x=kn+a$ 的等式成立，其中 $0\le a<n$，那么称 a 是 x 对 n 的模运算，可以记作 $a=x \bmod n$ 或者 $a=x\%n$。

定义 2.3 对于整数 x、y 以及正整数 n，若 x 和 y 之间的差值能够被 n 整除，即 $(x-y)/n$ 可以得到一个整数，则称整数 x 与 y 对模 n 同余，或者 x 同余于 y 模 n，可以记作 $x\equiv y \bmod n$，同余本质上表示的是两个整数之间的等价关系。

模运算符具有的性质如下：

（1） $x\equiv y \bmod n$ 等价于 $y\equiv x \bmod n$；

（2） $x \bmod n = y \bmod n$ 等价于 $x\equiv y \bmod n$；

（3）如果 $x\equiv y \bmod n$ 且 $y\equiv z \bmod n$，那么有 $x\equiv z \bmod n$。

2.3.3 模运算的操作规则

对于模 n 的运算是指把一切整数都映射到 $\{0,1,2,3\cdots n-1\}$ 上，并在此集合内进行的运算。模运算的规则和四则运算的规则相类似，满足交换律、分配率和结合律，但是其中除法运算是例外，相关的规则列举如下：

$$(x+y) \bmod n = (x \bmod n + y \bmod n) \bmod n \qquad (2\text{-}1)$$

$$(x-y)\bmod n=(x\bmod n-y\bmod n)\bmod n \quad (2\text{-}2)$$

$$(x\times y)\bmod n=(x\bmod n\times y\bmod n)\bmod n \quad (2\text{-}3)$$

$$x^y\bmod n=(x\bmod n)^y\bmod n \quad (2\text{-}4)$$

其中公式（2-4）可以将模幂运算转换成模乘运算，与指数运算本质上可以视作多次重复的乘法运算相类似，可以视 $x^y\bmod n$ 为 y 次方的模乘运算 $(x\bmod n)^y\bmod n$。例如，$25^7\bmod 13$ 可以按如下方式进行计算：

$$\begin{aligned}&25^7\bmod 13\\&=(25\bmod 13)^7\bmod 13\\&=12^7\bmod 13\\&=((12^2\bmod 13)^3\times 12\bmod 13)\bmod 13\\&=12\end{aligned}$$

2.3.4 费马小定理与欧拉定理

定义 2.4 设 a 为一个正整数，p 为一个素数且与 a 互素，则:

$$a^p\equiv a\bmod m \quad (2\text{-}5)$$

此即费马小定理的内容，公式（2-5）的运算都是通过 $\bmod p$ 实现的，故对所有整数元素 a 而言，此定理始终成立。此定理也可以表示为如下形式：

$$a^{p-1}\equiv 1\bmod m \quad (2\text{-}6)$$

这种形式在密码学中应用之一就是求有限域内某个元素的逆元，将原表达式改写为 $a\cdot a^{p-2}\equiv 1(\bmod p)$ 就是乘法逆元的定义。

定义 2.5 设 m 为正整数，$\phi(m)$是小于 m 的正整数并且与 m 互素的数的个数，譬如$\phi(1)=1$、$\phi(8)=4$，$\phi(m)$称作欧拉函数。若 a 也为正整数，且 $GCD(a,m)=1$，则

$$a^{\phi(m)}\equiv 1\bmod m \quad (2\text{-}7)$$

此即为欧拉定理。由欧拉定理可以扩展出以下几个定理：

（1）若 $GCD(m,n)=1$，则有 $\phi(mn)=\phi(m)\phi(n)$。

（2）若 $m\geq 1$，$GCD(a,m)=1$且存在 p 使得 $pa\equiv 1\bmod m$，则称 p 为 a 的模 m 的逆，记作 $a^{-1}(\bmod n)$。

（3）若 $a\equiv b(\bmod m_1)$、$a\equiv b(\bmod m_2)$、$a\equiv b(\bmod m_3)$、…、$a\equiv b(\bmod m_k)$，则有 $a\equiv b(\bmod(m_1m_2m_3\cdots m_k))$。

（4）若 p 为素数，$GCD(p,a)=1$且 $a^{p-1}\equiv 1(\bmod p)$，则可以推出 $a^p\equiv a(\bmod p)$。

实际上，费马小定理可视为欧拉定理的推论之一，将费马小定理的素数模数推广至任意整模数，就可以得到欧拉定理。费马小定理与欧拉定理及其扩展定理共同组成了 RSA 公钥算法主要的数学理论基础[55]。

2.4 RSA 公钥算法的安全性

RSA 公钥算法安全性的影响因素有很多，其中起决定性作用的是针对模数 n 的因子分解的困难性。这种说法仅仅是从理论上进行分析的，从实际技术角度上来说并不完全可靠。因为迄今为止，国际上还没有在数学上证明针对模数 n 的分解就是攻击 RSA 公钥算法的最好方案。事实上，过去几百年来许多的国际数学家对大整数因子分解的问题都进行了深入的研究，但这项世界性难题一直攻而不克，至今都尚未能找到一种有效的算法来解决。大多数的科学家研究的结论都是认为针对大整数因子分解的多项式算法并不存在，目前只能利用的计算机技术尝试分解。由于计算机的性能和处理能力的提高以及因子分解方法的改进，已经可以成功分解出 1024 位的密钥[56]。分析人员也采用了不少非因子分解的方式来攻击 RSA 公钥算法，但是这些方式都不比分解 n 更容易。

RSA 公钥算法从提出至今，已经广泛应用在了许多的领域，这足以说明 RSA 具有很高的可靠性，但是也绝不可认为它是攻不可破的[57]。之所以许多研究人员都试图改进 RSA 并对算法的实现细节进行精心设计，是因为在特定的条件下，RSA 算法的具体实现细节之处可能出现缺陷，从而使密码攻击者找出攻破算法的可能性。随着算法中密钥位数的增加，因子分解攻击所需的时间也会成指数倍地上升，只要模数 n 的长度满足一定的要求，并且素因子 p、q 和加密指数 e 选择合适的值，在一定程度上可以克服其安全性缺陷。

2.5 数字签名技术

数字签名是公钥密码体制衍生出的最重要的技术之一。相较于在纸上书写的物理签名，数字签名首先对信息进行处理，再将其绑附一个私钥上进行传输，形成一个比特串的表示形式，可以实现用户对电子信息来源的认证，并对信息进行签名，确认并保证信息的完整性与有效性。在数字签名出现之前，除了传统的物理签名，还有一种“数字化”签名技术，即在电子板上进行签名，接着再将签名传输到电子文件中，虽然相比物理签名，这种方式更

加便捷，然而仍然可以被非法地复制并粘贴到其他文件上，因此签名的安全性无法得到保障。不同于“数字化”签名技术，数字签名与手写签名的形式毫无关系，它实际上是使用密码学的技术将发送方的信息明文转换成不可识别的密文进行传输，在信息的不可伪造性，不可抵赖性等方面有着其他签名方式无法替代的作用。

2.5.1 数字签名的原理

数字签名发展至今，常见的签名算法除了 ElGamal 签名算法、RSA 签名算法，还有 Schnorr 算法[58]、DSA 等算法[59]。公钥密码体制并非都可以同时用作密码系统和进行数字签名，一般只能实现其中的一种功能，而本文所研究的 RSA 公钥密码体制，可以同时实现二者。下面给出数字签名的原理：

（1）系统的初始化，生成数字签名中所需的参数；

（2）发送方利用自己的私钥对消息进行签名；

（3）发送方将消息原文和作为原文附件的数字签名同时传给消息接收方；

（4）接收方利用发送方的公钥对签名进行解密；

（5）接收方将解密后获得的消息与消息原文进行对比，如果二者一致，那么表示消息在传输中没有受到过破坏或者篡改，反之不然。

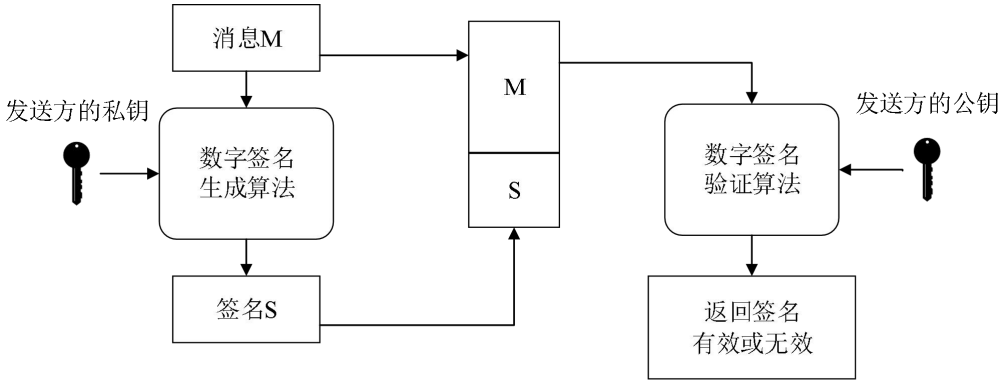

图 2-4 消息签名和验证的基本原理

Fig 2-4 Basic principles of message signing and verification

2.5.2 数字签名的安全性分析

针对数字签名的安全性的研究，即鉴定签名方案是否满足信息完整性、抗修改性和抗抵赖性等安全性质，一直是促进该项技术发展的重要方面之一。安全性的主要研究方法是从理论上进行分析，分析的技术分为三种[60]：

（1）安全性评估。在数字签名方案设计的过程中，设计者对所设计的方案进行相关的密码分析，尽可能保证其在所能考虑到的范围内的安全性。由于一般情况下，方案设计者无法穷举所有可能出现的情况，因此这种分析不是证明，只是一种对签名方案安全性的评估，可以在一定程度上使用户对方案拥有信心。

（2）安全性证明。这种方式主要应用在数字签名方案设计的过程中，证明数字签名方案被密码攻击者破译的难度与破解一个公认的难题相当。目前主要的技术方案有两类——基于随机应答模型与基于标准模型。前者就是假设 Hash 函数是一个对所有请求都会作出随机应答的随机黑盒应答器，相同的请求得到的应答完全相同。然而实际上，Hash 函数并不满足这个假设，所以基于随机应答模型的证明不完全可靠。后者是指在不需要不合理假设的情况下，证明方案是安全的，所谓的标准模型本质上并不是一个具体的模型，仅是针对随机应答模型而言的。

（3）攻击。此类技术是指对方案的安全缺陷进行研究，主要针对的是目前已知的数字签名方案。分析人员从攻击者的角度进行演绎与推理，尝试通过不同的手段获取在方案安全性规定下不应该或者不能够获取的信息。如果成功地获取到了这样的信息，就找出了已知方案所存在的安全性缺陷，然后再次分析整个推理过程。针对攻击的分析中，恶意分析者被称为攻击者，方案分析人员演绎与推理的过程被称为一个攻击方法。攻击分析促使人们对存在安全性缺陷的方案进行改进或者直接淘汰，从客观上提升了数字签名的安全性。

2.5.3 数字签名的特征与应用

一个优秀的数字签名应当具备下列的特征：

（1）消息发送方一旦给接收方发出签名之后，不可以再对他所签发的消息进行否认；

（2）消息接收方可以确认并证实发送方的签名，但是不可以伪造签名；

（3）如果签名是复制其他的签名获得的，那么消息接收方可以拒绝签名的消息，即不可以通过复制的方式将一个消息的签名变成其他消息的签名；

（4）如果消息接收方已收到签名的消息，就不能再否认；

（5）可以存在第三方确认通信双方之间的消息传输的过程，但是不可以伪造这个过程。

在互联网和通信技术迅猛发展的潮流下，电子商务一跃成为了商务活动的新模式，越来越多的信息都以数字化的形式在互联网上流动。在传统的商务系统中，一般都是利用纸质文件的签名或印章等物来规定某些具有契约性质的责任，而在电子商务系统中，当事人身份与数据信息的真实性是利用传送的文件的电子签名进行证实的。电子商务包括电子数据交换 EDI、管理信息系统 MIS、商业增值网 VAN、电子订贷系统 EOS 等，其中 EDI 既是电子商务这些部分中的核心，也是一项涉及多个环节的复杂的人机工程，同时对信息安全性的要求也相当高。互联网环境的开放性、共享性等特点使得网络信息安全变得异常脆弱，如何保证数据信息在互联网上传输的安全性以及交易双方的身份确认不仅是电子商务的必然要求，也是电子商务能否得到长足发展的关键。作为电子商务安全性的重要保障之一，RSA 公钥签名方案相对成功地解决了上述的问题，并在各种互联网行为之中都有着广泛的应用。

2.6 本章小结

本章首先对密码体制的分类进行了简单的介绍，然后重点介绍了公钥密码体制的概念和实现数字签名的优势。给出了 RSA 公钥算法所依赖的数学理论基础。另外，文中还阐述了数字签名技术，包括数字签名的基本原理，安全性分析以及在网络电子交易中的相关应用，为后续对 RSA 算法的优化实现及其在数字签名应用中的改进奠定了基础。

3 RSA 公钥算法的分析及改进

1024 位 RSA 算法已被证实存在风险，如果选择在实际中继续应用，就不得不使用更高比特的算法，因此如何进一步提高和保证 RSA 算法的安全性，有着十分重要的意义。算法在使用中大都选取 100 位左右的素因子，甚至更高，如此大的整数进行模幂乘运算，会对算法加密与签名的效率产生巨大的影响，因此本文也对算法中的某些局部过程进行探讨，并从多方面做了大量优化工作，以提升算法的效率。

3.1 传统 RSA 与四素数 RSA 算法

RSA 公钥算法在加密过程 E_{Ke} 和解密过程 D_{Kd} 中使用的是两种不同的密钥，即加密密钥 K_e 与解密密钥 K_d，解密者需要根据 K_d 确定 E_{Ke} 的逆。

3.1.1 算法描述

RSA 公钥算法包括密钥生成，消息加密，消息解密三个步骤，下面给出算法的描述：

（1）RSA 密钥生成

①任意选取两个不相干的大素数 p 和 q，并将其存储起来；

②计算 $n = pq$ 和欧拉函数 $\phi(n) = (p-1)(q-1)$；

③随机地选取一个正整数 e，使其满足 $1<e<\phi(n)$，并且要求 e 和 $\phi(n)$ 的最大公约数 $GCD(e,\phi(n)) = 1$，公钥为 (e,n)；

④计算解密密钥 d，使其满足 $0<d<\phi(n)$，而且 $ed \equiv 1(\mathrm{mod}\,\phi(n))$，私钥为 (d,n)。

（2）消息加密

在加密消息明文之前，首先要对明文进行处理和分组。分组的要求是每个明文的十进制数不大于 n；接着依次对每段数字化块明文 M 进行加密变换和下一步中的解密变换。

消息发送方利用公钥 (e,n) 对明文 M 进行加密：

$$C = M^e \bmod n \qquad (3\text{-}1)$$

（3）消息解密

消息接收方利用私钥 (d,n) 对密文 C 进行解密：

$$M = C^d \bmod n \qquad (3\text{-}2)$$

其中，M 代表待加密的明文，C 代表加密后的密文，e 代表加密密钥，d 代表解密密钥，n 代表模数。

下面对 RSA 公钥算法的正确性进行简要证明：要证明 RSA 的正确性，即证明由 $C^d \bmod n$ 能够推导得出 M。因为 $ed \equiv 1(\bmod \phi(n))$，所以有 $ed = k\phi(n)+1$，其中 k 为任意的正整数。根据 2.3.4 节中所阐述的欧拉定理可知：$M^{\phi(n)} \equiv 1 \bmod n$，则有

$$\begin{aligned} & C^d \bmod n \\ & = M^{ed} \bmod n \\ & = M^{k\phi(n)+1} \bmod n \\ & = (M^{k\phi(n)})M \bmod n \\ & = 1^k M \bmod n \\ & = M \bmod n \\ & = M \end{aligned} \qquad (3\text{-}3)$$

由（3-3）推导过程可见，RSA 公钥算法的加解密变换在一定条件下可以相互转换，这表明无论用于对信息的加密还是实现数字签名，RSA 的有效性都是可以得到验证的。

相比传统 RSA 算法的双素数因子，四素数 RSA 算法是选取四个随机的素因子，其描述如下：

（1）密钥生成

①随机产生四个不同的大素数 p、q、r 和 s，计算 $n = pqrs$ 和 $\phi(n) = (p-1)(q-1)(r-1)(s-1)$；

②随机地选取一个正整数 e，使其满足 $1 < e < \phi(n)$，并且要求 e 和 $\phi(n)$ 的最大公约数 $GCD(e,\phi(n)) = 1$，公钥为 (e,n)；

③计算解密密钥 d，使其满足 $0 < d < \phi(n)$，而且 $ed \equiv 1 \bmod \phi(n)$，私钥为 $(d，n)$。

（2）消息加密

明文的处理过程与传统算法类似，利用 (e,n) 对明文加密：

$$C = M^e \bmod n \qquad (3\text{-}4)$$

（3）消息解密

消息接收方利用 (d,n) 对密文 C 进行解密：

$$M = C^d \bmod n \qquad (3\text{-}5)$$

3.1.2 双素数与四素数 RSA 的效率对比

本文使用 Miller-Rabin 算法随机产生两个 1024bit 和四个 512bit 的素数，然后用传统 RSA 算法和四素数 RSA 算法分别对 512bit 的明文进行签名，并记录整个过程的耗时情况（单位：ms）。实验的硬件环境为：i5-2450M CPU，6GB 内存，450G 硬盘；操作系统为 Windows 7；开发工具：Visual Studio 2017。实验结果如表 3-1 所示：

表 3-1 两种算法耗时情况对比

Table 3-1 Comparison of the time consumption of the two algorithms

算法	第一次	第二次	第三次	第四次	第五次	平均
传统 RSA 算法	3547	3461	3501	3517	3489	3503
四素数 RSA 算法	2178	2209	2231	2197	2215	2206

从表 3-1 可以看出，在密钥总长度均为 2048bit 的情况下，四素数算法对 512bit 明文的签名效率比传统 RSA 算法提高了 37%。对于相同的密钥位数，四素数 RSA 算法比传统 RSA 算法所要求产生的随机素数要小，能够减少密钥生成的运算量。

3.2 RSA 算法的攻击及分析

RSA 公钥算法的破译方式主要分两大类。其一是密钥穷举法，即找出所有可能的密钥组合；其二是密码分析法。因为 RSA 算法的加解密变换具有庞大的工作量，用密钥穷举法进行破译基本上是行不通的，所以只能利用密码分析法对 RSA 算法密文进行破译。目前，关于密码分析法的攻击方式主要包含下列几种：

3.2.1 因子分解攻击

因子分解攻击是针对 RSA 公钥算法最直接的攻击方式，主要可以从三个角度进行：

（1）将模数 n 分解成两个素因子 p 和 q。如果分解成功，就可以计算出 $\phi(n)=(p-1)(q-1)$，再依据 $ed=1\bmod\phi(n)$，进而解得 d。

（2）不分解模数 n 的情况下，直接确定$\phi(n)$，同样可以得到 d。

（3）不确定$\phi(n)$，直接确定 d。

3.2.2 选择密文攻击

由于 n 是通过公钥传输的，因此恶意攻击者可以先利用公钥对明文进行加密，然后通过点击和试用找出其影响因素，如果任何密文与之匹配，攻击者就可以获取原始消息，从而降低 RSA 算法的安全系数。选择密文攻击是 RSA 公钥算法最常用、最有效的攻击方式之一，是指恶意攻击者事先选择不同的密文，并尝试取得与之对应的明文，由此推算出私钥或模数，进而获得自己想要的明文。例如，如果攻击者想破译消息 x 获取其签名，可以事先虚构两个合法的消息 x_1 和 x_2，使得 $x \equiv (x_1 x_2) \bmod n$，并骗取用户对 x_1 和 x_2 的签名 $S_1 = x_1^{\,d} (\bmod n)$ 和 $S_2 = x_2^{\,d} (\bmod n)$，就可以计算出 x 的签名 $S = x^d = (((x_1 x_2)(\bmod n))^d)(\bmod n) = ((x_1^{\,d} \bmod n)(x_2^{\,d} \bmod n)) \bmod n = (S_1 S_2)(\bmod n)$。

3.2.3 公共模数攻击

公共模数问题是指在公钥密码的实现过程中，一个系统中的不同用户共享同一个大整数，但却拥有不同的指数，即公钥 e。对于一个需要频繁加密的用户来说，如果这样的方法能够保证信息的安全，那么将极大地降低运营的成本。实际上，这不但不能保证安全性，还会是致命的，密码攻击者可以不需要任何私钥就能恢复出明文。最简单的一种情况是在两个用户共享同一个模数 n 的情况，又对同一明文 M 进行加密。假设密码攻击者获得了 n、e_1、e_2、C_1、C_2，则根据（3-6）和（3-7）

$$C_1 = M^{e_1} \bmod n \qquad (3\text{-}6)$$

$$C_2 = M^{e_2} \bmod n \qquad (3\text{-}7)$$

明文就能被破译。这是因为 e_1 和 e_2 互素，所以根据欧几里德(Euclid)算法能够确定 r 和 s，使得：

$$e_1 r + e_2 s = 1 \qquad (3\text{-}8)$$

由于公钥 e_1 和 e_2 都大于零，故 r 和 s 中必然有一个为负数，假设 r 为负数，再根据欧几里德算法可以计算出 $C_1^{(-1)}$，于是有

$$((C_1)^{(-1)})^{(-r)} (C_2)^s = M \bmod n \qquad (3\text{-}9)$$

由上述分析可以看出，对于同一段明文，如果选择不同的加密指数，不必做复杂的分解就能破译 RSA 密码体制。

3.2.4 小指数攻击

小指数攻击针对的是 RSA 算法的实现细节。根据算法原理，假设密码系统的公钥 e 和私钥 d 选取较小的值，算法签名和验证的效率可以得到提升，并且对于存储空间的需求也会降低，但是若 e、d 选取得太小，则可能会受到小指数攻击。例如，同一系统内的三个用户分别选用不同的模数 n_1、n_2、n_3，且选取 e=3；假设将一段明文 M 发送给此三个用户，使用各人的公钥加密得：

$$\begin{aligned} C_1 &= M^3 \bmod n_1 \\ C_2 &= M^3 \bmod n_2 \\ C_3 &= M^3 \bmod n_3 \end{aligned} \qquad (3\text{-}10)$$

为了保证系统安全性，一般要求 n_1、n_2、n_3 互素，根据 C_1、C_2、C_3 可得密文：

$$C = M^3 (\bmod n_1 n_2 n_3) \qquad (3\text{-}11)$$

如果 $M < n_1, M < n_2, M < n_3$，那么 $M^3 < n_1 n_2 n_3$，可得 $M = \sqrt[3]{C}$。

利用独立随机数字对明文消息进行填充，或者指数 e 和 d 都取较大位数的值，这样可以使得算法能够有效地抵御小指数攻击。

3.3 参数替换与多重密钥的改进方案

3.3.1 参数替换方案

RSA 的公钥和私钥中都包含参数 n，如果密码攻击者获得 n 的因素，就很容易得到私钥。为了克服 3.1.2 中所分析的针对参数 n 攻击的这一弱点，本文在传统 RSA 算法基础上，试图消除两个密钥中 n 的分布，用一种改变原模（两个素数乘积）的方法，即引入了一个新的参数 x 代替原参数 n，使得密码攻击者无法通过因子分解 n 的素因子获取私钥 d。

3.3.1.1 算法描述

改进后的 RSA 算法同样包含三个步骤，即生成的密钥、消息加密和消息解密。

（1）密钥生成

①选定两个任意的大素数 p 和 q，将其存储起来；

②计算 $n = pq$ 和欧拉函数 $\phi(n) = (p-1)(q-1)$；

③计算 x，用来代替 n，计算步骤如下：

a.如果 $q < p$，则定义 x：$(n-p) < x < n$，$GCD\ (x,n)\ =\ 1$

b.如果$q>p$，则定义x：$(n-q)<x<n$，$GCD\ (x,n)\ =\ 1$

④随机地选取一个正整数e，使其满足$\sqrt{n}<e<\phi(n)\phi(x)$，并且要求$e$和$\phi(n)\phi(x)$的最大公约数$GCD(e,\phi(n)\phi(x))=1$；

⑤计算解密密钥d，而且$ed\equiv 1(\mathrm{mod}\,\phi(x)\phi(n))$。此时，公钥为$(e，x)$，私钥$(d，x)$。

（2）消息加密

在加解密运算过程中，对明文进行分组和处理，使得每个明文的十进制数不大于x且与M互素；消息发送方使用$(e，x)$对明文M进行加密：

$$C=M^e \bmod x \qquad (3\text{-}12)$$

（3）消息解密

消息接收方使用$(d，x)$对密文C进行解密：

$$M=C^d \bmod x \qquad (3\text{-}13)$$

在四素数RSA中使用参数替换的方案，算法过程与在传统RSA算法中类似。下面给出基于四素数和参数替换的RSA公钥算法的流程图3-1：

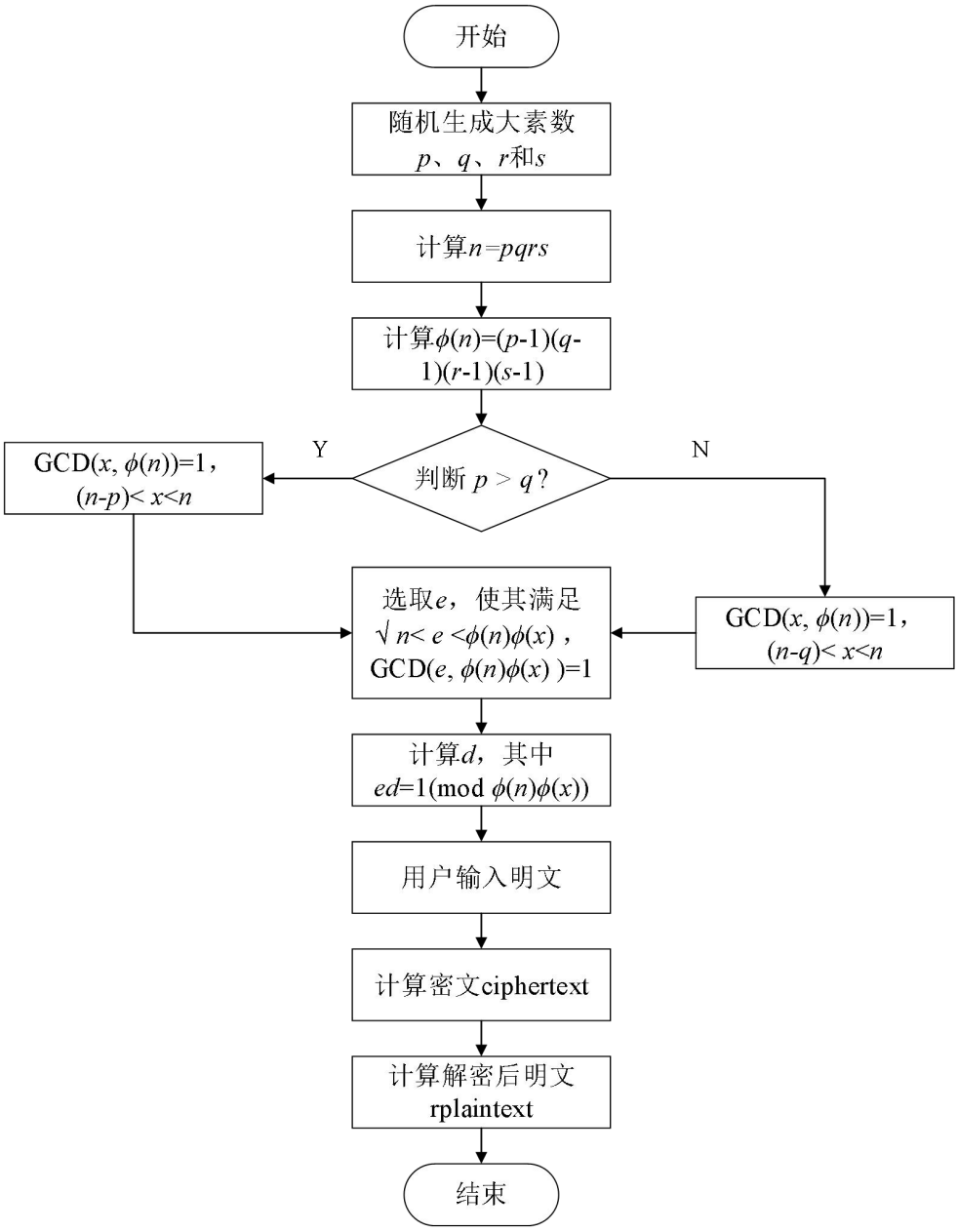

图 3-1 基于四素数和参数替换的 RSA 公钥算法

Fig 3-1 Improved RSA algorithm based on multi-prime number and parameter substitution

3.3.1.2 正确性证明

要证明改进方案的正确性，即证明加密后的密文可以通过解密变换 $C^d \bmod x$ 还原为 M。因为 $ed \equiv 1(\bmod \phi(x)\phi(n))$，故

$$\begin{aligned} & M^{ed} \bmod x \\ & = M^{ed} \bmod x \\ & = M^{1+k\phi(x)\phi(n)} \bmod x \\ & = M \times M^{k\phi(x)\phi(n)} \bmod x \end{aligned} \tag{3-14}$$

根据 2.3.4 节中的欧拉定理，可以得到：

$$M^{\phi(x)} \equiv 1 \bmod x \tag{3-15}$$

将公式（3-15）代入（3-14）可得

$$\begin{aligned} & M \times M^{k\phi(x)\phi(n)} \bmod x \\ & = M \times (1)^{k\phi(n)} \bmod x \\ & = M \bmod x \\ & = M \end{aligned} \tag{3-16}$$

综上可知，明文 M 经过加密变换得到的密文 C，可以再通过解密变换恢复为原始明文 M，这个过程可以实现对消息编码和解码的功能。

3.3.2 多重密钥方案

传统 RSA 算法中使用的是单整数密钥，文中在此基础上，给出了利用不同的公钥 e 指数对不同明文分组进行加密的方案，改进后 RSA 算法的描述如下：

（1）密钥生成

①任意选取两个不相干的大素数 p 和 q，将其存储起来；

②计算 $n = pq$ 和欧拉函数 $\phi(n)=(p-1)(q-1)$；

③选择两个或多个整数 $(e_1,e_2,e_3 \ldots e_k)$，使其满足 $1<(e_1,e_2,e_3 \ldots e_k)<\phi(n)$，$GCD((e_1,e_2,e_3 \ldots e_k),\phi(n))=1$，这里 $(e_1,e_2,e_3 \ldots e_k)$ 用作公钥指数；

④计算 $d = e^{-1}(\bmod \phi(n))$，根据(3)，有

$$\begin{cases} d_1 e_1 = \bmod(\phi(n)) \\ d_2 e_2 = \bmod(\phi(n)) \\ \ldots \\ \ldots \\ d_k e_k = \bmod(\phi(n)) \end{cases} \tag{3-17}$$

其中 $(d_1,d_2,d_3 \ldots d_k)$ 用作私钥指数。所以，公钥为 $(e_1,n),(e_2,n),(e_3,n) \cdots (e_k,n)$，私钥为 $(d_1,n),(d_2,n),(d_3,n) \cdots (d_k,n)$。

（2）消息加密

对明文进行处理，并将其分成 k 组，根据 $C=M^e \bmod n$，利用公钥 $(e_1,n),(e_2,n),(e_3,n)\cdots(e_k,n)$ 对明文分组 $M=(M_1,M_2,\cdots M_k)$ 进行加密：

$$C_i=M_i^{e_i}(\bmod n),i=1,2,...k \qquad (3\text{-}18)$$

（3）消息解密

根据 $M=C^d \bmod n$，利用私钥 $(d_1,n),(d_2,n),(d_3,n)\cdots(d_k,n)$ 对加密后的密文分组 $C=(C_1,C_2\cdots C_k)$ 进行解密：

$$M_i=C_i^{d_i}(\bmod n),i=1,2,...k \qquad (3\text{-}19)$$

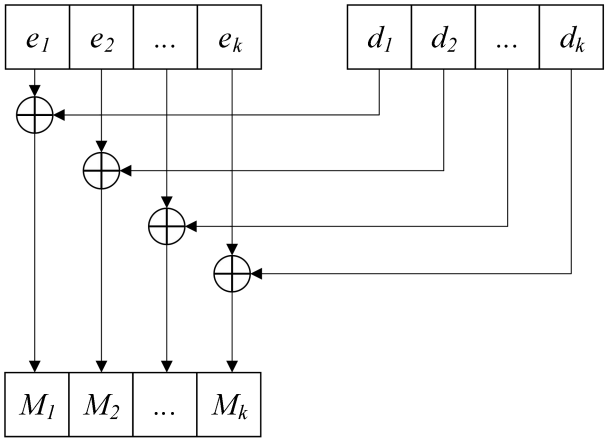

图 3-2 多重密钥 RSA 方案

Fig 3-2 Multiple Keys RSA Scheme

3.3.3 算法安全性分析

3.3.3.1 安全性评价指标

目前已知的攻击算法求出 d 所需的时间开销不低于因子分解问题的时间开销，对 RSA 算法分析的大部分讨论都集中于对模数进行因子分解。因此，因子分解的性能通常被作为算法安全性的评价指标。在计算机出现之前，对大整数进行因子分解是相当困难的；随着计算机出现，及其运算能力的不断提高，特别是基于网络的分布式计算处理，因子分解的难度逐渐降低。计算机运算性能的提高和因子分解算法的不断改进是密钥安全的主要威胁。例如，通过二次筛法（QS）或广义数域筛（GNFS）已能分解 180 多位的十进制素数，增加 p、q 的长度已成为许多安全应用系统的加密要求。另外，基于密码

系统设计的缺陷，一些基于非因子分解方式破解 RSA 算法的方案也逐渐被提出。针对大素数因子分解的算法有很多，如试除法，椭圆曲线分解法，Pollard's rho 分解法等。这些算法的特点如表 3-2 所示：

表 3-2 因式分解攻击算法对比

Table 3-2 Comparison of factorization attack algorithms

算法名称	时间复杂度	分解整数对象
试除法	$O(p(\log n)^2)$	< 110bit
Pollard's rho 分解法	$O(\sqrt{p}(\log n)^2)$	< 110bit
Pollard's P-1 分解法	$O(B\log B(\log n)^2)$	< 110bit
椭圆曲线分解法	$O(\exp((\sqrt{2}+o(1)(\ln p)^{1/2}(\ln\ln p)^{1/2}))$	< 110bit
基于完全平方的分解算法	$O(\exp((64/9)^{1/3}(\ln p)^{1/3}(\ln\ln p)^{2/3}))$	> 110bit

在参数替换的算法方案中出现的有 p、q、r、s、$\phi(n)$、x、n、e 和 d。加密端知道的只有 e 和 x，在解密端可以知道的是 p、q、r、s、x、n 和 d。其中密钥 d 可以将密文解密成为明文。如果 d 被泄露出去，那么对于明文的加密是无效的。综上所述，基于因子分解攻击的前提下，本文在传统 RSA 算法与四素数参数替换 RSA 算法的对比中，以对模数 n 进行因子分解所得到的素因子与原素因子匹配与否作为算法安全性评价指标，则系统攻破的指标可表示为：

$$\tau_{System}=\tau_{Brute-force}+\tau_{(p,q)/(p,q,r,s)} \tag{3-20}$$

其中 $\tau_{Brute-force}$ 为因子分解攻击，$\tau_{(p,q)/(p,q,r,s)}$ 为分解出的大素数的正确性。

在传统 RSA 算法与多重密钥 RSA 算法的对比中，以获取密钥 d 所需的时间为安全性评价指标，则系统攻破的指标可表示为：

$$\tau_{System}=\tau_{Brute-force}+\tau_{d} \tag{3-21}$$

其中 $\tau_{Brute-force}$ 为因子分解攻击，τ_{d} 为获取密钥 d 所需的时间。

3.3.3.2 双素数与四素数 RSA 的安全性对比

为了对传统双素数 RSA 算法和四素数 RSA 算法进行对比，在保证两种算法模数 n 长度相同的前提下，实验中使用 Robin-Miller 算法生成两个 32bit、48bit、64bit、80bit、96bit 的素数和四个 16bit、24bit、32bit、40bit、48bit

的素数，将它们分别作为传统 RSA 算法和四素数 RSA 算法中的素因子，并计算出两种算法模数 n 的值，然后用椭圆曲线的分解算法分别对两种算法中的模数进行因式分解，所耗时间如表 3-3 所示（单位：ms）。

表 3-3 因子分解时间测试与对比

Table 3-3 Contrast of factorization time

模数长度/bit	传统 RSA 算法/ms	四素数 RSA 算法/ms
64	1025	1083
96	2386	3575
128	4778	5047
160	18471	14961
192	92149	75706

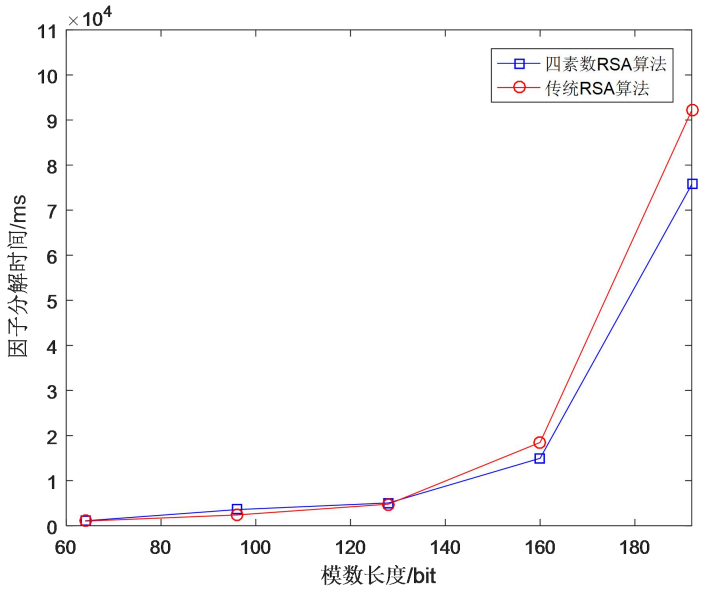

图 3-3 因子分解时间测试与对比

Fig 3-3 Contrast of factorization time

从表 3-3 中可以看出，当模数长度为 64bit 的情况下，对传统 RSA 算法模数 n 因子分解攻击的时间为 1025ms，与四素数 RSA 算法的 1083ms 几无差别；当模数长度为 96bit 和 128bit 的情况下，传统 RSA 算法所需时间低于四素数 RSA 算法约 33.4%和 4.1%；当模数长度超过 128bit 时，四素数 RSA 算

法要低于传统 RSA 算法。从图 3-3 中可以推断，当素数长度更高时，两种算法因子分解攻击耗时的差距会更大，即四素数 RSA 算法的安全强度要低于传统 RSA 算法。

根据公式 $ed \equiv 1 \bmod \phi(n)$ 可知，若要推导出 d，需要事先知道 e 和 $\phi(n)$。e 作为公钥指数，已经公开，如果攻击者能获取 $\phi(n)$，那么就可以成功分解模数 n。这是因为

$$\phi(n) = (p-1)(q-1) = pq - (p+q) + 1 = n - (p+q) + 1 \quad (3\text{-}22)$$

由（3-22）可知 $p+q = n+1-\phi(n)$，据此可以构造根为 p 和 q 的函数

$$x^2 - (n+1-\phi(n)) + n = 0 \quad (3\text{-}23)$$

对于四素数 RSA 算法，根据 $\phi(n) = (p-1)(q-1)(r-1)(s-1)$ 可知，只有知道 p、q、r 和 s，才能计算出 $\phi(n)$ 的值；由于 $n = pqrs$，若要获得 p、q、r 和 s，则必须要对大整数模数 n 进行因子分解。目前，椭圆曲线法可以分解出的素因子最高为 256bit，为了保证模数 n 的素因子不在此范围之内，对于 1024 位的 RSA 公钥算法，素因子不能超过 3 个[61]。虽然四素数 RSA 算法在效率上比传统 RSA 算法有一定程度的提升，但是由于因子分解算法的不断发展，RSA 中素数使用得越多，对算法的安全性威胁也越大。

3.3.3.3 四素数参数替换方案的安全性分析

与双素数 RSA 算法类似，四素数 RSA 算法中的密钥对同样为 (e,n) 和 (d,n)，模数 $n = pqrs$，因此对四素数 RSA 中的模数 n 进行因子分解后得到素因子与原素因子相同，即 $\tau_{(p,q,r,s)}$ 完全匹配。而在四素数参数替换 RSA 中，密钥对均消除了模数 n 的分布，以一个新的参数 x 代替原参数 n，变为 (e,x) 和 (d,x)，根据参数替换方案中模数 x 的生成原则：

（1）如果 $q < p$，则定义 x：$(n-p) < x < n$，$GCD\ (x,n) = 1$

（2）如果 $q > p$，则定义 x：$(n-q) < x < n$，$GCD\ (x,n) = 1$

可知，模数 x 并非由素因子的乘积构成，因此即使密码攻击者破译了模数 x，也无法对其进行因子分解获取 p、q、r、s 的值，即 $\tau_{(p,q,r,s)}$ 不能匹配。不能得到全部正确的素因子，也就无法推导出原始的解密密钥 d。根据安全性评价指标，在 $\tau_{Brute-force}$ 相同的情况下，相比传统 RSA 公钥算法，四素数参数替换 RSA 算法在 $\tau_{(p,q)/(p,q,r,s)}$ 方面有助于克服对因式分解攻击的弱点，在算法进行加密和签名之前进一步提高了算法的安全性。

3.3.3.4 多重密钥方案的安全性分析

由于传统 RSA 算法和多重密钥 RSA 算法都包含两个素因子，因此只需比较安全性评价指标中的获取密钥 d 的时间 τ_d。为了对获取 d 所需的时间进行充分对比，文中用欧几里得算法分别计算传统 RSA 算法和多重密钥 RSA 算法在素数长度为 256bit、512bit、1024bit，明文长度为 256bit 情况下获取密钥 d 的时间，多重密钥 RSA 算法中的参数 k 选取 5、10、15，即需要分别生成 5 对、10 对和 15 对密钥对。实验结果如表 3-4 所示（单位：ms）：

表 3-4 密钥 d 攻击的时间

Table 3-4 The time to obtain the key d

素数长度/bit	传统 RSA 算法	多重密钥 RSA 算法		
	k=1	k=5	k=10	k=15
128	9.2	14.1	19.5	36.8
256	13.3	23.7	34.6	53.2
512	29.2	32.9	55.3	83.4
1024	58.9	74.5	104.1	165.3

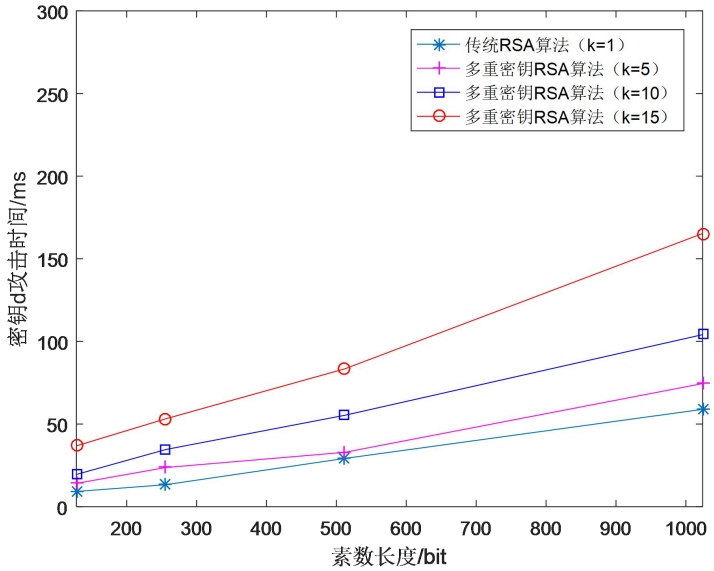

图 3-4 密钥 d 攻击时间对比

Fig 3-4 Contrast of factorization time in private key d

通过对表 3-4 的分析可知，当 k 取值越大时，获取 d 时间 τ_d 越长。从图 3-4 中可以观察到，当素数长度为 128bit 和 256bit 时，即多重密钥 RSA 算法在对于密钥 d 的攻击时间提升较少；当素数长度相对较高时，算法效果比较明显，实际中，k 取值要远大于所选的值。多重密钥 RSA 算法基于两个大素数将明文分成 k 组，使用 k 个密钥，相较于传统 RSA 公钥算法，密码攻击者在进行同等 $\tau_{Brute-force}$ 的前提下，仍需攻击并获取每段明文的密钥，才能破译全部的明文信息。另外，对于多重密钥 RSA 算法，3.1.2 节所分析的公共模攻击中的公式（3-4）和（3-5）变为 $C_1 = M_1^{e_1} \bmod n$ 和 $C_2 = M_2^{e_2} \bmod n$，无法据此得到 $((C_1)^{(-1)})^{(-r)}(C_2)^s = M \bmod n$。由于使用不同密钥对不同分组明文进行加密，即每段明文所依赖的加密指数的不同的，因此改进的方案可以很好地避免公共模数攻击。

3.4 基于同余对称特性的 SMM 算法的改进

与对称密码算法相比，RSA 公钥算法虽然克服了密钥传递困难的不足，但其自身也存在效率方面的问题，如算法中大素数的选择，因为位数要求越来越多，降低了素数产生的效率；运用算法对数据进行加解密的过程中需要进行大量模幂运算，密钥指数越复杂，数据处理的速度就越低，从而对算法的效率产生了很大的影响。因此，对 RSA 算法中的某些关键算法进行局部的优化，以加快其运行速度，是十分有意义且必要的。

随着人们对 RSA 公钥算法的不断研究与分析，关于改进 RSA 中某些局部特性的算法也在陆续地出现。常用的优化大数模幂的算法有基于乘同余对称特性的 SMM 算法、Montgomery 算法、指数 2^k 进制化算法等[62-63]。本文对 SMM 算法进行了分析，并给出相关的优化策略。

3.4.1 SMM 算法

传统 RSA 公钥算法中利用平方乘 BR(Binary Representation)算法来进行大整数的模幂运算，它的思想是将乘幂后再取模的运算转化成为重复的平方取模和相乘后取模的迭代运算。加密过程是对明文求幂剩余的过程，BR 算法先将指数转化为二进制的表示形式，再将幂剩余变成一系列乘同余的迭代。以 $C=M^E \bmod n$ 为例，先将指数 E 转化成二进制数的序列 $(e_{t-1}e_{t-2}e_{t-3}..e_1e_0)$，$t$ 表示指数 E 的二进制位数，所以 E 可以表示为

$$E=e_{t-1}2^{t-1}+e_{t-2}2^{t-2}+e_{t-3}2^{t-3}+\cdots+e_1 2+e_0 2^0=\sum_{k=0}^{t-1}e_k 2^k \qquad (3\text{-}24)$$

其中 e_k 的取值为 0 或 1。将公式（3-24）代入 $C=M^E \bmod n$，可得

$$\begin{aligned} C&=M^{\sum_{k=0}^{t-1}e_k 2^k} \bmod n \\ &=(((M^{e_{t-1}} \bmod n)^2 \bmod n\times M^{e_{t-2}} \bmod n)^2 \bmod n\cdots M^{e_1} \bmod n)^2\times M^{e_0} \bmod n \end{aligned} \qquad (3\text{-}25)$$

根据公式（3-25）可知，在计算 $C=M^E \bmod n$ 的过程中，每一步的迭代必会出现运算 $A=A^2 \bmod n$，可能会出现运算 $A=(A\times B)\bmod n$，其中 A 为迭代的中间结果，B 为待加密的明文，A、$B\in\{0,1,2,3\cdots,n-1\}$。

基于乘同余对称特性的 SMM(Symmetry of Modulo Multiplication)算法是用于降低加解密过程中乘法和求模运算量的一种快速算法，建立在 BR 算法的基础之上。在每一步的迭代计算中，SMM 算法对乘数和被乘数进行有条件的代换，具体方式为：A_{i-1} 表示第 i-1 步的迭代结果，那么到第 i 步的迭代时

（1）若 $B\le(n-1)/2$ 或 $A_{i-1}\le(n-1)/2$，则保持原数不变；

（2）若 $B>(n-1)/2$ 或 $A_{i-1}>(n-1)/2$，则使用 $n-B$ 或 $n-A_{i-1}$ 来分别地替代 B 或 A_{i-1}。

第一步主要是降低平方模运算的复杂度，第二步是降低模乘运算的复杂度，下面以 $A^2 \bmod n$ 和 $(AB)\bmod n$ 为例，证明算法正确性：

用 $(n-A)$ 替换 A，可得

$$(n-A)^2 \bmod n=(n^2-2An+A^2)\bmod n=A^2 \bmod n \qquad (3\text{-}26)$$

用 $(n-A)$ 替换 A，$(n-B)$ 替换 B，可得

$$(n-A)(n-B)\bmod n=(n^2-An-Bn+AB)\bmod n=(AB)\bmod n \qquad (3\text{-}27)$$

SMM 算法减小了运算中部分乘数和被乘数的绝对值，降低了求模运算量和乘法时间，因此使得 RSA 公钥算法在一定程度上得到了改善。

3.4.2 算法的改进过程

在 SMM 算法中，若令 $X=AB$ 或 $X=A^2$，将 X 写成二进制的形式：

$$X=\sum_{i=0}^{k-1}X_i 2^i \qquad X_i\in\{0,1\} \qquad (3\text{-}28)$$

其中 k 为 X 的二进制长度，那么 $(AB)\bmod n$ 或 $A^2 \bmod n$ 可表示为：

$$\begin{aligned} X \bmod n &= (\sum_{i=0}^{k-1} X_i 2^i) \bmod n \\ &= (\sum_{i=0}^{k-1} X_i (2^i \bmod n)) \bmod n \\ &= (\sum_{i=0}^{k-1} X_i r_i) \bmod n \end{aligned} \qquad (3\text{-}29)$$

其中 r_i 为 2^i 对 n 的余数，$r_i \in \{0,1,\cdots n-1\}$。因为 X_i 的取值只有 0 和 1，当 $X_i = 0$ 时，$X_i r_i = 0$；当 $X_i = 1$ 时，$X_i r_i = r_i$，所以 $X \bmod n$ 就被转化成了一系列余数 r_i 的和。又因为

$$r_i \equiv 2^i \bmod n \equiv 2(2^{i-1} \bmod n) \bmod n \equiv 2r_{i-1} \bmod n \qquad (3\text{-}30)$$

为了计算的方便起见，需要事先建立一个余数表，将预先产生的 2^i 的余数依次填入进去，当在进行求模运算的时候，只要把被模数的非零位找出来，查出表中对应的余数，然后进行求和即可。下面以 $20 \bmod 3$ 为例具体说明，$2^i \bmod 3$ 的余数表如表 3-5 所示：

表 3-5 余数表

Table 3-5 Remainder table

2^i	2^0	2^1	2^2	2^3	2^4	2^5
r_i	1	2	1	2	1	2

由于 $20 = (10100)_2$ 的非零位为 2^4 位和 2^2 位，查表可知 r_2 和 r_4 都等于 1，因此 $20 \bmod 3 = 1+1 = 2$，而直接计算 $20 \bmod 3$ 也等于 2，所得的结果是相同的。可以看出，被模数的二进制形式中所包含的非零元素的个数越少，即汉明重量(Hamming Weight)越小，余数求和的次数就越少，速度也就因此提升。如果能够降低 SMM 算法的指数二进制序列的汉明重量，则算法的迭代次数也将会随之减少。

容易证明，连续的二进制数“1”可表示成公式（3-31）的形式，这里用“$\bar{1}$”表示“-1”，

$$\begin{cases}(11)_2=(100)_2-(1)_2=10\bar{1}\\(111)_2=(1000)_2-(1)_2=100\bar{1}\\(1111)_2=(10000)_2-(1)_2=1000\bar{1}\\\cdots\\(\underbrace{11\cdots1}_{N个})_2=(\underbrace{100\cdots}_{N个})_2-(1)_2=\underbrace{10\cdots\bar{1}}_{N个}\end{cases}\tag{3-31}$$

如果运算中“110”和“1110”码型之后出现 k 个 $(k\geq 2)$ 连续的“1”时，也可以继续进行下面的二次代换

$$\begin{cases}(11011)_2=(100000)_2-(1)_2-(100)_2=10000\bar{1}-(100)_2=100\bar{1}0\bar{1}\\\cdots\\(\underbrace{11101\cdots1}_{N个})_2=\underbrace{1000\bar{1}0\cdots0\bar{1}}_{N-1个}\end{cases}\tag{3-32}$$

在公式（3-31）和（3-32）中，等号右边由1、0、$\bar{1}$ 构成的数称为二进制冗余数。不难看出，当连续“1”大于等于三个或者“110”、“1110”码型后面的连续“1”大于等于两个，二进制冗余数的汉明重量要低于原始二进制数的汉明重量。用上述的方法对二进制序列中三个以上的连续“1”以及“110”、“1110”码型后面两个以上的连续“1”进行替换。假设 $e=(101110101111)_2$，则 e 的二进制冗余数 $R(e)=(11000\bar{1}0\bar{1}000\bar{1})_2$，容易验证 $e=R(e)$。e 的汉明重量 $h(e)$ 为 9，而汉明重量 $h[R(e)]$ 仅为 5，可见二进制冗余数的汉明重量要低于原始二进制数的汉明重量。

以 $R(e)$ 来替代 e，则对明文 M 的加密可以表示为

$$C=M^{R(e)}\bmod n\tag{3-33}$$

其中 $R(e)=\sum_{i=0}^{k}e_i2^i$，$k=$ N或N-1，$e_i\in\{0,1,\bar{1}\}$。

对（3-33）进行迭代计算时，包含 $A_iB(\bmod n)$，$A_iB^{-1}(\bmod n)$ 和 $A_i^2(\bmod n)$ 三种基本运算。B^{-1} 是 B 对模数 n 的乘逆。

在前面的章节中提到过，欧拉函数 $\phi(n)$ 表示小于 n 且与 n 互素的数的个数，欧拉函数 $\phi(n)$ 与模数 n 之比为：

$$\begin{aligned}&\frac{\phi(n)}{n}\\&=\frac{(p-1)(q-1)}{pq}\\&=1-\frac{1}{p}-\frac{1}{q}+\frac{1}{pq}\end{aligned}\tag{3-34}$$

当所选取的 n 是 100 位的十进制数时，公式（3-34）中 $\phi(n)$ 与 n 的比值趋近于 1，说明几乎只要满足 $M\in\{0,1,\ldots,\ n-1\}$ 的 M 都与 n 互素，即

$$GCD(M,n)=1\tag{3-35}$$

因此，M 对模数 n 的乘逆 M^{-1} 一定存在。从这一点出发，可以构造一种更快速的 RSA 算法，具体的操作步骤如下:

（1）首先将加密指数 e 转换成二进制冗余数的形式 $R(e)$。自加密指数 e 的最高位开始，如果二进制指数 e 的序列中有 $k\geq 3$ 个连续的“1”，那么将这 k 个连“1”替换成 $1\underbrace{0\cdots 0}_{k-1}\bar{1}$。当序列中在“110”或“1110”码型之后有两个或以上连续的“1”时，也可以继续进行替换。

（2）根据欧几里得算法，可以由 $GCD(M,n)=1$ 计算出 M 对模数 n 的乘逆 M^{-1}。

（3）令

$$M=\begin{cases}M & M\leq\dfrac{n-1}{2}\\ n-M & M>\dfrac{n-1}{2}\end{cases}\tag{3-36}$$

$$M^{-1}=\begin{cases}M^{-1} & M^{-1}\leq\dfrac{n-1}{2}\\ n-M^{-1} & M^{-1}>\dfrac{n-1}{2}\end{cases}\tag{3-37}$$

（4）由 $R(e)$ 的最高位开始，迭代计算幂剩余。若 A_i 为第 i 步迭代计算的中间结果，那么可以得到具体的算法为：

① $$A_i=\begin{cases}A_i & A_i\le\dfrac{n-1}{2}\\ n-A_i & A_i>\dfrac{n-1}{2}\end{cases}$$ （3-38）

$$A_{i+1}=A_i^2(\mathrm{mod}\,n)$$

② $$A_i=\begin{cases}A_i & A_i\le\dfrac{n-1}{2}\\ n-A_i & A_i>\dfrac{n-1}{2}\end{cases}$$ （3-39）

$$A_{i+1}=A_iM(\mathrm{mod}\,n)$$

③ $$A_i=\begin{cases}A_i & A_i\le\dfrac{n-1}{2}\\ n-A_i & A_i>\dfrac{n-1}{2}\end{cases}$$ （3-40）

$$A_{i+1}=A_iM^{-1}(\mathrm{mod}\,n)$$

当 $R(e)$ 的第 i 位为“0”时，进行算法步骤①的计算；当第 i 位为“1”时计算步骤②和③；当第 i 位为“$\bar{1}$”时，进行算法步骤①和③的计算，直至 $R(e)$ 的每一位都被计算完为止。相比 SMM 算法的迭代步数：

$$l=\mathrm{N}+h(e)-2 \qquad (3\text{-}41)$$

改进的算法迭代步数为：

$$l'=\mathrm{N}+h[R(e)]-2 \qquad (3\text{-}42)$$

3.4.3 改进 SMM 算法的速度分析

下面分析改进后的 SMM 算法对于运算速度的优化。依据 Golomb 随机性公设，一个随机二进制数序列中出现 k 个连续“1”的概率为 $1/2^k$，在 RSA 密码系统中，公钥指数 e 的长度一般取 300bit 以上，则三个以上连续“1”出现的概率约为：

$$p=\sum_{k=3}^{n}\frac{1}{2^k} \qquad (3\text{-}43)$$

当 n 趋近于无穷时，p 的极限概率为 1/4。“110”和“1110”码型出现的概率分别为 $1/2^3$ 和 $1/2^4$，二者之后出现 $k(k\ge2)$ 个连续“1”的条件概率仍

为$1/2^k$，因此“110”、“1110”码型和$k(k\geq 2)$个连续“1”发生的联合概率为$1/2^{k+3}$和$1/2^{k+4}$，由此可知，对上述情况进行二次代换的可能性分别为：

$$p_1=\sum_{k=2}^{n-3}\frac{1}{2^{k+3}} \qquad (3\text{-}44)$$

$$p_2=\sum_{k=2}^{n-3}\frac{1}{2^{k+4}} \qquad (3\text{-}45)$$

公式（3-44）和（3-45）的极限概率分别为$1/16$和$1/32$。仅考虑一次和二次代换的情况，二者比特数占总比特数的的比率为：

$$p_3=(1/4)+(1/16)+(1/32)\approx 34.4\% \qquad (3\text{-}46)$$

若将k个连续“1”出现的概率考虑在内，那么在超过$1/3$的代换中，汉明重量降低了一半，因此平均减少的迭代步数大约为$1/2\times 34.4\%=17.2\%$。可以看出，改进后的算法对降低运算的迭代步数有一定的效果。

3.4.4 测试及结果分析

实验中选择素数p=97、q=157，则模数$n=15229$，欧拉函数$\phi(n)=14976$。根据$ed=1\bmod\phi(n)$和$GCD(e,\phi(n))=1$，用欧几里得算法生成20个随机密钥对，将每个密钥对$(e_i,d_i)(i=1,2,\cdots,20)$分别表示成二进制数和二进制冗余数的形式，再根据（3-41）和（3-42）分别计算SMM算法和改进算法的迭代步数，结果如表3-6所示：

表 3-6 SMM算法和改进算法迭代步数的对比

Table 3-6 Comparison of iteration number between SMM arithmetic and the improved arithmetic

序号 i	密钥对 (e_i,d_i)	SMM算法迭代步数 l	改进算法迭代步数 l'	迭代步数降低率 $(1-l'/l)\times 100\%$
1	(509,3413)	32	28	12.5%
2	(935,6551)	34	32	5.9%
3	(127,10495)	34	24	29.4%
4	(347,3539)	31	29	6.5%
5	(1259,1475)	31	28	9.7%
6	(191,5567)	34	25	26.5%

7	(251,179)	24	21	12.5%
8	(1057,8161)	32	27	15.6%
9	(1021,11221)	38	31	18.4%
10	(485,2285)	30	27	10%
11	(1939,7963)	36	32	11.1%
12	(2045,725)	33	27	18.2%
13	(1871,13103)	38	34	10.5%
14	(7897,11113)	40	36	10%
15	(3743,11231)	41	31	24.4%
16	(4607,6911)	43	30	30.2%
17	(9215,10367)	43	31	27.9%
18	(3059,12347)	38	32	15.8%
19	(8189,7253)	41	33	19.5%
20	(7487,7487)	42	34	19%

对表 3-6 中数据进行分析可知，改进后算法的迭代效率比 SMM 算法平均提升了16.68%，与理论值 17.2%接近。实验中的密钥对虽然只有 20 组，但是因为是随机生成的，通过对比，足以表明改进的 SMM 算法对效率优化的效果。又由于在 BR 算法的基础上，SMM 算法可以使求模运算中的迭代效率平均提升 30%以上，因此改进的 SMM 算法相较 BR 算法，效率提升为 30%+17.2%×70%=42%。除了降低了求模运算量外，SMM 算法还降低了乘法时间，因此实际在效率上的提升要大于 42%。

3.5 本章小结

本章所介绍的算法是课题的核心内容，目前针对 RSA 算法的攻击研究越来越深入，关于如何保证算法的安全性，文中提出了使用四素数与参数替换和多重密钥的改进方案，将传统 RSA 算法，四素数参数替换方案和多重密钥方案的安全性进行对比分析，表明了改进方案在安全方面的优势。对于加解密算法执行时间长的这一瓶颈，给出了相应的解决方案，在研究与分析现有的基于乘同余对称性 SMM 算法的基础上，对其做出了进一步的优化，并对优化的效果进行了分析与实验。结果表明，改进的 SMM 算法对于模幂乘运算的效率有明显的提升。

4 RSA 公钥算法在数字签名中的应用

基于公钥体制的 RSA 算法为信息安全传输的问题提供了新的解决思路和技术，也被用作数字签名方案，在实现消息认证方面得到了深入的应用。总的来说，RSA 公钥签名方案包括、消息空间、参数生成算法、签名算法和验证算法等几个部分。签名和验证的过程具体包括消息摘要的生成、大素数和密钥的生成以及消息签名、消息验证等几个步骤。本章详细分析了 RSA 公钥算法在数字签名中的应用。

4.1 RSA 签名方案

4.1.1 算法签名过程

传统的签名方案使用 RSA 公钥算法进行数字签名，签名的过程如下:

（1）参数的选择和密钥的生成；

（2）签名过程：用户 A 对消息 M 进行签名，则计算

$$S = Sig(M) = M^d \bmod n \qquad (4\text{-}1)$$

然后再将 S 作为用户 A 对消息 M 的数字签名附在消息 M 后面，发送给用户 B；

（3）验证过程：用户 B 验证用户 A 对 M 的数字签名 S，则计算

$$M' = S^e \bmod n \qquad (4\text{-}2)$$

然后判断 M' 和 M 是否相等，若二者相等，则可以证明签名 S 的确来源于用户 A，否则，签名 S 有可能为伪造的签名。

例如，假设用户 A 选取 p=823、q=953，那么模数 n=784319、$\phi(n)$=782544，然后选取 e=313 并计算出 d=160009，则公钥为 (313,784319)，私钥为 (160009,784319)。如果现在用户 A 拟发送消息 M=19070 给用户 B，用私钥对消息进行签名：

$$M:19070 \rightarrow S = (19070)^{160009} \bmod 784319 = 210625 \bmod 784319$$

用户 A 将消息和签名同时发送给用户 B，用户 B 接收消息和签名，并计算出 M'：

$$M' = 210625^{313} \bmod 784319 = 19070 \bmod 784319 \rightarrow M = M' \bmod n$$

因此，用户 B 验证了用户 A 的签名，并接收了消息。

4.1.2 传统签名方案分析

通过对上述的签名过程分析之后，传统 RSA 数字签名方案依然存在着若干缺陷：

（1）假设 S_1、S_2 分别是消息数据 M_1、M_2 的签名，由于在使用传统 RSA 公钥算法的签名方案中，有

$$Sig(M_1M_2)=Sig(M_1)Sig(M_2) \qquad (4\text{-}3)$$

所以任何人只要知道 M_1、M_2、S_1、S_2，就都可以伪造签名 S_1、S_2。

（2）由于对任意的 S，任何人都可以计算出 $C=M^e \bmod n$，所以可以随意地伪造对消息 M 的签名 S。

（3）在实际的应用过程中，待签名的消息一般都比较长，因此就需要将消息明文先进行分组，然后再对不同的分组明文分别进行签名。这样会致使算法对于长文件签名的效率十分低下。

4.2 签名方案实现的关键步骤及其改进

若要解决上述 RSA 公钥算法签名方案中的不足，可以利用单项函数，即在对消息进行签名之前，事先使用 Hash 函数对需要签名的消息做 Hash 变换，对变换后的消息再进行数字签名。

4.2.1 消息摘要

Hash 函数作为密码学中的一个重要组件，可以将任意长度的消息明文压缩为一段长度固定的消息摘要，可分为基于数学困难性问题、基于置换函数等类型[64]。函数的输入为某一任意长度的消息明文，输出为长度固定的摘要。Hash 函数中的一个键值可以对应多个真实值，而一个真实值只能够对应一个键值。RSA 公钥算法的运算效率较低，并且由于算法中的模幂运算的同构性的特点，会致使签名容易被攻击者伪造，将 Hash 函数应用到利用 RSA 公钥算法对消息明文进行数字签名的方案中，可以为数据的存取提供一种快速的方法，提高签名的速度，也为数据的修改提供了极大的便利。

当签名方拟对一段消息明文 m 进行签名时，可以首先利用 Hash 函数 H 计算出消息明文 m 的消息摘要 $z=H(m)$，然后计算签名 $y=Sig(d(z))$。另外，Hash 函数也可用于对消息明文完整性的检测。发送方在进行消息传输之前，先对消息做 Hash 变换，然后再传输消息明文，接收方在收到消息明文之后，同样对消息明文也做 Hash 变换，再将得到 Hash 变换值与消息传输前的 Hash 变换值相比较，如果两次的 Hash 变换值相同，那么可以认为传输过程中的

消息明文没有被篡改，否则必然已被非法篡改。常见的 Hash 函数有 MD4、MD5 和 SHA，包括 SHA-256、SHA-512[65]。MD5 具有较好的单向性和抗碰撞性，并且执行的速度快，本文采用 MD5 来进行消息摘要的计算。MD5 将输入的消息明文以 512 位为单位来分组处理，每一分组又被划分为 16 个每 32 位一组的子分组，经过后续的处理之后，算法的输出由四个 32 位分组组成，然后将这四个 32 位分组进行级联，生成一个 128 位散列值，最后得出结果。具体的实现步骤如下：

（1）首先填充数据，使得填充后的数据字节长度为 $n\times512$（n 为正整数），接着在数据后填充一个 1 和充分多的 0，直至数据长度满足 $n\times512-64$，即填充后的消息比 512 的倍数少 64 位。

（2）用一个 64 位二进制数表示原始数据的长度，将这个数字附加在填充后的数据的尾端。经过 1、2 步的处理后，新的数据字节长度恰为 512 的整数倍，满足下一步的处理过程对数据长度的要求。

（3）将数据以 512 位为单位进行分组处理，定义链接变量 A=0x01234567，B=0x89ABCDEF，C=0xFEDCBA98，D=0x76543210。当链接变量设置好后，算法开始进入四轮循环运算，循环运算的次数为 512 位数据分组的数目。接着将 A、B、C 和 D 四个链接变量复制到 a、b、c 和 d 另外四个变量中。

（4）定义 MD5 的四个非线性操作函数，对数据进行变换，每轮一个：

$$\begin{aligned} F(X,Y,Z) &= (X\,\&\,Y)\,|\,((\sim X)\,\&\,Z);\\ G(X,Y,Z) &= (X\,\&\,Z)\,|\,(Y\,\&(\sim Z));\\ H(X,Y,Z) &= X\,\hat{}\,Y\,\hat{}\,Z;\\ I(X,Y,Z) &= Y\,\hat{}\,(X\,|\,(\sim Z)); \end{aligned} \tag{4-4}$$

其中，&、~、|和^分别表示与、非、或和异或操作。若 X、Y 和 Z 的对应位是均匀且独立的，则运算结果的每一位也应是均匀且独立的。

再定义分别用于四轮变换的四个函数 FF、GG、HH 和 II，四轮的结构相似，每一轮变换的逻辑函数如表 4-1 所示：

表 4-1 逻辑函数

Table 4-1 Logical functions

FF(a, b, c, d,M[k], S,T[i]) :
a ← b + ((a + F(b, c, d) + M[k] + T[i]) << S)；
GG(a, b, c, d,M[k], S,T[i]) :

a ← b + ((a + G(b, c, d) + M[k] + T[i]) << S)；
HH(a, b, c, d,M[k], S,T[i]) :
a ← b + ((a + H(b, c, d) + M[k] + T[i]) << S)；
II (a, b, c, d ,M[k], S,T[i]) :
a ← b + ((a + I (b, c, d) + M[k] + T[i]) << S) 。

其中，M[k]表示数据的第 k 个子分组，取值从 1 到 15；T[i]为 $(2^{32})\times abs(\sin(i))$ 的整数部分，i 取值从 1 到 64，单位为弧度；<< S 表示循环左移 S 位。每一循环由对缓存 A，B，C，D 的 16 步操作组成，第四次循环的输出加到第一次循环的四个缓存输入作为处理下一分组的输入。

（5）输出最后的消息摘要是 abcd 的级联形式。

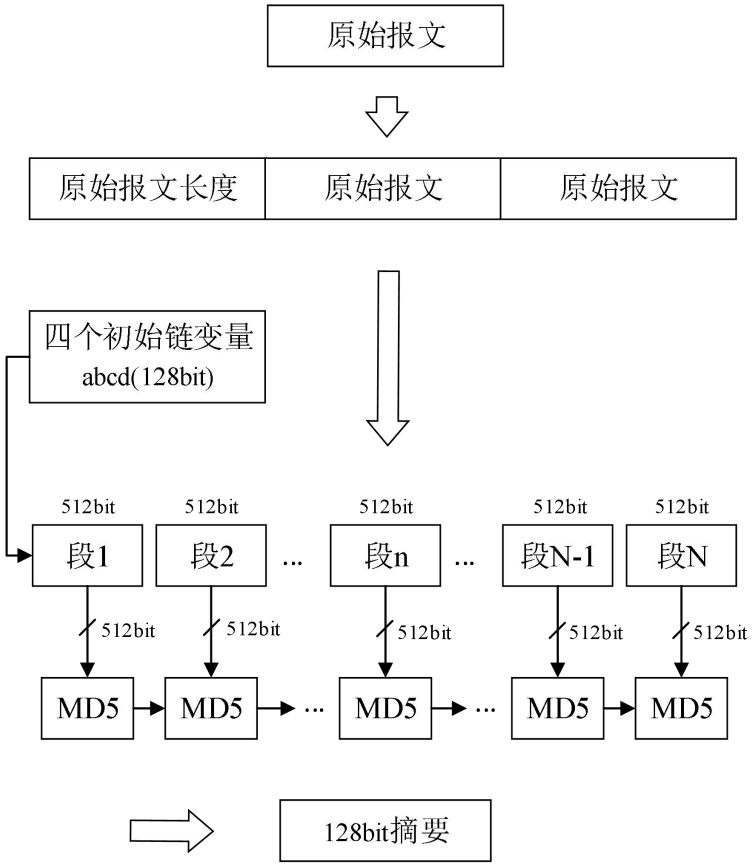

图 4-1 MD5 工作流程图

Fig 4-1 The flow chat about MD5

4.2.2 随机大素数的生成

通过前面对 RSA 算法的描述可知，算法中的第一步是寻找大素数，选取出符合要求的大素数不仅是后续步骤的基础，也是保障 RSA 算法安全性的基础。在正整数序列中，素数具有不规则分布的性质，无法用固定的公式计算出需要的大素数或者证明所选取的数是否为一个素数；从耗费和安全的角度而言，也不可能事先制作并存储一个备用的大素数表。因此，选取一个固定位数的大素数存在着一定的难度。检测素数的方法大体分为确定性检测方法和概率性检测方法两类。确定性检测方法有试除法、Lucas 素性检测，椭圆曲线测试法等，这类测试法的计算量太大，使用确定性检测方法判定一个 100 位以上的十进制数的素性，以世界上最快的计算机需要消耗 10^3 年，这种方法在实际应用中显然没有意义。在 RSA 算法的应用中一般都使用概率性检测方法来检测一个大整数的素性，虽然可能会有极个别的合数遗漏，但是这种可能性很小，在很大程度上排除了合数的可能，速度较确定性检测方法也快得多。常见的有费马素性检测法、Solovay-Strassen 检测法、Miller-Rabin 测试法，其中 Miller-Rabin 测试法容易理解，效率较高，在实际中应用也比较广泛。

4.2.2.1 Miller-Rabin 测试法

在前面介绍了费马小定理，即若 a 为一个正整数，p 为一个素数，且 a 与 p 互素，则有

$$a^{p-1} \equiv 1 (\mathrm{mod}\ p) \qquad (4\text{-}5)$$

但是当（4-5）成立时，即若有正整数 a 与 p 满足上式，并不能说明 p 一定为素数，此时称 p 为以 a 为基的伪素数。基于上述条件，设 $p-1=2^s t$，s 为非负整数，t 为正奇数。如果 a 与 p 互素，$a^t \equiv 1 \mathrm{mod}\ p$ 或存在 $0 \le r < s$，使得

$$a^{2^r s} \equiv -1 (\mathrm{mod}\ p) \qquad (4\text{-}6)$$

那么 p 称为以 a 为基的强伪素数。判断一个数是否为强伪素数，可以用 Miller-Rabin 算法来测试。

定义 4.1：设待测数 n 是一个奇数且 $n>2$，那么 n 一定可以写成 $n-1=2^k m$ 的形式，其中 k 为非负整数，m 为正奇数。设 $0<b<n$，若 $b^m \equiv 1 \bmod n$ 或者存在 $0<r\le k$，使得

$$b^{2^r m} \equiv -1 (\bmod n) \qquad (4\text{-}7)$$

那么称 n 通过了以 b 为基的 Miller-Rabin 素数检测。

由此可见，费马小定理是 Miller-Rabin 检测算法的理论基础，并且若 n 是素数，那么对于任意小于 n 的正整数，都有 $b^{n-1}=1 \bmod n$ 成立。Miller-Rabin 测试算法步骤如下：

（1）先用伪随机数生成器随机地产生一个奇数 n；

（2）将被测数 n 转化为 $n-1=2^k m$，计算出 k 和 m；

（3）随机选取 $b\in(1,n)$；

（4）$i=0$，计算 $z=b^m \bmod n$；

（5）若 $z=1$ 或者 $z=n-1$，则通过测试，n 可能为素数；

（6）若 $i>0$ 且 $z=1$，则 n 不是素数；

①$i++$，若 $i<k$ 且 $z\ne n-1$，令 $z=z^2 \bmod n$，然后返回（5）；若 $z=n-1$，则 n 可能是素数；

②若 $i=k$ 且 $z\ne n-1$，则 n 不是素数。

通过 Miller-Rabin 检测的数是合数的最大概率为 $1/4$。如果选取 t 个不同的 b 进行算法的运算，则通过 t 次 Miller-Rabin 素性测试后，被测数 n 为合数的概率不超过 $(1/4)^t$，即被测数 n 为素数的概率不小于 $1-(1/4)^t$。虽然不可能取尽所有的整数 b，只能随机选取若干个整数来验证被测数 n 是否为素数，但验证的次数越多，n 为素数的概率就越大。例如，当 $t=5$ 时，可以计算出 n 为素数的概率为 99.9%[66]。可见，Miller-Rabin 测试法提供了极其可靠的结果证明 n 是素数，测试流程图如图 4-2 所示：

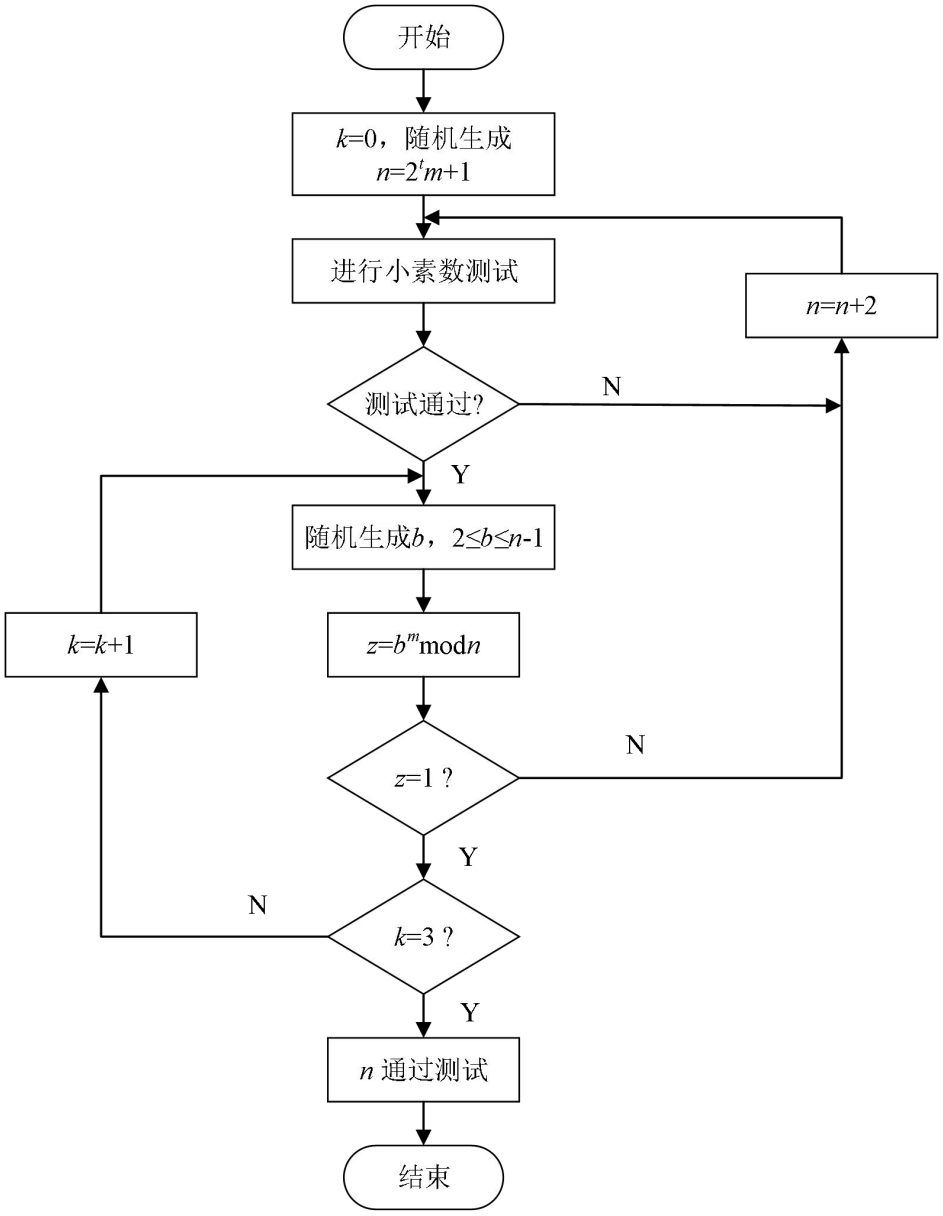

图 4-2 Miller-Rabin 测试法流程图

Fig 4-2 The flow chat about Miller-Rabin Test

4.2.2.2 改进的 Miller-Rabin 测试法

通过对 Miller-Rabin 测试法的分析可知，算法首先确定一个随机数 n，并且满足 $n=2^t m+1$，其中 t 为非负整数，m 为正奇数。本文对 Miller-Rabin 算法进行了改进，即首先确定一个满足算法要求的大整数的搜索范围，接着选取一个随机数，再以递增的方式在此范围内进行搜索，直到确定出一个素数。在实际生成随机数 n 过程中，使用下面两中方案实现：

（1）传统的计算随机数 n 的方法。先生成随机数 n，再计算 t 和 m；

（2）与（1）的方式相反，首先生成随机数 t 和 m，再根据 $n=2^t m+1$ 求出 n。

实验中需要进行大量模幂运算 $x^r \bmod p$，图 4-3 给出模幂运算的流程图：

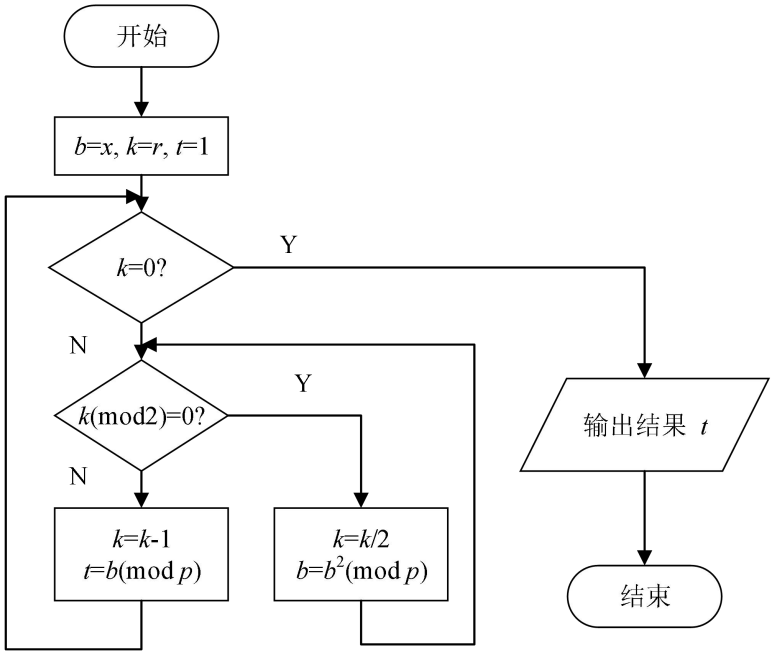

图 4-3 模幂运算流程图

Fig 4-3 The flow chat about modular exponentiation

根据 Miller-Rabin 测试法的原理，以生成 7 位十进制的随机素数为例，对上述两种方法进行了实验，分别产生 5 组（每组 500 个）随机素数，除了产生随机数 n 以外，实验中其余的函数都相同。记录两种方案产生随机素数所需的时间以及相邻两次时间差的绝对值，如表 4-2 所示（单位：ms）。

表 4-2 两种方案每产生 500 个随机素数所需时间及时间差

Table 4-2 The difference of time required for generating 500 random prime numbers between two schemes

实验	一	二	三	四	五	总计	平均
方案一	1541	1061	1201	1985	1012	6800	1360
间隔	0	480	140	784	973	2377	594
方案二	765	530	579	640	515	3029	606
间隔	0	235	49	61	125	470	118

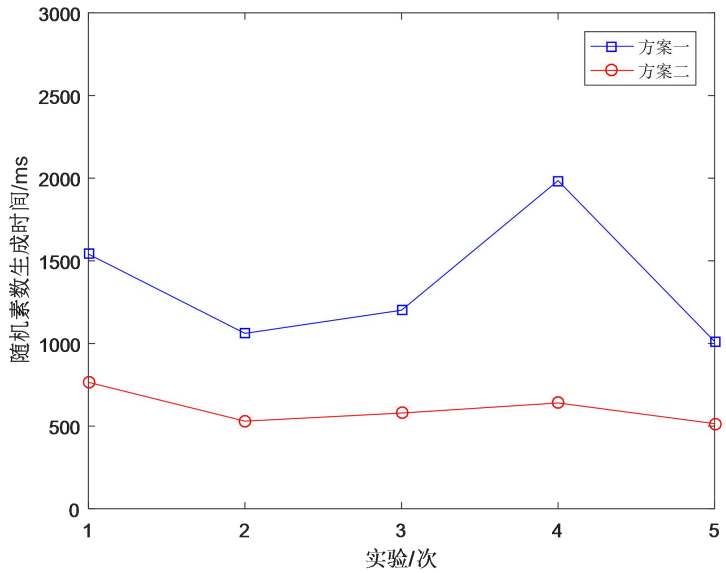

图 4-4 随机素数生成时间对比

Fig 4-4 Comparison of random prime number generation time

从表 4-2 中可知，第一种方案中，5 组实验总共耗时 6800ms，每组耗时从 1012ms 到 1985ms 不等，时间差比较大；方案二总共耗时 3029ms，每组耗时从 515ms 到 765ms 不等，时间差相对较小。每生成 500 个 7 位十进制的素数，方案二所耗时间比方案一减少约 700ms。图 4-4 是两种方案的时间对比，可见方案二在生成随机素数的效率上优于方案一，而且实验耗时较方案一波动更小。因此，在生成随机素数这个步骤上，改进的 Miller-Rabin 要优越些。

4.2.3 密钥的生成

公钥密码体制不同于对称密钥体制的最大区别，也是其最显著的一个特征就是存在一个公钥和私钥的握手阶段。在 RSA 公钥算法的实现过程中，关键的一环就是生成公钥和私钥。

4.2.3.1 最大公因子 GCD 的计算

Euclid 算法是计算两个数的最大公约数的经典算法，古希腊的数学家欧几里德最早在其著作《几何原本》中描述了这种算法，因为被命名为欧几里德算法。算法的本质是一种递归运算，每次运算的输出值成为下一次运算的输入值，根据这种性质，它又被称作为辗转相除法。设 p、q 为两个正整数，且 $p>q$，则算法的过程如下：

$$\begin{aligned} p &= b_0 q + r_0 && 0 \le r_0 < q \\ q &= b_1 r_0 + r_1 && 0 \le r_1 < r_0 \\ r_0 &= b_2 r_1 + r_2 && 0 \le r_2 < r_1 \\ &\cdots \\ r_{k-2} &= b_k r_{k-1} + r_k && 0 \le r_k < r_{k-1} \\ r_{k-1} &= b_{k+1} r_k \end{aligned} \qquad (4\text{-}8)$$

根据（4-8）可以看出，每一步运算所得到的余数一定小于上一步所得到的余数，即余数处在一个不断递减的过程，因此存在第 n 步使得 r_n 的值为 0，算法结束。程序伪代码可表示为如下：

```
function GCD(p, q)
while q≠0
    t ← q
    q ← pmodq
    p ← t
return p
```

4.2.3.2 模逆运算

以传统 RSA 算法为例，除了生成大素数 p、q，另一个至关重要的步骤是生成密钥 e、d 的步骤。一般在实际应用中，通常首先选取一个公开参数 e，

并使 e 满足 $GCD(e,\ \phi(n))=1$，私钥 d 满足公式 $ed \equiv 1 \bmod(\phi(n))$，即 d 可以看作是 e 的模 $\phi(n)$ 乘法逆元。假设 e 和 $\phi(n)$ 已知，则上式可以转化为求方程 $ed + x\phi(n) = 1$ 中的 x 和 d 的值。实际上，在 RSA 算法中只需求出该方程中的 d 即可，因此它是一个不定方程，可以使用扩展 Euclid 算法来求解。这类方程的通式如下：

已知正整数 $p, q(p < q)$，求整数 x 和 y，使得

$$xp + yq = GCD(p,q) \tag{4-9}$$

令 $GCD(p,q) = r$，根据上述的 Euclid 算法，有

（1）若 $r_0 = 0$，则 $GCD(p,q) = q$，$x_0 = 0$，$y_0 = 1$，得 $x = x_0$ 和 $y = y_0$；

（2）若 $r_0 \neq 0$，令 $x_0 = 1$，$y_0 = -b_0$，$x_1 = -b_1$，$y_1 = 1 + b_1 b_0$，则

$$px_0 + qy_0 = r_0 \tag{4-10}$$

$$px_1 + qy_1 = r_1 \tag{4-11}$$

将（4-10）和（4-11）代入 $r_{k-1} = b_{k+1} r_k + r_{k+1}$ 可得

$$r_2 = px_0 + qy_0 - b_2 (px_1 + qy_1) \tag{4-12}$$

又 $r_2 = px_2 + qy_2$，故

$$x_2 = x_0 - b_2 x_1 = 1 + b_1 b_2 \tag{4-13}$$

$$y_2 = y_0 - b_2 y_1 = -(b_0 + b_2 (1 + b_0 b_1)) \tag{4-14}$$

同理可以计算出

$$\begin{aligned} x_3 &= x_1 - b_3 x_2 \\ y_3 &= y_1 - b_3 y_2 \\ &\cdots \\ x_k &= x_{k-2} - b_k x_{k-1} \\ y_k &= y_{k-2} - b_k y_{k-1} \end{aligned} \tag{4-15}$$

当 $r_{k+1} = 0$ 时，$r_k = px_k + qy_k = GCD(p,q)$，得 $x = x_k$，$y = y_k$。

上述即为扩展 Euclid 算法的主要流程。若 $p \bmod q \neq 0$，在推导过程中，令 $x_0 = 1$，$y_0 = b_0$，$x_1 = b_1$，$y_1 = 1 + b_1 b_0$，可得

$$\begin{aligned}
&x_2 = x_0 + b_2 x_1 \\
&y_2 = y_0 + b_2 y_1 \\
&px_2(-1)^2 + qy_2(-1)^{2+1} = r_2 \\
&\cdots \\
&x_k = x_{k-2} + b_k x_{k-1} \\
&y_k = y_{k-2} + b_k y_{k-1} \\
&px_k(-1)^k + qy_k(-1)^{k+1} = r_k
\end{aligned} \tag{4-16}$$

当 $r_{k+1} = 0$ 时，$r_k = px_k(-1)^k + qy_k(-1)^{k+1} = GCD(p,q)$。

由于 x_k 的符号为 $(-1)^k$，y_k 的符号为 $(-1)^{k+1}$ 且 x_k 和 y_k 正负交替变换。如果只考虑 x_k 和 y_k 的绝对值，即解决 $px_k(-1)^k + qy_k(-1)^{k+1} = r_k$ 的符号问题，递推关系将更加简单，从而改进扩展 Euclid 算法。

当 k 为奇数时，q 的系数为正，而 k 为偶数时，有

$$(px_k(-1)^k - pq) + (pq + qy_k(-1)^{k+1}) = r_k \tag{4-17}$$

公式（4-17）中 q 的系数变为 $p - y_k$。

在扩展 Euclid 算法中，根据数学归纳法可得 $|x| < q$ 和 $|y| < p$，下面给出证明过程：

（1）若 $p \bmod q = 0$，则 $GCD(p,q) = q$，$x = 0$，$y = 1$，故上述结论成立；

（2）若 $p \bmod q \neq 0$，则 $r_k = px_k + qy_k = GCD(p,q)$，只需证明 $|x_k| < q$ 和 $|y_k| < p$ 即可。

①当 $k = 0$ 时，$p = b_0 q + r_0$，$|x_0| = 1 < q$，$|y_0| = b_0 < p$；

②当 $k = 1$ 时，$q = b_1 r_0 + r_1$，而 $p = b_0 q + r_0$，故 $p = (1 + b_1 b_0) r_0 + b_0 r_1$，$|x_1| = b_1 < q$，$|y_1| = 1 + b_0 b_1 < p$；

③当 $k = 2$ 时，$r_0 = b_2 r_1 + r_2$，设 $r_2 = qx' + r_0 y'$，由于 $r_2 = r_0 - b_2 r_1$ $= r_0 - b_2(q - b_1 r_0) = q(-b_2) + r_0(1 + b_1 b_2)$，故 $|x'| = b_2 < r_0$，$|y'| = 1 + b_1 b_2 < q$。又因为 $r_2 = qx' + r_0 y' = qx' + (p - b_0 q)y' = py' + q(x' - b_0 y')$，所以

$$\begin{cases} |x_2| = |y'| < q \\ |y_2| = |x' - b_0 y'| < |x'| + |b_0 y'| < r_0 + b_0 q = p \end{cases} \tag{4-18}$$

④假设 $r_k = qx' + r_0 y'$ 且 $|x'| < r_0$，$|y'| < q$ 成立。因为 $r_0 = p - b_0 q$，所以 $r_k = qx' + r_0 y' = qx' + (p - b_0 q)y' = py' + q(x' - b_0 y')$。又因为 $r_k = px_k + qy_k$，所以 $|x_k| = |y'|$，$|y_k| = |x' - b_0 y'|$。根据 $|x'| < r_0$ 和 $|y'| < q$ 可得

$$\begin{cases} |x_k| = |y'| < q \\ |y_k| = |x' - b_0 y'| \le |x'| + b_0 |y'| < r_0 + b_0 q = p \end{cases} \quad (4\text{-}19)$$

因此，$|x| < q$ 和 $|y| < p$ 成立，故 $p - y_k$ 为正。在 RSA 算法中，可根据下列步骤计算 e 的模 $\phi(n)$ 的乘法逆元 d。由于算法中不需要 x 的值，因此可以忽略 x 的计算：

（1） $p = \phi(n)$，$q = e$，$k = 1$，$y_0 = 0$，$y_1 = 1$，$r_0 = p$，$r_1 = q$，转向（2）；

（2）若 $r_1 \neq 1$，则 k++，$t = r_1$，$b = r_0 / t$，$r_1 = r_0 \bmod t$，转向（3）；否则转向（4）；

（3）若 $r_1 = 0$，则返回 false；否则 $r_0 = t$，$y = by_1$，$t = y_1$，$y_1 = y_0 + y$，$y_0 = t$，转向（2）；

（4）若 $k \bmod 2 = 0$，则 $y = p - y_1$；否则 $y = y_1$，结束并返回 true。

上述的改进方案消除了扩展 Euclid 算法中负数的运算，降低了 RSA 程序实现的复杂度，有利于 RSA 的应用。下面给出算法实现的伪代码：

```
bool exGCD(int p, int q, int &y)
if (q==0) return false;
    y0 ← 0; y1 ← 1;
    r0 ←p; r1 ←q;
    b ← 0
    while (r1!=1)
    {
      k++
      t=r1; b=r0/t ; r1=r0%t;
      if (r1==0) return false;
      r0=t; y=by1; y1=y0+y; y0=t;
    }
    if ((k%2)==0)y=p−y1;
    else y=y1;
return true;
```

实验测试：

输入：p=9，q=8　输出：返回 true，y=8；

输入：p=64，q=25　输出：返回 true，y=41；

输入：p=101，q=100　输出：返回 true，y=100；

输入：p=32，q=20　输出：返回 false，说明 p 与 q 不互素；

4.2.4 加解密运算

在 RSA 公钥算法中，加、解密过程都要进行 $C=M^E \bmod n$ 的运算，文中使用滑动窗口取幂算法与第三章中提出的改进 SMM 算法相结合。滑动窗口取幂算法的思想是通过对二进制数序列进行预处理，将序列中的非零数降到最低来优化算法的运算速度[67]。首先将指数 E 转换成二进制数 $E=(e_{t-1}e_{t-2}\cdots e_1e_0)_2$，取滑动窗口的固定长度为 k，从 E 的最高位 1 开始，向右取 k 个数据位，记作 W_{h-1}，并将 W_{h-1} 后的连续出现的零（可能不存在）记作 O_{h-1}；接着从 O_{h-1} 后的 1 开始再取 k 个数据位，记作 W_{h-2}，并将 W_{h-2} 后的连续出现的零记作 O_{h-2}；如此类推，直至最后一次以 1 开始取 k 个数据位（可能不足 k 位），记作 W_0；若 W_0 后仍有若干个零，则这若干个零记作 O_0。此时，指数 E 可表示为 $E=W_{h-1}\|O_{h-1}\|W_{h-2}\|O_{h-2}\|\cdots\|W_0\|O_0$。滑动窗口取幂算法实现的核心代码如下：

```
M1 ← M, M2 ← M
While 1 ≤ i ≤ 2^(k-1)−1
    M_{2i+1} ← M_{2i+1} M_2 mod n
S ← 1, i ← t
While i ≥ 0
    if (e_i=0)
    then {S ← S^2, i ← i-1}
    else {e_i e_{i+1} e_L, i-L+1 ≤ k, e_L=1}
    S ← S^{2i+L+1} M_{(e_i e_{i-1} ··· e_L)_2}, i ← L−1
return S
```

4.3 改进的 RSA 数字签名过程

根据 4.1.2 节中对传统签名方案的分析，文中使用改进的 RSA 算法签名方案实现数字签名的过程。

4.3.1 算法签名过程

在改进的 RSA 数字签名方案中，通信双方各拥有 2 个密钥。秘密密钥 $SK-A$、$SK-B$ 及公开密钥 $PK-A$、$PK-B$，分别对应的是加密算法 E_{PK-A}、E_{PK-B} 和解密算法 D_{SK-A}、D_{SK-B}。数字签名的实现过程如图 4-5 所示：

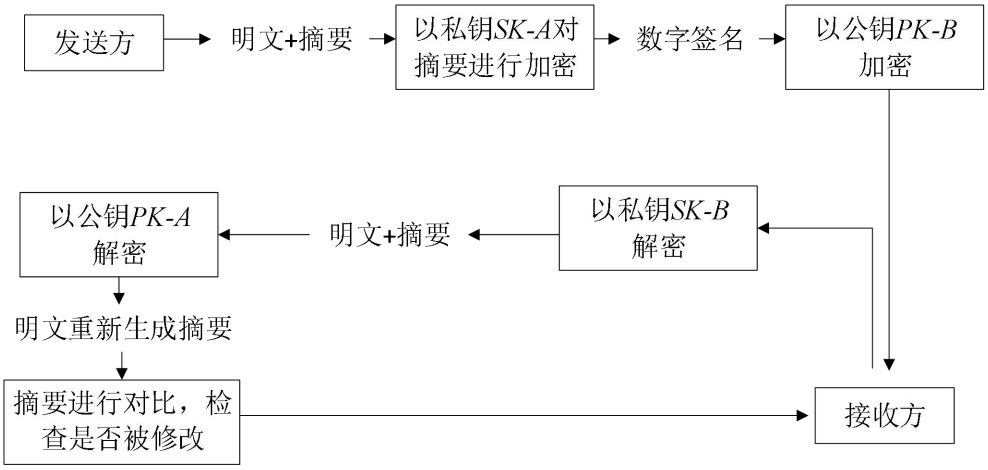

图 4-5 改进的 RSA 数字签名方案

Fig 4-5 Improved RSA signature scheme

若用户 A 拟传送待签名的消息 m 给用户 B，则 A 可以用自己的解密算法 D_{SK-A} 对摘要 h 进行加密，从而得到 $D_{SK-A}(h)$，再用 B 的加密算法 E_{PK-B} 对 $D_{SK-A}(h)$ 进行加密：

$$C = E_{PK-B}(D_{SK-A}(h)) \tag{4-20}$$

当用户 B 获得密文 C 后，首先用自己的解密算法 D_{SK-B} 对 C 进行解密：

$$D_{SK-B}(C) = D_{SK-B}(E_{PK-B}(D_{SK-A}(h))) = D_{SK-A}(h) \tag{4-21}$$

接着再用 A 的加密算法 E_{PK-A} 对 $D_{SK-A}(h)$ 进行解密：

$$E_{PK-A}(D_{SK-A}(h)) = h' \tag{4-22}$$

若 $h=h'$，则可以确定消息来自于 A，否则拒绝对方的签名。若 A 否认给 B 发送过消息 m，则用户 B 只需向公证方提供 A 的签名 $D_{SK-A}(h)$ 和公钥 e，公证方通过算法很轻易就能够证实签名的确属于 A；同样地，若用户 B 伪造了消息 m'，因为 B 不知道 A 的私钥，于是便无法提供正确的签名，这样消息通

信双方都有效防止了一方抵赖情况的发生，从而达到可靠的数字签名目的。

使用改进的 RSA 数字签名方案进行签名，输入的是待签名的消息 M 和 RSA 算法的私钥 (d,n)，输出的是 $(M\| r,S)$，算法的描述过程如下：

（1）需要签名的消息 M；

（2）生成伪随机数 r；

（3）对消息明文进行分组 $M\| r$；

（4）计算消息摘要的值 $h=H(M\| r)$；

（5）生成签名 $S\equiv h^d \bmod n$；

（6）输出 $(M\| r,S)$。

4.3.2 数字签名的创建

（1）需要进行签名的消息 M=let it be；

（2）生成本次签名的伪随机数

r=762176125042327111720717502505887831；

（3）对消息数据进行分组

$m\| r$ =let it be 762176125042327111720717502505887831；

（4）计算消息摘要的值

$h=H(m\| r)$ =1296A4AE899AD56EDD5C9AAADA94B130；

（5）生成签名 $S\equiv h^d \bmod n$，如表 4-3 所示：

表 4-3 生成签名

Table 4-3 Signature generation

835545503697915174671882775826921215615759627646632512393629264741652125551675014512680712789947426623756581158302203061732282556482248928799795113226205526677916309638766891565863074120371211208213318278624461445663504011209561819794084773597887841961701301281590899602911789810491017750473883612759913704566161529667921005536583521322452818321756039532018615082562039760504427231947076250870863915480863902420448416321316248699427713499570136659920482970139059132464346162291046622538616424659205124039170240033251098804276770119582641952881139186988 45

（6）输出 $(m\|r,S)$，如表 4-4 所示：

表 4-4 输出消息摘要和签名

Table 4-4 Output message digest and signature

let it be 762176125042327111720717502505887831,8355455036979151 746718827758269212156157596276466325123936292647416521255516750145126807127899474266237565811583022030617322825564822489287997951132262055266779163096387668915658630741203712112082133182786244614456635040112095618197940847735978878419617013012815908996029117898104910177504738836127599137045661615296679210055365835213224528183217560395320186150825620397605044272319470762508708639154808639024204484163213162486994277134995701366599204829701390591324643461622910466225386164246592051240391702400332510988042767701195826419528811391869884

4.3.3 数字签名的验证

验证过程的输入是待签名的消息 M，RSA 算法的公钥 (e,n) 和签名 S，输出的是接受签名或拒绝签名，算法首先计算消息摘要的值 $h=H(M\|r)$，接着计算 $h'=S^e \bmod n$，最后将 h 和 h' 进行比较，若 $h=h'$，则接受签名；否则，拒绝签名。

对上面创建的签名进行验证：

（1）还原签名的消息 h'=1296A4AE899AD56EDD5C9AAADA94B130；

（2）计算消息摘要的值：

$h=H(m\|r)$= 1296A4AE899AD56EDD5C9AAADA94B130；

（3）比较 h 和 h'，因为 $h=h'$，所以接受签名；

（4）签名有效。

4.4 本章小结

本章分析了传统 RSA 的签名方案存在的不足之处。在签名方案实现的关键步骤中，使用 Hash 函数和改进的 RSA 签名方案解决传统方案的不足，分析了随机大素数和密钥生成中使用的算法，对其中的 Miller-Rabin 算法进行

了改进，并完成了 RSA 数字签名的实现过程。下一章中，将对本文中算法的改进方案进行组合，并进行实验对比，验证其可行性与优越性。

5 组合算法的实现与测试

基于 RSA 公钥算法的理论分析，本章结合算法的安全性与各个关键步骤算法的优点，设计了一种组合的 RSA 公钥算法，分析了算法中大数模幂乘运算的核心过程，并将组合算法与传统算法进行了实验对比。

5.1 组合算法的选定依据

前面第三章所分析的算法中，四素数 RSA 算法是通过降低相同位数的模数下所需随机大素数的大小，从而降低了大素数生成的时间，但是算法的安全性也因此受到影响。参数替换 RSA 算法由于在两个密钥中均消除了模数 n 的分布，提升了传统 RSA 算法的安全性，多重密钥 RSA 算法虽然在通过增加了密钥攻击的时间一定程度上提升了算法的安全性，但是参数替换的方案在算法的构造上较之更具有优势。

基于同余对称特性的 SMM 算法是通过降低了 $C = M^E \bmod n$ 运算中基数 M 的大小来减少了算法的运算量，改进的 SMM 算法在此基础上进一步降低了指数 E 的汉明重量，从不同方面都在一定程度上提升了算法的效率；

在算法的设计问题，即组合算法的选择上，应充分考虑其准确性、高效性、安全性以及实用性原则。因此，本文针对四素数、参数替换方案、SMM 算法各自的优点，再与第四章中的滑动窗口取幂算法、改进的 Miller-Rabin 算法和扩展欧几里得算法相结合，重新组合出一种快速而高效的加密算法。

5.2 组合算法的实现

大数的模幂乘运算的实现过程是组合算法的核心。首先，处理加密指数，降低其二进制数序列中的汉明重量，接着采用滑动窗口取幂算法对其进行预处理，再在乘模运算中使用改进的 SMM 算法，将中间运算数值和模数进行大小比较，如果满足算法的替换条件，则进行替换，这样可以减小运算中基数的大小。下面具体分析组合算法实现的过程：

将指数 E 转换为二进制形式 $E = (e_t e_{t-1} \ldots e_1 e_0)_2$，然后利用 NAF 算法将其转化成二进制冗余数的形式 $E = (s_t s_{t-1} \ldots s_1 s_0)_2$，其中 $s_i \in \{-1, 0, 1\}$。具体实现过程如算法 5-1 所示：

算法 5-1 二进制冗余数转换
Input: $E=(e_te_{t-1}\ldots e_1e_0)_2$ Output: $E=(s_ts_{t-1}\ldots s_1s_0)_2$， $s_i\in\{-1,0,1\}$ $c_0\leftarrow 0$; while $0\le j\le t$ $c_{j+1}\leftarrow[(e_j+e_{j+1}+c_j)/2], e_i=0, i>t$ $s_j\leftarrow e_j+c_j-2c_{j+1}$ return $(s_ts_{t-1}\cdots s_1s_0)_2$

经过转换，降低了二进制数序列的汉明重量。其次，利用滑动窗口取幂算法进行运算，取窗口的固定长度为 k，则 $M^E \bmod x$ 可以表示为：

$$(((((((M^{W_{h-1}})^{2^k})^{2^{|O_{h-1}|}}\times M^{W_{h-2}})^{2^k})^{2^{|O_{h-2}|}}\times\cdots\times M^{W_1})^{2^k})^{2^{|O_1|}}\times M^{W_0})_x^{\,2^{|O_0|}} \qquad (5\text{-}1)$$

公式（5-1）中，取 $h=x/2$ 作为改进的 SMM 算法中对基数进行选择性替换的值。在实现过程中，可以分为预处理和模幂运算两步进行：

（1）预处理。首先产生三个线性表 R、S 和 T，其中 R 用于存放指数 E 的二进制冗余数中的非零码，S 用于存放 R 中的元素 $i-1$ 比 i 少运算的幂次数目，T 用于存放 $(M,M^3,M^5,\cdots,M^{2^k-1})\bmod x$ 的值，其中 R、S 和 T 的实现可以利用算法 5-2 和算法 5-3；利用欧几里得算法实现线性表 U，用于保存 $(M^{-1},M^{-3},M^{-5},\cdots,M^{-(2^k-1)})\bmod x$ 的值。

算法 5-2 产生线性表 T
$A=M\times M\%x$; $T[0]=M$; $i=1$; while $(i\le 2\ll(k-1)-1)$ $\{T[i]=T[i]\times A\%x$; $i++;\}$

算法 5-3 生成线性表 R 和 S
$i=0$; $d=0$;

```
m = −1;
while (i < x)
    {if (e[i] == 0)
       d++;
    else
       {m++;
       if (i + k − 1 < x)
          i = fun2to10(b[i + k − 1], b[i]); //将二进制 b[i + k − 1]⋯b[i]化为十进制
       else
          j = fun2to10(b[x − 1], b[i]); //将二进制 b[x − 1]⋯b[i] 化为十进制
       R[m] = j;
       S[m] = d;
       d = k;
       i = i + k − 1;}
i++;}
```

（2）模幂运算。算法 5-4 给出了利用线性表 R、S、T 和 U 计算 $C = M^E \bmod x$ 的过程。

算法 5-4 模幂运算

```
i = (R[m] − 1) / 2;
c = T[i];
s = U[i];
i = m;
d = 0;
while (1)
    {j = 1;
    while ( j <= S[i])
        {c = c × c%x;
        j = j + 1;}
        j = 1;
    while ( j <= U[i])
```

```
            {s = s × s%x;
            j = j + 1;}
            i − −;
        if (i < c)
            break;
        d = (R[i] − 1) / 2;
        c = c × T[d]%x;}
if (c >= x)
        c = c − x;
return c;
```

5.3 运行与测试

5.3.1 运行环境与平台

本文实验采用的硬件环境：i5-2450M CPU，6GB 内存，450G 硬盘；

操作系统：Windows 7；

开发工具：Visual Studio 2017。

RSA 加密系统实现过程中主要的类和函数如下：

```
class  CBigInt
{
public:
    unsigned  m_nLength; //大数在 0x100000000 进制下的长度
    unsigned long  m_ulValue[BI_MAXLEN]; //大数在 0x100000000 进制下的每个元素存储的数组
    CBigInt;
    ~CBigInt;
    void  Mov(unsigned_int64 X);
    void  Mov(CBigInt& X); //赋值运算
    CBigInt  Add(CBigInt& X);
    CBigInt  Sub(CBigInt& X);
    CBigInt  Mul(CBigInt& X);
    CBigInt  Div(CBigInt& X);
```

CBigInt *Mod*(CBigInt& X); //求模运算

CBigInt *Add*(unsigned long X);

CBigInt *Sub*(unsigned long X);

CBigInt *Mul*(unsigned long X);

CBigInt *Div*(unsigned long X);

unsigned long *Mod*(unsigned long X);

int *Cmp*(CBigInt& X); //比较运算

void *Get*(CString& *str*, unsigned int *system=HEX*); //字符串以十六进制的格式输入至大数中

void *Put*(CString& *str*, unsigned int *system=HEX*); //大数以十六进制的格式输出至字符串中

void *GetPrime*(int *bits*); //生成指定长度的随机大数

int *Rab*( ); //Miller-Rabin 素性测试法

CBigInt *Euc*(CBigInt& X); //欧几里得算法

CBigInt *Rsa*(CBigInt& X, CBigInt& Y); //传统 RSA 算法

CBigInt *Rsasmm*(CBigInt& X, CBigInt& Y); //SMM-RSA 算法

CBigInt *Rsamul*(CBigInt& X, CBigInt& Y); //多重密钥 RSA 算法

CBigInt *Rsacmb*(CBigInt& X, CBigInt& Y, CBigInt& A, CBigInt& B); //组合 RSA 算法

5.3.2 运行过程

（1）主体运行窗口

RSA 加密系统运行窗口如 5-1 图所示，可供选择的密钥位数为 128bit、256bit、512bit、1024bit 和 2048bit；对于随机大素数的生成，可以选择两种方式，分别是双素数因子和四素数因子。系统实现了传统 RSA 算法、改进的 SMM-RSA 算法、多重密钥 RSA 算法和组合 RSA 算法四种加解密的方式，为了克服多重密钥 RSA 算法在密钥生成效率上的不足，在实现时将其与中国剩余定理相结合。加解密时间窗口显示不同算法完成操作的耗时情况。下面介绍程序窗口中各个按钮的功能：

①随机生成双素数：生成大素数 p 和 q，选取出 e，计算出 d；

②随机生成四素数：生成大素数 p、q、r 和 s，选取出 e，计算出 d、x；

③清除：清除本次操作所产生的参数，预备进行下一次操作；

④RSA 测试：传统 RSA 算法对明文进行加密和解密操作；

⑤SMM-RSA 测试：改进的 SMM-RSA 算法对明文进行加密和解密操作；
⑥多重密钥 RSA 测试：多重密钥 RSA 对明文进行加密和解密操作；
⑦组合 RSA 测试：组合 RSA 算法对明文进行加密和解密操作。

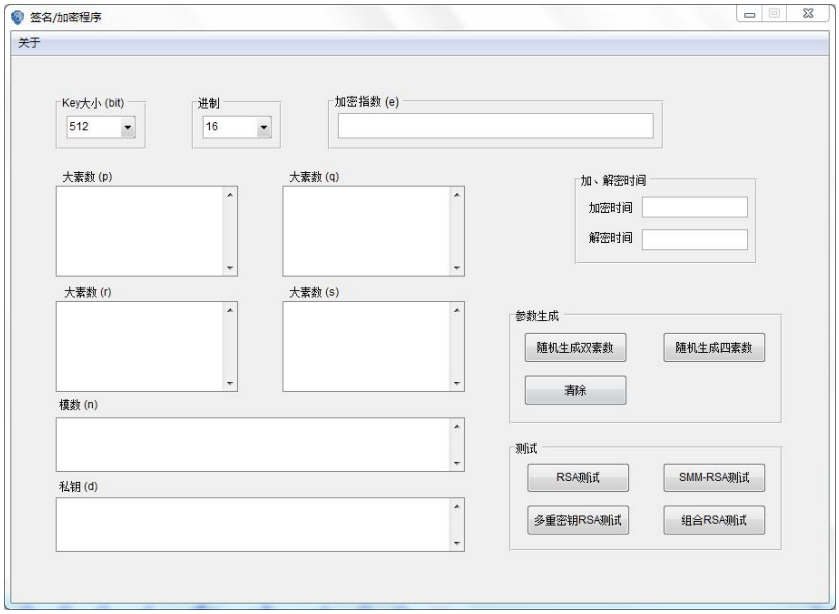

图 5-1 主体窗口

Fig 5-1 The main window

（2）密钥生成

以组合 RSA 算法生成 512bit 素数为例，点击“随机生成四素数”之后，系统计算参数 x，选取公钥 e 指数并计算得出私钥 d，密钥生成窗口如图 5-2 所示。

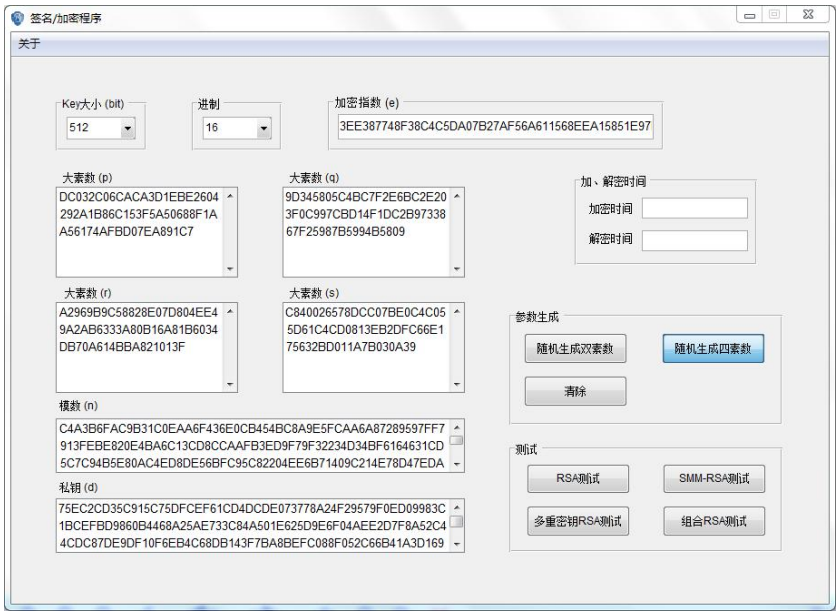

图 5-2 密钥生成窗口

Fig 5-2 The window for key generation

（3）加密和解密运算

点击“组合 RSA 测试”按钮，弹出明文加解密界面，在明文窗口中输入需要加密的内容，点击“加密（E）”按钮，即可获得密文，结果如图 5-3 所示。

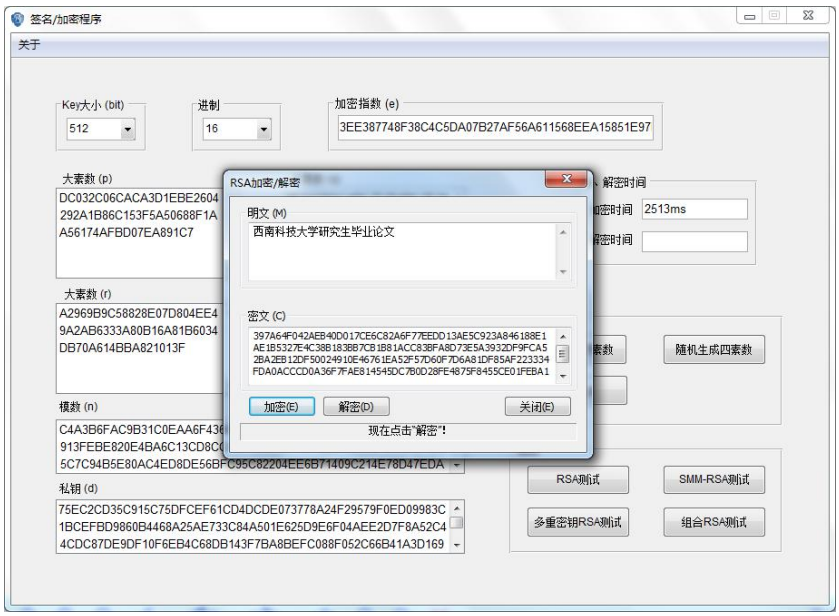

图 5-3 加密窗口

Fig 5-3 Encryption window

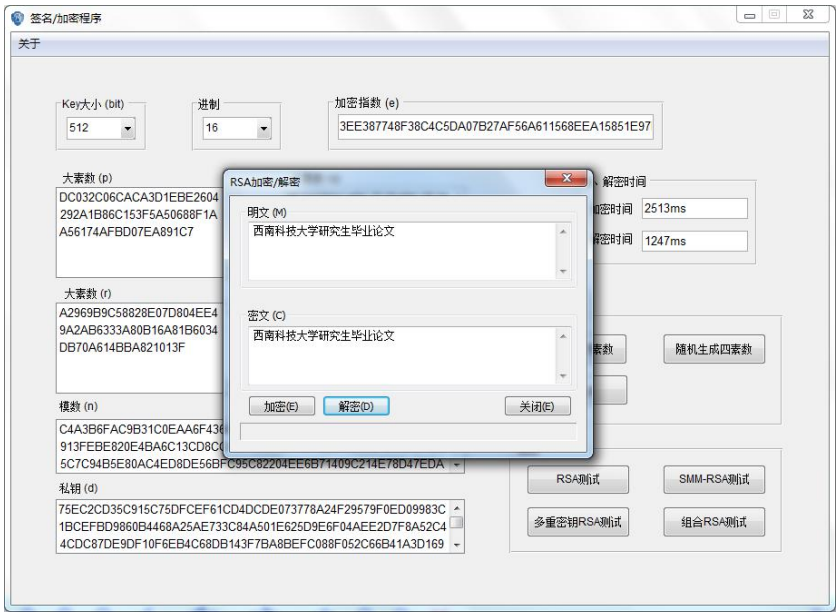

图 5-4 解密窗口

Fig 5-4 Decryption window

从图 5-3 中可以看到，加密后的明文无法被识别。继续点击“解密（D）”按钮，得到解密后的明文，窗口如图 5-4 所示，恢复后的明文与原始输入的明文相同。实验中选择了不同的大素数和 RSA 算法进行多次测试，程序可以正常运行，达到了预期的设计的效果。

5.3.3 测试结果与分析

为了对传统 RSA 算法、改进的 SMM-RSA 算法、多重密钥 RSA 算法和组合 RSA 算法的签名效率进行充分对比，文中用四种算法在密钥长度为 128bit、256bit、512bit 和 1024bit 的情况下，分别对 128bit、256bit、512bit 和 1024bit 的明文进行 5 次加解密操作，并记录平均所需的时间。多重密钥算法中 k 的取值为 10。实验测试的结果如表 5-1 所示（单位：ms）：

表 5-1 算法签名耗时对比

Table 5-1 Signature time consumption table of algorithms

明文大小/bit	密钥长度/bit	传统 RSA	SMM-RSA	多重密钥 RSA	组合 RSA
128	128	624	357	395	276
	256	1386	848	856	632
	512	2892	1643	1711	1235
	1024	5943	3290	3523	2678
256	256	1491	878	918	665
	512	2973	1755	1812	1310
	1024	6138	3394	3775	2879
512	512	3115	1841	1921	1472
	1024	6357	3552	3968	3004
1024	1024	6528	3697	4219	3177

分析表 5-1 中的每组数据可以看出，当明文长度相同时，随着密钥长度的增加，同一种算法的签名时间的增幅较大；当密钥长度相同，明文长度增加时，每种算法签名所耗时间的增幅较小。这是因为在模幂运算 $Y = X^E \bmod n$ 中，密钥指数对运算效率的影响要远大于基数的影响。

将四种算法的签名时间进行比较，相比传统 RSA 算法，改进的 SMM-RSA 算法的签名效率提升了约 40.6%，与理论值 42%相接近。多重密钥 RSA 算法和组合 RSA 算法的签名效率提升分别约为 39.1%和 54.3%。从分析的结果可知，组合算法签名效率的优化相对明显，获得了较为理想的效果。因此，可以将组合算法运用到 RSA 数字签名系统中，这样不仅保证了签名的安全性，还可以有效提升数字签名的效率。

5.4 本章小结

本章在算法效率上运用改进的 SMM 算法和 Miller-Rabin 算法，结合基于安全性的四素数参数替换的改进方案，实现了组合算法的 RSA 密码系统。为了对组合算法的效率进行验证，文中将其与传统 RSA 算法、改进的 SMM-RSA 算法和多重密钥 RSA 算法进行了多次的签名实验，并对比了每种

算法的签名时间。实验结果表明，改进后的组合算法较其他三种算法具有更好的签名效率。

结　论

由于针对 RSA 公钥算法攻击技术研究的不断深入，尤其是因子分解技术的不断提高，不得不在实际应用中选择更长的密钥位数，致使密钥生成和算法运算的效率大幅地降低，对 RSA 公钥算法的继续应用与推广产生了严重的影响。为了使 RSA 公钥算法在数字签名中得到更好的应用，设计更加安全、高效的数字签名方案，本文围绕算法的安全性和签名效率两个方面，对传统 RSA 公钥算法和数字签名进行了深入的研究，最终将问题聚焦到对 RSA 算法中的关键步骤算法和数字签名的实现方式上，并取得了一定的研究成果，主要成果总结如下：

（1）通过阅读大量国内外的相关文献，阐述了信息安全技术，尤其是 RSA 公钥算法和数字签名技术的研究现状，讨论了对 RSA 算法进行优化的必要性。介绍了公钥密码体制在实现数字签名上的优势，分析了模幂运算、同余式、乘法逆元等相关数学理论基础，对 RSA 加解密变换的实现过程进行了深入的研究，结合当前面临的安全问题和加解密时间长的发展瓶颈，对提高算法安全性与效率的原则进行了总结。

（2）针对模数 n 的因子分解攻击、选择密文攻击等攻击方式，提出了参数替换的方案，证明了四素数和参数替换 RSA 改进方案在理论上的正确性。对于公共模数攻击，提出在加解密变换过程中对不同的明文分组使用不同密钥的改进方案。依据算法的安全性评价指标，分析与总结了改进方案在提升算法安全性上的优越性。

（3）分析了 RSA 公钥算法中的几种关键算法。通过对 SMM 算法的研究，发现被模数二进制数序列中非零数的个数是影响模幂运算中迭代步数的关键因素，给出了使用二进制冗余数对二进制数序列中出现的连续数字“1”进行处理的方案，降低了二进制数序列的汉明重量，进一步提升了算法的效率。在改进的 RSA 签名方案的步骤中，对生成随机大素数的 Miller-Rabin 算法进行了改进，提升了素数的生成效率。

（4）最后，利用上述的改进方案重新组合出一种新算法，用以解决 RSA 公钥算法在数字签名应用中的不足。设计与实现了 RSA 密码系统，并对比了传统算法与组合算法所耗的签名时间，验证了组合算法的有效性。

由于课题时间与条件有限的关系，本文的研究工作存在有待完善的地方，仍需要进一步的研究：

（1）在大整数的模幂运算中，模数与被模数都会对运算的效率产生影响，本文在研究中仅考虑了对被模数的优化。由于模数与随机大素数之间存在一定的关系，因此对于模数的处理也是降低算法的运算量、优化算法中的重要内容，本课题在如何优化模数的方面研究不多。

（2）对于实现数字签名技术，这方面的算法还有很多，需要与本课题基于 RSA 公钥算法实现数字签名的方案进行对比研究，为满足数字签名技术的发展要求，提供持续的动力。另外，消息摘要也是影响签名安全性和效率的因素，因此对于复杂明文如何生成更安全、高效的消息摘要的研究，也是数字签名技术的一个研究方向。

致　　谢

三年研究生的学习时间，如白驹过隙，转眼即要画上句号。当此硕士学位论文完成之际，谨向三年来在学业上指导我的导师，和所有无私关心、帮助我的同学及亲友们表示我的最崇高的敬意与最诚挚的谢意！

首先我要感谢我的导师周金治副教授，周老师学高为师，身正为范，他的严谨认真，兢兢业业的治学态度，使我受到了深刻的熏陶，为我的学习科研指明了方向；周老师对我所研究的课题进行了耐心的指导，为我在实验中遇到的困难也提出了宝贵的意见。另外，周老师虽然工作繁忙，但也不忘经常关心和了解我们学习和生活的情况，并给予我们以支持和扶助。当以后的工作和学习中，我会不使自己忘记周老师治学严谨，待人宽和的优秀的品质，再次向周金治老师致以衷心的感谢。

其次，我要感谢师兄郑希和师姐康春香在我的研一期间，对我分享课程学习和课题研究上的宝贵经验；感谢同门的吴兴铨、蔡宜在课题的开展和理论研究中所给予我的无私的帮助；感谢师兄凡旭国在我进行实验平台的搭建和数据分析，遇到问题的时候，对我的相关的指导；感谢刘多多、万栋同学在论文的写作中给我提出的建议，感谢师弟涂道鑫和师妹郑琳文帮助我查阅论文中的不足与错误，我的课题与论文能最终顺利地完成，与你们所有人的助力与支持是分不开的。

还要感谢三年来，和我同寝室的室友黄迪和朱家磊在生活中对我的关心，与你们的同行，为我的研究生生涯添加了许多的趣处和欢乐；以后的日子我们即将分别，再不能像研究生时期这样的交流与陪伴，不过这一段隽永的同窗之情，是我无论任何的时候回忆起来，都值得回味的，在此我对你们致以最美好的祝愿。

另外，感谢我最亲爱的父母，感谢你们对我的培养，供我读书。虽然我很少与你们分享学业上的事情，把在学校的生活向你们说，但是你们对我的支持与鼓励，一直是我在学业上前进的动力，感谢你们的辛苦与付出。

最后，衷心感谢百忙中抽出时间来审阅本论文，参与答辩的各位专家和老师。

参考文献

[1] 国家计算机网络应急技术处理协调中心. 2016 中国互联网网络安全报告[M]. 北京: 人民邮电出版社, 2017:23-26.

[2] 刘德良. 我国网络信息安全问题的法律规制[J]. 信息安全与通信保密, 2011(6):31-35

[3] 刘海玲, 裴连群. 计算机网络信息安全问题及防护策略[J]. 自动化与仪器仪表, 2017(9):192-193.

[4] 覃力更, 何增镇, 王彬. 基于RSA加密算法的车道收费软件授权认证管理系统[J]. 西部交通科技, 2015(2):58-61.

[5] Mohanaprakash T A, Vinod A I, Raja S, et al. A Study of Securing Cloud Data Using Encryption Algorithms[J]. 2018, 03(01):730-734.

[6] 贾源泉, 杨磊. 基于公钥密码在网络信息安全中的应用研究[J]. 通讯世界, 2017(14):93-94.

[7] Babu S S, Balasubadra K. Revamping data access privacy preservation method against inside attacks in wireless sensor networks[J]. Cluster Computing, 2018:1-11.

[8] Denning D E. Digital signatures with RSA and other public-key cryptosystems[J]. Communications of the ACM, 1984, 27(4):388-392.

[9] Douglas R. Stinson. 密码学原理与实践[M].3 版. 北京: 电子工业出版社, 2016:222-234.

[10] Brown D R L. Breaking RSA may be as difficult as factoring[J]. Journal of Cryptology, 2016, 29(1):220-241.

[11] 李超, 王世雄, 屈龙江,等. 基于格的 RSA 密码分析[J]. 河南师范大学学报(自然版), 2017(3):1-13.

[12] Wang Y, Zhang H, Wang H. Quantum polynomial-time fixed-point attack for RSA[J]. China Communications, 2018, 15(2):25-32.

[13] Shor P W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer[J]. SIAM review, 1999, 41(2):303-332.

[14] 曹祖平. RSA 型公钥密码体制的解密指数攻击[D]. 西南大学, 2011.

[15] 勾云, 曾光, 王广赛,等. 对两类 RSA 变体的小解密指数攻击[J]. 四川大学学报(自然科学版), 2014, 51(4):689-695.

[16] Bunder M, Tonien J. New attack on the RSA cryptosystem based on continued fractions[J]. Malaysian Journal of Mathematical Sciences, 2017, 11:45-57.

[17] S Arora, Pooja. Enhancing Cryptographic Security using Novel Approach based on Enhanced-RSA and Elamal: Analysis and Comparison[J]. International Journal of Computer Applications, 2015, 112(13):35-39.

[18] Al-Hamami A H, Aldariseh I A. Enhanced method for RSA cryptosystem algorithm[C]//Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on. IEEE, 2012:402-408.

[19] Mustafi K, Sheikh N, Hazra T K, et al. A novel approach to enhance the security dimension of RSA algorithm using bijective function[C]// Information Technology, Electronics and Mobile Communication Conference. IEEE, 2016:1-6.

[20] Hassan A K S, Shalash A F, Saudy N F. Modifications on rsa cryptosystem using genetic optimization[J]. International Journal of Research and Reviews in Applied Sciences, 2014, 19(2):150.

[21] 胡军. RSA 加密算法的研究与实现[D]. 安徽工业大学, 2011.

[22] Tian W, Li Z. Study of RSA Encryption Algorithm and Its Improved Algorithm[J]. Shanxi Electronic Technology, 2013:90-91

[23] Rizzo O G. On the complexity of the 2k-ary and of the sliding window algorithms for fast exponentiation[J]. RIVISTA DI MATEMATICA DELLA UNIVERSITÀ DI PARMA, 2004, 7(3):289-299.

[24] Xiao Z J, Jiang Z T, Wang Y B, et al. Improved RSA Algorithm and Application in Digital Signature[C]//Applied Mechanics and Materials. Trans Tech Publications, 2015, 713:1741-1745.

[25] 谢琪. 一种新的 RSA 的快速算法[J]. 计算机工程, 2003, 29(2):51-52.

[26] 石小平, 姜浩. 模重复平方算法的 rho 改进算法[J]. 计算机应用与软件, 2011, 28(12):48-50.

[27] 党佳莉. 中国剩余定理在数字签名中的应用研究[D]. 青海师范大学, 2015.

[28] Mahanta H J, Khan A K. Implementation of RSA and CRT-RSA with MIST to Resist Power Analysis Attacks[J]. Journal of VLSI Design Tools & Technology, 2017, 7(3):1-12.

[29] Xu S, Lu X, Zhang K, et al. Similar operation template attack on RSA-CRT as a case study[J]. Science China Information Sciences, 2018, 61(3):032111.

[30] RIVEST R, SHAMIR A, ALDEMAN L. A method for obtaining digital signatures and public-key cryptosystems [J]. Communications of the ACM, 1978, 21(2):120-126.

[31] ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms[J]. IEEE Trans.inf.theory, 1985, 31(4):469-472.

[32] 陈勤. 数字签名算法的比较及其应用[J]. 计算机应用研究, 1999(10):10-11.

[33] Johnson D, Menezes A, Vanstone S. The elliptic curve digital signature algorithm (ECDSA)[J]. International journal of information security, 2001, 1(1):36-63.

[34] 覃征, 闫焱, 王立. 特殊签名及其在电子商务中的应用[J]. 计算机应用研究, 2003, 20(6):1-3.

[35] Bhala A S, Kshirsagar V P, Nagori M B, et al. Performance comparison of elliptical curve and rsa digital signature on arm7[C]//2011 International Conference on Information and Network Technology, IPCSIT. 2011, 4:58-62.

[36] 雷咏, 杨世平. 基于 PKI 的签密体制[J]. 通信技术, 2013(1):43-46.

[37] 陈亮. 基于格的数字签名方案及其应用[D]. 华中科技大学, 2013.

[38] 张瑞丽. 数字签名的相关研究及应用[D].陕西师范大学,2015.

[39] Shi Y, Han J, Fan H, et al. Protecting Encrypted Signature Functions Against Intrusions on Computing Devices by Obfuscation[J]. IEEE Access, 2016, 4:6401-6415.

[40] Barenghi A, Bertoni G M, Breveglieri L, et al. Fault attack to the elliptic curve digital signature algorithm with multiple bit faults[C]// International Conference on Security of Information and Networks, Sin 2011, Sydney, Nsw, Australia, November. DBLP, 2011:63-72.

[41] 张文芳, 熊丹, 王小敏,等. 基于RSA公钥密码体制的可选择可转换关联环签名[J]. 计算机学报, 2017, 40(5):1168-1180.

[42] Kamal Kr. Gola, Gupta B, Iqbal Z, et al. Modified RSA Digital Signature Scheme for Data Confidentiality[J]. International Journal of Computer Applications, 2014, 106(13):13-16.

[43] Hemant Kumar, Ajit Singh, An Efficient Implementation of Digital Signature Algorithm with SRNN Public Key Cryptography[J], IJRREST, 2012, 1(1):54-57.

[44] 王欣萍. 基于 DES 和 ECC 混合型数字签名算法研究[D]. 哈尔滨工程大学, 2009.

[45] 廖剑. 基于双线性对的数字签名的安全性研究[D]. 上海交通大学, 2006.

[46] 唐四薪, 李浪, 谢海波. 密码学及安全应用[M]. 清华大学出版社, 2016:1-20.

[47] 石峰, 戴冠中, 刘航,等. 基于门限方案的智能卡密钥管理系统的设计与实现[J]. 计算机应用, 2006, 26(9):2156-2159.

[48] DouglasR.Stinson 道格拉斯 R.斯廷森. 密码学原理与实践[M]. 电子工业出版社, 2016:2-28.

[49] 黄志荣, 范磊, 陈恭亮. 密钥管理技术研究[J]. 计算机应用与软件, 2005, 22(11):112-114.

[50] Radu T, Mircea S. Evaluation of DES, 3 DES and AES on Windows and Unix platforms[C]// International Joint Conference on Computational Cybernetics and Technical Informatics. IEEE, 2010:119-123.

[51] 周升力. RSA 密码算法的研究与快速实现[D]. 南昌大学, 2008.

[52] Diffie W, Hellman M. New directions in cryptography[J]. IEEE transactions on Information Theory, 1976, 22(6):644-654.

[53] 屈晓. 基于公钥密码体制的模幂算法执行效率研究[D]. 天津大学, 2013.

[54] 郁昱, 李祥学. 基于单向函数的伪随机产生器与通用单向哈希函数[J]. 西安邮电大学学报, 2016, 21(02):1-11.

[55] 刘俊峰. 混合密码模拟系统的研究和实现[D]. 电子科技大学, 2006.

[56] Bai S, Bouvier C, Kruppa A, et al. Better polynomials for GNFS[J]. Mathematics of Computation, 2016, 85(298):861-873.

[57] 冯登国, 赵险峰. 信息安全技术概论[M]. 北京: 电子工业出版社, 2012:30-32

[58] Chu H, Ge W. Digital multisignature schemes based on Schnorr algorithm[J]. Computer Engineering, 2005.

[59] Pajčin B R, Ivaniš P N. Analysis of Software Realized DSA Algorithm for Digital Signature[J]. Electronics, 2011, 15(2):39-61.

[60] 吴克力. 数字签名理论与算法研究[D]. 南京理工大学, 2005.

[61] 殷丽华, 方滨兴. 入侵容忍系统安全属性分析[J]. 计算机学报, 2006, 29(8):1505-1512.

[62] 肖振久, 胡驰, 陈虹. 改进的 RSA 算法在数字签名中的应用[J]. 计算机工程与应用, 2014, 50(17):106-109.

[63] Wu T, Li S G, Liu L T. Fast RSA decryption through high-radix scalable Montgomery modular multipliers[J]. Science China Information Sciences, 2015, 58(6):1-16.

[64] 龚征. 轻量级 Hash 函数研究[J]. 密码学报, 2016, 3(1):1-11.

[65] Goyal G, Singh K, Ramkumar K R. A detailed analysis of data consistency concepts in data exchange formats (JSON & XML)[C]//Computing, Communication and Automation (ICCCA), 2017 International Conference on. IEEE, 2017:72-77.

[66] 龙建超. 公钥算法中大素数生成方法的研究改进[D]. 云南大学, 2014.

[67] 陈燕. 一种改进的快速 RSA 密钥生成算法[J]. 重庆邮电大学学报(自然科学版), 2011, 23(4):500-504.

攻读硕士学位期间发表的学术论文及研究成果

发表的学术论文：

[1] 周金治, 高磊. 基于多素数和参数替换的改进RSA算法研究[J]. 计算机应用研究, 2019, 36(02): 1-6. （优先出版）

[2] 高磊, 周金治. 基于多重密钥与CRT的数字签名研究[J]. 物联网技术, 2018, 08(04): 39-41.

获得的奖项：

[1] 2016年“华为杯”第十一届中国研究生电子设计竞赛西南赛区三等奖

[2] 2016年“创青春”大赛四川省铜奖

[3] 2017年西南科技大学学生科技活动“先进个人”