Automatically ensuring that only our public keys are bound to our secure messaging account

Diana A. Vasile and Daniel R. Thomas and Alastair R. Beresford

April 2021

Extended Abstract

Popular secure messaging apps have successfully deployed end-to-end encryption to billions of active users in recent years. This ensures that users of the app can communicate securely with each other: their messages are encrypted on the sending device and stay encrypted while in transit, only to be decrypted at the receiving device. This is largely made possible through the use of public key encryption and a public key infrastructure, hereafter called key server, run as an online service by the app provider. The key server stores bindings from public keys to human-readable names and provides these to the users to aid contact discovery and the establishment of secure connections with their contacts.

With standard SMS it's trivial for users to check they are talking with their intended recipients by just checking at the beginning that they have the correct number, which protects from person-in-the-middle attacks. However, this is more difficult by secure messaging because current apps support multiple devices linked to the same account, which leads to a different kind of vulnerability (*ghost-user attacks*), where a compromise at the key server allows the attacker to add to the public keys associated with individual users. Undetected, the attacker now has a path into all future conversations of the user with any friends. We show how current approaches to check that users are communicating with the intended recipients fail to cover the ghost-user attack scenario.

In our work we leverage the ubiquity of mobile devices in users' day-to-day activities and human mobility to design a gossip protocol that can automate the verification of the key to name bindings provided by the key server. This is the first key verification protocol that focuses on the detection of ghost-user attacks. Our approach combines the benefit of automation, which takes the cost of verification away from the end user, and web-of-trust style distributed key checking, which takes advantage of a diversity of network paths typically seen by mobile devices over time. By sharing the cryptographic proof of the key-to-name bindings over local networks, we can constantly check our key bindings with our contacts, providing an alternative mechanism for verifying the correct operation of the key servers deployed by the services we use. The versatility of gossip protocols allows us to consider two extensions to the main gossip protocol to expand it past the direct friends network, as well as to consider a variety of network technologies to support the gossiping between end-user devices.

We evaluate the protocol's deployability using Wi-Fi usage data from over 11 000 Android devices over a period of six years, and call detail records from over 1 million subscribers of one telecommunications provider in a city over a period of 35 days. We also created a discrete-event simulator to model a generic key server and the type of events they facilitate, as well as modelling ghost-user attacks. We use this to analyse the protocol's security and privacy in line with the threat model.

This presentation focuses on describing the gossip protocol steps needed to automatically verify the public-key bindings provided by the key server.