Using Structural Diversity to Enforce Strong Authentication of Mobiles to the Cloud

Samy Kambou, Ahmed Bouabdallah

Using structural diversity to enforce strong authentication of mobiles to the cloud

Samy Kambou & Ahmed Bouabdallah
IMT Atlantique
IRISA, UBL
F-35576 Cesson Sévigné, France
{samy.kambou, ahmed.bouabdallah}@imt-atlantique.fr

***Abstract*—Modern portable devices such as smartphones are enhanced by advanced functionalities and may therefore soon become both the preferred portable computing device (thereby substituting laptops) and the personal trusted device. They are also increasingly used to access to online cloud services, including those particularly sensitive which require high security. This paper introduces an original and strong authentication method for mobiles. It involves a two factor scheme enhanced through network channels and devices diversity. Our solution combines an OTP-based approach using an IoT object as secondary device in addition to the smartphone. The diversity of the network's channels rests on the use of one of the LPWAN networks together with LTE or WIFI networks. Authentication factors are therefore transmitted over different channels through different devices thus greatly reducing the attack surface. The proposal is also enhanced by end-to-end encryption of the transferred sensitive contents. The link with the authorization issues is analyzed and the integration of our approach to OpenID Connect/OAuth 2.0 is investigated. A platform that implements this scheme has been developed, tested and evaluated under different attack scenarios.**

***Index Terms*—strong authentication, multi-factor authentication, internet of things, one-time password, diversity, cloud-based web service, OpenId Connect**

I. Introduction

Due to the rapid development of wireless technologies, Internet services and cloud capabilities, many companies are deploying part of their information system in the cloud and opening it to their different partners. It is therefore crucial to know the company's protected resources and services and to restrict the access to legitimate users. Mobile network operators for example define subscriber profiles which are made of several kinds of services. Customer who has subscribed to particular services must be able to access them as soon as he registers. Operators must therefore provide access while protecting their infrastructure from illegal access attempts by a person who has not subscribed to those services. Usually, the first step is to authenticate subscribers by checking the validity of proof of their identity.

Likewise, user authentication from a smartphone is a common requirement for cloud-based web services, as more and more personalized and controlled access services are operating online. Being the explicit support of the mobility capability gives the smartphone the privilege of accompanying its owner everywhere. It is therefore inevitable that it will gradually become the indispensable companion for all daily tasks while inaugurating a real change of use by allowing a subscriber to access permanently and from anywhere to the majority of online cloud services. From the security point of view, the smartphone is also gradually moving from the position of secondary device to the one of primary device (formerly occupied by the traditional computer). Unfortunately, several services always use weak authentication solutions (traditional log-in/password) that are vulnerable to attacks. To ensure that subscribers are who they claim to be, modern systems that imposed a higher level of security must integrate robust identity protection based on strong authentication approach.

In this paper, we study and develop an authentication protocol that delivers a higher level of authentication assurance based on two-factor authentication [1] and OTP (One-Time Password) [2] [3]. The proposed solution involves the use of two handheld devices:

- A *mobile phone* to submit the first authentication factor and then directly access a secure web service,
- A *smart-Thing* [4] which provides another wireless link, such as one of the Low-Power Wide-Area Networks (LPWAN) family [5] and allows to exchange the second OTP-based authentication factor.

To enhance security, each wireless link is secured with an efficient end-to-end encryption algorithm. Therefore, making it more difficult for attackers to undertake traditional attacks such as phishing, eavesdropping, man-in-the-middle, hijacking, replay, online guessing, brute force, etc. To the best of our knowledge, there have been no other authentication scheme proposed in the literature that use a LPWAN network as an alternative channel to provide two-factor authentication from a mobile phone to cloud-based web services.

Authentication is the first step leading to access to controlled or AAA[1] networks [6] [7]. The possible solutions to articulate the proposed authentication approach with the authorization issues to access to controlled resources or services are investigated. We show how the integration of our approach to OpenId Connect [9] and OAuth 2.0 [10] frameworks can be done.

This work has received funding from the France Brittany region research and innovation program AAP PME 2016, under grant agreement number 16004254, V2OLTERES project.

[1]AAA : Authentication, Authorization and Accounting

A Proof of Concept has been conducted to experimentally evaluate the proposed method.

The paper begins with some examples of multi-factor authentication schemes and a brief introduction to the IoT environment. Next, we present the proposed strong authentication method and describe in detail the authentication protocol together with its security evaluation. The articulation of our authentication approach with authorization issues is also discussed. At the end, we describe the proof of concept used to test our scheme.

II. Related works

A. Multi-factor Authentication

Multi-factor authentication is a security mechanism that requires more than one category of credentials used for identity verification. These categories of credentials are called *authentication factors* and the most commonly used are :

- A *Knowledge factor (Kf):* referred to as *something you know*, which is information known by users that they submit for authentication process, e.g. password, personal identification number (PIN), etc.
- A *Possession factor (Pf):* referred to as *something you have*, which is something that users must have in their possession for authentication, e.g. hardware token, ID card, smart-phone or smart-Things etc.
- An *Inheritance factor (If):* referred to as *something you are*, which are biometric characteristics of users that are compared during authentication process, e.g. fingerprint, facial, iris or voice recognition, DNA, etc.

The multi-factor credentials are managed by one or more devices and in the latter case the one from which the authentication involving the first factor is called the *primary device*. Several multi-factor authentication systems have been proposed for a wide variety of purposes. We focus on some of the most recently proposed mobile-based authentication methods. TDAS [11] is a touch dynamics based multi-factor authentication system for mobile devices. The proposed approach aims to study the feasibility and benefits of adopting an authentication method based on touch dynamics mechanism (*If*) by integrating it with the PIN-based authentication method (*Kf*). The authors presented how the data set may be used to strengthen the protection of resources that are accessible on mobile devices. Another approach is by Crossman and Liu [12], who propose a two-factor authentication based on NFC[2] smart-phone devices. Firstly, the user is asked to enter a password (*Kf*) which unlocks the protection of the key on its mobile phone acting as primary device. Then, the key (*Pf*) located in a passive object acting as secondary device, is transferred through NFC to complete the authentication process. In this system, the two factors are managed by the same mobile phone. This assumes a central point of vulnerability that can potentially be used by an attacker. Indeed, if the mobile phone is compromised, the two authentication factors will be easily available.

[2]NFC : Near Field Communication

Barkadehi et al. [13] proposed another two-factor authentication system by using the mobile phone as a mirror. In their approach, a web application launched from a desktop uses a log-in with a password as an authentication factor (*Kf*) in the first step and then a white box will be shown to the users. They can not see at first the mouse cursor in the box but must move their mouse in the box in order to click the right second password. By doing this, they receive a notification on their mobile phone (*Pf*) to open a mirror application. Then, they must accept the received request to continue the authentication process, and they will see their web-based cursor on a shuffled keyboard in their mobile application. To conclude the authentication process, the users need to select their second password. After a valid authentication process, they will have access to a web service through their desktop.

We are convinced that the use of two factors in the authentication process strengthens it in an interesting way. We argue, however, that its interest in reducing the attack surface of the authentication process can be advantageously improved by combining it with the use of multiple devices operating different protocol stacks. Our approach illustrates this point by using a smart-phone as primary device and a smart-Thing as secondary one.

B. IoT environment

With the rapid development of Internet services and wireless technologies, smart devices are well integrated into our daily life to provide customized IoT (Internet of things) to individuals. IoT interoperability appears as one of the greatest challenges for the IoT industrial environment [14] and limits the technology industry to achieve even larger IoT deployments. By achieving efficient interoperability between applications, sensors and networks, IoT is paving the way for the development of new and various services. Currently, smart-Things are able to interconnect, store data, send and receive commands to perform tasks requested by users. Heterogeneous system architectures are formed in which various types of devices and relevant communication techniques are deployed. Therefore, conventional security mechanisms [15] must be refined to fit the requirements from IoT environment.

An IoT device is integrated into our OTP-based two-factor authentication scheme. This smart-Thing must provide a network of LPWAN family such as LoRa or SigFox or one of the latest 3GPP technologies dedicated to the IoT which are LTE cat-M and NB-IoT technologies. The IoT device must be an active object. It must have its own power supply and be able to perform certain cryptographic operations.

III. Strong Mobile authentication Scheme for Web/Mobile Applications

Strong authentication is a technique relying on more than one authentication factor. By combining *something you know* and *something you have*, an adversary needs to physically steal your hardware token and also learn your password. This two-factor authentication scheme provides improved protection. In addition, communication channels' and devices' diversity is

another way to further improve the security of an authentication system. By this way, attacks such as man in the middle or eavesdropping are much more difficult to carry out since the adversary must simultaneously control both channels and devices. We introduce below a strong authentication method on mobile phone that uses in addition an IoT device and exploits these two improvements.

To authenticate a user through his smartphone and using an IoT device as a security token, certain components must be in place. Fig. 1 shows the basic architecture and the main components used to design our strong authentication scheme.

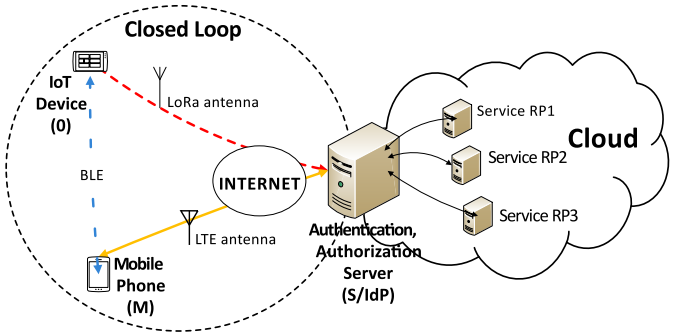

Fig. 1. Architecture of mobile authentication using an IoT device

The user must have a smartphone (M) connected to the Internet via the 3GPP LTE technology or WIFI network and also be in possession of an IoT device (O). Both devices must be equipped with a BLE[3] or NFC connection. With his mobile phone, the user can access to web services located in the cloud, subject to prior authentication. These services hosted by servers disseminated in the cloud are connected to S which is the front-end that will manage the authentication process. S is also connected to a LPWAN network (LoRa, SigFox, LTE Cat-M or NB-IoT) through Internet allowing the link with the IoT device O.

We propose a strong authentication scheme involving two distinct end user devices which communicate over different networks. One important point concerns the fact that both devices are controlled by the same user. This is ensured by a *closed loop* going through all the components involved in our architecture, illustrated in Fig. 1. The loop starts in the mobile phone requesting a service, goes through the network with S and then via the IoT device and back to the mobile phone. This closed loop can be realized in several ways. Below, we describe how to use it in the proposed strong authentication scheme.

IV. Authentication protocol in detail

We first summarize in the following table, all the notations used in the descriptions below.

The protocol rests on several security services. We first describe the main steps of the protocol while abstracting security issues. We then provide a detailed account of the required underlying security services.

[3]BLE : Bluetooth Low Energy

TABLE I
Notations

Notations	Description
u_{id}	User identifier
pwd	User password
M	Mobile phone
O	IoT device
S	Authentication server
id_m	Mobile phone identifier
id_s	Server identifier
t_1	Authentication token
t_2	Access token
sec_{OTP}	OTP secret
cod_{OTP}	OTP code
K_*^{priv}	Private key
K_*^{pub}	Public key
h_1, h_2	Hash functions
$E_{sec}(*)$	Encrypt function with the secret sec
$Sg_{see}(*)$	Signature encryption with the seed see
$XSg_{see}(*)$	Signature verification with the seed see

A. Protocol description

The proposed method rests on three distinct channels:

- a *primary* channel between the mobile phone M and the authentication server S
- a *secondary* channel between S and the IOT device O
- a *tertiary* channel between O and M

In the developed prototype we instantiate these three categories respectively with LTE, LoRa and BLE. There are thus plenty of other choices which must however take into account the technical constraints imposed by the devices. These three channels require at least confidentiality and integrity for the exchanged messages. To avoid depending on the security specificities of the protocol used to instantiate a channel, we use a security layer detailed in section IV-B to independently guarantee the requirements necessary for each communication channel.

We now focus on the flow of information required for an authentication as outlined in Fig. 2, the steps of which are detailed below:

1) By using his mobile phone M, the user submits a username (u_{id}) and password (pwd) to the authentication server S. These credentials (F1) are transmitted over an LTE link. This is the initial step of the authentication protocol during which the first *Kf* factor is used.
$$M \rightarrow S : [u_{id}, pwd]^{LTE}$$

2) S verifies the received data. If the data match with those stored in the database (DB), S replies with an authentication token (t_1) and an OTP secret (sec_{OTP}). t_1 is represented as a JWT (JSON Web Token) containing at least the u_{id} and works as a session identifier. sec_{OTP} is a random number generated at each authentication request. It is used as a seed in step 4 below.
$$S : u_{id}; pwd =? DB\{u_{id}; pwd\}$$
$$S \rightarrow M : [t_1, sec_{OTP}]^{LTE}$$

3) M verifies that S generated the OTP secret sec_{OTP}. Both received data are sent to O through a BLE link.

$$M : Generator(sec_{OTP}) =? \ S$$
$$M \rightarrow O : [t_1, sec_{OTP}]^{BLE}$$

4) O generates an OTP code (cod_{OTP}) by applying a function ($func$) involving sec_{OTP}. cod_{OTP} has a predefined validity period and is associated with t_1 to build a new credential (F2) needed to the second step of the authentication protocol. This second *Pf* factor is sent to S via a LoRa network.

$$O : cod_{OTP} = func_{OTP}(sec_{OTP})$$
$$O \rightarrow S : [t_1, cod_{OTP}]^{LoRa}$$

5) S computes its own version of the OTP code ($Xcod_{OTP}$) and compares it to the one contained in the credential (F2). If the data match, S validates the authentication. In anticipation of access authorization aspects analysed in section VI, S replies with an access token (t_2). We will explain in the section VI how t_2 can be used to access to resources or services.

$$S : cod_{OTP} =? \ Xcod_{OTP}$$
$$S \rightarrow O : [t_2]^{LoRa}$$

6) O forwards the access token to M via the BLE link.

$$O \rightarrow M : [t_2]^{BLE}$$

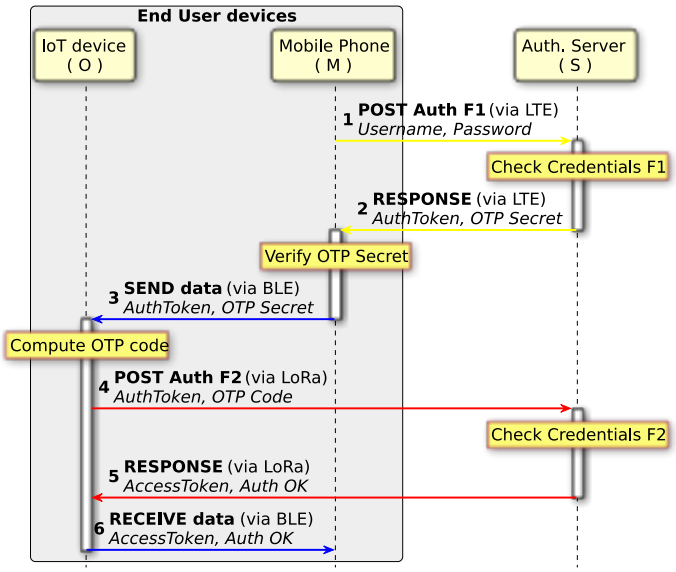

Fig. 2. Data flow of Authentication Procedure

B. Security enforcement of the protocol

Security brings a panoply of challenges in the authentication protocol design. It is therefore very difficult to build a secure authentication protocol [16]. Formal proofs could provide guarantees on the correctness of the protocol. However failing to provide a formal proof which at the time of writing this paper is still in progress, we introduce below the sound principles followed in the design of this protocol.

Firstly, OTP is introduced in our proposal as the second authentication factor. The OTP codes are generated by the IoT device (*Pf*) and can therefore appear totally random. As the name suggests, each OTP code is only used once. By changing each time an OTP code is needed, OTP solution introduces liveness. This is an important issue in security.

Secondly, we took advantage of JWT [17] which are tokens using a container to transport data between interested parties in JSON. These tokens are base64-encoded. We use them to authenticate all exchanged requests in a RESTFUL approach. They are defined with an expiration time that avoids forever valid tokens. The tokens are also protected against replay attacks by applying timestamps.

Thirdly, the three communication channels are secured with encryption algorithms.

1) *LTE Channel: Mobile Phone ↔ Authentication Server*

- *Key Agreement*: Before any data transmission, ECDH(E) (Elliptic Curve DiffieHellman, where final "E" stands for "ephemeral") and ECDSA (Elliptic Curve Digital Signature Algorithm) algorithms are used to create a secure channel between each mobile phone and the authentication server. These algorithms allow to define a trusted session key (ms). ECDH [18] is a well-known key agreement protocol used to define a shared secret over an insecure channel. Each involved party must have an elliptic curve public-private key pair. ECDH(E) is a variant of ECDH which provides temporary key pair instead of trusted static key pairs (via a certificate). ECDSA [19] is a variant of Digital Signature Algorithm (DSA) which uses the elliptic curve cryptography. This algorithm is used by a signatory to affix a *digital signature* on data and by a checker to prove the validity of the signature. Each party involved has a public-private key pair. The private key operates in the signature generation process and the public key is used for the signature verification. ECDSA provides message authentication, integrity and non-repudiation.

a) M generates its keys pair (K_m^{priv}, K_m^{pub}) and sends its public key and its signature to S.

$$M : K_m^{priv}, K_m^{pub}$$
$$M : Sg_{K_m^{pub}}(h_1(id_m))$$
$$M \rightarrow S : [K_m^{pub}, Sg_{K_m^{pub}}(h_1(id_m))]^{LTE}$$

b) S verifies the received signature to ensure that the data come from M (we make the assumption that S has previously registered the respective identities of all the legitimate devices which can be used for a user authentication). If the data come from a known device, S generates its pair of keys (K_s^{priv}, K_s^{pub}). Then, it computes the session key (ms) using its private key and the

public key of M. Finally, S replies to M with its public key and its signature.

$S : XSg_{K_m^{pub}}(h_1(id_m)) =?Sg_{K_m^{pub}}(h_1(id_m))$
$S : K_s^{priv}, K_s^{pub}$
$S : ms = Compute\ [K_s^{priv}, K_m^{pub}]$
$S \rightarrow M : [K_s^{pub}, Sg_{K_s^{pub}}(h_1(id_s))]^{LTE}$

c) M verifies the received signature to ensure that the data come from S (we make the assumption that M has previously registered the identity of the authentication server S). Then M computes the session key (ms) using its private key and the public key of S.

$M : XSg_{K_s^{pub}}(h_1(id_s)) =?Sg_{K_s^{pub}}(h_1(id_s))$
$M : ms = Compute\ [K_m^{priv}, K_s^{pub}]$

- *Symmetric Encryption*: After this negotiation, the channel is secured with the AES cipher [20] (standard recommended by NIST) using the session key (ms) to provide end-to-end data encryption and integrity ($E_{ms}(*)$). Furthermore, OTP secret is signed ($Sg_{id_m}(*)$) by the authentication server using the mobile identifier as a seed. Thus, the secure data are completely binded to a specific mobile phone which is by the way ensured that sensitive information received on this channel come from the authentication server. The complete respective expressions of step 1 and 2 introduced in section IV-A are thus:

$M \rightarrow S : [E_{ms}(u_{id}), E_{ms}(h_2(pwd))]^{LTE}$
$S \rightarrow M : [E_{ms}(t_1), Sg_{id_M}(E_{ms}(sec_{OTP}))]^{LTE}$

2) *BLE Channel: Mobile Phone ↔ IoT device*
This channel is secured with the AES symmetric encryption scheme. More precisely, the AES cipher with 128-bit pre-shared key length (mo) is implemented to provide end-to-end data encryption. The data being transferred over this channel are tokens and OTP secrets which are already secure. Step 3 and step 6 of IV-A are refined below:

$M \rightarrow O : [E_{mo}(E_{ms}(t_1)), E_{mo}(sec_{OTP})]^{BLE}$
$O \rightarrow M : [E_{mo}(E_{ms}(t_2))]^{BLE}$

3) *LoRa Channel: IoT device ↔ Authentication Server*
This link is secured with the AES cipher with 128-bit pre-shared key length (so) to provide end-to-end data encryption. Step 4 and step 5 of IV-A are refined below:

$O \rightarrow S : [E_{ms}(t_1), E_{so}(cod_{OTP})]^{LoRa}$
$S \rightarrow O : [E_{so}(E_{ms}(t_2))]^{LoRa}$

V. Security Evaluation

This section presents a security assessment of the proposed strong authentication method. This assessment is divided into four different scenarios. Each scenario examines potential threats and then discusses possible countermeasures.

A. Scenario I : End user devices

Threats:

- *Physical attack* : an adversary might try to steal the user devices to impersonate a legitimate user.
- *Software attack* : a malicious application can gain access to sensitive data stored on mobile phone or IoT device.
- *Spoofing* : an attacker might try to tamper with the IoT device to learn the shared key.

Countermeasures:

- Trusted Execution Environment (TEE) [21] is created to allow mobile phones and connected objects to achieve their requirements for speed and security. It is an isolated area inside the main processor that verifies the confidentiality and integrity of the code and information loaded in the TEE.
- Security Encrypted (SE) [22] is a tamper-resistant chip inside the modern mobile devices. It is completely separated from the main processor with its own CPU, RAM and persistent memory. It provides a very high level of security by saving sensitive information such as secret keys and nothing stored inside can be uploaded by an attacker.
- End user devices can easily be revoked by the authentication server in case of theft thanks to their unique identifier used during the authentication process.

B. Scenario II : Channel Mobile Phone – Authentication Server

Threats:

- *Phishing* : an adversary might try to impersonate a user by guessing his password.
- *Eavesdropping* : an attacker might try to eavesdrop on the link to learn the OTP secret.
- *Man-in-the-middle (MITM)* : an adversary might try to position himself between the user mobile phone and the authentication server in order to read or/and alter all information.

Countermeasures:

- The proposed scheme is protected with the user chosen password. This results in a two-factor authentication preventing misuse of stolen password even more a hash version of password is transmitted over the network and not the plain text password.
- The well-known cryptographic algorithm AES is used to avoid eavesdropping and MITM attacks. In additional, the OTP secret is signed by the authentication server. Thus, the secure data are intended for a given mobile phone that ensures that received information come from the real authentication server.

C. Scenario III : Channel Mobile Phone – IoT device

Threats:

- *Online guessing or dictionary attack*: an attacker might try to listen on the BLE channel to learn sensitive information.

Countermeasures:

- The BLE channel is secured with a symmetric encryption algorithm (AES) to perform end-to-end data encryption and integrity. In addition, the data being transferred over this BLE channel, are a token and a secret which are already secure by introducing liveness.

D. Scenario IV : Channel IoT device – Authentication Server

Threats:

- *Hijacking* : an adversary might try to hijacking the authentication token to get unauthorized access.
- *Replay attack* : an adversary might try to repeat or delay a valid sensitive data.

Countermeasures:

- The link between the IoT device and the authentication server is secured by a symmetric encryption algorithm (AES) to provide end-to-end data encryption and integrity over the link.
- The tokens are protected against replay attacks by using timestamps.
- OTP codes are not possible to guess since they appear as random numbers. Each code is only used once.

VI. Access to services

AAA networks [6] [7] provide resources that can be accessed by users following different authorization policies. Authentication is the first step to manage access to previously registered users. In this part, we investigate a possible solution to articulate our new authentication scheme with the authorization control to access to resources or services. In a previous work, we studied the particular case of services located in a trusted area [8] which is very common in practice. We investigate here the general case ("untrusted" area). We show how our strong authentication approach can be integrated into the OpenId Connect [9] and OAuth 2.0 [10] frameworks.

A globally "untrusted" area may consist of several trusted islands representing distinct administrative domains operated by different actors. Depending on the trust relationship between these different actors, it often happens that a successful authentication to access a resource in one domain is reused to access another resource of a second one. Even if we do not deal with the issues of the federation of domains and the Single Sign One (SSO), we choose the approach based on OpenId Connect [9] and OAuth 2.0 [10] which may easily be enriched to support the SSO use case.

A. Authentication and Authorization in the Cloud

Access delegation has generated a lot of interest and work to finally lead through OAuth 2.0 [10] to a relatively stable and proven solution [23] that has become essential for the development of the Cloud. By underestimating the importance of authentication, the initial proposal of OAuth created one of its own major weakness. The introduction of OpenID Connect [9] as an identity layer on top of the OAuth 2.0 protocol solved the issue. OpenId Connect with OAuth 2.0 can be used to manage the access to an application server called RP (*Relying Party*) providing resources or services. This access requires a preliminary authentication step summarized by the generation by an IdP (*Identity Provider*) of an *IdToken* for the authentication and an *AccessToken* for the authorization. The approach is sufficiently flexible and rich to be adapted to multiple configurations. The standard lists four main configurations [10], of which we indicate below the mode *Code Authorization* considered the most secure [23].

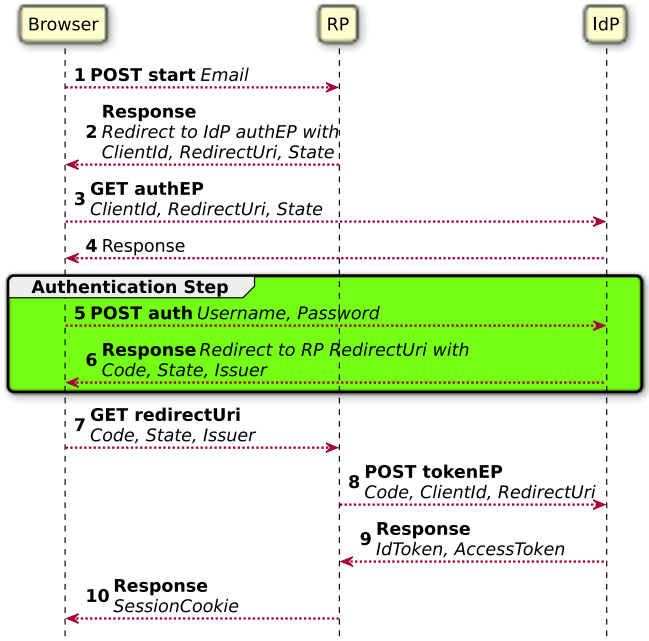

Fig. 3. OIDC call flow - Code Authorization [23]

B. Our solution

OpenId Connect associated to OAuth 2.0 provides a flexible framework the modularity of which can be exploited to integrate the proposed strong authentication method. This one is used as a preliminary step before controlling the access to an application server represented by RP in Fig. 4. We associate the IdP with our previous authentication server S to form a complete Authentication and Authorization Server named S/IdP.

The main idea consists in substituting to the basic authentication step (cf. "green block" of Fig. 3), our two factors method based on channel and device diversity. Due to constraints related to IoT environment, the integration of our two-factor authentication scheme with OpenId Connect/OAuth 2.0 requires several adaptations. The messages from S/IdP to O may contain a significant amount of information (*Authtoken2, Code, State and Issuer*) the processing of which will require high power consumption for the IoT device. Moreover, due to efficiency reasons applications on mobile phone are rarely browser-based. They are usually implemented as specific binaries one can download from some applications' store. For our purpose, we developed a dedicated one noted apk in Fig. 4. It

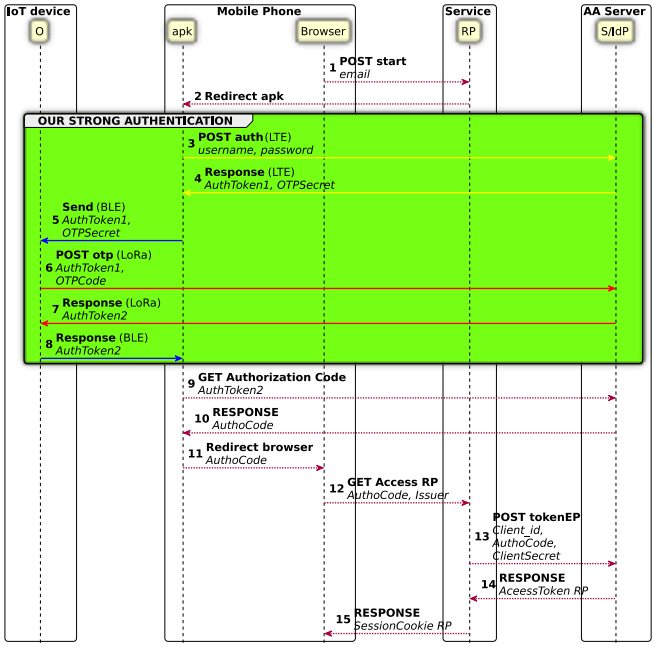

Fig. 4. Strong authentication call flow with OIDC from mobile phone application (apk) and browser

fits well with the fact that exchanges between user devices are carried out via BLE or NFC. At our current knowledge the development of the browser's APIs managing these network resources are still in progress. On the other hand the handle of BLE or NFC through the apk is straightforward.

Using the apk makes useless messages 3 and 4 of Fig. 3 which consisted in triggering the authentication process. The apk explicitly requests with message 9 of Fig. 4, the Authorization Code from S/IdP. $AuthToken2$ is required to allow S/IdP to establish the link with the previous authentication. Finally messages 12 to 15 of Fig. 4 are equivalent to messages 7 to 10 of Fig. 3.

VII. Proof of Concept

A platform has been developed to experimentally evaluate the proposed method. It rests on two open source certified OpenID Connect implementations: *node openid-provider* [24] and *node openid-client* [25] representing the IdP and the RP respectively. We experiment our approach using the two well-known services WebRTC and WebSocket represented by $RP1$ and $RP2$ in Fig. 5. This one shows the proposed architecture where two main parts can be distinguished. The first represents the "untrusted" area (i.e. the cloud) in which different servers have been implemented using mainly the open-source, cross-platform JavaScript run-time environment, called *Node JS* [26] [27]. The second part puts together the end user devices.

A. Cloud components

The cloud area contains a powerful physical infrastructure that uses virtualization software to provide three virtual servers. In addition, two application servers from different

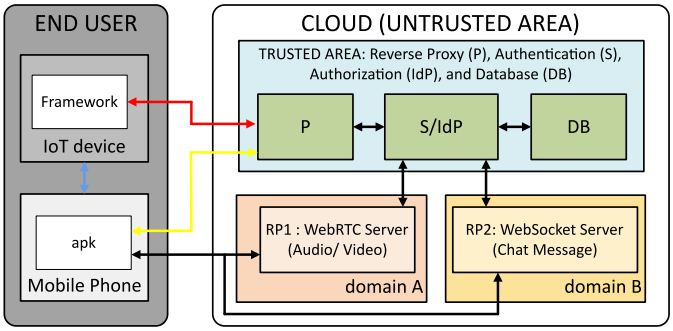

Fig. 5. Proof-of-concept Architecture

domains provide resources or services to authenticated and authorized users.

- *Reverse Proxy Server (P)*: This component is the entry point to the trusted area. It channels all requests from the Internet and then forwards them to authentication, authorization or application servers.
- *Data Base Server (DB)*: This component collects all users information (name, password, address, email, etc) in an organized way so that data are easy to access, manage and update. For this component, we used MongoDB [28] [29] which is a well-known cross-platform document-oriented database.
- *Authentication and Authorization Servers (S/IdP)*: Authentication server is one of the main component of the platform. It processes and responds to requests in order to determine if previously registered users are really who they claim to be. Authorization process occurs after a user is successfully authenticated. Its purpose is to determine precisely if an authenticated user has access to the requested resources or services by evaluating the authorization policy associated to his profile.
- *Applications Servers*: Different application servers can be deployed in the cloud. This approach requires a minimum security effort since web servers are distributed in an "unsecured" area and can be accessed by everyone. Therefore *HTTPS*) is used for any data exchange with the deployed servers. For the sake of experiment, we have implemented two application servers located in two different domains with digital certificates for encrypting the communication protocol (*HTTPS*):
  - The *WebRTC server (RP1)* [30] which provides real-time peer-to-peer audio, video or data (i.e. multimedia) communications by leveraging a set of plugin-free APIs that can be operated in both mobile phone and desktop browsers. Previously, external plugins were needed to obtain similar features to those offered by WebRTC.
  - The *WebSocket server (RP2)* [31] for a Chat Message Service. Websocket is an application layer protocol that allows full duplex communication over a single TCP connection. That protocol is used for a bi-directional real-time message flow in our application.

B. End User devices

There are two types of device managed by the end user. A smartphone or a tablet running under Android Oreo that represent the primary user device and an IoT device as the secondary one.

1) User primary device: The mobile phone (Samsung Galaxy S8) or the tablet (Samsung Galaxy Tab S3) is used to initiate an authentication process, and to perform either a WebRTC audio video call or chat in an instant messaging service. For this purpose, we used Ionic [32], which is an open-source hybrid mobile application development framework, to develop a mobile application (apk). Ionic uses Cordova plugins [33] to access to the mobile phone or tablet features such as GPS, bluetooth, camera, microphone, etc. Furthermore, Ionic can build apk for Android, iOS, or Windows.

2) User secondary device: As a secondary user device, we use the Next Generation IoT Platform (Pycom) [34] which provides powerful and affordable MicroPython enabled, multi-network micro-controller development boards. According to our needs in terms of communication capability, we have chosen the latest Pycom product on the current market. This is the FiPy board [35] which allows the following five communication stacks: BLE, LoRa, SigFox, LTE Cat-M and NB-IoT.

In the IoT environment, energy issues are critical. The hardware selected for our experiments makes it possible to explore the problem in an extensive way because of the numerous protocol stacks proposed, but it is however expensive in terms of energy consumption. We introduce a switch that obeys the automaton below (Fig. 6) in order to avoid energy waste.

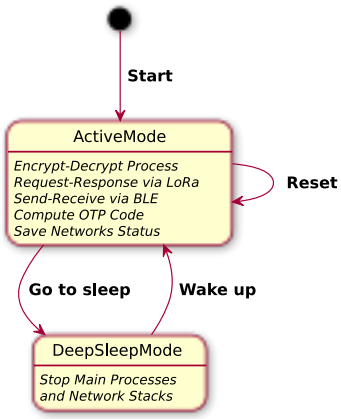

Fig. 6. Automaton to manage the energy consumption of the IoT device

The proposed solution to control the device energy consumption is quite simple. We have set two possible operating modes that form the behavior of the IoT device:

- An *Active Mode* in which all processes are initialized and ready. The IoT device waits for an authentication request via a BLE connection from the mobile phone, then computes the OTP code, encrypts the data and exchanges them with the authentication server via a LoRa network. At the end, the result of the authentication request is sent back to the mobile phone.
- A *Deep Sleep Mode*: By pressing a push button, the IoT device saves its status (networking) and stops all main processes and network stacks.

Fig. 7 shows the variation in energy consumption according to the IoT device's operating mode. In active mode the average consumption is approximately 165 mA and in deep sleep mode the average consumption drops to 0.33 mA.

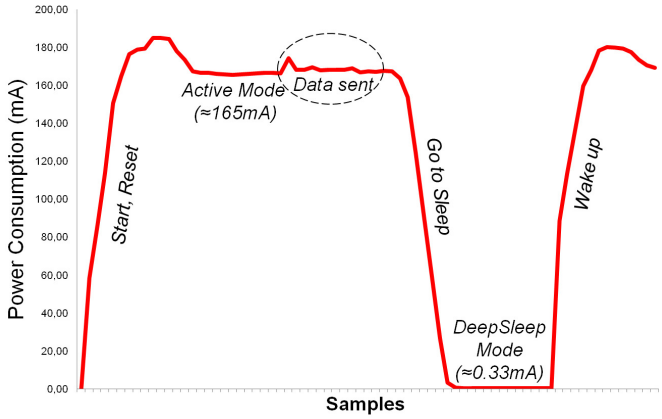

Fig. 7. IoT device Energy consumption

VIII. Future Works

The proposal implements a two-factor authentication using the diversity of channels and devices. End-to-end encryption of each involved channel enforces the overall security of the protocol. Its extension to more than two factors could be considered. For example, a method involving in additional contextual factors such as "time" or "location" would likely improve the robustness of the authentication scheme.

The proposed method involves a first traditional factor (login/password) enforced by a second strong factor based on OTP. It offers an interesting framework to study the substitution of the first factor by other user friendly and cheap authentication methods. This issue concerns the development of an authentication method which is simultaneously highly robust, ergonomic and user friendly.

A platform that implements this method has been developed, tested and evaluated under different attack scenarios. This first assessment of the robustness of the presented protocol will be completed by a comprehensive analysis involving formal proofs.

IX. Conclusion

This paper proposed an original and strong authentication method dedicated to mobile phones resting on IoT devices. The user owning an LPWAN-enabled hand-held device may directly access any cloud-based web service through his mobile phone. To the best of our knowledge, this is the first authentication scheme using an LPWAN network as an alternative channel to provide two-factor authentication from a mobile phone to cloud-based web services. Using an IoT device as an authentication token provides a very flexible and

secure solution. It appears as a very interesting alternative to all the existing authentication solutions that use the mobile phone as a security token. Smart things for authentication are very likely to become a popular choice for many users and companies in the near future.

The web services which can be accessed through the mobile phone are located in the Cloud. Our authentication method is integrated in a general framework like OpenId Connect articulating at the same time this work with authorization and federation approaches. The first results show how the integration of our approach can be done to the framework of OpenId Connect and OAuth 2.0.

X. Acknowledgment

The authors thank the company Acklio which graciously provided them with access to their LoRa network.

References

[1] D. P. Roberto, M. Gianluigi and S. M. Adriano, "A two-factor mobile authentication scheme for secure financial transactions," IEEE International Conference on Mobile Business (ICMB'05), pp. 28–34, 2005.

[2] M. H. Eldefrawy and K. Alghathbar and M. K. Khan, "OTP-Based Two-Factor Authentication Using Mobile Phones," 2011 Eighth International Conference on Information Technology: New Generations, pp. 327–331, Apr. 2011.

[3] F. Aloul, S. Zahidi and W. El-Hajj, "Two factor authentication using mobile phones," 2009 IEEE/ACS International Conference on Computer Systems and Applications, pp. 641-644, May. 2009.

[4] M. Kuniavsky, "Smart things: ubiquitous computing user experience design," Elsevier, 2010.

[5] L. Krupka, L. Vojtech and M. Neruda, "The issue of LPWAN technology coexistence in IoT environment," 17th International Conference on Mechatronics - Mechatronika (ME), pp. 1-8, Dec. 2016.

[6] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege and D. Spence, "AAA Authorization Framework," IETF RFC 2904, 2000.

[7] M. Nakhjiri and M. Nakhjiri, "AAA and Network Security for Mobile Access," Wiley,2005.

[8] S. Kambou and A. Bouabdallah, "A Strong Authentication Method for Web/Mobile Services", accepted in IEEE International Conference on Cyber Security and Cloud Computing (IEEE CSCloud 2019).

[9] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1,"https://openid.net/, 2014.

[10] D. Hardt, "The OAuth 2.0 Authorization Framework," IETF RFC 6749, 2012.

[11] P. Shen Teh, N. Zhang, A. Beng Jin Teoh, K. Chen, "TDAS: a touch dynamics based multi-factor authentication solution for mobile devices," International Journal of Pervasive Computing and Communications, Vol. 12 Issue: 1, pp.127–153, Feb. 2016.

[12] M. A. Crossman and H. Liu, "Two-factor authentication through near field communication," IEEE Symposium on Technologies for Homeland Security (HST), pp. 1–5, Mar. 2016.

[13] M.H. Barkadehi, M. Nilashi, O. Ibrahim, "A novel two-factor authentication system robust against shoulder surfing," J. Soft Comput. Decis. Support Syst., Vol.4 Issue: 1, pp. 19–25, Feb. 2017.

[14] K. Zhao and L. Ge, "A Survey on the Internet of Things Security," 2013 Ninth International Conference on Computational Intelligence and Security, pp. 663-667, Dec. 2013.

[15] Z.K. Zhang, M.C.Y. Cho, C.W. Wang, C.W. Hsu, C.K. Chen, and S. Shieh, "IoT security: ongoing challenges and research opportunities.," IEEE 7th international conference on service-oriented computing and applications, pp. 230-234, Nov. 2014.

[16] M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols," IEEE transactions on Software Engineering 22 (1), 6-15, 1996.

[17] M.B. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)", IETF RFC 7519, 2015.

[18] A. Maryam, S. Baharan, A. Difo, R. Amirhossein and O. Habeeb, "Diffie-Hellman and its application in security protocols," International Journal of Engineering Science and Innovative Technology (IJESIT), Vol.1, pp. 69–73, Nov. 2012.

[19] J. Don, M. Alfred and V. Scott, "The elliptic curve digital signature algorithm (ECDSA)," International journal of information security, Vol.1 Issue: 1, pp. 36–63, Jul. 2001.

[20] V. Saicheur and K. Piromsopa, "An implementation of AES-128 and AES-512 on Apple mobile processor," 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), pp. 389-392, Nov. 2017.

[21] S. Pinto, T. Gomes, J. Pereira, J. Cabral and A. Tavares, "IIoTEED: An Enhanced, Trusted Execution Environment for Industrial IoT Edge Devices," IEEE Internet Computing, Vol. 21 Issue: 1, pp.40–47, Jan. 2017.

[22] W. Anwar, D. Lindskog, P. Zavarsky, and R. Ruhl, "An alternate secure element access control for NFC enabled Android smartphones," International Journal for Information Security Research (IJISR), Vol. 4 Issue: 11, pp.391–399. Mar. 2014.

[23] D. Fett, R. Ksters and G. Schmitz, "The Web SSO Standard OpenID Connect: In-Depth Formal Security Analysis and Security Guidelines", IEEE 30th Computer Security Foundations Symposium, August 21-25, 2017, Santa Barbara, CA, USA, 2017.

[24] node openid-provider : https://www.npmjs.com/package/oidc-provider, (accessed March 31, 2019)

[25] node openid-client : https://www.npmjs.com/package/openid-client, (accessed March 31, 2019)

[26] W. Emily, "Nodejs In a Day", CreateSpace Independent Publishing Platform, 2016.

[27] NodeJS Web Site: https://nodejs.org/en/, (accessed March 31, 2019)

[28] B. Kyle, "MongoDB in action", Manning Publications Co., 2011.

[29] MongoDB Web Site: https://www.mongodb.com/, (accessed March 31, 2019)

[30] WebRTC Web Site : https://webrtc.org/, (accessed March 31, 2019)

[31] I. Fette and A. Melnikov , "The WebSocket Protocol," IETF RFC 6455, 2011.

[32] Ionic Web Site: https://ionicframework.com/, (accessed March 31, 2019)

[33] Cordova Plugin Web Site: https://cordova.apache.org/, (accessed March 31, 2019)

[34] Pycom Web Site: https://pycom.io/, (accessed March 31, 2019)

[35] FiPy:https://github.com/pycom/pycom-documentation/blob/master/product-info-datasheets/development/fipy.md, (accessed March 31, 2019)