ENHANCING SECURITY OF RFID-ENABLED IOT SUPPLY CHAIN

Halit Türksönmez[1*], Mehmet Hilal Özcanhan[2]

[1,2]Department of Computer Engineering, Dokuz Eylul University, Izmir, Turkey

Email: halit.turksonmez@ogr.deu.edu.tr[1*] (corresponding author), hozcanhan@cs.deu.edu.tr[2]

ABSTRACT

In addition to its benefits, the popular Internet of Things (IoT) technology has also opened the way to novel security and privacy issues. The basis of IoT security and privacy starts with trust in the IoT hardware and its supply chain. To ensure reliable IoT industry growth, counterfeiting, cloning, tampering of hardware, theft, and lost issues in the IoT supply chain must be addressed. Radio-frequency identification (RFID)-enabled solutions to bring security to the IoT supply chain have been proposed, by the same authors in four previous works. The works contain a similar RFID-traceable hardware architecture, device authentication, and supply chain tracing procedure. However, the same lightweight RFID authentication protocol variant proposal coupled with the offline supply chain has security vulnerabilities that make the whole supply chain unsafe. Our work proposes an online supply chain hop-tracking procedure supported by a novel RFID mutual authentication protocol based on strong matching of RFID readers, their operators and the central database server. The proposed Strong RFID Authentication Protocol (STRAP) has been verified by two well-accepted formal protocol analyzers Scyther and AVISPA. The verification results demonstrate that STRAP overcomes the previous works' vulnerabilities. Furthermore, our proposed novel online supply chain tracing solution removes the weaknesses of previous offline supply chain tracking solutions.

Keywords: ***Authentication, Hardware Security, Internet of Things, IoT Supply Chain, IoT Security***

1.0 INTRODUCTION

The Internet of Things (IoT) can be defined as a network of heterogeneous computation-capable entities. Network-enabled sensors and actuators embedded in present-day ubiquitous systems sit on top of the Internet infrastructure. Running on the Internet, IoT is changing the habits of modern society by facilitating daily human life on a global scale. Adapting well to individual context awareness, IoT is changing our lives at home, work and how we conduct our business. The number of connected IoT devices on earth is estimated to nearly triple, from 11.08 billion predicted in 2021 to more than 29.42 billion in 2030 [1]. The speed of IoT transformation is accelerating because of the progress in cloud computing and wireless sensor networks [2]. One of the benefiting industries from IoT expansion is the worldwide supply chains of commercial goods. IoT devices track the journey of critical and valuable goods from anywhere and anytime. In fact, IoT is helping monitor the increasing global goods traffic, which is the most cumbersome outcome of supply chains. Monitoring provides determination of goods transfer system malfunctions and their pinpoint localizations quickly.

However, IoT does not only offer benefits, but it also brings new security and privacy issues [3]. Not only the goods themselves but the IoT supply chain itself requires protection. While the traditional Internet was a network of uniform computer devices, unfortunately IoT exhibits heterogeneity. IoT devices are often resource-constrained in computing power, wireless operation distance, and storage capabilities. As a result, providing IoT device security and privacy is much harder than conventional systems. Therefore, protecting both the IoT devices in the supply chains and the IoT-tracked supply chains is not an easy task. A general structure of the IoT supply chain and its vulnerabilities are shown in Figure 1.

At the Integrated Circuit (IC) design level, intellectual property theft is a different scope than our present study. At the printed circuit board (PCB) design-fabrication-assembly-integration levels, the produced IC or PCB can be physically tampered with, overproduced, cloned (counterfeited), or tampered [4-9]. For example, hardware Trojans can be planted in IoT devices. In the distribution stage, Authentic IoT devices may be mixed with clones/fakes by untrusted supply chain partners, or their expensive components may be replaced/stolen. Finally, after their lifetime, IoT devices may be recycled or reintroduced into the market; or the supply chain.

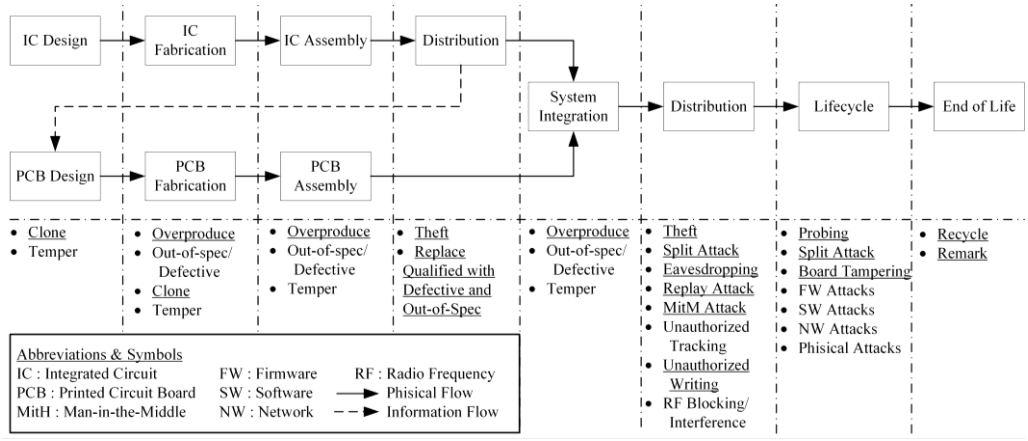


Fig. 1: The general structure of the IoT supply chain and its vulnerabilities at different stages

Once stolen, cloned or recycled, IoT devices can be installed at places other than their originally intended locations. As a result, both the IoT devices in the supply chain and the supply chains itself need protection from cyber-attacks. Challenges in IoT technology security are progressive and must be responded to with equal counter-research.

Authors of works [10-13] have proposed solutions for supply chain security and privacy issues of IoT devices and their supply chain. The confidentiality of exchanged messages of the proposed authentication protocols is provided by using symmetric encryption using Advanced Encryption Standard (AES). Message integrity is provided by the keyed Hash Message Authentication Code (HMAC). Even though the methods used by the authors can prevent most known cyber-attacks, their proposed authentication protocols contain weaknesses that destroy the security of the whole solution. In fact, the absence of an official announcement about secure supply chain standards proves that the issue still needs to be satisfactorily addressed. Consequently, the weaknesses in the solutions of works [10-13] motivated us to provide better security for IoT devices with embedded RFID tags (RFID-enabled) and RFID-enabled supply chains. Lack of support for innovative academic solutions in supply chain security aggravates the challenge but strengths our motivation in our present work [14].

Our contributions with respect to meeting the security challenges faced in proposals [10-13] are:

1. Removal of the inflated assumptions of trusted RFID readers in supply chains,
2. Removal of malicious employee (insider) threats assumption,
3. Provision of strong mutual authentication of supply chain server computers and embedded RFID tags at every supply chain hop,
4. Provision of shared secret keys enumerated and controlled by the server,
5. Provision of new (altering) session keys for all tags in every new round.

In the rest of the present work, we explain the IoT devices' supply chain hops in detail, in the next Section. Then, we present the comprehensive solution proposed by a group of researchers in four different works [10-13]. Next, we indicate the vulnerabilities caused by the false assumptions and weaknesses both in the authentication protocols and the offline supply chain hop-transfer procedure, in Section 2. In Section 3, we propose an online IoT supply chain hop-transfer procedure supported by our novel Strong RFID Authentication Protocol (STRAP). The informal and formal security analysis of our proposed protocol is provided in Section 4. In Section 5, the performance and security of ours and previous proposals are compared. Finally, we conclude in Section 6.

2.0 RELATED WORK

Among various studies, the authors of four articles present comprehensive proposals for providing security to the supply chain of IoT network devices. The four RFID-enabled Supply Chain (ReSC) authentication protocols are detailed below. Afterward, the vulnerabilities of the protocols are exposed and attacked. The result of our presented attacks is disruption of the ReSC supply chains, or breach of the IoT devices traveling through the supply chain. Either event is devastating for the supply chain.

2.1 ReSC-1 Supply Chain Framework

In work [13], Yang *et al*. proposed an RFID-enabled solution (named ReSC-1) to secure and manage an entire supply chain and its hops. ReSC-1 contains RFID readers, RFID-enabled network devices, locations of each reader, and a centralized database server (DB) for storing all information (e.g., tag identities, control chip identities, tag traces) and authenticating the traveling devices. Figure 2 shows the proposed ReSC-1 setup with three stages and three transitions. Each stage is described below:

Integration Stage. This is the production stage of the network device, when the Integrated Circuits (ICs, electronic chips) are produced and assembled. The control chip identification number (CC_ID) are generated using the physically unclonable function property of the embedded SRAM memory inside the chips. The embedded RFID tags are also assigned identification numbers (T_ID). Both identification numbers are stored in the DB. The stored information is used to monitor and verify the device's journey through the supply chain. Initializations and functionality tests are finalized to ensure proper hardware and software functioning.

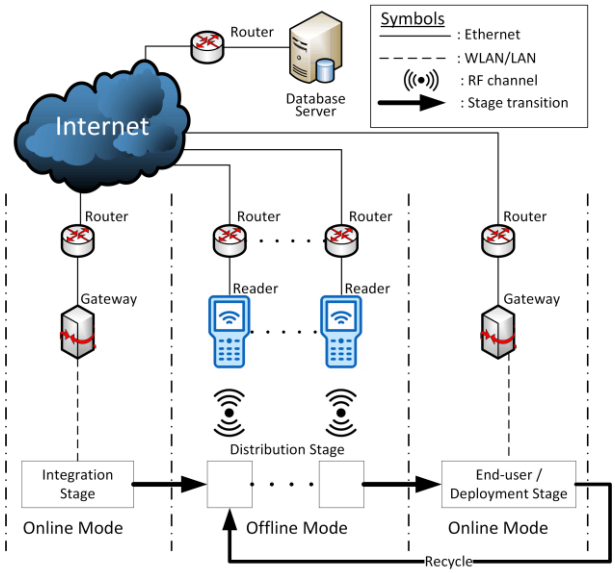


Fig. 2: Supply chain stages and transitions

Distribution Stage. The temporary storage and transfer (transportation) of network devices are carried out by supply houses, distributors, retailers, and installers. ReSc-1 transportation activities are followed offline. The devices are identified at the hops, by authenticating the embedded tag identities using readers/smartphones. Each hop reader adds its signature to the tag's memory and contributes to the unique tag trace. The tag traces are uploaded to the DB, when the readers get online. It is assumed that the DB authenticates the readers using their secret identities in its list.

Deployment Stage. In the final stage, network devices are dispatched to end-user premises, powered on, and connected to the DB over a secure link. Online authentication of the device and the validation of the unique tag trace is attempted. The device whose tag trace in its memory is invalid cannot be put to service and returned for control and redistribution.

2.1.1 ReSC-1 Authentication Protocol

The three-step, lightweight authentication protocol of ReSC-1 is shown in Figure 3. The protocol tries to verify the T_ID – CC_ID match and the tag trace. The unique CC_ID assigned at the system integration stage is matched to the unique T_ID of the embedded tag. Together (CC_ ID, T_ID) form a tuple in the DB. The signed tag trace is transmitted offline to the DB and stored as a unique supply chain hop record in the tag memory. At the end of the protocol, the tag trace is expected to carry all the necessary signatures of the authorized readers. The tag trace is validated by matching it with the DB records. The steps of ReSC-1 protocol are explained below:

At first, reader R_i interrogates tag T at a hop, using a *Query*. Tag T answers with its identity T_ID, previous stage reader values of $Indx_{i-1}$, timestamp TS_{i-1}, and signature Sgn_{i-1}. Previous reader R_{i-1} calculates the hash and then signs

it using its private key SKy_{i-1}; hence the equation $Sgn_{i-1} = Hsh_{SKyi-1}(T\_ID \,||\, Indx_{i-1} \,||\, TS_{i-1})$. Present reader R_i decrypts Sgn_{i-1} using the public key PKy_{i-1} of the reader R_{i-1}, and gets the hash value of $T\_ID \,||\, Indx_{i-1} \,||\, TS_{i-1}$. The symbol || indicates concatenation. Reader R_i calculates the same concatenation value locally, and authenticates reader R_{i-1}, if the calculated and decrypted hash values match. After successful authentication, reader R_i forms its signature Sgn_i by hashing $T\_ID \,||\, Indx_i \,||\, TS_i$ and signing it using its private key SKy_i. Then R_i transmits Sgn_i, $Indx_i$, TS_i to tag T, and updates the tag trace in the device memory. The complete tag trace is validated by matching it with the DB records, at the end of the supply chain.

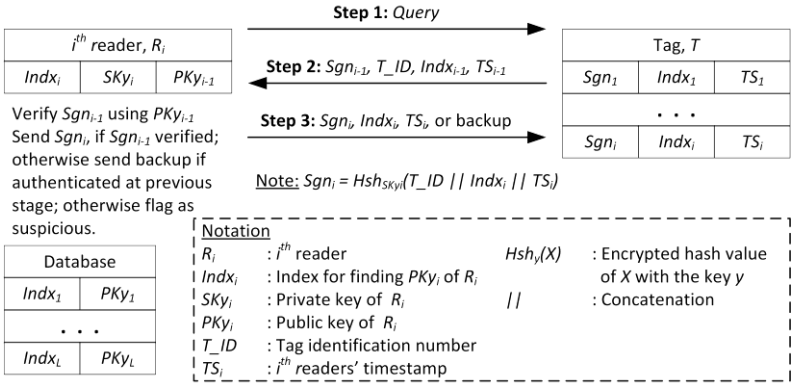


Fig. 3: Lightweight RFID protocol used in ReSC-1

2.1.2 Assumptions and Vulnerabilities of ReSC-1

In the RFID community, there are commonly respected assumptions that are constantly revised. The channel between the reader and the server is generally assumed to be secure. In contrast, the wireless channel between the readers and the tags is accepted to be insecure. The readers ─one of the peers of the wireless message exchange─ are known to be resourceful devices that support sophisticated cryptographic tools for encryption and decryption. Nowadays, it is commonly assumed that illegal (rogue) readers are also present in open wireless environments [15, 16]. Hence, RFID attacks mostly target the readers and the message exchanges between the readers and the tags. This development is documented in the formal Dolev-Yao model, where adversaries are assumed to be capable of listening, blocking, changing, and injecting messages into an exchange [17]. However, ReSC-1 protocol contradicts the Dolev-Yao model and the commonly accepted threats by assuming a trustful integrator, reader operator, and secure readers [13].

As a result, ReSC-1 has serious security weaknesses that jeopardize the security of the whole supply chain. The first weakness is the uploading of pre-calculated session keys, by the system integrator. The DB would have no knowledge in case initialization information is captured. A second weakness is the lack of matching and authentication of the reader with its operator by the DB, at each hop point. Thus, allowing a malicious user to use a legal or rogue reader is awarded. A third vulnerability is in Step 2 of Figure 3. The signature Sgn_{i-1}, the tag T_ID, the index $Indx_{i-1}$, and the timestamp TS_{i-1} of the previous hop are sent to the reader in plaintext. Even identification of the hash function is unnecessary, because the inputs accompany the result. An attacker team can obtain all Sgn_{i-1} and all their corresponding inputs to organize an attack with the help of reader operators. The last, simple but devastating vulnerability is at Step 3. A rogue reader can transmit a false tag trace backup or block the last message before the batch is transferred to the next hop, forcing a corrupted trace. Exploiting the above weaknesses, next we demonstrate devastating attacks on ReSC-1.

2.1.3 Launching Attacks on ReSC-1

Obviously, an attacker can be assumed to possess an ordinary or a legal ReSC-1 reader, according to Dolev-Yao's attacker model. Alternatively, the attacker can be a malicious user who can tamper with the reader's memory to obtain the secrets passing through or inside it. Furthermore, the hop numbers, locations, and the transportation path of supply chains are public in present-day trading. Therefore, designing an attack path by placing either a rogue reader, or operator at hop locations is possible. Based on its weaknesses, the following attacks can be launched on ReSC-1.

2.1.3.1 Disrupting the Supply Chain: Tag-Trace Contamination Attack on ReSC-1

The simplest attack can be launched by blocking the last message of ReSC-1, which disrupts the supply chain. After the message in Step 3 (Figure 3) is blocked, the batch continues to the next hop, with nobody aware of the missing previous tag trace. Hence, the corrupted trace records result in the rejection of the delivered batch. A slightly more sophisticated attack is also possible when a rogue reader transmits a false tag trace backup, in Step 3. The false Sgn_i again contaminates the tag trace, and the result is once more the supply chain disruption.

2.1.3.2 Tag Cloning by Hop Table Attack on ReSC 1

Attackers placed at the hop locations can query all the RFID tags of the devices with a legal reader. St every hop, the second and third step messages (Figure 3) are recorded. Hence, the recording of all the tag traces is completed at the last hop. Now, the attacker team has all legal device information. The records are copied to the fake RFID tag memories of illegally produced clones. The illegal devices only need a legal CC_ID. The CC_IDs are produced from the start-up signatures of the embedded static random access memories (SRAM), at the integration stage. SRAM signatures are physically unclonable function (PUFs) security primitives for generating IDs [14, 18-20]. Although PUFs generate unique CC_ID for each IC, they are vulnerable to side-channel attacks [21]. ReSC-1 authors wrongly assume that the integration stage is a trusted environment. However, the above malicious ring can obtain the CC_IDs of the devices while they are at the integration stage. Hence, an illegal device with a fully cloned RFID tag memory can pass the authentication at the end user stage of Figure 2. The legal devices are now in circulation.

2.2 ReSC-2 and ReSC-3 Supply Chain Framework

The ReSC-1 proposal was upgraded twice by the authors, which we named ReSC-2 and ReSC-3, respectively [10, 12]. The aim of protecting IoT devices and their supply chain remained the same. The same supply chain framework and hardware architecture shown in Figure 2 were used. Again, the protocol is the creation of a tag trace in the tag memory, validated by the signatures of the readers on the distribution path. The authentication protocol between the device tags and the readers is shown in Figure 4. *SE* designates symmetric-key encryption using the AES algorithm.

2.2.1 ReSC-2 and ReSC-3 Authentication Protocols

The authentication schemes of ReSC-2 and ReSC-3 consist of the same number of steps and phases as in ReSC-1. However, ReSC-3 has an additional feature named Neighborhood Attestation, which aims to detect theft at the end-user stage, by verifying the status of its neighbors. This side-procedure is out of our scope because there are more serious vulnerabilities before reaching the end user stage.

At first, the system integrator loads a set of session keys ($SnKy_1$, $SnKy_2$, …, $SnKy_n$) generated by encrypting each tag identity *T_ID* (Figure 4). Each distribution path RFID reader secret key Ky_{Ri} shared between the DB and readers is used in encryption. The Session key ($SnKy_i$) is used to encrypt the tag and the corresponding reader (R_i) communication. When the IoT device arrives at an intermediate hop, R_i first sends a *Query* with a random number *Non1* to the tag. The tag generates a new random number *Non2* and encrypts the tuple (*Non1*, *Non2*) using its session key $SnKy_i$. Then, the tag replies with its identity (*T_ID*) and the ciphertext $SE_{SnKyi}(Non1, Non2)$, in Step 2.

In Step 3, the reader R_i generates the session key $SnKy_i$ locally, by encrypting the tag identity (*T_ID*) using the master key Ky_{Ri}. Then, reader R_i authenticates the tag by decrypting $SE_{SnKyi}(Non1, Non2)$ and validating *Non1*. If the tag is authenticated, reader R_i generates its signature (Figure 4). $Indx_i$ is the index associated with the i^{th} reader, TS_i denotes the specific time when reader R_i updates the tag, || indicates concatenation operation, and $Hsh_{SKyi}(X)$ is the hash value of the input X using SKy_i (the private key of reader R_i). Next, the reader R_i encrypts the quad as $SE_{SnKyi}(Non2, Sgn_i, Indx_i, TS_i)$, using the session key $SnKy_i$, and transmits it to the tag, for trace update. The tag authenticates the reader R_i by decrypting $SE_{SnKyi}(Non2, Sgn_i, Indx_i, TS_i)$ and validating *Non2*. If the reader R_i is authenticated, the tag stores the reader update (Sgn_i, $Indx_i$, TS_i) in the tag memory.

2.2.2 Assumptions and Vulnerabilities of ReSC-2 and ReSC-3

ReSC-1 assumptions are preserved in ReSC-2 and ReSC-3. Although ReSC-2 and ReSC-3 are new versions of ReSC-1, the same security vulnerabilities exist. Again, the first vulnerability is uploading session keys in tag memories, by the assumedly trusted system integrators [10, 12]. The second is the repeated lack of authentication and matching between legal readers and operators, and the verification of DB. A third vulnerability is the plaintext

transmission of *T_ID* and *Non1* in steps 1 and 2. The plaintext messages are not needed in our attacks, but can be used in others. The fourth vulnerability is the missing acknowledgment message in Step 3; another unrectified ReSC-1 vulnerability.

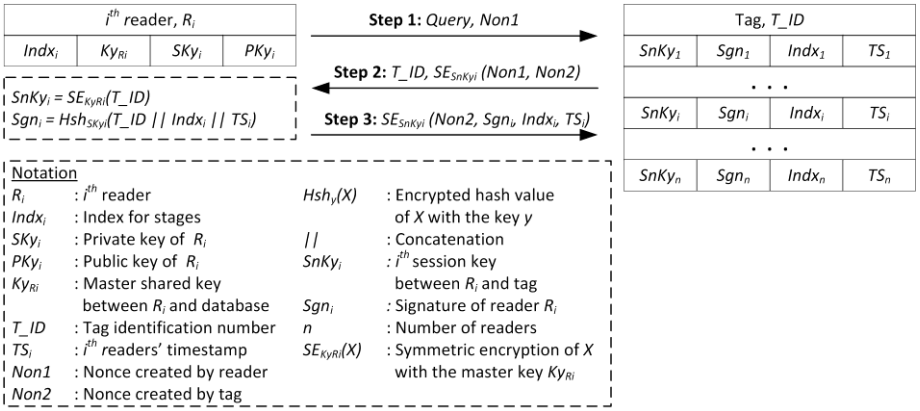


Fig. 4: Lightweight RFID protocol used in ReSC-2 and ReSC-3

2.2.3 Launching Attacks on ReSC-2 and ReSC-3

With the same attacker model of ReSC-1, attackers are assumed to have access to legal or illegal ReSC-n (n is the hop number) readers, thus access to the secrets in reader's memory. Furthermore, the supply chain hops are known by the employees. Therefore, forming a malicious organization by placing a rogue reader/operator at each hop is possible. Due to the above realistic vulnerabilities, we next demonstrate two attacks on ReSC-2 and ReSC-3.

2.2.3.1 Disrupting the Supply Chain: Tag-Trace Contamination Attack on ReSC-2 and 3

The step-by-step procedure of the disruption attack launched on ReSC-2 and ReSC-3 is as follows:

1. Fabricate a constant *Non1* (Figure 4),
2. Query all tags with any reader using the fabricated *Non1*,
3. After receiving the replies in Step 2, play a fabricated reply to all tags,
4. Repeat the above three steps infinite times with automated software.

At the end of the attack, tags that have *Non2* match by coincidence, will update their memory with fake information and increment their address pointers. Now, the tag has a corrupted tag trace that nobody is aware of. The above attack can be launched before or after the tags are queried by a legal reader; or even while in transit. Repeating the attack at many hops increases the success rate of the tag-trace contamination. Therefore, the supply chain of all ReSC solutions can be corrupted and disrupted.

2.2.3.2 Tag Cloning by Full Disclosure Attack on ReSC-2 and 3

By tampering, a malicious integrator employee can obtain the CC_ID, session keys $SnKy_i$, and *T_ID*s of the devices. Then, the employee may become a ring member or sell the information to a malicious ring. The reader key Ky_{Ri} and the session key $SnKy_i$ are also lost because they are in the memory of the exposed reader. Now, launching an attack is trivial because the attackers can capture *Non2* transmitted in Step 2. The message in Step 3 is formed and sent to the tag. The tag hop trace is captured. At the last hop, tag trace capture is completed. Thus the attacker ring has a full tag trace of all genuine devices. Now, the attacker ring can clone the tag and fabricate a fake device with a valid CC_ID. The illegal device passes the authentication at the End-User Stage and hence pirate network devices with cloned RFID tags with valid tag traces and CC_ID can be put into circulation for malicious trading.

2.3 CDTA Supply Chain Framework

Yang *et al.* presented yet another RFID-enabled solution named Counterfeit Detection, Traceability, and Authentication (CDTA) in work [11]. Although the name is different, the same ReSC supply chain framework is used (Figure 2). Therefore, it would be appropriate to put CDTA in the ReSC family. CDTA aims to detect all

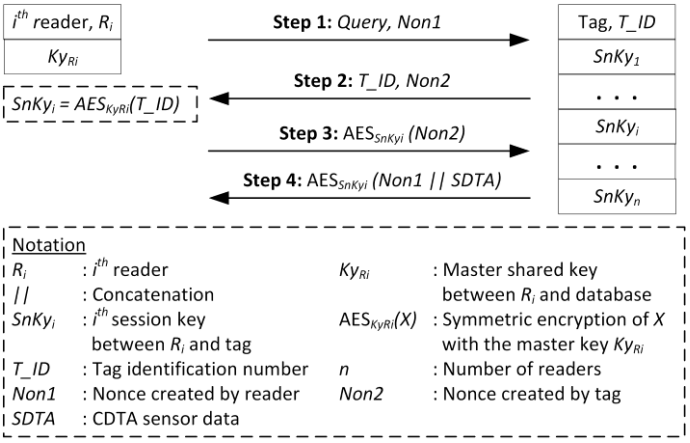


Fig. 5: Lightweight RFID protocol used in CDTA

counterfeit chips used in IoT devices with a new hardware architecture. Four types of sensors with different functionalities are used in CDTA [22-26]. At least one embedded sensor transmits data periodically to the RFID chip to be stored in its Electrically Erasable Programmable Read Only Memory (EEPROM).

2.3.1 CDTA Authentication Protocol

Figure 5 shows the lightweight authentication protocol of CDTA. The tag memory has a static, read-only *T_ID*, a set of session keys ($SnKy_1$, $SnKy_2$, ..., $SnKy_n$) where *n* is the number of readers on the supply chain, and the collected sensor data. Before entering the supply chain, the system integrator assigns the session keys. Each session key ($SnKy_i$) is used to encrypt the communication between the tag and the present hop reader (R_i). Surprisingly, the session key ($SnKy_i$) is the encryption of the constant *T_ID* {$SnKy_i = AES_{KyRi}(T\_ID)$}, where Ky_{Ri} is the master key shared between R_i and the DB.

When the IoT device arrives at a hop, the reader R_i sends a *Query* and a random number *Non1* to the tag. The tag simply replies with its identity (*T_ID*) and random number *Non2* in plaintext. After receiving *T_ID* and *Non2*, reader R_i generates the session key $SnKy_i$, as shown in the above paragraph. Then, the reader R_i encrypts *Non2* using $SnKy_i$ and transmits it to the tag. The tag authenticates the reader by decrypting $AES_{SnKyi}(Non2)$ and validates *Non2*. Next, the tag encrypts *Non1* || *SDTA* using the session key and transmits the encrypted concatenation to the reader. The reader R_i authenticates the tag by decrypting $AES_{SnKyi}(Non1 \,||\, SDTA)$ and validates *Non1*. Then, the reader trusts the device and stores the sensor data *SDTA*.

2.3.2 Assumptions and Vulnerabilities of CDTA

The same assumptions of the ReSC-n family are repeated for CDTA [11]. The devastating weaknesses of no authentication; or no legal reader─user matching is carried over to the CDTA. The same assumptions exist, such as a secure system integrator and no malicious hop operators or employees. Another vulnerability is handed to attackers at Steps 1 and 2 of the protocol (Figure 5), when both *Non1* and *Non2* are transmitted in plaintext. The vulnerability opens the avenue to plaintext-ciphertext matching and lookup table creation. The last vulnerability is the offline data stored on the reader that can be captured through memory tampering. However, disrupting the CDTA supply chain with the previous ReSC-n methods is impossible, because now the tag has a second reply added as the fourth step.

2.3.3 Tag Cloning by Full Disclosure Attack on CDTA

Due to the common ReSC-n protocol vulnerabilities, the device's session keys $SnKy_i$ and *T_ID* are open to theft. As before, the attacker corresponds the $SnKy_i$ with the *T_ID* and index number $Indx_i$. Thus, capturing the critical information is possible, as in ReSC-2 and ReSC-3. A rogue reader starts the attack with any fake *Non1*. The tag responds with *T_ID* and *Non2* tuple in plaintext. The rogue reader encrypts the valid *Non2* with the compromised session key $SnKy_i$ and returns it to the tag, in Step 3. After decrypting the received message, the RFID tag authenticates reader R_i. Then, the tag encrypts the concatenated *Non1* and sensor data *SDTA* with the session key $SnKy_i$, and transmits it to the rogue reader. After decrypting the message, the rogue reader captures the critical

sensor data. Thus, the attacker ring again has all the hop tag trace information necessary to clone RFID tags and introduce fake network devices with valid CC_IDs, into the supply chain. The fake network devices pass the authentication test at the End-User Stage because the valid SDTA has been captured.

3.0 THE PROPOSED AUTHENTICATION PROTOCOL

The above attacks demonstrate how the breakdown of a weak authentication protocol can devastate a whole supply chain. Therefore, stronger authentication protocols are needed in supply chains. In our novel solution, a smartphone equipped with a near-field communication (NFC) reader is used as the mediator between the NFC tag of the IoT device and the DB. Instead of using a theoretical tag, we propose MIFARE DESFire™ EV3 (EV3) produced by NXP Corporation. The EV3 is a perfect solution for flexible, wireless applications because it is Common Criteria EAL5+ security certified (the same security certification level demanded in banking smart cards or electronic passports). The high-level security is provided by either a cryptographic Triple Data Encryption Algorithm (3DES), or an AES algorithm hardware engine. In addition, the EV3 complies with the ISO/IEC 14443A standards and supports optional ISO/IEC 7816-4 commands. The EV3 receives its energy from the smartphone electromagnetically. The data transmission speed can go up to 848 kbit/s. The non-volatile card memory can go up to 8 kB. Thus, EV3 fulfills all the essential technical requirements of our proposed protocol in computation power, encryption, data transfer speed, and memory space [27].

The remote DB contains all cryptographic keys and the tag *T_ID* of each IoT, or network device. While the DB─smartphone connection is over the Internet, the smartphone-tag communication channel is a near-field wireless connection. Thus, all of the hops on the supply chain path are in online mode. The proposed technique in our novel protocol consists of three phases. These are:

1. Linking a legal smartphone with a legal user,
2. Mutual authentication of DB and tag with the generation of a secure session key,
3. IoT device status tracking by checking SDTA status and signature chain.

Our novel protocol is **ST**rong **R**FID **A**uthentication **P**rotocol for IoT supply chain, or briefly **STRAP**. Each tag's secret keys are created, enumerated and transmitted to the manufacturer over a secure channel by the DB. The *T_ID*s of the generated tags are passed on to the intellectual property (IP), or DB owner in an ordered list. The manufacturer should add the keys to the tags in the order of the *T_ID* list given to the IP owner. The tuples (*T_ID*, *key*) formed are saved in the DB. The generation of keys and tag identification numbers are preserved in our work and therefore omitted.

The assumptions of STRAP comply with the Dolev-Yao model. Therefore, they are more realistic than the ReSC family. Our assumptions are:

1. The air channels between the RFID or NFC tags and smartphones are not secure,
2. The smartphone and its operator cannot be trusted,
3. Valid user identification and user key (UID, UKey) pairs are generated and kept by the DB,
4. Smartphone Central Processing Unit Identification numbers (CPUIDs) are pre-registered in the DB.

3.1 Phases of STRAP

3.1.1 Phase-1 - Smartphone and User Registration and Matching

The protocol starts by linking an authenticated user with a valid smartphone, as shown in Figure 6. The RFID tag does not have any role in Phase-1. The k^{th} user has to enter username UID_k and password in the custom-developed application on a legal smartphone. The password is padded and encrypted with a utility program, like the *crypt* command in Linux, to form the $UKey_k$. The application encrypts the smartphone's $CPUID_{Spi}$ with the system time t_{Spi} and forms the message Msg_1, before transmitting Msg_1 and UID_k to the DB through the unsecured air medium. Details of encryption-specific, concatenation and padding operations are not the focus of the present work. Therefore, encryption details are not presented. The DB finds $UKey_k$ (user's key) in its database by using UID_k (user's login name) and decrypts Msg_1. Then, DB searches the database for a $CPUID_{Spi}$. Once the smartphone is verified as a valid device, the specified user is paired with it and smartphone-user bonding completes. The system time t_{Spi} of the smartphone is recorded.

Then, DB calculates the hash value of the tuple ($CPUID_{Spi}$, t_{Spi}), which proves that DB has the user's password to

decrypt the message Msg_1. Next, DB prepares message Msg_2 by encrypting the hash of ($CPUID_{Spi}$, t_{Spi}) with the user's public key and then signing the encryption with its private key. Then, DB sends UID_k with Msg_2 to the smartphone. The smartphone generates the same hash while waiting, then decrypts Msg_2 and gets the hash value generated by DB, if and only if it has the public key of DB and the user's private key. If the received and calculated hash is equal, then DB is authenticated. Thus, the mutual authentication of the DB server and the user, and the bonding of the smartphone to the user is completed. At the same time, DB finds the hop number using the user name and determines which key number (*KyNo*) to use in the next phase of STRAP. This process connects STRAP's Phase 1 to Phase 2, thus resisting any attacks that may exploit detached authentication phases. From this moment on, the smartphone is just a message mediator.

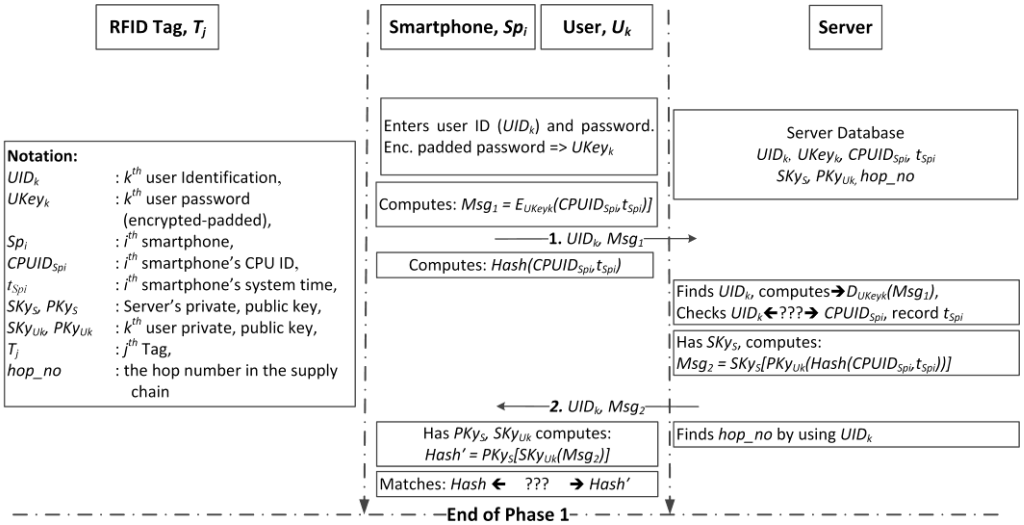


Fig. 6: Phase-1 of STRAP, "The matching of an authentic user with a legal smartphone"

3.1.2 Phase-2 - Session Key Generation, Mutual Authentication of Server and RFID Tag

As shown in Figure 7, the legal and user-bonded smartphone requests the ID of the targeted RFID tag. The RFID tag transmits its ID number (T_j_ID) to DB. Using the *hop_no* determined in Phase-1, DB server discovers the tag in its database and discovers the *KyNo* and KyX_{Tj}. The server generates the random number $Nonce_S$ and encrypts it using the tag's secret key KyX_{Tj}. The encrypted nonce is transmitted to the tag with the prefix *KyNo*. Thus, only the key to be used is pointed to the tag. The tag finds the key indicated by DB and uses it to decrypt message Msg_3. The value *tmp0* now has nonce $Nonce_S$ generated by DB. Next, the tag generates the message Msg_4 to prove to the server that it can decrypt DB-generated messages. Next, the tag generates its nonce $Nonce_{Tj}$ and hides it in the message Msg_5 before transmitting. After receiving messages Msg_4 and Msg_5 from the tag, DB decrypts message Msg_4 and Exclusive Or (XOR) it with its own message Msg_3. The resulting value is compared with the generated $Nonce_S$ for a match to authenticate the tag.

After successful authentication, Msg_5 is decrypted and XORed with Msg_4. The calculated *tmp3* value is the tag's nonce. Next, DB calculates message Msg_6 to show that it can decrypt messages from the tag. After transmitting message Msg_6, DB calculates the session key *SesKey* by XOR'ing the received and generated nonces. After receiving Msg_6, the tag decrypts and XORs it with the previously received Msg_5. The server is authenticated, if there is a match between the obtained value and the tag's generated nonce $Nonce_{Tj}$. Now mutual authentication of the tag and DB is complete and the tag calculates the same session *SesKey*, as the final step. Hence, a new session key generation is also complete. Mutual authentication must be restarted, if the protocol is halted at any step of Phase-2.

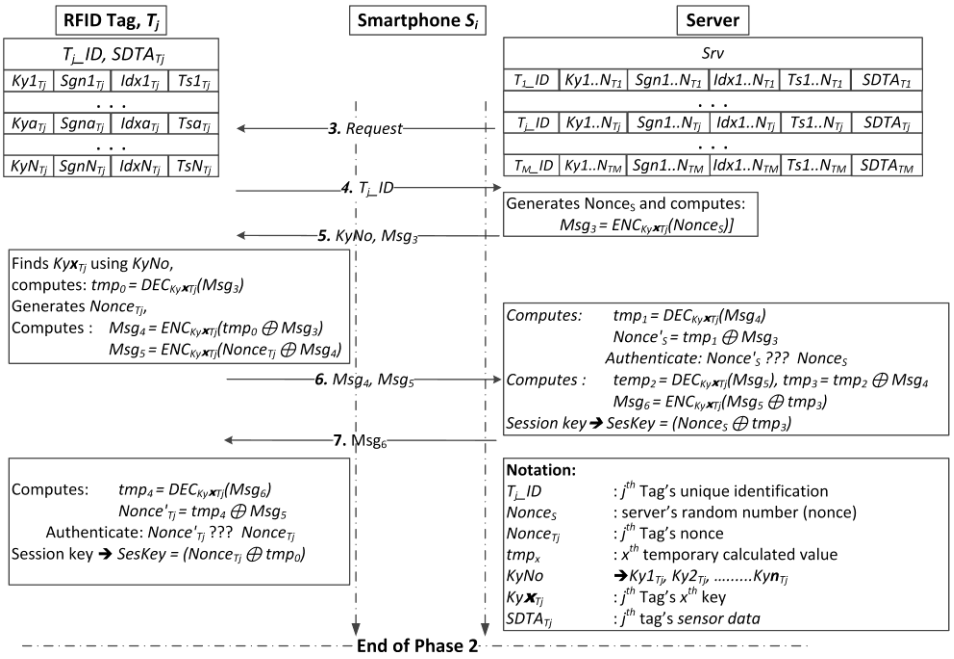


Fig. 7: Phase-2 of STRAP, "Session key generation and server-tag mutual authentication"

3.1.3 Phase 3 - Tag Signing and Tag Tracking by SDTA Checking

In the final phase (Figure 8), DB calculates the signature ($Sgni_{Tj}$) for the current session as $Sgni_{Tj} = Hash_{KyiTj}(T_j\_ID \,||\, Indi_{Tj} \,||\, Ts)$, where $Hash_{KyiTj}(X)$ indicates the encrypted hash value of X with key Kyi_{Tj} of the tag Tj. Sign || indicates the concatenation operator. DB encrypts its system time Ts, nonce of the tag, and calculated signature generated in Phase-2, and transmits it as Msg_7 to the tag. The tag decrypts message Msg_7, extracts, and checks $Nonce_{Ti}$. Then, the tag saves the signature in its memory and saves Ts to a temporary register. Then, Ts is XOR'ed with the old-time $oldTs$ found in the device's nonvolatile memory to form Msg_8. Initially, the $oldTs$ can be assumed to be zero. Next, $SDTA$ is formed, XORed with Ts and then encrypted with $SesKey$ to form Msg_9.

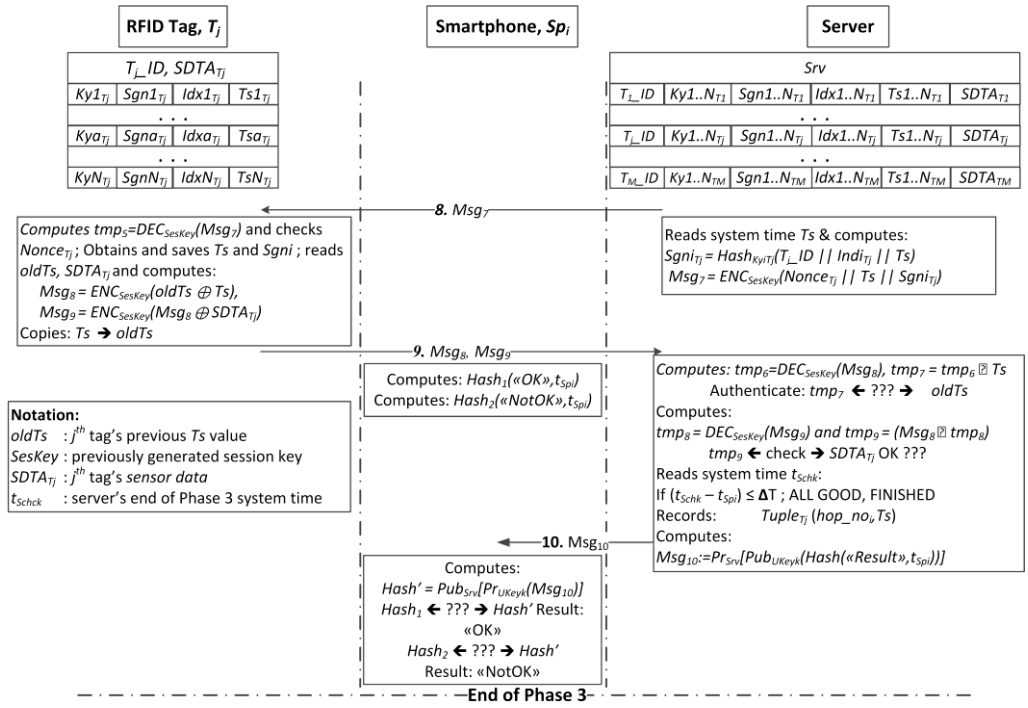


Fig. 8: Phase-3 of STRAP, "Tracking the device by checking SDTA and counting hops in the supply chain"

Before transmitting messages Msg_8 and Msg_9, the *oldTs* is overwritten by the new time *Ts* in the dedicated secure memory, thus registering the time of tag access. DB decrypts Msg_8 to verify that the tag knows the session key and has the correct *oldTs* value. After verification, the server decrypts Msg_9 to get the sensor status bits by XOR'ing Msg_8 with *temp8*. Next, *SDTA* is checked to determine if the sensors are reporting faulty play. Then, the time from the start of Phase-1 (t_{Spi}) to the end of Phase-3 (t_{Schk}) is checked to confirm that the protocol has ended within the allowed time. Then, DB increases *hop_no* and saves the current record of the tag. Finally, DB prepares the double-encrypted Msg_{10} and transmits it to the smartphone to acknowledge that STRAP has been completed successfully (Figure 8). Finally, DB drops the tag from the authentication list because a successful authentication has been completed and reported to the smartphone. Msg_{10} can only be opened by the smartphone. The aim is to compare the calculated hash with the received hash. A match notifies the operator through the smartphone that tracking of one IoT device has been completed successfully. Now DB can make the next query.

4.0 SECURITY ANALYSIS

Fending off attacks against supply chain authentication protocols is critical because if the message exchange between two peers is broken or exposed at the authentication level, the information shared in the rest of the exchange can be captured, changed, or stopped. Such a successful attack may disrupt and cause much loss for an IoT supply chain. Therefore, formal and informal security analysis of authentication protocols is an essential step in IT security, as it is also for our proposed STRAP protocol.

This Section covers the informal and formal analyses of our proposed STRAP. Formal analysis was conducted with well-accepted protocol validation tools Scyther and AVISPA [28-31]. Our attacker or adversary model is as strong as the Dolev-Yao model described in Section 2.1.2.

4.1 Informal Security Analysis of STRAP

Informal security analyses of authentication protocols depend on the authors' subjective mathematical or worded logical reasoning. Since the protocol's security is not defended by independent researchers, or by accepted protocol verifiers, informal security analyses are not accepted as a thorough security check of the proposed protocol. As a result, numerous attacks on informally defended authentication protocols exist in the literature. On the other hand, formal security analyses follow a thorough systematic method and have received the acceptance of the researchers, due to their success in verifying strongly secure protocols. Therefore, STRAP's informal security analysis against four attack types discussed in the original ReSC-1 is kept short, to give more detail on the formal analysis.

4.1.1 DoS Attack

Denial of Service (DoS) attack is intended to disrupt a system from running, without capturing any of its secrets. DoS attack is often launched by message blocking, altering, or loss. This type of attack is not a threat to STRAP for three reasons. If a DoS attack:

1. Blocks either STRAP's phase 1 or 2, the authentication is halted and repeated halts are detected.
2. Changes the key number, the authentication fails, the protocol is halted and repeated halts are detected.
3. Interferes and distorts STRAP's phase 1 or 2, the protocol times out and again repeated halts are detected.

In all of the above cases, a security officer is automatically informed about a failed DoS attack and its location, automatically by the reader/smartphone or the server. Afterward, measures can be taken to get rid of the perpetrators.

4.1.2 Full Disclosure Attack

Full disclosure attack aims at capturing all the secrets of an exchange. This attack can devastate an IoT supply chain because the successful attacker can quietly exploit the chain causing huge losses. Therefore, keeping the secrets of an authentication protocol is vital. In STRAP, only the j^{th} tag's unique identification number (T_j_ID) is passed in plain text for identifying a tag. Further analysis of STRAP messages will not disclose any secrets, as all other exchanged messages are encrypted by secret private and shared keys. Even if a rogue employee hands over the smartphone to an attacker after entering credentials, messages cannot be decrypted because the smartphone is a time-controlled mediator that cannot observe any secrets. Therefore, full disclosure attack on STRAP is not possible.

4.1.3 Lookup Table Attack

A common brute force attack becomes possible when attackers can prepare a challenge-response lookup table for an authentication protocol. Hence, the attacker can look up the response to a challenge from the look-up table and fabricate its response to fool the protocol. The ultimate goal is to get the secret message from the opposite peer. However, creating a plaintext-ciphertext pair table in STRAP is impossible because no plaintext messages are transmitted. The messages are encrypted with a new session key, every round. In addition, the transmitted messages always vary due to the changing sensor status bits and time combination. Finally, the acknowledgment of authentication to the smartphone at the end of each round is also encrypted and variable.

4.1.4 Replay Attack

Replay attack is the most trivial attack type. Prerecorded responses are replayed, hoping to complete the protocol, thus discovering as many secrets as possible. For a replay attack to be successful, replayed messages must satisfy the challenge. However, encrypted messages change in each STRAP round and are never retransmitted, as the encryption and session keys change in every new round. Therefore, replay attacks cannot succeed against STRAP.

The preliminary, informal security analysis demonstrates that STRAP is secure against the four attacks, which were successful against the ReSC protocols. The second leg of security analysis is formal analysis, which follows next.

4.2 Formal Security Analysis of STRAP

Nowadays, automated exhaustive analysis of proposed authentication protocols for verifying their security properties against known attacks is an unavoidable step in high-level academic IT security research. As a result, several security protocol verification tools have appeared, in recent years. Scyther and AVISPA are the most widely used formal security protocol analysis tools [29, 31]. Accordingly, we have analyzed STRAP with both Scyther and AVISPA tools. The tools are different in the graphical user interface, programming language, verification method, and the provided output. Since protocol analyzers are not encryption analyzers, they assume all cryptographic functions are flawless. Tracing and announcing attacks is the common characteristic of protocol verifiers. The type of the successful attack and its trace is reported for helping the repair of the weakness found.

Scyther accepts a bounded or unbounded number of message exchanges in a protocol as claims and proceeds to verify their security. The parameters used inside the messages are also tested against capture. The steps and messages are verified by “Ok” and “No attacks within bounds” comments. AVISPA also requires the description of the protocol to be verified. Then, a model checker tool analyzes the submitted authentication protocol. The result is declared as “SAFE” if no weaknesses are found, or the user is warned with a trace of the flaws. The user can use the trace to rectify the flaws in the protocol and rerun it through AVISPA for verification.

For a better understanding of Scyther and AVISPA, first the implementation of the well-known Needham-Schroeder (NSP) protocol is explained in Table 1. After the short NSP example, the analysis results of both tools for our STRAP are presented in the next sub-sections.

Table 1: Example implementation of Needham-Schroeder protocol

Alice (A) & Bob (B) Notation of Needham-Schroeder protocol (NSP)	
1. A -> B: {Na.A}_Kb 2. B -> A: {Na.Nb}_Ka 3. A -> B: {Nb}_Kb	
SPDL Code of NSP for Scyther Tool	HLPSL Code of NSP for AVISPA Tool

const pk: Function;
secret sk: Function;
inversekeys (pk,sk);
 	
// The protocol description
 	
protocol ns3(A,B)
{
role A
{
	role alice (A, B: agent, Ka, Kb: public_key,
	SND, RCV: channel (dy))
	played_by A def=
	local State : nat, Na, Nb: text
	init State := 0
	transition
	0. State = 0 /\ RCV(start) =|>
	State':= 2 /\ Na' := new() /\
	SND({Na'.A}_Kb)
	/\ secret(Na',na,{A,B})

```
                fresh na: Nonce;
                var nb: Nonce;

                send_1(A,B, {A,na}pk(B)
);
                recv_2(B,A, {na,nb}pk(A)
);
                send_3(A,B, {nb}pk(B) );
                claim_a1(A,Secret,na);
                claim_a2(A,Secret,nb);
                claim_a3(A,Niagree);
                claim_a4(A,Nisynch);
        }

        role B
        {
                var na: Nonce;
                fresh nb: Nonce;

                recv_1(A,B, {A,na}pk(B)
);
                send_2(B,A, {na,nb}pk(A)
);
                recv_3(A,B, {nb}pk(B) );
                claim_b1(B,Secret,na);
                claim_b2(B,Secret,nb);
                claim_b3(B,Niagree);
                claim_b4(B,Nisynch);
        }
}
```

```
                        /\
witness(A,B,bob_alice_na,Na')
   2.  State  = 2 /\ RCV({Na.Nb'}_Ka) =|>
            State':= 4 /\ SND({Nb'}_Kb)
                        /\
request(A,B,alice_bob_nb,Nb')
end role

role bob(A, B: agent, Ka, Kb: public_key,
                SND, RCV: channel (dy))
played_by B def=
  local State : nat, Na, Nb: text
  init State := 1
  transition
    1.  State  = 1 /\ RCV({Na'.A}_Kb) =|>
            State':= 3 /\ Nb' := new() /\
SND({Na'.Nb'}_Ka)
                        /\ secret(Nb',nb,{A,B})
                        /\
witness(B,A,alice_bob_nb,Nb')
    3.  State  = 3 /\ RCV({Nb}_Kb) =|>
            State':= 5 /\ request(B,A,bob_alice_na,Na)
end role

role session(A, B: agent, Ka, Kb: public_key) def=
  local SA, RA, SB, RB: channel (dy)
  composition
            alice(A,B,Ka,Kb,SA,RA)
     /\ bob  (A,B,Ka,Kb,SB,RB)
end role

role environment() def=
    const a, b              : agent,
              ka, kb, ki    : public_key,
              na, nb,
              alice_bob_nb,
              bob_alice_na : protocol_id
    intruder_knowledge = {a, b, ka, kb, ki, inv(ki)}
    composition
            session(a,b,ka,kb)
     /\ session(a,i,ka,ki)
     /\ session(i,b,ki,kb)
end role

goal
  secrecy_of na, nb
  authentication_on alice_bob_nb
  authentication_on bob_alice_na
end goal

environment()
```

The Needham-Schroeder Protocol (NSP) is a protocol in which two parties authenticate each other by exchanging recently generated large random numbers (nonces) that no one else can read. The simplest description of NSP is given in Alice & Bob Notation, in Table 1. Beneath the notation, the Security Protocol Description Language (SPDL) and the high-level protocol specification language (HLPSL) implementation of NSP are presented. The Scyther Tool uses SPDL and the AVISPA Tool uses HLPSL to verify protocols. The protocols in both languages define peer roles, control flow patterns, message structures, and then get tested against known adversary models.

4.2.1 Analysis Results of the Scyther Tool

Scyther is a popular, formal authentication protocol verification tool [31]. The tool has been used to verify many authentication protocols against known vulnerabilities [28, 32, 33]. The result of STRAP is shown in Figure 9(a). As a regular Scyther rule, the STRAP message exchanges were coded as 'send' and 'receive' parameters. The Sychter's 'Alive' response specifies that communicating partners are alive and their availability is ensured. The 'claim' events used for testing the security of the exchanged messages require parameters 'Secret', 'Alive', 'Weakagree', session-key reveal ('SKR'), 'Niagree' and 'Nisynch'. The 'Secret' parameter specifies that the coded STRAP parameters' (*CPUID*, *TSP*, *Tsp*, *CpuID*, *OldTs*, *NonT*, *SDTA*, *TID*, *Ts*, and *NonS*) secrecy must be verified by a "No attacks within bounds" comment. Verifiying the 'Weakagree' ensures the protocol is immune to impersonation attacks [34]. Figure 9(a) shows that the secrecy of the session key (SKR) is verified. The verification of 'Niagree' three times ensures that no messages can be injected into the exchanges. The synchronization of the STRAP steps is verified by the 'nisynch' parameter, meaning the recipient has received all messages transmitted. Every line's 'OK' declarations demonstrate that the protocol was completed successfully without attacks. Hence, the submitted STRAP is verified.

4.2.2 Analysis Results of the AVISPA Tool

AVISPA has a security protocol animator (SPAN) that offers a graphical interface for testing authentication protocols coded using the HLPSL [29-31]. The tool has four back-ends: On-the-fly model checker (OFMC), constraint-logic-based attack searcher (CL-AtSe), boolean satisfiability (SAT)-based model checker (SATMC), and tree automata-based protocol analyzer (TA4SP). The OFMC and CL-AtSe checkers are more popular than SATMC and TA4SP. Thus, we chose OFMC to verify our protocol. OFMC checks the submitted protocol and warns when it finds an attack against the protocol. Our protocol's HLPSL implementation was based on three roles: IoT device with an RFID tag, a smartphone with an RFID Reader, and a server. AVISPA has checked our model, turned STRAP into an equivalent logic equation, looked for the presence of truth values that our equivalent logic equation produces a "True" result (boolean satisfiability), and analyzed our protocol for weaknesses using the tree search technique. The analysis result of our protocol using OFMC has produced a 'SAFE' against attacks message, as shown in Figure 9 (b). The tool also confirms that our protocol had a bounded number of checker sessions. Our protocol parsing took almost no system time (0.00 seconds), and searching for a vulnerability took 0.07 seconds. Thus, STRAP is verified.

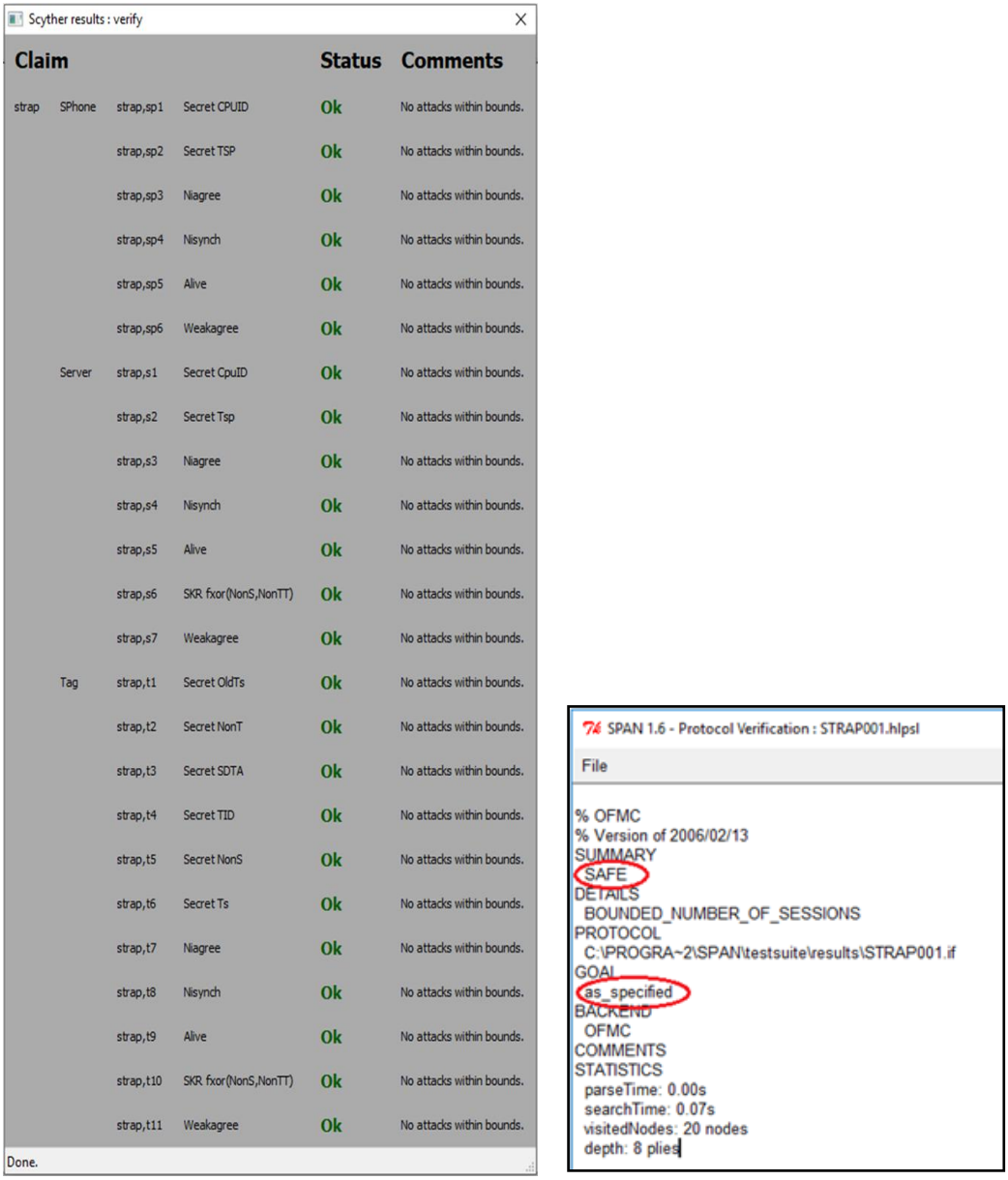
5.0 PERFORMANCE AND SECURITY COMPARISON

Our study focuses on something other than the hardware characteristics of the four protocols ReSC family. Nevertheless, STRAP has no hardware disadvantages, as it does not require any additional hardware. However, our work demonstrated that the authentication protocol vulnerabilities of the ReSC devices are fatal. The present work improves the security of IoT devices and the supply chain by proposing a stronger mutual authentication protocol.

5.1 Performance Comparison of ReSC-x and STRAP

In Table 2, the performances of the five protocols are compared. The first one is the number of steps used by the protocols. ReSC-1 finishes verifying the RFID tag in three steps with no mutual authentication. ReSC-2 and ReSC-3 also verify the RFID tag in three steps, but with mutual authentication. CDTA finishes mutual authentication in four steps, but exchanges nonces in plaintext. Our STRAP finishes authenticating the tag in eight steps but with a strong three-way mutual authentication. However, two steps are used for the essential authentication of the smartphone and its user in Phase-1, when there is no tag involvement. Authentication protocols consisting of message exchanges of less than four steps are not well-accepted mutual authentications and have known weaknesses [35]. Therefore, STRAP use of extra steps to ensure the complete security of the supply chain is justified. The disadvantage of extra steps is a few extra microseconds for a microcontroller with a megahertz clock.

The second performance parameter is the number of encryption/decryption operations carried out by the protocols. In STRAP, the RFID tag executes four encryption-decryption operations in a single round. The ReSC family performs two encryption operations at the expense of transmitting critical parameters in plain text, compromising the security of devices. The third property is the server connection established in the supply chain hops. STRAP is the only protocol that has a direct online connection between the devices and the server. The ReSC family stores confidential information on the reader, which is only transmitted when an online connection is available. The store and forward method of the ReSC family is baseless, as a wireless Internet connection is highly available in multiple types from many smartphone operators.



(a) (b)

Fig. 9: Formal analysis results of STRAP provided by Scyther and AVISPA

Table 2: Performance comparison of the five studied protocols

Authentication Protocols	ReSC-1	ReSC-2	ReSC-3	CDTA	STRAP
Number of Steps	3	3	3	4	8
Number of Encryptions	0	2	2	2	4
Online Connection	No	No	No	No	Yes
Memory Consumption	1	1	1	1	3
Power Consumption	Low	Low	Low	Low	Medium

Another performance metric is the proposed protocol's footprint (memory consumption). The higher memory consumption of STRAP is also justified, as it is needed to eliminate sharing any information with the insecure smartphone. STRAP consumes more power than the ReSC family, depending on the number of authentication steps and the number of encryptions/decryptions. This is not a problem in non-battery-operated devices, because the reader powers the RFID tag.

5.2 Security Comparison of ReSC-x and STRAP

Table 3 compares the security characteristics of the four protocols ReSC-1, ReSC-2, ReSC-3, and CDTA (ReSC family) against STRAP. ReSC family tries to provide security through hashing and symmetric encryption; however, some parameters are transmitted in plaintext. Thus, plaintext-ciphertext pairing is possible in ReSC-1, ReSC-2, ReSC-3, and CDTA. As a result, the ReSC family is vulnerable to the attacks listed in Table 3.

Table 3: Security comparison of the five studied protocols

Authentication Protocols	ReSC-1	ReSC-2	ReSC-3	CDTA	STRAP
Plain Text	Yes	Yes	Yes	Yes	No
Smart Phone User	No	No	No	No	Yes
Key Update	No	No	No	No	Yes
Dos Attack	Yes	Yes	Yes	No	No
Replay & Full Disclosure	Yes	Yes	Yes	Yes	No
Lookup Table Attack	Yes	No	No	No	No
Formal Verification	No	No	No	No	Yes

In contrast, the messages are encrypted in STRAP; therefore, plaintext-ciphertext pairing is impossible. Hence, our protocol is more secure compared to the ReSC family. The security of the intermediator smartphone and its user has crucial importance. Unfortunately, the ReSC family assumes a secure smartphone and user. However, STRAP guarantees the elimination of attacks by rogue smartphones, or malicious users. Another important security primitive is the key update. Neither of the ReSC family members provides a key update, but STRAP obscures the messages with a new session encryption key every round. An important improvement in STRAP is the requirement for a login-password duo for each supply chain hop. There are a limited number of hops, therefore a small-size password file on the server is enough to provide security. A very important disadvantage of the ReSC family is the need for formal protocol verification. Meanwhile, STRAP has been verified by not only one but two well-accepted protocol verifiers. Therefore, STRAP is the only strong mutual authentication protocol that has no vulnerability to known attacks and fits its primary security goal. It is commonly accepted that security experts always consider the performance/security ratio when choosing an authentication protocol. STRAP proves to have a considerably superior performance/security than the ReSC family.

5.3 Performance Comparison of RFID-Enabled Supply Chain Solutions

Our proposed supply chain solution is compared with some other RFID-enabled supply chain proposals, in Table 4. Related work on counterfeit-troubled IC supply chains and state-of-the-art blockchain-supported proposals have been included in the comparison. The comparison is necessary to shed light on the characteristic performances of the proposals. It is obvious from the comparison that our proposal does not contain a hardware alternative; but instead, it focuses on the security of the most important component, the authentication protocol architecture of the whole supply chain. STRAP is the only proposal with a fully online, operator-reader bonded and formally verified authentication protocol, with no provable attacks. Our supply chain setup is fully compliant with the Dolev-Yao attack model.

Table 4: Performance comparison of RFID enabled supply chain proposals

Supply Chain Proposals	Supply Chain	Hardware Proposed	Online Tracking	User-Reader Bonding	Dolev-Yao Model	Attack Announced	Formal Security Analysis
ReSC Family [10-13]	Yes	Yes	No	No	No	Yes	No
SHIELD [36]	Yes	Yes	No	No	No	Yes	No
CNTR-SHIELD [37]	Yes	Yes	No	No	No	Yes	No
LBRAPS [38]	Yes	No	Yes	No	No	No	Yes
STRAP	Yes	No	Yes	Yes	Yes	No	Yes

In contrast, none of the other supply chain solutions have DB-approved operator-reader bonding. Attacks have been announced on the [10-13, 36, 37] protocols, which lack formal verification. The offline setup and the weak Dolev-Yao attack models of these six solutions prove to be very destructive to the security of the whole supply chain. LBRAPS is another work with a fully online, formally verified authentication protocol, with no announced attacks yet. The operator-reader bonding is replaced by blockchain support. However, the reader keys shared at the chain hops, the XOR, the bitwise rotation and hashing operations used in the protocol (instead of yet-to-be-broken AES encryption) constitute serious weaknesses. The reader is invited to read our previous works on rotational cryptanalysis and look-up table attacks. Unfortunately, weaknesses are not blockchain-supported supply chain solutions' only problems. The disadvantages of blockchain inclusion in supply chain security are high cost, low market acceptability, lack of standards and scalability, privacy concerns due to third-party involvement, and interoperability problems [39].

6.0 CONCLUSION

This article presents an online supply chain hop-tracking procedure supported by a novel RFID mutual authentication protocol called STRAP. In present-day supply chain environments, STRAP is readily available to replace the vulnerable lightweight protocols proposed in the works [10-13]. Concerning the other compared solutions, STRAP has the following merits in securing the supply chains:

1. Strong security provided by STRAP is formally verified. The cost of strong security is slightly higher power consumption-communication speed-computation power, which are inherently available in all present-day smartphones.
2. STRAP defines and matches a legitimate smartphone with a legitimate user; the intermediary smartphone's role is just a message repeater between the database server and RFID tags.
3. No messages, secrets, or nonces are transmitted in plaintext. The secrets of STRAP's RFID-enabled devices are never revealed.
4. STRAP does not have the disadvantages of blockchain-supported solutions.

REFERENCES

[1] L. S. Vailshery, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030", Accessed: Dec. 22, 2022. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/#:~:text=Number%20of%20IoT%20connected%20devices,2021%2C%20with%20forecasts%20to%202030&text=The%20number%20of%20Internet%20of,billion%20IoT%20devices%20in%202030.

[2] V. Manjula and R. Thalapathi Rajasekaran, "Security Vulnerabilities in Traditional Wireless Sensor Networks by an Intern in IoT, Blockchain Technology for Data Sharing in IoT", *in SL. Peng, S. Pal, L. Huang, (eds) Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*. *Intelligent Systems Reference Library*, vol 174. Springer, Cham, 2020.

[3] N. Alhirabi, O. Rana, and C. Perera, "Security and Privacy Requirements for the Internet of Things: A Survey", *ACM Transactions on Internet of Things*, vol. 2(1), 2021.

[4] C. Dong, Y. Xu, X. Liu, F. Zhang, G. He, and Y. Chen, "Hardware Trojans in Chips: A Survey for Detection and Prevention", *Sensors*, vol. 20(18), 5165, 2020.

[5] S. Sidhu, BJ Mohd, and T. Hayajneh, "Hardware Security in IoT Devices with Emphasis on Hardware Trojans", *Journal of Sensor and Actuator Networks*, vol. 8(3), 42, 2019.

[6] J. Vosatka et al., "Confidence Modeling and Tracking of Recycled Integrated Circuits, Enabled by Blockchain", *2020 IEEE Research and Applications of Photonics in Defense Conference (RAPID)*, Miramar Beach, FL, USA, pp. 1-3, 2020.

[7] V. Hassija, V. Chamola, V. Gupta, S. Jain, and N. Guizani, "A Survey on Supply Chain Security: Application Areas, Security Threats, and Solution Architectures", *in IEEE Internet of Things Journal*, vol. 8(8), pp. 6222-6246, 2021.

[8] L. Azriel, R. Ginosar, and A. Mendelson, " SoK: An Overview of Algorithmic Methods in IC Reverse

Engineering", *ASHES'19: Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, pp. 65–74, 2019.

[9] H. Türksönmez, and M. H. Özcanhan, "A Survey on Integrated Circuit Trojans", Computer Engineering and Intelligent Systems, vol. 12(2), 2021.

[10] K. Yang, D. Forte, and M. M. Tehranipoor, "Protecting endpoint devices in IoT supply chain", 2015 IEEE/ACM International Conference on Computer-Aided Design (ICC_AD), Austin, TX, USA, pp. 351-356, 2015.

[11] K. Yang, D. Forte, and M. M. Tehranipoor, "CDTA: A Comprehensive Solution for Counterfeit Detection, Traceability, and Authentication in the IoT Supply Chain", *ACM Transactions on Design Automation of Electronic Systems*, vol. 22(3), 2017.

[12] K. Yang, D. Forte, and M. M. Tehranipoor, "ReSC: An RFID-Enabled Solution for Defending IoT Supply Chain", *ACM Transactions on Design Automation of Electronic Systems*, vol. 23(3), 2018.

[13] K. Yang, D. Forte, and M. M. Tehranipoor, "ReSC: RFID-Enabled Supply Chain Management and Traceability for Network Devices", *in Radio Frequency Identification, RFIDSec 2015, Lecture Notes in Computer Science*, vol. 9440, Springer, Cham, 2015.

[14] E. Oriero and S. R. Hasan, "Survey on recent counterfeit IC detection techniques and future research directions", *ScienceDirect*, vol. 66, pp. 135-152, 2019.

[15] B. A. Alzahrani and A. Irshad, "An Improved IoT/RFID-Enabled Object Tracking and Authentication Scheme for Smart Logistics", *Wireless Pers Commun*, vol. 129, pp. 399–422, 2023.

[16] H. Karim and D. B. Rawat, "TollsOnly Please—Homomorphic Encryption for Toll Transponder Privacy in Internet of Vehicles*", in IEEE Internet of Things Journal*, vol. 9(4), pp. 2627-2636, 2022.

[17] D. Dolev, and A. C. Yao, "On the security of public key protocols", *IEEE Transactions on Information Theory*, vol. 29, pp.198-208, 1983.

[18] C. Jin, and M. Van Dijk, "Secure and efficient initialization and authentication protocols for SHIELD", *IEEE Transactions on Dependable and Secure Computing*, vol. 16, p. 156–173, 2019.

[19] F. Farha, H. Ning, K. Ali, L. Chen, and C. Nugent, "SRAM-PUF-Based Entities Authentication Scheme for Resource-Constrained IoT Devices," *in IEEE Internet of Things Journal*, vol. 8(7), pp. 5904-5913, 2021.

[20] L. Kusters and F. M. J. Willems, "Secret-Key Capacity Regions for Multiple Enrollments With an SRAM-PUF", *in IEEE Transactions on Information Forensics and Security*, vol. 14(9), pp. 2276-2287, 2019.

[21] C. Yehoshuva, R. Raja Adhithan, and N. Nalla Anandakumar, "A Survey of Security Attacks on Silicon Based Weak PUF Architectures", *in S.M. Thampi, , G. Wang, , D.B. Rawat, R. Ko, and CI. Fan, (eds) Security in Computing and Communications. SSCC 2020. Communications in Computer and Information Science*, vol. 1364, Springer, Singapore, 2021.

[22] U. Guin, D. Forte, and M. M. Tehranipoor, "Design of Accurate Low-Cost On-Chip Structures for Protecting Integrated Circuits Against Recycling", *in IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24(4), pp. 1233-1246, 2016.

[23] U. Guin, X. Zhang, D. Forte, and M. M. Tehranipoor, "Low-cost on-chip structures for combating die and IC recycling", *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1-6, 2014.

[24] G. E. Suh, and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation", *44th ACM/IEEE Design Automation Conference*, pp. 9-14, 2007.

[25] X. Zhang, and M. M. Tehranipoor, "Design of On-Chip Lightweight Sensors for Effective Detection of

Recycled ICs", *in IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22(5), pp. 1016-1029, 2014.

[26] X. Zhang, N. Tuzzio, and M. M. Tehranipoor, "Identification of recovered ICs using fingerprints from a light-weight on-chip sensor", *DAC Design Automation Conference 2012*, pp. 703-708, 2012.

[27] NXP Corporation, "MIFARE DESFire EV3 contactless multi-application IC short data sheet", Accessed: May 01, 2023. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MF3DHx3_SDS.pdf

[28] M. Adeli, N. Bagheri, and H. R. Meimani, "On the designing a secure biometric-based remote patient authentication scheme for mobile healthcare environments", *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 3075–3089, 2021.

[29] A. Armando *et al.*, "The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications", *in K. Etessami, S. K. Rajamani (eds) Computer Aided Verification. CAV 2005. Lecture Notes in Computer Science*, vol. 3576. Springer, Berlin, Heidelberg, 2005.

[30] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, *Handbook of Model Checking*, Springer, 2018.

[31] C. J. F. Cremers, "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols", *in A. Gupta, S. Malik (eds) Computer Aided Verification. CAV 2008. Lecture Notes in Computer Science*, vol. 5123. Springer, Berlin, Heidelberg, 2008.

[32] M. Azrour, J. Mabrouki, A. Guezzaz, and Y. Farhaoui, "New enhanced authentication protocol for Internet of Things", *in Big Data Mining and Analytics*, vol. 4(1), pp. 1-9, 2021.

[33] M. Safkhani, C. Camara, P. P. Lopez, and N. Bagheri, "RSEAP2: An enhanced version of RSEAP, an RFID based authentication protocol for vehicular cloud computing", *Vehicular Communications*, vol. 28, 2021.

[34] M. Nikooghadam, and H. Amintoosi, "An improved secure authentication and key agreement scheme for healthcare applications", *in 2020 25th International Computer Conference, Computer Society of Iran (CSICC_)*, pp. 1-7, 2020.

[35] G. Dalkiliç, M. H. Özcanhan, and H. Ş. Çakir, "Increasing key space at little extra cost in RFID authentications", *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 22, no. 1, 2013.

[36] S. Leef, "Supply Chain Hardware Integrity for Electronics Defense (SHIELD)," Defense Advanced Research Projects Agency (DARPA) Microsystems Technology Office, 2018.

[37] C. Jin, and M. van Dijk, "Secure and efficient initialization and authentication protocols for SHIELD," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, pp. 156–173, 2019.

[38] S. Jangirala, A. K. Das, and A. V. Vasilakos, "Designing Secure Lightweight Blockchain-Enabled RFID-Based Authentication Protocol for Supply Chains in 5G Mobile Edge Computing Environment," *in IEEE Transactions on Industrial Informatics*, vol. 16(11), pp. 7081-7093, 2020.

[39] S. Jabbar, H. Lloyd, M. Hammoudeh, B. Adebisi, and U. Raza, "Blockchain-enabled supply chain: analysis, challenges, and future directions.", *Multimedia Systems*, vol 27, pp. 787-806, 2021.