Poster: Committee Moderation on Encrypted Messaging Platforms

Alistair Pattison
University of Minnesota and Carleton College
pattisona@carleton.edu

Nicholas Hopper
University of Minnesota
hoppernj@umn.edu

I. Introduction

Over the past decade, the increased prevalence of mobile computing and a growing desire for privacy have lead to a surge in the use of encrypted messaging services like WhatsApp, Facebook Messenger, and Signal. The deniability, anonymity, and security provided by these services are crucial to their widespread adoption, but by construction, these properties make it impossible to hold users accountable for the messages they send. With no accountability, these platforms are ripe for abuse, the spread of misinformation [1], and election tampering [2]. WhatsApp group chats have been used to spread misinformation that has influenced the Brazilian presidential election [3] and even incited the murder of an innocent woman in India [4]. With no way of identifying or verifying the senders of these messages, little can be done.

These concrete examples and a growing realization that complete anonymity and deniability has consequences have lead to a standoff between governments and messaging services [5], [6]. Is there a middle ground that balances accountability with privacy?

Previous works have addressed this concern by allowing a moderator to verify the identity of a message's sender if a message is reported; if not reported, messages maintain all security guarantees. However, all known protocols concentrate the power of oversight and moderation to a single individual. This is undesirable.

Using primitives from threshold cryptography, this work extends the message-reporting protocol Hecate [7] to a setting in which a group of moderators share the responsibility of determining whether a message requires moderating action. In this work, we provide examples of why this distributed moderation is desirable, describe its functionality, implement the new protocol in Rust, and benchmark its performance.

II. Previous Works

One obvious road to accountability is to to have everyone sign their messages, but this completely destroys the deniability property of the underlying encrypted messaging platform. This is an important feature in many use cases.

Tyagi, Grubbs, Len, *et al.* [8] addressed this problem by using a cryptographic primitive that allows signatures to be verified only by one person who is chosen at the time of signing. By making the designated verifier a trusted third party (for example, law enforcement or a school principal) and attaching to each message a zero-knowledge proof that the signature is valid, users can be confident that abusive messengers can be held accountable for the messages they send while still preserving deniability against everyone else. However, this protocol uses heavy crypto machinery and is quite expensive.

More recent work by Issa, Alhaddad, and Varia [7] introduced a protocol called Hecate that provides the same security guarantees as Tyagi, Grubbs, Len, *et al.* [8] but is significantly cheaper in terms of the number of invocations of cryptographic primitives. It works by distributing "tokens" containing encryptions of senders' identity to the users of a system. These tokens are cryptographically bound to messages such that if the message is reported, the moderator can decrypt the token to recover and verify the identity of the original sender. To everyone else, the token provides no information.

However, users of the system must trust the moderator to singlehandedly deem whether the message requires intervention and carry out the necessary action. By introducing threshold encryption and a committee of n moderators, we modify Hecate to alleviate this issue. Following Hecate's tradition of naming protocols after greek mythology, we call our new protocol Cerberus as a reference to the multi-headed dog that guards the underworld.

III. The Protocol

In the Hecate protocol, every message sent requires a unique token obtained from the moderator in advance. Each token contains (among other things) an encryption of the message-sender's identity,

$$x_1 \coloneqq \textsc{Enc}(id_{src}, pk_{mod}), \tag{1}$$

as well as a single-use ephemeral key pair (pk_{eph}, sk_{eph}). The token also includes a signature σ_{mod} made with the moderators private key that binds the key pair to x_1.

When a user sends a message, she consumes a token and attaches the signature

$$\sigma_{src} = \textsc{Sign}_{sk_{eph}}(x_2) \quad \text{where} \quad x_2 \coloneqq x_1 \oplus H(m) \tag{2}$$

to the message along with the original moderator token. The signature binds x_1 to the sent message, and this metadata is carried with the message throughout its entire forwarding

This work was supported by NSF grant 1814753.

chain. If the message is ever reported, the moderator decrypts x_1 with his private key to obtain the original sender's identity. We direct readers to Issa, Alhaddad, and Varia [7] for a richer description of the protocol and its security.

In our protocol, Cerberus, we modify the process by which x_1 is created and decrypted. Instead of 1 moderator, there are n, and k of them must cooperate to recover id_{src} from x_1. (These k and n values are tunable parameters.) The token-creation process is described below using Elgamal threshold encryption over a group G of order q with generator g. The moderators' public encryption key is pk_{mod}, and the private key is divided into shares $s_1, \ldots, s_n$ using a Shamir sharing scheme [9] and distributed to the moderators. Signatures are done with the FROST [10] signature algorithm, although different threshold schemes could be substituted.

function CREATETOKEN(id_{usr})
- Generate $r \leftarrow_{\$} \mathbb{Z}_q$ and an an ephemeral keypair (pk_{eph}, sk_{eph}).
- Compute $x_1 \coloneqq \langle g^r,\ id_{usr} \oplus H(pk_{mod}^r) \rangle$.
- Package x_1 into a token and send a signature request to each moderator along with the randomness r.
- Each moderator verifies that x_1 is a valid encryption of id_{src} with the provided randomness and returns their signature share on the token.
- Once sufficient responses are receive, combine the signature shares into a valid Schnorr signature σ_{mod} and distribute the token containing x_1 and the ephemeral keypair.

From here, the token is identical to one in the Hecate protocol [7], and the message is sent and processed as described in that paper. To report a message, a user uploads the message m and its metadata to a coordinator who sends out requests for decryption to each moderator. If a moderator believes that a message requires intervention, they respond with a decryption share d_i that serves as a vote that the message should be acted upon. If more than k decryption shares are received, one can recover the identity of the sender, id_{src}; if there are insufficient responses, id_{src} remains hidden.

function HANDLEREPORT(m, x_1)
- A coordinator sends a request to all n moderators containing the reported message m and the encrypted id, $x_1 = \langle c_1, c_2 \rangle$.
- Each moderator verifies the signatures and the well-formedness of the report. If the moderator believes that the message is worth acting upon, she responds with the decryption share $d_i \coloneqq c_1^{s_i}$.
- If enough decryption shares are received, the coordinator multiplies together $k+1$ decryption shares with appropriate Lagrange coefficients $\lambda_i \coloneqq \prod_{j \neq i} \frac{j}{j-i}$ to obtain

$$id_{usr} = H\left(\prod_i d_i^{\lambda_i}\right) \oplus c_2. \tag{3}$$

IV. BENCHMARKS

Issa, Alhaddad, and Varia [7] includes an implementation and benchmark of the whole message cycle (token creation, message sending, message receipt, and message reporting), so we focus on the modified portions of the protocol: token creation and message reporting. We implement these steps in Rust and run each party in a separate linux container communicating over HTTP. We do not have complete benchmark results yet, but we will for the final version. Code is available on github [11].

REFERENCES

[1] E. Dwoskin and A. Gowen, "On whatsapp, fake news is fast and can be fatal," *Washington Post*, Jul. 23, 2018. [Online]. Available: https://www.washingtonpost.com/business/economy/on-whatsapp-fake-news-is-fast-and-can-be-fatal/2018/07/23/a2dd7112-8ebf-11e8-bcd5-9d911c784c38_story.html (visited on 04/04/2023).

[2] S. Woolley and P. Howard, "Computational propaganda worldwide: Executive summary," Oxford University, Working Paper 2017.11, 2017. [Online]. Available: https://demtech.oii.ox.ac.uk/wp-content/uploads/sites/12/2017/06/Casestudies-ExecutiveSummary.pdf.

[3] L. Belli, "Opinion: Whatsapp skewed brazilian election, showing social medias danger to democracy," *PBS*, Dec. 5, 2018. [Online]. Available: https://www.pbs.org/newshour/science/whatsapp-skewed-brazilian-election-showing-social-medias-danger-to-democracy (visited on 04/04/2023).

[4] V. Goel, S. Raj, and P. Ravichandran, "How whatsapp leads mobs to murder in india," *The New York Times*, Jul. 18, 2018. [Online]. Available: https://nyti.ms/383uZ59 (visited on 04/04/2023).

[5] E. Woollacott, "Iran shuts down whatsapp and instagram as protests spread," *Forbes*, Sep. 22, 2022. [Online]. Available: https://www.forbes.com/sites/emmawoollacott/2022/09/22/iran-shuts-down-whatsapp-and-instagram-as-protests-spread/ (visited on 04/04/2023).

[6] S. McCallum and C. Vallance, "Whatsapp: Rather be blocked in uk than weaken security," *BBC*, Apr. 9, 2023. [Online]. Available: https://www.bbc.com/news/technology-64863448 (visited on 04/04/2023).

[7] R. Issa, N. Alhaddad, and M. Varia, "Hecate: Abuse reporting in secure messengers with sealed sender," in *31st USENIX Security Symposium (USENIX Security '22)*, USENIX Association, 2022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/issa.

[8] N. Tyagi, P. Grubbs, J. Len, I. Miers, and T. Ristenpart, *Asymmetric message franking: Content moderation for metadata-private end-to-end encryption*, Cryptology ePrint Archive, Paper 2019/565, 2019. [Online]. Available: https://eprint.iacr.org/2019/565.

[9] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979, ISSN: 0001-0782. DOI: 10.1145/359168.359176. [Online]. Available: https://doi.org/10.1145/359168.359176.

[10] C. Komlo and I. Goldberg, *Frost: Flexible round-optimized schnorr threshold signatures*, Cryptology ePrint Archive, Paper 2020/852, 2020. [Online]. Available: https://eprint.iacr.org/2020/852.

[11] A. Pattison, *Cerberus*, github, 2023. [Online]. Available: http://github.com/alipatti/cerberus.