Poster: Automatic Detection and Prevention of Fake Key Attacks in Signal

Tarun Kumar Yadav
Brigham Young University

Devashish Gosain
IIIT Delhi

Daniel Zappala
Brigham Young University

Kent Seamons
Brigham Young University

***Abstract*—The Signal protocol provides end-to-end encryption for billions of users in popular instant messaging applications such as WhatsApp. The protocol relies on an application-specific key server to distribute public keys and relay encrypted messages between the users. As a result, Signal prevents passive attacks but is vulnerable to some active attacks. A malicious or hacked key server can distribute fake keys to users to perform man-in-the-middle or impersonation attacks. While typical secure messaging applications provide a manual method for users to detect these attacks, this places an undue burden on users and studies show it is ineffective in practice.**

In this poster, we design several defenses for fake key attacks and use a threat analysis to identify which attacks each defense can automatically detect or prevent. We implement the attacks to demonstrate they are possible, and we use an implementation of two of the defenses to confirm that they work as designed and are feasible.

I. Introduction

The Signal protocol is an end-to-end encryption protocol that provides forward secrecy and was designed specifically for messaging applications. It is used by billions of users in popular instant messenger applications like WhatsApp, Signal Private Messenger and Facebook Messenger's "secret chat" feature, *etc.*

Currently, messaging applications that use the Signal protocol rely on a trusted key server for distributing identity keys among the communicating clients. A malicious (or hacked) key server (MKS) can launch a MITM attack against Alice and Bob, by providing them with fake keys. To defend against this attack, presently users have to manually compare keys with their contact when they start communication or when their contact update their keys. Most of the key updates are legitimate due to app reinstall by users, which leads to ignore these key update warnings by users. Hence there is a pressing need to somehow distinguish between benign key updates due to application re-installation and fake key attacks.

Recent user studies [1]–[3] have found that users are generally oblivious to the need to verify identity keys and are unable to authenticate when attacked.

The aim of our research is to automate the detection and prevention of fake key attacks in Signal to ease the burden of users having to manually verify keys. We are designing three approaches and analyzing which kinds of attacks they can prevent or detect. Automated defenses against fake key attacks can serve as a deterrent to the key server to act maliciously (*e.g.,* by responding to a government subpoena).

A. Threat Model

We consider an **impersonation attack** where an attacker gives a fake key for Alice to Bob to impersonate Alice to Bob, and a **MitM attack** where an attacker gives a fake key for Alice to Bob and a fake key for Bob to Alice. The attacker impersonates both Alice to Bob and Bob to Alice. The possible attack scenarios in which an attacker could launch a MitM or impersonation attacks are:

- Attacks on a new communication session setup
  - Attack on Alice/Bob conversations (pair targeted)
  - Attack on all of Alice's conversations (client targeted)
- Attacks on established communication sessions
  - Attack on Alice/Bob conversations (pair targeted)
  - Attack on all of Alice's conversations (client targeted)

II. Defenses

A. Key Transparency

We started with extending Signal to support CONIKS [4]. CONIKS uses Key Transparency and auditing to ensure that identity providers cannot equivocate about the public keys they advertise on behalf of users.

In general, the CONIKS architecture assumes there are multiple non-colluding identity providers. However, in messaging apps like WhatsApp and Signal there is only one key server that can act as a provider, and there is no existing consortium that provides non-colluding distributed entities that could act as identity providers. To integrate CONIKS with Signal, we analysed both 1) multiple provider hypothetical scenario and 2) single provider scenario.

B. Trust Network

In Trust Network, clients verifies the authenticity of the key from her *trusted* contacts. Trusted contacts are those contacts whose keys Alice has previously verified with this method. To initialize this set, Alice can perform manual key verification with several contacts. The basic idea of this defense is that Alice uses her trusted contacts to verify whether they have the same key for Bob that the key server gave her.

There are two mechanisms the Trust Network uses to verify keys—one uses mutual contacts and the other uses random contacts as relays.

TABLE I
TAXONOMY OF FAKE KEY ATTACKS AND ANALYSIS OF DEFENSES

	MITM				Impersonation			
	Pair Targeted		Client Targeted		Pair Targeted		Client Targeted	
Defense	New	Existing	New	Existing	New	Existing	New	Existing
Key Transparency (multiple-provider)	○	○	○	○	○	○	○	○
Key Transparency (single-provider)	○	○	○		○	○	○	
Trust Network (mutual contact)	●	●	●		●	●	●	
Trust Network (contact relay)	○	○	○		○	○	◔	
Anonymous Key Retrieval (basic)	○	○	○	○	○	○	○	○
Anonymous Key Retrieval (advanced)	●	●	●	●	●	●	◐	●
Mass key update monitoring (naive)				●				●
Mass key update monitoring (stealthy)				○				○†
Isolation monitoring								○‡
Key history monitoring	◔	◔	◔	◔	◔	◔	◔	◔

● prevents the attack, ○ detects the attack, ◐ sometimes prevents the attack, ◔ sometimes detects the attack, (blank) susceptible
† impersonating to Alice, ‡ impersonating as Alice

1) Mutual Contact: Whenever Alice starts communicating with a user, her client asks the new client to send all of its contacts' verification bundles. This gives Alice the hash of the identifier and identity key of all of her two-hop contacts (her contacts' contacts). The use of a hash function in the verification bundle protects the privacy of contacts, since Alice will not know the identity of two-hop contacts that she is not already herself contacts with.

2) Contact Relay: If Alice does not have a mutual contact with Bob, her client randomly selects any one of her contacts, eg. Carol, to act as a proxy for her. Alice asks Carol to obtain Bob's key bundle from the key server on her behalf. Carol may relay this request to additional random contacts. Once Alice receives a key from a relay, Alice compares this key with the one provided directly by the key server. On successful match, Alice trusts the key, otherwise an attack is detected.

C. Anonymous Key retrieval

Anonymous Key Retrieval defends against fake key attacks by anonymously requesting keys, thus making it more difficult for the server to distribute fake keys without being detected. Clients requesting keys do not include any uniquely identifiable information in their requests. Our design assumes an anonymization service such as Tor [5] that has a robust architecture to provide anonymity to its users.

We propose two levels of defense, basic and advanced.

1) Basic defense: As a basic defense, all Signal clients regularly perform two types of anonymous key monitoring: (1) Alice randomly *monitors her own key* at regular intervals to remain confident the key server is consistently distributing her key, and (2) Alice randomly *monitors new connections and recent key updates* at regular intervals until she has confidence that the key she received is correct. In both cases, Alice's client requests the key (her own or that of a contact) using Tor, so that the key server does not know who is requesting the key. server to attack new connections. This approach has a usability

2) Advanced Defense: The advanced defense prevents fake key attacks using two strategies. (1) The first time Alice communicates with Bob, she requests his key anonymously from the key server using Tor to make it difficult for the key cost due to a potential delay while Alice waits to communicate with Bob while she obtains Bob's key via Tor. (2) When Alice receives a key update message from the server for Bob, she communicates directly with Bob through a third-party channel to automatically confirm that the key change is legitimate. This prevents fake key attacks on existing connections with Alice.

D. Monitoring Heuristics

These heuristics are used in combination with the above defenses to strengthen weaknesses in those defenses.

1) Mass key update monitoring: Mass key update monitoring looks for multiple key updates within a short period of time with an existing contact.

2) Isolation monitoring: If Alice is unable to connect to any of her existing contacts during the monitoring interval, it's possible a client-targeted impersonation attack is in progress.

3) Key history monitoring: To detect a rapid fake-key attack, we propose that Alice maintain a *key update history* for Bob and send it to him for validation after each key-update message she receives for him.

III. CONCLUSION

Our proposed defenses may deter an MKS from launching a fake key attack. They increase security without requiring any user interaction. Security-conscious users can still choose to perform manual key verification for additional guarantees against fake key attacks.

REFERENCES

[1] Schröder S, Huber M, Wind D, Rottermanner C. When signal hits the fan: on the usability and security of state-of-the-art secure mobile messaging. InEuropean Workshop on Usable Security. IEEE 2016 Jul.

[2] Herzberg A, Leibowitz H. Can Johnny finally encrypt?: evaluating E2E-encryption in popular IM applications. InACM Workshop on Socio-Technical Aspects in Security and Trust (STAST) 2016 Dec 5.

[3] Vaziripour E, Wu J, O'Neill M, Whitehead J, Heidbrink S, Seamons K, Zappala D. Is that you, Alice? a usability study of the authentication ceremony of secure messaging applications. InThirteenth Symposium on Usable Privacy and Security (SOUPS 2017) 2017 (pp. 29-47).

[4] Melara MS, Blankstein A, Bonneau J, Felten EW, Freedman MJ. CONIKS: Bringing Key Transparency to End Users. In24th USENIX Security Symposium (USENIX Security 15) 2015 (pp. 383-398).

[5] Tor, "Tor project," https://www.torproject.org/.

Automating the Authentication Ceremony in Signal

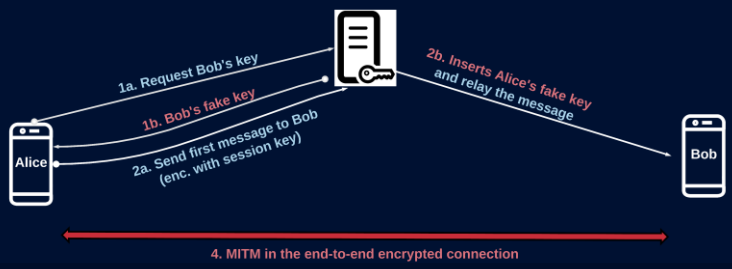
Tarun Kumar Yadav, Devashish Gosain, Daniel, Zappala, Kent Seamons
Computer Science Department, Brigham Young University

Signal Protocol

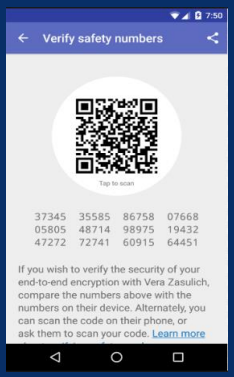
- Two-party protocol for exchanging authenticated and E2E encrypted messages with forward secrecy
- Used by billions of users

Problems

- Trusted centralized server may distribute fake keys
- Authentication ceremony is not usable, almost no one does it



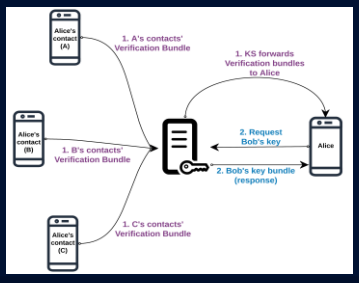
Authentication ceremony



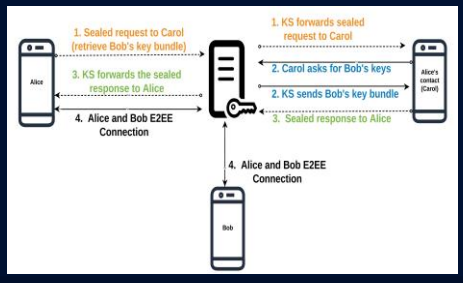
To avoid an active attacker, users should manually compare keys with their contacts when they start communicating and whenever an identity key changes

Automating the Authentication Ceremony – Approach 1: Trust Network

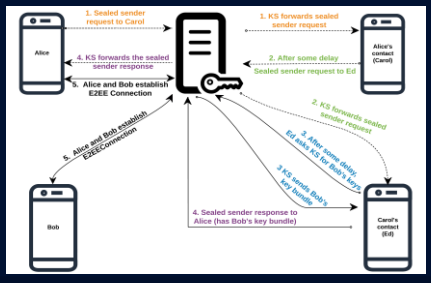
Using mutual friends



Using existing contacts to verify new contact's identity key

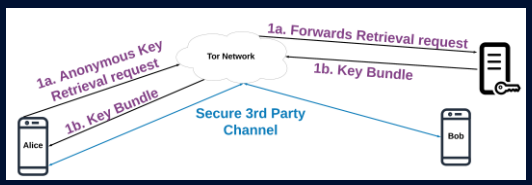


Using existing contacts to verify existing contact's key update



Approach 2: Anonymous Key Retrieval

- Retrieve contacts' keys anonymously
- Monitor own key anonymously
- 3rd Party Channel (TOR service) to verify key update



Analysis

TAXONOMY OF FAKE KEY ATTACKS AND ANALYSIS OF DEFENSES

Defense	MITM Pair Targeted New	MITM Pair Targeted Existing	MITM Client Targeted New	MITM Client Targeted Existing	Impersonation Pair Targeted New	Impersonation Pair Targeted Existing	Impersonation Client Targeted New	Impersonation Client Targeted Existing
Key Transparency (multiple-provider)	○	○	○	○	○	○	○	○
Key Transparency (single-provider)	○	○	○		○	○	○	
Trust Network (mutual contact)	●	●	●		●	●	●	
Trust Network (contact relay)	○	○	○		○	○	◑	
Anonymous Key Retrieval (basic)	○	○	○	○	○	○	○	○
Anonymous Key Retrieval (advanced)	●	●	●	●	●	●	◐	●
Mass key update monitoring (naive)				●				●
Mass key update monitoring (stealthy)				○				○† ○‡
Isolation monitoring								
Key history monitoring	◑	◑	◑	◑	◑	◑	◑	◑

● prevents the attack, ○ detects the attack, ◐ sometimes prevents the attack, ◑ sometimes detects the attack, (blank) susceptible
† impersonating to Alice, ‡ impersonating as Alice

Approach 3: An Auditable Key Directory

- CONIKS is a key management system that uses a auditable key directory
- Integrate CONIKS into Signal protocol
- Multi-provider scenario offers better detection
- Since Signal uses single provider, Signal clients audit blockchain
  - Client compares the signed tree root (STR) from the CONIKS provider to their contact's STR
  - Client audits STR when the new key is retrieved from the key server or key update happens
  - Client monitors their own keys every epoch

Conclusion

- Authentication ceremony is critical in the Signal protocol
- Prior research shows users don't verify keys manually
- Designed three automated key verification solutions
- Used threat analysis to compare how well they detect or prevent fake key attacks
- Potential to release the manual burden from billions of users

This work is supported by the NSF under Grant No. CNS- 1816929