International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 02 Issue: 04 | Jul-2015 www.irjet.net p-ISSN: 2395-0072

# Analysis of a Recent Hash Based RFID Authentication Protocol Intended for Telecare Medicine

Mehmet Hilal ÖZCANHAN[1]

[1] *Asst. Professor, Computer Engineering, Dokuz Eylul University, Izmir, Turkey*

---

Abstract - *Radio Frequency Identification (RFID) is technologically one of the most popular batch identification methods of our modern age. The low cost RFID stickers (tags) are gradually replacing traditional paper barcodes. However, the use of tags comes at a an additional expense in terms of die area and clock cycle resources needed for secure operation, through open air. Contrarily, due to the cost competition with traditional barcodes, limited resources can be spared for security. Therefore, guaranteeing the integrity and confidentiality of the transmitted information is a hard challenge, in RFID technology. Since the cryptographic algorithms used in computers cannot fit in tags, other strong encryption functions are needed to protect the exchanged information. This is the main reason behind the* ***RFID security community's*** *convergence on alternative function proposals for obscuring the exchanged information. Unfortunately, many of the proposed schemes have been demonstrated to show security weaknesses. One latest work relies on the strength of hash functions, to recommend an authentication protocol for the mission critical telecare medicine. Naturally, any breach in the security of the proposed protocol may mean a health or privacy risk for its users. The present work uses a two stage analysis to test the proposed protocol. An algebraic analysis is supported by the rainbow tables approach to expose the secrets of the protocol. The security evaluation demonstrates multiple security weaknesses in the* ***protocol's*** *design. The weaknesses prove to be so grave that the proposed authentication protocol may be classified as insecure and unsafe for its telecare users.*

*Key Words: Hash function, mutual authentication, rainbow tables, RFID security, traceability.*

## 1. INTRODUCTION

Paper barcodes have long been used for identifying commercial goods, in supply chains. But the wear and tear of paper led to the more expensive plastic fortified stickers. Furthermore, the new two-dimensional barcodes require precise printing. The two above factors have hindered reduction in the cost of the barcodes. Additionally, the barcode stickers have to be read one by one; clearly a time consuming and expensive operation. On the other hand, there are the RFID tags. Basically, a tag is a tiny size microcontroller with reduced memory and die area [1, 2]. An antenna coil is attached to the microcontroller for communication and electromagnetic energy transfer. In other words, a reader energizes and requests the unique identification number (ID or Electronic Product Code (EPC)), which resides inside the memory of the tag. The miniaturized electronics makes reading as many as 1000 tags/sec possible [2]. The operation distance and the batch reading of tags promote obvious advantages over barcodes. For detailed electronics and properties of RFID tags, the reader is referred to works [1, 2].

In the rest of this paper, Section 2 summarizes Related Works. Section 3 presents the analyzed proposal's authentication scheme. The exposure of the tag secrets is demonstrated in Section 4. In Section 5, the security analysis of the studied authentication is made and multiple vulnerabilities are shown. There is Conclusion in Section 6.

## 2. RELATED WORKS

The RFID authentication protocols are often classified according to the computational cost of the functions they support [3]. There are four classes named as "fully fledged, simple, lightweight and ultra-lightweight". At the lowest level, ultra-lightweight protocols support only bitwise operations like AND, OR, XOR, shift and modulo 2 addition. Lightweight protocols support random number generators and simple functions like cyclic redundancy code (CRC) check. Simple class protocols go further and support hash functions, in addition to random number generators. At the top of the classification, the fully-fledged protocols support the conventional cryptographic functions like symmetric encryption and public key algorithms. Examples of each class are given in work [3], but a regularly updated list can be reached at www.avoine.net/rfid/index.php. The focus of the present work is on a protocol in the simple class, which relies on the strength of the hash functions [4].

 

Hash functions h( ) are defined as one-way. Obscuring a given secret *x*, by calculating its hash value in $M = h(x)$ is easy. But, calculating the reverse function $x = h^{-1}(M)$ is hard. Therefore, exposing *x* by calculation is accepted as mathematically very difficult. Another property of the hash functions is that it is hard to find a value *x'*, where $h(x) = h(x')$. The above two properties guarantee that the hash functions produce no collision in their outputs, i.e. a unique input always produces a unique output. This favorable characteristic can turn into a disadvantage, as it will be revealed in Section 4. MD5 [5], SHA-1 [6] and the latest winner of NIST hash function contest Keccak [7] are popular examples for hash functions.

## 3. THE ANALYZED AUTHENTICATION SCHEME

The scheme of the analyzed authentication scheme of work [4] and the used notation are given in Figure 1. For simple reference, the scheme is named SAKM after its authors' names. The SAKM authentication protocol proceeds as follows. Initially, the server database contains the present and last session's secret keys ($SK_s$, $SK_{s-1}$), the unique identification number ($ID_k$) and the hash values $h(SK_s || ID_k)$, $h(SK_{s-1} || ID_k)$ of every tag. The initial value of the secret key $SK_s$ of every tag is zero. On the other hand, each tag contains its unique identification number $ID_k$ and the pre-shared session secret key $SK_s$. When the reader wants to acquire the $ID_k$ of a tag, it generates and sends a random number $R_r$. Upon receiving the request, the tag generates its own random number $R_t$ and tries to obscure its $ID_k$ before sending it to the reader, by computing A, B, and C as given in Figure 1. Next, the tag sends A, C and its timestamp $T_1$ used in the calculations, to the reader. The reader appends its random number and sends message A, C, $T_1$, $R_r$ to the server.

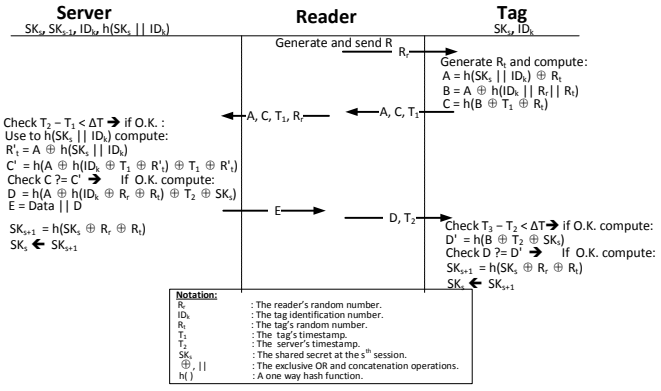


Fig -1: The authentication scheme of work [4].

Before going into the procedure of authenticating the tag, the server checks the tag's reply if it is within a legitimate time delay. Then, using message A and the database hash value $h(SK_s || ID_k)$ of each tag, the server computes a value R't. Using the calculated R't value the server computes a value C' and matches it with the received value C. A correct match identifies the tag and the corresponding $ID_k$ in the database can be used to compute the reply D, necessary to authenticate the server. The server computes D given in Figure 1, prepends the data of the tag to form E and sends E to the reader. After sending E to the reader, without waiting for an acknowledgment of the receipt of D, the server updates $SK_s$ with $SK_{s+1}$ and relegates used $SK_s$ to $SK_{s-1}$. In other words, the server finishes by losing the original $SK_{s-1}$, whether D reaches the tag or not.

The reader extracts the pre-fix "Data" and transmits message D to the tag. The tag first verifies that the response of the server is within a legitimate time delay. But to do so, the tag needs the timestamp $T_2$ of the server. This is strangely forgotten in SAKM, but rightfully added in Figure 1. $T_2$ is also needed in computing D'. If D and D' match, then the server is authenticated. As a last step, the tag updates the session secret key like the server, but drops the old session secret key. Hence, the mutual authentication is completed.

## 4. ANALYSING THE SAKM PROTOCOL

The SAKM protocol contains three message exchanges through air. During these exchanges $R_r$, A, C, $T_1$, D and $T_2$ are transmitted. Therefore, any adversary listening to the exchanges can store the messages for offline analysis. In fact, in a standard adversarial model the adversary can [8]:

- Query: Interrogate tags in the system.
- Send: Act as a tag in the system.
- Execute: Actively monitor the air channel.
- Block: Prevent a message reaching the intended receiver.

Using the above model, the following analysis of the SAKM protocol is performed.

### 4.1 De-synchronization Attack on SAKM Protocol

In the SAKM protocol, the server prepares its reply D and sends it inside message E, as shown in Figure 1. The reader keeps the "Data" and sends only D and $T_2$ to the tag. The tag needs this information to authenticate the server and update the secret key. Consider the case when message D fails to reach the tag; due to intentionally blocking of D, or not sending it at all. The server has already updated the session secret key $SK_s$ to a new value $SK_{s+1}$, changed $SK_{s-1}$ into the used $SK_s$ and dropped the previous $SK_{s-1}$. However, the tag still preserves the original $SK_s$. In the next authentication session the server identifies the tag using the $SK_{s-1}$ value in its database, which equates to the $SK_s$ value in the tag. But, when message D is blocked for a second consecutive time, the server goes through another round of update and drops the tag's $SK_s$ value (recorded as $SK_{s-1}$ value in its database). After this point, the secret key in the tag's memory does not match any of the updated secret keys in the server database. Hence, blocking the server's reply two consecutive sessions from reaching the tag, the server is persuaded not to recognize a perfectly legitimate tag. The user of the tag is wrongly eliminated from normal telecare medicine operations.

 

### 4.2 Full-Disclosure Attack on SAKM

The tag's secrets are the session key $SK_s$ and the tag $ID_k$. Any disclosure of these secrets may turn the sessions into public exchanges, which in turn can have devastating results to the tag's owner. In this section, an offline analysis for disclosing the secrets of the tag, consisting of two stages is demonstrated. The first stage is an algebraic analysis of the eavesdropped messages. In the second stage, the well-known rainbow table search method used in attacking hash functions is applied to the outcome of the first stage.

To start, consider the first session of brand new tags, when the initial value of $SK_s$ is zero. With a rogue reader, the tags can be challenged with a fabricated random number $R_r$, for an unlimited number of sessions [1, 2, 8]. In our analysis, the SAKM tag is challenged twice by playing $R_r$ = 0 , but response D is given only in the second session. The replies as $A^1$, $C^1$, $T^1{}_1$, $A^2$, $C^2$, $T^2{}_1$ are recorded, where the superscripts notate the session number. Notice that the shared session secret $SK_s$ remains the same, as it is not updated as long as message D is blocked. After two sessions, the adversary has the following four equations:

$$A^1 = h(0 \,||\, ID_k) \oplus R^1{}_t \quad (1)$$
$$C^1 = h\{h(0 \,||\, ID_k) \oplus h(ID_k \,||\, 0 \,||\, R^1{}_t) \oplus T^1{}_1\} \quad (2)$$
$$A^2 = h(0 \,||\, ID_k) \oplus R^2{}_t \quad (3)$$
$$C^2 = h\{h(0 \,||\, ID_k) \oplus h(ID_k \,||\, 0 \,||\, R^2{}_t) \oplus T^2{}_1\} \quad (4)$$

Observe that $R^1{}_t \oplus R^1{}_t$, $R^2{}_t \oplus R^2{}_t$ canceled out in equations (2) and (4), respectively. XORing (1) with (3) yields:

$$A^1 \oplus A^2 = R^1{}_t \oplus R^2{}_t \quad (5)$$

The above strategy of challenging with $R_r$ = 0 and gathering an infinite set of equations (1 – 5) with new values is possible [8]. Hence, it is possible to obtain interesting values for equation (5) using any two sessions y, z such that $R^y{}_t \oplus R^z{}_t = 2^n$, or zero, or all ones ($FFFF_H$). The first case means that $R^y{}_t$ differs from $R^z{}_t$ only at the $n^{th}$ bit location. In the second case, a zero result means $R^y{}_t = R^z{}_t$ . In the third case, $R^y{}_t = \neg R^z{}_t$. In other words, if $R^y{}_t$ is captured, the exposure of $R^z{}_t$ becomes trivial.

Taking equation (2), the analysis continues using the popular "rainbow tables" technique. The rainbow tables technique is a known attack aiming at finding the inverse of one-way hash functions. It is known that computing the inverse of a hash operation $M = h(x)$ to find $x = h^{-1}(M)$ is hard. But if, for every $x$ the corresponding $M$ is computed and recorded in a look-up table, then finding the value of $x$ for a given $M$ reduces to a single look-up in the prepared table. Such look-up tables prepared for obtaining the input of a hash function is named as rainbow tables [9]. It requires $2^n$ computations for preparing a full input-output rainbow table for a given hash function, where n is the bit length of $x$. Obviously, the rainbow table search method is simpler than the brute force attack, where the computation $M = h(x)$ is performed until the correct $M$ is reached. The trade-off in time, power and memory space needed for preparing rainbow tables is a hot topic in the hash function analysis community [9, 10]. Present work's scope is to use the fast and spacious Field Programmable Gate Arrays (FPGA) of work [11], because solving equations (2) and (4) makes the capture of the secrets of SAKM tags inevitable.

To prove our argument, let us consider the case when the value of $C^1$ of equation (2) is in a table similar to those presented in works [1, 2, 9]. After a single look-up, the corresponding input value "$h(0 \,||\, ID_k) \oplus h(ID_k \,||\, 0 \,||\, R^1{}_t) \oplus T^1{}_1$" is obtained. $T^1{}_1$ was transmitted in cleartext and recorded. XORing the obtained input with $T^1{}_1$ yields the value "$h(0 \,||\, ID_k) \oplus h(ID_k \,||\, 0 \,||\, R^1{}_t)$". Following a similar argument for $C^2$ of equation (2), the value "$h(0 \,||\, ID_k) \oplus h(ID_k \,||\, 0 \,||\, R^2{}_t)$" is obtained. XORing the two obtained values produces a new result $R = h(ID_k \,||\, 0 \,||\, R^1{}_t) \oplus h(ID_k \,||\, 0 \,||\, R^2{}_t)$, because "$h(0 \,||\, ID_k) \oplus h(0 \,||\, ID_k)$" cancels out. However, $R$ is the result of the XOR of two values from the output column of the rainbow table. $R$'s most significant bit inputs ($ID_k$) are identical, the middle bit inputs are zero and only the least significant bit inputs ($R^1{}_t$ and $R^2{}_t$) differ. This condition reduces the space to be searched by one third of the full rainbow table, as shown in Figure 2. As an example, for a 6 bit length input (n=6) the number of entries to be searched is $2^4$, instead of $2^6$.

| Input = x (n bits) | Output = h(x) |
|---|---|
| 0 .. .. .. 0 | *output 1* |
| 0 all zeros 0 0 ... 0 | ... |
| 0 ... ... | $h(ID_k \,||\, 0 \,||\, R^1{}_t)$ |
| 0 ... | |
| 0 | ... |
| 0 all zeros 0 1 1 | $h(ID_k \,||\, 0 \,||\, R^2{}_t)$ |
| .. .. .. .. .. .. | |
| NOT all zeros 0 | |
| C all zeros 0 0 ... 0 | |
| C ... ... | |
| C ... | |
| 1 C 1 all zeros 0 1 1 | |
| .. .. .. .. .. .. | |
| NOT all zeros 0 | |
| F all zeros 0 0 ... 0 | |
| F ... ... | |
| F ... | |
| 1 F 1 all zeros 0 1 1 | |
| .. .. .. .. .. .. | |
| NOT all zeros 0 | |
| .. .. .. .. .. .. | |
| 1 1 .. .. .. 1 1 | *output* $2^n$ |

Fig -2: The search space of our attack's rainbow table.

The search for the inputs is not complicated due to the property of the hash functions explained in Section 2: Since every entry in the output column of the rainbow table is unique, each of the two values $h(ID_k \| 0 \| R^1_t)$ and $h(ID_k \| 0 \| R^2_t)$ is also unique. Therefore, there are exactly two values that produce the final result *R*. By preparing a second table of R values using the first tables $2^{2n/3}$ number of inputs, the search for *R* is still kept at a single look-up. The inputs are XORed to get the corresponding R values and the results are tabulated, after noting down the XORed rows. Hence, finding the value of *R* reveals the values of the values $h(ID_k \| 0 \| R^1_t)$ and $h(ID_k \| 0 \| R^2_t)$, as well as the values of the inputs "$ID_k \| 0 \| R^1_t$" and "$ID_k \| 0 \| R^1_t$". Removing the zero bits, taking the most significant bits to reveal the value of $ID_k$ and taking the least significant bits to reveal the tag's random numbers ($R^1_t$ and $R^2_t$) is trivial. The reader is referenced to a similar work proposing custom made software using pre-calculated tables is detailed in work [12].

Remembering that initially both $SK_s$ and $R_r$ were zeros, the message D in Figure 1 can be fabricated by using a fake $T^2_2$, where $T^2_2 > T^2_1$. This would lead the tag to update $SK_{s+1}$ to $h(0 \oplus 0 \oplus R^2_t)$, i.e. $h(R^2_t)$. Now all secrets of the tag are exposed, hence the full-disclosure attack is complete. The obtained values can be easily verified by running a third session with the tag, using $R_r = 0$ and $SK_s = h(R^2_t)$. Our demonstrated attack is viable as long as the rainbow table contains the searched output values. It should be noted that the initial zero value of $SK_s$ is only a simplification in the attack. As long as the tag is prevented from updating, the attack methodology is the same for a non-zero $SK_s$.

## 5. THE SECURITY ANALYSIS OF SAKM PROTOCOL

The authors of SAKM argue that their protocol provides high security against most common known attacks, mainly because it is based on hash functions and a synchronized shared secret. However, in the previous section it has been demonstrated that the secrets of a tag can be exposed and the server can be convinced to not recognize a fully legitimate tag. Under the light of the above analysis the security of the SAKM protocol is now re-evaluated.

### 5.1 Secret Disclosure Resistance

The first argument of the designers of SAKM is that valuable information about the tag (tag secrets) cannot be obtained; and hence an attacker cannot pass the authentication steps. This argument is obviously unsound, after the demonstration of the above full-disclosure attack. Moreover, as shown an attacker can complete an authentication run, by being able to fabricate message D.

### 5.2 De-synchronization Resistance:

As far as de-synchronization resistance is concerned, the SAKM protocol relies on the argument that $SK_{s+1} = h(SK_s \oplus R_r \oplus R_t)$ cannot be computed by an attacker. But, the de-synchronization analysis demonstrated in the previous section proves that blocking two consecutive D messages from reaching the tag leads to an unshared secret. This is because the server updates twice, drops the originally shared value; while the tag holds on to a value that is no longer existent in the server database. Therefore, contrary to its designers' arguments SAKM's server falls into de-synchronization with its tags.

### 5.3 Forward Secrecy Resistance

It was demonstrated in our full-disclosure attack that if the second session is continued with fabricated D and $T^2_2$ values; then the tag updates to $SK_{s+1} = h(R^2_t)$. Such a result puts the forward secrecy and tracking resistance of the SAKM protocol into danger. Because, in the next tag authentication sessions, the recorded $SK_{s+1}$ and $ID_k$ pairs in the attacker's database can be used to decrypt the exchanges of between the tags and the server. Starting with A in Figure 1, after obtaining a value for $R^1_t$ the attacker simply goes after a match of computed C' and transmitted C values. As observed, the attacker uses the already exposed session secret key to blow the forward secrecy resistance of the SAKM protocol. However, the attack is not successful if the attacker misses a successfully completed session, because the originally exposed $SK_{s+1}$ would be changed during the missed session. Hence, the attacker can never find matching C values.

### 5.4 Traceability Resistance

With the $SK_{s+1}$ and $ID_k$ pairs in the attacker's database, an attacker can identify a tag, by listening to an authentication session between a server and a tag. Using every entry in its database, one by one the attacker tries to find a match for computed A', C' values and transmitted A, C values. A match means a previously exposed tag is identified. The attack is not successful if the attacker misses a $SK_{s+1}$ value update during a completed session, because the originally exposed $SK_{s+1}$ is no longer valid. Nevertheless, tracking new SAKM tags is quite possible and that represents a violation to user privacy.

## 6. CONCLUSIONS

The present work focuses on a recent work that recommends RFID tags with a hash based authentication protocol for telecare medicine. The health of patients being the most critical entity in medical projects, the proposed authentication protocol needs to be analyzed very carefully. Our detailed analysis shows that the security of the proposed protocol is not as strong as it is claimed by its authors. Two types of known attacks have been demonstrated, in present work. The simple de-synchronization attack forces the server and the tag to fall out of synchronization to a point where the server no longer recognizes a legitimate tag. Our second attack feeds the results of algebraic manipulations of transmitted messages to the popular rainbow table search methodology to fully disclose the secrets of a tag. A

 

software project like in work [12] can automate the exposure of the secrets of the SAKM tags. Exposed tag secrets pose serious threats on the privacy and health of the patients who are using the protocol. As a conclusion, the weaknesses of the analyzed protocol render its use insecure and unsafe in telecare medicine.

## REFERENCES

[1] M. H. Ozcanhan, G. Dalkilic and S. Utku, "Analysis of two protocols using EPC Gen-2 tags for safe inpatient medication," IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA), 2013.

[2] M. H. Ozcanhan, G. Dalkilic and S. Utku, "Is NFC a better option instead of EPC Gen-2 in safe medication of inpatients," Radio Frequency Identification. Berlin, Germany: Springer, 2013, p. 19-33.

[3] H.Y. Chien, "SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity," Dependable and Secure Computing, p. 337-340, 2007.

[4] K. Srivastava et al. "A hash based mutual RFID tag authentication protocol in telecare medicine information system," Journal of medical systems, vol. 39.1, p. 1-5, 2015.

[5] R. Rivest. "The MD5 Message-Digest Algorithm," RFC 1321, 1992.

[6] Bruce Schneier, Applied Cryptography, 1995, ISBN:0-471-11709-9.

[7] G. Bertoni, M. Peeters, G. Van Assche, "Keccak," Advances in Cryptology, EUROCRYPT 2013. Berlin Germany: Springer, p. 313-314, 2013.

[8] M.H. Ozcanhan, "Analysis of a Recent Quadratic Residue Based Authentication Protocol for Low-Cost RFID tags," International Journal of Novel Research in Engineering and Science, vol. 2, issue 1, p. 7-13, March 2015.

[9] J. Gómez et al. "Cryptanalysis of hash functions using advanced multiprocessing," Distributed computing and artificial intelligence. Berlin, Germany: Springer, p. 221-228, 2010.

[10] A. Gildas and X. Carpent, "Optimal Storage for Rainbow Tables," Information Security and Cryptology (ICISC) 2013, Springer International, p. 144-157, 2014.

[11] 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 (FPL), IEEE, 2010.

[12] M.H. Ozcanhan, "Improvement of a Weak RFID Authentication Protocol Making Drug Administration Insecure," Life Science Journal, vol. 11, issue 10 (2014).

## BIOGRAPHIES

The author is a Turkish Cypriot who has a Ph.D. in Computer Engineering (Dokuz Eylul University, 2011). The main research interest of the author is in security in embedded systems, specifically in RFID.

 