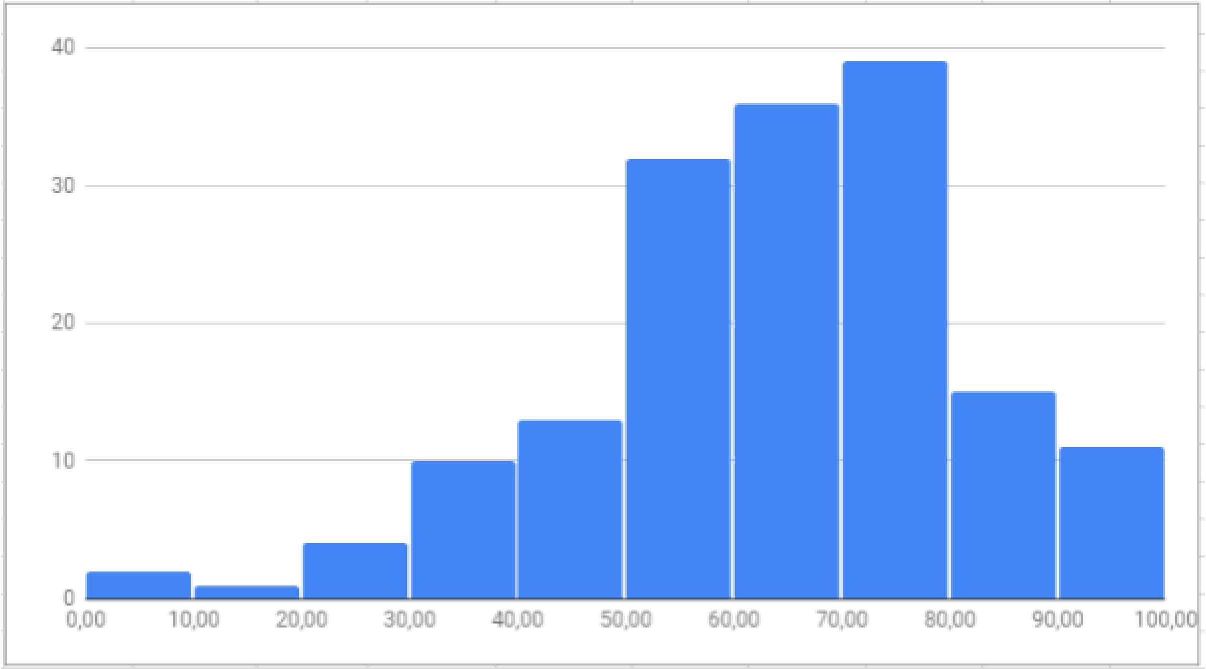
Feedback indicatif semestriel automne 2018
Indicative Student Feedback on Teaching Autumn 2018

14 November - 25 November

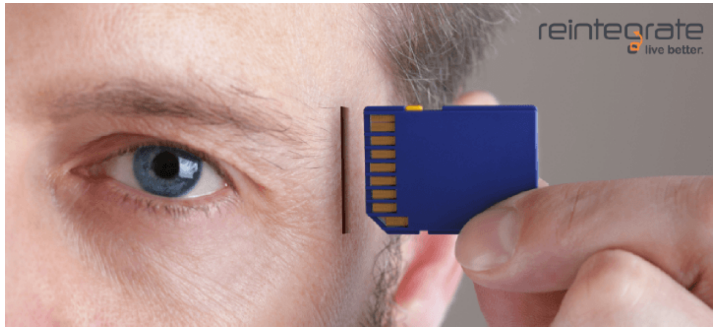
Please give us (more) feedback!



Spanish-ness reinterpretation

(Note that there was only one 100 and one 98 above 95)

- **Between 0-50: You really need to improve**. If you find concepts hard, come to the exercise sessions, come to office hours, and/or ask in the forum. → Really need to improve ☺
- **Between 50-65: You need to push a bit more.** Think about coming to office hours or asking in the course forum to fix the concepts that are not fully clear. → You are doing ok! But you need to fine tune
- **Between 65-80: You are keeping up,** but you need to keep working to maintain the level. → You are doing quite well! Do not stop working like you are
- **Between 80-100: You will probably do ok**, but do not lower your guard. → It would be weird that you fail



Learn, don’t memorize

More is not always better
(size does not matter)

Asymmetry adversary vs. defender

Why a MAC should not be constructed as Hash(k||m)

TAs & Carmela: Why are they designing their own crypto?!?!?

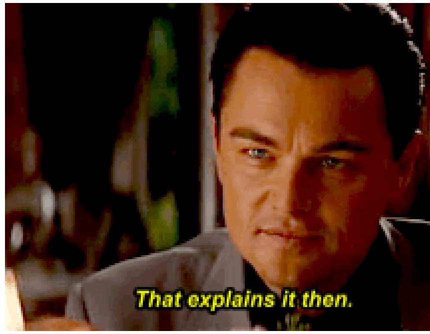
Why is this designing Crypto? You go from Hash to MAC.

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

Not exam material!!

Why a MAC should not be constructed as Hash(k||m)

TAs & Carmela: Why are they designing their own crypto?!?!?
Why is this designing Crypto? You go from Hash to MAC.



COM-208 Computer networks!!

Lecture 10 and "Computer Networking: A Top-Down Approach": MAC = Hash(k||I am Alice)

Not exam material!!

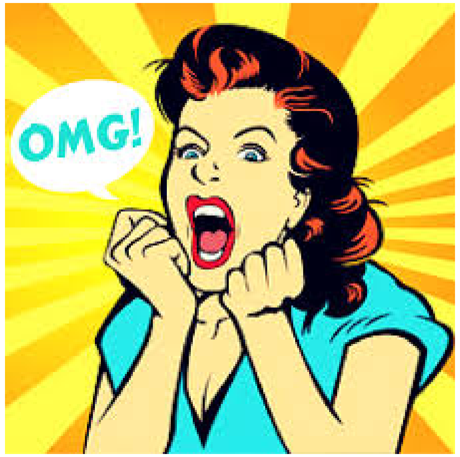
https://www.whitehatsec.com/blog/hash-length-extension-attacks/

Why a MAC should not be constructed as Hash(k||m)

TAs & Carmela: Why are they designing their own crypto?!?!?
Why is this designing Crypto? You go from Hash to MAC.

COM-208 Computer networks!!

Lecture 10 and "Computer Networking: A Top-Down Approach": MAC = Hash(k||I am Alice)



Is it wrong?!?!?!?

Not exam material!!

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

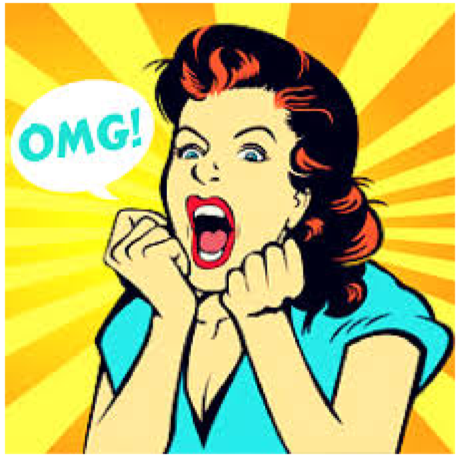
Why a MAC should not be constructed as Hash(k||m)

TAs & Carmela: Why are they designing their own crypto?!?!?

Why is this designing Crypto? You go from Hash to MAC.

COM-208 Computer networks!!

Lecture 10 and "Computer Networking: A Top-Down Approach": MAC = Hash(k||I am Alice)



Is it wrong?!?!?!?

Short answer: Not always

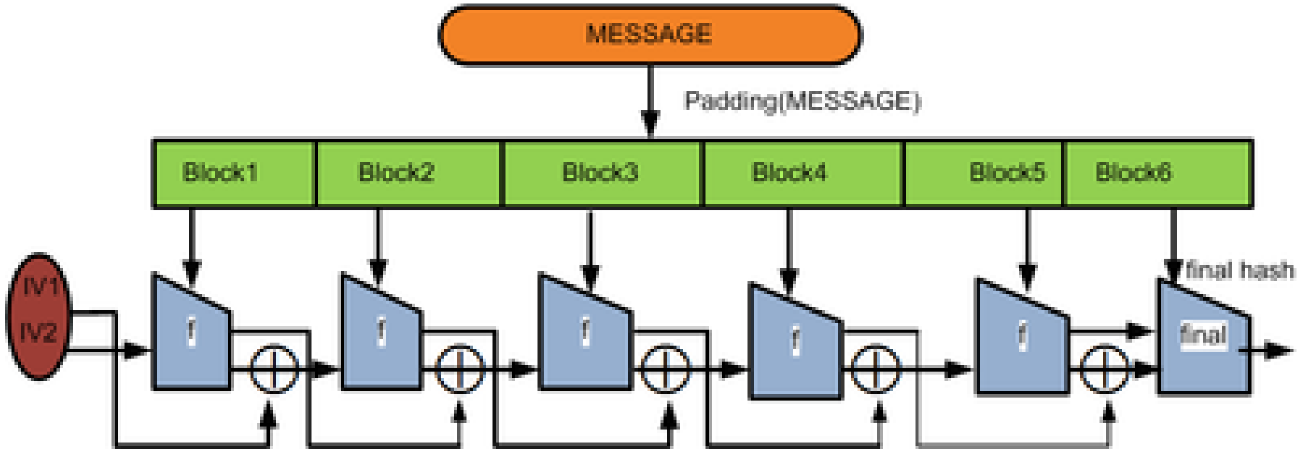
So when is it wrong?

Not exam material!!

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

Why a MAC should not be constructed as Hash(k||m)

Many Hash functions (MD5, SHA1, SHA256) are built using **the Merkle-Damgard paradigm**



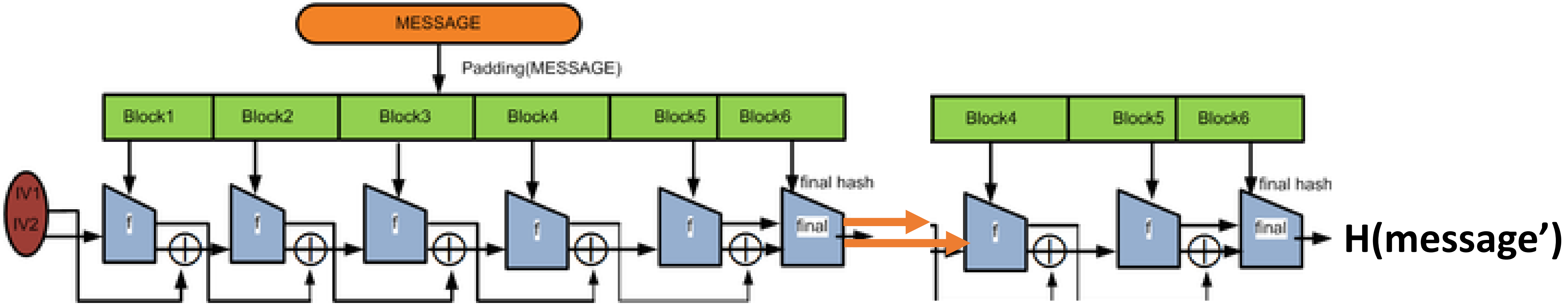
H(message)

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

Not exam material!!

Why a MAC should not be constructed as Hash(k||m)

Many Hash functions (MD5, SHA1, SHA256) are built using **the Merkle-Damgard paradigm**



Hash length extension attacks! (given H(m) obtain H(m||stuff)

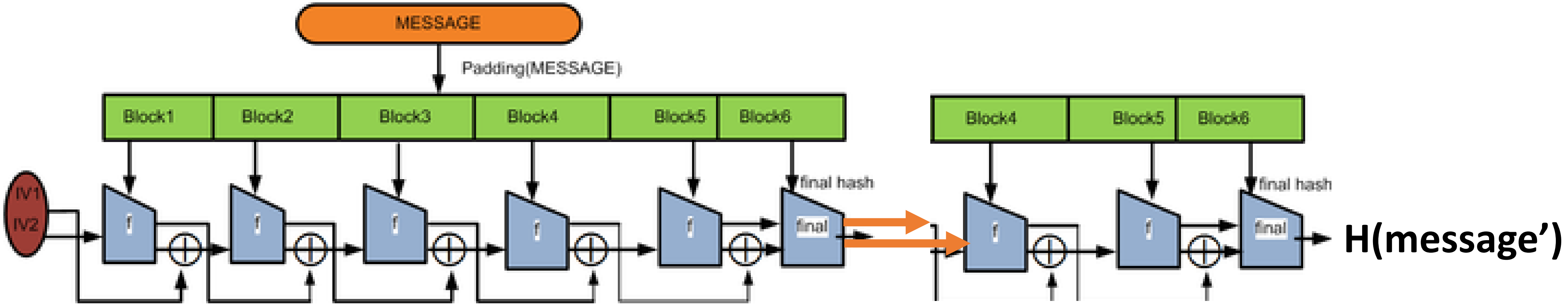
Example: H(k||this is Alice) → H(k||this is Alice, First of her name, Queen of the Andals and the First Men)

Not exam material!!

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

Why a MAC should not be constructed as Hash(k||m)

Many Hash functions (MD5, SHA1, SHA256) are built using **the Merkle-Damgard paradigm**



Hash length extension attacks! (given H(m) obtain H(m||stuff)

Example: H(k||this is Alice) → H(k||this is Alice, First of her name, Queen of the Andals and the First Men)

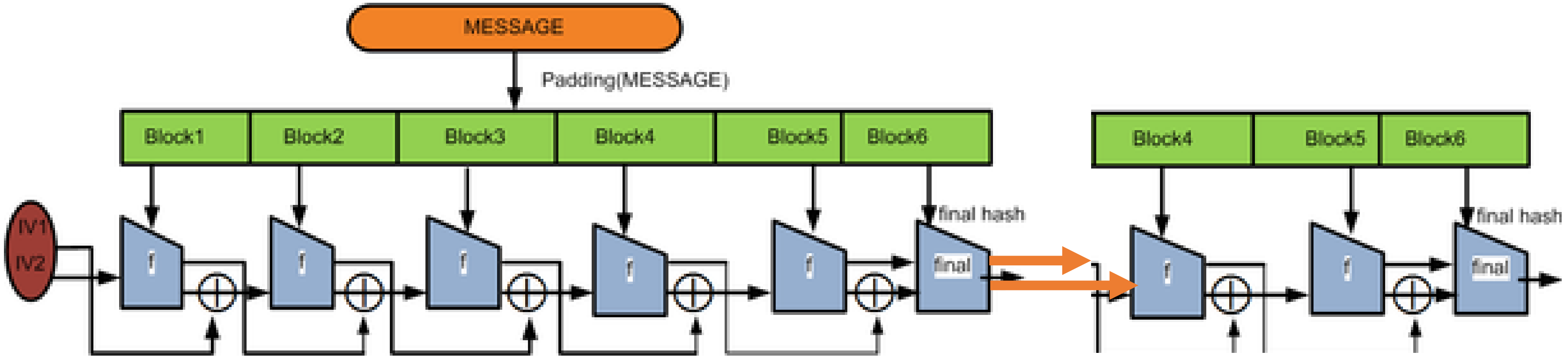
MAC=H(k||Alice) does not guarantee integrity

Not exam material!!

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

Why a MAC should not be constructed as Hash(k||m)

Many Hash functions (MD5, SHA1, SHA256) are built using **the Merkle-Damgard paradigm**



H(message')

Hash length extension attacks! (given H(m) obtain H(m||stuff)

Example: H(k||this is Alice) → H(k||this is Alice, First of her name, Queen of the Andals and the First Men)

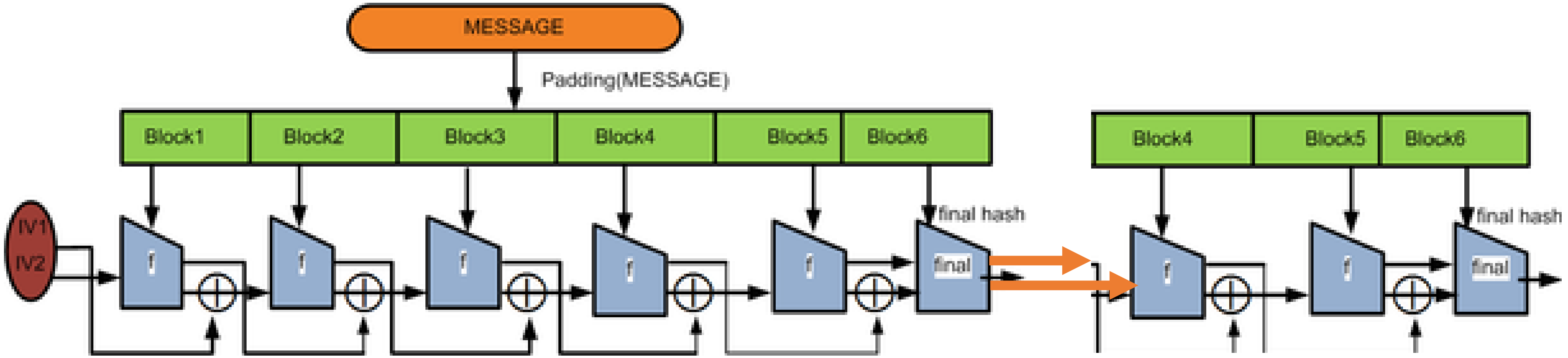
MAC=H(k||Alice) does not guarantee integrity
for Merkle-Damgard hash functions

Not exam material!!

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

Why a MAC should not be constructed as Hash(k||m)

Many Hash functions (MD5, SHA1, SHA256) are built using **the Merkle-Damgard paradigm**



Hash length extension attacks! (given H(m) obtain H(m||stuff)

Example: H(k||this is Alice) → H(k||this is Alice, First of her name, Queen of the Andals a

MAC=H(k||Alice) does not guarantee integri

for Merkle-Damgard hash functions

It is not wrong, but these are **many** hash functions. **Safe choice: don't do this.** CBC-MAC or HMAC!

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

An everyday declassification act

- Publishing code in GitHub

An everyday declassification act

- Publishing code in GitHub


@MisterCh0c Follow
I'm a hacker not a slacker ~ OSCP ~ #infosec ~ #Belgium
Feb 20, 2017 · 4 min read

Developers are unknowingly posting their credentials online. I chose to warn them instead of hacking them

Recently I started playing with GitHub dorks and asked myself how a black-hat hacker could exploit these in a large scale attack.

> *Dorking is a term that refers to the practice of applying advanced search techniques and specialized search engine parameters to discover confidential information from companies and individuals that wouldn't typically show up during a normal web search.*

It quickly became clear that it's harder to find stuff on GitHub because of programmers (:

Here are some example of what I found when I searched for all commits named "Remove password"

https://hackernoon.com/developers-are-unknowingly-posting-their-credentials-online-caa7626a6f84

An everyday declassification act

- Publishing code in GitHub

  Common practice: hardcode credentials on development

@MisterCh0c Follow
I'm a hacker not a slacker ~ OSCP ~ #infosec ~ #Belgium
Feb 20, 2017 · 4 min read

Developers are unknowingly posting their credentials online. I chose to warn them instead of hacking them

Recently I started playing with GitHub dorks and asked myself how a black-hat hacker could exploit these in a large scale attack.

> *Dorking is a term that refers to the practice of applying advanced search techniques and specialized search engine parameters to discover confidential information from companies and individuals that wouldn't typically show up during a normal web search.*

It quickly became clear that it's harder to find stuff on GitHub because of programmers (:

Here are some example of what I found when I searched for all commits named "Remove password"

https://hackernoon.com/developers-are-unknowingly-posting-their-credentials-online-caa7626a6f84

Ar

- Pu

gitleaks

build passing

Audit git repos for secrets

Gitleaks provides a way for you to find unencrypted secrets and other unwanted data types in git source code repositories.

As part of it's core functionality, it provides;

- Github support including support for bulk organisation and repository owner (user) repository scans, as well as pull request scanning for use in common CI workflows.
- Support for private repository scans, and repositories that require key based authentication
- Output in CSV and JSON formats for consumption in other reporting tools and frameworks
- Externalised configuration for environment specific customisation including regex rules
- Customisable repository name, file type, commit ID, branchname and regex whitelisting to reduce false positives
- High performance through the use of src-d's go-git framework

It has been sucessfully used in a number of different scenarios, including;

- Adhoc scans of local and remote repositories by filesystem path or clone URL
- Automated scans of github users and organisations (Both public and enterprise platforms)
- As part of a CICD workflow to identify secrets before they make it deeper into your codebase
- As part of a wider secrets auditing automation capability for git data in large environments

https://hackerr

Last two weeks: how adversaries think

THE SECURITY ENGINEERING PROCESS

1. **Define a security policy and a threat model**
2. **Define security mechanisms that support the policy within the threat model.**
3. **Build an implementation that supports / embodies the mechanisms.**

THE ATTACKER PROCESS

1. **Exploit poor definitions and/or unexpected capabilities**
2. **Exploit weaknesses in the design**
3. **Exploit implementation or operation errors**

Last two weeks: examples of common attacks

Insecure interactions between components

Usually enabled by lack of checks when processing input

'OS Command Injection'

```
$username = $_POST['user'];
$command = 'ls -l /home/' . $username;
system($command);
```

Problem if $username='; rm – rf'

'Cross-site scripting'

```
$username = $_GET['userName'];
echo '<div class="header"> Welcome, ' . $username . '</div>';
```

Problem if $username='<script>....</script>'

'Cross-site Request Forgery'

```
<form action='http://epflHR.ch/paystudent.php' id='form' method='post'>
<input type='hidden' name='firstname' value='Ugo'>
<input type='hidden' name='lastname' value='Damiano'>
<input type='hidden' name='amount' value = '1000 CHF'>
```

Problem if on the server side a user is logged in
(the server-side code will process the form with THAT user's privileges)

If not clear please write on the Forum or come to office hours

Last two weeks: examples of common attacks

Risky Resource Management

Enabled by lack of checks or careless programming

'Uncontrolled Format String'

```
#include<stdio.h>
int main(int argc, char** argv) {
    char buffer[100];
    strncpy(buffer, argv[1], 100);
    printf(buffer);
    return 0;
}
```

Problem if argv[1]='%d %d %d %d'
(the program will read from the stack)

Memory safety ('buffer overflow')

```
void vulnerable(char *buf) {
        free(buf);
        buf[12] = 42;
}
```

Accessing a freed memory region

Code Injection

Inject arbitrary code in the stack and execute it

Control Flow Hijack

Inject arbitrary return pointers to execute desired existing instructions

Last two weeks: examples of common attacks

Porous defenses

Authentication, Authorization, and Encryption design failures and bugs

[5]	CWE-306	Missing Authentication for Critical Function
[6]	CWE-862	Missing Authorization
[7]	CWE-798	Use of Hard-coded Credentials
[8]	CWE-311	Missing Encryption of Sensitive Data
[10]	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	CWE-250	Execution with Unnecessary Privileges
[15]	CWE-863	Incorrect Authorization
[17]	CWE-732	Incorrect Permission Assignment for Critical Resource
[19]	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[21]	CWE-307	Improper Restriction of Excessive Authentication Attempts
[25]	CWE-759	Use of a One-Way Hash without a Salt

Last two weeks: mitigations

Data execution prevention

Ensure that writable code is not executable (at page level)

\+ avoid injection and low overhead

\- low granularity and reduce functionality

Address Space Layout Randomization

Change program, variables, etc placement in memory

\+ avoids hijack

\- high performance impact, not universal (some parts cannot change), may leak information

Stack Canaries

Reserved locations in the stack to verify integrity of return

\+ avoids hijack

\- not universal, may leak information

Last two weeks: fuzzing

Automated software testing technique. The fuzzing engine generates inputs based on some criteria:

- Random mutation
- Leveraging input structure
- Leveraging program structure

If the program crashes, there is a bug!

Coverage to guide fuzzing:

- how much of the program was executed?
- Requires both control-flow and data-flow

New content!!

Computer Security (COM-301)
Trusted computing

Carmela Troncoso

SPRING Lab

carmela.troncoso@epfl.ch

Some slides/ideas adapted from: Wouter Lueks

Summary of the last two weeks

Securing systems and software is difficult!

Systems are complex: principles are hard to follow in practice

economy of mechanism – systems are large and interconnected

least common mechanism – efficiency requires reuse

complete mediation – wishful thinking

Sanitization is hard: so many layers and interactions

Even if cryptography is effective, building protocols is hard

What if we could rely on hardware?

TRUSTED HARDWARE

"A piece of hardware can be trusted if it always behaves in the expected manner for the intended purpose"

-Trusted Computing Group

I execute precisely the program that you need me to execute

Imagine you could build a small version of you that you could fully trust to work perfectly

And you could just put him/her inside a computer to execute your programs

What we will learn in this lecture

Trusted hardware

What properties it offers

isolation: it is not possible to “peek” inside

attestation: it can prove that it does what you think it is doing

sealing: it can store secrets in unprotected memory

Trusted hardware requires protection against side channel attacks

Example uses of Trusted Hardware

Examples of trusted hardware

TPM chip

Smart Card


Secure Enclave

Hardware security module

Intel SGX

Examples of trusted hardware

Integrated Trusted Hardware

TPM chip

Secure Enclave

Intel SGX

Smart Card

Trusted Hardware with physical isolation

Hardware security module

Trusted execution environments

Option 1: Dedicated devices

Strong **physical** protections

Protect against invasive adversaries

Limited functionality

Optimized for cryptographic operations

Can be used as a root of trust for larger operations

Booting of an operative system

Authentication procedures

Trusted execution environments

Option 2: Secure enclaves

Protected regions of memory created with help from a special processor

Provide confidentiality and integrity to processes, and can verify the executable code before execution

Can be used to securely run processes protecting them from:

- Operating System, or hypervisor
- BIOS, firmware, drivers
- Other Software “between BIOS and OS”
- By extension, any remote attack

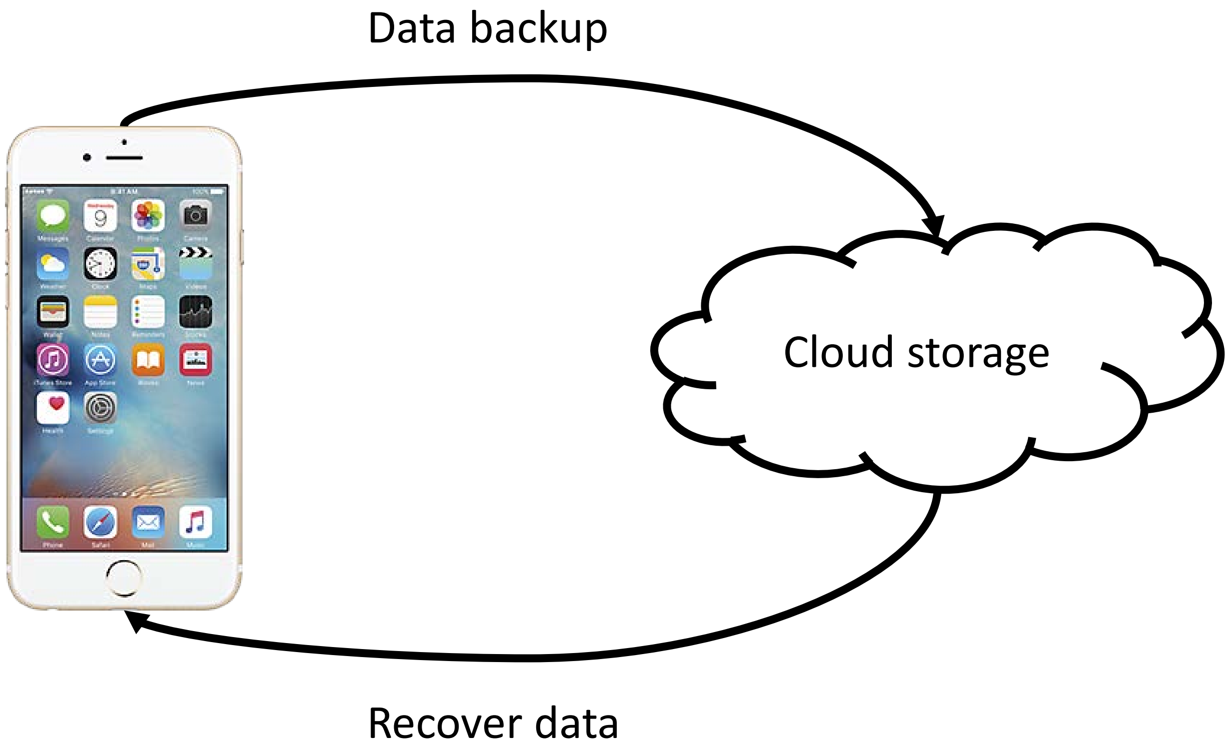
Trusted Hardware Key Properties

1. Isolation

Use case: secure backups of iPhones

Problem statement

We want users to be able to back up their data, and be **the only ones** that can access it



Threat Model

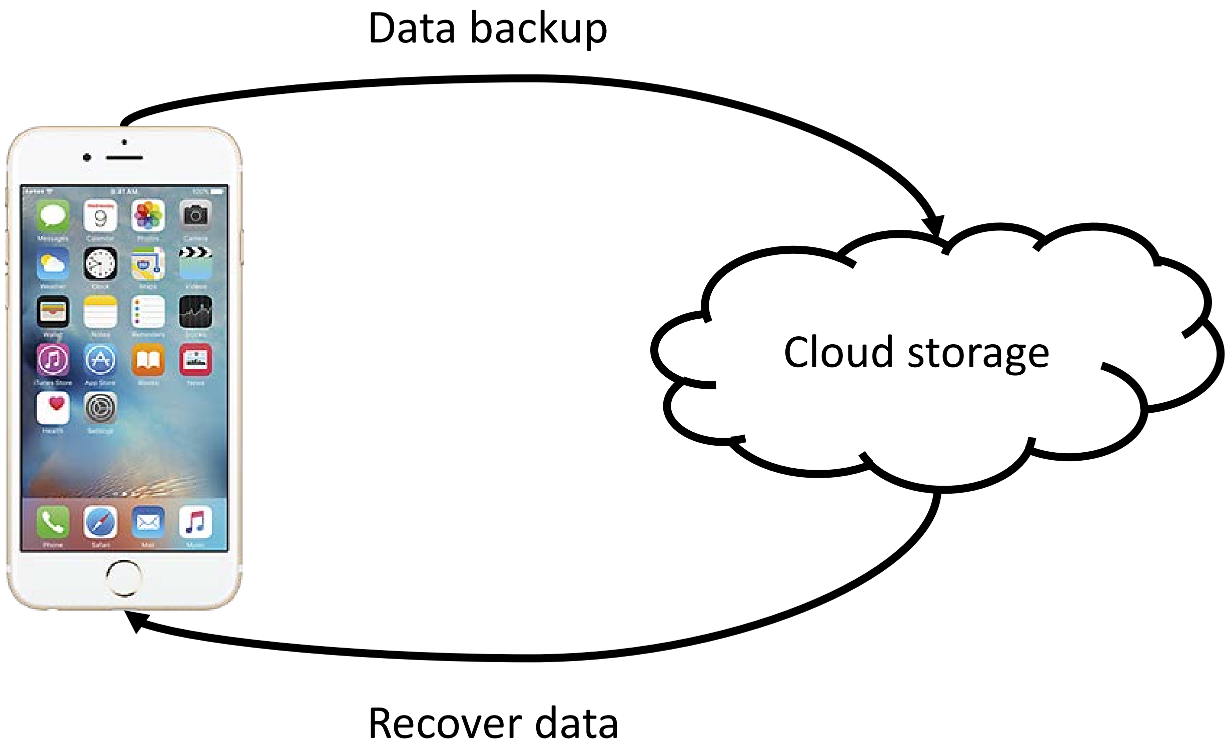
Who do you want to protect against/why?

https://blog.cryptographyengineering.com/2016/08/13/is-apples-cloud-key-vault-crypto/

Use case: secure backups of iPhones

Problem statement

We want users to be able to back up their data, and be **the only ones** that can access it



THREAT MODEL
Who do you want to protect against/why?

Other users
Cloud provider
Apple
External parties (possibly with subpoenas)

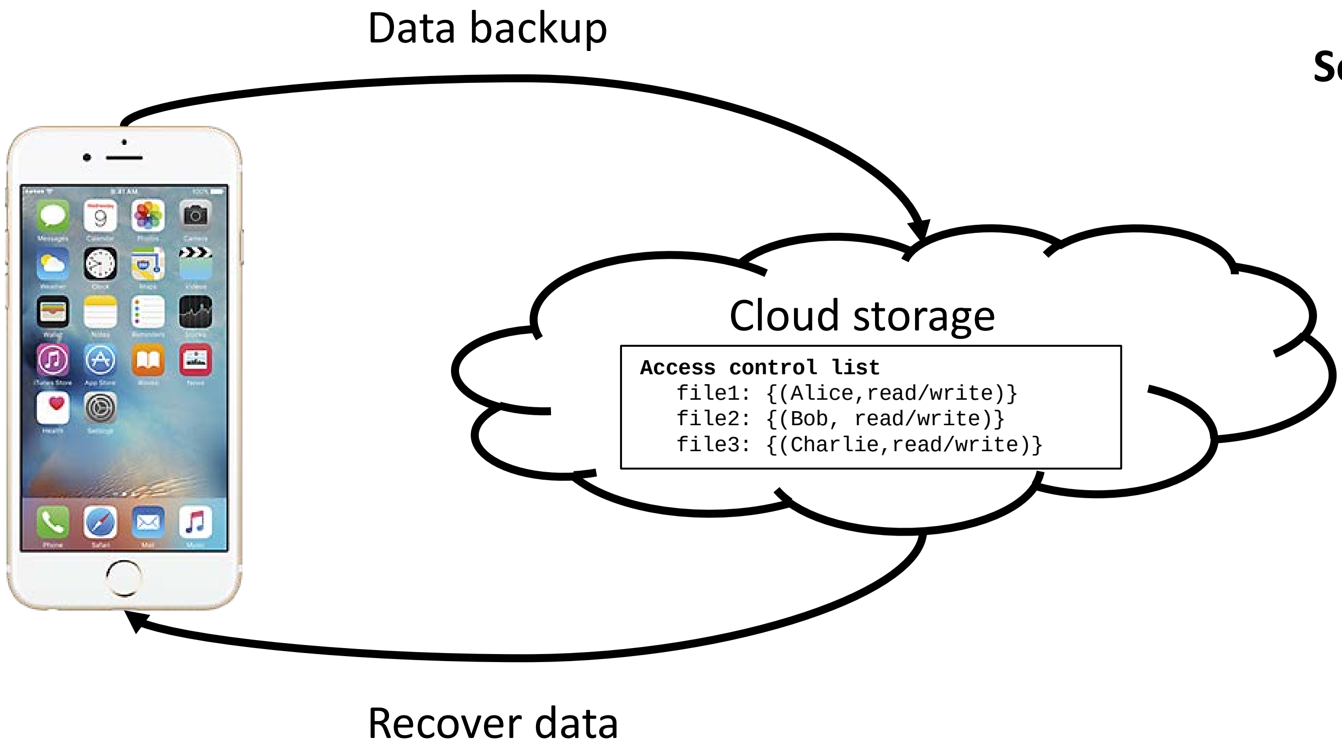
Adversaries may have a lot of computational power!!

Use case: secure backups of iPhones

Threat Model
- Other users
- Cloud provider
- Apple
- External parties (with subpoenas)

Problem statement

We want users to be able to back up their data, and be the only ones that can access it



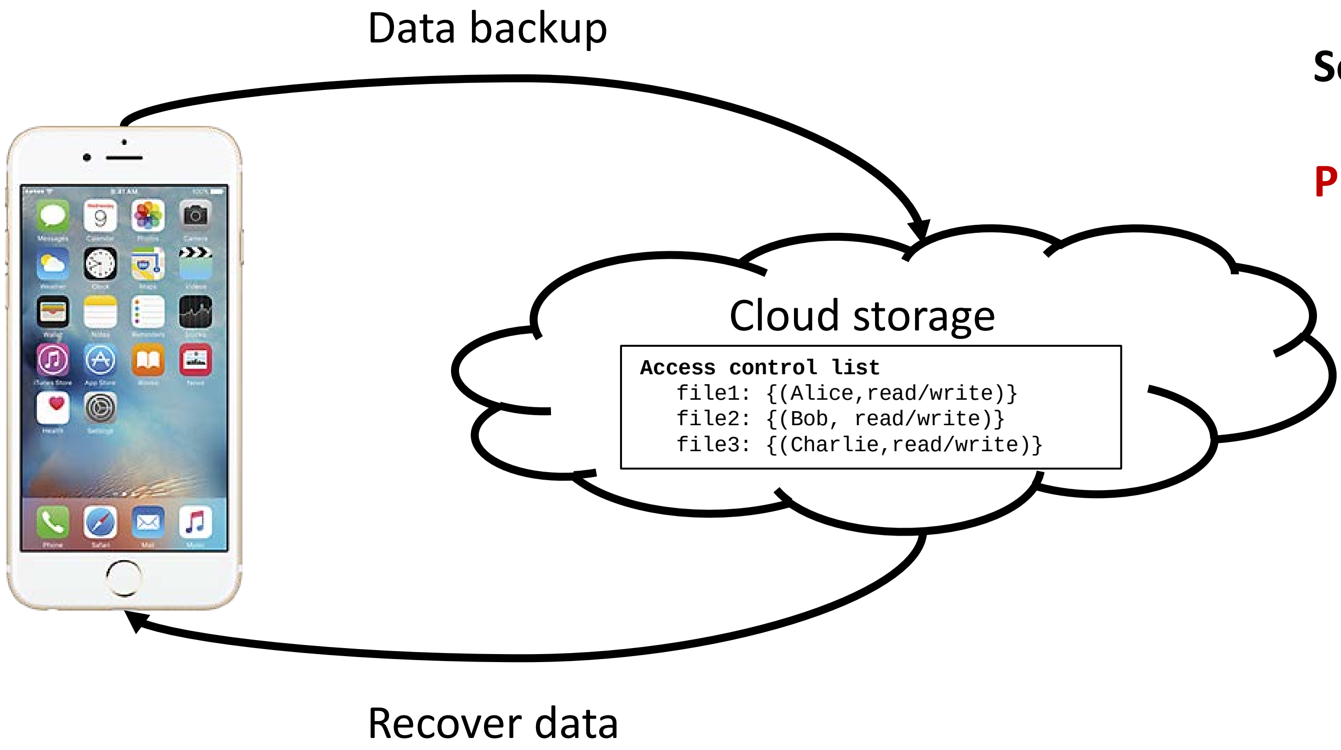
Solution 1: Access Control

Use case: secure backups of iPhones

THREAT MODEL
- Other users
- Cloud provider
- Apple
- External parties (with subpoenas)

Problem statement

We want users to be able to back up their data, and be the only ones that can access it



Solution 1: Access Control

Problems

Where is the TCB?

Apple and the Cloud are in your threat model!

Who has access to the data in the cloud?

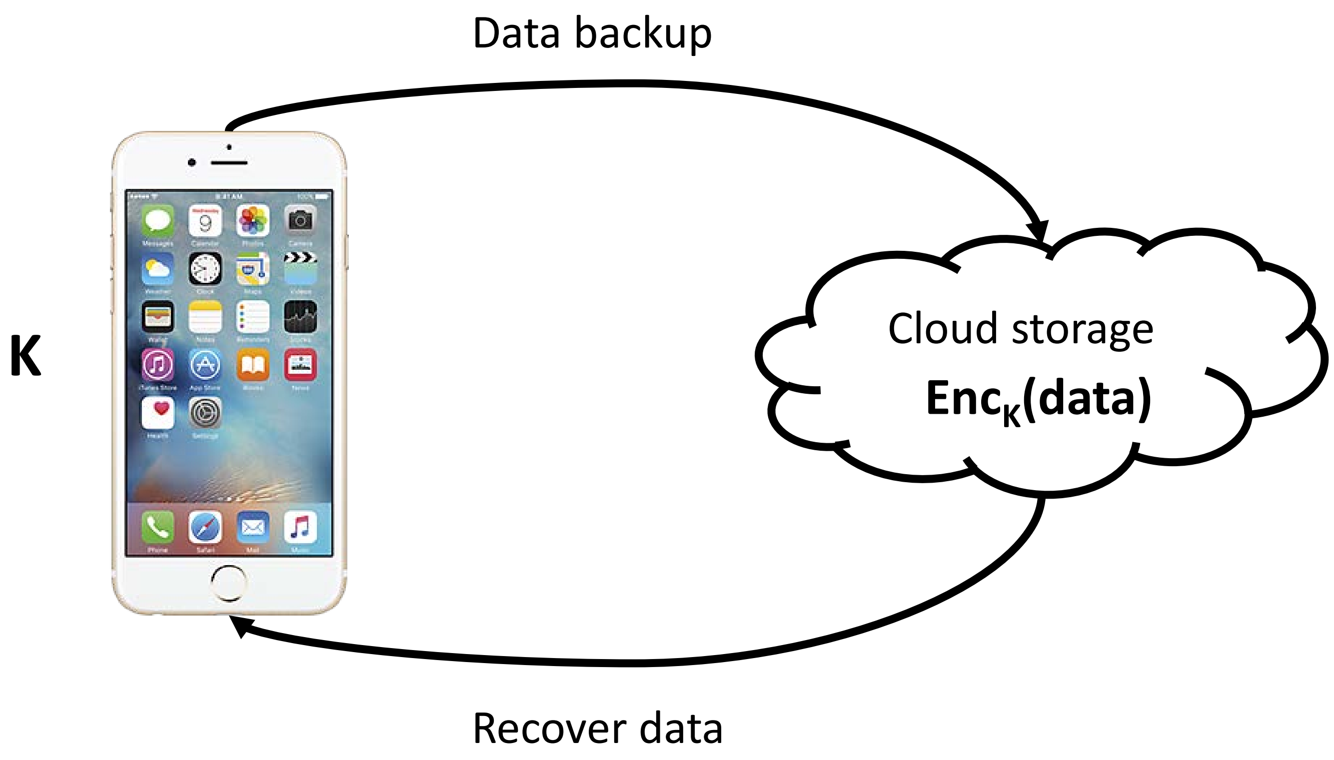
Even if Apple and the Cloud *were trusted* for running the system, access control just avoids that other users and third parties can access the data. (and this does not apply to third parties with subpoenas)

Use case: secure backups of iPhones

THREAT MODEL
- Other users
- Cloud provider
- Apple
- External parties (with subpoenas)

Problem statement

We want users to be able to back up their data, and be the only ones that can access it



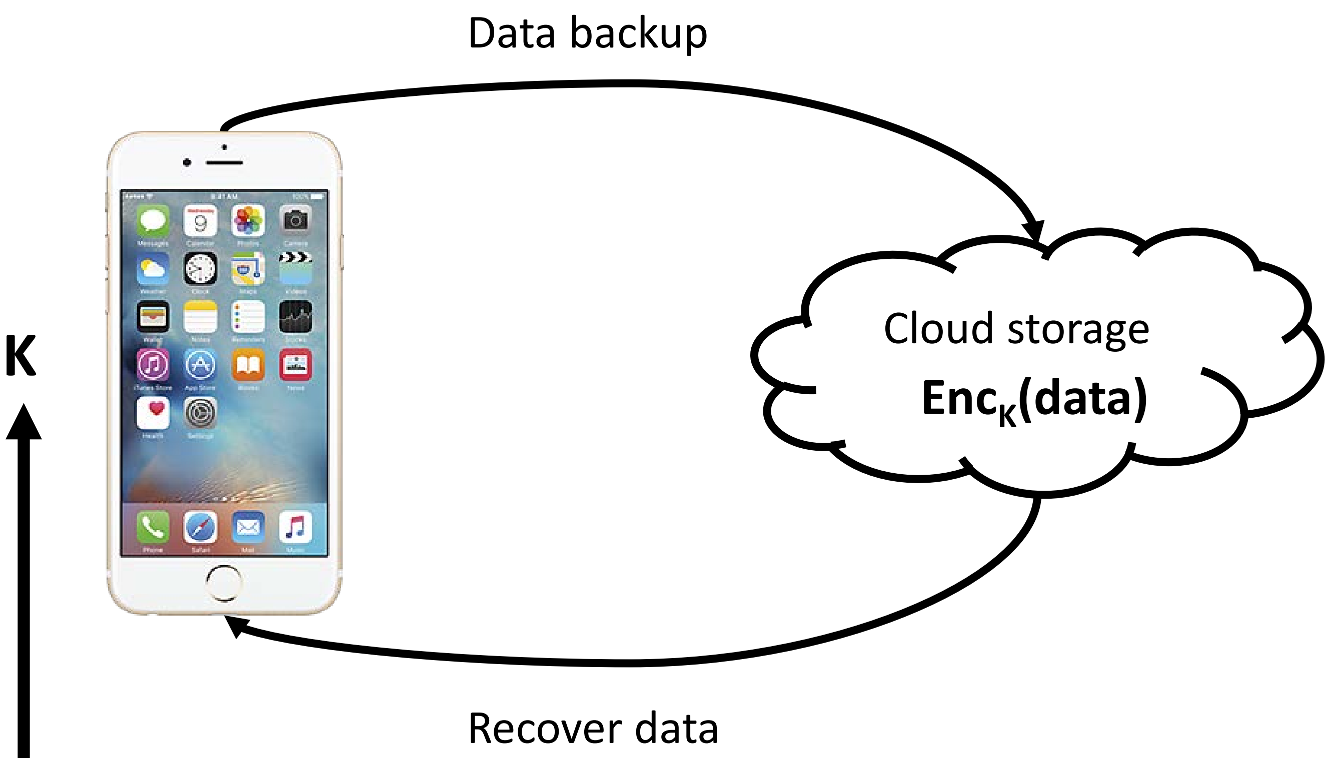
Solution 2: Cryptography

Use case: secure backups of iPhones

Threat Model
- Other users
- Cloud provider
- Apple
- External parties (with subpoenas)

Problem statement

We want users to be able to back up their data, and be the only ones that can access it



Solution 2: Cryptography

Problems

What happens if you lose the phone?

Good cryptography is made so that if you do not have the key you cannot decrypt the data

A backup system based on a key stored in the phone is pretty useless…

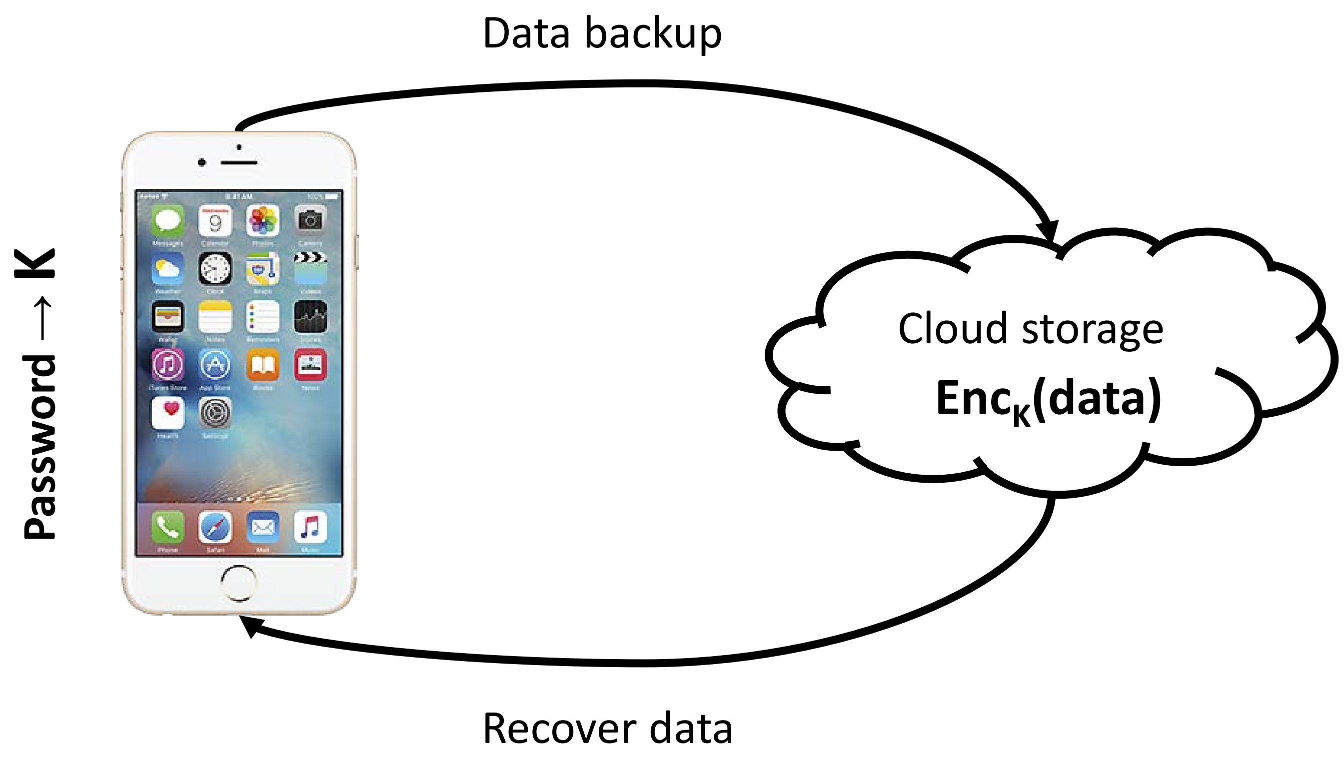
Cryptographic keys are large and cannot be remembered, they must be stored on the phone

Use case: secure backups of iPhones

Threat Model
- Other users
- Cloud provider
- Apple
- External parties (with subpoenas)

Problem statement

We want users to be able to back up their data, and be the only ones that can access it



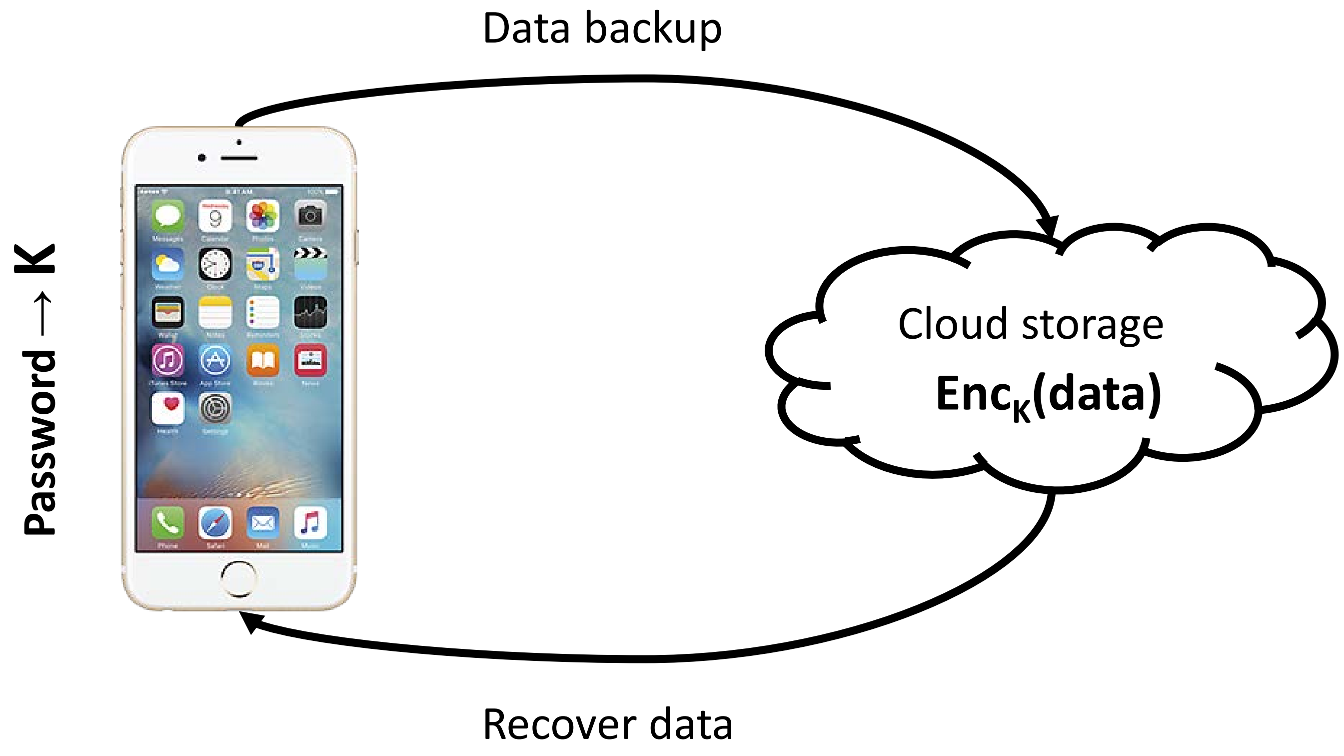
Solution 3: Password to bootstrap cryptography

Use case: secure backups of iPhones

THREAT MODEL
- Other users
- Cloud provider
- Apple
- External parties (with subpoenas)

Problem statement

We want users to be able to back up their data, and be the only ones that can access it



Solution 3: Password to bootstrap cryptography

Problems

You can attempt to retrieve data without the phone

Cannot rely on “what you have”

How easy it is to guess a password?

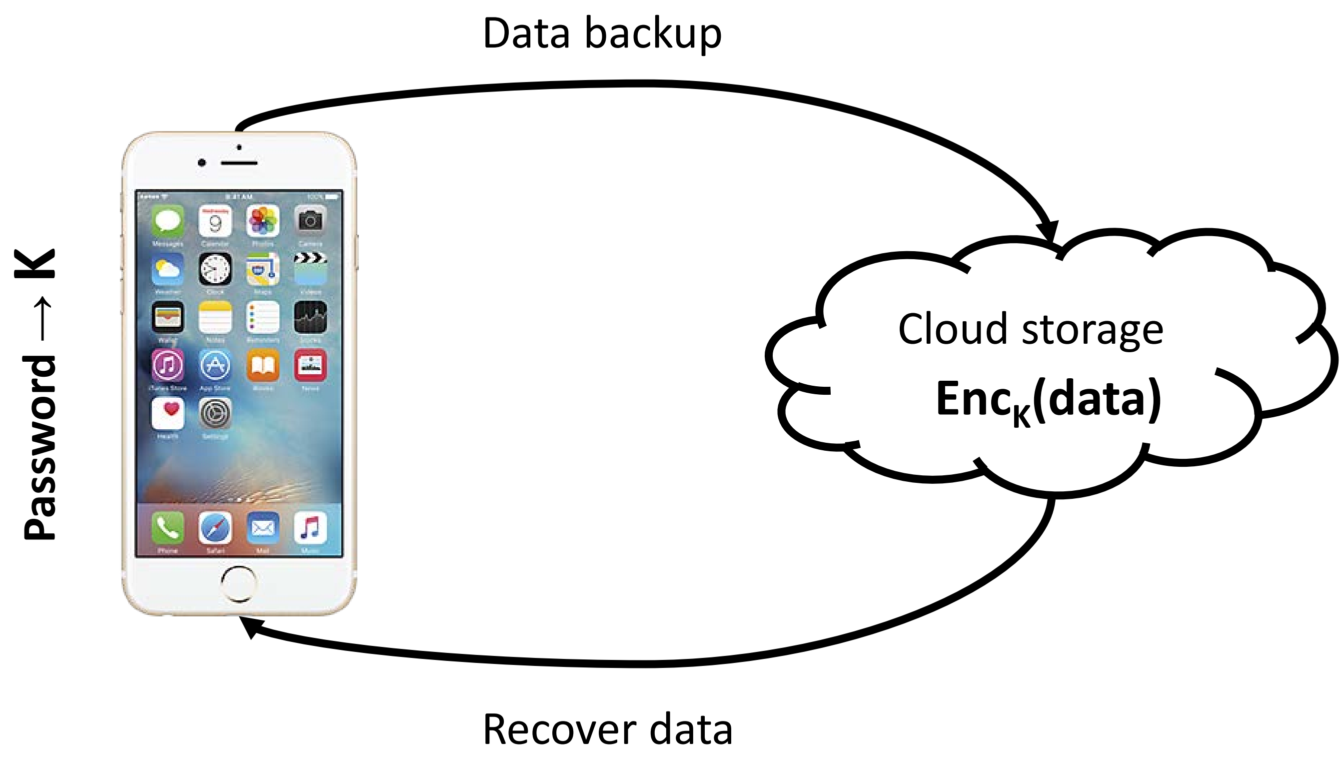
Humans can rarely able to remember more than 8 characters (and not random!)

Remember all the problems from the Authentication lecture

Use case: secure backups of iPhones

Problem statement

We want users to be able to back up their data, and be the only ones that can access it

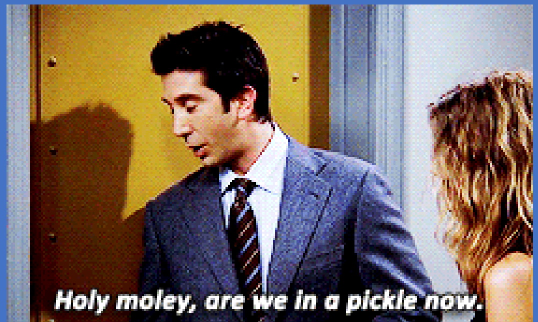


Solution 3: Password to bootstrap cryptography

CHALLENGE

How to decrypt data once the phone is lost,
Given that users can only remember a short passcode (6/8 characters)

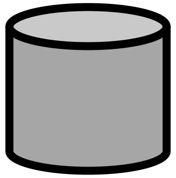
How to limit passwords to only a few attempts?
We do not have a TCB that can check!!



Apple’s secure backup solution: storing data

User’s device

iCloud server



Cloud storage

Hardware Security Module (HSM)

(SK_{HSM}, PK_{HSM})

Pair asymmetric keys

Encrypting a message with PK_{HSM} ensures that **only** the HSM can decrypt the content of the message

Apple's secure backup solution: storing data

User's device

iCloud server

(PK_{HSM} is hardcoded into the device)

1. Generate random symmetric key **k**
2. Encrypt phone data using **k**, and send it to the cloud
3. Send key **k** to the HSM, encrypted with PK_{HSM}

2. Enc(**k**, data)

Cloud storage

3. Enc(PK_{HSM}, [user, passcode*, **k**])

Hardware Security Module (HSM)

(SK_{HSM}, PK_{HSM})

User	Passcode*	K

*This is a simplification! In reality the HSM does not directly store the passcode, instead it uses a cryptographic protocol (PAKE/SRP - http://srp.stanford.edu/whatisit.html)

Apple's secure backup solution: retrieving key

1. Login →

2a. $c = Enc(PK_{HSM}, passcode)$ →

← 2d. k or None

iCloud server

2b. auth(user, c) →

← 2c. k or None

Hardware Security Module (HSM)

(sk_{HSM}, pk_{HSM})

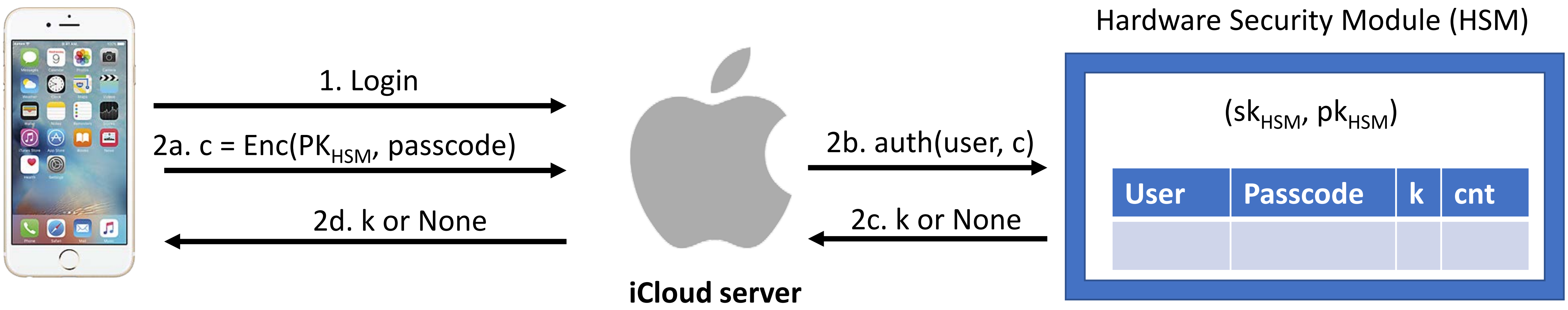
User	**Passcode**	**k**	**cnt**

1. Login to iCloud
2. Send passcode to HSM
3. Retrieve key k

```
Procedure auth(user, c)
passcode, PK_user = Dec(SK_HSM, c)

If passcode is correct & cnt < MAX:
    return k
Else:
    increment cnt
```

Apple's secure backup solution: retrieving key



1. Login to iCloud
2. Send passcode to HSM
3. Retrieve key k

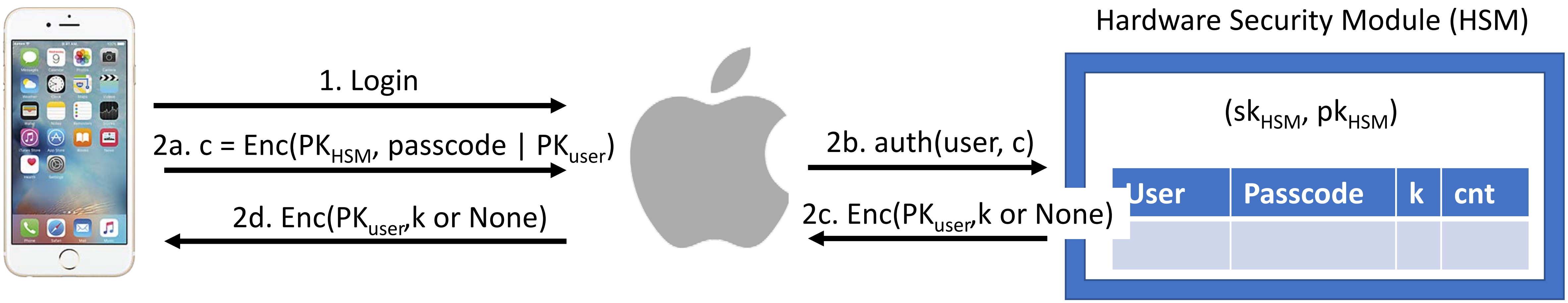
```
Procedure auth(user, c)
passcode, PK_user = Dec(SK_HSM, c)

If passcode is correct & cnt < MAX:
    return k
Else:
    increment cnt
```

Is there a security problem here?
(think about the threat model)

THREAT MODEL
- Other users
- Cloud provider
- Apple
- External parties (with subpoenas)

Apple's secure backup solution: retrieving key

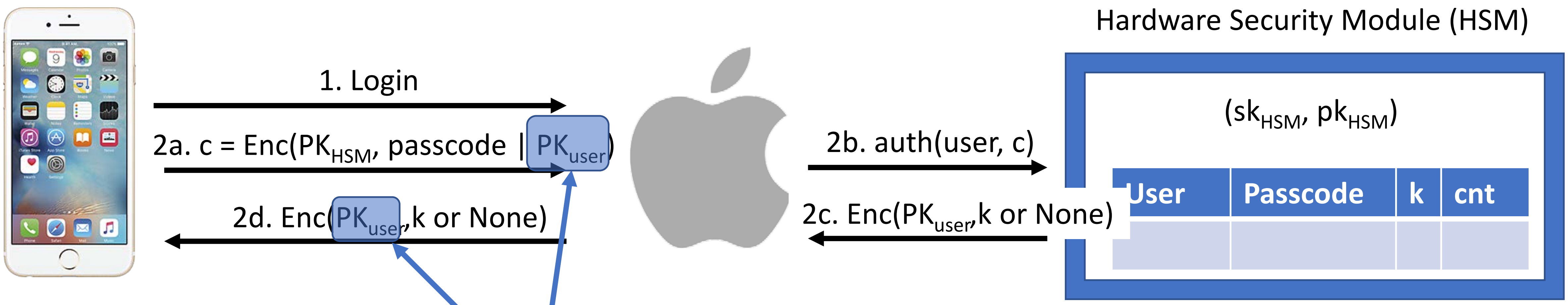


1. Login to iCloud
2. Send passcode to HSM
3. Retrieve key k

```
Procedure auth(user, c)
passcode, PK_user = Dec(SK_HSM, c)

If passcode is correct & cnt < MAX:
    return Enc(PK_user, k)
Else:
    increment cnt
```

Apple’s secure backup solution: retrieving key



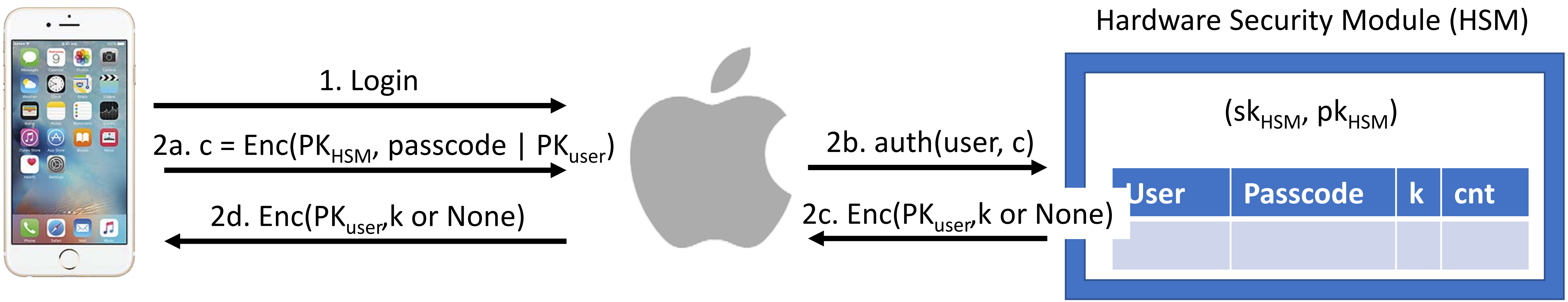
1. Login to iCl
2. Send passc
3. Retrieve ke

A symmetric key could also work but the real protocols (PAKE/SRP) require the use of asymmetric cryptography

```
Procedure auth(user, c)
passcode, PK_user = Dec(SK_HSM, c)

If passcode is correct & cnt < MAX:
    return Enc(PK_user, k)
Else:
    increment cnt
```

Apple's secure backup solution: retrieving key



1. Login to iCloud
2. Send passcode to HSM
3. Retrieve key k

What properties should the HSM satisfy?

```
Procedure auth(user, c)
passcode, PK_user = Dec(SK_HSM, c)

If passcode is correct & cnt < MAX:
    return Enc(PK_user, k)
Else:
    increment cnt
```

Isolation: Key property of secure hardware

ISOLATION

mechanism to constrain who and what has access to programs and data

- Trusted hardware offers one clear entry-point or interface to interact with the software

Isolation: Key property of secure hardware

Under a very strong adversary with physical access

ISOLATION

mechanism to constrain who and what has access to programs and data

- Trusted hardware offers one clear entry-point or interface to interact with the software

In case of hardware security modules/smart cards:

- **Tamper resistance**: hard to open
- **Tamper evident**: you can see if it has been opened
- **Tamper responsive**: delete keys when attacked
- **Resistance to side-channel attacks** and physical probing

Apple’s secure backup: protecting software

Apple’s HSMs run custom software. This software can be updated.

How would you protect this software?

Apple's secure backup: protecting software

Apple's HSMs run custom software. This software can be updated.

How would you protect this software?

1. Protect access keys: Apple's HSMs require smartcards to update, store in sealed and tamper evident bags while not used
2. Install software
3. Physically destroy access cards

Apple’s secure backup: protecting software

Apple’s HSMs run custom software. This software can be updated.

How would you protect this software?

1. Protect access keys: Apple’s HSMs require smartcards to update, store in sealed and tamper evident bags while not used
2. Install software
3. Physically destroy access cards

Security Principle #5
Separation of privilege

Apple's secure backup: protecting software

Apple's HSMs run custom software. This software can be updated.

How would you protect … ware?

1. Protect access keys: Appl… equire smartcards to update, store in sealed and ta… ent bags while not used
2. Install software
3. Physically … cess cards.

Still need to trust Apple to install secure software in the first place

Comparing Smart Cards and HSMs

- **Smart cards**
  - Low power devices, small set of cryptographic operations supported by a cryptographic co-processor
  - Usually programmed using Java Card (stripped down version of Java)
  - Programming these is not easy (memory management, byte-level interface)

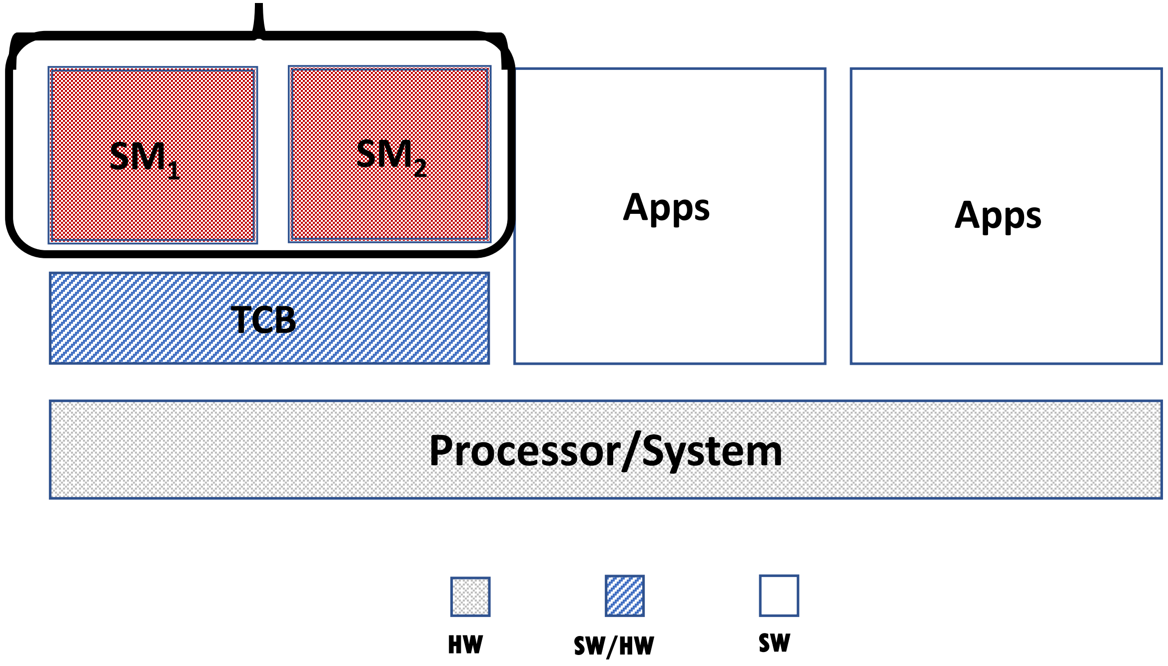
- **HSMs**
  - Can be <u>much more</u> powerful, support larger class of cryptographic primitives
  - Used to generate, store, manage cryptographic keys
  - Usually accessed via PKCS#11 interface, some allow generic programming

Trusted Hardware Key Properties

2. **Attestation**

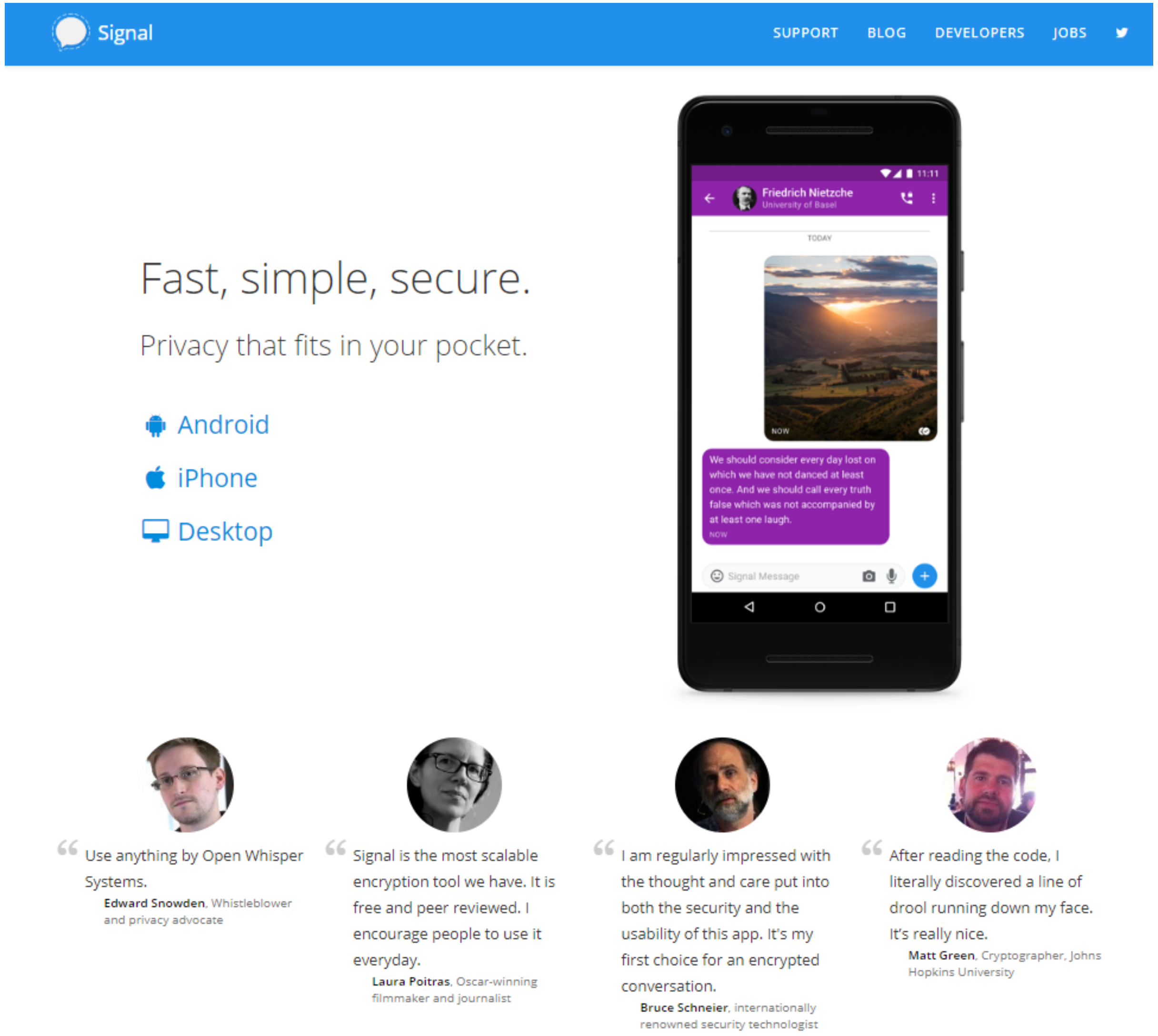
A note on architecture

Trusted execution environments built on
special processor ← Trusted Computing Base



Secure Enclave

Example: Private contact discovery in Signal



Signal is a messaging application designed to offer strong privacy protection

End-to-end encryption and more

Keys stored on the device

Open sourced!!

Security Principle #4
Open desing

https://signal.org/

Example: Private contact discovery in Signal

Who of my contacts uses signal?

Contact	Phone nr.
Alice	+41 12345689
Bob	+31 78492910
Charlie	+42 81992938

THREAT MODEL

Who do you want to protect against/why?

Signal users
+32 81477818
+41 12345689
+1 772888188
…

Example: Private contact discovery in Signal

Who of my contacts uses signal?

Contact	Phone nr.
Alice	+41 12345689
Bob	+31 78492910
Charlie	+42 81992938

THREAT MODEL
Who do you want to protect against/why?

Protect from Signal
Privacy: do not want to leak any information to Signal about who my contacts are. Especially not about those not using Signal.

Signal users
+32 81477818
+41 12345689
+1 772888188
…

Private contact discovery Signal
Can we solve it with access control?

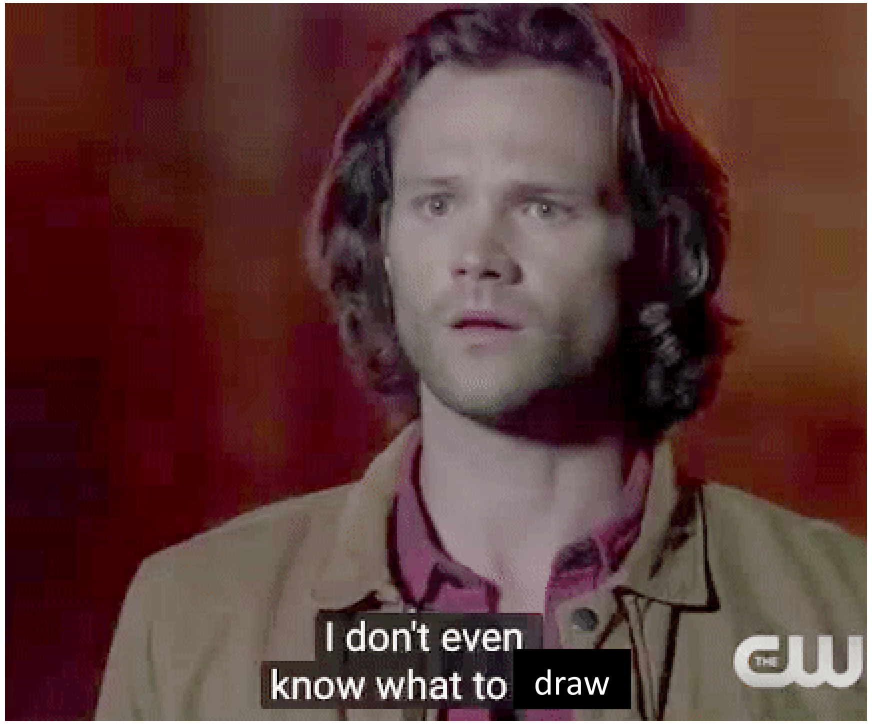
THREAT MODEL
Signal

Protect privacy of contacts

Private contact discovery Signal
Can we solve it with access control?

THREAT MODEL
Signal

Protect privacy of contacts



No, we cannot… same problem as before, where is the TCB? Signal is a threat in the threat model…

Private contact discovery Signal
Can we solve it with cryptography?

THREAT MODEL
Signal

Protect privacy of contacts

Private contact discovery Signal
Can we solve it with cryptography?

THREAT MODEL
Signal

Protect privacy of contacts

1. Enc(PK_{signal}, Set of contacts: X = {+31…, +41…, …})

2. Dec(Sk_{signal}, Set of contacts)

Private contact discovery Signal
Can we solve it with cryptography?

Threat Model
Signal

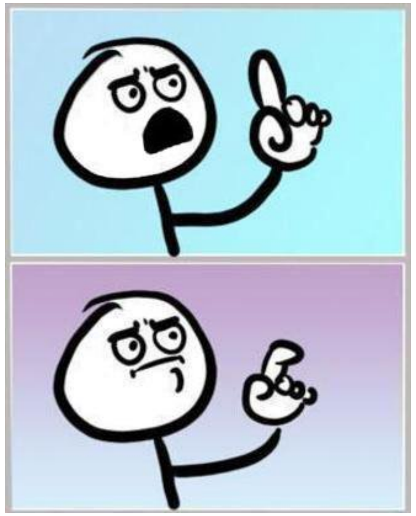
Protect privacy of contacts

1. Enc(PK_{signal}, Set of contacts: X = {+31…, +41…, …})

2. Dec(Sk_{signal}, Set of contacts)

OH WAIT!

The idea was to <u>not</u> show the contacts to Signal



Private contact discovery Signal
Can we solve it with more modern cryptography?

Set of contacts:
X = {+31…, +41…, …}

Set of Signal users:
Y = {+31…, +41…, …}

Private Set Intersection (PSI)

X ∩ Y

Protocol in which two parties jointly compute the intersection of their private input sets

Challenge:
Signal's set is very big. Current modern cryptography is still too slow to deploy in practice*.

*Latest results hint it is possible

More on modern cryptography at the end of the course. So far do not worry about this slide

Private contact discovery Signal Using Intel SGX

Set of contacts:
X = {+31…, +41…, …}

Public key of enclave: PK

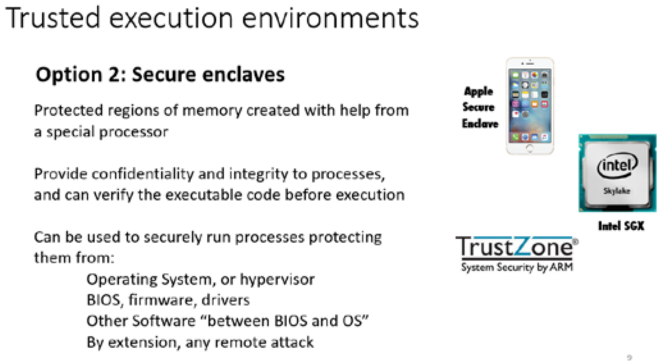
Enc(PK, X)

$X \cap Y$

Set of Signal users:
Y = {+31…, +41…, …}
Keys: (sk,pk)

Compute intersection

Intel SGX enclave



Trusted execution environments

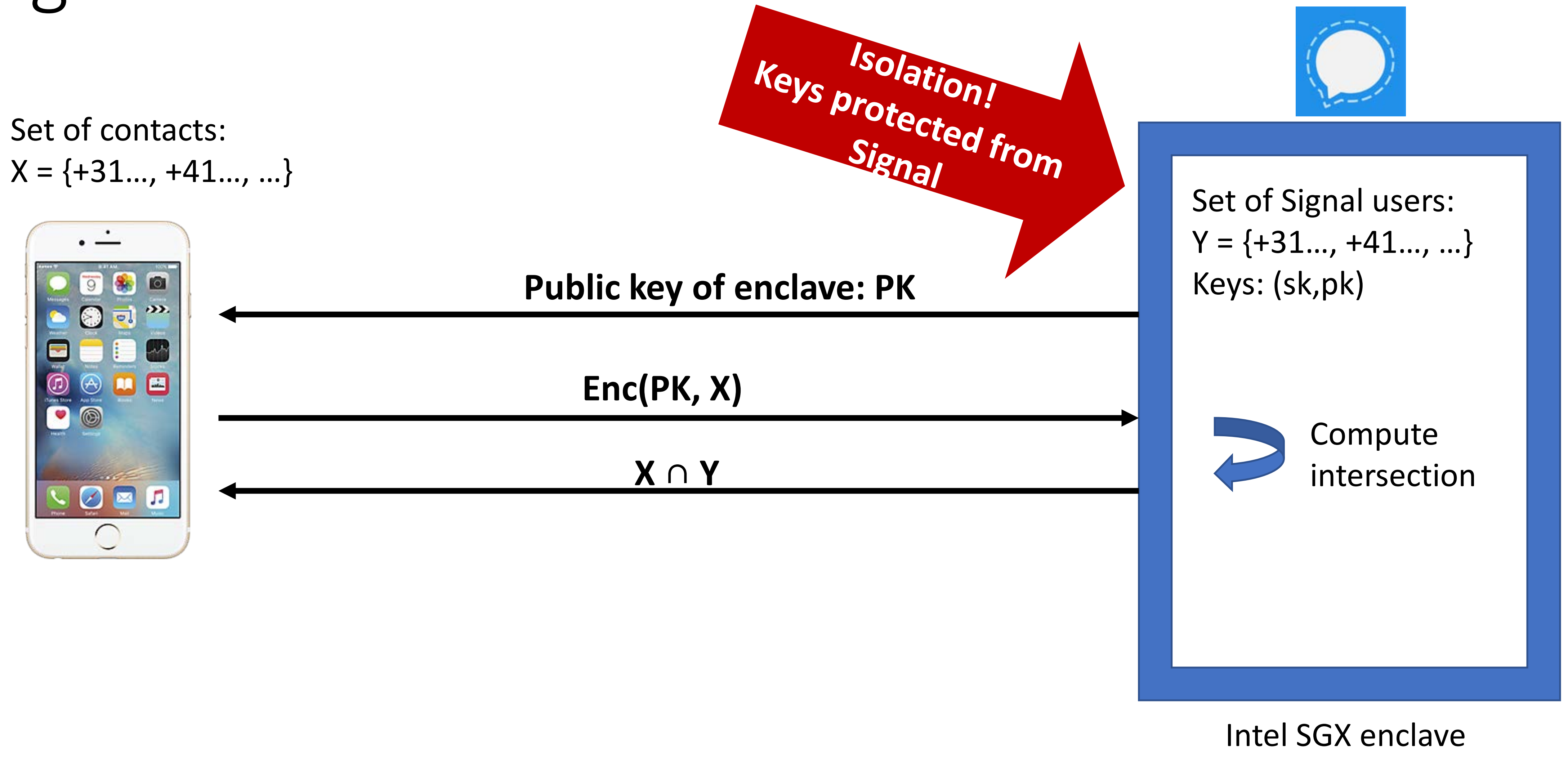
Option 2: Secure enclaves

Protected regions of memory created with help from a special processor

Provide confidentiality and integrity to processes, and can verify the executable code before execution

Can be used to securely run processes protecting them from:
- Operating System, or hypervisor
- BIOS, firmware, drivers
- Other Software "between BIOS and OS"
- By extension, any remote attack

Private contact discovery Signal Using Intel SGX



Private contact discovery Signal Using Intel SGX

Isolation!
Keys protected from Signal

Set of contacts:
X = {+31…, +41…, …}

Public key of enclave: PK

Enc(PK, X)

X ∩ Y

Set of Signal users:
Y = {+31…, +41…, …}
Keys: (sk,pk)

Compute intersection

Intel SGX enclave

CHALLENGE: *how does the phone know the enclave runs the right program?*

Attestation: Key property of secure hardware

ATTESTATION

mechanism that allows a hardware module to prove, to an authorized party, that it is in a specific state

http://web.cs.wpi.edu/~guttman/pubs/good_attest.pdf

Attest there is secure hardware:

the device has a key (endorsement key*) to prove it is genuine

Attest the state of the OS:

after a series of instructions the state of registers is as expected

Attest the state of the code:

signature on the code (needs to correspond with the register values!)

*https://trustedcomputinggroup.org/wp-content/uploads/IWG-EK-CMC-enrollment-for-TPM-v1-2-FAQ-rev-April-3-2013.pdf

Using Intel SGX for contact discovery

Set of contacts:
X = {+31…, +41…, …}

challenge →

← Public key of enclave: PK, **+attestation**

Enc(PK, X) →

← X ∩ Y

Set of Signal users:
Y = {+31…, +41…, …}
Keys: (sk,pk)

Compute intersection

Intel SGX enclave

CHALLENGE: *how does the enclave store this pair of keys? It only has persistent memory for the manufacturer established key!*

Trusted Hardware Key Properties

3. **Sealing**

Sealing: Key property of secure hardware

SEALING

a sealed storage protects private information by binding it to platform-configuration information including the software and hardware being used

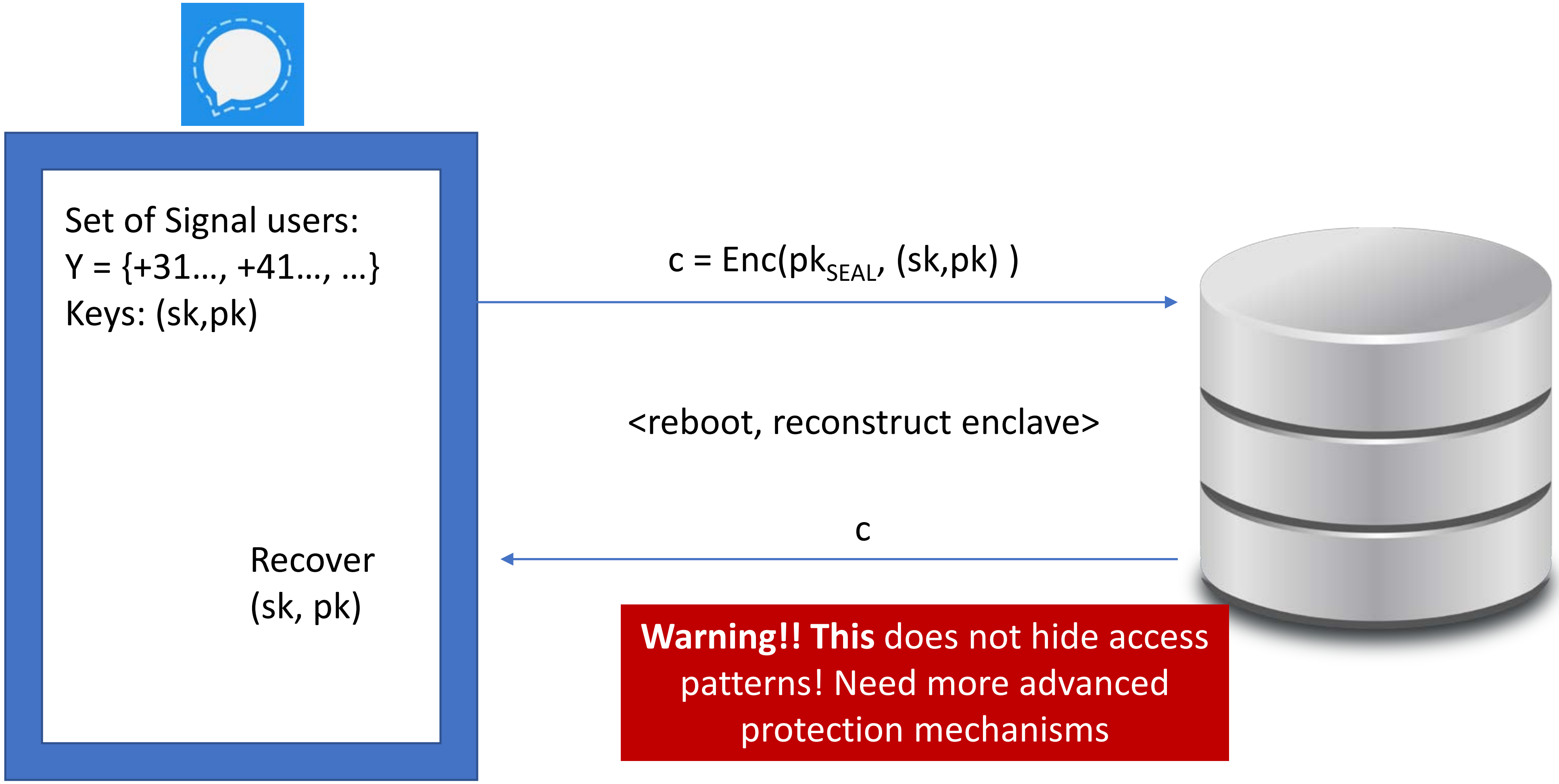
The device derives a key that is tied to its current status (e.g., Platform Configuration Registers (PCRs)) and stores the encrypted data

- Additionally may store authorization information

This way, data can only be decrypted by a device with the **same** status.

e.g., Bitlocker – full disk encryption by Microsoft

Using sealing to store keys



Intel SGX enclave

Side channels: trusted hardware nightmare

Trusted hardware
Relation to declassification (BLP)

Trusted hardware
Relation to declassification (BLP)

1. Do very secure operation

3. Result of very secure operation

How does this remind you of Bell La Padula?

Trusted Hardware

Secure Enclave

Smart Card

Hardware security module

2. Execute very secure operation **with secret information**

Trusted hardware
Relation to declassification (BLP)

1. Do very secure operation

3. Result of very secure operation

This is low level

DECLASSIFICATION

Trusted Hardware

Secure Enclave

Smart Card

2. Execute very secure operation **with secret information**

Hardware security module

This is high level

Should we care about covert channels?

1. Do very secure operation

3. Result of very secure operation

This is low level

DECLASSIFICATION

Trusted Hardware

Secure Enclave

Smart Card

Hardware security module

2. Execute very secure operation **with secret information**

This is high level

Trusted hardware
Relation to declassification (BLP)

1. Do very secur
operation

NO! Attestation can take care of this

This is low level

DECLASSIFICATION

Attestation: Key property of secure hardware

ATTESTATION
mechanism that allows a hardware module to prove, to an authorized party, that it is in a specific state
http://web.cs.wpi.edu/~guttman/pubs/good_attest.pdf

Attest there is secure hardware:
the device has a key (endorsement key*) to prove it is genuine

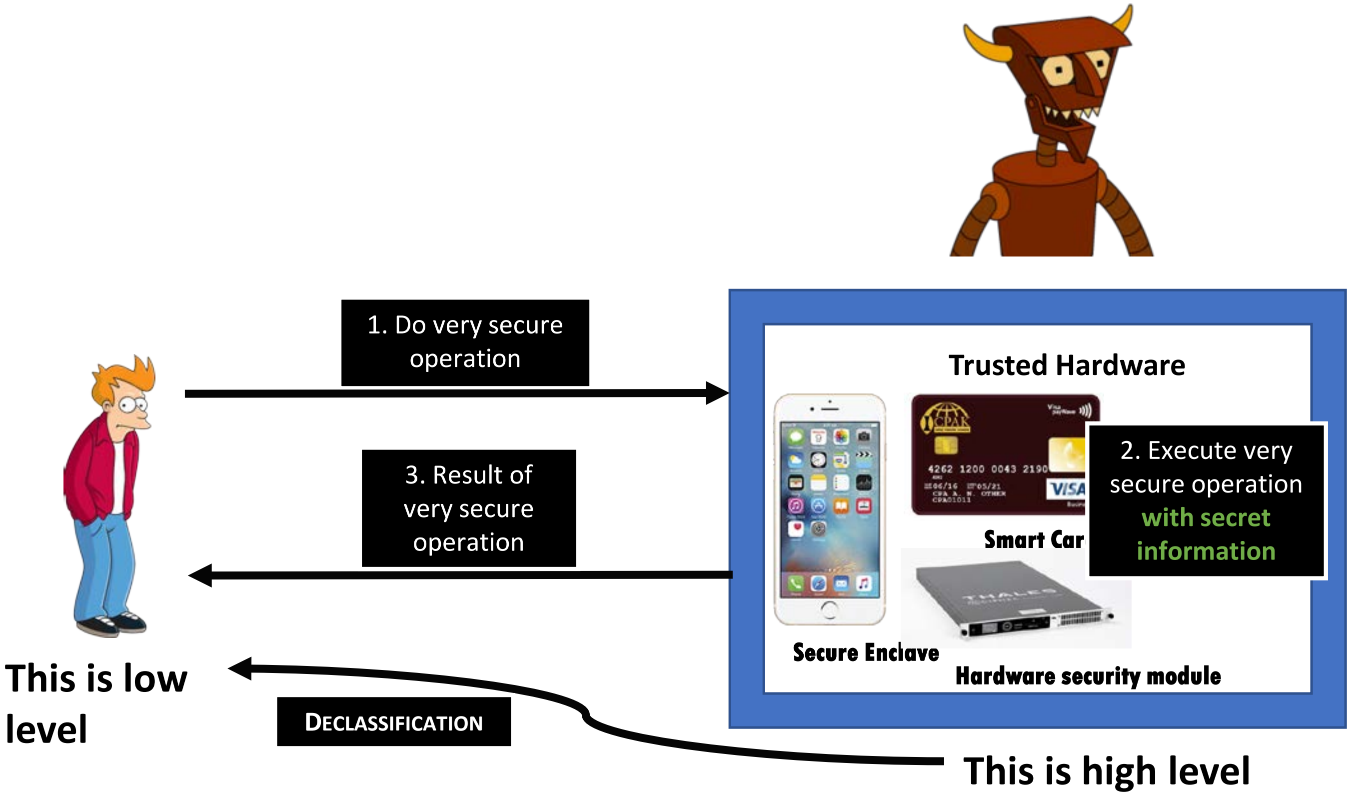
Attest the state of the OS:
after a series of instructions the state of registers is as expected

Attest the state of the code:
signature on the code (needs to correspond with the register values!)

This is high level

Side channels

There cannot be harm coming from inside the trusted hardware, what about looking from outside?



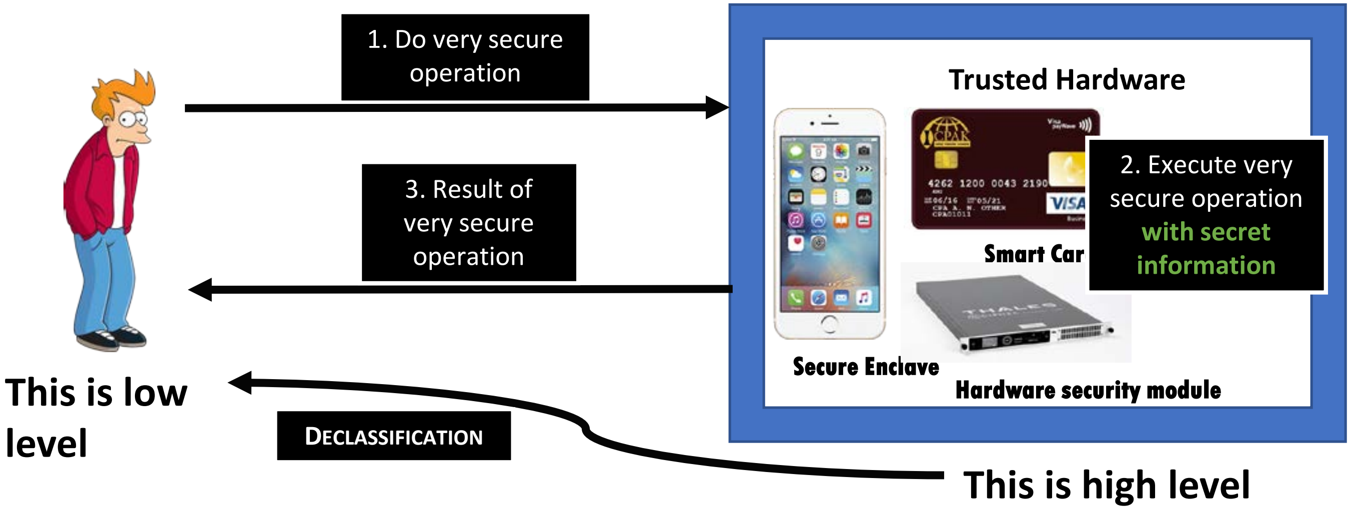
http://bristolcrypto.blogspot.com/2015/06/52-things-number-38-what-is-difference.html

Side channels

There cannot be harm coming from inside the trusted hardware, what about looking from outside?

Side Channel Attacks

Determine the secret key of a cryptographic device by measuring its execution time, its power consumption, or its electromagnetic field.



http://bristolcrypto.blogspot.com/2015/06/52-things-number-38-what-is-difference.html

Side channel types

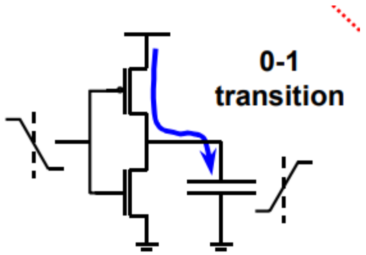
Basic idea: learn the system's secret by observing how different computations are

Time: how different computations take different time

Extremely powerful because **isolation** doesn't help (victim could be remote!)

Power: how different computations consume different power

Input	Output	Current
0 → 0	1 → 1	Low
0 → 1	1 → 0	Discharge
1 → 0	0 → 1	Charge
1 → 1	0 → 0	Low



Electromagnetic: how different computations have different emissions

Timing attack
RSA Cryptosystem

- Key generation:
  - Generate large primes P, Q
  - Compute N=PQ and $\varphi(N)=(P-1)(Q-1)$
  - Choose small e, relatively prime to $\varphi(N)$
  - Compute *unique* d such that $ed = 1 \bmod \varphi(N)$

For completeness, not important for this course

  - **Public key = (e,N)** ; **Private key = d**

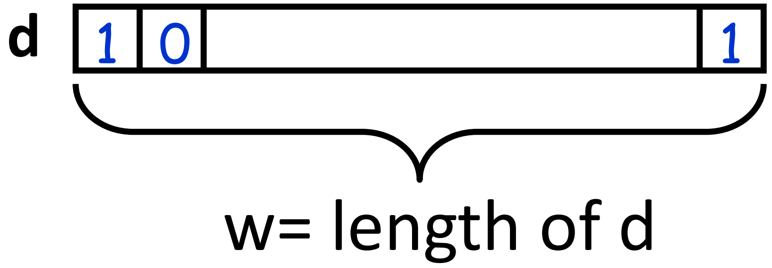
- Encryption of m (*simplified*!): $c = m^e \bmod N$
- Decryption of c: $c^d \bmod N = (m^e)^d \bmod N = m$

Timing attack
Implementing simple RSA in a naïve way

- RSA decryption: compute c^d mod N
  - A modular exponentiation operation

Naive algorithm: square and multiply

$s_0 = 1$
for $k = 0$ to w-1 **do**
    if $d[k] == 1$ **then**
        $R_k = (s \cdot c) \bmod n$
    else
        $R_k = s_k$
    end if
    $s_{k+1} = R_k^2 \bmod n$
end for
Return R_{w-1}



Timing attack
RSA Cryptosystem naïve implementation (3)

$s_0 = 1$
for $k = 0$ to w-1 **do**
  if $d[k] == 1$ **then**
    $R_k = (s \cdot c) \bmod n$
  else
    $R_k = s_k$
  end if
  $s_{k+1} = R_k^2 \bmod n$
end for
Return R_{w-1}

Whether iteration takes a long time depends on the **k**th bit of secret exponent

This takes a while to compute

This is instantaneous

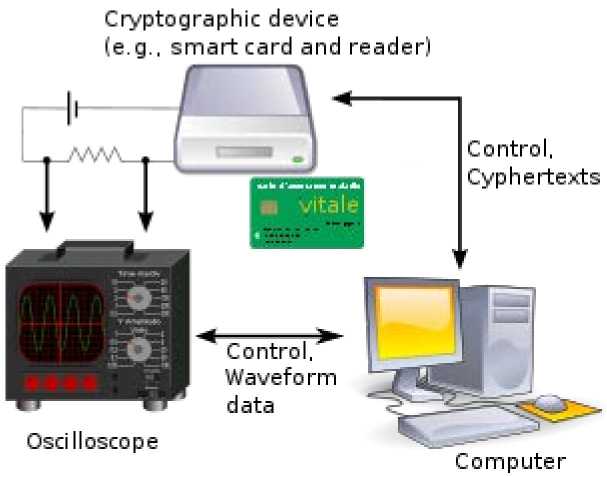
Observation: timing depends on number of 1's

https://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf

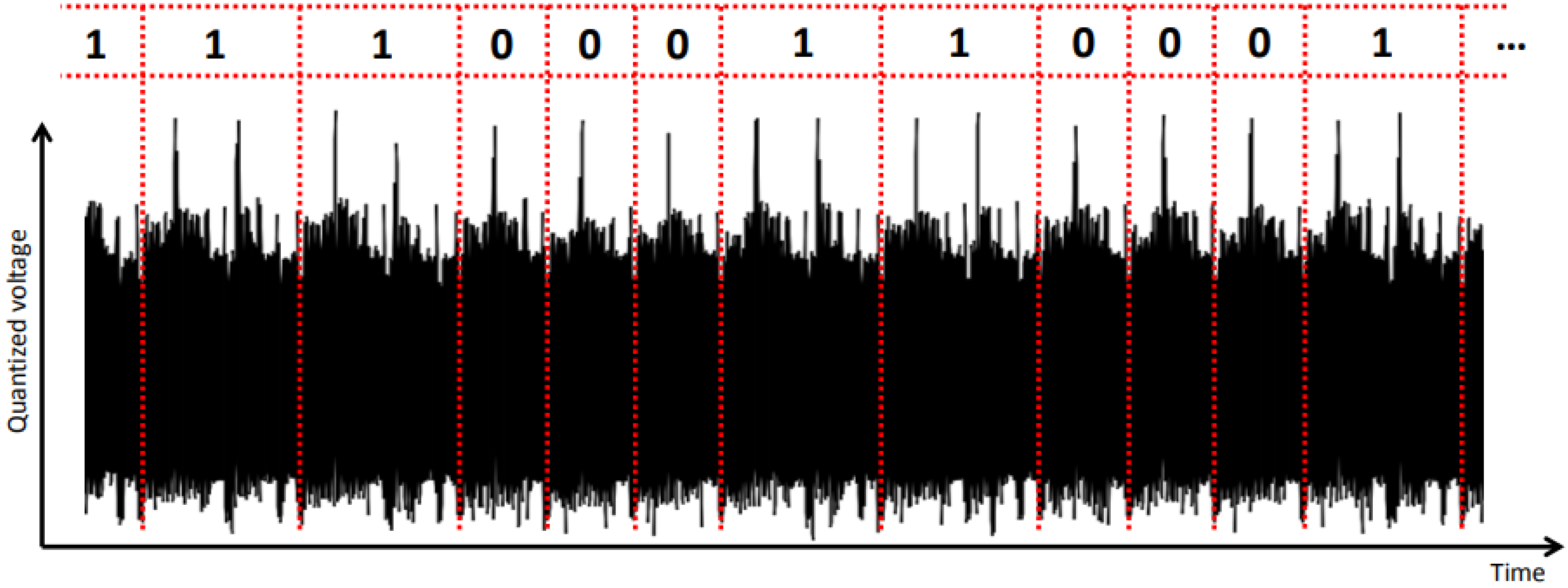
Power Analysis side channel

Elliptic-curve cryptography operation

Power depends on key bits



https://en.wikipedia.org/wiki/Power_analysis



Power trace from Josep Balasch (KU Leuven)

Security

Intel shrugs off 'new' side-channel attacks on branch prediction units and SGX

Been there, mitigated that, got the class actions says Chipzilla

By Simon Sharwood 28 Mar 2018 at 06:31

Intel's shrugged off two new allegations of channel attacks.

One of the new allegations was discussed last week, where University of Graz PhD S Michael Schwarz delivered a talk titled "W Intel SGX to stealthily steal Bitcoins."

SGX is Intel's way of creating secure encla "protected areas of execution in memory" t data from disclosure or modification." SGX inaccessible from the OS and even survive

Security

Side channel attacks on graphics processors can enable hackers to spy on web activity and steal passwords

The GPU-based attacks are enabled after the victim downloads an app with a malicious programme

Dev Kundaliya

07 November 2018

0 Comments

GPUs are increasingly being used for non-graphics applications, like scientific research and AI

Research conducted by a team of computer scientists at the University of California Riverside has demonstrated that hackers can target a computer's graphics processing unit (GPU) to steal passwords, break into cloud-based applications and spy on the web activity of a user.

Google

side channel

All Images News Videos Shopping More Settings Tools

About 50.400.000 results (0,29 seconds)

Side channel attacks on graphics processors can enable hackers to ...
www.v3.co.uk - 12 hours ago
Research conducted by a team of computer scientists at the University of California Riverside has demonstrated that hackers can target a ...
Researchers extend side-channel attacks to the GPU
bit-tech.net - 12 hours ago
View all

Side-Channel Vulnerability PortSmash Steals Keys
Infosecurity Magazine - 10 hours ago
The new side-channel vulnerability, called PortSmash, was discovered by researchers Billy Bob Brumley, Cesar Pereida García, Sohaib ul ...

Researchers show Nvidia GPUs can be vulnerable to side channel ...
TechSpot - 3 hours ago
In a paper titled "Rendered Insecure: GPU Side Channel Attacks are Practical," the computer scientists describe how they were able to reverse ...
Researchers Find GPUs Can Be Used to Spy on Users
Overclockers Club - 4 hours ago
View all

'PortSmash' Brings New Side-Channel Attack to Intel Processors
Dark Reading - 22 hours ago
A new Intel side-channel vulnerability dubbed PortSmash promises to lay encryption keys open to discovery by threat actors. PortSmash uses ...

Intel Skylake and Kaby Lake CPUs vulnerable to Portsmash side ...
The INQUIRER - 6 Nov 2018
IT'S NOT A GOOD YEAR for CPU security as security boffins have discovered yet another side-channel vulnerability in Intel Skylake and Kaby ...
PortSmash: Newly discovered side channel attack found with Intel ...
MSPoweruser - 6 Nov 2018
View all

PortSmash is the Latest Side-Channel Attack Affecting Intel CPUs
InfoQ.com - 4 Nov 2018
We detect port contention to construct a timing side channel to exfiltrate information from processes running in parallel on the same physical ...
New PortSmash Side-Channel Vulnerability (CVE-2018-5407)

Side channel countermeasures

GOAL: Prevent secret inference from observable state

- **Hiding**: lowers signal to noise ratio
  - Noise generator, randomized execution order, dualrail/asynchronous logic styles…
- **Masking**: (secret sharing) splits state into shares; forces adversary to recombine leakage
  - Boolean or arithmetic masking, Higher-order masking
- **Leakage Resilience**: prevents leakage aggregation by updating secret

Final remark: Trusted hardware = no trust on anyone?

Backup data from Apple

- **HSM manufacturer.** It controls the production of the "black box".
- **Apple.** To install the correct code the first time (attestation can help).

Private contact discovery for Signal

- **Intel.** It controls the production of the (black box) SGX system. It also controls the attestation keys built into every device.
- **Signal.** Not for running the correct set intersection code (the attestation allows to check that), but for not leaking any data from the smart phone application.

Trusted hardware - Lessons learned

It exists, and offers three properties 

isolation: it is not possible to “peek” inside

attestation: it can prove that it does what you think it is doing

sealing: it can store secrets in memory that can only be recovered by itself

Building trusted hardware is difficult: side channel attacks

Trusted hardware means to trust the manufacturer!

Separation of privilege! Use several cards with advanced cryptography

Trusted hardware - Lessons learned

It exists, and offers three properties

isola

atte

seal ed by itself

Build

Trus cryptography

MOST IMPORTANT LESSON

Having trusted hardware doesn't mean you don't need to think about protocols!!