Towards A Secure Platform for Political Activism

Yaseen Alkhafaji, William Hu, Mihir Khambete, Veronica Tang

May 20, 2021

Contents

1 Abstract

Large, in-person public demonstrations require a significant amount of communication and coordination. However, protests can be dangerous for the people involved, and we believe protests have unique security requirements which are unmet by existing messaging apps today.

We also believe that peaceful protests are necessary for a democracy to function. Oftentimes, established parties or people in power create a dominant discourse that excludes or marginalizes the voices of others, and democracies, in particular, rely on majority rule. Public demonstrations empower minority or dissident groups to be seen and heard.

Therefore, in this paper, we propose a system design for a secure application that facilitates the coordination of peaceful protests. Our system allows for pseudonymity of users within event-specific groups on the app and allows community members to hold bad actors accountable. We present details on how this app could be implemented, how it protects user security interests, and the limitations of our current design.

2 Background

Before we began designing our own system, we felt that it was necessary to examine strategies traditionally used by protesters in organizing large public demonstrations as well as the technologies and applications that protesters have gravitated towards in the past. This examination of our application's precedents informs the functionalities we decided on for our system, as well as the security features that we believed protesters would prioritize.

2.1 Traditional Protest Organization

First of all, guides on organizing protests emphasize the importance of broadcast messages. There is a great deal of logistical information that protesters must have access to on the day of a protest (for example, designated parking, where to find water vendors, how to reach medical personnel, the rights of protesters in their city or region, etc.) [Qui20]. The traditional methods of disseminating this information, however, are concerning. One method that protest coordinators have used is email, because it allows them to reach large numbers of people in a single transmission [kue21]. Another approach that protesters have relied on in the past involves assigning each person a few other phone numbers to call upon receiving any news that requires broadcast, so that people can call each other and spread the necessary information in a "gossip"-like system [kue21]. But, neither of these methods protect the identity of the protesters (emails and phone numbers can be linked to the people behind them), and the second method can take a little longer.

For protesters, the matter of protecting identity goes beyond the individual. Taking steps to protect your identity is not only about protecting yourself, but also necessary to protect those who may be more vulnerable: people who are undocumented, have a criminal record, or have an underlying health condition that would make arrest life threatening [GN20]. If law enforcement confiscates your phone, they may be able to identify other individuals through your communication records. And so, when designing our application, we set out with the goal of enabling broadcast messages through a system that preserves pseudonymity.

2.2 Bluetooth

Some articles about protesting safely in the age of surveillance actually suggest that protesters should leave their phone at home entirely or keep it off as much as possible. Naturally, however, neither option is conducive to effective communication. The concerns that inspire these approaches are unfortunately valid, though: law enforcement could subpoena mobile carriers and force them to reveal data about which devices connected to cell towers near the location of a protest during the time of protest [GN20]. U.S. police have also previously used IMSI catchers that impersonate cell towers and trick nearby cell phones into connecting with them [GN20].

These surveillance tactics, and the fear that governments may attempt to shut down communications entirely, have made Bluetooth an appealing medium of communication for protesters. However, we chose not to introduce Bluetooth into our own design due to its numerous limitations. For example, the range of a Bluetooth device is limited by the device's power. Bluetooth devices are classified into one of three classes depending on the maximum power of the device: Class 1 (100mW), Class 2 (2.5mW), or Class 3 (1mW); these have operating ranges of 100 meters, 10 meters, and 1 meter, respectively [blu]. Most smartphones are Class 2 Bluetooth devices, which puts them in the 10 meter range; for a large protest, this short range may be a disadvantage [Pat18]. Additionally, one Bluetooth device can only support up to 8 concurrent connections to that device, which would not be conducive to our goal of facilitating conversation in a group setting [Chu21]. And finally, Bluetooth communication is sensitive to objects in between the devices; in a crowded setting, in which there may be many people standing in between the two parties attempting to communicate, it may be difficult for Bluetooth signals to transmit successfully [Tig19]. For these reasons, we have decided against the use of Bluetooth for our proposed design, which is intended for centralized and moderated discussion between parties.

2.3 Previous Applications Used for Coordinating Protests

We examined the shortcomings and strengths of three messaging platforms that have either been used previously for coordinating protest or pseudonymous communication: Bridgefy, Signal, and Yik Yak.

2.3.1 Bridgefy

Bridgefy is an app that was originally marketed as secure mesh messaging via Bluetooth; its creators initially intended it to be a way of communicating without access to the Internet at concerts and other large or outdoor events. However, it was adopted by protesters in Hong Kong in 2019 as their chosen form of communication - they feared that the government would attempt to shut down communications on a wide scale, which made Bridgefy's Bluetooth-based communication system appealing to protesters. Bridgefy was later adopted by protesters around the world, and it was one of the forms of communication used by protesters during the Black Lives Matter protests in the USA [Marne].

Earlier this year, Lenka Mareková gave a presentation at the Real World Crypto Symposium 2021 on the security (or rather, the complete lack thereof) of Bridgefy [Marne]. Mareková and her team reverse engineered the Bridgefy app, and they discovered that there are significant vulnerabilities: users could be tracked, message confidentiality was not preserved, and since the handshake was not cryptographically authenticated, users could be impersonated as well [Alb19]. The compromised information could then be revealed to other organizations, such as law enforcement. As such, Bridgefy utterly fails to preserve the privacy of protesters and places them in considerable danger.

2.3.2 Signal

Signal is a platform for encrypted messaging that was released in 2014 and developed by a nonprofit organization. Because the messages are encrypted end-to-end, Signal itself cannot see the messages sent by users, let alone law enforcement or national security agencies. This security property has proven useful in the past; in 2016, the US government obtained access to Signal user data through a grand jury subpoena from the Eastern District of Virginia [siga]. However, despite the subpoena, the data Signal was able to provide was incredibly sparse and did not include any information about the content of messages or the composition of groups.

Signal's ability to resist subpoenas inspired the use of end-to-end encryption in our system as well. Nevertheless, the use of phone numbers is still an issue for Signal. While the messages may be protected from authorities, the phone numbers, and hence the identities of users, can still be unmasked. This was part of our rationale for using pseudonymity in our system.

Signal has also recently risen greatly in popularity due to widespread distrust of its primary competitor, WhatsApp. Protestors have always been wary of WhatsApp due to its affiliation with Facebook [Nie20]. However, in a very concerning announcement earlier this year, WhatsApp privacy policy was updated to say that it shared user data with its parent company, Facebook [Duf21]. And so, Signal's nonprofit status has also contributed to its success, which provides a good argument for requiring that a nonprofit organization be in charge of developing our application.

2.3.3 Yik Yak

Yik Yak was an anonymous social media app launched in 2013 that allowed people to create discussion threads and view them within a 5 mile radius. The app was monitored by the community, and users were able to downvote posts that people found offensive. If a post received enough downvotes, it would be removed. However, the privacy policy required a subpoena, court order, or search warrant to identify users who posed a risk. As a result, Yik Yak became criticized for facilitating cyberbullying [Saf17].

Yik Yak's failure underscores our application's need for accountability. In other words, there must be a process to expose bad actors without having to go through complex processes such as subpoenas. And while users' identities should still be protected, usage of our application ought to come with a mechanism for exposing users who exhibit irresponsible, bad, or dangerous behavior, in order to protect other honest users.

3 Our Application's Goals

3.1 Intended Context

Our application focuses specifically on communication during large-scale, in-person protests - public demonstrations that involve marches, parades, picket lines, sit-ins, or street theater. Therefore, our application specifically aims to implement the functionality and security requirements that protesters would need *at the scene* of a physical protest, and it focuses in particular on "group-centric" design. We explicitly did *not* set out to create a secure form of messaging for coordinating protests or activist movements online (for example, users of TikTok inflated attendance expectations last fall for Donald Trump's Tulsa rally [LBF20] - while this *is* an instance of political activism, we want to focus on the type of mass messaging needed at the scene during a large physical protest, instead of this type of pre-emptive coordination). As such, our application allows for the creation of group chats in which people can communicate pseudonymously, but the lifespan of each group is limited and short-term (our group felt that a maximum lifespan of a week would be appropriate, so that admin have time to set up the group chat, publicize details and logistics, follow up with participants afterwards - nearly all guides on organizing protests emphasize the importance of continuing the work that was started at a public demonstration - and still host a multi-day event).

We also recognize that the system we have developed may be applicable to other situations (for example, in organizing unionization or strikes). However, once again, for the sake of narrowing our scope, we have decided to focus specifically on protests.

And finally, because real-life public demonstrations and protests are typically organized by a single individual or a small committee of organizers, our group chats follow a hierarchical structure in which there are "admin" roles that differ from those of other members, all of whom we denote as citizens. Group chat administrators are decided at the time of the group chat's establishment, and they have specific roles and privileges under our system. By default, all members of the group chat are pseudonymous, so citizens do not know the identity of a particular admin, admin do not know the identity of the citizens, and the admin are mutually pseudonymous as well within the chat. There are a maximum of 5000 members in a group.

3.2 Security Policy

After considering other app examples, we were motivated to create a design with the primary goals of encrypting messages end-to-end and protecting the identities of users in a group chat through pseudonymity, while also preventing harassment by limiting bad actors in the system. We decided to limit the scope of the design to group membership, with limited broadcasting functionality, as the design is not intended to serve as an independent and feature-rich messaging platform. This is meant more as an extension to an existing full-fledged messaging platform; our design could be implemented by existing messaging apps that want to include a pseudonymous broadcasting feature.

It is also important to note that we distinguish between the terms **pseudonymous** and **anonymous**. Anonymous broadcasting means that every message in the group could be attributed to any other member in the group with equal probability. Pseudonymous broadcasting is weaker, and simply means that users message under a pseudonym. This makes it so that each participant is distinct, but their identity within the group cannot be linked to their real-life identity. For example, in a pseudonymous group chat, one might be able to see all the messages that have been sent by individual A and know that those messages came from a single, distinct sender; however, that individual's real-life identity cannot be discerned. We chose pseudonymity as opposed to anonymity, in which the senders of messages would be indistinguishable, not only for the purpose of making communication easier and clearer (some people might, for example, need to know the admins of a group chat or be able to distinguish different admins or participants), but also for the sake of accountability. If an individual exhibits dangerous behavior, begins harassing other members of the group chat, or engages in otherwise bad behavior, we want to be able to hold them accountable. Towards that end, we allow the participants of a group chat to unmask the identities of certain individuals (or rather, reveal the phone number of the account behind the pseudonym, since that creates a tie to the person's real-life identity) under a certain set of conditions, which will be elaborated further in which will be elaborated further in Section 4.4, Protocol for Identity Unmasking and Protecting Voting Power.

Furthermore, to constitute a user identity, we use a phone number for each account. This system mirrors that of Signal's which uses phone-based verification for admitting new users into a group. Phone numbers serve as unique identifiers for each user in our system because they are harder to obtain than emails. This makes it difficult for a person to impersonate multiple users, either by spoofing or by flooding a group with many members. The user's secret in a group is the phone number encrypted with a group key. The group key is not known to the central server, and thus the server cannot determine the plaintext phone number. While group members know the group key, they cannot determine users' plaintext phone numbers without the unmasking procedure. We place a strong emphasis on **confidentiality** and **availability** - the identities of users should be preserved, but it should be relatively easy to unmask bad actors.

3.3 Functionality

To implement our design, we support a number of key functions:

1. **broadcast(user, group, message)**: A user sends a message to all other members in a group

2. **join(user, group)**: A user joins a group

3. **leave(user, group)**: A user leaves a group they are currently in

4. **release_share(voter, group, bad_actor)**: A user elects to release shares of a presumed bad actor

Users are able to broadcast messages to groups, join groups, leave them, and unmask a bad actor. To support these functions, we have the following mechanisms. Upon joining a group, a user's identity is encrypted and sent to a server, which then creates secret shares for admins to hold. We represent a user's identity by splitting it into 1 share per admin, in a scheme as in Shamir's Secret Sharing. These shares are distributed to the admins, and the server makes subshares of admin shares to distribute to all members of the group ("citizens"). Admins are double-counted as citizens for simplicity. In particular, for each admin share, one subshare is created for each citizen in the group. If there are K admins and M total users, then from each admin's share, we create $2M$ subshares and distribute 2 to each citizen (which, for simplicity, includes other admins, so that admins can vote with either method). To unmask a bad actor, a certain threshold of admin votes needs to be met, or a certain threshold of user votes must be met. More detail on the scheme can be found in the section on Voting Power.

4 Our System Design

Our system makes use of Signal's Double Ratchet Algorithm, Shamir secret sharing, and public key cryptography in order to implement our end-to-end encryption, accountability system, and group mechanisms.

4.1 End-to-End Encrypted Messaging with Double Ratchet Algorithm

For our application, we chose to implement end-to-end encryption by using Signal's Double Ratchet Algorithm, in which messages are symmetrically encrypted, using new keys for every round of messaging. Under the Double Ratchet Algorithm, two users compute a shared value by doing Diffie-Hellman Key Exchange (such that they take turns using a new Diffie-Hellman private key each time). Next, a Key Derivation Function takes this value as input along with a chain key (which, for our application, will begin initially as the shared group key and then be derived from previous iterations of the Double Ratchet Algorithm for subsequent steps of the process). The Key Derivation Function will then produce a new chain key and message key. The new chain key will be used to generate the next set of keys, and the message key will be the new key used for the current round of messaging. In this way, the Double Ratchet Algorithm manages to change the shared secret key each time a message is sent or received. [sigb]

This algorithm can be broken down into two components - the Diffie-Hellman ratchet steps, and the Key Derivation Function. The Diffie-Hellman ratchet steps can be visually represented as follows:

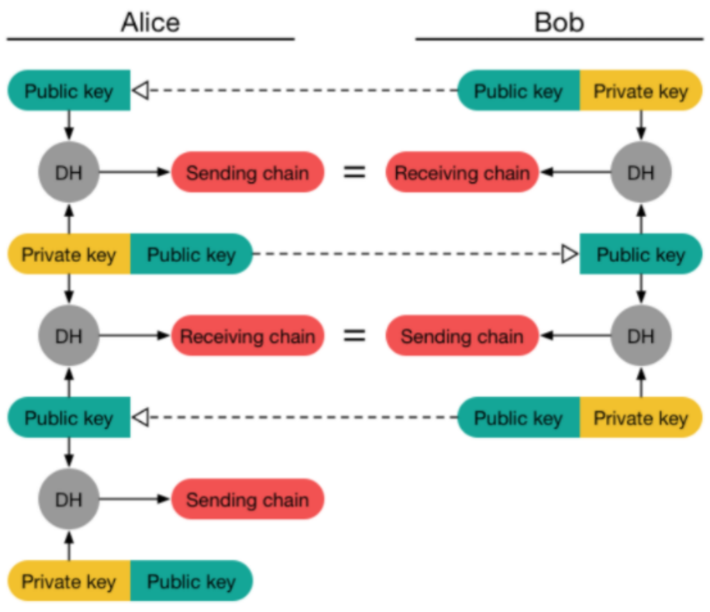

Figure 1: Figure from [sigb]

As shown above, the two parties take turns performing the Diffie-Hellman ratchet steps and introducing new sending chains.

As for the Key Derivation Function, it can be visually represented as follows:

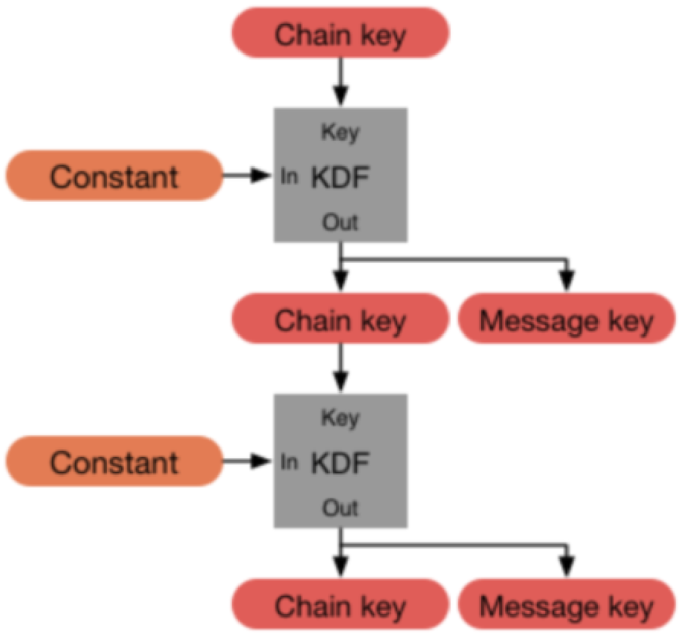

Figure 2: Figure from [sig21]

In the above diagram, the “Constant” values are the ones produced by the Diffie-Hellman ratchet steps.

The Double Ratchet Algorithm is a very cool protocol that Signal actually uses to encrypt messages end-to-end. It comes with a variety of nice properties (for example, forward secrecy and break-in recovery) [sig21], as well as its own method of handling out-of-order messages, skipped keys, fingerprinting, etc., all of which are documented in Signal’s documentation [sigb].

However, the Double Ratchet Algorithm is limited in that it only switches message keys for two people. And because the Double Ratchet Algorithm uses Diffie-Hellman, the newly established message keys are only shared between two people. Signal handles this by treating their group chats like a series of one-on-one messages; if a user sends a message to a group, they are actually sending a one-on-one message to every other member of that group chat under the hood.

For the sake of our application, we do the same thing, which means every distinct pair of members within the group chat have their own, constantly changing message key (a user’s keys change every time they send or receive a message). Also, for our initial chain key, we use a shared group key that is sent to members of the group at the time they join the group chat using public key cryptography (in a system that is reminiscent of Pretty Good Privacy, or rather, PGP - see the section on the Joining Protocol for more details). This shared group key will also be repurposed by our accountability system, as explained in Section 4.3.4, Identity Revelation.

4.2 Enforcing Accountability through Interpolation-Based Shamir's Secret Sharing

We implement accountability through interpolation-based Shamir secret sharing [Sha79]. As a brief review of Shamir secret sharing, an arbitrary secret integer s can be used as the constant term of a $(t-1)$ degree polynomial f whose other coefficients are randomly selected. The holder of the secret distributes shares of the form $(x, f(x))$ to N shareholders, each receiving one share (all x's are nonzero). Any t or more cooperating shareholders can combine their shares using interpolation to determine the unique polynomial f which produces these shares. After knowing f, the secret can be evaluated as $s = f(0)$. In our scheme, participants can collectively unmask the identity of a bad actor since the identity of each user is split into shares and distributed to the participants. The exact details on the number and composition of shares are given in the section on Protecting Voting Power.

4.3 Group Mechanics

4.3.1 Protocol for Establishing a Group

When the group is first established, each admin must generate their own public key/private key pair for exclusive use in the group. By reserving this public key for use within the newly established group, they cannot be linked through this key to another group chat that they might be participating in. They each must also randomly generate a value which will serve as their first private key for the Diffie-Hellman component of the Double Ratchet Algorithm.

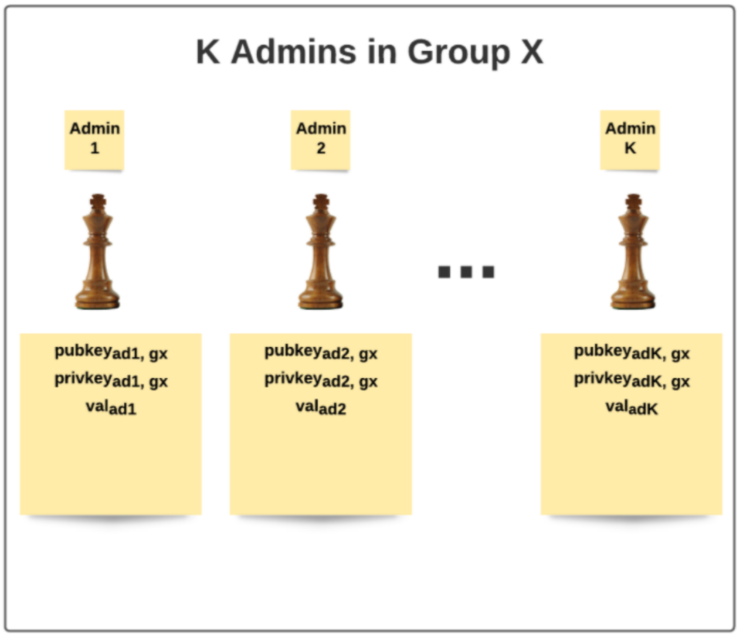

Figure 3: Administrators with the values they generate individually at group instantiation

Next, the admin must host a key signing party in which they digitally sign certificates containing their public key, their pseudonym within the group, and a time span indicating how long the group will last (or rather, how long this certificate will be valid for). Admin must get their respective certificates signed by all other admin. In this way, they validate each other; no one can replace a single admin's public key or attempt to replace one of the admin without having to overhaul the entire cohort of administrators. This key signing party would ideally be held in-person (having the organizers of a large, in-person event meet prior to the actual event hardly seems unreasonable).

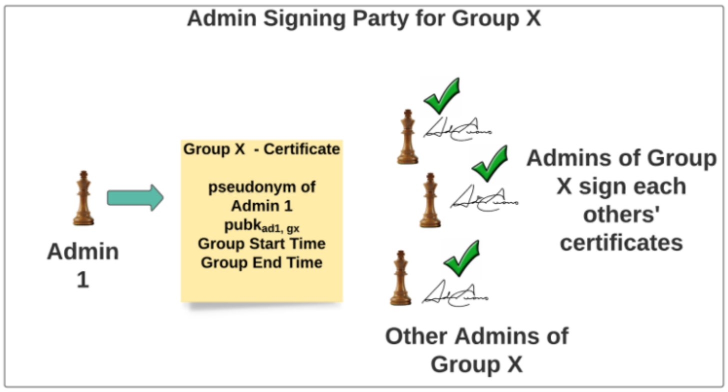

Figure 4: Key Signing Party for the Admin of Group X

We will also provide a hash of each admin's public key to the committee of administrators, who can then text it to intended group members through an alternate channel or publish it on a website somewhere. This allows joining group members to compare these hashes to the hashes of the public admin keys they receive. This is just an optional, additional step that admins may or may not choose to take so that users can sanity check themselves and ensure that they have joined a legitimate (or maybe just the correct) group chat. We chose to use a hash of the public key since public keys tend to be pretty long. And finally, the administrators must randomly generate a shared, symmetric, secret key for the entire group to use, which we will refer to as the SSK.

4.3.2 Protocol for Joining a Group

When a new person joins the group chat (for the sake of this example, let us call them Person A and say that they are joining group chat number 1, which has n admins), they must collaborate with the admin and adhere to the following protocol:

1. Person A must generate their own public key/private key pair, $pubk_{A,g1}$ and $privk_{A,g1}$, for exclusive use within the group. Once again, this is for the sake of preserving unlinkability between groups.

2. Person A must randomly generate a value, val_a, to be used as their first private key in the Diffie-Hellman component of the Double Ratchet Algorithm. Each admin must

publish their public key along with each key's respective certificate. And so, for example, admin 1 will publish `pubk_AD1, g1`, pseudonym 1, `pubk_AD1, g1`, timespan sign `privk_AD2, g1` ... `privk_ADn, g1`.

3. Next, when new members join the group, they must send their personal public key for that group to the admin. For the sake of our example, person A will send `pubk_A, g1` to the administrator. This allows the administrators to message the new members without worrying about the messages being read by any unwanted parties.

4. An admin will encrypt the SSK with the participant's public key and then sign it with their own private key to send it to the participant. For the sake of our example, admin 1 would send $\{\text{Enc}\}_{pubk_{A,g1}}(SSK)\}$ signed with $privk_{AD1,g1}$ to person A. This lets person A know that the key is valid and came from admin 1. Also, only the intended recipient (in this case, person A) will be able to see the SSK.

5. The new participant will use $pubk_{AD,g1}$ to verify the admin's signature.

6. The new participant will use their own private key to decrypt the message in order to get the group's SSK.

7. The new participant will encrypt their identity (the identifying feature of their account is their phone number, so in this case, it will be their phone number) with the SSK and send it to the third-party server.

8. Next, the server will use the new participant's encrypted identity in order to create their corresponding admin shares and citizen-class subshares for the rest of the members.

9. The server will then distribute the newly generated shares accordingly, such that each admin gets one of Alice's admin shares, and subshares of the admin shares are distributed to the other citizens. At the same time, the shares of all the other users in the group will be regenerated and redistributed to account for the membership change.

10. Finally, the new participant will be able to communicate with others in the group chat, and the SSK will serve as the initial chain key in the Double Ratchet Algorithm for messages they send and receive.

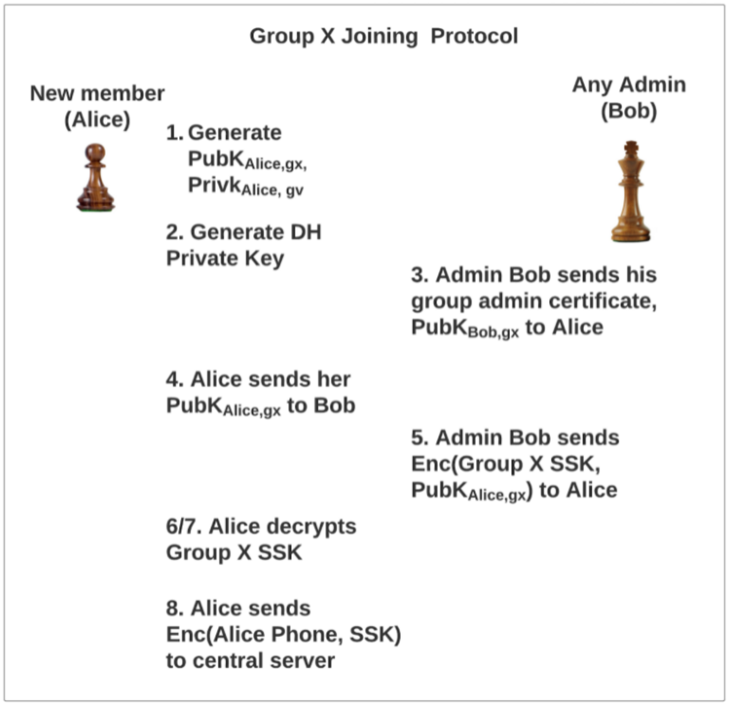
Group X Joining Protocol
New member (Alice)
Any Admin (Bob)
1. Generate $PubK_{Alice,gx}$, $Privk_{Alice, gv}$
2. Generate DH Private Key
3. Admin Bob sends his group admin certificate, $PubK_{Bob,gx}$ to Alice
4. Alice sends her $PubK_{Alice,gx}$ to Bob
5. Admin Bob sends Enc(Group X SSK, $PubK_{Alice,gx}$) to Alice
6/7. Alice decrypts Group X SSK
8. Alice sends Enc(Alice Phone, SSK) to central server

Figure 5: Steps 1-8 of the Joining Protocol

4.3.3 Share Regeneration as People Join and Leave the Group Chat

When a citizen leaves the group, the remaining shares must be regenerated and redistributed. The departing user's shares stay in the hands of the other members of the group - remaining members can still vote to expose departing users after they have left the group. However, members who join after a user departs do not get to vote on the unmasking of a departed user as seen in the section on Protecting Voting Power.

Instead of only redistributing existing shares as people join and leave the group chat, the server regenerates the shares entirely and distributes the new shares to all the members of the group chat accordingly. Otherwise, if the same set of previously generated shares were just redistributed among the new group members, older members of the group would be incrementally losing some of their privacy as more and more people aggregate the shares. Regeneration enables us to distribute shares multiple times without worrying about this, since new sets of shares would be completely incompatible with old sets of distributed shares. In this sense, the number of people that would need to be involved in revealing any individual's identity is always the same, regardless of the number of people who have joined and left the group chat since this individual joined.

Furthermore, shares are only distributed to group members who were, at one point, present in the group chat with the individual in question at the same time. Because our end-to-end encrypted messaging is implemented with the Double Ratchet algorithm, new members of a group are unable to read messages that were previously sent in the group. And since new members would not be able to read any messages sent by members of the group who left before they joined, it makes no sense for new members to have a say in identifying these former members. This is our rationale for distributing shares that pertain to a particular individual only to those who were previously in the group chat with them.

Meanwhile, the maximum number of shares a person can have that pertain to a single individual's identity is capped (see section on Identity Unmasking and Protecting Voting Power). This measure is taken to prevent a single participant from gaining too much power over another person's identity. And so, if the population of the group chat has shifted dramatically and an insufficient number of older members remain, some individuals may eventually become unidentifiable to current members of the group chat.

4.3.4 Identity Revelation

Community members have the power to unmask bad actors by voting to irreversibly release the subset of the bad actor's shares that they control (specific thresholds for unmasking are given in the section on identity unmasking protocol and protecting voting power). During a given "generation" of shares (sets of shares that are compatible with each other), individual shareholders release shares by broadcasting share contents to the group. If enough shares have been released during the generation, the user's secret (their phone number, encrypted with the group key) is revealed via interpolation. Users can then obtain the plaintext phone number by decrypting the encrypted phone number using the same group key. At the same time, the outed user can be removed from the group, in which case shares would be regenerated and distributed per the procedure in the section on leaving a group.

Meanwhile, the message revealing an individual's encrypted identity (their encrypted phone number) is sent using the same mechanism as the normal group chat messages. Therefore, the identity is protected not only by symmetric encryption with the chat's shared group key, but also by the message keys produced by the Double Ratchet Algorithm. This prevents previous members of the group (or others who may have somehow obtained the shared group key) from seeing the revealed identity in unencrypted form.

4.4 Protocol for Identity Unmasking and Protecting Voting Power

Using a hierarchical system of admins and citizens poses some unique challenges for voting to unmask a bad actor. One constraint is that identity unmasking should be possible through the intervention of either class of actors alone. That is, citizens need not be dependent on admins' cooperation to unmask a bad actor, and likewise honest admins must have a way to unmask a bad actor even if the citizens are indifferent or unwilling. Another is that one person's vote to unmask should not waste the vote of another. All participants should have voting power independent of the actions of other actors, although each admin is proportionally more powerful than a single citizen (assuming a large citizen to admin ratio).

We resolve these issues through the following scheme:

1. Suppose we have K admins and M citizens in our event group (admins are double-counted as citizens as well).

2. The secret s identifying user Eve is first split by the server into K admin-class shares (one issued to each admin, of which at least 60% are required to unmask a user's secret).

3. Each of Eve's admin-class shares is subsequently split by the server into $2M$ citizen-class subshares (of which any $M + 1$ are required to reveal the admin share).

4. Each citizen gets 2 citizen-class subshares from each of Eve's admin-class shares. Thus, each citizen holds a total of $2K$ citizen-class subshares derived from s, since we have K admins.

5. Thus, across the population of M citizens, there exist KM admin-class shares and $2KM^2$ citizen-class subshares.

6. If a user Bob wants to unmask Eve, he broadcasts all $2K$ of Eve's citizen-level subshares in his possession

7. If an admin Cathy wants to unmask Eve, she broadcasts her single admin-level share derived from Eve's secret s, as well as all $2K$ citizen-level subshares derived from s in her possession (as an admin, Cathy is double-counted as a citizen as well)

8. If a user Bob wants to unmask Eve, he must release all of Eve's shares in his possession. Partial subshare release is not permitted since this would complicate vote-counting and goes against the spirit of one-person one-vote.

Some caveats and edge cases are discussed at the end of this section.

The use of parallel yet interconnected share schemes resolves both issues raised at the beginning of this section. First, we show that each class can independently unmask a bad actor Eve:

- At least 60% of admins release their admin-level shares corresponding to Eve. Since citizens hold subshares derived from admin shares, citizen action is not required if a sufficient number of admins vote to unmask.

- A simple majority of citizens release all of Eve's citizen-level shares in their possession. This releases at least $M + 2$ citizen-level subshares per admin share, since citizens hold subshares from every admin. Effectively, this mechanism first unmasks all admin shares, which then decrypts Eve's secret since we exceed the 60% threshold for admin shares. In other words, a majority of the citizens means effective unanimity of the admins.

We additionally mitigate the issue of wasted votes by distributing subshares across citizens, rather than dividing the citizens into K "constituencies" represented with its own admin. In the latter system, if an admin of a constituency votes to unmask, their constituents' votes are wasted, since constituents hold subshares to unmask their representative admin's share. Our approach of distributing subshares does "dilute" citizens' power, but citizens are never disenfranchised by the actions of a single admin.

Some caveats with the scheme:

1. We do not allow cross-class collaboration to unmask. For example, we cannot have less than 60% of admins collaborate with a minority of citizens to unmask a bad actor.

2. Shares are replaced upon triggering of share regeneration with a new group key, which can happen after a group membership change or upon a periodic group key change (described in the previous section on Group mechanics).

3. A user Bob will never receive shares from a user Alice if Alice left the group before Bob joined. Similarly, if Bob were to leave the group, members who join after Bob's departure cannot receive any of Bob's shares. (The server can store timestamps of when individuals joined and left the group).

4. After a user Alice leaves, the number of people who had interval overlap with Alice and are still present is monotonically decreasing. Unchecked, this would allow longtime group members to gradually control a greater percentage of an old ex-users' shares, endangering the pseudonymity of members who left early. To stop this, we prevent a user from holding more than $\frac{1}{7}K$ of the shares of any other member. In effect, we say that no citizen can be more powerful than an admin. One small caveat of this is that unmasking an ex-user's identity by citizens may be impossible if enough co-users have left. However, this should be a minor issue since we have forward secrecy and break-in recovery, preventing new users from seeing messages sent by ex-users.

5 Security Evaluation

Our system design uses end-to-end encryption to protect messages, enforces conditional pseudonymity and accountability, and preserves forward secrecy and break-in recovery.

5.1 End-to-End Encryption

By using Signal's Double Ratchet Algorithm for our messaging, we ensure that only the senders and recipients of each message can read the message (as they are the only ones with the appropriate message keys). The third-party organization providing this application will not be able to read these messages, and so, even if they are subpoenaed by law enforcement as Signal was in the past, the amount of meaningful data they would be able to reveal would be limited.

5.2 Conditional Pseudonymity and Accountability

By giving the members of our group chats pseudonyms, we allow them some degree of identity protection without hindering effective communication. Our secret sharing system, on the other hand, enforces accountability and ensures that perpetrators of bad behavior or harassment can be held accountable and identified (without needing a subpoena!).

5.3 Forward Secrecy

Forward secrecy is preserved in our messaging system through the Double Ratchet Algorithm, which prevents new people who join the group chat from reading any of the old messages (since they do not have access to any of the message keys used to encrypt the old messages) [sig21]. Meanwhile, forward secrecy is preserved in our accountability system since new people do not have any say in identifying people who left the group chat before they joined, since they would never receive the relevant shares required.

5.4 Break-in Recovery and Effective Banning

Break-in recovery is preserved in our messaging system through the Double Ratchet Algorithm in the sense that future output keys look random to an adversary who learns the Key Derivation Function key at some point, provided that future inputs have added sufficient entropy [sig21]. Similarly, for us, this translates into the ability to effectively boot people from group chats. Since everyone is given the shared group key when they join the group chat, all people who were part of a group chat at some point learns the initial chain key that serves as the first input to the Key Derivation Function for every single thread of messaging. However, because of the Double Ratchet Algorithm's break-in recovery property, people who were kicked from the group chat cannot derive future message keys, assuming current members of the group chat use the system as intended and actually regenerate their Diffie-

Hellman private keys randomly after each round of messaging. As for break-in recovery and banning in our accountability system, adversaries cannot decrypt identities that the group has chosen to reveal (post-expulsion, for former members of the group). This is because we send these messages using the same mechanism as the group chat messages, which are encrypted with the ratcheting message keys. And so, people who have left the group chat would not be able to read these messages, much less use the shared group key to decrypt and discover the newly-revealed identity.

6 Conclusion

In this report, we present a system design for a pseudonymous messaging platform with community accountability to support the need for secure broadcast at in-person protests, and show how end-to-end encryption could be implemented in such a system. We learn from the successes and failures of existing apps (e.g end-to-end encryption on Signal, lack of accountability in Yik Yak) to create a system satisfying the unique security requirements of in-person protests. Our community accountability scheme uses Shamir secret sharing to ensure that the community can reveal bad actors on this platform. We use Signal's Double Ratchet Algorithm in order to encrypt our message end-to-end and to take advantage of its built-in security properties, like forward secrecy and break-in recovery. And finally, we use public key infrastructure in order to facilitate the distribution of a shared group key that allows us to encrypt identities so that we do not need to trust the application's third-party server. We present some limitations of the system as it stands currently, and offer future directions to add to system functionality going forward.

6.1 Limitations

While our system allows for pseudonymous, end-to-end messaging with accountability, there are malicious user behaviors that our system is unable to prevent.

A key limitation is the inability to protect pseudonymity for groups in which bad actors make up the majority of the group. Since just over 50% of citizens can release shares and thus vote to unmask, honest users have no identity protections in such a group if the bad actors conspire to unmask all new group members. Our scheme assumed that our groups would have a majority of honest users; we decided that a group with a majority of bad actors could be considered inherently compromised and thus out of our spec. This problem could be mitigated by increasing the threshold of citizens needed to unmask a user, but this makes it even harder to unmask a single bad actor in a community of mostly honest citizens.

Another concern is vigilantism, in which a private clique of users can unmask a pseudonym without making the unmasking public. For example, if 51 shares are required to expose a user "Alice", 48 of Alice's shares are already public, and Bob, Charlie, and Dana have unreleased shares from Alice, the three of them can run interpolation privately by disclosing their shares within their clique, and know who Alice is without disclosing that to the group. This is more of a concern when a user is already close to being identified. We categorize this as user behavior we cannot control. An alternative to this would be keeping disclosed shares on the server until the threshold for unmasking was reached. However, this grants the server greater power over user information, and we do not assume the server is a trusted entity, since the organization providing this messaging service can very well be subpoenaed or otherwise compromised. Also, should the announcement of a person's identity be released by the server instead of disseminated through the group chat, the message would no longer be protected by the changing message keys in the Double Ratchet Algorithm, which renders this part of the process susceptible to break-in recovery.

While the share cap prevents old users from being able to unmask ex-members from the far past with excessive ease, our system still has some inherent limitations due to dynamic group sizes (presumable, most groups will grow in size over time). Since we cannot force old users to delete their shares even after they have left the group, vigilantism among older users is more concerning than vigilantism among newer users (or really, users in larger groups), since fewer people would be required to band together in order to illegitimately reveal someone's identity.

A final concern is the possibility that this system could be abused by criminals to plan violent or nefarious acts. For example, criminals could form their own temporary groups or broadcast messages inciting violence and other criminal behavior in groups of mostly honest users. However, the possibility for abuse is an inherent risk for any technology, and we argue that we at least have protections in place to prevent malicious actors from infiltrating groups of mostly honest users through our accountability system. Though our application is pseudonymous, suspicious users can still be identified with sufficient community support, which is not the case for Yik Yak.

6.2 Future Directions

As we had to make several assumptions to limit the scope of the design, there are several directions that our design could take in future iterations. One aforementioned limitation is that all users have the ability to send messages, without any limit to volume, or “hierarchy” of messages. For organizing events, it may be helpful to know when an official leader is communicating instructions to the rest of the group, so a future iteration could help to identify messages specifically from admins. However, we would still want to preserve the pseudonymity of the individual admins. One way to achieve this is by creating a ring signature between all of the admins, such that admin messages are endorsed in the group, and could thus be distinguished.

Additionally, although our design has the goal of decentralization, there are aspects of the design that still rely on a third party server, both to forward messages through the internet, and to distribute shares. One technology that could be beneficial is bluetooth, which would rely on proximity of users rather than communicating through a third party server. Perhaps an approach to send messages through bluetooth could be a gossip protocol that sends messages to users nearby, as long as all are within range. This would still meet the goal of “day-of protest” communication, but a drawback is that group members would now need to be within bluetooth range of each other.

7 Acknowledgements

We would like to thank Professors Ron Rivest and Yael Kalai for a wonderful semester. We are also very grateful to the help of our LA mentor, Kyle, for answering our many questions and providing critical feedback on our system. And finally, we would like to thank TAs Billy, Deep, and Andres advice on our design during office hours.

References

[Alb19] Jensen Marekova Albrecht, Blasco. Mesh Messaging in Large-scale Protests: Breaking Bridgefy. 2019.

[blu] What is Bluetooth Class?

[Chu21] Mike Chu. Can You Connect Multiple Bluetooth Devices At Once? See 7 Concurrently. *Data Overhaulers*, Jan 2021.

[Duf21] Clare Duffy. Why messaging app Signal is surging in popularity right now. *CNN Business*, Jan 2021.

[GN20] Andy Greenberg and Lily Hay Newman. How to Protest Safely in the Age of Surveillance. *Wired*, May 2020.

[kue21] *Organizing for Effective Advocacy.* University of Kansas, 2021.

[LBF20] Taylor Lorenz, Kellen Browning, and Sheera Frenkel. TikTok Teens and K-Pop Stans Say They Sank Trump Rally. *The New York Times*, Jun 2020.

[Marne] Lenka Mareková. Real World Crypto Symposium: Group Messaging. 2021. [Online].

[Nie20] Amelia Nierenberg. Signal Downloads Are Way Up Since the Protests Began. *The New York Times*, Jun 2020.

[Pat18] Mihir Patkar. 5 Common Bluetooth Myths You Can Safely Ignore Now. *MUO*, Dec 2018.

[Qui20] Michaela Quigley. How to Stage a Protest. *Boston Magazine*, 2020.

[Saf17] Valeriya Safronova. The Rise and Fall of Yik Yak, the Anonymous Messaging App, May 2017.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[siga] Grand jury subpoena for Signal user data, Eastern District of Virginia.

[sigb] Specifications: The Double Ratchet Algorithm.

[sig21] The Signal Protocol and the Double Ratchet algorithm, Apr 2021.

[Tig19] Kristoffer Tigue. You're not alone if your Bluetooth headphones keep dropping their connection in the city. *CNBC*, Apr 2019.