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]

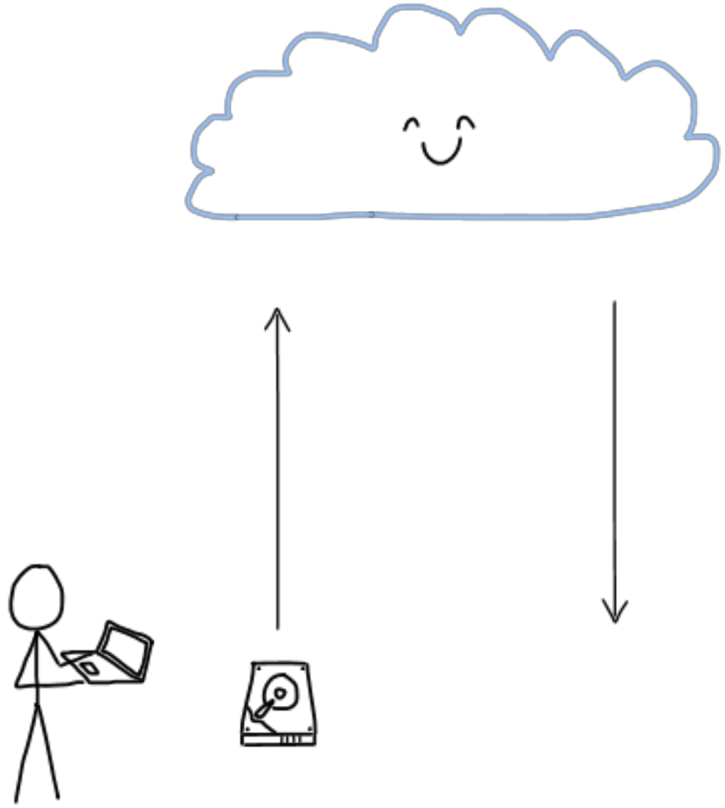
[1]ETH Zurich , [2]Seagate Technology, [3]IBM Research Zurich, [4]UC San Diego

Google, November 21, 2024

Cloud Storage

Benefits:

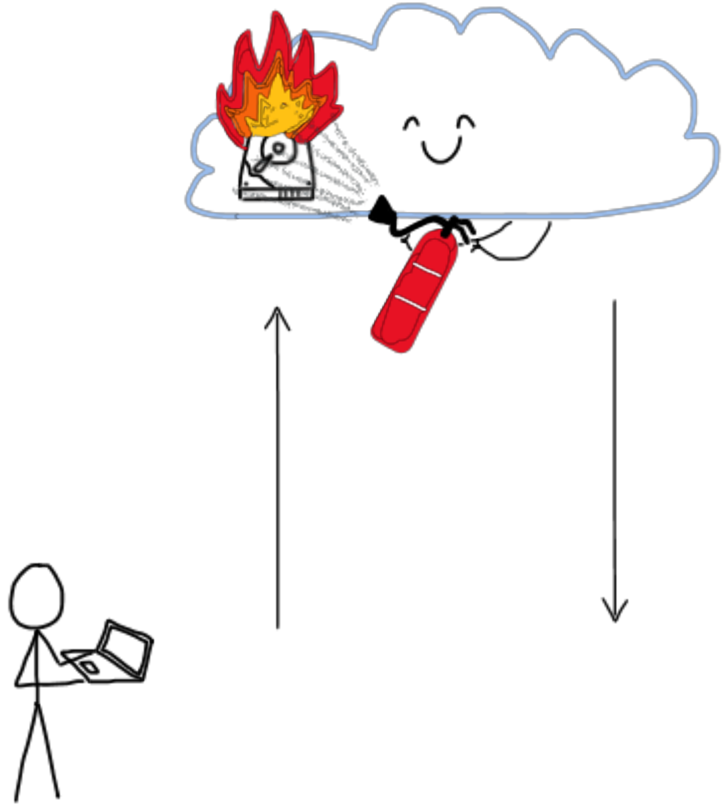
+ Availability



Cloud Storage

Benefits:

+ Availability
+ Redundancy



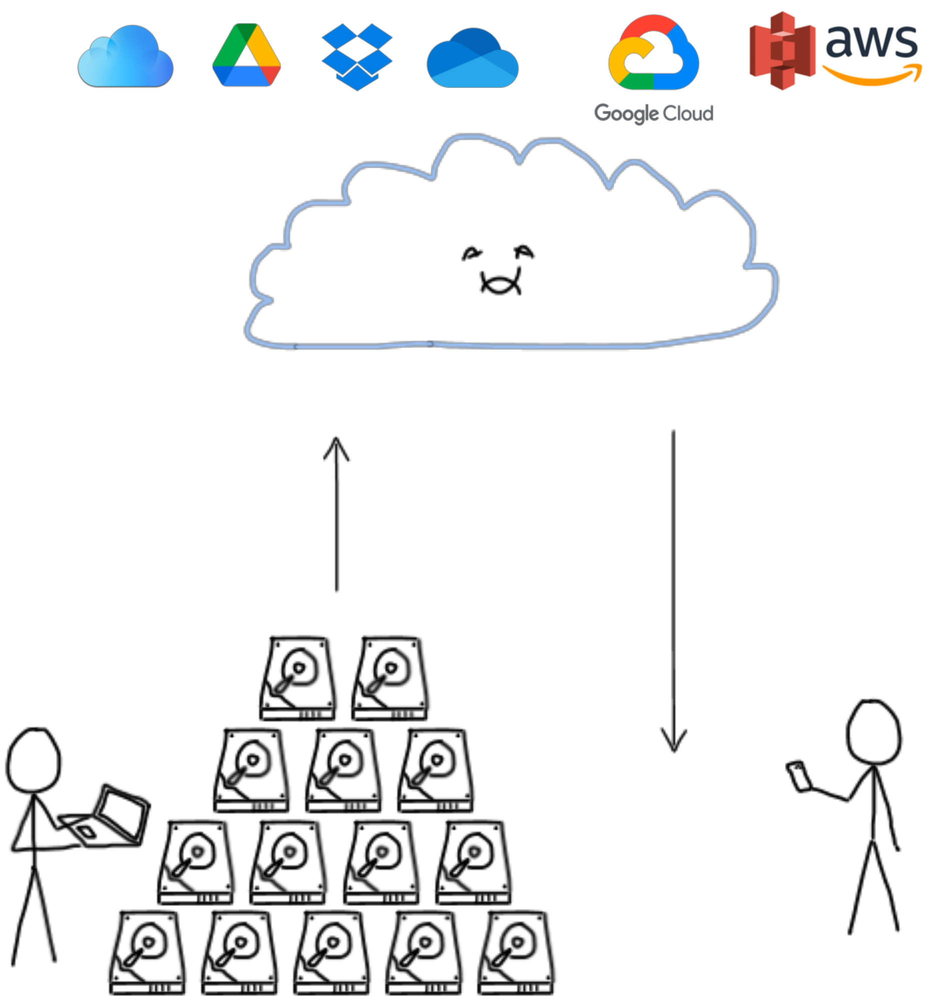
Cloud Storage

Benefits:

- + Availability
- + Redundancy
- + Scalability

Concerns:

- - Data leaks



STORING 50% OF ALL DATA BY 2025 [1]

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

Cloud Storage

Benefits:

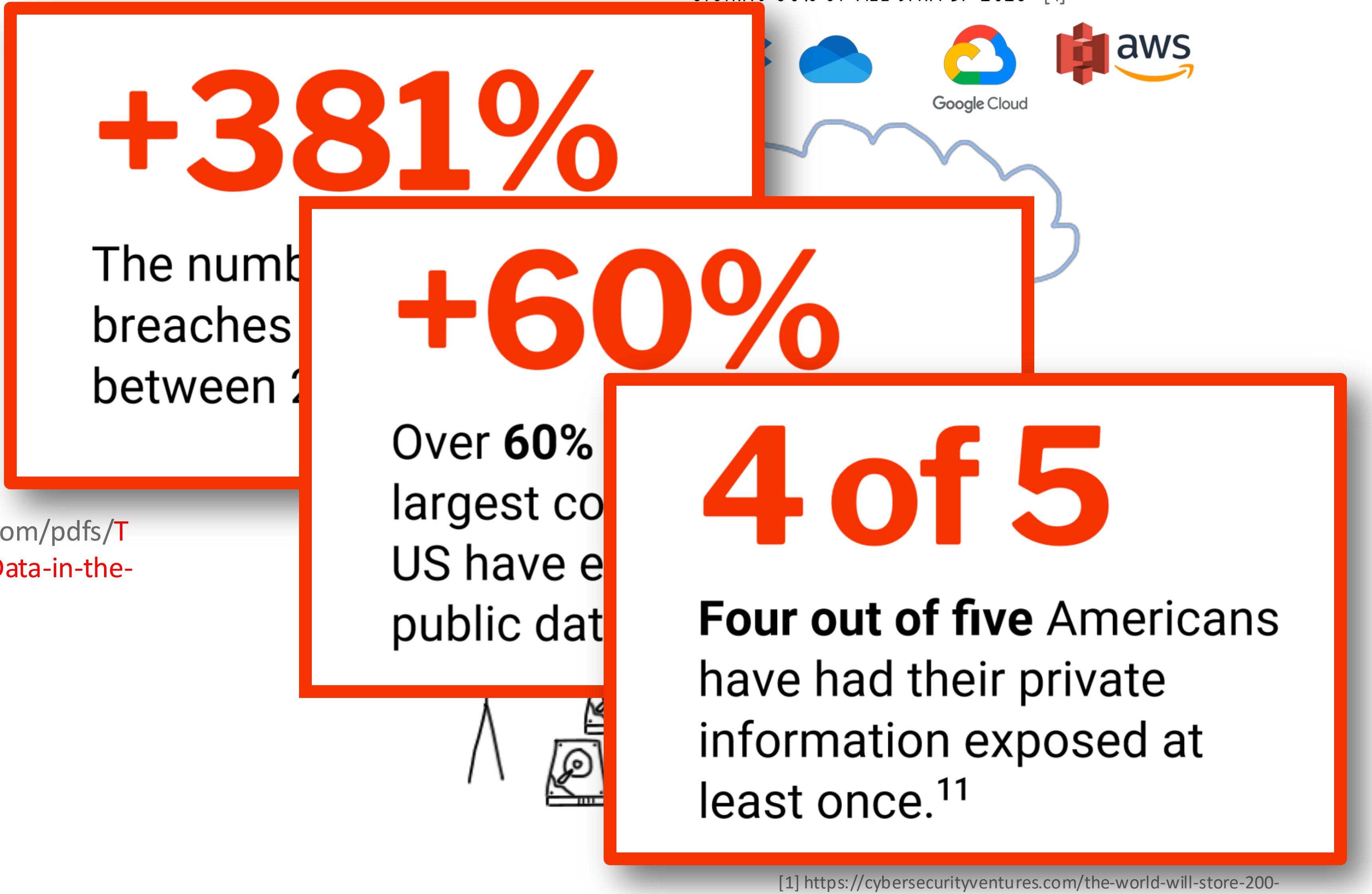
- + Availability
- + Redundancy
- + Scalability

Concerns:

- - Data leaks

https://www.apple.com/newsroom/pdfs/The-Rising-Threat-to-Consumer-Data-in-the-Cloud.pdf (December 2022)

STORING 50% OF ALL DATA BY 2025 [1]



Google Cloud
aws

+381%

The numb
breaches
between 2

+60%

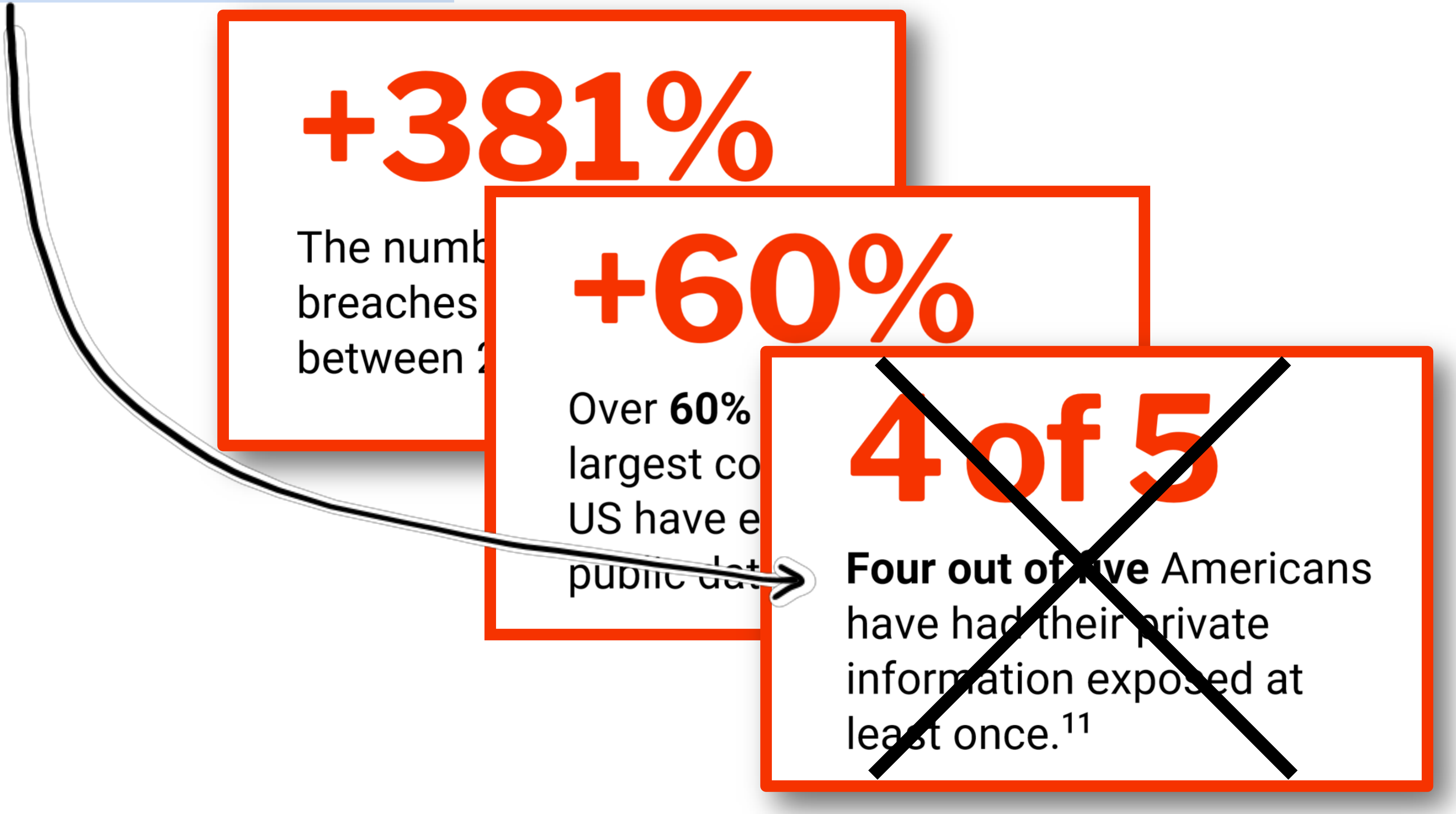
Over **60%**
largest co
US have e
public dat

4 of 5

Four out of five Americans have had their private information exposed at least once.[11]

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

Why E2E Security?



+381%

The numb
breaches
between 2

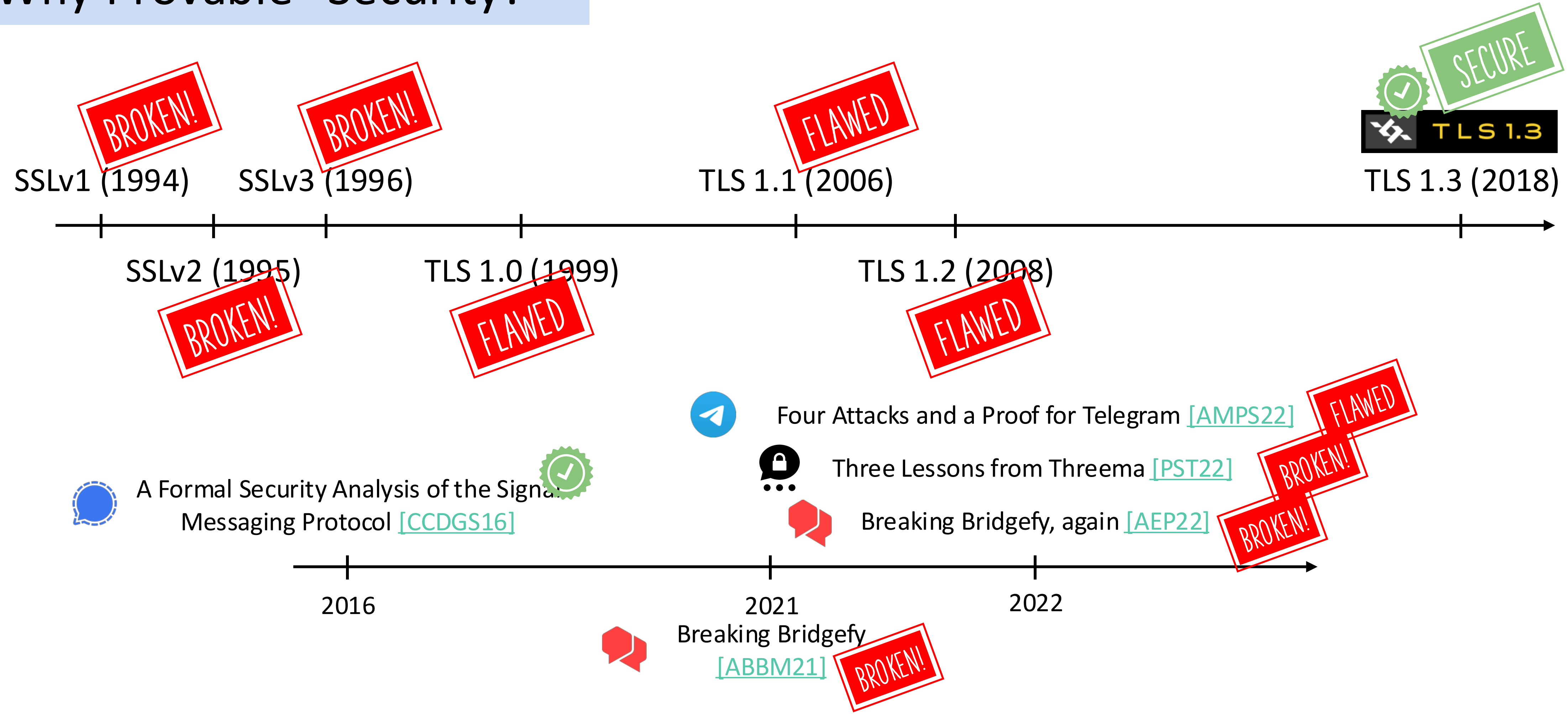
+60%

Over **60%**
largest co
US have e
public dat

~~4 of 5~~

~~**Four out of five** Americans have had their private information exposed at least once.[11]~~

Why Provable Security?



SSLv1 (1994) BROKEN!

SSLv2 (1995) BROKEN!

SSLv3 (1996) BROKEN!

TLS 1.0 (1999) FLAWED

TLS 1.1 (2006) FLAWED

TLS 1.2 (2008) FLAWED

TLS 1.3 (2018) SECURE

A Formal Security Analysis of the Signal Messaging Protocol [CCDGS16]

Breaking Bridgefy [ABBM21] BROKEN!

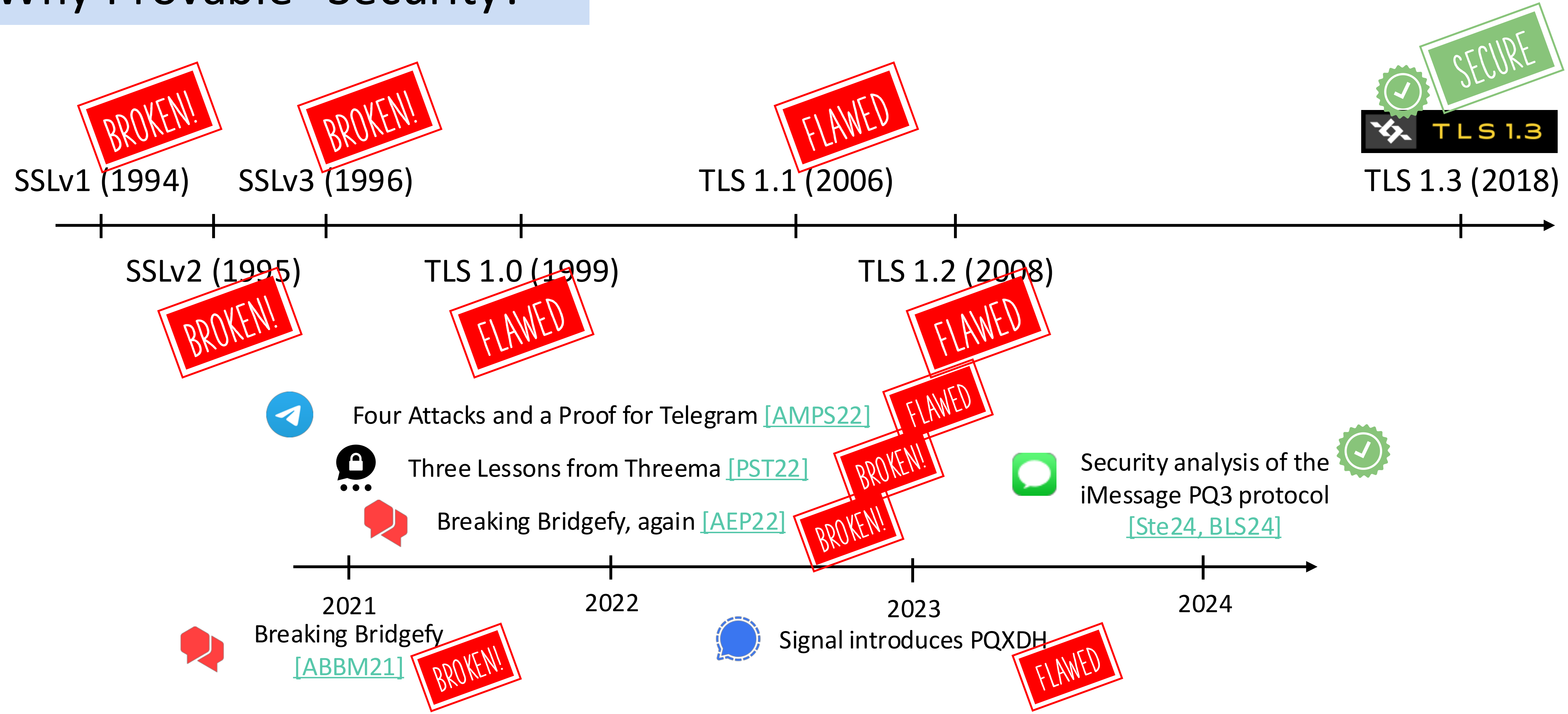
Four Attacks and a Proof for Telegram [AMPS22] FLAWED

Three Lessons from Threema [PST22] BROKEN!

Breaking Bridgefy, again [AEP22] BROKEN!

Logos from https://bridgefy.me/, fr.logodownload.org,
vecteezy.com, https://threema.ch/en/press &
https://commons.wikimedia.org/wiki/File:IMessage_logo.svgSecurity analysis of the iMessage PQ3 protocol

Why Provable Security?



SSLv1 (1994) BROKEN!

SSLv2 (1995) BROKEN!

SSLv3 (1996) BROKEN!

TLS 1.0 (1999) FLAWED

TLS 1.1 (2006) FLAWED

TLS 1.2 (2008) FLAWED

TLS 1.3 (2018) SECURE

2021

Breaking Bridgefy [ABBM21] BROKEN!

Four Attacks and a Proof for Telegram [AMPS22] FLAWED

Three Lessons from Threema [PST22] BROKEN!

Breaking Bridgefy, again [AEP22] BROKEN!

2022

2023

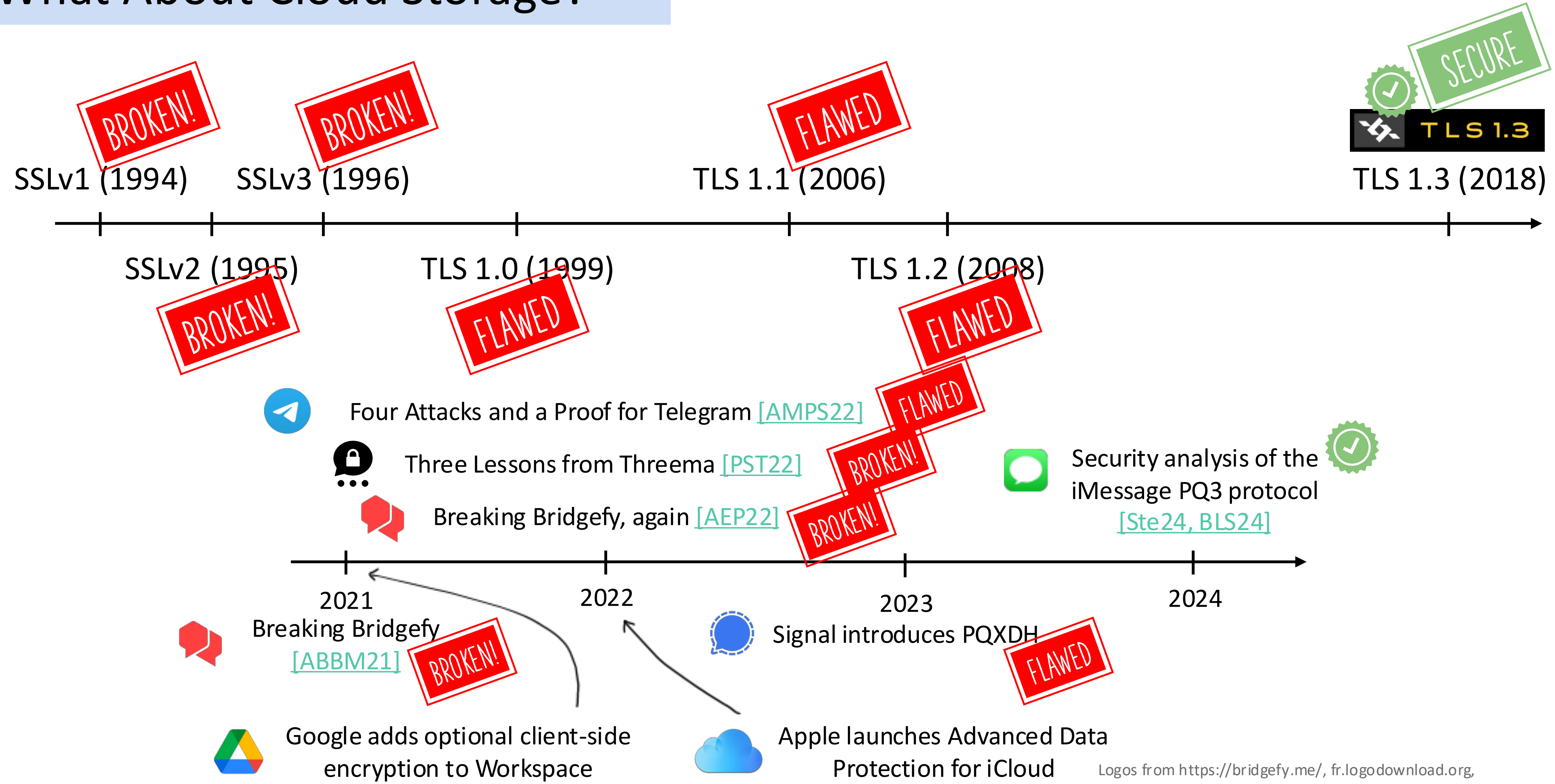
Signal introduces PQXDH FLAWED

2024

Security analysis of the iMessage PQ3 protocol [Ste24, BLS24] ✓

Logos from https://bridgefy.me/, fr.logodownload.org, vecteezy.com, https://threema.ch/en/press & https://commons.wikimedia.org/wiki/File:IMessage_logo.svgSecurity analysis of the iMessage PQ3 protocol

What About Cloud Storage?



Four Attacks and a Proof for Telegram [AMPS22]

Three Lessons from Threema [PST22]

Breaking Bridgefy, again [AEP22]

Security analysis of the iMessage PQ3 protocol [Ste24, BLS24]

Breaking Bridgefy [ABBM21]

Signal introduces PQXDH

Google adds optional client-side encryption to Workspace

Apple launches Advanced Data Protection for iCloud

Logos from https://bridgefy.me/, fr.logodownload.org, vecteezy.com, https://threema.ch/en/press & https://commons.wikimedia.org/wiki/File:IMessage_logo.svgSecurity analysis of the iMessage PQ3 protocol

E2EE Cloud Storage Providers

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

300 MILLION USERS

MEGA

THE GERMAN FEDERAL GOVERNMENT,
AMNESTY INTERNATIONAL,
& ETH Zurich

"ULTIMATE SECURITY"

Nextcloud

"FREE, ENCRYPTED, AND SECURE CLOUD STORAGE.
YOUR PRIVACY, SECURED BY MATH"

Proton Drive

"EXCEPTIONALLY PRIVATE CLOUD"

sync.com

"EUROPE'S MOST SECURE CLOUD STORAGE"

pCloud

icedrive

"THE STRONGEST ENCRYPTED
CLOUD STORAGE IN THE WORLD"

Seafile™

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

Case Studies: E2EE Cloud Storage

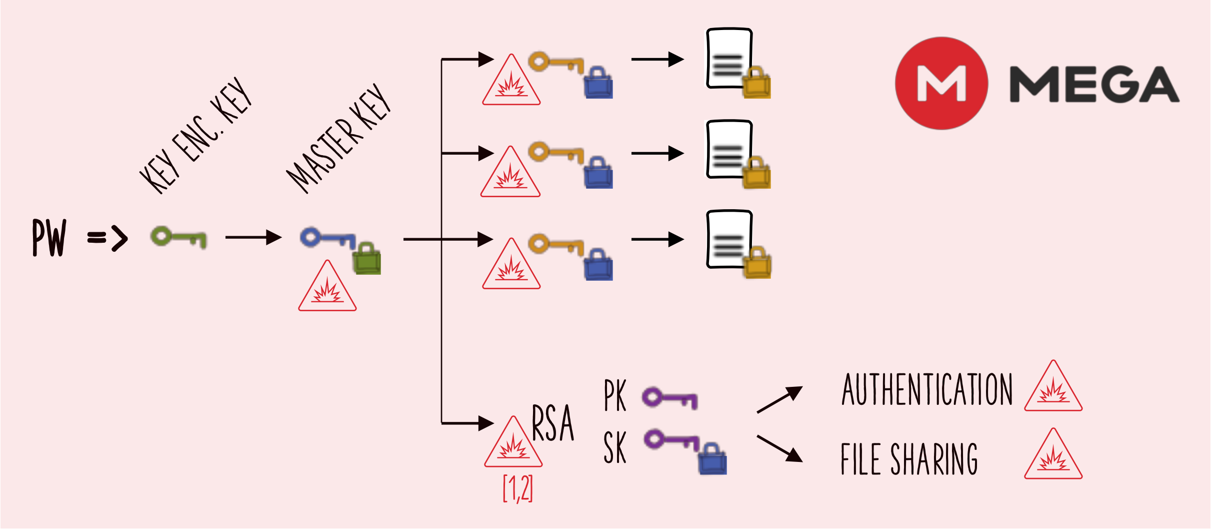
Challenges:

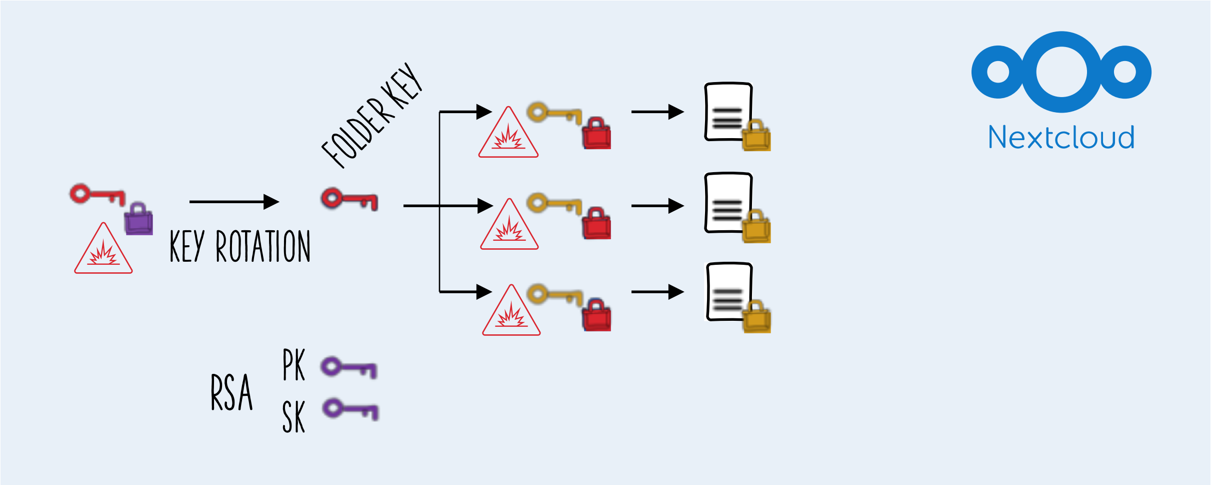
1. Stateless clients
2. No ciphertext integrity
3. Key recovery attacks [1,2]
4. Key reuse
5. File re-encryption infeasible
6. PKE has no authentication [3]

[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.

[3] Martin R. Albrecht, Matilda Backendal, Daniele Coppola, Kenneth G. Paterson. (2024). “Share with Care: Breaking E2EE in Nextcloud”. Euro S&P 2024.





Case Studies: E2EE Cloud Storage

… is surprisingly hard!

Challenges:

1 Stateless clients

2 No ciphertext integrity

3 Key recovery attacks [1,2]

4 Key reuse

5 File re-encryption infeasible

6 PKE has no authentication [3]

Implications:

- Design issues 2 4
- Password-based security 1
- Key distribution problem 1
- File sharing causes complex interactions 3 6
- Need to get it right the first time 5

[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.

[3] Martin R. Albrecht, Matilda Backendal, Daniele Coppola, Kenneth G. Paterson. (2024). “Share with Care: Breaking E2EE in Nextcloud”. Euro S&P 2024.

E2EE Cloud Storage Providers

"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]

"FREE, ENCRYPTED, AND SECURE CLOUD STORAGE.
YOUR PRIVACY, SECURED BY MATH"

Proton Drive

NOT PROVABLY SECURE

"EXCEPTIONALLY PRIVATE CLOUD"

sync.com

"EUROPE'S MOST SECURE CLOUD STORAGE"

pCloud

icedrive

"THE STRONGEST ENCRYPTED CLOUD STORAGE IN THE WORLD"

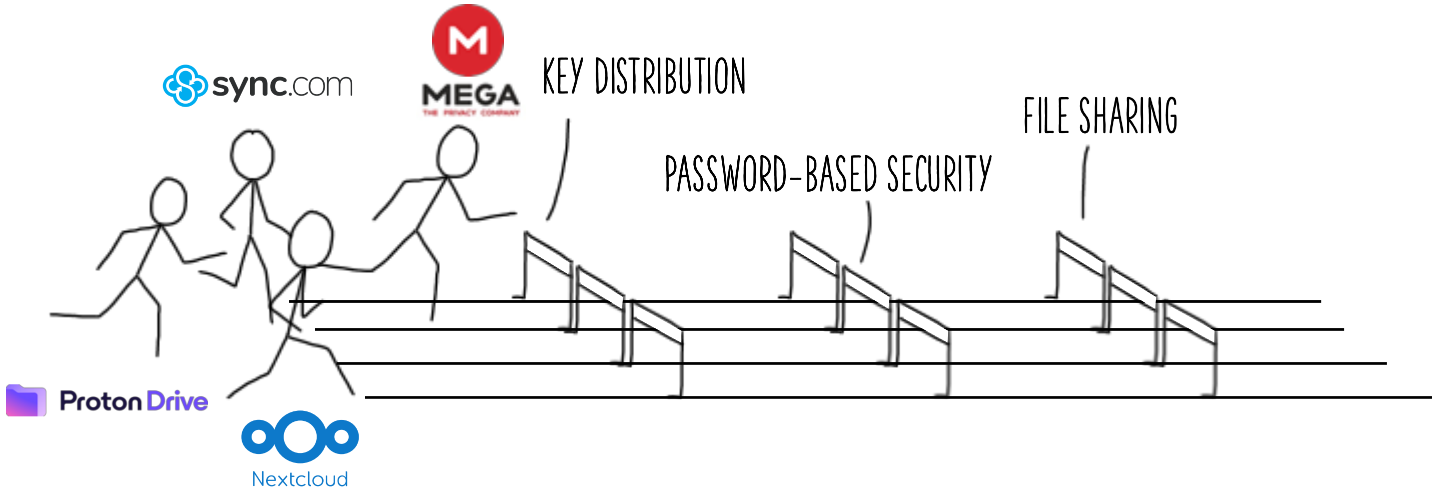
Seafile™

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

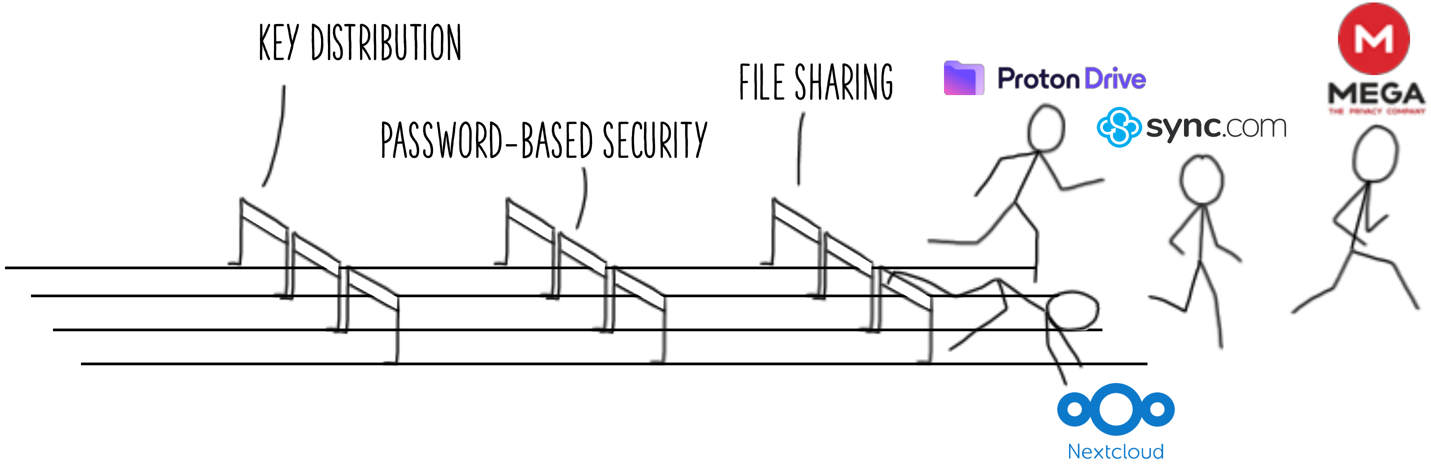
INSECURE!

[CCS:TH24]

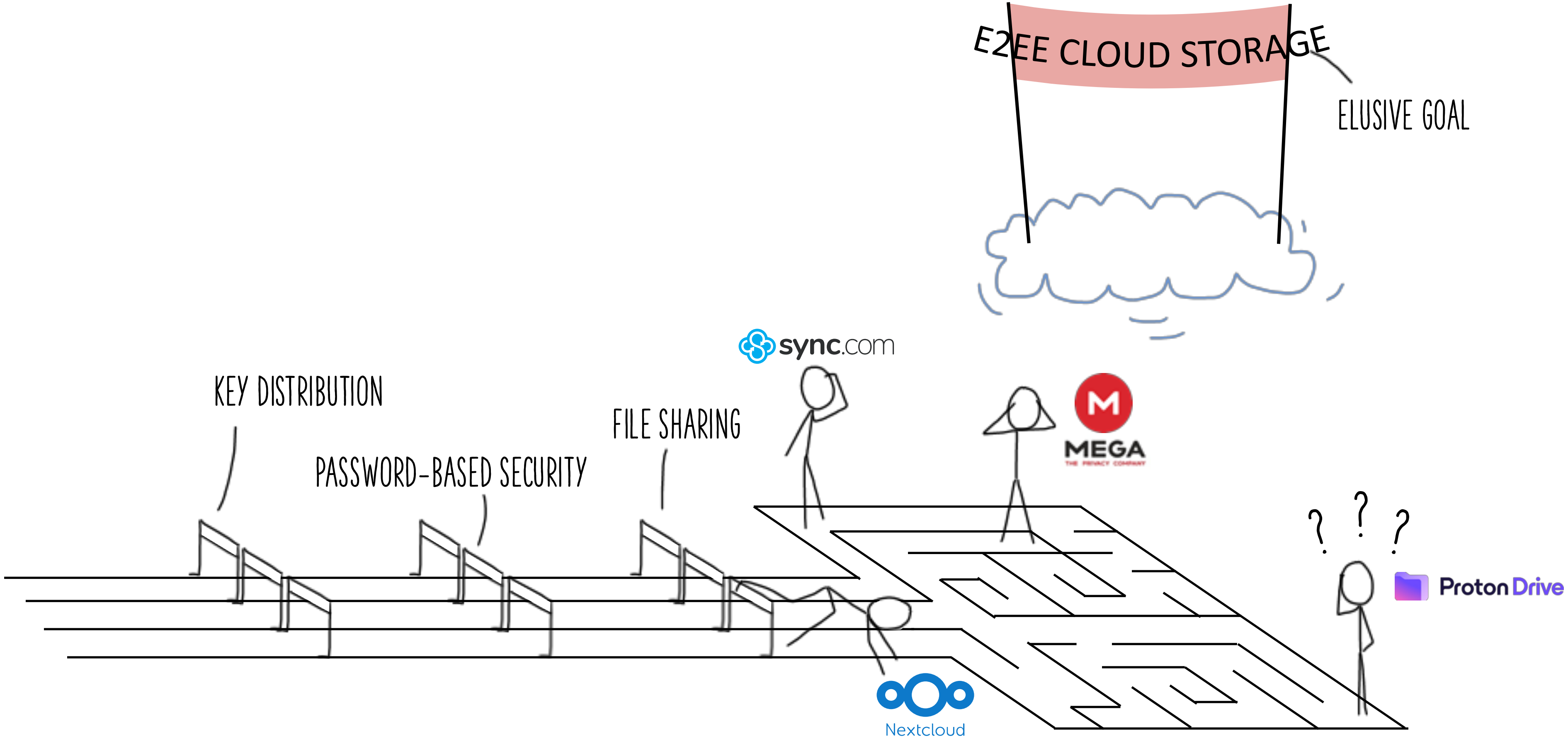
Why Is It Hard?



Why Is It Hard?



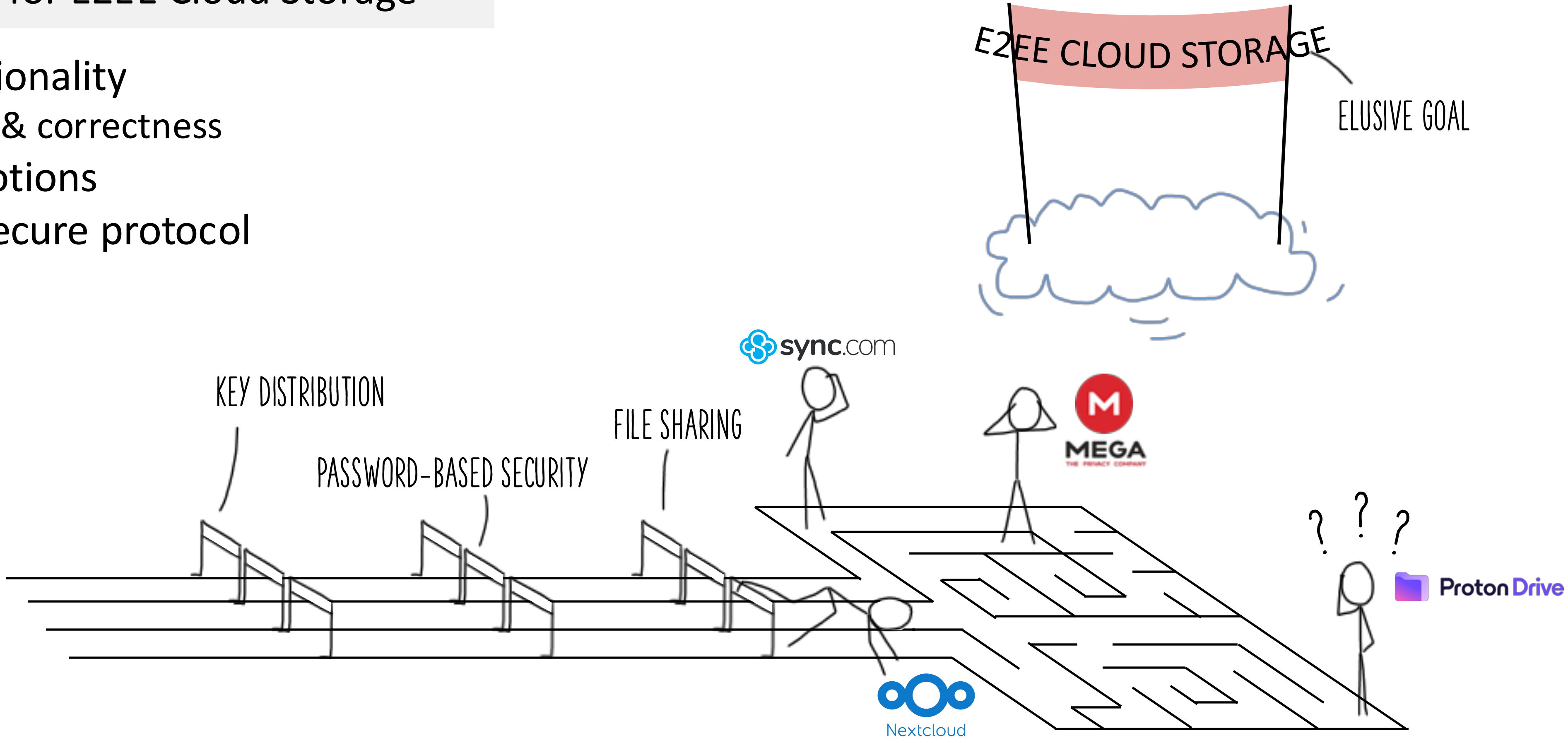
Why Is It Hard?



Our Work

Formal Model for E2EE Cloud Storage

- Core functionality
  → Syntax & correctness
- Security notions
- Provably secure protocol



1. Formalizing E2EE Cloud Storage

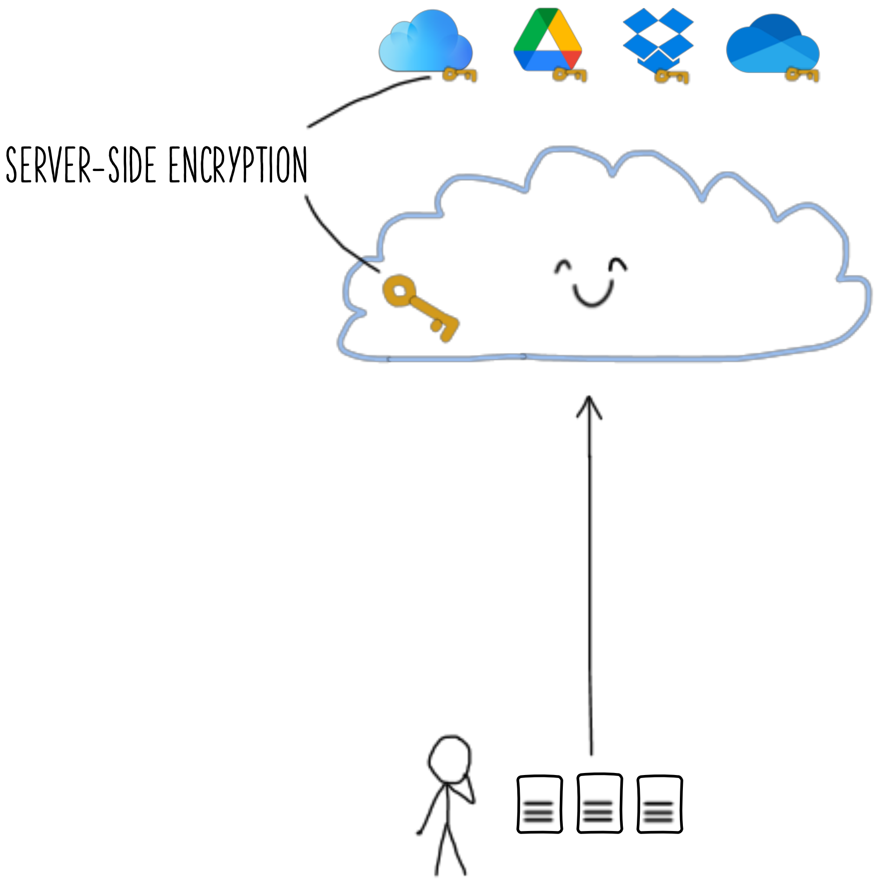
Formalizing E2EE Cloud Storage

Goal:

- Secure data at rest
- …with maximal functionality

Methods:

- Server-side encryption
  - + Plaintext access -> features
  - - Plaintext access -> less privacy



Formalizing E2EE Cloud Storage

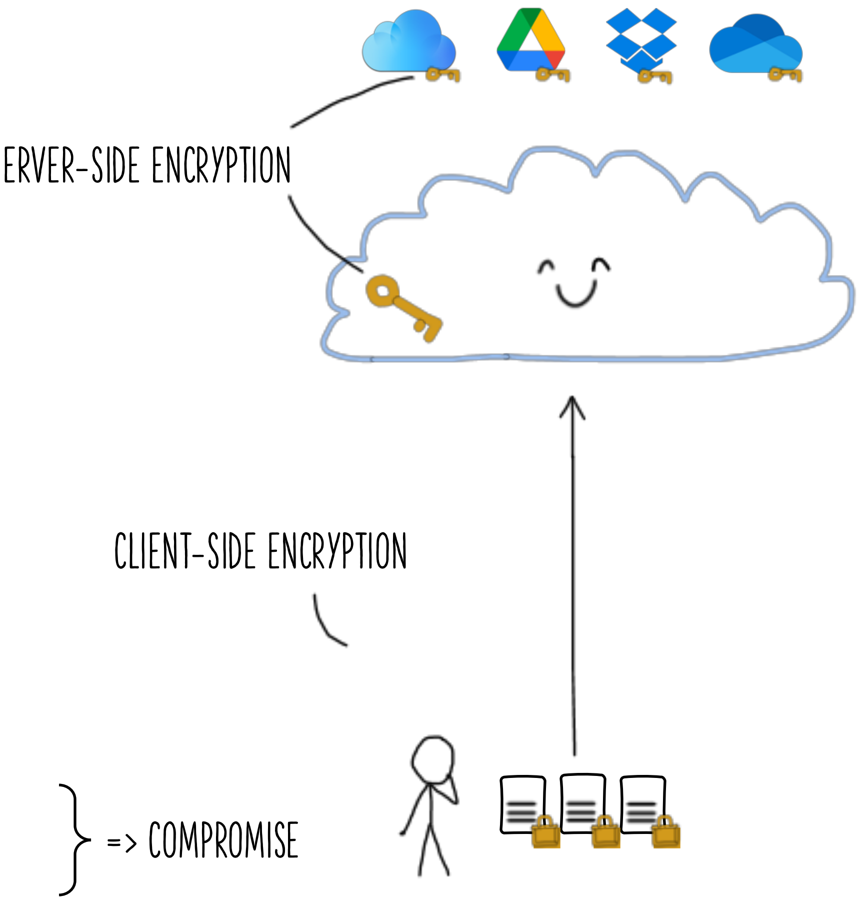
Goal:

- Secure data at rest
- …with maximal functionality
- …against a compromised server

Methods:

- Server-side encryption
  - + Plaintext access -> features
  - - Plaintext access -> less privacy

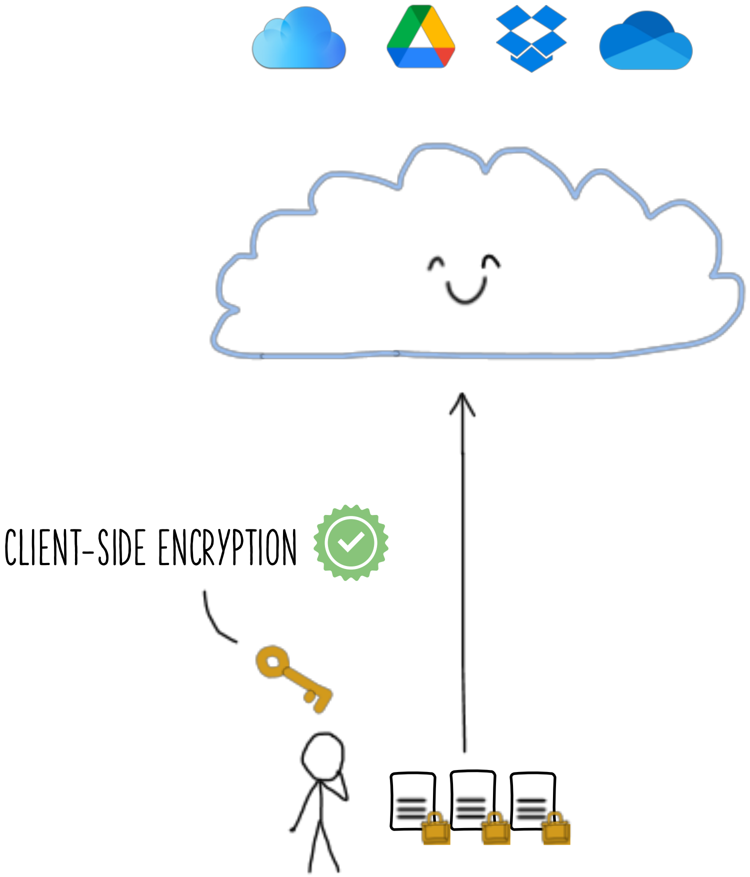
- End-to-end encryption
  - + No plaintext access -> privacy
  - - No plaintext access -> less features?



Formalizing E2EE Cloud Storage

In scope:

🟩 Provable security

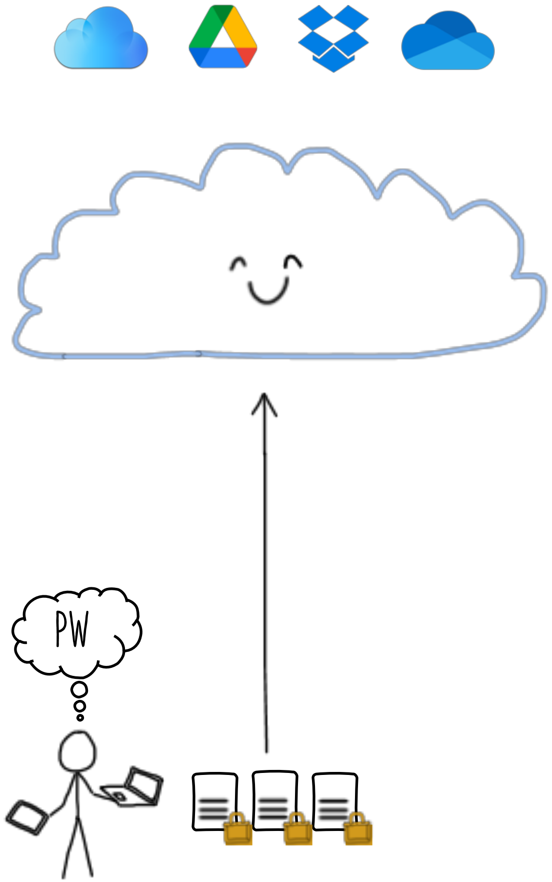


CLIENT-SIDE ENCRYPTION

Formalizing E2EE Cloud Storage

In scope:

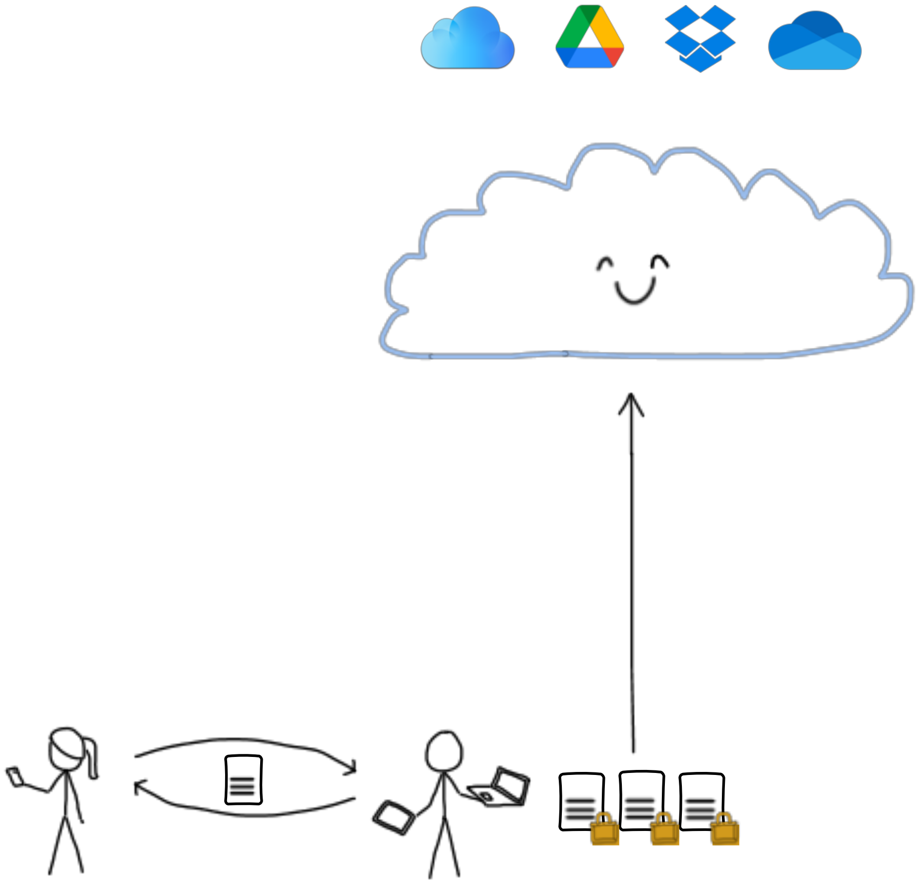
- 🟩 Provable security
- 🟩 Multi-device access



Formalizing E2EE Cloud Storage

In scope:

- Provable security
- Multi-device access
- File sharing



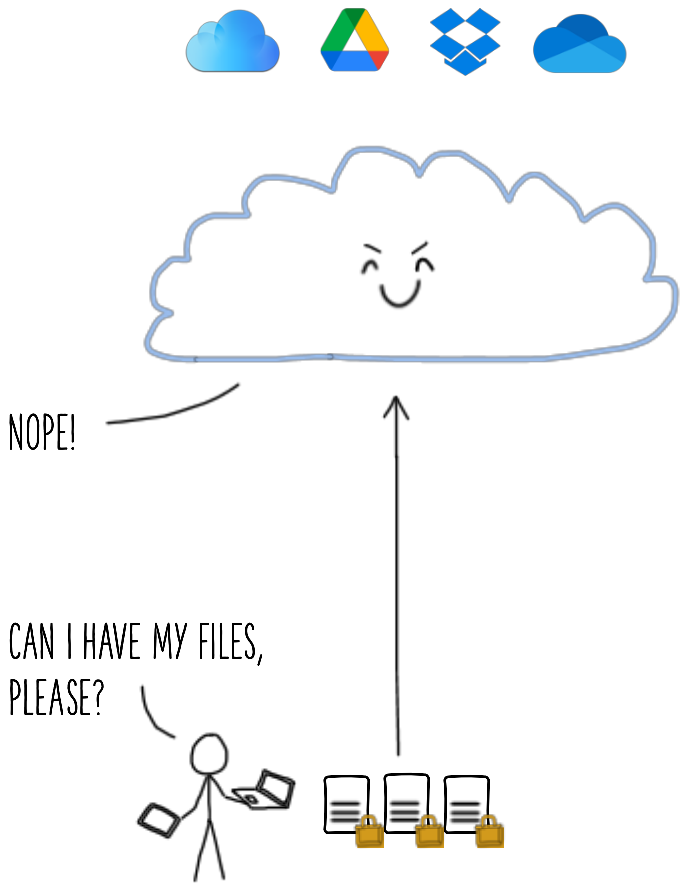
Formalizing E2EE Cloud Storage

In scope:

- 🟩 Provable security
- 🟩 Multi-device access
- 🟩 File sharing

Out of scope:

- 🟥 Availability



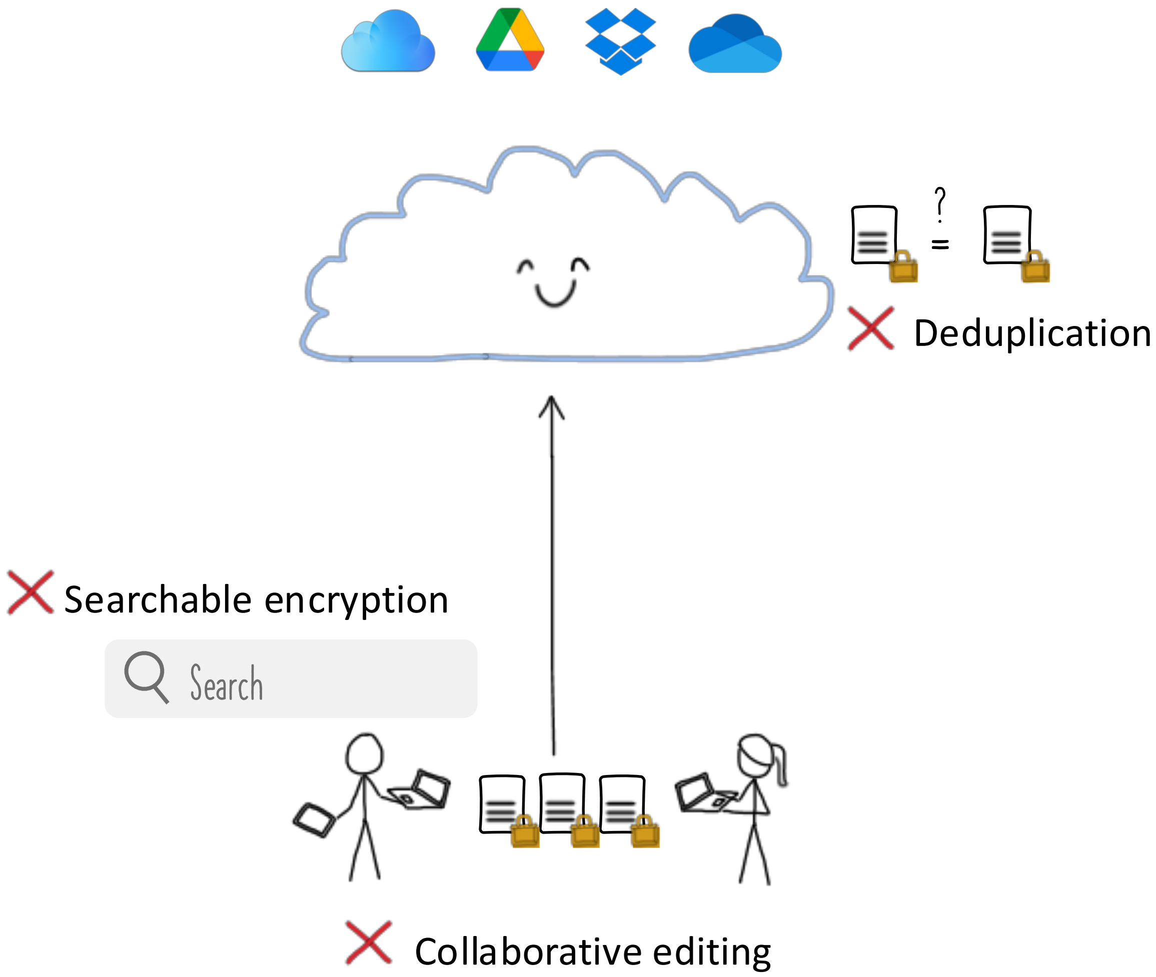
Formalizing E2EE Cloud Storage

In scope:

- 🟩 Provable security
- 🟩 Multi-device access
- 🟩 File sharing

Out of scope:

- 🟥 Availability
- 🟥 Server-side processing



✗ Deduplication

✗ Searchable encryption

✗ Collaborative editing

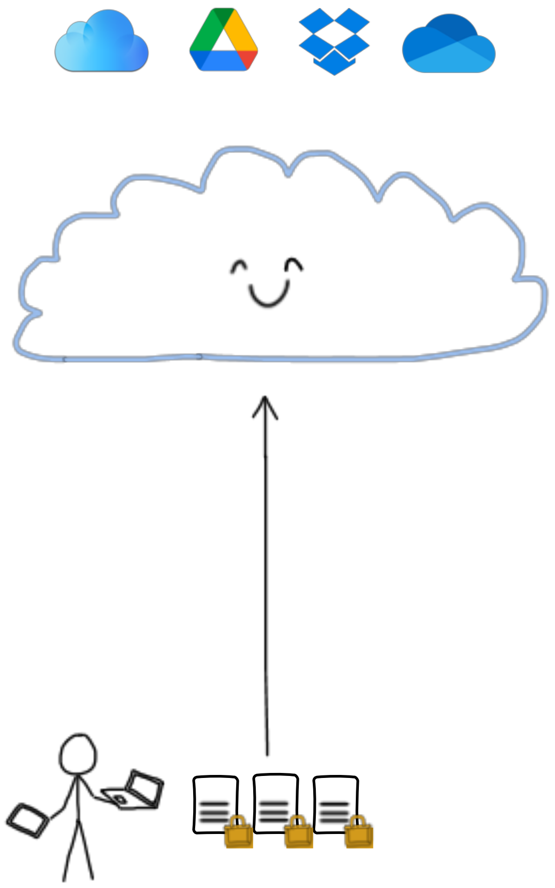
Formalizing E2EE Cloud Storage

In scope:

- 🟩 Provable security
- 🟩 Multi-device access
- 🟩 File sharing

Out of scope:

- 🟥 Availability
- 🟥 Server-side processing
- 🟥 Advanced Security
  - Metadata & access pattern hiding
  - Revocable access
  - Forward secrecy
  - …



Formalizing E2EE Cloud Storage

Model Goals

ALL MODELS ARE WRONG, BUT SOME ARE USEFUL!

MEGA

Nextcloud

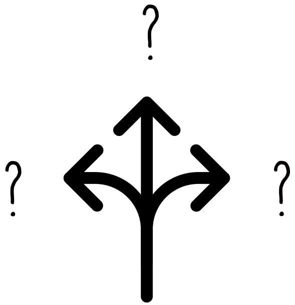
sync.com

Proton Drive

icedrive

pCloud

Seafile



Capture existing systems

Capture *real-world* systems

Capture future systems

1	Expressive
2	Faithful
3	Generic

Syntax

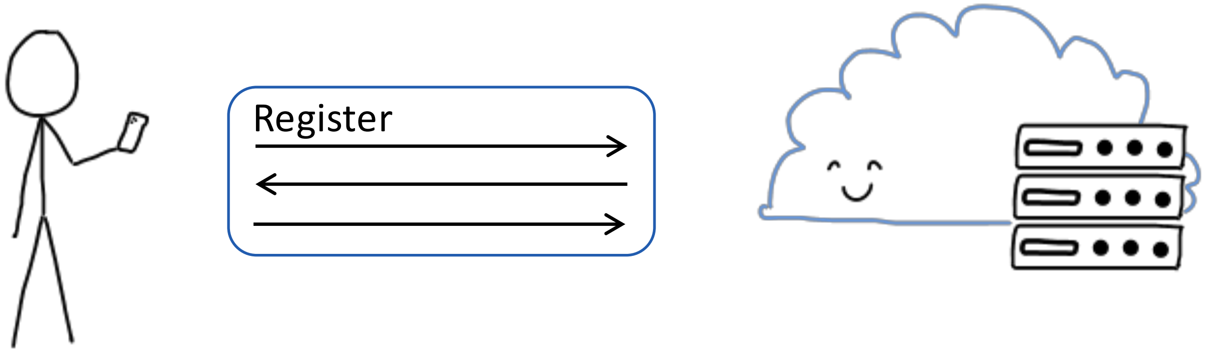
WHAT MAKES A CLOUD STORAGE A CLOUD STORAGE?

Core Functionality

? Anything missing?

- 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?

? Anything missing?

Core Functionality

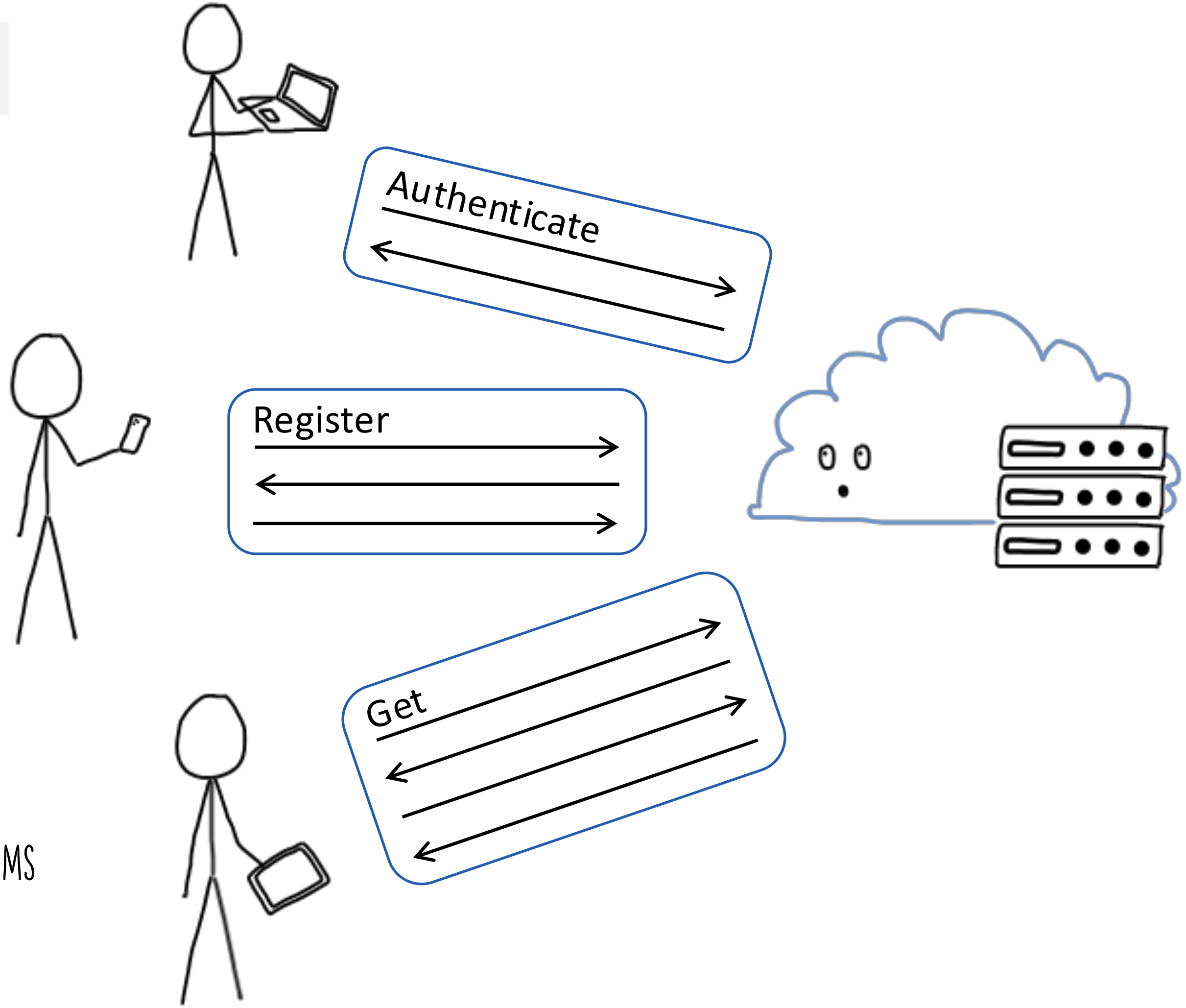
- Register (create account)
- Authenticate (log in)

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

} INTERACTIVE PROTOCOLS

Model Choices

- Non-atomic operations → FAITHFUL TO REAL-WORLD SYSTEMS



Syntax

HOW DO WE MAKE THE MODEL USEFUL?

Core Functionality

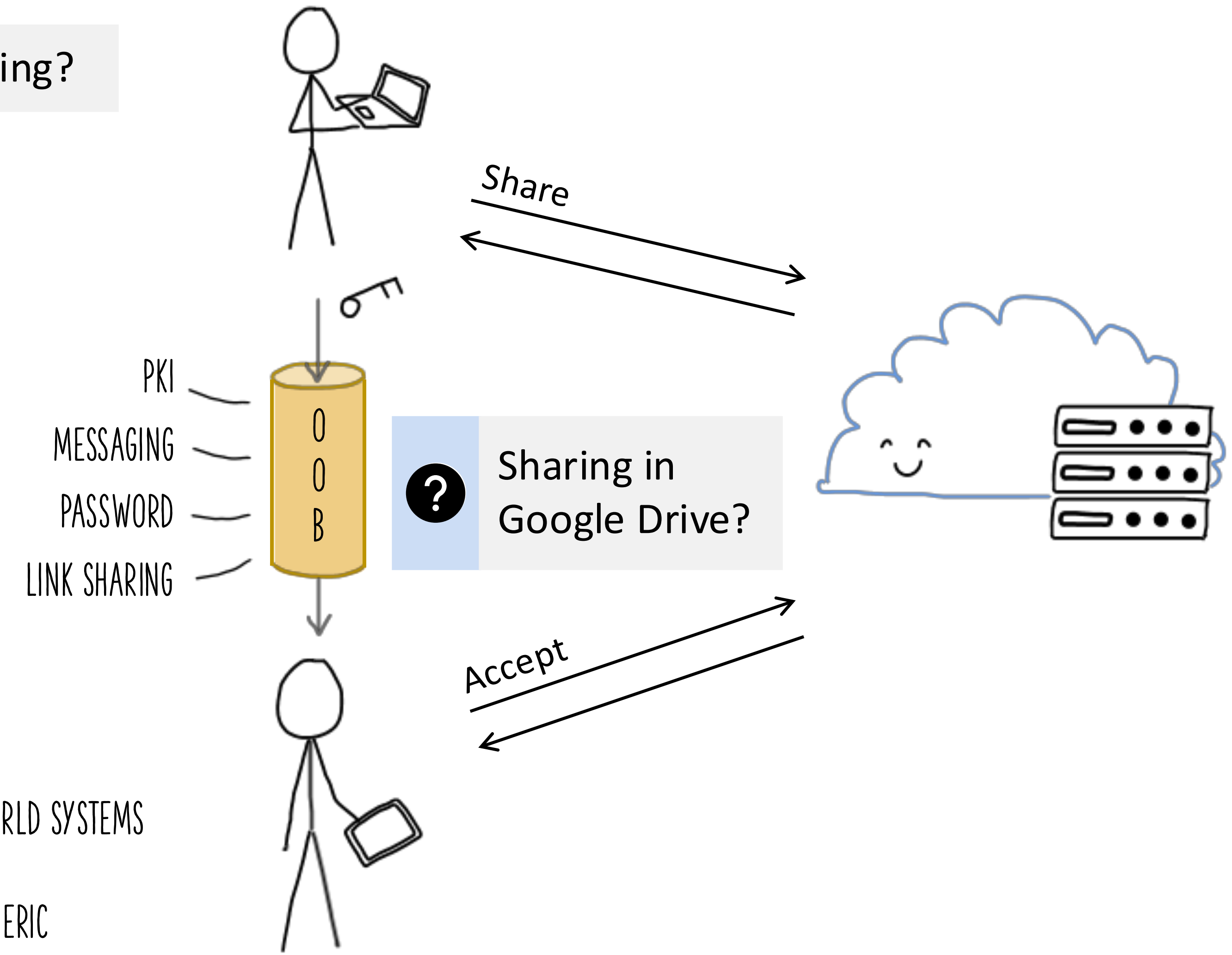
- Register (create account)
- Authenticate (log in)
- Put (upload a file)
- Update (modify content)
- Get (download)
- Share
- Accept (receive share)

INTERACTIVE PROTOCOLS

? Anything missing?

Model Choices

- Non-atomic operations → FAITHFUL TO REAL-WORLD SYSTEMS
- Abstract OOB channel for sharing → GENERIC



? Sharing in Google Drive?

Security Notions

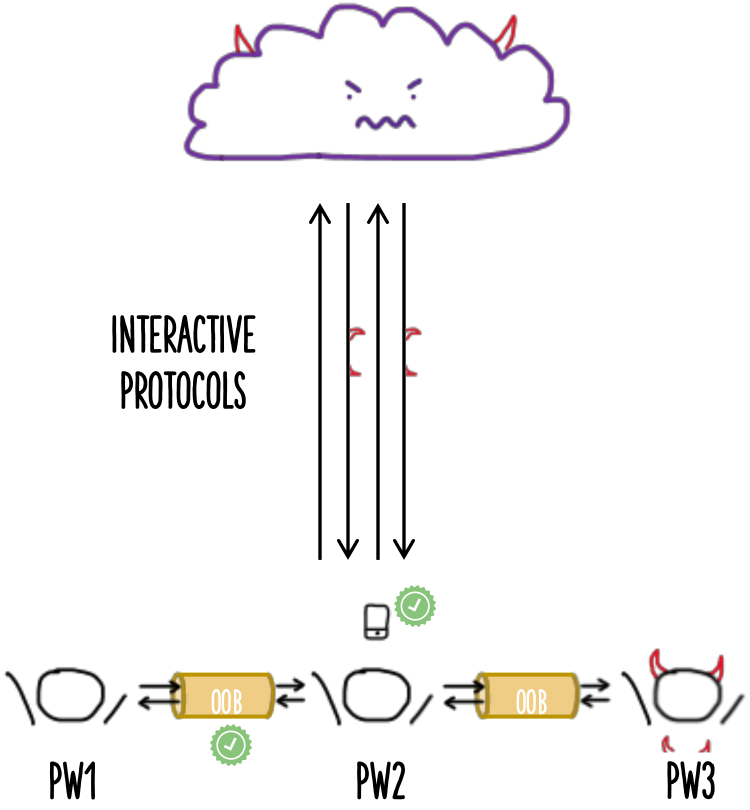
MALICIOUS SERVER SETTING

Threat model:

- Malicious cloud provider
- Trusted OOB-channels between honest users
- Trusted client code

Adversary capabilities:

- Control client protocol steps (which & when)
- Specify server responses
- Guess honest user passwords
- Compromise users (adaptive/selective)



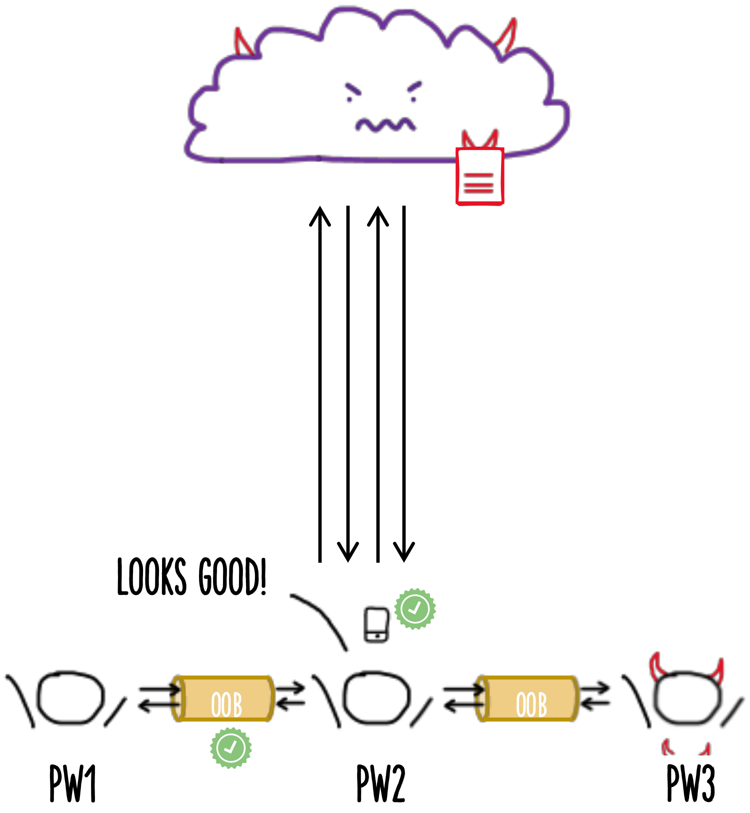
Security Notions

MALICIOUS SERVER SETTING

Integrity:

- Wins if adversary can, for an honest user,
  1. inject a file, or
  2. modify a file.

INT-PTXT-STYLE GAME



Security Notions

MALICIOUS SERVER SETTING

Integrity:

- Wins if adversary can, for an honest user,
  1. inject a file, or
  2. modify a file.

INT-PTXT-STYLE GAME

! Not INT-CTXT

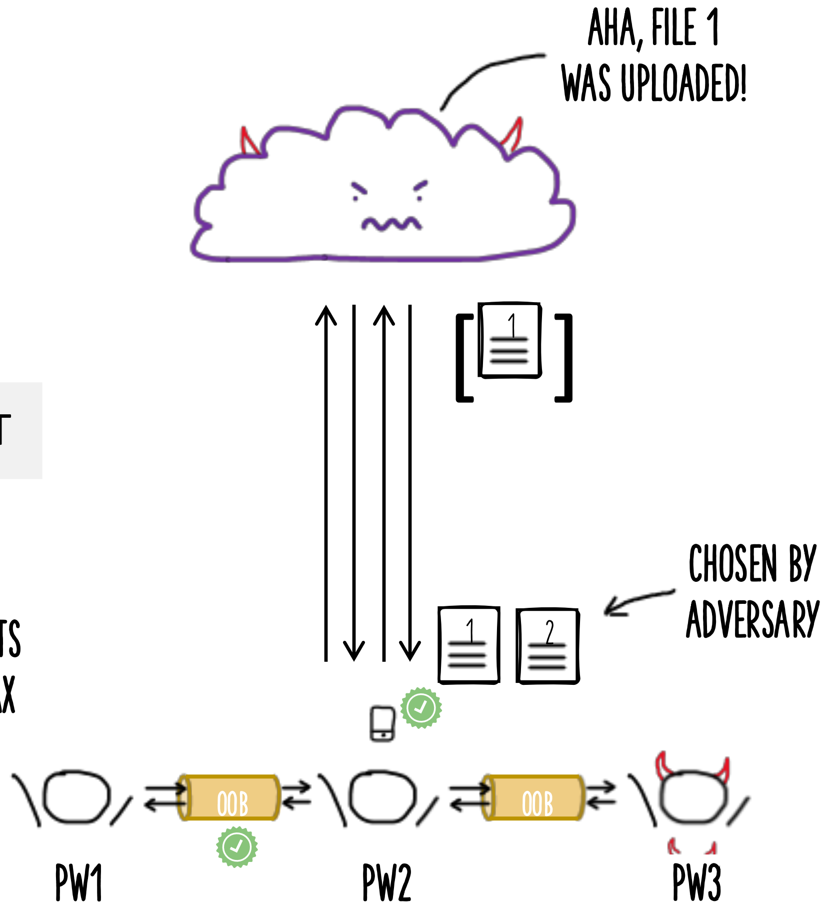
NO CIPHERTEXTS IN OUR SYNTAX

Confidentiality:

- Wins if adversary can, for an honest user,
  - learn any information and distinguish files

IND-CCA-STYLE GAME

! Not IND$



Security Notions

MALICIOUS CLIENT SETTING [ONGOING WORK]

Threat model:

- ~~Malicious~~ honest cloud provider, malicious clients
- Trusted OOB-channels between honest users
- Trusted client code

+ Trusted client-to-server channels?

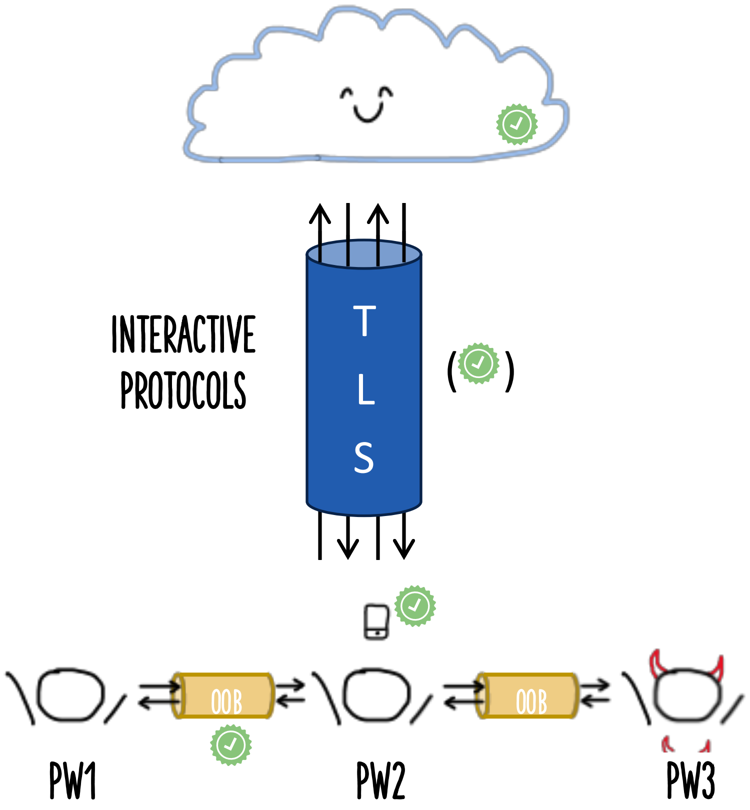
Adversary capabilities:

- Control client protocol steps (which & when)
- ~~Specify server responses~~
- Guess honest user passwords
- Compromise users (adaptive/selective)

Additional goals:

INFEASIBLE IN THE MALICIOUS SERVER SETTING!

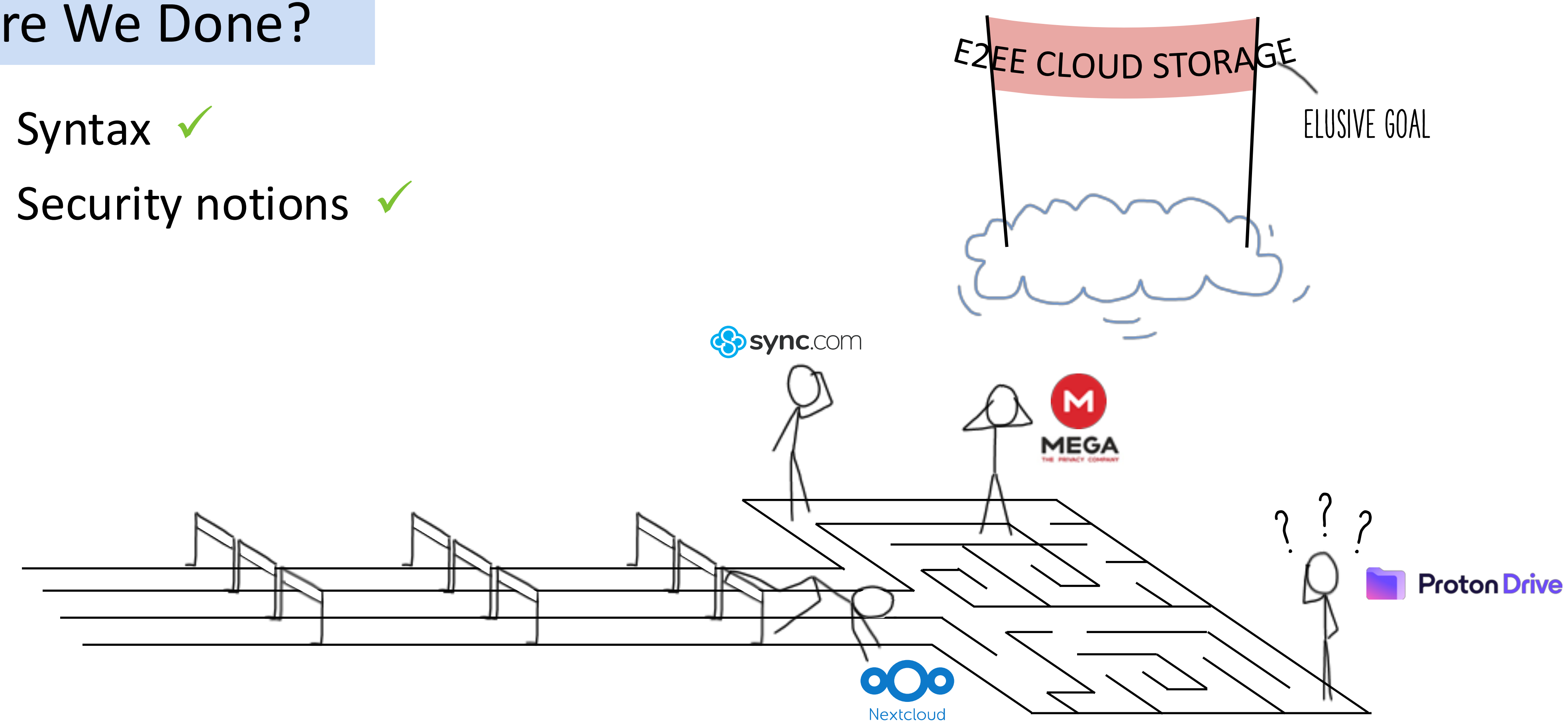
- Authentication & authorization
- No offline dictionary attacks on pw
- Availability for honest user files



? Are we missing any goals or attacks in both settings?

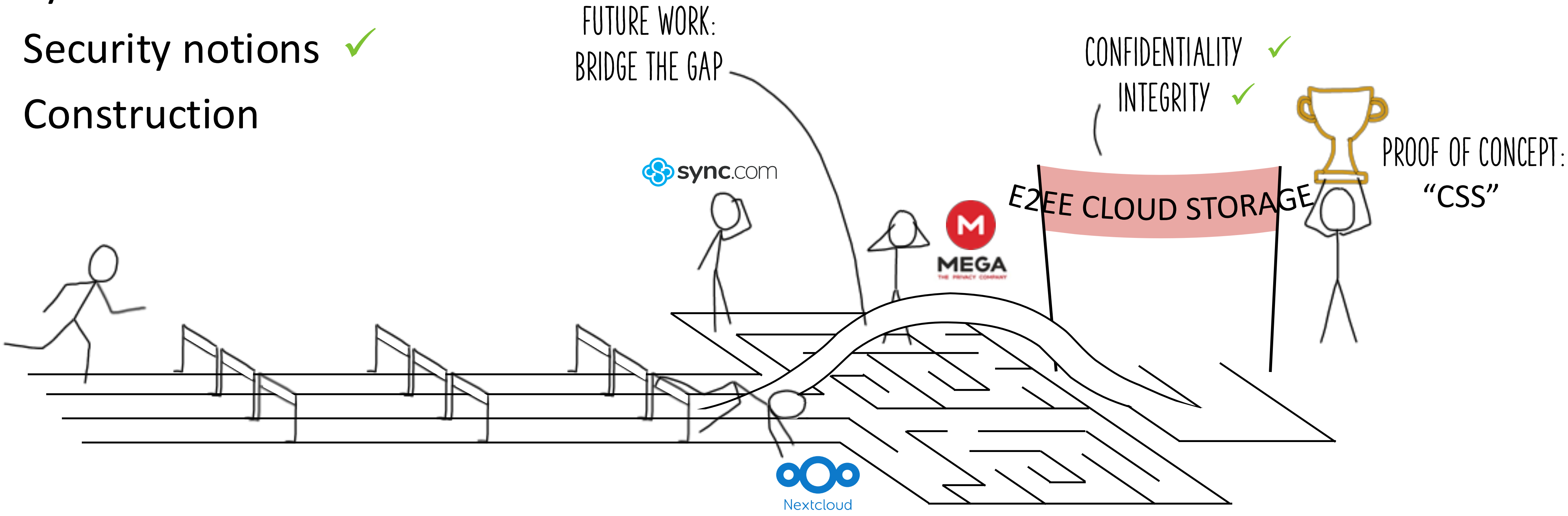
Are We Done?

- Syntax ✓
- Security notions ✓



Are We Done?

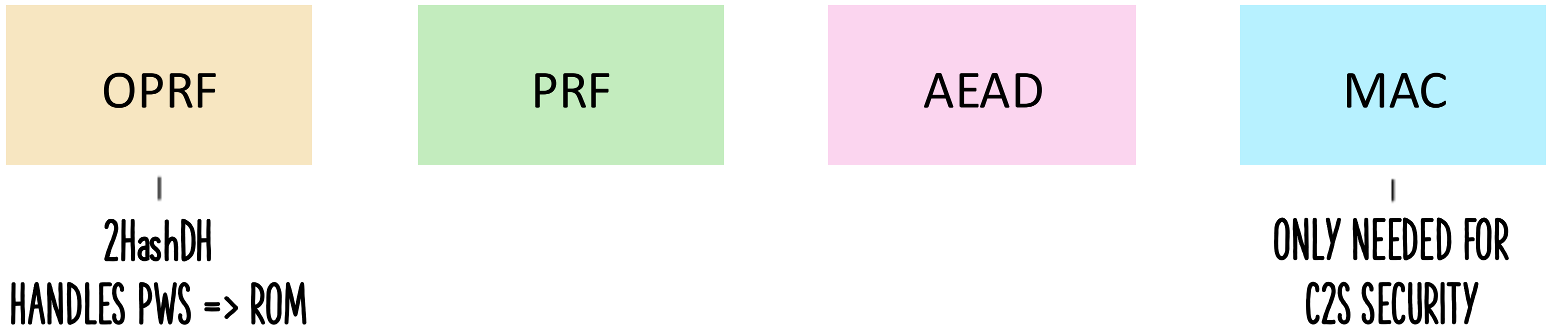
- Syntax ✓
- Security notions ✓
- Construction



2. Constructing E2EE Cloud Storage

CSS (Cloud Storage Scheme)

Building Blocks



OPRF

PRF

AEAD

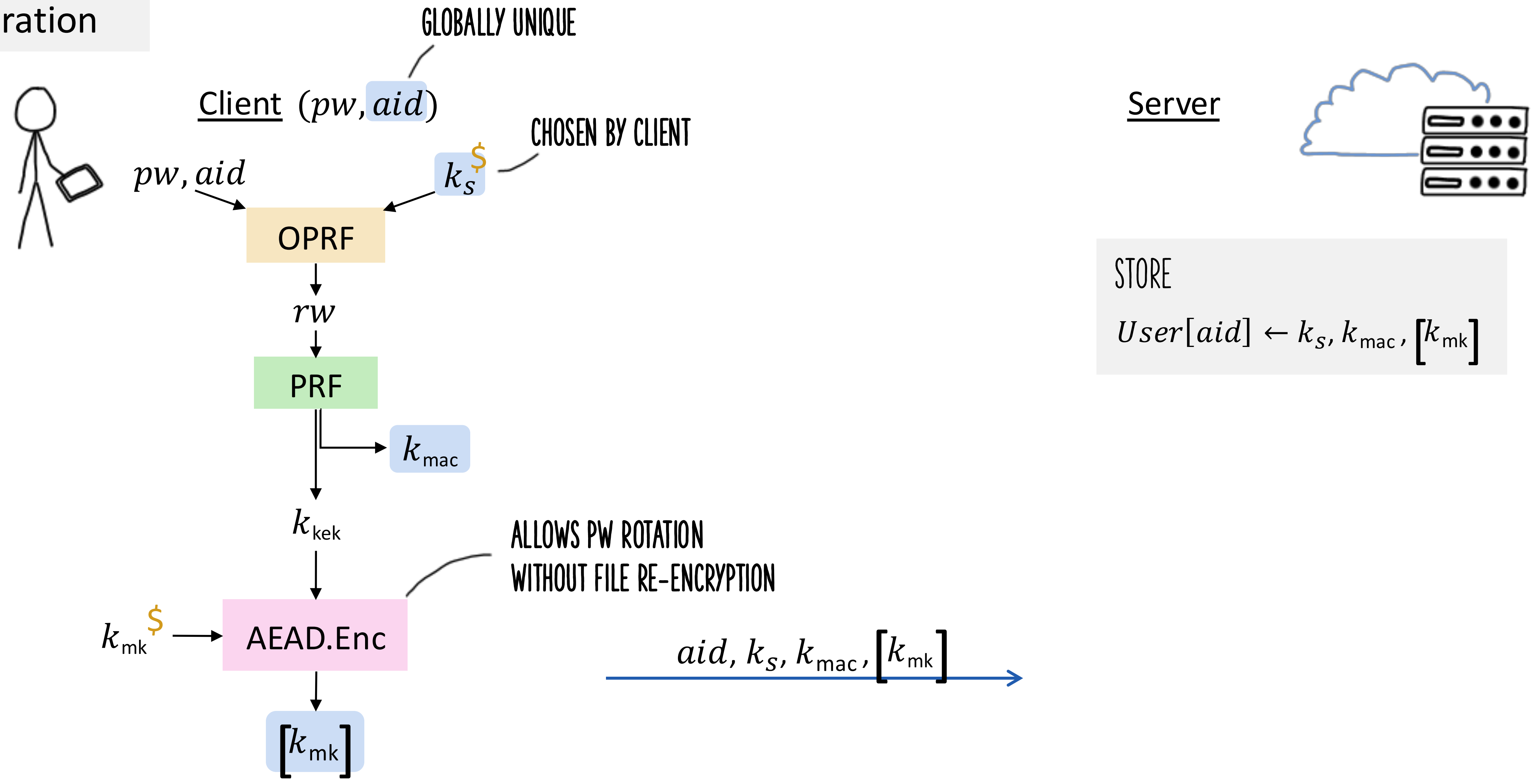
MAC

2HashDH
HANDLES PWS => ROM

ONLY NEEDED FOR
C2S SECURITY

CSS (Cloud Storage Scheme)

Registration



Client (pw, aid)

GLOBALLY UNIQUE

pw, aid

$k_s^{\$}$

CHOSEN BY CLIENT

OPRF

rw

PRF

k_{mac}

k_{kek}

ALLOWS PW ROTATION
WITHOUT FILE RE-ENCRYPTION

$k_{\text{mk}}^{\$}$

AEAD.Enc

$[k_{\text{mk}}]$

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

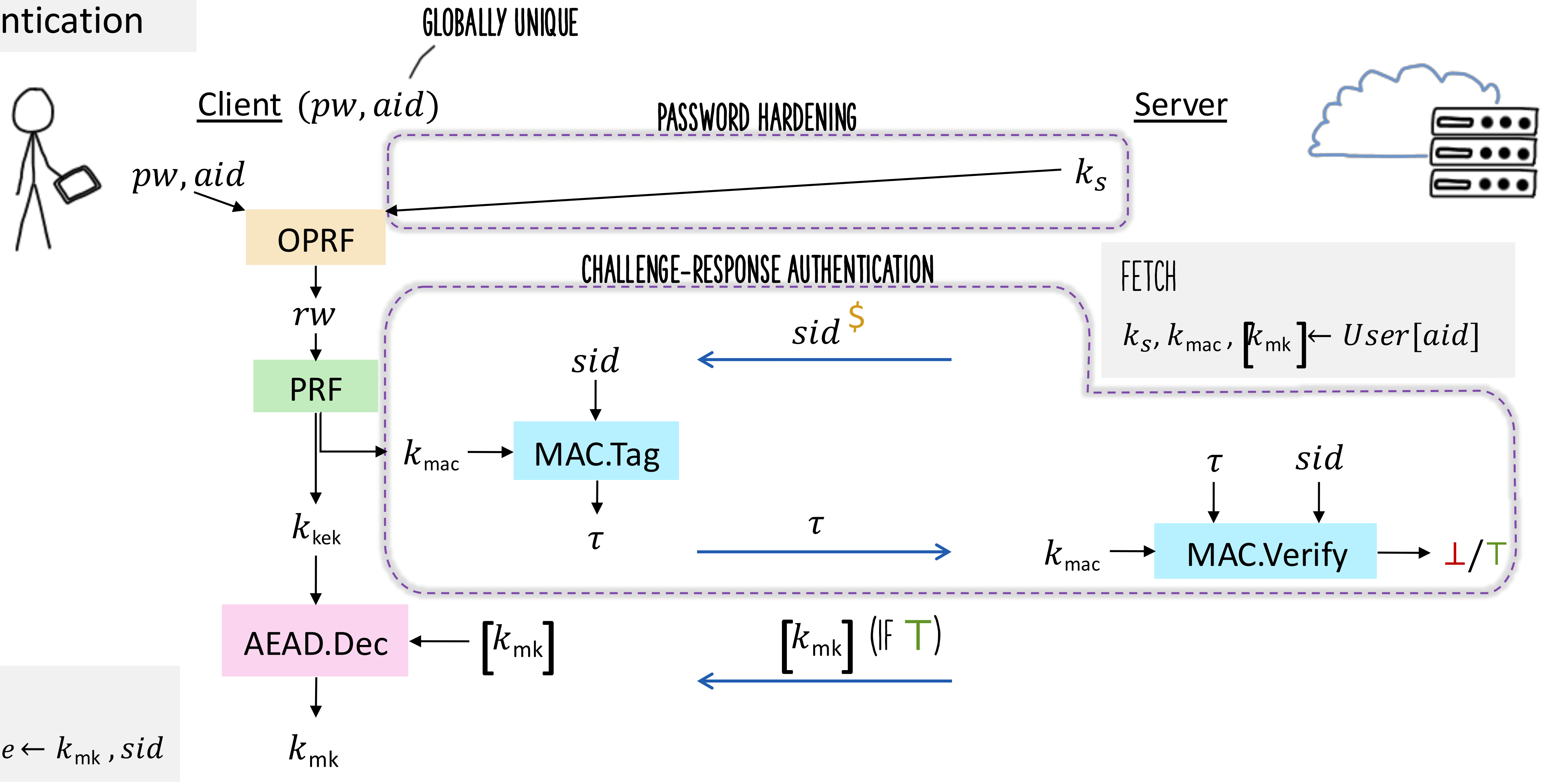
Server

STORE

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

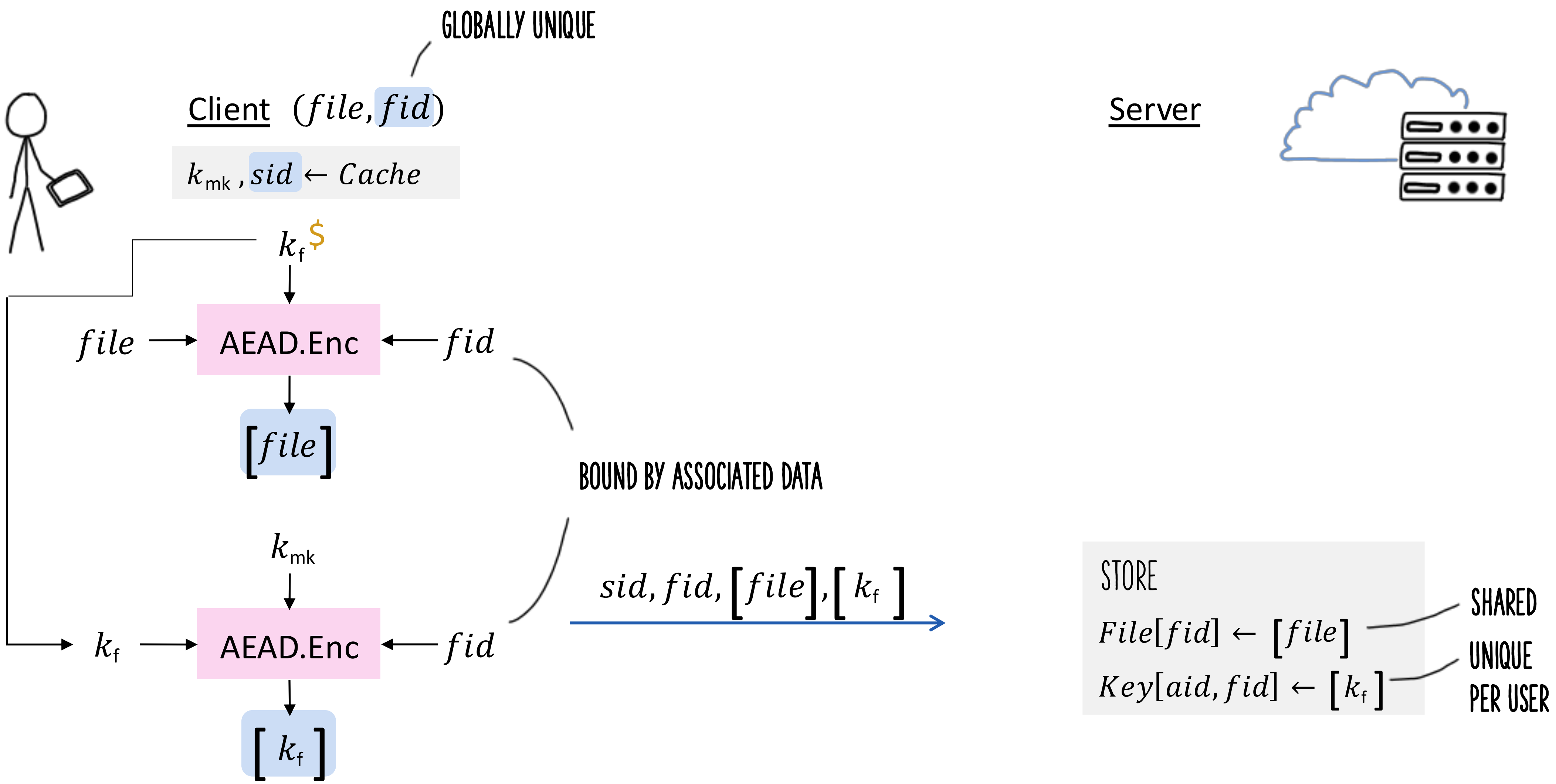
CSS (Cloud Storage Scheme)

Authentication



CSS (Cloud Storage Scheme)

Put



GLOBALLY UNIQUE

Client $(file, fid)$

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

$k_{\mathrm{f}}^{\$}$

$file \rightarrow$ AEAD.Enc $\leftarrow fid$

$[file]$

BOUND BY ASSOCIATED DATA

k_{mk}

$k_{\mathrm{f}} \rightarrow$ AEAD.Enc $\leftarrow fid$

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

$sid, fid, [file], [k_{\mathrm{f}}]$

Server

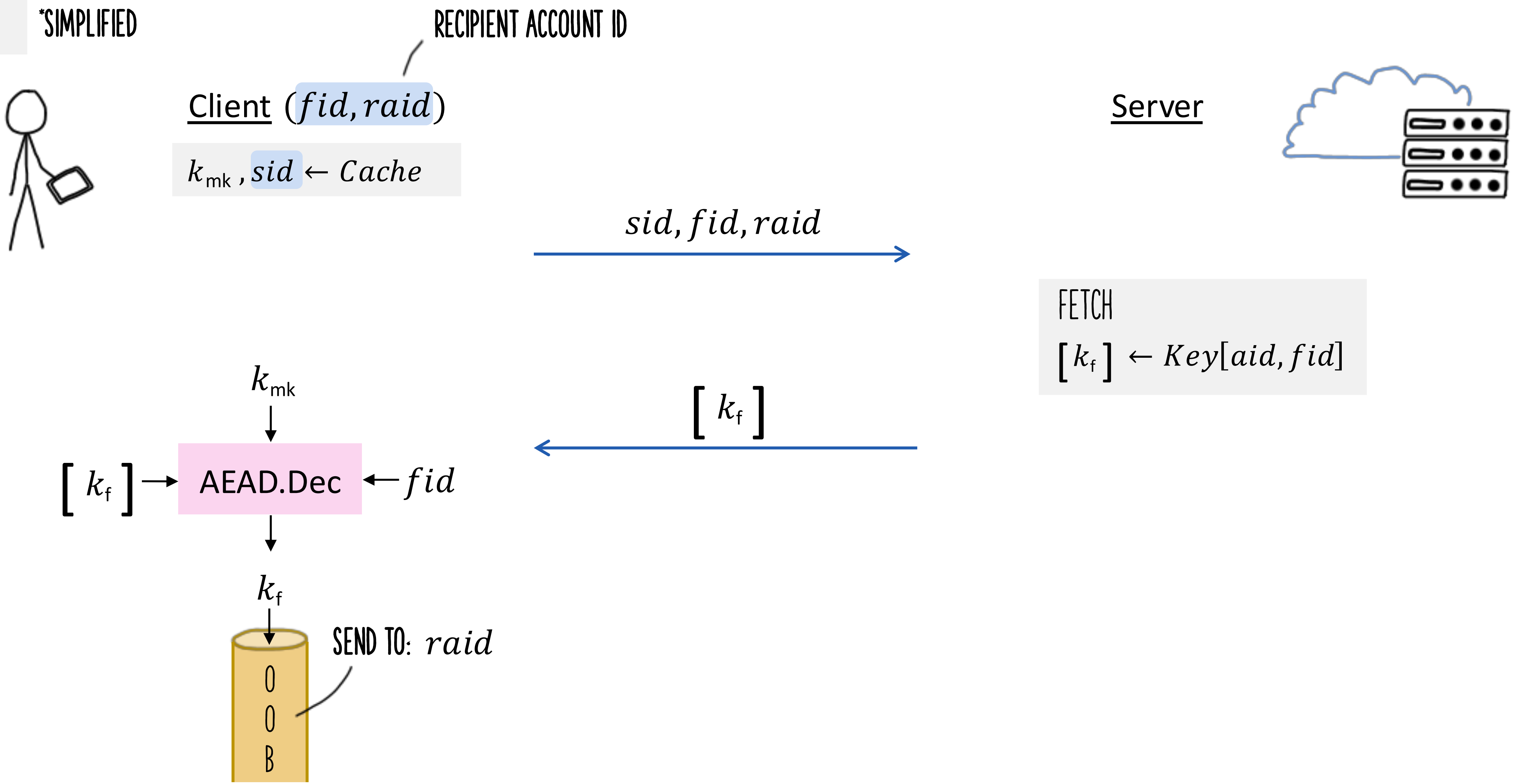
STORE

$File[fid] \leftarrow [file]$ SHARED

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

CSS (Cloud Storage Scheme)

Share



*SIMPLIFIED

RECIPIENT ACCOUNT ID

Client $(fid, raid)$

$k_{mk}, sid \leftarrow Cache$

$sid, fid, raid$

Server

FETCH

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

$[k_f]$

k_{mk}

$[k_f] \rightarrow$ AEAD.Dec $\leftarrow fid$

k_f

OOB

SEND TO: $raid$

CSS (Cloud Storage Scheme)

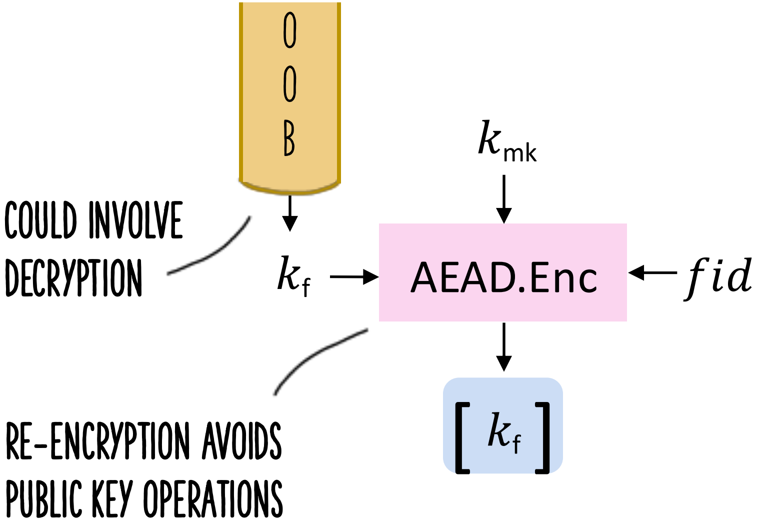
Accept

*SIMPLIFIED

Client (fid)

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

Server



COULD INVOLVE DECRYPTION

RE-ENCRYPTION AVOIDS PUBLIC KEY OPERATIONS

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

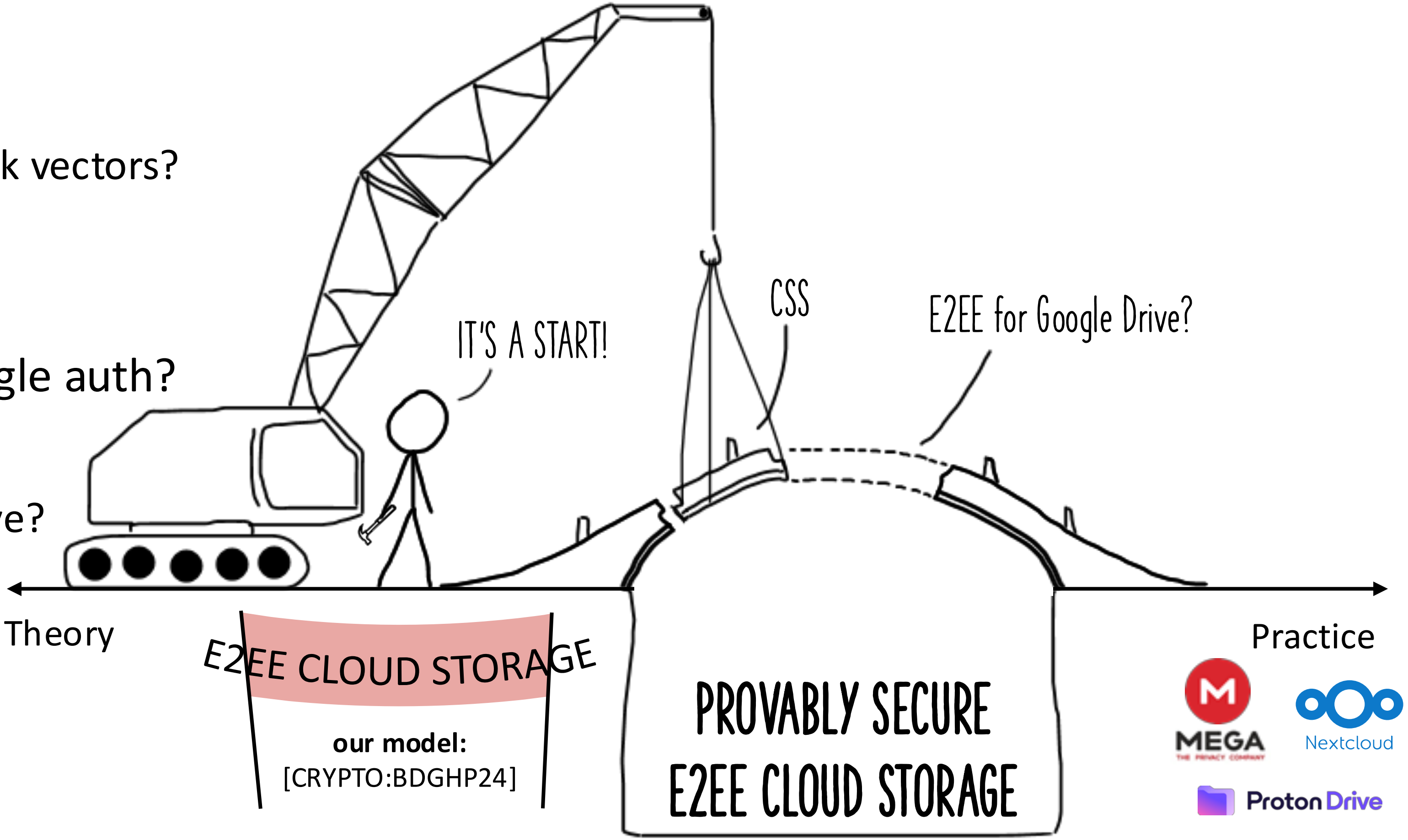
STORE

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

Discussing The Future of E2EE Cloud Storage

Your thoughts on:

- Our model:
  - Missing guarantees, or attack vectors?
- Our core functionality:
  - Missing features?
- Integrate reg + auth with Google auth?
- OOB channel for sharing:
  - Instantiation for Google Drive?
- Scalability of CSS?

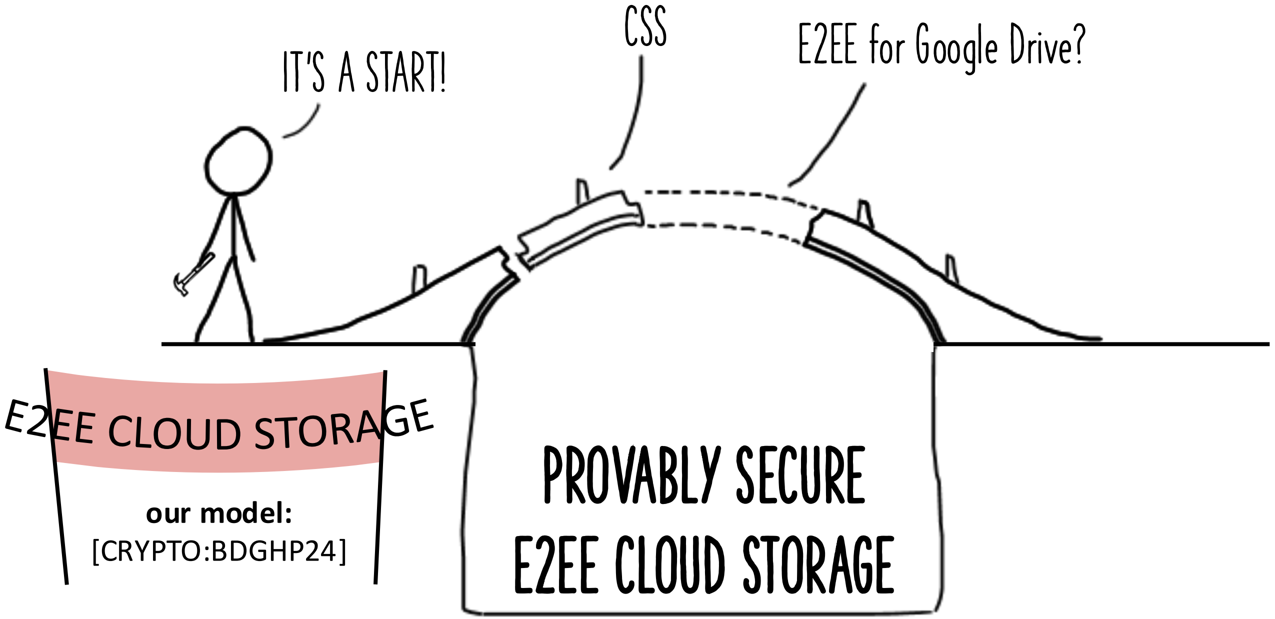


A Formal Treatment of
End-to-End Encrypted Cloud Storage

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

mbackendal@inf.ethz.ch mhaller@ucsd.edu

eprint.iacr.org/2024/989


IT'S A START!
CSS
E2EE for Google Drive?
E2EE CLOUD STORAGE
our model:
[CRYPTO:BDGHP24]
PROVABLY SECURE
E2EE CLOUD STORAGE