Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2014) 22: 155 – 165

doi:10.3906/elk-1201-77

Increasing key space at little extra cost in RFID authentications

Gökhan DALKILIÇ*, Mehmet Hilal ÖZCANHAN, Hafize Şen ÇAKIR
Department of Computer Engineering, Dokuz Eylül University, Tınaztepe, İzmir, Turkey

Received: 24.01.2012 • **Accepted:** 14.09.2012 • **Published Online:** 20.12.2013 • **Printed:** 20.01.2014

Abstract: Traditional authentication and key establishment protocols utilize nonce parameters as a means for message freshness, recent aliveness, and key derivation. Improving identity verification, increasing key space, or making secret updates more complex through nonces are not goals. Generating random numbers as nonces and not making the most out of them can be considered as a loss in resource stricken radio frequency identification (RFID) tags. By increasing the shared secrets slightly, a new functionality for the nonces is introduced, which makes the authentication and key establishment protocols of RFID systems more secure, in general. The proposed method contributes to the security of communication channels by increasing the key space. Attaining better security, with just a slight increase in the shared secrets and the already generated nonces, is beneficial compared to the existing costly, resource-demanding security primitives.

Key words: Nonce, RFID, mutual authentication, key establishment, tag

1. Introduction

One of today's popular trends is using ubiquitous computing to put multiple, diverse computational devices and systems into communication simultaneously. Radio frequency identification (RFID) is becoming one of the core technologies of ubiquitous systems as they are getting integrated into everyday objects. In addition to the identification property of traditional barcodes, RFID systems provide authentication. Basically, RFID systems consist of a database server, a reader, and a tag. The tag is the weakest point as the least resourceful component residing out in the field.

This attracts adversaries, who usually exploit the weakest point to penetrate the systems. The same powerful attacks that target computers are launched at the tags, as well. Therefore, the tags have to be equally provisioned to resist attacks, just like computers. To prevent attacks, many security protocols [1,2] have been devised. The protocols attempt to create a secure communication channel through an insecure environment by encrypting the messages. For encryption, a key known by the principals only is needed.

Key establishment (derivation of a common key by the principals) and authentication are the building blocks of creating secure communication channels. Authentication is simply verifying that the opposite peer (principal) is really the one that it claims to be [3,4]. RFID authentication is usually coupled with key establishment and shared secret updating. The established key is used to encrypt the rest of the exchanges. However, in recent years, intruders have stepped up their clandestine activities of capturing keys [5]. Meanwhile, the providers are trying to improve security against known [3] and newly formulated attacks [6]. To prevent attacks on keys, hence exposing the exchanged information, pseudorandom numbers called nonces are generated

*Correspondence: dalkilic@cs.deu.edu.tr

[3,4]. The generation of pseudorandom numbers is not free of cost; it requires energy and hardware and it consumes die area. Therefore, not using them to further reinforce authentication security can be a loss.

The lifetime and strength of today's security algorithms depend mostly on the key length. Another technique for prolonging the protocol lifetime is the increase of the key space by running the algorithm multiple times as in 3DES, or by devising a new algorithm with a longer key length, as in the advanced encryption standard (AES). Our proposed scheme however, increases the key space and provides varying keys in every run by increasing nonce functionality through an increase in shared secrets. This contribution has a better cost-performance ratio than a new protocol with a longer key length or running an algorithm many times.

In the rest of this paper, the general form and components of mutual authentication and key establishment in RFID is described in Section 2. In Section 3, our proposed method is described and demonstrated in 3 cases. Section 4 has an overall performance evaluation and security analysis of the cases, after the application of our proposed method. In Section 5, we conclude.

2. Authentication protocols using symmetric keys

The general form of a mutual authentication and key establishment process is shown in Figure 1. The initiator 'A' starts the exchange by sending its first set of parameters to challenge responder 'B' and negotiates the algorithms that will be used (if not predecided). In the second step, B answers by sending its own parameters and calculated authenticator. Hence, B tries to prove to A that it is alive, is really B, and agrees on the algorithms that will be used. In the third step, A verifies its own identity and presence. At the end, A and B are mutually authenticated. Usually, there is no fourth step, but some protocols have one to synchronize

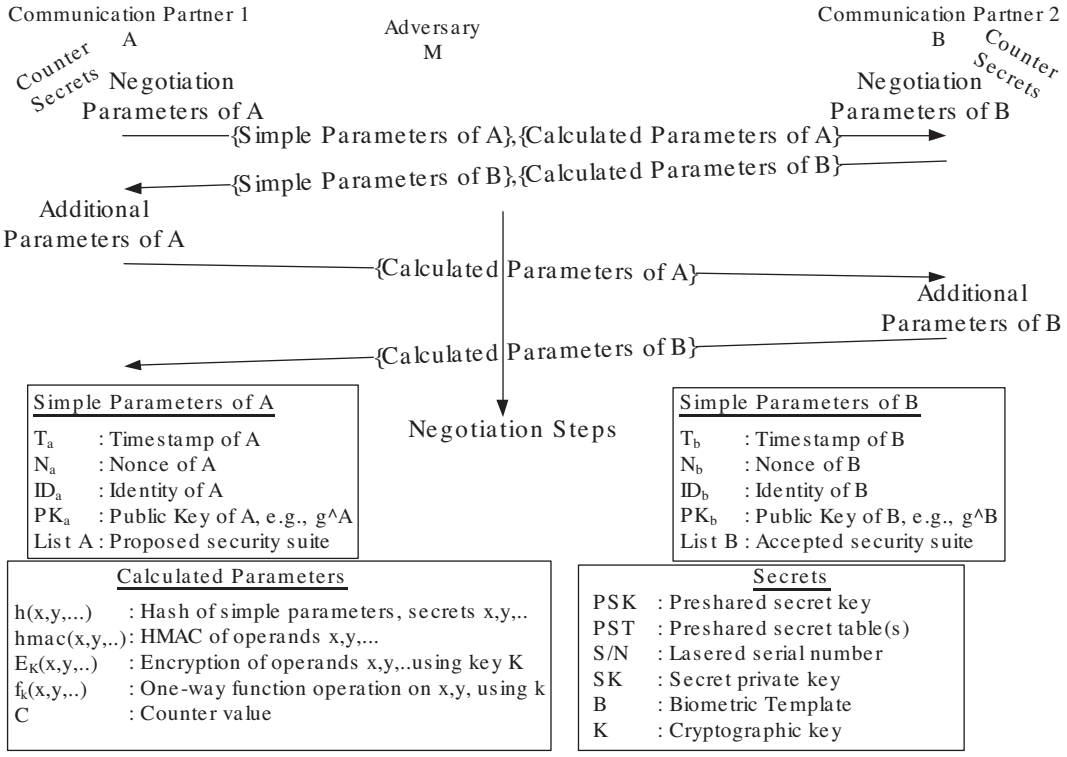

Figure 1. Authentication and key establishment in general.

updating. As can be observed, after the initial exchanges, the principals perform calculations to verify that the opposite has the preshared secrets and to reach a common key. If the same key is used by both sides, the protocol is said to be symmetric, leading to the terms 'symmetric key' and 'symmetric encryption'. Symmetric key encryption is widely used in RFID technology.

The principals have one or many preshared secrets and a function for encrypting or hashing the authenticators [4]. Preshared secrets can be fixed or variable size, lasered into the integrated circuit [7], which never leave the execution core, or biometric values that specifically belong to a person [8]. All of these features are an effort to ensure the confidentiality of the information passed. Unfortunately, apart from decryption, tampering, and side channel attacks, analysis of electromagnetic radiation and power traces are also used for exposing the encryption key. The adversaries try to exploit any weakness to capture the exchanged information [9].

Many weaknesses have been demonstrated in protocols like short key length, an inadequate number of inputs, limited complexity, and flawed equations. Weaknesses appear because sometimes the protocols are not fully analyzed prior to their release. Nowadays, efforts are focused on discovering hidden weaknesses. An example was given in [10], which used the same key throughout the lifetime of the device with no strategy of changing it. This property encourages brute force attacks [3]. Improvements are offered, but sometimes they cost energy, memory space, and computations that overwhelm RFID tags [10]. Our effort, however, increases keys at the cost of little extra memory.

3. Increasing the key space

Nonces are generated for proving message freshness, recent aliveness, and key derivation, and as the input to the hash or encryption of information (Figure 1). In key derivation, the participation of both principals' nonces is advised [3,4]. Our proposed scheme, shown in Figure 2, obeys this recommendation. The least significant bits (LSBs) of the nonces and the XORed nonces are used to obtain a pointer ($p(N_a)$, $p(N_b)$, $p(N_a \oplus N_b)$) to one of the preshared secrets. Randomly pointed secrets are used as inputs or keys to encryption/hashing

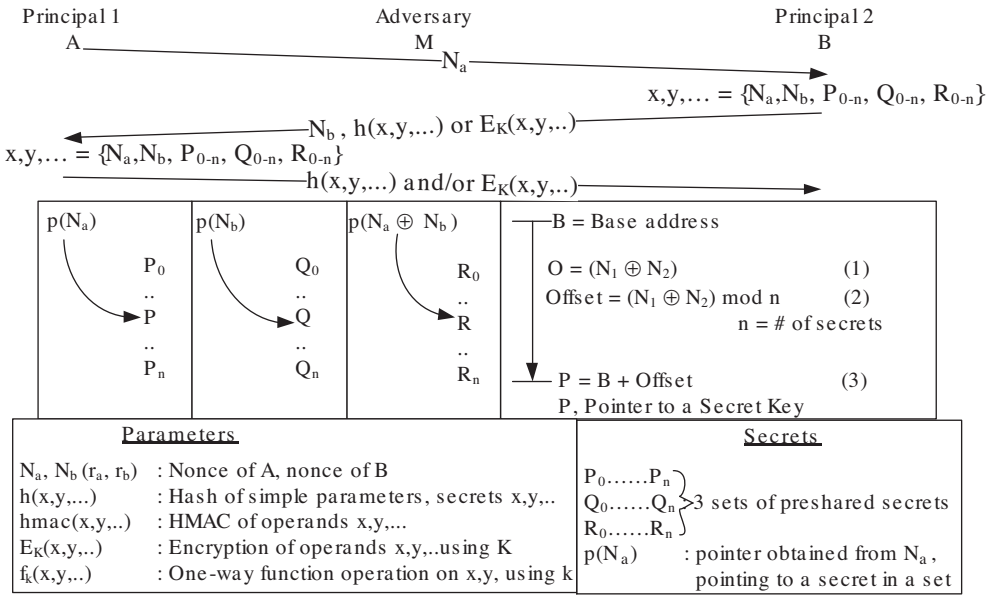

Figure 2. Proposed authentication primitives.

functions, making authentication unpredictable to outsiders. In summary, our proposal increases key space by increasing shared secrets. Our proposal can be applied to most symmetric key RFID authentication protocols. Three different protocols have been chosen to demonstrate the application of our proposal. First, a protocol using a constant key is studied. The second is a 2-step protocol that uses no nonces but only encryption. The third is a recently attacked protocol.

3.1. Case 1: a protocol using nonces and a hash function

The work in [11] is a recent RFID authentication protocol utilizing a hash function, given in Figure 3. The protocol is intended for ultra-light, low-cost tags, but the work in [12] put it in the light category due to the hash function and the nonce generator. The nonces are represented as N_1 and N_2. The most important property is the use of the same secret key k and its hash value in all of the tags, which is attacked by the work in [13]. Tracking, impersonation, and denial of service (DoS) attacks are demonstrated, based on a h(k) value compromised by tampering with any tag. However, we demonstrate another attack below, where h(k) is compromised without the need for physical tampering. The protocol is then made resistant to the described attacks using our proposal.

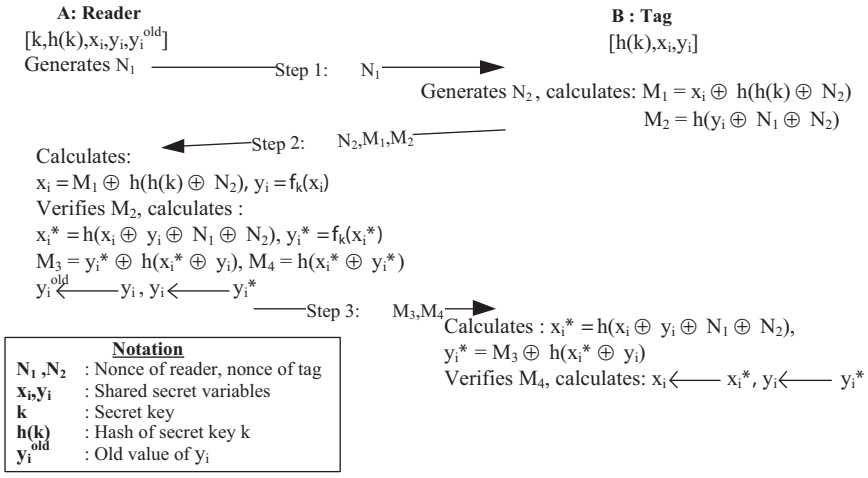

Figure 3. Protocol with the same secret key k for all tags [11].

The attack is launched by sending nonce N_1 to the tag, waiting for the tag's response, and then blocking the reader's messages, M_3 and M_4. This can be repeated unlimited times because the tag answers every challenge. Many nonces and related M_1^x, M_2^y pairs are recorded, where x and y denote the repetition number. XORing 2 repeated M_1 messages gives $M_1^x \oplus M_1^y = h(h(k) \oplus N_2^x) \oplus h(h(k) \oplus N_2^y)$. Simply denoting h(k) as a constant z, since it is so, we have numerous $Q = h(z \oplus N_2^x) \oplus h(z \oplus N_2^y)$-type equations. The Q, N_2^x, and N_2^y values are known and z is the only unknown. It is only a brute force attack to find a value for z that satisfies the equations, which becomes trivial for $N_2^x \cong N_2^y$ or $N_2^x \cong \mathrm{NOT}(N_2^y)$ pairs, as Q tends to be zero or all ones in these cases.

The same argument is true for message M_2, where y_i can be similarly deduced. Having exposed h(k), x_i can be calculated from M_1. We will stop short of claiming the capture of secret k, assuming that the one-way function in $y_i = f_k(x_i)$ is truly resistant to brute force attacks. Otherwise, a full disclosure attack is possible.

3.1.1. The proposed changes

The application of our proposed method for the same protocol is shown in Figure 4. There are 2 new security primitives. First is the manufacturer's lasered, unique ID (or S/N) that resists tampering and never leaves the execution core. Second is a systematic scheme of creating pointers from nonces and randomly selecting one of the available secrets. The LSBs of both nonces N_1 and N_2 are used to form pointers (Eqs. (1), (2), and (3) of Figure 2) that help to choose 3 secrets from the sets $\{p_0, p_1, ..., p_n\}$, $\{R_0, R_1, ..., R_n\}$, and $\{Q_0, Q_1, ..., Q_n\}$, and a key from $\{k_0, k_1, ..., k_n\}$. The number of LSBs used depends on n, e.g., for 8 secrets, 3 LSBs are used. Both nonces are used to choose p_x, $k_{N1 \oplus N2}$, and $R_{N1 \oplus N2}$ to prevent an adversary from sending the same nonce and forcing the use of the same keys or secrets. The unique ID of the tag is rotated left by an amount of $(p_x \oplus N_2)$ mod 16, since the word length (L) of an EPCglobal class-1 generation-2 standard tag is 16 bits. The obtained values $P_{N1, N2}$, and $k_{N1 \oplus N2}$ improve M_1 and M_2. Similar techniques are used in the update of the parameters x_i, M_3, and M_4.

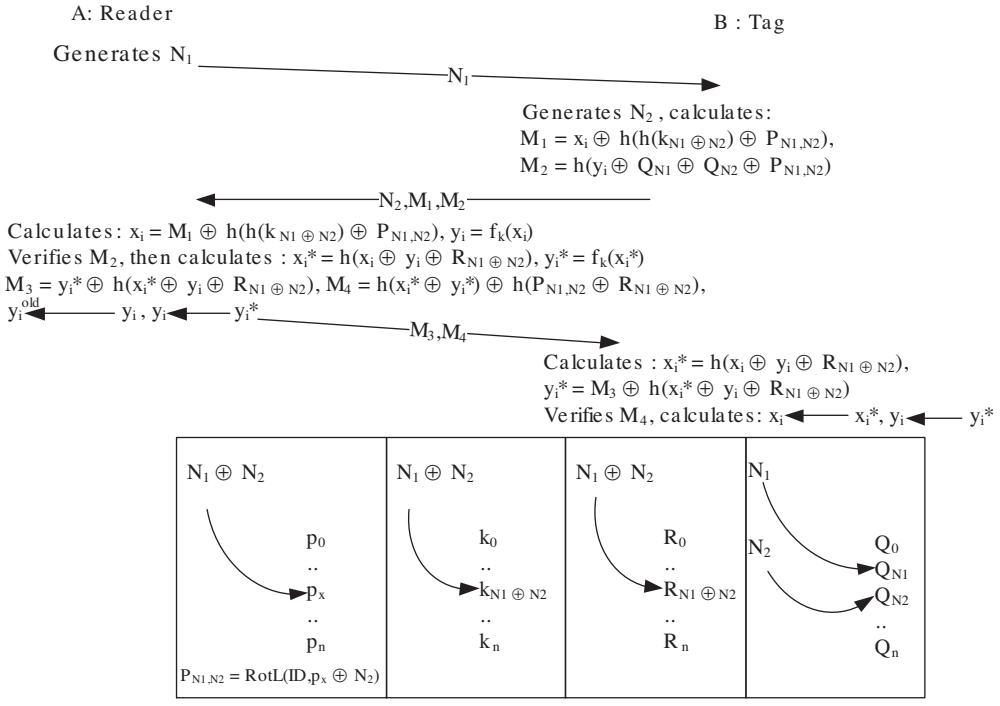

Figure 4. Proposed protocol in case 1.

Deducing h(k), x_i, or y_i from M_1, M_2, M_3, and M_4 is no longer possible. Neither the attacks described in [13] nor our above attack succeeds because of the presence of the terms $P_{N1, N2}$, Q_{N1}, Q_{N2}, $R_{N1 \oplus N2}$, and $k_{N1 \oplus N2}$.

3.2. Case 2: a protocol using no nonces, but only encryption

Feldhofer's work in [14,15] has shown that AES is going to be feasible in RFID tags in the future. A protocol using only AES-encrypted parameters was proposed in [10]. The protocol shown in Figure 5a uses 2 preshared secrets k_1 and k_2, a constant cryptographic secret key k, and a unique ID_k. No nonces are used and a collision detection protocol is assumed to singularize tag T out of many.

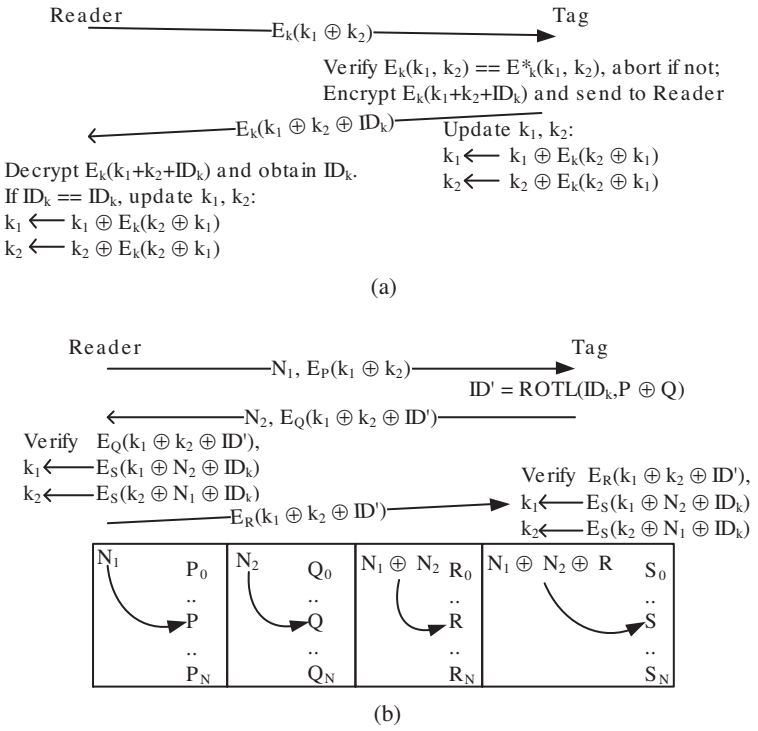

Figure 5. a) Flawed protocol of the work in [10] in case 2, b) Our proposed alternative in case 2.

First, the update flaw in [10] is shown, where the shared secret k_1 is updated as $k_1 = k_1 \oplus E_k(k_2 \oplus k_1)$. Actually, $E_k(k_2 \oplus k_1)$ equals $E_k(k_1 \oplus k_2)$ of the first message. Naming it as X, (new)$k_1 = k_1 \oplus X$ and (new)$k_2 = k_2 \oplus X$. With the new k_1 and k_2 values in the next round, the first message $E_k(k_1 \oplus k_2)$ becomes $E_k(k_1 \oplus X \oplus k_2 \oplus X)$. This updated message reduces to the same original value $E_k(k_1 \oplus k_2)$.

In addition, a 'nonceless' protocol is vulnerable to known attacks like DoS, man-in-the-middle, and exhaustion attacks. The works in [16,17] attacked the work in [10] and offered alternative protocols with improvements. The authors in [17] claimed that the work in [10] has traceability and that the work in [16] points out the danger of using a constant key, at all times, as in [10]. Both corrective works consist of 4 steps and use nonces with other security primitives. The work in [17] improved the protocol, but it also used a constant key. If the boundaries and padding of the concatenated secrets are not properly designed as described in [18], the protocol suffers a traceability flaw, because the encrypted $K_{pub}(ID)$ of [17] is detectable. The work in [16] offered better security with a scheme for changing the encryption key but lacked forward secrecy, as all of the keys were sequentially dependent on the previous. Both protocols have the same weakness of using a constant to XOR the first nonce sent to the tag. The work in [17] made 2 pseudorandom number generator (PRNG), 7 AES, and 1 XOR operations. The work in [16] made 1 PRNG and 6 AES but 4 XOR operations. Our method is applied to the work in [10] below and it is demonstrated that known weaknesses are removed with little overhead.

3.2.1. The proposed changes

The insertion of multiple secrets with 2 nonces into the work in [10] is shown in Figure 5b. The principals have 4 sets of preshared secret keys. A pointer is generated using N_1, as in Figure 2, to choose a key P (Figure

5b). A variable key, instead of the previous constant key k, is used in $E_P(k_1 \oplus k_2)$. Even with a repeated N_1, $E_P(k_1 \oplus k_2)$ produces a completely different result due to the strong avalanche effect of AES, on different k_1 and k_2 or p. With $N_1 \oplus N_2$, other keys Q, R, and S are chosen. In our proposal, the ID_k of the tag is obscured by ROTL(ID_k,P $\oplus$ Q), where ID_k is rotated left by an amount of (P $\oplus$ Q) mod 16. Next, it is encrypted and sent in $E_Q(k_1 \oplus k_2 \oplus$ ID'). After authenticating the tag, the reader updates k_1 and k_2 first. A third step is necessary to inform the tag of the update and avoid desynchronization. Therefore, $E_R(k_1 \oplus k_2 \oplus$ ID') is used. The encryption key S is obtained using $N_1 \oplus N_2 \oplus$ R. The updates go through AES operations $E_S(k_1 \oplus N_2 \oplus ID_k)$ and $E_S(k_2 \oplus N_1 \oplus ID_k)$, ensuring new values. Our protocol takes 3 steps, 1 PRNG, 5 AES operations, 1 rotation, and 10 XOR operations, i.e. less computation than in [16,17].

3.3. Case 3: a protocol attacked recently

The third example is a protocol that claims to offer strong security at a low cost [19], using a hash function as the only cryptographic primitive. Both the protocol and the presented security analysis were formally analyzed in [20], using just a hash function with standard techniques yet claiming that strong security is refuted with detailed demonstrations of replay, traceability, and desynchronization attacks. The contribution of our proposed method would be clearly proven if it removes the vulnerabilities of the protocol and makes it resistant to the attacks described in [20].

The proposed modifications are shown in Figure 6, as 3 sets of secrets and pointers to the secrets. Using the XOR of both nonces nr and nt, 3 secrets are pointed at. One of these secrets is used to add an authenticator h(ID, $P_{nr \oplus nt}$) in the second step. $P_{nr \oplus nt}$ is obtained by rotating secret $p_{nr \oplus nt}$ by an amount of (ID $\oplus$ nt) mod 16. The reader finds the tag in the database using h(ID). Using the nonces, the reader also obtains $P_{nr \oplus nt}$ and verifies the authenticator h(ID, $P_{nr \oplus nt}$). Next, the reader updates its secrets and passes 2 messages, h($Q_{nr \oplus nt}$) and h(ID', $S_{nr \oplus nt}$), to the tag. The tag verifies the reader through h($Q_{nr \oplus nt}$) and is now informed by h(ID', $S_{nr \oplus nt}$) that the reader's update has finished. Finally, the tag updates to ID = h(ID', $S_{nr \oplus nt}$) as well.

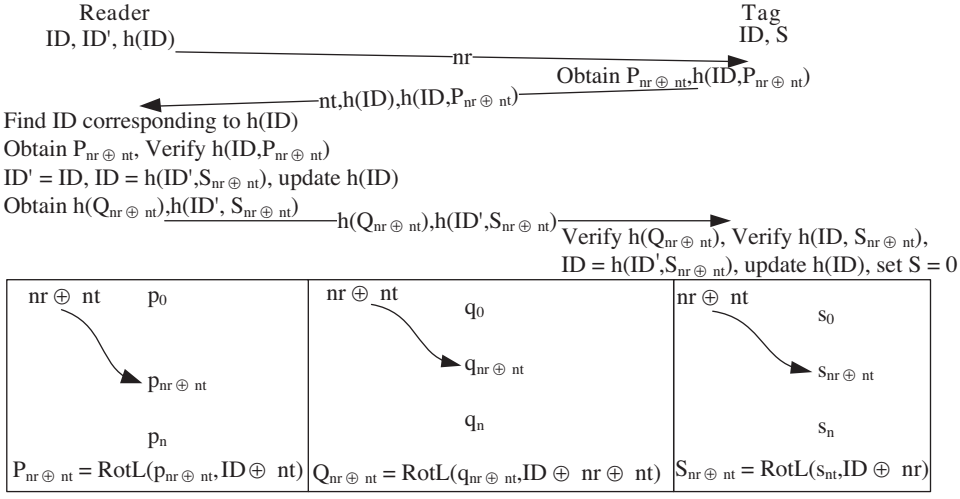

Figure 6. Proposed modified protocol in case 3.

The first attack launched in [20] was possible because the challenge and response of [19] were not related. Thus, an active attacker sends a nonce to the tag and gets a response that can be replayed later. In our improved

version, however, the response depends on the challenge through the $P_{nr \oplus nt}$ term of the authenticator h(ID, $P_{nr \oplus nt}$), which depends on both nonces. Moreover, the reader can verify if the nonce nt has changed on the way. Therefore, the modification of nt always results in a rejection of the tag's response by the reader. Since there is no S defined in the final step of our protocol, the learning and challenge phases of the traceability attack described in [20] cannot be launched.

The desynchronization attack in [20], based on the modification of nr, cannot go by unnoticed in our protocol. Any modification results in a mismatch with h(ID, $P_{nr \oplus nt}$) at step 2. The S bit defined in [20] is turned off at the end of the protocol, but still a mechanism is needed to test S in the next run. For preventing desynchronization attacks, the steps and transitions of the exchanges must be monitored by both sides. To achieve this goal, a tag with a rudimentary state machine is essential.

4. Performance evaluation and security analysis

The memory cost of our proposed method is shown in the Table, where the word length (L) is 16 bits. As an example, assuming 4 sets of shared secrets and keys (shown as P, Q, R, and k previously), each with 4 members, a total of 16 L or 32 bytes of memory is needed (line 2 of the Table). Two LSBs of the nonces are used. If each secret is eligible to be used and rotated by L, a maximum of $4^4 \times$ 16 different keys are possible. If AES-128 is going to be used, 8 keys must be merged. Using the table in [21], the average time required for an exhaustive key search is 5.4×10^{18} years at 10^6 decryptions/μs for a key length of 128 bits. In general, the time required after applying our proposal is (no. of keys) $\times$ 5.4×10^{18} years. Thus, our method is $(4^4 \times 16 - 1) / 8 \times 5.4 \times 10^{18}$ years more resistant in case 2, which has a single key. On the other hand, the additional number of operations for obtaining pointers from nonces is at most 6, e.g., XORing, ANDing, ORing, rotating, loading an index pointer, and fetching a secret.

Table. Memory cost and performance.

No. of LSBs (bits)	No. of set members	Total memory cost (L)	Memory cost in (bytes)	No. of keys	No. of operations
1	2	8	16	16×2^4	6
2	4	16	32	16×4^4	6
3	8	32	64	16×8^4	6
4	16	64	128	16×16^4	6

Running the 3 original protocols studied through the authentication verification tool AVISPA [22] revealed their weaknesses. However, the same tool shows no attacks for our modified protocols. The results can be reached through the link in [23]. Those attacks that cannot be checked by the tools have been detailed below.

4.1. Prevention of replay and reflection attacks

The role of the nonces is to provide message freshness and recent aliveness to the challenges and responses of an exchange [24], thus preventing replay and reflection attacks. Since nonces are added and not removed in our proposal, the security in the 3 cases increases. Moreover, our scheme forces the principals to pick the correct shared secret or key depending on the exchanged nonces; therefore, the security is further increased. Because an authenticator is calculated with different terms in every run, an adversary cannot fool either principal by simply replaying or reflecting a previously formed authenticator. This is a clear improvement in security.

4.2. Prevention of DoS and desynchronization attacks

There is no way of preventing DoS attacks simply with nonces, since the tags are stateless. For example, a tag will always reply to a challenge. Other complementary features like keeping counters or finite state machines are needed for the tags. However, exhaustion attacks that force the principals to go into overwhelming computations with repeated bogus challenges are resisted in our proposal, because a challenge is responded to by an authenticator formed with correctly chosen preshared secrets. A false authenticator halts the exchange and prevents exhaustive computations.

Desynchronization occurs when the tag and the reader are tricked into a state where they update their preshared secrets into different values. The update occurs at the end of the protocol and has to be verified. Omitting the verification of the update is the reason of the weakness. The reader should update first and inform the tag by sending a message with the updated values, to guide the tag to also update. This will remove the necessity of keeping old values on the tag. This is precisely the reason of the presence of a third step message in our proposals, obtained from the updated values.

4.3. Prevention of man-in-the-middle attacks

This type of attack is very efficient if complemented by a tampering attack. An adversary who acquires all of the shared secrets by physically tampering with the tag can intervene in the exchange and falsify a transaction without cloning the tag. Therefore, we propose the use of a secret ID lasered into a tag, whose value is given to the end user written on a paper by the manufacturer but never leaves the execution core. Rotating this unique ID depending on the nonces makes it a variable in the authenticator. Thus, this type of attack is resisted.

4.4. Detection of integrity loss and key lifetime

Any authenticator–nonce mismatch is detected in our proposed method. Therefore, the integrity of the nonces and the authenticators is guaranteed. Loss of exchange integrity is only possible if an attacker's bogus nonces perfectly match the chosen secrets, a far possibility.

The key's life-time is an important property of the encryption and hashing functions of the authentication process. Using the same key at all times in all devices is not desirable, as the compromise of the key means the loss of all of the past communications. Therefore, short key lifetimes are desirable. This property is provided in our proposal by the rotation and changing of the secrets and keys used in every run.

5. Conclusion

A new approach towards increasing the key space in RFID authentication and key establishment protocols has been introduced. The approach reinforces protocols in general and does not cause vulnerabilities. The 3 examples have demonstrated that the new approach makes protocols more resistant to known attacks. Our proposal is limited to deriving different keys and obtaining dynamic results. However, the idea can lead to other security enhancements like changing the order of operations according to the nonces. Although our proposal is straightforward and adds incremental security, it only costs a little extra memory space and 6 additional operations. This can be considered a good price/performance tradeoff for resourceless tags. A slightly larger memory and simple operations are less resourceful than the demanding security primitives of encryption and hashing.

Acknowledgment

This study was partially supported by DEU BAP Project 2009.KB.FEN.044.

References

[1] C. Boyd, A. Mathuria, Protocols for Authentication and Key Establishment, Berlin, Springer, 2003.

[2] G. Avoine, "Bibliography on security and privacy in RFID systems", Information Security Group, Louvain-La-Neuve, Belgium, Universite Catholique de Louvain, pp. 1–19, 2011.

[3] A. Menezes, P. Oorschot, S. Vanstone, Handbook of Applied Cryptography, Boca Raton, FL, USA, CRC Press, 1996.

[4] E.M. Ng, "Security models and proofs for key establishment protocols", MSc Department of Mathematics, University of Waterloo, Canada, 2005.

[5] I. Paul, "Nasty new worm targets home routers, cable modems", PC World, 2009, available at http://www.pcworld.com/article/161941/nasty_new_worm_targets_home_routers_cable_modems.html, last accessed 27 August 2012.

[6] J. Leyden, "Old worm learns conficker tricks", The Register, 2009, available at http://www.theregister.co.uk/2009/04/06/old_worm_adopts_conficker_tricks/, last accessed 27 August 2012.

[7] Dallas Semiconductor of Maxim Integrated Products Inc., "Protecting the R&D investment: two-way authentication and secure soft-feature settings", Application Note 3675, 2005.

[8] M.J. Atallah, K.B. Frikken, M.T. Goodrich, R. Tamassia, "Secure biometric authentication for weak computational devices", Proceedings of the 9th International Conference on Financial Cryptography and Data Security, pp. 357–371, 2005.

[9] P. Shaumont, K. Tiri, I. Verbauwhede, "Securing embedded systems", IEEE Security & Privacy, Vol. 4, pp. 40–49, 2006.

[10] B. Toiruul, K. Lee, "An advanced mutual-authentication algorithm using AES for RFID systems", International Journal of Computer Science and Network Security, Vol. 6, pp. 156–162, 2006.

[11] Y. Liu, "An efficient RFID authentication protocol for low-cost tags", IEEE International Conference on Embedded and Ubiquitous Computing, Vol. 2, pp. 180–185, 2007.

[12] H.Y. Chien, "SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity", IEEE Transactions on Dependable and Secure Computing, Vol. 4, pp. 337–340, 2007.

[13] I. Erguler, E. Anarim, "Attacks on an efficient RFID authentication protocol", 10th IEEE International Conference on Computer and Information Technology, pp. 1065–1069, 2010.

[14] M. Feldhofer, J. Wolkerstorfer, "Strong crypto for RFID tags – a comparison of low-power hardware implementations", IEEE International Symposium on Circuits and Systems, pp. 1839–1842, 2007.

[15] M. Feldhofer, J. Wolkerstorfer, "Hardware implementation of symmetric algorithms for RFID security", in: P. Kitsos and Y. Zhang, editors, RFID Security: Techniques, Protocols and System-on-Chip Design, Berlin, Springer, pp. 373–415, 2009.

[16] K. Kim, K. Chung, J. Shin, H. Kang, S. Oh, C. Han, K. Ahn, "A lightweight RFID authentication protocol using step by step symmetric key change", Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing, pp. 853–854, 2009.

[17] J. Shin, Y. Park, S. Kim, Y. Kim, K. Kim, W. Choi, K. Ahn, "A symmetric key based RFID authentication protocol using encrypted tag ID", Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing , pp. 851–852, 2009.

[18] S. Freinkel, H. Herbert, "The AES XCBC-MAC-96 algorithm and its use with IPsec", RFC 3566, IETF, 2003, available at http://www.ietf.org/rfc/rfc3566.txt, last accessed 27 August 2012.

[19] J. Ha, S. Moon, J. Nieto, C. Boyd, "Low-cost and strong-security RFID authentication protocol", Lecture Notes in Computer Science, Vol. 4809, pp. 795–807, 2007.

[20] T. von Deursen, S. Radomirović, "Security of RFID protocols – a case study", Electronic Notes in Theoretical Computer Science, Vol. 244, pp. 41–52, 2009.

[21] W. Stallings, Cryptography and Network Security, 5th ed., Upper Saddle River, NJ, USA, Pearson Education, 2011.

[22] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, L. Vigneron, "The AVISPA tool for the automated validation internet security protocols and applications", Proceedings of the 17th International Conference on Computer Aided Verification, Vol. 3576, pp. 281–285, 2005.

[23] Automated Validation of Internet Security Protocols and Applications, AVISPA, available at http://srg.cs.deu.edu.tr/avispa.zip, last accessed 27 August 2012.

[24] T. von Deursen, S. Radomirović, "Algebraic attacks on RFID protocols, information security theory and practices", Smart Devices, Pervasive Systems, and Ubiquitous Networks, Vol. 5746, pp. 38–51, 2009.