G2F: A Secure User Authentication for Rapid Smart Home IoT Management

Hongwei Luo, *Member, IEEE*, Chao Wang, *Student Member, IEEE*, Hao Luo, *Student Member, IEEE*, Fan Zhang, *Member, IEEE*, Feng Lin, *Senior Member, IEEE*, and Guoai Xu

***Abstract*—Internet-of-Things (IoT) devices are widely deployed nowadays. A large number of smart home IoT devices are hosted on a cloud server for easy management. Users can use their accounts to initiate operations and management on IoT devices through a cloud server, such as updating firmware and configuring devices. However, the cloud account may be hacked resulting in adversarial attacks to the hosted IoT devices. As a consequence, an adversary may perform malicious operations through the cloud remotely to the hosted IoT devices without user awareness. Motivated by this, in this article we propose gateway-based 2 factor authentication (G2F), a secure user authentication framework dedicated for a gateway based on the universal 2nd factor (U2F) protocol to enhance the security of IoT devices management. In G2F, the user authentication on the gateway is completed utilizing a hardware token that interacts with the local gateway node to guarantee the token owner's presence. Furthermore, G2F can grant multiple simultaneous operations on IoT devices through just one user authentication. We implement a prototype to further evaluate the performance of G2F. Based on our realization on the commercial IoT server, i.e., Alibaba Cloud, G2F demonstrates the ability to protect against malicious attacks with high authentication efficiency.**

***Index Terms*—Internet of Things (IoT), smart home, user authentication.**

I. Introduction

THE Internet of Things (IoT) has drawn increasing attention in recent years, especially in the smart home applications. As a rising portion of IoT, smart home [1] consists of a series of smart devices that are embedded with microprocessor-based controllers. These smart devices can work together to assist with the daily tasks of the user, such as cleaning, cooking, and health monitoring [2].

With the rapid growth of smart devices, smart home brings increasing security and privacy challenges due to its widespread deployment. Nowadays, a large number of smart home IoT devices are hosted on a cloud server for management. Due to lack of user interface of these devices, operations, such as firmware (FW) update, data reading, and state control on smart devices are performed on the cloud or assisted with a specific application (App). However, these kinds of management of IoT devices have potential vulnerabilities, such as cloud-account-stolen [3], or cross-site request forgery (XSRF) attack [4]. An adversary may get the password of a user's account by phishing and access the account to perform a malicious operation to IoT devices. For example, the attacker may update an infected or outdated FW which may pose a great threat to the hosted IoT devices to launch attacks, e.g., data exfiltration [5] and data manipulation [6]. Even if the IoT device can be updated with assistance from an App on a mobile phone for security consideration, the adversary may bypass such secure procedure by exploiting the vulnerabilities of the App [7]–[9], which is a basic concern [10].

There exist some security supports for IoT. Several Internet Engineering Task Force (IETF) working groups have been set up to solve existing security issues of IoT. IETF has proposed a series of protocols [11] which play vital roles in securing the communication between the resource-constrained IoT devices. Though effective, they have not offered a secure and effective architecture for the management of IoT devices. Although there are several proposed schemes [12]–[15] to protect mission-critical IoT applications, most of them are system-wide designs which require experienced network technician to maintain the security through professional tools. This limits these schemes being applied in the smart home environment as smart home usually lacks technical support, unlike an enterprise-scale IoT with its own dedicated IT department or technical team for maintenance and management.

Compared with App-assisted authentication, hardware token possesses a higher level of security due to its hardware features, such as tamper-resistant design. It is often used for online user authentication, e.g., access to bank account [16]. Most of the hardware tokens adopt the fast identity online (FIDO) alliance universal 2nd factor (U2F) protocol [17] which is based on a challenge-response scheme for user authentication, and FIDO2 [18], the passwordless evolution of FIDO U2F, is still based on the U2F model. Users use a

Manuscript received November 19, 2020; revised December 9, 2020; accepted January 6, 2021. Date of publication January 11, 2021; date of current version June 23, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2020AAA0107700 and Grant 2018YFB0803600; in part by the National Natural Science Foundation of China under Grant 61972348 and Grant 62072398; in part by the Zhejiang Key Research and Development Plan under Grant 2019C03133; in part by the Leading Innovative and Entrepreneur Team Introduction Program of Zhejiang under Grant 2018R01005; in part by the Alibaba-Zhejiang University Joint Institute of Frontier Technologies; and in part by the Research Institute of Cyberspace Governance in Zhejiang University. *(Hongwei Luo and Chao Wang contributed equally to this work.) (Corresponding author: Feng Lin.)*

Hongwei Luo and Guoai Xu are with the National Engineering Laboratory of Mobile Network Security, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: kevin.lhw@antfin.com; xga@bupt.edu.cn).

Chao Wang, Hao Luo, Fan Zhang, and Feng Lin are with the Institute of Cyberspace Research, Zhejiang University, Hangzhou 310027, China (e-mail: wangchao5001@zju.edu.cn; luohao421@zju.edu.cn; fanzhang@zju.edu.cn; flin@zju.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3050710

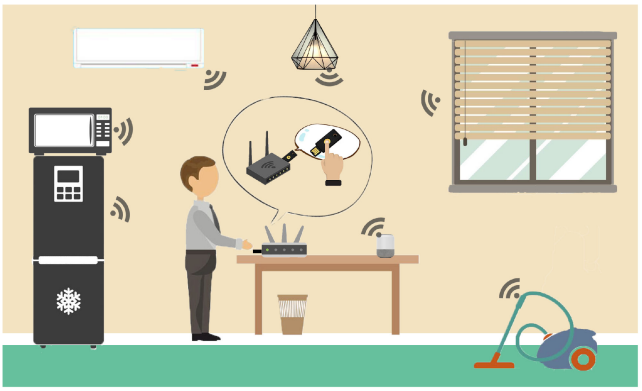


Fig. 1. G2F provides user authentication on the GWN in IoT environment, such as smart home. User can achieve the authentication utilizing a hardware token held by himself to secure the IoT devices.

hardware token to respond to the authentication request from a server through a user client. Although hardware token is a strong factor of authentication, it can only authenticate one client of the user at once. Considering the large amount of IoT devices and correspondingly a large number of clients, the protocol has a potential vulnerability of authentication efficiency in IoT. So we choose to shift the overload of authentication to the gateway which is proved to be a good solution.

In this article, we propose gateway-based 2 factor authentication (G2F), a secure user authentication framework dedicated for the gateway based on the U2F protocol for securing IoT devices from malicious attacks. When there is critical operation for IoT management[1] initiated from cloud server to IoT device, user authentication is required, which is achieved by a hardware token that serves as the strong factor and interacts with cloud server via the USB interface on the gateway. The illustrative diagram is shown in Fig. 1. As the critical operation (e.g., FW update) is not often so the proposed user-centric authentication will not distract and overwhelm the user. We also exploit G2F to support multiauthentication which means simultaneous management to multidevices can be authenticated by only one single user authentication on the gateway. This is significant for improving the efficiency of IoT management. We conduct security and latency evaluation of our prototype and received good results. We finally deploy G2F in a real-world case of IoT, i.e., FW update of IoT device, to give further support of our scheme. It is worth mentioning that G2F requires no extra security hardware-module added to IoT devices, which makes it easy to deploy.

In summary, the contributions described in this article are threefold as follows.

1) We propose a user-centric framework for user authentication on smart home gateway based on U2F protocol, which prevents IoT devices hosted on a cloud server from malicious operations without the user's awareness.
2) We combine the strong factor of authentication in U2F protocol, i.e., tamper-resisted hardware token, with the gateway-centric architecture of IoT to achieve high security and efficiency in the smart home IoT management, thus reducing the dependence of service providers.
3) We exploit a prototype of our proposed framework and perform comprehensive evaluations including security and latency time performance in the real-case study by leveraging a commercial cloud server, i.e., Alibaba Cloud, for the FW update. Considering the universality, G2F can theoretically work with other cloud platforms, e.g., AWS and Azure, because the architectures of these cloud platforms for IoT management are similar to Alibaba Cloud according to their open-source SDKs and documents. We use the former for the convenience of the region.

[1]We emphasize that the critical operation for IoT management is different from some operations (e.g., unlock the door or open the window using smart IoT devices) which happen more frequently in normal life. We define the critical operation for IoT management as the significant interaction between the IoT cloud server and local IoT devices(e.g., gateway or smart devices), which concerns the security of local IoT devices, such as FW update.

II. Motivations and Related Work

A. Motivations

1) Vulnerabilities of Existing Management of IoT: Authentication plays a vital role in the management of IoT, especially for some burgeoning scenes, such as smart home. In most cases, the management of smart home often lacks dedicated security professionals who can manage the complexities of a smart home network. Few householders have professional knowledge about security and therefore cannot deal with common problems of smart home properly and independently. To avoid the burden of management, most of the users turn to service providers for convenience, which means users can manage the smart home through the graphic user interface (GUI) supported by service providers without knowing the details of implementation. Users have to trust service providers to implement appropriate and sufficient security measures for their data and management of devices, while this is not often the case [11]. Specifically, there are some basic concerns about this approach. One of the concerns is that we cannot ensure the cloud server is secure and reliable enough to host and manage our devices, even the service provider. To decrease the dependence of service providers, it is necessary to develop a user-centric authentication model for the management of IoT.

2) Why Gateway: IoT has constrained system resources. Device controllers have traditionally been 8-b microcontrollers with very limited computational and storage resources, which astrict their abilities to implement complex security algorithms. As the central node of IoT, the gateway has more processing power and computational resources, which makes it a solution for some arduous and burdensome work spontaneously. Gateway node (GWN) often acts as a bridge to connect local IoT infrastructure to the cloud server. In terms of security, GWN acts as a firewall to protect IoT devices from cyber threats. It can centralize user authentication and apply access control to defend against unauthorized access to IoT devices.

B. Related Work

1) Smart Home Architecture for Security: There are many different proposals for smart home architectures. However,

none of them can support entire security for smart home. There are three popular architectures designed for Smart Home, which include middleware-based, cloud-based, and gateway-based architecture.

Middleware-based architectures utilize middleware, a kind of software layer between the low-level layer of devices and the high-level application layer (AL), to provide the user with a common interface and a standard data exchange structure to extract bottom details of the hardware. A middleware-based architecture can be a client-server architecture, such as VIRTUS [19] which includes both resource-rich devices (e.g., PCs) and resource-constrained devices (e.g., smartphones), or a peer-to-peer architecture, such as SMEPP [20] which supports resource-constrained embedded devices. Both of these IoT middleware solutions require extra complex software layers and cryptographic routines to be implemented on devices. As a general concern for middleware-based architecture, the coding vulnerabilities in middleware may pose security threats to the IoT devices which can be utilized by an adversary.

Cloud-based architectures are proposed to solve the performance problems of resource-constrained IoT devices. Kovatsch *et al.* [21] proposed a cloud-based architecture on the constrained application protocol (CoAP) to improve the service scalability for IoT devices. Alohali *et al.* [22] proposed a centralized scheme for the management of IoT devices on the cloud. These cloud-based solutions reduce the dependency of a separate home controller and offer a good way for IoT devices to connect and cooperate with the shortcoming of the heavy overhead of computation. However, new privacy issues are introduced due to the dependency of cloud service providers [23]–[25].

Gateway-based architectures are usually intended for the coordination of IoT devices [12], [14], [15]. As the central node in IoT, the GWN can improve the interaction and collaboration between IoT devices and cloud server [26]. An integrated access gateway (IAGW) architecture has also been proposed to support different IoT nodes through standard interfaces in a smart home environment [12]. With a powerful processor, a gateway can implement sophisticated management algorithms and can perform some of the mission-critical operations in smart home. However, most gateway-based architectures are either thoroughly designed [12], [15], [26] or require great changes to the gateway [14], which make it difficult to deploy in the real world.

2) Existing Authentication Mechanisms in IoT: The authentication in IoT can be divided into device authentication and user authentication.

Device authentication usually utilizes the real-time or hardware characteristics of devices [27]–[30]. Gu and Mohapatra [27] introduced BF-IoT, the first IoT secure communication framework for BLE-based networks, to guard against devices spoofing via monitoring the work-life cycles of devices. Novel hardware characteristic can be utilized to identify devices, such as clock skew [29], received signal strength [28] or radio frequency signature [30]. These methods often do not require any changes to devices but the above methods have problems with the efficiency and power consumption of authentication.

User authentication in IoT can be achieved by human biometric characteristics [31] and activity features [32]. Liu *et al.* [31] utilized WiFi signals to extract biometric characteristics of breathing for user authentication. Shi *et al.* [32] extracted the characteristics of a user's daily activity from WiFi fingerprint to accomplish the authentication.

User authentication can also be achieved by hardware, such as smart cards. Hardware-based methods can provide strong authentication because of the characteristics of hardware, such as tamper-resistant and unclonable. A robust and efficient authentication scheme is proposed by Vaidya *et al.* [13]. It is based on strong password approach to provide secure remote access in digital home network environments.

III. Backgrounds

In this part, we introduce some backgrounds about U2F protocol and hardware token.

A. Universal 2nd Factor Protocol

The hardware-based authentication scheme is based on the U2F protocol proposed by FIDO Alliance. The FIDO U2F is an online user authentication protocol which allows online services to augment the security of their existing password infrastructure by adding a strong second factor to user login. Users carry a single U2F device as the second factor. When the user succeeds to log in with a username and password as before, the online service will prompt the user to present a second-factor device at any time it chooses. User can use their FIDO U2F device across all online services that support the protocol leveraging built-in support in Web browsers. U2F protocol adopts a challenge-response scheme, which is extended with phishing and man-in-the-middle (MitM) protection, application-specific keys, device cloning detection, and device attestation. FIDO2 is the evolution of U2F but it is still based on FIDO U2F protocol.

B. Hardware Token

A hardware token is a physical device that can be used to verify someone's identity electronically. The token can be used to take place or in addition to a password which serves as a single factor to prove the owner's identity. Most of the hardware tokens have the tamper-resisted feature as it stores the secret key of the owner. It can be used for data encryption or digital signature. The human interface of hardware token can be a screen to display PIN code or a button for the user to respond to an authentication request. The physical interface of hardware token to interact with other devices, such as smartphone or a laptop, can be USB connector, Bluetooth wireless interfaces, etc. Many online services have supported U2F keys as an additional method of two-step authentication, including Chrome, GitHub, GitLab, Facebook, etc. Take the account logins for example. When a user wants to use the U2F key as the 2FA, he should enroll the key first before the authentication. First, he enrolls the key by launching the registration process on the website. Then the indicator (e.g., a LED) on the U2F key will blink. The user pushes the button on the key for the response. A series of secret messages are sent to the website for enrollment.

After successfully enrolling the key, the user can use it for authentication. The authentication process is similar to the enrollment except that the required operation (e.g., logging in to the account) will be performed after a successful authentication. Considering that the critical operation (e.g., FW update of IoT devices) is not frequent, the user is not expected to perform too much authentication every day.

The hardware-based authentication gains high security by the tamper-resistant hardware token. Although there are some works about secure analysis on the hardware token [33], [34], most of them are side-channel attacks which require professional electronic equipment to perform.

IV. Threat Model

Our work aims to build a user authentication mechanism for the purpose of protecting IoT devices from unauthorized-operation attacks. As a growing part of IoT, smart home is a typical scenario to take into consideration, where many IoT devices are hosted on the cloud for management through the user account.

A. Adversarial Goals

The adversary considered in this work has several goals related to performing unauthorized operations to the IoT devices. These goals are summarized as follows.

1) *Concealment:* An attacker does not want the presence of his or her to be known, which includes illegal login of the user's cloud account and unauthorized operations to devices through the cloud server. If an attack can be easily detected, preventative countermeasures like disconnecting IoT devices or freezing cloud account can often be taken to mitigate the damage done by the attack.
2) *Illegal Login of User Account:* To perform the attack, the adversary should first be able to log in to the user's account of the cloud server and then perform unauthorized operations from the cloud server. When there is an App-assisted second-factor authentication of the account, the adversary has to try to bypass the App-assisted authentication.

B. Assumptions

Adversary:

1) The malicious FW update to IoT devices has been proved to be practical [5], [6], [35]. So we assume the adversary is powerful enough to bypass the FW validation in the IoT Server.
2) We assume the adversary can get the username and password of the user's account by phishing [36] or some other similar methods. We also consider a more powerful adversary who can bypass the App-assisted second-factor authentication of account login by leveraging the vulnerabilities of the App on the user's mobile terminal [7].
3) The hardware token of a user may be stolen or lost and taken by the adversary. However, there are some limitations to the adversary. As the central device of IoT, the GWN cannot be physically accessed by the adversary.

Guarantee:

1) We assume that the hardware token is tamper-resistant and can be used as a strong security factor in the authentication.
2) We mainly focus on the authentication mechanism which is set up on the AL, so we assume that the security of communication layers under AL in TCP/IP model (RFC 1122) can be guaranteed by existing solutions and will not be discussed in this article.

C. Attack Type

Our scheme is applied to user authentication which is achieved by the interaction of hardware token, GWN, and cloud server. The security of communications between IoT devices and the cloud server is not in the scope of this work. Security analysis about regular cyberattacks targeting cloud server will be discussed in Section VIII, which includes phishing attack, XSRF attack [4], and MitM attack.

1) Phishing Attack: Phishing is a form of social engineering in which an attacker attempts to fraudulently acquire sensitive information from a victim by impersonating a trustworthy third party [3]. In the scenario of IoT, an adversary may fake an IoT service website to trick users into logging in to their accounts and then steal the user's account password. We will introduce that G2F is resistant to phishing attacks through detailed evaluations in Section VIII.

2) Cross-Site Request Forgery Attack: XSRF attack denotes a relative class of attack against Web application users. By launching a successful XSRF attack against a user, an adversary can initiate arbitrary HTTP requests from that user to the vulnerable Web application [4]. Thus, if the victim is authenticated, a successful XSRF attack effectively bypasses the underlying authentication mechanism. If an adversary can launch an XSRF attack on the Web of cloud service, the user's login name and password may be changed by the adversary. Considering a more severe situation, the adversary updates malicious FW to the hosted IoT devices.

3) Man-in-the-Middle Attack: The MitM attack is a common attack that has been used against a wide range of protocols, going from login protocols, entity authentication protocols, etc. In this article, we will present how G2F prevents MitM attack in our proposed authentication scheme.

V. Design Consideration

To implement G2F, there are several design aspects to take into consideration.

1) *Security:* The strong factor of authentication should be considered to defend against malicious attacks. The scheme should be user-centric and reduce the dependency of cloud service providers in the authentication.
2) *Latency:* G2F should not take too much time to accomplish the authentication but within a proper maximum of latency.
3) *Efficiency:* Considering a large number of IoT devices, only one authentication operation is required to operate multiple devices to avoid tedious operation and decrease the overheads of authentication.

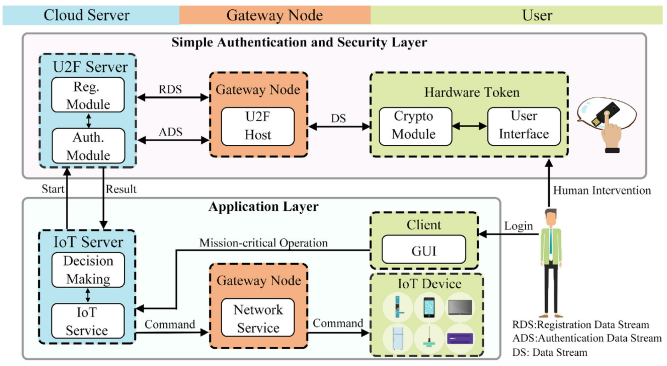


Fig. 2. System Overview. G2F consists of three parties, cloud server, GWN, and user. The cloud server consists of U2F Server and IoT Server.

TABLE I
NOTATION USED IN THE SCHEME

challenge	a string generated by U2F Server randomly
app_id	a string that declares which facets belong to this IoT application
origin	the URI of U2F Server
channel_id	TLS Channel ID
$\{k_{priv}, k_{pub}\}$	key pair(private key and public key) generated by hardware token
h	key handle generated by hardware token and identify keypair
t	total time of authentication from starting to returning the result
attestation cert	information about hardware token, can be used to verifying the authenticity of the hardware token
counter	a counter inside the hardware token, which is utilized to count authentications that have been performed

4) *Compatibility:* To make G2F practical, the authentication protocol should achieve user authentication without making changes to the IoT devices and existing communication protocols and can be directly deployed in a real IoT environment.

Challenges: There is always a tradeoff between security and efficiency in IoT. To achieve high security, the efficiency of authentication is often disregarded [19], [20], [27] and vice versa [31], [32]. To gain both, a thorough design of the protocol is often required [13], [28]–[32]. It is challenging to 1) introduce a strong-factor of authentication with high efficiency and to 2) avoid a total redesign of architecture or extra hardware changes to IoT devices. However, it is possible to solve these problems by introducing the U2F-based protocol on the gateway and design details about G2F.

VI. System Design

To realize U2F authentication on the gateway in smart home, the proposed scheme consists of two phases, i.e., the registration phase and the authentication phase. Before discussing the implementation, this section provides an overview of the system and its characters.

A. System Overview

The architecture is illustrated in Fig. 2. The system involves three parties, cloud server, GWN and user. The cloud server consists of U2F Server and IoT Server, which stands as a security module and provides IoT service, respectively. In addition to providing network service, the GWN is integrated with a software submodule, U2F Host, which guarantees the interaction between U2F Server and hardware token. The user holds the hardware token which consists of a crypto submodule and user interface, and owns a set of IoT devices. The owner of IoT devices host the devices on IoT Server and manage them through a GUI of a client on a computer or mobile terminal.

B. System Components

IoT Server: The IoT Server provides IoT services, such as IoT device remote control and configuration. A user logs in the IoT Server through a client GUI to host and manage IoT devices, such as FW update, sensor data relay and etc. The submodule Decision Making decides when to inform U2F Server to start registration and authentication and determines whether to perform the command from the user based on the results.

U2F Server: U2F Server coordinates with the hardware token to implement U2F registration and authentication. It stores the security elements about the hardware token, such as public keys, key handles associated with public keys, etc. It serves as a submodule of the cloud server to interact with IoT Server to achieve the user authentication.

Gateway Node: GWN is an entity that connects IoT devices to cloud server. In addition to collecting data from IoT devices and sending them to cloud server for the application, GWN possesses a USB interface to enable a hardware token to plug in. A software submodule called U2F Host listens on the port to relay the data associated with U2F registration and authentication, to support the interaction between hardware token and cloud server.

Hardware Token: The hardware token is a tamper-proofing a security element of the architecture with a high level of security, held by the user. It contains a series of private keys and other security elements and interacts with the GWN by USB interface. Users can respond to the registration and authentication requests by pressing the button on it.

IoT Device: IoT devices are owned by the user and hosted on the IoT Server for management. They connect to the GWN wirelessly to obtain network service. Users can perform operations on IoT devices through the client GUI of IoT Server.

C. Proposed Scheme

The proposed scheme is a GWN-based centralized authentication. All the authentication is accomplished on the GWN rather than the clients on IoT devices which is different from the Web-based U2F authentication. Some key items of the registration and authentication flows are referred to in Table I.

1) Registration Phase: The user should register the hardware token first before utilizing it for user authentication. In the registration phase, the user uses the hardware token to respond to the registration request from the U2F Server and then generate a secret key and publish the public key to the

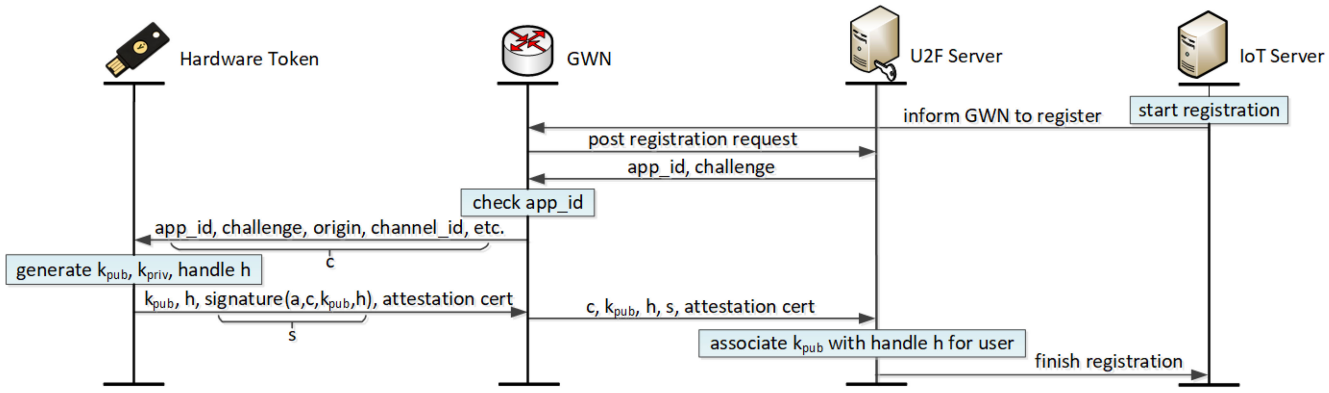


Fig. 3. Registration flow. To perform user authentication, the hardware token held by the user should be registered first.

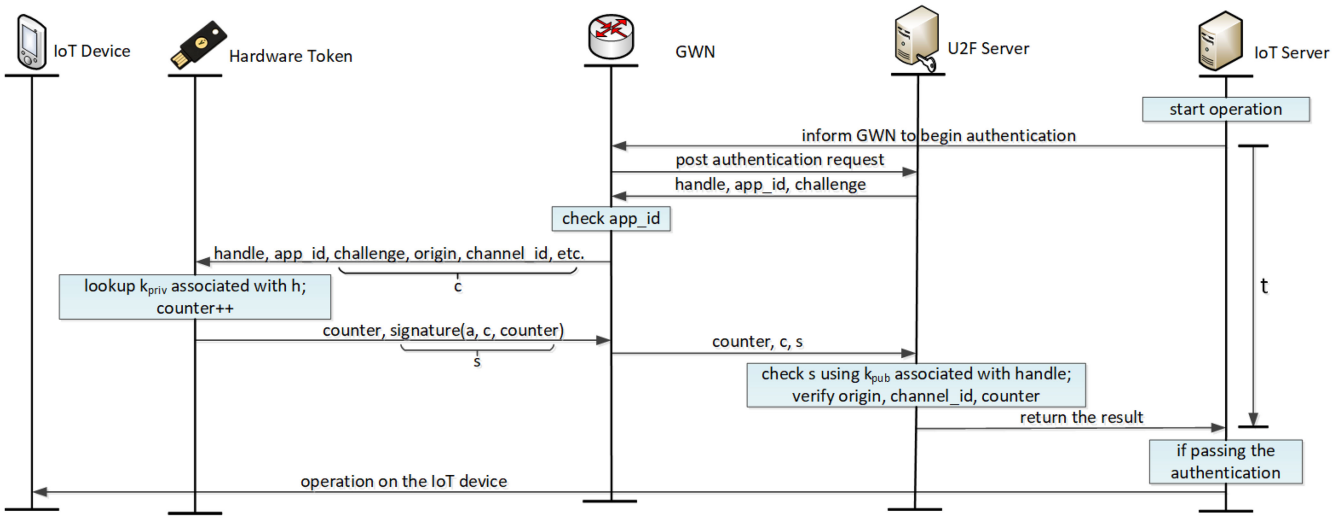


Fig. 4. Authentication flow. When an operation is initiated on the IoT Server, the IoT Server will start a user authentication first.

Cloud Server preparing for the authentication phase. The data flow of the registration phase is shown in Fig. 3.

1) The user starts the registration on the IoT Server. The IoT Server informs the GWN to post a registration request to the U2F Server.
2) The U2F Server responds to the request and sends {app_id, challenge} to the GWN.
3) The GWN checks the app_id and relays {app_id, challenge, origin, channel_id, etc.} to the hardware token to generate key pairs.
4) The hardware token generates $\{k_{\text{priv}}, k_{\text{pub}}, h\}$ based on the app_id.
5) The U2F Server verifies the signature with k_{pub} and stores $\{k_{\text{pub}}, h\}$ for the user if the verification is successful.
6) Finally, the U2F Server informs the IoT Server the registration is finished and IoT Server represents the result of registration to the user.

2) Authentication Phase: As Fig. 4 shows, user authentication is required if the user wants to initiate some operations on the IoT devices through the IoT Server.

1) If IoT Server receives any operation commands, it will inform GWN to issue an authentication request to U2F Server.
2) On receiving the request, U2F Server sends handle, app_id and a random challenge to GWN. The GWN checks app_id and relays {handle, app_id, challenge, origin, channel_id, etc.} to the hardware token.
3) The hardware token looks up k_{priv} based on key handle h, and signs {app_id, c, counter}. The insider counter adds up every time a signature is finished. Then the counter and signature are sent to GWN to relay to U2F Server.
4) The U2F Server checks the signature using k_{pub} which have been saved during the registration phase, based on key handle.
5) The IoT Server will take the next operation according to the authentication result.

3) Support for Multiple Operations: There is a need of multiple authentications for multiple operations to devices in the middleware-based and cloud-based architecture [19], [37]. Because these authentications require the devices to communicate with the authentication server, respectively, which indicates reduplicative requests, verifications, and responses. As for G2F, only a single user authentication between the gateway and U2F service is required. A set of similar operations, e.g., FW update of several devices, can be predefined on the IoT Server. After the successful user authentication, the commands corresponding to such a set of operations can be initiated.

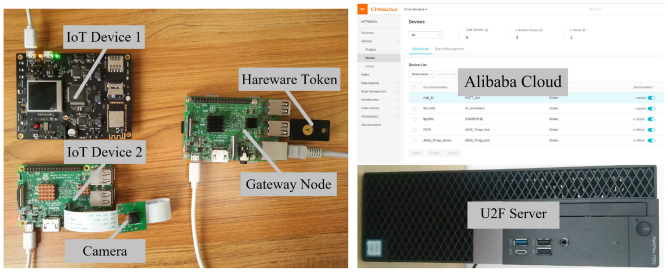
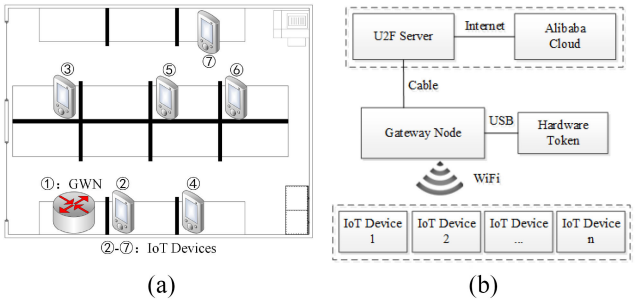


Fig. 5. System implementation. We adopt Raspberry Pi 3 as the GWN and a YubiKey as the hardware token. The IoT devices include a AliOS Developer Kit and a Raspberry Pi 3. We adopt Alibaba Cloud as the IoT Server to provide services for IoT devices, e.g., FW update, and we run the U2F Server on a Linux (Ubuntu 18.04 TLS) desktop.



(a) (b)

Fig. 6. (a) We perform experiments in a in-room laboratory scenario as shows. (b) Indicates the physical connection relationship of the components.

In a smart home scenario, if the user wants to perform some critical operations (e.g., update the FW of IoT devices and open the home surveillance camera), he can first log in to the IoT management page on the IoT platform (e.g., Alibaba Cloud, AWS, and Azure). Then the user plugs in the hardware token on the GWN and initiates one or several commands at one time to the IoT devices from the IoT Server. The IoT Server sends a message to inform the GWN to initiate the user authentication. The indicator (e.g., a LED) of the hardware token tells the user to respond to the authentication request (i.e., push the button on the hardware token). Then several identity-related messages are sent to the GWN. The GWN transmits the messages to the U2F Server for verification. Only if the user passes the verification, the IoT Server sends the required command/commands to the IoT device as the user wishes (i.e., the IoT device starts the FW update and the surveillance camera is opened).

VII. Evaluation Setup

A. System Implementation

For flexible exploitation, we adopt common IoT devices to implement the system, e.g., Raspberry Pi and developer kit. The details of G2F components are shown in Fig. 5. The physical connection of the components is indicated in Fig. 6(b).

1) Gateway Node: We adopt a Raspberry Pi 3 (ARM Cortex-A53 64-b processor with 1.4-GHz CPU clock) as the GWN. Many off-the-shelf IoT gateway products are even more powerful than it and most of them have USB ports, e.g., Linksys, TP-Link, and Google WiFi. Raspberry Pi 3 supports external USB devices and comes with built-in WiFi connectivity. We deploy U2F Host on it to achieve the interaction between hardware token and U2F Server.

2) Hardware Token: YubiKey is a commercial security key, offering strong two-factor authentication from industry leader Yubico. As U2F libraries associated with YubiKey has been open-sourced for developers, we adopt YubiKey as the hardware token.

3) U2F Host: We implement the U2F Host software module in python language and run it on the GWN. This module utilizes U2F libraries [38] to support the interaction between hardware token and U2F Server.

4) Cloud Server (Alibaba Cloud and U2F Server): To manage IoT devices, we adopt the commercial Alibaba Cloud which opens software API for developers as the IoT Server. The U2F Server is developed and deployed on a Linux desktop (Intel Core i7-7700 3.60 GHz 64-b CPU) to interact with YubiKey to achieve registration and authentication.

5) IoT Device: Considering that most of IoT platforms only support specific and limited IoT devices. We adopt Raspberry Pi 3 and AliOS Developer Kit as IoT devices because Alibaba Cloud gives full support for these two kinds of IoT devices and users can update their customized FW on the devices. AliOS Developer Kit is a powerful kit that adopts ARM Cortex-M4 processor with 80-MHz CPU clock and has abundant peripherals and interfaces, such as camera, LED screen, and more than eight other sensors.

B. Experiment Setup

We adopt FW update of IoT devices as the real-case study. Note that we only care about whether the cloud server will send the command of FW update to the IoT devices when there requires user authentication. The security analysis about the detailed process of the FW update is beyond our scope. We perform experiments in an in-room laboratory environment as Fig. 6(a) shows. The GWN is deployed at location 1 and IoT devices are deployed at location 2–7. There are some barriers in the scenario, such as tables and cross clapboards. The distance between the GWN and other locations has a minimum of 2 m and a maximum of 22 m. The network latency between GWN and IoT Server (Alibaba Cloud) is about 73 ms with a network bandwidth of 10 Mb/s. As the U2F Server is deployed in the same LAN with GWN, the network latency is less than 1 ms with a 10-Mb/s bandwidth.

C. Evaluation Metrics

1) For security evaluation, we adopt *block rate*, the ratio of successful defense of G2F to the total experiments to evaluate the security performance of G2F.
2) For the latency performance evaluation, we adopt the empirical cumulative distribution function (Empirical CDF) to find which time period the latency of G2F is mainly distributed, i.e., the duration corresponding to the sharp rise of the curve.

VIII. Security Evaluation

A. Unauthorized Operation to IoT Device

1) Defense Performance: Cyberattacks such as phishing, XSRF attack, and MitM attack will result in the same

consequence, namely the unauthorized operations to hosted IoT devices. To achieve this goal, the adversary can try two kinds of specific attack methods. One is "single attack," another one is "persistent multiple attacks."

Single Attack: We log in to the user's account as an adversary and try to initiate a malicious operation (e.g., FW-update command) to AliOS Developer Kit. Without hardware token for user authentication in 6 s (which is a practical value for the authentication, see Section IX for more details), a failure occurs indicating that this is an unauthenticated operation, which means the malicious operation to IoT device is blocked by G2F. We conducted single attack for 100 times, which achieves a block rate of 100%.

Persistent Multiple Attacks: We further evaluate G2F with persistent attacks, which means the user's computer suffers from an adversarial trojan which continuously sends a malicious operation to the IoT Server with certain intervals. When the legitimate user happens to perform an operation for IoT management before the IoT Server gets the result of authentication from the U2F Server, the malicious operation has a chance to bypass the authentication. Note that such a kind of trojan attack is a strong attack. Because preset secure factors, such as product key and device key, must be set up in the trojan programs to communicate with IoT platform [39]–[41]. So the adversary has to get the secure factors of target devices first to enable the trojan to call the APIs to communicate with IoT Cloud without being detected.

For the persistent attack, we run a trojan script on the user's desktop (Intel Core i7-7700 3.60GHz 64-b CPU) to send specific malicious operation requests to the user's Alibaba Cloud account persistently utilizing the open-sourced cloud APIs (we assume that the attacker is able to get the preset secure factors of the target devices). The script can send malicious requests to the cloud at different time intervals. When a user also happens to send legitimate operation requests to the cloud at the same time through the normal GUI of the cloud server (e.g., Web page) and accomplish the user authentication of G2F, the malicious operation request has a specific possibility to bypass the authentication. Because before the G2F authentication starts, the cloud service receives a series of operation requests but it cannot distinguish which ones of them are initiated by the legitimate user when there is a malicious request mixed in.

Considering that there are normally ten devices in the smart home, we run ten clients on IoT devices listening for FW update commands. In our experiments, a user sends operation requests to Alibaba Cloud for the FW update of IoT devices and the command sent to IoT devices from the IoT Server is less than 200 B with an average network latency of 200 ms. The trojan script also sends malicious requests at different time intervals to initiate outdated FW update. After 50 experiments for each interval set, we calculate the block rate of G2F, as the blue curve in Fig. 7(a) shows. Overall, the block rate increases with the attack interval and achieves 100% when the attack interval is above 6 s. Considering that when the trojan persistently sends malicious requests to IoT Server, there will be many authentication requests sent to the local GWN in G2F. When the user notices the unexpected authentication requests

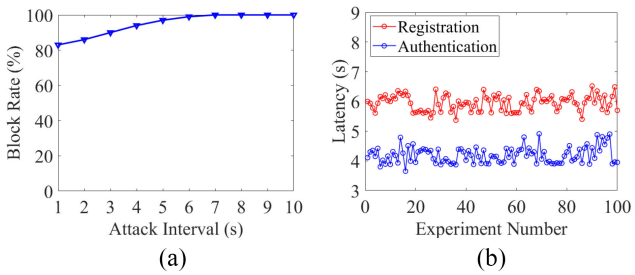


Fig. 7. (a) Impact of command size on block rate when there is a persistent attack with a 6-s attack interval. (b) Total latency of registration and authentication during the 50 experiments.

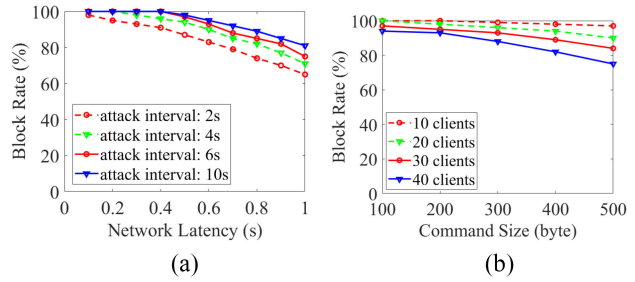


Fig. 8. (a) Impact of network latency on block rate. The diverse curves represent different attack interval, i.e., 2, 4, 6, and 10 s. (b) Impact of client number and command size on block rate.

(the LED indicator of hardware token will flare when there is an authentication request [42]), this is likely a malicious attack by an adversary, which is against the adversary's goal of concealment as we have demonstrated in Section IV-A. The user can further take protection measures, such as freezing the IoT Server account and disconnect local IoT devices.

The low block rate is caused by the system latency, which includes 1) the communication latency of sending multiple operation requests to cloud server; 2) the computational latency of local device (i.e., GWN and hardware token); and 3) the latency of handling these requests by the cloud server. Considering that the Alibaba Cloud is powerful enough to reduce the handling latency to less than 1 ms and local devices are normal IoT devices that can be realized with more powerful hardware in real products, we focus on the communication latency which takes the main part. We further explore the impact of network conditions, the number of clients, and the command size of operation on the block rate in Section VIII-A2.

2) Block Rate Analysis (Network Latency): We use netem [43] to control the network latency. The attack sends malicious operation requests to Alibaba Cloud persistently with different attack intervals (from 2 s to 10 s, as Fig. 8(a) shows). We change the network latency from 100 to 1000 ms and perform 50 experiments for each network condition. The block rate under these network conditions is calculated and shown in Fig. 8(a). The result in Fig. 8(a) indicates that the network condition has a significant impact on the block rate of G2F, especially when there is a small attack interval. However, when the network latency is less than 400 ms and the attack interval no less than 6 s, the block rate of G2F can achieve 100%.

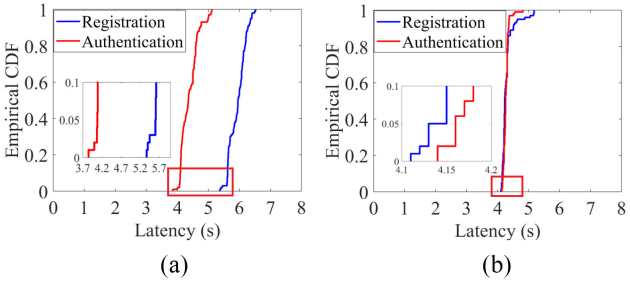


Fig. 9. Empirical CDF of total latency and user waiting time. (a) Total latency of registration and authentication, respectively, in the whole process. (b) User waiting time before the hardware token receives the request of authentication.

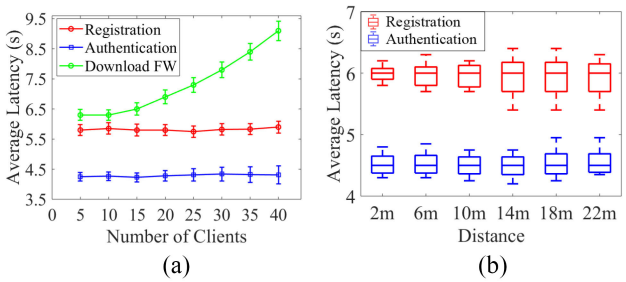


Fig. 10. (a) Indicates that clients have little impact on the average latency of registration and authentication, but consuming time of downloading FW increases with the number of clients. (b) Indicates that the real-time performance of G2F is stable within a range of 22 m which is a normal distance between GWN and IoT devices in a smart home environment.

Number of Clients and Command Size: As the block rate is closely related to the latency of the system, we perform further evaluation under different circumstances that influence the system latency, i.e., the number of clients on IoT devices and the command size of operations for IoT management. The attack interval of the adversarial trojan script is set up to 6 s. As Fig. 8(b) shows, G2F can support as many as 20 clients of devices authenticated at one time with the block rate of 100% when the size of command for IoT management is up to 300 B (which is far larger than a practical size of command for IoT management).

B. Hardware Token Stolen

We assume a more powerful adversary that not only is able to log in the user's account but also succeed to steal the user's hardware token. The adversary tries to send unauthorized operation command through the cloud server to IoT devices. However, unable to get access to the GWN physically and plug the hardware token in, the unauthorized operation is rejected by the IoT Server after a preset time of 6 s.

IX. Time-Efficiency Evaluation

In this part, we first evaluate the whole performance by the latency of registration and authentication, which means the consuming time of G2F during registration and authentication. Then we launch a series of clients on IoT devices and initiate multiple operations of FW update through the GUI of Alibaba Cloud to these devices. Considering that the wireless connection to GWN, the distance between IoT devices and GWN, and the size of a single command may be potential factors that have an impact on the performance of our prototype.

A. Latency of Registration and Authentication

We deploy a single IoT device (Raspberry Pi 3) within 5 m of the GWN. We perform 100 times of registration and authentication, respectively, as Fig. 7(b) shows and draw the result of empirical CDF for each in Fig. 9(a). The results of 100 times of experiments indicate that the registration phase has an average latency of 6.07 s and authentication 4.27 s as shown in Fig. 9(b). Compared with some hardware tokens for secure financial transactions which generally take a response time of 3 to 5 s, the average authentication time of 4.27 s is acceptable. The data flow in Figs. 3 and 4 can explain the different latencies between the two phases. Compared with Fig. 4, data flows in Fig. 3 have a larger number of data packets because of public-key distribution and extra information about hardware token required in the registration phase.

B. Support for Multiple Operations

To evaluate the efficiency of G2F, we initiate multiple operations of FW update for IoT devices through the GUI of the cloud server on a computer. Corresponding to the multiple operations, we launch almost 40 IoT clients on Raspberry Pi 3 to receive the command of these operations, respectively.

Fig. 10(a) shows the average latency and standard deviation of each experimental set. With growing number of clients, the average latency of registration is stable but the authentication time increases a little by 0.25 s from five clients to 40 clients. That is because the registration is only achieved by the interaction of GWN, hardware token, and U2F Server. As for the authentication, the GWN has to transfer several commands from the cloud server to IoT clients, which results in the little increase of latency with the growing number of clients, as shown by the blue curve. Due to the limitation of channel capacity, the green curve of FW-downloading latency rises a little. Considering that a latency of 0.25 s is imperceptible for a user and the number of devices in a single room can be less than 40 in the real world, the G2F shows a stabilized latency of authentication with around 4.3 s.

C. Impact of Distance

As the IoT devices connects to the GWN wirelessly, we take the distance impact into consideration. We launch five IoT clients on a Raspberry Pi 3 and change the distance between the device and GWN. We depict the box plot of the average latency for different distance setting. The impact of distance can be seen in Fig. 10(b). The latency of registration in Fig. 10(b) is changeless as before. The average latency of authentication rises about 0.3 s with the distance from 2 to 22 m. Although the performance of registration shows a little fluctuation, the average latency of authentication is stable at around 4.4 s.

D. Impact of Command Size

We deploy a single device connecting to the GWN at a distance of 5 m and change the size of command for 50

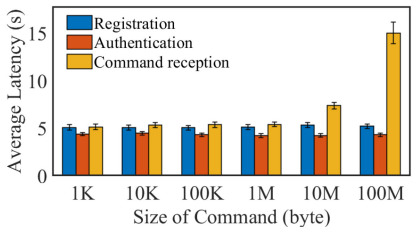


Fig. 11. Impact of command size. The result indicates that the registration and authentication of G2F works well with a steady low latency.

TABLE II
COMPARISON AMONG DIFFERENT ARCHITECTURES

	Phishing	XSRF	MitM	Efficiency
Alohali et al. [22]	×	×	✓	single
Simpson et al. [15]	×	✓	✓	single
Our	✓	✓	✓	multiple

[1] ×: cannot defend; ✓: can defend
[2] Efficiency: support for single or multiple operation(s) in one authentication.

experiments. As the yellow bar shows, the average latency of command is the consuming time for IoT devices to receive the whole bytes of a command. The average latency of each size of command is shown in Fig. 11. The average latency of registration (blue bar) and authentication (red bar) are stable around 5 and 4.3 s, respectively, which show steady trends compared with the whole latency of command (yellow bar). When the command from the server side has a size of 100M bytes, the latency rises to 15.3 s. In spite of this, the size of the command has little influence on the real-time performance of authentication as the results indicate.

E. Power Consumption and Bandwidth Requirement

As for power consumption, the implemented GWN (i.e., Raspberry Pi 3) takes most of the part, which is 1.7 W. Our experiments are performed under different network conditions as shown in Fig. 8(a) with an average bandwidth of 0.3 M/s, which is easy to satisfy for the smart home.

X. COMPARISON BETWEEN DIFFERENT ARCHITECTURES

We compare our architecture with representative cloud-based [22] and gateway-based [15] architectures in the aspects of security defense and efficiency of authentication as Table II shows. Note that G2F is resistant to phishing attacks and can effectively limit the adversary's power even if he has successfully stolen a user's account password. As for the efficiency of authentication, only G2F supports multiple operations to IoT devices in one single user authentication, which shows high efficiency in the management of IoT devices.

Wazid *et al.* [44] proposed a three-factor (i.e., smart card, password, and biometrics) user authentication scheme for the hierarchical IoT network. Their work mainly focuses on data access control of the sensor node which is different from our goal (e.g., secure the critical operations from the IoT Server). The authors propose a well-designed scheme and with elaborate security analysis but they do not perform the experiments on a real test-bed as we did. Barreto *et al.* [45] presented an authentication model to manage the identity of IoT devices and users in IoT clouds. They focus on the IoT cloud for authentication which relies heavily on the IoT cloud provider. They proposed an IoT-cloud-based centralized authentication and focus on the identity management of the IoT cloud which often needs expertise for the management. But our proposed method does not need the user to have specific knowledge about the IoT management to finish the authentication. Our experiments on the real test-bed (i.e., Alibaba Cloud) prove the usability of our method. Compared with the hardware-based key, software-based 2FA has been exposed to security problems. Dmitrienko [**?**] proposed an attack on mobile two-factor authentication apps which reveals the vulnerabilities of software-based two-factor authentication on mobile devices.

XI. DISCUSSION AND FUTURE WORK

The block rate in the security evaluation is largely determined by the system latency of G2F. As we have mentioned before, the short period of latency before the authentication is finished gives the adversary the chance to initiate malicious operations if he could log into the user's account. This problem can be solved in two aspects. First, we can utilize more powerful devices as the GWN, hardware token and U2F Server to decrease the system latency. Second, integrating the U2F Server with the U2F Server can reduce the communication overhead and make the authentication more efficient. Considering that some hardware tokens for secure financial transactions usually take three to 5 s to achieve the authentication, we think the authentication latency of G2F (about four to 5 s) is acceptable for some less-frequent and critical operations for IoT management.

Our future work will focus on how to improve the block rate of G2F and explore more efficient ways of multiple-operations authentication, such as integrating the IoT Server with the U2F Server. Besides, considering that it is difficult to determine which command is critical and when to ask users' approval, a context-aware authentication system may be a good solution for this problem.

XII. CONCLUSION

Motivated by the existing vulnerabilities of IoT management in the real world, we propose the first user authentication mechanism on the GWN based on U2F protocol, called G2F, for the purpose of enhancing IoT device security. User authentication can be achieved by utilizing a hardware token which interacts with the server side through the GWN. We first propose the scheme of G2F and then introduce our prototype of G2F which adopts a commercial Alibaba Cloud for real-case study. The security and real-time performance evaluation part shows that G2F can defend against representative attacks regarding IoT management on cloud server.

To the best of our knowledge, this is the first work utilizing strong factor of authentication on the gateway to guarantee the security and efficiency for the management of smart home IoT devices. We believe our study will help in exploring more secure and efficient infrastructure of IoT management.

REFERENCES

[1] M. Hasan, P. Biswas, M. T. I. Bilash, and M. A. Z. Dipto, "Smart home systems: Overview and comparative analysis," in *Proc. 4th IEEE Int. Conf. Res. Comput. Intell. Commun. Netw. (ICRCICN)*, Kolkata, India, 2018, pp. 264–268.

[2] V. Wade, J. Soar, and L. Gray, "Uptake of telehealth services funded by medicare in Australia," *Aust. Health Rev.*, vol. 38, no. 5, pp. 528–532, 2014.

[3] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, "Social phishing," *Commun. ACM*, vol. 50, no. 10, pp. 94–100, 2007.

[4] A. Barth, C. Jackson, and J. C. Mitchell, "Robust defenses for cross-site request forgery," in *Proc. 15th ACM Conf. Comput. Commun. Security*, 2008, pp. 75–88.

[5] A. Cui, M. Costello, and S. Stolfo, "When firmware modifications attack: A case study of embedded exploitation," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2013, pp. 134–141.

[6] J. Rieck, "Attacks on fitness trackers revisited: A case-study of unfit firmware security," 2016. [Online]. Available: arXiv:1604.03313.

[7] A. Dmitrienko, C. Liebchen, C. Rossow, and A.-R. Sadeghi, "On the (In)security of mobile two-factor authentication," in *Proc. Int. Conf. Financ. Cryptogr. Data Security*, 2014, Springer, pp. 365–383.

[8] H. Wang *et al.*, "Vulnerability assessment of OAuth implementations in android applications," in *Proc. 31st Annu. Comput. Security Appl. Conf.*, 2015, pp. 61–70.

[9] J. T. Beekman, "Breaking cell phone authentication: Vulnerabilities in AKA, IMS and Android," in *Proc. 7th USENIX Workshop Offensive Technol.*, 2013, p. 5.

[10] Y. Liu, C. R. Taylor, and C. A. Shue, "Authenticating endpoints and vetting connections in residential networks," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, Honolulu, HI, USA, 2019, pp. 136–140.

[11] H. Lin and N. W. Bergmann, "IoT privacy and security challenges for smart home environments," *Information*, vol. 7, no. 3, p. 44, 2016.

[12] F. Ding, A. Song, E. Tong, and J. Li, "A smart gateway architecture for improving efficiency of home network applications," *J. Sens.*, vol. 2016, Jan. 2016, Art. no. 2197237.

[13] B. Vaidya, J. H. Park, S.-S. Yeo, and J. J. P. C. Rodrigues, "Robust one-time password authentication scheme using smart card for home network environment," *Comput. Commun.*, vol. 34, no. 3, pp. 326–336, 2011.

[14] E. Kim and C. Keum, "Trustworthy gateway system providing IoT trust domain of smart home," in *Proc. IEEE 9th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Milan, Italy, 2017, pp. 551–553.

[15] A. K. Simpson, F. Roesner, and T. Kohno, "Securing vulnerable home iot devices with an in-hub security manager," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Kona, HI, USA, 2017, pp. 551–556.

[16] A. Hiltgen, T. Kramp, and T. Weigold, "Secure Internet banking authentication," *IEEE Security Privacy*, vol. 4, no. 2, pp. 21–29, Mar./Apr. 2006.

[17] S. Srinivas, D. Balfanz, E. Tiffany, F. Alliance, and A. Czeskis, *Universal 2nd Factor (U2F) Overview*, FIDO Alliance Proposed Stand., Mountain View, CA, USA, 2015, pp. 1–5.

[18] *Client to Authenticator Protocol (CTAP)*, FIDO Alliance, Mountain View, CA, USA, 2019. [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-id-20180227/fidoclient-to-authenticator-protocol-v2.0-id-20180227.pdf

[19] D. Conzon, T. Bolognesi, P. Brizzi, A. Lotito, R. Tomasi, and M. A. Spirito, "The VIRTUS middleware: An XMPP based architecture for secure IoT communications," in *Proc. IEEE 21st Int. Conf. Comput. Commun. Netw. (ICCCN)*, Munich, Germany, 2012, pp. 1–6.

[20] R. C. Benito, D. G. Márquez, P. P. Tron, R. R. Castro, N. S. Martín, and J. S. Martín, "SMEPP: A secure middleware for embedded P2P," in *Proc. ICT-MobileSummit*, vol. 9, 2009, pp. 1–8.

[21] M. Kovatsch, M. Lanter, and Z. Shelby, "Californium: Scalable cloud services for the Internet of Things with CoAP," in *Proc. Int. Conf. Internet Things (IoT)*, Cambridge, MA, USA, 2014, pp. 1–6.

[22] B. Alohali, M. Merabti, and K. Kifayat, "A secure scheme for a smart house based on cloud of things (CoT)," in *Proc. IEEE 6th Comput. Sci. Electron. Eng. Conf.*, Colchester, U.K., 2014, pp. 115–120.

[23] E. Luo, M. Z. A. Bhuiyan, G. Wang, M. A. Rahman, J. Wu, and M. Atiquzzaman, "PrivacyProtector: Privacy-protected patient data collection in IoT-based healthcare systems," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 163–168, Feb. 2018.

[24] *7 Most Infamous Cloud Security Breaches*. Accessed: Jan. 22, 2021. [Online]. Available: https://blog.storagecraft.com/7-infamous-cloud-security-breaches/

[25] R. Choubey, R. Dubey, and J. Bhattacharjee, "A survey on cloud computing security, challenges and threats," *Int. J. Comput. Sci. Eng.*, vol. 3, no. 3, pp. 1227–1231, 2011.

[26] N. W. Bergmann and P. J. Robinson, "Server-based Internet of Things architecture," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, 2012, pp. 360–361.

[27] T. Gu and P. Mohapatra, "BF-IoT: Securing the IoT networks via fingerprinting-based device authentication," in *Proc. IEEE 15th Int. Conf. Mobile Ad Hoc Sens. Syst. (MASS)*, Chengdu, China, 2018, pp. 254–262.

[28] J. Zhang, Z. Wang, Z. Yang, and Q. Zhang, "Proximity based IoT device authentication," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Atlanta, GA, USA, 2017, pp. 1–9.

[29] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Depend. Secure Comput.*, vol. 2, no. 2, pp. 93–108, Apr.–Jun. 2005.

[30] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, 2008, pp. 116–127.

[31] J. Liu, Y. Dong, Y. Chen, Y. Wang, and T. Zhao, "Leveraging breathing for continuous user authentication," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2018, pp. 786–788.

[32] C. Shi, J. Liu, H. Liu, and Y. Chen, "Smart user authentication through actuation of daily activities leveraging WiFi-enabled IoT," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, 2017, p. 5.

[33] J. Grand and J. Friday, "Advanced hardware hacking techniques," in *Proc. DEFCON Conf.*, vol. 12, 2004, p. 59.

[34] D. Oswald, B. Richter, and C. Paar, "Side-channel attacks on the Yubikey 2 one-time password generator," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, 2013, pp. 204–222.

[35] M. Bettayeb, Q. Nasir, and M. A. Talib, "Firmware update attacks and security for IoT devices: Survey," in *Proc. ArabWIC 6th Annu. Int. Conf. Res. Track*, 2019, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/3333165.3333169

[36] D. Pienta, J. B. Thatcher, and A. C. Johnston, "A taxonomy of phishing: Attack types spanning economic, temporal, breadth, and target boundaries," in *Proc. 13th Pre-ICIS Workshop Inf. Security Privacy*, vol. 1, pp. 18–36, 2018.

[37] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "A secure ICN-IoT architecture," in *Proc. Int. Conf. Commun. Workshops (ICC Workshops)*, Paris, France, 2017, pp. 259–264.

[38] *U2F Libraries*. Accessed: Jan. 22, 2021. [Online]. Available: https://developers.yubico.com/U2F/Libraries

[39] *Alibaba Cloud*. Accessed: Jan. 22, 2021. [Online]. Available: https://www.alibabacloud.com

[40] *Aws IoT*. Accessed: Jan. 22, 2021. [Online]. Available: https://aws.amazon.com/iot/

[41] *Microsoft Azure*. Accessed: Jan. 22, 2021. [Online]. Available: https://azure.microsoft.com/en-us/product-categories/iot/

[42] *Yubikey*. Accessed: Jan. 22, 2021. [Online]. Available: https://www.yubico.com/

[43] *Netem*. Accessed: Jan. 22, 2021. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem

[44] M. Wazid, A. K. Das, V. Odelu, N. Kumar, M. Conti, and M. Jo, "Design of secure user authenticated key management protocol for generic IoT networks," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 269–282, Feb. 2018.

[45] L. Barreto, A. Celesti, M. Villari, M. Fazio, and A. Puliafito, "Identity management in IoT clouds: A FIWARE case of study," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, Florence, Italy, 2015, pp. 680–684.

Hongwei Luo (Member, IEEE) received the master's degree in pattern recognition and intelligent system from Beijing University of Posts and Telecommunications, Beijing, China, in 2003, where he is currently pursuing the Ph.D. degree in information security.

He is currently the Senior Staff Standardization Engineer with Alibaba Group, Hangzhou, China, dedicated to standardization ecosystem in order for the development of e-commerce and Internet finance. His research interests include authentication technologies and mobile security.

Dr. Luo is a member of the IEEE Computer Society.

Chao Wang (Student Member, IEEE) received the bachelor's degree in information engineering from Zhejiang University, Hangzhou, China, in 2019, where he is currently pursuing the Ph.D. degree with the School of Cyber Science and Technology, College of Computer Science and Technology.

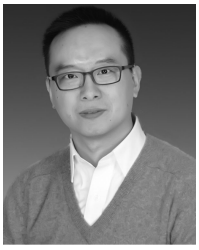
His research interests are wireless sensing, Internet of Things, machine learning, and deep learning.

Dr. Wang is a member of the IEEE Communications Society.

Hao Luo (Student Member, IEEE) received the bachelor's degree in computer science and technology from Huazhong University of Science and Technology, Wuhan, China, in 2017. He is currently pursuing the master's degree with the School of Cyber Science and Technology, College of Computer Science and Technology, Zhejiang University, Hangzhou, China.

His research interests are computer vision, biometrics, person reidentification, generative adversarial networks, and deep learning.

Dr. Luo is a member of the IEEE Communications Society.



Fan Zhang (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, University of Connecticut, Mansfield, CT, USA, in 2011.

He is currently a Full Professor with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include system security, hardware security, cryptography, and computer architecture.

Feng Lin (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, Tennessee Technological University, Cookeville, TN, USA, in 2015.

He is currently a Professor with the School of Cyber Science and Technology, College of Computer Science and Technology, Zhejiang University, Hangzhou, China. He was an Assistant Professor with the University of Colorado Denver, Denver, CO, USA, a Research Scientist with the State University of New York (SUNY) at Buffalo, Buffalo, NY, USA, and an Engineer with Alcatel-Lucent (currently, Nokia), Paris, France. His current research interests include mobile sensing, Internet-of-Things security, biometrics, AI, and IoT applications.

Prof. Lin was a recipient of the Best Paper Award from ACM MobiSys'20, IEEE Globecom'19, IEEE BHI'17, the Best Demo Award from ACM HotMobile'18, and the First Prize Design Award from the 2016 International 3-D printing competition.

Guoai Xu received the Ph.D. degree in signal and information processing from Beijing University of Posts and Telecommunications, Beijing, China, in 2002.

He was awarded the title of a Professor in 2011. He is currently an Associate Director of the National Engineering Laboratory of Security Technology for Mobile Internet, Beijing University of Posts and Telecommunications. His research interests include software security and data analysis.