**哈尔滨工业大学**

**硕士学位论文中期报告**

**题 目：基于安全多方计算下的时间序列异常检测**

**院 （系） 计算机科学与技术学院**

**学 科 计算机技术**

**导 师 张春慨**

**研 究 生 刘浩东**

**学 号 15S151525**

**中期报告日期 2017年3月16 日**

目 录

[1 课题主要研究内容及进度 1](#_Toc477388973)

[1.1 课题主要研究内容 1](#_Toc477388974)

[1.2 进度介绍 2](#_Toc477388975)

[2 目前已完成的主要研究工作及结果 2](#_Toc477388976)

[2.1 安全多方计算的实现 2](#_Toc477388977)

[2.1.1 同态加密算法以及BCP加密算法简述 2](#_Toc477388978)

[2.1.2 系统模型 4](#_Toc477388979)

[2.1.3 安全转换协议 5](#_Toc477388980)

[2.1.4 安全计算协议 6](#_Toc477388981)

[2.2 异常检测在安全多方下的实现 8](#_Toc477388982)

[2.3 实验验证 11](#_Toc477388983)

[3 后期拟完成的研究工作及进度安排 14](#_Toc477388984)

[4 存在问题及解决方案 15](#_Toc477388985)

[4.1 存在问题与困难 15](#_Toc477388986)

[4.2 解决方案 15](#_Toc477388987)

[5 如期完成全部论文工作的可能性 15](#_Toc477388988)

# 课题主要研究内容及进度

## 1.1 课题主要研究内容

在实际生活中，各个领域都包含大量的时间序列数据，例如病人的心电图数据、脑电图数据、发电厂大量传感器的参数数据以及网络流数据等等。而时间序列的异常子序列（模式）检测是一个十分重要的领域，含异常模式的时间序列大部分数据表现为正常形态，异常模式出现频率极少，但极少出现的异常模式却包含相当重要的信息。异常的心电数据意味着病人可能患有某种类型的心脏疾病，异常的脑电数据可能是由癫痫等脑科疾病引起，及时发现病人的异常心电或者脑电数据可以对后续的治疗起到指导作用；而工厂传感器数据异常可能意味着系统某个部分出现了故障，及时发现异常并对系统故障进行修护能减少损失。因此，时间序列数据的异常子序列检测研究极具现实意义。

在目前所有研究工作中都是认为时间序列数据是集中的，所采取的算法也都是认为当前的数据集是完备的。而在实际应用中，数据分布往往是分散的，例如患者的心电数据，可能在某一个地区的A、B、C三家医院都保存某一个患者的心电数据。A医院存有患者体检时的两分钟数据，B医院有患者测量的三小时数据，C医院有患者住院的两周心电数据，因此当A医院想要分析患者的心电数据的时候，通过两分钟的心电数据往往无法得出正确的结论，例如这两分钟都是运动后的心电数据，那对正常的数据来说这两分钟的数据都是异常的。因此我们想要得到准确的分析结果，就需要将A、B、C、三家的数据整合在一起形成一个完备的数据集，在完备的数据及上进行数据挖掘得出的结论我们认为是可信的。

但同时我们应该注意到，数据作为一个重要的资源，对于拥有者来说往往是很具有价值的，尤其是医院患者的信息，是一种典型的高度隐私数据。我们需要A、B、C 三家的数据来做数据挖掘，但是A、B、C中的任何一个都不希望另外两家知道自己的隐私数据，在这样的一个条件下，我们依然想得到从完备数据集中分析得出的结果，因此这就涉及到一个安全多方计算的问题。本课题将时间序列异常的检测扩展到适用于多方参与的情况下，并设计了安全检测算法检测时间序列异常，我们采取的安全加密算法允许输入的数据在不同的公钥加密，并能对不同公钥加密的数据进行运算。

本课题将从以下几个方面开展研究：

1. 安全多方计算框架的实现

多方参与下的时间序列异常检测需存在的一个问题就是安全性的问题，参与方都不想让其他方知道自己的数据，可是有依赖于其他方的数据才能完成数据挖掘任务。因此将自己的数据加密后再参与运算是必须的步骤，因此加密算法的选取对本课题的研究有着十分重要的意义。安全多方计算（SMC）是解决一组互不信任的参与方之间保护隐私的协同计算问题，参与方都不想让其他方知道自己的数据，可是又依赖于其他方的数据才能完成数据挖掘或者计算任务。

1. 异常检测在安全多方下的实现

异常，就是那些和数据集中大部分模式看上去不一致或者其产生方式极大地违背了大部分模式的生成机制的样本（模式）。而异常检测，就是基于一定的方法，如统计理论、数据挖掘或机器学习理论，从样本集中识别出异常模式的过程。通过分析和拆解异常检测算法，使得异常检测中的涉及的所有运算都能够在安全多方下实现。

## 1.2 进度介绍

目前已经完成了异常的定义以及基本检测算法的设计，另外对于加密算法和安全计算协议也设计了一个初步的方案，并基于暴力检测算法给出了安全多方的解决方案。经过试验证明除了在时间效率上有些不足，但是能都实现密文下时间序列异常检测。

# 2 目前已完成的主要研究工作及结果

## 2.1 安全多方计算的实现

安全多方计算（SMC）是解决一组互不信任的参与方之间保护隐私的协同计算问题，SMC要确保输入的独立性，计算的正确性，同时不泄露各输入值给参与计算的其他成员。而本课题采取了同态加密技术作为实现安全多方计算的基本加密算法。

### 2.1.1 同态加密算法以及BCP加密算法简述

同态加密技术是一种可以直接对密文数据进行处理的加密方法,可以有效的保障用户的数据内容安全,在云存储应用的背景下,具有着非常广阔的发展潜力。利用同态加密算法,不仅能够保障密文数据不会被进行统计分析,进而破解出对应的明文,而且还能够对密文实施加法操作和乘法操作等同态操作,并且可以在操作的同时保持该密文对应的明文顺序。检索过程中,使用的索引文件和关键字都是密文的形式,云端服务器无法从检索结果中获得用户数据的任何信息;索引文件小,不会增加云端服务器的存储压力;检索速度快,支持多关键字检索,方便用户的使用,拥有较高的安全性和较强的实用性等优点。

同态加密的意思是可以对密文进行运算操作，目前我们广泛采用的是同态加法，可以用下列公式的形式形式化定义：

其中*m*1和*m*2分别代表两条不同的明文信息，代表使用pk加密， 表示使用sk解密，上式的含义是对加密后的密文进行运算等价于对相应的明文运算后再加密，两者结果相同。

BCP加密系统是一种特殊的同态加密算法，BCP有一个很重要的特点就是它有两套解密服务。在前面我们解密用的是私钥，但在BCP中还有一种称之为主密钥的参数可以解密加密数据，我们用表示，只要是在同一个公共参数下生成的密钥，无论加密的公钥是否相同，都可以用解密得到密文。



图2-1 传统加密方案和全同态加密方案的对比示意图

BCP同态加密算法的基本步骤如下：

初始化：选择一个安全参数*κ，*针对一个二进制位数为κ 的安全素数RSA模数 *N = pq* ( ， 。其中 是两个不同的素数). 随机选择一个数 使得mod , . 要加密的明文数据在 空间内，BCP加密算法输出：

公共参数

主密钥

密钥生成（PP）：选择一个随机数 ，计算*h*mod . BCP加密算法得到：

加密流程（PP, pk）：给定一个明文数据 以及公钥，随机选择 BCP算法加密得到密文数据 的计算过程如下:

mod mod

解密流程（PP, sk）: 给定一个密文数据（A, B）以及私钥 **，**解密得到明文数据m的流程如下：

在介绍中我们提到BCP算法有一个很重要的特点就是它有两套解密服务，除了私钥可以解密，主密钥也可以解密，利用主密钥解密的流程如下：

主密钥解密（PP，PK，MK）：给定一个密文数据（A, B），用户公钥**，**以及其主密钥MK，令用户的私钥sk=（未知），首先，我们计算 ：

其中表示*k* 模 *N* 的逆. 接着我们计算 *:*

令 表示 模*N* 的逆，令 BCP 算法计算出明文数据*m* :

至此，我们解释了BCP提供两套解密服务的基本原理，只要是在同一个公共参数下生成的公钥（PP），都可以使用MK去解密其加密的数据，在下一节当中我们会着重介绍如何利用BCP的这种特性解决多密钥下密文计算的问题，

针对BCP的同态性，证明如下：给定两个明文数据 在相同的公钥下, 我们有

另有

密文刷新算法（**Ciphertext Refresh**）：密文刷新算法（CR）是指，在刷新密文数据的同时不会改变明文数据m，其流程是随机选择一个数 将密文数据做如下刷新

其中

### 2.1.2 系统模型

本课题采用了如图2-2所示的模型，服务器端采用两个半诚实的服务器S和服务器C构成，其中服务器C是与用户交互并计算的服务器，服务器S主要产生BCP公共参数与主密钥并且参与过渡计算。由于服务器S产生了公共参数和主密钥，因此它可以解密任何一个客户端的密文数据，这是十分危险的，因此我们整个系统的思想就是要利用S的主密钥解密数据，并且屏蔽S任何获取客户信息的行为。整个系统的工作流程如下：

1. 首先有服务器S产生BCP算法所需要的公共参数PP以及主密钥MK，然后将公共参数PP发送至服务器C。
2. 服务器C收到PP，并且分发到每一个客户端。客户端接收到PP后，根据PP生成公钥和私钥，并将公钥广播出去，服务器C中储存每一个用户的公钥。客户端将自己的数据通过自己的公钥加密后上传到服务器C上。
3. 客户端请求某个运算（涉及到多方数据），服务器C和服务器S通过安全转换协议，将不同密钥加密的数据转换成同一个密钥加密的数据。
4. 对于同一个密钥下加密的数据，服务器C和服务器S安全计算协议计算出结果，再通过安全转换协议将结果转换成请求方公钥的密文。
5. 将结果发送回请求方，请求方用自己的密钥解密结果，得到明文数据。



Server C

Server S



Users

图2-2 系统模型

### 2.1.3 安全转换协议

我们在上一节提到了同态加密算法，对于同态加密算法有一点我们需要确认的是，他的同态性是在同一个密钥下才成立的，即

这里的pk是不变的，由于我们研究的是安全多放下的时间序列异常检测，多方往往就意味着多个密钥。当pk不同时，上述等式就不能够成立，是不能直接运算的。因此我们需要用到上一节中所说到的BCP同态加密算法的特性来解决不同密钥下密文计算的问题。

因为同态加密的同态性是在同一密钥加密下的密文才成立，因此安全转换协议的主要功能是将不同密钥下加密的数据，转换为同一个密钥下加密的数据。同时这一过程的逆过程也是正确的。

安全转换协议一：**KeyProd**协议**。KeyProd** 协议完成的主要工作就是我们上述说道的将不同密钥下加密的数据，转换为同一个密钥下加密的数据。假设系统中有n个客户，每一个客户对应的公钥为 , 令 . 给定任意一个客户的密文数据, 通过服务器C和S 转换成格式的密文即加密的密文。**KeyProd**流程如下：

1. 服务器C 存储了客户i的密文数据，计算
2. C随机选择一个数据 ，使用加密 得到 计算
3. 将 发送至服务器S，服务器S 使用主密钥解密得到 ，使用机密得到, 发送给服务器C
4. 服务器C接收到, 将用加密得到. 计算

至此我们就可以将不同密钥下的数据都转换成基于加密下的密文，从而进行下一步的计算。这时另一个问题就出现了，当我们完成计算后，得到结果的形式是，我们需要将结果返回给请求的客户。但是客户的私钥只能够解密自己公钥加密的数据，是无法解密加密的数据。因此，将返回到客户的公钥是必须的，即KeyProd协议的逆过程。

安全转换协议二：**TransDec** 协议**。TransDec** 协议完成的主要工作室将加密的密文数据转化为每个客户公钥加密的数据。**TransDec** 流程如下：

1. 服务器C有一个请求结果 需要返还给客户i，首先随机选择，用 加密成
2. 计算 ，发送给服务器S
3. S用主密钥解密得到 ，分别用得到，将这n个加密数据发送给服务器C
4. C接收到n个数据，用客户i的公钥加密得到，计算

至此，**TransDec** 实现了将加密的数据转换为特定客户公钥加密的数据，我们使用两个安全转换协议将不同密钥密文转换成相同密钥下的密文，和其逆过程。

### 2.1.4 安全计算协议

针对算法所需要的加法、乘法和比较运算我们设计了三种协议去实现在安全多方下实现加法（SAP）、乘法（SMP）、和安全最小（SMIN）。

**安全加法协议**（Secure Addition Protocol, **SAP**）：给定两个在不同公钥下加密的密文 服务器C想要知道这两个数的和，我们SAP算法的目标就是得到。 是所有公钥的乘积，同样在BCP中可以用去解密 下加密的密文。

1. 首先选择两个随机数，服务器C 计算：
2. 将X和Y 发送到服务器S，S用主密钥解密，然后计算*Z =，*将Z加密成 再发送到服务器C
3. 服务器C接收到了，计算将R加密成 再计算

这样SAP算法就完成了，整个过程满足了我们先前所讨论的所有要求，因此可以认定是安全的。

**安全乘法协议**(Secure Multiplication Protocol, **SMP**)：给定两个在不同公钥下加密的密文 服务器C想要知道这两个数的乘积，我们SMP算法的目标就是得到。

1. 首先选择两个随机数，服务器C 计算：
2. 将X和Y 发送到服务器S，S用主密钥解密，然后计算*Z =，*将Z加密成 再发送到服务器C
3. 服务器C接收到了，分别计算

最后计算

**安全小于协议**（Secure Less Than, **SLT**），即给定两个在不同公钥下加密的密文 **SLT**协议就是期望获得 其中是表达x与y之间的大小关系。

1. 首先服务器C计算

服务器C抛掷一枚硬币，如果是正面，计算

如果是反面，计算

1. 服务器C选择一个随机数 *r,*  < , 计算 ，将发送到服务器S。
2. 服务器S解密 得到，=1 当 > 否则 = 0.然后将加密成 发送到服务器C
3. 服务器C接收到，如果刚刚抛掷的硬币是正面朝上，则不处理，否则计算
4. 如果, 表示 , 如果 , 表示 .

**安全最小协议**（Secure MIN protocol, **SMIN**）：给定两个在不同公钥下加密的密文SMIN协议就是获得.

1. 服务器C和服务器S共同计算以下公式：

1. 一旦获得了，服务器C计算

## 2.2 异常检测在安全多方下的实现

目前，检测异常的主要方法有三大类，一种是基于描述统计的方法，该方法基于数据是正态假设的分布，适合于低维数据的检测，一类是基于数理统计的方法，例如线性回归、主成分分析、等等，还有一类就是使用机器学习的算法，机器学习包含有监督的学习和无监督的学习，有监督的学习是指给出数据的类别标签，通过训练分类器去判断测试数据的类别。无监督学习主要有基于距离以及基于密度的算法。基于密度主要是局部异常点的检测，基于距离可以检测出全局异常模式。我们研究时间序列异常主要是研究异常模式（异常子序列），因此在本课题中我们采用基于距离的异常检测算法。异常检测算法目前主要的种类如图 2-3所示。

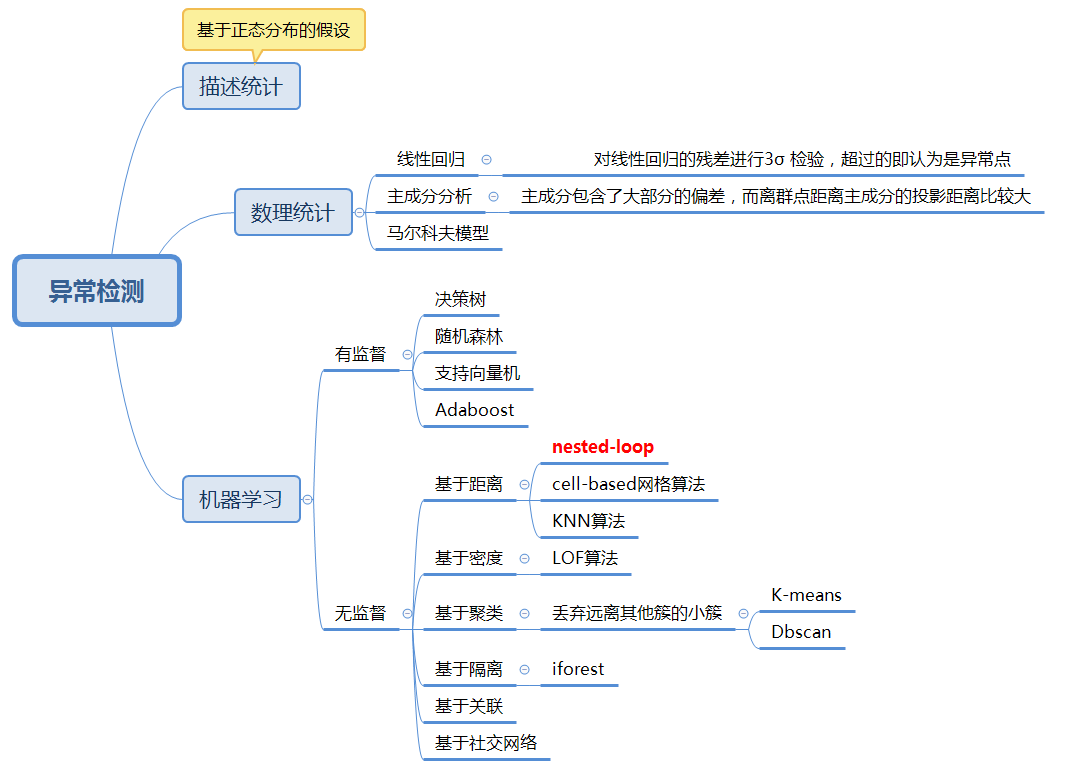


图2-3 常用异常检测算法

接下来在给出异常的定义之前，首先介绍几个基本的概念：

**时间序列欧式距离**：给定两个长度相同（n）的时间序列Q和P，则Q和P的欧氏距离为：

**子序列非自身匹配**：对于长度为m的时间序列*T*，设C为T中长度为n(n<m)的子序列，且其在时间序列*T*中的起始位置为*p*，设*M为T*中另一子序列，其在时间序列*T*中的起始位置为*q*，当时，我们称*M*为*C*的非自身匹配。

**异常子序列**：给定时间序列*T*，当子序列*M*为时间序列*T*中到其最近非自身匹配距离具有最大距离的时候，*M*为*T*的异常子序列。用公式表达可以表示为如下形式（其中*C*为子序列，*MT*表示*T*的非自身匹配，*MC*表示*C*的非自身匹配）：

通过异常的定义我们可以很清晰的看出，到其最近的非自身匹配的距离的最大的子序列为异常子序列。所以我们最容易想到的检测算法就是暴力搜索算法，采用双层循环结构。在外循环中使用滑动窗口提取子序列（假设时间序列长度为n，子序列长度为m，则外循环共需循环n-m次），在内循环中针对每一个子序列，查找到其最近的非自身匹配的距离，这个距离最大的子序列就是异常序列。表2-1就是暴力搜索法的实现。

表2-1暴力搜索法

|  |
| --- |
| 算法：暴力搜索法  输入：时间序列T，子序列长度n  输出：异常子序列位置及异常度anomaly\_loc, anomaly\_dist |
| 1. //初始化异常子序列位置及异常度  2. initialize anomaly\_loc and anomaly\_dist;  3. for each p in T //外层遍历候选子序列  4. initialize current\_dist;  5. for each q in T //内层循环  6. if p and q is no self match//排除自身匹配  7. dist(p,q) smaller than current\_dist then update current\_dist; //更新current\_dist  8. if current\_dist bigger than anomaly\_dist //更新异常位置及异常度  9. update anomaly\_loc and anomaly\_dist;  10 return anomaly\_loc and anomaly\_dist //返回结果 |
|  |

暴力算法无疑是复杂度较高的算法之一，在实际应用中，有大量的算法去优化检测过程，但是由于我们研究的是基于密文下的运算，暴力算法涉及的运算较为简单，容易理解。因此我们先尝试将暴力运算转化成密文下的运算，如果可以实现，在考虑算法的优化，如果在密文下连暴力算法都无法实现，那么一开始采用优化过后的算法也是没有意义的。

暴力算法拆解：将暴力算法拆解我们可以发现，暴力算法涉及到的运算是很简单的，其中主要是距离的计算和比较。其中距离计算就是计算欧式距离，通过欧式距离的定义我们可以知道，其中主要包含加法、乘法、和开方运算（减法也是一种加法）。而观察表2-1第6行，第8行可知，在暴力运算中，我们不需要真的去计算开方运算，因为对于得到的距离我们仅仅只做比较，而不需要确切的知道它的值。因此我们完全可以去掉开方运算，因为如果 那么一定有 (距离肯定都是非负的)。所以，将暴力法拆解后我们发现，要想实现安全多方下的时间序列异常检测，就必须要实现加法、乘法以及比较运算在密文下的计算。而在2.1.4中我们已经介绍了这三种运算在密文下的实现。

**2.3** 实验验证

以上我们介绍了三种安全计算协议，接下来我们初步设计实验测试了三种安全计算协议的效率，如下图所示，我们采用的BCP加密算法中，N的大小是1024比特，下图中所有的时间我们均采用了一千次运算求均值的方法防止其他原因影响结果。

表2-2 三种协议的效率（1000次取均值，N的长度为1024比特）

|  |  |  |  |
| --- | --- | --- | --- |
| Protocol | Server *C* compute | Server *S* compute | Total |
| **SAP** | 24.016 MS | 35.622 MS | 59.520 MS |
| **SMP** | 35.463 MS | 35.644 MS | 71.138 MS |
| **SMIN** | 144.08 MS | 179.67 MS | 368.37 MS |

在BCP加密算法中我们知道，N的长度很重要，公钥、私钥、主密钥的生成以及明文的长度都和N有着密切的关联。上面的表中我们只考虑了N的长度是1024比特时的情景，改变N的值后会有什么样的情况呢？如下图所示，同样的对于每一个数据，我们都是运算1000次求它的平均值。

图2-4 不同长度N对应的计算时间

从上图可以看出，随着N的长度的不断增加，算法开销有着明显的增长，这就意味着N越大，我们需要运算的时间就越多。

上面的试验中我们采用的明文长度都是一样的，在BCP加密算法中，我们对明文的要求是。那么明文大小的改变会影响BCP加密算法的效率么，我们给出下面的实验结果。

图2-5 不同长度明文对应的计算时间

从上图可以看出，随着明文长度的变化，各个协议所耗费的时间几乎不变，这就说明明文的大小对于BCP加密算法来说没有影响。可也是可以解释的，因为在BCP加密算法中，无论明文的大小多大，只要你满足，BCP，所以在基于密文的运算上，只要N保持不变，无论明文的大小多大，BCP加密算法都会将其加密成形如 , 所耗费的时间是差不多的。

在以上实验的基础上，我们进行了一些时间序列异常检测的实验，给出了两组时间序列的异常检测。

**ECG 数据**：心电图（ECG）是反映心脏兴奋的电活动过程，它对心脏基本功能及其病理研究方面，具有重要的参考价值。心电图可以分析与鉴别各种心律失常；也可以反映心肌受损的程度和发展过程和心房、心室的功能结构情况。在指导心脏手术进行及指示必要的药物处理上有参考价值。然而，心电图并非检查心脏功能状态必不可少的指标。因为有时貌似正常的心电图不一定证明心功能正常；相反，心肌的损伤和功能的缺陷并不总能显示出心电图的任何变化。所以心电图的检查必须结合多种指标和临床资料，进行全面综合分析，才能对心脏的功能结构做出正确的判断。下图表示的就是一个心电异常数据。

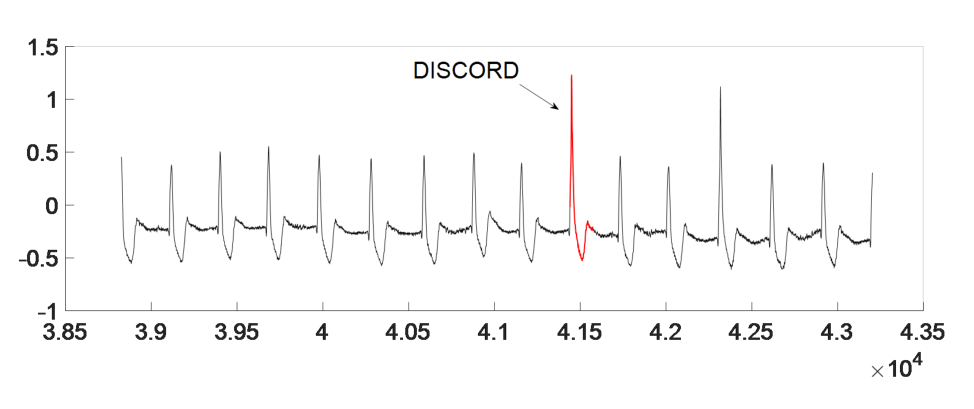


图2-6 心电数据异常

在上图中我们可以很明显得看出来红色部分是异常数据，因为它的模式和其他模式显著的不同，但是我们考虑一下某种特殊情况，如下图所示，时间序列A和时间序列B是分别属于A和B两家医院，B医院现在想在时间序列B上发现异常，如果B医院只考虑时间序列B的情况，那么按照时间序列异常的定义子序列x1是有最大异常度的序列，会判定x1为异常子序列，可实际上时间序列B中x1以后的数据都是不正常的，反而x1作为正常的数据在不正常的数据中间显得“不正常”。但是当我们考虑时间序列A时，由于A中的数据都是正常的，和x1的模式都很相似，结合之后发现x2拥有最大的异常度，因此，算法判定x2是异常子序列。所以我们可以看当，引入多放之后，可以解决在单方下无法解决的问题。

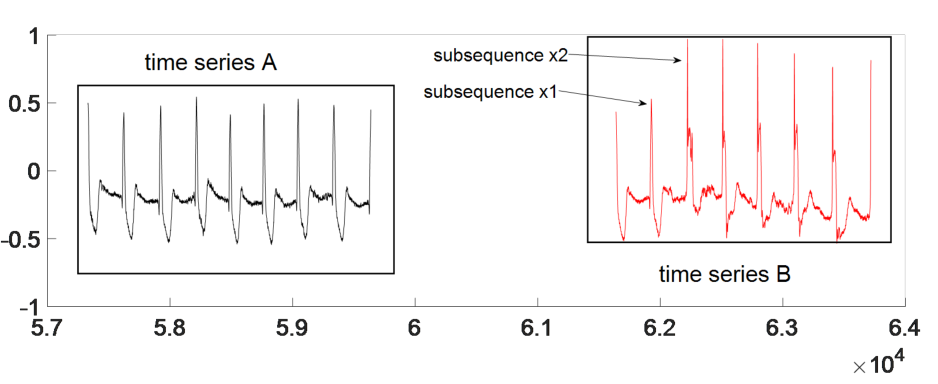


图2-7 多方参与下的心电数据

**持枪检测**：我们从一个视频图像中可以提取出一个演员持枪时他的持枪手坐标位置的变化，从而将一个视频文件转化成一个2D的时间序列，如下图所示，两条时间序列代表着演员持枪手的X坐标和Y坐标的变化。我们对每一种模式都做了标签，具体如下图所示。

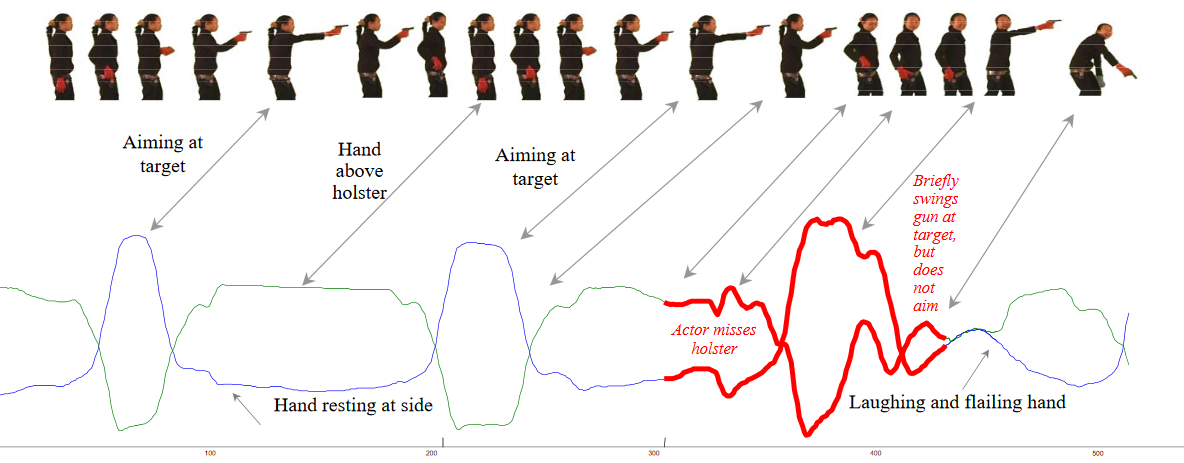


图2-8 枪支检测



图2-9 多方参与下枪支检测

如上图所示，两个时间序列A和B。其中B中所有的模式都是安全的，现在A想要找到她的危险模式出现在什么时候，如果A仅仅考虑自身的数据，调用算法，子序列a的异常度最大，因此算法会判定子序列a为异常。但是实际上a不是异常，因为a的模式和B中的模式是匹配的，因此我们如果把B考虑进来，再次调用异常检测算法，那么就会得到子序列b的异常度最大，那么算法就会判定子序列b为异常。

表2-3 时间效率对比

|  |  |  |
| --- | --- | --- |
|  | 明文计算耗时 | 安全多方计算下耗时 |
| ECG异常检测 | 1.01 S | 10hour |
| 枪支检测 | 0.6 S | 3hour+12min |

上述的两个实验都说明了在单方下异常检测出现的问题在多方中都很好的得到了解决，而我们设计的安全协议又解决了多方下的隐私问题。尽管时间复杂度很高，参见表2-3 明文下计算耗时与安全多方下计算耗时比较。但是都可以将异常准确的找出来，我们接下来的工作就是降低时间复杂度。

# 3 后期拟完成的研究工作及进度安排

后期需要进一步优化异常检测算法，我们在安全多放下提出的仅仅是最直观复杂度很高的暴力检测算法。有大量的优化方法我们还没有采用，下一步的计划就是在安全多方参与下如何去优化检测流程，提高检测效率。

另外，我们目前的使用的算法都是在单机上实现的，服务器C和服务器S也是由两台单独的主机实现的，我们后期的一项重要工作就是要将这些单机上运行的算法移植到分布式系统中。后期进度安排如下：

2017年3月- 2017 年5月：提出新的算法，并设计协议。

2017年5月- 2017年7月：进行编码、实现以及测试，分析实验结果，得出结论。

2017年7月-2017年9月：进行系统总体测试，总结。

2017年9月-2017年12月：总结研究结论，撰写毕业论文，准备答辩。

# 4 存在问题及解决方案

## 4.1 存在问题与困难

加密算法较为复杂，通过实验初步验证将暴力算法转化成安全多方下时，耗费的时间显著增加，在明文下可以实时监测出来的异常，转化到密文下需要数小时的时间去运算。当然我们可以采取优化算法去减少时间复杂度，但是优化算法无疑会采取更复杂的运算和更多的通讯，例如除法、开方等运算。这些运算和通讯在密文下都会导致复杂度的增加。因此，优化加密算法以及设计适合在密文下运算的异常检测算法是一个较为复杂的问题。

## 4.2 解决方案

查看相关文献，仔细研究加密算法的原理，研究优化异常检测算法的主要思想，查阅相关资料，寻求文献原作者或指导老师帮助。认真设计改进检测算法，采用不同的方法进行试验。与指导老师和同学多交流，寻找新思路。认真学习分布式计算相关知识，尤其对Spark这类通用并行框架加强了解。练习将算法设计成满足集群计算要求。

# 5 如期完成全部论文工作的可能性

本课题目前已经完成了暴力检测算法在安全多方下的实现，并给出实验证明其正确性。接下来除了给出更多的实验结果证明，还要对异常检测算法进行优化，争取是时间复杂度降到可以接受的范围内。

对于分布式的实现，现在实验室已经搭建了一个拥有三台主机的小型分布式系统，采用hadoop+spark架构，已经做出了一些初步的实验工作，在优化算法步骤完成后可将其移植到分布式平台上。

综上所述，论文能够按期完成，并取得一定的研究成果。