A Formal Treatment of End-to-End Encrypted Cloud Storage

Matilda Backendal[1], Hannah Davis[2], Felix Günther[3], Miro Haller[4], Kenny Paterson[1]

[1]ETH Zurich , [2]Seagate Technology, [3]IBM Research Zurich, [4]UC San Diego

Trail of Bits, September 11, 2024

Cloud Storage

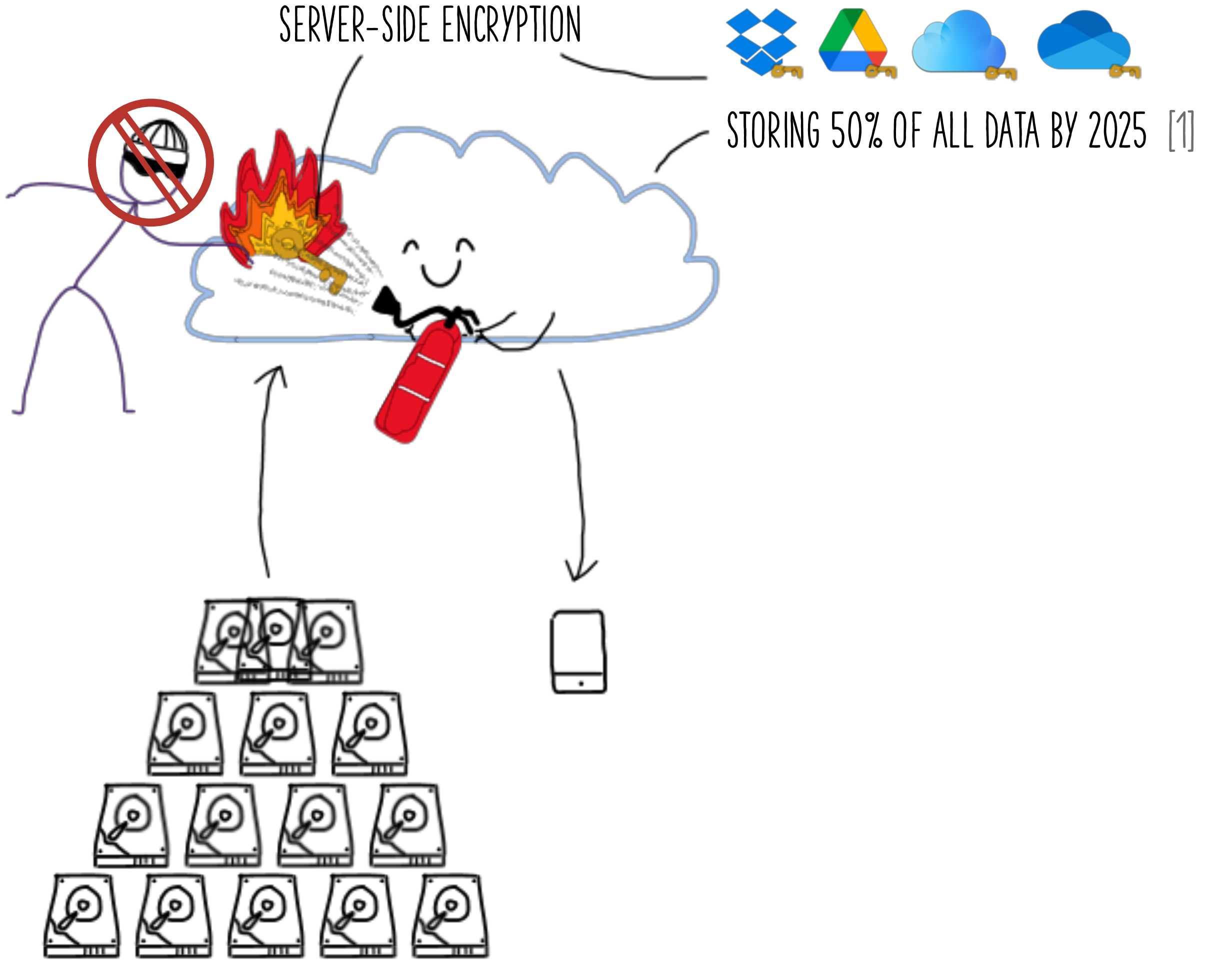
Benefits:

- + Availability
- + Redundancy
- + Scalability

Concerns:

- - Data leaks to third party

=> SERVER-SIDE ENCRYPTION



[1] https://cybersecurityventures.com/the-world-will-store-200-zettabytes-of-data-by-2025/ (Sausalito, Calif., Feb. 1, 2024)

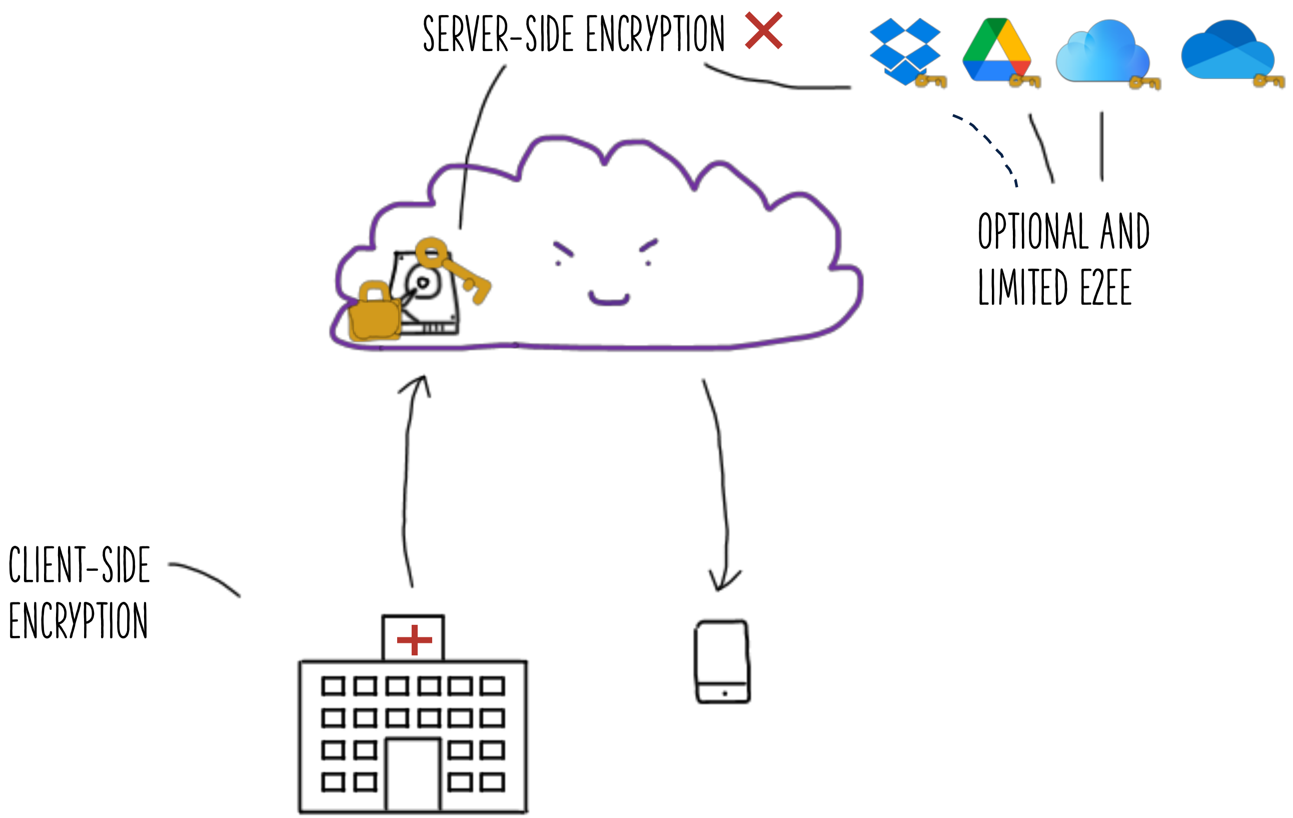
Cloud Storage

Benefits:

- + Availability
- + Redundancy
- + Scalability

Concerns:

- Data leaks to third party
  - => SERVER-SIDE ENCRYPTION
- Malicious server
  - => END-TO-END ENCRYPTION



https://www.hipaajournal.com/healthcare-cloud-usage-grows-but-protecting-phi-can-be-a-challenge/

E2EE Cloud Storage

"WITH **MEGA**, YOU CONTROL THE ENCRYPTION"

300 MILLION USERS

MEGA

INSECURE!

[SP:BHP23]
[EC:AHMP23]

AMNESTY INTERNATIONAL, THE GERMAN FEDERAL GOVERNMENT & ETH

"ULTIMATE SECURITY"

Nextcloud

INSECURE!

[EuroSP:ABCP23]

"EXCEPTIONALLY PRIVATE CLOUD"

sync.com

pCloud

"EUROPE'S MOST SECURE CLOUD STORAGE"

"THE STRONGEST ENCRYPTED CLOUD STORAGE IN THE WORLD"

icedrive

Seafile™

"SUPPORTS CLIENT-SIDE END-TO-END ENCRYPTION"

INSECURE!

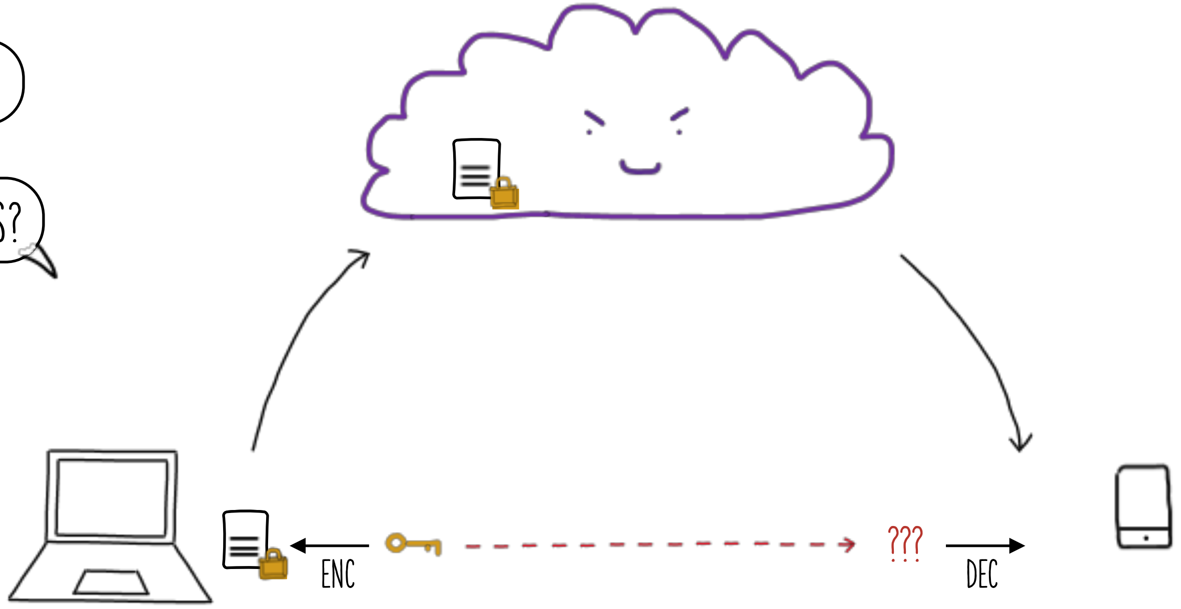
[CCS:TH24]

Why Is It Hard?

JUST USE YOUR FAVORITE AEAD SCHEME FOR CLIENT-SIDE ENCRYPTION!

HOW DO YOU TRANSFER KEYS BETWEEN DEVICES?

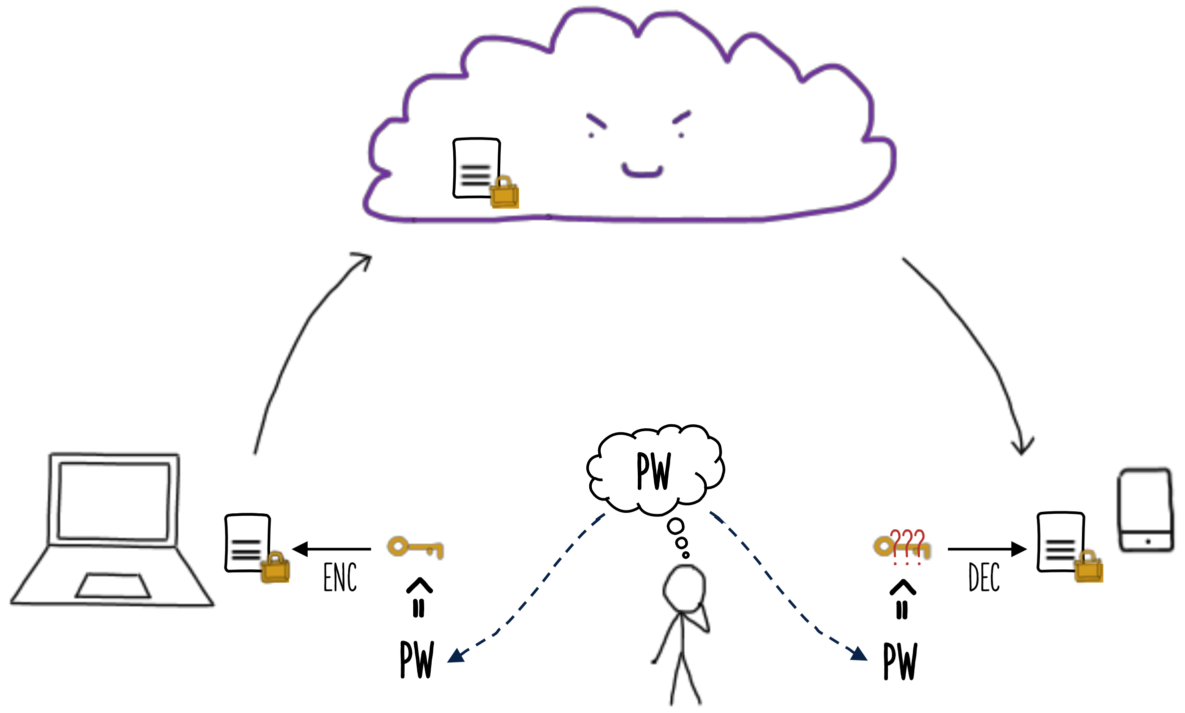
1 key distribution



Why Is It Hard?

DERIVE KEYS FROM THE PASSWORD!

1	key distribution
2	password-based security



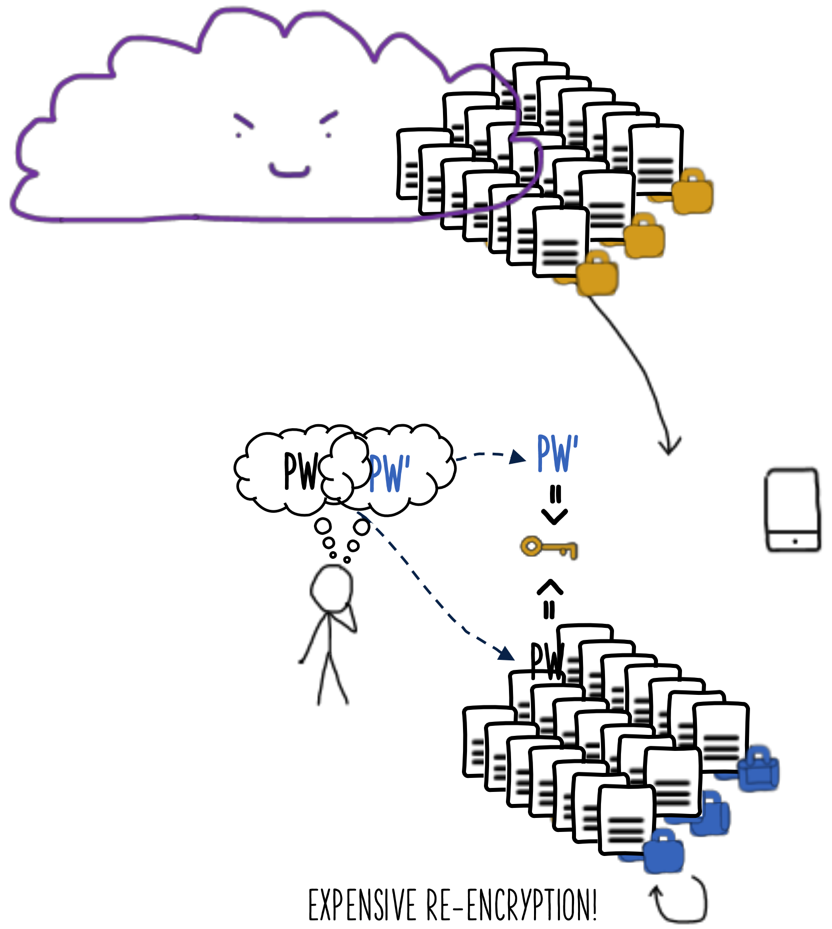
Why Is It Hard?

PROBLEM 1: PW CHANGE

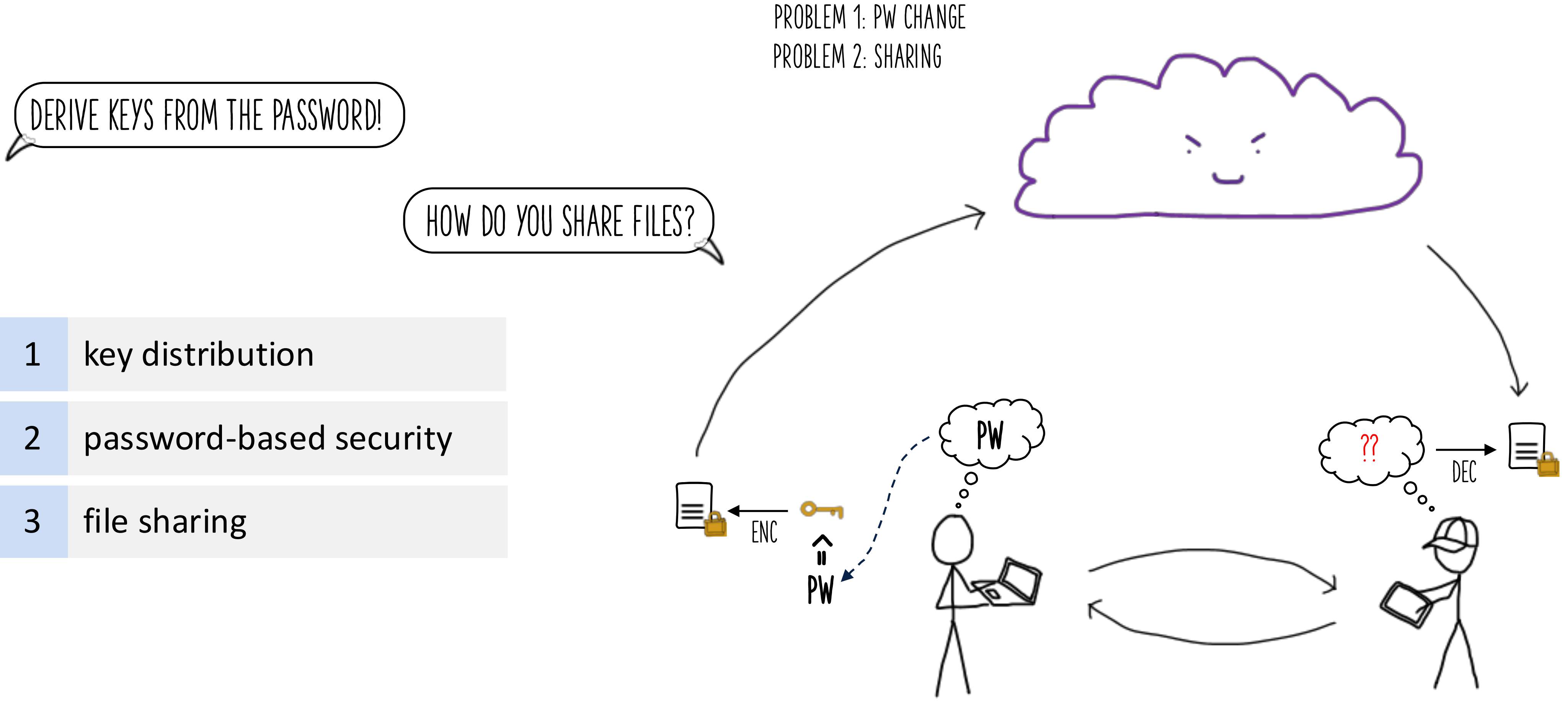
DERIVE KEYS FROM THE PASSWORD!

WHAT IF THE PASSWORD CHANGES?

1. key distribution
2. password-based security



Why Is It Hard?

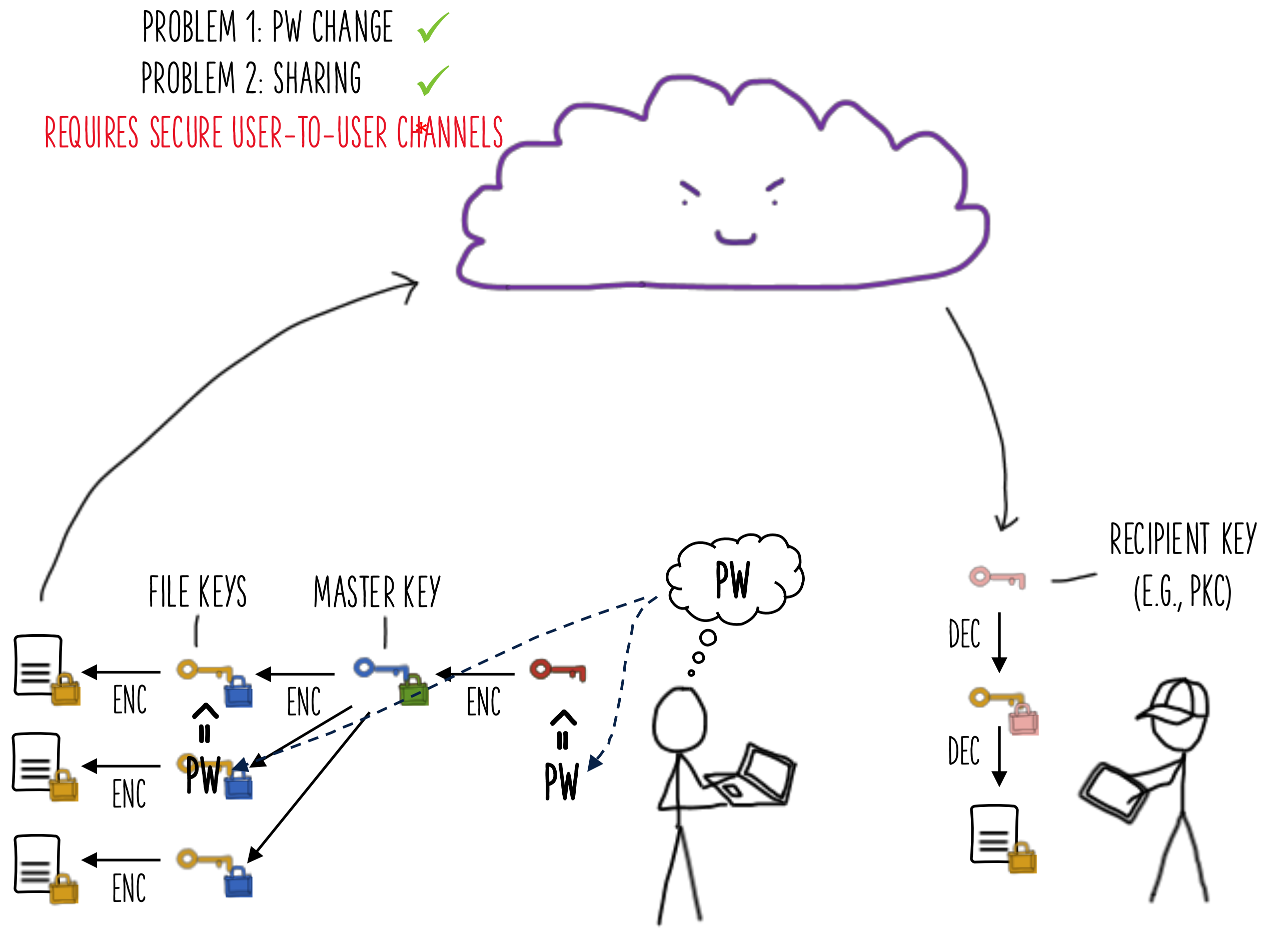


1	key distribution
2	password-based security
3	file sharing

Why Is It Hard?

BUILD A KEY HIERARCHY!

1	key distribution
2	password-based security
3	file sharing

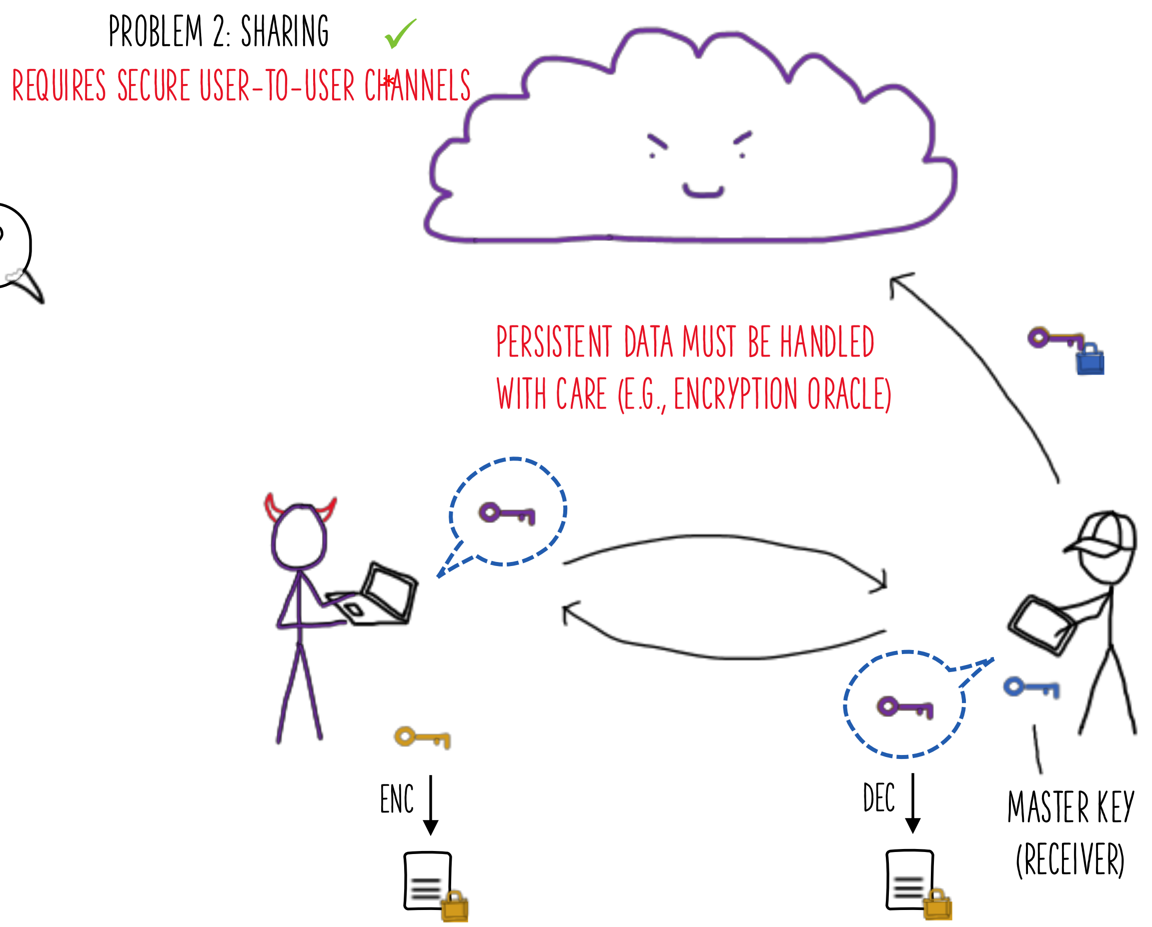


Why Is It Hard?

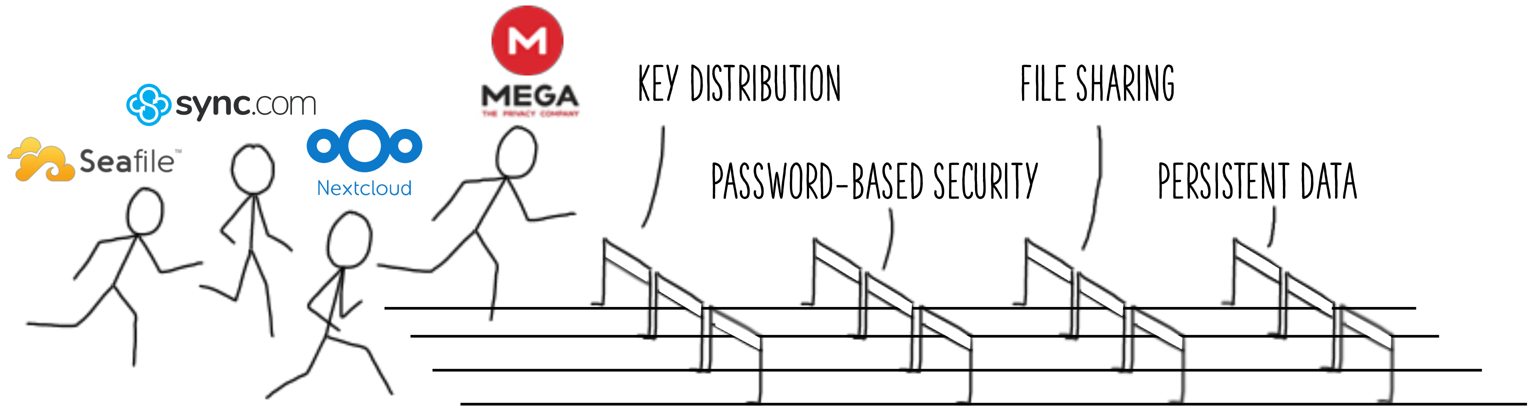
USE SECURE MESSAGING TECHNIQUES!

HOW TO PROTECT DATA AT REST?

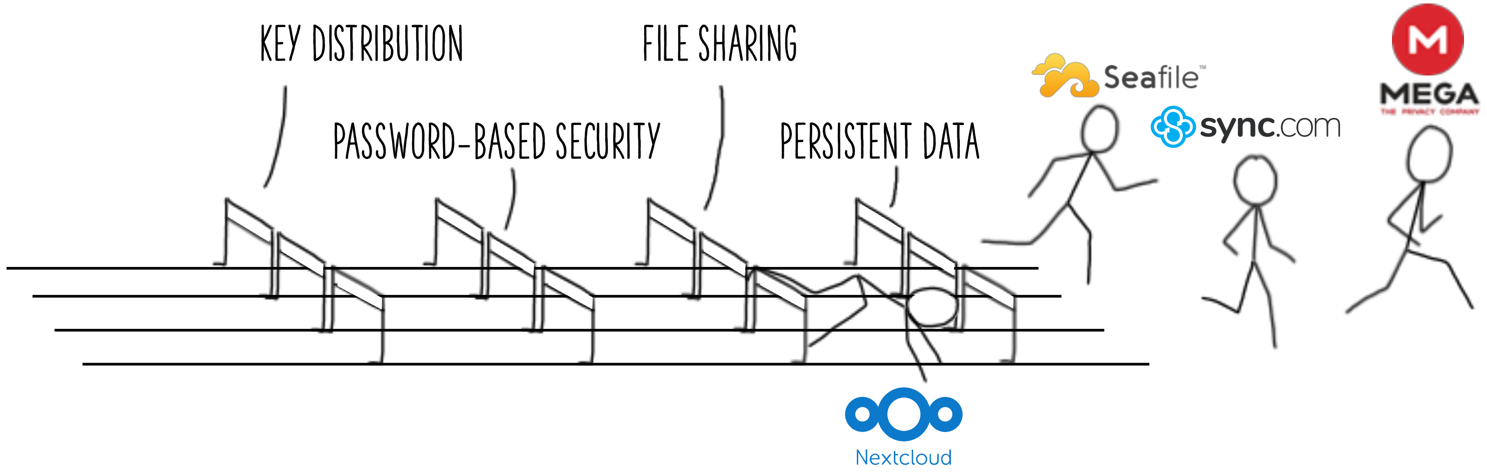
1	key distribution
2	password-based security
3	file sharing
4	persistent data


PROBLEM 2: SHARING ✓
REQUIRES SECURE USER-TO-USER CHANNELS
PERSISTENT DATA MUST BE HANDLED WITH CARE (E.G., ENCRYPTION ORACLE)
ENC
DEC
MASTER KEY (RECEIVER)

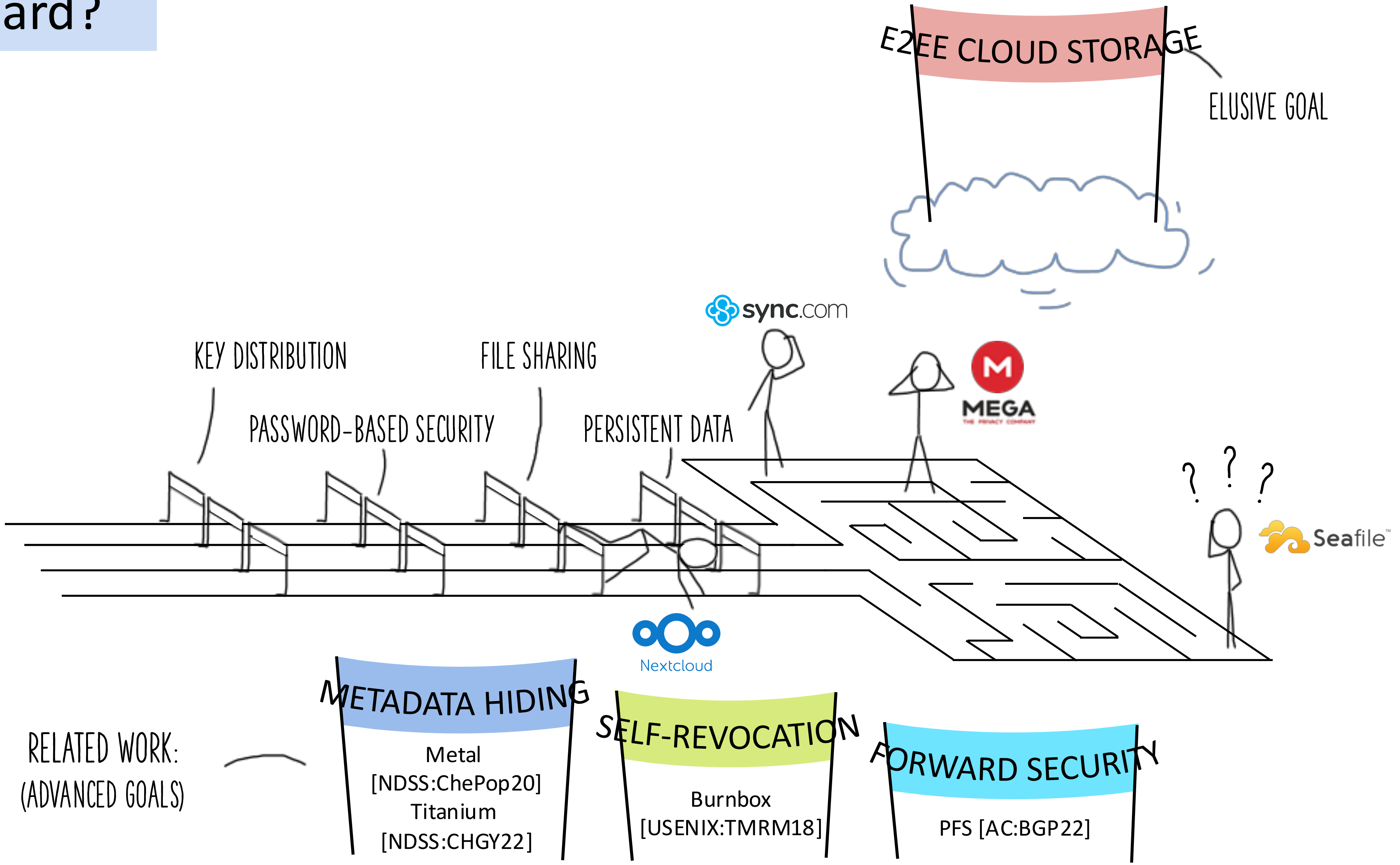
Why Is It Hard?



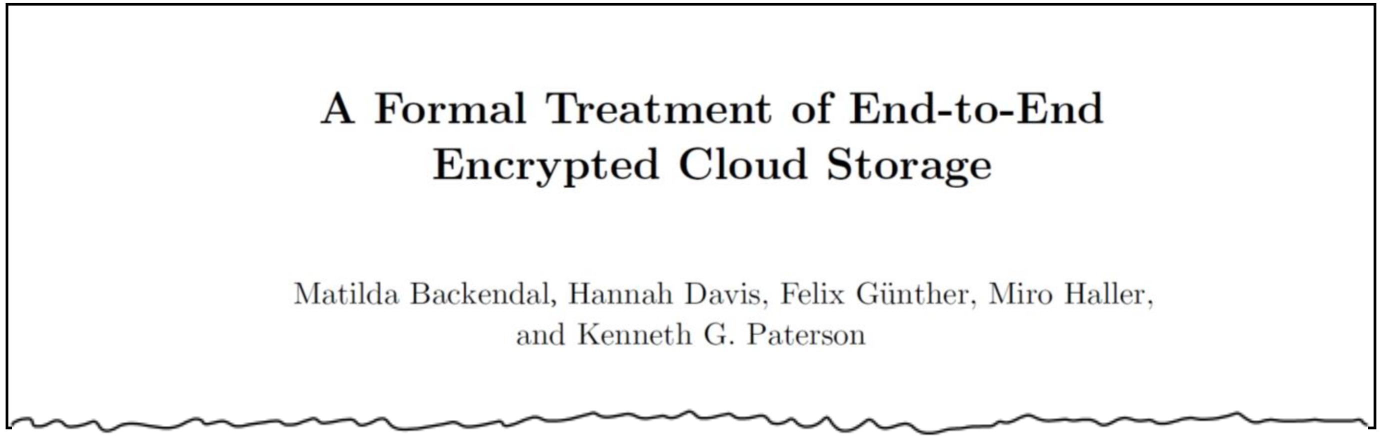
Why Is It Hard?



Why Is It Hard?



Contributions



A Formal Treatment of End-to-End Encrypted Cloud Storage

Matilda Backendal, Hannah Davis, Felix Günther, Miro Haller, and Kenneth G. Paterson

1 Formal Model

- Syntax
- Security games

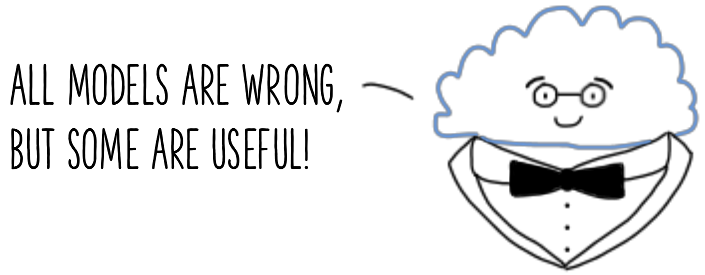
2 Construction

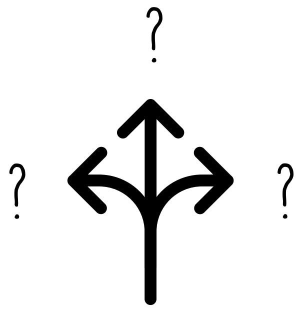
- CSS (Cloud Storage Scheme)
- Security proofs

1. Formalizing E2EE Cloud Storage

Formalizing E2EE Cloud Storage

Model Goals





Capture existing systems	Capture *real-world* systems	Capture future systems
1 Expressive	2 Faithful	3 Generic

Syntax

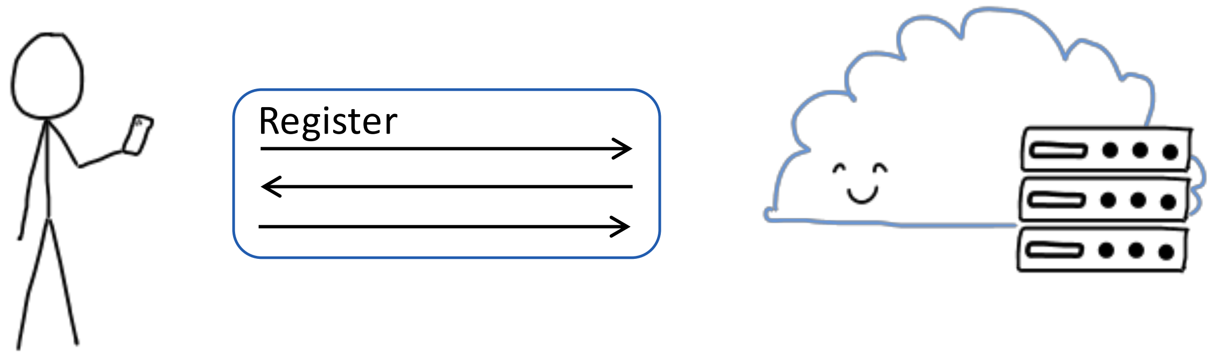
WHAT MAKES A CLOUD STORAGE A CLOUD STORAGE?

Core Functionality

- Register (create account)
- Authenticate (log in)

- Put (upload a file)
- Update (modify content)
- Get (download)
- Share
- Accept (receive share)

$\Big\}$ INTERACTIVE PROTOCOLS



Syntax

HOW DO WE MAKE THE MODEL USEFUL?

Core Functionality

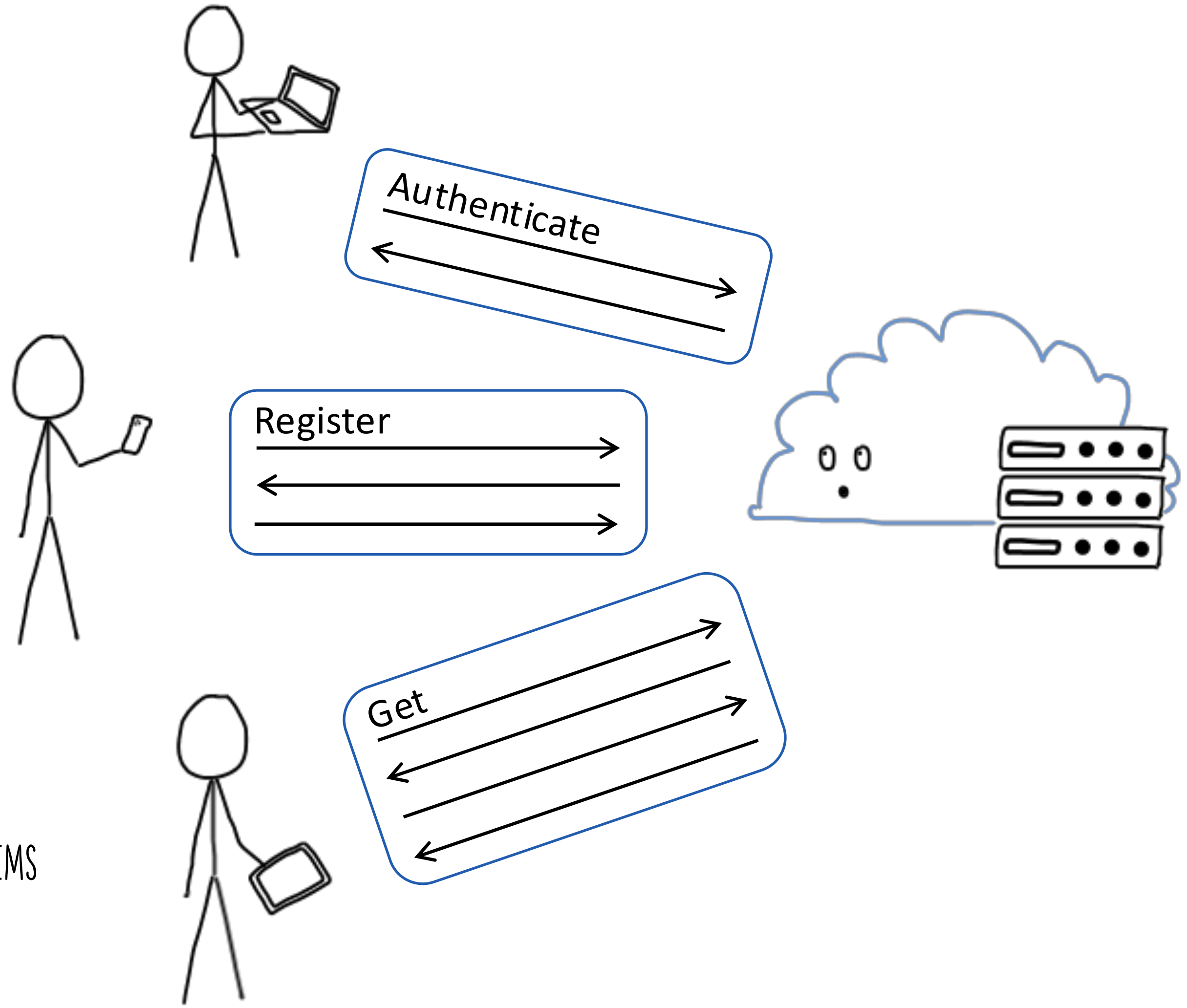
- Register (create account)
- Authenticate (log in)

- Put (upload a file)
- Update (modify content)
- Get (download)
- Share
- Accept (receive share)

$\}$ INTERACTIVE PROTOCOLS

Model Choices

- Non-atomic operations $\longrightarrow$ FAITHFUL TO REAL-WORLD SYSTEMS



Syntax

HOW DO WE MAKE THE MODEL USEFUL?

Core Functionality

- Register (create account)
- Authenticate (log in)

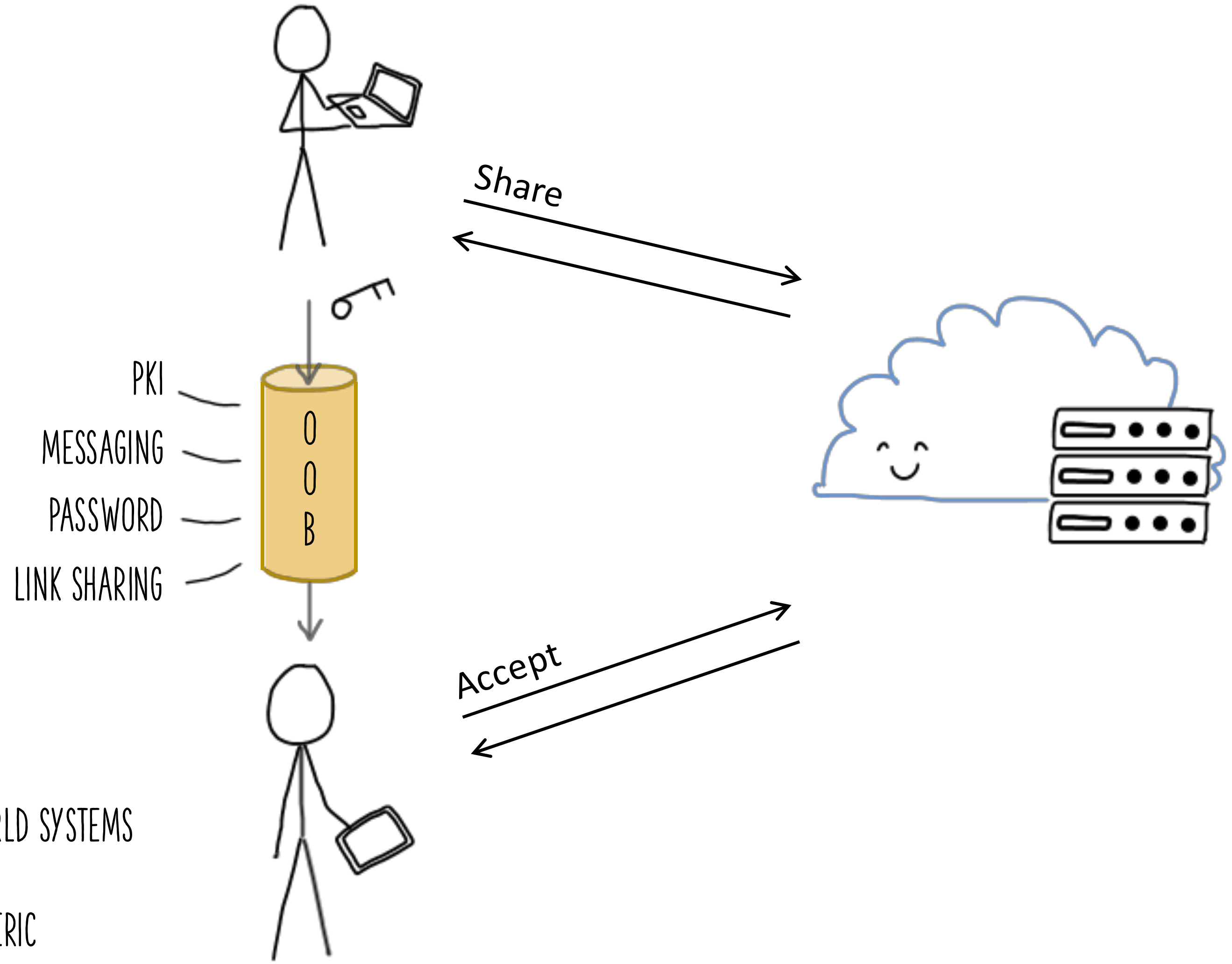
- Put (upload a file)
- Update (modify content)
- Get (download)
- Share
- Accept (receive share)

INTERACTIVE PROTOCOLS

OFTEN NOT CONSIDERED IN RELATED WORK

Model Choices

- Non-atomic operations $\longrightarrow$ FAITHFUL TO REAL-WORLD SYSTEMS
- Abstract OOB channel for sharing $\longrightarrow$ GENERIC



Security Notions

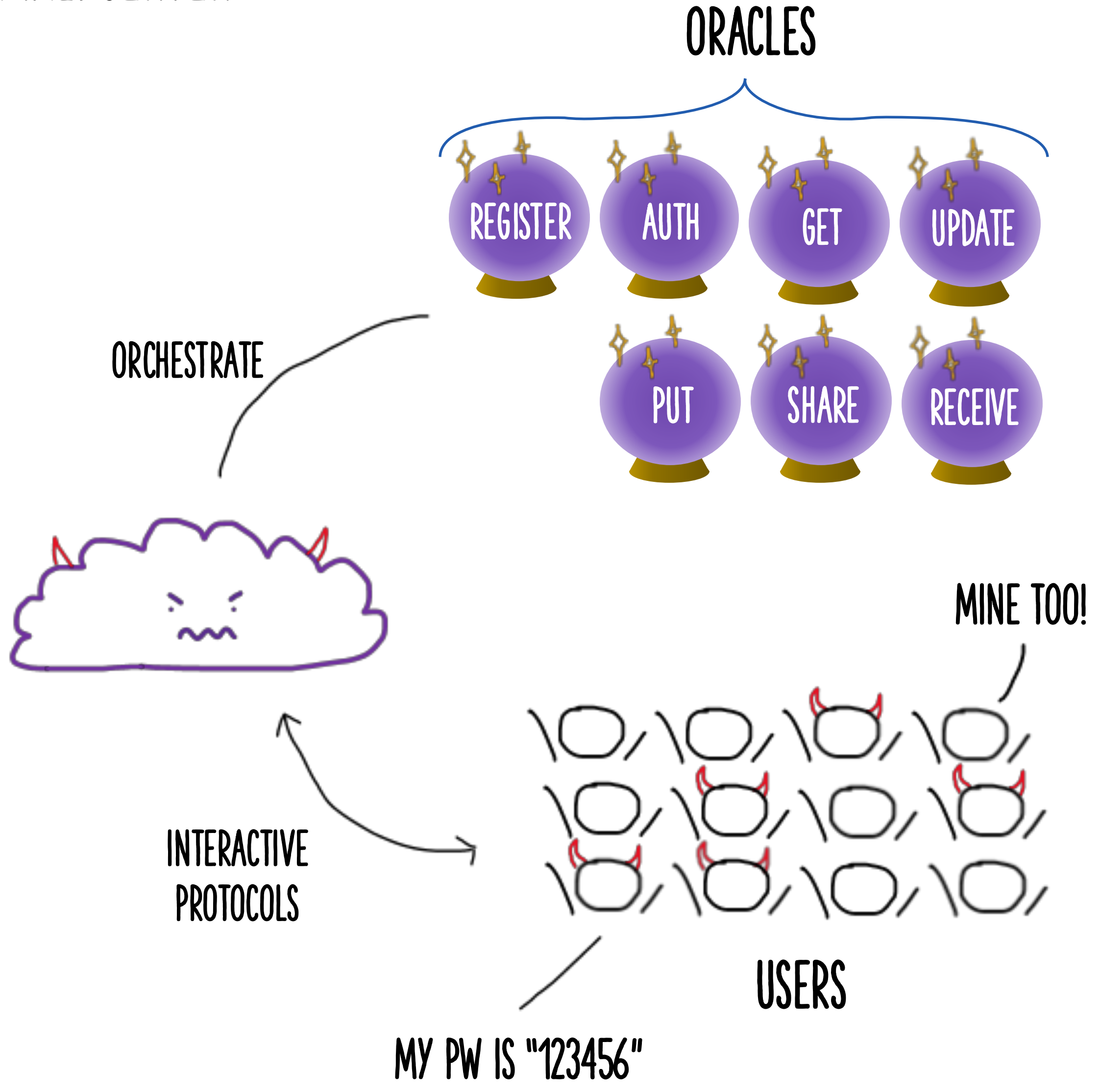
CLIENT-TO-CLIENT (C2C): MAL. SERVER

Threat model:

- Malicious cloud provider
- Full control over network & operations

Game mechanics:

- Correlated passwords
- Adversary can
  - Compromise users (adaptive/selective)
  - Control users (via oracles)
  - Control server (directly)

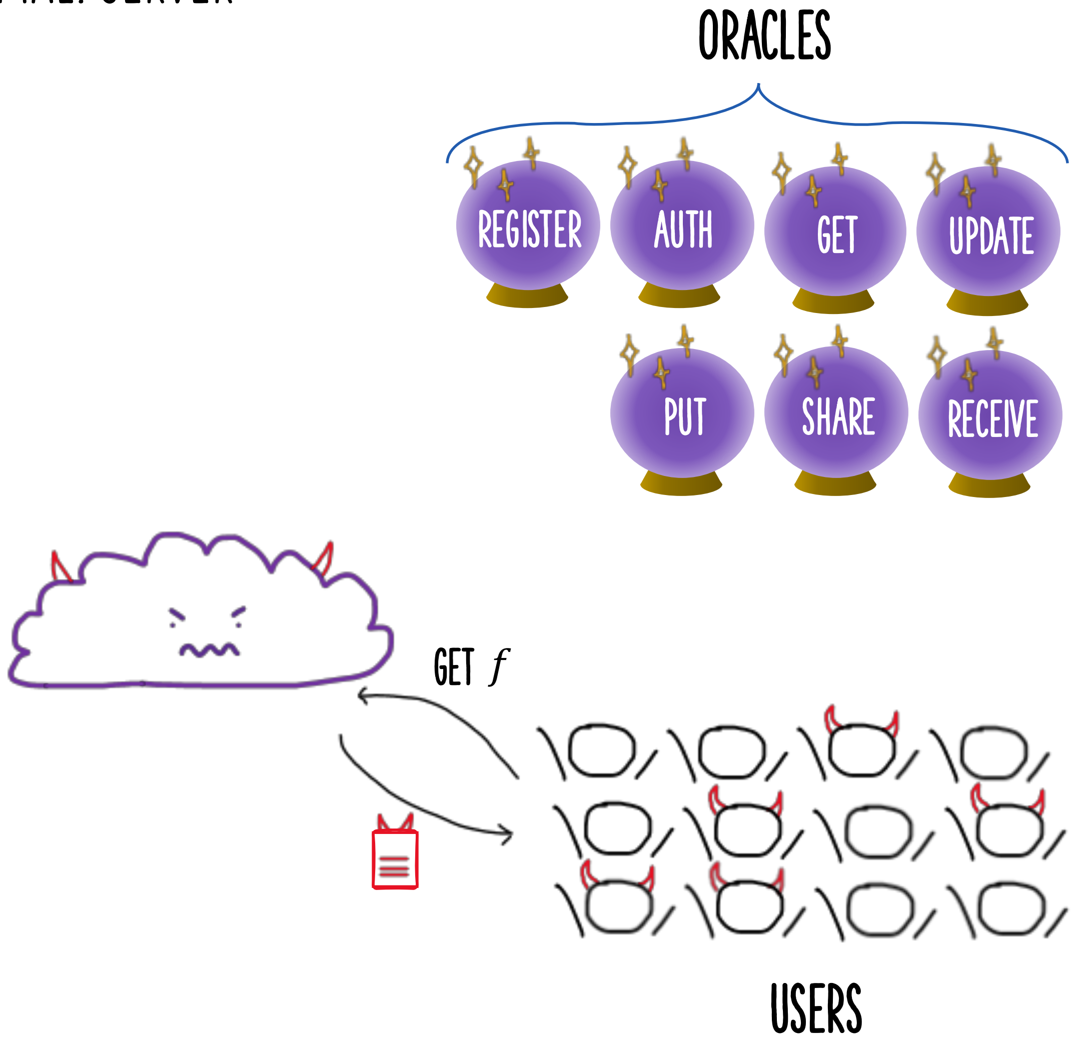


Security Notions

CLIENT-TO-CLIENT (C2C): MAL. SERVER

Integrity:

- Adversary simulates interaction
- Wins if it can, for an honest user,
  1. inject a file, or
  2. modify a file.



Security Notions

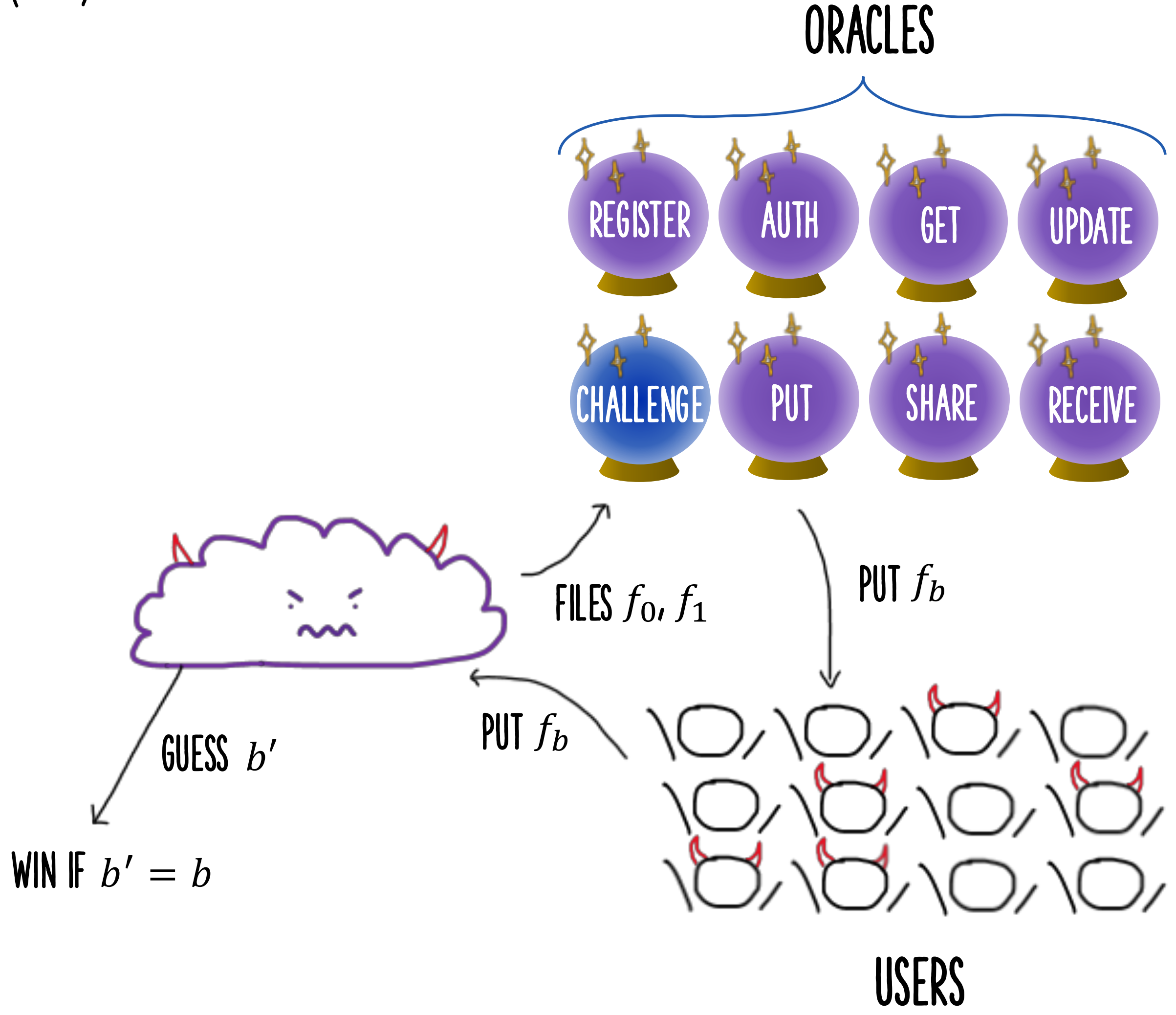
CLIENT-TO-CLIENT (C2C): MAL. SERVER

Integrity:

- Adversary simulates interaction
- Wins if it can, for an honest user,
  1. inject a file, or
  2. modify a file.

Confidentiality:

- Additional challenge oracle
  - Submit two files f_0, f_1
  - File f_b is uploaded
  - Guess bit b



Security Notions

PLAINTEXT VS. CIPHERTEXT

Integrity:

- Adversary simulates interaction
- Wins if it can, for an honest user,
  1. inject a file, or
  2. modify a file.

$\}$ NOT INT-CTXT

Confidentiality:

- Additional challenge oracle
  - Submit two files f_0, f_1
  - File f_b is uploaded
  - Guess bit b

$\}$ NOT IND$

No generic ciphertexts

$\hookrightarrow$ ALLOWS GENERIC SYNTAX

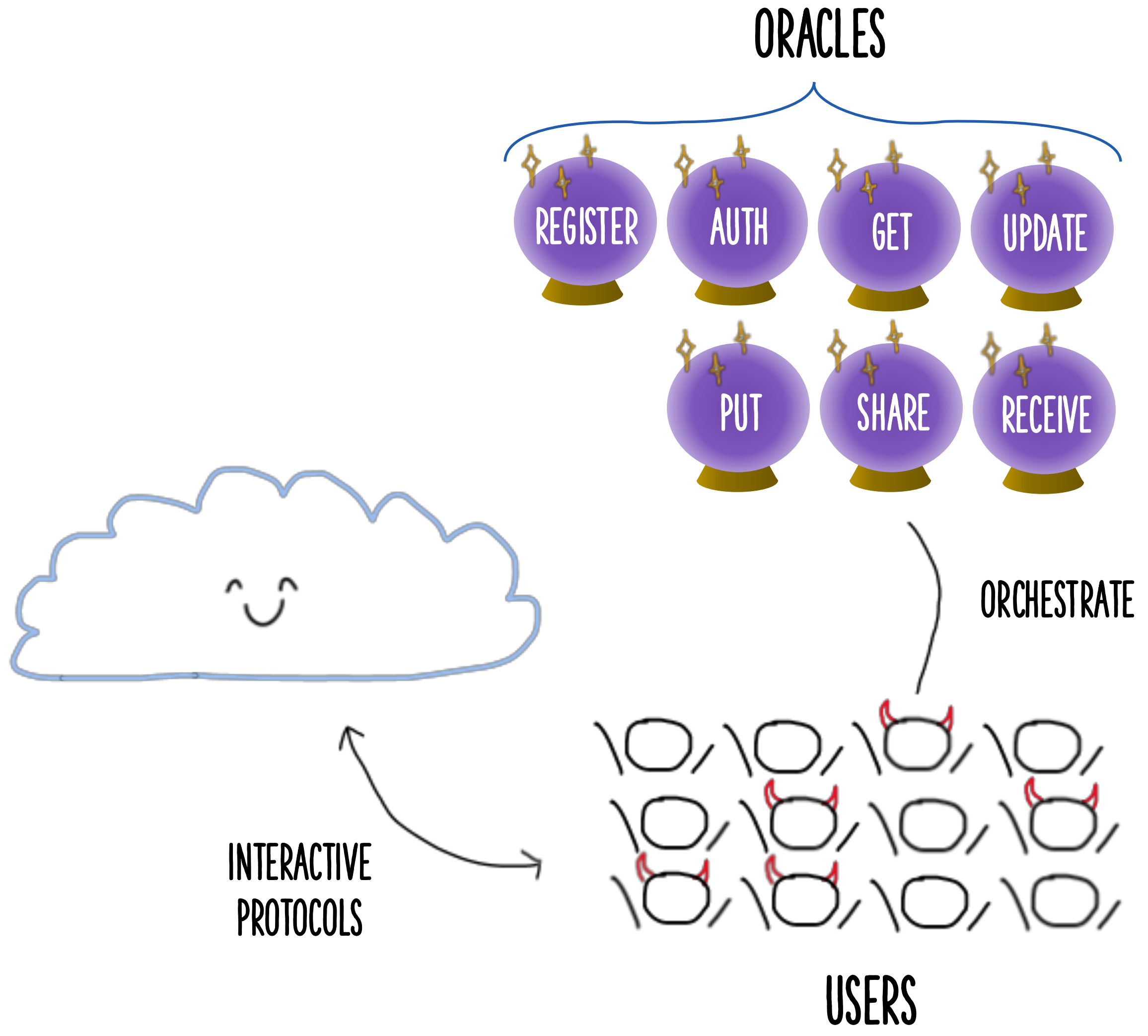
Security Notions

CLIENT-TO-SERVER (C2S): MAL. CLIENT [ONGOING WORK]

Threat model:
- Honest server
- Malicious clients
- Adversary controls honest user operations

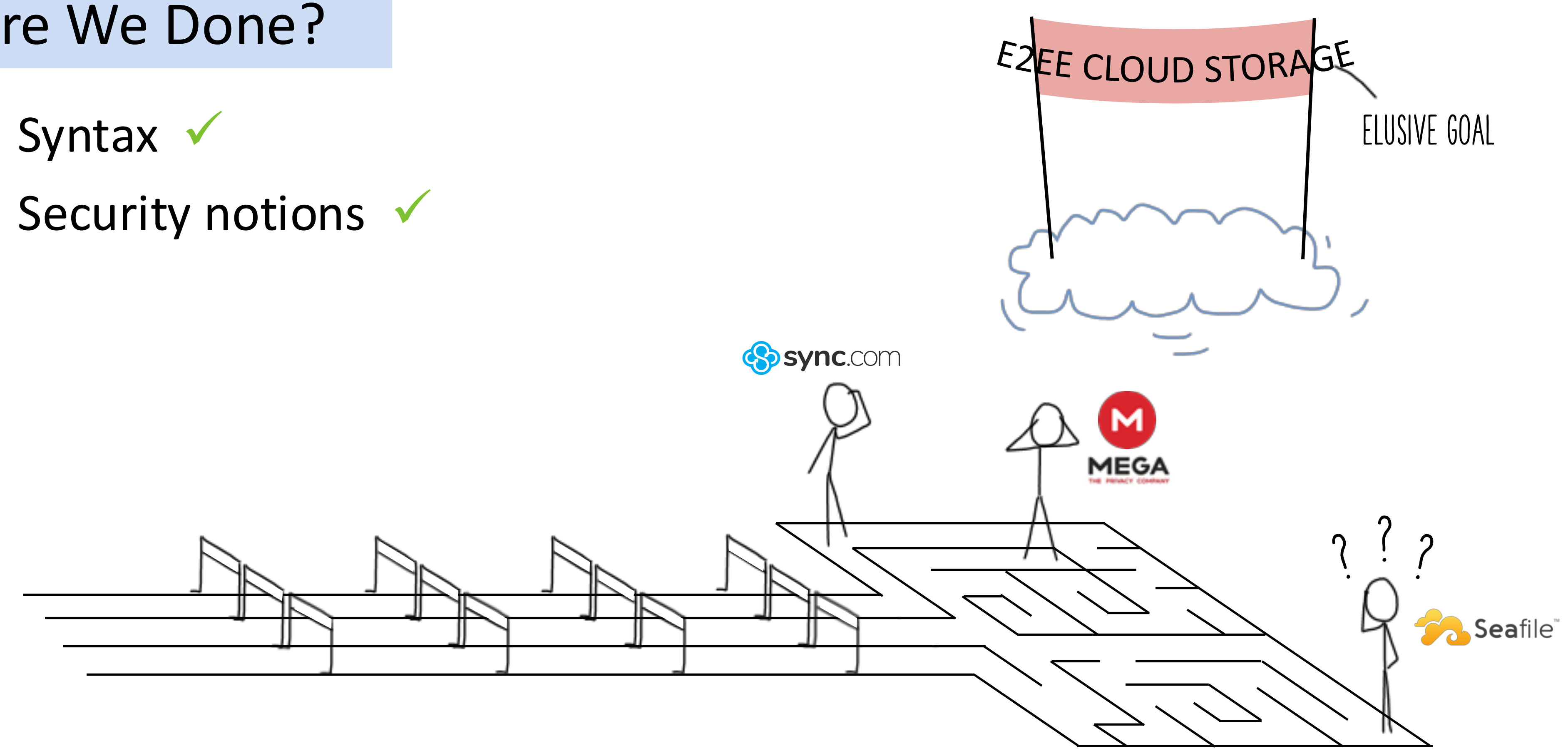
INFEASIBLE IN C2C!

Additional goals:
- Authentication & authorization
- No offline dictionary attacks on pw
- Availability for honest user files



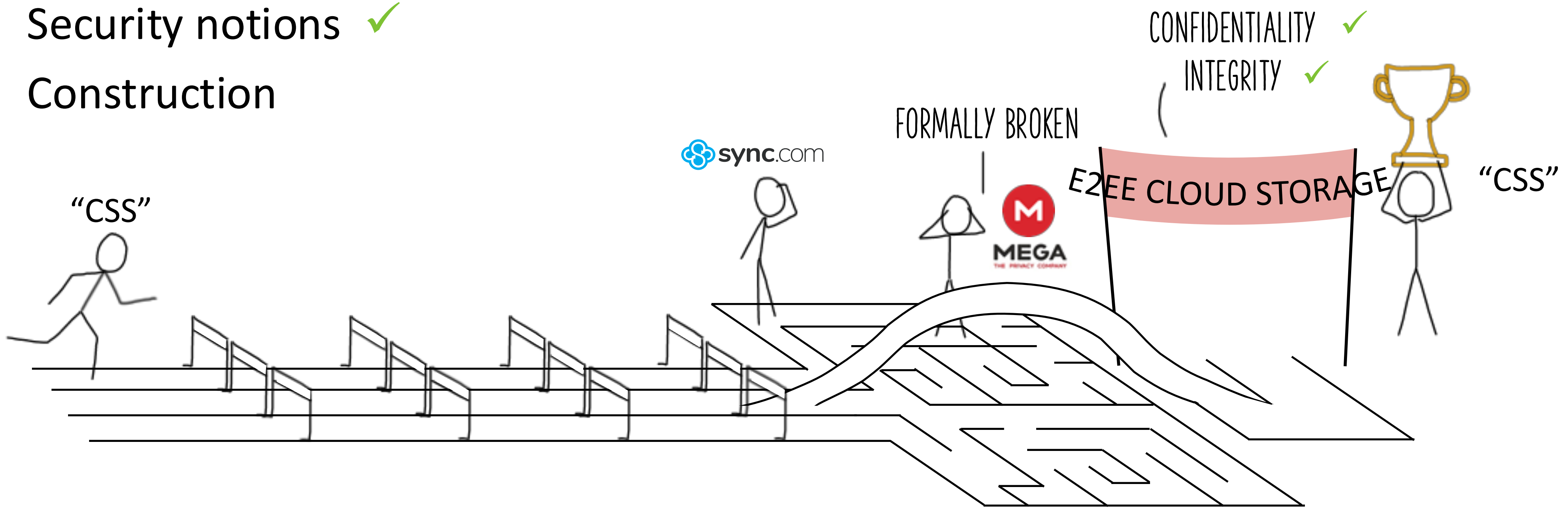
Are We Done?

- Syntax ✓
- Security notions ✓



Are We Done?

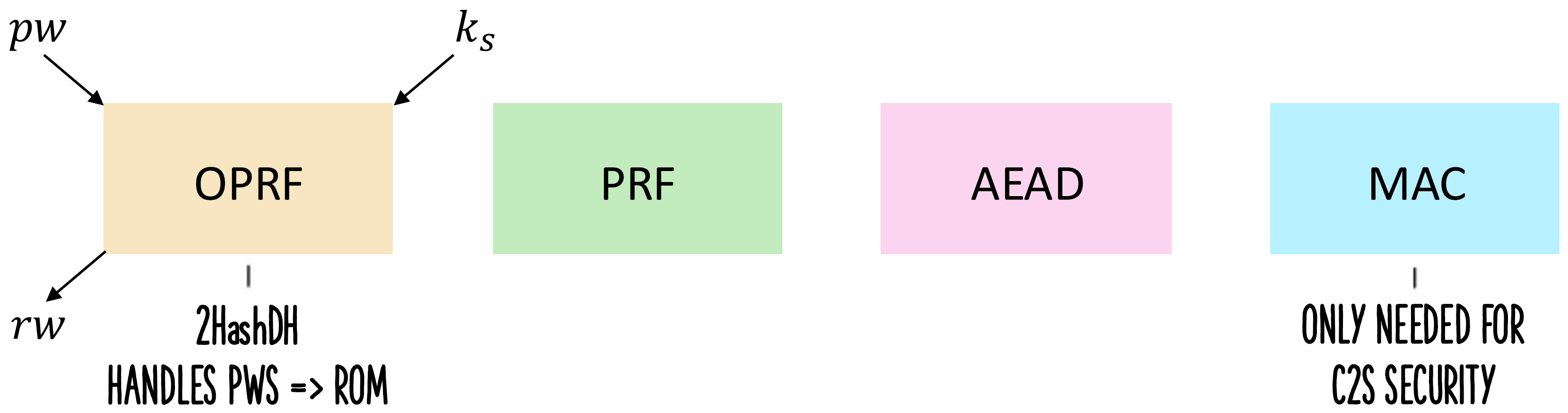
- Syntax ✓
- Security notions ✓
- Construction



2. Constructing E2EE Cloud Storage

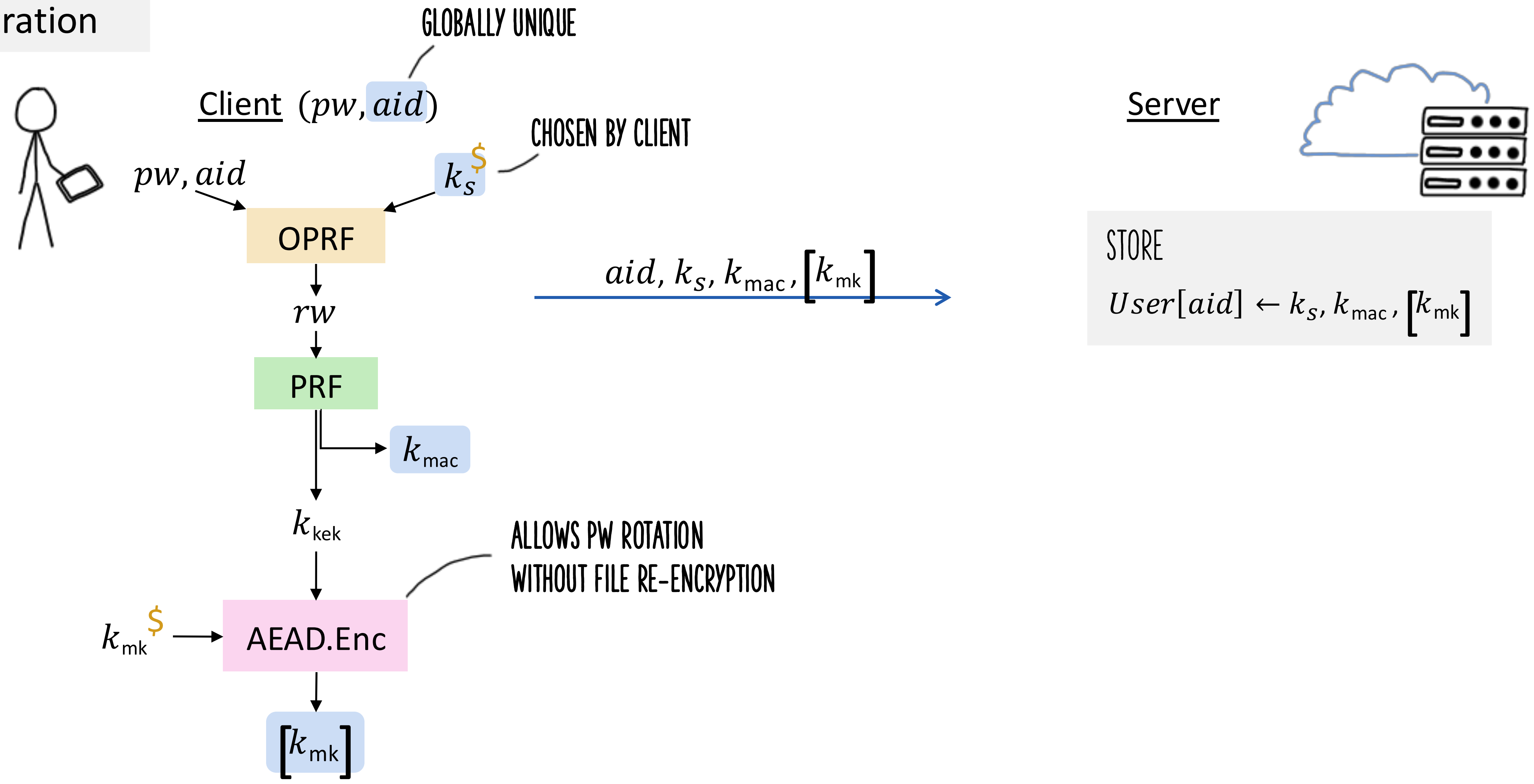
CSS (Cloud Storage Scheme)

Building Blocks



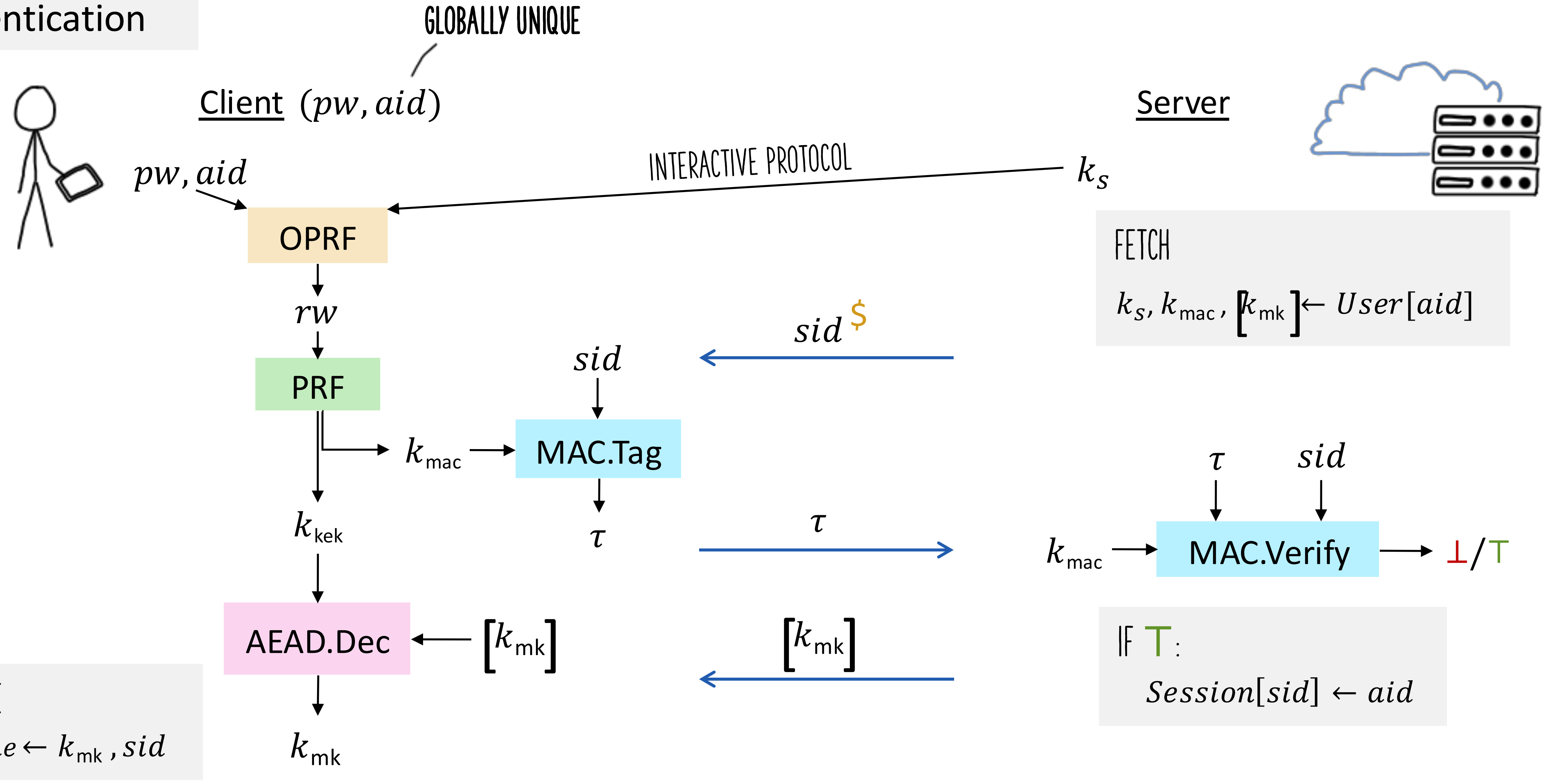
CSS (Cloud Storage Scheme)

Registration



CSS (Cloud Storage Scheme)

Authentication

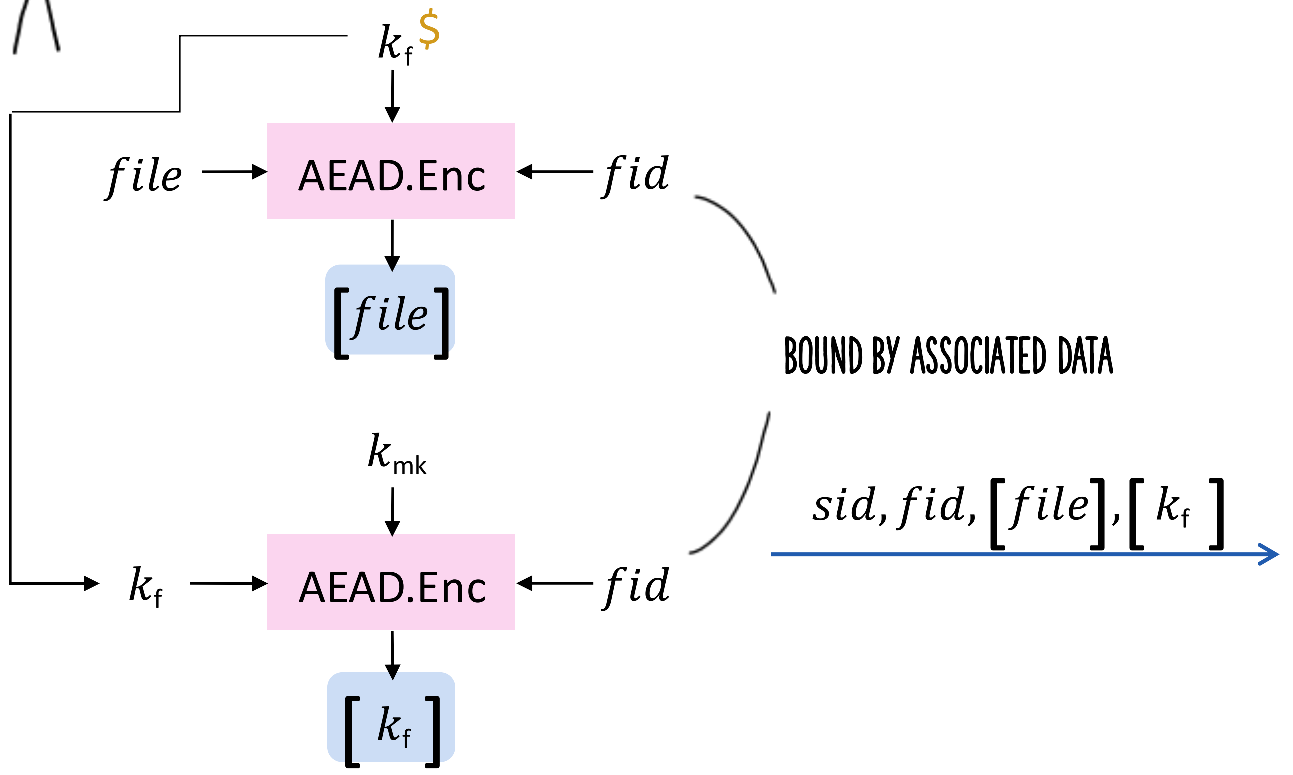


CSS (Cloud Storage Scheme)

Put

Client $(file, fid)$ GLOBALLY UNIQUE

$k_{mk}, sid \leftarrow Cache$



Server

FETCH
$aid \leftarrow Session[sid]$

STORE
$File[fid] \leftarrow [file]$ SHARED
$Key[aid, fid] \leftarrow [k_f]$ UNIQUE PER USER

CSS (Cloud Storage Scheme)

Share

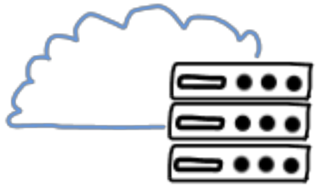
*SIMPLIFIED

RECIPIENT ACCOUNT ID

Client $(fid, raid)$

$k_{\mathrm{mk}}, sid \leftarrow Cache$

Server



$$sid, fid, raid \longrightarrow$$

FETCH

$aid \leftarrow Session[sid]$

$[k_{\mathrm{f}}] \leftarrow Key[aid, fid]$

$$\longleftarrow [k_{\mathrm{f}}]$$

$$\begin{tikzcd} & k_{\mathrm{mk}} \arrow[d] & \\ {[k_{\mathrm{f}}]} \arrow[r] & \boxed{\text{AEAD.Dec}} \arrow[d] & fid \arrow[l] \\ & k_{\mathrm{f}} \arrow[d] & \\ & \text{OOB} & \end{tikzcd}$$

SEND TO: $raid$

CSS (Cloud Storage Scheme)

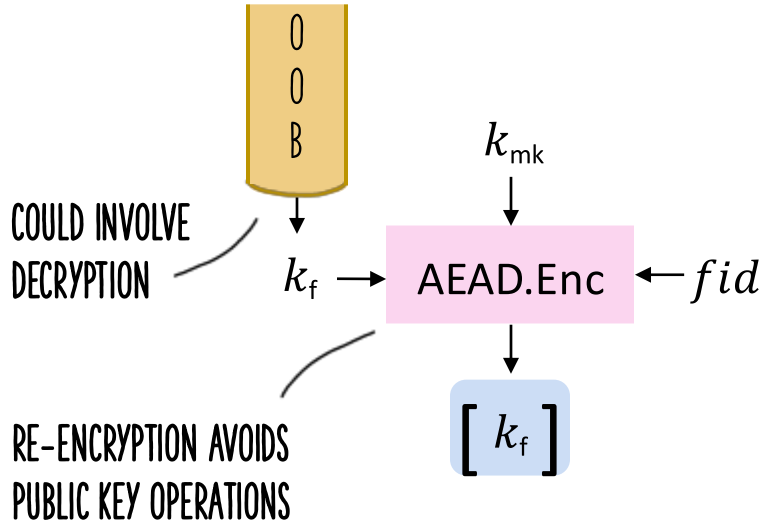
Accept

*SIMPLIFIED

Client (fid)

$k_{\text{mk}}, sid \leftarrow Cache$

Server



$$\xrightarrow{sid, fid, [k_{\text{f}}]}$$

FETCH

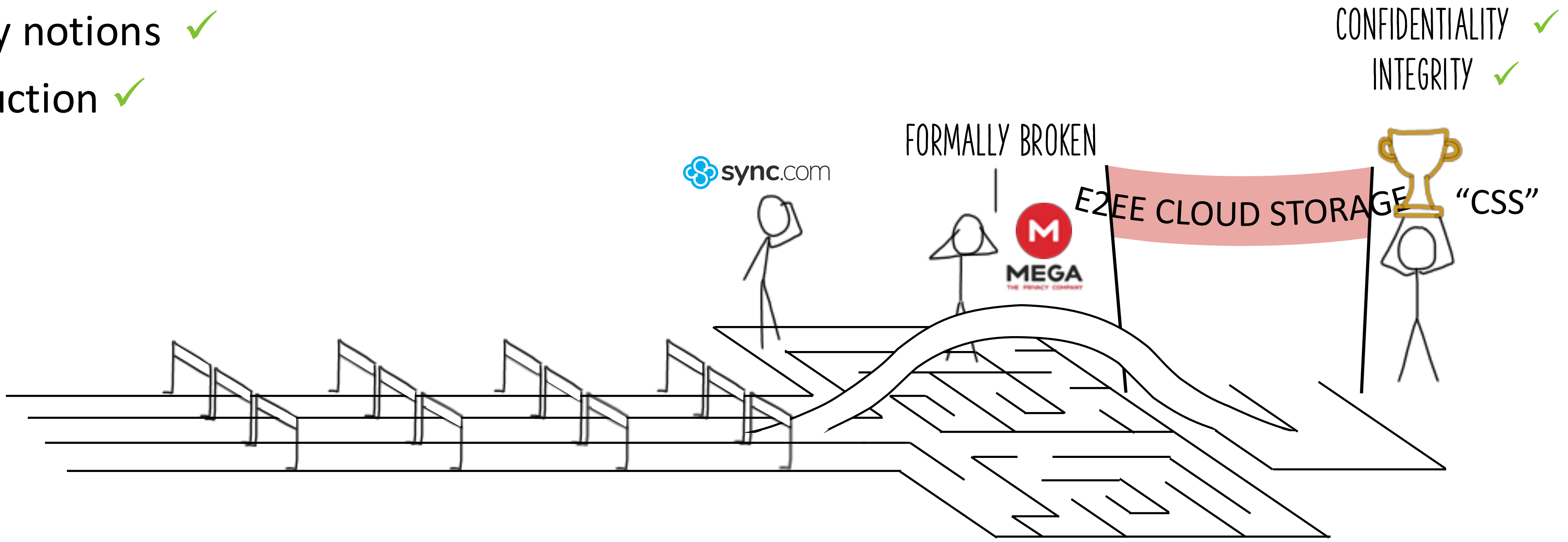
$aid \leftarrow Session[sid]$

STORE

$Key[aid, fid] \leftarrow [k_{\text{f}}]$

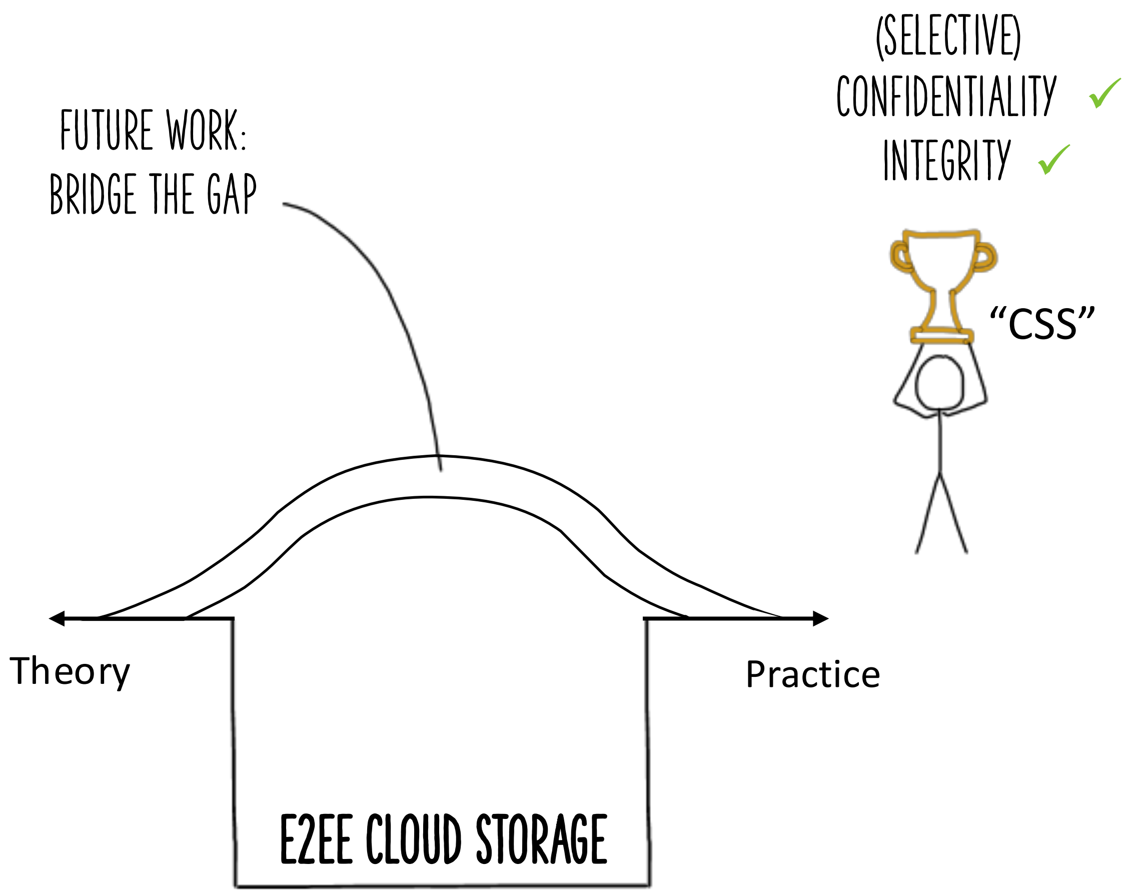
Are We Done?

- Syntax ✓
- Security notions ✓
- Construction ✓



Are We Done?

- Syntax ✓
- Security notions ✓
- Construction ✓

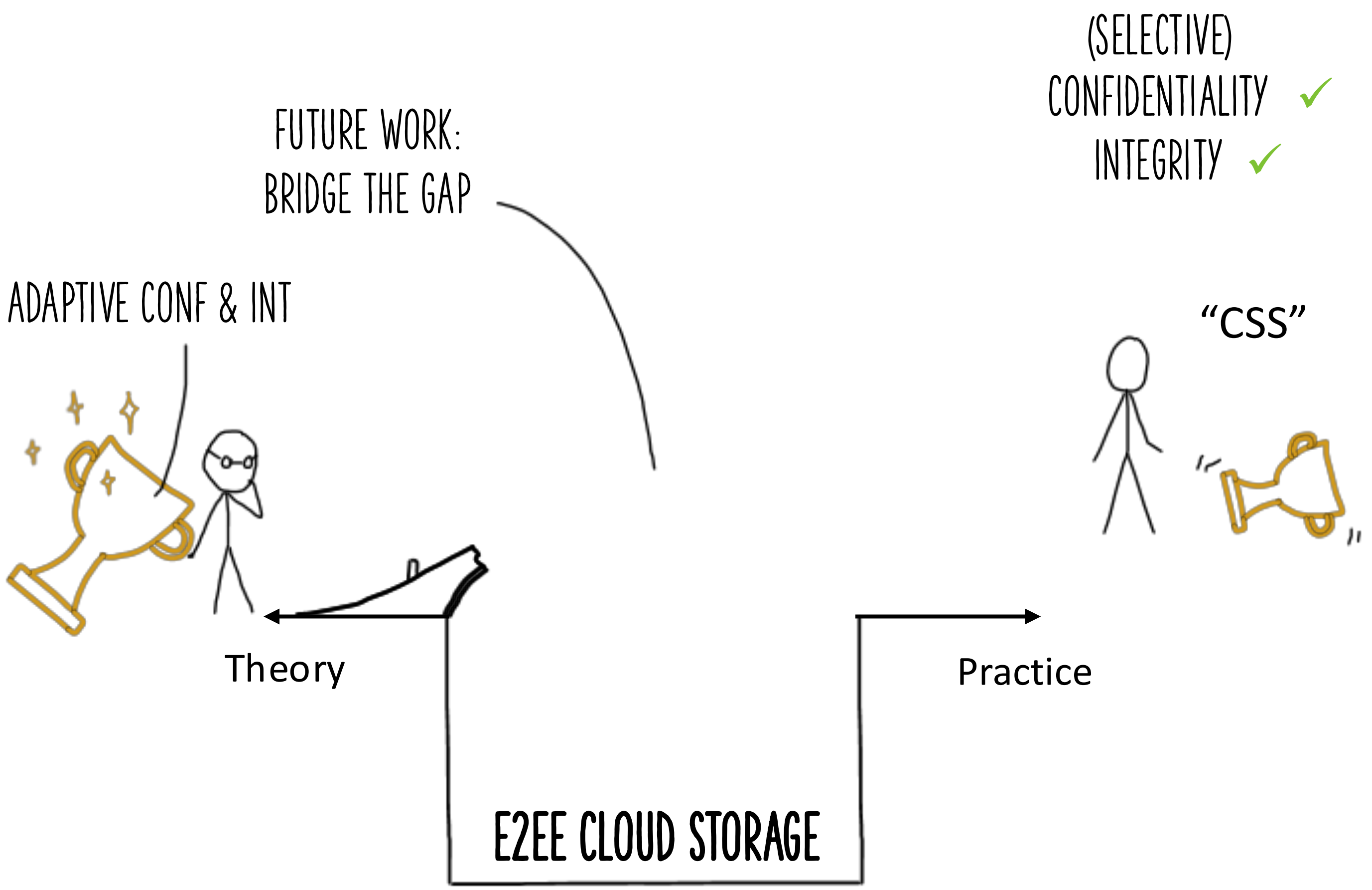


Are We Done?

- Syntax ✓
- Security notions ✓
- Construction ✓

Still missing:

- Adaptive security proof

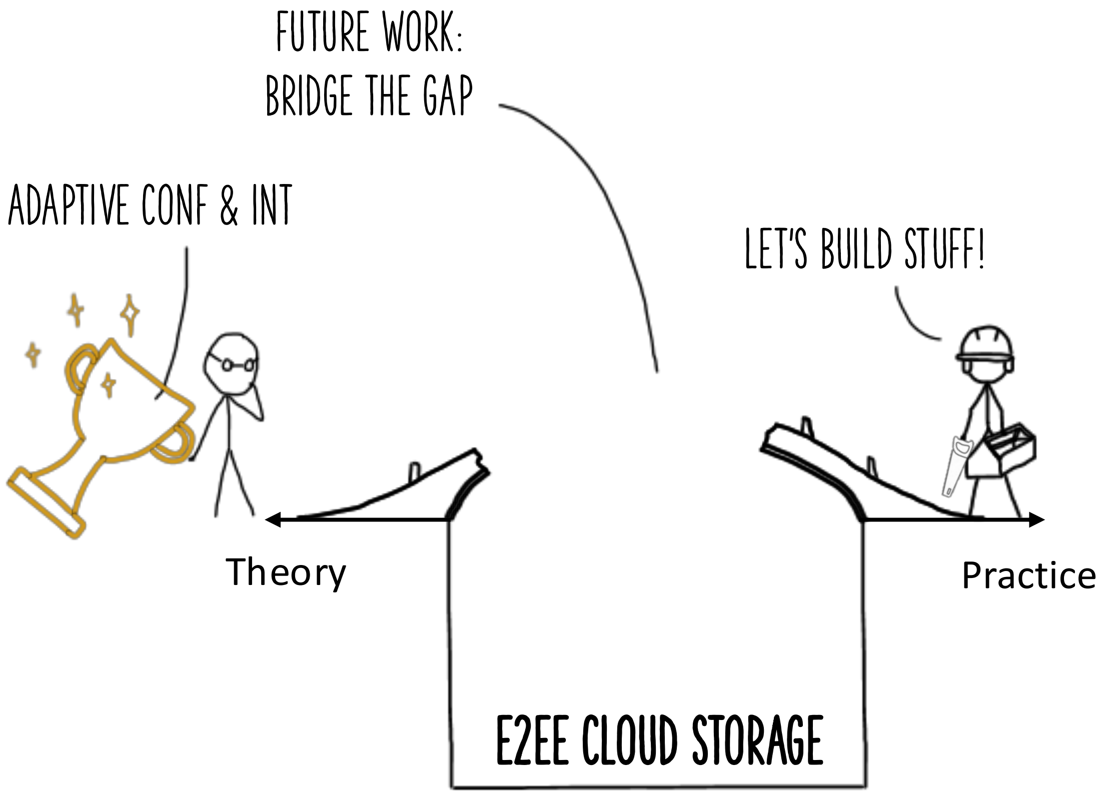


Are We Done?

- Syntax ✓
- Security notions ✓
- Construction ✓

Still missing:

- Adaptive security proof
- Implementation
- Feedback, model extensions, …



A Formal Treatment of End-to-End Encrypted Cloud Storage

Matilda Backendal, Hannah Davis, Felix Günther, Miro Haller, Kenny Paterson

mbackendal@inf.ethz.ch

mhaller@ucsd.edu

eprint.iacr.org/2024/989