# A Lightweight and Robust Dynamic Authentication System for Smarthome

Elisée Toé[1], Tiguiane Yélémou[1(✉)], Doliére Francis Somé[2], Hamadoun Tall[1], and Théodore Marie Yves Tapsoba[1]

[1] Nazi BONI University, Bobo-Dioulasso, Burkina Faso
tyelemou@gmail.com
[2] CISPA-Stanford Center for Cybersecurity, Saarbrücken, Germany
doliere.some@cispa.de

**Abstract.** The advent of smarthomes improves the comfort in the homes. In this work, we are interested in automatic opening of gate at the arrival of legitimate vehicles in order to avoid that the occupants have to wait at the door. Indeed, due to the rise of insecurity in our cities, cases of aggressions in front of the doors are regularly reported. Also, the noise pollution necessary to be noticed at the door disturb the neighborhood. The major challenge of the gate automation is security. Most of the proposed solutions do not sufficiently take into account the robustness of the vehicle authentication mechanisms. We propose a security enhancement for vehicle access control with a sensor node on board at a gate that also has a sensor node. Our mutual authentication protocol between the two sensor nodes takes into account the resource limitations of the used objects. It is based on dynamic one-time passwords. We exploit random number generation and processing functions, Elliptic Curve Diffie Hellman Ephemeral (ECDHE) key exchange principle and HMAC-SHA256 hash function. The lightweight system with only one message exchanged in total offers a very low reaction time for the gate. Its dynamic nature allows it to resist to cryptanalysis and spooking attacks.

**Keywords:** Internet of Things · Smarthome · Cybersecurity · dynamic authentication · one-time password · Elliptic Curve Diffie Hellman Ephemeral (ECDHE)

## 1 Introduction

More and more objects of everyday life are connected to the Internet or are accessible remotely in an Intranet. Many of these objects can thus be controlled automatically according to the physico-chemical conditions of their environment or the goodwill of their owners [1]. However, problems related to the security of these connected objects are considerably slowing down the evolution and deployment in the Internet of Things. The use of connected objects often meets the needs of users who are not sufficiently aware of security issues. The manufacturers of this type of object are often companies with little or no expertise in the


R. A. Saeed et al. (Eds.): AFRICOMM 2022, LNICST 499, pp. 50–63, 2023.
https://doi.org/10.1007/978-3-031-34896-9_4

field of security and focus on the functionalities and ease of use of objects to the detriment of security.

In smarthome, flaws in the authentication mechanisms of connected door locks, are at the origin of several cyber attacks because most of them are focused on an unsecured use in their classic form of these mechanisms. Moreover, objects cannot use classical security protocols (e.g. TLS) and most of the existing robust solutions do not allow to ensure good performances or they are not adapted to the capacities of objects which are limited in computing power. Beyond services and comfort offered by connected portals, mobile nodes such as cars connecting discontinuously to the system, a robust authentication mechanism is needed to govern the exchanges between the vehicle and the connected portal.

This work aims at implementing a security mechanism adapted to limited resources contexts that prevents any malicious or unauthorized object from accessing a home automation system, in particular the control of access to the gate by vehicles.

We propose a mutual authentication protocol suitable for resource-constrained objects between the vehicle and the portal. This protocol faces the above mentioned problems and challenges and thereby:

- it allows a light mechanism for a good implementation by our resource-constrained objects;
- it allows a quick authentication through a single exchange of information in order not to expose to assault the drivers of the vehicles if they have to wait for the opening of the gate;
- it provides dynamic authentication with one-time passwords to resist brute force, replay, cryptanalysis and man-in-the-middle attacks.

The rest of the paper is organized as follows. Section 2 is devoted to related work on authentication and access control mechanisms applied in IoT. We present our contribution in Sect. 3. The performance evaluation of our solution is discussed in Sect. 4. We conclude in Sect. 5.

## 2 Previous Work

Security is one of the major issues in increasing use of connected objects. The designers of these objects are often much more focused on innovation in functionalities and miniaturization. Also, due to the low capacities (memory, CPU, storage, embedded energy) the traditional robust security bricks cannot be deployed in these objects. However, sensitive information is often transmitted or stored in these IoT devices. They can be exposed to illicit exploitation. Resource-efficient (CPU, memory, energy) solutions must be developed to face multiple threats on these objects. In the last few years, many research works have focused on these security issues in the IoT. But many of them struggle to be efficient in contexts where the constraints of action delay or resources are high.

Communications with wireless connected objects are very exposed to sniffers. To make the authentication process of these objects efficient, one-time passwords are recommended.

HMAC-based one-time password (HOTP) [2] is one of the standards developed by the Initiative for Open AuTHentication (OATH) which is an international collaborative group aiming to promote strong authentication in open-source. The HOTP algorithm is based on a counter and a static symmetric key known only to the token and the validation service. Since the output of the HMAC-SHA [3] calculation is 160 bits. This value must be truncated to something that can be easily entered by a user (see Eq. 1).

$$HOTP(K, C) = Tronquer(HMAC - SHA(K, C)) \tag{1}$$

- Tronquer represents the function that converts an HMAC-SHA-1 value to a HOTP value.
- The key (K), the counter (C) and the data values are hashed by the first high byte.

Another technique for generating one-time passwords is Time-Based One-Time Password (TOTP) [4] appeared in 2011. It is also part of the standards developed by OATH. It is based on HOTP, with the only difference that the change factor is time and not a counter. It is based on "POSIX" time. The initial sharing of the "secret" between the two entities remains the same. However, the generation of the OTP will be done with the couple "secret" and time (or more precisely a "timestamp") over a defined period (usually 30 to 60 s). This means that TOTP uses time incrementally and each OTP is valid for the duration of the time interval. Basically, *TOTP=HOTP(K,T)*, where T is an integer and represents the number of time steps between the initial counter time T0 and the current time. These two mechanisms are often confronted with synchronization problems and reside on a shared secret. Their change factors can be easily broken through by cryptanalysis.

Mohamed Tahar HAMMI and al. [5,6] implements the OTP principle in a customized key management mechanism. In this solution, a mutual authentication of nodes enforced by key management and AES algorithm to guarantee the integrity and confidentiality of exchanges is proposed. First, the device makes an association request, receives an authentication request, generates and sends otp1, and finally, authenticated by the Personal Area Network Coordinator (CPAN) if it is legitimate. Then, a ku key (derived from the identifier) is generated using the Pseudo Random Function (PRF) defined in RFC 5246 [7] which allows to have very robust keys. The generation of the keys is established only after the authentication operation has been successfully completed in order to avoid unnecessary calculations. The next step is a secure kb broadcast key exchange mechanism called hidden key broadcast and a calculation of a second OTP (otp2) for CPAN authentication. The hidden key broadcast mechanism is realized in 2 phases:

1. A value named signature is generated by calculating a *HMAC (ku, otp1)*, then,
2. We XOR the result with *kb*.

This solution brings security bricks into wireless sensor networks. The work of Esfahani and al. [8,9] are almost in the same logic as Hammi. However, the key management technique used in this authentication algorithm requires quite a lot of computing power, which is a limitation for our objects, especially since it completes the authentication process only after several exchanges on the channel.

Research has been conducted to find cryptographic systems suitable for resource-constrained objects known as lightweight cryptography. These mechanisms are based on elliptic curves.

Badis Hammi and others [10] have proposed an Elliptic Curve Diffie Hellman Ephemeral (ECDHE) key exchange algorithm based on elliptic curves. It is an algorithm used for key exchange that allows two entities to establish a shared secret. It is actually an adaptation of the Diffie-Hellman (DH) key exchange protocol that uses elliptic curve cryptography to minimize key length and improve performance. Elliptic curves are widely used in various key exchange methods, including DH key agreement. Elliptic Curve Cryptography (ECC) [11] provides a similar level of security to RSA, but with smaller key sizes that allow for fast computations and lower power consumption for IoT devices. Previous authentication methods (such as RSA) use a certificate to authenticate devices. Although using a certificate provides a high level of security, signing and verifying the certificate uses a high computational process that increases CPU and power consumption. Therefore, instead of using certificate-based authentication algorithms, the proposed mechanism to use the Pre Shared Key (PSK) [12] authentication algorithm. The pre-shared key is used to authenticate other parties as well as the ECDHE key exchange algorithm. In this process, to use ECC, each party must agree on the domain parameters *(p, a, b, G, n, h)* that define the elliptic curve. In addition, the client and server must have a key pair for elliptic curve cryptography, consisting of a private key d (a random integer) and a public key *Q (Q=dG)*. Therefore, the private key dc and the public key *Qc (Qc=dcG )* are for the client and the keys ds and *Qs=dsG* are for the server. Then, the client and the server exchange their public keys. They compute the secret key S using their own private key and the public key of the other party. The client computes S=dcQs and the server computes *S=dsQs*. The shared secret key (S) is the same for both parties since $S = dcQs = dcdsG = dsdcG = dsQs$. The ECDHE algorithm does not provide authentication per se, since the key is different each time and neither party can be sure that the key comes from the intended party. Therefore, the authentication algorithm (PSK) is used with the ECDHE algorithm to authenticate both parties.

The PSK authentication algorithm applies a string of characters (64 hexadecimal digits) that is used as an authentication key (shared secret) and shared between the client and server in advance for text encryption. When the secret key is shared between them, they authenticate each other through the four-step procedure of the shared key authentication algorithm. The advantage of the PSK algorithm is that it avoids the heavy computation of the public key for authentication. The only problem is that if the attacker found the shared secret key, the previous and future sessions will be compromised. When the proposed

mechanism uses PSK with the ECDHE key exchange algorithm, it provides the Perfect Forward Secrecy (PFS) feature that protects past sessions from future compromise by providing a separate key for each session. Even if the attacker somehow accesses this shared secret, he will only compromise that specific session. Previous or future sessions would not be compromised.

For mobile nodes such as vehicles that connect discontinuously to the core system, basic authentication methods are easily defeated, especially by cryptanalysis and spoofing attacks. Also too robust methods [13,14] do not allow to have the desired speed of action. As a reminder, the desired operational objective is for the door to open as quickly as possible to prevent the vehicle from stopping at the door. This can expose the occupants to aggression.

## 3 Contributions

Authentication in the Internet of Things environment is very important to ensure that each legitimate node communicates with the target devices. Despite the resource constraints (computing power, storage space and bandwidth), the authentication mechanism for these objects must be strong. The desired functional requirements are:

- Efficiency: in the context of smart home aiming on the one hand at improving comfort, it is necessary that the authentication takes place as quickly as possible to avoid waiting too long in front of the gate and then being object of aggression while waiting. On the other hand, to ensure the efficiency of the communications during the mutual authentication phase, the number of messages and the size of the data exchanged must be minimized.
- Lightweight: in order to respect the resource constraints of smart objects, the authentication protocol must be lightweight. The primitives and functions used on smart objects must have a lower computational cost than traditional cryptographic algorithms (e.g. RSA, AES). Also the data storage on smart objects must be minimized.
- A distributed solution: for machine-to-machine (M2M) communication, intelligent objects must be able to communicate directly without requiring the intervention of a central server during the authentication phase.

Our approach is based on the combination of the One Time Password (OTP) mechanism and the password integrity check by exploiting the principle of the Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol for the generation of session keys for the hash function (HMAC-SHA256) in order to dynamize the hash computation. We exploit the Perfect Forward Secrecy feature of ECDH key management to exchange these session keys and session information for future connections.

By definition, an OTP is a password that is valid only once. Therefore, it is very robust against replay and cryptanalysis attacks. We opte for the asynchronous mode which is based on the challenge/response method because it does not require any prior agreement between the different nodes like the synchronous mode. In order to propose a robust and lightweight authentication, our approach has the following characteristics:

- a dynamic and mutual authentication;
- multi-factor authentication: biometrics-OTP-session key-location;
- authentication in a single information exchange;
- a mechanism for authenticating exchanges.

Some tricks have been observed to reduce the complexity of this mechanism. As far as we manage to integrate the identification and authentication information by sending a single message, the privacy management is not a concern for the operation of the portal because its implementation requires computing power and other additional steps. The only authentication step after an approved verification is enough to operate the portal without sending any other information and the dynamic aspect of the information used for authentication is a great asset. Figure 1 shows our solution for strengthening the portal access control.

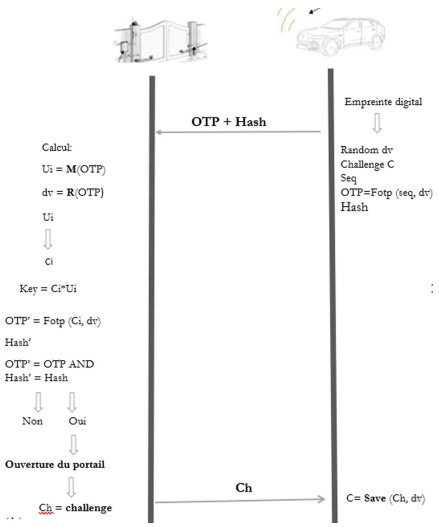


**Fig. 1.** Authentication protocol

- The Fotp function is used to generate the OTP. It takes as input a random number *dv*, the authentication session number seq and a challenge *C*. This number is incremented at each valid session.

- Ui represents the unique identifier of the node and is a number between 41 and 49. The *Tr(Ui)* function generates a four-digit random number using the median square method so that the two digits representing Ui are in the middle. This generated number will represent the first 4 characters of the OTP.
- When the gate side node receives the OTP, the Ui is retrieved by the*M(OTP)* function which extracts the first four characters of the OTP making up the number generated to mask the *Ui*. The retrieval of the *Ui* will allow the portal node to identify the node and to take on the challenge of calculating the OTP for the car and to verify its compliance.
- C represents the pre-exchanged challenge between the two nodes during the last session. This challenge is dynamic and unique for a session. It corresponds to the*Ui* for the first communication session between the two nodes. The challenge exchange between the two nodes is based on the ECDH key exchange principle via the *challenge(dv)* function which allows us not to transmit the real challenge on the channel. This function is executed after a valid authentication to allow an update of a new shared secret for the next communication. This challenge is used to create the session key for the next authentication request and at the same time boost the key of the hash function. On the car side, another function *Save(ch, dv)* is used to compute the real challenge using the received answer *(ch)* and the random number generated *(dv)* beforehand.
- The sending of the OTP is accompanied by its hash which is generated using the HMAC-SHA256 function that takes as input the OTP and the session key. It should be added that before this whole OTP generation process is triggered, a valid biometric authentication by fingerprint of the driver is required.
- If authentication fails, there will be no response from the gate side and the process ends without performing a data update.
  Figure 2 shows the details of the functionalities at the entity level and the exchanges between the two entities.

This algorithm allows to authenticate the vehicle in a single information sending. We exploit some functions to achieve this:

- the HMAC message authentication mechanism with the sha256 hash function: HMAC-SHA256 for hash generation
- the ECDH (Elliptic Curve Diffie-Hellman) key generation principle is exploited for HMAC-SHA256 key exchange: implemented by the *challenge()* function. By this trick we manage to exchange securely the session keys for the next connection.
- the method of median squares for the transformation of the UI identifier. The portal identifies the vehicle by doing the inverse operation on the number extracted from the OTP.

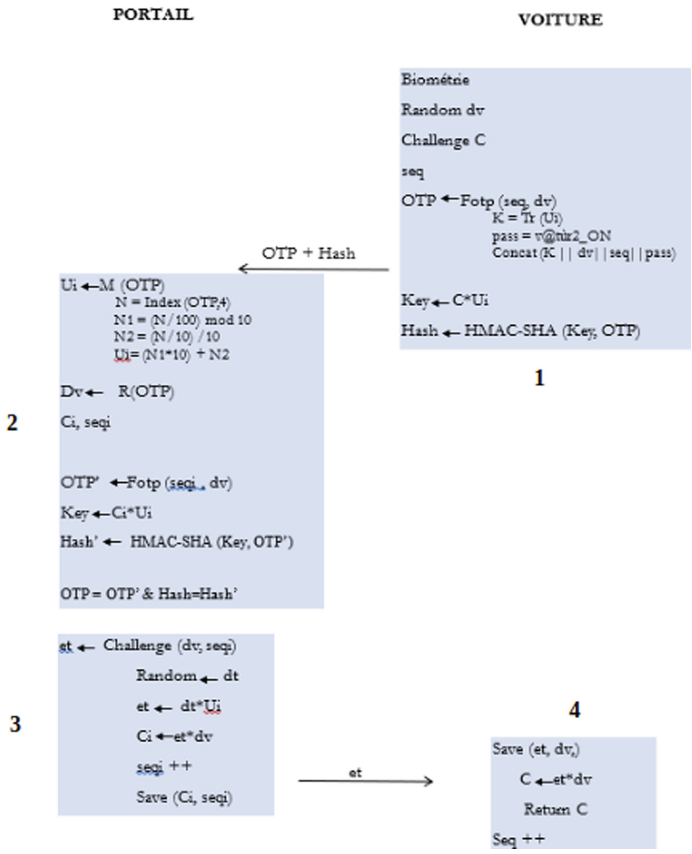


**Fig. 2.** description of functions

## 4 Performance Evaluation of Our Solution

In this section, we present experimental setup and main results.

### 4.1 Experimental Setup

Our connected gate access control system is an exchange of authentication information between the vehicle and the gate. Each sensor node, in general, communicates via radio transmission modules. We focus our study on a connected portal whose architecture is presented as follows (see Fig. 3):

- the sensor node of the vehicle consists of an Arduino Uno board equipped with an ultrasonic sensor hc05 and a transmission module nRF24L01.
- at the gate there is another Arduino Uno board equipped with a motor controller, a reception module NRF24L01, an ultrasonic sensor hc05, two (02) motors to operate the gate.

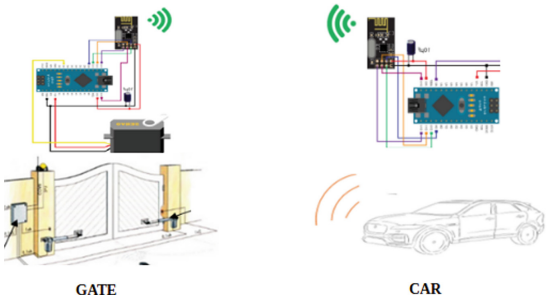


**Fig. 3.** experimental architecture

Figures 4 and 5 present extracts of source code of the different functions in C++ language. They have been edited and raised with the Arduino IDE.

```
// generation de l'otp
String fotp(int dv,int c,int seq)
{
    String otp;
    int n1 = random(1, 9)*1000;
    int n2 = ui*10;
    int n3 = random(1, 9);
    int K = n1+n2+n3;
    otp.concat(K);
    otp.concat(dv);
    otp.concat(seq);
    otp.concat(c);
    otp.concat(pass);
    return otp;
}

// recuperation de l'identifiant et dv de l'otp
int M (String otp, int n1,int n2)
{
  char u1[2];
  u1[0] = otp.charAt(n1);
  u1[1] = otp.charAt(n2);
  int u = atoi(u1);
  return u;
}
```

**Fig. 4.** OTP generation (fotp) and data recovery (M) functions

We captured the output of the algorithm through the IDE's serial monitor. Figure 6 shows us an example of execution and generation of OTPs followed by their hashes calculated by session keys.

```
// generation de la clé de hash pour la prochaine session
int challengeKey(int dv)
{
  int dt = random(22,55);
  int c = dt*ui;
  int save = c*dv;
  return c;
}

// notre fonction de hachage HMAC-SHA256
uint8_t hmacsha256(uint8_t key, int l, String otp)
{
  uint8_t *hachage ;
  Sha256.initHmac(key,l);            // clé et longueur de la clé en octets
  Sha256.print(otp);
  hachage = Sha256.resultHmac();              //Le résultat du hachage est
  return hachage;
}
// affichage du hash avec Ajout de salt au hash
void affichash(uint8_t* hash) {
  int i;
  for (i=0; i<32; i++) {
    Serial.print("0123456789abcdef"[hash[i]>>4]);
    Serial.print("0123456789abcdef"[hash[i]&0xf]);
  }
  Serial.println();
```

**Fig. 5.** the challenge exchange and hash (hmacsha256) functions

### 4.2 Responses to Functional Requirements

We analyze the performance of our authentication protocol in terms of computational costs, communication costs and storage requirements.

- Computational costs: we determine the computational cost of our algorithm by evaluating its complexity. We have a complexity O(1) which is the constant time. It is hard to beat since the execution time is always the same, regardless of the input value. However the complexity of the solution proposed by HAMMI Sect. 2 is of order $O(n)$ depending on the size of the session key chosen. We use our session key only to ensure the integrity of the OTP sent through the hash calculation.
- Communication costs: to determine the communication costs, we calculated the total bit size of the OTP transmitted during the authentication phase and we also counted the number of exchanges. Thus we have that at the output of our algorithm (see Fig. 6) an OTP that is the result of the concatenation $K$ // $dv$//$seq$//pass. We have:
  $K = xxxx$
  $dv = xx$
  $seq = x$
  $pass = yyyyyyxyyy$
  Hence $OTP = xxxxxxxyyyyxyyyy$ with $x$ and $y$ representing respectively an integer and a character. We can deduce a size of at least 16 character strings (128 bits) for the one-time passwords we generate. Also compared to the mutual authentication solution via OTPs and session keys that we saw in the Sect. 2 section, we see that our solution has a lower number of exchanges which is two (02) against four (04) in general.

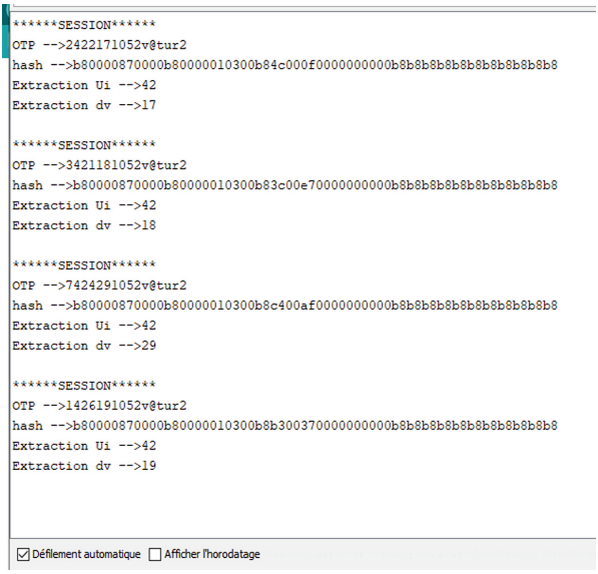


**Fig. 6.** result of a test: OTP + Hash

– Storage requirement: it's defined as the space used to store the data of the communicating entities in the system. In any protocol, this represents the domain parameters and additional data (such as credentials, shared keys and identities) stored at the end of the configuration phase. For our case, we store only three pieces of information: the identifier *(Ui)*, the challenge *(Ci)* and the authentication sequence number *(seq)*. These data occupy little memory space and only the challenge and the sequence number are updated after each authentication session.

### 4.3 Resistance to Attacks

Our security system must meet the requirements for various possible attacks. As discussed above (see Sect. 2), the attacks to which connected portals may be vulnerable are considered as our evaluation criteria in order to test the robustness of our solution. Our solution resists these attacks:

– Spoofing attack: the two communicating entities authenticate each other using OTPs. These are based on secret information pairs (not known by the intruder) and by a unique random or pseudo-random number valid only for a single use (*challenge C, dv, the hash key*). In addition, the use of the transformation with the method of squares are linked to the object identifier (*Ui*). Therefore, a node without knowledge of the personalized information can neither be authenticated nor impersonate a legitimate user.

- Replay attack: the fact that the OTP is valid for only one use, protects the system against malicious users trying to resend (replay) the same messages in order to have an unauthorized use of the system. The generation of the OTP includes the sequence number (*seq*) and since no two CONNECT packets can have the same number, no replay attack can be possible.
- Man-in-the-middle attack: our authentication protocol cannot protect the system when an intruder interrupts traffic during a communication. However, as explained in the description, if an intruder obtains all the exchanged messages, he cannot exploit them to obtain secret information or to forge a message because we ensure the integrity of the OTP by its hash that is sent to the portal. The HMAC-sha256 generation algorithm is robust because we use the session key which is dynamic in addition to the Salt that we add at the beginning and at the end of the OTP hash.
- Cryptanalysis attack: the use of the EDCH key exchange principle, random information transformation method and irreversible hash function with session keys, protects the system against any cryptanalysis attack that can be deployed to recover the keys or the exchanged data. Because it would be necessary to go back and discover all these processing methods used by our algorithm.
- Brute force attacks: the password is only valid for one node and lasts only for one session. Retrieving keys or one-time passwords for a communication session of a few seconds by a brute force attack is almost impossible. According to [15], finding a 12 character (96 bit) password using hardware with good performance can take 2 centuries. Our OTP has a length of 16 characters. Despite the evolution of technologies in computing power, the dynamic character of the exchanged passwords is an asset.
- Physical attack: an adversary can gain physical access to conduct attacks such as copying collected data (confidentiality breach), modifying or deleting collected data (integrity breach). The nodes themselves can be cloned and/or modified without the knowledge of their owner. However, our home application requires physical protection of the nodes with strong boxes and changing the default addresses of the radio module used for pipes.

## 5 Conclusion and Perspectives

Home automation networks are increasingly targeted by attackers. IoT attacks are increasing due to the ease with which hackers can penetrate IoT systems where functionality is generally privileged at the expense of security. It is essential to strengthen their security through techniques adapted to the resource constraints of the objects that are interconnected.

Authentication is the first line of defense for any system. We propose a robust and efficient authentication solution for smart home connected portals. After analyzing a connected portal solution, we were able to identify security breaches on the basic authentication mechanism used for most IoT solutions and the application constraints of some existing robust solutions. These are the authentication by ID/password or by SSL/TLS certificates.

We propose a lightweight and robust dynamic authentication solution based on one-time passwords. These OTPs are generated thanks to the exploitation of the ECDHE key exchange principles for the exchange of session keys used by a hash function. Our solution ensures a perfect mutual authentication adapted to the context where the authentication step follows to make decisions. It uses a single information exchange to ensure the action. It uses elliptic curve algorithms that are less resource intensive (CPU, memory). The effectiveness of the encryption mechanisms coupled with the dynamics of the passwords allows us to have a robust and efficient solution. Thus our solution would be more robust if we manage to reinforce the confidentiality of the exchanged OTP by exploiting or adapting the mechanisms of the light cryptography for the objects limited in computing power. This could allow an application of our solution in all types of IoT networks by strengthening the authentication mechanisms of the IoT lightweight protocols.

## References

1. Hammi, B., Khatoun, R., Zeadally, S., Fayad, A., Khoukhi, L.: IoT technologiesfor smart cities. IET Netw. **7**(1), 1–13 (2018). eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/iet-net.2017.0163
2. View, M., M'Raihi, D., Hoornaert, F., Naccache, D., Bellare, M., Ranen, O.: HOTP: An HMAC-Based One-Time Password Algorithm. Request for Comments RFC 4226, Internet Engineering Task Force (2005). Num Pages: 37
3. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: keyed-hashing for message authentication. Technical report, RFC2104, RFC Editor (1997)
4. View, M., Rydell, J., Pei, M., Machani, S.: TOTP: Time-Based One-Time Password Algorithm. Request for Comments RFC 6238, Internet Engineering Task Force (2011)
5. Hammi, M.T., Livolant, E., Bellot, P., Serhrouchni, A., Minet, P.: A lightweight IoT security protocol. In: 2017 1st Cyber Security in Networking Conference (CSNet), pp. 1–8 (2017)
6. Hammi, M.T., Livolant, E., Bellot, P., Serhrouchni, A., Minet, P.: A lightweight mutual authentication protocol for the IoT. In: Kim, K.J., Joukov, N. (eds.) ICMWT 2017. LNEE, vol. 425, pp. 3–12. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-5281-1_1
7. Rescorla, E., Dierks, T.: The Transport Layer Security (TLS) Protocol Version 1.2. Request for Comments RFC 5246, Internet Engineering Task Force (2008)
8. Han, J.-H., Kim, J.: A lightweight authentication mechanism between IoT devices. In: 2017 International Conference on Information and Communication Technology Convergence (ICTC), pp. 1153–1155 (2017)
9. Makhtoum, H.E.L., Bentaleb, Y.: An improved IOT authentication process based on distributed OTP and Blake2. IJWMT 11, 1–8 (2021). Ibn Tofail University/Engineering sciences laboratory, Kenitra, Morocco
10. Hammi, B., Fayad, A., Khatoun, R., Zeadally, S., Begriche, Y.: A lightweight ECC-based authentication scheme for internet of things (IoT). IEEE Syst. J. **14**(3), 3440–3450 (2020)
11. Igoe, K., McGrew, D., Salter, M.: Fundamental Elliptic Curve Cryptography Algorithms. Request for Comments RFC 6090, Internet Engineering Task Force (2011)

12. Blumenthal, U., Goel, P.: Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS). Request for Comments RFC 4785, Internet Engineering Task Force (2007)
13. Yan, S.C.S., et al.: Authentication of IoT device with the enhancement of one-time password (OTP). JITA **9**, 29–40 (2021)
14. Esfahani, A., et al.: A lightweight authentication mechanism for M2M communications in industrial IoT environment. IEEE Internet Things J. **6**, 288–296 (2019)
15. https://www.betterbuys.com/estimating-password-cracking-times/. Estimating Password Cracking Times