Breaking Bridgefy, again

Adopting libsignal is not enough

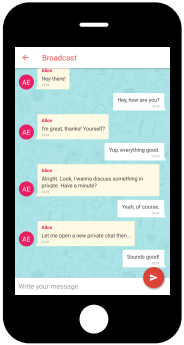
Martin R. Albrecht[1] **Raphael Eikenberg**[2] Kenneth G. Paterson[2]

[1]Information Security Group, Royal Holloway, University of London
[2]Applied Cryptography Group, ETH Zurich

31st USENIX Security Symposium

What is Bridgefy?

- Mobile messaging app that works without Internet connection
- Peers form a mesh network based on Bluetooth LE
- Downloaded more than 8 million times [13]
- Technology available as subscription-based SDK



(a) Broadcast chat with everyone

(b) Private chat with Alice

What makes Bridgefy interesting? [5]

Forbes

CONSUMER TECH

Ukrainians Prepping For Internet Loss By Getting Apps For Offline, Private, Mesh Communications

John Koetsier Senior Contributor
John Koetsier is a journalist, analyst, author, and speaker.

Follow

Feb 25, 2022, 01:01pm EST

Updated March 2 with a note on security in Bridgefy

The top apps being downloaded in Ukraine right now are Signal, the private messaging app, Bridgefy, which enables communications without the internet via mesh networking, Maps.me, an offline mapping app, and several "walkie-talkie" apps that enable free communication without sign-ups or personal information.

Top Apps | Top Publishers | Top SDK's | Trending Apps

1	Signal - Private Messe... Signal Messenger, LLC	^ 11
2	Bridgefy - Offline Mes... Bridgefy, Inc.	^ 36
3	MAPS.ME: Offline Ma... STOLMO LIMITED	^ 59
4	Zello Walkie Talkie Zello	^ 150

Research in 2020 revealed severe flaws in Bridgefy

Albrecht, Blasco, Jensen, and Mareková [1] broke the security of Bridgefy in early 2020. Findings included:

- Ability to track users (social graph)
- No authenticity (impersonation)
- No effective confidentiality protections (eavesdropping)
- Lack of resilience against adversarially crafted messages (DoS the entire network)

Mesh Messaging in Large-scale Protests: Breaking Bridgefy

Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková

Royal Holloway, University of London
{martin.albrecht,jorge.blascoalis,rikke.jensen,lenka.marekova}@rhul.ac.uk

Abstract. Mesh messaging applications allow users in relative proximity to communicate without the Internet. The most viable offering in this space, Bridgefy, has recently seen increased uptake in areas experiencing large-scale protests (Hong Kong, India, Iran, US, Zimbabwe, Belarus), suggesting its use in these protests. It is also being promoted as a communication tool for use in such situations by its developers and others. In this work, we report on a security analysis of Bridgefy. Our results show that Bridgefy, as analysed, permitted its users to be tracked, offered no

Bridgefy employed libsignal as a reaction to 'Breaking Bridgefy' [3]...

- No public independent security assessment was performed
- But developers continued to claim their app is secure:
  - *All messages will be end-to-end encrypted*
  - *A third person will no longer be able to impersonate any other user*
  - *Man-in-the-middle attacks done by modifying stored keys will no longer be possible*
  - *One-to-one messages sent over the mesh network will no longer contain the sender and receiver IDs in plain text*
  - *A third person will no longer be able to use the server's API to learn others' usernames*
  - *All payloads will be encrypted*
  - *Historical proximity tracking will not be possible*

– Bridgefy

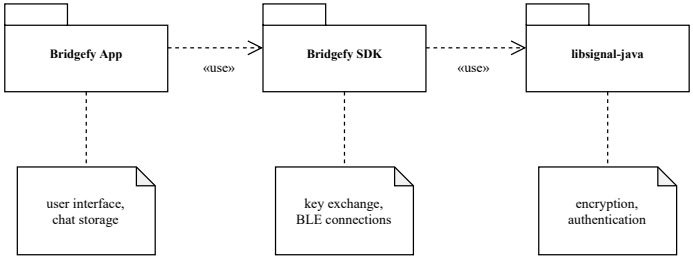
. . . but the security of Bridgefy has not improved

Findings of our research include:

- New attack on the confidentiality of private chats
- Plaintext recovery with unknown shared key
- Can still impersonate broadcast messages
- Can still build social graphs
- Can still DoS other peers, but not entire network at once

We reconstructed Bridgefy's architecture via reverse engineering

- Static analysis: CFR [2], Fernflower [8], Krakatau [7], and Procyon [12]
- Dynamic analysis: Frida [9] and objection [10]

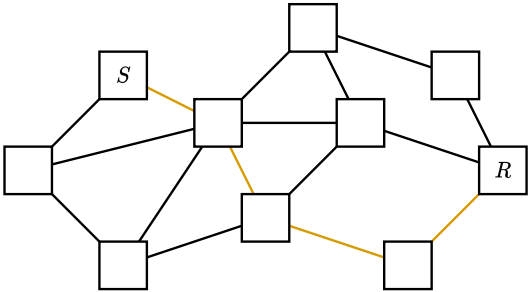


How user are identified

When the app starts for the first time

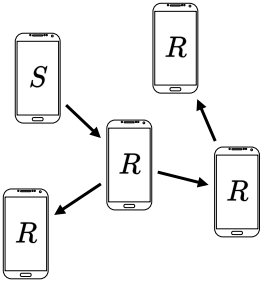
- the user chooses a non-unique display name, and
- the app generates a 128 bit random `userId` (UUID).

A message is delivered by multiple packets

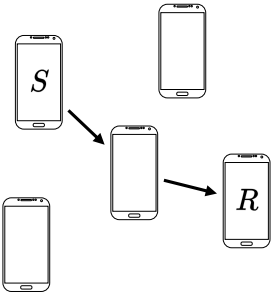


Single message, four packets (on this route),
text in message is called ‘payload content’

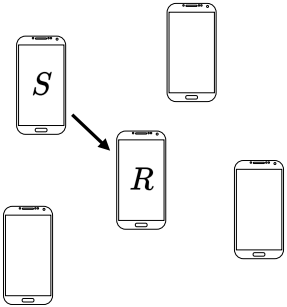
Three types of packets



(a) Broadcast packets



(b) Multi-hop packets



(c) One-to-one packet

Packet types and encoding

Type	*Used for*
`BleHandshake`	Handshakes
`BleEntityContent`	One-to-one packets
`ForwardTransaction`	Broadcast and multi-hop packets

Packets are

- serialised using MessagePack [6],
- compressed using gzip [4], and
- encrypted using libsignal [11] or AES-ECB.

Packet encryption overview

Data Category	Handshake	Broadcast	Multi-hop	One-to-one
Metadata	AES-ECB	AES-ECB	AES-ECB	AES-ECB
Payload	AES-ECB	AES-ECB	libsignal	libsignal

- Each app employing the Bridgefy SDK is assigned a dedicated shared key
- All devices using this app know its shared key
- Bridgefy messenger is publicly available
  $\Longrightarrow$ adversary knows its shared key

Simplified handshake

$$A \rightarrow B: \quad \texttt{ResponseTypeGeneral}(\mathrm{userId}_A) \tag{1}$$

$$B \rightarrow A: \quad \texttt{ResponseTypeGeneral}(\mathrm{userId}_B) \tag{2}$$

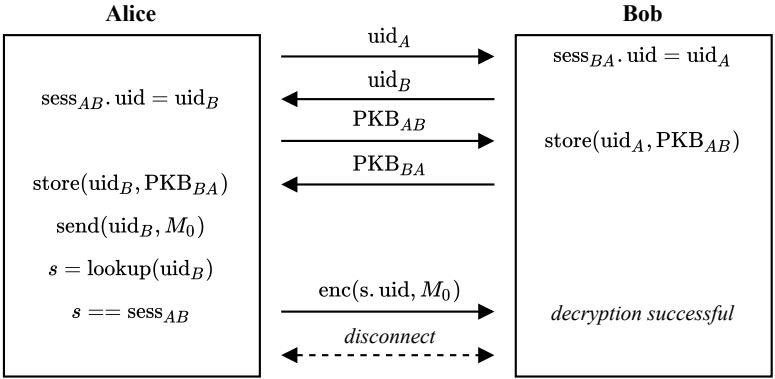
$$A \rightarrow B: \quad \texttt{ResponseTypeKey}(\mathrm{PKB}_A) \tag{3}$$

$$B \rightarrow A: \quad \texttt{ResponseTypeKey}(\mathrm{PKB}_B) \tag{4}$$

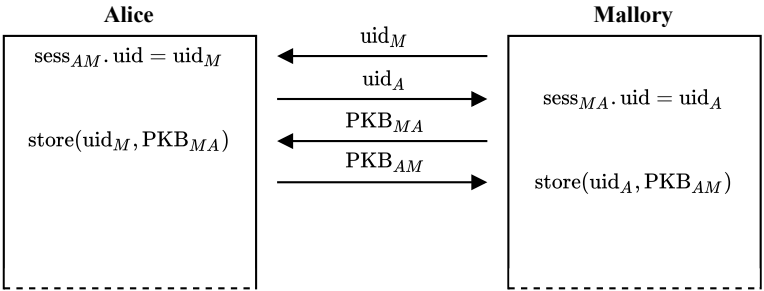
Attack #1: Breaking Confidentiality of Private Chats

- Bridgefy SDK suffers from a timing issue
- Assume knowledge of the shared key (valid for the Bridgefy messenger)
- Necessary steps:
  - Handshake between Alice and Bob
  - Handshake between Alice and Mallory
  - Race-condition to read messages from Alice intended for Bob

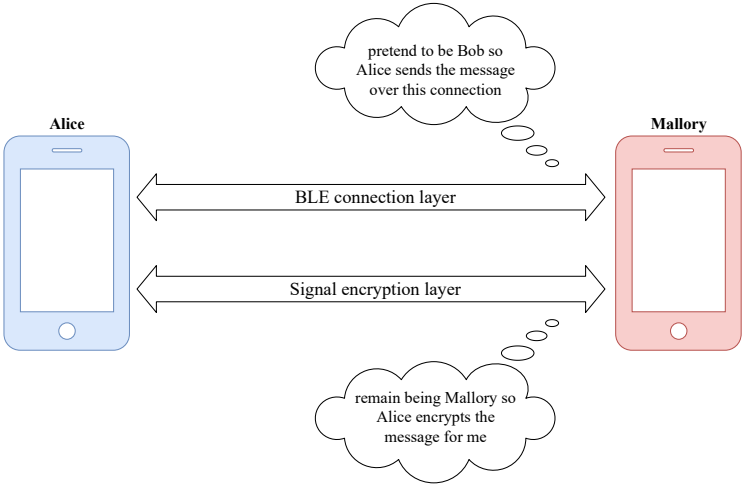
Attack #1: Handshake between Alice and Bob



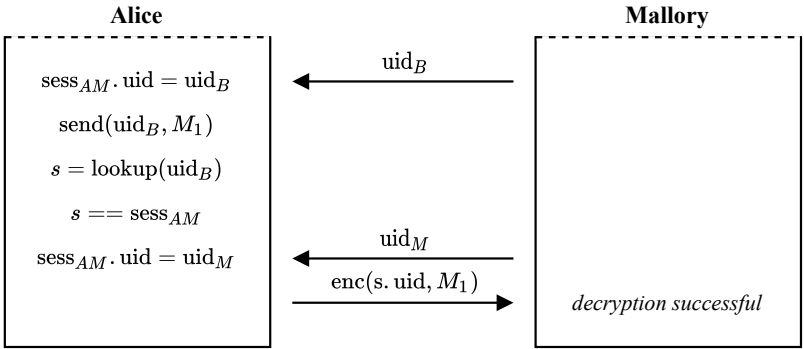
Attack #1: Handshake between Alice and Mallory



Attack #1: Reading messages from Alice intended for Bob



Attack #1: Reading messages from Alice intended for Bob



Attack #1: Demo

Switch to video playback...
(if time allows)

Attack #2: Broadcast Message Recovery

- Assume shared key is unknown (other app that uses the SDK)
- Encryption of broadcasts is deterministic
- Consider the payload contents ‘password’ and ‘11111111’
- Compression precedes encryption

$\Longrightarrow$ Can build something similar to a compression oracle attack

Attack #2: Assumptions

- Adversary $\mathcal{A}$, challenger $\mathcal{C}$, payload content π
- Assume the shared key is unknown to $\mathcal{A}$
- Set of possible payload contents $\mathscr{P} = \{p_1, p_2, \ldots, p_n\}$
- Broadcast messages contain $\pi \in \mathscr{P}$
- Message senders have usernames of equal length
- $\mathcal{A}$ can capture M packets at the first $H \leq 50$ hops $\implies M \cdot H$ packets

---

Challenge: varying timestamps and userIds cause gzip to output different lengths

Attack #2: Choices for $\mathscr{P}$

$$\mathscr{P}_{\mathrm{b}} = \{p_i | 32 \leq i \leq 126\}$$

where p_i is byte i

$$\mathscr{P}_{\mathrm{w}}^{\ell,n} = \{p_1, p_2, \ldots, p_n\}$$

where p_i is the ith most commonly used password of length ℓ in the rockyou password list

Attack #2: Phases

Simulation

$\mathcal{A}$ collects $|\mathscr{P}| \cdot N \cdot H$ simulated packets (performed offline) and derives distributions $\theta_{\ell|h,p}$: probability of observing length ℓ at hop h with payload content $\pi = p$.

Attack

- $\mathcal{A}$ collects $M \cdot H$ real packets (performed online) with $\pi \in \mathscr{P}$ and records the length of the ith broadcast at hop h as $l_{i,h}$.
- $\mathbf{l}$ abstractly denotes the collection of observed lengths.
- Maximum Likelihood Estimation (MLE) leveraged to derive a guess $\hat{\pi}$ based on $\theta_{\ell|h,p}$ and $\mathbf{l}$.

Attack #2: Clearly outperforms random guessing

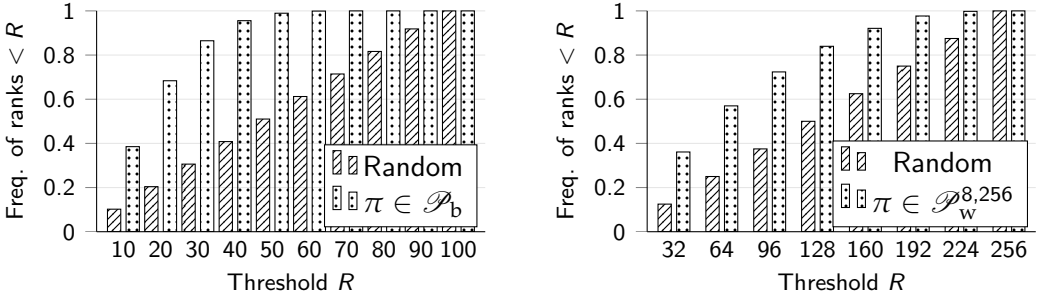
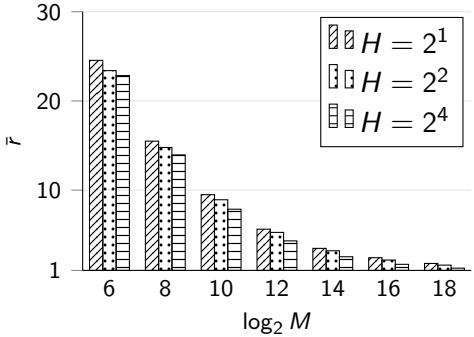
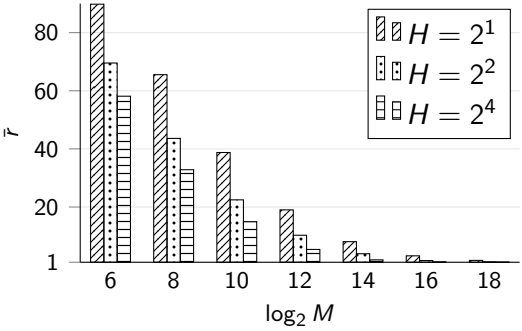


Figure: Cumulative frequency histogram of ranks, comparing our attack against random guessing for different $\mathscr{P}$. The plots show the portion of ranks less than R for $H = 2$, $N = 2^{20}$ and $M = 2^8$.

Attack #2: Improvements with increasing M and H



(a) $\pi \in \mathscr{P}_{\mathrm{b}}$



(b) $\pi \in \mathscr{P}_{\mathrm{w}}^{8,256}$

Figure: Average rank $\bar{r}$ for different (M, H).

Bridgefy's response on their blog

We were happy that our app was encrypted to the best of our capacity.*

– Bridgefy

And remember: try to avoid sharing sensitive information through any digital means, as no technical product is 100 % *safe!*

– Bridgefy

Conclusion

- The privacy issues of Bridgefy remain largely unresolved
- Its security track record does not meet the needs of its audience, higher-risk users
- The task of cryptographically securing an application is still non-trivial
- There is, seemingly, no good alternative to Bridgefy (Briar?)
  $\implies$ The problem of secure offline messaging remains unsolved in practice

Thank you!

Questions?

Contact: twitter.com/eikendev
Source: github.com/eikendev/breaking-bridgefy-again

References

[1] Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. Mesh messaging in large-scale protests: Breaking Bridgefy. In Kenneth G. Paterson, editor, *Topics in Cryptology - CT-RSA 2021*, volume 12704 of *Lecture Notes in Computer Science*, pages 375–398. Springer, 2021. doi: 10.1007/978-3-030-75539-3_16.

[2] Lee Benfield. `https://www.benf.org/other/cfr/`, no date.

[3] Bridgefy. Press release – major security updates at bridgefy! `https://bridgefy.me/press-release-major-security-updates-at-bridgefy/`, October 2020.

[4] L. Peter Deutsch. GZIP file format specification version 4.3. RFC 1952, May 1996. URL `https://rfc-editor.org/rfc/rfc1952.txt`.

[5] John Koetsier (Forbes). Ukrainians prepping for internet loss by getting apps for offline, private, mesh communications. `https://www.forbes.com/sites/johnkoetsier/2022/02/25/ukrainians-prepping-for-internet-loss-by-getting-apps-for-offline-private-mesh-communications/`, February 2022.

[6] Sadayuki Furuhashi. `https://msgpack.org/`, no date.

[7] Robert Grosse. `https://github.com/Storyyeller/Krakatau`, no date.

[8] JetBrains. `https://github.com/JetBrains/intellij-community/tree/master/plugins/java-decompiler/engine`, no date.

[9] Ole André V. Ravnås. `https://frida.re/`, no date.

[10] SensePost. `https://github.com/sensepost/objection`, no date.

[11] Signal. `https://github.com/signalapp/libsignal-protocol-java`, no date.

[12] Mike Strobel. `https://github.com/mstrobel/procyon`, no date.

[13] Twitter – Bridgefy. `https://twitter.com/bridgefy/status/1537484715694882817`, June 2022.