Analyzing Cryptography in Context: A Cryptography-Native Approach to Threat Modeling

Ran Canetti*[1], Julie Ha*[1], and Gabriel Kaptchuk*[2]

[1]Boston University, {canetti,hajulie}@bu.edu
[2]University of Maryland, College Park, kaptchuk@umd.edu

Abstract

We observe that the existing norms within cryptography do not expect protocol analysts to document the sociotechnical properties that a deployed system should have. To help close this potential gap, we develop a framework that allows bringing sociotechnical dimensions into analyses of cryptographic systems, and in particular facilitates "in-context" analysis of proposed cryptographic deployments on top of widely-accepted cryptographic modeling techniques. To explore the utility of our framework, we use Apple's 2021 CSAM scanning proposal as a case study. We show how our framework naturally surfaces many of the criticisms of Apple's proposal and helps us identify a previously undocumented property of the proposal.

1 Introduction

All cryptographic protocols are expected to have an accompanying security proof in a compelling and believable security model. But, designers have a heightened responsibility to document their system's properties when they will be deployed, as people's privacy will be at risk. Responsible deployment, therefore, demands carefully considering the sociotechnical properties of the system, including both cryptographic and non-cryptographic components, and studying the ethical ramifications of deployment.

While security proofs are a convincing way to demonstrate that a protocol realizes a set of idealized properties, we often lack the tools to systematically evaluate sociotechnical properties and ethics. For example, proving that a protocol meets the formal definition of secure multiparty computation (MPC) [23, 27, 36, 75] does not ensure that the MPC's inputs match the context's goals, nor does it prevent computation of an inappropriately disclosive circuit. This work develops a framework aimed at incorporating sociotechnical dimensions within the core analysis of cryptographic systems, rather than leaving such properties out of scope.

A real-world example. The gaps in our existing analysis tools are obvious when we look at recent real-world proposed deployments. Take, for example, Apple's 2021 announcement [15] that it would scan end-to-end encrypted photographs stored in iCloud for known instances of child sexual abuse material (CSAM) using a threshold private set intersection protocol. While the soundness of the security proofs accompanying the announcement was largely unquestioned, critics noted that the proposed system put too much trust in Apple's operators and gave clients no ability to detect abuse.

The vast majority of the analysis of this incident has focused on *ethics* (i.e., *should* such a system be deployed?) [5,26,30,32,35,39,40,45,48,52–54,58,59,64,68]. We, instead, focus on the *process* issue, which has been largely overlooked: Apple's team produced a report [13] that met community expectations but failed to highlight many of the aspects of their proposal that were found to be objectionable. That is, because the analysis they conducted was tightly scoped, the research community had to independently extract the sociotechnical properties associated with deployment—without the benefit of being Apple insiders—before even beginning to debate *ethics*. While Apple's analysis followed cryptographic best practice, we conjecture that had Apple's analysis incorporated some of the sociotechnical aspects of the system that were uncovered after release, both the system itself and the reaction to it might have been different. For this to happen, though, the community needs to have a methodology for incorporating societal aspects within the core security analysis.

Cryptography-native threat modeling. We argue that there are three, conceptually distinct steps when it comes to responsibly preparing a cryptosystem for deployment:

(1) proving that it has a set of properties using standard, cryptographic proof techniques,

(2) establishing the sociotechnical properties the cryptographic deployment will have, i.e., the way in which it interacts with its deployment context, and

* Authors are listed alphabetically following the norm in theoretical computer science.

(3) arguing that deployment is "good," "contextually appropriate," or "ethical."

Each of these steps is a *precondition* that informs the following step, as idealized properties dictate a cryptosystem's interaction with its context and failing to carefully account for the system's sociotechnical properties can radically change the ethical calculus. Moreover, different groups hold responsibility for the steps. Responsibility for (1) and (2) lies solely with cryptographers, who have the technical knowledge to understand the intricacies of the proposed cryptosystems. Effectively doing (3), on the other hand, requires an interdisciplinary team capable of weighing the benefits and risks of deployment from many perspectives.

In this work, we initiate the study of step (2), hopefully paving the way for future work on step (3). We approach step (2) by asking the following question:

What analytical tool should we use to establish the in-context, sociotechnical properties of a cryptographic deployment?

We recognize that building a consensus answer to this question requires full community engagement—and there may be no perfect tool. Our goal, therefore, is not to provide the final answer to this question, but rather to offer a candidate tool that can start this important community-wide conversation.

We observe that *threat modeling* appears to be a promising starting point for developing such a tool, as threat modeling asks analysts to identify the ways in which different system components (both technical and social) interact and the implicit trust assumptions the system makes. Moreover, there is a robust and well-developed threat modeling literature that is often used in practice. Unfortunately, directly importing existing tools poses difficulties, as they are fundamentally non-interoperable with the modeling techniques already commonly used in the cryptographic community and operate at the wrong layer of abstraction (e.g., surfacing the software that implements cryptography, which all cryptographers assume should be correct). Thus, we must develop cryptographic-native analogs of these techniques that will feel natural for cryptographers to use.

1.1 Our Contributions

Our approach demonstrates how existing cryptographic modeling tools can be adapted in a natural way to support the type of sociotechnical analytical work in which we are interested. Specifically, we make the following technical contributions:

1. A Cryptography-Native Threat Modeling Technique. We develop a threat modeling technique within the Universal Composability (UC) framework [25]. Indeed, the UC framework's ability to represent a system at multiple levels of abstraction and detail without losing rigor or precision makes it a convenient platform for capturing sociotechnical properties of security systems. In particular, our framework allows capturing such properties at one level of abstraction and then arguing about how such properties are impacted by implementation details which appear only at significantly lower levels.[1] Our approach asks protocol designers to situate ideal functionalities that will be realized cryptographically within a larger set of functionalities that capture the capabilities of the *full system*, connecting these functionalities with the following, concentrically composed protocols:

(1) $\Pi_{\text{Intended-use}}$: a protocol describing the envisioned functioning of non-cryptographic software components of the system. Importantly, specifying behavior within $\Pi_{\text{intended-use}}$ (as opposed to an ideal functionality) is an indication that no formal (i.e., cryptographically verifiable) guarantees are going to be made that this software will follow the prescribed steps.

(2) Π_{Social} : a protocol describing elements of the system that will not (or cannot) be rendered into software. Generally, these steps are ones for which it is difficult to determine if behavior is malicious.

When seen through the lens of the threat modeling literature, our framework is best understood as a cryptography-native data-flow diagram [33] which can be used to examine claimed sociotechnical system properties. These claimed properties can inform a separate, possibly interdisciplinary, discussion determining if the system *should* be deployed.

2. Case Study: Apple's CSAM Scanning Proposal. To study our threat modeling approach, we apply our framework to a real-world, proposed cryptographic system: Apple's CSAM scanning proposal. In Section 3, we use the framework to retrospectively analyze Apple's initial proposal. We find that this process (a) captures many of the research community's existing critiques, (b) helps identify weaknesses in Apple's initial cryptographic analysis and (c) pinpoints at least one meaningful risk associated with deployment that, to our knowledge, has not been publicly discussed despite multiple years of heightened scrutiny. Specifically, we observe that storing the images scanned by Apple's threshold PSI scheme using convergent encryption can leak the identities of users storing a sub-threshold number of CSAM images. We observe, however, that our framework is limited in the ways that it supports reasoning about heuristic properties of systems, like the semantic hash function used in Apple's proposal.

In the full version of the paper, we explore using our framework proactively within the cryptographic design

[1] Still, the conceptual insights motivating our work are not specific to the UC framework. We anticipate that the community will likely need an array of threat modeling frameworks, including some that interoperate with other proof approaches (e.g., game-based security definitions).

process by first specifying sociotechnical properties and then designing to meet them.

Why choose Apple's CSAM scanning proposal? When selecting a case study for this work, we required a proposal that realistically would be subjected to this type of analysis in the real world, both because it offers sufficient technical complexity and promises to have a rich social context. This ruled out "toy" examples (e.g., the X.509 ecosystem), as they would not help us learn how our framework handled complexity. Moreover, we needed a case study that had been subject to at least some ad-hoc sociotechnical analysis, so we could check if our framework systematically replicated these findings. Without this baseline against which we could compare, it would be difficult to have confidence that our framework performed at some minimum acceptable level. Apple's CSAM scanning proposal satisfied all these criteria.

1.2 Related and Concurrent Work

Our work is inspired by Rogaway's foundational work [65], in addition to recent similar work by Kamara [2, 3], Borradaile [24], and Rosenbloom [66], that highlights the need to think about cryptographic systems within the social context in which they will be deployed. Our work differs from theirs by proposing a concrete change to the ways in which cryptographic protocols are documented in service of this shared goal. We are similarly inspired by analyses of real-world deployed cryptosystems that demonstrate gaps between claimed properties and actual guarantees, e.g., [7, 9–12, 18, 34, 42, 73]. These works typically are post-deployment analyses, focusing on cryptography "in the wild," whereas the analyses we consider are intended to be pre-deployment. While these works typically focus on a single deployed system, some of these works, e.g., [73], demonstrate the broader risk of analyzing only sub-components of a larger cryptographic system.

Concurrently with our work, Albrecht et al. [8] put forward a compelling alternative vision for understanding the sociotechnical properties of cryptographic systems. Specifically, they outline a process by which researchers deeply embed themselves in a particular deployment context, leveraging observation and grounded theory. This approach can identify security properties that a cryptosystem should provide in order to be effective. We see our works as complementary, solving distinct but related problems. A key difference between our works is that we design a process that every real-world proposal can follow, whereas the type of ethnographic work they consider is not cryptography-native, making it a challenging candidate for broad adoption. Moreover, their process does not directly lend itself to providing documentation. On the other hand, their approach is significantly more thorough and can document the ways in which users interact with cryptographic systems in practice.

2 A Framework for "In-Context" Analysis

Design goals. We begin by specifying our design goals:

(1) *Broaden the analytical frame.* Existing tools for analyzing cryptographic systems focus on modularity and generalizability such that the cryptography can be plugged into future applications. A side effect of this approach is that it naturally suppresses the details of how the cryptographic components of the system will fit into the larger digital ecosystem. Analysis of these parts of the deployment is—at best—reserved for another document and is often performed without any supporting framework or critical review by security experts, despite their critical importance for analyzing the privacy implications of deployment. We require that our framework provides the affordances necessary to broaden the scope of the analysis performed by system designers.

(2) *Conceptually lightweight, or "cryptography-native."* We require that a new tool does not introduce significant conceptual complexity (beyond the non-trivial complexity associated with standard cryptographic analysis). Requiring too much learning or labor in order to leverage an analytical framework will limit uptake. If we believe that doing this kind of thorough analysis is the "right thing to do," we should make efforts to encourage broad usage. Looking ahead, we do this by piggybacking onto the language of ideal functionalities which are familiar to much of the cryptographic community.

(3) *Systematic and enumerative.* We require that the framework for conducting this analysis provides a clear script for system designers to follow. If a framework is systematic—like existing formal proof techniques—then it acts as compelling evidence, both to the author and the reader, that the system in question has been thoroughly considered. Any new framework that we propose should similarly take a systematic approach to analysis and leave obvious evidence when elements of the analysis have been skipped. We note, however, we stop short of trying to *prove* sociotechnical properties of systems (in a formal sense), but rather make analyzing them more systematic.

(4) *Highlights implications of trust choices.* Because cryptography enables system designers to re-imagine the trust relationships required to carry out computational tasks, we require that our framework must facilitate a clear understanding of the ways in which trust is allocated throughout the entire system. This includes highlighting the ways in which cryptography can minimize the trust required for the system to operate and aspects of the trust allocation that are beyond the cryptography's reach.

Why not use existing threat modeling tools? Given these goals, it might seem natural to reach for an existing tool from the threat modeling literature (we provide an in-depth background on threat modeling in the full version of the paper). Threat modeling tools are used in practice to analyze deployed systems and identify the ways in which they can be

used and abused. Importantly, they already aim to have a broad analytical scope (e.g., including the *cost* of social attacks), intend to be systematic and enumerative, and highlight the implications of trust choices. Unfortunately, they fail to be cryptography-native, making direct application challenging.

For example, consider trying to apply STRIDE [43] to a complex, cryptographic system. In STRIDE, the analyst begins by rendering the system into a data-flow diagram [33], divides the visualized system into trust zones, and then iterates through a set list of potential malicious actions each actor could take, mapping the implications of each of these actors. The key challenge in this approach is designing a data-flow diagram which captures cryptographic nuance. For example, cryptographic protocols include many messages that do not directly reveal information (or reveal only partial information about data held by protocol participants). Should these messages be captured in the data-flow diagram, and if so, how should they be labeled? Moreover, cryptographic protocols could be run over time, with non-cryptographic interactions interleaved between cryptographic messages, meaning it is not always possible to simply insert multiple rounds of cryptographic interaction into the data-flow diagram, annotated with the eventual outputs to the participating parties. More generally, how should the wide variety of possible malicious behaviors (which could *change* the information logically communicated by a cryptographic protocol) be captured within the diagram? And how should the proven properties of these messages be represented? These questions only scratch the surface. Simply put, the ways in which we formalize cryptographic systems do not naturally fit in with STRIDE's data-flow diagrams.

Another barrier to using existing threat modeling tools is that they are designed for software architects, and thus place implementation details in scope. For example, STRIDE's data-flow diagram should capture the various software components in the system (app, web browser, device resources, load balancers, databases, etc...), none of which may independently hold the cryptographic properties of the system. This is a mis-match for cryptographers as (1) cryptography generally assumes that the software implementation is correct, and (2) treating the system as software, rather than a set of properties, is likely to *distract* from the core, cryptographic parts of the system.

If we cannot leverage threat modeling tools directly, we should instead attempt to create cryptographic-native analogs of the best parts of threat modeling. Inspired by the above difficulties using existing tools with cryptography, we build a cryptographic-native version of the data-flow diagram.

2.1 Our Framework

As our starting point, we take the simulation-based security paradigm and Canetti's UC framework [25] in particular. The main motivation behind this choice is that simulation-based security allows cryptographers to define the properties of a cryptosystem using an ideal functionality. A well-written ideal functionality provides a clear rendering of both the security properties of the system and the way in which users can interact with the system. This level of abstraction cleanly matches the data-flow diagrams leveraged in established threat modeling tools, in which analysts are asked to illustrate the way data moves between system components. As our framework is intended to situate a cryptographic protocol within its deployment context, treating the protocol as a black box with some set of properties—an ideal functionality, in other words—is the right fit for the task at hand. While it would be possible to accomplish these goals using stand-alone simulation, UC offers a clean, well-defined run-time for analysis and composability guarantees. Moreover, part of studying a cryptosystem "in-context" is illustrating how it will be composed with other system components, making composability an attractive feature.

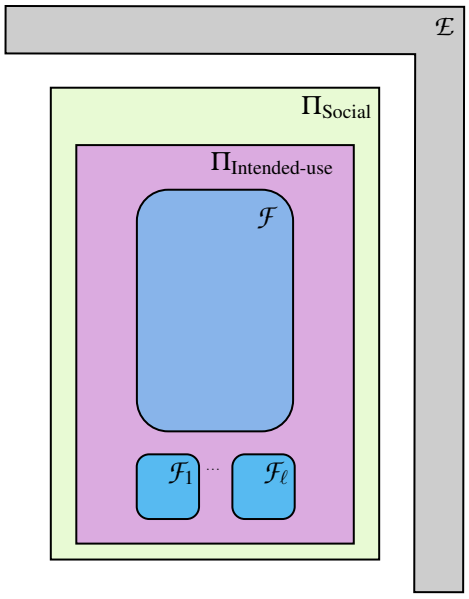

Figure 1: A visualization of our approach. In this rendering, our ideal functionality of interest, $\mathcal{F}$, is composed with several other ideal functionalities, $\mathcal{F}_1, \ldots, \mathcal{F}_\ell$, by $\Pi_{\text{Intended-use}}$, which specifies the expected functioning of the software components of the system not captured by $\mathcal{F}$. The protocol $\Pi_{\text{Intended-use}}$ is further wrapped by Π_{Social}, which describes the aspects of the system that are driven by human decisions or system components for which there is no "correct" behavior.

Step 0: Defining the ideal functionality $\mathcal{F}$. We assume that the protocol designers have some protocol Π and an ideal functionality $\mathcal{F}$ that they believe Π should realize. Before interacting with our framework, the protocol designers should separately provide a proof that Π realizes $\mathcal{F}$, allowing them to work with $\mathcal{F}$ alone.

Step 1: Extending the modeling. Next, the protocol designer puts $\mathcal{F}$ into context by extending the modeling they have done to cover the full system. Importantly, this modeling is going to use the same paradigms (i.e., language, level of abstraction, etc.) the cryptographer has already used to define $\mathcal{F}$ and prove that Π realizes it. Namely, the protocol designer specifies the following two *protocols*, as visualized in Figure 1:

$\Pi_{\text{Intended-use}}$: This protocol describes the ways that honestly written *software* is expected to interact with $\mathcal{F}$. Most notably, this includes specifying the inputs that are expected to be sent to $\mathcal{F}$ and explicitly describing the ways in which those inputs should be interleaved with other cryptographic components $\mathcal{F}_1, \ldots, \mathcal{F}_\ell$.[2] For example, $\Pi_{\text{Intended-use}}$ might specify that an output of $\mathcal{F}_1$ should be fed into a specific interface of $\mathcal{F}$. Alternatively, this might include the order and timing with which a protocol participant calls the multiple interfaces of $\mathcal{F}$.

By putting elements of the proposed system in $\Pi_{\text{Intended-use}}$ rather than $\mathcal{F}$, the designer is explicitly stating that cryptographic techniques will not be used to verify their correct operation. Indeed, because $\Pi_{\text{Intended-use}}$ is a *protocol*, rather than an ideal functionality, there is no proving required—and it is possible for software implementing these parts of the protocol to deviate maliciously without detection. Just as attempting (and failing) to formally reduce the security of a cryptographic protocol helps designers identify protocol weaknesses or required properties of cryptographic building blocks, specifying $\Pi_{\text{Intended-use}}$ highlights risks from malicious software.

Π_{Social}: Some system components will never be—or can never be—written in software because there is no clear expected behavior. Nonetheless, understanding the implications of these parts of the system is crucial. Π_{Social} provides an opportunity for protocol designers to specify these components. We call this part of the system "social" because we anticipate that much of this protocol will be mediated by human decision-making. These decisions—in which there is no clear *correct* decision—are inevitable in instances where digital systems meet the physical world. For example, Π_{Social} will often be responsible for capturing data and importing it into the system (e.g., manual data entry, recording audio data, etc…).

Formally specifying Π_{Social} poses a modeling challenge: *if we do not know of a PPT algorithm that captures what* should *happen in* Π_{Social}*, how can we describe it?* Conceptually, we can solve this by thinking of Π_{Social} as describing the *social meaning* of an arbitrary PPT algorithm that sends messages of the right form.

These two protocols are composed concentrically (see Figure 1); Π_{Social} makes subroutine calls to $\Pi_{\text{Intended-use}}$, and $\Pi_{\text{Intended-use}}$ makes calls to $\mathcal{F}$ (along with, optionally, $\mathcal{F}_1, \ldots, \mathcal{F}_\ell$). We emphasize that this approach does not change the proofs that designers are required to write; once a protocol Π UC realizes $\mathcal{F}$, it is "safe" (from a provable security perspective) to arbitrarily compose it with other functionalities. Our approach does not require proving anything about $\Pi_{\text{Intended-use}}$ and Π_{Social}. Instead, the process of their specification helps designers see elements of their system that may require improvement or should be the site for in-depth policy discussions.

Guiding Principle for Step 1: all data comes from somewhere. It is clear that the extent to which our framework broadens the analytical frame depends on how far the modeling is extended in Step 1. If the increase in scope is minimal, then threats that are more removed from the system itself are unlikely to be captured. On the other hand, modeling the whole world is clearly infeasible. Thus, a balance must be struck. For the purposes of our framework, we believe a nice balance is achieved if the modeling captures the *source* of all data in the system. Namely, any data that is relevant to $\mathcal{F}$ must either be imported into the system by an actor in Π_{Social} or be sampled/created within $\Pi_{\text{Intended-use}}$.

Step 2: Articulating high-level, sociotechnical properties. With $\Pi_{\text{Intended-use}}$ and Π_{Social} specified, the framework affords the ability to discuss desirable sociotechnical properties of the system. That is, one can make statements of the form "when [human operator] takes [action], then [result]" or "if all other participants' software components are correctly implemented, then [human operator] cannot [action]." Moreover, the composition of $\mathcal{F}$, $\Pi_{\text{Intended-use}}$ and Π_{Social} allows us to interrogate the plausibility of these statements. For example, we might be able to show how one of these sociotechnical properties can be violated by a deviation of the software captured by $\Pi_{\text{Intended-use}}$ or human decisions within Π_{Social}. This can be done in two ways:

(1) *Top-down specification of properties.* The first approach to specifying sociotechnical properties is "top-down." This approach requires articulating the properties that the designers *hope* the composed system will achieve from first principles. These goals should be independent of the modeling and could be done *before* Step 1.

(2) *Bottom-up specification of properties.* Alternatively, the sociotechnical properties of the system can be developed by working with the modeling. For example, one could leverage the systems developed in the threat modeling literature (STRIDE, LINDDUN, etc…) to elicit a set of risks and harms, using our modeling as a modified data-flow diagram. This would include iterating through the various interfaces between ideal functionalities and untrusted code in $\Pi_{\text{Intended-use}}$ (or looking at the decisions made in Π_{Social}) and seeing the extent to which they are vulnerable to spoofing, tampering, etc… Ultimately, this process could still result in sociotechnical goals that could be read as top-level properties

[2] $\mathcal{F}_1, \ldots, \mathcal{F}_\ell$ are not ideal functionalities used to *realize* $\mathcal{F}$, but parallel system components. If $\mathcal{F}$ is realized in a hybrid model, those ideal functionalities would be suppressed at this level of abstraction.

the composed system should have (e.g., "even if [human operator] spoofs their identity…").

Clearly, the specificity of these statements, and the quantifiers used therein, require immense care. To illustrate this dynamic, we can look ahead to our case study in Section 3.2: the statements "The system can only reveal CSAM material to Apple's administrators" and "The system can only reveal images that an NCMEC administrator has designated as CSAM to Apple's administrators" are extremely similar, but have wildly different policy implications. Thus, crafting sociotechnical properties requires care tantamount to game-based definitions or ideal functionalities.

Step 3: Interrogating the sociotechnical properties. The final step of the framework is to study the extent to which the system achieves the desired sociotechnical properties. The exact form of the argument that cryptographers are expected to make about these properties is up to them, but the goal should be that the argument is convincing to even a skeptical reader. We note, however, that *we stop short of proposing that our framework facilitates formal proofs about these sociotechnical properties*. While it might be possible to construct proofs for these statements, there are non-trivial modeling choices that must first be overcome, which might come into conflict with our goal to keep the framework conceptually lightweight. We believe that this more formal approach might be valuable future work.

Next steps: ethical analysis. Having established the system's sociotechnical properties, it is necessary to decide if those properties are "good." Often, this will be a fraught choice and it will be appropriate to enlist interdisciplinary perspectives during deliberation. We note that our framework provides *inputs* to this ethical debate, but is not designed to support it directly. There are, however, emerging efforts to create ethical frameworks for reasoning about computer security that may be valuable for navigating this process [49].

2.2 Benefits and Limitations

Hopefully, it is immediately evident that our approach *broadens the designer's analytical frame*, as elements of a deployment beyond $\mathcal{F}$ must be brought into conversation with $\mathcal{F}$. We argue that our approach is naturally *enumerative,* in that system designers can derive $\Pi_{\text{Intended-use}}$ and Π_{Social} by iterating through all of $\mathcal{F}$'s interfaces and ensuring that the origins of the data provided to those interfaces are well specified. Additionally, the division between $\mathcal{F}$, $\Pi_{\text{Intended-use}}$ and Π_{Social} clearly delineates the ways in which *trust is allocated throughout the system*. Procedures within $\mathcal{F}$ are assured cryptographically, whereas the correctness of procedures within $\Pi_{\text{Intended-use}}$ and Π_{Social} are not assured whatsoever. Our approach works with already familiar UC language, making it natural to work with for cryptographers. In fact, because analysts are not required to produce UC proofs, it could even be possible to use our framework (with reasonable safety) without a deep understanding of UC.

Limitations. Setting the analysis boundaries may be contentious for some applications. While we see our approach as systematic and enumerative, it cannot guarantee that all threats have been considered. Additionally, our analysis treats all parts of $\Pi_{\text{Intended-use}}$ equally, although the software may be written by different parties and could have different, innate failure rates. Finally, our approach implicitly argues that software is "not the cryptographer's problem." In practice, however, cryptographic engineering is critical for secure systems.

One potentially powerful alternative approach would be to specify an ideal functionality $\mathcal{F}_{\text{intended-use}}$ that $\Pi_{\text{Intended-use}}$ implements and interrogate its properties. While this would allow us to make more formal statements about the composed system, it suffers from two drawbacks: (1) this task is quite challenging, as $\Pi_{\text{Intended-use}}$ is likely not designed with formal analysis in mind; and (2) we will still need to analyze the sociotechnical properties of $\mathcal{F}_{\text{intended-use}}$, which will likely be a very complex and messy ideal functionality. As a result, we choose to study this more manageable approach.

3 The Apple CSAM Scanning Case Study

In August 2021, Apple announced a set of changes to their services with the goal of promoting child safety [15]. Most prominent among these changes was the introduction of a mechanism for scanning the contents of end-to-end encrypted image backups for known instances of CSAM [14, 20, 21, 31, 60]. The cryptographic core of the CSAM scanning proposal was a *threshold, private set-intersection* protocol with incremental computation.[3]

3.1 Criticism of Apple's Proposal

Apple's system is an instantiation of a *client-side scanning* system, in which content moderation is performed on the client before encryption.[4] This approach to content moderation was discussed for several years before Apple's announcement, e.g., [37, 38, 61, 67, 74] as part of the ongoing debate about the appropriate role of encryption in society [6, 19, 55]. Shortly before Apple's announcement, Kulshrestha and Mayer described a similar system [50] and described many of the following criticisms in their limitations discussion. Abelson et al. [5] remind us, therefore, that many criticisms of Apple's proposal identify inherent properties of client-side scanning as a whole. We note that Apple's proposal sparked a tremendous uproar (e.g., [5, 26, 30, 32, 35, 39, 40, 45, 48, 52–54, 58, 59, 64, 68]), so we limit our summary to only the most salient criticisms.

[3] We focus on the non-fuzzy version of their protocol for simplicity.

[4] See Scheffler and Mayer [70] and Kamara et al. [45] to contextualize client-side scanning within content moderation.

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

(The following are parameters of the functionality: identity of the server Apple, identity of client devices $\mathcal{U}_1, \ldots, \mathcal{U}_m$, and a threshold $\mathcal{T}$.) On activation, for all clients $\mathcal{U}_i$, initialize $\mathsf{IsCounting}_{\mathcal{U}_i} = \mathsf{False}$ and $\mathsf{IsInited}_{\mathcal{U}_i} = \mathsf{False}$.

Initializing Clients: Upon input $(\mathsf{InitScan}, \{\mathcal{U}_1, \ldots \mathcal{U}_{m'}\}, \{X_{\mathcal{U}_1}, \ldots, X_{\mathcal{U}_{m'}}\})$ from Apple:

1. For $\mathcal{U}_i \in \{\mathcal{U}_1, \ldots \mathcal{U}_{m'}\}$, 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 a client $\mathcal{U}$:

1. Rejection sample two hash functions h_0, h_1 such that they could be used to construct a Cuckoo table with $X_{\mathcal{U}}$. If Apple is corrupt, allow $\mathcal{S}$ to supply h_0, h_1.
2. If $\mathsf{IsInited}_{\mathcal{U}} = \mathsf{True}$, set $\mathsf{IsCounting}_{\mathcal{U}} = \mathsf{True}$ and send $(\mathsf{ProcessInitComplete}, |X_{\mathcal{U}}|, h_0, h_1)$ to $\mathcal{U}$. Otherwise, send $(\mathsf{ProcessInitComplete}, \perp)$ 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 a client $\mathcal{U}$:

1. If $\mathsf{IsCounting}_{\mathcal{U}} \neq \mathsf{True}$, return. If this is the first image received from $\mathcal{U}$, initialize variable $\mathsf{counter}_{\mathcal{U}}$, list $\mathsf{img\_id\_list}_{\mathcal{U}}$, and list $\mathsf{keys}_{\mathcal{U}}$.
2. If $\mathsf{img\_hash} \notin X_{\mathcal{U}}$ or $\mathsf{valid} = \mathsf{false}$, send $(\mathsf{ScanImageComplete}, \mathcal{U}, \mathsf{false})$ to Apple. `//valid models malformed messages`
3. If $\mathsf{img\_hash} \in X_{\mathcal{U}}$, increment $\mathsf{counter}_{\mathcal{U}}$, append $\mathsf{img\_id}$ to $\mathsf{img\_id\_list}_{\mathcal{U}}$ 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 $(\mathsf{ThresholdMet}, \mathcal{U}, \mathsf{img\_id\_list}_{\mathcal{U}}, \mathsf{keys}_{\mathcal{U}})$ to Apple.

Figure 2: Apple's CSAM scanning proposal. We have updated the notation and modeling (see Section A for details).

Criticism 1: The system can be easily re-targeted. No technical measure limited the scanning system to CSAM and therefore other types of content, e.g., politically disruptive content, could be targeted. Abelson et al. [5] call this *Abuse by Authorized Parties*, while the EFF [52] and Green [39] call this *Mission Creep*. While Apple has claimed that it would resist pressure to broaden the scope of its system [16], Apple has already made other concessions to governments [44, 56, 57], indicating that sustained resistance might be difficult.

Criticism 2: Malicious operators or hackers could abuse the system. Even if Apple were able to resist external pressure to broaden the scope of the system, a malicious operator could include non-CSAM content into the hashset. Abelson et al. [5] call this *Abuse by Unauthorized Parties* and Green [39] calls this *Unauthorized Surveillance*. Apple's system also provides no mechanism to ensure that each client device receives the same hashset [47]. Apple suggested that some kind of auditing measure could mitigate the risk of these attacks [13, 16] by making attacks *post-hoc* detectable, but the details of the auditing system were never specified.[5]

Criticism 3: The system is not sufficiently robust. Even if the integrity of the hashset is maintained, critics were concerned about the robustness of the system. Put succinctly by Abelson et al. [5], *"CSS [Client Side Scanning] Is Less Efficacious in Adversarial Environments."* This criticism spans two concrete concerns: (a) there is evidence that existing perceptual hash functions, including the one designed by Apple, are not robust [29, 51, 62, 72], which could lead to false positives and false negatives, and (b) malicious clients could encrypt or modify images to evade detection [50].

[5] Scheffler, Kulshrestha, and Mayer [69] study the problem of adding auditability into Apple's proposal.

3.2 Modeling Apple's Initial CSAM Scanning Proposal

We now illustrate applying our framework step by step.

Step 0: Defining the ideal functionality $\mathcal{F}_{\text{Apple-CSAM-Scan}}$. The initial formal description of the protocol provided by Apple is very similar to a classic two-party private set intersection protocol, with the added complexity of (1) only releasing the output when the set intersection exceeds some fixed threshold, and (2) switching the semantics of the private set intersection to be more "key-value," such that the intersection is performed on a set of keys, but the data associated with each of those keys constitutes the actual output. Working with this initial formalism within our framework poses several challenges. First, the description of the ideal functionality treats the user input as a *batch*, whereas input is actually provided to the clients incrementally. Second, Apple's formalism relies on parallel composition to support multiple clients. While this latter choice is technically sound, it makes exposition in this paper more difficult.

We provide an updated model of Apple's ideal functionality, which we call $\mathcal{F}_{\text{Apple-CSAM-Scan}}$, in Figure 2. We modify Apple's initial formalism by addressing the two difficulties outlined above. Specifically, we have clients provide their inputs incrementally rather than as a batch, and all clients interact with the *same* ideal functionality, rather than relying on composition. Additionally, we add some leakage that is

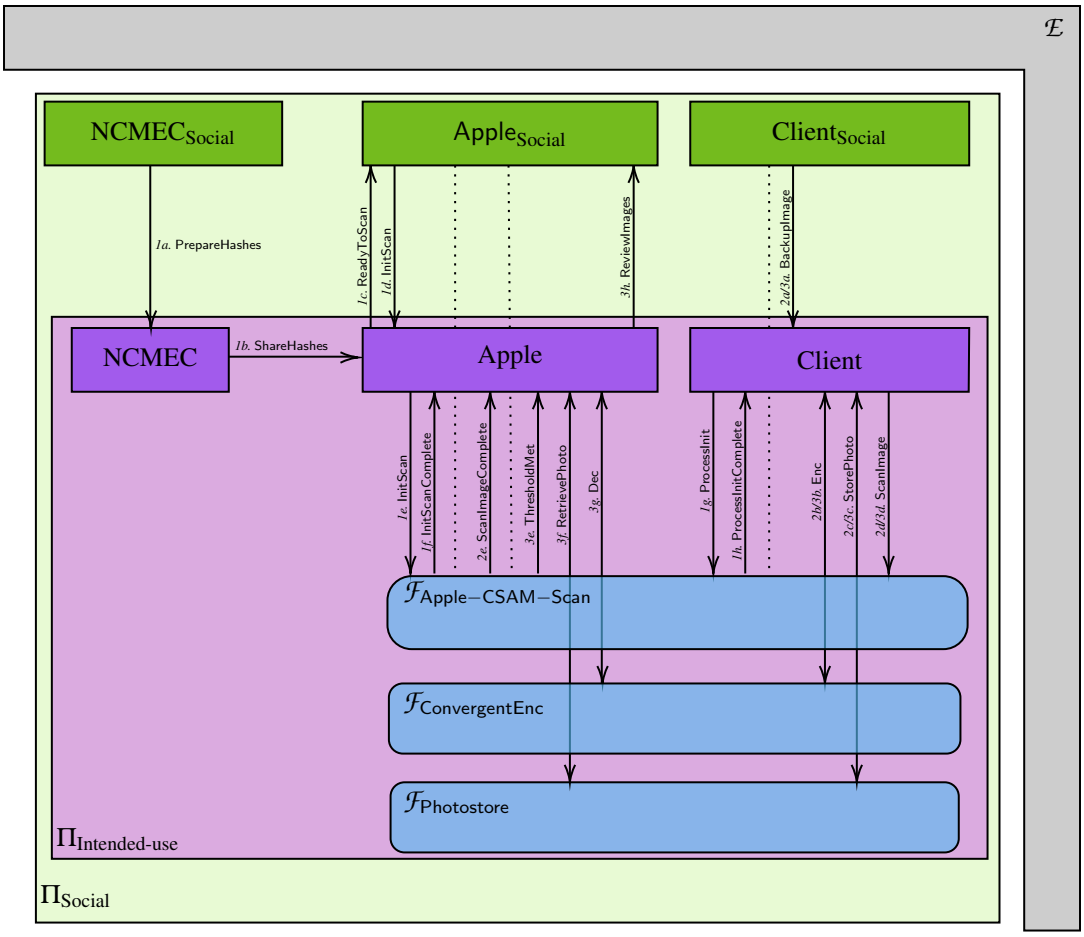

Figure 3: Overview of Apple's proposed protocol in our framework. Numbering indicates flows described in text below.

missing from Apple's initial modeling (namely, the hash functions used in their protocol that are sampled in an input-dependent way). We provide Apple's initial functionality and discuss the small protocol changes required to make their protocol realize $\mathcal{F}_{\text{Apple-CSAM-Scan}}$ in Section A.

Step 1: Extending the modeling. Following the guiding principle for Step 1 we laid out in Section 2.1, modeling Apple's scanning proposal within our framework requires "explaining" all of the data consumed by $\mathcal{F}_{\text{Apple-CSAM-Scan}}$. In this case, these inputs include the hashset X, the client-supplied images and cryptographic keys that will be input to $\mathcal{F}_{\text{Apple-CSAM-Scan}}$. This requires introducing an additional party and additional ideal functionalities. First, we introduce the National Center for Missing and Exploited Children (NCMEC) as the originator of the hashset X, with a component in both $\Pi_{\text{Intended-use}}$ and Π_{Social}. Next, we surface the *purpose* of running a private set-intersection protocol by introducing two additional simple ideal functionalities: $\mathcal{F}_{\text{PhotoStore}}$ and $\mathcal{F}_{\text{ConvergentEnc}}$. The former captures a system that stores ciphertexts (i.e., an arbitrary blob store) and the latter captures *convergent encryption* [22, 28, 46], a deterministic encryption mode Apple uses to reduce server-side storage [4, 17].[6] In convergent encryption, a large message (i.e., an image) is encrypted under a key derived from the message itself (i.e., $c = \mathsf{Enc}(\mathsf{KDF}(m), m)$) and then the derived key $\mathsf{KDF}(m)$ is encrypted under some user-selected key k. This construction is clearly not IND–CPA and, therefore, the blob store can reuse a single ciphertext for multiple clients. The practice of using convergent encryption to minimize storage is widespread, despite its clear security limitations. Descriptions of these ideal functionalities can be found in the full version of this paper. We are unaware of any ideal functionalities for convergent encryption [22, 46], so we write our own. We also implicitly use $\mathcal{F}_{\text{SMT}}$ [25] when sending messages.

[6] The iCloud Security Overview described this by saying that "The raw byte checksum of the photo or video" is stored under "standard encryption" rather than "end-to-end encryption," i.e., the hash of the image is stored after being encrypted under a key controlled by Apple servers [17]. In Apple's platform security guide, the documentation explicitly names this practice as *convergent encryption* [4].

An overview of rendering Apple's proposal into our framework can be seen in Figure 3. $\Pi_{\text{Intended-use}}$ and Π_{Social} can be found in Figure 4 and Figure 5 respectively. There are three primary information flows that we identify, each of which is labeled with a different number in Figure 3: (1) A flow in which the target images are selected, processed, and initialized into the system; (2) A flow describing a client backing up one of their images; and (3) A flow describing a backup that results in the threshold being met, including the resulting decryption.

Process. In order to determine the contents of the protocols $\Pi_{\text{Intended-use}}$ and Π_{Social}, we look through each interface of Figure 2 and note the data that is sent as input for each interface. As described above, it is clear that the hashes for which the functionality is scanning must be made concrete and the meaning of the inputs for each client image submission must be made concrete. Having introduced the additional parties and functionalities described above, we make sure that there is a clear source for each of these pieces of data and it has a path from its source to $\mathcal{F}_{\text{Apple-CSAM-Scan}}$. We note that all modeling in this work assumes *static* corruptions for simplicity.

Flow 1: NCMEC selects the images and Apple initializes scanning. The images in Apple's proposed system originate with NCMEC.[7] Thus, the first step (1a) is to have an administrator at NCMEC select a set of images which should be included in the scanning mechanism. The source of these images—and the mechanisms by which CSAM images are sifted from "other" images—is unspecified; for both of these choices there is no clear normative claim we can make about "honest" behavior or a verification function we could write to ensure that the NCMEC admin has behaved "honestly". As such, this process must happen in Π_{Social}. The NCMEC admin then invokes their software to hash each of the selected images with a perceptual hash function and sends the resulting hashes to Apple's systems (1b). Apple's systems inform an Apple admin that scanning can commence (1c). When the Apple admin is ready, they can initialize the system by calling start (1d) and sending the perceptual hashes to the ideal functionality $\mathcal{F}_{\text{Apple-CSAM-Scan}}$ (1e/f). Finally, client devices check $\mathcal{F}_{\text{Apple-CSAM-Scan}}$ for initialization (1g/h).

Flow 2: Client backing-up image. Clients might receive images from *anywhere* that they then choose to backup. For example, images might be downloaded from the internet, received from another client over an app, or might be a new image captured by the client device itself. As such, the origin of these images is not specified and is just input to the client from the environment in Π_{Social}. The second flow begins with a client choosing to backup some image (2a); notice that the choice to save an image to iCloud is user-driven and there is no assumed correct choice, meaning this choice needs to be situated in Π_{Social}. Next, the client device handles the processing of this image within $\Pi_{\text{Intended-use}}$, by encrypting the image with $\mathcal{F}_{\mathsf{ConvergentEnc}}$ (2b), generating a new cryptographic key, and sending the resulting ciphertext to $\mathcal{F}_{\mathsf{PhotoStore}}$ (2c). The image identifier, the perceptual hash of the image, and the key generated when interacting with $\mathcal{F}_{\mathsf{ConvergentEnc}}$ are then sent to $\mathcal{F}_{\text{Apple-CSAM-Scan}}$ (2d). Finally, $\mathcal{F}_{\text{Apple-CSAM-Scan}}$ notifies Apple's system that the client has submitted a voucher and informs Apple if the submitted voucher is a match.

Flow 3: Client backing-up image that triggers threshold. The process for client uploads that exceed the scanning threshold begins exactly as described in Flow 2 (see 3a-3d). However, when the threshold is met, $\mathcal{F}_{\text{Apple-CSAM-Scan}}$ sends the data associated with all matching submissions to Apple (3e). Using this information, Apple can retrieve the relevant ciphertexts from $\mathcal{F}_{\mathsf{PhotoStore}}$ (3f) and then decrypt each image (3g). These final two steps (3f and 3g) are called for each image in the intersection. The resulting image decryptions are then sent to Apple for manual review (3h). Again, there is no normative process that we can describe at this point. Presumably, an Apple administrator will be responsible for looking at the images and determining if they are indeed CSAM, a process for which there is no algorithm.

3.3 Examining Sociotechnical Properties

Having rendered Apple's proposal into our framework, we now interrogate its sociotechnical properties. We present the results of conducting Step 2 and Step 3 together, although we performed these steps sequentially. We begin with the "top-down" approach, leaning on sociotechnical goals that Apple set in a document release for public consumption. We then turn our attention to the "bottom-up" approach and show how this analytical approach can surface the concerns listed in Section 3.1.

Top-down properties. Within their non-technical overview [14], Apple provided a set of informally phrased guarantees that their system was supposed to provide. These represent the most explicit enumeration of the sociotechnical goals of Apple's proposal, although their authors likely did not anticipate them being used as we do below. We interrogate them in the order listed.

(1) *"Apple does not learn anything about images that do not match the known CSAM database."* Within our model, we see that the only way for Apple to learn information about the content of image files that users back-up is when key information is released to Apple via $\mathcal{F}_{\text{Apple-CSAM-Scan}}$. However, the existing phrasing has two ambiguous terms: "match" and "CSAM database." A reasonable reading of the term "match" might include an exact match, but in fact this is trying to capture the semantic matching properties provided by the perceptual hash function. As for the "CSAM Database,"

[7] Apple later clarified that it would only use images held by at least two child protection organizations.

$\Pi_{\text{Intended-use}}$ for $\mathcal{F}_{\text{Apple-CSAM-Scan}}$

(The protocol is parameterized by a perceptual hash function PHash*.)*

NCMEC: When NCMEC is invoked with the message $(\mathsf{PrepareHashes}, I = \{\mathsf{img}_1, \mathsf{img}_2, \ldots\})$ from $\text{NCMEC}_{\text{Social}}$:

1. Set $X = \{\}$, then for $\mathsf{img}_i \in I$, compute $\mathsf{img\_hash}_i = \mathsf{PHash}(\mathsf{img}_i)$ and set $X = X \cup \{\mathsf{img\_hash}_i\}$.
2. Send the message $(\mathsf{ShareHashes}, X = \{x_1, \ldots, x_n\})$ to Apple via an instance of $\mathcal{F}_{\mathsf{SMT}}$.

Apple: When Apple receives a message $(\mathsf{ShareHashes}, X = \{\mathsf{img\_hash}_1, \ldots, \mathsf{img\_hash}_n\})$ from NCMEC via $\mathcal{F}_{\mathsf{SMT}}$:

1. Store X and send $(\mathsf{ReadyToScan})$ to $\text{Apple}_{\text{Social}}$.

Apple: When Apple is invoked with the message $(\mathsf{InitScan})$ from $\text{Apple}_{\text{Social}}$:

1. Retrieve X. Let $\{\mathcal{U}_1, \ldots, \mathcal{U}_m\}$ be the set of all client devices. Send $(\mathsf{InitScan}, \{\mathcal{U}_1, \ldots, \mathcal{U}_m\}, \{X, \ldots, X\})$ to $\mathcal{F}_{\text{Apple-CSAM-Scan}}$. Receive $(\mathsf{InitScanComplete})$ in response and return.

Client: Until a message $(\mathsf{ProcessInitComplete}, |X_{\mathcal{U}}|, h_0, h_1)$ has been received from $\mathcal{F}_{\text{Apple-CSAM-Scan}}$, whenever the client device $\mathcal{U}$ is invoked it sends $(\mathsf{ProcessInit})$ to $\mathcal{F}_{\text{Apple-CSAM-Scan}}$ and processes the result.

Client: When a client device $\mathcal{U}$ is invoked with the message $(\mathsf{BackupImage}, \mathsf{img})$ from $\text{Client}_{\text{Social}}$:

1. Sample a unique random value $\mathsf{img\_id}$
2. Send $(\mathsf{Enc}, \mathsf{img\_id}, \mathsf{img})$ to $\mathcal{F}_{\mathsf{ConvergentEnc}}$ and receive the output $(\mathsf{EncComplete}, \mathsf{img\_id}, \mathsf{ctx}, \mathsf{k})$.
3. Send $(\mathsf{StorePhoto}, \mathcal{U}, \mathsf{img\_id}, \mathsf{ctx})$ to $\mathcal{F}_{\mathsf{PhotoStore}}$ and receive the output $(\mathsf{StorePhotoComplete})$ in response.
4. Compute $\mathsf{img\_hash} = \mathsf{PHash}(\mathsf{img})$ and send $(\mathsf{ScanImage}, \mathsf{img\_id}, \mathsf{img\_hash}, \mathsf{k})$ to $\mathcal{F}_{\text{Apple-CSAM-Scan}}$.

Apple: When Apple receives a message $(\mathsf{ScanImageComplete}, \mathcal{U}, \mathsf{img\_id}, \mathsf{match} \in \{\mathsf{true}, \mathsf{false}\})$ from $\mathcal{F}_{\text{Apple-CSAM-Scan}}$, return control flow to the environment.

Apple: When Apple receives a message $(\mathsf{ThresholdMet}, \mathcal{U}, \mathsf{img\_id}, \mathsf{img\_id\_list}_{\mathcal{U}}, \mathsf{keys}_{\mathcal{U}})$ from $\mathcal{F}_{\text{Apple-CSAM-Scan}}$:

1. Initialize the empty set $\mathsf{decrypted\_photos}_{\mathcal{U}}$.
2. For $i \in |\mathsf{img\_id\_list}_{\mathcal{U}}|$, send $(\mathsf{RetrievePhoto}, \mathcal{U}, \mathsf{img\_id}_i)$ to $\mathcal{F}_{\mathsf{PhotoStore}}$. If $(\mathsf{RetrievePhotoComplete}, \mathsf{img\_id}_i, \perp)$ is received in response, continue. Otherwise, if $(\mathsf{RetrievePhotoComplete}, \mathsf{img\_id}_i, \mathsf{ctx}_i)$ is received:
   (a) Send $(\mathsf{Dec}, \mathsf{img\_id}_i, \mathsf{ctx}_i, \mathsf{k}_i)$ to $\mathcal{F}_{\mathsf{ConvergentEnc}}$. If $(\mathsf{DecComplete}, \mathsf{img\_id}_i, \perp)$ is received in response, continue. Otherwise, if $(\mathsf{DecComplete}, \mathsf{img\_id}_i, \mathsf{img}_i)$ is received in response, add img_i to $\mathsf{decrypted\_photos}_{\mathcal{U}}$.
3. Send $(\mathsf{ReviewImages}, \mathcal{U}, \mathsf{decrypted\_photos})$ to $\text{Apple}_{\text{Social}}$.

Figure 4: $\Pi_{\text{Intended-use}}$ for $\mathcal{F}_{\text{Apple-CSAM-Scan}}$, specifying how non-cryptographic system components *should* operate.

this term could reasonably either refer to the set of images selected and passed through the perceptual hash function by NCMEC or could refer to the set of perceptual hashes Apple sends to the ideal functionality. Interrogating this property within the context of our model lends itself to seeing these differences because this property is clearly discussing the input-output behavior of the ideal functionality. As a result, an analyst might rephrase this property to be "Apple does not learn anything about images that do *not have the same perceptual hash as those selected by Apple for scanning*."

(2) *"Apple can't access metadata or visual derivatives for matched CSAM images until a threshold of matches is exceeded for an iCloud Photos account."* There are two problems with this property: the first is largely an issue of semantics/clarity, while the second undermines the veracity of this claim entirely. First, there is ambiguity in the terms "metadata or visual derivatives." Specifically, when a match is identified, Apple's software is alerted as to the existence of a new match; while this is metadata as technically understood, it might not be metadata as conversationally understood, leading to potential misunderstanding. Moreover, Apple's iCloud data security overview explicitly lists instances of metadata that are accessible to Apple under all circumstances—although this is unlikely to be the specific metadata about which this claim is made.

Much more notably, this claim becomes false *once the composition of the full system becomes apparent*, i.e., once the analysis includes the manner in which the images are stored and the fact that there are many clients in the system. Namely, the interaction between storing the images with convergent encryption, an encryption mode that is explicitly not IND–CPA, and a threshold PSI scheme that is designed to detect semantic matches undermines this claim.

Recall that convergent encryption allows Apple to identify when two users store bit-for-bit identical images. This will directly reveal instances where a user uploads an image that exactly matches one of the instances of known CSAM for which the system is scanning, but does not directly leak when a user uploads a semantically equivalent image (e.g., changing a single pixel value or rotating the image); it is possible for someone to identify the plaintext corresponding to a convergent encryption, but there is likely sufficient entropy in most images to make this challenging in practice. The threshold PSI system hides when different users share inputs,

Π_{Social} for $\mathcal{F}_{\text{Apple-CSAM-Scan}}$

(The protocol is parameterized by a threshold $\mathcal{T}$.)

$\text{NCMEC}_{\text{Social}}$: When $\text{NCMEC}_{\text{Social}}$ is invoked with $I = \{\mathsf{img}_1, \mathsf{img}_2, \ldots\}$ from Env:

1. Select a subset of images $I' \subseteq I$ that are CSAM, then invoke $\Pi_{\text{Intended-use}}$ with the message $(\mathsf{PrepareHashes}, I')$.

$\text{Apple}_{\text{Social}}$: When $\text{Apple}_{\text{Social}}$ receives $(\mathsf{ReadyToScan})$ from Apple, return control flow to the environment.

$\text{Apple}_{\text{Social}}$: When $\text{Apple}_{\text{Social}}$ is invoked with $(\mathsf{InitScan})$ from Env:

1. If $\mathsf{AppleAdmin}$ is ready to initialize scanning, invoke $\Pi_{\text{intended-use}}$ with the message $(\mathsf{InitScan})$.

$\text{Client}_{\text{Social}}$: When a client Client is invoked with image img by Env :

1. Determine if img is to be backed up. If so, invoke $\Pi_{\text{Intended-use}}$ with the message $(\mathsf{BackupImage}, \mathsf{img})$. Otherwise, return control flow to the environment.

$\text{Apple}_{\text{Social}}$: When $\text{Apple}_{\text{Social}}$ receives $(\mathsf{ReviewImages}, \mathcal{U}, I = \{\mathsf{img}_1, \mathsf{img}_2, \ldots\})$ from Apple:

1. Review the images I and determine if they are CSAM.
2. If any images are not CSAM, report the existence of a design flaw to all other parties.
3. If $|I| < \mathcal{T}$, report that either there is a design flaw or $\mathcal{U}$ is corrupt to all other parties.
4. Send the images to NCMEC or law enforcement as necessary to comply with existing laws.
5. Output I to Env.

Figure 5: Π_{Social} for $\mathcal{F}_{\text{Apple-CSAM-Scan}}$, describing parts of the system that will not be enshrined in software.

unless they both exceed the threshold. When combined, however, Apple could identify that a particular ciphertext is a semantic match for known CSAM if one client passes the threshold and convergent encryption will then allow Apple to identify all other users storing a bit-for-bit match with this image.

To make this concrete, consider the following scenario: Client A sends Client B an image that is a semantic match—but not a bit-for-bit match—with one of the images selected for scanning. When they each upload this image to the system, Client B passes the "threshold of matches," making this image accessible to Apple. The immediate implication is that Apple will also be able to access a "visual derivative" (i.e., the plaintext of the image itself) of one of Client A's photos, even though their iCloud account has not exceeded the match threshold.

(3) *"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."* This is perhaps the most contentious sociotechnical property provided in [14]. Without a quantitative way to evaluate "extremely low," this property likely is not analytically helpful. Leblanc-Albarel and Preneel [51] recently started studying the perceptual hash functions used in client-side scanning and found relatively higher rates of hash collisions, which may cast doubt on this sociotechnical property.

(4) *"Users can't access or view the database of known CSAM images."* This is a clearly specified sociotechnical property that appears to hold in the system. Specifically, the only leakage about the database of known CSAM images (both the images curated by NCMEC and the perceptual hashes pre-selected by Apple) is the size of the set and the hash functions h_0, h_1. While this latter leakage might allow an attacker to gain partial information about the images, they cannot facilitate "access[ing]" or "view[ing]" the images.

(5) *"Users can't identify which images were flagged as CSAM by the system."* Within the scope of our modeling, there is no way for a user to determine if their image back-up is a match or non-match. We note, however, that the system could be used to flag non-CSAM images, which could motivate slightly rewording this property. Moreover, there are aspects of the described system that are under-specified which make us unable to properly verify this property—imagine the flagged images were later introduced as court evidence.

Bottom-up criticisms. Next, we look to our modeling to see if it successfully allows us to observe the criticisms discussed in Section 3.1 in a bottom-up way. Ideally, these criticisms should follow naturally from asking the questions *What happens if $\Pi_{\text{Intended-use}}$ is not followed?* and *Are there choices within Π_{Social} that could be problematic?* In other words, does the fact that particular parts of the protocol are contained with $\Pi_{\text{Intended-use}}$ and Π_{Social}, rather than the ideal functionality itself, indicate that there is an opportunity for abuse?

Flow 1 and Criticism 1: First, we can see within Flow 1 that the entity responsible for curating the hashset can include arbitrary images, either by explicitly selecting images that others would not see as fitting a normative understanding of the purpose of the system (i.e., within Π_{Social}) or because of software error (i.e., within $\Pi_{\text{Intended-use}}$). Put another way, we can ask the question: *What happens when NCMEC chooses to include images that would not normally be understood to be CSAM in its hashset?* As a result, we see that the selection of

images is a *social* process and is, thus, vulnerable to typical pathways of social disruption, i.e., government pressure and shifting norms. This serves to highlight Criticism 1, which could be rephrased as a sociotechnical property of the system as: "the only protection against including non-CSAM images in the hashset is the judgment of the NCMEC administrator."

Flow 1 and Criticism 2: If we examine the second part of Flow 1, we observe that the server's software is expected to submit exactly the hashes supplied by NCMEC, but there is no mechanism to guarantee that this happens. Alternatively, we can see this by observing NCMEC provides no inputs to $\mathcal{F}_{\text{Apple-CSAM-Scan}}$. To frame this as a question, we could ask: *What happens when Apple's software selects images not supplied by NCMEC to the ideal functionality?* This serves to highlight Criticism 2, i.e., "Apple's management software can (untraceably) change the contents of the hashset." It is worth noting that the specifics of the user interface for Apple's management software (i.e., if there is an exposed choice to select photos) could easily shift this from a software concern to a social concern.

Flow 2/3 and Criticism 3: Finally, within Flow 2 and Flow 3, it becomes very clear that the client must voluntarily tie both the perceptual hash and the cryptographic key they submit to the ideal functionality to the images that they backup. Specifically, we could ask the question: *What happens when the client device does not follow* $\Pi_{\textit{Intended-use}}$ *when selecting inputs to* $\mathcal{F}_{\text{PhotoStore}}$, $\mathcal{F}_{\text{ConvergentEnc}}$, *and* $\mathcal{F}_{\textit{Apple-CSAM-Scan}}$? This shows that it is trivial for a malicious client to evade detection, as noted in Criticism 3.

3.4 Takeaways from analyzing Apple's proposal

Applying our framework to Apple's protocol proves to be a valuable process that systematically uncovers the known criticisms of Apple's protocol. More importantly, our process highlighted a weakness of Apple's proposal that has not previously been discussed in the academic or policy discussion of the protocol (to our knowledge) despite multiple years of scrutiny: the leakage that results from running a scanning protocol and convergent encryption within the same system.[8] This leakage is highly non-trivial as it, from a technical perspective, undermines the main aspirational goal that Apple's system will not introduce any privacy leakage to individuals who are not holding multiple instances of CSAM. Moreover, this particular privacy leakage is *actionable*. Specifically, if Apple were to uncover that a particular convergent encryption ciphertext corresponds to a known instance of CSAM via their scanning system, they seem to have a legal responsibility to *report* the event to the relevant authorities (see 18 U.S. Code §2258A [1]).[9] This report would likely include the existence of the ciphertext, the details of the individual whose input to the PSI protocol resulted in the decryption of the ciphertext, and the information of any other user who interacted with the ciphertext.[10] If Apple had deployed its proposed system, we very likely would have seen legal repercussions from this privacy leakage that the community had never discussed—further highlighting the need for broadening the scope of analysis.

Limitations. By treating the perceptual hash function as a property-less black-box, our framework fails to highlight sociotechnical properties associated with its false positive and false negative rates [41, 51, 63]. Our framework lets us contextualize cryptographic properties that have proven, idealized properties, but these hash functions rely on machine learning techniques that lack any provable properties. As such, our framework treats them as "worst case" functions with *no* properties. Clearly, developing modifications of our framework that help differentiate between different components within $\Pi_{\text{Intended-use}}$ is intriguing future work.

4 Discussion

We take a first step towards building a cryptography-native approach to illustrating sociotechnical properties of cryptographic systems. We see four significant benefits to making the use of our framework a communal *expectation*:

(1) Being very transparent helps ensure that there is never the appearance that protocol designers are trying to "hide" anything. Especially when there are real trade-offs associated with deployment, being clear about the costs and benefits is important.

(2) Agreeing upon a specific process helps ensure protocol designers do not take short cuts when thinking through their proposals. This is similar to the way *writing* a formal security proof can help highlight details missed during the protocol design process—in addition to being a compelling artifact for external reviewers of the protocol;

(3) Taking responsibility for establishing sociotechnical properties helps ensure that the marketing of cryptographic deployments stays accurate. If cryptographers do not do this work, someone less qualified *will* do it—and will get it wrong.

[8] There is a non-public podcast with the title "Apple iCloud Encryption, CSAM Scanning and Convergent Encryption", which might potentially touch on this topic, extrapolating from the title alone. Given the non-public nature of the piece, none of the authors have reviewed it. Even if this piece does discuss the interplay between the PSI protocol and convergent encryption, this has not made its way into the academic conversation or policy discussion, to our knowledge.

[9] This is a best-effort reading of the law; the authors are not lawyers.

[10] 18 U.S. Code §2258A states that the report should contain "Information relating to when and how a customer or subscriber of a provider uploaded, transmitted, or received content relating to the report or when and how content relating to the report was reported to, or discovered by the provider, including a date and time stamp"

(4) When proposals prove to be controversial, the artifact created by conducting this analysis can serve as a starting point for debating ethics. Importantly, this helps differentiate between disagreement about the sociotechnical properties a system has and whether those properties are "good enough" for deployment.

Post-hoc vs. proactive use of our framework. In the main body of our paper, we study applying our framework to an existing proposal in a post-hoc way, when a set of sociotechnical properties (i.e., critiques) were already well known. While this gives us confidence that our framework does not miss important properties, our prior exposure to these known sociotechnical properties might have made them easier to identify. On the other hand, the fact that our framework helped us identify a new property of Apple's proposal, despite years of community attention, indicates our framework represents progress.

In the full version of this paper, we explore an alternative approach: using our framework proactively within the protocol design process. Specifically, we flip the order of the steps in the following way: (1) enumerate the sociotechnical properties that the system should have; (2) design an ideal functionality and associated, realizing protocol; (3) extend the modeling to include $\Pi_{\text{Intended-use}}$ and Π_{Social}; and (4) interrogate the extent to which the design meets the desired sociotechnical goals. We study sociotechnical properties like *"Apple cannot scan client's devices for any perceptual hashes not designated as CSAM by NCMEC," "It is possible for any member of the public to verify that the system is scanning for the same perceptual hashes for all clients,"* and *"Claims that a client backed up CSAM to the photostore should be publicly verifiable—implying that clients cannot be framed."* We imagine this proactive approach might be particularly valuable when working with interdisciplinary teams in order to design cryptographic systems in response to community needs.

Towards ethics and cross-disciplinary conversations. As discussed in Section 1, we see establishing sociotechnical properties and arguing ethics as two conceptually distinct tasks, the first of which is a precondition for the second. Our work is aimed at supporting the former process, at least in part because we believe that arguing the ethics of deployment cannot be the responsibility of cryptographers alone. Input from experts with other disciplinary expertise (e.g., lawyers, policymakers, ethicists, etc.) is critical.

Eventually, we hope the community comes to consensus on a process by which sociotechnical properties of cryptosystems are established. It is important that this process produces artifacts that are valuable for this interdisciplinary conversation and keep it technically accurate. This type of artifact—one that can cross disciplinary boundaries without losing fidelity—is called a "boundary object" in sociology [71]. We observe that the properties produced by our framework have the potential to be high-quality boundary objects, as they are expressed in natural language and are exactly the types of properties that would be interesting to lawyers or policymakers. In principle, this could also provide an avenue for these communities to provide input, feedback, or set goals for cryptographic systems.

5 Conclusion and Future Work

In this work, we identified a gap in the existing expectations for analyzing cryptographic systems which allowed documentation to silently omit important properties of an eventual deployment—not out of malice, but simply because these properties were out of scope. This observation motivates the need for a new generation of analysis tools that expand the scope of analysis that cryptographers regularly conduct while also feeling native to the cryptographers using them. We re-examined UC security through the lens of the threat modeling literature and showed that it could be repurposed to meet this need. Our case study revealed that our framework is promising, as it both systematically identified known sociotechnical properties and helped us identify a property of the system that had been overlooked.

We hope that our work illustrates the need for more community attention on this issue. The framework we propose is intended to be a *first step* and significantly more community effort will be required before we reach consensus as to the right analysis tool.

We see several next research problems that are particularly important: (1) our work focused on UC security, but it is clear we will need analogous approaches that naturally interoperate with game-based definitions; (2) we stopped short of trying to *prove* that systems have sociotechnical properties. While designing a framework that would enable these proofs is challenging, it does not appear to be impossible. Future work could also consider using formal analysis tools from the programming literature to automate some of this work; and (3) rigorously analyzing the usability of our system (or similar future proposals) would be valuable.

Acknowledgments

All three authors are supported by DARPA under Agreement No. HR00112020021. The second author is also supported by the National Science Foundation under Grants No. 1718135, 1915763, and 2209194. Much of this work was completed while the third author was at Boston University. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

The authors would like to thank the many people who read early drafts of the work and provided invaluable feedback, including Zoe Ruha Bell, Shaanan Cohney, Rachel

Cummings, Shlomi Hod, Palak Jain, Ryan Little, Priyanka Nanayakkara, Daniel Roche, Jayshree Sarathy, and Kris Shrishak. We would like to especially thank Mayank Varia for supporting us through multiple resubmissions of this work and Daniel Votipka for encouraging us to properly contextualize this work within the threat modeling literature. Finally, we would like to thank the organizers of the Cryptographic Applications Workshop 2024, Miro Haller and Matilda Backendal, for providing a venue in which we could present an early version of this work and gather important feedback.

Ethical Considerations

Our work is directly inspired by the need for better tools that would facilitate studying the ethics of cryptographic systems. This is because cryptographic systems have the ability to significantly rearrange power, as Phil Rogaway articulated in his seminal work. As such, it is incumbent on researchers to clearly articulate the ethics of their work.

This specific project is not typical cryptographic research, in that our first-order priority is not the design of a new cryptographic system or demonstrating an attack on an existing cryptosystem. This significantly reduces the risk associated with conducting this research. Nevertheless, we attempt to clearly articulate the stakeholders, risks, and rationale behind our decision to carry out this research.

We identify the following stakeholders: (1) Cryptographers; and (2) Apple and the designers of Apple's CSAM scanning system

Notably, Apple's system was never deployed and there is no plan (of which we know) to deploy it. As such, end-users are not a stakeholder in this analysis.

We do note, however, that our analysis did identify a property of a proposed system that was previously unknown. This could change the eventual calculus when it comes to deploying a system based on Apple's initial proposal. We have not conducted a vulnerability disclosure because the system was never deployed, meaning our results do not put any user data at risk.

In the full version of our work, we do develop an alternative version of Apple's protocol with arguably stronger sociotechnical properties. The described protocol is a thought exercise, and not a deployment proposal. There is a risk, however, that describing such a system changes the probability that a system like Apple's is deployed. Our research team was internally divided on whether such a system would be ethical to deploy. However, we believe that the best way to work through the ethics of that decision is publicly with input from the whole community. As such, publication is an appropriate decision.

Open Science

We created no source code, data, or scripts as part of this research. The output artifacts are the analyses we conducted within the paper itself. Due to space constraints, not all analyses conducted as part of this work are contained within this version of the work.

References

[1] U.S. Code Title 18 - Crimes and Criminal Procedures. Accessed on 1 Aug 2025 at https://www.law.cornell.edu/uscode/text/18/2258A.

[2] Crypto for the People, Invited talk at Crypto 2020 by Seny Kamara, August 2020.

[3] USENIX Enigma 2022 - Crypto for the People (part 2), April 2022.

[4] Apple platform security. https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf, May 2024.

[5] 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, and Carmela Troncoso. Bugs in our pockets: The risks of client-side scanning, 2021.

[6] Harold Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield "Whit" Diffie, John Gilmore, Matthew Green, Susan Landau, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Michael A. Specter, and Daniel J. Weitzner. Keys under doormats. *Commun. ACM*, 58(10):24–26, sep 2015.

[7] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 5–17. ACM Press, October 2015.

[8] Martin R. Albrecht, Simone Colombo, Benjamin Dowling, and Rikke Bjerg Jensen. At-compromise security: The case for alert blindness. In *Advances in Cryptology – EUROCRYPT 2026: 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10–14, 2026, Proceedings, Part II*, page 191–221, Berlin, Heidelberg, 2026. Springer-Verlag.

[9] Martin R. Albrecht, Raphael Eikenberg, and Kenneth G. Paterson. Breaking bridgefy, again: Adopting libsignal is not enough. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 269–286. USENIX Association, August 2022.

[10] Martin R. Albrecht, Miro Haller, Lenka Mareková, and Kenneth G. Paterson. Caveat implementor! Key recovery attacks on MEGA. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 190–218. Springer, Cham, April 2023.

[11] Martin R. Albrecht, Lenka Mareková, Kenneth G. Paterson, and Igors Stepanovs. Four attacks and a proof for Telegram. In *2022 IEEE Symposium on Security and Privacy*, pages 87–106. IEEE Computer Society Press, May 2022.

[12] Martin R. Albrecht and Kenneth G. Paterson. Analyzing cryptography in the wild: A retrospective. *IEEE Security & Privacy*, 22(6):12–18, 2024.

[13] Apple. The apple psi system. https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf, August 2021. Since initial publication, Apple has modified the contents of the webpage. The initial announcement can be view at https://web.archive.org/web/20210805191006/https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf. Accessed on 18 Decemeber, 2023.

[14] Apple. Csam detection: Technical summary. https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf, August 2021. Since initial publication, Apple has modified the contents of the webpage. The initial announcement can be view at https://web.archive.org/web/20210805191352/https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf. Accessed on 18 Decemeber, 2023.

[15] Apple. Expanded protections for children. https://www.apple.com/child-safety/, August 2021. Since initial publication, Apple has modified the contents of the webpage. The initial announcement can be view at https://web.archive.org/web/20210805191220/https://www.apple.com/child-safety/. Accessed on 18 Decemeber, 2023.

[16] Apple. Expanded protections for children: Frequently asked questions. https://www.apple.com/child-safety/pdf/Expanded_Protections_for_Children_Frequently_Asked_Questions.pdf, Aug 2021.

[17] Apple. icloud data security overview. https://support.apple.com/en-us/102651, Feb 2025. Apple regularly updates their documentation. As such, a permenant copy of the version we consulted for this work can be found at https://web.archive.org/web/20250714063502/https://support.apple.com/en-us/102651.

[18] Matilda Backendal, Miro Haller, and Kenneth G. Paterson. MEGA: Malleable encryption goes awry. In *2023 IEEE Symposium on Security and Privacy*, pages 146–163. IEEE Computer Society Press, May 2023.

[19] Jim Baker, Katherine Charlet, Tom Donahue, Ed Felten, Avril Haines, Susan Hennessey, Chris Inglis, Sean Joyce, Susan Landau, Christy Lopez, Alex Macgillivray, Jason Matheny, Tim Maurer, Denis McDonough, Lisa Monaco, Laura Moy, Michelle Richardson, Ronald L. Rivest, Ari Schwartz, Harlan Yu, and Denise Zheng. Moving the encryption policy conversation forward. https://carnegieendowment.org/2019/09/10/moving-encryption-policy-conversation-forward-pub-79573, 2019.

[20] Mihir Bellare. The apple psi protocol. https://www.apple.com/child-safety/pdf/Technical_Assessment_of_CSAM_Detection_Mihir_Bellare.pdf, July 2021. Since initial publication, Apple has modified the contents of the webpage. The initial announcement can be view at https://web.archive.org/web/20210805192048/https://www.apple.com/child-safety/pdf/Technical_Assessment_of_CSAM_Detection_Mihir_Bellare.pdf. Accessed on 18 Decemeber, 2023.

[21] Mihir Bellare. A concrete-security analysis of the apple psi protocol. https://www.apple.com/child-safety/pdf/Alternative_Security_Proof_of_Apple_PSI_System_Mihir_Bellare.pdf, July 2021. Since initial publication, Apple has modified the contents of the webpage. The initial announcement can be view at https://web.archive.org/web/20210826041030/https://www.apple.com/child-safety/pdf/Alternative_Security_Proof_of_Apple_PSI_System_Mihir_Bellare.pdf. Accessed on 18 Decemeber, 2023.

[22] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 296–312. Springer, Berlin, Heidelberg, May 2013.

[23] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-

tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

[24] Glencora Borradaile. *Defend Dissent: Digital Suppression and Cryptographic Defense of Social Movements*. 2021.

[25] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[26] Ran Canetti and Gabriel Kaptchuk. The broken promise of apple's announced forbidden-photo reporting system – and how to fix it. https://www.bu.edu/riscs/2021/08/10/apple-csam/, 2021.

[27] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (abstract) (informal contribution). In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, page 462. Springer, Berlin, Heidelberg, August 1988.

[28] J.R. Douceur, A. Adya, W.J. Bolosky, P. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *Proceedings 22nd International Conference on Distributed Computing Systems*, pages 617–624, 2002.

[29] Brad Dwyer. Imagenet contains naturally occurring neuralhash collisions. https://blog.roboflow.com/neuralhash-collision/, Aug 2021.

[30] Filipe Espósito. Apple employees express concerns about new csam scanning. https://9to5mac.com/2021/08/12/apple-employees-express-concerns-about-new-csam-scanning/, Aug 2021.

[31] David Forsyth. Apple's csam detection technology. https://www.apple.com/child-safety/pdf/Technical_Assessment_of_CSAM_Detection_David_Forsyth.pdf, July 2021. Since initial publication, Apple has modified the contents of the webpage. The initial announcement can be view at https://web.archive.org/web/20210805192137/https://www.apple.com/child-safety/pdf/Technical_Assessment_of_CSAM_Detection_David_Forsyth.pdf. Accessed on 18 Decemeber, 2023.

[32] Sharon Bradford Franklin and Greg Nojeim. International coalition calls on apple to abandon plan to build surveillance capabilities into iphones, ipads, and other products. https://cdt.org/insights/international-coalition-calls-on-apple-to-abandon-plan-to-build-surveillance-capabilities-into-iphones-ipads-and-other-products/, Aug 2021.

[33] Chris Gane and Trish Sarson. *Structured systems analysis: tools and techniques*. McDonnell Douglas Systems Integration Company, 1977.

[34] Christina Garman, Matthew Green, Gabriel Kaptchuk, Ian Miers, and Michael Rushanan. Dancing on the lip of the volcano: Chosen ciphertext attacks on apple iMessage. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 655–672. USENIX Association, August 2016.

[35] Daniel Kahn Gillmor. Apple's new 'child safety' plan for iphones isn't so safe. https://www.aclu.org/news/privacy-technology/apples-new-child-safety-plan-for-iphones-isnt-so-safe, Aug 2021.

[36] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[37] Julie Inman Grant. End-to-end encryption: a challenging quest for balance. https://www.esafety.gov.au/newsroom/blogs/end-end-encryption-challenging-quest-for-balance, Feb 2020.

[38] Matthew Green. Can end-to-end encrypted systems detect child sexual abuse imagery? https://blog.cryptographyengineering.com/2019/12/08/on-client-side-media-scanning/, Dec 2019.

[39] Matthew Green, Vanessa Teague, Bruce Schneier, Alex Stamos, and Carmela Troncoso. An evaluation of the risks of client-side scanning. Real World Crypto 2022 (RWC 23). Recording available at https://www.youtube.com/live/AGlwtmfgMjk?si=16nnmGFRckmdOwxj&t=1520.

[40] Matthew D. Green and Alex Stamos. Apple wants to protect children. but it's creating serious privacy risks. https://www.nytimes.com/2021/08/11/opinion/apple-iphones-privacy.html, Aug 2021.

[41] Sophie Hawkes, Christian Weinert, Teresa Almeida, and Maryam Mehrnezhad. Perceptual hash inversion attacks on image-based sexual abuse removal tools. *IEEE Security & Privacy*, 23(3):64–73, 2025.

[42] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In Tadayoshi Kohno, editor, *USENIX Security 2012*, pages 205–220. USENIX Association, August 2012.

[43] Michael Howard and Steve Lipner. *The Security Development Lifecycle*, volume 34. 06 2006.

[44] Benjamin Ismail. Apple's censorship and compromises in hong kong. https://wp.applecensorship.com/wp-content/uploads/2022/12/Apps-at-Risk_-Apples-Censorship-and-Compromises-in-Hong-Kong.pdf, Dec 2022.

[45] Seny Kamara, Mallory Knodel, Emma Llansó, Greg Nojeim, Lucy Qin, Dhanaraj Thakur, and Caitlin Vogus. Outside looking in: Approaches to content moderation in end-to-end encrypted systems. https://cdt.org/wp-content/uploads/2021/08/CDT-Outside-Looking-In-Approaches-to-Content-Moderation-in-End-to-End-Encrypted-Systems-updated-20220113.pdf, Aug 2021.

[46] Sriram Keelveedhi, Mihir Bellare, and Thomas Ristenpart. DupLESS: Server-aided encryption for deduplicated storage. In Samuel T. King, editor, *USENIX Security 2013*, pages 179–194. USENIX Association, August 2013.

[47] Nadim Kobeissi. Nadim kobeissi on twitter. https://twitter.com/kaepora/status/1423387147172724741, Aug 2021.

[48] Nadim Kobeissi. An open letter against apple's privacy-invasive content scanning technology. https://appleprivacyletter.com/, August 2021. Accessed on 18 December, 2023.

[49] Tadayoshi Kohno, Yasemin Acar, and Wulf Loh. Ethical frameworks and computer security trolley problems: Foundations for conversations. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 5145–5162. USENIX Association, August 2023.

[50] Anunay Kulshrestha and Jonathan R. Mayer. Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 893–910. USENIX Association, August 2021.

[51] Diane Leblanc-Albarel and Bart Preneel. Black-box collision attacks on widely deployed perceptual hash functions. Cryptology ePrint Archive, Report 2024/1869, 2024.

[52] India McKinney and Erica Portnoy. Apple's plan to "think different" about encryption opens a backdoor to your private life. https://www.eff.org/deeplinks/2021/08/apples-plan-think-different-about-encryption-opens-backdoor-your-private-life, Aug 2021.

[53] Nat Meysenburg, Lauren Sarkesian, Ross Schulman, and Andi Wilson Thompson. A technical explainer on apple's concerning privacy changes. https://www.newamerica.org/oti/briefs/a-technical-explainer-on-apples-concerning-privacy-changes/, Aug 2021.

[54] Steven J. Murdoch. Apple letting the content-scanning genie out of the bottle. https://www.benthamsgaze.org/2021/08/17/apple-letting-the-content-scanning-genie-out-of-the-bottle/, Aug 2021.

[55] Engineering National Academies of Sciences and Medicine. *Decrypting the Encryption Debate: A Framework for Decision Makers*. The National Academies Press, Washington, DC, 2018.

[56] Jack Nicas. Apple's compromises in china: 5 takeaways. https://www.nytimes.com/2021/05/17/technology/apple-china-privacy-censorship.html, May 2021.

[57] Jack Nicas, Raymond Zhong, and Daisuke Wakabayashi. Censorship, surveillance and profits: A hard bargain for apple in china. https://www.nytimes.com/2021/05/17/technology/apple-china-censorship-data.html, May 2021.

[58] Kurt Opsahl. If you build it, they will come: Apple has opened the backdoor to increased surveillance and censorship around the world. https://www.eff.org/deeplinks/2021/08/if-you-build-it-they-will-come-apple-has-opened-backdoor-increased-surveillance, Aug 2021.

[59] Nilay Patel, Riana Pfefferkorn, and Jen King. Here's why apple's new child safety features are so controversial. https://www.theverge.com/22617554/apple-csam-child-safety-features-jen-king-riana-pfefferkorn-interview-decoder, Aug 2021.

[60] Benny Pinkas. A review of the cryptography behind the apple psi system. http://www.apple.com/child-safety/pdf/Technical_Assessment_of_CSAM_Detection_Benny_Pinkas.pdf, July 2021. Since initial publication, Apple has modified the contents of the webpage. The initial announcement can be view at https://web.archive.org/web/20210805190856/http://www.apple.com/child-safety/pdf/Technical_Assessment_of_CSAM_Detection_Benny_Pinkas.pdf. Accessed on 18 Decemeber, 2023.

[61] Erica Portnoy. Why adding client-side scanning breaks end-to-end encryption. EFF Blog. https://www.eff.org/deeplinks/2019/11/why-adding-client-side-scanning-breaks-end-end-encryption, Nov 2019.

[62] Jonathan Prokos, Neil Fendley, Matthew Green, Roei Schuster, Eran Tromer, Tushar Jois, and Yinzhi Cao. Squint hard enough: Attacking perceptual hashing with adversarial machine learning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 211–228, Anaheim, CA, August 2023. USENIX Association.

[63] Jonathan Prokos, Neil Fendley, Matthew Green, Roei Schuster, Eran Tromer, Tushar M. Jois, and Yinzhi Cao. Squint hard enough: Attacking perceptual hashing with adversarial machine learning. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 211–228. USENIX Association, August 2023.

[64] Eric Rescorla. Overview of apple's client-side csam scanning. https://educatedguesswork.org/posts/apple-csam-intro/, Aug 2021.

[65] Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015.

[66] Leah Namisa Rosenbloom. Cryptography and Collective Power. In Daniel Escudero and Ivan Damgård, editors, *Progress in Cryptology – LATINCRYPT 2025*, pages 3–39, Cham, 2026. Springer Nature Switzerland.

[67] Paul Rosenzweig. The law and policy of client-side scanning. https://www.lawfaremedia.org/article/law-and-policy-client-side-scanning, Aug 2020.

[68] Julian Sanchez. Apple's iphone: Now with built-in surveillance. https://www.cato.org/blog/apples-iphone-now-built-surveillance, Aug 2021.

[69] Sarah Scheffler, Anunay Kulshrestha, and Jonathan Mayer. Public verification for private hash matching. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 253–273, 2023.

[70] Sarah Scheffler and Jonathan R. Mayer. SoK: Content moderation for end-to-end encryption. *PoPETs*, 2023(2):403–429, April 2023.

[71] Susan Leigh Star and James R Griesemer. Institutional ecology,translations' and boundary objects: Amateurs and professionals in berkeley's museum of vertebrate zoology, 1907-39. *Social studies of science*, 19(3):387–420, 1989.

[72] Lukas Struppek, Dominik Hintersdorf, Daniel Neider, and Kristian Kersting. Learning to break deep perceptual hashing: The use case neuralhash. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 58–69, New York, NY, USA, 2022. Association for Computing Machinery.

[73] Kien Tuong Truong, Noemi Terzo, and Kenneth G. Paterson. Signal lost (integrity): The signal app is more than the sum of its protocols. Cryptology ePrint Archive, Paper 2026/484, 2026.

[74] Nicholas Weaver. Encryption and combating child exploitation imagery. https://www.lawfaremedia.org/article/encryption-and-combating-child-exploitation-imagery, Oct 2019.

[75] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

A Updating Apple's Initial Proposal

$\mathcal{F}_{\text{ftPSI-AD}}$

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 $\mathcal{F}_{\text{ftPSI-AD}}$:

- 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 $\bar{Y}$ 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 6: Apple's formalization of their ideal functionality.

We include Apple's ideal functionality verbatim in Figure 6; notation can be found in [13]. There are three main weaknesses in the modeling and formalism within Apple's initial proposal. The re-rendering of Apple's ideal functionality that we use in the main body of the work (Figure 2) directly addresses these limitations.

$\Pi_{\text{Apple-CSAM-Scan}}$

Parameters: threshold $\mathcal{T}$, generator $g \in \mathbb{G}$, set size n, and slack factor $\varepsilon \in [0,1]$.

Apple Initializing Scanning: When Apple is initialized on $X = \{\mathsf{img\_hash}_1, \ldots, \mathsf{img\_hash}_n\}$ and a client $\mathcal{U}$:

1. Sample a scalar α and set $L = g^{\alpha}$.
2. Sample h_0, h_1 and compute $T \leftarrow \mathsf{CuckooTable}_{h_0,h_1}(X)$. Repeat this process until $T \neq \perp$.
3. For $i \in [n']$, if $T[i] \neq \perp, p_i \leftarrow \mathsf{HashToCurve}(T[i])^{\alpha}$, otherwise set $p_i \xleftarrow{\$} \mathbb{G} \setminus \{0\}$
4. Send $\mathsf{pdata} = (L, p_1, \ldots, p_{n'}, h_0, h_1)$ to the client $\mathcal{U}$.

Client Device Setup: When the client $\mathcal{U}$ receives a message $\mathsf{pdata} = (L, p_1, \ldots, p_{n'}, h_0, h_1)$ from Apple:

1. If $L \notin \mathbb{G} \setminus \{0\}$, or $\exists\, i$ s.t. $p_i \notin \mathbb{G} \setminus \{0\}$, or $\exists\, i, j$ such that $P_i = P_j$, return $\perp$.
2. Sample and store an encryption key *adkey* and PRF key *hkey*.

Client Device Voucher Upload: When the client inputs a tuple $(\mathsf{img\_id}, \mathsf{img\_hash}, \mathsf{k})$:

1. Set $adct \leftarrow \mathsf{Enc}(adkey, \mathsf{k})$ and $sh \leftarrow \mathsf{SecretShareAtPoint}(adkey, \mathcal{T}, \mathsf{PRF}(hkey, \mathsf{img\_id}))$
2. Sample encryption key *rkey*, and compute $rct \leftarrow \mathsf{Enc}(rkey, (adct, sh))$
3. For $j \in \{0,1\}$:
   (a) Sample scalars β_j, γ_j and compute $q_j \leftarrow \mathsf{HashToCurve}(\mathsf{img\_hash})^{\beta_j} \cdot g^{\gamma_j}$ and $s_j \leftarrow (p_{h_j(\mathsf{img\_hash})})^{\beta_j} \cdot L^{\gamma_j}$
   (b) Set $ct_j \leftarrow \mathsf{Enc}(\mathsf{KDF}(s_j), rkey)$
4. Sample $b \xleftarrow{\$} \{0,1\}$ and send $(\mathsf{img\_id}, q_b, ct_b, q_{1-b}, ct_{1-b}, rct)$ to the server (i.e., shuffle the contents).

Apple Voucher Processing: When Apple receives $(\mathsf{img\_id}, q_0, ct_0, q_1, ct_1, rct)$ from a client $\mathcal{U}$:

1. For $j \in \{0,1\}$:
   (a) Set $rkey_j \leftarrow \mathsf{Dec}(\mathsf{KDF}(q_j^{\alpha}), ct_j)$
   (b) If $rkey_j \neq \perp$, set $(adct_j, sh_j) \leftarrow \mathsf{Dec}(rkey_j, rct)$
2. If there is exactly one index $j \in \{0,1\}$ for which $rkey_j \neq \perp$, add $(\mathsf{img\_id}, adct_j, sh_j)$ to $\mathsf{SHARES}_{\mathcal{U}}$.
3. If $|\mathsf{SHARES}_{\mathcal{U}}| > \mathcal{T}$, reconstruct *adkey* using $t+1$ shares in $\mathsf{SHARES}_{\mathcal{U}}$. Then, for each triple $(\mathsf{img\_id}_i, adct_i, sh_i) \in \mathsf{SHARES}_{\mathcal{U}}$, compute $k_i \leftarrow \mathsf{Dec}(adkey, adct_i)$ and add $(\mathsf{img\_id}_i, k_i)$ to $\mathsf{OUTSET}_{\mathcal{U}}$. Finally, output $\mathsf{OUTSET}_{\mathcal{U}}$.

Figure 7: Apple's Initial CSAM Scanning Protocol.

(1) Hash functions are not treated as leakage: In Apple's initial ideal functionality specification, $\mathcal{F}_{\text{ftPSI-AD}}$, the hash functions h_0, h_1 used to generate the Cuckoo table are not treated as leakage. This is clearly a problem, as the hash functions are rejection sampled as a function of the server's input. Simply put, there is no way around having this be a leakage associated with the system (even if that leakage, in practice, has very little impact). Therefore, we simply add this as an explicit leakage to the ideal functionality.

(2) PSI protocol is modeled as "one shot:" The second noteworthy weakness of $\mathcal{F}_{\text{ftPSI-AD}}$ is the choice to have the client supply all their inputs to the ideal functionality in one shot. It is clear, however, that this is not how the proposed system was going to work in practice. In fact, one of the beautiful aspects of Apple's proposal (from a cryptographic perspective) is that clients provide inputs *incrementally*: each time a client device wants to back-up an image, it creates a voucher for that image independently—a process which has computational and communication complexity independent of the number of images previously backed-up by the client device.

(3) Self composition for multiple parties: Apple's $\mathcal{F}_{\text{ftPSI-AD}}$ is between a single server and a single client and implicitly relies on *self composition* to scale up. Using self-composition in this way, however, means that the server is free to choose its inputs to each instance of $\mathcal{F}_{\text{ftPSI-AD}}$ independently.

A.1 Apple's Initial Protocol

Apple's proposed protocol is in Figure 7. $(\mathsf{Enc}, \mathsf{Dec})$ is a standard $\mathsf{IND\$-CCA}$ encryption scheme with *random key robustness*. We use these sub-protocols:

- $\mathsf{CuckooTable}_{h_0,h_1}(X)$ maps a set X into a table T such that for all $x \in X$, either $T[h_0(x)] = x$ or $T[h_1(x)] = x$, and empty elements are $\perp$. If a cycle exists, this function returns $\perp$.
- $\mathsf{HashToCurve}$ is a (programmable) random oracle from the image hash space to the group $\mathbb{G}$.
- $\mathsf{SecretShareAtPoint}(x, \mathcal{T}, r)$ generates the Shamir Secret share of x with threshold $\mathcal{T}$ at point r.

– KDF is a key derivation function that is modeled as a (programmable) random oracle.

Protocol limitations. One limitation of Apple's protocol is that Apple only proves *privacy* with respect to a malicious client. They do not, however, make any claims about the "correctness guarantee for the outputs" [13, Page 20]. Namely, there is no (formally proven) guarantee that the associated data that the server receives as output from the ideal functionality will match the inputs provided by the client. While this is not a problem if the PSI protocol is considered in isolation, this can be problematic when we require composability.

Apple's simulator for a malicious client [13, p. 24-25] stops after producing a simulated pdata and there is no description of the simulator's procedure for *extracting* the client's inputs for the ideal functionality. Creating a valid extraction strategy requires non-trivial changes to their protocol due to the properties of the Naor-Reingold randomized self-reduction. To see this problem, consider that the pdata message constructed by the simulator can embed at most n' candidate perceptual hashes. At the same time, the simulator will need to extract *all* of the perceptual hashes input by the client—which might be significantly more than n'. Thus, the simulator may eventually receive a voucher that it cannot "decrypt." For this voucher, there are two cases: the client honestly encoded a perceptual hash (which may or may not be in the ideal server's hash set) or the voucher is malformed and would never be "decryptable." The simulator cannot distinguish between these two settings by the properties of the randomized self-reduction, breaking the simulation.

A.2 Updating $\Pi_{\text{Apple-CSAM-Scan}}$

Enabling equivocation. The consequence of Apple's choice to model the private set intersection as "one-shot" is a subtle, cryptographic one, rather than a catastrophic compromise. Namely, when simulating the server, the simulator in Apple's initial formulation knows the final result of the private set-intersection protocol *before* it needs to simulate vouchers. As such, the simulation strategy is nearly trivial. When switching to an incremental formation of the ideal functionality, this strategy is no longer possible; the simulator knows if the voucher represents a match or a non-match, but does not know the "contents" of the voucher until *after* the threshold number of matches has been met.

We update the protocol to allow for this equivocation using standard techniques in the programmable random oracle model (an assumption already present in Apple's initial construction). For each perceptual hash for which the client wants to generate a voucher, they sample a random value r, query the random oracle on $(adkey \| r \| \mathcal{U} \| \mathsf{img\_id})$ to derive a per-input key $\mathsf{k}_{\mathsf{img\_id}}$, where $\mathcal{U}$ is a unique identifier for the client. $\mathsf{k}_{\mathsf{img\_id}}$ is then used to encrypt the image encryption key k (i.e., the data payload that the server should get from the private set intersection protocol) using an encryption scheme that is easy to equivocate (e.g., the one-time pad). The value r can then be released alongside $adct$. Once a server reconstructs $adkey$, as in the base protocol, they can query the random oracle consistently and retrieve the same key. Equivocation follows directly: the ciphertext contains a random string k and the simulator can program the random oracle such that decryption yields the correct values after the threshold is met.

Enabling extraction. Apple's proposal enables extraction by modeling the client device as strictly honest—possibly reasonable when the client software is written and managed by Apple. Another alternative would be to change the modeling on the HashToCurve function, but this would break the current extraction strategy for pdata (i.e., when simulating the server).

The much more conceptually simple—and, perhaps, more compelling—option: simply have the client prove that the voucher is well-constructed in zero-knowledge. The simulator can then extract directly from the zero-knowledge proof, circumventing the complexity of making changes to the internals of the protocol structure altogether. In practice, this approach is slightly non-trivial, given that the protocol makes use of multiple hash functions modeled as random oracles (specifically, HashToCurve and KDF) which cannot be used within the zero-knowledge proof. As such, we chose to change the modeling of KDF to just be a simple mapping function from group elements to the keyspace of the random-key robust encryption scheme. This change makes it difficult to extract s_j, but we can "fix" this problem by simply extracting from the zero-knowledge proof. Specifically, we generate the following proof:

$$\pi_{\text{voucher}} = \mathsf{PoK}\left\{\left(\{\beta_j, \gamma_j, g_j, s_j, i_j\}_{j\in\{0,1\}}\right) : \right.$$
$$\left. q_j = g_j^{\beta_j} g^{\gamma_j}, s_j = p_{i_j}^{\beta_j} L^{\gamma_j}, \mathsf{Dec}(\mathsf{KDF}(s_j), ct_j) \neq \perp \right\}$$

The simulator extracts $\{\beta_j, \gamma_j, g_j, s_j, i_j\}_{j\in\{0,1\}}$ from π_{voucher} and searches through the random oracle queries to see if g_j was generated as the output of a random oracle query. If yes, then the input to that random oracle query should be passed to the ideal functionality. If not, then the voucher will never be decryptable.

We note that this approach critically relies upon the random-key robustness of the encryption scheme. Without this property, it is conceivable that a malicious client could "work backwards" to make a decryptable voucher without picking any corresponding img_hash. Specifically, they could find a key $\mathsf{KDF}(\hat{s_j})$ that makes the decryption pass, and then solve for values of g_j, β_j, γ_j that produce such a value. But, if the value $\hat{s_j}$ is not selected at random, then this process would require solving a discrete log problem (i.e., by finding β_j, γ_j such that $\hat{s_j} = p_{i_j}^{\beta_j} L^{\gamma_j}$). Thus, it is enough to rely on the random-key robustness property.