

Computer (Cyber) Security

Definition and Challenges

Designing Secure Systems 2017/18

David Galindo

Based on slides by Nicolas Courtois (UCL)

What is security?



Class brainstorm

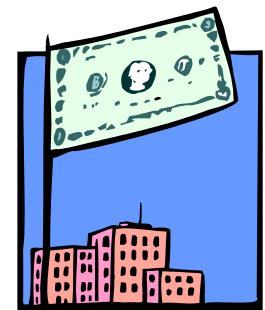
Security: protect assets

What assets?

- **Money** [economic security]

But NOT ONLY MONEY

- **Life** and the quality of life
- **Food** security
- **Freedom, justice, etc...**



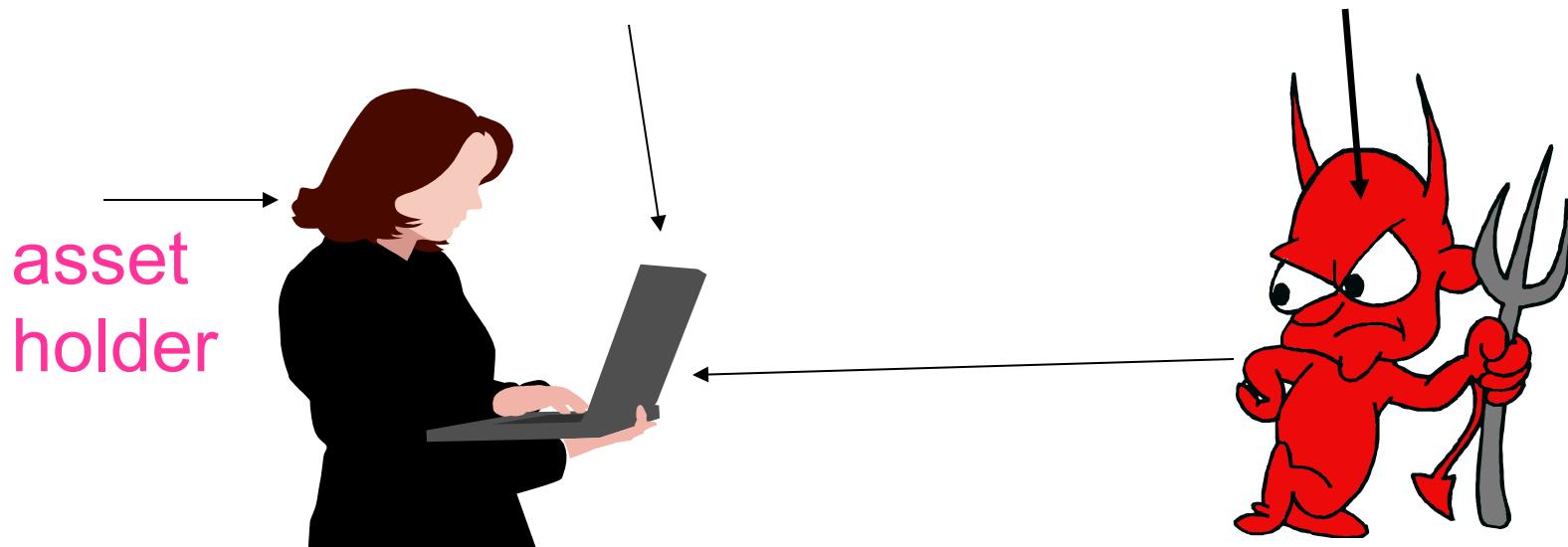
Computer Security



Common Criteria [ISO15408] :

an international standard for computer security certification

Protecting Digital Assets from Threats





Security \geq Safety

Difference:

security protects against **intentional** damages...

Notion of an

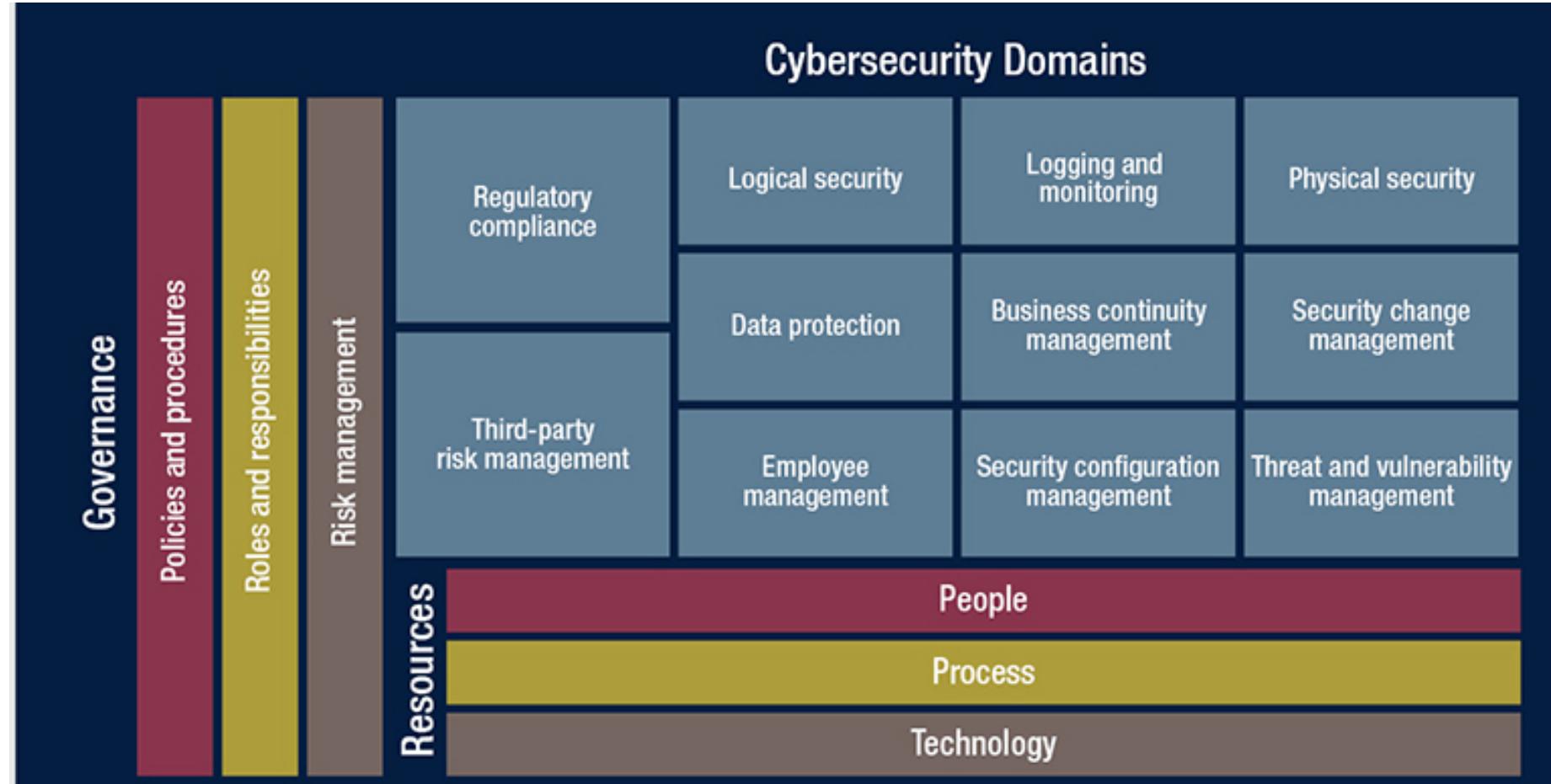
- Attacker / Adversary
- Attack



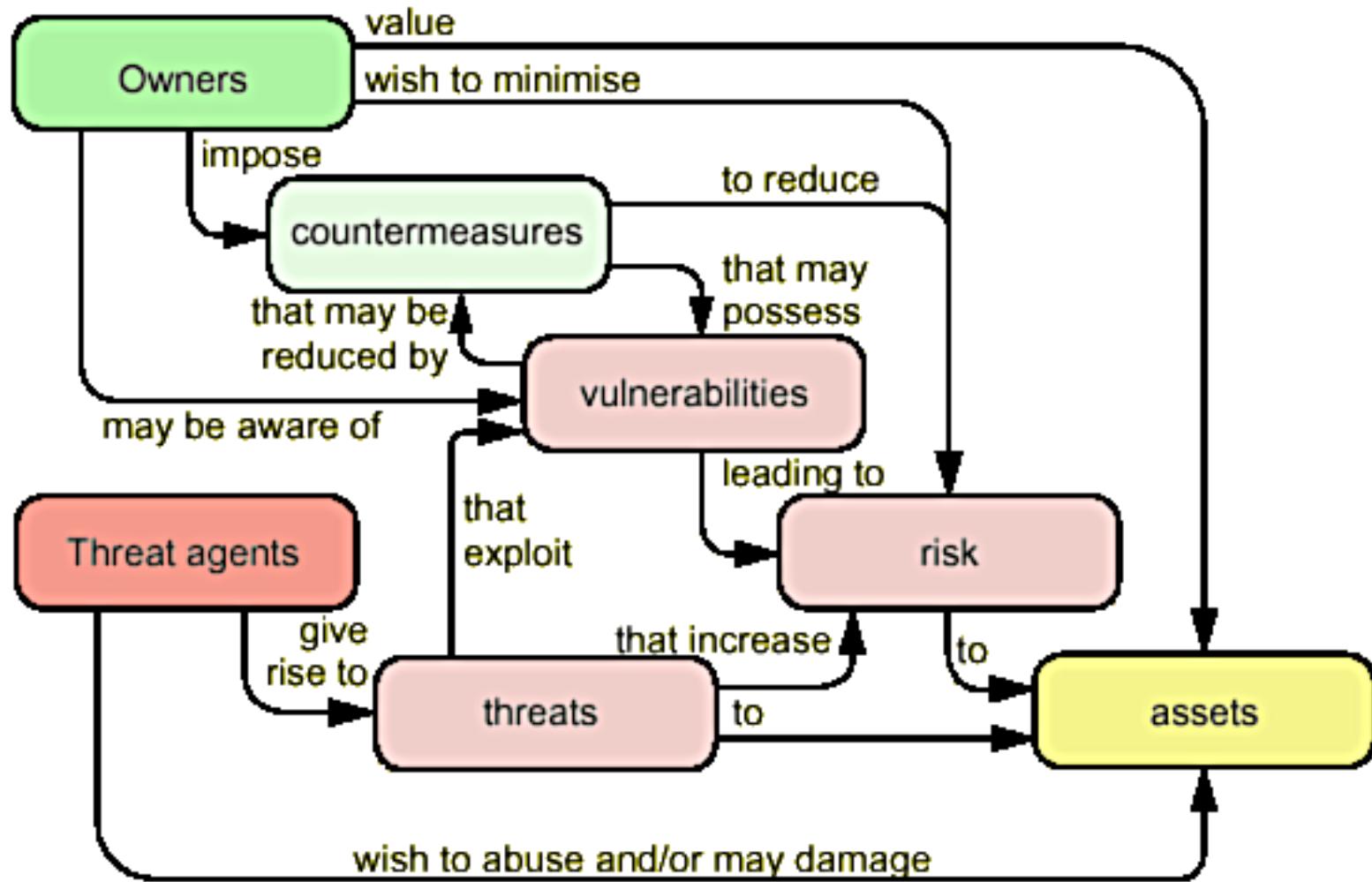
Dimensions of Computer Security

- Physical vs. Logical
- Psychological / Human
 - very different!
- Organizational / Business
 - very different!

Computer Security Dimensions



Computer Security on one slide



Our Definition of Secure System

Inability for attackers to achieve:

1. Adversarial goal
2. By means of: money, human resources, computing power, memory, risk, expertise... **resources of the adversary**
3. Access to the system



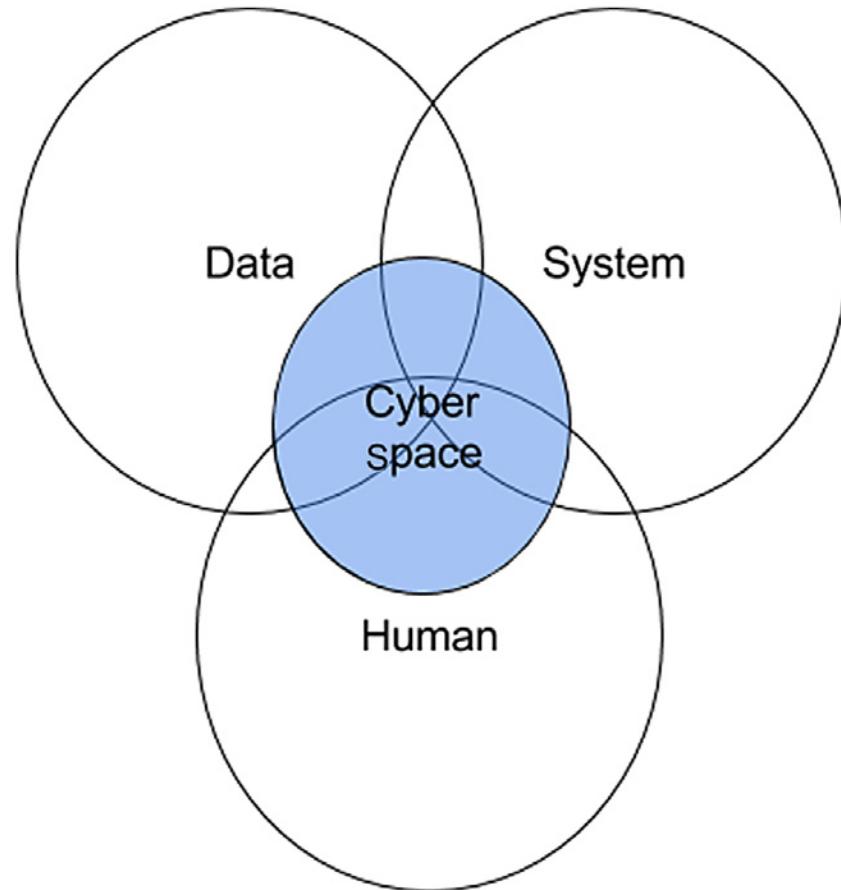
Main Adversarial Goals

Breaching any of:

- Confidentiality
- Integrity
- Authenticity
- Availability
- Accountability

Why is computer security hard?

Class brainstorm



Cyber space at the overlap of data, system, and human

Computer Industry and Security

Tech Background: “Industry Standards” such as:

- Intel CPU
- RAM and hard drives
- C language
- UNIX / Windows
- TCP/IP
- HTTP
- TLS

Social-Econ Background:

Science background:

Computer Industry and Security

“Industry Standards”

Social-Econ Background:

Science background:

- What technology “enablers”(computers) and “disablers” (cryptology,HWSec) can/ cannot achieve?
- How to define / classify security problems and find “good” solutions

Computer Industry and Security

“Industry Standards”

Social-econ background:

- software/hardware economics:
 - which industry dominates which
 - free market triumphs and disasters
- **humans that cannot be bothered to obey the policy...**
- bureaucratic organisations that just cannot get their best interest (?) right
- slow adoption of the technology academics/companies are creating
- adoption barriers
- theory vs. practice
- laws / regulations

insecure products!



The Product Development cycle

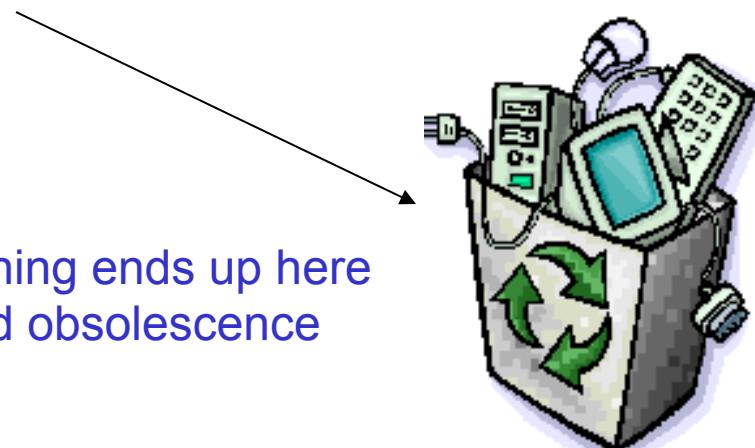
“Industry Standards”

Social-Econ Background:



Science background:

everything ends up here
rapid obsolescence



Attackers



Vocabulary

Attacker / Adversary / Threat Agent



Who are the attackers?

- Adventurous teenagers
- Petty criminals to organized criminals
- Foreign states
- Industrial spies
- Disgruntled employees
- Competitors
- Researchers
- Terrorists
- ...

Attacker means

- Software vulnerabilities
 - Buffer overflow attacks
 - SQL injection attacks
 - Javascript attacks (e.g., XSS)
 - Broken authentication, access control, and session management
- Security misconfiguration

Attacker means continued

- Social engineering
 - Phishing attacks
- Traffic interception (e.g. wireless)
- Hardware/physical attacks
- Ingenuity, hard work, good luck, brute force

Attacker motivation

- Profits and other benefits
 - Crime business
 - Reputation damage
- Political activism, terrorism
- Enjoyment, fame
- Development of science and offensive technology:
 - University researchers
 - Security professionals (defenders)
 - Professional hackers, pen testers, etc...



Recent Trend

The industrialization of hacking:

- division of labour, clear definition of roles
- forming a supply chain
- professional management

Cybercrime actors

- Exploit developers
 - Very smart people who reverse-engineer software
 - Develop and sell exploits packs and kits
- Botnet masters
 - Develop software and control vast numbers of *zombie machines* (i.e. infected by a bot)
 - Rent out their botnet to other actors
- Spammers
 - Advertise links for other actors
- Phishers
 - Setup scam sites to steal information
 - Work with spammers to spread the attack

Cybercrime actors contd

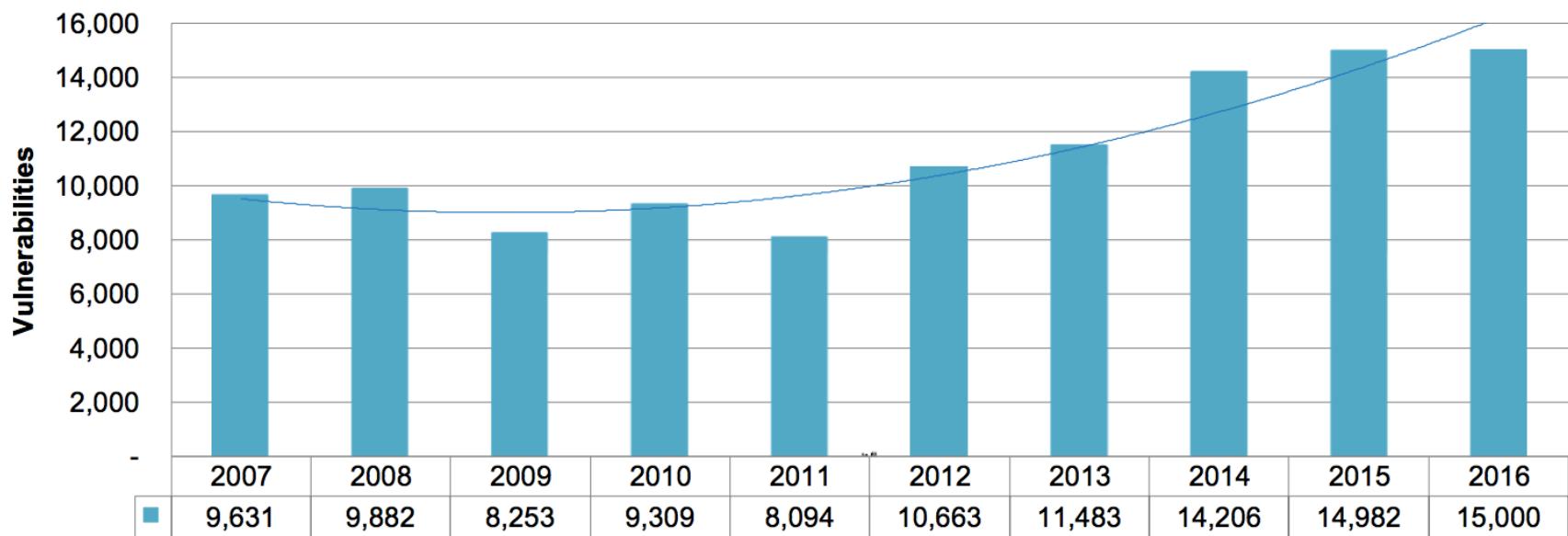
- Counterfeitors
 - Run websites selling fake goods
 - Must be able to clear credit cards
- “Bulletproof” Hosting Providers
 - Offer dedicated servers to other actors
 - Hosted in lawless parts of the Internet
- Carders, Cashiers, and Mules
 - Turn stolen bank accounts and credit cards into cash
 - Help launder money
- Crowdurfers
 - Create, verify, and manage fake accounts
 - Solve CAPTCHAS for a fee

Software vulnerabilities

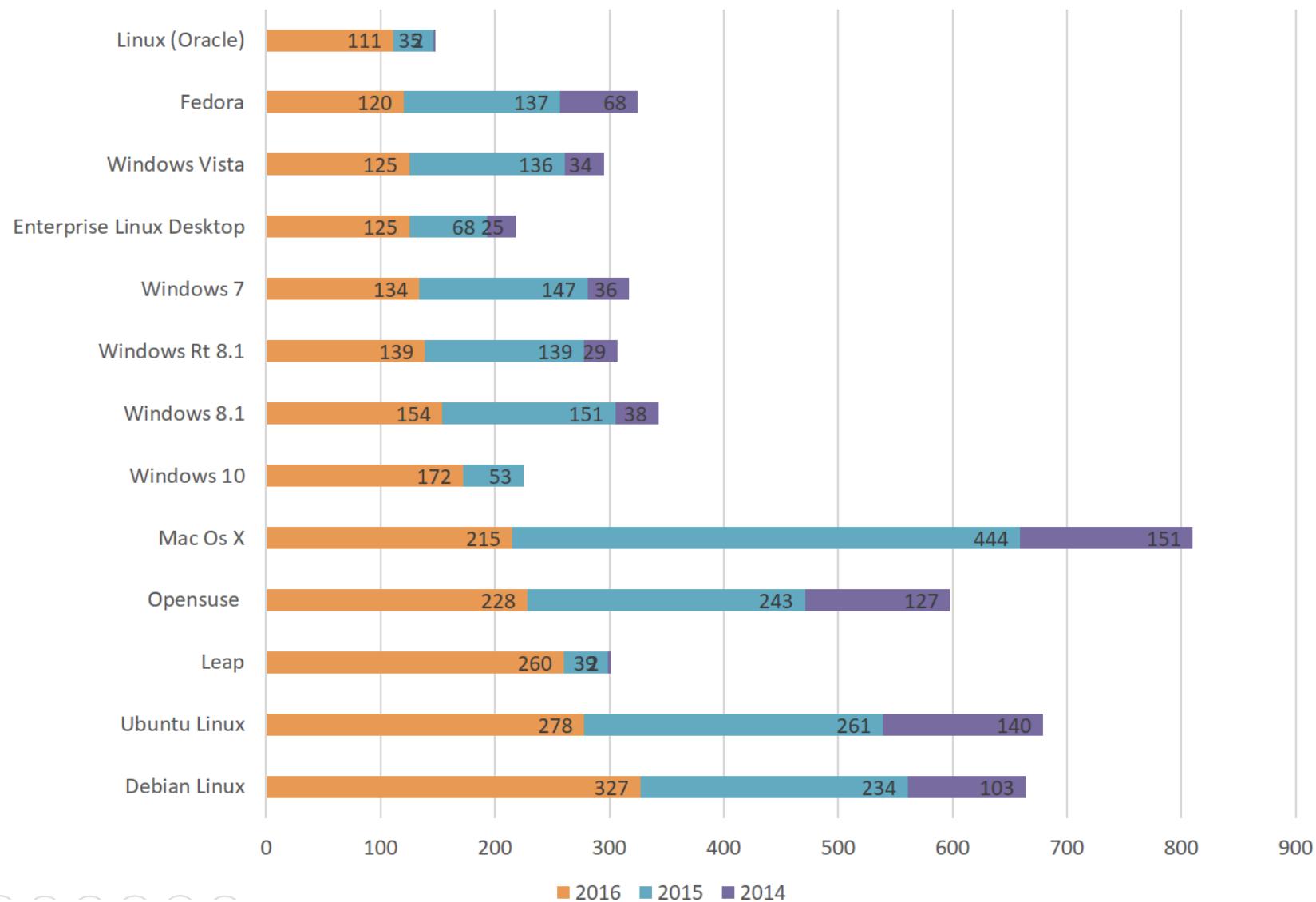


Reported Vulnerabilities stats

Vulnerabilities Reported by VulnDB¹

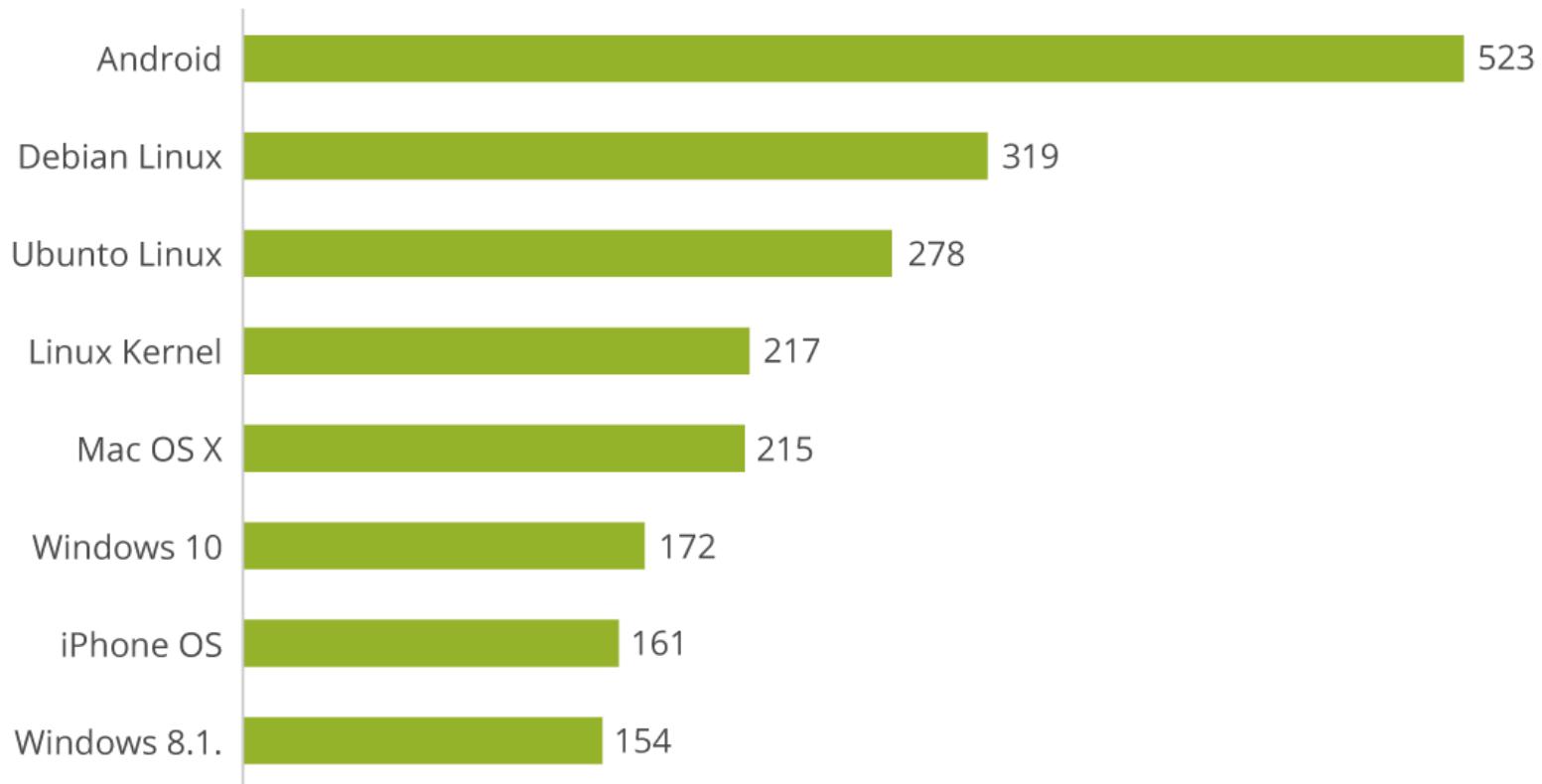


Top vulnerable operating systems in 2016



Android Is The Most Vulnerable Operating System

Number of vulnerabilities by operating system in 2016*



* Vulnerability defined as a mistake in software that can be directly used by a hacker to gain access to a system/network

CompSec and Economics



Question

Why do so many vulnerabilities exist in the first place?

Class brainstorm

Why does commercial security fail?

Claim: the link between “money” and security is still frequently **broken** today:

- Security is a public good
 - “private” incentives are weak
- Worse than “**market for lemons**”:
 - Not only the customer cannot see the difference between good security and bad

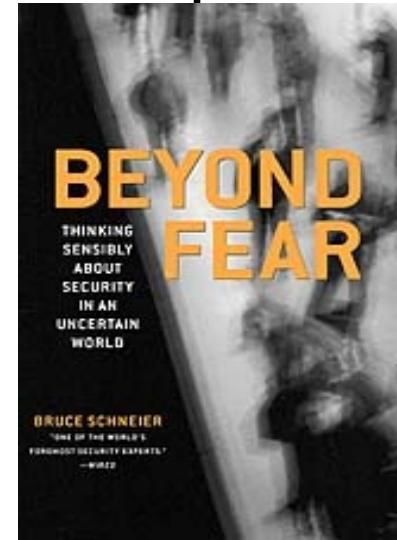
~~Frequently~~ Sometimes the manufacturer cannot either

The Very Nature of Security:



Bruce Schneier “Beyond Fear” book [2003], p.1:

Critical to any security decision is the notion of **security trade-offs**, meaning the **costs** – terms of money, convenience, comfort, freedoms, and so on - that inevitably attach themselves to any security system. People make security trade-offs naturally.



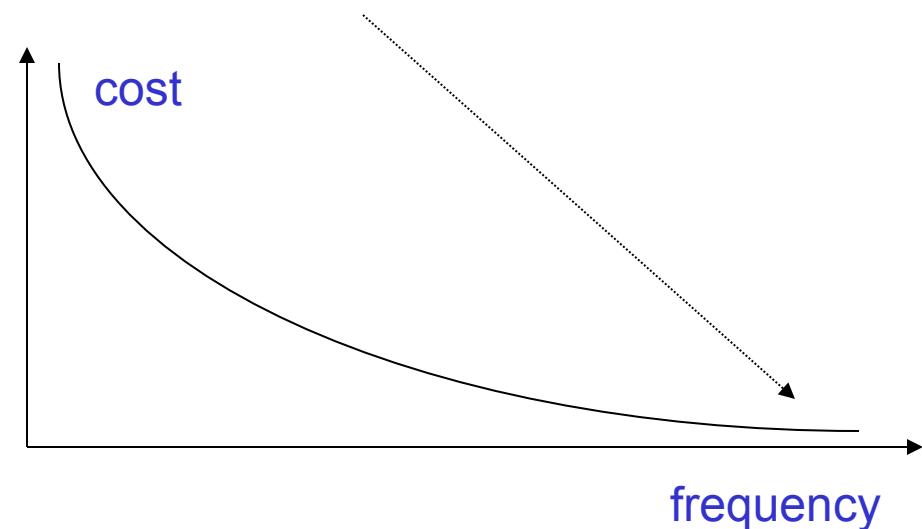
the Long Tail theory

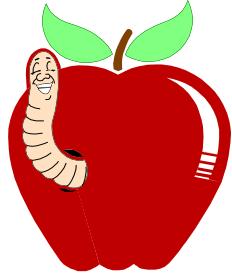
- One bank loses \$30M/year

- not more, risk management!
-

So they impose security measures on customers. Now:

- 300 M people lose 1 minute/day
= \$9000\$/Y





Why Things Happen?

Bugs... or don't care

- Programming with absence of security considerations
 - C/C++ is unsafe
 - Security/cryptography research developed with obsession with security. Both never met
- Economics/business:
 - customers do not see => do not care about security
 - usability: usage burden frustrates users

*Risk

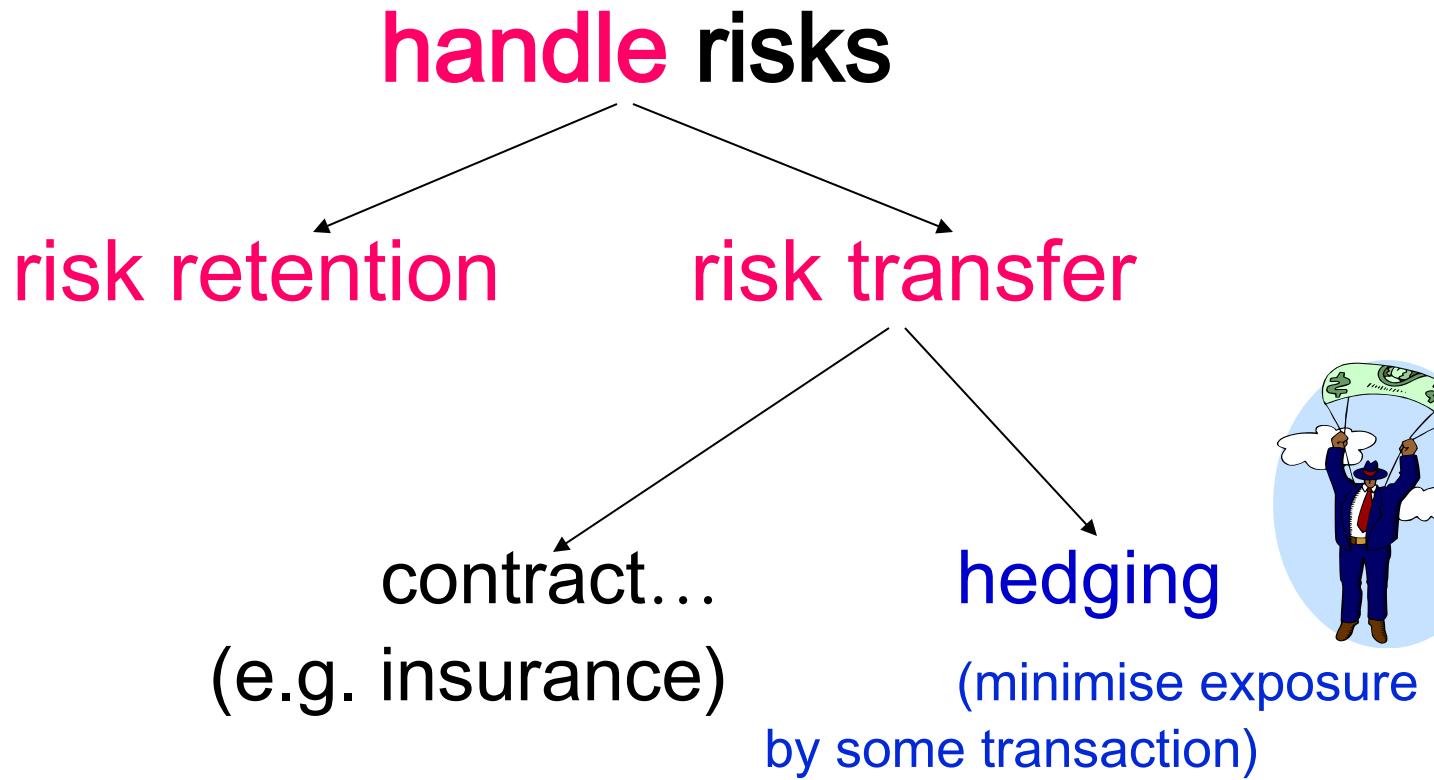


Risk Management = 1+2

A risk is the **potential** for something unwanted to happen (e.g., loss of C-I-A)

1. **Measuring or/and assessing risks**
2. **Developing strategies and solutions to manage risks:**
 - **reduce/avoid and**
 - **handle risks**

**Risk Management contd...



Residual Risk = def

what remains after defences are in place...

Defenders



3 Actions of Defenders

- Prevent
- Detect
- Respond



Types of Prevention

- Deter (discourage)
- Hinder (make harder)

Detection and Recovery

Detect

- Monitoring/logging
- Anomaly analysis

Recover

- Incident management
- Forensics
- Change procedures
- Install new technologies

Reasoning about security

Attack trees

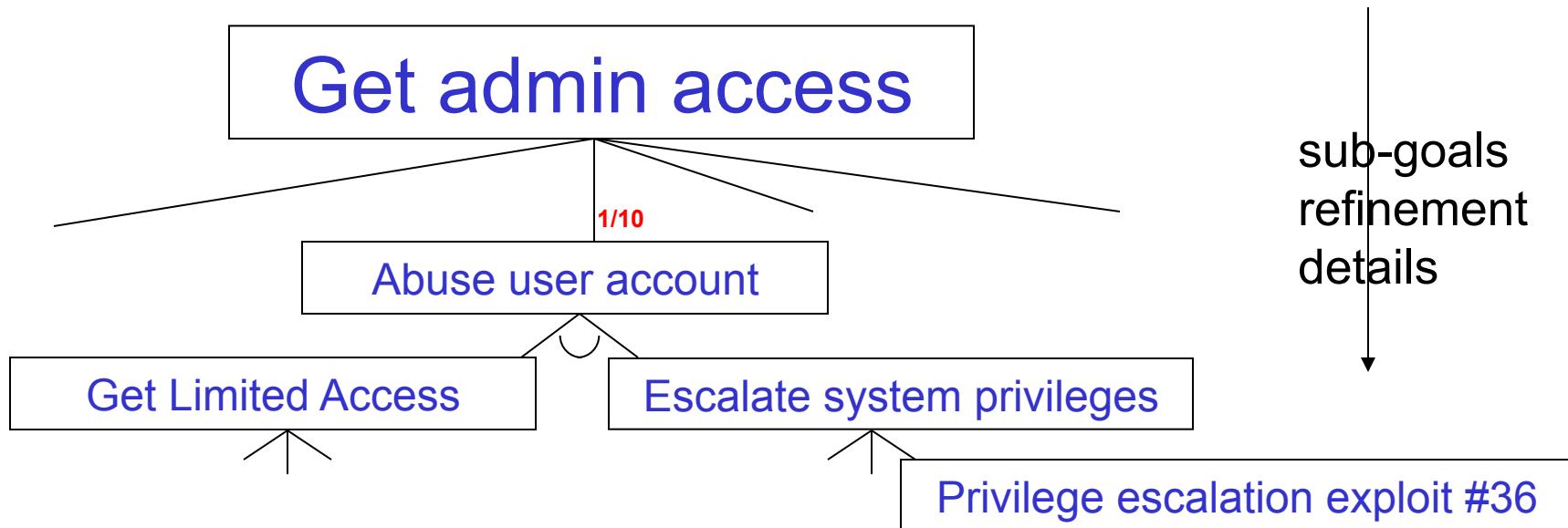
Attack tree

Formal analysis of all known attack avenues.

but what about unknown attacks?

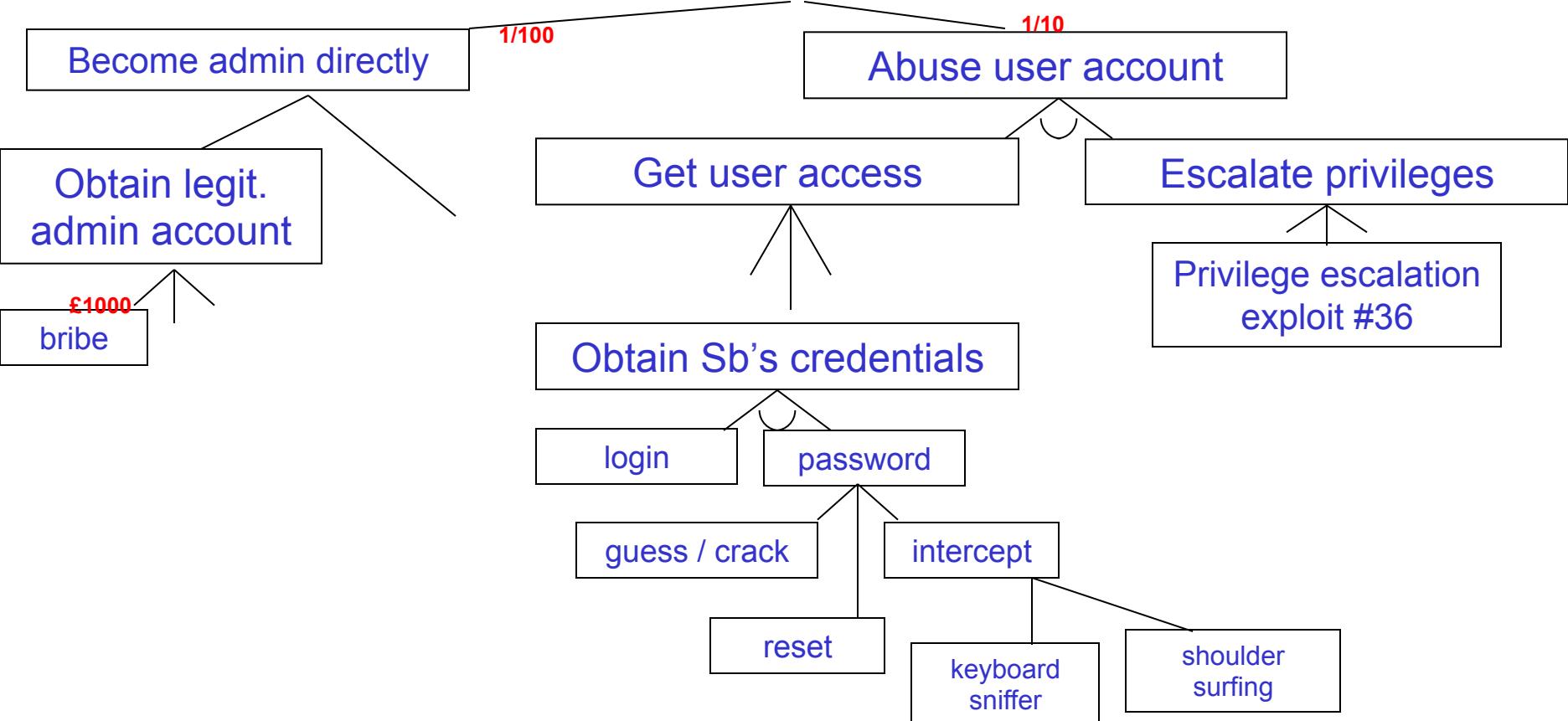
A tree with ~~OR~~ nodes and ~~AND~~ nodes.

nodes can be labeled with **probabilities** or **cost estimates**

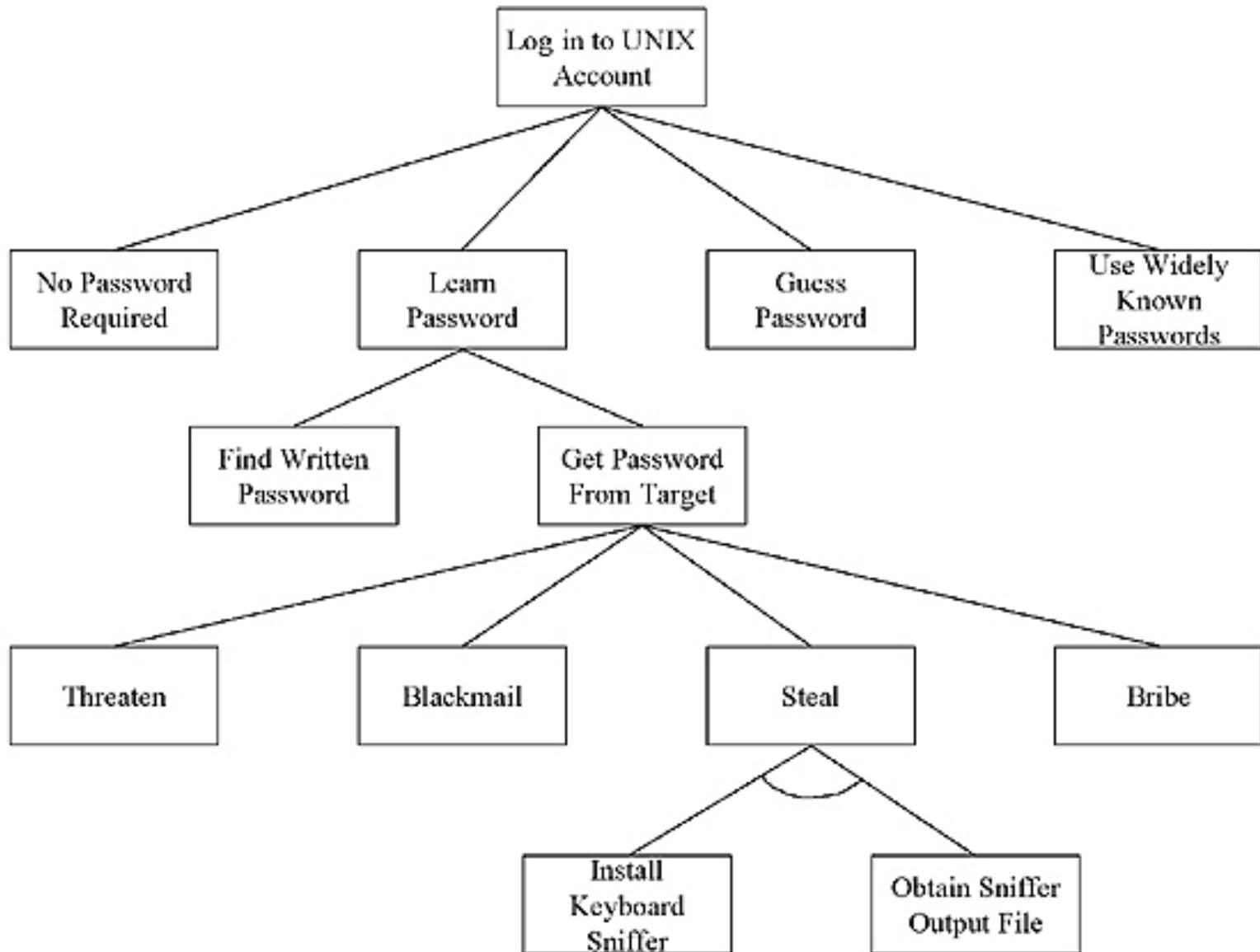


Expanded Example

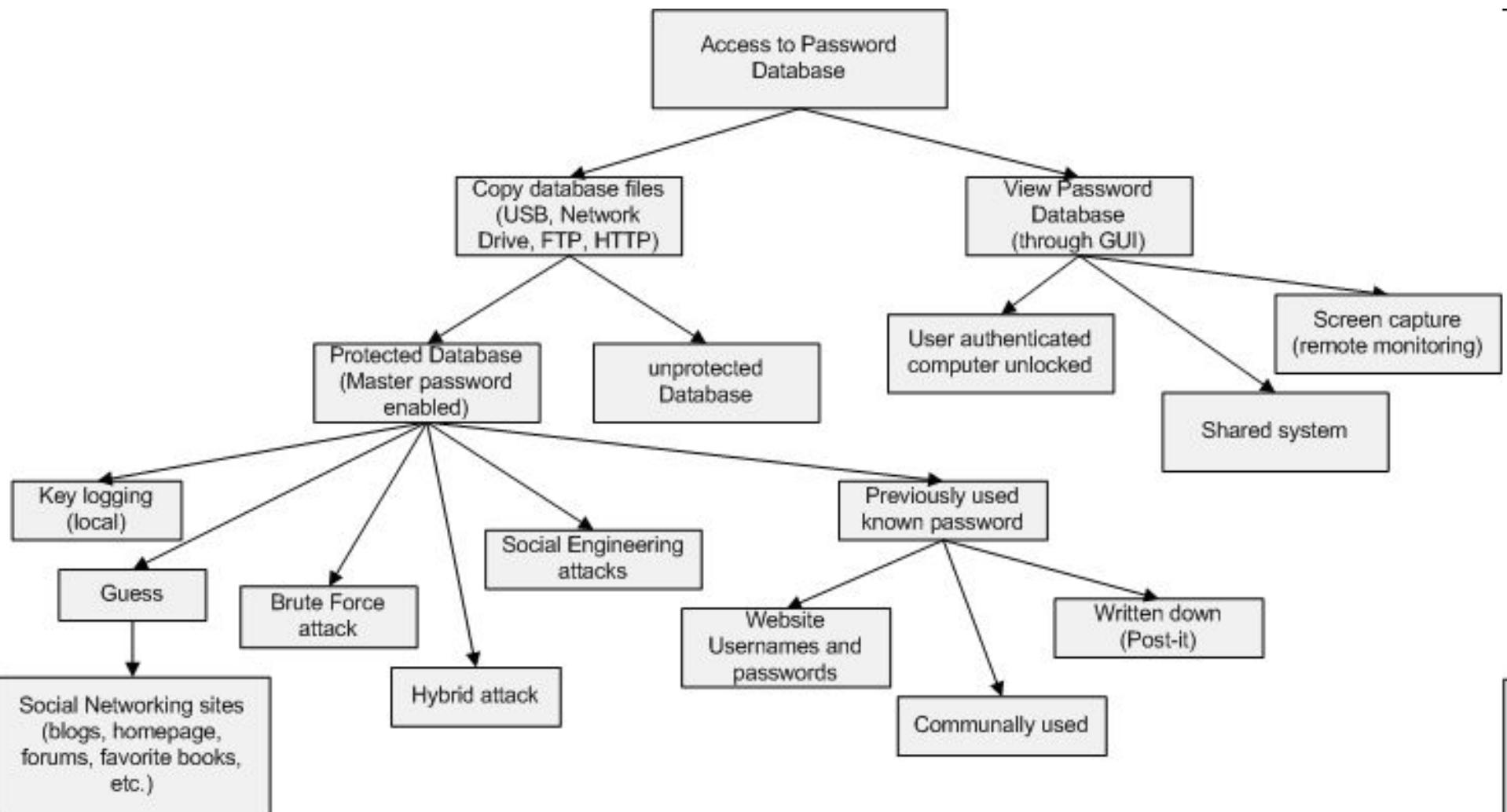
Get admin access



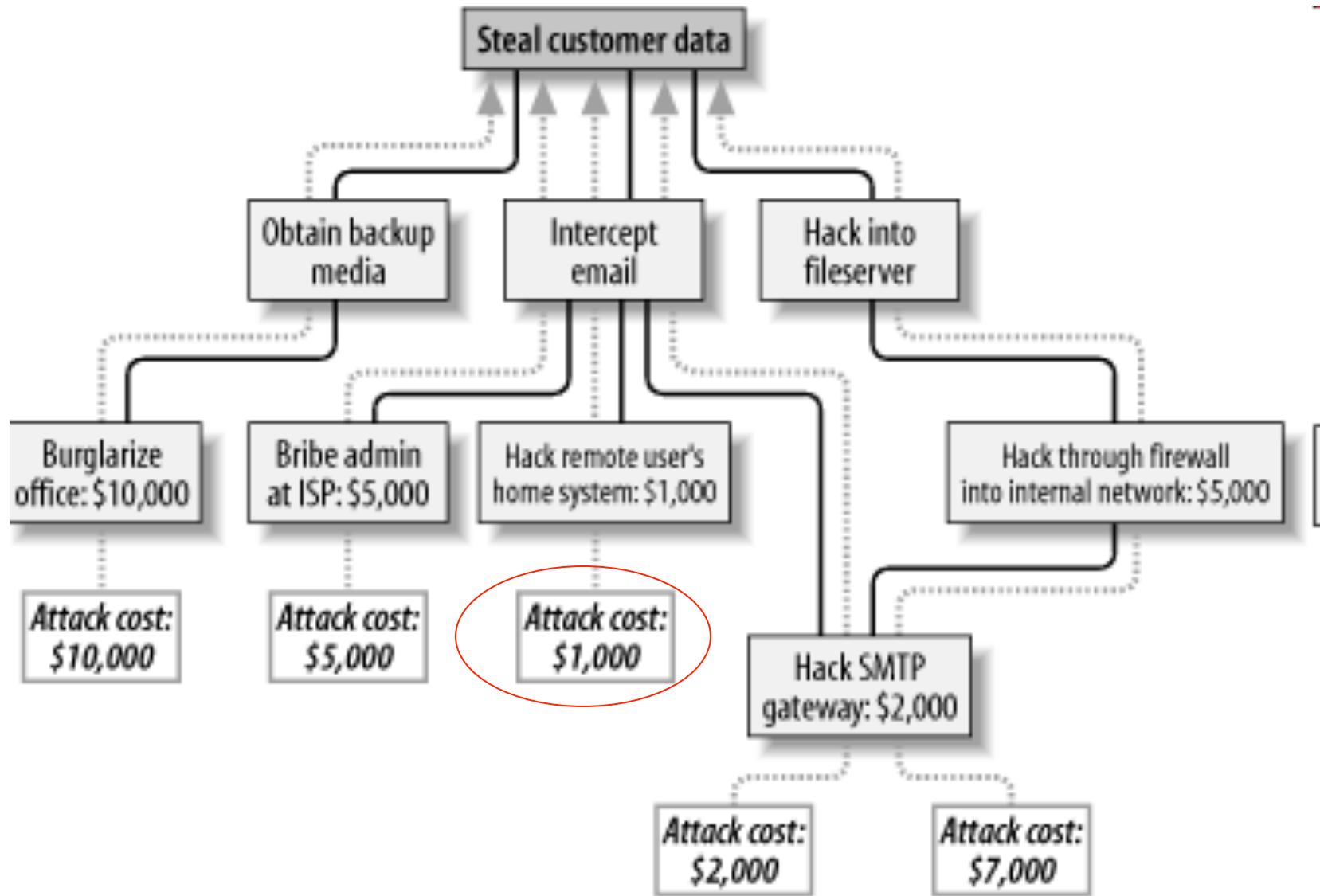
Unix Log In



Accessing Password Database



Stealing Data with Costs



Secrecy vs. Transparency

Open source vs. closed source

and security

Class brainstorm

Secrecy

Very frequently
an obvious
business decision



- Creates entry barriers for competitors
- But also defends against hackers

Kerckhoffs' principle (1883)

“The system must remain secure
should it fall in enemy hands ...”

“one ought to design systems under the assumption
that the enemy will immediately gain full familiarity
with them” reformulation by *Claude Shannon*



Kerckhoffs' principle (1883)

Most of the time: incorrectly understood

It doesn't mean that companies should disclose their designs

- Security when disclosed
- Better security when not disclosed

When is open source security good?

- Cryptography
 - AES, RSA, SHA256 etc, heavily tested, not yet broken
 - Compare closed-source crypto
 - Oyster card, car immobilisers, broken in months

Which model is better?

Open and closed security (if Kerckhoffs principle is followed) are **more or less equivalent**...

more or less as secure: opening the system **helps both the attackers and the defenders**

Ross Anderson: **Open and Closed Systems are Equivalent** (that is, in an ideal world). In Perspectives on Free and Open Source Software, MIT Press 2005, pp. 127-142

Ethics

or should Karate classes be legal?

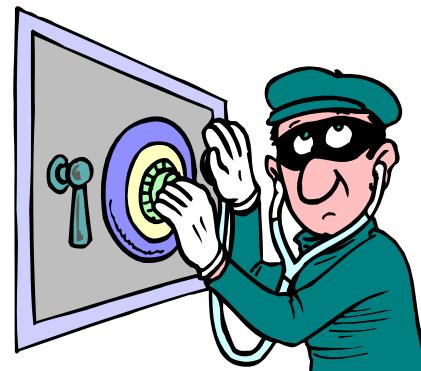


Key Question:

Is actively researching serious security vulnerabilities socially desirable?

- Of Course Yes!

...will tell you every professional hacker
and every academic code-breaker...



Bruce Schneier [14 May 2008]:

Problem: A hacker who discovers one [attack] can sell it on the black market, blackmail the vendor with disclosure, or simply publish it **without regard to the consequences**



Q: [...] is it ethical to research new vulnerabilities?

A: Unequivocally, **yes**. [according to Schneier]

Because:

- Vulnerability research is vital because it **trains** our next generation of computer security experts

http://www.schneier.com/blog/archives/2008/05/the_ethics_of_v.html

Responsible disclosure

Researchers should disclose vulnerabilities to the system owners, and give them “reasonable time” to fix them

especially if

...these vulnerabilities are likely to be rediscovered

Cf. E. Rescorla. “[Is finding security holes a good idea?](#)” In 3rd Workshop on the Economics of Information Security (2004)



Some principles of secure design

Designing Secure Systems 2017/18

David Galindo

Based on slides by Mark Ryan

Caveat: No magic formulas...

We have no silver bullet

On the contrary:

Security is about trade-offs

Conflicting engineering criteria....

Conflicting requirements...

Overcoming human,
technology and market deficiencies

12 Principles

These principles draw on the ideas of **simplicity** and **restriction**:

- **Simplicity** makes designs and mechanisms **easy to understand**. Importantly, less can go wrong with simple designs.
- **Restriction minimizes the power of an entity**: it can access only information it needs; it can communicate with other entities only when necessary, and in as few (and narrow) ways as possible

Design principles for protection mechanisms

[Saltzer and Schroeder 1975]

Jerome H. Saltzer, Michael D. Schroeder.
The protection of information in computer systems.
Proceedings of the IEEE 63(9): 1278-1308 (1975)

12 Principles

- 1. Secure the weakest link
- 2. Defence in depth
- 3. Fail secure
- 4. Grant least privilege
- 5. Economise mechanism
- 6. Authenticate requests
- 7. Control access
- 8. Assume secrets not safe
- 9. Make security usable
- 10. Promote privacy
- 11. Audit and monitor
- 12. Proportionality principle

1. Secure the weakest link

- Security practitioners often point out that **security is a chain**; and just as a chain is only as **strong as the weakest link**, a software security system is only as secure as its weakest component
- **Attackers go after the weakest point in a system**, and the weakest point is rarely a security feature or function. When it comes to secure design, make sure to consider the weakest link in your system and ensure that it is secure enough



Gene Spafford's story

- Imagine you are charged with transporting some gold securely from one homeless guy who lives in a park bench (we'll call him **Linux**) to another homeless woman who lives across town on a steam grate (we'll call her **Android**). You hire an armoured truck to transport the gold. The name of the transport company is "**Applied Crypto, Inc.**" Now imagine you're an attacker who is supposed to steal the gold.
- Would you attack the Applied Crypto truck, Linux the homeless guy, or Android the homeless woman?



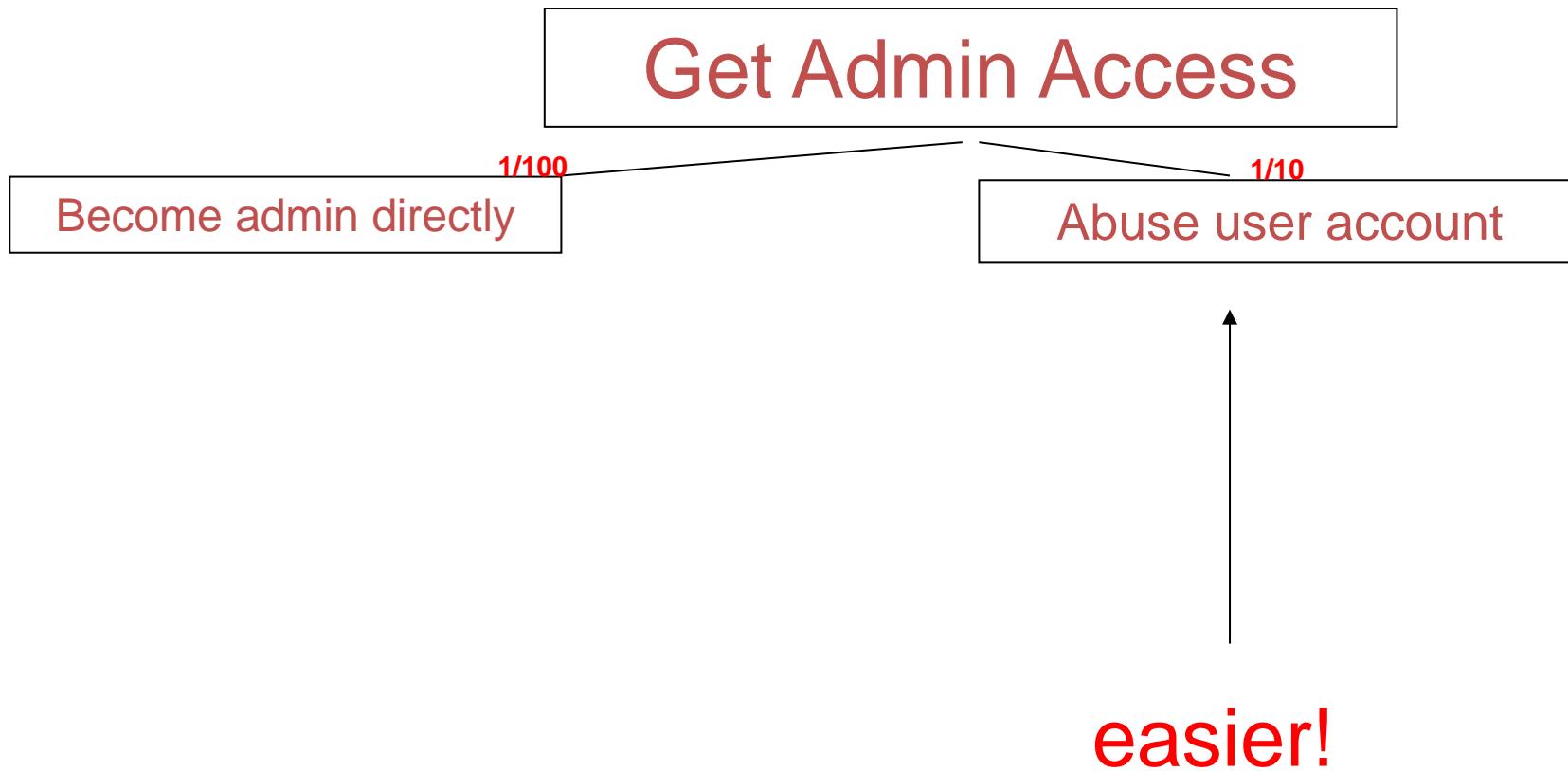
```
8ZRvQrB+H9j0==  
robots.stanford.edu,171.64.68.174 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAA  
AIvEas7W/tJB1IB1mqw7cUYgJnZjyBw80k95c/rKcNVB0TK3lgz6oIhAipBuHP+iQ  
2muCPaaRT60dqKyf5R0KB23xf8C9yIOMoAAvTjIPfzZghkevC08E+ZlOM8Q8vnxTh7  
Sgq0xns/mj9LRSZXYziD0NU3gzG5/6qu8BgWVZ297Gc=  
robot.cc ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAIEAs7W/tJB1IB1mqw7cUYgJn  
zJiyBw80k95c/rKcNVB0TK3lgz6oIhAipBuHP+iQ2muCPaaRT60dqKyf5R0KB23xf  
8C9yIOMoAAvTjIPfzZghkevC08E+ZlOM8Q8vnxTh7Sgq0xns/mj9LRSZXYziD0NU3g  
zG5/6qu8BgWVZ297Gc=  
udacity.com,173.255.251.252 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAwI  
KcmOFxdiiVI2Pd8TViPoXUWr0sr47rl3dgEbVsPRISE1wLMWrf4az5n07xV63xT  
Biehf5RSGkCWIXhtfsRt+876TPuLAjfi1PK0x2PjrgL3AZWQI0ul49tSX2S1fr  
GZzF7qzurpVximd78a2+lQHu4VSjMFZwYLCr0L4cb0l0T1Ny3kSNmfSCpDKicaVjOn  
TliliqhiwOzdncyh6sLcqHL20bcn0Nxt+hJyBx66EvPvxK9g090k4rjn6Z9l9Tnx  
MdZUccct0ZDPDSC912Y5YYh4Q+T599vlnNgn1+kfcw3Gkk1pAhYRVFr@WpGuG3bGzo  
025bxWK4Iw==  
> ssh -l dave udacity.com  
dave@udacity.com's password:
```



.Weakest Link



Security like a chain:





The human (caveat and precious resource)

- Humans are often considered the “weakest link” in the security chain
- They ignore security policies and stick passwords to their computer monitors
- They can be coerced, blackmailed, “socially engineered” (=tricked)
- Even security experts are vulnerable to phishing
- But, “the user is not the enemy”...



Are people the weakest link in computer security?

“Cybersecurity professionals have spent the last 25 years saying **people are the weakest link**. That’s stupid! They cannot possibly be the weakest link – they are the people that create the value at these organisations”

“What that tells me is that the **technical systems** we’ve built **are not built for people**. Techies build systems for techies, they don’t build technical systems for normal people”

Ian Levy, NCSC Director

The Guardian, 22 Sept 2017

2. Defence in depth (aka “Castle Approach”)

- The idea behind the “defence in depth” approach is to defend a system against any particular attack using several independent methods
- It is a layering tactic, conceived by the US National Security Agency as a comprehensive approach to information and electronic security

Redundancy and layering

- Examples: Don't count on your firewall to block all malicious traffic; use an intrusion detection system as well. If you are designing an application, prevent single points of failure with security redundancies and layers of defence
- The idea behind defence in depth is to manage risk with diverse defensive strategies, so that if one layer of defence turns out to be inadequate, another layer of defence will hopefully prevent a full breach

Defence in depth: some mechanisms

- Anti virus software
- Authentication and password security
- Biometrics
- Demilitarized zones (DMZ)
- Encryption
- Firewalls (hardware or software)
- Hashing passwords (secure password management)
- Intrusion detection systems (IDS)
- Logging and auditing
- Multi-factor authentication
- Vulnerability scanners
- Physical security (e.g. deadbolt locks)
- Timed access control
- Internet Security Awareness Training
- Virtual private network (VPN)
- Sandboxing
- Intrusion Protection System (IPS)

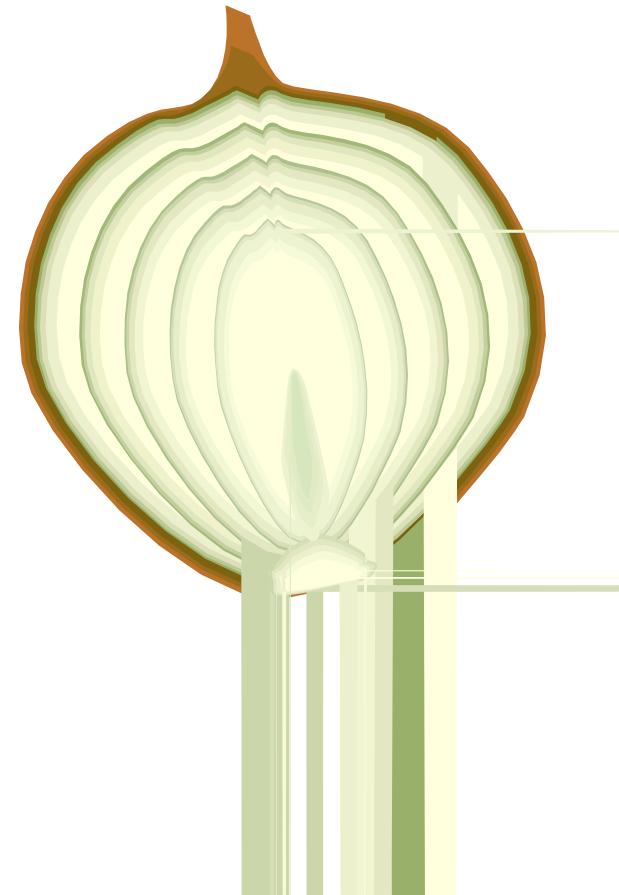
Military: Defence in Depth



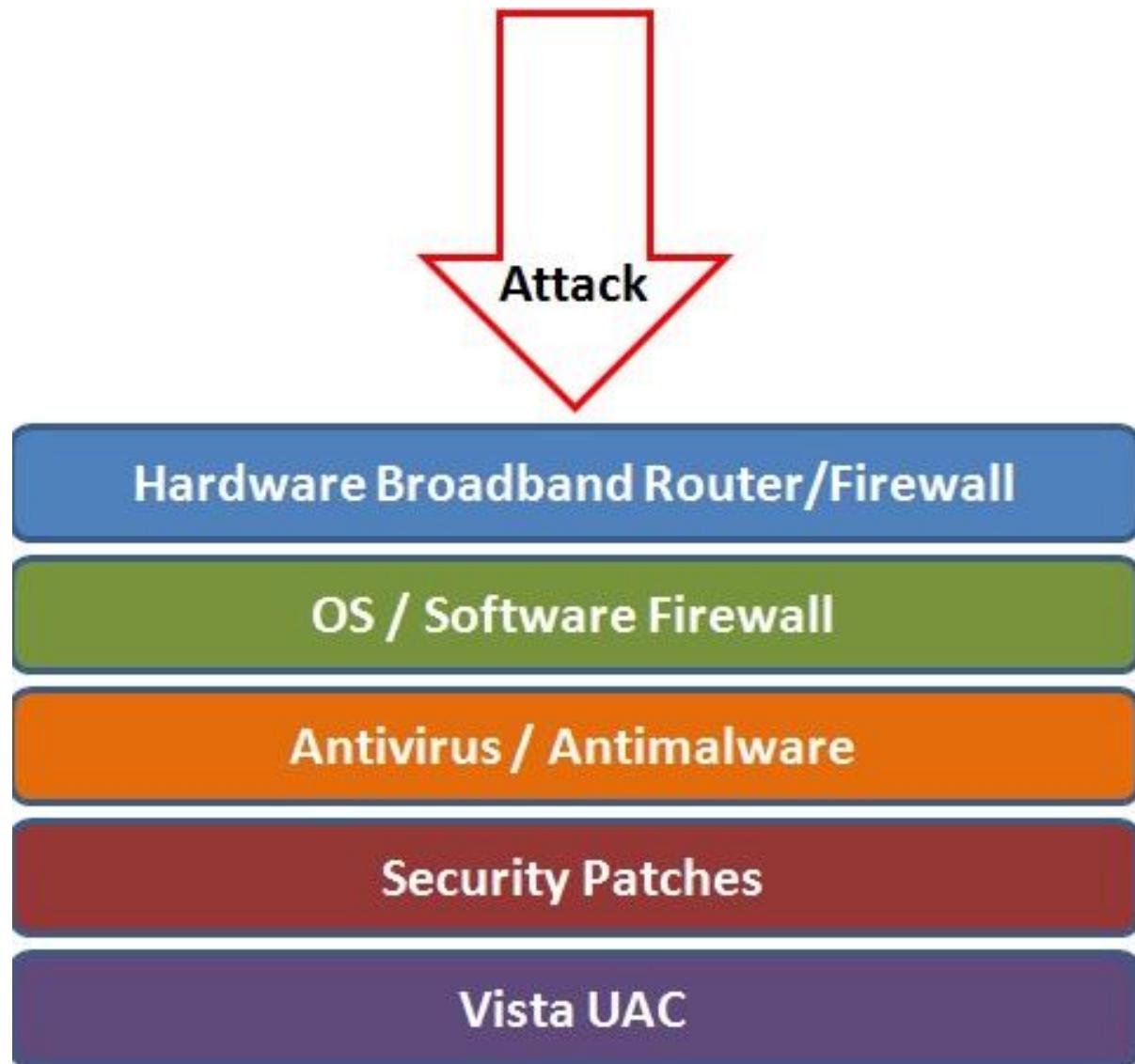
Layers

Computer systems have multiple layers, e.g.

- HW components
- Chipset/MB
- OS
- TCP/IP stack
- HTTP application
- Secure http layer
- Java script
- User/smart card interface



Example



Brainstorming defence in depth

- Helps to consider these three aspects:
 - People
 - Technology
 - Operations
- Example questions:
 - Attacker may get my users' passwords
 - Attacker may cause his own code to run on my servers

Attacker may cause his own code to run on my servers

Defence in depth:

- P: Education against social engineering and phishing
- P: Education on granting less privilege principle
- T: Sandboxing
- T: Access control
- T: Put additional security if not from familiar IP addr
- T: Use an Intrusion Detection System
- O: Set up Penetration Testing team
- O: Set up Software Secure Development Lifecycle

Attacker may get my users' passwords

Defence in depth:

- P: Education against social engineering and phishing
- P: Education about choosing good passwords
- T: Defend pw file using firewall
- T: Hash/encrypt/... pw file
- T: Put additional security if not from familiar IP addr
- T: Use one-time passwords
- O: Log guessing attempts
- O: Rate-limit guesses

Chains vs. Layers

3. Fail secure

- A fail-secure system is one that, in **the event of** a specific type of **failure**, responds in a way such that **access or data are denied**
 - Related: a **fail-safe system**, in the event of failure, causes no harm, or at least a minimum of harm, to other systems or to personnel
- Fail-secure and fail-safe may suggest **different outcomes**
 - For example, if a building catches fire, fail-safe systems would unlock doors to ensure quick escape and allow firefighters inside, while fail-secure would lock doors to prevent unauthorized access to the building

Fail secure

- Default is to deny access
- Programmers should **check** return values for **exceptions/failures**
- Base access decisions on **permission** rather than exclusion. This principle means that the default situation is lack of access, and the protection scheme identifies conditions under which access is permitted
 - The alternative, in which mechanisms attempt to identify conditions under which access should be refused, presents the wrong psychological base for secure system design

Secure-fail programming

- Look at the following (pseudo) code and see whether you can work out the security flaw:
- ```
DWORD dwRet = IsAccessAllowed(...);
if (dwRet == ERROR_ACCESS_DENIED) {
 // Security check failed
 // Inform user that access is denied
} else {
 // Security check OK
 // Perform task
}
```

- Secure-fail version:
- ```
DWORD dwRet = IsAccessAllowed(...);
if (dwRet == NO_ERROR) {
    // Security check OK
    // Perform task
} else {
    // Security check failed
    // Inform user that access is denied
}
```

Secure-fail programming

- Look at the following (pseudo) code and see whether you can work out the security flaw:

```
isAdmin = true;  
try {  
    codewhichMayFail();  
    isAdmin = isUserInRole( "Administrator" );  
}  
catch (Exception ex)  
{  
    log.write(ex.toString());  
}
```

- Secure-fail version:

```
isAdmin = false;  
try {  
    codewhichMayFail();  
    isAdmin = isUserInrole( "Administrator" );  
}  
catch (Exception ex)  
{  
    log.write(ex.toString());  
}
```

Fail-insecure designs

- Interrupted boot of OS drops user into root shell
- Browser unable to validate HTTPS certificate:
 - Tells user, allows user to click-through to proceed anyway
- Point-of-sale terminal unable to contact card-issuing bank:
 - Allows the transaction if less than a certain threshold
 - Risk management overrides security principle

Why we often have fail-insecure

- Fail-insecure often introduced by desire to support legacy (insecure) versions.
 - E.g., TLS allows old clients to use weak crypto keys
- Sometimes introduced because it is in general easier to work with short “blacklist” than with long “whitelist”
 - List of known-phishing websites
 - List of known-malicious users
 - List of known...

4. Grant least privilege

The principle of least privilege

- also known as the principle of minimal privilege or the principle of least authority

requires that in a particular abstraction layer of a computing environment, **every module** (such as a process, a user, or a program, depending on the subject) **must be able to access only the information and resources that are necessary for its legitimate purpose**

Least privilege: rationale

- When code is limited in the system-wide actions it may perform, vulnerabilities in one application cannot be used to exploit the rest of the system.
 - For example, Microsoft states “Running in standard user mode gives customers increased protection against inadvertent system-level damage caused by malware, such as root kits, spyware, and undetected viruses”

Least privilege: example

- A user account gets only those privileges which are essential to that user's work
 - A backup user does not need to install software: hence, the backup user has rights only to run backup and backup-related applications. Any other privileges, such as installing new software, are blocked.

Least privilege: example

- In UNIX systems, root privileges are necessary to bind a program to a port number less than 1024.
 - For example, to run a mail server on port 25, the traditional SMTP port, a program needs the privileges of the root user
- Once set up, the program should relinquish its root privileges, not continue running as root
- A large problem with many e-mail and other servers is that they don't give up their root permissions once they grab the mail port (Sendmail is a classic example)

Why we often fail to assign least privilege

- It's an effort to figure out what the least privilege needed actually is
- The unaware (or lazy or overworked) designer or programmer will often assign the max privilege, because that's easy
- "If we don't run as admin, stuff breaks"
- It's very hard to design systems that don't have some root/sysadmin that has all the privileges
 - E.g., Google employees getting fired for reading users gmail

Related: separate privileges

- Design your system with many different privileges
 - “file access” is not one privilege, but many
 - “network access” translates to many privileges
 - reading vs writing

Separation of Privileges

Split system into pieces, each with limited privileges

Implementation in software engineering:

Have computer program fork into two processes:

- The main program drops privileges (e.g. dropping root under Unix)
- The smaller program keeps privileges in order to perform a certain task
- The two halves then communicate via a socket pair
-

Benefits:

- A successful attack against the larger program will gain minimal access.
–even though the pair of programs will perform privileged operations

Related: Segregation of Duties

Achieved by the **Principle of Functional Separation**

- Several people should cooperate
- Examples:
 - one developer should not work alone on a critical application
 - the tester should not be the same person as the developer
 - If two or more steps are required to perform a critical function, at least two different people should perform them, etc

This principle makes it very hard for one person to compromise the security, on purpose or inadvertently.

A particular case is the **Principle of Dual Control**

- Example 1: in the SWIFT banking data management system there are two security officers: **left security officer** and **right security officer**. Both must cooperate to allow certain operations
- Example 2: nuclear devices command
- Example 3: cryptographic secret sharing

5. Economise mechanism

- Complexity is the enemy of security
- It's just too easy to screw things up in a complicated system, both from a design perspective and from an implementation perspective

Economise mechanism: related

- Keep it simple, don't look silly!
 - A design principle noted by the U.S. Navy in 1960
- Minimalistic design
 - Developers may create user interfaces made to be as simple as possible by eliminating buttons and dialog boxes that may potentially confuse the user
- Worse is better
 - Quality does not necessarily increase with functionality. Preference given to software that is limited, but simple to use
- "You ain't gonna need it" (YAGNI)
 - A principle of *extreme programming*, says that a programmer should add functionality only when actually needed, rather than when you just foresee that you need them

Why systems get complex

- Desire for features; mission-creep
- New technologies, new possibilities
 - Stretches the designs we already have
 - Causes interfaces to get used in ways not intended
 - Causes new software to be layered on top of old software
- Desire for legacy compatibility
 - Desire to keep things interoperable

Why complexity leads to insecurity

- Cognitive overload
 - Designer can't keep all the possibilities in her head at once
 - Security analyst can't reason about all the access possibilities

6. Authenticate requests

Be reluctant to trust

- Assume that the **environment** where your system operates **is hostile**. Don't let just anyone call your API, don't let just anyone gain access to your secrets!
- When using a cloud component, put in some checks to make sure that it has not been spoofed or otherwise compromised
- **Anticipate attacks** such as: command-injection, cross-site scripting, and so on

A Few Words on Trust

Definitions:

- A trusted system is one that can break the security policy (paradoxical definition)
- Trustworthy system is one that won't fail us

An employee who is selling secrets is trusted and not trustworthy

Suppose Dropbox is trustworthy, and yet Alice encrypts her files before putting them there. Thus for Alice, Dropbox is trustworthy but not trusted

Trust vs Trustworthiness

Questions:

What is **trusted and trustworthy**?

What is **trusted and not trustworthy**?

What is **not trusted but is trustworthy**?

Be Trustworthy

Public scrutiny promotes trust

Commitment to security **is visible** in the long run

7. Control access

- Every access and every object should be checked, every time
- Make sure that if permissions change on the fly in your system, that access is systematically rechecked
- Don't cache permission results that grant authority or wield authority
- In a world where massively distributed systems are pervasive and machines with multiple processors are the norm, these principles can be tricky to implement

8. Assume secrets not safe

- Security is not obscurity, especially when it comes to secrets stored in your code. Assume that an attacker will find out about as much about your system as a power user, and more
- The attacker's toolkit includes decompilers, disassemblers, and any number of analysis tools. Expect them to be aimed at your system
- Finding a crypto key in binary code is easy. An entropy sweep can make it stick out

9. Make security usable

- If your security mechanisms are too annoying and painful, your users will go to great length to circumvent or avoid them
- Make sure that your security system is as secure as it needs to be, but no more
- If you affect usability too deeply, nobody will use your product, no matter how secure it is. Then it will be very secure, and very near useless

You can Even Be More Friendly

- Don't discourage users
 - Minimize the number of clicks
 - Minimize the number of things to remember
 - Make security easy to understand and self-explanatory
 - Security should NOT impact users that obey the rules
- Established defaults should be reasonable
 - People should not feel trapped
- It should be easy to
 - Restrict access
 - Give access
 - Personalize settings

10. Promote privacy

- Collect only the user *personally identifiable information* (PII) you need for your specific purpose
 - EU General Data Protection Regulation (GDPR) applies in UK from May 2018
 - data breaches can cost up to 4% of a company's yearly revenue
- Store it securely, limit access
- Delete it once your purpose is done
- Let the users own their data (only store encrypted data)

11. Audit and monitor

- Record what actions took place and who performed them
 - This contributes to both disaster recovery (business continuity) and accountability

12. Proportionality Principle

Reminder: security isn't the only thing we want.

Maximize security???

-VS-

Maximize “utility” (the benefits)
while limiting risk

- to an acceptable level
- within reasonable cost
- commensurate with attacker's resources

Twelve principles

- 1. Secure the weakest link
- 2. Defend in depth
- 3. Fail secure
- 4. Grant least privilege
- 5. Economise mechanism
- 6. Authenticate requests
- 7. Control access
- 8. Assume secrets not safe
- 9. Make security usable
- 10. Promote privacy
- 11. Audit and monitor
- 12. Proportionality principle

Remember the caveat:
There are no magic formulas...

We have no silver bullet

Security is about trade-offs
Conflicting engineering criteria...
Conflicting requirements...

The goal is to have enough security to thwart the attackers
we care about

Web site security: authentication, encryption, and public-key certificates

Designing Secure Systems

Mark D. Ryan and David Galindo
University of Birmingham

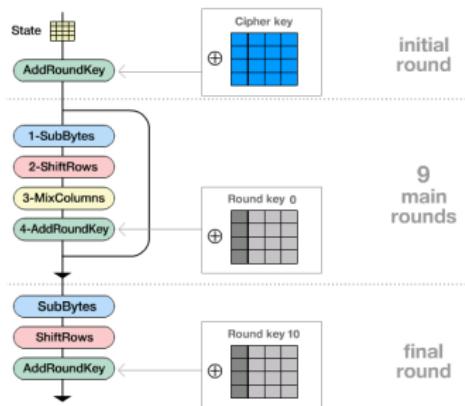
Contents

- 1) Cryptography
- 2) Authenticating websites and encrypting web traffic
- 3) The Certificate Authority model and its problems
 - The currently-used system
- 4) Certificate transparency
 - The being-rolled-out system
- 5) Certificate revocation

1. Cryptography

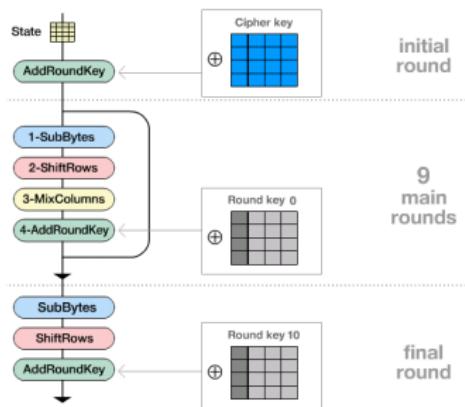
Symmetric key encryption (e.g., AES)

AES block cipher (a permutation substitution network)

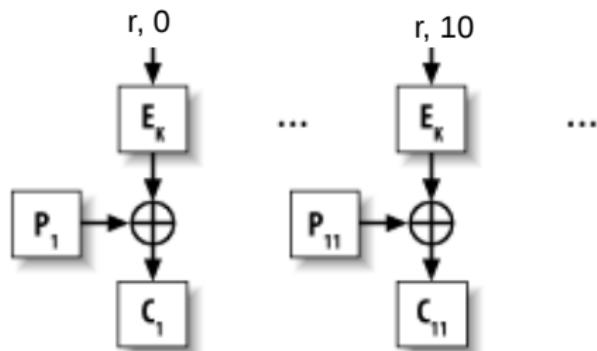


Symmetric key encryption (e.g., AES)

AES block cipher (a permutation substitution network)

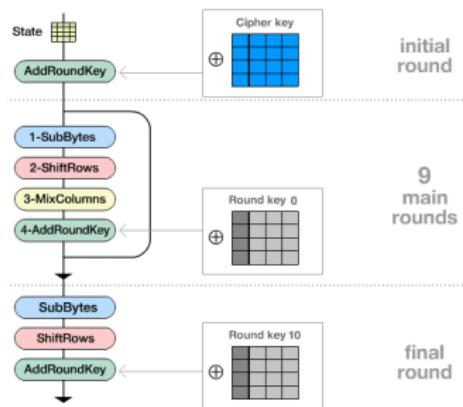


Counter-mode (a mode of operation for block ciphers)

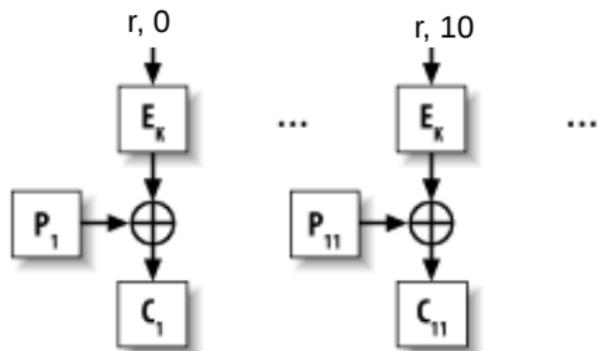


Symmetric key encryption (e.g., AES)

AES block cipher (a permutation substitution network)



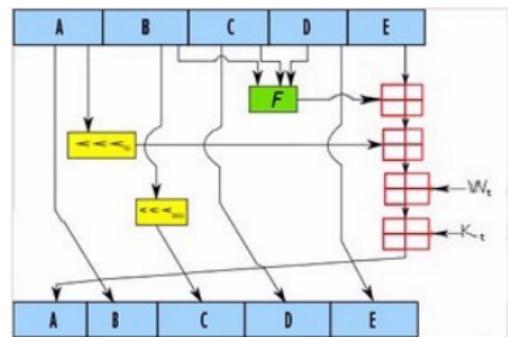
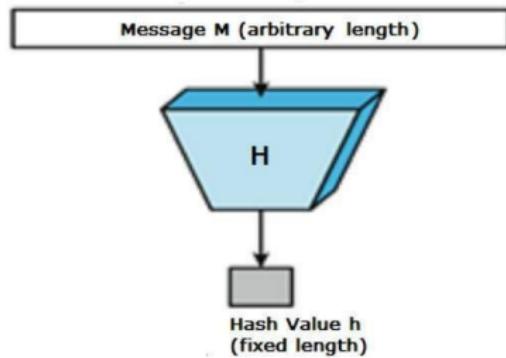
Counter-mode (a mode of operation for block ciphers)



$$\text{dec}(k, r, \text{enc}(k, r, m)) = m$$

$$\text{dec}(k, \text{enc}(k, m)) = m$$

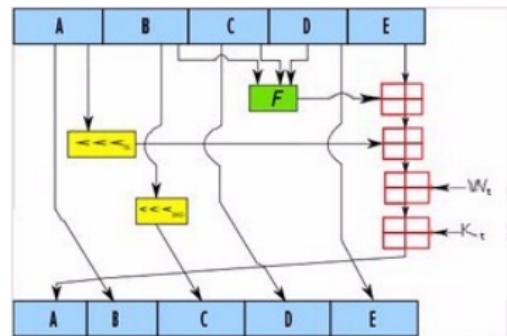
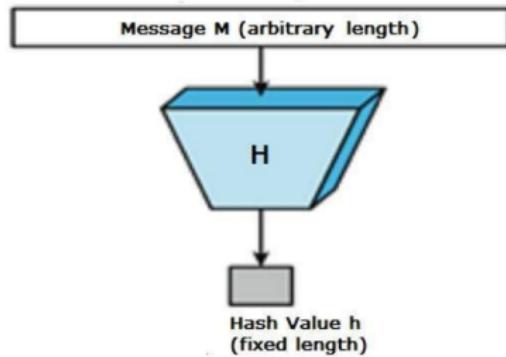
Hash functions (e.g., SHA2)



$H : \{\text{messages of any size}\} \rightarrow \{256\text{-bit messages}\}$
such that:

$H(x) = H(y)$ implies $x = y$ with huge probability

Hash functions (e.g., SHA2)



$H : \{\text{messages of any size}\} \rightarrow \{256\text{-bit messages}\}$
such that:

$H(x) = H(y)$ implies $x = y$ with huge probability

Public-key encryption (e.g., RSA-OAEP)

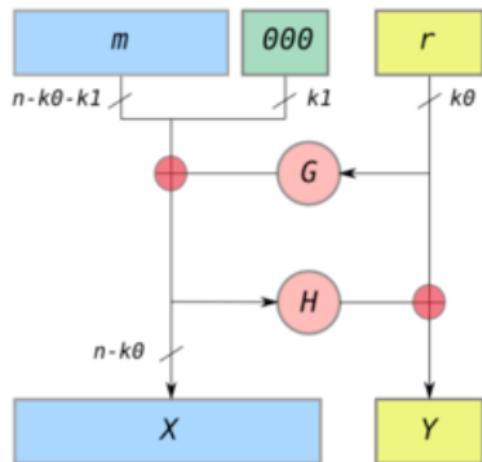
p, q prime

$$n = p * q$$

$$e * d = 1 \bmod (p-1)(q-1)$$

$$C = M^e \bmod n$$

$$M = C^d \bmod n$$



$$\text{dec}(\text{dk}(k), \text{enc}(\text{ek}(k), r, m)) = m$$

Hybrid encryption

- Unlike the case for symmetric key encryption, there are no “block modes” for public key encryption
 - Because public key encryption is too slow to do for very long messages.
- To encrypt a long message m using public key encryption with a public key pk :
 - Choose a random symmetric key, k
 - Encrypt m with k : $c1=enc(k, r1, m)$
 - Encrypt k with pk : $c2=enc(pk, r2, k)$
- Send $c1, r1, c2$.

Hybrid encryption

- Unlike the case for symmetric key encryption, there are no “block modes” for public key encryption
 - Because public key encryption is too slow to do for very long messages.
- To encrypt a long message m using public key encryption with a public key pk :
 - Choose a random symmetric key, k
 - Encrypt m with k : $c1=enc(k, r1, m)$
 - Encrypt k with pk : $c2=enc(pk, r2, k)$
 - Send $c1, r1, c2$.

Digital signatures (e.g., Schnorr)

- Initialization (security parameter k)
 p, q , two large primes such that

$$\begin{aligned}q &\mid (p-1) \\2^{k-1} &\leq q < 2^k\end{aligned}$$

g , element of \mathbb{Z}_p^* of order q

f , hash function

secret key $x \in \mathbb{Z}_q^*$

public key $y = g^{-x} \bmod p$

- Signature

- $- K \in \mathbb{Z}_q^*$
- $- r = g^K \bmod p$
- $- e = f(m, r)$
- $- s = K + xe \bmod q$
- $- \sigma_1 = r$ and $\sigma_2 = s$

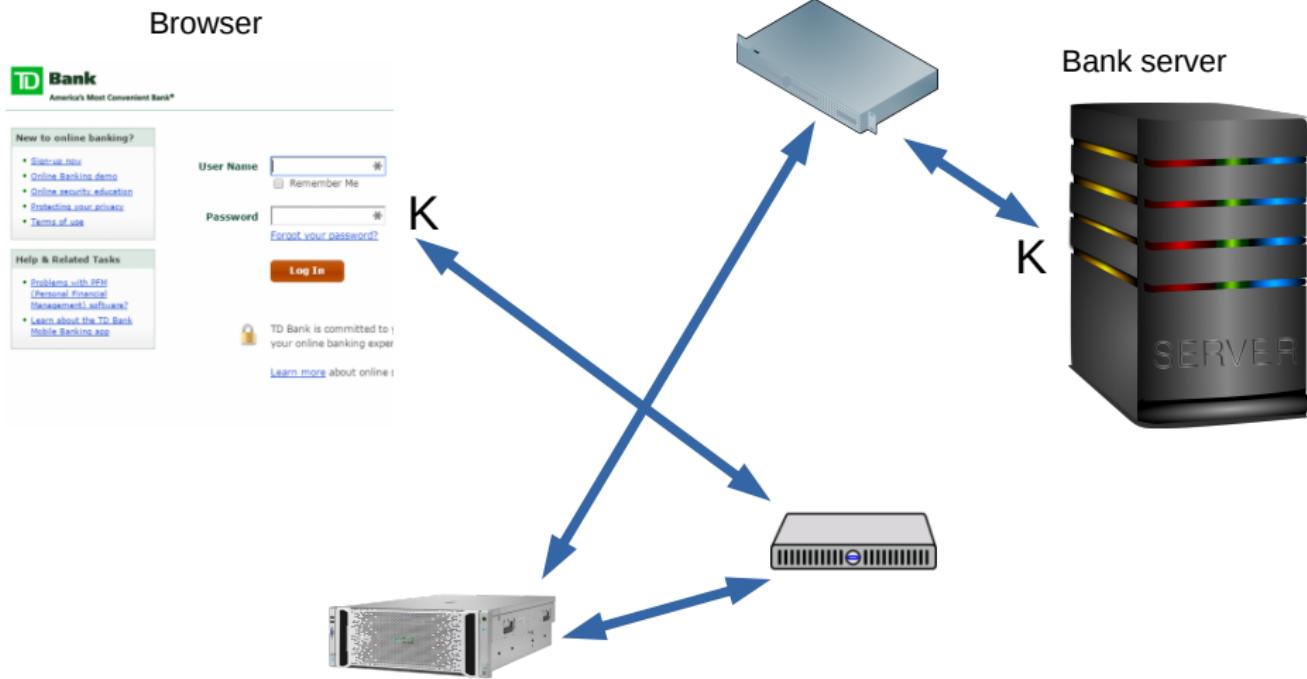
- Verification

- $- e \stackrel{?}{=} f(m, r)$
- $- r \stackrel{?}{=} g^s y^e \bmod p$

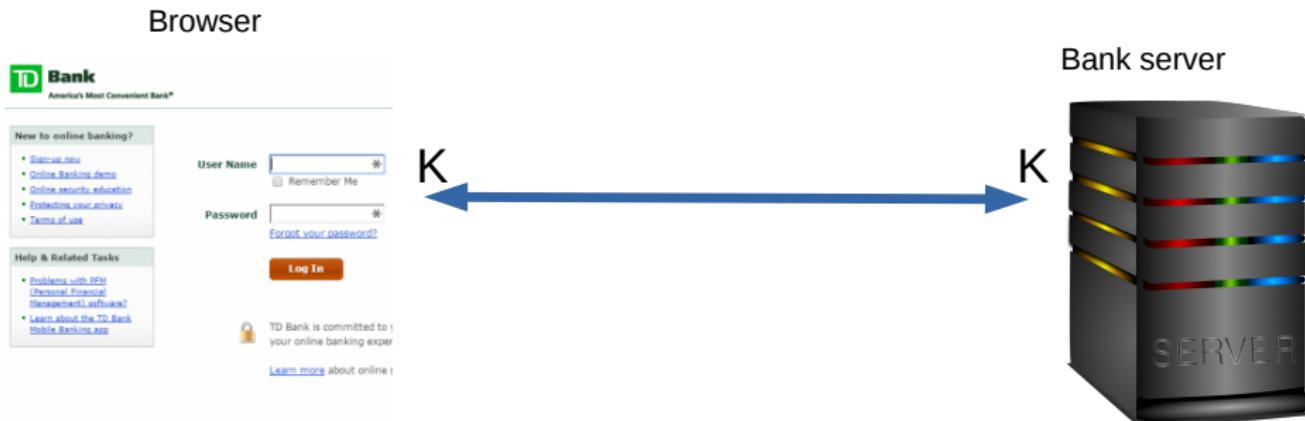
`checksig(vk(x), m, sign(sk(x), r, m)) = true`

2. Authenticating websites and encrypting web traffic

Encrypting browser-server communication: Encrypt with a shared key K



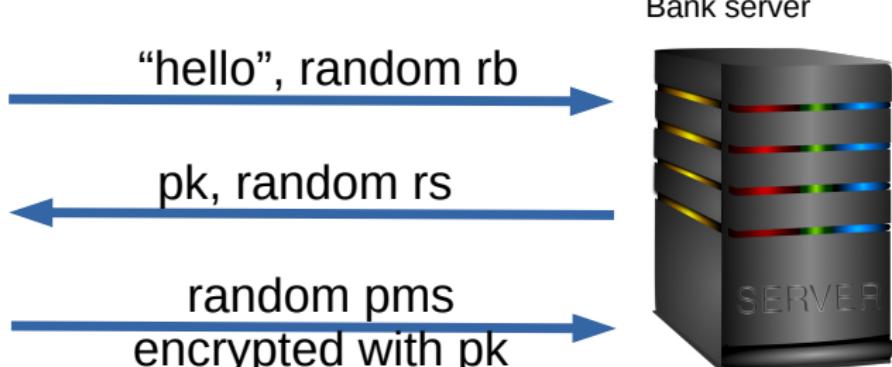
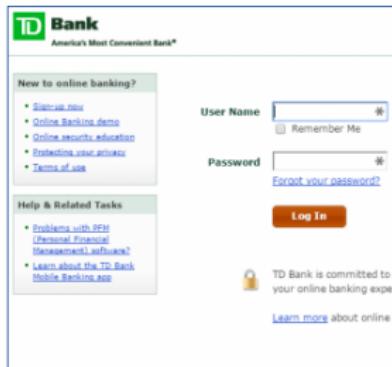
Encrypting browser-server communication



How to set up the shared key?

Let's say the server has a public key pk

Browser



$$K = f(pms, rb, rs)$$

$$K = f(pms, rb, rs)$$

That's TLS (in abstract form).

TLS

- In TLS, the session encryption key is established from the server's public key.
- Thus, the browser needs to obtain the server's public key.
- How to achieve that?
 - The server sends it!
 - But how can the browser *authenticate* it?

How can a browser verify that
a given public key belongs to a given website?

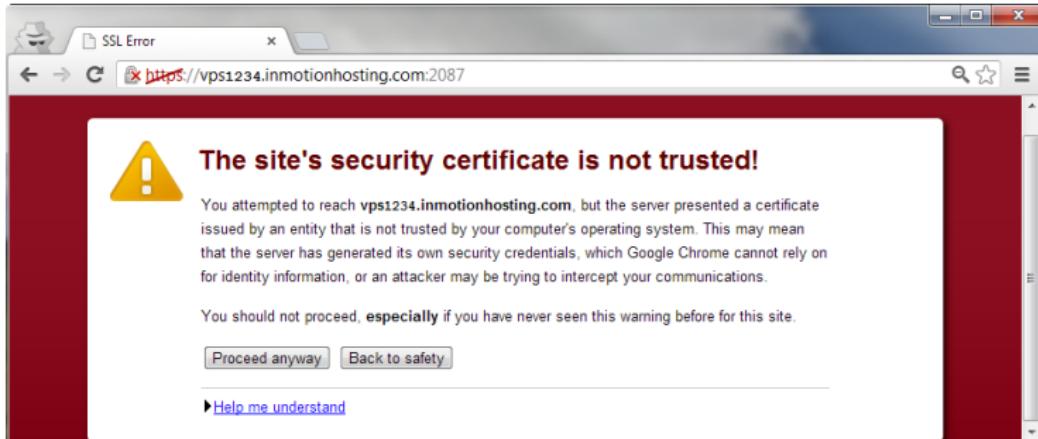
3. The Certificate Authority model

Certificate authorities

- A certificate authority (CA) is a trusted party that asserts that a given public key belongs to a given website.
- The CA signs a certificate
 $\text{Sign}(\text{sk}_{\text{CA}}, (\text{"HSBC Bank"}, \text{pk}))$
- The browser knows the verification key vk_{CA} needed to verify this signature.
 - It has lots of CA verification keys built in.

Problems with the CA model

- Usability:
Incomprehensible certificate warnings



- Insecurity:
Fake certificates... Any CA can sign an key

CA model broken

- **Comodo** reseller account broken into and used to fraudulently issue nine certificates, for Google, Microsoft, Yahoo, Mozilla, Skype. (March 2011)
- Wildcard certificate for Google issued by **DigiNotar**'s systems under the control of an attacker. (August 2011)
- MitM attack against Facebook in Syria, by click-through of self-signed certificates, allegedly inserted by Syrian Telecom Ministry. (May 2011)
- **Trustwave** issued a MitM certificate for a company, allowing it to issue valid certificates for any server; used for *data loss prevention* and monitoring of staff use of Gmail and hotmail. (Feb 2012)

How to avoid attacks like these?

How to ensure that a browser obtains the correct public key for a website?

Interlude: recalling the twelve principles

1. Secure the weakest link
2. Defend in depth
3. Fail secure
4. Grant least privilege
5. Economise mechanism
6. Authenticate requests
7. Control access
8. Assume secrets not safe
9. Make security usable
10. Promote privacy
11. Audit and monitor
12. Proportionality principle

Principles that seem relevant to the CA model

1. Secure the weakest link

Significant weak link: any CA can sign a cert. for any domain

2. Defend in depth

No layering of security

3. Fail secure

CA model *fails insecure*: invalid certs allow click-through

4. Grant least privilege

Not present. CAs have too much privilege

8. Assume secrets not safe.

CA signing keys may not be safe

9. Make security usable

Users don't understand certificate warnings

11. Audit and monitor

No opportunity for domain owners to audit what certificates are being used

4. Certificate transparency

Certificate transparency [Laurie, Kasper, Langley 2012]

Aim: ensure that whenever a CA signs a certificate, there is persistent **evidence** of this fact. A CA cannot sign certificates inadvertently/sneakily

Mechanism: An *append-only public log* is maintained, consisting of certificates issued by all CAs.

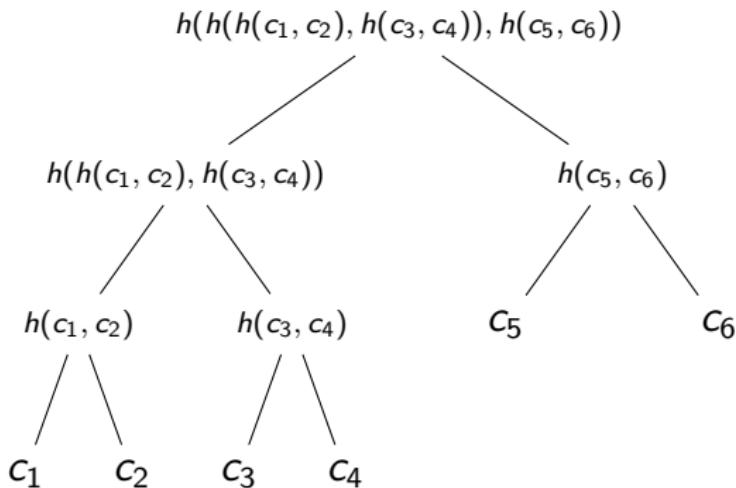
- A certificate is accepted only if it is included in the log
(The certificate comes with proof that it is included in the log)
- Domain owners can check that certificates about them in the log are really theirs

Users' client software checks the proofs, and also checks that the log is **append-only** and **linear**

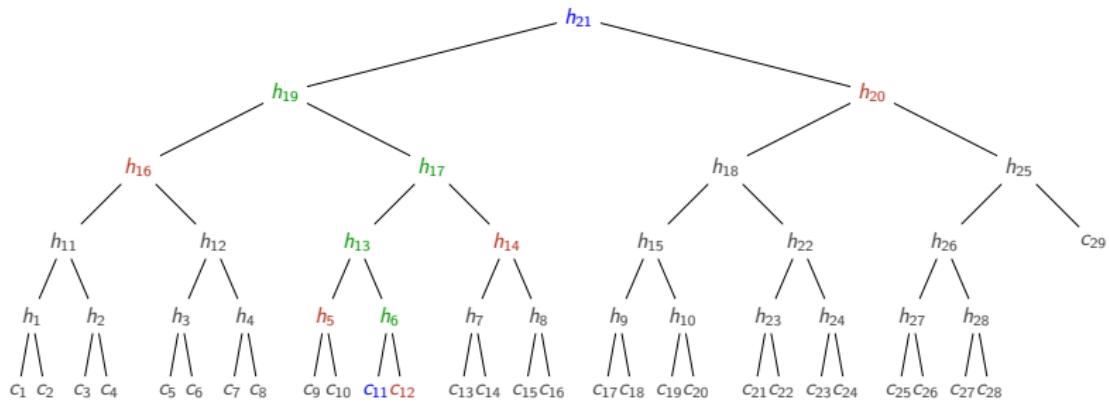
Status: IETF draft; RFC. January 2015: Google Chrome checks Google's CT logs for EV certificates.

Certificate transparency: append-only public log

A log storing certificates $c_1, c_2, c_3, c_4, c_5, c_6$:



$\text{prove_presence}(c_{11}, h_{21})$
 $= (c_{12}, h_5, h_{14}, h_{16}, h_{20})$



Algorithms on chronological trees (ChronTrees)

Algorithm	Complexity	Typical size 10^9 certif.
request_h()	$O(1)$	0.25 KB
prove_presence($h, cert$)	$O(\log n)$	2 KB

Certificate transparency: summary of operation

- A CT log maintains a list of all issued certificates.
- If a certificate is present in the log, it adds assurance as to the validity of the certificate, because . . .
- A domain owner can discover all the certificates about it that are present in the log.
- Browsers consult the log to verify that certificates presented to them are present in the log:
 - The log is organised as a Merkle tree, to facilitate *proofs of presence*.
- Browsers can also verify that the log is being maintained 'append only':
 - The Merkle tree organisation also facilitates *proofs of extension*.

CT is vulnerable to a *fork attack*



Browser sees a version of the log that contains an **invalid** certificate for the server.



Server sees a version of the log that contains only **valid** certificates for the server.

Browser and Server are seeing different views of the log. This kind of attack could be perpetrated by a powerful adversary, such as a government.

CT is vulnerable to a *fork attack*



Browser sees a version of the log that contains an **invalid** certificate for the server.



Server sees a version of the log that contains only **valid** certificates for the server.

Browser and Server are seeing different views of the log. This kind of attack could be perpetrated by a powerful adversary, such as a government.

Possible solution: *gossip protocols*

Revisiting the principles w.r.t. the CT model

1. Secure the weakest link

Significant weak link: *any* CA can sign a cert. for *any* domain. *Still true, but the domain owner now knows*

2. Defend in depth

CA and log maintainer provide layering

3. Fail secure

CT aim is to force hard-fail

4. Grant least privilege

CA privilege much reduced

8. Assume secrets not safe

Usefulness to attacker of CA signing key much reduced

9. Make security usable

Aim for users not to have to see cert warnings - hard fail instead

11. Audit and monitor

Certificate logging increases transparency

5. Certificate transparency: revocation

Key revocation

- If the secret key corresponding to a public key is accidentally divulged, or access to it is lost, then the key has to be *revoked*.
 - Revocation can be caused by lost/forgotten passwords, hacked accounts, lost computers or storage, etc.
- To support this, we need more than a *proof of presence*, i.e. a proof that the certificate was once issued.
 - We need *proof of currency* in the log: a proof that the certificate was issued, and has not subsequently been revoked.
 - Another way to see this is: we need *proof of absence* of a revocation in the log.
 $\text{current} = \text{present} \wedge \neg\text{revoked}$
- Technical challenge: extend CT to support efficient proofs of absence

Conclusions

- Authenticating websites is a surprisingly hard problem.
- The CA model that we have been using for 20+ years is no longer adequate
- Certificate transparency seems to be the most promising upcoming solution
- Related to the blockchain - also trendy :-)
- Certificate revocation can also be solved in this framework
- Still need to detail a solution for the fork attack

Device Security: Trusted boot and code signing

Mihai Ordean
Designing Secure Systems
University of Birmingham

Overview

- **Device security**
 - Is code on the device vulnerable to exploits ? (e.g. buffer overflows)
 - Is the code authenticated ? (i.e. has not been tampered with)
- **Data security**
 - Is the stored data is accessible to everyone? (e.g. encrypted)
 - Is the stored data authenticated?
- **Metadata security**
 - What does metadata reveal about data?
 - Can we tamper the metadata?
- **Protocol security**
 - Is data in transit visible?
 - Can data in transit be tampered with?

Overview

- Device security
 - Physical device security (e.g. side-channel attacks vulnerabilities)
 - **Firmware/OS security (e.g. bootloader, kernel)**
 - **Application security (e.g. code signing, sandboxing)**

Introduction

Security challenges

Protect devices against:

- Malicious applications
- Rootkits

Malicious applications

Malware distributed using OS specific applications, designed to exploit the operating system vulnerability's.

Issues:

- High success rate because they masquerade as useful applications
- Are often used as a means to install more dangerous malware: backdoors, rootkits

Rootkits

Code designed to enable access to a computer or areas of its software that would not otherwise be allowed.

Issues:

- Install themselves with highest privileges
- Prevent detection/removal with anti-malware tools

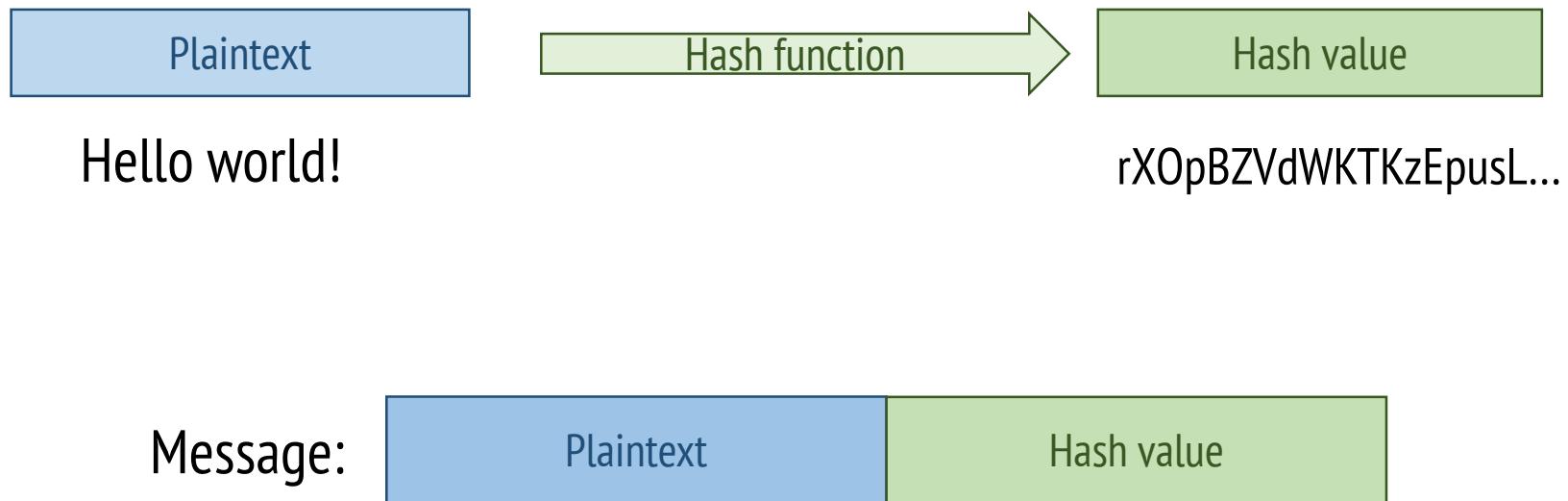
Mitigations

Defence in depth:

- Secure the boot process
- Secure the user space
- Secure the distribution channels

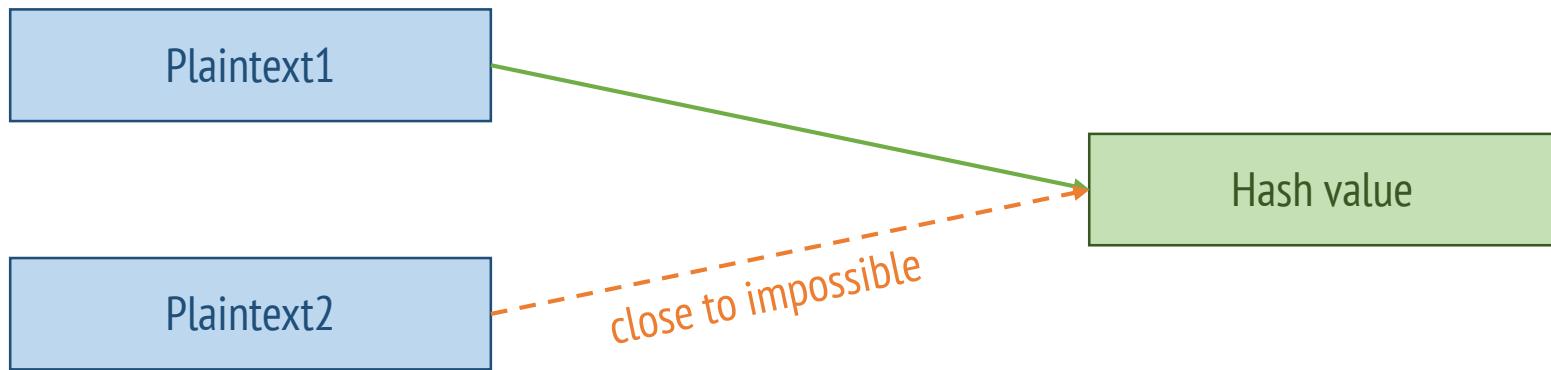
Hash functions and signatures

Hash functions



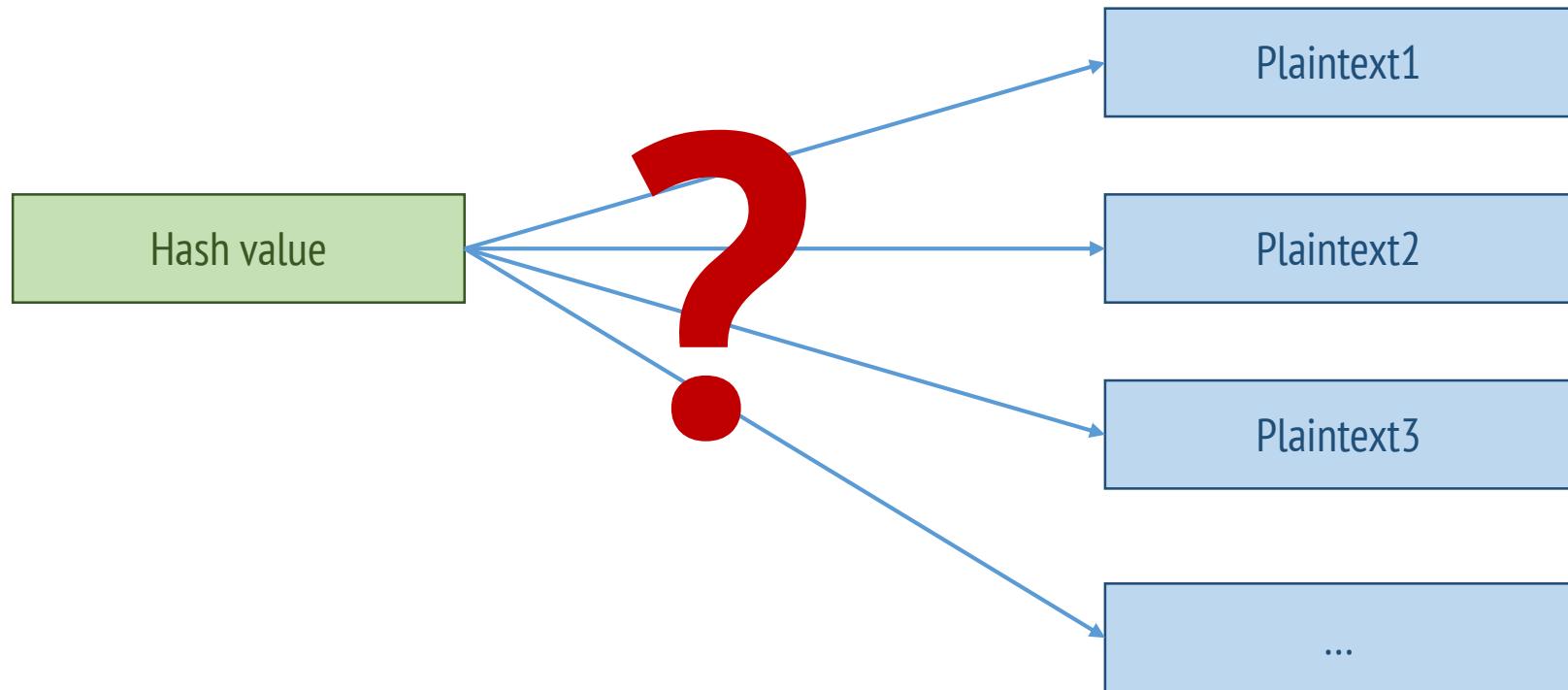
Hash functions

Unique per plaintext (low collision)



Hash functions

Difficult to reverse (one way)



Hash functions

Size of Hash value is independent from size of Plaintext.

Size of Hash value is given by the **hash function**.

Hash functions

Size of Hash value is independent from size of Plaintext.

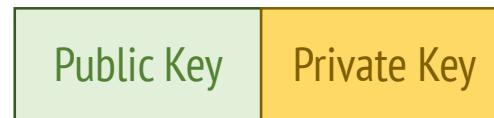
Size of Hash value is given by the **hash function**.

Example:

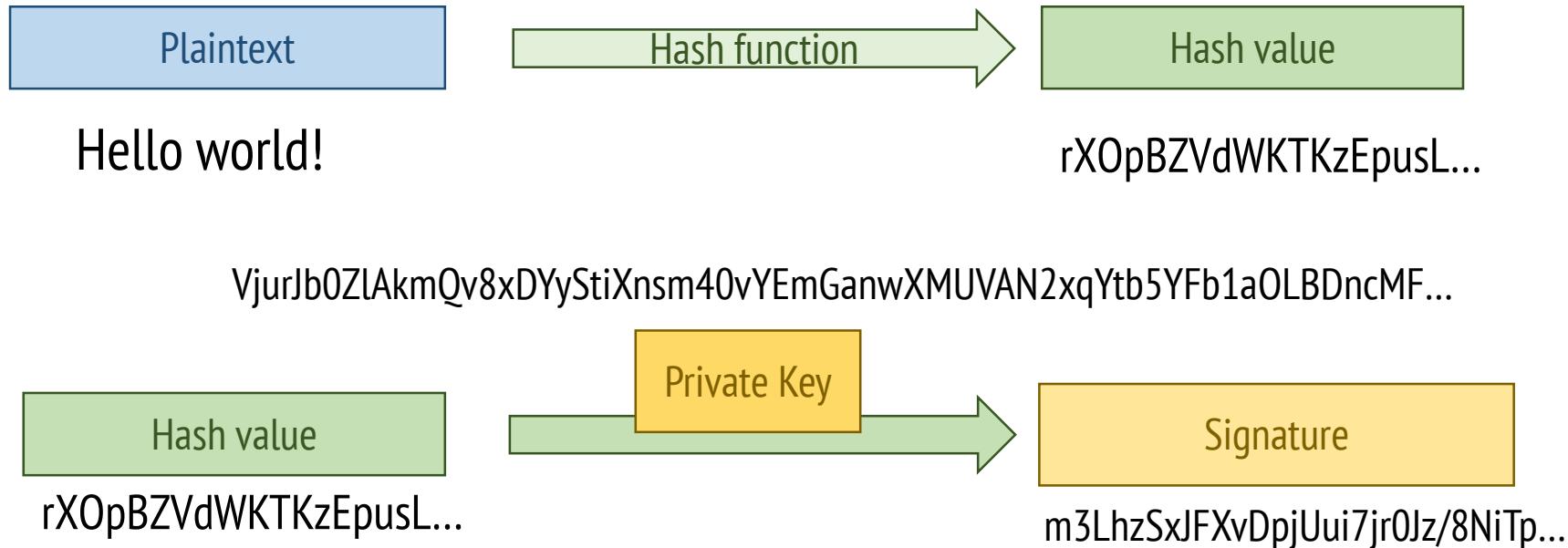
SHA256 = 256 bits

SHA512 = 512 bits

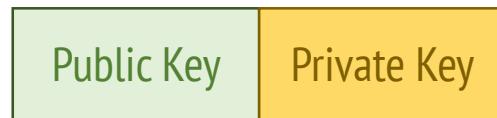
Signatures



Signing:



Signatures



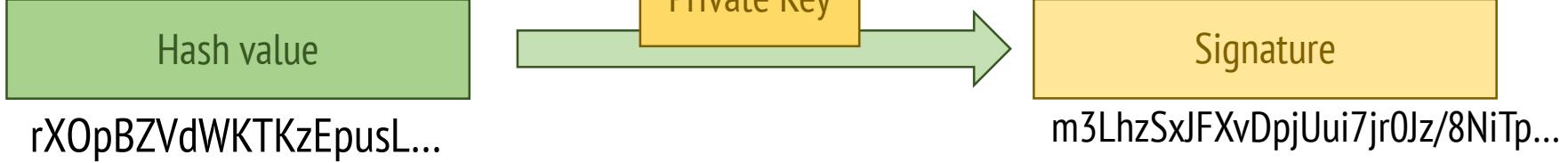
Signing:



Hello world!

rXOpBZVdWKTkzEpusL...

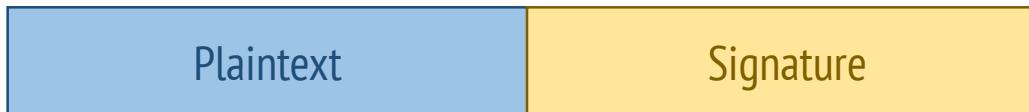
VjurJb0ZlAkmQv8xDYyStiXnsm40vYEmGanwXMUVAN2xqYtb5YFb1aOLBDncMF...



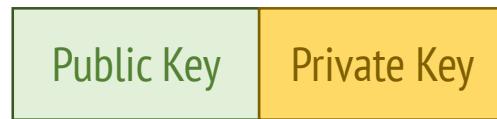
rXOpBZVdWKTkzEpusL...

m3LhzSxJFXvDpjUui7jr0Jz/8NiTp...

Message:

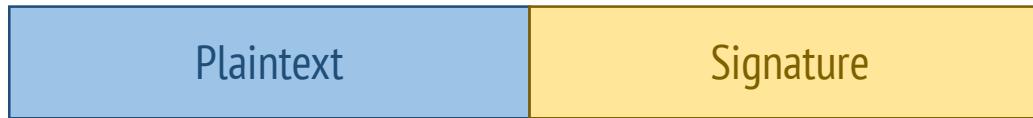


Signatures



Verification:

Message:



Plaintext

Hash function

Hash value

rXOpBZVdWKTkzEpusL...

WMWXV1cFZL7B4juLzULK7y2WFFv/9yyRVmDBuy6WbSWYVs...

Signature

Public Key

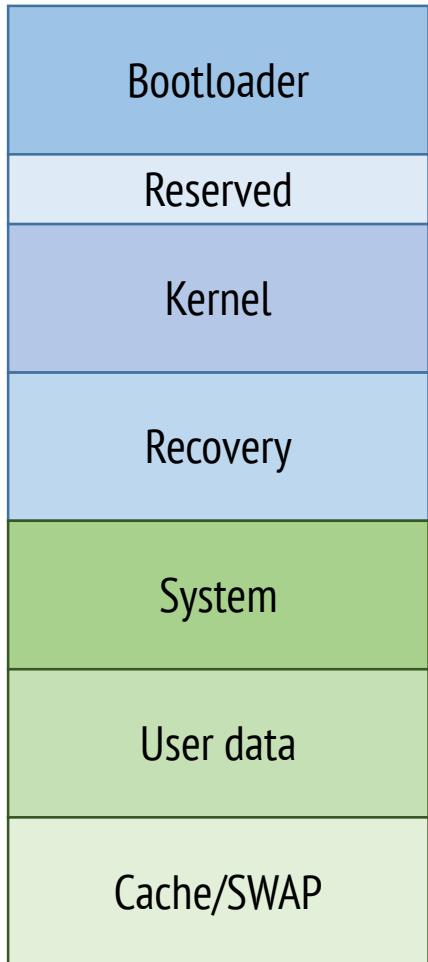
Hash value

m3LhzSxJFXvDpjUui7jr0Jz/8NiTp...

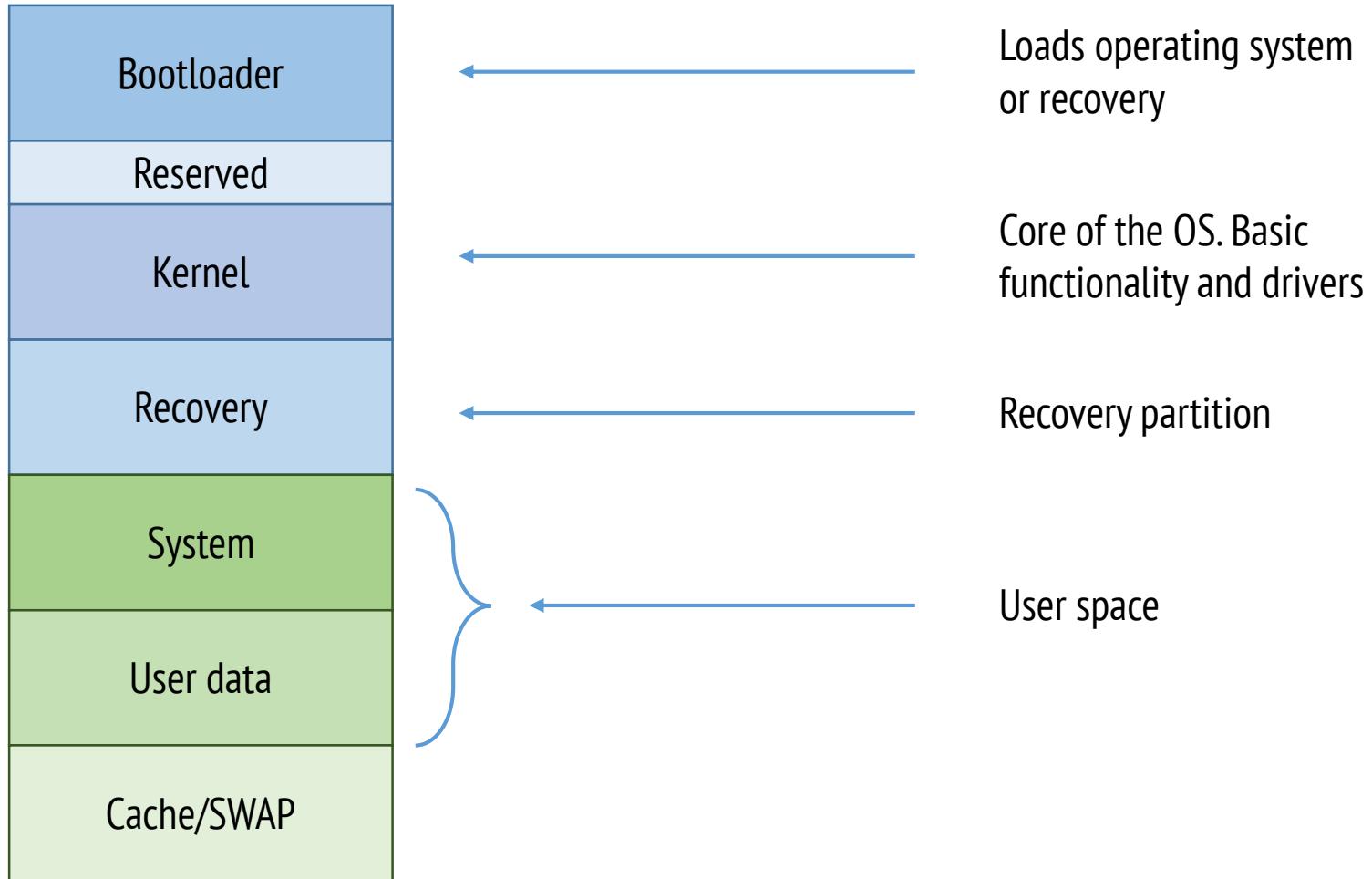
rXOpBZVdWKTkzEpusL...

System architectures

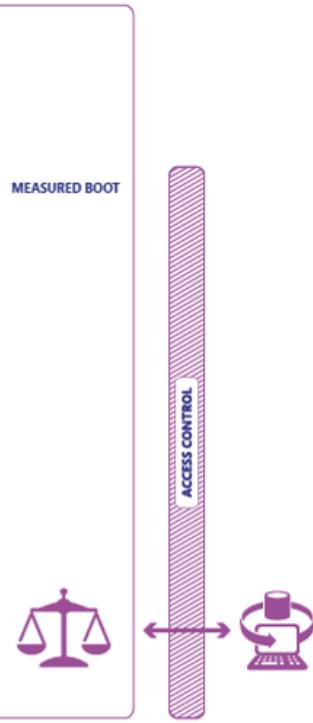
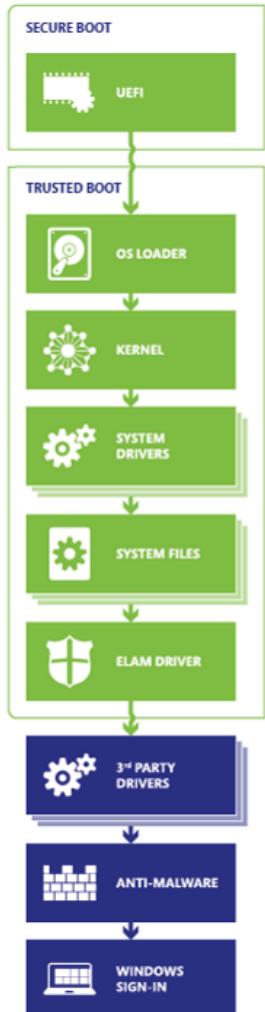
Generic architecture



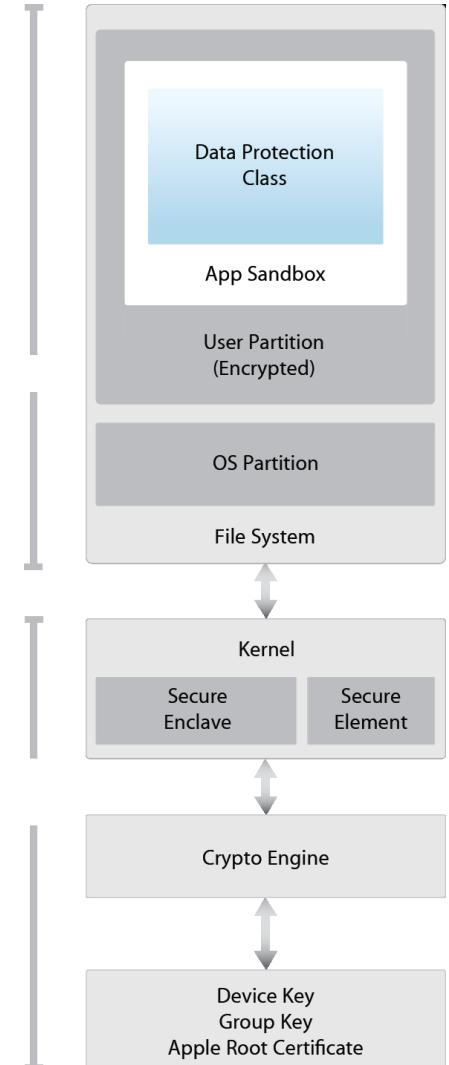
Generic architecture



System architecture



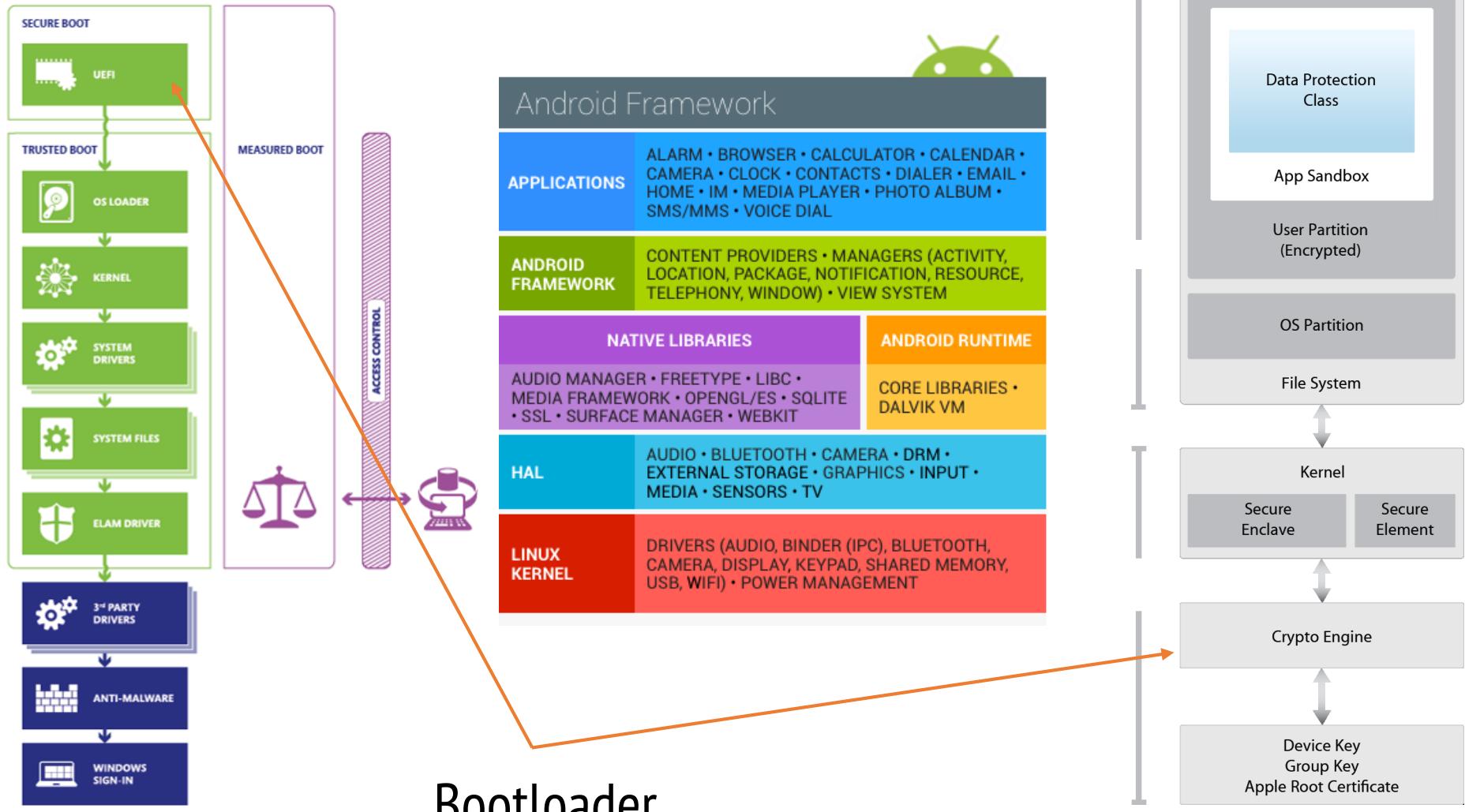
Android architecture



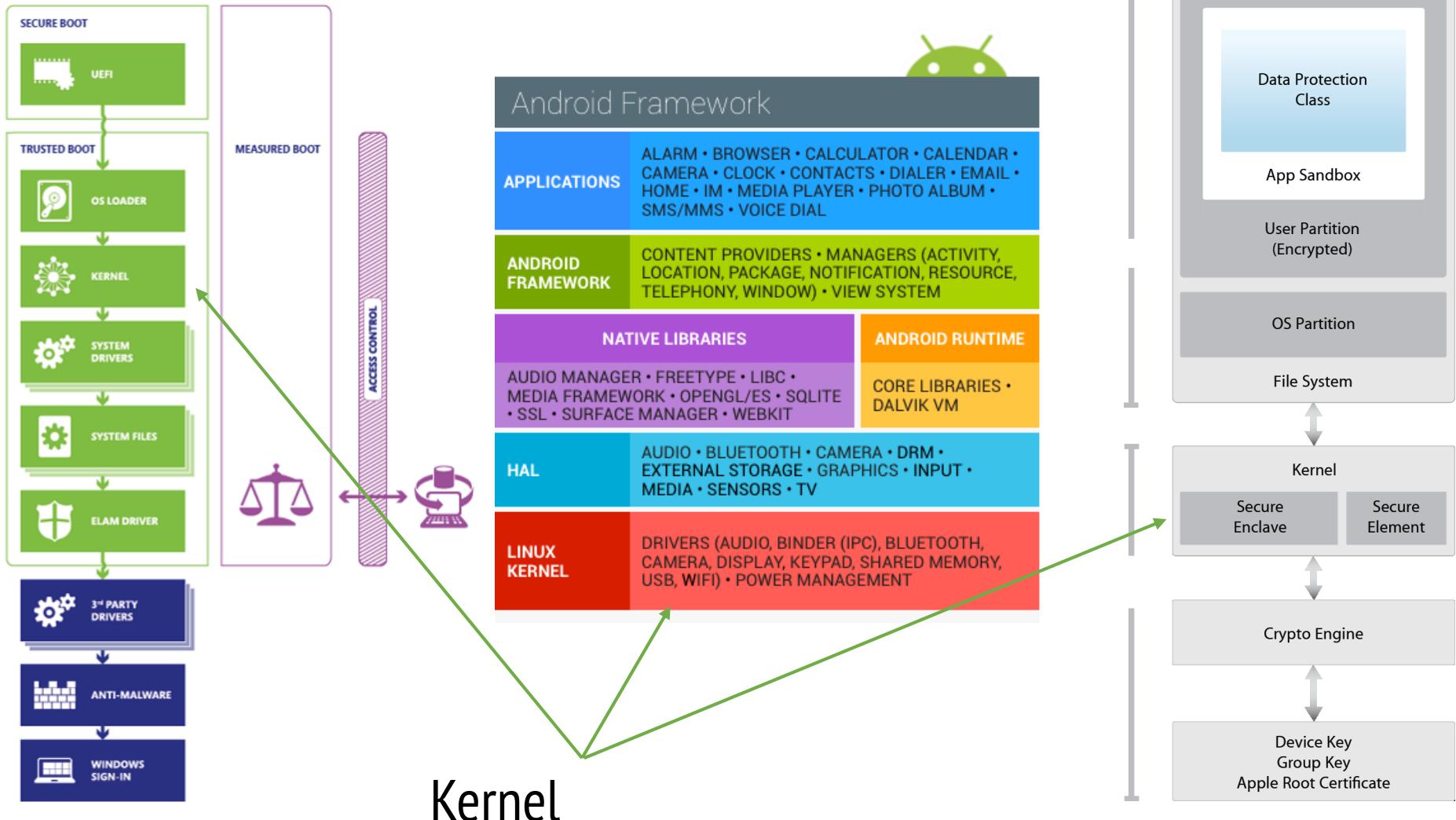
Windows architecture

iOS security architecture

System architecture

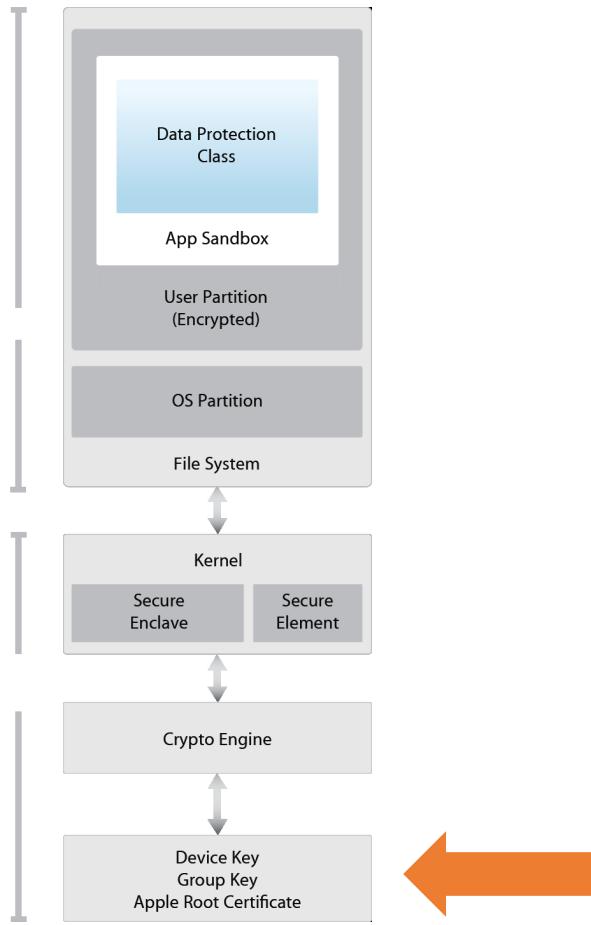


System architecture



Trusted boot

Root of trust



1. The bootloader is the guardian of the device state and is responsible for initializing the TEE and binding its root of trust.
2. If rooting software compromises the system before the kernel comes up, it will retain that access.
3. The bootloader verifies the integrity of the boot and/or recovery partition before moving execution to the kernel.
4. **Hardware** root of trust is fixed because is laid down during chip fabrication.
5. **Non-hardware** root of trust can be changed because is stored on non-volatile memory (e.g. NAND).

How to verify?

How to verify?

Directly hash its contents and compare them to a stored value.

How to verify?

Directly hash its contents and compare them to a stored value.

Some issues:

- Verifying an entire block device can take an extended period
- Will consume much of a device's power

Dm-verity (android)

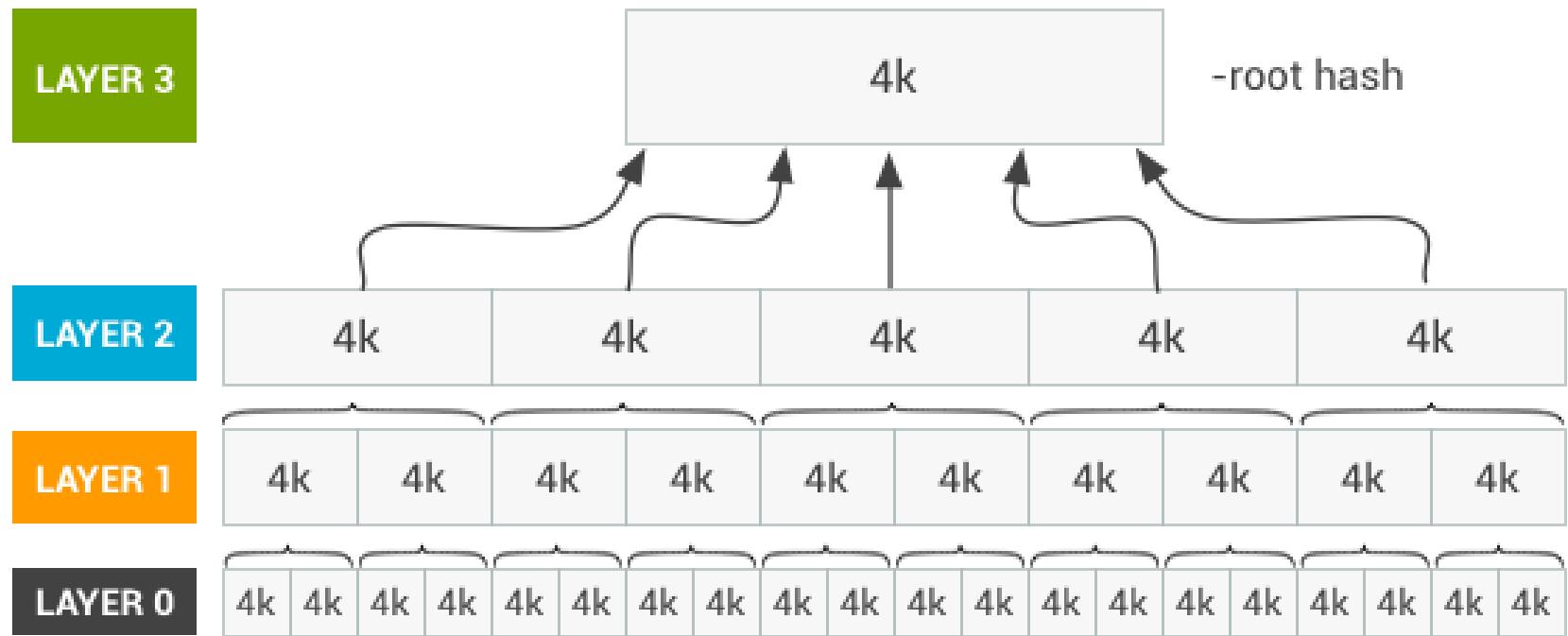
Uses:

- Block storage
- Cryptographic hash function, e.g. SHA256
- Cryptographic hash tree (i.e. Merkel Tree)

Benefits of Dm-verity

- Verifies blocks individually and only when each one is accessed
- The HASH operation is done when the block is read into memory: the block is hashed in parallel
- The hash is then verified up the tree

Boot/recovery partition



Android partition verification

Question

- What properties does the root hash provide?

Question

- What properties does the root hash provide?
 - Integrity

Question

- What properties does the root hash provide?
 - Integrity
- How can we provide authentication to the data?

Question

- What properties does the root hash provide?
 - Integrity
- How can we provide authentication to the data?
 - Use a key to sign the root hash

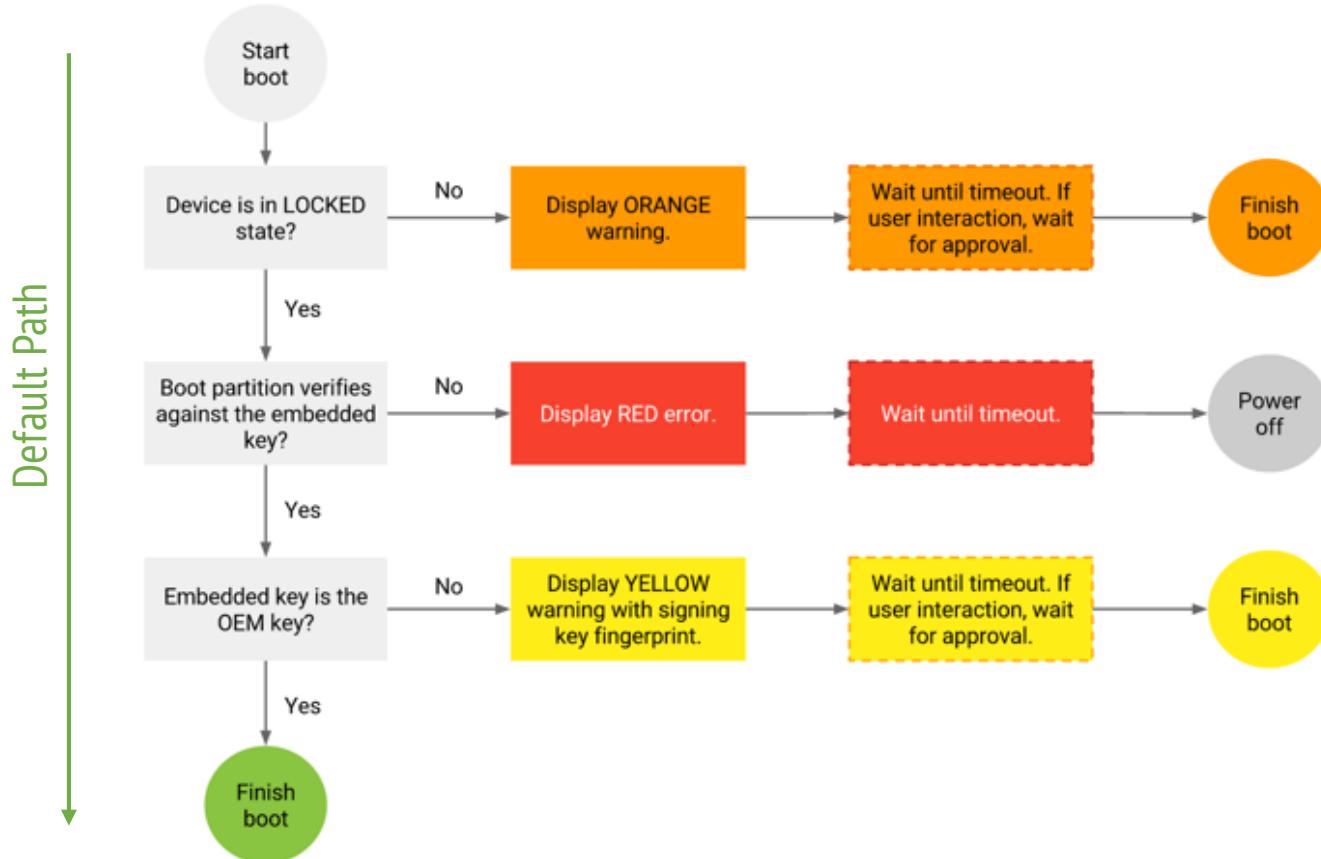
Question

- What properties does the root hash provide?
 - Integrity
- How can we provide authentication to the data?
 - Use a key to sign the root hash
- Can this method be applied to all types of partitions? Why?

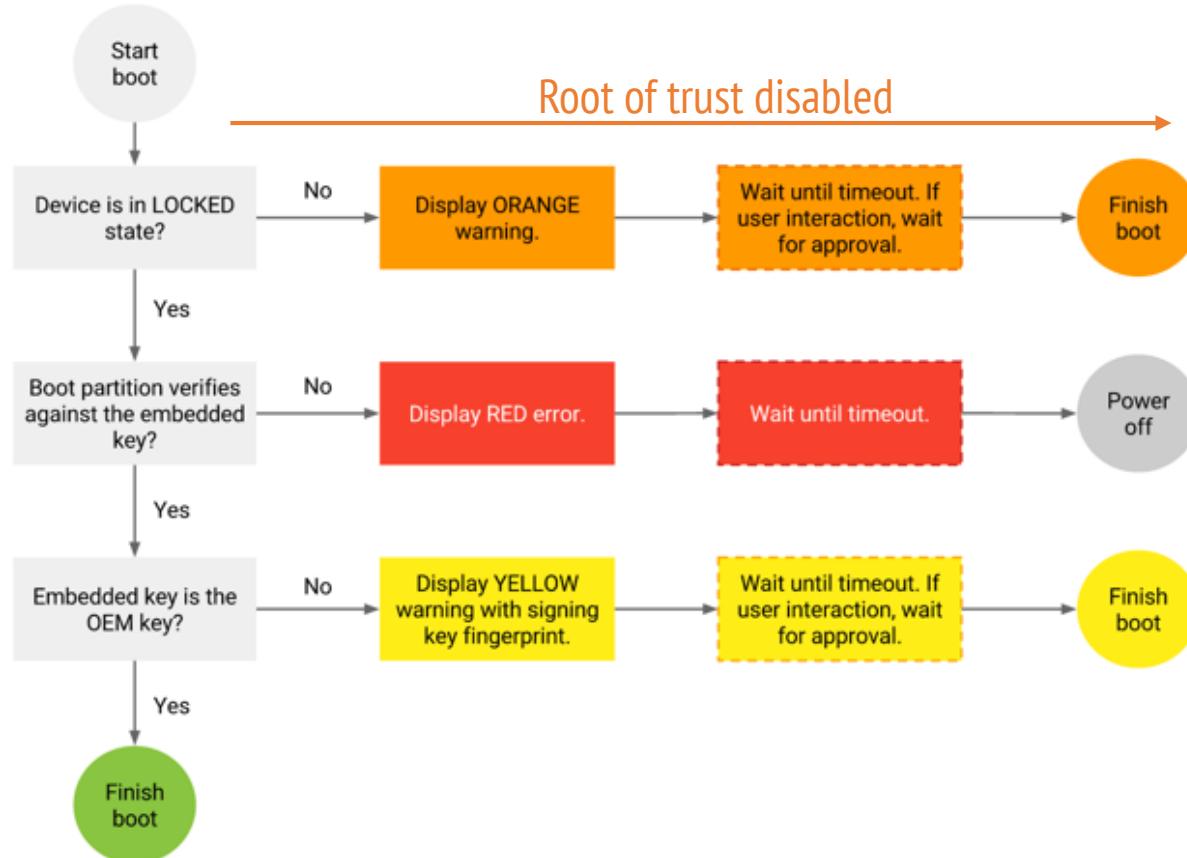
Question

- What properties does the root hash provide?
 - Integrity
- How can we provide authentication to the data?
 - Use a key to sign the root hash
- Can this method be applied to all types of partitions? Why?
 - Just to read-only. Read-write partitions' HASH values would change whenever data is modified.

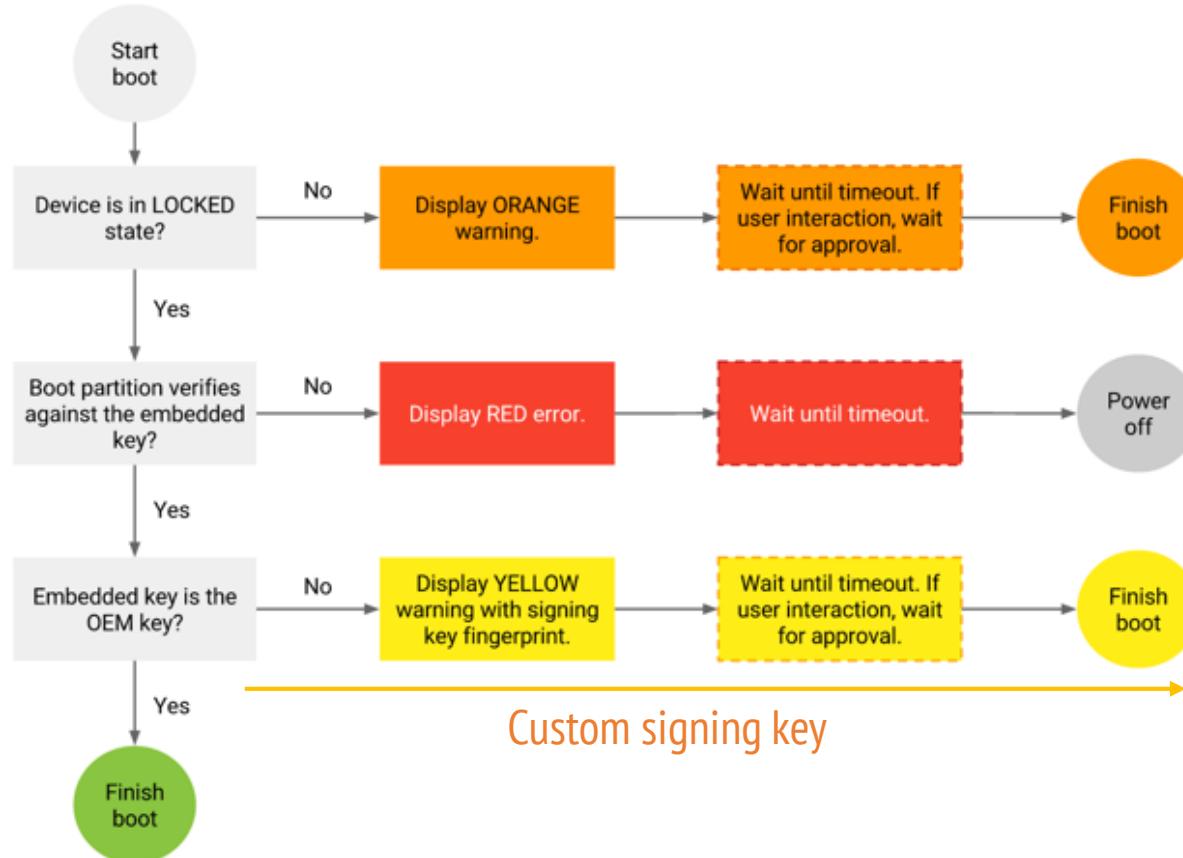
Boot flow (Android)



Boot flow (Android)



Boot flow (Android)



Main principles applied

- Authenticate: every component of the system is authenticated
- Control access: prevent loading the OS without a secure boot loader
- Make security usable: simple messages aimed at users
- Fail secure: stop the booting process if anything goes “wrong”
- ...

Exercise

Describe necessary steps to perform a firmware update

Exercise

Describe necessary steps to perform a firmware update

1. Boot in a secure/trusted mode

Exercise

Describe necessary steps to perform a firmware update

1. Boot in a secure/trusted mode
2. Check the signature of the firmware update using trusted ROM key

Exercise

Describe necessary steps to perform a firmware update

1. Boot in a secure/trusted mode
2. Check the signature of the firmware update using trusted ROM key
3. Install firmware update

Exercise

Describe necessary steps to perform a firmware update

1. Boot in a secure/trusted mode
2. Check the signature of the firmware update using trusted ROM key
3. Install firmware update
4. Update the bootloader to match the new firmware

Exercise

Describe necessary steps to perform a firmware update

1. Boot in a secure/trusted mode
2. Check the signature of the firmware update using trusted ROM key
3. Install firmware update
4. Update the bootloader to match the new firmware

Any thing else?

Individual research

Well...

Firmware downgrade attacks: installing an older firmware which might have (known) vulnerabilities.

Give some solutions to prevent this!

What else?

- Kernel secure?
 - Yes
- Can we update OEM keys in case of compromise?
 - No

What else?

- Kernel secure?
 - Yes
- Can we update OEM keys in case of compromise?
 - No
- What if we can still compromise the system by exploiting some design flaw(s) ?
 - Can we at least detect that and notify the user? (we could before)
 - **Can we prevent sensitive data compromise? (e.g. encryption keys)**

Secure the boot process
using hardware

Hardware anchored security

Trusted Platform Module (TPM)

- a secure cryptoprocessor
- generates cryptographic material (keys and random numbers)
- performs remote attestation
- protects cryptographic material by binding and sealing

Trusted Execution Environment (TEE)

- secure area of the main processor
(e.g. TrustZone in ARM, SGX in Intel)
- code and data loaded inside are protected with respect to confidentiality and integrity
- provides isolated execution

What can I do with a TPM?

- Platform integrity
- Disk encryption
- Password protection
- Digital rights management
- Protection and enforcement of software licenses
- Prevention of cheating in online games

What can I do with a TPM?

- **Platform integrity**
- Disk encryption
- Password protection
- Digital rights management
- Protection and enforcement of software licenses
- Prevention of cheating in online games

Platform integrity

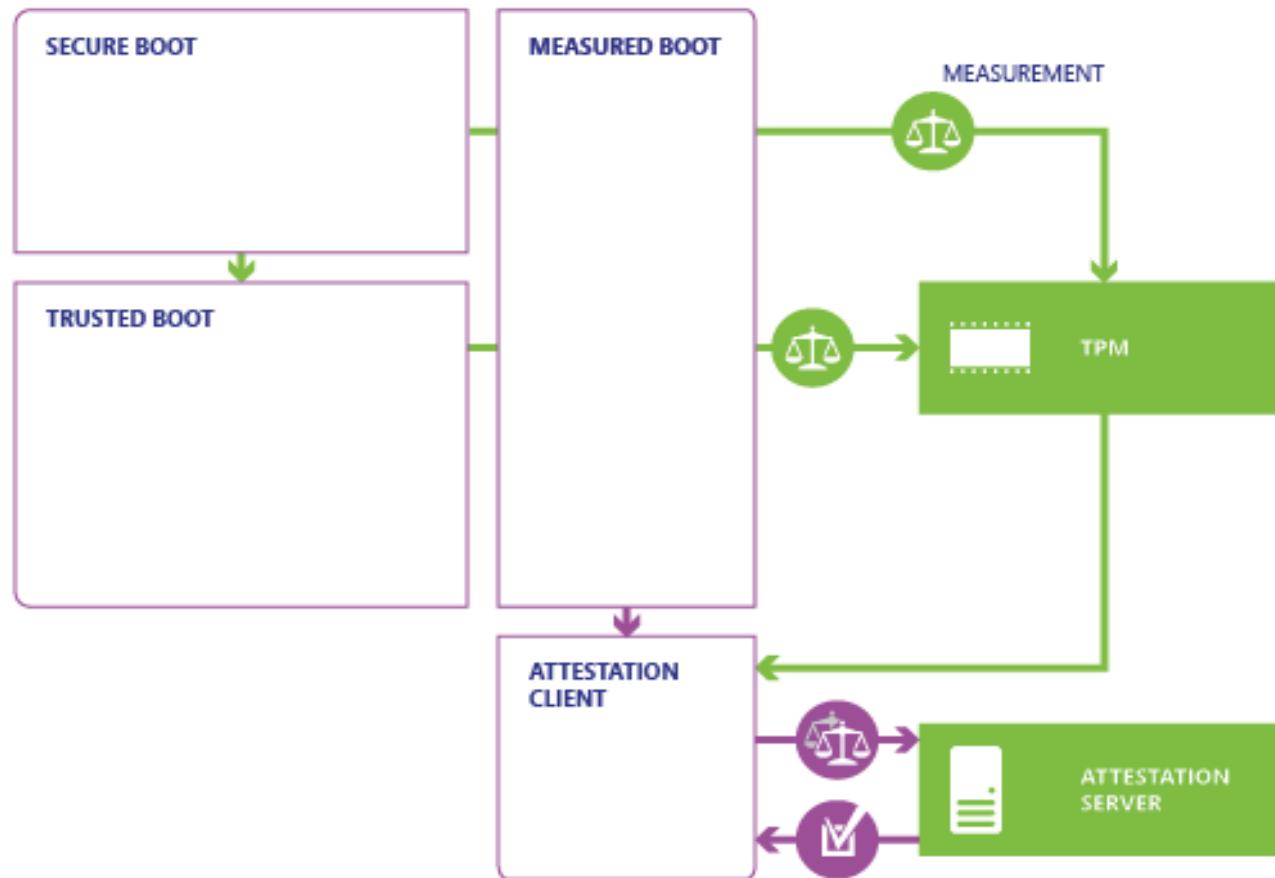
Requirements

- TPM
- UEFI
- Linux Unified Key Setup (LUKS) or BitLocker Drive Encryption

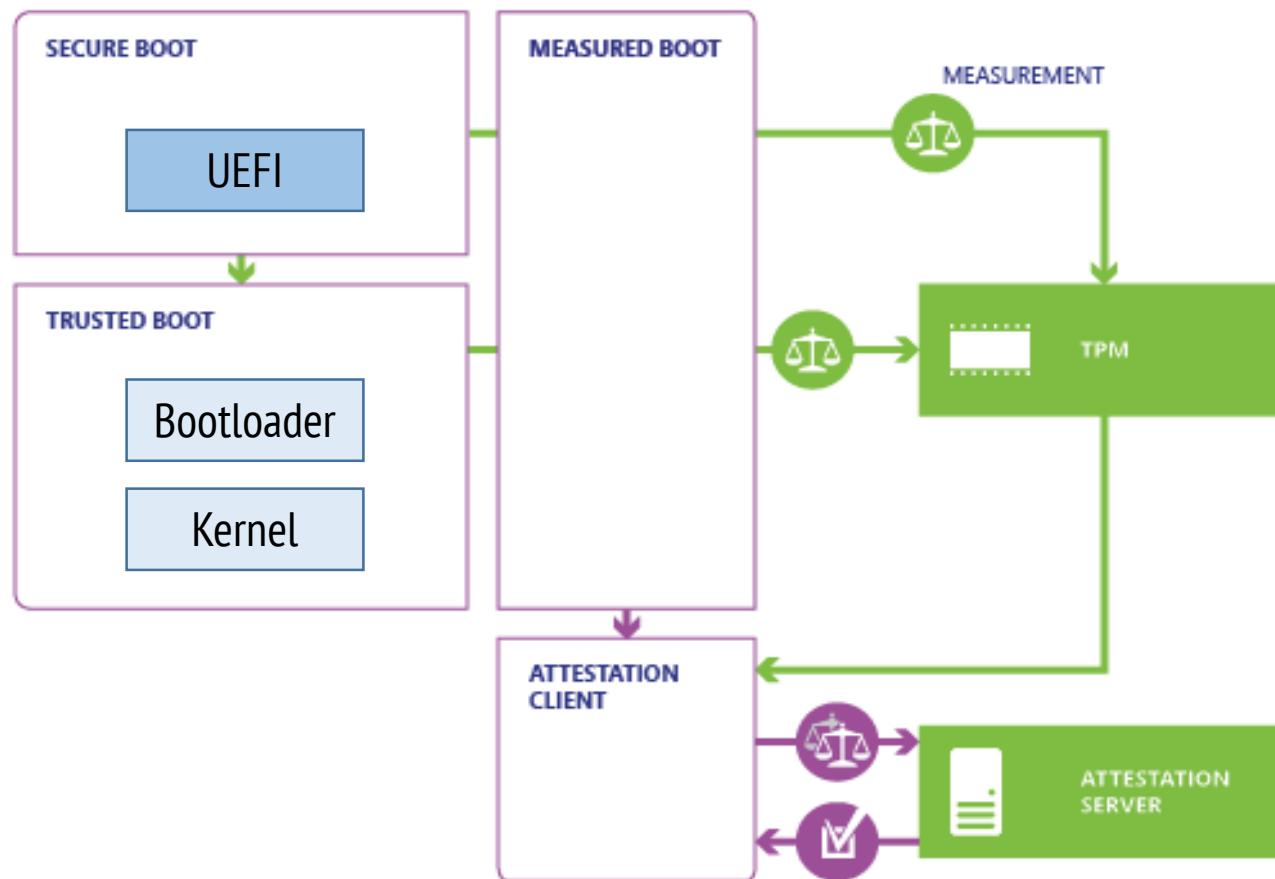
Enforces

- "root of trust"

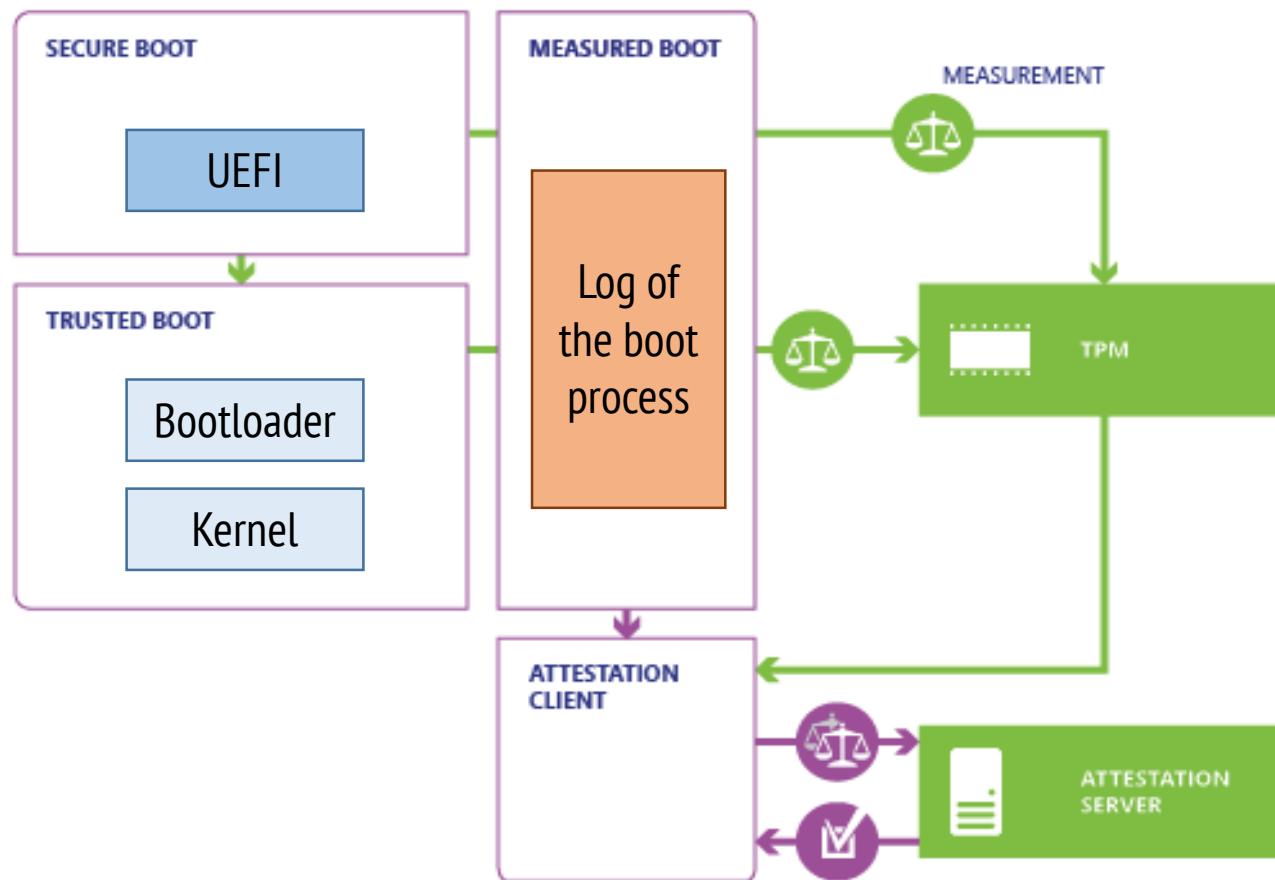
Securing the boot process with a TPM?



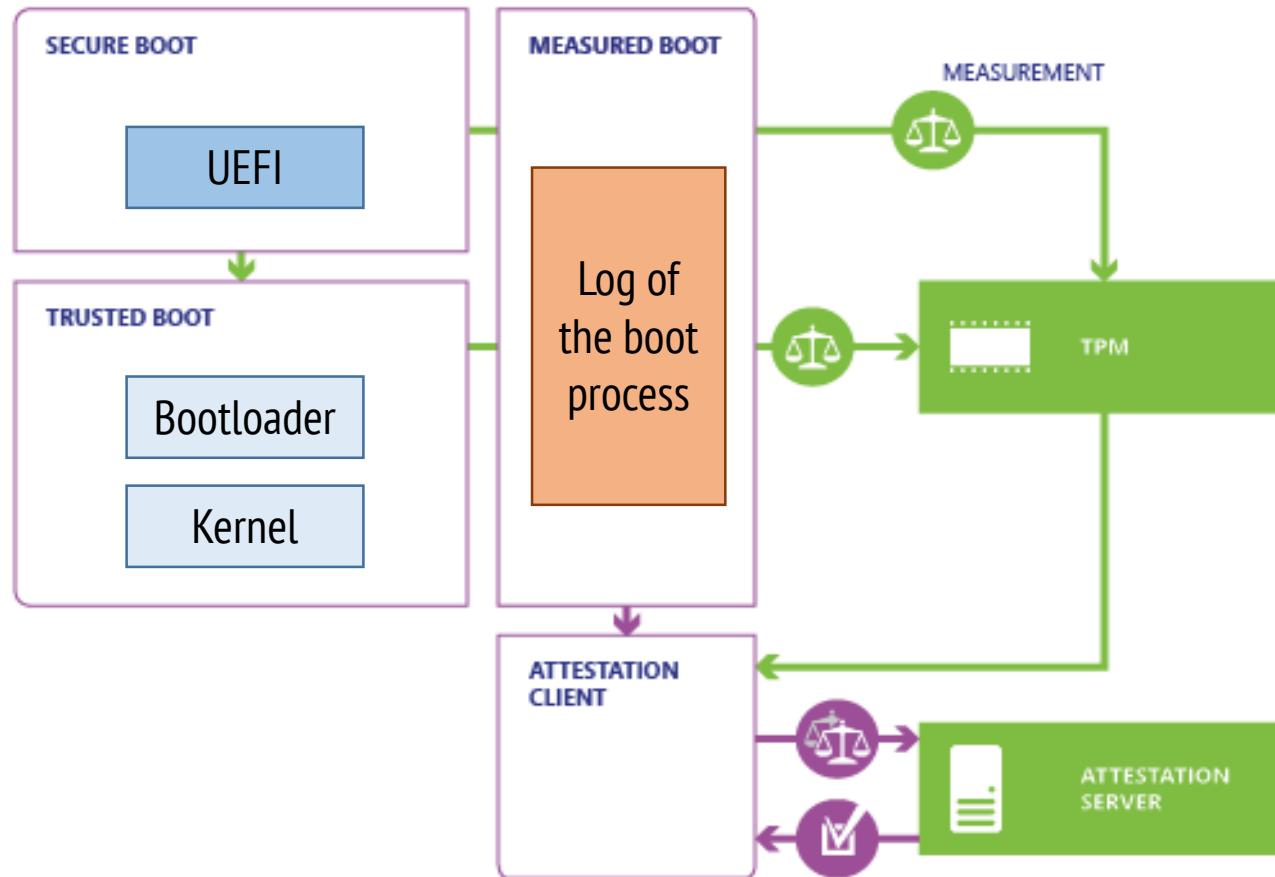
Securing the boot process with a TPM?



Securing the boot process with a TPM?



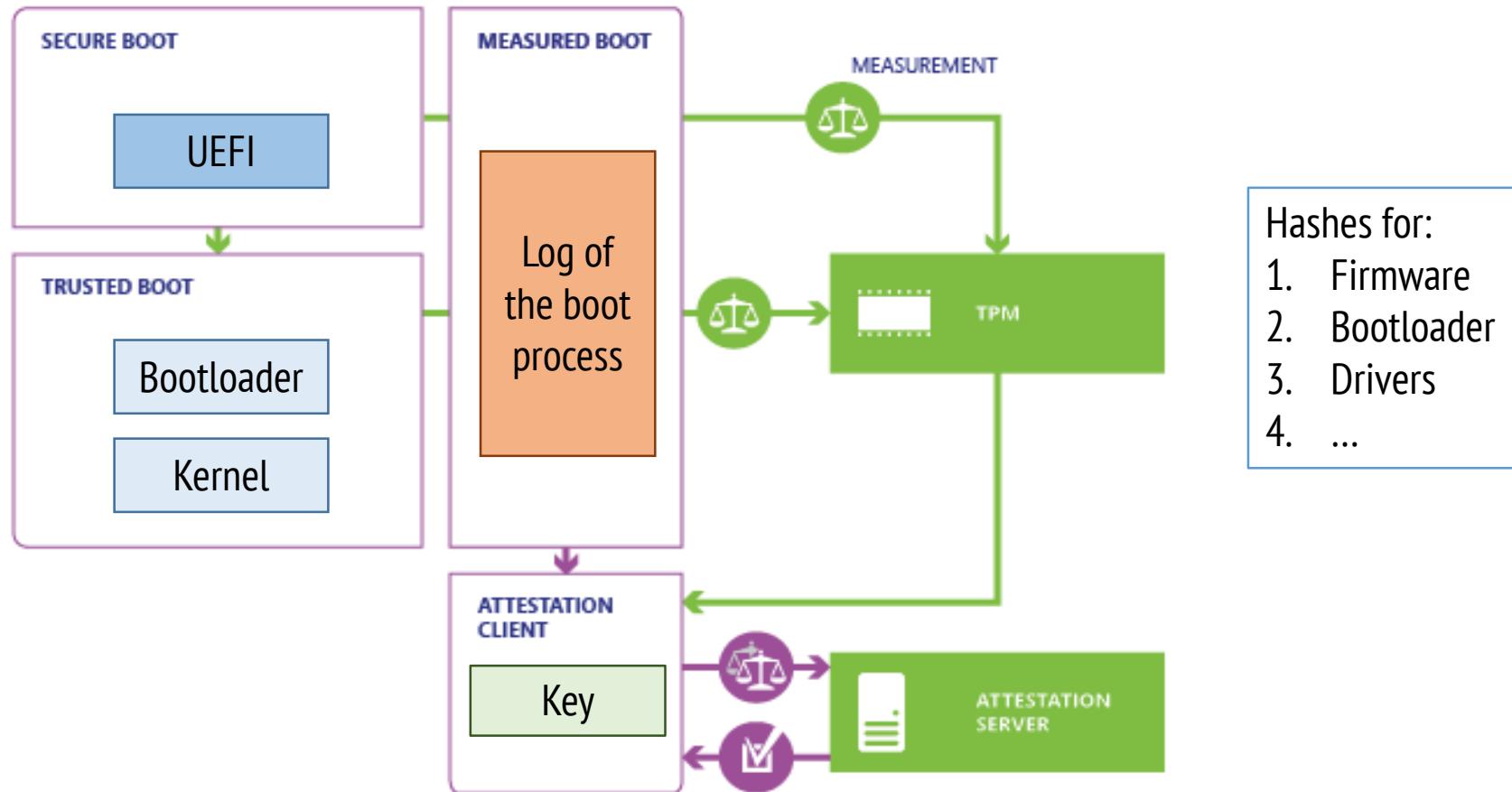
Securing the boot process with a TPM?



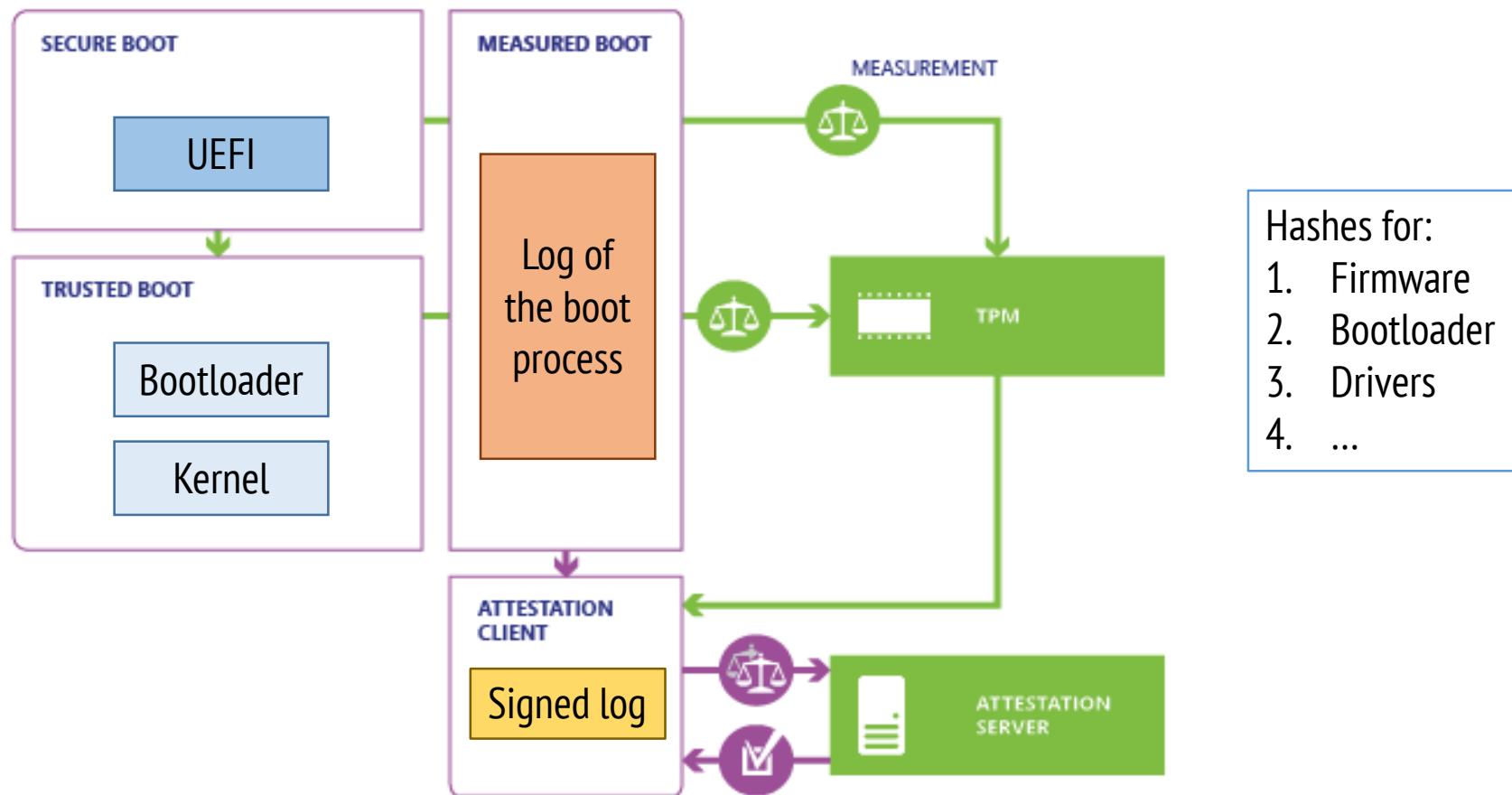
Hashes for:

1. Firmware
2. Bootloader
3. Drivers
4. ...

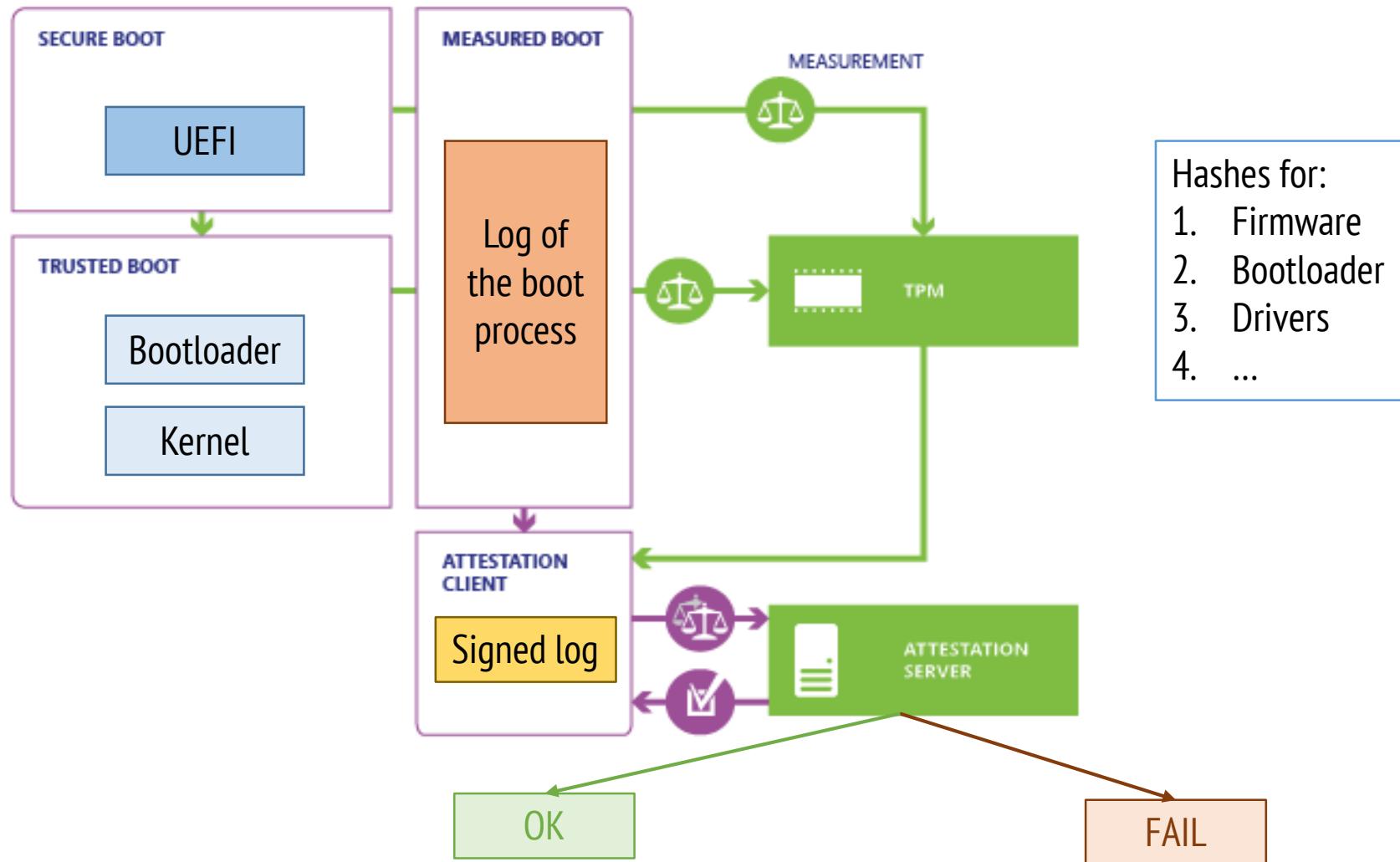
Securing the boot process with a TPM?



Securing the boot process with a TPM?



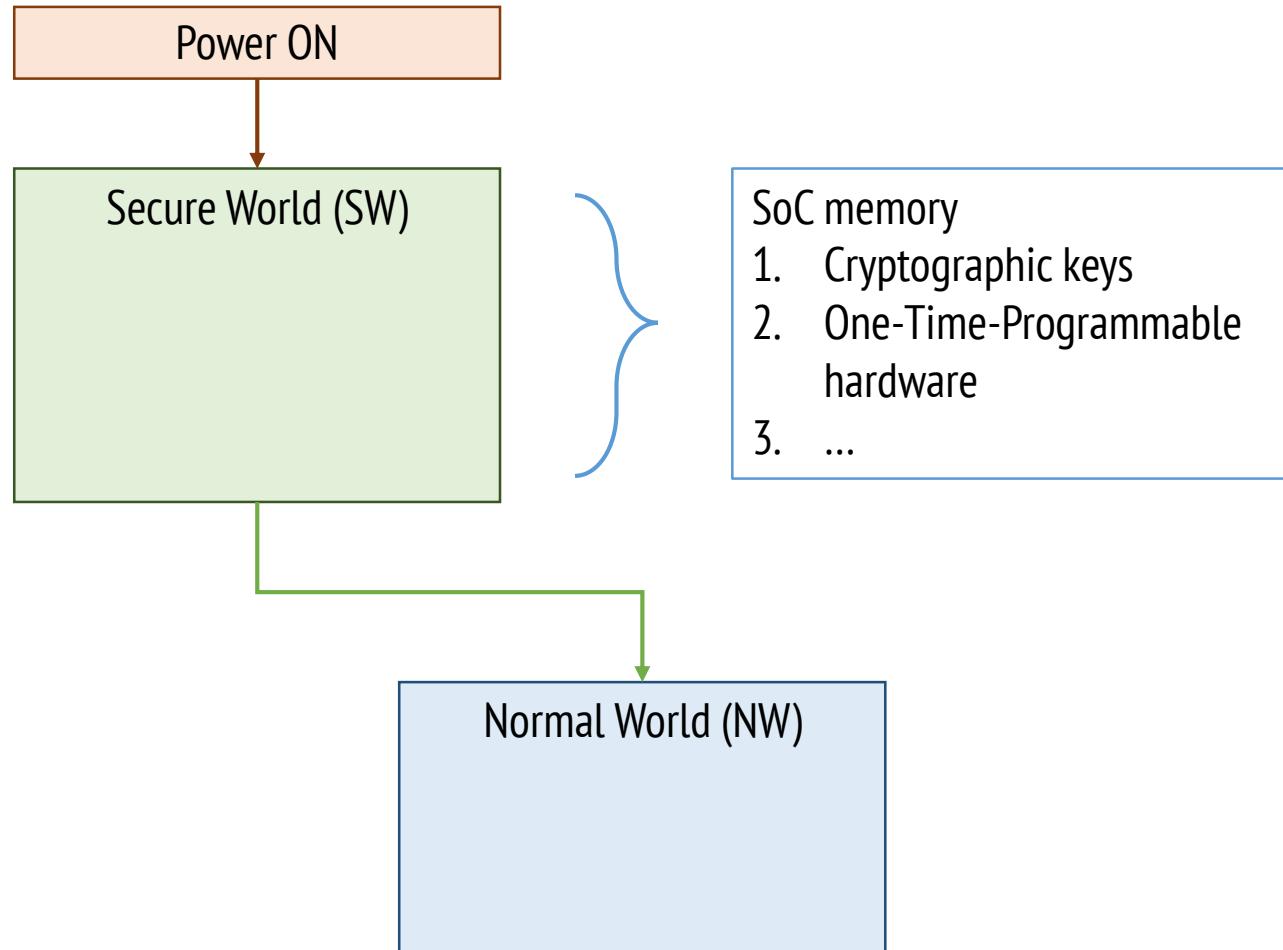
Securing the boot process with a TPM?



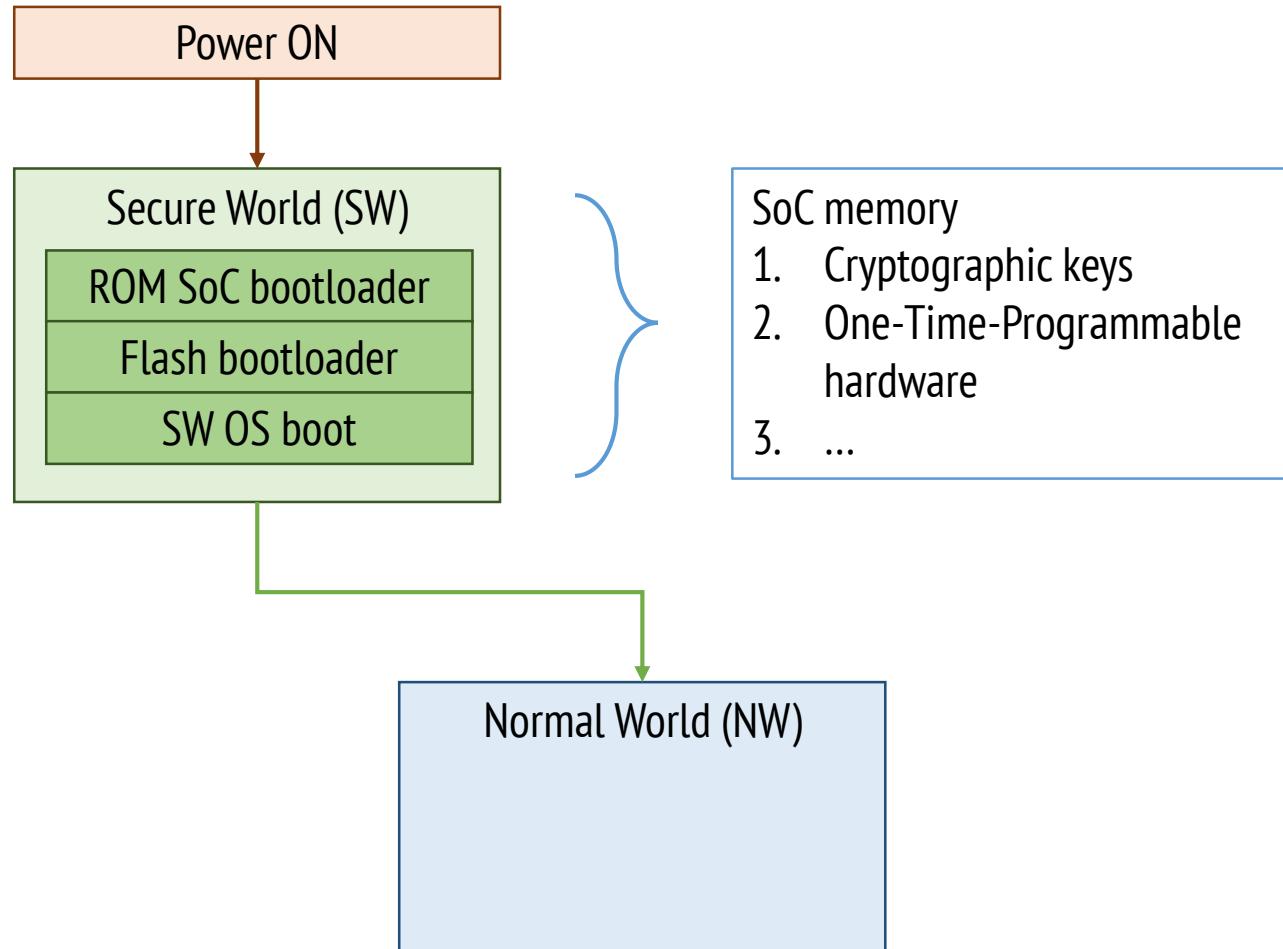
Main principles applied

- Authenticate requests: the TPM authenticates UEFI application and the bootloader
- Fail secure: if things go wrong stop
- Audit and monitor: keep logs of the boot process and verify them against a known and trusted logs
- ...

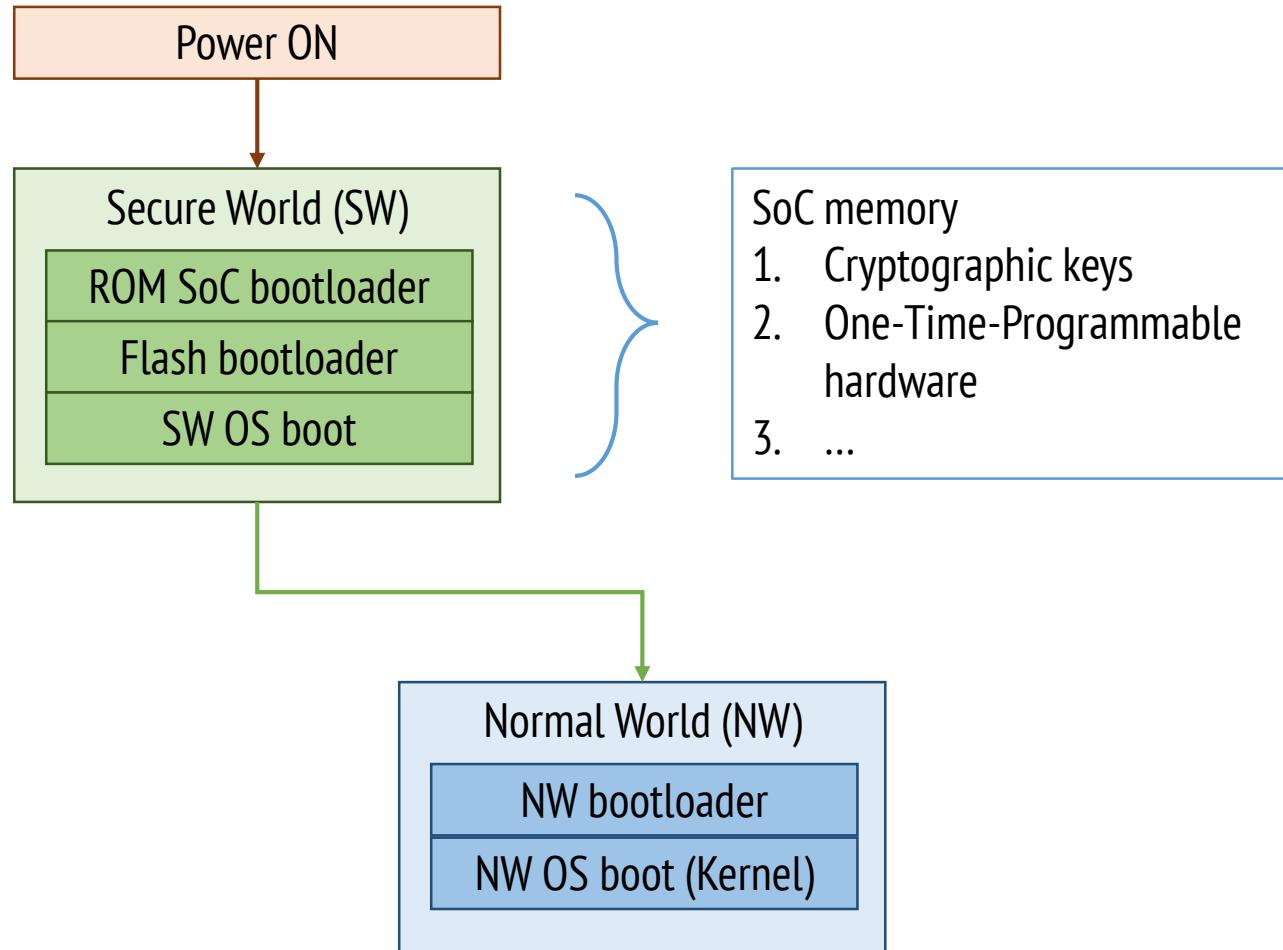
Platform integrity with ARM TrustZone



Platform integrity with ARM TrustZone



Platform integrity with ARM TrustZone



Main principles applied

- Economise mechanism: keep the system simple i.e. use a secure OS with controlled I/O to load the main kernel
- Fail secure: if things go wrong stop
- ...

Secure Enclave Processor (SEP)

- Enable sensitive data to be stored securely
- Performs secure services for the rest of the SOC
- Runs its own operating system (SEPOS) which includes: kernel, drivers, services, and applications
- Supports multiple services: TouchID, ApplePay...

Secure Enclave Processor (SEP)

Hardware functionality

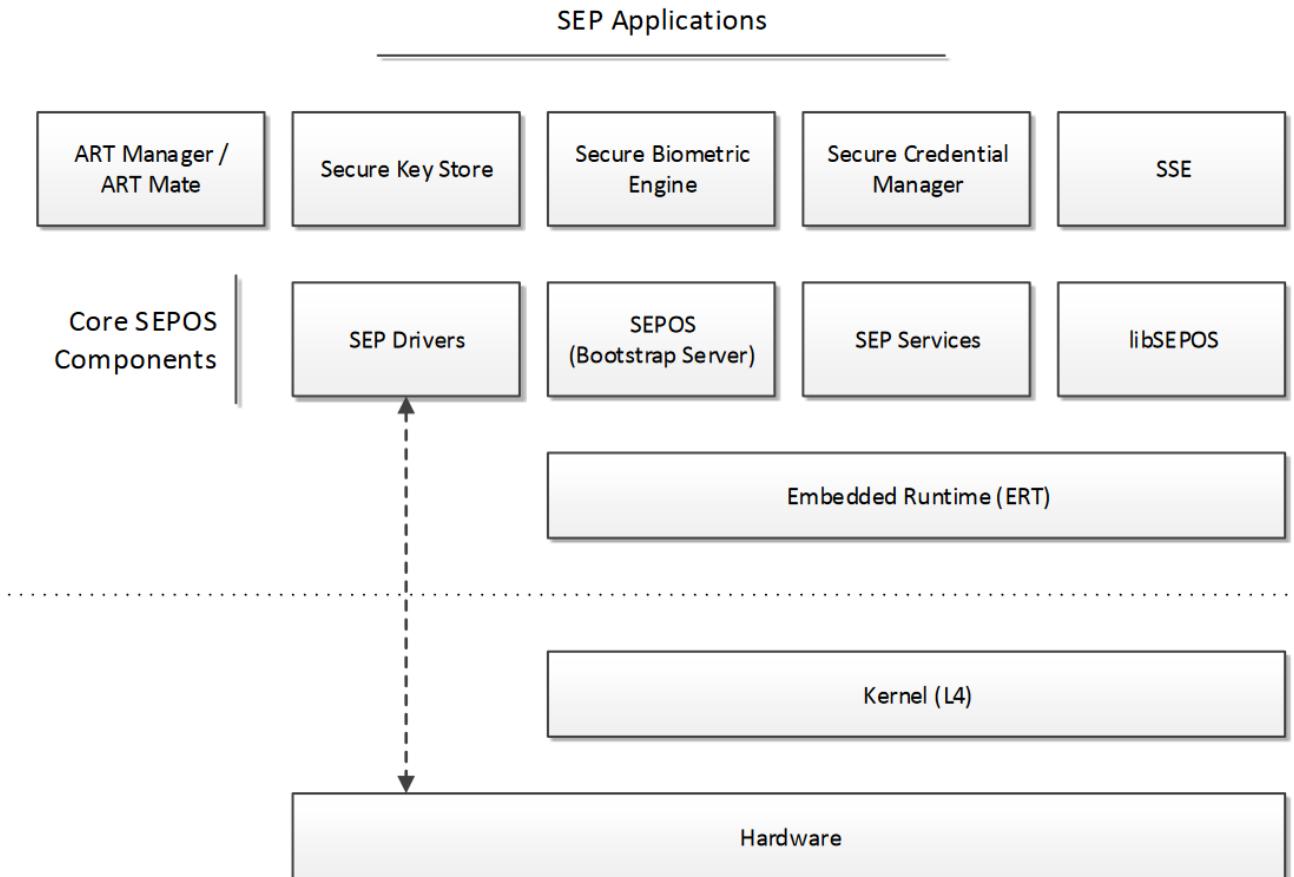
- Crypto engine
- Random Number Generator
- Fuses
- GID/UID
- Dedicated scratch RAM
- Hardware “filter” to prevent application processor (AP) to SEP memory access

Secure Enclave Processor

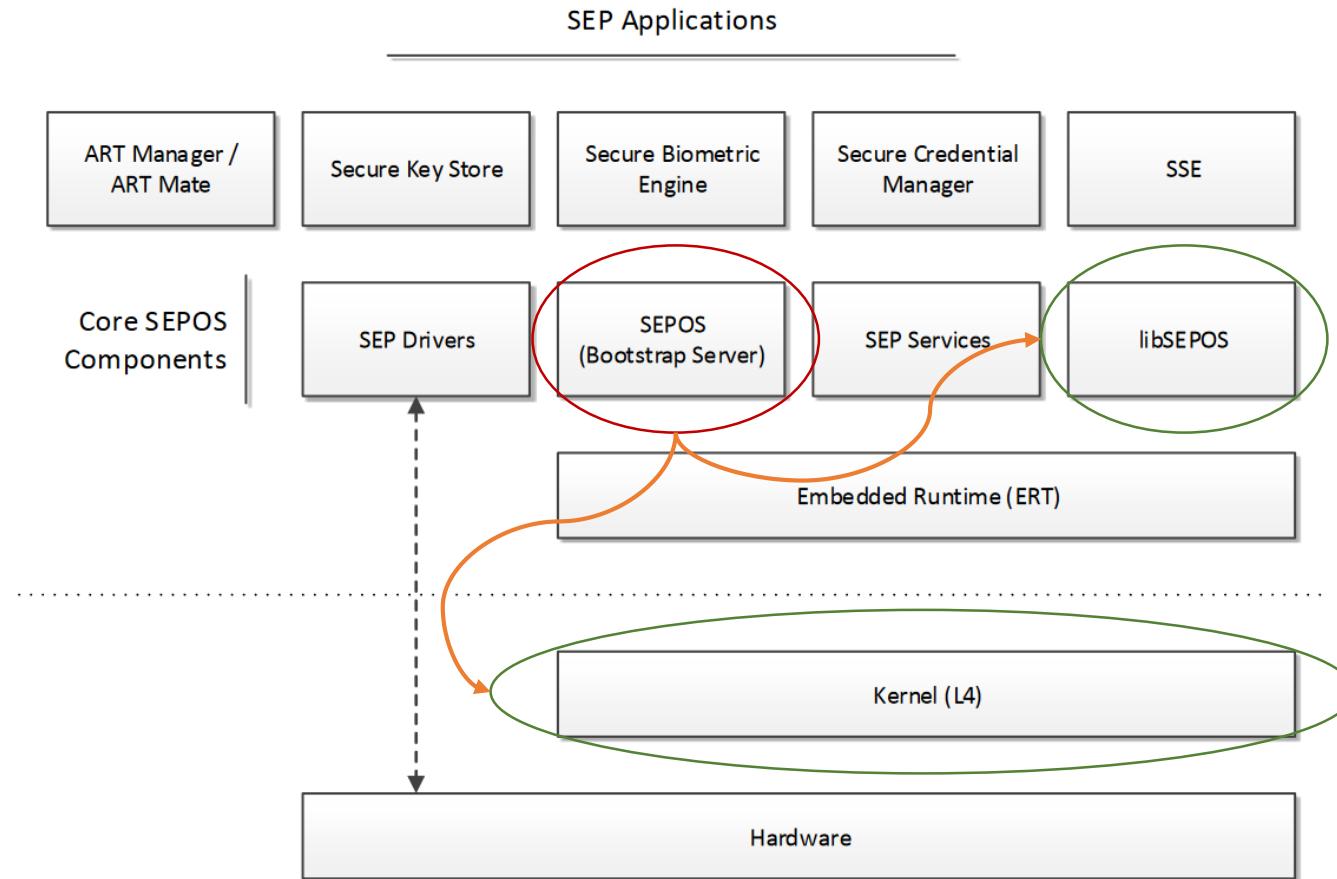
Shared functionality with application processor

- Clock
- RAM
- Power manager
- ...

SEP Architecture

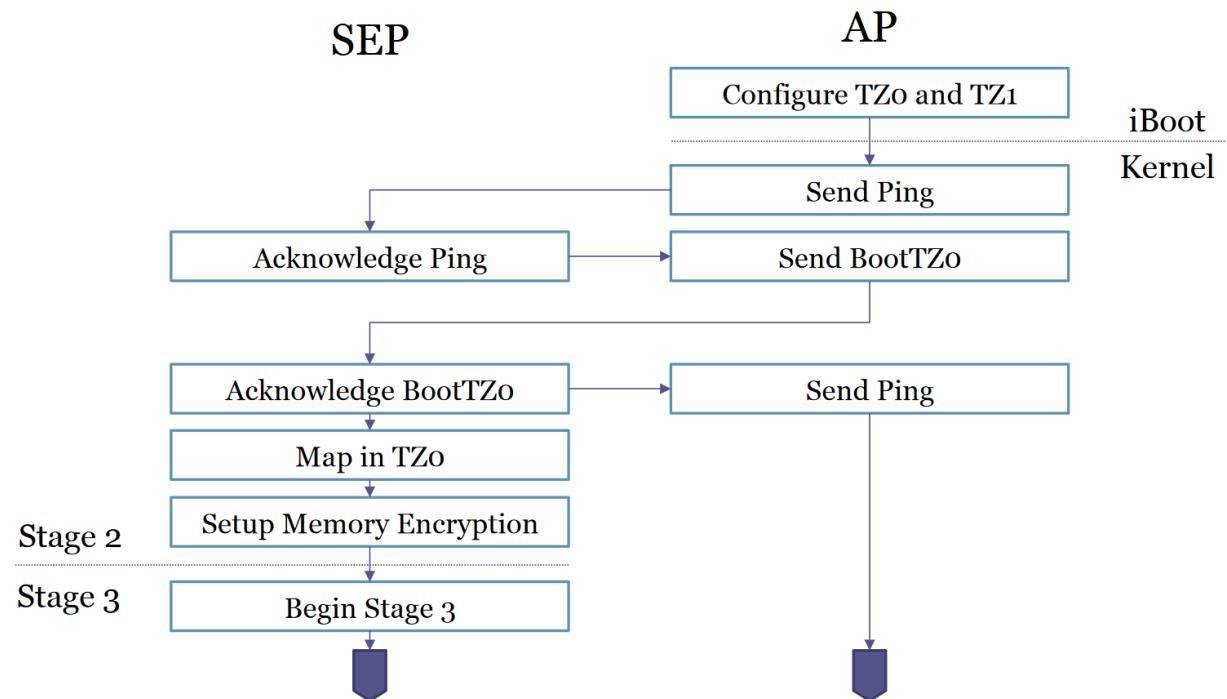


SEP Architecture



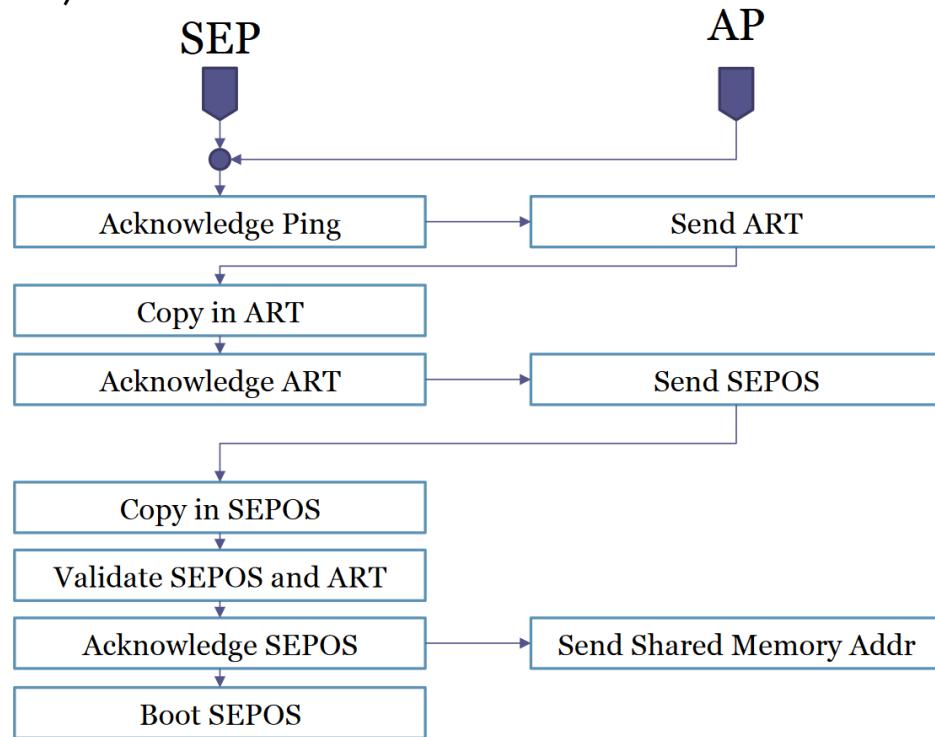
Secure Enclave Processor (SEP)

1. Configure Trust Zones 0/1
(TZ0-SEP, TZ1-AP)
2. Check for SEP
3. Configure memory protection



Secure Enclave Processor (SEP)

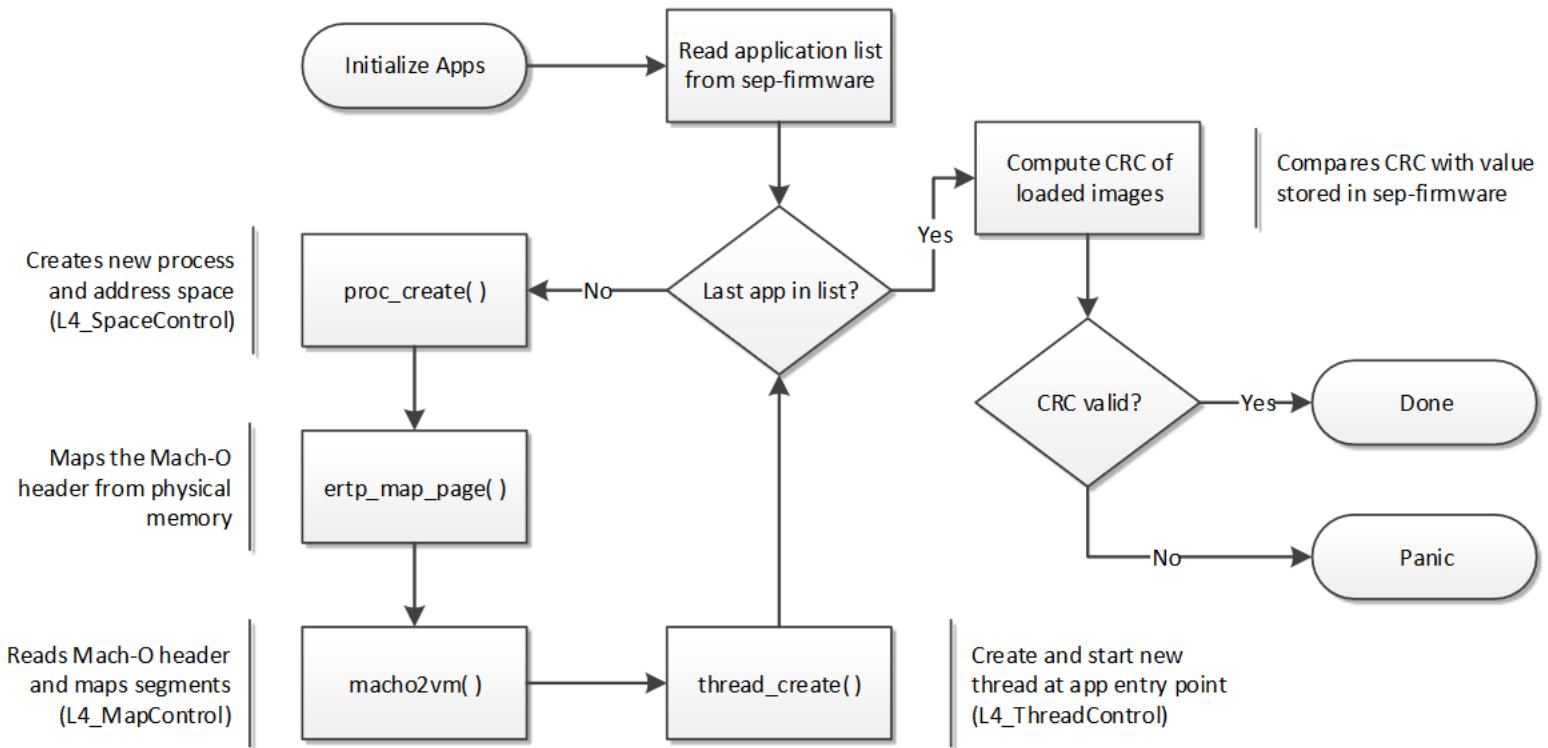
1. Send anti-replay token (ART)
2. Verify ART
3. Verify the SEP operating system (SEPOS)
i.e. 4096 bytes
4. Establish shared memory location



SEP communication

- Secure Mailbox allows the AP to communicate with the SEP
- Supported through the SEP Manager API
- Implemented using the SEP device I/O registers

SEP interaction



Main principles applied

- Authenticate requests: messages to the SEP are authenticated, apps are authenticated before use
- Fail secure: if anything goes wrong stop execution
- Segregation of duties: all cryptographic operations and accesses to secure hardware are intermediated by the SEP
- Secure the weakest link: protect access to memory

Application security

Multiple Layers of Defense



Main “techniques”

- Code signing
- Runtime security
 - a) Mandatory access control (MAC)
 - b) Sandboxing
 - c) Memory protection (e.g. address space layout randomisation (ASLR), ARM Execute Never (XN))

Code signing

- Ensure applications have an approved source and haven't been tampered with
- Executable code is signed with store specific certificates
- Prevent applications from loading unsigned code resources and self-modifying code
- Access hardware with OS APIs

Mandatory access control (MAC)

- Support over all OS
 - Selinux in Linux and Android
 - Mandatory Integrity Control in Windows Vista/7/8/10
 - TrustedBSD variant in Apple IOS and OSX
- MAC is usually enforced over all processes, even processes running with root/superuser privileges.
- MAC usually defaults to denial: anything that is not explicitly allowed is denied.

Sandboxing

- Restrict applications from using data and resources from other apps.
- Each app has its own random “home directory”
- Run applications as non-privileged user
- Mount “important” partitions as read-only

Memory protection

- Address space layout randomisation (ASLR)
 - protects memory from corruption
 - protects against attacks that target the stack and/or memory addresses
- Hardware support like ARM's Execute never
 - Supports “permissions” for memory pages
 - Marks memory pages as non-executable

Conclusions

- Multiple surfaces of attack
- Root of trust
- Defence in depth
- Access control
- Audit and monitor
- Make security usable
- Authenticate requests
- Control access
- ...

Data Security

Mihai Ordean
Designing Secure Systems
University of Birmingham

How to prevent a rollback attack?

How to prevent a rollback attack?

- Counter based version control
- Blacklist based version control
- eFuses
- Apple nonce based protocol (i.e. APTicket): random unique value generated at every restore and signed by Apple
- ...

Overview

- Device security
 - Is code on the device vulnerable to exploits ? (e.g. buffer overflows)
 - Is the code authenticated ? (i.e. has not been tampered with)
- **Data security**
 - Is the stored data is accessible to everyone? (e.g. encrypted)
 - Is the stored data authenticated?
- Metadata security
 - What does metadata reveal about data?
 - Can we tamper the metadata?
- Protocol security
 - Is data in transit visible?
 - Can data in transit be tampered with?

Overview

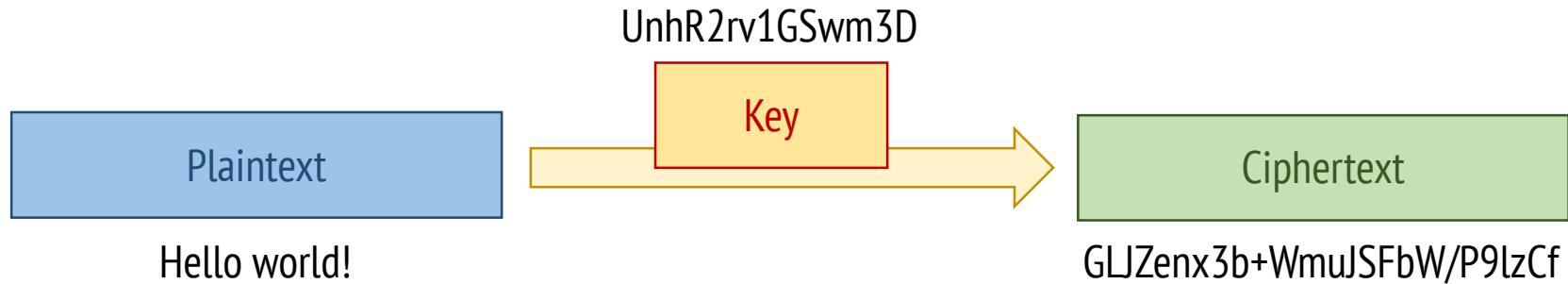
- Data security
 - Protecting the operating system partition
 - Protecting user data
 - Protecting user data in the cloud

Introduction

Symmetric Encryption

Key

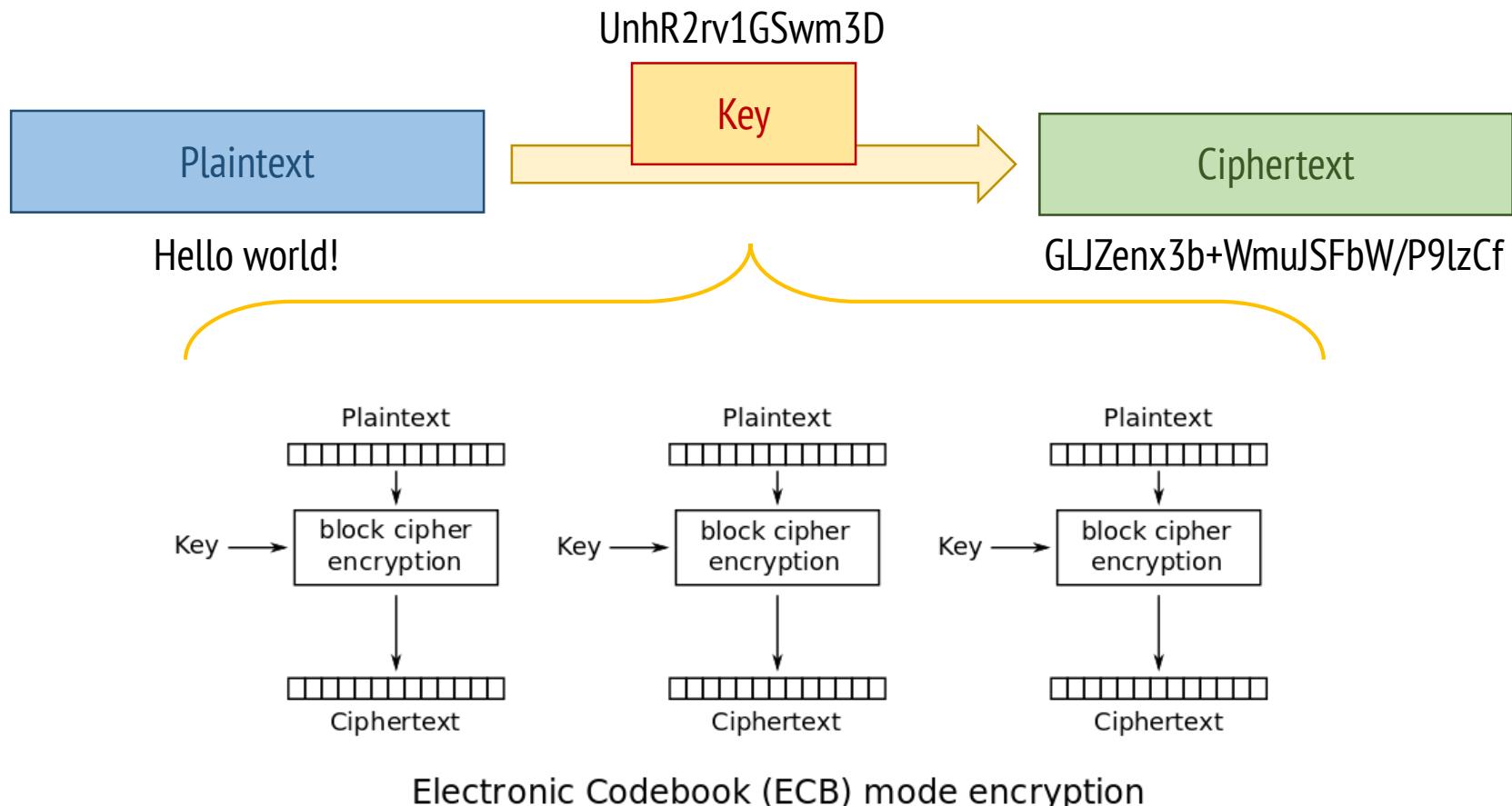
Encryption:



Symmetric Encryption

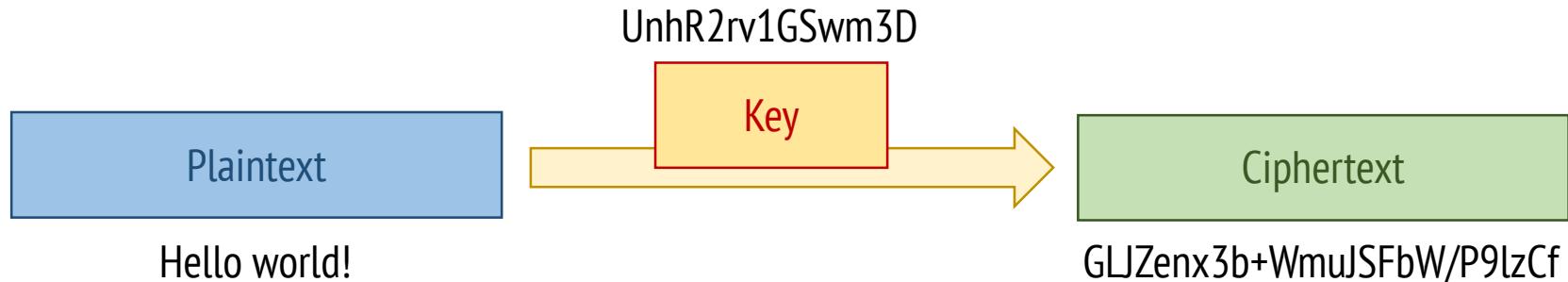
Key

Encryption:

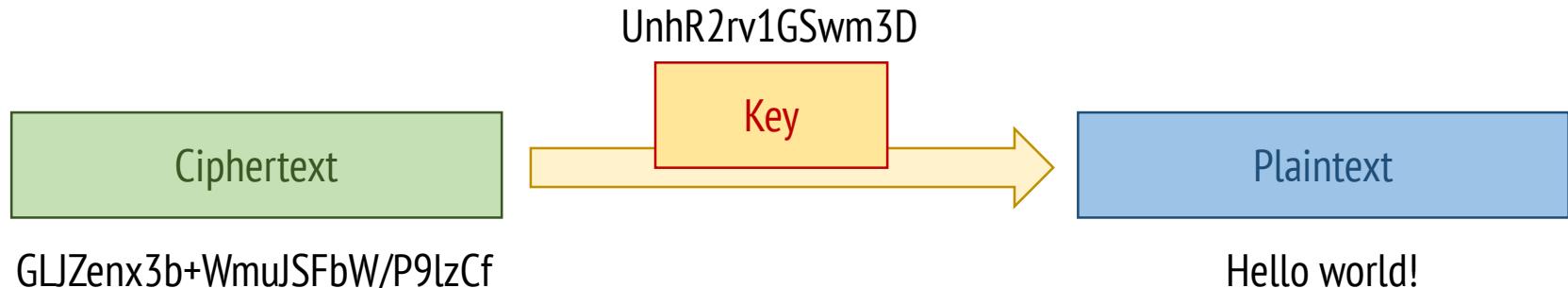


Symmetric Encryption

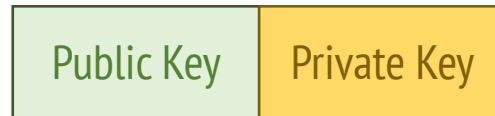
Encryption:



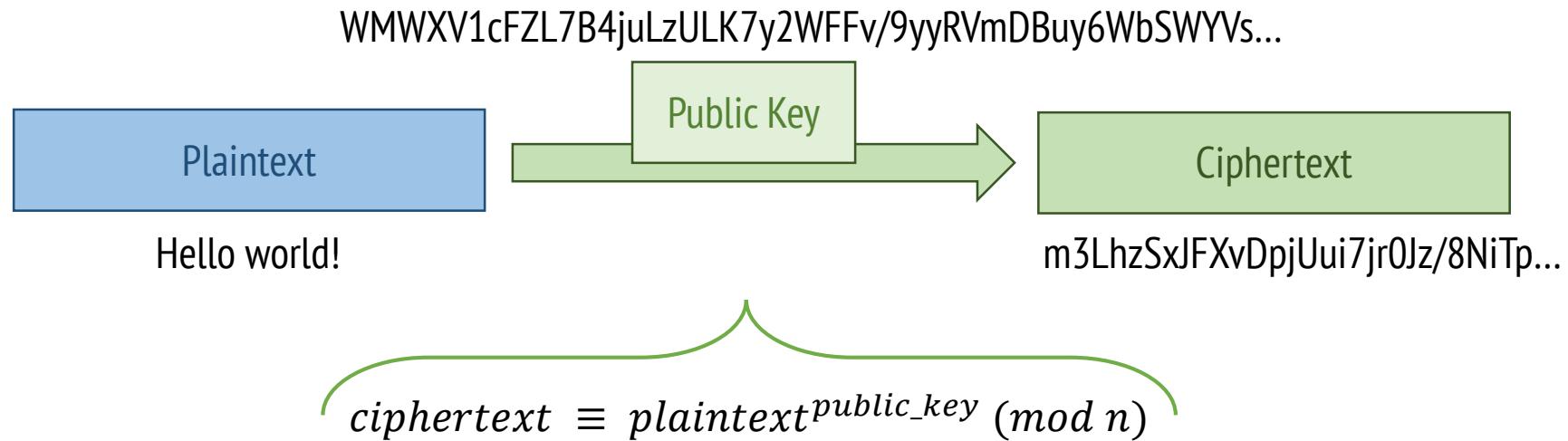
Decryption:



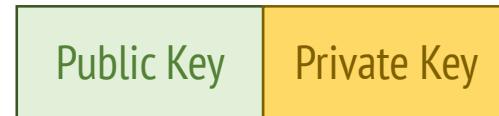
Public key encryption



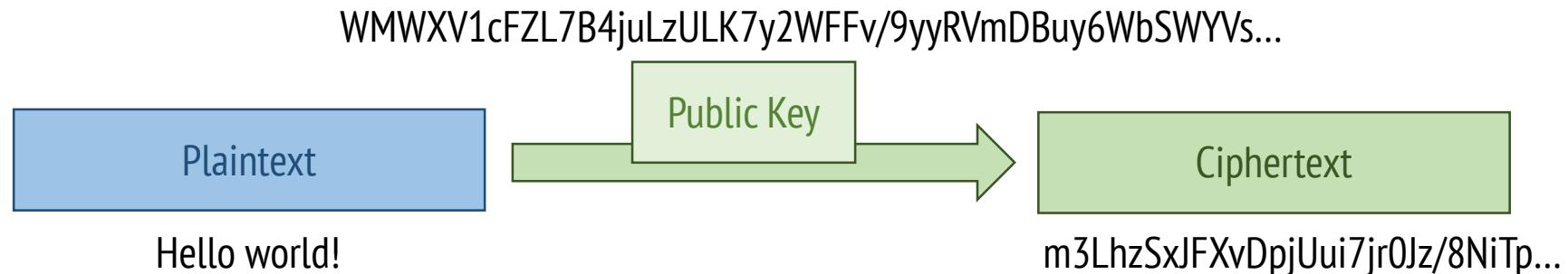
Encryption:



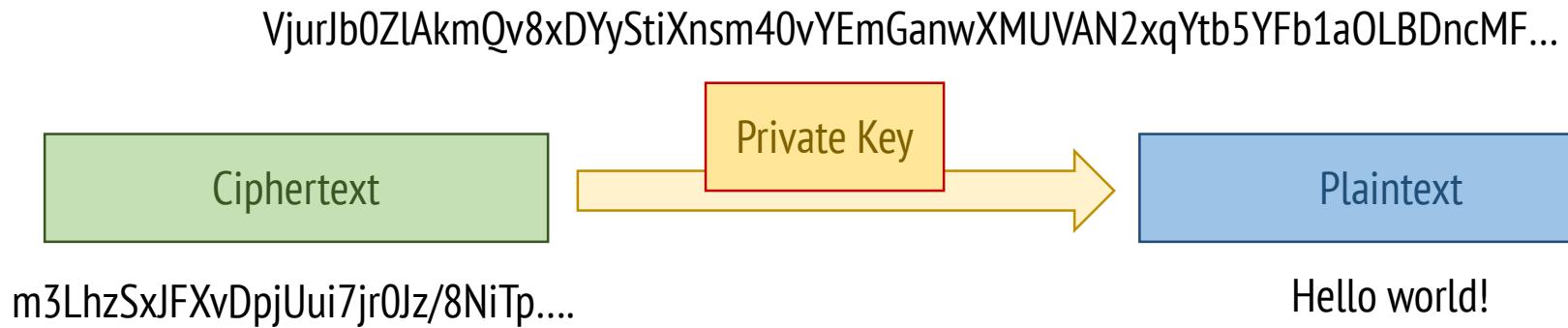
Public key encryption



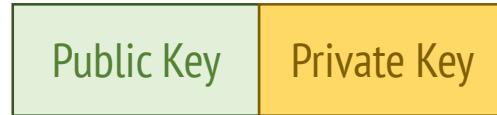
Encryption:



Decryption:



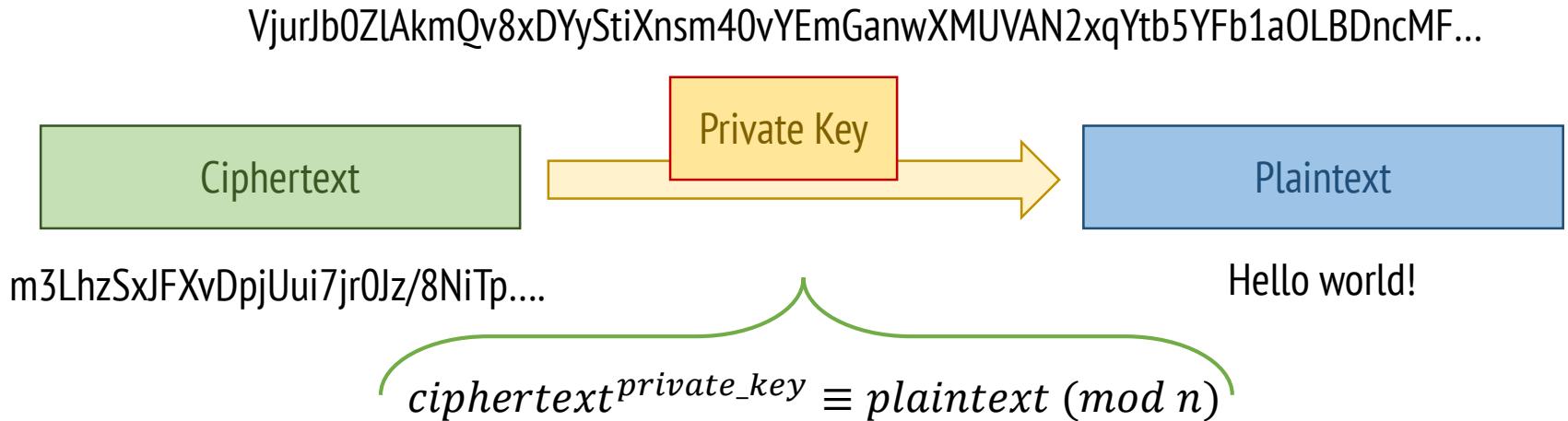
Public key encryption



Encryption:



Decryption:



Public key vs. symmetric key

Public key cryptography

- Anyone can encrypt messages and only the key owner can decrypt the ciphertext
- Public key requires longer keys
- The resulting longer ciphertexts are larger than the plaintext
- ...

Symmetric key cryptography

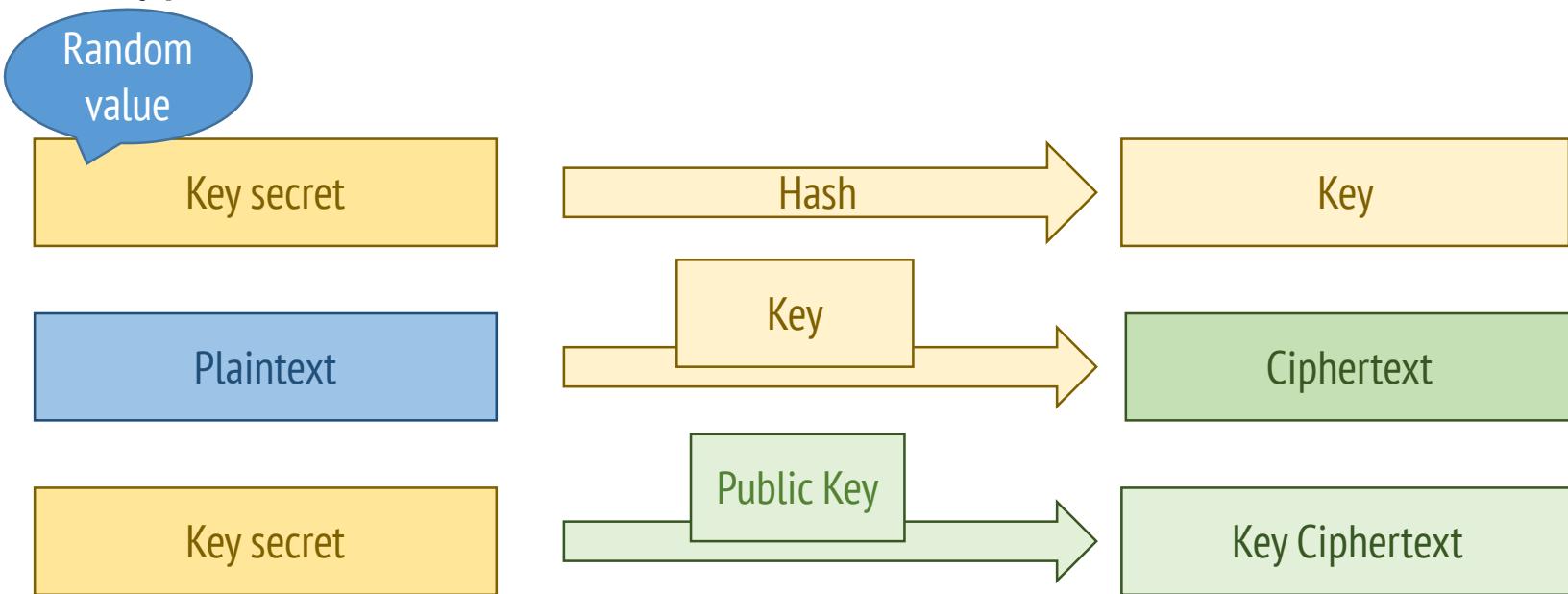
- The encryption/decryption key needs to be shared between parties
- Keys are relatively small
- Resulting ciphertext is about the same size as the plaintext
- ...

Key encapsulation mechanisms (KEM)

KEMs are an efficient method to securely share symmetric keys with the help of public keys.

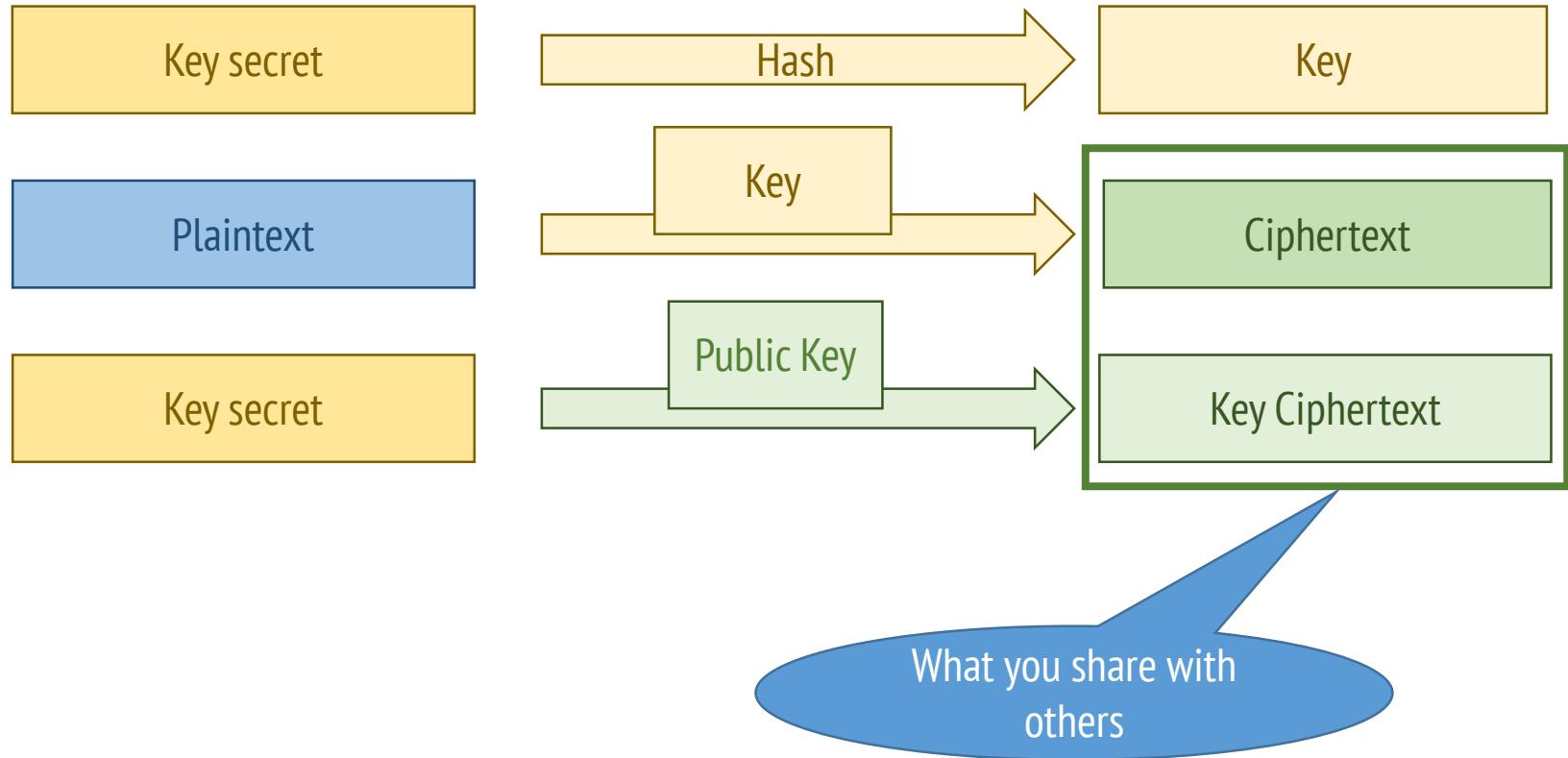
Key encapsulation mechanisms (KEM)

Encryption:



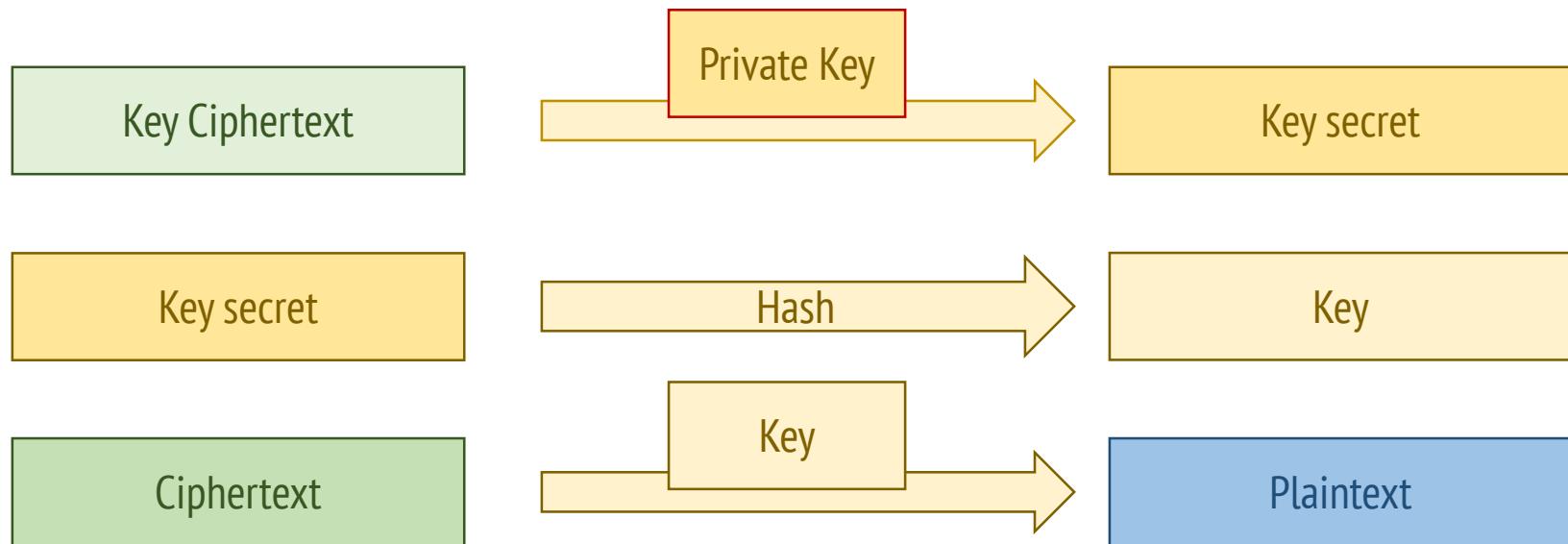
Key encapsulation mechanisms (KEM)

Encryption:



Key encapsulation mechanisms (KEM)

Decryption:



Key encapsulation mechanisms (KEM)

Advantages:

- Symmetric key has good entropy (output of hash function)
- Anybody can encrypt <plaintexts> for the private-key holder
- **Small overhead**

What can we protect?

- **Data at rest** is inactive data that is stored physically in any digital form e.g. files, databases, backups, but also swap
- **Data in use** is data being processed by a CPU or RAM.

What you get/don't get from encryption?

Encryption does:

- Protect data while resting (i.e. your device is off)
- Protect data from apps who don't have access to the keys (assuming sandboxing is used)
- Protect data from if un-authorised repairs are done (or device is stolen)

Encryption does not:

- Prevent data loss (it could actually make it easier).
- Make the system more resilient (quite the opposite: you will be more susceptible to DoS attacks).
- Data that has been decrypted in the volatile memory (RAM).

Challenges

Goal:

Complement the “trusted boot” with data confidentiality.

Challenges:

- 1a. How much information about data should be revealed?
- 2a. Who should be able access this information data?

...

- 1b. How to provide confidentiality to the **system-data** required to boot the system?

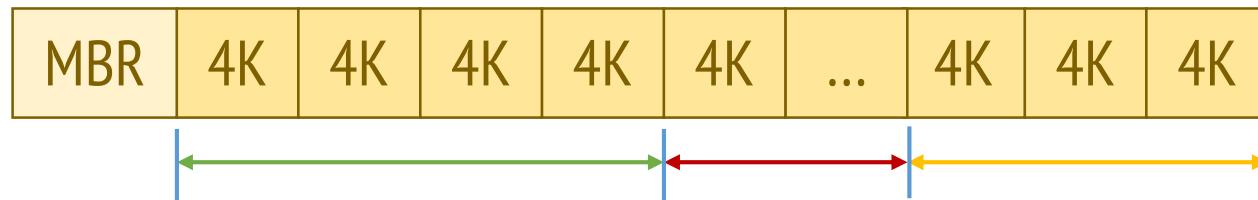
...

Types of data encryption

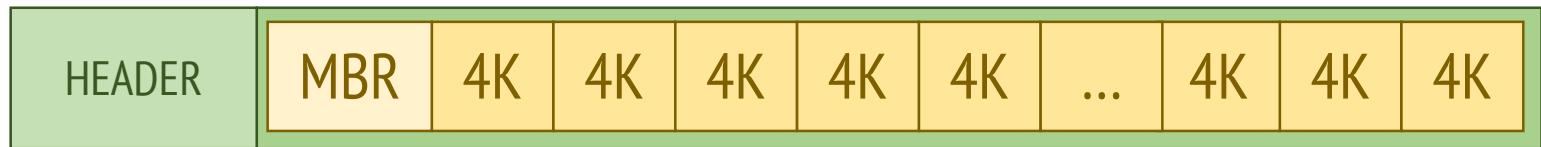
- Disk based
- File based

Disk based encryption

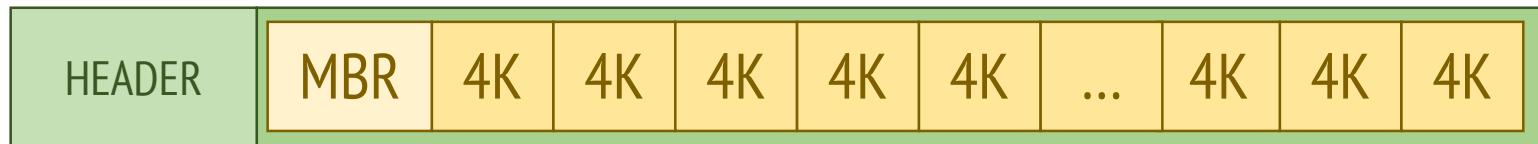
Partition:



Disk based encryption



Disk based encryption



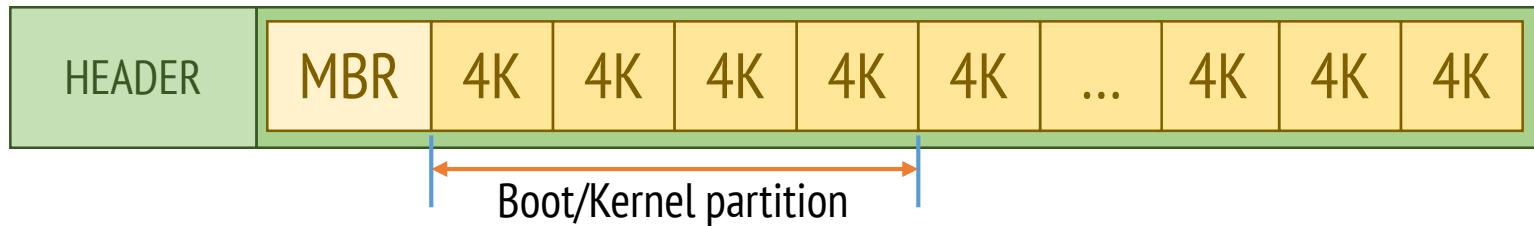
Disk based encryption



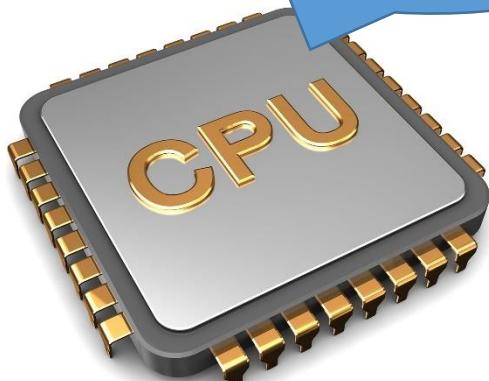
Type: LUKS/BitLocker/etc.
Cipher: AES
IV: `ndef731o4jf92`
Requirements: TPM
Comment: “main partition”
...

A blue curly brace originates from the bottom right corner of the "HEADER" box and curves upwards and to the right, enclosing the five lines of text to its right.

In practice challenges

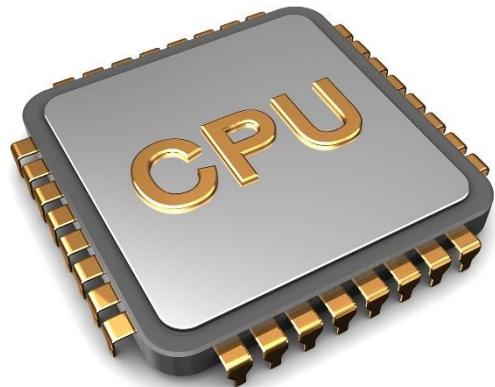
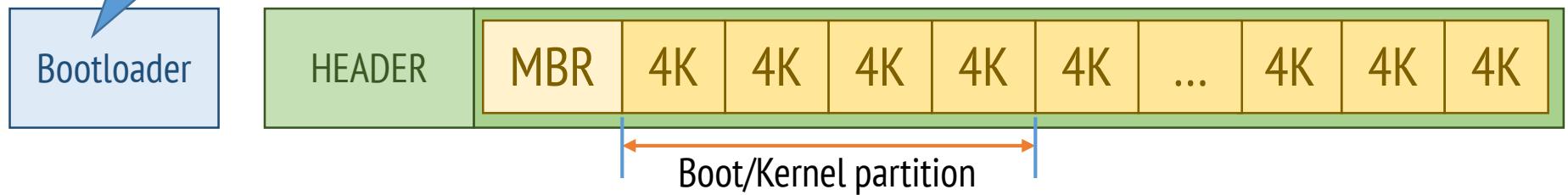


Where is my
decryption
algorithm?

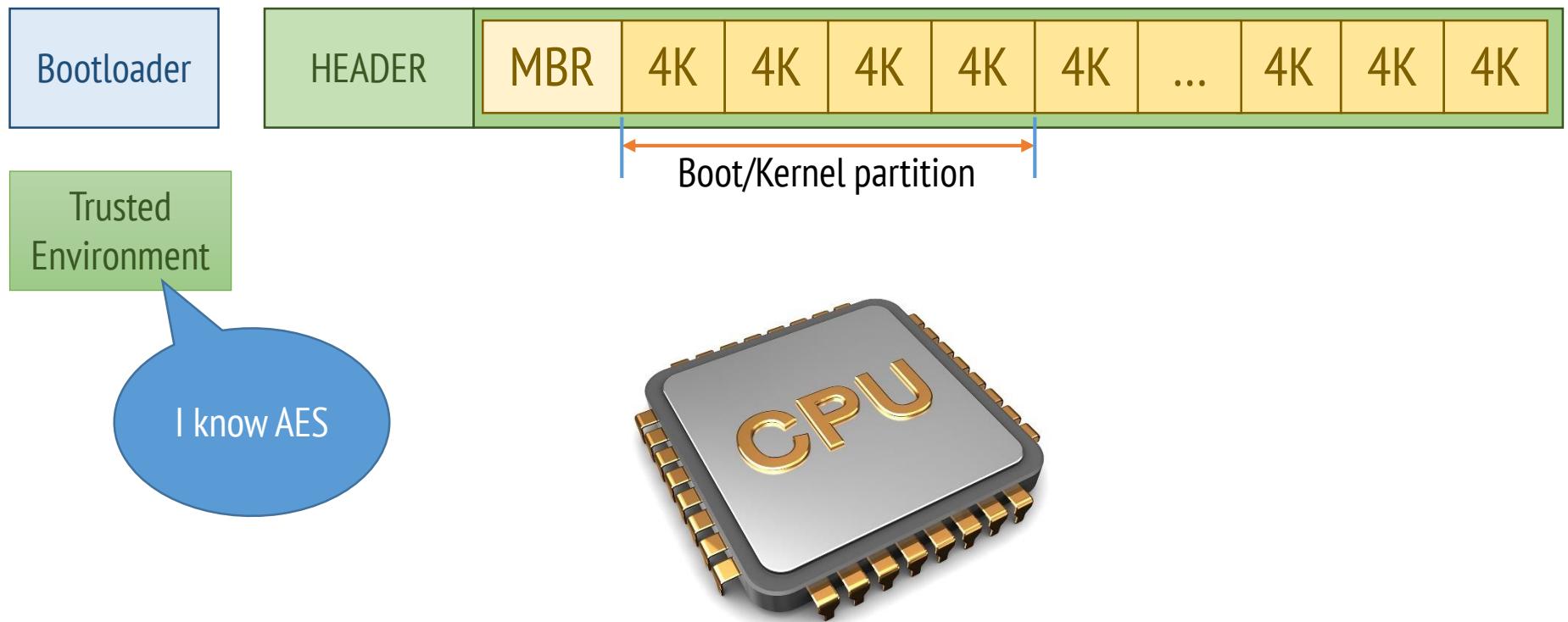


In practice challenges

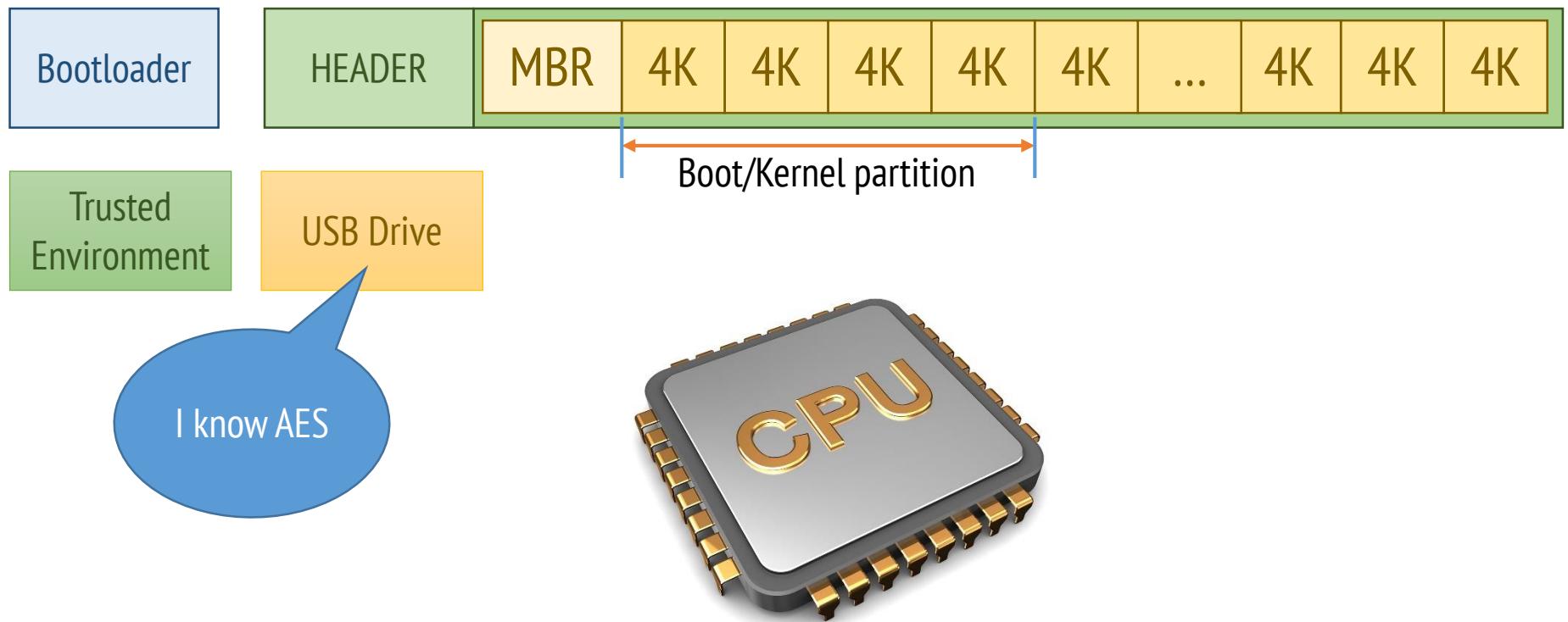
I know AES



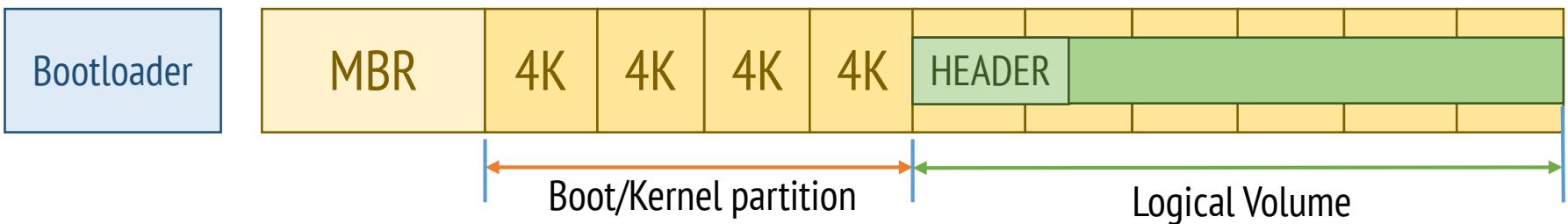
In practice challenges



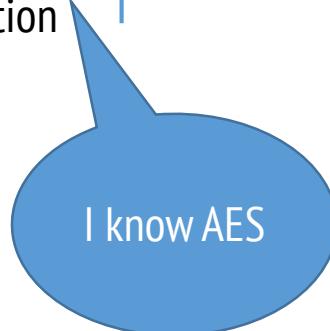
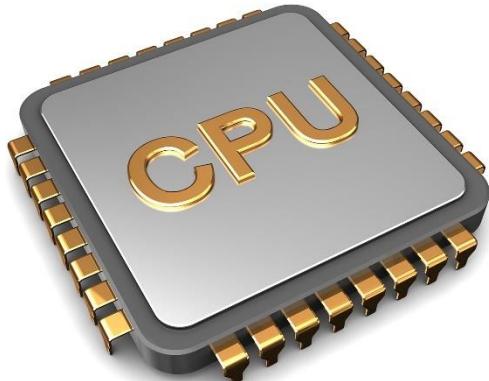
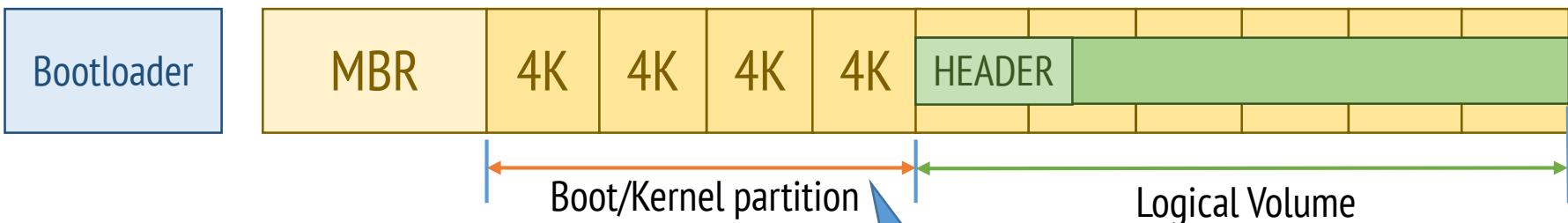
In practice challenges



In practice challenges



In practice challenges



Hold on, are we not missing
something important?

Hold on, are we not missing
something important?

Right... the **encryption key**.

Storing the key

On a USB stick:

- Easy
- Requires USB to be accessible to the system
- Vulnerable to stealing

In the TPM (or TEE)

- More difficult to set up
- Transparent
- Protected from stealing

Storing the key

On a SmartCard:

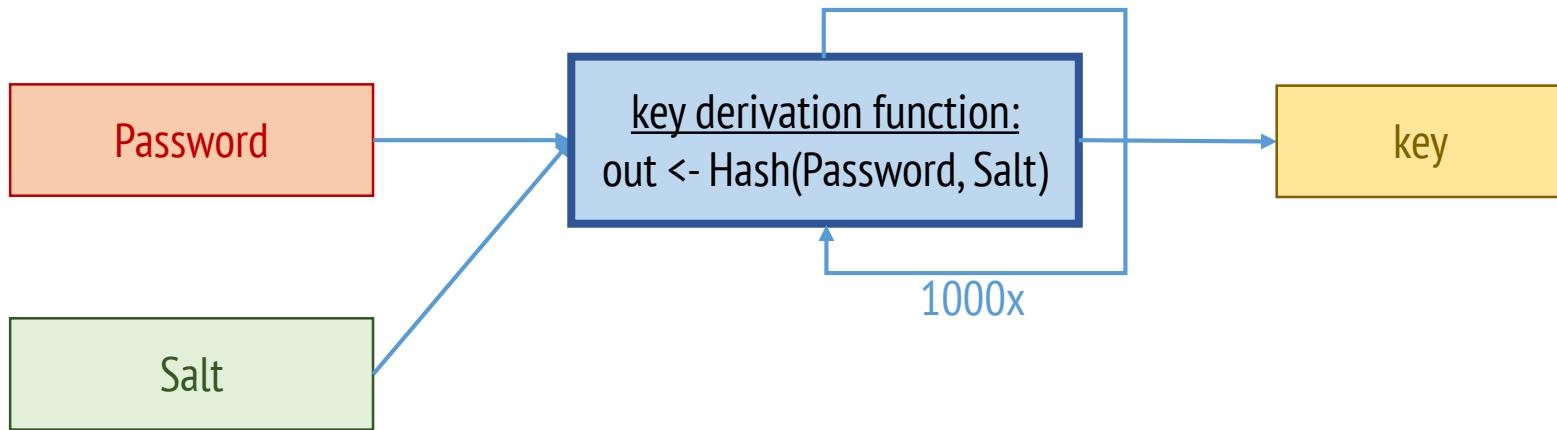
- Difficult to set up (e.g. requires special hardware)
- Requires presence of the card
- Protected from stealing

Deriving the key from a password

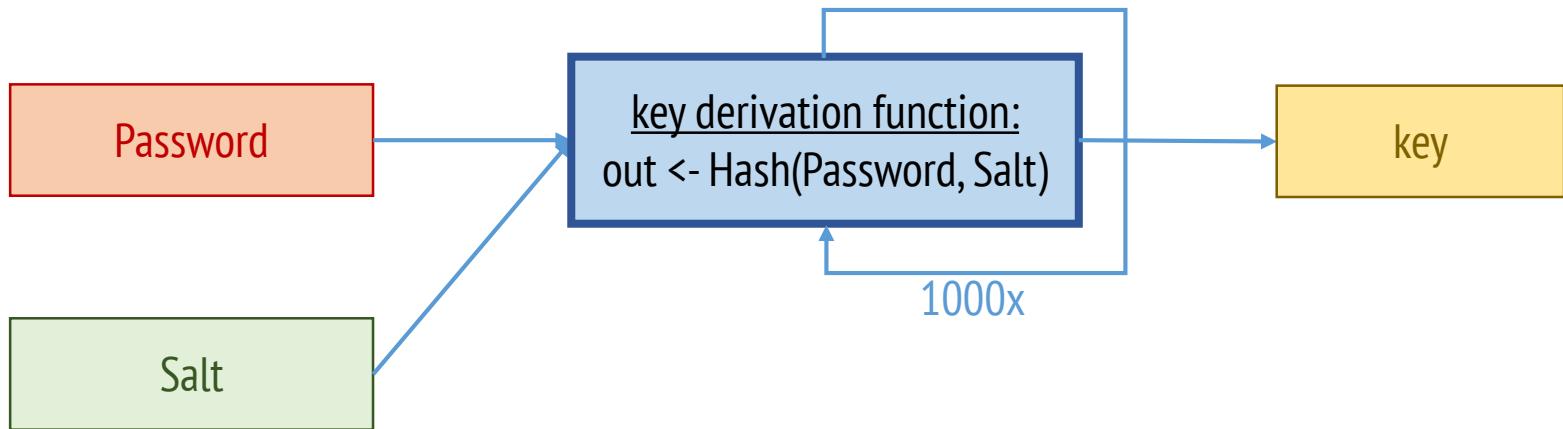
Challenge:

- Human generated passwords have low entropy!

Key derivation functions



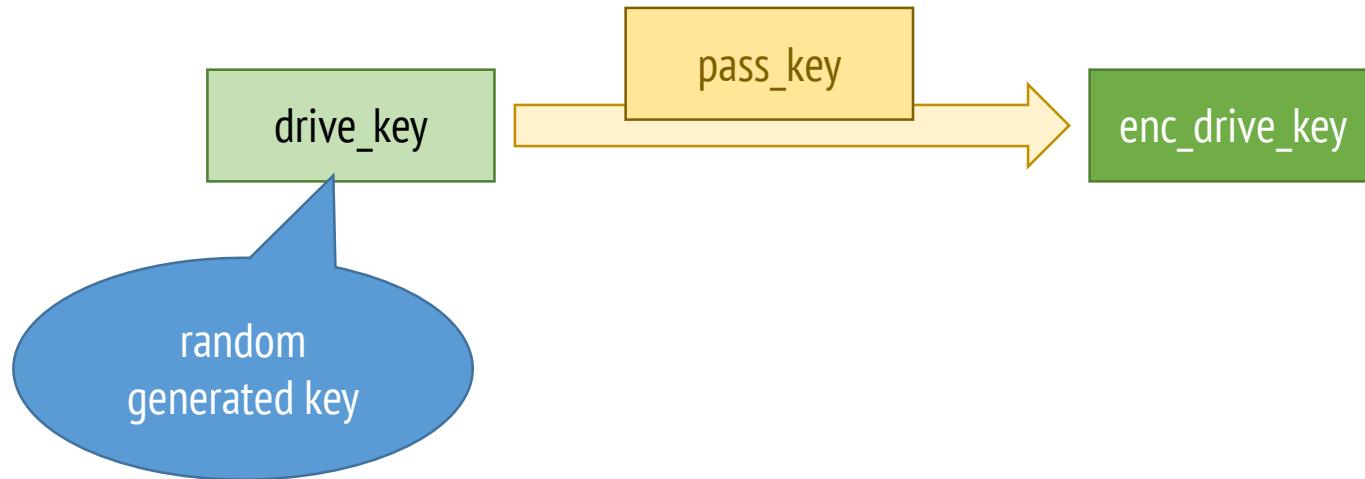
Key derivation functions



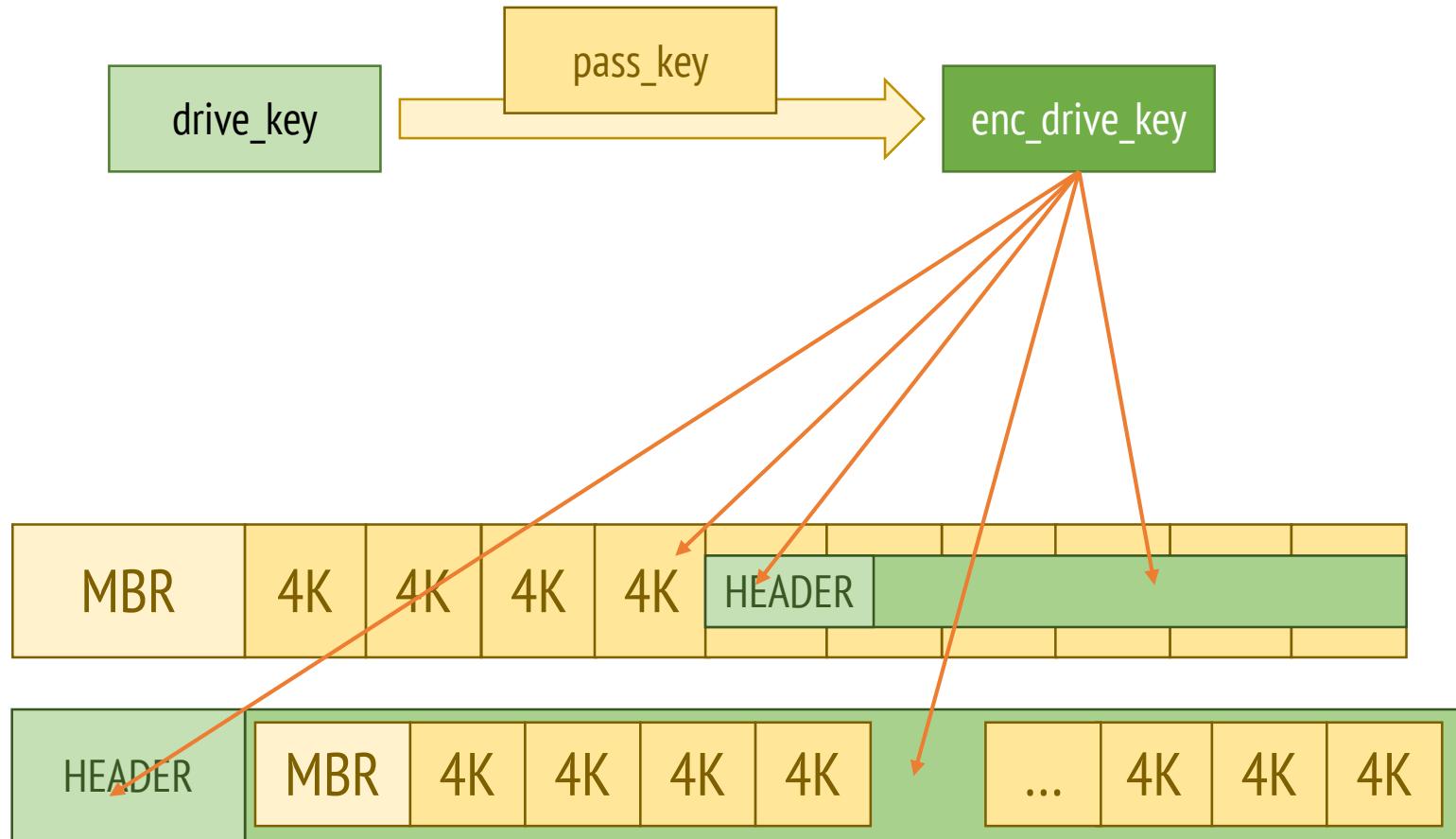
The key has sufficient entropy!

What if I want to change my password?
Do I have to re-encrypt the whole drive?

Key derivation functions



Key derivation functions



Decryption

Derived key decryption process:

- Load enc_drive_key from the drive
- Derive key from password
- Decrypt key
- Decrypt drive

Decryption

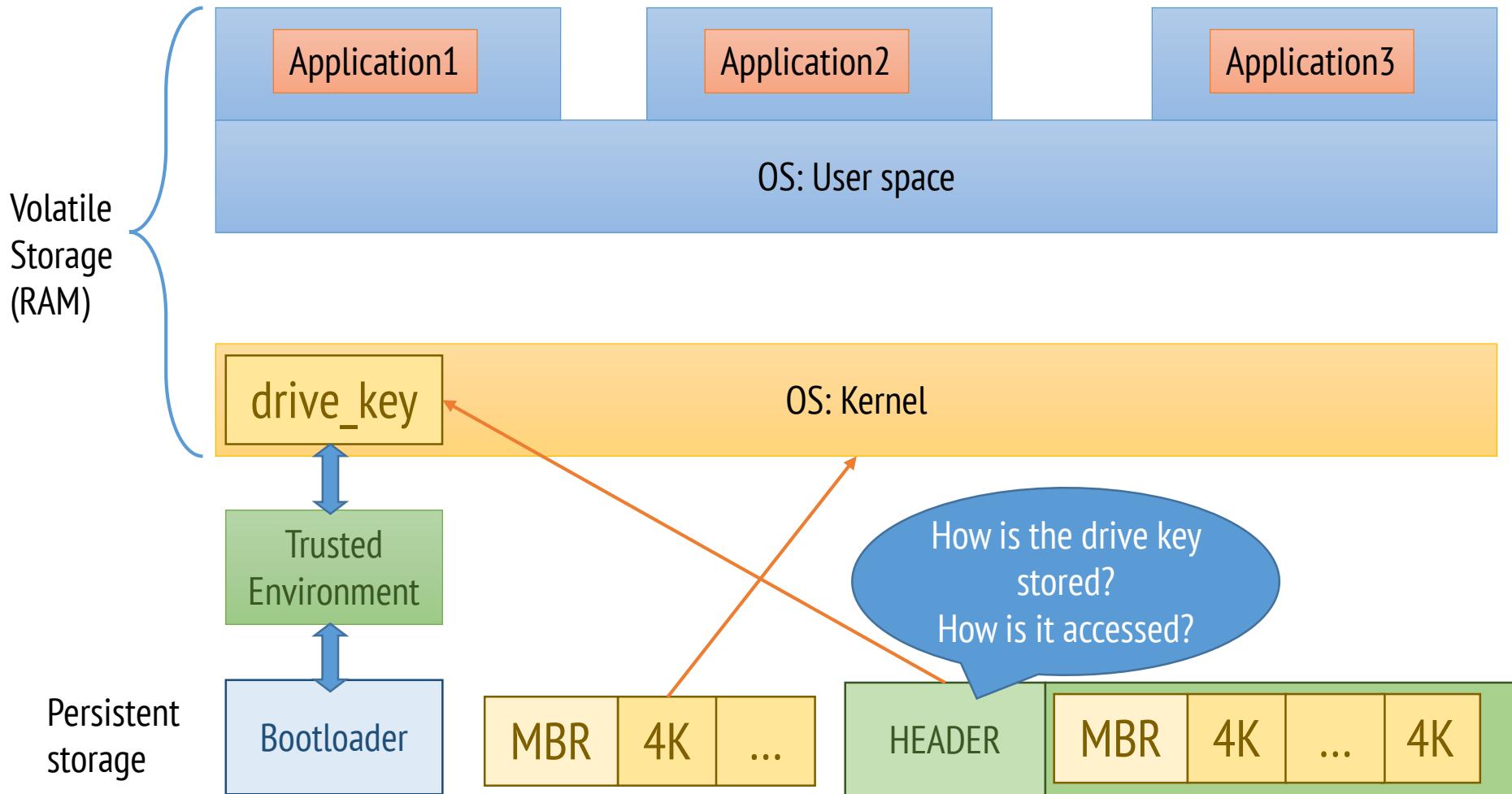
Stored key decryption:

1. Load key from USB
2. Decrypt drive

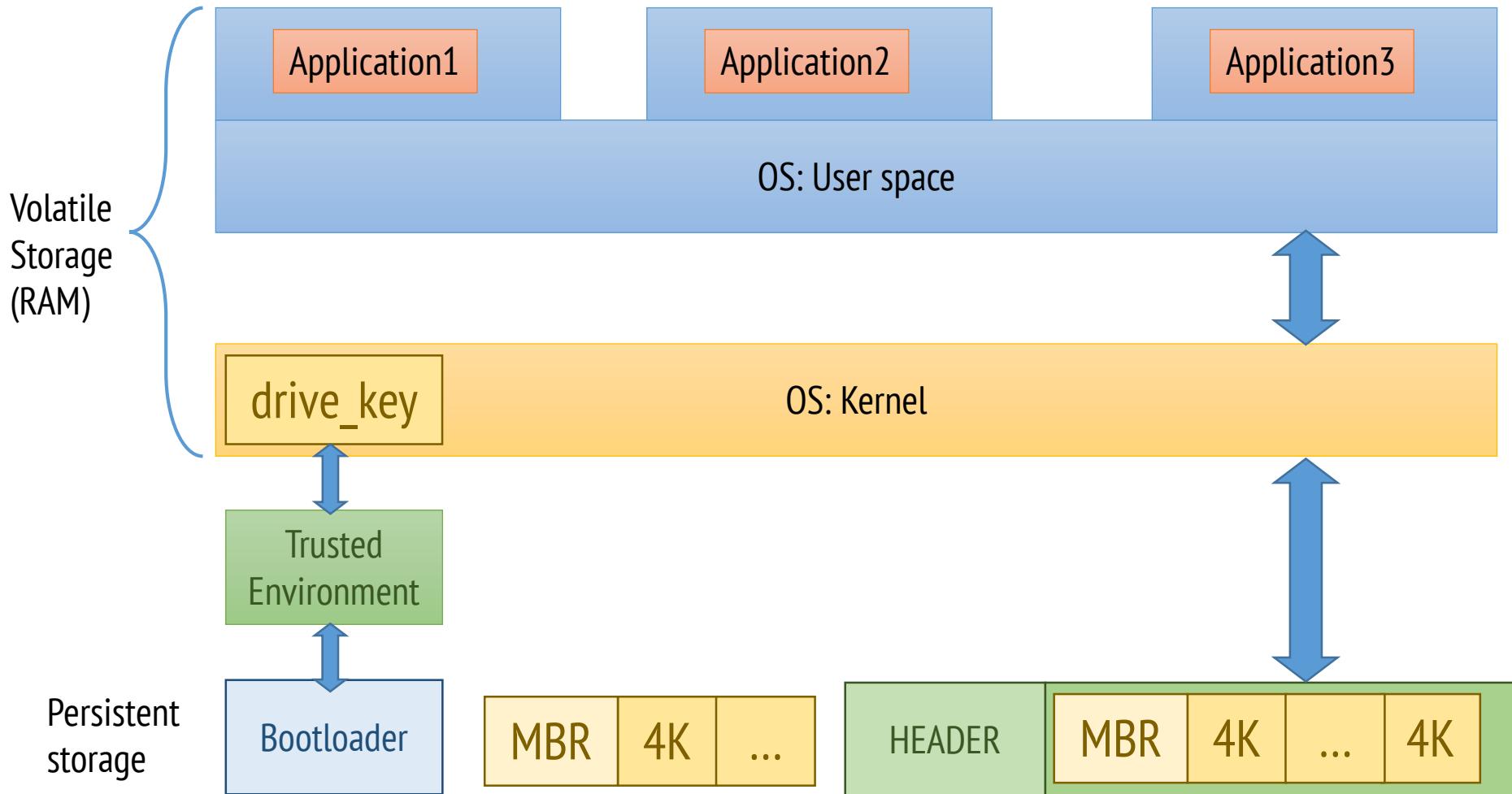
Or:

1. Load key from enc_drive_key
2. Load key from USB/SmartCard/TEE/TPM
3. Decrypt enc_drive_key
4. Decrypt drive

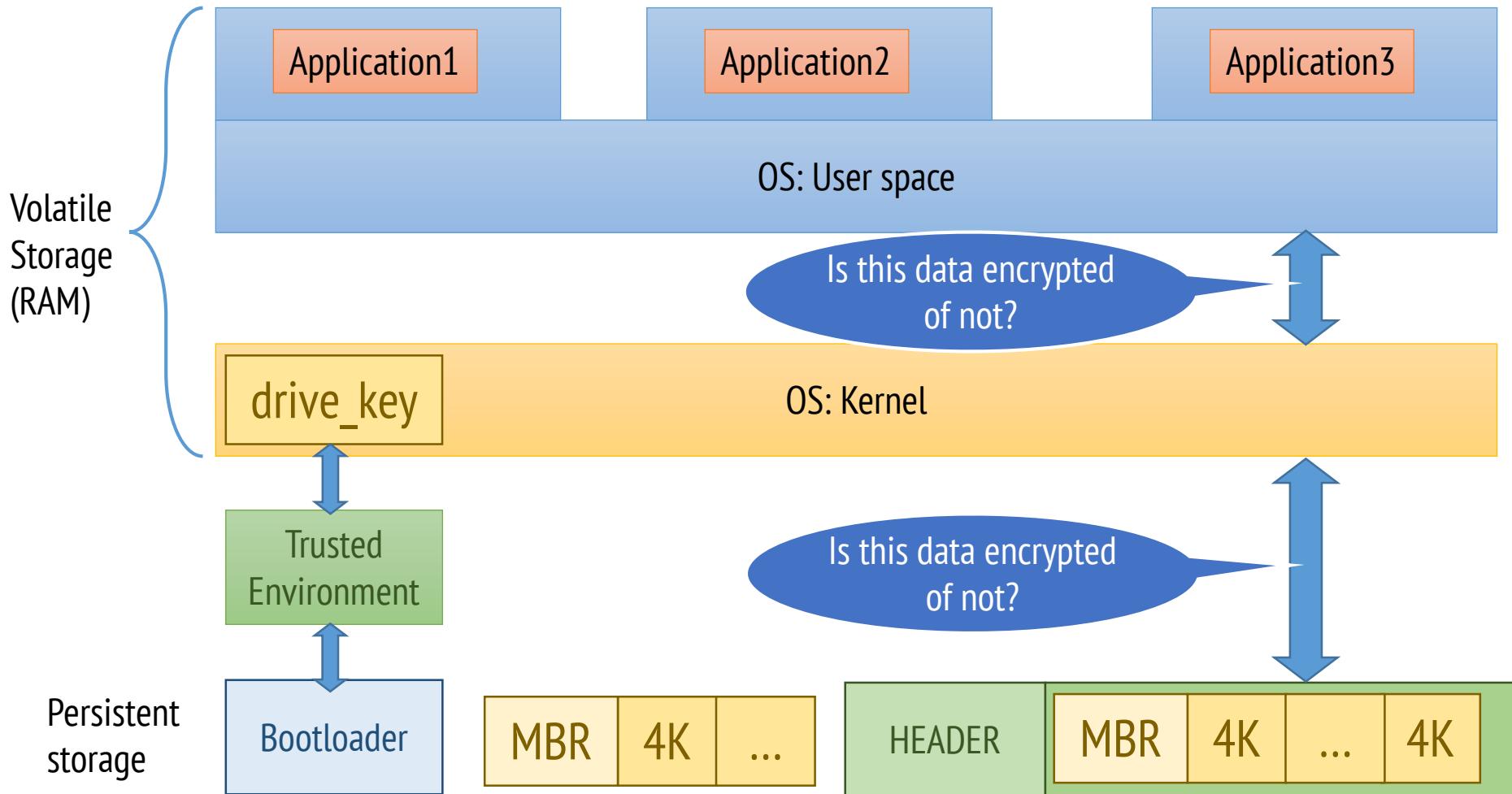
Usage



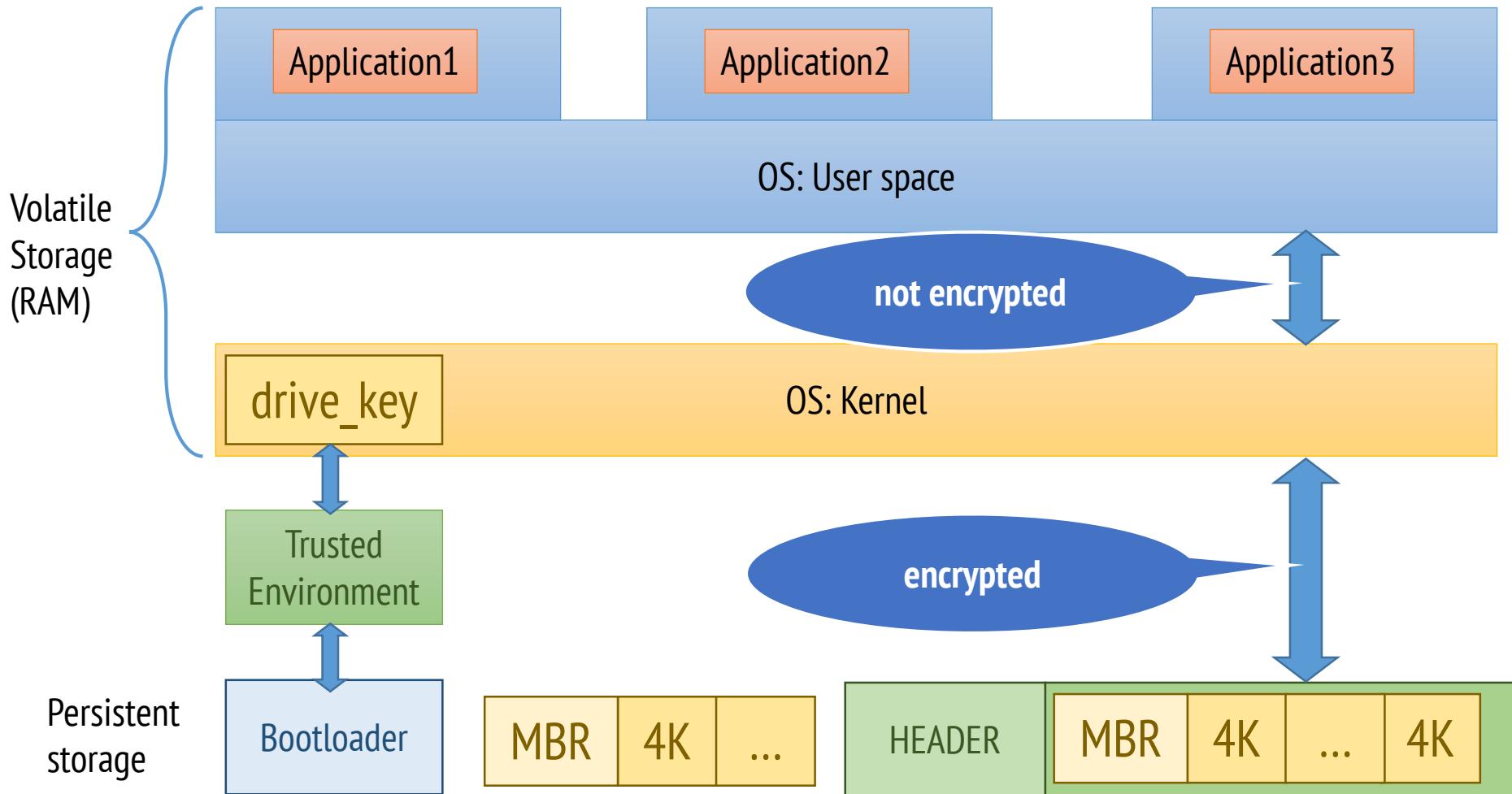
Usage



Usage



Usage



Real implementations

BitLocker

- **Transparent operation mode:** uses the capabilities of TPM hardware to provide for a transparent user experience by sealing it on the TPM chip.
- **User authentication mode:** the user has to authenticate before decryption starts.
- **USB/ smartcard key mode:** the user must insert a USB device that contains a the key into the computer.

BitLocker keys

Keys:

- Data Encryption Key (DEK): the drive generates the DEK and it never leaves the device. It is stored in an encrypted format at a random location on the drive. If the DEK is changed or erased, data encrypted using the DEK is irrecoverable.
- Authentication Key (AK): the key used to unlock data on the drive. A hash of the key is stored on drive and requires confirmation to decrypt the DEK.

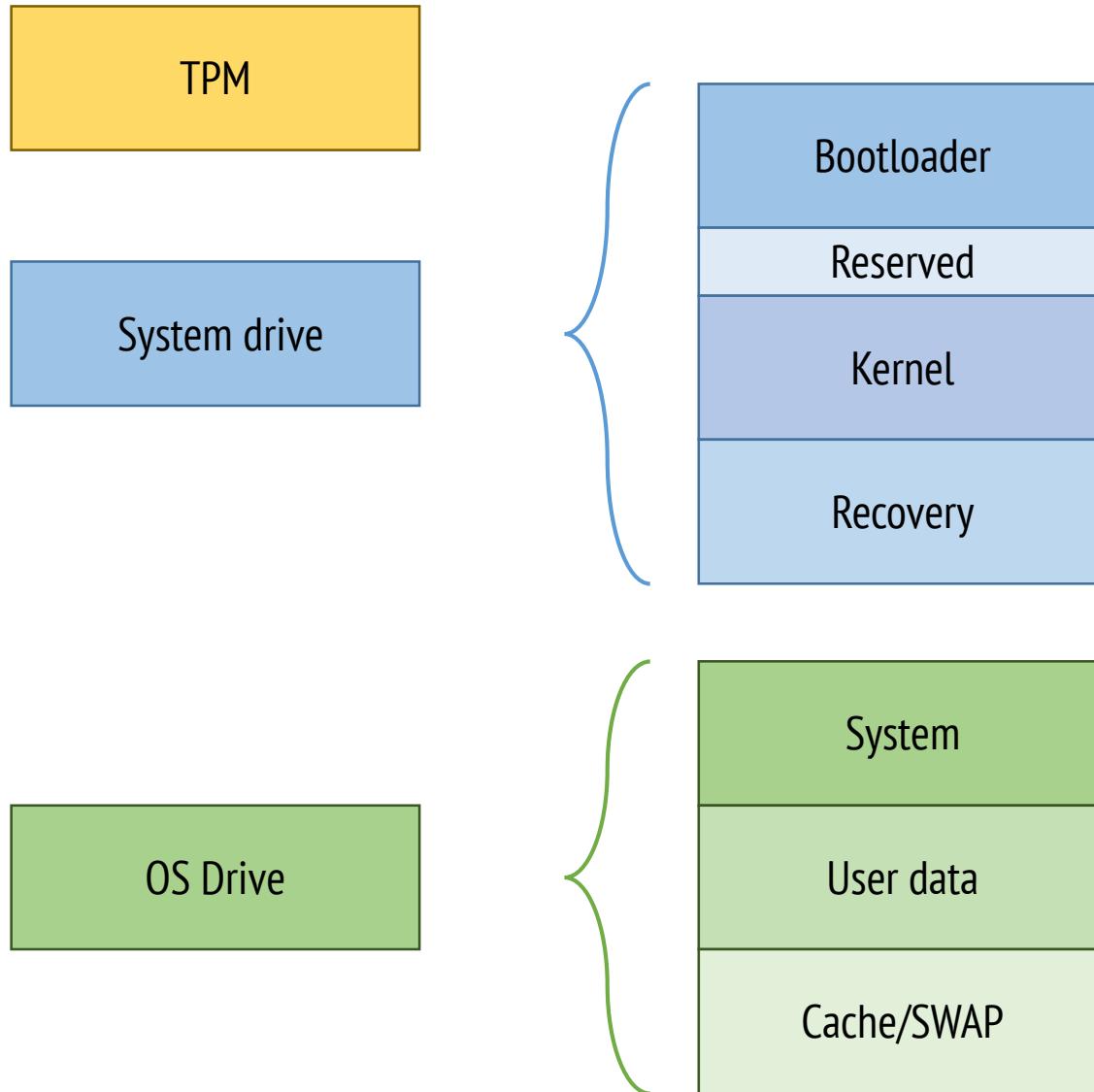
BitLocker

TPM

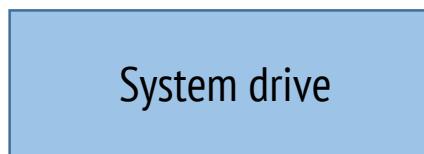
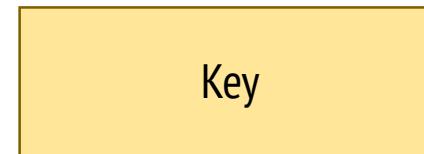
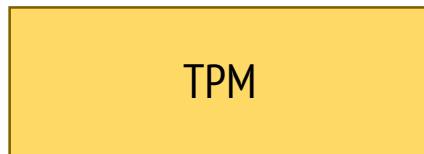
System drive

OS Drive

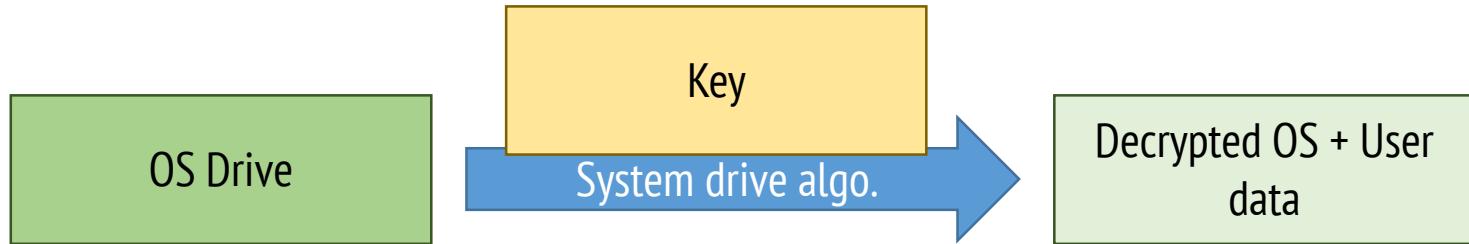
BitLocker



BitLocker



BitLocker



TPM extra protection

The **key** is sealed inside the TPM's memory

The **key** is only released if early boot files appear to be unmodified

Principles used

- Simple design
- Assume secrets not safe (e.g. the key is sealed inside the TPM)
- Make security usable (e.g. transparent BitLocker mode vs User authentication mode)
-

Android 5.0 (with Linux kernel & dm-crypt)

dm-crypt:

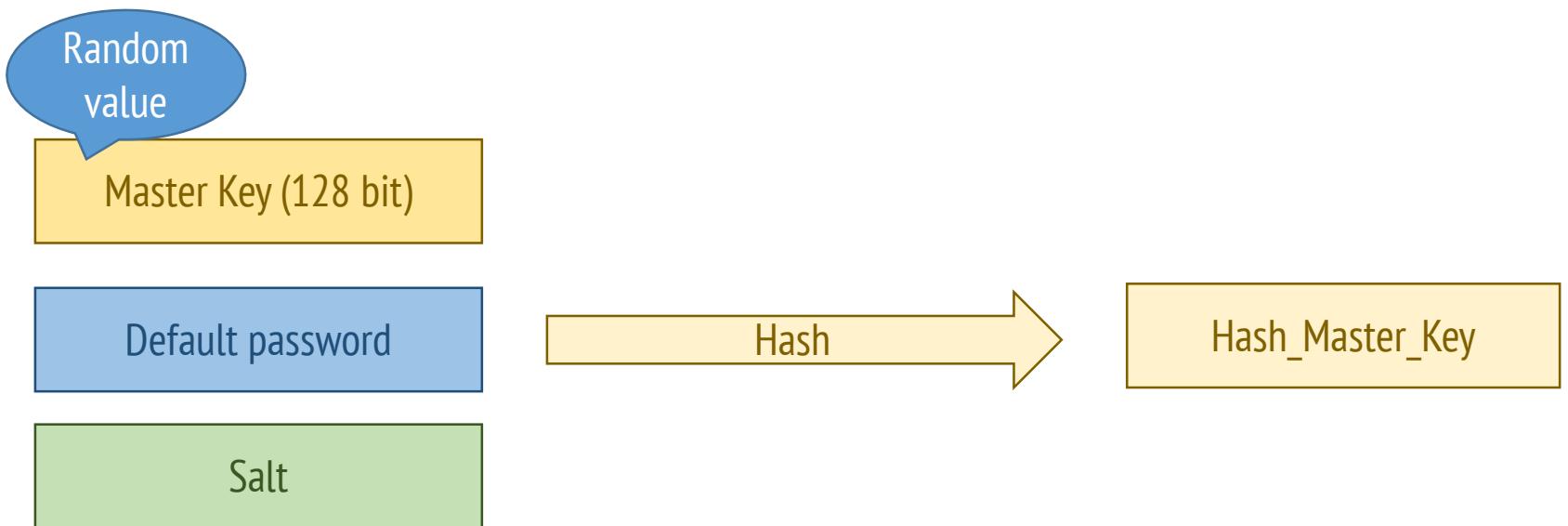
- Kernel module (runs in kernel space)
- Provides transparent disk encryption
- Supports the kernel only keys (i.e. logon keys)
- Uses cryptographic routines from the kernel's Crypto API

Android 5.0 (with Linux kernel & dm-crypt)

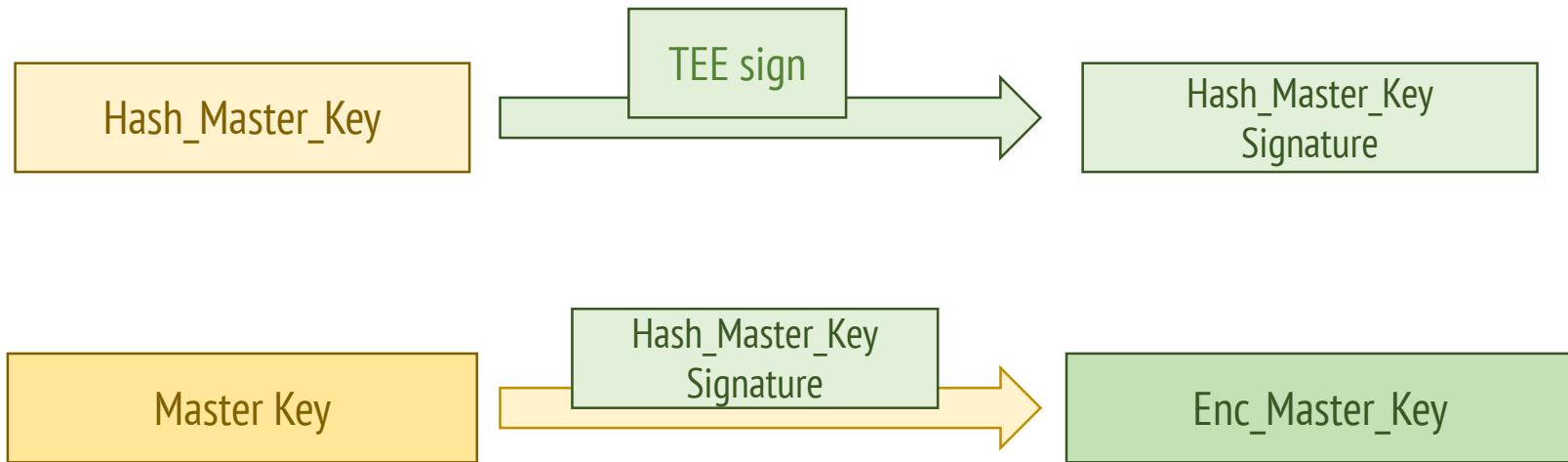
Android user authentication methods:

- default
- PIN
- password
- pattern

Android 5.0 (with Linux kernel & dm-crypt)



Android 5.0 (with Linux kernel & dm-crypt)

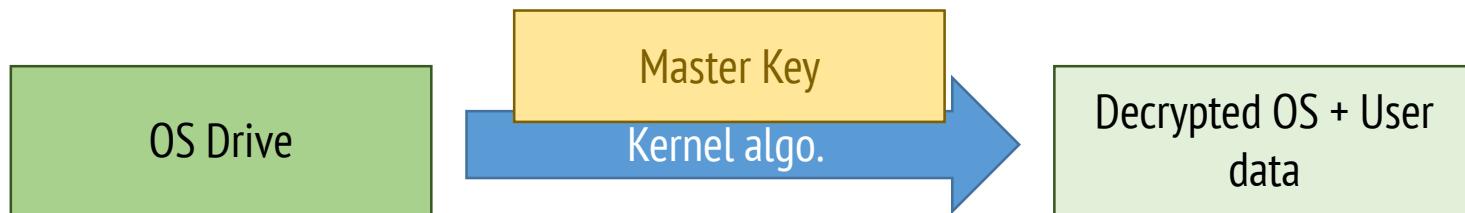


Android 5.0 (with Linux kernel & dm-crypt)

Which key am I using to decrypt my data?

Android 5.0 (with Linux kernel & dm-crypt)

Which key am I using to decrypt my data?



Android 5.0 (with Linux kernel & dm-crypt)

What happens to the Master Encryption key when I change my password, i.e. Default password?

Android 5.0 (with Linux kernel & dm-crypt)

What happens to the Master Encryption key when I change my password, i.e. Default Password?

Regenerate

Hash_Master_Key
Signature

using

Master Key

New password

Salt?

Does the salt change?

Android 5.0 (with Linux kernel & dm-crypt)

What happens to the Master Encryption key when I change my password, i.e. Default Password?

Regenerate using

Hash_Master_Key
Signature

Master Key

New password

New salt

Individual research

What are the security implications of using a public, hardcoded and known value for the “default password” in Android 5.0?

How does it compare to not using encryption at all?

Explain your answers by referring to security aspects (confidentiality, authentication...), and difficulty of use.

Advantages/disadvantages of full disk encryption

Full disk encryption

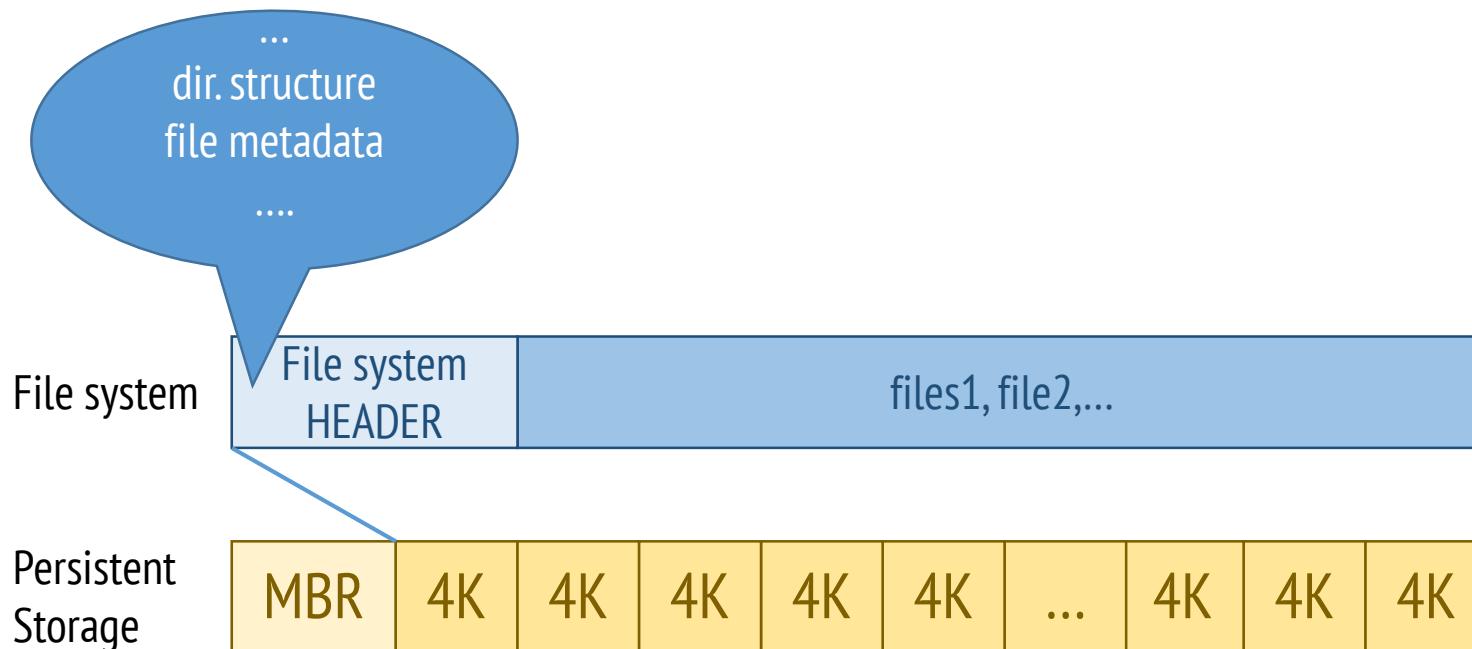
- Simple design: generally only one key is used.
- Protects filesystem meta data e.g. directory structure, file names, modification timestamps.
- If the key is compromised, the attacker has access to all files.

File based encryption

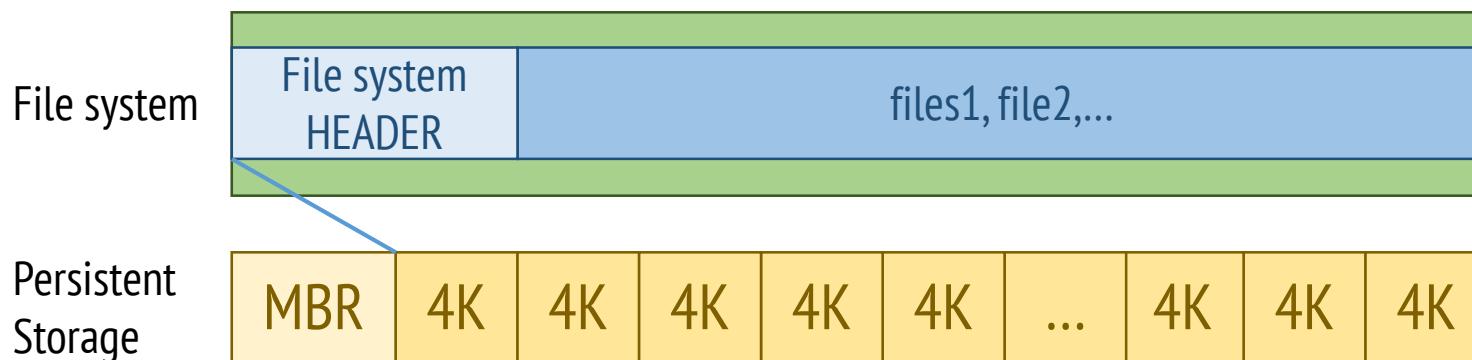
Components

- File contents
- File metadata
- Memory storage
- Disk storage
- Access control (type, user)

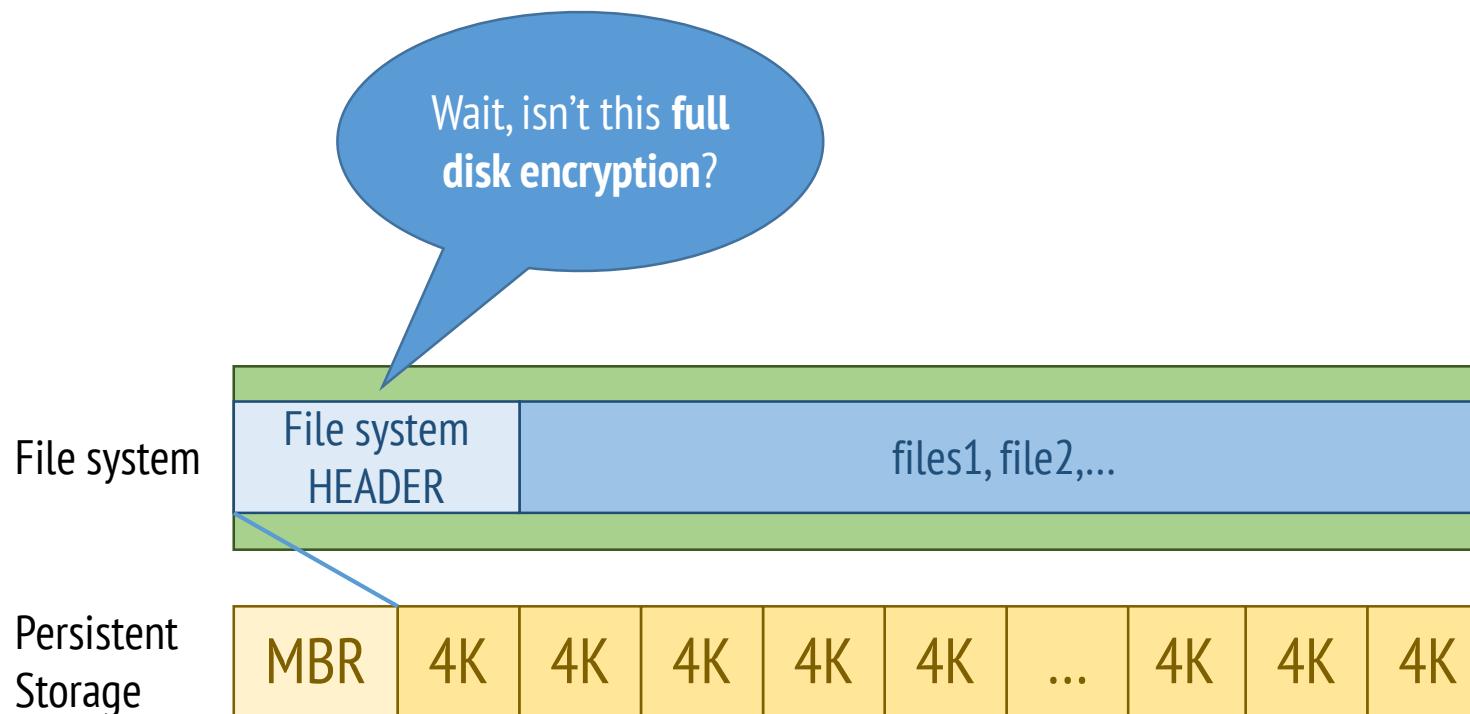
File based encryption



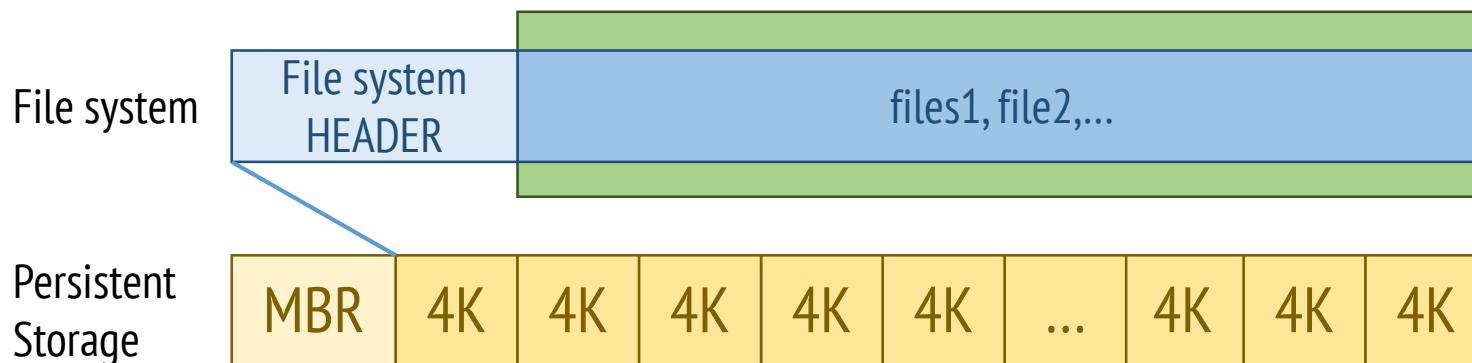
File based encryption



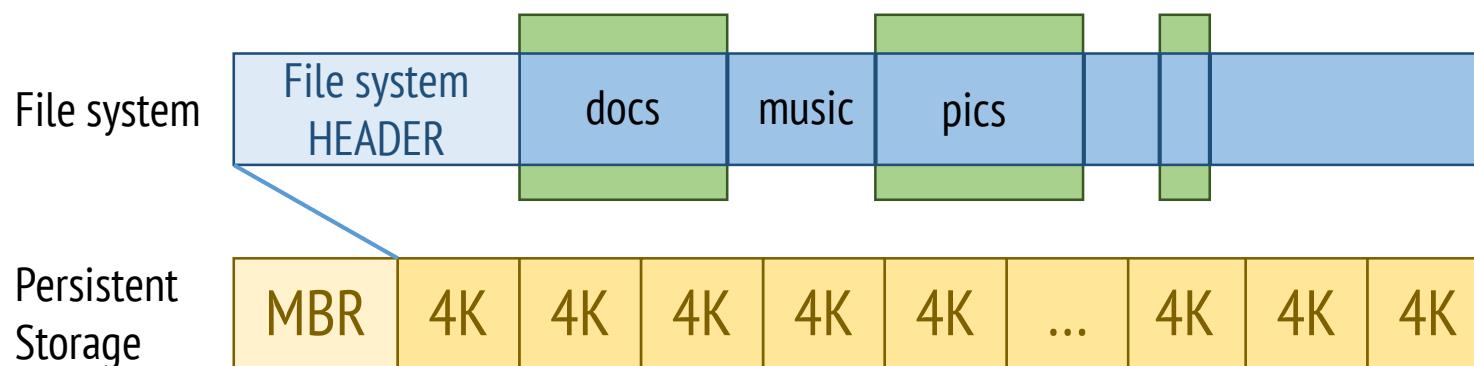
File based encryption



File based encryption



File based encryption

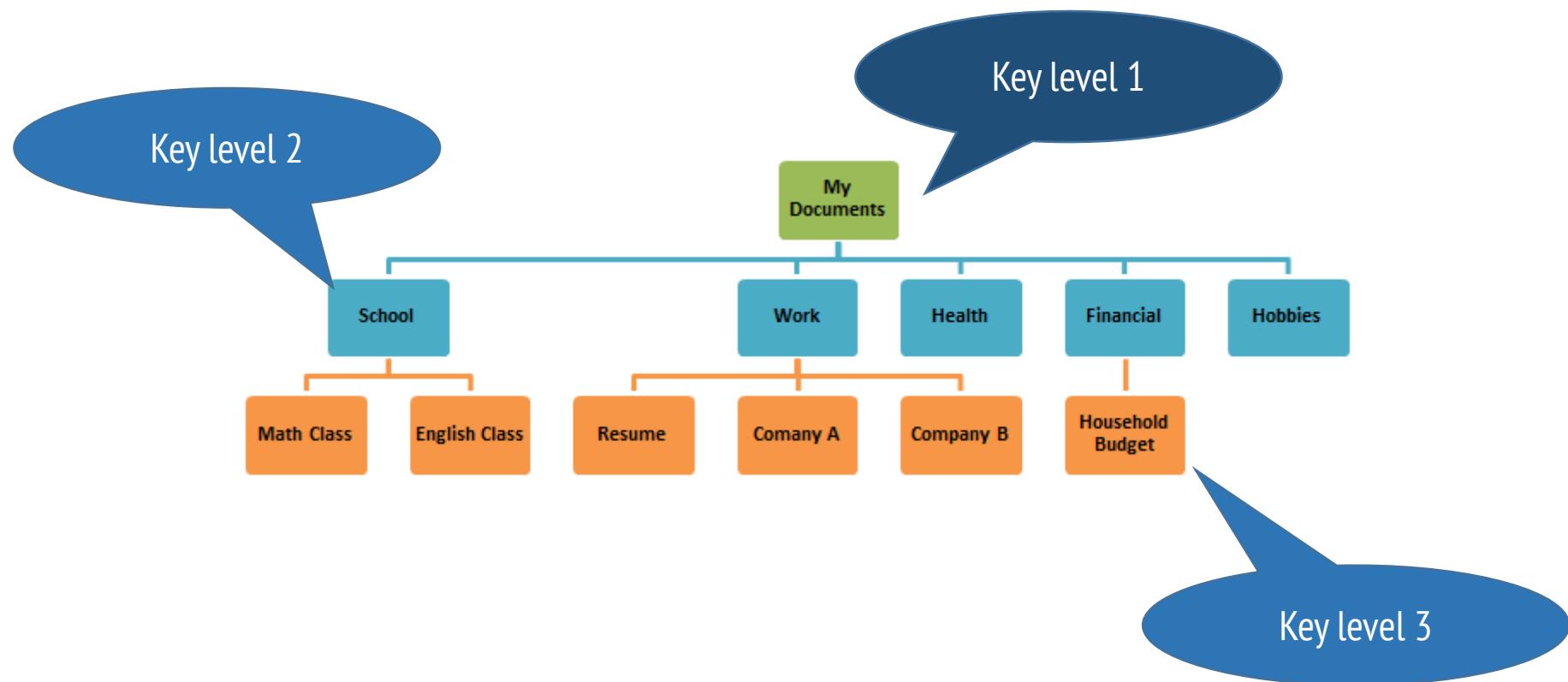


This is pretty cool. How do we do it?

Try to map a key structure to the file system structure.

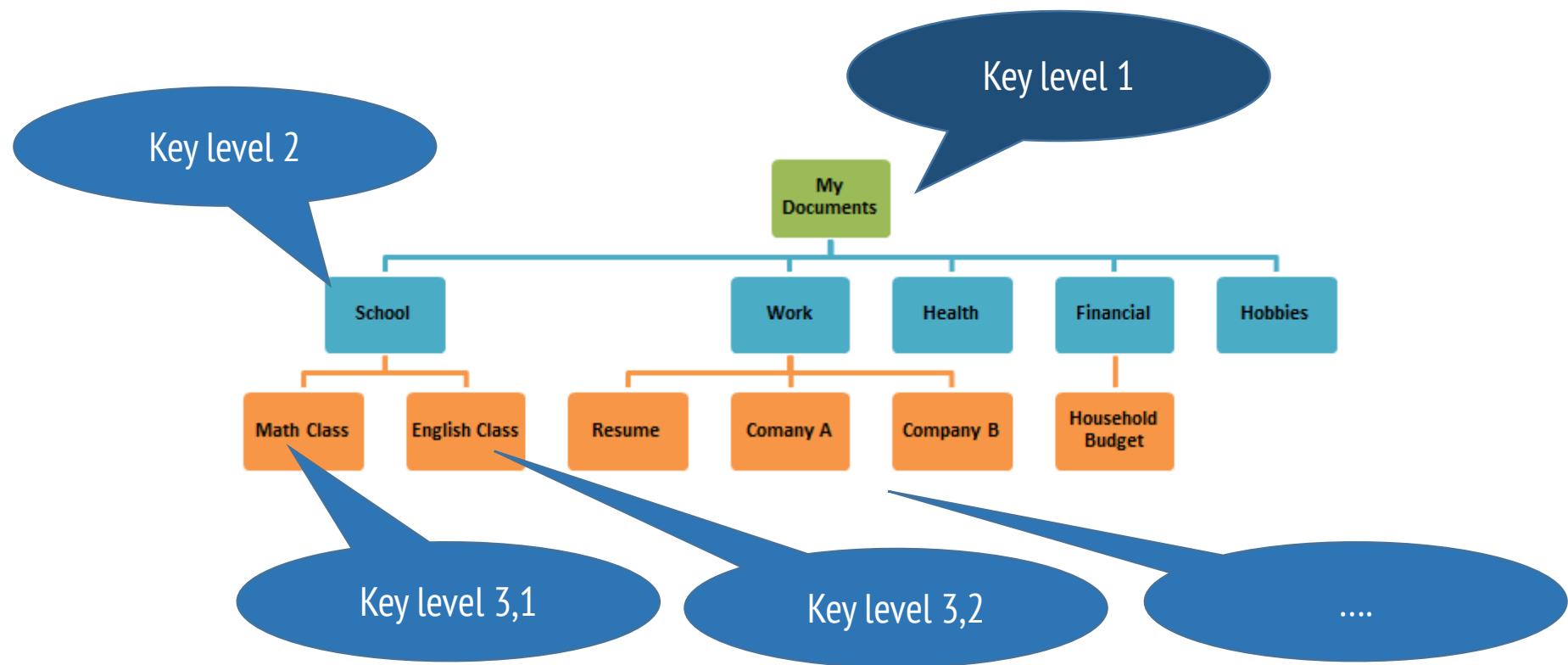
This is pretty cool. How do we do it?

Try to map a key structure to the file system structure.



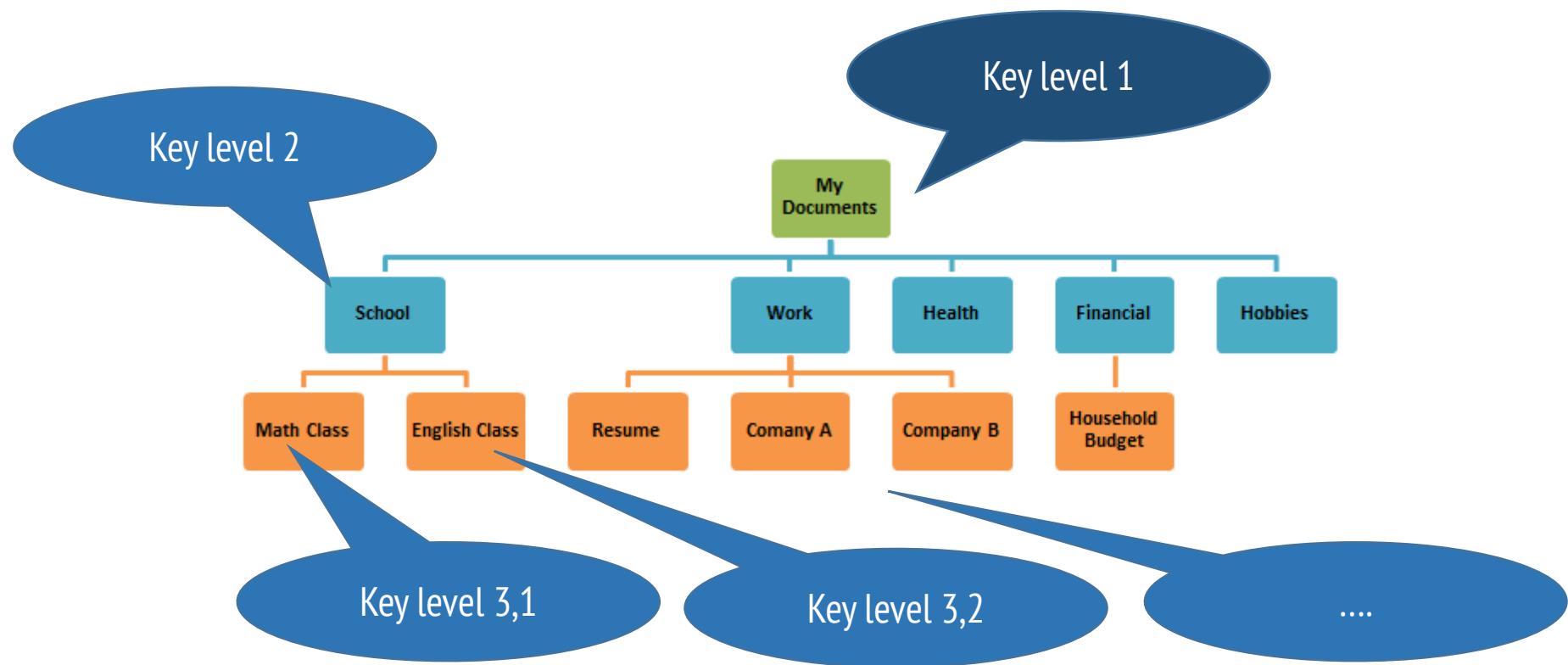
This is pretty cool. How do we do it?

Try to map a key structure to the file system structure.



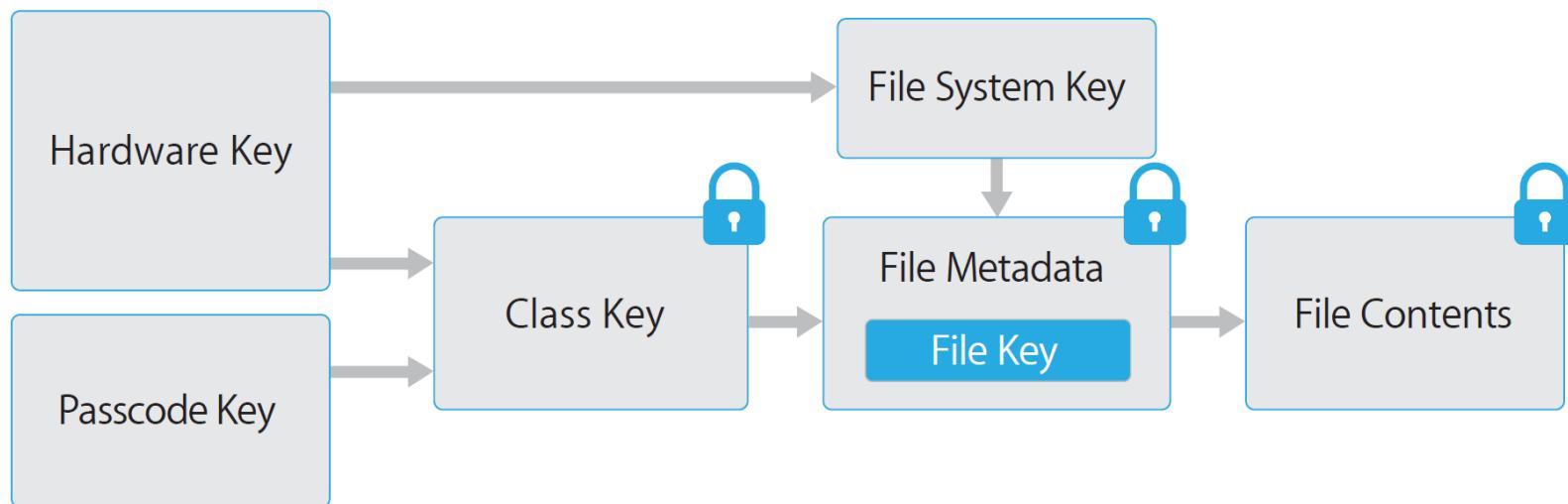
This is pretty cool. How do we do it?

Try to map a key structure to the file system structure.

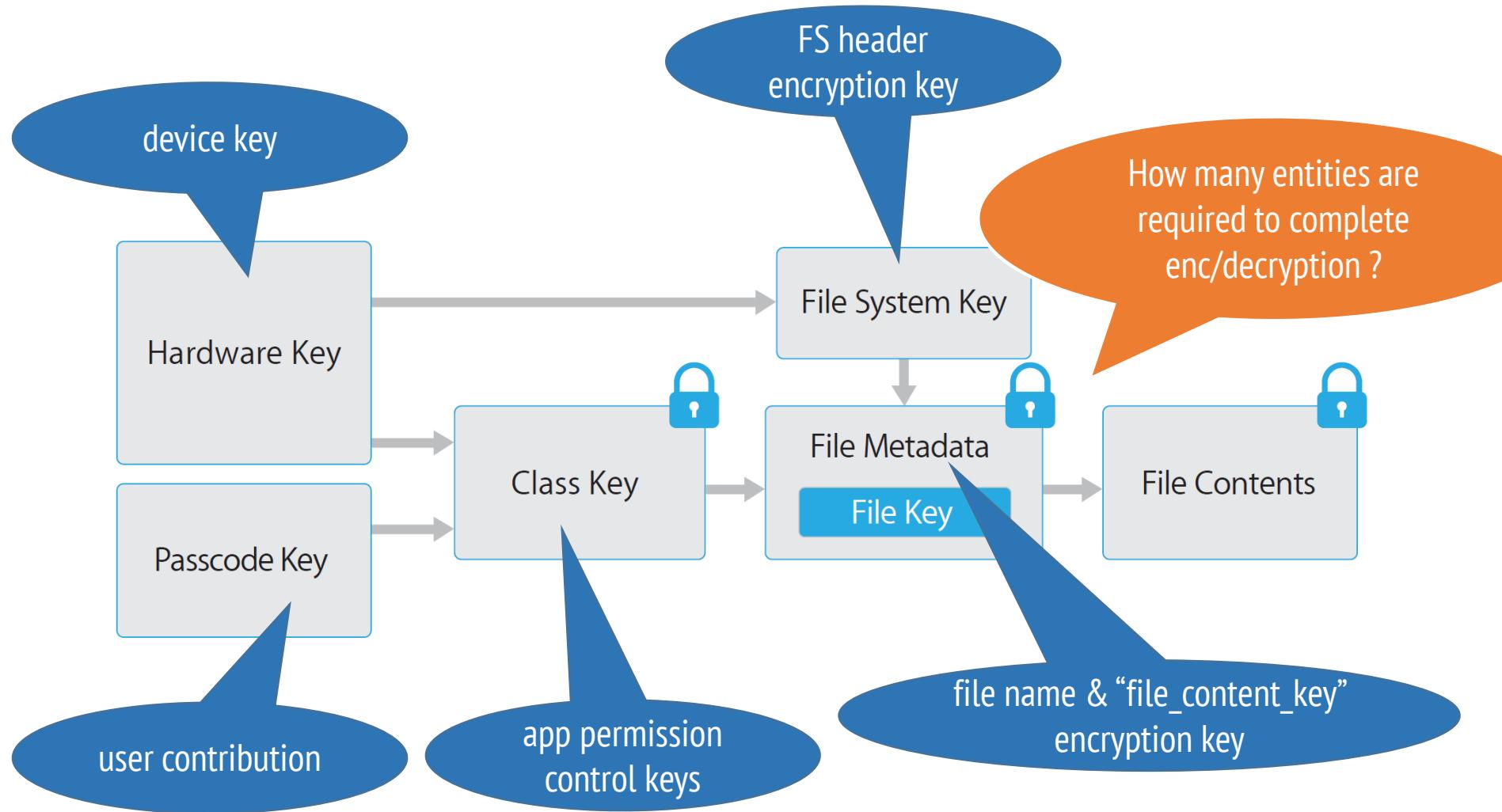


Real implementations

IOS file encrypt



IOS file encrypt

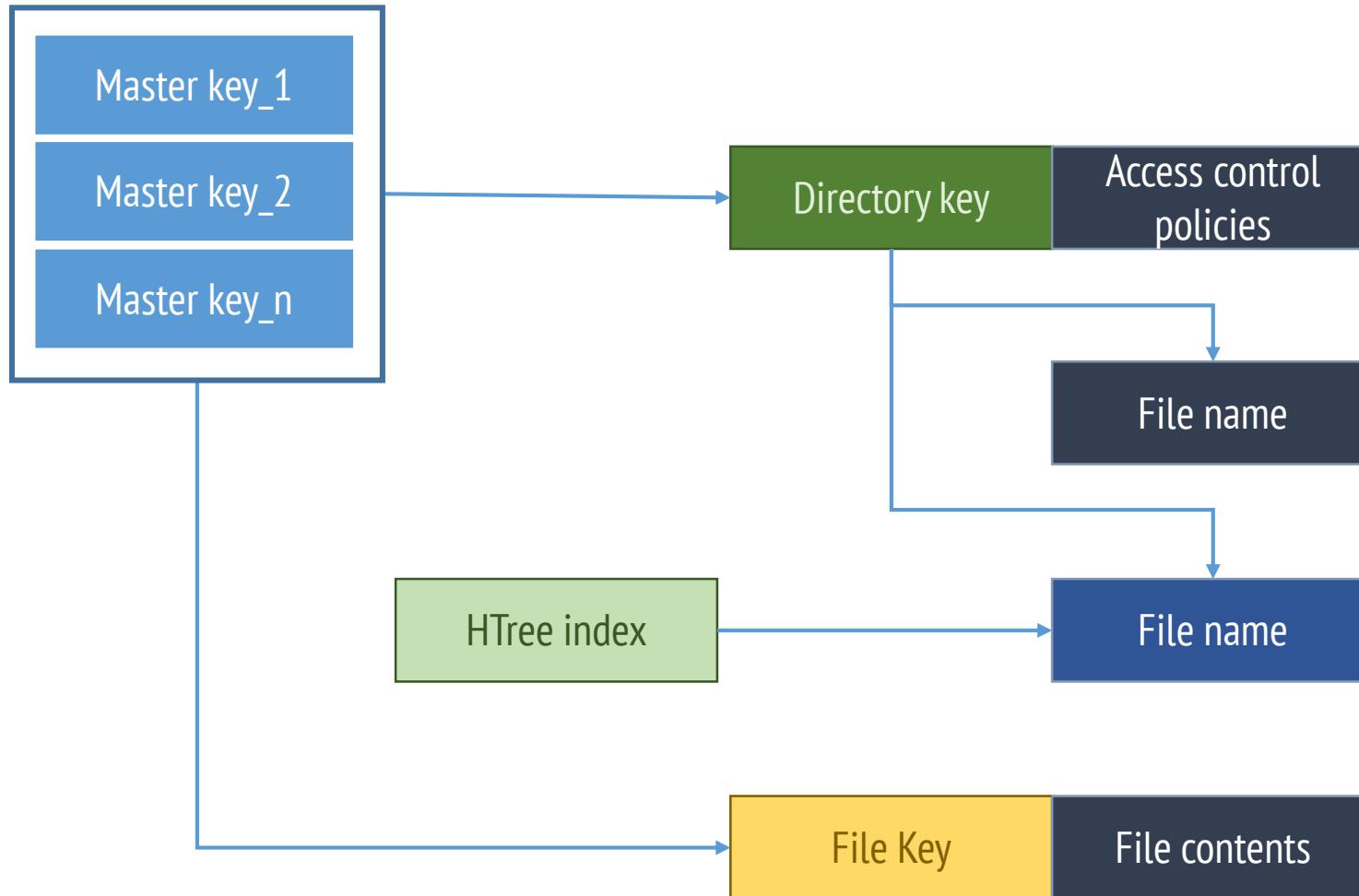


Android 7.0 (with ext4 & dm-crypt)

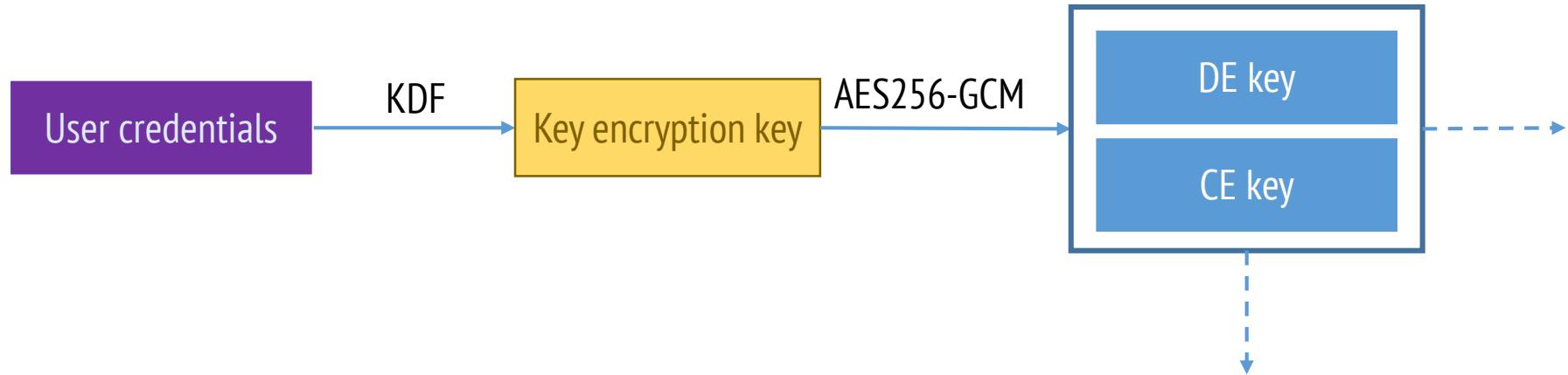
Challenges

- Follow the file structure (What does “directory” mean for ext4)
- What access control to allow if no key? (i.e. What is fail to safe?)
- How to do indexing/search?
- What can we protection can we afford? (i.e. Can we provide authentication?)

Android 7.0 (with ext4 & dm-crypt)



Android 7.0 (with ext4 & dm-crypt)



KEK is held in TEE.

Releasing KEK requires:

1. Stretched Credential: The users' authentication credentials
2. Auth Token: A cryptographically authenticated token generated by gatekeeper.
3. "secdiscardable hash": A hash of a random 16KB file that is stored for each user.

Advantages/disadvantages of file based encryption

File based encryption

- Complex design: generally many keys are used
- Does not protect metadata as well as full disk encryption
- If a key is compromised attacker gets limited access.
- More flexible

Conclusions

- Encryption provides confidentiality to data
- Full disk encryption has a simpler structure
- Full disk encryption hides metadata
- Full disk encryption usually uses one key per disk
- File based encryption has a complex structure
- File based encryption uses many keys thus is more resilient to key compromise
- File based encryption does not hide metadata as well as FDE

Cloud Data Security

Mihai Ordean
Designing Secure Systems
University of Birmingham

Overview

- Device security
 - Is code on the device vulnerable to exploits ? (e.g. buffer overflows)
 - Is the code authenticated ? (i.e. has not been tampered with)
- **Data security (in the cloud)**
 - Is the stored data is accessible to everyone? (e.g. encrypted)
 - Is the stored data authenticated?
- Metadata security
 - What does metadata reveal about data?
 - Can we tamper the metadata?
- Protocol security
 - Is data in transit visible?
 - Can data in transit be tampered with?

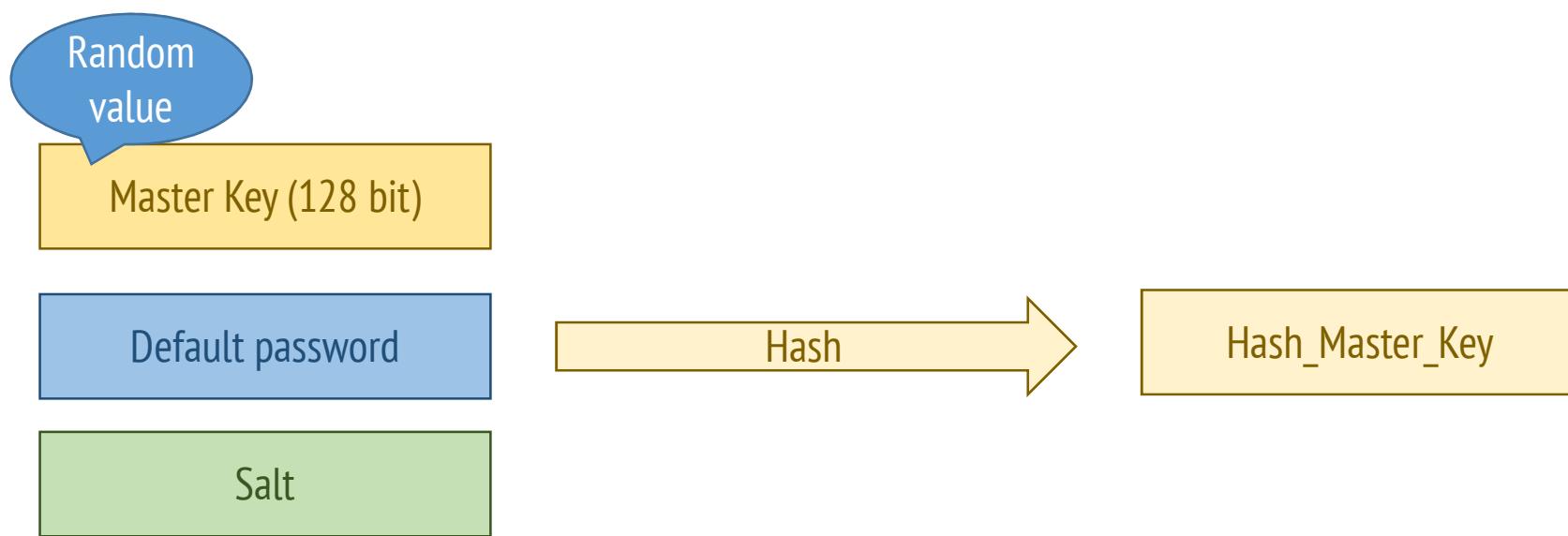
Individual research

What are the security implications of using a public, hardcoded and known value for the “default password” in Android 5.0?

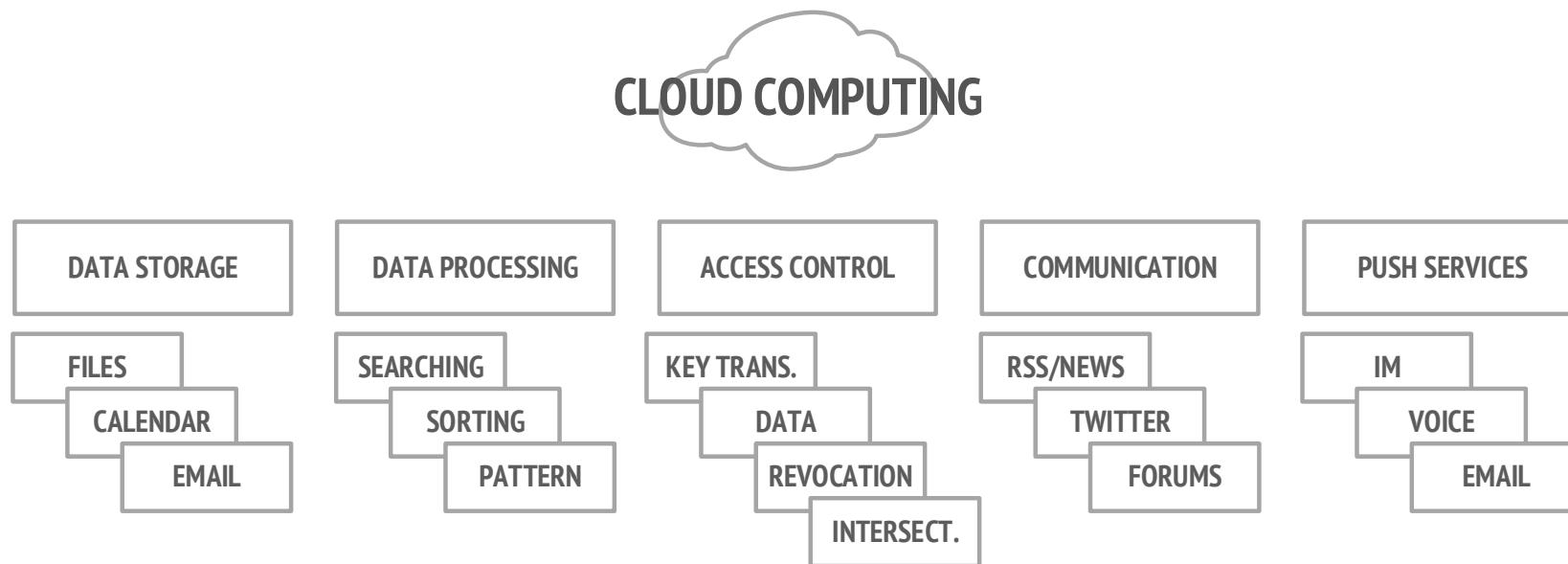
How does it compare to not using encryption at all?

Explain your answers by referring to security aspects (confidentiality, authentication...), and difficulty of use.

Android 5.0 (with Linux kernel & dm-crypt)



What is cloud computing?



Types of cloud

- Public cloud: anyone can share the cloud resource with you
- Private cloud: the cloud is owned/maintained by your own organization
- Community cloud : a group gets together to jointly own/manage a cloud infrastructure
- Hybrid cloud: some mix of public/private e.g., private cloud does data processing, public cloud does bulk storage

Cloud computation requirements



Motivation for using cloud computing

- Simplicity
- Cost
- Security
- Resilience
- Flexibility
- Pace of innovation

Security issues (generic)

- Prevent
 - Deter
 - Detect
 - Respond
 - Recover
 - ...
-
- Confidentiality, integrity, availability
 - Authorisation, management
 - Physical security, personnel security (e.g. employee vetting)
 - Privacy management

Cloud Security Alliance

- The Cloud Security Alliance (CSA) is a not-for-profit organization with a mission to “promote the use of best practices for providing security assurance within Cloud Computing, and to provide education on the uses of Cloud Computing to help secure all other forms of computing”.
- The CSA has over 80,000 individual members worldwide. CSA gained significant reputability in 2011 when the White House selected the CSA Summit as the venue for announcing the federal government’s cloud computing strategy.

Top 12 cloud security issues (CSA, 2016)

1. Data breaches
2. Insufficient identity, credential and access management
3. Insecure interfaces and APIs
4. System vulnerabilities
5. Account hijacking
6. Malicious insiders
7. Advanced Persistent Threats
8. Data loss
9. Insufficient due diligence
10. Abuse and nefarious use of cloud services
11. Denial of service
12. Shared technology issues

Data breaches

- **2017:** A breach exposed the social security numbers and other data. **Credit check firm Equifax says 143m Americans' social security numbers exposed in hack.**
- In mid-2015, **BitDefender**, an antivirus firm, had an **undisclosed number of customer usernames and passwords stolen** due to a security vulnerability in its public cloud application hosted on AWS. The hacker responsible demanded a ransom of \$15,000.
- The 2015 Anthem breach of more than **80 million customer records began with stolen credentials** on the corporate network. A third-party cloud service was used to transfer the huge data store from the company's network to the public cloud where it could be downloaded by the hackers.
- British telecom provider **TalkTalk reported multiple security incidents in 2014 and 2015**, which resulted in the theft of four million customers' personal information.

A data breach is an incident in which sensitive, protected or confidential information is released, viewed, stolen or used by an individual who is not authorized to do so.

Weak identity, credential and access management

- **Attackers Scrape GitHub for Cloud Service Credentials, Hijack Account to Mine Virtual Currency** "Cloud service provider credentials included in a GitHub project were discovered and misused within 36 hours of the project going live."
- **Praetorian Launches Cloud-based Password Cracking Service** "Praetorian, an Austin, Texas-based provider of information security solutions, has launched a new cloud-based platform that leverages the computing power of Amazon AWS in order to crack password hashes in a simple fashion."
- Data breaches and enabling of attacks can occur because of a lack of scalable identity access management systems, failure to use multi-factor authentication, weak password use, and a lack of ongoing automated rotation of cryptographic keys, passwords and certificates.
- Credentials and cryptographic keys must not be embedded in source code or distributed in public facing repositories such as GitHub, because there is a significant chance of discovery and misuse. Keys need to be appropriately secured and a well-secured public key infrastructure (PKI) is needed to ensure key-management activities are carried out.

Insecure APIs

- **The IRS Breach and the Importance of Adaptive API Security.** In mid-2015, the US Internal Revenue Service (IRS) exposed over 300,000 records via a vulnerable API ("Get Transcript").
- **Breach background.** Attackers first gathered several pieces of taxpayers' personal information from outside sources. They then used those information to clear a multi-step authentication process that protected the IRS' "Get Transcript" application.

Cloud computing providers expose a set of software user interfaces (UIs) or application programming interfaces (APIs) that customers use to manage and interact with cloud services. Provisioning, management, orchestration and monitoring are all performed with these interfaces. The security and availability of general cloud services is dependent on the security of these basic APIs.

System & application vulnerabilities

- **Magnified Losses, Amplified Need for Cyber-Attack Preparedness** "Heartbleed and Shellshock proved that even open source applications, which were believed more secure than their commercial counterparts ... , were vulnerable to threats. They particularly affected systems running Linux, which is concerning given that 67.7% of websites use UNIX, on which the former (Linux) is based."
- **Verizon 2015 Data Breach Investigations Report** "The Shellshock bug in Bash was 2014's second tumultuous OSS vulnerability event, quickly eclipsing Heartbleed due to many more successful attacks."
- **2014 Cyberthreat Defense Report** "75% of attacks use publicly known vulnerabilities in commercial software that could be prevented by regular patching."

System vulnerabilities are exploitable bugs in programs that attackers can use to infiltrate a computer system for the purpose of stealing data, taking control of the system or disrupting service operations. Vulnerabilities within the components of the operating system kernel, system libraries and application tools put the security of all services and data at significant risk.

System & application vulnerabilities

- **Shellshock** is a family of security bugs in the Unix Bash shell, the first of which was disclosed in September 2014. Many Internet-facing services, such as some web server deployments, use Bash to process requests, allowing an attacker to cause vulnerable versions of Bash to execute arbitrary commands. This can allow an attacker to gain unauthorized access to a computer system.
- **Heartbleed** is a security bug in the OpenSSL cryptography library, which is a widely used implementation of the Transport Layer Security (TLS) protocol. It was introduced into the software in 2012 and publicly disclosed in April 2014. Heartbleed results from improper input validation (due to a missing bounds check) in the implementation of the TLS heartbeat extension. The vulnerability allows a situation in which more data can be read than should be allowed; in particular, secret keys may be read by an attacker.

Account hijacking

- In April 2010, **Amazon experienced a cross-site scripting (XSS) bug** that allowed attackers to hijack credentials from the site. In 2009, numerous Amazon systems were hijacked to run Zeus botnet nodes.
- In June 2014, **Code Spaces' Amazon AWS account was compromised** when it failed to protect the administrative console with multifactor authentication. All the company's assets were destroyed, putting it out of business.

Account or service hijacking is not new. Attack methods such as phishing, fraud and exploitation of software vulnerabilities still achieve results. Credentials and passwords are often reused, which amplifies the impact of such attacks. Cloud solutions add a new threat to the landscape. If an attacker gains access to your credentials, they can eavesdrop on your activities and transactions, manipulate data, return falsified information and redirect your clients to illegitimate sites. Your account or service instances may become a new base for attackers. From here, they may leverage the power of your reputation to launch subsequent attacks.

Malicious insiders

- **Insider Threats to Cloud Computing**
"Overall, the 'inside job' is responsible for most cloud computing security woes. Enterprises have to become proactive in finding solutions to their security threats to protect their sensitive information."
- **Cloud's Privileged Identity Gap Intensifies Insider Threats**
"Organizations need to rein in shared accounts and do a better job tracking user activity across cloud architectures."
- **David Barksdale accessed users' accounts, violating the privacy of at least four minors** during his employment as *Systems Engineer* at Google. He boasted about it with friends. He was fired in July 2010 after his actions were reported to the company.

"A malicious insider threat to an organization is a current or former employee, contractor, or other business partner who has or had authorized access to an organization's network, system, or data and intentionally exceeded or misused that access in a manner that negatively affected the confidentiality, integrity, or availability of the organization's information or information systems."

Advanced Persistent Threats (APTs)

- **Carbanak: How Would You Have Stopped a \$1 Billion APT Attack?** "... Carbanak, an APT attack against financial institutions around the world, may well be considered the largest cyberheist to date. ... Unlike the usual cybercriminal method of stealing consumer credentials or compromising individual online banking sessions with malware, the brazen Carbanak gang targeted banks' internal systems and operations, resulting in a multichannel robbery that averaged \$8 million per bank."
- **Current Trends in the APT World** "The alleged Chinese Cyber-Espionage with its APTs caused the theft of "hundreds of terabytes of data from at least 141 organizations across a diverse set of industries beginning as early as 2006."

Advanced Persistent Threats (APTs) infiltrate systems to establish a long-lived foothold in the target platform, usually with political or commercial motives. APTs pursue their goals stealthily over extended periods of time. Once in place, APTs can move laterally through data center networks and blend in with normal network traffic to achieve their objectives.

Data loss

- In April 2011, Amazon EC2 suffered a crash that led to significant data loss for many customers.
- In June 2014, **Code Spaces, an online hosting and code publishing provider**, was hacked, leading to the compromise and complete destruction of most customer data. The company was ultimately unable to recover from this attack and went out of business.

Data stored in the cloud can be lost for reasons other than malicious attacks. An accidental deletion by the cloud service provider, or worse, a physical catastrophe such as a fire or earthquake, can lead to the permanent loss of customer data unless the provider or cloud consumer takes adequate measures to back up data, following best practices in business continuity and disaster recovery as well as daily data backup and possibly off-site storage.

Furthermore, the burden of avoiding data loss does not fall solely on the provider's shoulders. If a customer encrypts his or her data before uploading it to the cloud but loses the encryption key, the data will be lost as well.

Insufficient due diligence

- **Contract and Financial Viability** In 2013, Nirvanix, a cloud storage specialist that hosted data for IBM, Dell and its own customers, filed for Chapter 11 bankruptcy and shuttered its operations. Customers were given less than two weeks to move their data to another service. This caused huge problems for customers. For example:
 - Film and TV production studio Relativity Media was using Nirvanix's cloud as a hub through which employees in its global locations could collaborate and share massive digital files to accelerate production.
- **Non-Compliance:** Healthcare and financial services must retain data to meet government compliance regulations. If the data is lost, these services become non-compliant.

Companies should develop a good roadmap and checklist for due diligence when evaluating technologies and cloud service providers (CSPs). An organization that rushes to adopt cloud technologies and choose CSPs without performing due diligence exposes itself to a myriad of commercial, financial, technical, legal and compliance risks that jeopardize its success.

Abuse and nefarious use of cloud services

- The DDoS That Almost Broke the Internet

"The attackers were able to generate more than 300 Gbps of traffic. Their own network only had access to 1/100th of that amount of traffic."

- Poorly secured cloud service deployments, free cloud service trials and fraudulent account sign-ups via payment instrument fraud expose cloud computing models IaaS, PaaS, and SaaS to malicious attacks.
- Malicious actors may leverage cloud computing resources to target users, organizations or other cloud providers. Examples of misuse of cloud service-based resources include launching DDoS attacks, email spam and phishing campaigns; "mining" for digital currency; large-scale automated click fraud; brute-force compute attacks of stolen credential databases; and hosting of malicious or pirated content.

Denial of service

- **As cloud use grows, so will rate of DDoS attacks** "Cloud providers face increasing number of DDoS attacks, [similar to those that] private data centres already deal with today"
- **Feedly Knocked Offline by DDoS Attack Following Evernote and Deezer Attacks** "In what looks like a series of co-ordinated cyber-attacks by a criminal gang, three major cloud-based services have all been knocked offline in recent days. News aggregator Feedly, note-taking app Evernote and music streaming service Deezer have all come under attack from criminals in the last few days leading to all three suffering service outages."
- Denial-of-service (DoS) attacks are attacks meant to prevent users of a service from being able to access their data or their applications. By forcing the targeted cloud service to consume inordinate amounts of finite system resources such as processor power, memory, disk space or network bandwidth, the attackers cause an intolerable system slowdown and leaves all legitimate service users unable to access the service.

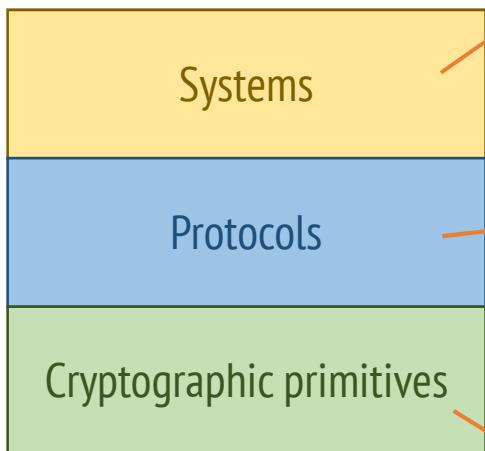
Shared technology issues

- **Cross-VM Side Channels and Their Use to Extract Private Keys** "...construction of an access-driven side-channel attack by which a malicious virtual machine (VM) extracts fine-grained information from a victim VM running on the same physical computer."
- **Understanding the VENOM Vulnerability**
"The unchecked buffer vulnerability (CVE-2015-3456) occurs in the code for QEMU's virtual floppy disk controller. A successful buffer overflow attack exploiting this vulnerability can enable an attacker to execute his or her code in the hypervisor's security context and escape from the guest operating system to gain control over the entire host."

Cloud service providers deliver their services scalably by sharing infrastructure, platforms or applications. Underlying components (e.g., CPU caches, GPUs, etc.) that comprise the infrastructure supporting cloud services deployment may not have been designed to offer strong isolation properties for a multitenant architecture (IaaS), re-deployable platforms (PaaS) or multicustomer applications (SaaS). This can lead to shared technology vulnerabilities that can potentially be exploited in all delivery models. A defense-in-depth strategy is recommended and should include compute, storage, network, application and user security enforcement and monitoring, whether the service model is IaaS, PaaS, or SaaS. A single vulnerability or misconfiguration can lead to a compromise across an entire provider's cloud.

Confidentiality from the cloud provider

The “security stack”



Goals: send encrypted messages (Signal), access network from remote location securely (IP-SEC)
Systems examples:
Secure messaging, secure file storage, Secure DB,...

Goals: Authenticate parties, generate session keys, establish end-to-end enc. channels...
Cryptographic protocol examples:
TLS, SSH, ...

Primitive examples:
Symmetric crypto: AES, 3-DES, ...
Public crypto: RSA, DH, ...
Hash functions: SHA(1-3), RC4, ...
Digital signatures
Random number generators
Private information retrieval
....

Attacker models

- An **attacker model** is a description of capabilities specifying the kind of access an attacker has to a system when attempting to "break" it.

Attacker models



Attacker models



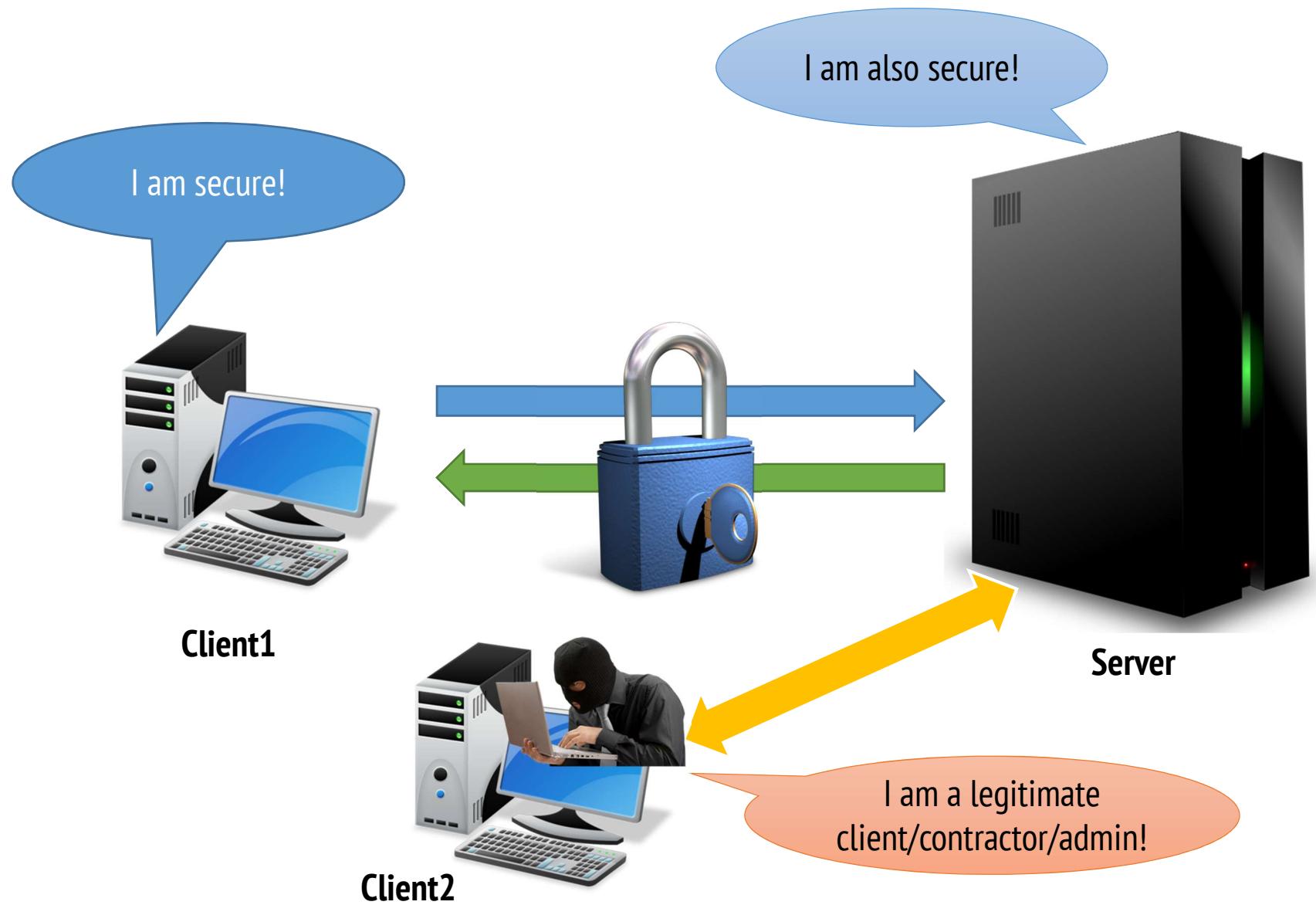
Attacker models



Attacker models



Attacker models



Confidentiality from the cloud provider Holy grail?

- Currently, the price of using the cloud is that you have to divulge your data to the cloud provider.
- The business model of many cloud providers, such as Google, is to exploit that data.
- Many designs for new architectures focus on the idea that it might be possible to get the benefits of cloud computing without paying that price.

Is confidentiality from the cloud provider enough?

- CIA...
- Do we need integrity and availability as well?
- A cloud provider can stealthily breach your data confidentiality.
- It cannot stealthily breach your data integrity or data availability.

Is confidentiality from the cloud provider enough?

The malicious-but-cautious attacker model is appropriate for the cloud service provider:

- Malicious, because the CSP will cheat if it can
- Cautious, because it wants to keep your custom.

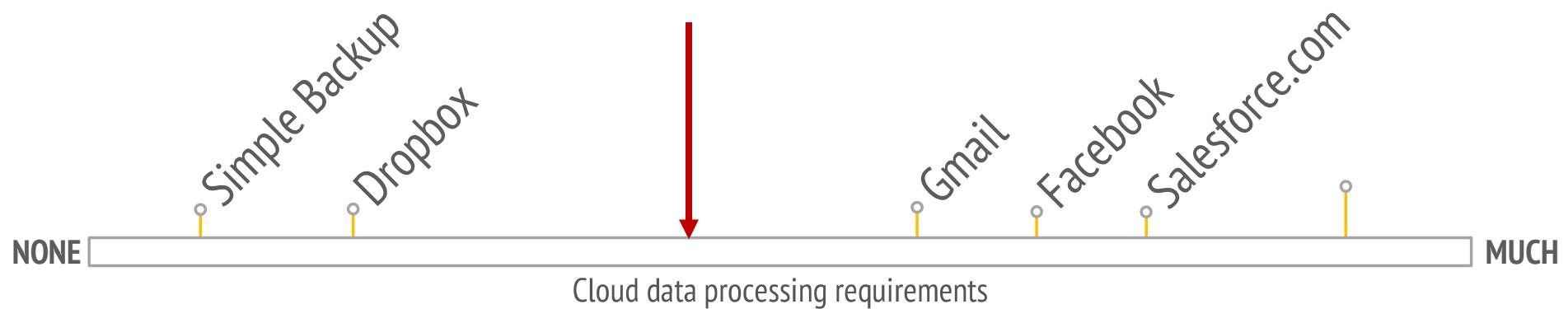
What issues are addressed by “confidentiality-from-the-cloud-provider”?

1. Data breaches
2. Insufficient identity, credential and access management
3. Insecure interfaces and APIs
4. System vulnerabilities
5. Account hijacking
6. Malicious insiders
7. Advanced Persistent Threats
8. Data loss
9. Insufficient due diligence
10. Denial of service
11. Shared technology issues

Confidentiality from the cloud provider

- Avoiding having to trust cloud provider means:
 - You don't have to trust its employees or subcontractors
 - You don't lose data confidentiality if cloud provider gets hacked

Cloud computation requirements

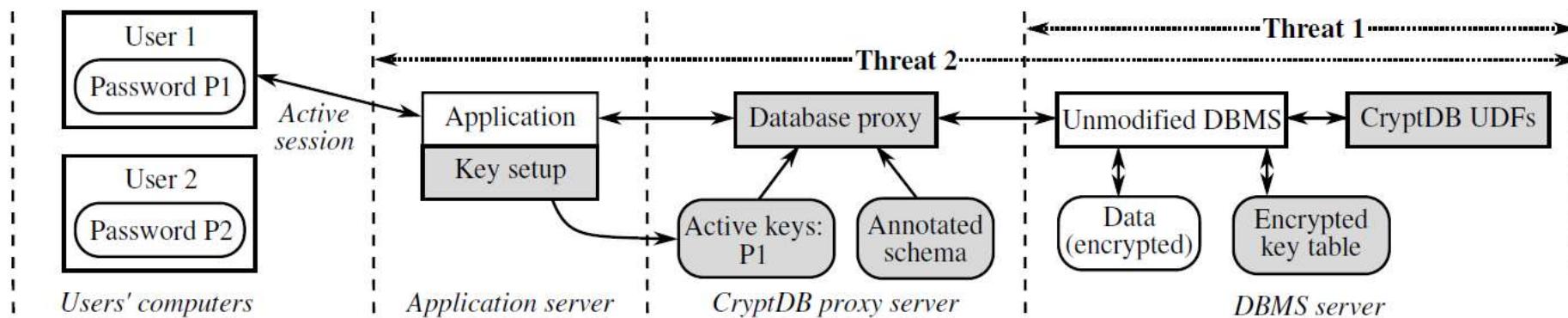


CryptDB

A secure database system

R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, “Cryptdb: protecting confidentiality with encrypted query processing,” in Proceedings of the 23rd ACM Symposium on Operating Systems Principles, 2011.

CryptDB: A secure database system



CryptDB: A secure database system

- CryptDB
 - Is a secure middleware that protects databases deployed on cloud servers
 - Requires a proxy server that is placed in a trusted zone to handle cryptographic key translation
 - Stored data is encrypted in multiple formats to permit database operations
 - Can dynamically adjust data encryption to accommodate for new database operations
 - Runs on unmodified database servers
- Threats
 - Threat 1 – Curious administrator with full access to the DB server
 - Threat 2 – If CryptDB proxy is compromised the adversary cannot obtain data belonging to users that are not logged in

Threat 1: Curious DB administrator

Employees

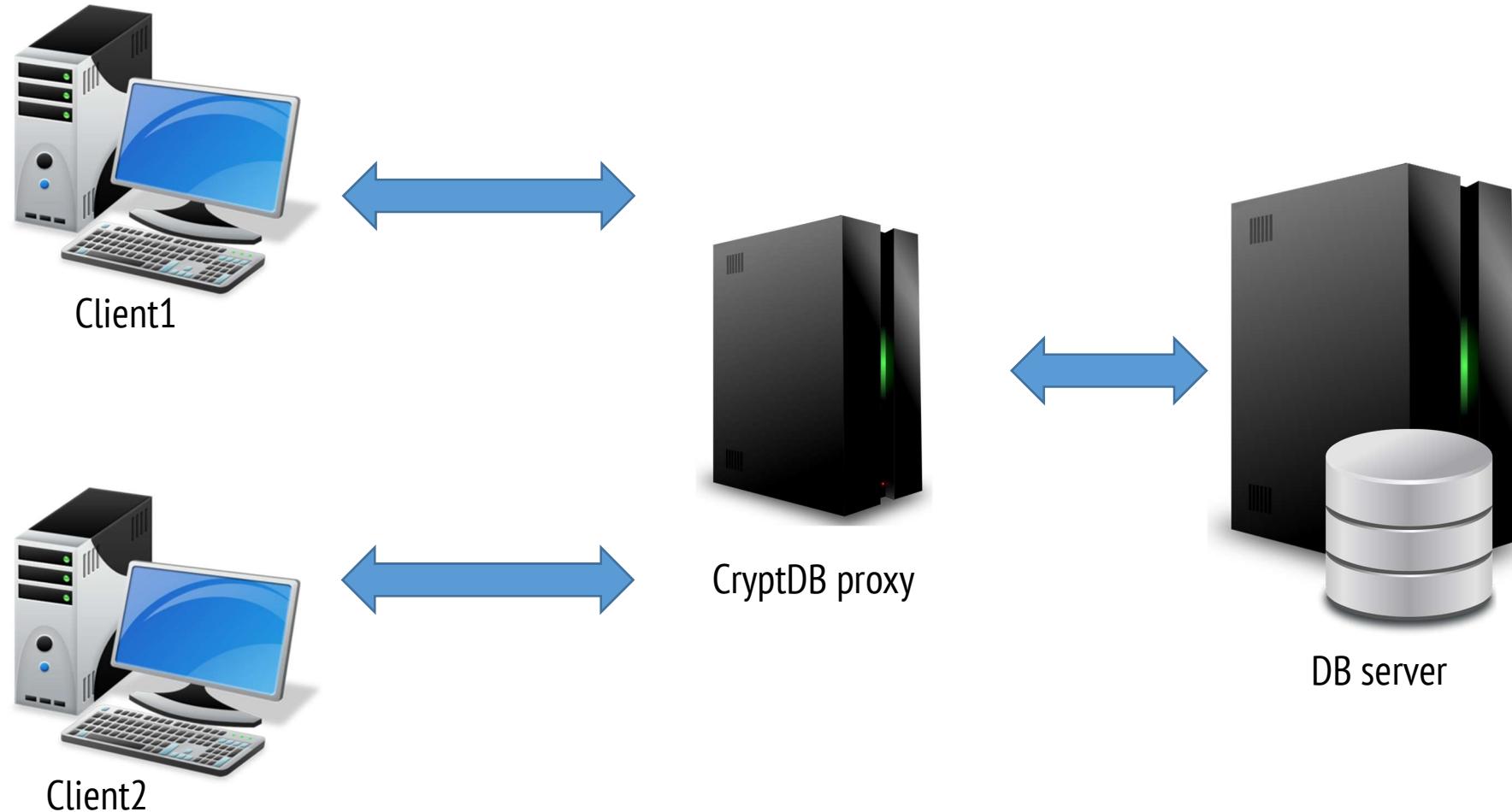
<i>ID</i>	<i>Name</i>
23	Alice

Change
to

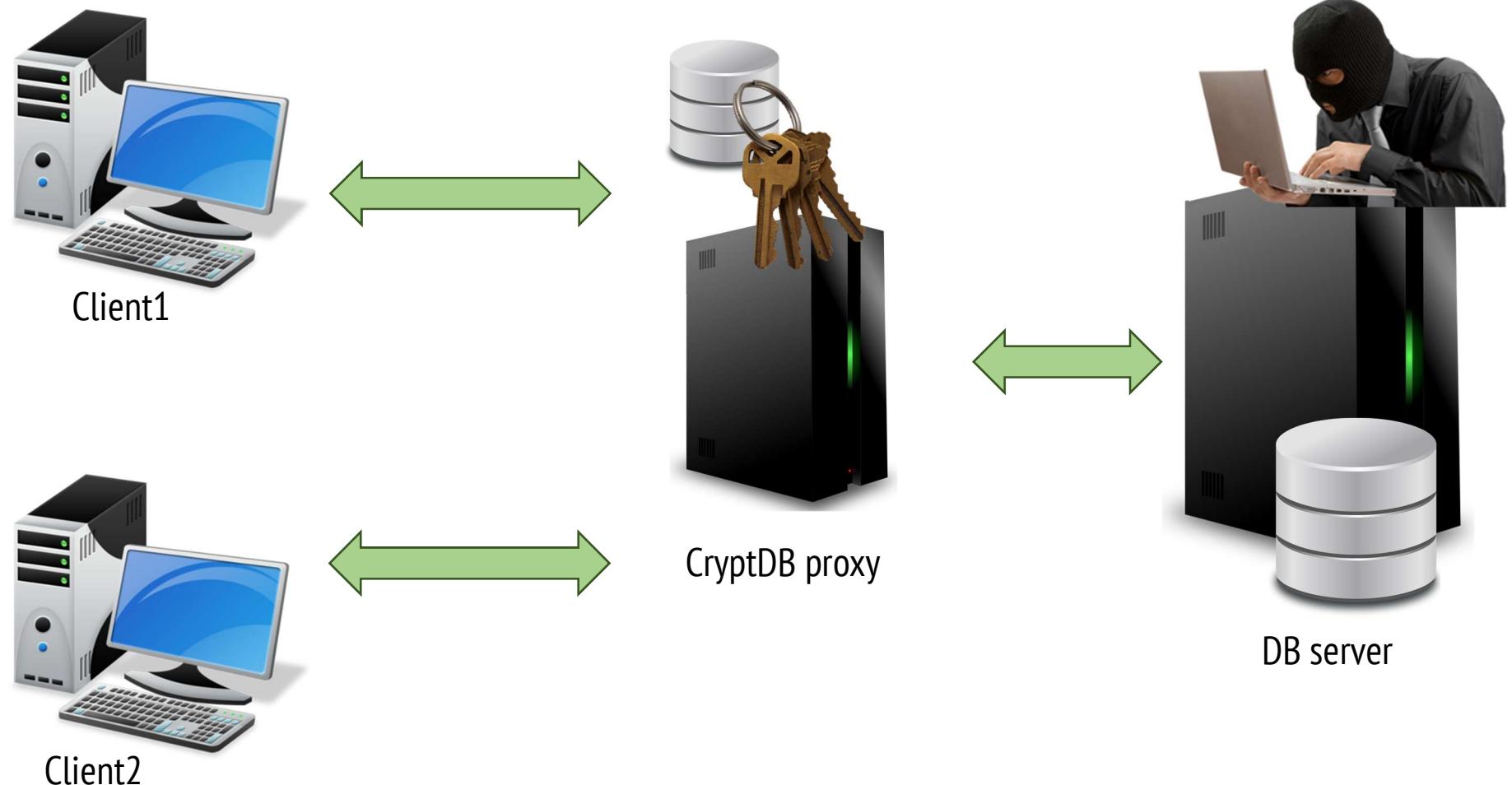
Table1

<i>C1-IV</i>	<i>C1-Eq</i>	<i>C1-Ord</i>	<i>C1-Add</i>	<i>C2-IV</i>	<i>C2-Eq</i>	<i>C2-Ord</i>	<i>C2-Search</i>
x27c3	x2b82	xcb94	xc2e4	x8a13	xd1e3	x7eb1	x29b0

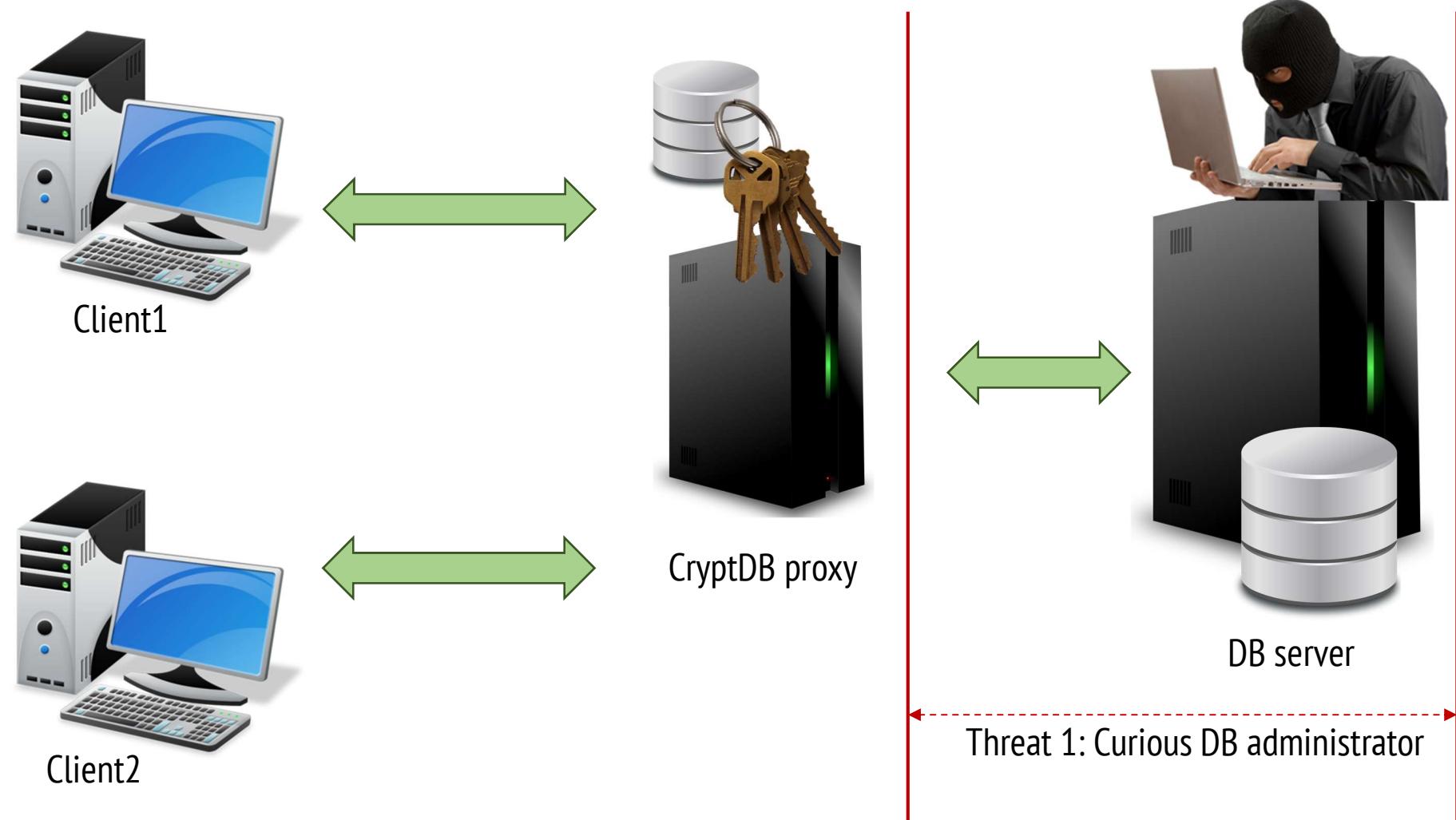
CryptDB



CryptDB



CryptDB



CryptDB: DB functionality

Naïve approach:

- Store keys in the CryptDB proxy.
- Encrypt database with keys.

Usage

- Download DB from server to CryptDB proxy
- Decrypt
- Perform DB operation (e.g. sort)
- Send result to user

CryptDB: DB functionality

Naïve approach:

- Store keys in the CryptDB proxy.
- Encrypt whole database with one key.

Usage

- Download DB from server to CryptDB proxy
- Decrypt
- Perform DB operation (e.g. sort)
- Send result to user

NOT EFFICIENT !

CryptDB: DB functionality

A more efficient approach?

- Store keys in the CryptDB proxy.
- **Encrypt each record of the DB with a key.**

Usage

- Download DB from server to CryptDB proxy
- Decrypt
- Perform DB operation (e.g. sort)
- Send result to user

CryptDB: DB functionality

A more efficient approach?

- Store keys in the CryptDB proxy.
- **Encrypt each record of the DB with a key.**

Usage

- Download DB from server to CryptDB proxy
 - Decrypt
 - Perform DB operation (e.g. sort)
 - Send result to user
-
- **EQUALLY NOT EFFICIENT !**

CryptDB: DB functionality

CryptDB approach:

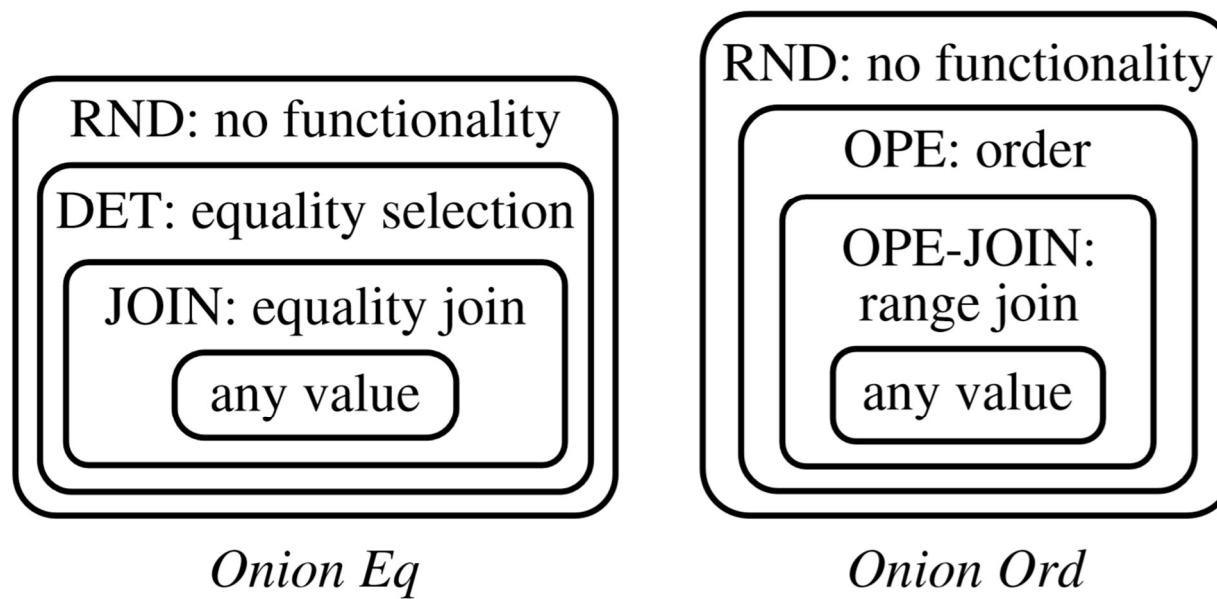
- Store keys in the CryptDB proxy.
- **Encrypt each record of the DB with a key.**
- **Use metadata information to specify how different DB components are to be used.**
- **Use different encryption methods to “augment” the usage.**

Threat 1: Curious DB administrator

Crypto that enables functionality:

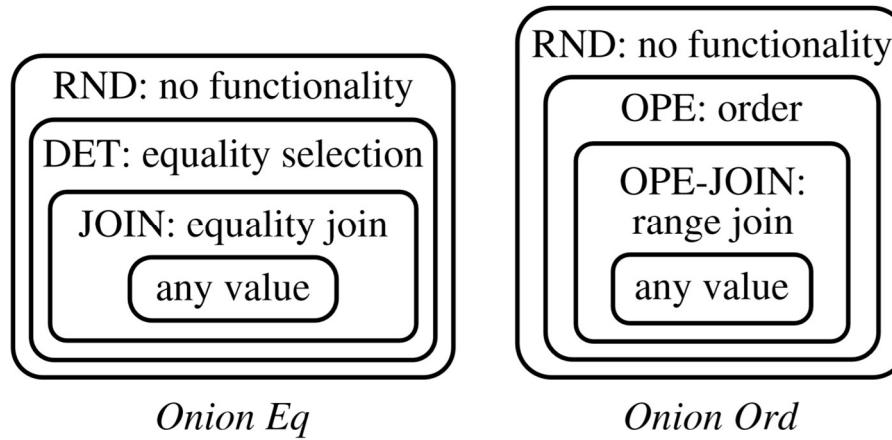
- Hide everything (RND): encrypt using non-deterministic encryption
- Equality comparison (DET): encrypt using deterministic encryption
- Sort functionality (order-preserving encryption OPE): make sure that if $x < y$ then $\text{enc}(x) < \text{enc}(y)$.
- Join functionality (**equality-join** and **range-join**): use same key for multiple DB columns.

CryptDB: DB functionality



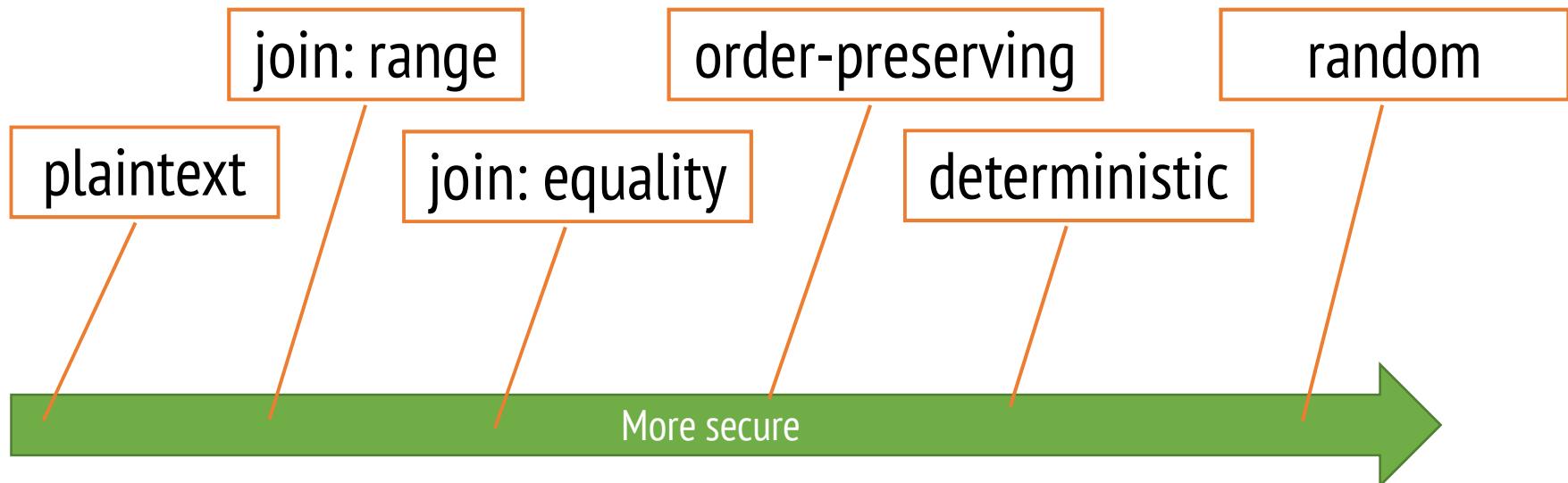
Encrypted DB record in CryptDB

Issues



Onion Eq

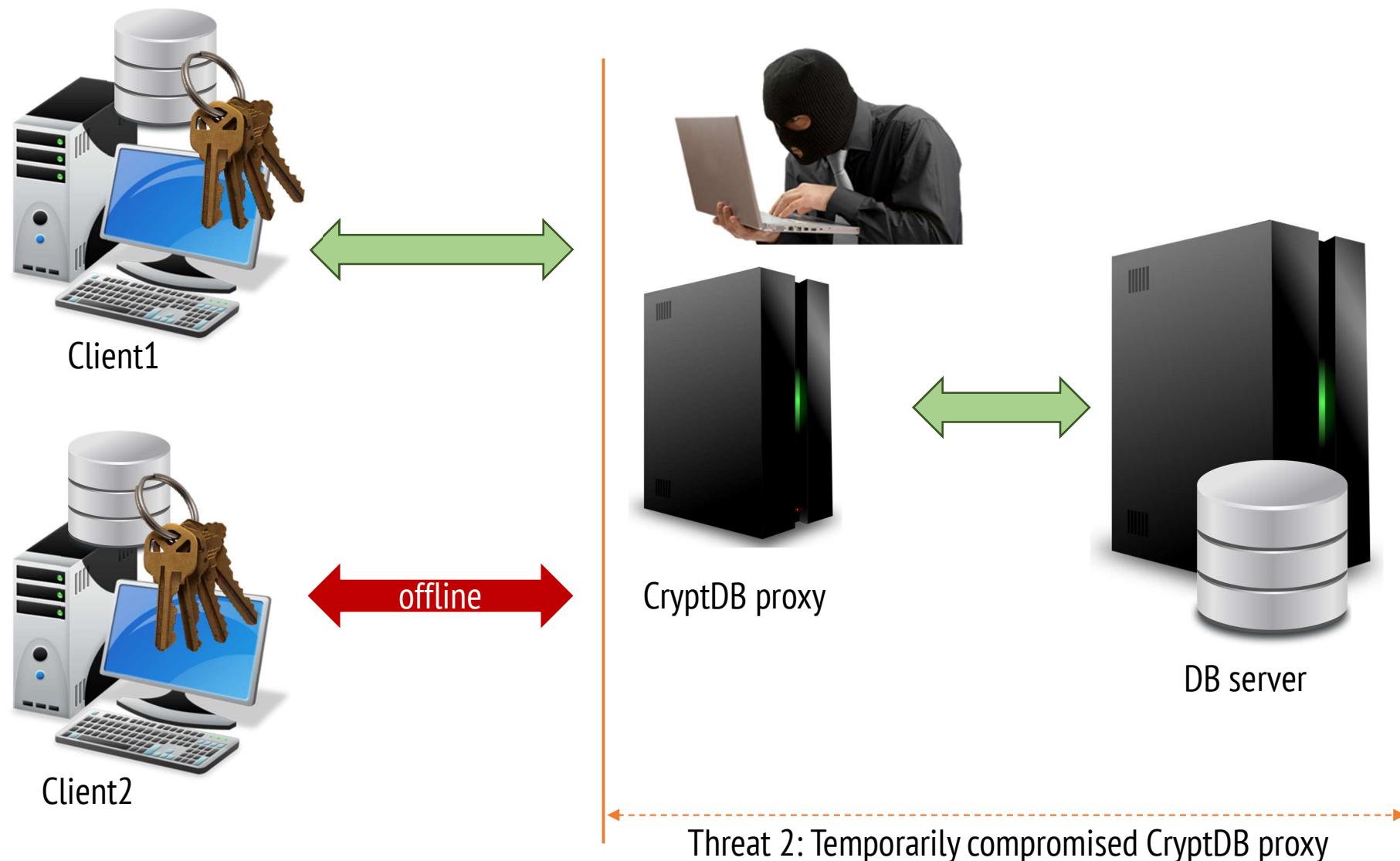
Onion Ord



Issues

- Not all records are equally secure.
- Over time all records will default to using the weakest security allowed.
- **FAIL TO SAFE is not respected.**

Threat 2: Tmp. compromised CryptDB proxy



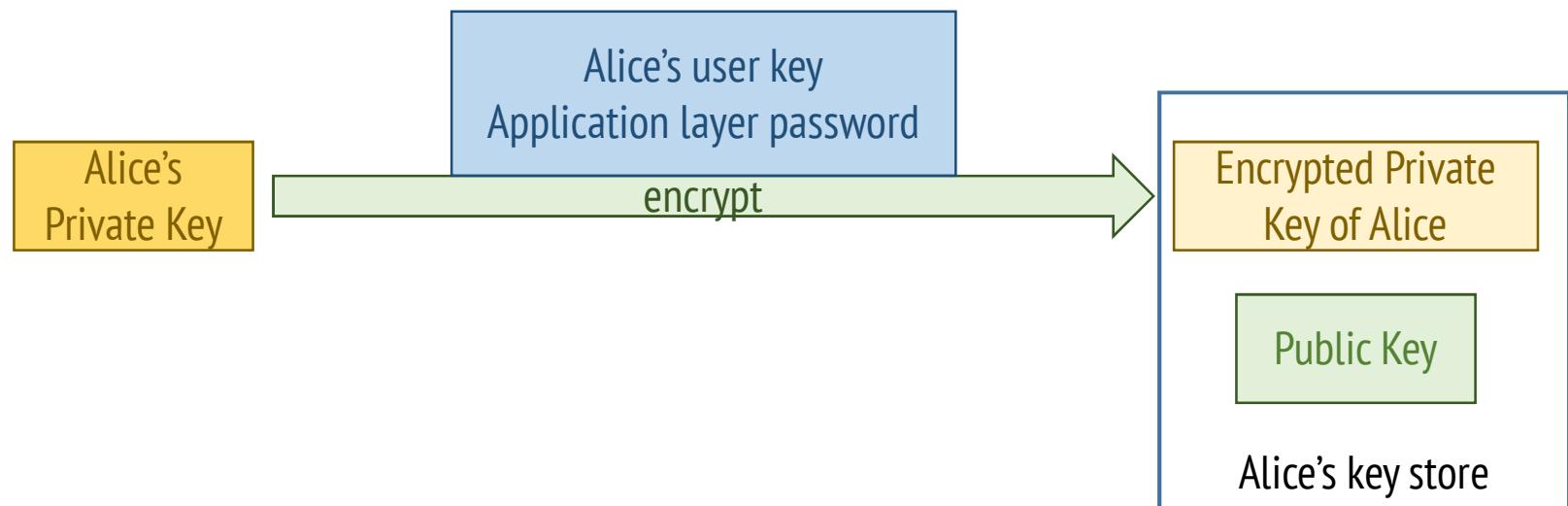
Key Chaining

The system must ensure that:

- CryptDB proxy can perform operations on behalf of the user.
- DB keys are derived from the user's password
- Keys are only derived when logged on to the CryptDB proxy
- CryptDB proxy is trusted to discard key after the operation is successfully completed

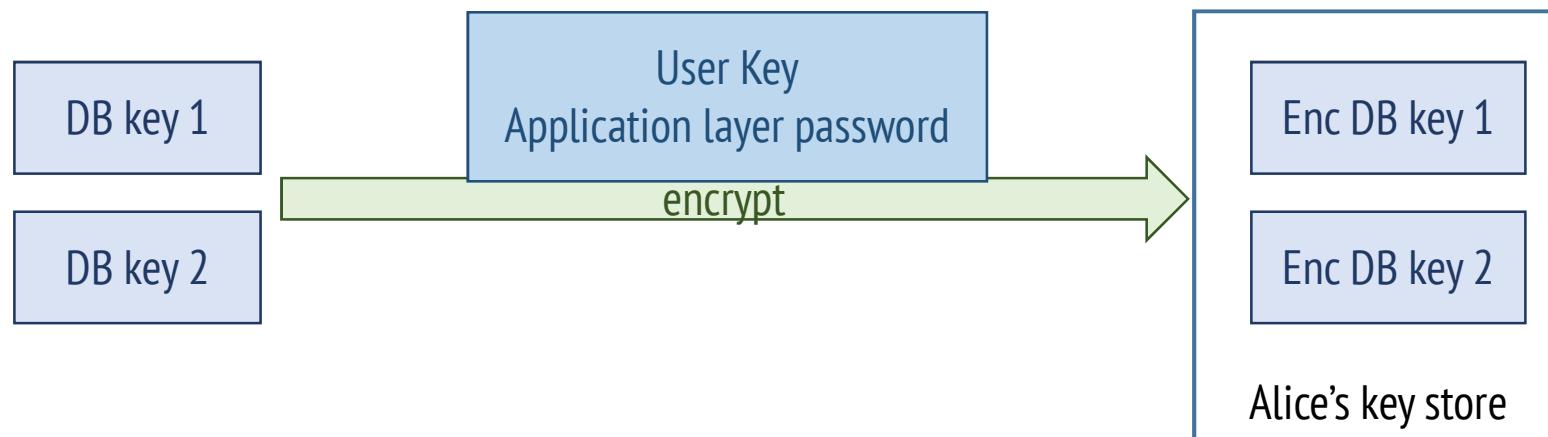
Key Chaining

User keys derivation



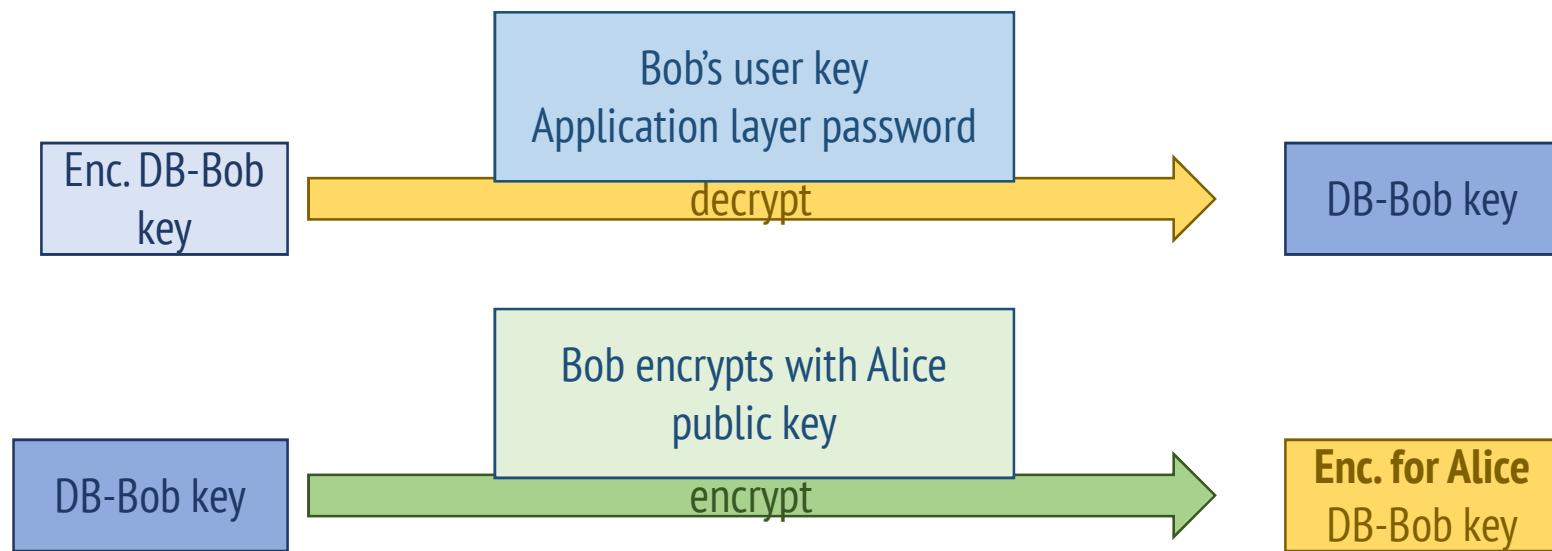
Key Chaining

DB access



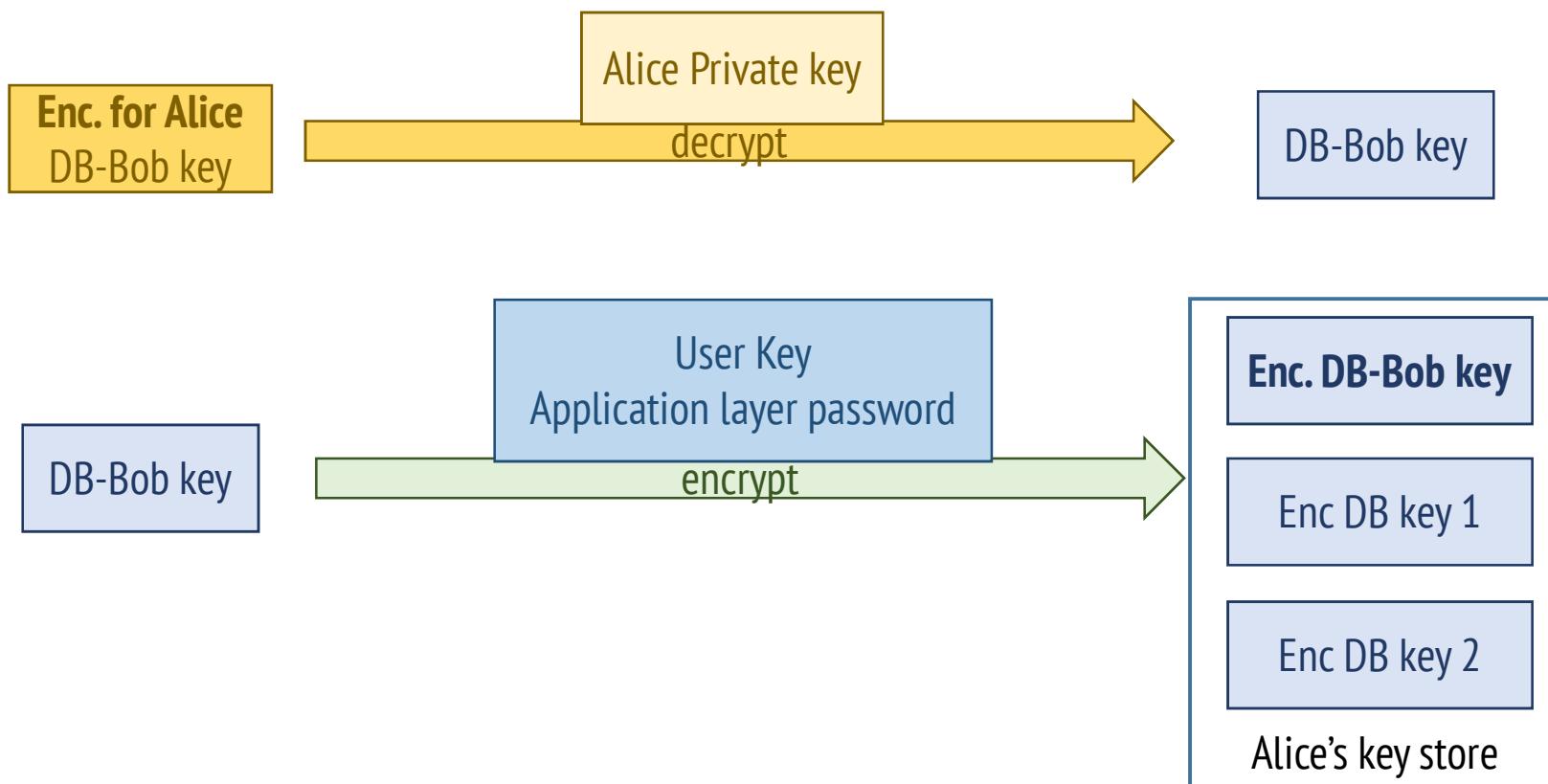
Key Chaining

Bob wants to share DB-Bob with Alice

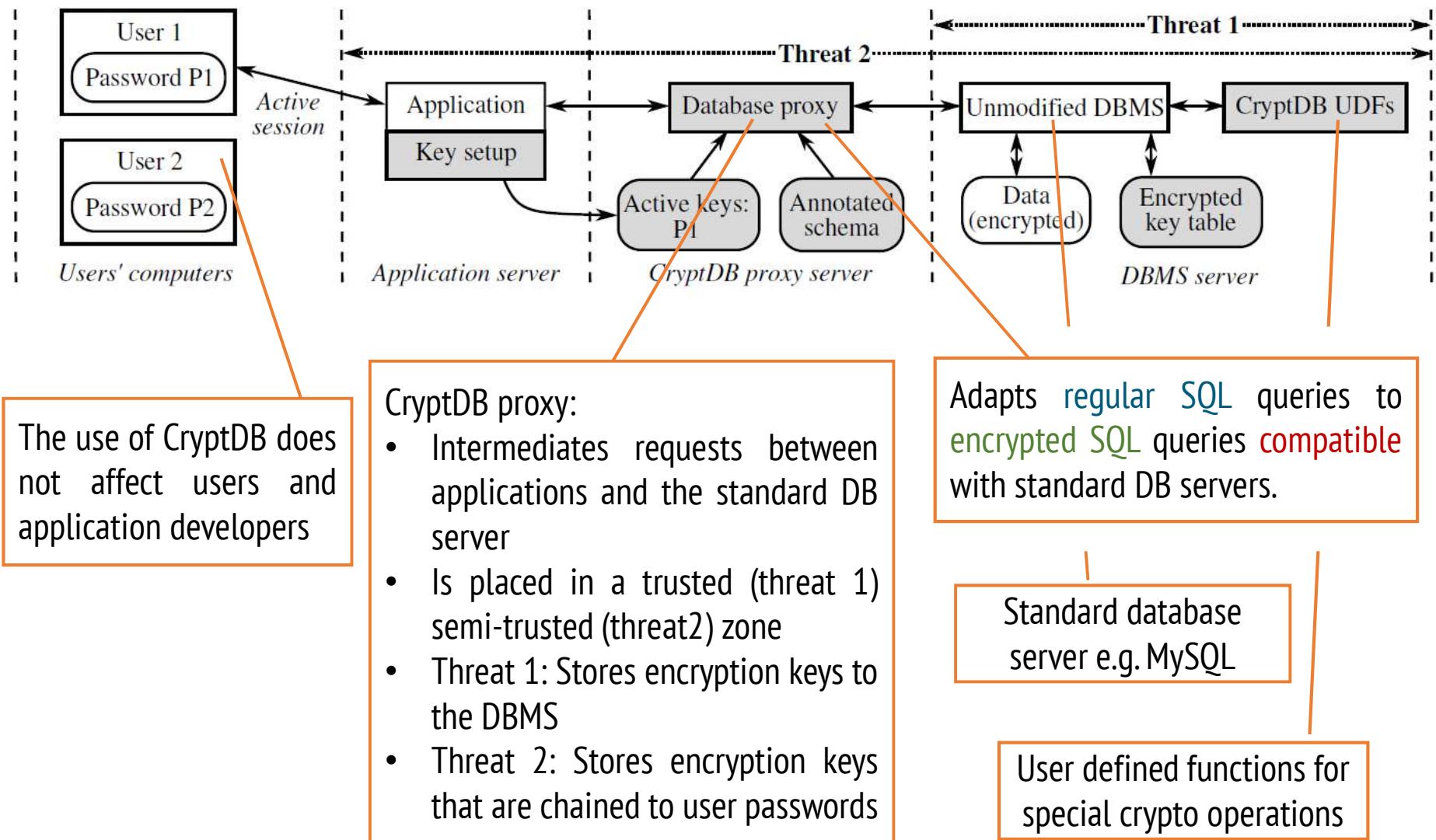


Key Chaining

Alice adds key to her store



CryptDB: A secure database system



ConfiChair

An application management system

Conference system functionality

1. Conference chair sets up her conference
2. Chair “opens” the conference for submissions
3. Authors submit papers to the conference
4. Chair “closes” the conf for submissions
5. Chair assigns (say) 3 PC members (aka reviewers) to each paper, and opens the ‘reviewing phase’
6. PC members login, see their assignments, download the papers, write their reviews
7. Once all the reviewers have reviewed a paper, the discussion phase begins for the paper. Reviewers read each other’s reviews and comment on them, aiming to reach accept/reject consensus.
8. Chair takes the accept/reject decision, usually following the recommendation of the reviewers.

Why do it in the cloud?

Advantages

- Cloud provider takes responsibility for all aspects of organising the service: storage, software, backup, availability, security, version management, ...
- And does it all for free!
- Problem
 - Cloud provider has full unmediated access to all the data.

EasyChair

- A **SaaS** system consisting of authors, reviewers, & chairs



“We believe that since 2006 we have become number one conference management system in the number of conferences, users and submissions. All together EasyChair proudly hosted 49,000 conferences and 1M users.”

The confidentiality problem

- EasyChair managers have direct access to **the submission and reviewing profiles** of 1M users across 49k conferences, including submission rate, acceptance rate, reviewer profile (fair/unfair, thorough/scant, prompt/late).
- EasyChair could [in principle, if it wished] **offer profiling services to appointment panels, awarding bodies, recruitment agencies.**
- The data could become a **target for hackers/crackers.**
- **Similar situation for other SaaS systems.**

ConfiChair: An application management system

- A secure application management system
 - Cryptographically protects applications and evaluations from the cloud provider
 - Uses data re-encryption to enable the administrative role user to provide differentiated access control
 - Uses an encrypted Key Purse to manage keys
 - Uses Stanford JavaScript Crypto Library (SJCL) and HTML5 to provide in-browser encryption
- Security properties
 - Secrecy properties – ConfiChair doesn't know the content of applications, evaluations, rankings, discussions, notifications.
 - Unlinkability property - ConfiChair does not know that a particular evaluator R evaluated an application submitted by a particular author A.

Security properties

Secrecy property:

ConfiChair does not know the content of:

- Papers
- Reviews
- Scores
- Discussions
- Decisions

Unlinkability property:

- ConfiChair does not know that a particular reviewer R reviewed a paper written by a particular author A.

Functionality

- Initialisation
 - Setup the cryptographic keys for a new conference
- Registration
 - Authors register their papers
- Evaluation/Review
 - Reviewers are given access to author's papers and can write reviews.
- Discussion
 - Reviewers are allowed to communicate with one-another for the purposes of agreeing on the features/short comes of the paper
- Notification
 - Authors are notified of their success

Functionality

There are 3 user roles:

- Chair
- Author
- Reviewer

All users have a keypurse which stores their cryptographic material encrypted with a password derived key

Keypurse contents

- Chairs
 - Public/private conference key pair
 - A symmetric conference key
- Authors
 - Unique keys for each paper submitted
- Reviewers
 - Pre-shared symmetric conference keys
- Cloud
 - No un-encrypted cryptographic material

ConfiChair: An application management system

The screenshot shows a web application interface for creating a new conference. On the left, there's a main form titled "Create a new conference". It has fields for "Acronym" (containing "SCC17") and "Title and description" (containing "Super cool conference 2017"). A "Create" button is at the bottom of this form. On the right, there's a sidebar with two sections: "Keys" and "Reviewers' key [K_{Conf}]". The "Keys" section contains a public key: 5621526c250adf9cd4b5ab56049f9d570bb7ca761b9c3e3759b8067ff5a06bbf58e0798025475dbd22d86131a6c1b74b1a87411aea40518b50546fe86a468c1958c4d38402161b6b97f5fd0abfea894f9f28a6e72352ee613a31890831dbf92a5. The "Reviewers' key" section contains a key: 4e417316d6a660324e1a1dab5f93f83fdd7239c08ad2bc088681f7fc2fe37f70.

CONFICHAIR

[about](#) | [test@test.com](#) | [role: chair \(change\)](#) | [users](#) | [logout](#)

Create a new conference

Acronym
SCC17

Title and description
Super cool conference 2017

Keys

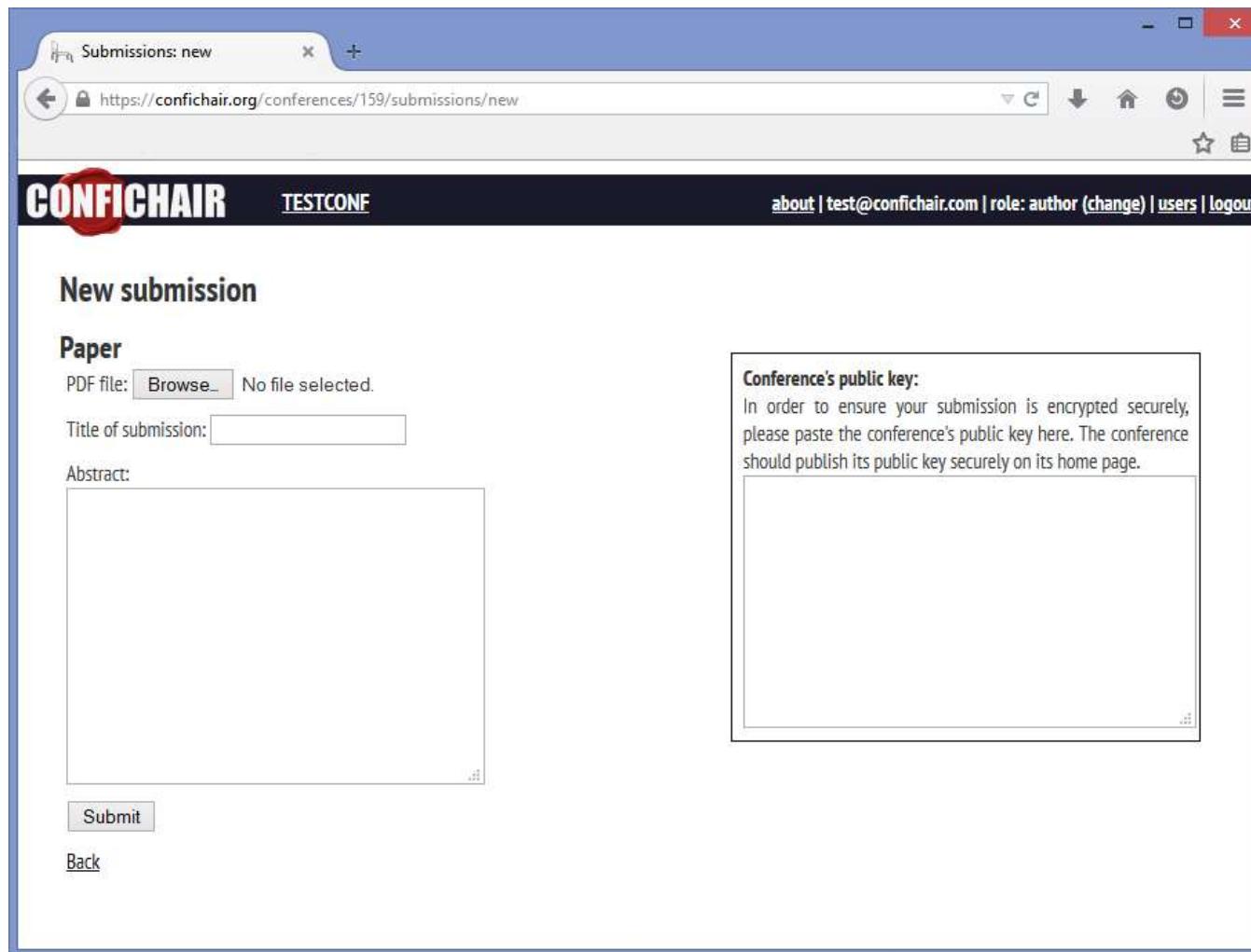
Public key [pub(Conf)]
You should publish this key on your conference's web site, so that authors can confirm that it is indeed your key

```
5621526c250adf9cd4b5ab56049f9d570bb7ca761b9c3e3759b8067ff5a06bbf58e0798025475dbd22d86131a6c1b74b1a87411aea40518b50546fe86a468c1958c4d38402161b6b97f5fd0abfea894f9f28a6e72352ee613a31890831dbf92a5
```

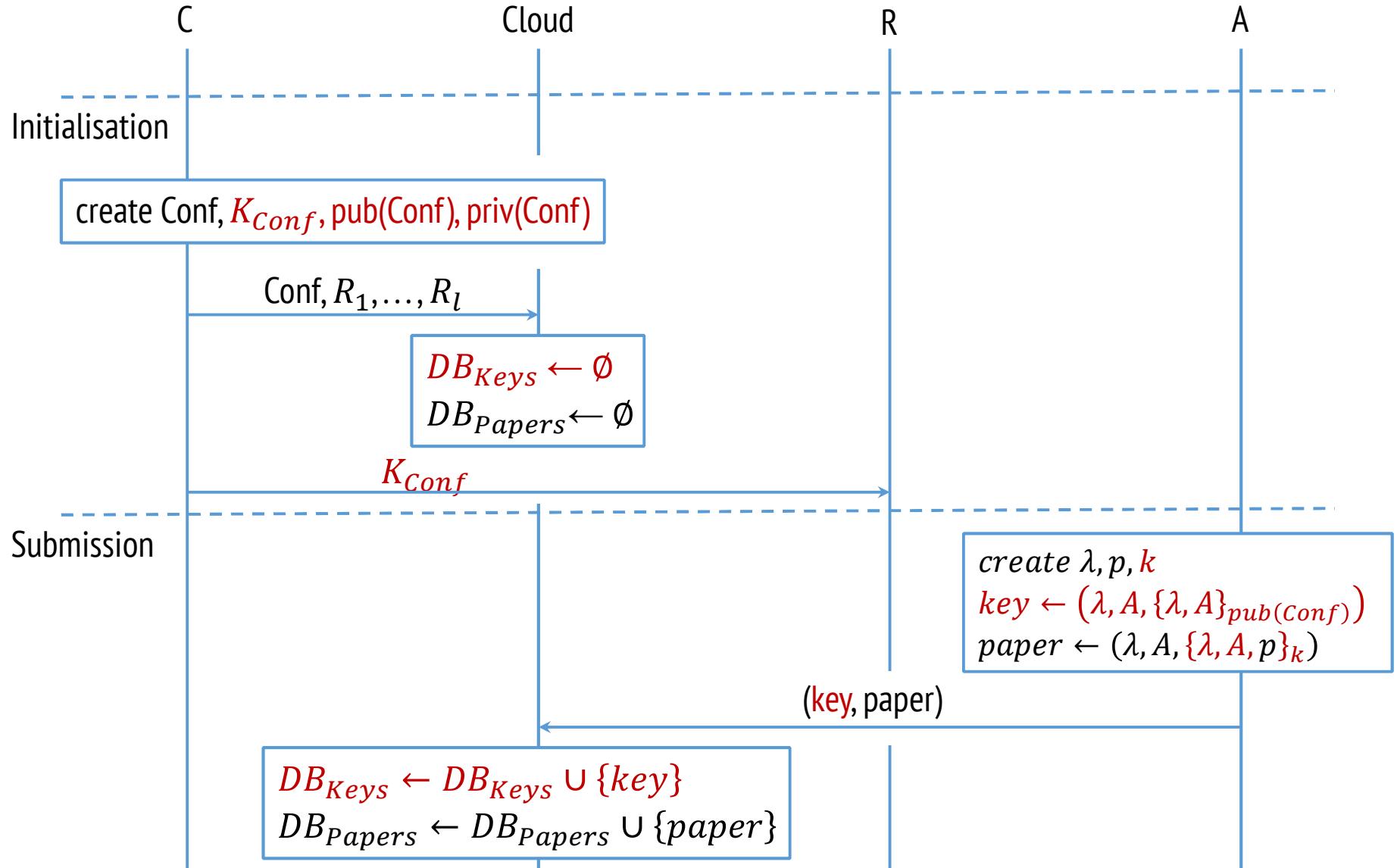
Reviewers' key [K_{Conf}]
You must E-mail this securely to anyone who you want to be a reviewer.

```
4e417316d6a660324e1a1dab5f93f83fdd7239c08ad2bc088681f7fc2fe37f70
```

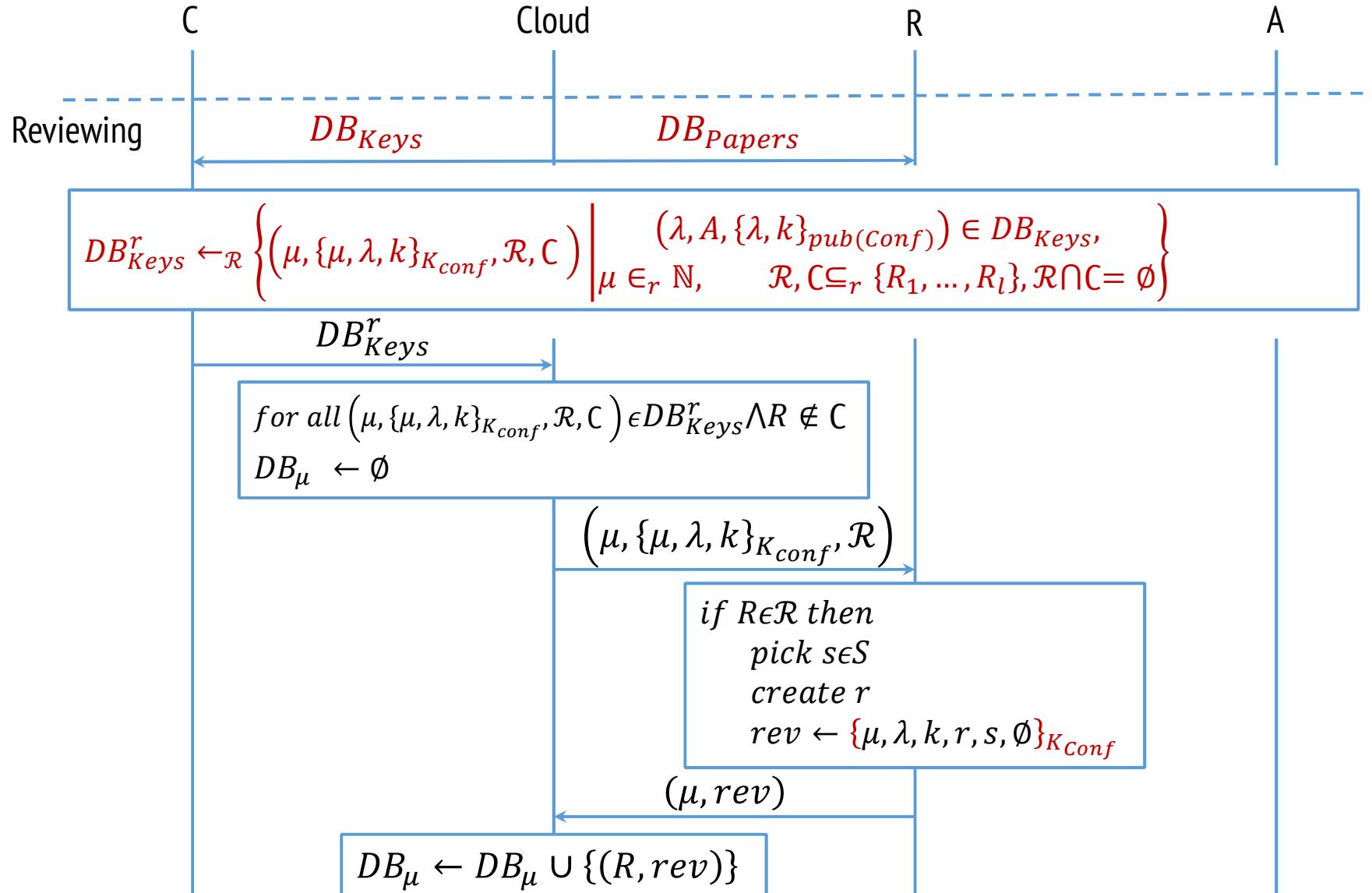
ConfiChair: An application management system



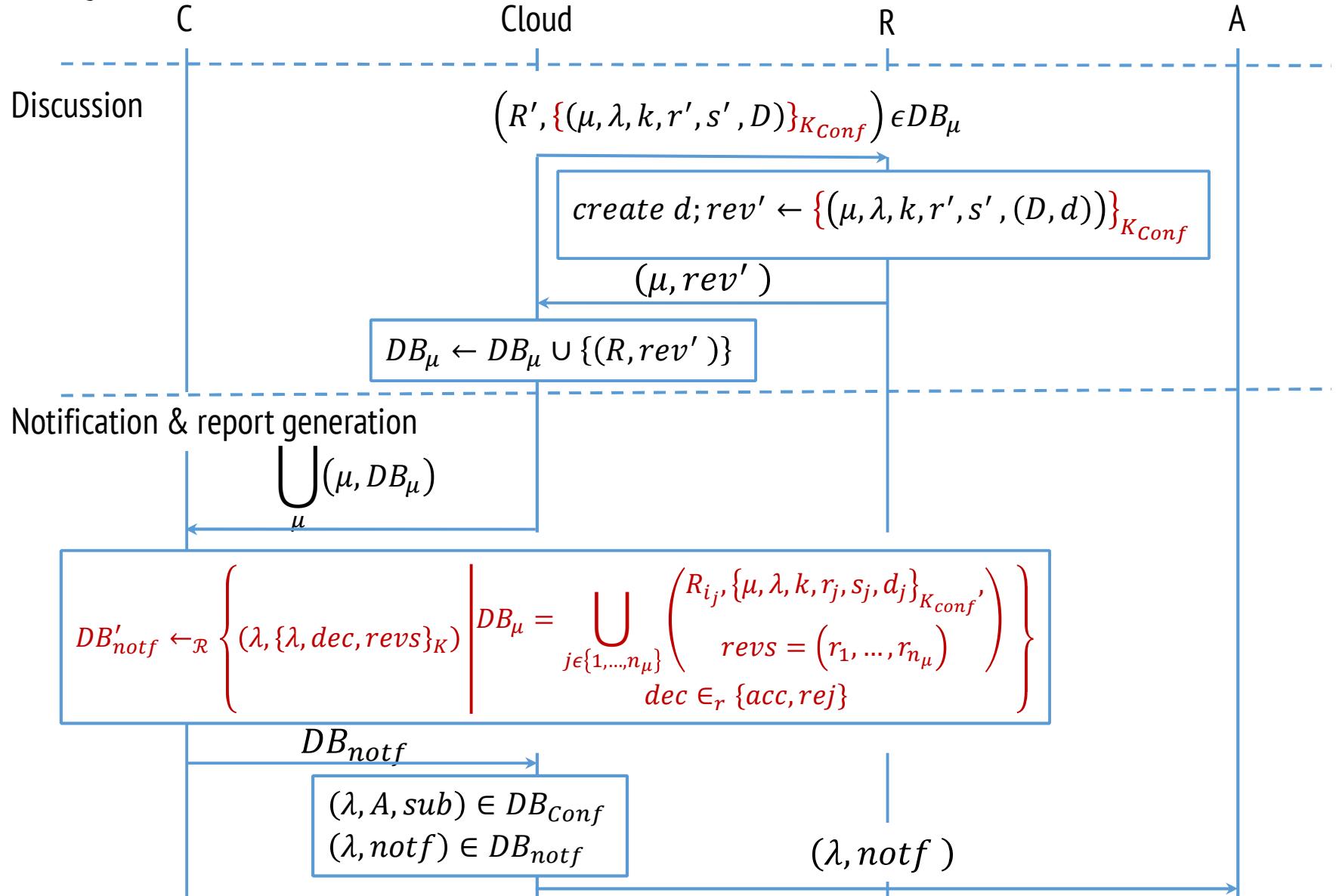
ConfiChair: An application management system



ConfiChair: An application management system



ConfiChair: An application management system



What is protected? What is the cost?

Not protected

- Accessed files
- Number of reviews
- “Intensity of the debate”

Usability

- Chair needs to re-encrypt all papers to allow reviewers to access them.
- Chair needs to re-encrypt all reviews to allow authors to see them

But wait, you said unlinkability!

- “ConfiChair does not know that a particular reviewer R reviewed a paper written by a particular author A.”
- This property is difficult to achieve because the CSP can simply follow the ciphertexts (without needing to decrypt them):
 1. Author Alice uploaded an encrypted paper
 2. Reviewer Richard later downloaded the same encrypted paper
 3. Therefore, Richard reviewed Alice’s paper
- The way we achieved the desired unlinkability property is not very good. We simply coded it so that **papers are divided into bundles of 10**, and when a reviewer asks to download one paper, he actually downloads the group of 10 that it belongs to. **This is not really secure.** It does not really satisfy the unlinkability property.

Conclusions

- We need to understand the security challenges against cloud based applications.
- Cloud based applications should be studied with multiple attacker models in mind.
- There are security and financial benefits in designing applications under the “confidentiality-from-the-cloud-provider” model.

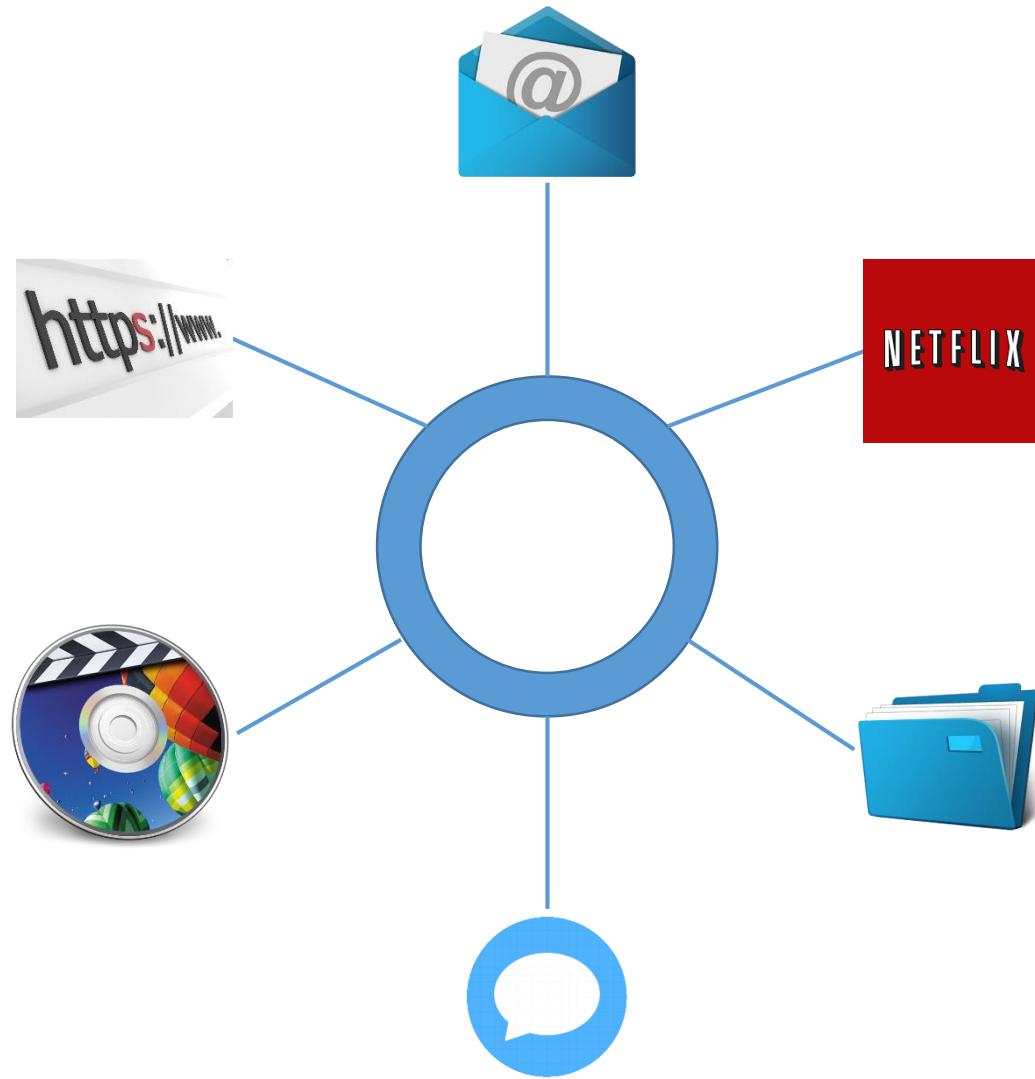
Metadata security

Mihai Ordean
Designing Secure Systems
University of Birmingham

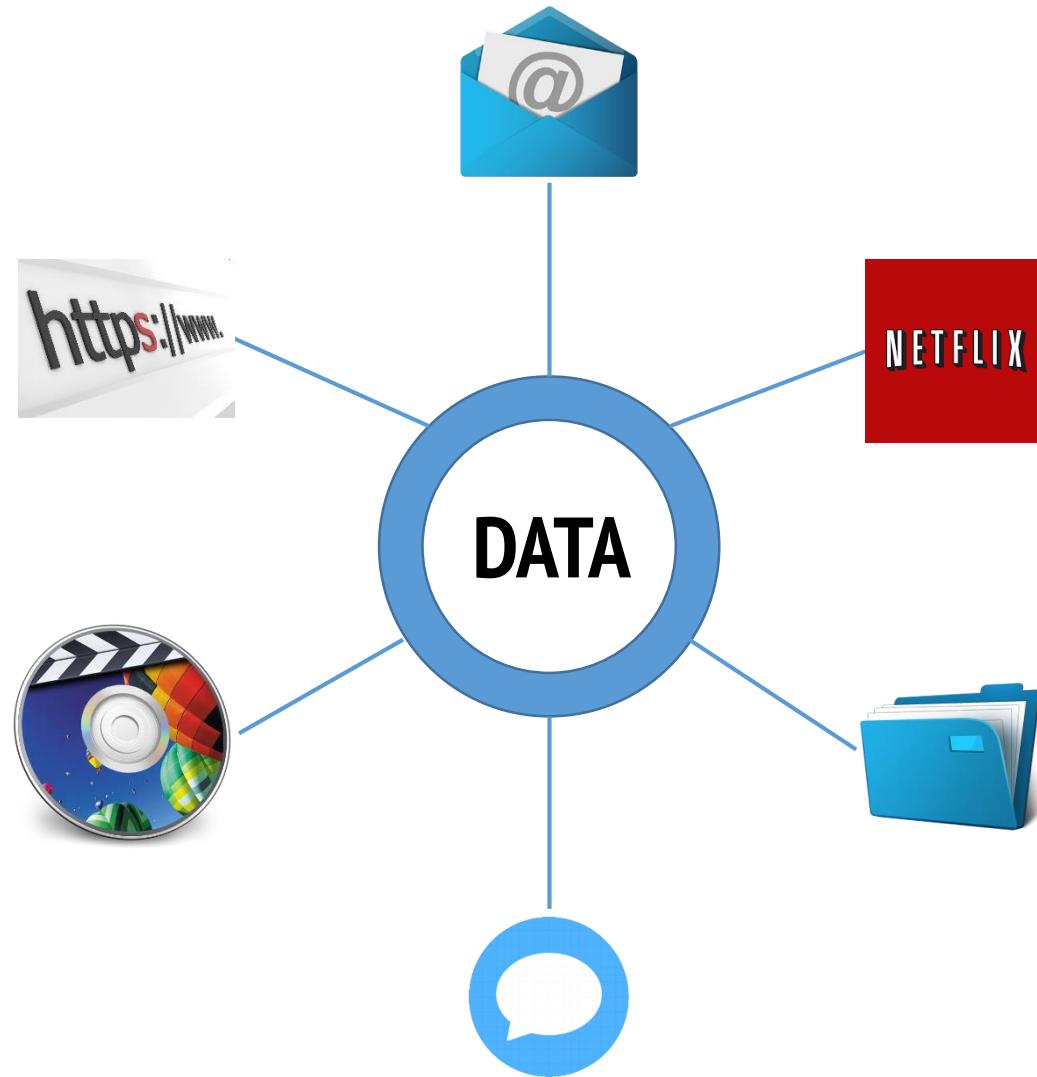
Overview

- Device security
 - Is code on the device vulnerable to exploits ? (e.g. buffer overflows)
 - Is the code authenticated ? (i.e. has not been tampered with)
- Data security (in the cloud)
 - Is the stored data is accessible to everyone? (e.g. encrypted)
 - Is the stored data authenticated?
- **Metadata security**
 - What does metadata reveal about data?
 - Can we tamper the metadata?
- Protocol security
 - Is data in transit visible?
 - Can data in transit be tampered with?

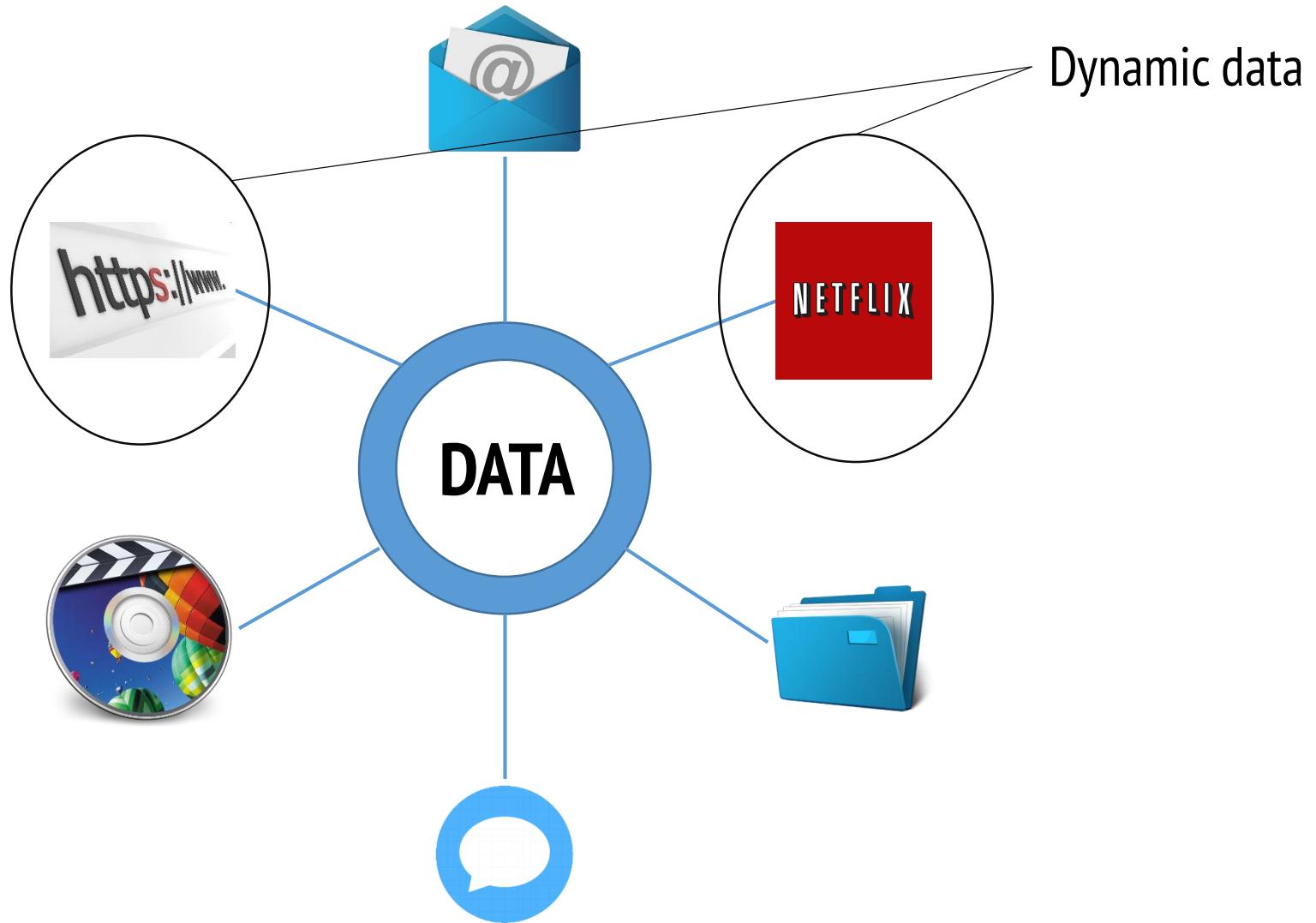
What can we encrypt?



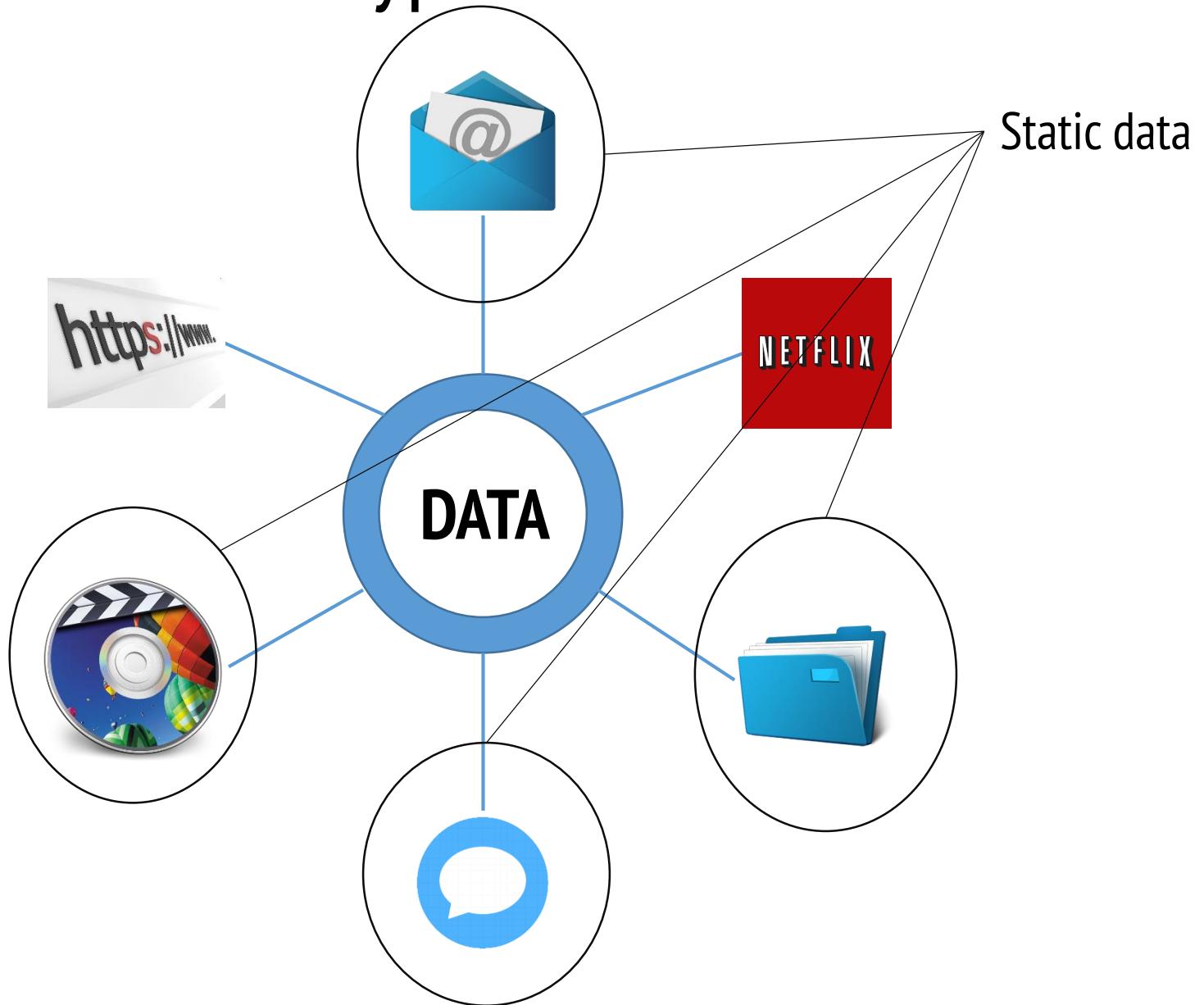
What can we encrypt?



What can we encrypt?



What can we encrypt?



Protecting dynamic data



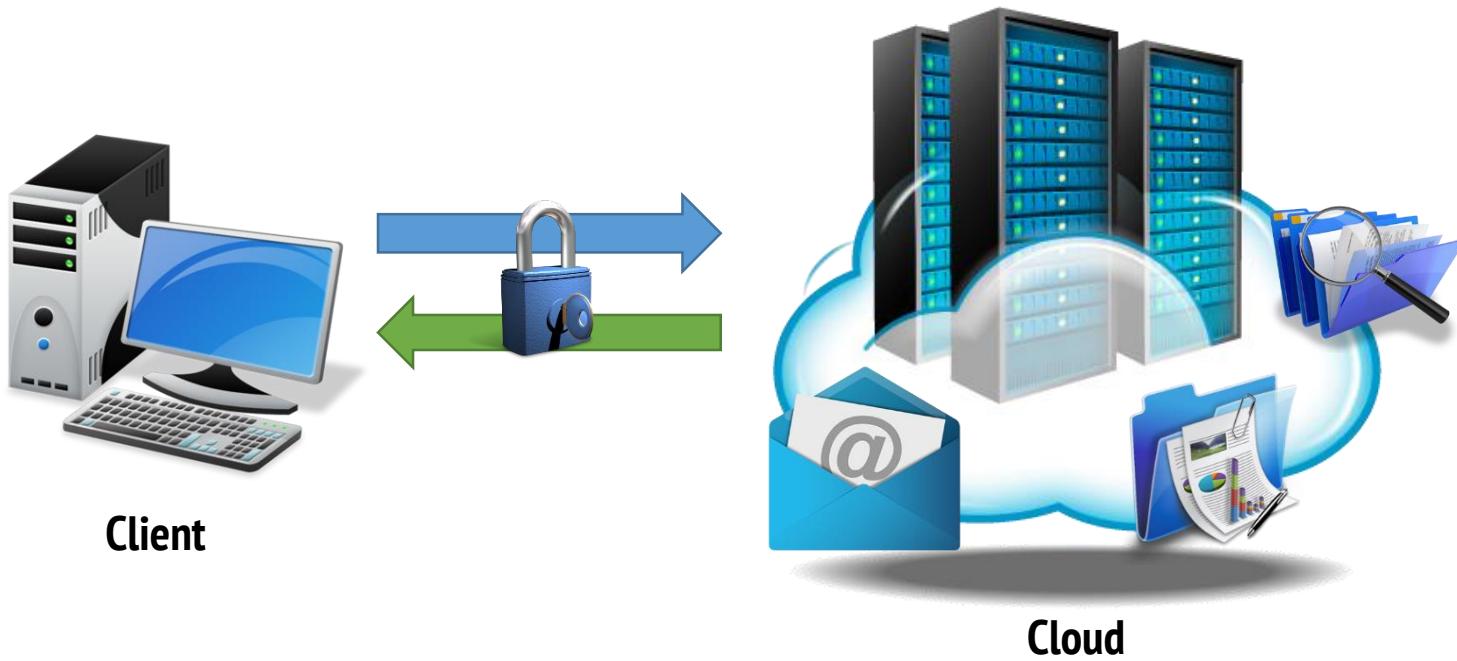
Protecting dynamic data



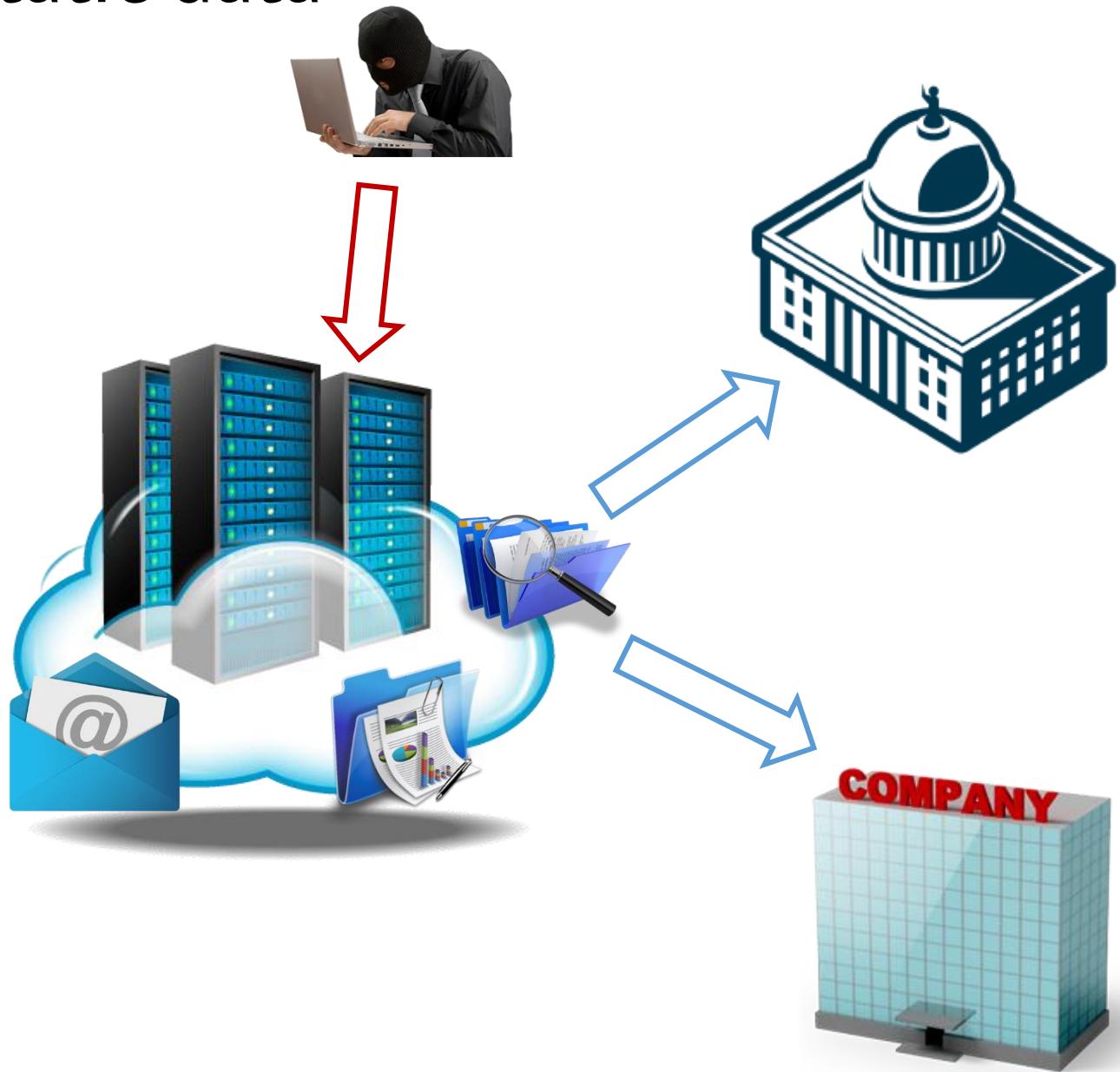
Protecting dynamic data



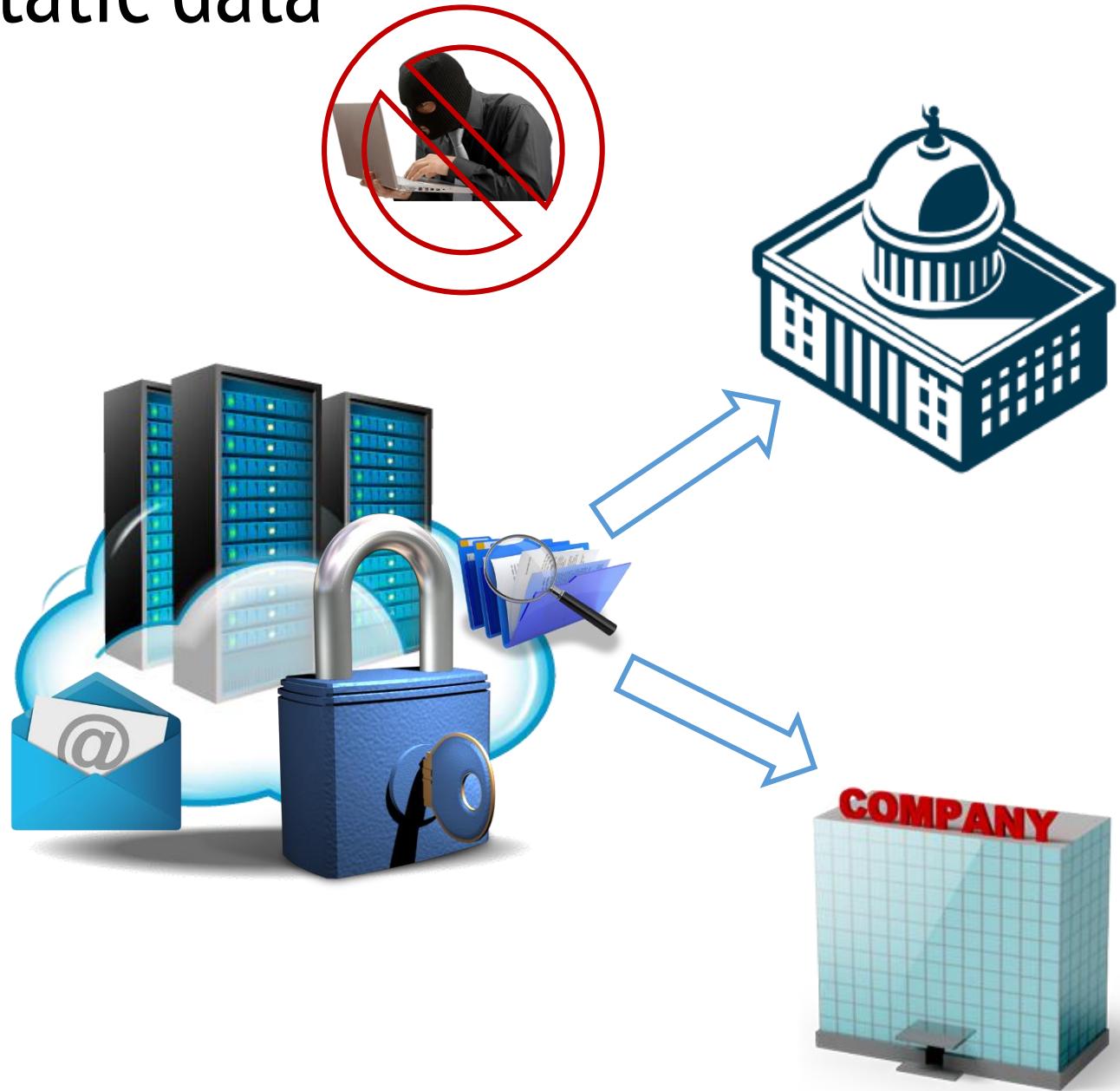
Protecting static data



Protecting static data



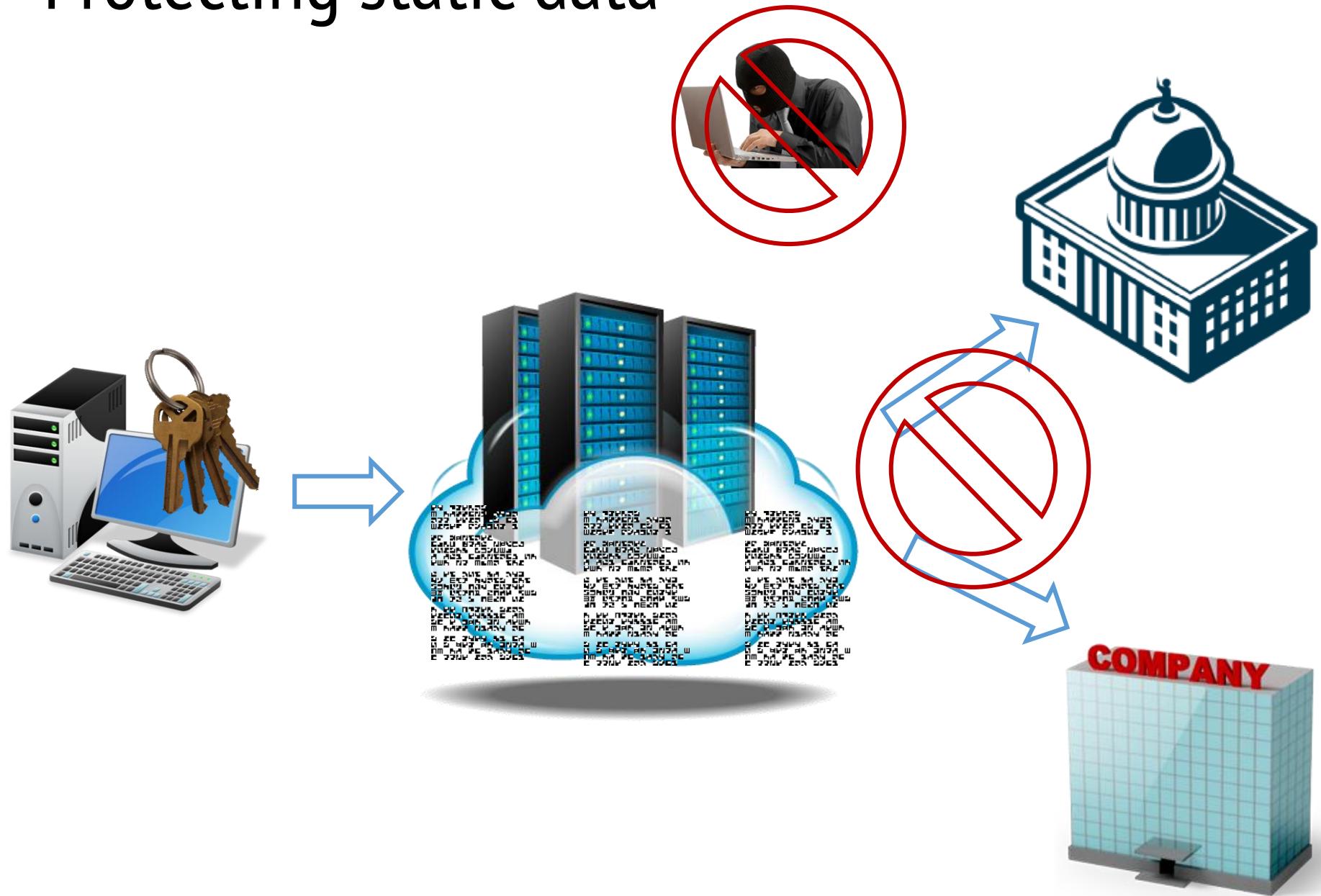
Protecting static data



Protecting static data



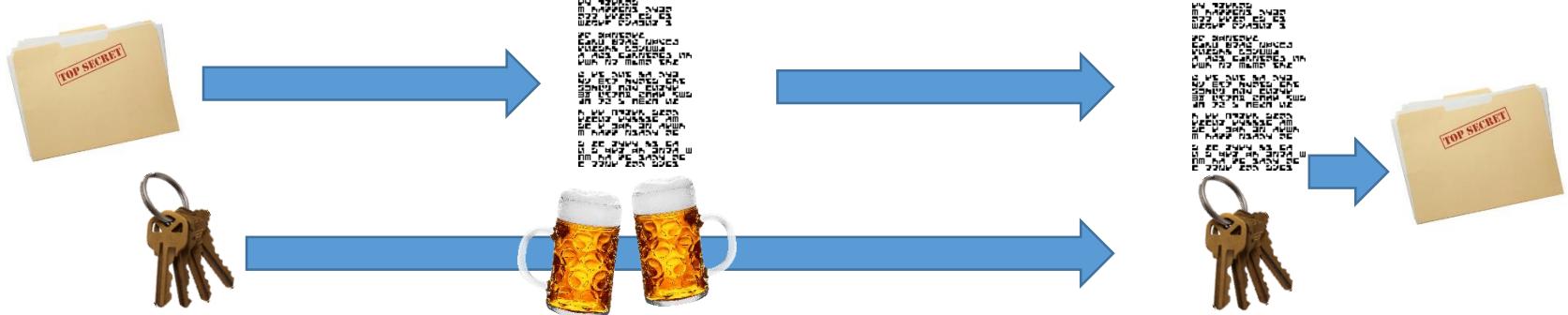
Protecting static data



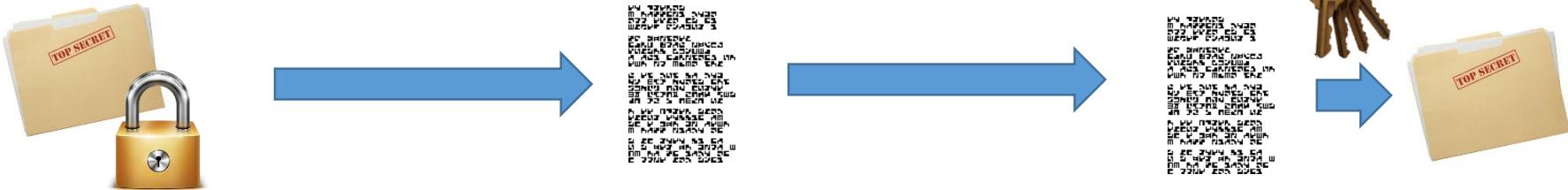
Protecting data from the cloud



Symmetric key encryption

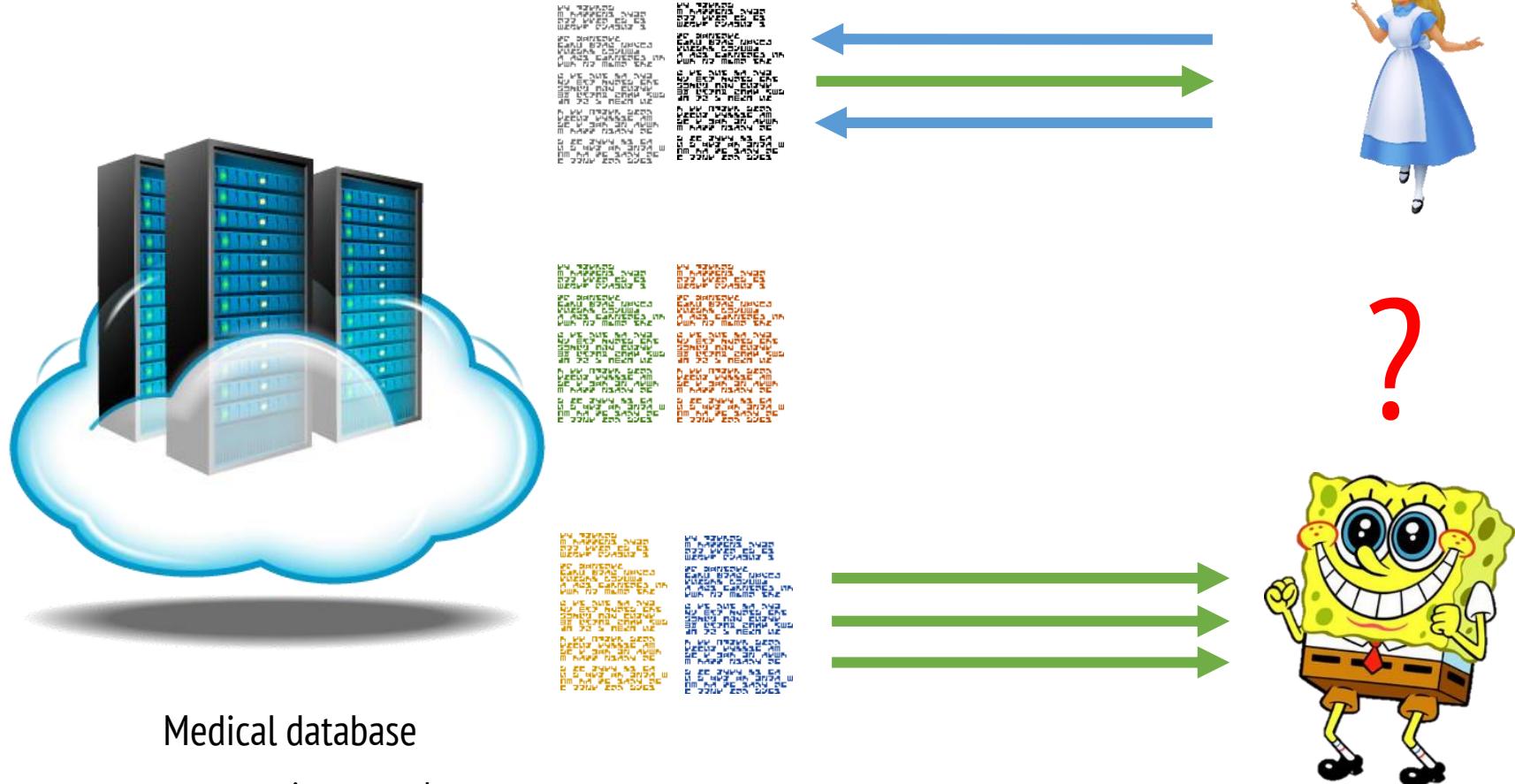


Public key encryption



Is encrypting data enough?

Analysing data access: who is the doctor?



Medical database

- patient records
- insurance records
- appointments

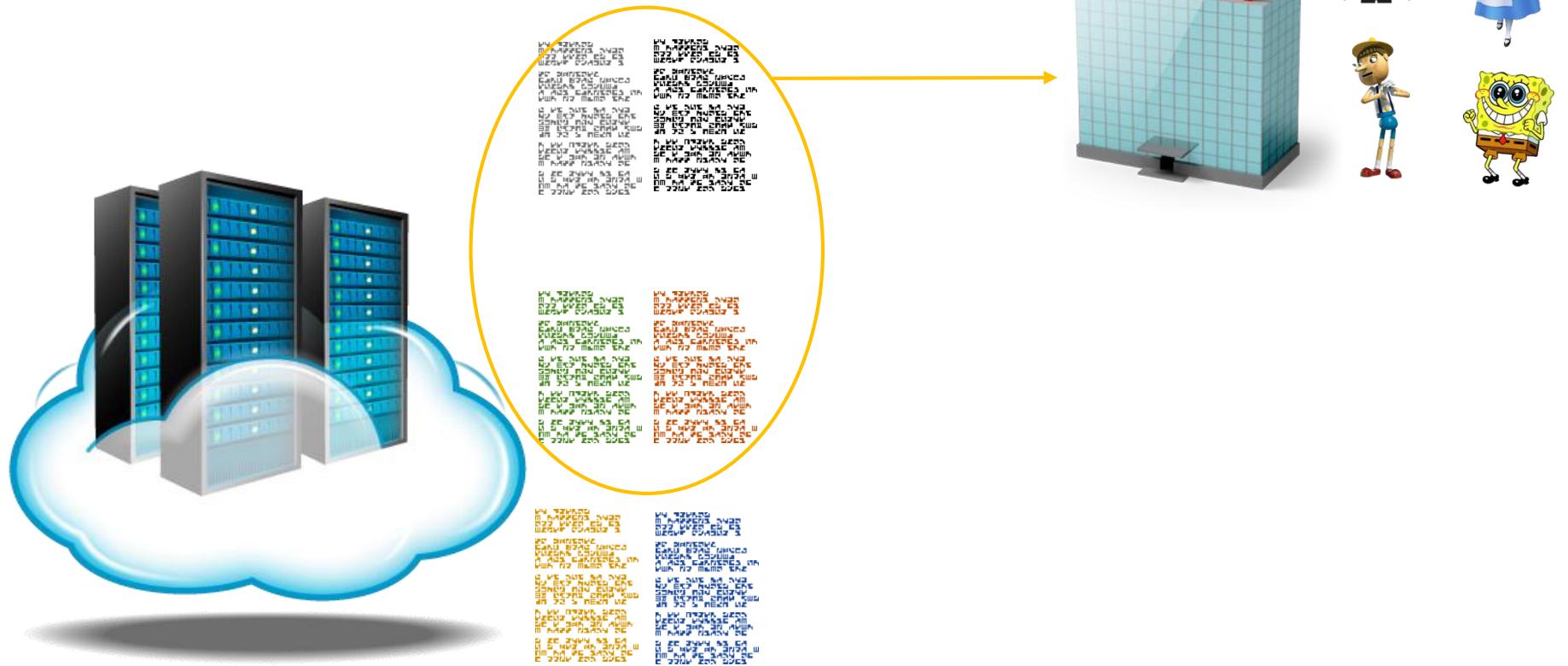
Analysing data access: who owns this ciphertext?



Medical database

- patient records
- insurance records
- appointments

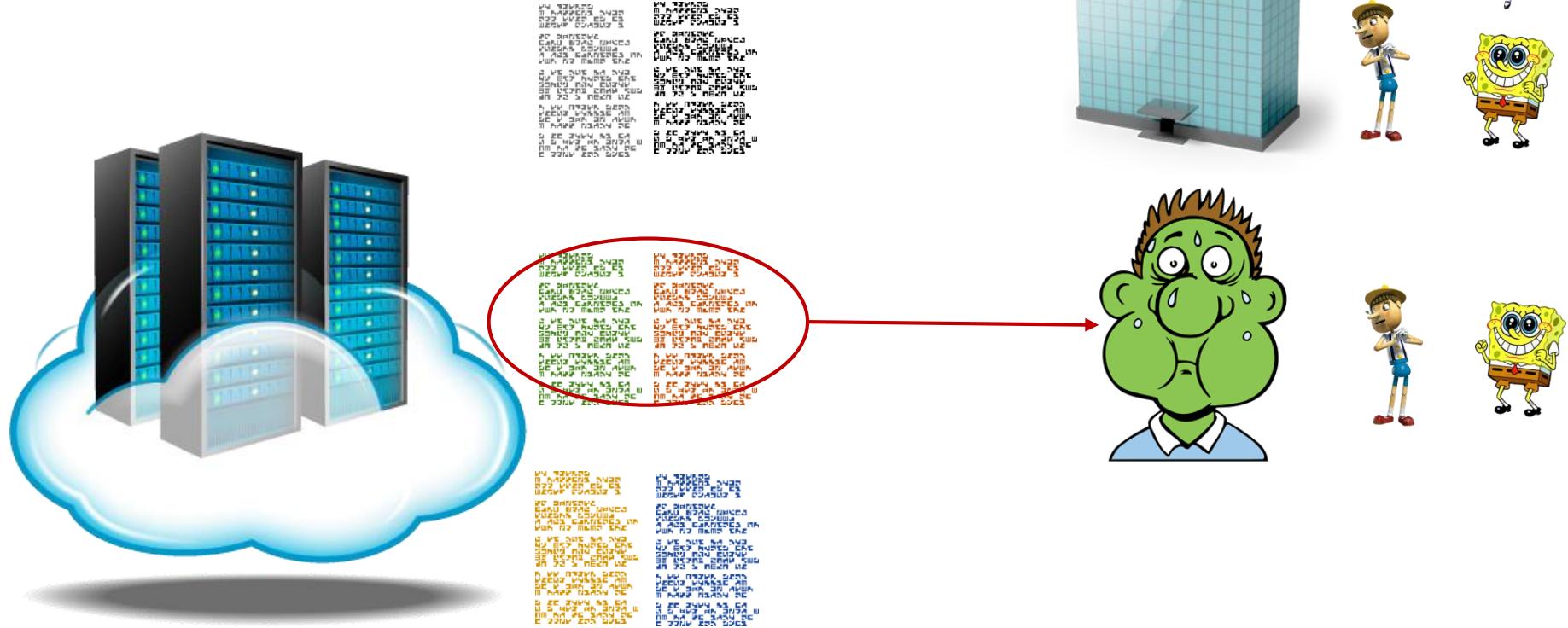
Analysing data access



Medical database

- patient records
- insurance records
- appointments

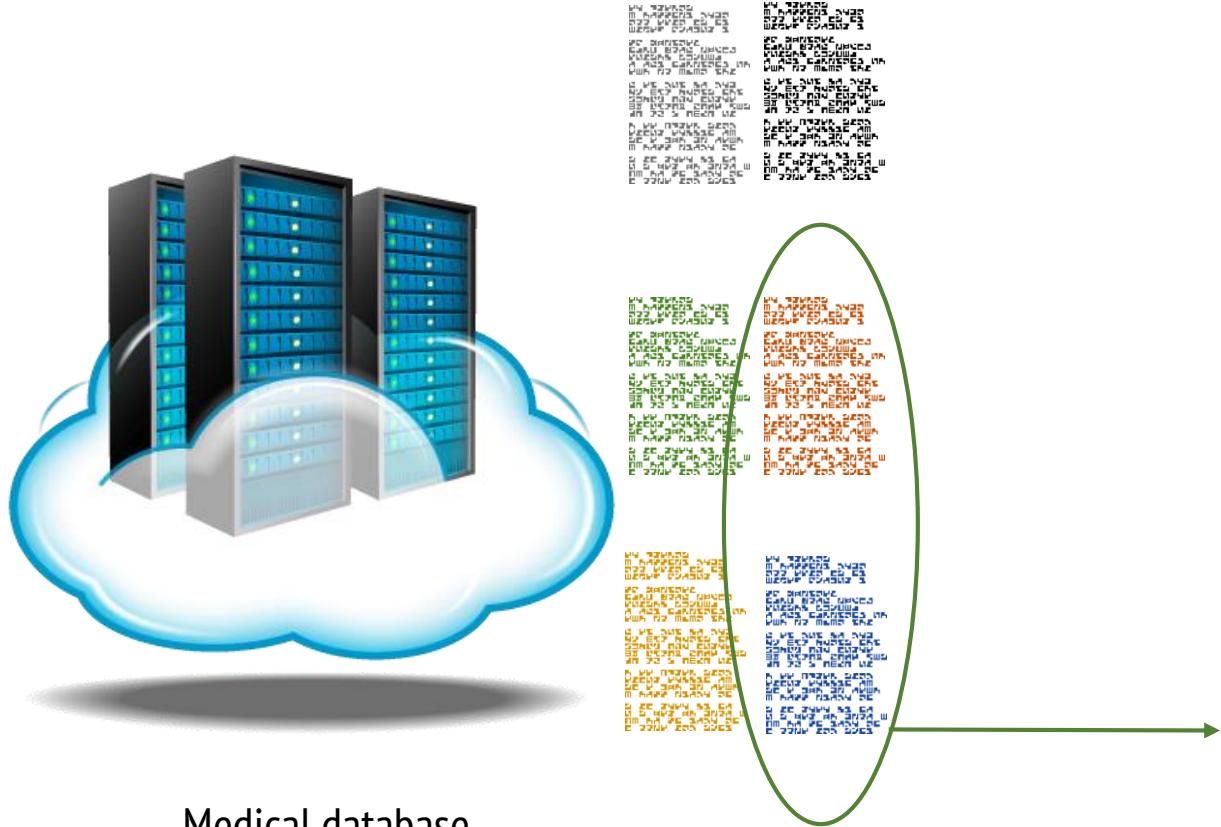
Analysing data access



Medical database

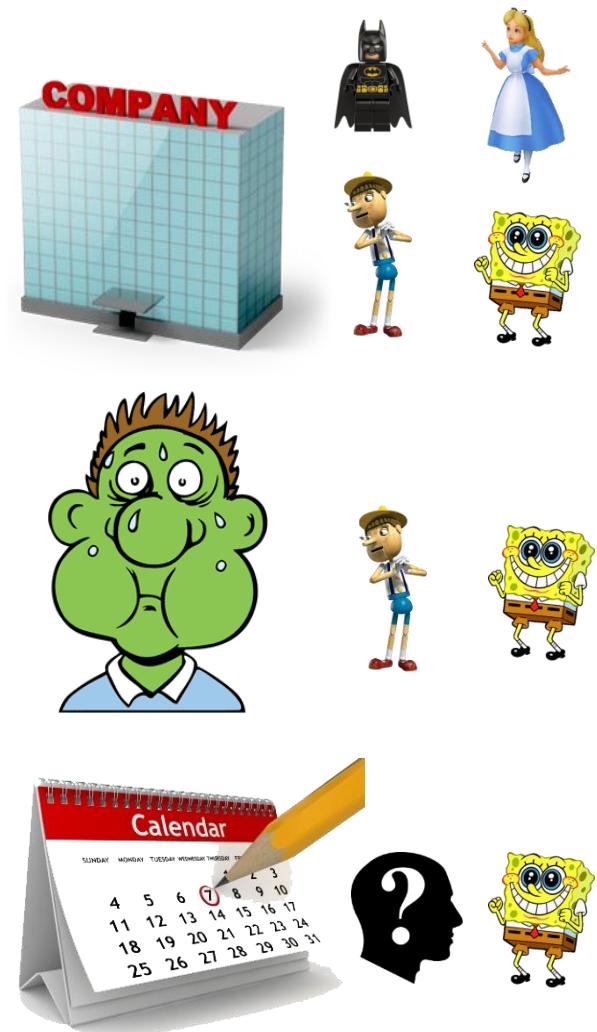
- patient records
- insurance records
- appointments

Analysing data access

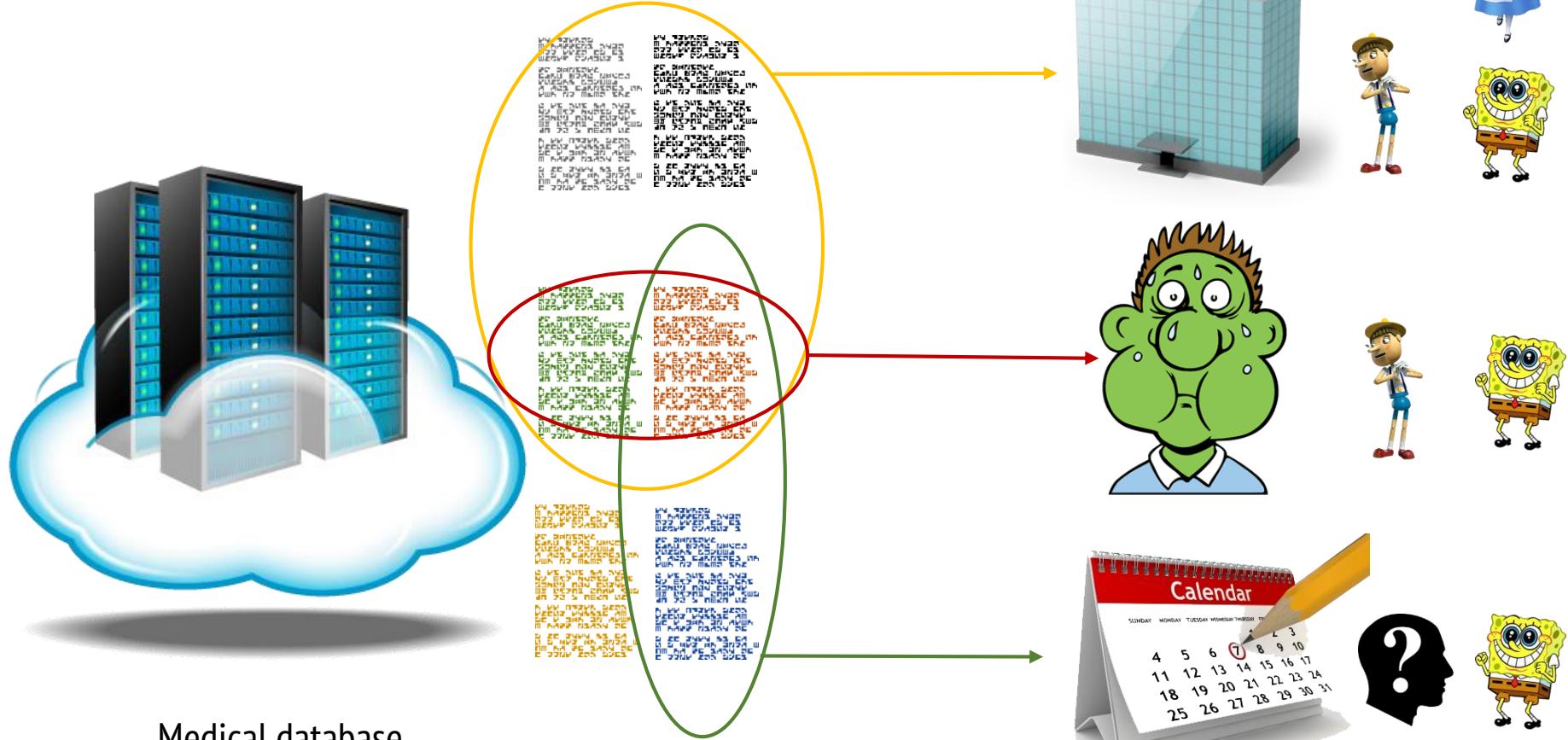


Medical database

- patient records
- insurance records
- appointments



Analysing data access



Medical database

- patient records
- insurance records
- appointments

Analysing data access

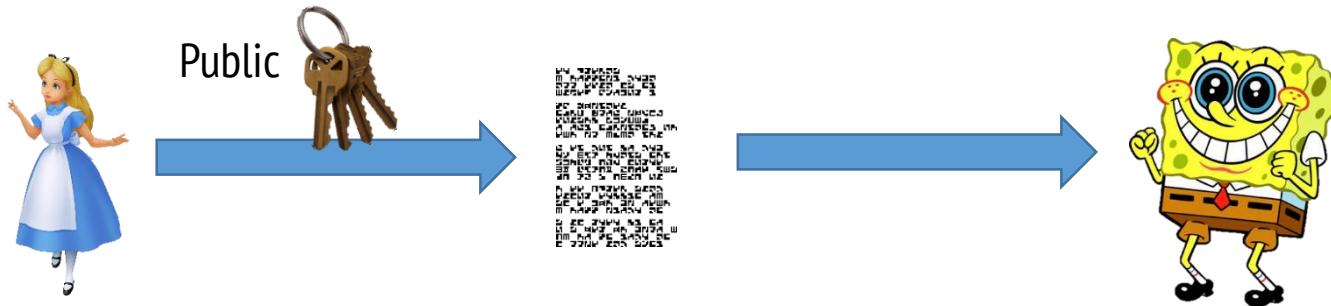


Medical database

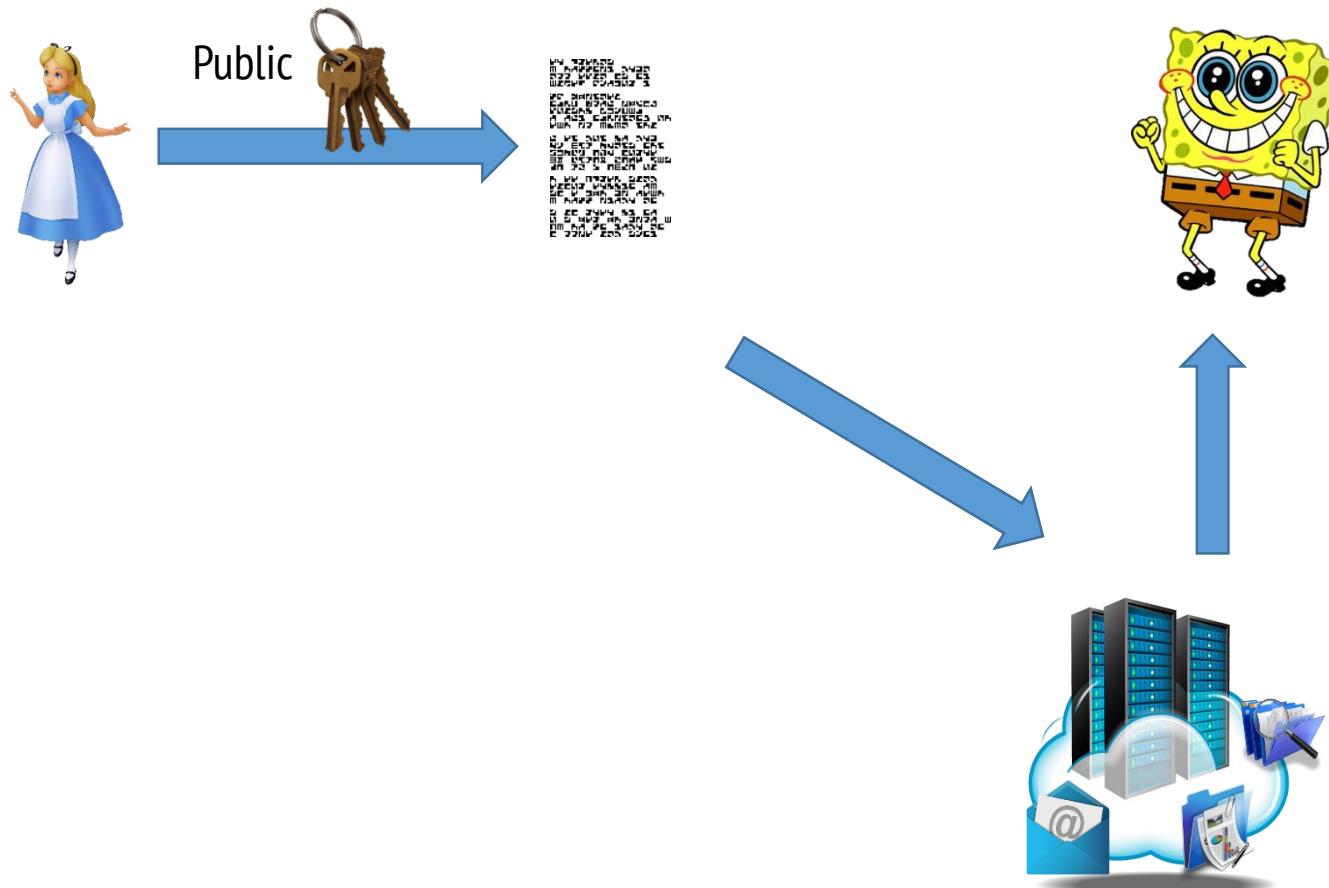
- patient records
- insurance records
- appointments

Cryptography issues

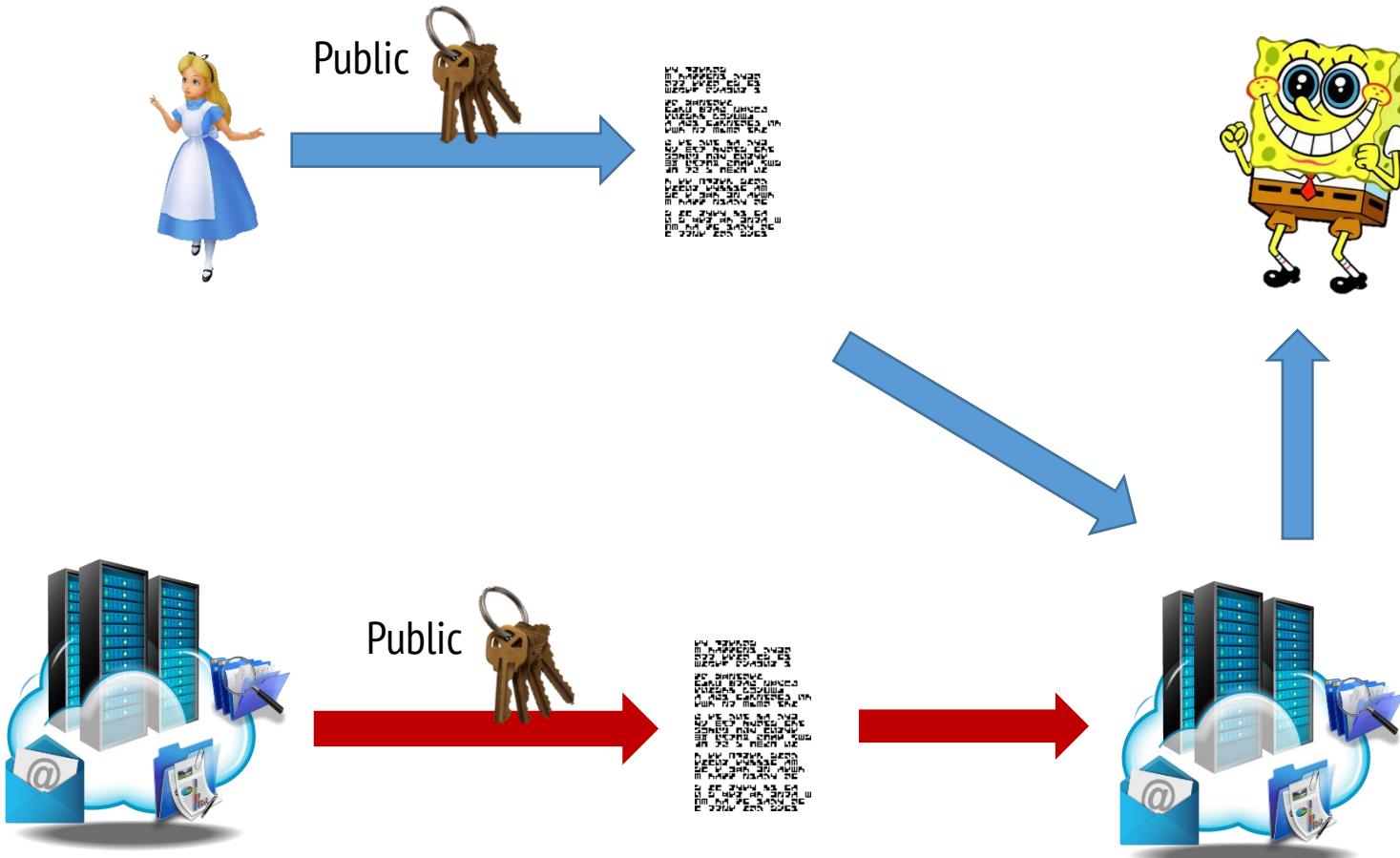
Anyone can use a public key to encrypt



Cryptography issues



Cryptography issues



Cryptography issues

- When protecting metadata, using public key crypto gives you a larger surface of attack.
- Symmetric crypto doesn't have this problem **and** is more efficient.
- Symmetric keys are difficult to share.
- Design schemes based on **symmetric keys** and use **simple public key exchange protocols** to share them.

Just using encryption is not enough

- ✓ **Content security** – the data is encrypted
- ✓ **Metadata security** – ownership information, timestamps, access rights, ciphertext length, etc.
- ✓ **Access pattern security** – when is the data accessed, who accesses the data, how is the data accessed, etc.

Searchable encryption

The challenge (in general)

- Assume we're using Gmail to communicate (with a browser).
- Assume we're using PGP to encrypt email (in browser).

- Can we decrypt email on the fly?
- Can we search through our emails?
- Who performs the search? Is it optimal?

The challenge (in general)

- Assume we're using Gmail to communicate (with a browser).
- Assume we're using PGP to encrypt email (in browser).
- Can we decrypt email on the fly?
 - YES
- Can we search through our emails?
 - Just the ones we decrypted
- Who performs the search? Is it optimal?
 - The client, in browser. Searching on the server would be the optimal choice

Solution?

- Assume we're using Gmail to communicate (with a browser).
- Assume we're using PGP to encrypt email (in browser).

- Generate a searchable index
- Store the index encrypted in the cloud

Solution?

- Assume we're using Gmail to communicate (with a browser).
- Assume we're using PGP to encrypt email (in browser).
- Generate a searchable index
- Store the index encrypted in the cloud
- Client has to download index every time
- Client still does the search, but it's much faster and can be done over all emails.

Solution?

- Assume we're using Gmail to communicate (with a browser).
- Assume we're using PGP to encrypt email (in browser).
- Generate a searchable index
- Store the index encrypted in the cloud
- ~~Client has to download index every time~~
- ~~Client still does the search, but it's much faster and can be done over all emails.~~
- THE SERVER SHOULD DO THE SEARCH! (no download, no computational effort)

Searchable Encryption

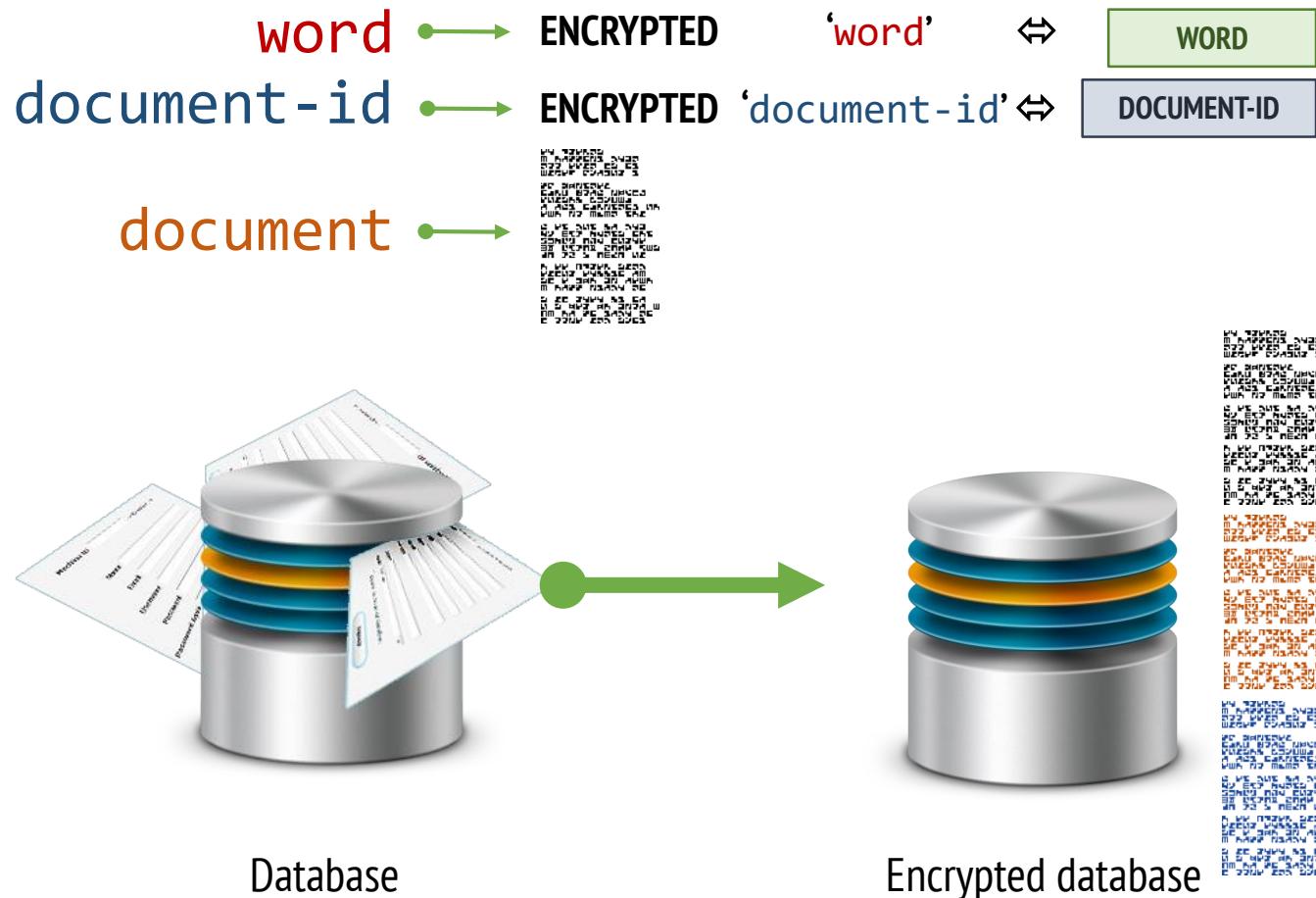


Searching



```
For each document in the database:  
  For each word in document:  
    if word = 'top-secret'  
  
      print document-id
```

Encrypting databases

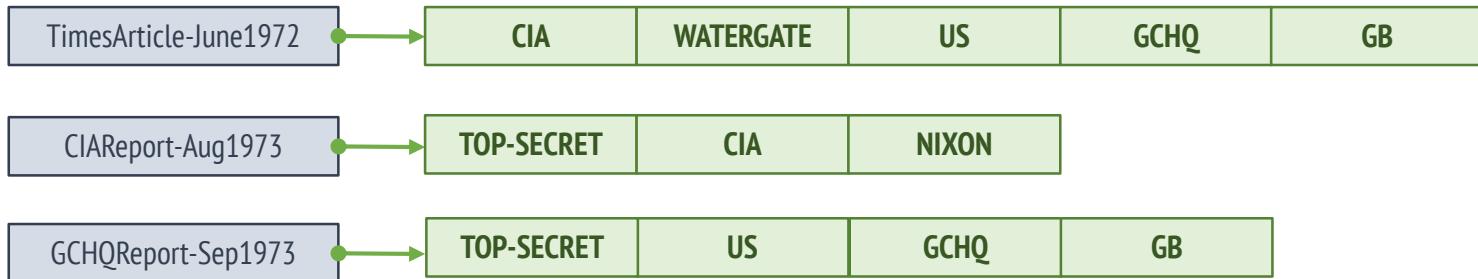


Searchable Encryption

KEYWORDS:

TOP-SECRET	CIA	WATERGATE	NIKON	US	GCHQ	GB
------------	-----	-----------	-------	----	------	----

Forward index



Efficiency of the index

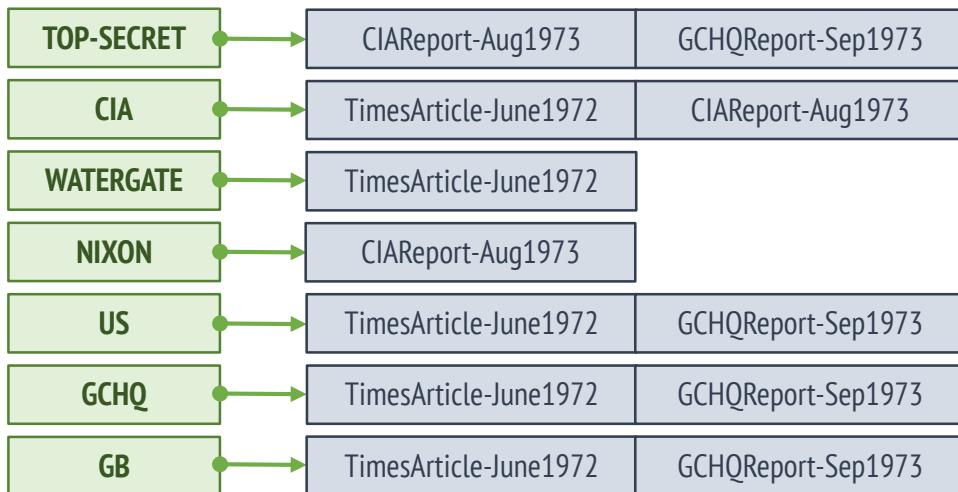
Number of **documents** increases => **time** increases

Number of **keywords** increases => **time** increases

Searchable Encryption

KEYWORDS:	TOP-SECRET	CIA	WATERGATE	NIXON	US	GCHQ	GB
-----------	------------	-----	-----------	-------	----	------	----

Inverted index



Efficiency of the index

Number of **keywords** increases => **time** increases

What do we want to protect?

What we search for

KEYWORDS: TOP-SECRET CIA WATERGATE ...

What is the result of the search query

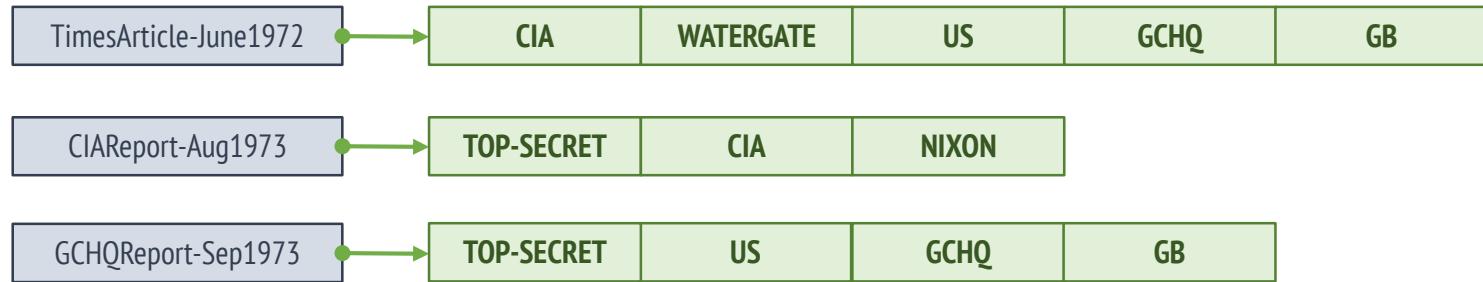
DOCUMENT NAMES: CIAReport-Aug1973 GCHQReport-Sep1973

How often we search for something

1: TOP-SECRET
2: CIA
...
n: TOP-SECRET

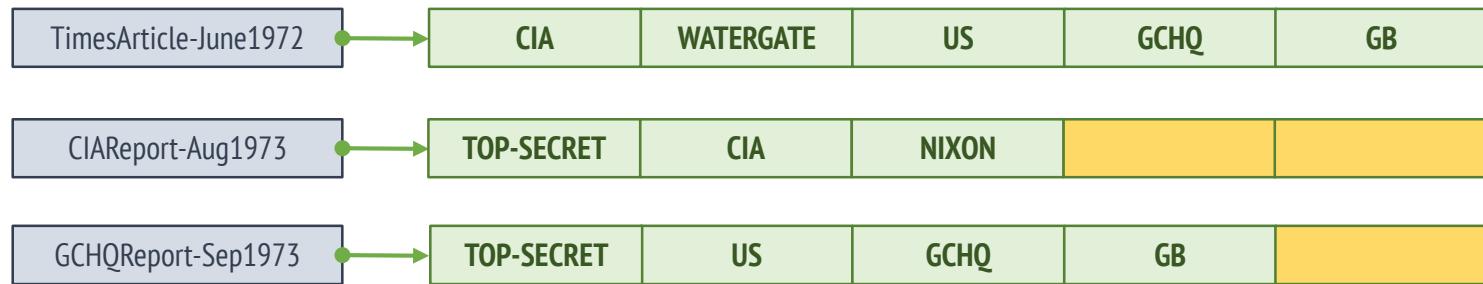
Padding

Forward index



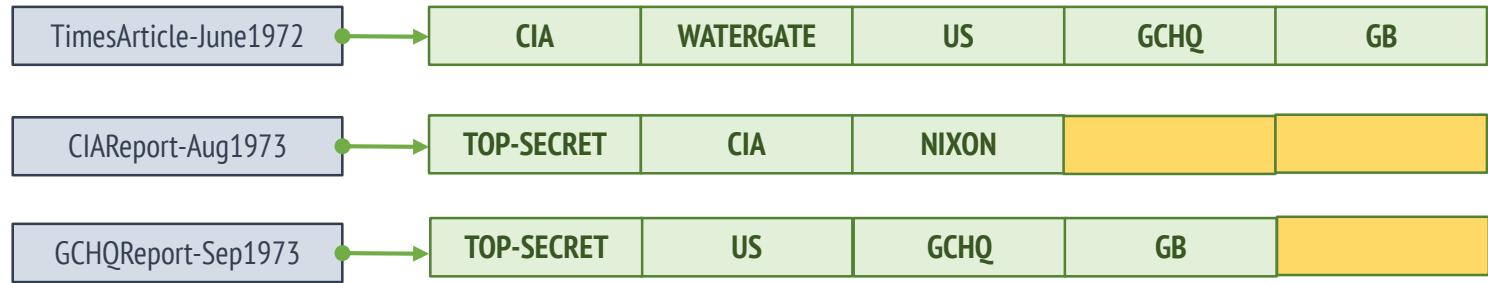
Padding

Forward index

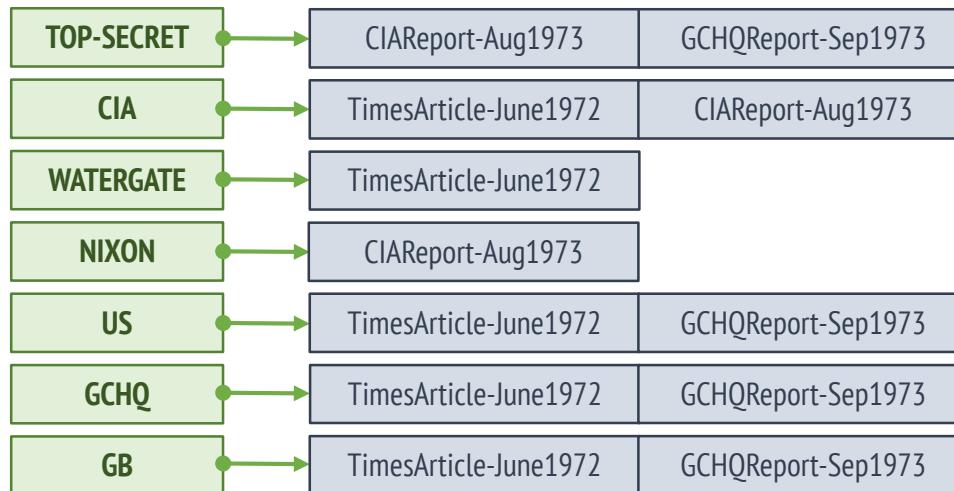


Padding

Forward index

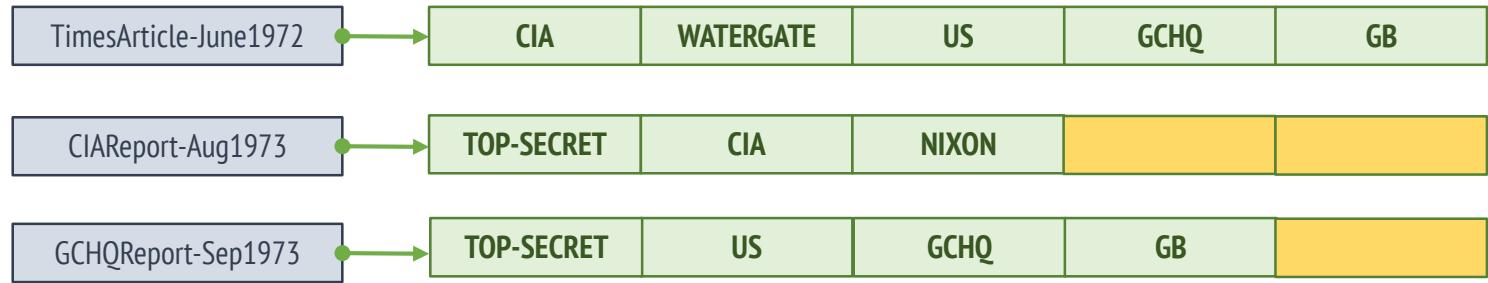


Inverted index

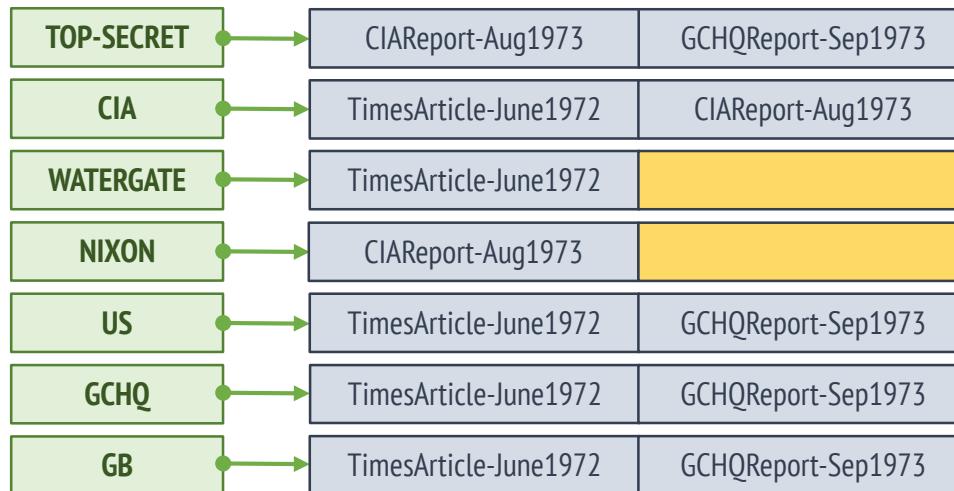


Padding

Forward index

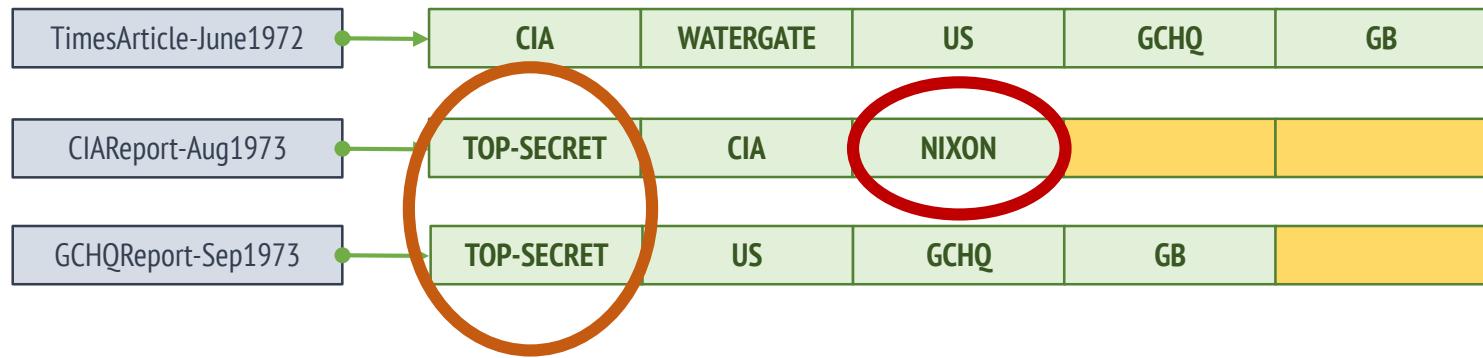


Inverted index



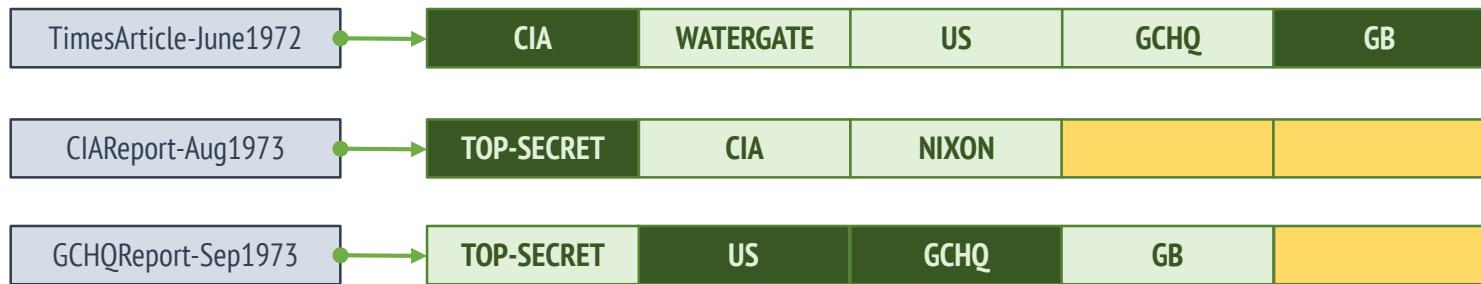
Intersections

Forward index



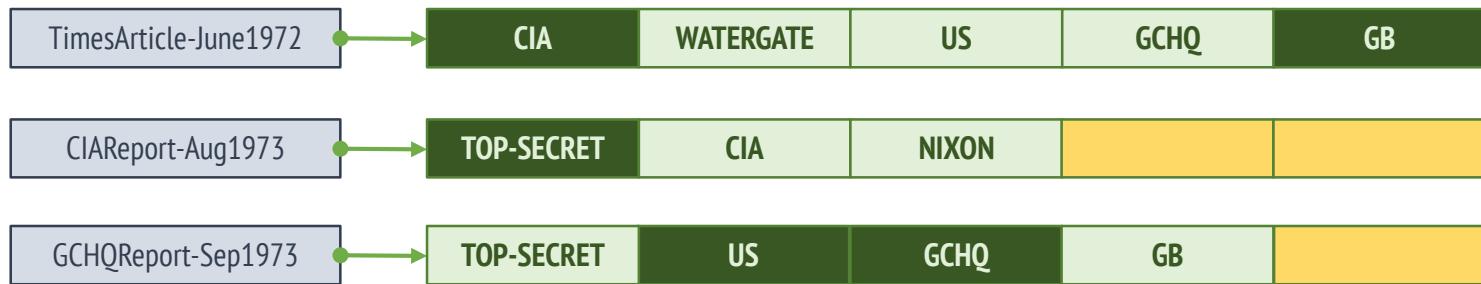
Intersections

Forward index

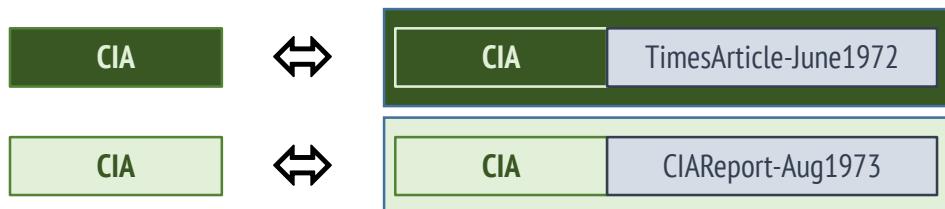


Intersections

Forward index



We want

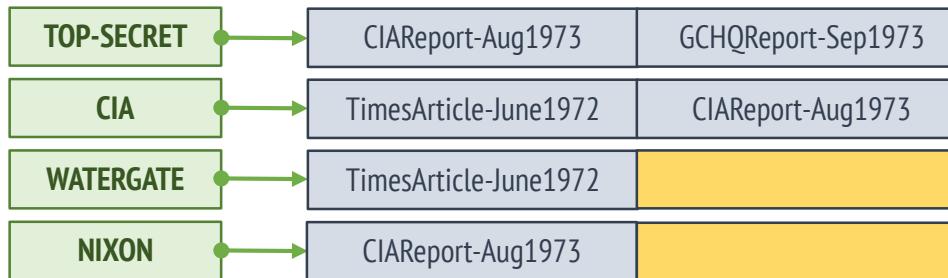


Server the computation

1. Client work needs to be as low as possible.
2. Server needs to do most of the search work.

Secure searching

Inverted index:



Symmetric key searchable encryption index:

ENC. DOC. NAMES:

CIAReport-Aug1973

GCHQReport-Sep1973

TimesArticle-June1972

INDEX:

CIAReport-Aug1973

TOP-SECRET

GCHQReport-Sep1973

TOP-SECRET

TOP-SECRET

TimesArticle-June1972

CIA

CIAReport-Aug1973

CIA

CIA

...

...

...

...

...

Secure searching

Server has:

ENC. DOC. NAMES:

CIAReport-Aug1973

GCHQReport-Sep1973

TimesArticle-June1972

INDEX:

CIAReport-Aug1973

TOP-SECRET

TimesArticle-June1972

CIA

GCHQReport-Sep1973

TOP-SECRET

TOP-SECRET

TOP-SECRET

CIAReport-Aug1973

CIA

TOP-SECRET

CIA

...

...

...

...

TOP-SECRET

...

Search term:

TOP-SECRET

Secure searching

Server has:

ENC. DOC. NAMES:

CIAReport-Aug1973

GCHQReport-Sep1973

TimesArticle-June1972

INDEX:

CIAReport-Aug1973

TOP-SECRET

TimesArticle-June1972

CIA

GCHQReport-Sep1973

TOP-SECRET

TOP-SECRET

TOP-SECRET

CIAReport-Aug1973

CIA

TOP-SECRET

...

...

...

...

...

Search term:

TOP-SECRET

Server computation:

CIAReport-Aug1973

GCHQReport-Sep1973

TimesArticle-June1972

Secure searching

Server has:

ENC. DOC. NAMES:

CIAReport-Aug1973

GCHQReport-Sep1973

TimesArticle-June1972

INDEX:

CIAReport-Aug1973

TOP-SECRET

TimesArticle-June1972

CIA

GCHQReport-Sep1973

TOP-SECRET

██████████

TOP-SECRET

CIAReport-Aug1973

CIA

██████████

CIA

██████████

...

...

██████████

...

...

██████████

...

Search term:

TOP-SECRET

Server computation:

CIAReport-Aug1973

TOP-SECRET

GCHQReport-Sep1973

TOP-SECRET

TimesArticle-June1972

TOP-SECRET

Secure searching

Server has:

ENC. DOC. NAMES:

CIAReport-Aug1973

GCHQReport-Sep1973

TimesArticle-June1972

INDEX:

CIAReport-Aug1973

TOP-SECRET

TimesArticle-June1972

CIA

GCHQReport-Sep1973

TOP-SECRET

██████████

TOP-SECRET

CIAReport-Aug1973

CIA

██████████

██████████

...

██████████

...

██████████

...

Search term:

TOP-SECRET

Server computation:

CIAReport-Aug1973

TOP-SECRET

GCHQReport-Sep1973

TOP-SECRET

Time Article-June1972

TOP-SECRET



Secure searching

Server has:

ENC. DOC. NAMES:

CIAReport-Aug1973

GCHQReport-Sep1973

TimesArticle-June1972

INDEX:

CIAReport-Aug1973

TOP-SECRET

TimesArticle-June1972

CIA

GCHQReport-Sep1973

TOP-SECRET

TOP-SECRET

TOP-SECRET

CIAReport-Aug1973

CIA

TOP-SECRET

...

...

...

...

...

Search term:

TOP-SECRET

Server computation:

CIAReport-Aug1973

TOP-SECRET

GCHQReport-Sep1973

TOP-SECRET

Time Article-June1972

TOP-SECRET

Result:

CIAReport-Aug1973

GCHQReport-Sep1973

Performance

Example 1 - OXT:

[Cash-Jarecki-Jutla-Krawczyk-Rosu-Steiner13]

- Encrypted database size: 13GB
- DB Contents: 1.5 million emails & attachments
- **Avg. search time: less than 500ms**

Example 2 – 2Lev:

[Cash-Jaeger-Jarecki-Jutla-Krawczyk-Steiner-Rosu14]

- Encrypted database size: 900GBs
- Setup time: 16 hours
- **Avg. query time: less than 200ms**

Searchable encryption limitations

- Encrypted search term is deterministic
- Only search patterns are hidden
- Setting up the index requires a significant amount of time
- Most schemes do not support index extensions

DISCLAIMER :-)

- Searchable encryption **solves** problems related to the security of the search index.
- Searchable encryption **does not solve** problems related to the security of subsequent data retrieval.

Even though the response to the search query has been done in a privacy preserving manner, the server can still learn what the result of the query was by simply observing what the client does next, e.g. **monitor the emails the client is going to access/download**.

Oblivious RAM

Oblivious RAM (ORAM)

- A cryptographic primitive originally designed to prevent reverse engineering by hiding access to memory.
- It has since repurposed for use in client-server scenarios with the purpose of hiding the ways in which data is accessed from the server.

ORAM security requirements

Hide **DATA CONTENTS** and:

1. Hide **which** data is accessed (e.g. My DSS course)
2. Hide **when** data was last accessed (e.g. 5mins ago)
3. Hide **how** data is accessed (i.e. read or write access)
4. Hide **how frequently** data is accessed (e.g. every day at 12pm)
5. Hide **access pattern** (e.g. sequential, random)

ORAM

- Uses symmetric encryption (e.g. AES) to encrypt small data structures (e.g. data ‘buckets’).
- Replaces read and write operations (i.e. download and upload) with a generic **access** operation which contains both a read and a write operation.
- The **access operation** has a significant overhead in order to disguise the exact data being accessed.

ORAM components

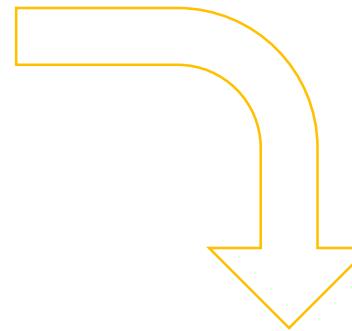
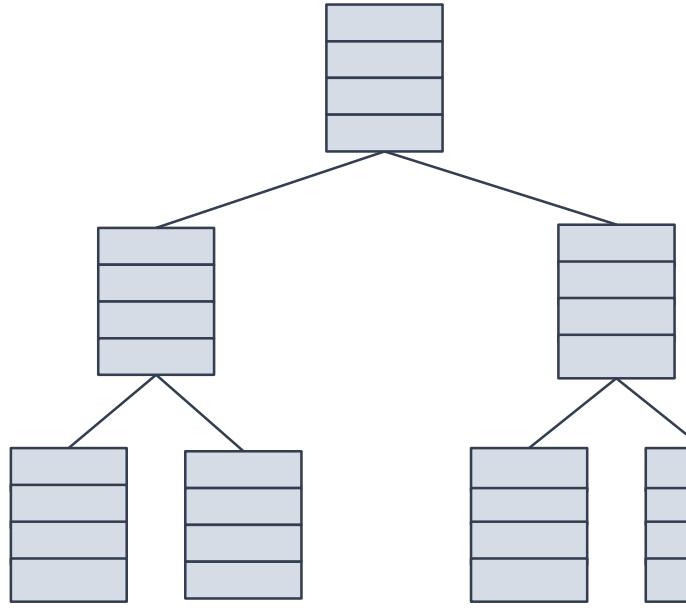


- Client stores an AES key
- Client stores a map
- Client stores a stash, i.e., a local cache structure



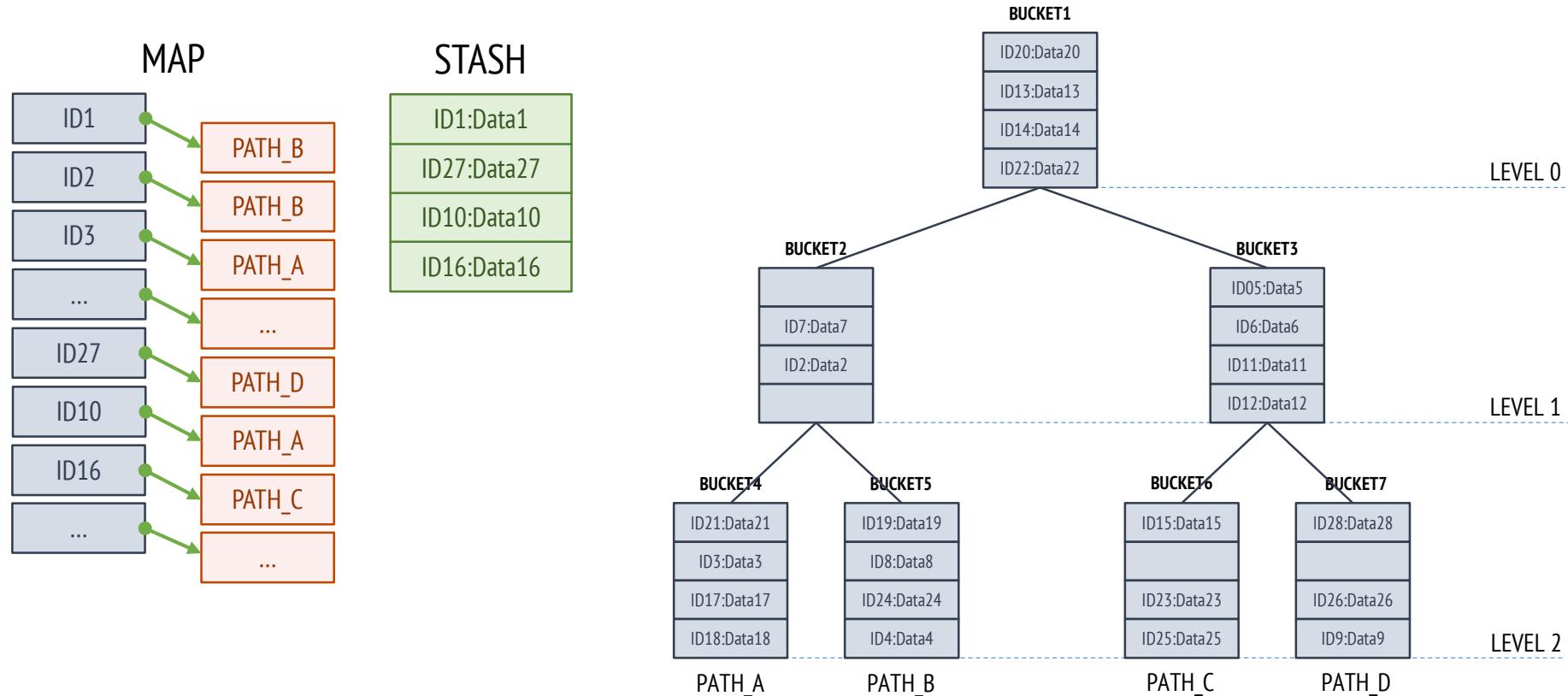
- Server storage is structured as a binary tree.
- On disk the three is stored as a flat data structure

Flat binary tree?



PathORAM

[Stefanov-van Dijk-Shi-Chan-Fletcher-Ren-Yu-Devadas13]



PathORAM access

READ:

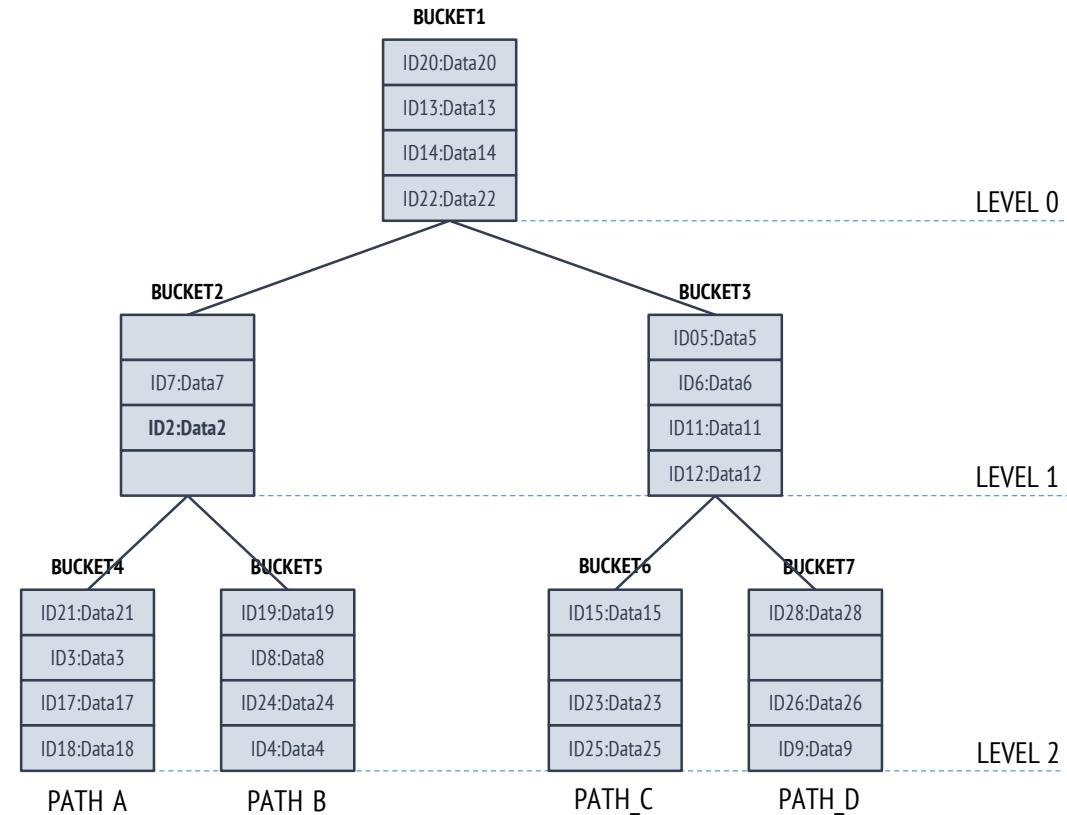
ID2:empty

MAP



REQUEST:

PATH_B



PathORAM access

READ:

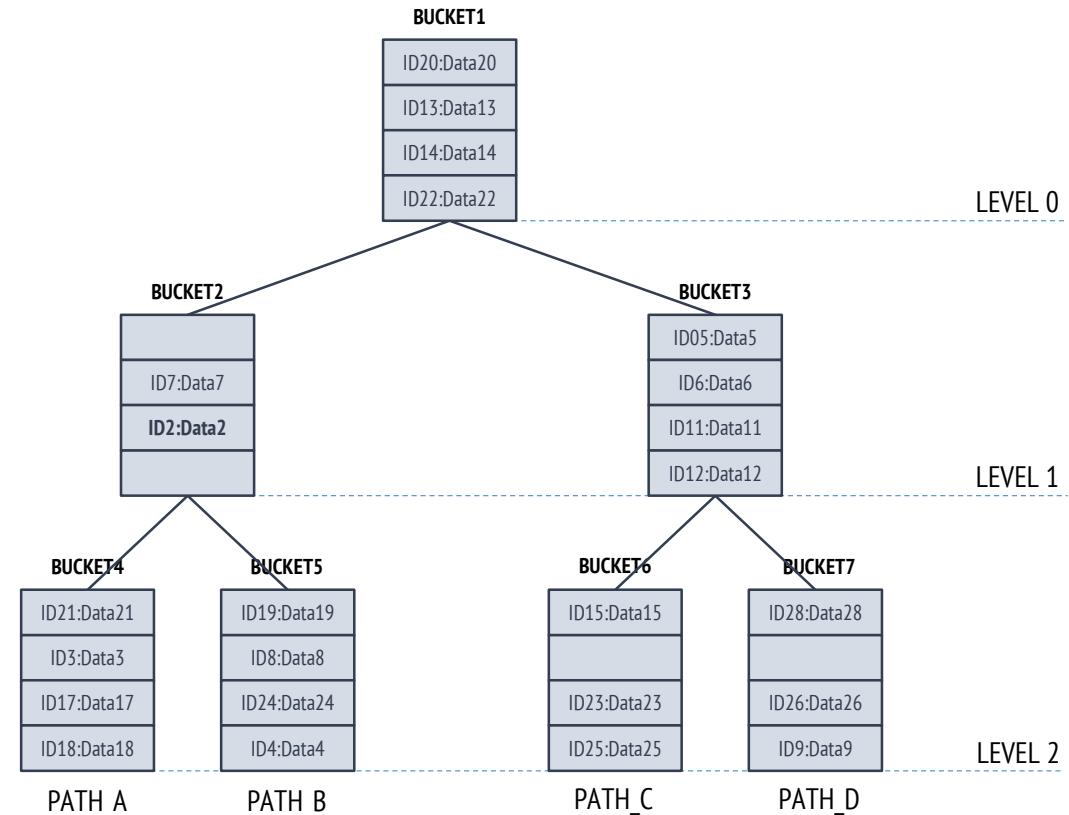
ID2:empty

MAP



REQUEST:

PATH_B



RECEIVE:

PATH_B											
Bucket5				Bucket2				Bucket1			
ID4:Data4	ID24:Data24	ID8:Data8	ID19:Data19		ID2:Data2	ID7:Data7		ID22:Data22	ID14:Data14	ID13:Data13	ID20:Data20

PathORAM access

READ:

ID2:empty

REQUEST:

PATH_B

RECEIVE:

PATH_B

Bucket5

Bucket2

Bucket1

ID4:Data4

ID24:Data24

ID8:Data8

ID19:Data19

ID2:Data2

ID7:Data7

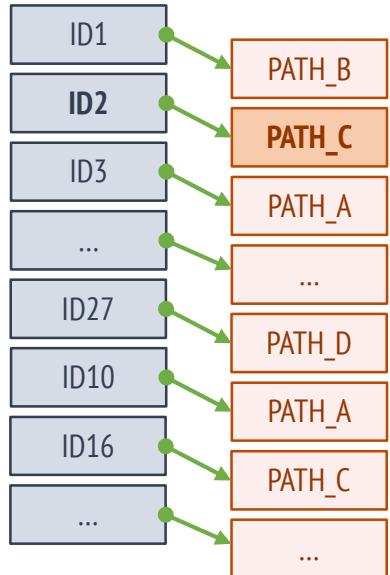
ID22:Data22

ID14:Data14

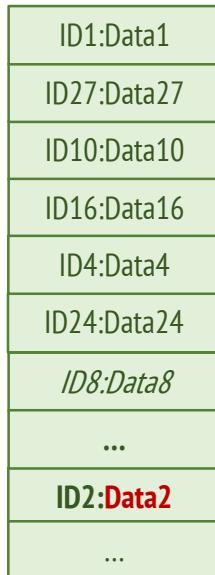
ID13:Data13

ID20:Data20

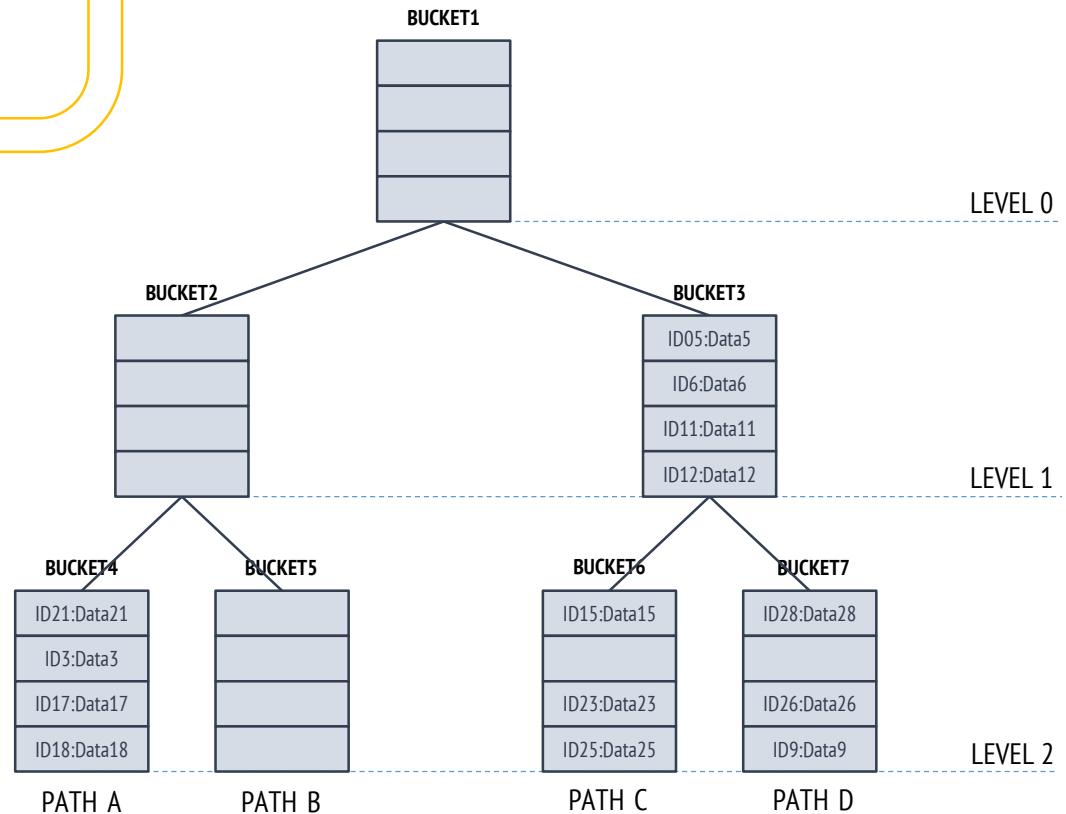
MAP



STASH



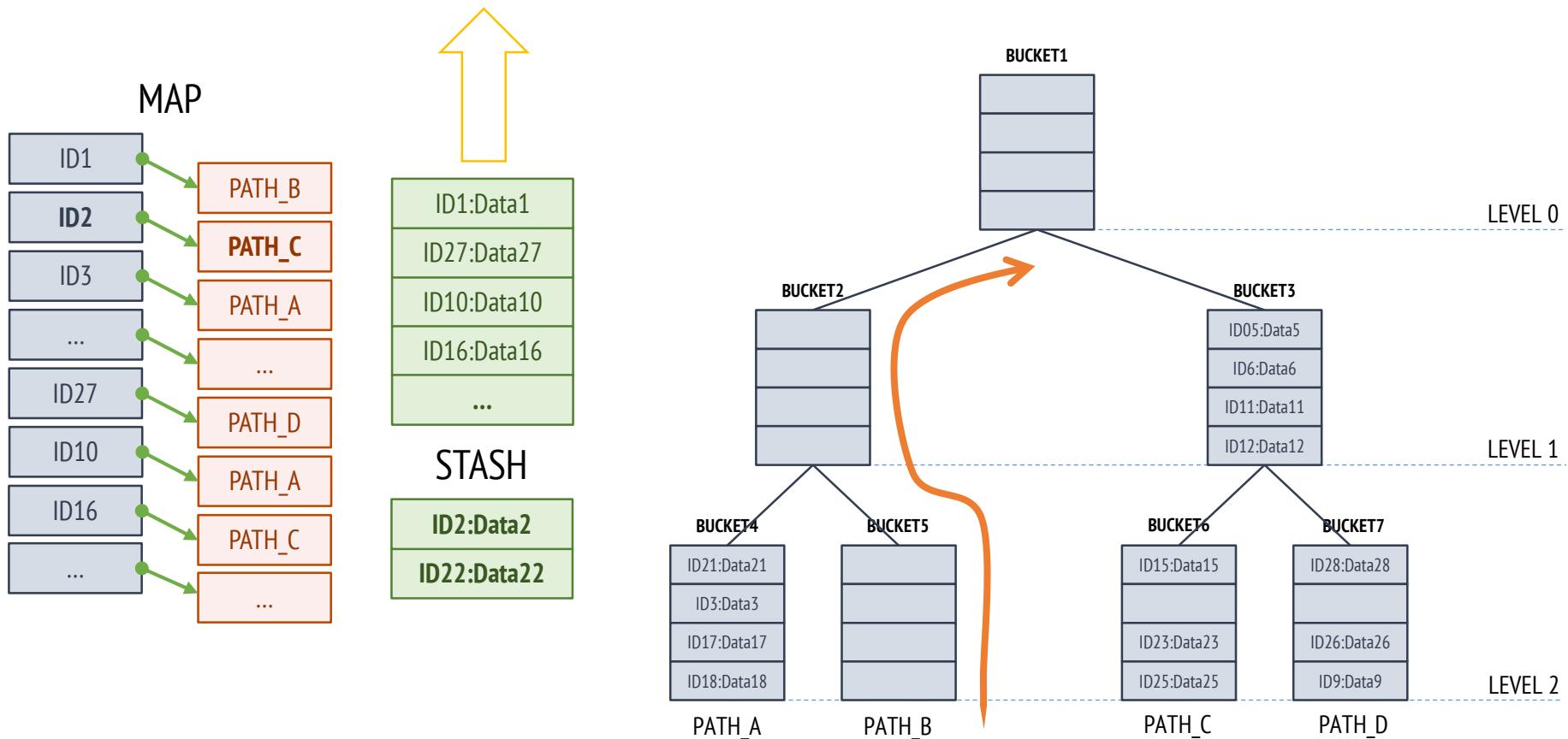
BUCKET1



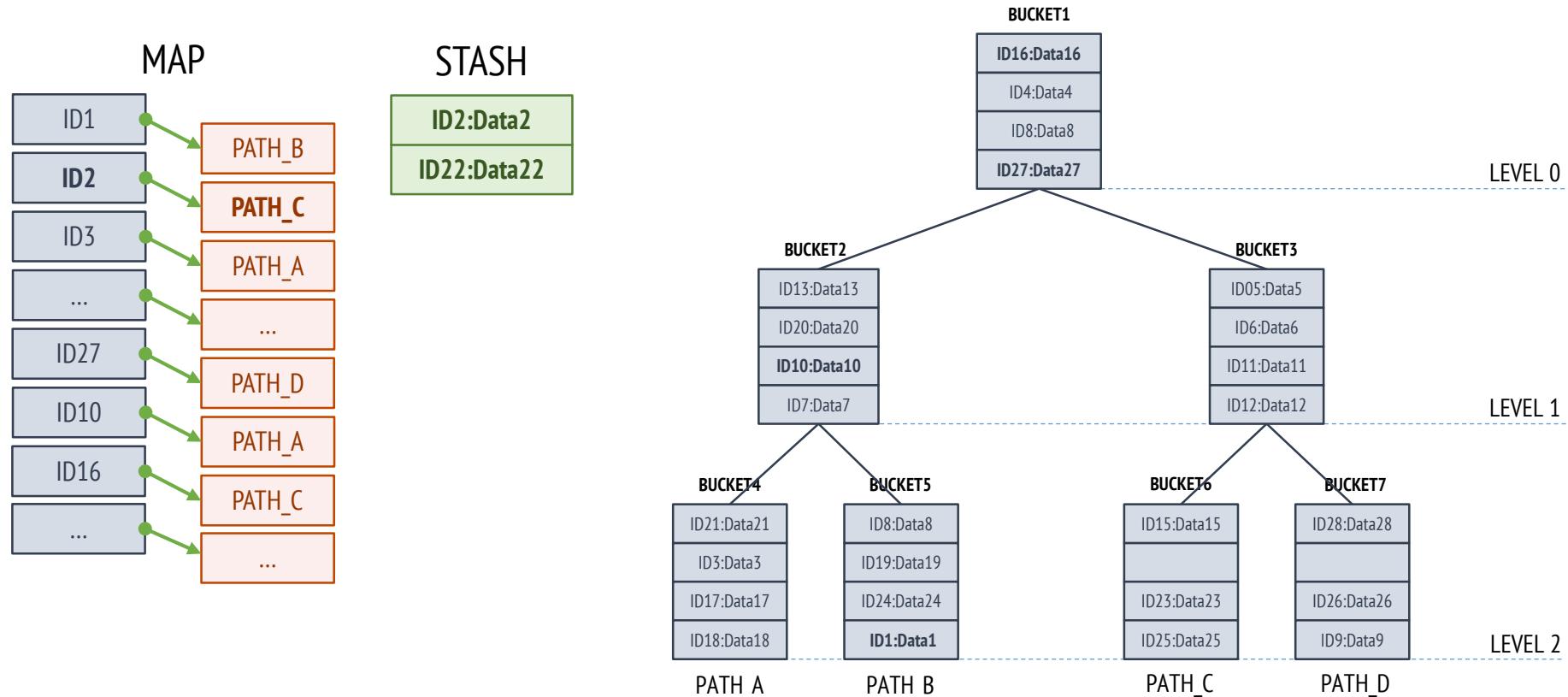
PathORAM access

WRITE:

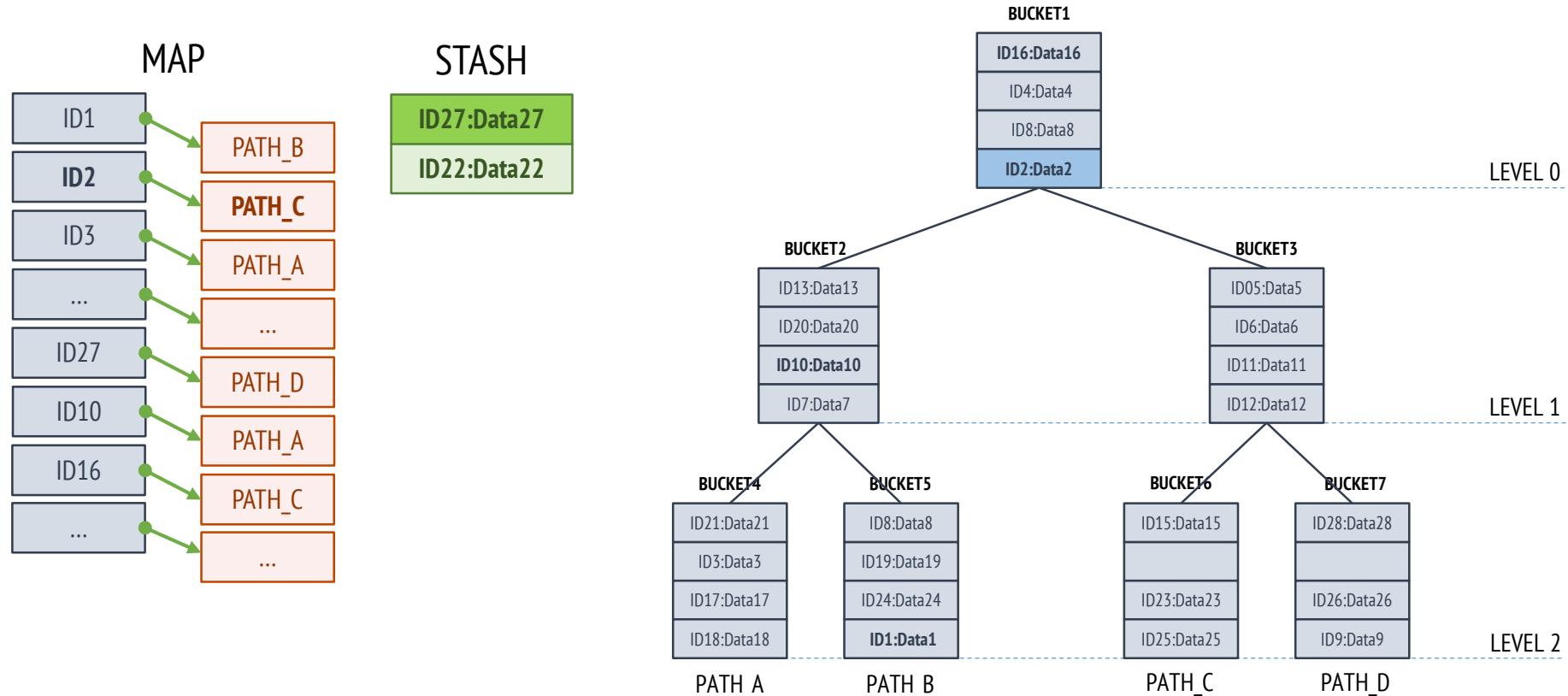
PATH_B												
Bucket5				Bucket2				Bucket1				
ID1:Data1	ID24:Data24	ID19:Data19	ID8:Data8	ID7:Data7	ID10:Data10	ID20:Data20	ID13:Data13	ID27:Data27	ID8:Data8	ID4:Data4	ID16:Data16	



PathORAM structure



PathORAM structure (alternative)

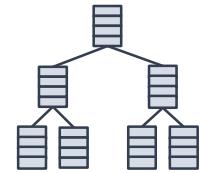
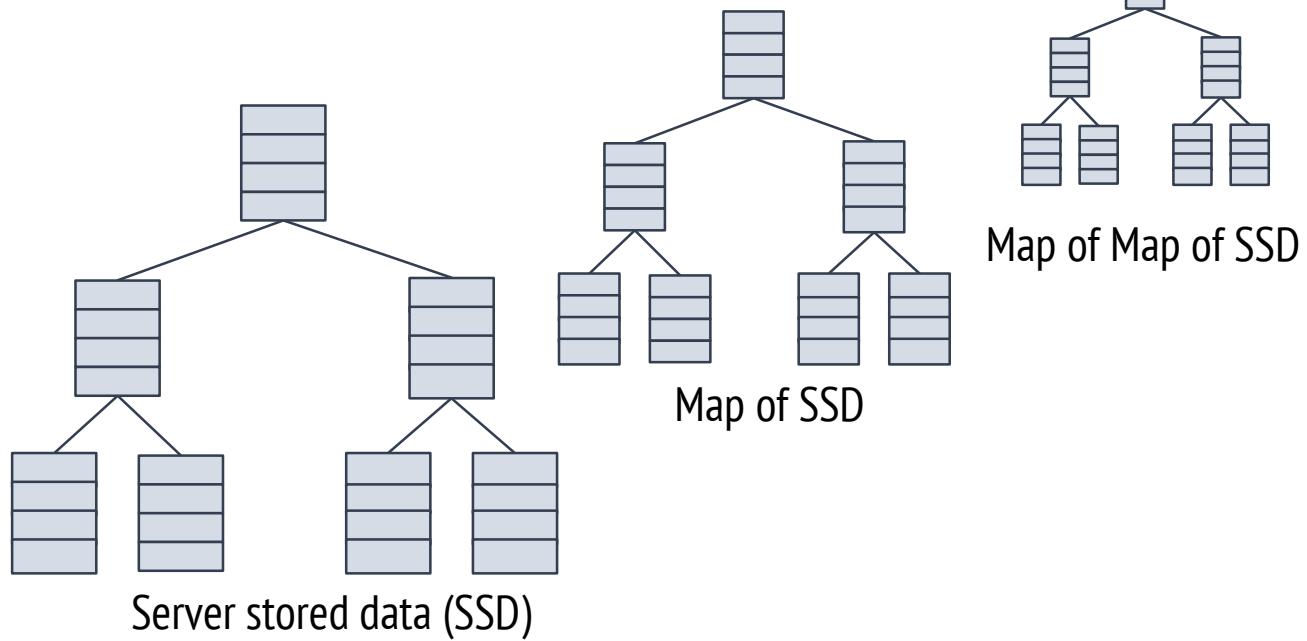


Algorithm

Access(op, a, data^{*}):

```
1:  $x \leftarrow \text{position}[a]$ 
2:  $\text{position}[a] \leftarrow \text{UniformRandom}(0 \dots 2^L - 1)$ 
3: for  $\ell \in \{0, 1, \dots, L\}$  do
4:      $S \leftarrow S \cup \text{ReadBucket}(\mathcal{P}(x, \ell))$ 
5: end for
6:  $\text{data} \leftarrow \text{Read block } a \text{ from } S$ 
7: if op = write then
8:      $S \leftarrow (S - \{(a, \text{data})\}) \cup \{(a, \text{data}^*)\}$ 
9: end if
10: for  $\ell \in \{L, L-1, \dots, 0\}$  do
11:      $S' \leftarrow \{(a', \text{data}') \in S : \mathcal{P}(x, \ell) = \mathcal{P}(\text{position}[a'], \ell)\}$ 
12:      $S' \leftarrow \text{Select min}(|S'|, Z) \text{ blocks from } S'.$ 
13:      $S \leftarrow S - S'$ 
14:      $\text{WriteBucket}(\mathcal{P}(x, \ell), S')$ 
15: end for
16: return data
```

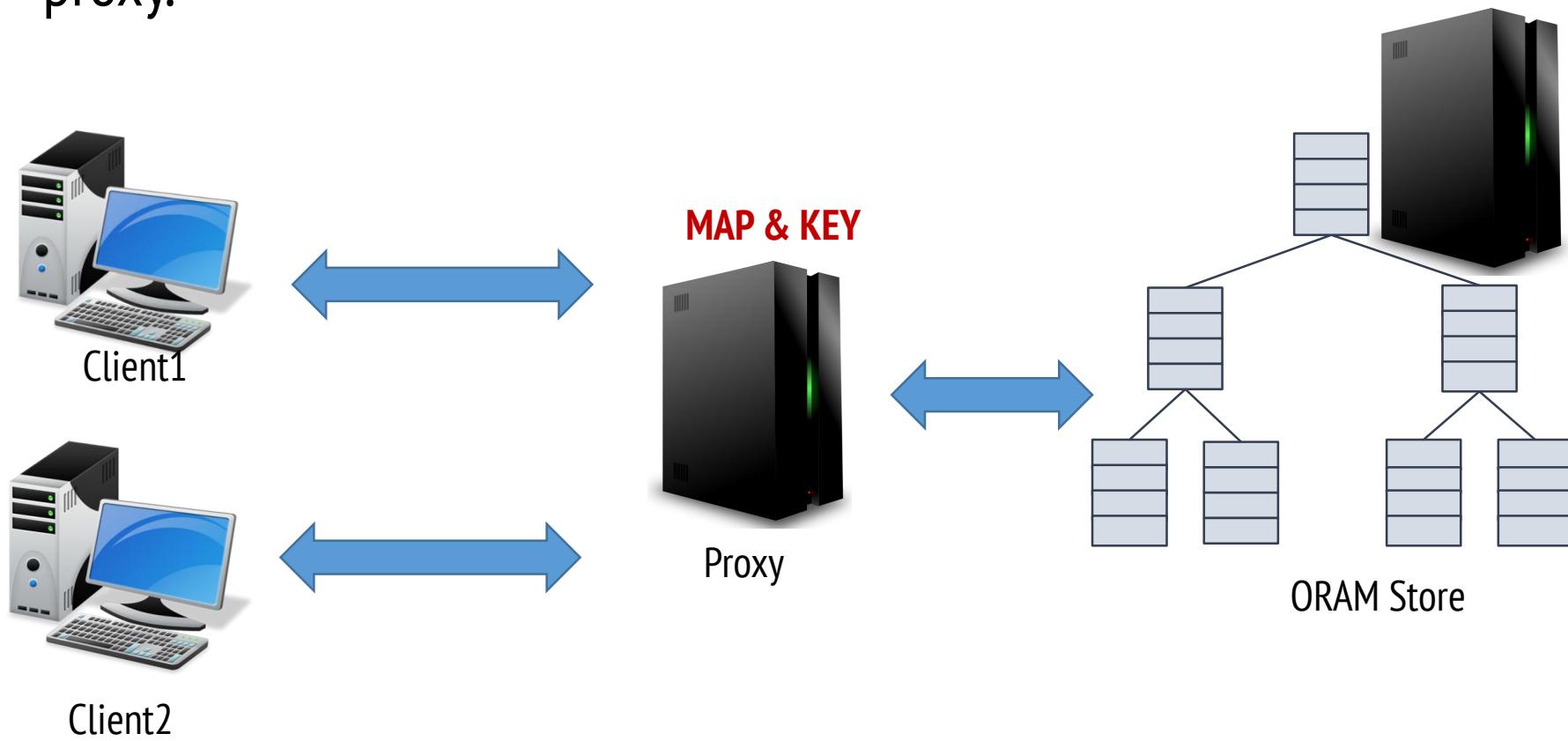
Recursive ORAM



Client stored
Map of ... of Map of SSD

Other Limitations

PathORAM is limited to a single user. If multiple users require access to the store (server), access must be done through a proxy.



Other Limitations

If multiple users access the store timing attacks can be leveraged by the server with respect to

1. Proxy data CACHING
2. Proxy duplicating requests (e.g. Client1 and Client2 request same data)
3. Volume of data (e.g. Client1 wants more data than Client2)

PathORAM performance

Example

Assuming a 128GB database with:

- $S = 64KB$ block size
- $Z = 5$ blocks per bucket
- $L = 20$ levels

`SecretDocument.txt`

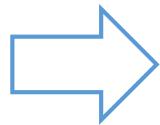
a 1MB document stored in the database

PathORAM performance

Example

Assuming a 128GB database with:

- $S = 64KB$ block size
- $Z = 5$ blocks per bucket
- $L = 20$ levels



What are the bandwidth requirements to access this document?

SecretDocument.txt

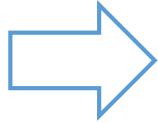
a 1MB document stored in the database

PathORAM performance

Example

Assuming a 128GB database with:

- $S = 64KB$ block size
- $Z = 5$ blocks per bucket
- $L = 20$ levels



$1MB = 1024KB$

Block per document N:

$N = 1024KB/64KB$ (size of the block) = 16

`SecretDocument.txt`

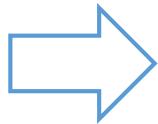
a 1MB document stored in the database

PathORAM performance

Example

Assuming a 128GB database with:

- $S = 64KB$ block size
- $Z = 5$ blocks per bucket
- $L = 20$ levels



$1MB = 1024KB$

Block per document N:

$N = 1024KB/64KB$ (size of the block) = 16

To send/receive ONE document

PathORAM requires: $N*S*Z*L = 100MB$

`SecretDocument.txt`

a 1MB document stored in the database

ORAM applications

- Personal health records
- Credit score systems
- GENOME related research
- As a private information retrieval (PIR) protocol

ORAM vs. Searchable Encryption

ORAM

- Provides anonymous access to data blocks
- Used as a private information retrieval (PIR) protocol
- Fully protects access patterns and data contents
- Requires a considerable overheads

Searchable encryption

- Enables users to securely search a precomputed index
- Used to efficiently **locate** data in databases
- Protects search terms and search results
- Only protects search patterns

Designing Secure Systems

guest lecture

Mark D. Ryan

15 November 2017



UNIVERSITY OF
BIRMINGHAM

Security
and
Privacy

Online services (aka cloud computing)

Communication (1995-)

- Email
- Gmail
- Whatsapp/
iMessage/
Telegram
- Facebook/
LinkedIn/
Snapchat/
Instagram

Online services (aka cloud computing)

Communication (1995-)

- Email
- Gmail
- Whatsapp/
iMessage/
Telegram
- Facebook/
LinkedIn/
Snapchat/
Instagram

Workflow (2005-)

- Canvas/
Easychair/
ServiceNow
- JustGiving/
SurveyMonkey
- Online docs
- Calendar
- Banking

Online services (aka cloud computing)

Communication (1995-)

- Email
- Gmail
- Whatsapp/
iMessage/
Telegram
- Facebook/
LinkedIn/
Snapchat/
Instagram

Workflow (2005-)

- Canvas/
Easychair/
ServiceNow
- JustGiving/
SurveyMonkey
- Online docs
- Calendar
- Banking

IoT (2015-)

- Thermostats
- Lights
- Fridge, kettle,
toilet, ...
- Speech-
understanding
speakers
- Cars, trains, ...

Online services (aka cloud computing)

Communication (1995-)

- Email
- Gmail
- Whatsapp/
iMessage/
Telegram
- Facebook/
LinkedIn/
Snapchat/
Instagram

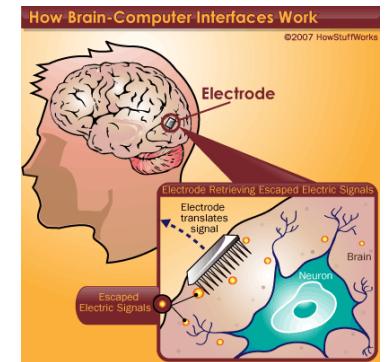
Workflow (2005-)

- Canvas/
Easychair/
ServiceNow
- JustGiving/
SurveyMonkey
- Online docs
- Calendar
- Banking

IoT (2015-)

- Thermostats
- Lights
- Fridge, kettle,
toilet, ...
- Speech-
understanding
speakers
- Cars, trains, ...

Brain (2025-)



Poll 1

- In 20 years time, when Apple (or Amazon or Google) announces a brain-computer interface for \$500, I will:
 - Never buy it. On the contrary: I will campaign to have it banned.
 - Resist it at first, but eventually succumb. I hate the idea, but I know that the incredible convenience will (as always) prove greater than my reservations.
 - Be an early adopter, as I was for smartphones, online-everything, speech-understanding speakers, IoT devices, driverless cars...

How should the data amassed by online services be used?

Acceptable uses (?)

- Providing the service
- Developing additional services
- Targeted advertising
- Necessary, proportionate, targeted surveillance

Unacceptable uses (?)

- Aggregation with data from other sources
- Mass surveillance
- Commercial pestering (spam, etc.)
- Blackmailing, coercion, ...
- **Algorithmic determination of eligibility for services**

Video

- <https://www.youtube.com/watch?v=yzrmdXWEh20>

How can we allow the acceptable uses while preventing the unacceptable ones?

Laws and regulations

- Flexible
- Hard to enforce

Clever technologies

- Inflexible
- Ideally, they are self-enforcing

An old technology: PGP encryption

PGP encryption

- Provides end-to-end encryption in mail systems
- Provides “web of trust” model to allow senders to authenticate the recipient’s public key
 - You get your friends to certify your key
 - Your interlocutor can then judge whether your key is “sufficiently” certified.

Prevents

- Cloud-side analysis of messages
 - Including cloud-side search of messages

Allows (in fact, exacerbates)

- Cloud-side analysis of metadata (aka communications data)

Some recent technologies: Signal protocol

Signal protocol

- Provides end-to-end encryption in messaging systems
 - Used by Whatsapp and Signal
 - 1Bn users
 - (Compare with PGP: hardly any users.)

Prevents

- Cloud-side analysis of messages
 - Including cloud-side search of messages

Allows

- Cloud-side analysis of metadata (aka communications data)

Some recent technologies: End-to-end encrypted “Google Docs”

“Private editing using untrusted cloud services” (Yan Huang, David Evans, Virginia University, 2011.)

- Participants share a password using some trusted means (e.g., email, whatsapp, ...)
- Documents are stored encrypted with key derived from the password.
- Participants can simultaneously edit the same document...
- Available as browser extension and server code.

Prevents

- Cloud-side analysis of documents
 - Including cloud-side search of documents

Allows

- Cloud-side analysis of metadata (aka communications data)

Some recent technologies: End-to-end encrypted Easychair (or Canvas or ServiceNow or ...)

“Privacy Supporting Cloud Computing: ConfiChair, a Case Study” (M. Arapinis, S. Bursuc, M. Ryan, University of Birmingham, 2012.)

- Participants share keys using trusted means
- Documents are stored encrypted with those keys.
- In-browser key translation....
- Supports more complex work flows.

Prevents

- Cloud-side analysis of documents
 - Including cloud-side search of documents
- Cloud-side analysis of some metadata (e.g., who reviews who's paper).

Usability hit?

- PGP: **heavy usability hit**. “Why Jonny can’t encrypt” became a famous paper.
- Signal protocol: **almost no hit!**
 - Possibly some issues arise if you change phone after a message has been sent.
- Huang/Evans private editing: **moderate hit** (copy/paste passwords).
- Confichair: **moderate hit** (copy/paste keys).

Poll 2

- In 20 years time, when everything is connected to the internet, and everything we say or do is recorded in the cloud, how ***will*** we routinely address this problem?
 - By ignoring it.
 - By having legislation aimed at curbing abuses of data.
 - By having in place clever technologies that precisely control how the data can be used.

What we want

Technology which

- Allows cloud to perform certain specified *computations* with user data, but prevents other computations being done.
 - “Reverse DRM”
- Provides a rich policy language to specify what kinds of computations are allowed.
- Provides secure evidence to clients about the policy: “attestation”.

Approaches to solving the “confidentiality from the cloud provider” problem

Crypto

- Fully homomorphic encryption
- Functional encryption
- Order-preserving encryption
- Multi-party computation
- White-box crypto
- Indistinguishability obfuscation

Challenges

- Restrictions on the program P
- Use-case restrictions
- Performance

Hardware

- TPM & Intel TXT
- ARM Trustzone
- **Intel SGX**

Challenges

- Requirement to trust HW design and implementation
- Size of TCB
- Business model
- Documentation

Binding keys to programs using Intel SGX remote attestation

Mark D. Ryan

Designing secure systems lecture

15 November 2017



UNIVERSITY OF
BIRMINGHAM

Security
and
Privacy

Intel SGX

Intel SGX is a set of processor instructions which allow one:

- To set up an enclave (code & memory) such that the code runs in a way that it and its memory are protected from interference from the OS and other software
- To securely report the state of the enclave, locally and remotely

Present on all (major) Intel processors from Skylake (2015) onwards

Not the first *hardware security anchor*

Trusted platform module (TPM)

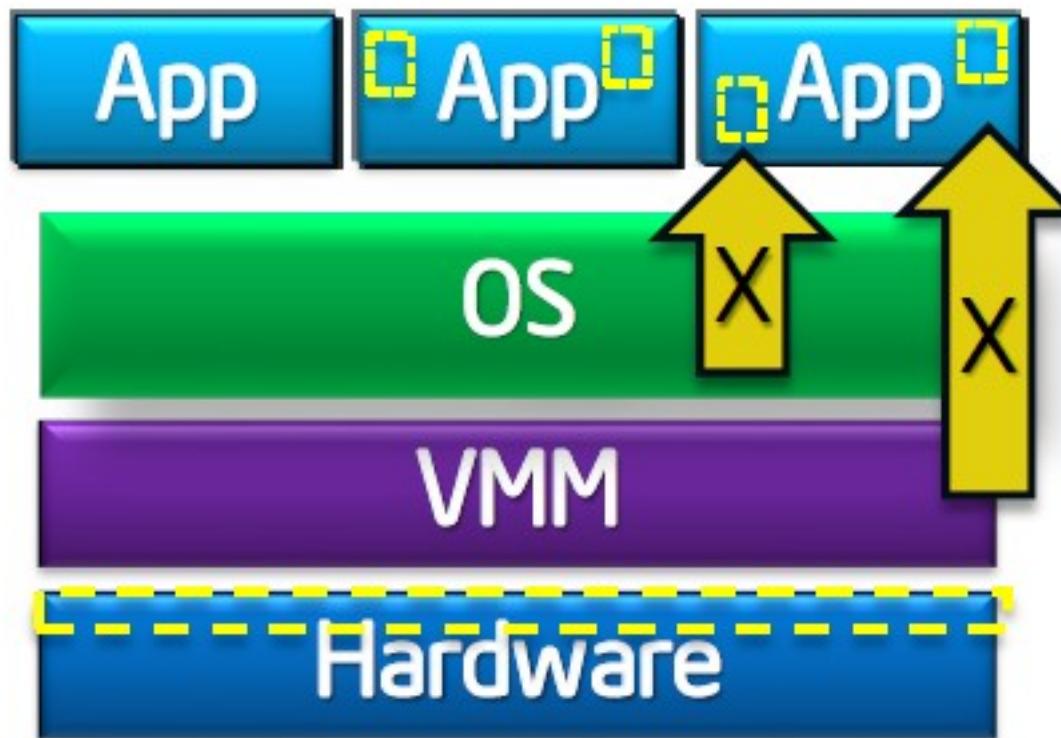
- Version 1 (2004), 1.2 (2008), 2.0 (2014-)
- Separate chip soldered to motherboard
- API that allows you to create keys whose secret part never leaves the TPM
 - A key can be locked to “authdata” (like a password to use the key)
 - And/or can be locked to PCR values, which “measure” the boot sequence

Best known use: Microsoft Bitlocker

ARM TrustZone

- ARM processors have two execution modes, with hardware-enforced access control between them:
 - “Normal world” Runs the rich OS (e.g., Android) and apps
 - “Secure world” Runs security-critical code.

Intel SGX: attacks addressed



An enclave within an app is protected from interference from other software, including the OS and VMM. Note that enclaves can only run in ring 3 (user space).

Intel SGX: attacks not addressed

- Side-channel attacks
 - Cache and page access patterns
 - Extraction of RSA secret keys, under assumptions, by co-located [enclave] processes
 - Programmer is expected to mitigate this attack
- Hardware attacks
 - Chip decapsulation
 - Trojan hardware: vulnerabilities possibly introduced in the supply chain

Intel SGX

Not suited for:

- Applications that involve I/O on the platform
 - Password managers
 - Banking apps

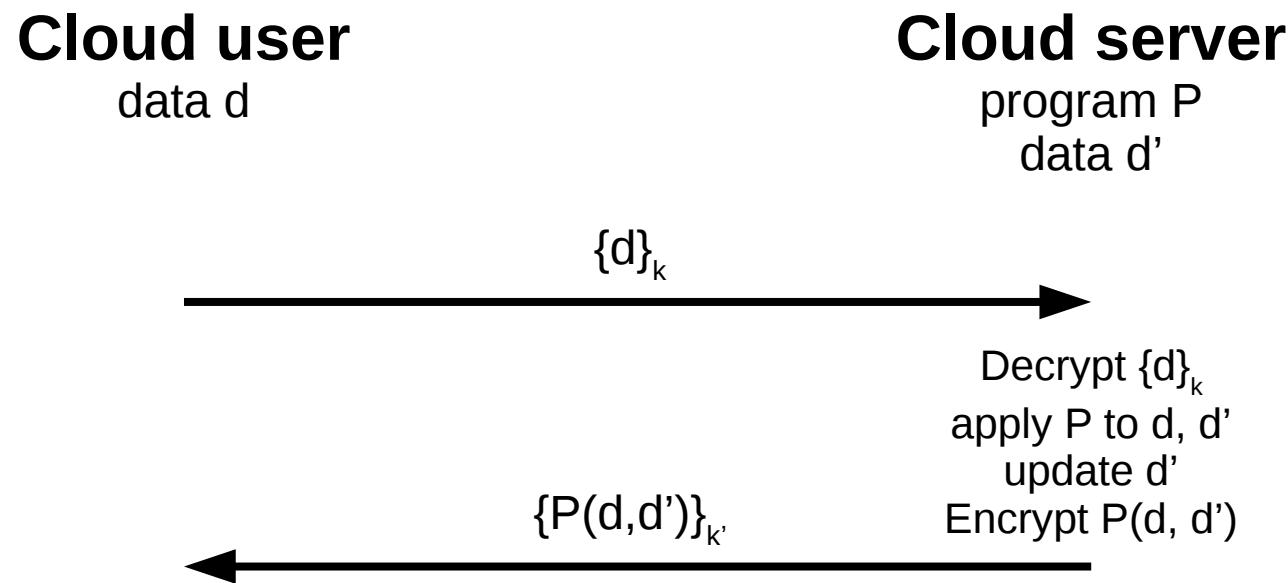
Well suited for:

- Cloud computing (“reverse DRM”), in which your device sends data to a cloud server, and you want to impose restrictions on how it is processed

Partly suited for:

- DRM, where a server delivers content to your device, along with restrictions on how you use it

Example: confidentiality from the cloud provider



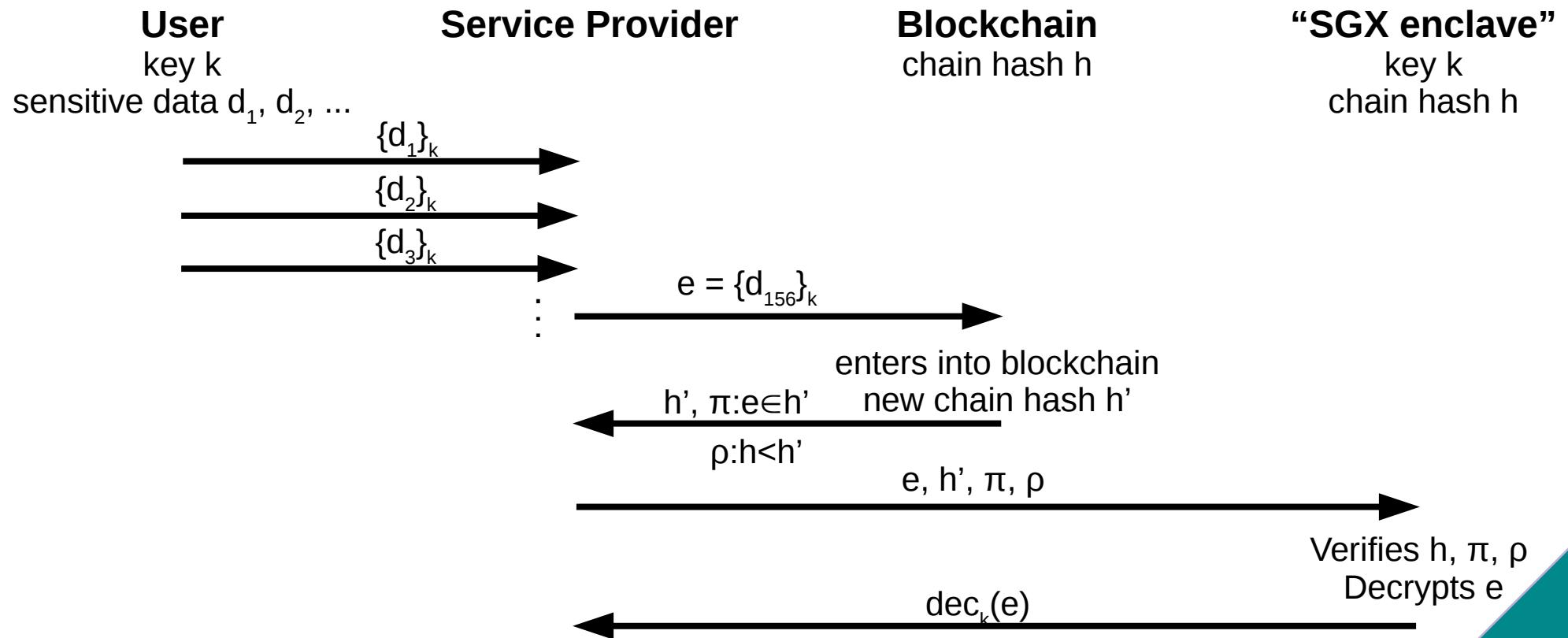
Bob cannot access d except by applying P to it and returning that to Alice.

In general, Bob does not know d, d' or k, k'
Bob does know P

Example: “accountable decryption”

Escrow with accountability: whenever it decrypts, SP creates evidence which cannot be suppressed or discarded.

Use case: user uploads her encrypted location continually; SP decrypts it only when she reports lost phone.



Intel SGX concepts

Protected memory

- Enclave Page Cache (EPC), access control, MEE

Enclave

- “*SGX enclave control structure*” (SECS)
 - Core data about the enclave, held in a dedicated EPC page.
- Life cycle of an enclave
 - Creation / loading / initialisation (aka launching) / teardown

Intel SGX concepts

Enclave measurement

- An enclave measurement (noted MRENCLAVE) is a hash of its code and initial data

Enclave identity

- MRENCLAVE: Its measurement is the strictest way to identify an enclave.
- MRSIGNER: An “enclave certificate” is a more flexible way to identify an enclave. The certificate is signed by the “independent software vendor” (ISV), and includes ISVPRODID and ISVSVN.
 - Allows data migration from old security versions to new ones.

Intel SGX concepts

As well as *processor instructions*...

- ECREATE, EADD, EEXTEND, EINIT, ... : managing the enclave life cycle
- EGETKEY, EREPORT, ... : managing data within an enclave.

... there are *Intel-provided enclaves*

- Launch enclave
- Provisioning enclave
- Quoting enclave

Intel SGX secret values

Some secret values are built into the platform.

Known to the processor and to Intel:

- *SGX Master derivation key*
 - Derived from *provisioning secret*

Known to the processor (but not to Intel)

- *Seal secret* (also known as `SEAL_FUSES`)
- `OWNER_EPOCH`

Setting up an enclave

- System software uses ECREATE to set up the initial memory page allocated to the enclave, which contains the SGX Enclave Control Structure (SECS)
- It uses EADD to allocate further pages containing enclave code and initial data
- It uses EEXTEND to update the enclave's 'measurement'
- After loading the initial code and data pages into the enclave, the system uses a 'Launch Enclave' (LE) to obtain an EINIT token
 - The token is provided to the EINIT instruction to initialise the enclave
 - LE is a privileged enclave provided (e.g.) by Intel, signed by Intel private key

Initialising an enclave (more detail)

Untrusted system software sets up SECS and the enclave certificate SIGSTRUCT

SECS

MRENCLAVE
MRSIGNER
ATTRIBUTES
- DEBUG
- XFRM
ISVPRODID
ISVSVN

SIGSTRUCT

ENCLAVEHOST
VENDOR
ATTRIBUTES
ATTRIBUTEMASK
ISVPRODID
ISVSVN
signature

EINITTOKEN

MRENCLAVE
MRSIGNER
ATTRIBUTES
launch enclave info
MAC

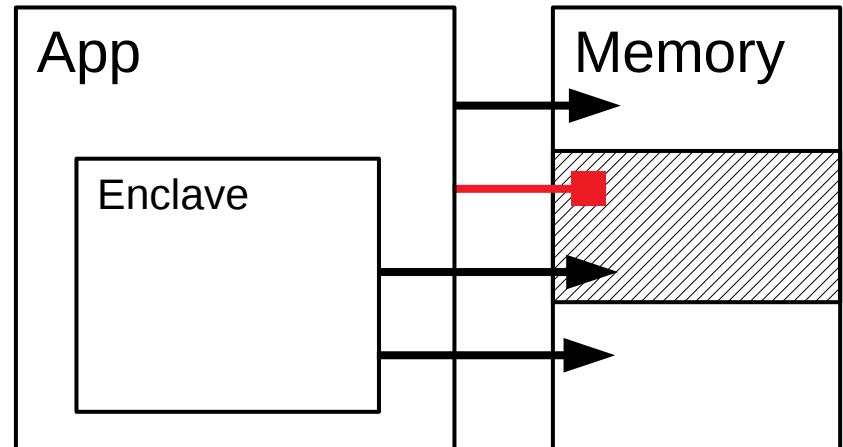
A launch enclave

- checks the enclave certificate SIGSTRUCT against SECS
- checks the “launch policy”
- produces EINITTOKEN
- Produces the EINITTOKEN MAC using a launch key obtained using EGETKEY

The processor instruction EINIT checks EINITTOKEN and initialises the enclave

What an enclave can do

- Computations
- Access its own [encrypted] memory
- Access app memory
- Communicate with user, but insecurely
- Communicate with another party, which can be secure if the enclave shares a key with the other party
- Attest its identity (a hash of its binary and initial data) to another party
- “Seal” data, i.e. encrypt data with a key that only it can access, for persistent storage
 - Can use Platform Service Enclave (PSE) for *trusted time* and *monotonic counter*
- Teardown

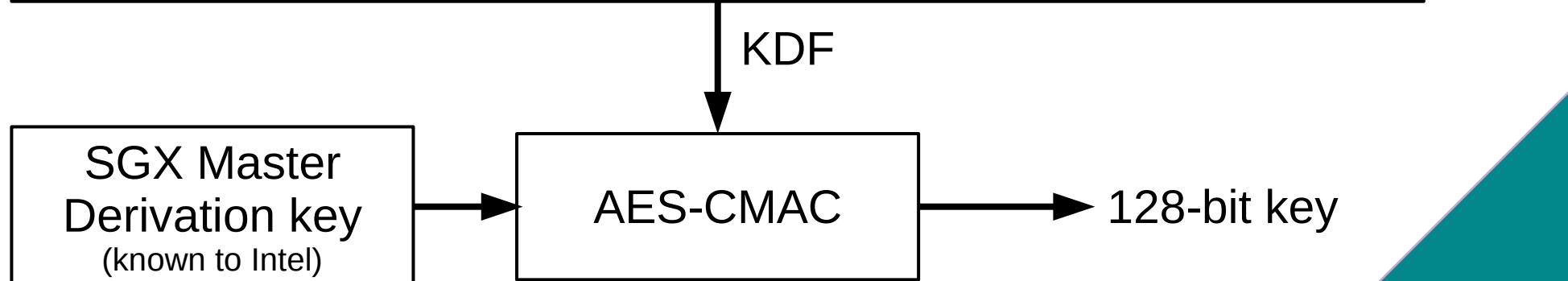


Seal keys obtained using EGETKEY

Key request

KEYNAME	e.g. seal key, report key, provisioning key
KEYID	
KEYPOLICY	MRENCLAVE and/or MRSIGNER
ATTRIBUTEMASK	
ISVSVN	must be \leq the caller's ISVSVN
CPUSVN	must be \leq the calling platform's CPUSVN

MRENCLAVE dep. on KEYPOLICY	MRSIGNER dep. on KEYPOLICY	MASKEDATTRIBUTES	
ISVPRODID	KEYNAME	ISVSVN	CPUSVN
OWNEREPOCH set by platform owner	SEALFUSES not known to Intel		KEYID



Migrating data between enclaves

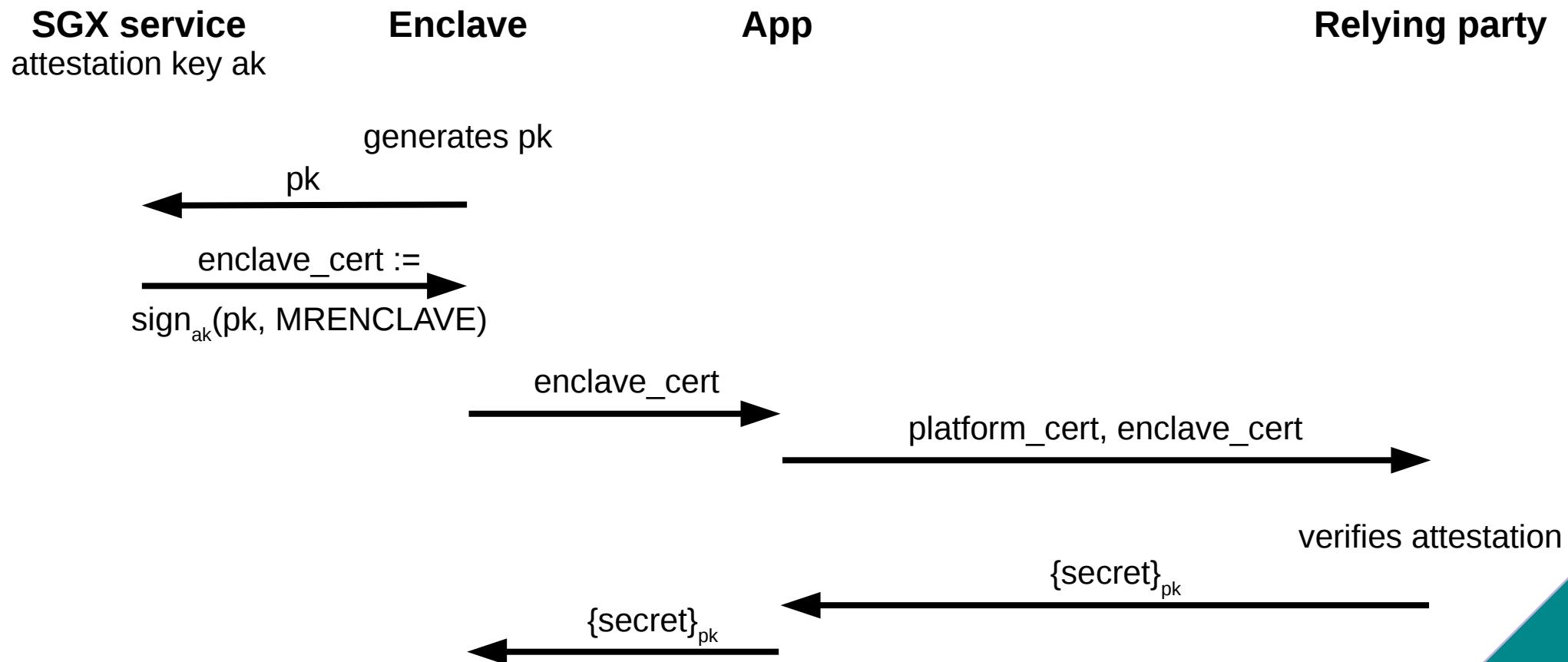
- **Same platform, same enclave (just a different instance):**
 - Sealed blob can migrate.
- **Same platform, different enclave:**
 - If it's a newer security version of the same ISVPRODID, and the KEYPOLICY is set to MRSIGNER, then the sealed blob can be migrated.
 - More generally, the EREPORT mechanism can be used to set up a secure channel between two arbitrary enclaves on the same platform
- **Different platform, same or different enclave:**
 - Need remote attestation.

Remote attestation

- **How can a remote party know that it is talking to a given enclave?**
 - An enclave is identified by MRENCLAVE [strict] or by MRSIGNER/ISVPRODID [more flexible]
- **How can a remote party know that a given key can be used exclusively by a given enclave?**

Simple remote attestation

Platform with SGX has an “attestation” signing key ak, and Intel has certified it : $\text{platform_cert} := \text{sign}_{\text{Intel}}(\text{pub}(\text{ak}))$

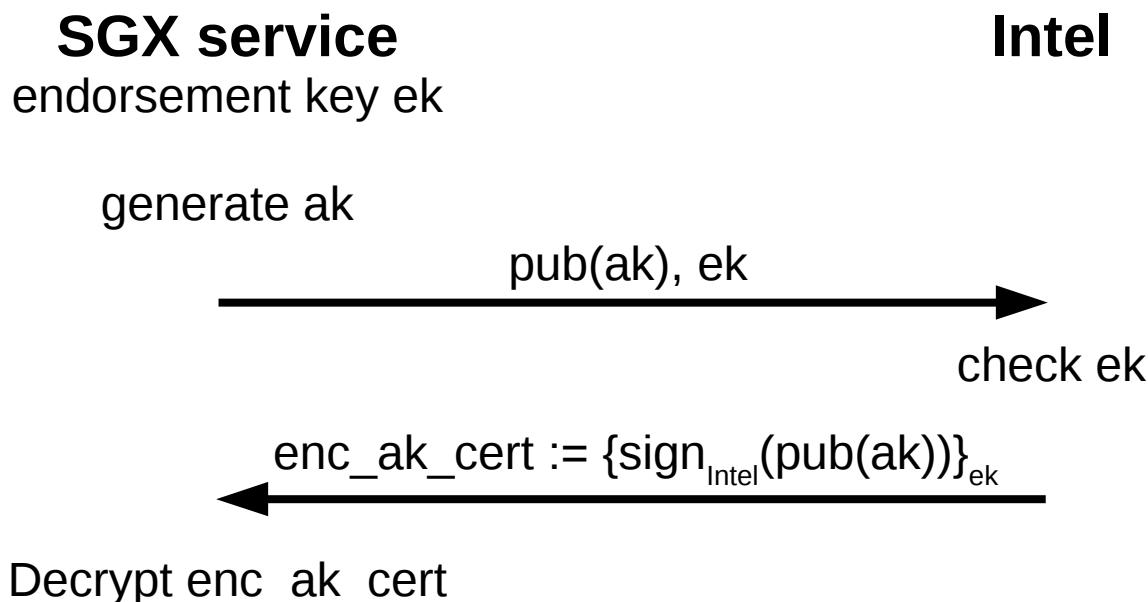


Objection 1: privacy concern

Privacy concern: not acceptable because RP can identify (using platform cert) *which* platform it is interacting with

This concern is not applicable if the attestation is that of a cloud service: cloud services do not require privacy

Solution 1: “Privacy CA” for provisioning ak



Solution 2: “Direct anonymous attestation” (DAA)

Objection 2: revocation concern

Intel would like to be able to revoke platform attestation keys if:

- Revocation based on private key:
the private part is seen in the wild (e.g. published on the Internet), or
- Revocation based on signature:
the key is perceived as signing erratically

Possible solutions

- Certificate revocation-list checking, or
- Short-lived certificates, that must be renewed periodically
(e.g., every month)

EPID Signatures and Verification

Issuer: gpk, isk

Join: P_i obtains sk_i by interacting with issuer

Sign: $\sigma = \text{sign}_{sk_i}^{gpk, \text{sigRL}}(m)$; or (*if sk_i is revoked*) $\sigma = \perp$

Verify: $\text{Verify}(gpk, m, \text{PrivRL}, \text{SigRL}, \sigma) = \text{valid or invalid}$

Revoke:

- $\text{RevokePriv}(gpk, sk_i)$
 - checks sk_i , and
 - adds sk_i to PrivRL
- $\text{RevokeSig}(gpk, \text{PrivRL}, m, \sigma)$
 - verifies σ , and
 - adds σ to SigRL

Remote attestation

Provisioning the attestation key

- A ‘provisioning enclave’ uses EGETKEY to obtain a symmetric ‘provisioning key’ which Intel can also compute
- It runs the EPID join protocol with Intel (protected by the provisioning key), obtaining its attestation signing key
- It uses EGETKEY to obtain a ‘provisioning seal key’ and stores the attestation key encrypted by the provisioning seal key

Remote attestation

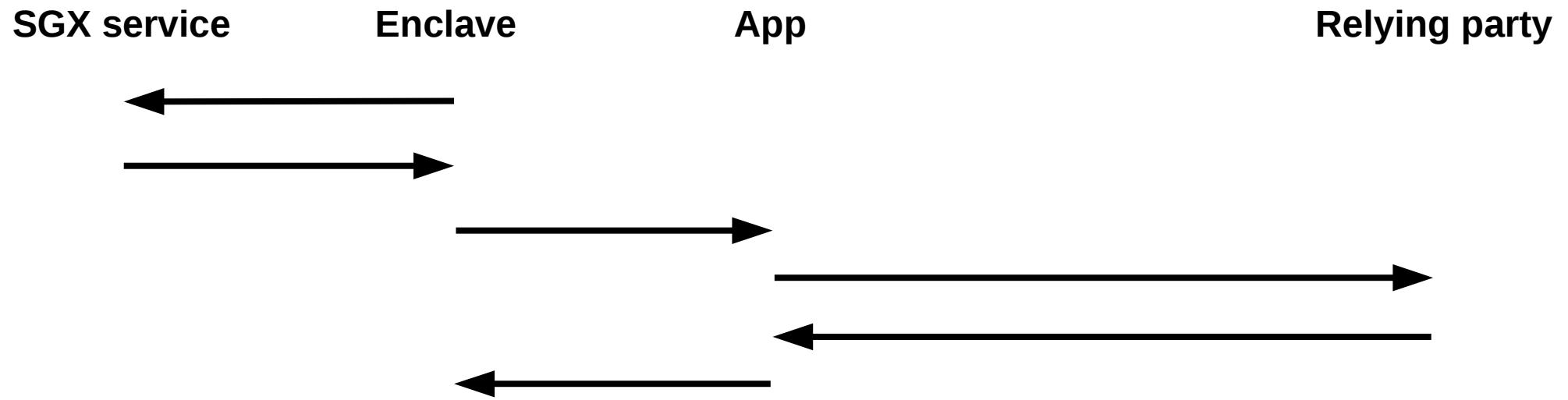
Producing a REPORT

- The attesting enclave uses EREPORT to produce a report structure, MAC'd with a report key
- The report is passed to a quoting enclave

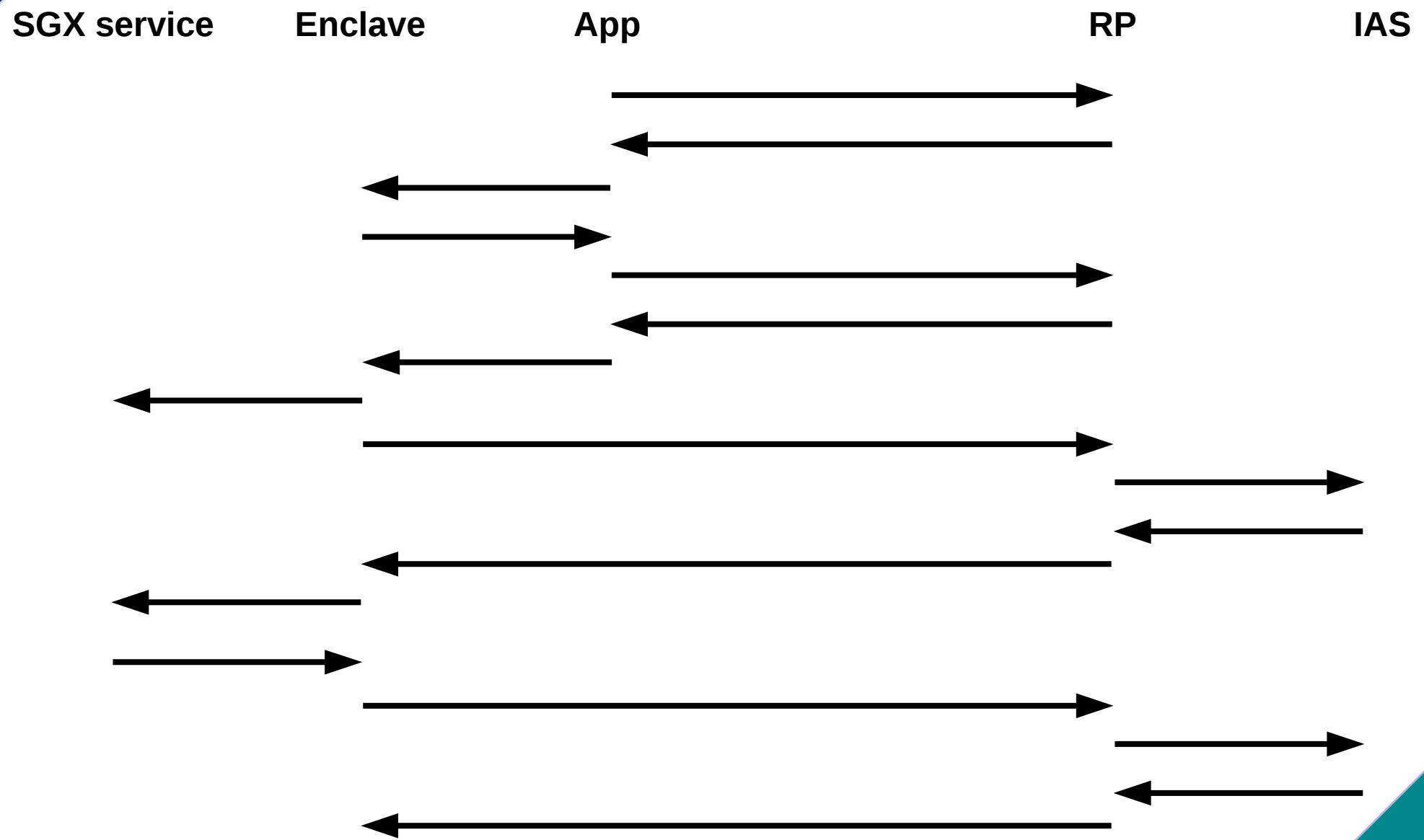
“Quoting” the report

- The quoting enclave uses EGETKEY to obtain a report key to check the report MAC
- It uses EGETKEY to obtain a provisioning seal key to decrypt the attestation key
- It uses the attestation key to sign the report (along with a received challenge)

Simple remote attestation



Intel's remote attestation



SGX uses in research literature

S. M. Kim, J. Han, J. Ha, T. Kim, D. Han. *Enhancing Security and Privacy of Tor's Ecosystem by Using rusted Execution Environments*. USENIX NDSI, 2017.

F. Schuster, M. Costa, D. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, M. Russinovich. *VC3: Trustworthy Data Analytics in the Cloud Using SGX*. IEEE S&P, 2015.

M. D. Ryan. *Making Decryption Accountable*. 25th Security Protocols Workshop, Springer LNCS, 2017.

K. Severinson, M. D. Ryan, C. Johansen. *Accountable Decryption Using Intel SGX*. In preparation.

- Very small, short-lived enclave (no page caching)

Conclusions

SGX: a powerful architecture for managing secret data

- + Enables processing of data that cannot be read by anyone, except for code running in the enclave
- + Minimal TCB: nothing trusted except for x86 processor
- + Not suitable for applications involving user I/O, but well suited for cloud-based applications
- Hardware and side-channel attacks
- Requires interaction with Intel at three distinct points:
 - Launch approval (by platform)
 - Join protocol to obtain attestation key (by platform)
 - Verify protocol to verify attestation (by relying party)
- Among other objections, this is privacy-invasive

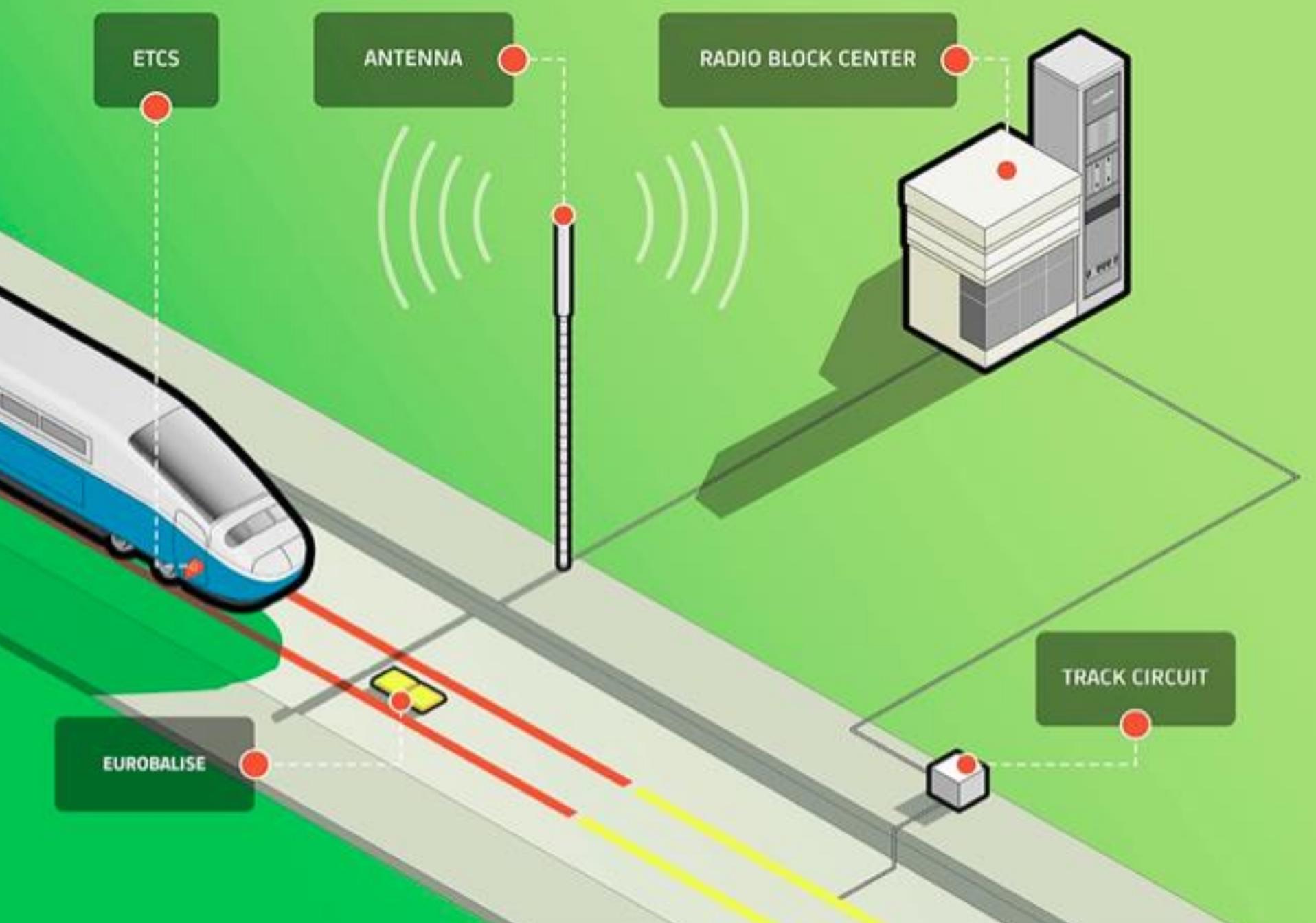
Protocol security

Mihai Ordean
Designing Secure Systems
University of Birmingham

Overview

- Device security
 - Is code on the device vulnerable to exploits ? (e.g. buffer overflows)
 - Is the code authenticated ? (i.e. has not been tampered with)
- Data security (in the cloud)
 - Is the stored data is accessible to everyone? (e.g. encrypted)
 - Is the stored data authenticated?
- Metadata security
 - What does metadata reveal about data?
 - Can we tamper the metadata?
- **Protocol security**
 - Is data in transit visible?
 - Can data in transit be tampered with?

European Rail Traffic Management System (ERTMS)



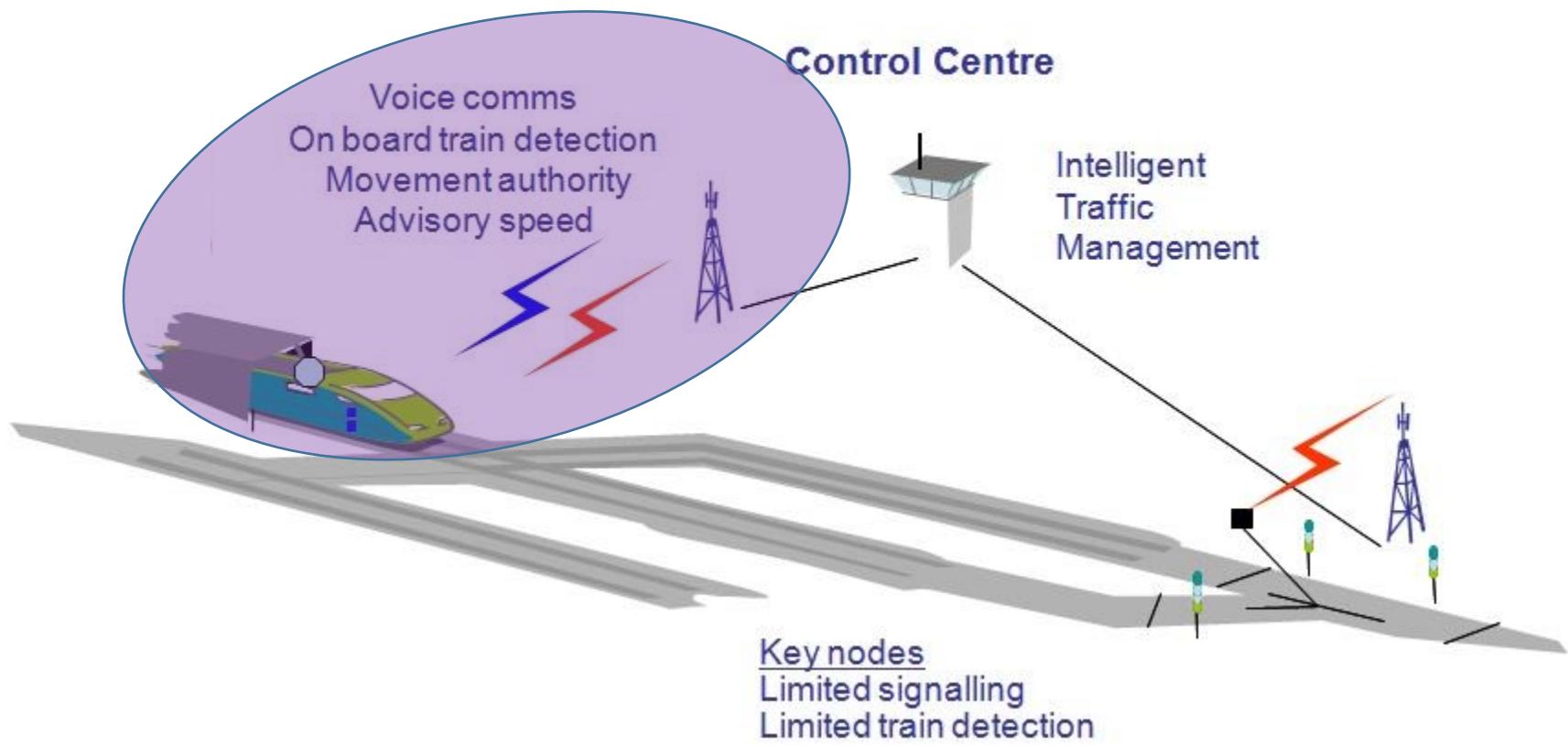
ERTMS Overview

The European Rail Traffic Management System (ERTMS) is a suite of protocols used to deliver next-generation train management and signalling.

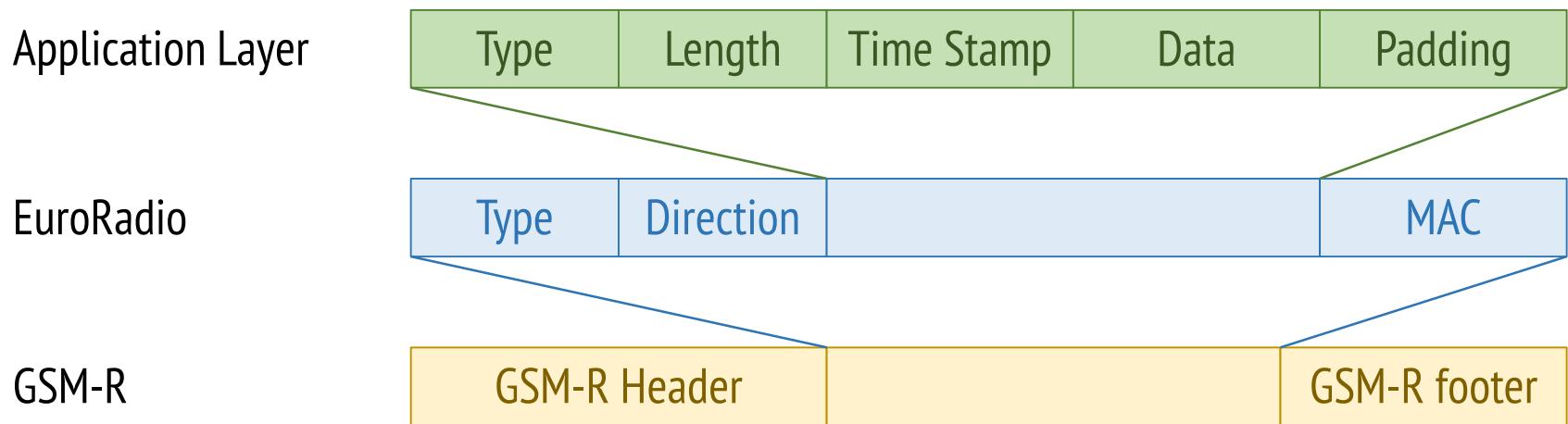
ERTMS components:

- GSM-R – encryption/physical interaction
- EuroRadio – message authentication
- Application Layer protocol - instructions

ERTMS Overview



ERTMS stack



ERTMS stack

Application Layer



EuroRadio



GSM-R



GSM-R

- Provides data encryption on the ERTMS stack
- Based on the GSM Mobile Communications Standard
(i.e. basically 2G) with:
 - different frequency ranges
 - rail-specific functionality (multi-party communication, emergency calling functionality, priority-based pre-emption, etc.)
- Crypto:
 - A5/1* a stream cipher based on (LFSRs)
 - A5/3 (optionally) a block cipher

* broken:

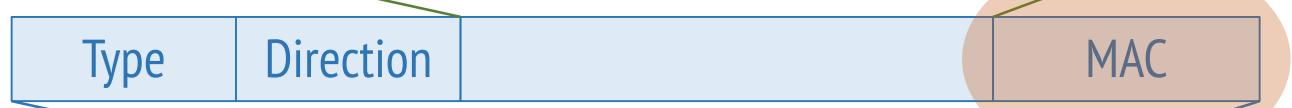
1. Elad Barkan, Eli Biham, Nathan Keller. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. *J. Cryptology* 21(3): 392-429 (2008)
2. L. Karstensen. GSM A5/1 rainbow tables in Oslo, Norway. Available: <https://lassekarstensen.wordpress.com/2013/08/08/gsm-a51-rainbow-tables-in-oslo-norway/>, 2015.
3. <https://www.ckn.io/blog/2016/01/25/gsm-sniffing-voice-traffic/>
4. https://www.youtube.com/playlist?list=PLRovDyowOn5F_TFotx0n8A79ToZYD2lOv

ERTMS stack

Application Layer



EuroRadio



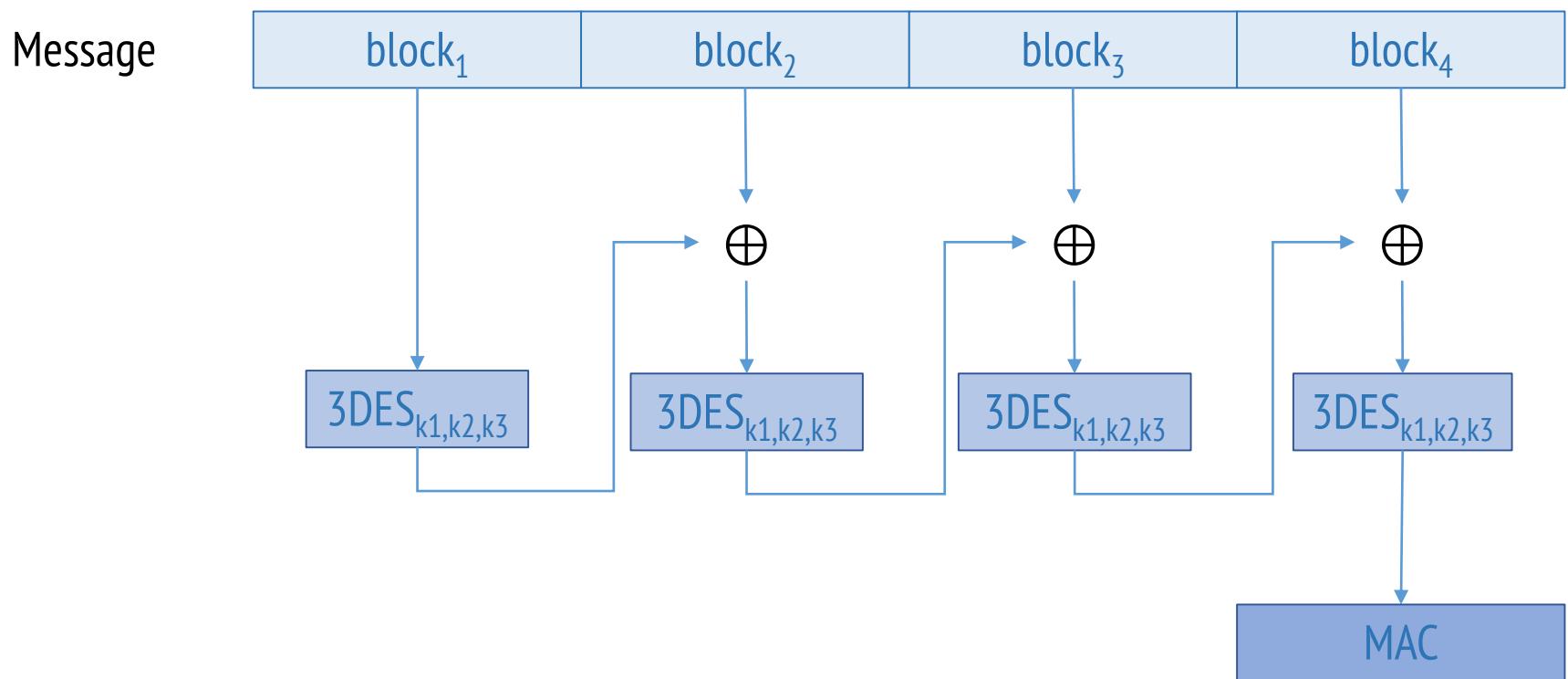
GSM-R



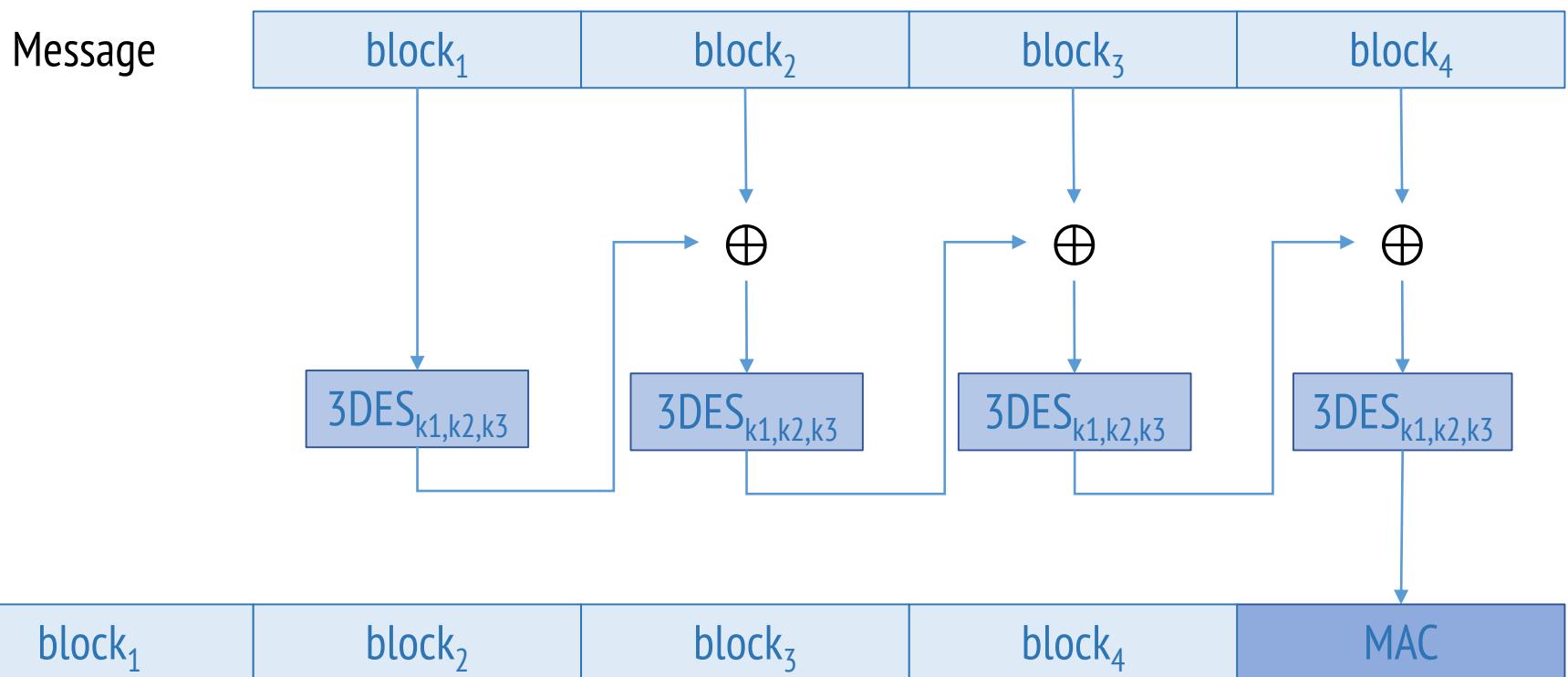
EuroRadio

- Provides authentication for messages on the upper layers
- Based on the ISO 9797-1 MAC Algorithm 3:
 - A CBC circuit which uses a combination of DES and 3DES
 - ISO 9797 padding, i.e. 0s are used as padding until data becomes a multiple of the block size
- Supports priority:
 - Normal priority: messages have a MAC
 - High priority: messages do not require a MAC (e.g. emergency stop messages)

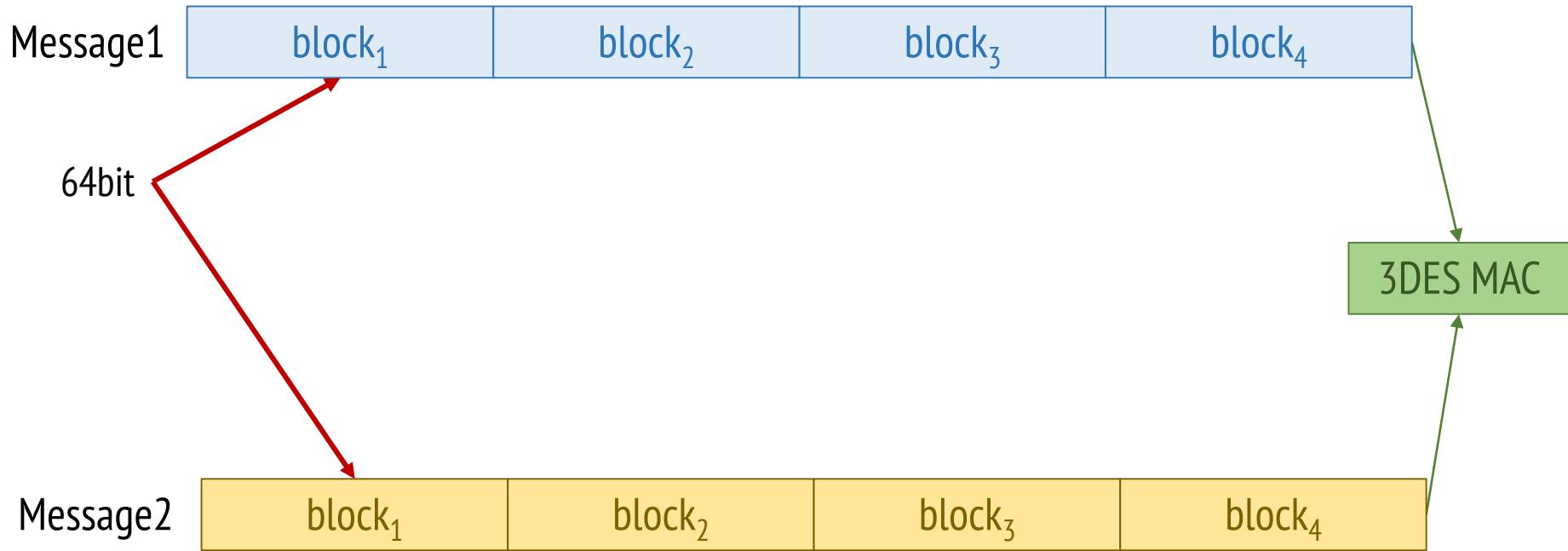
3DES-CBC-MAC



3DES-CBC-MAC

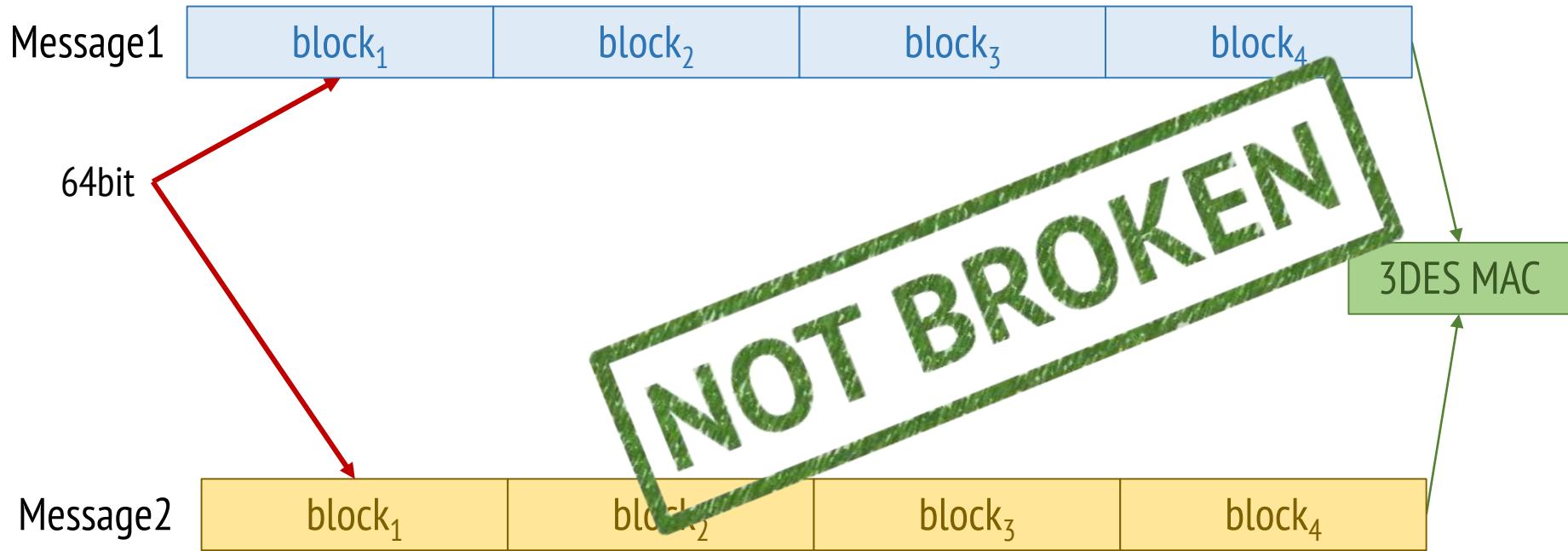


Collisions in ciphers with small block sizes



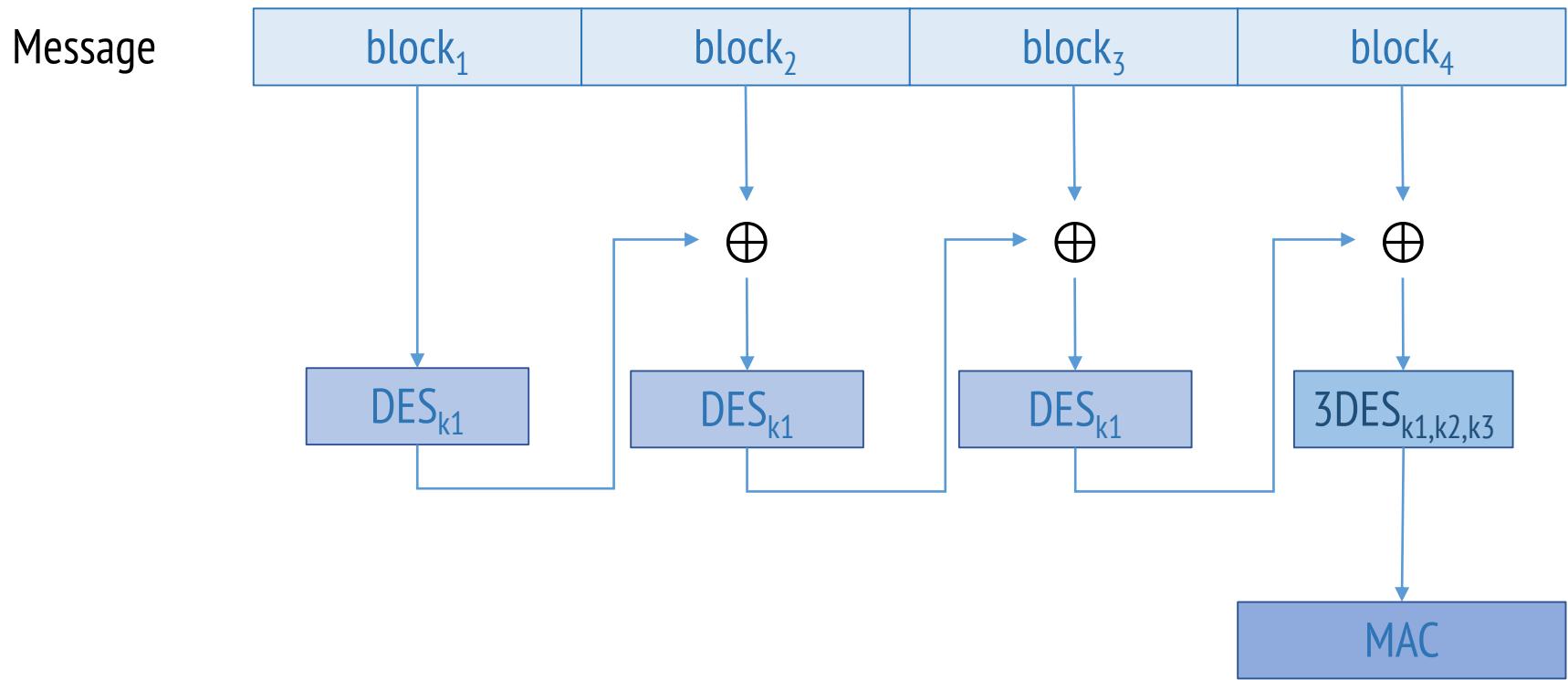
1. B. Preneel and P. C. van Oorschot. Key recovery attack on ANSI X9.19 retail MAC. *Electronics Letters*, 1996
2. H. Handschuh and B. Preneel. Minding your MAC algorithms. *Information Security Bulletin*, 2004.
3. Bhargavan, Karthikeyan, and Gaëtan Leurent. "On the practical (in-) security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN." *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

Collisions in ciphers with small block sizes

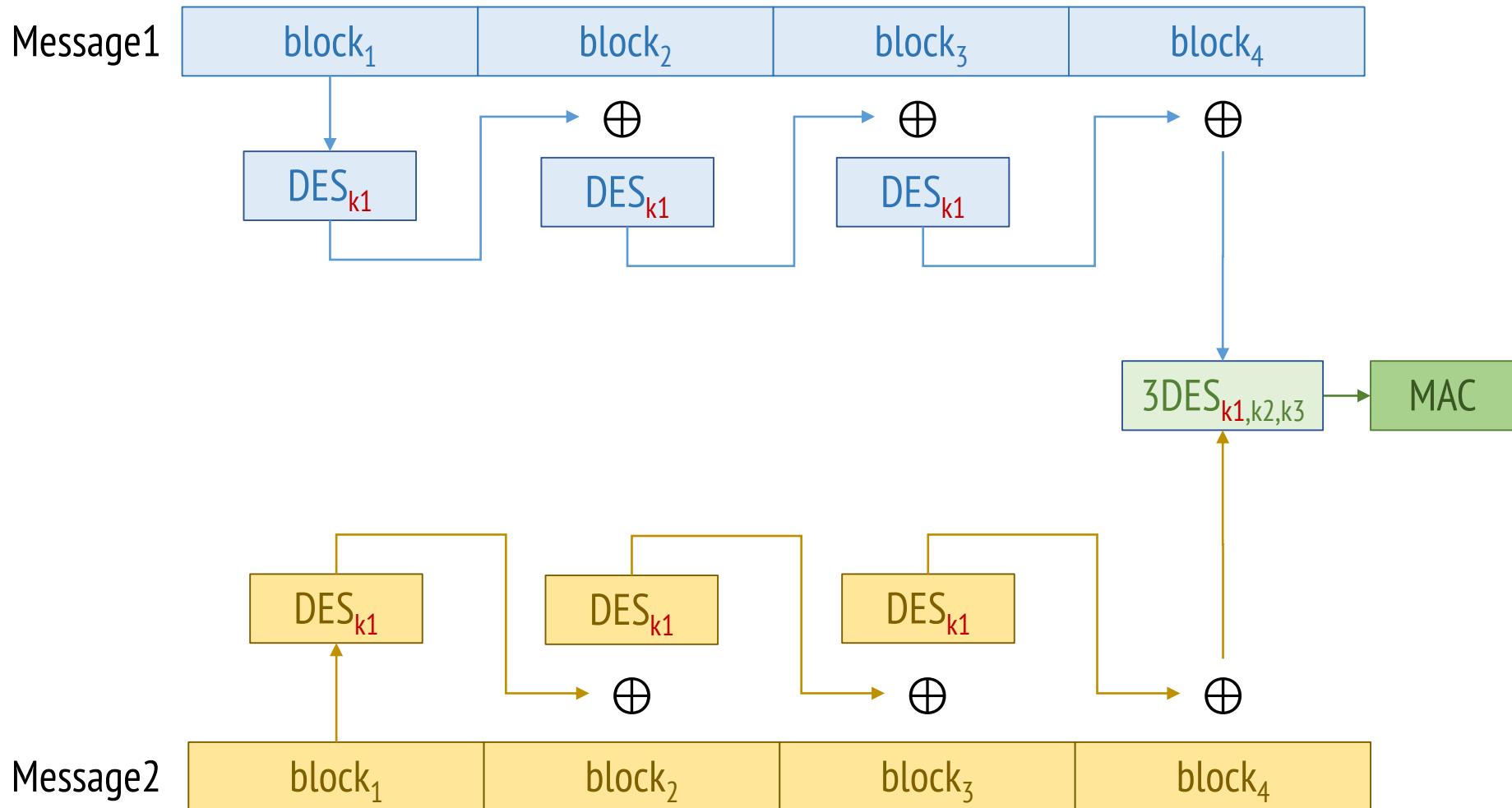


1. B. Preneel and P. C. van Oorschot. Key recovery attack on ANSI X9.19 retail MAC. *Electronics Letters*, 1996
2. H. Handschuh and B. Preneel. Minding your MAC algorithms. *Information Security Bulletin*, 2004.
3. Bhargavan, Karthikeyan, and Gaëtan Leurent. "On the practical (in-) security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN." *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

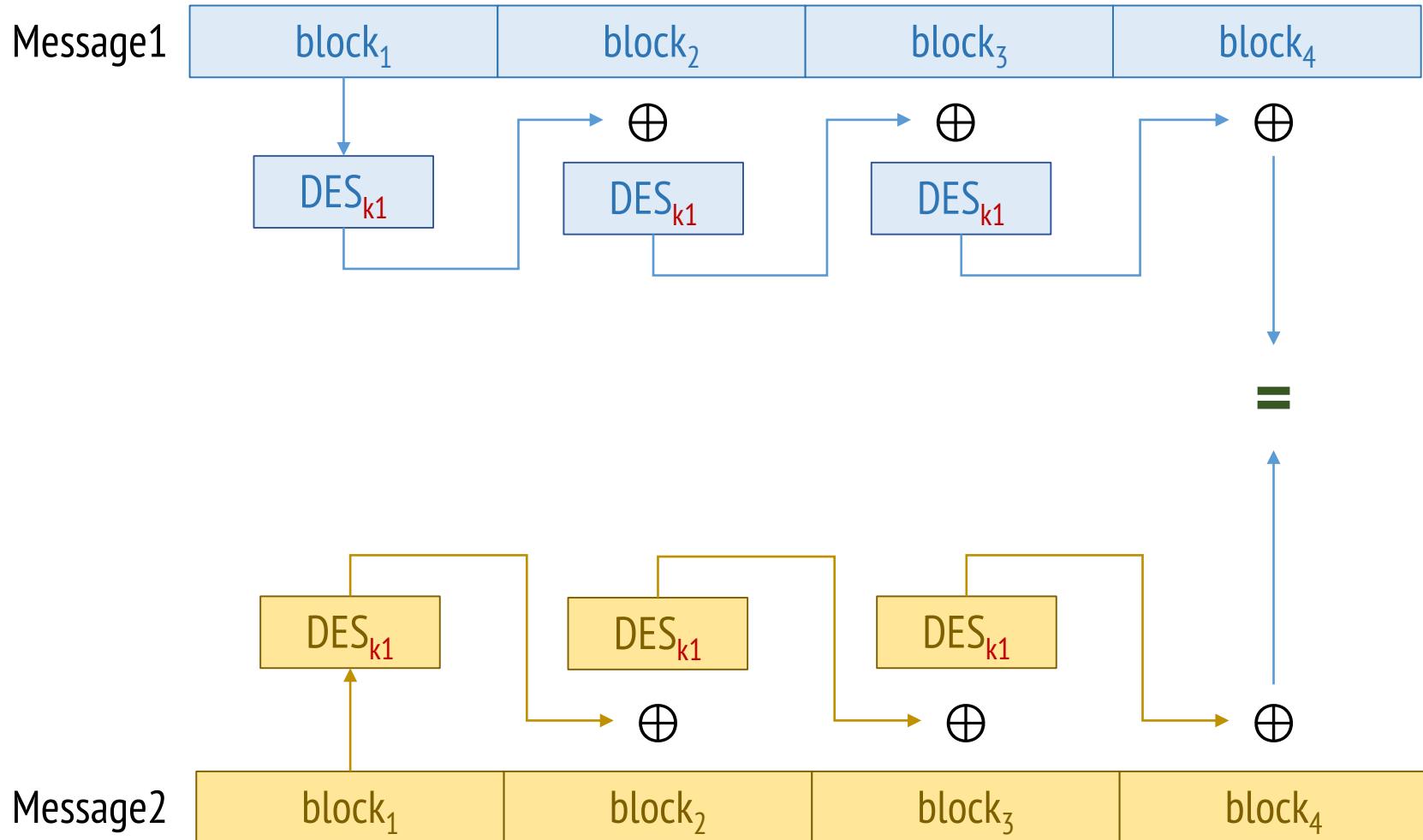
EuroRadio MAC



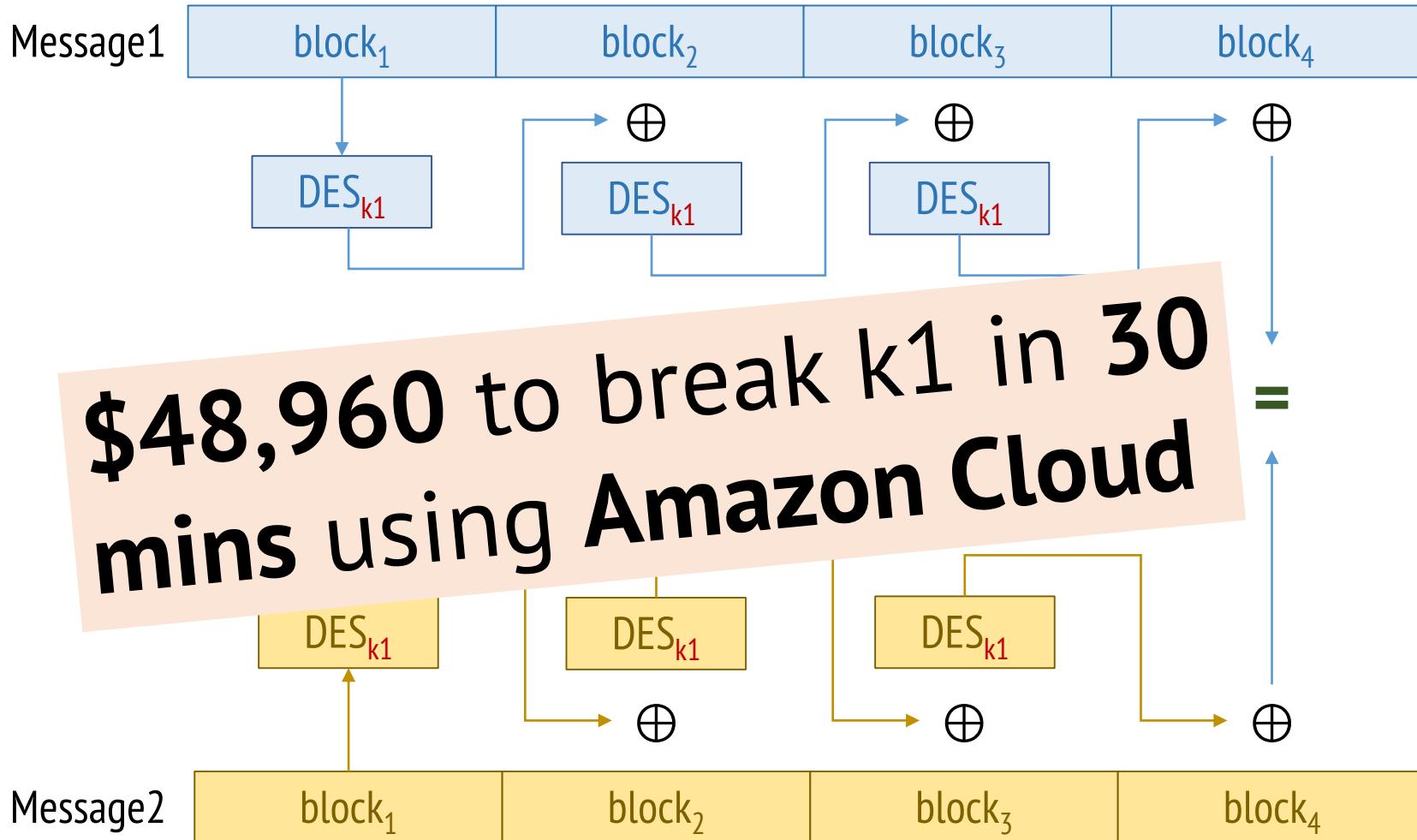
EuroRadio DES key recovery (when a collision happens)



DES key recovery (when a collision happens)



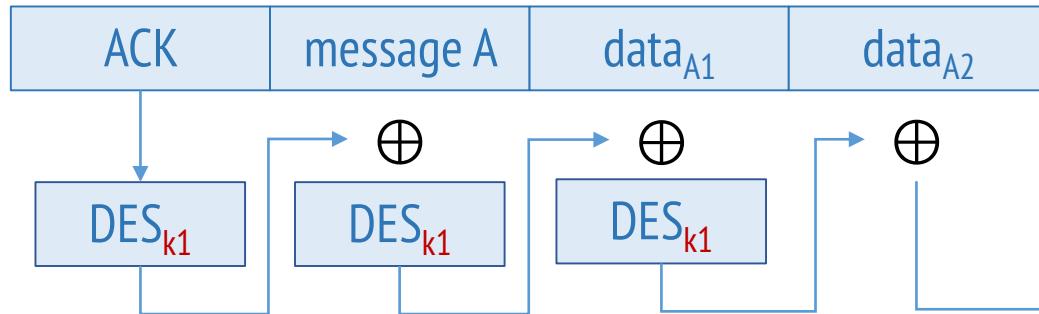
DES key recovery (when a collision happens)



Message forging

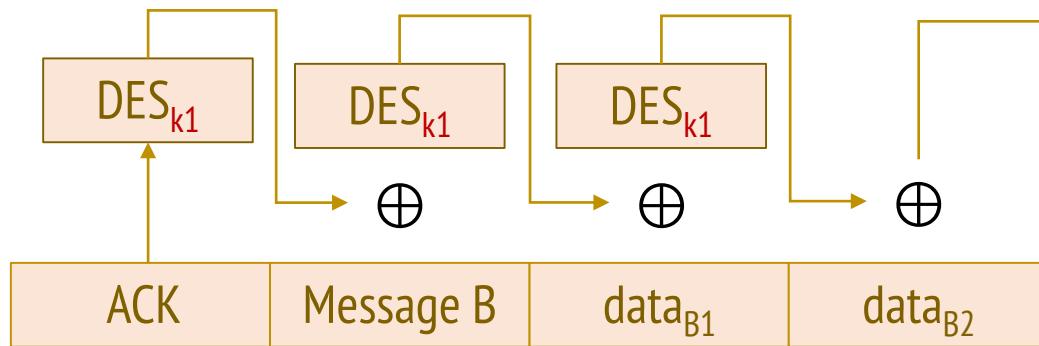
K1=✓
K2=✗
K3=✗

Message1



3DES_{k1,k2,k3} → MAC

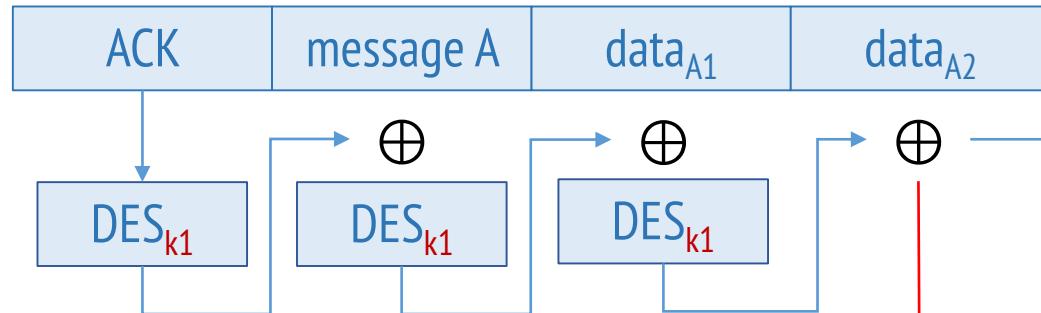
Forged
message



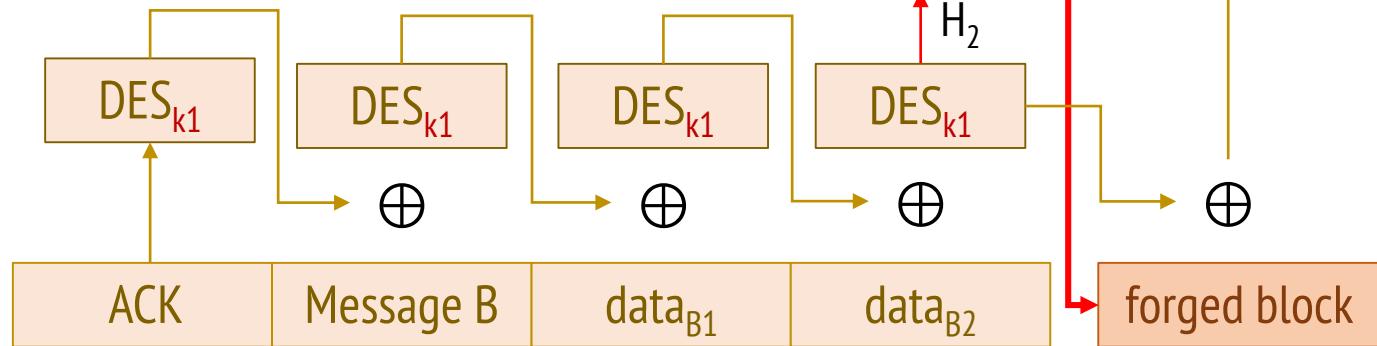
Message forging

K1=✓
K2=✗
K3=✗

Message1



Forged
message



Message forging

K1=✓
K2=x
K3=x

Message1



DES_{k1}

\oplus

DES_{k1}

DES_{k1}

\oplus

H_1

3DES _{$k1, k2, k3$}

MAC

Detected by the Application Layer!

Forged message



DES_{k1}

\oplus

DES_{k1}

\oplus

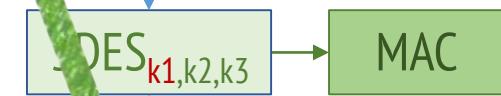
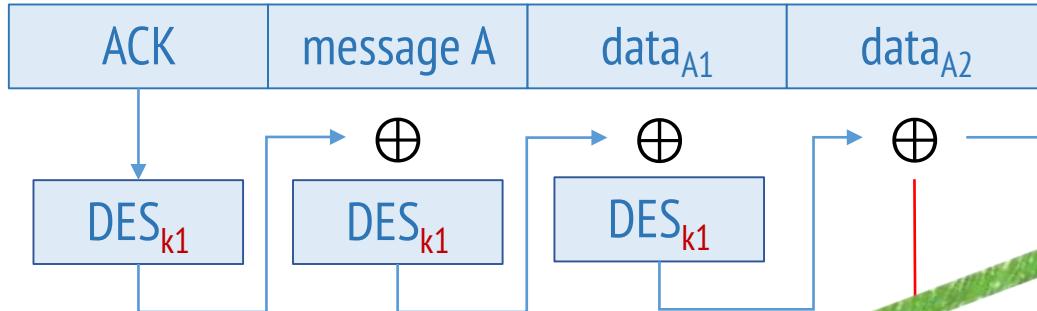
forged block

\oplus

Message forging

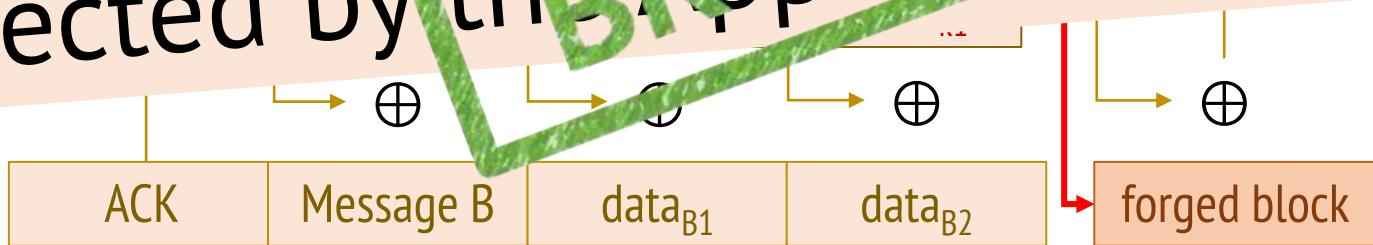
K1=✓
K2=x
K3=x

Message1



Detected by the Application Layer!

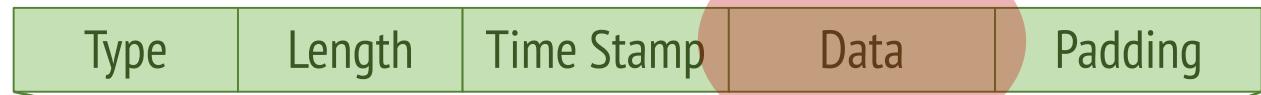
Forged message



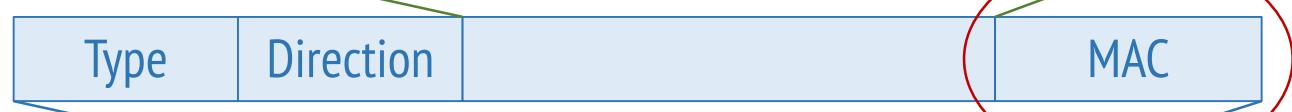
ERTMS stack

K1=✓
K2=x
K3=x

Application Layer



EuroRadio



GSM-R



ERTMS - Application Layer

- Transmits train control messages and signalling
- Messages can be of multiple types
 - Movement authorities
 - Display message
 - Acknowledgment message

Application layer

K1=✓
K2=✗
K3=✗

Message1

Movement	authority	150mph	100 miles	MAC ₁
----------	-----------	--------	-----------	------------------

Message2

Display	message	“speed has	increased”	MAC ₂
---------	---------	------------	------------	------------------

Application layer

K1=✓
K2=x
K3=x

Message1

Movement	authority	150mph	100 miles	MAC ₁
----------	-----------	--------	-----------	------------------

Message2

Display	message	“speed has increased”	MAC ₂
---------	---------	-----------------------	------------------

Message1

Message2

Movement	authority	150mph	100miles	Display	message	“speed has increased”	MAC ₃
----------	-----------	--------	----------	---------	---------	-----------------------	------------------

Application layer

K1=✓
K2=x
K3=x

Message1

Movement	authority	150mph	100 miles	MAC ₁
----------	-----------	--------	-----------	------------------

Message2

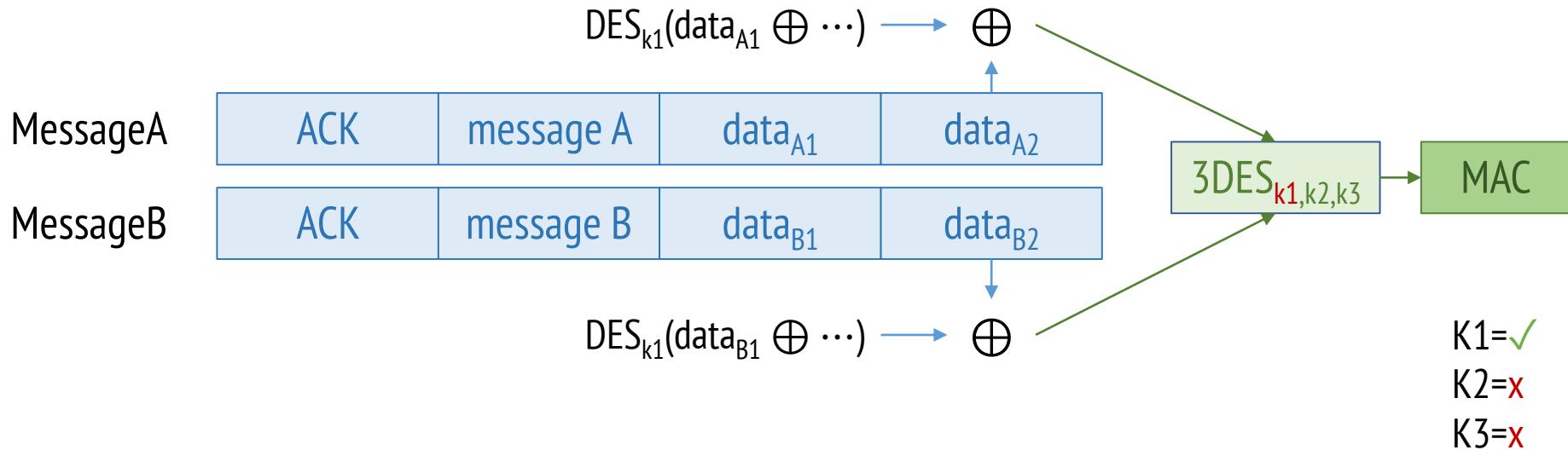
Display	message	“speed has increased”	MAC ₂
---------	---------	-----------------------	------------------

Message1

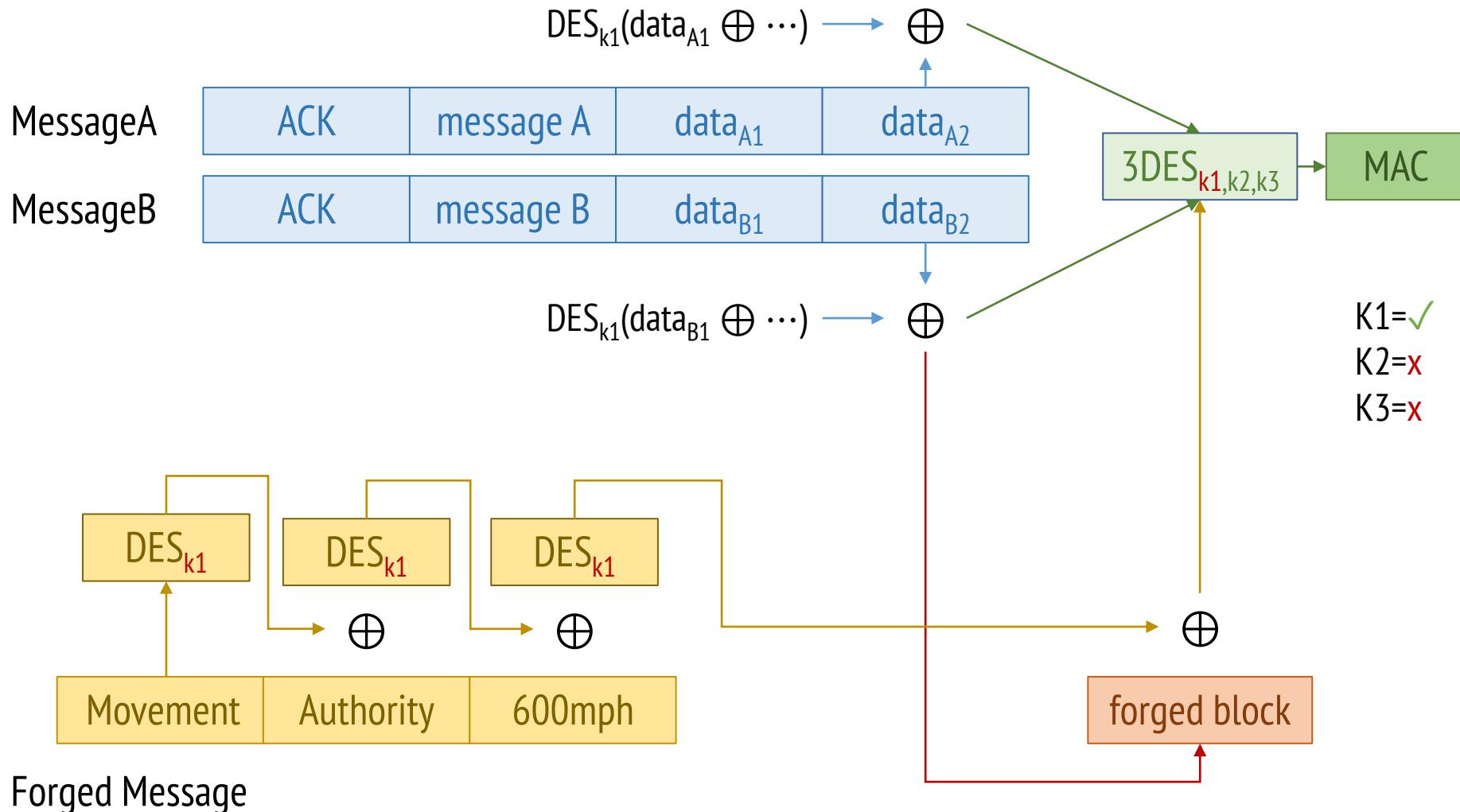
Display message accepts Unicode characters!

Movement	authority	150mph	100miles	Display	message	“speed has increased”	MAC ₃
----------	-----------	--------	----------	---------	---------	-----------------------	------------------

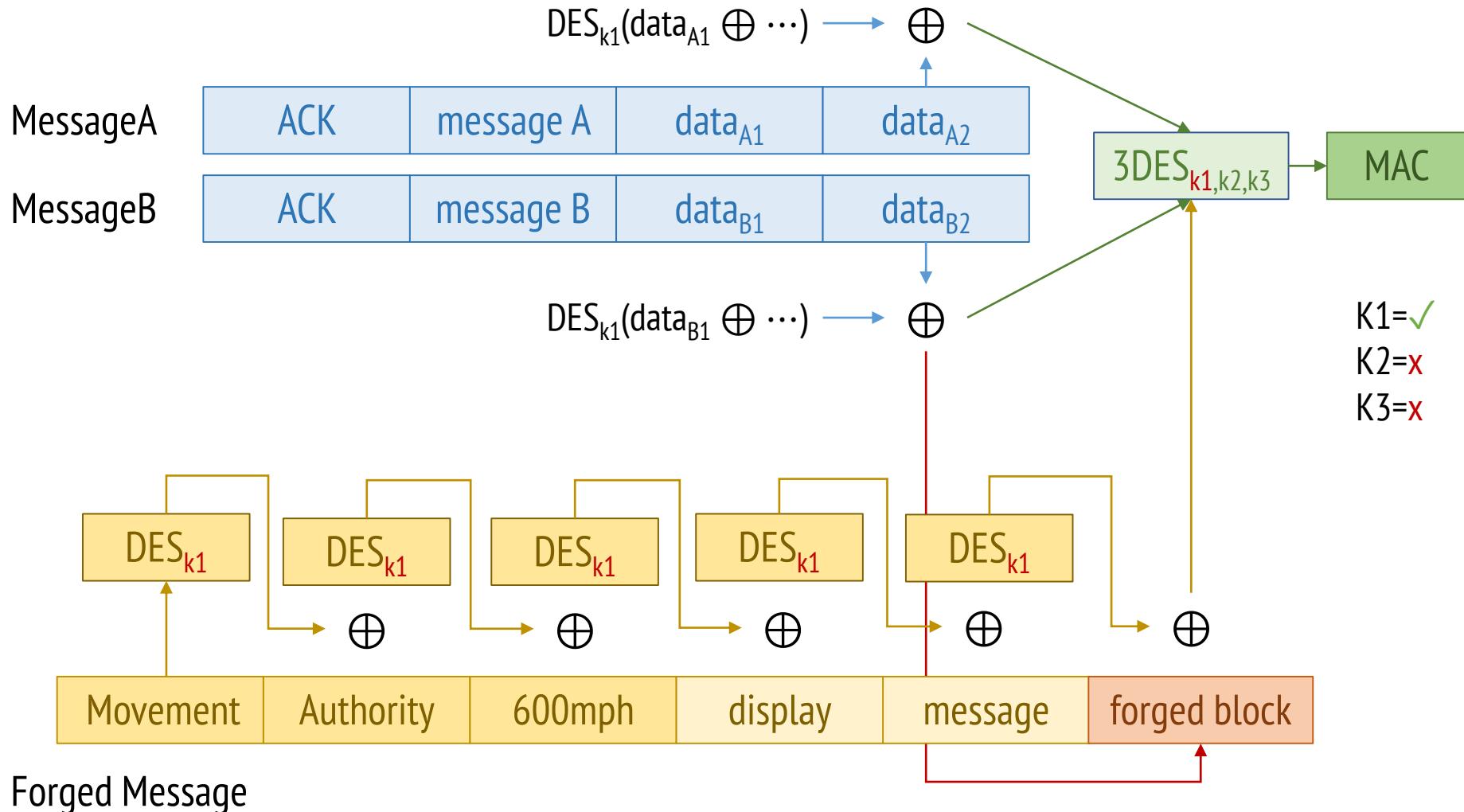
Leveraging collisions



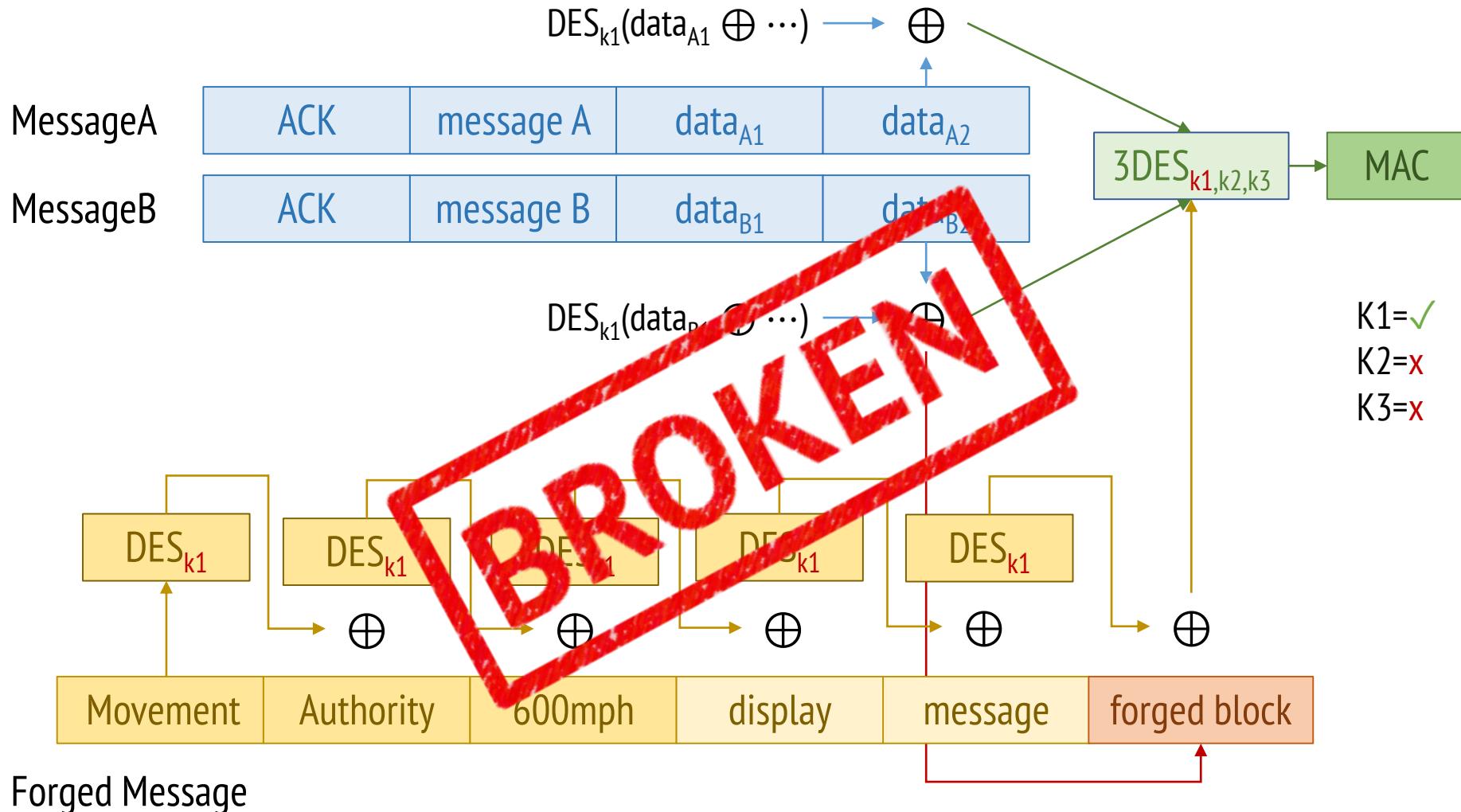
DES key recovery



Message concatenation



Message concatenation



ERTMS stack vulnerabilities

Application Layer



EuroRadio



GSM-R



ACK message collision

Two acknowledgement messages:

00120000020A9203A2105E0480000062105DFD0000000000

MAC: 80B7557F31566DBB

00120000020A9203AAE360078000006AE36000000000000

MAC: 80B7557F31566DBB

Forged movement authority

Variable	Length (bits)	Value	Description
NID_PACKET	8	0000 1111	Level 2/3 movement authority (only RBC)
Q_DIR	2	10	Both directions
L_PACKET	13	0 0000 0111 0001	113 bits
Q_SCALE	2	10	10 m
V_LOA	7	111 1000	600 km/h
T_LOA	10	11 1111 1111	Unlimited
N_ITER	5	0 0000	0 iterations
L_ENDSECTION	15	111 1111 1111 1111	327670 meter
Q_SECTIONTIMER	1	0	No section timer information
Q_ENDTIMER	1	0	No end section timer information
Q_DANGERPOINT	1	0	No danger point information
Q_OVERLAP	1	1	Overlap information to follow
D_STARTOL	15	000 0000 0000 0000	0 meter
T_OL	10	00 0000 0000	0 sec
D_OL	15	000 0000 0000 0000	0 meter
V_RELEASEOL	7	111 1110	Use onboard calculated release speed

Forged display message

Variable	Length (bits)	Value	Description
NID_PACKET	8	0100 1000	Packets for sending plain text messages
Q_DIR	2	00	Reverse
L_PACKET	13	0 0000 1101 1100	220 bits
Q_SCALE	2	10	10 m
Q_TEXTCLASS	2	00	Auxiliary
Q_TEXTDISPLAY	1	0	no, as soon until events fulfilled
D_TEXTDISPLAY	15	111 1111 1111 1110	327660 Meter
M_MODETEXTDISPLAY	4	1001	System failure
M_LEVELTEXTDISPLAY	3	000	Level 0
L_TEXTDISPLAY	15	000 0000 0000 0000	0 Meter
T_TEXTDISPLAY	10	00 0000 0000	0 sec
M_MODETEXTDISPLAY	4	1001	System failure
M_LEVELTEXTDISPLAY	3	000	Level 0
Q_TEXTCONFIRM	2	00	No confirmation required
L_TEXT	8	0001 0000	16 Chars
X_TEXT	128	...	Text messsage...

Encoded messages example

ACK M1: 00120000020A9203A2105E0480000062105DFD00000000000

MAC: 80B7557F31566DBB

ACK M2: 00120000020A9203AAE360078000006AE3600000000000000

MAC: 80B7557F31566DBB

FORGED MSG: continue at 600km/h; display “Z|1MB\%<w*RRf)8n/”

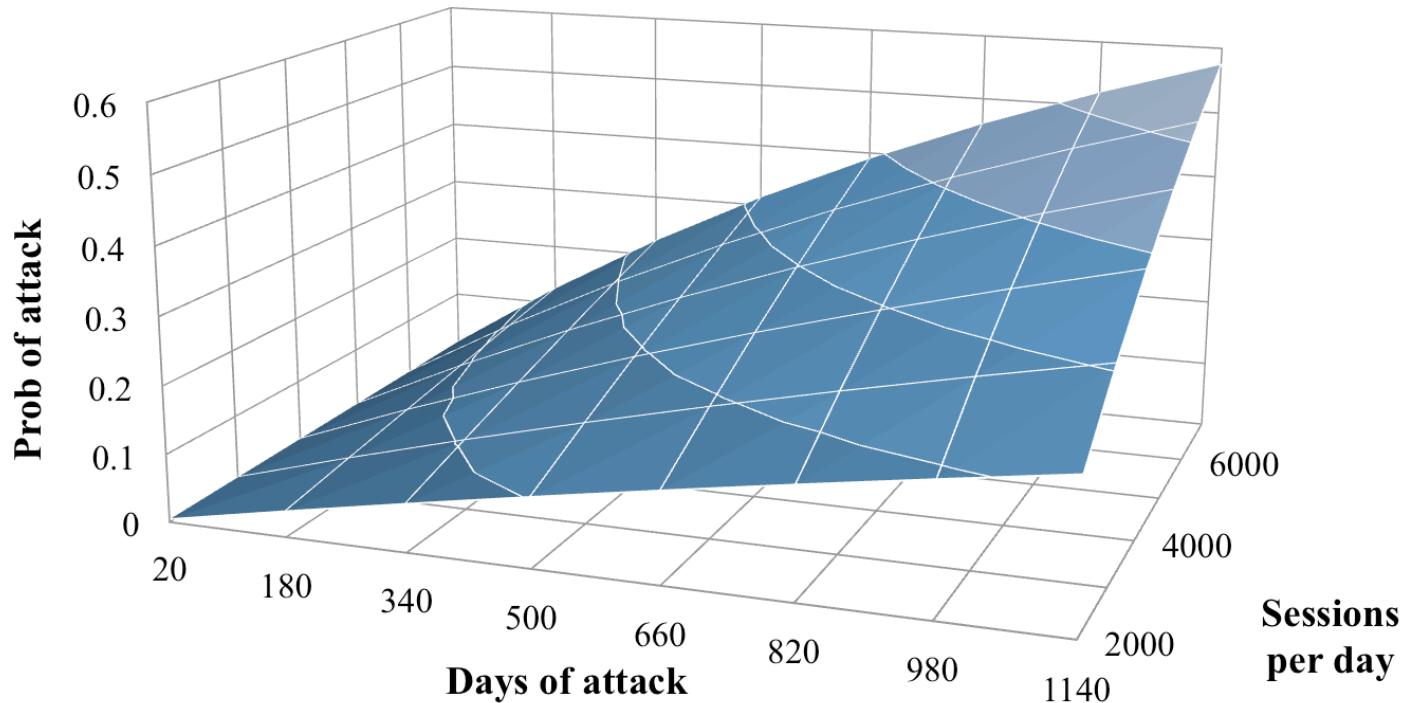
030CD3C677A100000021F01C651FF809C4080000000007E4801
B90FFF2000000120105A7C314D42253C772A52526629386E2F

MAC: 80B7557F31566DBB

Impact

1. The attack depends on the ability to discover a collision (assuming that the key used by DES can be brute-forced).
2. Cipher collisions depends on the ability to capture the right amount of traffic.

Impact



$$P_{collision} = 1 - \prod_{i=1}^{M-1} \left(1 - \frac{i}{N}\right)^S \approx 1 - e^{\frac{-M(M-1)}{2N} \cdot S}, N = 2^{64}$$

Assumptions

- Message collision chance: 1%
- Average message length 32 bytes
- GSM-R speed 10Kbps (14Kbps max)
- UK rail network:
 - 4000 trains per day
 - 10h sessions

Data capture

- 1% chance of collision requires ~600,000,000 messages.
 - *32 byte messages, 10 Kbps bandwidth*
- Safe limit for a EuroRadio session: 19 GB.
- This would require a single session lasting 22 days!
- No threat to current trains.

Data capture

- 1% chance of collision requires ~600,000,000 messages.
 - *32 byte messages, 10 Kbps bandwidth*
- If we could monitor next generation UK rail backbone(s).
 - 4000 trains per day, 10 hour sessions.
- 1% chance of attack in 45 days. 50% chance ~ 8 years
 - This *might* be a problem.

ERTMS conclusion

- Defence in depth did not help, there were issues on every protocol layer.
- The specification fails to meet safety standards.
- Poorly designed: ERTMS is the next gen in rail technology, but has been designed with obsolete ciphers.

Payment protocols

Payment protocols

- PayWave(Visa)/PayPass(MasterCart)
contactless protocols
- ApplePay protocol

Purpose of a payment protocol

1. Provide evidence to the shop reader that the card is genuine.
2. Give the shop reader a cryptogram (i.e., AC) which it can send to the bank as proof that a payment is due.

PayWave/PayPass protocols

- Based on the contact-based EMV standard
- 7 different payment protocol variations
 - Visa's PayWave and MasterCard's PayPass are the most popular
- Not compatible with each other, but use similar commands

Cryptographic elements of the protocol

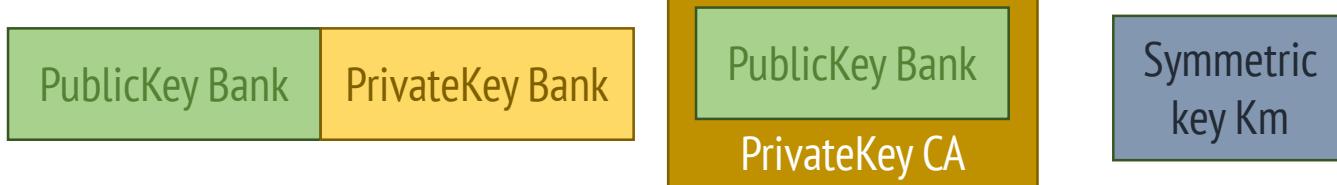
Card has:



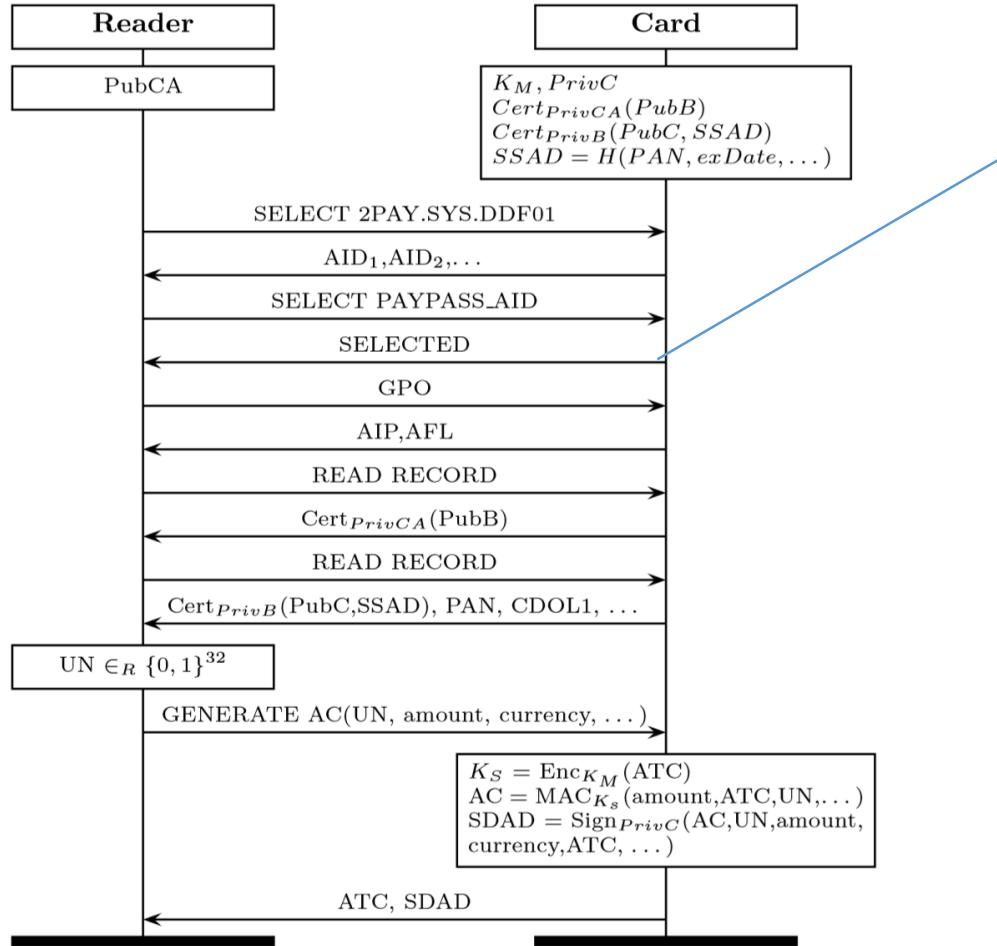
Card reader has:



Banks has:

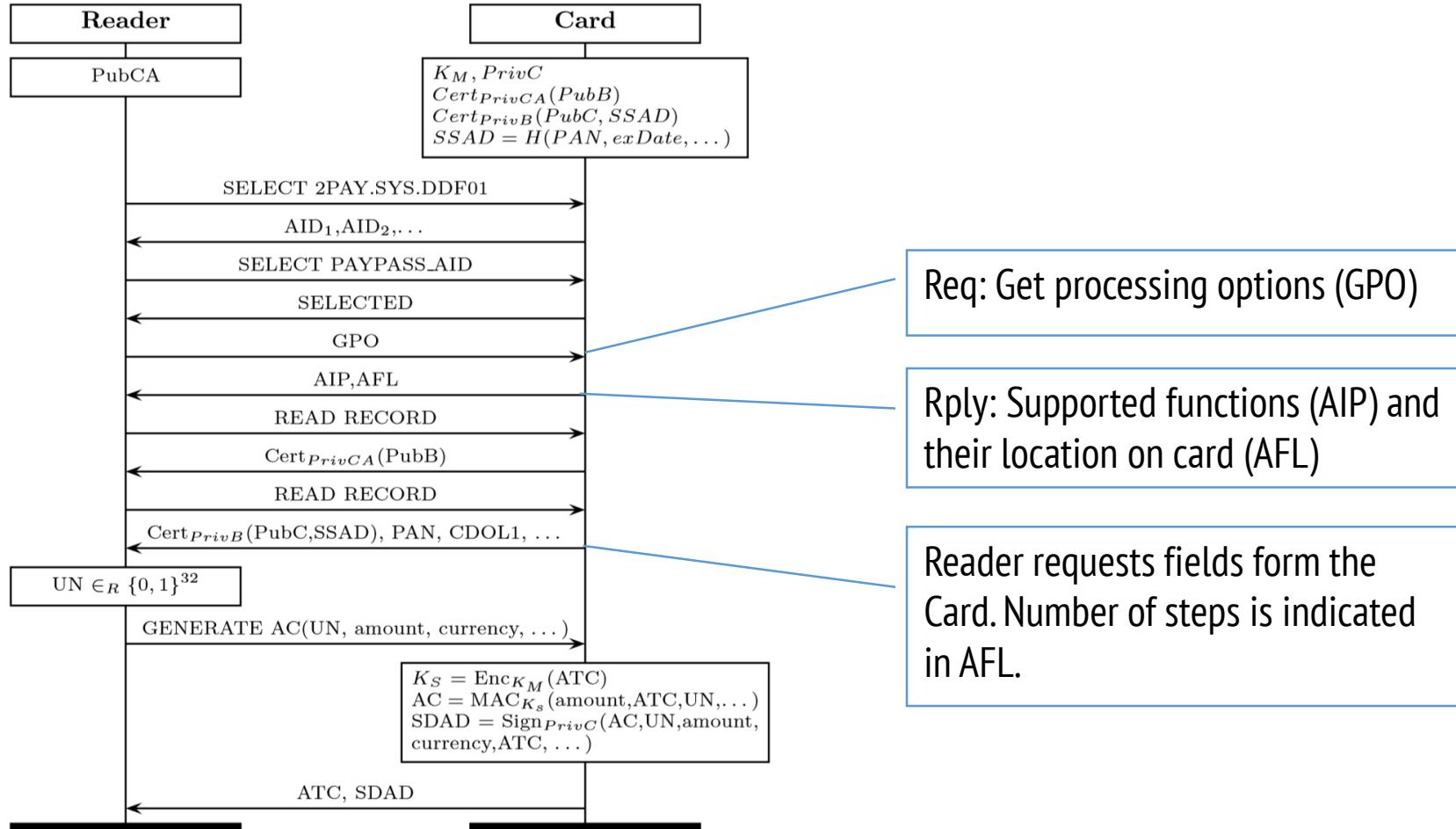


MasterCard PayPass protocol

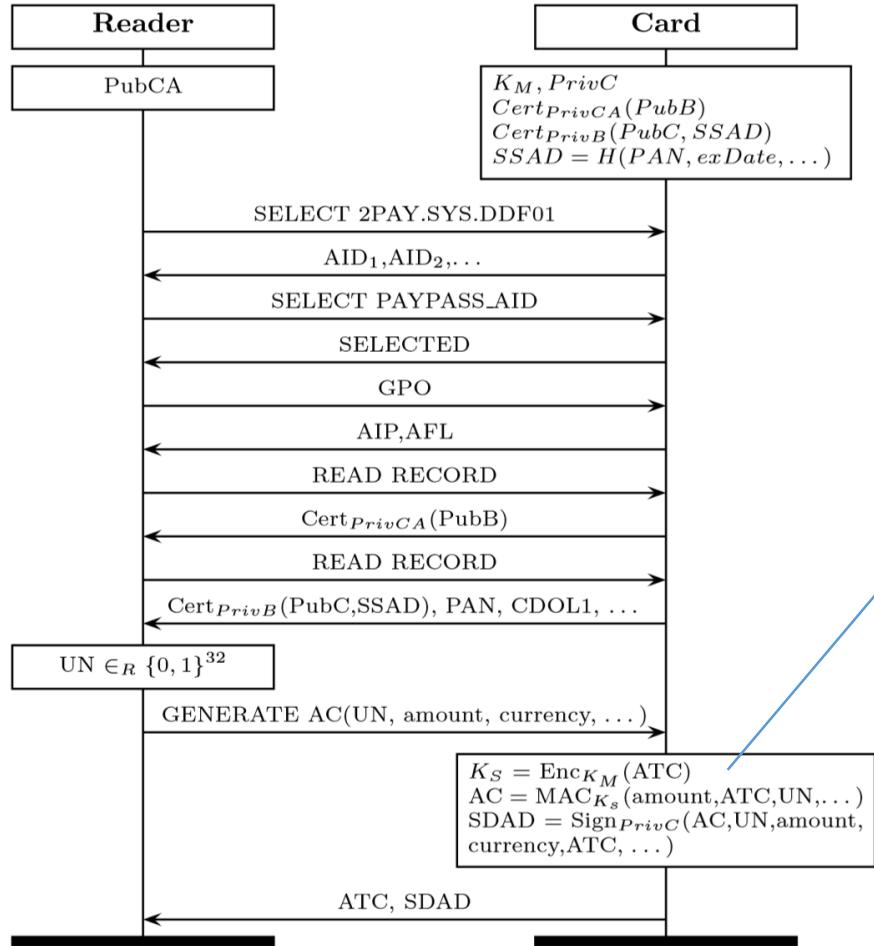


1. Shop: select payment application
2. Card: Present supported applications identities per protocol
3. Shop: select protocol
4. Acknowledge

MasterCard PayPass protocol

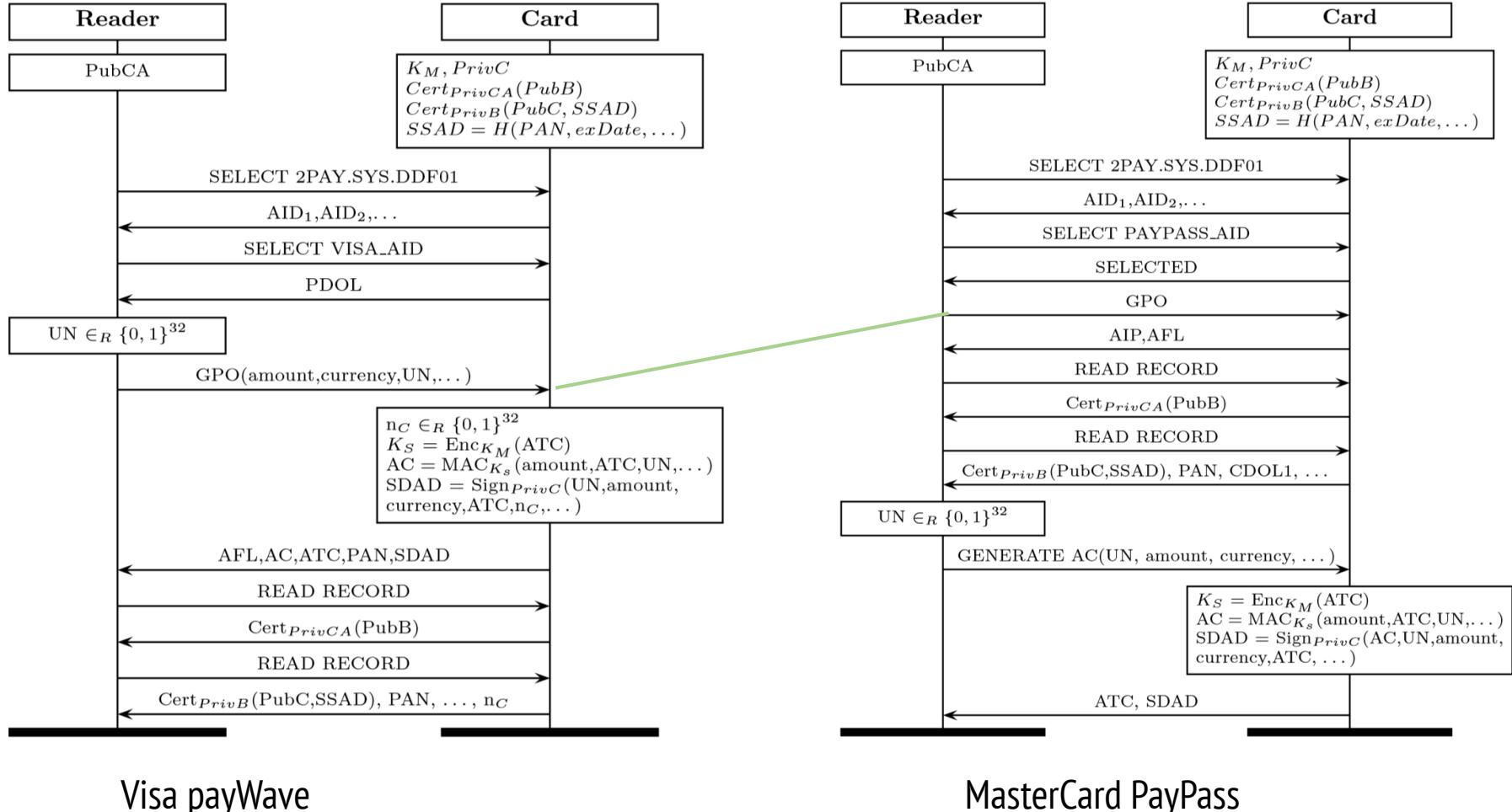


MasterCard PayPass protocol

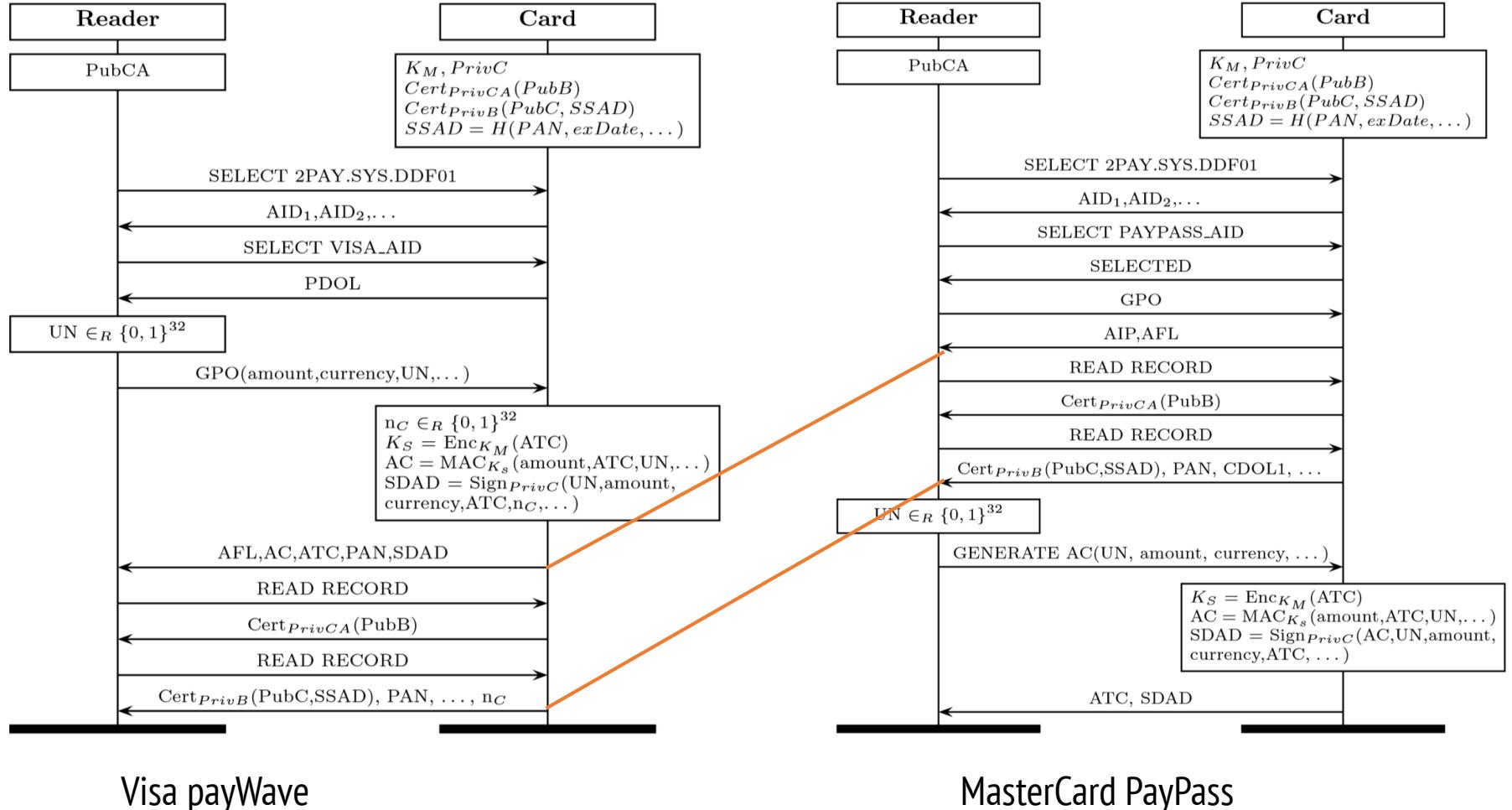


R: Generate nonce
 Request cryptogram (AC)
 C: Generate Ks from **Km** and **ATC**
 Generate cryptogram for bank using Ks (AC)
 Generate verifiable msg. for the Reader (SDAD)

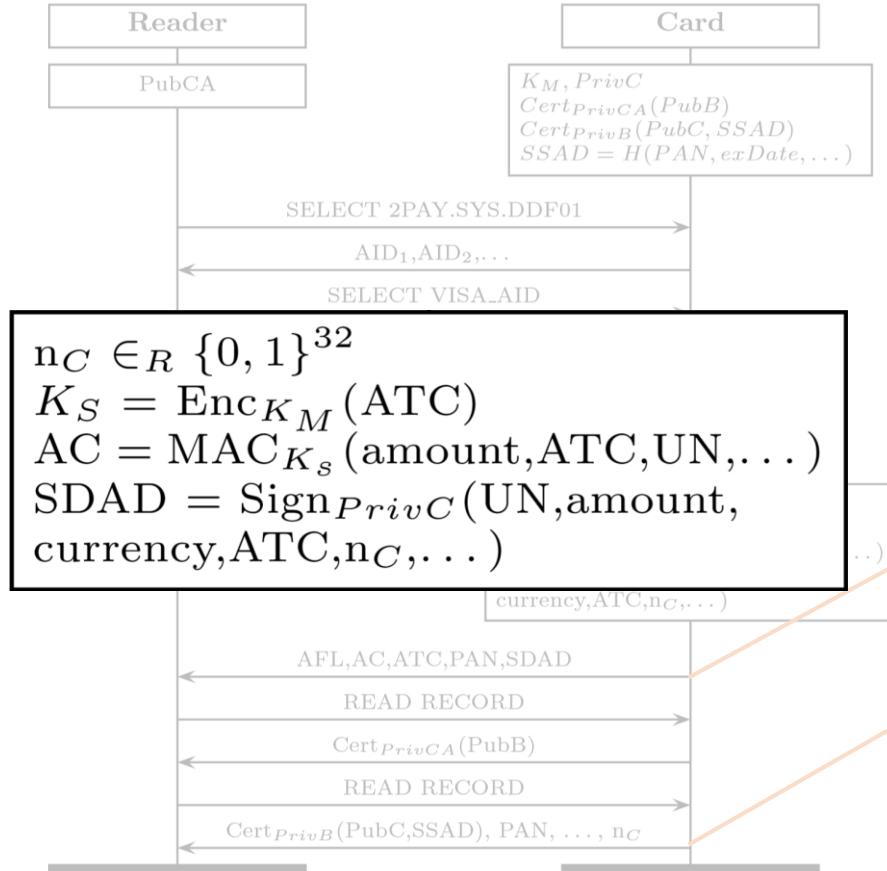
Visa PayWave protocol



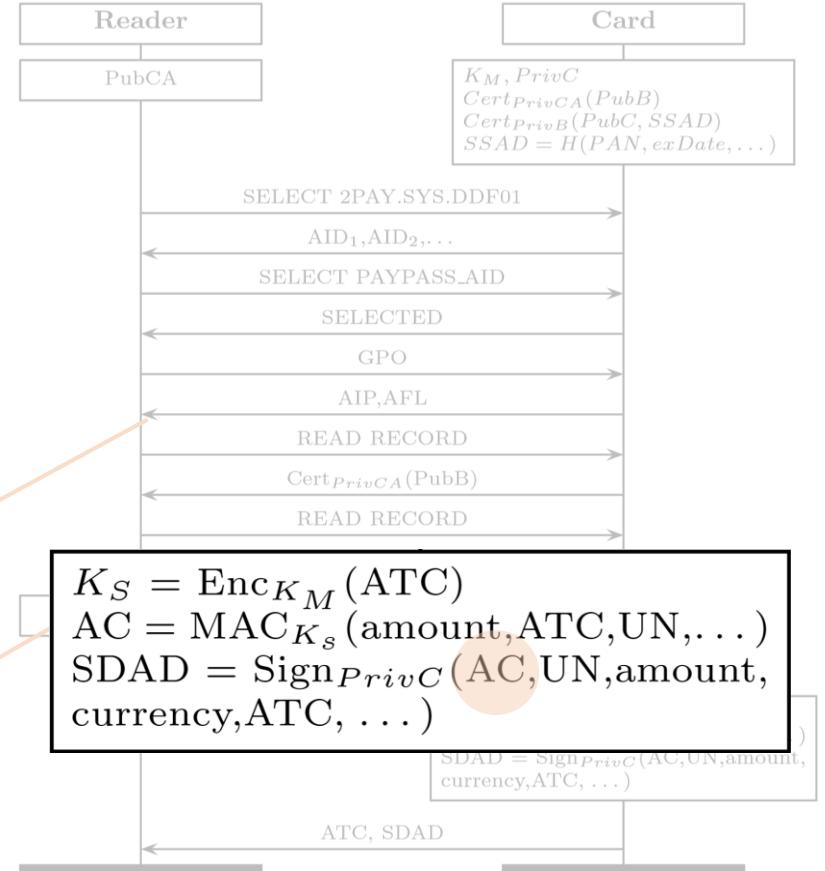
Visa PayWave protocol



Visa PayWave protocol

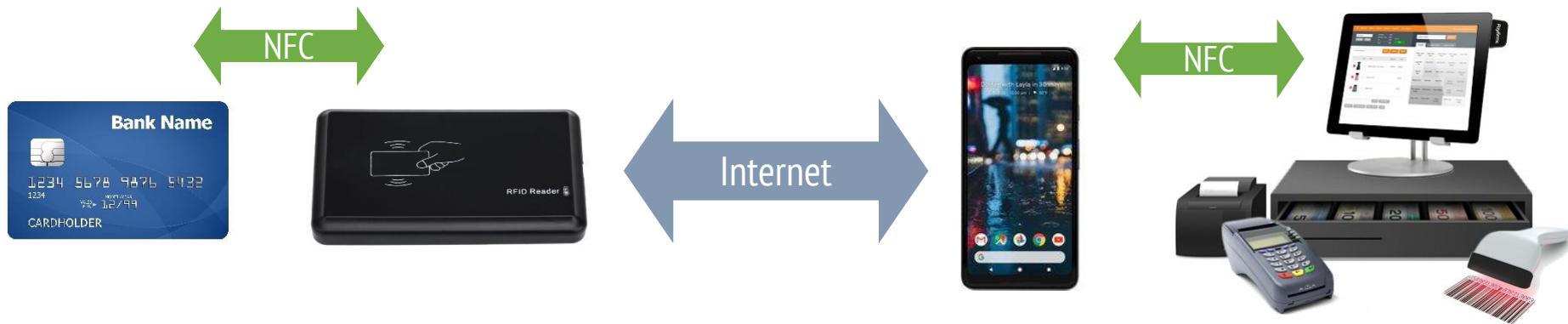


Visa payWave



MasterCard PayPass

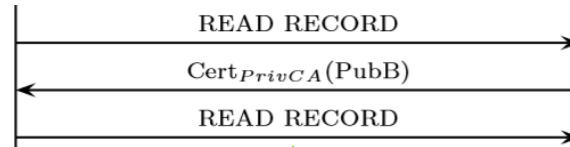
Relay attacks against EMV Contactless Smart Cards



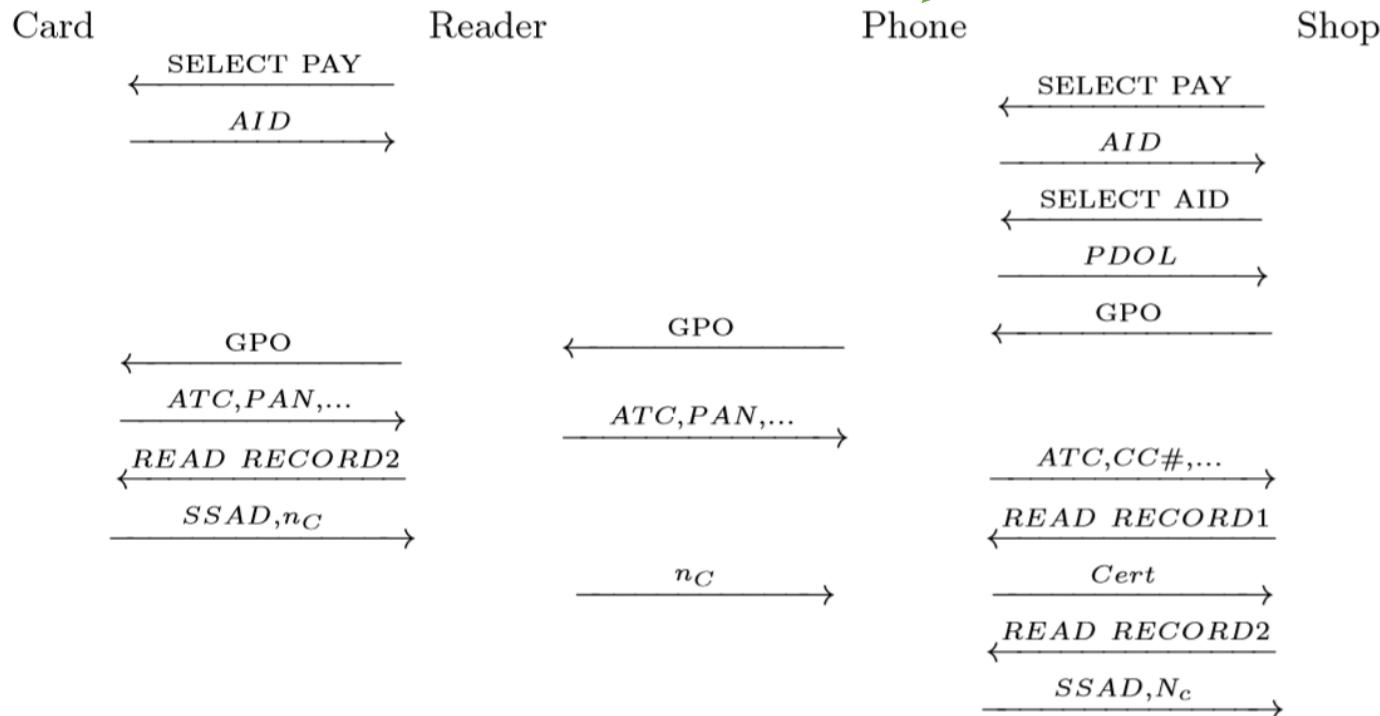


Relay attacks

1. Cache static data from card



2. Relay data to/from card



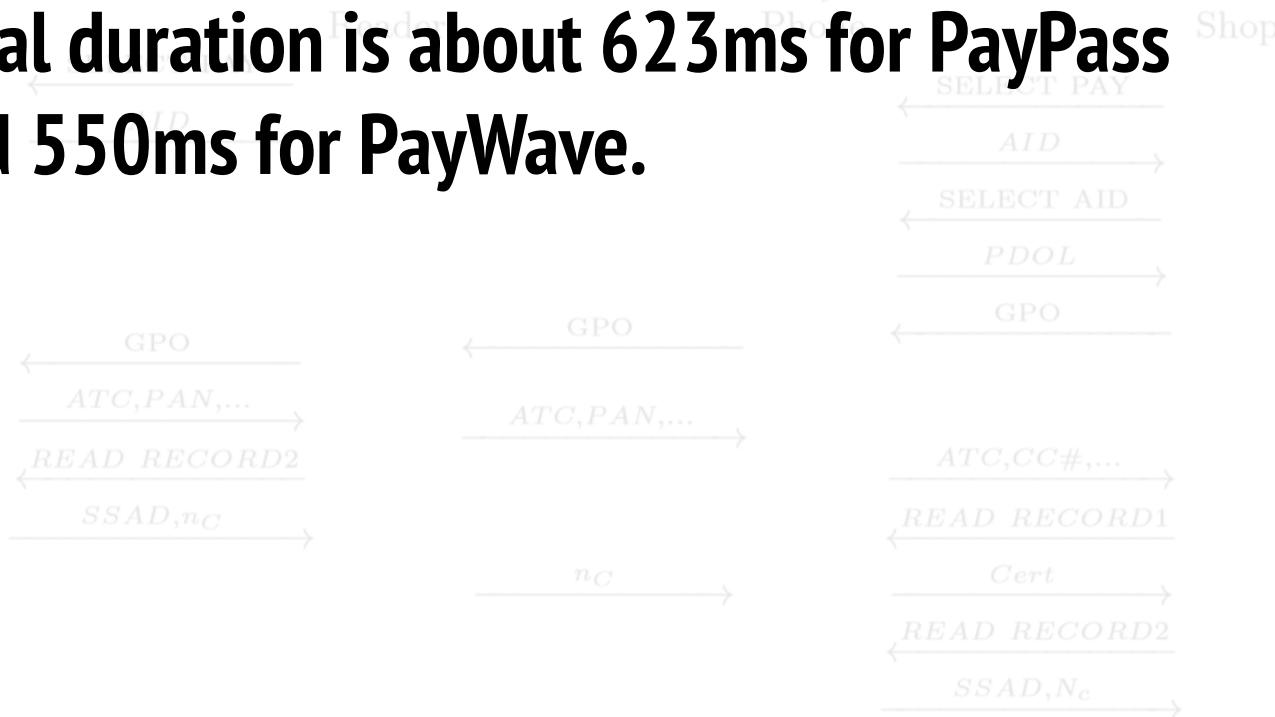
Relay attacks

1. Cache static data from card



2. Relay data to/from card

**Total duration is about 623ms for PayPass
and 550ms for PayWave.**



Fixing the protocol

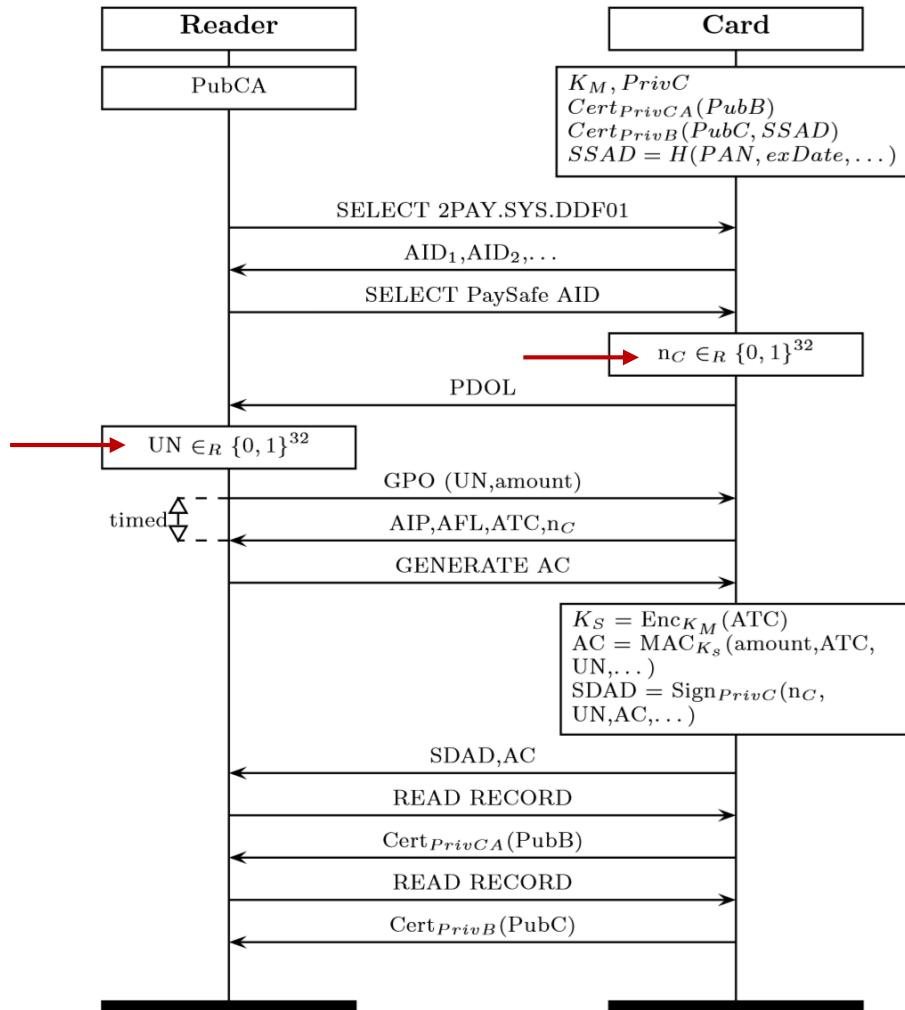
Challenges

1. The relay is very fast.
2. Cards and card positioning on the reader vary a lot and these affect protocol running time.
3. Find the right message exchange to measure.

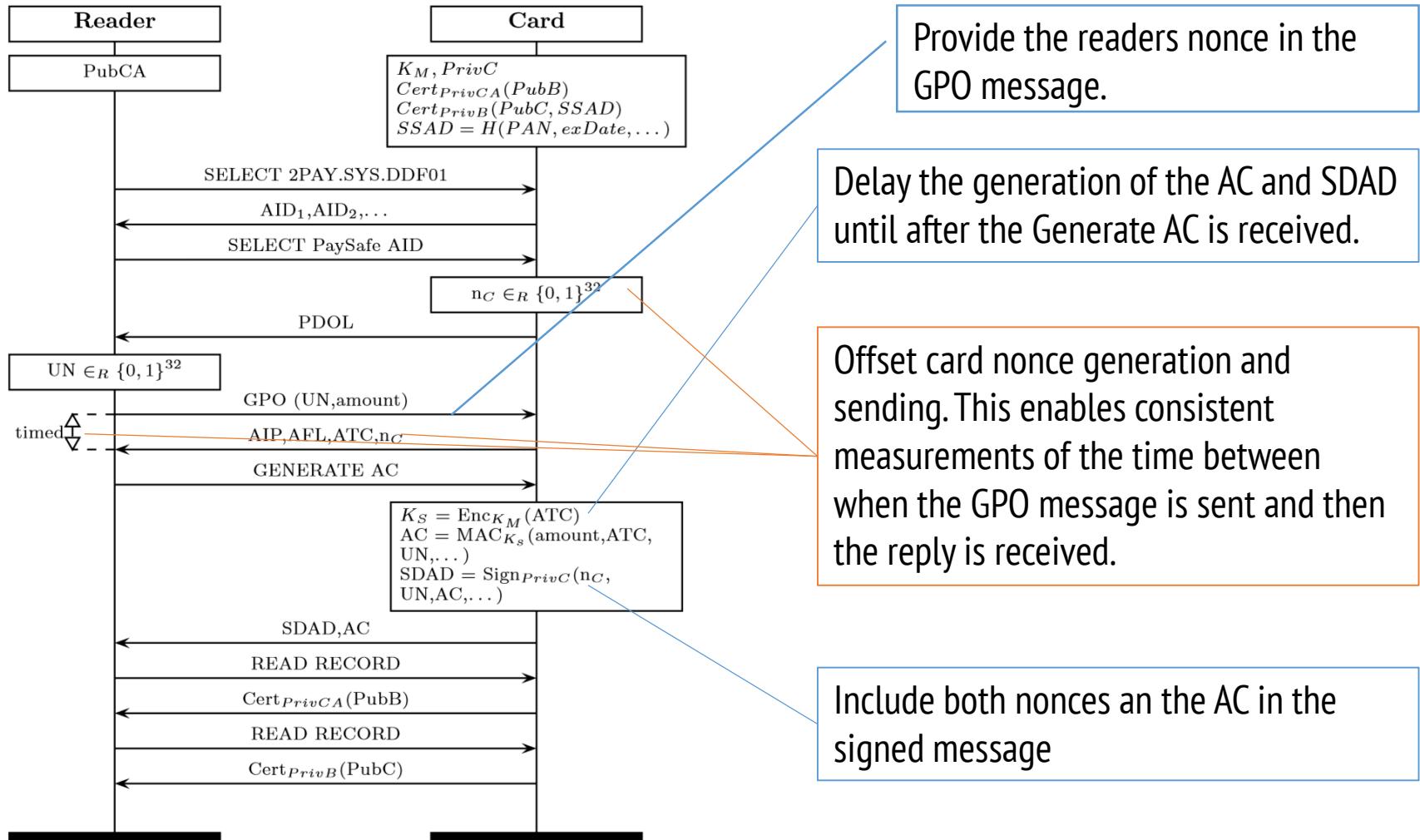
Solution

Split the challenge and response command from the generation of the signed authentication (SDAD) and the cryptogram (AC).

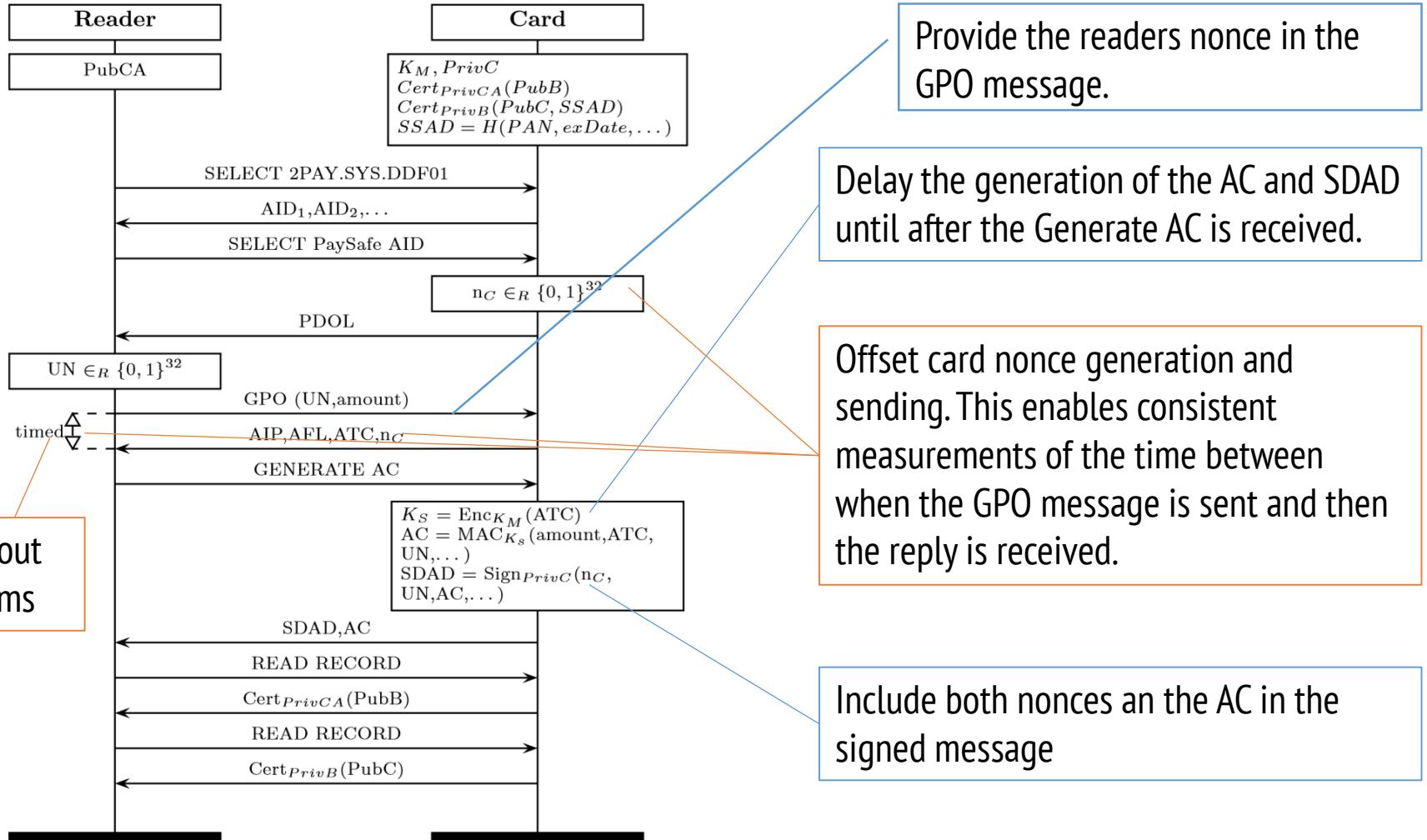
Fix: The PaySafe protocol



Fix: The PaySafe protocol



Fix: The PaySafe protocol



Relay attacks against EMV

- Relay attacks work against PayWave and PayPass.
 - Financial incentives are small (maximum transaction amount is ~30GBP).
-
- Time bounds the protocol to prevent relay attacks.
 - Ensures that no caching was done by including card and reader nonces.
 - Includes the cryptogram in the signed message to allow the “shop” to verify the transaction.

Mobile pay protocols

Components

- Secure element (SE)
- NFC controller
- Secure Enclave (SEP, TrustZone, etc.)
- Wallet App
- pay Servers

Secure element (SE)

“Secure Element (SE) is a tamper-resistant platform capable of securely hosting applications and their confidential and cryptographic data in accordance with the rules and security requirements set forth by a set of well-identified trusted authorities.” GlobalPlatform

Secure element (SE)

SE in mobile payments.

During a mobile payment the SE is used to emulate a contactless card using industry standard protocols.

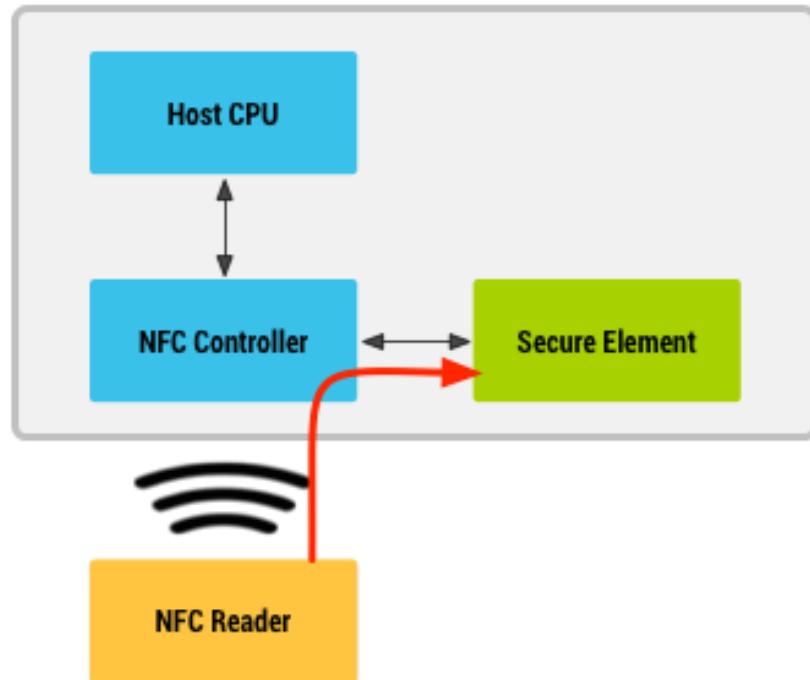
The SE can be:

Embedded in the phone

Embedded in the SIM card

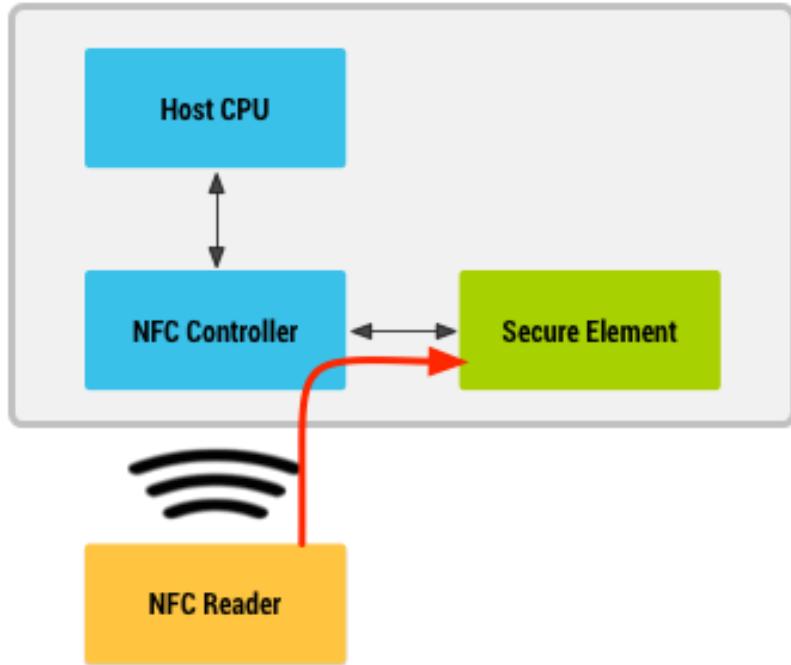
A cloud based service (Google HCE)

Simple protocol (Google Wallet v1.0)



1. Store card details (i.e. PAN) in the SE.
2. Use the NFC enabled device in card emulation mode to make payments.

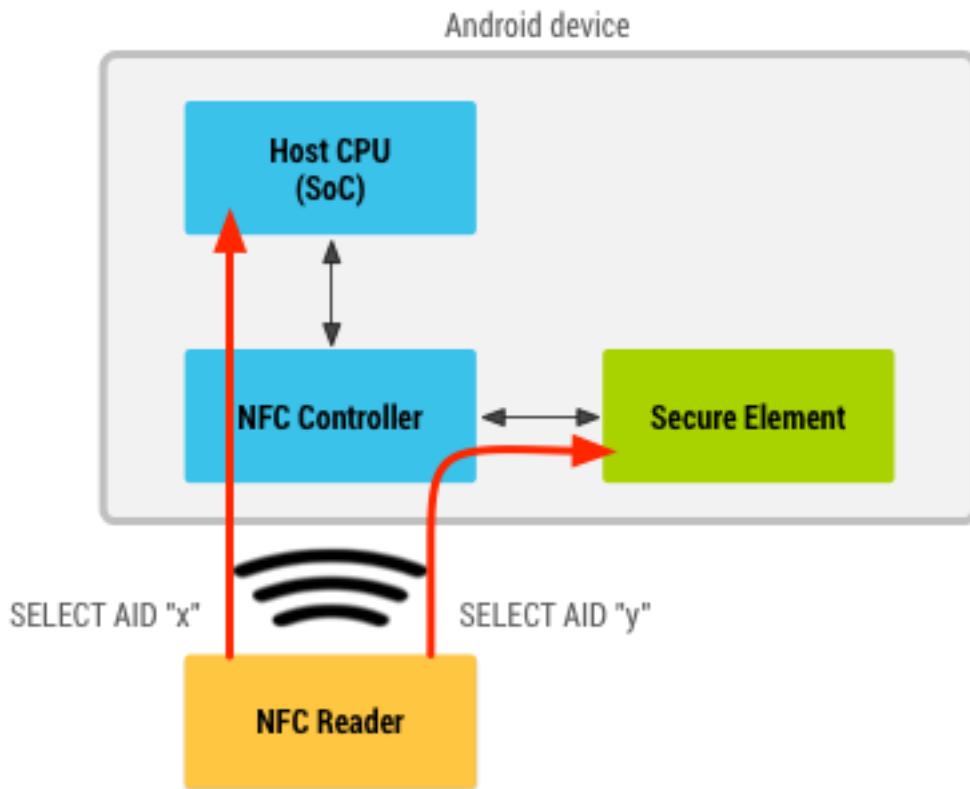
Simple protocol (Google Wallet v1.0)



1. Store card details (i.e. PAN) in the SE.
2. Use the NFC enabled device in card emulation mode to make payments.

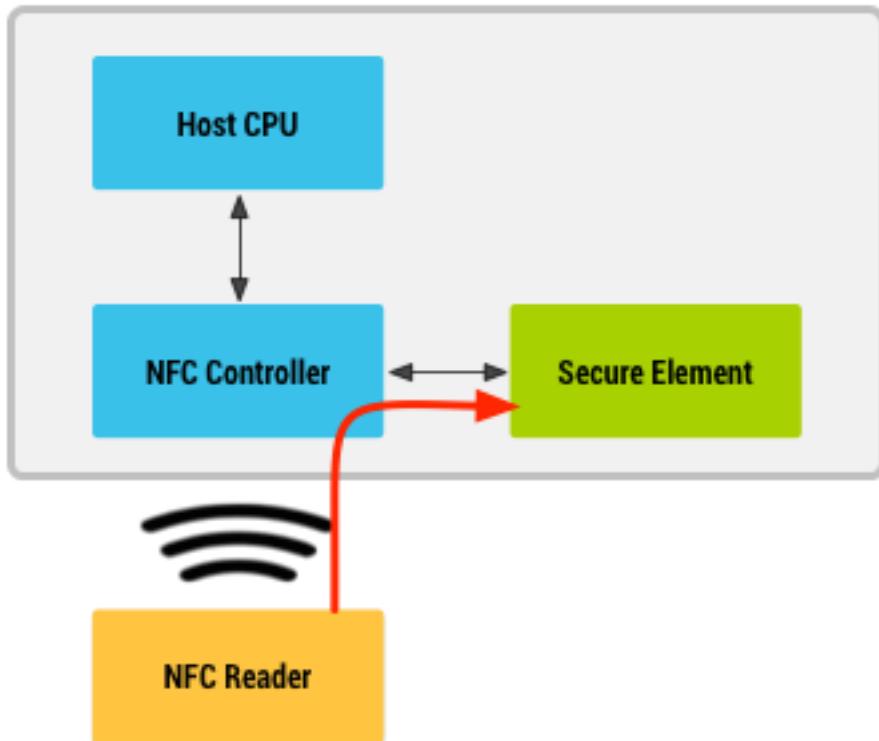
Didn't work out well for Google because network operators decided to support their own “wallets” (Softcard) and blocked access to the SE.

Host-based card emulation protocol (Google Wallet v3.0)



1. Use the NFC enabled device in card emulation mode to make a payment.
2. NFC communicates with the Host OS to request a virtual card number.
3. The Host OS forwards the transaction to the Google cloud.
4. Virtual card number is replaced with PAN and authorized with the Bank.

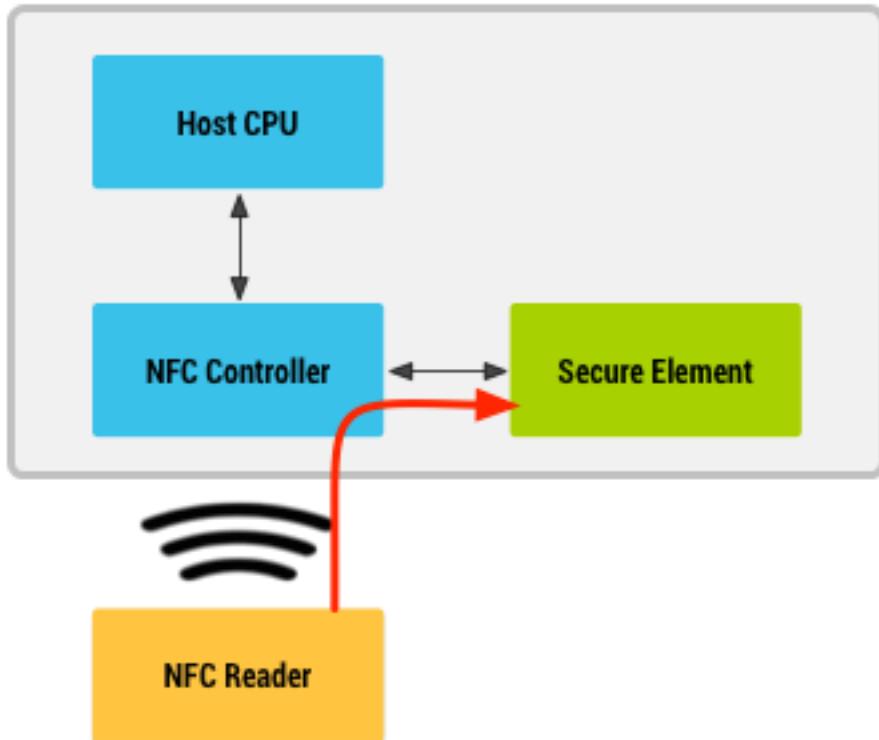
Apple Pay protocol



Similar to Google Wave 1.0:

1. Store card details (i.e. PAN) in the SE
2. Use the NFC enabled device in card emulation mode to make payments

Apple Pay protocol



Similar to Google Wave 1.0:

1. ~~Store card details (i.e. PAN) in the SE.~~
Store a EMVco Token.
2. Use the NFC enabled device in card emulation mode to make payments

EMVco Token

- A **token** is fake credit card number that looks and feels like a credit card number.
- The **tokenization** and the **de-tokenization** of a PAN are usually handled by the **Acquiring Bank**.
- In the **EMVCo.** tokenization standard the **de-tokenization** is performed by the **payment network** (e.g. MasterCard, Visa).

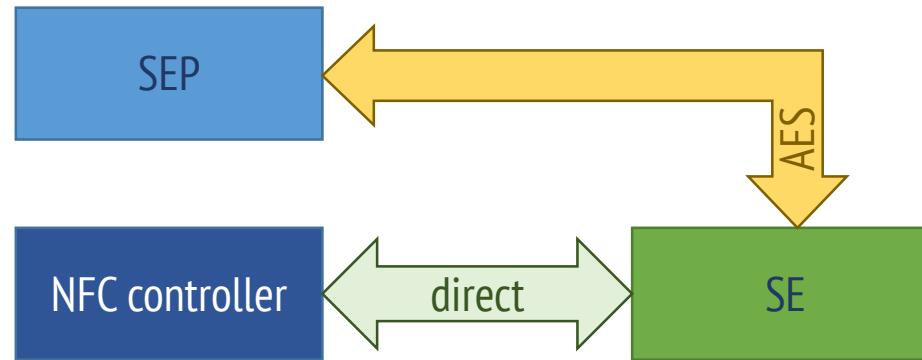
Provisioning a token

Adding a card to Apple Pay:

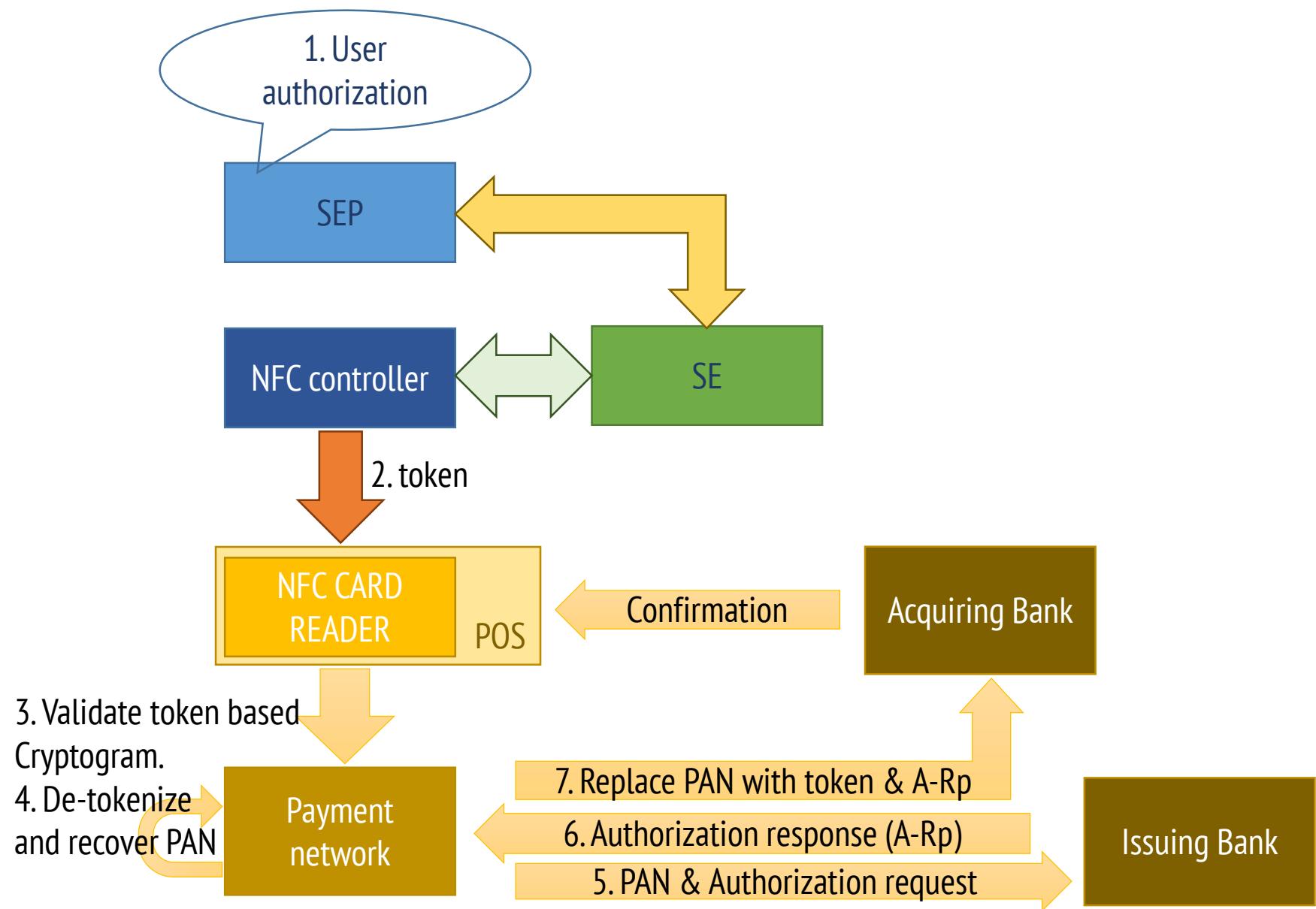
1. Submit PAN details to Apple Pay servers.
2. Apple Pay (i.e. the token requestor) contacts the payment network (i.e. the token service provider) and asks for a token.
3. The payment network contacts the card issuer and performs the card verification.
4. After a successful verification the payment network generates a token and sends it to Apple to provision the SE.

Apple Pay SE and SEP

- The SE communicates with the SEP over a serial interface
- SE and SEP are not directly connected, but have a pre-shared AES key provisioned during the manufacturing process.
- The key is based on the SEP UID and the SE UID
- The SE is tied to an authorization random (AR) generated in SEP.



The payment network



Erasing Cards

- The SE is tied with to an Authorization Random value inside the SEP.
- On receipt of a new Authorization Random the SE marks all previously added cards as deleted
- Cards added in the SE can only be used if the SE is presented with the same Authorization Random that was used during enrolment.
- The SEP can invalidate AR when password is disabled, device restored to default settings, etc.

Apple Pay design principles

- Promote privacy:
 - SE only knows tokens
 - Apple Pay only knows tokens after enrolment
- Assume secrets are not safe
 - SEP can revoke all keys in SE
 - SE can be compromised so only store EMVCo Tokens
- Defend in depth
 - PAN -> EMVCo Token -> SE -> SEP -> user authorization

User Authentication

Based on slides by Ninghui Li (Purdue U)

Readings for This Lecture

- Wikipedia
 - Password
 - Password strength
 - Salt_(cryptography)
 - Password cracking
 - Trusted path
 - One time password



Three A's of Information Security

- Security is about differentiating among authorized accesses and unauthorized accesses
 - Confidentiality, Integrity, Availability all require this
- Authentication
 - Figures out who is accessing
- Access control
 - Ensure only authorized access are allowed
- Auditing
 - Record what is happening, to identify attacks later and recover

Authentication & Access Control according to Wikipedia

- **Authentication** is the act of establishing or confirming something (or someone) as *authentic*, that is, that **claims made by or about the subject are true**. This might involve **confirming the identity of a person**, tracing the origins of an artifact, ensuring that a product is what its packaging and labeling claims to be, or **assuring that a computer program is a trusted one**.
- **Access control** is a system which enables an authority to control access to areas and resources in a given physical facility or computer-based information system.

User Authentication

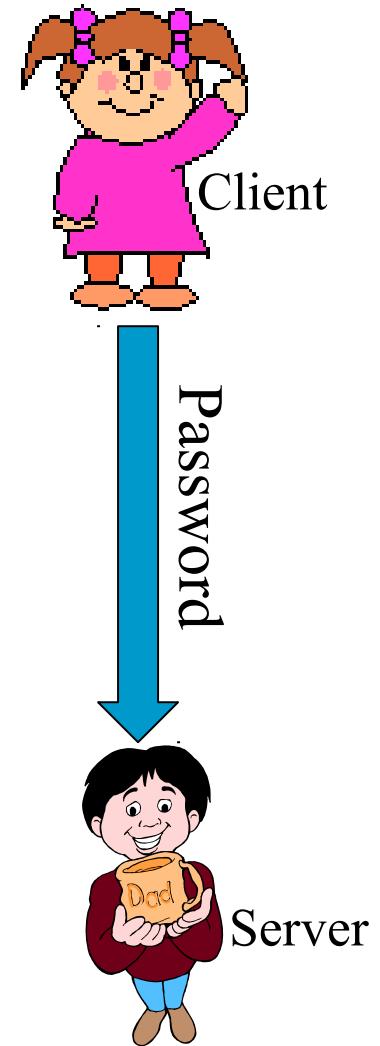
- Using a method to validate users who attempt to access a computer system or resources, to ensure they are authorized
- Types of user authentication
 - Something you know
 - E.g., user account names and passwords
 - Something you have
 - Smart cards or other security tokens
 - Something you are
 - Biometrics

Variants of Passwords

- Password
- Passphrase
 - a sequence of words or other text used for similar purpose as password
- Passcode
- Personal identification number (PIN)

Scenarios Requiring User Authentication

- Scenarios
 - Logging into a local computer
 - Logging into a computer remotely
 - Logging into a network
 - Access web sites
- Vulnerabilities can exist at client side, server side, or communications channel.



Threats to Passwords

- Eavesdropping (insecure channel between client and server)
- Login spoofing (human errors), shoulder surfing, keyloggers
- Offline dictionary attacks
- Social engineering (human errors)
 - e.g., pretexting: creating and using an invented scenario (the pretext) to persuade a target to release information or perform an action and is usually done over the telephone
- Online guessing (weak passwords)

Guessing Attacks: Two Factors for Password Strength

- The average number of guesses the attacker must make to find the correct password
 - determined by how unpredictable the password is, including how long the password is, what set of symbols it is drawn from, and how it is created.
- The ease with which an attacker can check the validity of a guessed password
 - determined by how the password is stored, how the checking is done, and any limitation on trying passwords

Password Entropy

- The entropy bits of a password, i.e., the information entropy of a password, measured in bits, is
 - The base-2 logarithm of the number of guesses needed to find the password with certainty
 - A password with, say, 42 bits of strength calculated in this way would be as strong as a string of 42 bits chosen randomly.
 - Adding one bit of entropy to a password doubles the number of guesses required.
 - On average, an attacker will have to try half the possible passwords before finding the correct one
- Aka. Guess entropy

Estimating Password Entropy

- People are notoriously remiss at achieving sufficient entropy to produce satisfactory passwords.
- NIST **suggests** the following scheme to **estimate** the entropy of human-generated passwords:
 - the entropy of the first character is four bits;
 - the entropy of the next seven characters are two bits per character;
 - the ninth through the twentieth character has 1.5 bits of entropy per character;
 - characters 21 and above have one bit of entropy per character.
- This would imply that an eight-character human-selected password has about 18 bits of entropy.

Towards Better Measurement of Password Entropy

- NIST suggestion fails to consider usage of different category of characters:
 - Lower-case letters, digits, upper-case letters, special symbols
- Orders also matter:
 - “Password123!” should have different entropy from “ao3swPd! 2s1r”

Example of Weak Passwords (from Wikipedia)

- Default passwords (as supplied by the system vendor and meant to be changed at installation time): *password*, *default*, *admin*, *guest*, etc.
- Dictionary words: *chameleon*, *RedSox*, *sandbags*, *bunnyhop!*, *IntenseCrabtree*, etc.
- Words with numbers appended: *password1*, *deer2000*, *john1234*, etc.,
- Words with simple obfuscation: *p@ssw0rd*, *l33th4x0r*, *g0ldf1sh*, etc.
- Doubled words: *crabcrab*, *stopstop*, *treetree*, *passpass*, etc., can be easily tested automatically.

Example of Weak Passwords (from Wikipedia)

- Common sequences from a keyboard row: *qwerty*, 12345, *asdfgh*, *fred*, etc.
- Numeric sequences based on well known numbers such as 911, 314159, or 27182, etc.,
- Identifiers: *jsmith123*, *1/1/1970*, *555–1234*, "your username", etc.,
- Anything personally related to an individual: license plate number, Social Security number, current or past telephone number, student ID, address, birthday, sports team, relative's or pet's names/nicknames/birthdays, etc.,
 - can easily be tested automatically after a simple investigation of person's details.

Mechanisms to Avoid Weak Passwords

- Allow long passphrases
- Randomly generate passwords where appropriate
 - Though probably inappropriate for most scenarios
- Check the quality of user-selected passwords
 - use a number of rules of thumb
 - run dictionary attack tools
- Give user suggestions/guidelines in choosing passwords
 - e.g., think of a sentence and select letters from it, “It’s 12 noon and I am hungry” => “I’S12&IAH”
 - Using both letter, numbers, and special characters

Balancing Password Entropy & Usability Concerns

- Forcing randomly generated passwords is often bad.
 - A user needs to remember passwords for tens, if not hundreds of accounts
 - High entropy passwords are difficult to remember
- Often times, guessing passwords is not the weakest link
 - One can use various ways to reduce adversary's abilities to test password guesses
 - When a user cannot remember the password for an account, there must be a way to allow a user to retrieve it.
 - The recovering method either has low security, or costs lots of money
 - It creates a weaker link.
- **Usability matters**

Storing Passwords (UNIX Case Study)

- Old UNIX
 - The file /etc/passwd stores H(password) together with each user's login name, user id, home directory, login shell, etc.
 - H is essentially a one-way hash function
 - The file /etc/passwd must be world readable
 - Brute force attacks possible even if H is one-way
 - how to most effectively brute-force when trying to obtain password of any account on a system with many accounts?

Password Salts

- More modern UNIX
 - Divide /etc/password into two files: /etc/password; and /etc/shadow (readable only by root)
- Store $[r, H(password, r)]$ rather than $H(password)$ in /etc/shadow
 - r is randomly chosen for each password
 - r is public, similar to Initial Vector in CBC & CTR modes
- Benefits
 - dictionary attacks much more difficult
 - cost of attacking a single account remains the same
 - if two users happen to choose the same password, it doesn't immediately show

Mechanisms to Defend Against Dictionary and Guessing Attacks

- Protect stored passwords (use both cryptography & access control)
- Disable accounts with multiple failed attempts
- Require extra authentication mechanism (e.g., phone, other email account, etc.)

Mechanisms to Defend Against Login Spoofing: Trusted Path

- Login Spoofing Attacks:
 - a program showing a login window on screen and record the passwords
- Defense: Trusted Path
 - Mechanism that provides confidence that the user is communicating with the real intended server
 - attackers can't intercept or modify whatever information is being communicated.
 - defends attacks such as fake login programs
 - Example: Ctrl+Alt+Del for log in on Windows
 - Causes a non-maskable interrupt that can only be intercepted by the operating system, guaranteeing that the login window cannot be spoofed

Spoofing & Defenses on the Web

- Phishing attacks
 - attempting to acquire sensitive information such as usernames, passwords and credit card details by masquerading as a trustworthy entity in electronic communication.
- Website forgery
 - Set up fake websites that look like e-commerce sites and trick users into visiting the sites and entering sensitive info
- Defense methods
 - Browser filtering of known phishing sites
 - Extended Validation Certificates to authenticate servers
 - User-configured authentication of servers
 - Ensures that the site is the one the human user has in mind
 - E.g., site key, pre-selected picture/phrases

KeyLogging

- Threats from insecure client side
- Keystroke logging (keylogging) is the action of tracking (or logging) the keys struck on a keyboard, typically in a covert manner so that the person using the keyboard is unaware that their actions are being monitored.
- Software -based
 - key-stroke events, grab web forms, analyze HTTP packets
- Hardware-based
 - Connector, wireless sniffers, acoustic based
- Defenses:
 - Anti-spyware, network monitors, on-screen soft keyboard, automatic form filler, etc.
- In general difficult to deal with once on the system



Using Passwords Over Insecure Channels

- One-time passwords
 - Each password is used only once
 - Defend against passive adversaries who eavesdrop and later attempt to impersonate
- Challenge response
 - Send a response related to both the password and a challenge
- Zero knowledge proof of knowledge
 - Prove knowledge of a secret value, without leaking any info about the secret

How to do One-Time Password

- Shared lists of one-time passwords
- Time-synchronized OTP
 - E.g., use $\text{MAC}_K(t)$, where K is shared secret, and t is current time
- Using a hash chain (Lamport)
 - $h(s), h(h(s)), h(h(h(s))), \dots, h^{1000}(s)$
 - use these values as passwords in reverse order



Challenge-Response Protocols

- Goal: one entity authenticates to other entity proving the knowledge of a secret, ‘challenge’
- Approach: Use time-variant parameters to prevent replay, interleaving attacks, provide uniqueness and timeliness
 - e.g., nonce (used only once), timestamps

Other Defenses

- Alternatives to passwords
 - graphical passwords
- Go beyond passwords
 - security tokens
 - biometrics
 - 2-factor authentication
 - Uses two independent authentication methods
 - US Banks are required to use 2-factor authentication by end of 2006 for online banking
 - Out of band authentication: uses a channel other than the internet
 - E.g., phone

Open Problems

- Alternatives to passwords?
 - The secret should be easy to remember, difficult to guess, and easy to enter into the system.
- Better ways to make user choose stronger passwords?
- Better ways to use other devices for authentication
- Effective 2-factored and/or out of band authentication for the Web
- Phishing defense