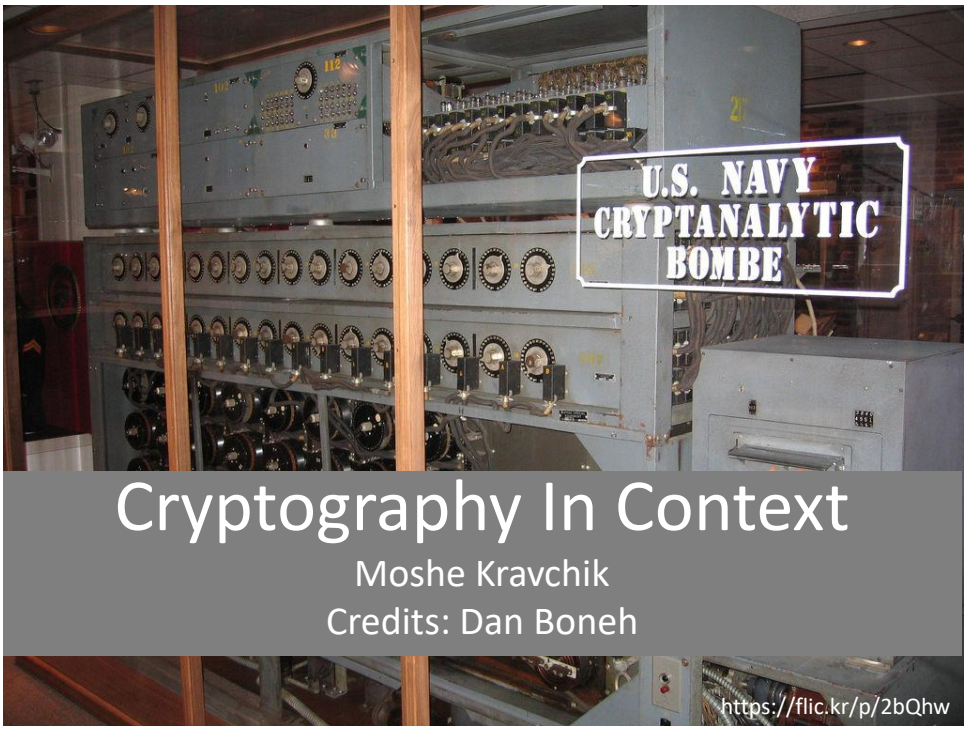


Cryptography In Context

Moshe Kravchik
Credits: Dan Boneh

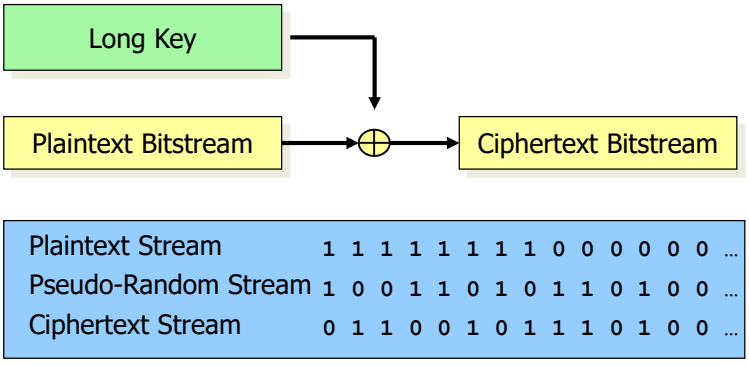
1

Agenda

- One Time Pads
- Key Length and Security
- Cryptographic protocols
- Protocol attacks
- Random number generators

2

One Time Pads

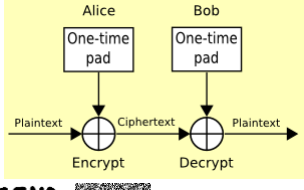


3

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

4

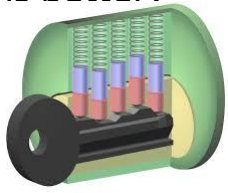
One Time Pads in practice

- 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?

5

Key Length

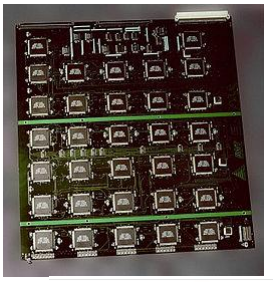
- Longer key == stronger security?
- 5 pins with 10 positions = 100 000 keys
- Trying all of them – 69h on average
  - 5 sec per key
- Is a lock with 7 pins and 12 positions better?



6

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



7

Key entropy

- But no one is using 56 bits anymore
- AES 128 or even 256 bits is recommended
- How the keys are 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

8

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	m	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

9

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 $2*2^{56}$

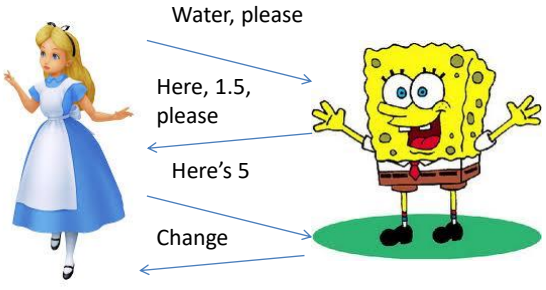
10

Protocol

- Human protocols — the rules followed in human interactions
  - Example: Asking a question in class
- Networking protocols — rules followed in networked communication systems
  - Examples: HTTP, FTP, etc.
- Security protocol — the (communication) rules followed in a security application
  - Examples: SSL

11

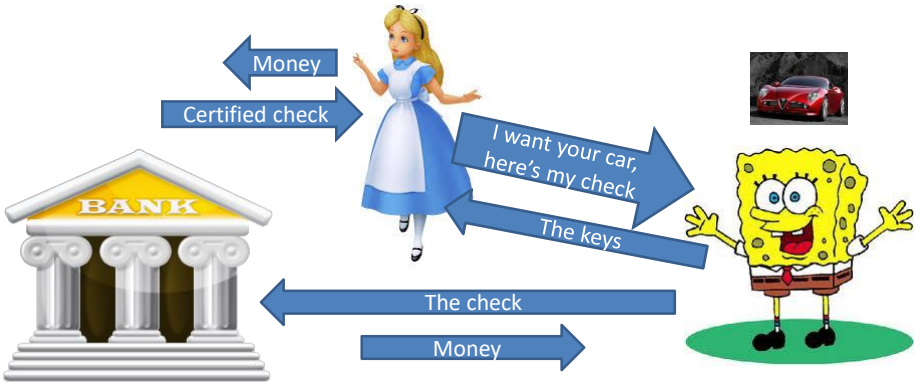
Building a real life protocol



- 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
  - …

12

Building a real life protocol



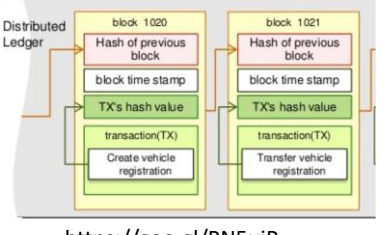
Bank is a trusted third party

- Will not steal or spend the money
- Will honor the check

13

Cryptographic protocols

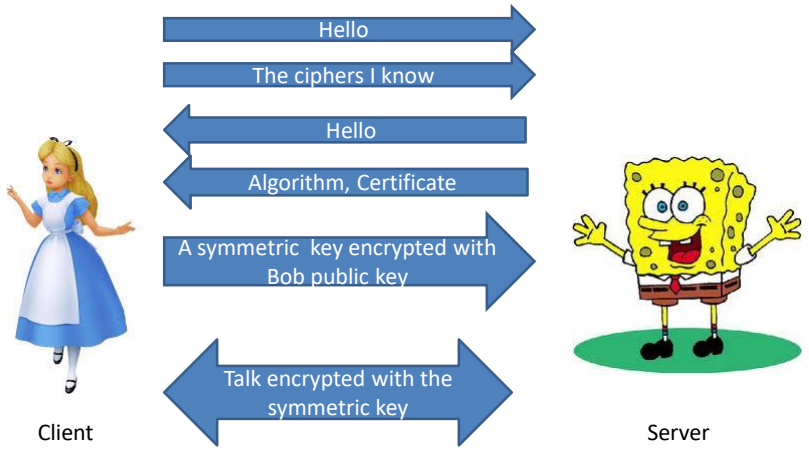
- Encryption algorithms, MACs, hash functions, signatures – are building blocks
- How do build a secure chat?
  - Use public-key crypto to generate random session key and them symmetric crypto for the conversation
- How do I create an unforgeable transaction log?
  - Calculate cryptographic hash of the transaction combined with the hash of the previous ones



https://goo.gl/PN5uiP

14

Simplified SSL



15

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

17

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**

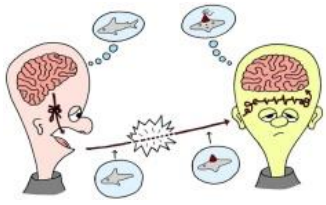

https://flic.kr/p/m71BmQ

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

19

Public or Proprietary

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


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

20

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…

21

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 unbreakable". 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"!

22

Don't Roll Your Own Crypto!

>>> Briefly, to decypher a ciphertext, a cracker needs to find out the key, and, to find out the key, cracker needs to find out the plaintext, because the key is dynamically updated according to plaintext and the jump path is chosen accoding to plaintext+encrypted text during encryption process; Probably not impossible, in theory, but in practice very difficult!
So it's vulnerable to a known plaintext attack?

> There are 2 factors which influences security:
> 1. Key length L
> 2. Number of jumps J

Actually, there are many more. Key space, _effective_ key space, basic operation (encryption or decryption) complexity, key setup complexity, block size, ...

Adhokshaj Mishra
Independent Security Researcher

A primary analysis of your code and algorithm reveals that:

1. The cryptosystem is poor one. Output is not so random. And yeah, I performed statistical tests on output.
2. There are many possibilities of side channel attacks.

Both issues make it almost useless for any actual use.

Most people with experience in cryptanalysis have better things to do. You wouldn't rely on a bridge built by amateurs either? Lack of proof it is bad just isn't sufficient. Proof of that it is good is the bar to reach.

23

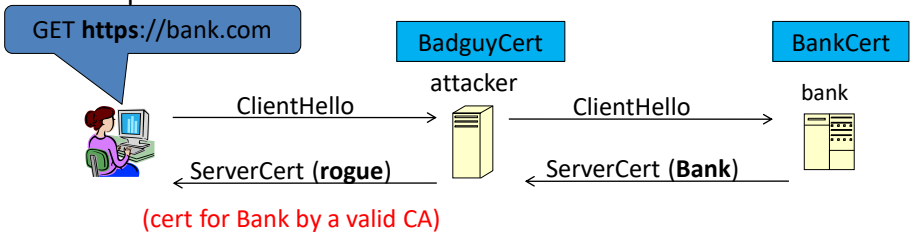
Protocol attacks

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

24

Man In The Middle

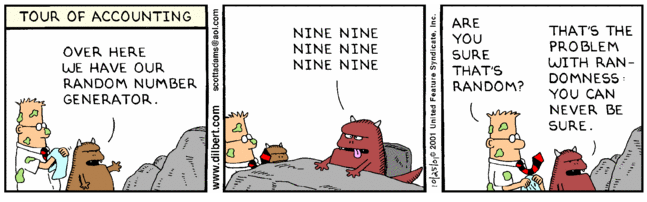
- Fake ATM example
- How do you know Bob is Bob?
- Many protocols prone to this attack
- Certificate pinning solves this
  - Specific server cert
  - Specific CA cert



25

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



26

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

27

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

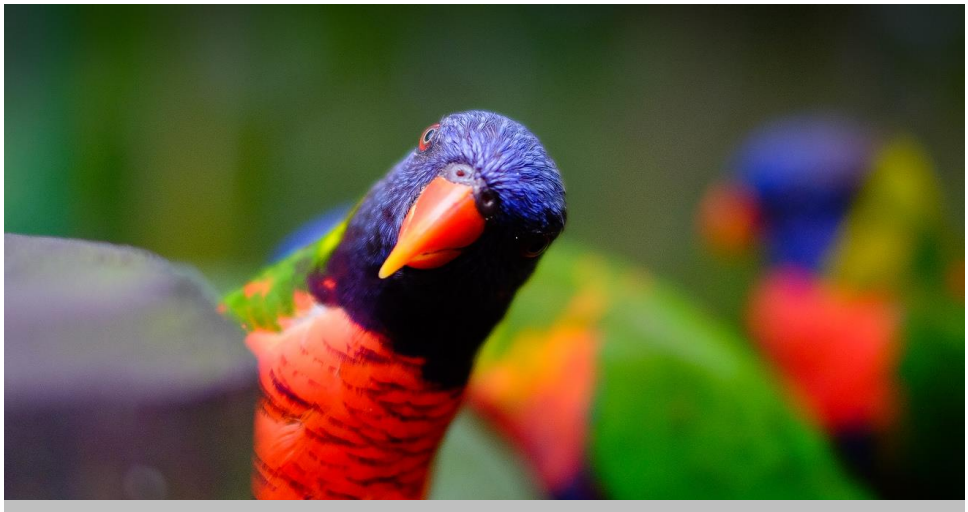
28

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
```

29



Questions?

30

Terms learnt

- One time pad
- Entropy
- Protocol
- Trusted Third party
- Man-In-The-Middle
- Random number generator

31

Summary

- Don’t confuse theoretical and practical security strength
- Longer key are not automatically more secure
- Building protocols from crypto blocks
- Prefer old and public protocols
- It’s hard to be unpredictable!

32