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

University of Maryland, September 16, 2024

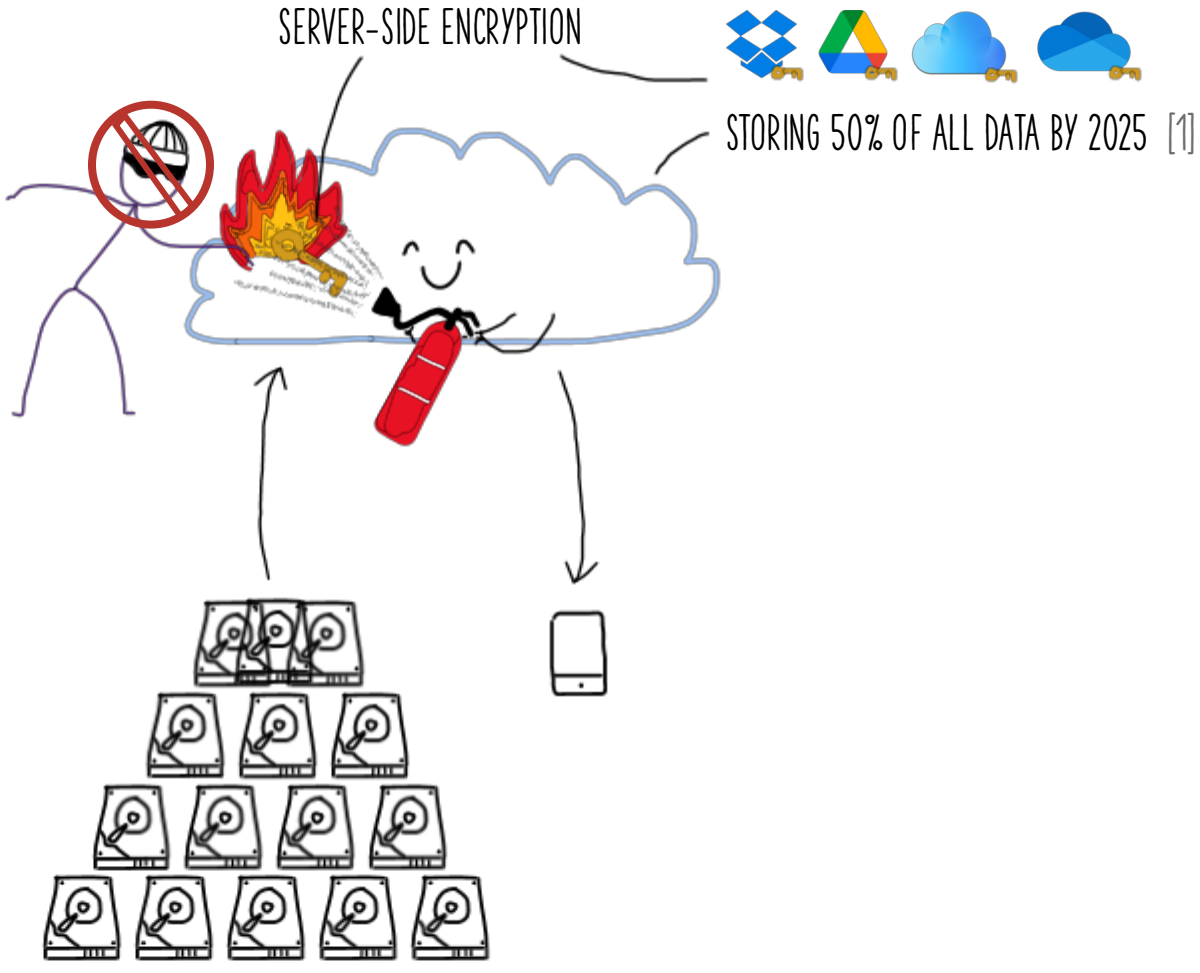
Cloud Storage

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)

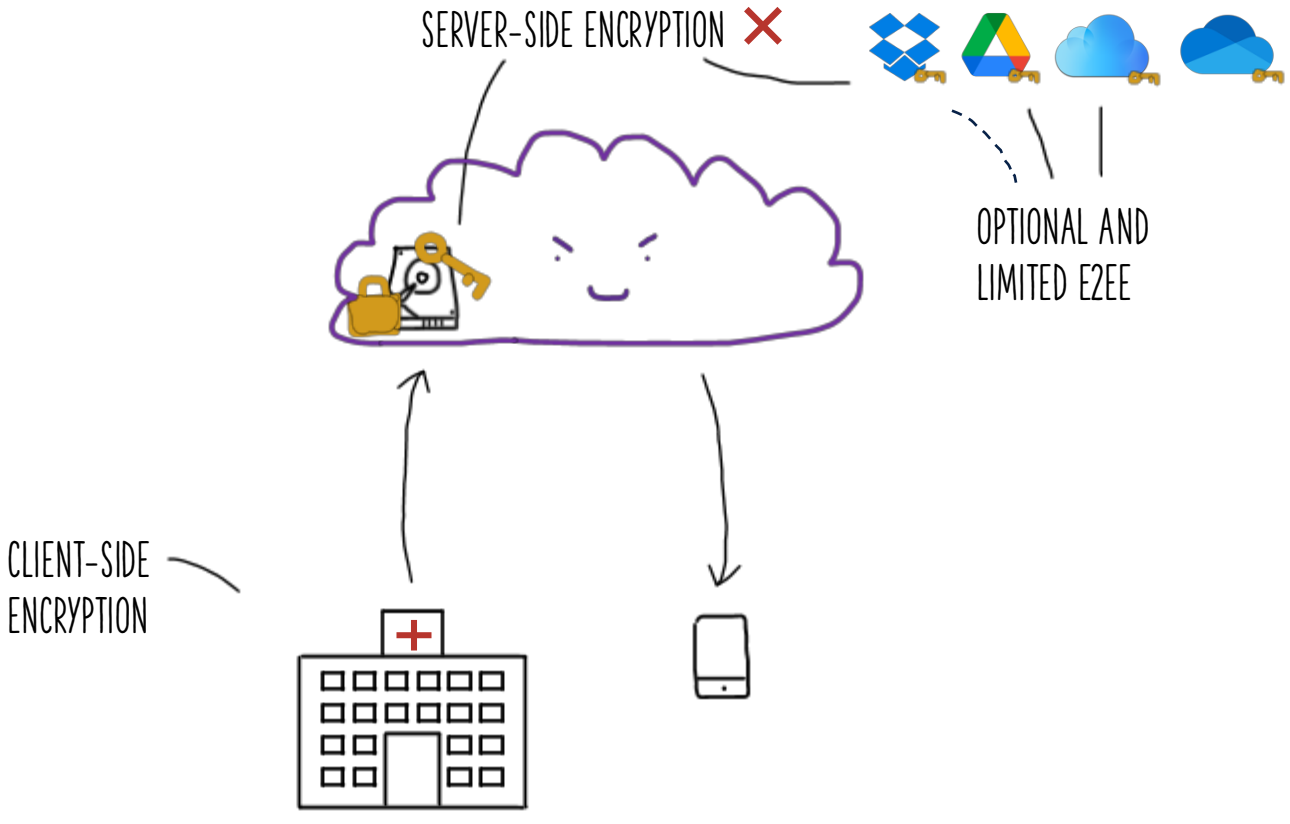
Cloud Storage

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

INSECURE!

[SP:BHP23]
[EC:AHMP23]

AMNESTY INTERNATIONAL, THE GERMAN FEDERAL GOVERNMENT & ETH — "ULTIMATE SECURITY"

INSECURE!

[EuroSP:ABCP23]

"EXCEPTIONALLY PRIVATE CLOUD"

"EUROPE'S MOST SECURE CLOUD STORAGE"

"THE STRONGEST ENCRYPTED CLOUD STORAGE IN THE WORLD"

INSECURE!

[CCS:TH24]

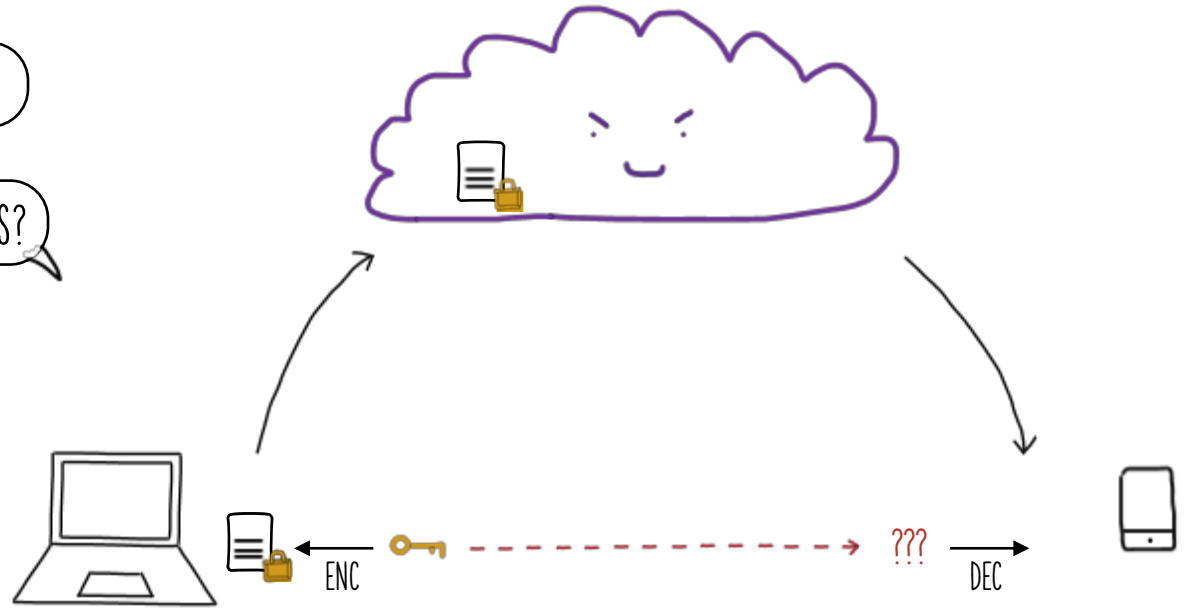
"SUPPORTS CLIENT-SIDE END-TO-END ENCRYPTION"

Why Is It Hard?

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

HOW DO YOU TRANSFER KEYS BETWEEN DEVICES?

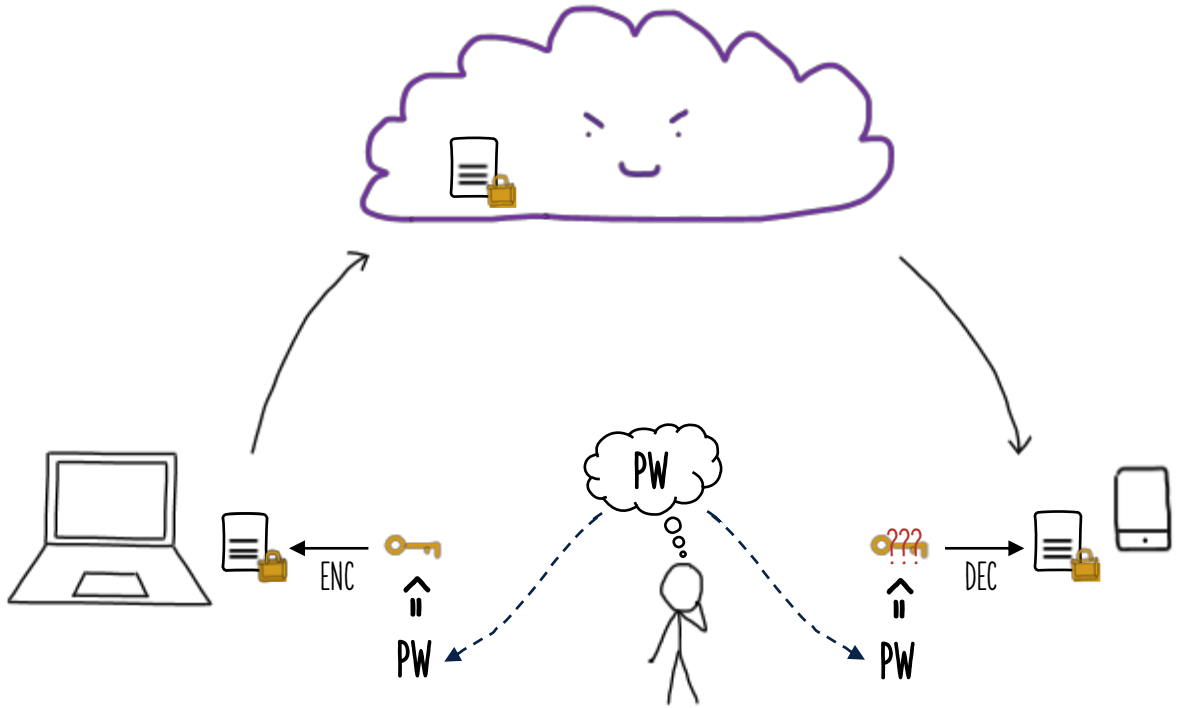
1 key distribution



Why Is It Hard?

DERIVE KEYS FROM THE PASSWORD!

1	key distribution
2	password-based security



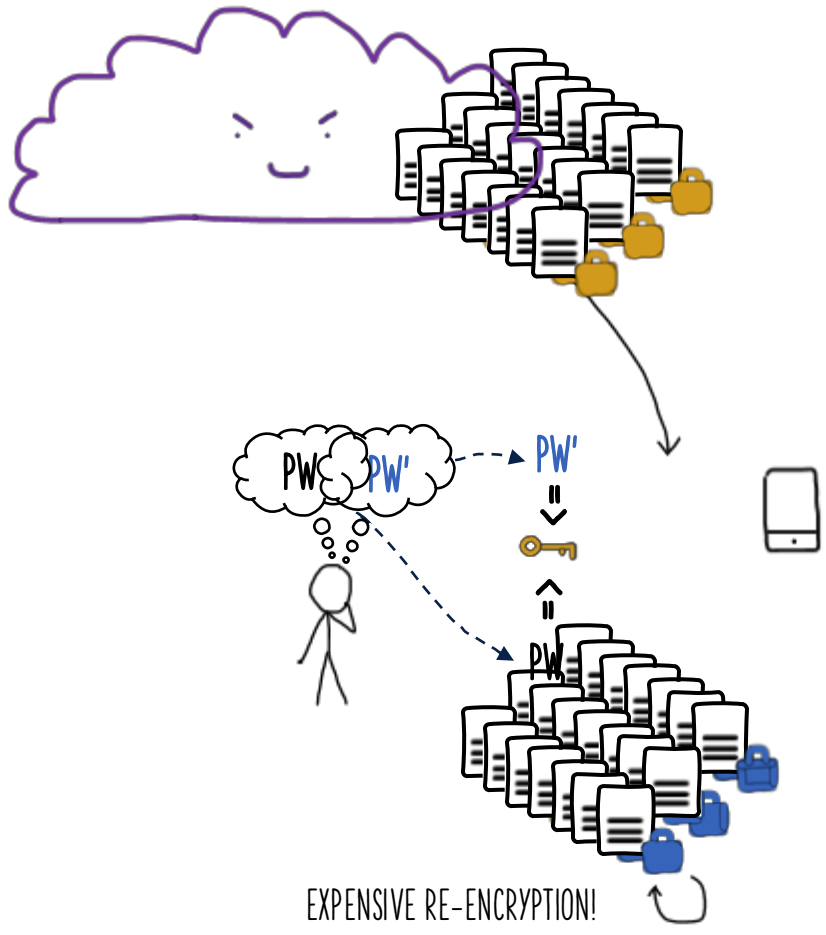
Why Is It Hard?

DERIVE KEYS FROM THE PASSWORD!

WHAT IF THE PASSWORD CHANGES?

1	key distribution
2	password-based security

PROBLEM 1: PW CHANGE

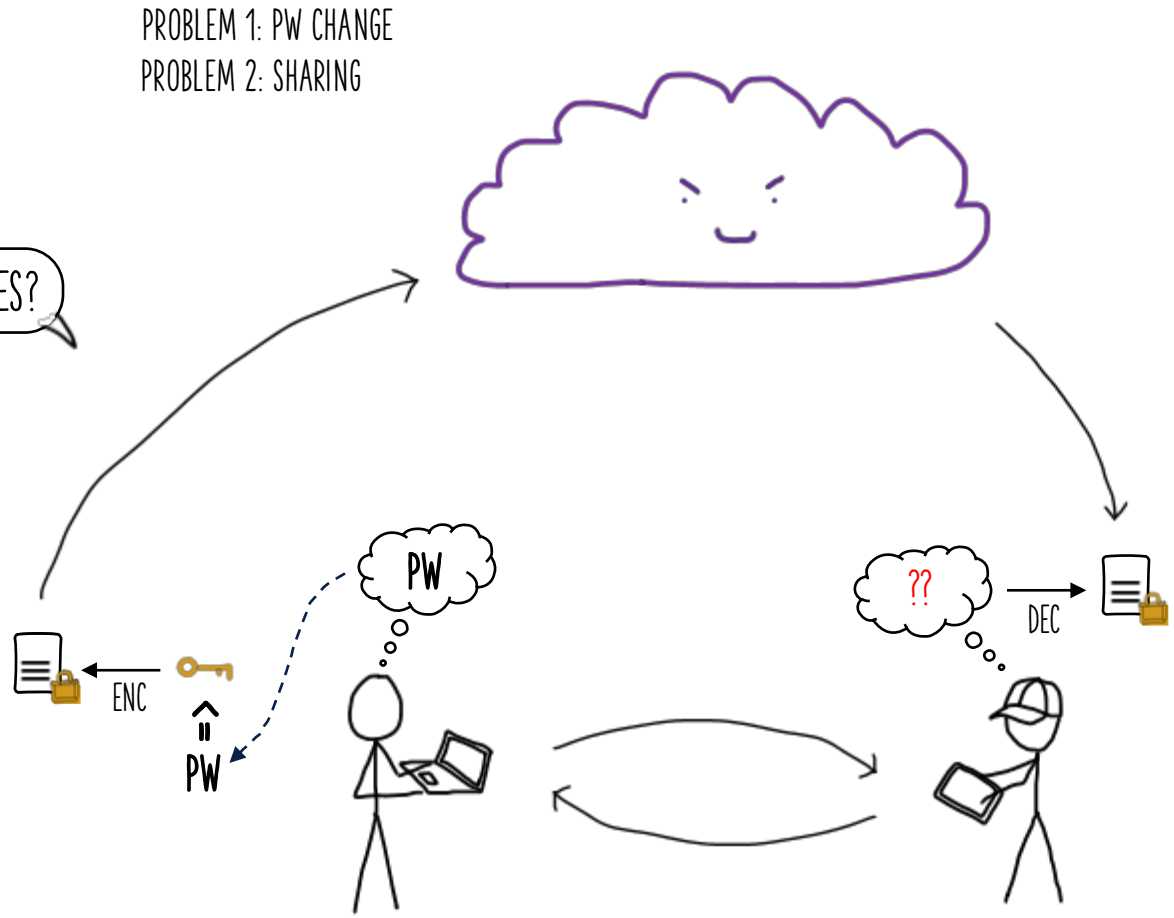


Why Is It Hard?

DERIVE KEYS FROM THE PASSWORD!

HOW DO YOU SHARE FILES?

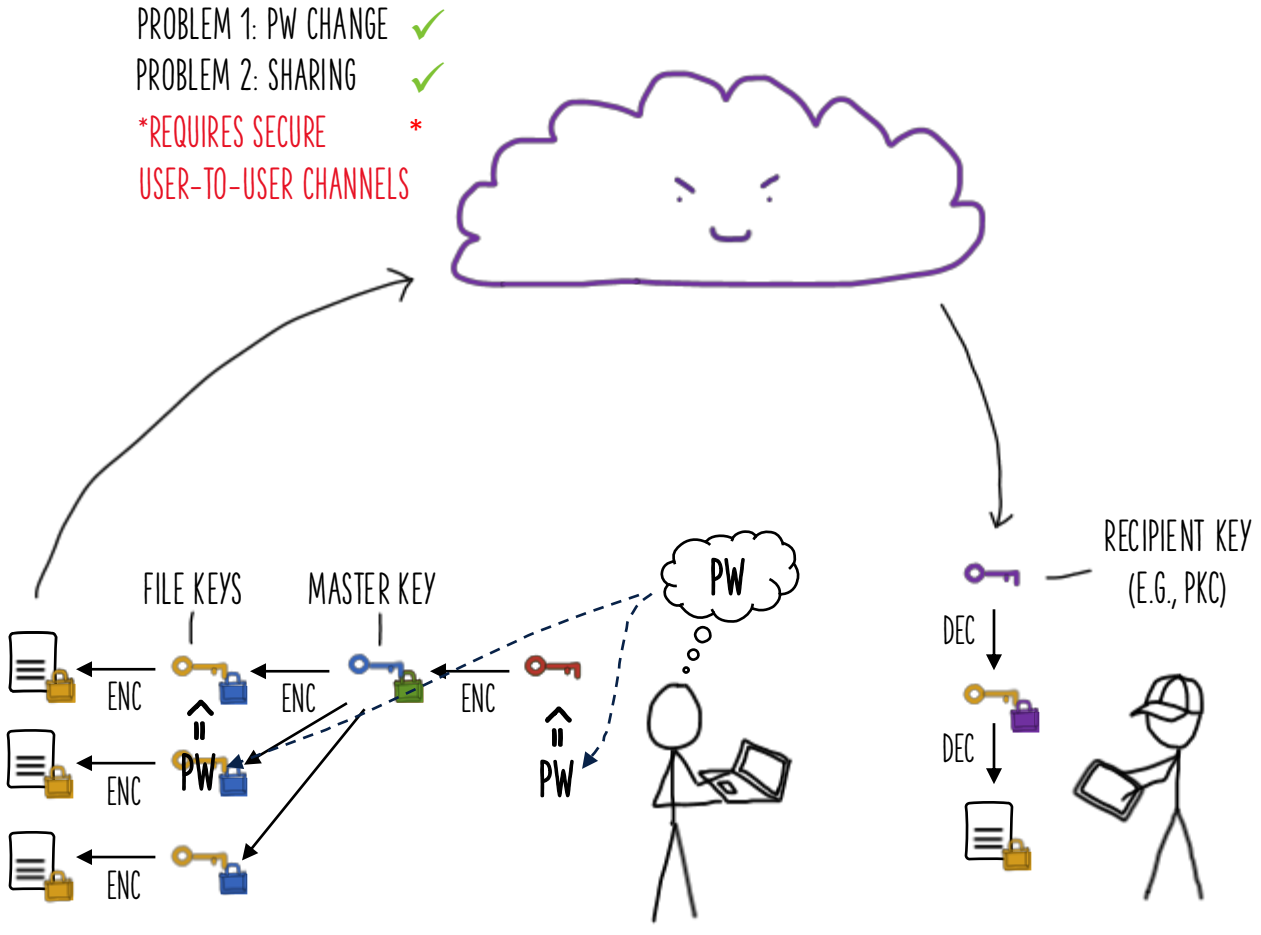
1	key distribution
2	password-based security
3	file sharing



Why Is It Hard?

BUILD A KEY HIERARCHY!

1	key distribution
2	password-based security
3	file sharing



Why Is It Hard?

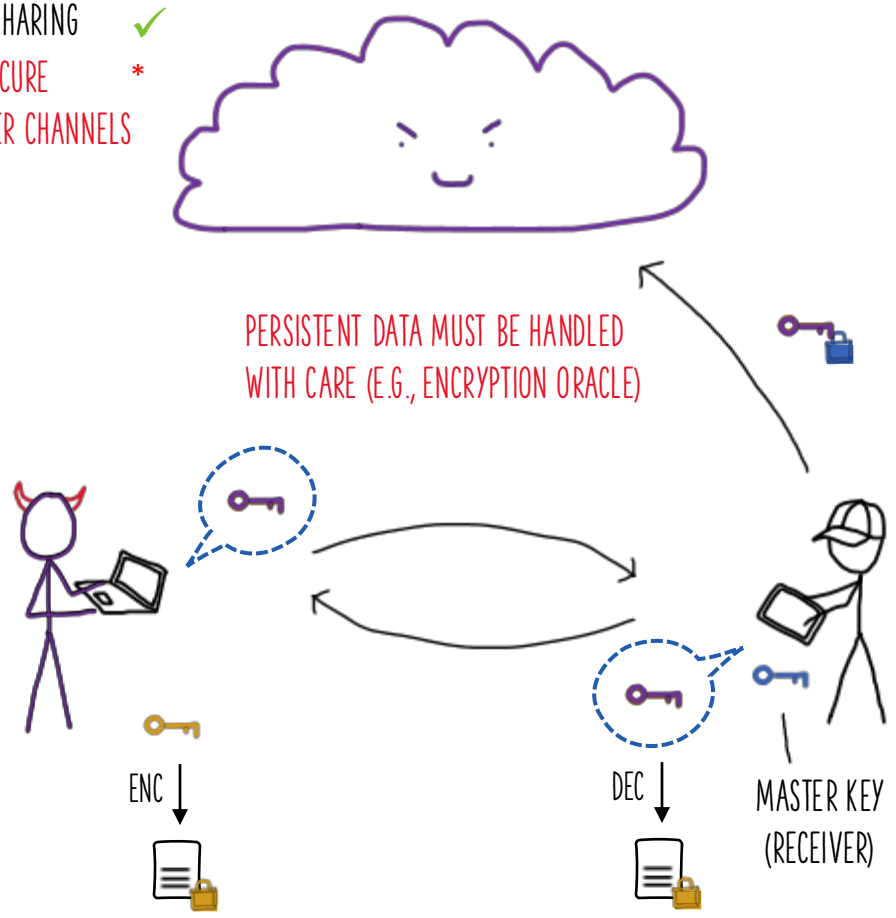
USE SECURE MESSAGING TECHNIQUES!

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



Interlude: MEGA

A CASE STUDY OF CRYPTOGRAPHY IN THE WILD

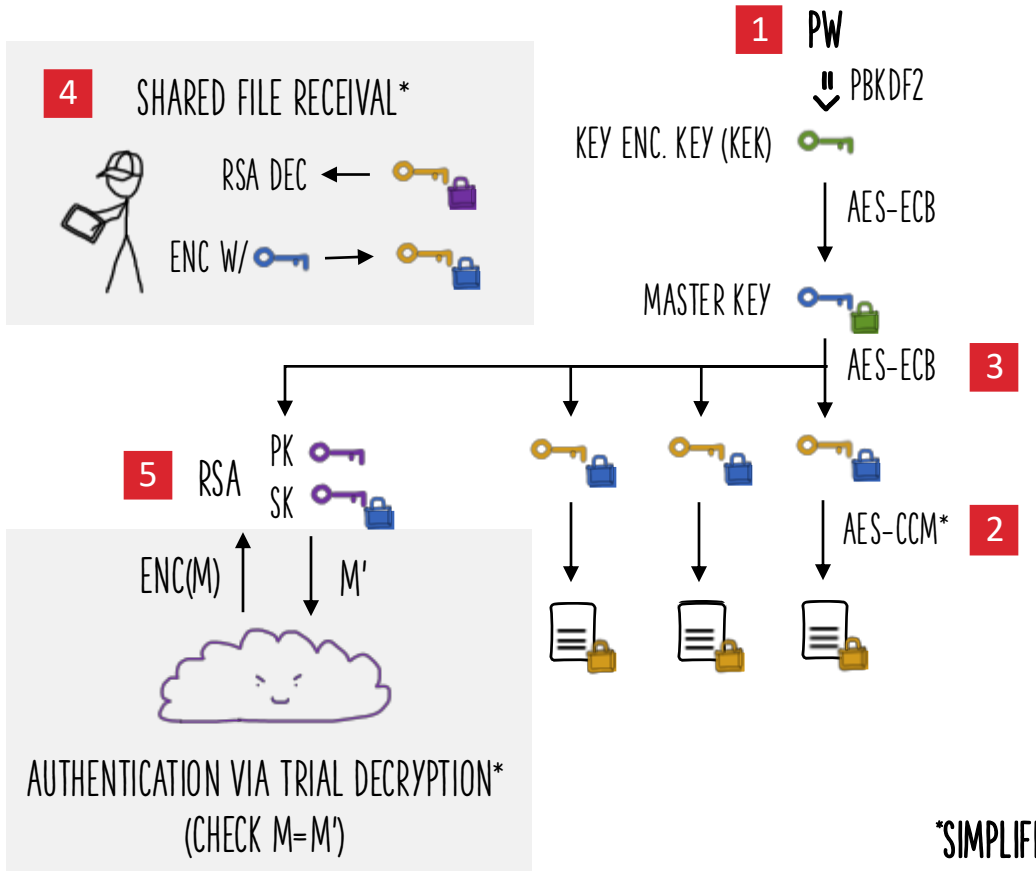
MEGA's challenges

1	Multi-device access	→ USERS ONLY NEED TO REMEMBER PW
2	File re-encryption	→ REPLACING AES-CCM > 180 DAYS
3	Ciphertext integrity	→ ENABLES ATTACKS IN [1, 2]
4	File sharing	→ RSA SECRET KEY DECRYPTION [2]
5	Key reuse	→ FILE KEY DECRYPTION [1]

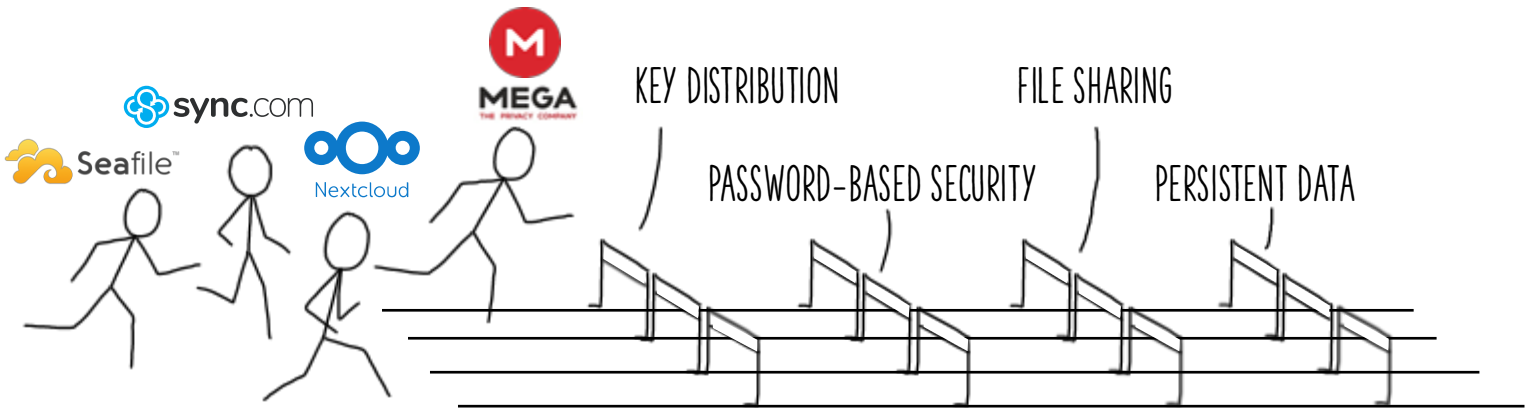
[1] Matilda Backendal, Miro Haller* and Kenneth G. Paterson. (2023). "MEGA: Malleable Encryption Goes Awry" IEEE S&P 2023.

[2] Martin R. Albrecht, Miro Haller, Lenka Mareková*, Kenneth G. Paterson. (2023). "Caveat Implementor! Key Recovery Attacks on MEGA" Eurocrypt 2023.

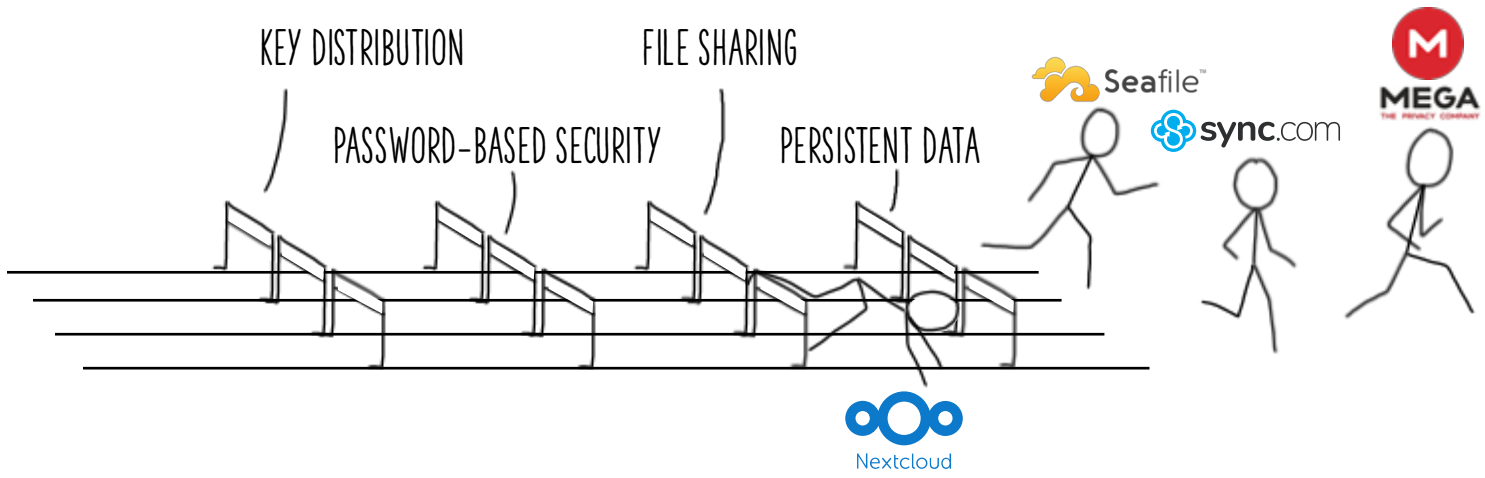
MEGA's key hierarchy*



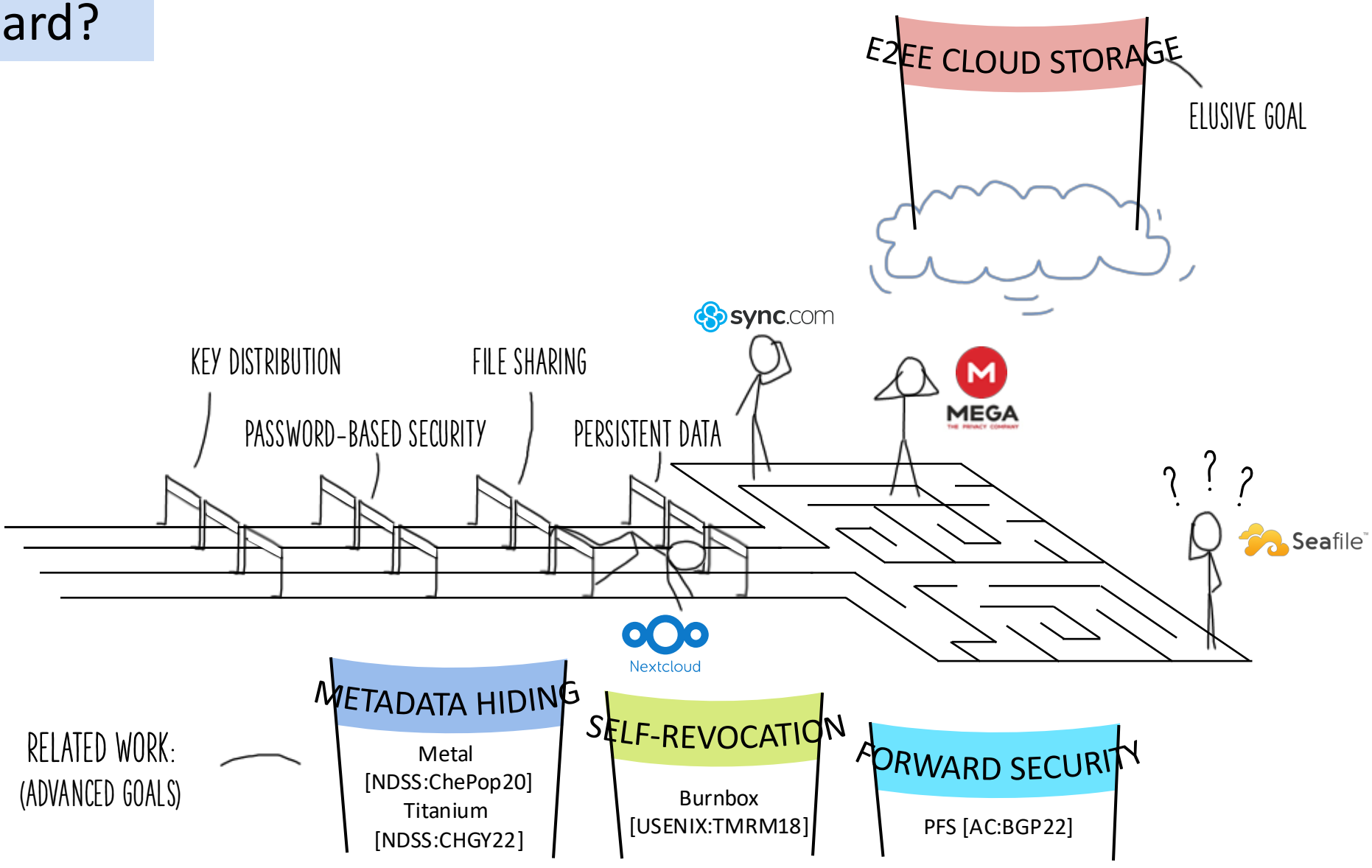
Why Is It Hard?



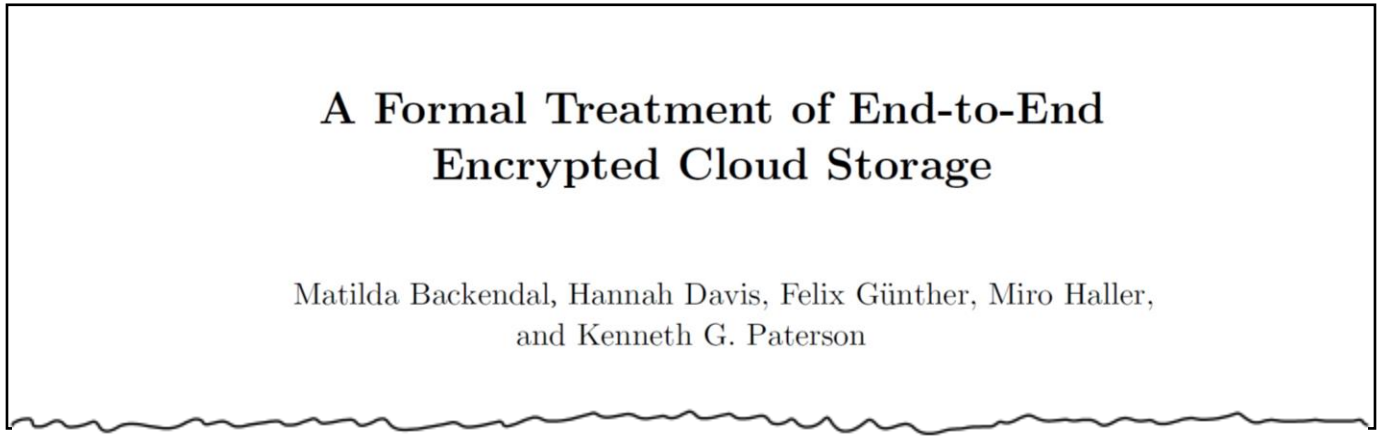
Why Is It Hard?



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

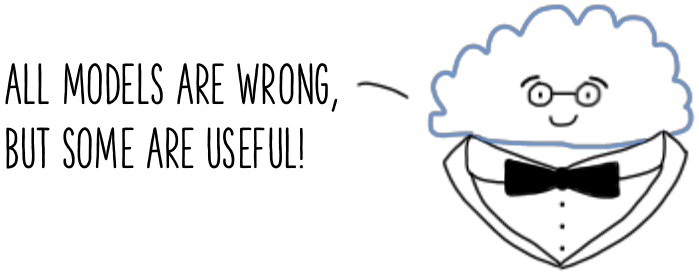
2 Construction

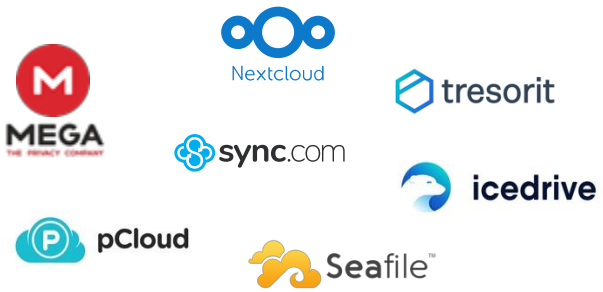
- CSS (Cloud Storage Scheme)
- Security proofs

1. Formalizing E2EE Cloud Storage

Formalizing E2EE Cloud Storage

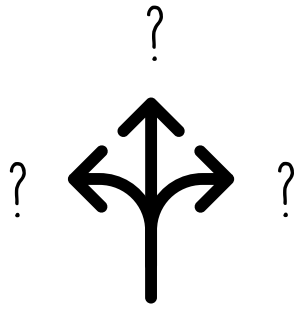
Model Goals





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?

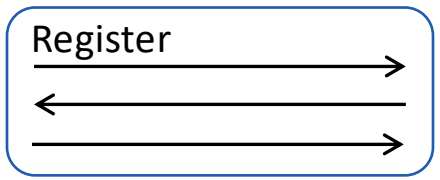
Core Functionality

1 EXPRESSIVE

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

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

INTERACTIVE PROTOCOLS



Syntax

HOW DO WE MAKE THE MODEL USEFUL?

Core Functionality

1 EXPRESSIVE

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

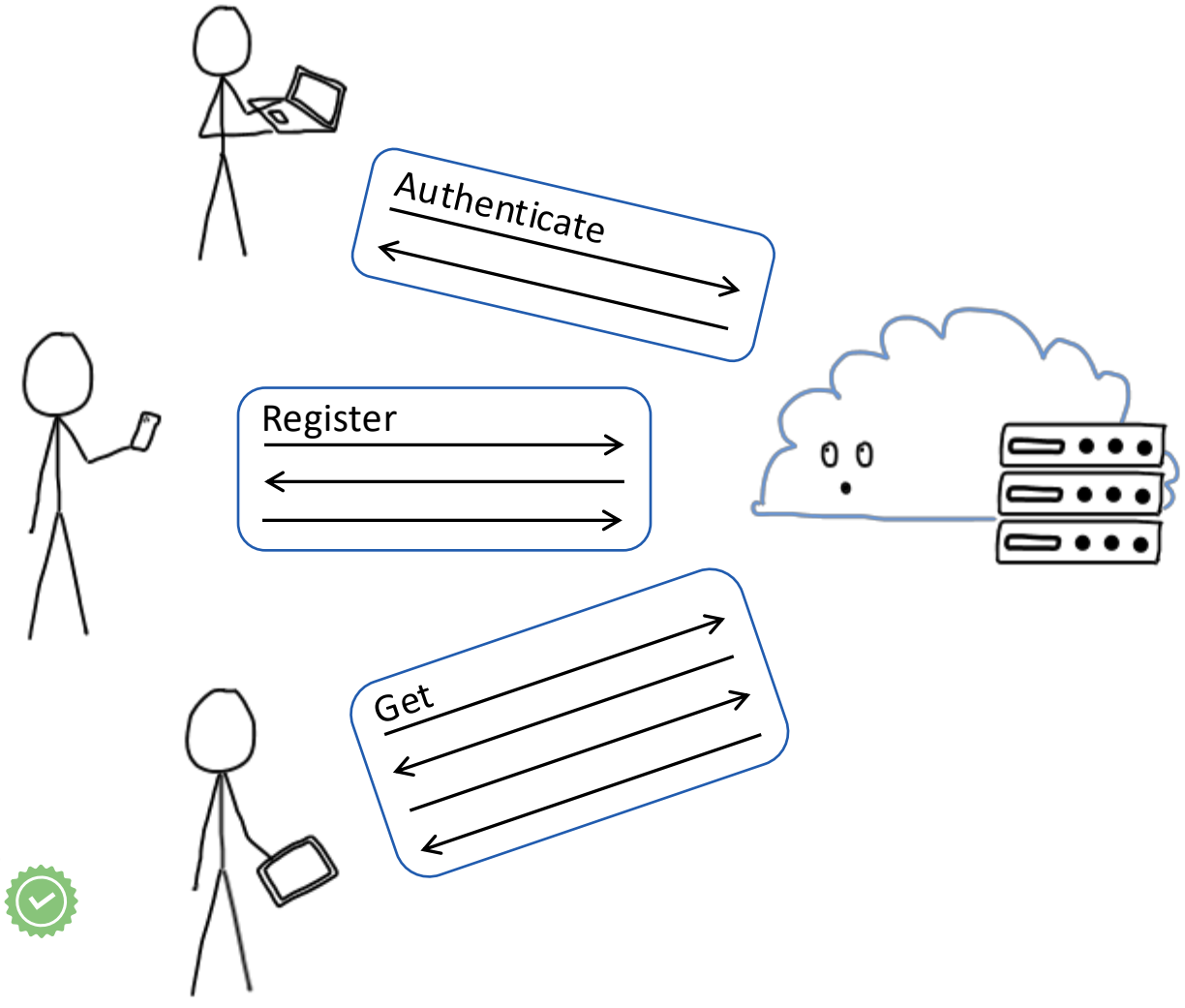
INTERACTIVE PROTOCOLS

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

Model Choices

- Arbitrary interleaving

2 FAITHFUL



Syntax

HOW DO WE MAKE THE MODEL USEFUL?

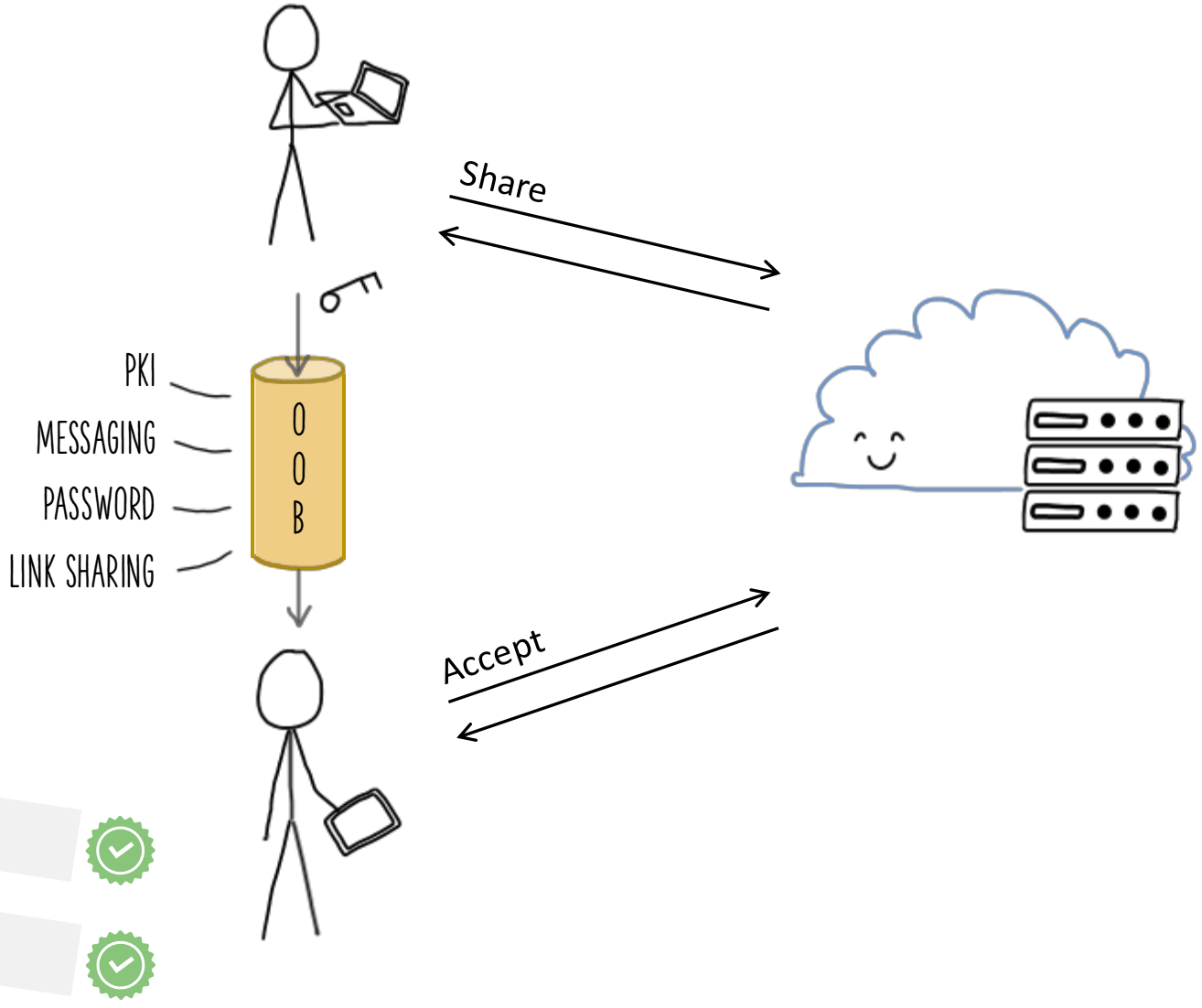
Core Functionality

1 EXPRESSIVE

- 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

- Arbitrary interleaving
- Abstract OOB channel for sharing

2 FAITHFUL

3 GENERIC

Security Notions

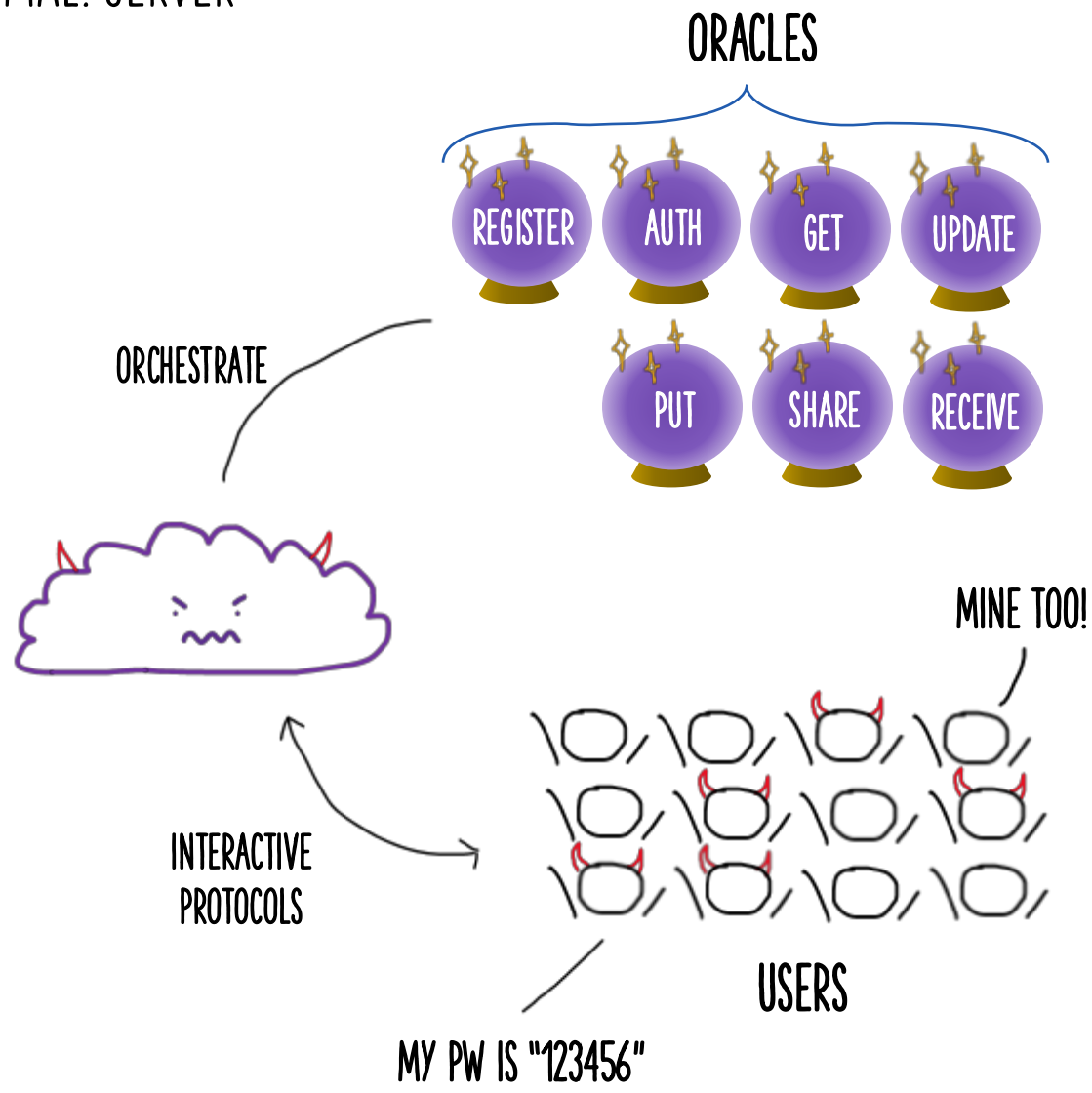
CLIENT-TO-CLIENT (C2C): MAL. SERVER

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)

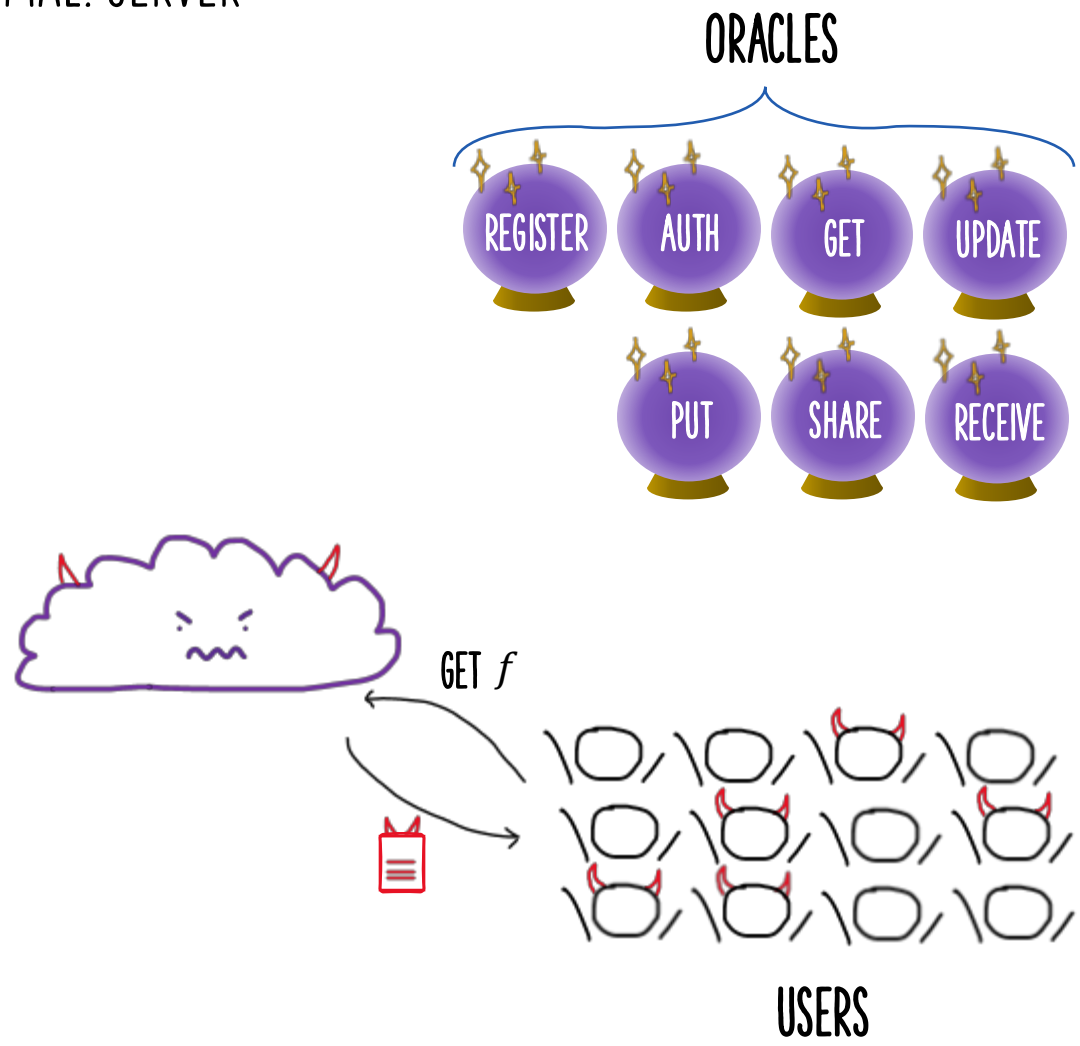


Security Notions

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

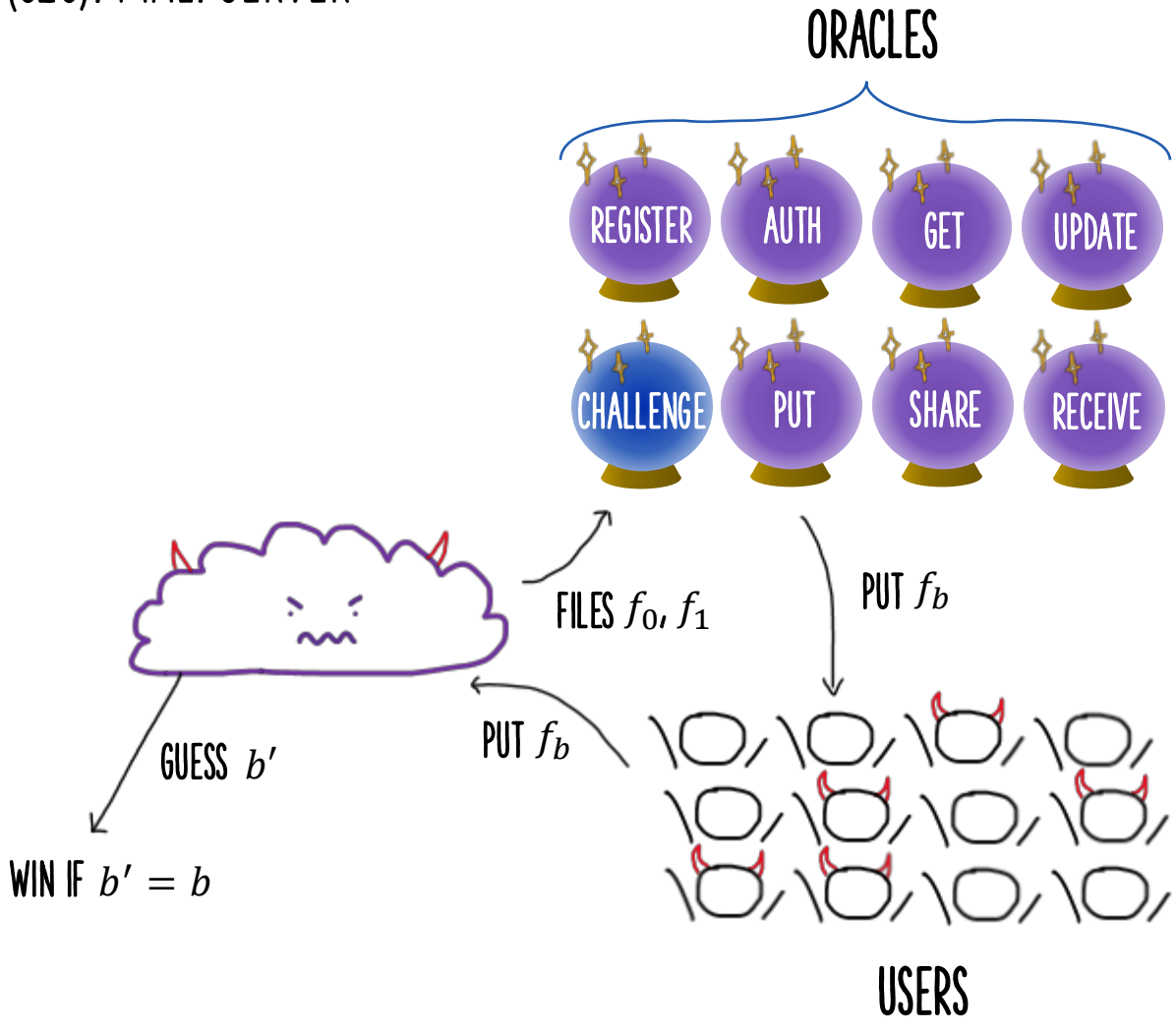
CLIENT-TO-CLIENT (C2C): MAL. SERVER

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: Considerations

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-CCA

1 No generic ciphertexts

↳ ALLOWS GENERIC SYNTAX

2 Adaptive & selective compromises

↳ AVOIDS COMMITMENT ISSUES

3 UC vs. game-based notions

↳ UC SECURE CHANNEL TECHNIQUES DO NOT APPLY

Security Notions

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

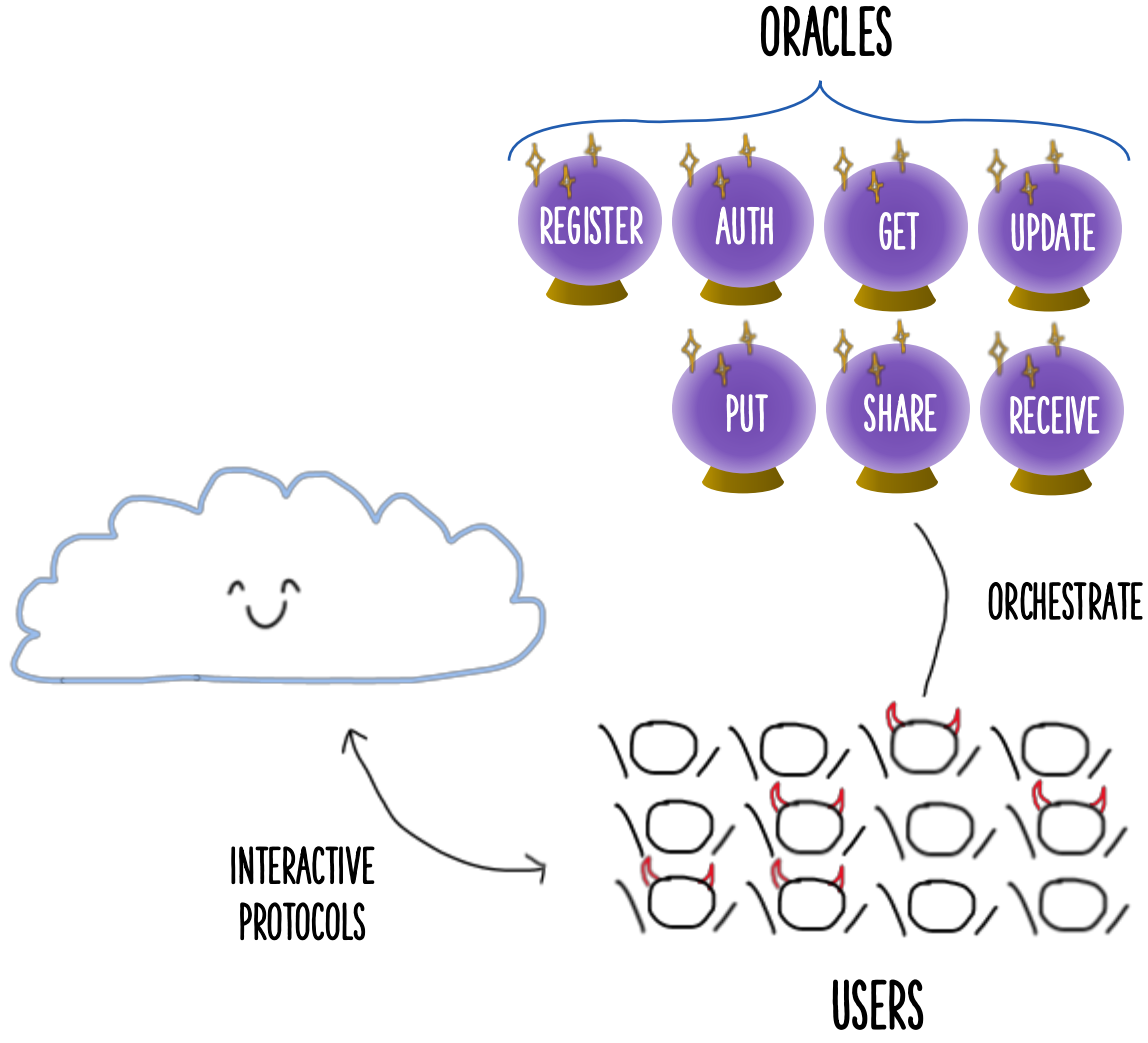
Threat model:

- Honest server
- Malicious clients
- Adversary controls honest user operations

Additional goals:

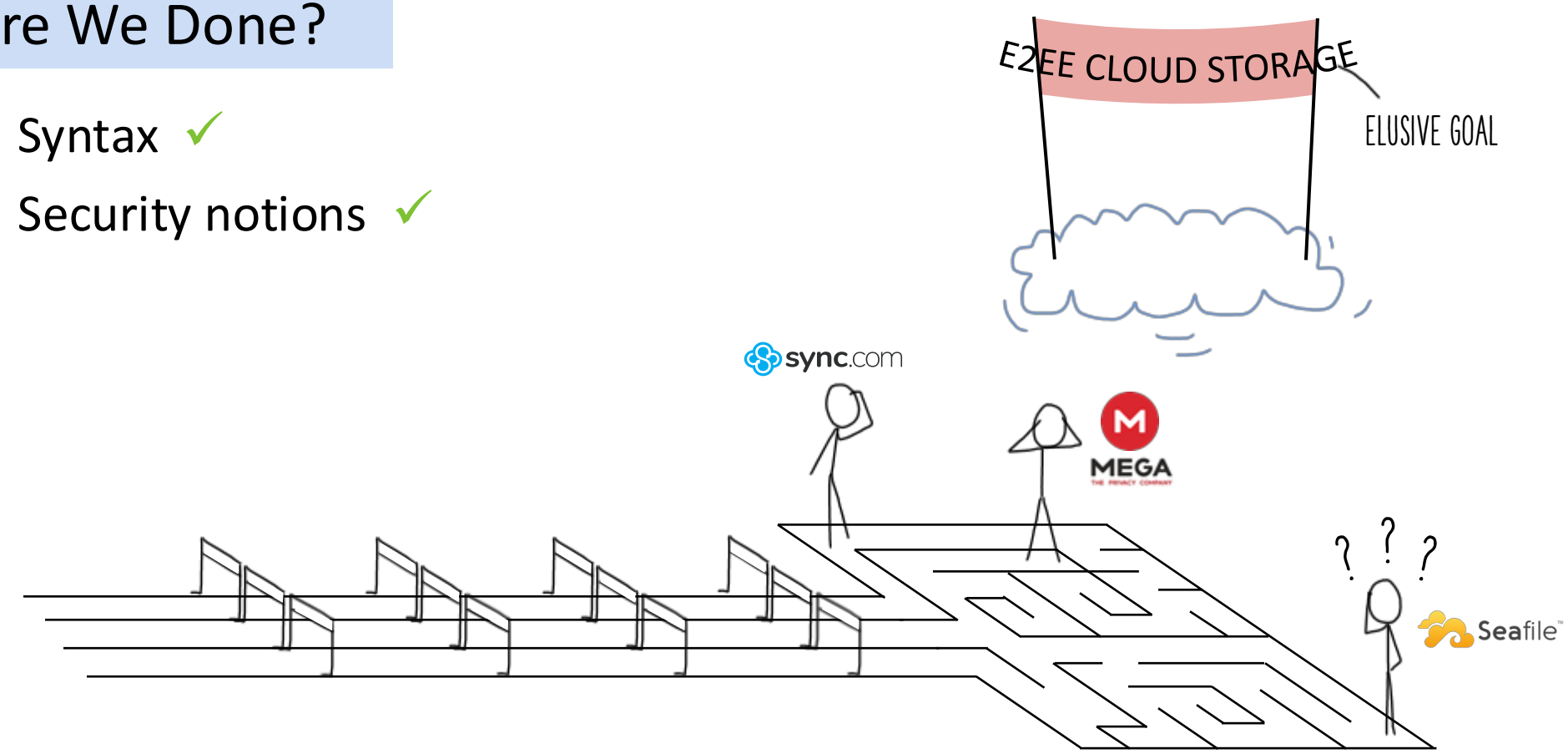
- Authentication & authorization
- No offline dictionary attacks on pw
- Availability for honest user files

INFEASIBLE IN C2C!



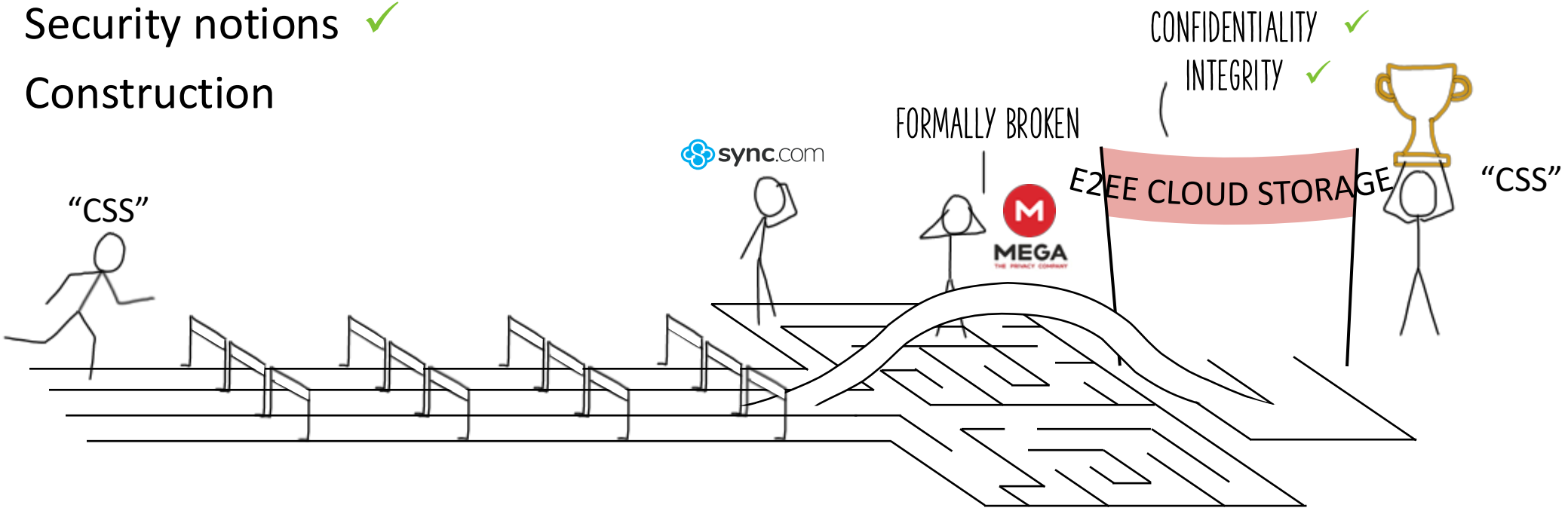
Are We Done?

- Syntax ✓
- Security notions ✓



Are We Done?

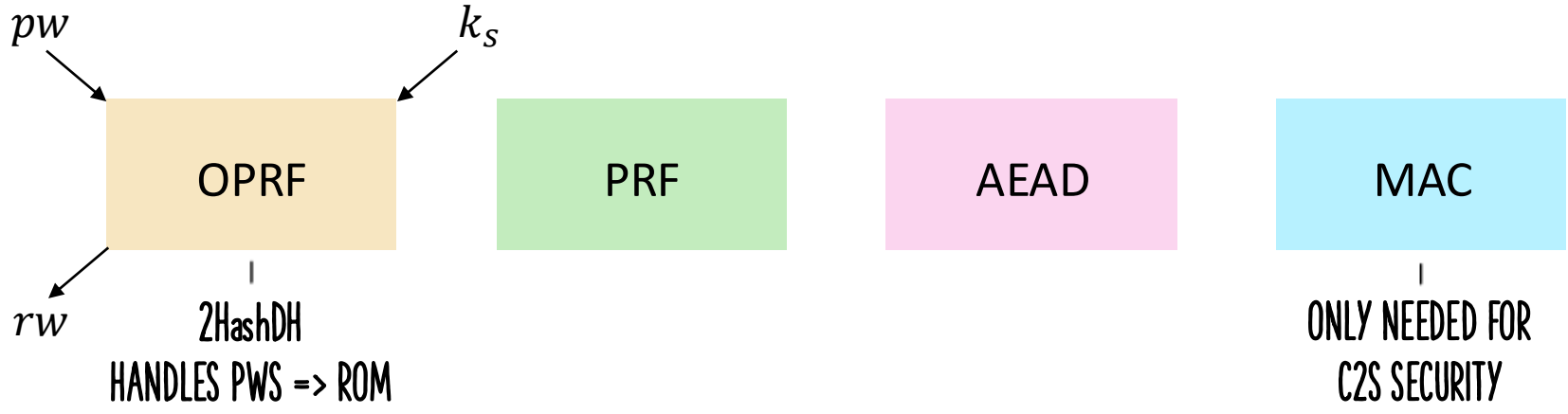
- Syntax ✓
- Security notions ✓
- Construction



2. Constructing E2EE Cloud Storage

CSS (Cloud Storage Scheme)

Building Blocks



CSS (Cloud Storage Scheme)

Registration

Client (pw, aid)

GLOBALLY UNIQUE

pw, aid

$k_s \xleftarrow{\$}$

CHOSEN BY CLIENT

OPRF

rw

PRF

k_{mac}

k_{kek}

$k_{mk} \xleftarrow{\$}$

AEAD.Enc

ALLOWS PW ROTATION
WITHOUT FILE RE-ENCRYPTION

$\lceil k_{mk} \rceil$

$aid, k_s, k_{mac}, \lceil k_{mk} \rceil$

Server

STORE

$User[aid] \leftarrow k_s, k_{mac}, \lceil k_{mk} \rceil$

CSS (Cloud Storage Scheme)

Authentication

GLOBALLY UNIQUE

Client (pw, aid)

Server

pw, aid

INTERACTIVE PROTOCOL

k_s

OPRF

rw

PRF

k_{kek}

k_{mac}

MAC.Tag

sid

τ

sid $

τ

k_{mac} τ sid MAC.Verify $\perp/\top$

AEAD.Dec $[k_{mk}]$

$[k_{mk}]$

k_{mk}

FETCH

$k_s, k_{mac}, [k_{mk}] \leftarrow User[aid]$

IF $\top$:

$Session[sid] \leftarrow aid$

STORE

$Cache \leftarrow k_{mk}, sid$

CSS (Cloud Storage Scheme)

Put

GLOBALLY UNIQUE

Client $(file, fid)$

$k_{mk}, sid \leftarrow Cache$

k_f $

$file$ → AEAD.Enc ← fid

$[file]$

BOUND BY ASSOCIATED DATA

k_{mk}

k_f → AEAD.Enc ← fid

$[k_f]$

$sid, fid, [file], [k_f]$ →

Server

FETCH

$aid \leftarrow Session[sid]$

STORE

$File[fid] \leftarrow [file]$

$Key[aid, fid] \leftarrow [k_f]$

SHARED

UNIQUE PER USER

CSS (Cloud Storage Scheme)

Share

*SIMPLIFIED

RECIPIENT ACCOUNT ID

Client $(fid, raid)$

$k_{mk}, sid \leftarrow Cache$

Server

$sid, fid, raid$

FETCH

$aid \leftarrow Session[sid]$

$[k_f] \leftarrow Key[aid, fid]$

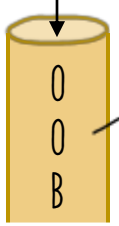
$[k_f]$

k_{mk}

$[k_f]$ → AEAD.Dec ← fid

k_f

SEND TO: $raid$

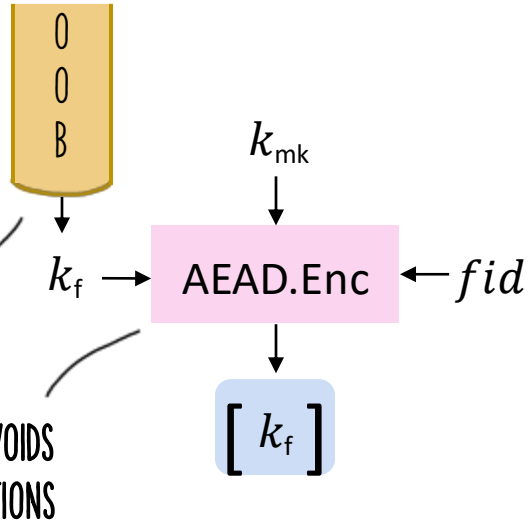


CSS (Cloud Storage Scheme)

Accept *SIMPLIFIED

Client (fid)

$k_{mk}, sid \leftarrow Cache$



$sid, fid, [k_f]$

Server

FETCH

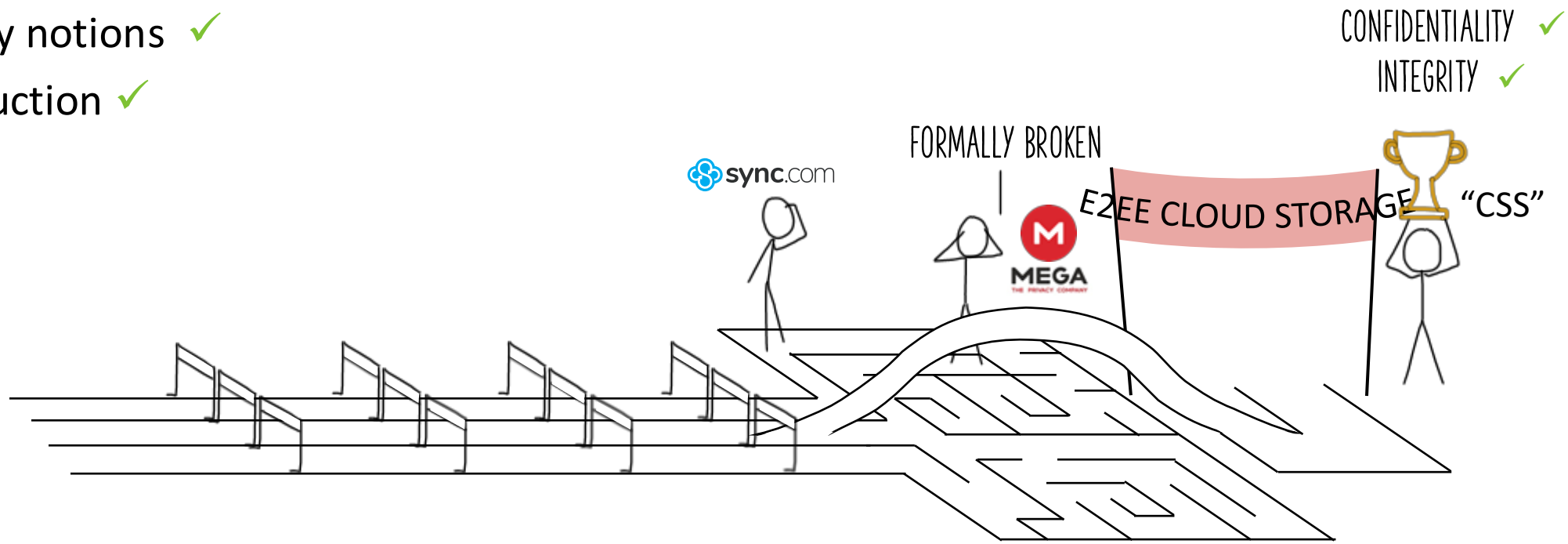
$aid \leftarrow Session[sid]$

STORE

$Key[aid, fid] \leftarrow [k_f]$

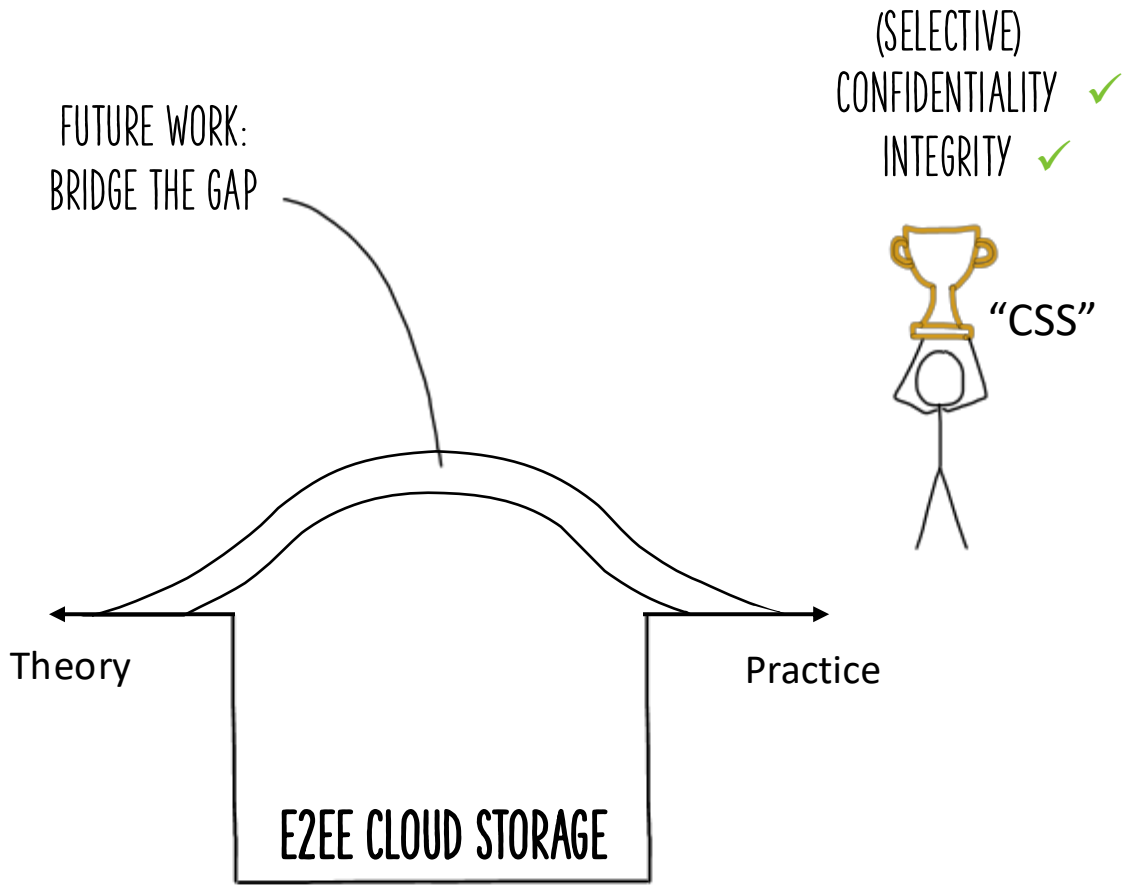
Are We Done?

- Syntax ✓
- Security notions ✓
- Construction ✓



Are We Done?

- Syntax ✓
- Security notions ✓
- Construction ✓

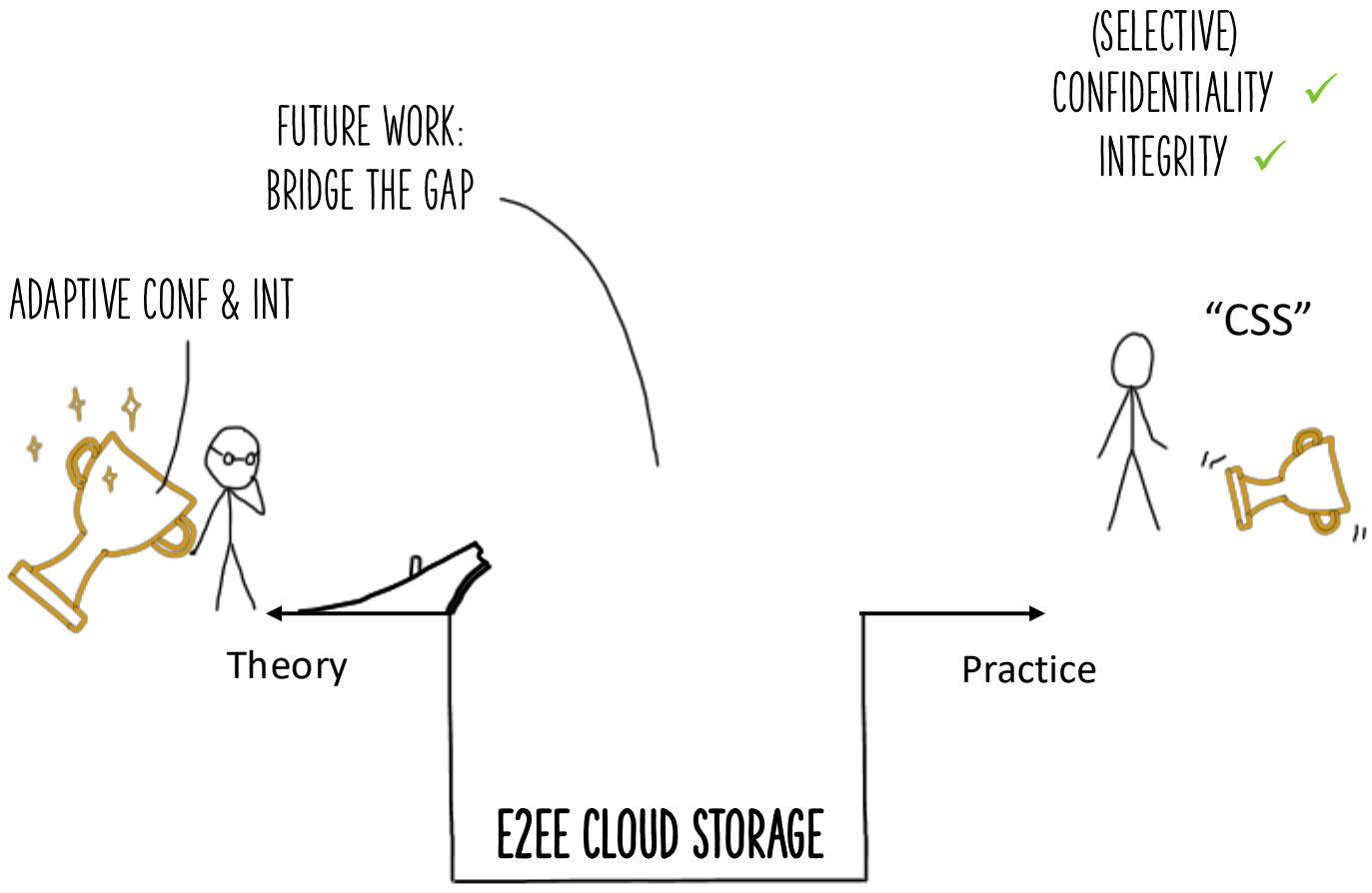


Are We Done?

- Syntax ✓
- Security notions ✓
- Construction ✓

Still missing:

- Adaptive security proof

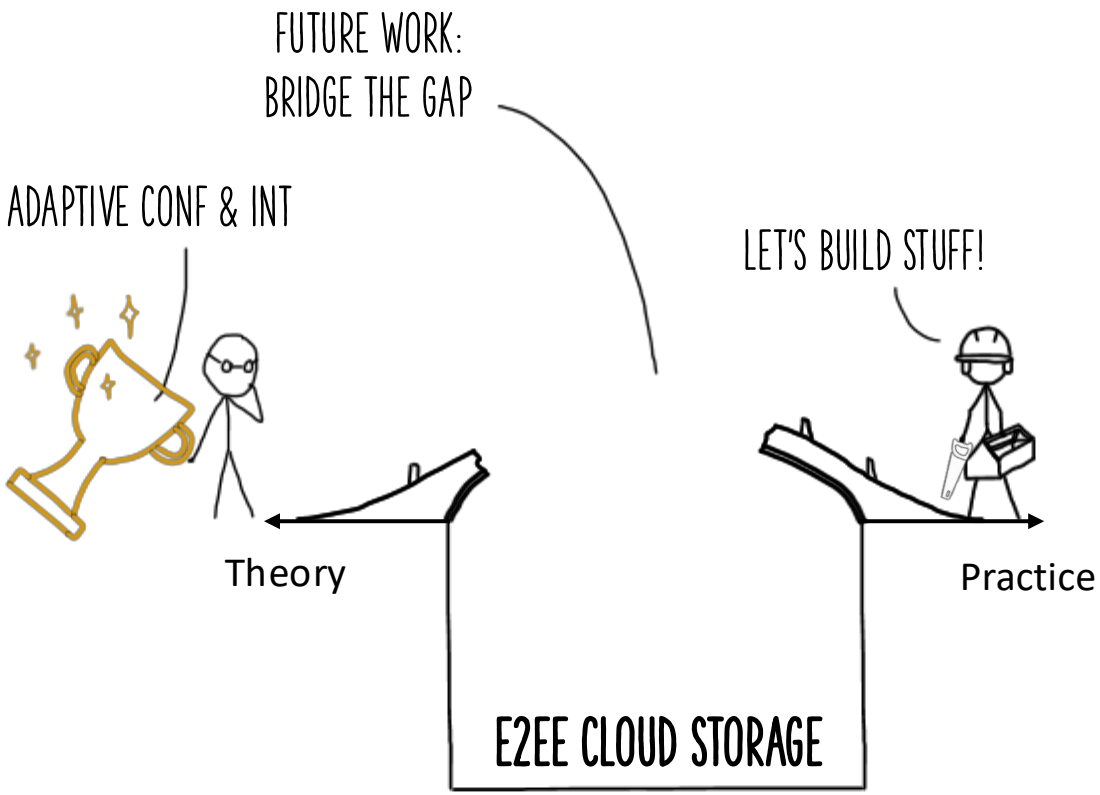


Are We Done?

- 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