Secure Architecture for RFID Enabled Supply Chain Hierarchy

Nauman Abdullah Shah
Dept. of Information Security,
Military College of Signals, NUST
Pakistan
naumanabd@yahoo.com

M. Arif Wahla
Dept. of Information Security,
Military College of Signals, NUST
Pakistan
arif@mcs.edu.com

Abstract- Radio Frequency Identification (RFID) tags possess the ability to uniquely identify every individual item at low cost. This emerging RF technology is well suited for supply chain management and is expected to replace barcodes in the near future. However, unlike barcodes, these tags have an extended range in which they are allowed to be queried by a reader. This may result in different attack scenarios such as unauthorized scanning by malicious readers. Therefore, a security protocol for RFID enabled supply chain is necessary to ensure the privacy and authentication between each tag, the corresponding reader and the backend server. In order to counter the evolving security issues, we propose a secure RFID based architecture for supply chain hierarchy. This architecture comprises of four different phases depicting the movement of a tagged item from source to destination. There are four different protocols supporting this architecture. Although this architecture was designed for use in supply chains, it can also be used for other RFID enabled applications by incurring minute changes.

I. INTRODUCTION

In today’s business environment barcodes technology is being used to manage the logistics. Although this technology has helped to automate the inventory of goods and control operations, barcode labels cannot help to enhance the production and distribution phases of the supply chain life cycle when a large number of items are palletized for shipment [1]. Modern supply chain management is focused on comprehensive information which barcode technology can not accomplish because of its limitations. RFID tags have overcome various deficiencies posed by the barcode labels resulting in a growing interest in RFID enabled supply chains[8][9]. RFID tags are able to uniquely identify individual items of a product type. This helps to segregate tagged items in different hierarchies and is useful when the transaction history of each item needs to be maintained or when an individual item needs to be tracked [2]. Furthermore, RFID tags do not require line-of-sight increasing the scanning process of tagged items significantly.

RFID based system consists of tags, readers and a backend server. Each tag has ability to store Information and perform different computations. Tags communicate with the readers using a coupling element such as an antenna. The reader interrogates the tags within its range and interacts with the backend server for more functionality. However, there is a growing concern about the security of the RFID based communication. A typical tag replies with its ID when interrogated by a reader which never changes. An attacker can easily acquire this ID and can keep on tracking the movement of tag raising privacy issues for the customers buying tagged item. The data contained within the tags if not protected may lead to confidentiality issues; an attacker may access and duplicate it to bogus tags.

Even when the content of the tags is protected, individuals may be tracked through predictable tag responses. Unprotected communication between tag, reader and the backend server may lead to eavesdropping, tracking, and spoofing and/or cloning attack [12][11]. Although such vulnerabilities can easily be removed using variety of cryptographic primitives there is always a need for an energy efficient protocol that requires less energy and storage capacities since RFID tags have limited computational and storage resources [18]. For example, a practical tag costing in the order of $0.05 US may be limited to have only few hundred bits of storage and roughly 500–5,000 gates. Passive RFID tags are industry favorites because of their small size and low cost. They are used more often in supply chains to tag different items [4]. Each RFID tag carries a unique code managed by EPCglobal [3] and has fields to store the manufacturer code and the product code that makes it easy to follow the inventory patterns.

In this paper we propose an architecture consisting of four different Secure Supply Chain Protocols (SSCP I through IV) covering the movement of a tagged item from secure vicinity A to B. Our architecture addresses privacy and authentication concerns by using a privacy bit that helps the tag to distinguish between secure and insecure environments. It allows high-
value goods to be protected against adversarial attackers and is sufficiently robust to withstand attacks such as replay attack, man-in-the-middle attack, eavesdropping and traceability [5][10]. Rest of the paper proceeds as follows. In section II, we elaborate the related work. In section III, we define our architecture. Section IV, presents the performance and security analysis of the architecture. We give concluding remarks in section V.

II. RELATED WORK

As the use of RFID tags has started to grow in different enterprise applications, lot of protocols have been proposed to secure their communication with the reader and the backend server [6][7][17]. The main focus of these protocols was to counter the prevailing authentication and privacy issues. We describe some of the related studies below.

1. Serverless Search and Authentication Protocols for RFID proposed by Tan et al [13]

The structure of authentication list is given below. \( k \) states that RFID tag is in the \( k \)th authentication process for tag \( T_j \).

\[
L \rightarrow h(\text{id}_j \mid S_j) : \text{id}_j : k
\]

In this protocol, reader \( R_i \) sends access request to tag \( T_j \). \( T_j \) checks request and generates a random number \( r_j \) and then \( T_j \) sends it to \( R_i \). \( R_i \) generates a \( r \) and sends \( r \) and its identifier \( \text{id}_i \) to \( T_j \). \( T_j \) computes \( h(h(\text{id}_i, S_j))_m \) and chooses first \( m \) bits to get \( t_1 \) and then sends it to \( R_i \). \( R_i \) searches his authentication list \( L \) using \( h(h(\text{id}_i, S_j))_m \) and then computes \( f_1 \) and sends to \( T_j \). \( T_j \) sets \( f_2 = h(h(\text{id}_i, S_j) \mid t_1 \mid r_1) \) and compares received \( f_2 \). If they are equal, \( T_j \) forwards its Pseudo-identity \( h(\text{id}_j) \) as \( f_3 \).

Limitations

Tan et al claimed that this protocol provides mutual authentication. They didn’t provide reader authentication in tag. They provide authentication using comparison of \( f_1 \), \( f_2 \) and \( f_3 \). Tag believes that reader is authorized to access tag through \( f_1 \). Reader believes that the tag is authorized by CA. however rounds and complexity have been increased. Moreover they didn’t address privacy because the same response is sent to the reader repeatedly which may result in traceability attack.

2. A Privacy and Authentication Protocol (PAP) for passive RFID tags proposed by Alex X.Liu [14]

In the PAP Protocol, each tag attached to a product stores secret key \( k \) shared by every reader and tag, a generic name of tag and an ID of tag. The protocol starts with the query request sent from reader to tag. The tag then sends its generic name and random nonce to the reader. The reader uses this information to determine secret key \( k \) of tag and applies one way hash function upon it, sending both hashed result and another random nonce to the tag. Tag verifies reader by performing hash function using its secret key \( k \) with nonce sent to reader. If this value matches the hashed result sent from reader, tag authenticates reader. The tag then performs another hash using its secret key \( k \) with the nonce received from reader and sends this hashed value to the reader. The reader then performs hash function with its secret key \( k \). If result matches, the reader authenticates the tag. Afterwards tag sends its ID to the reader.

Limitations

PAP is susceptible to DOS (Denial of service) attack. An adversary can continuously engage the tag by sending query commands to tag. Adversary can keep the tag busy in generating nonce values to reply adversarial queries. It may result in missing the valid requests from valid readers. Another vulnerability is the generic name given to each tag item. The same name is sent to the reader upon each query which may result in traceability attack. Moreover, the tag stores separate key for different readers (1) At Checkout (2) On the way and (3) In the store. The storage of separate key for communicating with multiple readers results in space overhead.

3. RFID Authentication Protocol proposed by Tsudik [15]

In this protocol, the tag, reader, and the back-end server share a common secret that is different for each tag. The protocol begins when the reader sends the current timestamp \( (TS) \) to the tag. The tag checks to see if this new timestamp is newer than the previous timestamp it had processed from the reader \( (TS) \) and also if the new timestamp is greater than TSmax. If these are untrue, the tag just uses a pseudo random number generator to generate an l-bit random number \( (H) \). Otherwise, it records the new timestamp and computes the hash value \( H \) of the new timestamp with the secret key \( x \). \( H \) is then sent to the reader which sends it to the server for validation.

Limitations

There are two drawbacks of this protocol: it is susceptible to denial of service (DoS) attack when an adversary sends an inaccurate timestamp and incapacitates the tag either temporarily or permanently; since the tag assumes that it is not authenticated more than once within a short duration of time. Depending on the time granularity, the tag could misidentify a valid request from a reader as invalid. Risk of replay attack also arises due to sole reliance on timestamps for authentication. An adversary can send a series of some future timestamps to the tag and can easily record their responses. When the time in the collected timestamps eventually becomes true, attacker can respond to requests from the reader appropriately without the tag being present.

4. RFID Mutual Authentication Scheme proposed by Lee, Asano, and Kim [13]

The protocol of Lee, Asano, and Kim uses both XOR and hash chains to authenticate tags and readers.

Limitations

If an adversary intercepts the fifth message \( (rC') \) between the reader and tag, it can prevent the tag from updating its secret. However, the back-end server has already updated its secret. This could cause database desynchronization. Although the messages passed seem random, an active adversary has control over some of the messages being passed. For example, the adversary can block the first message \( s \) and send a l-bit null
vector as x to the tag. The tag now computes rB as h(rA@x@0) = h(rA@x). Next, the adversary can modify rA to (rA@s) and send it to the reader. When the back-end server computes rB = h(rA@x@s) = h(x@s) from (s, (rA@s), rB) it receives from the reader, it would not be able to detect the adversary modified messages. The adversary can also send a random rC’ to the tag as the fifth message. The tag will not update its key since rC’ ≠ rC. The adversary can then impersonate the tag by always sending (rA@s, h(rA@x)) to the reader.

III. PROPOSED ARCHITECTURE

A. Notations

We use the notations summarized in Table 1 to describe our architecture through rest of this paper.

<table>
<thead>
<tr>
<th>TABLE 1: Notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>RID = Reader ID unique for each reader</td>
</tr>
<tr>
<td>Pbit = Privacy bit</td>
</tr>
<tr>
<td>ID = TagID</td>
</tr>
<tr>
<td>Data = Details of the tagged item</td>
</tr>
<tr>
<td>n = Nonce generated by the reader.</td>
</tr>
<tr>
<td>Pseudonym = Pseudo index</td>
</tr>
<tr>
<td>PubR = Public key of Reader</td>
</tr>
<tr>
<td>PubS = Public key of server</td>
</tr>
<tr>
<td>hKey = current hashing key</td>
</tr>
<tr>
<td>sKey = current symmetric key</td>
</tr>
<tr>
<td>Pseudo = Code for the product</td>
</tr>
<tr>
<td>Index = number of times a tag has been red.</td>
</tr>
<tr>
<td>nRID = Next reader ID on the way</td>
</tr>
<tr>
<td>nsKey = New symmetric key</td>
</tr>
<tr>
<td>nhKey = New hash key</td>
</tr>
<tr>
<td>n = Nonce generated by the tag.</td>
</tr>
<tr>
<td>nRID = Next reader ID</td>
</tr>
</tbody>
</table>

B. Assumptions

- It is assumed that the store is secure vicinity.
- All the readers are connected to the backend server.
- The private keys of backend server and reader are secure.
- Backend server is physically secure.
- It is assumed that once the item is marked with a RFID tag, the tag contains default keys.
- The proposed architecture addresses the security issues at network layer of RFID based communication.
- The reader and the backend server have secure access to the database that contains information about products, tagged items, readers and keys.

C. Initial setup

In the proposed architecture, each tag works in two modes. In first mode when the privacy bit is set to ‘0’, fields of tag (keys index and RID) are set to default values. In second mode when the privacy bit is set to ‘1’ fields of tag (keys, index and RID) are updated from the backend server. Supply chain consists of items from different class of products. We have segregated these products and have given pseudo values to each class of products. e.g. “Figure 2” shows the setup of supply chain containing products TV and Laptops from a company Ecomp. There are two different models of TV and laptops. The pseudo values are assigned to each product model. These pseudo values are used to retrieve key and index values form the backend database. Reader is only permitted to acquire keys form the database while backend server is allowed to view and update the database fields.

D. Designed protocols

Our architecture is divided into four protocols working in four different phases of supply chain movement. At first the tagged item is within the store (a secure vicinity), it then leaves the store (at check out), moves along the insecure environment and reaches its destination where it is again entered in to a secure vicinity. It is important to note that when the tag receives the new keys and reader id, it replaces the previous values i.e. sKey = nsKey, hKey = nhKey, RID = nRID.

a) Within the store (SSCP-I)

When the tagged item is in the store, it does not need to authenticate the querying reader because of the trusted environment. The tag is in default mode while residing in the store.

SSCP-I initiates when reader queries the tag by sending RID as shown in “Figure 3”. Tag responds by sending its ID and pseudonym to the reader. With the help of the received information from tag, the reader acquires the default key from the database. Reader generates the nonce and calculates the hash value using default hashing key (hKey). Reader then sends message (RID, ID, pseudonym, n, H(n)) to backend server encrypted with the public key of the backend server. After decrypting the message using its private key the backend server queries the backend database to retrieves the designated hKey. The backend server calculates H(n) and verifies the received hash value for authenticating the reader. Once the reader is authenticated the backend server sends back ID, data pertaining to that ID and the nonce value received by the
reader in message 3. The reader then verifies the value of $n_r$ received. This authenticates the backend server.

b) At checkout (SSCP-II)

SSCP-II initiates when the tagged items are about to move out of the secure vicinity and go through checkout process. Initially tag, reader and the backend server exchange same messages 1, 2, 3 and go through the same process as in SSCP-I. In the fourth message the backend server sends back index, id of the next reader in line ($n_{RID}$) along with the keys to be used with the next reader. These values are sent to the tag once the message is authenticated. The index value received from the backend server is verified by the tag. Once the index value is accepted tag changes its privacy bit from ‘0’ to ‘1’ as shown in “Figure 4”. The index value is also incremented.

c) On the way (SSCP-III)

When the tag moves into an insecure environment where the readers are not trusted Pbit is set to ‘1’. In SSCP-III when reader queries a tag by sending RID, tag first verifies it. Tag and reader then go through the challenge response session as shown in message 2, 3, and 4 in “Figure 5”. Reader decrypts the tag id using the sKey acquired from the database. Reader and the backend server go through the authentication process as in SSCP-II message 3 and 4. The index value is also incremented.

When the tag receives the valid index value it changes the privacy bit from ‘1’ to ‘0’. The index value is also incremented.

IV. PERFORMANCE AND SECURITY ANALYSIS OF THE ARCHITECTURE

In this section we elaborate the strength of the proposed architecture in terms of security and performance.

A. Performance Analysis

Based on the performance evaluation of the SSCP architecture, the following analysis is done:

1) A very light encryption algorithm can be used to secure transmission of data. We used TEA for the symmetric key encryption during testing of this architecture

2) The functions required on the tag are the nonce generators, hashing, comparison and XOR functions. We used built-in function of java for nonce generation and SHA-1 for hashing.

3) The asymmetric encryption scheme is used to secure the channel between reader and server. We used RSA with 1024 bit key to simulate the architecture.

4) The SSCP architecture requires no exhaustive search from the database because of the use of pseudo values for each product.

5) The SSCP architecture can easily be implemented as no heavy computations are required on the side of tag.
TABLE 2 Protocol comparisons

<table>
<thead>
<tr>
<th>Security Measures</th>
<th>SAP</th>
<th>PAP</th>
<th>Tsudik</th>
<th>Lee</th>
<th>SSCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementable on low cost tags</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Authentication at all levels</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Privacy preserved</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Traceability protection</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Anonymity of messages</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Possibility of Tag’s Cloning</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Possibility of Man in middle Attack</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Replay Attack Resistance</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Possibility of server overload</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Requirement of exhaustive Search</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>

( ☑ = YES , ☐ = NO )

B. Security Analysis

The proposed architecture is a trustable solution due to the following reasons:

1) Symmetric key encryption is used for the transmission of messages between reader and tag in an insecure environment.
2) Tags and readers are mutually authenticated in an insecure environment.
3) The communication between reader and backend server is always secured by encryption and mutual authentication. Random messages are transmitted on every query.
4) Privacy of tag is achieved by reserving a privacy bit in memory of tag. A tag is only open when the privacy bit is 0. A tag keeps privacy bit 1 on the way (Insecure Area) and requires authentication from all the querying readers.
5) Hashed and encrypted messages are transferred instead of plain text during communication. By eavesdropping, attacker could not get plain text ID so proposed architecture provides anonymity.

C. Attacks Prevented

The following attacks are being prevented by our solution.

1) Unauthorized Tag Cloning: These are integrity attacks in which attacker succeeds in cloning an identical tag after capturing the related information. In proposed scheme, tag emits secret information only after authenticating reader. As the secret information of tag travels in encrypted form, the response of tag does not reveal any identifying information.

2) Traceability: This attack violates the privacy of the tagged item. Traceability helps attacker trace tag through valid readers. To address the concern of traceability, messages being transmitted are random at each query resulting in a random response preventing possible attack of tracking.

3) Replay Attack: These are attacks in which attacker records a legitimate response and reuses it to gain unauthorized access later in time. In our proposed architecture all the messages transmitted are updated in a random sequence and are encrypted in the unsecure environment. Even if an attacker is successful in intercepting the messages he won’t be able to reuse them.

4) Man in middle Attack: This attack is launched when the data moves from one entity to another. An attacker can intercept the messages and manipulate them back and forth between RFID components [12]. This attack is not possible due to mutual authentication procedure within the architecture. Attacker cannot impersonate legitimate reader and cannot receive information from tag. In the same way, attacker cannot impersonate a tag because reader performs a strong authentication procedure.

5) Possibility of Eavesdropping: Due to mutual authentication procedure, even when an attacker eavesdrop the final response from the tag, it cannot pretend to be an authorized reader in the next authentication session. Also the keys are shared only between tag and reader so attacker cannot make any sense from encrypted messages sent to reader.

V. CONCLUSION

This paper has introduced an idea of a secure communication in a RFID enabled supply chain going through four different scenarios. The proposed solution ensures the privacy and authentication of the messages transmitted between the tag reader and the backend server as shown in Table 2. The methodology of pseudo numbers help to enhance database search and adds another level of security since the actual name of the product is never transmitted. The architecture adds layers of security and increases decision making power of the tag with minimum space requirements, since the four fields of the tag (RID, hKey, sKey, and index) are overwritten whenever the tag moves out of the default mode. The reader does not need to store any information since it retrieves all the information from the backend database hence reducing the space complexity. The backend server tracks the movement of the tagged item and updates the RID and index values.

The generic fields in the proposed architecture are particularly useful. An organization can use the encryption and hashing schemes according to its own functionality and capacity.

REFERENCES


[9]. Gartner, RFID Market $3 Billion in 2010. December 13th, 2005


[13]. Chiu C. Tan, Bo Sheng, Qun Li, "Severless Search and Authentication Protocols for RFID," percom, pp.3-12, Fifth IEEE International Conference on Pervasive Computing and Communications (PerCom’07), 2007


172