Security Vulnerabilities

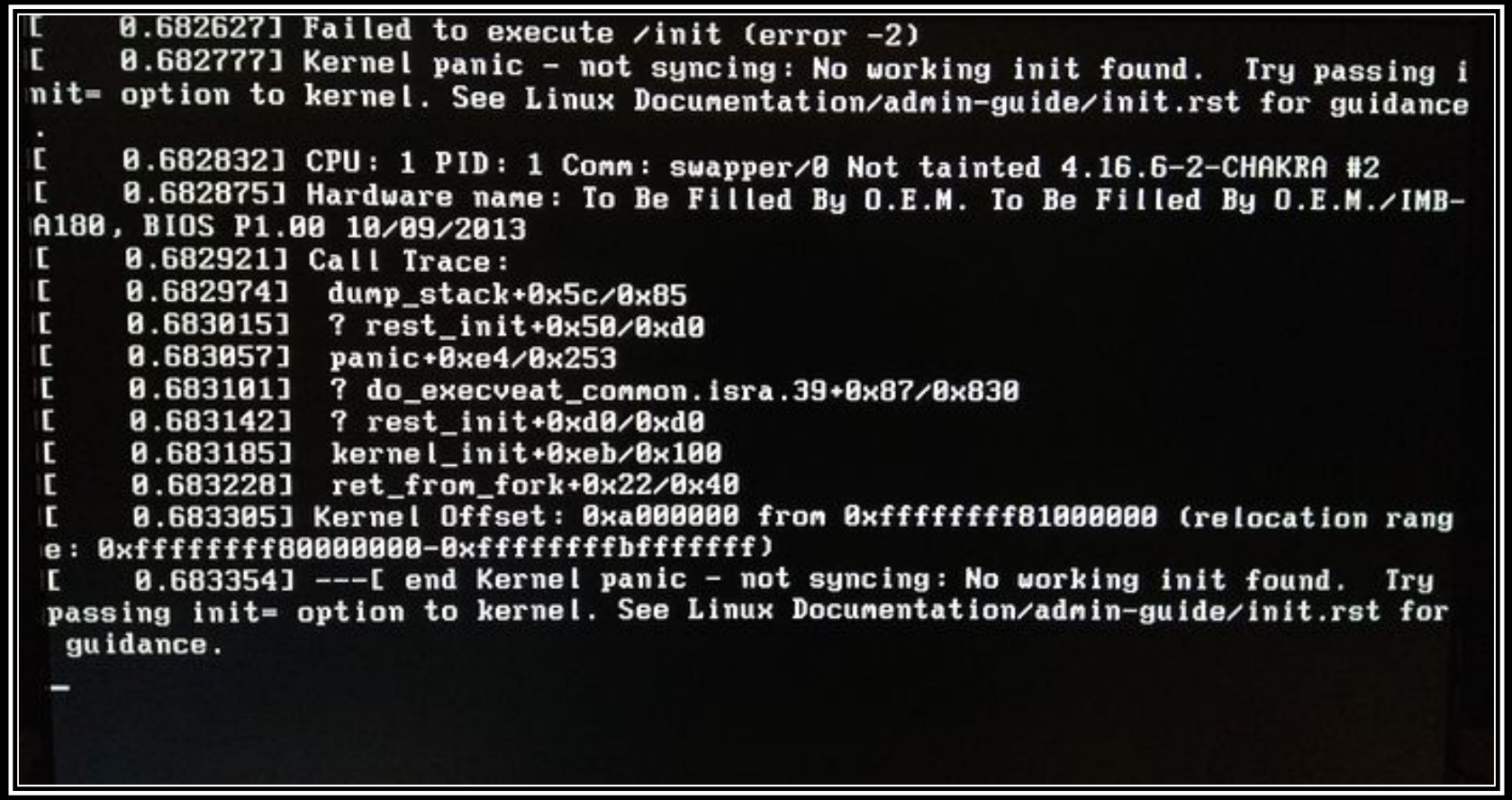
The devil is in the details

Errors, Bugs, and Failures

- Computers are composed of hardware whose behavior is determined by software (roughly…)
- Applications run on operating systems and interoperated through protocols
- Hardware and software are developed by humans and therefore aren’t perfect
- A human **error** may introduce a **bug** (or fault)
    - The IEEE Standard Glossary of Software Engineering Terminology defines “fault” as “an incorrect step, process, or data definition in computer program”
- When a fault gets triggers, it might generate a **failure**


Windows
A fatal exception 0E has occured at F0AD:42494C4C
the current application will be terminated.
* Press any key to terminate the current application.
* Press CTRL+ALT+DELETE again to restart your computer.
You will lose any unsaved information in all applications.
Press any key to continue


You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.
Veuillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.
Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.
コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。


[0.682627] Failed to execute /init (error -2)
[0.682777] Kernel panic - not syncing: No working init found. Try passing init= option to kernel. See Linux Documentation/admin-guide/init.rst for guidance.
[0.682832] CPU: 1 PID: 1 Comm: swapper/0 Not tainted 4.16.6-2-CHAKRA #2
[0.682875] Hardware name: To Be Filled By O.E.M. To Be Filled By O.E.M./IMB-A180, BIOS P1.00 10/09/2013
[0.682921] Call Trace:
[0.682974] dump_stack+0x5c/0x85
[0.683015] ? rest_init+0x50/0xd0
[0.683057] panic+0xe4/0x253
[0.683101] ? do_execveat_common.isra.39+0x87/0x830
[0.683142] ? rest_init+0xd0/0xd0
[0.683185] kernel_init+0xeb/0x100
[0.683228] ret_from_fork+0x22/0x40
[0.683305] Kernel Offset: 0xa000000 from 0xffffffff81000000 (relocation range: 0xffffffff80000000-0xffffffffbfffffff)
[0.683354] ---[end Kernel panic - not syncing: No working init found. Try passing init= option to kernel. See Linux Documentation/admin-guide/init.rst for guidance.

Security [Errors, Bugs, and Failures]

- A security error is made by a human
- As a consequence, a security bug is introduce in a program
  - A security bug is also called a “**vulnerability**”
- When a bug is triggered (or “exploited) it generates a security failure
- As a consequence, the security policy of a system is violated and the system is compromised


[mobile]
No exact OS matches for host
ALERT
ERROR : EMERGENCY SYSTEM FAILURE
Attempting to exploit SSHv1 CRC32 ... successful.
Reseting root password to "Z10N0101".
System open: Access Level <9>
ssh 10.2.2.2 -l root
root@10.2.2.2's password:
RRF-CONTROL> disable grid nodes 21 - 48
Warning: Disabling nodes 21-48 will disconnect sector 11 (27 nodes)
ARE YOU SURE? (y/n) y
Grid Node 21 offline...
Grid Node 22 offline...
Grid Node 23 offline...

Other Security Problems

There is an overall concept of “system security” in terms of

- Privacy / Confidentiality
- Integrity / Consistency
- Availability

Some applications might work as designed but contain vulnerabilities ...

- … when installed in systems with a conflicting security policy
  - “We allow students to have PHP applications in their web home directories”
- … when configured insecurely
  - The service is protected by a 16 character password (set to AAAAAAAAAAAAAAAA)

The “solution” to the Security problem

- Strong authentication on both services and users
- Reliable authorization / access control
- Effective abuse control
- Secure design of protocols, operating systems, and applications
- Bug-free implementation of protocols, operating systems, and applications
- Perfect security policy
- Perfect policy enforcement
- … and perfect users!

… and the real world

- Effective security protections are not deployed
- Administrators do not keep up with vendor updates/patches
- Sites do not monitor or restrict access to their internal hosts
- Organizations do not devote enough staff/resources to maintain security
- Users are not educated about security risks
- Sites do not implement policies (if they have one)

So what's possible?

Absolute security does not exist

- It is always a **tradeoff** between flexibility and ease of use of the system balanced against the risk of a successful attack
- We can get really high security, with a significant impact on utility

The goal of security design is not to avoid attacks. Rather to:

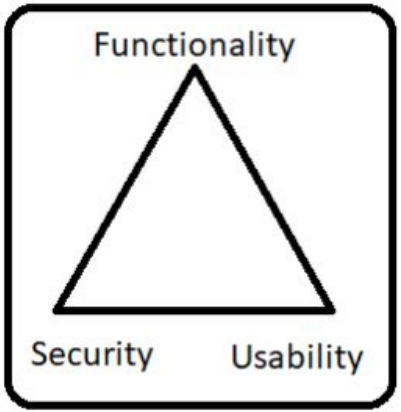
- Reduce the probability that an attack succeeds
- Reduce the damage produced by a successful attack

Security is expensive

- Procurement, deployment and management
- Performance impact
- Reduction in system utility

Absence of security is also expensive

- Attacks may produce a significant damage



Brief History of Hacking

Cap’n Crunch

- In 1972 John Draper finds that the whistle that comes with the Cap’n Crunch cereal produces a sound at the 2600 Hz frequency
- The 2600 frequency was used by AT&T to authorize long-distance calls

Phone Phreaking

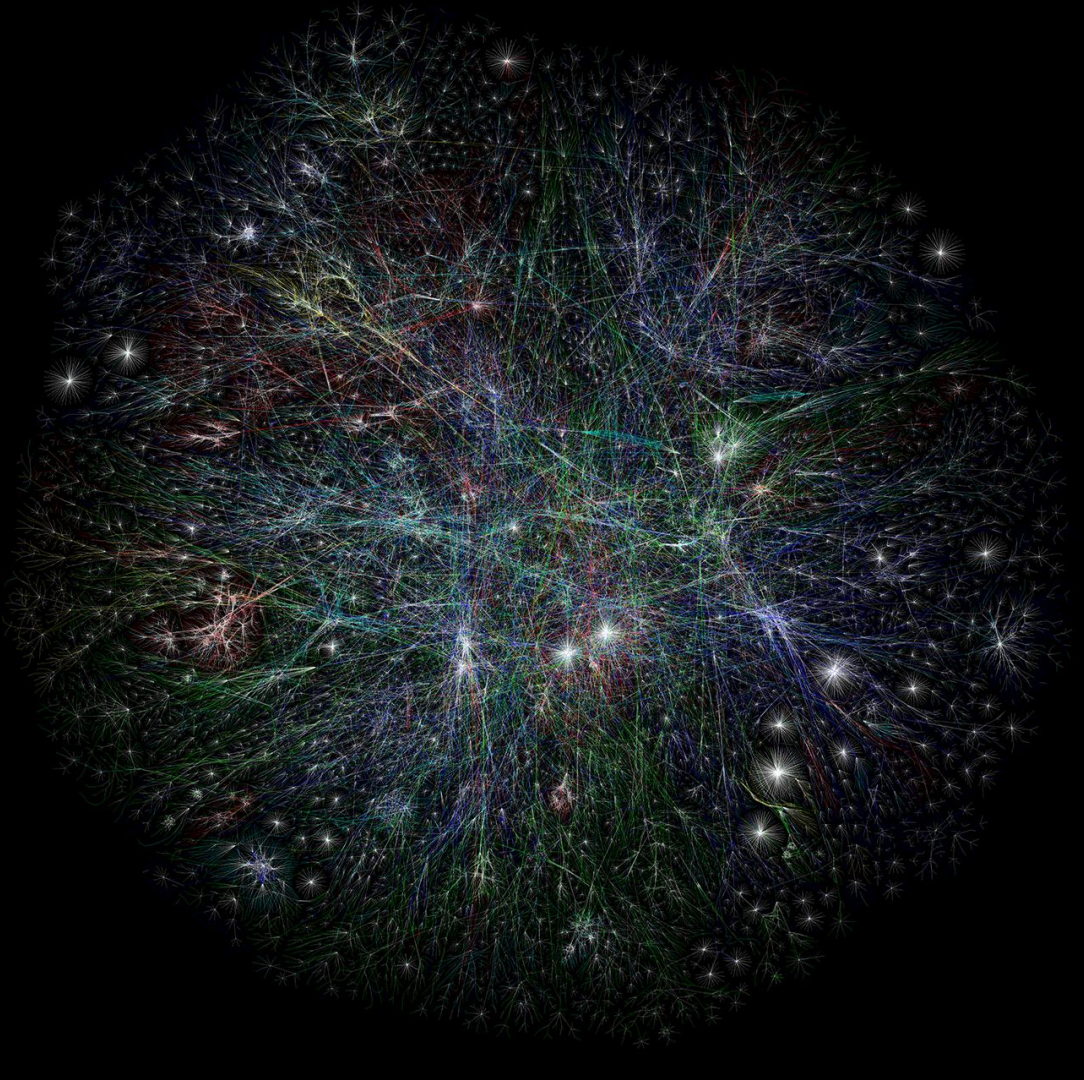
- John Draper became Captain Crunch and built a blue box
- The blue box produced a number of different tones that could be used for in-band signaling
- Draper was eventually sentenced for five years' probation for toll fraud
- His story became an integral part of hacker culture

The internet

- A network of networks
- Composed of a set of autonomous subnetworks
- Open architecture
- Different Administrative domains with different (and possibly conflicting) goals
- Governments, companies, universities, organizations rely on the Internet to perform mission-critical tasks

History (90’s)

- Fast growth (size and traffic volume)
- 1991: Tim Berners-Lee (CERN) creates the World-Wide Web
- 1993: The Mosaic browsers introduces the general public to the web
- The CGI specification (1993) supports web-based access to existing applications and services
- The Internet explodes



History (00-10’s)

- The web becomes part of our everyday life
- JavaScript and asynchronous communication create a new application paradigm
- Web-based services and applications become the way in which we access, process, and store information
- Smartphones become the most used platform to access the web
- Everything becomes networked (more or less): Internet of Things (IoT) ...

The Internet Worm

- November 2, 1988: The “Internet worm”, developed by Robert T. Morris, was injected in the internet
- A mistake in the replication procedure led to unexpected proliferation
- The internet had to be “turned off”
- Damages were estimated in the order of several hundred thousand dollars
- RTM was sentenced to three years’ probation, a 10k$ fine and 400 hours of community service

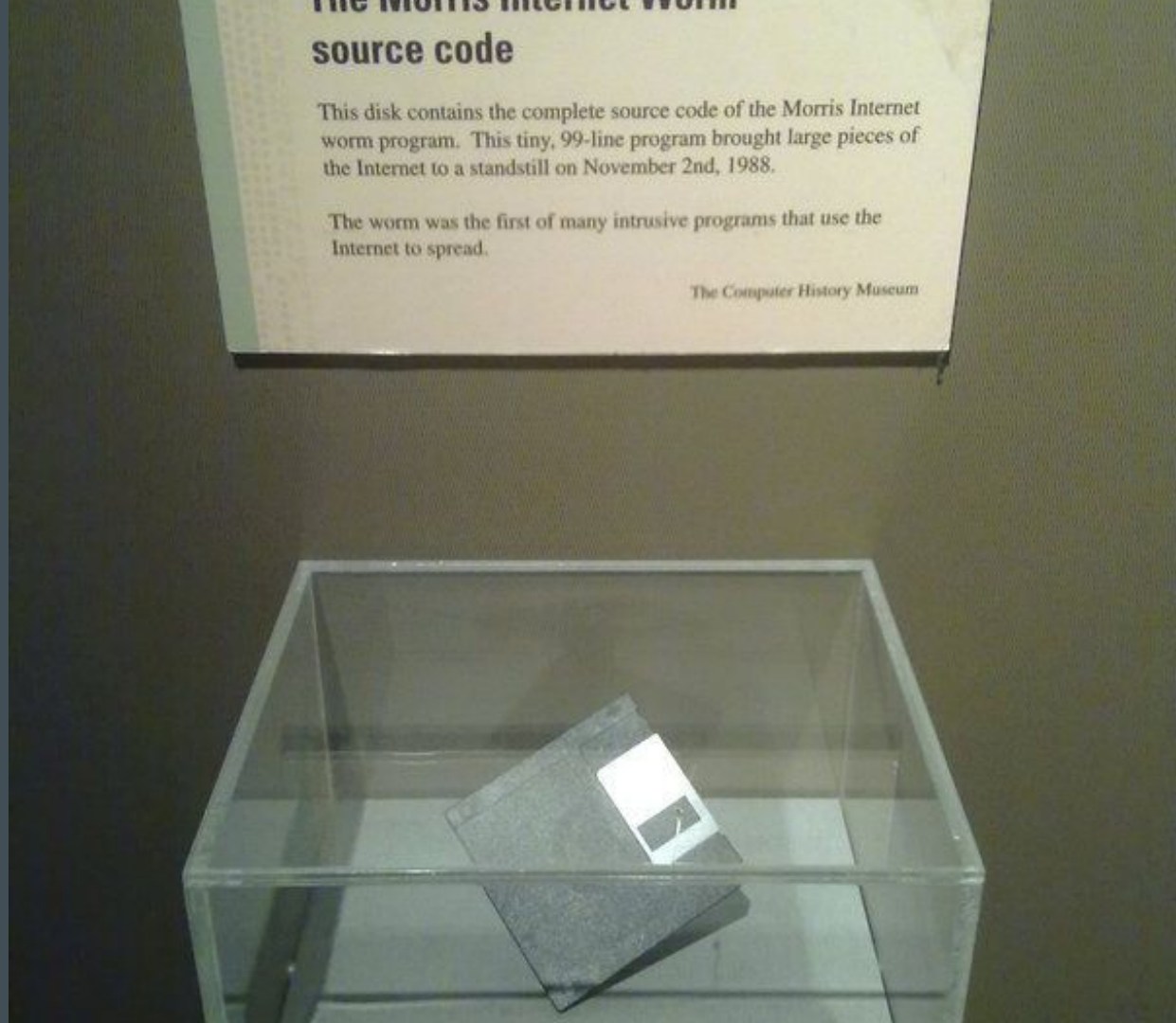
The Worm

A worm is a self-replicating program that spreads across a network of computers

The Morris Internet worm worked only on BSD UNIX

The worm consisted of two parts:

- A main program
- A bootstrap program



Bootstrap program: gain remote privileged access

Finger buffer overflow:

```
char line[512];
line[0] = '\0';
gets(line)
```

Sendmail: the DEBUG option allowed one to specify a number of commands to execute

- The bootstrap program (99 lines of C code) was transferred using a connection from the infecting machine

Main program

- Gathered information about the host's network interfaces and host with open connections to infect hosts
- Tried to break into hosts by using rsh, finger, sendmail
- Gathered more information on trusted hosts by examining
  - /etc/hosts.equiv
  - /.rhosts
  - ~/.forward in users home dirs
- Tried to rsh to the referenced hosts (password-cracking attack using the information contained in the password file, an internal dictionary of 432 words, and, eventually, the local UNIX dictionary)
- For each successful break-in the work was transferred

https://pdos.csail.mit.edu/6.828/2018

RTM is now professor of Operating Systems at MIT

Kevin Mitnick

- One of the most well-known hackers in the community
- 1982-1994: Sentenced many times for performing illegal activities
- 1994: California Department of Motor Vehicles issues $1-million warrant for Mitnick's arrest

WANTED
BY U.S. MARSHALS

NOTICE TO ARRESTING AGENCY: Before arrest, validate warrant through National Crime Information Center (NCIC).
United States Marshals Service NCIC entry number: (NIC/ W721440021).

NAME: .....MITNICK, KEVIN DAVID
AKS (S): .....MITNIK, KEVIN DAVID
MERRILL, BRIAN ALLEN

DESCRIPTION:
Sex: .....MALE
Race: .....WHITE
Place of Birth: .....VAN NUYS, CALIFORNIA
Date(s) of Birth: 08/06/63; 10/18/70
Height: .....5'11"
Weight: 190
Eyes: .....BLUE
Hair: .....BROWN
Skintone: .....LIGHT
Scars, Marks, Tattoos: .....NONE KNOWN

placeholder

For the plethora of high-profile hacking incidents

Hacking

What is a Hacker, anyway?

- First used at MIT in the 60s to describe “computer wizards”
- It has been eventually used to denote malicious hackers, that is, people that perform intrusions and misuse computer systems

Someone who lives and breathes computers, who knows all about computers, who can get a computer do anything. Equally important is the hacker’s attitude. Computer programming must be a hobby, something done for fun, not out of a sense of duty or for the money.

(Brian Harvey, University of Berkley)

Ethics

- Is malicious hacking legal? **NO**
- Is it legal to discuss vulnerabilities and how they are actually exploited? **YES**, and it is a good thing, provided that…
  - The goal is to educate and increase awareness
  - The goal is to teach how to build a more secure computing environment
- A full disclosure policy has been advocated by many respected researchers, provided that:
  - The information disclosed has been already distributed to the parties that may provide a solution to the problem (e.g., vendors)
    - See: responsible vulnerability disclosure process (IETF Internet Draft)
  - The ultimate goal is to prevent similar mistakes from being repeated

Legal Hacking: Penetration Testing

Vulnerability analysis followed by exploitation

Assumptions and hypothesis derived from a black-box analysis

Pentesting is part of the larger security auditing/analysis process

Not a good way to ensure the security of a system

A comprehensive security analysis process takes into account many other aspects (e.g., source code analysis, policy analysis, social engineering)

Security Vulnerabilities

Social Engineering

The Human Factor

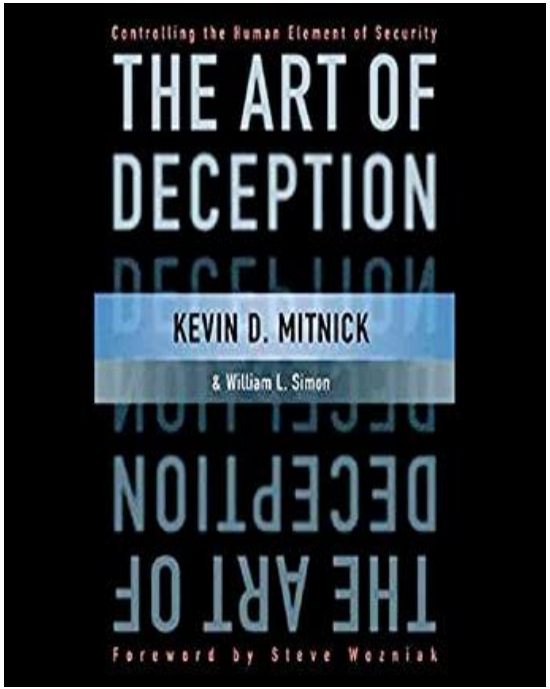
“To gain some advantage through human manipulation”

Typically it’s to obtain confidential information

- Passwords
- Financial data
- Confidential company data

Other instances

- Steal money
- Install malware



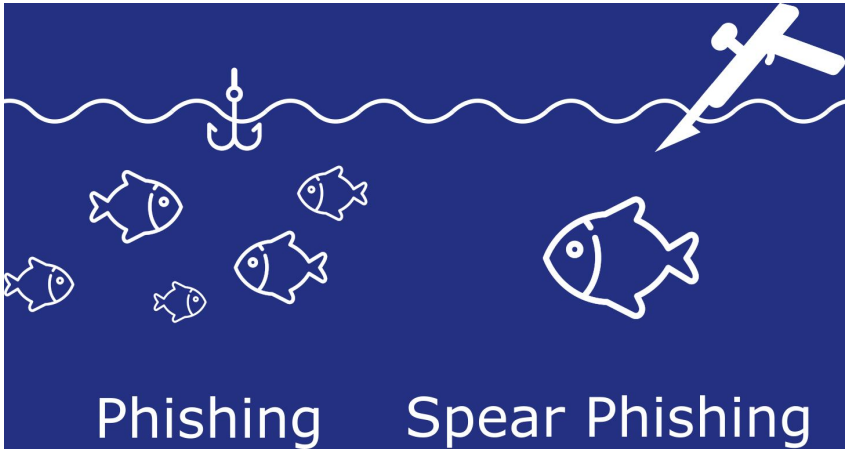
Common Examples

Phishing: mass attacks to steal some information.

Spear Phishing: email is used to carry out targeted attacks.

Baiting: promising victims a reward.

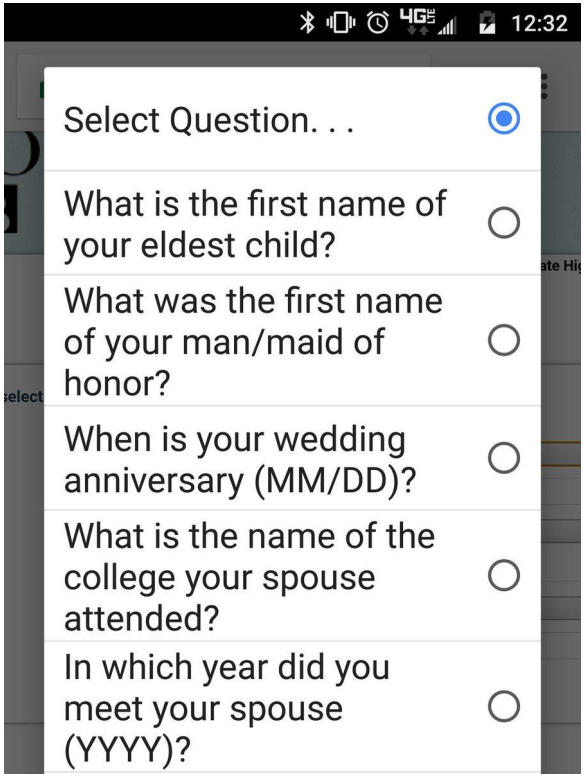
Tailgating: relies on human trust to give the criminal physical access to a secure building or area.



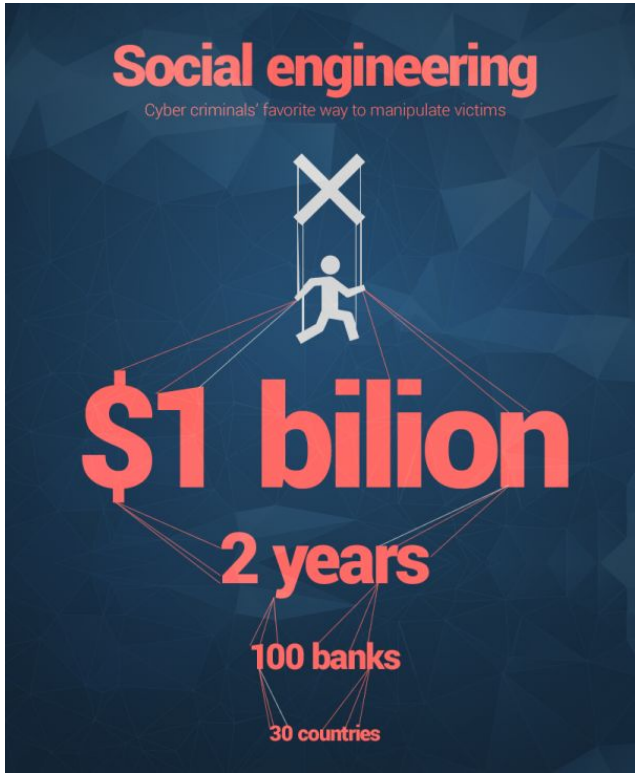
The Security Questions

Believe it or not, it is not difficult to guess your “secret” questions from an online account

- What’s your first pet
- Where were you born
- What’s your high school mascot
- What is your mother’s maiden name
- Add questions it’s better, but not foolproof

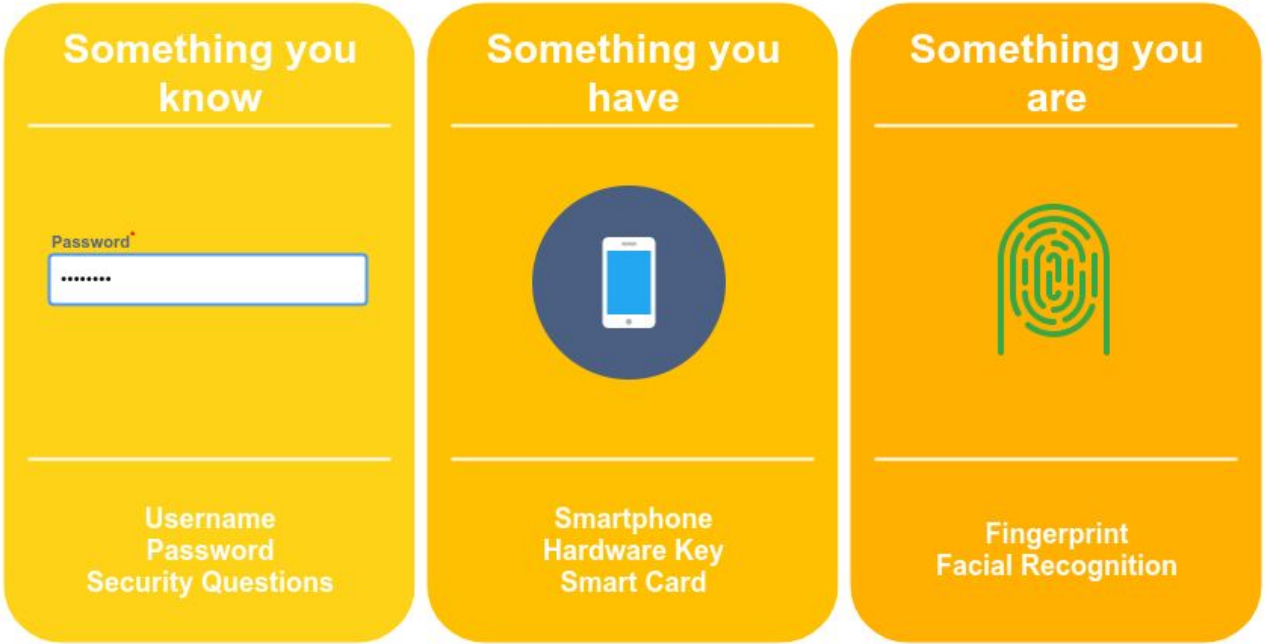


Consequences



Authentication Based Attacks

Factors of Identification



Threats to “something you know”

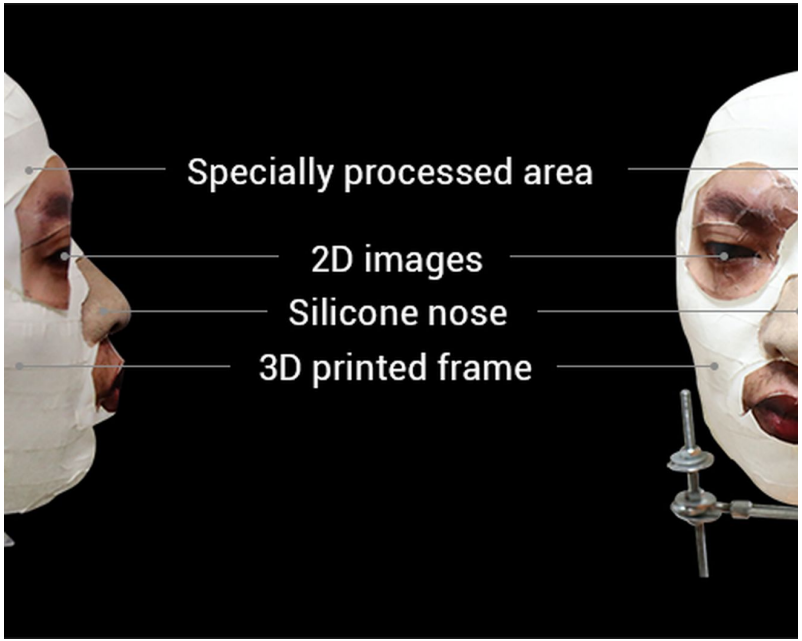
- Password authentication
  - Phishing
  - Poor password management
  - Key logging
  - Other eavesdropping
- Password based attacks
  - Password cracking

Threats to “something you have”

- Very few
- Usually protected with a chip
  - However, RFID copying
- Magnetic copying

Threats to “something you are”

- Some say the industry just isn’t there yet
- Many “facial recognition” systems are fooled with a print out of your face
- False positives and false negatives



Crypto (in-)securities

- We can try to attack the mathematical foundation of a cryptosystem
- If that doesn't work, we can try to attack the implementation

Side Channel Attacks

- We only want to sell even number of eggs
- We want to use RSA to protect the orders

(very sensitive information)

A parity problem

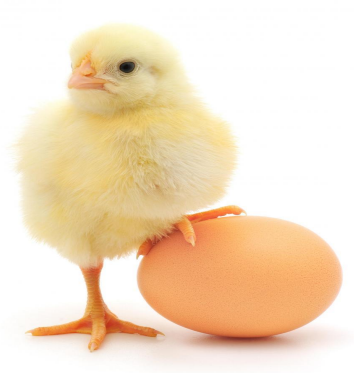
```
def check(c):
    m = decrypt(c)
    if is_even(m):
        return "ok"
    else:
        return "err"
```

n = 15 (p = 3, q = 5)

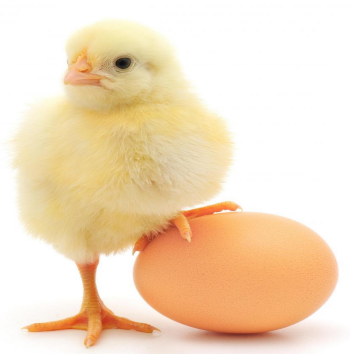
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

enc(m)

ok



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

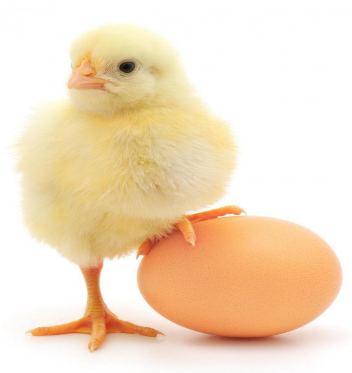


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

m even

enc(2·m)

ok

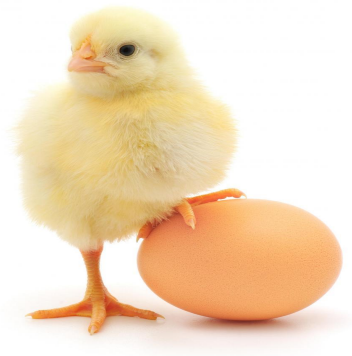


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Adaptive Ciphertext Attack

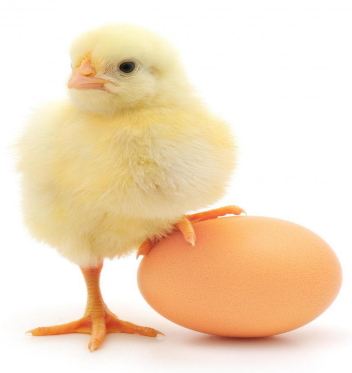
enc(**2**·m)

ok



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

$2m$ $2m - n$

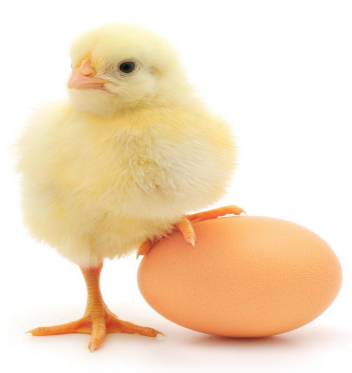


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

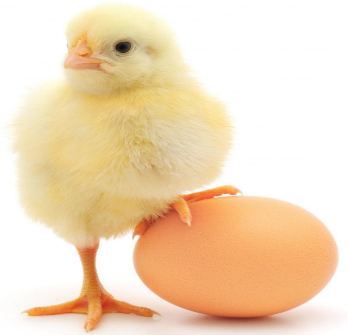
$$m \in \{0, 2, 4, 6\}$$

enc(**4**·m)

err



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

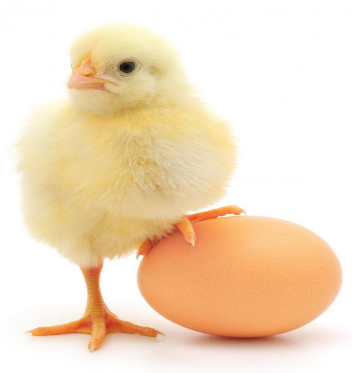


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

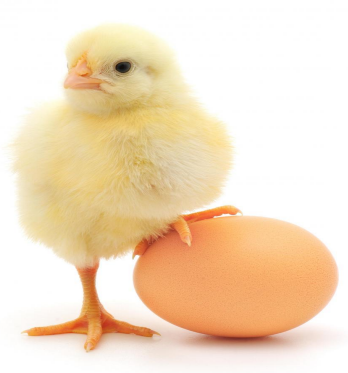
$$m \in \{4, 6\}$$

enc(8·m)

ok

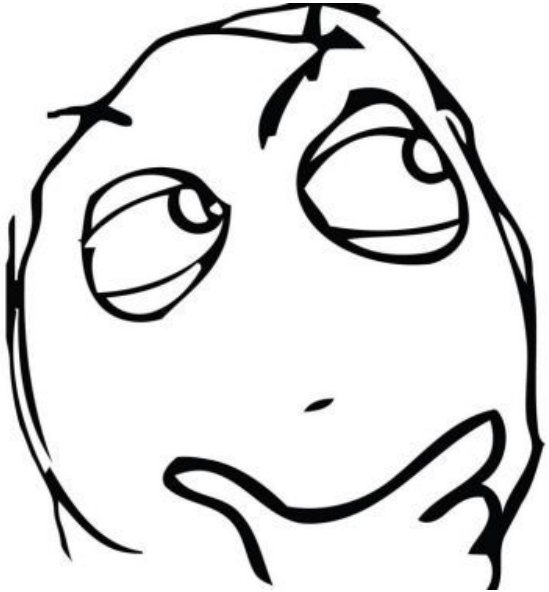


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

$$m = 4$$



How can we change the message?

$$enc(m) \rightarrow enc(2m)$$

$$(2^e \ mod_n) \cdot (m^e \ mod_n) = (2m)^e \ mod_n$$

$$enc(2m) = enc(2) \cdot enc(m)$$

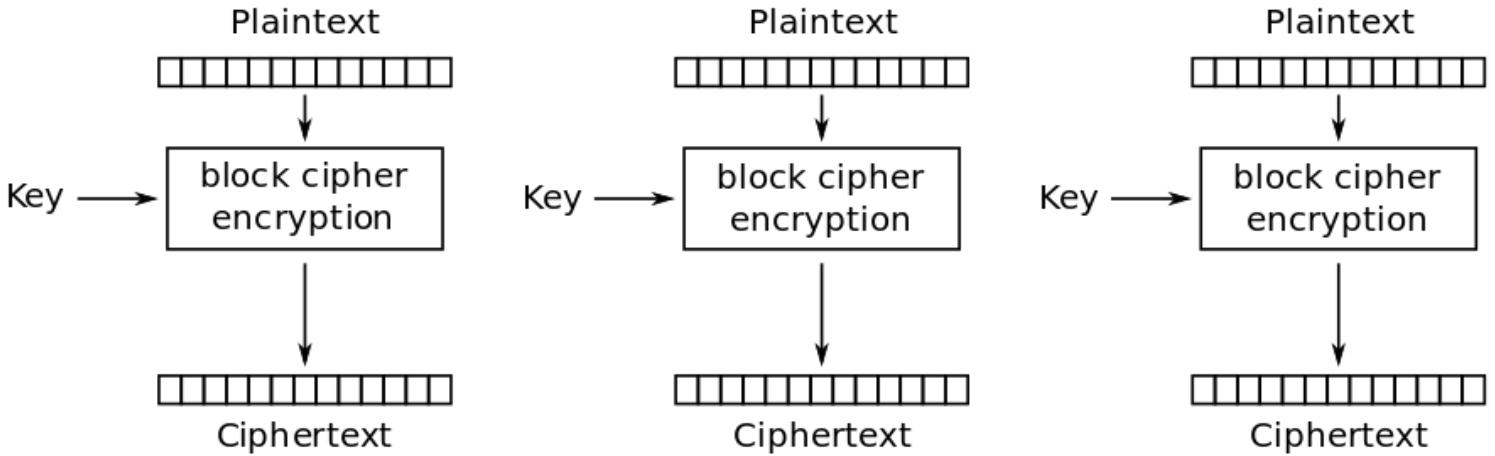
Multiplicative Property of RSA

Can we only hack farms?

0002	RANDOM PAD	00	MESSAGE

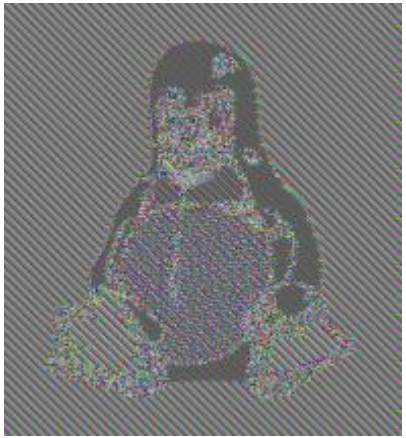
Broken by Bleichenbacher Attack (1998)

PKCS#1 v1.5

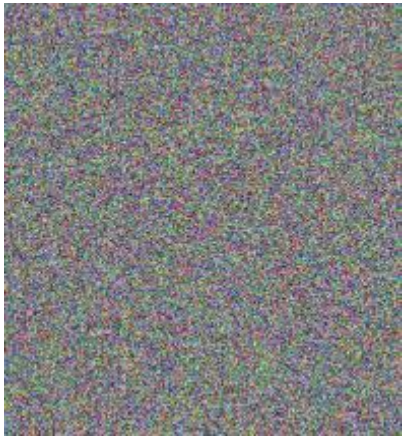


Electronic Codebook (ECB) mode encryption

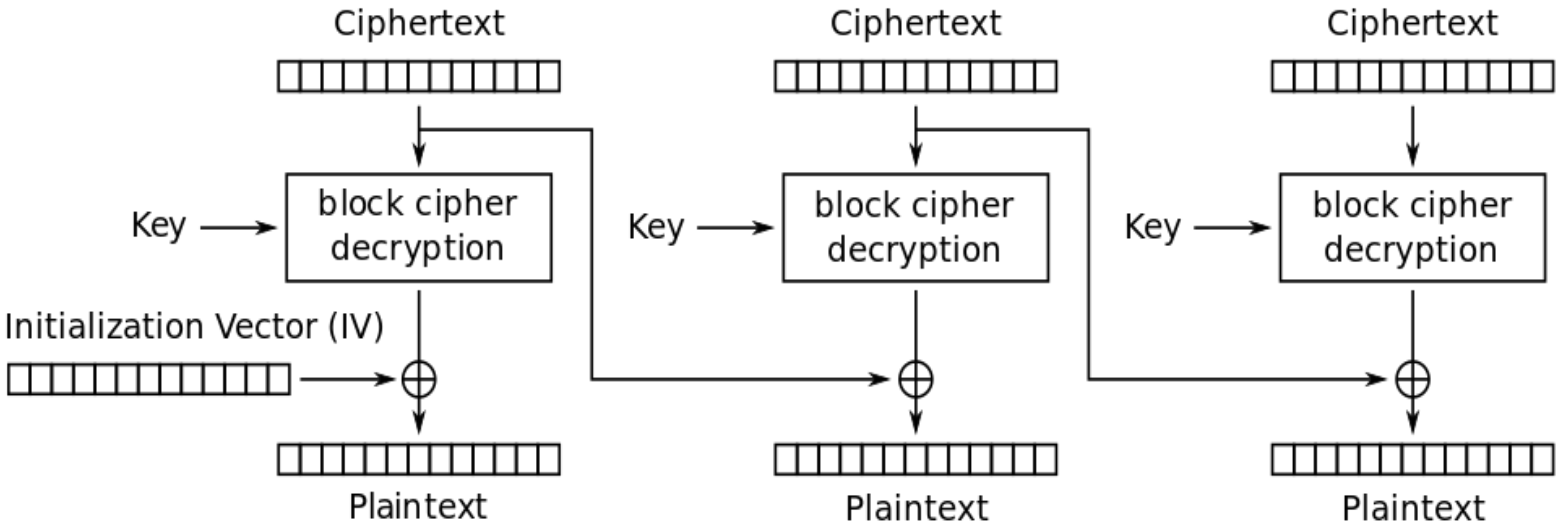
Electronic Codebook



ECB



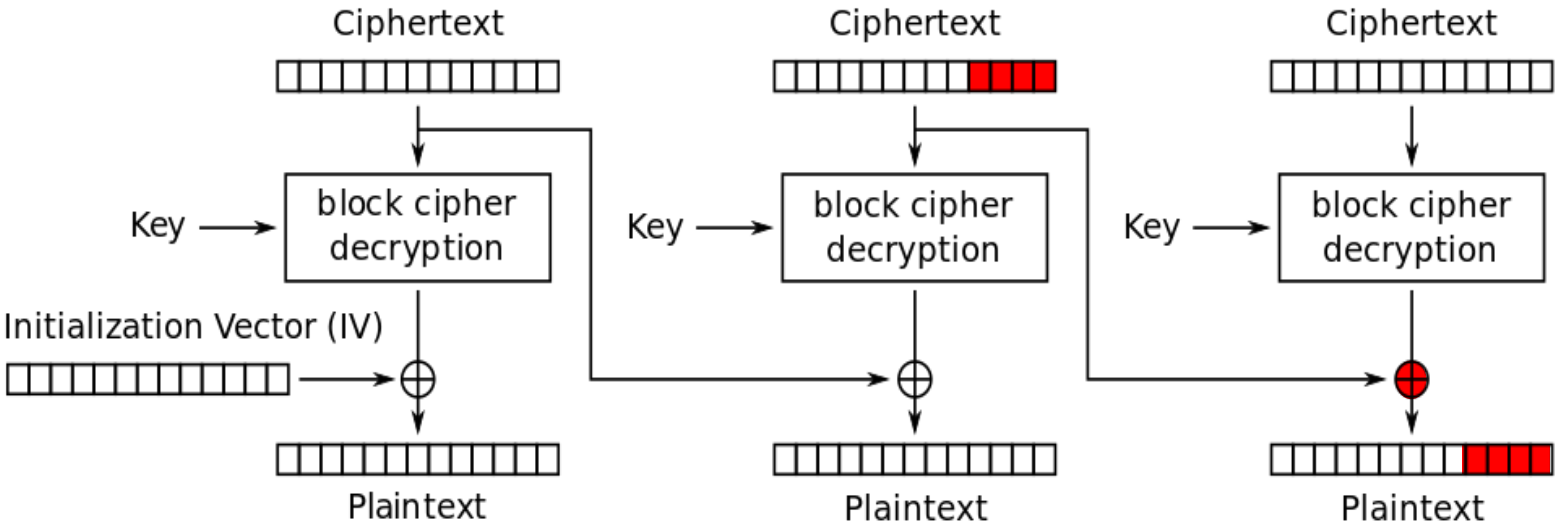
CBC



Cipher Block Chaining (CBC) mode decryption

Cipher Block Chaining

```
def cbc_mac(c):
    m = decrypt(c)
    if !pad_ok(m):
        return "pad error"
    if !mac_ok(m):
        return "mac error"
    ...
```

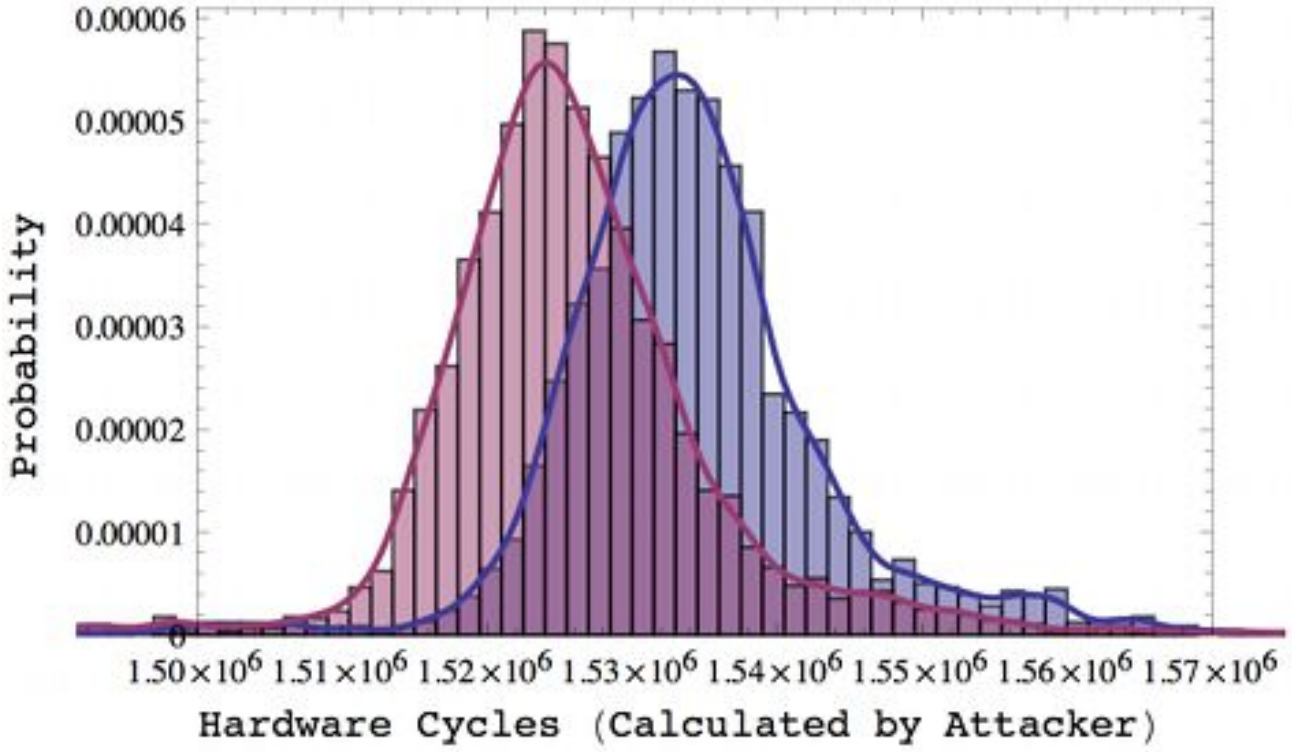


Cipher Block Chaining (CBC) mode decryption

https://www.infobytesec.com/down/paddingoracle_openjam.pdf

Padding Oracle Attack

```
def cbc_mac(c):
    m = decrypt(c)
    if !pad_ok(m) or !mac_ok(m):
        return "error"
    ...
```



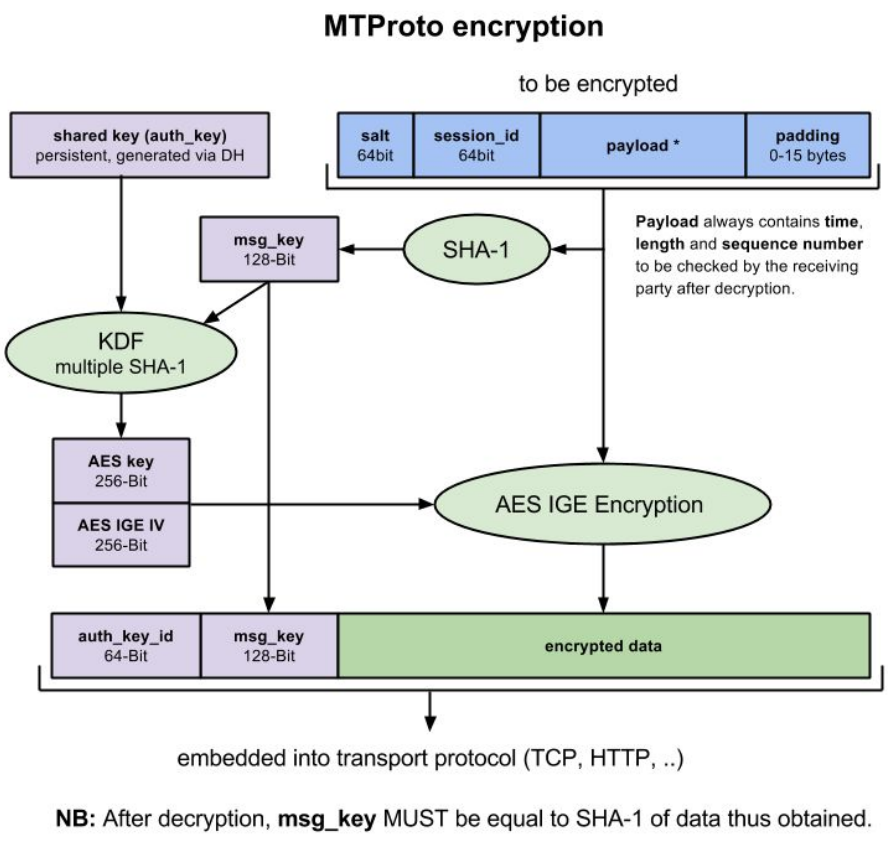
Timing Attack

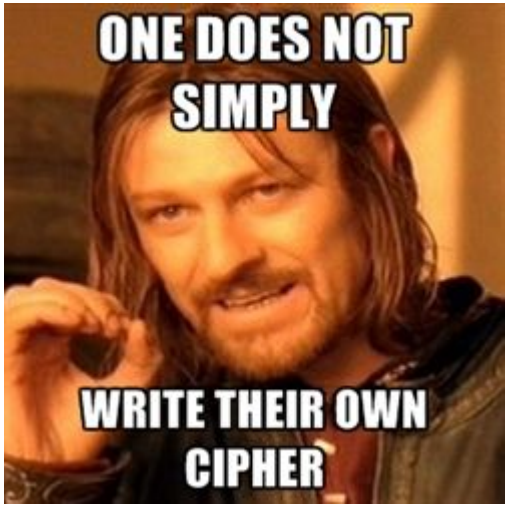
```
def cbc_mac(c):
    m = decrypt(c)
    if or(!pad_ok(m), !mac_ok(m)):
        return "error"
    ...
```

"Never ever implement your own cryptosystem"

(Dan Boneh)


Master


MTProto encryption
to be encrypted
shared key (auth_key)
persistent, generated via DH
salt
64bit
session_id
64bit
payload *
padding
0-15 bytes
msg_key
128-Bit
SHA-1
Payload always contains time, length and sequence number to be checked by the receiving party after decryption.
KDF
multiple SHA-1
AES key
256-Bit
AES IGE IV
256-Bit
AES IGE Encryption
auth_key_id
64-Bit
msg_key
128-Bit
encrypted data
embedded into transport protocol (TCP, HTTP, ..)
NB: After decryption, msg_key MUST be equal to SHA-1 of data thus obtained.


ONE DOES NOT SIMPLY
WRITE THEIR OWN CIPHER

MTProto

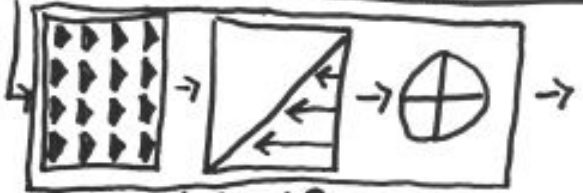
AES Crib Sheet
(Handy for memorizing)

Initial Round

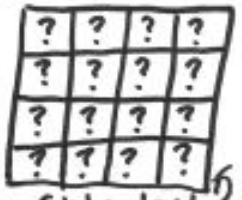
Intermediate Rounds

#	Key
9	128
11	192
13	256

X



Final Round



Ciphertext

General Math

11B = AES Polynomial = m(x)

$X^8 + X^4 + X^3 + X + 1$

Fast Multiply

$x \cdot a(x) = (a << 1) \oplus (a_7 == 1)\ ?\ 1B : 00$

$\log(x \cdot y) = \log(x) + \log(y)$

use $(x+1) = 03$ for log base

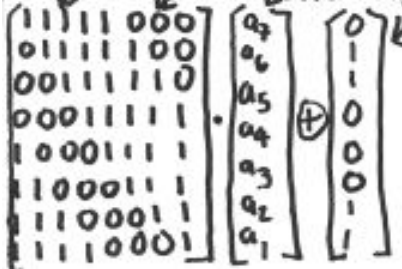
S-Box (SRD)

$SRD[a] = f(g(a))$

$g(a) = a^{-1} \bmod m(x)$

$f(a)$ Think $53 \oplus 63^T$

5 1's and 3 0's $[0110\ 0011]^T$



Key Expansion:

Round Constants: 01 02 04 08...

First Column:

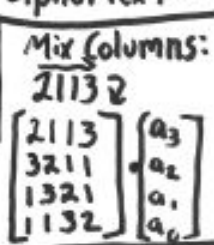
Round Key 0

Other Columns:

Prev Col ⊕ Col from Previous round key

Mix Columns:

2113 2



Inverse Mix

EBD9