Note set

Note set主要作用用于数据的完整性检查。

其步骤为:

- 用户在上传数据阶段，根据位置集$\{P_i\}$将note set插入到密文中，然后发送给CSP存储。

- 用户在完整性检查阶段，根据位置集$\{P_i\}$提取出自己的note set，然后检查其是否符合函数f。

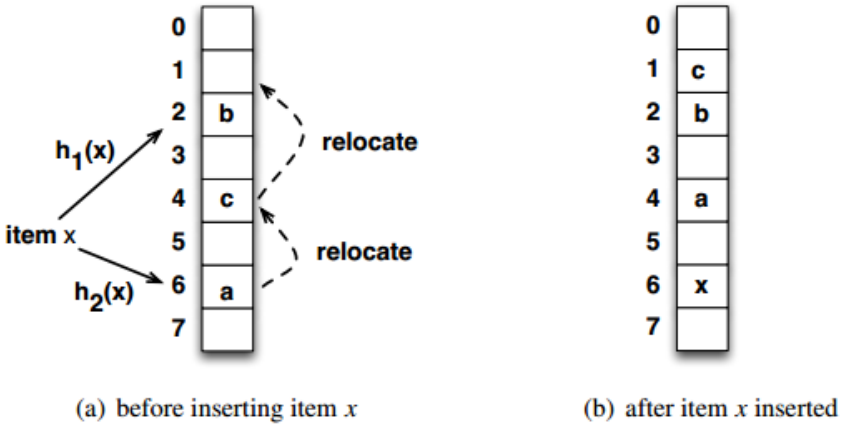
对比于在StealthGuard中，用户将这些看门狗插入文件F的随机选择位置，并将生成的文件存储在云中

Cuckoo filter CF

Cuckoo filter 主要作用用于数据的重复性检查。

其步骤为：

- 在系统设置阶段，CSP会将所有存储的数据对于的标签$\{x_j\}$进行哈希并加密$a_j = \mathrm{H}(x_j)^d mod\ N$插入到布谷鸟过滤器中
- 在重复性检查阶段，用户会通过布谷鸟过滤器的功能$CF.\mathrm{check}()$来检查自己的标签是否存在于过滤器中。



布谷鸟哈希表的插入

怎么防止AA的半可信

在文中AA主要工作在重复性检查时，$CT_{1,i} = E(B_{1,i}; DEK_1)$；$CK_1 = E(DEK_1; pk_{AA})$

顾虑：因为文中设定AA为semi-trust，认为AA可以用自己私钥sk_{AA}解密获得DEK_1，因此可以解密$CT_{1,i}$获得$B_{1,i}$。

但用户u_i在通信中传输的是$CTI_{1,i}$，由于AA不知道noteset和位置集$P_{1,i}$，所有无法将$CTI_{1,i}$转化为$CT_{1,i}$，因此可以防止AA窥探数据。

重复性检查

u_1

AA

CSP

$\{B_{1,i}\},\{y_{1,i}\}= H\left(H\left(B_{1,i}\right)\times P^*\right)$
P^*是系统实体之间共享的基点

$\{x_j\}$从以前的数据所有者中收集
$x_j = x_j \cup y_{1,i}$
假定CSP维护的标签已经大于DH

初始化RSA参数（e,d,N）广播e，N
计算$a_j = H(x_j)^d mod\ N$
$Sign\left(H\left(x_j\right)\right) = SIG_{SK_{c_j}}\left(H\left(x_j\right)\right)$
△为CSP持有的标签总数

校验:
1. △ =? N_s （N_s 为发送来的标签总数）
2. $H\left(x_j\right)$ =? $SIG^{-1}_{PK_{c_j}}(Sign\left(H\left(x_j\right)\right))$
3. $\prod H\left(x_j\right)$ =? $(\prod a_j)^d$

校验成功会假定CSP计算是正确的，
生成cuckoo filter CF，以a_j作为输入
For j = 1 to $N_s, CF = CF.Insert(a_j)$

$a_j, H\left(x_j\right), Sign\left(H\left(x_j\right)\right), \triangle$
$j = 1, \dots, N_s$

CF

生成随机数$r_{1,1}, \dots, r_{1,N_c}$ （N_c 为$\{y_{1,i}\}$ 的标签总数
计算$r_{1,i}^{inv} = r_{1,i}^{-1} mod\ N$
$r'_{1,i} = r_{1,i}^{e} mod\ N$
计算得For $i = 1\ to\ N_c$
$A[1,i] = H(y_{1,i}) \times r'_{1,i} mod\ N$

$A[1,i], i = 1, \dots, N_c$

For $i = 1\ to\ N_c$
$C[1,i] = (A[1,i])^d mod\ N$

$C[1,i], i = 1, \dots, N_c$

校验:
- $\prod A[1,i]$ =? $\prod C[1,i]^e$ 校验成功则证明了CSP计算的正确性

重复性检查:
IF $CF.\text{check}(C[1,i] \times r_{1,i}^{inv} mod\ N)$
$y_{1,i}$标签对应的块是重复的

$C[1,i] \times r_{1,i}^{inv} = (A[1,i])^d \times r_{1,i}^{-1} = (H(y_{1,i}) \times r_{1,i}^{e})^d \times r_{1,i}^{-1} = (H(y_{1,i}))^d$

Note set插入和数据上传

Data owner在上传是重复性检查时判断为不重复

u_1：Data owner

AA

CSP

$CT_{1,i} = E(B_{1,i}; DEK_1)$

$CK_1 = E(DEK_1; pk_{AA})$

生成符合隐藏函数f的note set和位置集$P_{1,i}$

根据$P_{1,i}$，将插入note set到$CT_{1,i}$中获得$CKI_{1,i}$

$CKI_{1,i}, CK_1, pk_i$

$\{y_{1,i}\}$:为了之后的重复性检查

$x_j = x_j \cup y_{1,i}$

$a_i = \mathrm{H}(y_{1,i})^{\mathrm{d}} mod\ N$

$Sign\left(H\left(y_{1,i}\right)\right) = SIG_{SK_{c_j}}\left(H\left(y_{1,i}\right)\right)$

校验:

1. $H\left(y_{1,i}\right) =? \ \mathrm{SIG}^{-1}_{\mathrm{PK}_{c_j}}(Sign\left(H\left(y_{1,i}\right)\right))$
2. $\prod H\left(y_{1,i}\right) =? \ (\prod a_i)^{\mathrm{d}}$

校验成功会假定CSP计算是正确的并通过$CF = CF.Insert(a_i)$更新cuckoo filter CF用于下一轮重复性检查

$a_i, H\left(y_{1,i}\right), Sign\left(H\left(y_{1,i}\right)\right)$

这里只将u_1：Data owner的标签$y_{1,i}$发送给了AA，而不是CSP维护的所有标签x_j

Note set插入和数据上传

u_2: Data Holder

AA

CSP

u_2在进行重复性检查发现是重复的后，通知CSP不做任何动作保留该块，CSP会检查的u_2拥有权，即是否$B_{2,i'} = B_{1,i}$

拥有权校验任务

拥有权校验协议：cryptoGPS identification scheme[34]

随机选择c

c

$h = H(B_{2,i'}) + (s_2 \times c)$

其中的s_2，V_2在本文中没有提到代表啥。

h, V_2

校验：$y_{1,i} =? H(hP^* + cV_2)$
若成立则证明$B_{2,i'} = B_{1,i}$

生成重加密密钥
$rk_{AA \to u_2} = RG(pk_{AA}; sk_{AA}; pk_2)$

$rk_{AA \to u_2}$

通过重加密将CK_1转换成CK_2
$R(rk_{AA \to u_2}; E(pk_{AA}; DEK_1)) = E(pk_2; DEK_1)$

完整性检查

u_1

CSP

初始化一个大数m和$b, gcb(b,m) = 1$作为一个秘密

根据位置集$P_{1,i}$，对每一列$(x_{1,i,0}, \dots x_{1,i,z})$选取一组随机数组$(d_{1,i,0}, \dots d_{1,i,z})$用来生成一组$(e_{1,i,0}, \dots e_{1,i,z})$

$if\ x \in P_{1,i}$，$e_{1,i,l} = N^l + d_{1,i,l}N^r$

$if\ x \notin P_{1,i}$，$e_{1,i,l} = d_{1,i,l}N^r$

计算$\{v_{1,i}\}, v_{1,i,l} = be_{1,i,l} mod\ m\ , l \in [1, \dots z]$

$Req_{1,i} = \{v_{1,i}, y_{1,i}\}$ →

计算$Resq_{1,i} = v_{1,i} \times B_{1,i}$

$Resq_{1,i}$ ←

计算$Res_{1,i} = Resq_{1,i} \times b^{-1} mod\ m$来获取查询的列，然后根据位置索引$P_{1,i}$挑出$noteset$检查其是否符合隐藏函数$f(n_{1,i,0}, \dots, n_{1,i,k}) = 0$。