An Enhanced Lightweight Authentication (ELA) Protocol for WISP Based RFID Applications

Gökhan Dalkılıç

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

Abstract: RFID is becoming one of the most incorporated technologies in Internet of Things (IoT). But, the two challenging issues of RFID - the safe mutual authentication of parties and the security of following message exchanges through the air - are still the main hesitation points. In this work, an enhanced lightweight authentication protocol for WISP type RFID tags is presented. The protocol is inspired from the Internet Key Exchange protocol IKEv2, an integral part of the famous Internet Protocol Security IPsec. The proposed protocol is a reduced; lightweight, yet unimpaired version of IKEv2. An extensive security analysis of the proposed protocol against most known attacks is provided.

[Dalkılıç G. **An Enhanced Lightweight Authentication (ELA) Protocol for WISP Based RFID Applications.** *Life Sci J* 2014;11(10):878-885]. (ISSN:1097-8135). http://www.lifesciencesite.com. 138

Keywords: Authentication; cryptographic protocols; information security; message authentication; RFID tags

1. Introduction

Radio frequency identification (RFID) is one of the hot topics of ubiquitous systems that has different and diverse usage areas like asset tracking, inpatient and medicine matching (Yao et al., 2012), transportation, fair collection systems and tracking of commercial goods in supply chains.

Table 1. RFID Classes (Sarma and Engels, 2003)

Class	Description
Class 1	Identity tags
Class 2	Higher functionality tags
Class 3	Semi-passive
Class 4	Active Ad-hoc
Class 5	Reader

For such diversity in application areas, the tags have diverse classes to satisfy the need, which differentiate their properties. Class 1 tags have Object Identifier and a small amount of memory to cut the cost (Table 1). Class 2 tags have read and write commands in the data link layer and encryption capabilities. Class 3 differs from Class 2 by containing its own power source. When out of power, Class 3 gets the energy from the reader and works like Class 2. Last tag type is the Class 4 which has all the properties of previous classes, but also broadband peer to peer communication skills. Class 5 defines the readers.

In brief, long-term usable tags are Class 1 and Class 2 tags. Class 1 tags are much popular, have worldwide usage, and they are defined in EPC Global Gen2 (Engels and Sarma, 2005). In November 2013, EPC Global GS1 announced UHF Class 1 Generation 2 / Version 2.0.0 (Gen2v2, 2013) that includes cryptographic suite indicators. So, Class1 and Class2 distinction is a grey area.

There are some powerful UHF passive tags like Wireless Identification and Sensing Platform-WISP (Chae et al., 2013) or one of the WISP based devices called Moo: A Batteryless Computational RFID and Sensing Platform (Zhang et al., 2011). Both platforms support ECP Global Class 1 Generation 2 protocol and are classified as passively powered tag with sensing and computation (Yeager et al., 2008).

WISP is a UHF tag where exercising symmetric cryptography is possible (Chae et al., 2013), (Szekely et al., 2013). WISP version 4.1 has a MSP430F2132 low power microcontroller which is capable of AES, and elliptic curve cryptography (Pendl et al., 2012). It is obvious that WISP supports true cryptography that can be used in critical applications, such as body area networks (BAN). In the near future, there will be sensors on human body, communicating with each other and other devices using Gen2 compatible protocol standards, creating a BAN (Hanley et al., 2013). So, secure communication is vital for BAN and technologies alike.

2. Related Work

A review of the following recent papers shows that RFID authentication is one of the most popular topics of ubiquitous systems, where RFID is becoming an integral part of IoT. For the lowest cost ultra-lightweight category, some works utilize CRC functions (Gao et al., 2014); one paper offered revocable RFID authentication protocol (Fan et al., 2013). Pang et al. (2013) recommend a new tag indexing method resistant to tag tracking attack. In their another work (Pang et al., 2013), the attack complexity of (Yeh et al., 2010) is increased from 2^{16} to 2^{32}, with less storage and computation power. On the server side, researchers started to propose cloud computing for RFID authentication (Xie et al., 2013). In some papers, securing RFID tag authentication used in inpatient medication has been given plenty of importance (Özcanhan, 2014), (Özcanhan et al., 2014).

There are some RFID authentication protocols offering Hash (Dong et al., 2013), (Ren et al., 2013) and MAC functions (Jung and Jung, 2013), but these functions generally fall in lightweight category. In work (Lee et al., 2013), location tracking and traffic analysis prevention are proposed, which uses RFID authentication protocol with a mobile reader. In another work, lightweight encryption algorithm (XTEA) is used for providing high security, with less computation power compared to three protocols using XTEA (Guangyu and Khan, 2013). With evolving RFID standards and tags, there are works trying to convince the community that elliptic curve cryptography is feasible in RFID (Hein et al., 2009), (Lee et al., 2008) and RFID authentication protocols (Liu et al., 2013), (Liao and Hsiao, 2013).

3. Modification of IKEv2 for RFID

Our motivation is a result of the need for more secure authentication mechanisms in the rapid deployment of UHF RFID technologies. As the data is sent over the air, it is easily eavesdropped. Clandestine users try to get the control of the tag by analyzing the gathered data; in many cases they succeed because of a weak authentication protocol or encryption algorithm.

The main problem of the passive RFID tags is the limited power resource which can only be gained from the reader. This limitation deprives the CPU of the much needed clock cycles and logic gates for an encryption algorithm. But with the relegation of advanced microprocessor production techniques to the tag processor production, the clock cycles and logic gates that can be spared for security in tags is increasing. WISP 5.0 processor is such an example, which contains an AES hardware engine for encrypting the inputs fed to it (MSP430FR5969, 2014). The expectation is WISP tags to be EPC-Gen2 compatible, because these tags are the closest to replacing the traditional barcodes. The developments in the tag integrated circuit technology are paralleled with the new version of Gen-2 standard. Version 2 of Gen-2 supports lightweight encryption. Hence implementing a cost competitive, lightweight version of AES is the key to Gen-2 success against the paper barcodes. Many lightweight versions of AES have been already proposed and many are on the way. Implementing the well proven AES algorithm in tag processors is not the real question, but the price of the resulting processor is. Therefore, using AES in our proposal - which is already supported by WISP 5.0 processors - should not come as a surprise.

Our contribution is to adapt a well-known and proven key generation and exchange mechanism to RFID tags, since the first step in tag authentication is the generation/exchange of secret keys to be used. As it can be observed most of the related works apply a proven method, a property or a function of a previous work to their RFID protocol proposal. None of the applied is a member of a larger security suite. But, there is such a large security suite; a set of standards complementing each other to secure a thing in IoT, which is briefly called IPsec. This is a collection of standards that provide security for computer networks over IP networks. And, IKEv2 is a component of IPsec where mutual authentication is described on the formal website of RFC5996. IKEv2 has found itself an important place in IPv6, as it can be used in important key exchange and authentication problems.

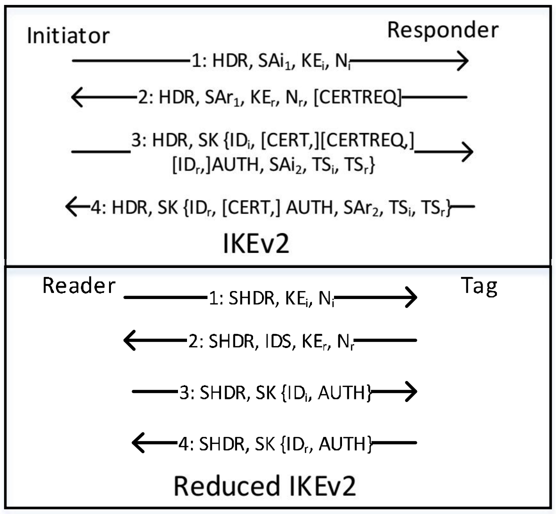

Figure 1. IKEv2 and RIKER comparison

IKEv2, can be a good basis for the key derivation in RFID protocols, because it is a proven key exchange protocol that is used by many applications for tunneling. As defined on the formal website of RFC5996, IKEv2 is burdensome and not suitable for RFID tags' scarce resources. Normally, in IKEv2, there are options to use 3DES, AES_CBC and AES_CTR as encryption algorithms; HMAC_MD5, HMAC_SHA1 and AES128_CBC as pseudorandom functions and HMAC_MD5, HMAC_SHA1, AES_XCBC as integrity algorithms as defined on the formal website of RFC4307. This makes the use of IPsec in different devices flexible; but for a specific device, only one of the algorithms can suffice for encryption, pseudorandom number generation, integrity checking and authentication all in one function. That means less chip area. IKEv2 may be specifically designed for IPsec, but IP related parts can be omitted to form a simple RFID key generation and authentication protocol. In Figure 1, regular IKEv2 versus a reduced IKEv2 is given. For distinction, we will call our simplified or reduced version Reduced IKEv2 for RFID - RIKER. The simplification steps are explained below:

STEP 1: The first step is the simplification of the complex IKEv2 header HDR to a reduced simple header SHDR. HDR consists of Initiator's Security Parameter Index (SPI), Responder's Security Parameter Index (SPI), Next Payload, Major Version, Minor Version, Exchange Type, Flags, Message ID and Length as shown in Figure 2.

Initiator SPI (64 bits)				
Responder SPI (64 bits)				
Next Payload (8 bits)	MjVer (4 bits)	MnVer (4 bits)	Exchange Type (8 bits)	Flags (8 bits)
Message ID				
Length				

Figure 2. HDR

There is no need to use initiator or responder SPI in RFID, because the server is unique and the responder is unknown in this step. Next Payload (RFC Payload Type 34) informs the next message to be sent, which is always the public key KE_i. Therefore Next Payload can be removed for simplification. Major and Minor versions can be removed for a specific standard. Exchange type to be used in this step is always IKE_SA_INIT which has a RFC Exchange Type value of 34 in regular IKEv2 as defined on the formal website of RFC5996, does not need to be declared repeatedly in a simplified version. Flags are used to declare if the message is from the initiator or a responder, in a two party authentication this is not necessary. Last two elements of the IKEv2 header are kept in SHDR. The Message ID (a sequence number) and the Message Length of each message are required for early detection of replay attacks. The length of the message is also important for message integrity. Having simplified the HDR, the next element in the first step is the SAi_1. This is where encryption and authentication algorithms are negotiated, in IKEv2. We propose AES to be used for encryption, authentication and pseudo random number generation. Therefore, by making AES as the only selected function for all purposes, SAi_1 can be also omitted. KE_i and N_i are the Diffie-Hellman public key and nonce values of the initiator, they have to remain.

STEP 2: Step2 is the responder's message to the initiator. HDR is reduced to SHDR due to the simplifications made in step 1. In the regular IKEv2 response, selected algorithms are dictated in SAr_1. As the algorithms are already reduced to a single AES function, this message is omitted. The tag sends its IDS (Pseudo ID), the public Diffie-Hellman responder key (KE_r) and the responder nonce (N_r), which are unreducible. CERTREQ is not used, as the need for a certification authority requires another air communication channel; which is against the nature of RFID.

STEP 3: The third step is reduced to sending SHDR, SK {ID_i, AUTH} from initiator to responder. CERT and CERTREQ have already been eliminated, as certification related parameters. ID_r is omitted, because it is not a mandatory parameter. Traffic selectors (TS_i and TS_r) are the parameters for the source and the destination addresses. But, in UHF RFID communication, tag and the reader identify their path via anti-collision methods, prior to this step. Therefore, there is no need to use the source and the destination parameters. SAi_2 is used for the negotiation of future algorithms, which have all been already agreed to AES, therefore it is redundant. In RIKER, this step includes the transmission of ID and the AUTH of the initiator, encrypted by the key SK, which will be defined in Section 4's Step 3: Server Reply Phase.

STEP 4: Step 4 is the response of the responder with a change. Instead of the initiator's ID, responder's ID is encrypted in this step. At the end of step 4, mutual authentication of the tag and reader is concluded and the newly generated session key will be used throughout the session.

Shortlist of Modifications

The following simplifications are made:

1) AES_XCBC is selected as the only algorithm for the authentication, random number generation, integrity checking and encryption. This increases security, as less secure algorithms cannot be chosen.

2) As the reader, server and the tag are under the control of the same administrator as the initial step, a preshared key can be easily distributed among the parties. According to using the certificates, shared key doesn't need third party, so that IKEv2's Shared Secret Key mode is selected as the only mode to use.

3) HDRs are reduced to SDHRs.

4) Elliptic Curve Diffie Hellman Key Exchange (ECDHKE) is proposed, because it needs less computation power and key length than regular Diffie Hellman; which is more suitably accommodated in UHF tags (Pendl et al., 2012).

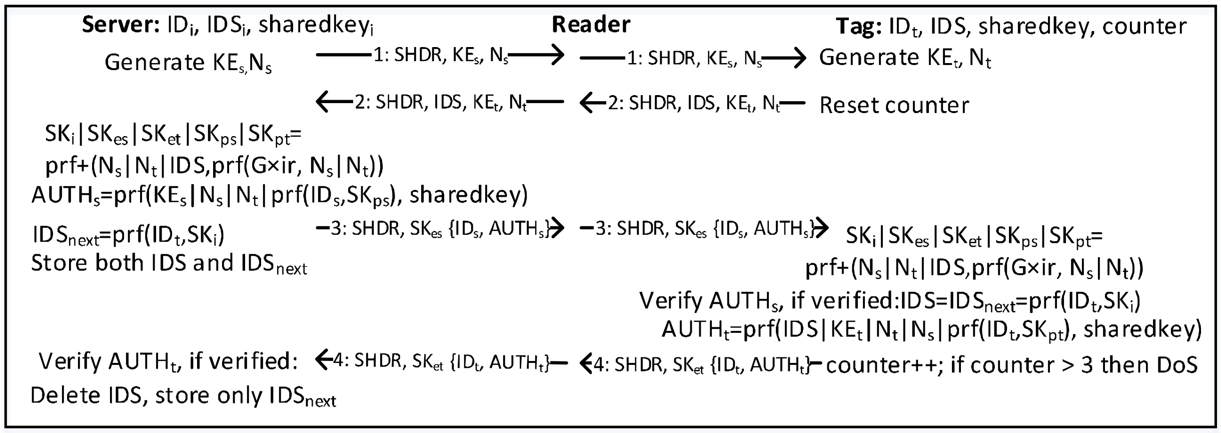
Server: ID_i, IDS_i, sharedkey_i
Reader
Tag: ID_t, IDS, sharedkey, counter
Generate KE_s, N_s
1: SHDR, KE_s, N_s
1: SHDR, KE_s, N_s
Generate KE_t, N_t
2: SHDR, IDS, KE_t, N_t
2: SHDR, IDS, KE_t, N_t
Reset counter
SK_i|SK_es|SK_et|SK_ps|SK_pt= prf+(N_s|N_t|IDS,prf(G×ir, N_s|N_t))
AUTH_s=prf(KE_s|N_s|N_t|prf(ID_s,SK_ps), sharedkey)
IDS_next=prf(ID_t,SK_i)
Store both IDS and IDS_next
3: SHDR, SK_es {ID_s, AUTH_s}
3: SHDR, SK_es {ID_s, AUTH_s}
SK_i|SK_es|SK_et|SK_ps|SK_pt= prf+(N_s|N_t|IDS,prf(G×ir, N_s|N_t))
Verify AUTH_s, if verified:IDS=IDS_next=prf(ID_t,SK_i)
AUTH_t=prf(IDS|KE_t|N_t|N_s|prf(ID_t,SK_pt), sharedkey)
Verify AUTH_t, if verified:
4: SHDR, SK_et {ID_t, AUTH_t}
4: SHDR, SK_et {ID_t, AUTH_t}
counter++; if counter > 3 then DoS
Delete IDS, store only IDS_next

Figure 3. Suggested Authentication Protocol

5) As there is no source or destination address, traffic selectors TS_i (source address) and TS_r (destination address) are removed.

6) SAi_1, SAi_2 and SAr_1, SAr_2 are removed, because there is a preselected algorithm AES_XCBC; there is no need to send parameters to choose the cryptographic algorithm.

4. Enhanced Lightweight Authentication (ELA)

Based on the above proposed RIKER, a secure RFID protocol for critical applications can be devised.

Assumptions - All communication channels; the server – reader and the reader – tag, are considered insecure. But most designs, inferiorly assume that the communication channel between the server and the reader is secure. The server or server farm is assumed to be powerful enough to make its calculations instantly. The role of the reader is assumed to have no knowledge of the messages it handles, as it only sends the messages from the server to the tag and vice versa. Hence, the reader can be assumedly forged, so it cannot be trusted. Tag is a UHF passive RFID tag like WISP having an integrated AES chip (Chae et al., 2013). All the abbreviations and notations used throughout the paper are given in Table 2.

Protocol Description - The protocol takes 4 steps for completing a server - tag mutual authentication as seen in Figure 3. All calculations that normally take place on the reader are now made on the server.

Initialization Phase - For each tag, server keeps a record consisting of a static ID, a dynamic index pseudonym IDS and constant shared key values. IDS value changes in every session. The tag has the ID_t, IDS, shared key and a counter before the start of the exchange. Counter is set to zero.

Server Request Phase – STEP 1: The server generates nonce (N_s) and the Diffie Hellman public key (KE_s) and sends them to the tag via reader.

Table 2. Abbreviations & Notations Used

Abbre-viation	Description
KE_s	Diffie Hellman public key of the server
KE_t	Diffie Hellman public key of the tag
N_s	Nonce of the server
N_t	Nonce of the tag
ID_t	Real ID of the tag
IDS	Pseudo ID of the tag that changes in each session
IDS_{next},	Pseudo ID of the tag to be used as IDS in the next session
ID_S	ID of the server
SK_{es}	Encryption key from server to tag specific for that session
SK_{et}	Encryption key from tag to server specific for that session
SK_{ps}	MAC key from server to tag specific for that session
SK_{pt}	MAC key from tag to server specific for that session
SK_i	IDS_{next} generation key specific for that session
$AUTH_s$	MAC of step 1
$AUTH_t$	MAC of step 2
{}	Encryption Operation
Prf	128 bit AES_XCBC
prf+	Extension function of prf
|	Concatenation

Tag Reply Phase – STEP 2: The tag generates its own nonce (N_t) and Diffie Hellman public key (KE_t) and sends them to the server with its

IDS. The counter that will be used to encounter DoS attacks is cleared.

Server Reply Phase – STEP 3: Using tag's IDS, the server searches the database and finds the tag ID ($ID_t \equiv EPC$) and the shared key. Then, 128 bit AES_XCBC is used to calculate prf(G×ir, $N_s|N_t$); where concatenation of N_s and N_t is the key and G×ir is generated as in ECDHKE (Hein et al., 2009), (Liu et al., 2013). The result of prf(G×ir, $N_s|N_t$) is the key to the prf+ function that generates five keys: Session key of the server for encryption (SK_{es}), session key of the server for MAC generation (SK_{ps}), session key of the tag for encryption (SK_{et}), session key of the tag for MAC generation (SK_{pt}), and session key used for IDS_{next} generation (SK_i). Prf+ is the same function of IKEv2 defined as:

$$prf+(S, K) = T_1 \mid T_2 \mid T_3 \mid T_4 \mid \ldots \quad \text{where:}$$
$$T_1 = prf(S \mid 0x01, K)$$
$$T_2 = prf(T_1 \mid S \mid 0x02, K)$$
$$T_3 = prf(T_2 \mid S \mid 0x03, K)$$
$$T_4 = prf(T_3 \mid S \mid 0x04, K)$$

Hence:

$$SK_i \mid SK_{es} \mid SK_{et} \mid SK_{ps} \mid SK_{pt} = prf+(N_s|N_t|IDS, prf(G\times ir, N_s|N_t)).$$

Therefore, nonces of both sides and IDS of the tag is used as the data, the nonces and the Diffie Hellman values are used as the key to the prf+ function. Next, $AUTH_S$ is generated to create a MAC for the transmitted value, as the server is not authenticated by the tag yet. The message sent from the server to the tag (KE_s, N_s), the nonce of the tag (N_t) and encrypted ID of the server are concatenated and encrypted with the shared key. Also, the IDS_{next} of the next session is calculated by encrypting ID_t with SK_i. The new and the old IDS values are saved. ID of the server (ID_S) and $AUTH_S$ are encrypted with SK_{es} and sent to the tag.

Tag Authentication Reply Phase – STEP 4: The tag first calculates the same session keys. Using one of the obtained keys; SK_{es}, it decrypts {ID_S, $AUTH_S$} to extract $AUTH_S$. Using the shared key, the tag computes its version of $AUTH_S$ and matches it with the received $AUTH_S$. If the match is good, the tag calculates its $AUTH_t$ and the same IDS_{next}. ID_t of the tag and the $AUTH_t$ values are encrypted with SK_{et} and sent to the server.

Before sending message 4, the counter value is increased by 1. If the value of the counter is greater than 3, somebody sent message 3 times, there may be a Denial of Service attack, session starts from the beginning. At the beginning of the session, the counter reset.

When the server receives the tag's authentication reply, the authenticator of the tag $AUTH_t$ has to be verified. If validated, the mutual authentication is complete and the server can safely replace the value of IDS with IDS_{next} and erase the old IDS value. If message 4 does not reach to the server, mutual authentication is not complete and further encrypted data exchange cannot follow. In the next round, the tag starts the negotiation with the new index pseudonym IDS_{next}. The server has this value and continues the authentication with the stored IDS_{next} value. If all goes well, message 4 reaches the server and the tag authenticator $AUTH_t$ is verified, server knows that only the previous message 4 was blocked. But, if tag's message 4 again fails to arrive, this can mean an attack on the authentication protocol. Therefore, if message 4 does not reach the server two consecutive times, the server halts any further protocol with the tag and puts the tag into a blacklist. Thereafter, administrator intervention is needed to remove the tag from the blacklist.

5. Simulation and Testing

The proposed protocol has been first implemented using cryptool2 (cryptool2, 2014) for functional testing. After this step, the correct functioning version has been implemented in C++ using openssl's random number, ECDH and AES functions in Eclipse development environment. The results of many protocol rounds of both implementations; hence the AES engines, have been compared by giving the same inputs. After obtaining results confirming each other, the test results of many runs have been collected in a file. The values were subjected to the full NIST, ENT randomness tests used in a previous study (Özcanhan et al., 2013). All of the nonces and authenticators were generated using AES. As expected, due to the strong properties of AES (Daemen and Vincent, 2002), the generated nonces and authenticators all passed the randomness tests. The values that are passed through the air during the protocol can be sniffed by Wireshark (Orebaugh et al., 2006) but cannot be decrypted, because to the best our knowledge AES encrypted ciphertext cannot be broken once it finishes its full 10 rounds. The source code and the step by step trace of different software runs can be found at http://srg.cs.deu.edu.tr/publications/2014/ela. Simply, ELA's security level is equal to the security level of AES, as it relies completely on AES. Using the fully random and non-decryptable exchanged values of ELA, the following fully convincing security analysis can be made.

6. Security Analysis

Full-disclosure Attack - For a full disclosure attack, all the data and secrets of the tag have to be exposed. As the IDS is sent in cleartext, an eavesdropper can easily get the IDS. But, the intruder can only analyze following exchanges for one round, because IDS changes in the next session. Next secret is the ID of the tag, but it never appears unencrypted, in the exchanges. The last secret is the shared key which the intruder attacks; yet again it is not passed between the peers throughout the communication. For exposing secrets, the keys of the AES encryption have to be exposed. It is to our knowledge that there is no work yet, that claims extracting an AES key from an encrypted AES ciphertext. Therefore, ELA's security level is equivalent to AES's security level and thus it is as resistant as the AES algorithm, to full-disclosure attacks.

Clone Attack - To clone a tag, the first secret to be stolen from the tag is the tag ID. As the tag ID is never transferred in plaintext throughout the authentication exchanges, the key used for encrypting the ID has to be captured, instead. The step to capture the key is a full-disclosure attack. But, once full disclosure attack succeeds, all secrets inside the tag are captured. Hence, cloning becomes possible. As full-disclosure attack is not possible, exposing secrets thus escalating to clone attack is not possible. But, cloning is always a threat through tampering.

Man-in-the-middle Attack - First two steps of the protocol include Diffie Hellman Key Exchange which is not secure against Man-in-the-middle Attack. But, in the third and fourth steps, $AUTH_S$ and $AUTH_t$ are used which include the MAC of the messages of the first and second steps. The man in the middle can try to push his Diffie Hellman public keys to both sides, but cannot generate the AUTH values of the third and fourth steps, because AUTH is generated by using a shared key known only by the server and the tag, so none of AUTHs can be forged.

Perfect Forward Secrecy - For forward secrecy, each time a new key has to be used, and by using session key generation based on Diffie-Hellman Key Exchange, this property is achieved. In perfect forward secrecy, a new exponential with a lifetime more than the session lifetime has to be used. In steps 3 and 4, SK_{es} and SK_{et} are used as the keys for AES encryption from server to the tag and from tag to the server. Both SK_{es} and SK_{et} are created using fresh parameters, every session.

Replay Attack - As the messages between the tag and the reader are sent through the air, these messages can be captured and replayed in a future round, by an attacker. To prevent this type of attack, fresh nonces are used for each session, in step 1 and 2, and also every message starts with SHDR having a sequence number. Additionally, the authenticators $AUTH_s$ and $AUTH_t$ are created in steps 3 and 4, including the fresh nonces. In all authentication steps, messages dependent on newly generated nonces are used by the peers, in every new round.

Impersonation Attack - As the ID of the tag (IDS) that is sent in cleartext, changes after all the successful authentication, and as every tag has a different shared key shared with the server, it is not possible to impersonate a tag. The attacker can collect several IDS values and can try to estimate the next IDS value, but each IDS value is calculated by using a different key specific to that session.

Identity Tracing - Tracing a tag, hence the beholder of the tag is a breach of one's privacy. Therefore, using a different pseudo ID in every new authentication round resists this type of attack. In step 2, tag sends in cleartext a pseudo ID (IDS), a value which is updated as $IDS_{next}=prf(ID_t, SK_i)$, in each session. The static tag ID (EPC) that is unchangeable is never transmitted in cleartext in any message during the protocol exchanges. Therefore, in every new session, the index pseudonym-IDS value changes and makes the tag untraceable. Hence, tag anonymity is provided.

DoS Attack - DoS attack tries to keep one or both of the communicating parties busy in futile computations by bogus challenges to such an extent that the party is unable to answer real protocol messages, before a step times out. To resist such an attack, the tag has a counter value zero in the initialization. After step 3, counter value is increased by one. If counter exceeds a threshold value (e.g. 3), it means the third step of the authentication is repeated before ending the authentication. This identifies the Denial of Service attack to the tag having scarce resources, and the tag doesn't accept third step parameters after the threshold value. Only way is to start the communication from the beginning. As the server has too many resources, any mechanism against DoS attacks aren't added on the server side, but counter values can be added on the server also to encounter DoS attacks to the server.

De-synchronization Attack - De-synchronization attack occurs whenever dynamic terms are updated to different values by the partners. After step 2, server calculates IDS_{next}, and stores both IDS and IDS_{next}. In step 3, an attacker can use a blocker tag (Juels et al., 2003) to block the message of the server to the tag. For the next session, the tag will start with the same IDS, and as the server stores the IDS value, the tag can easily be found from the database. This attack can happen several times, but each time server can find the tag by its IDS value. After step 3, tag calculates IDS_{next}, and updates its IDS with IDS_{next}. Again, the attacker can block the message to the server (step 4). The IDS value in the tag is updated, and the server stores both IDS and IDS_{next}. The server searches through the IDS and IDS_{next} values and finds the tag from the IDS_{next} value. If blocking of step 4 appears two consecutive times, the tag is put into the blacklist and can only be removed by the help of the administrator. If message 4 reaches to the server, it deletes the IDS value as the tag has already updated its IDS value with IDS_{next}.

7. Conclusion

In this study, a new lightweight authentication protocol ELA inspired from IKEv2 is proposed. IKEv2 has different algorithm choices for the peers for encryption, integrity checking, authentication and pseudo random functions. But, the list of algorithms is for different peers capable of running varying algorithms. In RIKER, AES_XCBC algorithm is chosen for all functions. The chosen algorithm is the highest level security function choice of IKEv2, and WISP RFID UHF passive tags have built-in AES chips. As the announced UHF Class 1 Generation 2 / Version 2 standard supports Crypto Suite, our proposal is well placed.

In the first two steps of the protocol, ECDHKE that has the same security level with less key size than DHKE, is used. ECDHKE is also a part of the IKEv2 standard accepted in July 2013, and WISP is capable of ECDHKE (Pendl et al., 2012).

For lower cost tags, other lightweight cryptographic algorithms can be chosen, like: KLEIN (Gong et al., 2012), LED (Guo et al., 2011), TWINE (Suzaki et al., 2011), compact version of AES (Moradi et al., 2011), KATAN (De Canniere et al., 2009), PRESENT (Bogdanov, 2007), SEA (Mace et al., 2007), DESL (Leander et al., 2007), HIGHT (Hong et al., 2006). The comparison of these algorithms can be found in (Guo et al., 2011) and (Çoban et al., 2012).

Based on RIKER, the protocol ELA is secure against full disclosure, clone, man-in-the-middle, replay, impersonation, DoS, de-synchronization attacks, and identity tracing, and satisfies perfect forward secrecy. As a future work, the implementation of the protocol on WISP 5.0 can be done.

Corresponding Author:
Dr. Gökhan Dalkılıç
Department of Computer Engineering
Tinaztepe Campus, Dokuz Eylul University
35160 Buca, Izmir, Turkey
E-mail: dalkilic@cs.deu.edu.tr

References

1. Yao W, Chu CH, Li Z. The adoption and implementation of RFID technologies in healthcare: a literature review. Journal of Medical Systems, 2012;36(6):3507-3525.
2. Sarma S, Engels DW. On the future of RFID tags and protocols. Technical report, Auto-ID Center, Massachusetts Institute of Technology, 2003.
3. Engels DW, Sarma SE. Standardization requirements within the RFID class structure framework. Technical report, Auto-ID Labs, Massachusetts Institute of Technology, 2005.
4. UHF Air Interface Protocol Standard "Gen2v2" (Generation 2 / Version 2), 2013, [online] from: http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2.
5. Chae HJ, Salajegheh M, Yeager DJ, Smith JR, Fu K. Maximalist cryptography and computation on the WISP UHF RFID tag. Wirelessly Powered Sensor Networks and Computational RFID, 2013;175-187.
6. Zhang H, Gummeson J, Randsford B, Fu K. Moo: A batteryless computational RFID and sensing platform. Technical report, Department of Computer Science, University of Massachusetts Amherst, 2011.
7. Yeager DJ, Sample AP, Smith JR. WISP: A passively powered UHF RFID tag with sensing and computation. RFID Handbook: Applications, Technology, Security and Privacy, CRC Press, 2008;261-276.
8. Szekely A, Höfler M, Stögbuchner R, Aigner M. Security enhanced WISPS: implementation challenges. Wirelessly Powered Sensor Networks and Computational RFID, 2013;189-204.
9. Pendl C, Pelnar M, Hutter M. Elliptic Curve Cryptography on the WISP UHF RFID Tag. RFID Security and Privacy, 2012;7055:32-47.
10. Hanley P, Fergus P, Bouhafs F. A wireless body sensor platform to detect progressive deterioration in musculoskeletal systems. Advances in Internet of Things, 2013;3:53-63.
11. Gao L, Ma M, Shu Y, Wei Y. An ultralightweight RFID authentication protocol with CRC and permutation. Journal of Network and Computer Applications, 2014;41:37-46.
12. Fan K, Li J, Li H, Liang X, Shen XS, Yang Y. RSEL: revocable secure efficient lightweight RFID authentication scheme. Concurrency and Computation: Practice and Experience, 2013.
13. Pang L, Li H, He L, Alramadhan A, Wang Y. Secure and efficient lightweight RFID authentication protocol based on fast tag indexing. International Journal of Communication Systems, 2013.
14. Pang L, He L, Pei Q, Wang Y. Secure and efficient mutual authentication protocol for RFID conforming to the EPC C-1 G-2 standard. Proceedings of Wireless Communications and Networking Conference, 2013;1870-1875.
15. Yeh TC, Wang YJ, Kuo TC, Wang SS. Securing RFID systems conforming to EPC Class 1 Generation 2 standard. Expert Systems with Applications, 2010;37(12):7678-7683.

16. Xie W, Xie L, Zhang C, Zhang Q, Tang C. Cloud-based RFID authentication. Proceedings of IEEE International Conference on RFID, 2013;168-175.
17. Dong Q, Zhang J, Wei L. A SHA-3 based RFID mutual authentication protocol and its implementation. Proceedings of IEEE International Conference on Signal Processing, Communication and Computing, 2013;1-5.
18. Özcanhan MH. Improvement of a Weak RFID Authentication Protocol Making Drug Administration Insecure. Life Science Journal, 2014;11(10):269-276.
19. Özcanhan MH, Dalkılıç G., Utku S. Cryptographically Supported NFC Tags in Medication for Better Inpatient Safety. Journal of Medical Systems, 2014;38(8):1-15.
20. Ren X, Xu X, Li Y. An one-way hash function based lightweight mutual authentication RFID protocol. J of Computers, 2013;8(9):2405-2412.
21. Jung SW, S. Jung S. HMAC-based RFID authentication protocol with minimal retrieval at server. Proceedings of the Fifth International Conference on Evolving Internet, 2013;52-55.
22. Lee HC, Eom T, Yi JH. Secure and Lightweight Authentication Protocol for Mobile RFID Privacy. Applied Mathematics and Information Sciences, 2013;7(1):421-426.
23. Guangyu Z, Khan GN. Symmetric key based RFID authentication protocol with a secure key-updating scheme. Proceedings of 26th Annual IEEE Canadian Conference on Electrical and Computer Engineering, 2013;1-5.
24. Hein D, Wolkerstorfer J, Felber N. ECC is ready for RFID–a proof in silicon. Selected Areas in Cryptography, 2009; 5381:401-413.
25. Lee YK, Sakiyama K, Batina L, Verbauwhede I. Elliptic-curve-based security processor for RFID. IEEE Transactions on Computers, 2008;57(11): 1514-1527.
26. Liu Y, Qin X, Wang C, Li B. A lightweight RFID authentication protocol based on elliptic curve cryptography. Journal of Computers, 2013;8(11): 2880-2887.
27. Cryptool2, 2014, [online] from: http://www.cryptool.org/en/cryptool2-en.
28. Özcanhan MH, Dalkılıç G, Gürle MC. An ultra-Light PRNG for RFID tags. Computer and Inf Sciences III. Springer, 2013;231-238.
29. Daemen J, Vincent R. The design of Rijndael: AES-the advanced encryption standard. Springer, 2002.
30. Orebaugh A, Ramirez G, Beale J. Wireshark & Ethereal network protocol analyzer toolkit. Syngress, 2006.
31. Liao YP, Hsiao CM. A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol. Ad Hoc Networks, 2013.
32. MSP430FR5969 16 MHz Ultra-Low-Power Microcontroller, 2014, [online] from: http://www.ti.com/product/msp430fr5969.
33. Juels A, Rivest RL, Szydlo M. The blocker tag: selective blocking of RFID tags for consumer privacy. Proceedings of the 10th ACM Conference on Computer and Communications Security, 2003;103-111.
34. Gong Z, Nikova S, Law YW. KLEIN: a new family of lightweight block ciphers. RFID Security and Privacy, 2012;7055:1-18.
35. Guo J, Peyrin T, Poschmann A, Robshaw M. The LED block cipher, Proceedings of Cryptographic Hardware and Embedded Systems, 2011;326-341.
36. Suzaki T, Minematsu K, Morioka S, Kobayashi E. Twine: A lightweight, versatile block cipher," Proceedings of ECRYPT Workshop on Lightweight Cryptography, 2011; 146-169.
37. Moradi A, Poschmann A, Ling S, Paar C, Wang H. Pushing the limits: a very compact and a threshold implementation of AES. Proceedings of Advances in Cryptology, 2011;69-88.
38. De Canniere C, Dunkelman O, Knežević M. KATAN and KTANTAN—a family of small and efficient hardware-oriented block ciphers," Proceedings of Cryptographic Hardware and Embedded Systems, 2009;5747:272-288.
39. Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJB, Seurin Y, Vikkelsoe C. PRESENT: An ultra-lightweight block cipher. Proceedings of Cryptographic Hardware and Embedded Systems, 2007;450-466.
40. Mace F, Standaert FX, Quisquater JJ. ASIC implementations of the block cipher sea for constrained applications. Proceedings of the Third International Conference on RFID Security, 2007;103-114.
41. Leander G, Paar C, Poschmann A, Schramm K. New lightweight DES variants. Fast Software Encryption, 2007;196-210.
42. Hong D, et al. Hight: A new block cipher suitable for low-resource device. Proceedings of Cryptographic Hardware and Embedded Systems, 2006;4249:46-59.
43. Çoban M, Karakoç F, Boztaş Ö. Biclique cryptanalysis of TWINE. Cryptology and Network Security, 2012;7712:43-55.

9/16/2014