

Mesh Messaging in Large-scale Protests: Breaking Bridgefy (Abridged* version)

Martin R. Albrecht

Royal Holloway, University of London
martin.albrecht@rhul.ac.uk

Rikke Bjerg Jensen

Royal Holloway, University of London
rikke.jensen@rhul.ac.uk

Jorge Blasco

Royal Holloway, University of London
jorge.blascoalis@rhul.ac.uk

Lenka Marekova

Royal Holloway, University of London
lenka.marekova.2018@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, Thailand), 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 perform a security analysis of Bridgefy. Our results show that Bridgefy permits its users to be tracked, offers no authenticity, no effective confidentiality protections and lacks resilience against adversarially crafted messages. We verify these vulnerabilities by demonstrating a series of practical attacks on Bridgefy. Thus, if protesters rely on Bridgefy, an adversary can produce social graphs about them, read their messages, impersonate anyone to anyone and shut down the entire network with a single maliciously crafted message. As a result, we conclude that participants of protests should avoid relying on Bridgefy until these vulnerabilities are addressed and highlight the resulting gap in the design space for secure messaging applications.

1 Introduction

Mesh messaging applications rely on wireless technologies such as Bluetooth Low Energy (BLE) to create communication networks that do not require Internet connectivity. These can be useful in scenarios where the cellular network may simply be overloaded, e.g. during mass gatherings, or when governments impose restrictions on Internet usage, up to a full blackout, to suppress civil unrest. While the functionality requirements of such networks may be the same in both of these scenarios – delivering messages from A to B – the security requirements for their users change dramatically.

In September 2019, Forbes reported “Hong Kong Protestors Using Mesh Messaging App China Can’t Block: Usage Up

3685%” [55] in reference to an increase in downloads of a mesh messaging application, Bridgefy [30], in Hong Kong. Bridgefy is both an application and a platform for developers to create their own mesh network applications.¹ It uses BLE or Bluetooth Classic and is designed for use cases such as “music festivals, sports stadiums, rural communities, natural disasters, traveling abroad”, as given by its Google Play store description [31]. Other use cases mentioned on its webpage are ad distribution (including “before/during/after natural disasters” to “capitalize on those markets before anybody else” [30]) and turn-based games. The Bridgefy application has crossed 1.7 million downloads as of August 2020 [67].

Though it is advertised as “safe” [31] and “private” [20] and its creators claimed it was secured by end-to-end encryption [55, 56, 63], none of the aforementioned use cases can be considered as taking place in adversarial environments such as situations of civil unrest where attempts to subvert the application’s security are not merely possible, but to be expected, and where such attacks can have harsh consequences for its users. Despite this, the Bridgefy developers advertise the application for such scenarios [21, 23, 24, 55] and media reports suggest the application is indeed relied upon.

Hong Kong International news reports of Bridgefy being used in anti-extradition law protests in Hong Kong began around September 2019 [19, 55, 70, 79], reporting a spike in downloads that was attributed to slow mobile Internet speeds caused by mass gatherings of protesters [32]. Around the same time, Bridgefy’s CEO reported more than 60,000 app installations in a period of seven days, mostly from Hong Kong [70]. However, a Hong Kong based report available in English [17] gave a mixed evaluation of these claims: in the midst of a demonstration, not many protesters appeared to be using Bridgefy. The same report also attributes the spike in Bridgefy downloads to a DDoS attack against other popular communication means used in these protests: Telegram and the Reddit-like forum LIHKG.

*The full version of this work including a description of Bridgefy’s message processing will be made available later.

¹As we discuss in Section 2.5, alternatives to Bridgefy are scarce, making it the predominant example of such an application/framework.

India The next reports to appear centred on the Citizenship Amendment Act protests in India [14] that occurred in December 2019. Here the rise in downloads was attributed to an Internet shutdown occurring during the same period [58, 74]. It appears that the media narrative about Bridgefy’s use in Hong Kong might have had an effect: “So, Mascarenhas and 15 organisers of the street protest decided to take a leaf out of the Hong Kong protesters’ book and downloaded the Bridgefy app” [64]. The Bridgefy developers reported continued adoption in summer 2020 [25].

Iran While press reports from Iran remain scarce, there is evidence to suggest that some people are trying to use Bridgefy during Internet shutdowns and restrictions: the rise of customer support queries coming from Iran and a claim by the Bridgefy CEO that it is being distributed via USB devices [59].

Lebanon Bridgefy now appears among recommended applications to use during an Internet shutdown, e.g. in the list compiled by a Lebanese NGO during the October 2019 Lebanon protests [73]. A media report suggests adoption [56].

US The Bridgefy developers report uptake of Bridgefy during the Black Lives Matter protests across the United States of America [24, 26]. It is promoted for use in these protests by the developers and others on social media [3, 24, 76].

Zimbabwe Media and social media reports advertised Bridgefy as a tool to counter a government-mandated Internet shutdown [57, 60] in summer 2020. The Bridgefy developers reported an uptick in adoption [27].

Belarus Social media posts and the Bridgefy developers suggest adoption in light of a government-mandated Internet shutdown [28].

Thailand Social media posts encouraged student protesters to install the Bridgefy application during August 2020 [11].

Contributions In Section 3, we report several vulnerabilities voiding both the security claims made by the Bridgefy developers and the security goals arising from its use in large-scale protests. In particular, we describe various avenues for tracking users of the Bridgefy application and for building social graphs of their interactions both in real time and after the fact. We then use the fact that Bridgefy implements no effective authentication mechanism between users (nor a state machine) to impersonate arbitrary users. This attack is easily extended to an attacker-in-the-middle (MITM) attack for subverting public-key encryption. We also present variants of Bleichenbacher’s attack [15] which break confidentiality using $\approx 2^{17}$ chosen ciphertexts. Our variants exploit the composition of PKCS#1 v1.5 encryption and Gzip compression in Bridgefy. Moreover, we utilise compression to undermine the advertised resilience of Bridgefy: using a single message “zip

bomb” we can completely disable the mesh network, since clients will forward any payload before parsing it which then causes them to hang until a reinstallation of the application. Overall, we conclude that using Bridgefy represents a significant risk to participants of protests. We discuss our findings, including possible mitigation strategies, and report on the disclosure process in Section 4.

2 Preliminaries

We denote concatenation of strings or bytes by `||`. Strings of byte values are written in hexadecimal and prefixed with `0x`, in big-endian order. We use e.g. “10B” to denote 10 bytes.

2.1 Bridgefy features

The key feature of Bridgefy is that it exchanges data using Bluetooth² when an Internet connection is not available. The application can send the following kinds of messages:

- one-to-one messages between two parties
 - sent over the Internet if both parties are online,
 - sent directly via Bluetooth if the parties are in physical range, or
 - sent over the Bluetooth mesh network, and
- Bluetooth broadcast messages that anyone can read in a special “Broadcast mode” room.

Note that the Bluetooth messages are handled separately from the ones exchanged over the Internet using the Bridgefy server, i.e. there is no support for communication between one user who is on the Internet and one who is on the mesh network.

Before parties who are in Bluetooth range can communicate, they perform a handshake. If the parties have never communicated before, the handshake includes the exchange of public keys.

2.2 Reverse engineering

We analysed the Bridgefy apk version 2.1.28 dated January 2020 and available in the Google Play store. It includes the Bridgefy SDK version 1.0.6. Since the source code was not available, we decompiled the apk to (obfuscated) Java classes using Jadx [72]. The initial deobfuscation was done automatically by Jadx, with the remaining classes and methods being done by hand using artefacts left in the code and by inspecting the application’s execution.

This inspection was performed using Frida, a dynamic instrumentation toolkit [4], which allows for scripts to be

²Bridgefy supports connections over both BLE and Bluetooth Classic, but the latter is a legacy option for devices without BLE support, so in our analysis we focused on BLE.

injected into running processes, essentially treating them as black boxes but enabling a variety of operations on them. In the context of Android applications written in Java, these include tracing class instances and hooking specific functions to monitor their inputs/outputs or to modify their behaviour during runtime.

2.3 Primitives used

Message encoding

To encapsulate Bluetooth messages and their metadata, Bridgefy uses MessagePack [38], a binary serialisation format that is more compact than and advertised as an alternative to JSON.

It is then compressed using Gzip [49], which utilises the widely-used DEFLATE compressed data format [48]. The standard implementation found in the `java.util.zip` library is used in the application. A Gzip file begins with a 10-byte header, which consists of a fixed prefix `0x1f8b08` followed by a flags byte and six additional bytes which are usually set to 0. Depending on which flags are set, optional fields such as a comment field are placed between the header and the actual DEFLATE payload. A trailer at the end of the Gzip file consists of two 4-byte fields: a CRC32 and the length, both over the uncompressed data. An actual Gzip file can consist of several such concatenated “files”, each with their own header and trailer, with no special separators between them.

RSA PKCS#1 v1.5

Bridgefy makes use of the (now deprecated) PKCS#1 v1.5 [50] standard. This standard defines a method of using RSA encryption, in particular specifying how the plaintext should be padded before being encrypted. The format of the padded data that will be encrypted is `0x0002 || <random non-zero bytes> || 0x00 || <message>`. If the size of the RSA modulus and hence the size of the encryption block is k bytes, then the maximum length of the message is $k - 11$ bytes to allow for at least 8 bytes of padding.

This padding format enables a well-known attack by Bleichenbacher [15] (for variants/improvements of Bleichenbacher’s attack see e.g. [10, 12, 16, 54]). The attack requires a padding oracle, i.e. the ability to obtain an answer to whether a given ciphertext decrypts to something that conforms to the padding format. Sending some number of ciphertexts, each chosen based on previous oracle responses, leads to full plaintext recovery. For RSA with $k = 128$, the number of chosen ciphertexts required has been shown to be between 2^{12} and 2^{16} [18].

In more detail, let c be the target ciphertext, n the public modulus, and (e, d) the public and private exponents, respectively. We have $\text{pad}(m) = c^d \pmod n$ for the target message m . The chosen ciphertexts will be of the form $c^* = s^e \cdot c \pmod n$

for some s . If c^* has correct padding, we know the first two bytes of $s \cdot \text{pad}(m)$, and hence a range for its possible values. The attack thus first finds small values of s which result in a positive answer from the oracle and for each of them computes a set of ranges for the value of $\text{pad}(m)$. Once there is only one possible range, larger values of s are tried in order to narrow this range down to only one value, which is the original message.

2.4 Related work

Secure messaging

Message layer security has received renewed attention in the last decade, with an effort to standardise a protocol – simply dubbed Message Layer Security (MLS) – now underway by the IETF [77], with several academic works proposing solutions or analysing security [7, 8, 33]. The use of secure messaging by “high-risk users” is considered in [36, 43]. In particular, those works analyse interviews with human rights activists and secure messaging application developers to establish common and diverging concerns.

Compression in security

The first compression side channels in the context of encryption were described by [53], based on the observation that the compression rate can reveal information about the plaintext. When combined with the ability to prepend or append known/chosen text to the original message, the ratio can reveal whether a given text is part of the message, and hence enable partial message recovery. Since then, there have been practical attacks exploiting the compression rate “leakage” in TLS and HTTP, dubbed CRIME [35] and BREACH [42] which enabled the recovery of HTTP cookies and contents, respectively. Similarly, [41] uses Gzip as a format oracle in a CCA attack. Beyond cryptography, compression has also been utilised for denial of service attacks in the form of so-called “zip bombs” [37] where an attacker prepares a compressed payload that decompresses to a massive message.

Mesh networking security

Wireless mesh networks have a long history, but until recently they have been developed mainly in the context of improving or expanding Wi-Fi connectivity via various ad hoc routing protocols, where the mesh usually does not include client devices. Flood-based networks using Bluetooth started gaining traction with the introduction of BLE, which optimises for low power and low cost, and which has been part of the core specification [68] since Bluetooth 4.0. BLE hardware is integrated in all current major smartphone brands, and the specification has native support in all common operating systems.

Previous work on the security analysis of Bluetooth focused on finding vulnerabilities in the pairing process or showing the

inadequacy of its security modes, some of which have been fixed in later versions of the specification (see [34, 44] for surveys of attacks focusing on the classic version of Bluetooth). As a more recent addition, BLE has not received as much comprehensive analysis, but general as well as IoT-focused exploits exist [51, 66, 71, 78]. Research on BLE-based tracking has looked into the usage of unique identifiers by applications and IoT devices [13, 80]. The literature on security in the context of BLE-based mesh networks is scarce, though the Bluetooth Mesh Profile [69] developed by the Bluetooth SIG is now beginning to be studied [5, 6].

2.5 Alternative mesh applications

We list various alternative chat applications that target scenarios where Internet connectivity is lacking, in particular paying attention to their potential use in a protest setting.

FireChat

FireChat [39] was a mobile application for secure wireless mesh networking meant for communication without an Internet connection. Though it was not built for protests, it became the tool of choice in various demonstrations since 2014, e.g. in Iraq, Hong Kong and Taiwan [9, 52, 61], and since then was also promoted as such by the creators of the application. However, it had not received any updates in 2019 and as of April 2020, it is no longer available on the Google Play store and its webpage has been removed, so it appears that its development has been discontinued.

BLE Mesh Networking

Bluetooth itself provides a specification for building mesh networks based on Bluetooth Low Energy that is referred to as the Bluetooth Mesh Profile [69]. While it defines a robust model for implementing a flood-based network for up to 32,000 participating nodes, its focus is not on messaging but rather connectivity of low-power IoT devices within smart homes or smart cities. As a result, it is more suitable for networks that are managed centrally and whose topology is stable over time, which is the opposite of the unpredictable and always-changing flow of a crowd during a mass protest. Further, it makes heavy use of the advertising bearer (a feature not widely available in smartphones), which imposes constraints on the bandwidth of the network – messages can have maximum size of 384 bytes, and nodes are advised to not transmit more than 100 messages in any 10 second window. The profile makes use of cryptography for securing the network from outside observers as well as from outside interference, but it does expect participating nodes to be benign, which cannot be assumed in the messaging setting. From within the network, a malicious node can not only observe but also impersonate other nodes and deny them service.

HypeLabs

The Hype SDK offered by HypeLabs [47] sets out a similar goal as Bridgefy, which is to offer secure mesh networks for a variety of purposes when there is no Internet connection. Besides Bluetooth, it also utilises Wi-Fi, and supports a variety of platforms. Among its usecases, the Hype SDK whitepaper [46] lists connectivity between IoT devices, social networking and messaging, distributed storage as well as connectivity during catastrophe scenarios and emergency broadcasting. While an example chat application is available on Google Play (with only 100+ downloads), HypeLabs do not offer the end-user solutions for those use cases themselves, merely offering the SDK as a paid product for developers. There is no information available on what applications are using the SDK, if any.

Briar

Briar [65] describes itself as “secure messaging, anywhere” [65] and is referenced in online discussions on the use of mesh networking applications in protests [62]. However, Briar does not realise a mesh network. Instead it opens point-to-point sockets over a Bluetooth Classic (as opposed to Low Energy) channel to nearby nodes. Its reach is thus limited to one hop unless users manually forward messages.

Serval

Serval Mesh [40] is an Android application implementing a mesh network using Wi-Fi that sets its goal as enabling communication in places which lack infrastructure. Originally developed for natural disaster situations, the project includes special hardware Mesh Extenders that are supposed to enhance coverage. While the application is available for download, it cannot be accessed from Google Play because it targets an old version of Android to allow it to run on older devices such as the ones primarily used in rural communities. Work on the project is still ongoing, as it is not ready for deployment at scale. Hence its utility in large-scale protests where access to technology itself is not a barrier is currently limited.

Subnodes

The use of additional hardware devices enables a different approach to maintaining connectivity, which is taken by the open source project Subnodes [1]. It allows local area wireless networks to be set up on a Raspberry Pi, which then acts as a web server that can provide e.g. a chat room. Multiple devices can be connected in a mesh using the BATMAN routing protocol [2], which is meant for dynamic and unreliable networks. However, setting up and operating such a network requires technical knowledge. In the setting of a protest, even

carrying the hardware device for one of the network’s access points could put the operator at risk.

3 Attacks

In this section, we show that Bridgefy does not provide confidentiality of messages and also that it does not satisfy the additional security needs arising in a protest setting: privacy, authenticity and reliability in adversarial settings.

3.1 Privacy

Here, we discuss vulnerabilities in Bridgefy pertaining to user privacy in contrast to confidentiality of messages. We note that Bridgefy initially made no claim about anonymity in its marketing but disabled mandating phone number verification to address anonymity needs in 2019 [22].

Local user tracking

To prevent tracking, Bluetooth-enabled devices may use “random” addresses which change over time (for details on the addressing scheme see [68, Section 10.8]). However, when a Bridgefy client sends BLE `ADV_IND` packets (something that is done continuously while the application is running), it also transmits an identifier computed from the user ID. The user ID does not change unless the user reinstalls the application, so passive observation of the network is enough to enable tracking all users.

In addition, the automatic handshake protocol composed with public-key caching provides a mechanism to perform historical contact tracing. If the devices of two users have been in range before, they will not request each other’s public keys, but they will do so automatically if that has not been the case.

Participant discovery

Until December 2019 [22], Bridgefy required users to register with a phone number. Users still have the option to do so, but it is no longer the default. If the user gives the permission to the application to access the contacts stored on their phone, the application will check which of those contacts are already Bridgefy users based on phone numbers and display those contacts to the user. When Bridgefy is predominantly installed on phones of protesters, this allows the identification of participants by running contact discovery against all local phone numbers. While an adversary with significant control over the network, such as a state actor, might have alternative means to collect such information, this approach is also available to e.g. employers or activists supporting the other side.

Social graph

All one-to-one messages sent over the mesh network contain the sender and receiver IDs in plaintext, so a passive adversary with physical presence can build a social graph of what IDs are communicating with whom. The adversary can further use the server’s API to learn the usernames corresponding to those IDs. In addition, since three receipts are sent when a message is received – “mesh reach” in clear, encrypted “delivery” receipt, encrypted “viewed” receipt – a passive attacker can also build an approximate, dynamic topology of the network, since users that are further away from each other will have a larger delay between a message and its receipts.

3.2 Authenticity

Bridgefy does not utilise cryptographic authentication mechanisms. As a result, an adversary can impersonate any user.

The initial handshake through which parties exchange their public keys or identify each other after coming in range relies on two pieces of information to establish the identities: a user ID and the lower-level Bluetooth device address. Neither of these is an actual authentication mechanism: the user ID is public information which can be learned from observing the network, while [66] shows that it is possible to broadcast with any BLE device address.

However, an attacker does not need to go to such lengths. Spoofing can be done by sending a handshake message which triggers the other side to overwrite the information it currently has associated with a given user. Suppose there are two users who have communicated with each other before, Ursula and Ivan, and the attacker wishes to impersonate Ivan to Ursula. When the attacker comes into range of Ursula, she will initiate the handshake. If the attacker responds with information gathered from a previous handshake with Ivan, Ursula’s application will merge the attacker and Ivan into one user, and therefore show messages from the attacker as if they came from Ivan. If the real Ivan comes in range at the time the attacker is connected to Ursula, he will be able to communicate with her and receive responses from her that the attacker will not be able to decrypt. However, he will not be able to see the attacker’s presence. We implemented this attack and verified that it works, see Section 3.3.

The messages exchanged over the mesh network (when users are not in direct range) merely contain the user ID of the sender, so they can be spoofed with ease.

We also note that although the handshake protocol is meant for parties in range, a part of the handshake can also be sent over the mesh network. This means that users can be convinced to change the usernames and phone numbers associated with their Bridgefy contacts via the mesh network.

3.3 Confidentiality

Confidentiality of message contents is both a security goal mentioned in Bridgefy’s marketing material and relied upon by participants in protests. In this section we show that the implemented protections are not sufficient to satisfy this goal.

IND-CPA

Bridgefy’s encryption scheme only offers a security level of 2^{64} in a standard IND-CPA security game, i.e. a passive adversary can decide whether a message m_0 or m_1 of its choosing was encrypted as c . The adversary picks messages of length 245B and tries all 255^8 possible values for PKCS#1 v1.5 padding until it finds a match for the challenge ciphertext c .

Plaintext file sharing

Bridgefy allows its users to send direct messages composed of either just text or containing a location they want to share. The latter is processed as a special text message containing coordinates, and so these two types are encrypted, but the same is not true about any additional data such as image files. While the application itself does not currently offer the functionality to share images or other files, it is part of the SDK and receiving media files does work in the application. The fact that files are transmitted in plaintext is not stated in the documentation, so for developers using the SDK it would be easy to assume that files are shared privately when using this functionality.

MITM

This attack is an extension of the impersonation attack described in Section 3.2 where we convince the client to change the public key for any user ID it has already communicated with. If Ivan is out of range, the attacker can initiate a handshake with Ursula which contains contradictory information, combining the attacker’s identity with information from Ivan. The logic of the handshake processing in Ursula’s client dictates that it has to make a request for an updated public key. Then the attacker only needs to supply its own key, which will get associated with Ivan’s user ID. Afterwards, whenever Ursula sends a Bluetooth message to Ivan, it will be encrypted under the attacker’s key. Further, Ursula’s client will display messages from the attacker as if they came from Ivan, so this attack also provides impersonation. If at this stage Ivan comes back in range, he will not be able to connect to Ursula. The attack is not persistent, though – if the attacker goes out of range, Ivan (when in range) can run a legitimate handshake and restore communication.

We verified this and the previous impersonation attack in a setup with four Android devices, where the attacker had two devices running Frida scripts that modified the relevant handshake messages. Two attacker devices were used to instantiate

a full attacker in the middle attack, which is an artefact of us hotpatching the Bridgefy application using Frida scripts: one device to communicate with Ursula on behalf of Ivan and another with Ivan on behalf of Ursula.

We also note that since the Bridgefy server serves as a trusted database of users’ public keys, if compromised, it would be trivial to mount an attacker in the middle attack on the communication of any two users. This would also impact users who are planning to only use the application offline since the server would only need to supply them the wrong keys during registration.

Padding oracle attack

The following chosen ciphertext attack is enabled by the fact that all one-to-one messages use public-key encryption but no authentication, so we can construct valid ciphertexts as if coming from any sender. We can also track BLE packets and replay them at will, reordering or substituting ciphertext blocks.

We instantiate a variant of Bleichenbacher’s attack [15] on RSA with PKCS#1 v1.5 padding using Bridgefy’s delivery receipts. This attack relies on distinguishing whether a ciphertext was processed successfully or not. The receiver of a message sends a message status update when a message has been received and processed, i.e. decrypted, decompressed and parsed. If there was an error on the receiver’s side, no message is sent. No other indication of successful delivery or (type of) error is sent. Since the sender of a Bridgefy message cannot distinguish between decryption errors or decompression errors merely from the information it gets from the receiver, we construct a padding oracle that circumvents this issue.

Suppose that Alice sends a ciphertext c encrypting the message m to Bob that we intercept. In the classical Bleichenbacher’s attack, we would form a new ciphertext $c^* = s^e \cdot c \pmod n$ for some s where n is the modulus and e is the exponent of Bob’s public key. Now suppose that c^* has a correct padding. Since messages are processed in blocks, we can prepend and append valid ciphertexts. These are guaranteed to pass the padding checks as they are honestly generated ciphertexts (we recall that there is no authentication). We will construct these blocks in such a way that decompression of the joint contents will succeed with non-negligible probability, and therefore enable us to get a delivery receipt which will instantiate our padding oracle.

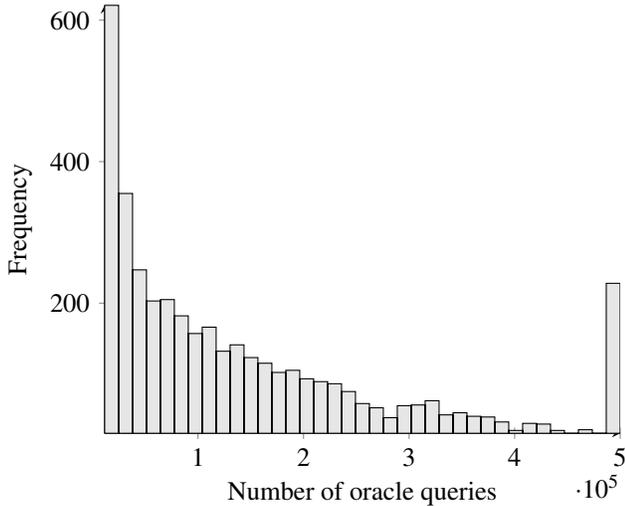
The Gzip file format [49] specifies a number of optional flags. If the flag `FLG.FCOMMENT` is set, the header is followed by a number of “comment” bytes, terminated with a zero byte, that are essentially ignored. In particular, these bytes are not covered by the CRC32 checksum contained in the Gzip trailer. Thus, we let c_0 be the encryption of a 10-byte Gzip header with this flag set followed by up to 245 non-zero bytes, and let c_1 be the encryption of a zero byte followed by a valid compressed MessagePack payload (i.e. of a message from the

attacker to Bob) and Gzip trailer.

When put together, $c_0 || c^* || c_1$ encrypts a correctly compressed message as long as $\text{unpad}(s \cdot \text{pad}(m))$ (which is part of the comment field) does not contain a zero byte, and therefore Bob will send a delivery receipt for the attacker’s message. The probability that the comment does not contain a zero byte for random s is $\geq (1 - \frac{1}{256})^{245} \approx 0.383$.

To study the number of adaptively chosen ciphertexts required, we adapted the simulation code from [18] for the Bleichenbacher-style oracle encountered in this attack: a payload will pass the test if it has valid padding for messages of any valid length (“FFT” in [12] parlance) and if it does not contain a zero byte in the “message” part after splitting off the padding. We then ran a Bleichenbacher-style attack 4,096 times (on 80 cores, taking about 12h in total) and recorded how often the oracle was called in each attack. We give a histogram of our data in Figure 1. The median is $2^{16.75}$, the mean $2^{17.36}$. Our SageMath [75] script, based on Python code in [18], and the raw data for Figure 1 are attached to the electronic version of this document.

Figure 1: Density distribution for number of ciphertexts required to mount a padding-oracle attack via Gzip comments.



We have verified the applicability of this attack in Bridgefy using ciphertexts c^* constructed to be PKCS#1 v1.5-conforming (i.e. where we set $s = \text{pad}(m)^{-1} \cdot \text{pad}(r) \bmod n$ where r is 245 random bytes). We used Frida to run a script on the attacker’s device that would send $c_0 || c^* || c_1$ to the target Bridgefy user via Bluetooth, and record whether it gets a delivery receipt for the message contained in c_1 or not. The observed frequency of the receipts matched the probability given earlier. This oracle suffices to instantiate Bleichenbacher’s original attack. In our preliminary experiments we were able to send a ciphertext every 450ms, suggesting 50% of attacks complete in less than 14 hours. We note, however, that our timings are based on us hotpatching Bridgefy to send the

messages and that a higher throughput might be achievable.

Padding oracles from a timing side-channel

Our decompression oracle depends on the Bridgefy SDK processing three blocks as a joint ciphertext. While we verified that this behaviour is also exhibited by the Bridgefy application, the application itself never sends ciphertexts that span more than two blocks as it imposes a limit of 256B on the size of the text of each message. Thus, a stopgap mitigation of our previous attack could be to disable the processing of more than two blocks of ciphertext jointly together.

We sketch an alternative attack that only requires two blocks of ciphertext per message. It is enabled by the fact that when a receiver processes an incorrect message, there is a difference in the time it takes to process it depending on what kind of error was encountered. This difference is clearly observable for ciphertexts that consist of at least two blocks, where the error occurs in the first block. We note that padding errors occurring in the second block can be observed by swapping the blocks, as they are decrypted individually.

Figure 2 and Table 1 (the raw data is attached to the electronic version of this document) show the differences for experiments run on the target device, measured using Frida. A script was injected into the Bridgefy application that would call the method responsible for extracting a message from a received BLE packet (including decryption and decompression) on given valid or invalid data. The execution time of this method was measured directly on the device using Java.

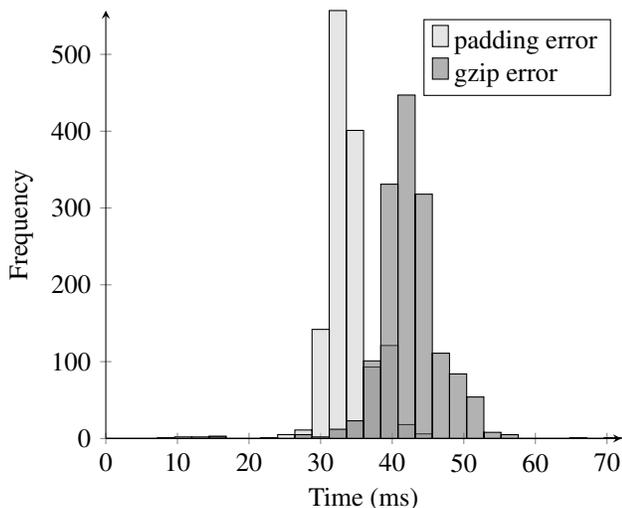
If multiple messages are received, they are processed sequentially, which enables the propagation of these timing differences to the network level. That is, the attacker sends two messages, one consisting of $c^* || c'$ where $c^* = s^e \cdot c \bmod n$ is the modified target ciphertext and c' is an arbitrary ciphertext block, and one consisting of some unrelated message, either as direct messages one after another or a mesh transaction containing both messages among its packets. The side-channel being considered is then simply the time it takes to receive the delivery receipt on the second valid message.

We leave exploring whether this could be instantiated in practice to future work, since our previous attacks do not require this timing channel. We note, though, that an adversary would likely need more precise control over the timing of when packets are released than that offered by stock Android devices in order to capture the correct difference in a BLE environment.

Table 1: Time of processing two ciphertext blocks in milliseconds, where N is the number of samples in each experiment.

error type	N	μ	σ	σ/\sqrt{N}
bad padding	1360	33.882956	3.137260	0.085071
gzip error	1508	42.557275	4.273194	0.110040

Figure 2: Processing of incorrect two-block ciphertexts.



3.4 Denial of service

Bridgefy’s appeal to protesters to enable messaging in light of an Internet shutdown makes resilience to denial of service attacks a key concern. While a flood-based network can be resilient as a consequence of its simplicity, some particularities of the Bridgefy setup make it vulnerable.

Broad DoS

Due to the use of compression, Bridgefy is vulnerable to “zip bomb” attacks. In particular, compressing a message of size 10MB containing a repeated single character results in a payload of size 10KB, which can be easily transmitted over the BLE mesh network. Then, when the client attempts to display this message, the application becomes unresponsive to the point of requiring reinstallation to make it usable again. Sending such a message to the broadcast chat provides a trivial way of disabling many clients at the same time, since clients will first forward the message further and only then start the processing to display it which causes them to hang. As a consequence, a single adversarially generated message can take down the entire network. We implemented this attack and tested it in practice on a number of Android devices.

Targeted DoS

A consequence of the attack from Section 3.3 is that it provides a way to prevent particular two users from connecting, even if they are in Bluetooth range.

4 Discussion

While our attacks reveal severe deficiencies in the security of both the Bridgefy application and the SDK, it is natural to ask

whether they are valid and what lessons can be drawn from them for information security research.

Given that our attacks are variants of attacks known in the literature, it is worth asking why Bridgefy did not mitigate against them. A simple answer to this question might be that the application was not designed for adversarial settings and that therefore our attacks are out of scope, externally imposing security goals. However, such an account would fail to note that Bridgefy’s security falls short also in settings where attacks are not expected to be the norm, i.e. Bridgefy does not satisfy standard privacy guarantees expected of any modern messaging application. In particular, prior to our work, the Bridgefy developers advertised the app/SDK as “private” and as featuring end-to-end encryption; our attacks thus broke Bridgefy’s own security claims.

More importantly, however, Bridgefy *is* used in highly adversarial settings where its security ought to stand up to powerful nation-state adversaries and the Bridgefy developers advertised their application for these contexts [21, 23, 24, 55]. Technologies need to be evaluated under the conditions they are used in. Here, our attacks highlight the value of secure by design approaches to development. While designers might envision certain use cases, users, in the absence of alternatives, may reach for whatever solution is available.

Our work thus draws attention to this problem space. While it is difficult to assess the actual reliance of protesters on mesh communication, the *idea* of resilient communication in the face of a government-mandated Internet shutdown is present throughout protests across the globe [9, 11, 14, 23, 23, 24, 26, 28, 52, 55, 57, 61]. Yet, these users are not well served by the existing solutions they rely on. Thus, it is a pressing topic for future work to design communication protocols and tools that cater to these needs. We note, though, that this requires understanding “these needs” to avoid a disconnect between what designers design for and what users in these settings require [36, 43].

4.1 Responsible disclosure

We disclosed the vulnerabilities described in this work to the Bridgefy developers on 27 April 2020 and they acknowledged receipt on the same day. We agreed on a public disclosure date of 20 August 2020. Starting from 1 June 2020, the Bridgefy team began informing their users that they should not expect confidentiality guarantees from the current version of the application [29]. On 8 July 2020, the developers informed us that they were implementing a switch to the Signal protocol to provide cryptographic assurances in their SDK. This switch would rule out many of the attacks described in this work. We note, however, that we have not reviewed these changes and we recommend an independent security audit when they are complete.

References

- [1] Subnodes. <http://subnodes.org/>, 2018.
- [2] B.A.T.M.A.N. Advanced. <https://www.open-mesh.org/projects/batman-adv/wiki>, 2020.
- [3] Bridgefy twitter search. <https://twitter.com/search?q=bridgefy>, June 2020. <http://archive.today/2020.06.10-145953/https://twitter.com/search?q=bridgefy>.
- [4] Frida: a dynamic instrumentation framework, v12.8.9. <https://frida.re/>, February 2020.
- [5] Alexandre Adomnicai, Jacques J. A. Fournier, and Laurent Masson. Hardware security threats against Bluetooth Mesh networks. In *2018 IEEE Conference on Communications and Network Security, CNS 2018, Beijing, China, May 30 - June 1, 2018*, pages 1–9. IEEE, 2018.
- [6] Flor Álvarez, Lars Almon, Ann-Sophie Hahn, and Matthias Hollick. Toxic friends in your network: Breaking the Bluetooth Mesh friendship concept. In Maryam Mehrnezhad, Thyla van der Merwe, and Feng Hao, editors, *Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop, London, UK, November 11, 2019*, pages 1–12. ACM, 2019.
- [7] Joel Alwen, Margarita Capretto, Miguel Cueto, Chethan Kamath, Karen Klein, Guillermo Pascual-Perez, Krzysztof Pietrzak, and Michael Walter. Keep the dirt: Tainted treekem, an efficient and provably secure continuous group key agreement protocol. Cryptology ePrint Archive, Report 2019/1489, 2019. <https://eprint.iacr.org/2019/1489>.
- [8] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Security analysis and improvements for the IETF MLS standard for group messaging. Cryptology ePrint Archive, Report 2019/1189, 2019. <https://eprint.iacr.org/2019/1189>.
- [9] The Guardian Archie Bland. FireChat – the messaging app that’s powering the hong kong protests. <http://web.archive.org/web/20200328142327/https://www.theguardian.com/world/2014/sep/29/firechat-messaging-app-powering-hong-kong-protests>, September 2014.
- [10] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. DROWN: Breaking TLS using SSLv2. In Holz and Savage [45], pages 689–706.
- [11] B1O15J. <https://twitter.com/B1O15J/status/1294603355277336576>, August 2020. <https://archive.vn/dkPqD>.
- [12] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. Efficient padding oracle attacks on cryptographic hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 608–625. Springer, Heidelberg, August 2012.
- [13] Johannes K Becker, David Li, and David Starobinski. Tracking anonymized Bluetooth devices. *Proceedings on Privacy Enhancing Technologies*, 2019(3):50–65, 2019.
- [14] Divya Kala Bhavani. Internet shutdown? why Bridgefy app that enables offline messaging is trending in India. <http://web.archive.org/web/20200105053448/https://www.thehindu.com/sci-tech/technology/internet-shutdown-why-bridgefy-app-that-enables-offline-messaging-is-trending-in-india/article30336067.ece>, December 2019.
- [15] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *CRYPTO’98*, volume 1462 of *LNCS*, pages 1–12. Springer, Heidelberg, August 1998.
- [16] Hanno Böck, Juraj Somorovsky, and Craig Young. Return of bleichenbacher’s oracle threat (ROBOT). In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018*, pages 817–849. USENIX Association, August 2018.
- [17] Masha Borak. We tested a messaging app used by Hong Kong protesters that works without an internet connection. <http://web.archive.org/web/20191206182048/https://www.abacusnews.com/digital-life/we-tested-messaging-app-used-hong-kong-protesters-works-without-internet-connection/article/3025661>, September 2019.
- [18] Gage Boyle. 20 years of Bleichenbacher attacks. Technical Report RHUL-ISG-2019-1, Information Security Group, Royal Holloway University of London, 2019.

- [19] Thomas Brewster. Hong Kong protesters are using this ‘mesh’ messaging app—but should they trust it? <https://www.forbes.com/sites/thomasbrewster/2019/09/04/hong-kong-protesters-are-using-this-mesh-messaging-app--but-should-they-trust-it/>, September 2019. <http://web.archive.org/web/20191219071731/https://www.forbes.com/sites/thomasbrewster/2019/09/04/hong-kong-protesters-are-using-this-mesh-messaging-app--but-should-they-trust-it/>.
- [20] Bridgefy. Developers | bridgefy. <https://blog.bridgefy.me/developers.html>. <https://archive.vn/yjg9f>.
- [21] Bridgefy. <https://twitter.com/bridgefy/status/1197191632665415686>, November 2019. <http://archive.today/2020.06.06-094214/https://twitter.com/bridgefy/status/1197191632665415686>.
- [22] Bridgefy. <https://twitter.com/bridgefy/status/1209924773486170113>, December 2019. <https://web.archive.org/web/20200418131931/https://twitter.com/bridgefy/status/1209924773486170113>.
- [23] Bridgefy. <https://twitter.com/bridgefy/status/1216473058753597453>, January 2020. <http://archive.today/2020.06.06-095214/https://twitter.com/bridgefy/status/1216473058753597453>.
- [24] Bridgefy. <https://twitter.com/bridgefy/status/1268905414248153089>, June 2020. <http://archive.today/2020.06.06-093929/https://twitter.com/bridgefy/status/1268905414248153089>.
- [25] Bridgefy. <https://twitter.com/bridgefy/status/1287768436244983808>, July 2020. <https://archive.vn/WQfZm>.
- [26] Bridgefy. <https://twitter.com/bridgefy/status/1268015807252004864>, June 2020. <http://archive.today/2020.06.06-094807/https://twitter.com/bridgefy/status/1268015807252004864>.
- [27] Bridgefy. <https://twitter.com/bridgefy/status/1289576487004168197>, August 2020. <https://archive.vn/zbxgR>.
- [28] Bridgefy. <https://twitter.com/bridgefy/status/1292880821725036545>, August 2020. <https://archive.vn/tKr0t>.
- [29] Bridgefy. <https://twitter.com/bridgefy/status/1267469099266965506>, June 2020. <http://archive.today/2020.06.06-101916/https://twitter.com/bridgefy/status/1267469099266965506>.
- [30] Bridgefy. Bridgefy. <https://web.archive.org/web/20200411143157/https://www.bridgefy.me/>, April 2020.
- [31] Bridgefy. Bridgefy – offline messaging. <https://web.archive.org/20200411143133/https://play.google.com/store/apps/details?id=me.bridgefy.main>, April 2020.
- [32] Victor Cortés. Bridgefy sees massive spike in downloads during Hong Kong protests. <http://web.archive.org/web/20191013072633/http://www.contxto.com/en/mexico/mexican-bridgefy-sees-massive-spike-in-downloads-during-hong-kong-protests/>, August 2019.
- [33] Cas Cremers, Britta Hale, and Konrad Kohbrok. Efficient post-compromise security beyond one group. Cryptology ePrint Archive, Report 2019/477, 2019. <https://eprint.iacr.org/2019/477>.
- [34] John Paul Dunning. Taming the blue beast: A survey of bluetooth based threats. *IEEE Secur. Priv.*, 8(2):20–27, 2010.
- [35] Thai Duong and Juliano Rizzo. The crime attack. *Presentation at ekoparty Security Conference*, 2012.
- [36] Ksenia Ermoshina, Harry Halpin, and Francesca Musiani. Can johnny build a protocol? co-ordinating developer and user intentions for privacy-enhanced secure messaging protocols. In *European Workshop on Usable Security*, 2017.
- [37] David Fifield. A better zip bomb. In *13th USENIX Workshop on Offensive Technologies (WOOT 19)*, Santa Clara, CA, August 2019. USENIX Association.
- [38] Sadayuki Furuhashi. Messagepack. <https://msgpack.org/>, 2008.
- [39] Open Garden. Firechat. <http://web.archive.org/web/20200111174316/https://www.opengarden.com/firechat/>, October 2019.
- [40] Paul Gardner-Stephen. The serval project. <http://www.servalproject.org/>, 2017.
- [41] Christina Garman, Matthew Green, Gabriel Kaptchuk, Ian Miers, and Michael Rushanan. Dancing on the lip of the volcano: Chosen ciphertext attacks on apple iMessage. In Holz and Savage [45], pages 655–672.

- [42] Yoel Gluck, Neal Harris, and Angelo Prado. Breach: reviving the crime attack. *Black Hat USA*, 2013.
- [43] Harry Halpin, Ksenia Ermoshina, and Francesca Musiani. Co-ordinating developers and high-risk users of privacy-enhanced secure messaging protocols. In Cas Cremers and Anja Lehmann, editors, *Security Standardisation Research - 4th International Conference, SSR 2018*, volume 11322 of *Lecture Notes in Computer Science*, pages 56–75. Springer, 2018.
- [44] Shaikh Shahriar Hassan, Soumik Das Bibon, Md. Shohrab Hossain, and Mohammed Atiquzzaman. Security threats in Bluetooth technology. *Comput. Secur.*, 74:308–322, 2018.
- [45] Thorsten Holz and Stefan Savage, editors. *USENIX Security 2016*. USENIX Association, August 2016.
- [46] HypeLabs. The Hype SDK: A technical overview. <https://hype4labs.io/documents/Hype-SDK.pdf>, 2019.
- [47] HypeLabs. <https://hype4labs.io>, 2020.
- [48] IETF. Deflate compressed data format specification version 1.3. <https://tools.ietf.org/html/rfc1951>, May 1996.
- [49] IETF. Gzip file format specification version 4.3. <https://tools.ietf.org/html/rfc1952>, May 1996.
- [50] IETF. Pkcs #1: Rsa encryption version 1.5. <https://tools.ietf.org/html/rfc2313>, March 1998.
- [51] Slawomir Jasek. GATTacking Bluetooth smart devices. <https://github.com/securing/docs/raw/master/whitepaper.pdf>, 2016.
- [52] Tech in Asia Josh Horwitz. Unblockable? unstoppable? FireChat messaging app unites China and Taiwan in free speech... and it's not pretty. <http://web.archive.org/web/20141027180653/> <https://www.techinasia.com/unblockable-unstoppable-firechat-messaging-app-unites-china-and-taiwan-in-free-speech-and-its-not-pretty/>, March 2014.
- [53] John Kelsey. Compression and information leakage of plaintext. In Joan Daemen and Vincent Rijmen, editors, *FSE 2002*, volume 2365 of *LNCS*, pages 263–276. Springer, Heidelberg, February 2002.
- [54] Vlastimil Klíma, Ondrej Pokorný, and Tomás Rosa. Attacking RSA-based sessions in SSL/TLS. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 426–440. Springer, Heidelberg, September 2003.
- [55] John Koetsier. Hong Kong protestors using mesh messaging app china can't block: Usage up 3685%. <https://www.forbes.com/sites/johnkoetsier/2019/09/02/hong-kong-protestors-using-mesh-messaging-app-china-cant-block-usage-up-3685/>, September 2019. <https://web.archive.org/web/20200411154603/https://www.forbes.com/sites/johnkoetsier/2019/09/02/hong-kong-protestors-using-mesh-messaging-app-china-cant-block-usage-up-3685/>.
- [56] Teknologii Lebanon. Lebanese Protesters are using this 'Bridgefy' Messaging App — What is it? <https://medium.com/@teknologiialb/lebanese-protesters-are-using-this-bridgefy-messaging-app-what-is-it-74614e169197>, January 2020. <https://archive.vn/udqly>.
- [57] Alex T Magaisa. <https://twitter.com/wamagaisa/status/1288817111796797440>, July 2020. <http://archive.today/DVRZf>.
- [58] Regina Mihindukulasuriya. Firechat, Bridgefy see massive rise in downloads amid internet shutdowns during caa protests. <http://web.archive.org/web/20200109212954/https://theprint.in/india/firechat-bridgefy-see-massive-rise-in-downloads-amid-internet-shutdowns-during-caa-protests/340058/>, December 2019.
- [59] Pavithra Mohan. How the internet shutdown in Kashmir is splintering India's democracy. <http://web.archive.org/web/20200408111230/https://www.fastcompany.com/90470779/how-the-internet-shutdown-in-kashmir-is-splintering-indias-democracy>, March 2020.
- [60] Farai Mudzingwa. This offline messenger that might keep you connected if the govt decides to shut down the internet. <https://www.techzim.co.zw/2020/07/bridgefy-is-an-offline-messenger-that-might-keep-you-connected-if-the-govt-decides-to-shut-down-the-internet/>, August 2020. <https://web.archive.org/web/20200816101930/https://www.techzim.co.zw/2020/07/bridgefy-is-an-offline-messenger-that-might-keep-you-connected-if-the-govt-decides-to-shut-down-the-internet/>.
- [61] BBC News. Iraqis use Firechat messaging app to overcome net block. <http://web.archive.org/web/20190325080943/https://www.bbc.com/news/technology-27994309k>, 2014.

- [62] Hacker News. Hong kong protestors using bridgefy's bluetooth-based mesh network messaging app. <https://web.archive.org/web/20191016114954/https://news.ycombinator.com/item?id=20861948>, August 2019.
- [63] Ben Ng. Bridgefy: A startup that enables messaging without internet. <https://www.ejinsight.com/eji/article/id/2230121/20190826-bridgefy-a-startup-that-enables-messaging-without-internet>, August 2019. <http://archive.today/2020.06.07-120425/https://www.ejinsight.com/eji/article/id/2230121/20190826-bridgefy-a-startup-that-enables-messaging-without-internet>.
- [64] Kunal Purohit. Whatsapp to Bridgefy, what Hong Kong taught India's leaderless protesters. <http://web.archive.org/web/20200406103939/https://www.scmp.com/week-asia/politics/article/3042633/whatsapp-bridgefy-what-hong-kong-taught-indias-leaderless>, December 2019.
- [65] Michael Rogers, Eleanor Saitta, Torsten Grote, Julian Dehm, and Benedikt Wieder. Briar. <https://web.archive.org/web/20191016114519/https://briarproject.org/>, March 2018.
- [66] Mike Ryan. Bluetooth: With low energy comes low security. In *Proceedings of the 7th USENIX Conference on Offensive Technologies*, WOOT'13, page 4, USA, 2013. USENIX Association.
- [67] Leo Schwartz. The world's protest app of choice. <https://web.archive.org/web/20200816101331/https://restofworld.org/2020/the-worlds-protest-app-of-choice/>, August 2020.
- [68] Bluetooth SIG. Core specification 5.1. <https://www.bluetooth.com/specifications/bluetooth-core-specification/>, Jan 2019.
- [69] Bluetooth SIG. Mesh profile specification 1.0.1. <https://www.bluetooth.com/specifications/mesh-specifications/>, Jan 2019.
- [70] Matthew De Silva. Hong Kong protestors are once again using mesh networks to preempt an internet shutdown. <https://qz.com/1701045/hong-kong-protestors-use-bridgefy-to-preempt-internet-shutdown/>, September 2019. <http://archive.today/2019.09.20-220517/https://qz.com/1701045/hong-kong-protestors-use-bridgefy-to-preempt-internet-shutdown/>.
- [71] Pallavi Sivakumaran and Jorge Blasco. A study of the feasibility of co-located app attacks against BLE and a large-scale analysis of the current application-layer security landscape. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019*, pages 1–18. USENIX Association, August 2019.
- [72] Skylot. Jadx - dex to java decompiler, v1.1.0. <https://github.com/skylot/jadx>, Dec 2019.
- [73] SMEX. Lebanon protests: How to communicate securely in case of a network disruption. <http://web.archive.org/web/20200408112421/https://smex.org/lebanon-protests-how-to-communicate-securely-in-case-of-a-network-disruption-2/>, October 2019.
- [74] India Software Freedom Law Centre. Internet shutdown tracker. <https://internetshutdowns.in/>, 2020.
- [75] William Stein et al. *Sage Mathematics Software Version 9.0*. The Sage Development Team, 2019. <http://www.sagemath.org>.
- [76] The Stranger. How to message people at protests even without internet access. <https://www.thestranger.com/slog/2020/06/03/43829749/how-to-message-people-at-protests-even-without-internet-access>, June 2020. <http://archive.is/8UrWQ>.
- [77] Nick Sullivan, Sean Turner, Benjamin Kaduk, Katriel Cohn-Gordon, et al. Messaging Layer Security (mls). <https://datatracker.ietf.org/wg/mls/about/>, November 2018.
- [78] Jason Uher, Ryan G. Mennecke, and Bassam S. Farroha. Denial of sleep attacks in Bluetooth Low Energy wireless sensor networks. In Jerry Brand, Matthew C. Valenti, Akinwale Akinpelu, Bharat T. Doshi, and Bonnie L. Gorsic, editors, *2016 IEEE Military Communications Conference, MILCOM 2016, Baltimore, MD, USA, November 1-3, 2016*, pages 1231–1236. IEEE, 2016.
- [79] Jane Wakefield. Hong Kong protestors using Bluetooth Bridgefy app. <http://web.archive.org/web/20200305062625/https://www.bbc.co.uk/news/technology-49565587>, September 2019.
- [80] Chaoshun Zuo, Haohuang Wen, Zhiqiang Lin, and Yinqian Zhang. Automatic fingerprinting of vulnerable BLE IoT devices with static UUIDs from mobile apps. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1469–1483. ACM, 2019.