An Architecture for Accountable Anonymous Access in the Internet-of-Things Network

Yuxiang Ma*‡§, Yulei Wu†, Jingguo Ge§ and Jun Li*

*Computer Network Information Center, Chinese Academy of Sciences, Beijing, 100190, China

†College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, UK

‡University of Chinese Academy of Sciences, Beijing, 100049, China

§Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100195, China

Abstract

With the rapid development of the Internet, more and more devices are being connected to the Internet, making up the Internet-of-Things (IoT). The accountability and privacy are two important but contradictory factors to ensure the security of IoT networks. How to provide an accountable anonymous access to IoT networks is a challenging task. Since the IoT network is largely driven by services, in this paper we propose a new and efficient architecture to achieve accountable anonymous access to IoT networks based on services. In this architecture, a self-certifying identifier is proposed to efficiently identify a service. The efficiency and overhead of the proposed architecture are evaluated by virtue of the real trace collected from an Internet service provider. The experimental results show that the proposed architecture could efficiently balance accountability and privacy with acceptable overheads.

Index Terms

Accountability, Anonymous, Internet-of-Things, Network Performance.

I. Introduction

With the development of the Internet, more and more devices (e.g., mobile phones, laptops, wearable devices, and smart appliances) and vehicles are being connected to the Internet, leading to the era of Internet-of-Things (IoT). Under the IoT paradigm, many of the objects that surround us will be on the network in one form or another [1]. IoT brings convenience to people, whilst causing hidden dangers. It might reveal privacy about users' behaviors. For example, when an attacker learns that people in a house do not use the network for a certain period of time, it may be because the family in the house is on vacation and currently away from home. If an attacker finds that the user activated the water heater (smart appliance) through the Internet, it may show that people living in the house may be on their way back to home. In addition, there are a large number of sites collecting a variety of privacy information regardless of users' objections [2]–[4]. Therefore, network users have shown strong concerns about the privacy protection when using the networks [5]. They always intend to hide their true identity to protect

their privacy in cyberspace (i.e., anonymous access) [6], because they do not want their identity information to be acquired by intermediate nodes, where attackers may appear, during packet transmission.

However, when network users use the devices connected to the Internet, the packets they send to the receivers (e.g., E-commerce website, data center, or other devices) will include their IP addresses. The IP address as an identity identifier, may expose their names, attributes, locations and other information [7]. Therefore, how to protect the user's privacy in this process is a challenging issue. The key to solving this problem is to effectively hide the user's IP address (i.e., source address) [6], [8]. Currently, there exist several projects that allow network users to use the network anonymously, e.g., Tor [9]. The main drawback of Tor is that we might never find out who would be responsible for illegal behaviors.

In contrast to the users' privacy, the trustworthiness of users is what the network and receivers (e.g., service providers) are most concerned about, where the user traffic is required to be accountable (e.g., obtaining the packet source address) in order to stop in-progress attacks and prevent future harmful actions [8], [10]. The anonymity and the accountability are therefore two conflicting factors. How to achieve accountable anonymous access in IoT networks becomes a hard problem.

The rapid development of networks has witnessed a line-speed growth of Internet traffic, therefore we need a more efficient way to achieve the balance of accountability and anonymity in the network. In addition, IoT offers a wide variety of services (e.g., communications, data storage, cloud computing, health management, and environmental monitoring), and the IoT network is also largely driven by services [1], [11], [12]. Considering these factors, in this paper, we propose a new architecture to balance accountability and privacy based on services. By aggregating packets and flows belonging to an identical service, the processing overhead in the system can be reduced (compared to other ways, e.g., packet-based or flow-based). The definition of "service" can be found in Section III.A. In addition, the proposed architecture is designed as a protocol at the network layer, because in many applications security at the network layer has many advantages over that provided in the other layers of the protocol stack [13], [14]. Example applications include access control, quality of service (QoS) guarantee, and security management [15]. This is because the network layer forwards packets without dealing with the packet payload, and intermediate nodes can check or verify (but not modify) the packet at any hop, which will facilitate fast problem detection. In summary, the main contributions of this paper are as follows:

- This paper proposes a new architecture, as a network layer protocol, for balancing accountability and privacy in the IoT network based on services.
- A new identifier, i.e., Service ID (SID) is designed to identify a user's behavior of connecting to the Internet and sending packets for a specific purpose. An SID can contain one flow or multiple flows. The verification information needs therefore only to be generated against each service, rather than each packet or each flow.
- A delegate is designed to prevent the client's identity information (i.e., IP address) from being gained by intermediate nodes or receivers. At the same time, the delegate can vouch for their clients' behavior [1].

[1]The terms *client* and *user* are used interchangeably in this paper. Both of them refer to people who use the Internet.

- We analyze the potential issues in the real world deployment, and propose the feasible solutions for different scenarios. We also analyze the security concerns of the proposed architecture.
- A real trace collected from an Internet service provider is used to evaluate the efficiency and overhead of the architecture. The results show that the proposed architecture can efficiently provide accountable anonymous access in the network; in addition, both the bandwidth usage and storage overhead in the process of transmitting data and caching verifications, respectively, in the network are acceptable.

The rest of the paper is organized as follows. The related work is provided in Section II. Section III describes the problem for this study, including definitions and use scenarios, and Section IV introduces the proposed architecture. We analyze how to deploy the proposed architecture in Section V. The security issues in real world deployment is discussed in Section VI, and the performance evaluation of the proposed architecture is shown in Section VII. Finally, we conclude this study in Section VIII.

II. Related Work

In this section, we provide details of the existing work related to our study in terms of accountability and privacy and self-certifying address.

A. Accountability and Privacy

Accountability and privacy have been considered as two important factors to ensure the success of the current Internet. Several existing work have been reported to address the problems raised by either accountability only or privacy protection only. For example, Accountable Internet Protocol (AIP) [16] was the most popular solution whose primary objective is the accountability. In AIP, each host maintained a small cache to store the hashes of packets sent very recently. The first-hop router verified the packet by challenging the source with the hash of a packet it sent. If not being successfully verified, the packet will be dropped by the router and a verification packet will be sent to the source. When a packet traversed across the boundary of an accountability domain, the previous accountability domain where the packets come from must decide whether the source address was valid. AIP can stop an attack by using the Shutoff protocol, where a victim host sends an explicit shutoff instruction to the host who generates such traffic.

In addition to the related work considering the accountability only, several existing solutions were conducted for privacy protection only. For instance, Tor [9] is one of the most famous projects to protect user privacy. In its onion routing, instead of making socket connections directly to a responding machine, Tor makes connections through a sequence of machines called onion routers. For example, if Alice wants to use the Tor network to communicate with Bob, the process can be described as follows:

1) The Alice's Tor client accesses a directory server and obtains a list of Tor nodes.
2) The Alice's Tor client picks up a random path to destination server. The path needs to go through three Tor nodes.

If at a later time, Alice visits another site or communicates with other people, her Tor client will select another random path.

The Tor protocol (or Tor network) can effectively protect user privacy. The main drawback is very obvious that we might never find out who would be responsible for the illegal behaviour. Furthermore, Lightweight Anonymity and Privacy (LAP) [17] was another popular contribution for privacy protection. It attempted to enhance anonymity by obscuring the topological location of an end-user based on two building blocks: packet-carried forwarding state and forwarding-state encryption. LAP allowed each packet to carry its own forwarding state, thus the accountability domain can determine the next hop from the packet itself instead of keeping per-flow state locally. In each accountability domain, a private key was used to encrypt/decrypt the forwarding information in the packet header, preventing other accountability domains from getting it.

Some IPv6-based communication, as envisioned in the IoT scenarios allowed individual host to be multi-addressed or to change addresses over time for privacy protection [18].

Currently, there is no efficient architecture for balancing accountability and privacy in the IoT network. In IoT, when a user sends data to achieve a certain purpose (e.g., sending instructions to several devices/sensors, or backing up photos to several devices), the data may be sent to multiple receivers, which will generate multiple flows. In addition, if the data size is very large, it is difficult to protect users' privacy or perform accountability based on its content or the associated network flows. Therefore, it is a challenging study to achieve accountability and privacy in IoT networks.

B. Self-certifying Address

It is worth noting that, in recent years many research efforts have been made on new types of network addresses, most of which emphasized self-certifying [19], [20]; this has been considered to be the trends in the development of communication networks. The self-certifying identifiers have been widely adopted in a variety of distributed systems. For example, in [21], self-certifying was established by binding three different entities: Real-World Identity (RWI), name, and public key. To achieve the goal of self-certifying name in the AIP [16], the name of an object was considered as the public key or the hash of the public key that corresponds to that object. In the NSF-funded MobilityFirst project [22], a self-certifying Globally Unique Identifier (GUID) was derived simply as a one-way hash of a public key, which was then used to support the authentication and security throughout the project. In Host Identity Protocol (HIP) [23], which has been standardized at the IETF, the host identifier was also derived by a fixed-size hash of the public key.

III. Problem Overview and Background

In this section, we describe the definition of privacy protection, accountability, and service, followed by the scenarios, the security assumptions, and the threat model to be considered in this study.

A. Definition

In order to meet the characteristics of IoT networks, a new architecture will be proposed for balancing privacy and accountability based on services. As introduced in Section I, the proposed architecture is designed as a protocol

at the network layer. In this study, the definitions of privacy and accountability are given below, which are consistent with many existing studies [6], [8], [24].

Definition 1: Privacy protection is to ensure that when users send packets, intermediate nodes (e.g., routers) cannot gain who is the sender according to the received packets.

Definition 2: Accountability means that when malicious behaviors happen we can find the sender who should be responsible for this behavior.

In general, accountability means that the network is recordable and traceable, therefore making it liable to those communication principles for its actions. Together with some suitable punishments or laws in the real world, it will prevent a number of attacks from being mounted [25].

The term, **service**, in this paper refers to the client requesting the services provided by the service provider, such as visiting a website for news, downloading music, or sending a task to a cloud computing center. This concept has been widely used in existing studies [26]. To efficiently identify a service, we propose a new identifier, i.e., service ID (SID). The **SID** is a fixed-length, location-independent name for a particular service. Each SID is mapped to a service at one time.

B. Scenario

In this study, we investigate the problem of balancing accountability and privacy in IoT networks. We consider a scenario where a user wants to visit a website at home, and he does not want the node in the transmission path to know his behavior. That is, the user does not want other nodes know his IP address when the data is being forwarded by them. Because the IP address can expose the user's location, attributes, and other information [7]. At the same time, if the user is a malicious user, and use anonymous tools such as Tor network to attack servers in the website, the service provider (the website company) and network administrator (e.g., Internet service provider or government) will not be able to find the attacker. Therefore, there is a need for an architecture that can not only protect the privacy of users, but also take measures to reduce the risk of attack and even accountability when needed.

We consider another simple scenario. Many families install cameras to monitor their houses. We assume that family member A is in the office and family member B is on the subway. When the camera in the house transmits video images to multiple end devices (e.g., A's computer and B's smartphone), we do not want the intermediate nodes (even service providers or equipment vendors) to know the house's location. Thus, what we need to do is to hide the camera's IP address when it is transmitting data. At this point, we need a solution to protect the user's identity, address and other information. We hope this architecture can make the behavior of transmitting data to multiple end devices as a service (no matter how many flows are generated).

More and more appliances, such as refrigerators, air conditioners, water heaters become smart appliances, and are connected to the Internet. We can remotely control the appliance by turning it on or off. When we are on the way back to home, we hope that air purifiers, water heaters, and smart appliances to start working before we get home. If an attacker learns that we are sending such instructions to smart appliances, and knows the IP address of

the device (where the instructions come from). The attacker will gain the user's location, and estimate when the user arrives home. Privacy protection is therefore very important in the IoT.

The proposed architecture can be used in a variety of environments, e.g., users transmitting data using wired networks or WiFi indoors (at home or office), or users using mobile communication networks (such as 4G and 5G networks) outdoors. More details can be found in Section V.

C. Security Assumption

We assume that the third party (i.e., delegate) is Honest-but-Curious. This assumption has been widely used in many related studies [27]–[29]. The delegate executes the task specified by the protocol, but is curious about any information that may be disclosed in this process. In addition, the delegate may be attacked and controlled by attackers. We will provide a way of reducing the negative effects of the third party when being no longer trusted or being attacked in Section VI.A.

D. Threat Model

In addition to the risk of malicious delegates, the architecture also faces a variety of security threats. Threats may come from the client/user, the service provider (e.g., computing center or websites), and the intermediate nodes (e.g., routers). In particular, the client might send malicious registration requests to the delegate. Client might also attack the service provider by sending malicious content to the servers at the service provider. The service provider can attack the client by providing invalid services or even sending malicious content to their clients. Service providers or the intermediate nodes might send a large number of invalid and malicious verify requests to the delegate, and the intermediate nodes may steal an SID, add it to the header of the packet, and pretend to be the content from a secure sender/client. In addition, the intermediate node may also initiate a replay attack using some packets of a content and the corresponding SID. Therefore, security threats may appear in the processes of client sending the data to service provider, the service provider verifying the service request, and the service provider returning the results back to client. The attack type includes DDoS attack, flood attack, replay attack, and identity theft. In the proposed architecture, we will address these threats.

IV. The Proposed Architecture

In this section, we will first give an overview of the proposed architecture. Then we will illustrate how to generate the self-certifying identifiers (including Service ID and Client ID), followed by the details of each process in the proposed architecture.

A. An Overview of the Proposed Architecture

In the proposed architecture, there are three main participants: client/user, delegate (the independent third party), and service provider (e.g., website, data center, or cloud computing center). Fig. 1 shows the relationship between these participants in the proposed architecture.

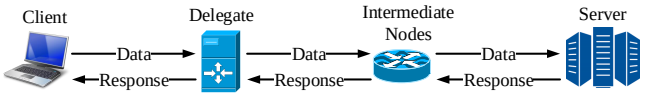


Fig. 1: The relationship between participants in the proposed architecture

Before the data is sent to the service provider, the client needs to interact with the delegate. This process is mainly prepared for data transmission. In this process, delegates will verify the authenticity and validity of the clients' requests. More details on this process will be discussed in Section IV.C. After the client interacts with the delegate, it can start to issue data. Upon receiving the data from the client, the delegate replaces the source address in the packet header and hide the client's address. More details on this process will be discussed in Section IV.D. Any node that receives the packet of the data, including delegates, servers, and intermediate nodes (e.g., routers), can challenge the validity of the sending data. More details on this process can be found in Section IV.E.

B. How to generate the self-certifying identifiers

In the proposed architecture, we introduce two self-certifying identifiers, i.e., Service ID (CID) and Client ID (CID).

When a client needs to use the service provider's servers for computing or other services, the client will generate an SID. The definitions of service and SID can be found in Section III.A. Because in the IoT network, a computing task or a request may be sent to multiple receivers (servers), an SID can correspond to multiple flows. Since the SID involves a lot of information, and is different from a flow, therefore it is actually more than just a simple hash. Its structure is shown as follows:

$$\text{SID} = H(Timestamp \| K^{+}_{client})$$

where *H* is a cryptographically secure hash function and $\|$ stands for concatenation. *Timestamp* is the start time of the service request, which is included in the SID to distinguish different services from the same sender. K^{+}_{client} is the public key of the sender, which is used to distinguish different senders if they share the same delegate. When a service is still going on, but the delegate receives the same SID of the service from another client, the delegate will notify the later client to regenerate another SID. In addition, K^{+}_{client} plays an important role in the self-certifying process [22], [30]. In this paper, we assume that the private key of client (i.e., K^{-}_{client}) is controlled by the real client itself.

In addition to the SID, we use the CID to identify clients. It can be generated by the hash of the client's public key. If a client moves to a new location, or if the IP address of the client is changed, its CID will remain unchanged. This also conforms to the idea of ID/Locator separation, which has been considered as a trend in the future networks [22], [30], [31]. Clients can periodically change their CIDs (for privacy protection purpose) by changing its public key.

C. Registration

As mentioned in Section III.C, delegates are honest-but-curious. In this paper, we mainly consider that the proposed architecture (protocol) is installed on the border router. In other words, the data within a network domain will be forwarded to its delegate (border router) first before being forwarded to other network domains. What happens if the data packet will not be forwarded to the delegate, because a server in the network domain is installed the protocol and can thus act as the delegate? We will briefly introduce this situation in Section V.A.

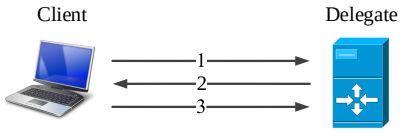


Fig. 2: The process of registration with the delegate

In the proposed architecture, clients need to register with delegates before transmitting data. The purpose of the registration process is to let the delegate know that the client is about to use the privacy protection service it provides. Fig. 2 shows this registration process, which is elaborated as follows:

1) The client sends a request to the delegate to tell that it wants to use the delegate's service to hide its IP address. At the same time, the client sends the SID of the service to the delegate.
2) If the delegate accepts the request, it will verify the authenticity of the SID. The delegate sends a random nonce N to the client who declares itself to be the sender.
3) The sender encrypts the random nonce N using its private key, and then sends the encrypted N along with the Timestamp, which is used by the sender in the registration process for this service, to the delegate.

The delegate decrypts the encrypted random number N using the public key of the client, and checks whether the random number N is what it sent to the sender. The verifier can then calculate whether the hash of the Timestamp, and client's public key is equal to SID (i.e., $H(Timestamp \| K^{+}_{client}) = \text{SID}$). In this process, the public key of the client can be obtained from Public Key Infrastructure (PKI). If the delegate could not obtain the public key of the client, the client can send its own public key to the delegate in Step 1).

Through these steps in the registration process, the delegate will verify whether the SID is a fake or stolen identifier, and whether the sender is the claimed client. This is the process of self-certifying.

Delegates can maintain a service mapping table to record the SID and the corresponding user information. Table I shows the main information that needs to be recorded. In addition to directly record the clients' IP addresses, delegates can use CIDs to record users' information.

D. Data Transmission

After the registration with the delegate, the client can start sending data. It should be noted that the packet header will always carry the SID during the data transmission process. The working principle of this process is shown in Fig. 3, including the following four steps:

TABLE I: The service mapping table

Service ID	Client Information
SID-1	202.196.96.***
SID-2	CID-1
SID-3	CID-2
.....	.....

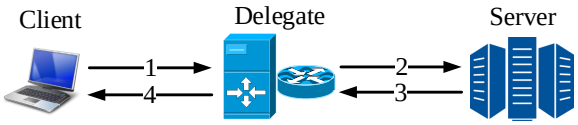


Fig. 3: The processes of data transmission

1) A client sends data to the delegate. In this step, the packet source address is the client's IP address, and the destination address is the delegate's IP address. When the delegate receives the data, it sets its own address as the source address, and the destination address is the IP address of the server (service provider). During this process, the SID in the packet header remains the same.
2) The delegate forwards data to the server. In this process, because the source address is the IP address of the delegate, the service provider will not know where the data comes from, and who is the real owner of the data.
3) The service provider responds to the user's request and returns the content (results) to the delegate.
4) The delegate looks up the service mapping table. A delegate can find the receiver of the content according to the SID. Then, the delegate can forward the content to the client.

Clients can generate a set of key pairs by themselves (i.e., $K^{+}_{sender}/K^{-}_{sender}$), and then send the public key to the service provider along with the data. The service provider uses K^{+}_{sender} to encrypt the content which will send to the client (who request the content). Therefore, only the client can use the private key to decrypt the content. It is worth noting that the key pair used to encrypt the data (i.e., K^{+}_{sender}) is different from the one (i.e., K^{+}_{client}) used to generate the SID mentioned above. K^{+}_{client} is used for authentication and can be used for long periods of time. But K^{+}_{sender} is generated by the client itself, and only used for data encryption. If a client uses a key pair for a long time, the service provider may analyze the user's identity based on the user's behavior. Because the service provider may guess that the same public key (K^{+}_{sender}) corresponds to a user, the client can periodically generate key pairs for result encryption, and even use different key pairs for different services. Since the data and the results are encrypted by the client and the service provider, respectively, a delegate cannot know the specific content it forwards. Delegates only know that there exists a communication between a client and a receiver.

E. How to Verify A Service and Stop An Attack

During the transmission process, the verifier (e.g., intermediate nodes and the receiver) can challenge whether the data is valid, to make sure it is not an attack. The verifier can send a verify request to the delegate to ask whether the delegate has forwarded something. The process can be expressed as:

$$\begin{aligned}\text{Verify}(\textit{service}) = SID\|P_{header} \\ \|MAC_{K_{verify}}(SID\|P_{header})\end{aligned}$$

where MAC denotes the message authentication code, and $MAC_{K_{verify}}$ is included to make sure the request is what the verifier wants. When the verify request arrives, delegates need to check two things, named two-step checking: 1) SID exists in the delegate (i.e., whether the service is present), and 2) the service has not been stopped (shutoff). If both checks pass, it means that there is a client who sends packets to the server/receiver. Then the delegate replies VERIFIED to the verifier, showing that this flow is vouched by some delegate even the sender is unknown by public.

We can set the verification interval to adjust the frequency of verification. If a service/flow has just been verified, it will be added to the whitelist. After the expiration time, the verifier can send verify message again if the service/flow is still alive. The evaluation of the verify rate and the impact of verification interval on the whitelist size can be found in Section VII.C.

A victim of cyber attack can stop malicious or harmful flows by Shutoff. In the proposed architecture, a shutoff message is sent to delegates. Typically, receivers issue the shutoff message when they learn the flow or the content is malicious. This process is as follows:

$$\begin{aligned}\text{Shutoff}(\textit{service}) =& SID\|P_{header}\|duration \\ &\|MAC_{K_{victim}}(SID\|P_{header}\|duration)\end{aligned}$$

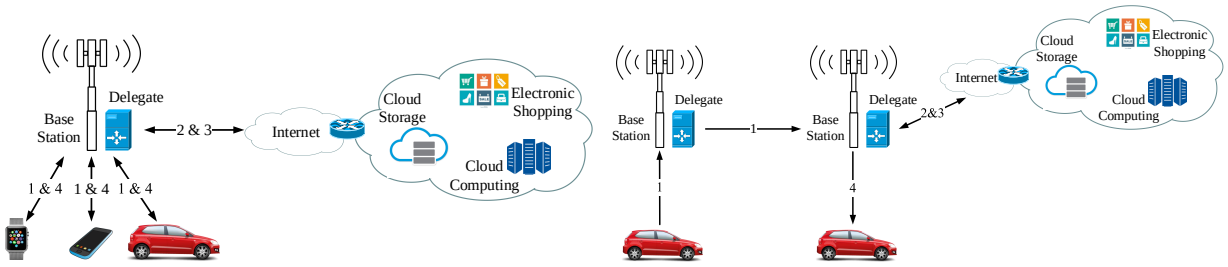
where the *duration* is the time the victim wants the delegate to stop verifying the flows sent by the sender.

V. Analysis in Real-World Deployment

The proposed architecture can be used in different scenarios. In this section, we will first introduce the location of delegates, then we will take wired networks and wireless (mobile) networks as examples to show how our architecture works in different scenarios.

A. The Location of Delegates

In this paper, we assume that the proposed architecture (protocol) is installed on border routers, therefore the data sent by clients will go through the delegate. If a delegate is a server in the network domain, the client needs to send certain information to the delegate so that the delegate can respond to the verification requests from verifiers. For example, after sending data to the receiver, the client can send the hash of the data to the delegate. If a verifier is the receiver, it can use the hash of the data to verify whether the client (sender) is vouched by the delegate. Alternatively, the client can send a hash of packets (he just sent) to the delegate. Verifiers can send the hash of any packet in the content to the delegate to verify whether the forwarded data is malicious.



(a) When the user move slowly in a network domain (b) When the user move from one network domain to another

Fig. 4: Users use network services provided by the base station

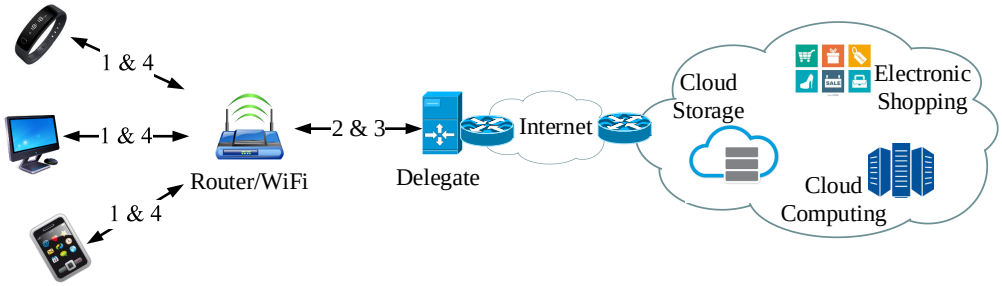


Fig. 5: The processes of using the proposed architecture in wired networks

B. Use the Proposed Architecture indoors

When users connect to the network at home or office, they might use computers, laptops over wired networks, or use cell phones, wearable devices (e.g., smart bracelet) and other devices (mobile terminal) through wireless networks (e.g., WiFi). Fig. 4 shows how users use the proposed architecture when they are indoors. The Steps 1) – 4) in Fig. 4 corresponds to the four steps in Fig. 3. More details on these steps can be found in Section IV.D.

In this scenario, the router in user's house or office, as the first hop for devices to connect to the Internet, will record devices' identifiers (i.e., CIDs) and SIDs when users use laptops, cell phones, smart bracelets, or other devices to connect to the Internet. The router, instead of mobile devices, is responsible for responding to the verification requests from delegates. The purpose of this design is to save the power consumption of devices because many mobile devices have limited power. Thus, more verification work should be done by the router in the house. At the same time, the router will be responsible for the user's behavior. When the router receives a notification from the delegate that the device connected to it has malicious behavior, the router should prevent malicious users from continuing sending data, otherwise the delegate will stop verifying all the data from this router (see Section IV.E).

C. Use the Proposed Architecture outdoors

Fig. 5 shows how to use the proposed architecture when users are using mobile communication networks, such as 4G or 5G networks. The Steps 1) – 4) in Fig. 5 also corresponds to the four steps in Fig. 3 (see Section IV.D). In this scenario, the delegate can be located at the base station. When the base station forwards the user's data

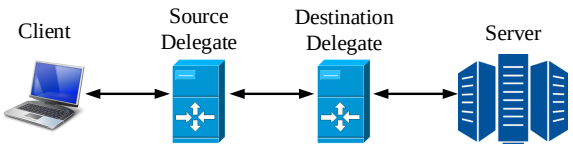


Fig. 6: The double-delegate paradigm

(packets), it will protect the privacy of the user and, at the same time, be responsible for the user's behavior. When an attack occurs, the delegate at the base station will stop providing services to the malicious user, thereby preventing malicious actions timely.

Fig. 5(a) shows the situation when the user moves slowly. When users are moving fast (e.g., vehicles), they can choose delegates located in the incoming signal coverage area to provide services for themselves, according to their direction and speed of movement. This scenario is shown in Fig. 5(b). This approach benefits from allowing users to choose delegate on their own, which will reduce the frequency of base station switching and improve user experience.

VI. Security Analysis

In this section, a list of potential problems to be considered in real-world deployment of the proposed architecture and their countermeasures are analyzed.

A. The Trustiness of Delegates

The delegate has three basic responsibilities in the proposed architecture: 1) protecting the privacy of their clients, 2) verifying the validity of the service via cached verification information, and realizing the accountability to their clients, and 3) dropping invalid or malicious packets if needed.

The delegate is assumed to be Honest-but-Curious in the proposed design. However, what is harmful if the delegate is no longer trusted? It may release client privacy, may not perform verification function or respond with incorrect verification results. Thus, they cannot take the role of accountability, and the clients can decide to replace the delegate and take legal measures to punish delegates [32].

If the client (i.e., sender) and the receiver (e.g., service provider) are in different network domains, and they have different delegates, we can use a double-delegate paradigm to improve the performance of the architecture of protecting the client privacy. Fig. 6 shows the relationship between different entities. The delegate close to the client is called the source delegate, and the one close to the service provider (i.e., destination domain) is called the destination delegate. A client can specify source delegate and destination delegate for the data transfer process.

Before the data is transmitted, the client will interact with the two delegates, respectively. This process is similar to what we described in Section IV.C, but the information sent by the client is different. In this process, a client will send the SID, client IP address (or CID), and destination delegate information (e.g., IP address) to the source delegate. Meanwhile, the client will send the SID, source delegate address, and receiver address to the destination delegate. Both the source delegate and the destination delegate will record the information sent by the client.

If we use the double-delegate paradigm design in the real world, only when the two delegates (i.e., both the source delegate and the destination delegate) are attacked and co-operated with each other, the user's behavior could have the risk of leakage. It should be noted that, even if the delegate was attacked and leaked clients' privacy (behavior), the content the client has sent will not be gained. This is because the data is encrypted during the transmission process, and the delegate cannot decrypt it (see Section IV.D). What the attacker can know is that the user has established a communication with the receiver.

B. Attacks from Clients

Malicious clients might send a flood of requests to a delegate. The delegate can take some policies to prevent this attack, such as using bloom filters or setting a maximum number of requests received per second from the same host. This behavior can be observed. For example, a delegate receives lots of requests from a sender, but there is no data issued to the delegate.

C. Attacks from Receivers

A receiver may send a large number of invalid and malicious verify requests to delegates. If a delegate discovers that the request from a verifier is too frequent and abnormal, the delegate can drop verify requests from the verifier directly, and even punish it if the attacker is a service provider. A receiver (e.g., service provider) may attack clients by providing invalid services to users or even sending malicious content to their clients. If many clients report a service provider's malicious behavior, delegates can tell new users about the service provider's reputation.

D. Attacks from Intermediate Nodes

Once a client successfully completes the service registration, an intermediate node (attacker) might replay any packet from that service/content to other nodes in the network. In this case, the victim will send shutoff message to inform the delegate which SID has been used by the attacker. When subsequent packets of the malicious content (used for attack) enter the delegate, the delegates will stop forwarding them. For attacks that occur between delegate and receivers, delegate will stop vouching for that content and the routers will stop forwarding that content, and the attack will eventually be blocked. It should be noted that, in the proposed architecture, intermediate nodes or attackers cannot use someone else's SID, and place it in the packet header of a malicious content. That is because the stolen identifier cannot pass the self-certifying. The process of self-certifying has been described in Section IV.C.

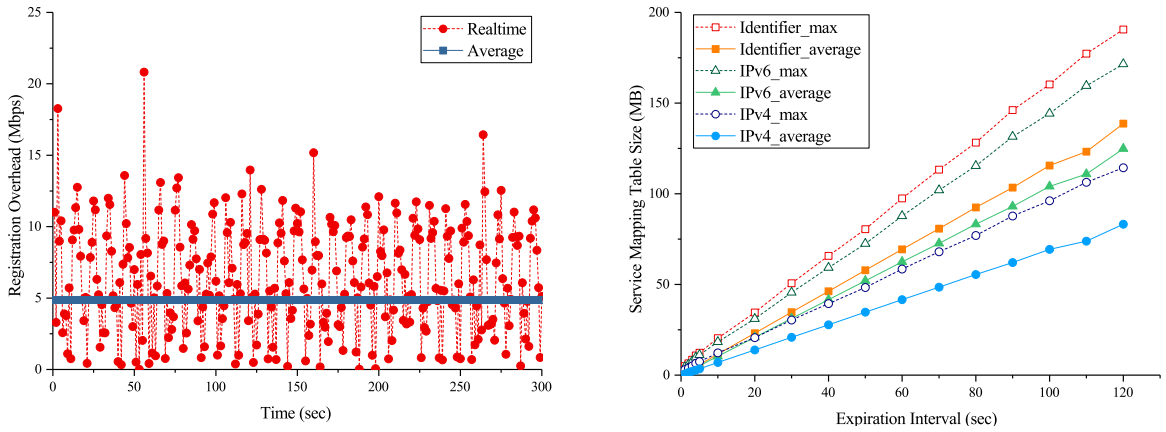
VII. Performance Evaluation

The NetFlow date traced from a border router of China Science and Technology Network (CSTNet), an Internet service provider in China, is used to evaluate the cost and feasibility of the proposed architecture. This fifteen minutes' trace was taken on June 3, 2016 from 15:15 to 15:30, containing 36.57 million flows.

The cost of bandwidth in the registration process and the required storage in the delegate to cache service mapping table are evaluated. Because in the proposed architecture we need to add the SID in the packet header,

therefore the data transmission process encounters additional overhead. The increased bandwidth usage during the data transmission process is evaluated in this section. In addition, the verify rate and the whitelist size in intermediate nodes (e.g., routers and verifiers) are calculated.

In this section, we assume that one service contains only one flow. In other words, each flow corresponds to a service in the statistics to facilitate depicting figures [33]. This consideration reflects the worst case on the overhead of the proposed architecture. Since in realistic scenarios a service may contain multiple flows, i.e., a data might be sent to different servers, which will generate many flows [34].



(a) The required bandwidth of sending SID

(b) The service mapping table size in delegate

Fig. 7: The overhead in the registration process

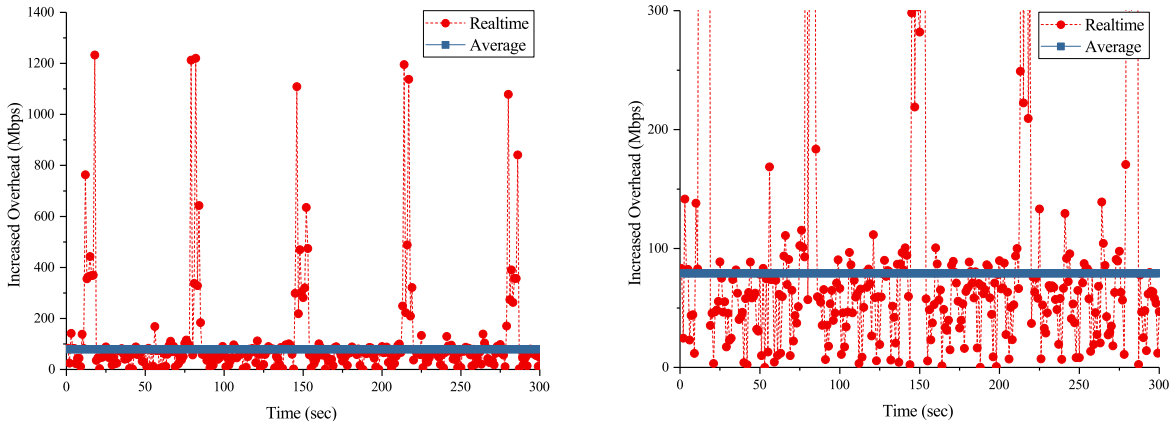
A. Overhead in the Registration Process

In this section, we evaluate the bandwidth required by clients to send information to delegates. We calculate how many flows per second, and then obtain the required bandwidth according to the size of each SID. Since SHA-1 (Secure Hash Algorithm) produces 20-byte digests, it needs 20 bytes to send an SID in the proposed architecture. Fig. 7(a) depicts the required bandwidth for clients to send SIDs to delegates against the time horizon from 15:20 to 15:25 (300 seconds). We can observe that the maximum required bandwidth for sending SIDs in our architecture is 20.81 Mbps, and the average required bandwidth is 4.86 Mbps.

In addition to the bandwidth overhead, the delegate needs to maintain the service mapping table. Fig. 7(b) shows the storage space required for different service durations. If the SIDs and client information are calculated according to the size of the self-certifying identifier (i.e., 20 bytes for each identifier), and services will last for 60 s, 97.47 MB is enough to store the service mapping table. If delegates record the IPv6 address of the clients and the service providers, instead of their self-certifying identifiers, it only needs 171.49 MB (120 s) to record these entries, and 114.33 MB (120 s) is enough if clients and service providers use IPv4 addresses.

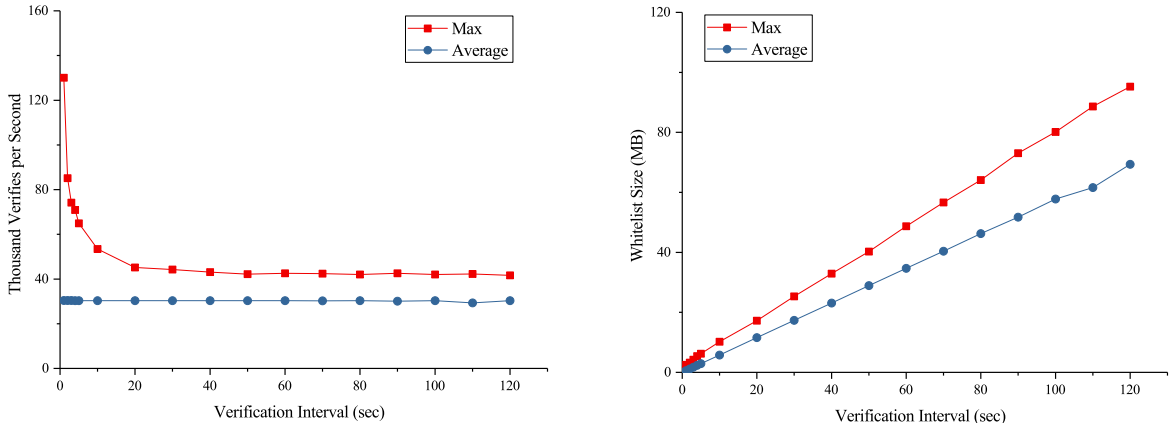
B. Overhead in the Data Transfer Process

Because in the proposed architecture we have added the SID in the packet header, therefore the data transmission process requires additional overhead. We calculate the number of packets per second and then evaluate the increased bandwidth. Fig. 8 shows the results. Fig. 8(b) is the sub-figure of Fig. 8(a) with the scale in y-axis from 0 to 300 Mbps to make the figure clearer. In Fig. 8(a), we can observe that the maximum increased bandwidth for sending data in our architecture is 1232.74 Mbps. But it is just instantaneous bandwidth usage, and the dataset is collected from a border router of an ISP. In Fig. 8(b), we can observe that the average increased bandwidth is 79.07 Mbps. This overhead can be accepted for a backbone network.



(a) The increased bandwidth in data transmission (b) The sub-figure of (a)

Fig. 8: The overhead in the data transfer process



(a) The verification rate at clients and delegates (b) The whitelist size in intermediate nodes

Fig. 9: The overhead in the verify process

C. Overhead in the Verify Process

The verification rate is calculated by analyzing the trace data. We first do the statistics on the max and average number of verified flows within a given time, i.e., the verification interval. Then, we divide the interval into seconds. The result reflects the verifying pressure a delegate should withstand. Fig. 9(a) shows how many verifications per second delegates can serve against the verification intervals. From the figure, we can find that the pressure of a delegate is associated with the verification interval. After 20 s, with the increase in the verification interval, the maximum verification rate does not reduce significantly. This is because the impact of bursty traffic will gradually decrease along with the increase in the time interval, so that the number of verifications per second is gradually stable and is close to the average level.

As introduced in Section IV.E, a verifier maintains a whitelist for recording recently-validated flows. Therefore, 20 bytes (i.e., an SID) are required for each entry in the whitelist. The number of flows in a given verification interval is counted, and the whitelist size is then calculated and shown in Fig. 9(b). From the figure, we can observe that in the proposed architecture, if the verification interval is set to be 20 s, 17.24 MB is enough to store all whitelist entries. Even when the verification interval goes up to 120 s, the average size of the whitelist in the proposed architecture requires only 69.35 MB.

VIII. Conclusion

In this paper, we have proposed a new architecture to balance accountability and privacy for IoT networks based on services. A new identifier (i.e., service ID) has been used to identify and distinguish different services. How to improve the security and reliability of the system in real-world deployment has been discussed. Using the real NetFlow data collected from an Internet service provider, the feasibility of the proposed architecture has been evaluated. The proposed architecture is designed based on TCP/IP, and can therefore be readily applied to the current Internet.

References

[1] J Gubbi, R Buyya, S Marusic, and M Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems*, 29(7), 1645-1660, 2013.

[2] C Wang, B Zhang, K Ren, JM Roveda, CW Chen, and Z Xu, "A privacy-aware cloud-assisted healthcare monitoring system via compressive sensing," in *Proc. IEEE INFOCOM'14*, 2130-2138, 2014.

[3] N Xia, HH Song, Y Liao, M Iliofotou, A Nucci, ZL Zhang, and A Kuzmanovic, "Mosaic: Quantifying privacy leakage in mobile networks," in *Proc. ACM SIGCOMM'13*, 279-290, 2013.

[4] Z Qin, Y Yang, T Yu, I Khalil, X Xiao, and K Ren, "Heavy Hitter Estimation over Set-Valued Data with Local Differential Privacy," in *Proc. ACM CCS'16*, 192-203, 2016.

[5] JR Mayer and JC Mitchell, "Third-party web tracking: Policy and technology," in *Proc. IEEE SP'12*, 413-427, 2012.

[6] J Ghaderi and R Srikant, "Towards a theory of anonymous networking," in *Proc. IEEE INFOCOM'10*, 686-694, 2010.

[7] S Han, V Liu, Q Pu, S Peter, T Anderson, A Krishnamurthy, and D Wetherall, "Expressive privacy control with pseudonyms," in *Proc. ACM SIGCOMM'13*, 291-302, 2013.

[8] DD Clark, J Wroclawski, KR Sollins, and R Braden, "Tussle in cyberspace: defining tomorrow's Internet," *IEEE/ACM Transactions on Networking*, 13(3), 462-475, 2005.

[9] R Dingledine, N Mathewson, and P Syverson, "Tor: The second-generation onion router," in *Proc. USENIX SSYM'04*, 303-320, 2004.

[10] M Sung, J Xu, J Li, and L Li, "Large-scale IP traceback in high-speed internet: practical techniques and information-theoretic foundation," *IEEE/ACM Transactions on Networking*, 16(6), 1253-1266, 2008.

[11] L Atzori, A Iera, and G Morabito, "The internet of things: A survey," *Elsevier Computer networks*, 54(15), 2787-2805, 2010.

[12] A Al-Fuqaha, M Guizani, M Mohammadi, M Aledhari, and M Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, 17(4), 2347-2376, 2015.

[13] M Blaze, "Trust management and network layer security protocols," in *Security Protocols*, 1550, 109-118, LNCS, 1999.

[14] H Luo, J Kong, P Zerfos, S Lu, and L Zhang, "URSA: ubiquitous and robust access control for mobile ad hoc networks," *IEEE/ACM Transactions on Networking*, 12(6), 1049-1063, 2004.

[15] K Lu, Y Qian, and HH Chen, "A secure and service-oriented network control framework for wimax networks," *IEEE Communications Magazine*, 45(5), 124-130, 2007.

[16] DG Andersen, H Balakrishnan, N Feamster, T Koponen, D Moon, and S Shenker, "Accountable Internet protocol (AIP)," in *Proc. ACM SIGCOMM'08*, 339-350, 2008.

[17] HC Hsiao, THJ Kim, A Perrig, A Yamada, SC Nelson, M Gruteser, and W Meng, "LAP: Lightweight anonymity and privacy," in *Proc. IEEE SP'12*, 506-520, 2012.

[18] R Hummen, H Wirtz, JH Ziegeldorf, J Hiller, and K Wehrle, "Tailoring end-to-end IP security protocols to the Internet of Things," in *Proc. IEEE ICNP'13*, 1-10, 2013.

[19] D Mazieres, M Kaminsky, MF Kaashoek, and E Witchel, "Separating key management from file system security," *SIGOPS Operating Systems Review*, 33(5), 124-139, 1999.

[20] S Matsumoto, RM Reischuk, P Szalachowski, THJ Kim, and A Perrig, "Authentication Challenges in a Global Environment," *ACM Transactions on Privacy and Security*, 20(1), 1:1-1:34, 2017.

[21] A Ghodsi, T Koponen, J Rajahalme, P Sarolahti, and S Shenker, "Naming in content-oriented architectures," in *Proc. ACM ICN'11*, 1-6, 2011.

[22] A Venkataramani, JF Kurose, D Raychaudhuri, K Nagaraja, M Mao, and S Banerjee, "Mobilityfirst: a mobility-centric and trustworthy Internet architecture," *SIGCOMM Computer Communication Review*, 44(3), 74-80, 2014.

[23] R Moskowitz and P Nikander, "Host identity protocol (HIP) architecture," RFC 4423, May 2006.

[24] R Küsters, T Truderung, and A Vogt, "Accountability: definition and relationship to verifiability," in *Proc. ACM CCS'10*, 526-535, 2010.

[25] J Liu and Y Xiao, "Temporal accountability and anonymity in medical sensor networks," *Mobile Networks and Applications*, 16(6), 695-712, 2011.

[26] YC Lee, C Wang, AY Zomaya, and BB Zhou, "Profit-driven service request scheduling in clouds," in *Proc. IEEE CCGrid'10*, 15-24, 2010.

[27] T Shu, Y Chen, and J Yang, "Protecting multi-lateral localization privacy in pervasive environments," *IEEE/ACM Transactions on Networking*, 23(5), 1688-1701, 2015.

[28] C Wang, S Chow, Q Wang, K Ren, and W Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, 62(2), 362-375, 2013.

[29] J Shao, R Lu, and X Lin, "Fine: A fine-grained privacy-preserving location-based service framework for mobile devices," in *Proc. IEEE INFOCOM'14*, 244-252, 2014.

[30] D Han, A Anand, F Dogar, B Li, H Lim, M Machado, A Mukundan, W Wu, A Akella, DG Andersen, JW Byers, S Seshan, and P Steenkiste, "XIA: Efficient support for evolvable internetworking," in *Proc. USENIX NSDI'12*, 309-322, 2012.

[31] W Ramirez, X Masip-Bruin, M Yannuzzi, R Serral-Gracia, A Martinez, and MS Siddiqui, "A survey and taxonomy of ID/Locator Split Architectures," *Elsevier Computer Networks*, 60, 13-33, 2014.

[32] A Boukerche and Y Ren, "A trust-based security system for ubiquitous and pervasive computing environments," *Computer Communications*, 31(18), 4343-4351, 2008.

[33] S Oueslati, J Roberts, and N Sbihi, "Flow-aware traffic control for a content-centric network," in *Proc. IEEE INFOCOM'12*, 2417-2425, 2012.

[34] H Kllapi, E Sitaridi, MM Tsangaris, and Y Ioannidis, "Schedule optimization for data processing flows on the cloud," in *Proc. ACM SIGMOD'11*, 289-300, 2011.

ACKNOWLEDGMENTS

This work is partially supported by the National Key Technology Research and Development Program (No. 2017YFB0801801), the National Science and Technology Major Project of the Ministry of Science and Technology of China (No. 2017ZX03001019), and the National Natural Science Foundation of China (No. 61672490 and No. 61303241).