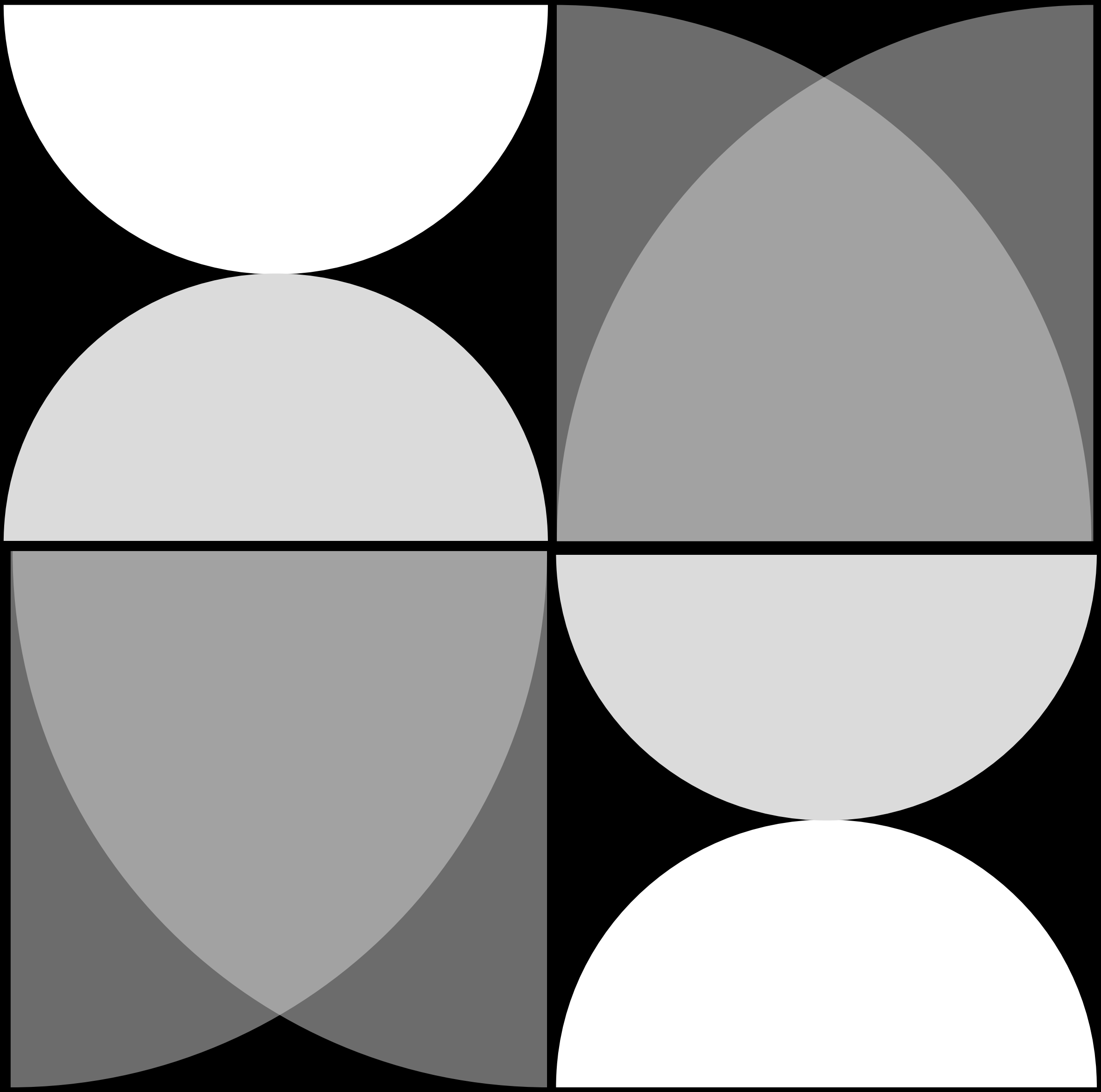
Content Moderation in Encrypted Platforms

Tech, Media, Democracy

HNG in there

Novia Wu
Haitong Lin
Grace Nho



May 2023

TABLE OF CONTENTS

Introduction

The **Content Moderation in Encrypted Platforms Report** is a document that aims to provide an overview of available technical infrastructures that balance **privacy** and **content moderation**. The report is intended to serve as a general guide for social platforms and encrypted services to understand the possibility of addressing this dilemma. The report calls for companies to consider implementing technical solutions to combat online misconduct through moderation.

Balancing privacy concerns and combating online crime such as sexual exploitation has been an ongoing debate. With the rise of social media and encrypted messaging services, it's harder to responsibly moderate content while preserving users' privacy. Many companies found themselves having to take sides, either guaranteeing privacy with little to no moderation or centralizing moderation with a strict takedown and banning rule. Because of the encrypted nature of some messaging apps and the promise to keep users' data private, companies struggle to find technical solutions that support effective content moderation.

To address this question, this report focuses on synthesizing and analyzing the current landscape of technical solutions that deal with this dilemma. The report can serve as a guide for companies and platforms to stay updated with new technologies.

We will first explain the concept of E2EE and the difficulty of content moderation on such a platform. Then we will explore the possible technical solutions, tackling the problem from prevention and reporting. Prevention techniques mainly involve detection of harmful content and analysis of behaviors to stop potential malicious users from posting. Report techniques mainly involve attribution of malicious content.

We found that the most effective prevention technique should focus on providing transparency and ensuring accountability through publicly verifiable hashing, and the most effective report mechanism should focus on holding malicious users accountable. Through the report, we hope to provide concrete guidelines for companies and platforms to take action and actively research and implement technical infrastructures to moderate content ethically.

Background on E2EE

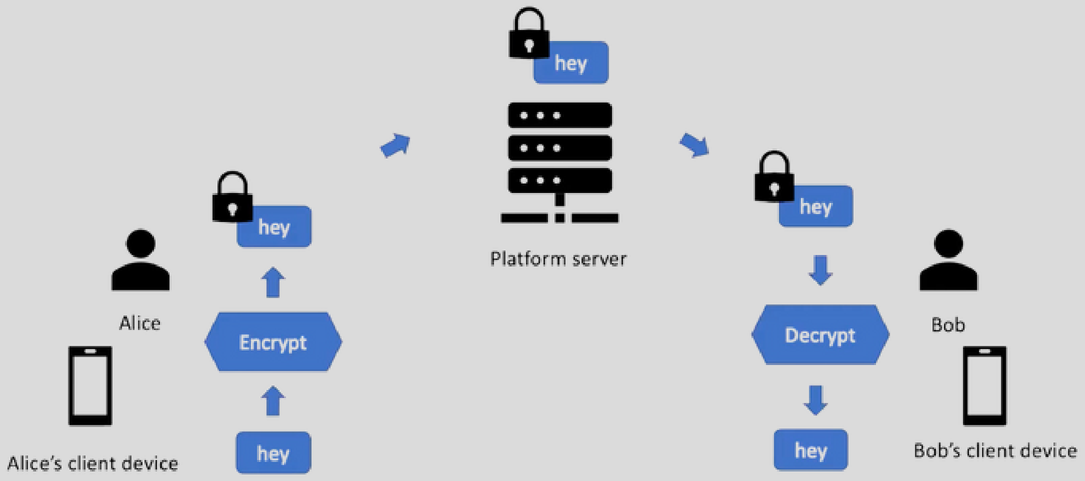


Figure 1. Simplified protocol of an end to end encrypted messaging service [1]

Encrypted services like Whatsapp, Signal, and Telegram are growing in popularity due to the demand for user privacy. Many platforms use End-to-End Encryption (E2EE) to protect the data their users uploaded. However, the exact mechanism of E2EE can be murky to the general public. Here we explain a simplified protocol of E2EE messaging service as an example.

End-to-End Encryption (E2EE) means that messages and content are only visible from one client-end (sender) to another client-end (receiver). The platform server as a relay hub will not see the content. In the example above, E2EE can be achieved by encrypting the plaintext content of Alice on her device, using a key that is known to only Alice and Bob through secret sharing. Then the encrypted message, or the ciphertext, will be relayed to the server, then to Bob's device. Bob's device is able to decrypt the message using the key. The server or any malicious person that wants to hijack the ciphertext will not be able to recover the plaintext.

Content moderation is a process through which the platform identifies and detects harmful content, and takes the necessary steps to prevent or punish such behaviors. The difficulty of content moderation increases when the server is not able to see the content users post. While privacy is being preserved, detection and removal of harmful materials become harder in E2EE setting.

Our Approach

While there is currently no industry-wide consensus on what is the best way to address the problem of content moderation in E2EE platforms, research are done to further the discussion. We want to highlight in this report some of the state-of-the-art technologies available and allow platforms to continue to build on this foundation.

Based on the current literature, we divided our approach into two parts: prevention and report. We think prevention techniques can help stop the harm before it happens, and reporting mechanisms allow platforms to identify malicious users and hold them accountable.

Prevention	Detection of harmful content through active filtering or prevention of posts from malicious users
Report	Provide technical capabilities that allows for tracing of the source of harmful content

For each of the technical protocol or system, we present the background and motivations for the approach, then show the technical details in simple language, and analyze its use case, potential challenge, and whether if we think the approach would be beneficial.

Prevention Techniques

Prevention techniques mainly involve automatic detection of harmful content or verification of non-harmful content. Here is an overview of the four main categories of technical, preventative solutions that aim to preserve users' privacy.

01 | **Metadata Analysis**

Analysis of group messaging traffic, deletion time, file length and type etc. to identify potential dangerous groups and content

02 | **Verifiable Hashing**

Detection of harmful content, before users can post, done on client-side using publicly verifiable hash matching

03 | **ZK Middlebox**

Prevent users from posting harmful content by making them prove that their content is not in the harmful set

04 | **Machine Learning**

Client-side ML models can classify newly generated malicious content and detect malware and spam

Metadata Analysis

Summary:

Metadata analysis is a machine learning technique that

- analyzes the metadata associated with harmful content, instead of actual content
- perform binary classification of content based on different metadata characteristics

Background and Motivation:

E2EE platforms have a privacy guarantee that the content of individual users will not be known to the servers. This poses substantial difficulties when platforms need to detect potentially dangerous materials. To address this problem, metadata about the content such as the file name users uploaded, the file path, extension type, etc. can be analyzed to classify if that file contains harmful materials.

Example Technical Mechanism:

The intuition behind metadata analysis is to train a classification model using publicly open dataset on file names and their association with CSAM [2]. Research showed that Convolutional Neural Networks model performed best with accuracy of 0.97 and recall of 0.94. Different adversarial models are also tested and the model withstand the attacks.

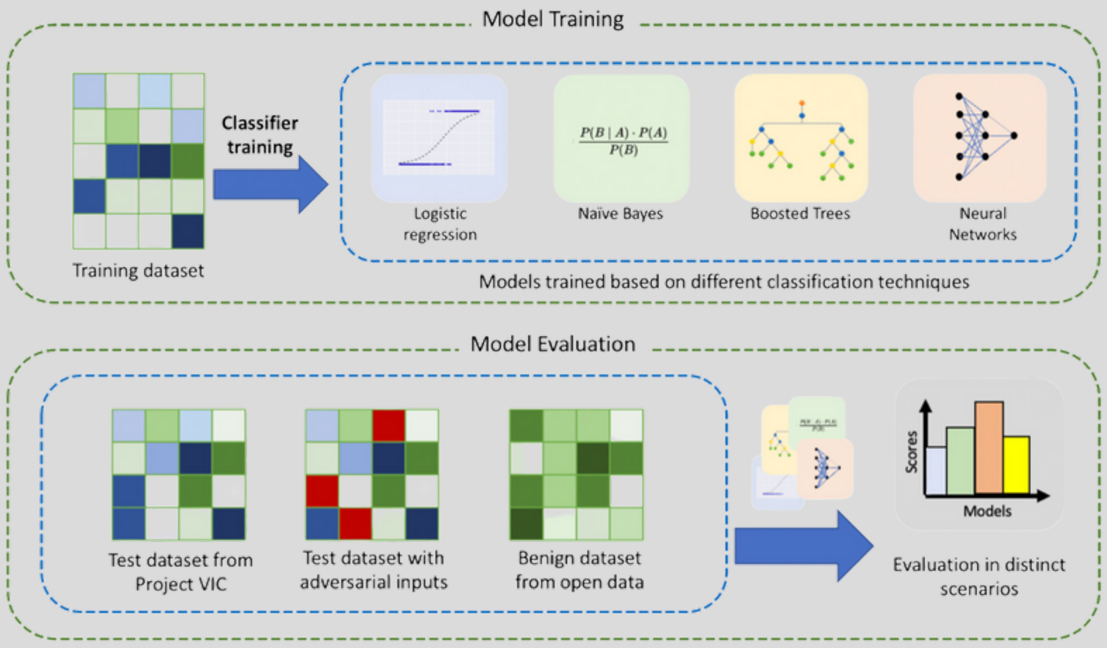

Model Training
Training dataset
Classifier training
Logistic regression
$\frac{P(B|A)\cdot P(A)}{P(B)}$
Naïve Bayes
Boosted Trees
Neural Networks
Models trained based on different classification techniques
Model Evaluation
Test dataset from Project VIC
Test dataset with adversarial inputs
Benign dataset from open data
Scores
Models
Evaluation in distinct scenarios

Figure 2. Schema of training classification models and evaluation [2]

Analysis, Potential Use, and Problems:

The solution provided here on metadata analysis can be a potential way to protect users' privacy. Since E2EE platforms do not conceal metadata, they are readily available to use for content detection and moderation. However, research must be done on the accuracy and feasibility of these models to ensure that they don't reveal sensitive personal data that can be inferred from the metadata.

Verifiable Hashing

Summary:

Public verifiable perceptual hash matching (PHM) [3] is a cryptographic technique that allows:

- certified groups to approve the set of harmful content
- ensure no lawful content is in the set of harmful content
- notify users of false positives

Background and Motivation:

One of the most researched areas of detection of harmful material in the E2EE setting is perceptual hashing. It involves hashing plaintext, image, and video into a unique fingerprint. Then that fingerprint is used to compare to a database of known illegal content fingerprints, curated by trusted third parties.

For example, National Center for Missing and Exploited Children (NCMEC) curated a database of Child Sexual Abuse Material (CSAM) [4], and platforms such as PhotoDNA [5] and Content Safety API [6], use this database to detect known harmful material.

5

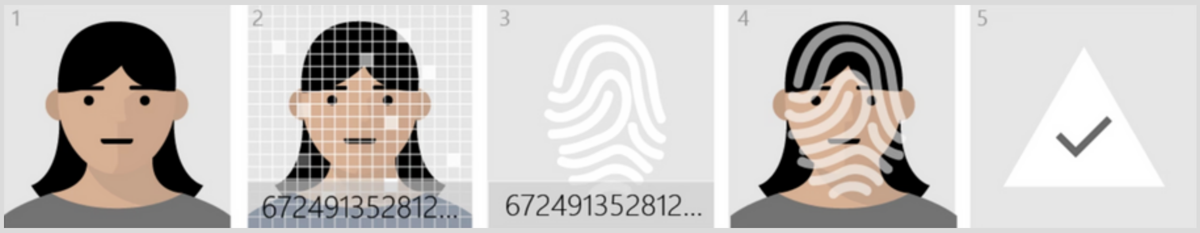


Figure 3. Steps of how PhotoDNA works [5]

Perceptual hash matching is designed such that the hashing will be done on the client side, and the server only gets the fingerprint and not the actual content, in an effort to preserve privacy. However, there has been much debate about this method. Specifically, Apple's proposed PSI system received much backlash [7]. Privacy experts are concerned that such technology will allow authoritative regimes or other malicious groups to pressure the platform to check for materials they deem "inappropriate". The potential abuse of power can be seen as an infringement on people's right to speech and privacy.

To address this problem, researchers considered ways to build on hold the platforms and institutions accountable by allowing the public to verify the hash set of harmful materials.

Example Technical Mechanism:

The intuition behind it is that multiple institutions that can be regarded as trusted third parties will have to verify and approve the database of harmful content [3].

- Threshold certification of the hash set
    - A service can prove that the hash set was certified by a specific combination of external groups. These could be child safety groups, such as NCMEC and the Internet Watch Foundation (IWF).
- Zero-knowledge proof of non-inclusion
    - Given the hash of the image, the server needs to prove that the hash is a non-member of the known hashset of harmful content without revealing anything about the hashset to the users.
- Eventual notification of false positives
    - Servers are cryptographically committed to alerting users if their content has been marked as potentially harmful when it is not.

Analysis, Potential Use, and Problems:

Overall, the techniques mentioned above utilizes cryptographic commitment schemes, threshold signature, zero-knowledge proofs, and other protocols to leverage the balance between user privacy and safety. However, the solutions presented here are iterations of the controversial perceptual hash matching, and they do not guarantee desirable behavior from all parties involved. We do believe that in extreme use cases, such as CSAM, it is possible to embed public accountability into the hashset to ensure transparency will allow platforms to address the potential abuse criticism of perceptual hash matching.

Zero Knowledge Middlebox

Summary:

Using cryptography, Zero-Knowledge (ZK) middlebox [8] has the following features:

- users send ZK proofs to the middlebox between them and the server
- users prove to the middlebox that their content is compliant with policy without reveal
- middlebox and server would not gain formation about the content

Background and Motivation:

Recent developments in cryptography, especially ZK-proof systems, have been fruitful and inspiring for many fields. A ZK proof protocol works in the following way: It allows a prover to convince a verifier of the truth of a statement while keeping secret the basis of its argument. If the statement is false, then the verifier is highly unlikely to be convinced [9]. In an E2EE setting, the need for encryption and verification can be the key to content moderation. Researchers have been trying to implement ZK-proof protocols into E2EE or other Network systems in an attempt to preserve clients' data.

Example Technical Mechanism:

The intuition of ZK middlebox is that it serves as a middleman in the local network between the client and the server [8]. ZK middlebox will establish some policy guideline, the client must send along a proof that their content follows those policy. The middlebox will verify the proof and send the network to the server if proof is valid. The protocol is designed in below 5 steps:

1. The middlebox sets up policy regarding the connection. Example could be a list of blocked content, blocked accounts etc.
2. When client joins the network, middlebox will relay the policy to the client.
3. When client wishes to send a message to a server, the client need to establish a secure channel first.
4. Client generate a ciphertext based on the plaintext M they want to transmit, and generate a proof that the plaintext is in compliance with policy.
5. The middlebox will verify the proof and verified ciphertext will be replayed to server.

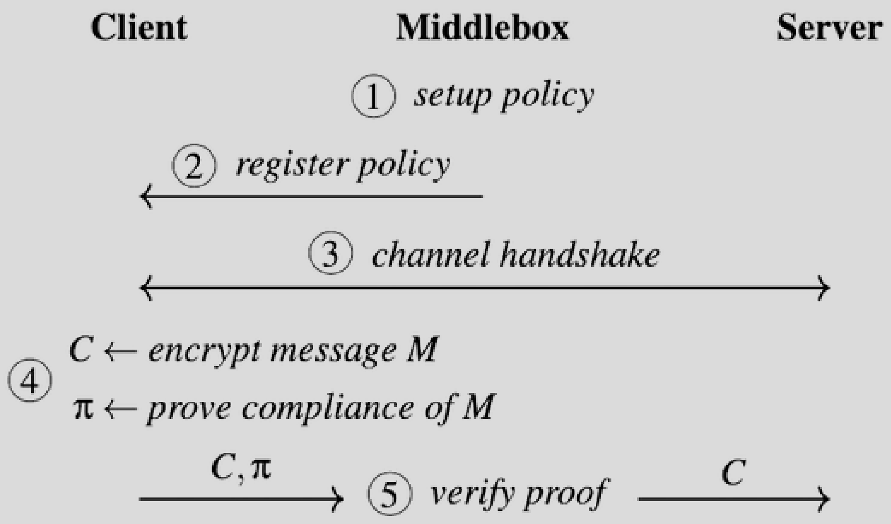


Figure 4. Steps of how ZK middlebox scheme [8]

Analysis, Potential Use, and Problems:

Using ZK proof in general can be a good direction for privacy and security. ZK middlebox can in theory allow for proactive prevention of harmful material by making users prove their innocence. However, ZK-proof implementation as of today is still very limited, largely because of its weak performance on latency. Platforms that prioritize performance may have a hard time adopting this technology. Future research should be done to mitigate this tradeoff for the protocol to be more applicable.

Machine Learning

Summary:

Many Machine Learning empowered models can help mitigate and prevent harmful content through:

- spam and phishing website detection through URL
- federated learning for malware detection
- fully client-side scanning for unseen harmful content (controversial)

Background and Motivation:

With the growth in capabilities in machine learning, especially natural language processing, computer vision, and federated learning, ML models can be helpful in the detection and prevention phase. Many of the predators start to lure victims by sending them phishing websites that look like Twitter or Facebook, then they can gain access to the victim's personal information through these social media accounts. To cut the problem from the start, ML can be deployed on a large scale to filter and detect potential spam and phishing websites and other malware. ML can also be particularly helpful for never before seen content due to its learning abilities.

Example Technical Mechanism:

Different ML models can be engineered and tuned to perform specific tasks.

- To detect and filter harmful websites, Deep Neural Networks, Logistic Regression, and a hybrid of both all can be tuned to achieve high accuracy and performance while maintaining privacy [10].
- For newly generated harmful content, predictive models such as classifiers, object detectors that utilize Convolutional Neural Networks, and other Deep Learning models can be helpful [11].

Analysis, Potential Use, and Problems:

Machine learning can be powerful if used in the right place in the right way. It can help detect phishing links that start the process of online harassment and can help identify newly generated harmful content. Platforms can try to adopt spam filters and warnings to engage users to protect themselves from malicious agents. However, using ML-based client-side scanning remains controversial due to the ease of abuse of these systems. Training data or personal data can be leaked by the models if not carefully designed.

Reporting Techniques

While it's important to prevent malicious users from posting harmful content and reaching mass audiences, sometimes those preventative techniques don't always work. When situations escalate and harmful content has already been posted and spread, giving the platform the ability to investigate the source of the harmful content is crucial. We will discuss two technologies developed to safely trace back the origin of encrypted content, without revealing private information.

01 | **Message Traceback**

Use cryptographic approach to trace back the source of malicious content while preserving E2EE's privacy guarantee

02 | **Message Franking**

Binding of sender and receiver identity to prevent malicious receivers from framing senders for content they did not send

Message Traceback

Summary:

Message traceback enables E2EE platforms to track down the source of malicious content:

- keeps a trace for the message forward tree while protecting clients' identities [12]
- balances privacy and security, confidentiality and accountability goals
- easy to implement and integrate into existing E2EE platforms

Background and Motivation:

One of the key challenges of content moderation in E2EE platforms is holding malicious users accountable without compromising the confidentiality of honest users. Especially in a group chat setting, messages and contents are being sent and forwarded, clouding the source of harmful content. When users report certain harmful messages, platforms need the ability to investigate the source. However, the task is much harder when the content is encrypted.

Example Technical Mechanism:

To achieve tracing capabilities and get to the source of malicious content, cryptographic tracing schemes can be used [12].

- Path traceback scheme uses symmetric encryption to add tracing tags to ciphertext.
- Tree traceback is an extension of path traceback, it allows the trace to reveal all other recipients of the same ciphertext.

Explanation of message tracing algorithm is explained in below annotated diagram.

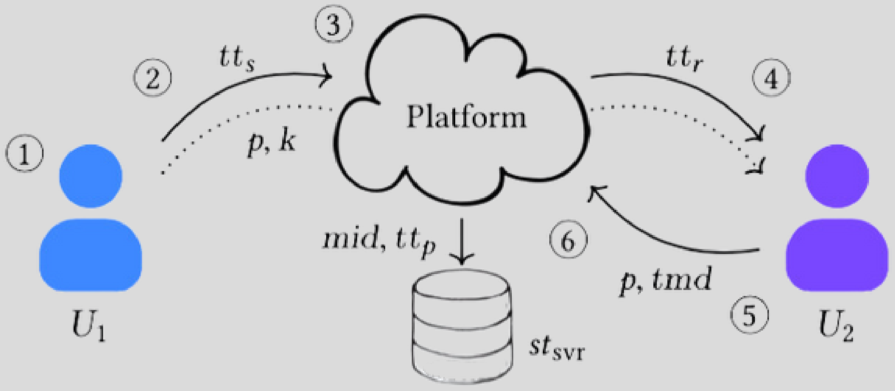
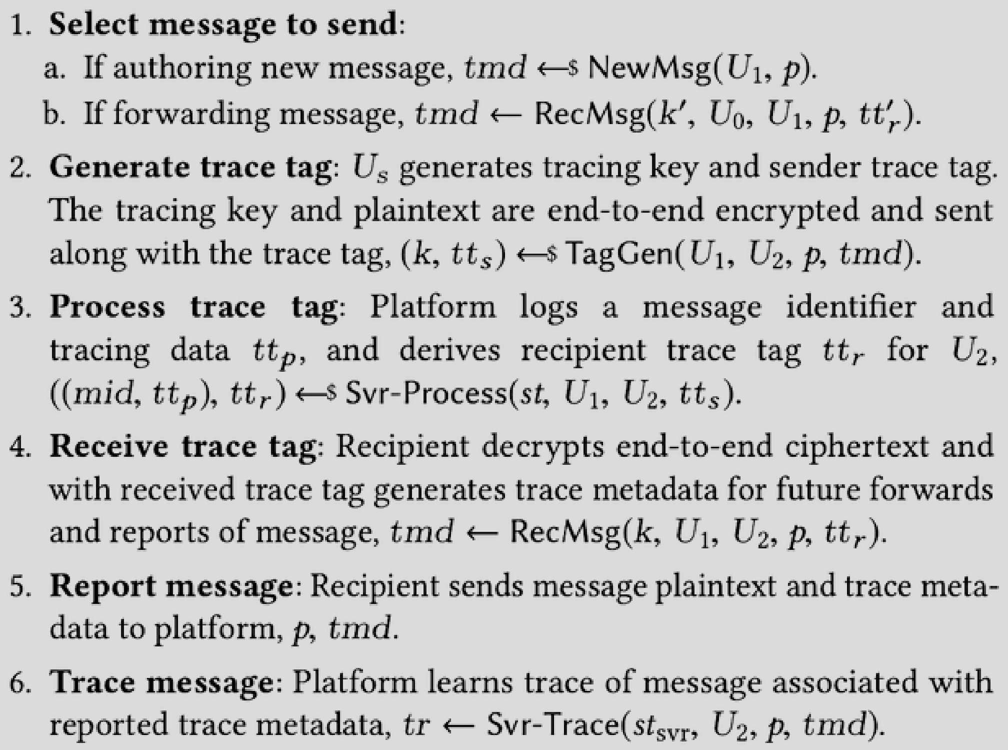


Figure 5. Usage of message tracing algorithm [12]



1. **Select message to send**:
   a. If authoring new message, $tmd \leftarrow\$ \mathsf{NewMsg}(U_1, p)$.
   b. If forwarding message, $tmd \leftarrow \mathsf{RecMsg}(k', U_0, U_1, p, tt'_r)$.
2. **Generate trace tag**: U_s generates tracing key and sender trace tag. The tracing key and plaintext are end-to-end encrypted and sent along with the trace tag, $(k, tt_s) \leftarrow\$ \mathsf{TagGen}(U_1, U_2, p, tmd)$.
3. **Process trace tag**: Platform logs a message identifier and tracing data tt_p, and derives recipient trace tag tt_r for U_2, $((mid, tt_p), tt_r) \leftarrow\$ \mathsf{Svr\text{-}Process}(st, U_1, U_2, tt_s)$.
4. **Receive trace tag**: Recipient decrypts end-to-end ciphertext and with received trace tag generates trace metadata for future forwards and reports of message, $tmd \leftarrow \mathsf{RecMsg}(k, U_1, U_2, p, tt_r)$.
5. **Report message**: Recipient sends message plaintext and trace metadata to platform, p, tmd.
6. **Trace message**: Platform learns trace of message associated with reported trace metadata, $tr \leftarrow \mathsf{Svr\text{-}Trace}(st_{\mathrm{svr}}, U_2, p, tmd)$.

Figure 6. Logic schema of tracing algorithm [12]

Analysis, Potential Use, and Problems:

Message tracking is potentially a practical way to protect users' privacy while ensuring the security of the platform. Path traceback is particularly useful when a user reports a particular message, the tracing tags can be decrypted to reveal the chain for message forwarding. Tree traceback is a valuable tool in scenarios like online sex trafficking. It can be used to identify perpetrators and people who are engaging in illicit trade. The lightweightness of the system also allows for a fast and smooth adaptation of this technique by E2EE platforms. Overall, message tracing can be a good remedy to uncover criminals after reporting.

Message Franking

Summary:

Message franking is a cryptographic scheme for verifiable user reporting [13]:

- accountability of the malicious content sender, if they are the ones sending
- confidentiality of the content, unless voluntarily revealed through reporting

Background and Motivation:

Malicious users could be both the sender and the receiver. In the latter case, there were not enough mechanisms to prevent malicious receivers from framing a sender for potential abuse material. To address this issue, Facebook published its message franking technique [14]. It is essentially a cryptographic scheme that allows these three properties: authenticity, confidentiality, and third-party deniability.

The authenticity property ensures that if a user submits a report then the message must have legitimately originated from the sender’s device. The confidentiality property ensures that no outside party — including Facebook — should learn the content of a message unless a participant in a secret conversation voluntarily shares that information. Finally, the third-party deniability property ensures that no party outside of Facebook can cryptographically determine the validity of a report.

To extend on this promising technology, many researchers have developed new cryptographic schemes such as compactly committing authenticated encryption with associated data (AEAD) [13].

Example Technical Mechanism:

The sender of the message needs to cryptographically sign their message, binding their identity to the message while keeping confidentiality. The receiver can then verify that the sender has complied with the protocol. If the receiver chooses to report, the report will go to a moderator who will judge the validity of the report [13].

AEAD achieves 2 properties: sender binding and receiver binding [15].

- sender binding: the sender cannot cryptographically forge a signature that can evade the moderator's verification
- receiver binding: the receiver cannot cryptographically forge the sender's signature to frame them for content violation

Analysis, Potential Use, and Problems:

Message franking can ensure that both the sender and the receiver end are held accountable for their actions, and malicious users will not be able to easily bypass the moderation system. While ensuring the confidentiality of the content, mechanism like message franking is needed to safeguard the reporting system of E2EE platforms. Verifying user behaviors and establishing accountability is an important aspect of content moderation. Platforms should utilize this cryptographic scheme and develop a system to keep potential abusive users in check.

Future Recommendation

We hope that by now you have a good sense of what are the possible technical infrastructures to combat the problem of content moderation in E2EE platforms. By no means are the technologies comprehensive, but they serve as a guide for future research to build on. We urge platforms to invest in internal research and collaboration with others in the industry and academia to find better ways to deal with content moderation. Take action and start designing and implementing privacy-preserving content moderation.

Due to the complicated nature of each platform's tech stack and their varied capacity to implement the array of technical solutions, we will not recommend a single technical solution as superior or the most probable. Instead, We urge companies to take a proactive approach and consider investing in the below 3 areas of technical possibilities to develop the most applicable solution for their use case.

01 | **Allow Public Verifiability**

One of the most effective ways for change is to be transparent and allows the public to verify platform behaviors. Through publicly verifiable hashing, the potential abuse of client-side scanning can be minimized. Platforms should ensure that PHM systems have a narrow scope, operate transparently, and are accountable to the people they affect

03 | **Hold Malicious Users Accountable**

Malicious users need to be held accountable to protect vulnerable groups on the platforms. Cryptographic techniques such as message traceback enable the platform to tackle the source of the problem and take the necessary next steps.

04 | **Decentralize content moderation**

Distribute content moderation to client sides, so that servers won't even see the content. This idea emcompasses technologies like public veriafiable percetual hashing, client-side predictive models, etc.

References

1. Digital Live Seminar talk by Armin Namavari. https://vod.video.cornell.edu/media/Digital%20Life%20Seminar%20%7C%20Armin%20Namavari/1_cu2sco9o/175727441
2. Mayana Pereira, Rahul Dodhia, Hyrum Anderson, and Richard Brown. 2020. Metadata-based detection of child sexual abuse material. arXiv preprint arXiv:2010.02387 (2020).
3. Sarah Scheffler, Anunay Kulshrestha, and Jonathan Mayer. 2023. https://eprint.iacr.org/2023/029.pdf
4. NCMEC. Child Sexual Abuse Material. https://www.missingkids.org/theissues/csam
5. PhotoDNA, Microsoft. https://www.microsoft.com/en-us/photodna.
6. Content Safety API, Google. https://protectingchildren.google/tools-for-partners/#learn-about-our-tools.
7. PSI, Apple. https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf
8. Paul Grubbs, Arasu Arun, Ye Zhang, Joseph Bonneau, and Michael Walfish. Zero-Knowledge Middleboxes. 2022. https://www.usenix.org/system/files/sec22fall_grubbs.pdf
9. Zero-Knowledge Proofs. https://ethereum.org/en/zero-knowledge-proofs/
10. Imtiyazuddin Shaik, Nitesh Emmadi, Harshal Tupsamudre, Harika Narumanchi, and Rajan Mindigal Alasingara Bhattachar. 2021. Privacy Preserving Machine Learning for Malicious URL Detection. In International Conference on Database and Expert Systems Applications. Springer, 31–41.
11. Carey Shenkman, Dhanaraj Thakur, Emma Llansó. Do You See What I See? Capabilities and Limits of Automated Multimedia Content Analysis. 2021. https://cdt.org/wp-content/uploads/2021/05/2021-05-18-Do-You-See-What-I-See-Capabilities-Limits-of-Automated-Multimedia-Content-Analysis-Full-Report-2033-FINAL.pdf
12. Nirvan Tyagi, Ian Miers, and Thomas Ristenpart. Traceback for End-to-End Encrypted Messaging. 2019. https://eprint.iacr.org/2019/981.pdf
13. Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message Franking via Committing Authenticated Encryption. 2017. https://eprint.iacr.org/2017/664.pdf
14. Messenger Secret Conversations Technical Whitepaper. Facebook. 2017. https://about.fb.com/wp-content/uploads/2016/07/messenger-secret-conversations-technical-whitepaper.pdf
15. Sarah Scheffler, Jonathan Mayer. SoK: Content Moderation for End-to-End Encryption. 2023. https://arxiv.org/pdf/2303.03979.pdf

Contact

HNG in There team
Novia Wu: xw637@cornell.edu
Haitong Lin: hl929@cornell.edu
Grace Nho: en268@cornell.edu