支持多布尔函数的公开可验证委托方案①

胡海英, 商 威

(北京中电普华信息技术有限公司, 北京 100192)

摘 要: 提出了支持多布尔函数的公开可验证委托模型, 并基于支持非单调访问结构的 KP-ABE(Key Policy Attribute Based Encryption)方案给出了具体的构造. 该方案能够使任何第三方对布尔函数的委托计算结果进行验证. 与 Parno 等人的方案相比较, 该方案通过将输入值与委托计算的布尔函数进行绑定, 实现了一次系统建立后, 可对多个布尔函数进行委托计算, 提高了系统的效率.

关键词: 委托计算; 公开可验证; 布尔函数; 属性加密; 非单调访问结构

Publicly Verifiable Delegation of Multi-Boolean-Function

HU Hai-Ying, SHANG Wei

(Beijing China Power Information Technology Company, Beijing 100192, China)

Abstract: This paper proposes a publicly verifiable delegation model for multi-boolean-function, and gives a construction based on key-policy attribute based encryption (KP-ABE) supporting non-monotonic access structure. In our scheme, any third party could verify the result of the delegation for the boolean-function. Compared to the construction proposed by Parno, our scheme is more efficient, since it can delegate multi-boolean-function after the setup of the system, by binding the inputs to the specific boolean-function .

Key words: delegation of computation; publicly verifiable; boolean-function; attribute based encryption; non-monotonic access structure

随着云计算技术的发展和应用, 将耗时的计算交由云平台来完成成为了计算能力较弱的客户端的一个选择. 但在将计算外包的同时, 如何保证外包计算结果的正确性, 是实际应用中所必须考虑的一个安全性问题.

目前已有的方案大致分为三类:一是基于计算复杂性理论的方案, 主要采用基于交互式证明的技术[1,2]、概率验证技术[3,4]以及基于随机预言机的计算合理性证明技术[5]等知识证明理论构造. 这些技术多是从抽象的图灵机角度去考虑, 因此难以直接在应用系统中使用;二是基于安全硬件的方案, Smith 等人[6]从电子商务的需求出发, 提出了基于安全协处理器的适用于任何计算的方案. Hohenberge[7]则针对加解密运算中计算量较大的问题, 基于特殊的安全模块, 实现了高效的模指数运算的安全委托. 但基于硬件的方案只能保证计算环境的可信性, 而并不保证算法本身以及输入的数据未被篡改. 这也是基于安全硬件的委托计算技术一个内在的缺点;三是基于密码学技术的方案, Gennaro 等人[8]通过借鉴 Yao 等人[9]所提出基于混淆电路的两方安全计算方案, 利用全同态加密技术[10]构造了一个适用于任何计算的可验证委托计算方案. Chung 等人[11]基于 Gennaro 等人所提方案, 针对其中的需要将计算首先转换为二进制逻辑电路这一不足, 提出了一个相对高效的将计算过程作为一个黑盒的改进方案. 但在这两个方案中, 验证密钥都是不公开的, 即只有委托方才能够对计算结果进行验证. 针对这一不足, Parno 等人[12]基于非单调的 KP-ABE 方案[13,14], 提出了一个适用于布尔函数的高效的可公开可验证的委托计算方案, 即用于验证的密钥是公开的, 任何第三方都可以对计算结果进行验证.

① 收稿时间:2012-12-21;收到修改稿时间:2013-01-16

但在 Parno 等人的方案中, 要为每一个布尔函数单独建立一次系, 即在系统建立时就需要确定委托的布尔函数, 无法有效地重用系统参数. 本文针对这一不足, 提出了支持多布尔函数的公开可验证委托模型, 并给出了具体构造, 实现了系统参数的重用, 即系统建立时无需确定需要委托的布尔函数, 一次系统建立可以支持委托多个布尔函数, 提高了系统的整体效率.

1 预备知识

本文的核心方案主要使用了与 KP-ABE 相关的技术. 因此本节首先对 KP-ABE 所涉及的访问结构概念进行了介绍, 其次对 KP-ABE 的算法和安全模型进行了形式化地描述. 在随后第 3 节的核心方案构造以及安全性证明中, 将直接引用本节所介绍的基本概念和符号表示.

1.1 访问结构

访问结构是 KP-ABE 的核心概念, 是 KP-ABE 内在的“一对多”性质的直接体现. 无论是 KP-ABE 的算法模型还是安全模型, 都直接与访问结构相关. 访问结构的具体定义如下:

定义 1.[15]令 $P=\{P_1,P_2,...,P_n\}$是参与方的集合, 一个访问结构$\mathbb{A}$是 2^P 的一个非空子集, 即$\mathbb{A}\subseteq 2^P\setminus\{\emptyset\}$. 若访问结构$\mathbb{A}$是单调的, 则有$\forall$B, C, 若 B$\in\mathbb{A}$且 B$\subseteq$C, 则 C $\in\mathbb{A}$. 访问结构$\mathbb{A}$中的集合称为授权集合, 不在访问结构$\mathbb{A}$中的集合称为非授权集合.

一般来说, 单调访问结构可以通过逻辑“与”和逻辑“或”来表示, 而非单调访问结构需要通过逻辑“非”来表示.

1.2 KP-ABE

一个支持非单调访问结构的KP-ABE方案由4个多项式时间算法(Setup, KeyGen, Encrypt, Decrypt)组成[13].

● Setup(1^λ) $\longrightarrow$ (MSK, PK) : 输入安全参数 1^λ, 输出系统公钥 PK 和主私钥 MSK, 其中 PK 隐含了系统的属性集合;

● KeyGen($\mathbb{A}$, MSK, PK) $\longrightarrow SK_{\mathbb{A}}$: 输入一个访问结构$\mathbb{A}$, 以及主私钥 MSK, 系统公钥 PK, 输出关于访问结构$\mathbb{A}$的私钥 $SK_{\mathbb{A}}$;

● Encrypt(ω, M, PK) $\longrightarrow C$: 输入加密时所使用的属性集合 ω, 明文 M 以及系统公钥 PK, 输出为密文 C, 其中 C 包含了属性集合 ω 的信息;

● Decrypt(C, $SK_{\mathbb{A}}$, PK) $\longrightarrow M$: 输入密文 C, 用户的私钥 $SK_{\mathbb{A}}$, 系统公钥 PK. 若 ω 满足访问结构$\mathbb{A}$, 则输出明文 M; 否则输出 $\perp$.

通过一个攻击游戏来定义支持非单调访问结构的 KP-ABE 方案的安全模型.

● Init: 敌手选择一个属性集合 ω*;

● Setup: 挑战者运行 KP-ABE 方案的 Setup 算法, 将系统公钥 PK 返回给敌手;

● Phase1: 敌手可以询问关于访问结构$\mathbb{A}$的私钥 $SK_{\mathbb{A}}$, 但要求 ω 不满足访问结构$\mathbb{A}$;

● Challenge: 敌手选择两条长度相等明文 M_0, M_1. 挑战者从 M_0 和 M_1 中随机选择一条明文 M_b, 并使用属性集合 ω*对 M_b 进行加密, 并将最终计算出的询问密文 C*返回给敌手;

● Phase2: 与 phase1 相同, 敌手继续提交对用户私钥的询问;

● Guess: 敌手输出对 b 的猜测 b'. 若 $b=b'$, 则敌手获胜;

敌手的攻击优势为: $\varepsilon = \Pr[b=b'] - 1/2$;

定义 2.[13] 一个支持非单调访问结构的 KP-ABE 方案是安全的, 当且仅当对于上述的攻击游戏, 任何多项式时间的敌手的攻击优势是可忽略的;

2 模型定义

2.1 方案模型

一个支持多布尔函数的公开可验证的委托方案由五个多项式时间算法(Setup, KeyGen, ProbGen, Compute, Verify)组成: 其中 Setup 算法由委托方运行, 用于生成系统参数; KeyGen 算法也由委托方运行, 用于生成针对布尔函数的委托计算密钥; ProbGen 算法可由任意第三方运行, 用于在委托计算前, 对布尔函数的输入值进行预处理; Compute 算法由被委托方执行, 用于对布尔函数进行委托计算; Verify 算法可由任意的第三方运行, 用于对委托计算结果进行最终的验证. 这些算法的形式化描述如下:

● Setup(1^λ)$\longrightarrow$(SK, PK) : 输入安全参数 1^λ, 输出系统公钥 PK 和私钥 SK.

● KeyGen(SK, F_i)$\longrightarrow EK_{F_i}$: 输入系统私钥 SK, 委托的布尔函数 F_i, 输出委托计算密钥 EK_{F_i} ;

● ProbGen(PK, x, F_i)$\longrightarrow(\sigma_{F_i,x}, VK_{F_i,x})$: 输入系统公钥 PK, 布尔函数的变量值 x, 指定委托的布尔函数 F_i, 输出 x 编码后的值 $\sigma_{F_i,x}$, 以及用于验证的公开密钥 $VK_{F_i,x}$;

● Compute(PK, EK_{F_i}, $\sigma_{F_i,x}$)$\longrightarrow\sigma_{F_i,y}$: 输入系统公钥 PK, 委托计算的密钥 EK_{F_i}, 编码后的输入值 $\sigma_{F_i,x}$, 输出编码后的计算结果 $\sigma_{F_i,y}$;

● Verify($VK_{F_i,x}$, $\sigma_{F_i,y}$)$\longrightarrow y$ 或者 $\perp$: 输入验证密钥 $VK_{F_i,x}$, 计算值 $\sigma_{F_i,y}$, 若验证通过, 输出 $\sigma_{F_i,y}$ 解码后的值 y, 否则输出 $\perp$.

2.2 安全模型

支持多布尔函数的公开可验证的委托方案的安全性通过如下攻击游戏定义:

● Init: 敌手选定输入的变量值 x^*, 以及挑战的布尔函数 F_i^* (i 是 F_i^* 为分配的唯一标识)发送给挑战者;

● Setup: 挑战者运行方案的 Setup 算法, 并将系统公钥 PK 返回给敌手;

● KeyGen: 对于敌手关于布尔函数 F_j 的询问, 挑战者调用 KeyGen 算法, 将用于委托计算的密钥 EK_{F_j} 返回给敌手;

● ProbGen: 挑战者以敌手选定的变量值 x^*和布尔函数 F_i^* 为参数运行方案的 ProbGen 算法, 并将 x^*编码后的值 $\sigma_{F_i^*,x^*}$ 和用于对计算结果验证的密钥 $VK_{F_i^*,x^*}$ 返回给敌手;

● Challenge: 敌手输出计算结果 $\sigma_{F_i^*,y^*}$;

● Verify: 挑战者运行 Verify 算法, 若输出不为 $\perp$, 且 $y^*\neq F_i^*(x^*)$, 则敌手获胜.

定义 3. 一个支持多布尔函数的公开可验证的委托方案是安全的, 当且仅当对于上述的攻击游戏, 任何多项式时间敌手获胜的概率是可忽略的.

3 具体方案

3.1 基本思想

在 Parno 等人所提的方案中, 由于变量值并未与某个具体的布尔函数绑定, 因此被委托方可以使用多个不同的委托计算密钥对同一个输入进行计算, 并得到相反的且能通过验证的计算结果. 从而对每一个布尔函数, 都需要为其建立一套单独的系统使用.

本文的方案借鉴 Parno 等人方案的思想, 通过为每一个布尔函数分配唯一的标识, 并将该标识作为一个特殊的属性与输入值相绑定, 使得对某一个输入值, 被委托方只能使用唯一的一个与其相对应的密钥进行计算, 实现了输入值与布尔函数的绑定. 在实际使用时, 系统只需建立一次, 即可委托计算多个布尔函数.

3.2 方案

令 ABE 为一个支持非单调访问结构 KP-ABE 方案. 利用 ABE 构造的支持多布尔函数的公开可验证委托方案描述如下:

● Setup: 委托方运行两次 $ABE.Setup$ 算法, 生成两对公私钥对(PK_0, MSK_0)和(PK_1, MSK_1). 系统公钥为 $PK=(PK_0, PK_1)$, 系统私钥 $SK=(MSK_0, MSK_1)$;

● KeyGen: 对于布尔函数 F: 首先为其分配唯一的编号 i, 并且将 i 作为一个属性; 其次, 令 $F_i'=F\wedge i$, $\overline{F_i'}=\overline{F}\wedge i$, 将 $F_i', \overline{F_i'}$ 作为两个访问结构. 然后计算:

$$\begin{cases} EK_{F_i}^0 = ABE.KeyGen\left(F_i', MSK_0, PK_0\right) \\ EK_{F_i}^1 = ABE.KeyGen\left(\overline{F_i'}, MSK_1, PK_1\right) \end{cases}$$

令 $EK_{F_i}=(EK_{F_i}^0, EK_{F_i}^1)$;

● ProbGen: 设委托方需要计算 $F_i(x)$的值. 首先, 将 x 作为一组属性的集合, 令 $\omega=x\cup i$; 其次, 随机选取两条等长的消息 M_0 和 M_1, 计算:

$$\begin{cases} \sigma_{F_i,x}^0 = ABE.Encrypt\left(\omega, M_0, PK_0\right) \\ \sigma_{F_i,x}^1 = ABE.Encrypt\left(\omega, M_1, PK_1\right) \end{cases}$$

令 $\sigma_{F_i,x}=(\sigma_{F_i,x}^0, \sigma_{F_i,x}^1)$;

然后选取一个哈希函数 $hash$, 计算:

$$\begin{cases} VK_{F_i,x}^0 = hash\left(M_0\right) \\ VK_{F_i,x}^1 = hash\left(M_1\right) \end{cases}$$

令 $VK_{F_i,x}=(VK_{F_i,x}^0, VK_{F_i,x}^1)$;

● Compute: 被委托方计算:

$$\begin{cases} \sigma_{F_i,y}^0 = ABE.Decrypt\left(\sigma_{F_i,x}^0, EK_{F_i}^0, PK_0\right) \\ \sigma_{F_i,y}^1 = ABE.Decrypt\left(\sigma_{F_i,x}^1, EK_{F_i}^1, PK_1\right) \end{cases}$$

令 $\sigma_{F_i,y}=(\sigma_{F_i,y}^0, \sigma_{F_i,y}^1)$;

● Verify: 验证方获取到 $VK_{F_i,x}$ 和 $\sigma_{F_i,y}$ 后, 计算最后的输出值 y:

$$y=\begin{cases} 1, & VK_{F_i,x}^0 = hash\left(\sigma_{F_i,y}^0\right)\wedge VK_{F_i,x}^1 \neq hash\left(\sigma_{F_i,y}^1\right) \\ 0, & VK_{F_i,x}^0 \neq hash\left(\sigma_{F_i,y}^0\right)\wedge VK_{F_i,x}^1 = hash\left(\sigma_{F_i,y}^1\right) \\ \perp, & \text{其它} \end{cases}$$

说明: 在实际使用该方案时, 可以将属性空间分

为两个不相交的空间, 其中一个属性空间用于为布尔函数分配唯一的标识, 而另一个属性空间用于布尔函数输入变量的分配. 为了避免在系统建立时显示地指定所支持的布尔函数个数, 可选用支持大属性集合的 *ABE* 方案[14], 以降低系统公钥参数长度.

3.3 安全性证明

定理 1. 若 3.2 节中的构造方案所使用的支持非单调访问结构的 *ABE* 方案在 1.2 节所定义的安全模型下是安全的, 则该构造方案在 2.2 节所定义的安全模型下是安全的.

证明: 采用反证法. 若存在一个敌手 *Adv* 在攻击上述方案时所具有的获胜概率 ε 不可忽略, 则可以构造一个有效的算法 *B*, 以不可忽略的优势攻破所使用的 *ABE* 方案. 算法 *B* 的描述如下:

● Init: B 运行敌手 *Adv*, 敌手选定输入的变量值 x^*, 以及挑战的布尔函数 F_i^*, 不失一般性性, 设 $F_i^*(x^*)=0$. 令 $\omega^*=x^*\cup i$, *B* 将 ω^* 发送给挑战者;

● Setup: 首先, 挑战者运行 *ABE.Setup* 算法, 生成公私钥对(PK_0, MSK_0), 并将 PK_0 返回给 *B*; 其次, *B* 运行 *ABE.Setup* 算法, 生成公私钥对(PK_1, MSK_1). 注意到, *B* 并不完全拥有系统的私钥 *SK*;

● Phase1: 对于敌手关于布尔函数 F_j 的询问: 根据 3.2 中的方案构造, 以及 Init 阶段的假设, 不论 *j* 是否与 *i* 相等, 都有 ω^* 不满足访问结构 F_j', 因此 *B* 可以通过向挑战者询问 F_j' 所对应的私钥获取委托计算密钥 $EK_{F_i}^0$; 同时由于 *B* 拥有 MSK_1, 因此 *B* 可以直接计算 $EK_{F_i}^1$.

● Challenge: *B* 首先选取两条等长的消息 M_0 和 M_1 发送给挑战者, 挑战者从 M_0 和 M_1 中随机选择一条消息 M_b, 计算:

$$\sigma_{F_i^*,x^*}^0 = ABE.Encrypt(\omega^*, M_b, PK_0)$$

然后, *B* 计算:

$$\sigma_{F_i^*,x^*}^1 = ABE.Encrypt(\omega^*, M_1, PK_1)$$

最后 *B* 选取一个哈希函数 *hash*, 计算:

$$\begin{cases} VK_{F_i^*,x^*}^0 = hash(M_0) \\ VK_{F_i^*,x^*}^1 = hash(M_1) \end{cases}$$

并将编码值 $\sigma_{F_i^*,x^*}=(\sigma_{F_i^*,x^*}^0, \sigma_{F_i^*,x^*}^1)$ 和验证密钥 $VK_{F_x^*,x^*}=(VK_{F_i^*,x^*}^0, VK_{F_i^*,x^*}^1)$ 发送给敌手;

● Guess: 敌手返回 $\sigma_{F_i^*,y^*}$. *B* 运行 Verify 算法: 若输出为 $\perp$ 或者为 0, 则 *B* 输出对 *b* 的一个随机猜测; 若输出为 1, 则 *B* 输出对 *b* 的猜测 *b*'=0;

分析:

● 若 *b*=0, 则 $VK_{F_x^*,x^*}$ 是一个合法的验证密钥, 此时敌手的获胜概率为 ε, 从而 *B* 猜测正确的概率为:

$$\begin{aligned} &\Pr[b'=b \mid b=0] \\ &=\Pr\left[Verify\left(VK_{F_i^*,x^*},\sigma_{F_i^*,y^*}\right)\neq 1\right]\Pr[b'=0] \\ &\quad+\Pr\left[Verify\left(VK_{F_i^*,x^*},\sigma_{F_i^*,y^*}\right)=1\right]\Pr[b'=0] \\ &=\frac{1}{2}(1-\varepsilon)+\varepsilon \\ &=\frac{1}{2}\varepsilon+\frac{1}{2} \end{aligned}$$

● 若 *b*=1, 则 $\sigma_{F_i^*,x^*}^0$ 是对 M_1 加密后的结果, 因此 $VK_{F_x^*,x^*}$ 并不是合法的验证密钥. 根据 *hash* 函数的性质, $hash(\sigma_{F_i^*,y^*}^0)=hash(M_1)$ 的概率是可忽略的, 因此敌手获胜的概率也是可忽略的.

最终, *B* 攻击 *ABE* 的优势为:

$$\begin{aligned} &\Pr[b=b']-\frac{1}{2} \\ &=\Pr[b=b']\Pr[b'=0]+\Pr[b=b']\Pr[b'=1]-\frac{1}{2} \\ &=\frac{1}{2}\left(\frac{1}{2}+\frac{\varepsilon}{2}\right)+\frac{1}{2}\cdot\frac{1}{2}-\frac{1}{2} \\ &=\frac{\varepsilon}{4} \end{aligned}$$

定理 1 证明完毕.

在上述证明中,我们假设在 Init 阶段,有 $F_i^*(x^*)=0$. 因此在 Phase1 阶段, 即使 *j*=*i*, 属性集合 ω^* 仍然不满足访问结构 F_j', 因此可以询问挑战者相应的密钥. 若敌手选定的挑战值有 $F_i^*(x^*)=1$, 则在攻击游戏中, 挑战者与 *B* 的角色对调即可, 即由 *B* 来生成(PK_0, MSK_0)和 $EK_{F_i}^0$, 而由挑战者来生成(PK_1, MSK_1)和 $EK_{F_i}^1$.

4 结语

本文在 Parno 等人所提出的布尔函数公开可验证委托方案的基础上, 通过将委托计算值与指定的布尔函数相绑定, 实现了支持多布尔函数的公开可验证委托方案. 该方案能够在一次系统建立后, 委托计算多个不同的布尔函数, 解决了 Parno 等人所提的方案中一次系统建立只能委托计算一个布尔函数的不足, 提高了系统的效率.

参考文献

1 Goldwasser S, Micali S, Charles R. The knowledge complexity of interactive proof-systems. Proc. of the 17th annual ACM symposium on Theory of computing. New York: ACM, 1985: 291–304.

2 Goldwasser S, Kalai YT, Rothblum GN. Delegating computation: interactive proofs for muggles. Proc. of the 40th Annual ACM Symposium on Theory of Computing. New York: ACM, 2008: 113–122.

3 Kilian J. Improved efficient arguments. In: Coppersmith D, ed. Advances in Cryptology-CRYPTO'95. Berlin: Springer-Verlag, 1995: 311–324.

4 Kalai Y, Raz R. Probabilistically checkable arguments. In: Halevi S, ed. Advances in Cryptology-CRYPTO'09. Berlin: Springer-Verlag, 2009: 143–155.

5 Micali S. CS proofs. Proc. of the 35th annual symposium on Foundations of Computer Science. Washington DC: IEEE Computer Society, 1994: 436–453.

6 Smith SW, Weingart S. Building a high-performance, programmable secure coprocessor. Computer Networks, 1999, 31(8):831–860.

7 Hhenberger S, Lysyanskaya A. How to securely outsource cryptographic computations. In: Kilian J, ed. Theory of Cryptography. Berlin: Springer-Verlag, 2005: 264–282.

8 Gennaro R, Gentry C, Parno B. Non-interactive veriable computing: Outsourcing computation to untrusted workers. In: Gennaro R, Gentry C, Parno B, eds. Advances in Cryptology-CRYPTO'10. Berlin: Springer-Verlag, 2010: 465–482.

9 Yao AC. Protocols for secure computations. Proc. of the 23rd Annual Symposium on Foundations of Computer Science. Washington DC: IEEE Computer Society, 1982: 160–164.

10 Gentry C. A fully homomorphic encryption scheme[Ph.D. Thesis]. Stanford University, 2009.

11 Chung KM, Kalai Y, Vadhan S. Improved delegation of computation using fully homomorphic encryption. In: Gennaro R, Gentry C, Parno B, eds. Advances in Cryptology – CRYPTO'10. Berlin: Springer-Verlag, 2010: 483–501.

12 Parno B, Raykova M, Vaikuntanathan V. How to delegate and verify in public:Verifiable computation from attribute-based encryption. In: Parno B, Raykova M, Vaikuntanathan V, eds. Theory of Cryptography-TCC 2012. Berlin: Springer-Verlag, 2012: 422–439.

13 Lewko A, Sahai A, Waters B. Revocation systems with very small private keys. Proc. of the 2010 IEEE Symposium on Security and Privacy. Washington DC:IEEE Computer Society, 2010: 273–285.

14 Ostrovsky R, Sahai A, Waters B. Attribute-based encryption with non-monotonic access structures. Proc. of the 14th ACM conference on Computer and Communications Security. New York: ACM, 2007: 195–203.

15 Beimel A. Secure Schemes for Secret Sharing and Key Distribution[Ph.D. Thesis]. Israel Institute of Technology, 1996.