Can Johnny Finally Encrypt? Evaluating E2E-Encryption in Popular IM Applications

Amir Herzberg
Dept. of Computer Science
Bar Ilan University

Hemi Leibowitz
Dept. of Computer Science
Bar Ilan University

ABSTRACT

Recently, many popular Instant-Messaging (IM) applications announced support for end-to-end encryption, claiming confidentiality even against a rogue operator. Is this, finally, a positive answer to the basic challenge of usable-security presented in the seminal paper, 'Why Johnny Can't Encrypt' [29]?

Our work evaluates the implementation of end-to-end encryption in popular IM applications: WhatsApp, Viber, Telegram, and Signal, against established usable-security principles, and in quantitative and qualitative usability experiments. Unfortunately, although participants expressed interest in confidentiality, even against a rogue operator, our results show that current mechanisms are *impractical to use*, leaving users with only the *illusion* of security.

Hope is not lost. We conclude with directions which may allow usable End-to-End encryption for IM applications.

1. INTRODUCTION

Instant-Messaging (IM) applications have become a predominant media for person-to-person communication, including sensitive communication. Alongside growing concerns about privacy and the integrity of computer communication, including surveillance [9], it has become vital to protect IM communication against attacks, including powerful MitM attackers; this kind of protection requires end-to-end encryption. Indeed, end-to-end encryption was recently integrated into popular messaging services, claiming defense (even) against a rogue MitM operator.

While IM applications are designed for maximal usability, and are indeed used by billions of users, it is challenging to provide *usable encryption*. The goal of usable encryption is to give every standard user (aka 'Johnny') the ability to securely encrypt private messages sent over insecure channels. The challenge of usable encryption received significant attention following the seminal paper 'Why Johnny Can't Encrypt' [29], showing that PGP encryption is not sufficiently usable, i.e., users often fail to properly encrypt. Over the years, researchers have repeatedly found [7, 20–23] that such systems fail to provide usable encryption.

This raises the natural question: do these IM applications really provide *usable, secure end-to-end encryption*? A positive answer would mean that the long outstanding challenge of usable secure encryption [7, 20, 22, 23, 29] has finally been overcome - at least for the important use case of IM applications.

We evaluated the usability of *WhatsApp, Viber* and *Telegram*, which are three of the most popular IM applications, along with *Signal*, all under Android. Both WhatsApp and Viber recently integrated end-to-end encryption automatically for all of their conversations. WhatsApp recently announced to have more than one billion users and Viber is also highly popular with a successful VoIP service. In contrast to WhatsApp and Viber, which are proprietary applications, Telegram is an open-source cloud-based service. Therefore, by default, messages are only client-server encrypted. Users can *actively* create 'secret chats', in which all messages are end-to-end encrypted. Finally, Signal, with a significantly smaller user base compared to other popular services, has a reputation of providing strong privacy and end-to-end encryption. Both WhatsApp and Viber's end-to-end encryption implementations (along with Facebook Messenger and Google Allo) are based on Signal's security protocol. That said, all four applications have different approaches and characteristics that set them apart from each other with regard to end-to-end encryption. (See Table 1 for a summary of end-to-end encryption properties in these applications.) For a detailed explanation on why we focused on these services and did not include other services, see Appendix A.

All four applications support *two modes* of end-to-end encryption, which differ considerably in their user experience and security properties. One mode is an *opportunistic end-to-end encryption mode*, which requires almost no user interaction. The other is an *authenticated end-to-end encrypted mode*, which requires considerable user interaction. The *opportunistic* mode is invoked *automatically* for all conversations (in Telegram, only in 'secret chats'). Every application uses different indicators, showing the user that the conversation is encrypted, the mode of encryption, and changes in the key of the peer. But, end-to-end encryption does not only involve a challenge of 'person-to-person' communication. Supporting end-to-end encryption in group chats and in multiple devices linked to the same account goes beyond trivial. It introduces an array of usability problems to an already complex task. These include authenticating

STAST '16 Los Angeles, CA USA

DOI:

large groups - especially where group members are not always familiar with each other, or synchronizing End-to-End encrypted messages among devices.

In the security and cryptographic literature, the term 'end-to-end encryption' is generally used only for encryption that offers a defense against (even) a rogue (MitM) operator - as provided only in 'authenticated' mode. In contrast, the security of the opportunistic mode only defends against a *passive (honest but curious)* operator (or MitM) attacker. Opportunistic mode may seem to be a manifestation of opportunistic security [11], but because an active MitM attacker can perform a 'key reset attack', the opportunistic mode is actually weaker and does not achieve opportunistic security. Unfortunately, the relatively-complex notion of 'honest-but-curious' attacker is not presented to users, whether in the user-interface, manual, or web sites of the applications. When we tried to explain this 'honest-but-curious' security model to users, they consistently failed to recognize it as providing an additional security benefit compared to encryption that is only between client and server. They did not agree to consider it as providing an acceptable, if weaker, form of end-to-end encryption. In fact, a common response was that users felt *misled* by the claims of end-to-end encryption, especially since all applications mention protection from the operator ('not even we can read your messages').

Our evaluation focuses on the 'authenticated' end-to-end encryption mode. Only this mode corresponds to the classical notion of end-to-end encryption, and ensures confidentiality even against a rogue operator. Authenticated end-to-end encryption is required to protect against privacy breaches by powerful adversaries who may coerce or manipulate operators. This is also the goal explored in previous works on usable-security for encryption [7, 20–23, 29].

Using authenticated mode in all four of the IM applications evaluated, requires the user to perform a special *authentication ceremony*. Users are required to perform a set of manual operations to verify that they are indeed talking directly with one another. This set of operations is known as a 'ceremony' [5], because of the ritualistic, auto-response behavior that users develop when performing them, especially because they need to be performed repeatedly. However, each application implements the ceremony differently. Because correct user interaction in these ceremonies is critical to ensuring security against a rogue operator, this begs an obvious question. *Do users who care about confidentiality against a rogue operator know they need to invoke these ceremonies and how to complete them?*

Unfortunately, we found several usability failures in all the applications we evaluated. Our evaluation is based on usable-security principles, usability experiments measuring the proper use of these mechanisms, and a qualitative evaluation of participants' responses. Our main findings are:

Users want protection from rogue operators. Somewhat surprisingly, we found that most users do care about confidentiality against rogue operators. This shows that the IM applications made a wise choice in adding such features.

Users are unaware of opportunistic mode. Users consistently failed to understand the need to move to authenticated mode in order to ensure privacy against rogue operators. In fact, *none* of the users realized that opportunistic mode implies that a rogue, active operator can breach privacy. Consequently, users are misled into believing they are protected from a rogue operator when using opportunistic mode, which is vulnerable to such adversaries.

The authentication ceremony is not usable. Users were mostly unable to perform the authentication process based on the user interface and documentation. Furthermore, even when users were shown how to authenticate, they generally failed to manage it. For the low percentage of users who succeeded in using it, the process was very difficult and slow. We also found basic usability and accessibility failures in the authentication ceremonies and indicators, such as failures for color-blind users.

Users fail to authenticate upon reset. In all four IM applications, the key of a peer may change. A key change may occur for legitimate reasons, such as re-installation of the IM application, but can also be the result of an active attack (by the operator or by an attacker). If the conversation was in 'opportunistic mode' (not authenticated), users are unlikely to notice the 'reset' and even more unlikely to authenticate or otherwise validate the connection. 'opportunistic mode' does not even provide the security guarantees of 'Trust on First Use (TOFU)' - and even TOFU is argued to provide insufficient protection [13]. Furthermore, even if the conversation was authenticated, a key-change returns it to the 'opportunistic' mode, and users are unlikely to notice or to properly re-authenticate. In short, an attacker can eavesdrop on a (previously) protected end-to-end conversation, even if users performed authentication.

To summarize, we believe that *'opportunistic mode' gives users a false sense of security.* Users incorrectly believe that applications offer protection against rogue operators, without understanding that this requires authentication. Even when they do understand this, they do not re-authenticate upon a key reset caused by an attack or benign reasons. Furthermore, the authentication process is not really usable. This implies that it is very rarely used and is unlikely to be used even by the (very few) users who are *both* diligent and savvy. We stress that our evaluation - and negative results - focus on the issue of *usability of end-to-end encryption.* There are other aspects of security and privacy for IM applications that we did not evaluate, e.g., [12, 14, 25, 27].

1.1 Related Work

Achieving usable-security is challenging. In the seminal paper "Why Johnny Can't Encrypt" [29], Whitten and Tygar argue that it is hard to "force" behavior on users. They observed how people behave using a PGP client, and even tried to educate them to use the system correctly. One of their conclusions was that educating people to use a system securely is hard to achieve. These findings triggered a lot of follow up research, e.g., Garfinkel et al. [15], who interpreted Whitten and Tygar's findings differently. They claimed that the observed results were influenced by the underlying key certification model, and that by replacing it, they could achieve better results.

Ruoti et al. [21] explored the effects of delivering security transparently. While this might seem like a good idea, they found that in some cases, the fact that the security was 'completely' transparent to users, caused users to behave

Property	WhatsApp	Viber	Telegram	Signal
Opportunistic mode	By default	By default	By default, but only in 'secret chats'	By default
Encryption mode indicators	Opportunistic only, using a message	Colored lock icon reflecting all encryption modes	Opportunistic only, using a message	Opportunistic only, static lock icons, display key change
Authentication ceremony	Manual key comparison or QR scan	Manual key comparison via Viber call	Manual key or image comparison	Manual key comparison (QR requires separate application)
Key reset indicators	Message, not enabled by default	Lock icon changing color	No	Obtrusive notification
end-to-end encrypted group chat	Opportunistic only using a message	Opportunistic only using lock icon	No	Opportunistic only, static lock icons
Standalone secondary device	No	Yes	Yes	Yes
Web & Desktop support	Yes, Mirroring only	Desktop only	No	Desktop only

Table 1: Summary of end-to-end encryption properties implemented in WhatsApp, Viber, Telegram, and Signal

incorrectly, and even expose their security. Moreover, people who had to perform manual encryption, reported a stronger sense of security.

Unger et al. [27] compared existing systems of secure messaging. They found that even though there are many different services and approaches, there are still some core problems without a decent *practical* solution. One of the reasons was insufficient user community. Many of the systems tested, attempt to achieve security and anonymity attributes; however, without a large user base, client anonymity cannot be achieved because "anonymity loves company" [10]. Assal et al. [2] evaluated the usability of privacy oriented services, i.e. ChatSecure, ObscuraCam and Tor, and found usability deficiencies in these solutions. In [26], Schröder et al. evaluated the security and usability of Signal. In this work, we focus on popular services that have billions of users, and cover many aspects in a broader manner, some which are not relevant to Signal's implementation.

Several works [17, 24] examined how security indicators are perceived by users. The common findings are that users tend to ignore or misinterpret security signs, but there are approaches that can significantly improve users' understanding and yield better results. For example, these include active and highly-visible alerts, security enabled by default, and more.

1.2 Contributions

The contributions of this work are as follows:

1. We analyze the usability requirements and challenges of end-to-end encryption for IM applications.
2. We discuss the key-reset vulnerability and explain why it is likely to succeed against the tested IM applications.
3. We apply usable-design principles to analyze the usability of four popular IM applications.
4. We present the findings of usability studies showing that all four applications do not offer usable authenticated end-to-end encryption; in particular, users are shown to be vulnerable to the key-reset attack.

2. END-TO-END ENCRYPTION FOR IM APPLICATIONS

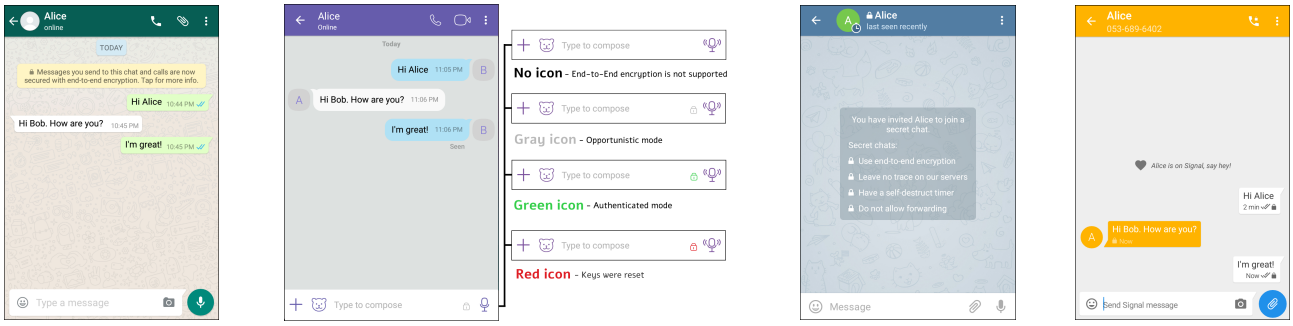
Most messaging services operate in client-server model, where users relay their messages through the operator's servers. A straightforward way to protect communication from an unauthorized third-party is to encrypt the communication between each client and the server; however, this does not protect the user's privacy from the operator itself. With end-to-end encryption, messages are encrypted before they leave the sender's device and can only be decrypted by their designated recipient. This way, even though messages are relayed through servers and networks, no one other than the recipient can decrypt them.

To ensure that no one other than endpoint users can decrypt the messages, endpoint users have to employ a secure way to agree on cryptographic keys. This is a complicated task in general, especially in the world of mobile messaging. Operators cannot rely on people to have a secure off-band channel where they can agree on their cryptographic keys. Moreover, if this process is not done transparently and requires action by the clients, it may result in failure. The million-dollar question is: how can users agree on cryptographic keys in a way that is both usable and secure (even against the operator itself)?

Despite its attractive properties, end-to-end encryption was not supported by popular IM applications until recently. Not long ago, several popular IM applications incorporated support for end-to-end encryption, with two modes: *opportunistic mode* and *authenticated mode*. In *opportunistic mode*, the two applications perform key-exchange, authenticated only by the operators; hence, this process is vulnerable to a malicious, active operator, and protected against a passive attacker or an active MitM attacker who fails to circumvent the authentication by the operator. In *authenticated mode*, the user is required to perform a special *authentication ceremony*. Even after the ceremony was performed, an attacker may launch a key-reset attack to achieve MitM capabilities. We now elaborate on these adversary models.

2.1 Adversary Models

Users may be misled into thinking that 'authentication' is not related to the confidentiality of their conversations, but this is incorrect; the two modes actually refer to two different *adversary models*. Furthermore, *both* of these modes differ from the classical notion of end-to-end encryption, as inves-



(a) WhatsApp **(b)** Viber **(c)** Telegram **(d)** Signal

Figure 1: end-to-end encryption messages and indicators in all four applications. WhatsApp and Telegram notify users on every chat screen that end-to-end encryption exists for messages, but do not reflect whether the conversation was authenticated. In Telegram, once the chat screen contains even one message, the end-to-end encryption message disappears. Viber uses a colored lock icon to indicate different modes of the encryption. Signal does not employ any special indication of encryption mode, except for constant lock icons, which are also employed by other applications, like Telegram.

tigated in the academic literature. To add to the confusion, the 'opportunistic mode' is *similar* to another adversary model. This other model is used in several designs that try to achieve an easier-to-deploy notion of end-to-end encryption (cf. the 'classical' notion), albeit with weaker security guarantees (i.e., assuming a weaker adversary model). The situation is further complicated by the existence of MitM attackers who can cause key resets. The result is several non-traditional adversaries with different capabilities. We now try to explain the difference between these models.

Internet-only MitM: In this model, the attacker has only MitM capabilities on Internet communication between IM clients and servers; however, it does not have any access to the IM operator's systems. For example, a network-only MitM attacker can eavesdrop, inject, modify, or drop traffic between clients and servers (including client-to-client, client-to-server, and server-to-server communication). However, she cannot control servers, access information stored and processed by a server, or modify client or server software.

Internet and telephony MitM: In this stronger MitM attacker model, the attacker can further interfere in communication over the phone network. Numerous vulnerabilities exist in the phone network, especially for mobile phones; hence, this is a very realistic attack model. The added capabilities of this attacker are significant because IM applications usually use the client's phone number as a unique identifier. This allows an attacker to assume the identity of other clients.

Passive-server-exposures: In addition to its network MitM capabilities, this attacker also has read-only (passive) access to the servers (e.g., due to subpoena).

Corrupt servers: In addition to network MitM capabilities and passive-server-exposures, this attacker can also control the servers. For example, this may be due to intrusion or insider cooperation by an IM service provider employee, the operator of a cloud service used by the IM provider, or an employee of the cloud service.

TOFU (then corrupt) servers: This is similar to the corrupt servers model, but we assume that corruption only occurs *after* first use. The servers are 'trusted', or have only passive exposures, until and during the first communication between users.

Restricted device access: Attackers in this model have a limited time frame in which they can physically perform semi-passive operations on the victim's device. This means they cannot install or execute malicious code but can invoke operations on the already installed applications (e.g., read verification messages displayed on lock screen, link a secondary device to an existing account, etc.). A variation of this model is when the attacker gains access to the SIM card of the victim for a short time, undertaking the victim's phone identity to obtain his IM application identity.

Out of scope: We do not discuss adversary models involving breaching end-system security, such as forward and future security, since these are less related to the usability aspects. Accordingly, we assume that end-host software is not corrupted.

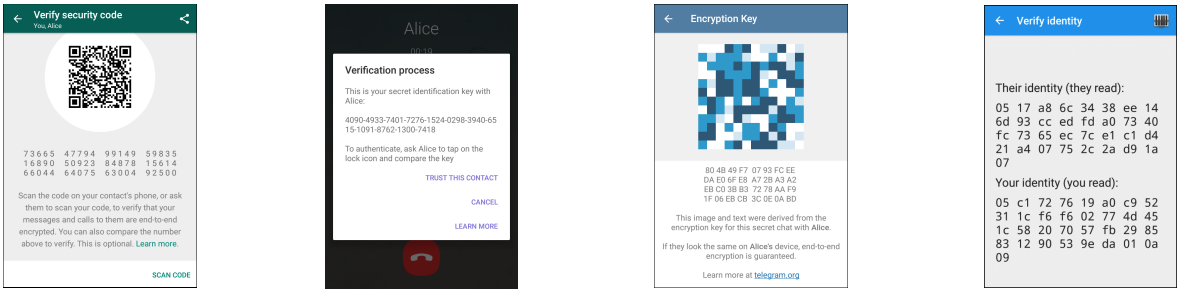
3. CURRENT IMPLEMENTATIONS

Incorporating end-to-end encryption into IM applications involves several major challenges, which are a direct by-product of the main IM building blocks. These building blocks comprise three basic features: '1-on-1' chats, group chats, and multiple device support (portability). We explain how each application implemented the opportunistic mode and the authenticated mode, and how it handles key resets. Next, we explain how they handle end-to-end encryption in group chats and their multiple-device support for end-to-end encryption. Our study focuses on the usability of the implementations and the security that is derived from it, but not on the security of the cryptographic implementation of end-to-end encryption.

3.1 Encryption Modes and Indicators

WhatsApp. In WhatsApp, users are informed that chats are end-to-end encrypted by a special notification message (see Figure 1a). This message appears at the beginning of every chat, and in some cases, during the chat as well.

Viber. In Viber, the mode of encryption is indicated using a lock icon located near the input text field of the contact's chat screen. When no icon appears, this means end-to-end encryption is not supported by one of the end-point devices. A gray icon means that end-to-end encryption is in opportunistic mode. The lock icon also has other possible colors, indicating whether the contact is authenticated



(a) WhatsApp **(b)** Viber **(c)** Telegram **(d)** Signal

Figure 2: Authentication ceremony for different applications

or not. For a complete description of the different modes, see Figure 1b.

Telegram. In Telegram, a cloud-based service, messages are only client-server encrypted by default and stored securely on Telegram's servers. Users who wish to use opportunistic end-to-end encryption, need to *actively* create a *secret chat*, in which all messages are opportunistically end-to-end encrypted. An explanation message on the contact's secret chat screen explains that messages in the chat are end-to-end encrypted (see Figure 1c).

Signal. As in WhatsApp and Viber, all messages are end-to-end encrypted by default; however, other than constant lock icons, which appear on the chat screen, there are no special messages or indicators of end-to-end encryption (see Figure 1d).

Discussion. The need for authentication is not directly displayed to users, and requires probing in order for them to stumble upon the mechanism. Although Viber does have an icon that reflects the current mode of encryption, it is likely to be ignored or misinterpreted by many users. In opportunistic mode, although not presented, users are actually not protected from a rogue operator (e.g., 'corrupt servers' attacker). A user has no way of knowing if he receives the wrong encryption key for a contact, and therefore, opportunistically, must trust the operator to be honest. In such scenarios, the user thinks his messages are end-to-end encrypted, when in reality, he shares a key with the attacker. When it comes to both security and usability, not encrypting all messages by default is very controversial.

3.2 Authentication Ceremony

In all evaluated applications, users must perform an authentication ceremony in order to activate authenticated end-to-end encryption. This ceremony ensures they are really talking directly with each other, and no one else is eavesdropping on messages sent between them. WhatsApp, Viber, Telegram, and Signal, have all implemented a built-in authentication mechanism, allowing Alice and Bob to ensure their end-to-end encrypted conversation has not been compromised. The main idea of authentication is similar in all applications: compute and display a unique identifier based on Alice and Bob's identities. If Alice and Bob are indeed talking to one another without any MitM, the unique identifier that is computed and associated with their chat is identical on both ends. If they are talking through a MitM and actually sharing keys with the MitM (and not directly with one another), the chat's unique identifier will be different at both endpoints. We now explore how each application implemented the authentication mechanism.

WhatsApp. In WhatsApp, clients can authenticate one another in one of two ways: they can either scan a QR code or compare a key of 60 characters (see Figure 2a). When users scan the QR code, the result is displayed only in the validating (scanning) screen.

Viber. Clients can authenticate one another by engaging in a dedicated Viber call. During the call, they both need to press a designated lock button and compare strings of 48 characters (see Figure 2b). Finally, users need to inform Viber that the keys were identical; this is done by pressing an approval button. When the ceremony is complete, the lock icon turns green, indicating that the conversation has been authenticated (see Figure 1b).

Telegram. Clients can either compare a key visualization image or compare a 64 character-long code (see Figure 2c). The key visualization image is a group of colored squares, grouped together in a unique form according to the key.

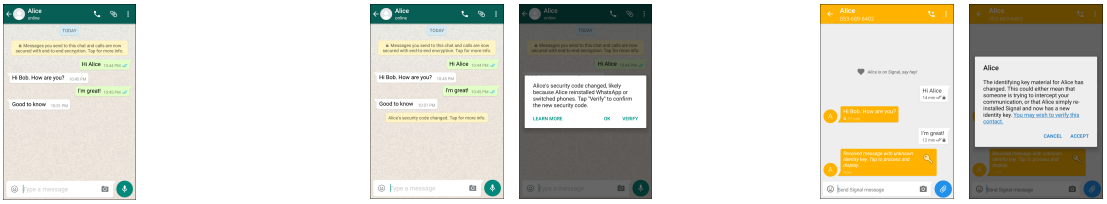
Signal. In Signal, authentication is performed using two keys: a key the user is expected to read, and a separate key that the other contact is expected to read. Each key is 66 characters long, resulting in a total of 132 characters. Alternatively, users can scan a QR representation of the key. Unfortunately, QR scanning and code generation is not incorporated into Signal. Users have to download and install a separate application to generate and read the QR code.

Discussion. In WhatsApp, Telegram, and Signal, there is an implicit assumption that users have a secure channel they can use to compare the identifier, either physically or remotely. This is a non-trivial assumption, especially since the QR method assumes co-location, which is often unrealistic. On the other hand, WhatsApp, Viber, and Telegram, do present an explanation text saying that this authentication is required to 'guarantee' end-to-end encryption, but this appears only after the ceremony is initiated.

3.3 Key Reset

In all IM applications we evaluated, the end-to-end encryption keys used by a peer may change. This can occur due to benign reasons, such as re-installation of the application, or due to an attacker attempting to impersonate the peer (such that the new key will be known to the attacker). This scenario is especially important in 'authenticated mode', because the peer is no longer authenticated and the authentication ceremony needs to be performed again. Each of the applications handle such key-reset events differently.

WhatsApp. By default, if a peer's key changed, the user is not informed of the event (see Figure 3a). Users can enable a notification feature via the application's menu, which will inform the user via a message (see Figure 3b).



(a) WhatsApp's default after reset **(b)** WhatsApp's enabled feature for resets **(c)** Signal's default after key reset

Figure 3: In all figures, key reset was performed between the third and fourth message. Figure 3a demonstrates WhatsApp's default behavior, which is to ignore the change. Figure 3b demonstrates how WhatsApp handles resets after the appropriate mechanism was enabled (displays a message and extra explanation on click). In Signal, the mechanism is enabled by default.

Viber. When in opportunistic mode, there is no indication of key reset. In authenticated mode, the green colored lock icon turns red (see Figure 1b).

Telegram. Two users can simultaneously have several separate active secret chats. If a user deletes a secret chat from his application, the other client is notified by a message that prevents him from sending further messages via this chat. However, if a user re-installs the application, he needs to create a new secret chat with his new keys. This does not prevent the peer from continuing to send messages using the 'old' (no longer valid) secret chats, even though these messages will never arrive at the new installation. Because each secret chat is separate from other secret chats, even with the same contact, each chat needs to be authenticated separately.

Signal. Signal obtrusively informs users when a key was changed. A message appears in the chat notifying the user of the change and why this may have happened (see Figure 3c). Users can choose to perform authentication or to accept the replaced keys without authentication.

Discussion. The *key reset vulnerability* allows an attacker to gain substantial capabilities. For example, an Internet & Telephony MitM attacker - Mallory, can assume the identity of Bob, but cannot access the previously sent messages because they are on Bob's device. By default, in WhatsApp, he can continue to talk with Alice, without Alice noticing that anything is amiss. If Alice does enable the notification feature, she will see a message that Bob's key has changed. In Telegram, Mallory can create a new secret chat with Alice, impersonating Bob. Although Viber would change the lock icon to red in such scenario, unless Alice notices it *and* understands what it means, Mallory can talk to Alice. Interestingly, in Signal, even though authentication does not change the mode of end-to-end encryption, users are obliged to actively allow the communication after key change was detected. Unfortunately, as our study showed, users do not interpret or handle this message correctly.

3.4 Group Chats

Secure provisioning of end-to-end encrypted group chats introduces additional usability challenges. In contrast to regular one-on-one chats, where messages are only encrypted once according to the recipient keys, group chats usually involve many group members. Normally, without end-to-end encryption, this can be easily handled by the server: the client sends the message to the server, who then distributes it to the rest of the members. In end-to-end encryption, the server cannot do that because it does not have the end-to-end encryption keys of the group members, and it does not have access to the message itself. Encrypting *every* message at the client according to *each* member's end-to-end encryption keys, has a significant overhead, because the client needs to send each message multiple times. Using a shared group key simplifies that overhead, but introduces security challenges.

In Telegram, end-to-end encrypted group chats are not included. In the other applications, the usability attributes of end-to-end encrypted group chats are similar to those of their one-on-one chats. The authentication is not group-based but rather pairwise based; but there are no special indicators for groups.

Discussion. Authenticating a single user is not a simple task, so how are users expected to authenticate an entire group? Group authentication actually relies on pairwise individual authentication, which is already questionable on its own. This is even more challenging when group members are not necessarily acquainted with one another, making authentication a lot more complex.

3.5 Support for Multiple Devices

In order to guarantee end-to-end encryption, both the data and cryptographic keys are stored on the user's device. However, users often want to use multiple devices, for example, to link their smartphone and tablet to the same account in parallel. Others prefer to use non-mobile interfaces, like web and desktop interfaces, either as their primary or secondary interface. How can end-to-end encryption be guaranteed and implemented correctly when the user is no longer bound to a single (primary) device?

WhatsApp. The user's mobile device is his *primary* (and only) device. A user can switch to another device, but cannot simultaneously have two *standalone* devices with the same account. Secondary devices are supported for web and desktop versions, but only by *mirroring* the primary device. In other words, messages are sent to the primary device, decrypted there, and then re-encrypted and relayed to the secondary device. In the opposite direction, messages sent from secondary devices are sent to the primary device, where they are decrypted, re-encrypted (according to the actual recipient) and re-sent. The primary and secondary device are linked by scanning a QR code. Users can see the linked devices and history via the application's menu.

Viber. A secondary device is a completely standalone device, and can be linked to a primary device by scanning a QR code. Viber has a desktop interface but no web interface. Users can see linked devices and history via the application's menu.

Telegram. Multiple devices can be linked to the same account. These devices share the cloud chats but maintain different end-to-end encrypted chats. These chats are stored locally and can only be accessed by the specific device. Web and desktop interfaces are standalone, but do not support

end-to-end encrypted 'secret chats'. Users can see linked devices and history via the application's menu.

Signal. Signal supports a Chrome desktop application as a standalone secondary device. Users can see linked devices and history via the application's menu.

Discussion. WhatsApp's mirroring involves extra in the "relay" process. Moreover, the device has to be online and accessible for the interface to function. For example, users cannot use these interfaces while their devices are broken or shutdown. On the other hand, maintaining multiple standalone devices also has an overhead. Every message has to be encrypted and sent multiple times to multiple devices, when most of the time users are actually operating only one device.

4. ANALYSIS OF SECURE USABILITY

Usability is critical to the success of any system, and often even more vital than security. In particular, users prefer simplicity and ease-of-use over security. While the most popular highly-secure services have millions of users, popular messaging services (which provide less secure services) have hundreds of millions or even billions of users. Gutmann and Grigg [16] noted that although it is hard to determine the correct relation (and trade-offs) between security and usability, security must follow usability, and not the other way around.

Focusing on usability must take into consideration human nature and habits. Humans have a tendency to develop behavioral patterns and to act according to the pattern that best suits the current situation. Therefore, if you design a system that requires behavioral patterns that contradict popular existing patterns, you are destined to fail. These patterns are also known as *ceremonies*. A *ceremony* that was successfully performed again and again is likely to be triggered and used as a force of habit. In fact, these patterns are usually so fixed and so automatically executed when triggered, that Cialdini [5] illustrated it as if these patterns are pre-recorded tapes that are played based on a trigger. A good example is the result of the infamous "crying wolf" syndrome. So many systems have presented useless and misunderstood alerts and error messages, causing users to develop 'ignore and click-through' habits.

4.1 Evaluation Based on Usable-Security Principles

All four implementations have multiple obvious and critical usability issues, as we next discuss. Such critical issues cause us to seriously doubt the reliability and effectiveness of these mechanisms, and, in fact, to doubt that all four applications have applied minimal usability experiments to their implementations. This is esp. significant, considering that IM applications are designed to be extremely usable. Following the usable security principles which were identitifed and discussed in [18] and also in [28], we present our evaluation findings in the context of these principles:

Users must understand which security actions are required of them. In WhatsApp and Telegram, the message which is presented to users imply that opportunistic mode is enough and no further action is required. Moreover, in Telegram, new chats are not end-to-end encrypted by default, and user are expected to understand that they need to actively choose 'secret chat' for end-to-end encryption. Even users that will notice Viber's lock icon, cannot deduce from it that further actions are required and which. Once users do start the authentication ceremony, WhatsApp, Telegram and Viber provide a short explanation of what needs to be done. Sadly, this explanation is problematic, because users often ignore such information, especially if it contains "technical" keywords (e.g., authentication). In Signal, the authentication screen displays which key needs to be read by which user, but no further explanation exists.

The mental and physical load of a security action must be tolerable, including when performed repeatedly. Although tolerance is subjective, performing current authentication ceremonies requires too much effort from users, and therefore, bound to fail. Even in most optimistic scenarios, comparing long keys is both tedious and error prone, and bound to be in-feasible for repeated use. Telegram's image comparing is simpler than comparing characters, but much more problematic, because it is fair to assume that users will not compare "square by square"; therefore, this is much more prone to mistakes. Viber's built-in call mechanism offers a remote channel where users can authenticate one another, but still requires long key comparison.

The system must provide the user with sufficient information for deriving the security conclusion. In WhatsApp, Telegram and Signal, when users perform manual authentication, they have no indication that the ceremony was completed successfully. In a way, it as if they claim that it can never fail, i.e., can never show unmatched keys. But, what if it does happen? In what way clients are expected to behave as a result of a fail authentication? This principle is especially problematic when considering key reset scenarios. Not reflecting it to users (WhatsApp's default behavior) or reflect it (too) unobtrusively (Viber's color change icon) will result in users who do not understand that they need to perform authentication again.

The mental load of deriving the security conclusion must be tolerable. While Signal's obtrusive approach for key reset seems wise from security perspective, it inevitably brings up the original problem of whether 'normal' users can comprehend the cryptographic explanations and illustration, and not just "click through" the notification (due to intolerance).

The unmotivated user property. The primary goals of users are to send messages, share images etc., and they expect security "to be there" to protect them. Therefore, users postpone (sometimes indefinitely) security related tasks, in favor of their primary goals. Therefore, in security and privacy related mechanisms, where users do not see an immediate consequence or benefit (esp. in tiresome processes), they tend to find it annoying or useless, and dismiss the need to perform the process in the future. When users perform manual authentication and the application does not reflect its results (e.g. in authenticated mode), users are likely to ignore authentication process in the future. In WhatsApp, Telegram and Signal, the application is not aware if authentication was performed (because users manually compare keys), therefore, they do not reflect it to the user. As a result, users are not feeling "rewarded" for their efforts.

Unrealistic assumptions. Scanning QR codes, or comparing visual images, requires clients to be physically next to each other or to have an authenticated off-band channel where they can send the code/image to one another. In all four applications, users may try to perform authentication also remotely, by comparing the digits (in Viber, this is the

only option). This requires an authenticated means of communicating the code; this may not be a viable assumption.

4.2 Hypotheses

The fact that existing mechanisms use cryptography, does not mean that users need to understand cryptography. The currently used approaches and mechanisms are targeted and designed for normal users, therefore, need to be usable by this exact audience. Therefore, we devised a set of hypotheses aimed to examine the awareness and thoughts of users in regard to end-to-end encryption in IM applications in general, and how they feel about the usability of WhatsApp, Viber, Telegram and Signal implementations in particular.

HYPOTHESIS 1 (MINDSET). *Many users care about the privacy of their messages, incl. privacy from the operator.*

Reasoning. CIGI-Ipsos global survey on Internet security and trust [6] has shown that users are indeed interested in achieving good security guarantees. As an example, 83% of the participants changed their online activity in order to control their online shared information. Another possible cause might be the result of Snowden's leaks [4]. On top of that, the fact that operators themselves include an integrated mechanisms indicates that operators believe that their clients will want to use it.

HYPOTHESIS 2 (AWARENESS). *Users of IM applications are not aware of the need to authenticate.*

Reasoning. Current interfaces do not include any clear indication that urges users to perform authentication. On the contrary, some existing indications of security and also current messages displayed to users cause a false sense of security.

HYPOTHESIS 3 (SIMPLICITY). *Many users will fail to authenticate (on their own).*

Reasoning. There are no clear and straightforward indications of the need in such operations. The existing indications (or their lack of) are creating the (false) assumption that no further action is required. It is more than likely that users will "take shortcuts" which collapses the entire mechanism.

HYPOTHESIS 4 (TOLERANCE). *Many users will not authenticate properly in regular use (even after being shown how).*

Reasoning. Completing the task is cumbersome, tedious and time consuming. Comparing dozens of characters or detailed images are not a feasible operation for users, and is especially error prone which will only lead to discomfort. Also, instead of 'one click operation' users are required to click their way through several screens and operations.

HYPOTHESIS 5 (RELIABILITY). *Many users will fail to detect change of keys (need to re-authenticate).*

Reasoning. In all evaluated applications, the indications of change in a key - requiring re-authentication - is hardly visible, non-intuitive or simply not enabled by default. Relying on users to distinguish a change in colors of small icons to conclude that authenticity is broken or for them to actively enable alerts of such scenarios is problematic.

5. EMPIRICAL EVALUATION

In order to test our hypotheses, we conducted two separate experiments. The first experiment involved WhatsApp, Viber and Telegram, with variety of participants from different ages and backgrounds. After the conclusive results of the first experiment, demonstrating that authentication ceremonies are indeed unusable, we wanted to test whether more experienced ('heavy') users in IM applications would yield different results. In other words, we examined if IM oriented users would be more tolerant and successful with current implementations. We conducted a separate second experiment with experienced users, which are very active in IM applications on a daily basis. The second experiment included all four applications (WhatsApp, Telegram, Viber and Signal), and although users displayed slightly better results from the first experiment, the overall findings corroborated most of the findings of the first experiment.

5.1 Participants and Ethics

Experiments were done on smartphones provided by us, and did not use the smartphones of the users or any private information of the users. All the participants participated fully voluntarily, and the goals of the experiments were clearly articulated to the participants; in fact, after completing the experiment, the majority of participants expressed interest in the experiment and satisfaction with their participation. We have obtained approval from our faculty's ethical review board.

The first experiment included 23 participants (figure 4a), from a variety of ages (figure 4d) and backgrounds (figure 4b). 13% of the participants defined themselves as people who send/receive an average of 10-20 messages a week, 65.2% defined themselves as people who send/receive an average of 10-100 messages a day and the rest as people who send/receive more than 100 messages a day (figure 4c).

The second experiment included 16 participants, in the age range of 16-22. All participants were without background in the field of secure communication. All participants (but one) defined themselves as people who send/receive more than 100 messages a day and well familiar with at least one of the evaluated applications.

5.2 Methodology

Both of the experiments were conducted similarly. The first experiment included WhatsApp, Telegram and Viber, and the second experiment was conducted with all four applications (WhatsApp, Viber, Telegram and Signal).

Each experiment was divided into two parts.

First part. In the first part, we evaluated the prior knowledge and thoughts that people have in regard to end-to-end encryption, and examined whether people will be able to handle current encryption and authentication mechanisms. Most importantly, we wanted to evaluate how people will react to end-to-end encryption in general, and how they will handle the authentication process. In order to do so, we have conducted the following steps:

Introduction. Participants were interviewed about their knowledge on security and privacy in messaging services, and how they feel about it. During this part users were introduced to the concept and goals of end-to-end encryption. We emphasized to users that in end-to-end encryption*no one* can eavesdrop to their conversations, including the operator, and that only the sender and receiver can read the messages

content.

Active experimentation. First, after explaining the concept of end-to-end encryption, users were asked to send an end-to-end encrypted message in each application. We emphasized to participants that they should only declare task completion when they think the message they sent was end-to-end encrypted, and no one other than endpoint users can decrypt the message (in particular, that the provider cannot obtain the message). We then explained to each participant about authentication and its importance, and why it needs to be performed in order to ensure end-to-end encryption, and informed them that all applications have a built-in authentication mechanisms. Participants were instructed to reproduce the previous task, but (if they have not done so in the first attempt) use the built-in authentication mechanism. (In reality, most participants did not performed authentication before we explained the need.)

Survey. Participants were asked a series of further questions, designed to understand what they've experienced and what they think about the entire process.

Second part. In the second part, we wanted to examine whether people are able to learn current end-to-end encryption mechanisms, and whether they be able to detect authentication related problems, such as key reset (and loss of authentication). We therefore conducted the second part only after a significant 'break' from the first part (of few hours or days). The steps of the second part are:

Preparation: change key. We re-installed the applications, causing them to re-negotiate a key (without authentication - in 'opportunistic' mode).

Active experimentation. We asked the participant to send an end-to-end encrypted message in all applications.

Survey. Participants were asked a series of questions designed to understand whether they noticed that the authentication was broken, and why they performed authentication or did not perform authentication.

5.3 Results

Hypothesis 1: Many users care about the privacy of their messages, incl. privacy from the operator. We asked participants to state how much they care about the privacy of their messages (figure 4e). In both studies, none of the participants stated that they do not care about messages privacy. Roughly, half of the participants stated that they care in specific cases, and the other half stated they care most of the time. When asked whether they are familiar with the definition of end-to-end encryption (figure 4f), most participants were not familiar with what it means. All participants received an explanation on what end-to-end encryption is, and what are its goals, and then asked whether they would like their messaging application to support such feature. 13% of the participants in the first experiment stated they occasionally would. All other participants stated they would always want this feature. Interestingly, these results brings up the 'privacy paradox' [3]. In the paradox, people tend to express concerns about their privacy, yet their actions and behavior compromises their privacy. In our experiments, although users initially expressed clear favor towards privacy, once they understood the price it entails, they were not willing to pay the price, as depicted later on.

Hypothesis 2: Users of IM applications are not aware of the need to authenticate. We instructed participants to send an end-to-end encrypted message in each application (in a random order) and to declare when they finish. We emphasized that when they complete the task, they consider the message sent end-to-end encrypted and that only them and the recipient can read the message. In the first experiment, only one participant successfully performed authentication (using WhatsApp); however, she failed to do so in other applications. All other participants sent messages without authenticating the identity of the recipient. Moreover, 65.2% have sent regular client-server encrypted messages in Telegram, and not end-to-end encrypted messages (figure 4g). In the second experiment, 56.25% did not performed authentication at all, 18.75% performed in only one application and the remaining 25% authenticated in two or three applications.

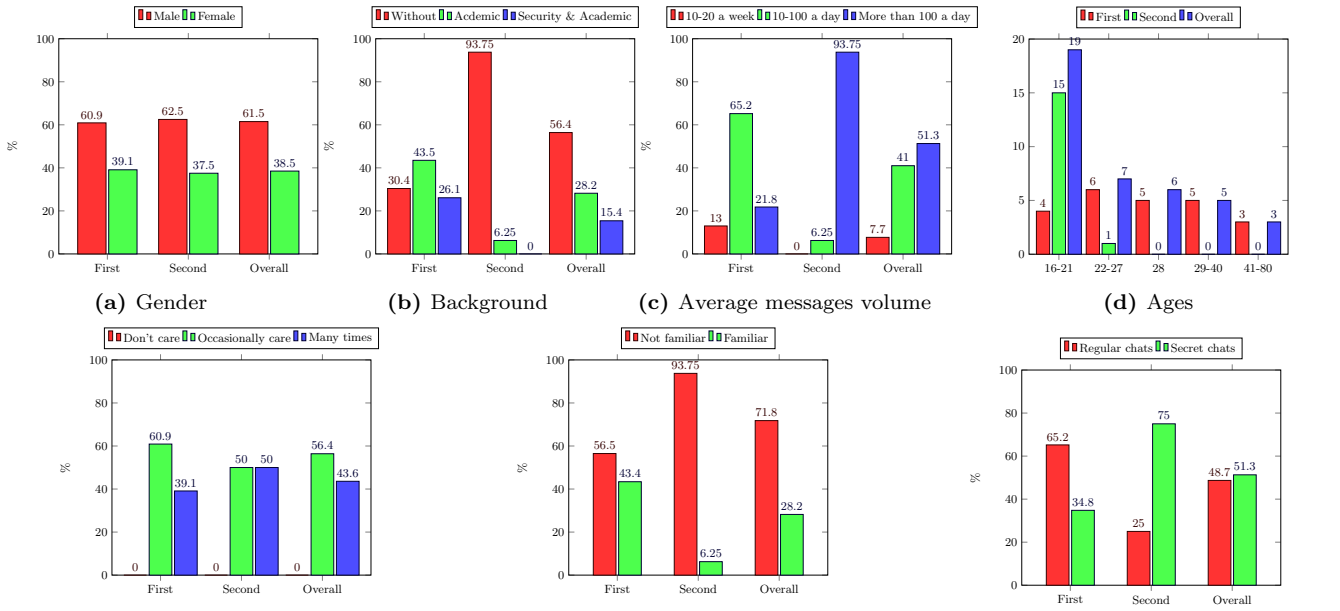
Hypothesis 3: Many users will fail to authenticate (on their own). In the first experiment, only 13% were able to locate and operate the mechanism in all three applications. 56.5% failed to complete the task in all three applications. In the second experiment, the experienced users failed to authenticate in 34.4% of the attempts. Overall, 43.6% described the authentication process (in general) as complicated, cumbersome or non-intuitive.

Hypothesis 4: Many users will not authenticate properly in regular use (even after being shown how). When asked whether or not they will use these mechanisms in the future, 39.1% in the first experiment and 62.5% in the second stated they will not use them in the future. Yet, we consider these results to be overly optimistic, and the actual use rates to be much lower; this is indicated by the responses of users when we asked them for their opinion of usage by other users, where they were significantly less optimistic about the usage.

Hypothesis 5: Many users will fail to detect change of keys (need to re-authenticate). After completing the first part of the experiment, we performed key reset and asked participants to send end-to-end encrypted message. 73.9% in the first experiment and 75% in the second, indeed re-performed the authentication process. However, only 8.7% in the first experiment and 12.5% in the second, mentioned to notice that something was changed (Viber's colored lock); the rest stated that they only performed authentication because of the first part of the experiment, and did not notice anything. In addition, in the second experiment, only 31.25% noted that they performed authentication in Signal because of the alert message that Signal displayed. In our experiment, we performed key reset to all participants. It may be prudent in future work to split participants into two groups in this part, performing key reset on only one group. Overall, the average time it took to successfully perform authentication was 70 seconds.

5.4 Qualitative Feedbacks and Observations

General observations. Many users found the experiment very interesting and insightful. Users noted that even though they were not closely familiar with end-to-end encryption and its importance, the experience gave them vital information to think of in future messaging (P23: "Now that I'm aware, I might use it in the future", P32: "Interesting, but a bit stressful that our messages are not fully end-to-end encrypted "). It shows that users are indeed willing to take better care of their privacy if only systems will allow them to *easily* do so. Some users concluded that the experiment



(a) Gender **(b)** Background **(c)** Average messages volume **(d)** Ages

(e) Messages privacy **(f)** Familiarity with end-to-end encryption

(g) How many users chose 'regular chats' vs how many chose 'secret chats' in Telegram for end-to-end encrypted messages?

Figure 4: Participants demography and studies results

caused them to reduce their trust in operators.

It is interesting to notice that even though people were instructed to use end-to-end encryption and even to authenticate, they struggled in completing the task. It is likely to assume that in non-lab environments, where users tend to be even less focused on security (see "The unmotivated user property" in 4.1), they are likely to produce even worse results. Furthermore, even though users were asked to perform authentication only to a single conversation, during authentication, several users tried to take *shortcuts*, and compared only samples of the code (or of the image).

Usability observations. Many of those who failed to authenticate, did stumble upon one of the authentication screens, but could not figure out what it meant. Some of the users that did understood how to use it, were not sure about how they are supposed to act if they are not physically near one another (P39: "Whether you are familiar with it or not, it is very difficult to locate", P33: "It was frustrating and unreasonable"). A few users attempted to send the code through the chat itself, or through some form of an off-band channel (like mail).

The presence of lock icons has caused many users to (falsely) feel secure (P20: "I thought that lock icon means security"). Users declared (incorrectly) that they have completed the authentication task successfully, because they misinterpreted the lock icons they came across with during the task. Users excessively searched for some kind of a feature they can enable in the general menu screens, and almost did not check for the menus in the chats themselves. Users expected it to be a mechanism that they will enable and it will automatically work on all chats.

Users indicated that for short, occasional chats, or rather chats with unimportant contacts, they would not bother authenticating contact's identity. They also mentioned that even if they completely adopt and behave as they are expected, they had to undergo a guided experience in the experiment to acquire the proper understanding of end-to-end encryption; how would people that did not do the experiment be able to participate in authentication process?

Accessibility. Telegram offers a visual representation of squared colors (see figure 2c), that users can compare with one another instead of the long sequence of digits. Color-blinded and elderly participants indicated that they were not able to use the image visualization with confidence and had to resort to code comparison. This was also mentioned in Viber's lock icon display, which relies on different colors to indicate different states. They indicated discomfort when asked to classify the current color of the icon.

WhatsApp observations. WhatsApp's QR code was liked by participants, because of it relatively easy to operate requirements. That said, some participants were not sure what needs to be done, or even how to scan the code. Although there is a designated scan button in the bottom of the screen, some users thought a separate QR reader application is needed.

A rather insignificant bug caused significant stress among few participants. WhatsApp's authentication screen displayed the authentication code differently when in RTL or LTR mode. The code was displayed in a reverse order (see figure 5). As a result, when users attempted to compare authentication codes - they (allegedly) had different codes. Users were unable to deduce that the problem is with the display and not with the code itself.

Figure 5: WhatsApp's RTL/LTR bug

Viber observations. Viber's built-in call authentication has proven to be very problematic. Even though some users indicated that because the call was 'fused' into the ceremony - they understood how to complete the process, many attempts to establish a call were unsuccessful, and in other cases, application's connectivity and sound quality was very problematic. Moreover, users struggled with locating the lock icon and reported problems working with it. Despite the fact that many users did not detect that the authenticity was broken (in the second part), when asked if they noticed that Viber's lock icon changed colors, many claimed to have noticed it, but ignored it. This only emphasizes that even when people are focused on security related tasks, they still not paying enough attention to security.

Telegram observations. In Telegram, some users indicated that they prefer the image visualization over scanning QR code (in WhatsApp), because for them, looking on an image is easier than scanning a code. By default, users sent messages in Telegram the same way they sent in WhatsApp and Viber. They did not notice or understand the meaning of "secret chat" and that they need to use it in order to be end-to-end encrypted.

Signal observations. Signal's approach of displaying an obtrusive notification on key reset, has proven to have a negative effect more than a positive one. Even though participants encountered it in an advanced stage of the experiment, they could not properly interpret what it meant and did not conclude they should re-perform the authentication ceremony. Furthermore, when specifically asked whether they understood the meaning or reason of this message, they could not supply valid answers (P38: "I thought the message is a result of a connectivity problem", P31: "People will interpret the message as if there is a problem with the service, and will change to another application").

6. PROMISING DIRECTIONS

Presenting complete solutions to the many challenges that were mentioned, is beyond the scope of this paper. Nevertheless, we present several directions that can serve as a base to better and stronger usability of end-to-end encryption in IM applications. We stress that some of these directions still have some major challenges yet to be figured out, and incorporating them *successfully* into modern and popular IM applications, is a work in progress.

6.1 Avoiding Authentication Ceremony

The most repeated comment made by participants is that they envision authentication to be more transparent. Users expect it to be "part of the package" by default, and performed automatically without their active intervention.

A promising direction towards transparent end-to-end encryption in messaging services is CONIKS[19]. CONIKS offers clients the ability to automatically perform authentication - *without* the need to *trust* anyone, esp. not the operators. In CONIKS, if an attacker is attempting to forge cryptographic keys of a user, both the user itself and other users can detect it. Therefore, client applications can make sure the cryptographic keys which are used to secure the communication, are indeed belong to the designated recipient. This eliminates the need to either trust the operator or to perform manual authentication ceremony with the peer.

6.2 Improving Current Implementations

Straightforward security. Applications should not hide features and should not "sugar coat" realities. On the contrary, they need to find more simple and straightforward ways to explain the expected behavior to users. If in order to obtain better security, users are required to perform specific ceremonies (e.g. authentication), they need to be aware of that. If users choose not to authenticate - it is okay, but they have to understand what this means. Lack of clarity only leads to bad reaction by users, when they are confronted with the reality.

Simpler ceremonies. Manual authentication must be simplified. One way of achieving simplicity, can be by presenting more reasonable visualizations of the key, through mapping. For instance, mapping a series of senseless characters to words, sentences or images, can significantly improve the usability of the ceremony. Another approach could be using *gamification* [8], in order to transform the ceremony into a more pleasant procedure. Alternatively, perhaps the entire ceremony needs rethinking. SafeSlinger [13] is an example of a protocol for secure public key exchange between users, with a different ceremonial requirements.

Prioritizing. Users indicated during our studies that they feel differently towards their contacts, and would employ different level of vigilance based on who they are talking with. Therefore, it is logical for the application to behave differently if current contact is more/less important. By asking the user to classify the importance of a contact before the first time they converse, the application can have the ability to notify the user better. If this contact is important, it is reasonable to notify the user on more things and in a more obtrusive way. It only requires the user to classify any contact once, which is reasonably acceptable.

7. CONCLUSIONS

Users of IM applications are - rightfully - concerned about privacy, and appreciate the importance of end-to-end encryption. However, users do not have the expertise to fully understand the differences between different adversary models and 'variants/modes' of end-to-end encryption, and trust the operator to provide this functionality securely and to indicate clearly any limitations.

We evaluated the usability of the end-to-end encryption mechanisms in four popular IM applications: WhatsApp, Viber, Telegram and Signal. Participants in our studies expressed feelings of being misled by the claims of end-to-end encryption by IM application providers, when they realized that to fully protect themselves against a rogue operator, they are supposed to perform the authentication process - which all found inconvenient and many considered completely impractical to use.

We concluded with promising directions both in eliminating the need to actively perform authentication procedures, and in producing system with more usable security. We hope that our work will encourage researchers, as well as developers of IM applications, to seriously investigate and evaluate innovative designs, to provide truly secure end-to-end encryption for users of IM services.

Acknowledgment

This research was supported by grants from the Israeli Ministry of Science and Technology.

References

[1] Scorecard update: We cannot credit skype for end-to-end encryption. https://www.eff.org/deeplinks/2014/11/scorecard-update-we-cannot-credit-skype-end-end-encryption.

[2] H. Assal, S. Hurtado, A. Imran, and S. Chiasson. What's the deal with privacy apps?: a comprehensive exploration of user perception and usability. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, pages 25–36. ACM, 2015.

[3] S. B. Barnes. A privacy paradox: Social networking in the united states. *First Monday*, 11(9), 2006.

[4] P. R. Center. Americans' privacy strategies post-Snowden, 2015. URL http://pewrsr.ch/1LiLHAM.

[5] R. B. Cialdini. *Influence: Science and practice*, volume 4. Pearson Education Boston, MA, 2009.

[6] CIGI-Ipsos. 2016 cigi-ipsos global survey on internet security and trust, 2016. URL https://www.cigionline.org/internet-survey-2016.

[7] S. Clark, T. Goodspeed, P. Metzger, Z. Wasserman, K. Xu, and M. Blaze. Why (special agent) Johnny (still) can't encrypt: A security analysis of the APCO project 25 two-way radio system. In *USENIX Security Symposium*. USENIX Association, 2011. URL http://static.usenix.org/events/sec11/tech/full_papers/Clark.pdf.

[8] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon. Gamification. using game-design elements in non-gaming contexts. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems*, pages 2425–2428. ACM, 2011.

[9] T. Dinev, P. Hart, and M. R. Mullen. Internet privacy concerns and beliefs about government surveillance–an empirical investigation. *The Journal of Strategic Information Systems*, 17(3):214–233, 2008.

[10] R. Dingledine and N. Mathewson. Anonymity loves company: Usability and the network effect. In *WEIS*, 2006. URL http://weis2006.econinfosec.org/docs/41.pdf.

[11] V. Dukhovni. Opportunistic security: Some protection most of the time. Technical report, 2014.

[12] A. Elliott and S. Brody. New yorkers on mobile messaging and implications for privacy. Technical report. URL https://simplysecure.org/resources/techreports/NYC15-MobMsg.pdf.

[13] M. Farb, Y.-H. Lin, T. H.-J. Kim, J. McCune, and A. Perrig. Safeslinger: easy-to-use and secure public-key exchange. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 417–428. ACM, 2013.

[14] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk, and T. Holz. How secure is textsecure? In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 457–472, 2016. . URL http://dx.doi.org/10.1109/EuroSP.2016.41.

[15] S. L. Garfinkel and R. C. Miller. Johnny 2: a user test of key continuity management with S/MIME and Outlook express. In *Proceedings of the 2005 symposium on Usable privacy and security*, pages 13–24. ACM, 2005.

[16] P. Gutmann and I. Grigg. Security usability. *IEEE security & privacy*, 3(4):56–58, 2005.

[17] A. Herzberg and A. Jbara. Security and identification indicators for browsers against spoofing and phishing attacks. *ACM Transactions on Internet Technology (TOIT)*, 8(4):16, 2008.

[18] A. Jøsang, M. A. Zomai, and S. Suriadi. Usability and privacy in identity management architectures. In *Proceedings of the fifth Australasian symposium on ACSW frontiers-Volume 68*, pages 143–152. Australian Computer Society, Inc., 2007.

[19] M. S. Melara, A. Blankstein, J. Bonneau, M. J. Freedman, and E. W. Felten. Coniks: A privacy-preserving consistent key service for secure end-to-end communication. *IACR Cryptology ePrint Archive*, 2014:1004, 2014.

[20] H. Orman. Why won't Johnny encrypt? *IEEE Internet Computing*, 19(1):90–94, 2015. URL http://dx.doi.org/10.1109/MIC.2015.16;http://doi.ieeecomputersociety.org/10.1109/MIC.2015.16.

[21] S. Ruoti, N. Kim, B. Burgon, T. Van Der Horst, and K. Seamons. Confused Johnny: when automatic encryption leads to confusion and mistakes. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, page 5. ACM, 2013.

[22] S. Ruoti, J. Andersen, D. Zappala, and K. E. Seamons. Why Johnny still, still can't encrypt: Evaluating the usability of a modern PGP client. *CoRR*, abs/1510.08555, 2015. URL http://arxiv.org/abs/1510.08555.

[23] M. Sasse. Has Johnny learnt to encrypt by now? examining the troubled relationship between a security solution and its users. In *5th Annual PKI R&D Workshop (2006)*, 2006.

[24] S. Schechter, S. Egelman, and R. W. Reeder. It's not what you know, but who you know: a social approach to last-resort authentication. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1983–1992. ACM, 2009.

[25] S. Schrittwieser, P. Frühwirt, P. Kieseberg, M. Leithner, M. Mulazzani, M. Huber, and E. R. Weippl. Guess who's texting you? Evaluating the security of smartphone messaging applications. In *NDSS*, 2012.

[26] S. Schröder, M. Huber, D. Wind, and C. Rottermanner. When signal hits the fan: On the usability and security of state-of-the-art secure mobile messaging. 2016.

[27] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith. SoK: Secure messaging. In *IEEE Symposium on Security and Privacy*, pages 232–249. IEEE Computer Society, 2015. ISBN 978-1-4673-6949-7. URL http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7160813.

[28] A. Whitten and J. Tygar. Usability of security: A case study. Technical report, DTIC Document, 1998.

[29] A. Whitten and J. D. Tygar. Why Johnny can't encrypt; A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium (SECURITY-99)*, pages 169–184, Berkely, CA, Aug. 23–26 1999. Usenix Association.

APPENDIX

A. WHY THESE SERVICES?

In this paper, we are focusing on four popular IM applications: WhatsApp, Viber, Telegram and Signal. We did not include other services because of several reasons. The first reason is popularity and user adoption. This work focuses on popular services which are used by billions, in order to evaluate their real-world usability. While some systems might be offering other approaches - they have not been adopted by many users, and therefore are unfit for this paper. Other services (e.g., anonymity oriented services) not only do not have large user base, but usually do not have implementations for mobile environments. Many of these services suffer from bigger issues due to their computation and bandwidth requirements, which are not suitable for mobile environments, esp. in regard to usability.

We also did not include popular services that share (too) many common features as one of the applications we already evaluated. For example, Facebook Messenger has recently integrated end-to-end encryption also based on Signal's protocol, and users can enable end-to-end encryption functionality from the application's menu. After that, users still need to actively create end-to-end encrypted chats (like in Telegram) and perform authentication ceremony (which is similar to Signal's ceremony). Google also launched an IM service called Allo, where end-to-end encryption is also not enabled by default, and, at least for now, there are no special approaches which differ from already discussed applications. Some popular services, like Skype, were not evaluated, because they are very vague [1] about their end-to-end encryption support.