Reading Notes DiSE: distributed symmetric encryption

August 21, 2023

这篇文章是 2018-727 DiSE: Distributed Symmetric-key Encryption 的读书笔记, 作者在这篇文章中提出了分布式对称加密的概念, 并提出了一种可以基于任何分布式伪随机函数 (DPRF) 进行阈值认证加密的通用构造方案.

阈值密码学是密码学中的重要组成部分, 通过将秘密密钥共享给多个参与方来保护密钥, 然后共同执行密码操作. 攻击者即使腐蚀了一定数量的参与方也无法恢复秘密或破坏安全, 在上一篇文章 Secret Sharing 中我们已经对阈值密码学中尤为重要的秘密分享概念进行了性质和方案上的讲解.

令人惊讶的是, 在这篇文章之前, 没有分布式对称密钥加密的正式定义和处理方案. 阈值密码学的先前正式处理通常集中在非对称密钥设置上, 即公钥加密和签名方案, 其中签名/解密密钥和算法在多个参与方之间分布. 如上文所言作者在此文中提出了分布式对称加密的概念和一种通用构造方案, 我们首先来看作者对这一概念给出的模型和定义:

定义 (阈值对称密钥加密) 阈值对称密钥加密方案 TSE 由一个元组 (Setup, DistEnc, DistDec) 给出，满足下面的一致性属性:

- $\text{Setup}(1^\kappa, n, t) \to (\llbracket \mathbf{sk} \rrbracket_{[n]}, pp)$: Setup 是一个随机算法, 接受安全参数作为输入, 并输出 n 个密钥 $sk_1, \ldots, sk_n$ 和公共参数 pp. 第 i 个密钥 sk_i 分配给第 i 个参与方.
- $\text{DistEnc}(\llbracket \mathbf{sk} \rrbracket_{[n]}, [j : m, S], pp) \to [j : c/\perp]$: DistEnc 是一个分布式协议, 通过它, 参与方 j 在集合 S 中的参与方的帮助下加密消息 m. 在协议结束时， j 输出密文 c(或 $\perp$ 表示失败). 所有其他参与方没有输出.
- $\text{DistDec}(\llbracket \mathbf{sk} \rrbracket_{[n]}, [j : c, S], pp) \to [j : m/\perp]$: DistDec 是一个分布式协议, 通过它, 参与方 j 在集合 S 中的参与方的帮助下解密密文 c. 在协议结束时, j 输出消息 m(或 $\perp$ 表示失败). 所有其他参与方没有输出.

一致性 对于任何 $n, t \in \mathbb{N}$ 使得 $t \le n$, 对于所有由 $\text{Setup}(1^\kappa)$ 输出的 $(\llbracket \mathbf{sk} \rrbracket_{[n]}, pp)$, 任何消息 m, 任何两个满足任意一个集合的元素个数均大于阈值 t 的集合 $S, S^{'} \subset [n]$, 以及两个集合中的任何两个参与方 $j \in S, j^{'} \in S^{'}$ 并且假定所有参与方都诚实行事, 那么存在一个可以忽略的函数 negl, 使得

$$\Pr\left[[j' : m] \leftarrow \text{DistDec}\left(\llbracket \mathbf{sk} \rrbracket_{[n]},\ [j' : c, S'], pp\right) \mid [j : c] \leftarrow \text{DistEnc}\left(\llbracket \mathbf{sk} \rrbracket_{[n]}, [j : m, S], pp\right)\right] \ge 1 - \text{negl}(\kappa),$$

其中概率体现在参与 DistEnc 和 DistDec 的参与方的随机投掷上.

1 预备知识

1.1 PRF: Pseudo-random Function

PRF 是一种数学函数, 具有确定性, 但其输出在统计上与随机函数的输出无法区分. 它涉及一个设置过程, 其中每个参与方都获得自己的密钥. 对输入的计算由密钥持有者执行. PRF 应满足两个主要要求: (i) 确定性: 对于相同的密钥和输入, 输出始终相同; (ii) 伪随机性: 对于不知道密钥的观察者, 输出应该是随机的.

定义 (伪随机函数) 伪随机函数 PRF 是由两个算法 (Setup, Eval) 组成的元组.

- $\text{Setup}(1^\kappa) \to sk$. Setup 算法生成密钥 sk.
- $\text{Eval}(sk, x) \to y$. Eval 算法为给定值 x 生成输出 y。通过使用密钥 sk 和输入 x 运行 Eval 来计算输出.

确定性 对于任何 $x \in \mathbb{X}$，所有由 $\text{Setup}(1^\kappa)$ 生成的密钥 sk，Eval 的输出始终相同：

$$\text{Eval}(sk, x) = \text{Eval}(sk, x).$$

伪随机性 对于不知道密钥 sk 的观察者，Eval 的输出 y 应该是随机的。即，对于所有多项式时间算法 A，存在一个可以忽略的函数 negl，使得

$$|\Pr[A(\text{Eval}(sk, x)) = 1] - \Pr[A(r) = 1]| \leq \text{negl}(\kappa),$$

其中 r 是从与 y 相同的分布中随机抽取的值，概率体现在 sk 的选择和 Eval 使用的随机性上. PRF 与分组密码有着紧密的联系. 由于两者在概念上十分接近, 所以当我们设计了一个以 PRF 为基本模块的密码体制后, 我们可以在理论上使用 PRF 的安全模型证明其安全性, 而在实际应用时, 可以用安全的分组密码代替 PRF 而不失安全性.

1.2 DPRF: Distributed Pseudo-random Function

文章中所有构造都使用分布式伪随机函数（DPRF）作为构建块. DPRF 是标准 PRF 的分布式模拟. 它涉及一个设置过程，其中每个参与方都获得自己的密钥和公共参数. 对输入的计算由任何 t 个参与方共同执行, 其中 t 是一个阈值 ($t \leq n$). 重要的是, 在协议结束时, 只有一个特殊的参与方（评估者）了解输出. DPRF 应满足两个主要要求:(i) 一致性：评估应独立于参与方集合; (ii) 伪随机性：即使敌手破坏了所有其他 t−1 个参与方并恶意行事, 评估的输出对除评估者以外的所有人都应该是伪随机的.

定义 (分布式伪随机函数) 分布式伪随机函数 (DPRF) DP 是由三个算法 (Setup, Eval, Combine) 组成的元组，满足一致性属性.

- $\text{Setup}(1^\kappa, n, t) \to ((sk_1, \ldots, sk_n), pp)$. Setup 算法生成 n 个密钥 $(sk_1, sk_2, \ldots, sk_n)$ 和公共参数 pp. 第 i 个密钥 sk_i 分配给第 i 个参与方.
- $\text{Eval}(sk_i, x, pp) \to z_i$. Eval 算法为给定值生成伪随机份额. 参与方 i 通过使用 sk_i, x 和 pp 运行 Eval 来计算值 x 的第 i 个份额 z_i.
- $\text{Combine}(\{(i, z_i)\}_{i \in S}, pp) =: z/\perp$. Combine 算法将集合 S 中的参与方的部分份额 $\{z_i\}_{i \in S}$ 组合以生成值 z. 如果算法失败，则其输出由 $\perp$ 表示.

一致性 对于任何 $n, t \in \mathbb{N}$ 使得 $t \leq n$，所有由 $\text{Setup}(1^\kappa, n, t)$ 生成的 $((sk_1, \ldots, sk_n), pp)$，任何输入 x，任何两个大小至少为 t 的集合 $S, S' \subset [n]$，存在一个可以忽略的函数 negl，使得

$$\Pr[\text{Combine}(\{(i, z_i)\} i \in S, pp) = \text{Combine}(\{(j, z'_j)\} j \in S', pp) \neq \perp] \geq 1 - \text{negl}(\kappa),$$

其中对于 $i \in S$ 有 $z_i \leftarrow \text{Eval}(sk_i, x, pp)$, 对于 $j \in S'$ 有 $z'_j \leftarrow \text{Eval}(sk_j, x, pp)$. 这一性质表明, 满足符合阈值人数的参与方参与恢复秘密时, 无论哪些人参与, 恢复出的秘密保持一致.

1.3 Commitment

定义 1.3 一个 (非交互的) 承诺方案 Σ 包含了两个多项式时间复杂度的算法 $(\text{Setup}_{com}, \text{Com})$ 并且满足如下两个称之为" 隐藏性" 和" 绑定性" 的性质:

• $\text{Setup}_{com}(1^\kappa) \to pp_{com}$: 它以安全参数 κ 作为输入, 并输出公共参数 pp_{com}.

• $\text{Com}(m, pp_{com}; r) =: \alpha$: 它以消息 m、公共参数 pp_{com} 和随机数 r 作为输入, 并输出一个承诺 α.

隐藏性

如果对于所有 PPT 对手 A, 任意两条消息 m_0, m_1, 存在一个可忽略的函数 negl, 使得对于 $\text{ppcom} \leftarrow \text{Setupcom}(1^\kappa)$, A 永远无法分辨出承诺阶段被加密的消息是 m_0 还是 m_1, 数学语言表达为:

$$|\Pr[A(\text{ppcom}, \text{Com}(m_0, \text{ppcom}; r_0)) = 1] - \Pr[A(\text{ppcom}, \text{Com}(m_1, \text{ppcom}; r_1)) = 1]| \leq \text{negl}(\kappa)$$

绑定性

如果对于所有 PPT 对手 A, 如果 A 输出 m_0, m_1, r_0 和 r_1 $((m_0, r_0) \neq (m_1, r_1))$. 给定 $\text{ppcom} \leftarrow \text{Setupcom}(1^\kappa)$, 那么存在一个可忽略的函数 negl, 使得 A 能够通过伪造消息和随机数来使得结果通过 Com 函数运算得到相同承诺的概率可以忽略, 数学表达为:

$$\Pr[\text{Com}(m_0, \text{ppcom}; r_0) = \text{Com}(m_1, \text{ppcom}; r_1)] \leq \text{negl}(\kappa)$$

这个定义描述了一个承诺方案, 它是密码学中用于确保一方在未来不会更改其声明的工具. 隐藏性确保在解密前对手不能从承诺中获取有关隐藏消息的信息, 而绑定属性确保承诺方不能更改其承诺的消息. 这在许多密码协议中有重要作用, 特别是在需要确保公平性和诚实性的情况下.

1.4 PPT: Probabilistic Polynomial Time

在密码学和计算机科学中, "PPT" 是"Probabilistic Polynomial Time" 的缩写, 中文可以翻译为 "概率多项式时间" . 这是一种用来描述算法复杂性的术语.

PPT 算法是一种可以在多项式时间内运行的随机算法. 这意味着算法的运行时间是输入大小的多项式函数, 并且算法在执行过程中可以访问某些随机位. 这些随机位可以用于做随机选择, 从而可能影响算法的输出.

在密码学中, PPT 常用来描述诸如攻击者或加密方案等实体的计算能力. 当我们说一个算法是 PPT 的时候, 我们通常是在强调该算法在实际中是可行的, 因为它的运行时间与输入大小成多项式关系, 而不是指数关系. 这与一些理论上可行但实际上运行时间过长的算法形成对比.

1.5 一些符号约定

在密码学中，符号 || 通常用来表示连接（concatenation）操作, 这一标记法在本协议也有涉及. 这种记法表示如果你有两个字符串或位序列 A 和 B, 那么 A||B 就是将 B 附加到 A 的末尾所得到的新字符串或位序列. 这个操作在许多密码学算法和协议中都很常见, 特别是在构造复杂的消息或密钥材料时. 在本协议中, 需要使用这一连接操作来实现对于消息完整性的验证.

2 协议流程

2.1 预设

在这个协议中我们需要事先准备几个在前文介绍过的函数:

- 一个 (n,t) 门限的 DPRF 协议 $DP := (\mathrm{DP}.\mathrm{Setup}, \mathrm{Eval}, \mathrm{Combine})$.
- 一个多项式复杂的伪随机数生成器 PRG.
- 一个承诺方案 Σ:=(Σ .Setup, Com).

2.2 初始化阶段

$\mathrm{Setup}(1^\kappa, n, t) \to \big([\![\mathbf{sk}]\!]_{[n]}, \mathrm{pp}\big)$: 运行 $\mathrm{DP.Setup}(1^\kappa, n, t)$ 获取 $((\mathrm{rk}_1, \ldots, \mathrm{rk}_n), pp_{DP})$ 和 $\Sigma.\mathrm{Setup}(1^\kappa)$ 获取 pp_{com}. 设置 $\mathrm{sk}_i := \mathrm{rk}_i$，其中 $i \in [n]$, 并且公共参数 $\mathrm{pp} := (pp_{DP}, pp_{com})$.

2.3 分布式加密

$\mathrm{DistEnc}([\![\mathbf{sk}]\!]_{[n]}, [j : m, S], \mathrm{pp}) \to [j : c/\perp]$: 要在集合 S 中的参与方的帮助下加密消息 m:

- 方 j 计算 $\alpha := \mathrm{Com}(m, pp_{com}; \rho)$，其中 ρ 是随机选择的，并将 α 发送给 S 中的所有方.
- 对于 $i \in S$，方 i 运行 $\mathrm{Eval}(\mathrm{sk}_i, j||\alpha, \mathrm{pp})$ 获取 z_i，并将其发送给方 j.
- 方 j 运行 $\mathrm{Combine}(\{(i, z_i)\} i \in S, \mathrm{pp})$ 获取 w 或 $\perp$. 在后一种情况下，它输出 $\perp$. 否则，它计算 $e := \mathrm{PRG}(w) \oplus (m||\rho)$，然后输出 $c := (j, \alpha, e)$.

2.4 分布式解密

$\mathrm{DistDec}([\![\mathbf{sk}]\!]_{[n]}, [j : c, S], \mathrm{pp}) \to [j : m/\perp]$: 要在集合 S 中的参与方的帮助下解密密文 c:

- 方 j' 首先将 c 解析为 (j, α, e). 然后，它将 $j||\alpha$ 发送给 S 中的所有方.
- 对于 $i \in S$, 方 i 接收 x 并检查它是否为 $j^*||\alpha^*$ 的形式，其中 $j^* \in [n]$. 如果不是，则它发送 $\perp$ 给方 j'. 否则，它运行 $\mathrm{Eval}(\mathrm{sk}_i, x, \mathrm{pp})$ 获取 z_i，并将其发送给方 j'.
- 方 j' 运行 $\mathrm{Combine}(\{(i, z_i)\} i \in S, \mathrm{pp})$ 获取 w 或 $\perp$。在后一种情况下，它输出 $\perp$。否则，它计算 $m||\rho := \mathrm{PRG}(w) \oplus e$ 并检查 $\alpha = \mathrm{Com}(m, pp_{com}; \rho)$ 是否成立. 如果检查成功，它输出 m; 否则，它输出 $\perp$.