Content Moderation for End-to-End Encrypted Messaging

Jonathan Mayer
Princeton University

October 6, 2019

Thursday evening, the Attorney General, the Acting Homeland Security Secretary, and top law enforcement officials from the U.K. and Australia sent an open letter to Mark Zuckerberg. The letter emphasizes the scourge of child abuse content online, and the officials call on Facebook to press pause on end-to-end encryption for its messaging platforms.

The letter arrived the same week as a widely shared New York Times article, describing how reports of child abuse content are multiplying. (As the article notes, it is unclear how much of the increase is due to improved reporting and detection and how much is due to growing criminal activity.) The article provides a heartbreaking account of how the National Center for Missing and Exploited Children (NCMEC) and law enforcement agencies are overburdened and under-resourced in addressing horrible crimes against children.

The law enforcement letter, remarks at a Department of Justice event on Friday, the New York Times article, and new NCMEC materials on encryption appear to reflect a common technical assumption:

1. Content moderation is fundamentally incompatible with end-to-end encrypted messaging.[1]

The law enforcement remarks appear to reflect an additional technical assumption:

[1] Law Enforcement Letter: "Our understanding is that much of [Facebook's reporting on harmful content] . . . will no longer be possible if Facebook implements its proposals as planned. NCMEC estimates that 70% of Facebook's reporting – 12 million reports globally – would be lost."
Deputy Attorney General Rosen Remarks: "The [hash-based] monitoring practices I've described are inconceivable with end-to-end encryption."
FBI Director Wray Remarks: "With the spread of user-controlled default encryption, providers frequently can't identify horrific images within encrypted data. . . . Most of the tips Facebook currently provides are based on content. With end-to-end encryption, those would dry up. Facebook itself would no longer be able to see the content of its users' accounts."
NCMEC Materials: "The use of end-to-end encryption would prevent the companies or any third-party from detecting illegal activity occurring on their platforms . . . ."
New York Times Article: "Data obtained through a public records request suggests Facebook's plans to encrypt Messenger in the coming years will lead to vast numbers of images of child abuse going undetected."

2. Enabling content moderation for end-to-end encrypted messaging fundamentally poses the same challenges as enabling law enforcement access to message content.[2]

My goal in this discussion paper is to provide a technical clarification for each of these points.

1. Forms of content moderation may be compatible with end-to-end encrypted messaging, without compromising important security principles or undermining policy values.

2. Enabling content moderation for end-to-end encrypted messaging is a different problem from enabling law enforcement access to message content. The problems involve different technical properties, different spaces of possible designs, and different information security and public policy implications.

I aim to demonstrate these clarifications by formalizing specific content moderation properties for end-to-end encrypted messaging, then offering at least one possible protocol design for each property.

Context and Scope

Child exploitation is an extraordinarily sensitive topic, and encryption policy has been a policy firestorm for years. Before turning to technical analysis, I would like to briefly emphasize the context for and narrow scope of this discussion paper.

First, my technical clarification in response to law enforcement and NCMEC is in no way a criticism. A couple years ago, I had the opportunity to visit NCMEC's headquarters and engage with law enforcement officers in child exploitation units. I have the utmost respect for the professionals who work around the clock to combat child exploitation.

Second, this discussion paper is inherently preliminary and an agenda for further interdisciplinary research (including my own). I have written it to organize my current thinking and to reflect constructive conversations with colleagues in computer science, law, and public policy. I am not yet prepared to normatively advocate for or against the protocol designs that I describe below. I am not claiming that these concepts provide sufficient content moderation capabilities, the same content moderation capabilities as current systems, or sufficient robustness against evasion.[3] I am also not claiming that these designs adequately address

[2] The framing for the Department of Justice event invoked "lawless spaces" and "warrant-proof encryption," linking the challenges of content moderation to the challenges of law enforcement access to data. Remarks at the event from Attorney General Barr, Deputy Attorney General Rosen, and FBI Director Wray all raised content moderation in connection with law enforcement access. Similarly, Alan Rozenshtein at the University of Minnesota Law School has written a blog post linking the New York Times article to encryption and law enforcement access.

[3] Note that robustness against evasion is also a challenge for messaging services that do not implement end-to-end encryption. A user might encrypt content before sending it, for example, or might switch to a different service.

information security risks or public policy values, such as free speech, international human rights, or economic competitiveness.

I do not know if there is a viable path forward for content moderation and end-to-end encrypted messaging that will be acceptable to technology platforms, law enforcement, NCMEC, civil society groups, information security experts, and other stakeholders. I do have confidence that, if such a path exists, we will only find it through open research and dialogue.

Technical Properties for End-to-End Encrypted Messaging

A foundational principle of applied cryptography is that, when designing or evaluating a system, we should be precise about the technical properties that we aim to accomplish. So, what are the properties that we might want for addressing harmful content in an end-to-end encrypted messaging app? Here is my attempt at a nonexhaustive taxonomy.

- **Content Moderation**
    - **User Reporting:** If a user receives a message that he or she believes contains harmful content, can the user report that message to the service provider?
    - **Known Content Detection:** Can the service provider automatically detect when a user shares content that has previously been labeled as harmful?
    - **Classifier-based Content Detection:** Can the service provider detect when a user shares new content that has not been previously identified as harmful, but that an automated classifier predicts may be harmful?
    - **Content Tracing:** If the service provider identifies a message that contains harmful content, and the message has been forwarded by a sequence of users, can the service provider trace which users forwarded the message?
    - **Popular Content Collection:** Can the service provider curate a set of content that has been shared by a large number of users, without knowing which users shared the content?
- **Lawful Access**
    - **Message History:** With appropriate legal process, can the service provider disclose messages that a user previously sent or received to law enforcement?
    - **Message Interception:** With appropriate legal process, can the service provider disclose messages that a user sends or receives going forward to law enforcement?

Note that each of these technical properties is distinct. Each poses a different set of design challenges. And, importantly, there may be room for stakeholder consensus on content moderation that has proven elusive on lawful access.

In the balance of this paper, I will discuss how to possibly implement each of the content moderation properties.

User Reporting

In an end-to-end encrypted messaging system, a service provider cannot read the content of messages—but intended recipients can. Enabling user reports is, therefore, not a technical challenge. If a user receives a message that he or she believes contains harmful content, the user can simply submit that message to the service provider (e.g., with a "Report Message" option).

The technical challenge, rather, is *authenticating* user reports. How can a service provider have confidence that a specific sender actually sent the flagged message, and at a specific time? Authentication is especially important for law enforcement, since harmful content in a messaging app may become forensic evidence in a criminal proceeding.

There is, thankfully, a straightforward means of establishing whether a message reported by a user is authentic. The service provider can associate each message with a cryptographic statement of authenticity that it was sent by a particular sender at a particular time, without learning the content of the message. If a user reports a message to the service provider, the provider can then check the statement of authenticity and verify the flagged message.

Facebook dubbed this design "message franking," and it is already deployed in Facebook Messenger. More recent academic work has formalized and expanded on Facebook's approach (Grubbs et al. 2017, Dodis et al. 2018, Leontiadis & Vaudenay 2018, Tyagi et al. 2019, Huguenin-Dumittan & Leontiadis 2019).

Known Content Detection

In a traditional messaging platform, when a user sends a message, the service provider can check the content against a dataset of known harmful content. How can a service provider accomplish that sort of known content matching in an end-to-end encrypted messaging app?

The technical foundation for known content detection is a concept called "perceptual hashing." A perceptual hash is simply a number derived from a piece of content, with several important properties: If pieces of content are identical, they will have the same hash. If pieces of content are perceptually similar (e.g., scaled or rotated images), they will likely have similar hashes.[4]

[4] Hamming distance is a common metric for evaluating whether two perceptual hashes are similar.

And if two pieces of content are not perceptually similar, they will likely not have similar hashes.

Perceptual hashing is widely used today for detecting known harmful content. Service providers can share perceptual hashes without sharing harmful content, and perceptual hash-based content matching requires both relatively little storage and relatively little computation. Perceptual hashing algorithms like Facebook PDQ and TMK+PDQF, Microsoft PhotoDNA, wHash, dHash, pHash, and aHash are free and relatively easy to implement.

A possible direction for detecting known harmful content in an end-to-end encrypted messaging app is to implement perceptual hashing within the app. Consider the following design for checking the content of a message that a user sends or receives.

1. Begin with a set of perceptual hashes, derived from known harmful content.

2. Insert the hashes into a data structure that is small enough to fit in device storage and that can quickly check whether a new hash is similar to a hash in the data structure.[5] Note that this data structure would be probabilistic; depending on the implementation, it could generate false positives or false negatives.

3. Distribute the data structure with the messaging app, and update the data structure as the service provider learns about new perceptual hashes of harmful content.

4. When a user sends or receives a message, the app generates a perceptual hash for the message content and checks it against the data structure.

5. If there is a possible match to known harmful content, the app automatically reports the unencrypted message to the service provider for further evaluation (e.g., manual review).[6]

This design involves a tradeoff between storage space on the user's device and the performance of similarity detection. That could have significant implications: a false positive means the messaging app has disclosed content that is not known to be harmful, and a false negative means the messaging app has failed to disclose content that is likely harmful.

It may be possible to improve this tradeoff between device storage and detection performance. For example, it may be possible to construct a protocol (either in addition to or separate from the design above) that checks a perceptual hash against a service provider's database, without

[5] A combination of locality-sensitive hashing and a Bloom filter could provide these properties, as could a more sophisticated distance-sensitive Bloom filter (see Kirsch & Mitzenmacher 2006, Hua et al. 2011, and Goswami et al. 2017).

[6] If there is not a match, it may also be possible for the app to generate a zero-knowledge proof that the message content is not known to be harmful.

revealing the hash that the app is checking.[7] Cloudflare and Google, for example, already use a similar approach for checking whether a user's hashed password is in a large dataset of compromised hashed passwords (see Thomas et al. 2019 and Li et al. 2019).

Classifier-based Content Detection

Facebook and Google (among other technology platforms) have publicly confirmed that they use machine learning classifiers to detect new instances of harmful content. How could they continue to apply classifiers in an end-to-end encrypted messaging environment?

A straightforward approach would be to generate machine learning models for harmful content, then deploy the models in the messaging app. That type of decentralized, in-app machine learning application is now quite common, and it is supported by popular machine learning libraries.

Consider the following design, which a service provider could use for each distinct category of harmful content:

1. Generate a classifier for the category of harmful content.[8] This step could require a large volume of data and significant computational capacity; a service provider would implement it on its own infrastructure.

2. If necessary, reduce the size of the classifier so that is small enough to fit in device storage.[9]

3. Distribute the classifier with the messaging app, and update the classifier as the service provider implements improvements or incorporates new data.

4. When a user sends or receives a message, the app runs the classifier on the message content. This step can be fast—evaluating with a machine learning classifier is much easier than training the classifier.

5. If the classifier predicts that the message content is harmful, the app automatically reports the unencrypted message to the service provider for further evaluation (e.g., manual review).

---

[7] This task could possibly be formulated as an oblivious transfer, private information retrieval, or private set intersection problem, for instance. Or it could be structured as a database query with k-anonymity, trading some rigor for easier implementation.

[8] This could be a straightforward application of supervised learning to labeled harmful and non-harmful content.

[9] There is an active research subfield examining how to generate compact or pruned machine learning models, trading accuracy and other performance characteristics against storage and computational requirements.

Much like the approach to known content detection, this approach would have tradeoffs. More sophisticated models would require greater on-device storage and computation, but they could also increase classification accuracy (or other performance characteristics). False positive and false negatives would pose the same problems as for known content detection.

One possible improvement to the protocol would be to tune the in-app models for efficiency and a low false negative rate. If a model predicts that message content may be harmful, the app could submit the unencrypted content (or features derived from the content) to a more sophisticated server-side model.[10]

Another possible, longer-term improvement would be to use a protocol that allows a messaging app to submit message content to the service provider for classification, without revealing the content to the service provider. This type of privacy-preserving "outsourcing" for machine learning is an active area of research, and recent progress is promising (e.g., Dowlin et al. 2016, Liu et al. 2017, Juvekar et al. 2018, and Jiang et al. 2019).

Content Tracing

When a user receives a message that contains harmful content, sometimes the user will forward the message to additional users.[11] If a service provider cannot read a piece of content, how can it determine the sequence of forwards that preceded the content?

One approach would be to retain metadata for all messages. When a piece of content is flagged as harmful, the service provider could try to use metadata attributes (e.g., message size and timestamp) to reconstruct the content's path. That approach is probabilistic and may be inaccurate, though, and retaining message metadata has a significant privacy impact.[12]

A better approach would be to build on the message franking concept, which enables a service provider to authenticate a reported message.[13] Each message could come with not just a

[10] The app could attempt to protect a user's identity unless the server-side model classifies content as harmful. For example, the app could submit the content for classification using an anonymizing network, a mixnet, or a metadata-private messaging protocol. Since content often contains identifying information, these approaches would not necessarily be effective.
[11] This issue has been especially problematic for WhatsApp, which responded by limiting message forwarding.
[12] Facebook has indicated that it plans to use message metadata to identify and trace harmful content.
[13] Consider the following possible design: At each stage of forwarding, the sender would add the prior message franking material to the bundle, generate a random symmetric key, and encrypt the bundle with the new key. It would then encrypt the key with an asymmetric key for the service provider, append the encrypted key to the message content, and send the message with franking. If the message recipient or the app flags the content as harmful, the app can share the latest message, its franking, the franking bundle, and the encrypted franking bundle key with the service provider. The service provider can then authenticate the message, decrypt the franking bundle key, and iteratively decrypt each franking in the bundle. At the end of this process, the service provider has sufficient information to authenticate the entire series of forwards that preceded a message.

franking for the last sender, but also a "franking bundle" for previous forwards. If a user forwards a message, the app could add the latest franking to the bundle. The service provider would generate a cryptographic statement of the bundle's authenticity, without either the provider or the recipient learning the bundle's contents. If a user or the app flags a message, the service could then unpack the bundle, read the sequence of message frankings, and reconstruct the sequence of message forwards.

Tyagi et al. 2019 have proposed a similar concept, using server-side storage rather than a franking bundle.[14] Their paper also proposes an approach for forward content tracing, enabling a service provider to trace an entire tree of forwarding activity if a piece of content is reported.

Popular Content Collection

A service provider might want to know about pieces of content that have been shared by a large number of users. One reason is for manual labeling—a provider might want to have human moderators review popular content. Another reason is for automated detection of harmful content—building a machine learning model for harmful content requires training data that consists of *non-harmful* content. How could an end-to-end secure messaging platform compile a collection of popular content, when it is unable to access message content?

One possibility would be to rely on the subset of users who have not enabled end-to-end encryption. So long as end-to-end encryption is optional for a messaging app, that might be sufficient.

Another option would be to rely on platforms that don't use end-to-end encryption. For example, it might be a reasonable assumption that popular content on Facebook is roughly similar to popular end-to-end encrypted content on WhatsApp.

Yet another option would be to enable a service provider to collect widely shared content—content that is likely to be non-harmful—without revealing which users shared the content. For example, a service provider could maintain a count of how often content with a specific hash is shared, without knowing which users have shared content with that hash.[15] When the shares of a specific hash exceed a threshold, the service provider could flag it, and the

[14] One possible advantage of the server-side approach is that it does not disclose the number of preceding forwards, which may be a desirable property. A franking bundle, by contrast, reveals the number of forwards with its size. That issue could be mitigated by using a fixed size for franking bundles, at the expense of additional message overhead.

[15] This design could possibly be implemented as a differentially private count for each hash. The hashes should be cryptographic hashes (rather than perceptual hashes), since the design depends on the property that the service provider cannot learn about content from its hash. It may be possible to improve the design by using a secret sharing approach for each hash, such that a service provider would only be able to request a copy of the content associated with a hash if it can cryptographically demonstrate that a large number of users have shared content with that hash.

next time a user shares matching content, their app could submit the content to the service provider in a way that does not disclose the user's identity.[16]

A possible longer-term solution to this challenge would be enabling the service provider to refine its classifiers for harmful content, without sending non-harmful content back to the service provider. Each instance of the messaging app could make small improvements to the classifiers using the user's message content, and the app would periodically submit those improvements to the service provider—without disclosing any message content. This type of privacy-preserving "federated learning" is an area of ongoing, promising research (e.g., Geyer et al. 2018).

Parting Thoughts

In closing, I would like to reemphasize the narrow goal of this paper: demonstrating that forms of content moderation may be technically possible for end-to-end secure messaging apps, and that enabling content moderation is a different problem from enabling law enforcement access to content. I am not yet advocating for or against the protocols that I have described. But I do see enough of a possible path forward to merit further research and discussion.

Acknowledgments

Thanks to Joe Bonneau, Ed Felten, Nick Feamster, Ben Kaiser, Anunay Kulshrestha, Arvind Narayanan, and Laura Roberts for comments on an earlier draft. All views, errors, and omissions are solely my own.

[16] The app could, for example, use an anonymizing network, a mixnet, or a metadata-private messaging protocol.