Searchable Encryption for Conjunctive Queries with Extended Forward and Backward Privacy 阅读笔记

李忆诺

2024 年 5 月 1 日

1 基本信息

1.1 论文来源

Zuo C, Lai S, Yuan X, et al. Searchable encryption for conjunctive queries with extended forward and backward privacy[J]. Cryptology ePrint Archive, 2021.

1.2 概述

本文提出了两种 DSSE 方案，命名为 SDSSE-CQ 和 SDSSE-CQ-s，其核心是引入两个连接查询来完善其安全性。除此之外，本文提出两个新的后向隐私级别，用于更准确地描述使用 OXT 框架的连接查询的泄露。

2 论文要点

2.1 背景

大多数前向与后向私有 DSSE 方案只支持单个关键字查询，即使是 Patranabis 等人基于 OXT 框架引入用于连接查询的前向与后向私有 DSSE，其对于连接查询的安全性也不够全面，并且“延迟删除”带来了更多通信成本，且在某些情况下无法删除文件。本文基于这些问题，提出对基于 OXT 框架的前向与后向私有 DSSE 方案的改进操作。

2.2 价值

- 提出了一种基于 OXT 框架的，新的带有连接词查询的 DSSE 方案 SDSSE-CQ，并保证尽可能减少客户端与服务器之间的交互，该方案可以保证"TSET" 和"XSet" 的前向隐私；
- 在 SDSSE-CQ 的基础上，提出了 SDSSE-CQ-s，该方案可以通过增加"XSet" 进一步减少 SDSSE-CQ 的泄露；
- 为上述两种方案引入了两个新的后向隐私定义级别 $Type-O$ 和 $Type-O^-$，其中 $Type-O^-$ 比 $Type-O$ 更加安全；

- 与 ODXT 相比，本文提出的方案在计算量和通信量增加较少的情况下实现了更好的安全性。

2.3 问题陈述

本文基于最先进的单关键字 DSSE 方案 Aura 进行改进。Aura 实现了非交互式删除，并且满足前向与 $type-ii$ 型后向隐私。

Aura 方案包含的算法与协议如下：

- $(\sigma, EDB) \leftarrow \Sigma \cdot \textbf{Setup}(1^\lambda)$：由客户端运行，通过输入安全参数 λ，输出状态 σ 和加密数据库 EDB；
- $(\sigma', EDB') \leftarrow \Sigma \cdot \textbf{Update}(op, (w, ind), \sigma; EDB)$：该协议在客户端与服务器之间运行，客户端输入操作 op，一个 (w, id) 对和状态 σ，服务器输入加密数据库 EDB，最终，客户端输出更新后的状态 σ'，服务器输出更新后的加密数据库 EDB；
- $\textbf{Res} \leftarrow \Sigma \cdot \textbf{Search}(w, \sigma; EDB)$：该协议在客户端与服务器之间运行，客户端输入一个关键字 w，状态 σ，服务器输入加密数据库 EDB，最终，客户端输出搜索结果 **Res**。

由于 ODXT 方案有两个数据集”TSet” 和”XSet”，但只有”TSet” 的前向隐私能够保证，”XSet” 则不能。为了解决这个问题，本文希望提出两种带有连接关键词查询的 DSSE 方案，旨在改进 ODXT 方案的前向隐私性，支持非交互式删除，并进一步减少泄露。

2.4 方法

2.4.1 SDSSE-CQ

该方案进行了如下改动：

- $(\sigma, EDB) \leftarrow \Sigma \cdot \textbf{Setup}(1^\lambda)$：由客户端运行，通过输入安全参数 λ，然后从伪随机函数 F 中选择一个密钥 k，从伪随机函数 F_p 中选择密钥 k_x, k_i, k_z；设置一个空图 CT，用于存放 (w/c) 对；定义两个 Aura，用于”TSet” 和”XSet”，输出状态 $\sigma = (k, k_x, k_i, k_z, CT, \sigma_T, \sigma_X)$ 和加密数据库 $EDB = (EDB_T, EDB_X)$；
- $(\sigma', EDB') \leftarrow \Sigma \cdot \textbf{Update}(op, (w, ind), \sigma; EDB)$：该协议在客户端与服务器之间运行，客户端输入操作 op，一个 (w, id) 对和状态 σ，服务器输入加密数据库 EDB，客户端生成加密标识符 e, y 和 $xtag$，然后客户端与服务器通信，要求根据关键字 w 更新 $(e||y||c)$ 和 $xtag$，最终，客户端输出更新后的状态 σ'，服务器输出更新后的加密数据库 EDB；
- $\textbf{Res} \leftarrow \Sigma \cdot \textbf{Search}(w, \sigma; EDB)$：该协议在客户端与服务器之间运行，客户端输入一个多关键字查询 $q = (w_1, w_2, \ldots w_n)$，状态 σ，服务器输入加密数据库 EDB，首先，客户端根据最低频繁关键字 w_1 从 CT 中获得 c，然后他根据每一个 i 和 j 生成 $xtoken$，发送给服务器，之后客户端与服务器通信，从 EDB_T 中寻找包含关键字 w_1 的文件，和从 EDB_X 中寻找包含关键字 $(w_2, \ldots, w_n)$ 的文件，服务器筛选出最终查询结果并返回给客户端。最终，客户端解密出搜索结果。

该方案实现了多关键字下 DSSE 方案的前向安全，因为它保证了"TSet" 和"XSet" 两个数据库均满足前向安全。

除此以外，新搜索到的 $xtag$ 仍然可以被先前发出的搜索查询使用（使用不同的 w_1），本文称之为 **Type-O 后向隐私**。

为了进一步减小泄露，本文引入了更强的后向隐私级别 $Type-O^-$ ，在该定义下，新生成的 $xtag$ 不能被先前发出的搜索查询使用。

为了实现 $Type-O^-$，本文根据 (w/c) 对生成一个新的随机数，并将其加入 $xtag$

2.4.2 SDSSE-CQ-s

该方案在 SDSSE-CQ 的基础上进行了如下改进：

- $(\sigma, EDB) \leftarrow \Sigma \cdot \textbf{Setup}(1^\lambda)$：从伪随机函数 F_p 中额外选择密钥 k'_x, k'_z，输出状态 $\sigma = (k, k_x, k_i, k_z, k'_x, k'_z, CT, \sigma_T, \sigma_X)$；
- $(\sigma', EDB') \leftarrow \Sigma \cdot \textbf{Update}(op, (w, ind), \sigma; EDB)$：根据 (w/c) 对生成一个新的随机数，并加入 $wxtag$ 中，用于实现 $Type-O^-$ 后向隐私；
- $\textbf{Res} \leftarrow \Sigma \cdot \textbf{Search}(w, \sigma; EDB)$：客户端输出一个根据 w_1 生成一个随机函数 $F_p = (k'_z, w_1)$ 给 $wxtokens$，并生成新的 token$wxk[i][k]$，当服务器想要恢复 $xtag$ 时，$F_p(k'_x, w_j||c_j)$ 将会被去除，$F_p(k'_z, w_1)$ 会被添加到新的 $xtag$ 上，因此这些 tags 只能被用于当前搜索过程。

2.5 结果

两种方案对基于 OXT 框架的多关键词查询 DSSE 方案的安全性更加全面，其中 SDSSE-CQ 满足前向与 $Type-O$ 后向隐私，SDSSE-CQ-s 满足前向与 $Type-O^-$ 后向隐私。

除此之外，由于本文提出的两种方案无需将删除的文件发送至服务器，因此减少了服务器与客户端的交互，有效降低了通信成本。

3 评论

3.1 局限性

1. 由于本方案需要为"TSet" 和"XSet" 生成"CSRE" 密文,因此两种方案的添加时间比 ODXT 要高；
2. 本文方案的搜索时间会随着 w_2 相关文件的数量增加而增加，ODXT 的搜索时间只受 w_1 影响，不同的 w_2 搜索时间几乎相同，因此本文方案的搜索时间复杂度并不稳定
3. 为了实现更高的安全性，本文的两种方案其更新成本与通信成本要略大于 ODXT。

3.2 扩展阅读

Sun S, Steinfeld R, Lai S, et al. Practical non-interactive searchable encryption with forward and backward privacy[C]//Usenix Network and Distributed System Security Symposium 2021. Internet Society, 2021.

3.3 启示

看完这篇论文后，我发现我上次看 ODXT 方案是实际是囫囵吞枣的。该论文与 ODXT 方案相结合，让我对 ODXT 方案本身有了更深刻的理解，如"TSet" 和"XSet" 的作用，及是否满足前向安全等等。与此同时，该篇论文从提高安全性与减少通信成本两个方面，在 ODXT 方案的基础上进行改进，引入 Aura 结构并通过添加随机数进一步维护其安全性，效果良好。

也许在理解 ODXT 程序代码之后，可以在其基础上进行尝试，以实现该方案。