RSAConference2018

San Francisco | April 16 – 20 | Moscone Center

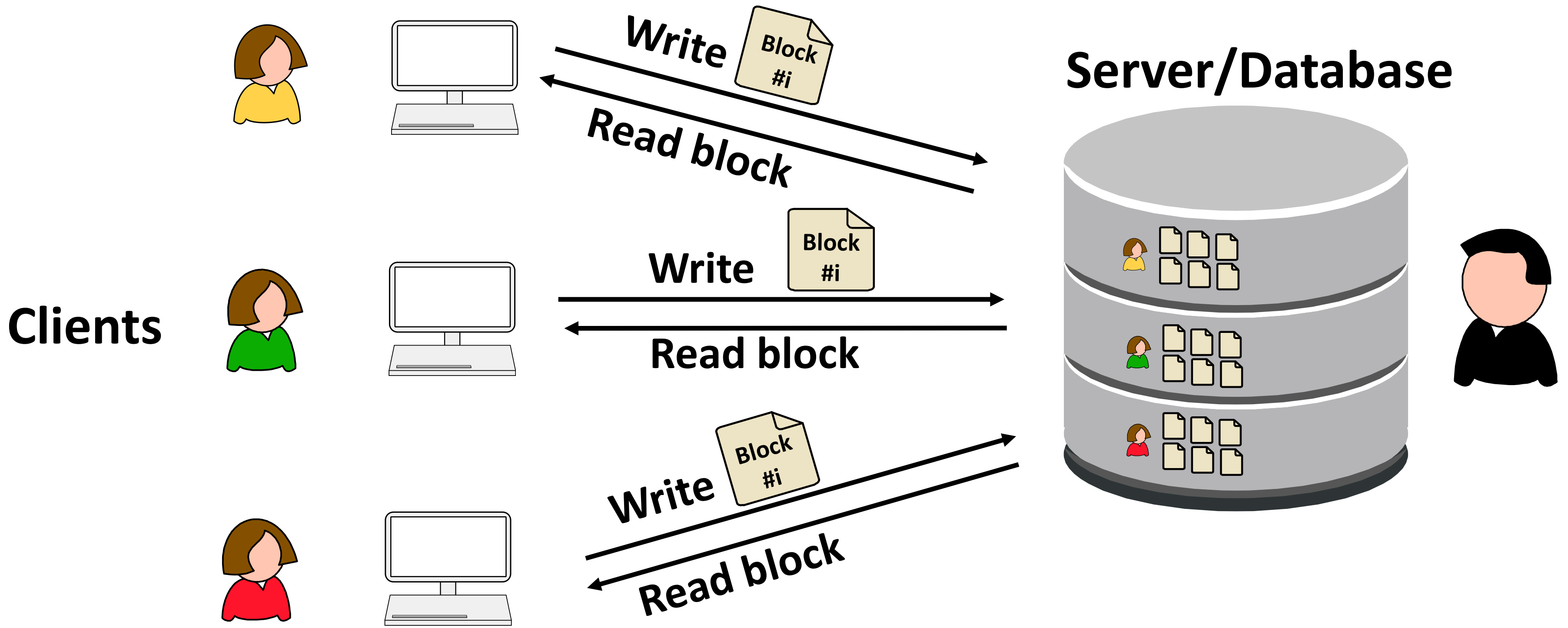
SESSION ID: CRYP-R14

COMPOSABLE AND ROBUST OUTSOURCED STORAGE

Christian Badertscher and Ueli Maurer

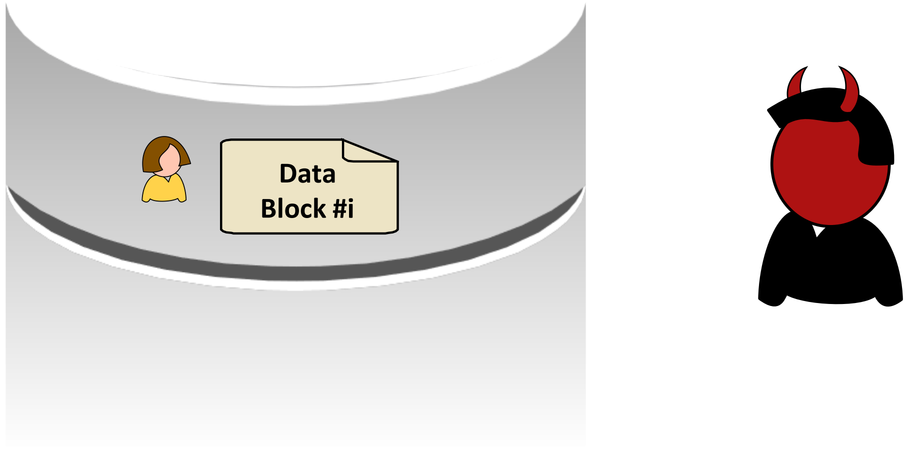
ETH Zurich, Switzerland

Motivation


Write
Block #i
Read block
Write
Block #i
Read block
Write
Block #i
Read block
Server/Database
Clients

Outsourced Storage: Security Goals

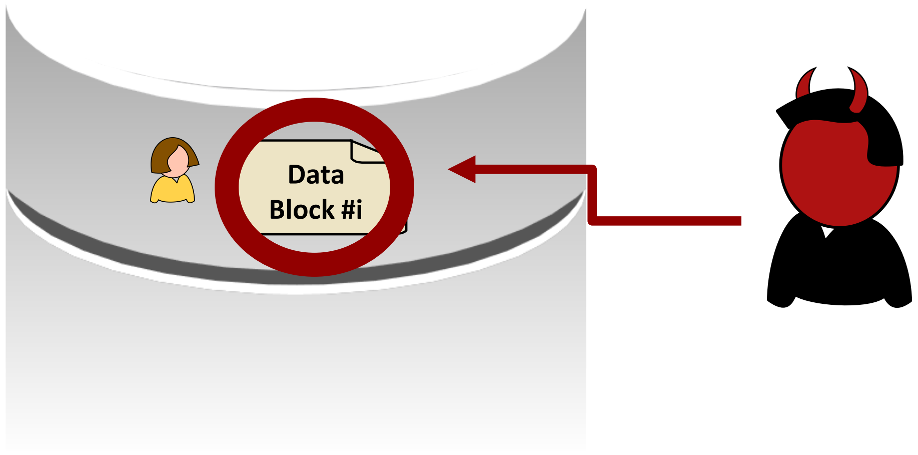
Server/Database



In general: Insecure

Outsourced Storage: Security Goals

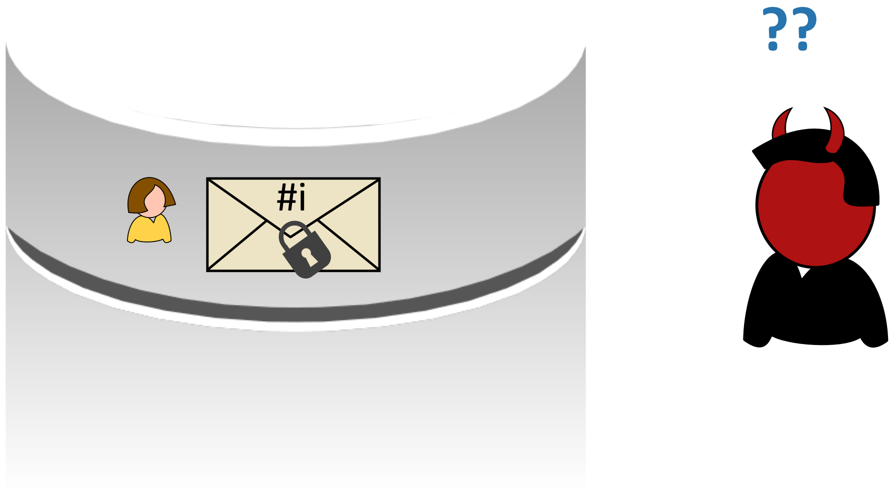
Server/Database



- **Detect malicious modifications**
- **Detect rollbacks of valid data blocks**

Outsourced Storage: Security Goals

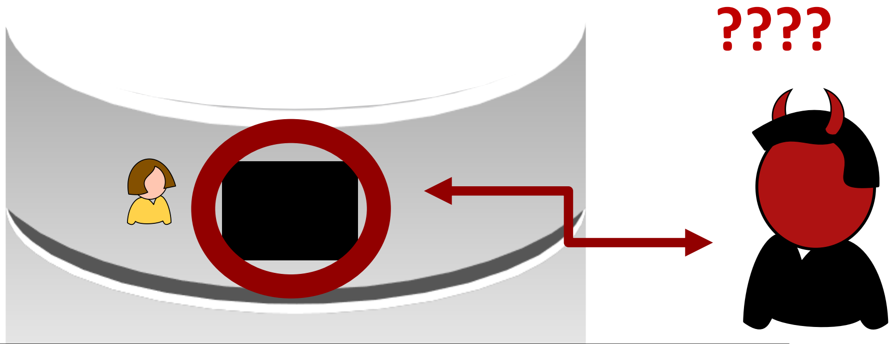
Server/Database



- Confidentiality of the content

Outsourced Storage: Security Goals

Server/Database



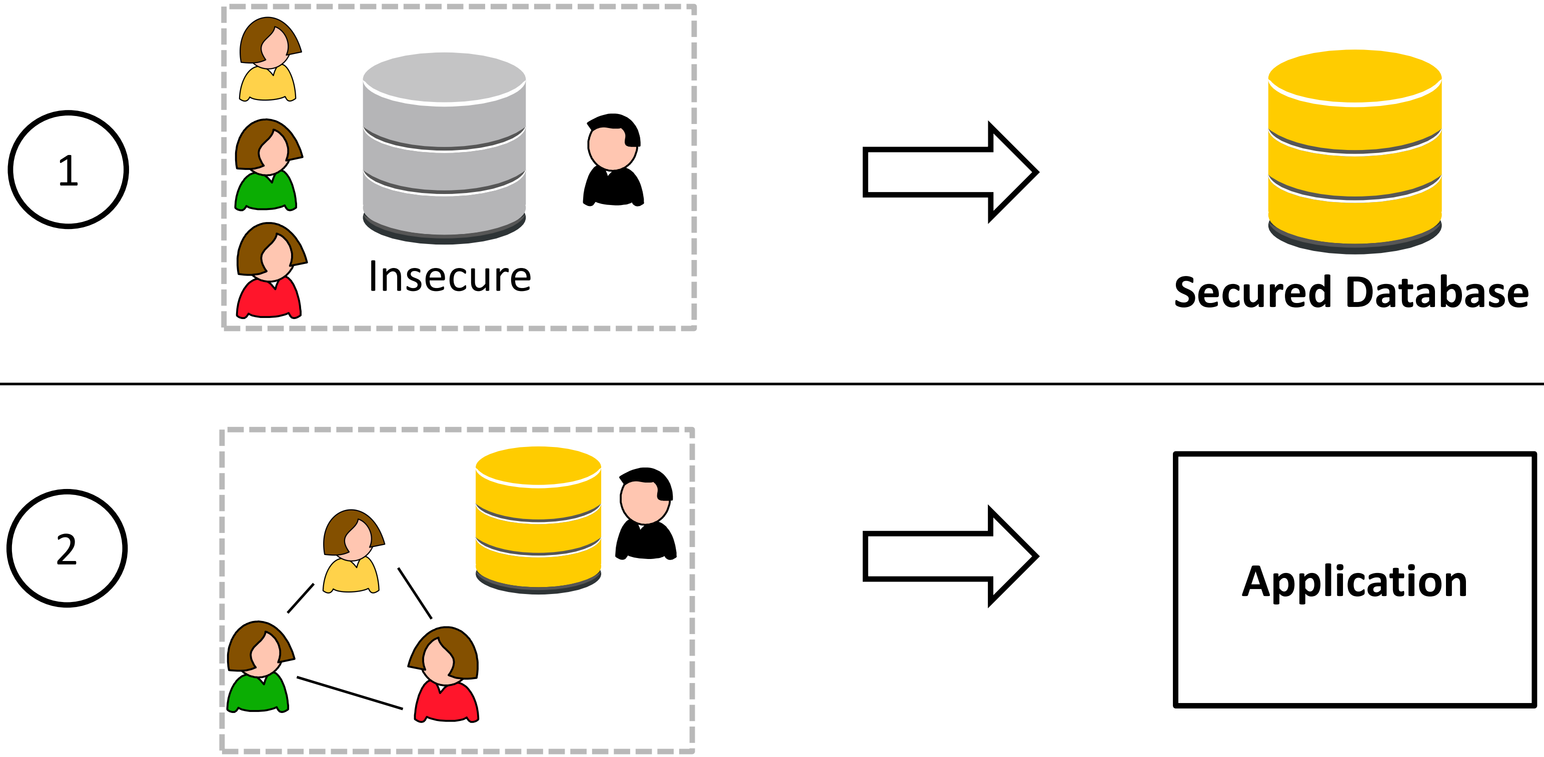
Alice's server memory should look like a black box to the server provider:

- **Leaks** at most **number of accesses**
- **Hides access pattern** and content
- **No undetected modifications possible**

Applications of a Storage Abstraction

- Use the storage abstraction in cryptographic protocols
  - Store and retrieve information
  - Design and prove entire networked file systems
- Conduct a modular proof in a composable framework
  - Assume an outsourced storage resource as hybrid
  - Construct stronger from weaker resources

Composability



Robustness

- **Abort-on-Error** is a common mechanism (e.g., TLS sessions)

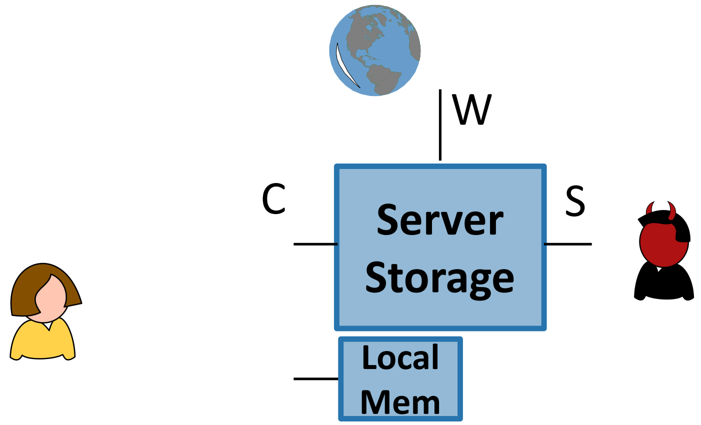
Robustness

- **Abort-on-Error** is a common mechanism (e.g., TLS sessions)

- Different with outsourced storage
  - Recovery, memory dump, …
  - In general: access whatever is there (e.g., **after** a failure or security breach)
  - Solutions: Distribute, Replicate, **or: Robust Storage Protocols**

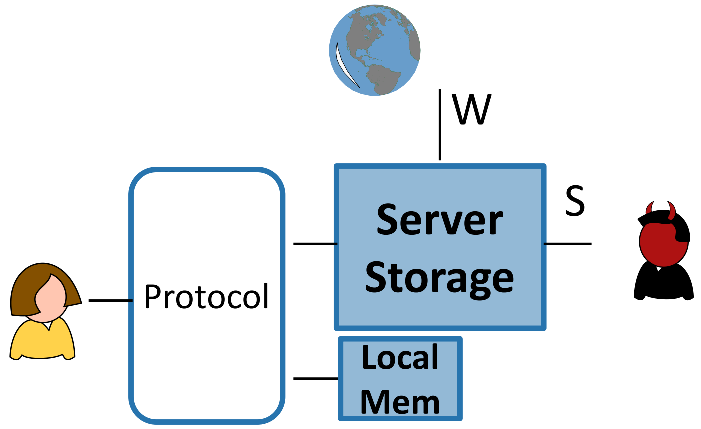
- However: **Robustness could compromise security!**

Constructions



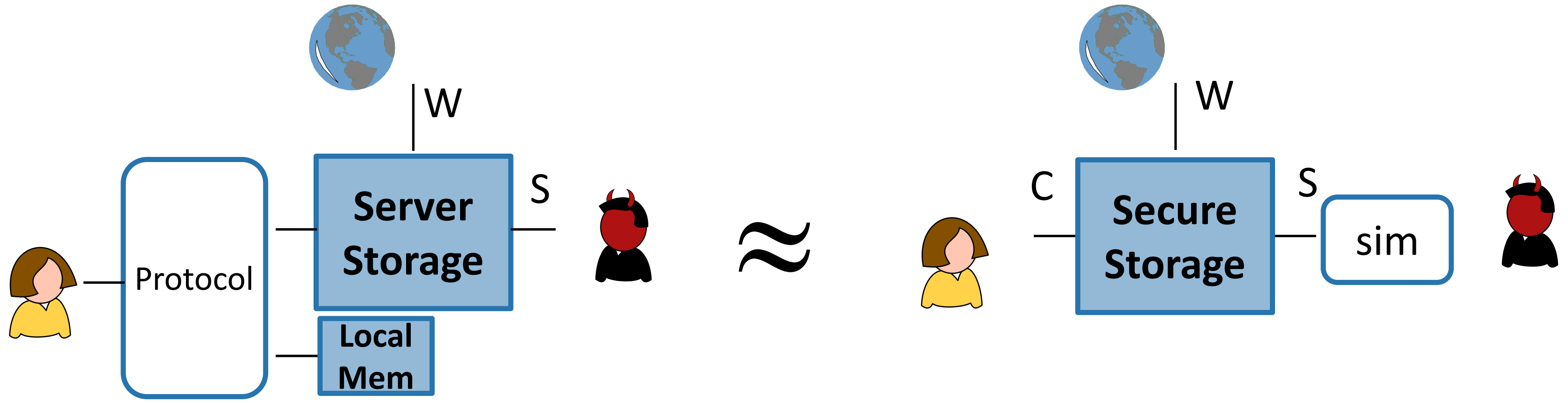
The real world

Constructions



The real world

Constructions



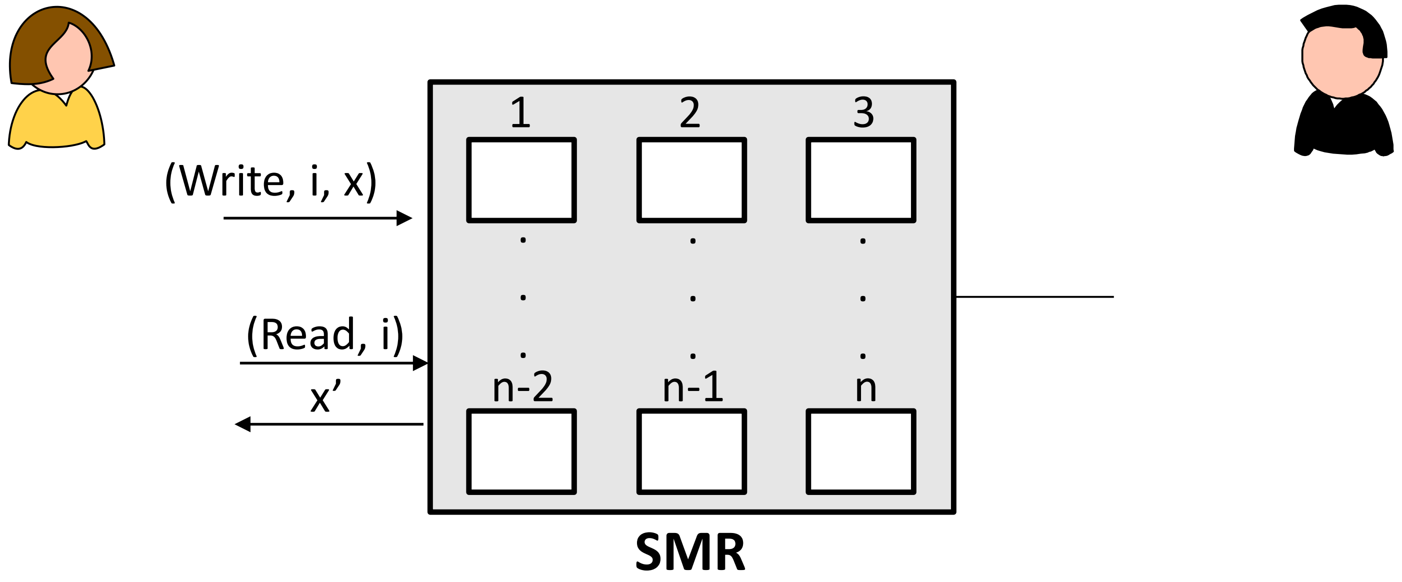
The real world

The ideal world

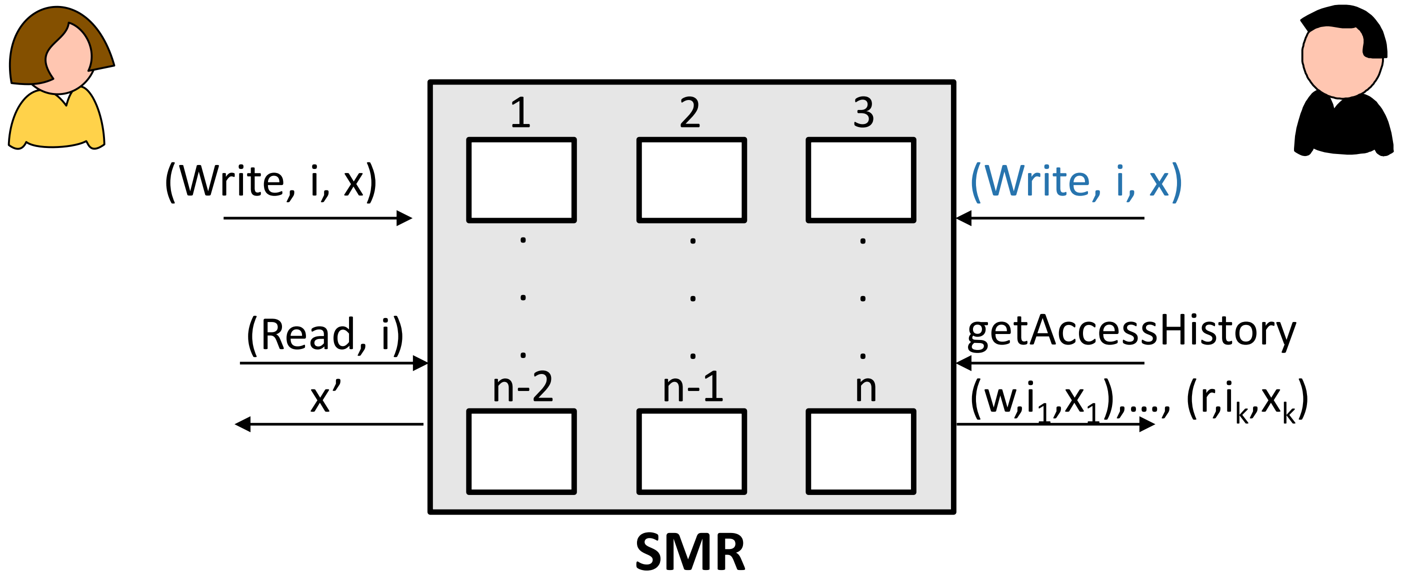
A New Model for Outsourced Storage

- We **design a formal model** for composable and robust outsourced storage.
- We capture various **client-side security provisions** including composable **retrievability guarantees**.
- We design robust **schemes** that ensure these guarantees and **review** the security of **existing schemes**.

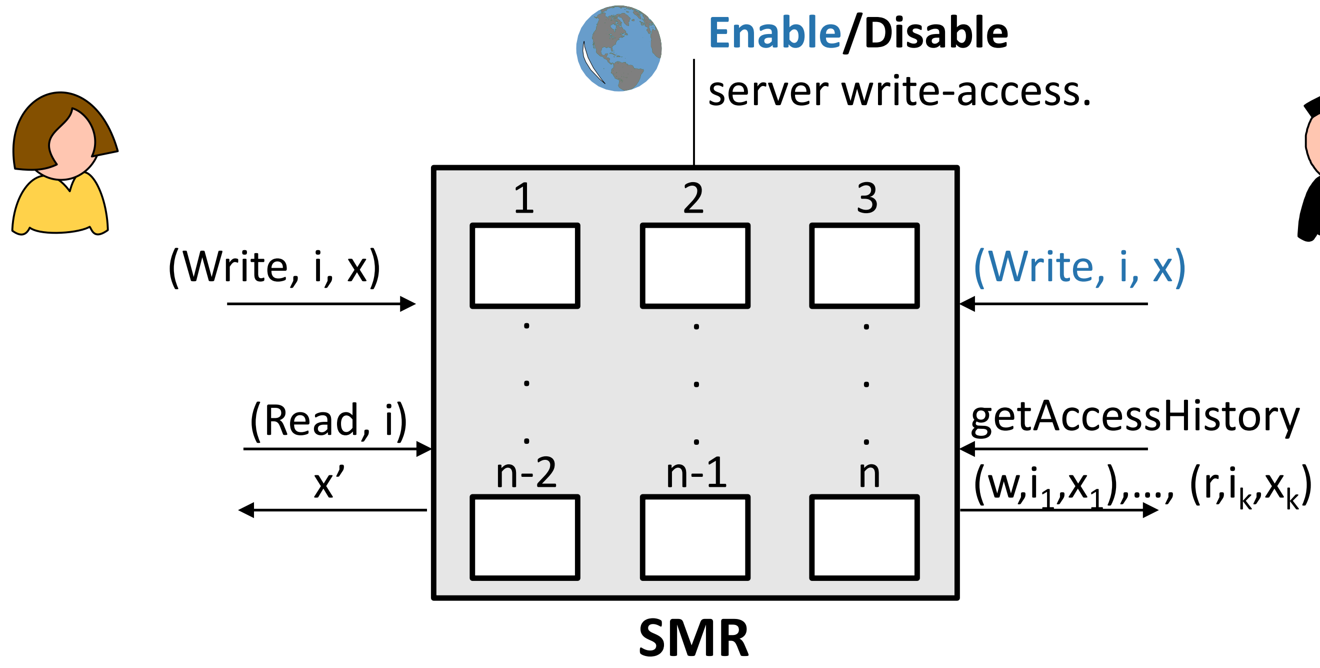
Basic Server-Memory Resource



Basic Server-Memory Resource



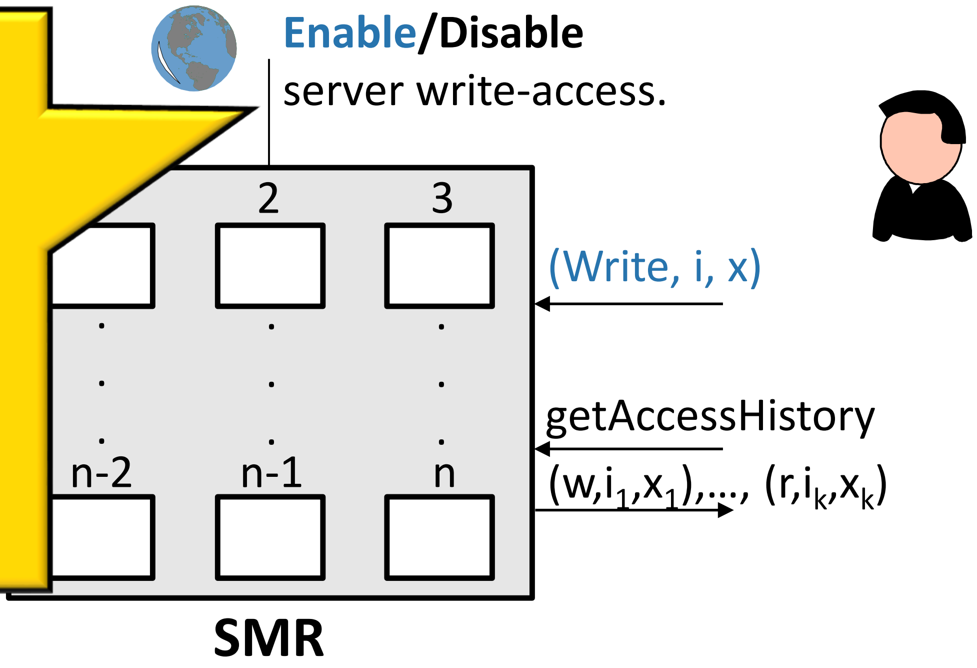
Basic Server-Memory Resource



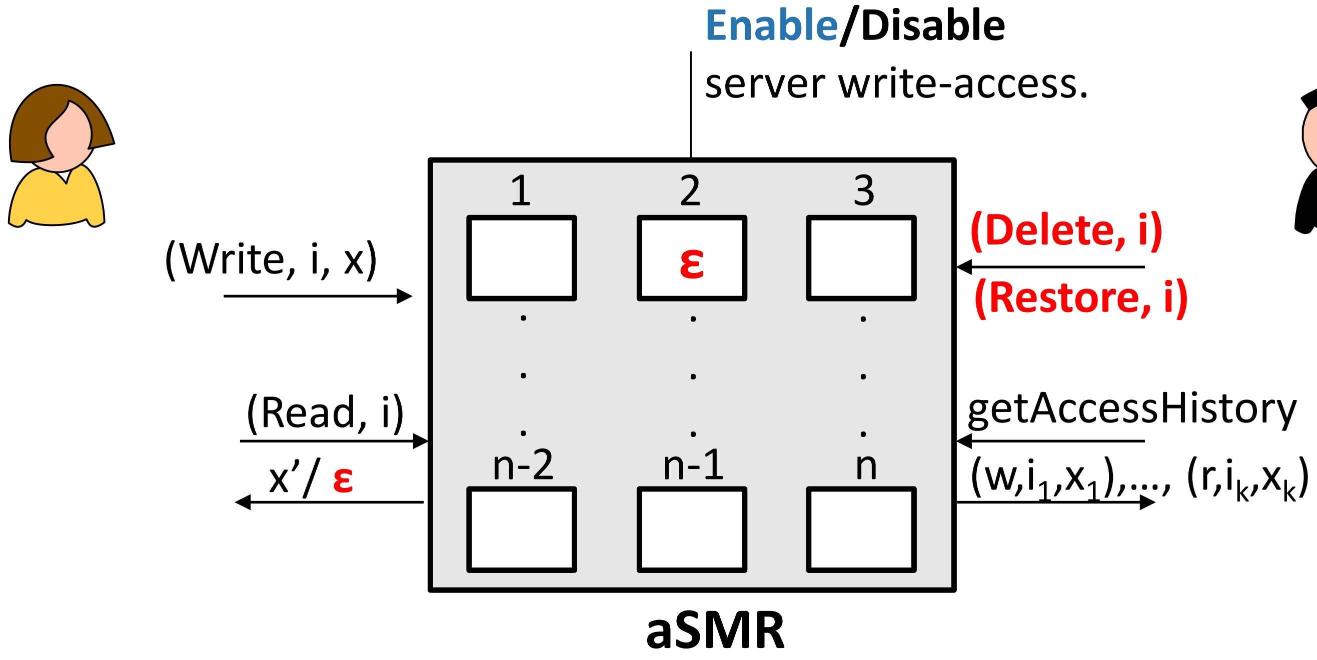
Basic Server-Memory Resource

Direct interaction with resources at interface W:

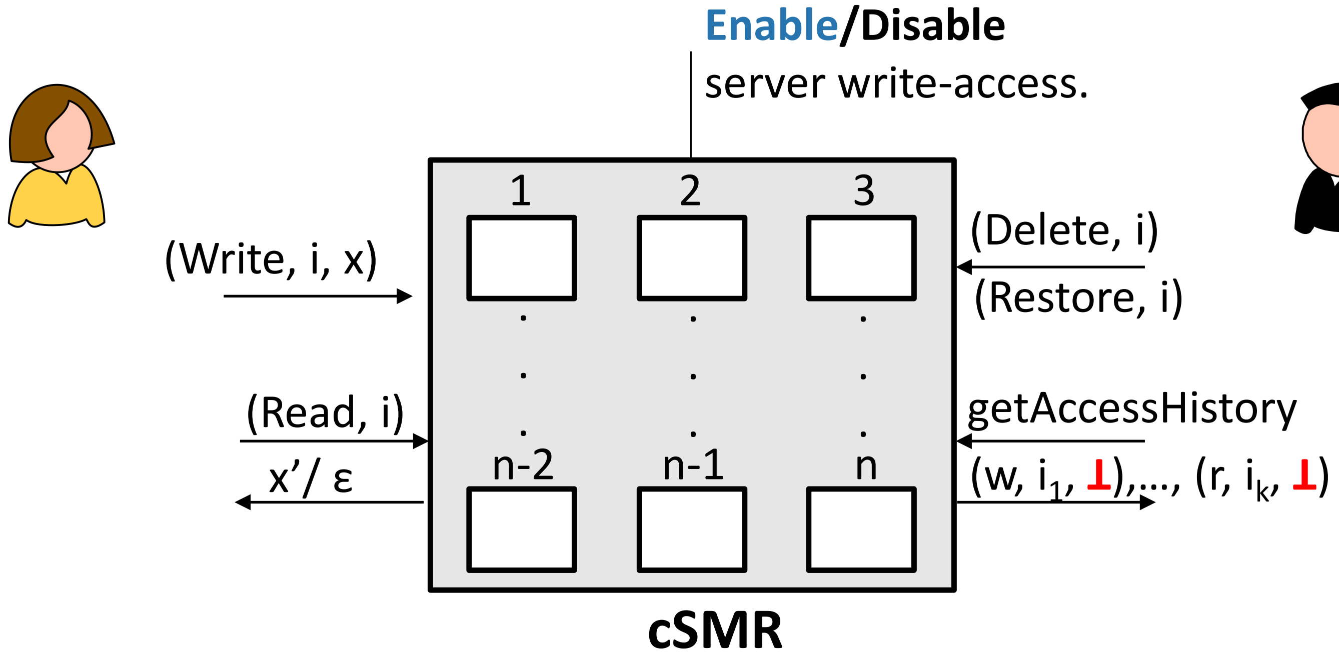
- Not a hard-coded adversarial capability
- But this typical worst-case is also covered
- Specific form of robustness is modeled



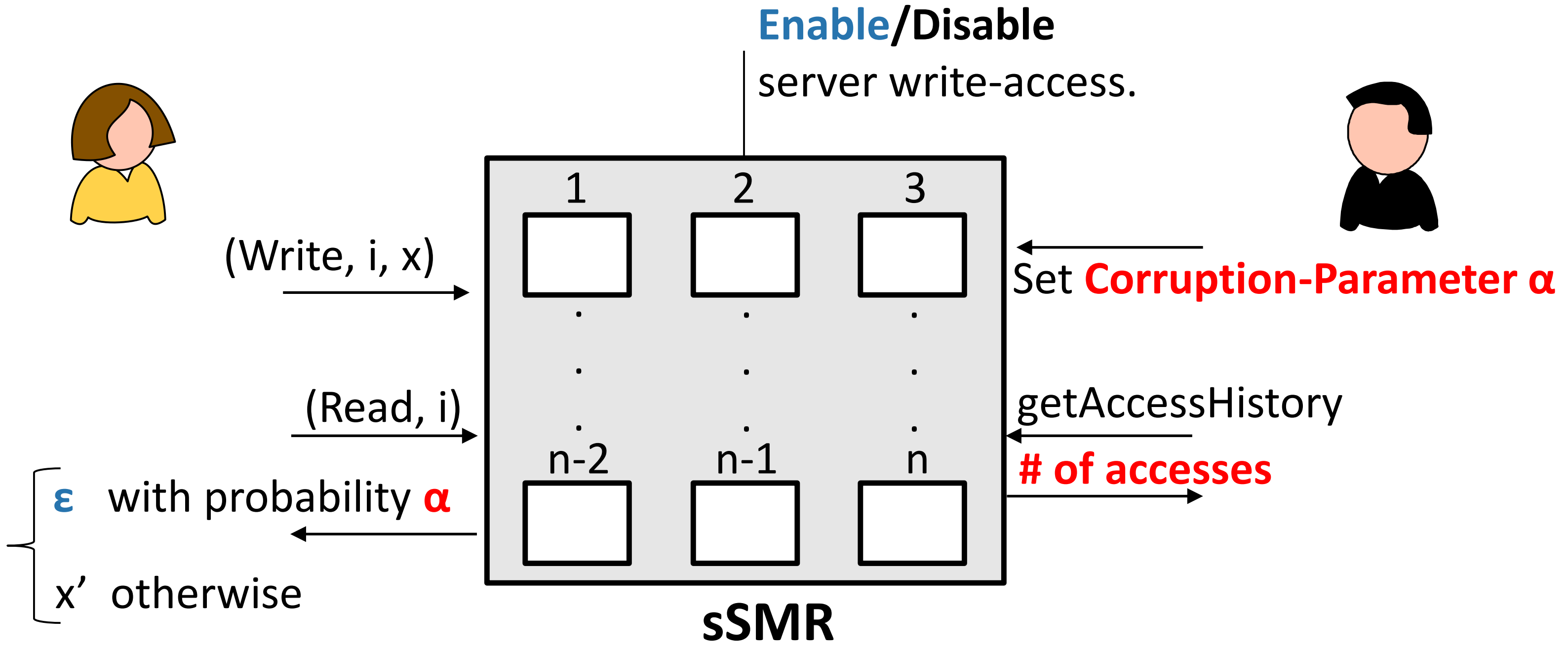
Authentic Server-Memory Resource



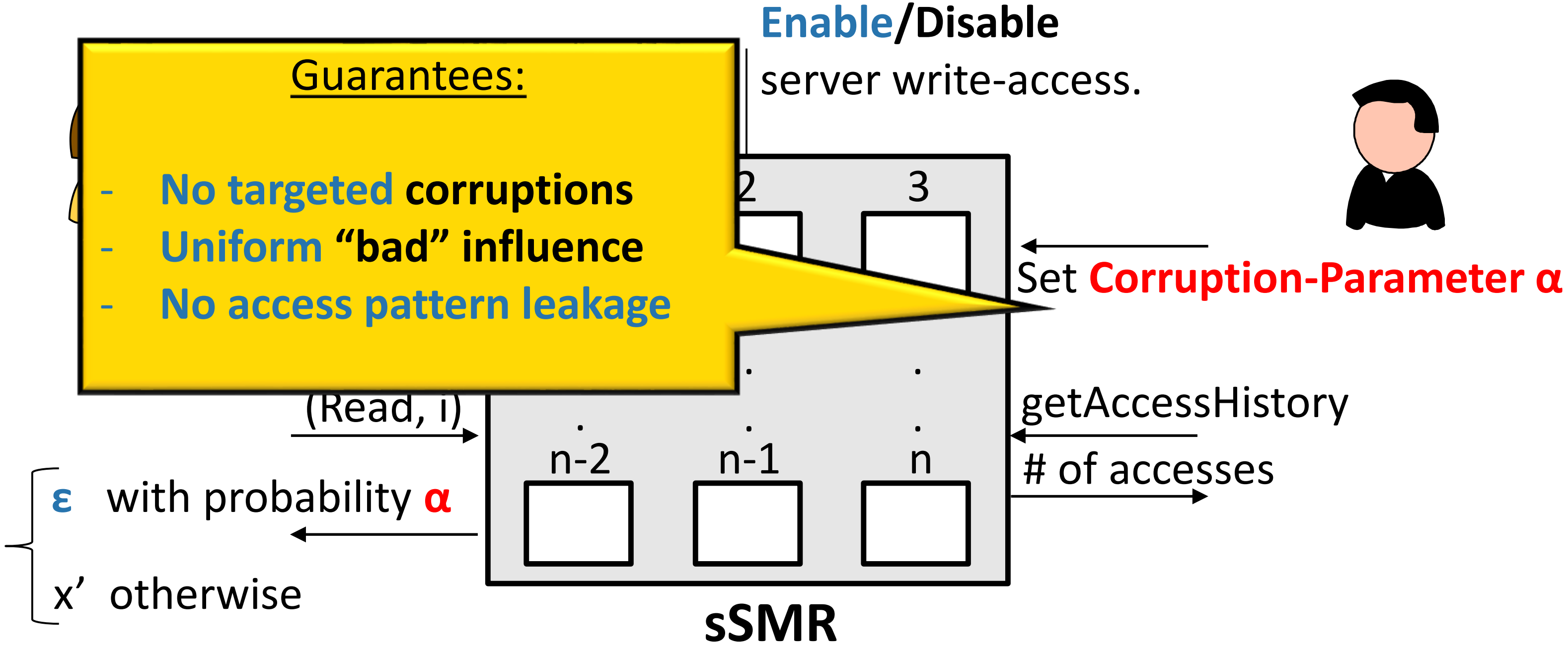
Confidential Server-Memory Resource



Secure Server-Memory Resource



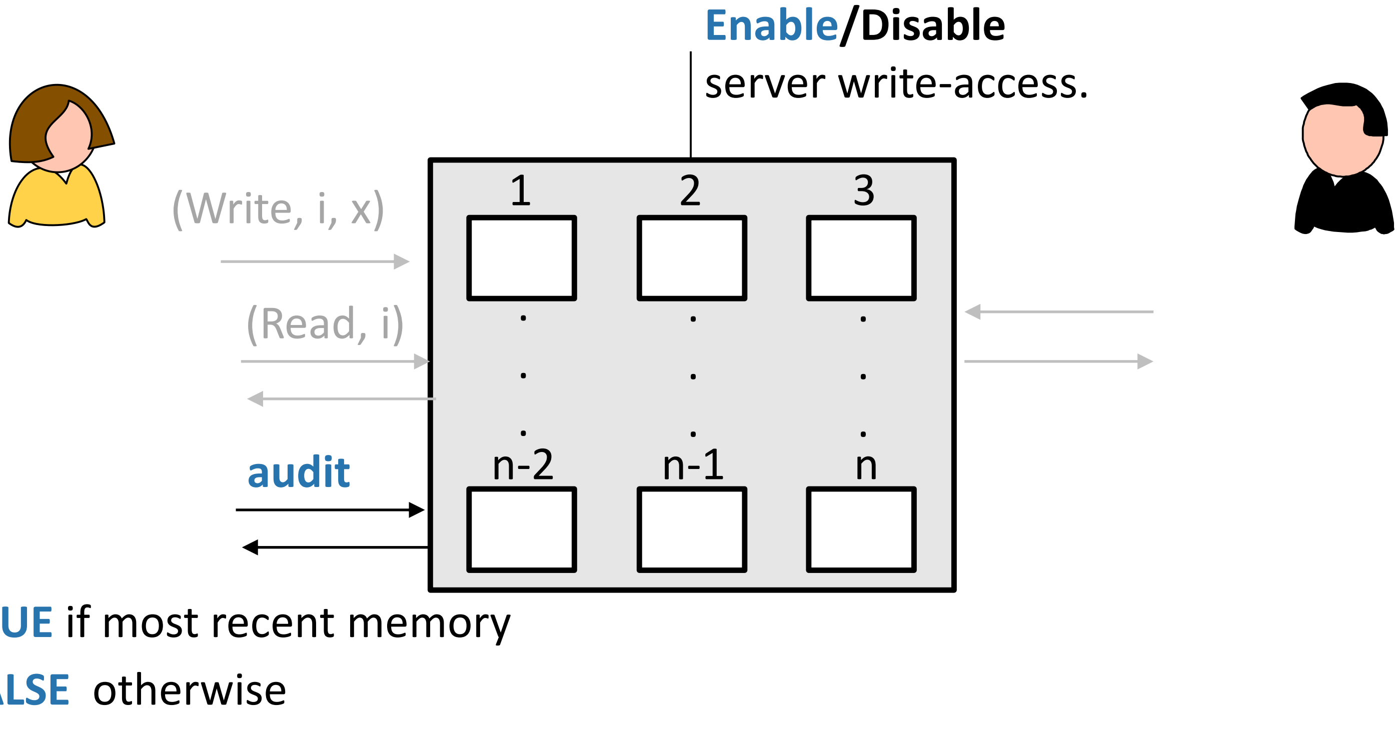
Secure Server-Memory Resource



Guarantees:

- No targeted corruptions
- Uniform "bad" influence
- No access pattern leakage

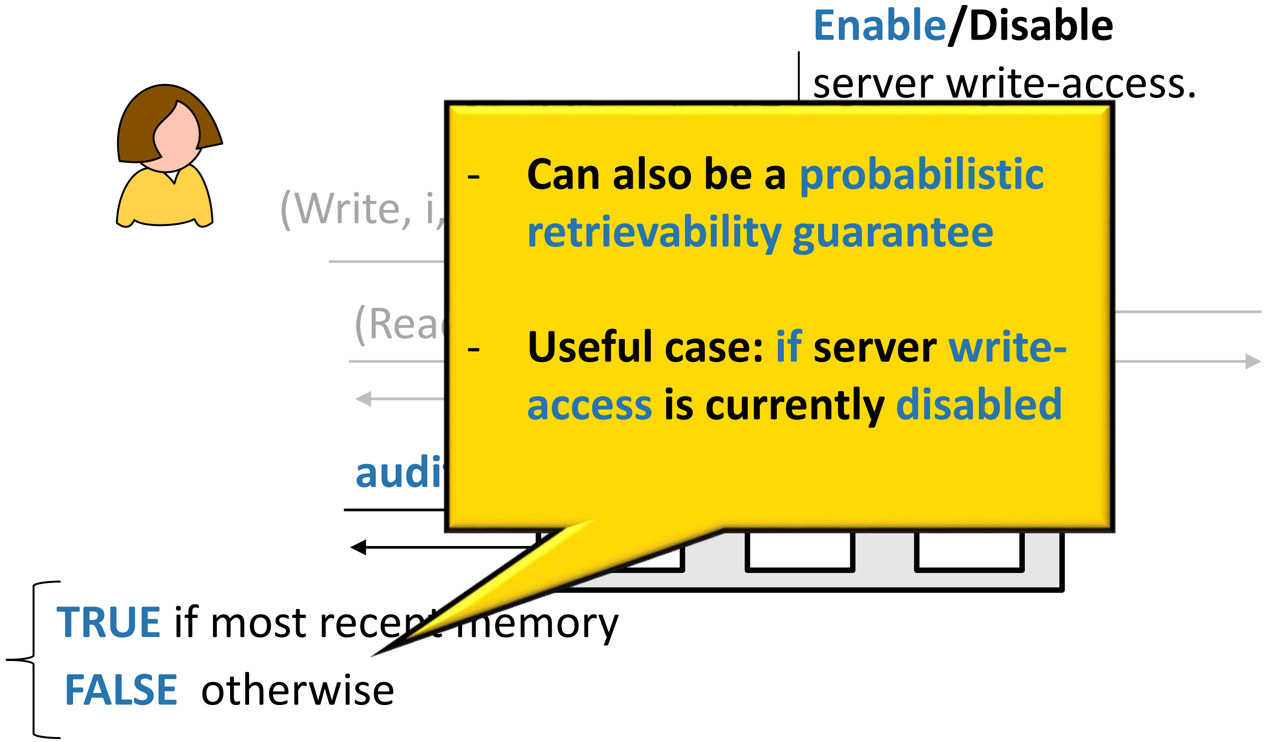
Auditable Server-Memory Resource



TRUE if most recent memory

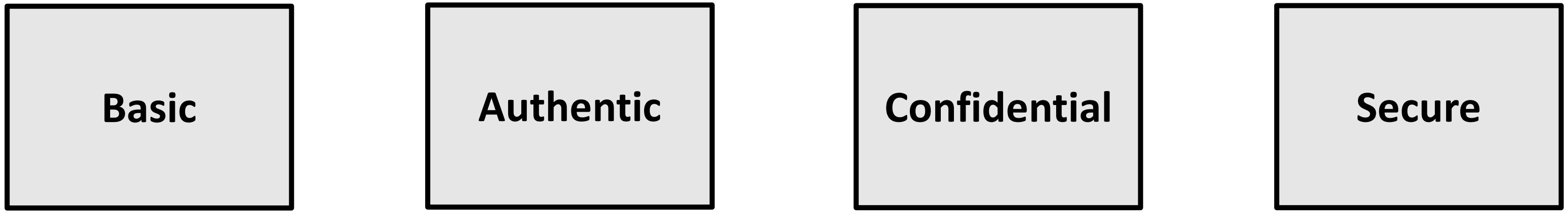
FALSE otherwise

Auditable Server-Memory Resource

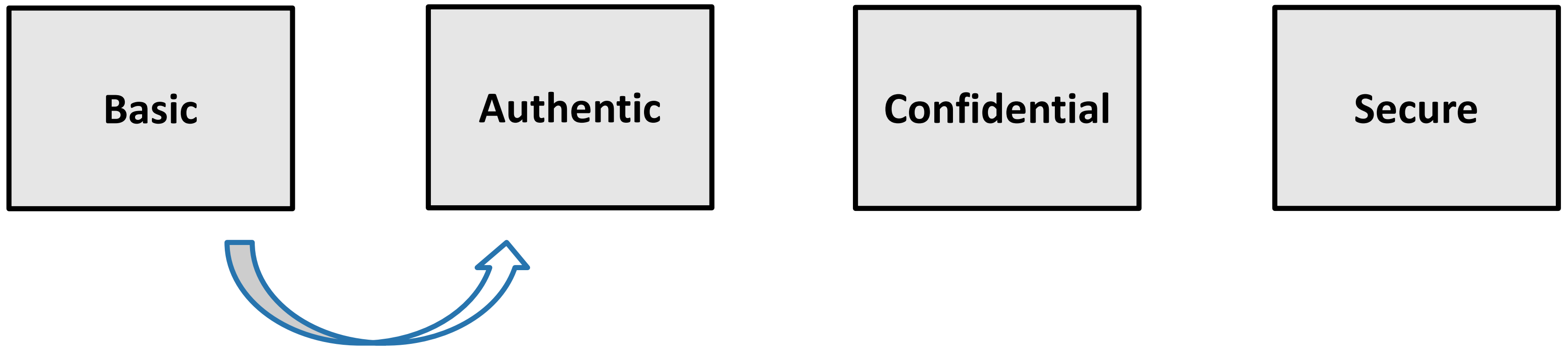


- **Can also be a probabilistic retrievability guarantee**
- **Useful case: if server write-access is currently disabled**

Protocols

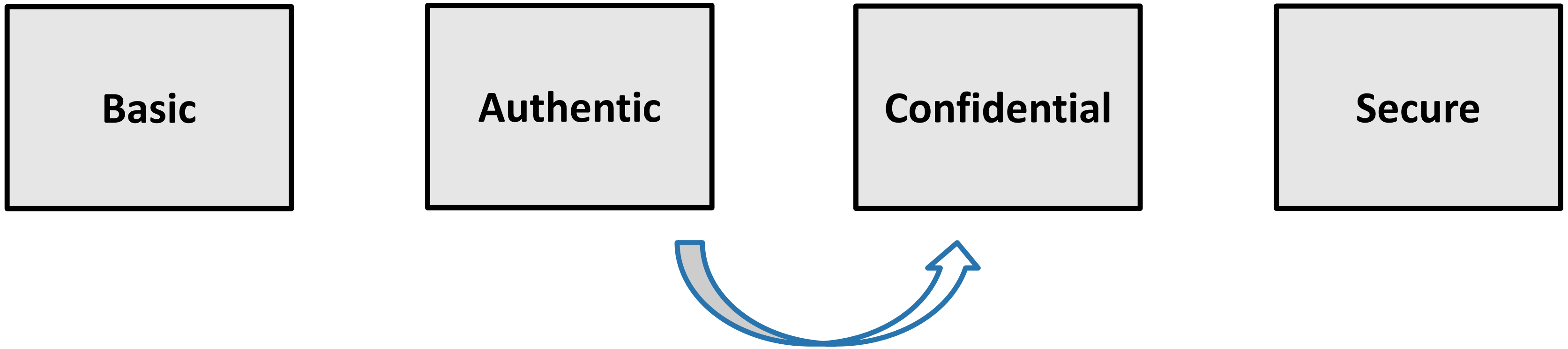


Protocols



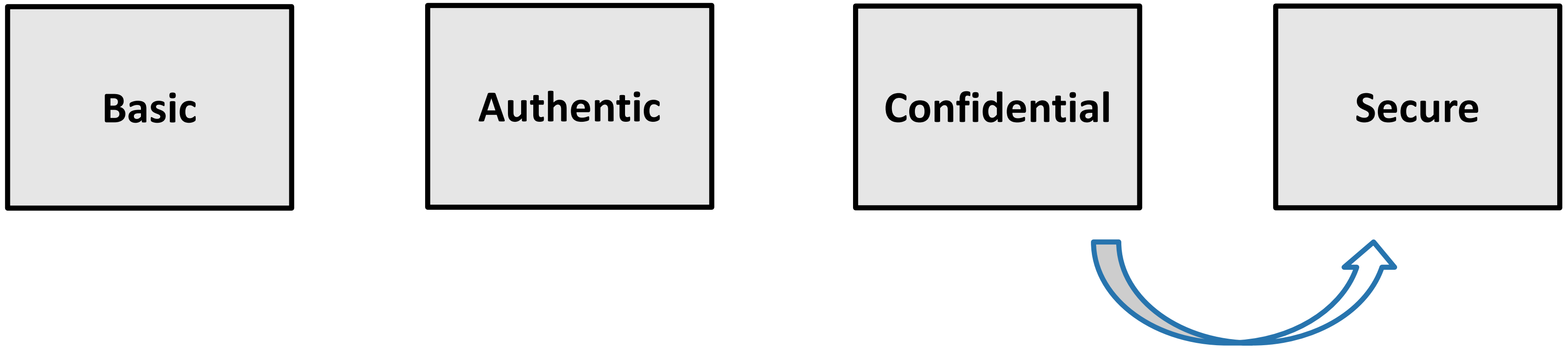
- **M**essage-**A**uthentication **C**odes + Authentication Trees (e.g., Blum)

Protocols



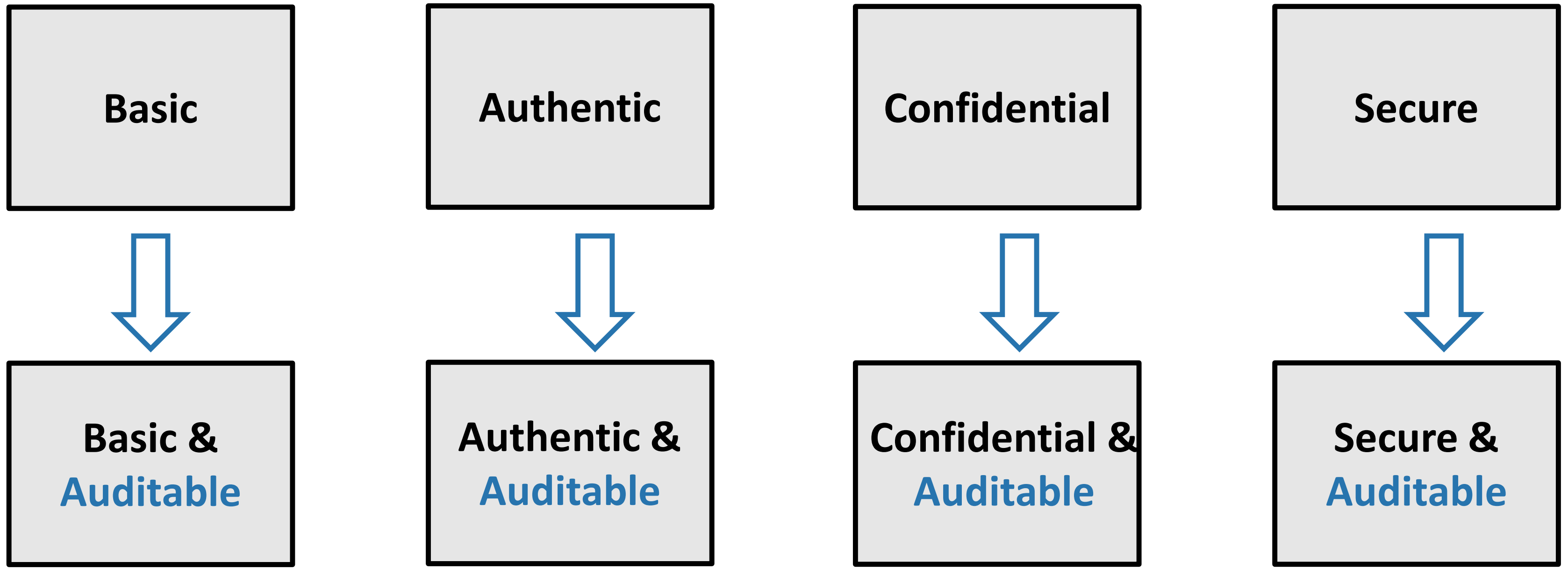
- Symmetric Encryption

Protocols

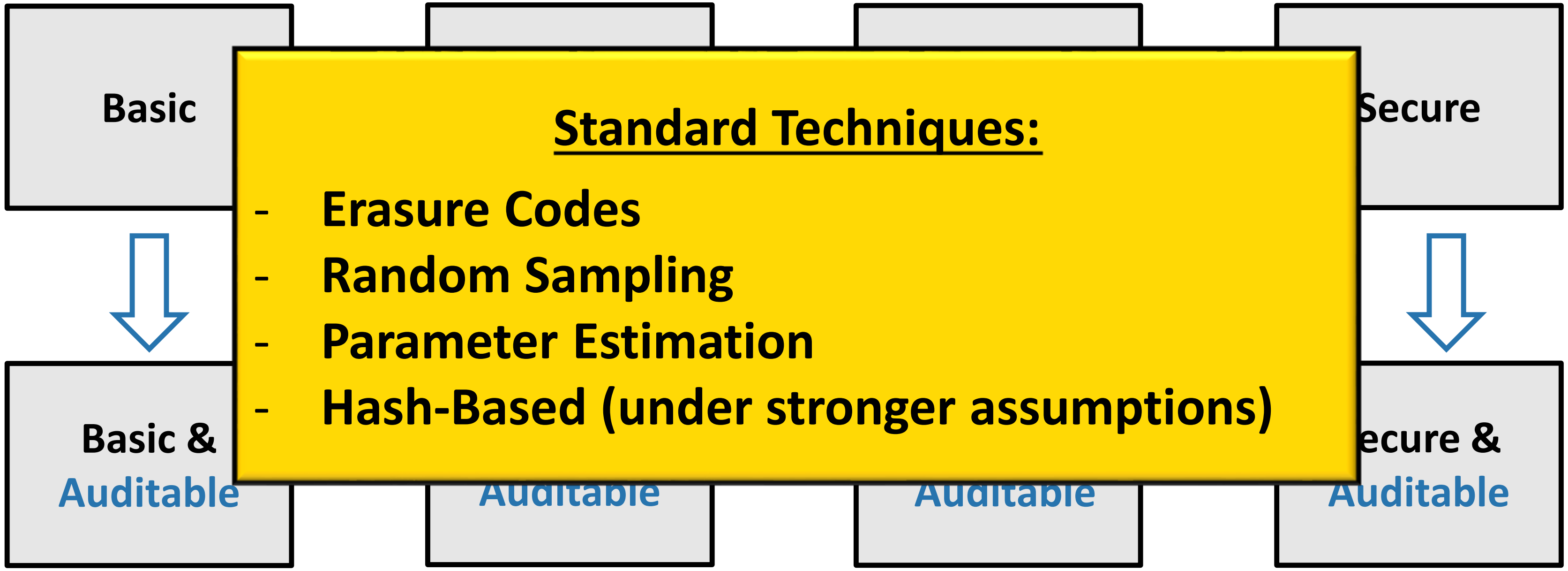


- Strengthened Oblivious RAM (e.g., Path-ORAM + Error Handling)

Protocols - Audits



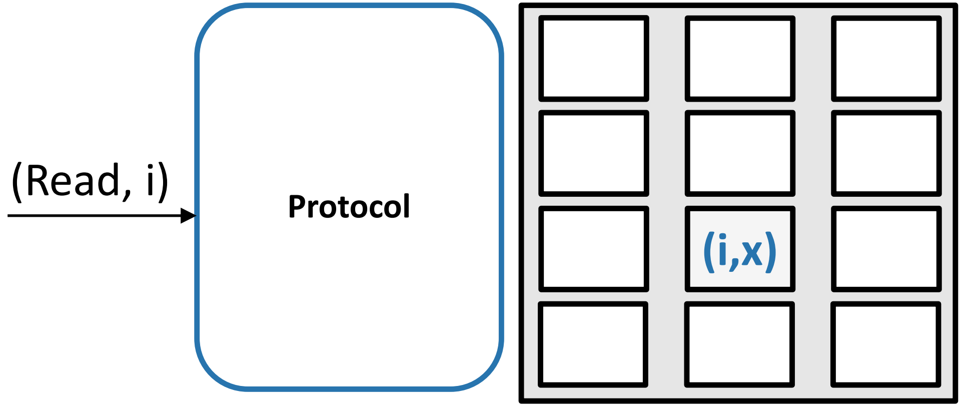
Protocols - Audits



<u>Standard Techniques:</u>

- **Erasure Codes**
- **Random Sampling**
- **Parameter Estimation**
- **Hash-Based (under stronger assumptions)**

Special Case: Achieving Secure Storage



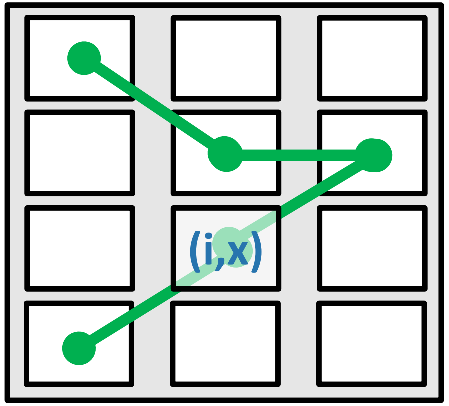
Authentic & Confidential
Server Memory Resource

Special Case: Achieving Secure Storage

(Read, i)

1.) Create **pseudo-random access sequence** to server locations

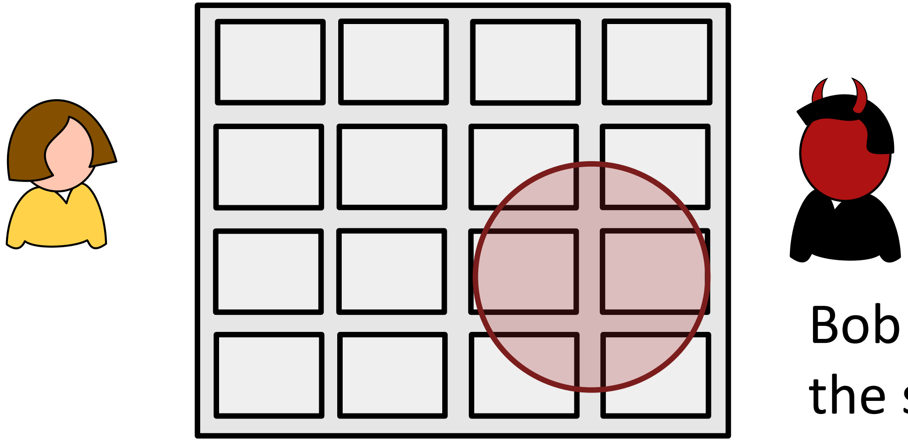
2.) **Re-structure** part of **memory**



Authentic & Confidential Server Memory Resource

The Issue with Side-Channels

Authentic & Confidential Server Memory Resource

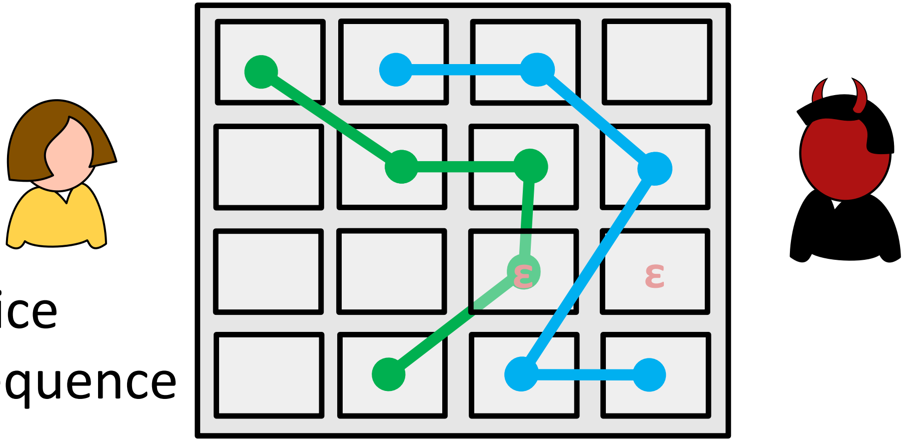


1

Bob deletes part of the storage where he assumes that Alice stores her logical block i.

The Issue with Side-Channels

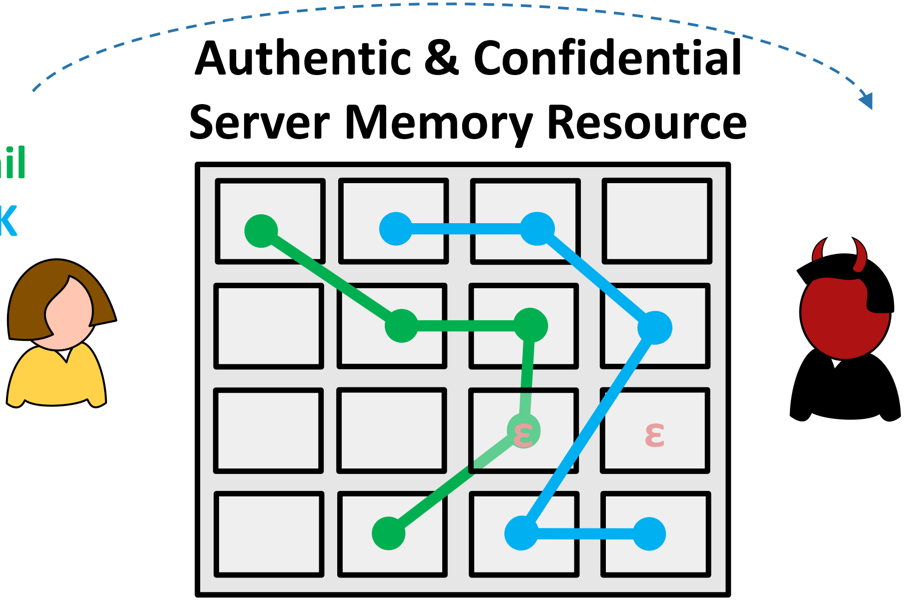
Authentic & Confidential Server Memory Resource



2 Assume Alice makes a sequence of requests.

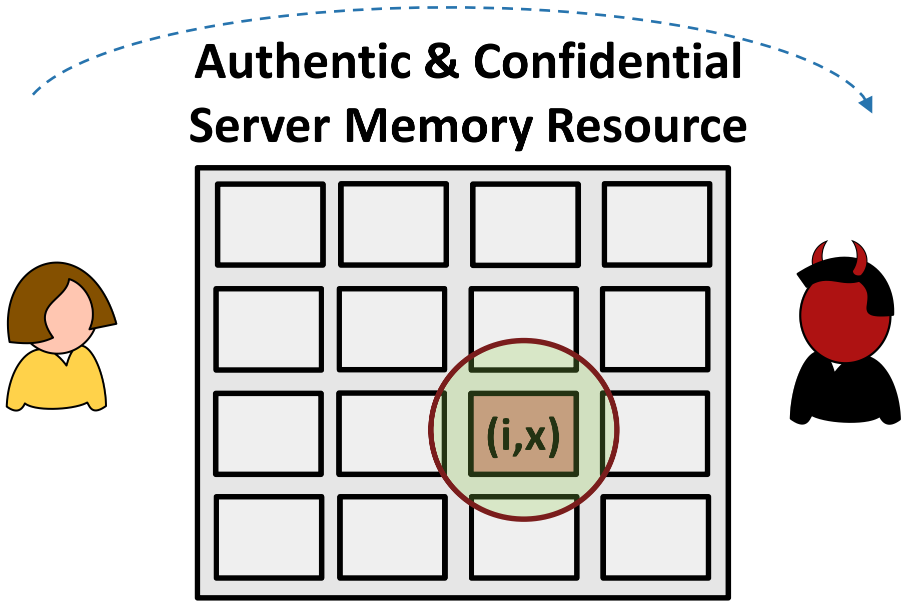
The Issue with Side-Channels

Access 1: Fail
Access 2: OK
…



3 Assume Bob learns which requests by Alice failed to retrieve a block.

The Issue with Side-Channels



4 **If** Alice's **protocol** allows Bob to **guess correctly with some bias**, then the **error pattern reveals information** on the access pattern!

Summary and Outlook

- We present a security model for outsourced storage following a modular approach building a hiearchy of storage resources.
- We show how to achieve each of the storage resources with concrete protocols.
- Our strongest notion provides a very high level of security and supports audits. Existing protocols often fail to provide this level of security.

RSAConference2018

NOW MATTERS

CRYPTOGRAPHY: SECURE STORAGE

Session-ID CRYP-R14

#RSAC

RSAConference2018

San Francisco | April 16 – 20 | Moscone Center

SESSION ID: CRYP-R14

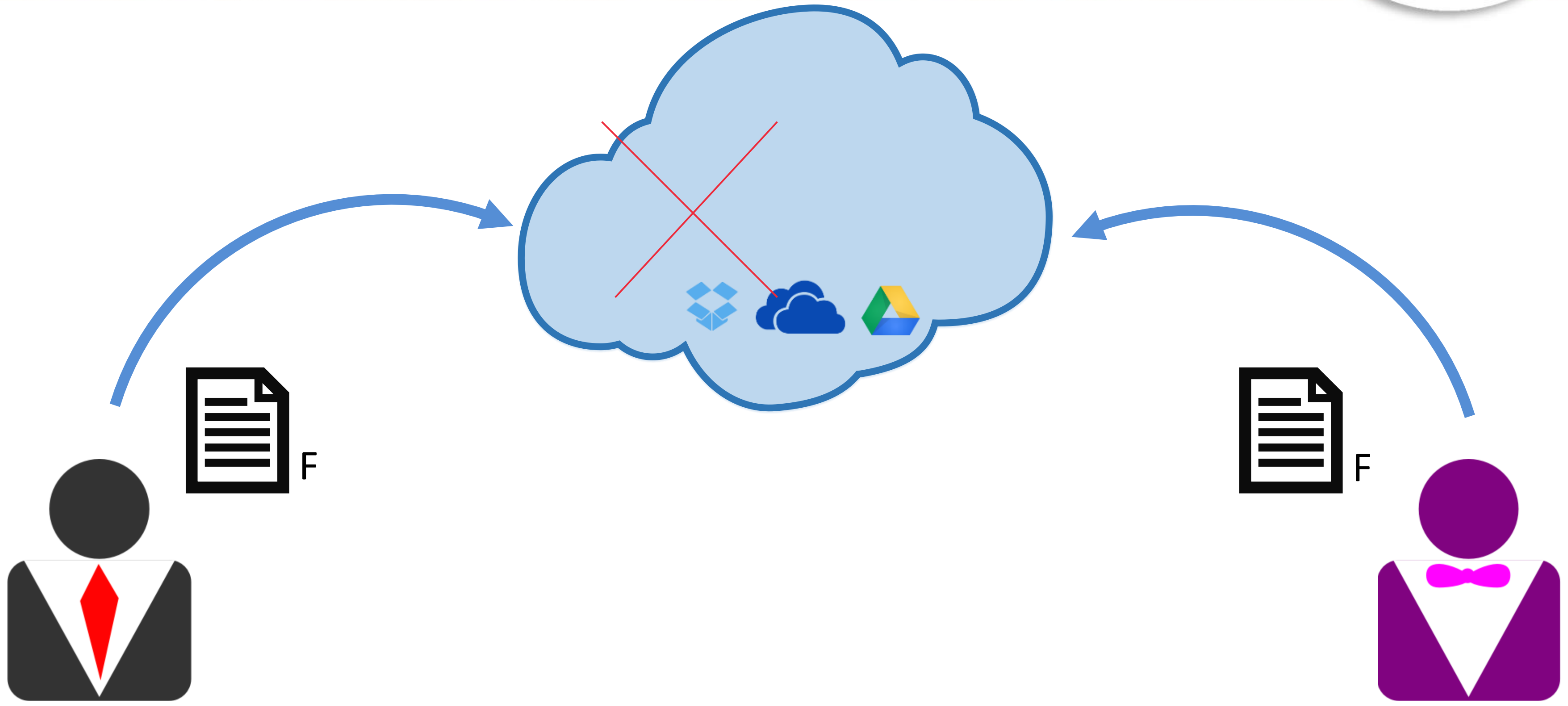
SECURE DEDUPLICATION OF ENCRYPTED DATA: REFINED MODEL AND NEW CONSTRUCTIONS

Jian Liu

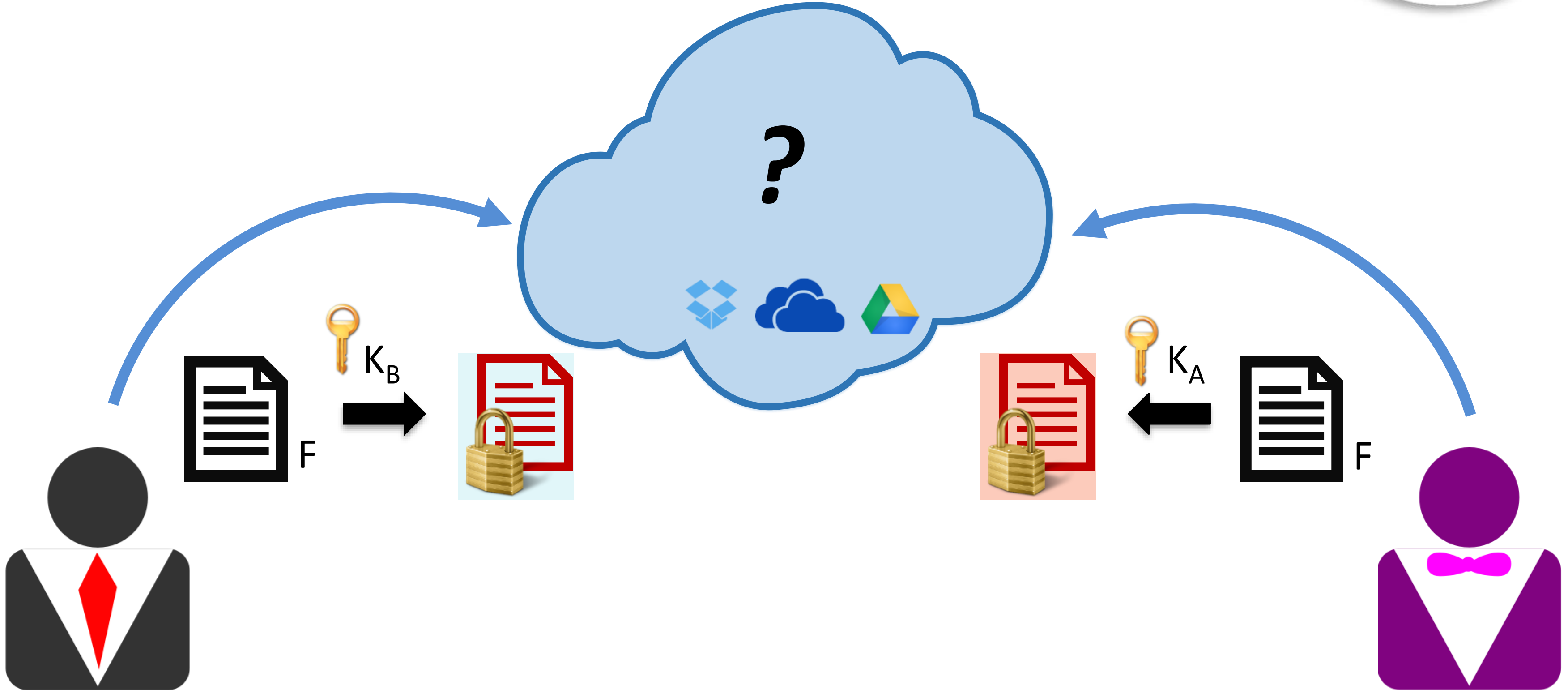
PhD Candidate
Aalto University

Cloud Storage

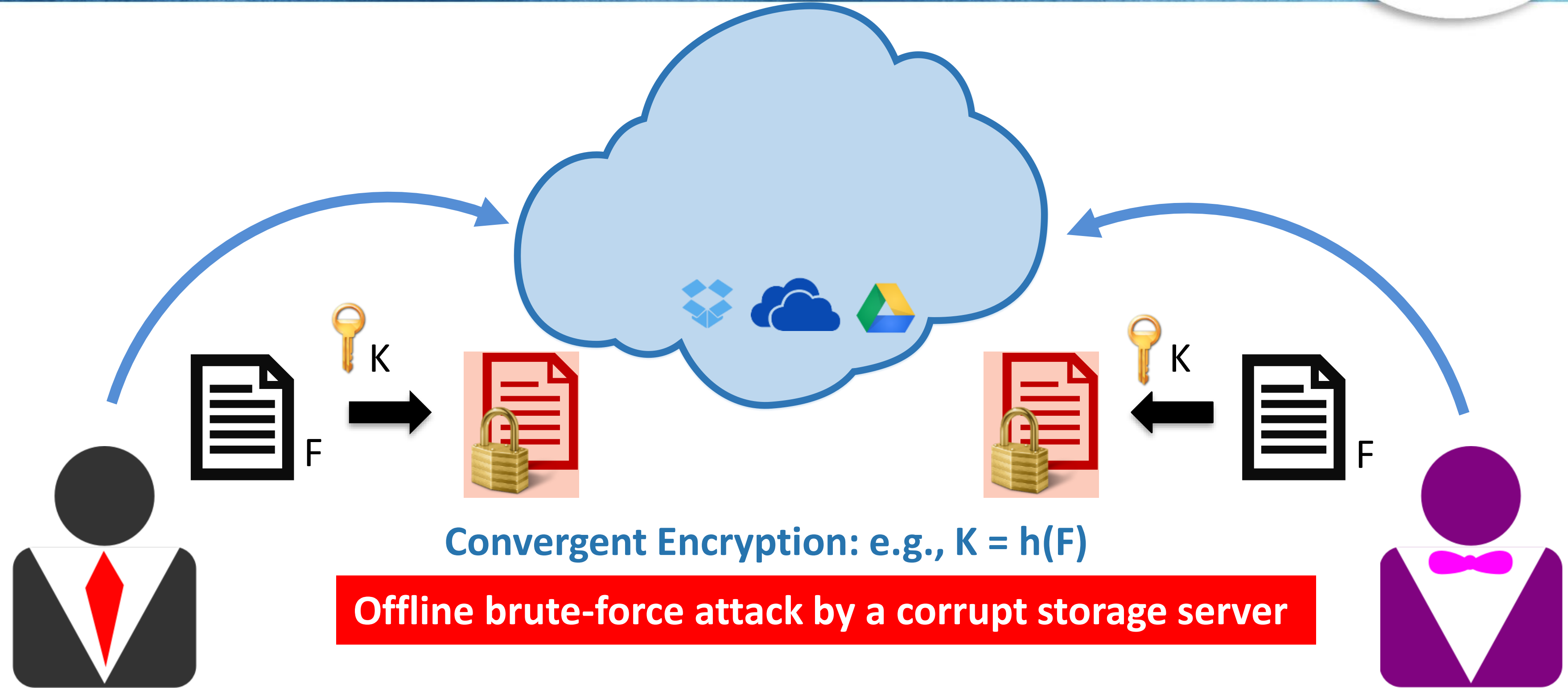
Deduplication



Secure Deduplication of Encrypted Data (SDoE)



Convergent encryption



Convergent Encryption: e.g., K = h(F)

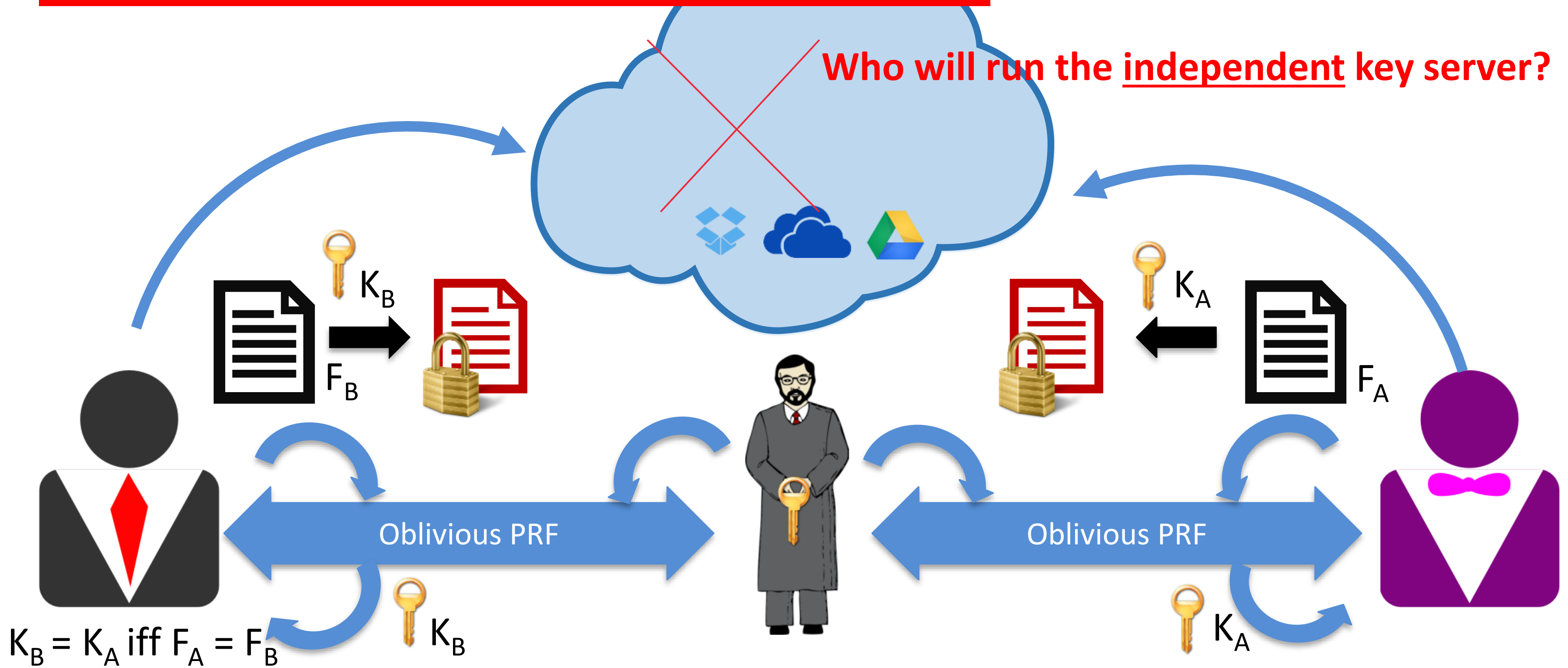
Offline brute-force attack by a corrupt storage server

J. R. Douceur, et al. Reclaiming space from duplicate files in a serverless distributed file system. In ICDCS' 02.

DupLESS: Independent Key Server

Online brute-force attack by a corrupt storage server

Who will run the <u>independent</u> key server?

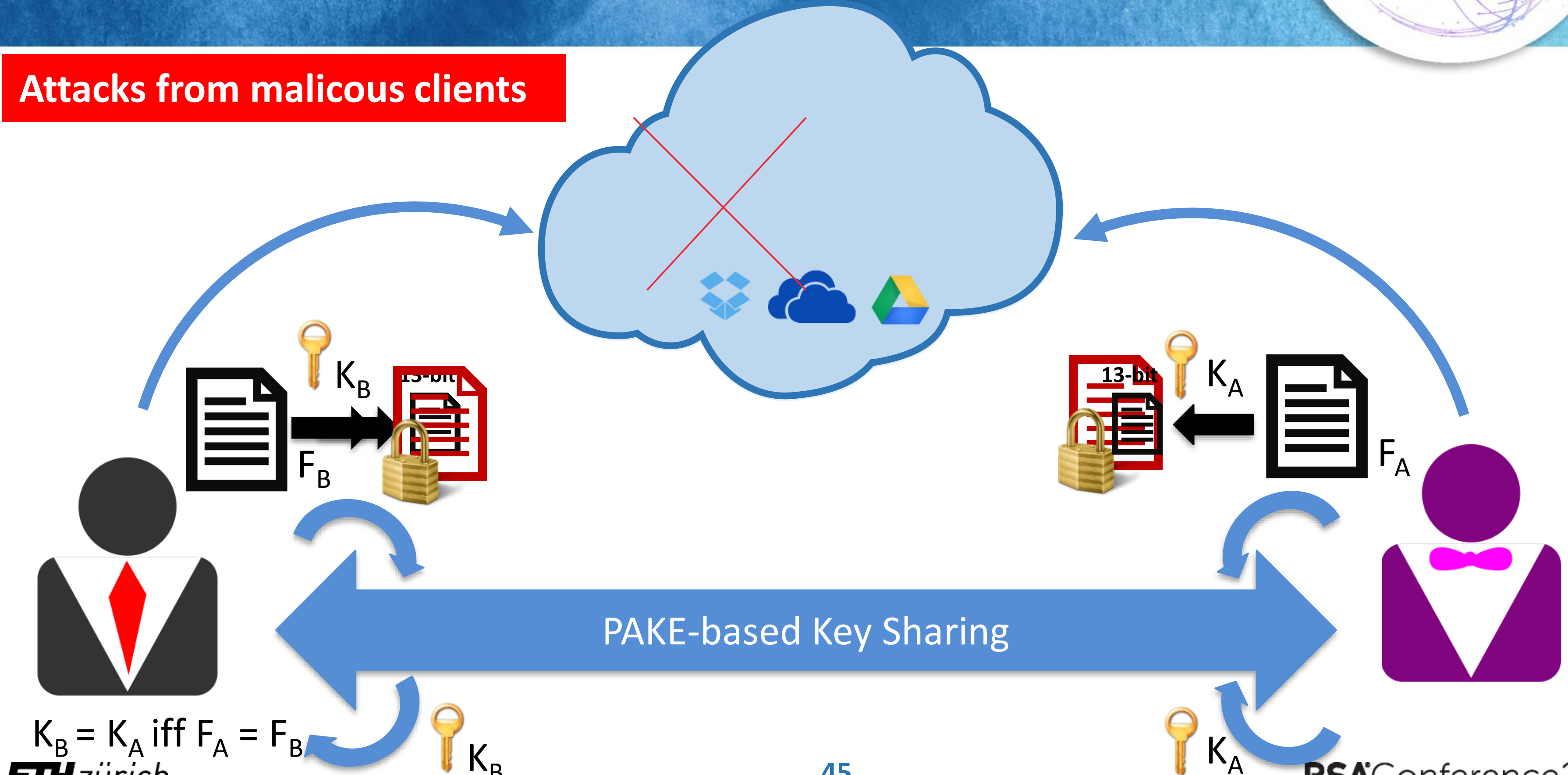


$K_B = K_A$ iff $F_A = F_B$

M. Bellare, S. Keelveedhi, and T. Ristenpart. DupLESS: server-aided encryption for deduplicated storage. USENIX' 13

PAKE-based SDoE

Attacks from malicous clients

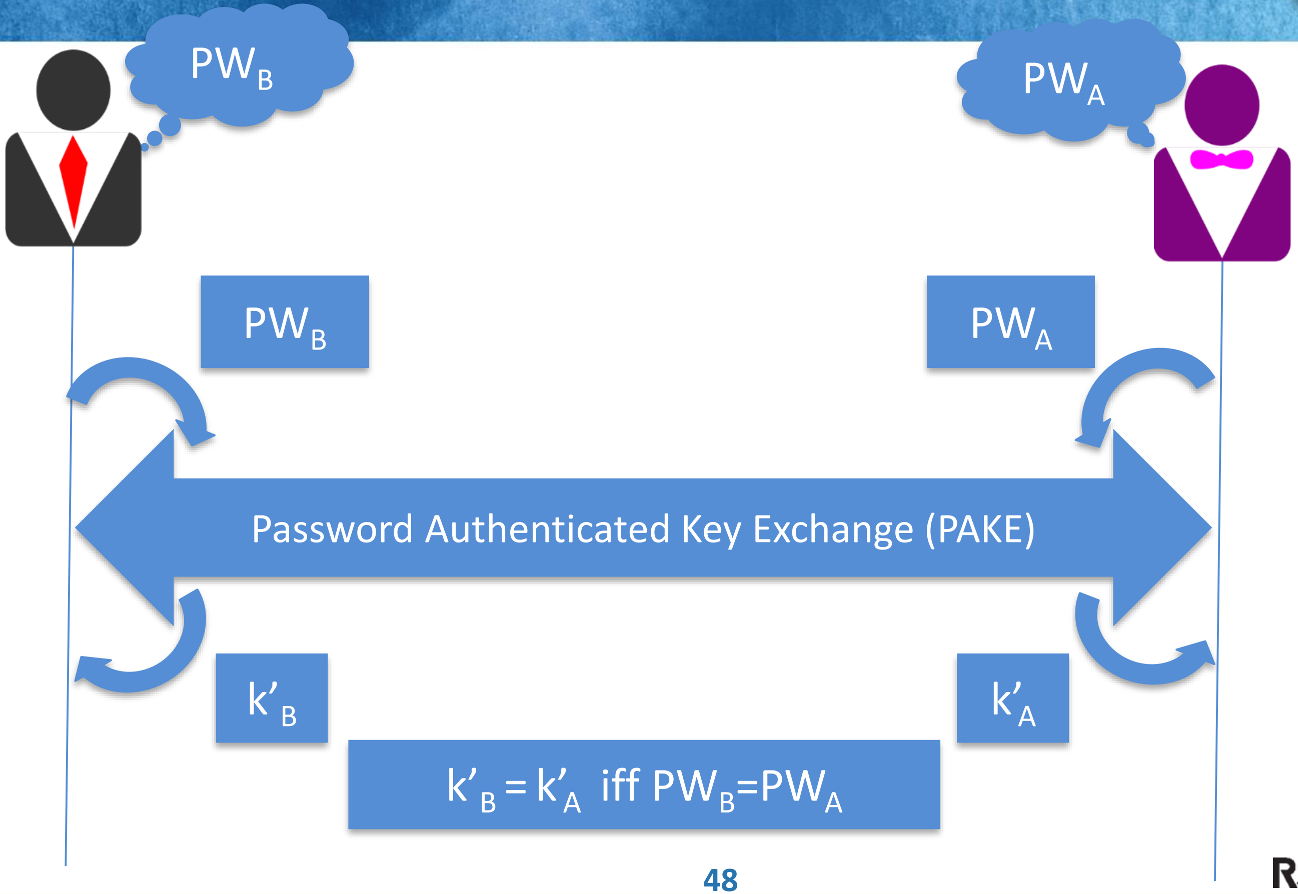


$K_B = K_A$ iff $F_A = F_B$

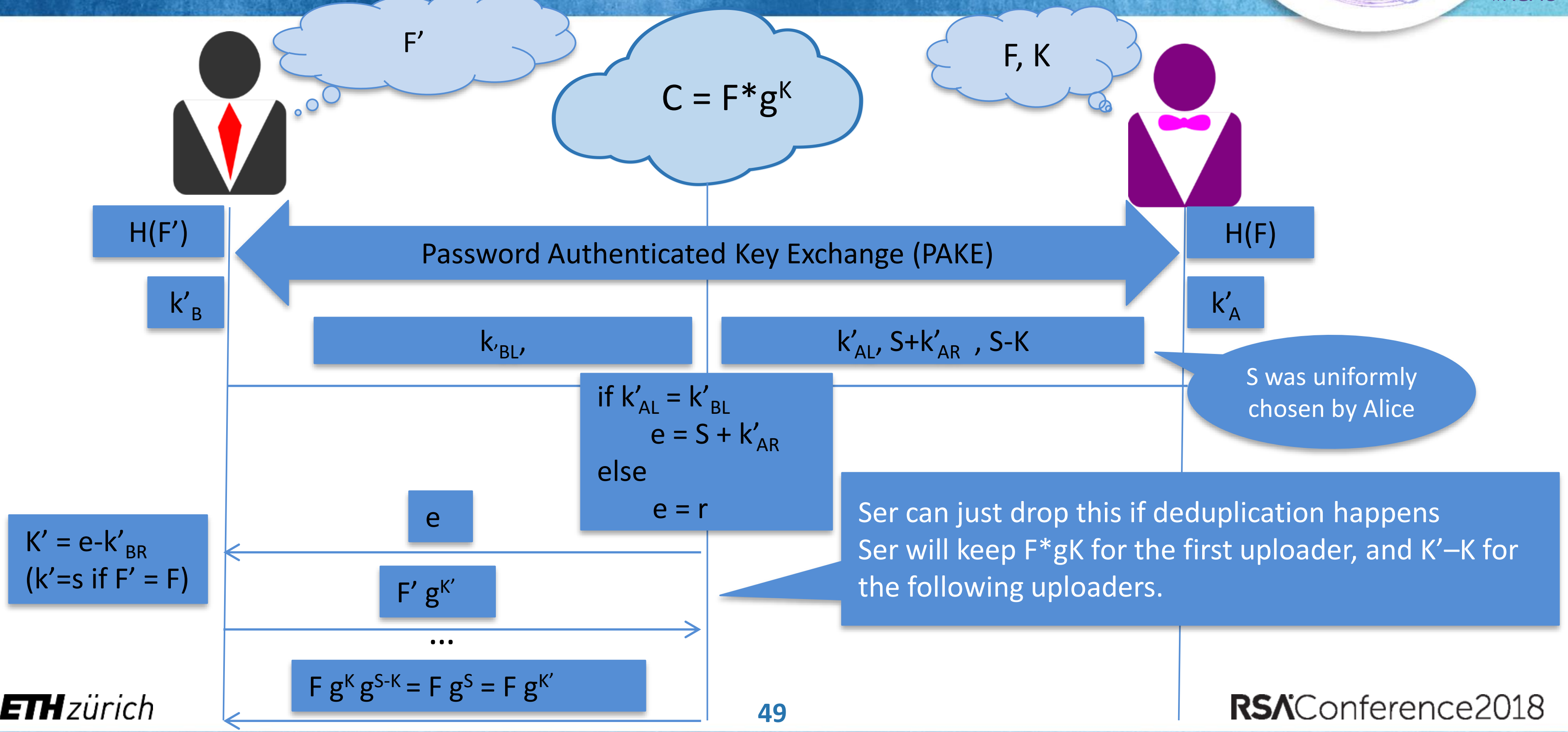
Contributions

- Formal security model for SDoE
- Two single-server SDoE that are provable secure
- Realistic simulations

Password Authenticated Key Exchange (PAKE)


PW_B
PW_A
PW_B
PW_A
Password Authenticated Key Exchange (PAKE)
k'_B
k'_A
$k'_B = k'_A$ iff $PW_B=PW_A$

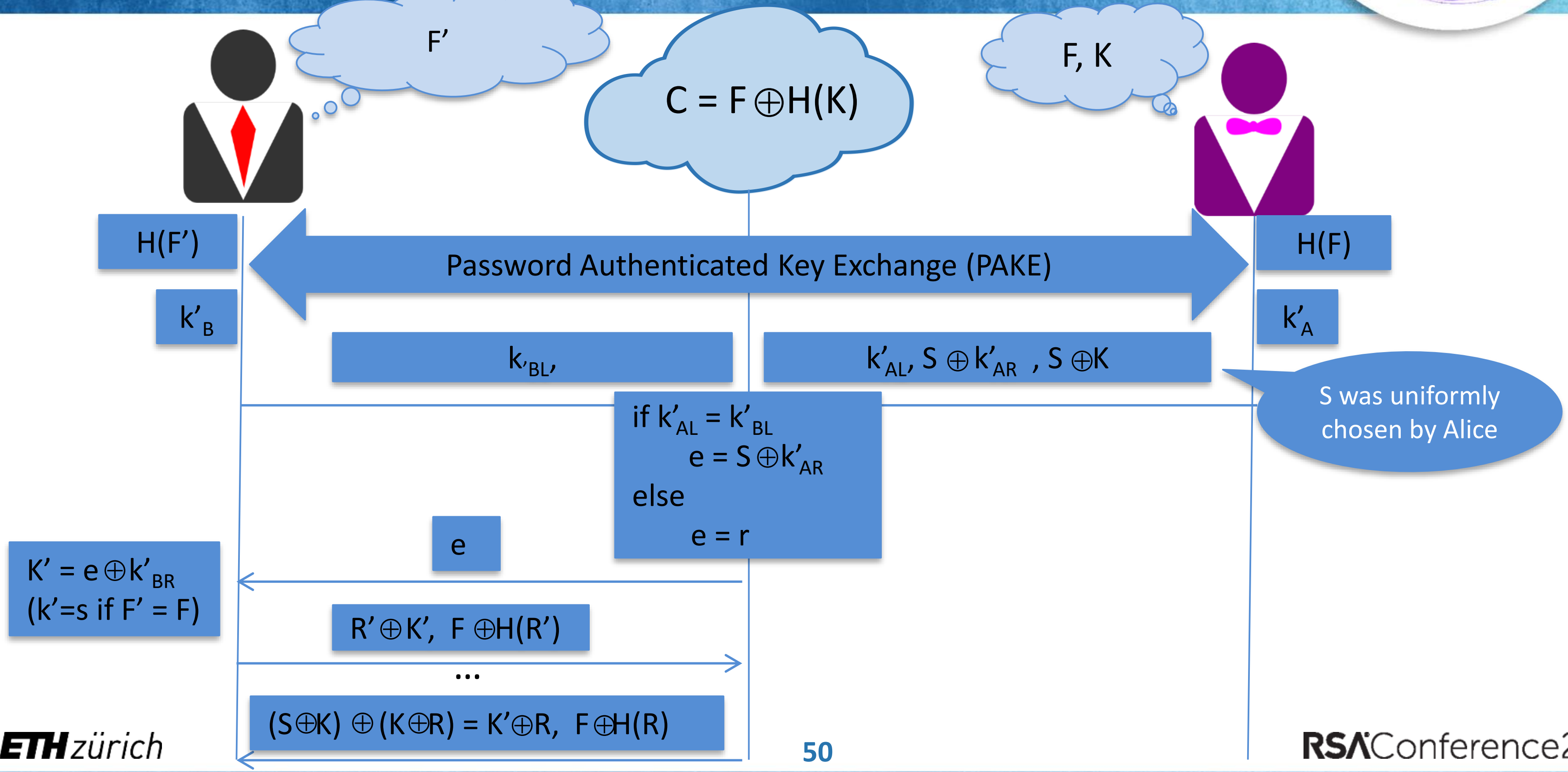
SDoE (1)



S was uniformly chosen by Alice

Ser can just drop this if deduplication happens
Ser will keep F*gK for the first uploader, and K'–K for the following uploaders.

SDoE (2)

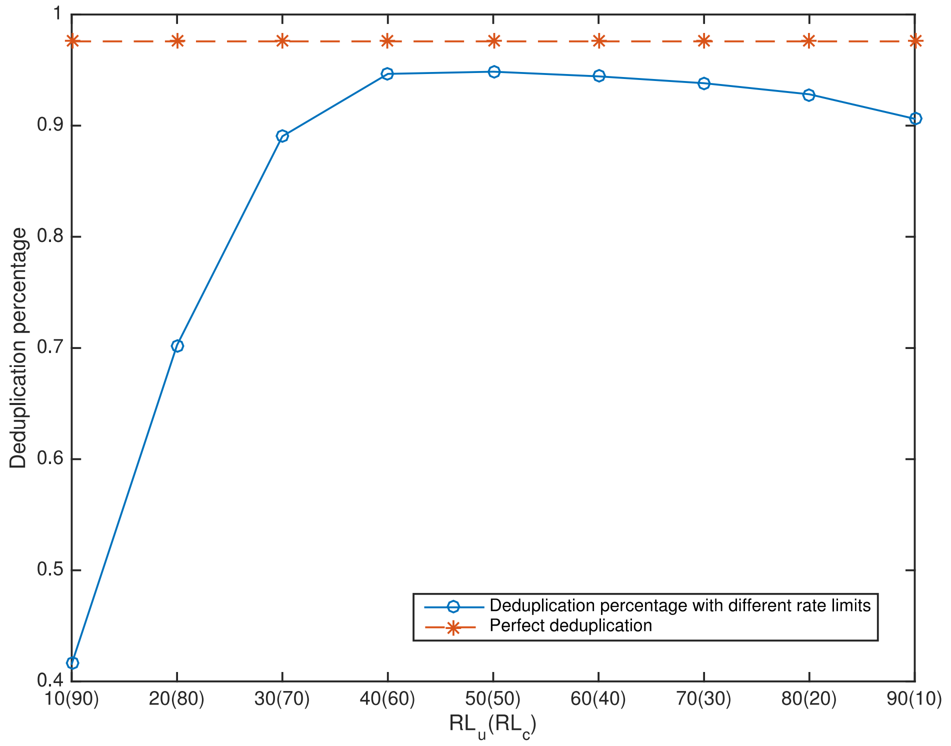


F'
$C = F \oplus H(K)$
F, K
$H(F')$
$H(F)$
Password Authenticated Key Exchange (PAKE)
k'_B
k'_A
$k_{'BL'}$
$k'_{AL}, S \oplus k'_{AR}\ , S \oplus K$
S was uniformly chosen by Alice
if $k'_{AL} = k'_{BL}$
$e = S \oplus k'_{AR}$
else
$e = r$
e
$K' = e \oplus k'_{BR}$
$(k'=s \text{ if } F' = F)$
$R' \oplus K',\ F \oplus H(R')$
…
$(S \oplus K) \oplus (K \oplus R) = K' \oplus R,\ F \oplus H(R)$

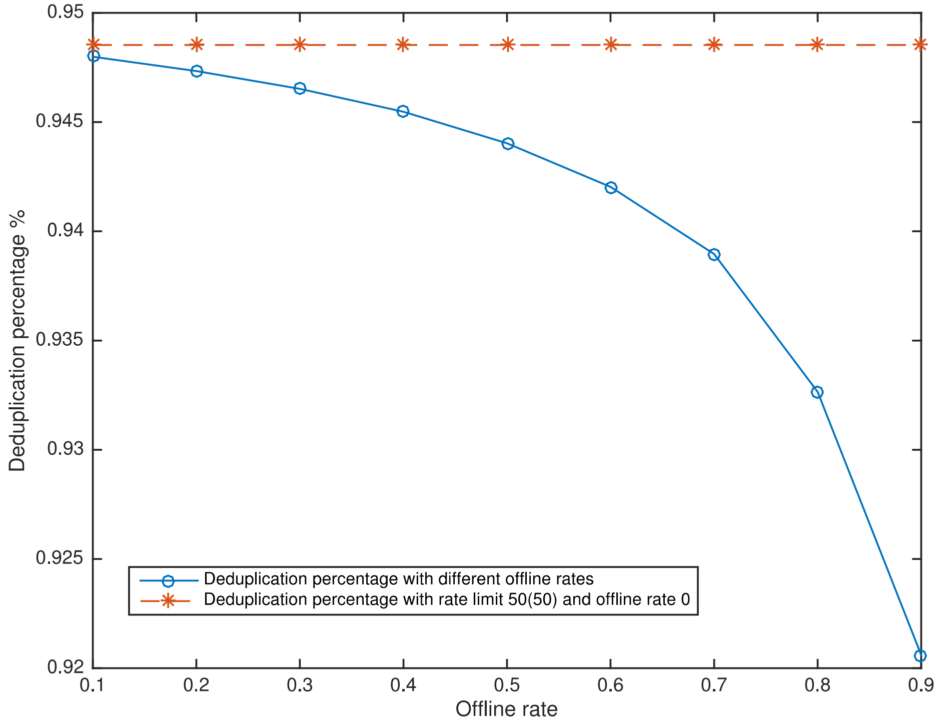
Simulation - dataset

- Android application popularity: 7396235 uploads, 178396 distinct
- Extend 5x by Synthetic Minority Over-sampling Technique (SMOTE)
- Model the real-world upload stream
  - Assuming the upload requests of a single file follows normal distribution $N(m,s^2)$
  - The number of copies of a file uploaded at time point *t* is $y_i = \frac{1}{s_i\sqrt{2p}} e^{-\frac{(t-u_i)^2}{2s_i^2}} x_i$
  - The total number of files uploaded at time point *t* is

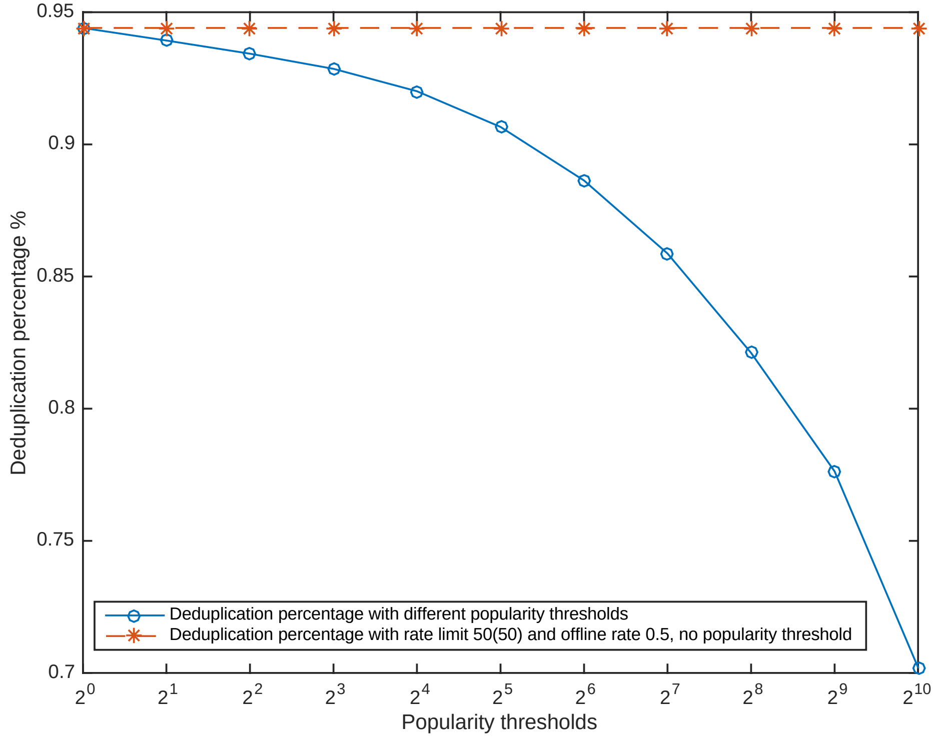
Simulation – Rate Limiting



Simulation – Offline Rate



Simulation – Popularity threshold



Q & A