symmetry

MDPI

Article

PUF Based Authentication Protocol for IoT

An Braeken

Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium; an.braeken@gmail.com; Tel.: +32-468-104-767

Received: 11 July 2018; Accepted: 11 August 2018; Published: 20 August 2018

Abstract: Key agreement between two constrained Internet of Things (IoT) devices that have not met each other is an essential feature to provide in order to establish trust among its users. Physical Unclonable Functions (PUFs) on a device represent a low cost primitive exploiting the unique random patterns in the device and have been already applied in a multitude of applications for secure key generation and key agreement in order to avoid an attacker to take over the identity of a tampered device, whose key material has been extracted. This paper shows that the key agreement scheme of a recently proposed PUF based protocol, presented by Chatterjee et al., for Internet of Things (IoT) is vulnerable for man-in-the-middle, impersonation, and replay attacks in the Yao–Dolev security model. We propose an alternative scheme, which is able to solve these issues and can provide in addition a more efficient key agreement and subsequently a communication phase between two IoT devices connected to the same authentication server. The scheme also offers identity based authentication and repudiation, when only using elliptic curve multiplications and additions, instead of the compute intensive pairing operations.

Keywords: physical unclonable function; authentication; elliptic curve cryptography; internet of things

1. Introduction

Internet of Things (IoT) is experiencing worldwide growth. Not only classical computing and communication devices are connected, but also a whole range of other gadgets that are used in our daily life, such as thermostats, light switches, door locks, refrigerators, etc. Typically, these devices are characterised by a very small amount of memory and computational power. The communication between these devices over the Internet and on local networks needs to be secured to gain trust and acceptance and to avoid direct physical harm to humans, even loss of life. By secure communication, we mainly consider the security features of confidentiality, integrity of the transmitted messages, and authentication of the sending and receiving devices [1].

In 2001, Physical Unclonable Functions (PUFs) have been introduced as an interesting cryptographic primitive [2] and can be seen as the hardware equivalent of a one-way function. A silicon PUF is a physical entity embodied in a physical structure that is easy to fabricate but practically infeasible to clone, duplicate or predict, even if the exact manufacturing process is produced again. Instead of using a private key that is linked to the device identity, the authentication of PUFs is based on the usage of so-called challenge–response pairs. The electrical stimulus, called the challenge, is applied to the physical structure in order to react, called the response, in an unpredictable manner due to the complex interaction of the stimulus with the physical micro-structure of the device, which is dependent on physical factors introduced during the manufacturing process in an unpredictable and uncloneable manner. PUFs have relatively low hardware overhead and are thus very interesting in an IoT context. PUFs on devices have been already applied for device identification and authentication, binding hardware to software platforms, secure key storage, keyless secure communication, etc. Figure 1

illustrates the usage of PUFs for device identification as they can be seen as the unique fingerprint of the device.

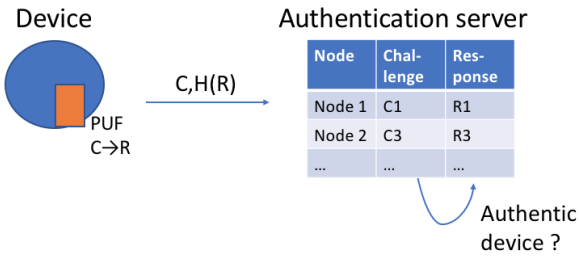

Figure 1. Example of Physical Unclonable Functions (PUFs) usage for device identification (C = Challenge, R = Response).

A protocol is called a PUF based protocol when at least one of the entities is able to construct PUF challenges and responses, which are used in the rest of the protocol. We assume for our construction the existence of a PUF mechanism in the devices.

In [3], Chatterjee et al. proposed a PUF based protocol for secure communication among two IoT devices that are connected to the same server, where the server has securely stored a list of challenge–response pairs for each device. The scheme consists of a PUF based key agreement protocol and a secure communication protocol. The PUF is used to generate the public key of each device in the key agreement protocol. Unfortunately, we show that their protocol is not resistant to man-in-the-middle, impersonation, and replay attacks. In addition, we present an alternative PUF based protocol for the key agreement phase, which is even more efficient. Also for the secure communication phase, we show that there are more efficient solutions in the literature as the proposed one. Similar to [3], for the initialization phase, a large amount of challenges and responses is stored on the server side and used during the key agreement phase to construct a public key, certified by the authentication server.

2. Related Work

We here focus the related work on the key agreement protocols in IoT as this is the part where PUFs are involved. For the secure communication phase, we can refer to the multiple identity based signcryption schemes [4,5] in literature, which are cryptographic primitives in which both signature generation and encryption are performed in one single phase.

Two main categories of key agreement schemes in IoT can be distinguished, being key agreement from device to server and key agreement between two different devices. Note that we do not get involved in the discussion of IoT devices with user access, where two-factor and three-factor authentication is possible. We refer to the survey of [6] for these types of schemes and [7] for an anonymous symmetric key based key agreement protocol. The most important trends and schemes in each of these categories are now discussed.

2.1. Key Agreement from IoT Device to Server

For the first category of key agreement schemes, for simplicity, we assume that both device and server are registered with the same trusted third party (TTP). If not, the negotiation is further dealt on the level of the TTPs. In the registration phase, there are two possible types of key material to be shared at the device side in order to guarantee that the identity of the device is linked with the claimed key material. The first type is through classical identity based public key cryptography. This can be arranged using different types of key establishment schemes, being the typical identity based schemes [8], certificateless [9] and certificate based schemes [10]. The identity based key establishment schemes require the usage of computationally intensive pairing operations and have inherent key

escrow. Moreover, the schemes are vulnerable for a curious and honest TTP, which is a TTP that is honest in the sense that it performs all the required steps but is curious in retrieving the data for own purposes (e.g., selling to third parties). Both the key escrow problem and the vulnerability for a curious and honest TTP can be avoided in the other two types of key establishment schemes since the resulting private key is generated there by means of secret information coming both from the device and the TTP. The main difference between certificateless and certificate based schemes is that, in the first one, a secure channel is required for sharing the secret information of the device. Consequently, the certificate based key establishment schemes offer the most interesting features and the Elliptic Curve Qu-VanStone (ECQV) [11] certificates mechanism is considered as the most efficient one as it is based on Elliptic Curve Cryptography (ECC). In the ECQV scheme, the public key of the user is generated by means of the identity of the user and the certificate generated by the TTP. Consequently, instead of sharing the public key and identity of the user, it is sufficient to share the certificate and the identity of the user. Any other party, knowing the public key of the TTP, is then able to generate the public key of the user. Note that our proposed key agreement scheme is based on this principle. A key agreement mechanisms based on the ECQV scheme has been proposed in [12]. In [12], Porambage et al. have proposed a key agreement protocol between sensor and server using the ECQV principle to be applied in the context of wireless sensor networks in distributed IoT applications.

The second type of key material, which can be used in these key agreement protocols, are the PUF based challenges and responses. The main advantage with PUF based key material is that the attacker cannot take over the identity of a tampered device, whose key material has been extracted. There exist multiple PUF based key agreement protocols for device to server in literature. In [13], twenty-one server-device/token key agreement protocols have been classified with respect to the features, device authenticity, server authenticity, device privacy, leakage resilience, number of authentications, resistance to noise, modelling resistance, denial of service (DoS) prevention, and scalability. It has been shown that only a very limited number is able to satisfy these features in a reasonable level. The main problems were vulnerability for DoS attacks, replay attacks, impersonation attacks, and synchronization problems. In the lightweight category of proposals are the Slender PUF, [14] noise bifurcation [15] and PUF lockdown protocol [16] retained, while in the non-lightweight category only Reference protocol II-A [13] and the protocol proposed by Sadegi et al. [17]. The main difference between these protocols [13–17] and our PUF based protocol is that these protocols take noisiness of the PUF into account, while our protocol considers the usage of a strong and controlled PUF. Moreover, Refs. [14,15] also take countermeasures to offer resistance against machine learning attacks, although they cannot be completely excluded [13]. The proposed protocol in [16] prevents an attacker from querying a token for CRPs that have not yet been disclosed during a prior protocol run. The main weakness of [17] is that it does not scale well, as the server needs to exhaustively evaluate a pseudo random function for each registered token. Another method for key agreement with the usage of PUFs is described in [18]. Here, the private key of the device is securely stored using a PUF construction. During the first communication message, the certificate issued by the manufacturer needs to be included. This approach is interesting, but strongly relies on the trustworthiness of the manufacturer, which is in many cases not verifiable by the device. In [19], the concept has been explained of how PUFs in combination with Blockchains are able to establish authentication for IoT devices. Although the idea is promising, the impact of Blockchains on the performance of IoT devices is not fully clear for the moment.

2.2. Key Agreement between Two IoT Devices

For the key agreement between two IoT devices, e.g., in the case of sensor nodes in automobiles, smart home and smart cities, we assume that both devices are registered with the same TTP. During the key agreement process, the TTP can be involved or not. In the case when the TTP is not involved, the most efficient identity based authentication and key agreement protocol proposed in literature can be found in [20], which only requires two phases and is also based on the ECQV key establishment

protocol. We also mention in this context the standard key agreement scheme in IoT, called the Datagram Transport Layer Security (DTLS) protocol [21] with raw public keys, which has as a main difference the usage of less efficient certificates. In [22], the authors have replaced in the DTLS protocol these more compute intensive certificates by the ECQV certificates and evaluated the impact of it on the DTLS protocol.

However, without involvement of the TTP, a revoked device cannot be detected by the other party, without storage of a revocation list, which is very memory demanding and not advisable in a constrained IoT device. Therefore, it makes sense to also consider key agreement protocols with an active involvement of the TTP. Such protocol using classical public key cryptography mechanisms is evident, assuming that the TTP stores the list of valid identities and corresponding public keys of the participating IoT devices. Note that this scheme will be used as a benchmark to also compare the efficiency of our proposed scheme (see Figure 5).

For the PUF based key agreement protocols between two IoT nodes with the aid of a common server (taking the role of TTP), who has stored the challenge–response pairs of the PUFs from the different nodes, Chatterjee et al. recently proposed a protocol in [3]. The public keys of the devices are generated using the PUF results, followed by an identity based encryption mechanism for the actual secure communication. We show in this paper several weaknesses of this PUF based key agreement protocol and present a corresponding solution.

3. Preliminaries

3.1. System Architecture

For an easier comparison and replacement of a solution, we consider similar architecture and protocol phases as in [3].

Consequently, the scheme consists of an enrolment phase, authentication and key sharing phase, and a final secure communication phase.

- In the enrolment phase, each IoT node shares through a secure channel challenge–response pairs with a server, which is considered as a trusted and authenticated party.
- In the authentication and key sharing phase, two nodes supervised by the same server are able to generate a private and public key pair. The public key of the one node is shared with the other with the help of the server to guarantee the authenticity. Note that we here also consider this simplified setting. In [3], details are given on how to deal with the situation where two nodes are registered with a different server.
- Finally, the secure communication phase allows both nodes to start a secure communication.

3.2. Cryptographic Operations

We first briefly explain the two most important building blocks to be used in the scheme. Next, some other required operations are shortly mentioned.

3.2.1. PUFs

There are two types of PUFs: strong and weak. The difference is related to the number of responses that can be generated. A strong PUF is able to generate a large amount of challenge–response pairs, while, for a weak PUF, the number is limited, often to one.

The practical realisation of a strong PUF is challenging. PUFs also have problems for stabilising the noise when generating the responses. In order to solve this issue, error correcting codes and assisting computed helper data are required. A good construction of both is essential to avoid leakage of information and resistance against fault and reliability attacks. Recently, [23] proposes the construction of PUF-FSM, a controlled strong PUF without the need for error correcting codes and helper data by only using the error-free responses, which are fed in a finite state machine. We do not focus on the

design of a PUF in this paper, but assume the usage of this type of PUF-FSM in our protocol. Thus, we are able to generate a large amount of challenges and responses in a controlled way. Note that also in [3] a PUF with these features is required. In contrast, they designed their own PUF, even without error correction codes and helper data, assuming the absence of noise, which is not a realistic assumption. When using the PUF-FSM, we can assume that the PUF evaluation behaves as a random oracle.

3.2.2. Public Key Related Operations

The public key related operations in our proposed scheme rely on ECC [24], offering more lightweight public key cryptographic operations than the classical discrete logarithms or RSA based systems. For example, 1024-bit RSA corresponds with a 160-bit ECC scheme from a security point of view.

Let us denote the elliptic curve (EC) $E_{p(a,b)}$ to be used in our scheme by $y^2 = x^3 + ax + b$ with a and b two constants in F_p and $\Delta = 4a^3 + 27b^2 \neq 0$. In [25,26], standardized curve parameters are described for p between 113 and 571 bits. The base point generator P of $E_{p(a,b)}$ is of prime order q. All points on $E_{p(a,b)}$, together with the infinite point form an additive group.There are two elementary operations related to ECC resulting in another point of the EC, the EC multiplication $R = rP$ with $r \in F_q$ and the EC addition $R_1 + R_2$. ECC relies on two computational hard problems.

- The Elliptic Curve Discrete Logarithm Problem (ECDLP). This problem states that, given two EC points R and Q of $E_{p(a,b)}$), it is computationally hard for any polynomial-time bounded algorithm to determine a parameter $x \in F_q^*$, such that $Q = xR$.
- The Elliptic Curve Diffie–Hellman Problem (ECDHP). Given two EC points $R = xP, Q = yP$ with two unknown parameters $x, y \in F_q^*$ and P, the base point generator of the defined EC, it is computationally hard for any polynomial-time bounded algorithm to determine the EC point xyP.

3.2.3. Other Operations

Furthermore, we denote the operation H as the one-way cryptographic hash function (e.g., SHA2 or SHA3) that results in a number of F_q. The concatenation of two messages M_1 and M_2 is denoted by $M_1 \| M_2$.

We assume that these functions, the EC parameters together with the EC operations, and a reliable and secure PUF are implemented in each node participating the scheme. The server has also implemented all these parameters and functions, except the PUF.

4. Security Attacks on the System of Chatterjee et al.

We start with a short description of the enrolment phase and the authentication and key sharing phase in [3]. Next, we explain the different attacks in more detail and the main lessons to be learned.

4.1. Enrolment Phase

In the enrolment phase of the IoT device with the server, the server sends a set of k challenges to the device. Each challenge is passed through the PUF mechanism of the IoT device in order to obtain the corresponding response. The k resulting responses are sent to the server using a secure channel. The server is responsible for the secure storage of these k challenge–response pairs for each IoT device.

4.2. Description of the Authentication and Key Sharing Phase

Figure 2 describes the main steps to be performed by the IoT nodes 1 and 2 and the server. In short, the following steps are executed.

- Node 1 sends a request to the server that it wants to communicate with Node 2.
- The server randomly selects two challenges C_1, C_2 related to Node 1 and two challenges C_3, C_4 related to Node 2 from its database. Next, it calculates the values Δ_1, Δ_2 by hashing the corresponding responses R_1, R_2 and R_3, R_4, respectively, with the timestamp TS. Next, this

TS is hidden for outsiders by xoring it with the concatenation of the responses of Node 1 and Node 2 in order to obtain TS'_1 and TS'_2, respectively. Finally, the Server sends C_1, C_2, TS'_1 to Node 1 and C_3, C_4, TS'_2 to Node 2.

- Upon arrival of these messages, Node 1 and Node 2 perform similar actions in the rest of the protocol. Let us explain the steps of Node 1. First, it derives TS from TS'_1 after evaluating the PUF of C_1 and C_2. Next, it computes ID_1 by hashing the responses of the challenge with the obtained TS. Then, it calculates P_1, which is the hash of the xor of the two challenges and results in an ECC point. Based on a randomly chosen value $t \in F_q$, the public key P_{1PUB} is computed as the EC multiplication of t with P_1. The corresponding private key P_{1PRV} equals $t \cdot ID_1$, which is transferred to an EC point. Next, a hash function of the xor of the different obtained values is computed to guarantee the integrity, resulting in d_1. The message containing ID_1, P_1, K_{1PUB}, d_1 is sent to the server.
- The server first authenticates Node 1 by checking the equality of ID_1 with the previously computed value Δ_1 and the correctness of d_1. If so, the received public key K_{2PUB} of Node 2, together with ID_2 and P_2, is sent to Node 1. A hash function on the xor of these values, together with the responses R_1, R_2, is sent to Node 1.

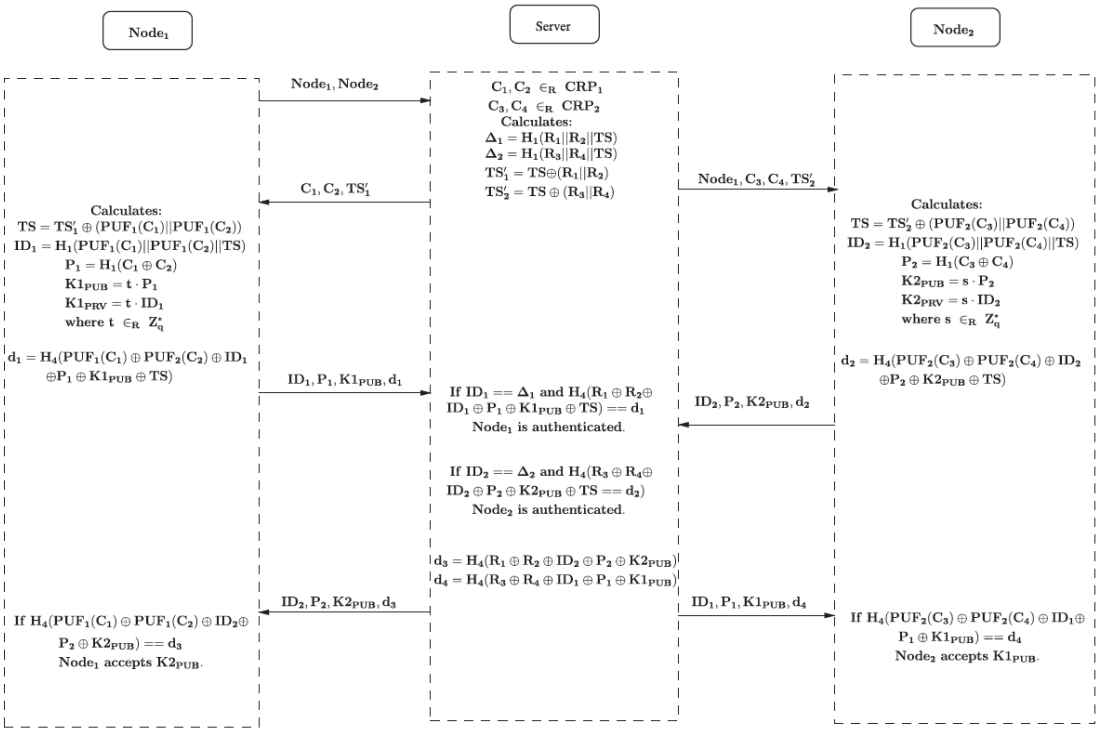

Figure 2. Key agreement scheme [3].

4.3. Security Attacks

We consider the Dolev–Yao attack model [27]. Here, the attacker is able to eavesdrop on the traffic, inject new messages, replay and change messages, or spoof other identities. Consequently, the attack is only limited by the constraints of the cryptographic methods used. We may assume that the attacker carries the message. The main goal is to obtain illegitimate data access.

To be more in practice, eavesdropping, intercepting and modifying data are activities that can be launched at any point that the traffic passes from the IoT device to the authentication server. Some examples of how this can happen are as follows:

- In a local network:

  - Anyone who is connected to the same Wi-Fi is able to read the traffic.
  - If the router (or some other part of the network) is hacked, the hacker can read and modify the traffic.
  - The person that legitimately controls the network, e.g., the responsibility of the authentication server can read and modify the traffic without even having to hack anything.
- Over the internet:
  - The Internet Service Provider (ISP) is able to read and modify all the traffic, since it controls the hardware it passes through. The traffic can also go through other networks owned by unknown companies, eventually of different countries, and those can also read and modify the traffic as well.
  - When a nation state hands over a court to one of these companies passing the traffic, it can also read and modify the data (e.g., NSA).

We have discovered two man-in-the middle attacks, an impersonation attack with a malicious insider node, a replay attack, and weaknesses related to DoS attacks for the scheme of [3]. We now discuss these attacks more in detail.

4.3.1. Man-in-the-Middle Attack

There are two different man-in-the-middle attacks distinguished. One attack happens in the beginning of the protocol and the other at the end.

The first one is very trivial and related to the fact that in the request, Node 1 submits both the names of Node_1 and Node_2 in cleartext. In addition, the server sends to Node_2 the name of Node_1 in cleartext. In the rest of the protocol, these names are never linked with the resulting obtained values (challenges, responses, timestamp, etc.), both from the side of Node 1 and Node 2. As a consequence, Node 1 is not guaranteed to receive the security parameters of Node 2 and Node 2 is not guaranteed to communicate with Node 1 during the last step of the key agreement scheme, as an attacker can change the names of these nodes while intercepting the messages without being noticed by the nodes. As a result, during the secure communication scheme, it is not guaranteed that Node 1 is using the credentials belonging to Node 2 and vice versa.

In the second attack, the attacker changes the values ID_2, P_2, P_{2PUB} in the last step of the key agreement to Node 1 and leaves d_3 unchanged. Consequently, the attacker should find $ID_2^*, P_2^*, P_{2PUB}^*$ for which $P_{2PUB}^* \oplus P_2^* \oplus ID_2^* = P_{2PUB} \oplus P_2 \oplus ID_2$. In order to find such a tuple, it first chooses a random value $s^* \in F_q$ and a random point P_2^* of the EC. Next, it computes $P_{2PUB}^* = s^* P_2^*$. Then, it chooses ID_2^* such that it satisfies the following equality:

$$ID_2^* = (s^*)^{-1} \cdot (P_{2PUB}^* \oplus P_2^* \oplus P_{2PUB} \oplus P_2 \oplus ID_2).$$

As a consequence, the attacker has computed a valid private and public key to be used in the communication protocol, without being noticed by Node 1 since the resulting hash value d_3 is left unchanged. A similar process can be performed in the communication to Node 2 by changing the values ID_1, P_1, P_{1PUB} and leaving d_4 unchanged.

4.3.2. Impersonation Attack with Malicious Insider Node

In [3], it has been proven that the protocol was able to withstand impersonation attacks in the case of malicious nodes. Moreover, it was claimed that a small list of challenge–response pairs per node was required as they could not be reused.

Unfortunately, we show that this is not the case since Node 1 can easily retrieve the responses R_3, R_4 from the transmitted message of the server to Node 2. This follows from the fact that TS is the same in TS_1' and TS_2'. Similarly, Node 2 can retrieve the responses R_1, R_2 from TS_1'. As a consequence, in case the server reuses the same challenges in a communication request with another entity, the malicious

node can use the responses to impersonate that node. Even stronger, at this point on, the malicious node is able to impersonate the server and to send a communication request from a fake entity.

4.3.3. Replay Attack

Although a timestamp is involved in the first steps of the protocol and embedded in the identities of the node, the validity of the timestamp cannot be verified by the nodes in the last phase. Consequently, if the same pair of challenges is reused twice, the credentials of another entity $ID_i^*, P_i^*, K_{iPUB}^*$ can easily replace the ones of the original message, without being notified by the node.

4.3.4. Denial of Service Attack

For launching DoS attacks, first an open port should be found and then lots of traffic needs to be sent. The scheme is extremely weak with respect to this type of attack in particular at the side of the nodes. This attack can lead to serious energy depletion at the nodes.

For instance, an attacker can send a massive amount of requests with challenges immediately to the nodes. Help can be found by other hackers or infected zombies, which leads to a distributed Denial of Service (DDoS) attack. Upon arrival of these messages, the node needs to compute each time a computational intensive EC multiplication and two hash-to-point operations. Only when receiving the resulting parameters of these operations by the server, contradictions will be noticed. The key factor for success is the scale of the attack, being the amount of requests, multiplied by the complexity of the requests. The main problem here in the design is that the nodes are unable to check the validity of the message containing the challenges using less compute intensive operations.

4.4. Conclusion of the Security Attacks

The main lessons learned from the detected security attacks on the protocol are the following:

1. The node identity used in the initial request should be utilised throughout the whole protocol.
2. The timestamp should be verifiable by all entities in each step of the protocol up to the end.
3. The integrity and authentication of the message should be checked at each single step in the protocol.
4. The responses should only be sent under a hash value in order not to leak information on them and to allow reusability of the challenge–response pairs.

5. Proposed Solution

We assume the same enrolment phase as in [3]. We propose a new version for both the authentication and key agreement phase and the secure communication phase.

5.1. Authentication and Key Agreement Phase

Due to the symmetry of the protocol, it suffices to explain the steps from the side of Node 1 after the second step. Figure 3 illustrates the complete overview of the different steps.

- To start the key agreement phase, Node 1 shares with the server the identities ID_1, ID_2 of Nodes 1 and 2, respectively, together with the current timestamp TS. Node 1 opens a new session and stores the three parameters ID_1, ID_2, TS in it.
- Upon arrival of the request, the server checks its database for the existence of ID_1 and ID_2. If so, it then selects two pairs $(C_1, R_1), (C_2, R_2)$ of Node 1 and two pairs $(C_3, R_3), (C_4, R_4)$ of Node 2 from its database. Based on this information, it computes both

$$\begin{aligned} C_2' &= C_2 \oplus H(R_1 \| TS \| ID_2), \\ C_4' &= C_4 \oplus H(R_3 \| TS \| ID_1). \end{aligned}$$

In addition, a hash value to ensure the integrity and authentication of the server needs to be included. This value for Nodes 1 and 2, respectively, is computed as follows:

$$h_{11} = H(C_1 \| C_2 \| TS),$$
$$h_{12} = H(C_3 \| C_4 \| TS).$$

The message C_1, C_2', h_{11} is sent to Node 1 and $ID_1, TS, C_3, C_4', h_{12}$ is sent to Node 2. The server opens a new session in which the parameters $ID_1, ID_2, R_1, R_2, R_3, R_4, TS$ are stored.

- Upon arrival of the message, Node 1 first checks if the message consists of three parameters of the expected length. If so, it opens the stored session and retrieves the parameters ID_1, ID_2, TS. Then, Node 1 computes R_1 for C_1 in order to derive C_2. This allows for verifying h_{11}. If correct, Node 1 also derives R_2 and randomly chooses $q_1 \in F_q$ to compute $Q_1 = q_1 P$. Next, it computes $h_{21} = H(R_1 \| R_2 \| TS \| ID_2 \| Q_1)$. The message (ID_1, h_{21}, Q_1) is sent to the server and R_1, R_2 are additionally stored in the session. Similarly, Node 2 generates the message (ID_2, h_{22}, Q_2) and the parameters ID_1, ID_2, TS, R_3, R_4 are stored in a new session at Node 2.
- Upon arrival of messages consisting of two parameters of the expected length, the server first verifies h_{21}, h_{22} for the identities involved in any of the open sessions stored on the server side, where the current timestamp is in a reasonable timeframe with the corresponding stored timestamp. The verification of h_{21}, h_{22} ensures the integrity and authenticity of both nodes. If it is correct, it starts with the derivation of the certificates for both nodes. Therefore, it randomly chooses a variable $q_s \in F_q$ and computes $Q_S = q_s P$. Next, the two certificates of the nodes are computed as c_1 and c_2:

$$\begin{aligned} c_1 &= Q_1 + Q_s, \\ c_2 &= Q_2 + Q_s. \end{aligned}$$

These certificates are used to compute the auxiliary information r_1, r_2 for Node 1 and Node 2 to compute their private and public key pair, respectively. Recall that (d_s, P_s) is the private-public key pair of the server, where P_s is publicly published:

$$\begin{aligned} r_1 &= H(c_1 \| TS \| ID_1 \| ID_2) q_s + d_s, \\ r_2 &= H(c_2 \| TS \| ID_1 \| ID_2) q_s + d_s. \end{aligned}$$

In addition, to guarantee the integrity of the communication, the values h_{31}, h_{32} are also computed:

$$\begin{aligned} h_{31} &= H(R_1 \| R_2 \| r_1 \| c_1 \| c_2), \\ h_{32} &= H(R_3 \| R_4 \| r_2 \| c_2 \| c_1). \end{aligned}$$

The values r_1, c_1, c_2, h_{31} are sent to Node 1. The values r_2, c_2, c_1, h_{32} are sent to Node 2. The stored session is now closed on the server side.

- When a message of four parameters of expected length is received, Node 1 opens the stored session(s) where the current timestamp is in a reasonable timeframe with the stored timestamp. Node 1 first checks the hash value h_{31}. If OK, it continues with the computation of its private key $d_1 = H(c_1 \| TS \| ID_1 \| ID_2) q_1 + r_1$. Its public key P_1 equals $d_1 P$, but also to

$$P_1 = H(c_1 \| TS \| ID_1 \| ID_2) c_1 + P_s.$$

Consequently, given the publicly available values TS, ID_1, ID_2, c_1, P_S, everybody is able to compute this key. The private key is only known by the node itself. Using c_2, Node 1 is able to compute the public key P_2 of Node 2. Note that this mechanism is based on the ECQV Implicit Certificate Scheme [11]. After a successful generation of the key material, both nodes close the

stored session and open a new session storing its private and public key pair together with the public key and identity of the other node.

Node 1	Server	Node 2
	$\xrightarrow{ID_1 \| ID_2 \| TS}$	
	$C_2' = C_2 \oplus H(R_1 \| TS \| ID_2)$ $h_{11} = H(C_1 \| C_2 \| TS)$ $C_4' = C_4 \oplus H(R_3 \| TS \| ID_1)$ $h_{12} = H(C_3 \| C_4 \| TS)$	
	$\xleftarrow{C_1 \| C_2' \| h_{11}}$	$\xrightarrow{ID_1 \| TS \| C_3 \| C_4' \| h_{12}}$
Derive R_1 $C_2 = C_2' \oplus H(R_1 \| TS \| ID_2)$ Check h_{11} Derive R_2 $q_1 \in F_q, Q_1 = q_1 P$ $h_{21} = H(R_1 \| R_2 \| TS \| ID_2 \| Q_1)$		Derive R_3 $C_4 = C_4' \oplus H(R_3 \| TS \| ID_1)$ Check h_{12} Derive R_4 $q_2 \in F_q, Q_2 = q_2 P$ $h_{22} = H(R_3 \| R_4 \| TS \| ID_1 \| Q_2)$
	$\xrightarrow{h_{21} \| Q_1}$	$\xleftarrow{h_{22} \| Q_2}$
	Check h_{21}, h_{22} $q_s \in F_q, Q_s = q_s P$ $c_1 = Q_1 + Q_s$ $r_1 = H(c_1 \| TS \| ID_1 \| ID_2) q_s + d_s$ $h_{31} = H(R_1 \| R_2 \| r_1 \| c_1 \| c_2)$ $c_2 = Q_2 + Q_s$ $r_2 = H(c_2 \| TS \| ID_1 \| ID_2) q_s + d_s$ $h_{32} = H(R_3 \| R_4 \| r_2 \| c_2 \| c_1)$	
	$\xleftarrow{r_1 \| c_1 \| c_2 \| h_{31}}$	$\xrightarrow{r_2 \| c_2 \| c_1 \| h_{32}}$
$H(R_1 \| R_2 \| r_1 \| c_1 \| c_2) == h_{31}$? $d_1 = H(c_1 \| TS \| ID_1 \| ID_2) q_1 + r_1$ $P_1 = d_1 P$ $P_2 = H(c_2 \| TS \| ID_1 \| ID_2) c_2 + P_s$		$H(R_3 \| R_4 \| r_2 \| c_2 \| c_1 == h_{32}$? $d_2 = H(c_2 \| TS \| ID_1 \| ID_2) q_2 + r_2$ $P_2 = d_2 P$ $P_1 = H(c_1 \| TS \| ID_1 \| ID_2) c_1 + P_s$

Figure 3. Proposed key agreement scheme.

5.2. Secure Communication Phase

At the beginning of this phase, both nodes possess the identity and the corresponding public key of the other entity, which is authenticated by the server. As a consequence, a signcryption scheme, which is a mechanism providing authentication and encryption in one phase, based on EC cryptography can now be applied. We propose using the currently most efficient and secure scheme available in the literature [5]. It has been proven in the random oracle model [28] that the scheme is chosen-ciphertext secure and existentially unforgeable. Figure 4 presents the different steps to be provided in the protocol.

Node 1	Node 2
Choose $q \in F_q, Q = qP$ $K = qP_2$ $c = m \oplus H(K)$ $h = H(m \| Q \| ID_1 \| c_1 \| TS)$ $d = q - hd_1$	
$\xrightarrow{h \| c \| d}$	
	$Q' = dP + hP_1$ $K' = d_2 Q'$ $m' = c \oplus H(K')$ $h == H(m' \| Q' \| ID_1 \| m' \| TS)$

Figure 4. Proposed secure communication scheme.

6. Security Evaluation

We now discuss the security features established in the proposed scheme. The focus is on the key agreement scheme, as the communication scheme corresponds with a recently published signcryption scheme [5], whose security has been proven in [28].

- *Integrity.* Integrity is obtained in every step of the protocol since every message contains a hash, which includes the other parts (or derivations of it) of the message.
- *Authentication of the Node.* The nodes are authenticated if the check on the values h_{21}, h_{22} are correct. These hash values contain the responses on the challenges, which can only be generated by the node possessing the PUF.
- *Authentication of the Server.* The server is authenticated by the node in both received messages. In the first received message by Node 1, the value h_{11}, computed by the server, includes C_2, while this value is not part of the received message. Instead, C_2 is masked by a hash value, which includes the response on the first challenge. Consequently, as only the server is aware of the challenge–response pairs, Node 1 ensures that the server has sent the message. In the second received message, the hash value h_{31} is included. This hash value contains the responses on the two challenges, which is only known by the server. Due to the symmetry of the protocol, the same reasoning holds for Node 2.
- *Resistance against man-in-the-middle attacks.* As the authenticity of the sender is verified by the receiver in each step of the protocol, resistance against man-in-the middle attacks is guaranteed.
- *Resistance against impersonation attacks.* Even if Node 1 is malicious, it is not possible for the node to derive information on the challenge–response pairs of the other node from the messages exchanged between the server and Node 2. This follows from the fact that only hash values on the responses are included in the messages, which do not leak any information.
- *Resistance against replay attacks.* Replay attacks are avoided, due to the usage of the timestamp, which is included in all the hashes of the different transmitted messages.
- *Protection against denial of service attacks.* Besides the first message, both the nodes and the server can check at each step of the key agreement scheme the integrity and authentication by verifying the hash included in the messages. Consequently, in case a massive amount of false messages flow over the network, it will be very quickly discovered.

7. Performance

Both the computational and communication costs are considered. We compare the efficiency of our solution with [3], as it is the only PUF-based protocol in the literature with comparable architecture, i.e.; deriving a common secret key among two IoT devices.

Note that both protocols consist of two parts, being the key agreement phase and the secure communication phase. For the secure communication phase, the scheme of [5] is used, which has been shown to outperform the existing literature [28,29]. With respect to the key agreement scheme, which is based on the existence of PUFs, we utilize a straightforward public key variant as benchmark. The different steps to be performed in the benchmark scheme are presented in Figure 5. In this scheme, we assume that the authentication server stores the identity and public key of each of the registered IoT devices, instead of the challenge–response pairs. The IoT devices possess the public key P_s of the central server. Denote the private-public key pairs of Node 1, Node 2, and the authentication server by (d_1, P_1), (d_2, P_2) and (d_s, P_s), respectively. The symmetric key encryption of message M under key K to obtain the ciphertext C is denoted by $C = E_K(M)$ and the corresponding decryption operation by $M = D_K(C)$.

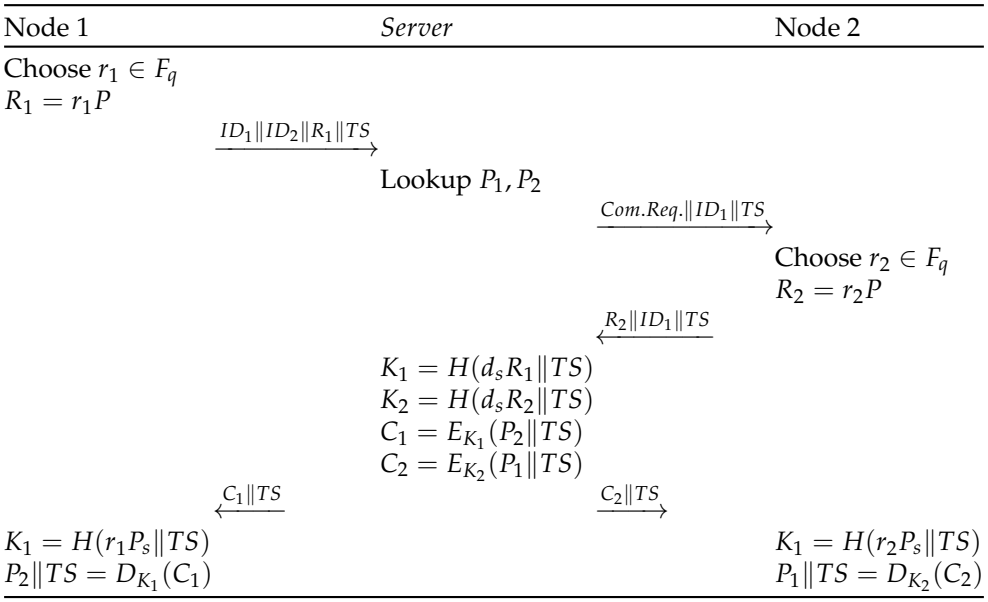

Figure 5. Key agreement scheme based on public key cryptography used as a benchmark.

7.1. Computational Cost

We only focus on the most compute intensive operations, being pairings (P), HashToPoint operations (HP), EC multiplications (EM), EC additions (EA), symmetric key encryptions/decryptions (S) and hashes (H). Table 1 compares the number of operations and the corresponding resulting time between our scheme and the ones of [3] for the key agreement scheme. Table 2 does the same for the secure communication phase. For the computation of the timings of both protocols, we have considered the numbers derived in [30], where all the operations have been evaluated on a personal computer with an Intel I5-3210M 2.5 GHz CPU (Intel, Santa Clara, CA, USA) and Windows 7 operating system (Microsoft, Redmond, DC, USA). The cryptographic operations have been implemented using the MIRACL cryptographic library. We have also assumed that there is only one stored session at the nodes and server side, similar to [3].

Table 1. Comparison of computational costs in the key agreement phase.

Key Agreement						
Entity	**This**		[3]		**Benchmark**	
Node	5H + 3EM + 1EA	2.967 µs	1HP + 2H + 2EM	16.269 µs	1H + 1EM + 1S	0.991 µs
Server	10H + 1EM + 2EA	1.004 µs	4HP + 2H	57.174 µs	2H + 2EM + 2S	1.982 µs

Table 2. Comparison of computational costs in the communication phase (Node 1 is the sending node and Node 2 is the receiving node).

Communication Phase				
Entity	**This**		[3]	
Node 1	1EM + 2H	0.988 µs	2HP + 2H + 2P	62.590 µs
Node 2	3EM + 1EA + 2H	2.96 µs	2HP + 2H + 2P	62.590µs

As can be concluded from the table, our proposed protocol is considerably faster from the node and the server side, both in the key agreement phase and in the communication phase. In particular, for the constrained devices of the nodes, our protocol outperforms approximately five times in the key agreement phase and between 21 and 63 times in the secure communication phase for the receiving and sending node, respectively. This follows from the fact that we do not need to compute the

intense hash-to-point operations in the key agreement phase and also the pairing operations in the communication phase. However, with respect to the Benchmark protocol, our key agreement scheme is approximately three times slower on the node side since we need to compute more intensive elliptic curve multiplications. This is the price to pay for being resistant to hacking.

7.2. Communication Cost

For the determination of the communication cost, we take into account that the nodes' identity and the timestamp correspond to 32 bits. The challenges are represented by 64 bits and the responses by 48 bits, as in [3]. An EC with $q = p = 160$ bits is used, which is also the length of the result of the hash value. As a consequence, the resulting sizes of the transmitted messages by both the node and the server in the key agreement protocol and by the sending node in the communication phase are enumerated in Table 3. It can be concluded that the communication cost for the secure communication is equal and our proposal is slightly better for the key agreement phase on the side of the node. However, compared to the benchmark key agreement scheme, our scheme still requires an approximately three times higher amount of bits to be transmitted.

Note that the proposed scheme and [3] have the same structure and thus the same number of exchanged messages, being four between Node 1 and Server and three between Node 2 and Server. These numbers are one lower in the case of the benchmark key agreement scheme.

Table 3. Comparison of communication costs in key agreement and communication phase.

Key Agreement						
Entity	**This (Bits/Bytes)**		**[3] (Bits/Bytes)**		**Benchmark (Bits/Bytes)**	
Node 1	416	52	704	88	256	32
Server	1120	140	864	108	224	28
total	1536	192	1568	196	480	60
Communication Phase						
Entity	**This (Bits/Bytes)**		**[3] (Bits/Bytes)**			
Node	480	60	480	60		

8. Conclusions

We have presented in this paper several important weaknesses of a recently proposed protocol for the authentication of IoT devices using PUFs [3]. The main lessons learned from the analysis are enumerated. A solution based on the ECQV for the PUF-based key agreement scheme has been presented. Also for the communication phase, we propose using a much more efficient existing EC signcryption scheme.

In addition, the proposed protocol is in all phases more efficient than [3], both from a communication and computation point of view. We also show that the performance of our PUF based key agreement protocol is reasonable, compared to a straightforward public key based approach.

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PUF	Physical Unclonable Function
IoT	Internet of Things
DoS	Denial of Service
ECC	Elliptic Curve Cryptography
EC	Elliptic Curve
ECQV	Elliptic Curve Qu Vanstone

References

1. Das, A.K.; Zeadally, S.; He, D. Taxonomy and analysis of security protocols for Internet of Things. *Future Gener. Comput. Syst.* **2018**, *89*, 110–125. [CrossRef]
2. Pappu, S.R. Physical One-Way Functions. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2001.
3. Chatterjee, U.; Chakraborty, R.S.; Mukhopadhyay, D. A PUF-Based Secure Communication Protocol for IoT. *ACM Trans. Embed. Comput. Syst.* **2017**, *16*, 67. [CrossRef]
4. Zheng, Y. Digital Signcryption or How to Achieve Cost (Signature & Encryption) << Cost (Signature) + Cost (Encryption). In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 165–179.
5. Braeken, A.; Shabisha, P.; Touhafi, A.; Steenhaut, K. Pairing free and implicit certificate based signcryption scheme with proxy re-encryption for secure cloud data storage. In *CloudTech*; IEEE: Rabat, Morocco, 2017.
6. Tashi, J.J. Comparative analysis of smart card authentication schemes. *IOSR J. Comput. Eng.* **2014**, *16*, 91–97.
7. Braeken, A. Efficient anonymous smart card based authentication scheme for multi-server architecture. *Int. J. Smart Home* **2015**, *9*, 177–184. [CrossRef]
8. Shamir, A. Identity-Based Cryptosystems and Signature Schemes. In *Workshop on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1984; Volume 196, pp. 47–53.
9. Al-Riyami, S.S.; Paterson, K.G. Certificateless Public Key Cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 452–473.
10. Gentry, C. Certificate-Based Encryption and the Certificate Revocation Problem. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 272–293.
11. Certicom Research. 2013. SEC4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme, Standards for Efficient Cryptography Group. Version 1.0. Retrieved 30 October 2017. Available online: http://www.secg.org/sec4-1.0.pdf (accessed on 10 July 2018).
12. Porambage, P.; Schmitt, C.; Kumar, P.; Gurtov, A.; Ylianttila, M. Two-phase Authentication Protocol for Wireless Sensor Networks in Distributed IoT Applications. In Proceedings of the 2014 IEEE Wireless Communications and Networking Conference (WCNC), Istanbul, Turkey, 6–9 April 2014; pp. 2728–2733.
13. Delvaux, J. Security Analysis of PUF-Based Key Generation and Entity Authentication. Ph.D. Thesis, Katholieke Universiteit Leuven (KULeuven), Leuven, Belgium, 2017.
14. Rostami, M.; Majzoobi, M.; Koushanfar, F.; Wallach, D.S.; Devadas, S. Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching. *IEEE Trans. Emerg. Top. Comput.* **2014**, *2*, 37–49. [CrossRef]
15. Yu, M.D.; Verbauwhede, I.; Devadas, S.; M'Raihi, D. A noise bifurcation architecture for linear additive physical functions. In Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Arlington, VA, USA, 6–7 May 2014; pp. 124–129.
16. Yu, M.D.; Hiller, M.; Delvaux, J.; Sowell, R.; Devadas, S.; Verbauwhede, I. A lockdown technique to prevent machine learning on PUFs for lightweight authentication. *IEEE Trans. Multiscale Comput. Syst.* **2016**, *2*, 146–159. [CrossRef]
17. Sadeghi, A.R.; Visconti, I.; Wachsmann, C. Enhancing RFID security and privacy by physically unclonable functions. In *Towards Hardware Intrinsic Security—Foundations and Practice*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 281–305.
18. Tuyls, P.; Batina, L. RFID-Tags for Anti-counterfeiting. In *Topics in Cryptology—CT-RSA 2006*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; Volume 3860.
19. Guartime and Intrinsic ID. Internet of Things Authentication: A Blockchain solution using SRAM Physical Unclonable Functions. 2017. Retrieved 30 October 2017. Available online: https://www.intrinsic-id.com/wp-content/uploads/2017/05/gt_KSI-PUF-web-1611.pdf (accessed on 10 July 2018).
20. Simplicio, M.A., Jr.; Silva, M.V.; Alves, R.C.; Shibata, T.K. Lightweight and escrow-less authenticated key agreement for the internet of things. *Comput. Commun.* **2017**, *98*, 43–51. [CrossRef]

21. Wouters, P.; Tschofenig, H.; Gilmore, J.; Weiler, S.; Kivinen, T. RFC 7250—Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS), June 2014. Available online: https://www.rfc-editor.org/rfc/rfc7250.txt (accessed on 10 July 2018).
22. Ha, D.A.; Nguyen, K.T.; Zao, J.K. Efficient authentication of resource-constrained IoT devices based on ECQV implicit certificates and datagram transport layer security protocol. In Proceedings of the Seventh Symposium on Information and Communication Technology, Ho Chi Minh City, Vietnam, 8–9 December 2016; pp. 173–179.
23. Gao, Y.; Ma, H.; Al-Sarawi, S.F.; Abbott, D.; Ranasinghe, D.C. PUF-FSM: A Controlled Strong PUF. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2017**. [CrossRef]
24. Hankerson, D.; Menezes, A.J.; Vanstone, S. *Guide to Elliptic Curve Cryptography*; Springer: New York, NY, USA, 2003; ISBN 038795273X.
25. SEC 2: Recommended Elliptic Curve Domain Parameters, Certicom Research, Standards for Efficient Cryptography Version 1.0, September 2000. Available online: http://www.secg.org/collateral/sec2final.pdf (accessed on 10 July 2018).
26. Recommended Elliptic Curves for Federal Government Use, National Institute of Standards and Technology, August 1999. Available online: http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf (accessed on 10 July 2018).
27. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
28. Lu, Y.; Li, J. Efficient Certificate-Based Signcryption Secure against Public Key Replacement Attacks and Insider Attacks. *Sci. World J.* **2014**. [CrossRef] [PubMed]
29. Le, M.-H.; Hwang, S.O. Certificate-Based Signcryption Scheme without Pairing: Directly Verifying Signcrypted Messages Using a Public Key. *ETRI J.* **2016**, *38*, 724–734. [CrossRef]
30. He, D.; Zeadally, S.; Wang, H.; Liu, Q. Lightweight Data Aggregation Scheme against Internal Attackers in Smart Grid Using Elliptic Curve Cryptography. *Wirel. Commun. Mob. Comput.* **2017**. [CrossRef]