- The communication between the reader and the tag is initiated by the reader, i.e. tags are passive.
- The reader contacts the tag through a wireless channel which is susceptible to different attacks such as location tracking, replay attacks, DoS attacks and impersonation attacks.
- The communication channel between the reader and server is secure.
- The tag's data are stored in non-volatile memory such as EEPROM or Flash memory, where they can be updated.
- All the operations in the tag are atomic i.e. either all of the operations or none are processed. If the attacker kills the electromagnetic field between the reader and tag or simply the tag walks away from the reader's signal, the tag will execute all the computations simultaneously or not at all.
- We assume that the proposed protocol supports a multiple readers scenario, all connected to a central server, so that a tag can be read in many different locations.

Protocol behaviour

The proposed protocol has the following main features:

- The proposed protocol uses random numbers in an attempt to prevent location tracking and replay attacks.
- The server stores both the old and the new values of the data in order to prevent desynchronization attacks.
- After a successful authentication between the server and tag, both parties update their values to be used in the next transaction.

Notation

The notation used in the proposed protocol is presented below:

1) The notation related to the server database is,
   - ID_{old}: The tag's old ID
   - ID_{new}: The tag's new ID
   - K_{old}: The old secret key
   - K_{new}: The new secret key
2) The notation related to the tag is,
   - T_i: The i^{th} tag of the RFID system, where $1 \leq i \leq N$
   - ID: The tag's ID, shared with the server's ID_{old} or ID_{new}
   - K: The tag's secret key shared with the server's K_{old} or K_{new}
3) Other notation used in the proposed protocol is,
   - x: The value kept as either new or old to show whether the tag uses the old or new values of ID and K
   - R1: A pseudo random number generated by the reader
   - R2: A pseudo random number generated by the tag and serving as a temporary secret for the tag
   - H: A hash function, $h:\{0,1\}^* \rightarrow \{0,1\}^L$, where L is equal to the length of the data
   - $A \oplus B$: Message A is XORed with message B
   - A || B: Message A is concatenated with message B
   - $A \leftarrow B$: The value of A is updated to that of B
   - j: The transaction number
   - N: The number of tags managed by the server

Protocol description

The scheme consists of two processes namely initialization, and authentication.

1) Initialisation Process: This stage only occurs during manufacturing when the manufacturer assigns the initial values in the server, and in the tag. The initialisation process is summarised below:
   - The server assigns random values for each tag it manages to (ID_{new}, K_{new}) in the server and (ID, K) in the tag.
   - Initially, (ID_{old}, K_{old}) in the server is set to null.
2) Authentication Process: The authentication process is shown in Table 1.
   - Reader: The reader generates a random number R1 of L bits and sends it to the tag.
   - Tag:
     - The tag generates a random number R2 of L bits as a temporary secret for the session, and computes:
       HID=H(ID || R1),
       M1=H(K || R1 || R2),
       $M2=ID \oplus R2$
     - The tag sends HID, M1 and M2 to the reader.
   - Reader: The reader sends R1, HID, M1, and M2 to the server.
   - Server:
     - For all the stored IDs, the server computes H(ID || R1) until it finds a match with the received value of HID:

- If there is a match in ID_{new}, then the server marks x=new. The server retrieves data (ID_{new}, K_{new}), extracts R2 i.e. $R2=ID_{new} \oplus M2$, and re-computes M1 i.e. $M'1=H(K_{new} \| R1 \| R2)$ to authenticate the tag.
- If there is a match in ID_{old}, then the server marks x=old, retrieves the data (ID_{old}, K_{old}), extracts R2 i.e. $R2=ID_{old} \oplus M2$ and re-calculates $M'1=H(K_{old} \| R1 \| R2)$ to authenticate the tag.
- The server computes $M3=H(ID_x \| K_x \| R1 \| R2)$, and transmits it to the reader.
- The server updates the data as follows:
  If x=new, where ID is found in ID_{new}

  $ID_{new}^{j+1} \leftarrow H(ID_{new}^{j})$
  $ID_{old}^{j+1} \leftarrow ID_{new}^{j}$
  $K_{new}^{j+1} \leftarrow H(K_{new}^{j} \oplus ID_{new}^{j+1})$
  $K_{old}^{j+1} \leftarrow K_{new}^{j}$

  Else if x=old, where ID is found in ID_{old}:
  No updates
  If there is no match in ID_{new} and ID_{old} or $M'1 \neq M1$ or $M'2 \neq M2$, then the server sends an end session message to the reader to terminate the session.

- Reader: Once the reader receives M3, it sends M3 to the tag.
- Tag: The tag determines whether the received value of M3 is equal to $H(ID \| K \| R1 \| R2)$. If there is a match, the tag authenticates the server and updates its values to:

  $ID^{j+1} \leftarrow H(ID^{j})$
  $K^{j+1} \leftarrow H(K^{j} \oplus ID^{j+1})$

  If the check fails or M3 is not received, the tag keeps the current values unchanged.

5. Protocol analysis

In this section, we analyse the proposed protocol in terms of informal and formal analysis using a privacy model, CasperFDR and AVISPA. Finally, we present the expected performance measurement.

5.1 Informal security analysis of the protocol

In Table 2 We compared our protocol with the related research work against the main requirements shown in Section 2. Our proposed protocol provides the following goals:

Our proposed protocol provides the following goals:

1) Tag anonymity: The tag stores two values, namely (ID, K) that supposed to be secret and not revealed to any entity except the legitimate server. Only the legitimate server that has information related to the tag can extract these values.
2) Tag location privacy (untraceability): In the proposed protocol, the tag's responses are changed with new updated values and fresh random numbers, thus the attacker will obtain new responses every time he eavesdrops on a session. Moreover, if the previous authentication session failed and the tag's data remain unchanged, HID, M1 and M2 responses will change due to the existence of new fresh random numbers.
3) Resistance to replay attack: The proposed protocol utilises a challenge-response scheme, where each party maintains a set of random numbers it has seen from previous protocol run to avoid repeated random numbers. Thus, when the tag or server detects repeated random numbers, it will terminate the session.
4) Resistance to desynchronisation attack: In the proposed protocol, the desynchronisation attack is avoided via storing the previous values of the data (ID_{old}, K_{old}), and thus reach synchronization. Moreover, the server does not update its data when there is a match in (ID_{old}, K_{old}), it keeps the stored data the same. Thus when the attacker blocks M3 more than once respectively, the tag's data (ID, K) will still match the server's data (ID_{old}, K_{old}).
5) Resistance to tag impersonation attack: To impersonate the tag, the attacker must be able to compute a valid response (HID, M1 and M2) to a server query. However, it is hard to compute such responses without knowledge of ID, and K.
6) Resistance to server impersonation attack: To impersonate the server, the attacker must be able to compute a valid response (M3). However, it is hard to compute such responses without knowledge of ID_x, K_x, and R2.

5.2 Privacy analysis

The researchers have proposed number of privacy models to evaluate the privacy of the RFID protocols such as [21, 35]. The model in [21] is summarised as follows: An adversary (A) controls the communication channel between a tag (T) and a reader (R) by interacting either passively or actively with them. The adversary can run the following queries:

- *Execute (R, T, i) query*: The adversary can passively eavesdrop on a session (i) and obtain access to the exchanged messages between R and T.
- *Send (U, V, m, i) query*: The adversary can perform active attacks by impersonating an entity such as V ∈ T and sends a message (m) to entity U ∈ R during session (i). Also, he can alter or block some of the exchanged messages.
- *Corrupt (T, K′) query*: The attacker can physically access the tag's memory T and read the tag's secret value (K′).
- *Test (i, T_0, T_1) query*: This query is used to define the untraceability test. When this query is invoked for session (i), a random bit b2 ∈{0, 1} is generated and then, A is given T_b ∈{T_0, T_1}. Informally, A wins if he can guess the bit b.

Untraceable privacy (UPriv) is defined as a game (g) played by the adversary (A) and a collection of the reader and the tag instances. The game consists of three phases:

1) Learning phase: The adversary (A) can send the Execute, Send, and Corrupt queries to any random T_0 and T_1 tags.
2) Challenge phase: The adversary (A) is given a tag T_b ∈ {T_0, T_1}, and sends any Execute, and Send queries to T_b.
3) Guess phase: A terminates the game and outputs a bit b_0, which is its guess of the value of b.

The success of A in winning (g) and breaking the untraceability privacy (UPriv) is achieved in terms of A's advantage in distinguishing whether A received T_0 or T_1, i.e. it correctly guessing b. This is denoted by Adv_A^{UPriv}k, where k is the security parameter. Now, we will evaluate the privacy of our proposed protocol using this model. We found that the adversary cannot invade the privacy of the tag and trace its location as shown below:

1) Learning phase: The adversary eavesdrops a valid session between R and T_0. He sends the *Execute* command and then maintains the following values, which are sent :

   R1, HID=H(ID_0 || R1)
   M1=H(K_0 || R1 || R2)
   M2= ID_0 ⊕ R2

2) Challenge phase: A is given a tag T_b ∈ {T_0, T_1} randomly. He starts a new session with T_b by impersonating the reader and sends R1 to T_b within the *Send* query and terminates the session. T_b responds can be:

   HID=H(ID_b ||R1)
   M1=H(K_b || R1 || R2′)
   M2= ID_b ⊕ R2′

However, A will not be able to guess the correct tag (bit b) as the received messages M1 and M2 contain a random number (R2) generated by the tag, which changes in every session, and not known to the adversary. Moreover, regarding HID message, if the tag encounters a repeated random number such as R1, it will terminate the session.

5.3 Formal Analysis of the Protocol Using CasperFDR

CasperFDR is a compiler that takes a high level description of the protocol and analyses the protocol description against the stated specification to show whether the protocol meets the main requirements.

Previously, researchers [36, 37] have attempted to model their protocol using a Communication Sequential Process (CSP) and a Failure-Divergence Refinement (FDR). CSP is a language for specifying the protocol behaviour. The generated CSP file is analysed by FDR. FDR is a model checker that analyses a protocol and verifies the given specifications. Gavin Lowe has developed CasperFDR tool [19], which takes a high level description of the protocol together with its security requirements and produces a CSP code checked and verified by FDR.

CasperFDR is used to verify the authentication and secrecy requirements of the protocol. Authentication has two forms of specification namely *Agreement* and *NonInjectiveAgreement*. *Agreement* means if Bob meets the Agreement specification, he confirms that Alice has run the same protocol, and agreed on the exchanged values. For example *Agreement* (T, S, [R1, R2, ID, K]) means that the tag is authenticated to the server and both parties agreed on the data values (R1,R2, ID, K), and it is one to one relationship i.e. each run of the tag corresponds to a unique run of the server. Another form of authentication specification is *NonInjectiveAgreement* which differs from *Agreement* whereas each run can be repeated and overlap.

Table 1. Goals and requirements comparison

	WP	WP2	OP	MP	DP	DucP	CP	SG	PP	SP2	YP	YoonP	HP	Sec.4
Tag anonymity	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Location privacy	×	√	√	√	×	√	×	√	√	√	√	×	√	√
Replay attacks	×	×	×	√	√	×	√	√	√	√	√	×	√	√
Denial of service attacks	√	√	√	√	×	×	×	×	×	×	√	√	×	√
Tag impersonation attacks	×	×	√	√	√	×	×	√	√	√	×	×	×	√
Server impersonation attacks	×	×	×	×	×	×	×	√	×	√	×	×	×	√

Table1. The proposed protocol authentication process

Server $ID_{new}, ID_{old}, K_{new}, K_{old}$	Reader	Tag ID,K
	R1 →	Generate R2 HID=H(ID ‖ R1) M1=H(K ‖ R1 ‖ R2) M2=ID ⊕ R2
	← HID,M1,M2	
For all the IDs, compute $H(ID_x‖R1)$ Retrieve the matched ID_x , K_x	← R1,HID,M1,M2	
Mark x=new or old Compute $R2=ID_x \oplus M2$, Re-compute $M'1=H(K_x‖R1‖R2)$ Calculate $M3=H(ID_x‖K_x‖R1‖R2)$		
	M3 →	
If x=new, update: $ID_{new}^{j+1} \leftarrow H(ID_{new}^{j})$ $ID_{old}^{j+1} \leftarrow ID_{new}^{j}$ $K_{new}^{j+1} \leftarrow H(K_{new}^{j} \oplus ID_{new}^{j+1})$ $K_{old}^{j+1} \leftarrow K_{new}^{j}$ Else if x=old, No updates	M3 →	Re-compute M′3=H(ID ‖ K ‖ R1‖ R2) If M3 = M′3 Update: $ID^{j+1} \leftarrow H(ID^{j})$ $K^{j+1} \leftarrow H(K^{j} \oplus ID^{j+1})$ If the check fails or M3 is not received, the tag keeps the current values unchanged.

Similarly secrecy has two forms of specification namely *Secret* and *StrongSecret*. Secret checks whether the intruder could know the secret value at the end of the protocol. While, *StrongSecret* checks whether the intruder can obtain the secret value before the protocol finishes its run.

CasperFDR checks the authentication and secrecy specifications by examining the associated events. In other words, *Running* and *Commit* events are attached to the authentication specifications. The server and tag are both depicted in CasperFDR as CSP processors. When the tag sends (HID, M1, M2, R1) to the server, the server performs the *Running* event, which means that the server starts running the protocol apparently with the tag. Then, the tag performs the *Commit* event when it receives the server's reply M3, which means that the tag has finished a run of the protocol with the server [38].

CasperFDR checks the secrecy specifications via an event called *Claim_Secret*, which is performed by both parties. When the tag receives the server's message (M3), it performs *Claim_Secret* to ensure that (ID, K, R2) are secret [38]. This process is shown in Figure 1. For simplicity, we treat the server as a reader and a database.

5.3.1 The proposed protocol requirement illustration in CasperFDR

Figure 1 shows how the main requirements are achieved in CasperFDR (the numbers in the figure presents the same goals shown below). The following goals are achieved as follow:

1) Mutual authentication: Before the tag sends M1 and M2, it performs *Running.T.S.[ID,K,R1,R2]* event, which means the tag starts a run of the protocol, apparently with the server agreeing on data. Later, the server will perform *Commit.S.T.[ID,K,R1,R2]* event at the end of its part of the protocol, which means the server has finished the protocol with the tag agreeing on the received data. Similarly, before the server sends M3, it performs *Running.S.T.[ID,K,R1,R2*], and when the tag receives M3 it performs *Commit.T.S.[ID,K,R1,R2]*.
2) Resistance to impersonation attack: The server performs a *Running* event such as *Running.S.T.ID.R2*, which means that the server starts a run of the protocol, apparently with the tag, agreeing on ID and R2 in M2. Then, the server performs *Running.S.T.K.R1.R2*, which means that the server agrees on K, R1 and R2 in M1. Later, the tag will perform the *Commit.T.S.ID.R2* and *Commit.T.S.K.R1.R2* events at the end of its part of the protocol, which means that the tag has finished the protocol with the server agreeing on the values of ID, K, R1 and R2.
3) Tag anonymity is depicted as *Claim_Secret.T.S.ID* and *Claim_Secret.T.S.K* and *Claim_Secret.T.S.R2* events, which means that the three values of ID, K and R2 should be kept secret between the tag and the server.
4) Resistance to replay attack is illustrated as a scenario where the tag is engaging in the protocol twice. The tag firstly runs the protocol with the server, and the intruder obtains R1. Then, the intruder runs the protocol with the same tag and resends R1 to the tag. Therefore, in our protocol the tag will not perform the *Commit* event as it received the same random number R1. Similarly, if the server engages in the protocol run by receiving duplicate messages from the intruder or the tag, it will not perform the *Running* event.

We prepared the CasperFDR script to show some indicative results if there is an attack on the protocol or not. The script is shown in Appendix A. The section #Specification, specifies the security and authentication requirements of the protocol, the lines *Secret (T, K, [S]), Secret (T, ID, [S]),* and *Secret (T, R2, [S]),* indicate that the values of K, ID, and R2 should only be known by the tag (T) and legitimate server (S). The lines starting with Agreement are for providing authentication for instance, *Agreement (T, S, [R1, R2, ID, K])* means that the tag is authenticated to the server using the data values (R1, R2, ID, K).

In addition, in the #Intruder information section, the intruder is defined to be Mallory, who can take a full control of the session; he can impersonate any entity in the protocol, generate a random number, read the messages transmitted in the network, intercept, analyse, and/or modify messages.

CasperFDR did not find any feasible attacks on the proposed protocol.

5.4 Formal analysis of the protocol using AVISPA

In addition to using CasperFDR for formally analyzing the proposed protocol, we presented our protocol using a High Level Protocol Specification Language (HLPSL), a specification language for formalizing protocols [20]. This language is then translated with the Automated Validation of Internet Security Protocols and Applications (AVISPA) model checker tool by using a translator called HLPSL2IF and four different integrated verification

backends called: On the Fly Model Checker(OFMC) [39], Constraint-Logic based Attack Searcher (CL-AtSe) [40], SAT based Model-Checker (SATMC) [41] and Tree Automata based Protocol Analyser (TA4SP) [42]. These backends implement a variety of automatic analysis techniques to check the main goals of the protocol such as secrecy and authentication.

In these backends, the intruder is modelled via using the channel(dy) which stands for the Dolev-Yao intruder model [43]. Under this model, the intruder has full control over the network, such that all messages sent by agents can be eavesdropped by the intruder. Moreover, the intruder may intercept, analyse, modify messages, and/or send any message he/she composes to other agents pretending to come from a legitimate agent.

In AVISPA, there are three roles. Firstly, *basic role*, which defines the agent who runs the protocol and the initial information and parameters the agent holds. Secondly, in the *composition role*, we describe the sessions of the protocol by specifying how the agents interact with each other. A top-level role *environment role* contains global constants and a composition of one or more sessions, where the intruder may play some roles as a legitimate user. Moreover, environment role shows what knowledge the intruder initially has.

The main requirements and goals are declared in a section called *goal*. There are two main goals in AVISPA namely secrecy and authentication. Secrecy is modelled via the goal predicate *secret* Authentication is modelled by means of the goal predicates *witness* and *request.*

The script is shown in Appendix B. To elaborate the script, there are two agents namely a server and a tag. We assume that the server acts as a reader and a database. The server and tag shares a *symmetric key* (K) and they both utilise a hash function (H) for the calculation of the messages. The tag ID, ID_{old} and random numbers R1 and R2 are defined as *text*. ID defines ID_{new}. R1 and R2 are freshly generated using the *new* function.

In the script, authentication is achieved via *witness* and *request* goals i.e. *witness(S, T, trid, ID)* which declares that agent S asserts to be the peer of agent T, agreeing on the value ID, trid is the name of the ID authentication shown in the goal section. *request(T,S,trid,Auth')* can be read as "agent T accepts the value Auth1 and now relies on the guarantee that agent S exists and agrees with it on this value".

Regarding secrecy, *secret(ID,id,T,S)* means that the value of ID should be a secret between agents T and S, and id is the name of the secret term, which is defined in the *goal* section.

In the *environment role*, the *intruder* is identified and we assume that the intruder knows the other agents (tag and server), keys he shares with the agents, and hash function.

Our protocol script has been analysed by the OFMC and CL-AtSe backends as they support an exclusive-OR properties. The results show that the protocol is safe. The other backends SATMC and TA4SP do not support an exclusive-OR property that is why the result shows inconclusive.

5.5 Performance analysis

In this section, we conduct a comparative analysis of the performance cost regarding storage cost, and communication cost.

a) Storage cost: Due to the limitation of tag memory, the tag should store minimum amount of data. In the proposed protocol, the tag stores two values in a rewritable flash memory namely (ID, K), as they change in different authentication sessions, each of which has a length of 224 bits. Since the tag's memory can store 1 Kilobyte of data, in our protocol the tag securely stores 224*2=448 bits in the memory. Additional tag memory is necessary in our protocol to store a list of random numbers received from previous queries, for example by adding extended on-chip non-volatile memory on the RFID tags.

b) Communication cost: In the proposed protocol, the tag sends three messages (HID, M1 and M2) in order to be successfully authenticated. A total of 672 bits are sent over the channel as the length of one message is 224 bits. Hence, it provides a relatively low communication cost.

6. Conclusion

In this paper we presented a lightweight RFID mutual authentication protocol that based on the strength found in previous protocols and prevent their deficiencies. The protocol has been informally and formally analysed using formal methods. Firstly, based on the informal analysis, the results demonstrate that the protocol performs better than the selected protocols and offers immunity against a broad range of attacks. Secondly, the privacy of the tag's data is evaluated via the privacy model, which showed that the tag's data cannot be traced or compromised. Thirdly, the secrecy and authentication requirements have been analysed via CasperFDR and AVISPA formal tools. CasperFDR and AVISPA did not show any feasible attacks. Finally, we conducted a performance comparative analysis in terms of storage, communication costs and server scalability, and we concluded that the proposed protocol is compatible with the RFID systems requirements.

7. Acknowledgment

This research was supported by the Ministry of Higher Education and King Khaled University in Saudi Arabia.

8. References

[1] K. Finkenzeller and R. Waddington, *RFID handbook: radio-frequency identification fundamentals and applications*. Wiley New York, 1999.

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

[3] K. Finkenzeller, "*RFID handbook: Fundamentals and applications in contactless smart cards and identification*," 2nd ed. New York, USA, John Wiley, 2003.

[4] J. Li, Y. Wang, B. Jiao, and Y. Xu, "An authentication protocol for secure and efficient RFID communication," in *Logistics Systems and Intelligent Management, 2010 International Conference on*, vol. 3. IEEE, 2010, pp. 1648–1651.

[5] P. Pessl and M. Hutter, "Pushing the limits of SHA-3 hardware implementations to fit on RFID," in *Cryptographic Hardware and Embedded Systems-CHES 2013*. Springer, 2013, pp. 126–141.

[6] B. Song and C. Mitchell, "RFID authentication protocol for low-cost tags," in *Proceedings of the first ACM conference on Wireless network security*. ACM, 2008, pp. 140–147.

[7] H. Chien and C. Chen, "Mutual authentication protocol for RFID conforming to EPC Class 1 Generation 2 Standards," *Computer Standards Interfaces*, vol. 29, no. 2, pp. 254–259, 2007.

[8] D. Duc, H. Lee, and K. Kim, "Enhancing security of EPCglobal Gen-2 RFID against traceability and cloning," in *Symposium on Cryptography and Information Security*. The Institute of Electronics, Information and Communication Engineers, 2006.

[9] T. Dimitriou, "A lightweight RFID protocol to protect against traceability and cloning attacks," in *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*. IEEE, 2005, pp. 59–66.

[10] M. Shemaili, C. Yeun, and M. Zemerly, "RFID lightweight mutual authentication using shrinking generator," in *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*, 2009, pp. 1–6.

[11] G. Poulopoulos, K. Markantonakis, and K. Mayes, "A Secure and Efficient Mutual Authentication Protocol for Low-Cost RFID Systems," in *Availability, Reliability and Security, 2009. ARES'09. International Conference on*. IEEE, 2009, pp. 706–711.

[12] S. Weis, S. Sarma, R. Rivest, and D. Engels, "Security and privacy aspects of low-cost radio frequency identification systems," *Security in pervasive computing*, pp. 50–59, 2004.

[13] M. Ohkubo, K. Suzuki, S. Kinoshita *et al.*, "Cryptographic approach to "privacy-friendly" tags," in *RFID Privacy Workshop*, vol. 82. MIT, Cambridge, MA, 2003.

[14] S. Weis, "Security and privacy in radio-frequency identification devices," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[15] T. Yeh, Y. Wang, T. Kuo, and S. Wang, "Securing RFID systems conforming to EPC Class 1 Generation 2 Standard," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7678–7683, 2010.

[16] E. Yoon, "Improvement of the securing RFID systems conforming to EPC Class 1 Generation 2 Standard," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1589–1594, 2012.

[17] Y. HanataniI, M. Ohkubo, S. Matsuo, K. Sakiyama, and K. Ohta, "A study on computational formal verification for practical cryptographic protocol: The case of synchronous RFID authentication," *Financial Cryptography and Data Security*, pp. 70–87, 2012.

[18] D. N. Duc and K. Kim, "Defending RFID authentication protocols against DoS attacks," *Computer Communications*, vol. 34, no. 3, pp. 384–390, 2011.

[19] G. Lowe, "Casper: A compiler for the analysis of security protocols," in *Computer Security Foundations Workshop, 1997. Proceedings., 10th*. IEEE, 1997, pp. 18–30.

[20] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani *et al.*, "The AVISPA tool for the automated validation of internet security protocols and applications," in *Computer Aided Verification*. Springer, 2005, pp. 281–285.

[21] K. Ouafi and R. Phan, "Privacy of recent RFID authentication protocols," in *Information Security Practice and Experience*, ser. Lecture Notes in Computer Science, L. Chen, Y. Mu, and W. Susilo, Eds. Springer Berlin Heidelberg, 2008, vol. 4991, pp. 263–277. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-79104-1_19

[22] A. Mitrokotsa, M. Rieback, and A. Tanenbaum, "Classifying RFID attacks and defenses," *Information Systems Frontiers*, vol. 12, no. 5, pp. 491–505, 2010.

[23] H. Kim, J. Oh, J. Choi, and J. Kim, "The Vulnerabilities Analysis and Design of the Security Protocol for RFID System," in *Computer and Information Technology, 2006. CIT'06. The Sixth IEEE International Conference on*. IEEE, 2006, pp. 152–152.

[24] B. Song, "RFID authentication protocols using symmetric cryptography," Ph.D. dissertation, Royal Holloway, University of London, 2009.

[25] G. Avoine, E. Dysli, and P. Oechslin, "Reducing time complexity in RFID systems," in *Selected Areas in Cryptography*. Springer, 2006, pp. 291–306.

[26] D. Molnar and D. Wagner, "Privacy and Security in Library RFID: issues, Practices, and Architectures," in *Proceedings of the 11th ACM conference on Computer and communicationssecurity*. ACM, 2004, pp. 210–219.

[27] P. Peris-Lopez, J. Hernandez-Castro, J. Estévez-Tapiador, and A. Ribagorda, "LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags," in *Proc. of 2nd Workshop on RFID S*

[28] D. Coppersmith, H. Krawczyk, and Y. Mansour, "The shrinking generator," in *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-*

26, 1993, Proceedings, ser. Lecture Notes in Computer Science, vol. 773. Springer, 1993, pp. 22–39.

[29] S. Cai, Y. Li, T. Li, and R. Deng, "Attacks and improvements to an RFID mutual authentication protocol and its extensions," in *Proceedings of the second ACM conference on Wireless network security*. ACM, 2009, pp. 51–58.

[30] P. Rizomiliotis, E. Rekleitis, and S. Gritzalis, "Security analysis of the Song-Mitchell authentication protocol for low-cost RFID tags," *Communications Letters, IEEE*, vol. 13, no. 4, pp. 274–276, 2009.

[31] S. Abughazalah, K. Markantonakis, and K. Mayes, "A vulnerability in the Song authentication protocol for low-cost RFID tags," in *Security and Privacy Protection in Information Processing Systems*, ser. IFIP Advances in Information and Communication Technology, L. Janczewski, H. Wolfe, and S. Shenoi, Eds. Springer Berlin Heidelberg, 2013, vol. 405, pp. 102–110.

[32] S. Abughazalah, K. Markantonakis, and K. Mayes, "Desynchronisation attacks on an RFID mutual authentication protocol," to be published.

[33] M. Habibi, M. Aref, and D. Ma, "Addressing flaws in RFID authentication protocols," in *Progress in Cryptology–INDOCRYPT 2011*. Springer, 2011, pp. 216–235.

[34] J. Hernandez-Castro, P. Peris-Lopez, M. Safkhani, N. Bagheri, and M. Naderi, "Another fallen hash-based RFID authentication protocol," *Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems*, pp. 29–37, 2012.

[35] G. Avoine, "Adversarial model for radio frequency identification." *IACR Cryptology ePrint Archive*, vol. 2005, p. 49, 2005.

[36] G. Lowe, "Breaking and fixing the Needham-Schroeder public-key protocol using FDR," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 1996, pp. 147–166.

[37] G. Lowe and B. Roscoe, "Using CSP to detect errors in the TMN protocol," *Software Engineering, IEEE Transactions on*, vol. 23, no. 10, pp. 659–669, 1997.

[38] A. Alshehri and S. Schneider, "Formally defining nfc m-coupon requirements, with a case study," *The 5th International Workshop on RFID Security and Cryptography 2013 (RISCï¿½13), Internet Technology and Secured Transactions*, pp. 58–64, December 2013.

[39] D. Basin, S. Modersheim, and L. Vigano, "An on-the-fly model-checker for security protocol analysis," in *Computer Security " ESORICS 2003*, ser. Lecture Notes in Computer Science, E. Snekkenes and D. Gollmann, Eds. Springer Berlin Heidelberg, 2003, vol. 2808, pp. 253–270. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-39650-5_15

[40] M. Turuani, "The CL-ATSE protocol analyser," in *Term Rewriting and Applications*, ser. Lecture Notes in Computer Science, F. Pfenning, Ed. Springer Berlin Heidelberg, 2006, vol. 4098, pp. 277–286. [Online]. Available: http://dx.doi.org/10.1007/11805618_21

[41] K. McMillan, "Interpolation and SAT-based model checking," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, J. Hunt, WarrenA. and F. Somenzi, Eds. Springer Berlin Heidelberg, 2003, vol. 2725, pp. 1–13. [Online]. Available: http://dx.doi.org/-10.1007/978-3-540-45069-6_1

[42] Y. Boichut, P. Héam, O. Kouchnarenko, and F. Oehl, "Improvements on the Genet and Klay technique to automatically verify security protocols," in *Proc. AVIS*, vol. 4, 2004.

[43] D. Dolev and A. Yao, "On the security of public key protocols," *Information Theory, IEEE Transactions on*, vol. 29, no. 2, pp. 198–208, 1983.

Appendix A
CasperFDR Script

```
#Free variables
T : Agent
S : Server
R1 : initialSeq
R2 : Sequence
IDold, ID : SessionID
Kold , K : SessionKey
h: HashFunction
InverseKeys= (Kold,Kold), (K,K) ,
(IDold,IDold)  , (ID , ID)
#Protocol description
0.   -> S : T
1. S -> T : R1
2a. T -> S : R2 (+) ID
2b. T -> S : h(K, R1, R2)
 [IDold = = ID and Kold = = K
or ID = = ID and K == K ]
4. S -> T : h(ID, K, R1, R2)
#Processes
RESPONDER(T, S, R2, K, ID)
SERVER (S, T, R1, Kold, IDold, K, ID)
#Actual variables
Tag, Mallory : Agent
ServerDB : Server
Rr1 : initialSeq
Rr2 : Sequence
R3 : Sequence
IDentityO,IDentityT:SessionID
KeyOld, KeyTag : SessionKey
InverseKeys=(KeyOld,KeyOld),
(KeyTag, KeyTag), (IDentityO,IDentityO),
,(IDentityT,IDentityT)
#Specification
Aliveness(S, T)
Secret(T, K, [S])
Secret(T, ID, [S])
Secret(T, R2, [S])
Agreement(T, S, [R1, R2, ID,K])
Agreement(S, T, [R1, R2, ID,K])
#System
RESPONDER(Tag,ServerDB,Rr2,KeyTag,IDentityT)
SERVER(ServerDB, Tag, Rr1, KeyOld, KeyTag
IDentityO, IDentityT)
#Intruder Information
Intruder = Mallory
IntruderKnowledge ={Tag,ServerDB,Mallory,R3}
```

Appendix B
AVISPA Script

```
role server(S, T: agent,

        K : symmetric_key,
        H:hash_func,
        SND, RCV: channel(dy))
```

```
played_by S def=

      local State: nat,
      R1,R2, ID, IDold:text,
      Auth1:hash(symmetric_key.text.text),
      Kold : symmetric_key


init State := 0

transition

0.  State = 0  /\ RCV(start)
    =|>   State' := 1
                /\ R1' := new()
                /\ SND(R1')

1. State = 3   /\  K' = K  /\ ID' = ID
                /\ RCV(H(K.R1.R2').xor(R2',ID)) =|>
  State' := 4   /\ Auth1':= H(K.R1.R2')
                /\ request(S,T,id3,Auth1')
                /\ SND(H(ID.K.R1.R2'))
                /\ witness(S, T, trid, ID)
                /\ witness( S, T, trk, K)
                /\ ID':= new()
                /\ IDold' :=new()
                /\ K' := new()
                /\ Kold' := new()

1.  State=3    /\  Kold'= K /\  IDold'= ID
                /\ RCV(h(K.R1.R2').xor(R2',ID)) =|>
  State' := 4   /\ Auth1' := H(Kold.R1.R2')
                /\ request(S,T,id3,Auth1')
                /\ SND(H(IDold.Kold.R1.R2'))
                /\ witness(S, T, trid, IDold)
                /\ witness( S, T, trk, K)
end role

%%%%%%%%%%%%%%%%%%%%%%%
role tag( T,S: agent,
        K: symmetric_key,
        H: hash_func,
        SND,RCV: channel(dy))
played_by T def=

      local State   : nat,
      R1,R2 , ID: text,
      Auth : hash(text. text.text)

 init  State := 0

 transition

  0. State = 0  /\ RCV(R1')
   =|> State' := 1
                /\ R2' := new()
                /\ SND(h(K.R1'.R2').xor(R2',ID))
                /\ witness( T, S, trid,ID)
                /\ witness( T, S, trk, K)


  1. State = 1  /\ RCV(h(ID.K.R1'.R2'))

  =|> State' := 2
                /\ Auth' := h(ID.K.R1'.R2')
                /\ request(T,S,trid,Auth')
                /\ secret(ID,id, {T, S})
                /\ secret (R2', id2, {T, S})
                /\ secret(K,id3, {T, S})
                /\ ID':=new()
                /\ K' := new()

end role

%%%%%%%%%%%%%%%%%%%%%%%
role session( T,S : agent,
              K : symmetric_key,
             Hash : hash_func)
def=

local SND, RCV: channel (dy)

composition

  tag(T,S,K,Hash, SND, RCV)

/\ server (S,T,K,Hash, SND, RCV)
end role
%%%%%%%%%%%%%%%%%%%%%%%

role environment( ) def=
const
id, id2, id3, id4, id5,trid, trk: protocol_id,
h : hash_func,
k, kti,ksi: symmetric_key,
tag, server: agent

intruder_Knowledge = {tag,server,h,i,kti,ksi}
composition

     session(tag,server,k,h)
  /\ session(tag,i,kti,h)
  /\ session(i,server,ksi,h)
end role

%%%%%%%%%%%%%%%%%%%%%%%
goal

secrecy_of id, id2, id3

authentication_on trid

authentication_on id3
authentication_on trk

end goal
%%%%%%%%%%%%%%%%%%%%%%%
environment()
```

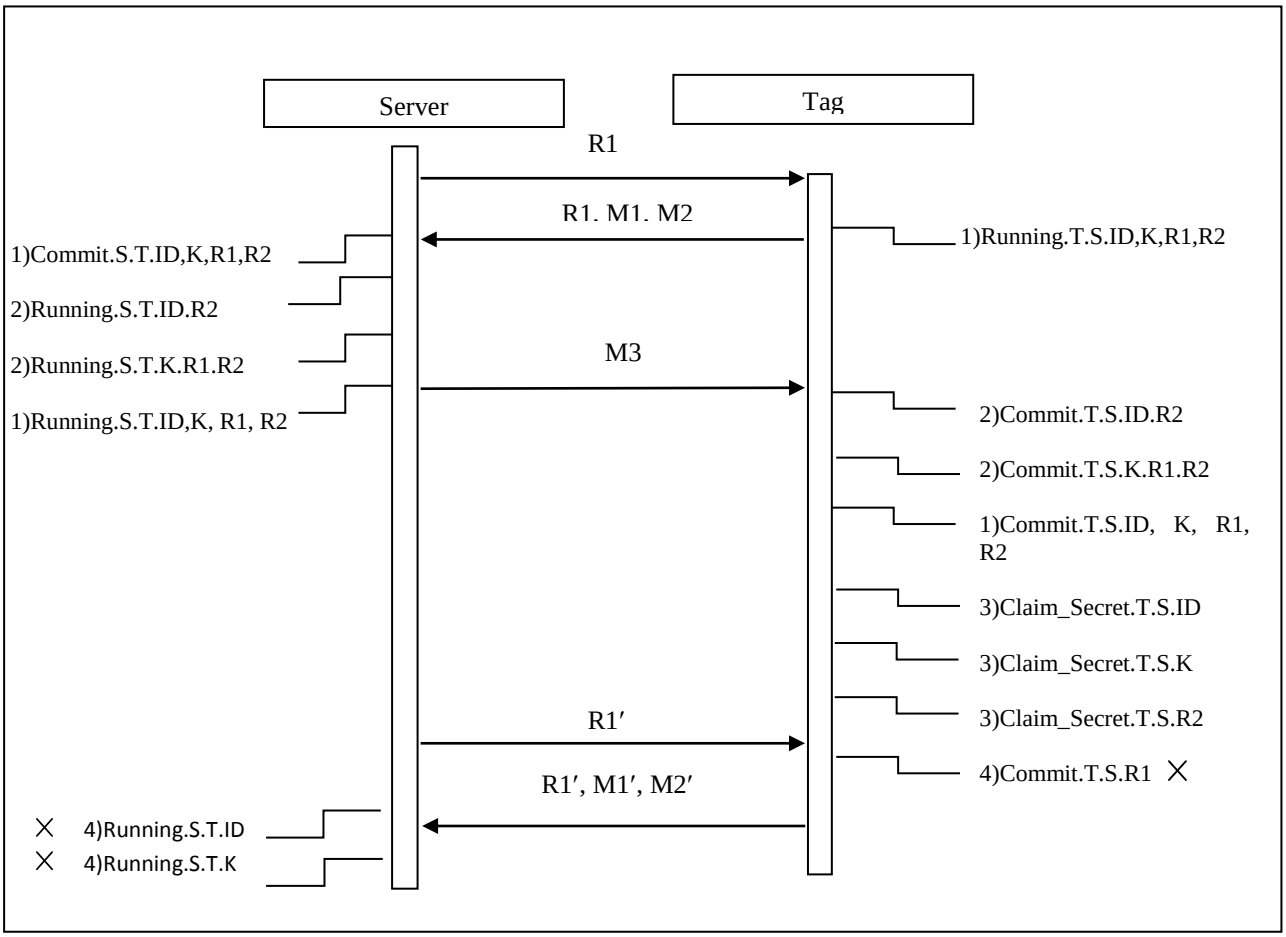

Figure1. The proposed protocol requirement illustration in CasperFDR