Amigo: Secure Group Mesh Messaging in Realistic Protest Settings

David Inyangson[1*], Sarah Radway[2*],
Tushar M. Jois[3], Nelly Fazio[3] and James Mickens[2]

[1] Johns Hopkins University
dinyang1@jhu.edu
[2] Harvard University
sradway@g.harvard.edu, mickens@g.harvard.edu
[3] City College of New York
tjois@ccny.cuny.edu, nfazio@ccny.cuny.edu

Abstract. In large-scale protests, a repressive government will often disable the Internet to thwart communication between protesters. Smartphone mesh networks, which route messages over short range, possibly ephemeral, radio connections between nearby phones, allow protesters to communicate without relying on centralized Internet infrastructure. Unfortunately, prior work on mesh networks does not efficiently support cryptographically secure group messaging (a crucial requirement for protests); prior networks were also evaluated in unrealistically benign network environments which fail to accurately capture the link churn and physical spectrum contention found in realistic protest environments. In this paper, we introduce Amigo, an anonymous mesh messaging system which supports group communication through continuous key agreement, and forwards messages using a novel routing protocol designed to handle the challenges of ad-hoc routing scenarios. Our extensive simulations reveal the poor scalability of prior approaches, the benefits of Amigo's protest-specific optimizations, and the challenges that still must be solved to scale secure mesh networks to protests with thousands of participants.

Keywords: mesh messaging · censorship circumvention · continuous group key agreement · mesh routing protocols · mobility modeling

1 Introduction

Protests are an important tool in the fight against authoritarian governments. By occupying a physical space, ordinary citizens can express their grievances and bring visibility to their concerns. In countries where political freedoms are curtailed, protests may be the *only* mechanism for the public to express their grievances.

When protests become movements comprising thousands of people, communication becomes vital for organizing protest actions and keeping participants safe. In a modern protest, the demonstrators often rely on smartphone-based messaging platforms like WhatsApp or Telegram to communicate; using these apps, protesters learn about rally points and inform each other about the location of the police or the military [ABJM21a]. Authoritarian governments are thus highly motivated to intercept or alter the messages that protesters exchange. Using intercepted messages, the government can identify protest leaders and shift police deployments to disrupt nascent rally points; using altered messages, the government can send spoofed messages, ostensibly from protest leaders, that direct

[*]These authors contributed equally to this work.

protesters towards harm. These concerns are not theoretical. For example, law enforcement – even in relatively liberal governments – possess *cell-site simulators* which create fake cellphone towers [Cel15]. By inducing protester phones to pair with a simulator, law enforcement can launch of variety of attacks, e.g., downgrading the phone's cellular connection to the unencrypted 2G protocol, allowing the government to intercept messages to and from that phone [Ele23].

A repressive government may also try to degrade or completely disrupt app-based communication during a protest. For example, during the 2019 Hong Kong anti-ELAB protests, the government (or its partisans) launched a denial-of-service attack against Telegram servers [BeW19]. In 2019, Iran responded to protests involving economic discontent by enacting such a large-scale Internet shutdown [Amn20]. Similarly, in Myanmar, Internet shutdowns are frequently imposed to prevent civilians from documenting human rights violations or accessing information about military operations [Acc24a]. Overall, there were as many as 283 Internet shutdowns across 39 counties in 2023 alone – the worst year for Internet shutdowns on record [Acc24b].

Given this situation, protesters need a robust way to exchange authenticated, encrypted messages, even if the government has disabled Internet connectivity or tampered with cellular routing infrastructure. A promising approach is for protesters to exchange encrypted messages via an *ad-hoc mesh network* built from point-to-point radio links (e.g., Bluetooth connections between smartphones). In a mesh network, there is no centralized routing infrastructure; instead, user devices exchange messages over (potentially ephemeral) short-range links created via physical proximity as users move in space. Multi-hop message forwarding atop the point-to-point links enable information exchange between users too far away to establish direct connections.

Because mesh networks do not rely on centralized infrastructure, there is no single point of failure for a regime to attack. Unfortunately, classic mesh networking protocols [AW09, AWW05, ZLH06] do not safeguard user privacy, revealing message senders and recipients to even passive observers. These protocols are ill-suited for protest scenarios where users wish to hide their identities from the government [ABJM21a].

The research community has introduced mesh networks that attempt to hide user activity [PJW$^+$22, PSEB22, BRT23, RL08, RYLZ09, LQK09, Sen12, SS14, PAC14, HBDF$^+$13]. However, previous systems cannot practically handle large-scale protests for two major reasons. First, prior systems lack efficient (or any) support for private groups, despite the widespread use of group-based messaging in protest settings [ABJM21a]. Second, prior systems use routing protocols which are scalable in benign network conditions, but often experience persistent congestion collapse for the mobility patterns and smartphone radio technologies found in realistic protest settings; the result of the persistent network congestion is a very low rate for application-level message delivery.

To address these challenges, we introduce *Amigo*, a new system for efficient, secure mesh communication in protest settings. Amigo's high-level novelty is that it is designed as an *end-to-end* system which accounts for both security challenges at the cryptographic layer and performance/reliability challenges at Layers 1 and 2 of the network stack. To protect messages exchanged by protesters, Amigo leverages a new protocol for continuous group key agreement (CGKA), efficiently providing security properties specifically required by protesters (e.g., post-device-compromise security, and fast removal of a member from a group). In tandem, Amigo introduces a new mesh routing protocol which, compared to earlier protocols, enjoys higher delivery rates for mobility patterns often seen in real protests. Our comprehensive simulation experiments, which leverage those realistic mobility models and high-fidelity representations of low-level network phenomena like radio collisions between nearby phones, demonstrate that Amigo's protest-specific optimizations provide secure, protester-desired functionality with better routing performance than earlier work. However, our evaluation results also reveal fundamental scalability challenges for all known approaches to secure mesh networking (including Amigo); a fundamental contribution of the paper is illuminating those practical challenges (previously unknown due to the lower-fidelity simulations of prior work), and suggesting several directions for future work.

Contributions: In summary, we contribute the following:

- We introduce a new protocol for decentralized group key agreement in mesh networks. The protocol tolerates an unreliable mesh that may drop or reorder messages, providing important security properties missing from prior key management schemes. The protocol is efficient with respect to network consumption and CPU usage—a critical feature for a protocol that runs atop resource-constrained phones.
- Amigo introduces a new routing approach, based on clique-level forwarding, which is tailored to the vagaries of protest-based mobility patterns.
- To evaluate Amigo, we introduce new mobility models which capture how real-life protesters organize in physical space, both during stable times and when external shocks arise (e.g., the unexpected appearance of the police). Our evaluation experiments also capture important Layer 1 and Layer 2 network phenomena that were ignored in prior work on seucre mesh networking.
- We test Amigo's performance when Amigo uses our clique-based routing approach and routing approaches from prior work, demonstrating (1) the benefits of Amigo's protest-specific optimizations at the cryptographic layer and the routing layer, and (2) the scalability and robustness challenges (previously unknown) that still remain unsolved, given the challenging network environment created by protest-based mobility patterns and the idiosyncrasies of real-life short-range communication technologies. Our results suggest mesh routing protocols previously thought to be practical will often function poorly in realistic protest environments due to a self-reinforcing cycle of network congestion. We discuss the implications of our results on future research into shutdown-resistant communication.

Once the paper is published, we will open-source all of our code and benchmarks, to spur additional work on secure messaging systems for protest scenarios.

2 Motivation

A large-scale protest involves hundreds or thousands of people gathered in (and moving throughout) a shared physical space. The success of a such a protest increasingly depends on the ability of protesters to organize via smartphone-based communication [Shi11, LA10, ABJM21a, AG15]. Without the ability to effectively communicate, critical information regarding group movement, law enforcement activity, and collective decision-making propagates slowly among protesters. The consequences for the protesters are dire: physical assault, imprisonment, or even death [CG18].

As a case study, consider the widespread 2019–2020 protests in Hong Kong. The protests erupted in response to the government's attempt to pass the Extradition Law Amendment Bill (ELAB). The bill would have allowed Hong Kong citizens to be extradited to China; many Hong Kong citizens feared that the bill would further erode Hong Kong's legal independence from China [Lee20, LYTC19]. Using smartphone apps like WhatsApp, Telegram, and Signal, protesters decided where and when to hold demonstrations. Once protesters arrived on site, these apps allowed protesters to respond in real time to events like the arrival of police [ABJM21a].

The ability of citizens to efficiently organize *during* the protest itself was a marked departure from static, pre-smartphone-era demonstration strategies. In the ELAB protests, a column of marchers might quickly decide to split into groups, each taking a path down a different street and abruptly changing directions in response to newly-placed barricades; using apps like Telegram, the independent groups could synchronize their long-term movement trajectories and later reform into a single human chain [BeW19].

2.1 Protester Needs

Albrecht et al. performed wide-ranging interviews with 11 anti-ELAB protestors from Hong Kong [ABJM21a]. Below, we highlight the relevant findings from that study with respect to how communication apps for protesters should be designed:

1. **Protest communication occurs primarily in group chats:** Groups can range from a handful of members to tens of thousands of participants.
2. **Groups must be able to securely add members as a protest unfolds.** Some protesters may hear about a demonstration offline, e.g., word-of-mouth. Upon arrival, such a person may wish to join an online group for protest coordination. A protest-focused messaging app requires the ability to add new members on-the-fly.
3. **Groups must be able to remove members.** Protesters are resigned to the fact that law enforcement may be able to infiltrate groups (especially larger ones). Protesters also understand that law enforcement may coerce a loyal group member to nonetheless relinquish their device and their login credentials. Thus, revoking a user's group membership is critical functionality.
4. **Communications should be possible during an Internet shutdown.** Protest-focused communication apps should still function if centralized Internet infrastructure has been degraded. During the Telegram blackout in the ELAB protests, some protesters attempted to use Bridgefy [Bri22], a mesh networking app, to exchange messages. Unfortunately, Bridgefy failed to handle the load presented by the large numbers of protesters [ABJM21a].
5. **Communication should be anonymous.** The threat of retaliation from law enforcement requires participation in an online protest group to be anonymous. An encrypted message should not reveal the real-life identity of the sender or the receiver. The peer-to-peer nature of mesh networking does not automatically provide anonymity. For example, security researchers have found that Bridgefy's communication protocol protects neither message confidentiality nor message authenticity, and allows outsiders to deanonymize users [ABJM21b].

In summary, protesters need an end-to-end encrypted messaging app that is designed for group communication and supports group membership changes, does not rely on traditional Internet infrastructure, and provides anonymity.

2.2 Related Work

Prior work has explored smartphone-based mesh networks as a communication substrate for protesters without centralized Internet access. Mesh networks leverage short-range radio technologies like Bluetooth or Wi-Fi Direct, routing messages between a sender and a receiver across potentially ephemeral radio connections between mobile nodes. In the protest context, end-to-end encryption of signed/authenticated messages ensures that network eavesdroppers cannot inspect cleartext or alter messages without detection. An ideal mesh network would also limit the power of an attacker who discovers the keys used by protesters at time t. Forward secrecy ensures the attacker cannot break the confidentiality of messages sent before t. Post-compromise security means that, after the key leakage at t, the attacker cannot break the confidentiality of messages sent *after* t. Broadly, forward and post-compromise security are implemented by having participants rotate their keys periodically.

Moby [PJW$^+$22] is a mesh network that supports pairwise messaging. Moby uses Signal's symmetric key ratchet protocol [Sig20] to provide end-to-end encryption and forward secrecy. Each pair of communicating nodes shares a key; a node determines whether a received message is destined for the local node by seeing whether any of the node's pairwise keys enables successful recreation of the message's HMAC. Using this HMAC validation trick, Moby avoids the need to embed cleartext sender or receiver IDs

Table 1: Related work comparison. E2E: end-to-end encryption; FS: forward secrecy; PCS: post-compromise security; Anon: anonymity; Pair: pairwise communication.

Scheme	E2E	FS	PCS	Anon	Pair	Groups
Moby [PJW+22]	✓	✓	✗	✓	✓	✗
ASMesh [BRT23]	✓	✓	✓	✓	✓	✗
Rangzen [LFBD+16]	✗	N/A	N/A	✓	✗	✗
Perry et al. [PSEB22]	✓	✗	✗	✓	✓	✓
Amigo (this)	✓	✓	✓	✓	✓	✓

in messages, enabling anonymity. Moby's routing protocol uses flooding, such that two peers attempt to exchange all locally buffered messages upon encountering each other. ASMesh [BRT23] is another network which augments Signal's Double Ratchet algorithm and routes messages using flooding.

The network of Perry et al. [PSEB22] uses flooding and provides both end-to-end encryption and anonymity via a CPA-secure signcryption scheme; the signcryption scheme supports group communication by having a group creator generate a single secret key and share the key with each individual member of the group. Perry et al.'s scheme does not provide forward secrecy or post-compromise security.

Peers in a Rangzen mesh [LFBD+16] propagate messages via epidemic flooding. However, a user Alice's phone gives preference (i.e., more local buffer space) to "trusted" messages. Alice determines the trust of a message from Bob by examining how many mutual friends Alice and Bob share in a social networking graph that is maintained out-of-band with respect to Rangzen. Rangzen does not end-to-end encrypt messages because Rangzen is intended to spread "micro-blogs" (akin to tweets) that are destined for all users and are assumed to contain no private information.

As a mesh network grows, flooding-based routing approaches scale poorly, requiring bandwidth that is quadratic in the number of nodes. Perry et al. [PSEB22] reduce the bandwidth costs using Bloom filters. When two peers encounter each other, each one generates and shares a summary of the messages that are stored locally; when a node receives such a summary, the node consults its own summary and fetches only those messages which have not already been downloaded. Perry et al. describe a further optimization in which nodes are partitioned into cliques, such that when a node wants to send a message to a peer outside the local clique, the node sends the message to a "clique leader" node. Clique leaders form a communication backbone for the network, forwarding cross-clique messages between leaders until a message reaches its destination clique (at which point the message is broadcast to all clique members). Importantly, Perry et al. evaluate their routing approaches using a custom simulator that does not capture important phenomena in Layers 1 and 2 of the network stack. For example, their simulator does not model Layer 1 collisions between simultaneously transmitting nodes. As we demonstrate in Section 7, low-level network events can affect higher-level mesh routing protocols in significant ways. Table 1 summarizes prior work and compares it to Amigo.

3 System Design

At a high level, Amigo's goal is to provide a mesh network that is reliable, low-latency, secure, and resource-efficient with respect to the consumption of CPU cycles, RAM, and network bandwidth. *Reliability* means that Amigo successfully forwards the vast majority of messages to their intended recipients, despite the stochastic nature of pairwise node connectivity. *Low-latency* forwarding means that Amigo delivers messages as quickly as possible, given the prevailing network conditions. *Secure* forwarding ensures that messages are end-to-end encrypted, integrity-protected, and enjoy both forward security and post-compromise security; additionally, key agreement among group members should not leak cryptographic information to participants outside of the group. *Resource efficiency* is important because Amigo targets smartphone deployments in which network bandwidth, CPU cycles, memory space, and available power are limited compared to traditional server-based environments. To the best of our knowledge, Amigo meets or outperforms all prior work along these evaluation metrics. However and importantly, as we discuss in Sections 7 and 8, our evaluation experiments illuminate fundamental performance and robustness challenges for secure mesh networks—challenges that must be solved for these systems to reliably scale to protests involving thousands of participants or more.

3.1 Architecture

Figure 1 depicts Amigo's high-level architecture. Users interact with Amigo via the **application layer**, creating and destroying groups, adding or removing members to groups, and exchanging messages with group members. Users directly generate and read text messages or picture messages; behind the scenes, those messages are protected with keys negotiated by Amigo's **session layer**. The session layer implements an efficient CGKA scheme which we describe in detail in Section 4. Amigo focuses on small to medium-sized groups with 10-200 members; such close-knit groups are perceived by protesters as more trustworthy and less susceptible to infiltration by the government [ABJM21a].

The session layer sends and receives user-generated messages and CGKA-generated messages via the **routing layer**. As we describe in Section 5, Amigo is compatible with three schemes from prior work on mesh networking. Amigo also introduces a new approach, dynamic clique routing, which is tailored for the unique connectivity challenges of protest scenarios. As shown in Figure 1, the basic idea behind all of the routing schemes is that, when two nodes wander into communication range, they exchange locally-buffered messages that were generated locally or received from other peers in the past; the routing schemes differ in which messages are exchanged with which peers.

At the bottom of the Amigo stack is the **link layer**. Similar to prior work on secure meshes [LFBD$^+$16, PJW$^+$22, PSEB22], Amigo leverages off-the-shelf point-to-point radio technologies like Bluetooth and Wi-Fi Direct. Using off-the-shelf hardware for Layers 1 and 2 of the network stack makes Amigo imminently deployable; however, as we discuss in Sections 7 and 8, traditional short-range radio protocols lack understanding of end-to-end congestion metrics, and therefore may unwittingly overload the network and trigger precipitous declines in application-level delivery rates.

3.2 Threat Model

As explained by Albrecht et al. [ABJM21a], the core goals of an authoritarian regime during a protest are to:

1. identify message senders and group members,
2. read the cleartext content of in-transit messages,
3. tamper with in-transit messages, and
4. compromise protester devices after a protest to decrypt logged messages from earlier in time or inspect/tamper with messages that protesters will sends in the future.

We assume that the adversary has several vantage points on the network, but cannot view all traffic emanating from all nodes. We also assume that the adversary, through malicious or lawful means, can access a group member's device [ZJG21]; after doing so, the adversary gains access to the group messages and group key material of all groups associated with that device. This knowledge allows the attacker to monitor group communication indefinitely unless removed.

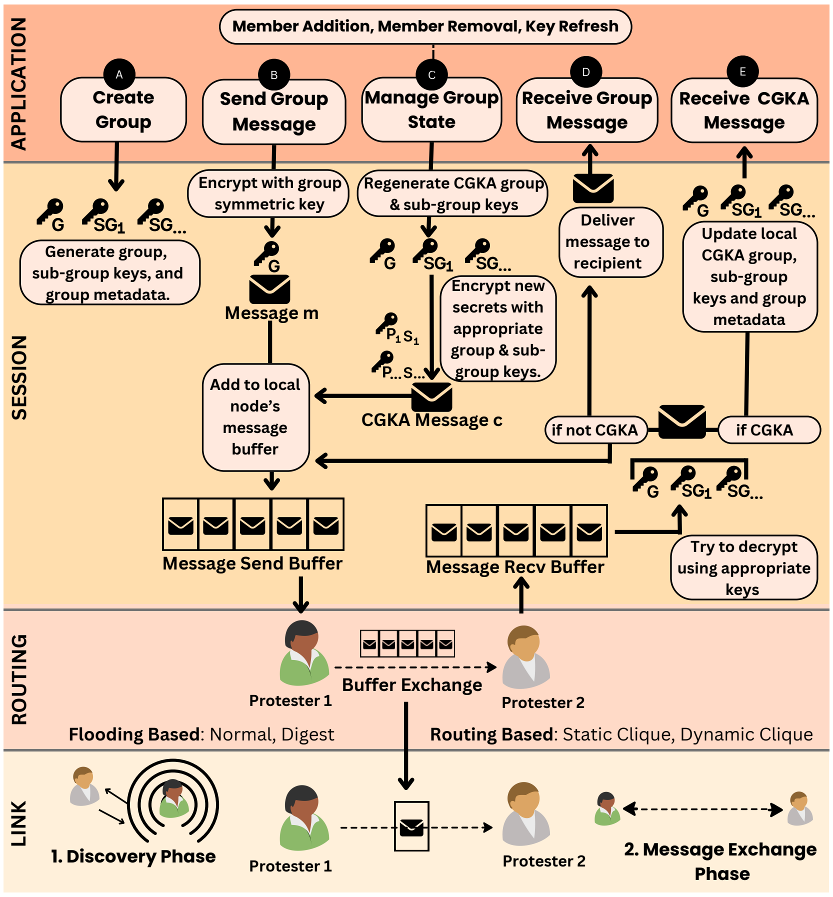
APPLICATION
Member Addition, Member Removal, Key Refresh
A Create Group
B Send Group Message
C Manage Group State
D Receive Group Message
E Receive CGKA Message
SESSION
Generate group, sub-group keys, and group metadata.
Encrypt with group symmetric key
Message m
Regenerate CGKA group & sub-group keys
Encrypt new secrets with appropriate group & sub-group keys.
CGKA Message c
Add to local node's message buffer
Deliver message to recipient
Update local CGKA group, sub-group keys and group metadata
if not CGKA
if CGKA
Try to decrypt using appropriate keys
Message Send Buffer
Message Recv Buffer
ROUTING
Protester 1
Buffer Exchange
Protester 2
Flooding Based: Normal, Digest
Routing Based: Static Clique, Dynamic Clique
LINK
1. Discovery Phase
Protester 1
Protester 2
2. Message Exchange Phase

Figure 1: An overview of Amigo and its layers.

Given such an adversary, Amigo's goal is to provide messages with confidentiality and integrity, including forward security and post-compromise security, as well as outsider anonymity. We discuss these properties further in Section 4.1. Amigo does not try to provide message deniability, e.g., Amigo does not prevent traffic analysis techniques that a multi-vantage-point attacker can use to correlate message sends with messages receptions and thereby unmask senders and receivers. Amigo also does not prevent denial of service attacks in which malicious nodes refuse to forward packets or jam the physical spectrum. Amigo is compatible, however, with prior work that augments message routing with a trust layer that allows nodes to prioritize communicating with peers estimated to be well-behaved [LFBD+16, PJW+22].

4 A CGKA Protocol for Mesh Messaging

We now discuss our security goals, the implementation of Amigo's application and session layer as shown in Figure 1, and the properties our CGKA protocol provides to enable confidential and anonymous group communication.

4.1 Security Definitions

Protesters require confidentiality of their messages from outsiders, as well as a way to verify message authenticity. Since adversaries may sometimes successfully compromise group member's devices and key material, protesters also require the properties of forward secrecy and post-compromise security. Additionally, messages should be outsider-anonymous to protect protesters from retaliation. We define these security requirements:

Message Confidentiality: A message m sent to subgroup ω' of a group ω, should remain confidential from both an adversary who has not achieved device compromise and members of ω who are not in ω'.

Message Authentication: A recipient of a message m sent by a member in group ω can prove it was sent by another member in ω and m was not tampered with.

Forward Secrecy: Applying a state operation m' to a group state Ω_t results in a new group state Ω_{t+1}, where a successful compromise of Ω_{t+1} does not allow an adversary to decipher messages sent in group states Ω_t or prior.

Post-Compromise Security: Applying a state operation m' to a group state Ω_t results in a new group state Ω_{t+1}, where a successful compromise of Ω_t does not allow an adversary to decipher messages sent in group states Ω_{t+1} or later.

Outsider Anonymity: A message m intended for subgroup $\omega' \subseteq \omega$ leaks no information about the identity of the recipients of the message to anyone who is not part of the subgroup ω'. Any member of ω', however, may learn who the other intended recipients are. This relaxed notion of anonymity [FP12] is appropriate here since protesters vet other members before inviting them into group communication and do not need to be anonymous within their groups [ABJM21a].

4.2 Design

As opposed to pairwise approaches where duplicate messages are encrypted under different keys, a CGKA protocol utilizing a ratchet tree results in a single message encrypted under a shared key. Traditional CGKA protocols like TreeKEM [BBR18] utilize *ratchet trees* to provide efficient key derivation and encrypted communication [KPPW+21, CCG+18, CLM23]. A ratchet tree, displayed in Figure 2, is a binary tree where each node in the tree contains asymmetric key material. Leaf nodes ($\texttt{Member}_A$, $\texttt{Member}_B$, $\texttt{Member}_C$, $\texttt{Member}_D$) represent the participants in the group communication.

Importantly, group state changes using a ratchet tree are *logarithmic* in the size of the group, rather than linear, as in pairwise approaches. In a traditional CGKA protocol such as TreeKEM [BBR18], if $\texttt{Member}_C$ in Figure 2 wishes to update the group state, they will generate a new unique key pair (sk'_C, pk'_C). They then iteratively hash sk'_C to generate secret keys for their ancestors: $sk'_2 = H(sk'_C)$, $sk'_R = H(H(sk'_C))$. These new secret keys are then encrypted under the corresponding ancestor's current public key: $\mathsf{Enc}_{pk_2}(sk'_2), \mathsf{Enc}_{pk_R}(sk'_R)$. Members of the group can then recover the new secrets; all members can recover the new root sk'_R, but only $\texttt{Member}_D$ can recover sk'_2. The public keys derived from the new secret keys pk'_2, pk'_R are distributed to all members.

Each member only has the secret keys on their path to the root of the ratchet tree, which allows for efficient (i.e., logarithmic) communication to subsets of the group. In Figure 2, if $\texttt{Member}_C$ wishes to message the whole group, they can encrypt using pk_R. But, if $\texttt{Member}_C$ wants to send a message only to $\texttt{Member}_A$ and $\texttt{Member}_B$, they may simply encrypt using pk_1. As descendants of $\texttt{Node}_1$, only $\texttt{Member}_A$ and $\texttt{Member}_B$ will have sk_1 and can decrypt.

While these traditional protocols are efficient, they require all members to apply state operation messages in the same sequence. Otherwise, member's respective views of the ratchet tree will be inconsistent. While a central server could provide ordered delivery of messages, this can not be relied upon during an Internet shutdown.

As a result, Amigo requires a way for members to receive out-of-order state operation messages and still reach a consistent state. To do this, we apply the idea proposed by Causal TreeKEM [Wei19]; instead of state operations that use hashing to overwrite key

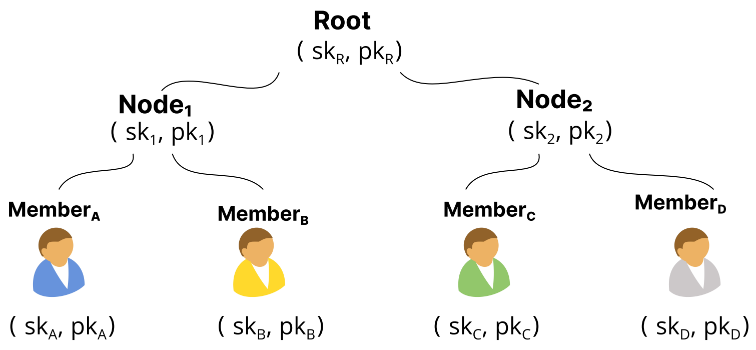

Figure 2: A ratchet tree for asymmetric key material, as in Amigo.

material held by nodes in the ratchet tree, state operations may *combine* existing and newly generated material, an operation denoted by $\star$. If two keypairs arrive at different times, we can combine them together into a valid keypair, relaxing the need for ordered message delivery: $(sk_x, pk_x) \star (sk_y, pk_y) = (sk_{x\star y}, pk_{x\star y})$. We maintain the same structure as Figure 2, with group members only having access to the secret keys of their ancestors, but when state changes occur, we *combine* instead of *replace* key material. We discuss further in Section 4.3.

Our Amigo CGKA is built around elliptic curve Diffie-Hellman using Curve25519. Elliptic curve public keys are derived by multiplying a (secret) scalar sk with the curve's generator point G. Thus, we can instantiate $\star$ as $(sk_{x\star y} = sk_x + sk_y,\ pk_{x\star y} = (sk_x + sk_y) \cdot G)$. Note that the original Causal TreeKEM construction is not forward-secure *within* a group state [WKHB21]; we extend our CGKA to use *updateable* public key encryption [ACDT19] to provide this property.

Amigo's CGKA will converge with eventual consistency, i.e., when a group's state messages eventually reach the specified user. But, due to the additive nature of our group key agreement implementation (and the fact that it does not check for liveness of nodes), group fragmentation may occur if certain nodes lack key material for the groups for which they are a part. Affected nodes can query the group for this missing material via the root node's keys, however. Additionally, a group may choose to restructure, effectively re-creating and re-adding live members.

We also note that an application-level mechanism for group healing may be viable. All members can retain their prior ratchet tree state for a certain number of epochs – this window can be decided at group creation. When a member who has failed to receive CGKA state update messages transmits a message intended for the group, they will unknowingly use material from a prior state. Any recipient of this message may successfully decrypt and recognize this, as long as the relevant key material is within their window, and respond with the `Root` node's key material for the current epoch. This mechanism would provide flexibility when participants may become disconnected from the rest of the mesh, at the potential risk of weakening forward secrecy.

4.3 Amigo Operations

We now describe the application layer operations Amigo provides, and how our session layer CGKA supports them.

Group Communication: Although nodes in a ratchet tree hold asymmetric key material, we can optimize communication by using symmetric key encryption. To send messages to the entire group a member derives the group symmetric key using the root node's

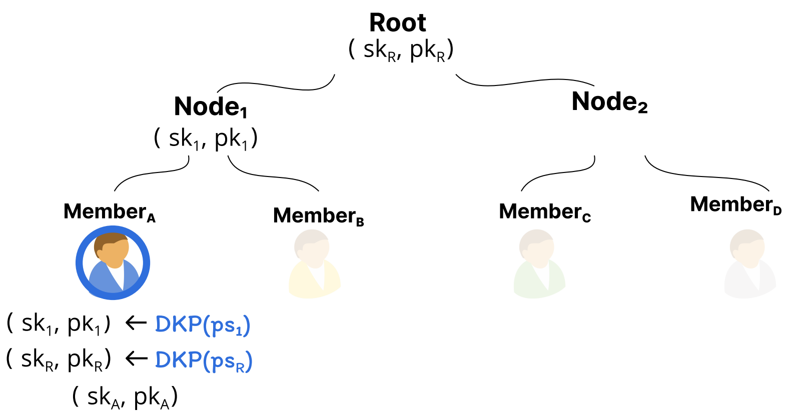

Figure 3: Creating a group in Amigo. Only Member$_A$ is in the group.

asymmetric key material. A hash of the concatenation of the secret and public keys of Root $H(sk_R||pk_R)$ is passed into a key derivation function to produce the group symmetric key. A similar process can be used for in subgroup communication. A member passes their message and this key into AES-GCM to produce the corresponding ciphertext for transmission.

If a member wishes to send a message to a subgroup they are not a part of, they may encrypt under the corresponding public key of that subgroup. Due to our updateable scheme, this allows for notions of forward secrecy and post-compromise security within group epochs, as encryption results in a new public key for the intended subgroup that can be broadcast to everyone. Decryption results in the corresponding secret key for the intended recipients.

Group Creation: In addition to the core messaging functionality, we must also consider the lifecycle of groups in Amigo. Every group in Amigo has an associated group state, consisting of public keys, secret keys, members, and the group structure. To create a group, a user generates and stores a new, empty ratchet tree of a specified size. As seen in Figure 3, the founding member Member$_A$ inserts themselves into the ratchet tree by deriving separate, randomly sampled path secrets for each ancestor node along its direct path to the root. The ancestor nodes of Member$_A$, Node$_1$ and Root, have derived keypairs (DKP) associated with path secrets ps_1 and ps_R, respectively. Thus, initial group state Ω_0 contains only key material from the founding member.

Member Addition: If Member$_A$ in Figure 3, wishes to add NewUser as Member$_B$, NewUser needs the necessary group state for a correct ratchet tree. Thus, Member$_A$ sends a *welcome message* encrypted to NewUser's public key. This communication can be bootstrapped via an in-person interaction (e.g., via a QR code), in line with how groups are formed on the frontlines during protests [ABJM21a]. Using the notation of Figure 3, the welcome message contains the group symmetric key sk_R along with the information about members, all known public keys pk_A, pk_1, pk_R within the ratchet tree, and all secrets that Member$_A$'s shares with the leftmost open node sk_1, sk_R. It is in this position that NewUser inserts themselves into the tree – becoming Member$_B$ in Figure 4 – and generates new path secrets ps_1, ps_R for each ancestor on its direct path to the root. If key material already exists for a particular node in Member$_B$'s direct path to the root, the existing material is combined with the newly derived material. In Figure 4, this means (sk_1, pk_1) and (sk_R, pk_R) are combined with (sk'_1, pk'_1) and (sk'_R, pk'_R), respectively. This results in a new epoch and fresh group state Ω_1 from Member$_B$'s perspective. For the rest of the group to derive this fresh group state, Member$_B$ must share the key material they previously generated and

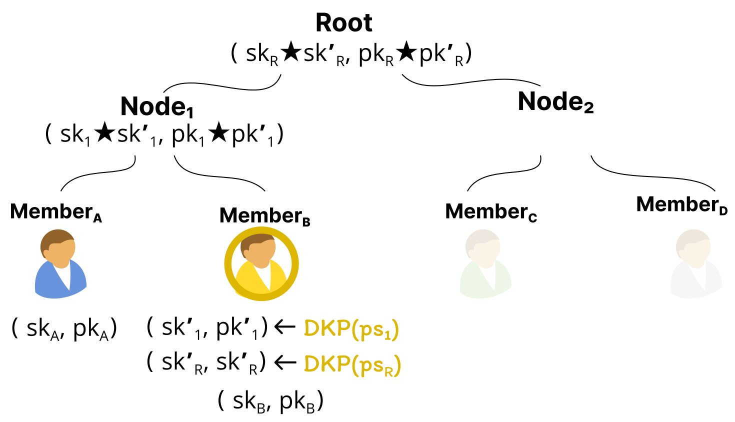
Root
(sk_R★sk'_R, pk_R★pk'_R)
Node_1
(sk_1★sk'_1, pk_1★pk'_1)
Node_2
Member_A
Member_B
Member_C
Member_D
(sk_A, pk_A)
(sk'_1, pk'_1) ← DKP(ps_1)
(sk'_R, sk'_R) ← DKP(ps_R)
(sk_B, pk_B)

Figure 4: A new user is added as $\texttt{Member}_B$ by updating the ratchet tree.

applied. We send this update to the group using the `Root` symmetric key of the previous epoch Ω_0. The path secrets are additionally encrypted under the previous public key of the corresponding ancestor node pk_1, pk_R.

As a result, each member only has the secret keys associated with their path to the root of the ratchet tree. New public keys, however, are made available to the entire group. In Figure 2, suppose $\texttt{Member}_D$ has just joined the group by processing the welcome message. After $\texttt{Member}_D$ has sent their update message, $\texttt{Member}_A$ and $\texttt{Member}_B$ can decrypt the path secret sent under `Root`'s current public key pk_R to derive the new asymmetric material sk'_R, pk'_R and combine it with the existing material: $(sk_R, pk_R) \star (sk'_R, pk'_R)$. They will also be able to combine the broadcasted public key for $\texttt{Node}_2$ with the existing key $(pk_2 \star pk'_2)$ but *not* secret keys associated with $\texttt{Node}_2$. Since $\texttt{Member}_C$ knows the complete key material held by $\texttt{Node}_2$ and `Root`, however, they will be able to decrypt both sk'_R and sk'_2 and combine it with the current secret keys.

Note that in our instantiation, member addition is most effective when members join left-to-right, i.e., the most recently joined member adds the new user; upon joining they have all the needed secrets (if they exist) for their ancestor nodes. If this order cannot be enforced, at a minimum a newly joined member will have the group secret, which they can use to communicate with the group and query nodes for the relevant secrets they are missing, as mentioned above.

Member Removal: Essential to the protest setting is efficient removal from a group, as mass arrests or police infiltration could compromise the keys of group members. In Amigo, the member initiating removal recurses to find the minimum number of subgroups such that everyone except the node being removed can receive new key material. The member then generates and encrypts the same path secret using the public key corresponding to each intended subgroup. Members decrypt the information sent to their respective subgroup and apply this updating material to every node in the ratchet tree. They also clear the state of the removed node in the ratchet tree, and initiate a new epoch under a fresh group state.

As shown in Figure 5, if $\texttt{Member}_C$ is removing $\texttt{Member}_D$, the path secret generated will be applied by $\texttt{Member}_C$ and the children of $\texttt{Node}_1$. Here our CGKA construction is particularly effective. The number of subgroups for removal is logarithmic in the group size, and our $\star$ operator enables members to update all appropriate key material using a single path secret. Thus, our CGKA allows for a quick transition from a compromised to secure state.

Key Refresh: Members can also refresh their key material. A member can generate new

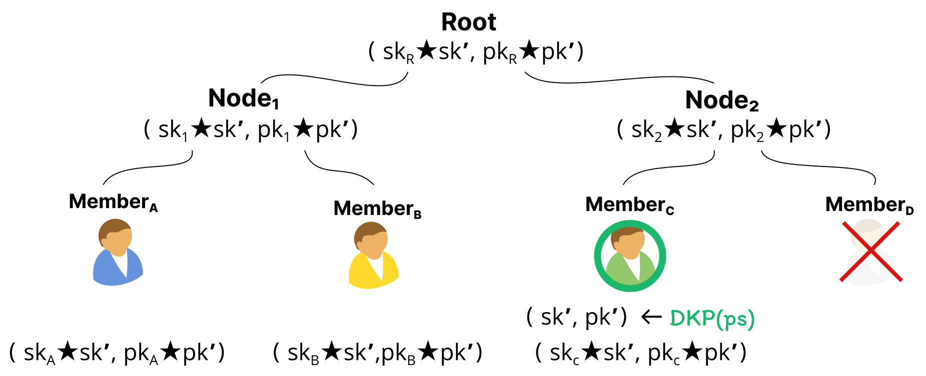

Figure 5: $\texttt{Member}_C$ removes $\texttt{Member}_B$ from the group.

path secrets for their leaf node as well as their direct path to the root, apply this material, and send the updating material to the rest of the group. When the other members apply this update, they can only decrypt path secrets for ancestors they are descendants of. Like removal, the key refresh operation provides logarithmic efficiency. Afterwards, the updating member and the entire group enter a new epoch with a fresh group state.

4.4 Security Analysis

We sketch how Amigo satisfies our security properties. Amigo achieves message confidentiality and authentication through the use of AES-GCM with the group symmetric key. These properties hold for both messages to the entire group and messages to subsets of the group (i.e., removal); messages to subsets of the group are hybrid-encrypted with the group symmetric key and the subset's public key.

We achieve forward secrecy between group states when state operations (member addition, removal, and key update) update secret keys throughout the ratchet tree; we achieve the same within a group state based on the aforementioned updateable asymmetric cryptosystem [ACDT19]. These continuous changes to secret keys ensure adversaries who obtain key material at time t cannot decipher messages sent previously.

We achieve post-compromise security through key refresh and member removal. In situations where only key material at a non-root node has been compromised, a key refresh operation is sufficient to provide post-compromise security; previous CGKA works take this approach [KPPW+21, CCG+18, CLM23, BBR18]. After compelled decryption of a mobile device, though, the adversary would hold all group cryptographic information, including the group secret at the root of the ratchet tree. In this situation, the rest of an Amigo group can recover by removing the compromised group member and continuing communication. Removed members can not decrypt the update message that initiated their removal. Thus, the new group state cannot be computed solely from data available to the compromised node, and post-compromise security is achieved here as well.

Amigo ciphertexts have no identifying metadata, and group members use trial decryption on each received message. If a member cannot decrypt a message with the keys in their ratchet tree, all the member knows is that the message is not intended for them; they learn nothing about to whom the message is destined, satisfying outsider anonymity. Intended recipients may be able to learn who other recipients are, e.g., in Figure 2, $\texttt{Member}_C$ knows a message sent to $\texttt{Node}_2$ can also be read by $\texttt{Member}_D$; however, this is outside the scope of outsider anonymity.

5 Routing

In Amigo routing, a *message* represents (1) a user-generated piece of text, or (2) cryptographic material for the CGKA protocol. Each message is tagged with a time-to-live (TTL) in seconds. A message's destination is either a group or an individual user (a size-1 group).

Each Amigo node has a *message buffer* which stores a single copy of each unique message the node has recently received. When a node wants to send a message, the node inserts the message into the buffer, evicting the oldest message in the buffer if necessary to make space; given the choice between a CGKA message and a non-CGKA message, Amigo prefers to evict the latter, as CGKA messages are important in synchronizing group states. A node periodically decrements the TTL field of all messages in the buffer, evicting messages when their TTLs reach 0.

When two nodes discover each other through physical proximity, the nodes use a *routing protocol* to determine whether and how to share their local message buffers. Amigo can use four such routing protocols – three from prior work in mesh networking, and a fourth, dynamic clique routing, which Amigo introduces, that is tailored to the mobility patterns of protests. We now describe these four in detail.

Epidemic Flooding: In this approach, each node divides time into 60-second intervals, transmitting a *beacon* at a random time within each interval to avoid Layer 2 collisions. If a peer receives such a beacon, the peer establishes a Layer 2 channel with the beacon originator, and the two nodes exchange their complete message buffers. Epidemic flooding aggressively probes the network for routes between senders and receivers, but risks network congestion due to the indiscriminate nature of message forwarding. Prior work has used epidemic flooding as a baseline for evaluating more sophisticated mesh routing protocols [PSEB22, PJW$^+$22, LFBD$^+$16, BRT23].

Digest Flooding: To prevent two peers from exchanging messages that are already present on both nodes, each peer can first exchange Bloom filters [Blo70] which summarize the locally-buffered messages; a peer X only sends a particular message to peer Y if Y's Bloom filter indicates that Y does not already store the message. Perry et al. [PSEB22] first propose this optimization to epidemic flooding, referring to a Bloom filter as a *message digest.* Details about Amigo's Bloom filter implementation can be found in Appendix C.

Static Clique Routing: Perry et al. [PSEB22] sketched an optimized form of digest routing based on *cliques.* The basic idea is that each node belongs to a clique, with one node in each clique serving as the *clique leader.* When node X wants to send a message to node Y, X sends the message to X's clique leader; the clique leader then exchanges messages with other clique leaders via digest routing, with the hope that the message will eventually reach Y's clique leader (who will then transmit that message directly to Y). Amigo implements clique-based routing by treating each Amigo group as a clique, with a group's administrator acting as the clique leader. A node's leader is static because the leader does not change in response to the spatial arrangement of nodes or other properties of the mesh network.

Intuitively, static clique routing should decrease overall network pressure compared to flooding, since only clique leaders perform bulk message exchanges. This approach might result in worse end-to-end delivery rates, though, if clique leaders do not encounter each other often, or if clique followers do not encounter their leaders often.

Dynamic Clique Routing: To decrease the likelihood of the problems mentioned in the last paragraph, Amigo introduces *dynamic clique routing.* In this approach, both a node's clique and a node's clique leader can change over time.

To determine a node's dynamic clique, Amigo divides the 2D plane into adjacent grid squares, of three meter width. A node uses GPS to identify its current grid square; all members within the same grid reside in the same dynamic clique. To determine the leader for that clique, Amigo divides time into epochs. The epoch length balances maximizing channel utilization for message delivery and ensuring swift recovery when bootstrapping is unsuccessful. For most mobility models, cliques remained largely stable over a five-minute period; therefore, Amigo uses five minute epochs.

Epochs are divided into two segments. The first minute of an epoch is reserved for electing clique leaders. The remaining segment is used to exchange messages.

Amigo's leader elections are based upon the Android Group Owner implementation [And24]. Nodes choose a "willingness" value (0-100), indicating their readiness to serve as a clique leader. The node in each given region with the highest "willingness" value is elected as the leader. Each node picks a random offset within the election period of the epoch, and generates a beacon containing the node's approximate location, the node's Layer 2 identifier, and the leader "willingness" value; the random offset decreases the likelihood of Layer 2 collisions.

A node also listens for beacons from other peers. At the end of the election period, a node selects the beacon-generating peer with the highest leader willingness in their region to be its clique leader. If a node hears no other beacons, the node becomes its own clique leader.

Under perfect conditions, all nodes in a clique will appoint the same leader—the node with the highest "willingness" value; however, if beacons are not consistently received, nodes may not elect the correct node as a leader. If this is the case, the node sits out the remainder of the epoch, and attempts to rejoin in the next iteration.

The "willingness" value is vulnerable to attack; malicious nodes could send high "willingness" values to be elected clique leaders, and subsequently disrupt traffic. While we do not address this problem in Amigo, we note that for added robustness, nodes could instead include random values in beacons to determine the leader, using methods such as distributed randomness beacons [CMB23].

Empirically, a protest may alternate between moments of tight, highly-ordered spatial density (e.g., when protesters form a human chain), interspersed with less dense spatial arrangements and/or more chaotic mobility patterns (e.g., when protesters randomly scatter due to the sudden arrival of law enforcement). We expect dynamic clique routing will better adapt to these patterns than static clique approaches.

We note our use of location as an oracle does not impact any security properties (§4.1); it is used solely to organized cliques and is not communicated across the mesh.

6 Modeling Protester Dynamics

To rigorously evaluate the performance of a secure mesh network, we need realistic models for user mobility, message traffic, and low-level network phenomena.

6.1 Mobility Models

A mobility model describes how protesters move through a physical space. The model determines how the spatial density of protesters varies over time, and how long protesters linger within the communication radii of each other's short-range radios. Accurate mobility models are crucial for evaluating mesh networks because different mobility models have a dramatic impact on when (and whether) full or partial spanning trees for routing will emerge.

Unfortunately, mobility models from prior work are not well-suited for analyzing protest dynamics. For example, Perry et al. [PSEB22] considered spatial densities of one person every 2.5-15 ft^2, with each person located at the centroid of a grid square, and with a communication radius of 10 m. selected communication radius would mean that each protester could directly contact 400 peers. A protest's high density area might have as little as $0.2\,\mathrm{m}^2$ per protester, however; if protesters were arranged in a grid, then 1,600 protesters would be within the communication range of any particular individual [Sti19]. Furthermore, in a large-scale protest, multiple kinds of mobility patterns are likely to occur through space and time, as different protesters responds to different stimuli (e.g., the arrival of police or more protesters) in different parts of the protest region.

To build more realistic mobility models, we draw insights from a crowd-sourced guide to protesting, written in 2019 by Hong Kong anti-ELAB demonstrators [Ano23]. The document provides practical advice about the planning and operational execution of a protest, describing various spatial arrangements that protesters can use. Based on this information, we create several mobility models rooted in protest scenarios, which we describe below.

Marches: A march involves a group walking together in the same direction. In our march mobility model, we assign a starting area for the march, randomly distribute nodes within the starting area, and have all nodes move in the same direction at approximately the same speed of $1.3\,\mathrm{m/s}$ [MMA+21]. The spatial density is about $2\,\mathrm{m}^2$ per protester.

Human Chains: A human chain entails a line of stationary people who pass materials from one end of the line to the other. Typically, the line begins in a storage area, and ends in front of the protest [Ano23]. In comparison to a march, there is no mobility in a human chain, and the spatial density is sparser, with nodes placed in a line about $2\,\mathrm{m}$ apart.

In our human chain mobility model, we pick two points within the simulated space, and generate a pathway between them. This pathway is not a direct line – instead, we add randomness to account for obstructions between the two points and general stochastisity in the locations of people along the line. We implement this randomness by selecting 0-10 additional random points between the two endpoints, and applying subtle curves to the paths originally generated with linear interpolation. Finally, we assign each protester to a location on the chain, spacing the protesters as above.

Gatherings: In the Hong Kong protest guide, the authors state that protests can also take the form of "gatherings with a specific focus." Examples of such gatherings are musical performances and communal decoration of building walls [Ano23]. These kinds of protests involve crowds in which individual participants generally stay within the congregation area, but can migrate within that area at walking speeds.

In our gatherings mobility model, we assign an area (roughly a city block in size) where an event will take place. We randomly assign nodes to occupy the space, and select a subset of nodes to move within the space, using a random waypoint model [Joh96] to simulate random movement.

Blockades: In a large-scale protest, a blockade occurs when protesters prevent entrance to a given space. For example, Anti-ELAB protesters blockaded the Cross-Harbor Tunnel to stop police from entering a university campus [PL19].

We implement two blockade-based mobility models. The first represents an "object blockade" in which demonstrators place objects along the blockade frontier. For example, protesters in the Hong Kong protests scattered road blocks, then set them ablaze. In our object blockade mobility model, protesters move at roughly 1.4 m/s between areas where materials are stored and the protest frontier. At each location, protesters briefly pause to pick up or drop off material.

Our "human blockade" model captures a scenario is which protesters have gathered near a blockade frontier, and are being monitored by law enforcement. In this model, clumps of police officers move towards the frontier, as do the protesters. If a police officer and a protester get within half a meter of each other for 30 seconds, one of them will retreat to the nearest clump of like-minded individuals; this behavior represents a participant being injured or physically overwhelmed by a member of the opposing group.

Static and random waypoint models: For comparison, we also implement two standard mobility models from prior literature: a static model and a random waypoint mobility model. For more details on these models, see Appendix B.

6.2 Physical Layer Simulations

Prior work [PSEB22, PJW+22, LFBD+16] uses custom network simulators to evaluate mesh routing protocols. Unfortunately, these works do not model physical-layer events like message collisions between two peers within communication range who transmit data simultaneously. Because prior simulations did not capture these Layer 1 effects, they overestimated the available bandwidth at Layers 2+.

To avoid this problem, we evaluate Amigo using ns-3, a discrete-event network simulator which *does* model important low-level characteristics. We extended ns-3 to support Wi-Fi Direct, a Layer 1+2 technology that, like Bluetooth, allows devices to communicate directly in a peer-to-peer fashion. We built our Wi-Fi Direct implementation atop ns-3's preexisting support for ad-hoc Wi-Fi; implementation details are in Appendix A. Our implementation uses 802.11ax/Wi-Fi 6, with a maximum data rate of 143.4 Mbps for advertising (MCS Index 11), and 263.3 Mbps for data exchange (MCS Index 6). This network setup is feasible on modern smartphones and representative of Wi-Fi Direct, which can provide up to 250 Mbps [All]. We limit signal propagation to 10 m, as in previous work [PSEB22].

We found ns-3 hit scaling limits as we increased the simulated protest size. For example, to complete a single run of a 250 node simulation using a static mobility model and digest routing, we required ∼10 GB of memory and one week of wall clock time, even on a high-end cloud instance (GCP `c3d-highmem-180`). Nonetheless, as we demonstrate in Section 7, even modestly-sized protests can trigger crippling network congestion in standard approaches for secure mesh routing (and sometimes in the Amigo-specific dynamic clique approach). These results suggest a variety of important areas for future work (Section 8).

6.3 Traffic Model

Our traffic model expresses the rate at which application-level behavior generates text messages and CGKA messages (§4). For a given simulation that lasts an hour of simulated time, we generate 20 CGKA messages total, with each one sent from a random group member and sent at a random time in the first half of the simulation; the latter constraint ensures the message has sufficient time to propagate to the other group members before the simulation ends. This is roughly equivalent to each group adding one member, removing one member, and performing one key refresh during a protest.

Non-CGKA traffic corresponds to standard text messages sent to a group. We implement a basic traffic model for non-CGKA messages: each node in the simulation sends a message every minute. We choose these values based upon the protester guide, which states chats see an excess of 100 messages per minute during an event's peak [Ano23].

To generate the senders and receivers for messages, we create group chats of 25 members, based on nodes' locations: we assume 0.8 of a clique is nearby the node, and select the other 0.2 randomly. We imagine many members of a close-knit group chat may enter the protest together, and a fraction may attend separately, or remain home or nearby to perform wellness checks. Mobility models may cause group members to move physically apart from one another during the simulation; for example, to carry supplies, talk to someone new, etc. We assume people within a group chat send messages to each other. Each message is 250 bytes large and has a TTL of 5,000 seconds—slightly less than the length of an average action [Ano23].

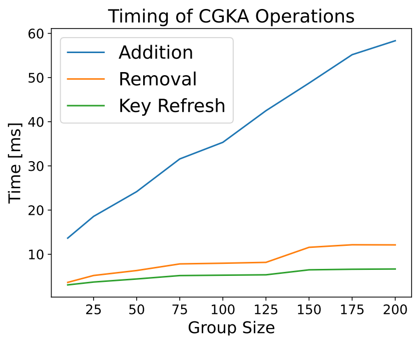

Figure 6: Timing measurement of Amigo state operations.

Table 2: Operation timings on representative message sizes.

Operation	Size: 250 B	Size: 2 MB	Size: 10 MB
Encryption	78.45 µs	57.517 ms	311.24 ms
Decryption	69.08 µs	23.677 ms	282.38 ms

7 Results

7.1 CGKA Microbenchmarks

Amigo is intended for deployment atop resource-constrained devices. To evaluate the resource consumption of Amigo's CGKA, we ran microbenchmarks on a Raspberry Pi 4 with a 1.8 GHz quad-core ARMv8 SoC and 8 GB RAM: similar specifications to those of a low-end modern smartphone. It also provided a convenient environment for low-level benchmarking. We used the Criterion [cri21] for the benchmarks, running each for a minimum of 1000 iterations.

Are CGKA operations fast? As discussed in Section 2.1, key refreshing and the removal of a group member have to be fast: when a group member is compromised, e.g., because their phone is confiscated by authorities, all other group members are at risk. Figure 6 shows, in a 200 person group, Amigo can perform member removal and key refreshing in under 15 ms. Group addition is much slower, but adding a new member to a group is not as critical as removing a compromised one; furthermore, we expect additions will be infrequent. At a group size of 200, our test device could execute roughly 82 member removals, 150 key refreshes, or 17 member additions in one minute. See Appendix D for more detailed results.

What is the CGKA-induced network load? For non-CGKA messages, Amigo is more efficient than pairwise group approaches [Sig14] where the same message is sent encrypted multiple times, once per each symmetric key negotiated with an individual group member. For group management operations, Amigo generates one message for each member removal or key refresh. Member addition requires two messages: a welcome message to bootstrap in the new member, and an update message that brings the group into a new epoch with updated group state. See Appendix D for more detailed results.

Is Amigo battery efficient? Appendix D contains a full discussion. For now, we note Amigo's energy consumption is reasonable and will not unduly drain a phone's battery.

Table 3: Average packets sent of various type across various routing mechanisms; results averaged across
3 runs of static mobility models for 5,000 KB buffer size. Packet delivery rate in parentheses.

	Epidemic Flooding	Digest Flooding	Static Clique Routing	Dynamic Clique Routing
Messages	422,623,852 (0.21)	411,590,675 (0.31)	2,030,682 (1.0)	13,522,625 (0.98)
Digests	-	24,087,630 (0.95)	472,836 (1.0)	776,972 (1.0)
Willingness Beacons	-	-	-	113,859
Message Beacons	-	-	-	4,800
Link Layer Beacons	6,000	6,000	6,000	1,200
Probe Requests	575,236 (1.0)	574,816 (1.0)	11,258 (1.0)	114,359 (1.0)
Probe Responses	574,064 (1.0)	573,609 (1.0)	11,258 (1.0)	113,875 (1.0)
Total	423,779,152	436,832,727	2,532,033	14,647,688

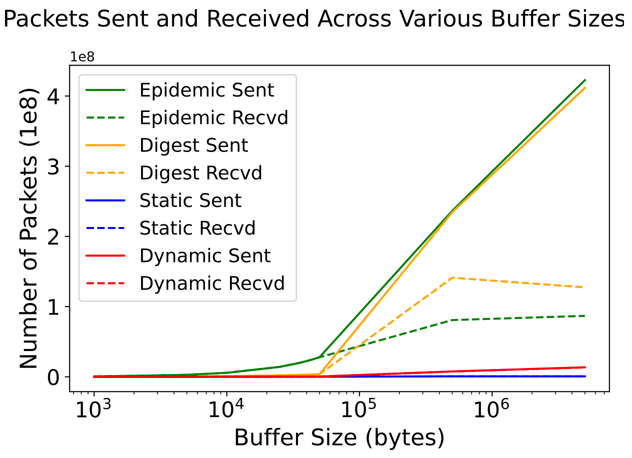

Figure 7: The total number of packets sent and received routing protocols, for 100 nodes in a static mobility model; average of 3 runs.

7.2 Network Congestion and Stability

How does buffer size impact network saturation? Figure 7 shows the packet volume generated by various buffer sizes, with all other aspects of the simulation held constant (e.g., traffic model, mobility model, etc.). Table 3 provides a breakdown of the various packet types sent, and their end-to-end delivery rates, given a 5,000 KB buffer.

As expected, flooding generates significantly more traffic than other approaches; as buffer size increases, the difference grows (Figure 7). The bulk of the traffic arises from messages and digests, not from Layer 2 management packets (i.e., beacons, probe requests, and probe responses).

Notably, end-to-end delivery rates for epidemic routing and digest routing are very low (e.g., ∼30% for digest routing). The reason is that both schemes gradually congest the network and then make no attempt to reduce their sending rate, even once message buffers have become steady-state full. When buffers become full and sending rates do not back off, new messages are constantly added to buffers, reducing the effectiveness of the digest optimization. As a result, nodes often exchange complete or near-complete copies of their local buffers, resulting in larger Layer 2 messages and more Layer 1 collisions (triggering retransmissions of the associated Layer 2 messages). The net result is, as time progresses, the expected delivery rate (Figure 8) for packets sent at time $t + 1$ trend monotonically lower than the expected delivery rate for packets sent at time t.

The relatively poor performance of digest routing is surprising, given previous work [PSEB22] introduced digest-based routing as an optimization. However, by modeling realistic Layer 1 and Layer 2 effects, we see with epidemic routing, only message packets collide with

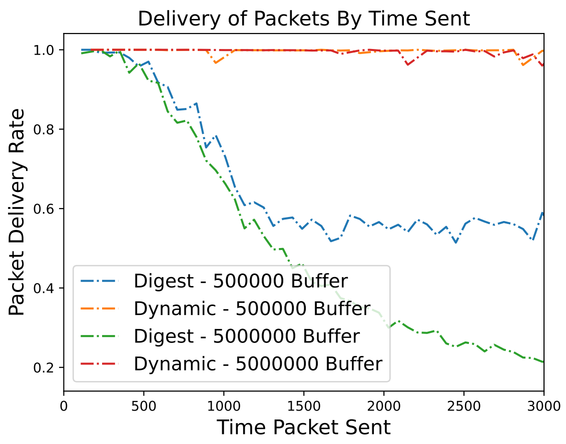

Figure 8: Average delivery rates for packets sent at time t for our digest routing and dynamic routing, for 100 nodes in a static mobility model.

themselves, whereas with digest routing, message packets *and digest packets* collide, exacerbating the effects of network saturation. Thus, as shown in Table 3, digest routing's performance is not superlatively better than that of vanilla epidemic routing.

Also note that the level of network saturation in Table 3 represents a scenario in which each Layer 1 collision domain contains 100 protesters. A high-density protest could have as little as $0.2\,\mathrm{m}^2$ per protester [Bre], meaning 1,600 protesters would be in a single collision domain. Such large protests will suffer from even more intense congestion collapse.

How does buffer size impact end-to-end delivery rates? All of Amigo's routing schemes use per-node buffers to hold the messages nodes exchange with each other. Given a fixed mobility pattern and traffic matrix, decreasing the buffer sizes will result in fewer messages exchanged when nodes encounter each other, resulting in less network congestion; however, due to contention for buffer space, older messages (i.e., messages with lower TTLs) in a buffer will be evicted more aggressively to make room for new messages, potentially reducing how often messages are shared (and thus potentially reducing end-to-end message delivery rates). Conversely, increasing the buffer size will give messages more opportunities to be transmitted, potentially increasing message delivery rates; however, the resulting increase in traffic volume may induce network congestion that *decreases* end-to-end delivery rates. The tensions between increased congestion and increased delivery rates materialize in different ways for different protest environments.

Figure 9 is a time-series view of the number of messages evicted from node buffers during simulations using the gather mobility model with dynamic routing or digest routing. For 5,000 KB buffers, no evictions occur. For 500 KB, buffer evictions begin to increase at approximately $t = 1200$. Eviction rates remain high, with the exception of a brief hiatus for dynamic routing during advertising periods.

This eviction behavior is relevant to Figure 10, which shows the delivery rates for packets sent at various time intervals. Figure 10 shows that delivery rates are initially high across all routing mechanisms for buffer sizes of 500 KB and 5,000 KB. However, at approximately $t = 1200$, delivery rates become consistently poor. The reason is, once buffer evictions start at $t = 1200$, the effective number of transmission chances a message gets will decrease, hurting the message's likelihood of being end-to-end delivered. For the simulation settings in Figure 10, the network has not become fully utilized before buffer evictions start occurring; this means, for the gather mobility pattern examined in Figure 10, a buffer size of 500 KB is unable to fully utilize the available network bandwidth, resulting in unnecessary degradation of end-to-end delivery rates. The overall takeaway is that, even

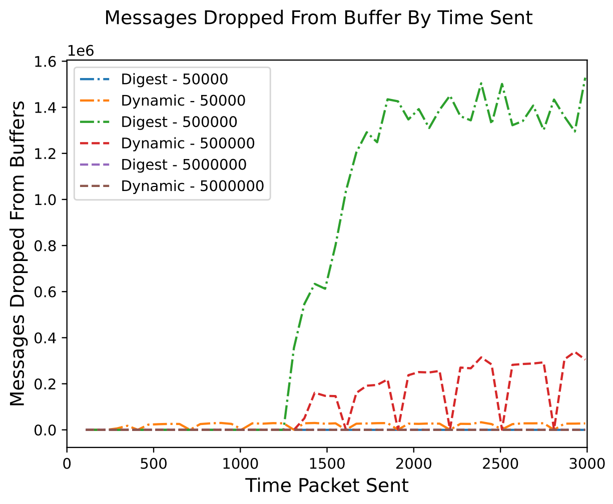

Figure 9: Number of messages evicted in digest and dynamic mechanisms; 100 nodes in gather mobility model, bucketed into 60 second intervals.

Table 4: Total message packets sent for all routing mechanisms and mobility models; results averaged across 3 runs for 5,000 KB buffer size. Message delivery rate in parentheses. Checkmarks if CGKA group maintained at the end of the simulation.

	Epidemic Flooding	Digest Flooding	Static Clique Routing	Dynamic Clique Routing
static	422,613,855 (1.000) ✓	411,579,195 (1.000) ✓	596,074 (0.705) ✗	13,555,092 (1.000) ✓
random	425,916,153 (1.000) ✓	420,206,611 (1.000) ✓	2,073,922 (0.980) ✗	8,626,361 (0.963) ✗
gather	416,506,046 (1.000) ✓	405,078,143 (1.000) ✓	1,964,185 (0.941) ✗	12,956,503 (1.000) ✓
chain	34,706,437 (1.000) ✓	34,732,147 (1.000) ✓	36,241 (0.017) ✗	14,378,903 (0.999) ✓
march	50,609,963 (0.867) ✗	50,325,119 (0.867) ✗	29,145 (0.026) ✗	6,208,141 (0.726) ✗
blockade1	11,073,410 (0.271) ✗	11,079,296 (0.271) ✗	18,214 (0.008) ✗	87,310 (0.025) ✗
blockade2	435,542 (0.033) ✗	434,633 (0.033) ✗	8,286 (0.005) ✗	12,380 (0.006) ✗

for Amigo's protest-optimized dynamic routing, the lack of on-the-fly adjustment to buffer dynamics and network congestion leads to non-optimal routing performance. We discuss this topic further in Section 8.

7.3 Protester Mobility and Network Partitioning

How does protester mobility impact message delivery? Table 4 explores the number of packets send and message delivery rates for the routing protocols and various mobility patterns when using a 5,000 KB buffer. Simulations ran for $3,600$ seconds, but Table 4 only considers messages sent before $t = 3,000$ to not mark as dropped a message that was in-transit at the end of the simulation and would have eventually been delivered if the simulation had continued. Table 4 does include the "warm-up" period at the beginning of the simulation, because a warm-up phase would also occur at the beginning of a real protest.

As shown in Table 4, static clique routing was the worst performing approach across all mobility patterns. The reasons were that (1) a clique leader often had trouble encountering its clique members or other clique leaders, and (2) peers had no way to attach to a better-suited leader. Ignoring the blockade mobility patterns for a moment, epidemic routing and digest routing outperformed dynamic clique routing because those algorithms did not trigger congestion collapse for the mobility patterns and node counts explored in the simulations; as a result, the more aggressive forwarding behavior of these algorithms

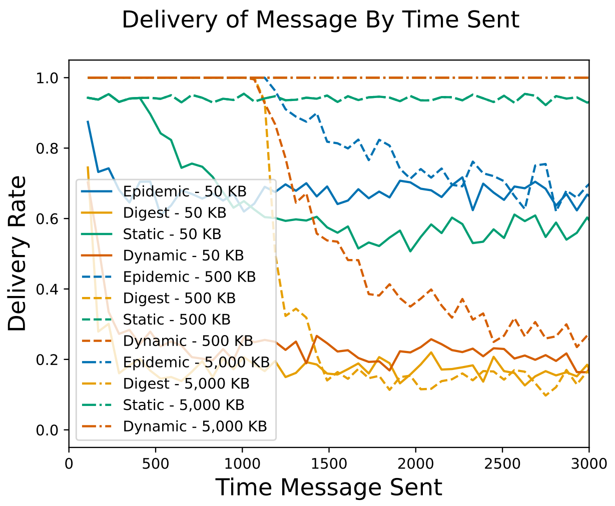

Figure 10: Message delivery for the gather mobility model; initial send times for messages on the x-axis, bucketed into 60 second intervals.

(compared to that of dynamic clique routing) led to better end-to-end delivery rates.

Static clique routing's performance for the chain and march patterns were essentially the same. For the other three routing protocols, the march model was more challenging than the chain model; this is because while the chain model is static, the march model is dynamic, making message transmission and clique dynamics harder to maintain.

Table 4 shows that all four routing protocols struggled with the blockade mobility patterns. In blockade1, protesters retrieved materials to form a barrier made of physical objects, whereas in blockade2, protesters generally approach a police line, and then spread out due to police intimidation. Both models had low spatial density, with nodes often ten meters apart or more, reducing chances for nodes to hear beacons (or exchange messages more generally).

Overall, we note that our new dynamic clique routing protocol achieves 95% or higher delivery rate on multiple mobility models (including some of the protester-oriented ones) while sending far fewer messages over the course of the simulation. This differences ranges from ∼2× to ∼50×, depending on the mobility model evaluated. In other words, our simulations show that dynamic clique routing can achieve similar levels of message delivery at a fraction of the messages sent – important for smartphones, which are constrained by battery size and CPU capability, as well as for overall network stability, as discussed in Section 7.2.

Can protester groups maintain key agreement? Amigo's CGKA protocol offers strong security in theory; however, those protections are moot in practice if Amigo nodes struggle to maintain CGKA key state due to an unreliable routing layer. Amigo's CGKA protocol tolerates out-of-order messages (§4.2), but it does not tolerate a group member *never* receiving a CGKA message—in this scenario, the member will be unable to continue in the group. As discussed in Section 5, Amigo prioritizes the delivery of CGKA messages, but is this policy sufficient to allow the CGKA protocol to succeed in practice?

Table 4 shows that dynamic clique routing achieves comparable message delivery rates with far fewer messages, and achieves CGKA convergence for most of the same cases as the flooding protocols. Unlike the flooding protocols, it does not do so for the random mobility model. As noted in Section 4.2, Amigo's CGKA assumes eventual consistency; flooding protocols are highly redundant, so they are more likely to achieve this state (i.e., perfect CGKA message delivery) with enough time. But, dynamic clique routing still has a relatively high delivery rate, which provides the opportunity to apply the the application-level healing mechanism described in Section 4.2. Even if a member node misses some CGKA messages, any other member nodes available could help recover missing group state. We believe decentralized CGKAs that are robust to low message delivery rates is an interesting direction for future work.

8 Discussion

Secure communication protocols require participants to create and destroy cryptographic keys. Messages associated with key management are exchanged over the same network that handles regular message traffic. Thus, the reliability of the network impacts the reliability of key management. In traditional (i.e., non-ad-hoc) networks, a Layer 4 secure messaging protocol like TLS can rely on the Layer 3 TCP protocol to provide dependable, in-order message delivery and avoid network congestion. However, the ephemeral, unpredictable nature of node connectivity in a mesh network makes it hard for Layer 3+ protocols to understand (and react to) network dynamics like congestion in real time. Prior work on secure mesh routing has focused on the details of key management, and made simplifying assumptions about the behavior of the network layer (§6.2). However, our results in Section 7 demonstrate that low-level network behavior has a critical impact on the performance and correctness of a secure mesh network. For example, Layer 1 phenomena like collisions and multi-path interference will frequently be triggered in real-life protest situations involving densely-packed people and physical obstacles like buildings. Furthermore, Layer 2 network partitions in the peer-to-peer spanning tree can also be triggered by realistic movement patterns. For example, protesters may gather in one place and then separate into groups with distant centroids; later, the members in a particular group may scatter due to engagements with law enforcement. The limited channel count of real-life peer-to-peer radio technologies like Wi-Fi Direct also limits the scalability of network bandwidth, reducing the effectiveness of the mesh network. These factors suggest secure mesh networking protocols which appear feasible when deployed atop well-behaved networks may in fact perform badly in realistic scenarios. Optimizing cryptographic protocols is not enough; we must also complement improved cryptographic schemes with network-layer innovations.

Our results (§7) suggest important future research:

- congestion detection and avoidance for mesh networks,
- key management protocols more tolerant to packet loss,
- adaptive routing protocols which modify forwarding in response to dynamic estimates of global properties, and
- user-facing feedback mechanisms that allow users to understand current network dynamics and possibly adapt user-level behavior (e.g., by not sending video messages when high network congestion is detected).

Properly evaluating these approaches for protests of larger size will also require fundamental optimization work involving network simulators like ns-3. ns-3 is a discrete event simulator; a priority queue holds network events (e.g., corresponding to packet transmission and reception), such that the main simulation loop iteratively pops the event with the earliest timestamp and executes the callback function associated with the event (possibly triggering the insertion of new events into the priority queue). ns-3 currently uses standard C++ data structures for priority queues [321], but our experience is that stock data structures prevent ns-3 from scaling to simulation protests beyond a few hundred nodes – ns-3 either runs out of memory and crashes, or cannot finish a simulation despite running for days of wall-clock time. Optimizing ns-3's use of memory and compute, e.g., Devastator-style techniques [BYJ+24], is crucial for understanding how secure mesh networks behave in practice.

9 Conclusion

We present Amigo, which combines continuous group key agreement and a new, clique-based routing protocol to provide a novel mesh messaging system for protesters. We evaluate Amigo using detailed simulations that consider realistic protester mobility models and realistic low-level network phenomena. Our results demonstrate substantial improvements over prior work, but suggest that further efforts are needed at lower layers in the protocol stack to support strong privacy and efficient message routing.

Acknowledgments

This work is supported by National Science Foundation (NSF) award 1955172, an NSF Graduate Research Fellowship, a Sui Foundation Academic Research Award, a PSC-CUNY award, and a Google Cyber NYC award. This work also used Jetstream2 computing resources through allocations from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by NSF awards 2138259, 2138286, 2138307, 2137603, and 2138296. The authors would also like to thank Rosario Gennaro for contributing cloud computing resources.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors. Any mention of specific companies or products does not imply any endorsement by the authors, by their employers, or by the sponsors.

References

[321] ns 3. Events and Simulator. https://www.nsnam.org/docs/manual/html/events.html#scheduler, 2021. Online; accessed 1 Sept 2024.

[ABJM21a] Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. Collective information security in large-scale urban protests: the case of hong kong. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 3363–3380. USENIX Association, August 2021.

[ABJM21b] Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. Mesh messaging in large-scale protests: Breaking bridgefy. In *Topics in Cryptology – CT-RSA 2021: Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17–20, 2021, Proceedings*, page 375–398, Berlin, Heidelberg, 2021. Springer-Verlag.

[Acc24a] Access Now. Myanmar's iron curtain: internet shutdowns and repression in 2023. https://www.accessnow.org/press-release/myanmar-keepiton-internet-shutdowns-2023-en/, May 2024. Online; accessed 1 Sept 2024.

[Acc24b] Access Now. Shrinking democracy, growing violence: Internet shutdowns in 2023. https://www.accessnow.org/wp-content/uploads/2024/05/2023-KIO-Report.pdf, May 2024. Online; accessed 1 Sept 2024.

[ACDT19] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Security analysis and improvements for the IETF MLS standard for group messaging. Cryptology ePrint Archive, Paper 2019/1189, 2019.

[AG15] Nezar AlSayyad and Muna Guvenc. Virtual uprisings: On the interaction of new social media, traditional media coverage and urban space during the 'arab spring'. *Urban Studies*, 52(11):2018–2034, 2015.

[All] Wi-Fi Alliance. Wi-fi direct.

[All21] Wi-Fi Alliance. Wi-Fi Direct Specification. https://www.wi-fi.org/file/wi-fi-direct-specification, 2021. Online; accessed 1 Sept 2024.

[Amn20] Amnesty International. Iran: Internet deliberately shut down during November 2019 killings – new investigation. https://www.amnesty.org/en/latest/press-release/2020/11/iran-internet-deliberately-shut-down-during-november-2019-killings-new-investigation/, November 2020. Online; accessed 1 Dec 2023.

[And24] Android Developers. WifiP2pConfig. https://developer.android.com/reference/android/net/wifi/p2p/WifiP2pConfig, 2024. Online; accessed Oct 2024.

[Ano23] Anonymous Authors. The HK19 Manual - Part 2B: How Tos. https://docs.google.com/document/d/1UROUN37_gUqrDd4FYDFYXQAmYuXtsBAfaDLo47Im-Kk/edit#heading=h.9n9f9eiq3xhs, 2023. Online; accessed 1 Dec 2023.

[AW09] Ian F Akyildiz and Xudong Wang. *Wireless mesh networks*. John Wiley & Sons, 2009.

[AWW05] Ian F Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Computer networks*, 47(4):445–487, 2005.

[BBR18] Karthikeyan Bhargavan, Richard Barnes, and Eric Rescorla. TreeKEM: Asynchronous Decentralized Key Management for Large Dynamic Groups A protocol proposal for Messaging Layer Security (MLS). Research report, Inria Paris, May 2018.

[BeW19] Bewater:seven tactics that are winning hong kong's democracy revolution. https://www.newstatesman.com/politics/2019/08/be-water-seven-tactics-that-are-winning-hong-kongs-democracy-revolution-2, 2019.

[Blo70] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[Bre] Robert M. Brecht. Festival and Concert Production: Crowd Safety. https://tseentertainment.com/festival-and-concert-production-crowd-safety/. Online; accessed 1 Sept 2024.

[Bri22] Bridgefy. Bridgefy Messaging App - Offline Messaging. https://bridgefy.me, 2022. Online; accessed 1 Dec 2023.

[BRT23] Alexander Bienstock, Paul Rösler, and Yi Tang. ASMesh: Anonymous and secure messaging in mesh networks using stronger, anonymous double ratchet. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1–15. ACM Press, November 2023.

[BS16] Girish Bekaroo and Aditya Santokhee. Power consumption of the raspberry pi: A comparative analysis. In *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, pages 361–366, 2016.

[BYJ+24] John Bachan, Jianlan Ye, Xuan Jiang, Tan Nguyen, Mahesh Natarajan, Maximilian Bremer, and Cy Chan. Devastator: A scalable parallel discrete event simulation framework for modern c++. In *Proceedings of the 38th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 35–46, 2024.

[CCG+18] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1802–1819. ACM Press, October 2018.

[Cel15] Department of justice policy guidance: Use of cell-site simulator technology. https://www.justice.gov/opa/file/767321/dl, 2015.

[CG18] Darin Christensen and Francisco Garfias. Can you hear me now? how communication technology affects protest and repression. *Quarterly journal of political science*, 13(1):89, 2018.

[CLM23] Céline Chevalier, Guirec Lebrun, and Ange Martinelli. Quarantined-TreeKEM: a continuous group key agreement for MLS, secure in presence of inactive users. Cryptology ePrint Archive, Paper 2023/1903, 2023. https://eprint.iacr.org/2023/1903.

[CMB23] Kevin Choi, Aathira Manoj, and Joseph Bonneau. Sok: Distributed randomness beacons. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 75–92. IEEE, 2023.

[cri21] Criterion.rs documentation. https://bheisler.github.io/criterion.rs/book/criterion_rs.html, 2021.

[Ele23] Electronic Frontier Foundation. CELL-SITE SIMULATORS/ IMSI CATCHERS. https://sls.eff.org/technologies/cell-site-simulators-imsi-catchers, March 2023. Online; accessed 1 Sept 2024.

[FP12] N. Fazio and I.M. Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. In *IACR Public Key Cryptography—PKC '12*, pages 225–242, Heidelberg, 2012. Springer. LNCS 7293.

[HBDF+13] Shaddi Hasan, Yahel Ben-David, Giulia Fanti, Eric Brewer, and Scott Shenker. Building dissent networks: Towards effective countermeasures against {Large-Scale} communications blackouts. In *3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI 13)*, 2013.

[Joh96] D Johnson. Dynamic source routing in ad hoc wireless networks. *Mobile Computing/Kluwer Academic Publishers*, 1996.

[KPPW+21] Karen Klein, Guillermo Pascual-Perez, Michael Walter, Chethan Kamath, Margarita Capretto, Miguel Cueto, Ilia Markov, Michelle Yeo, Joël Alwen, and Krzysztof Pietrzak. Keep the dirt: Tainted treekem, adaptively and actively secure continuous group key agreement. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 268–284, 2021.

[LA10] Jeroen Van Laer and Peter Van Aelst. Internet and social movement action repertoires. *Information, Communication & Society*, 13(8):1146–1171, 2010.

[Lee20] Francis L. F. Lee. Solidarity in the anti-extradition bill movement in hong kong. *Critical Asian Studies*, 52:18 – 32, 2020.

[LFBD+16] Ada Lerner, Giulia C. Fanti, Yahel Ben-David, Jesus Garcia, Paul Schmitt, and Barath Raghavan. Rangzen: Anonymously getting the word out in a blackout. *ArXiv*, abs/1612.03371, 2016.

[LQK09] Xiangfang Li, Lijun Qian, and Joseph Kamto. Secure anonymous routing in wireless mesh networks. In *2009 International Conference on E-Business and Information System Security*, pages 1–5. IEEE, 2009.

[LYTC19] Francis Lee, Samson Yuen, Gary Tang, and Edmund Cheng. Hong kong's summer of uprising: From anti-extradition to anti-authoritarian protests. *China Review*, 19:1–32, 11 2019.

[Mas23] Stephen Mash. What is wifi direct and its benefits, January 2023.

[MMA+21] Elaine M Murtagh, Jacqueline L Mair, Elroy Aguiar, Catrine Tudor-Locke, and Marie H Murphy. Outdoor walking speeds of apparently healthy adults: A systematic review and meta-analysis. *Sports Medicine*, 51:125–141, 2021.

[PAC14] Abhinav Prakash, Dharma P Agrawa, and Yunli Chen. Network coding combined with onion routing for anonymous and secure communication in a wireless mesh network. *International journal of Computer Networks & Communications (IJCNC)*, 6(6):1–14, 2014.

[PJW+22] Amogh Pradeep, Hira Javaid, Ryan Williams, Antoine Rault, David Choffnes, Stevens Le Blond, and Bryan Alexander Ford. Moby: A blackout-resistant anonymity network for mobile devices. *Proceedings on Privacy Enhancing Technologies*, 2022(3):247–267, 2022.

[PL19] Jessie Pang and Kate Lamb. Highway blockade reveals splits in Hong Kong protest movement. https://www.reuters.com/article/world/highway-blockade-reveals-splits-in-hong-kong-protest-movement-idUSKBN1XP06R/, November 2019. Online; accessed 1 Dec 2023.

[Pon24] Poniie pn2000 plug-in kilowatt electricity usage monitor electrical power consumption watt meter w/ extension cord. https://www.poniie.com/products/6, 2024.

[PSEB22] Neil Perry, Bruce Spang, Saba Eskandarian, and Dan Boneh. Strong anonymity for mesh messaging. *arXiv preprint arXiv:2207.04145*, 2022.

[RL08] Kui Ren and Wenjing Lou. A sophisticated privacy-enhanced yet accountable security framework for metropolitan wireless mesh networks. In *2008 The 28th International Conference on Distributed Computing Systems*, pages 286–294. IEEE, 2008.

[RYLZ09] Kui Ren, Shucheng Yu, Wenjing Lou, and Yanchao Zhang. Peace: A novel privacy-enhanced yet accountable security framework for metropolitan wireless mesh networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(2):203–215, 2009.

[Sen12] Jaydip Sen. Secure and privacy-preserving authentication protocols for wireless mesh networks. In *Applied Cryptography and Network Security*, pages 3–34. IntechOpen, 2012.

[Shi11] Clay Shirky. The political power of social media: Technology, the public sphere, and political change. *Foreign Affairs*, 90:28–41, 2011.

[Sig14] Signal: Private group messaging. https://signal.org/blog/private-groups/, 2014.

[Sig20] The double ratchet algorithm. https://signal.org/docs/specifications/doubleratchet/#symmetric-key-ratchet, 2020.

[SS14] Nazatul Haque Sultan and Nityananda Sarma. Papar: Pairing based authentication protocol with anonymous roaming for wireless mesh networks. In *2014 International Conference on Information Technology*, pages 155–160. IEEE, 2014.

[Sti19] G. Keith Still. Standing Crowd Density. https://www.gkstill.com/Support/crowd-density/CrowdDensity-1.html, February 2019. Online; accessed 1 Oct 2024.

[Wei19] Matthew Weidner. Group messaging for secure asynchronous collaboration. Master's thesis, University of Cambridge, 2019.

[WKHB21] Matthew Weidner, Martin Kleppmann, Daniel Hugenroth, and Alastair R. Beresford. Key agreement for decentralized secure group messaging with strong security guarantees. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 2024–2045, New York, NY, USA, 2021. Association for Computing Machinery.

[ZJG21] Maximilian Zinkus, Tushar M. Jois, and Matthew Green. Sok: Cryptographic confidentiality of data on mobile devices. *Proceedings on Privacy Enhancing Technologies*, 2022:586 – 607, 2021.

[ZLH06] Yan Zhang, Jijun Luo, and Honglin Hu. *Wireless mesh networking: architectures, protocols and standards.* CRC Press, 2006.

A Simulation Details

Changes to simulate Wi-Fi Direct: In order to implement more than one channel in ns-3 on each device under Ad Hoc Wi-Fi, we create a network interface for each channel on each given node (representing an individual's device). This means that nodes have more than one identifier (i.e. IP address), as they have an identifier for each interface. However, this means that we need to bind an IP address to the node's MAC address in our simulations. We therefore implement an "address discovery" period prior to simulation start. During this time, each node sends a message on each channel to each other node. This allows for the nodes to exchange ARP packets, and learn each others' MAC addresses. We modify the ARP cache expiration values to last the entire duration of the simulation; in this way, we account for any potential discrepancies arising from our implementation.

Wi-Fi Direct Protocol Structure: At a high level, the Wi-Fi Direct protocol entails two phases: the discovery phase and the data exchange phase.

In the discovery phase, devices exchange information on advertisement channels about how to proceed with data exchange. In line with the WiFi Direct [Mas23], we use the

2.4 GHz band for the advertisement channels, with a channel width of 20 MHz. We implement three advertising channels, the standard number of non-overlapping channels for this scenario (channels 1,6, and 11).

We simulate the discovery phase, through sending advertising-related packets; we select reasonably representative packet sizes through examining the Wi-Fi and Wi-Fi Direct specifications [All21], and determining which conditional parameters would likely be included for a basic case. While we predetermine the parameters associated with communication for our simulations, in true Wi-Fi Direct, the information in these packets allows for communication channels and rates, SSIDs, and network parameters to be negotiated. Nodes advertise by sending beacon packets (of size 155 bytes) on these channels at random offsets every minute. When a node receives a beacon packet, it sends a probe request packet (of size 124 bytes) to the original sender. Then, the original sender sends a probe response packet (of size 199 bytes). In this process, various set up procedures are established, for example, a channel for future data exchange communication is negotiated.

Then, the data exchange phase may begin. In the data exchange phase, devices exchange data on a selected channel (in our case, exchanging messages from their message buffers). Data exchange could occur in either the 2.4 GHz band or the 5GHz band. We choose to implement it in the 5 GHz band, with a channel width of 80 MHz. We implement three data exchange channels; nodes select a channel to exchange data on randomly, determined during the probe request/response process.

B Standard Mobility Models

Though they standard mobility models are in network evaluation space, we provide some context surrounding static and random waypoint mobility models.

The *static model* entails placing nodes a set distance apart in a grid, where they remain for the duration of the simulation. This is the system used in previous work [PSEB22]. We use this model to represent a baseline, to help us understand the impact of movement. In our simulation runs, we place protesters 1m apart.

In the *random waypoint model*, each node first pauses for selected amount of time, then selects a random location, and a random speed at which to move to that location. Once the node has reached the selected location, it again pauses, before repeating the process. We use this model to represent a baseline for movement (i.e., when no protest events are occurring).

There are two parameters we must set: (1) the range of possible node speeds and (2) the node pause lengths. For our simulation, we set the range of node speeds to resemble the general range of human mobility from a very slow walk (1.5 km/h) to a quick run (13.5 km/h). We distribute these speeds skewing towards lower speeds, following log-normal distribution, centered at 1.3m/s (average walking speed). For the node pause length, we set a range of 0-50 seconds.

C Bloom Filters

Our Bloom filters are formatted as bit arrays. The size of our Bloom filters are determined by the size of the buffer it represents; we maintain a consistent hash count (n=17) and false positive rate (0.00001). For example, a 50 KB message buffer has a .4KB Bloom filter digest, a 5,000 KB message buffer has a 479 KB Bloom filter digest.

Table 5: Microbenchmarks of state operations.

Size	Addition		Removal		Key Refresh	
	Time (ms)	CS (kB)	Time (ms)	CS (KB)	Time (ms)	CS (KB)
10	13.652	21.721	3.634	3.046	3.060	3.517
25	18.565	47.147	5.193	3.805	3.711	3.988
50	24.173	89.281	6.316	4.565	4.405	4.471
75	31.578	131.612	7.816	5.331	5.174	4.933
100	35.339	173.581	7.971	5.307	5.263	4.933
125	42.494	215.304	8.163	5.331	5.347	5.331
150	48.743	257.453	11.570	6.078	6.475	5.356
175	55.178	299.696	12.146	6.094	6.605	5.344
200	58.334	341.517	12.119	6.042	6.664	5.402

Table 6: Power consumption (measured in Watts) and energy consumption (measured in Joules) of Amigo state operations.

	Baseline		Addition		Removal		Refresh	
	PC	EC	PC	EC	PC	EC	PC	EC
Idle	2.39	143.40	4.05	243.00	4.22	253.20	4.33	259.80
Video	3.97	238.20	4.88	292.80	5.00	300.00	4.83	289.80
Browsing	4.56	273.60	5.13	307.80	5.18	310.80	5.40	324.00

D Additional CGKA Microbenchmarks

Timing benchmarks: In Table 5, we display each state operation (addition, removal, key refresh), and the corresponding time measurements in milliseconds, ciphertext sizes in kilobytes. The number of messages required be sent over the network for addition, removal, and key refresh are two, one, and one respectively.

Power and energy benchmarks: Smartphones have limited battery life, so Amigo should also be power efficient. In practice, smartphones run several processes in parallel, so we measure the additional energy consumed when running Amigo alongside other tasks. Taking inspiration from prior work [BS16], we choose the following tasks: idling, web browsing (we use Reddit.com), and watching a video (we use a 480p YouTube video).

Our measurement setup consisted of the Raspberry Pi attached to a display via (micro-)HDMI and a mouse and keyboard connected via USB. Power consumption was measured every second using the Poniee PN2000 wattmeter [Pon24]. We note that, since the Raspberry Pi is not optimized for battery usage, our measurements reflect an upper bound.

First, we performed each task for 1 minute to determine baseline power consumption. Afterwards, we used Criterion to run Amigo operations for a group size of 200 members alongside each task. We measured the power consumption every second, and calculated the combined energy consumed in Joules. We then calculated the average energy consumed by Amigo operations by by subtracting the combined measurement from the baseline, and then dividing by the number of Amigo operations executed in the time frame. Our results are in Tables 6 and 7.

Under idle conditions, Amigo expectedly contributes the most to the Raspberry Pi's energy consumption. When idle, the Raspberry Pi's power management mechanisms ensure the CPU and other components are in low power states, resulting in higher energy spikes when woken. When already under load, we can see Amigo's contributions decrease.

Table 7: Additional energy consumed by a single Amigo operation alongside each task.

Task	Addition	Removal	Key Refresh
Idle	5.859 J	1.340 J	0.776 J
Video	3.211 J	0.754 J	0.344 J
Browsing	2.012 J	0.454 J	0.336 J

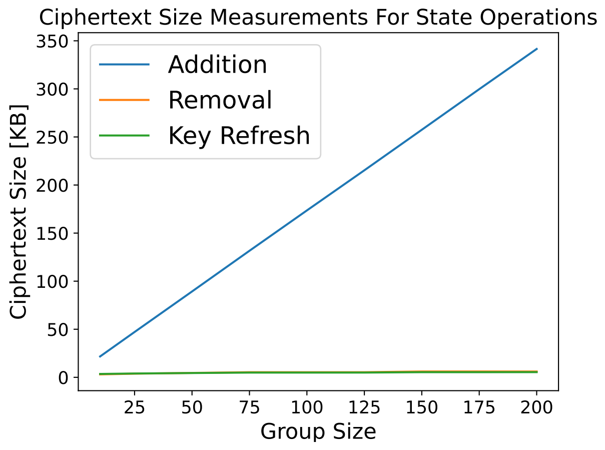

Figure 11: Ciphertext size measurements of Amigo state operations.

Amigo's most frequent state operations, removal and key refresh, should be the most energy efficient. As we can see in Table 7, the removal and key refresh operations in each task are at least 4 times more energy efficient than addition. While the device is already under load (during the video and browsing tasks), a single removal or key refresh requires less than 1 additional Joule of energy be consumed by the device. Our results are promising as protesters will be able to use Amigo alongside other applications without significant degradation in battery life.

Network load benchmarks: As groups increase in size, state operation message sizes also see an increase. Highlighted in Figure 11, member addition incurs the largest message sizes with 341.517 kilobytes of data needing to be propagated through the network for a group size of 200. As noted previously, however, member additions are not particularly frequent. Other state operations likely to occur more frequently, member removal and key refresh, incur significantly less message sizes which is beneficial to maximizing the performance of the mesh backbone.

E Latency

Figure 12 shows the latency for our mechanisms across our routing protocols. We note end-to-end message latency is mostly consistent over time across all mechanisms. We see that, as expected, flooding protocols tend to incur less end-to-end message latency, as they do not have to adhere to an epoch system, nor route messages through a leader; both of which limit message delivery. Dynamic clique routing looks 'spikey' because of the epoch structure; the fact that messages are not exchanged during the election period causes some messages to go unsent for longer periods of time.

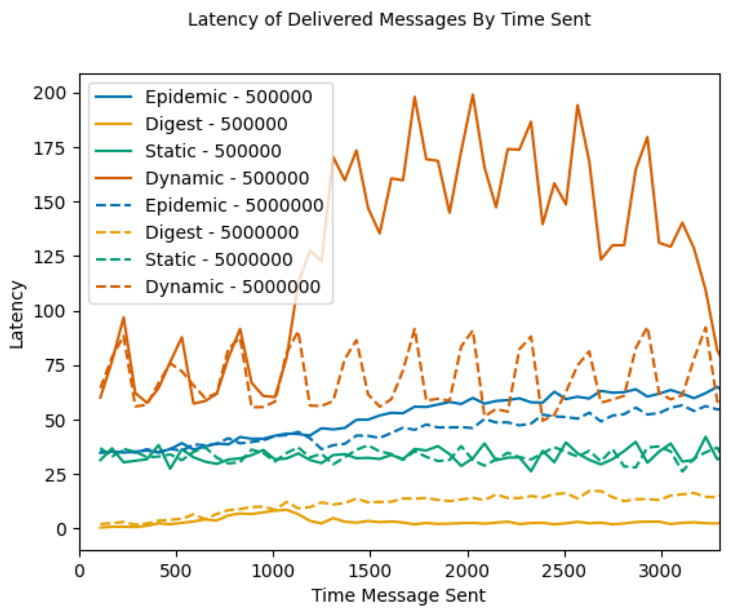

Figure 12: Latency over time for the gather mobility model.