



密码编码学与网络信息安全 期末作业

题目： 网络封包分析、联邦学
习、量子计算、可信计
算、深度学习报告

姓 名： 干皓丞

学 号： 2101212850

院 系： 信息工程学院

专 业： 计算机应用技术

研究方向： 通信及信息安全技术

导 师： 朱跃生 教授

二〇二二年六月

摘要

本作业为密码编码学与网络信息安全的期末个人报告，其内容包含了讨论 Wireshark 等网络抓包分析工具的使用流程与联邦学习跟隐私安全，同时概略整理量子计算信息安全、可信计算技术、深度学习等在密码学现况的学习的总结报告，其第一章说明讨论 Wireshark 工具的使用与 HTTP 跟 HTTPS 的分析，同时尝试运用 PHP 跟 XAMPP 的工具进行测试，另外还使用 APACHE 伺服器在 Windows 环境下进行证书的设定，最后对 WiFi 探针进行说明。第二章为量子信息在密码学与资讯安全等影响进行工作总结，而第三章则是可信计算技术与近来深度学习在密码学等工作梳理，最后进行报告的总结。

关键词：网络封包分析、联邦学习与隐私安全、量子计算信息安全、可信计算技术、深度学习、密码学

目录

第一章 网络与信息流程分析.....	1
1.1 Wireshark 工具安装与说明.....	1
1.2 Linux tcpdump 封包分析.....	3
1.2.1 Linux 抓包原理	4
1.2.2 tcpdump 抓取 TCP 包分析	6
1.2.3 TCP 三次握手.....	7
1.2.4 TCP 四次握手.....	9
1.3 Chrome 封包分析.....	11
1.4 XAMPP 的 PHP 测试范例与凭证设定	11
1.4.1 XAMPP 说明.....	12
1.4.2 OpenSSL 说明	13
1.4.3 Git 说明.....	14
1.4.4 测试范例	16
1.4.5 PHP 的 Web Server 方案	20
1.4.6 凭证设定	22
1.5 移除凭证.....	22
1.6 进行 HTTP 与 HTTPS 分析	23
1.7 WiFi 探针原理与说明.....	24
1.7.1 何为 WiFi 探针	24
1.7.2 工作原理	25
1.7.3 深入了解	26
1.7.4 WiFi 探针工作	28
1.7.5 WIFI 探针能采集到的数据	28
1.7.6 数据释义	29
1.7.7 安全性.....	30
1.7.8 用何种设备.....	31
1.8 WiFi 探针与移动轨迹.....	32
第二章 联邦学习与隐私安全.....	41

第三章 量子计算机与信息安全对密码学的影响	43
3.1 发展量子计算与通信的意义	43
3.1.1 量子计算的部分	43
3.1.2 量子通信的部分	44
3.2 量子力学的数学	44
3.2.1 量子力学基本概念和原理整理	45
3.3 量子计算的整理	46
3.3.1 量子比特及其特性	46
3.3.2 量子计算机的类型	47
3.4 量子算法	48
3.4.1 量子 Fourier 变换	50
3.4.2 量子搜索	51
3.5 量子通信与量子密码研究进展	52
3.5.1 量子密码研究进展及主要思想	52
3.5.2 量子密钥分发	52
3.5.3 量子加密	53
3.5.4 量子签名	55
3.6 密码学界面对威胁的应对	55
3.6.1 CRYSTALS-KYBER	56
3.6.2 NTRUEncrypt 与 NTRU-HRSS-KEM 合并的 NTRU	57
3.6.3 SABER	57
3.6.4 Classic McEliece	58
3.6.5 CRYSTALS-DILITHIUM	58
3.6.6 FALCON	58
3.6.7 Rainbow	59
第四章 可信计算技术与近来深度学习与密码学	61
4.1 可信计算概述	61
4.1.1 为何需要可信计算	61
4.1.2 何为是可信计算	62
4.1.3 可信计算的发展概况	63
4.1.4 可信计算技术	64
4.1.5 围绕可信计算的一些争议	66

目录

4.2 深度学习与密码学	67
4.2.1 CryptoNets.....	67
4.2.2 CryptoNN.....	68
4.2.3 CV-QNN.....	69
第五章 工作总结	73
参考文献	75
致谢	77

主要符号对照表

x, y, m, n, t	标量, 通常为变量
K, L, D, M, N, T	标量, 通常为超参数
$x \in \mathbb{R}^D$	D 维列向量
(x_1, \dots, x_D)	D 维行向量
$(x_1, \dots, x_D)^T$ or $(x_1; \dots; x_D)^T$	D 维行向量
$x \in \mathbb{R}^{KD}$	(KD) 维的向量
\mathbb{M}_i or $\mathbb{M}_i(x)$	第 i 列为 $\mathbf{1}$ (或者 x), 其余为 $\mathbf{0}$ 的矩阵
$diag(\mathbf{x})$	对角矩阵, 其对角元素为 x
I_N or I	($N \times N$) 的单位阵
$A \in \mathbb{R}^{D_1 \times D_2 \times \dots \times D_K}$	大小为 $D_1 \times D_2 \times \dots \times D_K$ 的张量
$\{x^{(n)}\}_{n=1}^N$	集合
$\{(x^{(n)}, y^{(n)})\}_{n=1}^N$	数据集
$\mathcal{N}(x; \mu, \Sigma)$	变量 x 服从均值为 μ , 方差为 Σ 的高斯分布

① 本符号对照表内容选自邱锡鹏老师的《神经网络与深度学习》^[1]一书。

第一章 网络与信息流程分析

本章針對在深圳研究院的 WLAN 中，从请求接入开始，分别 HTTP 应用以及一种 HTTPS，[E]使用 Wireshark 网络抓包工具，进行抓包，并对抓取的信息流程进行分析，试论 Wi-Fi 探针的原理与应用。而本作業在過程中所面對的問題如下五個章節分[E]進行[E]明，首先講述在 Windows 平台上進行 Wireshark 工具安装，其二使用 XAMPP 框架使用 PHP 跟 APACHE 進行測試，其三是在 Mac 平台上使用 Wireshark 工具分析 HTTP 与 HTTPS 的網頁應用，最後在[E]明 WiFi 探针。

1.1 Wireshark 工具安装与说明

本作业此节说明 Wireshark 工具安装与使用，该节使用的平台为 Windows，在官网下载安装指定的 Windows 方案，其版本是 Wireshark 3.6.3 64 位元，此外在安装的程序中额外选择了 Npcap 1.55 和 USBPcap 1.5.4.0。安装后可看到有波动的网路流量，则是可以分析的目标。进入后则是可以看到整体个流量的状况，同时使用 Windows 平台下的命令提示字元去 Ping GitHub Page，来查看封包状况。

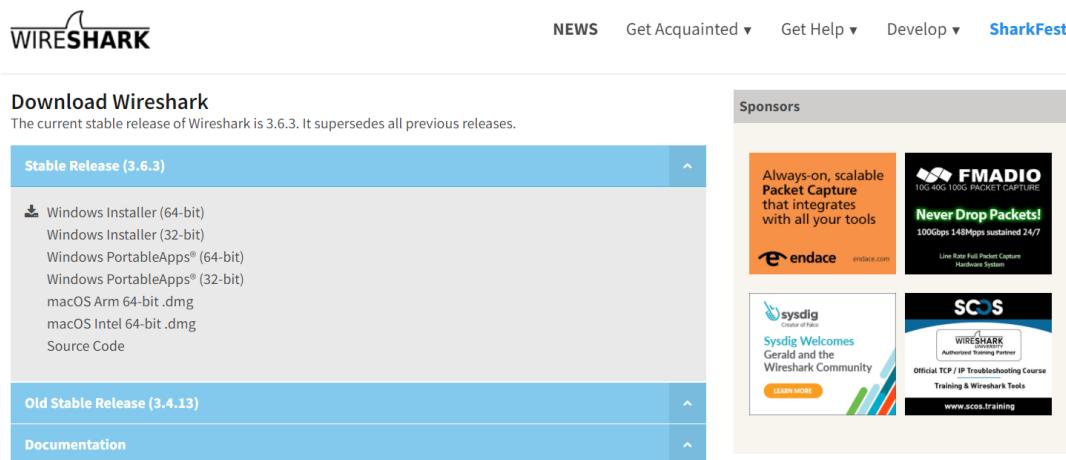


图 1.1 Wireshark 官方网站

Wireshark 最早名为 Ethereal，其溯源于 1997 年底的一位密苏里大学堪萨斯分校毕业生杰拉德·康姆斯 (Gerald Combs) 在一家小型的网际网路服务供应商上班，他因工作需求，迫切需要一个能够追踪网路流量的工具软体辅助其工作故而开始撰写。在经过几次中断开发的事件过后，于在 1998 年 7 月释出其第一个版本。此后康姆斯收到了来自全世界的修补程式、错误回报与鼓励信件。而 Ethereal 的发展就此而始。不久之后，Gilbert Ramirez 看到了这套软体的开发潜力并开始参予低阶程式的开发。1998 年 10 月，

来自 Network Appliance 公司的 Guy Harris 在寻找一套也是网路封包撷取 tcpview 更好的工具，于是他也开始参与 Ethereal 的开发工作。1998 年底，一位在教授 TCP/IP 课程的讲师 Richard Sharpe，看到了这套软体的发展潜力，而后开始参与开发与加入新协定的功能。在当时，新的通讯协定的制定并不复杂，因此他开始在 Ethereal 上新增的封包撷取功能，几乎包含了当时所有通讯协定。自此之后，数以千计的人开始参与 Ethereal 的开发，多半是因为希望能让 Ethereal 撷取特定的、尚未包含在 Ethereal 预设的网路协定的封包而参予新的开发，最后 2006 年因商标的问题，Ethereal 更名为 Wireshark。

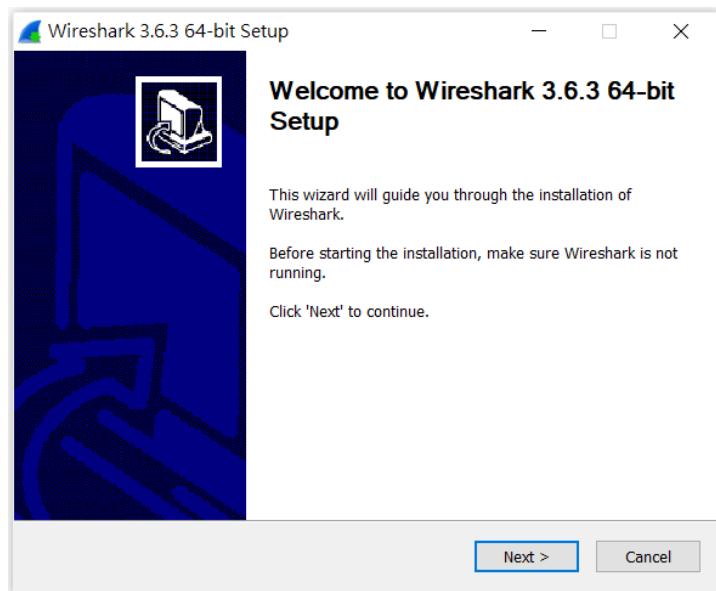


图 1.2 Wireshark Windows 应用程式安装流程

Npcap 是 Nmap 项目的用于 Microsoft Windows 的数据包捕获和发送库，其本身使用定制的 Windows 内核驱动程序以及 Windows 构建的优秀 libpcap 库来实现开放的 Pcap API。这允许 Windows 软件使用简单、可移植的 API 捕获包括无线网络、有线以太网、本地主机流量和许多 VPN 等原始网络流量。Npcap 也允许发送原始数据包，因为 Mac 和 Linux 系统已经包含 Pcap API，因此 Npcap 允许流行的软件，例如 Nmap 和 Wireshark 使用单个代码库在所有这些平台上运行。其 Npcap 最早始于 2013 年，作为对现已停止的 WinPcap 库的一些改进，但与之差别的是从那时起，前者已在很大程度上被重写，经过数百个版本提高了 Npcap 的速度、可移植性、安全性和效率。

USBPcap 是可以让 Wireshark 来获得 Windows 的 USB 数据包，前者可以让后者能够对来自 USB 的封包进行分析，同样地该工具也是开源许可。

在本节可以看到 Windows 平台的 Wireshark 顺利的进行安装，同时对 GitHub Page 进行分析，所谓的 GitHub Page 是开放原始码的版本控制平台 GitHub，其所提供的静态页面服务，多用于该开源项目或者使用者进行项目的展示与说明之用。从分析过

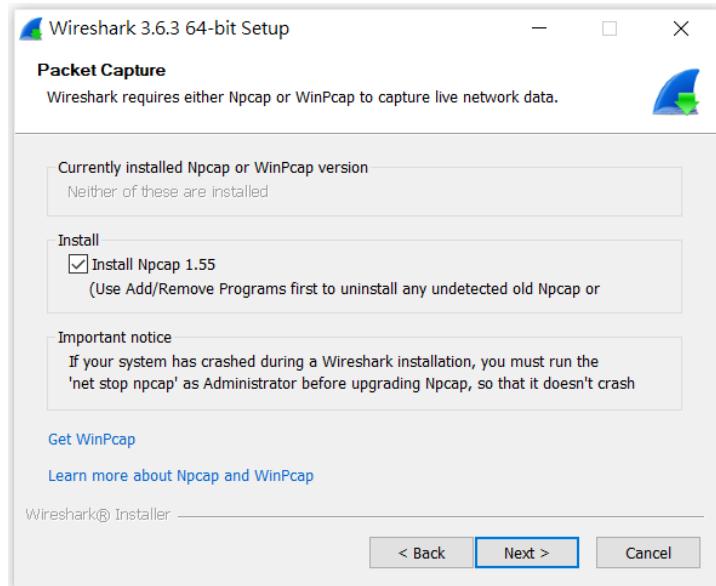


图 1.3 Install Npcap 1.55

程中可以看到 Wireshark 顺利的侦测到 GitHub Page 的状态，后面本作业会尝试说明 OpenSSL 与 CA 的设定。

```
> ping kancheng.github.io
```

1.2 Linux tcpdump 封包分析

在网络问题的调试中，tcpdump 应该说是一个必不可少的工具和大部分 Linux 下的优秀工具一样，它的特点就是简单而强大。它是基于 Unix 系统的命令行式的数据包嗅探工具，可以抓取流动在网卡上的数据包。默认情况下，tcpdump 不会抓取本机内部通讯的报文。根据网络协议栈的规定，对于报文，即使是目的地是本机，也需要经过本机的网络协议层，所以本机通讯肯定是通过 API 进入了内核，并且完成了路由选择，比如本机的 TCP 通信，也必须要有 Socket 通信的基本要素。如果要使用 tcpdump 抓取其他主机 MAC 地址的数据包，必须开启网卡混杂模式，所谓混杂模式，用最简单的语言就是让网卡抓取任何经过它的数据包，不管这个数据包是不是发给它或者是它发出的。一般而言，Unix 不会让普通用户设置混杂模式，因为这样可以看到别人的信息，比如 telnet 的用户名和密码，这样会引起一些安全上的问题，所以只有 root 用户可以开启混杂模式，开启混杂模式的命令是：ifconfig en0 promisc，而 en0 则是假设所要打开混杂模式的网卡。

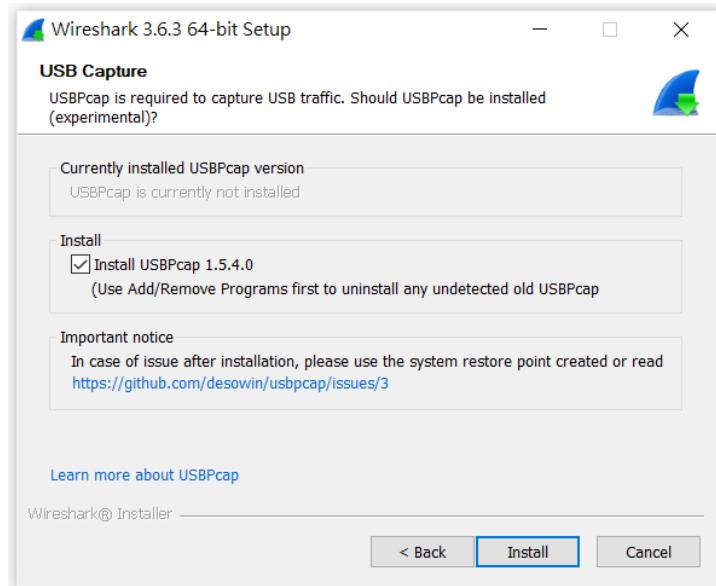


图 1.4 Install USBPcap 1.5.4.0

1.2.1 Linux 抓包原理

Linux 抓包是通过注册一种虚拟的底层网络协议来完成对网络报文，准确来说是网络设备的消息处理权。当网卡接收到一个网络报文之后，它会遍历系统中所有已经注册的网络协议，例如以太网协议、x25 协议处理模块来尝试进行报文的解析处理，这一点和一些文件系统的挂载相似，就是让系统中所有的已经注册的文件系统来进行尝试挂载，如果哪一个认为自己可以处理，那么就完成挂载。当抓包模块把自己伪装成一个网络协议的时候，系统在收到报文的时候就会给这个伪协议一次机会，让它来对网卡收到的报文进行一次处理，此时该模块就会趁机对报文进行窥探，也就是把这个报文完完整整的复制一份，假装是自己接收到的报文，汇报给抓包模块。当然 Wireshark 是一个网络协议检测工具，支持 Windows 平台、Unix 平台、Mac 平台，一般只在图形界面平台下使用 Wireshark，如果是 Linux 的话，则可直接使用 tcpdump，因为一般而言 Linux 都自带的 tcpdump，或者用 tcpdump 抓包以后用 Wireshark 打开分析。而在 Mac 平台下，Wireshark 通过 WinPcap 进行抓包，且封装跟使用方便，可很容易的制定抓包过滤器或者显示过滤器。两者相比，tcpdump 是用来抓取数据非常方便，Wireshark 则是用于分析抓取到的数据比较方便。

```
tcpdump [-adeflnNOpqStvx] [-c< 数据包数目>]
[-dd] [-ddd] [-F< 表达文件>] [-i< 网络界面>]
[-r< 数据包文件>] [-s< 数据包大小>] [-tt]
[-T< 数据包类型>] [-vv] [-w< 数据包文件>] [输出数据栏位]
```

- -a 尝试将网络和广播地址转换成名称。

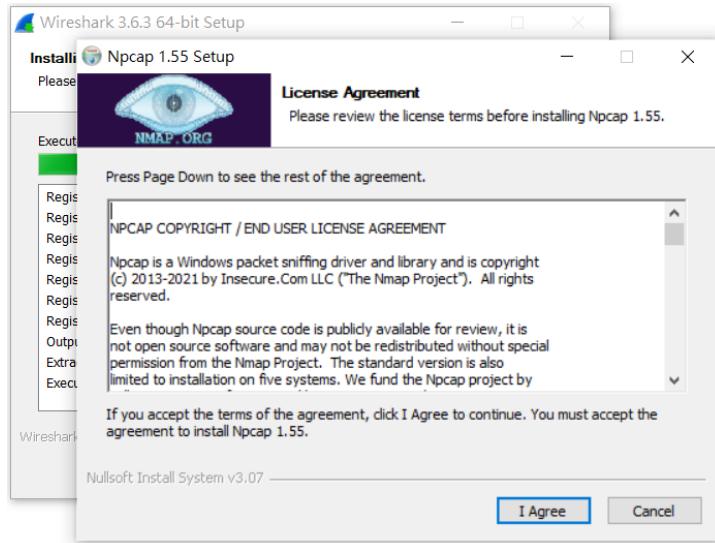


图 1.5 Npcap 1.55 流程

- -c < 数据包数目 > 收到指定的数据包数目后，就停止进行倾倒操作。
- -d 把编译过的数据包编码转换成可阅读的格式，并倾倒到标准输出。
- -dd 把编译过的数据包编码转换成 C 语言的格式，并倾倒到标准输出。
- -ddd 把编译过的数据包编码转换成十进制数字的格式，并倾倒到标准输出。
- -e 在每列倾倒资料上显示连接层级的文件头。
- -f 用数字显示网际网络地址。
- -F < 表达文件 > 指定内含表达方式的文件。
- -i < 网络界面 > 使用指定的网络截面送出数据包。
- -l 使用标准输出列的缓冲区。
- -n 不把主机的网络地址转换成名字。
- -N 不列出域名。
- -O 不将数据包编码最佳化。
- -p 不让网络界面进入混杂模式。
- -q 快速输出，仅列出少数的传输协议信息。
- -r < 数据包文件 > 从指定的文件读取数据包数据。
- -s < 数据包大小 > 设置每个数据包的大小。
- -S 用绝对而非相对数值列出 TCP 关联数。
- -t 在每列倾倒资料上不显示时间戳记。
- -tt 在每列倾倒资料上显示未经格式化的时间戳记。
- -T < 数据包类型 > 强制将表达方式所指定的数据包转译成设置的数据包类型。
- -v 详细显示指令执行过程。

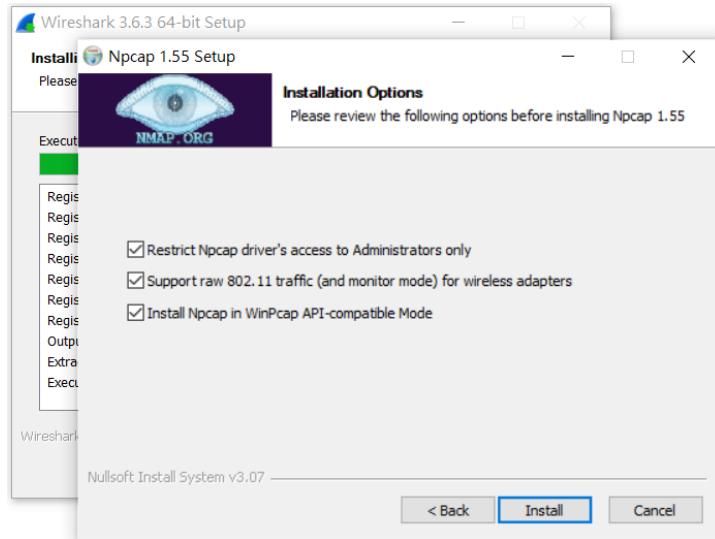


图 1.6 Npcap 1.55 设定

- -vv 更详细显示指令执行过程。
- -x 用十六进制字码列出数据包资料。
- -w < 数据包文件 > 把数据包数据写入指定的文件。

```
$ # 1. 显示 TCP 包信息
$ tcpdump
$ # 2. 显示指定数量包
$ tcpdump -c 20
$ # 3. 精简显示
$ tcpdump -c 10 -q //精简模式显示 10 个包
$ # 4. 转换为阅读格式
$ tcpdump -d
$ # 5. 转换成十进制格式
$ tcpdump -ddd
$ # 6. 通过 tcpdump 截获主机 www.baidu.com 发送与接收所有的数据包
$ tcpdump -i en0 host www.baidu.com
$ sudo tcpdump -i en0 host www.baidu.com
```

1.2.2 tcpdump 抓取 TCP 包分析

TCP 传输控制协议是面向连接的可靠的传输层协议，在进行数据传输之前，需要在传输数据的两端，也就是客户端和服务器端中创建一个连接，这个连接由一对插口地址唯一标识，即是在 IP 报文首部的源 IP 地址、目的 IP 地址，以及 TCP 数据报首部的源端口地址和目的端口地址。其 TCP 首部结构如下。

同时要注意在通常情况下，一个正常的 TCP 连接，都会有三个阶段，也就是 1、TCP 三次握手；2、数据传送；3、TCP 四次挥手，其中在 TCP 连接和断开连接过程中的

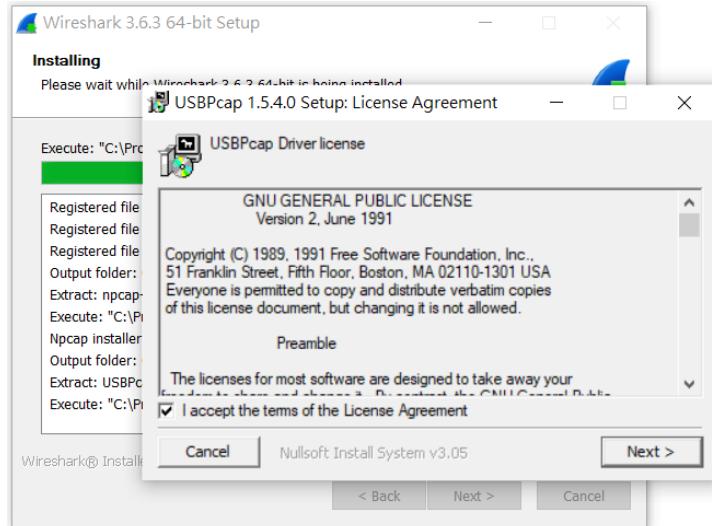


图 1.7 USBPcap 1.5.4.0 流程

关键部分如下：

- 源端口号：即发送方的端口号，在 TCP 连接过程中，对于客户端，端口号往往由内核分配，无需进程指定；
- 目的端口号：即发送目的的端口号；
- 序号：即为发送的数据段首个字节的序号；
- 确认序号：在收到对方发来的数据报，发送确认时期待对方下一次发送的数据序号；
- SYN：同步序列编号，Synchronize Sequence Numbers；
- ACK：确认编号，Acknowledgement Number；
- FIN：结束标志，FINish；

1.2.3 TCP 三次握手

三次握手的过程如下：

- Step1. 由客户端向服务器端发起 TCP 连接请求。Client 发送：同步序列编号 SYN 置为 1，发送序号 Seq 为一个随机数，这里假设为 X，确认序号 ACK 置为 0；
- Step2. 服务器端接收到连接请求。Server 响应，其同步序列编号 SYN 置为 1，并将确认序号 ACK 置为 X+1，然后生成一个随机数 Y 作为发送序号 Seq（因为所确认的数据报的确认序号未初始化）；
- Step3. 客户端对接收到的确认进行确认。Client 发送，将确认序号 ACK 置为 Y+1，然后将发送序号 Seq 置为 X+1（即为接收到的数据报的确认序号）；

至于为什么是三次握手而不是两次对于 Step3 的作用，假设一种情况，客户端 A 向

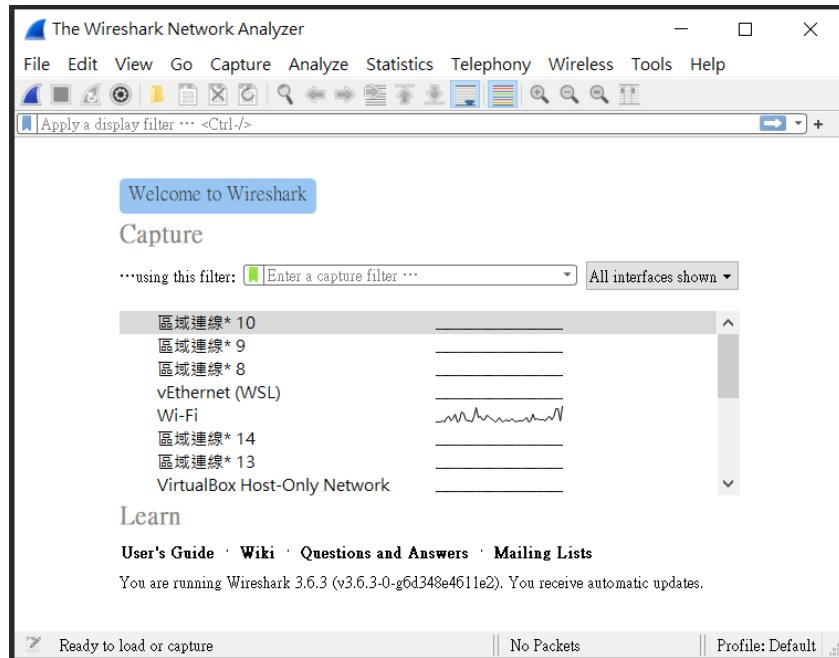


图 1.8 Wireshark 应用程式画面

服务器 B 发送一个连接请求数据报，然后这个数据报在网络中滞留导致其迟到了，虽然迟到了，但是服务器仍然会接收并发回一个确认数据报。但是 A 却因为久久收不到 B 的确认而将发送的请求连接置为失效，等到一段时间后，接到 B 发送过来的确认，A 认为自己现在没有发送连接，而 B 却一直以为连接成功了，于是一直在等待 A 的动作，而 A 将不会有任何的动作了。这会导致服务器资源白白浪费掉了，因此，两次握手是不行的，因此需要再加上一次，对 B 发过来的确认再进行一次确认，即确认这次连接是有效的，从而建立连接。对于双方，发送序号的初始化为何值有的系统中是显式的初始化序号是 0，但是这种已知的初始化值是非常危险的，因为这使得一些黑客钻漏洞，发送一些数据报来破坏连接。因此，初始化序号因为取随机数会更好一些，并且

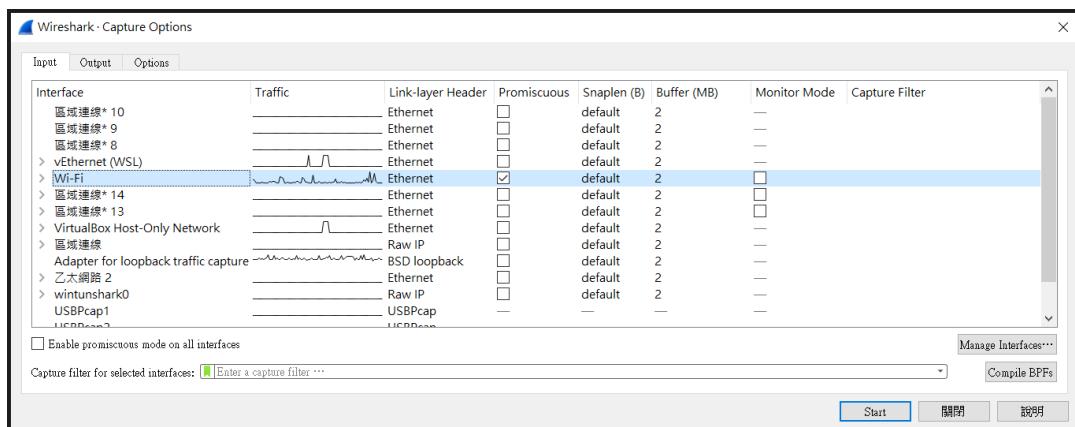


图 1.9 Wireshark 封包选项

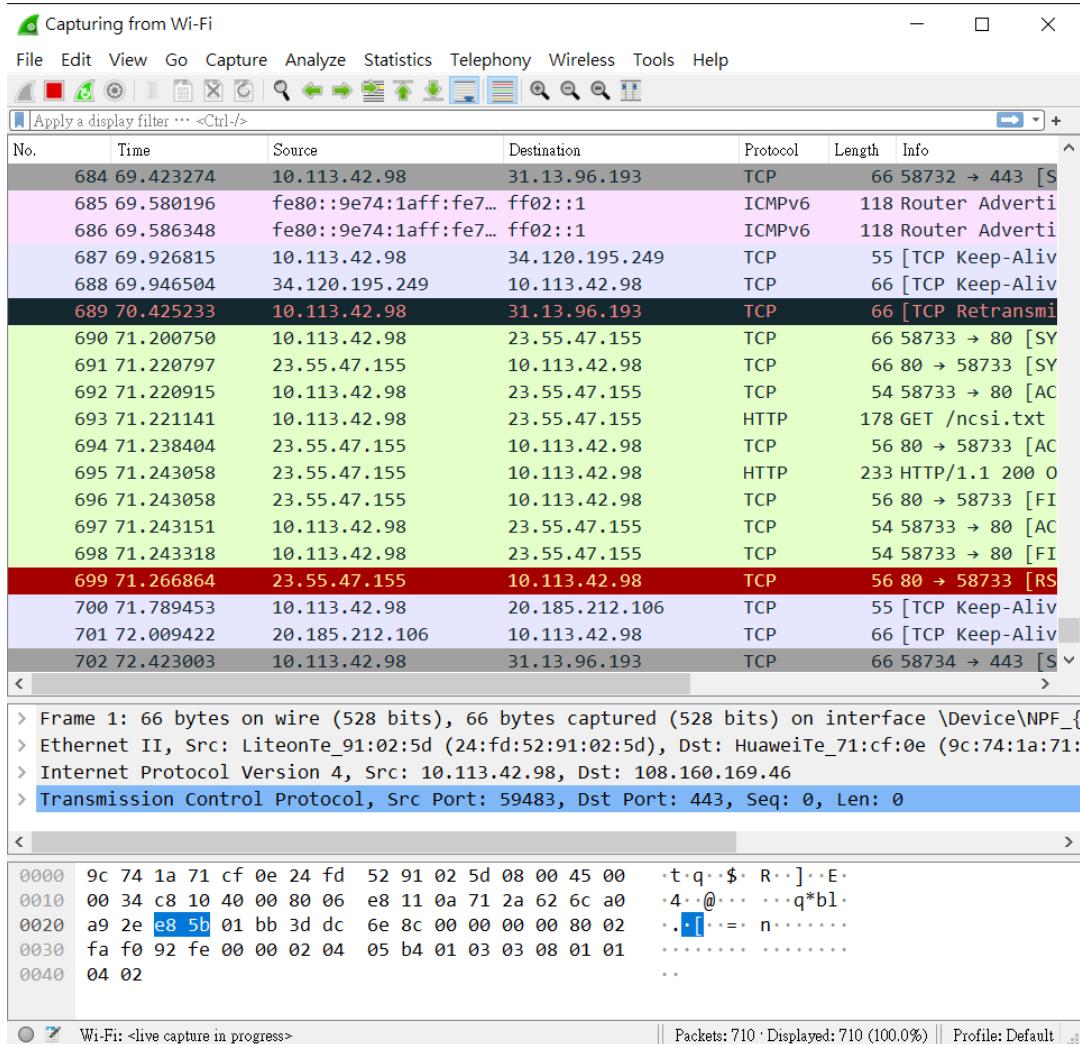


图 1.10 WiFi 追踪

是越随机越安全。

1.2.4 TCP 四次握手

连接双方在完成数据传输之后就需要断开连接。由于 TCP 连接是属于全双工的，即连接双方可以在一条 TCP 连接上互相传输数据，因此在断开时存在一个半关闭状态，即有一方失去发送数据的能力，却还能接收数据。因此，断开连接需要分为四次。主要过程如下：

- Step1. 主机 A 向主机 B 发起断开连接请求，之后主机 A 进入 FIN-WAIT-1 状态；
- Step2. 主机 B 收到主机 A 的请求后，向主机 A 发回确认，然后进入 CLOSE-WAIT 状态；
- Step3. 主机 A 收到 B 的确认之后，进入 FIN-WAIT-2 状态，此时便是半关闭状态，即主机 A 失去发送能力，但是主机 B 却还能向 A 发送数据，并且 A 可以接

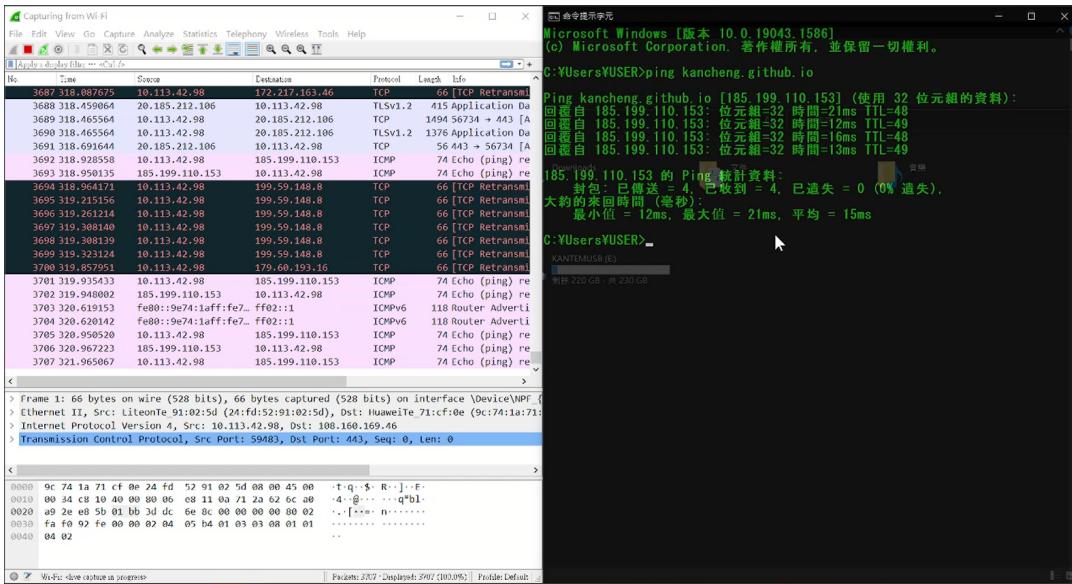


图 1.11 使用 GitHub Page 测试

图 1.12 tcpdump

收数据。此时主机 B 占主导位置了，如果需要继续关闭则需要主机 B 来操作了；

- Step4. 主机 B 向 A 发出断开连接请求，然后进入 LAST-ACK 状态；
 - Step5. 主机 A 接收到请求后发送确认，进入 TIME-WAIT 状态，等待 2MSL 之后进入 CLOSED 状态，而主机 B 则在接受到确认后进入 CLOSED 状态；

为何主机 A 在发送了最后的确认后没有进入 CLOSED 状态，反而进入了一个等待 2MSL 的 TIME-WAIT 主要作用有两个：第一，确保主机 A 最后发送的确认能够到达主机 B。如果处于 LAST-ACK 状态的主机 B 一直收不到来自主机 A 的确认，它会重传断开连接请求，然后主机 A 就可以有足够的时间去再次发送确认。但是这也只能尽最大力量来确保能够正常断开，如果主机 A 的确认总是在网络中滞留失效，从而超过了 2MSL，最后也无法正常断开；第二，如果主机 A 在发送了确认之后立即进入 CLOSED 状态。假设之后主机 A 再次向主机 B 发送一条连接请求，而这条连接请求比之前的确认报文更早地到达主机 B，则会使得主机 B 以为这条连接请求是在旧的连接中 A 发出的报文，并不看成是一条新的连接请求了，即使得这个连接请求失效了，增加 2MSL

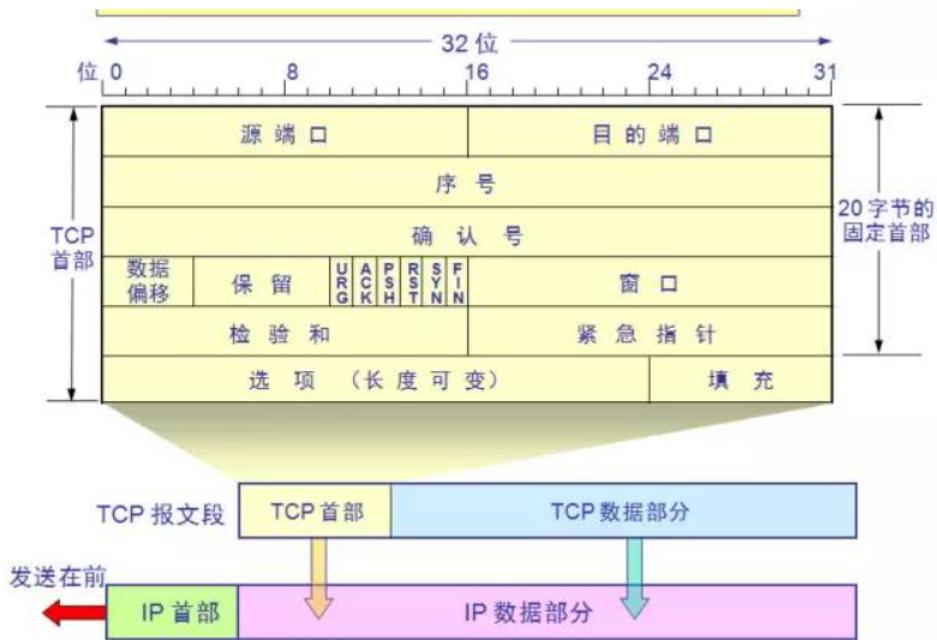


图 1.13 tcpdump

的时间可以使得这个失效的连接请求报文作废，这样才不影响下次新的连接请求中出现失效的连接请求。为什么断开连接请求报文只有三个，而不是四个因为在 TCP 连接过程中，确认的发送有一个延时（即经受延时的确认），一端在发送确认的时候将等待一段时间，如果自己在这段事件内也有数据要发送，就跟确认一起发送，如果没有，则确认单独发送。而我们的抓包实验中，由服务器端先断开连接，之后客户端在确认的延迟时间内，也有请求断开连接需要发送，于是就与上次确认一起发送，因此就只有三个数据报了。

1.3 Chrome 封包分析

所谓的抓包，即抓取本地电脑与远端服务器通信时候所传递的数据包，而 Chrome 开发者工具是一套内置于 Google Chrome 中的 Web 开发和调试工具，可用来对网站进行迭代、调试和分析，而想要打开 Chrome 开发者工具则是在 Chrome 界面中按下键盘的 F12 键，又或者是在页面元素上右键点击选择后，选取检查。

1.4 XAMPP 的 PHP 测试范例与凭证设定

此节本作业除了说明 OpenSSL 等概念，也会尝试使用 XAMPP 与 PHP 分别进行凭证设定与范例，此节本作业会写程式范例同时也会提供 CA 设定档，根据写好的地区设定，在 Apache 伺服器中执行 CA 认证。

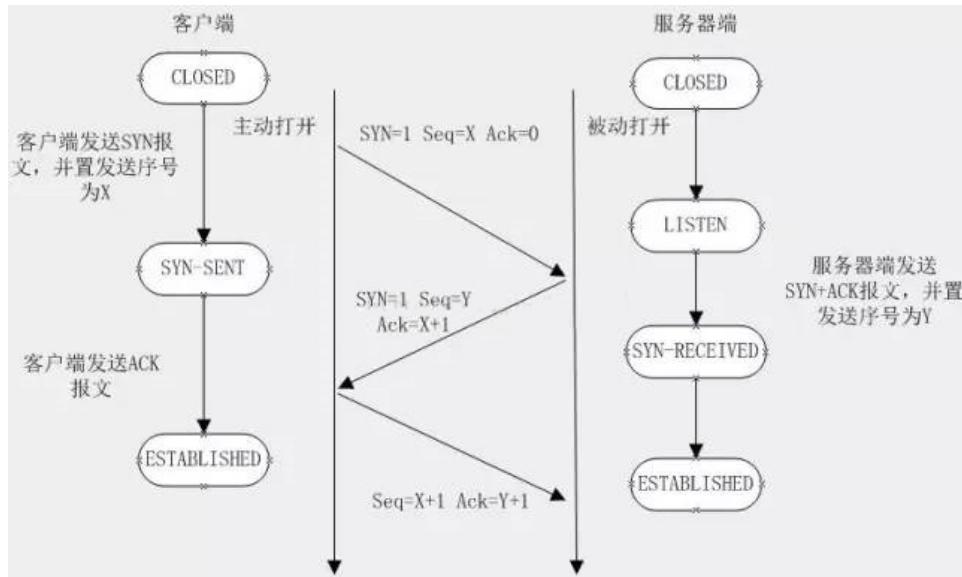


图 1.14 TCP 三次握手

1.4.1 XAMPP 说明

XAMPP 是一个可以简单整合网页伺服器 Apache、伺服器端语言 PHP、程式语言 Perl 及资料库 MariaDB 的软体包，只要透过 XAMPP 就可以方便开发者快速架站，多数使用者在架站时需要透过网页伺服器让访客能够连到所开发的网站上，目前比较普遍的网页伺服器有简称 Apache 的 Apache HTTP 伺服器、Nginx、以及简称为 IIS 的 Microsoft Internet Information Server 等，基本上透过网页伺服器就能将网页包含图像、影音等各种档案提供给请求的使用者。

PHP 最初是由勒多夫在 1995 年开始开发的，现在 PHP 的标准由 the PHP Group 维护，该语言是一种开源的通用计算机脚本语言，尤其适用于网络开发并可嵌入 HTML 中使用，其语法借鉴吸收 C 语言、Java 和 Perl 等流行计算机语言的特点，易于一般程序员学习。而 PHP 的主要目标是允许网络开发人员快速编写动态页面，但 PHP 也被用于其他很多领域。

另外当中的 MySQL 则是在 1995 年，Michael Widenius, David Axmark 及 Allan Larsson 创立了瑞典的 MySQL AB 公司，随之推出了现今最具知名度的同名产品 MySQL，作为关联式资料库的管理所使用。所以 MySQL 从字面上来理解就是一种关联式资料库管理系统 (Relational Database Management System; RDBMS)，同时也因为 MySQL 的问世，让 MySQL AB 曾经成为过去全球最大的开放源码公司。MySQL AB 运用双重许可，一方面 MySQL 属于 GPL 的开放协议，让软体在 GPL 的规范下可以无偿使用，但这样的规范对于部分公司可能不敷使用，例如必须使用到没有被开源的程式码或技术，那就只能靠付费的方式来获得。另一方面，MySQL AB 也靠着顾问服务以及认证的方

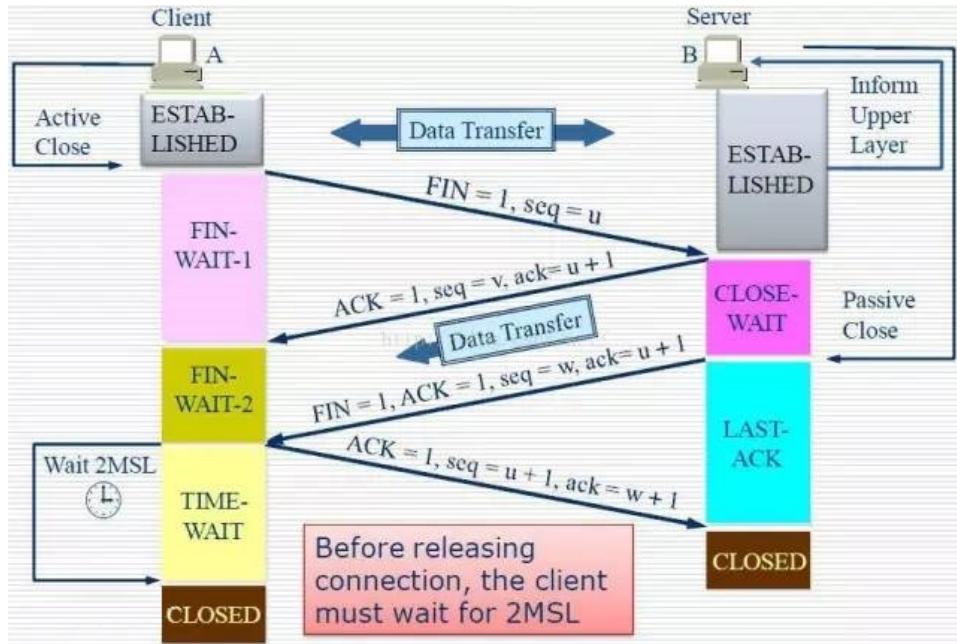


图 1.15 TCP 四次握手

式赚取收入，例如透过开班授课、培训的方式，来取的 MySQL 的 Certificate 等方式获取利润；通过就是卖服务，MySQL 可能不需要付费，但是如果要原厂支援的话则需要付费。而在 2008 年，升阳软体公司 (Sun Microsystems) 透过 10 亿美金的价格收购，而在 2009 年升阳也被甲骨文公司 (Oracle) 收购，于是乎 MySQL 成为 Oracle 旗下产品。但收购后 Oracle 大幅调涨其商业版的售价，让许多开源人士不再看好，担心有一天开源社群版最后就被会商业版取代，于是 Michael Widenius 以 MySQL 为基础，成立分支计划 MariaDB。而一些使用 MySQL 的开源专案逐渐也转向 MariaDB，例如维基百科就在 2013 年正式转换。不过因为 MariaDB 就是直接用 Fork 分流的方式从 MySQL 原始码开始开发，所以原使用者要转换到 MariaDB 上并不太困难，且外部连线函式库都可以共用。

1.4.2 OpenSSL 说明

再来说明 OpenSSL，在计算机网路上 OpenSSL 是一个开放原始码的软体函式库套件，计划于 1998 年开始，该项目的目标是发明一套自由的加密工具，使之在网际网路上使用。其 OpenSSL 以 Eric Young 以及 Tim Hudson 两人所开发的 SSLeay 为基础，但随着两人前往 RSA 公司任职后，SSLeay 与 1998 年 12 月停止开发，故因此在 1998 年 12 月，社群另外分支出 OpenSSL 继续开发。目前 OpenSSL 应用程式可以使用这个套件来进行安全通讯，来避免窃听，同时确认另一端连线者的身分，此套件广泛被应用在网际网路的网页伺服器上。另外该应用的主要函式库皆是以 C 语言所写成，实作

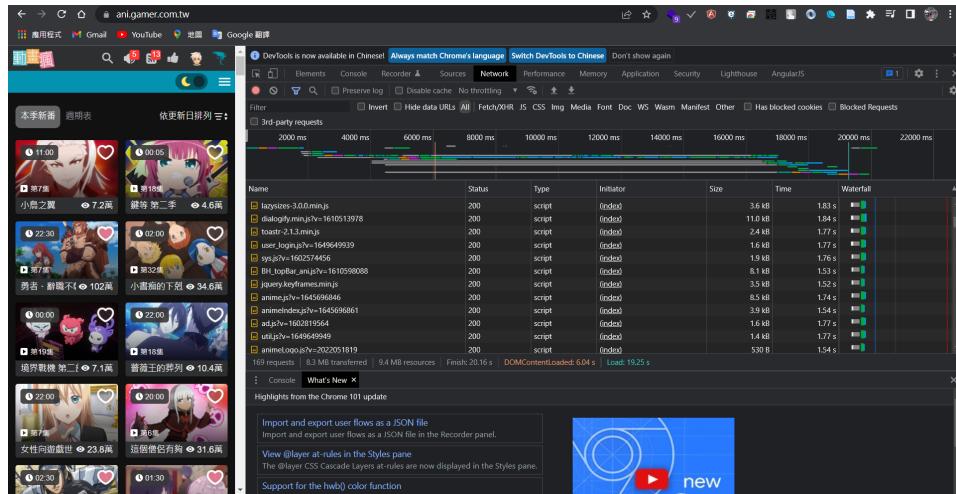


图 1.16 Chrome 介面

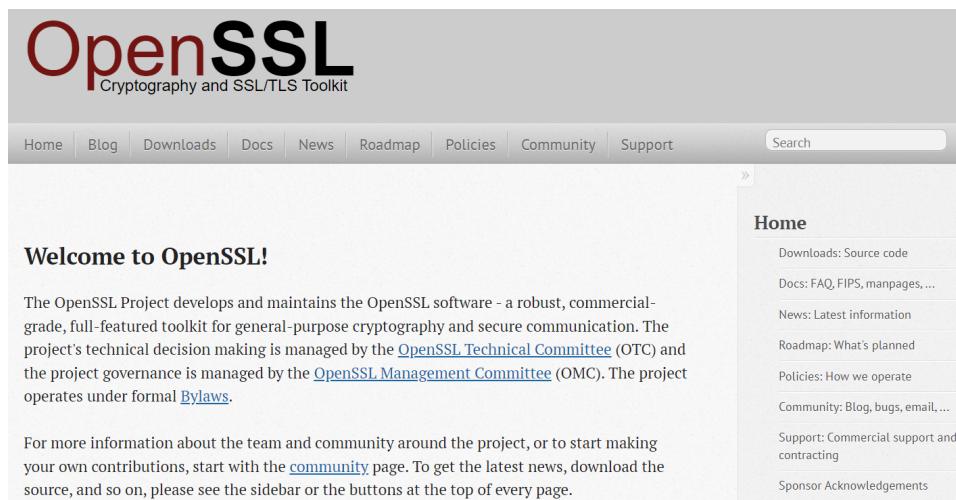


图 1.17 OpenSSL 官方页面

了基本的加密功能，实作了 SSL 与 TLS 协定，此外 OpenSSL 可以运行在 OpenVMS、Microsoft Windows 以及绝大多数如 Solaris, Linux, Mac OS X 与各种版本的开放原始码 BSD 作业系统类 Unix 作业系统上。虽然此软体是开放原始码的，但其授权书条款与 GPL 有冲突之处，故如 Wget 等 GPL 软体使用 OpenSSL 时必须对 OpenSSL 给予例外。

1.4.3 Git 说明

Windows 的 OpenSSL 其本作业在此使用 Windows 平台版本中的 Git 内建的 OpenSSL，另外要说明的是 Git 是一个分散式版本控制软件，最初由 Linux 开发者林纳斯·托瓦兹创作，并于 2005 年以 GPL 授权条款释出，其最初目的是为了更好地管理 Linux 核心开发而设计。同时应注意的是，这与 GNU Interactive Tools 等类似 Norton Commander 界

面的文件管理器有着根本上不同。Git 最初的开发动力来自于 BitKeeper 和 Monotone，其最初只是作为一个可以被其他如 Cogito 或 Stgit 前端包装的后端而开发，但后来 Git 内核已经成熟到可以独立地用作版本控制，并被很多软体专案广泛使用其中包括 Linux 核心。

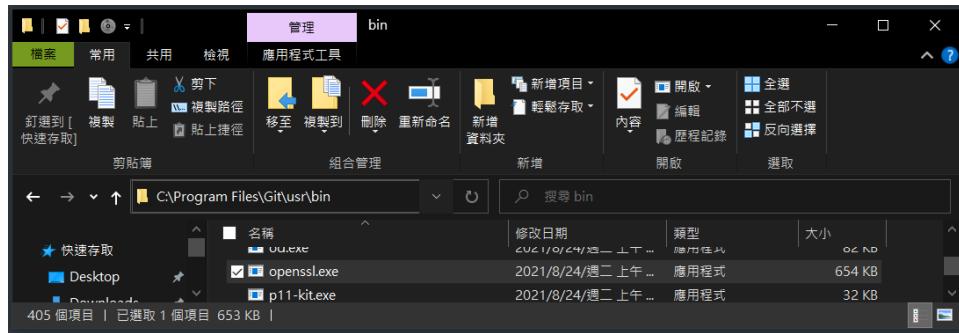


图 1.18 Git 的 Windows 版本内建 OpenSSL

1.4.3.1 OpenSSL 的产生流程与指令结果

在此可以看到 OpenSSL 的产生流程，但本次作业考量到便捷性故考虑使用在 Windows 的 Git 所提供的命令介面。

```

MINGW64/d/git-project/test
USER@Aspire-R7 MINGW64 /d/git-project/test
$ openssl genrsa -out privkey.pem 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....+
e is 65537 (0x010001)

USER@Aspire-R7 MINGW64 /d/git-project/test
$ openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:CN
Locality Name (eg, city) []:Shenzhen
Organization Name (eg, company) [Internet Widgits Pty Ltd]:PKU
Organizational Unit Name (eg, section) []:CS
Common Name (e.g. server FQDN or YOUR name) []:KAN
Email Address []:zz@pku.com

USER@Aspire-R7 MINGW64 /d/git-project/test
$ |

```

图 1.19 OpenSSL 执行

1. 指令

```

openssl genrsa -out privkey.pem 4096
openssl ssh-keygen -t rsa -b 4096 -f privkey.pem

```

2. 结果

```
USER@Aspire-R7 MINGW64 /d/git-project/test
\$ openssl genrsa -out privkey.pem 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....+++++
..++++
e is 65537 (0x010001)

USER@Aspire-R7 MINGW64 /d/git-project/test
\$ openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095
You are about to be asked to enter information that will be incorporated
into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

-----
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:CN
Locality Name (eg, city) []:Shenzhen
Organization Name (eg, company) [Internet Widgets Pty Ltd]:PKU
Organizational Unit Name (eg, section) []:CS
Common Name (e.g. server FQDN or YOUR name) []:KAN
Email Address []:zz@pku.com

USER@Aspire-R7 MINGW64 /d/git-project/test
```

1.4.4 测试范例

在此本作业为此写了 PHP 测试范例，同时也为了 APACHE 准备了设定 CA 凭证的设定档案，当中设定档案的根据皆为 XAMPP 的 APACHE 目录下。在其原目录建立 crt 目录，并放入 cert.conf 与 make-cert.bat 档案，当中 BAT 档案为 Windows 的命令档案，后者会根据前者的设定产生 CA 凭证。最后会在其 crt 目录下看到所产生的 localhost 目录。凭证设定完成后要修改其 apache _ conf _ extra _ httpd-xampp.conf 档案进行设定，最后重启执行。从该目录可以看到产生的 CA 档案。另外 PHP 与 JS 档案则是根据 XAMPP 的预设目录，将其 PHP 与 HTML 档案放于 htdocs 目录下。

1.4.4.1 cert.conf

为根据 XAMPP 的 APACHE 与台湾地区所考量的 CA 设定档案。与 BAT 命令档案同放于 APACHE。

```
[ req ]  
  
default_bits      = 2048  
default_keyfile   = server-key.pem  
distinguished_name = subject  
req_extensions    = req_ext  
x509_extensions   = x509_ext  
string_mask        = utf8only  
  
[ subject ]  
  
countryName          = Country Name (2 letter code)  
countryName_default  = TW  
  
stateOrProvinceName  = State or Province Name (full name)  
stateOrProvinceName_default = Taiwan  
  
localityName         = Locality Name (eg, city)  
localityName_default  = Taipei  
  
organizationName      = Organization Name (eg, company)  
organizationName_default = Personal Reserach  
  
commonName           = Common Name (e.g. server FQDN or YOUR name)  
commonName_default    = localhost  
  
emailAddress          = Email Address  
emailAddress_default  = test@example.com  
  
[ x509_ext ]  
  
subjectKeyIdentifier = hash  
authorityKeyIdentifier = keyid,issuer  
  
basicConstraints      = CA:FALSE  
keyUsage               = digitalSignature, keyEncipherment  
subjectAltName         = @alternate_names  
nsComment              = "OpenSSL Generated Certificate"  
  
[ req_ext ]  
  
subjectKeyIdentifier = hash
```

```
basicConstraints      = CA:FALSE
keyUsage              = digitalSignature, keyEncipherment
subjectAltName        = @alternate_names
nsComment             = "OpenSSL Generated Certificate"

[ alternate_names ]

DNS.1                = localhost
```

1.4.4.2 make-cert.bat

为对应 CONF 的设定档案所设定的命令档案，在 Windows 平台上使用 PowerShell 进行执行，最后产生 CA 档案。

```
@echo off
::set /p domain="Enter Domain: "
set domain=localhost
set OPENSSL_CONF=../conf/openssl.cnf

if not exist .\%domain%\% mkdir .\%domain%\%

..\bin\openssl req -config cert.conf -new -sha256 -newkey rsa:2048 -nodes -keyout\
\%domain%\%server.key -x509 -days 3650 -out \%domain%\%server.crt

echo.
echo -----
echo The certificate was provided.
echo.
```

1.4.4.3 http-s-index.php

为测试 HTTP 和 HTTPS 所特地准备的档案，当中的 PHP 根据路径与资讯进行判断。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>TEST - HTTP & HTTPS</title>
<style type="text/css">
body {
    text-align: center;
```

```

        }
    </style>
</head>
<body>
    <h1>TEST - HTTP & HTTPS</h1>
    <div>
        <?php
            if (!empty($_SERVER['HTTPS']) && ('on' == $_SERVER['HTTPS'])) {
                $uri = 'https://';
                echo " 目前是 HTTPS";
            } else {
                $uri = 'http://';
                echo " 目前是 HTTP";
            }
            $uri .= $_SERVER['HTTP_HOST'];
            phpinfo();
        ?>
    </div>
</body>
</html>

```

1.4.4.4 http-s-index.html

为测试 HTTP 和 HTTPS 所特地准备的档案，当中的 JS 根据路径与资讯进行判断。

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>TEST - HTTP & HTTPS</title>
    <style type="text/css">
        body {
            text-align: center;
        }
    </style>
</head>
<body>
    <h1>TEST - HTTP & HTTPS</h1>
    <div>
        <span id="show"></span>
    </div>
</body>
<script type="text/javascript">

```

```
var ishttps = 'https:' == document.location.protocol ? true: false;
if(ishttps){
    // alert(" 这是一个 HTTPS 请求");
    document.getElementById("show").textContent=" 这是一个 HTTPS 请求";
} else{
    // alert(" 这是一个 HTTP 请求");
    document.getElementById("show").textContent=" 这是一个 HTTP 请求";
}
</script>
</html>
```

1.4.4.5 httpd-xampp.conf

安装完凭证后，需要在 APACHE 对 httpd-xampp.conf 进行设定，最后重启 APACHE。

```
## localhost
<VirtualHost *:80>
    DocumentRoot "C:/xampp/htdocs"
    ServerName localhost

    ServerAlias *.localhost

    RewriteEngine On
    RewriteCond \ %{HTTPS} off
    RewriteRule (.*) https://\ %{SERVER_NAME}/\$1 [R,L]
</VirtualHost>
<VirtualHost *:443>
    DocumentRoot "C:/xampp/htdocs"
    ServerName localhost

    ServerAlias *.localhost
    SSLEngine on
    SSLCertificateFile "crt/localhost/server.crt"
    SSLCertificateKeyFile "crt/localhost/server.key"
</VirtualHost>
```

1.4.5 PHP 的 Web Server 方案

此外本作业针对 XAMPP，同时也准备了备案，其 PHP 内建有 Web Server 的方案，其作用是提供使用者一个方便的测试用途。本作业根据此作为测试的备案。从 PHP 的指令中可以看到，其参数为大写的 S，最后加上 IP 位址与 Port 号。

1. 指令

```
> php -S 127.0.0.1:9988
```

2. PHP 帮助

```
Usage: php [options] [-f] <file> [--] [args...]
php [options] -r <code> [--] [args...]
php [options] [-B <begin_code>] -R <code> [-E <end_code>] [--] [args...]
php [options] [-B <begin_code>] -F <file> [-E <end_code>] [--] [args...]
php [options] -S <addr>:<port> [-t docroot] [router]
php [options] -- [args...]
php [options] -a

-a           Run as interactive shell
-c <path>|<file> Look for php.ini file in this directory
-n           No configuration (ini) files will be used
-d foo[=bar] Define INI entry foo with value 'bar'
-e           Generate extended information for debugger/profiler
-f <file>   Parse and execute <file>.
-h           This help
-i           PHP information
-l           Syntax check only (lint)
-m           Show compiled in modules
-r <code>   Run PHP <code> without using script tags <?..?>
-B <begin_code> Run PHP <begin_code> before processing input lines
-R <code>   Run PHP <code> for every input line
-F <file>   Parse and execute <file> for every input line
-E <end_code> Run PHP <end_code> after processing all input lines
-H           Hide any passed arguments from external tools.
-S <addr>:<port> Run with built-in web server.
-t <docroot> Specify document root <docroot> for built-in web server.
-s           Output HTML syntax highlighted source.
-v           Version number
-w           Output source with stripped comments and whitespace.
-z <file>   Load Zend extension <file>.

args...      Arguments passed to script. Use -- args when first argument
            starts with - or script is read from stdin

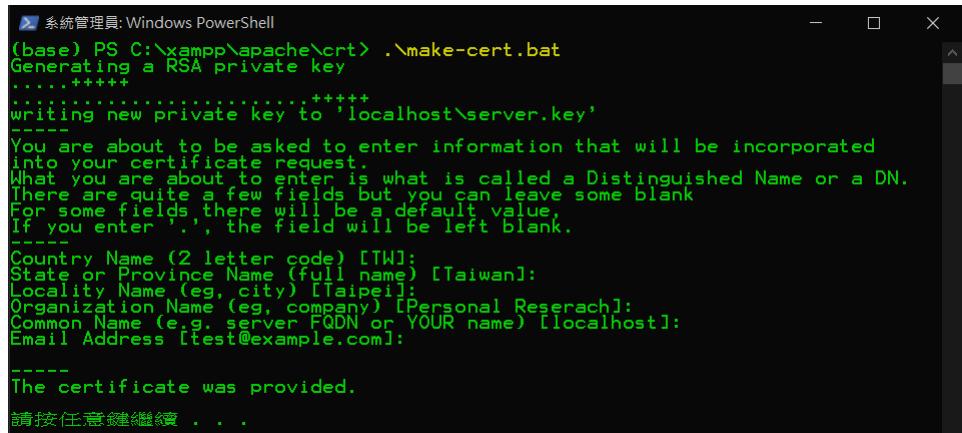
--ini        Show configuration file names

--rf <name>  Show information about function <name>.
--rc <name>  Show information about class <name>.
--re <name>  Show information about extension <name>.
```

```
--rz <name>      Show information about Zend extension <name>.  
--ri <name>      Show configuration for extension <name>.
```

1.4.6 凭证设定

根据本作业前几个小节，可以知道其事前工作的准备。同时可以看到此小节执行 BAT 与 APACHE 产生凭证后，对其进行安装，最后于 Windows 平台检视其结果。同时也看到测试范例显示的资讯。



```
系統管理員: Windows PowerShell  
(base) PS C:\xampp\apache\crt> .\make-cert.bat  
Generating a RSA private key  
.....+++++  
.....+++++  
writing new private key to 'localhost\server.key'  
-----  
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value.  
If you enter '.', the field will be left blank.  
-----  
Country Name (2 letter code) [TW]:  
State or Province Name (full name) [Taiwan]:  
Locality Name (eg, city) [Taipei]:  
Organization Name (eg, company) [Personal Reserach]:  
Common Name (e.g. server FQDN or YOUR name) [localhost]:  
Email Address [test@example.com]:  
-----  
The certificate was provided.  
請按任意鍵繼續 . . .
```

图 1.20 执行 BAT 范例

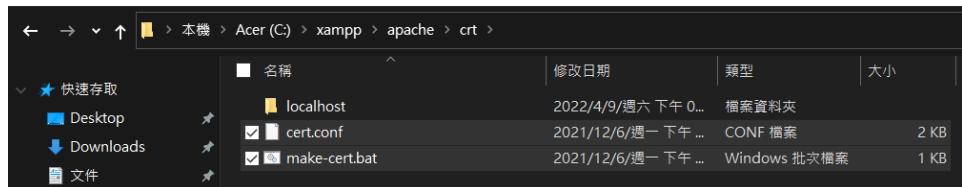


图 1.21 APACHE 结果

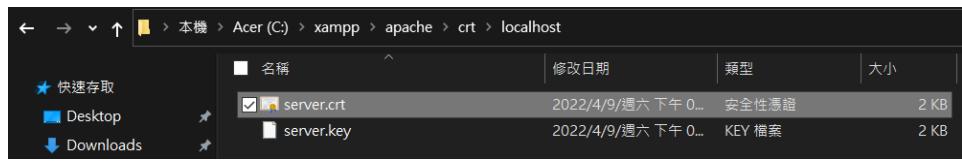


图 1.22 证书

1.5 移除凭证

本节接续前一小节说明在 Windows 平台安装后的凭证移除，首先找 MMC，进入主控台进行移除的设定，同时将凭证选项加入管理，同时设定帐户的权限范围，最后对



图 1.23 安装证书

成功的证书进行移除，移除成功后根据上一节的状况，要将 APACHE 的设定移掉，最后重启 APACHE。

1.6 进行 HTTP 与 HTTPS 分析

本作业此节使用 Mac 的 Wireshark，对 HTTPS 与 HTTP 两个现有的网路服务进行分析，同时使用 Mac 平台的 Wireshark 工具进行判断与分析，从结果来说，很明显得可以从各自的 TCP 串流中可以看到安全性问题。



图 1.24 汇入流程

1.7 WiFi 探针原理与说明

1.7.1 何为 WiFi 探针

所谓的 WiFi 探针技术是根据 WiFi 探测技术来识别用途名为 AP 的无线访问接入点，附近已开启 WiFi 的智能手机或者类似于笔记本，平板电脑等 WiFi 终端，其无需使用者接入 WiFi，WiFi 探针就能够识别使用者的信息。而当使用者走进探针信号覆盖区域内且使用者的 WiFi 设备开启时，其的设备就能被探针探测并发现，所以无论是 Apple 的 IOS 或者是 Google 安卓系统都能轻易检测，并且获取设备的 MAC 地址与相关资讯。其特点如下所示：

- 用户无需参与，无需连接到网络
- Android , IOS 全兼容
- 自动探测区域内手机 MAC 地址
- 手机，平板均能探测

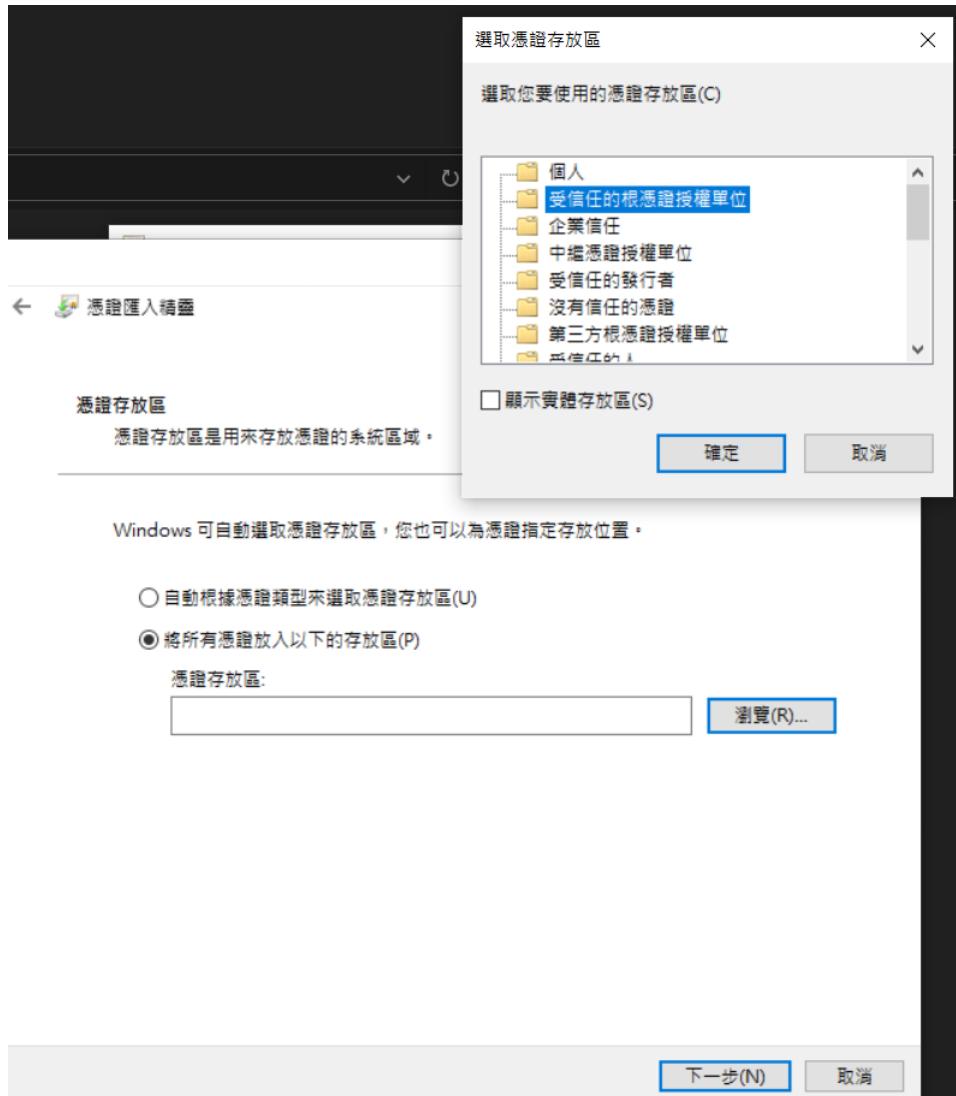


图 1.25 受信任的跟凭证授权单位

1.7.2 工作原理

WiFi 是基于 IEEE 802.11a/b/g/n 协议而成，在标准协议中，定义了名为 AP 的无线接入点和名为站或客户端的 STA 的两种工作模式，同时协议中规定了 BEACON、ACK、DATA、PROBE 等多种无线数据帧类型，在站连接到无线接入点时进行交互的就是数据帧和应答帧、同时 AP 周期性发送 BEACON。在站点没有连接到无线接入点上，手机客户端等站点也会发送 PROBE 帧进行探测询问哪个 AP 是可以连接使用。其 WiFi 探针就是基于各种无线数据帧来抓获手机等 WIFI 客户端的 MAC 地址信息。每个 AP 每隔几十毫秒到几秒不等一定时间会向周围的 STA 和 AP 广播 BEACON 帧，就是告诉周围的 STA 和其他的 AP，自己是 xxxx 的 bssid，发出快来连接我的请求，同时每个 STA，如手机、笔记本等除了默默监听周边 AP 发送的 BEACON 帧以外，还会偷偷发



图 1.26 设定存放

送 PROBE 帧，类似于我是 xxxx 的 MAC 地址，我能够连结的请求。

1.7.3 深入了解

要深入了解 WiFi 探针技术，首先先认识 WiFi 所使用的网络协议，WiFi 采用的是 IEEE802.11 协议集，此协议集包含许多子协议。其中按照时间顺序发展，主要有：(1) 802.11a,(2) 802.11b,(3) 802.11g,(4) 802.11n。同时在网络通信中，数据被封装成了帧，而帧就是指通信中的一个数据块。但是帧在数据链路层传输的时候是有固定格式的，不是随意的封装和打包就可以传输，大小有限制，最小 46 字节，最大 1500 字节，所以必须按照此规则来封装。下面为 802.11 的帧结构，且从上面的结构可以知道，前两个字节为：帧控制字段，控制字段的前 2 BIT 节为协议类型，且目前此值为：0。

- 控制帧 (Control Frame)：如 RTS 帧、CTS 帧、ACK 帧用于竞争期间的握手通信和正向确认、结束非竞争期等。
- 管理帧 (Management Frame)：如 Beacon 帧、Probe Request 帧，主要用于 STA 与 AP 之间协商、关系的控制，而控制行为如关联、认证、同步等。



图 1.27 完成凭证

- 数据帧 (Data Frame)：为承载数据的载体，用于在竞争期和非竞争期传输数据。

1.7.3.1 信标帧

信标帧 (BeaconFrame) 是相当重要的维护机制，主要来宣告某个 AP 网络的存在，同时会定期发送的信标，可让移动 WiFi 设备得知该网络的存在，从而调整加入该网络所必要的参数。而在基础网络里，AP 必须负责发送 Beacon 帧，而 Beacon 帧的所及范围即为基本服务区域。同时在基础型网络里，所有沟通都必须通过接入点，因此 WiFi 设备不能距离太远，否则无法接收到信标。下图可以看到帧格式的说明。

1.7.3.2 管理帧

管理帧 (Probe Request) 为探测请求帧，WiFi 设备将会利用 Probe Request 帧，扫描所在区域内目前有哪些 802.11 网络。下图为其帧格式的说明。

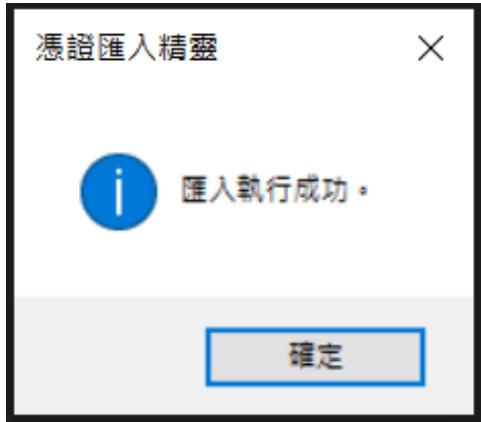


图 1.28 确定凭证

1.7.3.3 数据帧

数据帧 (Data) 为当接入点要送出一个帧给 WiFi 设备但是不必确认之前所传送的信息时，就会使用标准的数据帧。其标准的数据帧并不会征询对方是否有数据待传，因此不允许接收端传送任何数据。无竞争周期所使用的纯数据 (Data-Only) 帧和无竞争周期所使用的数据帧完全相同。

1.7.4 WiFi 探针工作

如图中描述的一样，其的 WiFi 探针其实就是一个 AP，它会定时的向自己的四周广播发送 Beacon 帧，用来通知附近的 WiFi 设备，AP 是存在，就好比它一直在向周围喊着，我在这里，大家快来连接我啊。此时 WiFi 设备如手机，平板电脑等，也会不停的发送着 probe 帧，去寻找附近可用的 AP。而在 probe 帧的介绍中就可以看到 probe 帧包含了设备的 MAC 地址，而当的 AP 接收到 probe 帧之后就获取了此设备的 MAC 地址，而该 AP 就是我们的 WiFi 探针。因此只要在 WiFi 探针覆盖区域内的设备打开着 WiFi，探针就能收集到他的 MAC 地址。

1.7.5 WIFI 探针能采集到的数据

- 设备 MAC 地址
- WiFi 信号强度
- WiFi 信号频道
- 信号帧类型

此外从采集数据图例中可以看到，其记录格式，探针从 MAC 抓取的设备 MAC 设备发送的 WiFi 包的的类型、子类型、信号强度与时间戳。

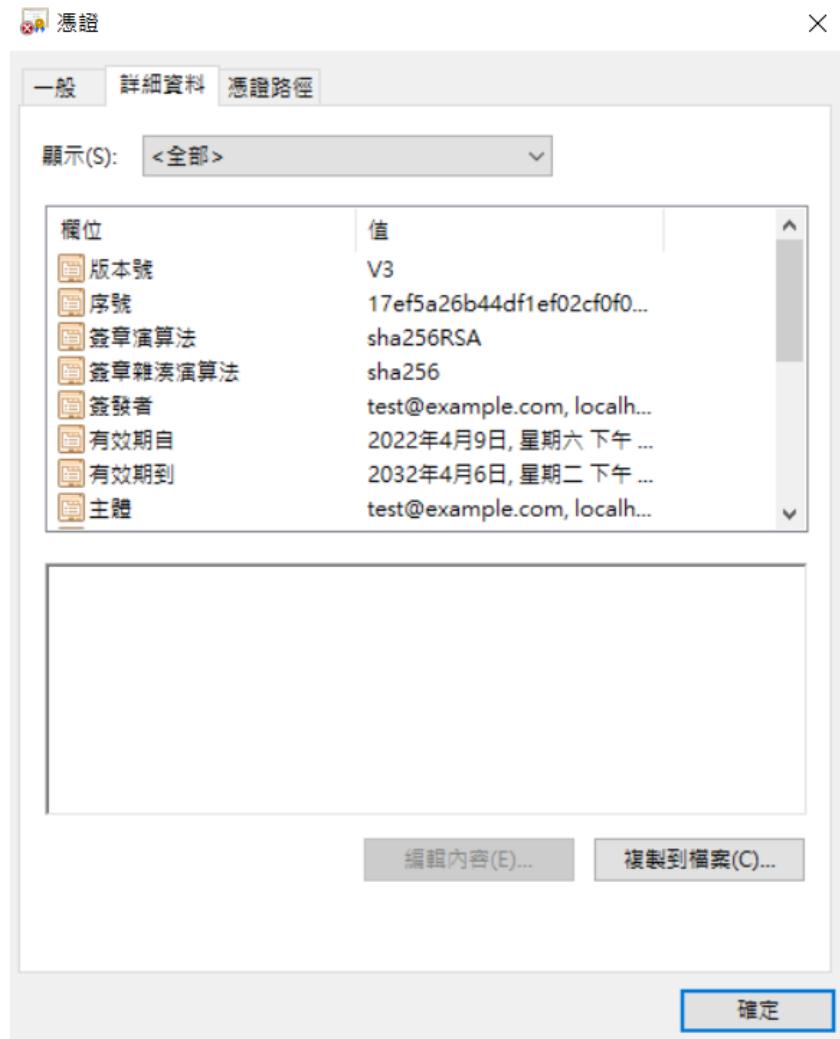


图 1.29 检视凭证资讯

1.7.6 数据释义

- 探针 MAC : 就是探针本身的 MAC 地址。
- 抓取的设备 MAC : 指探针抓取到的 WiFi 信号的发射设备的 MAC 地址，一般为手机。
- 信号强度: 指探针抓取到的 WiFi 信号的强度，其最小值为"-100"，一般来说其值越大表示发射设备离探针越近。
- 设备发送的 WiFi 包的类型: 指探针抓取到的 WiFi 信号的类别，其末位数的值为 0、4、8 时，分别表示抓取到的 WiFi 信号为管理帧、控制帧、数据帧。
- 时间戳: 指探针抓取到 WiFi 信号的时间，如果探针在区域网路内使用而没有接入广域网的话，时间戳可能是不准确的。



图 1.30 执行画面

1.7.7 安全性

在此讨论 WiFi 探针会不会侵犯个人隐私的问题，探针所收集的数据内容我们来看看 WiFi 探针设备究竟会收集什么信息，而从前面的分析已经看到，在不连接 WiFi 的情况下，移动设备只会发送 probe 帧，此时我们并不能通过探针访问网络进行数据传输，探针仅仅只能接收到 WiFi 设备发送的 probe 帧，收集 probe 帧携带的 MAC 地址，所以此时收集到信息是绝对无关用户个人信息和设备上其他信息。同时探针的数据处理由于探针本身设计仅仅是探测周边有些什么设备，因此并不产生大量数据，设计的

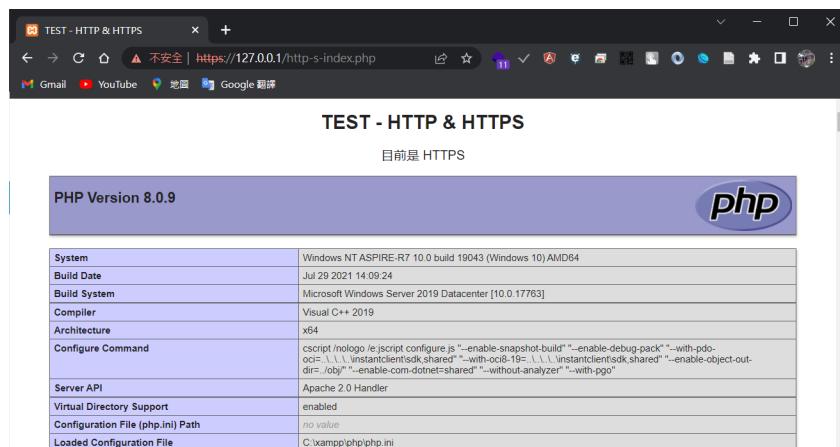


图 1.31 PHP 版本



图 1.32 JS 版本

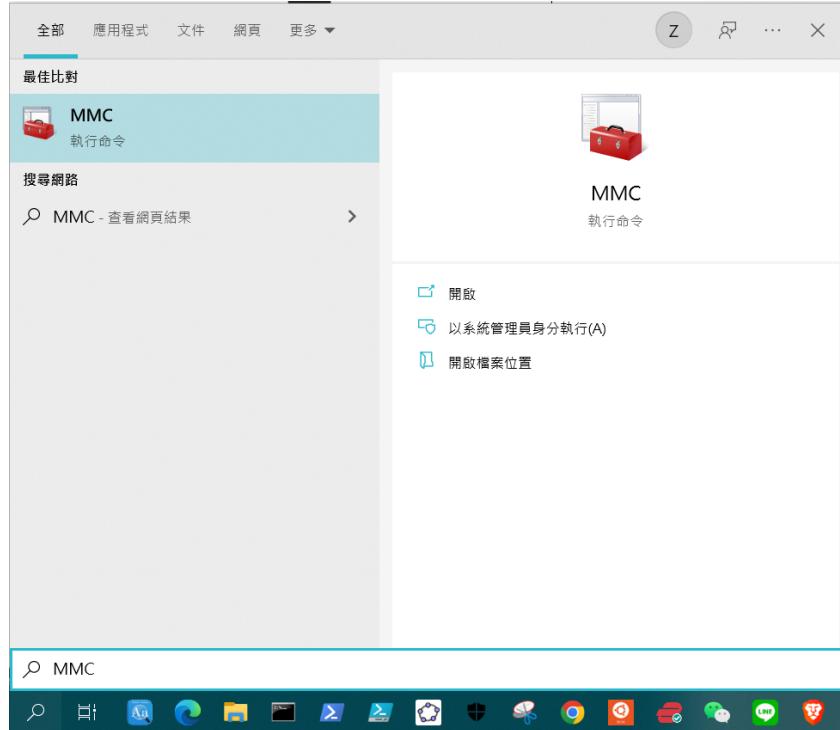


图 1.33 找 MMC

时候就不会将收集到的数据存储在本身，而是通过有线连接直接发送到中心服务器上，这样即使有恶意的人将探针取走，也不能获得探针收集到的信息。同时有线连接也保证数据传输过程不容易通过电磁波的形式被监听和窃取。中心服务器一般都是在 IDC 机房里，而要进入 IDC 机房是需要经过 IDC 层层许可。因而不论是数据的传输还是存储，探针的数据都是安全的。

1.7.8 用何种设备

WiFi 探针设备和普通的无线路由器很接近，实际普通路由器就能做，只修改无线路由器的驱动部分，直接在驱动中抓取周边手机的信息。同时许多厂商也推出了专用的 WiFi 探针设备，集成为管理平台，更方便的管理收集的信息等特性。



图 1.34 主控台移除

1.8 WiFi 探针与移动轨迹

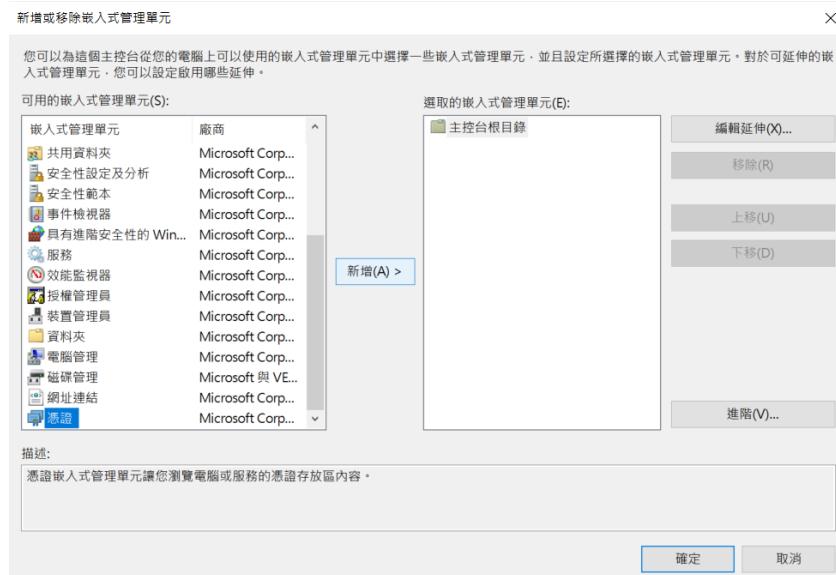


图 1.35 将凭证加入管理



图 1.36 设定帐户

北京大学密码编码学与网络信息安全期末作业

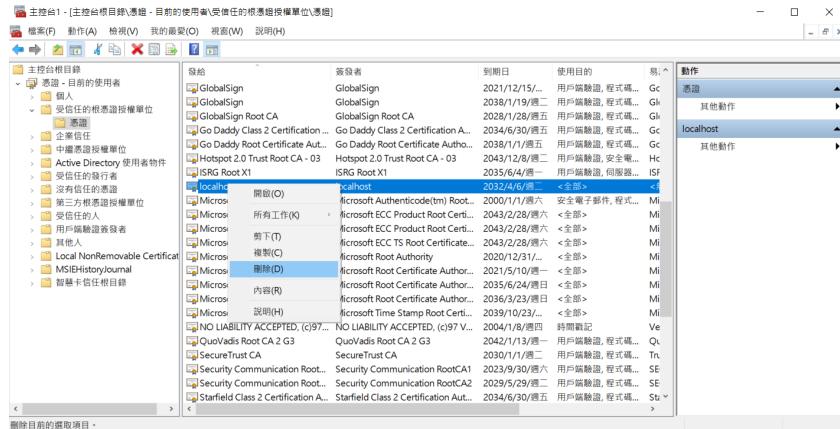


图 1.37 移除

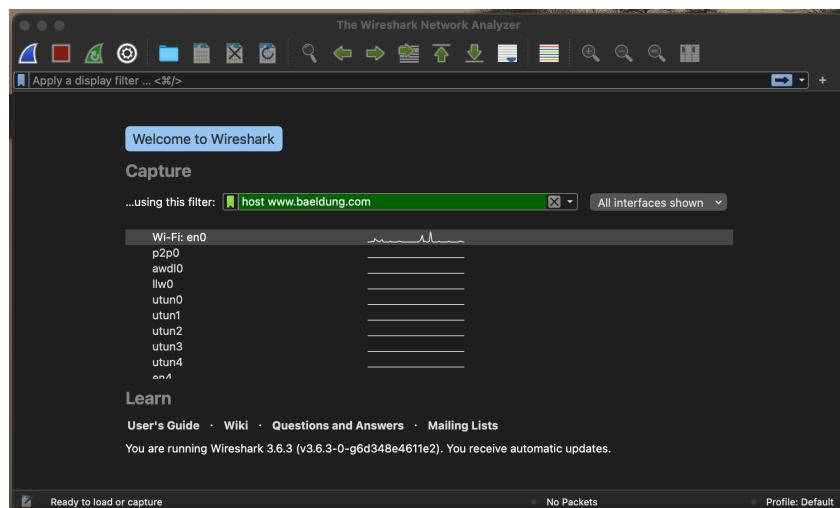


图 1.38 Mac 的 Wireshark

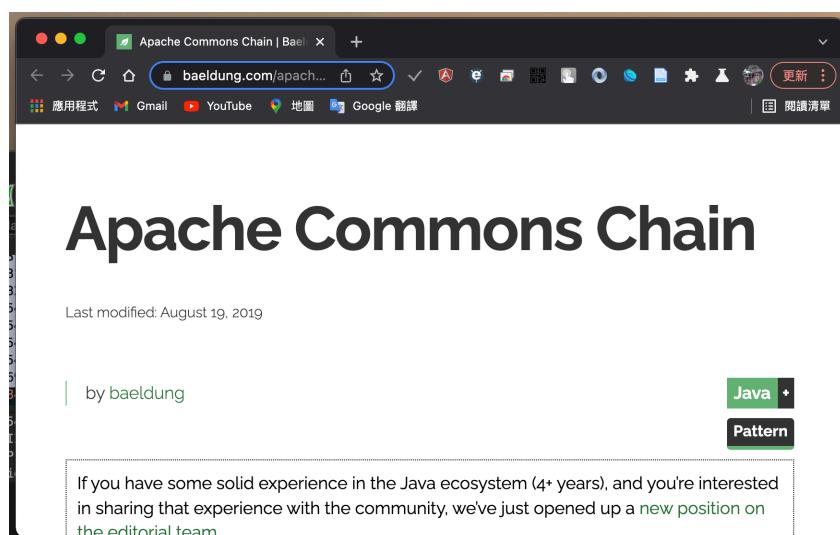


图 1.39 HTTPS 服务范例

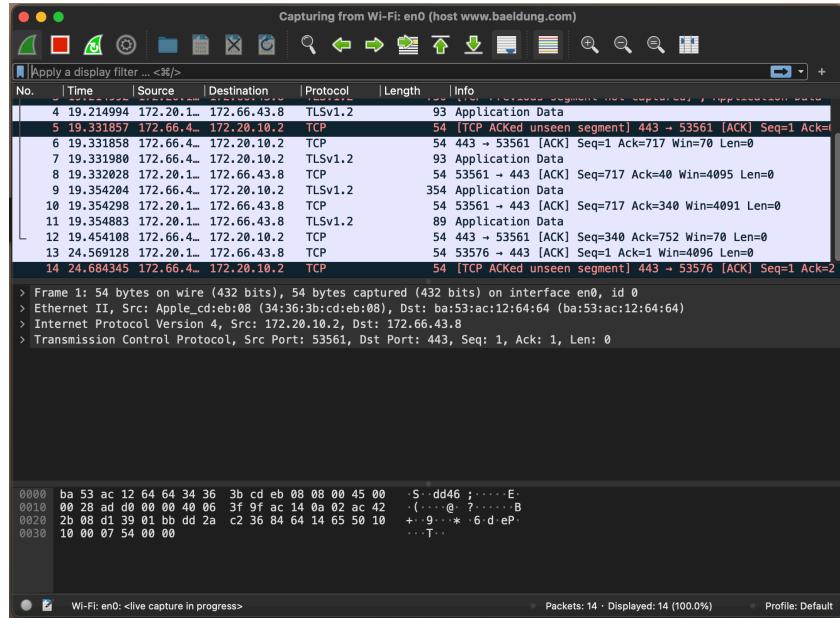


图 1.40 Wireshark 判断 HTTPS

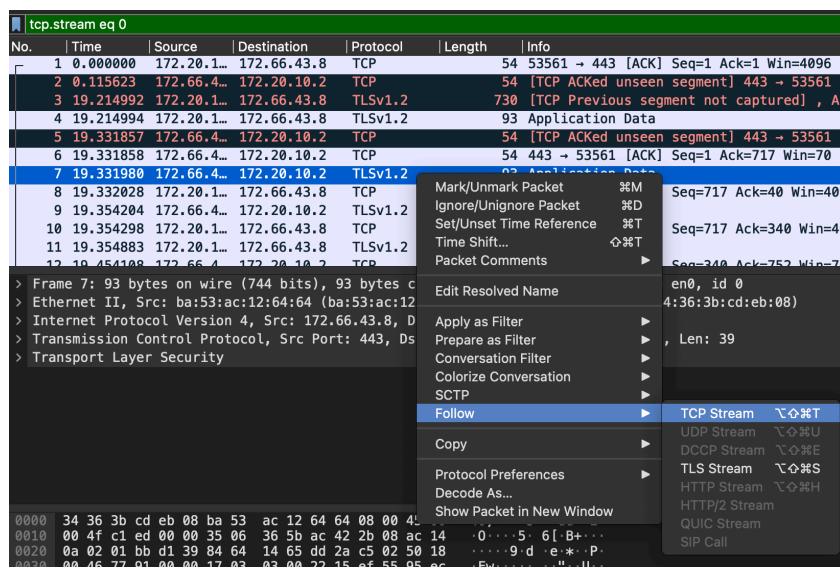


图 1.41 找 TCP 串流

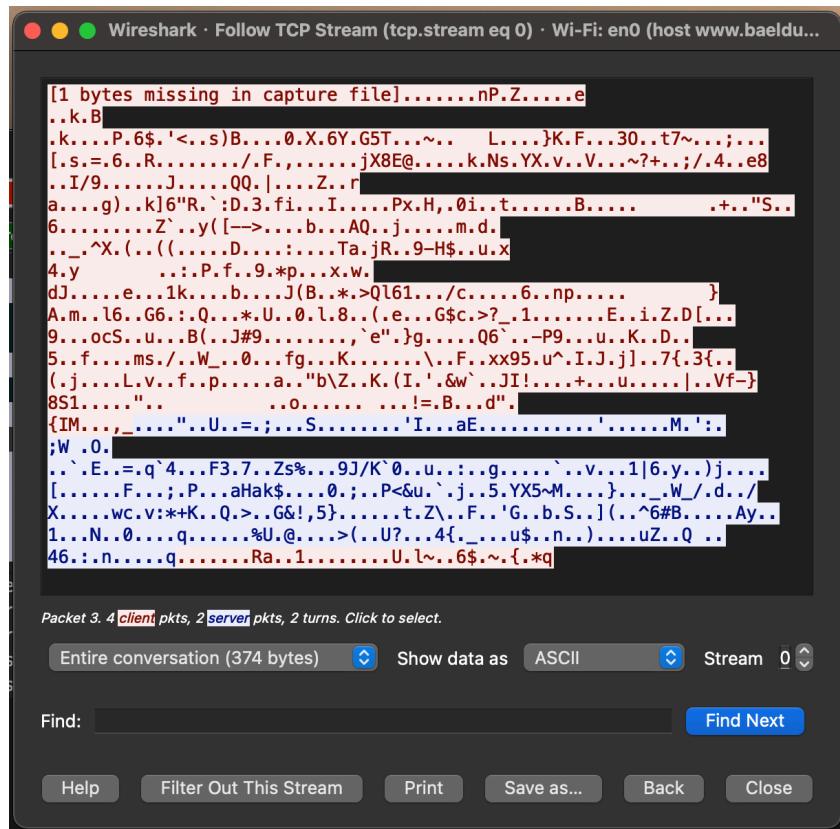


图 1.42 HTTPS 结果

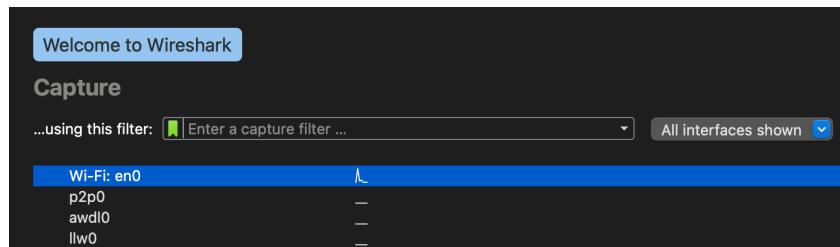


图 1.43 测试 HTTP

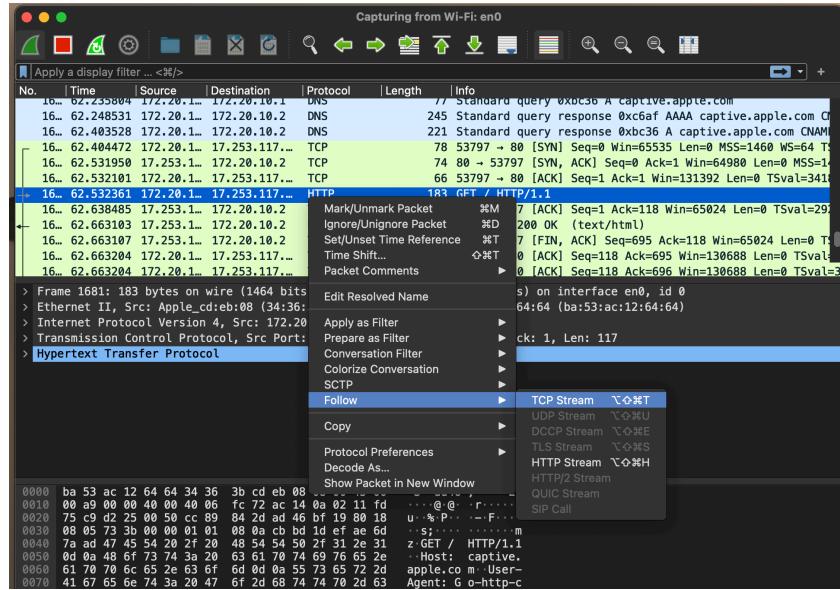


图 1.44 Wireshark 判断 HTTP

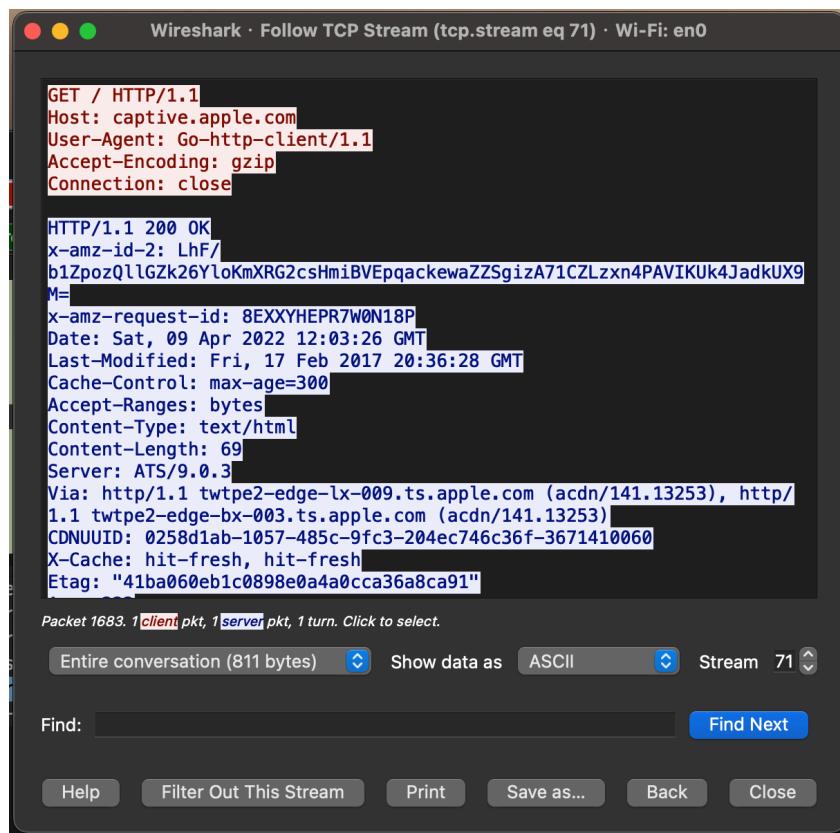


图 1.45 HTTP 结果

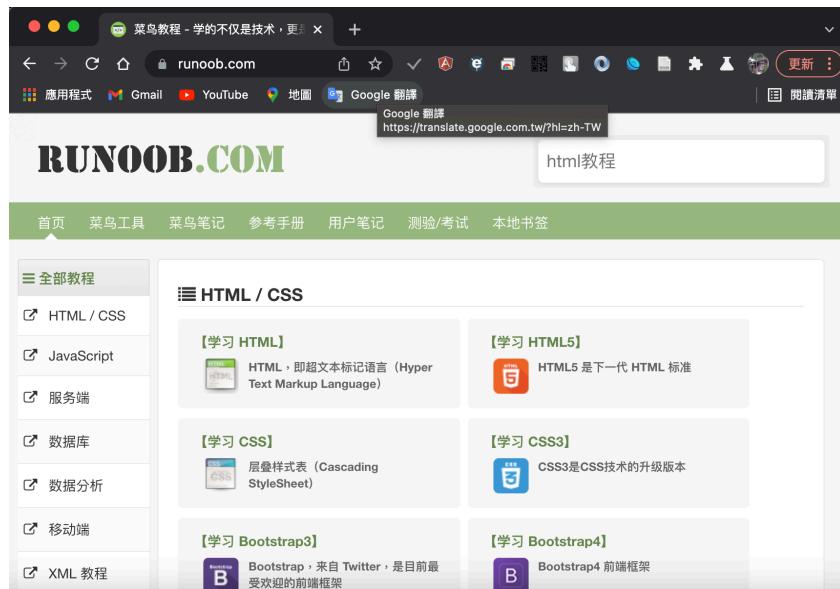


图 1.46 HTTP 服务范例

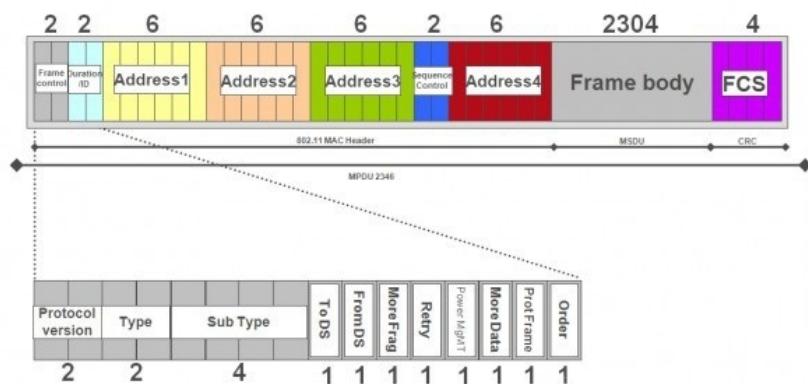


图 1.47 802.11 的帧

字段名	描述	长度(字节)
Frame Control(帧控制)字段	描述与控制MAC帧相关信息	2
Duration字段	计算帧持续时间的作用	2
Destination Address	MAC帧的目的地址	6
Source Address	MAC帧的源地址	6
BSSID(基本服务集ID)	用于过滤收到的MAC帧(在基础型网络里为工作站所关联的接入点的MAC地址)	6
Sequence Control(顺序控制字段)	用来重组帧片段以及丢弃重复帧	2
帧主体	用以传递上层信息	0-2312
FCS(帧校验序列)	验证传来的帧是否有误	4

图 1.48 帧控制字段

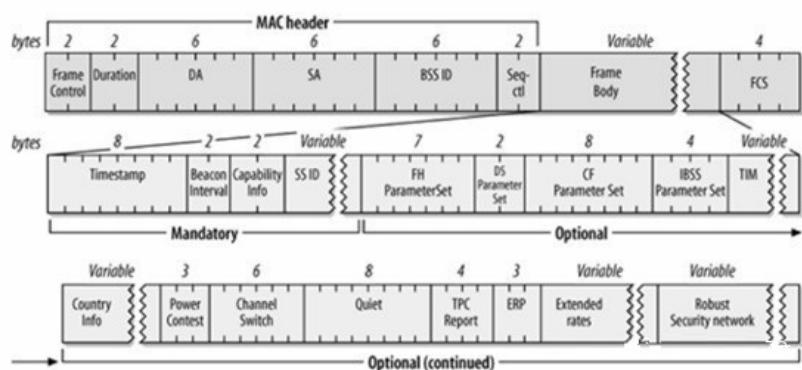


图 1.49 信标帧

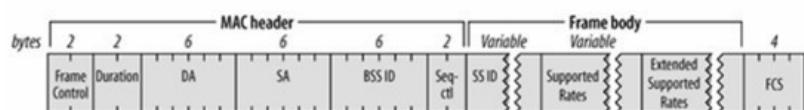


图 1.50 管理帧



图 1.51 WiFi 探针工作

```
COM3 - PuTTY
5c:cf:7f:dc:7e:58|60:01:94:0F:BD:EC|28:6C:07:5A:91:16|-65|0
5c:cf:7f:dc:7e:58|54:14:73:72:84:B4|28:6C:07:5A:91:16|-76|1
5c:cf:7f:dc:7e:58|54:EE:75:60:4C:98|28:6C:07:5A:91:16|-49|0
5c:cf:7f:dc:7e:58|60:01:94:0F:BD:EC|28:6C:07:5A:91:16|-55|0
5c:cf:7f:dc:7e:58|7C:7D:3D:B6:76:57|28:6C:07:5A:91:16|-101|0
5c:cf:7f:dc:7e:58|60:01:94:0F:BD:EC|28:6C:07:5A:91:16|-65|0
5c:cf:7f:dc:7e:58|3C:46:D8:DA:43:79|28:6C:07:5A:91:16|-57|0
5c:cf:7f:dc:7e:58|A4:71:74:4B:F2:0F|28:6C:07:5A:91:16|-87|1
5c:cf:7f:dc:7e:58|3C:46:D8:DA:43:79|28:6C:07:5A:91:16|-56|0
5c:cf:7f:dc:7e:58|74:D4:35:20:86:A1|D0:C7:C0:3E:26:62|-92|0
5c:cf:7f:dc:7e:58|74:27:EA:F6:43:55|D0:C7:C0:3E:26:62|-86|0
5c:cf:7f:dc:7e:58|5C:E0:C5:1C:55:25|C8:E7:D8:D8:B3:82|-54|0
5c:cf:7f:dc:7e:58|B8:86:87:5E:29:C7|FF:FF:FF:FF:FF:FF|-88|0
5c:cf:7f:dc:7e:58|7C:7D:3D:B6:76:57|28:6C:07:5A:91:16|-99|0
5c:cf:7f:dc:7e:58|60:01:94:0F:BD:EC|28:6C:07:5A:91:16|-67|0
5c:cf:7f:dc:7e:58|74:D4:35:20:86:A1|D0:C7:C0:3E:26:62|-94|0
5c:cf:7f:dc:7e:58|74:27:EA:F6:43:55|D0:C7:C0:3E:26:62|-82|0
5c:cf:7f:dc:7e:58|68:3E:34:64:11:20|28:6C:07:5A:91:16|-89|1
5c:cf:7f:dc:7e:58|5C:E0:C5:1C:55:25|C8:E7:D8:D8:B3:82|-49|0
5c:cf:7f:dc:7e:58|68:3E:34:64:11:20|28:6C:07:5A:91:16|-74|0
5c:cf:7f:dc:7e:58|3C:46:D8:DA:43:79|28:6C:07:5A:91:16|-55|0
5c:cf:7f:dc:7e:58|7C:7D:3D:B6:76:57|28:6C:07:5A:91:16|-97|1
5c:cf:7f:dc:7e:58|54:A0:50:7D:02:5F|28:6C:07:5A:91:16|-52|0
```

图 1.52 WIFI 探针能采集到的数据

第二章 联邦学习与隐私安全

第三章 量子计算机与信息安全对密码学的影响

在本章量子计算机与信息安全对密码学的影响的部分，本作业尝试整理近来量子计算机的发展对信息安全领域的影响的研究与资讯，包括对密码学领域所带来的挑战与对策。当中所谓的量子计算与量子密码是根据量子力学与效应所发展的计算技术跟与相应发展的密码技术，其前者根据利用运行上千的量子位元跟量子算法的量子计算机的特性，在将来会直接挑战目前传统的资讯安全算法，从而导致目前广泛与大两使用的 RSA 密码手段被破解的可能，同时也使原本高强度的密码技术不如以往。同时量子通信领域中的量子密钥分发技术也大大的改变了过往的模式，其于 1984 年由 Bennett 和 Brassard 两人所提出的第一个量子密钥分发协议，开始了量子密码学的研究，1994 年 Shor 运用量子 Fourier 变换的特性，去设计了第一个实用的量子算法，其原理则是在多项式时间内对大整数进行因子分解，另外于 1996 年 Grover 提出了量子搜索算法，该法能够对无结构数据进行二次加速。此两个方法展现了量子计算的优越性，同时还对传统基于数学困难问题的密码学体制造成威胁。本作业第一节为量子计算与通信的意义，第二节为量子力学的数学，第三节为量子计算的整理，第四节为量子算法的整理，第五节为量子通信与量子密码研究进展，最后进行结论。

3.1 发展量子计算与通信的意义

对目前计算机发展影响深巨的在于电晶体与积体电路的发明，其相关发展带动目前消费者电子产品、智慧型手机、平板与个人电脑等不同嵌入式设备，在其市场的蓬勃发展，间接带动网际网路的发展。其中密码学演算法下的公钥密码演算法为基础，其保障人们可以在网路上安全的进行互动。

上述的发展皆从量子力学中发展而来，而所谓的量子力学则是研究微观尺度的量子物理理论，根据量子力学的原理，进而发展了半导体物理的理论，从而制造出电晶体与积体电路，同时量子通信等领域深入研究也导致新的改变。另外随着近年该领域的不断发展，也出现如量子时钟、量子成像、量子传感、量子计算与通信等具有巨大应用潜力的技术应用，而当中与资讯安全相关的为量子计算和量子通信。

3.1.1 量子计算的部分

根据摩尔定律，在成本不变的情况下，每隔 18 至 24 个月，在积体电路上可容纳的电晶体密度会增加一倍。虽然其定律在过往半世纪以来相当有效，但随着电晶体大小缩小到原子大小，半导体的会面对无法散热跟漏电的问题，导致无法运作。所以若

想继续发展更大计算能力的计算机，则可能要另辟蹊径，当中使用量子算法的量子计算机很可能即为这当中的具有希望的方案之一，由此来满足人们对于此的需求。

此外另一个重点是量子计算机对资讯安全的影响在于，一旦通用型的量子计算机发展到可以处理的量子位元数目达到一定规模后，现有传统的密码学体制将会被量子的 Shor 演算法轻松地破解，当下现在网际网路上进行的通信和金融交易等活动的安全皆将会受到直接的挑战，其解决办法是将这些演算法更换为可以抵抗量子计算攻击的密码演算法。

3.1.2 量子通信的部分

量子通信技术领域中相对成熟的应用是量子密钥分发及其相关技术，而传统密码技术的一大困难是在于无法真正保证通信双方共享的密钥的不分是绝对安全，也因此经典的位元十分容易复制且不易被察觉，所以通信双方无法真正确认共享密钥是否已被恶意攻击者获取。而量子通信技术利用光子的量子特性，在传输量子信息的过程中，一旦受到窃听和破坏，必然会引起干扰，这样就可以通过物理的手段将攻击行为检测出来，保证信息传输的安全，从根本上解决点对点的密钥分发问题，确保点对点通信的安全。也因此量子密钥分发有时也被称为量子密码，量子密码之名意义为可以抵抗量子计算攻击的后量子密码演算法的区分。其量子密钥分发所可以证明的安全是经典的保密通信无法做到的，与因此量子密钥分发是未来保密通信的重要发展方向。

3.2 量子力学的数学

量子力学理论给出了研究物理系统规律的数学框架，通过这个框架，物理世界和量子力学的数学描述得以联系起来。量子力学所描述的微观世界，与人们熟悉的宏观世界有很大的不同，从而显得奇妙而神秘，但如果将其放在量子力学的数学视角下，则不过是普通的 Hilbert 空间向量的运算与变换，本节对量子力学的基本数学框架进行简要介绍。

一个孤立的物理系统对应一个复 Hilbert 空间，该空间称为系统的状态空间，而系统的状态由 Hilbert 空间中的向量描述，为了描述和运算方便，一般用英国理论物理学家狄拉克引入的狄拉克符号表示，如 $|\varphi\rangle$ ，在此需要注意的是，该数学描述并没有给出 Hilbert 空间的具体形式。

孤立物理系统的状态随时间的变化由 Hilbert 空间中的酉变换描述，如果系统在 t_1 时的状态为 $|\varphi_1\rangle$ ， t_2 时变为 $|\varphi_2\rangle$ ，则存在一个仅与 t_1 和 t_2 有关的酉变换 U ，使得 $|\varphi_2\rangle = U|\varphi_1\rangle$ ，同样需要注意的在于此一数学描述没有给出酉变换的具体形式。

复合物理系统使用张量积描述，即复合系统的状态空间表示为各分系统状态空间

的张量积。这一描述也提供了用分系统构造复合系统的方法。

对量子系统的测量与经典测量有很大的不同，量子测量由一组满足完备性的测量算子 M_m 描述，其中， m 表示可能的结果。如果测量前系统为 $|\varphi\rangle$ ，则测量后以概率 $\langle\varphi|M_m^\dagger M_m|\varphi\rangle$ 得到结果 m ，且系统变为 $M_m|\varphi\rangle/\sqrt{\langle\varphi|M_m^\dagger M_m|\varphi\rangle}$ 。

3.2.1 量子力学基本概念和原理整理

在此小节会针对量子力学的基本概念和原理进行整理，当中包含了量子比特、态叠加原理、不确定性原理、未知量子态不可克隆、非正交量子态不可区分。

3.2.1.1 量子比特

量子比特是二维 Hilbert 空间中的一个单位向量。在取定 2 个正交的基态 $|0\rangle$ 和 $|1\rangle$ 后，一个量子位元可表示为 $\alpha|0\rangle+\beta|1\rangle$ ，其中 α 和 β 是复数，且满足 $|\alpha|^2+|\beta|^2=1$ 。其多个量子比特可以复合，如果每个量子位元取定基 $|0\rangle$ 和 $|1\rangle$ ，则 n 个量子位元复合后的系统可表示为 $\sum_{i=0}^{2^n-1} \alpha_i|i\rangle$ ，其中 α_i 为复数，且 $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$ ，另外 $|i\rangle$ 为复合系统的基，称为计算基。而复合量子比特是量子计算的基础。

在复合量子比特中，如果其状态向量不能表示为各量子比特状态向量的直积形式，则称该系统处于纠缠态。通俗来讲，处于纠缠态的系统，各子系统状态不能分开。纠缠态在量子计算与量子信息中起着重要作用，常用的纠缠态有两粒子纠缠的 Bell 态、三粒子纠缠的 GHZ 态等。

3.2.1.2 态叠加原理

如果 $|\varphi_i\rangle (i = 0, 1, 2, \dots, n)$ 是系统的可能状态，则其线性叠加 $\sum_{i=0}^n \alpha_i |\varphi_i\rangle$ 也是系统的可能状态。从数学的观点看，因为系统状态是 Hilbert 空间中的向量，而 Hilbert 空间是线性空间，所以线性叠加性成立。

在量子计算中，一般会使用计算基的叠加态，由于酉算子是线性算子，酉算子在叠加态上的作用，相当于在各计算基上作用的叠加，从而获得真正意义上的并行计算能力。

3.2.1.3 不确定性原理

不确定性原理是量子系统的内在属性，与测量设备的精度以及测量设备对系统的扰动无关。原理中指出如果 2 个力学量所对应的算符不对易，则不能同时确定这 2 个力学量。如在测量光子偏振状态的过程中，线偏振状态和圆偏振状态不能同时确定，这也是 BB84 协议工作的理论基础。

更一般地，测量一个量子态时，能否获得精确测量结果依赖于该量子态是否为测量算符对应的本征态，如果该状态是测量算符对应的本征态，则可得到精确测量结果，否则无法得到精确测量结果。

3.2.1.4 未知量子态不可克隆

1982 年 Wootters 和 Zurek 首次提出了著名的量子不可克隆定理，在量子力学中不存在一个对未知量子态精确复制的物理过程，即未知量子态不可能精确复制，使得每个复制比特和初始量子比特完全相同。1986 年，Yuen 推广了量子不可克隆定理，指出表示克隆过程的酉变换使得 2 个量子态被克隆，当且仅当它们相互正交，即非正交态不可克隆。从上述可知未知量子态的不可克隆性，虽然对量子计算中比特复制造成一定困难，但对量子密码学中安全体制的设计提供了重要保障。

3.2.1.5 非正交量子态不可区分

对于 2 个非正交量子态，没有一个物理过程可对其进行完美区分。这是由未知量子态的不可克隆性决定的。例如对于 2 个量子位元，如果它们是非正交的，则任何操作或测量都不能将它们完美区分开来，总是会产生一些错误的结果。同不可克隆性一样，非正交量子态的不可区分性给量子计算带来了很多困难，但在量子密码学中的应用，有着举足轻重的价值。

3.3 量子计算的整理

运用量子位元的特性来进行计算的方式被称为量子计算。量子计算在很多方面都与传统的经典计算机的计算运作方式有着明显不同，这使得前者量子计算机在某些方面相较后者具有巨大的优势。

3.3.1 量子比特及其特性

众所周知，经典的计算机系统使用二进制的位元来编码信息，每个位元可以取值为 0 或者 1，但是不能同时取 0 和 1。量子技术中采用的是量子位元 (qubit)，其量子位元不但可以表示 0 或者 1，而且可以同时有 0 的成分和 1 的成分，形成所谓的叠加态，如表中所示。当量子位元状态表达式中的某些系数为 0，则可以表示经典位元。通常在处理量子位元时可以同时对 0 和 1 进行操作，这种操作具有内在的并行特性，可以加快运算的速度。

一个量子位元可以同时保存的状态个数为 2，这种指数增长的规律正是量子计算的根本威力来源。另外多个量子位元之间的量子纠缠是量子系统的重要特性，也是一

种十分重要的计算资源，一些经典方式无法实现的效果，如隐形传态，可以利用量子纠缠的方式进行来实现。实验上实现量子比特的方式有多种，例如超导环路、囚禁离子、量子点、拓扑量子比特和金刚石空位等，这些方式各有利弊，都仍在不断发展。

位元类型与数量	可表示状态
1 个 位元	0 或 1
2 个 位元	00 或 01 或 10 或 11
1 个 量子位元	0, 1 的线性组合，表示为 $\alpha 0\rangle + \beta 1\rangle$
2 个 量子位元	00, 01, 10, 11 的线性组合，表示为 $\alpha_{00} 00\rangle + \alpha_{01} 01\rangle + \alpha_{10} 10\rangle + \alpha_{11} 11\rangle$

3.3.2 量子计算机的类型

设计量子计算机的技术路线主要有两种类型，共分通用型和专用型，这两种类型的典型代表的对比如下表中所列。经典计算机的基础部件是逻辑门电路，通过对位元进行逻辑运算来实现计算功能。与这种架构类似，对于量子位元来说，其也存在相应的量子逻辑门。量子晶片上如果能制备出所需的量子位元以及对量子位元进行操作和运算的量子逻辑门，就可以制备成通用型量子计算机。这种量子计算机通过控制量子位元依次通过不同的量子逻辑门来达到编程的目的，通过组合不同的量子逻辑门可以实现各种量子演算法。但是由于量子位元十分脆弱，很容易失去叠加和纠缠特性，因此通用型量子计算机建造难度很大，尤其是可以处理多个量子位元的情形。当前实验室环境可以做到同时处理 10 个左右的量子位元。另外 IBM 公司于 2016 年 5 月开放的量子计算平台是可以处理 5 个量子位元的通用型量子计算机，这种规模的量子计算机难以满足实用运算的要求，更倾向于技术的展示，现在所有人都可以通过 Web 访问的方式来体验量子编程和量子计算。

与通用型量子计算机不同，总部位于加拿大的 D-Wave 系统公司制造了一种可以运行量子退火算法的专用型量子计算机，它的量子位元之间的连接方式是固定的，但是参数可调，因此制造难度较小。D-Wave 量子计算机已经实现商用，最新的产品 2000Q 集成了 2048 个量子位元。然而 D-Wave 的量子计算机只能专门解决一类特殊的最优化求解问题，由于优化问题在机器学习领域的重要性，D-Wave 的产品已经引起了包括 Google 在内的许多公司的注意。Google 利用 D-wave 的产品发现在处理某些特定问题上量子计算机的确具有很大的优势，比传统计算机单核运行速度快 1 亿倍。因此 D-Wave 的产品可以称为量子退火机。参照通用型量子计算机的名词定义，其 D-Wave 的产品可

以说是属于一种专用型量子计算机，由于其应用局限性，目前还无法引入到与密码计算相关的领域中。

差异	IBM	D-Wave
组成与运行方式	量子位元通过量子逻辑门进行操作	由量子位元形成二维结构，其量子位元之间的作用强度参数可调
编程方式	通过控制量子逻辑门的操作顺序进行编程	通过设定参数，经过绝热演化得到特定问题的解
目前具有的量子位元数	5 个	最新产品为 2048 个
扩展难度	高	较低
是否商用	否	是

3.4 量子算法

量子计算机的研制之所以受到各方重视的一个诱因是 1994 年 Peter Shor 发现了可以快速对大整数分解其素数因子的量子演算法，这个演算法可以破解现在广泛使用的 RSA、椭圆曲线等公钥密码。例如，RSA 密码算法的安全性需要基于以下事实，将某一具有两个素因子的长整数的素因子找出，当这个长整数很大时，是极其困难的，同时运算代价很大以至于实际上不可行。这个事实在没有发现量子 Shor 演算法之前被认为是正确的，因为最好的分解整数的经典演算法，其复杂度也是亚指数形式的，这也是 RSA 算法可以被放心使用的原因。但是在量子 Shor 演算法发现之后，这个结论在量子算法领域不再成立。Shor 演算法是一个多项式复杂度演算法，若有可以执行 Shor 算法的通用型量子计算机一旦制造成功，同时它能够处理的量子位元数目足够多，则现在广泛使用的 1024 位元和 2048 位 RSA 演算法便不再安全。这意味着网路上的信息安全保护基本手段完全失效，这将会是一个致命的挑战。理论上 Shor 演算法分解一个位元的长整数大约需要 2 个量子位元，这意味着破解 RSA-1024 需要一台可以同时处理 2048 个量子位元的量子计算机。

就技术现状来说，通用型量子计算机的制造难度很大，到目前为止经过 20 年左右的发展，也只制造出处理大约 10 个量子位元的通用型量子晶片。当前实验室环境中可以用 Shor 算法分解的最大整数是 21，因此 RSA 等密码算法当前还是安全的，但量子的挑战仍然持续存在。在可以想象的未来某天，科学家发现了一种实现量子位元和量子逻辑门的方法，且容易进行大规模扩展时，也许量子晶片也会有类似摩尔定律那样的快速发展。那时破解密钥长度很大的 RSA 算法将会变得不再困难。此情况的出现可能需要十年、几十年、甚至更久，目前也已成为各大科研机构追逐的目标。解决 Shor 算法威胁的更好办法是将 RSA 等公钥密码算法替换为可以同时抵抗经典算法和量子算

法攻击的密码算法，此类演算法也被称为后量子密码算法或抗量子攻击密码算法。经典算法和量子算法都无法在多项式计算复杂度内解决的困难且可用于构造后量子密码算法，这些算法早已开始研究，并且有多个候选方案，其中包括基于格困难问题构造的密码算法等。

在对称密码的量子威胁方面，量子 Grover 算法是研究热点，该算法可以在无序的数据中进行快速搜寻。在一个无序的数据中搜寻某个数据是否存在，其经典方法是逐个元素进行判断，比如使用穷举法，平均查找次数为 $n/2$ ，复杂度为 $O(n)$ 。但是量子 Grover 算法的复杂度为 $O(\sqrt{n})$ ，相比传统方法有平方加速的效果。该算法可以辅助进行分组算法或哈希算法的穷举攻击，这种攻击一旦实现，意味着现有对称密码算法的安全强度将大幅降低。若要达到之前的密码强度，需要将对称密钥长度加倍，例如需要将 AES-128 升级为 AES-256。Shor 算法和 Grover 算法对密码学算法的威胁和解决办法总结在下表中列出。

密码算法	类型	用途	大规模通用型 量子计算机的影响
AES, SM4	对称算法	加密	需要更长 的密钥
SHA-2, SHA-3, SM3	杂凑算法	计算信息摘要	需要更长的摘要输出
RSA	公钥算法	签名、 密钥协商	遭受直接攻击； 需更换为后量子密码算法
DSA 数字签名算法 ECDSA, ECDH, SM2 等椭圆曲线算法	公钥算法	签名、 密钥交换	遭受直接攻击； 需更换为后量子密码算法

量子计算通过量子逻辑门和连线构造量子线路实现。而量子逻辑门在数学上由复 Hilbert 空间的酉变换描述。1985 年 Deutsch 引入了量子线路模型。1995 年，Barenco, Bennett 和 Cleve 等人证明了单量子比特门和受控非门 (controlled-NOT, CNOT) 的通用性，为量子线路模型提供了完善的理论保证。酉变换是可逆的，即量子逻辑门是可逆的，然而经典逻辑门大多不可逆，这些不可逆的经典逻辑门没有对应的量子逻辑门，所以不能用量子线路直接模拟经典线路。幸运的是我们可以用可逆的 Toffoli 门实现经典逻辑门，从而等价地构造经典线路。Toffoli 门是可逆门，可以用量子逻辑门实现，从而使得量子线路可以间接模拟经典线路，在量子道路上实现任何经典计算。

量子计算中，常用的量子逻辑门有 Pauli-X 门、Pauli-Y 门、Pauli-Z 门、Hadamard 门、相位门、受控门、交换门等。量子态的叠加性决定量子计算具有并行性的特征，它可以同时计算一个函数在许多点处的函数值，使得量子计算的能力从本质上超越经典计算的能力。然而由于量子测量的特点，无法直接从叠加态中直接抽取信息，这大大限

制了量子计算的能力。虽然如此，还是可以通过巧妙的设计，有效利用量子计算的优越性，为计算问题加速，这也是量子算法设计的一个重要特点。从早期的 Deutsh-Jozsa 算法和 Simon 算法始起，开始出现了许多量子算法，其中最具有里程碑意义的是 Shor 算法和 Grover 算法，它们分别使用了量子 Fourier 变换和量子搜索，本节对其思想进行简要介绍。

3.4.1 量子 Fourier 变换

量子 Fourier 变换是定义在标准正交基 $|0\rangle, |1\rangle, \dots, |2^n - 1\rangle$ 上的一个酉变换，在这些基态上的作用为 $|j\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i j k / 2^n} |k\rangle$ ，通过代数计算，变换后的结果可以表示为

$$\begin{aligned} \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k / 2^n} |k\rangle = \\ \frac{1}{2^{n/2}} (|0\rangle + e^{2\pi i 0 \cdot j_n} |1\rangle) \cdots (|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \cdots j_n} |1\rangle) \end{aligned}$$

量子 Fourier 变换是定义在标准正交基 $|0\rangle, |1\rangle, \dots, |2^n - 1\rangle$ 上的一个酉变换，在这些基态上的作用为 $|j\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i j k / 2^n} |k\rangle$ ，通过代数计算，变换后的结果可以表示为

其中 $j_1 j_2, \dots, j_n$ 是 j 的二进制表示， $0.j_1 \dots j_n$ 为二进制小数。上述变换可通过图中所示的量子线路实现。线路中使用了 Hadamard 门 H 、相位门 R_k 和交换门 \times ，其中 R_k 的矩阵表示为 $\begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{pmatrix}$ 。

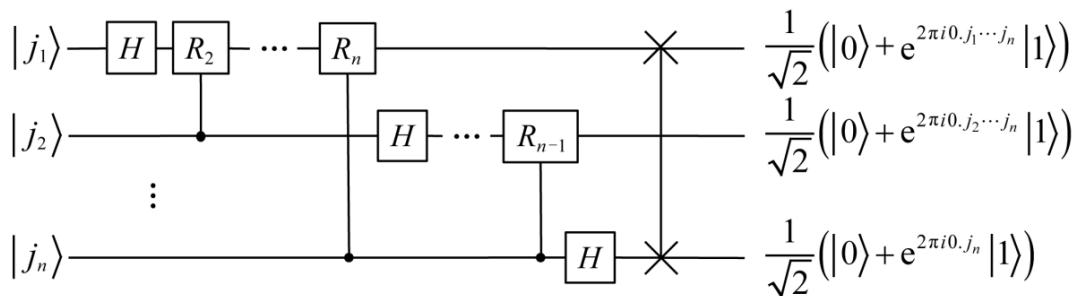


Fig. 1 The circuit of quantum Fourier transform

图 1 量子 Fourier 变换线路

图 3.1 量子 Fourier 变换线路

通过量子 Fourier 变换可以实现相位估计，设 $|u\rangle$ 是酉算子 U 的特征值为 $e^{2\pi i u}$ 的

一个本征态，则可大概率得到 φ 的指定精度的近似值。相位估计是众多量子算法的关键部分，结合经典算法，可以有效解决求阶、求周期问题，更一般地可有效解决隐含子群问题。Deutsch-Jozsa 算法、Shor 大整数分解算法、求离散对数等都是隐含子群问题的特例。目前大整数分解和求离散对数在经典计算机上还没有有效的求解方法，通过量子 Fourier 变换，在量子计算机上可有效求解，这也体现了量子计算较之经典计算的优越性。

3.4.2 量子搜索

量子搜索对无结构数据的搜索提供了二次加速。设在一个大小为 N 的无结构数据空间中有 M 个解，量子搜索通过大约 \sqrt{N} 次操作，可以找到一个解。虽然没有基于量子 Fourier 算法的指数加速效果，但由于搜索问题的普遍性，量子搜索算法仍具有很大的意义。

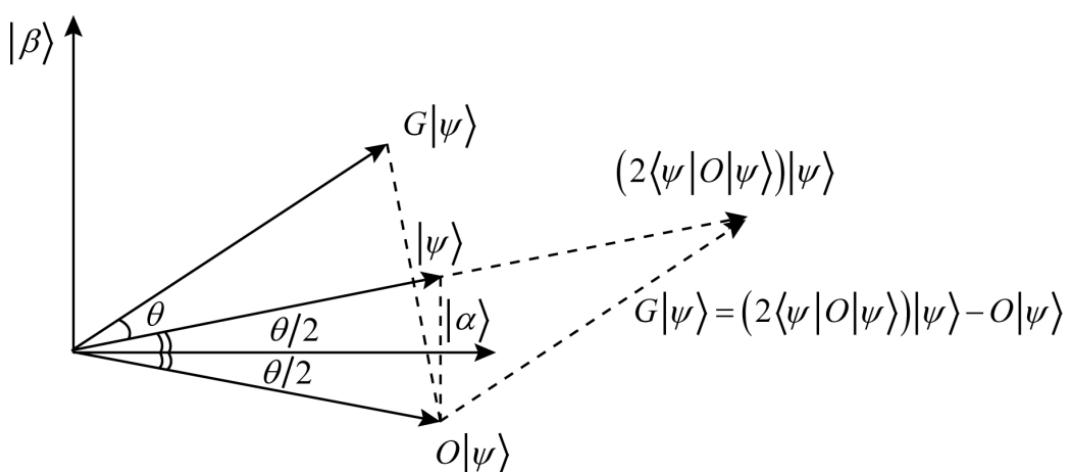


Fig. 2 The geometric intuitive of a Grover iterative

图 2 Grover 迭代几何直观

图 3.2 Grover 迭代几何直观

算法中使用了称为 Grover 迭代的算子，Grover 迭代可表示为 $(2|\psi\rangle\langle\psi| - I)O$ ，其中 $|\psi\rangle = H^{\otimes n}|0\rangle$ ， O 为识别搜索问题解的 Oracle。直观上来看 Grover 迭代实现了由初始量子态和搜索问题解组成的均匀叠加态张成的二维空间中的一个旋转，如图所示。其中 $|\alpha\rangle = \sum_x |x\rangle / \sqrt{N-M}$ 为归一化的非搜索问题解的叠加态， $|\beta\rangle = \sum_y |y\rangle / \sqrt{M}$ 为归一化的搜索问题解的叠加态， G 为 Grover 迭代算子， θ 满足 $\cos(\theta/2) = \sqrt{(N-M)/N}$ 。每经过一次 Grover 迭代，初态 $|\psi\rangle$ 向 $|\beta\rangle$ 方向靠近 θ 。经过 $O(\sqrt{N})$ 次迭代， $|\psi\rangle$ 接近 $|\beta\rangle$ ，在计算基中测量将以很高的概率输出搜索问题的一个解。

3.5 量子通信与量子密码研究进展

BB84 协议提出后，量子密码学正式登上历史的舞台。量子密码学以量子密钥分发为核心，对应于经典密码学领域的其他研究分支也得到了广泛关注，并形成各个不同的研究分支。本文对量子密钥分发、量子加密、量子签名和其他研究领域这 4 个方面的主要思想及进展情况进行介绍。

3.5.1 量子密码研究进展及主要思想

3.5.2 量子密钥分发

密钥分发用来在通信双方 (Alice 和 Bob) 安全分发一个密钥，后续可以用该密钥安全通信。BB84 协议是第一个量子密钥分发协议，被研究的最多，也最具代表性，在量子密码研究中占有重要地位，我们在此以 Bennett 和 Brassard 提出的原始协议为基础对其进行介绍。

BB84 协议使用光子作为量子态的载体，使用 2 组偏振基编码数据。一种为线偏振基 $\{\Box, \square\}$ 记为 +，水平偏振状态记为 $|\leftrightarrow\rangle$ ，垂直偏振状态记为 $|\updownarrow\rangle$ ，另一种为圆偏振基 $\{\circlearrowleft, \circlearrowright\}$ 记为 \bigcirc ，左旋偏振状态记为 $|\nearrow\rangle$ ，右旋偏振状态记为 $|\searrow\rangle$ 。在这 2 组基下，比特 “0” 分别被编码为 $|\leftrightarrow\rangle$ 和 $|\nearrow\rangle$ ，比特 “1” 分别被编码为 $|\updownarrow\rangle$ 和 $|\searrow\rangle$ 。描述光子线偏振和圆偏振的力学量算符不可对易，由 Heisenberg 不确定性原理，这 2 种偏振状态无法被同时确定。BB84 协议需要一条量子信道和一条经典信道。其量子信道可以是光纤或自由空间，经典信道为普通的公共信道，安全性不需考虑。这 2 种信道都允许第三方 (Eve) 监听。

1. Alice 对于某个安全参数 n ，随机选择稍多于 $4n$ 个比特，对每个比特随机选取线偏振基或圆偏振基进行编码，并将编码后的光子序列通过量子信道发送给 Bob。
2. Bob 收到光子序列后，随机选取线偏振基和圆偏振基对光子序列进行测量。
3. Bob 与 Alice 通过经典信道联系，对比他们所选择的基序列，舍弃选择不同基的比特，一般而言，他们将得到稍多于 $2n$ 个比特。
4. Alice 选择 n 个比特与 Bob 对比检查是否有第三方监听，如果错误率超过某一个阈值，则放弃本次协议其监听会造成对量子态的干扰，从而显着增大错误率。
5. Alice 和 Bob 对剩下的 n 个比特执行密钥纠错和安全性增强，得到最终的密钥。

最后 BB84 协议的工作过程可用下图所示的例子直观描述。

在 BB84 协议工作的过程中，Bob 收到 Alice 发送的光子序列后，并不知道 Alice 编码这些光子所用的基，他在随机选择测量基时，有 $1/2$ 的概率和 Alice 使用的基相同，因此在作基比对后，他们能得到大概原始比特数一半的比特形成的序列，在这个比特序列中，由于设备、环境等因素的影响，会出现一定的错误，记错误率为 ξ_0 。如果协

Alice chooses random bits	0	0	1	0	1	1	0	1	0	1	1	1	0
Alice chooses encoding basis randomly	+	○	○	+	○	+	○	+	+	○	+	○	○
Alice sends quantum bits	$ \leftrightarrow \rangle$	$ \varphi^+ \rangle$	$ \psi^- \rangle$	$ \leftrightarrow \rangle$	$ \psi^- \rangle$	$ \psi^+ \rangle$	$ \varphi^+ \rangle$	$ \psi^- \rangle$	$ \psi^+ \rangle$	$ \psi^- \rangle$	$ \psi^+ \rangle$	$ \psi^+ \rangle$	$ \varphi^+ \rangle$
Bob chooses measuring basis randomly	+	+	○	+	○	○	○	+	+	○	+	○	○
Bob's measuring result ^①	$ \leftrightarrow \rangle$	$ \leftrightarrow \rangle$	$ \psi^- \rangle$	$ \leftrightarrow \rangle$	$ \psi^- \rangle$	$ \varphi^+ \rangle$	$ \psi^+ \rangle$	$ \psi^- \rangle$	$ \psi^+ \rangle$	$ \psi^- \rangle$	$ \psi^+ \rangle$	$ \psi^+ \rangle$	$ \varphi^+ \rangle$
Alice and Bob compare their basis	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓
Alice and Bob keep the bits from the same basis	0		1	0	1			1					0
Comparison	Half of the reserved bits are compared, and the protocol is aborted when the error rate exceeds the threshold												
Error correction	Alice and Bob perform error correction on the remaining bits												
Security enhancement	Alice and Bob enhance the security of the corrected bits to get the final key												

① If Bob chooses the same base as Alice, he will obtain the correct results; otherwise, he will get one of the two states with uniform probability.

Fig. 3 The process of BB84 protocol

图 3 BB84 协议的工作过程

图 3.3 BB84 协议的工作

议过程中存在 Eve 监听, Eve 截获 Alice 发送的光子序列后, 受未知量子态不可克隆原理的限制, 他无法对光子序列进行复制, 为了获取信息, Eve 必须在原始光子序列上测量。然而, Eve 也不知道 Alice 编码光子所用的基, 他只能随机选择测量基, 在测量的过程中必然会对光子产生扰动, 使得在 Alice 和 Bob 作比特比对时, 得到的错误率超过 ξ_0 , 由此可以发现监听。

Alice 和 Bob 在比特比对后, 需要对剩下的比特序列纠错, 其基本思想是将这些比特序列分为若干区, 对每个区进行奇偶校验, 如果校验通过, 则放弃一个比特后保留该区, 如果校验不通过, 则放弃整个区, 经过若干次重复, 可确保他们有非常高的概率持有相同的比特序列。纠错后 Alice 和 Bob 对共享比特序列进行安全性增强, 如随机选择 Hash 函数对其进行压缩, 得到最终的共享密钥。

不同于经典密码学的安全性基于数学困难问题, BB84 的安全性基于量子不可克隆和不确定性原理等物理学定律, 它提供了无条件安全性, Shor 和 Preskill 于 2000 年对其进行了证明, 确认了这是一个可证安全的密钥分配方案, 符合现代密码学设计的基本要求。

3.5.3 量子加密

3.5.3.1 量子一次一密

1917 年 Vernam 提出了一种完善保密的加密方法, 称为一次一密 (one-timepad). 与之对应到量子密码学也有量子一次一密算法。根据明文、密钥和密文分别是经典比特还是量子比特, 量子一次一密算法主要有 3 种类型。

1. 使用 2 位经典比特加密一位量子比特明文，得到一位量子比特密文。该算法由 Boykin 和 Roychowdhury 提出。
2. 超密编码：由 Bennett 和 Wiesner 首先提出。超密编码开始需要通信双方 Alice 和 Bob 共享一对处于纠缠态的量子比特，如 Bell 态。Alice 对自己手中的量子比特作 Pauli 门操作或不作任何操作后，将其发送给 Bob，Bob 通过对这一对纠缠比特作合适测量，可得到 Alice 想要发送的 2 位经典比特明文。超密编码可以看作是使用一位量子比特作为密钥，加密 2 位经典比特明文，得到一位量子比特密文。
3. 量子隐形传态：开始时 Alice 和 Bob 共享一个 EPR 对，每人拥有 EPR 对的一个量子比特。Alice 将待发送的量子态 $|\varphi\rangle$ 与自己手中的一半 EPR 对作联合测量，得到两比特的经典信息，然后其发给 Bob，Bob 可以根据这两比特信息对自己的另一半 EPR 对作相应测量，得到 $|\varphi\rangle$ 。Gisin 等人将其视为明文、密钥和密文都是量子比特的一次一密。

3.5.3.2 量子公钥加密

2000 年 Okamoto 等人在美密会上首先提出了量子公钥加密方案。方案中消息的发送方、接收方以及敌手都被抽象成量子多项式时间图灵机，并且在量子计算模型下构造了量子单向陷门函数。在这之后，多种多样的量子公钥密码方案被提出。2003 年 Yang 提出了一个基于经典 NP 完全问题的量子公钥加密方案。2008 年 Nikolopoulos 基于量子比特旋转变换提出了一个公钥加密方案。2009 年 Gao 等人使用对称密钥构造了量子公钥加密方案。2012 年 Liang 等人提出了一个信息论安全的加密方案。2014 年 Zheng 等人提出了面向比特的概率型量子公钥加密方案。2015 年 Vlachou 等人提出了基于量子随机游走的方案。2017 年 Wu 等人提出了基于 Bell 态的公钥加密方案。

3.5.3.3 量子同态加密

2012 年 Rohde 等人使用玻色子采样和量子行走模型实现了有限的量子同态加密。2015 年 Liang 基于通用量子线路，构造了量子全同态加密方案，该方案中可以对加密数据执行任意量子变换。2015 美密会上，Broadbent 和 Jeffery 基于经典 FHE 的存在提出了 2 种 QHE 方案。他们提供了 2 种不同的方法来完成具有有限数量的非 Clifford 门线路的同态加密，还提出了 QFHE 及其安全性的正式定义。2016 年 Dulek 等人扩展了这项工作，以便有效地评估任意多项式大小的线路，并提供了一种新的紧凑型 QHE 方案。2017 年 Ouyang 等人提出了一种 (n, n) 阈值秘密共享方案，该方案允许对共享秘密上的量子线路进行评估而无需对其进行解码。此外还有一些其他的量子同态加密方案。

3.5.4 量子签名

量子签名是量子密码学的一个重要分支，在 2001 年由 Gottesman 等人首次提出，它通过量子力学原理保证数据签名的安全性。同经典签名一样，其安全性需要满足 3 个属性。

- 1 不可伪造，没有人能伪造一个合法的签名。
- 2 不可否认，签名者不能对自己的签名否认。
- 3 可公开验证，接收到消息的任何人都可通过。

公钥验证消息签名的合法性，人们最开始研究的量子签名是依赖于仲裁的，第一个具体方案由 Zeng 等人在 2002 年提出，此后 Curty 等人、Zou 等人、Gao 等人利用经典签名协议的分析方法，给出了该方案的一些安全漏洞。2009 年 Li 等人提出了基于 Bell 态的仲裁量子签名。2015 年 Li 和 Shi 提出了基于 CNOT 链加密的仲裁量子签名。2018 年 Feng 等人提出了基于连续变量量子态的仲裁量子签名。

3.6 密码学界面对威胁的应对

面对量子计算突飞猛进的威胁，密码学界以美国国家标准与技术研究院 (NIST) 为首，早早就开始了应对布局，全面开展了一种公开的程序挑选一个或多个公钥密码算法的进程。这些新的公钥密码标准将为数字签名、公钥密码以及密钥建立算法规定一个或多个另外的算法。新标准将对 FIPS 186-4 数字签名标准 (DSS) 以及特别出版物 800-56A 版本 3 (使用离散对数密码的双序密钥建立方案建议) 和 SP 800-56B (使用整数因子分解的双序密钥建立方案建议) 进行扩充。NIST 的意图是，在可以预见的未来，包括量子计算机出现之后，这些算法将能够保护美国政府的敏感信息，这个竞赛过程称为 NIST PQC 后量子密码标准化进程。

PQC 标准化进程是 NIST 对量子计算机技术发展的响应。量子计算机利用了量子力学现象来求解困难的或传统计算机难以处理的难题。如果大容量规模的量子计算机被建造出来了，它们将能够破解 NIST 目前标准化的这些公钥密码系统，包括数字签名和密钥建立方案。量子计算机对对称密码系统也有一定的影响，但其影响并不强烈。后量子密码的目标，是发展对量子计算机和经典计算机都安全的密码系统，并且能够与现有协议和网络互操作。在启动 PQC 标准化进程之前，NIST 于 2015 年 4 月就成立了一个工作组，讨论与后量子密码以及未来可能的标准化相关的问题。一年之后，NIST 发布了后量子密码报告 NISTIR，共享了 NIST 对量子计算和后量子密码的状态的理解，概括了 NIST 推进该领域的初始计划，NIST 于 PQCrypto 2016 会议上以报告形式宣布了 PQC 标准化进程的初步细节。

2016 年 8 月，NIST 以一个联邦注册公告的方式，公布了对后量子密码的公开注

释建议的要求和评估判断准则。此后，又基于公开的反馈意见，对这些要求和评估判断准则进行了更新，并且稍后将它们包含在 2016 年 12 月 20 日的第二次联邦注册公告 FRN-Dec16 内。该公告正式征集关于后量子密码算法的公开提案，标志着 NIST 开始了 PQC 标准化的进程。候选提案的最后期限预定在 2017 年 11 月 30 日，在那个时日，NIST 接收到了 82 个提案包。这是世界范围内密码社区，在 1998 年为高级加密标准竞赛提出过 21 个候选算法，在 2008 年为安全哈希算法 3 (SHA-3) 竞赛提交过 64 个文件包作出的强烈反应，NIST 于 2017 年 12 月 20 日宣布接受了满足提交要求和最小接受判断准则的 69 个第一轮候选算法。这 69 个提案包含了 20 个数字签名方案和 49 个公钥密码 (PKE) 或密钥封装机制 (KEM)。第一轮候选算法的提交包都被在线张贴在 <https://www.nist.gov/pqcrypto> 网页上，征求公开的评论和注释。

NIST 于 2018 年 4 月 11-13 日在佛罗里达 Lauderdale 组织了第一次 PQC 标准化会议，其研究成果收录在 PQCrypto 2018 中。获得接受的第一轮候选算法的提交者被邀请到会上介绍他们的算法。NIST 还讨论了其缩小候选算法池、聚焦整个社区的关注点以及启动第二轮标准化进程的计划。在第一轮竞赛中，NIST 接收到密码社区的大量反馈。基于对第一轮候选算法的公开反馈和内部评估，NIST 于 2019 年 1 月 30 日宣布挑选出 26 个算法作为第二轮候选算法，进入该标准化进程的第二阶段。2019 年 7 月 20 日宣布挑选出其中的 7 个算法作为第三轮最终候选算法 PK，另外 8 个算法作为后补。

最终入选 7 算法如下：

3.6.1 CRYSTALS-KYBER

Kyber 为一族密钥封装机制，它基于模误差学习 (MLWE) 问题的后量子困难问题假设，提供选择性的密文安全性 (即 IND-CCA)。Kyber 包含了一个标准的带误差学习 (LWE) 风格的 CPA 安全的公钥密码方案，其基础的代数目标是在 2 的幂次的圆分割环上的一个模。通过对这个参数的精选，使其能够采用数论变换 (NTT) 进行效率非常高的计算。其噪声按照一个有中心的二项式分布来进行采样。CCA 安全性是通过 Fujisaki-Okamoto 变换的一种著名的变体来达到，其中会话密钥是基于一种加密的方法来传送的。对 Kyber 的安全性强度水平的调整十分简单。一种调整是仅仅改变底层模的秩序 (在 2, 3, 4 的范围内)，并且调整其噪声分布参数 (分别在 5, 4, 3 的范围内)。对于密钥交换而言，Kyber 的性能为最具竞争力的密钥交换设计之一。

Kyber 的安全性依赖于一个已深入研究问题的变体。其提交文档提供了一种基于随机预言模型 (ROM) 的严密安全性证明，以及一种基于量子随机预言模型 (QROM) 的非严密安全性证明，二者都是基于 MLWE 假设。我们注意到一个潜在的问题是，这个安全性证明并不直接应用于 Kyber 自身，而是应用于该方案的一个不压缩公钥的改进

版本。如果没有这种改进，Kyber 实际上是依赖于一个不同的(或附加的)类似取整的假设。如果是这样的话，将带来密码分析上的担忧，因为在 MLWE 和模取整学习(MLWR)之间进行的那些已知的缩减设计，可能并不要求 Kyber 选取的那些参数。

3.6.2 NTRUEncrypt 与 NTRU-HRSS-KEM 合并的 NTRU

NTRU 密码为一种基于格的单向 CPA 安全(OW-CPA)的公钥密码方案，它大约发明于 1996 年。经过了数十年的研究，NTRU 密码的安全性已经得到了相当好的理解和深入的研究审查。已经发展出该方案的许多变体及其派生出的密钥封装机制，包括 NTRUEncrypt 和 NTRU-HRSS-KEM 方案。这两个提案已经宣布合并为一种方案，称之为 NTRU。NTRU 的安全性基础是比 LWE 或 RLWE 稍微强一点的一种假设，有时称之为 NTRU 假设。在这个合并中，KEM 为一种严密的 IND-CCA2 安全性转换，源自于量子随机预言模型的一种确定性 OW-CPA 安全的公钥密码方案。该提案对于 KEM 中的确定性 PKE 具有两个选项。其派生的 KEM 没有解密失败问题，同时避免了密钥产生期间的“对 1 值的评估”和可逆校验这两个问题。该方案以三个环结构实例详细说明了三个参数集。KEM 产生的密钥和密文都具有良好的尺寸，其封装和解封效率看起来都很高。虽然它没有形成已知的安全薄弱点，在具有相似效率的 RLWE 和 MLWE 密码系统中，NTRU 的构造产生的结构稍多一些(由于其矢量比期望的矢量更短)。这个方面还需要进一步研究。

3.6.3 SABER

SABER 是一个基于格的 PKE 和 KEM 簇，它基于取整的模学习量子困难问题来达到 CPA 和 CCA 安全性。该方案使用在一个 2^k 的幂次的分圆环上具有固定维度和模数的不同秩的多个模块，其安全级别分别为 1、3 以及 5。其 MLWR 秘密分布为有中心的二项式分布，其会话密钥哈希值再由公钥进行哈希运算，以实现多目标防护。其多项式相乘按照 Toom-Cook 和 Karatsuba 的详细描述进行。该方案通过 Fujisaki-Okamoto 变换的一个变量来达到 CCA 安全性。其会话密钥使用一种基于带噪声的密钥一致性协调的密码方法来传送。在所有后量子密钥交换中，SABER 的性能优势具有非常强的竞争力。在每个安全级别中，它的带宽代价都是比较低的。关于 SABER 安全性的最明显担忧，是它是否过分扩展了(模)取整学习的实际难度。虽然尚未有已知的利用所选参数的与(M)LWR 相关的攻击，在取整样本 MLWE 和 MLWR 之间仍然存在很宽泛的缩减空间，并且还没有应用到 SABER 中。NIST 将 SABER 当作一个机遇，以突出对具有较小参数和取整样本的(M)LWR 进行深入分析的需求，有利于推进该方面的研究工作，继续全面增强对该假设的信心。

3.6.4 Classic McEliece

Classic McEliece 是一种 IND-CCA2 安全级别的密钥封装机制 (KEM)。该提案以有名的 McEliece 密码系统为基础，McEliece 密码系统为 1978 年公布的第一个基于编码的公钥密码系统。该 KEM 的公钥机制确定一个随机二元 Goppa 码，通过在码字上加入误差来产生密文，通过解码来实现解封。其安全性基于解码一个常规线性码的难度，并且假设一个随机二元 Goppa 码不能通过一个随机线性码来辨别。其安全问题已经具有很长的分析研究历史，但并没有显著改变其攻击复杂度。Classic McEliece 也具有产生的密文非常短的特点，大概 200 个字节左右，看起来具有良好的封装和解封性能。McEliece 类型的密码系统的主要缺点是具有很大的公钥尺寸，超过了一百万个字节。该提案仅仅包含了第 5 类安全性的参数集，因而希望提交者生成其它安全类别的参数集。

3.6.5 CRYSTALS-DILITHIUM

Dilithium 为一种格基签名方案，采用了 Fiat-Shamir 启发方法来构造，其安全性是依赖于 MLWE 的困难问题。Dilithium 与密钥交换机制 Kyber 都是 CRYSTALS 的组成部分。Dilithium 的主要新颖性在于通过忽略一些低阶 bit 来减小其公钥尺寸；为了进行补偿，每个签名都包含了一个额外的“线索”，允许验证者对签名进行核对。Dilithium 提供了相当良好的性能，其实现也相对简单。对 Dilithium 的最知名的攻击是基于格的偏移缩减，这种攻击没有有效利用 MLWE 问题的代数结构。Dilithium 的参数选择以对这些攻击代价的保守估计为基础。Dilithium 具有一个基于经典随机预言机模型的形式化的安全性证明。这个证明是极不平凡的，它突破了量子随机预言机模型；但是还没有任何已知攻击的实例。NIST 鼓励对这个方案的所有方面都进行深入的分析研究。

3.6.6 FALCON

FALCON 为一种格基签名方案，它基于 GPV(Gentry-Peikert-Vaikuntanathan) 高斯取样，应用了 NTRU 格。其主要的创新性在于，它应用了一种树形的数据结构(即 FALCON 树)，使其成为对高斯采样的一种非常快的递归算法。对 FALCON 方案的最知名的攻击是基于格的偏移缩减，这种攻击没有有效利用 NTRU 格的特殊结构。FALCON 具有一个基于经典随机预言机模型的形式化的安全性证明。FALCON 提供了非常优良的性能，但是其实现相当复杂，因为它严重依赖于数域的多域塔结构，并且需要双精度浮点算法。该方案还需要进行更多的研究工作，确保其签名算法面对边信道攻击时是安全的。

3.6.7 Rainbow

Rainbow 是一种多变量数字签名方案，它是 UOV 结构的一种推广，允许进行参数优化，使其以增加代数结构的代价获得更高的效率。Rainbow 签名方案所提交的格式已经具有长达 15 年的针对各种参数的研究历程。Rainbow 方案声称，由于使用了随机加盐的一种哈希构造，因而具备 EUF-CMA 安全性。Rainbow 参数谱允许在各种各样的大量应用场景中进行优化。Rainbow 的另一优势是它已经在其它包括轻量级应用在内的场景中得到了研究。Rainbow 提案针对 NIST 号召建议中的所有安全级别，提供了若干个参数集。作为一种入选第二轮的候选算法，期望能够将研究重点聚焦于一组更窄的细节规格上面。此外，还希望能激发针对 Rainbow 密钥以及文献中已有的、而 Rainbow 提案中并未考虑的那些优化技术开展更多的研究，最终使研究社区在方案的可行性上面取得一致。此外，Rainbow 的实现也可能得到很大的改进。

实际上我们并不处在建造一台有能力破译目前公钥密码(学)的量子计算机的中期阶段(例如 5 年)内。执行肖尔算法所需的量子比特的数量，远远超过今天我们制造的最强大的量子计算机的能力，而且肖尔算法大规模的可靠的应用还没有被展示出来。今天，我们并不清楚诸如量子比特的退相干和管理噪声以减少量子系统中的差错等面临的挑战能否被解决。正因为如此，在短期或者中期内建造这样一台可规模化的扩展的量子计算机似乎不太现实。

然而，放眼较长的时间(20 年)，忽视量子计算机对于目前的现代密码学标准的威胁将会是错误的。正如美国国家标准与技术研究院的描述：“一些人预测在接下来的 20 年左右，足够规模的量子计算机会被制造出来，基本上可以打破目前使用中的所有公钥方案。在历史上，我们花了差不多 20 年时间去配置现代公钥密码基础架构。因此，不论我们能否准确估计量子计算时代到来的时间，我们都必须现在开始准备我们的信息安全系统，使之能够对抗量子计算”。

第四章 可信计算技术与近来深度学习与密码学

本章针对可信计算技术与近来深度学习与密码学等进来资讯进行整理。当中共分为可信计算技术与深度学习两大部分进行说明。

4.1 可信计算概述

4.1.1 为何需要可信计算

如今信息技术已经成为了人们生活中不可分割的一部分，人们每天都通过计算机和互联网获取信息、进行各种活动。但计算机与网络空间并不总是安全的，一方面黑客们会通过在网络中散布恶意病毒来对正常用户进行攻击，例如 2017 年 5 月爆发的勒索病毒；另一方面许多不良厂商会在自己的软件中“开后门”，趁用户不注意时获取用户的隐私或者弹出弹窗广告，这些都给维护网络空间的信息安全带来了巨大的挑战。为了使人们能够正常地通过计算机在互联网上进行各种活动，我们必须建立一套安全、可靠的防御体系来确保我们的计算机能够按照预期稳定地提供服务。

目前多数的网络安全系统主要由防火墙、入侵检测、病毒防范等组成。这种常规的安全手段只能在网络层、边界层设防，在外围对非法用户和越权访问进行封堵，以达到防止外部攻击的目的。由于这些安全手段缺少对访问者源端—客户机的控制，加之操作系统的不安全导致应用系统的各种漏洞层出不穷，其防护效果正越来越不理想。此外，封堵的办法是捕捉黑客攻击和病毒入侵的特征信息，而这些特征是已发生过的滞后信息，属于“事后防御”。随着恶意用户的攻击手段变化多端，防护者只能把防火墙越砌越高、入侵检测越做越复杂、恶意代码库越做越大，误报率也随之增多，使得安全的投入不断增加，维护与管理变得更加复杂和难以实施，信息系统的使用效率大大降低，而对新的攻击毫无防御能力。近年来，“震网”“火焰”“Mirai”“黑暗力量”“WannaCry 勒索病毒”等重大安全事件频频发生，显然，传统防火墙、入侵检测、病毒防范等“老三样”封堵查杀的被动防御已经过时，网络空间安全正遭遇严峻挑战。安全防护手段在终端架构上缺乏控制，这是一个非常严重的安全问题，难以应对利用逻辑缺陷的攻击。目前利用逻辑缺陷的漏洞频繁爆出，如“幽灵”“熔断”，都是因为 CPU 性能优化机制存在设计缺陷，只考虑了提高计算性能而没有考虑安全性。由这种底层设计缺陷导致的漏洞难以修补，即使有了补丁其部署难度也是越来越大。幽灵、熔断的补丁部署后会使性能下降 30%。补丁难打、漏洞难防已经是当前信息安全防御主要问题之一。

可信计算正是为了解决计算机和网络结构上的不安全，从根本上提高安全性的技术方法，可信计算是从逻辑正确验证、计算体系结构和计算模式等方面的技术创新，以解决逻辑缺陷不被攻击者所利用的问题，形成攻防矛盾的统一体，确保完成计算任务的逻辑组合不被篡改和破坏，实现正确计算。

4.1.2 何为是可信计算

可信计算概念最早可以追溯到美国国防部颁布的 TCSEC 准则。1983 年美国国防部制定了世界上第一个《可信计算机系统评价标准》(TCSEC)，第一次提出了可信计算机和可信计算基 (Trusted Computing Base, TCB) 的概念，并把 TCB 作为系统安全的基础。

4.1.2.1 可信的定义

可信计算的首要问题是要回答什么是可信。目前关于可信尚未形成统一的定义，不同的专家和不同的组织机构有不同的解释。主要有以下几种说法。1990 年，国际标准化组织与国际电子技术委员会 ISO/IEC 在其发布的目录服务系列标准中基于行为预期性定义了可信性：如果第 2 个实体完全按照第 1 个实体的预期行动时，则第 1 个实体认为第 2 个实体是可信的。1999 年，国际标准化组织与国际电子技术委员会在 ISO/IEC15408 标准中定义可信为：参与计算的组件、操作或过程在任意的条件下是可预测的，并能够抵御病毒和一定程度的物理干扰。2002 年，TCG 用实体行为的预期性来定义可信：一个实体是可信的，如果它的行为总是以预期的方式，朝着预期的目标。这一定义的优点是抓住了实体的行为特征，符合哲学上实践是检验真理的唯一标准的基本原则。IEEE 可信计算技术委员会认为，可信是指计算机系统所提供的服务是可信赖的，而且这种可信赖是可论证的。

4.1.2.2 信任的获得方法

信任的获得方法主要有直接和间接两种。设 A 和 B 以前有过交往，则 A 对 B 的可信度可以通常考察 B 以往的表现来确定，我们称这种通过直接交往得到的信任值为直接信任值。设 A 和 B 以前没有任何交往，但 A 信任 C，并且 C 信任 B，那么此时我们称 A 对 B 的信任为间接信任。有时还可能出现多级间接信任的情况，这时便产生了信任链。

4.1.2.3 可信计算的基本思想

在计算平台中，首先创建一个安全信任根，再建立从硬件平台、操作系统到应用系统的信任链，在这条信任链上从根开始一级测量认证一级，一级信任一级，以此实

现信任的逐级扩展，从而构建一个安全可信的计算环境。一个可信计算系统由信任根、可信硬件平台、可信操作系统和可信应用组成，其目标是提高计算平台的安全性。

4.1.3 可信计算的发展概况

早在 20 世纪 60 年代，为了提高硬件设备的安全性，人们设计了具有高可靠性的可信电路，可信的概念开始萌芽。到 20 世纪 70 年代初期，Anderson 首次提出了可信系统的概念，为美国后续的 TCSEC（彩虹系列），可信计算机、可信计算基（TCB）、可信网络、可信数据库等的提出奠定了基础。彩虹系列是最早的一套可信计算技术文件，标志着可信计算的出现，也使系统的可信不断丰富可信的内涵，可信计算的理念和标准初具雏形。

从 20 世纪 90 年代开始，随着科学计算研究的体系化不断规范、规模的逐步扩大，可信计算产业组织和标准逐步形成体系并完善。1999 年，IBM、HP、Intel 和微软等著名 IT 企业发起成立了可信计算平台联盟（TCPA, Trusted Computing Platform Alliance），这标志着可信计算进入产业界。2003 年，TCPA 改组为可信计算组织（TCG, Trusted Computing Group）。目前，TCG 已经制定了一系列的可信计算技术规范，如可信 PC、可信平台模块（TPM）、可信软件栈（TSS）、可信网络连接（TNC）、可信手机模块等，且不断地对这些技术规范进行修改完善和版本升级。

早在 2000 年伊始，我国就开始关注可信计算，并进行了立项、研究，和国外不同，我国在可信计算上走的是先引进技术后自主研发、先产品化后标准化的跨越式发展。2004 年，武汉瑞达生产了中国第一款 TPM，之后联想、长城等基于 TPM 生产了可信 PC。2005 年 1 月，全国信息安全标准化技术委员会成立了可信计算工作小组（WGI），先后研制制定了可信密码模块（TCM）、可信主板、可信网络联接等多项标准规范。2005 年，国家出台“十一五”规划和“863”计划，把“可信计算”列入重点支持项目，我国出现了一系列的可信计算产品。截止到目前，国际上已形成以 TPM 芯片为信任根的 TCG 标准系列，国内已形成以 TCM 芯片为信任根的双体系架构可信标准系列。其国际与中国国内两套标准最主要差异如下所列。

1. 信任芯片是否支持国产密码算法，国家密码局主导提出了中国商用密码可信计算应用标准，并禁止加载国际算法的可信计算产品在国内销售。
2. 信任芯片是否支持板卡层面的优先加电控制，国内部分学者认为国际标准提出的 CPU 先加电、后依靠密码芯片建立信任链的模式强度不够，为此，提出基于 TPCM 芯片的双体系计算安全架构，TPCM 芯片除了密码功能外，必须先于 CPU 加电，先于 CPU 对 BIOS 进行完整性度量。
3. 可信软件栈是否支持操作系统层面的透明可信控制，国内部分学者认为国际标

准需要程序被动调用可信接口，不能在操作系统层面进行主动度量，为此，提出在操作系统内核层面对应用程序完整性和程序行为进行透明可信判定及控制思路。

4.1.4 可信计算技术

4.1.4.1 信任根

TCG 定义的信任根包括 3 个根，其包含负责完整性度量的可信度量根 (RTM)，负责报告信任根的可信报告根 (RTR)，负责存储信任根的可信存储根 (RTS)。其中，RTM 是一个软件模块、RTR 是由 TPM 的平台配置寄存器 (PCR) 和背书密钥 (EK) 组成、RTS 是由 TPM 的 PCR 和存储根密钥 (SRK) 组成。实践中，RTM 在构建信任链的过程中，将完整性度量形成的信息传递给 RTS，RTS 使用 TPM 的平台配置寄存器存放度量扩展值、使用 TPM 提供的密码学服务保护度量日志。RTR 主要用于远程证明过程，向实体提供平台可信状态信息，主要内容包括平台配置信息、审计日志与一般由背书密钥或者基于背书密钥保护的身份密钥承担的身份密钥。

4.1.4.2 信任链

信任链的主要作用是将信任关系扩展到整个计算机平台，它建立在信任根的基础上。信任链可以通过可信度量机制来获取各种各样影响平台可信性的数据，并通过将这些数据与预期数据进行比较，来判断平台的可信性。建立信任链时遵循以下 3 条规则。

1. 所有模块或组件，除了 CTRM，也就是信任链构建起点，第一段运行的用于可信度量的代码，在没有经过度量以前，均认为是不可信的。同时，只有通过可信度量且与预期数据相符的模块或组件，才可归入可信边界内。
2. 可信边界内部的模块或组件，可以作为验证代理，对尚未完成验证的模块或组件进行完整性验证。
3. 只有可信边界内的模块或组件，才可以获得相关的 TPM 控制权，可信边界以外的模块或组件无法控制或使用可信平台模块。

TCG 的可信 PC 技术规范中提出了可信 PC 中的信任链，如图中所示，TCG 的信任链很好地体现了度量存储报告机制。即对平台可行性进行度量，对度量的可信值进行存储。另外名词解释如下所示：

1. 度量：该信任链以 BIOS 引导区与 TPM 为信任根，其中 BIOS 引导区为可信度量根 (RTM)，TPM 为可信存储根 (RTS)、可信报告根 (RTR)。从 BIOS 引导区出发，到 OS Loader、再到 OS、应用，构成一条信任链。沿着这条信任链，一级

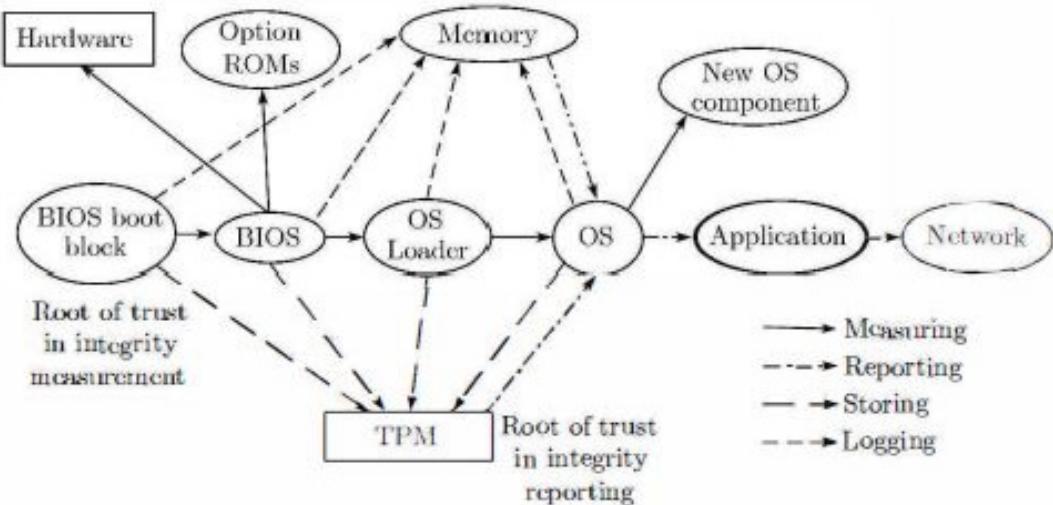


图 1 TCG 的信任链

图 4.1 TCG 信任链

度量一级，一级信任一级，确保平台资源的完整性。

2. 存储：由于可信平台模块存储空间有限，所以采用度量扩展的方法即现有度量值和新度量值相连再次散列来记录和存储度量值到可信平台模块的 PC 中，同时将度量对象的详细信息和度量结果作为日志存储在磁盘中。存储在磁盘中的度量日志和存储在 PCR 中的度量值是相互印证的，防止磁盘中的日志被篡改。
3. 报告：度量、存储之后，当访问客体询问时，可以提供报告，供访问对象判断平台的可信状态。向客体提供的报告内容包括 PCR 值和日志。为了确保报告内容的安全，还须采用加密、数字签名和认证技术，这一功能被称为平台远程证明。

4.1.4.3 可信平台模块

目前主要的可信平台模块主要有三种，分别是 TCG 的 TPM、中国的 TCM 和 TPCM，本小节只介绍 TCG 的 TPM。可信平台模块是可信计算平台的信任根的 RTS、RTR，它本身是一个 SOC 芯片，由 CPU、存储器、I/O、密码协处理器、随机数产生器和嵌入式操作系统等部件组成，主要用于可信度量的存储、可信度量的报告、密钥产生、加密和签名、数据安全存储等功能。

TCG 先后发布过多个版本的 TPM 标准，其中，TPM 1.2 使用较为广泛，但随着信息计算机技术的不断发展，TPM 1.2 无法满足新技术下的需求，2014 TCG 发布了 TPM 2.0，相较于 TPM 1.2，TPM 2.0 有如下改进，(1) 吸收原有 TPM (TPM1.2) 和中国 TCM 的优点、(2) 改进原 TPM 在密码算法灵活方面存在的问题、(3) 使之成为一个国际标准，解决不同国家的本地需求，并保持较好的兼容性，如中国国内的 TPM 2.0 芯片支持国

家密码局允许的密码学算法 SM3、SM2、SM4 等。TPM 2.0 结构及模块功能如图所示。

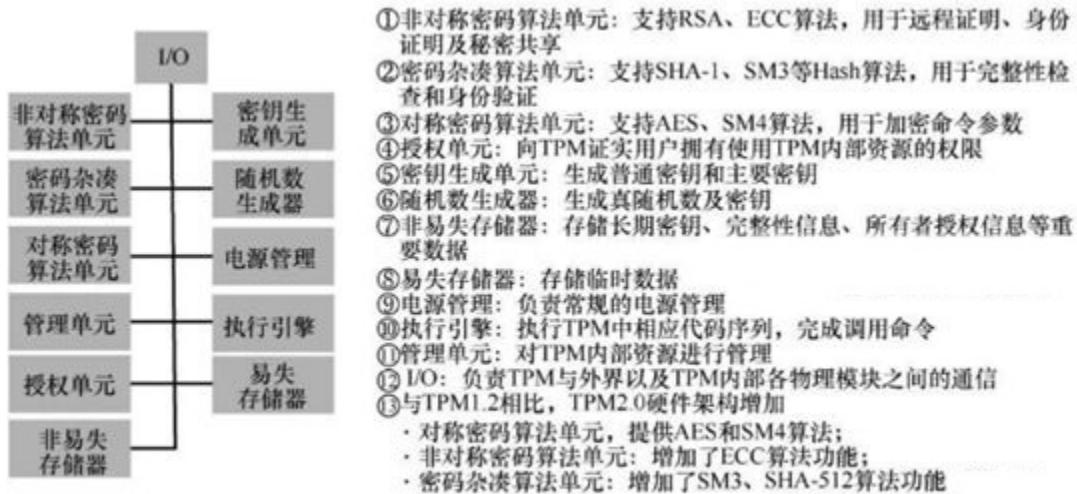


图2 TPM 2.0功能模块

图 4.2 TPM 2.0

4.1.4.4 可信支撑软件

可信支撑软件是操作系统层面安全应用可以调用可信计算平台提供的可信服务接口，从而为用户提供可信服务。TCG Software Stack (TSS) 是可信计算平台上 TPM 的支撑软件，其 TSS 的作用主要是为操作系统和应用软件提供使用 TPM 的接口。目前，TSS 主要有 TSS 1.2 和 TSS 2.0 两个版本。其中基于 TPM 2.0 的 TSS 2.0 是最新的版本，其结构如下图所示。此外开放授权原始码可于 GitHub 项目平台，名 [tpm2-software/tpm2-tss](#) 找到。

4.1.5 围绕可信计算的一些争议

尽管可信计算一经提出就获得了众多学者的大力支持，但可信计算的反对者指出：保护计算机不受病毒和攻击者影响的安全机制，同样会限制其属主的行为。剑桥大学的密码学家 Ross Anderson 他们指出这将使得强制性垄断成为可能，从而会伤害那些购买可信计算机的人们。Anderson 还总结道“最根本的问题在于控制可信计算基础设施的人将获取巨大的权力。拥有这样的权力就像是可以迫使所有人都使用同一个银行、同一个会计或同一个律师。而这种权力能以多种形式被滥用。”

除此之外，由于装有可信计算设备的计算机可以唯一证明自己的身份，厂商或其他可以使用证明功能的人就能够以非常高的可能性确定用户的身份。可信计算的赞成者指出，它可以使在线购物和信用卡交易更安全，但这可能导致计算机用户失去访问互联网时希望拥有的匿名性。批评者指出这可能导致对政治言论自由，新闻记者使用

TCG Software Stack 2.0

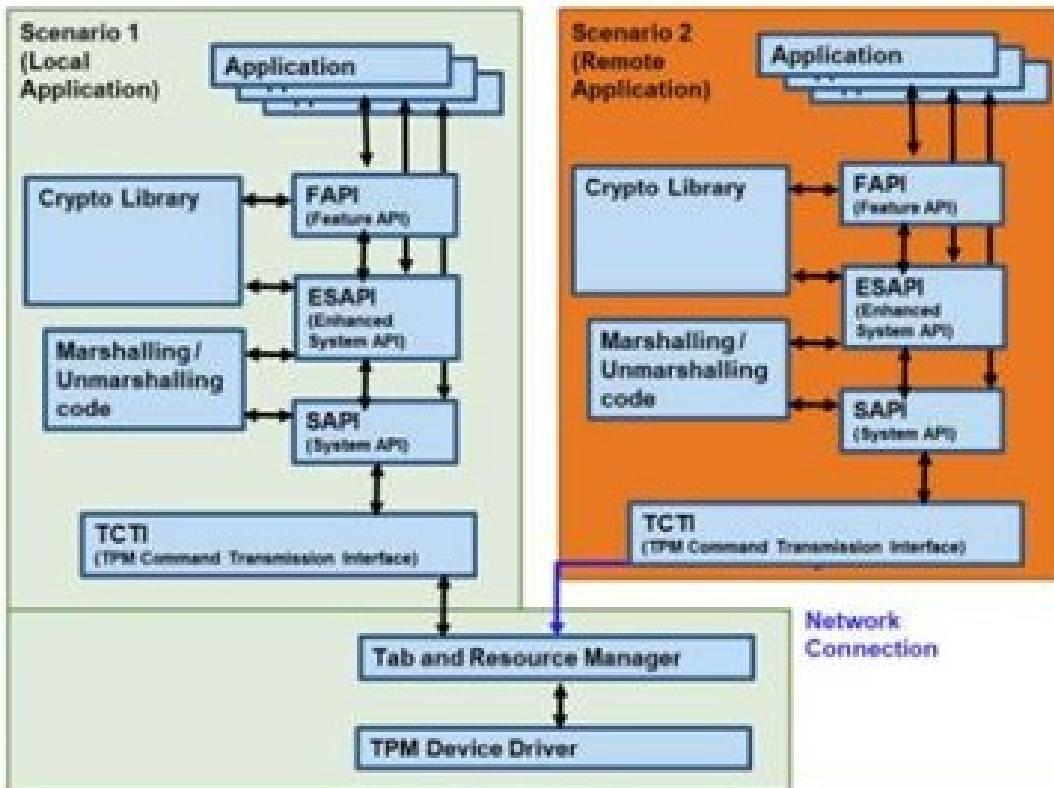


图 4.3 TCG Software Stack

匿名信息源，揭发政治博客，以及其他公众需要通过匿名性来防止报复的领域，产生抑制作用。

4.2 深度学习与密码学

加密领域是资安攻防的一大重点，举凡 RSA、AES、DES、Blowfish 和 Twofish 这些，都是著名的加密演算法，而这些演算法，可以让攻击者花较多时间破解，并配合验证机制以减少资料库被盗取的机会。在人工智能开始流行之后，也渐渐开始出现能以机器学习、深度学习实现的加密演算法。

4.2.1 CryptoNets

CryptoNets 采用 MNIST 资料集来做机器学习，并配合名为同态加密技术 (HE; Homomorphic encryption)，让机器能够辨识加密资料，从而提高安全性。另外同态加密技术用于将资料托给第三方的时候，亦可适用于云端。它是一类对称型加密演算法，也是公钥加密技术的一种。同时著名的 RSA 加密演算法属于 HE 的雏型，此外 CryptoNets

也可用 GAN 的方式来实作。

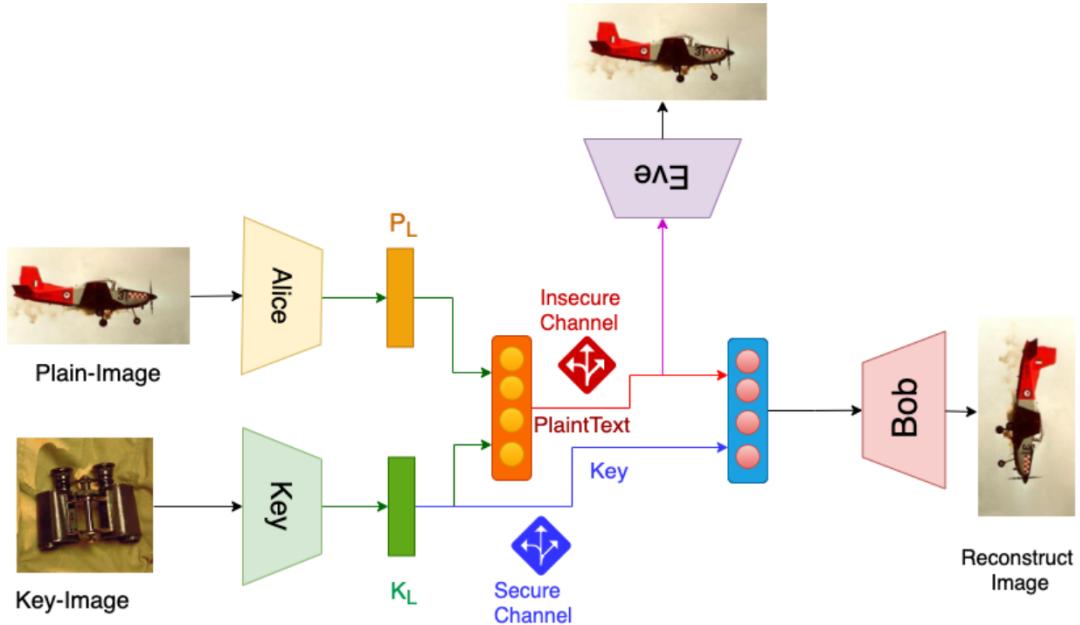


图 4.4 CryptoNets 的模式图

4.2.2 CryptoNN

CryptoNN 为以自身研究所开发的神经网路结合功能性加密的演算法，在神经网路领域方面，其架构如图所示，该研究的论文作者表示此类型的加密演算法适用所有神经网路架构，同时在论文中采用以 CNN 改写的架构进行，而在此架构中，使用的演算法为自行开发的功能性加密技术 (Functional Encryption; FE)，并于之后并加入安全矩阵运算 (Secure Matrix Computation)，其算法与架构如图所示。

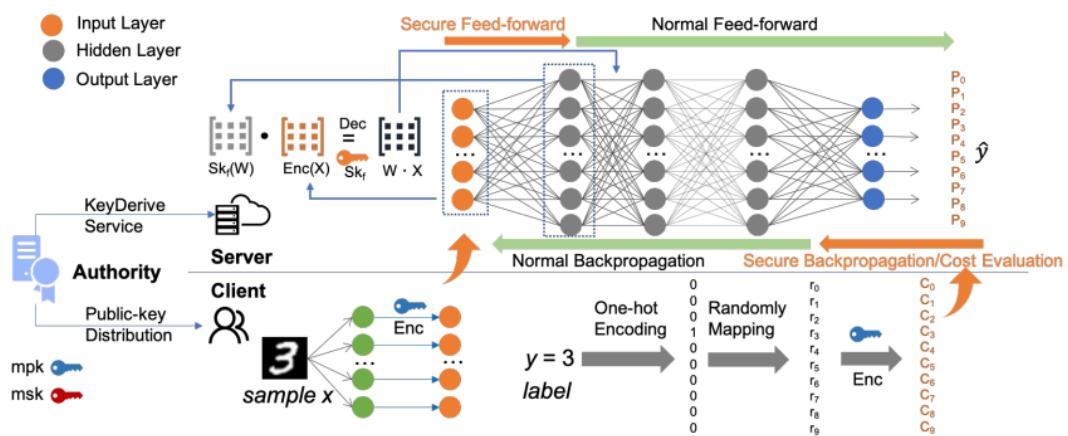


图 4.5 CryptoNN 架构

4.2.3 CV-QNN

CV-QNN 结合了量子电脑、类神经网路与加密演算法的概念，十分先进。但由于目前量子电脑技术尚未成熟，故仍暂时无法实作，而只能模拟的方式。其 CV-QNN 的发明是因应量子电脑的产生，可能会对于传统加密演算法产生不小的冲击。在该篇研究中，提及的 CV-QNN 架构如下图所示。其主要构造较近似于 RSA 加密，只是加密与解密的主要机制需要依靠量子电脑运算，同时配合类神经网路相关演算法，可以有效阻挡攻击者以暴力方式破解密码而获取资料。此型态在资安上为一新趋势，或许，在不远的未来，也可以和经生物学或基因与蛋白质的 Encoding 结合，发展出另一些新型态的加密技术，保障使用者的资料安全。

Algorithm 1: secure matrix computation scheme

Input: \mathbf{X} , \mathbf{Y} , a permitted function $f \in \mathcal{F}$ over encrypted matrix.

Output: Computation result \mathbf{Z} and pre-process result $[[\mathbf{x}]]$, $[[\mathbf{X}]]$, $\mathbf{sk}_{f,y}$.

```

1 The authority initializes and distributes public key  $mpk_{FEIP}, mpk_{FEBO}$ 
2 function secure-computation( $[[x]]$ ,  $[[X]]$ ,  $f$ ,  $\mathcal{F}$ ,  $sk_{f,y/Y}$ ,  $Y$ )
3   initialize an empty matrix  $\mathbf{Z}$ 
4   if  $f \in \mathcal{F}$  and  $f$  is dot-product then
5     for  $i \leftarrow 0$  to size of  $sk_{f,y}$  do
6        $y := i$ -th row of  $\mathbf{Y}$ 
7       for  $j \leftarrow 0$  to size of  $[[x]]$  do
8          $Z[i][j] := FEIP.Decrypt(mpk_{FEIP}, x[j], sk_{f,y}[i], y)$ 
9   else
10    for  $i \leftarrow 0$  to row of  $[[X]]$  do
11      for  $j \leftarrow 0$  to column of  $[[X]]$  do
12         $Z[i][j] := FEBO.Decrypt(mpk_{FEBO}, sk_{f,Y}[i][j],$ 
13           $[[X]][i][j], \mathcal{F}, Y[i][j])$ 
14 return  $\mathbf{Z}$ 
15 function pre-process-encryption( $X$ )
16   initialize an empty list  $[[\mathbf{x}]]$  and an empty matrix  $[[\mathbf{X}]]$ 
17   for  $i \leftarrow 0$  to column size of  $X$  do
18      $x := i$ -th column of  $X$ 
19      $[[x]][i] := FEIP.Encrypt(mpk_{FEIP}, x)$ 
20     for  $j \leftarrow 0$  to row size of  $X$  do
21        $[[X]][j][i] := FEBO.Encrypt(mpk_{FEBO}, X[j][i])$ 
22 return  $[[\mathbf{x}]], [[\mathbf{X}]]$ 
23 function pre-process-key-derivative( $Y, f, \mathcal{F}$ )
24   initialize an empty list  $\mathbf{sk}_{f,y}$  and matrix  $\mathbf{sk}_{f,Y}$ 
25   for  $i \leftarrow 0$  to row size of  $Y$  do
26     if  $f \in \mathcal{F}$  and  $f$  is dot-product then
27        $y := i$ -th row of  $\mathbf{Y}$ 
28        $sk_{f,y}[i] := sk_f$  from authority by  $y$ 
29     else
30       for  $j \leftarrow 0$  to column size of  $Y$  do
31          $sk_{f,Y}[i][j] := sk_f$  from authority by  $Y[i][j]$ 
32 return  $\mathbf{sk}_{f,y}$  or  $\mathbf{sk}_{f,Y}$ 
```

图 4.6 CryptoNN 算法

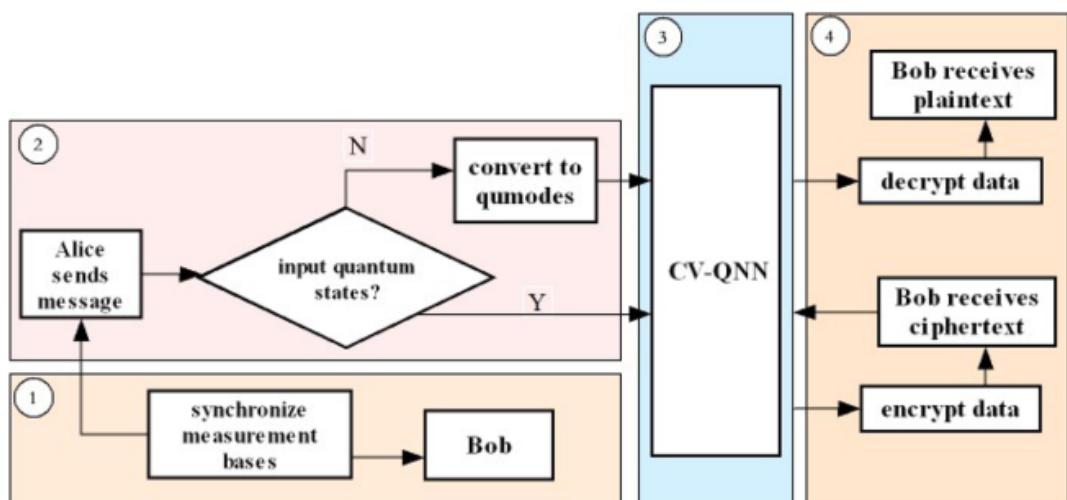


图 4.7 CV-QNN

第五章 工作总结

本次作业内容包含了讨论 Wireshark 网络抓包分析与量子计算信息安全、可信计算技术、深度学习等在密码学现况的学习总结报告，其第一章说明讨论 Wireshark 工具的使用与 HTTP 跟 HTTPS 的分析，同时尝试运用 PHP 跟 XAMPP 的工具进行实地测试，另外还使用 APACHE 伺服器在 Windows 环境下进行证书的设定，最后对 WiFi 探针进行说明。第二章与第三章则是将量子信息、可信计算技术与深度学习在密码学领域所搜集来的技术文章与文献进行整理，作为该科报告的总结。

参考文献

- [1] 邱锡鹏. 神经网络与深度学习[M/OL]. 北京: 机械工业出版社, 2020. [https://nndl.github.io/.](https://nndl.github.io/)

致谢

非常感谢我的导师朱跃生教授，在密码编码学与网络信息安全课让学生上充分实践了 Wireshark 的网路封包分析，同时也测试了 PHP 、 XAMPP 跟 HTTP 还有 HTTPS 等范围做出详细的测试，同时也将 WiFi 探针、量子信息、可信技术与深度学习等在密码学领域的研究文献与技术文章进行充分整理。最后感谢在这一年来一起寒窗苦读得同学与所有老师，还有默默在开源社群与前沿研究奉献的技术人员跟研究者们。