Exercise 3 go over it

Exercise 2 – frequency analysis isn't magic, don't expect to get meaningful results on the first shot… try it and change it to get the results we want…

Not effective on shorter text in comparison to longer ones – no magic

Don't assume it is Caeser cipher always, think of it as scrambled and not constant shift

Previous lectures:

- Symmetric and asymmetric cryptography
    - Public, private keys
- Caser cipher
- Letter substitution cipher
    - More random – harder to break more keys
- Message authentication code
- Hash functions
- Digital certificates

Shannon in info theory

- What is good crypto
- One time pad
    - Ultimate crypto

Presentation 4:

Key length and security

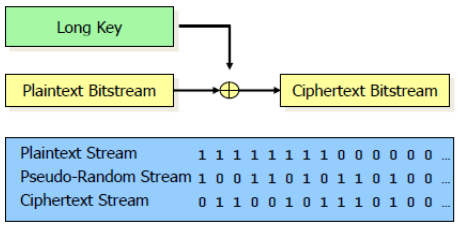
Crypto protocols

Random number generators

One-time pads

- Take plaintext in bit string and XOR with key (random sequence)
- Then we get the cipher text bitstream

One Time Pads

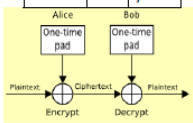


Xor again with the key to get the plaintext back easily

One Time Pads

- Perfect crypto !?
  - Every crypto text can be generated by every plaintext
  - Crypto text gives NO information on plain
- But:
  - The key stream must be completely random
  - Used only once (see here why)
  - Known only to sender and receiver (and is VERY long)

Cipher	Key	Plain
c	1	b
c	2	a
c	3	z
c	4	y



https://goo.gl/kCQlUT

One time pad is symmetric – can cause issues

C in cipher, with 1 key it means different things

- Very good

Every text we generate via every plaintext

No information can get from cipher text about plaintext

Very long key, as long as message

Hard to generate truly random messages (for key)

Method to delivery key to bob can be used to deliver message

Both key and message are secret

Key is used only once

Known only to sender and receiver (and is VERY long)

Use same key for two messages – get 2 random sequences but since same key is used can get back the original message via XOR to get this mirrored image which contains message itself

One Time Pads in practice

- How to generate completely random stream?
- Project Venona: how to make sure OTP is not reused?
- The key is as long as the message, how to distribute them?

Russians used it in WW2 often

- Used a lot and had problems with generating random numbers

- Actually used same keys and US intel. Noticed that and could reveal lots of secrets
- Revealed spies

Distributed keys via those small books

Can bypass asymmetric communication – so not always a good idea

Use diplomatic mail – deliver lots of letters and is confidential and low tech

Key Length

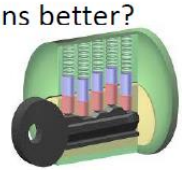
Longer key == stronger security?

5 pins with 10 positions = 100 000 keys

Trying all of them – 69h on average

– 5 sec per key

Is a lock with 7 pins and 12 positions better?



10^5

Longer the key = stronger security – hard to break

Brute force – find the key

12^7– sounds better

Key length

- Consider brute force
  - DES with 56 bits - 72,057,594,037,927,936 keys
  - EFF DES cracker - 56 hours
  - Cloudcracker does it in less than a day

DES 56

EFF built super cracker

Now its much faster

Break code with moore's law

- Need longer keys with more bits

Key entropy

- But no one is using 56 bits anymore
- AES 128 or even 256 bits is recommended
- How are the keys generated?
- Entropy – measure of uncertainty
  - 2^{128} is a maximum, attackers looks for a minimum
  - How many of 128 bits are random?

https://flic.kr/p/kVoeN

It can be broken but not yet

How the keys are effective?

- 56 bit key but is it truly random in 56 bits and how to measure randomness of key is via entropy
  - It gives us the measure of uncertainty / randomness we have in the key

Ex: pick someone in the street and ask someone if they pick a gender of someone 50/50 (m/f)

If we want to speak in entropy , instead of probabilities… we have 2 options … 1 in entropy it will be 1

- Idea on average is use 1 bit to give information .. but has 0 and 1
- Can use 1 bit to tell us the uncertainty in sex is 50/50
- 1 bit 0 for male and 1 for female
  - Info needed to understand the chance of probability

Idea of how many bits we need in order to tell us uncertainty in sequence

Key is 56 length – entropy in bits needs to be the same?

- Max in key is 56 but less if we find patterns in sequence gives us entropy in sequence

Attackers will find patterns to see if key can be broken with less

Key length

- Password as a key:
  - 10 characters = 80 bits
  - No high bit
  - Letter frequency
  - < 4 bits of entropy per char
  - Dictionary attacks

ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	n	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	-	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

Is it really 80 bits?

Not all bits will be used

2^8 bits and we use 80 only

1st bit is same for all letters

We can use 7 bits now

Passwords usually are easier to remember vs random strings

- Names, colors, etc

Use frequency analysis to overcome this barrier

- Now we are left with less than 4 bits

Entropy from 8 bits its < 4 bits

- 4 bits for 1 character
- Less than 40

Now we can use dictionary attacks – easier to attack

- From larger than age of universe to less than a minute

In practice, its hard to implement

Key length

Randomly generated keys

- Random number generator quality (more on that later)

Weak algorithm that uses only part of the key

- A5/1 64 bit encryption is broken in 2^{40} time
- Microsoft's MS-CHAP instead of 2^{128} actually $2*2^{56}$

A5 is cell communications

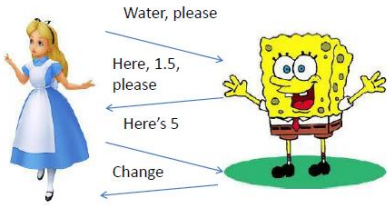
- From 64 bits , entropy less than 30 bits
  - Easy to crack

MS chap – Microsoft – way to generate keys

- 2^56 takes about a day to break

- Protocols how to implement security practices
- Human protocols - the rules followed in human interactions
- –Example: Asking a question in class
- Raising hand and pick a student to get permission to give answer (used a lot)
- •Networking protocols - rules followed in networked communication systems
- –Examples: HTTP, FTP, etc.
-
- •Security protocol - the (communication) rules followed in a security application
- –Examples: SSL
-

Building a real life protocol



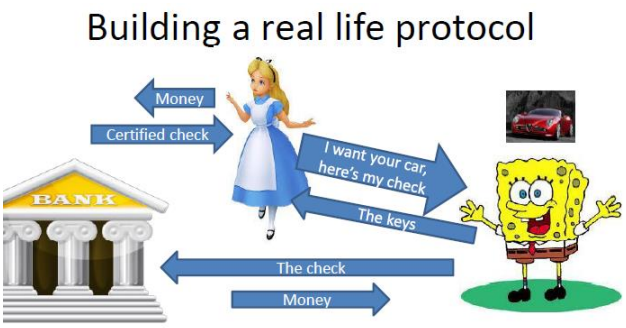
- Is it secure?
  - Bob peeking in Alice wallet
  - Bob selling fake products
  - Bob running with the change
  - Alice paying with fake money
  - Alice pulling a gun and robbing Bob
  - …

Bob wants to sell water to alice

- Protocol to how to do it
- Secure?
  - No change
  - Bob can use fake water
  - Run away

Protocol to buy expensive things

-
- Use a trusted 3rd party
- Mutual trust
- Is car reliable
    - Get it checked out by mechanic



Bank is a **trusted third party**
- Will not steal or spend the money
- Will honor the check

Expect bank to honor check, wont steal or spend $$

MAC, Hash, digital signature = building block for crypto protocols

Digital signature

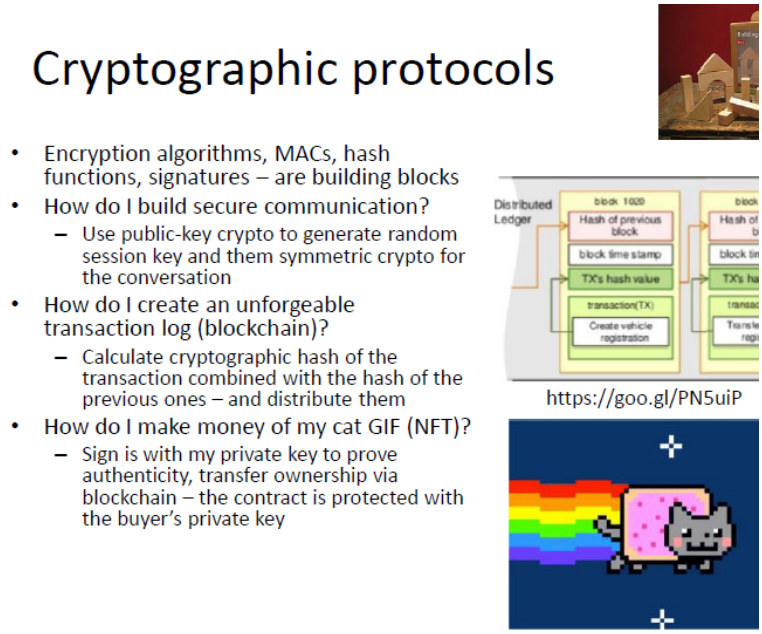
- Asymmetric crypto
- Uses private key – signature

Hash function – gives us integrity

- If we change plaintext, we get different hash

MAC – authentication

Secure comms.



Symmetric is much faster than asymmetric

Why do we need to make a log?

Blockchain – authentication and integrity

- If log is changed we will know its been changed
- About $ its sensitive… if someone changed the figures we want to know
- Solution: transactions in past = hash them and take a newer transaction – hash it then put it in a chain

- Uses MAC algorithm

People making money from GIFs

- Sold for half a million dollars

These are autographs – how to sign the image? With private key

- Put on blockchain to show chain of ownership

SSL protocol

Alice connects to server (no public key or anything , no encryption) – alice -> bob

- Alice communicates with bob
  - Sends hello
  - And the cipher she knows (algorithms and some math)
  - Bob then says hello
  - And picks one algorithm that alice uses (bob has a lot of algorithms) and also attaches certificate to algorithm
  - And now alice knows that bob is truly bob
    - In certificate
      - Signature of CA cert authority – signed over bob's certificate (same as a bank)
      - And bob's public key
      - Now alice picks up a symmetric key encrypted with bobs public key
        - From asymmetric to symmetric communication
      - Bob now talks encrypted with symmetric key

alice takes services from bob, so alice doesn't need to approve anything

Firewalls have this issue

Crypto protocols around us

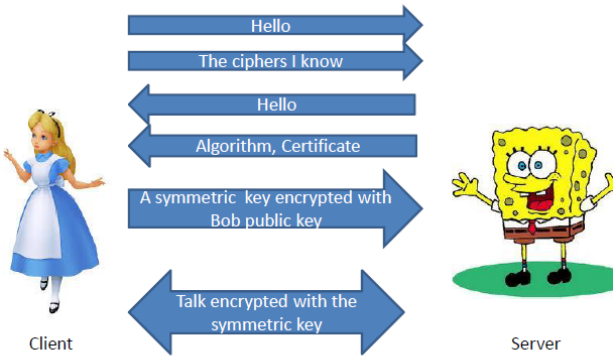
- Any communication needs a protocol
  - Browser – Web Server
  - Computer – Domain Server
  - Cellphone – Base Station
  - Remote Control – Car
  - Smart Card – Set-top box

Choosing a protocol/algorithm

- How do you know it is good?
- No way to 100% prove it is secure
  - Formal verification might help one day
- Either break or fail trying…
- Need many experts over many years
  - Discrete logarithms break thru in 2013
  - New ways to attack RSA key exchange in TLS - 2018
- **Prefer old and public over new and proprietary**

The 9 Lives of Bleichenbacher's CAT: New Cache ATtacks on TLS Implementations

Simplified SSL



How to chose a protocol and how do you know if it is good?

- Lots of people use it
- Can prove parts of protocols but not the whole thing
- Try to break it
- RSA – broken by Israeli comp sci.

Avoid security via obscurity

Ipsec – used by vpns

- Communication protocol
- Broken but improved and released a proper version

But Microsoft

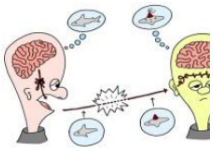
- Not open source
- Point to point tunneling

Win 95 and 98 – from security perspective = failure

- Tried to fix and patch but made it worse

Public or Proprietary

IPSec
- Designed in 1992
- Analyzed, broken, changed
- Draft in 1995, debates, discussion
- 1998 – revised official version



https://flic.kr/p/8VBLrM

PPTP (Microsoft)
- Same thing, all reinvented: hashes, authentication, key generation
- Used in NT, 95 and 98
- All of then hacked

Proprietary
- Security thru obscurity
- Patents prevent experts from looking

We've got patents, secured, etc … do we believe or trust them? No, because protected by patent… processes to break and change it

Obscurity fails

Obscurity fails

- All proprietary algorithms got reversed:
  - DVD encryption
  - FireWire encryption
- Public algorithms should stay secure despite being public
- Compare this to a proprietary medicine…

-
- Hacker groups cracked the algorithm easily
- Dvd used own crypto which wasn't strong

- Russians an german crackers
- Published algorithm – reverse engineering
  - Jon Lech Johansen published algo – arrested and trial
    - Got nothing
      - Because he did it for his own dvds that he bought not illegal

New medicine to cure cancer

- Would we try it?
- Want stats and proof that it works

Don't Roll Your Own Crypto!


Ismail Kizir
CTO at Hohha Internet Services Ltd.
Follow

An "unbreakable encryption algorithm"

Nov 16, 2015 | 248 views 2 Likes 2 Comments

Finally, I've discovered an "unbreakable encryption algorithm" I call "Hohha Dynamic XOR Encryption".
It's based on a model that I've never seen before: "The key itself changes by the parameters of the data being encrypted or decrypted; which makes it practically unbrekable". And it is quite fast: ~%80 faster than the fastest mode of AES 256,

I am not a "crypto expert", I haven't mathematical background to prove that's "secure", as the "authorities" do!
But I believe in common sense and collaboration. I'd rather prefer to try to improve transparently a collaborative work, than relying on suspicious "Cryptoanalysis Gods" and "Authorities"!

-
- AES inst so fast , he suggested an alternative one
- People took him seriously
- Known plaintext attack is **a scenario in which the attacker has access to pairs of known plaintexts and their corresponding ciphertexts**. The goal is to guess the secret key (or a number of secret keys) or to develop an algorithm which would allow him to decrypt any further messages.

Protocol attacks

- Passive (eavesdropping)
  - HTTP
  - Radio
  - Not detectable!
  - Metadata/SIGINT
- Active
  - Changing messages
  - Inserting
  - Deleting
  - Replaying

AES and DES – long bit keys

Encryption algo with long keys

But if the key is longer, will it be safer?

- Strong and protected door… the person will try the window to break in or bypass protections

Passive attacks

- Just listening
- http is exposed, just need to find a way to get between two parties to collect info
- radio
- hard to detect because wont change anything
- signal intelligence
- can know that ariel speaks with ahron and when they speak even though we don't know what they are speaking about

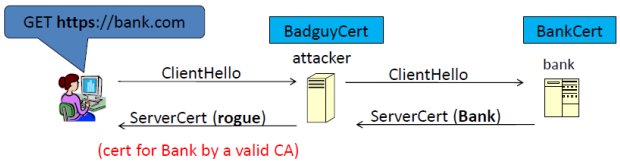
Active attacks

- can change even though not understood
  - modify it too
- replay , can record the key and in a few minutes can use it ourselves even though we don't know what the key is

most famous one is MITM – man in the middle

Man In The Middle

- Fake ATM example
- How do you know Bob is Bob?
- Many protocols prone to this attack
- Certificate pinning solves this
  - Specific server cert
  - Specific CA cert
  - ... see the next slide...



The hacker intercepts and gets your pin and card number etc

Client connecting to bank but really gets to attacker

Attacker is passive

- client sends hello to bank but it gets to attacker first -> passes message to bank
- bank sends back the protocol (certificate) but attacker gets it before
  - and passes to client (knows bank communicates with client)
  - just looks at data transport

uses fake protocol – then alice (client)

hacker gets bank protocol – public key, digital signature of the CA

but doesn't pass it to alice

attacker can get a good certificate that's similar update it , and fool alice

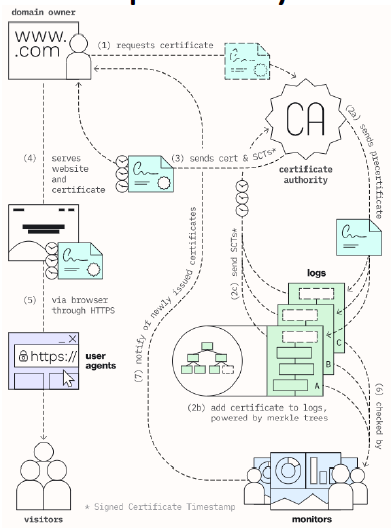
now use attacker public key and encrypt using public key -> connect to attacker -> give password and other info to use it against alice

how to handle the attack – be sensitive

- for small cert change, don't accept it
  - certificate pinning

Certificate Transparency

- Certificate pinning problems:
  - CA's manage of the certificates
  - Products (browsers and OS) decide which CA's to use
    - Hard to build the set that work
  - Errors make site unusable
    - User fatigue instead of safety
- Certificate Transparency
  - Blockchain for certificates
    - Public, cryptographically-assured, monitored, validated
    - All new certificates submitted to the log before returning
  - Signed Timestamps returned from the logs and are included in the certificates
    - Log has 24 hours to check the certificate and include it
  - Browser will not accept a certificate without SCT
    - The certificate should appear in the next log
      - Not there – invalid



Random Number Generators

- Almost every cryptographic scheme needs random numbers: for keys, IVs, etc.
- If these are not random – the scheme is easily compromised
- 600 000$ casino loss due to bad random number generator

important but hard to get

In April 1994, Daniel Corriveau won $600,000 CAD playing keno. He picked 19 of the 20 winning numbers three times in a row. Corriveau claims he used a computer to discern a pattern in the sequence of numbers, based on chaos theory. However, it was later found that the sequence was easy to predict because the casino was using an inadequate electronic pseudorandom number generator. In fact, the keno machine was reset every morning with the same seed number, resulting in the same sequence of numbers being generated. Corriveau received his winnings after investigators cleared him of any wrongdoing

Same number sequence every day

Randomness

- How do we get random numbers from a deterministic computer?
- Instead of truly random we need:
  - Unpredictable
  - Irreproducible
- Use external data: network packets arrival, microphone noise, mouse movements
- Using external sources as seeds to an algorithm

how to get random numbers

- unpredictable
- cannot repeat themselves

how to get random numbers -> from external resources

take camera and extract position of lava lamp balls

famous way:

- lottery machine with balls/bingo – not computers

Pseudo randomness is hard

- 2007 – easy way to predict all random SSL values on Windows
- 2008 – openssl developer removes unnecesary code. Certificate key, SSL session keys, OpenSSH keys must be regenerated
- 2012 – lots of RSA keys can be discovered due to bad randomness
- 2013 – Java RNG flaw used to steal Bitcoins on Android
- 2015 – Unknown hackers change Juniper RNG to get ability to decrypt all VPN traffic

examples of how people find ways to attack SSL , RSA with pseudo-random values

Openssl – compiler pointed a var isn't initialized so he gave it a 0

Pseudo randomness is hard

```
--- openssl-0.9.7e.orig/crypto/rand/rand_unix.c 2003-12-27 16:01:52.000000000
+0000
+++ openssl-0.9.7e/crypto/rand/rand_unix.c      2006-04-19 15:42:32.000000000
+0100
@@ -160,6 +160,9 @@
     const char **egdsocket = NULL;
 #endif

+    /* Keep valgrind happy */
+    memset(tmpbuf, 0, sizeof tmpbuf);
+
 #ifdef DEVRANDOM
     /* Use a random entropy pool device. Linux, FreeBSD and OpenBSD
      * have this. Use /dev/urandom if you can as /dev/random may block
```

Responsible for random numbers, so computer gave it garbage value

Hw is hands on

Python overview:

- easy to learn, dynamic language
- OOp
- Great libraries for data science
- In comparison to matlab,