Analyzing Cryptography in Context:
The Case Study of Apple's CSAM Scanning Proposal

Ran Canetti (Boston University) Julie Ha (Boston University), and
Gabriel Kaptchuk (University of Maryland)

Cryptographic Applications Workshop 2024

“How *should* we analyze cryptographic *deployments*?”

The Apple PSI System

Abhishek Bhowmick
Apple Inc.

Dan Boneh
Stanford University

Steve Myers
Apple Inc.

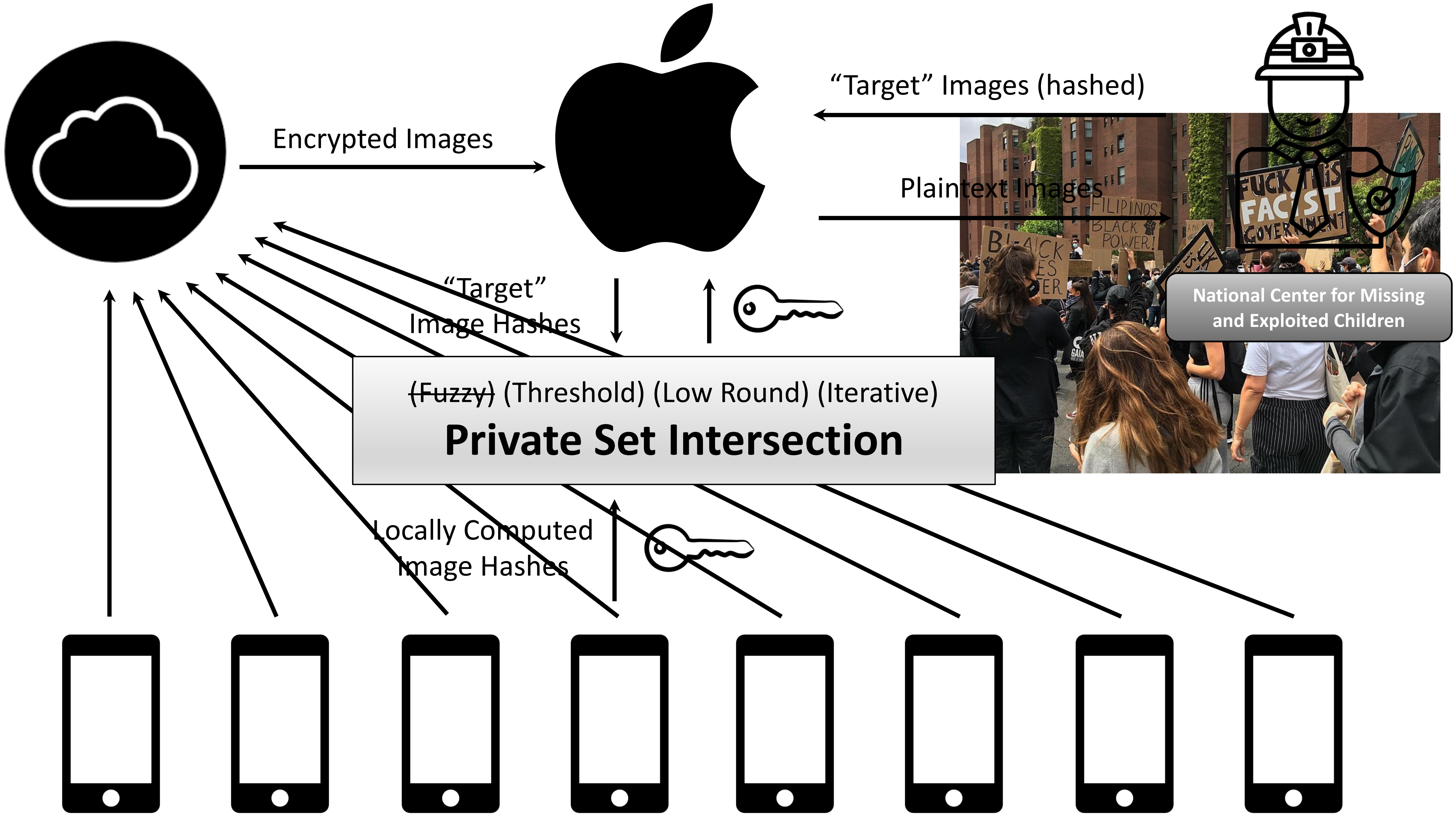
Kunal Talwar
Apple Inc.

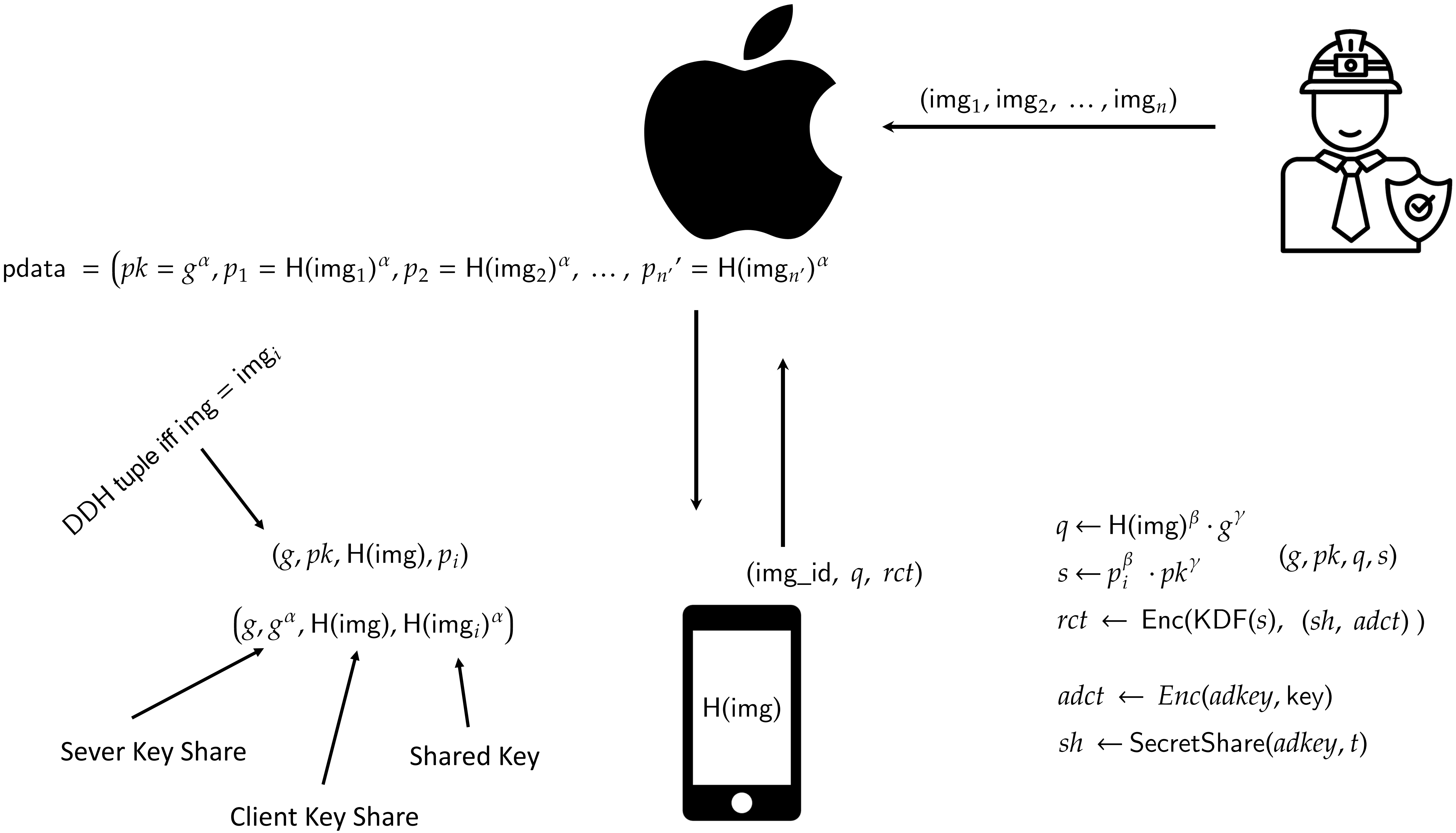
Karl Tarbe
Apple Inc.

July 29, 2021

Abstract

This document describes the constraints that drove the design of the Apple *private set intersection* (PSI) protocol. Apple PSI makes use of a variant of PSI we call *private set intersection with associated data* (PSI-AD), and an extension called *threshold private set intersection with associated data* (tPSI-AD). We describe a protocol that satisfies the constraints, and analyze its security. The context and motivation for the Apple PSI system are described on the main project site.


Encrypted Images
“Target” Images (hashed)
Plaintext Images
FILIPINOS
BLACK POWER!
“Target”
Image Hashes
National Center for Missing
and Exploited Children
~~(Fuzzy)~~ (Threshold) (Low Round) (Iterative)
Private Set Intersection
Locally Computed
Image Hashes



$(\mathrm{img}_1, \mathrm{img}_2, \ldots, \mathrm{img}_n)$
$\mathrm{pdata} = \left(pk = g^{\alpha}, p_1 = \mathrm{H}(\mathrm{img}_1)^{\alpha}, p_2 = \mathrm{H}(\mathrm{img}_2)^{\alpha}, \ldots, p_{n'}' = \mathrm{H}(\mathrm{img}_{n'})^{\alpha}\right)$
DDH tuple iff $\mathrm{img} = \mathrm{img}_i$
$(g, pk, \mathrm{H}(\mathrm{img}), p_i)$
$\left(g, g^{\alpha}, \mathrm{H}(\mathrm{img}), \mathrm{H}(\mathrm{img}_i)^{\alpha}\right)$
Sever Key Share
Client Key Share
Shared Key
$(\mathrm{img\_id},\ q,\ rct)$
H(img)
$q \leftarrow \mathrm{H}(\mathrm{img})^{\beta} \cdot g^{\gamma}$
$s \leftarrow p_i^{\beta} \cdot pk^{\gamma}$
(g, pk, q, s)
$rct \leftarrow \mathrm{Enc}(\mathrm{KDF}(s),\ (sh,\ adct))$
$adct \leftarrow Enc(adkey, \mathrm{key})$
$sh \leftarrow \mathrm{SecretShare}(adkey, t)$

Ma
@m

I've had
people t
CSAM s

1:59 AM · A

1,741 Retw

Ma
Re
Th
ma
ma

Ma
Init
clo
sur

Ma
Th
be
let

Eric Rescorla

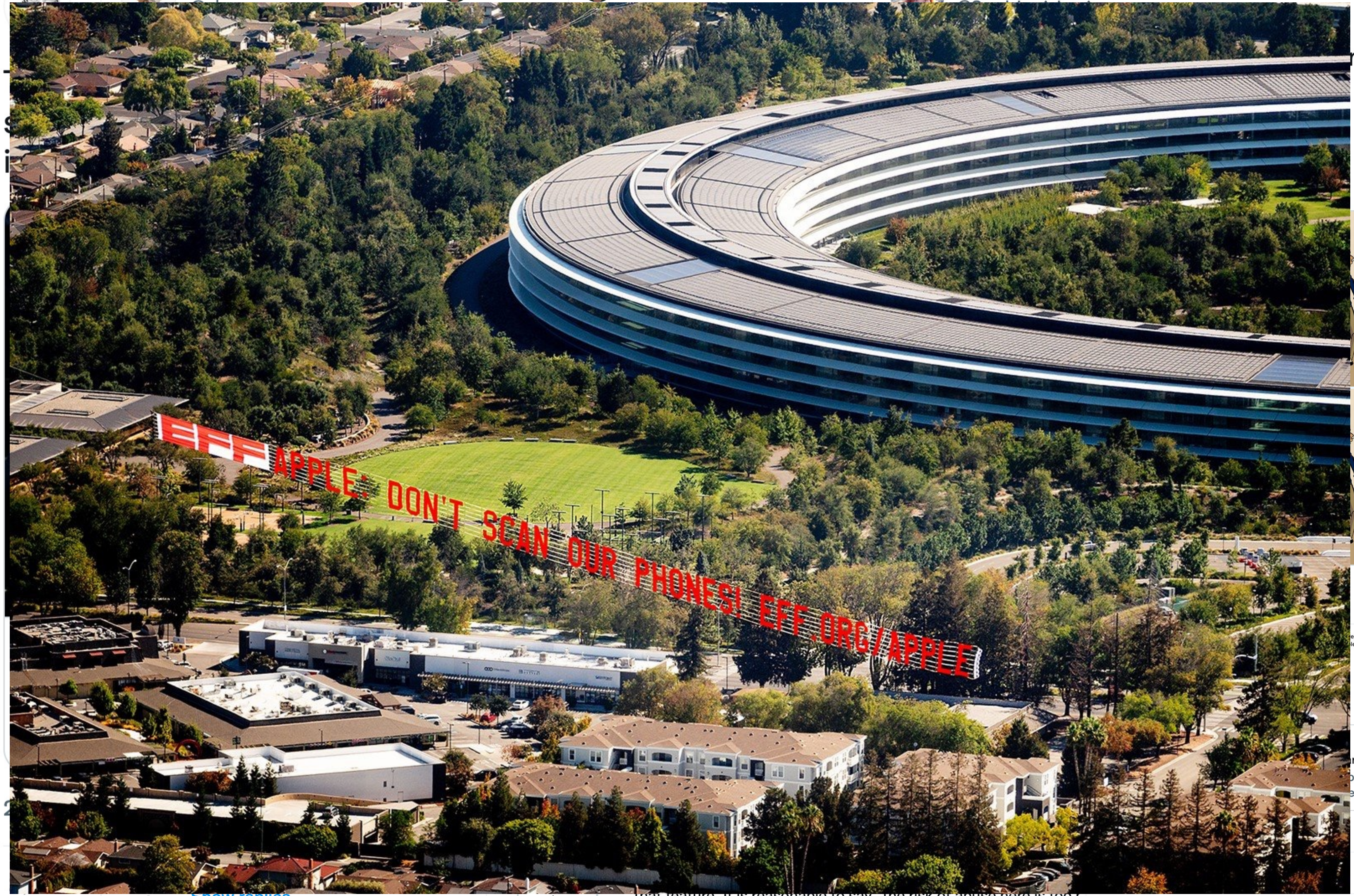
Matthew Green
@matthew_d_gree

Matthew Green
@matthew_d_green

Sarah Jami
@SarahJam

Sarah Jamie Lewis

nically,



If threshold of matches is exceeded, only then can Apple interpret voucher content for CSAM matches

Reviewed by Apple, report sent to NCMEC

round new Photo scanning featu...
of new features coming later this
en. The new features include ...

Show replies

2) E
sec

that feature. It is reasonable to say "the risk of abuse here is too l
use cloud-side scanning.

1

The system can be easily re-targeted

2

Malicious operators or hackers could abuse system

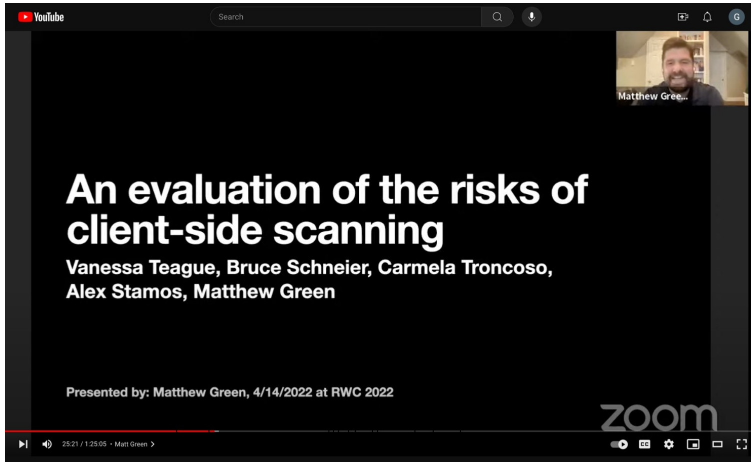
3

The system lacks robustness against malicious clients

Bugs in our Pockets:
The Risks of Client-Side Scanning

Hal Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Jon Callas, Whitfield Diffie, Susan Landau, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Vanessa Teague, Carmela Troncoso

October 15, 2021



“How *should* we analyze cryptographic *deployments*?”

“Well you should probably prove that your construction is secure…”

Parameters known to all parties:

- two parties: server and client,
- B is the maximum set size for the server and client,
- t is the threshold,
- $s_{\max}$ is the maximum size of the set S of synthetics,
- all associated data values ad in $\mathcal{D}$ have the same public length.

The functionality $\mathcal{F}$:

- Wait for input $X = \{x_1, x_2, \ldots\}$ from the server; abort if the server is corrupt and $|X| > B$.
- Send $|X|$ to the client; abort if the client is corrupt and aborts.
- Wait for input $\bar{Y} \in (\mathcal{U} \times \mathcal{ID} \times \mathcal{D})^m$ and $S \subseteq id(\bar{Y})$ from the client; abort if the client is corrupt and ($m > B$ or $|S| > s_{\max}$).
- Send $\bar{Y}_{id}$ to the server.
- If $|id(\bar{Y} \cap X) \smallsetminus S| > t$:
  send $\bar{Y}[id(\bar{Y} \cap X) \smallsetminus S]_{\{id,ad\}}$ and S to the server,
  otherwise:
  send $id(\bar{Y} \cap X) \cup S$ to the server.

Figure 1: The ideal functionality $\mathcal{F}$ for ftPSI-AD

Game $\mathbf{G}^{\mathrm{ss}}_{\Pi}$

INIT:

$c \leftarrow\$ \{0,1\}$ // Random challenge bit

H $\leftarrow\$$ Π.HS // Pick a function to be the random oracle

POST(X_0, X_1): // Adversary calls with sets X_0, X_1.

Require: $|X_0| = |X_1|$ and $X_0, X_1 \subseteq$ Π.U

$(pdata, skey) \leftarrow\$$ Π.SePost$^{\mathrm{HASH}}(X_c)$

Return $pdata$ // Adversary is given this

HASH(W): // The random oracle

Return H(W)

FIN(c'): // Adversary provides guess $c' \in \{0,1\}$ for c

Return $(c = c')$ // Result of game, true if $c' = c$ and false otherwise

Figure 2: Game defining server-security of protocol Π.

Downside: modularity is a double edged sword

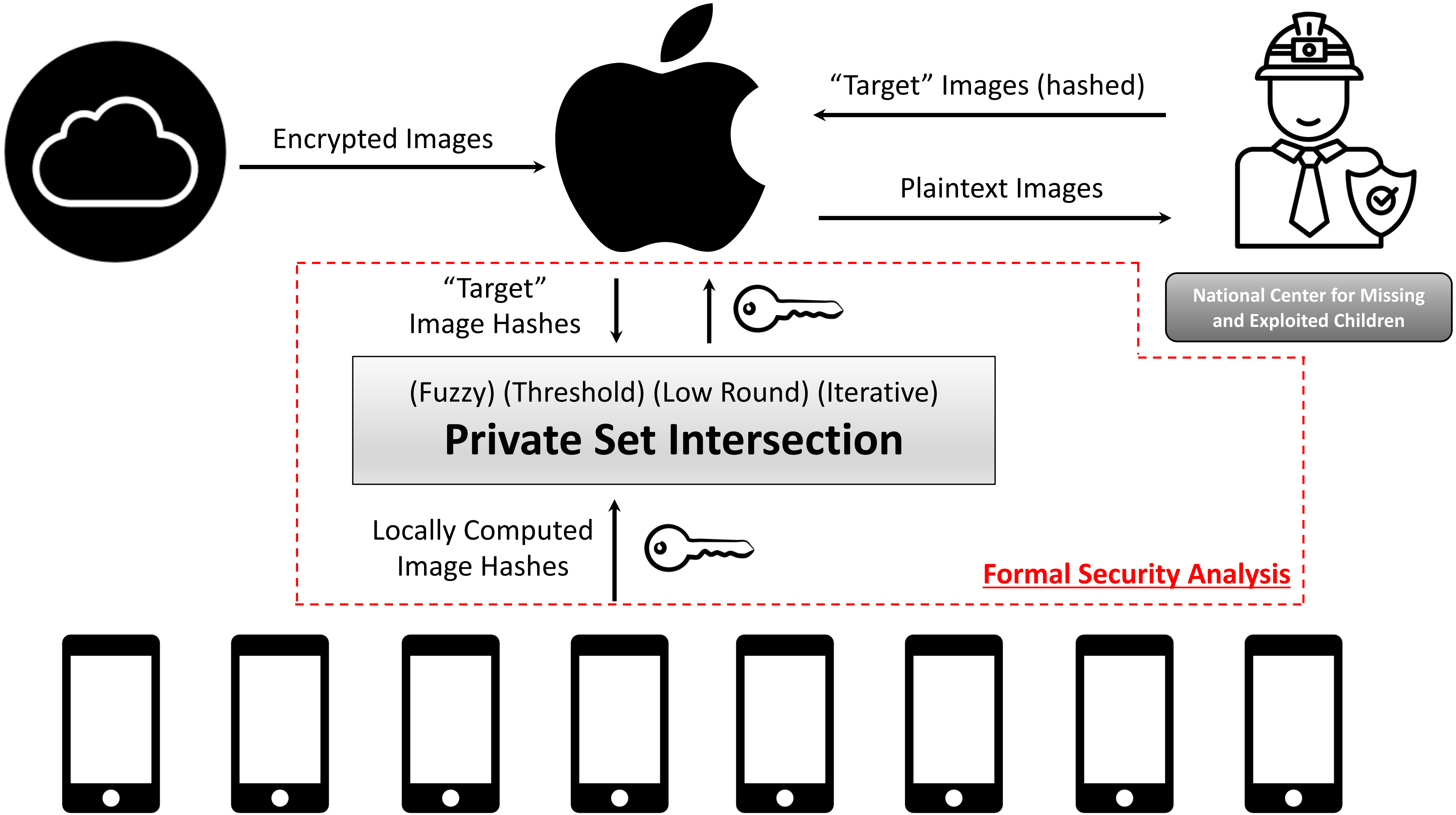
Downside: Game based proofs aren’t very expressive

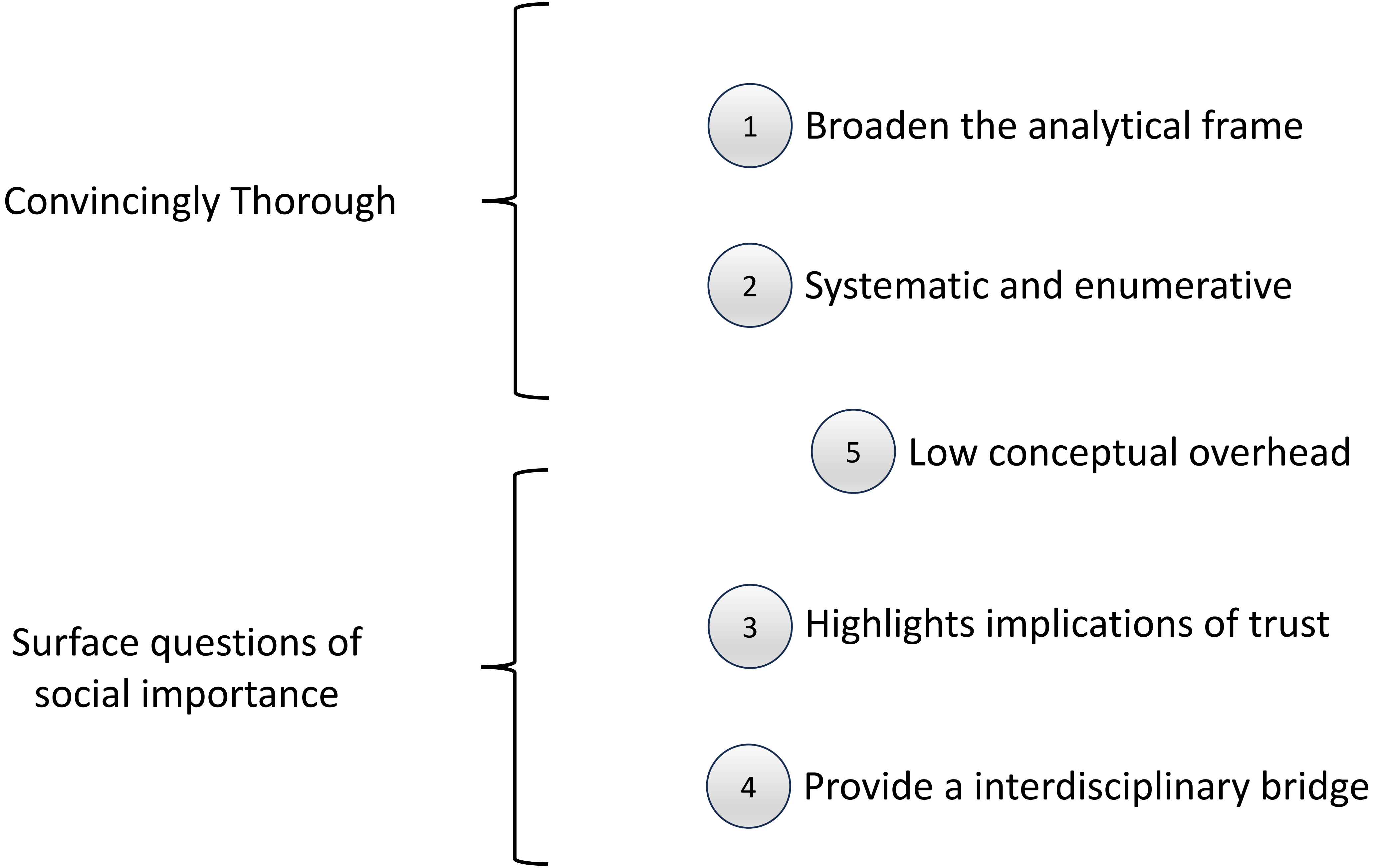
“How *should* we analyze cryptographic *deployments*?”

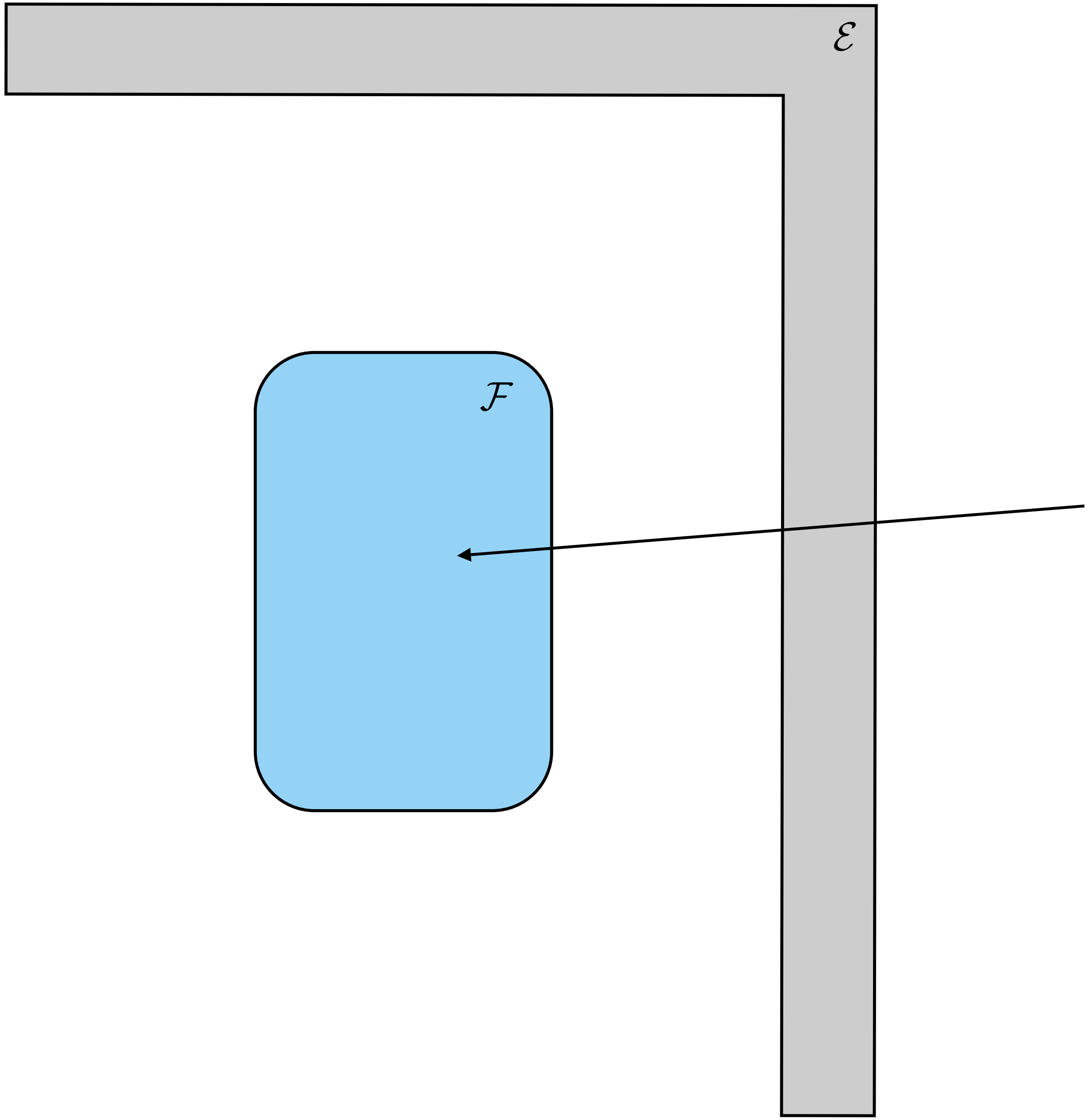
Our answer: “<u>In context</u>”

Option 1: Apple didn’t know about the issues, and their analysis failed to highlight the problems

Option 2: Apple knew about all these issues, and our standard cryptographic analysis tools make it easy to suppress them

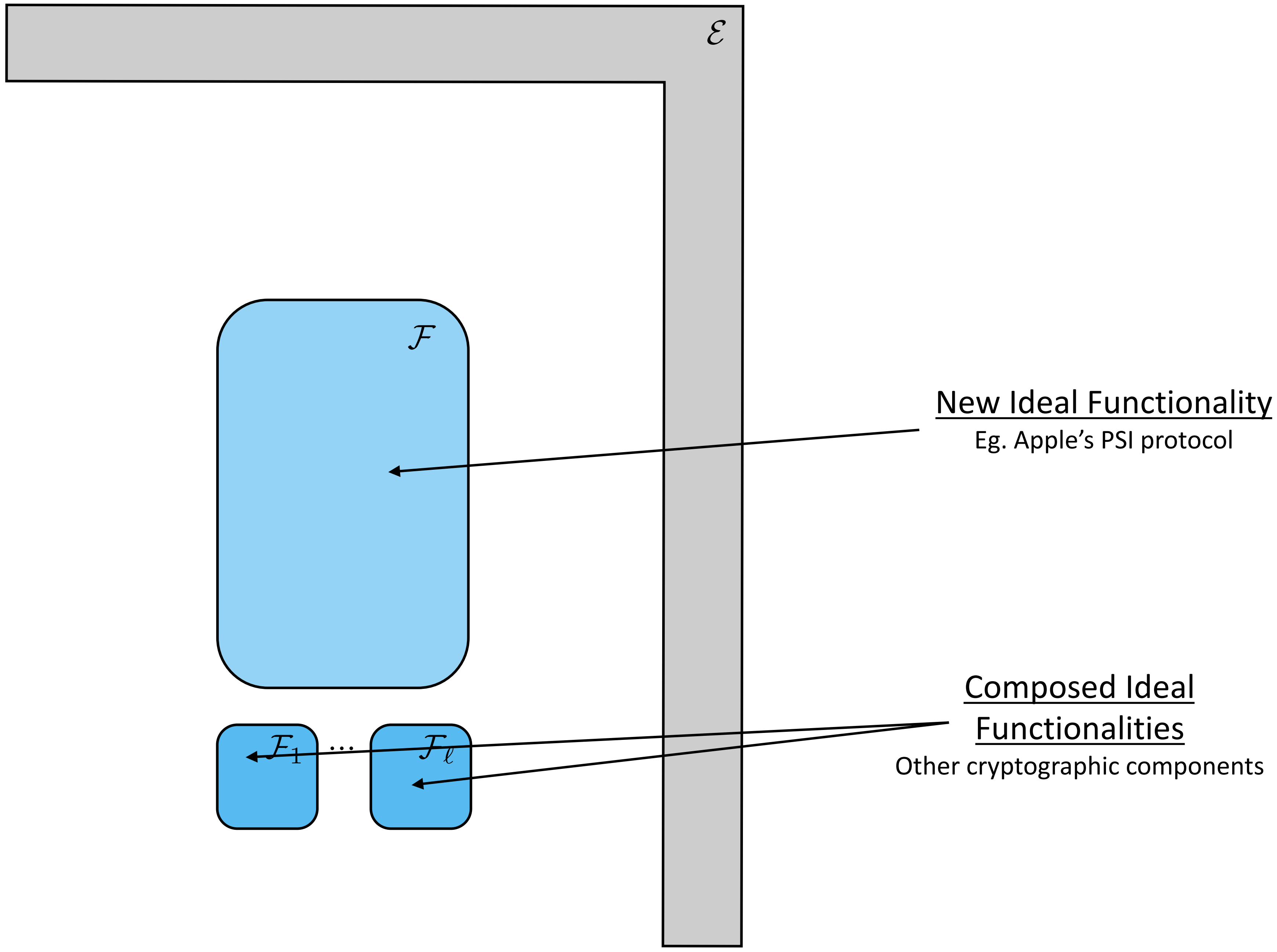

Encrypted Images
"Target" Images (hashed)
Plaintext Images
National Center for Missing and Exploited Children
"Target" Image Hashes
(Fuzzy) (Threshold) (Low Round) (Iterative)
Private Set Intersection
Locally Computed Image Hashes
Formal Security Analysis

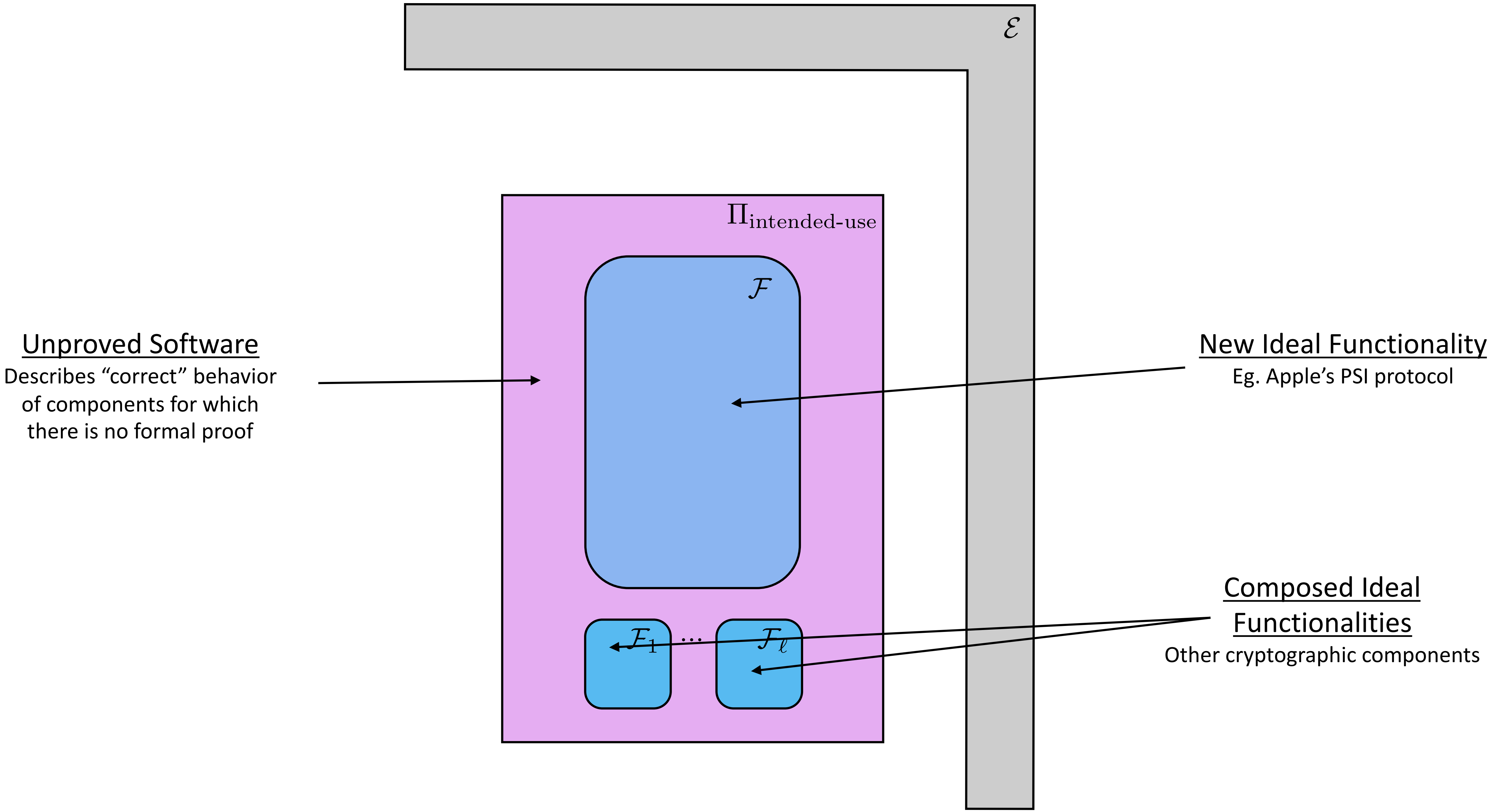

Convincingly Thorough
Surface questions of social importance
1 Broaden the analytical frame
2 Systematic and enumerative
5 Low conceptual overhead
3 Highlights implications of trust
4 Provide a interdisciplinary bridge



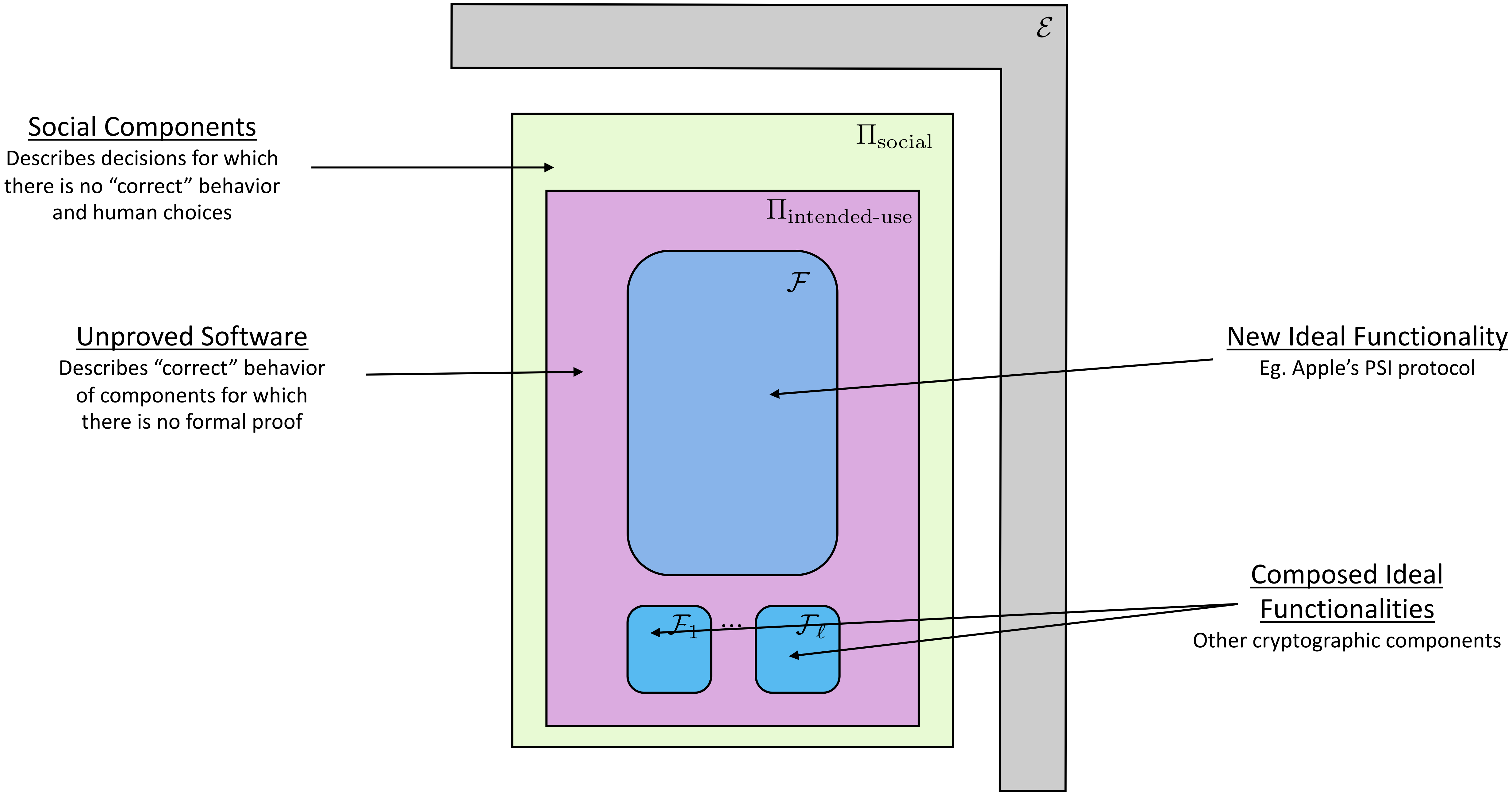
New Ideal Functionality

Eg. Apple's PSI protocol


$\mathcal{E}$
$\mathcal{F}$
$\mathcal{F}_1$
...
$\mathcal{F}_\ell$
New Ideal Functionality
Eg. Apple's PSI protocol
Composed Ideal Functionalities
Other cryptographic components



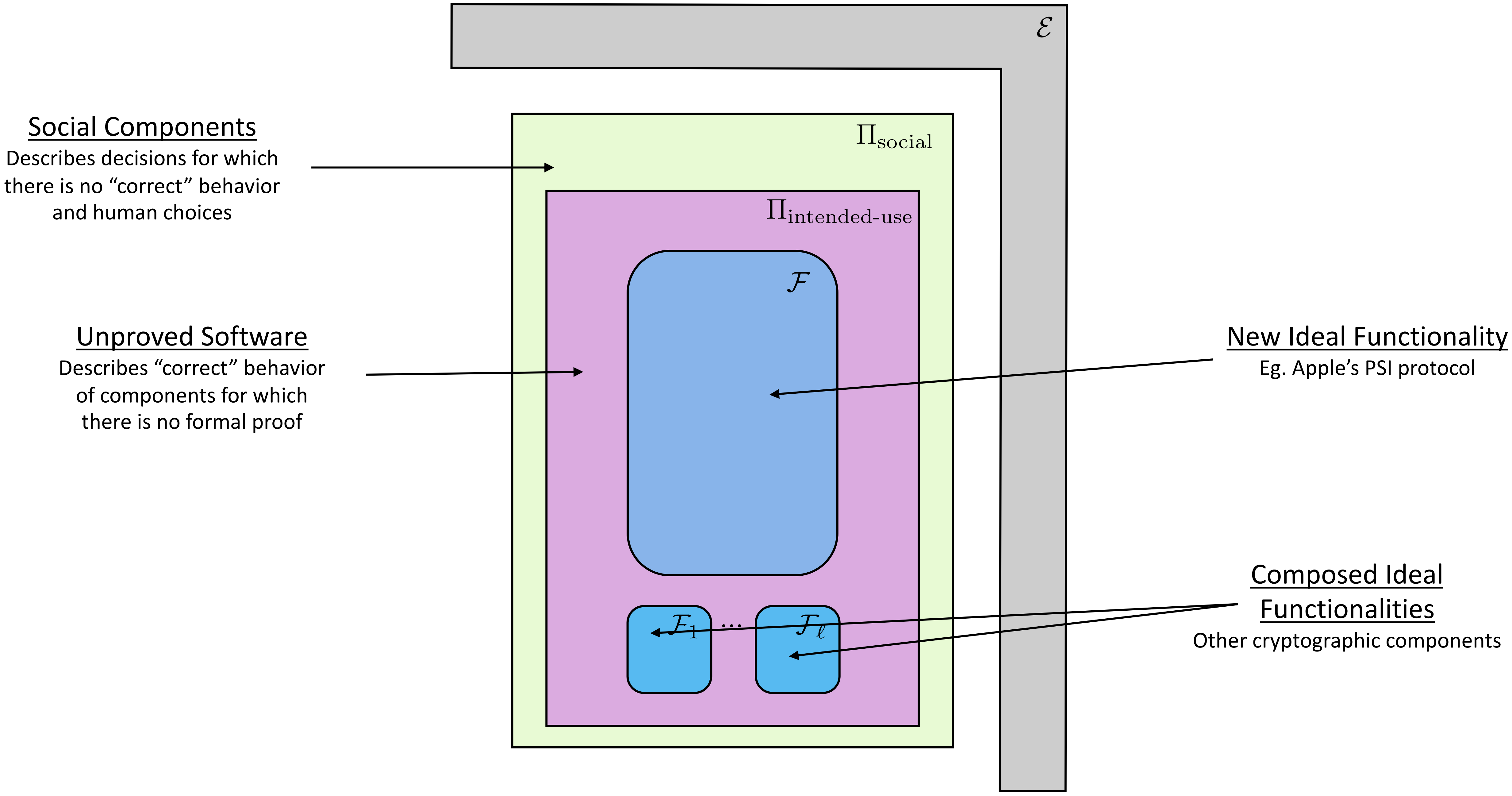
$\mathcal{E}$
$\Pi_{\text{intended-use}}$
$\mathcal{F}$
$\mathcal{F}_1$
...
$\mathcal{F}_\ell$
Unproved Software
Describes "correct" behavior of components for which there is no formal proof
New Ideal Functionality
Eg. Apple's PSI protocol
Composed Ideal Functionalities
Other cryptographic components



$\mathcal{E}$
Π_{social}
$\Pi_{\text{intended-use}}$
$\mathcal{F}$
$\mathcal{F}_1$ … $\mathcal{F}_\ell$
Social Components
Describes decisions for which there is no "correct" behavior and human choices
Unproved Software
Describes "correct" behavior of components for which there is no formal proof
New Ideal Functionality
Eg. Apple's PSI protocol
Composed Ideal Functionalities
Other cryptographic components

Did we meet our goals?

1. Broaden the analytical frame – Trace full lifecycle of the data
2. Systematic and enumerative – Fixed process for generating protocols
3. Highlights implications of trust – Trust choices are explicit
4. Provide a interdisciplinary bridge – Highlights social components in ”natural” language
5. Low conceptual overhead – *UC is super simple and easy to understand/work with*


$\mathcal{E}$
Π_{social}
$\Pi_{\text{intended-use}}$
$\mathcal{F}$
$\mathcal{F}_1$ ... $\mathcal{F}_\ell$
Social Components
Describes decisions for which there is no "correct" behavior and human choices
Unproved Software
Describes "correct" behavior of components for which there is no formal proof
New Ideal Functionality
Eg. Apple's PSI protocol
Composed Ideal Functionalities
Other cryptographic components

Case Study: Apple's Proposed* CSAM Scanning System

$\mathcal{F}_{\textbf{Apple-CSAM-Scan}}$

The functionality is parameterized by a threshold $\mathcal{T}$. For all clients $\mathcal{U}$, initialize $\mathsf{IsCounting}_{\mathcal{U}} = \mathsf{False}$.

Initializing Clients: Upon input $(\mathsf{InitScan}, \{\mathcal{U}_1, \ldots \mathcal{U}_n\}, \{X_{\mathcal{U}_1}, \ldots, X_{\mathcal{U}_n}\})$ from $APPLE$:

1. For $\mathcal{U}_i \in \{\mathcal{U}_1, \ldots \mathcal{U}_n\}$, if $\mathsf{IsInited}_{\mathcal{U}_i} = \mathsf{True}$, continue. Otherwise record $X_{\mathcal{U}_i}$ and set $\mathsf{IsInited}_{\mathcal{U}_i} = \mathsf{True}$. Finally, send $(\mathsf{InitScanComplete})$ to $APPLE$.

Process Initialization: Upon input $(\mathsf{ProcessInit})$ from $\mathcal{U}$:

1. If $\mathsf{IsInited}_{\mathcal{U}} \neq \mathsf{True}$, set $\mathsf{IsCounting}_{\mathcal{U}} = \mathsf{True}$ and send $(\mathsf{InitScanComplete}, |X_{\mathcal{U}}|)$ to $\mathcal{U}$. Otherwise, send $(\mathsf{InitScanComplete}, \perp)$ to $\mathcal{U}$.

Scan Image: Upon input $(\mathsf{ScanImage}, \mathsf{img\_id}, \mathsf{img\_hash}, \mathsf{k}, \mathsf{valid} \in \{\mathsf{true}, \mathsf{false}\})$ from client $\mathcal{U}$:

1. $\mathsf{IsCounting}_{\mathcal{U}} = \mathsf{True}$, return. If this is the first image received from $\mathcal{U}$, initialize variable $\mathsf{counter}_{\mathcal{U}}$, list $\mathsf{photolist}_{\mathcal{U}}$, and list $\mathsf{keys}_{\mathcal{U}}$.

   //$\mathcal{F}$ alerts the Server that the user has uploaded a photo.

2. If $\mathsf{img\_hash} \notin X_{\mathcal{U}}$ or $\mathsf{valid} = \mathsf{false}$, send $(\mathsf{ScanImageComplete}, \mathcal{U}, \mathsf{false})$ to $APPLE$.
3. If $\mathsf{img\_hash} \in X_{\mathcal{U}}$, increment $\mathsf{counter}_{\mathcal{U}}$, append $\mathsf{img\_id}$ to $\mathsf{photolist}_{\mathcal{U}}$ and append k to $\mathsf{keys}_{\mathcal{U}}$. Then,
   (a) If $\mathsf{counter}_{\mathcal{U}} < \mathcal{T}$ send $(\mathsf{ScanImageComplete}, \mathcal{U}, \mathsf{true})$ to $APPLE$.
   (b) If $\mathsf{counter}_{\mathcal{U}} \geq \mathcal{T}$, send $(\mathsf{ThresholdMet}, \mathcal{U}, \mathsf{photolist}_{\mathcal{U}}, \mathsf{keys}_{\mathcal{U}})$ to $APPLE$.

One Shot vs. Incremental

Parameters known to all parties:

- two parties: server and client,
- B is the maximum set size for the server and client,
- t is the threshold,
- $s_{\max}$ is the maximum size of the set S of synthetics,
- all associated data values ad in $\mathcal{D}$ have the same public length.

The functionality $\mathcal{F}$:

- Wait for input $X = \{x_1, x_2, \ldots\}$ from the server; abort if the server is corrupt and $|X| > B$.
- Send $|X|$ to the client; abort if the client is corrupt and aborts.
- Wait for input $\bar{Y} \in (\mathcal{U} \times \mathcal{ID} \times \mathcal{D})^m$ and $S \subseteq id(\bar{Y})$ from the client; abort if the client is corrupt and ($m > B$ or $|S| > s_{\max}$).
- Send $\bar{Y}_{id}$ to the server.
- If $|id(\bar{Y} \cap X) \setminus S| > t$:
  send $\bar{Y}[id(\bar{Y} \cap X) \setminus S]_{\{id,ad\}}$ and S to the server,
  otherwise:
  send $id(\bar{Y} \cap X) \cup S$ to the server.

Figure 1: The ideal functionality $\mathcal{F}$ for ftPSI-AD

$\mathcal{F}_{\textbf{Apple-CSAM-Scan}}$

The functionality is parameterized by a threshold $\mathcal{T}$. For all clients $\mathcal{U}$, initialize $\mathsf{IsCounting}_{\mathcal{U}} = \mathsf{False}$.

Initializing Clients: Upon input $(\mathsf{InitScan}, \{\mathcal{U}_1, \ldots \mathcal{U}_n\}, \{X_{\mathcal{U}_1}, \ldots, X_{\mathcal{U}_n}\})$ from $APPLE$:

1. For $\mathcal{U}_i \in \{\mathcal{U}_1, \ldots \mathcal{U}_n\}$, if $\mathsf{IsInited}_{\mathcal{U}_i} = \mathsf{True}$, continue. Otherwise record $X_{\mathcal{U}_i}$ and set $\mathsf{IsInited}_{\mathcal{U}_i} = \mathsf{True}$. Finally, send $(\mathsf{InitScanComplete})$ to $APPLE$.

Process Initialization: Upon input $(\mathsf{ProcessInit})$ from $\mathcal{U}$:

1. If $\mathsf{IsInited}_{\mathcal{U}} \neq \mathsf{True}$, set $\mathsf{IsCounting}_{\mathcal{U}} = \mathsf{True}$ and send $(\mathsf{InitScanComplete}, |X_{\mathcal{U}}|)$ to $\mathcal{U}$. Otherwise, send $(\mathsf{InitScanComplete}, \perp)$ to $\mathcal{U}$.

Scan Image: Upon input $(\mathsf{ScanImage}, \mathsf{img\_id}, \mathsf{img\_hash}, \mathsf{k}, \mathsf{valid} \in \{\mathsf{true}, \mathsf{false}\})$ from client $\mathcal{U}$:

1. $\mathsf{IsCounting}_{\mathcal{U}} = \mathsf{True}$, return. If this is the first image received from $\mathcal{U}$, initialize variable $\mathsf{counter}_{\mathcal{U}}$, list $\mathsf{photolist}_{\mathcal{U}}$, and list $\mathsf{keys}_{\mathcal{U}}$.

   //$\mathcal{F}$ alerts the Server that the user has uploaded a photo.
2. If $\mathsf{img\_hash} \notin X_{\mathcal{U}}$ or $\mathsf{valid} = \mathsf{false}$, send $(\mathsf{ScanImageComplete}, \mathcal{U}, \mathsf{false})$ to $APPLE$.
3. If $\mathsf{img\_hash} \in X_{\mathcal{U}}$, increment $\mathsf{counter}_{\mathcal{U}}$, append $\mathsf{img\_id}$ to $\mathsf{photolist}_{\mathcal{U}}$ and append k to $\mathsf{keys}_{\mathcal{U}}$. Then,

   (a) If $\mathsf{counter}_{\mathcal{U}} < \mathcal{T}$ send $(\mathsf{ScanImageComplete}, \mathcal{U}, \mathsf{true})$ to $APPLE$.

   (b) If $\mathsf{counter}_{\mathcal{U}} \geq \mathcal{T}$, send $(\mathsf{ThresholdMet}, \mathcal{U}, \mathsf{photolist}_{\mathcal{U}}, \mathsf{keys}_{\mathcal{U}})$ to $APPLE$.

Self Composition

Parameters known to all parties:

- two parties: server and client,
- B is the maximum set size for the server and client,
- t is the threshold,
- $s_{\max}$ is the maximum size of the set S of synthetics,
- all associated data values ad in $\mathcal{D}$ have the same public length.

The functionality $\mathcal{F}$:

- Wait for input $X = \{x_1, x_2, \ldots\}$ from the server;
  abort if the server is corrupt and $|X| > B$.
- Send $|X|$ to the client; abort if the client is corrupt and aborts.
- Wait for input $\bar{Y} \in (\mathcal{U} \times \mathcal{ID} \times \mathcal{D})^m$ and $S \subseteq id(\bar{Y})$ from the client;
  abort if the client is corrupt and $(m > B$ or $|S| > s_{\max})$.
- Send $\bar{Y}_{id}$ to the server.
- If $|id(\bar{Y} \cap X) \setminus S| > t$:
  send $\bar{Y}[id(\bar{Y} \cap X) \setminus S]_{\{id,ad\}}$ and S to the server,
  otherwise:
  send $id(\bar{Y} \cap X) \cup S$ to the server.

Figure 1: The ideal functionality $\mathcal{F}$ for ftPSI-AD

Extractability

4.4.1 Privacy for the server's dataset X against a malicious client

First, let's show that a malicious client $\mathcal{A}_c^H$ that interacts with an honest server with input X, learns nothing about X other than its size.

For an adversary $\mathcal{A}_c^H$, a simulator Sim_c, and an environment $\mathcal{Z}$, define the following two random variables:

- $\mathrm{REAL}_{\Pi,\mathcal{A}_c^H,\mathcal{Z}}$ is the random variable defined as the output of $\mathcal{Z}$ in a real world execution after interacting with the malicious client $\mathcal{A}_c^H$ as in Figure 4a.
- $\mathrm{IDEAL}_{\mathcal{F},\mathrm{Sim}_c,\mathcal{Z}}$ is the random variable defined as the output of $\mathcal{Z}$ in an ideal world execution after interacting with the simulator Sim_c as in Figure 4b.

Definition 2. We say that a protocol Π for $\mathcal{F}$ is **server private** if for every efficient adversary $\mathcal{A}_c$, there exists an efficient adversary Sim_c, such that for every efficient environment $\mathcal{Z}$,

$$\left|\Pr\left[\mathrm{REAL}_{\Pi,\mathcal{A}_c^H,\mathcal{Z}} = 1\right] - \Pr\left[\mathrm{IDEAL}_{\mathcal{F},\mathrm{Sim}_c,\mathcal{Z}} = 1\right]\right| \leq \epsilon$$

for some negligible ϵ.

4. Sim_c sends $\mathsf{pdata} \leftarrow (L, P_1, \ldots, P_{n'})$ and the nonces to $\mathcal{A}_c^H$.

This completes the description of Sim_c.

Case Study: Apple's Proposed* CSAM Scanning System

$\mathcal{F}_{\textbf{Apple-CSAM-Scan}}$

The functionality is parameterized by a threshold $\mathcal{T}$. For all clients $\mathcal{U}$, initialize $\mathsf{IsCounting}_{\mathcal{U}} = \mathsf{False}$.

Initializing Clients: Upon input $(\mathsf{InitScan}, \{\mathcal{U}_1, \ldots \mathcal{U}_n\}, \{X_{\mathcal{U}_1}, \ldots, X_{\mathcal{U}_n}\})$ from $APPLE$:

1. For $\mathcal{U}_i \in \{\mathcal{U}_1, \ldots \mathcal{U}_n\}$, if $\mathsf{IsInited}_{\mathcal{U}_i} = \mathsf{True}$, continue. Otherwise record $X_{\mathcal{U}_i}$ and set $\mathsf{IsInited}_{\mathcal{U}_i} = \mathsf{True}$. Finally, send $(\mathsf{InitScanComplete})$ to $APPLE$.

Process Initialization: Upon input $(\mathsf{ProcessInit})$ from $\mathcal{U}$:

1. If $\mathsf{IsInited}_{\mathcal{U}} \neq \mathsf{True}$, set $\mathsf{IsCounting}_{\mathcal{U}} = \mathsf{True}$ and send $(\mathsf{InitScanComplete}, |X_{\mathcal{U}}|)$ to $\mathcal{U}$. Otherwise, send $(\mathsf{InitScanComplete}, \perp)$ to $\mathcal{U}$.

Scan Image: Upon input $(\mathsf{ScanImage}, \mathsf{img\_id}, \mathsf{img\_hash}, \mathsf{k}, \mathsf{valid} \in \{\mathsf{true}, \mathsf{false}\})$ from client $\mathcal{U}$:

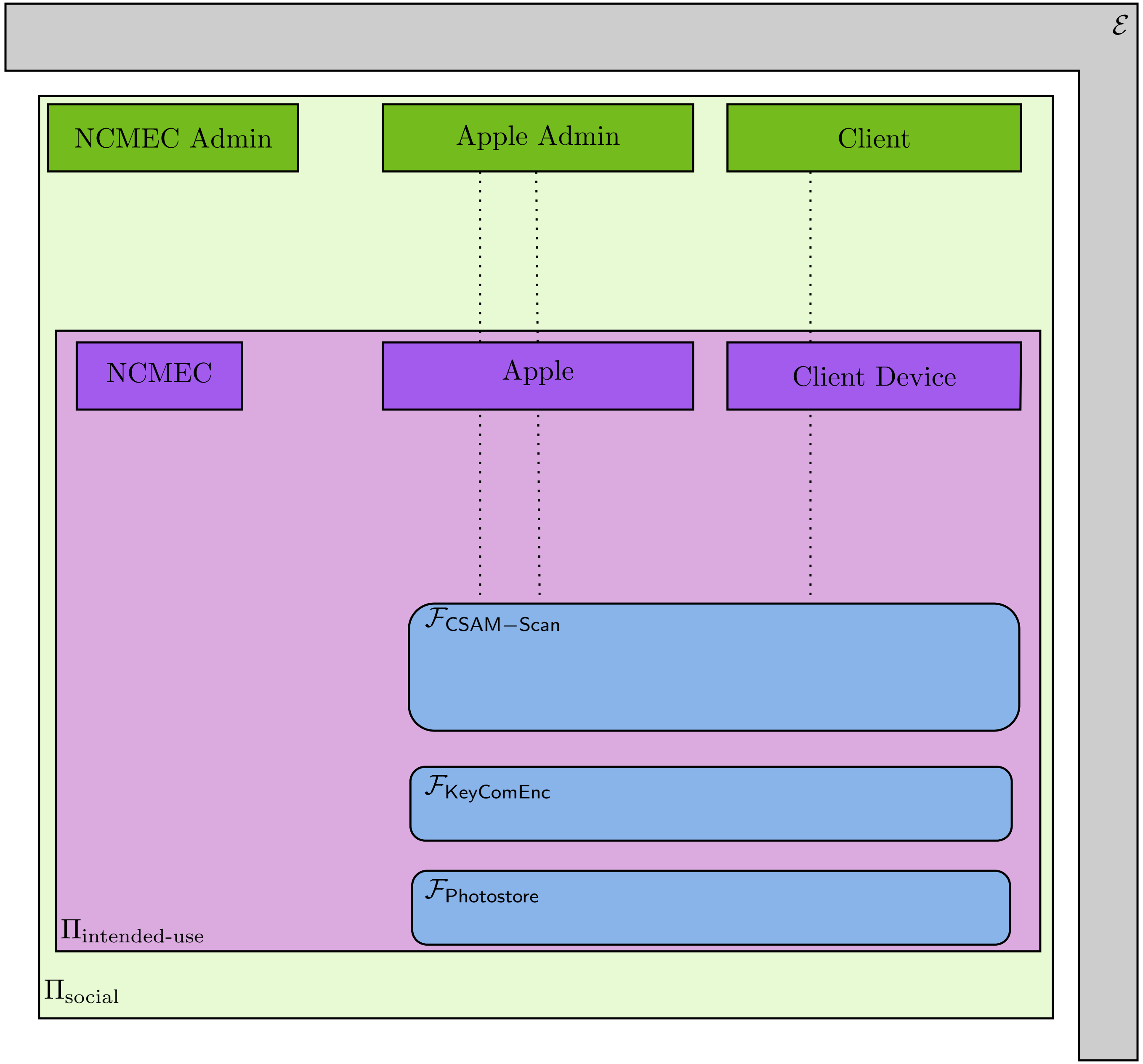
1. $\mathsf{IsCounting}_{\mathcal{U}} = \mathsf{True}$, return. If this is the first image received from $\mathcal{U}$, initialize variable $\mathsf{counter}_{\mathcal{U}}$, list $\mathsf{photolist}_{\mathcal{U}}$, and list $\mathsf{keys}_{\mathcal{U}}$.

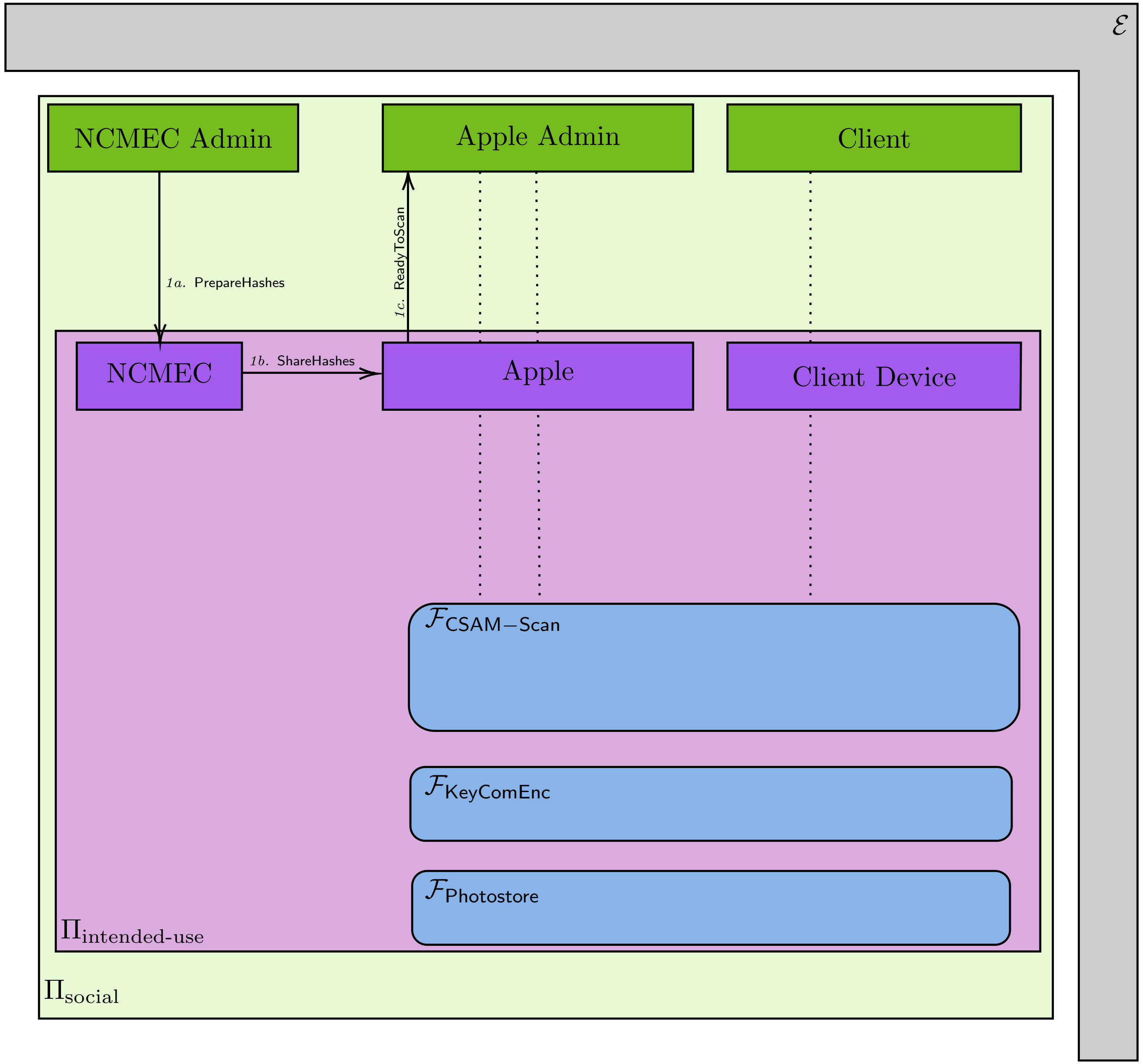
   //$\mathcal{F}$ alerts the Server that the user has uploaded a photo.

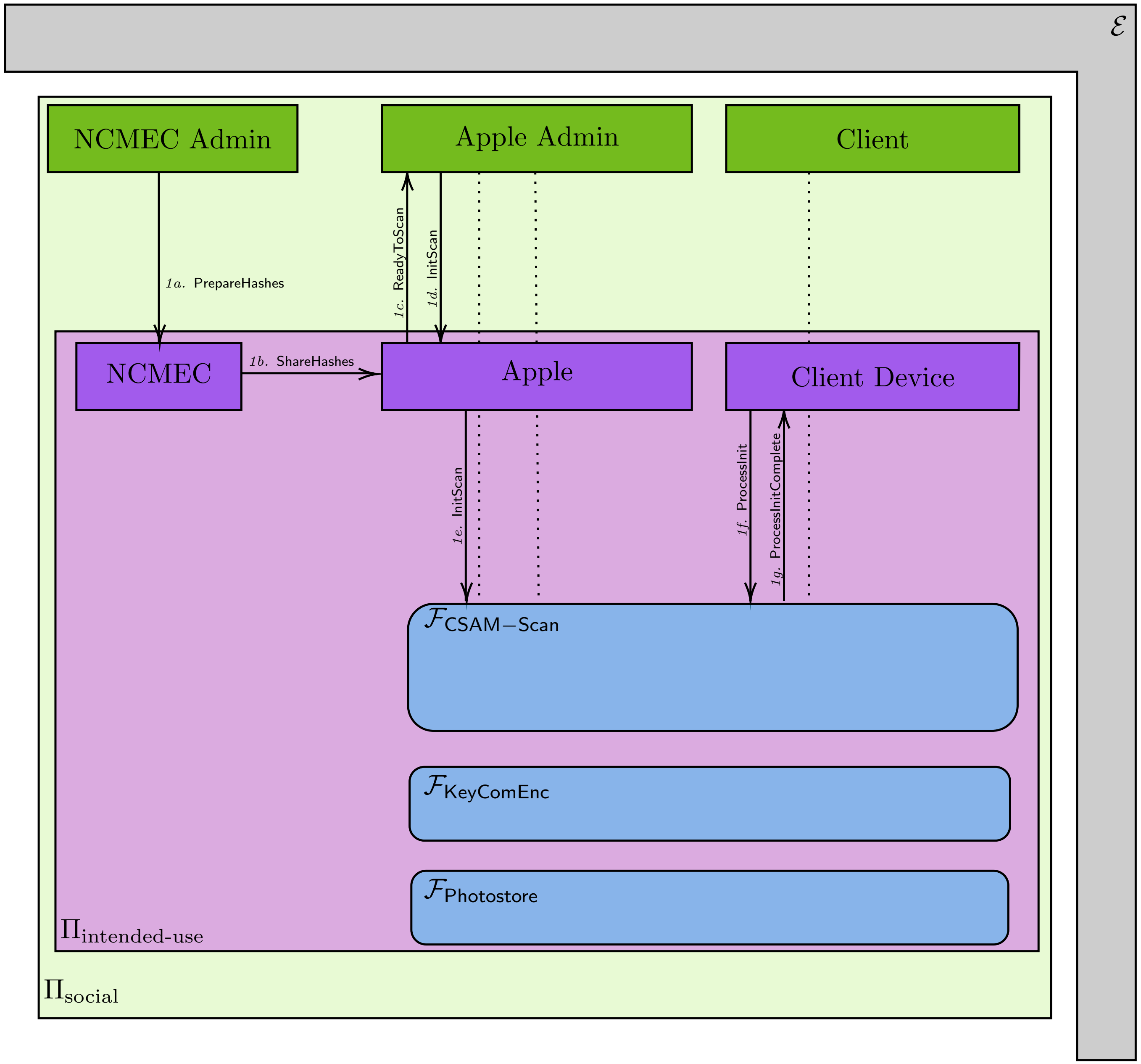
2. If $\mathsf{img\_hash} \notin X_{\mathcal{U}}$ or $\mathsf{valid} = \mathsf{false}$, send $(\mathsf{ScanImageComplete}, \mathcal{U}, \mathsf{false})$ to $APPLE$.
3. If $\mathsf{img\_hash} \in X_{\mathcal{U}}$, increment $\mathsf{counter}_{\mathcal{U}}$, append $\mathsf{img\_id}$ to $\mathsf{photolist}_{\mathcal{U}}$ and append k to $\mathsf{keys}_{\mathcal{U}}$. Then,

   (a) If $\mathsf{counter}_{\mathcal{U}} < \mathcal{T}$ send $(\mathsf{ScanImageComplete}, \mathcal{U}, \mathsf{true})$ to $APPLE$.

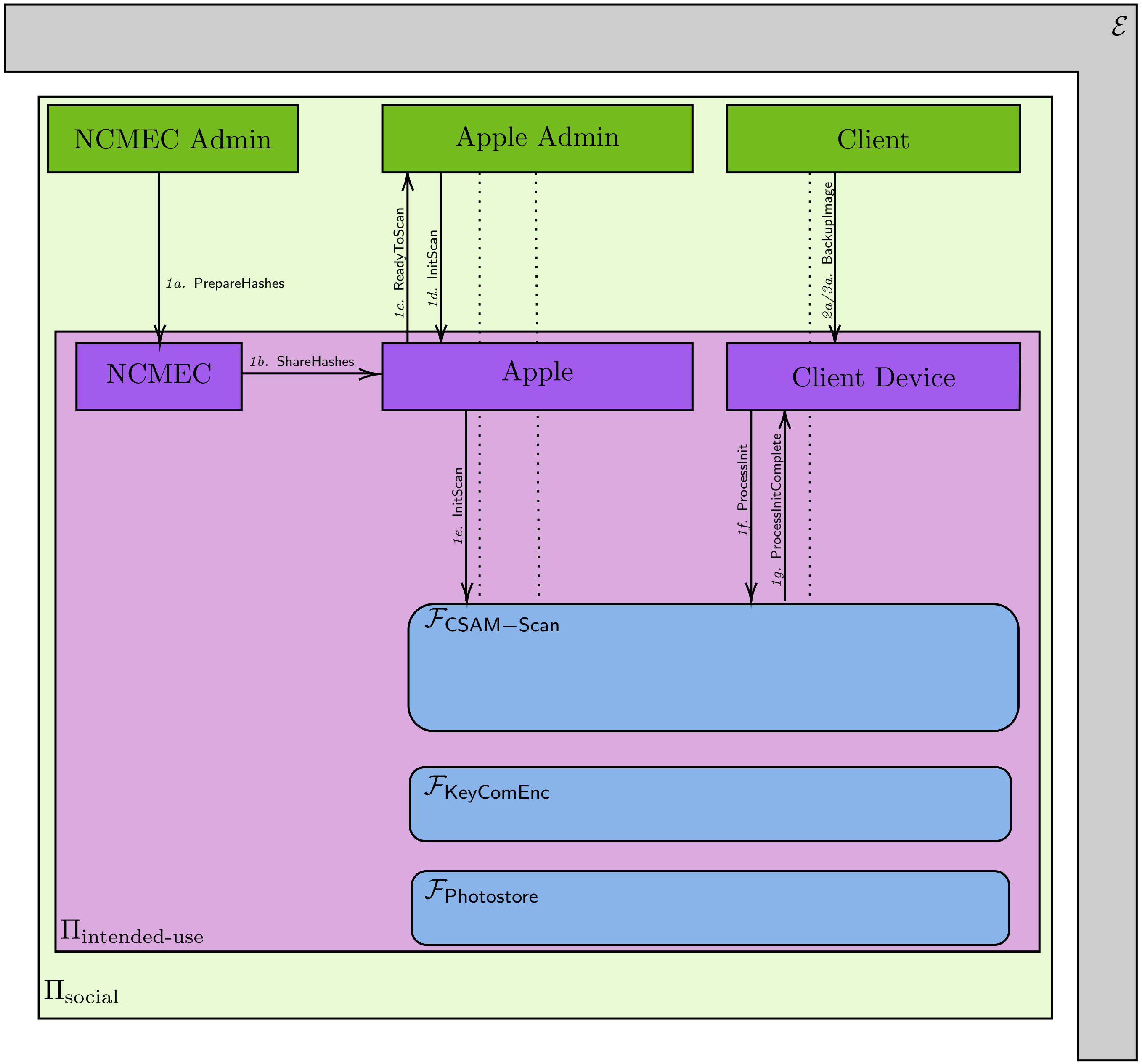
   (b) If $\mathsf{counter}_{\mathcal{U}} \geq \mathcal{T}$, send $(\mathsf{ThresholdMet}, \mathcal{U}, \mathsf{photolist}_{\mathcal{U}}, \mathsf{keys}_{\mathcal{U}})$ to $APPLE$.


$\mathcal{E}$
NCMEC Admin
Apple Admin
Client
NCMEC
Apple
Client Device
$\mathcal{F}_{\mathsf{CSAM-Scan}}$
$\mathcal{F}_{\mathsf{KeyComEnc}}$
$\mathcal{F}_{\mathsf{Photostore}}$
$\Pi_{\text{intended-use}}$
Π_{social}

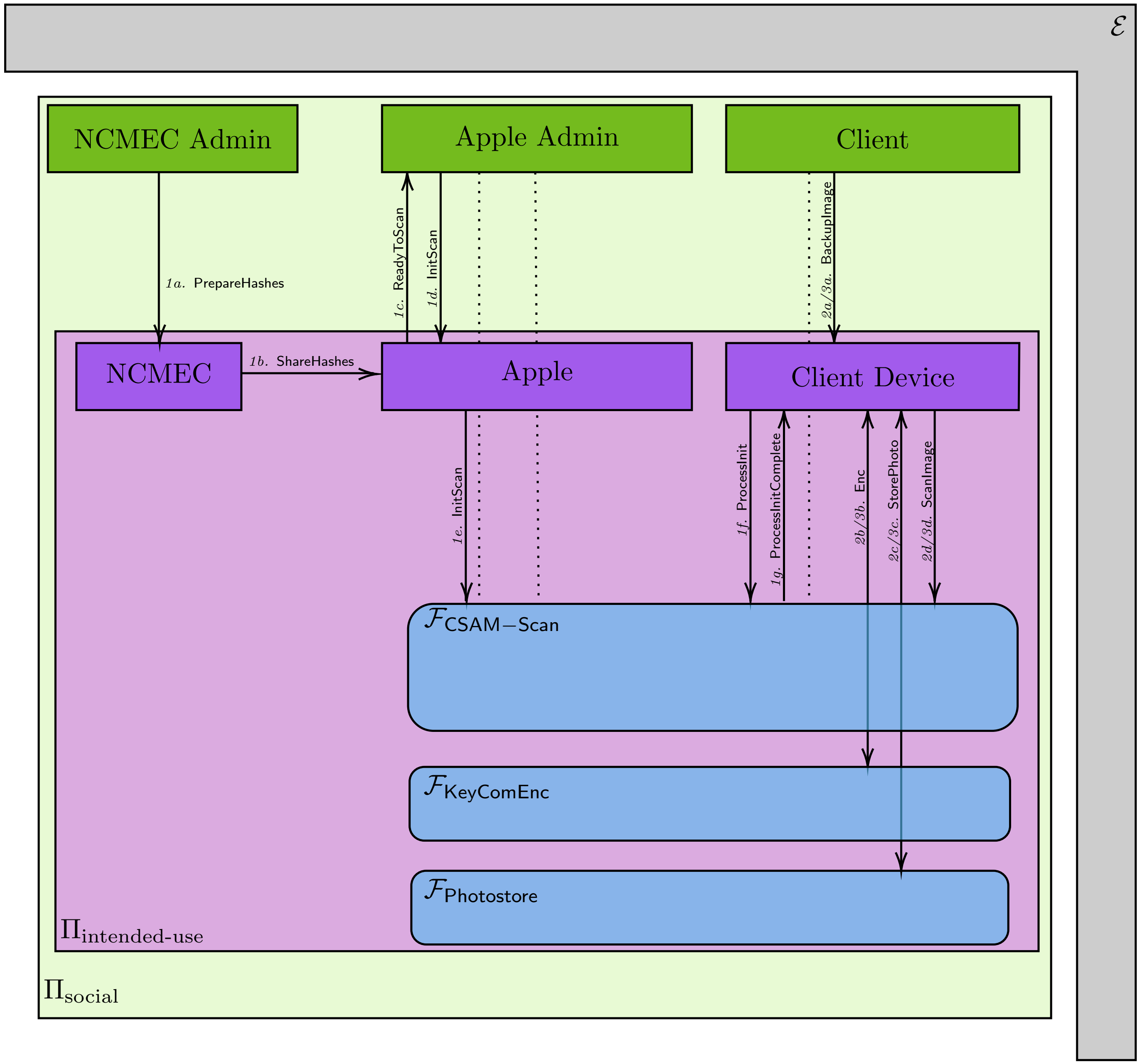

$\mathcal{E}$
NCMEC Admin
Apple Admin
Client
1a. PrepareHashes
1c. ReadyToScan
NCMEC
1b. ShareHashes
Apple
Client Device
$\mathcal{F}_{\mathsf{CSAM-Scan}}$
$\mathcal{F}_{\mathsf{KeyComEnc}}$
$\mathcal{F}_{\mathsf{Photostore}}$
$\Pi_{\text{intended-use}}$
Π_{social}



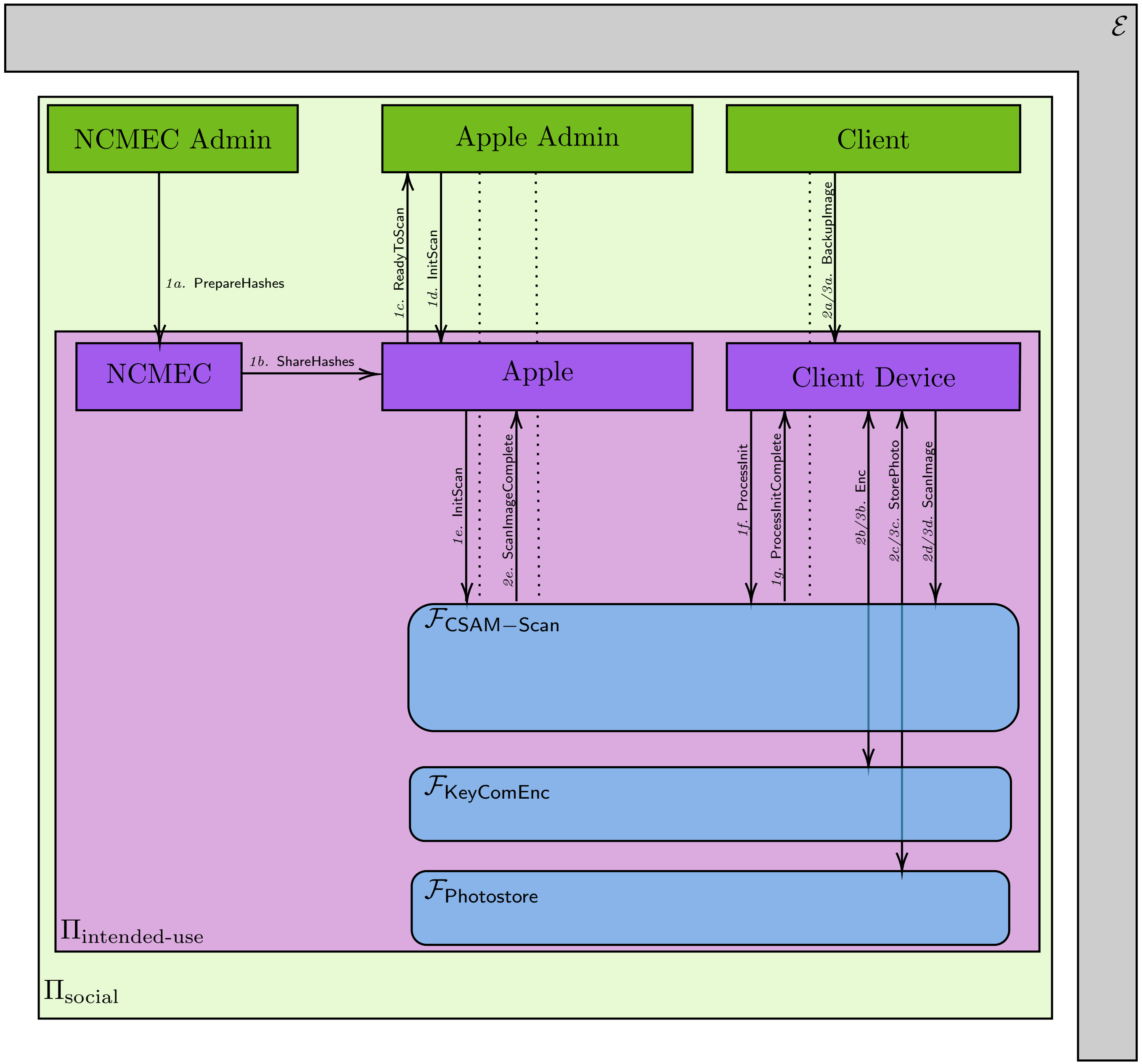
$\mathcal{E}$
NCMEC Admin
Apple Admin
Client
1a. PrepareHashes
1c. ReadyToScan
1d. InitScan
NCMEC
1b. ShareHashes
Apple
Client Device
1e. InitScan
1f. ProcessInit
1g. ProcessInitComplete
$\mathcal{F}_{\mathsf{CSAM-Scan}}$
$\mathcal{F}_{\mathsf{KeyComEnc}}$
$\mathcal{F}_{\mathsf{Photostore}}$
$\Pi_{\text{intended-use}}$
Π_{social}



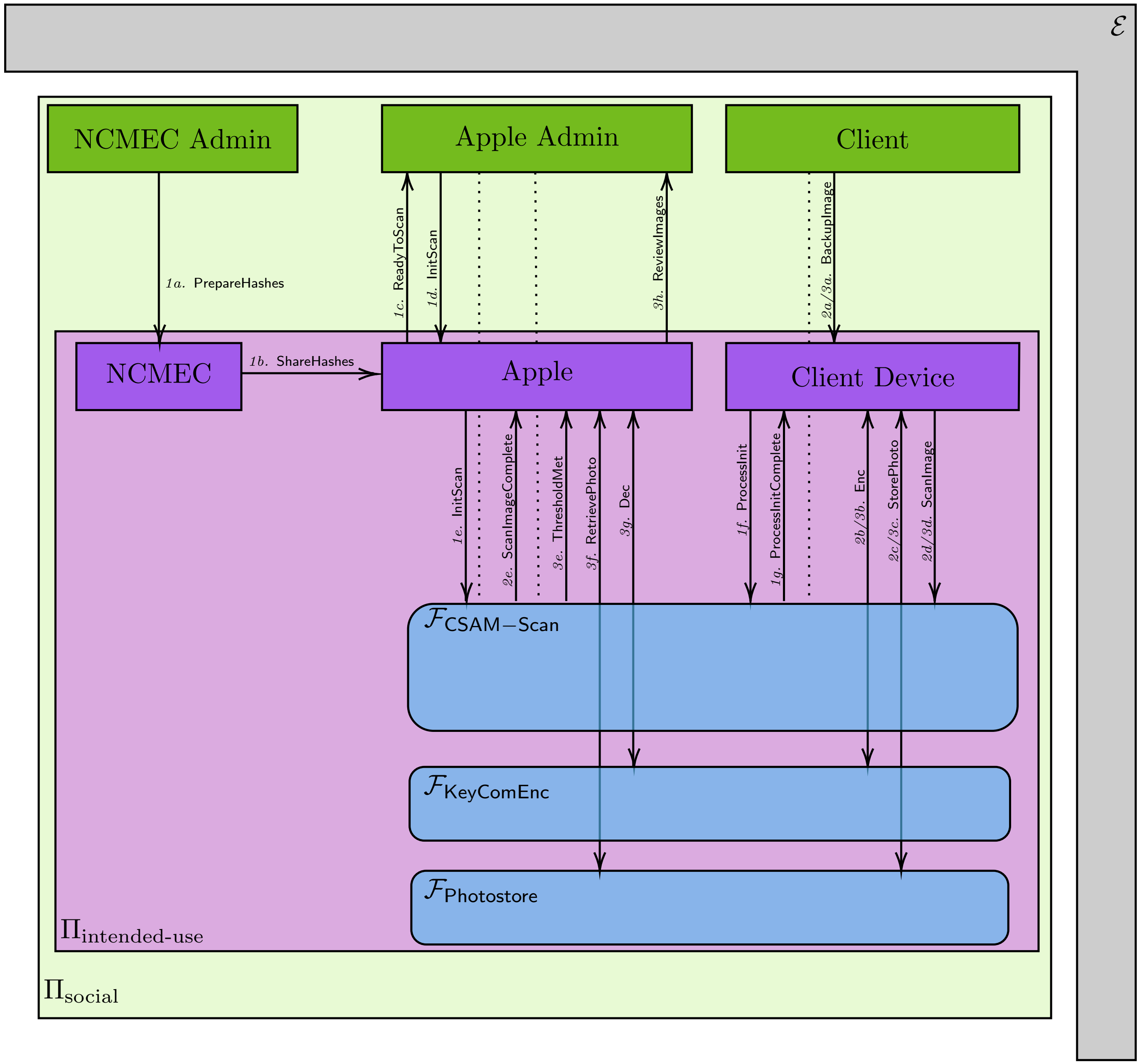
$\mathcal{E}$
NCMEC Admin
Apple Admin
Client
1a. PrepareHashes
1c. ReadyToScan
1d. InitScan
2a/3a. BackupImage
NCMEC
1b. ShareHashes
Apple
Client Device
1e. InitScan
1f. ProcessInit
1g. ProcessInitComplete
$\mathcal{F}_{\mathsf{CSAM-Scan}}$
$\mathcal{F}_{\mathsf{KeyComEnc}}$
$\mathcal{F}_{\mathsf{Photostore}}$
$\Pi_{\text{intended-use}}$
Π_{social}



$\mathcal{E}$
NCMEC Admin
Apple Admin
Client
1a. PrepareHashes
1c. ReadyToScan
1d. InitScan
2a/3a. BackupImage
NCMEC
1b. ShareHashes
Apple
Client Device
1e. InitScan
1f. ProcessInit
1g. ProcessInitComplete
2b/3b. Enc
2c/3c. StorePhoto
2d/3d. ScanImage
$\mathcal{F}_{\mathsf{CSAM-Scan}}$
$\mathcal{F}_{\mathsf{KeyComEnc}}$
$\mathcal{F}_{\mathsf{Photostore}}$
$\Pi_{\text{intended-use}}$
Π_{social}



$\mathcal{E}$
NCMEC Admin
Apple Admin
Client
1a. PrepareHashes
1c. ReadyToScan
1d. InitScan
2a/3a. BackupImage
NCMEC
1b. ShareHashes
Apple
Client Device
1e. InitScan
2e. ScanImageComplete
1f. ProcessInit
1g. ProcessInitComplete
2b/3b. Enc
2c/3c. StorePhoto
2d/3d. ScanImage
$\mathcal{F}_{\mathsf{CSAM-Scan}}$
$\mathcal{F}_{\mathsf{KeyComEnc}}$
$\mathcal{F}_{\mathsf{Photostore}}$
$\Pi_{\text{intended-use}}$
Π_{social}



$\mathcal{E}$
NCMEC Admin
Apple Admin
Client
1a. PrepareHashes
1c. ReadyToScan
1d. InitScan
3h. ReviewImages
2a/3a. BackupImage
NCMEC
1b. ShareHashes
Apple
Client Device
1e. InitScan
2e. ScanImageComplete
3e. ThresholdMet
3f. RetrievePhoto
3g. Dec
1f. ProcessInit
1g. ProcessInitComplete
2b/3b. Enc
2c/3c. StorePhoto
2d/3d. ScanImage
$\mathcal{F}_{\mathsf{CSAM-Scan}}$
$\mathcal{F}_{\mathsf{KeyComEnc}}$
$\mathcal{F}_{\mathsf{Photostore}}$
$\Pi_{\text{intended-use}}$
Π_{social}

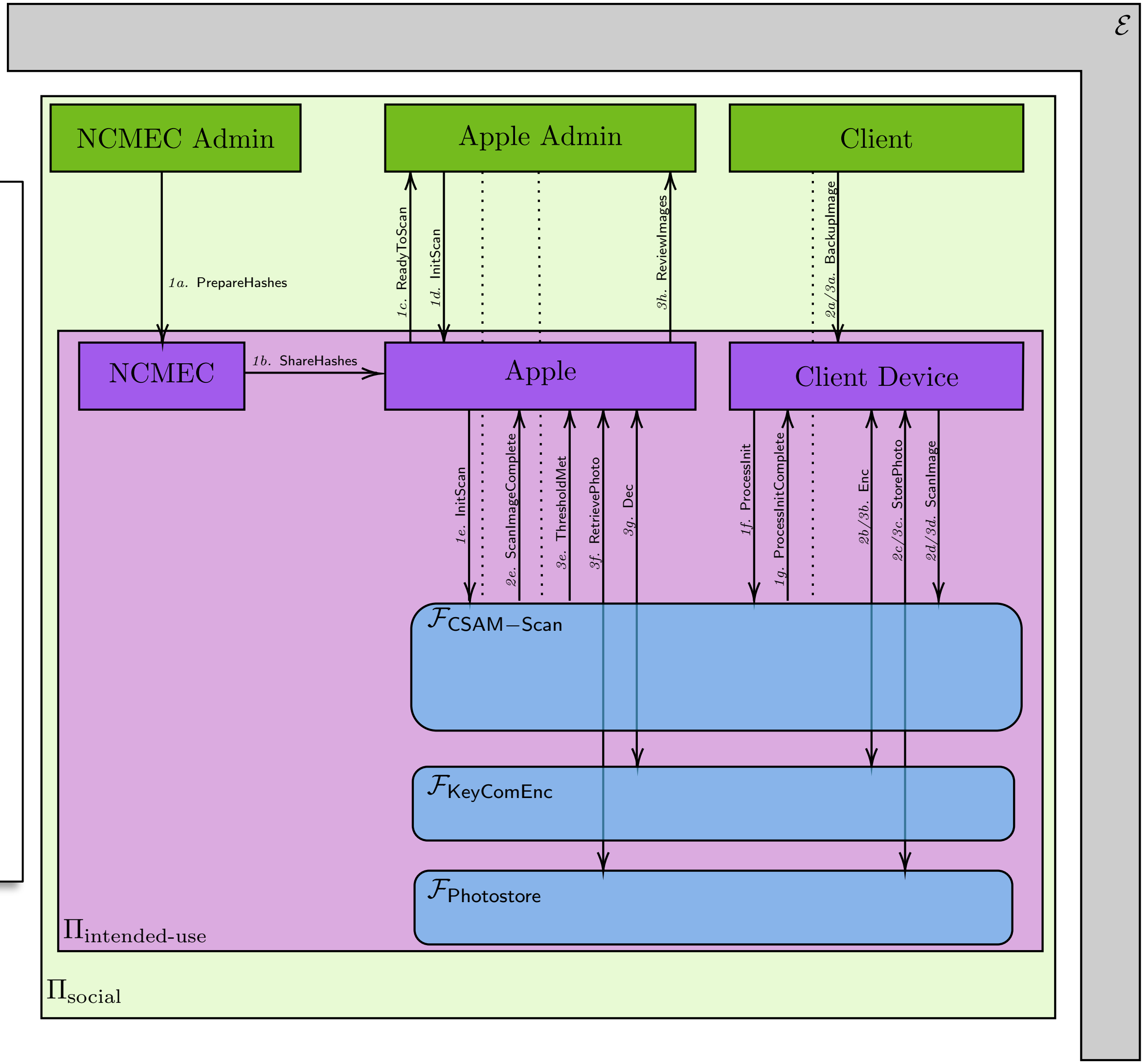
Does this *actually* help?

Top Down Analysis:

CSAM Detection provides these privacy and security assurances:

- Apple does not learn anything about images that do not match the known CSAM database.
- Apple can't access metadata or visual derivatives for matched CSAM images until a threshold of matches is exceeded for an iCloud Photos account.
- The risk of the system incorrectly flagging an account is extremely low. In addition, Apple manually reviews all reports made to NCMEC to ensure reporting accuracy.
- Users can't access or view the database of known CSAM images.
- Users can't identify which images were flagged as CSAM by the system.

For detailed information about the cryptographic protocol and security proofs that the CSAM Detection process uses, see The Apple PSI System.

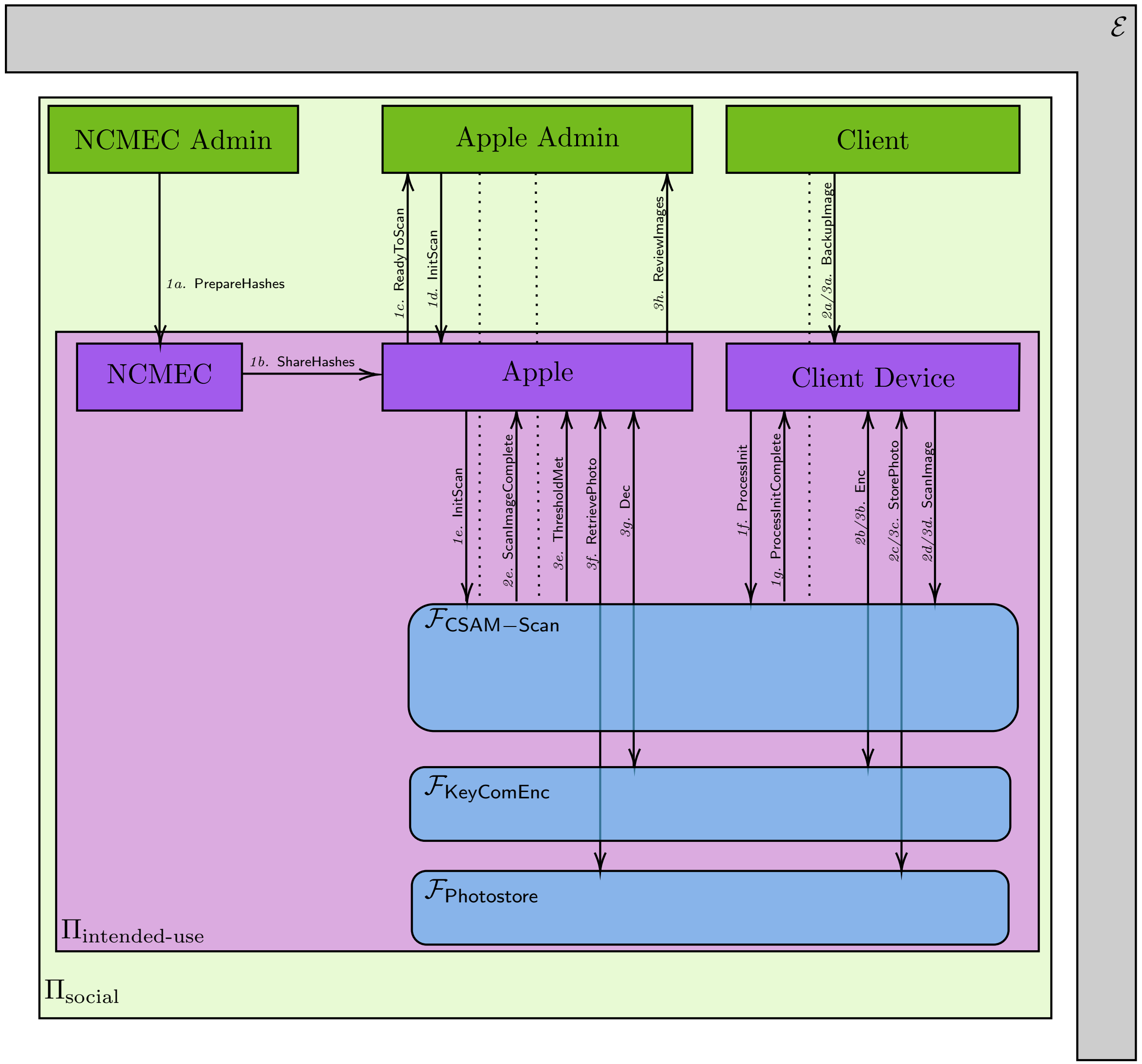


https://web.archive.org/web/20210805191352/https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf

Does this *actually* help?

Bottom Up Analysis:

- What are the consequences of divergent/surprising/non-normative choices in Π_{social}?
- What are the consequences of malicious/buggy software running $\Pi_{intended-use}$?
- What if software dependencies within $\Pi_{intended-use}$ do not have desired properties?



https://web.archive.org/web/20210805191352/https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf

Where do we go from here?

More Social properties!

Potential Additional Properties:

1. Client devices can detect if apple is attempt to scan for any perceptual hashes not designated as CSAM by Apple [SchefflerKulshresthaMayer23]

2. A pre-determined judge must approve of any image decryptions before they are to happen

3. Even if the pre-determined judge becomes compromised, images can only be decrypted after the match threshold has been met

4. False positives should leave an indelible audit trail

5. The system should be publicly auditable. Namely, the public should be able to verify:
    - That the server is using the same set of hashes for all clients
    - That the perceptual hashes for which the server is scanning are those selected by NCMEC
    - That any images that the server claims were the decrypted as a result of the system were actually uploaded by the accused client

6. Clients can only use the photostore if they honestly use the scanning system

Key Take Aways:

1

We desperately need new analysis tools that prevent privacy theater

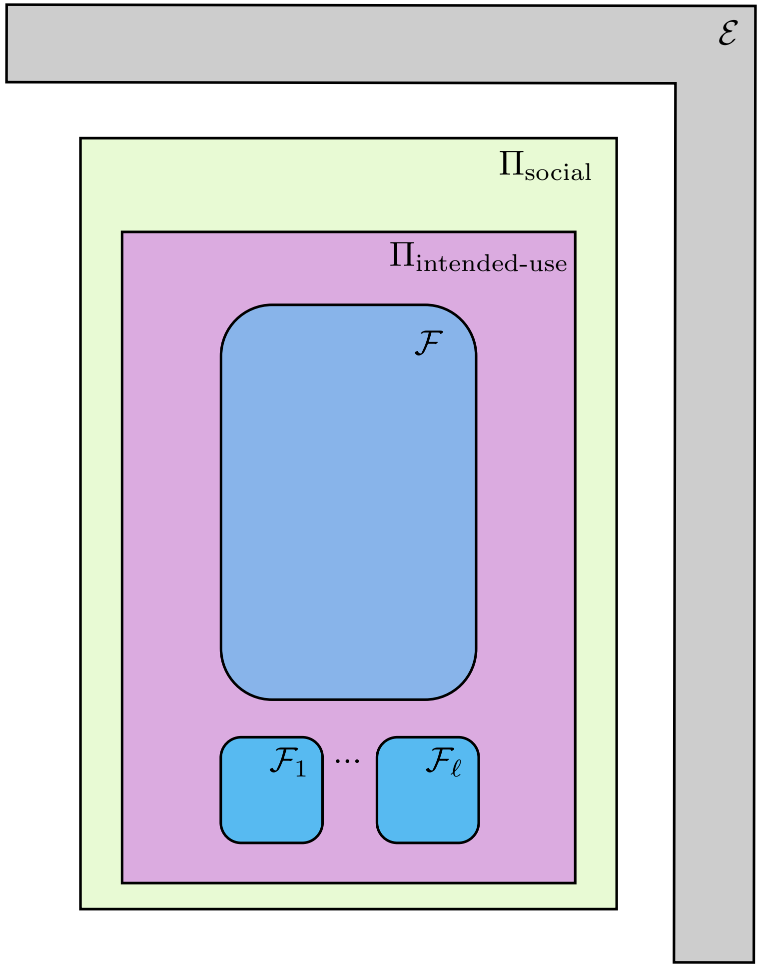
2

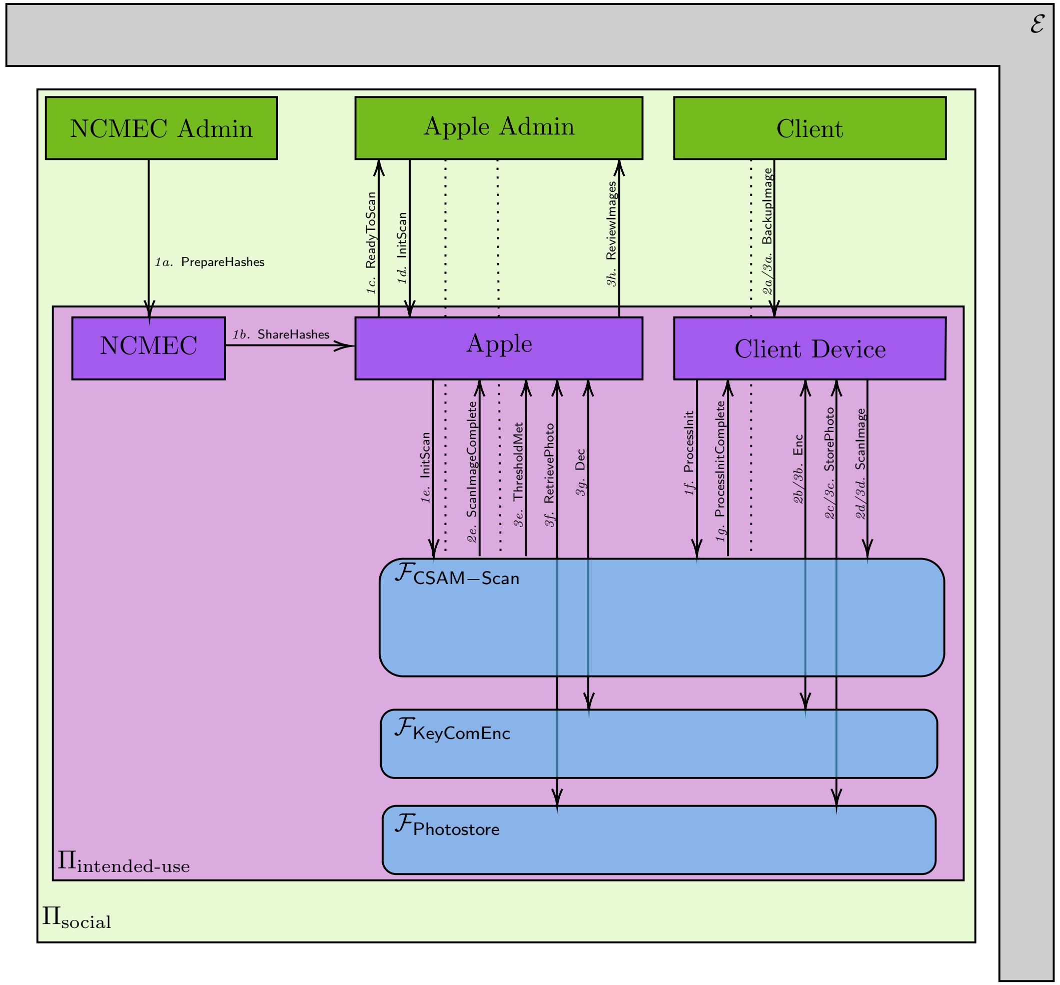
There is an opportunity to create Interdisciplinary boundary objects [Star and Griesmemer 1989]

3

Apple’s proposed system is a valuable case study and it remains understudied

Thanks!





Gabriel Kaptchuk --- kaptchuk@umd.edu --- https://kaptchuk..com