doi:10.13582/j.cnki.1672-9102.2014.01.024

云环境下安全外包椭圆曲线点的乘法

胡杏[1,2],唐春明[1]

(1.广州大学 数学与信息科学学院,数学与交叉科学广东普通高校重点实验室,广东 广州 510006;
2.湖南科技大学 数学与计算科学学院,湖南 湘潭 411201)

摘 要:云计算是一种新兴的计算模式,它为云用户提供了强大的计算环境,但同时也引起了用户安全性和隐私性问题的关注.模幂运算是大多数现行的密码系统的基本运算之一,也是公钥密码系统在计算资源限制型设备上的计算瓶颈所在.使用传统的平方-乘算法计算一个n比特的指数的模幂运算,平均需要$1.5n$个模乘,对于资源有限型用户(或设备,例如智能卡)来说,这个计算量是个很沉重的负载.外包计算是云计算模式的优点之一,它使得云用户的计算能力不再受限于各自的资源约束型设备,通过外包工作负载给云,云用户可以使用云提供的无限资源来完成高代价的计算.本文围绕"外包模幂运算"这个问题展开研究,为椭圆曲线的点的乘法的计算问题,提出了一个安全的外包计算方案,并且将本文的方案应用于加速椭圆曲线数字签名的验证.

关键词:云计算;外包计算;椭圆曲线点的乘法;椭圆曲线数字签名

中图分类号:TP309.2 **文献标志码**:A **文章编号**:1672-9102(2014)01-0119-05

Securely outsourcing computation of point multiplication on elliptic curves in cloud computing

HU Xing[1,2], TANG Chun-ming[1]

(1. School of Mathematics and Information Science, Guangzhou University, Key Laboratory of Mathematics and Interdisciplinary Sciences of Guangdong Higher Education Institutes, Guangzhou 510006, China;
2. School of Mathematics and Computational Science, Hunan University of Science and Technology, Xiangtan 411201, China)

Abstract: Cloud computing is a novel computing paradigm which provides powerful computing resources for cloud users. However, this paradigm also brings new challenges for security and privacy. The modular exponentiation is one of basic operations among most of current cryptosystems. It also presents the computational bottleneck in most public-key cryptography on computationally limited devices. Without outsourcing, a device using the traditional square-and-multiply algorithm need 1.5n modular multiplications on average to carry out a modular exponentiation for n-bit exponents. Thus, it is a very time-consuming operation for device with limited computational resources, such as smart card. Cloud computing has a great deal of computational resources and then the computation/storage intensive tasks can be performed through being delegated to resource-abundant cloud servers. A practically and securely outsourced scheme was present, which was used for computing point multiplication on elliptic curves, and was applied to speeding up signature verification of the ECDSA (Elliptic Curves Digital Signature Algorithm).

Key words: cloud computing; outsourced computation; point multiplication on elliptic curves; ECDSA

收稿日期:2013-11-18

基金项目:国家自然科学基金资助项目(11271003);教育部博士点基金博导类项目(20134410110003);广东省自然科学基金资助项目(S2012010009950);广东省高层次人才项目;广州市教育局项目(2012A004)

通信作者:唐春明(1972-),男,湖南怀化人,博士,教授,博士生导师,主要从事密码学、云计算研究. E-mail:ctang@gzhu.edu.cn

1 概述

云计算(cloud computing)是用户通过网络以按需付费的方式获得所需的服务和资源的一种新兴的计算方式. 有了这种方式,资源限制型(包括数据处理能力、存储空间等)用户(client)可以将计算/存储密集型任务委托给资源充裕的云服务器(server)来完成. 传统的计算方式中,用户需要自己建立相应的基础设施来完成全部计算,若用户能够安全的把计算任务委托给云服务器来完成,就可以很大程度地缩减他们的资本支出,减轻用户端的工作负载,提高效率.

云计算为云用户提供了诸多便利,但它也给云用户数据的安全性带来了威胁. 由于用户无法得知云内部的操作细节, 因此, 存在着各种动机, 使得云服务器的行为不诚实. 例如, 对需要大量计算资源的计算, 如果用户无法判断云计算输出的正确性, 云可能会为了节约资源而偷懒". 此外, 还可能存在软件 bug 和恶意的外部攻击, 这些都会影响计算结果的质量. 所以, 云计算环境中的隐私安全、内容安全是云计算研究的关键问题之一.

众所周知,模幂运算是大多数现行的公钥密码系统(例如 RSA 和 ElGmal)在计算资源限制型设备上的计算瓶颈所在. 在用户端独立计算的情形下, 著名的平方 - 乘算法平均需要执行 $1.5n$ 个模乘来完成一个 n 比特指数的模幂运算,因此,对一个计算资源有限的设备来说,模幂运算的代价是非常高昂的. 签名方案是快速求幂的一个主要应用环境. 由于消息签名阶段中需要进行的模幂运算是可以由用户端预计算的,因而,消息签名可以很快完成. 但签名的验证是需要进行在线模幂运算的. 因此,对于一个资源有限的用户,在线验证签名的计算负载是非常重的. 而且,繁重的模幂计算量也影响了签名机制在其他资源有限的设备上的应用,例如智能卡、无线传感设备等.

由此可见,对于资源有限的云用户来说,有快速完成指数运算的迫切需求,而云服务器虽然拥有无限的计算资源但却是不受信任的. 因此,如果能够在云环境中实现安全的外包计算,那么就可以解决资源限制型设备需要执行模幂运算的困境.

2 相关的研究工作

服务器 - 辅助计算是一种特殊的外包计算. 在外包协议中,用户租赁具有强大计算资源的云计算服务商提供的服务器进行计算,而在服务器 - 辅助计算中,用户是委托一个不被信任的服务器来进行计算.

文献[1]中第一次提出了服务器 - 辅助的模幂运算协议. 随后,相继出现了许多有关服务器 - 辅助计算的协议,例如文献[2 - 5].

在文献[2]中,Hohenbergera 和 Lysyanskaya 研究了使用 2 个不受信任的(可能是恶意的)加密辅件 U_1, U_2 来完成安全外包模幂运算的问题. 这个外包方案提及使用 EBPV 生成器[4] 来产生形如 (x, g^x) 的随机对,也包含了对 U_1, U_2 返回的计算结果进行验证的过程. 当用户需要计算一个模幂运算时,需要在用户端执行一个预处理(预计算一些新的模幂运算),另外,这个外包方案在调用 2 个不受信任的(可能是恶意的)加密辅件 U_1, U_2 之前,用户对 U_1, U_2 隐藏了要被外包计算的 2 个参数(底数和基),且假设"2 个不受信任的(可能是恶意的)加密辅件 U_1, U_2 在执行计算过程中不能相互通信",在这个假设下, U_1, U_2 不能相互"勾结",从而保证了数据的隐私性. 但在云环境中,攻击者可能拥有控制所有云服务器的能力,各服务器间是可以相互通信的,那么攻击者就可以利用获得的数据恢复出被隐藏的输入. 因而, Hohenbergera 和 Lysyanskaya 提出的外包方案不适用于云服务器相互串通的云计算环境下的外包计算. 文献[6]中给出了模素数的指数运算的一个新的安全外包算法. 与文献[2]不同的是,该算法是在"单恶意"的模型下的,即,他们假设 U_1, U_2 之中至多只有一个是恶意的. 另外,文献[6]中改进了文献[2]中的 Exp 算法,因而在效率上也优于文献[2].

在文献[7]中,Jakobsson 和 Wetzel 研究了如何减少模幂运算的本地工作负载的问题,通过外包指数运算来生成安全的服务器 - 辅助签名. 在外包模幂运算给服务器之前,用户需要采用盲化的方式隐藏模幂运算的指数. 因此,用户端需要预计算一些值并将这些值存储在本地. 虽然这个方案支持在服务器端批处理模幂计算,但批处理的规模不大. 文献[8]也研究了批处理指数运算的安全外包问题. 在他们的方案中,一批指数(例如 t 个指数运算)可以被高效的外包计算.

另外,在文献[9 - 12]中,关于外包模幂运算的问题也被研究了,主要是围绕公钥密码算法的服务器 - 辅助计算协议这个问题展开研究和探讨,例如,允许将 RSA 签名快速生成用于诸如智能卡之类的低消耗加密处理器. 而在其它一些文献中也提及了加速模幂运算的本地执行的技术,例如文献[13 - 17]. 这些技术都涉及了预处理技术,在提高模幂运算的本地运算效率上具有很大的优越性,但整体的平均计算复杂度,包括离线(预处理)阶段和在线阶段,仍然在 $1.5n$(n 是指数的比特数) 左右,这个数字对于资源限制型设备来说,仍然是很巨大的.

本文针对模幂运算的一个应用实例——椭圆曲线点的乘法的计算问题展开研究,假设执行倍运

算与加法(或减法)花费的时间大致相同,则使用"倍数 - 和差"算法直接计算椭圆曲线点 P 的倍数 aP 平均需要 $1.5l$(其中 $l = \lfloor \log_2 a \rfloor + 1$)个模加,当 a 很大时,这个计算量对资源有限的设备(例如智能卡、无线传感设备等)来说也是不小的计算负载.

本文探讨了如何安全外包计算椭圆曲线点的乘法的问题,将椭圆曲线点 P 的倍数 aP 的计算外包给不受信任的云服务器,同时对云服务器隐藏倍数 a 以及 aP. 而且,用户可以验证云服务器返回的计算结果. 最后,利用这个外包机制来加速椭圆曲线数字签名的验证.

3　外包计算模型

本文考虑的外包计算模型中有 2 个参与实体,即云用户和云服务器. 云用户拥有大量的计算昂贵的计算问题(比如模幂运算),需要外包给云,而云服务器拥有大量的计算资源但却不受用户信任.

这个模型面对的安全性威胁主要来自于云服务器的不诚实行为. 在云环境下,云服务器可能被攻击者收买,根据攻击能力的不同,攻击者可以被分为被动的和主动的 2 种. 被动的攻击者只是得到被收买者的全部信息,但被收买者仍然忠实的执行协议;主动的攻击者可以篡改数据、不忠实的执行协议. 在本文中,假设云服务器的行为是主动的.

从应用角度出发,一个外包协议被称为是高效的可验证协议, 如果它满足如下的 5 个性质:

1)完整性:任何诚实的按协议运行的云服务器能以极大的概率让用户接受它的输出.

2)合理性:任何不诚实的云服务器不能以不可忽略的概率让用户接受它的输出.

3)隐私性:在云服务器执行协议期间,任何来自用户私有数据的敏感信息都不能被服务器推导出来.

4)可验证性:用户能以极大概率验证诚实的云服务器输出的正确性,也能以极大概率验证不诚实的云服务器输出的不正确性.

5)高效性:由用户端完成的本机计算量应充分小于他独自解决原问题的计算量.

4　随机对生成算法

在详细介绍"椭圆曲线上点的乘法的安全外包协议"之前,先描述一个随机对生成算法,在稍后的椭圆曲线点的乘法外包协议中,调用了这个算法作为子程序.

在下文中,令 Z_p 是含有 q 个元素的有限域, E 是定义在 Z_p 上的椭圆曲线,其中 $p > 3$ 是素数. 令 $Q \in E(Z_p)$ 是椭圆曲线 E 上的阶为 q 的一个随机点.

4.1　算法描述

算法 $Pe(\cdot)$ 用于随机生成一个数 x 并返回随机对 $(x, xQ \bmod p)$. $Pe(\cdot)$ 初始化为素数 p 和椭圆曲线上的一个点 $Q \in E(Z_p)$, n, k ($1 \leqslant k \leqslant n$)为 $Pe(\cdot)$ 的参数. 每次调用 $Pe(\cdot)$ 都会产生一个随机的,相互独立的对 $(x, xQ \bmod p)$, 其中 $x \in Z_r^*$, $r = \mathrm{ord}(Q)$. 本文利用了著名的 EBPV 生成器[4]来实现算法 $Pe(\cdot)$.

算法 $Pe(\cdot)$ 包括 2 个阶段:预计算和生成随机对. 描述如下:

预计算:生成 n 个随机数 $\alpha_1, \alpha_2, \cdots, \alpha_n \in Z_r$, 对每个 $j = 1, \cdots, n$, 计算 $Q_j = \alpha_j Q \bmod p$, 然后将 α_j 和 Q_j 存于一张表中.

生成随机对:当需要形如 (x, xQ) 的随机对时,随机的生成集合 $S \subseteq \{1, \cdots, n\}$, 满足 $|S| = k$, 且对每个 $j \in S$, 随机选择一个数 $x_j \in \{1, \cdots, h - 1\}$, $1 \leqslant h \leqslant q - 1$, 然后计算

$$x \equiv \sum_{j \in S} \alpha_j x_j \pmod r \text{ 和 } X \equiv \sum_{j \in S} Q_j x_j \pmod p.$$

如果 $x \equiv 0 \bmod r$, 则重新选择 S, 否则,返回对 (x, X).

注 1:预计算阶段的表是一次性生成的,并存于用户端(保密的).

注 2:对参数 n, k 的选择,则要求 C_n^k 足够大,使得相应的子集和问题是难解的,且生日攻击是不可行的. 一般可取 $n > 500, k = 64$.

4.2　算法 $Pe(\cdot)$ 的效率分析和安全性分析

效率分析:对任何输出对 (x, X), 实际上得到了 $X \equiv xQ \pmod p$, 这个算法需要存储 Z_r 中的 n 个元素 $\alpha_1, \alpha_2, \cdots, \alpha_n$ 和 Z_p^* 中的 n 个元素 $Q_1, Q_2, \cdots, Q_n$. 如果 h 很小,那么计算 x 的代价也是很小的(计算 x 需要执行 $k - 1$ 模乘和 $k - 1$ 个模加,但这 2 个因素都很小),计算 X 也只需要执行 $k - 1$ 个模乘. 特别地,当 $h = 2$ 时,计算 x 不需要占用任何存储空间,只要计算 $k - 1$ 个模加,计算 X 也只需要执行 $k - 1$ 个模加[4]. 尤其是在批处理时,预计算只要执行一次,整个算法的效率将大大提高.

安全性分析:这个算法的安全性可以规约到"子集和问题",而子集和问题已经被证明是一个 NP - 完全问题. 文献[18]指出,给定一组随机的权重 $a_1, a_2, \cdots, a_n$ 和一个模数 M, 一个随机的子集和 $b \bmod M$ 与同样长度的随机串是不可区分的. 在算法 $Pe(\cdot)$ 中,攻击者只能看到输出,即一些子集和(内部产生的),而权重是保密的(由用户端计算). 因此,在实际的应用中选择合适的参数,可以使得解决 $Pe(\cdot)$ 中的子集和问题是困难的,且当初始的随机对(即初始的 (x, xQ) 对,等等)足够多的话, $Pe(\cdot)$ 的输出序列与真正地随机序列是计算上不可区分的[4].

5 安全外包椭圆曲线点的乘法协议(SEC 协议)

本文提出的椭圆曲线点的乘法的安全外包计算协议 SEC 由 4 个步骤构成,即,初始化、发送、响应、验证. 该协议中包含一个用户和一台云服务器 S2 个参与方,每次计算椭圆曲线上点的单个乘积. 若要批计算点的多个乘积,可多次执行.

设 p 是一个大于 3 的素数, E 是定义 Z_p 上的椭圆曲线. 令 P 是 E 上阶为素数 q 的一个点. 用户想要使用云服务器 S 来计算 $aP, a \in Z_q^*$, 其中 a 是秘密参数. SEC(a,P) = aP 描述如下:

● 初始化:用户 4 次调用 $Pe(\cdot)$, 分别得到 (b,bP) 和 $(u,uP),(v,vP),(r,rP)$, 令 $U = uP$. 然后计算 $c = a - b(\mathrm{mod}q)$, 如果 $c \equiv 0 \mathrm{mod} q$, 则调用 $Pe(\cdot)$ 重新生成一个新的随机对 (b,bP).

● 发送:用户将 2 个数对 $(cr^{-1},rP),(uv^{-1},vP)$ 随机(无固定次序)发送给 S.

● 响应:云服务器 S 计算 $cP = cr^{-1} \cdot rP, uP = uv^{-1} \cdot vP$, 然后将计算结果 (cP,uP) 返回给用户.

● 验证:首先,用户验证对 S 进行的测试查询,即校验 uP 的值. 如果 S 返回的值 uP 不满足 $uP = U$ 时,就说明 S 是不诚实工作的,于是,用户拒绝接受 S 所有的计算结果. 否则,用户接受云服务器计算的计算结果,并计算 aP 如下:

$$aP = bP + cP.$$

下面来证明,上述外包计算协议 SEC 是椭圆曲线上点的乘法的可验证安全外包实现.

定理 1:在单 - 云服务器模型中,上述外包计算方案 SEC(a,P)满足完整性要求.

证明:显然,如果云服务器是诚实的按照协议执行的,就一定能让用户接受它的输出. 因为,用户是否接纳由 S 返回的计算结果,取决于对值 uP 的验证. 如果返回的值 uP 满足 $uP = U$, 就说明 S 是诚实工作的.

定理 2:在单 - 云服务器模型中,上述外包计算方案 SEC(a,P)满足合理性要求.

证明:云服务器的不诚实行为是返回 2 个不正确的值给用户. 但由于验证值 U 是秘密的且随机的,云服务器这种不诚实行为是无法通过校验等式 $uP = U$ 的验证的.

定理 3:在单 - 云服务器模型中,上述外包计算方案 SEC(a,P)满足隐私性要求.

证明:从语义上来讲,一个椭圆曲线点的乘法的外包方案满足隐私性的要求,也就意味着恶意攻击者不能推导出关于这个倍乘系数的任何信息. 对该方案的隐私性分析,必须考虑这样的攻击. 假设存在恶意的 PPT 攻击者 Bob 控制了所有的云服务器. 当用户利用 S 完成了 SEC 方案时,Bob 可以得到所有的输入 $(cr^{-1},rP),(uv^{-1},vP)$ 以及输出 cP, uP. 但 b 以及 bP 是随机生成的且对云服务器是隐藏的,因此,攻击者 Bob 无法获得有关秘密参数 a 以及 aP 的任何信息.

定理 4:在单 - 云服务器模型中,上述外包计算方案 SEC(a,P)满足可验证性要求.

证明:对一个 PPT 攻击者 Bob,它想要成功欺骗用户的概率是可忽略的. 因为 $a \in Z_q^*$, 它猜测出验证值 U 的概率是 $\frac{1}{q-1}$, 而猜测出验证值 U 在 $(cr^{-1},rP),(uv^{-1},vP)$ 中的正确位置的概率是 $\frac{1}{2}$.

令 X 表示"攻击者 Bob 伪造结果,成功欺骗用户"这一事件,对一个 PPT 攻击者 Bob,有 $\Pr[X] \leqslant \frac{1}{2(q-1)}$.

定理 5:在单 - 云服务器模型中,上述外包计算方案 SEC(a,P)满足高效性要求.

证明:上述方案 SEC 调用了 4 次 $Pe(\cdot)$ 以及另外 2 个模乘 uv^{-1} 和 cr^{-1} (当然还有 2 个模加,但相比于乘法,执行加法的时间复杂度是可以忽略的). 由 3.2 节中对算法 $Pe(\cdot)$ 的效率分析可知,本文提出的协议是满足高效性要求的.

接下来,将上述安全外包方案 SEC(a,P)应用于 ECDSA 的安全外包实现.

6 安全外包 ECDSA (Elliptic Curves Digital Signature Algorithm)协议

设 p 是一个素数或 2 的幂次方, E 是定义在 F_p 上的椭圆曲线. 设 A 是 E 上阶数为 q 的一个点,使得在 $< A >$ 上的离散对数问题是难处理的. 设 $P = \{0,1\}^*, A = Z_q^* \times Z_q^*$, 定义

$$K = \{(p,q,E,A,m,B): B = mA\},$$

其中 $1 \leqslant m \leqslant q-1$, 值 p,q,E,A 为公开参数, m 为私钥.

该协议由 3 个步骤组成,即,密钥生成、签名生成以及签名验证.

● 密钥生成:由密钥管理中心生成公钥 PK 和私钥 SK, 其中 $PK = B = mA, SK = m$.

● 签名生成: $Sign_K(x,k) = \sigma = (r,s)$

发送方选取一个秘密的随机数 $k, 1 \leqslant k \leqslant q-1$, 给定消息 $x \in \{0,1\}^*$

Step1:发送方计算 $kA = h = (u,v)$

Step2:发送方计算 $r = u \mathrm{mod} q, s = k^{-1}(\mathrm{SHA}-1(x) + mr) \mathrm{mod} q$

(如果 $r = 0$ 或者 $s = 0$, 重新选择 k)

Step3:发送方输出签名 $\sigma = (r,s)$.

● 签名验证:给定公开参数 $PK = B = mA$, 消息 $x \in \{0,1\}^*$, 以及签名 $\sigma = (r,s)$

Step1:接收方计算 $w = s^{-1} \bmod q$

$$i = w * \mathrm{SHA}-1(x) \bmod q$$

$$j = wr \bmod q$$

Step2:接收方分别调用方案 SEC(i,A)和 SEC(j,B),得到 $\alpha = iA$,$\beta = jB$.

Step3:接收方计算 $(u,v) = \alpha + \beta$

Step4:接收方验证 $u \bmod q = r$.

可以证明,上述安全外包 ECDSA 协议是可以抵抗适应性选择消息攻击的.

7　结论

本文提出了一个安全外包椭圆曲线上点的乘法的协议 SEC. 该协议实现了在云环境下的,椭圆曲线上点的乘法的安全外包,将该协议应用于椭圆曲线数字签名的验证阶段,有效地加速了签名的验证.

参考文献:

[1] Matsumoto T, Kato K, Imai H. Speeding up secret computations with insecure auxiliary devices [C]// Advances in Cryptology—CRYPTO'88. Springer New York, 1990: 497-506.

[2] Hohenberger S, Lysyanskaya A. How to securely outsource cryptographic computations [M]. Berlin, Heidelberg: Theory of Cryptography. Springer Berlin Heidelberg, 2005.

[3] Béguin P, Quisquater J J. Secure acceleration of DSS signatures using insecure server [M]. Berlin, Heidelberg: Advances in Cryptology—ASIACRYPT 94. Springer Berlin Heidelberg, 1995.

[4] Nguyen P Q, Shparlinski I E, Stern J. Distribution of modular sums and the security of the server aided exponentiation [M]. Berlin, Heidelberg: Cryptography and Computational Number Theory. Birkhäuser Basel, 2001.

[5] Kawamura S, Shimbo A. Fast server-aided secret computation protocols for modular exponentiation [J]. Selected Areas in Communications, IEEE Journal on, 1993, 11(5): 778-784.

[6] Chen X, Li J, Ma J, et al. New algorithms for secure outsourcing of modular exponentiations [M]. Berlin, Heidelberg: Computer Security - ESORICS 2012. Springer Berlin Heidelberg, 2012.

[7] Jakobsson M, Wetzel S. Secure server-aided signature generation [C]//Public Key Cryptography. Springer Berlin Heidelberg, 2001: 383-401.

[8] Ma X, Li J, Zhang F. Efficient and secure batch exponentiations outsourcing in cloud computing [C]// Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on. IEEE, 2012: 600-605.

[9] Van D M, Clarke D, Gassend B, et al. Speeding up exponentiation using an untrusted computational resource [J]. Designs, Codes and Cryptography, 2006, 39(2): 253-273.

[10] Burns J, Mitchell C J. Parameter selection for server-aided RSA computation schemes [J]. Computers, IEEE Transactions on, 1994, 43(2): 163-174.

[11] Hong S M, Shin J B, Lee-Kwang H, et al. A new appraoch to server-aided secret computation [C]// ICISC. 1998: 33-45.

[12] Ernvall A M, Nyberg K. On server-aided computation for RSA protocols with private key splitting [C]// Proceedings of Nordsec. 2003.

[13] Agnew G B, Mullin R C, Vanstone S A. Fast exponentiation in GF (2n) [C]//Advances in Cryptology—EUROCRYPT'88. Springer Berlin Heidelberg, 1988: 251-255.

[14] Lim C H, Lee P J. More flexible exponentiation with precomputation [C]//Advances in cryptology—CRYPTO'94. Springer Berlin Heidelberg, 1994: 95-107.

[15] De Rooij P. Efficient exponentiation using precomputation and vector addition chains [C]// Advances in Cryptology—EUROCRYPT'94. Springer Berlin Heidelberg, 1995: 389-399.

[16] Boyko V, Peinado M, Venkatesan R. Speeding up discrete log and factoring based schemes via precomputations [M]. Berlin, Heidelberg: Advances in Cryptology—EUROCRYPT'98. Springer Berlin Heidelberg, 1998.

[17] Brickell E F, Gordon D M, McCurley K S, et al. Fast exponentiation with precomputation [C]//Advances in Cryptology—EUROCRYPT'92. Springer Berlin Heidelberg, 1993: 200-207.

[18] Impagliazzo R, Naor M. Efficient cryptographic schemes provably as secure as subset sum [J]. Journal of Cryptology, 1996, 9(4): 199-216.