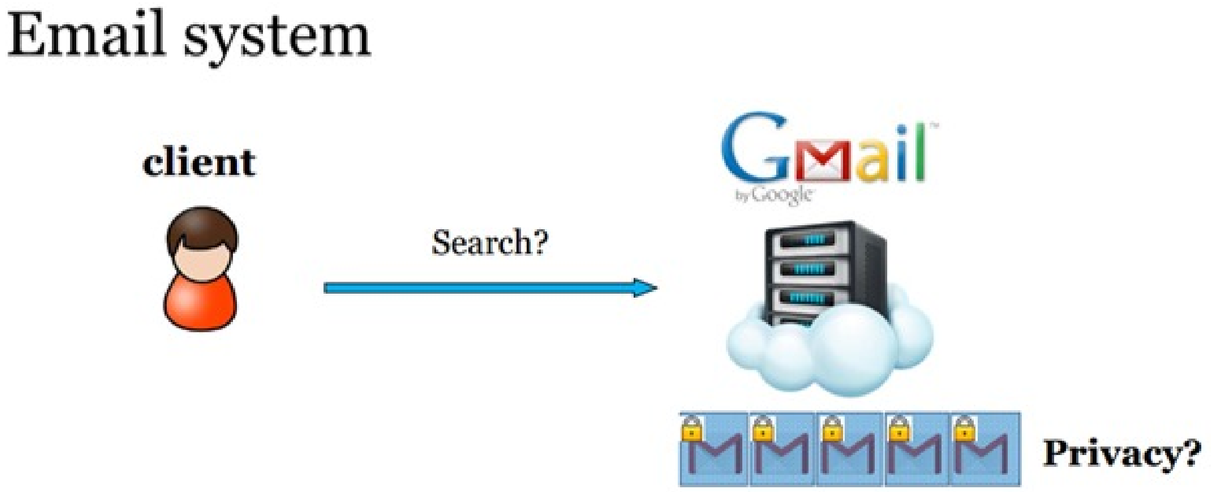
第六章 密文查询

目 录

1. 基本概念

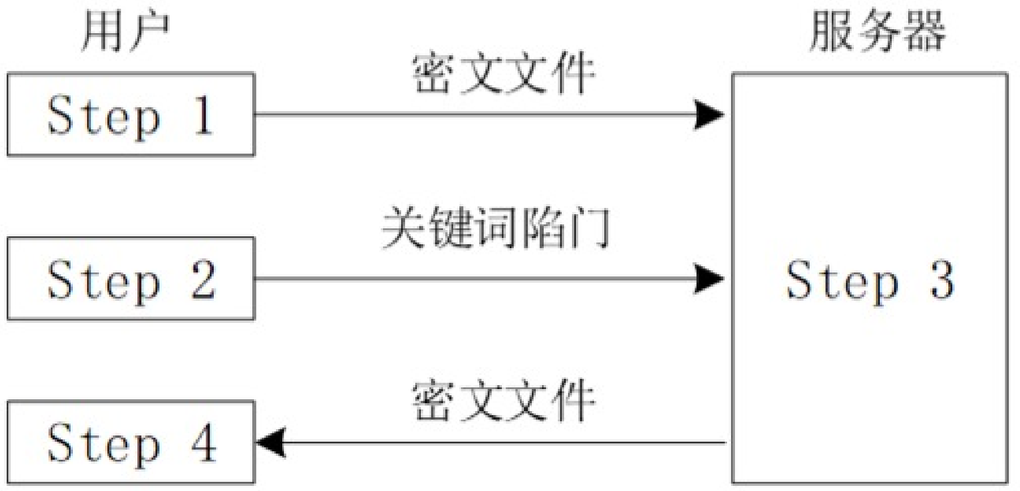
- **关键词检索是常见的操作**：比如数据库全文检索、邮件按关键词检索、在Windows系统里查找一个文件等。
- **可搜索加密**（Searchable Encryption，简称SE）则是一种密码原语，它允许数据加密后仍能对密文数据进行关键词检索，允许不可信服务器无需解密就可以完成是否包含某关键词的判断。



1. 基本概念

可搜索加密可分为4个子过程：

- **加密过程**：用户使用密钥在本地对明文文件进行加密并将其上传至服务器；
- **陷门生成过程**：具备检索能力的用户使用密钥生成待查询关键词的陷门，要求**陷门不能泄露关键词的任何信息**；
- **检索过程**：服务器以关键词陷门为输入，执行检索算法，返回所有包含该陷门对应关键词的密文文件，要求服务器除了能知道密文文件是否包含**某个**特定关键词外，无法获得更多信息；



- **解密过程**：用户使用密钥解密服务器返回的密文文件，获得查询结果。

1. 基本概念

可搜索加密可以分为对称可搜索加密和非对称可搜索加密

- **对称可搜索加密(Symmetric searchable encryption, SSE)**：加解密过程采用相同的密钥，陷门生成也需要密钥的参与，适用于**单用户模型**，具有计算开销小、算法简单、速度快的特点。
- **非对称可搜索加密(Asymmetric searchable encryption, ASE)**：公钥用于对明文信息加密和目标密文的检索，私钥用于解密密文信息和生成关键词陷门。非对称可搜索加密算法通常较为复杂，加解密速度较慢，其公私钥相互分离的特点，非常适用于私钥生成待检索关键词陷门，可用于**多用户同时进行关键词检索**的场景等。

1. 基本概念

动态可搜索加密（Dynamic Searchable Encryption, DSE）则指用户可以动态地对存储在服务器上的密文文档进行更新(update)。

更新操作包括两种：**添加(add)**和**删除(delete)**。

目 录

1. 形式化定义

定义在字典$\Delta=\{W_1, W_2, \cdots, W_d\}$上的对称可搜索加密算法可描述为五元组：

$$SSE = (KeyGen, Encrypt, Trapdoor, Search, Decrypt)$$，其中，

- $\boldsymbol{K = KeyGen(\lambda)}$：输入安全参数$\lambda$，输出随机产生的密钥$K$；
- $\boldsymbol{(I, C) = Encrypt(K, D)}$：输入对称密钥$K$和明文文件集$D = (D_1, D_2, \cdots, D_n)$，输出索引$I$和密文文件集$C = (C_1, C_2, \cdots, C_n)$。对于无需构造索引的SSE方案，$I = \emptyset$；
- $\boldsymbol{T_W = Trapdoor(K, W)}$：输入对称密钥$K$和关键词$W$，输出关键词陷门$T_W$；
- $\boldsymbol{D(W) = Search(I, T_W)}$：输入索引$I$和陷门$T_W$，输出包含$W$文件的标识符集合$D(W)$；
- $\boldsymbol{D_i = Decrypt(K, C_i)}$：输入对称密钥$K$和密文文件$C_i$，输出相应明文文件$D_i$。

1. 基本定义

- **正确性**

如果对称可搜索加密方案SSE是正确的，

那么对于$KeyGen(\lambda)$和$Encrypt(K, D)$输出的K和(I, C)，都有

$$\boldsymbol{Search\left(I, Trapdoor(K, W)\right) = D(W)}$$

和$\boldsymbol{Decrypt(K, C_i) = D_i}$成立，

这里$C_i \in C, i = 1,2, \cdots, n.$

2.基于正向索引的构造

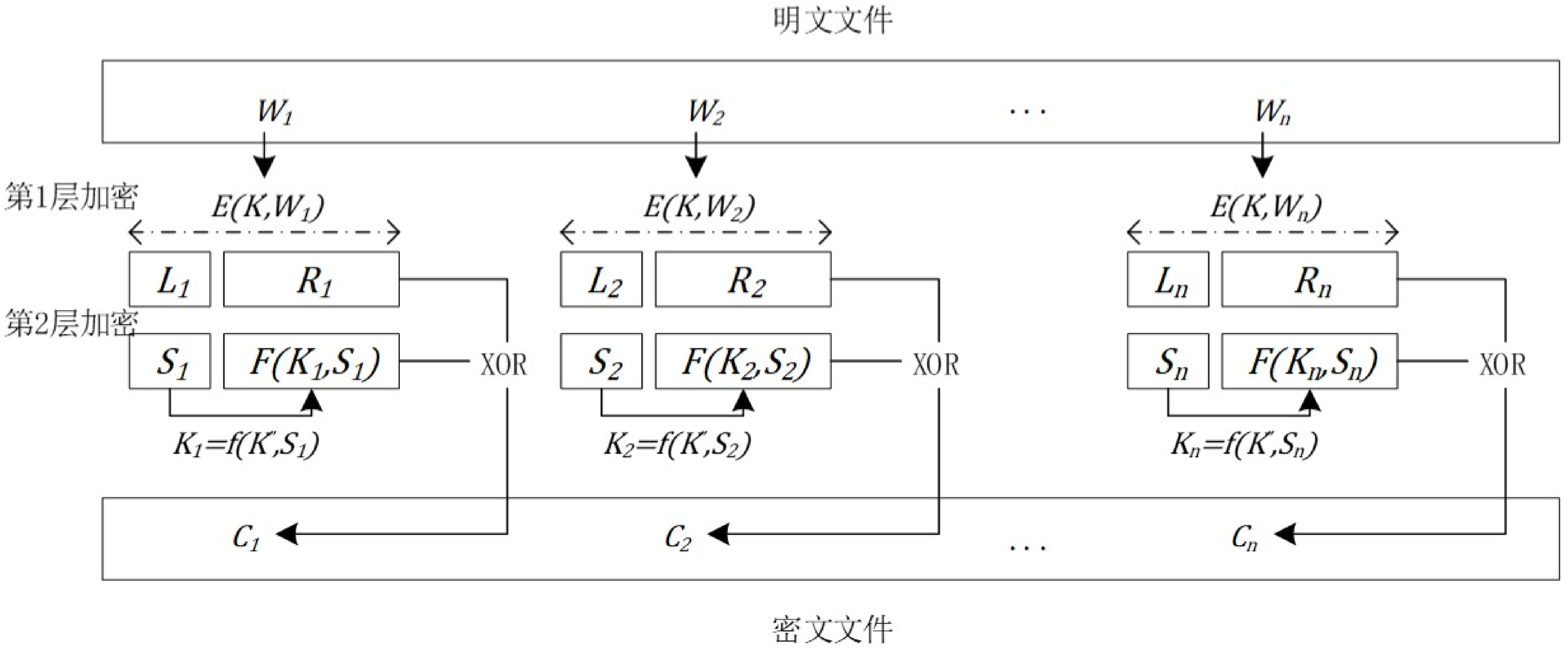
基本构造思路是:

- 将文件进行**分词**；**对每个关键词进行加密处理**；
- 在搜索的时候，提交**密文关键词**或者**可以匹配密文关键词的中间项**作为陷门，进而得到一个包含待查找的关键词的密文文件。

因为是按照“文档标识ID：关键词1，关键词2，关键词3，…，关键词n”的方式组织文档与关键词的关系，我们可以称这种方式为正向索引（和后面倒排索引进行区分）。2000年，Dawn Song所提出的SWP方案，就采用了这种方式。

2.基于正向索引的构造

典型构造：SWP方案



2.基于正向索引的构造

优点：SWP方案通过植入“单词”位置信息，能够支持**受控检索**（检索关键词的同时，识别其在文件中出现的位置）。例如，将所有“单词”以W||α形式表示，α为W在文件中出现的位置，但查询时可增加对关键词出现位置的约束。

SWP方案存在一些缺陷：

- **效率较低**，单个单词的查询需要扫描整个文件，占用大量服务器计算资源；
- **在安全性方面存在统计攻击的威胁。**例如，攻击者可通过统计关键词在文件中出现的次数来猜测该关键词是否为某些常用词汇。

3.基于倒排索引的构造

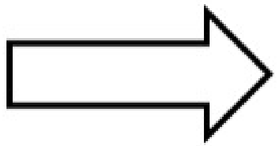
- 对称可搜索加密多数基于索引结构来提升检索的效率，比如倒排索引。
- **倒排索引（英语：Inverted index），**也常被称为反向索引、置入档案或反向档案，是一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。它是文档检索系统中最常用的数据结构。通过倒排索引，可以根据单词快速获取包含这个单词的文档列表。如图所示，关键词w_1索引了一个列表，列表中存储了所有关联的文档的标识ID。

Key	Value
w_1	$D_1, D_4, \ldots, D_{15}$
w_2	$D_1, D_3, \ldots, D_{22}$
$\vdots$	$\vdots$
w_m	$D_2, D_3, \ldots, D_n$

3.基于倒排索引的构造

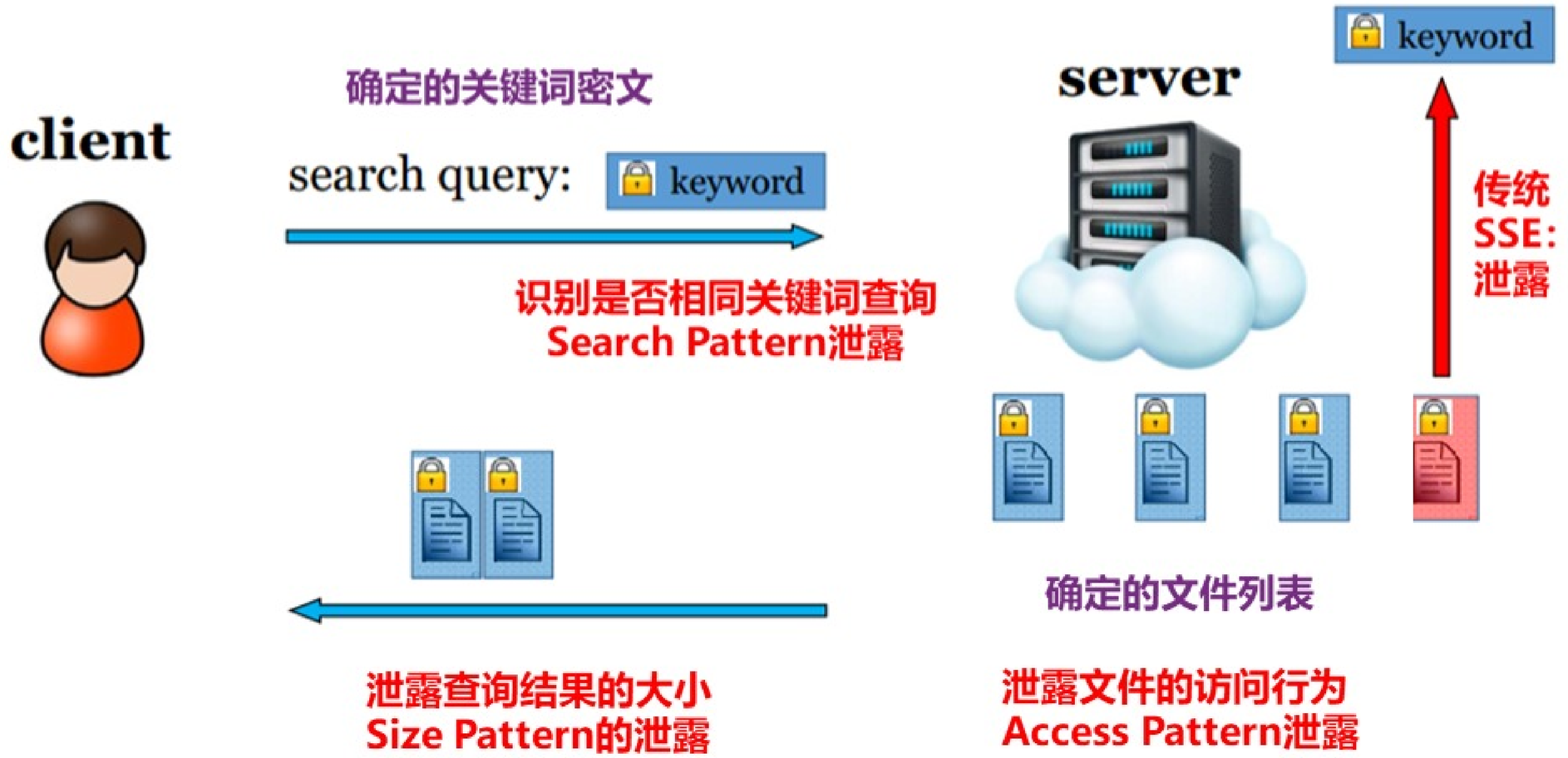
- 一种基于倒排索引的简单构造方法为：**将关键词加密，提交密文关键词或者可以匹配密文关键词的中间项作为陷门**，进而快速检索到匹配的密文文件列表，获得相应的文件标识ID。

Key	Value
w_1	$D_1, D_4, \ldots, D_{15}$
w_2	$D_1, D_3, \ldots, D_{22}$
⋮	⋮
w_m	$D_2, D_3, \ldots, D_n$



Key	Value
$E(w_1)$	$D_1, D_4, \ldots, D_{15}$
$E(w_2)$	$D_1, D_3, \ldots, D_{22}$
⋮	⋮
$E(w_m)$	$D_2, D_3, \ldots, D_n$

4. 模式泄露


client
确定的关键词密文
search query:
keyword
识别是否相同关键词查询
Search Pattern泄露
server
keyword
传统
SSE:
泄露
确定的文件列表
泄露查询结果的大小
Size Pattern的泄露
泄露文件的访问行为
Access Pattern泄露

4. 模式泄露

三种信息的泄露：

- **查询模式（Search Pattern）泄露**：泄露是否是相同关键词的查询。
- **访问模式（Access Pattern）泄露**：泄露了文件的访问行为，进而可以鉴别是否是相同关键词的查询、相同文件被访问了多少次等。
- **大小模式（Size Pattern）泄露**：泄露了查询的结果大小、更新操作影响的文件多少或者文件包含的关键词个数等。

单选题 1分

考虑到性能，对称可搜索加密主要基于如下哪种索引结构来构造

A 正向索引

B 倒排索引

C 布隆过滤器

D 分词索引

目 录

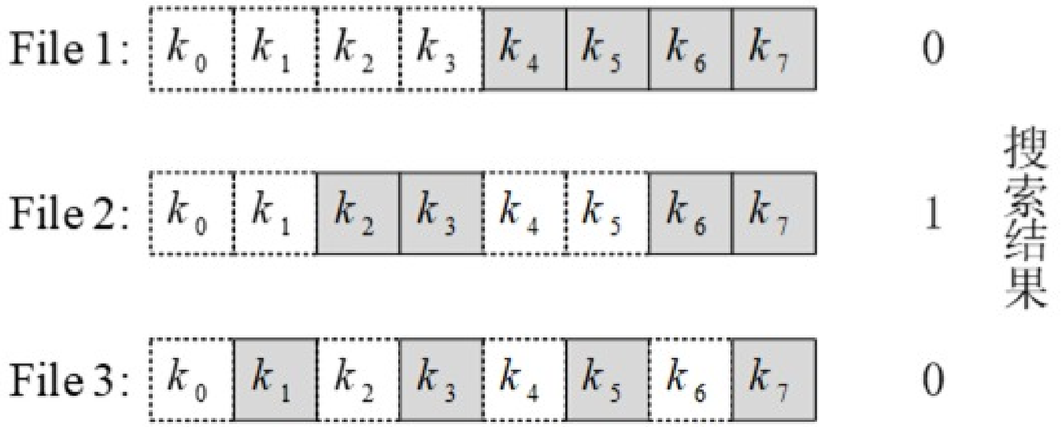
1. 文件注入攻击

查询模式的泄露可以引发相应攻击，通过注入一定数量的文件，关键文件插入到哪个关键词所关联的文件列表，就可以实现加密关键词的破解，这就是**文件注入攻击**。

典型的攻击场景：在邮件应用中，假定邮件服务器是好奇的，它试图猜测Alice的加密邮件的内容。邮件服务器可以伪造加密邮件（用Alice公钥加密邮件内容）给Alice。由于Alice希望将邮件存储到邮件服务器，因此，她会将邮件分词后基于倒排索引的对称可搜索加密机制，加密后发送到服务器端。也就意味着邮件服务器对Alice进行了文件注入攻击，试图猜测每个加密的关键词是什么。

1. 文件注入攻击

查询模式的泄露可以引发相应攻击，通过注入一定数量的文件，关键文件插入到哪个关键词所关联的文件列表，就可以实现加密关键词的破解，这就是**文件注入攻击**。



- 例子中$|K| = 8$, 首先让服务器生成一组要注入的$\log |K|$（即3个）个文件F，每个文件包含K/2的关键词:
  - 如果文件File1插入到了某关键词对应的文件列表中，意味着K4
  - 如果文件File2和File3插入到了某关键词对应的文件列表中，意味着K3
  - 如果文件File2插入到了某关键词对应的文件列表中，意味着K2
  - ……

1. 文件注入攻击

对于这种攻击，文件是非自适应生成的，与陷门无关。

一旦这些文件被注入，服务器可以恢复与客户端发送的任何未来陷门对应的关键字。

这种攻击所需的注入文件数量是相当合理的；

有了10000个关键字，以加密邮件系统为例，每天只向客户端发送一封电子邮件的恶意邮件服务器可以在短短2周内注入必要的文件。

1. 文件注入攻击

- **为进一步降低每次注入的文件的长度，可以采用层次搜索攻击。**
- 这种攻击首先**将关键字集划分为$\lceil |K|/T \rceil$个子集，每个子集包含T个关键字。**服务器注入包含每个子集中的关键字的文件，以了解客户端关键字位于哪个子集。此外，可以对这些子集的相邻对进行二进制搜索攻击，以准确地确定关键字，总计需要注入$\lceil |K|/2T \rceil \cdot (\lceil \log 2T \rceil + 1) - 1$个文件。

- **当关键字集合的大小为$|K| = 5000$，阈值为$T = 200$时，服务器只需要注入131个文件，且注入文件的数量随关键字集合的大小呈线性增长。**
- 我们再次强调，相同的注入文件可以用来恢复对应于任意数量的陷门的关键字；也就是说，一旦这些文件被注入，服务器就可以恢复客户端的任何未来搜索的关键字。

2. 前向安全

- **前向安全意味着更新操作不会泄露任何关于已更新关键字的信息。特别是，服务器无法得知更新的文档与之前查询的关键字匹配关系，可以保护更新操作的查询模式。**

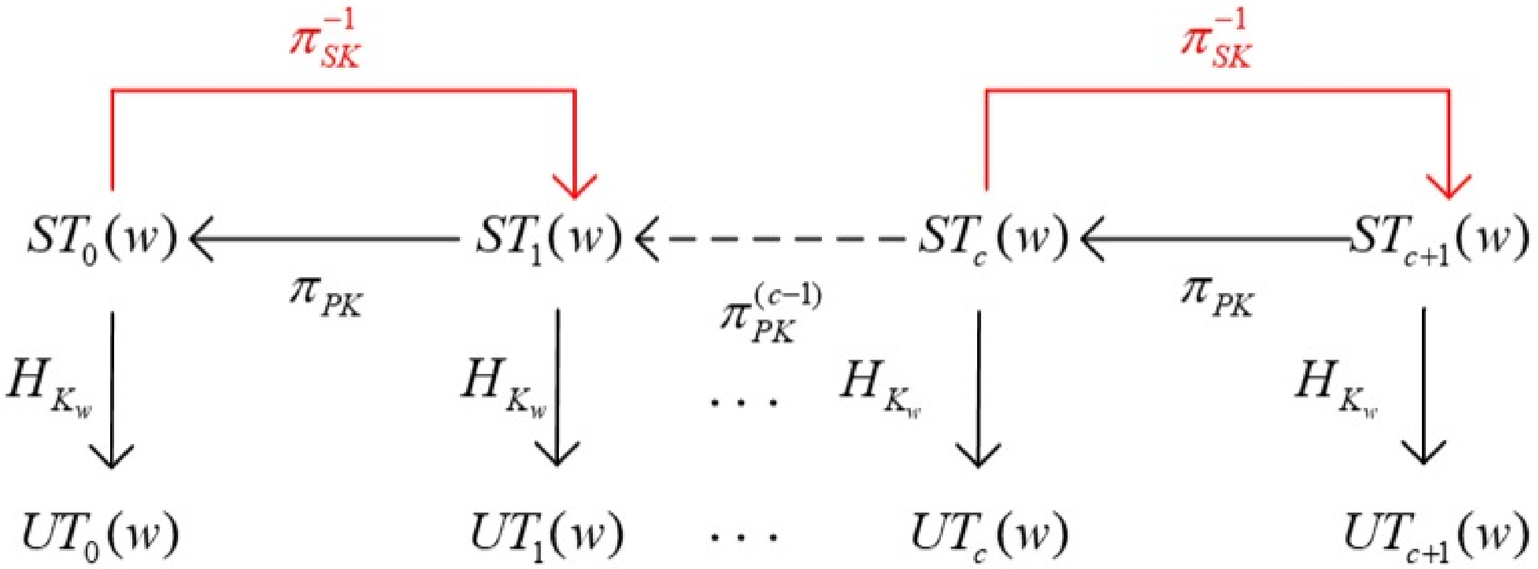
  更正式地说，我们给出前向安全的定义：

  定义（前向安全）我们说一个自适应安全的可搜索加密方案是前向安全的，当且仅当其更新操作的泄露函数可以写成

$$\mathcal{L}^{Updt}(op, in) = \mathcal{L}'(op, \{(ind_i, \mu_i)\})$$

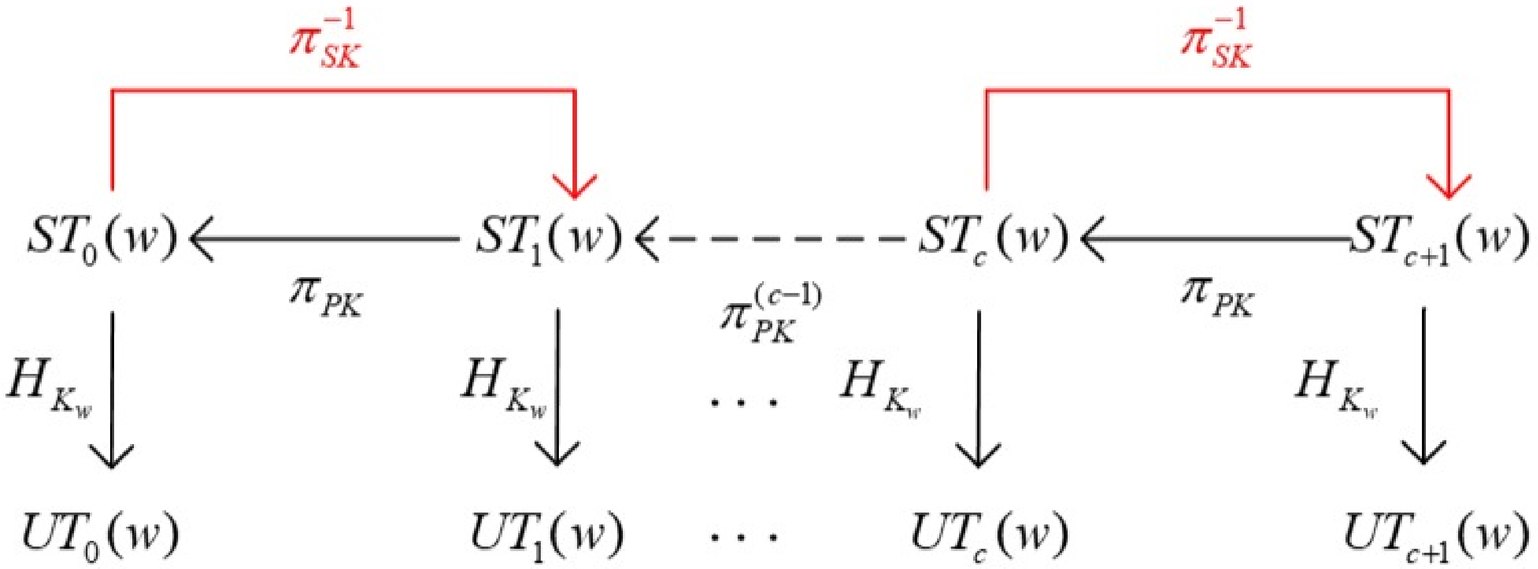
- 其中，$\{(ind_i, \mu_i)\}$是更新的文档与其包含的关键词的合集，ind_i文档更新了μ_i个关键词。

3. 基本构造



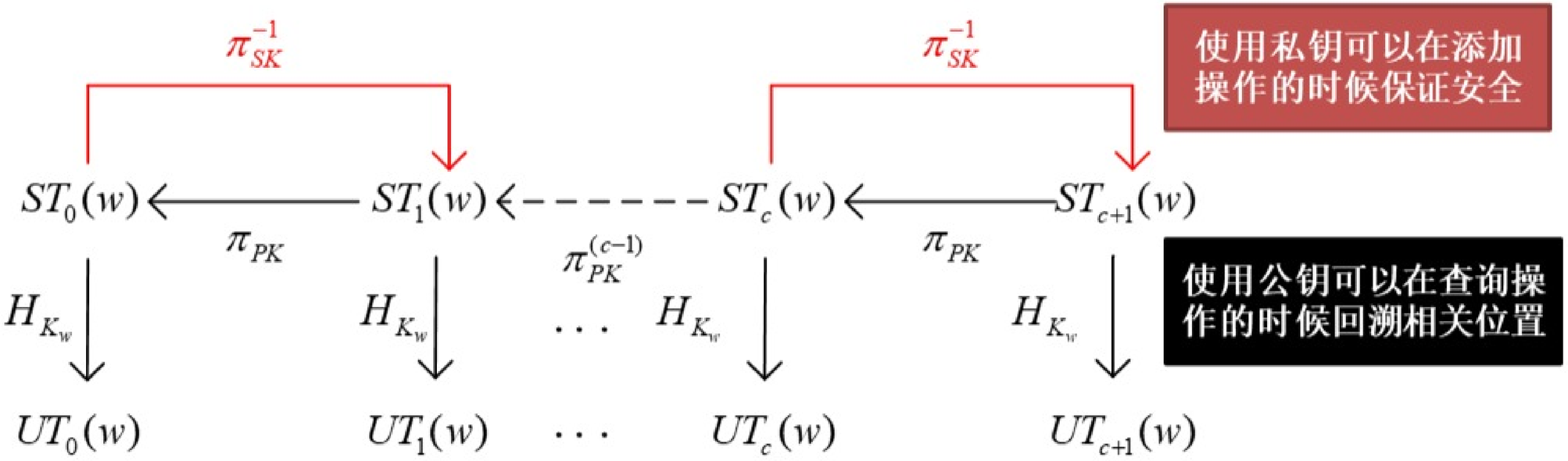
- 2016年，第一个前向安全的可搜索加密方案提出，即EOYO方案。该方案中，每个关键字对应一个匹配文档的索引列表$(ind_0, ind_1, \cdots, ind_{n_w})$。然后，**该列表中的每个元素$ind_c$都被加密并存储在派生自$w$和$c$的（逻辑）位置，我们称此位置为$UT_c(w)$。**

3. 基本构造



- **添加：** 当客户端想要添加一个匹配w的文档时，他**计算一个新的位置**$\boldsymbol{UT_{n_w+1}(w)}$，将**文档索引**$\boldsymbol{ind_{c+1}}$**加密为**$\boldsymbol{e}$，并发送$(\boldsymbol{UT_{n_w+1}(w)}, \boldsymbol{e})$到服务器。
- **查询：** 当客户端对w执行搜索查询时，他将**发出一个搜索陷门**$\boldsymbol{ST_{c+1}(w)}$，允许服务器重新计算进而得到所有之前插入的包含w的索引块的位置。

3. 基本构造



- **陷门置换**是指一个置换函数$\pi(x) \to y$，$\pi(x)$满足单向性，即通过y很难反算出x；但是如果谁拥有陷门t，就能实现逆计算$\pi_t^{-1}(y) \to x$。
- **公钥密码体制就是一个典型的陷门置换，这里的陷门t就是私钥。**
- **使用公钥加密就是一个单向置换的过程，使用私钥解密就是求逆的过程。**

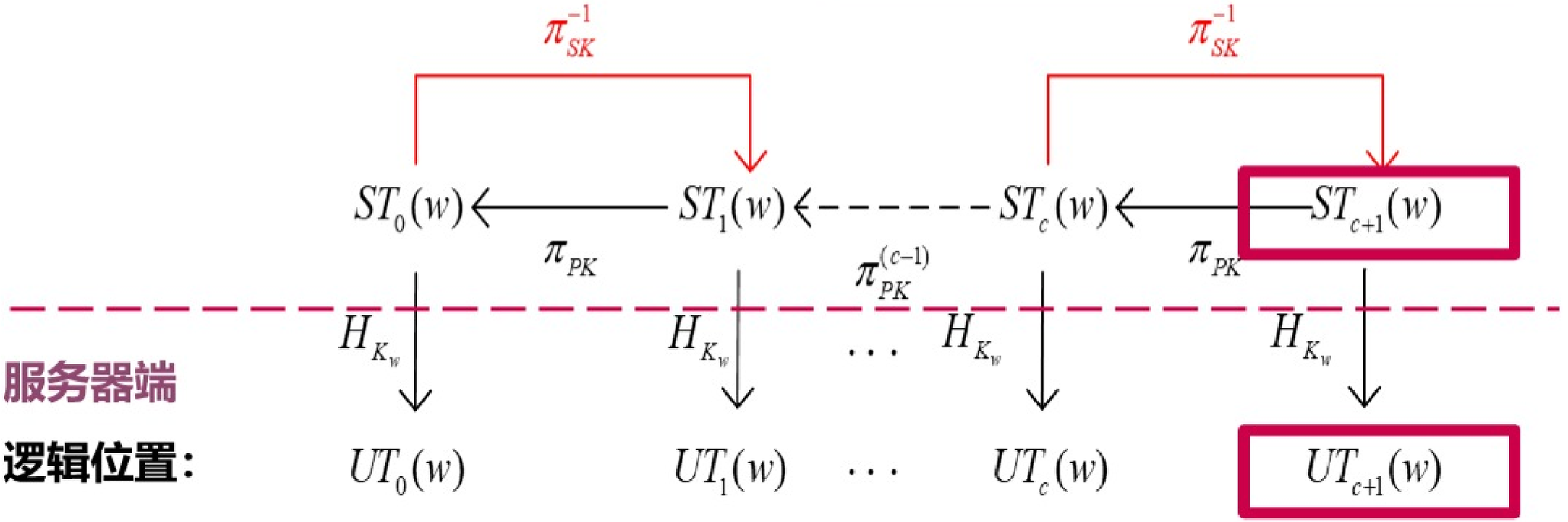
3. 基本构造

私钥解密

客户端

拥有状态信息表W[w]： c

添加： $ST_{c+1}(w) = \pi^{-1}(ST_c(w))$



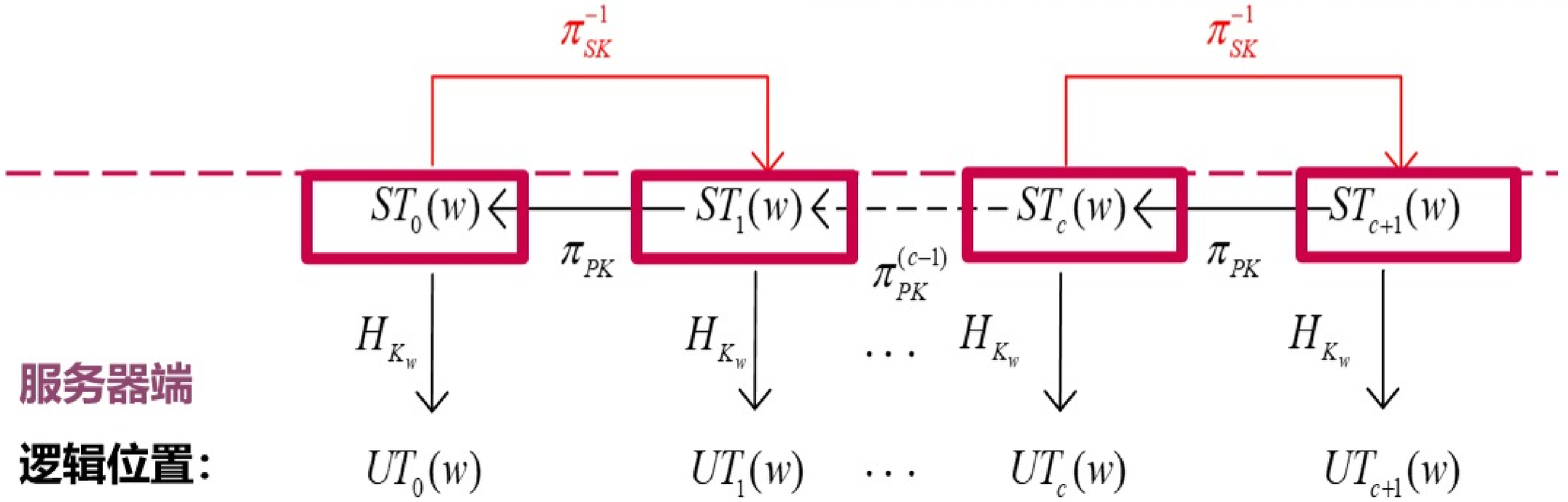
3. 基本构造

公钥加密

客户端

拥有状态信息表W[w]： c

查询： $ST_c(w) = \pi(ST_{c+1}(w))$



服务器端

逻辑位置：

4. 缺点

基于公钥密码体制构建，性能低：

- 客户端产生陷门，将执行c次私钥解密
- 服务器端遍历需要执行c次公钥加密

5. 应用方式

具有前向查询安全的对称可搜索加密在密态数据库中具有广泛的应用前景：

- **全文检索 & 模糊查询**
  - 直接适用数据库大文本的全文检索 & **短文本的模糊查询问题**（关键词组合）
- **[语义安全]的相等查询和连接查询**
  - 语义安全要求相同明文多个密文，**前向安全性与语义安全要求相匹配**
- **范围查询**
  - 通过将**待查询的范围转化为范围内的多个关键词**的查询，可以实现范围查询

单选题 1分

文件注入攻击利用了可搜索加密的哪种模式泄露？

A 查询模式

B 大小模式

C 访问行为模式

D 以上都不是

目 录

1. 基本定义

定义（**非对称可搜索加密**，public key encryption with keyword search, PEKS）非对称密码体制下可搜索加密算法可描述为$PEKS = (KeyGen, Encrypt, Trapdoor, Test)$：

- $(pk, sk) = \boldsymbol{KeyGen}(\lambda)$：输入安全参数$\lambda$，输出公钥$pk$和私钥$sk$；
- $C_W = \boldsymbol{Encrypt}(pk, W)$：输入公钥$pk$和关键词$W$，输出关键词密文$C_W$；
- $T_W = \boldsymbol{Trapdoor}(sk, W)$：输入私钥$sk$和关键词$W$，输出陷门$T_W$；
- $b = \boldsymbol{Test}(pk, C_W, T_{W'})$：输入公钥$pk$、陷门$T_{W'}$和关键词密文$C_W$，根据$W$与$W'$的匹配结果，输出判定值$b \in \{0,1\}$。

2. 典型应用

不可信赖邮件服务器路由问题的解决思路：

用户Alice掌握着私钥，并将相对应的公钥公开，**为了让电子邮件网关分拣接收到的邮件，Alice会事先将一些特定关键字的陷门T_W发送给电子邮件网关，使得它能够通过判断邮件中是否包含关键字W来选择接受设备。**与此同时，电子邮件网关在判断的过程中无法获得关于关键字和邮件内容的有效信息。

2. 典型应用

不可信赖邮件服务器路由问题的解决思路：

- Bob使用Alice的公钥pk加密邮件和关键词，并将形如$(PKE.Encrypt(pk, MSG),$ $PEKS.Encrypt(pk, W_1), \cdots, PEKS.Encrypt(pk, W_n))$的密文发送至邮件服务器。这里，$PKE.Encrypt$为公钥密码加密算法，$MSG$为邮件内容，$W_1, \cdots, W_n$为与$MSG$关联的关键词。
- Alice将$T_{\text{“}urgent\text{”}}$或$T_{\text{“}lunch\text{”}}$长驻服务器。
- 新邮件到来时，服务器自动对其关联的关键词执行与$T_{\text{"}urgent\text{"}}$或$T_{\text{"}lunch\text{"}}$相关的$Test$算法，如果输出1，便将该邮件转发至Alice的手机或个人电脑。

3. 典型构造

2004年，Boneh提出了第一个非对称的可搜索加密方案，具体构造如下：

令$e: G_1 \times G_1 \to G_2$为双线性对，函数$H_1: \{0,1\}^* \to G_1$和$H_2: G_2 \to \{0,1\}^{\log p}$为哈希函数。

- **Keygen**：输入安全参数，该安全参数决定群G_1和G_2的阶p，随机挑选$\alpha \leftarrow Z_p^*$和G_1的生成元g，输出$pk := (g, h = g^{\alpha})$和$sk := \alpha$。
- **Encrypt**：输入关键词，随机选$r \leftarrow Z_p^*$，计算$t := e(H_1(w), h^r)$，输出$c := (g^r, H_2(t))$。
- **TrapDoor**：输入私钥和关键词，输出$td := H_1(w)^{\alpha}$。
- **Test**：输入陷门和密文$c = (c_1, c_2)$，若$H_2(e(td, c_1)) = c_2$，输出1，否则输出0。

正确性：

$$\mathrm{e}(td, \mathrm{c}_1) = \mathrm{e}(\mathrm{H}_1(\mathrm{w})^{\alpha}, \mathrm{g}^{\mathrm{r}}) = \mathrm{e}(\mathrm{H}_1(\mathrm{w}), \mathrm{g}^{\alpha \mathrm{r}}) = \mathrm{e}(\mathrm{H}_1(\mathrm{w}), \mathrm{h}^{\mathrm{r}}),$$

$$\mathrm{H}_2(\mathrm{e}(td, c_1)) = c_2$$

4. 关键词猜测攻击

PEKS本身定义存在严重的安全隐患：关键词猜测攻击。

关键词猜测攻击是由于**关键词空间远小于密钥空间**，而且用户通常使用常用关键词进行检索，这就给攻击者提供了只需采用字典攻击就能达到目的的“捷径”。

导致关键词猜测攻击的原因可归结为：

①关键词空间较小，且用户集中于使用常用词汇，给攻击者提供了遍历关键词空间的可能；

②PEKS算法一致性约束，使攻击者拥有对本次攻击是否成功的预先判定：执行*Test*算法，返回1说明本次攻击成功。

4. 关键词猜测攻击

对策：为抵御关键词猜测攻击，很多方案提出。

- 比如可以在服务器端进行模糊陷门测试，过滤大部分不相关邮件，最后在本地精确匹配，得到检索结果。
- 这种方法通过引入模糊陷门，一定程度降低了接收者外部PEKS算法一致性，使其能够抵御关键词猜测攻击，但增加了客户服务器通信量和用户端计算量。

谢　谢