PUMAP: A PUF-Based Ultra-Lightweight Mutual-Authentication RFID Protocol

Ramzi Bassil, Wissam El-Beaino, Wassim Itani, Ayman Kayssi, Ali Chehab
Department of Electrical and Computer Engineering
American University of Beirut, Lebanon
{rtb01, wte01, wgi01, ayman, chehab}@aub.edu.lb

Abstract

Radio Frequency Identification (RFID) is a technology used for automatic identification of objects, people, and virtually anything one can think of. Applications of RFID technology are expanding and its usage is being adopted worldwide. As such, major efforts have been made to secure the communications in RFID systems and to protect them from various attacks. This paper surveys RFID systems, citing some of their applications as well as the numerous security vulnerabilities they suffer from. Then, some of the proposed solutions that guard against these vulnerabilities are presented and discussed. Then, a novel approach to achieve mutual authentication for ultra-lightweight tags is proposed using Physically Unclonable Functions (PUFs). The proposed approach provides robust security properties as well as good performance characteristics. A proof of concept implementation of the proposed protocol was done on Java programming language that proved the feasibility and efficiency of the protocol.

1. Introduction

An RFID system consists of three main components that enable it to operate and function properly, and those are: a reader, a set of tags, and a backend database or a server. The reader is a device that queries the tags to identify them. RFID tags are very cheap devices that consist of small integrated circuits equipped with a radio antenna. These tags are given each a unique ID number and are mounted on all the objects that are intended to be identified. The processing power of RFID tags varies according to their type. Passive tags do not have an internal battery and are powered by the signal sent by the reader. Those tags have very little processing power and do not support any cryptographic operations. Active tags are self-powered and support more complicated operations, and thus are better suited for secure protocols. The drawback of such tags is that they are more expensive and may not be widely adopted. Semi-active tags are a compromise between the two, as they have an on-board battery but cannot initiate communication and can serve as sensors in specific environments. The third component is the backend database that contains detailed information about each tag; this database is connected to the reader so that when the reader queries a tag, it sends to the backend database the ID that it received from the tag, and the backend database will be able to uniquely identify the tag and to provide all the details related to the corresponding object.

RFID technology allows the tagging of a product with an Electronic Product Code (EPC), which has several advantages over the traditional Universal Product Code (UPC) associated with a bar code. Hence, the major use of RFID systems is in supply chain management, where the manufacturer can track the production process starting from its early stages up until the customer buys the product, and even beyond that. This last point creates a lot of controversy and is inhibiting the wide scale deployment of RFID tagged products due to the privacy concerns of consumers.

The rest of the paper is divided as follows: in section 2 the main security concerns in RFID systems are presented, in section 3 a literature survey of proposed protocols that address these concerns is provided, and in section 4 we present our proposed authentication protocol. Sections 5 and 6 present the security and performance analysis of the protocol respectively, followed by a proof of concept implementation in section 7, and finally the conclusions in section 8.

2. Security threats and Attacks

In [1], Ari Juels et al. state that the adoption of RFID technology will present "unique privacy and security concerns". The authors move on to divide RFID security threats into two main categories: the ones affecting corporations and large companies, and the ones that affect individuals. Concerning the first threat, tagging all produced objects using RFID tags will expose the producing firm to corporate espionage threat, due to the fact that competitors might be able to gather confidential supply chain data. Also, corporate threats include competitive marketing threat whereby illegal access to consumer preferences is obtained and infrastructure threat where competitors can easily jam RFID radio signals. The threats that affect individuals are all related to privacy. Those include action threat whereby the individual's behavior is known by monitoring the group of tags that he possesses, association threat where the customer's identity is linked to a certain tag, and location threat where the

owner's location is divulged when covert readers scan their RFID tags. An additional vulnerability that renders the RFID system prone to more attacks is the cloning threat, whereby an attacker can build a cloned tag which will be interpreted by the reader as a legitimate tag, due to the fact that most tags are not tamper-proof.

Another classification of RFID security threats is presented in [2] whereby Song and Mitchell classify the attacks into two types: weak attacks and strong attacks. Weak attacks are characterized by the fact that they are feasible just by eavesdropping and manipulating communications between tags and readers. Weak attacks lead to threats that include tag impersonation attack where an attacker could forge a tag that is authenticated by the reader. Another threat is the replay attack where an attacker resends the same information used in previous sessions between the tag and the reader in order to authenticate itself. Moreover, denial of service can take place when an attacker intercepts and blocks the communication between the tag and the reader. This type of attack might have dangerous implications as it may cause the reader and the tag to lose their synchronization, thus inhibiting any future communications between the two. Strong attacks become possible when the malicious attacker compromises a tag and obtains access to its non-tamper-proof memory. When this occurs, the attacker would be able to obtain knowledge about the tags previous interactions as well as helping the attacker to identify future interactions, and those are known as backward traceability and forward traceability, respectively. A third attack is concerned with server impersonation when the attacker is able to impersonate a legitimate server, thus making future sessions with the legitimate sever impossible.

3. Previous Work

Before presenting formal security protocols that are used to resolve the vulnerabilities discussed in section 2, a high level description of security enhancing practices will be described first. In [3], Ari Juels conducted a survey on RFID tags by stating some approaches that may help in preserving security such as killing the tag following the purchase of the tagged product, but this solution will prevent the user from getting any post-sale benefits. Another approach is renaming, and that is used to prevent tracking and protect the user's privacy. This could be done by changing the RFID tag identifiers over time. Minimalist cryptography uses pseudonyms where each tag responds differently to different reader queries. Thus, an unauthorized reader will be unable to correlate different appearances of the same tag. Moreover, the proxy approach could be used, where consumers might carry their own privacy-protecting devices such as "RFID Guardian" and "RFID Enhancer Proxy". The guardian acts like an RFID firewall. It selectively simulates tags under its control. The guardian uses different frequencies or different channels such as GPS or internet connections. Also, blocking approaches are discussed, whereby blockers could be used in order to ban any tag reading. However, this method fails when sophisticated readers are deployed. A different scheme is based on antenna-energy analysis. This technique protects privacy by revealing less information when the reader is further away. Thus distance is considered as an important trust metric in this case, and it is concluded through signal-to-noise ratio measurements.

In [4] Wehbe et al. make use of the capabilities of Electronic Product Code Class-1 Generation 2(C1G2) tags. These tags have a 16-bit pseudo-random number (PRN) generator and the ability to compute cyclic redundancy checks (CRC), among other features. In this protocol three secrets (S1, S2, and S3) are shared between the tag and the backend database. The initial query from the reader should contain a manufacturer ID (MID) number, so that the reader would communicate with a tag produced by this manufacturer. The tag that has the specific MID stored in it would generate a random number, XOR it with S1, concatenates its "hidden" tag ID with them, and sends the result to the reader. The reader will then perform several operations on the received message, and it will be able to retrieve the tag ID and the tag's three secrets if it were a legitimate reader. The reader will then perform a cryptographic cyclic redundancy check (CCRC) to the random number sent by the tag and appends to it S2 XORed with a new random number produced by the reader. Upon receiving this message, the tag can authenticate the reader by checking the CCRC, and then it will perform a CCRC to the random number sent by the reader using the third secret S3. The reader will then examine the last message, and thus it can authenticate the tag. After the mutual authentication phase, new pseudonyms are produced for the tag to protect its privacy and to disable tracking in future sessions. The authors argue that this scheme protects against most RFID security threats, but it will fail when the reader is compromised. Additionally, random number generation and CCRC operations cannot be performed by passive tags.

In [2], one of the most secure and reliable RFID security protocols in the literature is presented. Song and Mitchell try to account for the limited processing and computation power in tags by minimizing the use of complex cryptographic functions, and instead using right and left shifts and bit-wise XOR operations. Moreover, the protocol uses a hash function, a message authentication code (MAC), and a pseudorandom bit generator (PRBG) to achieve security. The protocol is divided into two steps: the initialization phase and the authentication phase. The

initialization phase takes place during the manufacturing of the tag. The manufacturer assigns a string u_i of size l bits to each tag T_i, and stores the hash of u_i ($t_i = h(u_i)$), in the tag. For each tag, two sets, $(u_i, t_i)_{old}$ and $(u_i, t_i)_{new}$ are stored in the backend database to provide immunity against de-synchronization. The authentication process takes place as follows:

- The reader generates a random bit-string r_1 and sends it to T_i.
- The tag generates a random bit-string r_2, computes $M_1=t_i \oplus r_2$ and $M_2 = MAC(r_1 \oplus r_2)$ using t_i as the secret key, and sends M_1 and M_2 to the reader.
- Upon receiving M_1 and M_2, the reader will forward them along with r_1 to the backend database.
- The database will try to identify the tag using the received parameters from the reader and if no tag is found the session is stopped. Otherwise, the database will compute M_3 by first performing right circular shift of r_2 by $l/2$ bits ($r_2 >> l/2$) and XORing the result with u_i and it sends the result, M3, to the reader. Also, the database will update the values of $u_{i(old)}$ and $t_{i(old)}$ and sets $u_{i(new)}=(u_i << l/4) \oplus (t_i >> l/4) \oplus r_1 \oplus r_2$, and $t_{i(new)}=h(u_{i(new)})$.
- The reader sends M_3 to the tag.
- The tag performs checks to authenticate the reader, by computing $u_i = M_3 \oplus (r_2 >> l/2)$, then hashing it; if this does not match the value of t_i stored then it will stop the session. Otherwise, it will update the values of u_i and t_i in the same way as in (4) to stay in sync with the backend database.

The protocol described above is highly regarded as one of the best protocols to secure RFID systems because it tackles the most critical security threats that were presented in section 2, and at the same time achieves that with very good performance and minimal storage requirements, but it assumes that tags are able to compute hash functions which is not always a valid assumption like the case of passive tags.

While the above protocol seemed to present the optimal security for active tags, the work done in [5] proves three vulnerabilities in this protocol. Cai et al. argue that tag impersonation, reader impersonation, and de-synchronization attacks are possible due to the extensive use of inexpensive security operations such as XOR and right and left circular shifts. The authors modified the protocol to render it immune to the attacks mentioned above. The main contribution is that secure hash functions were used instead of right and left circular shifts. Although hash functions are computationally expensive, they are an essential replacement for the shift operations. So, the revised protocol includes one additional hash calculation, which is an acceptable additional overhead according to the authors who prove that all the security goals are achieved in this modified protocol. The next step in RFID security according to the authors is to find the lower bound for computational and storage requirements of "secure" protocols.

In [6], Oren et al. present an efficient authentication protocol that provides security and privacy by using a low-resource public-key identification scheme. The implemented protocol respects the power and area limitations of RFID tags. The cryptographic scheme is called Weizmann-IAIK Public key for RFID (WIPR). The motivation is to allow the tag to transmit its ID to a reader without being revealed to any adversary. The authentication process takes place as follows:

- The public key n will be saved in the tag with a unique identifier ID. The reader is provided with the private key (p,q)
- The reader generates a random bit string r_r with length $|r_r| = \alpha$. The tag generates $r_{t,1}$ and $r_{t,2}$ such that $|r_{t,1}| = |n| - \alpha - |ID|$ and $|r_{t,2}| = |n| + \beta$, α and β are security parameters.
- The reader sends r_r to the tag.
- The tag generates plain text P as follows:

  P = BYTE_MIX (r_r # $r_{t,1}$ # ID) where BYTE_MIX is a function that performs bit interleaving on P in order not to have large consecutive segments of P dominated by the reader or the tag and (#) denotes concatenation. The tag then transmits the message M as follows:

  $M = P^2 + r_{t,2}.n$

- The reader uses the private key to decrypt M. If the value of the challenge r_r is found, the reader will output the value of ID, else authentication will fail.

This protocol offers various security benefits including increased secrecy and privacy, robustness in the network (no private keys are saved on the tags), and no tags rewrites. When compared with other cryptographic hardware implementations, WIPR seems to be efficient in terms of mean current consumption (compared to ECC-192) and chip area (compared to SHA-1, SHA-256 and ECC-192). Nevertheless, this protocol is devised for active tags and cannot be adopted on the widely deployed cheap passive tags.

In [7], the authors review the EC-RAC protocol and Schnorr's protocol for authentication in RFID tags. Both protocols use the idea of a challenge-response by sending random numbers between the tag and the server. Both derived schemes could be compromised by an attacker placed between a communicating reader and tag. The attacker can add

previously sent data (by eavesdropping) to the currently sent one and by checking if the server accepts the forged messages, the attacker may deduce that the tag is the same as the one used in the previous session which will enable clandestine tracking. Finally the authors described their novel one-round search protocol which aims to identify a specific tag from a pool of tags while limiting the computational complexity. In this protocol, only one query message is sent between tag and reader. The authentication process is also dedicated since the server needs the private key (x_1) and public key (x_2) of a certain tag, and only the designated tag can properly respond to this query; since it has both x_1 and x_2. Also, this protocol provides protection against replay attacks since it uses a counter that will be updated each time a valid message is received. Both tag and reader have two counters, c_t and c_s respectively. Counter c_s is incremented each time the reader initiates a session with a given tag and c_t is incremented whenever the tag receives a valid message.

The protocols presented above seem to make inaccurate assumptions regarding the tag's processing power, as they require it to perform hash functions and random number generation, which are outside the scope of the tag's capabilities in most cases. In [8], [9], and [10] a family of ultra-lightweight protocols is presented that comply with the limited capabilities of RFID tags. In [8], Lopez et al. propose LMAP which is a lightweight mutual authentication protocol that can be implemented on the 3000 gates available for security purposes on passive tags. The protocol was deemed weak because it used AND and OR functions to build public sub-messages. This is considered a vulnerability since when using a bitwise AND or OR operation, even over random inputs, the probability of obtaining a zero (for AND operation) or one (for OR operation) is 3/4. In other words, the result will be strongly biased and can be spoofed. After discovering these weaknesses in LMAP, another ultra-lightweight protocol was discussed in [9]. SASI showed resistance to the security threats but the analysis in [11] showed that the immunity to tractability can be broken.

In [10], Lopez et al. present the most recent ultra-lightweight protocol and called it the Gossamer protocol. In this work, the tradeoff between tag price and security level of the proposed protocol is discussed; the higher the tag price the more secure the protocol is, but the less this protocol will be used as tags with higher prices are less common. Moreover, a new classification to RFID tags was described as follows: high cost tags and low cost tags. High cost tags are in turn divided into full-fledged tags that support advanced cryptographic functions and simple tags that support random number generation and one-way hash functions. Low cost tags are either lightweight tags that support CRC and random number generation or ultra-lightweight that support only bitwise XOR, AND, OR and other simple operations. So the most challenging task is to devise a protocol to secure ultra-lightweight tags, but the attacker model in this scenario is simplified to a passive attacker due to the severe restrictions presented by the tag circuitry. Moving on, the Gossamer protocol is described and it comprises three phases: tag identification, mutual authentication, and finally the updating phase. It is considered ultra-lightweight as it only uses addition, XOR, AND, rotation operator, and an additional operator called MIXBITS which actually consists of additions and right shifts.

In [12], a new approach to implement ultra-lightweight protocols is discussed and this is based on utilizing minimalistic cryptography such as Physically Unclonable Functions (PUF) and Linear Feedback Shift Registers (LFSR). PUFs and LFSRs are very efficient in hardware and particularly suitable for the low-cost RFID tags. The importance of PUF functions is that they exploit the inherent variability of wire delays and parasitic gate delays in manufactured circuits, and may be implemented with an order-of-magnitude reduction in gate count as compared to traditional cryptographic functions, and this is essential in ultra-lightweight tags.

4. Proposed Approach

After reviewing all the above protocols, we propose a protocol to solve the problem of mutual authentication in ultra-lightweight tags. During the initial phase PUF functions should be implemented on all the tags. This PUF function will be used to produce what will be called the secret value of the tag (SVT_i). Each tag will have a different secret value. The operation that produces SVT_i is as follows:

$$SVT_i = PUF\ (random\ number)$$

The random number used in this equation is not produced by the tag, instead it is provided from an external source and this is possible as it is done in the initial phase before deploying the tags. This condition is imposed to adhere to the limited capabilities of ultra-lightweight tags.

Additionally, another secret value relative to the reader (SVR_i) is stored in the tag and it is obtained as follows:

$$SVR_i = PUF\ (SVT_i)$$

This means that each tag will have a unique pair of SVT_i and SVR_i stored in it initially. The backend database will store these associated pairs for all the tags.

The tags for which this protocol is intended are ultra-lightweight; they only support simple operations such as bitwise AND, OR, and XOR operations in addition to right and left shifts. Operations like random number generation and hash functions are not supported by the tag. In the protocol, the operation Rot(x,y) is used which is a circular shift on the value of x by (y mod N) positions to the left for a given value of N [10]. The value of N is 96 bits as SVT_i and SVR_i are 96 bits. The protocol takes place as shown in Figure 1 where n_1 and n_2 are two 96-bit random numbers produced by the reader. The protocol is divided into three main phases and those are:

- Tag Identification: After the reader sends the "hello" message, the tag responds with its SVT_i . The reader will try to find the entry corresponding to this SVT_i in the backend database. If it succeeds, the mutual authentication phase starts. Otherwise, a new request is sent by the reader, but this time the tag will reply with the un-updated SVT_i to account for possible de-synchronization between the reader and the tag.
- Mutual Authentication: After identifying the tag, the reader will generate two 96-bit random numbers n_1 and n_2. Using these random numbers along with SVT_i and SVR_i , the reader produces the message A || B || C as shown in Figure 1. The tag will then use sub-messages A and B to find the random numbers n_1 and n_2. Then it computes C' and compares it with the version it received from the reader. If C≠C' the tag will stop the authentication procedure, otherwise the reader is authenticated because it has both SVT_i and SVR_i The tag then moves on to build sub-messages D, E, and F. The reader authenticates the tag by computing D' and comparing it to D in order to make sure that the tag was able to retrieve the correct random numbers.
- Updating Phase: After authenticating the reader, the tag will update SVT_i and SVR_i as shown in Figure 1 and it will also keep the old values stored to prevent de-synchronization. Upon receiving sub-messages E and F, the reader extracts the values of SVT_i and SVR_i and stores them in the backend database.

5. Security Analysis

- Mutual Authentication and Data Integrity: The protocol assures mutual authentication as only a legitimate reader will be able to build messages A, B, and C; while only a legitimate tag can build D, E, and F. Data integrity is assured as both the tag and reader construct local versions of sub-messages C and D respectively, and they compare them to the received versions.
- Tag Location Privacy: The tag responds with a unique SVT_i each time it is queried, and thus it cannot be linked to one SVT_i , thus making tracking unfeasible.
- Tag Impersonation Attack: Since each tag is identified uniquely by an identifier resultant from its own PUF function, tag impersonation attacks are rendered very hard to achieve. Even if an attacker clones a certain tag, it is not possible to achieve the same SVT_i as it is the output of the PUF function. Physical compromise of the tag is unaccounted for in this protocol, as the assumption in any ultra-lightweight protocol is that the threat model is a passive attacker.
- Reader Impersonation Attack: Only a legitimate reader has the correct mappings between SVT_i and SVR_i , thus an impersonating reader will not be authenticated by a legitimate tag.
- Replay Attack: As the protocol is a challenge response scheme that uses random numbers and a PUF with an update phase, replaying the same messages by an eavesdropper will lead to a failed authentication. To provide further immunity, the tag should store n_1 so that if old SVT_i and SVR_i pairs are used, in case of an unsuccessful update phase, the tag can be sure that the reader is legitimate by comparing the random number sent in this session to the one stored. If different random numbers were used, the reader would be authenticated; otherwise the tag would be able to detect the replay.
- Denial of Service Attack: If the attacker blocks the final message sent by the tag, de-synchronization occurs. This problem can be overcome by storing two versions of SVT_i and SVR_i at the tag, one couple before the update and the other after the update.
- Forward Security: Even if a tag is compromised at a later stage, the attacker cannot deduce any of the tag's previous interactions because during each session, freshly generated random numbers are used and each time a different input was used in the PUF function.

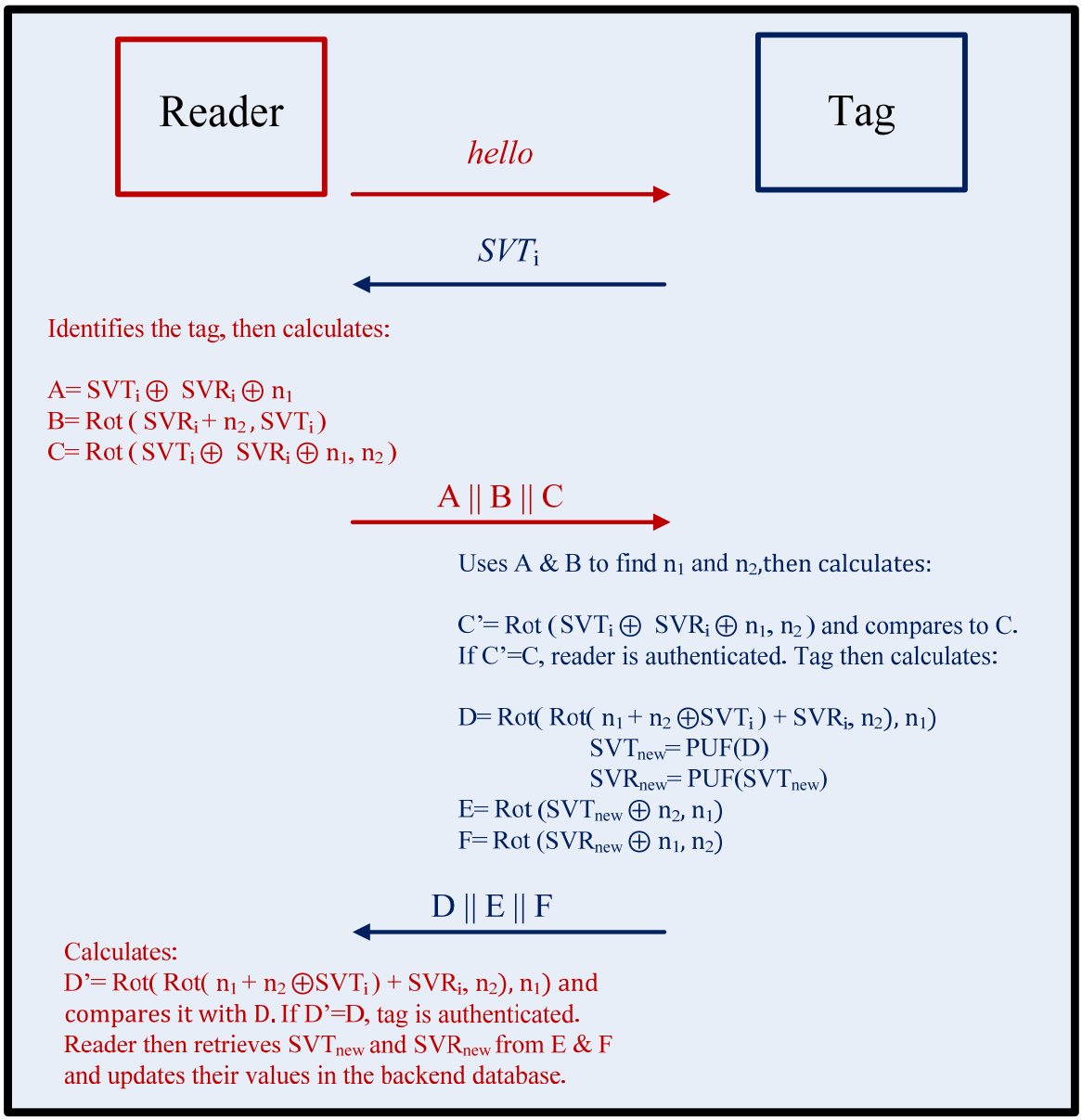

Figure 1. Proposed Protocol

6. Performance Analysis

The performance and storage requirements of the protocol are studied in this section.

- Computational Cost: All the operations used in this protocol are compliant with ultra-lightweight tags and can be very efficiently implemented in hardware. According to [13], the implementation of the PUF function requires six to eight gates for each input bit; thus a 96-bit PUF function will require at most 768 gates. This means that PUFs are a much better solution than using hash functions that require at least 3500 gates to be implemented. Furthermore, the rotation operation is also lightweight and can be implemented easily on ultra-lightweight tags.

- Storage Requirements: Each tag needs to store two couples of SVT_i and SVR_i, old and updated values, in addition to n_1. Each of these identifiers has a length of 96 bits in compliance with EPCGlobal. Additionally, the tag needs to store a maximum of five intermediate 96-bit values during the authentication phase. All of these values are stored in a rewritable memory because they change during different authentication sessions. So the total storage requirement on the tag is:

$$10*96 = 960 \text{ bits}$$

- Communication Cost: Assuming that the "hello" message is 5 bytes and knowing that messages A through F and SVT_i all have a length of 96 bits, the total communication cost of this protocol is:

7*96 +5*8= 712 bits

Table 1 compares the performance of the different ultra-lightweight protocols surveyed in this work.

Table 1. Performance Comparison

Protocol	Storage Req.(bits)	Communication Cost (bits)
Gossamer	960	520
LMAP	1056	520
SASI	864	520
Work in [12]	864	424
Proposed Work	960	712

The communication cost in this work could be significantly decreased if the update phase is performed separately in the tag and in the reader, but then the updated values would not involve the use of PUFs and hence there is a tradeoff between uniqueness of updated values and communication cost.

7. Implementation

In this section we describe a sample implementation of the security protocol described in Figure 1. We simulated the operation the RFID tag and reader using a client/server program developed in Java. In this simulation model the RFID tag plays the role of the client while the reader assumes the role of the server. The network interaction is realized using Java sockets that abstract a TCP client/server connection. The reader waits for tag connections on a specified IP address and port. Once an RFID tag successfully establishes a connection with the reader, the protocol steps, as specified in Figure 1, are executed resulting in a mutual authentication between the RFID tag and reader. We believe that this simulation model presents a viable proof of concept that demonstrates the correctness of the protocol before future deployment on real RFID tags and readers. The NetBeans v7.0.1 Integrated Development Environment (IDE) [14] is used for developing the client/server program using the Java 7 platform [15]. The server machine runs Ubuntu Linux v 9.04 and having the following hardware specifications:

- Processors: Intel(R) Core(TM) i7 CPU Q 720 running at 1.6 GHz
- Memory: 4GB RAM

The client machine runs Windows 7 with the following hardware specifications:

- Processors: Two Intel(R) Xeon CPUs running at 3.8 GHz
- Memory: 2 GB RAM

The snapshots presented in Figures 2 and 3 respectively demonstrate a sample execution of the RFID tag and reader protocol steps. The diagnostic messages provided by the client and server programs indicate the successful execution of the different protocol steps and the values of the parameters involved in the mutual authentication procedure. To test the validity of the protocol exchanged messages and their full compliance with the protocol specifications; we deployed the Wireshark network protocol analyzer [16] on the server machine to capture the inbound and outbound protocol packets exchanged by the server network interface. Figure 4 shows the details of the "*hello*" packet sent from the reader to the tag as captured by Wireshark. Figure 5 represents the tag to reader message containing the value of "*SVT0*". The Wireshark packet representation of Figures 6 demonstrates the message containing the values of the *A*, *B*, and *C* parameters sent from the reader to the tag. Similarly, the packet representation of Figure 7 indicates the message containing the values of the *D*, *E*, and *F* parameters sent from the tag to the reader.

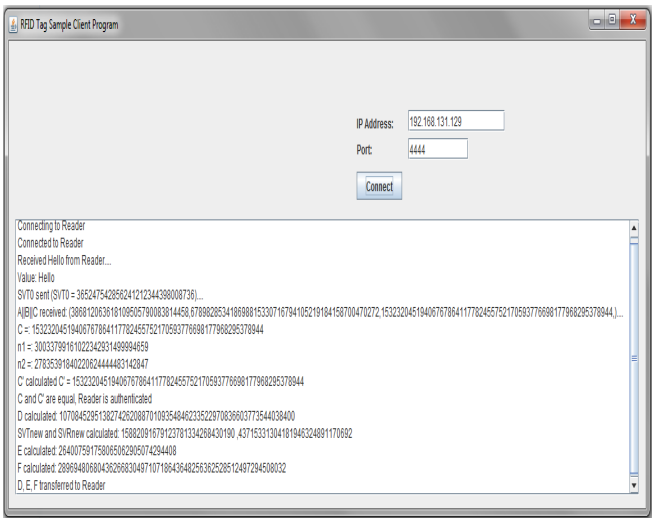

Figure 2. Client-side program executing the tag protocol.

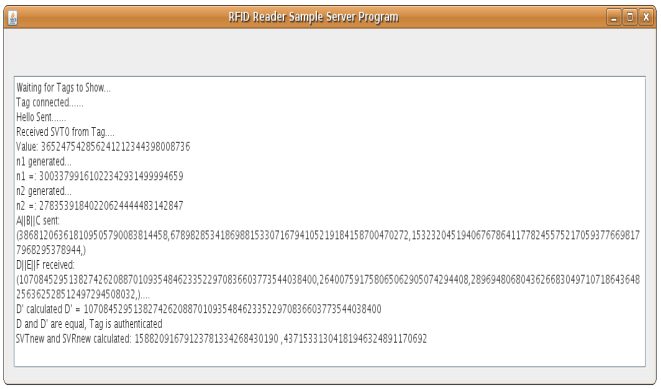

Figure 3. Snapshot of the server-side program executing the RFID reader protocol steps

By comparing the parameter values displayed by the tag and reader diagnostic messages (see Figures 2 and 3) and the packet contents captured by Wireshark (see Figures 4, 5, 6, and 7) we realize that the protocol sample implementation demonstrates a 100% compatibility with the protocol specification.

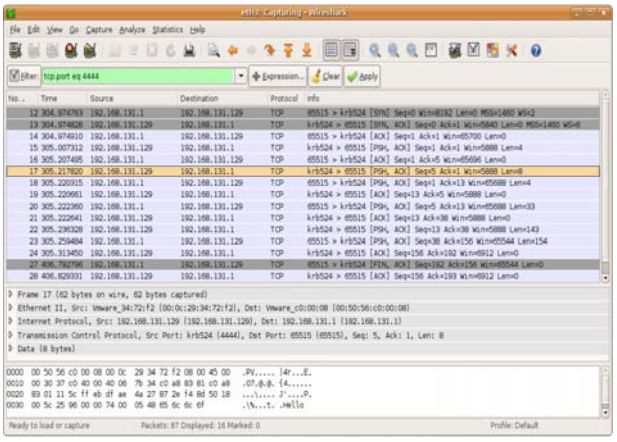

Figure 4. Wireshark capture showing the "hello" message

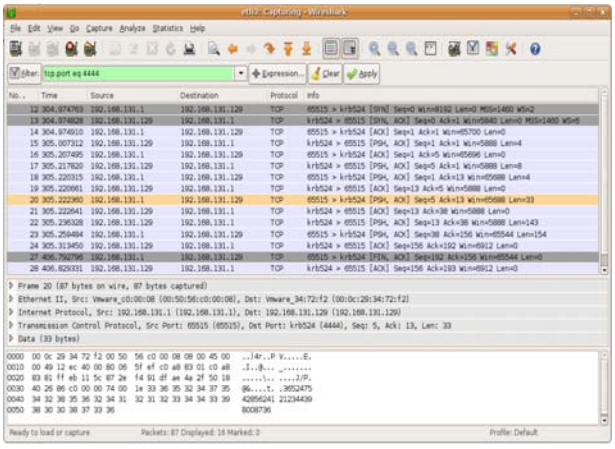

Figure 5. Wireshark capture of the packet representing the "SVT0" message from the tag to the reader

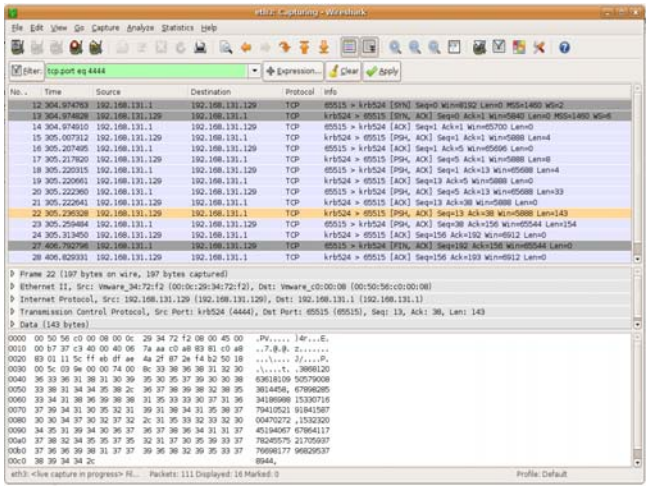

Figure 6. Wireshark capture of the packet representing the "A||B||C" message from the reader to the tag.

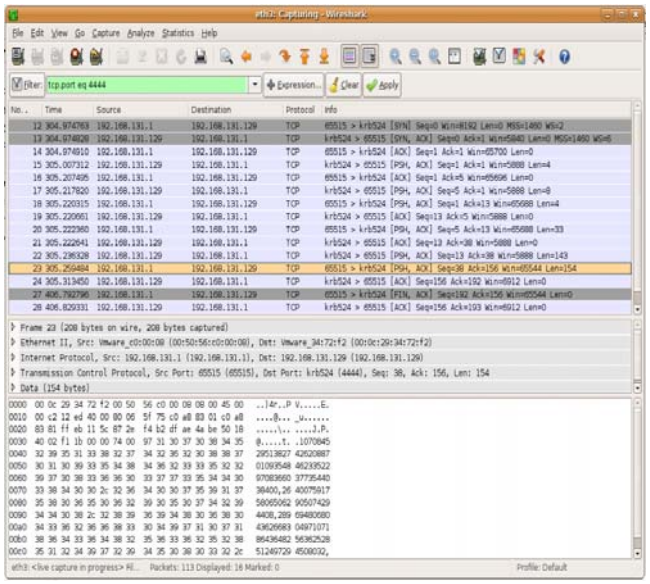

Figure 7. Wireshark capture of the packet representing the "D||E||F" message from the tag to the reader

8. Conclusion

This paper surveyed RFID systems in general stating their architecture and applications. Then, some recently proposed RFID security protocols were presented while focusing on the ultra-lightweight family as those comply with the limited processing power of the most commonly used passive tags. Building on that, a protocol that achieves mutual authentication for ultra-lightweight tags was proposed. The protocol comprises three main stages: tag identification, mutual authentication, and an update phase. It uses light operations and a PUF circuit that only requires about 3000 gated to be implemented and dedicated for security on passive

tags. Finally, the security and performance of the proposed protocol were analyzed leading to the conclusion that the protocol offers immunity against a broad range of attacks while having an excellent performance.

9. References

[1] Garfinkel, S.L.; Juels, A.; Pappu, R.; , "RFID privacy: an overview of problems and proposed solutions," *Security & Privacy, IEEE* , vol.3, no.3, pp. 34- 43, May-June 2005.

[2] Boyeon Song and Chris J Mitchell, "RFID Authentication Protocol for Low-cost Tags," *WiSec*, Alexandria, Virginia, March 31–April 2, 2008.

[3] Juels, A.; , "RFID security and privacy: a research survey," *IEEE Journal on Selected Areas in Communications,* vol.24, no.2, pp. 381-394, Feb. 2006.

[4] Wehbe, S.; Kayssi, A.; Chehab, A.; Elhajj, I.; , "Mutual Authentication Scheme for EPC Tags-Readers in the Supply Chain," *3rd International Conference on New Technologies, Mobility and Security (NTMS), 2009*, pp.1-5, 20-23 Dec. 2009.

[5] Shaoying Cai, Yingjiu Li, Tieyan Li, Robert H. Deng, "Attacks and Improvements to an RFID Mutual Authentication Protocol and its Extensions", *WiSec*, Zurich, 2009.

[6] Yosef Oren, Martin Feldhofer, "A Low-Resource Public-Key Identification Scheme for RFID Tags and Sensor Nodes", *WiSec,* Zurich, 2009.

[7] Y. K. Lee, Lejla Batina, Dave Singelée, Ingrid Verbauwhede, "Low-Cost Untraceable Authentication Protocols for RFID", *WiSec,* New Jersey, 2010.

[8] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M.E. Tapiador, and Arturo Ribagorda, "LMAP: A Real Lightweight Mutual Authentication Protocol for Low-cost RFID tags".

[9] Hung-Yu Chien; "SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity," *IEEE Transactions on Dependable and Secure Computing,* vol.4, no.4, pp.337-340, Oct.-Dec. 2007.

[10] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M.E. Tapiador, and Arturo Ribagorda, "Advances in Ultralightweight Cryptography for Low-Cost RFID Tags: Gossamer Protocol", *Springer,* pp. 56-68, 2009.

[11] Phan, R.C.-W.; "Cryptanalysis of a New Ultralightweight RFID Authentication Protocol—SASI," , *IEEE Transactions on Dependable and Secure Computing*, vol.6, no.4, pp.316-320, Oct.-Dec. 2009.

[12] Kulseng, L.; Zhen Yu; Yawen Wei; Yong Guan; , "Lightweight Mutual Authentication and Ownership Transfer for RFID Systems," *IEEE INFOCOM, 2010* , pp.1-5, 14-19 March 2010.

[13] Bolotnyy, L.; Robins G.; "Physical Privacy and Security in RFID Systems", *Auerbach Publications,* 2009.

[14]The NetBeans homepage: http://www.netbeans.org.

[15] The Java homepage: http://www.oracle.com/us/technologies/java/

[16] The Wireshark network protocol analyzer homepage: www.wireshark.org

