Review for : Automatic Detection of Fake Key Attacks in Secure Messaging

Siddhant Gupta[1], Samyak Jain[2] and Advait Shrivastava[3]

[1]International Institute of Information Technology, Hyderabad[2022201037]
[2]International Institute of Information Technology, Hyderabad[2022201048]
[3]International Institute of Information Technology, Hyderabad[2022201069]

Table of Contents

1. Introduction
2. Adversary Attacks
3. Defense Approaches
4. Analysis
5. Critiques
6. Improvements/Suggestions

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Introduction

- Secure messaging apps use public key cryptography for end-to-end encryption.
- The authentication ceremony is a crucial step in verifying the public key of the recipient.
- Users can verify public keys by scanning QR codes or reading fingerprints over voice calls.
- Studies show that only 13-14 % of the users complete the authentication ceremony.
- This indicates that users don't understand the risk of fake key attacks and have difficulty finding and completing the authentication summary.
- This paper presents approaches that seek to automatically verify public keys, relieving the burden of an authentication ceremony on users by distinguishing between legitimate key changes and attacks.

Adversary Attacks

- The adversary $\mathcal{A}$ is an attacker, it can access all encrypted messages when it has control over public key distribution.
- Attacker attacks when users request the public key from the server, the attacker creates a **fake public key** and distributes it to them.

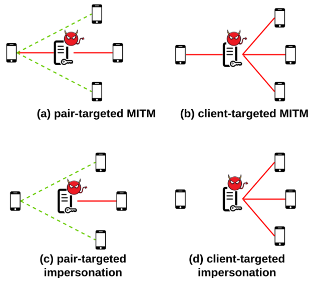

(a) pair-targeted MITM
(b) client-targeted MITM
(c) pair-targeted impersonation
(d) client-targeted impersonation

Figure: 1.0

Types of Adversary Attacks

1. **Man in the middle attack**
   - In this case, the attacker generates two fake keys.
   - Attacker gives the first key to Alice claiming that it is Bob's.
   - Similarly gives the other key to Bob claiming that it is Alice's.
   - Attacker can read/modify their messages when they communicate.
2. **Impersonation Attack**
   - An attacker can initiate this kind of attack by imitating Alice and giving one of her contacts a fake key.
   - In this scenario, contacts of Alice will think they are talking to Alice.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Defense Approaches

Types of Defense Approaches :-

KTCA

Key Transparency with Client Auditors

AKM

Anonymous Key Monitoring

KTACA

Key Transparency with Anonymous Client Auditors

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

1. Key Transparency with Client Auditors

- KTCA is an adaptation of key transparency that relies on secure messaging clients to audit the server.
- At each epoch, the server generates a **STR**(Signed Tree Root) by signing the root of the Merkle tree, along with other metadata, such as a hash of the previous epoch's STR and epoch number.
- Along with current STR, the server also provides a client with a **POI**(Proof of Inclusion) for their key. It contains the leaf node corresponding to the client's key and the hash of each sibling sub-tree along the path from the leaf node to the root of the Merkle tree
- All clients perform the following for key monitoring and server auditing : (1) monitor their own key, (2) monitor the keys for all their contacts, (3) audit the server to detect equivocation, and (4) perform short-lived attack monitoring.

2. Anonymous Key Monitoring

- In AKM, Server distributes public keys to clients directly and also supports an **AKR**(Anonymous Key Request) for key monitoring.

➥ Client monitors its key using AKR at the start of each epoch to make sure the server consistently distributes its key.

➥ Whenever a key update of client contacts is received, the client monitors the contact's key using an AKR at the beginning of each epoch from epoch $e + 1$ to $e + m$, where m is the number of monitoring requests.

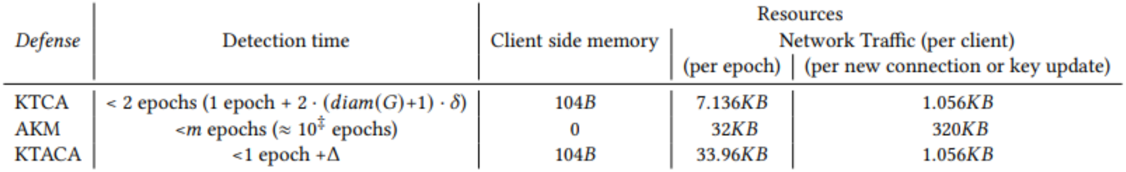
➥ Similar to KTCA, whenever a key update is received, the client performs short-lived attack monitoring to detect whether A quickly restores a correct key.

3. Key Transparency with Anonymous Client Auditors

- Similar to **KTCA**, the server maintains a Merkle tree containing the public key of all registered clients and it supports an Anonymous STR Request
- ➥ All clients verify (a) the server is publishing a linear history of STRs,(b) the STR's signature is valid, and (c) that their public key is in the tree using PoI. If STR is invalid or if the client does not receive an STR or PoI within 2 · time it considers the server under attack and disconnects.
- ➥ Clients retrieve ASR at the start of every epoch and if a client has conflicting STRs and detects an attack with a Proof of Misbehavior.
- ➥ Client receives a POI for verifying the key is in the tree for every key update and new key lookup.
- ➥ Similar to KTCA and AKM, clients perform short-lived attack monitoring to verify if an attacker quickly restores a correct key on every key update.

Analysis/ Result

Below table represent detection time, client side memory and network traffic for one epoch for all the three approaches.



Defense	Detection time	Client side memory	Resources: Network Traffic (per client) (per epoch)	Resources: Network Traffic (per client) (per new connection or key update)
KTCA	$<$ 2 epochs (1 epoch + $2 \cdot (diam(G)+1) \cdot \delta$)	$104B$	$7.136KB$	$1.056KB$
AKM	$<m$ epochs ($\approx 10^{\ddagger}$ epochs)	0	$32KB$	$320KB$
KTACA	$<$1 epoch $+\Delta$	$104B$	$33.96KB$	$1.056KB$

Figure: 2.0

Critiques

Based on Assumptions

- Paper assumes that in AKM method only short messages are sent using anonymous key monitoring, but this might fail when the length of message is large.
- In many countries, monitoring through anonymous key logging is illegal unless it is authorized by the law enforcement organizations. [1]
- Paper assumes Tor to be one of the good anonymous network providers but over the years, there have been various successful de-anonymization attacks carried out by governments and law enforcement agencies. [2]
  To read more refer report.

Critiques

Based on Technical approach

- Verification using the Merkle tree depends on the hashed value of the root. Single root hash to verify the whole tree causes a single point of failure.
- When the size of the dataset is less, the Merkle tree works very well but as the size increases, the overhead of constructing a tree also increases which might degrade the performance.
- In KTCA and KTACA methods clients wait for $2 \cdot \delta$ time and $2 \cdot \Delta$ after the beginning of an epoch before client considers this an attack, and the client disconnects from the server.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Critiques

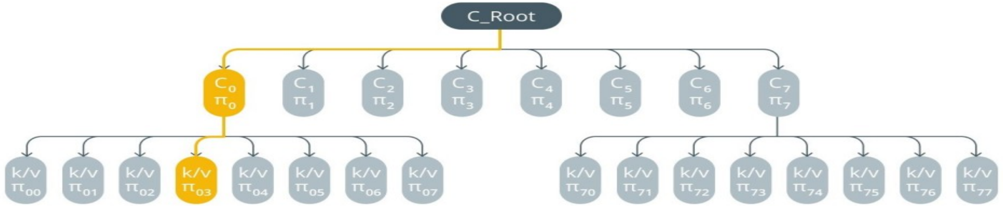
Based on Analysis/Result

- Authors tested that in every epoch 33.96KB of resources are used at the server side.[Fig 2.0]. When the client comes online after a long time, it will have to validate all the previous STRs and PoI which will cause excess traffic in the network and a lot of computation.
- Some attacks are discussed in the paper that are detectable without any PoM (proof of misbehavior).So, a dishonest client could falsely accuse a server of a fake key attack.
- Likewise, an adversary can respond to an accusation by accusing the client of making a false report. There is no way for a third party to determine which claim is correct.

Improvements/Suggestions

Use of Verkle Trees (A better alternative to Merkle trees) [3]

- Verkle trees allow you to organize a considerable quantity of data and create a brief ”witness” of each item of data or group of related pieces that someone can confirm with access to the tree’s root.
- It requires less than 150 bytes to produce a proof for a tree with a billion data points, compared to a typical binary Merkle tree’s around 1 kilobyte. This allows for efficient storage and verification of large sets of keys.



Improvements/Suggestions

Strong Consistency Required

- Today, chat applications require multiple servers for scalability and load balancing.
- In such cases, these servers must maintain a consistent copy of STRs to prevent false detection of fake key attacks.
- We suggest implementing strong consistency protocols to achieve this.

Improvements/Suggestions

Implementing Out of band channel for client-to-client communication

- Targeted clients have to manually alert their contacts about the attacks.
- We can design an out-of-band channel for client-to-client communication to relay the attack information among clients and their contacts effectively.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Improvements/Suggestions

Alternatives to digital signatures in Merkle Tree

- KTCA and KTACA use digital signatures for signing the root of the Merkle tree, and this signed data was forwarded to clients for key verification.
- Some better alternatives to digital signatures are:
    - Threshold Signatures [4]
    - BLS signatures [5]

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Improvements/Suggestions

Whitelisting

- We can prevent chat applications from being prone to fake key attacks by using the whitelisting technique.
- Chat applications can assign clients public keys from a set of trusted keys and only allow messages to be encrypted/decrypted using these.
- But if the application scales and has millions of users, the set might get exhausted.

Improvements/Suggestions

Alternative to TOR Network

- The paper implements TOR network for anonymous communication in AKM and KTACA.
- TOR provides anonymity to clients but has some drawbacks, such as slow speed, vulnerability to timing attacks, and susceptibility to Sybil attacks.
- Better alternatives are:
  - I2P Anonymous Network
  - P2P Network

Improvements/Suggestions

Use Private Information Retrieval (PIR) technique in anonymous networks

- In AKM approach, every client sends a distinct, single-key AKR(anonymous key request) for each key they monitor at the start of every epoch.
- This approach helps in anonymization from adversary by sending bulk requests.
- An alternate and better approach to prevent deanonymization while sending bulk requests is to use the PIR technique.
- Through the PIR technique, a client can retrieve information from a database in possession of a server without revealing which item it retrieved.