Exchange of Identity Attributes Using Cryptography

Omid Mogharian

Abstract

This paper proposes a method for secure exchange of attributes in a peer to peer network. It aims to gives nodes in the network the full control over shared information.

1 Introduction

In recent years, the number of deployed mobile and Internet of Things (IoT) devices have been growing drastically. The Edge Computing Networks created by these IoT/mobile devices enables participates of the network to offload their tasks to get assisted to deliver the assigned jobs. However, this introduces more security threats since it increases the real-world attack surface, especially in multi-party scenarios where not all the devices are owned by one party.

Considering how much information is generated or carried by an IoT device, organizations have good reasons to keep the data confidential. While it is needed to share information with others, currently it is hardly measured to ensure that one cannot share information without the consent of the owner. However, if the trustworthy requester of the information, is interested in mutually confirmed data by parties involved (e.g. device and owner), a system with high integrity can be designed. Such a system should guarantee a certain level of protection of each party and offers a fine-grained access management system.

The access control models designed for web applications and cloud computing mainly consists of different permissions about Read & Write. Such a model would never be satisfiable in edge computing due to the more complexity of the system and their enabled applications. This calls for fine-grained access control that addresses the questions such as who can access which devices for what reason at when and how [1].

2 Background

2.1 Asymmetric Encryption

In contrast to common symmetric encryption, asymmetric encryption [2] uses a key pair. When something is encrypted with one key it can only ever be decryption with the other key. Therefore if something decrypts with one key we know that it was encrypted with the other key from the pair.

2.2 Zero Knowledge Proof

In a Zero Knowledge Proof (ZK Proof) [3] one party proves to the others that it aware of some information without revealing the information itself. It means, for a given statement, if the statement is true, one will be convinced without learning about the detail of statement other than that the it is true.

For example, ZK Proof can be used for identification. One can generate a key pair as in 2.1 and keep one key private, named secure key (*sk*) and publish the other, called public key (*pk*). This person can prove to be the owner of *sk* by publishing a message encrypted with it. Anyone with access to *pk* can now verify that the sender knew *sk*.

2.3 Digital Signature

A group can use the technique mentioned in 2.2 to replace physical signatures [4]. If each member of the group generates a (*sk*,*pk*) pair, keeps *sk* secret and publishes *pk* to the other group members. Then each individual can prove that who is the source of a message. This is analogous to a physical signature.

2.4 End-to-End Encryption

A peer-to-peer(P2P) network set up based on 2.3 has the benefit of a secure communication as described in 2.1. If one member wants to send a message to another without any other party being able to read it, They solely have to encrypt it with the recipient's *pk*. As no one knows the recipient's *sk*, none but the owner of *sk* can read it.

3 Core Protocol

To exchange attributes about certain peers of networks, we demonstrate a P2P network with three main actors involved: Subject, Requester, and

Prover. All peers in this network communicate securely as described in 2.3 and 2.4.

In a common scenario, Requester asks for information from a Subject but requires to have a confirmation from the corresponding Prover (Figure 1). For example, an institute (Requester) wants to verify that a person or a device (Subject) works for the claimed organization (Prover).

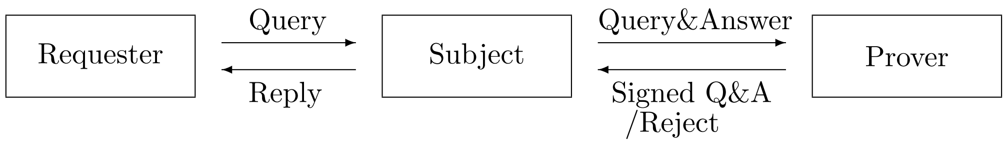

Figure 1: Interactions of actors based on the Core Protocol

Based on described interactions between three actors in Protocol 1, the Requester receives an answer that is signed twice, therefore, it is ensured that both Subject and Prover are agreed with it. Worth mentioning, this protocol built on an ultimate trust in the Prover, as the only source of truth.

Protocol 1 Core Protocol

1. Requester sends a query to Subject, determining a Prover.
2. If Subject decides to answer the query, it sends the query & answer to the Prover. Otherwise, sends a rejection.
3. Prover signs one of these messages and sends it to the Subject for the specified Requester.
   (a) Subject's message plus date of reply, If agrees with the answer.
   (b) Reject, in case decides to not to answer the query.
4. Subject can now forward the reply to the Requester.

4 Adapted Protocol for High Availability

This protocol extension allows the Subject to get answers ahead of time and store them for later reply. In theory any peer in the network even the Subject itself, can keep the answers. However for the sake of this paper, we define a separate actor called Cacher; Nodes who store the data temporally, to increase the availability of the system.

By involving Cacher in the scenario, Subject can answer the queries which are already asked even at the absence of Prover (Figure 2). This protocol is practically similar as Protocol 1, except that the Subject is the origin of the Request with the intention to store the doubly signed query, answer and the date of reply (hereafter Token) for certain Requester for a later use (Protocol 2).

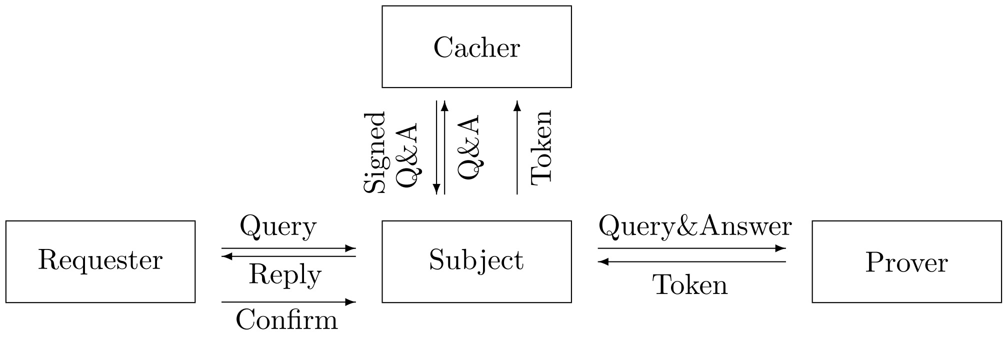

Figure 2: Interactions of actors based on Protocol 2 & Protocol 3

Protocol 2 High Availability Protocol - Storing

1. Subject decides which question needs to be answered later. Determining the Requester and the Cacher, it sends the query plus the answer to the Prover.

2. Prover signs one of these messages and sends it to the Subject for the specified Requester and Cacher.

   (a) Subject's message plus date of reply, If it agrees with the answer and the choice of Requester and Cacher.

   (b) Reject, in case decides to not to answer the query.

3. Subject can now forward the Message to the Cacher.

4. Cacher stores the Message (Token) and confirms this to the Subject.

When Requester wants information about a Subject, If a Token stored previously at a Cacher, Subject has the option to forward request to the

Cacher (Protocol 3).

Protocol 3 High Availability Protocol - Retrieval

1. Requester forms a querry and sends it to the Subject
2. If Subject decides to answer the query, it can forward the query & answer to an available Cacher. Otherwise, sends a rejection.
3. Cacher compares the Subject's answer with the available Tokens for the specified Requester. If they matched, it responses the query and answer (including original date of Prover's reply), otherwise rejects.
4. Subject can forward the message to the Requester.
5. Requester reviews the reply and can either accept or reject it due to the choice of Cacher.

An important point is that Cacher does not send Requester Prover's Token. The Incoming query & answer should be verified with available Tokens and reply will be a proof of The Subject's answer validity as a ZK Proof. This allows the Token to be more specific than the question. As an example, consider a person that is 37 years old. She can generate two tokens one stating "Subject is older than 36", and the other "Subject is younger than 38". She is able now to answer any query about her age.

Note that, in most cases Subject would be capable of storing all tokens related to itself. A Cacher, as a separate node, becomes a necessity when the size of data exceed the Subject capacity. Further more in the case of Subject absence, as describe in the next section, a Cacher node can replace the role of the Prover.

5 Protocols Benefits

As mentioned in section 3, all communications are singed and encrypted, avoiding any illegal listener to alter the exchanged data, or even screen it. As a result, peers of the network can benefit from various points, as summarized in this section.

- Subject can answer queries over data with least information revealed.
- Subject has the full control of whether a query should be answered or not.

– Subject can track which answers it has been given to whom.
– Prover gains control over information that is shared through the Protocol.
– Requester can get answers that are verified by both Subject and Prover.

In section 4, we aimed to increase the availability of a the system. By adding Protocol 2 and Protocol 3, peers of the network gain more advantages as listed below.

– Subject can get questions answered more reliably, even when Prover is offline.
– Prover does not have to be highly available.
– Prover has control over which Cacher stores the Tokens.
– Requester can receive its answer with lower latency.
– Cacher as a peer in the network can monetize its service.
– This extension enables ZK Proof over multitude of data.

6 Future Works

To reach more practicality there are number of further extensions that can aide peers of this network.

- **Privacy Assistant for Subject and Prover:** By tracking previews questions and computes the Requester's information gain, the Privacy Accountant can advice Subjects and organizations to reject the query due to being suspicious of privacy loss.
- **Negotiateiton for Sanitized Queries:** When a question is rejected, the protocol should allow the parties to search new question that Subject and Prover are willing to answer and offer it to satisfy the Requester. This could be done through Sanitation/Removal of Quasi-identifiers (K-Anonymity [5], L-Diversity [6], Differential Privacy [7])
- **Multiple Provers Confirmations:** In the case that the validation of an information should be done by multiple Provers, an extension of the protocol is needed in order to enable a chain of confirmations.

Connecting the digital identifier of the organizations(Public Keys) to real life identity of an organization(e.g. company name), is one more reason that judges the necessity of existence of an authority in the network.

References

[1] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. “Security Analysis of Emerging Smart Home Applications”. In: *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2016, 636fffdfffdfffd654. ISBN: 9781509008247. DOI: `10.1109/SP.2016.44`. URL: `http://ieeexplore.ieee.org/document/7546527/`.

[2] Clifford C. Cocks. “A note on non-secret encryption”. In: *CESG Memo* (1973).

[3] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. “The Knowledge Complexity of Interactive Proof Systems”. In: *SIAM Journal on Computing* 18.1 (Feb. 1989), pp. 186–208. DOI: `10.1137/0218012`.

[4] Anna Lysyanskaya. “Signature schemes and applications to cryptographic protocol design”. PhD thesis. Massachusetts Institute of Technology, 2002.

[5] Latanya Sweeney. “k-anonymity: A model for protecting privacy”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 557–570.

[6] Ashwin Machanavajjhala et al. “l-diversity: Privacy beyond k-anonymity”. In: *22nd International Conference on Data Engineering (ICDE’06)*. IEEE. 2006, pp. 24–24.

[7] Cynthia Dwork, Aaron Roth, et al. “The algorithmic foundations of differential privacy”. In: *Foundations and Trends in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407.