Secure Object Tracking Protocol for the Internet of Things

Biplob R. Ray, Member, IEEE, Morshed Chowdhury, Member, IEEE and Jemal Abawajy, Senior Member, IEEE

Abstract—In this paper, we propose a secure object tracking protocol to ensure the visibility and traceability of an object along the travel path to support the Internet of Things (IoT). The proposed protocol is based on Radio Frequency Identification (RFID) system for global unique identification of IoT objects. For ensuring secure object tracking, lightweight cryptographic primitives and Physically Unclonable Function (PUF) are used by the proposed protocol in tags. We evaluated the proposed protocol both quantitatively and qualitatively. In our experiment, we modeled the protocol using Security Protocol Description Language (SPDL) and simulated SPDL model using automated claim verification tool Scyther. The results show that the proposed protocol is more secure and require less computation compared to existing similar protocols.

Index Terms—Internet of Things, Secure object tracking, RFID.

I. INTRODUCTION

The Internet of Things (IoT) is a paradigm where all the things (objects, people and so on) around us can globally and actively identify, connect, sense and report itself to the system. This requires seamless unique identification of each things in IoT. Many applications such as a supply chain, a health care management system, a smart home and so on require working collaboratively to achieve global object identification which is one of the main business goals of the IoT [1, 2, 3, 4]. As RFID does not require a line of sight to identify an object and can identify many objects at a time, it is an ideal system for object identification in IoT. An object in IoT may need to report to a number of partners before it reach to its owner or user [1, 5]. Hence, the objects in a global networked chain might travel backwards from the user to the transporter and finally to the supply chain for replacement or repair.

The IoT system needs to verify the travel path and ensure on-site tracking of an object’s movement to ensure connectivity and visibility [1]. It is important also to collect information on the status of the object along the path to improve forecasting accuracy, which helps to formulate strategies, increase visibility and reduce object transit time for users [6, 1]. However, in order to achieve these outcomes, the IoT system has to share and exchange information between objects and partners from a number of administrative domains along the path via wireless media. A secure object tracking protocol should ensure that an adversary should not be able to compromise the privacy of the partners, the current owner or the objects while tracing and tracking things globally [7, 8, 9, 10, 11]. It should also protect the tag from cloning and ensure the non-repudiation property of the system to provide accountability of the system’s actions [7]. Although few protocols that address the tracking, tracing and path verification of an object exist, they have either security vulnerabilities or not feasible to implement even in existing EPC active tags [8, 9, 10, 11]. In addition, existing tracking protocols did not clearly address non-repudiation, injection of fake objects and unclonability issues while tracking the object along the path.

In this paper, we propose a secure tracking protocol to ensure the visibility and the traceability of an object along the path. The proposed protocol also ensures the correctness of the travel-path while protecting the privacy of the current owner, the partners and the users. Our main contributions are:

• A secure tracking protocol for IoT to improve objects visibility and traceability for users along the travel path.
• Security assurance of the IoT system by ensuring non-repudiation, privacy protection for the system and users.
• Integration of PUF (Physically Unclonable Function) to prevent injection of fake objects and cloning.
• Modeling of the protocol using SPDL and security claim verification using Scyther to eliminate attacks.

The remainder of the paper is organized as follows. In Section II, the background and related work are presented. In Section III, the system model and initialization process are detailed. The proposed secure tracker protocol is presented in Section IV. The analysis of the protocol is detailed in Section V. This is followed by the conclusion in Section VI.

II. BACKGROUND

In this section, we formalize and define system components followed by a review of related work to contextualize and underpin the necessity of the proposed work.

A. Networked RFID System and IoT

The Networked RFID System (NRS) makes it possible to perform object identification between widely distributed partners [6]. A collection of NRSs (such as a supply chain, a health care management system, a smart home and so on) require working collaboratively to achieve global object identification for IoT [1, 2, 3, 4]. Fig. 1 illustrates a modified version of NRS presented in [12, 13] with two independent business partners that have authorized access to the backend system and some RFID tagged IoT objects. In this system, IoT objects can move forward and backward from one business entity to the other. The RFID-enabled supply chain network is an example of NRS that requires forward and backward
The main security and business objectives of Tracker protocol along the network chain of the tags on-site while ensuring privacy, non-repudiation, and check oracles securely to verify the authenticity and tracking order, a secure tracker protocol performs a read, write and check oracles defined in [16], we define a secure tracker protocol as follows:

Definition 3 (Tracker protocol): Given a set of tags that may move, a Tracker protocol consists of a read, write and check oracles defined in [16], we define a secure tracker protocol as follows:

Definition 3 (Tracker protocol): Given a set of tags that may travel between a set of NRS. A Tracker protocol performs a read, write and check oracles to verify the authenticity and tracking of the tags on-site while ensuring privacy, non-repudiation, protection from the injection of fake objects as well as visibility among the network chain.

The main security and business objectives of Tracker protocol are:

- To verify the authenticity (to protect the system from injection of fake objects) of the tag and its travel path.
- To track and trace the tag along the path.
- To ensure privacy of the tag, partners and current owner.
- To protect non-repudiation security property of the system.

Our protocol utilizes the Physically Uncloneable Function (PUF) [17] to prevent injection of fake tags by an adversary. For our implementation, we will use a generalized definition of PUF as stated in definition 3 which is adapted from [18].

Definition 4: Let \( \ell \in \mathbb{N} \) be a security parameter, \( \gamma, \kappa \in \mathbb{N} \) be a polynomial, bounded by \( \ell \) and \( f: [0,1]^\gamma \rightarrow [0,1]^\kappa \) be a function. An ideal PUF is a function PUF(\( c \)) with the following properties:

1. For all challenges \( c \in [0,1]^\gamma \) and all tuples \( (r_i, r_j) \in \text{PUF}(c)^2 \), probability \( \Pr[ r_i = r_j ] = 1 \)
2. Any attempt to physically tamper with the implementation of \( \text{PUF}(c) \) for the object \( i \) will result in the destruction of \( \text{PUF}(c) \) and create \( \text{PUF}(c) \), where \( \text{PUF}(c) \neq \text{PUF}(c) \).
3. The advantage of the adversary \( A \) to regenerate legitimate response \( r \) by interrogating the tag using a polynomial number of queries of challenge \( c \) is negligible.

Fuzzy Extractors [19] can be used to eliminate noise (e.g., temperature and/or supply voltage variations) impact on PUF.

B. Adversary model

The adversary (\( A \)) model used in this paper is based on the notion of a normal adversary defined in [7]. \( A \) is capable of monitoring all wireless communications (between tags and partner readers) and gathering side channel information using the CreatTag, DrawTag, Free, Launch and SendReader oracles [7]. In addition, \( A \) is capable of gathering some side channel information (except hardware based) using the modified “Result” oracle as defined below:

Result (\( \pi \)) oracle: \( A \) is capable of running a “Result” oracle to obtain time between sending and receiving the object (each pass), travel time between \( R_i \in \mathbb{R} \) and \( R_j \in \mathbb{R} \) where \( i \neq j \) to generate timing attacks. When \( \pi \) is complete, it returns 1 if Output \( \neq \perp \) and 0 otherwise. Note that \( \pi \) is limited to timing information only and explicitly excludes all types of hardware based information.

C. Literature Review

Most of the existing tracking protocol aims to increase visibility of an object in a specific NRS (such as supply chain) using tracking and tracing along the network. The work of Bi and Lin [6] identified the importance of finding the object along the path for tracking and tracing to minimize inventory holding costs, improve forecasting accuracy, formulate strategies and reduce transit time in supply chain. However, in the protocol [6], while tracking and tracing the object, the partner’s data was collected in an insecure mode. Quafi and Serge proposed a "path checker" object tracing and tracking protocol [8] that stores state information to the tag and uses key based hash to perform the information exchange between the tag and the reader. The protocol uses many different readers to authenticate and update a tag. Also, the tag has to follow the order of the path sequentially. The protocol assumes tags are tamper resistant and capable of anti-counterfeit mechanisms which are not valid for RFID tags and thus only secure on a weak advisory.
model scenario. Also, the path of an object can be checked at the end of the destination only.

The protocol in [9] uses a proxy re-signature to allow path segment verification while using read/write only tags, where the tags store the signature of the last trusted party it has visited. The protocol does not address the problem of implementing practical proxy re-signatures without a trusted third party. The object authenticity verification protocol in [10] validates the authenticity of an object through its travel path. The proposal requires a centralized trusted party called a “manager” to carry out path verification. Verification can only be performed once the tag arrives at the manager. This limits the wide deployment of such a solution in a context where partners do not trust each other and demand to be able to verify product authenticity in real-time “on-site”. The protocol in [11] used polynomial based encoding to represent the path in a supply chain. Path validity can be checked in each partner by readers. However, a malicious reader may compute an incorrect signature for the tag and write it on the tag. This will make it impossible to track the tag. The protocol is also weak regarding a cloning attack. In this protocol [11] readers have heavy computational overhead and require costly tags (120 Bytes storage). The path-checking solution [20] and path-checking protocol [21] aim to combine authentication and path checking to ensure tracking and tracing of the tag in a supply chain. To make these proposals possible, all the partners in the network chain need to know the full path and store this information in the database. In addition, the tag receives the information of its next destination (the updated path) from its current partner. This makes the protocol [20, 22] vulnerable if there is a semi-trusted partner in the chain. A semi-trusted partner might pass the incorrect next path to the tag to get a business advantage or to generate an attack on the next partner. If the partner is a competitor of the business then it might obtain an illegitimate business advantage by knowing the full path of the tag.

The survey above has illustrated that existing tracking protocols are too heavy to implement on existing low cost RFID systems using passive tags. In addition, they are all vulnerable to tampering, physical cloning attacks and ignored non-repudiation security requirement. None of the existing proposals fulfill the requirement of a tracker protocol as stated in Definition 3, which rationalizes our attempts to propose a tracker protocol.

III. SYSTEM MODEL AND INITIALIZATION PROCESS

This section present the system model and initialization process of the proposed protocol.

A. System Model

The IoT system in Fig. 2 illustrates the system model used in this paper. In Fig. 2, a current owner (CO) owns a tag (Τᵢ ∈ T) that travels to a number of partners to achieve a business objective. The current owner (CO) keeps track of the tag along the travel path via partner readers. Let P,IDᵢ be the unique ID of partner Pᵢ. The system calculates path code (PC) using Equation 1 where n number of partner’s PID is XORED and then hashed.

\[ PC = h(P,ID₁ \oplus ... \oplus P,IDₙ) \]  

Let R,IDₐ be the unique identification of the current owner reader. The system calculates a common challenge value (c) by performing hash operation on XORed value of the path code (PC) and current owner reader’s ID (R,IDₐ) in Equation (2).

\[ c = h(PC \oplus R,IDₐ) \]  

The partner specific challenge text (cᵢ) is calculated by XORing the generic challenge (c) with partner’s unique ID (PIDᵢ) in Equation (3).

\[ cᵢ = c \oplus PIDᵢ \]  

The partners have a shared secure communication link with the current owner over a cloud using conventional secure communication techniques such as IPSec. In addition, the partners are semi-trusted. The tagged objects can travel to a partner (Pᵢ ∈ P) in any order but can only travel once in a specific journey. Both the current owner and partner readers are capable of executing a group of Diffie-Hellman algorithm to generate a shared secret key (kₗPIDᵢ). They also exchange information to agree on a symmetric encryption algorithm that will be used in the protocol execution. Partners need to register themselves to current owner using their unique PID to become a semi-trusted partner. Each registered partner (P) receives a tuple below from current owner (CO) to securely execute the tracker protocol where n represents total number of partners the tag may travel throughout the path.

\[ P = (c, n) \]

The tag has a PUF () function and can perform a XOR and h() function to support the required cryptographic operations. Both PUF() and h() functions have been practically implemented in EPC passive tags [17, 23] that only require 700 and 1500 logic gates respectively on a tag. All tags have a 6 bits user memory to store the travel state ‘st’ where length(st) = n is based on the total number of partners(n). This memory is password protected and can be read by current owner(s) after proving their ownership verification. The tag itself is the only one who can write on this protected user memory. Initially, all the bits in the protected memory are populated by ‘OFF’ = 0 bit. Upon successful mutual verification of the travel status between the current owner, partners and the tag, the tag will set a ‘st’ bit ‘ON’ = 1 in a certain position (po). The position value is determined by the lth value of the partner so that po = i. Let TKᵢ be the unique ticket for a specific travel path of the tag (Τᵢ) calculated by XORing path code (PC) with the tag code TCᵢ as shown in Equation (14) of Fig. 3, (c₁ ... cₙ) is the set of challenge codes calculated using Equation (3) for n number of partners along the travel path, a response (rᵢ) from tag’s PUF for challenge cᵢ is calculated using Equation (6) of Fig. 3 and a tag code TCᵢ is calculated using tag’s PUF and tag ID (TIDᵢ) as shown in Equation (7) of Fig. 3. Each tag stores an initial tuple below for a specific travel path.

\[ Tᵢ = (E(TKᵢ), TIDᵢ, st, (c₁ ... cₙ), n) \]

The backend of the current owner stores the tuple COΤᵢ for each tag and a tuple COPC for the path a tag may travel where kₗPIDᵢ is the secret key shared between current owner and the tag (Τᵢ).

\[ COΤᵢ = (TIDᵢ, st, TCᵢ, kₗPIDᵢ, rᵢ) \]
IV. DETAILS OF THE TRACKER PROTOCOL

In this section, we detail communication and computation process of the proposed protocol. In Fig. 3, we summarize the communication process between the tag, partners and the current owner of the proposed protocol. There are 5 communication steps in the protocol as outlined in Fig. 3.

Step 1) The protocol is invoked when a valid partner senses a tag for the first time in its vicinity. The partner sends the challenge value specific to the tag (c₁) to the tag to authenticate itself. A partner (Pᵢ) uses c, n and its PIDs to calculate cᵢ using Equation (4) of Fig. 3 where PRF() is a pseudorandom function. Note that cᵢ is the random challenge value specific to the Pᵢ. The tag uses its own pre-stored challenge code (c₁) and total partner numbers n to verify the correctness of received challenge value cᵢ from partner (Pᵢ) using Equation (5).

\[ (c₁ \oplus PRF(n)) \oplus cᵢ \]

If Equation (5) returns true, then the tag is communicating with a genuine partner (P₁) and travelling valid path.

Step 2) The tag then generates a response value (r₁) using \[ PUF(cᵢ) \] and c₁ in Equation (6). This process creates a challenge–response pair (cᵢ, r₁). The tag also creates a tag code (TCᵢ) as shown in Equation (7).

\[ r₁ = PUF(cᵢ) \]
\[ TCᵢ = PUF(TIDᵢ) \]
\[ resp₁ = h(cᵢ, r₁, TCᵢ) \oplus vst \]

Step 3) The partner Pᵢ encrypts the received response one (resp₁) value using a secret shared key \[ k^{Pᵢ}_{CO} \] as shown in Equation (10). It uses the symmetric encryption algorithm that was negotiated between current owner (CO) and partner (Pᵢ) in the early stage of the protocol.

\[ resp₂ = E(resp₁)^{Pᵢ}_{CO} \]

The partner then sends the response two (resp₂) to the current owner to verify the tag and the partner’s authenticity.

Step 4) The current owner decrypts the received response two (resp₂) value to retrieve response one (resp₁) value as shown in Equation (11).

\[ D(resp₂)^{Pᵢ}_{CO} = resp₁ \]

The current owner then calculates response one prime (resp₁) using Equation (12) with virtual travel status (vst) which is calculated using Equation (9) and pre-stored values (r₁, cᵢ, TCᵢ).

\[ resp₁ = h(cᵢ, r₁, TCᵢ) \oplus vst \]

The response one prime (resp₁) is compared with the retrieved response one (resp₁) in Equation (13).

\[ resp₁ = resp₁ \]

If Equation (13) returns true, then the system can be sure that the tag is genuine as tag code (TCᵢ) is used to prepare resp₁. This also ensures the authenticity of the partner as partner’s unique ID (PIDs) is used to prepare resp and validated in
We have created a virtual machine with an Ubuntu operating the tag, partners and the owner reader of the proposed protocol. Secure Object Tracking Protocol for the Internet of Things correctness of the tag’s travel status and correctness of its travel path as virtual travel status (vst) and path code (PC) are used to calculate response one (resp₁) and common challenge value (c) respectively. At this point, the current owner creates a ticket (TKₐ) in Equation (14) where pre-stored travel path code (PC) and tag code (TCᵢ) of Tᵢ is used.

\[ TKₐ = (PC ⊕ TCᵢ) \] (14)

The current owner encrypts TKᵢ using shared secret key of the tag and the current owner. It then XOR the encrypted value with vst to create resp₂ as shown in Equation (15). The ticket (TKᵢ) is used in tag to verify partner and current owner’s authenticity.

\[ resp₂ = E(TKᵢ)_{C₀} ⊕ vst \] (15)

The current owner calculates XOR between resp₂ and resp₂ before encrypting it using partner and current owner’s shared secret key to create resp₃ in Equation (16).

\[ resp₃ = E(TKᵢ)_{C₀} ⊕ vst \] (16)

The current owner sends response four (resp₄) to Pᵢ to finalize the protocol. It also updates the status value for the tag at backend by adding an “ON” bit in iᵗʰ position (po = i) of the st. It also records that the tag has travelled to the partner Pᵢ to prevent travel duplication.

Step 5) The partner decrypts the received resp₄ and retrieves resp₃ as shown in Equation (17). It then sends the response three (resp₃) to the tag to let tag verify current owner’s identity.

\[ resp₃ = D(resp₄)_{C₀} = (resp₃ ⊕ resp₂) ⊕ resp₂ \] (17)

The tag calculates response three prime (resp₃) using the stored encrypted ticket value \( E(TKᵢ)_{C₀} \) and virtual status value (vst). It then validates response three (resp₃) in Equation (18).

\[ resp₃ = resp₃ \] (18)

If Equation (18) returns true, then the tag updates status by adding an “ON” bit in the iᵗʰ position (po = i) of st. The execution of this protocol lets the current owner, partner and tag perform mutual authentication. It also lets current owner track the tag and verify the travel path of the tag while protecting the required security of the system and users. The tag’s travel status is updated in both the tag and in the current owner’s backend.

V. EXPERIMENT SECURITY ANALYSIS AND COMPARISON

In this section, we detail the proposed protocol’s security claim verification using a Scyther model checking tool which is followed by security analysis of the protocol. We also present a performance analysis and comparative study of the proposed protocol with similar recently proposed protocols.

A. Protocol’s Claim Verification

We have used Scyther [24, 25] as a claim verification tool to verify the security claims to protect communications between the tag, partners and the owner reader of the proposed protocol. We have created a virtual machine with an Ubuntu operating system in a VMware Player that has version 4.0.0 (build-471780). We are using Scyther V.1.0 for our experiment. Table II presents parameters that are used in this experiment.

We modeled our proposed protocol and adversary model using the Security Protocol Description Language (SPDL) [26]. Every micro details, such as XOR function, PUF () function, and the dataflow of our proposed protocol were used to build a realistic model of the proposed protocol. Scyther analyzes the SPDL protocol model and provides a finite representation of all traces that contain an execution of the protocol role [15]. If there is only one trace pattern for a role then all traces that include the initiator role correspond to a valid protocol execution [26]. Thus, any synchronization claim at the end of the initiator role is correct [26]. As illustrates in Fig. 4, our protocol execution was valid as only one trace pattern was found for each involved protocol’s role.

Table III, Fig. 5 and Table IV summarize our experiment outcome from Scyther. In Table III, we see that the proposed protocol meets Niagree, Nisynch, Alive and Weakagree. There are no attacks found in the protocol execution while satisfying Niagree, Nisynch, Alive and Weakagree. In Fig. 5, we have presented simulation result for each roles (Pᵢ, Tᵢ and CO).

![Fig. 4. Role characterization of the tracker protocol.](image1)

![Fig. 5. Claim verification result from Scyther.](image2)
involved in the proposed protocol. As we can see from the result in Fig. 5, no attacks found on Scyther’s automatic security claims. As illustrated in Table IV, the automatic claim verification of Scyther has also proved the secrecy of all the required values to ensure privacy. Due to limitations on space, we have not include screenshots of all Scyther output windows.

By doing this experiment, we have proven that our proposed protocol ensures security properties, such as the secrecy of all the exchanged values to provide privacy, aliveness of the protocols roles and transactions, authentication and traceability of the roles and their transactions, Non-injective agreement (Niagree), Non-injective synchronization (Nisynch) between roles, Aliveness and authenticity using Weakagree.

In Table V, we map the security objectives of the tracker protocol (according to Definition 3 in sub-section II.A) to the Scyther experiment output.

While doing so, we use the definitions of the Scyther experiment outputs (security properties) from [15, 24, 25, 26]. Table V shows, we verified all our security claims in this experiment.

The Scyther verification tool also assisted us to fix the flow of our protocol. During the design of the protocol, we found a number of attacks by running the protocol model in Scyther. We then redesigned our protocol to eliminate those attacks. One of the attack identified by Scyther simulation was falsified attack as illustrated in Fig. 6. In the attack graph of Fig. 6, a false current owner reader can fool a valid partner using falsification of exchange 4 from CO to T1. We investigated the attack graph of Fig. 6 and realized that we needed to include the ID of the current owner reader in our first exchange to eliminate this attack. Our initial data exchange frame between the partner and the tag is currently “send_1(P1,T1,\{add(c,P1)\})ci(T1)\)”. We added RID CO in Equation (2) and made some more amendments to eliminate other identified attacks. Our amended initial data exchange frame between the partner and the tag is currently “send_1(P1,T1,\{XOR(\{XOR(PC,RID)\}c,P1,PRF(n))\}ci(T1))\)”. As verified by the experiment, the security claims of the proposed protocol were successfully achieved while ensuring the business needs for the IoT. We will further analyse our proposed security protocol in the following subsection.

B. Security Analysis

The proposed protocol ensures that an adversary cannot compromise privacy (untraceability) of T1, P1 and CO by capturing some or all of transmitted information such as c′1, resp1, resp3 using unsecure channel. We prove this using Lemma 1 below.

**Lemma 1:** The protocol ensures privacy of all involved parties as all the transmitted values (c′1,resp1,resp2,resp3,resp4) are prepared by performing cryptographic primitive between two or more random and/or session values using Equations (4,8, 10, 15, 16, 17) and satisfies Equation (19) and (20).

\[ D(\text{or } E(\text{or } k_{T1}^{CO} \text{or } k_{CO}^{T1})) \rightarrow \text{resp}_2, \text{resp}_3, \text{resp}_4 \tag{19} \]

\[ h() \text{ or } \oplus \rightarrow \text{resp}_1, c_1 \tag{20} \]

**Proof:** The protocol satisfies Equation (19) as the transmitted values resp2, resp3, resp4 are encrypted using shared secrets in Equations (10), (16), (17). Without compromising secret session keys k_{T1}^{CO} or k_{CO}^{T1}, an adversary (A) cannot reveal the information of these exchanges to compromise privacy or untraceability. The protocol also satisfies Equation (20) as the values c′1 and resp1 are prepared using cryptographic primitive (such as hash and XOR) between random and session values (such as vst, PRF(n) and so on) to ensure the privacy of P1 and CO. By capturing the transmitted values, A cannot conclude that it is from a specific partner P1 or tag T1 or CO as none of these transmissions reveal identity of involved roles to an adversary. As a result, the privacy and untraceability of the tag, the current owner and the partner are protected.

Our proposed protocol ensures non-repudiation as proved in Lemma 2.

**Lemma 2:** The protocol ensures that activities within the protocol can be linked with the right roles. The proposed protocol guarantees this by satisfying the conditions below.

\[ !\exists \text{PID}_i; c_1 \forall \exists \text{TID}_i; \text{resp}_1 \]

\[ !\exists k_{T1}^{CO}; \text{resp}_2, \text{resp}_3 \text{ and } !\exists k_{CO}^{T1}; \text{resp}_4 \tag{21} \]

**Proof:** The conditions in Equation (21) will be satisfied if there exists at least one identification value which is uniquely associated to a role T1 or P1 or CO. In Equation (4) and (8), we have involved PIDi and TCi (which are associated with P1 and T1) required to generate correct c′1 and resp1. In Equation (10), (15) and (16), we use secret shared value k_{CO}^{T1} and k_{T1}^{CO} to encrypt. Without having the correct secret shared values, an adversary cannot calculate the correct values using Equation (10), (15) and (16). The involvement of shared secret keys ensures the association between calculated values and involved roles.

Lemma 3 below proves the protocol ensures that A will not be able to insert a cloned or fake tag to the system and make the system believe it is an authentic tag.

**Lemma 3:** The protocol prevents inserting an injection of fake tag as it satisfies conditions in Equation (13), (22) and (23).

\[ T_C \propto \text{PUF}(\text{TID}) \propto T_1 \tag{22} \]

\[ r_1 \propto \text{PUF}(c_1) \propto T_1 \tag{23} \]
Fig. 6. Falsified attached to compromise secrecy
secure Object Tracking protocol for the Internet of Things

**Proof:** According to properties 1 and 2 from definition 4 of PUF (detailed in sub-section II.A), a cloned tag $T_j$ will always have $T_{C1} \neq T_{C2}$ and $T_i \neq T_j$. This means a cloned tag will not be able to satisfy Equation (22) or (23). This means a clone tag can never authenticate itself as a valid tag. According to property 2 from definition 4 of PUF (detailed in sub-section II.A), a fake tag $T_j^{\text{TRID} \neq \text{TRID}}$ will always have $PUF_j(T_{\text{TRID}}) \neq PUF_j(T_{\text{TRID}})$ and $PUF_j(c_i) \neq PUF_j(c_i \text{ or } c_j)$. This means a fake tag can never satisfy Equation (22) or (23) or authenticate itself as a valid tag even if holds some correct values such as $T_{\text{TRID}}$. This prevents fake tag or cloned tag injection impossible in the system as long as PUF satisfies its properties.

One of the main objectives of the proposed protocol is to ensure tracking of the object along the path and verifying the correctness of the travel path. We prove both of these using Lemma 4 below.

**Lemma 4:** The protocol needs to satisfy Equations (24), (25), and (26) to ensure successful tracking of the tag along the path. It also needs to satisfy Equations (26) and (27) to assure the tag has travelled the correct path so far.

1. $resp_1 \propto T_{C1} \propto T_{ID_1} \propto T_i$ (24)
2. $resp_1 \propto c_i \propto c \propto R_{ID_{CO}}$ (25)
3. $resp_1 \propto c_i \propto P_{ID_1} \propto P_i$ PID$_i \in \text{PC}$
4. $\vee resp_1 \in \text{PC}$
5. $st_{i-1} \propto st_i^{\text{size}(st_i^{\text{ON}}) > \text{size}(st_{i-1}^{\text{ON}})}$ (26)
6. $st_{i-1} \propto st_i^{\text{size}(st_i^{\text{ON}}) > \text{size}(st_{i-1}^{\text{ON}})}$ (27)

**Proof:** Equation (24) is satisfied as the tag code ($T_{C1}$) is prepared using tag ID ($T_{ID_1}$) in Equation (7) and used in Equation (8) to prepare response one ($resp_1$). Partner specific challenge value $c_i$ is required to calculate $resp_1$ as shown in Equation (8). The Equation (25) is satisfied as the partner specific challenge value $c_i$ is calculated using generic challenge value ($c$) which is calculated in Equation (2) using path code (PC) and current owners ID ($R_{ID_{CO}}$). Finally, the protocol checks the validity of response one ($resp_1$) using Equation (13) by re-creating response one ($resp_1$) using backend database values. This proves the tag has travelled a partner ($P$) and successfully tracked by the current owner ($CO$). From Equation (1), we can say that $P_{ID_1}$ is an element of path code (PC) as $PC \propto P_{ID_1}^{\text{size}(P_{ID_1}) > \text{size}(st_i)}$.

Using Polynomial reduction computational complexity theory [27], we can conclude that $resp_1$ is a set of PC. This proves Equation (26). Upon successful validation of the current states and authenticity of the tag using Equation (13), the backend and the tag get their status value $st_i$ updated to recognize that it traveled the correct path. The update will replace a "OFF=0" bit of $st$ to "ON=1" bit and the tag validates this update using Equation (18) which satisfies Equation (27). This proves that so far the tag has traveled a valid path and is currently travelling to a valid partner.

**C. Comparison:**

In this section, we compare our protocol with existing similar protocols based on protocols cryptographic requirements and performance. Table VI shows the results of the comparison (√ = protocol satisfies the titled description, ∆ = partially satisfied, and X = does not satisfy the titled description).

**TABLE VI**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Security</th>
<th>CFT</th>
<th>IFO</th>
<th>P</th>
<th>NR</th>
<th>SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ouafi, Ka and V. Serge, [2009] [8]</td>
<td>Key based hash</td>
<td>∆</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Burbridge, T. and Soppora, A [2010] [9]</td>
<td>Public key encryption to generate signature</td>
<td>∆</td>
<td>∆</td>
<td>X</td>
<td>√(80 bytes)</td>
<td></td>
</tr>
<tr>
<td>Blass, E. et al. [2011][10]</td>
<td>X</td>
<td>∆</td>
<td>∆</td>
<td>X</td>
<td>√(120 Bytes storage)</td>
<td></td>
</tr>
<tr>
<td>Elkhiyaoui, K., et al. [2012] [11]</td>
<td>Elliptic curve public key encryption</td>
<td>∆</td>
<td>√</td>
<td>X</td>
<td>(a bits storage)</td>
<td></td>
</tr>
<tr>
<td>Our Proposal</td>
<td>Hash function, PUF() and XOR</td>
<td>√</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Fig. 7.** Comparative study of performance evaluation

protocols, the protocol in [11] satisfied most of the security requirements. However, it requires high tag storage and heavy cryptographic computation. Regarding performance of a cryptographic protocol, our proposed protocol perform better than protocol [9] and [11]. Protocol [8] has better performance than our proposed protocol but it violates many required security properties for a tracker protocol. Fig.7 shows performance comparison between our proposed protocol and some similar existing protocols.

In addition, Table VI shows that non-repudiation was not addressed by any existing protocols. This is an important requirement for a robust system like IoT, because without it, any dispute regarding object tracking may be impossible to address.

**VI. CONCLUSION**

The IoT requires the traceability and visibility of the object throughout the chain. While doing so, the protocol has to ensure security such as privacy, non-injection of fake tags and non-
repudiation. However, existing proposals have many vulnerabilities and they cannot be implemented in a passive RFID tag system. This paper proposed a secure tracker protocol for IoT to increase an object’s tracking and tracing to enhance the visibility of objects in IoT. It ensures non-repudiation along with privacy of the NRS which are both required for an IoT system. It uses the PUF function to prevent fake tag injection into the chain. The protocol is computationally feasible to implement in low cost RFID tags and is application independent. Further work needs to be done to develop a generalized protocol that can collect the relevant context information of an object to ensure context awareness. This will increase control over an object to accelerate development of the IoT.

REFERENCES


[26] C Cremers, The Scyther Tool, 2013, Dept. of Computer Science, university of OXFORD, [online] Available online: online


BIPLOB R. RAY is an ICT lecturer at the School of Engineering and Technology, CUUniversity, Australia. He is a member of IEEE and has twelve years of experience within the Education and ICT Industries in Philippines and Australia. Dr. Ray has published a number of refereed journals, conferences and book chapters in security and privacy of Internet of Things (IoT), secure communication protocols and health informatics. He has been a reviewer for a number of IEEE and Elsevier journals since 2010.

MORSHEDE U. CHOWDHURY is an academic staff member in the School of Information Technology, Deakin University, Australia. Dr. Chowdhury has more than 12 years of industry experience in Bangladesh and Australia. He has published more than 125 research papers including a number of journal papers, conference papers and book chapters.

JEMAL H. ABAWAJY is a Professor at the School of Information Technology, Deakin University, Australia. He is a senior member of IEEE and leads the Parallel and Distributed Computing Lab at Deakin University. Professor Abawajy is actively involved in funded research and has published more than 200 articles in refereed journals, conferences and book chapters.