Improvement of a Weak RFID Authentication Protocol Making Drug Administration Insecure

Mehmet Hilal Özcanhan

Department of Computer Engineering, Tinaztepe Campus, Dokuz Eylul University, Buca-Kaynaklar, Izmir 35160, Turkey
hozcanhan@cs.deu.edu.tr

Abstract: Many RFID authentication protocols have been proposed for safe drug administration. Unfortunately, many of the proposed protocols have security weaknesses that put the safety and/or privacy of inpatients into danger. A recent protocol is analyzed in this work, which proves to be another such example. The secrets of the inpatient tag are exposed by software and a table created by the present author. The protocol is security upgraded to meet the patient safety expectations, by using lightweight cryptography instead of the hash function of the analyzed protocol. The proposed protocol is complemented by server control mechanisms. Performance and security comparisons show that the present proposal truly guarantees patient safety and has better performance results than its predecessor.

[Mehmet Hilal Özcanhan. **Improvement of a Weak RFID Authentication Protocol Making Drug Administration Insecure.** *Life Sci J* 2014;11(10):269-276]. (ISSN:1097-8135). http://www.lifesciencesite.com. 38

Keywords: RFID; authentication; security; cryptographic protocols; medical applications; healthcare services

1. Introduction

The use of mobile devices and radio frequency identification (RFID) technologies in healthcare services is gaining momentum (Wamba, 2012), (Yao et al., 2012). The aim is to facilitate the healthcare services and face off the negative statistical reports. According to reports, adverse drug events (ADE) due to wrong drug administration are increasing. ADE are harming inpatients and prolonging hospital stays, resulting in loss of human life, health and money (Hickner et al., 2010), (Eurobarometer, 2006). Researchers are trying to solve the problem through RFID aided drug administration protocols.

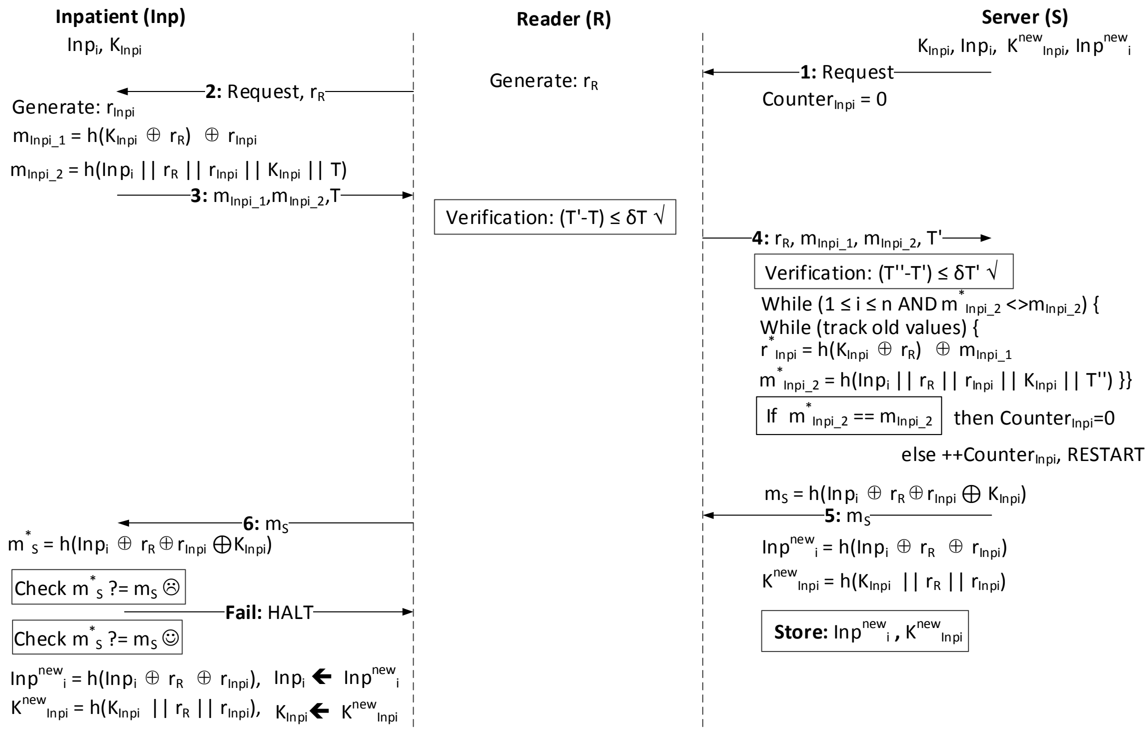
Observing the success of RFID in tracking commercial goods, many protocols have been proposed to solve the ADE problem with the same RFID material and methods (Wamba, 2012). The goal is to identify and match patients with their drugs, by using low-cost RFID tags. But, the gap between providing the required patient safety put forward in the National Patient Safety Goals (Joint Commission, 2009) and the limited resources of low cost tags is huge. The result of the proposals can be revealed by making a small search in the well-known academic indices. Simply, the number of security attacks made on the proposed low-cost RFID protocols almost equals the number of proposals.

The use of RFID enabled mobile devices in healthcare has been described in different works (Wamba, 2012), (Yao et al., 2012), (Özcanhan et al., 2013), (Lathela et al. 2008). Although in 2013, EPC Global GS1 defined version 2 of the well-known EPC Global Class 1 Generation 2 Standard (Gen-2) for low cost tags, there is no standard for healthcare RFID systems. Therefore, it is necessary to at least identify the mandatory security characteristics of RFID aided drug administration protocols, so that use of low cost tags without true confidentiality functions is abandoned.

2. Related Work

Two pioneering works recommending RFID tags in drug administration have been presented in works (Wu et al., 2005) and (Sun et al., 2008). But, they lack detailed description, propose the use of paper barcodes on medicine packets and personal computers as mobile devices. But, barcodes have limited capabilities and disadvantages in patient safety (Chen and Wu, 2012) and personal computers are not mobile. A proposal that identifies both the inpatient's drug and the inpatient using Gen-2 tags was made in work (Huang and Ku, 2009). The security flaws of the proposal however are demonstrated and corrected in work (Chien et al., 2011), which turns out to be also vulnerable according to another work (Yen et al., 2012).

There are numerous works using XOR ($\oplus$), pseudo-random number generator (PRNG) and cyclic redundancy check (CRC) functions – the only available in old Gen-2 version tags. But, it has been shown that these functions do not provide good confidentiality as well as cryptographic functions (Özcanhan et al., 2013), (Van Deursen and Radomirovic, 2009). Hence, protocols based on them demonstrate vulnerabilities. A recent protocol using a non-standard function is presented in (Kaul et al. 2013). The protocol; named Kea for short, is shown in Figure 1. Briefly, Kea accepts hashing as lightweight and uses it with the PRNG and XOR.



Inp_i	Index pseudonym of the ith inpatient wristband
K_{Inpi}	Shared secret key of Inp_i
T, T′, T′′	Timestamp generated by the tag, reader and the server
r_R , r_{Inpi}	Pseudo random number generated by reader and tag Inp_i
h()	A one way hash function
m_{Inpi_x}	xth (x=1,2) message generated by tag Inp_i
m_s	Message generated by server
$\oplus$	XOR function
n	Number of tags within the range of the reader

Figure 1. The analyzed Kea Protocol.

The protocol has three phases. In Initialization Phase, the server and the tag are loaded with an ordered 'secret-identity' pair (K_{Inpi}, Inp_i), each of bit length l. No secret is shared with the reader. In the Authentication Phase, the reader generates and sends a pseudo random number (nonce) r_R to the tag. The tag generates its nonce r_{Inpi}, hides it in m_{Inpi_1} and then prepares its authenticator m_{Inpi_2}; by using an unfounded timestamp T. The tag sends {m_{Inpi_1}, m_{Inpi_2}, T} to the reader. The reader checks the timestamp and then sends {r_R, m_{Inpi_1}, m_{Inpi_2}, T′} to the server. T′ is reader's timestamp which is validated by the server. Trying (K_{Inpi}, Inp_i) pairs in its database one by one, the server computes values for r^*_{Inpi} and $m^*_{Inpi\_2}$ to look for a match between the received m_{Inpi_2} and computed $m^*_{Inpi\_2}$. The match decides the identity of the tag. A counter limits the number of server's matching attempts.

Next, the server computes and sends its authenticator m_S, proving the possession of the nonces and the shared secret pair. Finally, the secrets are updated without waiting for an acknowledgement (ack) from the tag. The old secret pair values are preserved, in case message 5 does not reach the tag. The reader relays m_S to the tag, who computes its own m^*_S version and compares it with the received m_S. A match means successful mutual authentication and the tag goes into its own Update Phase. A tag counter is briefly mentioned, but not described.

Kea is declared as resistant to known RFID attacks, but our crypt-analysis demonstrates inherent weaknesses leading to the conclusion that the proposed protocol is insecure.

3. Security Analysis of Kea Protocol

The following are given in Kea's paper:

1. The bit length of Kea is l; i.e. $Inp_i \in \{0,1\}^l$, $K_{Inpi} \in \{0,1\}^l$, $r_{Inpi} \in \{0,1\}^l$, $r_R \in \{0,1\}^l$.
2. h() is a one way hash function such that h(x) ➔ $\{0,1\}^l$ is relatively easy to compute for any x ➔$\{0,1\}^*$, and for any x it is computationally infeasible to find $z \neq x$ such that h(z) = h(x).
3. $$m_{Inpi\_1} = h(K_{Inpi} \oplus r_R) \oplus r_{Inpi} \qquad (1)$$

4. $m_S = h(Inp_i \oplus r_R \oplus r_{Inpi} \oplus K_{Inpi})$ (2)

Thus, the following can be deduced:

1. $\oplus$ operation on parameters r_{Inpi}, r_R, Inp_i, K_{Inpi} also give l bit results.
2. h() is deterministic such that for any x, both the reader and the tag calculate the same h(x). Since h() is public, a table $T_{h()}$ of 2^l records can be prepared, as in Table 1.

Table 1. Deterministic table $T_{h()}$

x	h(x)	h(˜x)	Q=h(x) ⊕ h(˜x)
In_1	Out_1	Out_{23}	Out_{XOR1}
In_2	Out_2	….	Out_{XOR2}
In_3	Out_3	….	Out_{XOR3}
….	….	….	….
$In_{65,536}$	$Out_{65,536}$	Out_1	$Out_{XOR65,536}$

DoS Attack

Denial of Service (DoS) attack is possible both on the server and the tag of Kea. Consider the computations after the reader sends message 4 to the server. The server first computes r^*_{Inpi}, then $m^*_{Inpi\_2}$ and compares $m^*_{Inpi\_2}$ with the received m_{Inpi_2}. The server repeats this computation cycle for all tag records in its database, until it finds a valid match. Therefore, the possible maximum number of computations is n × (two XOR, four concatenation, two hash operations). If the received m_{Inpi_2} is bogus of an adversary, the match fails and another session is started. Assuming the server counts up to j wrong attempts, [(n + j) × (two XOR, four concatenations, two hash operations)] computations have to be made. If $n + j$ values are high enough, the server's reply to tag cannot reach before a tag timeout.

The second weakness is when the server assumes a completed authentication after sending m_S; computing a new secret pair and keeping the old. But the tag does not update, if m_S is altered or blocked on the way. A server command sent after authentication is denied by the tag and a new authentication is necessary with the saved old secrets. Message m_S can be blocked repeatedly; therefore a second counter is required to count the number of authentication attempts, but it is missing. But, every failed m_S matching forces the tag to increment its own counter. After a few consecutive failed attempts the tag halts the protocol, but the server is unaware and falls out of sync with the tag and the DoS attack succeeds.

Full-disclosure Attack

Let for the ith session m_{Inpi_1} and m_S be denoted as $m^i_{Inp\_1}$ and m^i_S, respectively. Assuming an adversary plays r^1_R = 0000_H to the tag with a dishonest reader in session 1, from Figure 1:

$$m^1_{Inpi\_1} = h(K_{Inpi}) \oplus r^1_{Inpi} \quad (3)$$

$$m^1_S = h((Inp_i \oplus K_{Inpi}) \oplus r^1_{Inpi}) \quad (4)$$

The adversary blocks m^1_S to the tag and plays r^2_R = $FFFF_H$ to the tag, in session 2:

$$m^2_{Inpi\_1} = h(\sim K_{Inpi}) \oplus r^2_{Inpi} \quad (5)$$

$$m^2_S = h(\sim (Inp_i \oplus K_{Inpi}) \oplus r^2_{Inpi}) \quad (6)$$

Although the server updates Inp_i and K_{Inpi} after each unsuccessful session, the tag does not. Therefore Inp_i and K_{Inpi} are the same in the two sessions. From (3):

$$h(K_{Inpi}) = m^1_{Inpi\_1} \oplus r^1_{Inpi} \quad (7)$$

Let the value of m^1_S in (4) to be Out_3, in Table 1. A search is made in the h(x) column until Out_3 is found. Let the corresponding input value be In_3. From (4):

$$In_3 = (Inp_i \oplus K_{Inpi}) \oplus r^1_{Inpi}) \quad (8)$$

Substituting the value of r^1_{Inpi} in (8), into (7):

$$h(K_{Inpi}) = m^1_{Inpi\_1} \oplus (Inp_i \oplus K_{Inpi}) \oplus In_3 \quad (9)$$

Repeating the same argument above for (5) and (6):

$$h(\sim K_{Inpi}) = m^2_{Inpi\_1} \oplus r^2_{Inpi} \quad (10)$$

$$In_2 = \sim (Inp_i \oplus K_{Inpi}) \oplus r^2_{Inpi} \quad (11)$$

Substituting the value of r^2_{Inpi} in (11) into (10):

$$h(\sim K_{Inpi}) = m^2_{Inpi\_1} \oplus \sim (Inp_i \oplus K_{Inpi}) \oplus In_2 \quad (12)$$

Now, XORing (9) with (12) and simplifying:

$$h(K_{Inpi}) \oplus h(\sim K_{Inpi}) = \sim m^1_{Inpi\_1} \oplus In_3 \oplus m^2_{Inpi\_1} \oplus In_2 \quad (13)$$

Because (Inp_i⊕K_{Inpi})⊕˜(Inp_i⊕K_{Inpi})=$FFFF_H$. Hence, h(K_{Inpi})⊕h(˜K_{Inpi}) is a value that can be found from exchanged values (13). For simplicity replacing ˜$m^1_{Inpi\_1}$⊕In_3⊕$m^2_{Inpi\_1}$⊕In_2 with notation Q:

$$h(K_{Inpi}) \oplus h(\sim K_{Inpi}) = Q \quad (14)$$

From Table 1, the rows which satisfy $\boldsymbol{Q}$ are found. The corresponding x =K_{Inpi}, h(K_{Inpi}) and h(˜K_{Inpi}) values are obtained. Substituting in (3) and (5), r^1_{Inpi}, r^2_{Inpi} values are obtained. Finally from (8) Inp_i is obtained. Timestamps are passed in plain text, therefore T and T' are available. Substituting the obtained values in the calculated $m^*_{Inpi\_2}$ = h(Inp_i||r_R||r_{Inpi}||K_{Inpi}||T) and checking with transmitted m_{Inpi_2}; the case where $m^*_{Inpi\_2}$ = m_{Inpi_2} holds, the Inp_i and K_{Inpi} values are correct. Hence, the secrets are captured. The capture of the secrets is devastating because this vulnerability can be exploited for malicious intentions.

The above algebraic attack has been simulated by software available from the author through e-mail, which is run offline after eavesdropping on two consecutive sessions. Random values for Inp_i, K_{Inpi} and nonce r_{Inpi} are assumed and Table 2 is prepared for l = 16 and 32. The two tables have 65,536 and 4,294,967,296 records; and are sorted, based on [h(K_{Inpi})⊕h(˜K_{Inpi})] values.

Table 2. 16 and 32 bit simulation results

Run #	Inp_i	K_{Inpi}	r_{Inpi}	Av. time (ms)
1, l = 16	39017	17767	9158	1.02
2, l = 16	23807	18547	56401	1.00
3, l = 16	55199	29283	49715	1.10
4, l = 16	55211	31949	22714	1.12
5, l = 16	43491	16882	7931	0.97
6, l = 32	3959529863	26500	1245719430	9864
7, l = 32	4016134551	24464	2426312360	9921
8, l = 32	616919038	491	2189695706	19720
9, l = 32	3220908140	14604	3221546860	19703
10, l = 32	2755712171	23281	33640809	4911

The software searches Table 1 and for a Q value found, the corresponding $x = K_{Inpi}$, $h(K_{Inpi})$ and $h(\tilde{K}_{Inpi})$ values are read. The obtained values are substituted to test the match $m^*_{Inpi\_2} ?= m_{Inpi\_2}$. The match exposes the secrets. Table 2 shows the results of 5 simulation sessions each, for l = 16 and 32. Full-disclosure of secrets takes on the average 1.04 ms for l = 16 and 12.82 sec. for l = 32. In l = 32, the number of Q value candidates is very high and accordingly the secret capture time is around 13 seconds. Nevertheless, the attack is feasible. The equipment used is an Intel i-7 quad core notebook with Microsoft Windows 7 operating system.

4. Proposed protocol – SC-SRP

Weak encryption and lack of multiple counters led to the demonstrated Kea vulnerabilities. Therefore, the main goal should be to design a secure protocol which truly guarantees patient safety. This is the main motivation behind the proposed Server Complemented Secure RFID Protocol (SC-SRP) in Figure 2, which uses the new version Gen-2 standard's support for cryptographic functions.

Assumptions and the notation of SC-SRP

The assumptions of Kea have been preserved. Hence, the reader – server channel is secure, but the tag – reader air channel is not. The reader can be dishonest which can be used to block or interfere with the exchanged messages. The same notation of Kea is used, but the hash is replaced by a lightweight cryptographic function; like KLEIN (Gong et al., 2012) or LED (Guo et al., 2011), in accordance with Gen-2 ver2. The tag EPC (ID_i), shared secrets and timestamps are 64 bit. The PRNG produces 32 bit nonces, which can be used to produce 64 bit values by concatenating two nonces.

The proposed protocol

SC-SRP has three major differences than Kea. First, there is an additional 4th step; second, a lightweight encryption function is used; and third there are two counters for each tag in the server database (Table 3). The tag record also contains a static ID_i, a dynamic index pseudonym Inp_i, a shared secret K_{Inpi} and three previous secret values, notated by superscripts. The initial values loaded on each side are shown under the parties (Figure 2). No secrets are shared with the reader.

The Request Phase– The server provides timestamp T to the reader, because passive tags cannot. The reader acts as a mediator, generates its message freshness nonce r_R and sends it to the tag with T. Sending T provides security and removes the need for time synchronization.

Tag Response Phase– The energized tag generates a nonce r_{Inpi} to use with r_R and T, in formulating its replies. A key ($K_{Inpi} \oplus T$) is formed to encrypt K_{Inpi} and then XOR it with the concatenated nonces to hide r_{Inpi}, in m_{Inpi_1}. Then m_{Inpi_2} is prepared to pass Inp_i and prove the reception of T.

Tag Verification and Server Secret Update Phase– The reader first validates the time of the tag response. If response delay is within limits, the tag messages are passed on to the server, with the reader timestamp T' and r_R. The server checks both time delays. Then, the server tries each value of K_{Inpi} in the database to compute a tag nonce r^*_{Inpi}. Using the computed nonce value, authenticator $m^*_{Inpi\_2}$ is calculated and compared to the received m_{Inpi_2}. A match identifies the tag's record shown in Table 3. If a match is not found, the server increments counter C_{Inpi_1}. If the counter has reached the threshold, an alarm is raised to announce a possible DoS attack or the presence of a non-classified tag; the tag is placed in the blacklist. Before continuing the authentication of the identified tag, the server increments counter C_{Inpi_2}. If the threshold has been reached, an alarm is raised to indicate that this tag has been identified three times before, but its previous authentications have not finished correctly; the tag is put into the blacklist. If no alarms are raised, the server updates and ensures that the new secret K^{new1}_{Inpi} is nonzero and not equal to other tag secrets (no collision). Normally, update continues with Inp^{new1}_i computation and now $K^0_{Inpi} = K^{new1}_{Inpi}$ and $Inp^0_i = Inp^{new1}_i$; $K^{-1}_{Inpi} = K_{Inpi}$ and $Inp^{-1}_i = Inp_i$ as shown in Table 3. Next, authenticator m_{S_1} is computed to prove that server has ID_i and the dynamic (Inp_i, K_{Inpi}) pair of the tag.

In the Update Phase, the new values are obtained by encrypting the old secrets XORed with nonces, using a dynamic key obtained from ID_i, concatenated nonces and T. Hence, the update depends on both sides' parameters and the unique ID_i. If the server does not receive the ack message 8 from the identified tag, it keeps the pairs (Inp^0_i, K^0_{Inpi}), (Inp^{-1}_i, K^{-1}_{Inpi}) and the C_{Inpi_2} value.

Server Verification and Tag Secret Update Phase– The reader simply relays the authenticator m_{S_1} to the tag. The tag computes a new key for encryption to find its version $m^*_{S\_1}$. If the computed and the

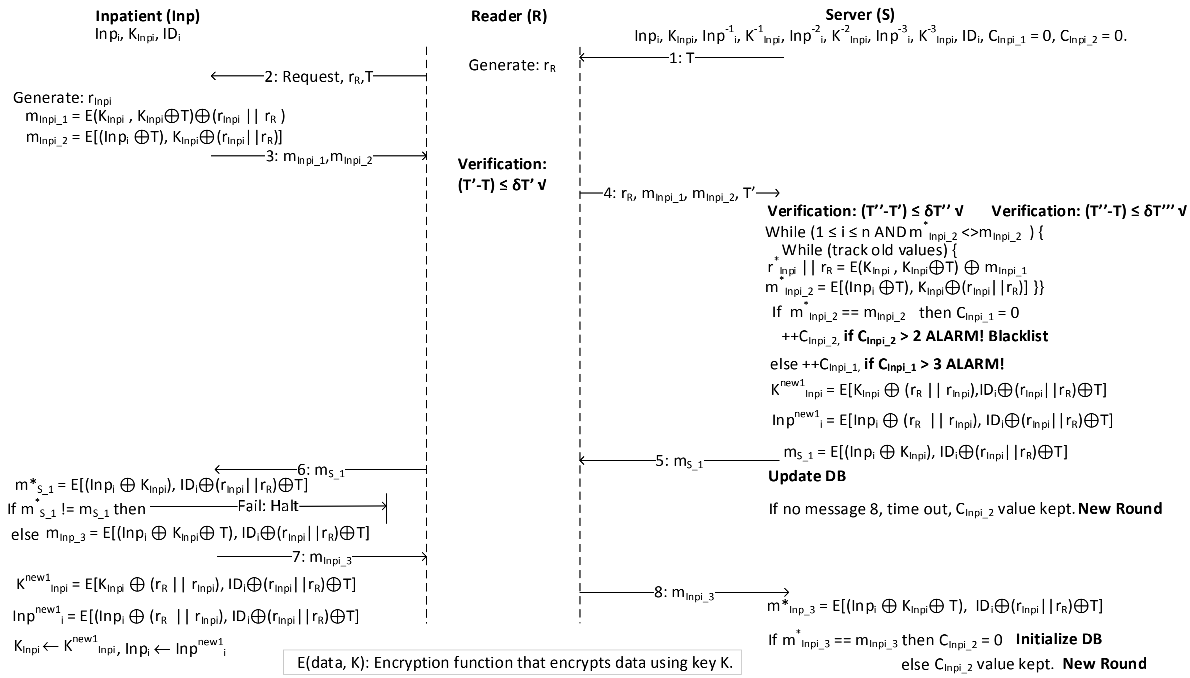


Figure 2. Proposed protocol:SC-SRP.

received authenticators match, mutual authentication of the parties is complete. Then, the tag computes, sends its ack m_{Inpi_3} and updates its secrets. The reader simply relays the ack.

The Tag Update Acknowledgement Phase– After receiving the tag's ack, the server confirms tag update. Only then the server resets C_{Inpi_2} and initializes the record of the tag to the 1st row of Table 3. Notice that if a tag ack is not received the new session starts with the old the secret pair (Inp^{-1}_i = Inp_i, K^{-1}_{Inpi} = K_{Inpi}). If another tag ack fails, the record of the tag is the 2nd session row of Table 3. After three consecutive failed tag acks, C_{Inpi_2} overflows and the tag is placed on the blacklist. But, the oldest values (K^{-3}_{Inpi} = K_{Inpi} and Inp^{-3}_i = Inp_i) are not flushed out of the tag's record. Hence, there is no risk of losing the tag. The tag can be re-admitted into the whitelist, after administrator intervention.

5. Comparison and evaluation of protocols

The proposed SC-SRP's performance and security are compared with Kea's. The security analysis covers the attacks launched on Kea in this work and the known RFID attacks. But first, two apparent and critical design errors of Kea need to be pointed out. The tag of Kea supposedly produces a timestamp T. But passive tags do not have a power source to supply continuous energy for a timer. Secondly, T is not passed to the server by the reader (Figure 1). As a result, the server can not verify m_{Inpi_2}. Omission of T cannot be a typing mistake, because verification of T by the server is also missing. T's failure to reach the server opens the avenue to suppress-replay attack (Gong, 1992). Thus, T has to reach the server to despair any dishonest reader. SC-SRP does not have these ambiguities.

Performance analysis of SC-SRP

The chip area requirement of a hardware implementation is measured in μm^2, which is dependent on the fabrication technology. In order to compare the area requirements independently, a measurement called gate equivalents (GE) – hardware complexity– is used (Paar et al., 2009). One GE is equivalent to the area which is required by a two-input NAND gate; i.e. GE is derived by dividing the area in μm^2 by the area of a two-input

Table 3. Each tag's record in the database

Session	K^0_{Inpi}	Inp^0_i	K^{-1}_{Inpi}	Inp^{-1}_i	K^{-2}_{Inpi}	Inp^{-2}_i	K^{-3}_{Inpi}	Inp^{-3}_i	C_{Inpi_1}	C_{Inpi_2}	EPC
0	K_{Inpi}	Inp_i	-	-	-	-	-	-	X	0	ID_i
1	K^{new1}_{Inpi}	Inp^{new1}_i	K_{Inpi}	Inp_i	-	-	-	-	X	1	ID_i
2	K^{new2}_{Inpi}	Inp^{new2}_i	K^{new1}_{Inpi}	Inp^{new1}_i	K_{Inpi}	Inp_i	-	-	X	2	ID_i
3	K^{new3}_{Inpi}	Inp^{new3}_i	K^{new2}_{Inpi}	Inp^{new2}_i	K^{new1}_{Inpi}	Inp^{new1}_i	K_{Inpi}	Inp_i	X	3	ID_i

NAND gate. Using this method, the GE for each gate and flip flop; hence, the total GE of the registers and functions of a theoretical scheme can be estimated.

Table 4 covers the GEs and clock cycles spent for encrypting one block of data, by some popular hashing and encryption functions. The product of the GE and clock cycles is accepted as a measure of computational complexity (Feldhofer and Wolker, 2009). A lightweight implementation of the NIST's SHA-3 contest winner Keccak spends 900 clocks and uses 2520 GE (Kavun and Yalçin, 2010). Thus, Keccak's computational complexity is over 2.2 million GE-clocks. The hash functions are known to have larger computational complexity values than encryption functions (Feldhofer and Wolkerstorfer, 2009). A lightweight implementation of the popular AES function (Moradi et al., 2011) has four times less computational complexity than Keccak (Kavun and Yalçin, 2010). However, the lightweight KLEIN encryption function has the lowest complexity with 252,540 GE-clocks; almost nine times less than Keccak. It is clear that KLEIN based SC-SRP consumes less die area and spends less number of clocks than Kea. From Table 4, lightweight encryption functions (Gong et al., 2012), (Guo et al., 2011), (Moradi et al.,2011) all appear to have less computational complexities than Kea.

Table 4. Gate Equivalents (GE) and Clock Cycles used by the compared algorithms

Algorithm	Encryp. Load (Clock Cycles)	Chip Area (GE)	Area × Delay (Complexity)
LED-64	1248	966	1,205,568
KLEIN-64	207	1220	252,540
Light AES	226	2400	542,400
AES	1032	3400	3,508,800
Keccak	900	2520	2,268,000

For a better, direct comparison of SC-SRP and Kea, Table 5 is given. The server memory used per tag in SC-SRP is higher. But, additional $4l$ and a counter (total of $9 \times 64 + 2 \times 8 = 592$ bits/tag) is not a decisive load on a server with giga bytes of primary and secondary memory. Meanwhile, the only additional memory that appears to be used on a SC-SRP tag is l; which is not critical. Therefore, memory requirements are not critically different. Another minor difference is in the total number of exchanged messages, where SC-SRP transmits only l additional bits. But the decisive factors like the total number of clock cycles taken by the protocol and the maximum clock cycles of a particular step are very different.

The same functions are used in both protocols except for encryption, which is denoted as f() in Table 5. But, that difference results in a big advantage on SC-SRP's performance over Kea's. Even if 21 total function calls are made; 8 more than Kea (Table 5's last row), SC-SRP is still more efficient. Simply because the hash function of Kea spends hundreds of more clock cycles. For example, the total 3 hash calls of Kea requires 3 * 900 = 2700 clock cycles, compared to SC-SRP's total encryption calls 4 * 207 = 828 clock cycles. The XOR, concatenate and PRNG operations spend 2, 1 and 72 clocks (Özcanhan et al., 2013) respectively.

Table 5. Certain characteristic of study sites

Characteristic	Protocol	
	Kea	SC-SRP
Architecture	16 bit	32 bit
Server Memory	5 l + 1 counter	9 l + 2 counter
Tag Memory	2 l	3 l
Exchanged Bits	5 l	6 l
Functions used	PRNG, $\oplus$, ǁ, hash	PRNG, $\oplus$, ǁ, encryption
# of Calls: PRNG, $\oplus$, ǁ, f()	1, 5, 4, 3	2, 11, 4, 4

Both SC-SRP and Kea use their encryption or hashing function maximum twice in one step. Neglecting the clock cycles of XOR and concatenation operations, SC-SRP spends 414 and Kea 1800 clock cycles. Therefore, SC-SRP is more efficient than Kea both in the overall total and the maximum per step clock cycles.

Security analysis of SC-SRP

Not only vulnerable, Kea fails to consider the possibility of collision of shared secrets. But, SC-SRP avoids collision of secrets or zero value secrets; and resists known attacks, including those launched on Kea in this work.

Resistance to DoS Attack– The above demonstrated or other DoS attacks on the server are resisted by two counters placed in each tag's record. After a number of failed authentications or ack receptions, alarm generation through counters C_{Inpi_1}, C_{Inpi_2} and blacklist relegation stop the attack. DoS attack on the tag cannot succeed either, because tag drops an exchange if m_{S_1} matching fails. Therefore, SC-SRP is more secure than Kea, against DoS attacks.

Resistance to De-synchronization Attack– The second counter C_{Inpi_2} in the record for each tag dissolves any de-synchronization attack. A tag is placed in blacklist after three consecutive failed acks. The counter's maximum value is such that originally shared (K_{Inpi}, Inp_i) pair is never flushed from the tag record, shown as row 3 of Table 3. Therefore, a blacklisted tag can be promoted back to whitelist after administrator intervention. Thus, the tag is not lost and medication can proceed.

Resistance to Full Disclosure or Cloning Attack– SC-SRP employs a 64 bit encryption algorithm supported with dynamic keys. Therefore, the secret values Inp_i, K_{Inpi} and ID_i are secure until KLEIN is fully analyzed. Without capturing the secrets of exchanged messages, tag cloning is not possible. Cloning through physical tampering of low cost tags is outside the scope of this work. The pre-calculated table attack of Section 3 fails because of two reasons. Firstly, the table length is now 2^{64}, therefore the table search used to attack Kea is infeasible. Secondly, the keys of encryption operations are changed even within a single session. Therefore, the brute force attacks launched at exposing a constant encryption key by ciphertext-only attacks cannot be used on SC-SRP. Hence, SC-SRP is more secure than Kea against full disclosure attacks.

Resistance to Man in the Middle Attack– This attack cannot succeed in SC-SRP because the man in the middle cannot conclude an authentication with either the tag, or the server. With the secrets encrypted and unknown to the adversary m_{Inpi_1}, m_{Inpi_2}, m_{S_1} and the acknowledgement m_{Inpi_3} cannot be fabricated. Therefore, attacks of any adversary playing in the middle fail.

Resistance to Traceability Attack– Tracing the tag in SC-SRP is not possible because index pseudonym Inp_i is updated at the end of every successful authentication. The new Inp_i is obtained through encryption of the old value, obscured with the exchanged nonces and with a dynamic key dependent on the ID_i, nonces and the server timestamp T. The tag placed into a blacklist is not traceable either, because it is removed from use.

Resistance to Replay and Parallel Session Attacks– These attacks are resisted by the use of the two nonces and the server timestamp T, which is absent in Kea. The server provides T to prevent replay of old successful session values. Also, the use of both generated nonces provides mutual message freshness. In SC-SRP, a parallel attack is not possible either, because of the (K_{Inpi}, Inp_i) and T timestamp bonding.

Resistance to Impersonation Attack– Neither the tag, nor the server can be impersonated because their authentication steps cannot be formulated by an adversary. The secrets cannot be decrypted because they are dynamically encrypted with keys depending on the nonces and the server timestamp T. Therefore, even if the exchanged messages are collected, they cannot be used to fabricate the authenticators and impersonate a party.

Conclusion

A recent healthcare RFID authentication protocol which proposes keeping time on passive tags and one way hashing for creating authenticators has been analyzed. Analyses reveal that the Kea protocol is erroneous and vulnerable to multiple attacks. With a generic table and software created by the present author; the secrets of the used tags are fully-exposed, which makes Kea's drug administration insecure. An alternative protocol SC-SRP complying with the new standard version of Gen-2 has been presented. SC-SRP uses lightweight cryptography complemented by database server control mechanisms. In addition, SC-SRP achieves encryption with varying keys, instead of a static key. Comparison shows that SC-SRP has overall better performance and higher security than its predecessor. Therefore, SC-SRP is a better tool for RFID aided ubiquitous hospital applications, based on low cost tags. Supported by standards, lightweight cryptography can be the mandatory confidentiality algorithm for healthcare RFID protocols.

Corresponding Author:
Dr. Mehmet Hilal Özcanhan
Department of Computer Engineering
Tinaztepe Campus, Dokuz Eylul University
Buca-Kaynaklar, Izmir 35160, Turkey
E-mail: hozcanhan@cs.deu.edu.tr

References

1. Wamba SF. RFID-enabled health care applications, issues and benefits: An archival analysis (1997–2011). Journal of Medical Systems 2012;36(6):3393-3398.
2. Yao W, Chao HC, Zang L. The adoption and implementation of RFID technologies in health care: a literature review. Journal of Medical Systems 2012;36(6):3507-3525.
3. Hickner J, Zafar A, Kuo GM, Fagnan LJ, Forjuoh SN, Knox LM, Tierney WM. Field Test Results of a New Ambulatory Care Medication Error and Adverse Drug Event Reporting System—MEADERS. The Annals of Family Medicine 2010;8(6):517-525.
4. Special Eurobarometer 241. Medical errors. Wave 64.1 & 64.3. TNS Opinion and Social. European Commission, Jan. 2006.
5. Joint Commission on Accreditation of Health care Organizations. Approved: 2010 National Patient Safety Goals (NPSGs). Joint Commission Perspectives 2009;29(10).
6. Özcanhan MH, Dalkılıç G, Utku S. Is NFC a Better Option Instead of EPC Gen-2 in Safe Medication of Inpatients. Radio Frequency Identification 2013;(8262):19-33.
7. Lathela A, Hassinen M, Jylha V. RFID and NFC in Health care: Safety of Hospitals Medication Care. Proceedings International Conference on Pervasive Computing Technologies for Healthcare. IEEE 2008;241-244.
8. Wu F, Kuo F, Liu LW. The application of RFID on drug safety of inpatient nursing health care.

Proceedings 7th International Conference on Electronic Commerce. ACM 2005;85–92.
9. Sun PR, Wang BH, Wu F. A new method to guard inpatient medication safety by the implementation of RFID. Journal of Medical Systems 2008;32(4):327–332.
10. Chen CL, Wu CY. Using RFID yoking proof protocol to enhance inpatient medication safety. Journal of Medical Systems 2012;36(5):2849-2864.
11. Huang HH, Ku CY. A RFID grouping proof protocol for medication safety of inpatient. Journal of Medical Systems 2009;33(6):467-474.
12. Chien HY, Yang CC, Wu TC, Lee Cf. Two RFID-based solutions to enhance inpatient medication safety. Journal of Medical Systems 2011;35(3):369-375.
13. Yen YC, Lo NW, Wu TC. Two RFID-Based Solutions for Secure Inpatient Medication Administration. Journal of Medical Systems 2012;36(5):2769-2778.
14. Van Deursen T, Radomirović S. Algebraic attacks on RFID protocols. Information Security Theory and Practice. Smart Devices, Pervasive Systems, and Ubiquitous Networks. Springer 2009;5746:38-51.
15. Kaul SD, Awasthi AK. RFID authentication protocol to enhance patient medication safety. Journal of Medical Systems 2013;37(6):1-6.
16. Gong Z, Nikova S, Law YW. KLEIN: a new family of lightweight block ciphers. RFID Security and Privacy. Springer 2012;7055:1-18.
17. Guo J, Peyrin T, Poschmann A, Robshaw M. The LED block cipher. Cryptographic Hardware and Embedded Systems–CHES 2011. Springer 2011;6917:326-341.
18. Gong L. A security risk of depending on synchronized clocks. ACM SIGOPS Operating Systems Review 1992;26(1):49-53.
19. Paar C, Poschmann A, Robshaw MBJ. New designs in lightweight symmetric encryption. RFID Security. Springer 2009;349-371.
20. Feldhofer M, Wolkerstorfer J. Hardware implementation of symmetric algorithms for RFID security. RFID Security. Springer 2009;373-415.
21. Kavun EB, Yalçın T. A lightweight implementation of Keccak hash function for radio-frequency identification applications. Radio Frequency Identification. Springer 2010;258-269.
22. Moradi A, Poschmann A, Ling S, Paar C. Pushing the limits: a very compact and a threshold implementation of AES. Advances in Cryptology–EUROCRYPT 2011. Springer 2011;6632:69-88.
23. Özcanhan MH, Dalkılıç G, Gürle MC. An ultra-Light PRNG for RFID tags. Computer and Information Sciences III. Springer 2013;231-238.

6/6/2014