Security of an RFID Based Authentication Protocol with Bitwise Operations for Supply Chain

Muhammad Arslan Akram
Information Technology University of the Punjab: Information Technology University
Adnan Noor Mian (adnan.noor@itu.edu.pk)
Information Technology University of the Punjab: Information Technology University
https://orcid.org/0000-0003-1034-0140

Short Report

Keywords: Authentication protocol, PUF, Cryptanalysis, RFID, Imperson- ation Attack, Supply Chain

Posted Date: April 26th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-413438/v1

Wireless Personal Communications
DOI: 10.1007/•••••-•••-•••-••••-•

Security of an RFID based Authentication Protocol with Bitwise Operations for Supply Chain

Muhammad Arslan Akram[1] · Adnan Noor Mian[1]

Abstract Due to the stringent computational capabilities of low-cost RFID tags, several lightweight secure authentication protocols have been proposed for an RFID-based supply chain using bitwise operations. In this paper, we study the vulnerabilities associated with bitwise operations by doing cryptanalysis of a secure lightweight authentication protocol for RFID tags. The bitwise operations like rotation and XOR show that the protocol is vulnerable to tag, reader, and supply chain node impersonation attacks. We find that the major cause of the vulnerability is bitwise operations and suggest using the physically unclonable functions rather than bitwise operations to secure such lightweight protocols.

Keywords: Authentication protocol, PUF, Cryptanalysis, RFID, Impersonation Attack, Supply Chain

1. Introduction

Supply chain is the management of the entire flow of goods, data, finance and production, and supervises the processes until it transforms them into final products or reaches their destination. A well-managed and immutable supply chain is needed to identify the origin of counterfeit goods which have somehow reached to the consumers Dabbene, Gay, and Tortia (2014). In supply chain many departments link with each other by using RFID tags for the acquisition of their own data.

Recently, several lightweight authentication protocols have been proposed with the goal of achieving secure authentication through bitwise operations

✉ A.N.M.
adnan.noor@itu.edu.pk

M.A.A.
arslan.akram@itu.edu.pk

[1] Department of Computer Science, Information Technology University, Lahore, Punjab, Pakistan.

because of limited computational capabilities of low-cost RFID tags. For low cost computation and energy constraints devices, often bitwise operations are suggested without crypto-primitives, which lead to different vulnerabilities Xin, Zhang, and Yang (2020). These bitwise operations have not been shown to help and create secure protocol Safkhani and Shariat (2018); Sidorov *et al.* (2019); Mujahid, Najam-ul Islam, and Sarwar (2017); Safkhani *et al.* (2021); Sun and Mu (2017); Izza, Benssalah, and Drouiche (2021). In this paper, we show that single or multiple uses of rotation (ROT) functions and XOR operations without crypto-primitives does not secure protocol against various attacks, by doing cryptanalysis of a recently published Jangirala et al.'s protocol, "Designing secure lightweight blockchain-enabled RFID-based authentication protocol for supply chains in 5G mobile edge computing environment (LBRAPS)" Jangirala, Das, and Vasilakos (2019).

The LBRAPS protocol is based on one way hash function, bitwise rotation (ROT) function and bitwise exclusive OR (XOR). It comprises of two phases, 1) initialization, 2) authentication and key agreement. The authors have proved that LBRAPS is immune to various active attacks by formal security analysis based on Automated Validation of Internet Security Protocols and Applications (AVISPA) tool and claimed that the protocol is secure against many known threats such as mutual authentication, tag impersonation attack and reader impersonation attack. However, the key issue related to their design is that an attacker can easily acquire the credentials by the means of capturing the transmitted messages as it based on only bitwise rotate function. We demonstrate that the protocol is not immune to reader impersonation, tag impersonation and supply chain node impersonation attacks. We have also proposed countermeasures to secure the protocol.

2. Review of Jangirala et al.'s Protocol

There are three participants in the protocol, the tag $\mathcal{T}$, the reader $\mathcal{R}$ and the supply chain node $\mathcal{S}$. The common notations used are shown in Table 1. The protocol has two phases: initialization phase, and login and authentication phase.

2.1. Initialization Phase

To setup the protocol, the identity $ID_{\mathcal{T}}$ of tag $\mathcal{T}$ or reader $\mathcal{R}$ is considered as password, and the blockchain produces public key address for each account identifier. Therefore, the tag stores the record $\{ID_{\mathcal{T}}, Bal_{BC}\}$ in its database, where balance amount and tag identity in blockchain under department $Dept_i$ are Bal_{BC} and $ID_{\mathcal{T}}$, respectively. Similarly, every reader $\mathcal{R}$ also saves $ID_{\mathcal{R}}$ in its repository. Consequently, the $\mathcal{S}$ and $\mathcal{R}$ exchanges a secret-key $X_{RS} = h(ID_{\mathcal{S}} \| B_{\mathcal{S}} \| ID_{\mathcal{R}})$, where $B_{\mathcal{S}}$ denotes the blockchain connected with the $\mathcal{S}$. $\mathcal{R}$ initiates the transaction message and forwards it towards $\mathcal{T}$. Additionally, $\mathcal{R}$ must have an initial balance in its account in order to make transactions. Consequently, Bal_{BC} denotes the balance of $\mathcal{T}$ in the blockchain and for every new transaction it is presented as $Bal_{New} = Bal_{BC} + S_{Amount}$, where S_{Amount} denotes the amount of $\mathcal{S}$ transactions.

Table 1. Common used notations

Notations	Elucidations
$\mathcal{R}$, $\mathcal{T}$, $\mathcal{S}$	Reader, tag and supply-chain node
$ID_{\mathcal{R}}$, $ID_{\mathcal{T}}$, $ID_{\mathcal{S}}$	Identity of the reader, tag, and supply chain node
X_{RS}	Secret key between reader and supply chain
$Dept_i$	ith department
Bal_{BC}	$Dept_i$ balance amount in blockchain
SK	Session key
$B_{\mathcal{S}}$	Blockchain associated with $\mathcal{S}$
$ROT(X,Y)$	Left rotate of X by hamming weight of Y
$RROT(X,Y)$	$X \ggg Y$, right rotate of X by hamming weight of Y

2.2. Login and Authentication Phase

The entities $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{R}$ follow the subsequent steps for establishment of session key between $\mathcal{T}$ and $\mathcal{S}$.

Step 1: The reader $\mathcal{R}$ engenders a nonce R_N and current time-stamp $T_{\mathcal{R}}$. Furthermore, it calculates: $M_{\mathcal{R}} = ROT(R_N \oplus ID_{\mathcal{T}} \oplus T_{\mathcal{R}}, T_R \oplus ID_{\mathcal{T}})$ and $C_{\mathcal{R}} = h(M_{\mathcal{R}}||ID_{\mathcal{T}}||R_N)$, and then sends the request message $MSG_1 = \{M_{\mathcal{R}}, C_{\mathcal{R}}, T_{\mathcal{R}}\}$ to $\mathcal{T}$ through an insecure channel.

Step 2: After receiving the message MSG_1 from $\mathcal{R}$, $\mathcal{T}$ first validates the time-stamp $T_{\mathcal{R}}$. If it does not hold, the session is aborted by $\mathcal{T}$. Otherwise, $\mathcal{T}$ fetches the nonce R_N of $\mathcal{R}$ as $R_N = (M_{\mathcal{R}} \ggg (ID_{\mathcal{T}} \oplus T_{\mathcal{R}})) \oplus ID_{\mathcal{T}} \oplus T_{\mathcal{R}}$, and computes: $C'_{\mathcal{R}} = h(M_{\mathcal{R}}||ID_{\mathcal{T}}||R_N)$ and validates: $C'_{\mathcal{R}} \stackrel{?}{=} C_{\mathcal{R}}$. If it holds, $\mathcal{T}$ also calculates: $C_{\mathcal{T}} = h(R_N \oplus ID_{\mathcal{T}} \oplus Bal_{New})$, $M_{\mathcal{T}} = ROT(R_N \oplus ID_{\mathcal{S}} \oplus T_{\mathcal{T}}, T_{\mathcal{T}} \oplus ID_{\mathcal{T}})$ and $Auth_{\mathcal{R}} = h(C_{\mathcal{T}}||R_N||M_{\mathcal{T}}||ID_{\mathcal{T}}||T_{\mathcal{T}})$. After above computations, $\mathcal{T}$ transmits the message $MSG_2 = \{C_{\mathcal{T}}, Auth_{\mathcal{R}}, M_{\mathcal{T}}, T_{\mathcal{T}}\}$ towards $\mathcal{R}$ via an insecure channel.

Step 3: After receiving the message MSG_2 from $\mathcal{T}$, $\mathcal{R}$ validates the time-stamp $T_{\mathcal{R}}$ to check the authenticity of the received message. If it holds, $\mathcal{R}$ verifies if $Auth_{\mathcal{R}} \stackrel{?}{=} h(C_{\mathcal{T}}||R_N||M_{\mathcal{T}}||ID_{\mathcal{T}}||T_{\mathcal{T}})$. If it holds, $\mathcal{R}$ then engenders two nonces R_a, R_b and selects $T'_{\mathcal{R}}$, and calculates: $M_P = R_a \oplus ID_{\mathcal{S}} \oplus R_b$, $M_Q = X_{RS} \oplus R_b$, and $Reader_{check} = h(R_a \oplus ID_{\mathcal{S}} \oplus Bal_{New} \oplus (R_b||T'_{\mathcal{R}}))$. Afterwards, $\mathcal{R}$ transmits the message $MSG_3 = \{M_Q, M_P, Reader_{check}, T'_{\mathcal{R}}\}$ to $\mathcal{S}$ belongs to $Dept_1$ of the blockchain.

Step 4: Upon receiving the message MSG_3 from the reader $\mathcal{R}$ and $\mathcal{S}$ verifies the authenticity of time-stamp $T'_{\mathcal{R}}$. If it holds, $\mathcal{S}$ starts the predefined smart-contract on the blockchain to proceed the authentication mechanism. The authentication mechanism is enabled via the $\mathcal{S}$ of blockchain by validating if the $ID_{\mathcal{T}}$ presents in $\mathcal{S}$ repository. If it is not available, the session is aborted, else $\mathcal{S}$ gets Bal_{BC-REC} and performs the following

calculations: $R_b = X_{RS} \oplus M_Q$, $R_a = M_P \oplus ID_{\mathcal{S}} \oplus R_b$, $S_{checkA} = h(R_a \oplus ID_{\mathcal{S}} \oplus Bal_{BC-REC} \oplus (R_b||T'_{\mathcal{R}}))$ and $S_{checkB} = h(R_a \oplus ID_{\mathcal{S}} \oplus (Bal_{BC-REC} + S_{Amount}) \oplus (R_b||T'_{\mathcal{R}}))$. Afterwards, the validation is completed by checking the condition $(S_{checkA} = Reader_{check})$ and if it holds, $\mathcal{S}$ records $Bal_{BC-REC} = Bal_{BC}$, otherwise if $(S_{checkB} = Reader_{check})$ is true, $\mathcal{S}$ acknowledges $ID_{\mathcal{R}} \xrightarrow{S_{Amount}} ID_{\mathcal{T}}$, and records $Bal_{BC-REC} = Bal_{BC-REC} + S_{Amount}$ in distributed ledger $Ledger_{BC}$. Consequently $\mathcal{S}$ engenders a random nonce $S_{\mathcal{R}}$ at current time-stamp $T_{\mathcal{S}}$ to calculate $S_P = ROT(T_S, ID_{\mathcal{S}} \oplus X_{RS}) \oplus ROT(S_{\mathcal{R}}, X_{RS})$, $S_Q = ROT(S_{\mathcal{R}}, ID_{\mathcal{S}}) \oplus ROT(T_{\mathcal{S}}, X_{RS})$, $SK_{ST} = h(ID_{\mathcal{T}} \oplus Bal_{BC-REC} \oplus S_{\mathcal{R}} \oplus ID_{\mathcal{S}})$ and $S_{\mathcal{S}} = h(SK_{ST}||S_{\mathcal{R}}||Bal_{BC-REC})$. Here, SK_{ST} is the session key enabled by $\mathcal{S}$ so that by validating the correct message, $\mathcal{T}$ can maintain the similar session key. The $\mathcal{S}$ then transmits the message $MSG_4 = \{S_P, S_Q, S_{\mathcal{S}}, T_{\mathcal{S}}\}$ to $\mathcal{R}$ through an insecure channel.

Step 5: After receiving the message MSG_4 from $\mathcal{S}$, $\mathcal{R}$ verifies the validity of received time-stamp $T_{\mathcal{S}}$. If it holds, $\mathcal{R}$ extracts the random nonce $S_{\mathcal{R}}$ of $\mathcal{R}$ as $S'_{\mathcal{R}} = RROT(S_Q \oplus ROT(T_{\mathcal{S}}, X_{RS}), ID_{\mathcal{S}})$ and checks it to authenticate $\mathcal{S}$ by validating $S_P \stackrel{?}{=} ROT(T_{\mathcal{S}}, ID_{\mathcal{S}} \oplus X_{RS}) \oplus ROT(S'_{\mathcal{R}}, X_{RS})$. If it validates, $\mathcal{R}$ further calculates $R_Q = ROT(S_{\mathcal{R}}, ID_{\mathcal{R}}) \oplus ROT(T_{\mathcal{S}}, R_N)$ and sends the message $MSG_5 = \{S_{\mathcal{S}}, R_Q, T_{\mathcal{S}}\}$ towards $\mathcal{T}$ through an insecure channel.

Step 6: After receiving the request message MSG_5 from $\mathcal{R}$, $\mathcal{T}$ fetches nonce $S_{\mathcal{R}}$ as $S'_{\mathcal{R}} = RROT(R_Q \oplus ROT(T_{\mathcal{S}}, R_N), ID_{\mathcal{R}})$, calculates session key $SK_{ST} = h(ID_{\mathcal{T}} \oplus Bal_{New} \oplus S'_{\mathcal{R}} \oplus ID_{\mathcal{S}})$ to validate both $\mathcal{S}$ and $\mathcal{R}$ by checking the condition $S_{\mathcal{S}} = h(SK_{ST}||S'_{\mathcal{R}}||Bal_{BC-REC})$. If it does not match, $\mathcal{T}$ rejects the communication request, else if it holds, then $\mathcal{T}$ modifies $Bal_{New} = Bal_{BC} + S_{Amount}$ in its database record. After maintaining the session key $SK_{ST} = h(ID_{\mathcal{T}} \oplus Bal_{New} \oplus S'_{\mathcal{R}} \oplus ID_{\mathcal{S}})$ between $\mathcal{T}$ and $\mathcal{S}$ with the help of $\mathcal{R}$, the blockchain balance has modified in the distributed-ledger with updated balance Bal_{New}. The reason for maintaining the session key between $\mathcal{T}$ and $\mathcal{S}$ is that depending on the potential need, the blockchain will intercept with the concerned department where $\mathcal{T}$ and $\mathcal{S}$ want to communicate safely with the SK_{ST} session key.

3. Cryptanalysis of the protocol

We assume adversarial model same as Jangirala et al.'s in which an adversary $\mathcal{A}_{ad}$ can block, alter or even delete the message on the radio link between a reader and tag. It can also perform cloning and physical attack. Further, we suppose that there are various readers in the system and the adversary has control over the public channel.

3.1. Reader Impersonation

The tag stores $\{ID_T\}$ in its memory during initialization process. Suppose ID_T are some how leaked (stolen or retrieved through power analysis Yang *et al.* (2016)) to $\mathcal{A}_{ad}$, we shall show that the SK can be constructed and impersonation is possible, because the reader uses $\{ID_T\}$ in the generation of login message $MSG_1 = \{M_{\mathcal{R}}, C_{\mathcal{R}}, T_{\mathcal{R}}\}$. Therefore, an $\mathcal{A}_{ad}$ can easily steal these parameters and can utilize them to mount impersonation attack on a legitimate reader. For this purpose, an $\mathcal{A}_{ad}$ performs the following steps:

Step 1: First of all, the $\mathcal{A}_{ad}$ randomly selects $R_N^{\mathcal{A}_{ad}}$ at time $T_{\mathcal{R}}^{\mathcal{A}_{ad}}$ and computes: $M_{\mathcal{R}}^{\mathcal{A}_{ad}} = ROT(R_N^{\mathcal{A}_{ad}} \oplus ID_T \oplus T_{\mathcal{R}}^{\mathcal{A}_{ad}}, T_{\mathcal{R}}^{\mathcal{A}_{ad}} \oplus ID_T)$ and $C_{\mathcal{R}}^{\mathcal{A}_{ad}} = h(M_{\mathcal{R}}^{\mathcal{A}_{ad}} \| ID_T \| R_N^{\mathcal{A}_{ad}})$.

Step 2: After the above calculations, $\mathcal{A}_{ad}$ sends the request message $MSG_1 = \{M_{\mathcal{R}}^{\mathcal{A}_{ad}}, C_{\mathcal{R}}^{\mathcal{A}_{ad}}, T_{\mathcal{R}}^{\mathcal{A}_{ad}}\}$ to the tag.

Step 3: Upon receiving request message MSG_1, the tag first checks validity of time period $T_{\mathcal{R}}$. Then, it extracts R_N as $R_N = (M_{\mathcal{R}}^{\mathcal{A}_{ad}} \ggg (ID_T \oplus T_{\mathcal{R}}^{\mathcal{A}_{ad}})) \oplus T_{\mathcal{R}}^{\mathcal{A}_{ad}} \oplus ID_T$. Actually, this R_N is generated by adversary during login process. Hence, Both R_N and $R_N^{\mathcal{A}_{ad}}$ always will be equal. Afterwards, tag calculates $C_{\mathcal{R}}^{'} \stackrel{?}{=} h(M_{\mathcal{R}}^{\mathcal{A}_{ad}} \| ID_T \| R_N)$ and also validates $C_{\mathcal{R}}^{'} \stackrel{?}{=} C_R^{\mathcal{A}_{ad}}$, hence it holds true and tag deems the adversary as a legal reader.

Hence, $\mathcal{A}_{ad}$ has successfully impersonated as a legal reader and the protocol is vulnerable to reader impersonation attack. The detail is given in the Fig. 1.

3.2. Tag Impersonation

The tag stores $\{ID_T\}$ in its memory during initialization process. Suppose ID_T are some how leaked (stolen or retrieved through power analysis Yang *et al.* (2016)) to $\mathcal{A}_{ad}$. However, an adversary $\mathcal{A}_{ad}$ can easily extract these parameters and can easily impersonate a valid tag after acquiring the login message $MSG_1 = \{M_{\mathcal{R}}, C_{\mathcal{R}}, T_{\mathcal{R}}\}$. To impersonate a legitimate tag the adversary follows these steps:

Step 1: Whenever the reader sends message $MSG_1 = \{M_{\mathcal{R}}, C_{\mathcal{R}}, T_{\mathcal{R}}\}$ to the tag, the $\mathcal{A}_{ad}$ intercepts it and extracts $R_N^{\mathcal{A}_{ad}}$ as $R_N^{\mathcal{A}_{ad}} = (M_{\mathcal{R}} \oplus (ID_T \oplus T_{\mathcal{R}}^{\mathcal{A}_{ad}})) \oplus T_{\mathcal{R}}^{\mathcal{A}_{ad}} \oplus ID_T$ and calculates: $C_{\mathcal{R}}^{\mathcal{A}_{ad}} = h(M_{\mathcal{R}} \| ID_T \| R_N^{\mathcal{A}_{ad}})$. Next, he checks $C_{\mathcal{R}}^{\mathcal{A}_{ad}} \stackrel{?}{=} C_{\mathcal{R}}$, if it does not hold true, then the session will be terminated, otherwise the $\mathcal{A}_{ad}$ generates $T_T^{\mathcal{A}_{ad}}$ and calculates: $C_T^{\mathcal{A}_{ad}} = h(R_N^{\mathcal{A}_{ad}} \oplus ID_T \oplus Bal_{New}^{\mathcal{A}_{ad}})$, $M_T^{\mathcal{A}_{ad}} = ROT(R_N^{\mathcal{A}_{ad}} \oplus ID_S \oplus T_T^{\mathcal{A}_{ad}}, T_T^{\mathcal{A}_{ad}} \oplus ID_T)$, and $Auth_{\mathcal{R}}^{\mathcal{A}_{ad}} = h(C_T^{\mathcal{A}_{ad}} \| R_N^{\mathcal{A}_{ad}} \| M_T^{\mathcal{A}_{ad}} \| ID_T \| T_T^{\mathcal{A}_{ad}})$. Afterwards, the $\mathcal{A}_{ad}$ sends $MSG_2 = \{C_T^{\mathcal{A}_{ad}}, Auth_{\mathcal{R}}^{\mathcal{A}_{ad}}, M_T, T_T\}$ to the reader through insecure channel.

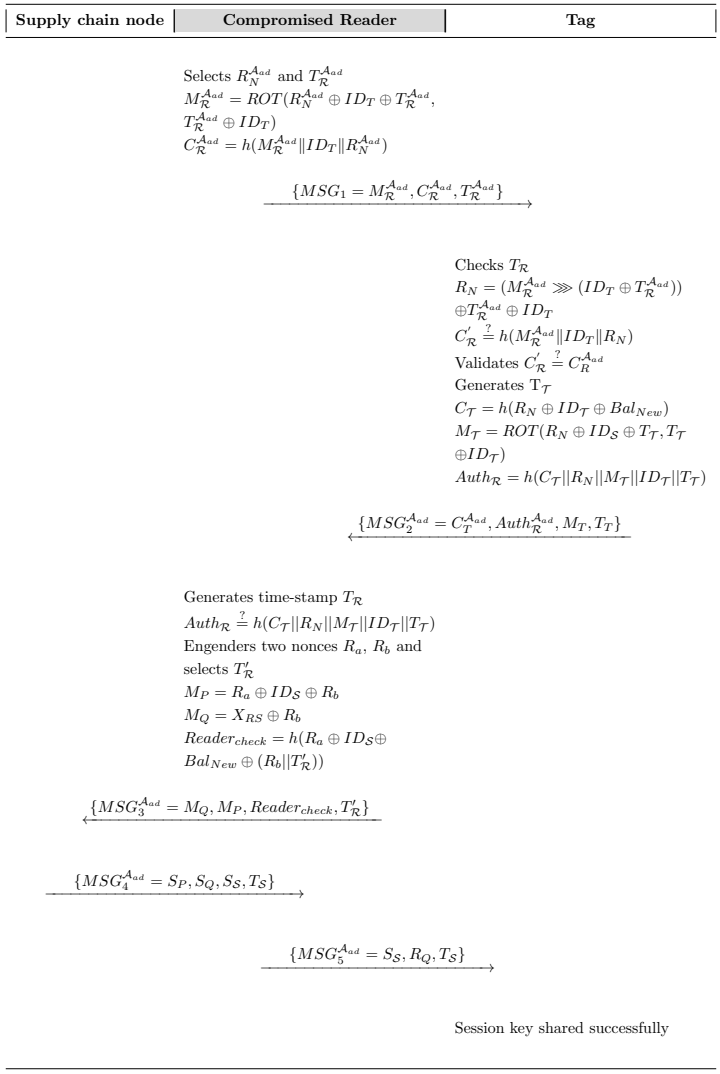

Figure 1. Reader impersonation attack.

Step 2: On receiving message $MSG_2 = \{C_T^{\mathcal{A}_{ad}}, Auth_{\mathcal{R}}^{\mathcal{A}_{ad}}, M_T^{\mathcal{A}_{ad}}, T_T^{\mathcal{A}_{ad}}\}$ from tag, reader checks validity of time-stamp $T_{\mathcal{R}}$ and calculates $Auth_{\mathcal{R}} = h(C_T^{\mathcal{A}_{ad}} \| R_N^{\mathcal{A}_{ad}} \| M_T^{\mathcal{A}_{ad}} \| ID_T \| T_T^{\mathcal{A}_{ad}})$. After the above calculations, reader checks whether $Auth_{\mathcal{R}}^{\mathcal{A}_{ad}} \stackrel{?}{=} Auth_{\mathcal{R}}$. The verification check will be passed and reader sends message to the supply chain, it means that the $\mathcal{A}_{ad}$ has successfully impersonated to a legitimate tag.

Hence, the $\mathcal{A}_{ad}$ can successfully impersonate a valid tag and the protocol is exposed to tag impersonation attack. The detail is illustrated in the Fig. 2.

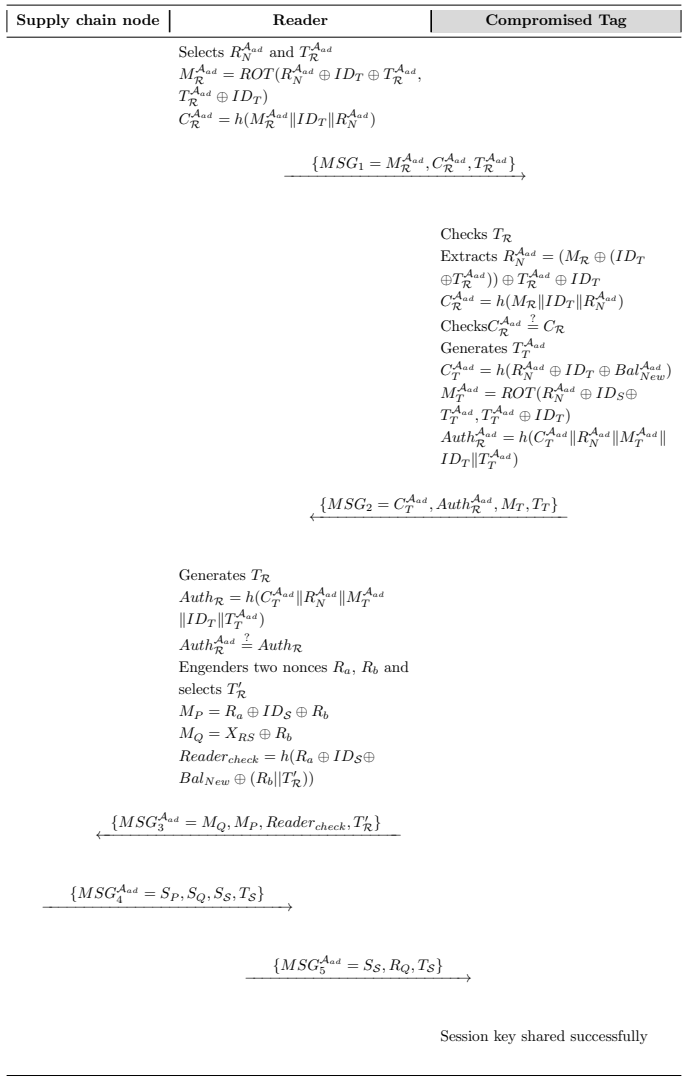

Figure 2. Tag impersonation attack.

3.3. Supply Chain Node Impersonation Attack

The supply chain node and reader share a secret key $X_{RS} = h(ID_S \| B_S \| ID_{\mathcal{R}})$ during initialization phase, which is session specific. The reader and supply chain node has also used this shared secret key during login and authentication phase. However, it is obvious that both the supply chain node and the reader need to store X_{RS} in some memory so that X_{RS} can be used later during login and authentication phase. Suppose, an $\mathcal{A}_{ad}$ has compromised the reader and revealed X_{RS} via power analysis Yang *et al.* (2016). Since ID_T and $ID_{\mathcal{R}}$ is already revealed to an $\mathcal{A}_{ad}$ (Sec. V of refJangirala, Das, and Vasilakos (2019)),

therefore, after extracting these parameters $\{ID_{\mathcal{R}}, ID_T, X_{RS}\}$, supply chain node impersonation attack is possible in this protocol. The detail of this attack is given below.

Step 1: Whenever the reader sends MSG_3 to supply chain, the $\mathcal{A}_{ad}$ intercepts the message $MSG_3 = \{M_Q, M_P, Reader_{check}, T_{\mathcal{R}}^{'}\}$ and saves them for later use. Next, an $\mathcal{A}_{ad}$ requires these parameters $\{S_P, S_Q, S_s, T_S\}$ to impersonate legal supply chain node. To get these parameters, $\mathcal{A}_{ad}$ execute subsequent steps:

Step 2: Firstly, $\mathcal{A}_{ad}$ randomly generates a number $S_{\mathcal{R}}$, T_S and then calculates $S_P = ROT(T_S, ID_S \oplus X_{RS}) \oplus ROT(S_{\mathcal{R}}, X_{RS})$ and $S_Q = ROT(S_{\mathcal{R}}, ID_S) \oplus ROT(T_S, X_{RS})$.

Step 3: The $\mathcal{A}_{ad}$ calculates valid session key $SK_{ST} = h(ID_T \oplus Bal_{BC-REC} \oplus S_{\mathcal{R}} \oplus ID_S)$ and $S_S = h(SK_{ST} \| S_{\mathcal{R}} \| Bal_{BC-REC})$.

Step 4: Then, the $\mathcal{A}_{ad}$ can send request message $MSG_5 = \{S_P, S_Q, S_S, T_S\}$ as legal supply chain node.

Step 5: Upon receiving request message $MSG_4 = \{S_P, S_Q, S_S, T_S\}$, the reader checks validity of time-stamp and calculates $S_{\mathcal{R}}^{'} = RROT(S_Q \oplus ROT(T_S, X_{RS}), ID_S)$. After the above calculation the reader checks whether $S_P \stackrel{?}{=} ROT(T_S, ID_S \oplus X_{RS}) \oplus ROT(S_{\mathcal{R}}^{'}, X_{RS})$. The validation check will be passed then it means $\mathcal{A}_{ad}$ has successfully impersonated to a legal supply chain node.

Hence, the $\mathcal{A}_{ad}$ can impersonate on behalf of legitimate supply chain node and the protocol is exposed to supply chain node impersonation attack. The detail is shown in the Fig. 3.

4. Countermeasures

The protocol is vulnerable to tag and reader impersonation attacks because during the registration phase their identity is stored in their memory and can be extracted. Also, we have seen that the use of bitwise rotation functions is not enough for designing secure protocol, although point multiplication has also be used. Similarly, the supply chain node also stores its secret key in its memory in the registration phase which can also be revealed.

One way to make authentication protocol secure is to use bitwise operations with secure communication using strong crypto-primitives such as one-way hash function, encryption, elliptic curve cryptography and public key cryptography, etc. These techniques require more computation and energy resources. Though, it would be a challenge for attackers to impersonate the tag, reader and supply chain nodes, but this solution is not suitable for low cost tags with energy and computational constraints and also relies on the assumption that the server is trusted and physically secured.

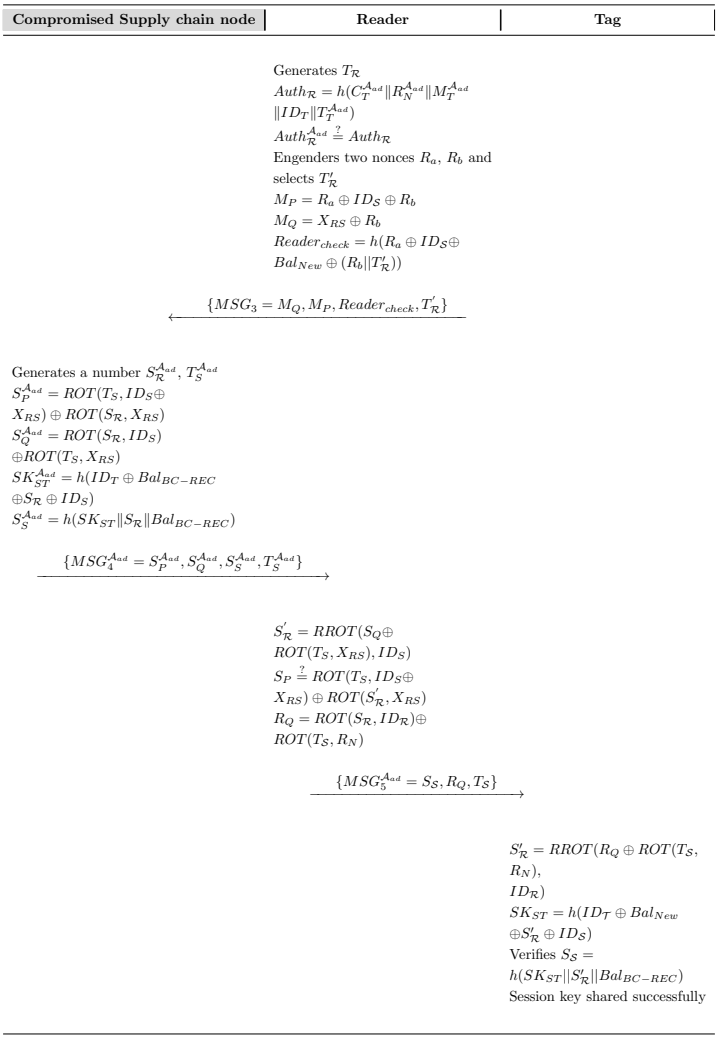

Figure 3. Supply chain node impersonation attack.

The other solution which we propose for light protocol is use of physically unclonable function (PUF)Koeberl *et al.* (2013) can be utilized embedded with the micro-controller of RFID tag and reader. The certificate authority randomly generates a small subset of challenges during registration phase and applies them to the PUF in order to produce a corresponding set responses. For each token the challenge response pair (CRPs) are stored in a secure database by the certificate authority. Later, the CRPs are used for the token authentication. Without access to a given PUF (Weak or Strong), it is impossible for attacker to arrive at response corresponding to a challenge to impersonate tag and reader. Since the output of PUF is always unique and depends on the physical characteristics of the device for which it is determined, therefore, any attempt to temper the memory

of RFID tag or reader will automatically change the behavior of the PUF. The key from PUF can be generated only when required for a cryptographic operation and can be instantaneously erased thereafter. Consequently, the output of the challenge-response pair will be changed and the adversary can be resisted to impersonate as a valid entity.

5. Conclusion

In this paper, we have shown the vulnerabilities associated with using bitwise operations by doing the cryptanalysis of a RFID-based supply chain protocol, which uses operations like ROT and XOR. We have shown that protocol is vulnerable to tag, reader and supply chain node impersonation attacks due to bitwise operations and propose using Physically Unclonable Function for such kind of lightweight protocols. In future, we plan to implement and analyze the RFID PUF-based solution for supply chain.

6. Declarations

Availability of Data and Material
There is no data or any other material associated with this manuscript.

Competing Interests
The authors proclaim that they have no competing interests.

Author's Contributions
MAA and ANM analyzed the requirements of the security for supply chain infrastructure, crypt-analyzed the protocol, and proposed the remedy to overcome the security flaws. The initial draft of manuscript was written by MAA and later reviewed by ANM. Both the authors have read and approve the final manuscript.

Code Availability
Not Applicable

Funding
Not Applicable

Acknowledgement
Not Applicable

References

1 Dabbene, F., Gay, P., Tortia, C.: 2014, Traceability issues in food supply chain management: A review. *Biosystems engineering* **120**, 65. [dabbene2014traceability]

2 Izza, S., Benssalah, M., Drouiche, K.: 2021, An enhanced scalable and secure RFID authentication protocol for WBAN within an IoT environment. *Journal of Information Security and Applications* **58**, 102705. [izza2021enhanced]

3 Jangirala, S., Das, A.K., Vasilakos, A.V.: 2019, Designing secure lightweight blockchain-enabled RFID-based authentication protocol for supply chains in 5G mobile edge computing environment. *IEEE Transactions on Industrial Informatics*. [jangirala2019designing]

4 Koeberl, P., Li, J., Rajan, A., Vishik, C.: 2013, *Offline device authentication and anti-counterfeiting using physically unclonable functions*, Google Patents. US Patent App. 13/313,298. [koeberl2013offline]

5 Mujahid, U., Najam-ul-Islam, M., Sarwar, S.: 2017, A new ultralightweight RFID authentication protocol for passive low cost tags: Kmap. *Wireless Personal Communications* **94**(3), 725. [mujahid2017new]

6 Safkhani, M., Shariat, M.: 2018, Implementation of secret disclosure attack against two IoT lightweight authentication protocols. *The Journal of Supercomputing* **74**(11), 6220. [safkhani2018implementation]

7 Safkhani, M., Camara, C., Peris-Lopez, P., Bagheri, N.: 2021, Rseap2: An enhanced version of RSEAP, an RFID based authentication protocol for vehicular cloud computing. *Vehicular Communications* **28**, 100311. [safkhani2021rseap2]

8 Sidorov, M., Ong, M.T., Sridharan, R.V., Nakamura, J., Ohmura, R., Khor, J.H.: 2019, Ultralightweight mutual authentication RFID protocol for blockchain enabled supply chains. *IEEE Access* **7**, 7273. [sidorov2019ultralightweight]

9 Sun, D.-Z., Mu, Y.: 2017, Security of grouping-proof authentication protocol for distributed RFID systems. *IEEE Wireless Communications Letters* **7**(2), 254. [sun2017security]

10 Xin, X., Zhang, Y., Yang, J.: 2020, Elp2im: Efficient and low power bitwise operation processing in DRAM. In: *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 303. IEEE. [xin2020elp2im]

11 Yang, Q., Gasti, P., Zhou, G., Farajidavar, A., Balagani, K.S.: 2016, On inferring browsing activity on smartphones via usb power analysis side-channel. *IEEE Transactions on Information Forensics and Security* **12**(5), 1056. [yang2016inferring]

Figures

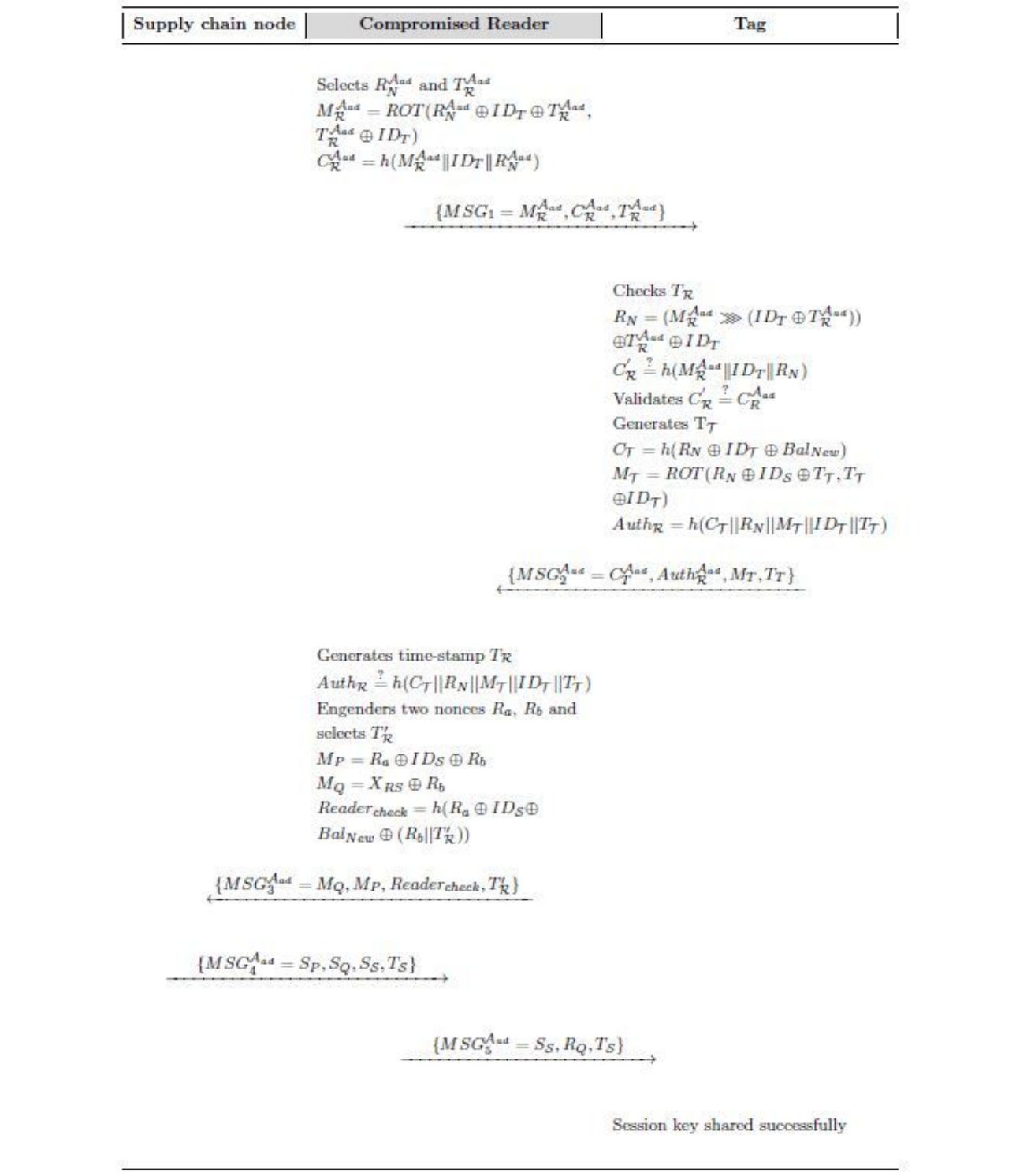

Figure 1

Reader impersonation attack.

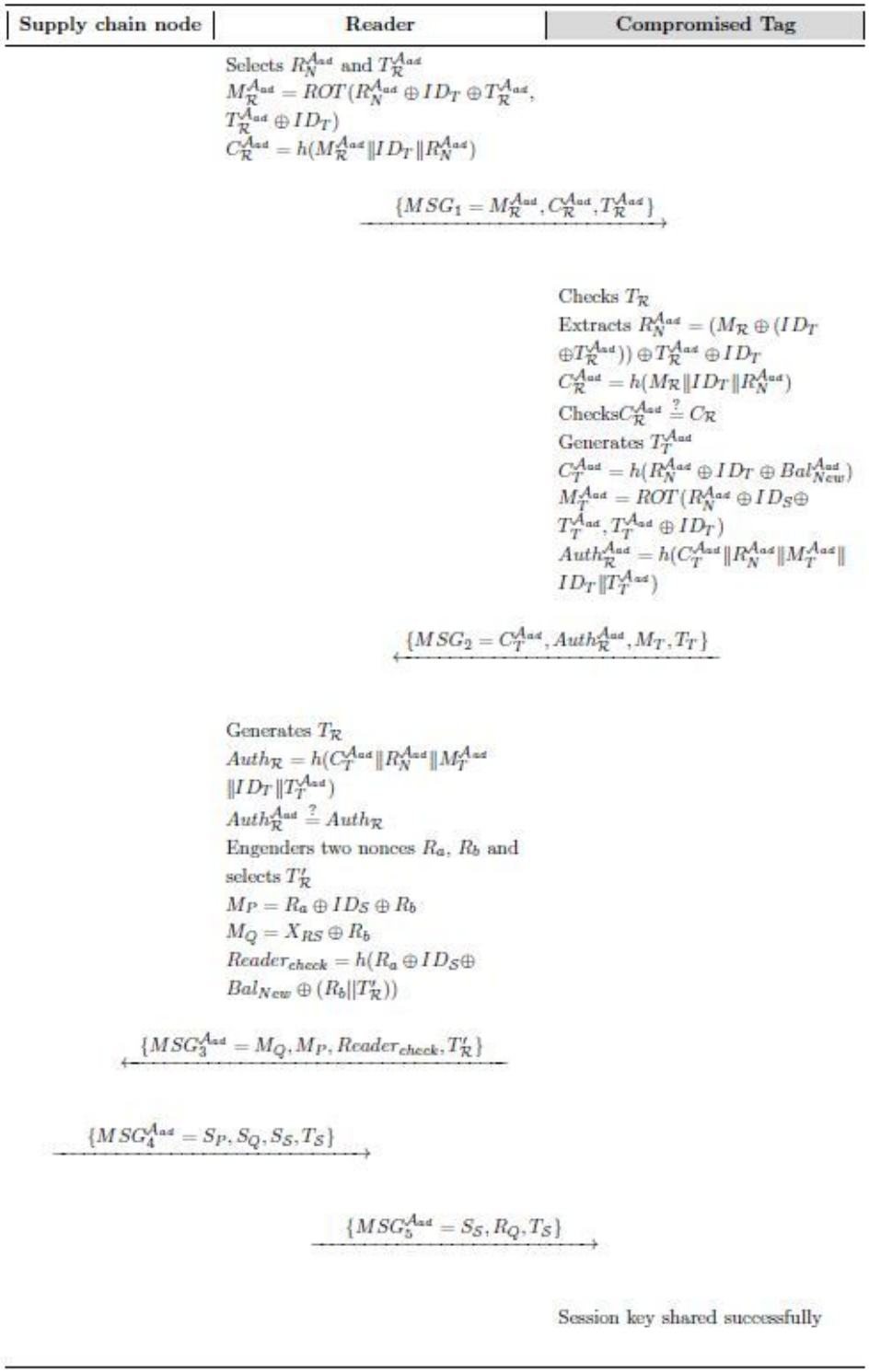

Figure 2

Tag impersonation attack.

Compromised Supply chain node	Reader	Tag
	Generates $T_{\mathcal{R}}$ $Auth_{\mathcal{R}} = h(C_T^{Aad} \| R_N^{Aad} \| M_T^{Aad}$ $\| ID_T \| T_T^{Aad})$ $Auth_{\mathcal{R}}^{Aad} \stackrel{?}{=} Auth_{\mathcal{R}}$ Engenders two nonces R_a, R_b and selects $T'_{\mathcal{R}}$ $M_P = R_a \oplus ID_S \oplus R_b$ $M_Q = X_{RS} \oplus R_b$ $Reader_{check} = h(R_a \oplus ID_S \oplus$ $Bal_{New} \oplus (R_b \| T'_{\mathcal{R}}))$	
	$\longleftarrow \{MSG_3 = M_Q, M_P, Reader_{check}, T'_{\mathcal{R}}\}$	
Generates a number $S_{\mathcal{R}}^{Aad}$, T_S^{Aad} $S_P^{Aad} = ROT(T_S, ID_S \oplus$ $X_{RS}) \oplus ROT(S_{\mathcal{R}}, X_{RS})$ $S_Q^{Aad} = ROT(S_{\mathcal{R}}, ID_S)$ $\oplus ROT(T_S, X_{RS})$ $SK_{ST}^{Aad} = h(ID_T \oplus Bal_{BC-REC}$ $\oplus S_{\mathcal{R}} \oplus ID_S)$ $S_S^{Aad} = h(SK_{ST} \| S_{\mathcal{R}} \| Bal_{BC-REC})$		
	$\longrightarrow \{MSG_4^{Aad} = S_P^{Aad}, S_Q^{Aad}, S_S^{Aad}, T_S^{Aad}\}$	
		$S'_{\mathcal{R}} = RROT(S_Q \oplus$ $ROT(T_S, X_{RS}), ID_S)$ $S_P \stackrel{?}{=} ROT(T_S, ID_S \oplus$ $X_{RS}) \oplus ROT(S'_{\mathcal{R}}, X_{RS})$ $R_Q = ROT(S_{\mathcal{R}}, ID_{\mathcal{R}}) \oplus$ $ROT(T_S, R_N)$
		$\longrightarrow \{MSG_5^{Aad} = S_S, R_Q, T_S\}$
		$S'_{\mathcal{R}} = RROT(R_Q \oplus ROT(T_S,$ $R_N),$ $ID_{\mathcal{R}})$ $SK_{ST} = h(ID_T \oplus Bal_{New}$ $\oplus S'_{\mathcal{R}} \oplus ID_S)$ Verifies $S_S =$ $h(SK_{ST} \| S'_{\mathcal{R}} \| Bal_{BC-REC})$ Session key shared successfully

Figure 3

Supply chain node impersonation attack.