International Journal of Computer Science and Software Engineering (IJCSSE), Volume 7, Issue 11, November 2018
ISSN (Online): 2409-4285 www.IJCSSE.org Page: 270-274

Attacking A PUF Based Mutual Authentication Protocol Designed For Internet of Things

Mehmet Hilal Özcanhan[1] and Semih Utku[2]

[1, 2] Computer Engineering, Dokuz Eylul Universitesi, Faculty of Engineering, Izmir, 35390, Turkiye

[1]hozcanhan@cs.deu.edu.tr, [2]semih@cs.deu.edu.tr

ABSTRACT

The security of the message exchanges among Internet of Things (IoT) devices is a big concern, in the field of information security and privacy. Due to the limited resources of IoT devices and their unsecure environments, strong authentication protocols are needed for the machine to machine (m2m) conversations. The physically unclonable functions (PUFs) inherent in the integrated circuits of the IoT devices are seen as an answer, for the security and privacy of the m2m communication. Although PUFs have strong advantages, the authentication protocols can show weaknesses, if not well-designed. In present work a mutual authentication protocol proposed for IoT devices is analyzed. It is demonstrated that with the same attack model of the authors, multiple attacks can be launched on the proposed protocol. A recommendation is also made for providing stronger security to the proposed protocol.

Keywords: *Authentication, Disclosure Attack, Internet of Things, Machine to Machine, Security.*

1. INTRODUCTION

Internet of Things (IoT) is the present day's buzz word. Simply, IoT are devices which enable machine to machine: communication, information sharing and taking action over the Internet. Without active intervention, the humans gain the capability of making better decisions and taking faster action, through IoT. Latest reports disagree in the range of billions about the number of IoT devices (devices) that will be connected to the Internet, by the year 2020. Figure 1 shows the m2m/IoT, Sector Map [1]. As it can be seen, there are numerous types of devices at every possible location, in all sectors. Some devices, such as low cost tags in the retail sector, contain 8 or 16 bit microcontrollers and have very limited memory. In fact, there are many types of devices with limited resources [2, 3]. This characteristic brings about a serious handicap to the security of IoT communication; because resources are needed for providing security, while authentication or sharing of data. Furthermore, devices are expected not to leak any secrets, even if captured. Therefore, it is no surprise that bridging the gap between limited device resources and secure communication is the subject of many research work [2].

In present work, we show that providing security is not an easy task, by demonstrating the weaknesses of an authentication protocol presented for IoT communication [3]. The analyzed work is based on the characteristic of physically unclonable functions (PUF) inherent in devices. The PUF characteristic used by the proposals as a security primitive is the variations in the nano-structure of the integrated circuits, PUFs result in a unique one-way functions that produce random outputs, specific to the nano-structure production of each integrated circuit [4]. Since the building blocks of the IoT microcontrollers are integrated circuits, each device is equipped with at least one unique, unclonable PUF. The authentication proposals exploit the property by encrypting exchanged messages using the unique PUF; and not storing any secrets on the device. PUFs that do not require heavy computation or large memory resources are named as lightweight PUFs [5]. Being lightweight, PUFs are seen as ideal for the resource constraint IoT applications. But, PUFs are divided into strong PUFs and weak PUFs in some works [6]. Certain PUFs that considered to be strong in the past are no longer shielded from the off-line attacks, by high end mobile computers. Therefore, authentication protocols relying on PUFs can still be shown to possess weaknesses that can be exploited, as we show in the next section.

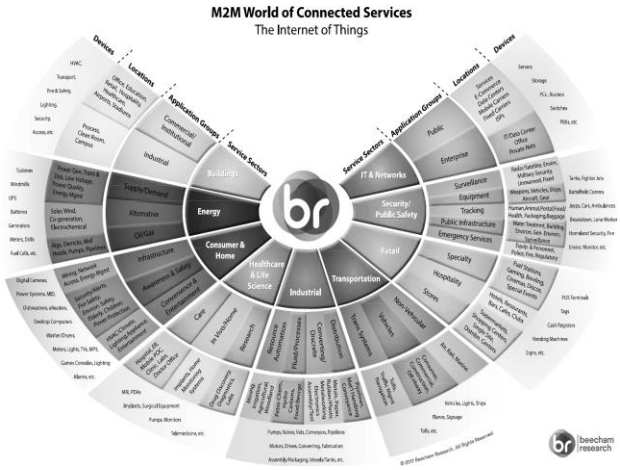

Fig. 1 Beecham Research's M2M/IoT Sector Map© [1].

2. PROPOSED ANALYSIS

The authentication protocol analyzed is shown in Figure 2. Due to the first name letters of its authors, the analyzed protocol will be named as MKB protocol, for simplifying explanations. Concisely, MKB protocol proceeds and ends as follows: The device reporting data has a unique constant identification number $\mathbf{IoT_{ID}}$. Similarly, the server has $\mathbf{S_{ID}}$. To initiate a new session of communication the device sends its IoT_{ID} and a fresh nonce $\mathbf{nonce^i_{IoT}}$. The authentication session number is indicated as a superscript over the parameters, indicating that the value changes every session.

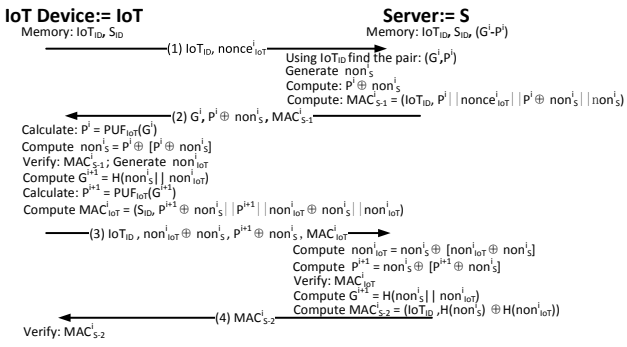

Fig. 2 The analyzed MKB protocol [3].

Using the device IoT_{ID}, the server finds two values $\mathbf{G^i}$ and $\mathbf{P^i}$. Parameter P^i is a value computed by the device using its unique $\mathbf{PUF_{IoT}}$. The server never calculates P^i. G^i is a value calculated by both sides through a hash function H(). G^i is the seed value for the PUF uniqueness verification, in the protocol. After generating its own nonce $\mathbf{non^i_S}$, the server hides P^i in ($\mathbf{P^i}$ $\mathbf{non^i_S}$). Finally, the server computes a message authentication code (MAC) $\mathbf{MAC^i_{S-1}}$ by using a standard, public MAC function to provide integrity and freshness to the message sent in step 2. Using the G^i value supplied by the server, the device calculates the corresponding P^i value, by the help of its unique PUF_{IoT}. XORing the computed P^i value with received ($P^i \oplus non^i_S$) value, the device obtains the non^i_S of the server. Using the same public MAC function, the device calculates a MAC^i_{S-1} value and compares it with the received MAC^i_{S-1} value. If the two values match, the device verifies the integrity of the received message. Then, it generates its own nonce $\mathbf{non^i_{IoT}}$ to obtain a new seed value for the next session's $\mathbf{G^{i+1}}$, by hashing the concatenation of the two nonces of either side, through a standard public hash function **H()**. Hash functions are one way functions that produce a unique output for a unique input [7]. Although it is easy to compute the output of a given input; it is hard to reverse the function to obtain the input, given an output [9]. In the next two calculations, the device calculates the device specific $\mathbf{P^{i+1}}$ value and a $\mathbf{MAC^i_{IoT}}$. Notice the S_{ID} of the server in the calculation of the MAC^i_{IoT} value. For the challenge of step 2, the device sends its response message in step 3. The server obtains the non^i_{IoT} through XORing its own non^i_S with the received ($\mathbf{non^i_{IoT}} \oplus \mathbf{non^i_S}$) value. With a similar non^i_S XOR operation, the P^{i+1} value is obtained from the received $P^{i+1} \oplus non^i_S$ value. Using the above obtained two values, the server calculates and matches received MAC^i_{IoT} to verify the integrity of message 3 and the identity of the device. In the next two steps, the server computes the G^{i+1} value and its second $\mathbf{MAC^i_{S-2}}$. To prove its identity to the device, the server sends MAC^i_{S-2}, in step 4. After receiving the MAC^i_{S-2} value, the device checks and verifies the identity of the server, proving that the opposite side has successfully isolated its nonce value and has the secret $\mathbf{S_{ID}}$ to verify its message 3. That ends the mutual authentication process and data transfer can start. In the rest of work [3], after a series of proofs the MKB protocol is declared secure.

2.1 Attack Model and Tool Used

The attack model: In order to work under fair conditions, we assume the same attack model of the MKB protocol. Namely, the devices are deployed in the open, physically unprotected. Therefore, they are easily susceptible to tampering or cloning attacks. Furthermore it is accepted that an adversary can eavesdrop on the communication channel, capture and save messages exchanged, inject packets, initiate a session, replay old messages or mimic one of the communicating partners. The aim of the adversary is to launch an undetectable attack to authenticate itself with the server or any of the IoT devices.

The attack tool used: The method of our attack is the exploitation of rainbow tables prepared for public hash functions [7]. Rainbow tables are created by offline hashing of every possible input and storing the results in a look-up table, to be used later in attacking authentication protocols. By finding the output value of a hash operation in the rainbow table and then reading the corresponding input value, exposes the input value of hash function in the protocol analyzed. For an input of ι bits long, the rainbow table length is 2ι. Hardware implementations using high-end Field Programmable Gate Arrays has been proposed for fast creation of rainbow tables, for hash functions such as MD5 and SHA-1 [8].

2.1 The Analysis of MKB Protocol

As in the attack model an adversary eavesdrops and saves the messages exchanged for two consecutive i^{th} and $i+1^{th}$ sessions. From Figure 2, the recorded messages in order:

(1) $\mathbf{IoT_{ID}}$ (7) $\mathbf{P^{i+1}}$ $\mathbf{non^i_S}$ (12) $\mathbf{P^{i+1}}$ $\mathbf{non^{i+1}_S}$
(2) $\mathbf{nonce^i_{IoT}}$ (8) $\mathbf{MAC^i_{IoT}}$ (13)$\mathbf{MAC^{i+1}_{S-1}}$
(3) $\mathbf{G^i}$ (9) $\mathbf{MAC^i_{S-2}}$ (14)$\mathbf{non^{i+1}_{IoT}}$ $\mathbf{non^{i+1}_S}$
(4) $\mathbf{P^i}$ $\mathbf{non^i_S}$ (10) $\mathbf{nonce^{i+1}_{IoT}}$ (15)$\mathbf{P^{i+2}}$ $\mathbf{non^{i+1}_S}$
(5) $\mathbf{MAC^i_{S-1}}$ (11) $\mathbf{G^{i+1}}$ (16) $\mathbf{MAC^{i+1}_{IoT}}$
(6) $\mathbf{non^i_{Io}}$ $\mathbf{non^i_S}$ (17) $\mathbf{MAC^i_{S-2}}$

While capturing the exchanges, the saved exchanges can be searched for an instance when the value of the XOR ($\mathbf{non^i_{IoT}} \oplus \mathbf{non^i_S}$) in (6) is equal to one of the values in the set {all zeros (00….000), or all ones (FF….FFF), or a value with only a single bit with value "1" (i.e. 000…001 or 0010..000 etc.); or a value with only a single bit with value "0" (i.e. 111…110 or 1101..111 etc.)}. Capturing exchanges and analyzing saved sessions can be easily carried out in parallel, with a high end notebook. For simplicity, assume the instance of all zeros values for (6). Such a value indicates that $\mathbf{non^i_{IoT} = non^i_S}$, due to the property of XOR operation. The $\mathbf{G^{i+1}}$ value is available in cleartext, during the next session at (11). Searching through a pre-calculated rainbow table of length 2ι (ι:= length of non^i_S) such as one shown in Table 1, the row entry where the $\mathbf{G^{i+1}}$ value is found, exposes non^i_S.

Table 1: A typical rainbow table for the hash function H() of Figure 2

$\mathbf{non^i_S}$	$\mathbf{G^{i+1}=H(non^i_S \| non^i_S)}$
0000……..0000	……..
0000……..0000	……..
$\mathbf{non^j_S}$	$\mathbf{G^{j+1}}$
……..	……..
1111……..1111	……..

Using the recorded values, the attack proceeds as follows:

Since $\mathbf{non^i_{IoT} = non^i_S}$, $\mathbf{non^i_{IoT}}$ is captured. (18)

XORing the captured $\mathbf{non^i_S}$ with (4), $\mathbf{P^i}$ is exposed. (19)

XORing the captured $\mathbf{non^i_S}$ with (7), $\mathbf{P^{i+1}}$ is exposed. (20)

XORing $\mathbf{P^{i+1}}$ (20) with (12) $\mathbf{non^{i+1}_S}$ is exposed. (21)

XORing $\mathbf{non^{i+1}_S}$ (21) with (14) $\mathbf{non^{i+1}_{IoT}}$ is exposed. (22)

XORing $\mathbf{non^{i+1}_S}$ (21) with (15) $\mathbf{P^{i+2}}$ is exposed. (23)

Using $\mathbf{MAC^i_{S-1}}$ and $\mathbf{MAC^i_{S-2}}$ all captured values are verified. After the verification of the values, $\mathbf{G^{i+2}}$ is calculated using $\mathbf{G^{i+2}=H(non^{i+1}_S || non^{i+1}_{IoT})}$. The adversary now has consecutive (G^i, P^i), (G^{i+1}, P^{i+1}), (G^{i+2}, P^{i+2}) pairs, non^i_S, non^i_{IoT}, non^{i+1}_S, non^{i+1}_{IoT}. The only unknown left is S_{ID}, at this stage.

For simplicity of step explanations, the instance of all zeros values for ($non^i_{IoT} \oplus non^i_S$) was assumed. But this is not the only case where non^i_{IoT} is exposed. For the case when Eq. (6) is equal to { all ones (FF….FFF)}, $non^i_{IoT} = \neg non^i_S$ (inverse of non^i_S), or vice versa. The attack proceeds as before. A value is derived for non^i_{IoT}, from the rainbow table of Table 1. The value of non^i_{IoT} and non^i_S are verified by the value that matches the constructed MAC^i_{S-1} and the received MAC^i_{S-1}. For the case when Eq. (6) is equal to {a value with only a single bit with value "1"; or a value with only a single bit with value "0"}, the analysis is just a few easy steps longer. The case indicates that non^i_{IoT} is different from non^i_S only at the single "mismatches" bit, by the property of $\oplus$ operation. As in the previous cases, the derived value of non^i_{IoT} from Table 1is accepted as true. For the value of non^i_S, the "mismatched" bit value is flipped. The values are tested using MAC^i_{S-1}. If they are not verified, the "mismatched" bits of both non^i_{IoT} and non^i_S are flipped. This time MAC^i_{S-1} match verification is obtained and the analysis ends. Therefore, there are 4 cases when the analysis produces exposed values. In fact, the attack can be extended over to two "mismatched" bits, but we will not pursue multiple bit analysis as the point is made. But, it is obvious that the public hash functions short rainbow table (256 for ι =8 and 65536 for ι =16) is definitely a weakness.

2.1 Successful Attacks on MKB Protocol

Forward secrecy: Since all successive (G^i, P^i), (G^{i+1}, P^{i+1}), (G^{i+2}, P^{i+2}) pairs have been exposed, the adversary can decode the rest of the future authentications. In other words forward secrecy of the communications is

definitely breached. But at this stage only forward secrecy is not at stake.

Session Hijacking: Consider the absence of the server or its deliberate blocking from receiving IoT device messages. Having captured the secrets above, a clandestine server generates a fake nonce non^{i+2}_{Sfake} and plays it with the easily calculated $P^{i+2} \oplus non^{i+2}_{Sfake}$ and $MAC^{i+2}_{S\text{-}1fake}$ = $(IoT_{ID}, P^{i+2}||nonce^{i+2}_{IoT}||P^{i+2} \oplus non^{i+2}_{Sfake}||non^{i+2}_{Sfake})$, at step 2 of Figure 2. The device takes the value of G^{i+2} and calculates P^{i+2} to extract the non^{i+2}_{Sfake}. Then, using the two values the device verifies $MAC^{i+2}_{S\text{-}1fake}$, without suspecting any foul play. Next the device generates non^{i+3}_{IoT} to calculate the next session's G^{i+3}, P^{i+3} values and its own MAC^{i+2}_{IoT}. After receiving message 3, the clandestine server exposes non^{i+2}_{IoT} and P^{i+3}, easily. Next, verification of MAC^{i+2}_{IoT} is simply skipped. Then, G^{i+3} and $MAC^{i+2}_{S\text{-}2fake}$ are calculated. $MAC^{i+2}_{S\text{-}2fake}$ is sent in message 4 which is verified by the device, without any complaint. Thus a full session is hijacked and the adversary has successfully mimicked a server.

In fact, any valid (G^k, P^k) pair can be replayed with a fake non^k_S to the device because, the device does not have a memory of the last session and simply recalculates P^k. Once a fake $nonk^i_S$ is forged into the device, the attack succeeds as it leads to the capture of the device non^k_{IoT}. Session hijack can continue, as long as the server does not reply message 1 of the device. Now, assume the authentic server comes alive and responds the device with an old (G^i, P^i) pair. The protocol runs flawlessly and the server detects no past malicious activity. Hence, the aim of the adversary to launch an undetectable attack to authenticate itself with the device has succeeded.

Full Disclosure attack: If the attack model of possible tampering is accepted, the MKB protocol has one last big weakness. The authors claim that no secret is kept at the IoT device memory. But, on the contrary a parameter S_{ID} is silently used in the protocol's MAC^i_{IoT} message. In order to use S_{ID} in its every session calculations, the device has to have it in its memory. According to the attack model, this value can be captured through tampering with the device. But, another way of exposing the S_{ID} is analyzing the MAC^i_{IoT} value. If the public MAC function used is similar to the used hash function H(),a similar rainbow table analysis can expose the S_{ID} value; since it is the only unknown in the Eq. of MAC^i_{IoT}. Even a table with an entry space of 2^{46} rows can be searched by an off-the-shelf CPU, within a couple of hours [5]. The disclosure of S_{ID} is a disastrous development, since it opens the way to device mimicking or cloning. This is against the original claim of MKB authors that their IoT device provides security, even if captured by an adversary.

3. DISCUSSION

3.1 Attack Success Probability

It has been demonstrated that MBK protocol proposed for IoT information transfer can be successfully attacked. But, the probability of the success of the attack in a real IoT environment has to be provided. Therefore, the probability of our attack's success is calculated, next. The set of values necessary for our analysis has been given, in the above section. There may be **m** devices that report to the server, **r** times per hour (**r** sessions/hour). The bit length of popular low cost IoT device microcontrollers is either 8 or 16. Thus, generated nonce (Figure 2) length ι = 8 or 16. Using these parameters:

Probability of Eq. (6) to be all zeros is $P1 = 1/2^\iota$

Probability of Eq. (6) to be all ones is $P2 = 1/2^\iota$

Probability of a single bit of (6) to be 1 is $P3 = \iota /2^\iota$

Probability of a single bit of (6) to be 0 is $P4 = \iota /2^\iota$

Total Probability:

$$P_T = P1 + P2 + P3 + P4 = (2\iota + 2)/2^\iota \quad (24)$$

For **m** devices present in the environment:

$$P_{mT} = [m \times (2\iota + 2)/2^\iota] \quad (25)$$

For m devices with **r**/hour:

$$P_{mTr} = [r \times m \times (2\iota + 2)/2^\iota] \quad (26)$$

For probability $P_{mTr} = 1$ (a successful attack), r=1/hour:

For ι =8 bits, $m = \frac{256}{18} = 14$; (every 14 device authentications a device is exposed/hour).

For ι =16 bits, $= \frac{65536}{3400}$; every 1927 device authentications a device is exposed/hour.

For r = 100 sessions/hour, ι =16 bits:

$m = \frac{65536}{3400}$; every 19 device authentications a device is exposed/hour.

Every week, 168 (24 hours x 7 days) devices are guaranteed to be exposed, totaling to 5040 in a month. Therefore, it is obvious that with the use of the MKB protocol, a complete IoT environment of approximately 5000 devices can be breached, within one month.

3.2 Cracking Unique PUFs of IoT Devices

Another important weakness of lightweight PUFs can be the limited input space [5]. As proven is our attack, consecutive (G^k, P^k) pairs are exposed. The exposed pairs can be saved in a table, for later look-up. Such a table opens up the way of solving a PUF, numerically. If the table is short, looking at the G^k value sent in cleartext in step 2, the corresponding value of P^k

becomes trivial. Then, exposing non^k_S from the cleartext P^k non^k_S sent in step 2, becomes easy. Low cost devices like Radio Frequency Identification (RFID) tags are also in the IoT device category. And their microcontroller architecture is mostly 8 or 16 bits. Therefore, the nonces used in the authentication protocols are not adequately long for strong security. This is depicted in the short length of the rainbow table for the used public hash function in the analyzed MKB protocol. Instead, an elliptic curve cryptography supported PUF protocols can provide better resistance to our attack [10]

4. CONCLUSIONS

Multiple known attacks on a mutual authentication protocol designed for IoT devices have been demonstrated. The attack can expose thousands of IoT devices in a system, within a month. The existence of rainbow tables for hash functions has to be accepted as a reality. As such, authentication protocols using 8 or 16 bit length message-exchange words should consider using strong cryptographic encryption functions.

REFERENCES

[1] Beecham Research's M2M/IoT Sector Map©, http://www.beechamresearch.com/ download.aspx?id=18, (Accessed on 29 th October 2018).
[2] D. Miorandi, S. Sicari, F. Pellegrini, and I. Chlamtac, "Internet of Things: Vision, Applications and Research Challenges", Ad Hoc Networks, Vol. 10, No. 7, 2012, pp.1497-1516.
[3] A. M. Naveed, K. C. Chua and B. Sikdar, "Physically secure mutual authentication for IoT", in Conference on Dependable and Secure Computing, IEEE, 2017.
[4] A. Cherkaoui, L. Bossuet, L. Seitz, G. Selander and R. Borgaonkar, "New paradigms for access control in constrained environments", in International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip, 2014, IEEE, 2014.
[5] D. Mukhopadhyay, "PUFs as promising tools for security in Internet of things", IEEE Design & Test, Vol. 33, No. 3, 2016, pp. 103-115.
[6] R. Maes, "Physically Unclonable Functions: Constructions", Properties and Applications, New York USA: Springer-Verlag, 2013, pp. 1–172.
[7] M. H. Özcanhan, "Analysis of a Recent Hash Based RFID Authentication Protocol Intended for Telecare Medicine", International Research Journal of Engineering and Technology, Vol. 2, No. 4, 2015, pp, 1520-1524.
[8] K. Theocharoulis, I. Papaefstathiou and C. Manifavas, "Implementing rainbow tables in high-end FPGAs for super-fast password cracking", International Conference on Field Programmable Logic and Applications, IEEE, 2010.
[9] M. H. Özcanhan, "Improvement of a Weak RFID Authentication Protocol Making Drug Administration Insecure", Life Science Journal, Vol. 11, No. 10, 2014, pp. 269-276.
[10] J. R. Wallrabenstein, "Practical and secure IoT device authentication using physical unclonable functions", in International Conference on Future Internet of Things and Cloud, 2016, IEEE.