Presentazione di fine terzo anno

Enrico Bacis
Advisor: **Prof. Stefano Paraboschi**

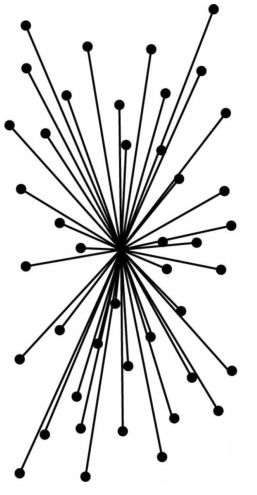
Ingegneria e Scienze Applicate (XXXII° ciclo)
Università degli Studi di Bergamo

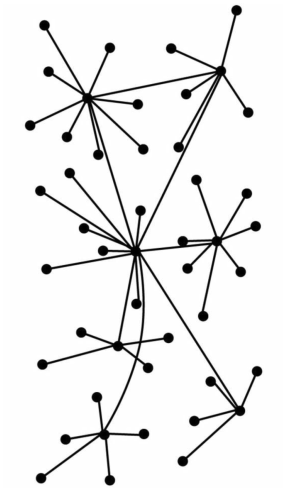
Agenda

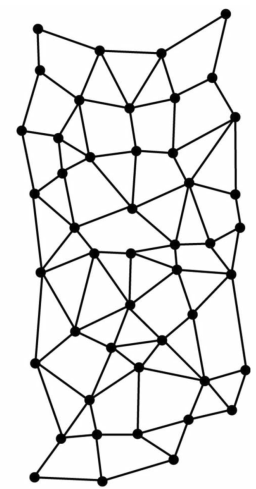
- **Research**
  - Mix&Slice: Efficient Access Revocation in the Cloud
  - Securing Resources in Decentralized Cloud Storage
  - I *Told* You Tomorrow: Practical Time-Locked Secrets using Smart Contracts

- **Participation to European Projects**
- **Publications**
- **Other Activities**

Protecting Resources and Regulating Access in Centralized and Decentralized Cloud Systems







Mix&Slice: Efficient Access Revocation in the Cloud

(Centralized Systems)

Scenario

- **Encryption** is increasingly used to protect stored data from both other users and cloud providers.
- In the presence of multiple users, the **access policy** maps to the ability of each authorized user to **get/derive the encryption key** of a resource
- **Problem**: **enforcement of authorization revocation and protection in case of key leakage**

Centralized Cloud Service Providers are considered **honest-but-curious**

(they comply with users' request
but might access data if unprotected)

Alternatives to manage user revocation

- **The key becomes unavailable to the revoked user**
  - Efficient, but the user may have kept a local copy of the encryption key
- **The resource is re-encrypted with a fresh key**
  - Inefficient, because the resources have to be re-encrypted
  - It is especially inefficient in a cloud environment
- **The resource is protected with an additional encryption layer (over-encryption)**
  - Requires support by the server

Mix&Slice Solution

Encryption structure that makes the decryption dependent on all the pieces

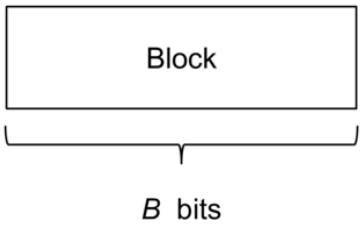
- **policy update**: update the encrypted representation of a randomly chosen piece
- ***n* pieces**: the resource becomes unavailable by updating 1/n-th of the resource

Advantages

- Protection against revoked users keeping their keys
- Efficient (arbitrarily more efficient than resource re-encryption)
- No need of dedicated support by the storage server

Basic Concepts

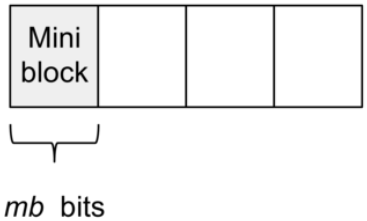
Block: the unit of work for a symmetric block cipher



Basic Concepts

Block: the unit of work for a symmetric block cipher

Mini-block: portion of a block that defines the atomic unit of information

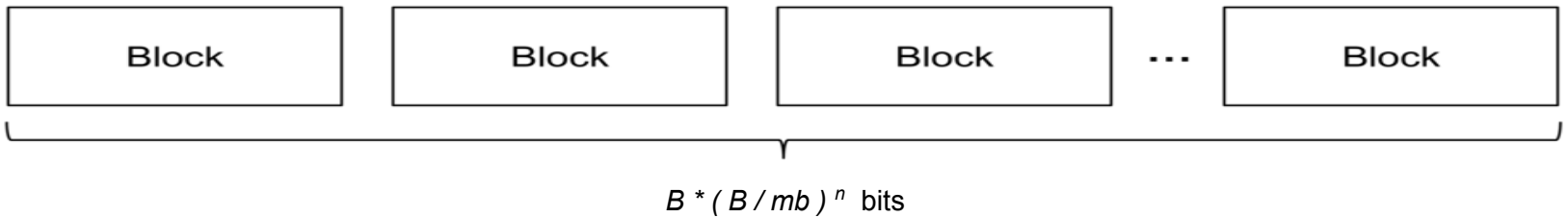


Basic Concepts

Block: the unit of work for a symmetric block cipher

Mini-block: portion of a block that defines the atomic unit of information

Macro-block: it allows extending the mixing of the block cipher to longer sequences



Basic Concepts

AES is today the most common block cipher. Using AES, the protection provided is:

- Block: ***128*** bits
- Mini-block: ***32*** bits (4 mini-blocks per block)
- Macro-block: $32 * (128/32)^i$ bits (with $i \geq 1$) - e.g., *128* bits, *512* bits, *2048* bits, …

With AES the maximum mini-block size is 64 bits, which is still *brute-forceable*.

It is possible to integrate AES with OAEP to overcome this limitation.

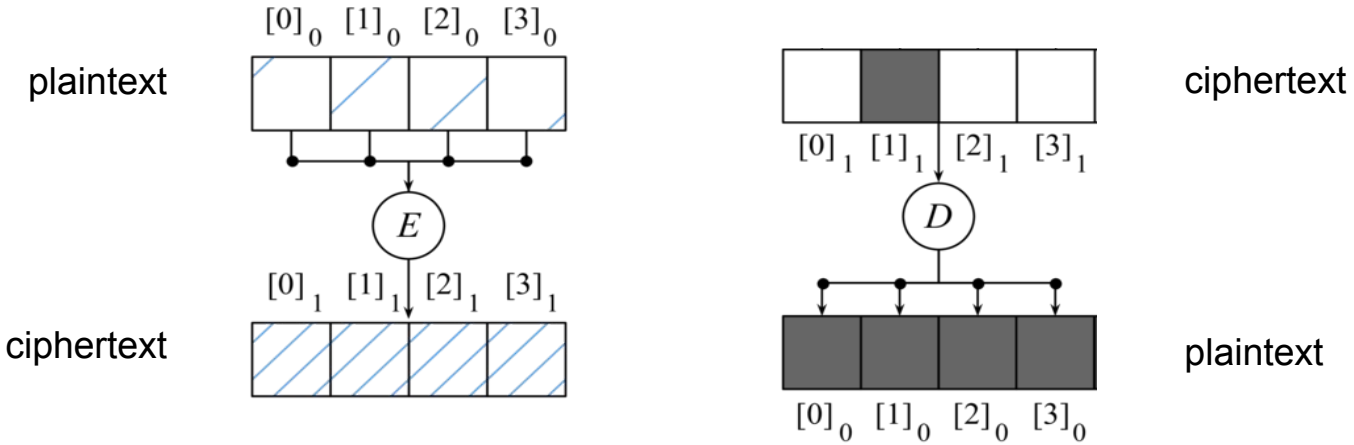
Mix&Slice

Mix&Slice partitions the resource in distinct, equally sized macro-blocks.

Then, as the name suggests, it performs two processes:

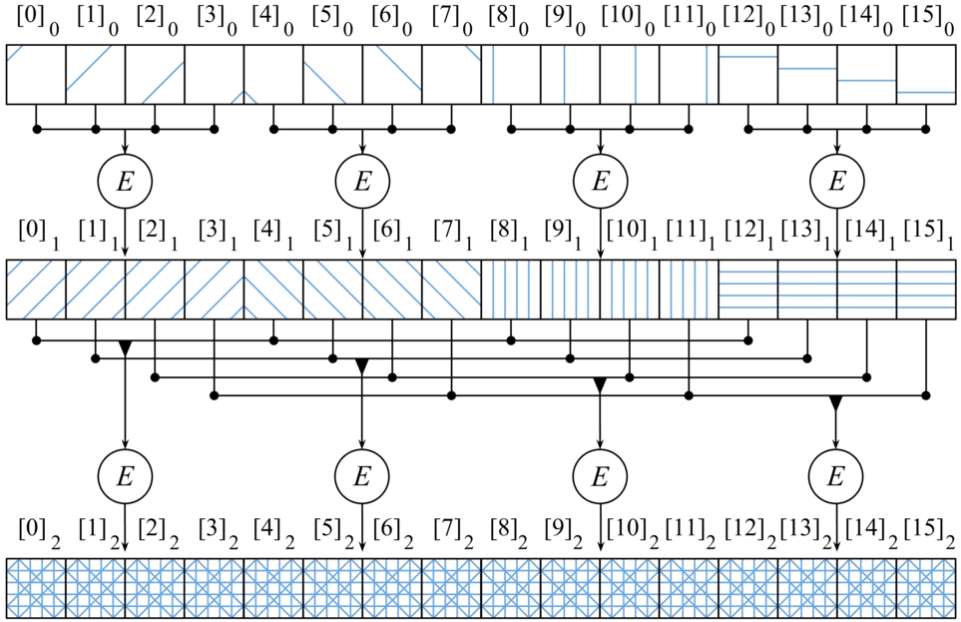
- **Mixing**
- **Slicing**

Mixing



A block cipher guarantees **complete inter-dependency** of the encrypted result.

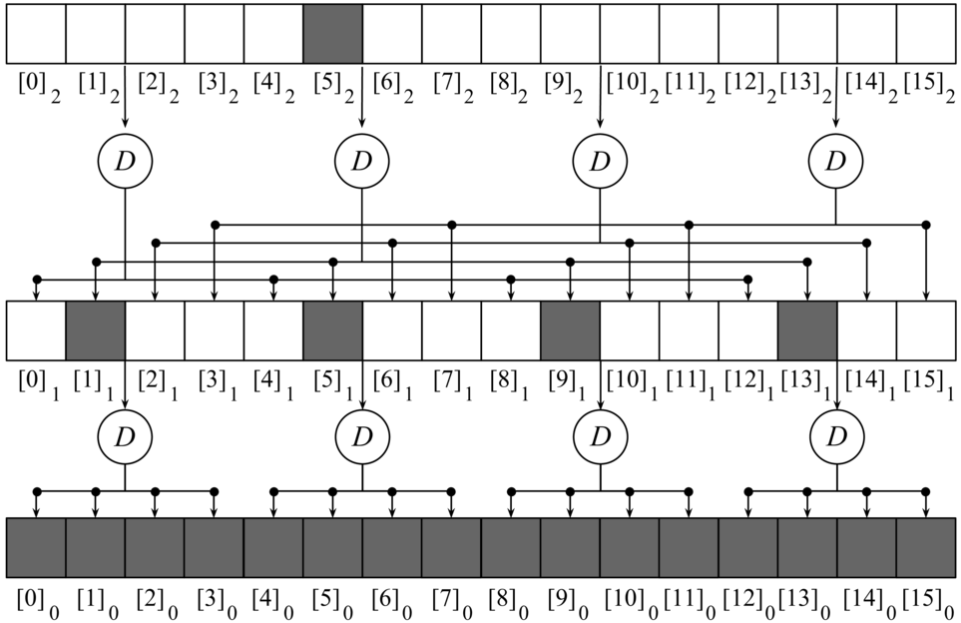
This guarantee is not provided when the data to encrypt is larger than a block.



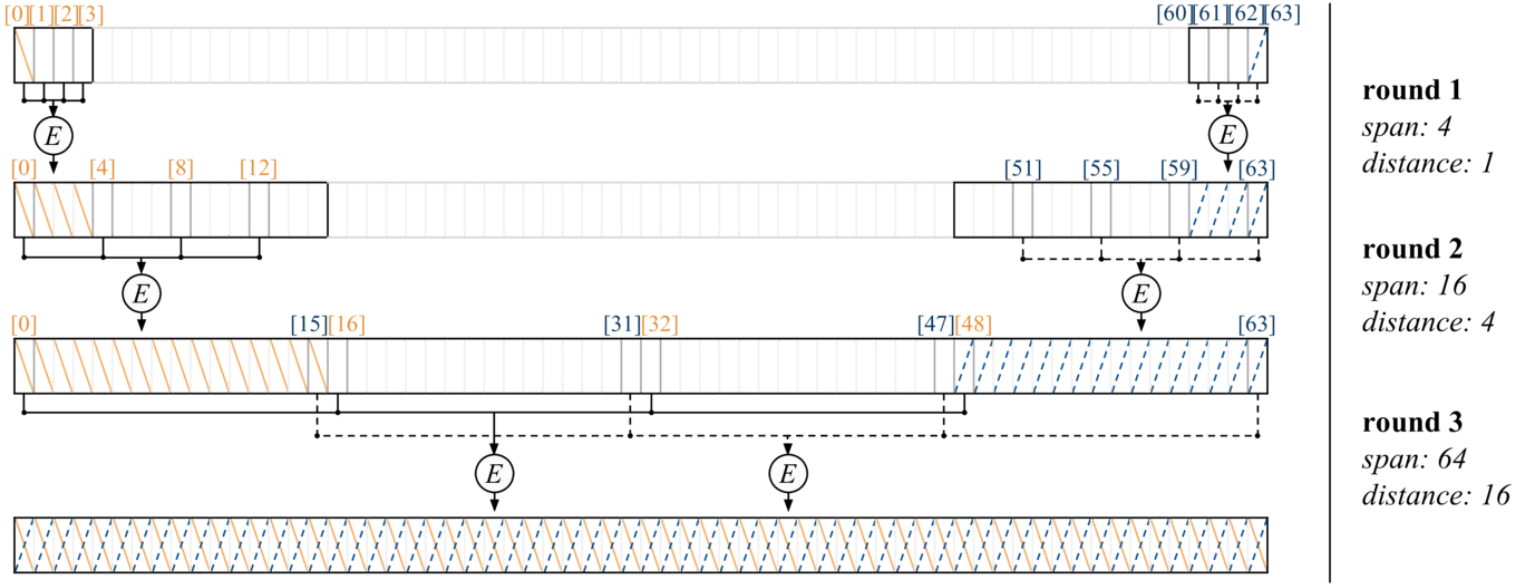
plaintext

ciphertext

The content of each macro-block is processed by an **iterative application of multiple encryption**. This ensures that each input mini-block has impact on each output mini-block.



If a mini-block in the encrypted representation is missing,
the encryption **cannot be inverted**, even with knowledge of the key.



The approach can manage arbitrarily large macro-blocks.

Comparison with other AONT

- This property is known as **All-or-Nothing Transform** (**AONT**)
- Other AONT techniques derive a key from the encrypted representation.
    - **Storing this key permits to bypass the protection.**

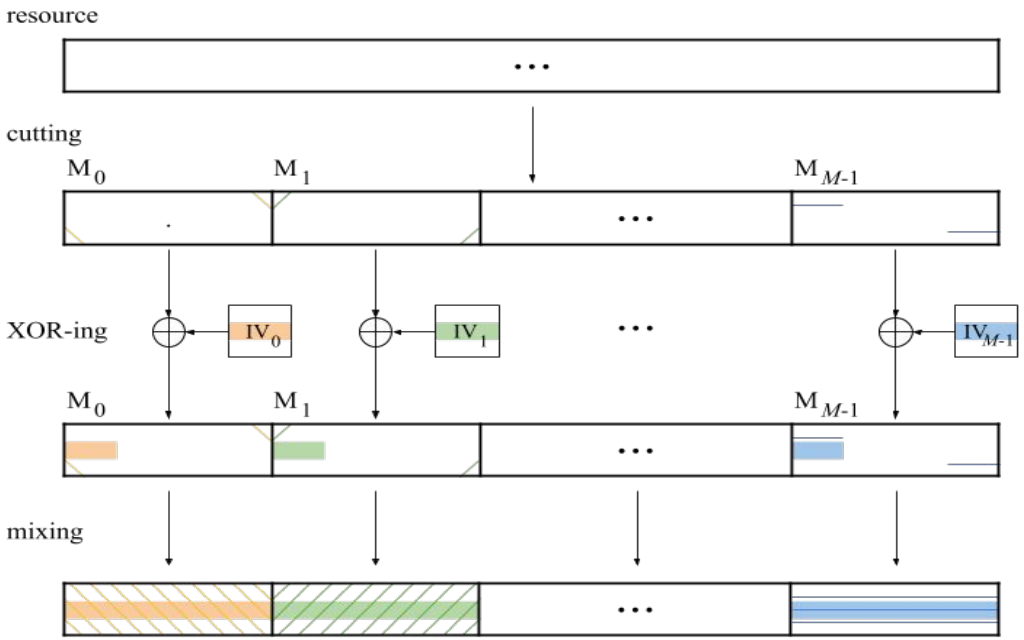
- Mix&Slice: a revoked user is not able to bypass the protection with a c**omputational and local storage cost significantly lower** than the cost of storing the resource itself.

Slicing

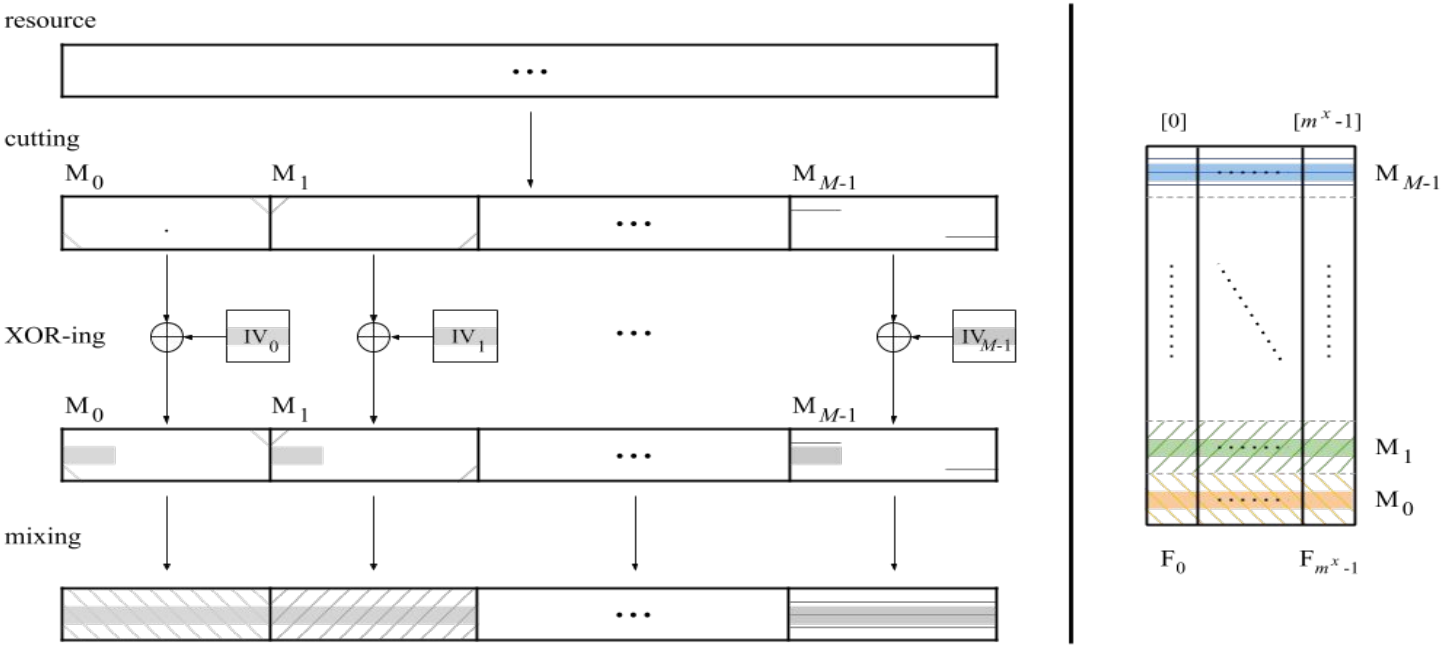
Large macro-blocks introduce shortcomings such as:

- reduction in **decryption efficiency**
- **data transfer overhead** when only a small portion of the macro-block is needed
- **increased latency**

For these reasons, after the **Mix** phase, we introduced the **Slice** phase.



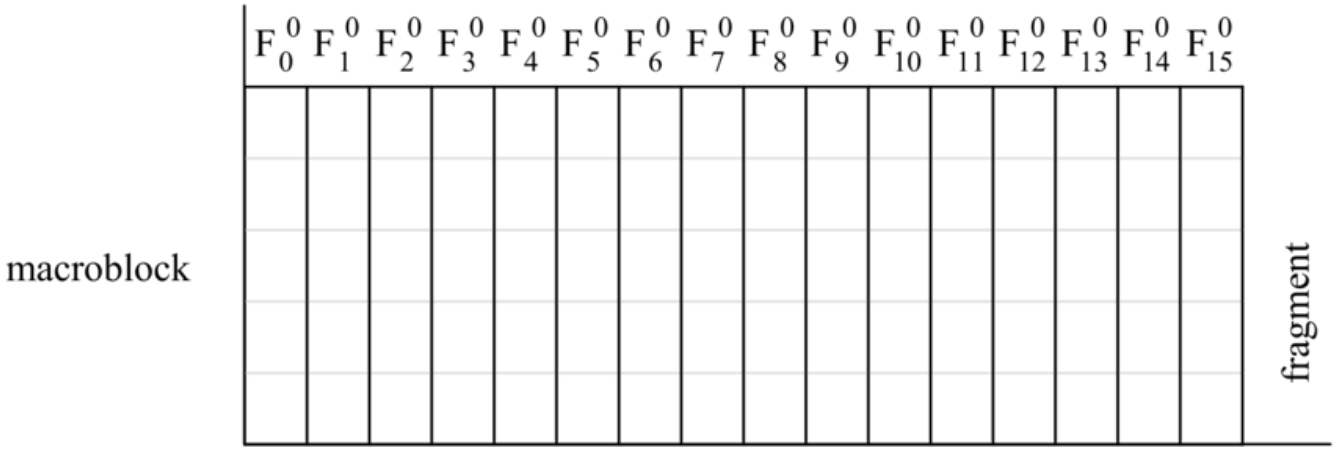
First, the resource is split into a set of macroblocks, on each of them the Mix phase is applied.



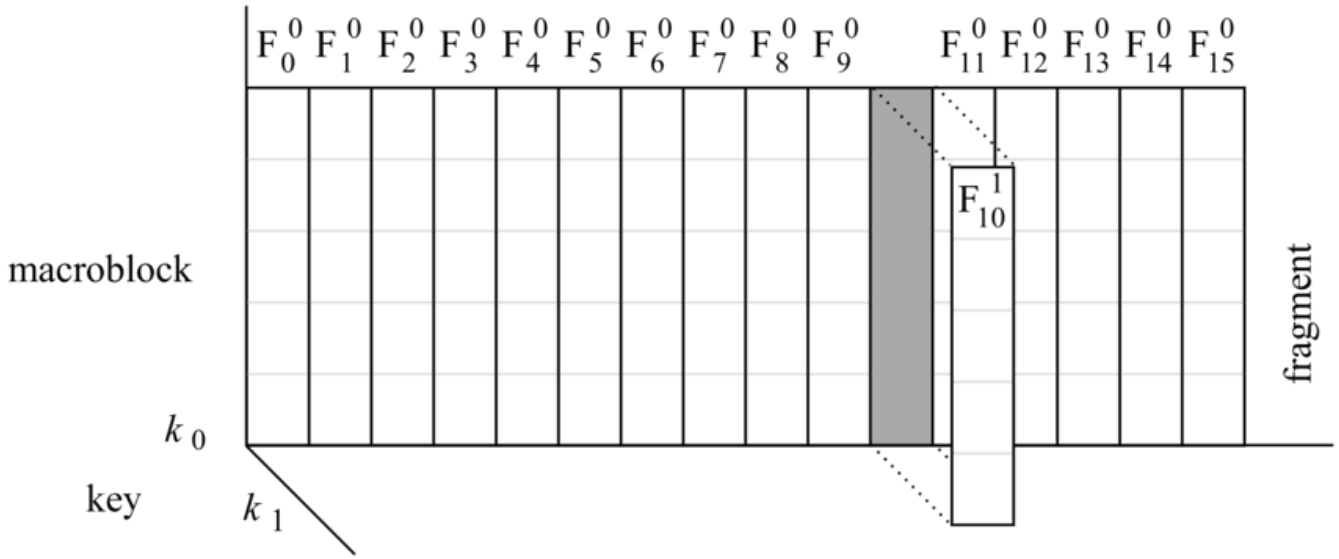
Mini-blocks in the same position are **sliced** into **fragments**, all needed to reconstruct the resource.

Policy Update

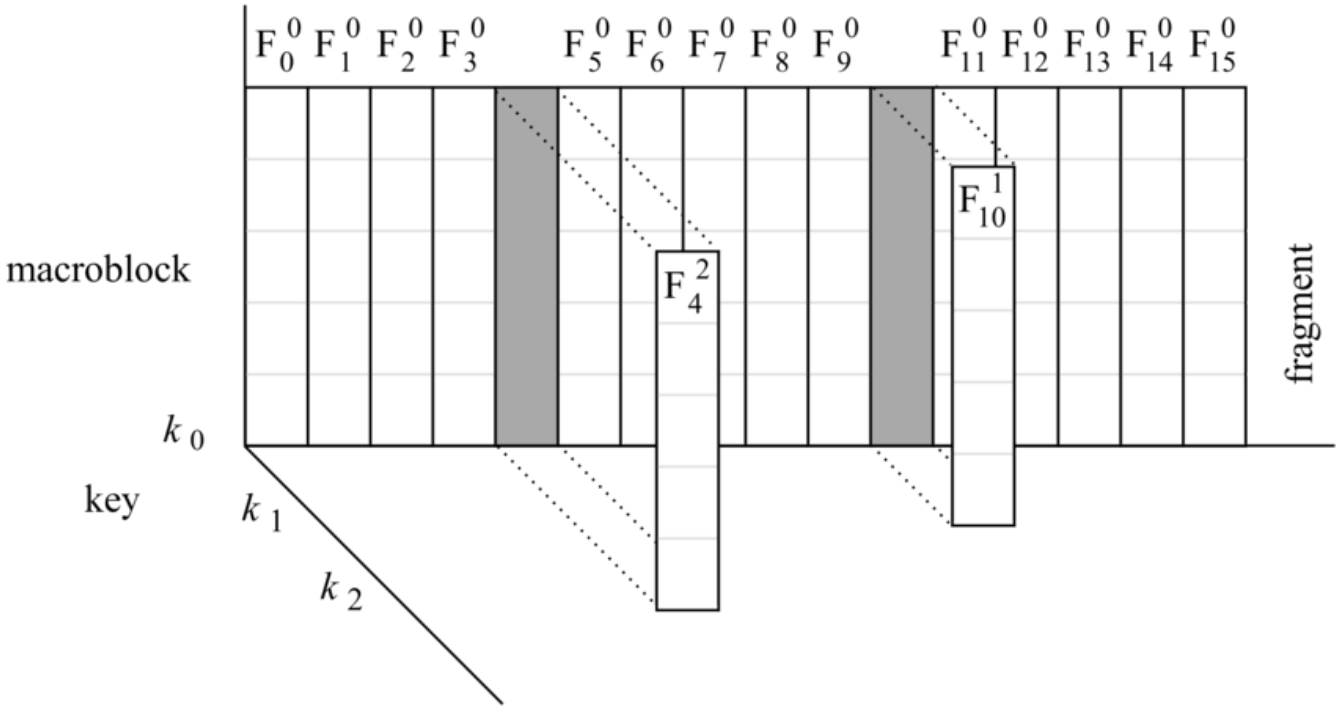
- When a user is revoked access, only a **randomly chosen fragment** needs to be **re-encrypted** with a fresh key k_i.
- Only authorized users will be able to access $\boldsymbol{k_i}$.
- Using a key regression technique based on RSA, all keys $\boldsymbol{k_i}$ can be derived from $\boldsymbol{k_j}$, $\boldsymbol{\forall\, i < j}$, with no need of additional storage.



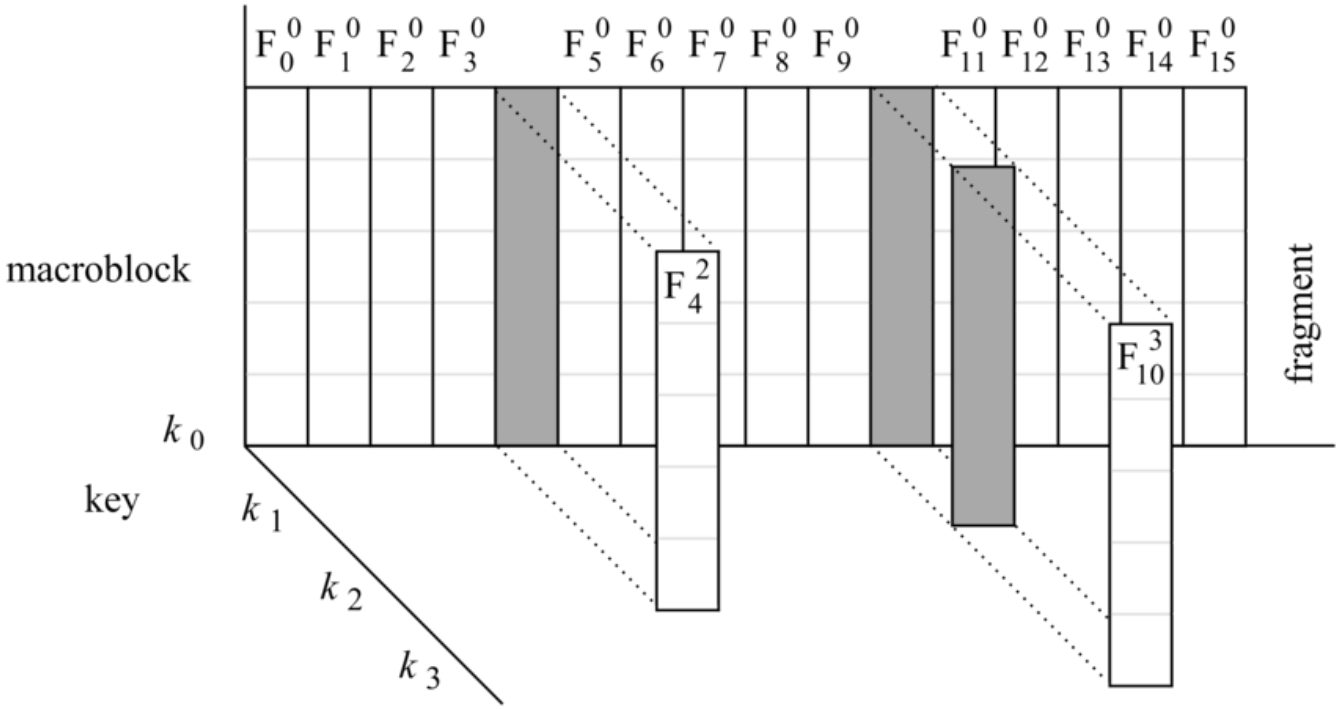
Initial state: base fragments are produced by the application of Mix&Slice



Policy update: one random fragment is re-encrypted with a fresh key k_1



Policy update: one random fragment is re-encrypted with a fresh key k_2



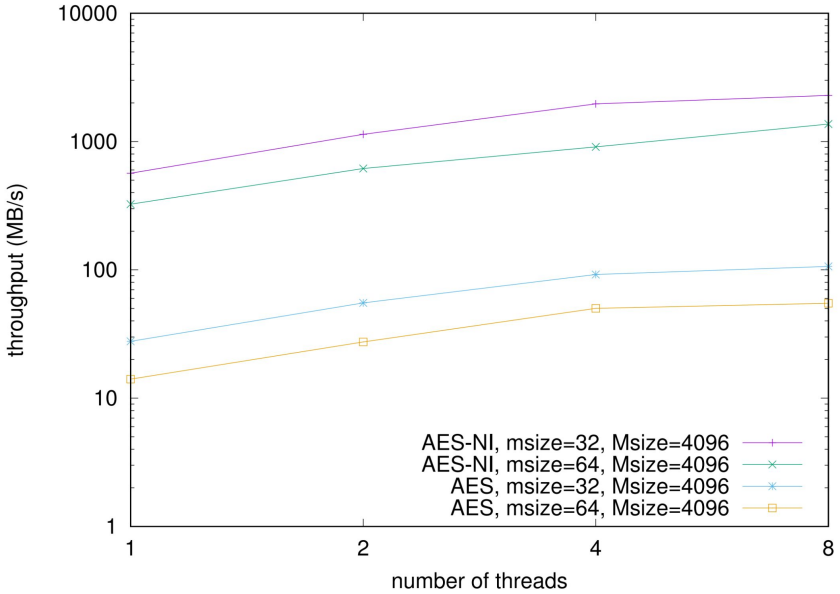
Policy update: one random fragment is re-encrypted with a fresh key k_3

Security Profile

- The revoked user can copy the (*decrypted*) resource to another location (*external*), when she is still allowed
  - Impossible to mitigate
  - But, resources can be extremely large
    - (e.g., multimedia collections)
- **This strategy is associated with high storage and transfer costs**

Implementation

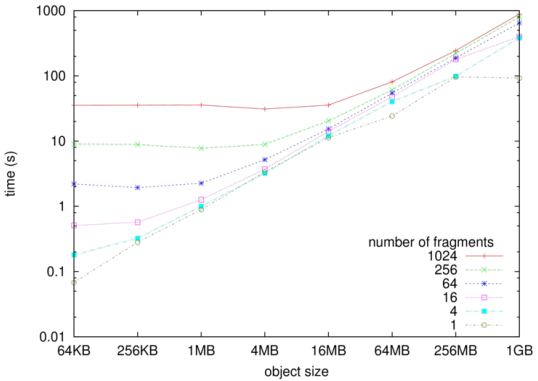
The client implementation has been written in C and is able to leverage hardware implementation of AES (**AES-NI**), with excellent performance (up to **2.5GB/s** on multi-threaded environments).

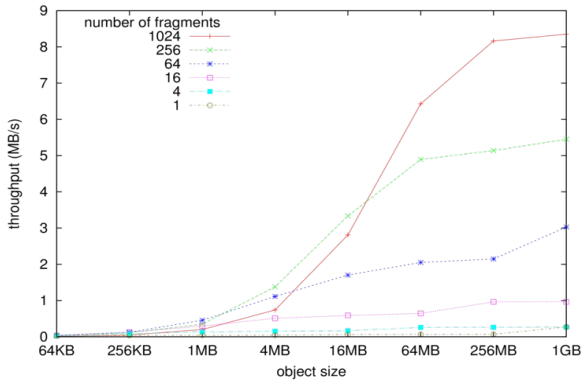


https://github.com/unibg-seclab/aesmix

Implementation

Experiments on OpenStack Swift: the number of fragments has to be selected based on the resource size, as the **fragmentation overhead is significant for small resources**.





Conclusions

- Mix&Slice **efficiently enforces** access revocation on encrypted resources stored at external providers even for users locally maintaining copies of previously-used keys.

- The implementation and experimental evaluation confirm the **effectiveness** of the proposal, and confirms its compatibility with currently available **honest-but-curious cloud storage**.

Publications

E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati, “***Mix&Slice: Efficient Access Revocation in the Cloud***”. in *Proc. of the 23rd ACM Conference on Computer and Communication Security (CCS)*. 2016

E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, A. Sajjad, P. Samarati, M. Bjorkqvist, C. Cachin, A. Taha, P. Metzler, S. Mazoor, N. Suri. “***D2.6 – Final Report on Data Protection and Key Management Solutions”***. Horizon 2020 project deliverable, ESCUDO-CLOUD.

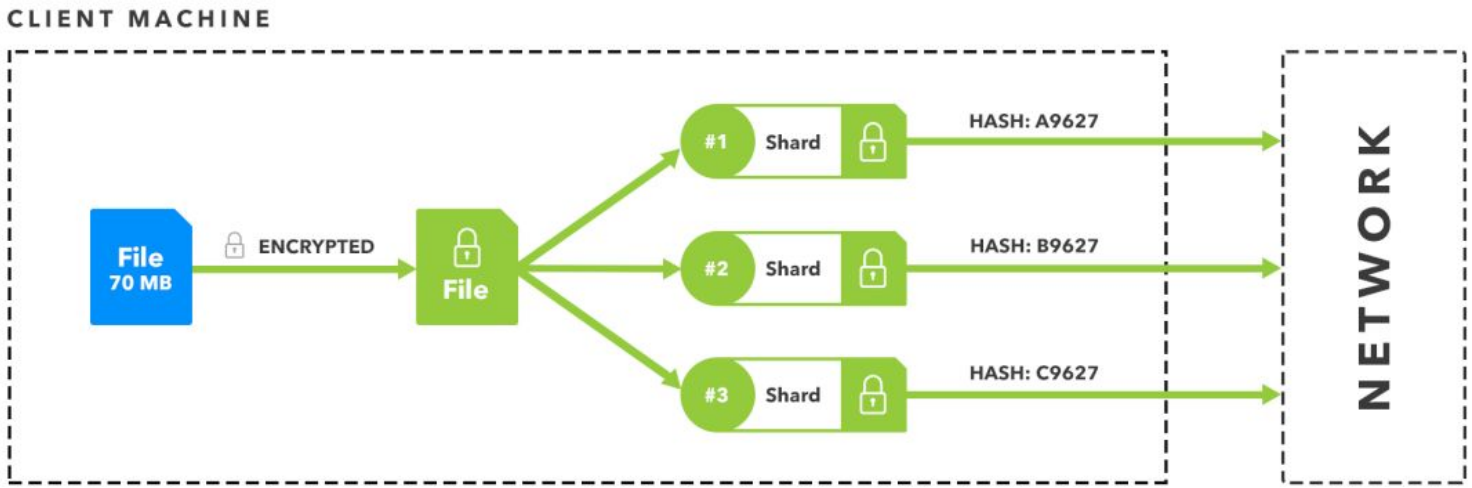
Securing Resources in Decentralized Cloud Storage

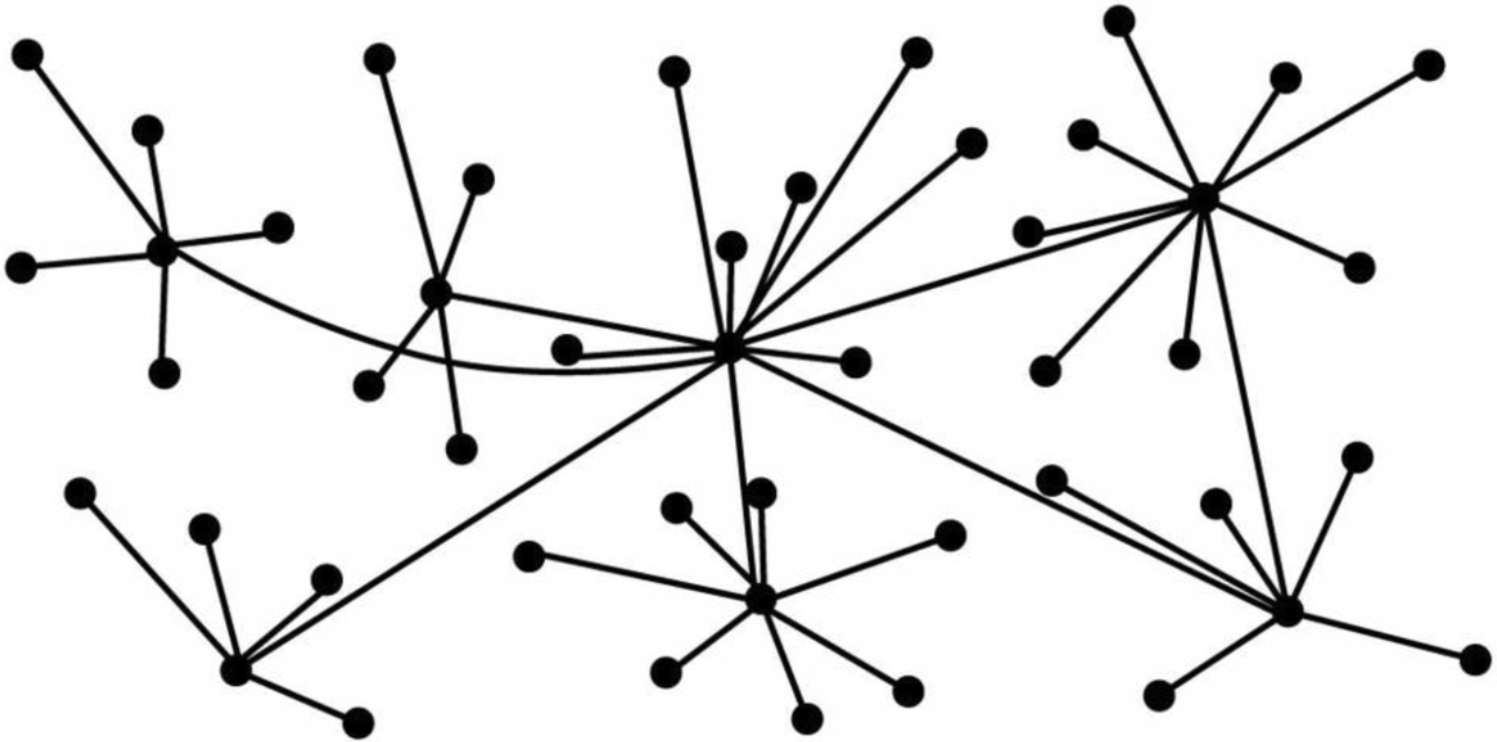
(Decentralized Systems)

Decentralized Cloud Storage

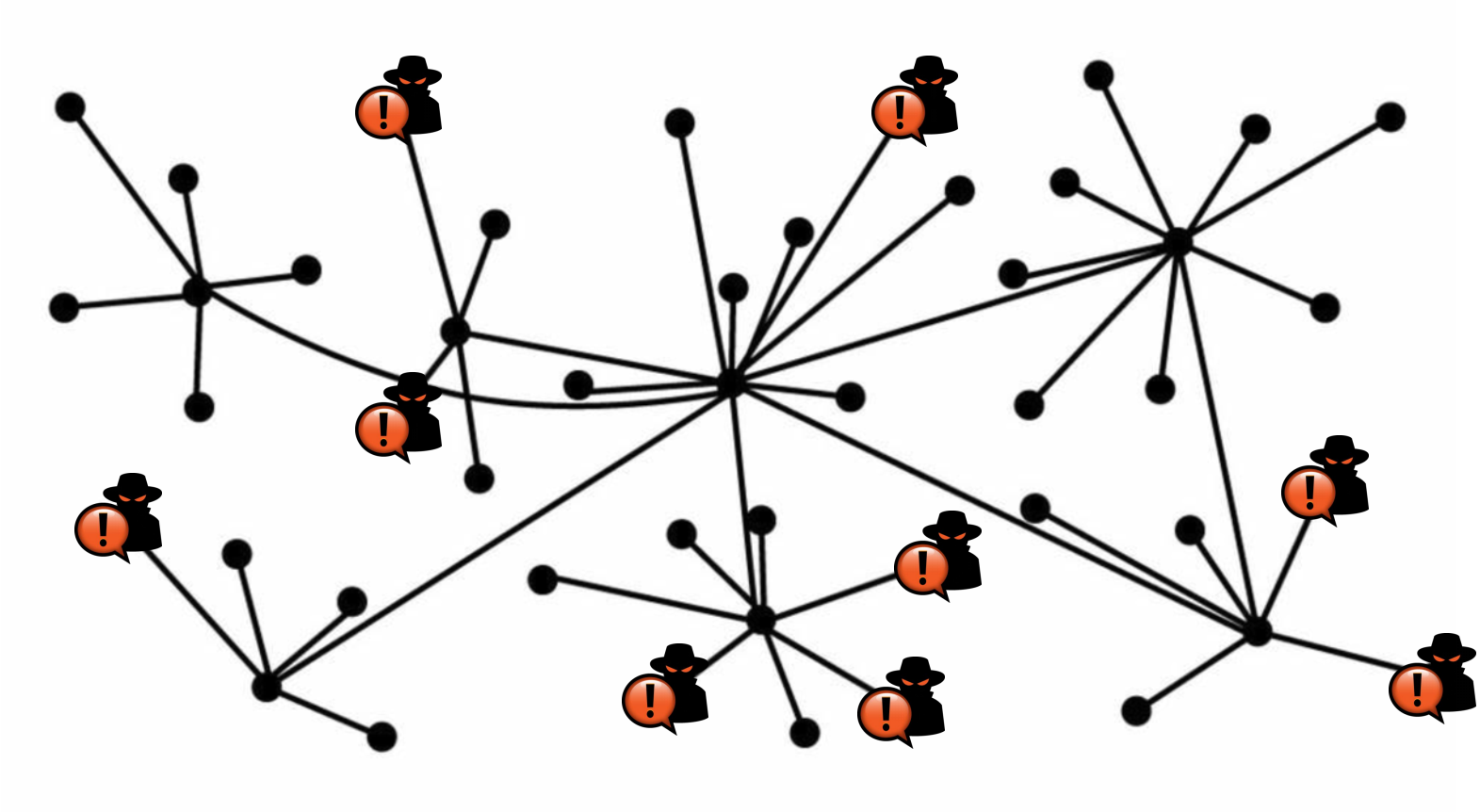
- A resource is encrypted and split into **shards**
- Each shard is stored on a different **node** (user of the network)
- Users get paid (**cryptocurrency**) for their storage space
- **Coordinator nodes** verify that the files are online, and manage availability

Sharding Process



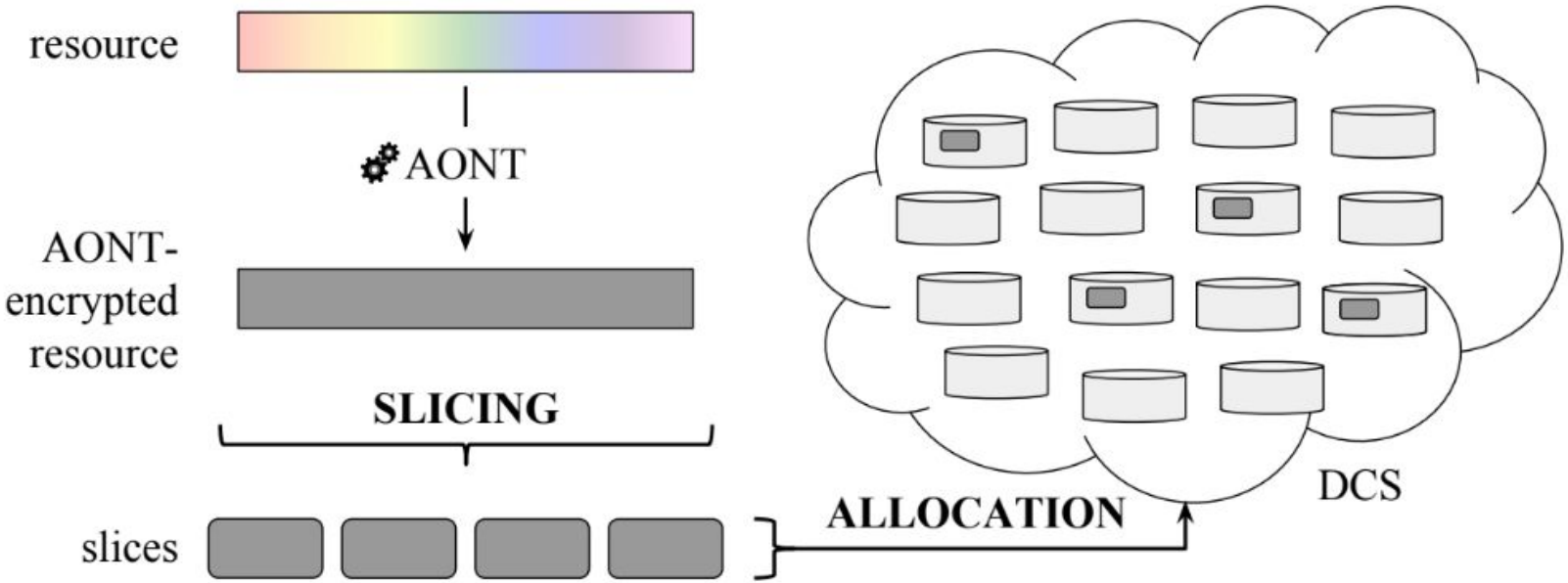


Decentralized Cloud Storage: still **honest-but-curious ?**



Challenges and Opportunities

- Node **failures** (downtimes) are order of magnitude higher than centralized CSPs
- There is no single **Policy Decision Point** and **Policy Enforcement Point**
- How to integrate **access revocation** in such a scenario where a percentage of nodes might **disobey** to perform **revocations** to maximize their revenue?



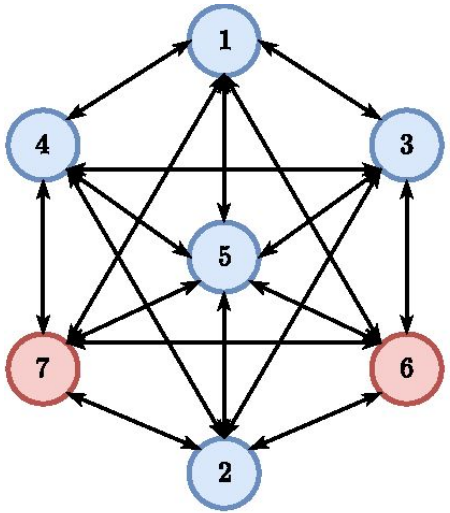
Reference Scenario: AONT provides guarantees for secure deletion:
1 missing fragment makes the resource inaccessible

Our Proposal

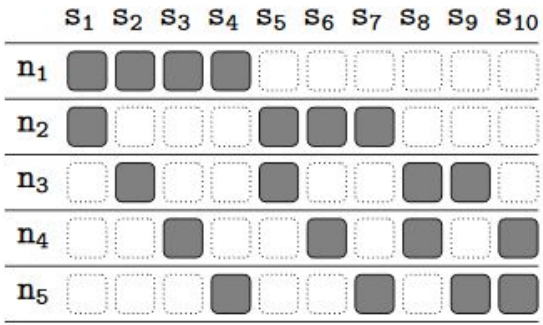
1. Describe the ***network*** as:
   - Node fault probability, p_u
   - Node compromisation probability (*coalition*), p_c

2. Describe the ***requirements*** as:
   - Maximum overall fault probability, P_u
   - Maximum overall compromisation probability, P_c

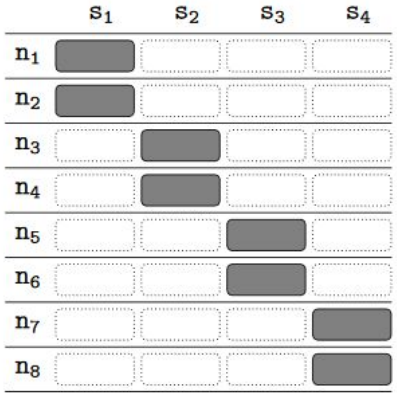
3. Find a suitable ***allocation*** of slices to nodes



Allocation Strategies



Min_nodes: minimizes the overall number of nodes

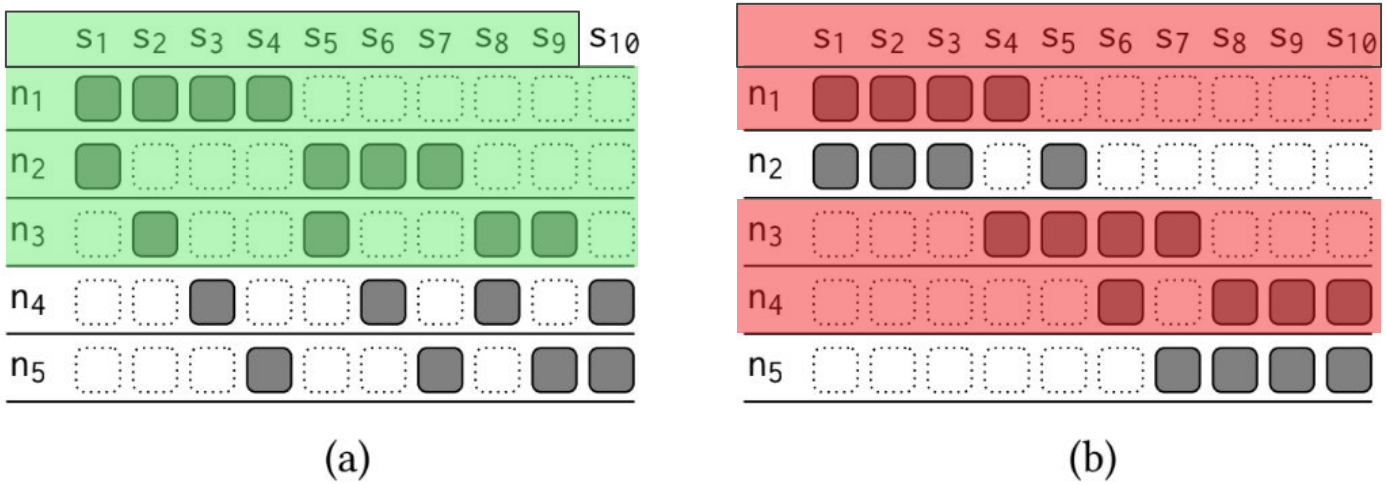


Min_slices: minimizes the overall number of slices

Allocation Properties

- **Replication (*r*)**
  There are at least ***r*** replicas of each slice in the network.

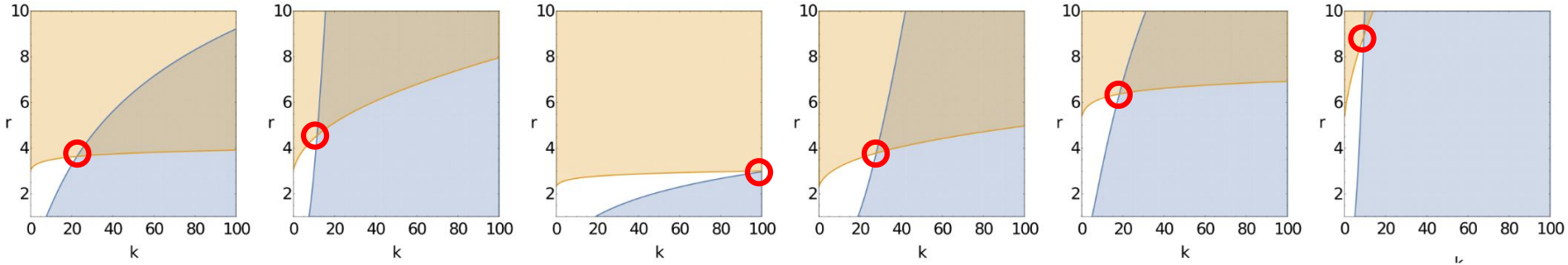
- **Protection (*k*)**
  A minimum of ***k+1*** nodes need to be involved when downloading a resource.



An example of two **2-replicated** allocation functions,

(a) is **3-protected**, (b) is only **2-protected**

Trading-off availability and security guarantees

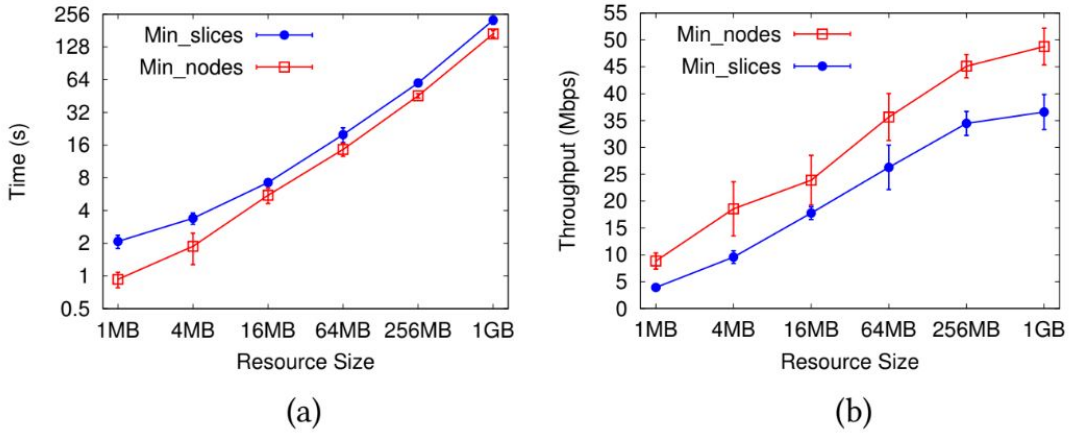


(a) *Min_slices*, p_u=0.005, p_c=0.2 (b) *Min_nodes*, p_u=0.005, p_c=0.2 (c) *Min_slices*, p_u=0.001, p_c=0.5 (d) *Min_nodes*, p_u=0.001, p_c=0.5 (e) *Min_slices*, p_u=0.05, p_c=0.1 (f) *Min_nodes*, p_u=0.05, p_c=0.1

Min_slices and *Min_nodes* (*k*, *r*)-allocations that guarantee $P_u \leq 10^{-7}$ and $P_c \leq 10^{-6}$ with different values for p_u and p_c

	IPFS	Decent	MaidSafe	Storj	SiaCoin	BlockCDN
Compensation Model	Data blocks bartered with reciprocated file sharing (Bitswap)	Per purchase (essentially per download) and per storage space-time	Payment per storage space, CPU, bandwidth, and online time	The platform currently pays storage providers monthly and bills storage renters monthly	Determined by a file contract between a storage renter and a provider	Per bandwidth use
Who Provides Payment?	File downloaders	Content consumers and content creators	Users and generated tokens	Storage renter (via Storj)	Storage renter	Content providers--those selling content to users
Blockchain Foundation	None, uses bitswap credit protocol	Independent DECENT blockchain	None: uses close groups consensus	Counterparty bitcoin blockchain	Independent Sia blockchain	BCDN tokens on Ethereum network
Target Use Cases and Scenarios	File hosting, versioning, web hosting, content distribution, etc	Media streaming	Encrypted cloud storage, web hosting, streaming	Encrypted cloud storage	Encrypted cloud storage	Optimize content delivery speed for centralized or decentralized content

Storj Implementation



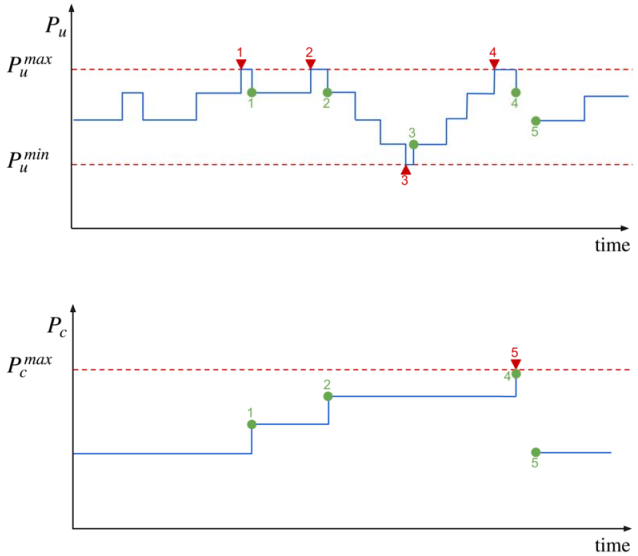
Completion time (a) and overall throughput (b) in the Min_slices and Min_nodes allocation strategies

Leveraging the dynamicity of DCS

Some DCS are currently migrating from replication to **erasure codes** (*k-of-n* shards needed to access the resource)

In DCS networks, a node may be only **temporarily offline**, by taking this aspect into account, we can **limit** the number of re-uploads.

Will be discussed in details by Marco Rosa



Conclusions

- Decentralized cloud storage providers can **not** be considered **honest**.
- AONT (such as Mix&Slice) provides guarantees for deletion/revocation.
- We identified two **allocation properties**: *replication* and *protection*.
- We implemented two **allocation strategies** (*Min_nodes* and *Min_slices*)
    - We evaluated their performance in Storj.

Publications

- E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati. ***"Securing Resources in Decentralized Cloud Storage"***. In IEEE Transactions on Information Forensics and Security (TIFS), Vol. 15, n.1. IEEE, 2019.
- E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati, ***"Dynamic Allocation for Resource Protection in Decentralized Cloud Storage"***. In *Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019.

I *Told* You Tomorrow: Practical Time-Locked Secrets using Smart Contracts

(Distributed Systems)

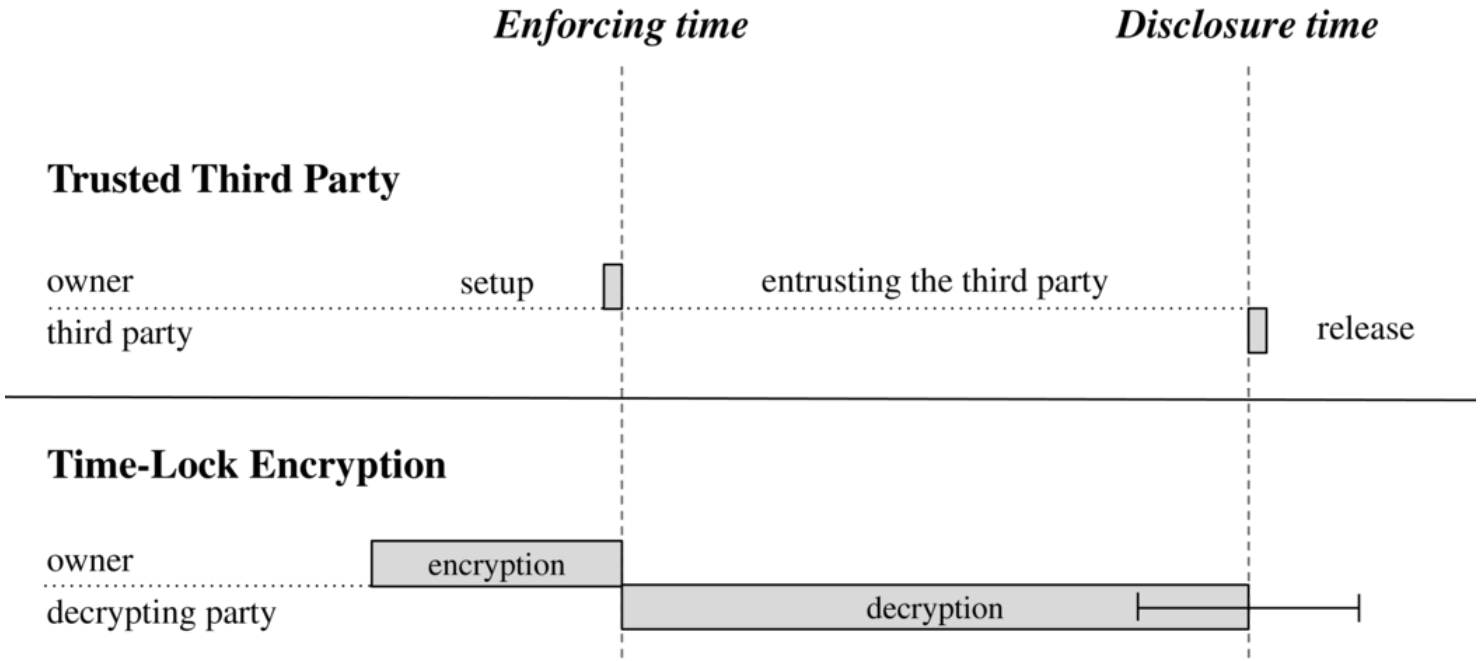
The previous solution work as long as there is a **somewhat trusted party** that checks that the slices are online and reassign them otherwise.

How can we achieve full distribution without the need of the data owner being online or relying on a trusted party?

Idea *(Delegated Challenge-Response)*

- The data owner prepares a **series of challenges and responses** when still online (time ***t***) and offload them to the distributed cloud provider, along with the data.
- At time ***t + d1***, a **challenge is revealed**. The storage node has to read the challenge data and generate a response to prove that she is still storing the data.
- At time ***t + d2***, the **response is revealed** and compared with the node's response. If they differ, the node is discarded and new nodes can step in as replacement.

We need a way to automatically reveal data at a future time

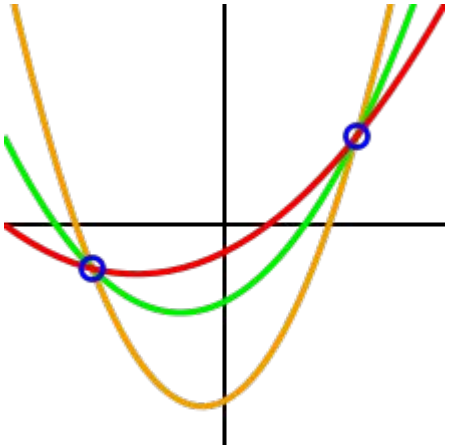


April 29 '19

Programmers solve MIT's 20-year-old cryptographic puzzle

Threshold Cryptography

Threshold cryptography permits to split a ***secret*** into parts, giving one to each of the ***n*** participants. To reconstruct the original secret, a number of parts ***k (<n)*** is required.



Shamir's Secret Sharing

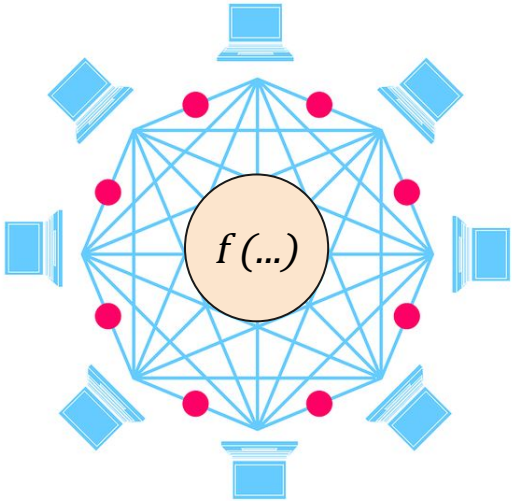
- Take a random ***k-1*** degree polynomial with ***f(0) = secret***
- Sample ***n*** points to get the ***n*** shares.
    - 2 points are sufficient to define a line,
      3 points define a parabola, etc.

Secure Multi-party Computation

sMPC is a cryptographic protocol that allows multiple parties to jointly **compute a function** over their **inputs** while keeping them **private**.

ITYT uses it to generate the Shamir shares:

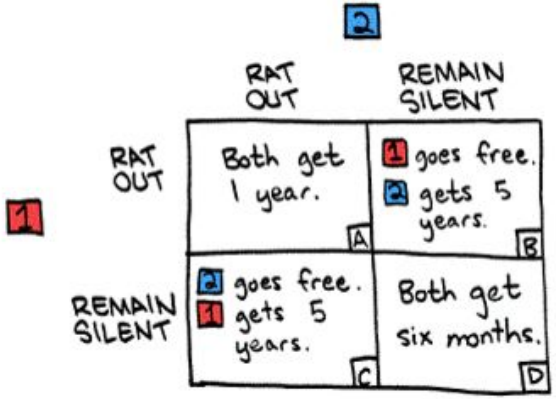
- Each participant is the **only one** to see her **share**
- The **data owner** gets their **commitments** (*hash*)



Game Theory

As we use threshold cryptography we already have strong guarantees that **if no *k* participants collude, the secret remains protected**.

Even if *k* participants would like to collude, by using game theory, we define economic parameters so that **rational participants** will always behave **honestly,** as the honest behavior is a *Nash equilibrium* and any other choice is economically disadvantageous.

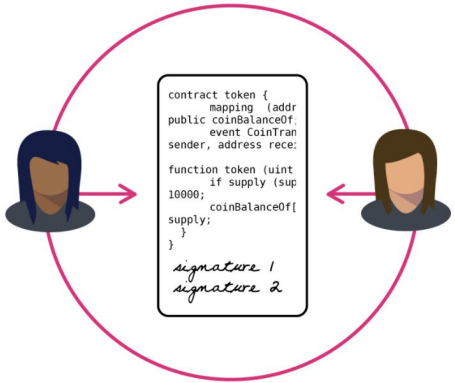


Blockchains and Smart Contracts

ITYT leverages blockchains as they offer:

- a distributed tamper-proof **timestamp**;
- a **permanent record** of transactions and their associated data.

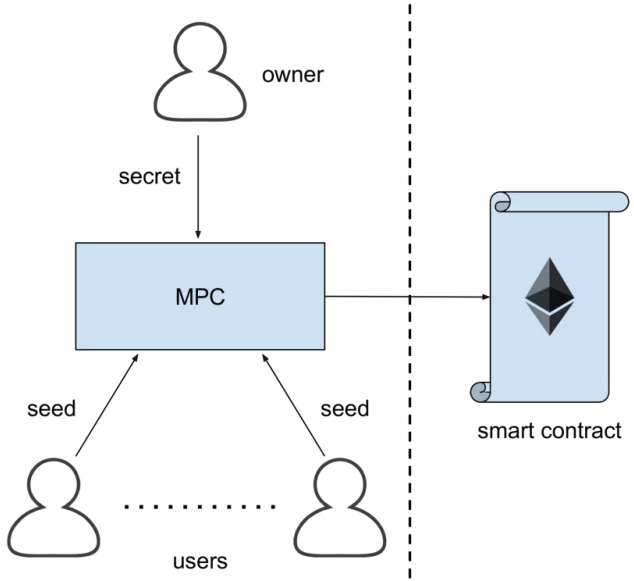
Smart contract permit to code programs that executes in a publicly verified way on the blockchain, **without the need of any trusted third party**.



I Told You Tomorrow

a secret-keeping game played on the blockchain

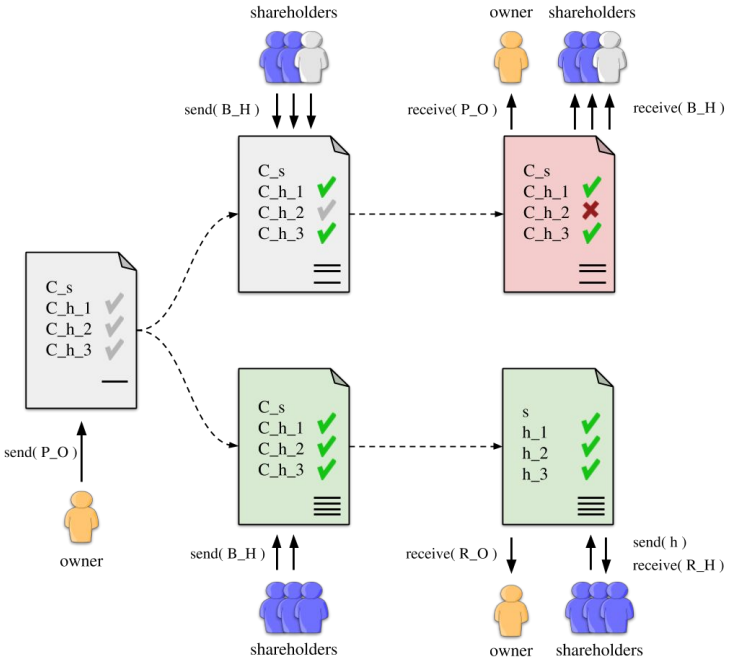
- Participants receive a **share** of the secret in exchange for **depositing a bid**
- Participants receive a **reward** (pre-deposited by the data owner) if they disclose their share only after the **disclosure time**
- Participants **lose** their bid if they disclose their share ***before*** or ***collude*** to retrieve the secret.



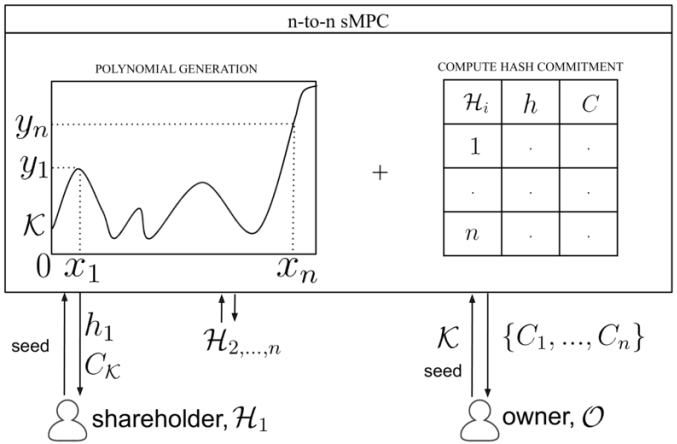
Reporting Misbehaviors

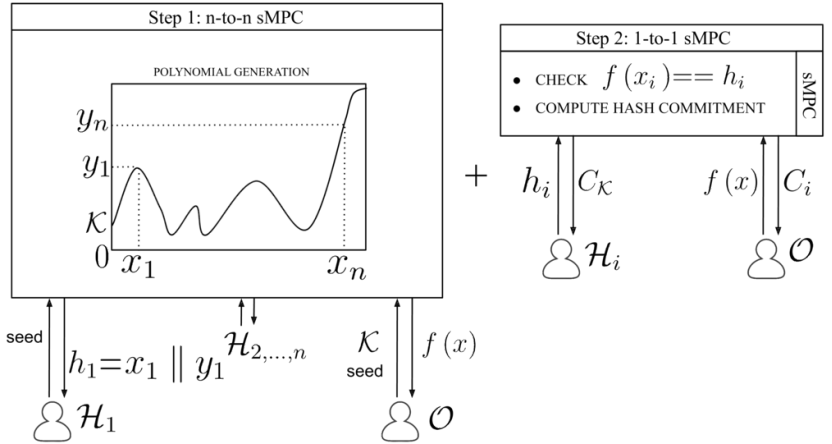
We defined two reporting protocols:

- By submitting **another user's share**, the reporter gets a reward. This counter participants selling/disclosing shares.
- By submitting **the secret itself**, the reporter gets an even higher reward. This counter coalitions reconstructing the secret ahead of time.



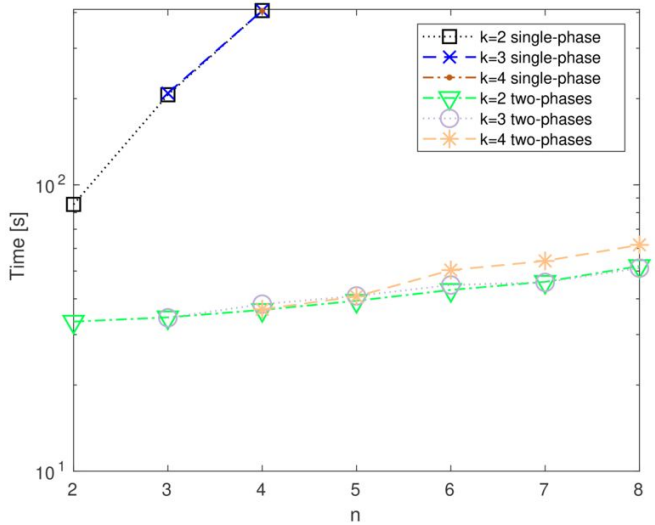
1-phase vs 2-phase sMPC *(protocols)*

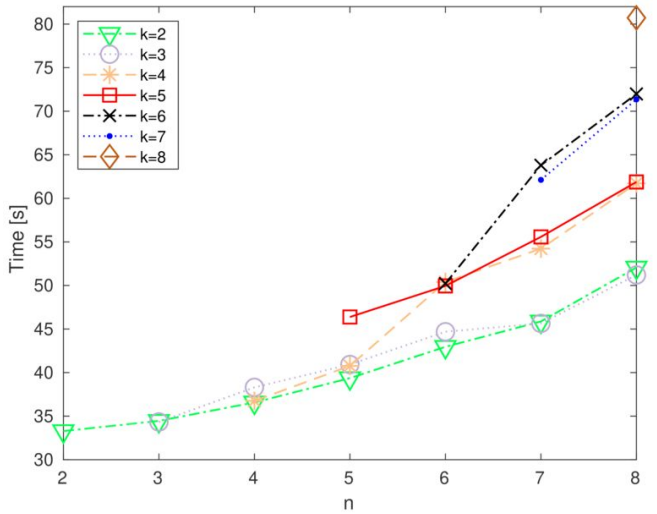




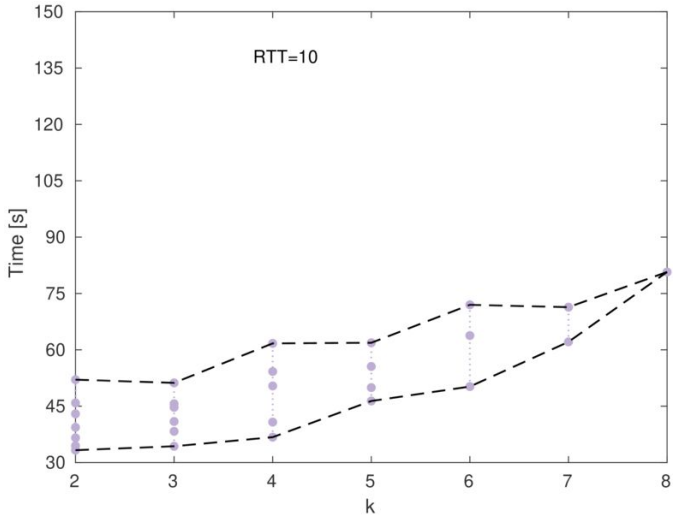
MiMC-hash (only modular arithmetic operations, ***slow***) to compute the commitments inside the sMPC.
Separating share generation from commitment computation in 2 sMPC, lowers the computation time.

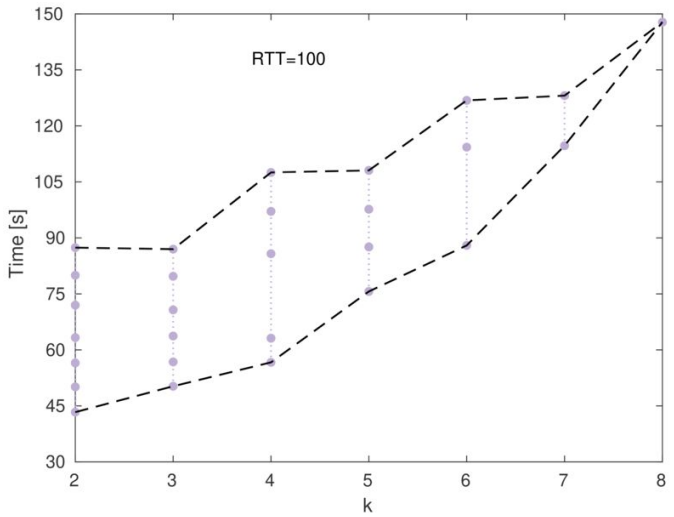
1-phase vs 2-phase sMPC *(time)*

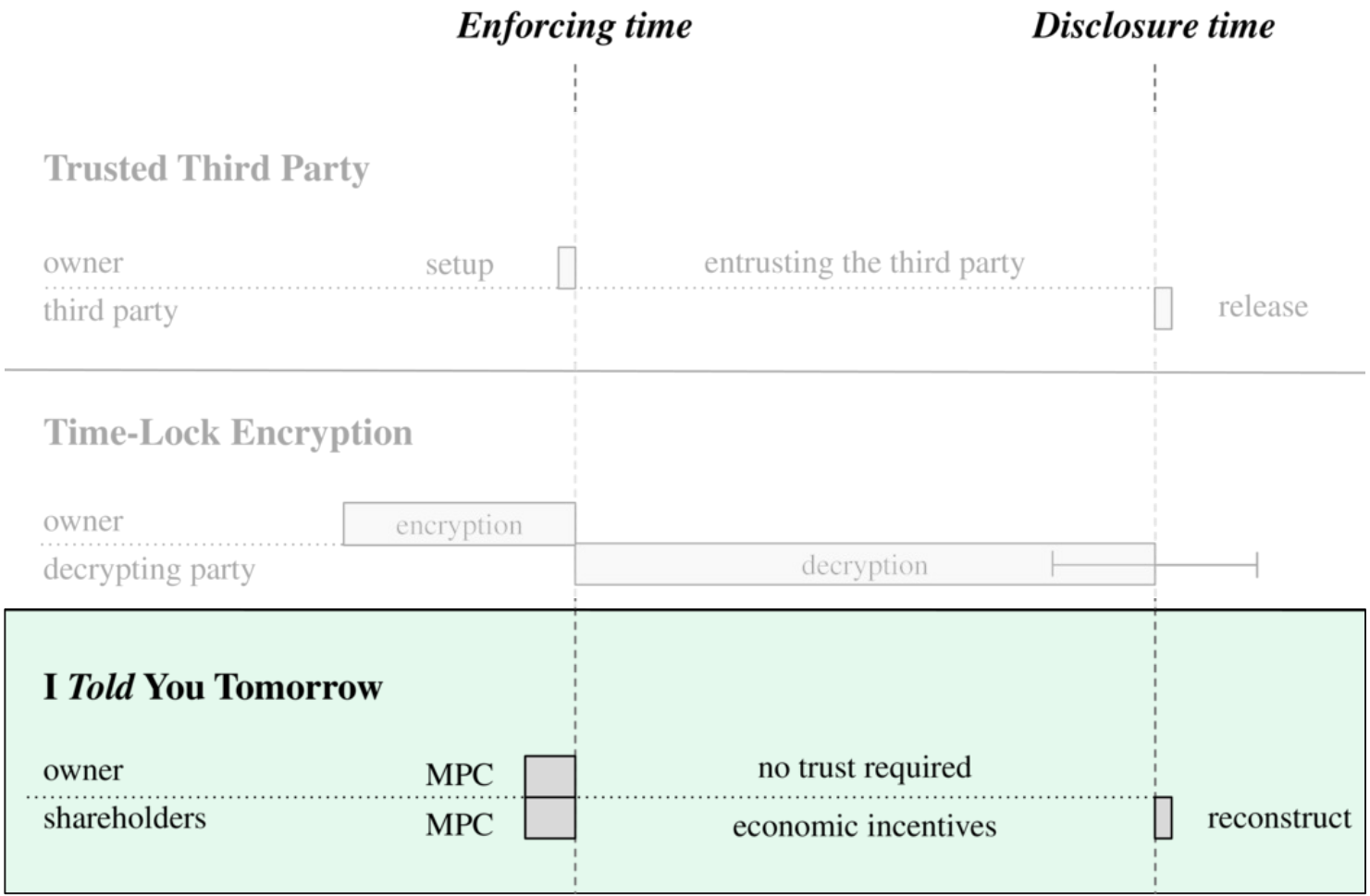




1-phase vs 2-phase sMPC *(RTT)*






Enforcing time
Disclosure time
Trusted Third Party
owner
setup
entrusting the third party
third party
release
Time-Lock Encryption
owner
encryption
decrypting party
decryption
I Told You Tomorrow
owner
MPC
no trust required
shareholders
MPC
economic incentives
reconstruct

Conclusions

- We proposed a novel way to deploy **Time-Locked secrets** based on threshold cryptography, multi-party computation, smart contracts and game theory.
- Our proposal works in an economic model in which players are **rational**, but requires **neither** trusted third parties **nor** guessing the future computing power.
- **Resources are not wasted** in decrypting the secret.
- This protocol is the first step to realize a **fully distributed cloud storage**.

Publications

- E. Bacis, D. Facchinetti, M. Rosa, M. Guarnieri, S. Paraboschi, **“I Told You Tomorrow: Practical Time-Locked Secrets using Smart Contracts”**. (under submission)

EscudoCloud & *MOSAICrOWN*
European Projects

(Horizon 2020)

ESCUDO-CLOUD

EncSwift

- Protection for outsourced data
- The server is not able to read the data but can help in applying an encryption access policy

Distributed Shuffle Index

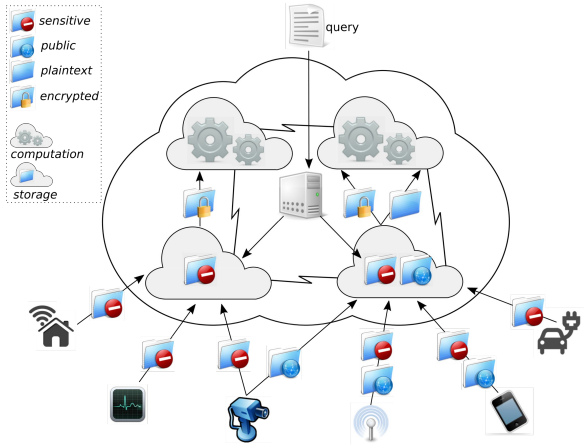
- Access pattern protection
- The server is not able to derive statistical information by looking at the access pattern

Will be discussed by Marco Rosa

MOSAICrOWN

Multi-Provider Secure Processing of Sensors Data

- Specify a data access policy for a multi-tenant environment.
- Second-stage query optimizer that generate a collaborative multi-provider query plan that complies with the policy.



Will be discussed by Dario Facchinetti

Publications

Articles and Bookchapters

Articles in Journals

• E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati. "**Securing Resources in Decentralized Cloud Storage**". *IEEE Transactions on Information Forensics and Security (TIFS) Vol. 15, n.1*. IEEE, 2019.

Chapters in Books

• E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati. "**Protecting Resources and Regulating Access in Cloud-Based Object Storage**". From Database to Cyber Security. 2018.

Proceedings Papers

Papers in Proceedings of International Conferences and Workshops

• E. Bacis, S. de Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati. "**Access Control Management for Secure Cloud Storage**". *12th EAI International Conference on Security and Privacy in Communication Networks (SECURECOMM)*. EAI, 2016.

• E. Bacis, S. De Capitani di Vimercati, S. Foresti, D. Guttadoro, S. Paraboschi, M. Rosa, P. Samarati, A. Saullo. "**Managing Data Sharing in OpenStack Swift with Over-Encryption**". *3rd Workshop on Information Sharing and Collaborative Security (WISCS)*. ACM, 2016.

• E. Bacis, S. de Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati. "**Mix&Slice: Efficient Access Revocation in the Cloud**". *23rd ACM Conference on Computer and Communications Security (CCS)*. ACM, 2016.

• E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati, "**Distributed Shuffle Index in the Cloud: Implementation and Evaluation**". *Proceedings of the 4th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, 2017.

Proceedings Papers (II)

• E. Bacis, A. Barnett, A. Byrne, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati. "**Distributed Shuffle Index: Analysis and Implementation in an Industrial Testbed**". *Proceedings of the 5th IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2017.

• E. Bacis, M. Rosa, A. Sajjad. "**EncSwift and Key Management: An Integrated Approach in an Industrial Setting**". *3rd Workshop on Security and Privacy in the Cloud (SPC)*. IEEE, 2017.

• E. Bacis, S. de Capitani di Vimercati, D. Facchinetti, S. Foresti, G. Livraga, S. Paraboschi, M. Rosa, P. Samarati. "**Multi-Provider Secure Processing of Sensors Data**". *Proceedings of the 17th International Conference on Pervasive Computing and Communications (Percom)*. IEEE, 2019.

• E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati, "**Dynamic Allocation for Resource Protection in Decentralized Cloud Storage**". *Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019.

• E. Bacis, D. Facchinetti, M. Rosa, M. Guarnieri, S. Paraboschi, "**I Told You Tomorrow: Practical Time-Locked Secrets using Smart Contracts**". Under Review.

European Project Deliverables

• S. Paraboschi, C. Cachin, E. Bacis, M. Rosa, D. Bernau, S. Delaitre, "**D2.3 - Report on Data and Access Protection**". ESCUDO-CLOUD. 2016

• E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati, B. Tackmann, D. Bernau "**D4.3 Final Versions of Tools for Security With Multiple Providers**". ESCUDO-CLOUD. 2017

• E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, A. Sajjad, P. Samarati, M. Bjorkqvist, C. Cachin, A. Taha, P. Metzler, S. Mazoor, and N. Suri. "***D2.6 – Final Report on Data Protection and Key Management Solutions"***. ESCUDO-CLOUD, 2017.

• D. Bernau, A. Fischer, A. Tueno, C. Cachin, B. Tackmann, E. Bacis, S. De Capitani di Vimercati, S. Foresti, G. Livraga, S. Paraboschi, M. Rosa, P. Samarati, R. Sassi, A. Taha, N. Coppick, R. Trapero, and N. Suri. "***D3.5 – Report on Secure Information Sharing in the Cloud"***. ESCUDO-CLOUD, 2017

• A. Taha, H. Zhang, N. Suri, S. Foresti, G. Livraga, A. Byrne, A. Barnett, E. Bacis, M. Rosa, and A. Sajjad. "***D4.4 – Report on Multi Cloud and Federated Cloud***". ESCUDO-CLOUD, 2017

Other Activities

Attended Courses

Courses in the Master Degrees of the University of Bergamo

- *Intelligenza Artificiale* - Prof. Francesco Trovò

Course in the Ph.D. program in Eng. and Applied Sciences or at other Universities

- *Introduzione alla Programmazione Scientifica* - Prof. Francesco Fassò - UniBG
- *Internet Economics* - Prof. Nicola Gatti - PoliMI
- *Biometric Systems* - Prof. Angelo Genovese - UniBG
- *Uso di Strumenti Informatici a Supporto del Ricercatore* - Prof. Angelo Gargantini - UniBG
- *Data Security and Privacy in the Cloud* - Prof.ssa Sara Foresti - UniBG
- *Practical Approaches to Cloud Assurance and Security* - Prof. Claudio Ardagna - UniMI

PhD Schools and Presentations

Schools for Ph.D. Students

- *Python for Scientific Computing* (Florence)
- *Google 2nd PhD Summit on Web Application Security* (Munich)
- *Google PhD Interns Summit Conference 2017 / 2018 / 2019* (San Francisco)
- *Google 5th / 6th PhD Summit on Compilers and Programming Languages* (Munich)

Presentations at International Conferences

- *ACM Workshop on Information Sharing and Collaborative Security (WISCS), 2016*
- *IEEE International Conference on Communications and Network Security (CNS), 2017*
- *Cyber Security Awareness Week (CSAW), 2017*
- *IEEE International Conference on Pervasive Computing and Communications (PERCOM), 2018*

Teaching and Seminars

Teaching Activities

- Teaching assistant for "*Basi di Dati 2*" (UniBG - Prof. Paraboschi)
- Teaching assistant for "*Basi di Dati e Web*" (UniBG - Prof. Paraboschi)
- Teaching assistant for "*Sicurezza Informatica*" (UniBG - Prof. Paraboschi)

Other Seminars and Dissemination Activities

- *Binary Analysis and Reverse Engineering* (UniBG)
- *Programming Paradigms* (UniBG)
- *Competitive Programming* (UniBG)

- *Improving Android Security* (Tech Talk @ Google Munich)
- *Bitcoin and Blockchains* (Ordine degli Ingegneri di Bergamo)
- *Non solo Bitcoin* (Ordine degli Ingegneri di Como)

Research Periods Abroad

Google Munich (July 2017 - September 2017)

- Advanced the state of WebAssembly memory accesses bounds checks
- Reduced runtime code patching, improving correctness, security and performance

Google London (July 2018 - October 2018)

- Improved the root exploit detection in Android native binaries and libraries

Google Zurich (July 2019 - now)

- Researching to assess and improve users' web privacy

Thanks!
Q & **A**

WebAssembly Security

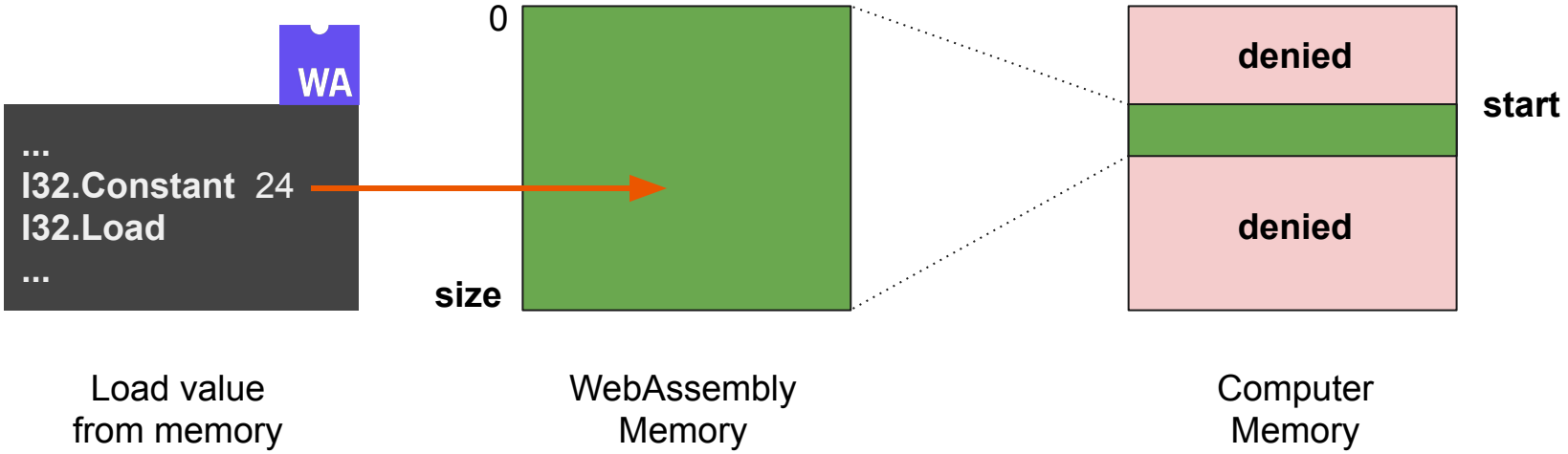
(Google Munich)

WebAssembly

- WebAssembly is a cross-browser effort to establish a low-level assembly-like language that runs on the web.
- WebAssembly can be used as a compilation target for languages such as C/C++ and aims at applications that require high and predictable performance, like games and video decoders.

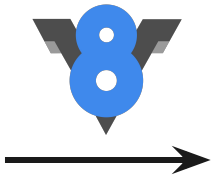
WebAssembly

WebAssembly Memory Bounds Checks



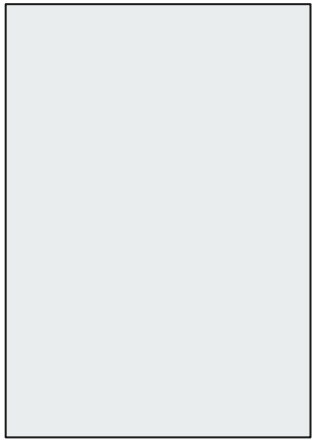
WebAssembly Memory Bounds Checks

WebAssembly Module



```
push  rbp
movq  rbp,  rsp
push  0xa
movq  rax, 0x0  // mem_size
movq  rbx, 0x0  // mem_start
call  0x2c2c1c506980
movq  rsp,  rbp
pop  rbp
retl
```

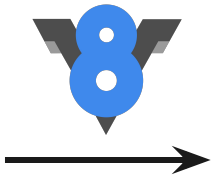
Machine Code



Memory

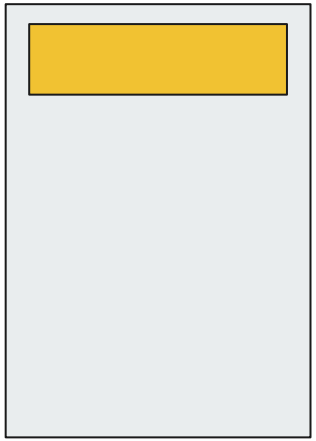
WebAssembly Memory Bounds Checks

WebAssembly Module



```
push  rbp
movq  rbp,  rsp
push  0xa
movq  rax, 0x10000
movq  rbx, 0x5575823583c0
call  0x2c2c1c506980
movq  rsp,  rbp
pop  rbp
retl
```

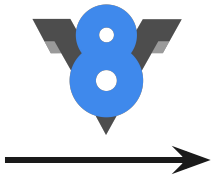
Machine Code



Memory

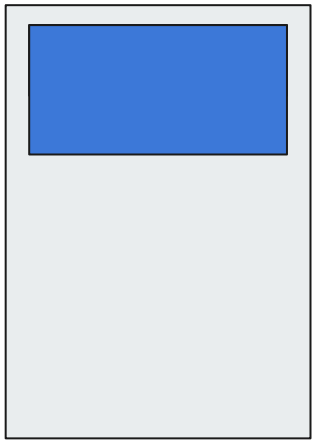
WebAssembly Memory Bounds Checks

WebAssembly Module



```
push  rbp
movq  rbp,  rsp
push  0xa
movq  rax, 0x20000
movq  rbx, 0x5575823583c0
call  0x2c2c1c506980
movq  rsp,  rbp
pop  rbp
retl
```

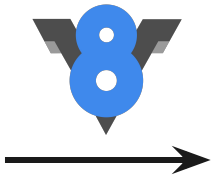
Machine Code



Memory

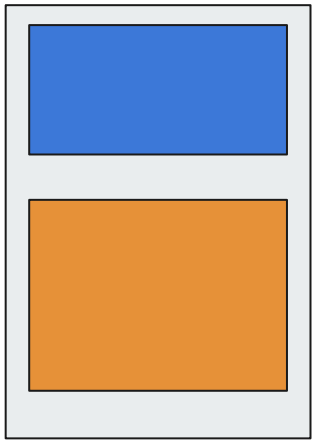
WebAssembly Memory Bounds Checks

WebAssembly Module



```
push  rbp
movq  rbp,  rsp
push  0xa
movq  rax, 0x30000
movq  rbx, 0x648abc430da2
call  0x2c2c1c506980
movq  rsp,  rbp
pop  rbp
retl
```

Machine Code



Memory

Why is run-time patching bad?

- Run-time patching can break correctness
- The code must have read-**write-execute** permissions
- Security issues can arise

Avoiding run-time patching

- Create a **Context Object** to store
  - Memory size
  - Memory start address
- Pass the Context Object to every WebAssembly function (starting from the JS entry point)
- Variables can then be changed in memory and the functions will always read correct values

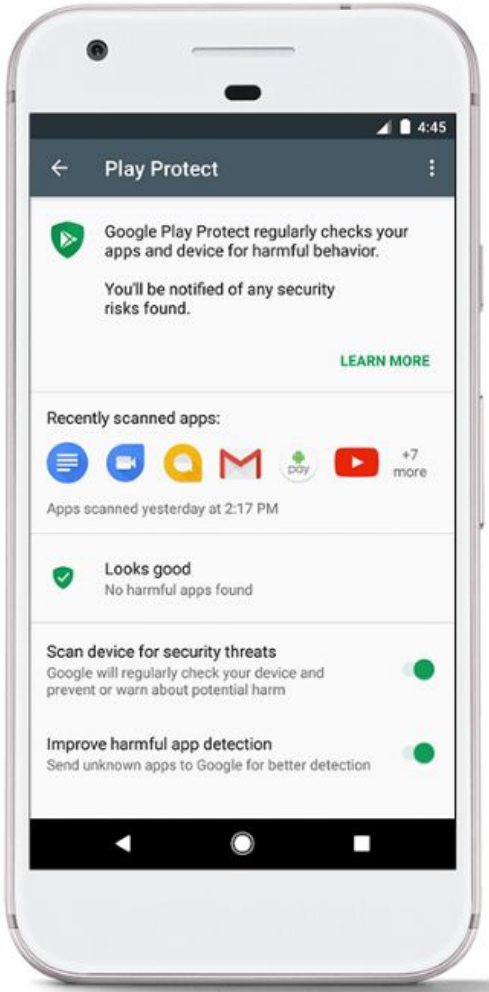
Android Security

(Google London)

Android Security

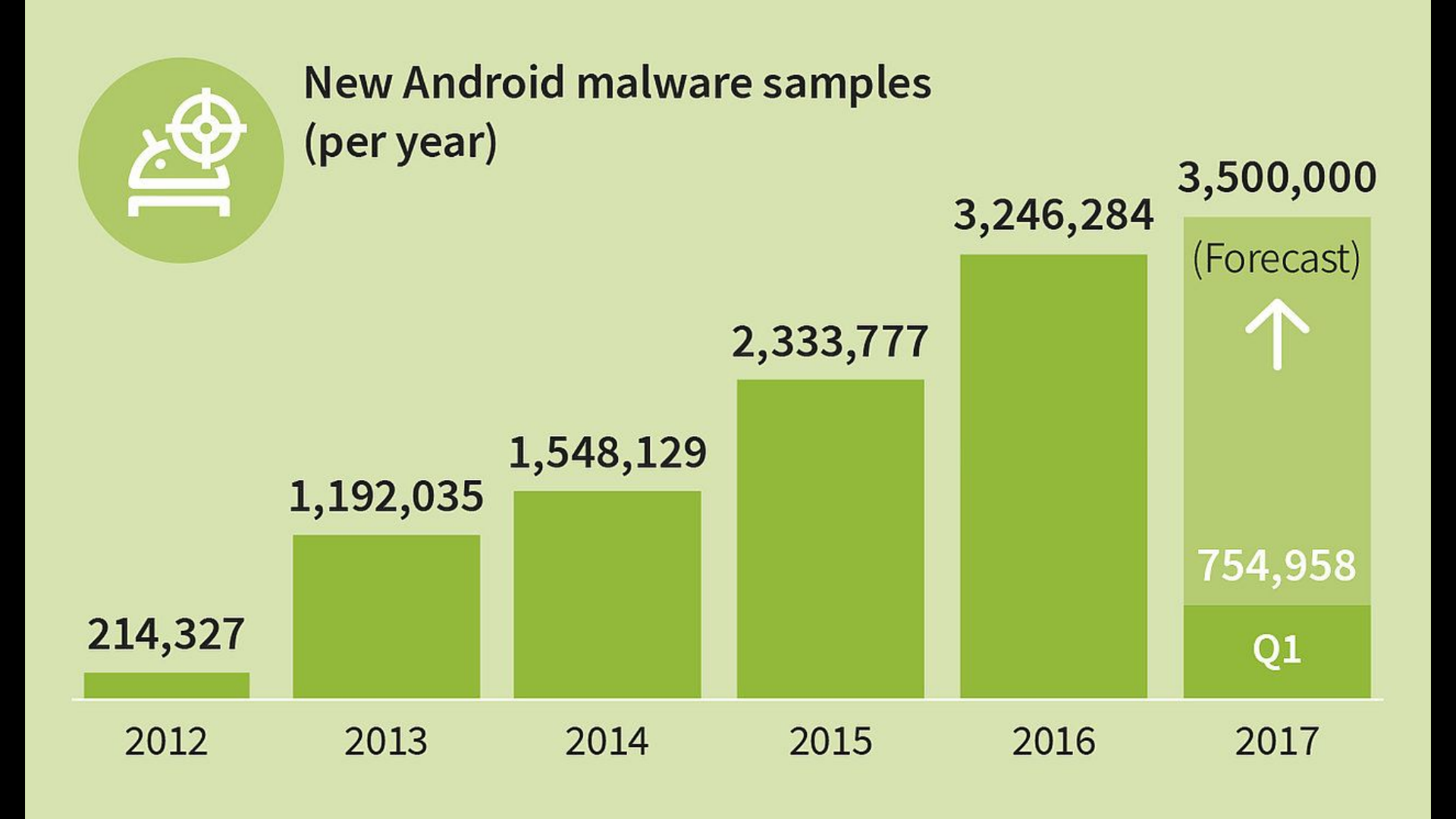
Google Play
Protect

- Android is one of the main mobile operating systems. It is thus very profitable to write malware to attack Android users.
- Google tries to identify malware app in different ways in order to prevent them from being uploaded to the Google Play Store and also to warn users about malicious apps downloaded from other sources.



Scanning and verifying over 50 billion apps every day

All Android apps undergo rigorous security testing before appearing in the Google Play Store. We vet every app and developer in Google Play, and suspend those who violate our policies. Then, Play Protect scans billions of apps daily to make sure everything remains spot on. That way, no matter where you download an app from, you know it's been checked by Google Play Protect.


New Android malware samples (per year)
214,327
1,192,035
1,548,129
2,333,777
3,246,284
3,500,000
(Forecast)
754,958
Q1
2012
2013
2014
2015
2016
2017

Malware Analysis

Static Analysis

- Examine program code without executing it
- Works with both source and compiled code
  - Source code is easier to understand
- Can be manual or automatic

Con

- Does not work for obfuscated code

Dynamic Analysis

- Code is executed on the target machine
- Input values are supplied
- Internal memory is analyzed

Con

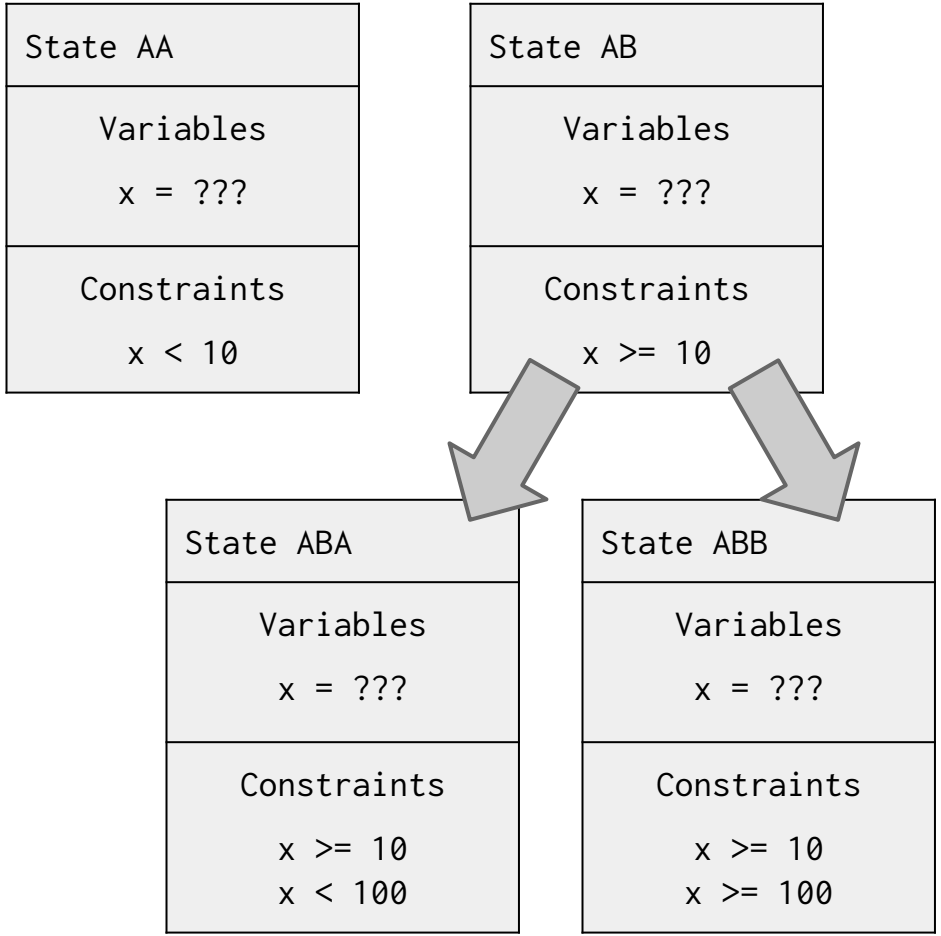
- Hard to reach good coverage

How to generate test input for an obfuscated program?

Concolic Analysis（Concrete + Symbolic）

1. Classify a particular set of variables as ***symbolic*** *variables*.
2. **Instrument** the program so that each operation which affects a symbolic variable is logged.
3. Choose an arbitrary input to begin with.
4. **Execute** the program.
5. Symbolically re-execute the program, generating a set of symbolic constraints.
6. Negate the last path condition not already negated in order to visit a new execution path.
7. Invoke an **automated theorem prover to generate a new input**.
8. Return to step 4.

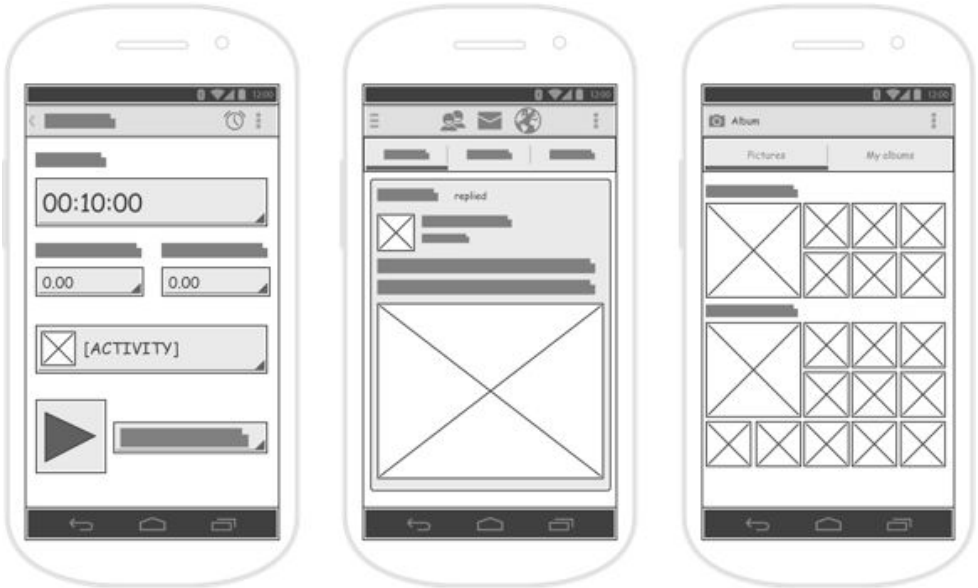
```
int x;
scanf(“%d”, &x);
if (x >= 10) {
    if (x < 100) {
        printf(“Some.\n”);
    } else {
        printf("Lots!\n");
    }
} else {
    printf("Few.\n");
}
```



Then, create an app.

Once the test inputs have been generated,

It’s time to create the app and analyze it.



Q & A