AsiaCrypt'20-Erwig-Fuzzy Asymmetric Password-Authenticated Key Exchange

概述：(S) PAKE有两种适用于实际场景的变体，aPAKE和fPAKE，一方面，两者并未被同时研究过。另一方面faPAKE有特定的应用场景(可使用fuzzy认证数据且server不会拿到认证数据)，(T)作者对两者的结合进行研究，在untrusted server和noisy passwords的情景完成协议(A)作者完成了两种构造，一种使用RSS和OT以及汉明距离完成构造，并在UC框架中进行了形式化分析，另一种是从aPAKE中构造faPAKE (R)在UC框架中完成了安全性证明，并且进一步通过性能评估，表明方案的可行性(消息和吃文件大小与方案的错误容忍能力无关)

优点：①首次研究了fuzzy PAKE及aPAKE的结合 ② 安全性、性能及可用性分析都给出了描述

③给出了两种faPAKE的构造形式给出了详细的描述与分析。

问题：①本文的方案要求使用Ideal Cipher，但未给出实际的实现方法

②本文未考虑 pre-computation table 攻击

③本文提出的第二种方案，故手可以一次性排除k个吃，在真实环境中面对成本时较大

④假设的吃在存在字符不一致时仍可完成密钥协商，这一种构造思路应放在实际场景中进行安全性分析。

▲PAKE的两个变体及不适合直接用于fuzzy aPAKE的原因

1> asymmetric PAKE：当认证S不是完全可信时即使吃文件丢失也可保证吃安全，如A-EKE(CCS'93) 但当吃非完全匹配时，无法完成协议，当前没有aPAKE方案可带有fuzzy属性 ← 无可用password hardening技术

2> fuzzy PAKE：允许吃不是完全匹配，适用于fuzzy设定，例如生物特征作吃，但是当吃直接存在S中时，会泄漏用户隐私，另外fuzzy密码方案通常不适用于低熵的吃。

▲安全模型.

1> $\mathcal{F}_{aPAKE}$ 由DHP+假设于fPAKE 和 CMR06的$\mathcal{F}_{apwKE}$组合而成，当$d(pw, pw') \le \delta$时，密钥交换可以成功，方案可扩展性较好。

2> Roles: P_C 持有吃pw；P_S 可以访问FILE中的值，由pw'生成。但P_S无法得知pw'的值.

3> 故手的能力

①.故手一次协议执行只能猜测1次吃. TESTPWD

②.在破坏server后，故手可以访问吃文件FILE.，但故手能控制实体或更改实体的内部状态，STEALPWDFILE；此时故手可以做离线字典攻击，标记为OFFLINETESTPWD ← 存储者为(OFFLINE, pw)

③故手可以预计算 H(pw)，在获取吃文件后从预计算值中拿到客户端吃 ← 使用于.

④故手可以利用被破坏的服务器的FILE，与用户进行 key exchange，IMPERSONATE

4> δ为success threshold，γ为security threshold，允许使用ECC等为构建块证明安全性，可以更准确地描述安全性.(当$\delta=\gamma$时，可实现最优安全性)

▲Fuzzy aPAKE from Secret Sharing

以使用Hamming distance作为度量吃相似度的方法。 Π_{faPAKE}

User $(\widetilde{pw}, g)$	Server (pw, t, q, g)
File Registration Phase	
parse $\widetilde{pw} =: \widetilde{pw}_1 \| \cdots \| \widetilde{pw}_n$	parse $pw =: pw_1 \| \cdots \| pw_n$. $n \leftarrow \|pw\|$, $l \leftarrow n-2t$, $k \xleftarrow{R} Z_q$, $K \leftarrow g^k$, $P \leftarrow g^{pw}$ $(s_1, \cdots, s_n) \leftarrow Share^n_{l,q}(k)$, $(r_1, \cdots, r_n) \xleftarrow{R} Z_q^n$ $a_{pw_i, i} \leftarrow g^{s_i}$, $i \in [n]$, $a_{pw_i \oplus 1, i} \leftarrow g^{r_i}$, $i \in [n]$. 存储 FILE $\leftarrow ((a_{0,i}, a_{1,i})_{i\in[n]}, P, K)$. 删除 $pw, k, (s_i)_{i\in[n]}, (r_i)_{i\in[n]}$
key Exchange Phase	$k' \xleftarrow{R} Z_q$, $K' \leftarrow K^{k'}$, $A \leftarrow (a_{0,i}^{k'}, a_{1,i}^{k'})_{i\in[n]}$
$\mathcal{F}_{IC}$ ⟷ Ideal Cipher.	(ENC, K', (A, P)) → C ; C →
$\mathcal{F}_{OT}^n$	(REC, $(\widetilde{pw}_i)_{i\in[n]}$) → $(b_i)_{i\in[n]}$; (SEND, A)
	$k_S \leftarrow PRG(k')$, 输出 k_S. 传输.
	(DEC, K, C) → (A', P') 解密.
parse $(\widetilde{a}'_{0,i}, \widetilde{a}'_{1,i})_{i\in[n]} \leftarrow A'$.	
若$\exists i$有$\widetilde{a}'_{\widetilde{pw}_i, i}$或者$\widetilde{pw}$有$d(\widetilde{pw}, \widetilde{pw}) < t \wedge g^{\widetilde{pw}} = P'$. → 无法完成协议	
则令 $x \xleftarrow{R} Z_q$，否则令 $x \leftarrow k'$	
令 $k_C \leftarrow PRG(x)$，输出 k_C	

▲$(n, l-1, l+t)$-RSSExp 方案，其中 $n = l+2t$，协议中 $\gamma = 2t$，$\delta = t$，该协议可以对抗 static. byzantine corruptions 及 adaptive server compromise.

▲ 安全证明概要：给定1个独立于吃的仿真的协议，证明其与真实协议不可区分.

1> Honest session: ① C和S的一个交互通过UC-Secure OT完成

②另一次互通过IC，由于IC是生成1个均匀随机的密文，故simulator可以将IC的输出替换为任意值(但未知k)，故simulator可以在独立于password执行

2> Corrupted client：此时C要一次性提交n个吃bit，而不能基于之前的OT输出自适应地改变吃bit. 通过使用non-adaptive n次1-2 OT执行完成。故手A可以在仿真C的OT输出前，询问TESTPWD. 如果TESTPWD返回为S的吃，A可以仿真出合法的OT输出，否则A输出1个与真实执行不可区分的输出值

3> Corrupted server：当S发送了包含OT输入和 g^{pw} 时，A将pw从输入重构为IC和来自environment的generic group operations. A可以通TESTPWD检查pw是否与C的口令足够接近，若足够多接近，simulator可以获得客户端口令，并仿真C. 否则C的行为与口令无关，A可以仿真1个任意的与S的口令不接近的值.

4> Server Compromise.

① 仿真口令文件. A构建1个表格，其中随机群元素句柄作为口令文件，另有1个与g^k相关的F随机句柄。当环境Z通过询问CGM对元素中的子集做解码，A可以拿到这些询问. 若是子集与口令相关，则simulator提交口令到OFFLINETESTPWD，若响应中包含S的口令，A会运行CGM使解码结果在g^k句柄中.

② 伪装攻击. Z可以使用1个file来伪装成server，Z必须更改密文C来加密文件. 当Z向F_{IC}发送1个包含元素P的加密询问时，simulator检查CGM是否包含(pw, P)，若是，A就可以对pw执行TESTPWD，且当pw与pw足够接近时获得C的口令pw

若没有(pw, P)，A检查P是否从文件(A', P')通过向CGM发送$f^{r'}(P)$得到，若是，A可使用相同的f发起IMPERSONATE query.

5> MITM attack on honest session：由于C和S的通信除了UC-secure的OT之外，只有从S发向C的IC的输出，若Z篡改这个值，可以被检测到. A可以据此来仿真.

▲在实际应用中，口令必须加盐，不存储$g^{(sid||pw)}$，sid为相应的会话标识符，此时使用相同口令的两个C的口令文件不同.

汉明距离适合在生物特征的应用中测量两个口令的距离.

▲Fuzzy aPAKE from standard aPAKE (更适用于用户记忆的口令的方式)

从aPAKE构造faPAKE，本质是C与S运行aPAKE多次. 输入所有与初始注册口令接近的口令

close(pw) := {pwi | d(pw, pwi) < δ}，产生所有的认证口令.

例如δ代表着忽略大小写被忽略 close(holy-moly!) = {Holy-moly!, holy-moly!}.

k := |FILE|，代表口令的数目，C与S执行aPAKE k次，C每次输入口令，S输入集合中的值

S也需要加密口令文件证明行为的正确性. C可以解密并查找相应的口令文件，最后使用解密密钥执行显式认证(否则S未知输出的key)，C可以识别所有的失败的解密并拿到最后的结果

若希望纠正n-bit口令的任意1个错误，口令文件大小为 k=n+1.

若至多有t个错误，则 $k := 1 + \sum_{i=1}^{t} \binom{n}{i}$

▲敌手一次性可以排除k个口令

▲考虑的攻击 F'_{faPAKE}

1> Passive attacks：Z只有使用3相应的输入请求了F_{IC}和F_{RO}才可区分真实和仿真的输出，由于输入是诚实实体均匀随机选取的，故可区分的概率极低.

2> Active message tampering：Z可以在1协议中插入信息，加密向量 e_i 和显式认证消息h. 替换h会导致输出的key不同，通过NEWKEY发送上来仿真.

而修改它，应只考虑其中的一个组件，否则会降低输出相同key的概率，simulator需调整各方输出相同密钥的概率，即以相同概率输出key，不同则通过NEWKEY输出上

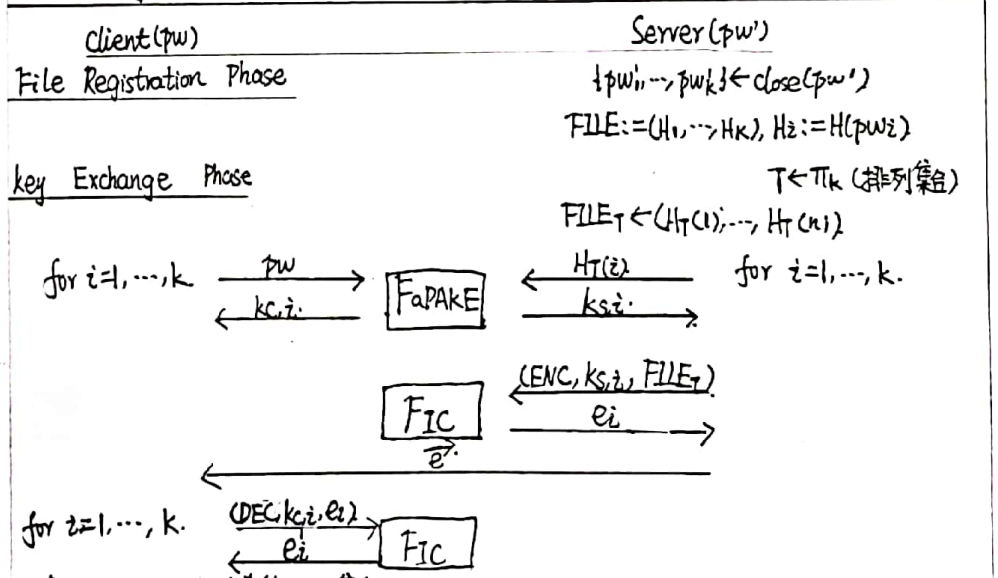
3> (Static) Byzantine corruption：

① corrupted server，给定敌手计算的它，simulator生成k个可使用的password，通过TESTPWD给F'_{faPAKE}并使用真实输出继续仿真. 若corrupted server不遵守协议，则通过NEWKEY接口输出上表明仿真失败.

② corrupted client：利用k个被立的TESTPWD询问类似处理.

4> Server compromise：口令文件在未知口令的情况下通过随机哈希值仿真. 若Z计算H(pw)，A可以提交pw到OFFLINETESTPWD接口；当得到server的真实口令时，A运行RO使口令文件包含所有与pw相近的口令的哈希值

5> Attacking FaPAKE：需仿真FaPAKE，使其与F'_{faPAKE}的应用匹配.



▲Robust Secret Sharing in the exponent

t-out-of n secret sharing scheme 允许将secret S分成n份，给定至少t个share 可以重构，而小于t构成的元组与S是独立的

RSS 引入了malicious share，$(n, l-1, r)_q$-RSS是1个t-out-of-n secret sharing scheme，允许n-r个corrupted shares，有r个不同的secret share可以保证S的重构.

▲对于1个向量 $c \in F_q^n$，集合 $A \subseteq [n]$，C_A代表映射 $F_q^n \to F_q^{|A|}$

▲$(n, t, r)_q$ RSS方案 包含2个概率算法 Share: $F_q \to F_q^n$， Rec: $F_q^n \to F_q$

右确保 t-privacy：对于 $s, s' \in F_q$，若 $|A| \le t$，则 Share(s)和 Share(s')在|A|上的映射相互独立

r-robustness: 若 $|A| \ge r$，则与 Share(s)相同映射的 c，可以重构出 s

▲RSS in the exponent (牺牲了Rec的正确率)

令 RSS=(Share', Rec') 为 $(n, t, r)_q$ RSS方案，$G=\langle g \rangle$ 为阶为q的循环群，$(n, t, r)_q$ RSSExp 方案包含两个概率算法：Share: $F_q \to G^n$； Rec: $G^n \to G$.

Share(s)：输入 $s \leftarrow F_q$，得：$(s_1, \dots, s_n) \leftarrow$ Share'(s)，输出为 $(g^{s_1}, \dots, g^{s_n})$.

Rec $(g^{\hat{s}_1}, \dots g^{\hat{s}_n})$：输出 $g^{\hat{s}}$，其中 $\hat{s} \leftarrow$ Rec'$(\hat{s}_1, \dots, \hat{s}_n)$

其中 r-robustness 变为有可忽略的概率存在 $Rec(\hat{c}) = g^{\hat{s}}$，但 t-privacy 可以实现

▲原始文献说明可以从任意MDS构建RSS，$(n+1, k)_q$ MDS code 为1个linear q-ary code，长度为n，rank为k，可以修正至多 $\lfloor (n-k+1)/2 \rfloor$ errors.

▲本文专注于linear codes的解码算法：从 t-out-of-(l+2t) Shamir's secret sharing 构建1个 $(n, l-1, l+t, q)$-RSSExp 方案，

l-1 privacy 可由DHPT18得知，Rec为 unique decoding by randomized enumeration 该方法会为 random subset 解码直到找到冗余，通过拉格朗日插值对指数中的share应用，若 $t < (n+1-l)/2$，$t \cdot l = O(n \cdot \log n)$ 则Rec会有极大概率成功，需要 $\text{poly}(n) \cdot O(\log q)$ 次操作。由于 $t < (n+1-l)/2$，有 (l+t)- robustness 完成.

▲F_{aPAKE}，安全参数 λ，S为成功阈值，r为安全阈值，$S \le r$，敌手S. 实体 Pc, Ps

1. Password Registration.

①Ps 给的 (STOREPWDFILE, sid, Pc, pw)，若为第1个STOREPWDFILE消息，记录(FILE, Pc, Ps, pw)，标记为 uncompromised.

2. Stealing Password Data.

①敌手S给的 (STEALPWDFILE, sid)，若无记录(FILE, Pc, Ps, pw)，返回"no password file"给S

否则，若 record uncompromised，标记为 compromised；另外，对于所有的记录 (OFFLINE, pw')，计算 $d \leftarrow d(pw, pw')$，若 $d \le S$，发送 ("correct guess", pw') 给 S；若没有 pw' 记录，返回 "password file stolen" 给 S

②敌手S给的 (OFFLINE-PWD, sid, pw')：

若记录 (FILE, Pc, Ps, pw) 标记为 compromised，令 $d \leftarrow d(pw, pw')$

- 若 $d \le S$，record 标记为 compromised，发送 correct guess 给 S
- 若 $d > S$，record 标记为 interrupted，发送 wrong guess 给 S

否则，记录 (OFFLINE, pw')

3. Password Authentication.

① Pc 给的 (USRSESSION, sid, ssid, Ps, pw')，

发送 (USRSESSION, sid, ssid, Pc, Ps) 给 S，若这是 ssid 的第1个 USRSESSION，记录 (ssid, Pc, Ps, pw') 并标记为 fresh.

②Ps 给的 (SRVSESSION, sid, ssid)

检索 (FILE, Pc, Ps, pw)，发送 (SRVSESSION, sid, ssid, Pc, Ps) 到 S，若这是 ssid 的第1个 SRVSESSION 消息，记录 (ssid, Ps, Pc, pw) 并标记为 fresh.

4. Active Session Attacks.

① S 给的 (TEST-PWD, sid, ssid, P, pw')，若有记录 (ssid, P, P', pw) 标记为 fresh，令 $d \leftarrow d(pw, pw')$.

- 若 $d \le S$，标记记录为 compromised，并发送 correct guess 给 S.
- 若 $d > S$，标记记录为 interrupted，并发送 wrong guess 给 S

②S 给的 (IMPERSONATION, sid, ssid, f)，若记录 (ssid, Pc, Ps, pw) 标记为 fresh，且记录 (FILE, Pc, Ps, pw') 标记为 compromised，令 $d \leftarrow d(pw, f(pw'))$

- 若 $d \le S$，记录为 compromised，发送 correct guess 给 S.
- 若 $d > S$，interrupted，wrong ....

5. key Generation and Implicit Authentication

①S 给的 (NEWKEY, sid, ssid, P, k)，$|k| = \lambda$，$k = \perp$，若记录 (ssid, P, P', pw) 未标记 completed：

- 若记录标记为 compromised，或 P, P' 之一 corrupted，发送 (sid, ssid, k) 给 P
- 若记录为 fresh，(sid, ssid, k') 发送给 P'，若另有记录 (ssid, P', P, pw) 且 $d(pw, pw') \le S$ 为 fresh，则 (sid, ssid, k') 发送给 P
- 否则，令 $k'' \xleftarrow{R} \{0,1\}^\lambda$，并发送 (sid, ssid, k'') 给 P.

最后令 (ssid, P, P', pw) 为 completed.

▲更改的 distance check：

$d \le S$，	compromised，	响应 $L_c(pw, pw')$
$S < d \le r$，	compromised，	响应 $L_m(pw, pw')$
$r < d$，	interrupted，	响应 $L_f(pw, pw')$

▲Leakage function 的例子

① No leakage $L_c^N(pw,pw') = L_m^N(\cdot,\cdot) = L_f^N(\cdot,\cdot) = \perp$

② Correctness of guess $L_c(pw,pw') = $"Correct guess"

$L_m(pw,pw') = L_f(pw,pw') = $"wrong guess"

③ Matching positions ("mask")

$L_c^M(pw,pw') = (\{i\} \text{ s.t. } pw[i] = pw'[i], \text{"correct guess"})$

$L_m^M(\cdot,\cdot) = (\ldots, \text{"wrong guess"})$

$L_f^M(pw,pw') =$ "wrong guess"

④ Full Password

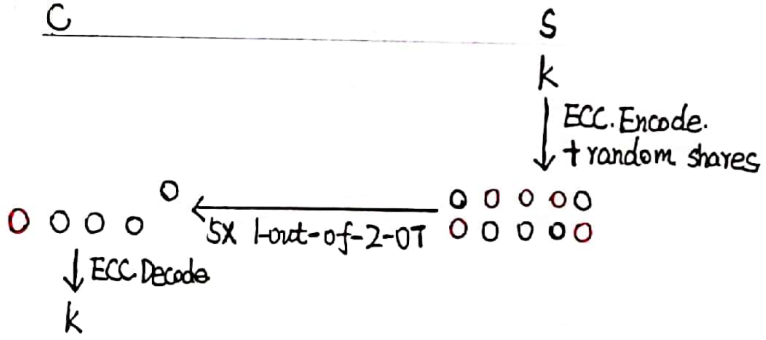
$L_c^P(pw,pw') = L_m^P(pw,pw') = pw, \quad L_f^P(pw,pw') =$ "wrong guess"

▲Fuzzy aPAKE from Secret Sharing

S使用ECC来编码1个密钥k,结果codeword传输给C, C解码并得到密钥

为了使K变成口令依赖的(与口令有关的), S在口令文件中存储了codeword及randomness, 真实的codeword的位置由口令bit标识.

例:



S的口令为01110, 存在一定的randomness, C不需要获取整个口令文件, 而是每列只看到1个, 可以通过1-2 OT协议. C的口令的错误在error correction threshold之内就可以让C解码得到K。上图中, C使用的口令为11110, 也就是5号是正确的。

此时,故手面对的是猜测codeword, 比如在(n-2t)-n RSS中, 需找到(n-2t) shares, $\binom{n}{m}$ 从n个元素中取m个的可能性,

故攻击者直接取成功的概率为: $\frac{\binom{n}{t}}{\binom{n}{t}\cdot 2^{n-2t}} = \frac{1}{2^{n-2t}}$, n为口令空间大小, t为容忍的错误数

但上述协议只能生成1个k. 但若故手使用pw和pw由1个, 则可运行2次后得到password file

解决方法是随机化每次协议运行中的口令文件

为了在UC框架中证明安全,需要1个理想假设,故手可以得知对某个口令的猜测,且口令在以固定的形式存储在文件中,也为了在adaptive server compromise的情况下证明安全性.

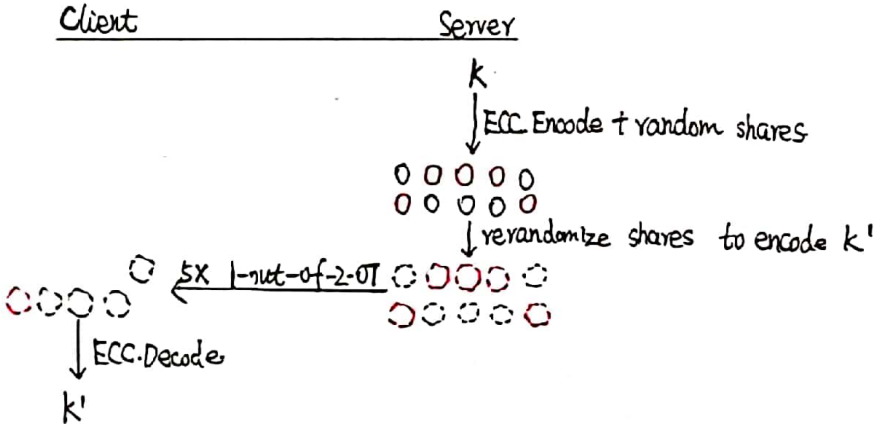
→在generic group model证明,让口令文件在指数上存在。C也需要在指数上解码,

▲考虑恶意client和server.

1> malicious client: 输入错误口令或输出错误的密钥不会构成攻击

2> malicious server: 可以在OT中只输入正确的codeword, 而无randomness, 此时无论C使用什么口令都可以完成key exchange协议. (包含pw.)

↳为了阻止此类攻击,需让server证明自己的行为(加密自己的协议view)



在此过程中, 协议传输的codeword由password加密,通过一个可以容忍口令错误的对称加密方法,称为fuzzy symmetric cipher. 需设计相对称的场景,可以隐藏口令,且允许评估输入的distance.