A New Lightweight Mutual Authentication Protocol to Secure Real Time Tracking of Radioactive Sources

Mouza Ahmed Bani Shemaili[1], Chan Yeob Yeun[2](✉), Mohamed Jamal Zemerly[2], Khalid Mubarak[1], Hyun Ku Yeun[1], Yoon Seok Chang[3], Basim Zafar[4], Mohammed Simsim[6], Yasir Salih[4], and Gaemyoung Lee[5]

[1] Computer Information and Science Division, HCT, Abu Dhabi, UAE
{malshemaili,kalhammadi1,hyun.yeun}@hct.ac.ae
[2] Department of Electrical and Computer Engineering, Khalifa University, Abu Dhabi, UAE
{cyeun,jamal.zemerly}@kustar.ac.ae
[3] School of Air Transportation and Logistics, Korea Aerospace University, Goyang, Korea
yoonchang@kau.ac.kr
[4] Department of Electrical Engineering, Umm Al-Qura University, KSU, Mecca, Saudi Arabia
{bjzafar,ysali}@uqu.edu.sa
[5] College of Engineering, Jeju National University, Jeju, Korea
myounglk@jejunu.ac.kr
[6] Ministry of Hajj, KSU, Mecca, Saudi Arabia
msimsim@hajj.gov.sa

Abstract. Radioactive applications are employed in many aspects of our life, such as industry, medicine and agriculture. One of the most important issues that need to be addressed is the security of the movement of radioactive sources. There are many threats that may occur during the transportation of the radioactive sources from one place to another. This paper investigates the security issues in the transportation of the radioactive sources. Thus, it is an attempt to build a secure, real time freight tracking system in which the radioactive source can be under inspection and control at all times during transportation from the shipment provider to the end user. Thus, we proposed a novel lightweight mutual authentication protocol to be used for securing the transportation of radioactive materials. Also, the security requirements for the proposed protocol were verified using the Scyther tool.

Keywords: Radioactive sources · Cyber security, mutual authentication Scyther tool · Real-time tracking

1 Introduction

Radioactive applications have grown rapidly in our lifetime as they can be used for a variety of purposes. For example, radioactive sources are used in the field of medicine for cancer therapy and blood irradiation. Additionally, these sources can be used in engineering to check flow gauges or to test soil moisture and material thickness/integrity

J. Hu et al. (Eds.): MONAMI 2017, LNICST 235, pp. 231–246, 2018.
https://doi.org/10.1007/978-3-319-90775-8_19

for construction or by specialists to irradiate food to prevent it from spoiling. However, radioactive sources can be dangerous in that they can negatively affect the user's health; thus, in order to overcome this problem, there is a need to instruct the user in how he/she can use these applications in a safe way [1, 2].

To handle these radioactive sources in a safe way, they must be controlled under a professional authority or organization. There are several organizations all over the world that regulate the usage of radioactive applications. The most famous of these organizations is the International Atomic Energy Agency (IAEA). The IAEA is the world's centre of cooperation in the nuclear field. It was set up as the world‘s "Atoms for Peace" organization in 1957 within the United Nations family. This agency works with its member states and multiple partners worldwide to promote safe, secure and peaceful nuclear technologies [3]. In the United Arab Emirates (UAE), there is a dedicated agency that addresses radioactive sources, known as the Federal Authority for Nuclear Regulation (FANR), which was established in 2009. FANR is the authority that is responsible for regulating all of the nuclear activities and licenses that involve the use of radioactive sources in UAE. FANR cooperates with the IAEA to satisfy international practice of the peaceful use of nuclear energy [4].

Using a radioactive source can lead to a disaster if the official oversight is insufficient. In this case, a poorly regulated source is known as a vulnerable source. A vulnerable source which safety cannot be ensured and which is not under regulatory control due to its being lost or stolen can become an orphan source. Both of these sources can carry risk levels that can lead to the damage of human health. The potential dangers can include injury or death. A serious example is the Teletherapy Heads accident, which happened on Samut Prakarn, Thailand in 2000, resulting in the deaths of three workers as a consequence of their exposure to the radiation [5].

In addition, orphan sources can be used in terrorist activities, such as using it to produce a radiological dispersal device (RDD). RDD can be used maliciously to produce a 'dirty bomb' or it can be spread deliberately and therefore expose innocent people to radioactive radiation. Terrorists routinely use this as a weapon to destroy the peace of the community by placing radioactive sources in public areas [6, 7].

To overcome the vulnerabilities that may occur during the transportation of the radioactive source, IAEA has come up with certain regulations and measures that should be considered during the design stage, such as measuring the quantity and the physical and chemical form of the radioactive material. Additionally, the mode of transport and the type of packaging used to transport the source must be taken into account.

We can add to these regulations the importance of identifying the threats and malicious acts that may occur during the transportation of the radioactive sources and provide rapid preventative counter measures to any unauthorized attempts to access the source. This is used to prevent, disrupt and defeat a terrorist operation before it occurs. Thus, one of the most important issues that require addressing is the security of the movement of the radioactive sources, as there is a significant threat that may occur during the transportation of these sources from one place to another.

This paper focuses on the security issues in the transportation of the radioactive sources. This security can be achieved by building a secure, lightweight (in terms of CPU workload) real time freight tracking system in which the radioactive source can be under inspection and control at all times during its transportation from the shipment provider to the end user. The real time tracking system can be achieved by combining technologies from the Real Time Location Systems (RTLSs), which are the Global Positioning System (GPS) [8], Global System for Mobile Communications (GSM) [9], the 3rd Generation Project (3GPP) and active Radio Frequency IDentification (RFID) in the form of E-Seal [10], while the Wireless Sensor Network (WSN) [11] adds an environment sensing ability to the moving radioactive sources. However, combining all of these technologies can also give rise to certain security issues such as replay, modification, eavesdropping and Man-in-the-Middle attacks that the system needs to address.

The contribution of this paper is to address the security threats and secure communication of several entities in the real time tracking of radioactive transportation system and having them all synchronized and mutually authenticating each other. This is combined with formal methods verification proof and performance comparison with other protocols.

The main motivation of this paper is to design a secure real time freight tracking system. Thus, within the tracking system there is a need to secure the messages of transporting a shipment in a vehicle from the shipment provider to the end user. Consequently, in order to send messages in secure manner there is a need to design a secure communication among the two communication parties.

Thus, this paper designed a new mutual authentication protocol and is organized as follows. Section 2 explains the radioactive sources transportation scenario. Then, the related work is clarified in Sect. 3. Section 4 explains our proposed mutual authentication protocol, which is used to secure the system communication messages during the transportation of the radioactive materials from source to destination. Section 5 discusses the security analysis of our proposed mutual authentication protocol. Finally, Sect. 6 concludes the paper.

2 Scenario of Radioactive Sources

E-Seal is an active RFID tag that is used to lock the containers of physical goods. If anyone tries to tamper or remove the E-Seal tag, it will send an alarm to alert the shipment provider that there could be a physical attack on this content. The E-Seal can then provide evidence of the authenticity and integrity of the goods as well as physical protection. For more information about E-Seal, the reader can refer to our previous contributions about securing an E-Seal real time tracking system for the Internet of things [12, 13]. In our system, the E-Seal uses the cryptography symmetric approach in [14], which depends on a shared secret key between the shipment provider and another trusted party at the end user side. Thus, nobody can unlock the truck until it reaches its destination (including the driver of the truck himself). The process of transporting a radioactive source shipment in a truck from a shipment provider to an end user consists of three main stages as follows:

1. At the Shipment Provider (Main Center)
2. From the Shipment Provider to the Destination
3. At the End User

Using the cryptography symmetric approach increases the protection of the truck [12, 13]. If any attacker tries to intercept the truck on its way to the end destination, he/she will not be able to unlock the E-Seal because the driver does not have the key that can unlock the truck. Thus, E-Seal can provide security to the truck until it reaches the final destination.

The next section explains the security threats that can occur during the transportation of the radioactive materials from the shipment provider to the destination.

3 Related Works

The application that we need to secure is a real time freight tracking system. Usually for tracking system we only need to authenticate the truck responses. However, the tracking of our system can have messages from the server to change the truck path. Thus, there is a need to design a mutual authentication protocol that authenticates both of the server and the truck. The SSL/TLS authentication protocol is deemed heavy due to required PKI infrastructure and instead the lightweight proposed protocol described in [14] is used. To the best of our knowledge no such end to end real-time tracking of radioactive sources systems exist in the literature.

As explained before the mutual authentication protocol that are discussed in this paper are related to the suggested solutions of the EPC networks since the designs of the real time freight tracking system is based on the EPC networks. Thus, this paper investigated the related work on the EPC protocols.

One of the most famous proposed protocols is that of Chien [15]. He proposed an ultra-lightweight mutual authentication protocol for RFID tags. The protocol consists of two main parts, which are the initialization phase and authentication phase. At the initialization phase, the server randomly chooses an initialization key Ki and initialization access key Pi. Additionally, each tag stores three values, which are the Electronic Product Code (EPCx) of the tag, the initial authentication key Ki0 and the initial access key Pi0. After each successful authentication, the Ki0 and Pi0 keys are updated and stored as Ki and Pi. The server saves 6 values, which are: EPCx, the old authentication key KOLD, the new authentication key KNEW, the old access key POLD, the new access key PNEW and DATA that is related to information on each tag. However, this protocol is under replay attack which is presented in Peris-Lopez et al. [16].

Lo and Yeh [17] improved the Chien's protocol. Their tag stored five values, which are the authentication key Ki, the database access key Pi, the Electronic Product Code (EPCx), the transaction number (TID) and the last successful transaction number (LST). Both the Ki and Pi are generated from the PRNG function at the initialization time of the tag. The server side saved 12 values, which are the new authentication key KNEW, the old authentication key KOLD, the database access key Pi, the new Electronic Product Number Code (EPCx-NEW), the old Electronic Product Number Code (EPCx-OLD), the new fast search key (PRNG (EPCx-NEW)), the old fast search key (PRNG (EPCx-OLD)), the transaction number (TID) and the last successful transaction number (LST). Additionally, the server implements two functions, which are the binary string length function LEN () and the binary string truncation function TRUNC () that are used by the server side to validate and authenticate the tag. Nevertheless, the protocol still suffers from tracing attacks [18, 19].

Yeh et al. [20] tried to improve the Chien's family protocols. With this protocol, the tag saved 4 values, which are the authentication key Ki, the access key Pi, the database index Ci set to Ci = 0 at the start of the communication and the Electronic Product Code (EPCx). The reader saves the Reader Identification number (RID) only and implements the hash function H(.), where the server saves 9 values, which are the new authentication key KNEW, the old authentication key KOLD, the new access key PNEW, the old access key POLD, the new database index CNEW, the old database index COLD, RID, EPC and DATA of each tag.

The protocol in [21] has two vulnerabilities, which are data integrity and forward secrecy problems; this is why Yoon [22] proposed an improvement to this protocol. The protocol adds XORing a session random value at the first message on the tag in order to provide forward secrecy. Additionally, at the server side, a new message called MAC = H(DATA ⊕ NR) is added to verify the message sent from the server side and to ensure the integrity of the data. However, this protocol still adds too much load on the server side and is vulnerable to tracking attacks.

Mohammad Ali et al. [23] noted that the Yoon's protocol is under data forgery, server and tag impersonation attacks. For this reason, this paper proposed an improvement to the protocol by having the tag generate two random numbers NT and NT' and change the way of computing the M1 and E messages. Unfortunately, this protocol still adds computational load on the server side. Additionally, the protocol is under desynchronization attack because the Ci index is updated whether the session is successful or not, where: Ci is corresponding to the database index stored in the tag to find the corresponding record of the tag in the database; thus, the attacker could then launch a false message just to change the Cx value on the server side so that it will not match the value sent from the tag side. Thus, we proposed a new lightweight mutual authentication protocol that provides fewer loads on the server and the tag than other protocols in Sect. 6.

4 The Proposed Lightweight Mutual Authentication Protocol

A new lightweight protocol is one that is dependent on light computation, such as Pseudo-Random Number Generator (PRNG), and simple functions, such as (CRC) checksum, but not hash functions. Our proposed protocol depends on the PRNG only, which is why it is considered as a lightweight authentication protocol. Additionally, our protocol reduces the load on the server side. The SSL/TLS authentication protocol is deemed heavy due to required PKI infrastructure and instead the lightweight proposed protocol described in [14] is used.

Notations: The following notations are used in the proposed protocol:

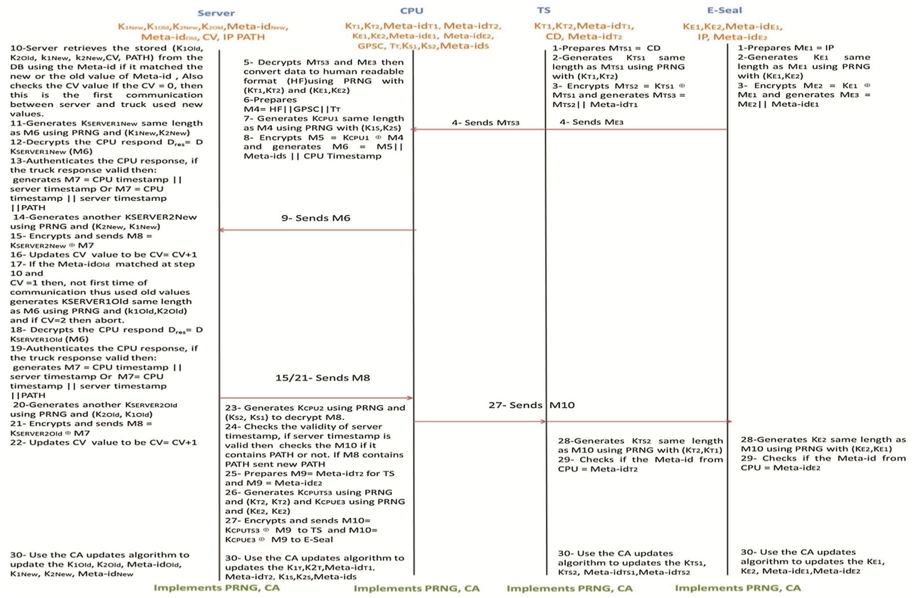
- ⊕ denotes XOR.
- || denotes concatenation.

- **TS**: Active RFID Tag and sensor installed on the container. Each container has its own TS. Here, for the sake of simplicity we assume that we have only one entity as TS.
- **E-Seal**: Active RFID and sensor used to lock the truck shipment.
- **CPU**: Central Processing Unit in the truck.
- **DB**: Database
- **PRNG**: The SGCA cipher stream algorithm proposed in [22] has two functions which are SG and CA. SG is used to produce PRNG stream to encrypt the exchanged messages on the communication implemented in TS, E-Seal, CPU and Server. Whereas, CA is used to update the keys used in the communication.
- **K_{T1}**: Stored shared secret between the TS and the CPU. It is used to authenticate the TS to the CPU and stored in both of the TS and CPU.
- **K_{T2}**: Stored shared secret between the TS and the CPU. It is used to authenticate the CPU to the TS and stored in both of the TS and CPU.
- **K_{E1}**: Stored shared secret between the E-Seal and the CPU. It is used to authenticate the E-Seal to the CPU and stored in both of the E-Seal and CPU.
- **K_{E2}**: Stored shared secret between the E-Seal and the CPU. It is used to authenticate the CPU to the E-Seal and stored in both of the E-Seal and CPU.
- **CD**: Data related to the container status provided from TS.
- **Information Packet (IP)**: E-Seal has an information packet with a length of 49 bytes, which contains an EPC for each tag, E-Seal Serial Number (SN), timestamp (TS) and Cyclic Redundancy Check (CRC) calculated from IP to detect accidental changes within the tag data. Thus, **IP = CRC (EPC, SN, TS)**.
- **Meta-id$_{T1}$**: A pseudonym randomly chosen for each TS. This value is stored in both the TS and the CPU used to identify the communication from the TS to the CPU.
- **Meta-id$_{T2}$**: A pseudonym randomly chosen for the CPU stored at the both of the TS and CPU. This value is used to identify the communication from the CPU to the TS.
- **Meta-id$_{E1}$**: A pseudonym randomly chosen for each E-Seal. This value is stored in both the E-Seal and the CPU used to identify the communication from the E-Seal to the CPU.
- **Meta-id$_{E2}$**: A pseudonym randomly chosen for the CPU stored in both of the E-Seal and CPU. This value is used to identify the communication from the CPU to the E-Seal.
- **GPSC**: GPS coordinates saved in the CPU. This information is retrieved from the GPS receiver attached to the CPU.
- **T_T**: Time of the truck retrieved from the GPS receiver attached to the CPU.
- **K_{S1}**: Stored shared secret between the CPU and the Server used to authenticate the CPU to the Server and stored in both the CPU and the Server (K_1 in the Server).
- **K_{S2}**: Stored shared secret between the CPU and the Server used to authenticate the Server to the CPU and stored in both the CPU and the Server (K_2 in the Server).
- **Meta-id$_S$**: A pseudonym randomly stored in the CPU. This value is stored in both the CPU and the Server used to identify the records of the CPU in the DB.
- **K_1**: Stored shared secret between the CPU and the Server used to authenticate the CPU to the database. The database stores the K_1 [Old, New], which refers to old and new values.

- **K_2**: Stored shared secret between the CPU and the Server used to authenticate the Server to the CPU. The database stores the K_2 [Old, New], which refers to old and new values.
- **Meta-id**: A pseudonym randomly stored at the Server. This value is stored in both the CPU and the Server used to identify the records of the CPU at the DB. The DB stores the Meta-id [Old, New], which refers to old and new values.
- **CV**: Stands for counter value saved on the server side, which has a value of 0 by default and can be increased by one with each successful communication between the CPU and the Server.
- **PATH**: This set the new path of the truck in case that the path needs to be change. At the start there is plan path between A and B that are saved in the truck so in case of emergency the truck can change its track based on PATH data that is received from the server.

The SGCA mutual authentication protocol is shown in Fig. 1. The protocol is used to secure the communication within the proposed real time tracking system. The truck has two types of messages, which are;

1. Messages sent from the TS
2. Messages sent from the E-Seal.


Server
CPU
TS
E-Seal
K1New,K1Old,K2New,K2Old,Meta-idNew, Meta-idOld, CV, IP PATH
KT1,KT2,Meta-idT1, Meta-idT2, KE1,KE2,Meta-idE1, Meta-idE2, GPSC, TT,KS1,KS2,Meta-ids
KT1,KT2,Meta-idT1, CD, Meta-idT2
KE1,KE2,Meta-idE1, IP, Meta-idE2
4- Sends MTS3
4- Sends ME3
9- Sends M6
15/21- Sends M8
27- Sends M10
Implements PRNG, CA
Implements PRNG, CA
Implements PRNG, CA
Implements PRNG, CA

1-Prepares MTS1 = CD
2-Generates KTS1 same length as MTS1 using PRNG with (KT1,KT2)
3- Encrypts MTS2 = KTS1 ⊕ MTS1 and generates MTS3 = MTS2|| Meta-idT1

1-Prepares ME1 = IP
2-Generates KE1 same length as ME1 using PRNG with (KE1,KE2)
3- Encrypts ME2 = KE1 ⊕ ME1 and generates ME3 = ME2|| Meta-idE1

5- Decrypts MTS3 and ME3 then convert data to human readable format (HF)using PRNG with (KT1,KT2) and (KE1,KE2)
6-Prepares
M4= HF||GPSC||TT
7- Generates KCPU1 same length as M4 using PRNG with (K1S,K2S)
8- Encrypts M5 = KCPU1 ⊕ M4 and generates M6 = M5|| Meta-ids || CPU Timestamp

10-Server retrieves the stored (K1Old, K2Old, k1New, k2New,CV, PATH) from the DB using the Meta-id if it matched the new or the old value of Meta-id , Also checks the CV value If the CV = 0, then this is the first communication between server and truck used new values.
11-Generates KSERVER1New same length as M6 using PRNG and (K1New,K2New)
12-Decrypts the CPU respond D_{res}= D KSERVER1New (M6)
13-Authenticates the CPU response, if the truck response valid then:
generates M7 = CPU timestamp || server timestamp Or M7 = CPU timestamp || server timestamp ||PATH
14-Generates another KSERVER2New using PRNG and (K2New, K1New)
15- Encrypts and sends M8 = KSERVER2New ⊕ M7
16- Updates CV value to be CV= CV+1
17- If the Meta-idOld matched at step 10 and
CV =1 then, not first time of communication thus used old values generates KSERVER1Old same length as M6 using PRNG and (k1Old,K2Old) and if CV=2 then abort.
18- Decrypts the CPU respond D_{res}= D KSERVER1Old (M6)
19-Authenticates the CPU response, if the truck response valid then:
generates M7 = CPU timestamp || server timestamp Or M7= CPU timestamp || server timestamp ||PATH
20-Generates another KSERVER2Old using PRNG and (K2Old, K1Old)
21- Encrypts and sends M8 = KSERVER2Old ⊕ M7
22- Updates CV value to be CV= CV+1

23- Generates KCPU2 using PRNG and (KS2, KS1) to decrypt M8.
24- Checks the validity of server timestamp, if server timestamp is valid then checks the M10 if it contains PATH or not. If M8 contains PATH sent new PATH
25- Prepares M9= Meta-idT2 for TS and M9 = Meta-idE2
26- Generates KCPUTS3 using PRNG and (KT2, KT2) and KCPUE3 using PRNG and (KE2, KE2)
27- Encrypts and sends M10= KCPUTS3 ⊕ M9 to TS and M10= KCPUE3 ⊕ M9 to E-Seal

28-Generates KTS2 same length as M10 using PRNG with (KT2,KT1)
29- Checks if the Meta-id from CPU = Meta-idT2

28-Generates KE2 same length as M10 using PRNG with (KE2,KE1)
29- Checks if the Meta-id from CPU = Meta-idE2

30- Use the CA updates algorithm to update the K1Old, K2Old, Meta-idOld, K1New, K2New, Meta-idNew

30- Use the CA updates algorithm to updates the K1T,K2T,Meta-idT1, Meta-idT2, K1S,K2S,Meta-ids

30- Use the CA updates algorithm to updates the KTS1, KTS2, Meta-idTS1,Meta-idTS2

30- Use the CA updates algorithm to updates the KE1, KE2, Meta-idE1,Meta-idE2

Fig. 1. The proposed mutual authentication protocol

As explained before, the truck has a reader that reads messages from the TS and E-Seal. The reader sends all these to the central processing unit (CPU) located in the truck. The proposed protocol assumes that the channel between the TS/E-Seal and the reader is insecure and the channel between the reader and the CPU is also not secure. The TS sends encrypted messages regarding the status of the containers, while the E-Seal sends the Information Packet (IP).

CPU is a connector that connects the data sent from the container/E-Seal to the Central tracking Server and vice versa. Thus, in the proposed communication the CPU stored four shared keys between container TS/E-Seal and the CPU (K_{T1}, K_{T2}) for TS and (K_{E1}, K_{E2}) for the E-Seal. Also, another two shared keys between the CPU and the Server (K_{S1}, K_{S2}).

The CPU collects all of the messages received from the reader then decrypts using the shared secret key's (K_{T1}, K_{T2}) and converts them to a human readable language. Also, the CPU makes sure of the numbers of containers available in the truck and the status of each one. Also, the GPS receiver is attached to the CPU to acquire the GPS coordinates and time data. Thus, the CPU first prepares a message about the radioactive sources shipment status, quantities, GPS coordinates and time. Then, the CPU encrypts this message with the shared secret keys between the CPU and the Server (K_{S1}, K_{S2}) and sends it to the server using SMS through the GSM/GPRS/3G network. On the server side, there is a layer called the security center that is responsible for decrypting the SMS and then sending the data to the upper layer, as explained in Sect. 3. Thus, the communications in the SGCA protocol are between four main entities which are: TS, E-Seal, CPU and Server.

The protocol steps are as follows:

1. The TS and E-Seal first prepare their messages separately and send them at the same time. TS prepares $M_{TS1} = CD$ where E-Seal prepares $M_{E1} = IP$.
2. The TS then generates a key using the proposed stream cipher (PRNG) called K_{TS1} that is used to encrypt the communication between the TS and the CPU. The PRNG uses (K_{T1}, K_{T2}) to generate K_{TS1}. The length of K_{TS1} is the same length of M_{TS1}. Also, The E-Seal generates a key using the proposed stream cipher (PRNG) called K_{E1}, that is used to encrypt the communication between the E-Seal and the CPU. The length of K_{E1} is the same length of M_{E1}.
3. The TS encrypts the message $M_{TS2} = K_{TS1} \oplus M_{TS1}$, generates $M_{TS3} = M_{TS2} || \text{Meta-id}_{T1}$. Also, The E-Seal encrypts the message $M_{E2} = K_{E1} \oplus M_{E1}$, generates $M_{E3} = M_{E2} || \text{Meta-id}_{E1}$
4. The TS sends M_{TS3} and the E-Seal sends M_{E3} to the CPU through reader that will relay these messages to the CPU. For simplicity we did not add the reader as its job is only to relay the message from the TS/E-Seal to the CPU and vice versa.
5. The CPU gets M_{TS3} and M_{E3} from the reader and then retrieves the stored (K_{T1}, K_{T2}, K_{E1}, K_{E2}) from the DB using the Meta-id_{T1} and Meta-id_{E1} depending on the matched value. Then, CPU decrypts these messages using either PRNG with (K_{T1}, K_{T2}) for the TS or PRNG with (K_{E1}, K_{E2}) for the E-Seal. Also, the CPU converts these messages to a human readable format (HF). HF contains the information related to the status of the containers, E-Seal IP information and truck overall status, if the truck status is valid or not valid.
6. The CPU prepares $M4 = HF || GPSC || T_T$.
7. The CPU then generates a key using the proposed stream cipher (PRNG) called K_{CPU1} that is used to encrypt the communication between the CPU and the Server.

The PRNG uses (K_{S1}, K_{S2}) to generate K_{CPU1}. The length of K_{CPU1} is the same length of M4.

8. The CPU encrypts the message M5 = $K_{CPU1} \oplus$ M4, and generates M6 = M5||Meta-id$_S$.
9. The CPU sends M6 to the Server.
10. The server gets M6 from the CPU and then retrieves the stored (K_{1Old}, K_{2Old}, K_{1New}, K_{2New}, CV, PATH) from the DB using the Meta-id depending on the matched value if it is New or Old. Also, the server checks the CV value. If the CV = 0, then this is the first communication between the server and the truck and the New values of the Keys need to be used.
11. The server generates key K_{S1New} using the PRNG algorithm and the shared (K_{1New}, K_{2New}) between the CPU and the server with the same length of the received M6.
12. The server decrypts the CPU response D_{res} = D K_{S1New} (M6).
13. The server authenticates the CPU response by checking the serial number SN of the packet on the DB and the validity of the timestamp; this information is then stored in the IP in the database. If the timestamp is valid and SN is correct, then the server generates M7 = CPU timestamp || server timestamp. If there is a need to set a new path to the track then M7 = CPU timestamp || server timestamp || PATH.
14. Next, the server generates another K_{S2New} using PRNG, but this time it switches to (K_{2New}, K_{1New} as input to the PRNG algorithm), which will produce another stream cipher to encrypt M7.
15. The server encrypts the message M8 = $K_{S2New} \oplus$ M7 and sends it to the CPU.
16. The server updates the CV value to be CV = CV + 1.
17. If at step 10 the matched value of the Meta- id is Old and CV = 1, then it means that this is not the first communication between this CPU and the server; thus, the server uses the old (K_{1Old}, K_{2Old}) that are saved in the DB to generate K_{S1Old}. If the counter value = 2, then the communication is aborted as it may be a type of attack on the communication. Thus, we cannot use the old IVs more than twice.
18. The server decrypts the CPU response D_{res} = D K_{S1Old} (M6).
19. The server checks again if the SN is correct or not and the validity of the timestamp. If it is correct, then the server generates M7 = CPU timestamp || server timestamp. If there is a need to set a new path to the truck then M7 = CPU timestamp || server timestamp || PATH.
20. The server generates another K_{S2Old} using PRNG, but this time it switches to (K_{2Old}, K_{1Old}) that will produce another stream cipher to encrypt M7.
21. The server encrypts the message M8 = $K_{S2Old} \oplus$ M7 and sends it to the CPU.
22. The server updates the CV value to be CV = CV + 1.
23. The CPU receives M8 from the server and generates K_{CPU2} using PRNG and (K_{S2}, K_{S1}) to decrypt M8.
24. The CPU checks the validity of the server timestamp and, if the CPU timestamp is the same, it makes sure that the message is sent from the server. Also, in case of receiving a new path from the server the truck needs to set new truck path. If everything is satisfactory, then the truck moves to step 25; otherwise, the communication is closed.
25. The CPU prepares M9 = Meta-id$_{T2}$ for the TS and M9 = Meta-id$_{E2}$ for the E-Seal.

26. The CPU generates another K_{CPUTS3} using PRNG, but this time it switches to (K_{T2}, K_{T1}) that will produce another stream cipher to encrypt M9 for the TS. Also, The CPU generates another K_{CPUE3} using PRNG, but this time it switches to (K_{E2}, K_{E1}) that will produce another stream cipher to encrypt M9 for the E-Seal.
27. The CPU encrypts and sends $M10 = K_{CPUTS3} \oplus M9$ to the TS. Also, The CPU encrypts and sends $M10 = K_{CPUE3} \oplus M9$ to the E-Seal.
28. The TS gets M10 and generates K_{TS2} same length as M10 using PRNG with (K_{T2}, K_{T1}) to decrypt M10. Also, the E-Seal gets M10 and generates K_{E2} same length as M10 using PRNG with (K_{E2}, K_{E1}) to decrypt M10.
29. The TS checks if the Meta-id from CPU = Meta-id$_{T2}$. Also, E-Seal checks if the Meta-id from CPU = Meta-id$_{E2}$.Ifit is correct then move to step 30 else close the connection.
30. All of the TS, E-Seal, CPU and Server use the CA update algorithm to update their stored such as TS, E-Seal use CA algorithm to update Keys and Meta-ids. Where, the server uses CA algorithm to updates the New Keys and Meta-ids, where the old value of the keys and Meta-id is the same as the new value of the final session.

5 Security Analysis of the Proposed Protocol

The proposed protocol can guarantee the mutual authentication requirements that were explained previously in the introduction, which are the secrecy, originality, freshness and integrity of the messages. The first requirement is the secrecy of the messages between the truck and the server, which can be achieved by encrypting messages between them using the proposed algorithm as explained before in Sect. 3. Additionally, the originality of the messages can be ensured because both the CPU and the server exchange secret keys between them, as well as encrypting/decrypting the messages that were relayed between them successfully. The same goes for the TS and E-Seal as the messages that are sending between them and the CPU are encrypted using another shared secret between the CPU and the TS, and the CPU and the E-Seal. Additionally, the protocol can ensure the freshness of the messages using the timestamp. The last requirement is the integrity, which can be achieved using the CRC (the CRC contained in the IP).

In addition, the protocol can resist the previously mentioned attacks, which can be divided into two parts as follows:

1. **Solutions to attacks on the truck used to transport the radioactive sources**, such as cloning, spoofing and physical attacks, are as follows:
   Cloning attack: To prevent cloning attacks, the protocol uses in total six shared secret keys, which are two shared secret between TS and CPU (K_{T1}, K_{T2}), two shared secret between E-Seal and CPU (K_{E1}, K_{E2}) and two other shared secret keys between the CPU and the server (K_{S1}, K_{S2}). Also, between the CPU and the server there is a timestamp, and between the CPU and TS/E-Seal there are agreed Meta-ids values. In M6, the CPU uses a timestamp that is encrypted using K_{CPU1} to authenticate the CPU. The server checks the validity of the timestamp and then replies with M8, which encrypts the timestamps of the CPU and the server so that the CPU makes sure that the message comes from the server and the server authenticates itself to the

CPU using the K_2 as an encryption key. Also, the CPU authenticates itself to the TS/E-Seal using Meta-idT2 and Meta-idE2 that are stored in both of the CPU and TS/E-Seal.
Spoofing attack: The protocol can prevent this attack using the PRNG algorithm to encrypt the messages that are sent between the communication entities (TS, E-Seal, CPU and Server). Thus, even if the attacker captures the sent messages, he/she will not be able to recognize them.

2. **Solutions to attacks on the communication between the truck and the central tracking server**. These attacks can be the Tracing, desynchronization attack, Man in the Middle (MITM) attack and Replay attack as follows:
Tracing: The protocol resists tracing attacks using the Meta-id, which is not connected to the truck's ID. This can provide truck anonymity because the Meta-id is a pseudonym that is randomly chosen for each tag/WS/E-Seal. The value of the Meta-id is then updated after each successful authentication, which identifies the next communication between the CPU and the server.
Desynchronization attack: The attacker can launch a desynchronization attack on the CPU by increasing the CPU's Counter Value (CV) on the server side. However, the server specifies a time window for the validity of the CPU's CV such that the DB will accept the CPU only within the specific time window which is CV [0, 2]. Furthermore, the server stores both the previous and the new values that are used in the communication between the truck and the server. Consequently, this can be used as a backup plan that may help to increase the possibility of the system availability. Thus, even if the attacker is able to prevent the last message from updating new values (Steps 15/21), the protocol will still be running and DB will still be able to accept the CPU's old values.
MITM attack: First, the shared secret keys are exchanged in a secure manner. Thus, the four communicating parties can securely exchange the encrypted messages. In addition, the encryption of the messages can help to prevent the man in the middle attack. Additionally, the two parties use the challenge and response technique to ensure that the messages sent between the two parties are coming from legitimate parties and not modified by a third party because the message can be decrypted correctly by both parties.
Replay attack: The timestamp is also used to identify each protocol session between the CPU and the server. Thus, in this case, if the attacker is able to capture the previous message and wants to send it back to the server, and the timestamp is not sufficiently recent in the current time window, then the message will be rejected. Note that devices require secure synchronized clocks. Also, between the TS, E-Seal and the CPU there are agreed two Meta-ids that identify each communication between them. Also, the Meta-id values are updated after each successful authentication protocol.

In addition, the protocol can reduce the computational load on the server side using a Meta-id that identifies the truck in the backend database with O(1) computational complexity. The server can directly use the Meta-id to locate the corresponding entry in its database and perform necessary computations for this matched entry only. Thus, the four communicating parties can securely exchange the encrypted messages. In addition, the encryption of the messages can help to prevent the man in the middle attack. Additionally, the two parties use the challenge and response technique to ensure that the messages sent between the two parties are coming from legitimate parties and not modified by a third party because the message can be decrypted correctly by both parties. The next subsection shows the protocol formal methods verification using the Scyther tool.

5.1 The Proposed Protocol Verification Proof with Scyther

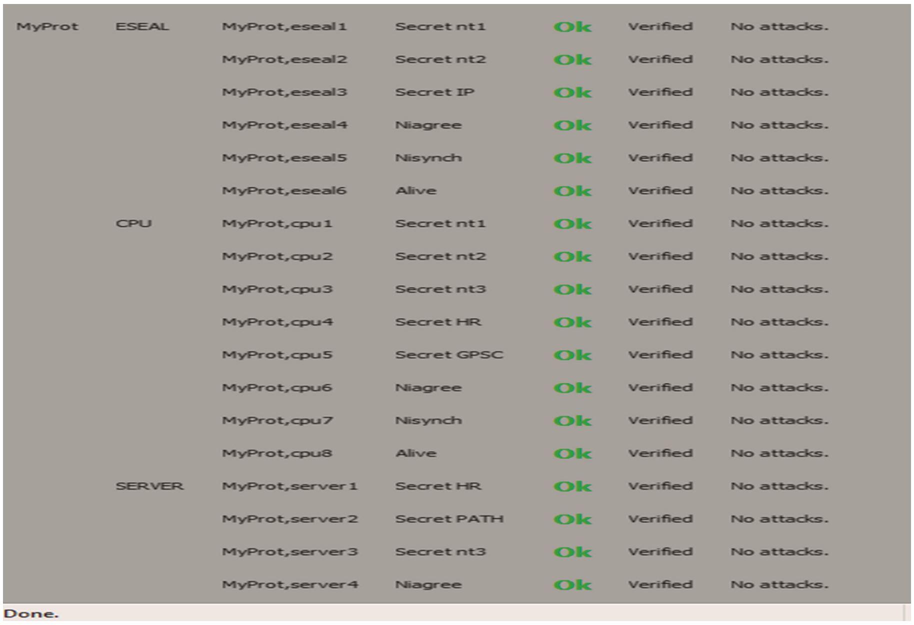
To verify our protocol, we analyzed it using the protocol analyzer tool called Scyther [24]. Also, Scyther is a tool that can be used to analyze the security protocols under what is called the perfect cryptography assumption. The perfect cryptography assumption assumes that all cryptographic functions are perfect; for example, the attacker knows nothing from an encrypted message unless he knows the decryption key. The tool can be used to discover problems that arise from the way the protocol is assembled. In general, this problem is undecipherable, but in practice many protocols can be proven correct or attacks can be found. The main feature of the Scyther tool is that it is the only tool that can verify the synchronization [25]. The protocol synchronization means that the messages are transmitted exactly as was agreed upon by the protocol description.

The Scyther tool can verify the requirements of the mutual authentication protocol which are secrecy, integrity, fresh and originality. The first claim ensures the secrecy and integrity of the messages because the key used in the communication is secret, where the second claim ensures the originality of the messages. The third claim ensures the integrity as well because the communication between the two parties is continuous and they are able to answer each other with the encrypted messages. Additionally, the fourth claim shows that the two parties of the system communicate after receiving a message from each other. Thus, the availability and the freshness of the system can be guaranteed based on the fresh nonce that are used in the protocol model. Note that the default values provided by Scyther are used in the verification.

Figure 2 shows the results of verifying our protocol using Scyther. Our protocol passed all the claims.

Moreover, these results prove that the proposed protocol can be used to secure the communication in our system.

Table 1 shows a comparison between our protocol and the other protocols that are considered as lightweight mutual authentication protocols and described in Sect. 5. As shown in the table, our protocol provides all the security requirements, which are the ability to resist tracing, the replay attack, the desynchronization attack and the MITM attack. In addition, the protocol provides less computation load on the tag and server than other protocols using the Meta-id, which reduces the server load to O(1) instead of searching all the entries of the database.



```
MyProt   ESEAL    MyProt,eseal1     Secret nt1     Ok   Verified   No attacks.
                  MyProt,eseal2     Secret nt2     Ok   Verified   No attacks.
                  MyProt,eseal3     Secret IP      Ok   Verified   No attacks.
                  MyProt,eseal4     Niagree        Ok   Verified   No attacks.
                  MyProt,eseal5     Nisynch        Ok   Verified   No attacks.
                  MyProt,eseal6     Alive          Ok   Verified   No attacks.
         CPU      MyProt,cpu1       Secret nt1     Ok   Verified   No attacks.
                  MyProt,cpu2       Secret nt2     Ok   Verified   No attacks.
                  MyProt,cpu3       Secret nt3     Ok   Verified   No attacks.
                  MyProt,cpu4       Secret HR      Ok   Verified   No attacks.
                  MyProt,cpu5       Secret GPSC    Ok   Verified   No attacks.
                  MyProt,cpu6       Niagree        Ok   Verified   No attacks.
                  MyProt,cpu7       Nisynch        Ok   Verified   No attacks.
                  MyProt,cpu8       Alive          Ok   Verified   No attacks.
         SERVER   MyProt,server1    Secret HR      Ok   Verified   No attacks.
                  MyProt,server2    Secret PATH    Ok   Verified   No attacks.
                  MyProt,server3    Secret nt3     Ok   Verified   No attacks.
                  MyProt,server4    Niagree        Ok   Verified   No attacks.
Done.
```

Fig. 2. Scyther verification proof for the proposed protocol

Table 1. Comparison between the proposed protocol and others

	Chien [15]	Lo and Yeh [17]	Yeh [20]	Yoon [22]	Amin [23]	Proposed protocol
Resistance to tracking	No	No	Yes	Yes	Yes	**Yes**
Resistance to replay attack	Yes	Yes	Yes	Yes	Yes	**Yes**
Desynchro nization attack	Yes	Yes	Yes	Yes	No	**Yes**
Resistance to MITM	Yes	Yes	Yes	Yes	No	**Yes**
Server load	3 CRC, 2 PRNG	8 PRNG, 8 CRC	7 PRNG, 1 Hash	8 PRNG, 1 Hash	13 PRNG, 1 Hash	**4 PRNG, 3 CA**
Reader load	1 PRNG	1 PRNG	1 PRNG, 1 Hash	1 PRNG, 2 Hash	1 PRNG, 2 Hash	**-**
Tag load	3 PRNG, 3 CRC	6 PRNG, 3 CRC	6 PRNG	6 PRNG	7 PRNG	**2 PRNG, 3 CA**

Furthermore, our protocol uses tag load less than the other protocols in the table with running 2 times PRNG, 3 times CA (Cellular Automata) [14] and 1 CRC for the E-Seal only not for the TS (TS does not need CRC). Also, for the server side our protocol run the PRNG 4 times and CA for 3 times to update the keys and Meta-id new value which is considered less than the rest except Chien and Chen protocol that uses 2 times PRNG and 3 times CRC. Thus, in general we achieved our aims of reducing the computation load and providing security requirements. This also proves that our protocol is lightweight, as it does not consume as much computation as the other protocols that are compared in Table 1 based on Server load, Reader load and Tag load.

6 Conclusions

We have introduced a new lightweight mutual authentication protocol that can secure the communication involved in transporting radioactive materials. This paper analyses the threats that surround the transportation of the radioactive material and then emphasizes the importance of providing a mutual authentication protocol that can secure the communication during the transmission. Our protocol can resist the available attacks, such as the tracing, cloning, spoofing attacks, desynchronization attack, MITM attack and Replay attack. In addition, the proposed mutual authentication protocol is a lightweight solution that does not add too much computational load on the communication parties, such as server and tags because the protocol uses less PRNG calculations than the other lightweight protocols. Additionally, the proposed solution is strengthened by adding a robust PRNG to generate the keys of the communication.

The proposed mutual authentication protocol is verified using the Scyther tool, and the protocol has passed all five claims. Then, a simulation of the tracking system that implements the proposed protocol is provided to ensure its ability to secure the communication within the tracking system. The system used SMS messages to communicate with the size of the encrypted message, which is only 42 bytes. Additionally, the system can be customized to send information about the radioactive materials in a short time; thus, the data are sent very quickly between the communicating parties (for example, within 1.5 μs). In the future, we are going to implement our solution in a real life application. Also some future work will be done to evaluate the trade-off with other approaches such as: TPM (Trusted Platform Module), and SSL/TLS mutual authentication protocols. Also we would like to investigate if the designed protocol could be used in different scenarios.

Acknowledgements. The authors wish to acknowledge Information and Communication Technology Fund (ICT Fund) for the continued support for the educational development and research.

References

1. Radioactive Sources: Uses, Safety, and Security, Australian Nuclear Science and Technology Organization, August 2016. http://www.arpansa.gov.au/
2. Sheldon, F.T., Walker, R.M., Abercrombie, R.K., Cline, R.L.: Tracking radioactive sources in commerce. In: WM 2005 Conference, Tucson, AZ, USA, 27 February–3 March 2005

3. Security of radioactive sources Interim guidance for comment, Printed by the IAEA in Austria, August 2016. http://www-pub.iaea.org/MTCD/Publications/PDF/te_1355_web.pdf/
4. Federal Authority for Nuclear Regulation, August 2016. http://www.fanr.gov.ae/en/Pages/default.aspx
5. Ya-Anant, N., Tiyapun, K., Saiyut, K.: Radiological accident and incident in Thailand: lesson to be learned. Radiat. Prot. Dosim. **146**(1–3), 111–114 (2011)
6. Security in the Transport of Radioactive Material, IAEA nuclear security series No. 9, Vienna, August 2016. http://www-pub.iaea.org/MTCD/publications/PDF/Pub1348_web.pdf/
7. Radiological Dispersal Device (RDD): Argonne National Laboratory, EVS, Human Health Fact Sheet, August 2005
8. Misra, P., Enge, P.: Global Positioning System: Signals, Measurements, and Performance, 2nd edn. Ganga-Jamun Press, Massachusetts (2010)
9. Schwieger, V.: Positioning within the GSM network. In: Proceedings on 6th FIG Regional Conference, San Jose, Costa Rica, 12–15 November 2007
10. Hunt, V., Puglia, A., Puglia, M.: RFID: A Guide to Radio Frequency Identification, 1st edn. Wiley, Hoboken (2007)
11. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Commun. Mag. **40**(8), 102–114 (2012)
12. Shemaili, M.B., Yeun, C.Y., Mubarak, K., Zemerly, M.J., Chang, Y.S.: Securing E-seal real time tracking system for internet of things. In: Proceedings of 8th International Conference on Internet Technology and Secured Transactions, 9–12 December 2013, London, UK, pp. 65–69 (2013)
13. Yeun, C.Y., Shemaili, M.A.B., Zemerly, M.J., Mubarak, K., Yeun, H.K., Chang, Y.S.: ID-based secure real-time tracking system. Int. J. Adv. Logist. **4**(2), 100–114 (2015)
14. Yeun, C.Y., Shemaili, M.A.B., Mubarak, K., Zemerly, M.J.: A new lightweight hybrid cryptographic algorithm for the internet of things. In: Proceedings of 7th International Conference on Internet Technology and Secured Transactions, 10–12 December 2012, London, UK, pp. 87–92 (2012)
15. Chien, H.Y.: SASI: a new ultra-lightweight RFID authentication protocol providing strong authentication and strong integrity. IEEE Trans. Dependable Secur. Comput. **4**(4), 337–340 (2007)
16. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J., Ribagorda, A.: Cryptanalysis of a novel authentication protocol conforming to EPC-C1G2 standard. Comput. Stand. Interfaces **31**(2), 372–380 (2009)
17. Lo, N.W., Yeh, K.-H.: An Efficient Mutual Authentication Scheme for EPCglobal Class-1 Generation-2 RFID System. In: Denko, M.K., Shih, C.-s., Li, K.-C., Tsao, S.-L., Zeng, Q.-A., Park, S.H., Ko, Y.-B., Hung, S.-H., Park, J.H. (eds.) EUC 2007. LNCS, vol. 4809, pp. 43–56. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77090-9_5
18. Chen, C.L., Deng, Y.Y.: Conformation of EPC class 1 generation 2 standards RFID system with mutual authentication and privacy protection. Eng. Appl. Artif. Intell. **22**(8), 1284–1291 (2009)
19. Habibi, M.H., Gardeshi, M., Alaghband, M.R.: Practical attacks on a RFID authentication protocol conforming to EPC C-1 G-2 standard. J. Ubicomp **2**(1), 1–13 (2011)
20. Yeh, T.C., Wang, Y.J., Kuo, T.C., Wang, S.S.: Securing RFID systems conforming to EPC class 1 generation 2 standard. Expert Syst. Appl. **37**(12), 7678–7683 (2010)
21. Safkhani, M., Bagheri, N., Sanadhya, S.K., Naderi, M.: Cryptanalysis of improved Yeh et al.'s authentication protocol: an EPC class-1 generation-2 standard compliant protocol, IACR Cryptology ePrint Archive, Report. 2011/ 426 (2011)

22. Yoon, E.J.: Improvement of the securing RFID systems conforming to EPC class 1 generation 2 standard. Expert Syst. Appl. **39**(12), 1589–1594 (2012)
23. Mohammadali, A., Ahmadian, Z., Aref, M.R.: Analysis and improvement of the securing RFID systems conforming to EPC class 1 generation 2 standard, IACR Cryptology ePrint Archive, pp. 66–76 (2013)
24. Cremers, C.J.F.: The Scyther tool: verification, falsification, and analysis of security protocols. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 414–418. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70545-1_38, http://www.cs.ox.ac.uk/people/cas.cremers/scyther/
25. Cremers, C., Mauw, S., de Vink, E.: Injective synchronisation: an extension of the authentication hierarchy. Theor. Comput. Sci. **367**(10), 139–161 (2006). Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. **147**, 195–197 (1981)