目录

内容摘要

本研究主要针对于https协议中的SSL协议控制包的交互过程(主要是SSL握手交互过程)在经过shadowsocks的代理客户端与代理服务器中间的加密通道的侧信道信息泄漏的问题，提供对应的混淆策略的设计。混淆策略主要的目标是去除在代理客户端与代理服务器中间的加密数据包的方向、大小、时序三者组合的可识别的规律。混淆策略主要提供混淆算法、混淆协议、以及混淆数据库三者结合来去除侧信道的信息泄漏。其中混淆算法是用数据流量的交互过程代替控制包的交互过程，混淆协议的作用是给混淆算法提供代理客户端与代理服务器之间定义交互的数据结构，混淆数据库给混淆算法提供一个随机选取的数据流量特征选择的多样性、随机性。研究的代理shadowsocks的版本选取为shadowsocks的go语言版本。

关键词：代理、侧信道信息泄漏、shadowsocks、流量混淆

abstract

The aim of this reasearch is to solve the side channel attack between shadowsocks client and shadowsocks server when https stream across the shadowsocks encrypt channel.The reasearch provide mix method.It can mix the rules of direction、length、time series which can be recognized.The mix method provide a mix algorithm、mix protocl and mix database to solve the side channel attack.The mix algorithm select the normal data stream exchange schema to replace the control stream schema.The mix protocl provide the data strucate for the mix algorithm.The mix database provide a lot of the normal data stream for mix algorithm to random select.The proxy of this reasearch is the shadowsocks base on golang.

keywords: proxy、side channel attack、shadowsock 、data stream mix

1.绪论

1.1项目的研究背景与研究意义

随着网络技术日新月异的发展，代理技术也在越来越多的互联网场景中产生应用，如组建vpn (Virtual Private Network),在公用网络上建立专用网络，进行加密通讯。在企业网络中有广泛应用。VPN网关通过对数据包的加密和数据包目标地址的转换实现远程访问。

在如此多的应用场景中，代理技术的安全问题在当今也是一个值得关注的部分，代理如果出现安全问题，会导致中间人窃取代理间交互的流量，篡改代理流量，截断代理流量等问题。代理安全问题的出现会严重影响业务的安全性。

代理服务器与代理客户端之间对实际应用流量采用加密算法进行加密，对应的加密算法主要是对称加密、流加密等形式。加密算法的的类型多种多样，且经过严密的数学计算证明其安全性。所以直接针对加密算法的攻击难度较大。但是当流量经过加密通道后，针对侧信道的信息泄漏(又称边信道攻击 side channel attack)，可以通过相应的流量监听方法获得，且较容易的获取侧信道的相应规律，这对于识别代理种类，以及代理模式起到了关键的作用，获得这些信息后有针对性的进行对已知类型代理的攻击，针对对应代理的相应漏洞，简化攻击过程，降低攻击难度。

所以针对侧信道的信息泄漏，对于代理的安全性也起到了至关重要的作用。本研究主要针对网络上著名的开源项目shadowsocks的go语言版本，在https协议中的ssl协议的控制包通过shadowsocks的加密通道中穿过时，会造成侧信道的信息泄漏，导致代理客户端与代理服务器之间存在代理应用被识别的问题，针对该问题，提出混淆方案，解决侧信道的信息泄漏。

1.2项目研究现状与趋势

现阶段侧信道的信息泄漏在许多安全场景中有所体现，如通过CPU缓存来监视用户在浏览器中进行的快捷键及鼠标操作，进而重塑用户使用情景。针对加密电子设备在运行过程中的时间消耗、功率消耗或电磁辐射之类的侧信道信息泄露而对加密设备进行攻击等等。边信道攻击所需要的设备成本低、攻击效果显著，严重威胁了密码设备的安全性。

针对于加密代理的侧信道信息泄漏研究较少，且因代理在互联网中应用场景之多，所以对于该类研究意义较大。

1.3研究过程

(1)针对https流量通过流加密的侧信道信息通过wireshark进行抓包分析，确定侧信道信息泄漏的模式。

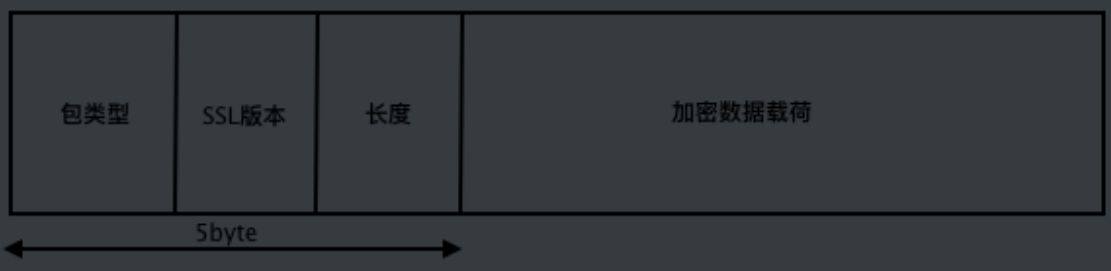
(2)编写模拟https请求的python脚本，便于混淆代码的编写的调试、分析

(3)针对shadowsocks 进行混淆算法、混淆协议的设计与开发

2.相关技术

2.1 https协议

HTTPS（全称：Hyper Text Transfer Protocol over Secure Socket Layer 或 Hypertext Transfer Protocol Secure，超文本传输安全协议），是以安全为目标的HTTP通道，简单讲是HTTP的安全版。即HTTP下加入SSL层，HTTPS的安全基础是SSL，因此加密的详细内容就需要SSL。

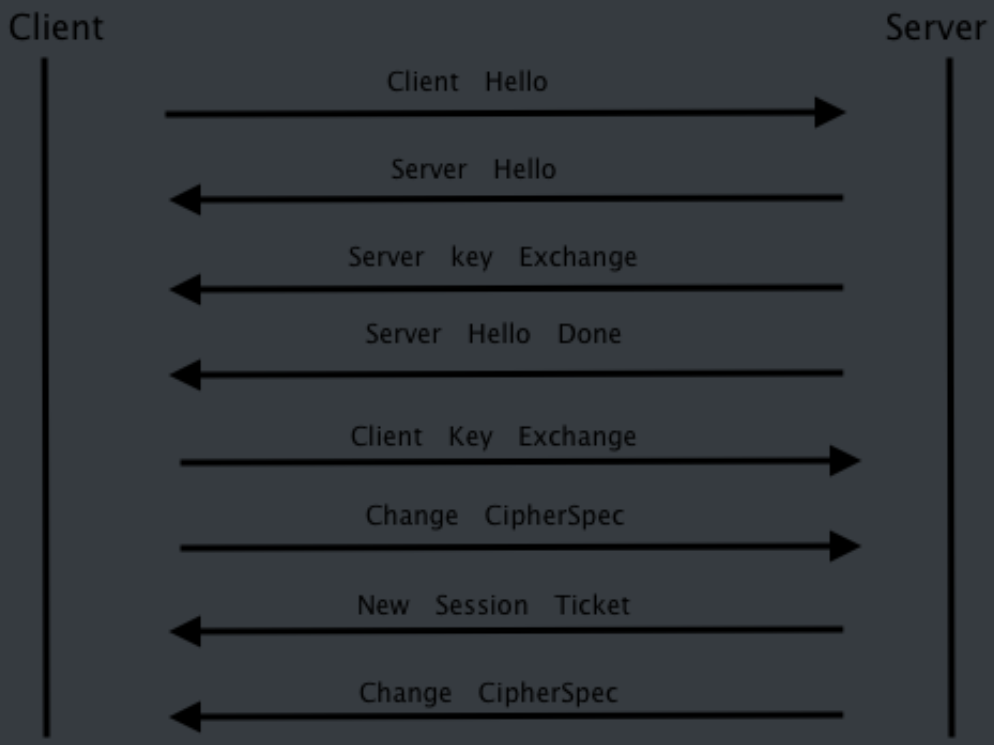


包类型：

16进制	10进制	类型
0x14	20	ChangeCipherSpec
0x15	21	Alert
0x16	22	Handshake
0x17	23	Application Date

SSL协议的协议头主要有三部分构成：包类型、SSL版本、包长度。此次研究主要针对SSL的控制包的侧信道泄漏，所以需要通过SSL协议头的版本号来区分控制包的意义。SSL头中包类型可以用来区分包是否为控制包，上表中除了0x17以外，其它的都是控制包。

而控制包的集中出现的区域是SSL握手的过程，所以研究针对于SSL的握手过程进行详细的分析。下图为SSL的握手的过程，在该过程中，每一个包都是以键值对的形式组织成相应的控制包，对于每一个字断的值都有一定的浮动范围，所以最终导致宏观上整个包的大小的范围是可预测的。



所以整个握手过程中的交互包的大小可以体现在下表中。

编号	意义	方向	大小(整个帧)
1	Client Hello	C—>S	400 ±
2	Server Hello+Certificate+Server key Exchange+Server Hello Done	C<—S	1494(tcp满载荷)
3	同上	C<—S	1494(tcp满载荷)
4	同上	C<—S	1494<
5	Client Key Exchange+Change CipherSpec	C—>S	192
6	New Session Ticket + Change CipherSpec	C<—S	300 ±

对于该交互过程，可以把上表中每一个包可以抽象为三个属性来表述：no(时序顺序),driaction(方向:有0、1两个方向),len(包的长度)，上表中的交互过程，可以抽象为下面抽象数字特征。

对应数字特征：

[{"no":1,"driaction":0,"len":400 ± }

{"no":2,"driaction":1,"len":1494 }

{"no":3,"driaction":1,"len":1494 }

{"no":4,"driaction":1,"len":1494< }

{"no":5,"driaction":0,"len":192 }

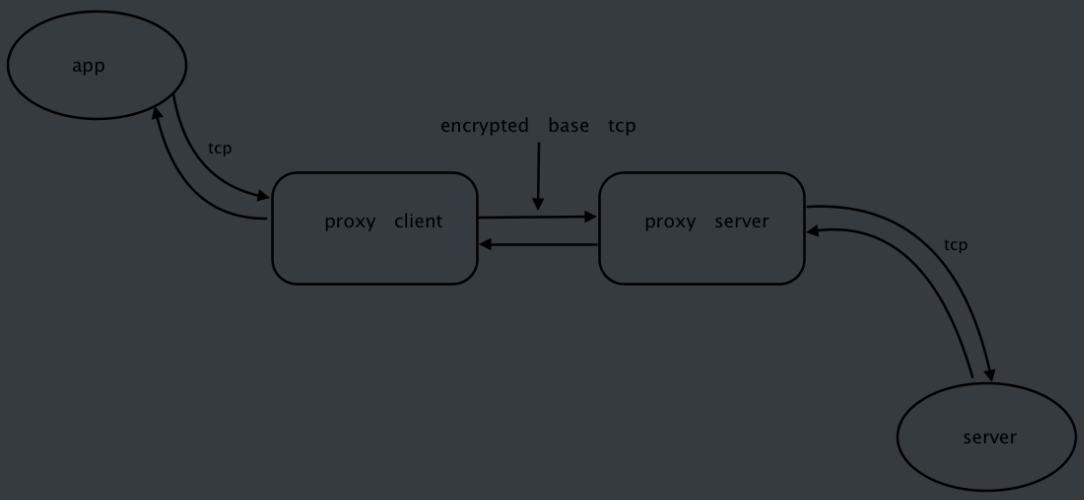
{"no":6,"driaction":1,"len":300 ± }]

2.2 shadowsocks代理

简介:shadowsocks代理是一款基于socks5协议的开源安全代理，使用异步I/O和事件驱动编程的前沿技术，采用行业级加密算法。灵活支持自定义算法。优化的移动设备和无线网络，没有任何保持活着的连接。可在大多数平台上使用，包括Windows、Linux、Mac、Android、iOS和OpenWRT。完全免费和开源。一个致力于提供无bug代码和长期支持的全球社区。易于部署与pip, aur, freshports和许多其他包管理器系统。

研究的基础为shaowsocks的go语言版本，选择的加密协议为 aes-256-cfb。

代理架构:



app通过socks5协议建立app与proxy client(代理客户端的的tcp连接)，代理客户端与代理服务器建立tcp连接，在tcp之上对数据进行shadowsocks自定义的流加密，代理客户端与目标服务器建立tcp连接，形成一个完成的连路，在通过socks5协议把整个链路建立成功后，对于app与server之间代理客户端与代理服务器之间是透明的。

对于研究存在以下特征:

(1) 代理客户端与代理服务器之间的加密协议，明文与密文在长度上的比值为1:1

(2)对于代理客户端与代理服务器，两者之间的功能是对称的，都有把应用流量进行加密、发送到对端，接收对端的加密流量返回给应用的功能。

2.3 总结

针对于2.1中SSL协议在握手过程中的交互特征：no(时序顺序),driaction(方向:有0、1两个方向),len(包的长度)，这三项的特性，经过代理客户端与代理服务器的明文：密文＝1:1的加密传输后，在代理客户端与代理服务器之间就可以收集到与不经过代理的SSL相同的交互特征，造成了侧信道信息泄漏。

3.混淆算法与协议的设计

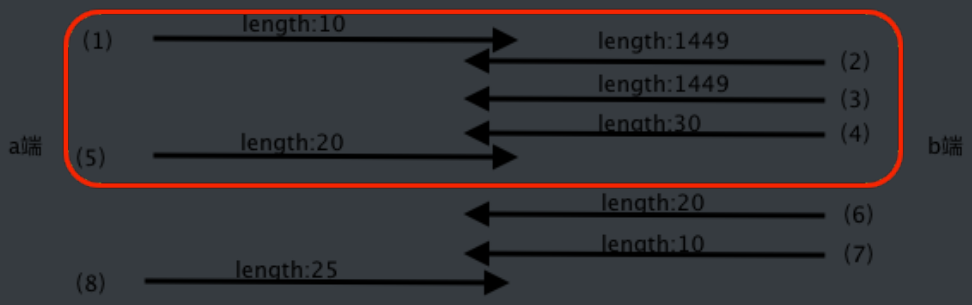
3.1混淆算法

组建一个混淆流量的库，存储一些普通流量(即https进入数据交互时候的流量)，然后选取部分交互过程，把控制流量混淆进普通交互流量的交互过程中。

选取混淆区域

假如代理客户端收到应用发出的SSL握手过程中的client hello包，长度为27，代理客户端。

代理客户端随机在混淆数据库中选取混淆数据特征文件，且随机到流量库中有如下数据流量交互：



因为client hello的长度为27，选取a到b端多个请求长度大于27的,所以选取（1）、（5）号包承载client hello(10+20>27)。

且在混淆数据的交互过程中b向a端发送了（2）、（3）、（4）三个包。

3.2混淆流量库，应用流量的抽象：

混淆数据库是由抓区的大量网站的https流量交互过程中的普通流量的交互过程（头部的包类型为0x17）的序列，包含时序序号(no)、方向(driaction)、包长度(len)的信息。提供给混淆算法随机选取来达到把控制包放入普通流量中，使得侧信道不能监听到SSL的握手模式。

混淆数据库的获取过程：

⑴用wireshark收集大量的应用流量，且所收集的应用流量为https交互中，不含控制包的交互过程

⑵抽象为混淆需要的结构,每条数据三个参数

⑶代理服务器与代理客户端有相同的混淆流量库

例(上方选取的混淆区域的json格式)：

[{"no":1,"driaction":0,"len":10},

{"no":2,"driaction":1,"len":1449},

{"no":3,"driaction":1,"len":1449},

{"no":4,"driaction":1,"len":30},

{"no":5,"driaction":0,"len",20 },

{"no":6,"driaction":1,"len",20 }]

⑶存为json文件，文件名是从1开始的整数（1.json、2.json...），供混淆算法随机选取文件

3.3混淆协议的设计

混淆协议的结构由控制位、真实要传输的数据、混淆数据组成的。混淆控制位用来控制代理两端进行对于混淆包的相应操作提供必要信息：是否需返回纯混淆包、混淆数据的开始位、整个包的长度、该包对应的混淆文件的编号，以及文件中的包的时序编号。数据段是用做插入真正业务需要的真实数据。混淆数据是用来影响包的长度。

控制位：

值	长度(byte)	作用	对应位置
0/1	1	是否需要返回混淆包，0为不需要、1为需要	[0,1)
0~65535	2	混淆数据开始的位置	[1,3)
0~65535	2	包整体的长度	[3,5)
0~255	1	普通流量库的文件号	[5,6)
0~255	1	普通流量库的某文件号的第几号包	[6,7)

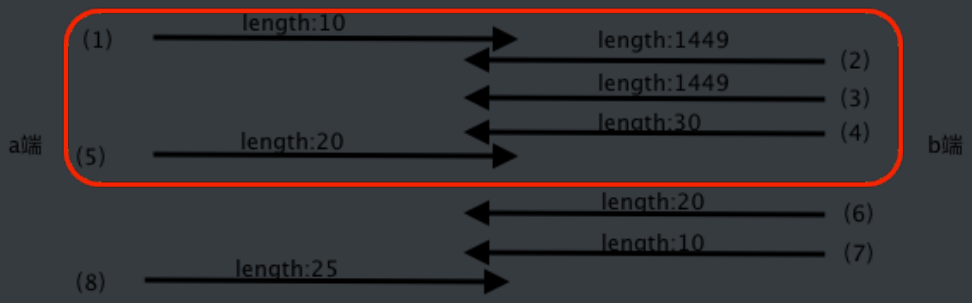
3.4混淆协议与算法结合示例

通过三个部分：混淆算法、混淆流量库、混淆协议，三个的结合来实现对与SSL控制包侧信道流量的泄露的混淆。

假如app向代理客户端发送了一个长度为27 byte的https控制包client hello

1.代理客户端接受client hello,通过密码学随机数从混淆流量数据库随机选取一个混淆特征文件4.json

2.通过计算选取以下部分(介绍混淆算法中有介绍如何计算)



3.代理客户端把client hello分为两部分，分别为长度为10、17的部分

长度为10的(1)号包对应头部表格

对应字段值(位置顺序从小到大)	作用
1	需要代理服务器返回纯混淆包
10	混淆填充位开始的位置
10	混淆包总长度
4	混淆库中对应的文件编号
1	该包模拟的为该文件中包序号

长度为17的(5)号包对应头部表格

对应字段值(位置顺序从小到大)	作用
0	不需要代理服务器返回纯混淆包
17	混淆填充位开始的位置
20	混淆包总长度
4	混淆库中对应的文件编号
5	该包模拟的为该文件中包序号

用以上头部构造包(1)、(5)号包

4.代理客户端发送(1)号包，且等待接受对端发出的纯混淆包 (2)、(3)、(4)

5.代理服务器收到(1)号包后

a.通过混淆协议头，确定需要混淆包的返回

b.定位对应4.json,(1)号包后对应的b端向a端发送的包(2)、(3)、(4)的长度依次为1449、1449、30

c.按照1449、1449、30构造混淆包

d.依次发送

e.向目标服务器发送（1）号包取出的真实数据

对应第一个1449纯混淆包的头部表格：

对应字段值(位置顺序从小到大)	作用
0	不需要代理客户端返回混淆包
0	混淆填充开始的位置
1449	混淆包的总长度
4	混淆库中对应的文件编号
2	该包模拟的为该文件中包序号

其它的纯混淆包构造形式类似。

5.代理客户端依次接收纯混淆包(2)、(3)、(4)并向代理服务器发送(5)号包

6.代理服务器收到(5)号包后

a.通过混淆协议头，确定不需要混淆包的返回

b.向目标服务器发送（5）号包取出的真实数据

总结：

整个混淆长度为27的client hello的过程就完成了，整个过程中发送client hello的代理客户端扮演着导演的角色，他要求对端返回或不返回纯混淆的包，代理服务器收到代理客户端发送的包，通过混淆协议头，返回或不返回纯混淆的包。当SSL控制包是从代理服务器发向代理客户端的时候，导演的角色就给了代理服务器，混淆的过程相同。总体上，通过混淆数据在包中的填充影响包的长度，通过在交互过程中填充纯混淆数据来控制交互过程中的包的数量，通过模拟混淆流量数据库中的交互模式来影响交互过程中的时序。

4.混淆实现技术点

4.1 buffer的设定

在代理客户端、代理服务器实现接收TCP流数据buffer(缓冲区)，达到基于流协议的分包

（1）由于shadowsocks是基于tcp流协议，代理客户端要识别shadowsocks中https包的头来区分是否为控制包(除0x17数据包外)，所以要通过实现buffer在接收https流量，收集整个包后，通过头部的判断，将控制包通过混淆算法进行混淆。

（2）同样是因为基于tcp流协议，所以混淆协议也需要通过buffer来做到对混淆协议进行分包操作，进而对混淆协议进行 混淆协议的处理。

代理服务器是对称的作用，所以，在客户端与代理服务器都需要两个buffer(缓冲区)来达到对https的流量的分包以及对混淆协议的分包。

4.2混淆流量库的生成

混淆库是由多个由不同编号(编号由1开始的整数)文件组成的应用流量特征json的数据库，生成特征库需要生成大量的应用流量来提取特征。

⑴收集https站点的列表

⑵通过python脚本实现对列表中https网站的访问，并且通过tshark软件抓区对应的https流量包(每个网站对应一个包)

⑶通过python脚本处理抓区的流量包，通过python的scapy模块解析tshark数据包，过滤掉tcp握手控制包(如ack不带数据)，对每一个数据包提取出对应的no（时序编号）、drication（方向）、len（长度）生成对应包的特征json文件。

4.3收发协程并发控制

由于实验基于shadowsocks的go语言版本，且是通过go语言的协程来达到流量并行转发，在此基础上增加混淆协议后，需要协程之间的控制。

shadowsocks中每一个代理会话中有分别有两个tcp连接在代理客户端与代理服务器中。

代理客户端:⑴app与代理代理客户端tcp连接

⑵代理客户端与代理服务器的tcp连接

代理服务器:⑴代理客户端与代理服务器的tcp连接

⑵代理服务器与目标服务器的tcp连接

代理客户端中的转发伪代码:

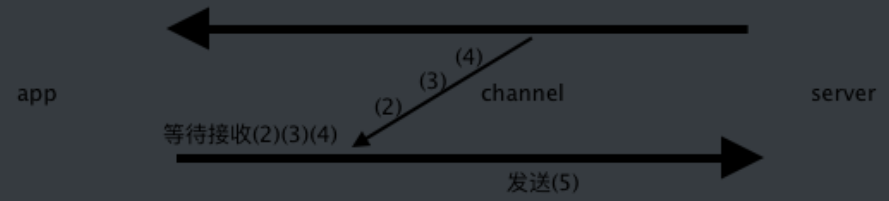
```
connToApp = ReciveAppTcpConn(Local_ListenPort)
connToProxyServer = CreateConnToServer(ServerIP,ServerPort)
go PipeThenClose_to_SSpipe(connToApp,connToProxyServer)
go PipeThenClose_to_app(connToProxyServer,connToApp)


function PipeThenClose_to_SSpipe(src tcpconnect ,dst tcpconnect){
  for{
    receive = src.read( )
    dst.write(receive)
         }
}


function PipeThenClose_to_app(src tcpconnect ,dst tcpconnect){
  for{
    receive = src.read()
    dst.write(receive)
         }
}
```

代理服务器的实现与代理客户端的实现类似

对于代理客户端，通过这两个tcp连接，建立了两个流量通道，分别为从app到代理服务器，代理服务器到app的通道实现转发，为了不与go语言中的channl混淆，以下将app到代理服务器的通道、 代理服务器到app的通道分别用用appToServer、serverToApp代替

对于上面转发与转发与混淆协议结合的示例中，当发出(1)号包后，appToServer需要知道的serverToApp收到的(2)、(3)、(4)号包才能发送(5)号包来达到模拟混淆的过程，所以协程之间需要通知机制。

做serverToApp 向 appToServer 的channel,当收到(2)、（3）、（4）号包后，serverToApp 的通道向channel内发送(2)、(3)、(4)号已经收到的信号量。当处于阻塞状态位于 appToServer中的channel接收到了(2)、（3）、（4）已经接收，然后发送(5)号包，保证时序、大小、方向的模拟。



5.结论与展望

混淆策略可以把经过代理之间的SSL协议的控制包混淆到应用流量的交互模式中，成功的隐藏了SSL握手阶段交互模式的侧信道信息泄漏。在整个研究过程中，混淆策略的模式经过了多重迭带，从实现简单的功能向完整混淆策略进行演变。从简单的设计，到结合代码开发相结合，最终达到能满足实践开发的理论设计。

对应一个控制请求一次混淆，填充了更多的冗余信息来做到握手模式的混淆，这对于性能有相应的损耗，是否能进一步改进混淆模式来做到混淆策略的性能优化。

参考文献

致谢

在理论与实践的过程中，我遇到很多理论构建问题，理论不能支持实践问题，从而经过了多次重构，迭代相应功能，实现结合实践的理论的设计，以及满足理论的程序。这无疑要感谢学校以及老师的培养，对于计算机科学的知识的传授，对于解决问题方法论的培养。同时我要感谢我的指导老师，不断的向我提供意见以及帮助，且给予我自由的环境，从而能让我大胆的试错，进行各种尝试。

最终感谢所有审稿的各位专家，感谢您的批评指正