Wispy: Haunting zk-Promises

Emma Shroyer
University of Maryland
College Park, Maryland
Email: eshroyer@umd.edu

***Abstract*—Honest communication is often hindered by fear of judgment or reprisal, which can discourage individuals from sharing ideas within their communities. For example, junior faculty may feel intimidated by senior or tenured faculty and not share their ideas for fear of judgment or loss of employment. When given the option of anonymity, the fear of consequences decreases and ideas can be shared. However, the freedom of anonymity also introduces challenges for maintaining respectful discourse and enforcing community values.**

Building on prior work with zk-Promises, we introduce Wispy, a protocol for anonymous messaging, pseudonymous messaging, and anonymous polling. We also describe mechanisms for secure cryptographic moderation. In this work, we extend the protocol to Signal, a widely used end-to-end encrypted messaging platform. By layering our protocol atop Signal, we preserve Signal's own privacy guarantees while enabling new forms of safe, anonymous interaction—enabling users to contribute without compromising security or community standards.

1. Introduction

Anonymity offers a space for honest expression and connecting with others with minimal stakes or judgment [1]. Social media platforms and forums usually implement some form of anonymity at scale. This anonymity usually is linked to a phone number and takes the form of pseudonyms or fake accounts [2]. However, anonymity is often used in environments where users engage with unfamiliar or untrusted parties-friends, family, and colleagues rarely use anonymity. Trust and pre-existing knowledge of others usually negate anonymity usage. However, deviating from the norm within these groups may have more perceived consequences or judgment than an online community.

Professional settings in academia and business often have complex relationships and systems that can prevent open and honest communication. Lack of confidence, job security, and interpersonal relationships can make it difficult for employees to speak openly [3]. Prior research has shown that employees are often reluctant to speak up about projects, colleague performance, and policies [4]. Anonymity would provide a neutral setting where ideas can be considered without fear of judgment or repercussions. In the past, schemes like suggestion boxes have tried to redress these feelings. However, these may not be shared with other employees and may not be truly anonymous.

Many workplaces now use instant messaging platforms like Slack and WhatsApp to communicate in professional or ad hoc contexts [5]. These platforms could be extended to provide more support for free expression. They could support anonymity in groups and anonymous polling.

Even if these platforms provide this feature, they will need a method of moderation. Professional and social settings operate under shared norms, and systems that permit anonymous participation must also provide accountability. Without accountability, a group's functionality and culture can erode. On the other side, protections must be in place to prevent administrators from abusing moderation features.

In this work, we present Wispy, a system for anonymous messaging, pseudonymous messaging, and anonymous polling for groups within Signal, a widely used end-to-end encrypted messaging platform. Building on prior work on zk-promises, we introduce a cryptographically verifiable moderation mechanism for anonymity in Signal group chats. Our system preserves privacy while enabling accountability, and is designed with security, usability, and real-world deployment in mind. By extending Signal in this way, we contribute a practical tool for enhancing trust, participation, and safety in both professional and social group communication.

2. Related Work

End-to-end encrypted messaging provides security and privacy for users if implemented correctly. End-to-end encryption means that only the intended recipients should be able to see the plaintext of the messages being sent. An eavesdropper should not be able to infer message contents, but they can learn metadata. This is not protection against an active attacker. Some end-to-end encrypted messaging platforms, like Signal and WhatsApp, have the benefits of widespread adoption [6] [7].

Signal is particularly committed to maximizing privacy for users. Metadata, such as membership or group names, can reveal hints about underlying activity, weakening privacy guarantees. Work from 2020 introduced an improved group administration protocol within Signal that preserves privacy [8]. In this work, the authors propose a method that

allows group administrators to securely add new members without revealing group data or metadata to Signal [9].

There is also Signal's Sealed Sender protocol to ensure sender privacy using encrypted envelopes that hide sender metadata from Signal [10]. In this protocol, sender identities are hidden from Signal. Signal does need to know the recipient to deliver the message; however, the server does not know who is messaging whom [11]. Sealed Sender is now the default for all messages sent using Signal. Signal's existing infrastructure, including Sealed Sender for sender anonymity and its usability features, like reactions, make it a good candidate to extend anonymous moderation to.

There has been other work in anonymous communication and moderation. Namavari et al. provides a hierarchial governanace model for managing interactions in an end-to-end encrypted platform [12]. However, their MlsGov messaging system does not extend to the anonymous setting. Hecate moderates and offers source tracking for end-to-end encrypted messaging [13]. Hecate's protocol has every client authenticating the moderator to get a token to consume when sending anonymously. However, reports of malicious behavior will deanonymize users to the moderator.

2.1. zk-Promises

To enable anonymous moderation, among other features, we build on the cryptographic framework of *zk-Promises* [14]. zk-Promises enables advanced moderation logic while providing confidentiality, integrity, and unlinkability. zk-Promises describes a generic system for manipulating objects via methods that produce callbacks. These callbacks are posted to a bulletin where they can later be called, ingested, and applied to a user's object. It guarantees that a user cannot maliciously evade updating their state.

2.1.1. zk-Objects. A *zk-object* stores client state. This can include account data, reputation scores, or ban status. A zk-object can be updated by executing methods that modify the client's state. An zk-object includes a nonce that is revealed on update, preventing replay of stale data. Updates to zk-objects are performed using zero-knowledge proof showing that the user has been following the protocol without linking the user's private information.

2.1.2. Bulletin. These objects must be able to maintain global visibility where state can be maintained. This global view is done with a global append-only log called the *bulletin*. The bulletin stores cryptographically hidden commitments to objects. In zk-Promises' implementation the bulletin is an append-only Merkle tree, supporting efficient zero-knowledge set membership proofs. When users update their zk-objects, they query the bulletin to proccess invoked callbacks, which are described below.

2.1.3. Callbacks. A *callback* is a function in zk-promises that modifies a zk-object and is managed by a separate zk-object called the *callback manager*. The bulletin enables callback ingestion after a callback is invoked. A callback has the following life cycle:

1) **Callback Creation**: When a user performs an action, e.g., sends a message, their zk-object generates a callback ($\texttt{tik}$) and appends them to the bulletin along with cryptographically signed metadata. The proof of callback creation (π_{msg}) ensures that the callback conforms to the system's policies:
$$\pi_{\text{msg}} = \text{ZK-SNARK}(\text{obj}, \texttt{tik}, \Phi_{\text{msg}}). \quad (1)$$
2) **Callback Invocation**: Administrators invoke callbacks by posting $(\texttt{tik}, \text{args})$ to the bulletin, where args encodes moderation actions such as banning users or updating reputation.
3) **Callback Ingestion**: Users scan the bulletin to detect invoked callbacks and update their zk-objects. Scans occur periodically, but non-membership proofs ensure that callbacks cannot be skipped when scans so occur. The update is validated using a zero-knowledge proof:
$$\pi_{\text{msg}} = \text{ZK-SNARK}(\Phi_{\text{msg}}(\text{obj}, \text{obj}', \texttt{tik}, \text{args})). \quad (2)$$

2.1.4. Security Properties of zk-Promises. zk-Promises provides the following informal cryptographic guarantees:

1) **Confidentiality:** A zk-object's contents are only visible to the owner; however, function callers may learn some information based on the call they make on the object.
2) **Obliviousness:** Updating a zk-object does not reveal which object was updated and multiple updates cannot be linked.
3) **Integrity:** All updates must be applied in accordance to programmed methods and by authorized entities.
4) **Atomicity:** There is only one valid zk-object at a given time and it cannot be rolled back.

3. Overview

In this section, we review the security goals of our system, provide a high-level overview of the construction, and summarize key features that have been implemented.

3.1. Security Goals

We consider a group messaging system where users communicate through an end-to-end encrypted platform such as Signal. Group members may post normally, anonymously, pseudonymously, or vote in anonymous polls.

In our setting, we consider a potentially malicious server and potentially malicious clients. As in zk-Promises, we require at least two honest, unbanned users to ensure k-anonymity where k=2 [14]. We define a ban in this context as the inability to post anonymously in the group; even if a user is banned, they may still interact with the group as normal. This anonymity guarantee fails should one user become banned, either through honest or malicious action by the moderator. Additionally, a malicious moderator could

strategically ban honest users, breaking the assumption of k k-anonymity and potentially leaking information about the remaining users.

Signal's Sealed Sender protocol ensures that even Signal cannot determine the sender of a message, the members of a group, or most metadata. However, our implementation assumes a more relaxed deployment context. In particular, our system introduces a proxy server that mediates anonymous communication and moderation. This proxy is not operated by Signal and, therefore, must not be explicitly trusted not to violate user anonymity. To mitigate this risk, we assume that all communication done between the client and the proxy server is through anonymous channels as not to deanonymize the user over the network.

Our threat model does not hold against stylometric or social engineering attacks. Adversaries leveraging may exploit linguistic patterns, semantic identifiers, or repeated phrasing to deanonymize users [15]. For example, if a user consistently omits punctuation when posting under their real identity and continues this habit anonymously, authorship may be inferred. To reduce the threat of stylometric attacks, users could utilize an LLM to obfuscate writing characteristics.

Social engineering attacks present another challenge. Even with LLMs, underlying content and core ideas could still indicate authorship. Adversaries with prior knowledge of group members, such as personal opinions, may attempt to link anonymous posts to individual users. However, this is out of the scope of our system.

3.2. Construction

An overview of our construction is shown in Figure 1. The figure omits the initial setup phase, during which the user initiates an interaction and registers with the server. Note that the server does not ask for the user's identifying phone number. Instead it simply requires the unique group ID to prove membership. The server never knows which group member it is talking to.

Later, when the user wishes to anonymously message the the group, the user communicates to the server to prove they reviewed the bulletin and processed all outstanding callbacks. If the server verifies the user's proof, it forwards a message to Signal to be posted in the group. The server also monitors for responses from the group to update callbacks as needed. Throughout this process, standard group communication continues unaffected. Meanwhile, normal group communication can occur.

To enable anonymity in this setting, the server must be able to post within the group. Therefore, an additional "$n+1$" Signal account is added to the group. This account acts as a proxy to forward messages from the server to Signal. At this time, it is up to the administrator to acquire and share this $n+1$ number with the server. It is also important that no one but the server is posting under this number within the group.

Our construction utilizes a third-party library, signal-cli-client [16]. Messages are posted to a daemon communicating with the Signal server using the provided JSON-RPC API. While this interface is more limited than official Signal interfaces, it exposes several features useful for our system:

1) **Timestamps:** Signal attaches unique timestamps to each message in the group. In our construction, we define the posting time in the callback, allowing moderators to identify which message should lead to updated state for the author.
2) **Group ID:** Signal assigns a unique group ID to each group chat. This value is required for message delivery and is also used to prove cryptographically group membership to the server.
3) **Emoji Reactions:** Signal allows users to respond directly to a post using emojis. This feature is the primary method by which reputation scoring is calculated. For example, multiple thumbs-down emoji reaction may signal disagreement with a post.

3.3. Features

At the current time, we have implemented the following features have been added onto the Signal messaging platform:

1) **Anonymous Posting:** Users can submit anonymous messages to a Signal group by sending a command containing the plaintext message, a group ID, and a valid cryptographic proof to our server. The server broadcasts the message to the group using the JSON-RPC interface provided by an open-source Signal CLI implementation [16]. Members of the group, including the sender, can observe that the $n+1$-th user has posted without the original authorship being revealed.
2) **Pseudonymous Posting:** Users may post under a recurring alias or pseudonym by sending a command containing the message, the group ID, valid cryptographic proof, and a pseudonym ID to the server. The server maps the pseudonym ID to a unique name, , e.g., 'Fuzzy Bunny', provided by a rust name library [17]. Later posts using the same pseudonym ID will provide linkability between posts and unlinkability between the pseudonym and the author's real identity. An example of pseudonym usage is shown in Figure 2.
3) **Anonymous Polling:** Users can participate in group polls while preserving anonymity. Each submission to the server includes the group ID, unique poll ID, the user's vote. Users could trivially change their vote by resending the same submission to the server. The server would only need to update the vote for the poll ID. Without exposing individual voter identities. Votes are tallied and revealed once everyone has voted or the time frame has passed. A prototype of anonymous voting is also displayed in Figure 2.
4) **Moderation and Banning:** Group administrators and even other users will have the ability to moderate anonymous content. Each anonymous or pseudonymous message includes a unique timestamp, which can be referenced by an administrator to initiate a ban on

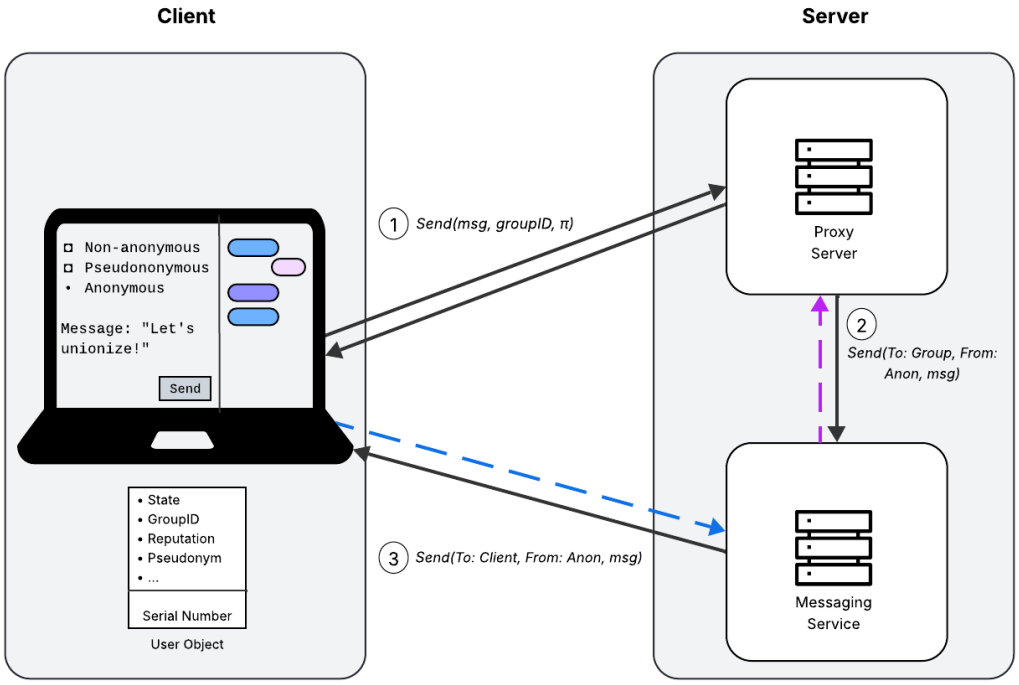

Figure 1. **Protocol Design Using zk-Promises.** (1) To send an anonymous message, the user submits the message, group identifier, and a cryptographic proof to the proxy server. (2) The proxy server verifies the proof and forwards the message to the messaging service using a designated $(n+1)$-th account. (3) The messaging service then delivers the message to the target group. Non-anonymous messages, indicated by the blue dashed arrow, follow a different path: they are sent directly to the messaging service, which then optionally contacts the proxy server (acting as a client) for moderation or logging purposes. The messaging service can also communicate with the proxy server to initiate user bans or reputation updates, as represented by the purple dashed arrow.

the associated sender. This ban is currently defined as the inability to post anonymously using this system. The server processes these calls for bans and posts a callback, preventing further anonymous messages from the offending user. Currently, an implementation of administrator registration and authentication is still ongoing work.

As an alternative to administrator moderation, we also support a reputation system. In this construction, messages receiving sufficient negative feedback, e.g., thumbs-down emoji reactions, will decrease the sender's reputation score. When the score falls below a threshold, an administrator or the server itself can enforce a ban on future anonymous posts from that user.

4. Cryptographic Protocols

In this section, we define primitives for anonymous posting, pseudonymous posting, and anonymous polling:

Notation:

- $(pk, sk) \leftarrow \mathsf{KeyGen}()$: Key generation
- PRF: Pseudorandom function
- π: Zero-knowledge proof
- n_k, n_{server}: Random nonces (user-generated and server-provided)

Anonymous Posting:
Client computation:

1. $(pk, sk) \leftarrow \mathsf{KeyGen}()$
2. $n_k \leftarrow \{0,1\}^\lambda$
3. $alias_k \leftarrow PRF(sk \| n_k)$
4. $\pi_{msg} \leftarrow \mathsf{ZKProofOfPost}(msg, groupID)$
5. $\pi_r \leftarrow \mathsf{ZKProofOfAlias}(alias_k)$
6. *Client* $\rightarrow$ *Server:* $(msg, groupID, \pi_{msg}, alias_k, n_k, \pi_r)$

Pseudononymous Posting:
Client computation:

1. $(pk, sk) \leftarrow \mathsf{KeyGen}()$
2. $n_k \leftarrow \{0,1\}^\lambda$ -chosen once and reused
3. $alias_k \leftarrow PRF(sk \| n_k)$
4. $\pi_{msg} \leftarrow \mathsf{ZKProofOfPost}(msg, groupID)$
5. $\pi_r \leftarrow \mathsf{ZKProofOfAlias}(alias_k)$
6. *Client* $\rightarrow$ *Server:* $(msg, groupID, \pi_{msg}, alias_k, n_k, \pi_r)$

Unlike anonymous posting where the nonce is used only once, a user can post multiple times under the same

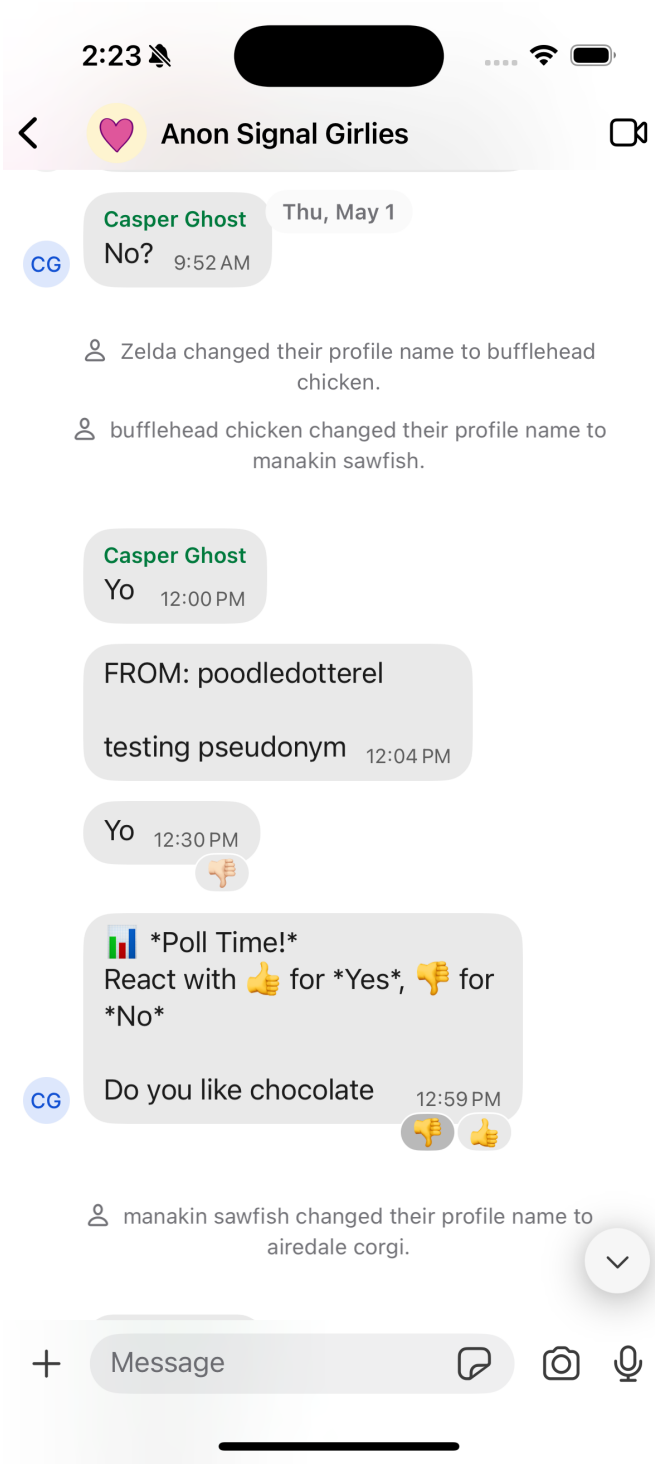

Figure 2. The Signal iOS interface showcasing pseudonyms and anonymous polling. In this example, *Casper Ghost* is the $n+1$-th number that all non-banned users may use to post anonymously.

pseudonym by reusing the same nonce, n_k. Using the same n_k multiple times proves to the server that a user is using that pseudonym. The user stores a list of previously used nonces.

Anonymous Polling:

Server-side:

1. $n_{\text{server}} \leftarrow \{0,1\}^{\lambda}$ -unique to the user
2. *Server* $\rightarrow$ *Client:* (n_{server})

Client computation:

4. $(pk, sk) \leftarrow \mathsf{KeyGen}()$
5. $alias_k \leftarrow PRF(sk \| n_{\text{server}})$
6. $\pi_{vote} \leftarrow \mathsf{ZKProofOfVote}(vote, groupID)$
7. *Client* $\rightarrow$ *Server:* $(vote, groupID, \pi_{vote}, alias_k)$

5. Future Work

This work provides a simple configuration for integrating and moderating anonymous and pseudonymous posting into group chats. Below we describe future work to extend the functionality and usability of the system.

5.1. Configurable Moderation

Social contracts and rules are context-dependent, varying across communities and communication environments. Our current design presents a basic moderation mechanism, but more sophisticated and configurable policies may be needed for different communities. Future work may enable support for more custom and complex moderation mechanisms. With these new features, administrators or users could define and update custom moderation to fit their needs.

For instance, rate-limiting mechanisms could restrict the number of anonymous posts a user could submit in a time period. Time-bound bans could temporarily suspend users' ability to post anonymously, and systems could issue warnings prior to enforcing bans. Rather than revoking a user's ability to post anonymously entirely, groups could selectively ban specific pseudonyms.

Incentive mechanisms could also be introduced. For example, pseudonymous users who consistently post inoffensive material could receive perks, e.g. badges or or increased privileges, such as the ability to vote multiple times.

5.2. Multi-Group Moderation

The protocol may be extended to support moderation across multiple groups. Currently, bans are local to individual group chats. However, many communities, including workplaces, online forums, or academic institutions, are structured into subgroups that share overarching norms and expectations. It may be desirable to enforce consistent moderation policies across multiple groups to prevent harmful behavior from migrating between them. For example,

faculty might have group chats based on research area. In such cases, misconduct in one group should result in consequences across other groups.

One possible extension would allow groups to share a common bulletin, while users retain their private zk-objects. In this construction, callbacks would be posted on the same bulletin board, so changes in status would affect all participating groups. The system would enable coordinated enforcement of group policies while still maintaining unlinkability and anonymity.

5.3. Claimable Messages

There are scenarios in which users may wish to de-anonymize their own posts later in time—for example, to claim authorship of a well-received anonymous message or to prevent misattribution or messages. The system already partially supports this functionality. Users can prove authorship to the server by resubmitting the nonce corresponding to the post or posts in question. Remaining functionality then requires proof of authorship be shown to the group. The server cannot do this directly since it does not know the identity of the user. Trivially, the user could post in the group a proof claiming authorship and the server verifies the proof. Future work may further simplify and formalize this process, ensuring that authorship claims are user-friendly and maintain overall system privacy.

5.4. Usability Study

While this work focuses on the design and cryptography of anonymous group communication and moderation, future work should explore its usability in real-world settings. A usability study could evaluate whether users find the features intuitive, whether the moderation mechanisms are perceived as effective, and how anonymity affects group dynamics. Such a study could reveal a need for additional features or changes that could lead to system adoption.

6. Conclusion

In this work, we extend the Signal messaging platform with support for anonymous messaging, pseudonymous identities, and anonymous polling. All features not natively supported by Signal or other mainstream secure messengers. Wispy leverages Signal's existing infrastructure, including Sealed Sender for sender anonymity and it's usability features, like reactions, preserving compatibility with the original Signal system. To ensure accountability, we also integrate zk-Promises, a cryptographic moderation system. Together, these components show how regulated anonymous communication can be integrated with a widely deployed platform while still preserving privacy guarantees.

Acknowledgments

The author would like to thank the other people on this project: Rachel, Gabe, Ian, Hari, and Oliwia.

References

[1] C. Guo and K. Caine, "Anonymity, User Engagement, Quality, and Trolling on Q&A Sites," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW1, pp. 1–27, Apr. 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3449215

[2] "Anonymity and identity shielding | eSafety Commissioner." [Online]. Available: https://www.esafety.gov.au/industry/tech-trends-and-challenges/anonymity

[3] F. J. Milliken, E. W. Morrison, and P. F. Hewlin, "An Exploratory Study of Employee Silence: Issues that Employees Don't Communicate Upward and Why," *Journal of Management Studies*, vol. 40, no. 6, pp. 1453–1476, 2003, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-6486.00387. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-6486.00387

[4] S. Burnett and L. Illingworth, "Anonymous knowledge sharing in a virtual environment: a preliminary investigation," *Knowledge and Process Management*, vol. 15, no. 1, pp. 1–11, 2008, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/kpm.294. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/kpm.294

[5] K. Paerata, "The use of workplace instant messaging since covid-19," *Telematics and Informatics Reports*, vol. 10, p. 100063, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772503023000233

[6] Signal Messenger LLC, "Signal: Specifications," https://signal.org/docs/, 2016.

[7] "About end-to-end encryption | WhatsApp Help Center." [Online]. Available: https://faq.whatsapp.com/820124435853543

[8] M. Chase, T. Perrin, and G. Zaverucha, "The Signal Private Group System and Anonymous Credentials Supporting Efficient Verifiable Encryption," 2019, publication info: Published elsewhere. Major revision. ACM CCS 2020. [Online]. Available: https://eprint.iacr.org/2019/1416

[9] "Technology Preview: Signal Private Group System." [Online]. Available: https://signal.org/blog/signal-private-group-system/

[10] I. Martiny, G. Kaptchuk, A. J. Aviv, D. S. Roche, and E. Wustrow, "Improving signal's sealed sender," in *NDSS*, 2021.

[11] "Technology preview: Sealed sender for Signal." [Online]. Available: https://signal.org/blog/sealed-sender/

[12] A. Namavari, B. Wang, S. Menda, B. Nassi, N. Tyagi, J. Grimmelmann, A. Zhang, and T. Ristenpart, "Private hierarchical governance for encrypted messaging," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 255–255.

[13] R. Issa, N. Alhaddad, and M. Varia, "Hecate: Abuse reporting in secure messengers with sealed sender," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 2335–2352.

[14] M. Shih, M. Rosenberg, H. Kailad, and I. Miers, "zk-promises: Making zero-knowledge objects accept the call for banning and reputation," Cryptology ePrint Archive, 2024.

[15] K. Lagutina, N. Lagutina, E. Boychuk, I. Vorontsova, E. Shliakhtina, O. Belyaeva, I. Paramonov, and P. Demidov, "A Survey on Stylometric Text Features," in *2019 25th Conference of Open Innovations Association (FRUCT)*, Nov. 2019, pp. 184–195, iSSN: 2305-7254. [Online]. Available: https://ieeexplore.ieee.org/document/8981504/

[16] S. Scheibner, "AsamK/signal-cli," May 2025, original-date: 2015-05-11T10:49:42Z. [Online]. Available: https://github.com/AsamK/signal-cli

[17] "petname - crates.io: Rust Package Registry," Apr. 2024. [Online]. Available: https://crates.io/crates/petname