



Automating Key Fingerprint Comparisons in Secure Mobile Messaging Apps: A Case Study of Signal

Mashari Alatawi
Texas A&M University
College Station, Texas, USA
mashari@tamu.edu

Nitesh Saxena
Texas A&M University
College Station, Texas, USA
nsaxena@tamu.edu

ABSTRACT

End-to-end encryption (E2EE) is a critical security feature in mobile messaging apps like Signal, WhatsApp, and Skype, protecting private conversations from unauthorized access, even by service providers. However, this security heavily relies on users participating in an *authentication ceremony* to verify encryption key fingerprints, thwarting potential man-in-the-middle (MitM) attacks. This authentication ceremony involves comparing QR codes or readable/exchangeable codes (e.g., numeric or hexadecimal) to ensure a match. Failure to match these codes due to human errors or click-through behavior makes users vulnerable to MitM attacks, as recent research highlights users' inability to perform this crucial comparison task in current E2EE applications.

This paper presents an innovative automated approach integrated into the Signal application to simplify the authentication ceremony by automating the safety number comparison task. The new approach streamlines the comparison process by removing the manual burden from users. It prompts users to share the safety number through two out-of-band (OOB) channels: an SMS message and orally during a phone call with the other party. Simultaneously, the approach automatically performs a comparison between the shared safety number and the locally generated safety number on the recipient's phone. This automated comparison ensures accuracy, reduces human error, and enhances security by making it doubly difficult for attackers, as both channels must be compromised simultaneously. The approach demonstrates improved effectiveness, eliminating human errors and resisting MitM attacks, making it more robust than the current implementation in the Signal application. The results show a 0% false acceptance and 0% false rejection rate, significantly enhancing security and usability in this critical authentication process.

CCS CONCEPTS

• Security and privacy;

KEYWORDS

Authentication Ceremony, Automated Comparison, Safety Number

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CODASPY '24, June 19–21, 2024, Porto, Portugal

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0421-5/24/06
<https://doi.org/10.1145/3626232.3653251>

ACM Reference Format:

Mashari Alatawi and Nitesh Saxena. 2024. Automating Key Fingerprint Comparisons in Secure Mobile Messaging Apps: A Case Study of Signal. In *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy (CODASPY '24)*, June 19–21, 2024, Porto, Portugal. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3626232.3653251>

1 INTRODUCTION

People are increasingly turning to secure messaging applications such as Signal [30], WhatsApp [38], Skype [31], Viber [36], Telegram [33], and Facebook Messenger [11] to provide end-to-end encryption (E2EE) for their private audio, video, and text communications. In order to provide the E2EE feature in these mobile applications, all messages and calls between two or more parties are encrypted using a cryptographic key. The parties can agree on this key by using a key exchange protocol [2, 12, 41]. This protocol often generates key fingerprints, which are unique identifiers for each key. The parties involved in the communication can verify that they are using the same key by comparing their key fingerprints. This can be done by participating in an optional task called an *authentication ceremony*. This task requires the user to scan a QR code generated by the peer's application if the two users are in the same physical location. If they are not in close proximity, they can exchange and compare their key fingerprints over any out-of-band (OOB) channel (like a text message, email, or phone call). If the fingerprints do not match, the communication can be canceled or rejected. This helps prevent man-in-the-middle (MitM) attacks, where an attacker intercepts the communication and attempts to substitute their own key [40].

In many E2EE applications, fingerprints are often presented as a hash of a public key encoded in a readable or exchangeable format, such as a numeric or hexadecimal code. This allows end users, even if they are not physically together, to compare and verify their key fingerprints effectively. For instance, the Signal application uses a term called *Safety Number* in its authentication ceremony to refer to the key fingerprint [23]. The safety number in the Signal app is crucial for verifying conversational partners' identities. It is generated from users' public keys, phone numbers, and initial exchanged messages. Comparing these safety numbers ensures protection against MitM attacks. During the authentication ceremony, the Signal app allows users to compare key fingerprints either by scanning QR codes or by comparing a 60-digit numeric string divided into 12 blocks. This string changes whenever a user reinstalls the application or switches devices due to a new key generation. Notably, the new key generation might also occur due to a MitM attack, where an adversary substitutes a fake key for an existing one. Typically, secure messaging applications implement the E2EE feature in an opportunistic E2EE mode, which involves establishing

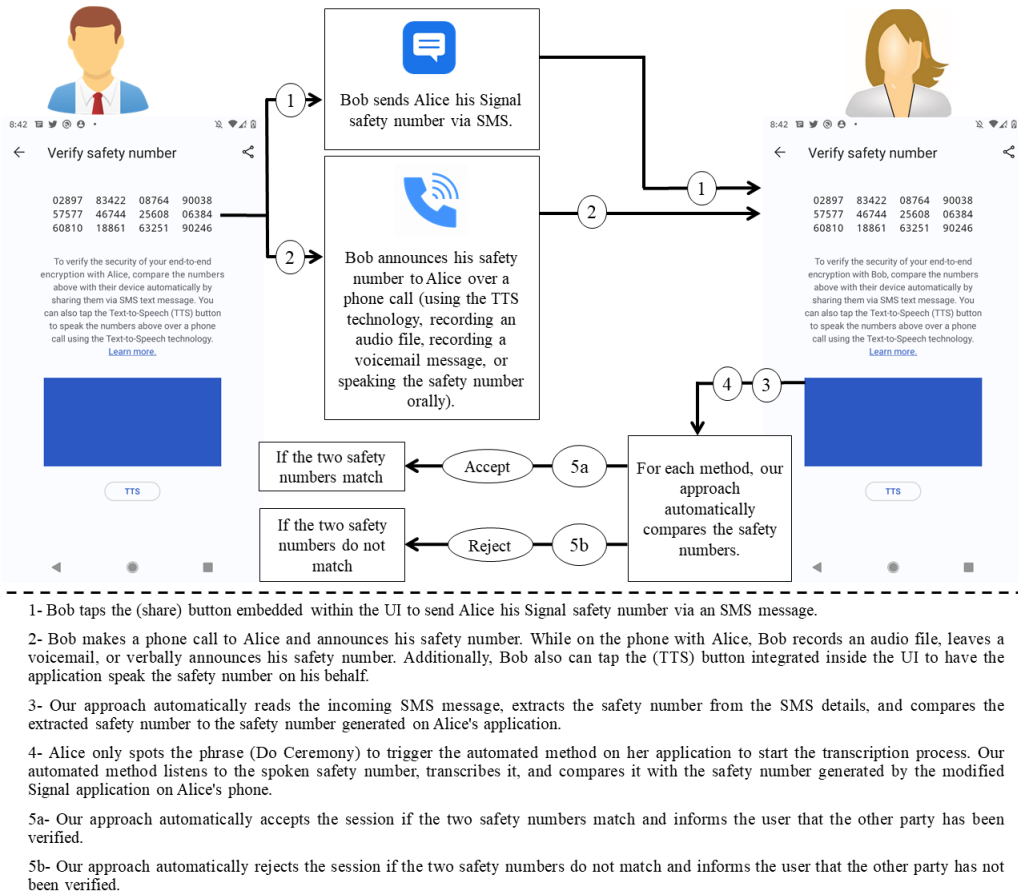


Figure 1: Our automated approach of comparing safety numbers through both SMS and voice channels during the underlying authentication ceremony of Signal

a secure connection between two parties without authenticating the other [19]. This mode ensures confidentiality only if users trust the service providers. While this mode prevents passive MitM attacks, it leaves the system vulnerable to active MitM attacks, such as those launched by the service provider acting as a rogue server or by compromising the server itself [13, 39]. Therefore, E2EE applications rely on users to compare and verify their key fingerprints in order to enable the authenticated E2EE mode and provide security against active MitM attacks.

However, research studies that investigated the authentication ceremony revealed that users performed poorly across the majority of key verification procedures [13, 25, 27, 29, 35]. They have shown that users are especially vulnerable to MitM attacks due to a combination of human errors and usability problems. For example, when users are required to compare key fingerprints to ensure that they are communicating securely, they may make errors such as not paying close enough attention to ensure that the key fingerprints match. As a result, some mismatches are detected as matches, and some matches are detected as mismatches, making it difficult for users to determine whether their communication is secure. These errors and usability problems can be compounded by other factors such as time pressure, fatigue, or distractions, further increasing

the likelihood of successful MitM attacks. To mitigate this vulnerability, it is essential that developers prioritize user education and design more intuitive security features that minimize the potential for errors and make it easier for users to verify the authenticity of their mobile communications.

In this paper, we introduce an automated approach for comparing safety numbers within the underlying authentication ceremony of the Signal application [30]. Our approach leverages two OOB channels, namely the short message service (SMS) and voice channel, to motivate users to engage in the authentication process. By automating the safety number comparison task, our aim is to simplify and streamline the performance of the authentication ceremony. The significant issue currently prevalent in all E2EE apps, including Signal [30], is the necessity for users to manually compare and verify their fingerprints to transition into the authenticated E2EE mode [1]. This manual comparison task is exposed to potential human errors or click-through behavior during the authentication ceremony, consequently resulting in susceptibility to MitM attacks [13, 25, 27, 29, 35]. Given this significant concern, our proposed solution involves the complete automation of the fingerprint comparison task within the authentication ceremony, effectively

eradicating the possibility of human errors during fingerprint verification. To achieve this, we have adopted OOB channels to enable users to seamlessly share or transmit their fingerprints, particularly in remote scenarios, while the comparison procedure is done automatically within the application. We aim to examine whether or not it is feasible to simplify and fully automate the process of comparing safety numbers during the authentication ceremony. Our automated approach leverages the *BroadcastReceiver* class that is available in the Android Studio in order to receive messages sent over the SMS channel [8]. It listens for an incoming SMS message and automatically extracts the safety number from the SMS details. It then performs the automated comparison on behalf of the user. Our automated approach also utilizes the Speech-to-Text (STT) technology integrated into the Android system to automatically transcribe and compare the safety number over a phone call [7]. Our automated approach of comparing safety numbers through both an SMS channel and a voice channel can add an extra layer of security to the authentication ceremony, helping to protect users' private data and mobile communications from potential threats. The implementation of our automated approach, utilizing two OOB channels simultaneously, will increase the complexity of an attacker's mission, as they will now need to successfully conduct a MitM attack on both channels concurrently. Our automated approach demonstrated excellent effectiveness and feasibility, with a 0% false acceptance and 0% false rejection rate. This automated approach significantly improves the security and usability of E2EE applications, making them more robust against MitM attacks and helping to ensure that private conversations remain private. In Figure 1, our automated approach for comparing safety numbers during Signal's authentication ceremony is depicted using both SMS and voice channels.

Our Contributions: Our contributions are as follows:

- **Automated Comparison Tool Using Two OOB Channels:** We present a novel and effective automated comparison tool that can be used to eliminate human errors associated with the task of comparing key fingerprints in real-world secure mobile applications during their underlying authentication ceremonies. Our approach encourages users to perform the authentication ceremony by automating the comparison of key fingerprints. To facilitate the sharing of key fingerprints with the other party, we utilize two OOB channels. This includes sharing the key fingerprint through SMS or by orally announcing it during a phone call. Since comparing key fingerprints is performed manually in the current authentication ceremonies for E2EE applications, we believe that converting this manual task into an automated process could be valuable to the performing authentication ceremony and could thwart any potential MitM attacks.
- **Design and Implementation of the Automated Comparison Tool:** We present our automated comparison tool designed for safety number comparison during the Signal application's authentication ceremony. Through modifications to the app's current version and a redesigned authentication ceremony, our tool takes advantage of automatic SMS reading and speech-to-text (STT) technology over phone

calls. Our design and implementation details are presented in Section 3.

- **Evaluating the Effectiveness and Feasibility of Integrating the Automated Comparison Tool into the Signal Application:** We integrate our automated comparison tool into the Signal application to evaluate its effectiveness and feasibility in comparing safety numbers during the authentication ceremony. We replace the current methods with our automated comparison tool. Our results show that our automated comparison tool can be implemented in a real-world secure mobile application (like Signal [30]) to automatically perform the comparison task in the authentication ceremony on behalf of the user. We also demonstrate that our automated comparison tool successfully distinguishes matching safety numbers (benign cases) from non-matching ones (malicious or attacking cases). Our evaluation of our automated approach is presented in Section 4.
- **Evaluating the Automated Comparison Tool Against MitM Attacks:** We assess our automated comparison tool's resistance to MitM attacks by creating scenarios leading to mismatched safety numbers. Our findings reveal the tool's ability to detect mismatches and thwart MitM attacks. Also, compared to manual methods, our tool eliminates the risk of MitM attacks due to human errors or click-through behavior during the authentication ceremony. The details of this part of the work are described in Section 4.

2 BACKGROUND

The Signal Protocol, established in 2013 by Open Whisper Systems, is a popular E2EE protocol designed to secure online communications using sophisticated cryptographic methods [4, 12]. Adopted by many E2EE applications, including the Signal app [30], this protocol generates key fingerprints from public keys to prevent MitM attacks. These fingerprints serve as unique identifiers for user public keys, enabling identity verification through an authentication ceremony. During this authentication ceremony, users compare their key fingerprints and are required to cease communication if the fingerprints do not match. The use of an authentication ceremony is a common strategy in numerous E2EE applications to counteract MitM attacks. In the Signal app's current version, the safety number acts as a representation of the key fingerprint, ensuring conversation security and thwarting MitM attacks [23]. The safety number, unlike real-world fingerprints, is not confidential and can be freely shared on public platforms, easing participation in the authentication ceremony. A match confirms no MitM attack, validating the communication partner's identity. Conversely, mismatched safety numbers indicate a potential MitM attack.

However, the authentication ceremony is still performed manually in all E2EE applications, including the Signal application [1]. The Signal application provides its users with two different methods to compare and verify their safety numbers [23]. First, users can scan a QR code that encodes the safety number but they must be in the same physical place. In case users are not located in close proximity to one another, they can manually compare and verify their safety numbers over any OOB channel. In order to compare such a safety number, the authentication ceremony requires users

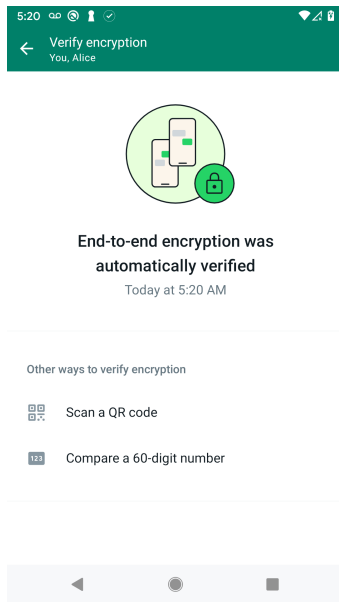


Figure 2: Authentication ceremony in WhatsApp

to manually compare the 60-digit numeric string either in person or remotely. Performing such an authentication ceremony as a manual task is susceptible to human errors, which could lead to MitM attacks. It is also possible that users will *click-through* (sometimes referred to as *skip-through*) as observed in previous research on authentication ceremonies in E2EE applications [29], meaning they will accept the comparison task without proper attention or completion of the task. Moreover, human errors in benign cases, such as rejecting matching safety numbers, may have a negative impact on the usability of Signal communications. This is due to legitimate sessions often being rejected because of human errors, leading to the need for re-establishment.

2.1 Related Work

Key Fingerprint Representations: Two research studies have delved into and compared the usability of different key fingerprint representations. Initially, Dechand et al. [6] scrutinized six textual key fingerprint representation schemes. Their findings revealed that users are more susceptible to attacks when employing hexadecimal, alphanumeric, or numeric representations, as opposed to a series of words or sentences. Subsequently, Tan et al. [32] scrutinized eight representations of textual and visual fingerprints. Their investigation revealed that users are similarly exposed to vulnerabilities when utilizing hexadecimal and numeric representations, in contrast to employing alternative textual representations like words and sentences. Notably, users adopted diverse strategies for comparison, often comparing subsets from the start, middle, and end of fingerprints or selecting random portions of fingerprints. Despite the results of these studies [6, 32], the prevalent trend persists in most E2EE apps, where hexadecimal or numeric key fingerprint representations are favored, with users manually tasked to perform the comparison. Alatawi and Saxena [1] scrutinized

popular E2EE apps and discovered that around 75% of these apps utilized hexadecimal, alphanumeric, or numeric representations for their key fingerprints. Consequently, our proposed automated approach stands as a viable solution to streamline and automate the fingerprint comparison task, effectively removing the human user from the process.

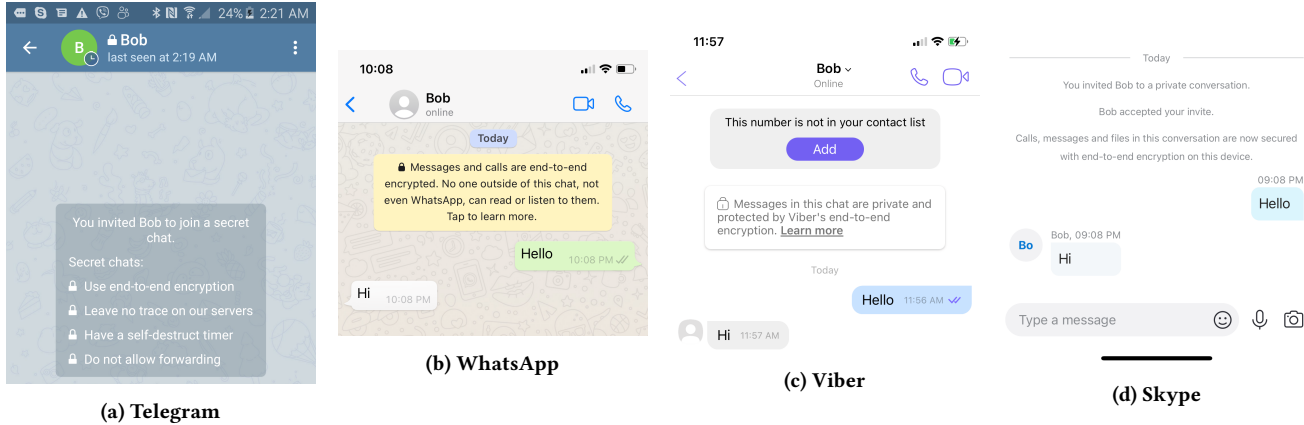
Furthermore, Dechand et al. [6] explained that motivating users was beyond the scope of their study, emphasizing their commitment to finding the optimal comparison of key fingerprints within a security context. One of the primary hindrances in the current authentication ceremony within real-world E2EE apps lies in the manual comparison of key fingerprint representations. This reliance on manual verification leads to potential MitM attacks due to human errors, a concern highlighted by previous research studies [13, 25, 27, 29, 35]. In contrast, our proposed automated approach offers an incentive for users to engage in the authentication ceremony, as the burden of the fingerprint comparison task is eliminated, thereby reducing the risk of errors and subsequent attacks.

Automated Authentication Ceremony: Several recent works attempted to fully or partially automate the authentication ceremony in E2EE applications to reduce the effort required by the user. First, Vaziripour et al. [34] partially automated the authentication ceremony using social media accounts, similar to the Keybase service [17]. They suggested automating the authentication ceremony and distributing trust among additional service providers. However, users' lack of trust in social media accounts points to the need for more trustworthy third-party actors. It is essential to note that users must have social media accounts for this approach. Our work completely automates the authentication ceremony that takes place on the receiver's side, and we do not ask the receiver to perform any additional activities at any point during the authentication ceremony. We developed our automated approach using Android's built-in SMS reading and STT technology, without relying on any third-party services.

Second, Shirvanian and Saxena [28] proposed an approach that makes use of speech transcription technology as a means of enhancing the level of safety offered by Crypto Phones. In their work [28], the authors designed and implemented a novel method for eliminating human users from the loop for checksum comparison called *Closed Captioning Crypto Phones* (CCCP). Using standard STT engines, they created a partially automated fingerprint comparison tool. However, CCCP requires the users to manually run the protocol and only speak the fingerprint verbally over a voice call. In our work, we also streamline the process of comparing safety numbers by automating it, eliminating the need for the user to manually compare numbers during the authentication ceremony in the Signal application. In contrast to the CCCP system, our automated approach involves utilizing the voice channel and requires the user to either verbally state the safety number or tap the text-to-speech (TTS) button, which triggers the TTS engine to speak the safety number on the user's behalf. This may help with announcing a long code, such as the safety number in the Signal application, to enhance the usability of the system. In the CCCP system, the user is only required to verbally recite a Short Authentication String (SAS), which is suitable for short codes but not for long codes. Furthermore, we use not only the voice channel but also the SMS channel to transmit the safety number. Unlike CCCP, we demonstrate the

Table 1: Comparison between the Current Signal Application [30], CCCP [28], and Our Automated Approach

	The Current Version of Signal in the Remote Setting	CCCP	Our Automated Approach
Performing the Authentication Ceremony	Optional	Optional	Optional
Speaking the Fingerprint	Orally	Orally	Orally, or Automatically by using TTS technology
Fingerprint Comparison Method in the Authentication Ceremony	Manual	Partially Automated	Fully Automated
Fingerprint Comparison Error Rate	High	Low	None
Fingerprint Size Supported	Short	Short	Long
Integrating with a Real-World Application	N/A	No	Yes, in the Signal application

**Figure 3: Users are informed that the opportunistic E2EE mode is active and their messages are end-to-end encrypted through the use of distinctive indicators like special notification messages and lock icons.**

feasibility of our automated approach by incorporating it into a real-world E2EE application like the Signal application [30]. Table 1 summarizes the differences between our automated approach and the CCCP approach [28], as well as the current implementation of the authentication ceremony in the Signal application [30].

Lastly, another approach seeks to automate the authentication ceremony, centering around the concept of using key transparency. CONIKS [24], alongside recent derivative works [3] and [22], outlines a key transparency framework where a service provider links users with their public keys, while providers mutually audit each other to identify discrepancies. In this system, clients verify their service provider's key through transparency logs. The setup assumes multiple independent providers publishing each other's signed tree roots, prompting clients to download and compare these from different sources to check for inconsistencies. However, in the context of real-world E2EE applications, a lack of collaboration among providers exists. WhatsApp [38], notably, has introduced an automated verification system based on key transparency, drawing from CONIKS [24], SEEMless [3], and extensions from recent research called Parakeet [22]. Yet, WhatsApp solely relies on its own servers to manage a directory mapping public keys to user accounts and an external audit record, potentially leaving room for collusion between a malicious server and a third-party auditor. This vulnerability, as acknowledged in WhatsApp's key transparency whitepaper [20], prompts the inclusion of proofs of consistency

for key updates as a supplement, rather than a complete replacement, for manual verification of key fingerprints provided on the encryption page for a contact.

Figure 2 illustrates WhatsApp's provision of manual verification methods such as QR code scanning or comparing a 60-digit number, alongside their automated key transparency system. This system runs discreetly in the background, with users informed only of the verification status. This mode reflects an opportunistic strategy, similar to the opportunistic E2EE mode found in various E2EE applications [1]. Here, E2EE applications establish a secure connection without immediate authentication, expecting users to conduct the authentication ceremony to transition to an authenticated E2EE mode. Figure 3 depicts the opportunistic E2EE mode employed in E2EE apps, using a range of cues, such as special notification messages and lock icons, to signify its activation. Similar to this opportunistic mode, WhatsApp employs a key transparency system where users are notified only of the verification result occurring in the background by WhatsApp servers (see Figure 2). Consequently, users must place trust in WhatsApp servers for the security guarantees to hold, introducing a dependency on server trust. To cater to those seeking alternative verification methods, WhatsApp's whitepaper [20] suggests resorting to traditional manual security code verification to bypass reliance on their servers. In contrast to this third-party reliance, our approach prioritizes authentication originating directly from end-users, excluding involvement from external entities not engaged in private communication. In the

current landscape of E2EE applications, the primary impediment hindering users from completing the authentication ceremony is the manual comparison task. Users often struggle and are susceptible to errors when performing manual comparisons, potentially leading to MitM attacks [13, 25, 27, 29, 35]. Thus, our proposed automated solution entirely removes users from this manual comparison task, automating the fingerprint comparison process on their behalf.

2.2 Threat Model

In this paper, we assume a model similar to the common threat model used by secure messaging applications. In such an E2EE session, Alice and Bob, who are both trusted users, want to start communicating securely through an insecure channel utilizing the Signal application that is pre-installed on their respective phone devices. Therefore, in our threat model, we make the assumption that all end points, such as phone devices and Signal client applications, are trusted and secure for both Alice and Bob. However, we assume that the key will be exchanged over a communication channel that the adversary has complete control over. The adversary in this situation would be able to intercept and alter the key exchange messages by performing a MitM attack. Consequently, the adversary can mount a key substitution attack during the key exchange protocol by interfering with the key exchange protocol [39]. The attacker, for example, can compromise the Signal service provider that maintains the public keys and replace them with fake public keys. The goal of the attack is for the MitM attacker to attempt to engage in impersonation sessions with Alice and Bob. As a consequence of the attack, the generated safety numbers at the two parties do not match. However, the MitM attack will be successful if the users erroneously accept safety numbers that do not match. Detection of such an attack requires the implementation of an authentication ceremony, which can be conducted either in-person or remotely. The current Signal application provides a manual mechanism for users to compare and verify safety numbers, a process susceptible to human errors, particularly in remote settings. Consequently, our proposed automated approach seeks to mitigate these errors linked with fingerprint comparison by automating this task. The proposed automated approach involves utilizing two separate OOB channels to facilitate the exchange of safety numbers in both proximity and remote scenarios. Since the safety number should not be treated as secretive and suspicious, users can freely share it on public platforms accessible to others. This perspective aligns with the general assumption adopted by Signal's developers, encouraging them to exchange safety numbers via such OOB channels. The Signal application streamlines this process using a share button within the application, enabling the direct transfer of safety numbers.

In our work, we have developed a modified version of the Signal application that integrates an authentication ceremony for Alice and Bob, specifically aimed at countering MitM attacks following the completion of the key exchange protocol. This protocol generates a safety number for each conversation, represented as a numerical string. During the authentication ceremony, both users are presented with a single safety number displayed on their respective screens, which can be exchanged via SMS or voice call.

The user's decision to accept or reject the conversation depends on the comparison of these safety numbers: matching safety numbers indicate a successful secure conversation, whereas non-matching safety numbers indicate a MitM attack. Consequently, we want to make MitM attacks as difficult as possible by automating the process of comparing the safety numbers and taking the user out of the loop of comparing and verifying the safety numbers.

The term *MitM attack* in this paper specifically pertains to a data MitM attack. Note that another type of attack, called the voice MitM attack, has been introduced in [26]. The attack involves tampering with the voice channel of communication using advancements in voice synthesis and conversion technology. The attacker compromises the key exchange protocol and inserts their own voice or a morphed/converted voice of the other user into the communication in an attempt to deceive the user into accepting the speaker as legitimate. The voice MitM attack is more complex to execute than the common data MitM attack, which is the most straightforward form of attack. In a data MitM attack, the attacker intercepts and alters the data being transmitted between two users. However, in a voice MitM attack, the attacker needs to alter the voice channel of communication by inserting their own or a converted voice, which is a more sophisticated task. Therefore, the voice MitM attack is not within the scope of this work. Also, given that many E2EE applications only ask the users to perform the comparison task to detect data MitM attacks and do not explicitly ask the users to perform the speaker verification task to detect voice MitM attacks, it is important to note that our approach aims to detect only data MitM attacks.

3 DESIGN AND IMPLEMENTATION

In this section, we present two novel methods for automating the safety number comparison process in the authentication ceremony of Signal. We also describe how we have integrated these methods into the current version of the Signal application. Our modified version of the Signal application is shown in Figure 1. We redesigned its underlying authentication ceremony to automate the process of safety number comparison using SMS and the voice channel.

3.1 Automated Comparison Tool Using SMS

The idea behind the use of SMS in our automated approach is inspired by the concept of the iPhone's SMS code capture and two-factor authentication (2FA) code [14, 37]. This approach comprises two fundamental tasks. The first task is to share a safety number over a text message, utilizing SMS at one endpoint. The second task is to listen for any incoming SMS message at the other endpoint and automatically extract, compare, and verify the safety number that is included in the SMS details. This idea provides a fully automated comparison of safety numbers since users are unable to perform the authentication ceremony successfully in current E2EE applications as reported in many research studies [13, 25, 29, 35]. Older versions of the Signal app required users to compare two sets of hexadecimal characters or scan two QR codes [23]. The updated version simplifies this into a single comparison or scan task. However, manual safety number comparison was found to be error-prone in previous studies, leading to false positives, false negatives, and low success rates [13, 25, 29].

We integrated our automated safety number comparison tool into Signal by redesigning its authentication ceremony. Our goal was to automate the safety number comparison process immediately upon receiving incoming SMS messages. We place our automated comparison tool in the modified version of the Signal application, which listens for any incoming SMS message (sent by one end, such as Bob as a sender in Figure 1) and handles its details that contain the safety number at the other end (Alice as a receiver in Figure 1), followed by comparing the safety number included in the SMS message with the locally generated safety number on Alice's application. This contrasts with the manual comparison in the current version of the Signal application, making sharing and comparison effortless.

In our automated comparison tool, we use the *BroadcastReceiver* class in the Android studio to receive SMS messages [8]. Our automated comparison tool will send out system broadcasts of events like receiving an SMS message. These system broadcasts will contain intents that are meant to be received by the *BroadcastReceiver* class. To receive SMS messages through the modified version of the Signal application, users must grant permission to receive and read SMS on the app. When a new SMS message is received, the app will automatically extract the safety number and compare it to the number generated on the receiver's app. If they match, the sender is verified, and a message is displayed. If they do not match, a message will be displayed indicating that the sender is not verified. This confirms that an adversary is participating in the communication by acting as a MitM attacker.

3.2 Automated Comparison Tool Using Voice

The idea behind the use of the phone call in our automated approach consists of two main tasks. The first task is to transmit a voice speaking the safety number over a phone call. The second task is to convert the spoken audio signals into a written text by recognizing the phonemes of the spoken speech and then comparing and verifying the spoken safety number.

Our new automated approach offers users four different methods for transmitting the spoken safety number over a phone call. We refer to these methods as *the delivery methods of the spoken safety number*. These methods include tapping the TTS button to activate the TTS service and have the application speak the generated safety number, speaking the safety number aloud during the call, creating an audio file by recording the spoken safety number, or leaving a voicemail message with the safety number. We utilize the TTS service supported by Android to transform the generated safety number within the user's application into speech [10], which can subsequently be played back instantly, similar to Google's TTS feature on Android phones [5]. Our approach can recognize both TTS-generated speech and natural human voice, providing flexibility for users.

We redesigned Signal's authentication ceremony to demonstrate the feasibility of integrating our automated comparison tool using voice over a phone call. In our modified version of the Signal application, we built our automated comparison tool on top of the powerful STT API (`android.speech`) supported by Android [7]. Our concept automates the safety number comparison by leveraging speech

transcription during authentication. We utilize the *SpeechRecognizer* class in Android Studio to activate STT technology during a phone call [9]. In our modified Signal app, Bob speaks the safety number, which gets transcribed and compared at Alice's end. This eliminates Alice's need for manual comparison and is a contrast to Signal's current version. Our approach allows Bob to audibly share the safety number during a phone call, with Alice's app automatically handling the comparison. Additionally, our modified app can trigger the transcription process via a specific keyword (e.g., "Do Ceremony"), thereby simplifying the authentication ceremony.

4 EVALUATION

This section presents our experiment evaluating the integration of our proposed automated methods into the Signal app's authentication ceremony. We demonstrate the ease of incorporating these methods using two OOB channels, leveraging Android's features. Our approach ensures seamless automation of the safety number comparison during the ceremony. We not only demonstrate its effectiveness in benign scenarios but also highlight its resistance to MitM attacks.

4.1 Integrating with the Signal Application

In order to evaluate the effectiveness and feasibility of our proposed automated methods for performing the authentication ceremony in the Signal application, we modified the underlying authentication ceremony in the current version of the Signal application based on using two OOB channels.

Experimental Setup: Using Android Studio, we modified the Signal app, incorporating our two new methods into the authentication ceremony task. Our modifications included altering the view layout to include our automated tasks. Figure 1 illustrates this modified Signal app with our added features. We provide clear instructions to guide users through utilizing our automated methods for safety number comparison. The blue box presents the outcomes of these automated mechanisms. Our experiment involved testing this modified app on two separate Android devices, each with a distinct SIM card and phone number. We installed our modified version of the Signal application on both of them and registered Bob and Alice as users. Bob's contact information is stored on one phone, whereas Alice's information is stored on the other phone. The purpose of the contact information is to facilitate establishing a conversation and performing an authentication ceremony during testing.

We tested our modified Signal app's handling of new automated methods in the authentication ceremony. For SMS-based testing, a safety number was generated on Bob's app and sent to Alice's phone via SMS. Our approach utilized Android's *BroadcastReceiver* class to manage incoming SMS messages on Alice's phone. After installing our app, users are asked for permission to automatically read these messages, ensuring privacy. Our method extracted the safety number from the SMS and compared it to the number generated by Alice's app. If the safety numbers are identical, a verification message will appear in the blue box to indicate that the safety numbers match, confirming that the conversation with a partner is private. If the safety numbers do not match, a verification message will show up in the blue box, indicating that a MitM attacker has compromised the conversation.

To test our automated comparison and verification process using STT technology over a phone call, we made a call between two users and compared the safety numbers during the call. Bob's Signal application generated a safety number, which could be read aloud by activating the TTS engine with the TTS button (see Figure 1). Bob had the choice to announce the safety number using the spoken delivery options supported by our automated approach. At the receiving end, our method automatically compared the spoken safety number with the one generated on Alice's Signal application. During the authentication ceremony, our method is triggered by a predefined phrase ("Do Ceremony") that activates the STT technology for transcription. After processing and comparing the transcribed audio, a verification message would appear in the blue box if the numbers match, confirming a secure conversation. If the generated safety number and the transcription do not match, a verification message in the blue box will indicate a compromised conversation due to a MitM attacker.

In order to ensure that our results were accurate and consistent, we ran this experiment 10 times, each time using a different OOB channel (SMS and phone call, including the usage of the delivery methods of the spoken safety number). When using SMS or any other form of the spoken safety number, we tested our automated comparison tool 10 times when the two safety numbers matched and 10 times when they did not match. Efficiency is assessed by determining the average time that each method spends on the comparison task, providing a robust metric for evaluating the time spent by each method.

Observations: We are examining the feasibility of fully automating the safety number comparison during the Signal application's authentication ceremony. We are considering using OOB channels to exchange safety numbers and then enable automatic comparison and verification of safety numbers. The results indicate that our proposed methods can indeed achieve full automation. By exchanging safety numbers through an SMS message, the receiver's application can automatically compare the received number with its own generated safety number. For instance, when Bob sends Alice an SMS with the safety number, Alice's automated comparison tool reads and compares it to her generated safety number, as demonstrated in Figures 4a and 4c.

Our results also confirm that our second automated method can effectively convert spoken safety numbers during a phone call into text (digits) using STT technology. This transcribed text (digits) is then compared to the safety number generated by the modified Signal application on the receiver's phone. Notably, our automated approach is versatile and capable of transcribing spoken safety numbers whether uttered directly, spoken using TTS technology, recorded in voicemail messages, or saved as audio files. Once the transcription is received, it is automatically compared to the generated safety number on the receiver's application. Importantly, the receiver does not need to take any action or press any buttons; the transcription process is triggered by the keyword "Do Ceremony," much like real-world voice commands in Google (e.g., "Hey Google"). Our observations indicate that when the receiver uses this keyword during a phone call, our automated method initiates the transcription process. Figures 4b and 4d illustrate this process, where the keyword triggers the automated method to transcribe and compare the spoken safety number.

The efficiency of our automated approach is an important consideration in our study. Therefore, a study was conducted to compare the time taken for different methods during the authentication ceremony, including SMS, TTS, oral speaking, an audio file, and voicemail. The average time taken for each method was calculated, and the results were analyzed. As shown in Table 2, the SMS method was the fastest, taking almost no time at all. However, the study showed that the other methods also took a reasonable amount of time. The average time taken by the automated method via TTS was 23.38 seconds, which was faster than the oral speaking method, which took 31.36 seconds. The audio file and voicemail methods took almost the same time, with an average time of 30.96 seconds and 31.22 seconds, respectively.

Using our automated comparison tool, we also determined the accuracy rate for each automated testing case (where the test is done using SMS or a phone call with the related delivery methods of the spoken safety number). The overall accuracy rate is computed by dividing the number of test cases that were correctly compared by the total number of test cases taken with the modified Signal application. Table 2 displays the results of the correct and accurate comparison rates. As can be seen from the table, our automated comparison tool yields a 100% accuracy rate. From Figure 4, we can see that our modified Signal application displays the output of our automated approach (using either SMS or a phone call) in the blue box to inform the user whether or not the safety number matches and whether or not the other party is verified. Our two automated methods can help the user perform the authentication ceremony and verify the other party without imposing more work on the user.

4.2 Robustness Against MitM Attacks and Accuracy in Benign Sessions

To evaluate the robustness of our automated methods against MitM attacks, we are examining their response to mismatched safety numbers during the authentication ceremony. This evaluation helps us compute the false acceptance rate (FAR), representing the likelihood of erroneously accepting such mismatches. Simulating a MitM attack, we manually generate mismatched safety numbers as a simulation since executing an actual MitM attack is not in the scope of our work. Additionally, given that the existing Signal authentication relies on manual safety number comparison, there is a possibility of users mistakenly rejecting legitimate sessions due to difficulties in this comparison. To understand the performance of our automated methods during benign sessions, we are also investigating how frequently they reject matching safety numbers. This analysis helps us determine the false rejection rate (FRR), reflecting the probability of wrongly rejecting a benign session even when the transmitted safety number matches the recipient's generated one.

Experimental Setup: We defined three cases of mismatched safety numbers for the MitM attack scenarios. We created, as shown in Figure 5, a safety number with a single mismatched digit, a safety number with a single block of mismatched digits, and a completely incorrect safety number. For each instance of attack, the experiment was done 10 times over SMS and 10 times over the phone. To perform the simulated MitM attack, we used two Android phone devices as in the previous subsection. We installed our modified

Table 2: The Results of Evaluating Our Automated Approach in the Modified Signal Application

Data type	SMS	Phone Call			
		TTS	Orally	Audio File	Voicemail
	Text	Audio	Audio	Audio	Audio
The feature used to extract the safety number	BroadcastReceiver on Android system	The STT technology built into Android	The STT technology built into Android	The STT technology built into Android	The STT technology built into Android
Ask the user for permission	Yes, for automatically reading SMS	Yes, for recording audio	Yes, for recording audio	Yes, for recording audio	Yes, for recording audio
Fully automate the process of comparing the safety numbers	Yes	Yes	Yes	Yes	Yes
The correct comparison rate of matching safety numbers	100%	100%	100%	100%	100%
The correct comparison rate of mismatching safety numbers	100%	100%	100%	100%	100%
Average time taken (in seconds)	Immediately	23.38	31.36	30.96	31.22
FAR in attacking case (single mismatched digit)	0%	0%	0%	0%	0%
FAR in attacking case (a single block of mismatched digits)	0%	0%	0%	0%	0%
FAR in attacking case (completely incorrect safety number)	0%	0%	0%	0%	0%
FRR in benign case	0%	0%	0%	0%	0%

version of the Signal application on both of them and registered Bob and Alice as users with their exchanging contact information. One user was used as a peer in the conversation to share or say out loud the incorrect safety number via SMS or phone call. For security assessment against MitM attacks when using our automated approach via SMS, one user (Bob) tapped on the (share) button embedded within the UI to send the recipient (Alice) his Signal safety number via an SMS message. Also, for security assessment against MitM attacks when using our automated approach over a phone call, one user (Bob) audibly announced the incorrect safety number using one of the delivery methods for the spoken safety number. As soon as the incorrect safety number was delivered to the receiver's application by SMS or a phone call, our automated comparison tool compared it to the safety number that had been generated on the receiver's application.

Our experiment involved having Bob verify the safety number in one direction only. While our automated approach can support verification in both directions, it is important to note that the authentication ceremony in the Signal application is currently unidirectional, with safety number comparison occurring one way. This enhancement is facilitated by the safety number concept, reducing the need for two comparisons to just one [23]. By assigning a unique safety number to each conversation, the focus shifts to per-conversation verification rather than per-user verification. This results in a streamlined verification process, requiring just one comparison and eliminating the need for reciprocal verifications as previously required. For benign scenarios, we also used the same experimental setup. However, we introduced one instance of a matching safety number for testing our automated approach. This involved sending the matching safety number via SMS or making a phone call to announce it. This test was conducted 10 times for

each OOB channel (SMS and phone call) to ensure accurate and consistent results in benign cases.

Observations: To study the robustness of our automated methods against MITM attacks, we evaluate the occurrences of accepting mismatching safety numbers that are transmitted over an SMS message or a phone call, which ultimately results in the success of a MitM attack. To this end, we compared the incorrect safety number transmitted via an SMS message or a phone call with the generated safety number on the receiver's application. The results show that our automated approach, which makes use of both OOB channels (SMS and phone call), is able to reject any incorrect safety number, regardless of the type of mismatched safety number that occurred (a single mismatched digit, a single block of mismatched digits, or a completely incorrect safety number). Using our automated comparison tool, and given that our correct comparison rates are 100%, the results demonstrate that the FAR is demonstrably equal to 0% in every attack scenario when our automated approach is used in the authentication ceremony. This indicates that our automated approach is more resistant to MitM attacks. Therefore, our automated approach significantly reduces the risk of accepting an attacked safety number during a session, as none of the challenges in our experiment matched any valid safety number. This demonstrates how automated comparison can prevent MitM attacks during the Signal application's authentication ceremony. Additionally, our study revealed that automating the safety number comparison eliminates the risk of human error, thereby enhancing the system's robustness. As can be seen in Table 2, by utilizing our automated approach in the authentication ceremony and taking into account the fact that our correct comparison rates are 100%, the results demonstrate that the FRR is demonstrably equal to 0% in every benign session.

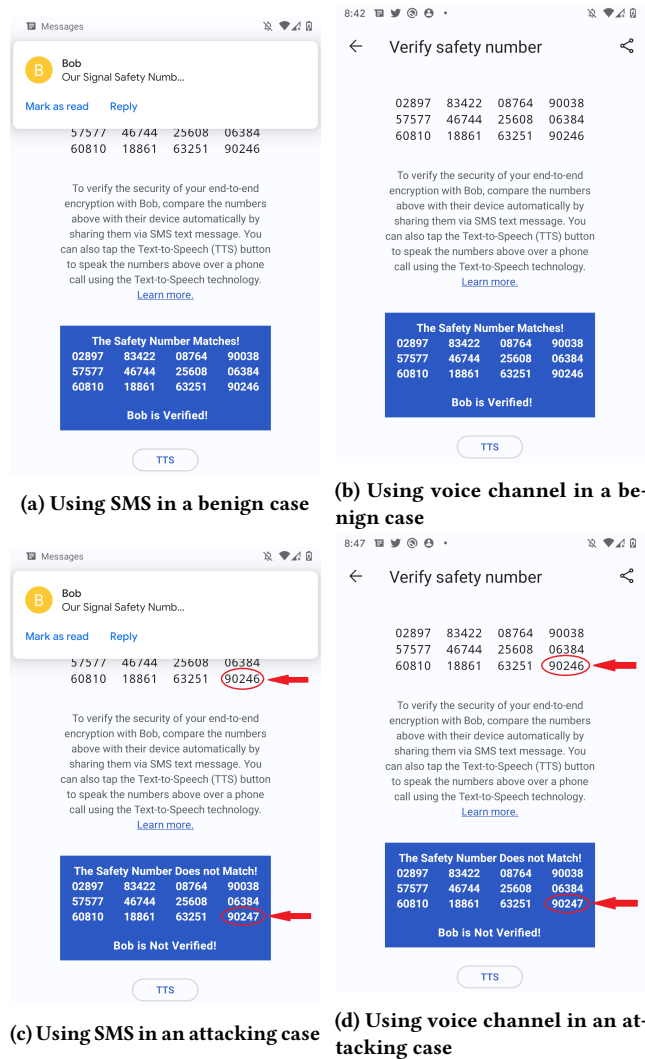


Figure 4: The interface of our automated comparison tool in the underlying authentication ceremony of Signal

5 DISCUSSION

5.1 Strengths and Limitations of Our Study

We believe that our study has several strengths. Our automated comparison tool can prevent MitM attacks and address skip-through problems in E2EE applications. Integrating this tool can enhance security in messaging apps like Signal [30], WhatsApp [38], Viber [36], and Skype [31]. Our automated tool can successfully decrease both FAR and FRR to 0% by automating the process of comparing safety numbers during the underlying authentication ceremony of the Signal application. This is achieved by exchanging safety numbers through two separate OOB channels (SMS and voice), using the *BroadcastReceiver* class in Android Studio to automatically receive SMS messages, and the *SpeechRecognizer* class in Android Studio to activate STT technology during a phone call.

Our modified version of Signal automates the comparison of safety numbers, eliminating the need for a usability study. The current manual comparison process in Signal [30] can be challenging, especially when dealing with long numeric codes. Our automated approach aims to remove this burden from users, improving the user experience significantly without depending on manual input. Therefore, our approach automates fingerprint comparison, freeing users from manual tasks. OOB channels facilitate remote authentication, enabling fingerprint sharing when users are not physically present. For instance, the Signal app provides a "share" button for direct safety number transmission via SMS, automatically copying the safety number to the clipboard for sharing. We have also introduced a TTS button for phone calls, which reads the locally generated safety number using the TTS engine. On the receiver side, no action is required, as our method automatically processes the incoming SMS message, aligning with the iPhone's SMS code capture concept. Similarly, by employing STT functionality during a phone call, our method transcribes and compares the spoken safety number, eliminating the need for manual fingerprint comparison, a significant advantage in remote settings.

It is also worth noting that the CCCP system [28] depends on SAS that is converted into words, where a string of separate words (not meaningful text) must be transcribed. Therefore, the dictionary used in CCCP is chosen from a phonetically balanced speech collection that encompasses a language commonly spoken in general conversation. Due to this selection, the terms in the dictionary are not always phonetically unique from each other, leading to a reported false rejection rate of 24.57% for 4-word fingerprints and 63.17% for 8-word fingerprints. In our work, we only use numbers for the transcription of the safety number by the STT technology, since the Signal application utilizes a numerical representation for the safety number. This way, the STT technology can be used only to recognize and transcribe numbers. The accuracy of Google's STT technology varies depending on various factors, such as the quality of the audio input, the speaker's accent, the background noise level, and the specific language being spoken. However, in general, Google's STT technology is highly accurate and considered one of the best in the industry [16]. Using the voice channel, we conducted our experiment with a background noise level similar to that of a quiet home, ranging between 20 dB and 25 dB. The primary objective was to assess the feasibility of implementing our automated approach in real-life applications like Signal [30], without considering usability concerns. We encountered no challenges in utilizing the STT technology to accurately transcribe the numerical values during the authentication ceremony when exchanging safety numbers via phone calls, as the technology exclusively recognizes digits. However, it is worth noting that a potential avenue for future research involves exploring the implementation of our automated approach using phone calls in specific circumstances to evaluate the efficiency and usability of the STT technology in such scenarios.

In current E2EE apps like Signal [30], the optional authentication ceremony can be skipped or hurriedly executed, increasing vulnerability to MitM attacks. In practice, it has been shown that this kind of behavior is common in any authentication or verification process [18, 29]. Our automated approach eliminates manual key fingerprint verification, making the ceremony more likely to be completed accurately. With a 0% FAR, it robustly detects ongoing MitM attacks,

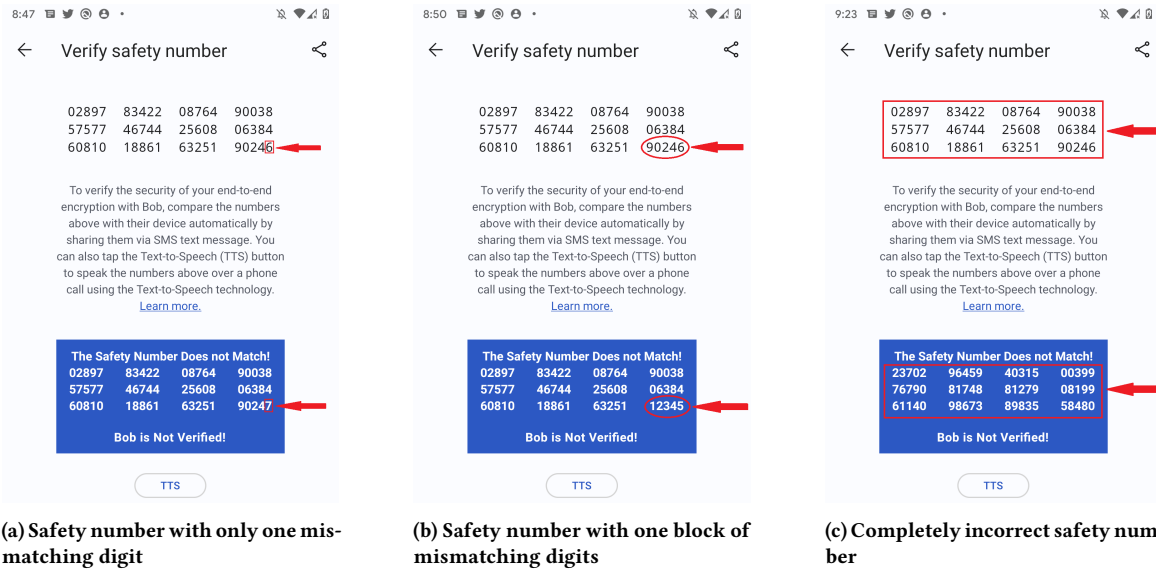


Figure 5: Three examples of the different instances of MitM attacks

empowering users to safeguard their mobile communications. This could enhance E2EE security by encouraging users to complete the authentication ceremony and maintain encrypted communication. Our automated approach demonstrates high accuracy due to its straightforward process, especially in the case of SMS verification. In this scenario, the application fetches plain text from SMS messages and automatically compares the numbers. Additionally, our experiments with phone call verification were conducted under controlled conditions, simulating a background noise level comparable to that of a quiet home, ranging between 20 dB and 25 dB. Furthermore, the use of modern phone microphones equipped with sophisticated noise-cancellation technology contributes to the precision of our approach, ensuring reliable and accurate comparisons even in less-than-ideal acoustic environments.

5.2 Enhancing Group Authentication Ceremonies in E2EE Applications

Our proposed automated comparison tool that utilizes SMS and STT technology could be an effective solution to the challenges of manual key comparison during group authentication ceremonies in E2EE applications. The majority of E2EE applications follow the same pairwise individual authentication procedure for both one-to-one and group communication scenarios, which can make the authentication process difficult and challenging when the group has a large number of members. The tool can eliminate human errors that may occur during the current authentication procedures that rely on pairwise individual authentication, reducing the risk of potential MitM attacks. As highlighted in a recent security analysis of Letter Sealing [15], the vulnerability of the key-derivation stage in group message encryption can be exploited by end-to-end adversaries, malicious group members, or users to mount impersonation or forgery attacks on the group message encryption scheme. Although most E2EE apps follow similar authentication procedures

for one-to-one and group scenarios, some vary in terms of their authentication ceremony settings, including presenting keys to users during the ceremony while keeping the fingerprint verification pairwise. While some apps, like Facebook Messenger [11] and LINE [21], list all group members or their keys in one list to facilitate the participation of group members in the authentication ceremony, the Zoom app offers group-based authentication. Here, our proposed tool can enhance the verification of the security code for all Zoom meeting members by enabling users to compare and verify the 40-digit number through SMS or STT technology, mitigating the risk of a potential MitM attack. Therefore, incorporating our proposed tool in group authentication ceremonies for E2EE applications can facilitate the participation of group members and increase the usability of the ceremony, thus enhancing the efficiency and security of the authentication process.

6 CONCLUSION

In this paper, we introduce a novel automated approach to the authentication ceremony in the Signal application, which eliminates human errors and fully automates the safety number comparison task. The integration of our automated approach into the Signal application demonstrates its effectiveness in resisting MitM attacks and significantly improving security and usability. By using two OOB channels and removing users from the comparison process, our automated approach significantly improves the security of the authentication ceremony, making it more robust than the current implementation in the Signal application. The effectiveness and feasibility of our automated approach have been demonstrated, and our results show that it provides a 0% false acceptance and 0% false rejection rate for the safety number comparison, thereby significantly improving security and usability. Our automated approach eliminates human errors in the benign case and completely

resists MitM attacks, making it a practical and effective solution for addressing the challenges in the authentication ceremony process.

ACKNOWLEDGMENTS

We would like to give special thanks to the anonymous reviewers for their valuable feedback on this paper.

REFERENCES

- [1] Mashari Alatawi and Nitesh Saxena. 2023. SoK: An Analysis of End-to-End Encryption and Authentication Ceremonies in Secure Messaging Systems. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '23)*. Association for Computing Machinery, New York, NY, USA, 187–201. <https://doi.org/10.1145/3558482.3581773>
- [2] Nikita Borisov, Ian Goldberg, and Eric Brewer. 2004. Off-the-Record Communication, or, Why Not to Use PGP. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society (WPES '04)*. Association for Computing Machinery, New York, NY, USA, 77–84. <https://doi.org/10.1145/1029179.1029200>
- [3] Melissa Chase, Apoorva Deshpande, Esha Ghosh, and Harjasleen Malvai. 2019. SEEMless: Secure End-to-End Encrypted Messaging with Less Trust. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*. Association for Computing Machinery, New York, NY, USA, 1639–1656. <https://doi.org/10.1145/3319535.3363202>
- [4] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. 2020. A formal security analysis of the signal messaging protocol. *Journal of Cryptology* 33, 4 (2020), 1914–1983. <https://doi.org/10.1007/s00145-020-09360-1>
- [5] James Davis. 2023. *How to Use Google Text-To-Speech on Android*. Retrieved October 11, 2023 from <https://drfone.wondershare.com/android-tips/text-to-speech.html>
- [6] Sergej Dechand, Dominik Schürmann, Karoline Busse, Yasemin Acar, Sascha Fahl, and Matthew Smith. 2016. An Empirical Study of Textual Key-Fingerprint Representations. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 193–208. <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/dechand>
- [7] Google Developers. 2023. *android.speech*. Retrieved October 11, 2023 from <https://developer.android.com/reference/android/speech/package-summary>
- [8] Google Developers. 2023. *Broadcasts overview*. Retrieved October 03, 2023 from <https://developer.android.com/guide/components/broadcasts>
- [9] Google Developers. 2023. *SpeechRecognizer*. Retrieved October 11, 2023 from <https://developer.android.com/reference/android/speech/SpeechRecognizer>
- [10] Google Developers. 2023. *TextToSpeech*. Retrieved October 11, 2023 from <https://developer.android.com/reference/android/speech/tts/TextToSpeech>
- [11] Facebook Messenger. 2024. <https://www.messenger.com/>.
- [12] Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Jörg Schwenk, and Thorsten Holz. 2016. How Secure is TextSecure?. In *2016 IEEE European Symposium on Security and Privacy (EuroSP)*. IEEE, 457–472. <https://doi.org/10.1109/EuroSP.2016.41>
- [13] Amir Herzberg and Hemi Leibowitz. 2016. Can Johnny Finally Encrypt? Evaluating E2E-Encryption in Popular IM Applications. In *Proceedings of the 6th Workshop on Socio-Technical Aspects in Security and Trust (STAST '16)*. Association for Computing Machinery, New York, NY, USA, 17–28. <https://doi.org/10.1145/3046055.3046059>
- [14] Apple Inc. 2023. *Automatically fill in SMS passcodes on iPhone*. Retrieved October 03, 2023 from <https://support.apple.com/guide/iphone/automatically-fill-in-sms-passcodes-iphc89a3a3af/ios>
- [15] Takanori Isohe and Kazuhiko Minematsu. 2018. Breaking Message Integrity of an End-to-End Encryption Scheme of LINE. In *European Symposium on Research in Computer Security*, Javier Lopez, Jianying Zhou, and Miguel Soriano (Eds.). Springer, Springer International Publishing, Cham, 249–268. https://doi.org/10.1007/978-3-319-98989-1_13
- [16] Nishit Kamdar. 2022. *Measuring and Improving Speech-to-Text Accuracy | Google Cloud Platform*. Retrieved October 11, 2023 from <https://medium.com/google-cloud/measuring-and-improving-speech-to-text-accuracy-google-cloud-platform-eba62c50b8ac>
- [17] Keybase. 2024. <https://keybase.io/>.
- [18] Cynthia Kuo, Jesse Walker, and Adrian Perrig. 2007. Low-cost manufacturing, usability, and security: An analysis of bluetooth simple pairing and wi-fi protected setup. In *Financial Cryptography and Data Security: 11th International Conference, FC 2007, and 1st International Workshop on Usable Security, USEC 2007, Scarborough, Trinidad and Tobago, February 12-16, 2007. Revised Selected Papers 11*. Springer, 325–340. https://doi.org/10.1007/978-3-540-77366-5_30
- [19] Adam Langley. 2009. Opportunistic encryption everywhere. In *W2SP (2009)*.
- [20] Sean Lawlor and Kevin Lewi. 2023. *Deploying key transparency at WhatsApp*. Retrieved November 27, 2023 from <https://engineering.fb.com/2023/04/13/security/whatsapp-key-transparency/>
- [21] Line. 2024. <https://line.me/en/>.
- [22] Harjasleen Malvai, Lefteris Kokoris-Kogias, Alberto Sonnino, Esha Ghosh, Ercan Öztürk, Kevin Lewi, and Sean Lawlor. 2023. Parakeet: Practical Key Transparency for End-to-End Encrypted Messaging. *Cryptology ePrint Archive* (2023). <https://doi.org/10.14722/ndss.2023.24545>
- [23] Moxie Marlinspike. 2016. *Safety number updates*. Retrieved September 09, 2023 from <https://signal.org/blog/safety-number-updates/>
- [24] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. 2015. {CONIKS}: Bringing key transparency to end users. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 383–398. <https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/melara>
- [25] Svenja Schröder, Markus Huber, David Wind, and Christoph Rottermann. 2016. When SIGNAL hits the Fan: On the Usability and Security of State-of-the-Art Secure Mobile Messaging. In *European Workshop on Usable Security, IEEE*. 1–7. <https://doi.org/10.14722/eurosec.2016.23012>
- [26] Maliheh Shirvanian and Nitesh Saxena. 2016. Wiretapping via Mimicry: Short Voice Imitation Man-in-the-Middle Attacks on Crypto Phones. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Association for Computing Machinery, New York, NY, USA, 868–879. <https://doi.org/10.1145/2660267.2660274>
- [27] Maliheh Shirvanian and Nitesh Saxena. 2015. On the Security and Usability of Crypto Phones. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC '15)*. Association for Computing Machinery, New York, NY, USA, 21–30. <https://doi.org/10.1145/2818000.2818007>
- [28] Maliheh Shirvanian and Nitesh Saxena. 2017. CCCP: Closed Caption Crypto Phones to Resist MITM Attacks, Human Errors and Click-Through. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1329–1342. <https://doi.org/10.1145/3133956.3134013>
- [29] Maliheh Shirvanian, Nitesh Saxena, and Jesvin James George. 2017. On the Pitfalls of End-to-End Encrypted Communications: A Study of Remote Key-Fingerprint Verification. In *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC '17)*. Association for Computing Machinery, New York, NY, USA, 499–511. <https://doi.org/10.1145/3134600.3134610>
- [30] Signal. 2024. <https://signal.org/>.
- [31] Skype. 2024. <https://www.skype.com/en/>.
- [32] Joshua Tan, Lujia Bauer, Joseph Bonneau, Lorie Faith Cranor, Jeremy Thomas, and Blase Ur. 2017. Can Unicorns Help Users Compare Crypto Key Fingerprints?. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 3787–3798. <https://doi.org/10.1145/3025453.3025733>
- [33] Telegram. 2024. <https://telegram.org/>.
- [34] Elham Vaziripour, Devon Howard, Jake Tyler, Mark O'Neill, Justin Wu, Kent Seamons, and Daniel Zappala. 2019. I Don't Even Have to Bother Them! Using Social Media to Automate the Authentication Ceremony in Secure Messaging. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. <https://doi.org/10.1145/3290605.3300323>
- [35] Elham Vaziripour, Justin Wu, Mark O'Neill, Ray Clinton, Jordan Whitehead, Scott Heidbrink, Kent Seamons, and Daniel Zappala. 2017. Is That You, Alice? A Usability Study of the Authentication Ceremony of Secure Messaging Applications. In *Proceedings of the Thirteenth USENIX Conference on Usable Privacy and Security (SOUPS '17)*. USENIX Association, USA, 29–47.
- [36] Viber. 2024. <https://www.viber.com/en/>.
- [37] Vineet Washington. 2020. 'Autofill Code From Messages' Feature Surfaces to Make Entering 2FA Codes From SMS Easier: Report. Retrieved October 03, 2023 from <https://www.gadgets360.com/apps/news/android-autofill-code-from-messages-feature-rolling-out-select-report-2178596>
- [38] WhatsApp. 2024. <https://www.whatsapp.com/>.
- [39] Tarun Kumar Yadav, Devashish Gosain, Amir Herzberg, Daniel Zappala, and Kent Seamons. 2022. Automatic Detection of Fake Key Attacks in Secure Messaging. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. Association for Computing Machinery, New York, NY, USA, 3019–3032. <https://doi.org/10.1145/3548606.3560588>
- [40] Ruishan Zhang, Xinyuan Wang, Ryan Farley, Xiaohui Yang, and Xuxian Jiang. 2009. On the Feasibility of Launching the Man-in-the-Middle Attacks on VoIP from Remote Attackers. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS '09)*. Association for Computing Machinery, New York, NY, USA, 61–69. <https://doi.org/10.1145/1533057.1533069>
- [41] Phil Zimmermann, Alan Johnston, and Jon Callas. 2011. ZRTP: Media path key agreement for unicast secure RTP. *Internet Engineering Task Force (IETF)* (2011), 2070–1721.