

TDSC'19 - Vasco - A key for John Doe: modeling and designing Anonymous Password-Authenticated key Exchange protocols

概述: (S) 匿名PAKE协议设计包括客户端匿名和服务器端匿名, 之前的实现没有考虑服务器端匿名, 且客户端匿名通常需要客户端存储或记忆额外的信息。(T) 本文希望设计一个 password-only 的APAKE方案, 实现C和S端的匿名。(A) 首先重新定义了安全模型和敌手的能力, 给出了两种构造方法, 一种基于OPRF, 一种基于LBE, 将两种方案与同类方案进行了对比与分析, 并在ROM下进行了形式化安全证明。(R) 结果表明构造方案可实现key secrecy和anonymity, 且运算开销可接受。

优点: ①文中构建敌手优势时考虑Zipf's Law. ②首次给出了APAKE的形式化定义
③给出了与相关方案的对比与分析

问题: ①匿名性只针对honest-but-curious敌手.
②本文并未显式定义及考虑forward secrecy, 应详细描述这一点.
③本文在评估时, 未给出选择参数的原因, 例如 $\mu=50$, 而选择接近实际口令的参数会更合理.

1. Anonymous PAKE的匿名: (双方只能得知 p 是否属于 D)

1> Client Anonymity: S 无法区分 C 的口令 p 是否在数据库 D 中

2> Server Anonymity: D 中的信息不会泄漏给 C

APAKE是在认证层实现匿名性的关键.

2. 原来的APAKE可以分为两类, 一种是需要客户端存储的匿名凭证系统, 另一种是需要用户记忆index的基于OT的方案, 均不符合 password-only 的要求.

3. 通用构造APAKE (Client仅持有及记忆口令) \rightarrow 本文旨在实现 password-only 的APAKE

▲ 口令字典 $D \subseteq \{0,1\}^*$, 假设口令的长度相同, 表示为 μ

▲ 参与者 clients, 每一个持有 D 中的一个口令, 个数为 k_c

servers, 每一个持有 D 中的一个有限口令集合, 个数为 k_s

集合 $C := \{C_1, \dots, C_{k_c}\}$, $S := \{S_1, \dots, S_{k_s}\}$, $\mu = CUS$, $CNS = \phi, z_1, \dots, z_{k_c}, j_1, \dots, j_{k_s}$ 为不同数字

▲ 协议概览: 客户端 C 持有口令 $p \in D$, 服务器 S 持有 $Ds \in D$ (大小为 n_s)

1> offline phase: server 输入安全参数 1 , 输出参数集合 $params$, 并可选地输出高熵的 master key k , 其中 $params$ 与client共享.

2> online phase: server 输入 Ds 和 k , client 输入 p , 若 $p \in Ds$, 则可以分离生成 sk, sid 和 pid .

4. Security Model. 基于BPR Model, 根据APAKE的安全需求给出特定的修改, 另外 server 在每次执行中可能使用多个口令

1> 每一个实体 $U_i \in U$, 可能并行执行多项式个协议的instance, 表示为 U_i^t
通信网络由敌手控制, 可以延迟, 插入和删除消息.

2> 协议实例 U_i^t 的变量

- ① used $_{U_i^t}$: 实例是否已用于协议运行
- ② state $_{U_i^t}$: 维护协议执行时的状态信息
- ③ term $_{U_i^t}$: 标识执行是否终止
- ④ sid $_{U_i^t}$: 可能的公共会话的标识符, 可作为本次执行的 sk 的标识符.
- ⑤ pid $_{U_i^t}$: 两实体选择的唯一的临时label, 目的是在此次执行时建立key
- ⑥ test $_{U_i^t}$: 标识实例是否被敌手test过, 初始化为false
- ⑦ acc $_{U_i^t}$: 标识实体是否接受 sk
- ⑧ sk $_{U_i^t}$: 标识 sk 的值, 在accept前值为NULL.

3> 敌手能力

- ① Send(U_i^t, M): 主动, 发送消息 M 给实例 U_i^t 并返回响应 (可开启一次协议执行)
- ② Execute(U_i^t, U_j^t): 被动窃听, 敌手获取到所有的通信信息
- ③ Reveal(U_i^t): 返回 $sk_{U_i^t}$ 及 $sid_{U_i^t}$
- ④ Test(U_i^t): 针对key secrecy, 只针对fresh的会话 (对于重复的查询, 答案保持一致)
- ⑤ Corrupt(U_i): 返回实体持有的口令或口令集 (在本文中与anonymity相关)
- ⑥ Reveal $^t(S_j^t)$: 表示servers的honest-but-curious行为, 返回offline phase生成的 k

4> 本文中的几个概念 (与PAKE略有不同)

▲ Correctness (不允许使用send oracle)

最终双方实体: $acc_C = acc_S = true$, $sid_C = sid_S$, $pid_C = pid_S$, $sk_C = sk_S \neq NULL$

在 $p \in Ds$ 时有极大概率发生, 而当 $p \notin Ds$ 时, $acc_C = acc_S = false$ 也是从极大概率发生

▲ Partnering: C_i^t 与 S_j^t 为partnered, 除了correctness属性外, 还要求与其它实体不构成此类关系

▲ Freshness: 若 U_i^t 为fresh, 则 acc_C 为true, 且敌手未询问过Reveal(U_i^t)或其伙伴Reveal(U_j^t)
也没有询问Corrupt(U_j)并随后询问其伙伴Send($U_i^t, *$)

▲ 本文不考虑内部攻击者.

▲ key secrecy: $Adv_A(t) \leq \epsilon(t, q) + negl(t)$, t 为安全参数, q 为send(U)的次数

5> Anonymity


APAKE中, C 和 S 可以不共享口令, anonymity通过口令是否会泄漏来决定

先前研究: ①未考虑server时集的泄漏 (即server匿名性) ②不允许敌手 (作为server) 使用不同数据库识别窃听者口令

APAKE 考虑敌手可利用多个 database (与多个 server 交互), 在一次执行中的两个实体可能未共享口令, 而要求敌手不知道双方在执行后, 产生了相同的 key. (无法通过 Execute() 得知是否 accept)

▲ 本文的定义限制了 Reveal, Reveal⁺ 和 Corrupt Oracle 的使用
 例如, 若要区分两个口令哪一个 client 持有的, 对于与该 client 交互且含有其中一个的 server, 敌手不可以执行 Reveal.

① Distinguishing Clients and Servers

△ 选择两个口令 p_0, p_1 , 若 S 是 distinguishing, 则 $|D_S \cap \{p_0, p_1\}| = 1$, 即 S 中 R 有 Δ 选到个
 △ 选择两个集合 D_0, D_1 , 若 C 是 distinguishing, 则 $p \in D_0 \Delta D_1$. Δ 代表 

a) Client Anonymity: 敌手 A 和 challenger Ch .
 A 选择两个口令 p_0, p_1 , 并发送给 Ch , Ch 随机选 p_b 并在合法 Client C_{ch}
 A 不可以使用 Send, 也不可以 Reveal 与 C_{ch} 交互的 S_i (distinguishing)
 也不可以对 distinguish S 使用 Reveal⁺ 并与 C_{ch} 交互.
 也不能 Reveal 或 Corrupt C_{ch} .

若 A 可以得知 b , 则胜利 (注意定义包含了先 Corrupt(S_i) 再执行 Execute 的情景)
 $Adv_A^{anon}(t) := |2 \cdot Succ(t) - 1| \leq negl(t)$

b) Server Anonymity: 捕获在协议执行时数据库的泄漏
 A 选择两个口令集 D_0, D_1 , 发送给 Ch , Ch 选择其中一个并作为 S_{ch} 参与协议 ($|D_0| = |D_1|$)
 A 不可以 Reveal 与 S_{ch} 交互的 distinguishing Client, 也不能 Reveal 或 Corrupt S_{ch} .
 同样有: $Adv_A^{anon}(t) := |2 \cdot Succ(t) - 1| \leq negl(t)$

▲ 在 Anonymity 中包含了 password-privacy.

5. Tools.

1> Authenticated Encryption: $AEnc = (keyGen, Enc, Dec)$. 机密性 + 认证
 可以通过 block cipher 与 MAC 实现, 若是敌手生成的密文, 解密算法极大概率输出 INVALID

2> OPRF: Π 实现功能函数 $g(v, w) = (L, f_r(w))$, $\{f_r\}_{r \in P}$ 为伪随机函数族
 两个阶段: ① $\Pi.Setup$: 输出 $\{f_r\}_{r \in P}$ 的参数 $params_\Pi$

② $\Pi.Run$ 交互协议 C 输入 w , S 输入 y , 输出 $f_r(w)$, 但 S 无法获知其它有关 w 的信息

3> IBE. blind anonymous identity based encryption scheme IBE

- ① IBE.Setup 输入安全参数 L , 输出公共参数 $params$, 和 msk
- ② IBE.Extract 输入 msk 和 id , 输出与 id 相关的 $skid$.
- ③ IBE.Enc 输入消息 m 和 id , 输出密文 c .
- ④ IBE.Dec 输入 c 和 $skid$, 输出明文消息 m
- ⑤ IBE.BlindExtract 交互协议, 客户端输入 id , 服务器输入 msk , 协议终止时, 客户端输出 $skid$, 服务器不输出.

▲ Id-privacy
 Ch 运行 IBE.Setup 生成 $params$ 和 msk 并转发给 A ; A 生成两个 id : id_0 和 id_1 发送给 Ch .
 Ch 均匀随机选择 b , m , 并计算 $c = IBE.Enc(params, id_b, m)$, 并发送给 A .
 A 输出对 b 的猜测

b. OPRF-based APAKE ($OPRF \Pi + AEnc = (keyGen, Enc, Dec)$ 4个哈希函数 G, H, G', H')

C

Offline Phase

S

for AEnc

$ks, params_{AEnc} := keyGen(1^t)$

$params_\Pi := \Pi.Setup(1^t)$

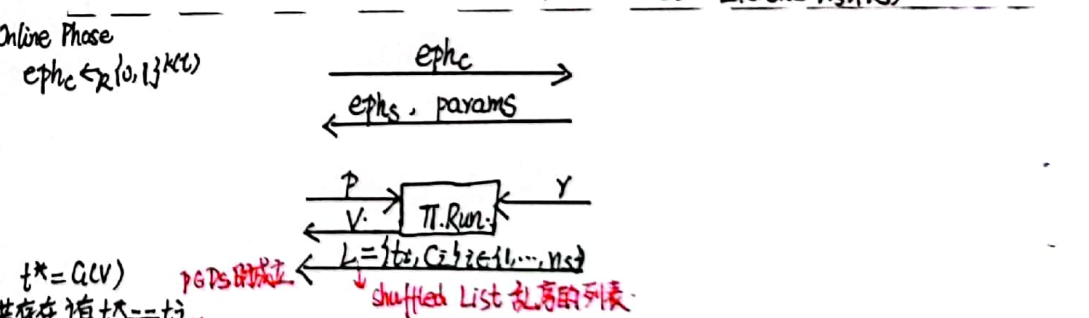
$params := params_{AEnc}, params_\Pi$

$N_s \leftarrow_R \{0, 1\}^{m(t)}, y \leftarrow_R P, eph_s \leftarrow_R \{0, 1\}^{k(t)}$

For $i \in \{1, \dots, N_s\}$ ephemeral

$ti := G(fr(pi)), ki := H(fr(pi))$ identity.

$ci := Enc(ki, N_s || ks)$



$t^* = G(v)$ 若存在 $t^* = t_j$ peps 的成立
 $kp := H(v), N_s || ks := Dec(kp, c_j)$
 $N_c \leftarrow \{0, 1\}^{m(t)}, c := Enc(ks, N_c)$
 $acc_c := TRUE, pids := (eph_s || eph_c)$
 $side := G'(c || pids || N_s)$
 $sk_c := H'(pids || N_s)$
 否则: $C \leftarrow \{0, 1\}^{t(t)}$
 $acc_c := FALSE$

C

S

$N_c := Dec(ks, c)$

若 $N_c \neq L$: $pids := (eph_s || eph_c)$

$side_s := G'(pids || N_s)$

$sk_s := H'(pids || N_s)$

否则: $acc_s := FALSE$

保护窃听者无法得知协议执行是否成功

7. IBE-Based APAKE (在D中的pw可用作 client identities, 服务器实例为ID; 哈希函数H, G, G')

C

S

offline phase

$msk, params := IBE.Setup(C)$
 $N_s \leftarrow \mathcal{R}\{0, 1\}^{m(c)}$, $eph_s \leftarrow \mathcal{R}\{0, 1\}^{k(c)}$
 For $i \in \{1, \dots, n_s\}$.
 $C_i := IBE.Enc(p_i, N_s)$
 $sk_{p_i} := IBE.Extract(p_i)$

online phase

$eph_s, params$

$eph_c \leftarrow \mathcal{R}\{0, 1\}^{k(c)}$
 $N \leftarrow \mathcal{R}\{0, 1\}^{m(c)}$
 $\gamma := IBE.Enc(p, N)$

eph_c, γ

$\xrightarrow{p} \boxed{IBE.BlindExtract} \xleftarrow{msk} sk_p$

For $i \in \{1, \dots, n_s\}$.
 $L = \{C_i, t_i | i \in \{1, \dots, n_s\}\}$, $t_i := H(IBE.Dec(sk_{p_i}, \gamma))$

$H(N)$ 当 $p = p_i$ 时

$t^* := H(N)$

若存在 $t^* = t_i$

$N_s := IBE.Dec(sk_p, C_i)$

$N_c \leftarrow \mathcal{R}\{0, 1\}^{m(c)} \setminus \{0^{m(c)}\}$

$acc_c := TRUE$

$pid_c := (eph_s || eph_c)$

$sid_c := G'(pid_c || N_c || N_s)$

$sk_c := H'(pid_c || N_c || N_s)$

否则: $N_c := 0^{m(c)}$ \rightarrow 保证与弱实例不可区分

$acc_c := FALSE$

$C := IBE.Enc(ID, N_c) \xrightarrow{C} sk_{ID} := IBE.Extract(ID)$

只有 server 生成 sk_{ID}

才可以解密

$N_c := IBE.Dec(sk_{ID}, C)$

若 $N_c \neq 0^{m(c)}$:

$acc_s := TRUE$

$pid_s := (eph_s || eph_c)$

$sid_s := G'(pid_s || N_c || N_s)$

$sk_s := H'(pid_s || N_c || N_s)$

否则 $acc_s := FALSE$