Accepted Manuscript

A security authorization scheme for smart home Internet of Things devices

Bogdan-Cosmin Chifor, Ion Bica, Victor-Valeriu Patriciu, Florin Pop

PII: S0167-739X(17)31102-0
DOI: http://dx.doi.org/10.1016/j.future.2017.05.048
Reference: FUTURE 3502

To appear in: *Future Generation Computer Systems*

Received date : 17 November 2016
Revised date : 17 May 2017
Accepted date : 28 May 2017

Please cite this article as: B.-C Chifor. B.-C Chifor, I. Bica, V.-V Patriciu. V.-V Patriciu, F. Pop, A security authorization scheme for smart home Internet of Things devices, *Future Generation Computer Systems* (2017), http://dx.doi.org/10.1016/j.future.2017.05.048

Highlights:
Paper title: A security authorization scheme for smart home Internet of things devices

- a lightweight identity stack for IoT for digital identity;
- a Cloud federated authentication for smart home;
- extension of FIDO authentication messages;
- a theft resistant security scheme using a keep-alive protocol;
- validation using Kaa IoT cloud tested network topology.

A security authorization scheme for smart home Internet of Things devices

Bogdan-Cosmin Chifor

Military Technical Academy, 81-83 George Cosbuc Boulevard, Bucharest

Ion Bica

Military Technical Academy, 81-83 George Cosbuc Boulevard, Bucharest

Victor-Valeriu Patriciu

Military Technical Academy, 81-83 George Cosbuc Boulevard, Bucharest

Florin Pop

University Politehnica of Bucharest, Computer Science Department, Splaiul Independentei 313, Sector 6, Bucharest 060042, Romania
National Institute for Research and Development in Informatics (ICI), 8-10, Maresal Averescu, 011455, Bucharest, Romania

Abstract

The Internet of Things (IoT) is becoming an important factor in many areas of our society. IoT brings intelligence to critical aspects like transportation, industry, payments, health and many others. The interaction between embedded devices and Cloud based web services is a common scenario of IoT deployment. From the security point of view, both users and smart devices must establish a secure communication channel and have a form of digital identity. Most of the times, the usage of IoT devices requires an already existing infrastructure which cannot be controlled by the device owner, for instance in a smart home. This scenario requires a security stack suitable for heterogeneous devices which can be integrated in already existing operating systems or IoT frameworks. This paper proposes a lightweight authorization stack for smart-home IoT applications, where a Cloud-connected device relays input commands to a user's smart-phone for authorization. This architecture is user-device centric and addresses security issues in the context of an untrusted Cloud platform.

Keywords: IoT, Security, Identity, Embedded, Cloud Computing, Smart Home.

*Florin Pop - Corresponding Author
Email addresses: chiforbogdan86@gmail.com (Bogdan-Cosmin Chifor), ibica@mta.ro (Ion Bica), vip@mta.ro (Victor-Valeriu Patriciu), florin.pop@cs.pub.ro (Florin Pop)

1. Introduction

The Internet of Things and the embedded devices are becoming ubiquitous computing elements in our lives. These devices are used in many areas, from industrial, health, transportation to smart city and smart home scenarios. The adoption rate of these computing elements, especially in the smart home area, depends on the security level provided by the applications. Privacy is an important element for regular users and the IoT (Internet of Things) enabled application deployed in a smart home must be designed with a robust security mechanism [1]. The implementation of security and privacy features raises a functionality issue, because an IoT solution comprises multiple elements: embedded devices, user interface elements, cloud computing for data processing, device control and many others [2], [3].

The IoT solution consists of both open-source and proprietary elements, and some of them cannot be controlled by the user. Regarding the smart home, a common IoT deployment scenario consists in a plethora of embedded devices connected to a cloud solution which permits the data processing and unifies the devices access and control. In such a scenario the user can control remotely, the smart home devices by means of a smart-phone application [4].

Taking into consideration this conditions, this papers proposes a lightweight security solution which eases the authentication and authorization mechanism even if an untrusted cloud is used for data processing and transport. We designed a federated authorization mechanism, adequate for a cloud solution which controls the IoT device with the user's approval [5]. In order to design a robust security solution, a smart-phone component was implemented along with a password-less authentication protocols which is highly supported by many device manufacturers, using the FIDO ("Fast IDentity Online") model.

The main contributions of this paper are:

- we proposed a lightweight identity stack for IoT, in order to provide a digital identity for both the smart devices and for the users which interact with them;
- we analyzed the integration of our proposed solution in an already existing software and hardware architecture, requiring minimum costs from the user side;
- we device a Cloud federated authentication for smart home by underling the FIDO authentication messages;
- we proposed a theft resistant security scheme using a keep-alive protocol that is executed periodically and every time the user request a FIDO authentication through the cloud platform;
- we conducted experiments using Kaa IoT Cloud tested network topology and measure the delay times, which are very important to ensure near real-time constraints of a smart-home application, which are respected for our proposal.

The paper is structured as follows. Section 2 presents related work in the files analyzing several solutions already validated by scientific community. Then, in Section 3 we describe the proposed architecture, presenting the security system general scheme by adding a FIDO extension. The system implementation using Kaa IoT cloud platform is presented in Section 4. Experimental results are analyzed in Section 5. The paper ends with conclusions and future work, presented in Section 6.

2. Related work

Both mobile and embedded devices need mature authentication stacks in order to be deployed in security sensitive areas. Although there are multiple authentication mechanism suitable for resource constrained devices, there is a lack of standardized solutions accepted by software developers and hardware manufacturers. In [6] Lindemann et al. present a universal authentication framework protocol, called FIDO, which provides a passwordless authentication mechanism. This protocol is widely adopted by many organizations and has an important market share. The FIDO protocol is used as a first factor authentication mechanism and uses cryptographic keys instead of passwords. The FIDO cryptographic keys are stored in a module called authenticator which is unlocked by the user using biometrics or other security mechanisms.

Despite the fact that FIDO was designed to address security issues on smartphones, the protocol is considered adequate for IoT devices. For instance Hannes Tschofenig argues in [7] that an FIDO architecture can be applied to an IoT network. In this paper are presented the most important IoT design patterns: the cloud based connectivity and the local area based connectivity. The author presents a security architecture where users can authenticate to the IoT devices by using a federated protocol. The last scheme uses OAuth along with the FIDO protocol. In [8] Yoon Miyeon and Baek Jonghyun designed an IoT security architecture where end-point IoT devices authenticate to the IoT gateway using the FIDO protocol. This work also stresses the importance of the security by design principle when developing an IoT software product.

A similar effort to FIDO has been done by Frank Stajano in designing PICO [9], a security system which targets the authentication problem. As FIDO, PICO employs cryptographic keys for a passwordless authentication mechanism, the system being scalable and feasible for embedded devices. Moreover PICO complements an authentication mechanism with theft-resistant and loss-resistant by implementing a keep-alive protocol between the embedded devices owned by the same user. Securing the communication between the IoT devices is a critical issue from the security point of view. A lot of work has been done in this direction, with DTLS as *de facto* standard. In the IoT paradigm, DTLS [10] is used in conjunction with CoAP (Constrained Application Protocol) [11]. Even though DTLS is a mature solution for establishing a secure communication session, authentication is an issue because the IoT devices have no pre-operational digital identity. Beside using digital certificates, a DTLS session can authenticated using pre-shared key mechanisms or raw public keys. In [12]

was designed a scalable pre-shared key management mechanism for DTLS which includes among others a session key derivation, key expiration and key revocation. In our work a user's smart-phone, considered a trust anchor, distributes pre-shared keys to IoT devices to be used for authenticating DTLS sessions and for challenge-response based CoAP messages exchanged in the HAN (Home Area Network). IoT systems are mainly comprised of embedded devices with limited computing capabilities while having a cloud component which processes the data and delivers it to the end-users. Although the IoT Cloud solutions are not standardized they often follow the same design pattern having similar security requirements.

In [13] Raham et al. show a general scheme where IoT devices are integrated with a cloud environment. The latter presents the vulnerabilities and the security issues which must be handled by the IoT systems. In [14] Horrow et al. present an authentication and authorization framework for the IoT cloud connected devices. The scenario presented in the previous work consists in an embedded network where the data between the sensor and the actuator is transfered using a cloud system.

In a smart-home environment the devices provide complex services by interacting with each other, thus communicating with a trusted and uncompromised module is critical. In [15] Barreto et al. realized an architecture where TPM (Trusted Platform Module) equipped IoT devices are part of an authentication model. This system uses digital certificates and it is role based. The role based system is suitable for an IoT deployment scenario because it provides a segregation between different tasks performed by the users (owner) and by the administrators (device manufacturer). In [16] Abera et al. present various attestation methods for the IoT devices. Among the attestation methods, they stress the Intel SGX, software and even hybrid solutions suitable for resource constrained IoT devices.

Privacy protection is a security issue taken into consideration by end-users when interacting with IoT enabled applications. By disclosing a real user identity a potentially malicious IoT device could correlate user actions and leak private data. This security issue is addressed by Alpár et al. in [17] by stressing in a position paper the potential of privacy preserving attribute protocols like U-Prove or Idemix. In the previous paper are presented the advantages of using privacy preserving attribute protocols in a smart home, because devices can authorize the users without knowing their identity. In [18] Bethencourt et al. present an attribute based cryptography scheme called CP-ABE (Ciphertext Policy - Attribute Based Encryption). This scheme permits a role based encryption policy, having multiple applications, including IoT security scenarios.

Even though it is considered out of the scope of this paper, the mutual authentication between devices is an issue which must be analyzed in the smart-home context. Devices could interact with each other by means of resource discovery mechanisms, thus needing a *device-to-device* mutual authentication. Untrusted transient devices could connect to HAN and by enforcing a device-to-network mutual authentication a series of security attacks will be mitigated. Integrating IoT devices is difficult from the security point of view because the

devices do not have a digital identity when purchased or a security anchor to facilitate their authentication process. In our particular scenario the mutual authentication between devices is addressed by using pre-shared keys in the context of the keep-alive theft-resistant protocol. In a complex smart-home scenario, where devices interact with each other in an ad-hoc manner, this mechanism has scalability and user-experience issues. In [19] and in EAP-NOOB (Extensible Authentication Protocol - Nimble Out Of Band) Internet Draft has been designed an authentication framework adequate for IoT devices. The previous solution is based on EAP messages which rely on a user managed out of band channel to facilitate the device mutual authentication. The EAP-NOOB performs a Diffie-Hellman key exchange and relies on the user (out of band secure channel) to mitigate a MITM (Man-in-the-middle) attack. The security framework presented in this paper can be integrated as an out-of-band authentication mechanism of the EAP-NOOB protocol, thus facilitating a user-controlled mutual authentication between devices. In our future work we will extend the security framework presented in this paper with an EAP-NOOB plug-in in order to design a adequate mutual authentication mechanism for complex smart-home IoT applications. Concerning the portability, the message transport system is solution agnostic and the architecture could be migrated to any IoT Cloud system.

This paper presents a lightweight security stack where users can authenticate to their smart home IoT devices through an untrusted cloud system. We propose a passwordless authentication method using FIDO, where the FIDO server is a module hosted by the IoT device, deployed in a smart home scenario. The proposed solution is a practical one and it is tested using one of the most mature cloud software stack for IoT devices.

We augment the FIDO security solutions with PICO elements in order to provide a theft-resistant solution and we add to FIDO a privacy preserving extension in order to protect the user identity and to permit an attribute access controlled method to the IoT device for administrative purposes.

The security solution proposed in our work can be integrated in an already existing software and hardware architecture, requiring minimum costs from the user side. Among other things, our work encompasses security elements described in the related work papers and proposes a complete architecture with a clear vision regarding the software product life-cycle.

3. System architecture

The system architecture comprises three main entities: the smart home IoT devices, the cloud platform and a smart-phone management application. The first module consists in the IoT heterogeneous devices deployed in a smart home environment.

The cloud platform is in charge with processing the IoT devices data and the smart-phone management application is used to remotely control the smart home devices. In this scenario the cloud platform is considered to be an untrusted entity, not necessary a malicious one. The cloud platform can be man-

aged by a device manufacturer or can be considered a third-party service. This system has two main tasks: data processing and data transport. The data processing service can include aggregation or machine learning services. The second tasks facilitates the communication between the user and the devices or between the devices placed in different locations.

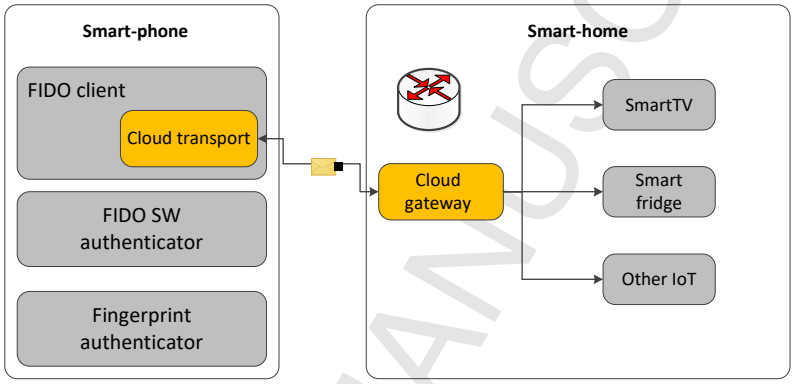

Figure 1: The security system general scheme.

The security system general scheme is presented in Figure 1. For the authentication mechanism, the FIDO UAF (Universal Authentication Framework) protocol was chosen. FIDO UAF is a passwordless first factor authentication protocol developed by the FIDO Alliance. This protocol employs cryptographic key pairs stored on a module called authenticator. In order to use these keys, the user must unlock the authenticator using biometrics, pin or other security mechanisms. We chose FIDO because it has a good support from the smartphone manufacturers, probably with TEE (Trusted Execution Environment) implementations for the authenticator module. Although FIDO is currently used for accessing web resources, in this paper it is used to access a physical resource: a smart home embedded device. The embedded device acts as a FIDO server and an Android management application controlled by the user, acts as a FIDO client. The authenticator module has a software implementation and uses OpenTEE in order to have a TPM (Trusted Platform Module) compliant interface.

After the user purchases a new IoT device, it must be installed in the smart-home environment. The device install consists in a user-to-device registration process. This process consists in imprinting the embedded device with the user FIDO public key. This process can take place only if the device is not already imprinted with a cryptographic key. This registration process is executed in the smart-home context, by using the local Wi-Fi connection. In order for the registration to begin, the user must prove to the IoT device a proper authorization.

This feature is realized by adding a FIDO extension which consists in a authorization data, computed using a privacy preserving attribute protocol. In this scenario, the device manufacturer issues an anonymous token to a user

who purchased a device. The user presents this token in a customized FIDO registration protocol. By performing this protocol, the IoT device knows that the user who wants to imprint its FIDO public key is a legitimate purchaser from the device manufacturer.

By applying this security scheme, the device cannot find any private information about the user because no account creation is performed. This scheme can be easily extended by the manufacturer by adding multiple attributes, related to the package purchased by the user. During the registration process, the IoT device must present to the FIDO client (the Android management application) a list of allowed authenticators. By using this method, the IoT device can select only the trusted FIDO authenticators, this being a feature of the FIDO protocol. When engaged in the FIDO registration protocol, the client side authenticator must sign a device generated random challenge with an attestation key. The authenticator module stores an X.509 attestation certificate along with the associated private key. The attestation certificate is signed by the authenticator module manufacturer. The attestation method permits the IoT device to trust a third-party authenticator (produced by another manufacturer) only by employing the trusted attestation certificate chain.

The FIDO protocol has a un-registration message which has the purpose of removing the association between an account and the cryptographic key stored on the server side. This type of FIDO message is considered out of the scope of this paper because it cannot be applied in a usage or security scenario for the proposed security scheme.

The scenario presented in this paper assumes that the smart home devices are connected permanently with the device owner through a cloud platform which is not controlled by the user. By employing this security mechanism, a user can be authenticated by the smart home device. The successful user authentication process assures the device that the command which is being issued originates from the real owner. The FIDO messages are transported using the cloud platform internal messaging system: MQTT (Message Queueing Telemetry Transport), HTTP (Hypertext Transfer Protocol), CoAP (Constrained Application Protocol) or others. The FIDO authentication protocol implies three entities: the authenticator, the client and the server. In our scenario the authenticator is a TPM compliant software module which runs as an Android application. This module authenticates the user by means of fingerprint. The user-to-device authentication is realized by using a series of security functionalities of the Android 6 operating system.

A scenario where the smart home IoT devices are connected to a cloud platform requires a particular data flow from the device to the cloud. The cloud platform can query the IoT devices in order to extract certain application specific data. The same scenario could result in certain commands issued by the cloud platform to the device. These commands could collude with a user-defined policy, thus resulting in a security breach. This paper proposes a FIDO based authorization framework, where each cloud issued command to the IoT device must be authorized by the user. When the cloud tries to access certain data from the device, the latter computes an authorization policy which must

be approved by the user. The user receives the authorization policy by means of a FIDO protocol extension.

In the first sequence there is a request from the cloud module to the IoT device. After the IoT device detects an unauthorized request, it redirects the cloud platform to the user's smart-phone. The user is shown the authorization policy by means of a user interface. If the user agrees with the data requested by the cloud platform it begins a FIDO authentication protocol with the IoT. After the FIDO authentication occurs, the cloud platform is signaled by the user to request the data again. This time the IoT device compares the policy data against the local policy database. If the hash of the policy is authorized then it delivers the requested data to the IoT device. This security scheme is scalable because it can be extended to other devices from the smart home. By using such a policy, the communication between cloud and a certain device or the communication between devices can be controlled and authorized by the user.

4. System design and implementation

Regarding the implementation, multiple open-source IoT cloud platforms were considered. The chosen solution was IoT Kaa [20], one of the most mature solutions available. This cloud platform has a web interface for management and a code generator module for multiple programming languages. For instance one could generate a Kaa end-point API for Android platform (using Java), for iOS devices (using Objective-C) or for other embedded devices (using C/C++). The generated code can be linked to third-party applications in order to develop Kaa enabled solutions. Kaa provides a protocol agnostic messaging system where multiple devices placed in a group can exchange messages. The most important Kaa types of messages are the following: events and notifications. The events are messages generated by an endpoint and distributed to others end-points. The notifications are messages originating from the cloud platform and which targets the action of an IoT device. In this paper the events and notifications are used to transport the FIDO registration and authentication messages.

Regarding the FIDO protocol we used a Java implementation open-sourced by eBay. The eBay implementation comprises three main parts: a client module (Android), a server module (Apache Tomcat and Jersey) and a FIDO core module. We added an Open-TEE based implementation to the authenticator elements from the FIDO core, in order to provide a hardware-independent TPM solution.

The FIDO registration process has the purpose of establishing a trust relation between the user and the device, by imprinting the device with the user's public FIDO key. The imprinting process does not involve the untrusted cloud entity. One important feature of this process is the fact that the registration is accountless, the user identity being preserved. In order to achieve this, the user authenticates to the IoT device with the attributes issued by the device manufacturer. The attributes can be application specific and certain attributes activate different device functionalities. For the attribute based authorization,

we used CP-ABE (Ciphertext Policy Attribute Based Encryption) from the ACSC (Advanced Crypto Software Collection) implementation. In this architecture, the device manufacturer issues to the user a collection of attributes along with a private key. The private key is used to decrypt a nonce, generated by the device, in a FIDO registration challenge-response protocol.

The device manufacturer issued attributes are the administrator status and the subscription period. The attributes are transferred to the user Android application and are used in the device interaction process. The device hosts a CoAP web server used for out-of-band management (the data is not transferred through the cloud platform). This server is used in the initial FIDO registration protocol. After the FIDO registration takes place, subsequent user-to-IoT device authentication takes place by employing the FIDO keys. The CP-ABE logic artifacts are transported in the FIDO messages, in order for the solution to be FIDO compliant. After the user application discovers the IP address of the IoT device, it accesses a registration URL, which responds with a FIDO registration request message.

The challenge field is generated by the IoT device, being a random value in the FIDO protocol. The importance of the challenge field in this paper is augmented, because it represents a nonce encrypted with the administrator attributes. By using this technique, the client module will decrypt the challenge, using the CP-ABE private key generated by the manufacturer. The challenge decryption process is handled by the FIDO user agent which routes the modified FIDO registration request message to the authenticator.

The authenticator will process the message as specified in the FIDO protocol and will return to the IoT device a FIDO *RegistrationResponse* message. Because the custom FIDO processing is handled outside the authenticator, third-party hardware authenticator devices or built-in smart-phone authenticators can be used to establish a trust relation with the IoT device.

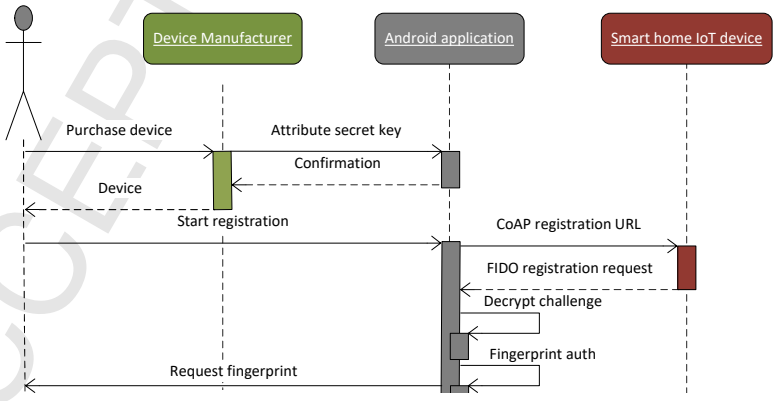

Figure 2: Smart home device registration.

The FIDO registration process is presented in Figure 2. After the FIDO

registration process is completed, subsequent messages between the user and the device can be exchanged by using the untrusted cloud platform. All the messages will be protected by using the FIDO authentication protocol.

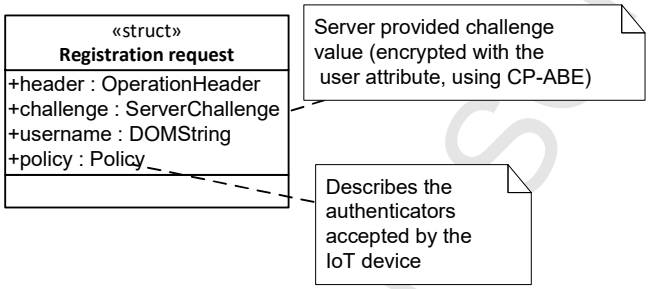

Figure 3: Registration request message format.

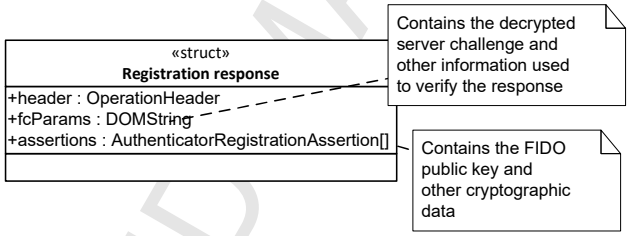

Figure 4: Registration response message format.

The structure of the FIDO registration request and registration response messages are detailed in Figure 3 and Figure 4.

4.1. User authentication

When the user wants to communicate with the IoT device, it will send a JSON formatted command to the device via the cloud platform. After receiving the command message, the device will reply back with a FIDO authentication request: header, challenge, transaction, policy. The transaction field is an array, where each element has the following structure: content type, content and display characteristics. In the FIDO protocol, the transaction field represents an action which must be authorized by the user. In this paper, the transaction field represents the original command message which triggered the authentication process. By using this technique, the user can verify that the device received an authentic command message. In the authentication response message, the IoT device waits for a signature value computed over a challenge and other fields.

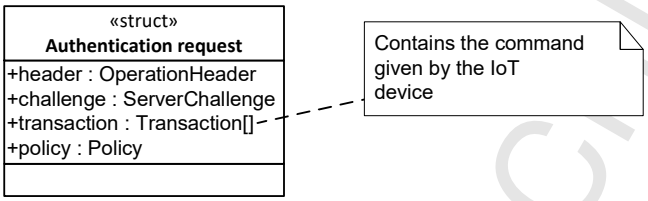

Figure 5: Authentication request message format.

The FIDO authentication response contains the following fields: *headers*, *fcParams* (cryptographic elements) and *signature assertions*. After the authentication response is validated, the IoT device executes the command issued by the user.

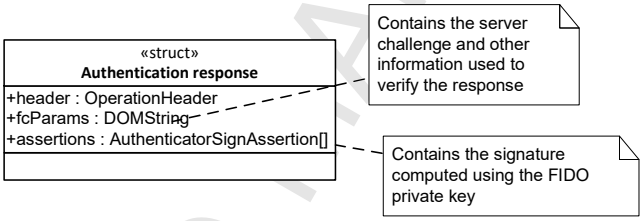

Figure 6: Authentication response message format.

When verifying the FIDO authentication response, the IoT device searches into the internal database the permissions associated with the FIDO public key which is the current authentication process. In this scenario, the IoT device authenticates the user, because the latter has to authenticate himself to the FIDO authenticator module using biometrics means (fingerprint). The structure of the FIDO registration request and registration response messages are presented in Figure 5 and Figure 6.

Taking into consideration the fact that the cloud entity is considered untrusted, a level of privacy must be designed in order to protect the messages (commands) exchanged between the controller and the IoT device against eavesdropping. Even though the message integrity is guaranteed by the protocol described above, the cloud entity could log the messages issued by the user, aggregate data and gather private information about the device owner. In order to mitigate this security issue we designed an encryption layer which protects the user-to-IoT device and the IoT device-to-user messages.

During the IoT device initialization a pair of ECC keys are generated and the public key can be retrieved via a CoAP URL. The same process is executed at the controller side and the public keys are exchanged during the device imprinting stage. Every message exchanged between the controller and the device is encapsulated in a cryptographic datagram which has a lightweight structure

with a series of fields encoded using a type-length-value method. The message payload is encrypted with a symmetric algorithm and a new key is generated for each message. The symmetric encryption key is then encrypted with the ECC public key and all the data is encapsulated in a datagram which contains the following fields:

- Protocol version;
- Asymmetric cryptographic algorithm identifier;
- Symmetric cryptographic algorithm identifier;
- Encrypted symmetric key;
- Encrypted payload.

4.2. Cloud federated authentication

The cloud federated authentication is an extension of the user authentication mechanism. This process also underlies the FIDO authentication messages. The federated scenario is used when the cloud platform needs to process the IoT device data which requires the user authorization. When the cloud platform sends a JSON formatted command to the IoT device, the latter processes it and sends a FIDO authentication request to the user.

The cloud issued command is stored in the transaction field of the FIDO authentication request. If the user authorizes the command, it sends back to the IoT device a FIDO authentication response. The device owner authorizes a cloud command, after visualizing the required actions on the smart-phone screen. This process is stateful on the IoT device side and after the FIDO message verification occurs, the device can send data to the cloud platform or can execute certain commands.

The federated authentication mechanism is detailed in Figure 7. The messages exchanged between the cloud platform and the IoT device may contain sensitive information, thus the privacy issue must be handled in this scenario. The cloud platform is considered an untrusted entity and the purpose of this paper is to design a security solution which runs above the cloud platform modules. In our scenario the cloud platform has two major components: the server-side solution and the binaries which run on the IoT device and both of these modules cannot be controlled by our security solution.

The cloud platform transmits the messages in a TLS (Transport Layer Security) session, but controlling this aspect is considered out of the scope of this paper. In order to solve the cloud-to-device and the device-to-cloud privacy protection we propose a solution where an IPSEC (Internet Protocol Security) policy is enforced at the gateway side when communicating with the cloud servers. We consider such a solution adequate because the low resource IoT devices will be offloaded from encrypting the communication channel and this security feature will be controlled by a single device [21], [22], [23].

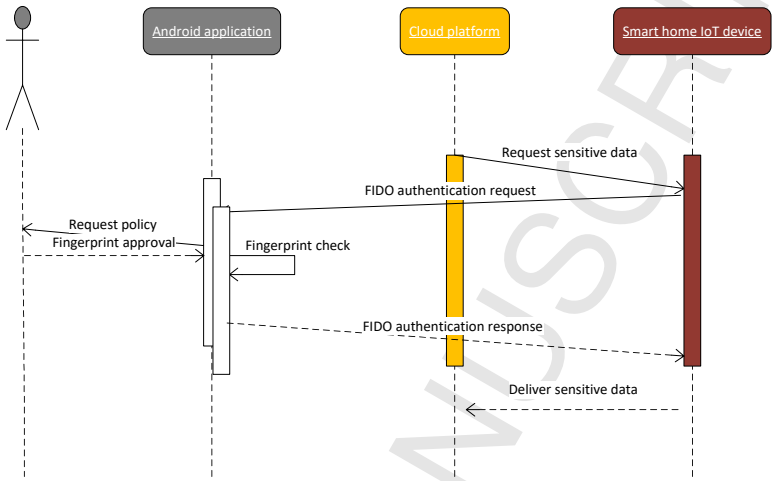

Figure 7: Cloud federated authentication.

4.3. Theft resistant security scheme

One important feature of a security scheme for smart-home IoT devices is the theft-resistant property. This paper proposes a theft-resistant scheme were a user can detect if the IoT device which is issued a command was removed from the smart-home environment. To achieve this, we propose a security scheme similar to the anti-theft solution from the PICO protocol. The PICO protocol solves the theft issue by implementing a feature called PICO siblings. In this scheme, a trust relation is created between multiple PICO devices, using a k-out-of-n architecture. PICO proposes a protocol were keep-alive messages are exchanged between the siblings. If at least k-out-of-n siblings are reachable, then the device will maintain a secret. Otherwise the device will wipe a secret key, thus locking itself. The PICO siblings solution is adapted for devices like PICO authenticators or wearables [24] [25]. We propose a similar solution, where a trust relation is established between the smart-home devices [26], [27]. If the device is relocated and the keep-alive protocols fails the device will lock itself and it will not respond to user input. This security scheme relies on the fact that it would be difficult for an attacker to relocate all the smart-home device siblings in a short time.

The keep-alive protocol consists in a series of messages exchanged via the CoAP protocol. Every smart-home device is a CoAP server and client. The trust relation established between the devices relies on the user management application as security anchor. After the user acquires a new device and imprints it with its FIDO public key, it can choose to create a sibling relation with other smart-home devices. The user creates the trust relation by distributing secret keys to both the devices. The user authenticates to the device, using FIDO messages, transported using CoAP in a DTLS (Datagram Transport Layer Security) session.

After the user is authenticated to the device, it will imprint it with a tuple which consists in a secret key and the sibling device CoAP URL, a random secret key being generated for each two devices. The keep-alive protocol has a simple structure: the client device makes a GET query to the sibling CoAP URL which has as payload a random challenge. The sibling CoAP server will respond with a HMAC (Hash-based message authentication code) message which has as input the client challenge and the secret key. We also implemented another variant of the keep-alive protocol where the devices exchange messages transmitted in a DTLS session with pre-shared key (secret key) as mutual authentication mechanism.

The keep-alive protocol is executed periodically and every time the user request a FIDO authentication through the cloud platform. If the sibling message exchange cannot be realized, the device will lock itself. This security scheme has a k-out-of-n structure in order to avoid a device lockout if a sibling is unable to respond to the keep-alive protocol at a certain time.

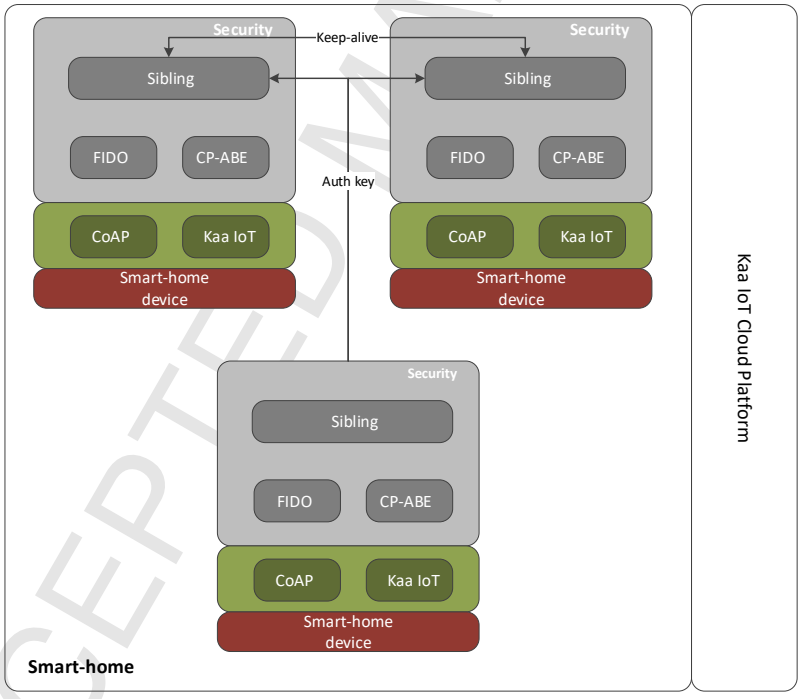

Figure 8: Smart home siblings security scheme.

The siblings security scheme is explained in Figure 8. If the device does not receive k-out-of-n keep-alive messages, then all the FIDO public keys and the manufacturer attribute keys are wiped. By executing this, the owner cannot be lured in connecting to a compromised stolen device. Also, because the manufacturer keys are deleted, the device cannot be imprinted with other FIDO public

keys.

5. Experimental results

In order to evaluate the authentication protocol proposed in this paper, we simulated a smart-home network scenario. We evaluated the delay caused by the protocol compared to a scenario where the security is delegated to the IoT cloud platform. In a typical smart-home scenario there is a central node (typically the user with his smart-phone) which sends different actuation commands to nodes (smart devices from home). When using an IoT cloud platform, like Kaa IoT, the central node (controller) can send unicast or broadcast commands to the slave nodes. We tested a scenario where the controller sends a broadcast command (like a start/stop command) to all the smart-home devices and we measured the mean delay time after which the devices received the message.

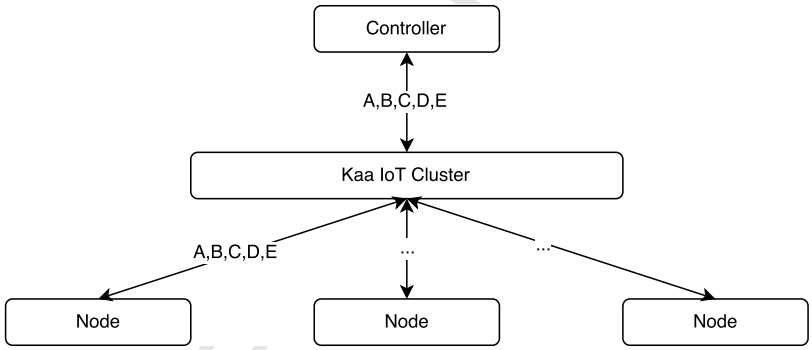

Figure 9: Kaa IoT cloud tested network topology.

To determine the authentication protocol overhead, we tested a scenario where the controller sends a broadcast message to all the devices and then initiates a unicast authentication protocol with each device. In this test case we observe the mean delay time necessary for the actual command (after the authentication messages are transmitted) to be received by all the devices. The Kaa IoT platform is open-source and can be deployed in various ways: directly into the Cloud, as a sandbox running in a virtual machine, it can be installed on a Linux distribution (Debian and RPM packages) or it can be compiled. For the experimental part of our work we used an Ubuntu 16.04 LTS Linux distribution as a host operating system with the Kaa IoT platform deployed in a Virtual Box environment.

For measuring the delay caused by the additional authentication commands we used a bash script which orchestrates the simulated IoT network. The IoT nodes are simulated by instantiating the same code multiple times. The controller code and the IoT node code is not identical because they employ different command types. The orchestration script instantiates an IoT device by performing the following steps: a new directory is created for each node, the executable

code is copied in the directory and each node process is started. Each node process is deployed in a different directory because a unique data is generated by the Kaa SDK library during the node runtime. After the initialization steps, the script waits a certain amount of time necessary for all the nodes to connect to the Kaa server. With all the nodes being connected to the server, the controller process is launched and the simulated message exchange is started. The orchestration scripts waits for all the communication messages to be exchanged, stops all the previously launched processes, collects the logs from each simulated IoT node and computes the mean delay time.

The Kaa SDK used by the controller and by the nodes was generated by defining an application type and multiple message types in the Kaa IoT web management platform. In the Kaa IoT paradigm each node has a unique network address and a digital identity when the node boots (in our case when the process is started). A Kaa IoT message type (event) is defined in the web platform by providing a JSON schema and the message direction: a node can be a source, a sink or both. Different message types are grouped in event class families and all this information gets embedded in a SDK (JAR file) which is linked with the client application code. By using the Kaa IoT abstract modules a developer can target multiple platforms by generating a Kaa SDK in multiple programming languages. Kaa SDK generation feature a developer can target multiple platforms.

Each message has a simple structure with two fields represented as *String* data types: payload and timestamp. The message types employed by the nodes and by the controller were defined in the Kaa IoT web management by using the *admin* role. The Kaa *devuser* role was used for defining the message type application mappings and for generating the SDK.

The API exposed by the Kaa client module has an event driven structure with callback methods available for most of the functionalities. For our simulation we designed the message exchange between the controller and the nodes as generic events, thus every IoT node has to subscribe to a topic in order to receive the message. In order to establish an event based communication channel the nodes must be attached to the same user profile, a feature which offers by design a suitable security framework for smart-home IoT devices. After a node attaches to the user profile, it registers callback functions for every message type in order to be able to communicate with the controller.

The authentication protocol contains the following message types:

1. Command request (broadcast sent from controller to nodes);
2. Authentication request (sent from node to controller);
3. Authentication response (sent from controller to node);
4. Authentication confirmation (sent from node to controller);
5. Secured command (sent from controller to node).

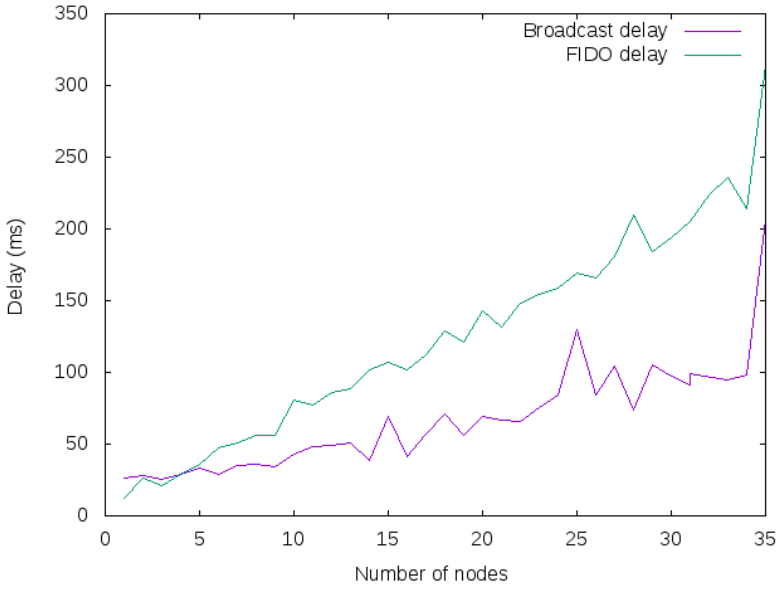

Figure 10: Kaa IoT cloud benchmark delay results.

In Figure 9 is depicted the testing network topology along with the messages transmitted between the controller and the client. In Figure 10 are presented the results for the delay benchmark. As it can be observed, the delay caused by the protocol does not impact drastically. Even though the latency increases along with the number of nodes in the network, the near real-time constraints of a smart-home application are still respected. For this experimental setup we assumed a reliable communication channel were the retransmission of the security protocol messages does not have to be handled by the application layer. In a lossy communication channel retransmitting the security messages will cause additional delays and the presented security protocols will have to be redesigned in order be packet loss tolerant.

Another issue which may cause additional delays are the cryptographic operations. In our simulation we used a desktop device, thus the overhead caused by the asymmetric and symmetric cryptographic operations does not impact the security protocol delays. An aspect which must be taken into consideration is the fact that our simulation assumes identical IoT nodes. In a real-world smart-home there is a heterogeneous environment with low resource embedded devices which could induce additional delays. In order to overcome this issue the controller must handle multiple simultaneous device authentications in an asynchronous manner and keep an authentication session per device. Taking into consideration the fact that the controller software runs on an Android smartphone device we must analyze the test case results from a usability point of view. The recent versions of Android have a strict policy regarding the activity

of a background running process in order to conserve the battery.

In our scenario it is important for a multiple device authorization process to run to completion in order for a command to be enforced on the IoT device side. Because the authorization process has a relatively low overhead the entire process is suitable to run in foreground and thus not being the subject of the Android battery conservation policy. In order to optimize the authorization process and to conserve the Android device battery an authorization session associated with each device is cached a certain amount of time, thus saving a series of user actions and cryptographic operations.

Regarding the implementation of the anti-theft security mechanism we used the Java based Californium library for implementing the CoAP communication. We considered this Java implementation because we can have the same code base for both the Android implementation and for the embedded devices which support a JVM. For implementing the CoAP security communication channel we employed the same library (Scandium sub-module) which supports DTLS 1.2. For the anti-theft keep-alive mechanism we tested a challenge-response security protocol based on HMAC and also a DTLS secure communication, obtaining similar performance penalties. Taking into consideration the smart-home scenario and the reduced frequence for the keep-alive messages we appreciate that the DTLS communication channel with mutual authentication using pre-shared keys is more suitable for protocol similar to ours than a challenge-response mechanism. By using DTLS, a high level security protocol with a variable number of messages can be developed using an out-of-the-box low level communication protocol with minimum performance penalties.

6. Conclusion

Smart-home IoT solutions are still in a early stage, security being a critical factor which could impact their adoption rate. One of the main challenges in designing a software security solution for smart-home scenarios is the fact that multiple elements are not under user-control. Another challenge consists in integrating the security solution with already existing software infrastructure, while taking into consideration user-experience elements.

In this paper we presented a device authorization scheme for smart-home IoT devices which are connected to an untrusted cloud system. In addition we presented a proof-of-concept software solution which uses the FIDO protocol for users to authenticate to their devices. Moreover, we proposed a series of extensions and usages of the FIDO protocol, suitable for an IoT deployment scenario. The protocol presented in this paper preserves the user anonymity: manufacturers cannot create a linkage between different user accounts, because the only user related information is the FIDO public key and a pseudonym.

Experimental results have shown that the additional delay introduced by the FIDO authentication protocol has a low impact for a smart-home application. Regarding the future work, our efforts will focus on the following directions: testing the FIDO protocol with hardware authenticators, improving

the current security scheme with FIDO U2F elements (second factor authentication), adapting the software security stack for multiple hardware platforms and programming languages and proposing an IoT device attestation scheme for smart-home, using the security primitives from this paper.

The *user-to-device* interaction is critical to our scenario taking into consideration the user-device centric character of the proposed security framework. The authorization messages presented in this paper are secured through the FIDO protocol and unlocking the FIDO private key requires biometric (fingerprint) authentication on the user-device side. The *user-to-device* authentication relies on the Android security framework and on the smart-phone hardware security modules (like fingerprint reader or ARM TrustZone).

For future directions we would like to extend our simulation setup to a RIoT virtual network and deploy the Kaa IoT application on RIoT operating system. Such a setup would be useful in order to observe the protocol overhead when injecting packet loss in a network and when the clients are deployed in a resource-constrained operating system. We intend to extend our testing scenario to real embedded devices in order to observe the protocol overhead in real-life conditions. Other important parameter for testing is the overhead of attaching the node to the a user account. We plan to test this feature with the Google+, Facebook and custom authentication modules. Also another future step would be testing the protocol presented in this paper with other IoT cloud providers like AWS IoT.

Acknowledgment

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS - UEFISCDI, project number PN-II-RU-TE-2014-4-2731 - *DataWay*: Real-time Data Processing Platform for Smart Cities: Making sense of Big Data.

References

[1] C. G. Garcia, D. Meana-Llorian, B. C. P. G-Bustelo, J. M. C. Lovelle, N. Garcia-Fernandez, Midgar: Detection of people through computer vision in the internet of things scenarios to improve the security in smart cities, smart towns, and smart homes, Future Generation Computer Systems (2017) –.

[2] C. J. D'Orazio, K.-K. R. Choo, A technique to circumvent ssl/tls validations on ios devices, Future Generation Computer Systems (2016) –.

[3] C. J. D'Orazio, K.-K. R. Choo, Circumventing ios security mechanisms for {APT} forensic investigations: A security taxonomy for cloud apps, Future Generation Computer Systems (2016) –.

[4] C. J. D'Orazio, R. Lu, K.-K. R. Choo, A. V. Vasilakos, A markov adversary model to detect vulnerable ios devices and vulnerabilities in ios apps, Appl. Math. Comput. 293 (C) (2017) 523–544.

[5] N. H. A. Rahman, K.-K. R. Choo, A survey of information security incident handling in the cloud, Computers & Security 49 (2015) 45 – 69.

[6] R. Lindemann, D. Baghdasaryan, E. Tiffany, Fido universal authentication framework protocol, Version v1. 0-rd-20140209, FIDO Alliance, February.

[7] H. Tschofenig, Fixing user authentication for the internet of things (iot), Datenschutz und Datensicherheit-DuD 40 (4) (2016) 222–224.

[8] M. Yoon, J. Baek, A study on framework for developing secure iot service, in: Advances in Computer Science and Ubiquitous Computing, Springer, 2015, pp. 289–294.

[9] F. Stajano, Pico: No more passwords!, in: International Workshop on Security Protocols, Springer, 2011, pp. 49–81.

[10] Z. Shelby, K. Hartke, C. Bormann, The constrained application protocol (coap), https://tools.ietf.org/html/rfc7252.

[11] C. Bormann, A. P. Castellani, Z. Shelby, Coap: An application protocol for billions of tiny internet nodes, IEEE Internet Computing 16 (2) (2012) 62–67.

[12] S. Raza, S. Duquennoy, J. Hoglund, U. Roedig, T. Voigt, Secure communication for the internet of things - a comparison of link-layer security and ipsec for 6lowpan, Security and Communication Networks 7 (12) (2014) 2654–2668.

[13] A. F. A. Rahman, M. Daud, M. Z. Mohamad, Securing sensor to cloud ecosystem using internet of things (iot) security framework, in: Proceedings of the International Conference on Internet of things and Cloud Computing, ACM, 2016, p. 79.

[14] S. Horrow, A. Sardana, Identity management framework for cloud based internet of things, in: Proceedings of the First International Conference on Security of Internet of Things, ACM, 2012, pp. 200–203.

[15] L. Barreto, A. Celesti, M. Villari, M. Fazio, A. Puliafito, An authentication model for iot clouds, in: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015, ACM, 2015, pp. 1032–1035.

[16] T. Abera, N. Asokan, L. Davi, F. Koushanfar, A. Paverd, A.-R. Sadeghi, G. Tsudik, Invited-things, trouble, trust: on building trust in iot systems, in: Proceedings of the 53rd Annual Design Automation Conference, ACM, 2016, p. 121.

[17] G. Alpár, L. Batina, L. Batten, V. Moonsamy, A. Krasnova, A. Guellier, I. Natgunanathan, New directions in iot privacy using attribute-based authentication, in: Proceedings of the ACM International Conference on Computing Frontiers, ACM, 2016, pp. 461–466.

[18] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: 2007 IEEE symposium on security and privacy (SP'07), IEEE, 2007, pp. 321–334.

[19] M. Sethi, E. Oat, M. Di Francesco, T. Aura, Secure bootstrapping of cloud-managed ubiquitous displays, in: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '14, ACM, New York, NY, USA, 2014, pp. 739–750.

[20] KaaIoT Technologies, LLC, Kaa Open-Source IoT Platform 2017 - IoT cloud platform the Internet of Things solutions and applications that set the standard. (n.d.)., `https://www.kaaproject.org/`, online; Retrieved May 16, 2017.

[21] C. Esposito, A. Castiglione, K.-K. R. Choo, Encryption-based solution for data sovereignty in federated clouds, IEEE Cloud Computing 3 (1) (2016) 12–17.

[22] A. Castiglione, F. Palmieri, K.-K. R. Choo, Enhanced network support for federated cloud infrastructures, IEEE Cloud Computing 3 (3) (2016) 16–23.

[23] O. M. Achim, F. Pop, V. Cristea, Reputation based selection for services in cloud environments, in: 2011 14th International Conference on Network-Based Information Systems, 2011, pp. 268–273.

[24] Q. Do, B. Martini, K.-K. R. Choo, Is the data on your wearable device secure? an android wear smartwatch case study, Software: Practice and Experience 47 (3) (2017) 391–403.

[25] N. D. W. Cahyani, B. Martini, K.-K. R. Choo, A. Al-Azhar, Forensic data acquisition from cloud-of-things devices: windows smartphones as a case study, Concurrency and Computation: Practice and Experience.

[26] G. Carullo, A. Castiglione, G. Cattaneo, A. De Santis, U. Fiore, F. Palmieri, Feeltrust: providing trustworthy communications in ubiquitous mobile environment, in: Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on, IEEE, 2013, pp. 1113–1120.

[27] C. Esposito, M. Ficco, F. Palmieri, A. Castiglione, Interconnecting federated clouds by using publish-subscribe service, Cluster Computing 16 (4) (2013) 887–903.

***Biographies (Text)**

Eng. Bogdan-Cosmin Chifor received a MS in Computer Science from the Military Technical Academy of Bucharest in 2014. He has an undergoing PhD at the same university with a theme entitled "Internet of Things Security". Mr. Chifor is a security software developer. His field of interest include electronic signature schemes, applied cryptography and developing embedded security solutions. Bogdan Chifor has 4 years of experience in development and research. He is the author of 4 publications in international conferences and journals. He collaborates closely with the industry on data security solutions. During the last 4 years, he implemented and developed many security software and hardware solutions related to data encryption, digital signatures, PKI enabled applications, embedded kernel development (Linux and FreeBSD), FPGA based security solutions, cryptographic IPSEC based appliances and Android VoIP security solutions. He was involved in a research project where he was in charge with developing an FPGA based security solution and currently participates in a EU funded H2020 project which targets mobile security solutions.

Professor Ion Bica leads the Faculty of Military Electronic and Information Systems at Military Technical Academy of Bucharest. He received his PhD in Computer Science at the same academy in 2004 with a thesis on electronic signatures and trusted third parties. He is teaching about computer networks security and security evaluation of information systems. His main research interests are cryptographic algorithms and protocols for data security. Currently, he is interested in efficient security solutions for cloud and IoT. He published 6 books, and more than 80 papers in journals and conference proceedings, co-edited 2 conference volumes published by Springer and delivered invited talks at many universities and international conferences. He is active member in several COST Actions and NATO Working Groups on cyber security.

Professor Victor-Valeriu **PATRICIU** received a PhD in Computer Science at the Politehnica University of Bucharest in 1992 with *"Magna cum laude"* distinction. His main research interests are in the field of cyber defense and information security. Prof. Victor-Valeriu PATRICIU is PhD adviser from 1997 in Military Technical Academy Doctoral School of Electronic, Informatics and Communication Systems for Defense and Security, in Computer Science & Information Technology domain. Professor Victor-Valeriu PATRICIU is visitng/associate professor at ENSIETA Brest, France, Institute National Polytechnique, Toulouse France, Academy of Economic Studies Bucharest, BRIE- Bulgarian-Romanian Inter-University Center of Europe, Titu Maiorescu University. He has published 15 books and courses, more than 100 papers in the last 10 years on the security informatics and cyber defense area. He was / is a member ACM (Association for Computing Machinery), USA, IFIP/WG11.8 Committee on Information Security Education and Training- member, AFCEA – member of the Romanian Board, New York Academy of Science- member, PfP Consortium Working Group "Impact of Information Technologies on National Security"-member, NATO Research and Technology Organization, Information Systems Technology Working Group member representing Romania. Professor Victor Valeriu PATRICIU has remarkable contributions in the fields of Information Security developing, deploying and auditing, electronic signatures and PKI, Biometrics, Cyber defense, Ad-hoc networks, IDS, honeypots systems, cloud security, IoT etc. The main contributions in the field of cyber security are also the developing the higher education field of security informatics in Romania; his major contributions to the development of Security of Information Technology Master program in Military Technical Academy; his contribution in research on Cryptographic mechanisms, services and protocols, Electronic Signature, PKI infrastructures and bridge technologies, Smart-card applications, Electronic payment systems, Security of network protocols, E-mail and document security.

Professor Florin Pop received his PhD in Computer Science at the University Politehnica of Bucharest in 2008 with "Magna cum laude" distinction. His main research interests are in the field of large scale distributed systems concerning scheduling and resource management (decentralized techniques, re-scheduling), adaptive and autonomous methods for data handling, IoT and Big Data platforms, multi-criteria optimization methods, middleware tools and applications development (satellite image processing and environmental data analysis), prediction methods, self-organizing systems, data retrieval and ranking techniques, contextualized services in distributes systems, evaluation using modeling and simulation (MTS2). Florin Pop was awarded with two Prizes for Excellence from IBM and Oracle, three Best Paper Awards (in 2013, 2012, and 2010), and one IBM Faculty Award. He is involved in many national and international research projects (6 as project leader). He is active reviewer for several Journals (TPDS, FGCS, ASOC, Soft Computing, Information Sciences, etc.) and he has acted as Guest Editor for several special issues in FGCS and Soft Computing. The results were published in 4 books, more than 10 chapters in edited books, over 30 articles in major international peer-reviewed journal, and over 100 articles in well-established international conferences and workshops. He has long-running collaborations with several institutes from EU and around the world, including INRIA Rennes (KerData team), VU Amsterdam (The Netherlands), and University Marie and Pierre Curie Paris 6 (France).

***Biographies (Photograph)**

Eng. Bogdan-Cosmin Chifor

Professor Ion Bica

Professor Victor-Valeriu **PATRICIU**

Professor Florin Pop