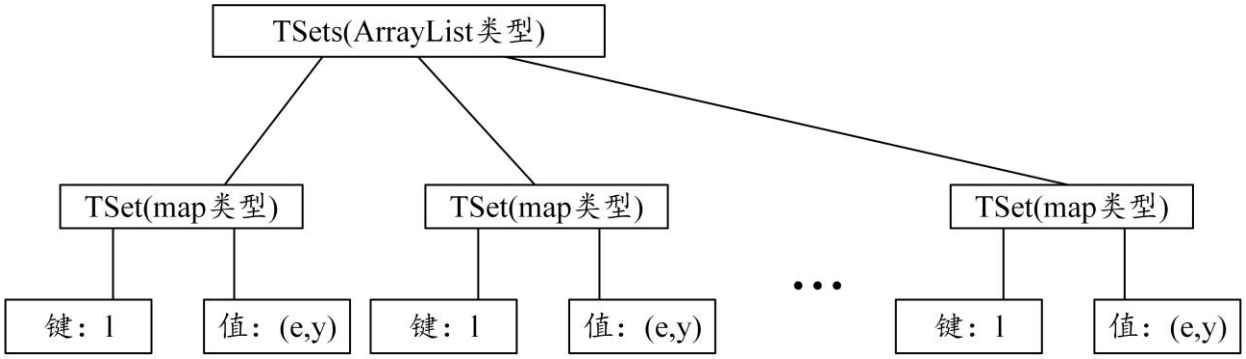
西安电子科技大学
XIDIAN UNIVERSITY

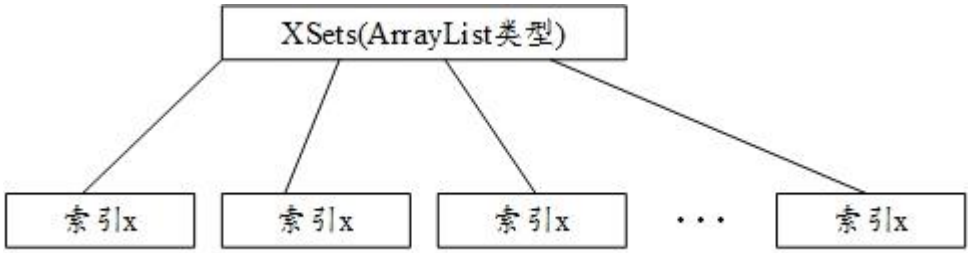
Sun方案实现架构

——汇报人：张中俊

Sun方案的总览



图：TSets的结构

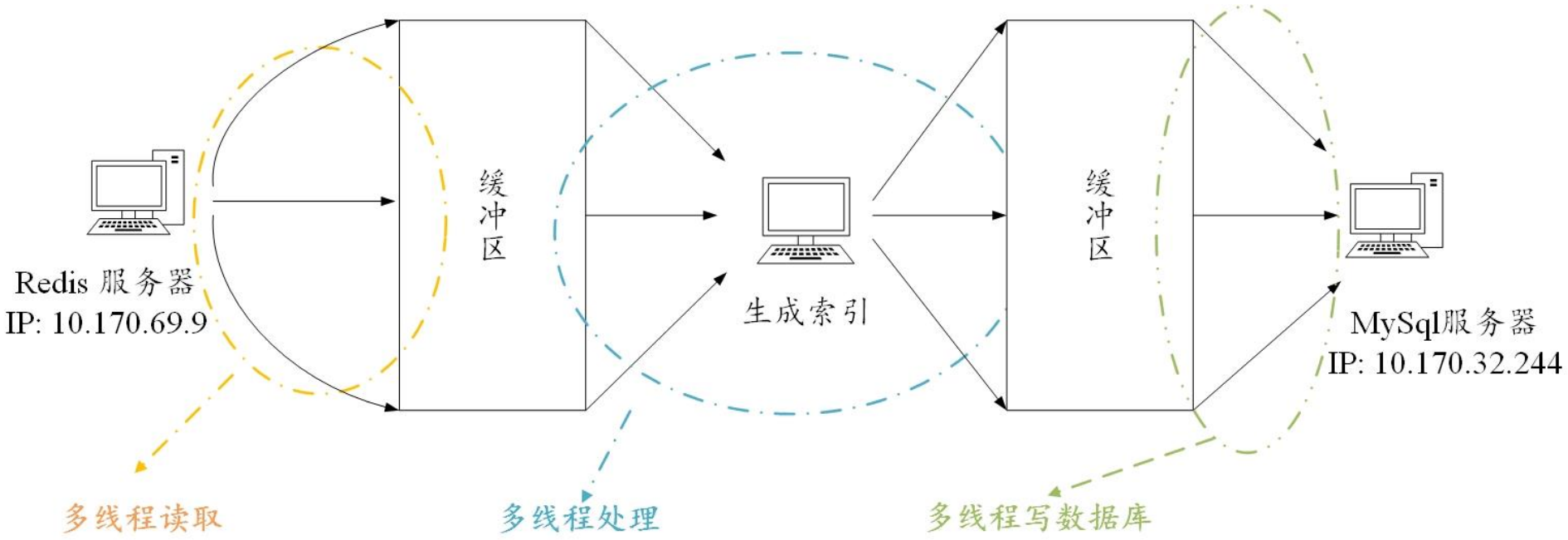


图：XSets的结构

- 键l：$F(F(K_s, g_1^{w^{-1}}), c)$
- 索引y：$F_p(K_I, ind) \cdot \left(F_p(K_Z, g_2^{w^{-1}} || \mathrm{c})\right)^{-1}$
- 索引x：$g^{F_P\left(K_X, g_3^{w^{-1}}\right) \cdot F_p(K_I, ind)}$
- 搜索的token：$xtoken[c,i] = g^{F_p\left(K_Z, g_2^{w_1^{-1}} || \mathrm{c}\right) \cdot F_p\left(K_X, g_3^{w_i^{-1}}\right)}$
- 搜索：$xtoken[c,i]^y \in XSets$

如何存储原始文件

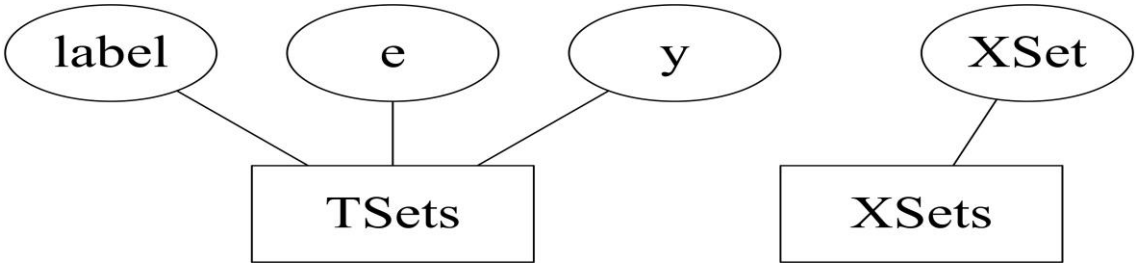
- 使用redis数据库存储，原因如下：
  1. no-sql类型的数据库，完美契合关键词-文件名的数据形式
  2. 内存型数据库，读取速度快



图：整体架构

如何存储TSet和XSet

- 存储在MySql数据库中
- 存储类型主要考虑在搜索时候各个字段的行为：
  - 在搜索中，XSets中主要是做匹配，存储为字符串类型即可，长度为311，分配400的长度
  - 在搜索中，TSets中的label也是为了做匹配，存储为字符串类型即可，测试下长度为70多，最多情况16 * 4 + 16 + 16 + 2 = 98，分配120的长度
  - 在搜索中，需要密文e 和 索引y，我们将其存储为blob类型，之后再反序列化为java对象即可



图：数据库的E-R图

如何存储Master Key

- 本算法设计到的Master Key包括：
  - KI：用于生成索引y
  - Kz：用于生成索引y
  - Kx：用于生成索引x
  - Ks：用于生成加密文件名的密钥Ke
  - g：双线性对的生成元
  - g1：Zr域中的随机元素
  - g2：Zr域中的随机元素
  - g3：Zr域中的随机元素
- 把他们序列化到txt文本文件中，用到的时候再反序列化出来。

- pairing不必存储，因为我们使用的pairing是固定的

如何存储ABE相关的信息

1. 密钥：包括PublicKey，PrivateKey， MasterKey：事先生成好，序列化到文本文件中
2. attribute list：只有加密时候使用，直接调用，不进行存储
3. policy：只有解密时候使用，直接调用，不进行存储

搜索的详细步骤——客户端

1. 首先做出预测，含有关键词w1的文件不超过100个(实际做实验的时候是：事先记录好某1000个关键词对应的文件名个数，然后直接调用)
2. 接着生成xtoken矩阵，其中kws是所有搜索的关键词
3. 生成w1对应的stag
4. 提交stag 和 xtoken到服务器

	i = 1	i = 2	……	i = |kws|
c=0				
c=1				
……				
c=99				

搜索的详细步骤——服务器

- 令$c=1$，根据stag和c生成l，根据l找到(e,y)，这里的$y=F_p(K_I,ind)\cdot\left(F_P\left(K_Z,g_2^{w_1^{-1}}||\mathrm{c}\right)\right)^{-1}$
- 如果对于xtoken第c行的所有的元素xtoken[c][i]，都有$xtoken[c][i]^y\in XSet$，则将e加入到答案中
- 将c加1，循环直到$TSet$不含有生成的l

- 索引x：$g^{F_P\left(K_X,g_3^{w^{-1}}\right)\cdot F_p(K_I,ind)}$
- $xtoken[c,i]=g^{F_p\left(K_Z,g_2^{w_1^{-1}}||\mathrm{c}\right)\cdot F_p\left(K_X,g_3^{w_i^{-1}}\right)}$