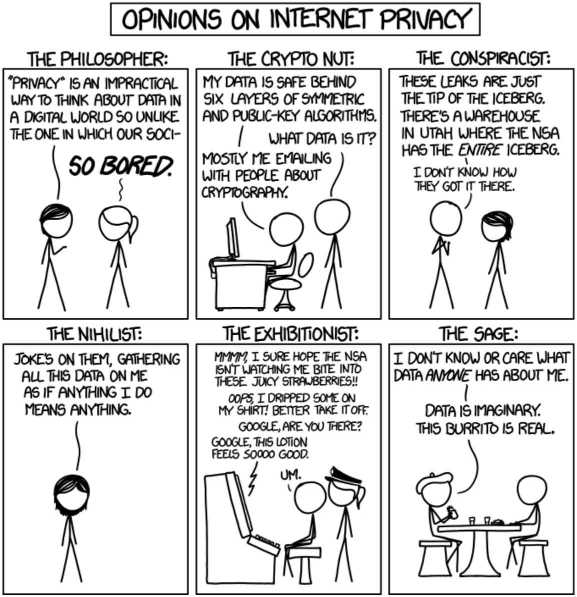
DOTA2024:8 Defense of the Ancients Eighth topic - Infrastructure

Hugh Anderson

National University of Singapore
School of Computing

July, 2024

Life... https://xkcd.com/1269/


OPINIONS ON INTERNET PRIVACY
THE PHILOSOPHER:
"PRIVACY" IS AN IMPRACTICAL WAY TO THINK ABOUT DATA IN A DIGITAL WORLD SO UNLIKE THE ONE IN WHICH OUR SOCI-
SO BORED.
THE CRYPTO NUT:
MY DATA IS SAFE BEHIND SIX LAYERS OF SYMMETRIC AND PUBLIC-KEY ALGORITHMS.
WHAT DATA IS IT?
MOSTLY ME EMAILING WITH PEOPLE ABOUT CRYPTOGRAPHY.
THE CONSPIRACIST:
THESE LEAKS ARE JUST THE TIP OF THE ICEBERG. THERE'S A WAREHOUSE IN UTAH WHERE THE NSA HAS THE ENTIRE ICEBERG.
I DON'T KNOW HOW THEY GOT IT THERE.
THE NIHILIST:
JOKE'S ON THEM, GATHERING ALL THIS DATA ON ME AS IF ANYTHING I DO MEANS ANYTHING.
THE EXHIBITIONIST:
MMMM, I SURE HOPE THE NSA ISN'T WATCHING ME BITE INTO THESE JUICY STRAWBERRIES!!
OOPS, I DRIPPED SOME ON MY SHIRT! BETTER TAKE IT OFF.
GOOGLE, ARE YOU THERE?
GOOGLE, THIS LOTION FEELS SOOOO GOOD.
UM.
THE SAGE:
I DON'T KNOW OR CARE WHAT DATA ANYONE HAS ABOUT ME.
DATA IS IMAGINARY. THIS BURRITO IS REAL.

Outline

1. **Authentication, logging in...**
   - Banking systems, Public key infrastructure
   - Authentication without passing passwords
2. **Other infrastructural building blocks**
   - Voting, coin flipping, oblivious transfer
   - Hardware ... IoT, DVD, DNS...

Case study 1: SWIFT

SWIFT is for sending money from bank to bank...

Society for Worldwide International Financial Telecommunications.

It was set up a long time ago (before PK and before PKI). The system had an unusual set of constraints. Banks did not trust the system (i.e. they did not want dishonest SWIFT people to steal from them), and authenticity was independent of confidentiality because some countries forbade cryptography (!)

Systems also had to enforce non-repudiation, and had to support the control of transactions under some policy or other (two-controls, balancing of books).

Now it is used over e-mail...

Case study 1: SWIFT

SWIFT mechanisms

- Authentication: Messages have a MAC (Message Authentication Code - a keyed hash function) attached. Banks would exchange Keys for the MAC in face-to-face meetings, or via third channels (certainly not via SWIFT). As a result SWIFT took no part in authentication.
- Non-repudiation: Countries log messages at a regional processing centre (RGP), before sending on to SWIFT. RGPs are more or less independent.
- Confidentiality: Used proprietary encryption devices, keys were hand-carried for distribution.
- Dual controls: Roles were separated - one clerk entered data, another checked and transmitted it.

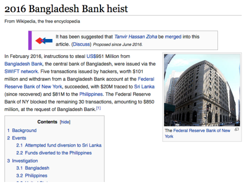
Case study 1: SWIFT

We saw this before... What has gone wrong?

For 20 years it ran well without any external fraud. Since 1996 SWIFT has used PK cryptography (so it may suffer from PK flaws or attacks).

Most frauds have an insider bypassing a dual control somehow. Stanley Rifkin, bank computer consultant, got hold of an authorization code, and used this over a telephone to authorize the purchase of $10M worth of diamonds.

In February 2016, attackers tried to steal US$951 Million from Bangladesh Bank, the central bank of Bangladesh, using the SWIFT network. Five transactions worth $101 million were withdrawn from a Bangladesh Bank account at the Federal Reserve Bank of New York, with $20M traced to Sri Lanka (since recovered) and $81M to the Philippines.



2016 Bangladesh Bank heist

From Wikipedia, the free encyclopedia

It has been suggested that *Tanvir Hassan Zoha* be merged into this article. (Discuss) *Proposed since June 2016.*

In February 2016, instructions to steal US$951 Million from Bangladesh Bank, the central bank of Bangladesh, were issued via the SWIFT network. Five transactions issued by hackers, worth $101 million and withdrawn from a Bangladesh Bank account at the Federal Reserve Bank of New York, succeeded, with $20M traced to Sri Lanka (since recovered) and $81M to the Philippines. The Federal Reserve Bank of NY blocked the remaining 30 transactions, amounting to $850 million, at the request of Bangladesh Bank.[1]

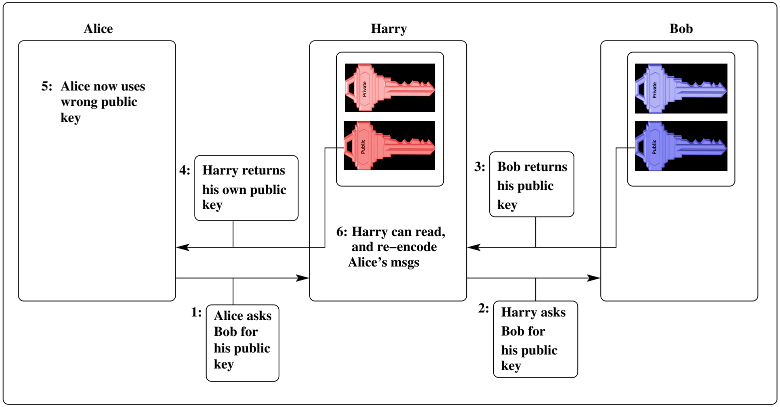
Contents [hide]

1 Background
2 Events
2.1 Attempted fund diversion to Sri Lanka
2.2 Funds diverted to the Philippines
3 Investigation
3.1 Bangladesh
3.2 Philippines

The Federal Reserve Bank of New York

Reminder: Man-in-the-middle for Public Keys

Motivation for PKI:



Alice
Harry
Bob
Private
Public
Private
Public
5: Alice now uses wrong public key
4: Harry returns his own public key
3: Bob returns his public key
6: Harry can read, and re–encode Alice's msgs
1: Alice asks Bob for his public key
2: Harry asks Bob for his public key

Reminder: Man-in-the-middle for Public Keys

Motivation for PKI:

How to ensure that keys belong to Bob?

Need some trusted authority/heirarchy to certify that keys are OK.

How to trust the Authority?

Authority's public credentials are already resident on your machine

What does the Authority do?

It digitally signs a certificate using the CA's private key. The certificate binds a public key and the identity of the user together.

Authorities: No single heirarchy:

Anyone or thing that you consider acceptable could sign a certificate.

For money and commercial interests, we want a bit more trust:

There are many certifying authorities - Verisign, Entrust and so on. Your browser comes pre-loaded with their public keys.

Case study 2: PKI

Summary:

The overall mechanism provides certificates that contain public keys, bound to identities (web sites for example). Certifying authorities provide some level of trust, providing certificates.

How is it used? ... In SSL for example

Main points...

- We have to trust (certifying) authorities.
- We may not know WHO our computer trusts.
- Bad certificates originally had no way of being discarded. Even now the work-around is awkward.

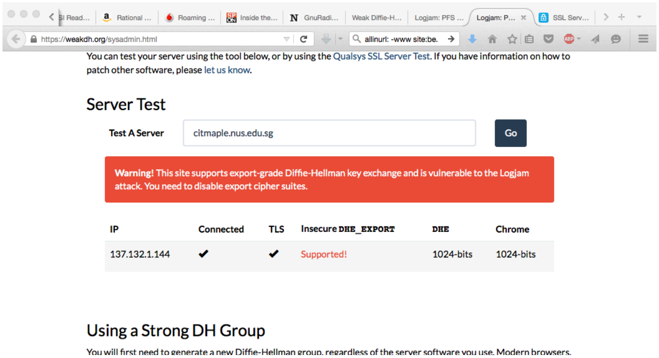
Case study 3: WeakDH

June 2013: Snowden releases NSA documents...

... and in them we discover that NSA has been routinely listening in on skype calls, and intercepting IPsec and https:// conversations.

My colleague Martin suggests that, in this day and age, most people couldn't care less. Maybe he is right. But in any case, how can it be done?

October 2015: Plausible explanation... https://weakdh.org/



You can test your server using the tool below, or by using the Qualsys SSL Server Test. If you have information on how to patch other software, please let us know.

Server Test

Test A Server: citmaple.nus.edu.sg [Go]

Warning! This site supports export-grade Diffie-Hellman key exchange and is vulnerable to the Logjam attack. You need to disable export cipher suites.

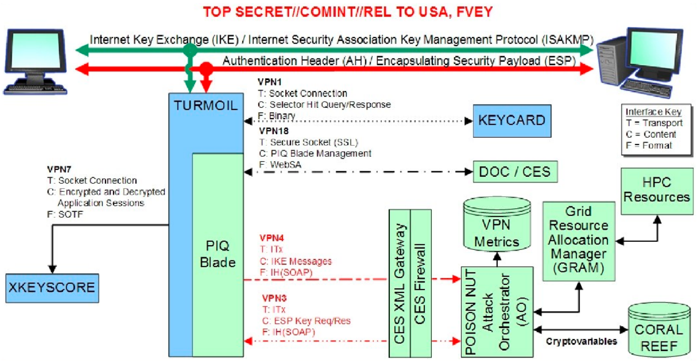
IP	Connected	TLS	Insecure DHE_EXPORT	DHE	Chrome
137.132.1.144	✓	✓	Supported!	1024-bits	1024-bits

Using a Strong DH Group

You will first need to generate a new Diffie-Hellman group, regardless of the server software you use. Modern browsers,

Case study 3: large scale monitoring of IPSec

Architecture of an NSA system from leaked document



Notice the diagram is clearly indicating the IKE (Internet Key Exchange), and the use of IPsec - the AH and ESP packets.
There is so much supporting material, that it is clear that this is not FUD, but is a real mechanism, in use since at least 2009.
It relies on the use of the fixed large primes used in Diffie-Hellman key exchange software.

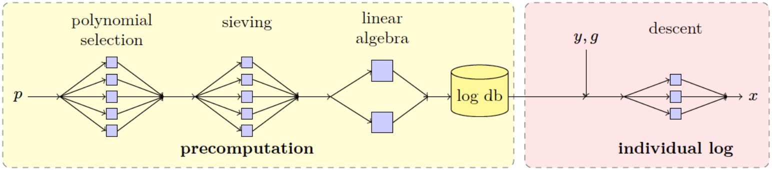
Case study 3: large scale monitoring of IPSec

What are the steps?

- During initial setup of the link, downgrade the Diffie-Hellman key exchange to one using smaller (EXPORT-GRADE) bitsize primes.
- Send IKE (Internet Key Exchange) values to supercomputer, which returns actual keys within 15 minutes.
  - October 2015 paper succeeds in 70 seconds: logjam - a mitm downgrade to 512 bit primes.

Solved $y = g^x \bmod p$ discrete log problem? ... No.

Common (NFS) algorithm can use pre-computation, for a specific prime, of 3 out of the 4 steps:



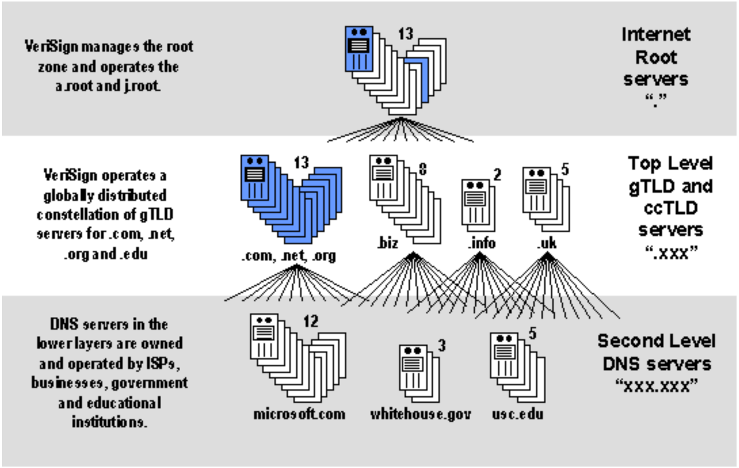
Case study 3: large scale monitoring of IPSec

Solutions? (From the paper)

- Browser - check you have the updates which stop *you* being vulnerable
- Server...
  - Disable Export Cipher Suites. There is little downside in disabling them.
  - Deploy (Ephemeral) Elliptic-Curve Diffie-Hellman (ECDHE). No known feasible cryptanalytic attacks
  - Use a Strong, Diffie Hellman Group. Use 2048-bit or stronger Diffie-Hellman groups with "safe" primes.

Case study 4: The DNS

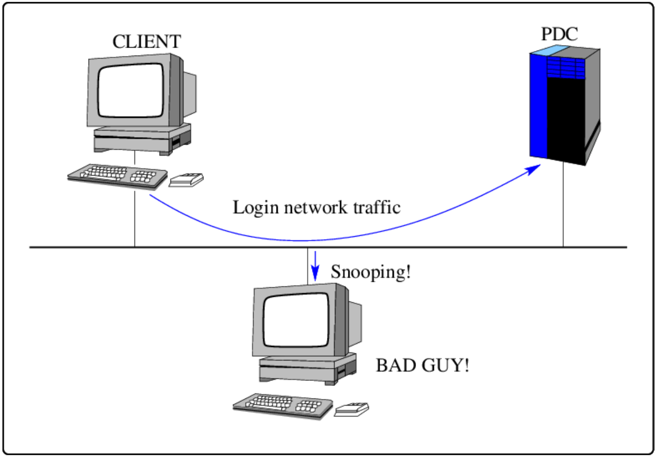
A heirarchy, root servers, domains, local...


VeriSign manages the root zone and operates the a.root and j.root
13
Internet Root servers "."
VeriSign operates a globally distributed constellation of gTLD servers for .com, .net, .org and .edu
13
.com, .net, .org
8
.biz
2
.info
5
.uk
Top Level gTLD and ccTLD servers ".xxx"
DNS servers in the lower layers are owned and operated by ISPs, businesses, government and educational institutions.
12
microsoft.com
3
whitehouse.gov
5
usc.edu
Second Level DNS servers "xxx.xxx"

Forms a tree, and a new request at a leaf has to traverse the tree.
A distributed database, with distributed caching of recent name requests.
Plenty of opportunity to inject in false entries.

Challenge response

Challenge response protocol for password checking...



Challenge-response protocol

When a client wishes to use a resource...

... it requests a connection and negotiates the protocol.

In the reply to this request the server generates and appends an 8 byte, random value - this is stored in the server after the reply is sent and is known as the ***challenge*** - different for every client connection.

The client then uses the hashed password (16 byte values described above), appended with 5 null bytes, as three 56 bit DES keys, to encrypt the challenge 8 byte value, forming a 24 byte value known as the ***response***. This calculation is done on *both* hashes of the user's password, and *both* responses are returned to the server, giving two 24 byte values.

Challenge-response protocol

The server then...

... reproduces the above calculation, using its own value of the 16 byte hashed password and the challenge value that it kept during the initial protocol negotiation.

It then checks to see if the 24 byte value it calculates matches the 24 byte value returned to it from the client. If these values match exactly, then the client knew the correct password and is allowed access.

There are good points about this:

- The server never knows or stores the *cleartext* of the users password - just the 16 byte hashed values derived from it.
- The cleartext password or 16 byte hashed values are never transmitted over the network - thus increasing security.

Challenge-response protocol

However, there is also a bad side:

- The 16 byte hashed values are a "password equivalent". You cannot derive the users password from them, but they can be used in a modified client to gain access to a server.
- The initial protocol negotiation is generally insecure, and can be hijacked in a range of ways. One common hijack involves convincing the server to allow clear-text passwords.
- Despite functionality added to NT to protect unauthorized access to the SAM, the mechanism is trivially insecure

Attacks on Windows (NT) systems...

Even *without* network access, it is possible by various means to access the SAM password hashes, and *with* network access it is easy.

The attack considered here is the use of either a dictionary, or brute force attack directly on the password hashes (which must be first collected somehow).

Kerberos/Cerberus

What is it?

Kerberos is a network ***authentication*** protocol. It provides strong authentication for client/server applications using symmetric or public key cryptography.

Kerberos is freely available in source form, and also is in commercial products (Windows uses Kerberos for login).

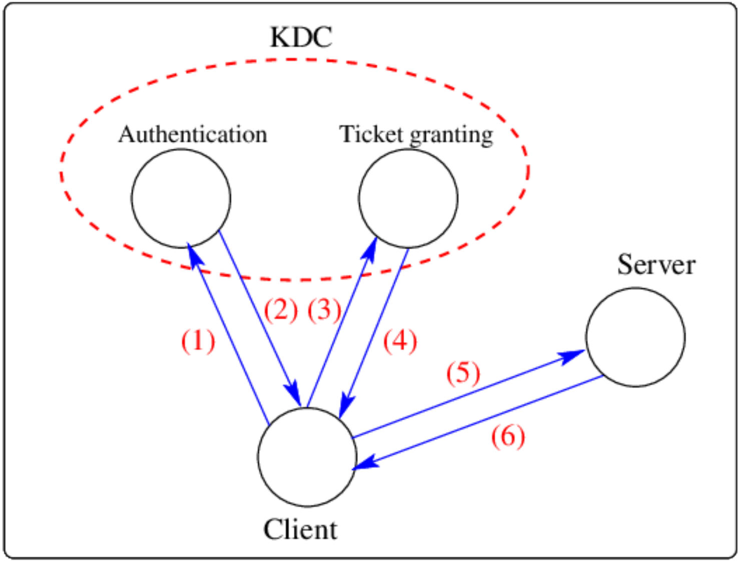
Authentication... and encryption.

Client and server use Kerberos to prove their identity, and encrypt all of their communications. There is a Key Distribution Center (KDC)

Kerberos uses a Needham-Schroeder symmetric key protocol.

Kerberos

Sequence of messages for granting tickets



Kerberos

Notation

A *key* $K_{x,y}$ is a session key shared by both x and y.
When we encrypt a message M using the key $K_{x,y}$ we write it as $E(K_{x,y}, M)$.

Sequence of messages for granting tickets

When a client first authenticates to Kerberos, she:

1. Talks to KDC, to get a *Ticket Granting Ticket* .
2. Uses that to talk to the *Ticket Granting Service* and get a particular ticket for a particular service.
3. Uses that ticket, to interact with the server.

This way a user doesn't have to reenter passwords every time they wish to connect to a Kerberized service.

Note that if the Ticket Granting Ticket is compromised, an attacker can only masquerade as a user until the ticket expires.

Kerberos protocol

Two sorts of credentials: tickets and authenticators:

A *ticket* $T_{c,s}$ contains the client's name and network address, the server's name, a timestamp and a session key,encrypted with the server's secret key (so that the client is unable to modify it).

An *authenticator* $A_{c,s}$ contains the client's name, a timestamp and an optional extra session key, encrypted with the session key shared between the client and the server.

Alice wants session key for communication with Bob:

1. Alice sends message to Ted containing her identity, Ted's TGS identity, and one-time value $(n) : \{a, tgs, n\}$.
2. Ted responds with a key encrypted with Alice's secret key (which Ted knows), and a ticket encrypted with the TGS secret key:
   $E(K_a, \{K_{a,tgs}, n\})$ $E(K_{tgs}, T_{a,tgs})$.
   Alice now has ticket and session key: $E(K_{tgs}, T_{a,tgs})$, $K_{a,tgs}$
3. Alice can prove her identity to the TGS, as she has session key $K_{a,tgs}$, and *Ticket Granting Ticket*: $E(K_{tgs}, T_{a,tgs})$.

Kerberos protocol

Later, Alice can ask the TGS for a specific service ticket:

1. When Alice wants a ticket for a specific service (say with Bob), she sends an *authenticator* along with the *Ticket Granting Ticket* to the TGS: $E(K_{a,tgs}, A_{a,b})\ E(K_{tgs}, T_{a,tgs}), b, n$.
2. The TGS responds with a suitable key and a ticket: $E(K_{a,tgs}, \{K_{a,b}, n\})\ E(K_b, T_{a,b})$.
3. Alice can now use an authenticator and ticket directly with Bob: $E(K_{a,b}, A_{a,b})\ E(K_b, T_{a,b})$.

Properties of the protocol - weaknesses -

Host security: Kerberos makes no provisions for host security; it assumes that it is running on *trusted* hosts with an *untrusted* network.

KDC compromises: Kerberos uses a principal's password (encryption key) as the fundamental proof of identity.

Salt: This is an additional input to the one-way hash algorithm.

Voting protocols

A voting protocol is one in which...

...independent systems vote in a kind of election..

We may want to put conditions on this. For example, afterwards we might want to check that the vote was correct, or that each voter only got a single vote. Systems should be corruption-proof.

Voting systems are commonly found in computer systems. For example, if we had multiple servers being used for redundancy, then the servers might vote to see who is in charge.

Too-simple voting systems may be subject to attack. For example, a protocol like first-to-the-post may result in always choosing only one server, or even worse, be tricked by a fast-responder. (Consider the DHCP attack in the “Topic5” videos).

Voting protocols

Voting using public key encryption...

If Alice votes v_A, then she broadcasts:

$$R_A(R_B(R_C(E_A(E_B(E_C(v_A))))))$$

where

1. R_A is a random encoding function which adds a random string to a message before encrypting it with A's public key:

   $$R_A(m) \equiv E(K_{A_{\text{pub}}}, m + rand())$$

2. E_A is public key encryption with A's public key: $E_A(m) \equiv E(K_{A_{\text{pub}}}, m)$.

Each voter then engages in a process of signing and decrypting and rebroadcasting one level of every message/vote.

At the end of the protocol, each voter has a complete signed audit trail and is ensured of the validity of the vote.

First round...

One round...

Who	Receives and removes one level...		and sends on (with sigs)...
Alice:	$R^1_A(R^1_B(R^1_C(E^1_A(E^1_B(E^1_C(v_1))))))$	$\rightarrow$	$R^1_B(R^1_C(E^1_A(E^1_B(E^1_C(v_1)))))$
	$R^2_A(R^2_B(R^2_C(E^2_A(E^2_B(E^2_C(v_2))))))$	$\rightarrow$	$R^2_B(R^2_C(E^2_A(E^2_B(E^2_C(v_2)))))$
	$R^3_A(R^3_B(R^3_C(E^3_A(E^3_B(E^3_C(v_3))))))$	$\rightarrow$	$R^3_B(R^3_C(E^3_A(E^3_B(E^3_C(v_3)))))$
Bob:	$R^1_B(R^1_C(E^1_A(E^1_B(E^1_C(v_1)))))$	$\rightarrow$	$R^1_C(E^1_A(E^1_B(E^1_C(v_1))))$
	$R^2_B(R^2_C(E^2_A(E^2_B(E^2_C(v_2)))))$	$\rightarrow$	$R^2_C(E^2_A(E^2_B(E^2_C(v_2))))$
	$R^3_B(R^3_C(E^3_A(E^3_B(E^3_C(v_3)))))$	$\rightarrow$	$R^3_C(E^3_A(E^3_B(E^3_C(v_3))))$
Charles:	$R^1_C(E^1_A(E^1_B(E^1_C(v_1))))$	$\rightarrow$	$E^1_A(E^1_B(E^1_C(v_1)))$
	$R^2_C(E^2_A(E^2_B(E^2_C(v_2))))$	$\rightarrow$	$E^2_A(E^2_B(E^2_C(v_2)))$
	$R^3_C(E^3_A(E^3_B(E^3_C(v_3))))$	$\rightarrow$	$E^3_A(E^3_B(E^3_C(v_3)))$

In the first round (after 3 transfers) - each voter has agreed that their vote has been counted. If not, they do not continue.

To show agreement, voters sign the votes they forward.

Second round...

Then actually discover the votes...

Who	Receives and removes one level...		and sends on (with sigs)...
Alice:	$E_A^1(E_B^1(E_C^1(v_1)))$	$\rightarrow$	$E_B^1(E_C^1(v_1))$
	$E_A^2(E_B^2(E_C^2(v_2)))$	$\rightarrow$	$E_B^2(E_C^2(v_2))$
	$E_A^3(E_B^3(E_C^3(v_3)))$	$\rightarrow$	$E_B^3(E_C^3(v_3))$
Bob:	$E_B^1(E_C^1(v_1))$	$\rightarrow$	$E_C^1(v_1)$
	$E_B^2(E_C^2(v_2))$	$\rightarrow$	$E_C^2(v_2)$
	$E_B^3(E_C^3(v_3))$	$\rightarrow$	$E_C^3(v_3)$
Charles:	$E_C^1(v_1)$	$\rightarrow$	v_1
	$E_C^2(v_2)$	$\rightarrow$	v_2
	$E_C^3(v_3)$	$\rightarrow$	v_3

Only Alice can remove her level of encryption, but anyone can check that it was done correctly (by re-encrypting).

In the second round, someone can tamper with the vote, but at the end each vote can be re-encrypted, and checked against the set of signatures. The tamperer will be found.

Tossing a coin

Lets pretend we are gambling...

Alice and Bob want to toss a coin, so adopt the following first steps:

1. Alice calculates two primes p, q and calculates $N = pq$, sends N to Bob. $N = 35 = 5 * 7$
2. If Bob can factorize the number, then Bob wins a coin toss.
3. Bob selects random x, and sends $y = x^2 \text{ mod } N$ to Alice. $y = 31^2 \text{ mod } 35 = 16$
4. Alice calculates the four square roots of 16:

$$\begin{array}{lclclcl} 4^2 \text{ mod } 35 & = & 16 & \quad & 31^2 \text{ mod } 35 & = & 16 \\ 24^2 \text{ mod } 35 & = & 16 & \quad & 11^2 \text{ mod } 35 & = & 16 \end{array}$$

This is easy for Alice, as she knows the prime factors of N. She then sends one of these back to Bob.

Tossing a coin

So what does Bob do?

If Bob receives x or $-x$, then he learns nothing, but if Bob receives either of the other values:

$$\begin{aligned} \mathrm{GCD}(24+31,35) &= \mathrm{GCD}(55,35) \\ &= 5 \end{aligned}$$

Bob can calculate a factor, and Alice is unable to tell she has divulged the factor.

Note that...

... If you know two different square roots of a number modulo $N = pq$, you can find the factors of N.

Coin flipping — Issue #1

Alice can compute square roots knowing p, q?

Yes. When Alice receives the value $y = x^2 \bmod n$, she calculates $\sqrt{y} \bmod p$ and $\sqrt{y} \bmod q$. She then uses these two values in the Chinese Remainder Theorem to find a root. Using the example in class, $y = 16$, and Alice knows $p = 5$ and $q = 7$. She computes $\sqrt{16} \bmod 5$ and $\sqrt{16} \bmod 7$, noting that $16 = 1 \bmod 5$ and $16 = 2 \bmod 7$:

$$\begin{aligned} \sqrt{1} \bmod 5 &= \pm 1 \\ \sqrt{2} \bmod 7 &= \pm 4 \end{aligned}$$

and then CRT helps find x:

$$\begin{array}{rclcrclcrcl} x \bmod 5 &=& 1 & \& & x \bmod 7 &=& 4 & \Longrightarrow & x &=& 11 \\ x \bmod 5 &=& 1 & \& & x \bmod 7 &=& -4 & \Longrightarrow & x &=& 31 \end{array}$$

We have the four square roots of $16 \bmod 35$: ± 11 and ± 31.

Example of a square root algorithm modulo an odd prime...

If $p \equiv 3 \bmod 4$ then $x = y^{\frac{p+1}{4}} \bmod p$.

For example, if $p = 3851$ and $y = 3298$, then $x = 3298^{\frac{3852}{4}} \bmod 3851 = 231$.

Contract signing and oblivious transfer

Scenario: signing contracts can be difficult:

If one party signs the contract, the other may not. We have one party bound by the contract, and the other not.

In addition, both may sign, and then one may say "*I didn't sign any contract!*" afterwards.

Use oblivious transfer...

In an oblivious transfer, randomness is used to convince participants of the fairness of some transaction

---

In the coin-tossing example, Alice knows the prime factors of a large number, and if Bob can factorize the number, then Bob wins a coin toss. Alice will either divulge one of the prime factors to Bob, or not, with equal probability. Alice is unable to tell if she has divulged the factor, and so the coin toss is fair.

Alice is "oblivious" - she is unaware if she has revealed a secret.

Contract signing

Use oblivious transfer for contract signing...

Oblivious transfer used for contract-signing where the following points are important:

- Up to a certain point neither party is bound
- After that point both parties are bound
- Either party can prove that the other party signed

Alice and Bob pre-agree on some probability difference that they will live with (say 5%), and then exchange (say 1000) signed oblivious-transfer messages, each possibly transfering a bit of information. They are becoming “more” bound by a contract with ever-increasing probability - perhaps each successfully received bit is worth 1% to the receiver.

What if someone tries to terminate early?

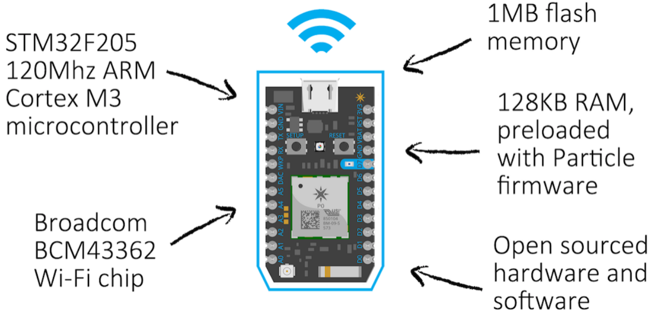
In the event of early termination of the contract, either party can take the messages they have to an adjudicator, who chooses a random probability value (42% say) before looking at the messages.

If messages are over 42% then both parties are bound. If less then both parties are free.

(a): IoT

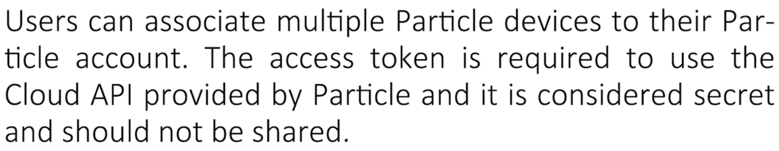
A well known IoT device...

The **Particle Photon** is a popular IoT development kit that is relatively inexpensive, at US$19.

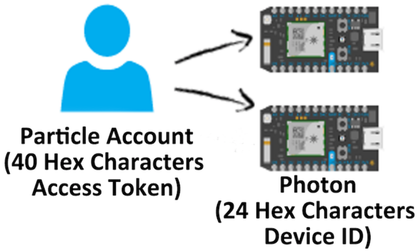

STM32F205 120Mhz ARM Cortex M3 microcontroller
Broadcom BCM43362 Wi-Fi chip
1MB flash memory
128KB RAM, preloaded with Particle firmware
Open sourced hardware and software

(a): IoT

Single token needed for access...



Users can associate multiple Particle devices to their Particle account. The access token is required to use the Cloud API provided by Particle and it is considered secret and should not be shared.



(a): Accessing the IoT device

The device access is via the “Particle cloud API”

Whenever a user sends an instruction to the photon, the user has to supply a device ID, and the access token. The Particle cloud will authenticate the token and device ID before relaying instructions to the device.

The access token is all-powerful! With the token, you can list out all the device IDs. i.e. you do not need to know the device ID to send instructions. As a result, getting the access token ... gets you the photon!

Acquiring the access token...

Students identified at mechanisms to get an access:

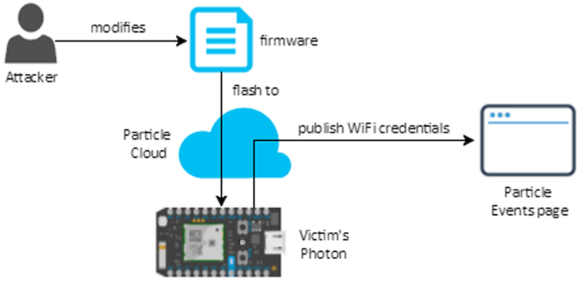
- Social Engineering
- Man-in-the-middle attacks

They were effective mechanisms, and could be used to get the access token for any number of photons, nearby or remote. The students even found access tokens through google!

(a): IoT

Single token needed for access...

An attacker can **build a modified firmware** that **publishes the stored Wi-Fi credentials**. The attacker can then flash the firmware using the Particle Cloud onto the Photon device. Firmware flashing takes only a short while and **does not overwrite the current user application** installed on the device. This allows the attacker to compromise the Photon device **without disrupting its usual functions.**



(a): An attack on IoT and photons

My students engineered this nasty attack

And they are talking to the manufacturers to come up with a resolution. What can be done to mitigate it?

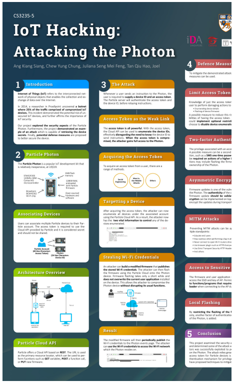
- Limit the access token capabilities.
- Two factor authentication
- Authenticate the clients who initiate software updates...

The firmware steals WiFi credentials for the remote network near the photon. This is obviously pretty serious.

.
BAD STUDENTS!

(a): IoT

Poster and papers, responsible disclosure...



CS3235-5

IoT Hacking: Attacking the Photon

Ang Kiang Siang, Chew Yung Chung, Juliana Seng Mei Feng, Tan Qiu Hao, Joel

1 Introduction

Internet of Things (IoT) refers to the interconnected network of physical objects that enables the collection and exchange of data over the Internet.

In 2014, a researcher in Proofpoint uncovered **a botnet where 25% of the traffic comprised of compromised IoT devices.** This incident demonstrated the potential risk of unsecured IoT devices, and further affirms the importance of IoT security.

Our project **explored the security aspects** of the Particle Photon. Furthermore, the project **demonstrated an example of an attack** which is capable of **retrieving the device secrets**. Finally, **potential defense measures** are proposed to better secure the device.

2 Particle Photon

The **Particle Photon** is a popular IoT development kit that is relatively inexpensive, at US$19.

STM32F205 120Mhz ARM Cortex M3 microcontroller
Broadcom BCM43362 Wi-Fi chip
1MB flash memory
128KB RAM, preloaded with Particle firmware
Open sourced hardware and software

Associating Devices

Users can associate multiple Particle devices to their Particle account. The access token is required to use the Cloud API provided by Particle and it is considered secret and should not be shared.

Particle Account (40 Hex Characters Access Token)
Photon (24 Hex Characters Device ID)

Architecture Overview

Particle Cloud API

Particle offers a Cloud API based on **REST**. The URL is used as the primary resource locator, which can be used to perform functions such as **GET** variables, **POST** a function call, or **PUT** new firmware.

3 The Attack

Whenever a user sends an instruction to the Photon, the user is required to **supply a device ID and an access token.** The Particle server will authenticate the access token and the device ID, before relaying instructions.

Access Token as the Weak Link

The **access token is all powerful.** With the access token, the Cloud API can be used to **enumerate the device IDs,** effectively **disregarding the need to know** the device ID to send instructions. **When the access token is compromised, the attacker gains full access to the Photon.**

Acquiring the Access Token

To acquire an access token from a user, there are a range of methods.

Targetting a Device

After acquiring the access token, the attacker can now enumerate all devices under the associated account using the Particle Cloud API. As a result, the attacker now has the **two vital information to control** any of the devices enumerated.

Stealing Wi-Fi Credentials

An attacker can **build a modified firmware** that **publishes the stored Wi-Fi credentials.** The attacker can then flash the firmware using the Particle Cloud onto the Photon device. Firmware flashing takes only a short while and **does not overwrite the current user application** installed on the device. This allows the attacker to compromise the Photon device **without disrupting its usual functions.**

Result

The modified firmware will then **periodically publish** the Wi-Fi credentials to the Photon events page. The attacker can **use the Wi-Fi credentials to access the Wi-Fi network** which the Photon resides on.

4 Defence Measur

To mitigate the demonstrated attack
measures can be used.

Limit Access Token

Knowledge of just the access token
user to perform damaging actions to

- Enumerating device details
- Flashing of device firmware

A possible measure to reduce this ri
bilities of having the access token.
could **implement optional restricti**
choose to **disable device enumeratio**

Two-factor Authenti

The privilege associated with an acce
A possible measure can be a second
tion, such as a **SMS one-time passw**
be **required on actions of a higher l**
tions may include flashing the firmw
ownership of the Photon.

Asymmetric Encryp

Firmware updates is one of the vuln
the Photon. The **authenticity** of the
firmware update **should be valida**
cryption can be implemented on top
encrypt the updates during transpor

MITM Attacks

Preventing MITM attacks can be a
tiple standpoints:

- Educate end users
- Stay cautious when performing a log in at
- Never connect to open Wi-Fi routers dire
- Use browser plugin such as HTTPS Everyw
- Use Strict Transport Security HTTP Heade
- And others

Access to Sensitive

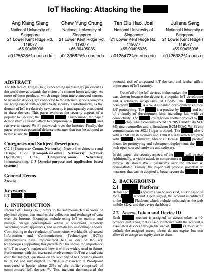
The firmware and user application
access the SSID and key of AP. Permi
to functions/programs that require
loader when connecting to the AP d

Local Flashing

By **restricting the flashing of the f**
only, another factor of authenticatio
of the Photon, is added.

5 Conclusion

This project examined the security a
and determined some of the attack v
tors was successfully exploited to p
on the Photon. The attack relies grea
access token for Particle devices is
thentication mechanism for privileg
have proposed techniques to mitigat



IoT Hacking: Attacking the [redacted]

Ang Kiang Siang, National University of Singapore, 21 Lower Kent Ridge Rd, 119077, +65 90495036, a0125528@u.nus.edu
Chew Yung Chung, National University of Singapore, 21 Lower Kent Ridge Rd, 119077, +65 90495036, a0133662@u.nus.edu
Tan Qiu Hao, Joel, National University of Singapore, 21 Lower Kent Ridge Rd, 119077, +65 90495036, a0125473@u.nus.edu
Juliana Seng, National University o Singapore, 21 Lower Kent Ridge F 119077, +65 90495036, a0126332@u.nus.e

ABSTRACT

The Internet of Things (IoT) is becoming increasingly prevalent as the world moves towards the vision of a smarter home and city. As more of these products, which range from interconnected sensors to wearable devices, get connected to the Internet, serious concerns are being raised with regards to its security. Unfortunately, as the domain of IoT is relatively new, security is inadequately considered in these devices. This paper explores the security aspects of a popular IoT device, the [redacted] [redacted]. Furthermore, the paper demonstrates a viable attack to compromise a [redacted] [redacted], and retrieve its stored Wi-Fi passwords over the Internet. Finally, the paper proposes potential defense measures that can be adopted to better secure the [redacted] [redacted].

Categories and Subject Descriptors

C.2.1 [**Computer-Comm. Networks**]: Network Architecture and Design; C.2.3 [**Computer-Comm. Networks**]: Network Operations; C.2.6 [**Computer-Comm. Networks**]: Internetworking; C.3 [**Special-purpose and application based systems**]

General Terms

Security.

Keywords

IoT, [redacted] [redacted]

1. INTRODUCTION

Internet of Things (IoT) refers to the interconnected network of physical objects that enables the collection and exchange of data over the Internet. Examples include using IoT to monitor and manage energy consumption within a household, remotely switching on/off appliances, and automatically unlocking of doors. Contributing to the revolution of smart cities worldwide, advanced Information and Communication Technologies (ICTs) infrastructures have implemented IoT as one of the key technologies supporting this growth.[6] This shows the importance of IoT in today's market and how it will be widely used in future. Furthermore, with this increased involvement of IoT on critical data over the Internet, questions on the security of IoT devices should be raised and investigated. In 2014, a researcher in Proofpoint uncovered a botnet where 25% of the traffic comprised of compromised IoT devices [5]. This incident demonstrated the potential risk of unsecured IoT devices, and further affirm
importance of IoT security.

Out of all of the IoT devices in the market, the [redacted] [redacted]
was chosen because the device is a popular IoT developme
and is relatively inexpensive, at US$19. The [redacted] [redacted]
henceforth [redacted], is a Wi-Fi enabled development kit mea
IoT prototyping. The [redacted] is a product by [redacted], and is
of a family of development kits, including kits with ce
connectivity. The [redacted] leverages on another product by [redacted]
a [redacted] chip, which contains a STM32F205 120Mhz ARM C
M3 microcontroller and a Broadcom BCM43362 Wi-Fi chi
communicates on 802.11b/g/n protocol. The [redacted] also c
with a 1MB flash memory and 128KB RAM which are prel
with [redacted]'s firmware libraries. Because [redacted] product
meant for prototyping and subsequent deployment, the [redacted]
both open sourced hardware and software.

In this paper, the security aspects of the [redacted] will be expl
Additionally, a viable attack to compromise a [redacted] [redacted]
retrieve its stored Wi-Fi passwords over the Internet wi
demonstrated. Finally, the paper will propose potential de
measures that can be adopted to better secure the [redacted] [redacted]

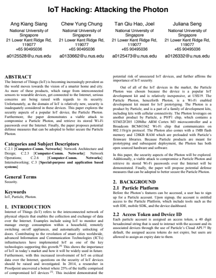
2. BACKGROUND

2.1 [redacted] Platform

Before the [redacted]'s features can be accessed, a user has to si
for a [redacted] account. Upon signup, the account is entitled a
to the [redacted] Platform, which include tools such as the web
mobile SDK, and the device dashboard.

2.2 Access Token and Device ID

Each [redacted] account is assigned an access token, a 40
hexadecimal string that is used to interact with the account a
associated devices through the use of [redacted]'s Cloud API.
default, the assigned access tokens do not expire, but use
allowed to assign an expiry date to them.



IoT Hacking: Attacking the Photon

Ang Kiang Siang, National University of Singapore, 21 Lower Kent Ridge Rd, 119077, +65 90495036, a0125528@u.nus.edu
Chew Yung Chung, National University of Singapore, 21 Lower Kent Ridge Rd, 119077, +65 90495036, a0133662@u.nus.edu
Tan Qiu Hao, Joel, National University of Singapore, 21 Lower Kent Ridge Rd, 119077, +65 90495036, a0125473@u.nus.edu
Juliana Seng, National University of Singapore, 21 Lower Kent Ridge Rd, 119077, +65 90495036, a0126332@u.nus.edu

ABSTRACT

The Internet of Things (IoT) is becoming increasingly prevalent as the world moves towards the vision of a smarter home and city. As more of these products, which range from interconnected sensors to wearable devices, get connected to the Internet, serious concerns are being raised with regards to its security. Unfortunately, as the domain of IoT is relatively new, security is inadequately considered in these devices. This paper explores the security aspects of a popular IoT device, the Particle Photon. Furthermore, the paper demonstrates a viable attack to compromise a Particle Photon, and retrieve its stored Wi-Fi passwords over the Internet. Finally, the paper proposes potential defense measures that can be adopted to better secure the Particle Photon.

Categories and Subject Descriptors

C.2.1 [**Computer-Comm. Networks**]: Network Architecture and Design; C.2.3 [**Computer-Comm. Networks**]: Network Operations; C.2.6 [**Computer-Comm. Networks**]: Internetworking; C.3 [**Special-purpose and application based systems**]

General Terms

Security.

Keywords

IoT, Particle, Photon.

1. INTRODUCTION

Internet of Things (IoT) refers to the interconnected network of physical objects that enables the collection and exchange of data over the Internet. Examples include using IoT to monitor and manage energy consumption within a household, remotely switching on/off appliances, and automatically unlocking of doors. Contributing to the revolution of smart cities worldwide, advanced Information and Communication Technologies (ICTs) infrastructures have implemented IoT as one of the key technologies supporting this growth.[6] This shows the importance of IoT in today's market and how it will be widely used in future. Furthermore, with this increased involvement of IoT on critical data over the Internet, questions on the security of IoT devices should be raised and investigated. In 2014, a researcher in Proofpoint uncovered a botnet where 25% of the traffic comprised of compromised IoT devices [5]. This incident demonstrated the potential risk of unsecured IoT devices, and further affirms the importance of IoT security.

Out of all of the IoT devices in the market, the Particle Photon was chosen because the device is a popular IoT development kit and is relatively inexpensive, at US$19. The Particle Photon, henceforth Photon, is a Wi-Fi enabled development kit meant for IoT prototyping. The Photon is a product by Particle, and is a part of a family of development kits, including kits with cellular connectivity. The Photon leverages on another product by Particle, a P0/P1 chip, which contains a STM32F205 120Mhz ARM Cortex M3 microcontroller and a Broadcom BCM43362 Wi-Fi chip that communicates on 802.11b/g/n protocol. The Photon also comes with a 1MB flash memory and 128KB RAM which are preloaded with Particle's firmware libraries. Because Particle products are meant for prototyping and subsequent deployment, the Photon has both open sourced hardware and software.

In this paper, the security aspects of the Photon will be explored. Additionally, a viable attack to compromise a Particle Photon and retrieve its stored Wi-Fi passwords over the Internet will be demonstrated. Finally, the paper will propose potential defense measures that can be adopted to better secure the Particle Photon.

2. BACKGROUND

2.1 Particle Platform

Before the Photon's features can be accessed, a user has to sign up for a Particle account. Upon signup, the account is entitled access to the Particle Platform, which include tools such as the web IDE, mobile SDK, and the device dashboard.

2.2 Access Token and Device ID

Each particle account is assigned an access token, a 40 digit hexadecimal string that is used to interact with the account and its associated devices through the use of Particle's Cloud API.[3] By default, the assigned access tokens do not expire, but users are allowed to assign an expiry date to them.

(b): Media and DVD security

What is the point of DVD regions?

The DVD CSS is a Content Scrambling System - data encryption scheme, developed by commercial interests to stop copying... but it is easy to copy a DVD: CSS only prevents decrypting, changing and re-recording.

The CSS algorithm details are a trade secret. There is a master set of 400 keys stored on every DVD, and the DVD player uses these to generate a key needed to decrypt data from the disc.

LINUX and the CSS

Linux users were excluded from access to CSS licenses because of the open-source nature of Linux.

The source code for decoding DVDs is available on a T-shirt, because in October 1999, hobbyists/hackers in Europe cracked the CSS algorithm.

Since then DVD industry players have been trying to prevent distribution of any software

(b): DVD security

What do we learn?

The lesson to learn from this is that *security-through-obscurity* is a very poor strategy.
The source code and detailed descriptions for a CSS descrambler is available at: `http://www-2.cs.cmu.edu/~dst/DeCSS/Gallery/`

Description of the key/descrambling process:

First one must have a master key, which is unique to the DVD player manufacturer. It is also known as a player key. The player reads an encrypted disk key from the DVD, and uses its player key to decrypt the disk key. Then the player reads the encrypted title key for the file to be played. (The DVD will likely contain multiple files, typically 4 to 8, each with its own title key.) It uses the decrypted disk key (DK) to decrypt the title key. Finally, the decrypted title key, TK, is used to descramble the actual content.

(b): DVD security

Confusion and diffusion...

```
#define m(i)(x[i]^s[i+84])<<
unsigned char x[5],y,s[2048];main(n){for(read(0,x,5);read(0,s,n=2048);
write(1,s ,n))if(s[y=s[13]%8+20]/16%4==1){int i=m(1)17^256+m(0)8,k=m(2)
0,j=m(4)17^m(3)9^k *2-k%8^8,a=0,c=26;for(s[y]-=16;--c;j*=2)
a=a*2^i&1,i=i/2^j&1<<24;for(j=127;++j<n ;c=c>y)c+=y=i^i/8^i>>4^
i>>12,i=i>>8^y<<17,a^=a>>14,y=a^a*8^a<<6,a=a>>8^y<<9,k=s [j],k="7
Wo~'G_\216"[k&7]+2^"cr3sfw6v;*k+>/n."[k>>4]*2^k*257/8,s[j]=k^(k&k
*2&34) *6^c+~y;}}
```

Summary

Many opportunities for manipulation:

1. We looked at four large infrastructural systems:
    1. SWIFT, a worlwide banking mechanism. Happily there have only been isolated attacks, and it is still largely working.
    2. PKI, which has a history of worrysome attacks, with BAD certificates issued by “authorities”.
    3. How the (US) NSA has been messing with Diffie Hellman, to do large scale monitoring of supposedly secure https and IKE.
    4. The DNS.
2. We looked at the general idea of challenge-response, to perform authentication without having to reveal passwords, along with Kerberos, a strong challenge-response system.
3. We then looked at some other infrastructural things, and how they might be done:
    1. Voting systems
    2. Coin flipping
    3. contract signing with oblivious transfer.
4. Finally, we looked at some hardware - IoT devices, and DVDs, and the security in them (or perhaps lack of security in them).