Double Verification Protocol via Secret Sharing for Low-Cost RFID Tags

Y. Liu[a], M. F. Ezerman[b,*], H. Wang[b]

[a]*College of Computer Science and Technology, Jiangsu Normal University, Xuzhou, 221116, China.*
[b]*School of Physical and Mathematical Sciences, Nanyang Technological University, 21 Nanyang Link, Singapore.*

Abstract

RFID tags have become ubiquitous and cheaper to implement. It is often imperative to design ultralightweight authentication protocols for such tags. Many existing protocols still rely on triangular functions, which have been shown to have security and privacy vulnerabilities. This work proposes UMAPSS, an ultralightweight mutual-authentication protocol based on Shamir's $(2,n)$ secret sharing. It includes mechanisms for double verification, session control, mutual authentication, and dynamic update to enhance security and provide a robust privacy protection. The protocol relies only on two simple bitwise operations, namely addition modulo 2^m and a circular shift $\mathrm{Rot}(x,y)$, on the tag's end. It avoids other, unbalanced, triangular operations.

A security analysis shows that the protocol has excellent privacy properties while offering a robust defense against a broad range of typical attacks. It satisfies common security and the low-cost requirements for RFID tags. It is competitive against existing protocol, scoring favourably in terms of computational cost, storage requirement, and communication overhead.

Keywords: RFID, low-cost, mutual authentication, secret sharing, ultralightweight.

1. Introduction

Radio Frequency Identification (RFID) brought automatic object identification by electromagnetic wave into sensor technology, requiring no physical contact, which was revolutionary. As costs steadily drop, RFID systems are increasingly deployed in varied environments, raising numerous security and privacy concerns. Many works have pointed out that RFID is vulnerable to practical malicious attacks (see [1] and [2]) and security threats (see [3] and [4]). These include eavesdropping, message interception and modification, blocking, jamming, counterfeiting, spoofing, traffic analysis, man in the middle (MITM), traceability, and desynchronization attacks. Effective authentication protocols to improve robustness, reliability, and security against major attacks, both passive and active, are crucial.

Based on memory type, power consumption, and price, RFID tags are either high-cost or low-cost. In 2007, Chien proposed a tag classification based on computational cost and supported on-tag operations [5]. High-cost tags fall into either *full-fledged* or *simple* class. The

[*]Corresponding author
Email addresses: lyl1980115@163.com, liuyali@jsnu.edu.cn (Y. Liu), fredezerman@ntu.edu.sg (M. F. Ezerman), HXWang@ntu.edu.sg (H. Wang)

low-cost tags are either in the *lightweight* or *ultralightweight* class. Low-cost RFID tags have between 5000 to 10000 logic gates with only 250 to 3000 of them to use for security functions. It remains very challenging to deploy conventional cryptographic protocols on tags, especially the ultralightweight ones. Their typical authentication protocol uses only simple bitwise operations such as XOR, OR, AND, and rotation.

This paper incorporates threshold secret sharing into ultralightweight authentication protocols (UAPs) for RFID tags. We start by evaluating existing UAPs to establish the security requirements and identify common vulnerabilities. We then propose a new protocol that incorporates Shamir's $(2,n)$ secret sharing, henceforth $(2,n)$ SS. There are several advantages. The $(2,n)$ SS scheme is cheap to implement. It boosts the security of the overall system.

Measured against previous proposals, the followings are our contributions.

1. Our proposal provides a strong impersonation resistance. Its double verification mechanism demands key verification and secret recovery verification.

2. To counter threats that exploit timeout, we devise an overtime-exit function using a session control mechanism to properly regulate the round-trip time of every challenge-response cycle.

3. Unlinkability and forward security of the authentication sessions are enhanced by a dynamic update mechanism. The mechanism prevents desynchronization among the tags, the readers, and the trusted database (TD).

4. A double-entity-round mutual authentication mechanism is added to maintain authenticity and integrity during transmission and updating. The protocol thwarts malicious attacks that modify or block the exchanged messages and, hence, differs from the proposals in References [5] and [6] that resist only passive attacks.

5. The protocol meets the requirement of being ultralightweight, requiring only simple bitwise operations at the tag's end.

The rest of this paper is organized as follows. Section 2 provides some preliminaries. Section 3 reviews related works. The core discussion of our protocol is in Section 4. Sections 5 and 6 provide, respectively, the security and attack model analysis. A formal security analysis based on the CasperFDR model can be found in Section 7. Section 8 gives the performance evaluation and some conclusions will be given in Section 9.

2. Preliminaries

The first part of this section presents a typical deployment and assumptions of an RFID system. The second part recalls Shamir's (t,n) secret sharing scheme.

A typical deployment involves three types of legitimate entities: the *tags*, the *readers*, and a *verifier* (or a *backend database*). The low-cost tags are passive with very limited capabilities. They do not maintain clocks. Before any tag is attached to an object, its unique static identifier ID and an index-pseudonym IDS are written in its ROM and EEPROM, respectively, together with several secret values for authentication. The reader establishes some communication channel(s) with the tags to be able to query them and to keep a record of proofs for each session. The adversary should not be able to manipulate the record. The backend database TD is the only trusted entity that may share some secret information, *e.g.*, cryptographic keys, with the readers.

The communication channel between the reader and the backend database is secure and has high performance. The channels between a reader and the tags are wireless and, hence, insecure. Communications between the reader and the tags are initiated by the reader. Once the required messages have been exchanged, the reader and the tags declare the protocol complete.

An RFID authentication protocol should comply with essential security and privacy requirements (see, *e.g.*, [7]) to ensure data confidentiality, tag anonymity, forward security, untraceability, and robustness against malicious attacks. Such protocols are mainly concerned with security issues at the protocol layer and not with physical or link layer issues.

A cryptography primitive that we would like to incorporate into our protocol is secret sharing. Consider the scenario where n participants want to share a secret among themselves. They fix a number $1 < t \leq n$ and require that any t of them can recover the secret while any $t-1$ of them gain no additional information whatsoever about the secret. Schemes to accomplish such secret sharing were independently introduced by Blakley in [8] and Shamir in [9] as a solution for safeguarding secret keys. This work uses (t,n)-SS to denote Shamir's scheme where n is the number of participants and t is the threshold. The scheme relies of the fact that a polynomial of degree $t-1$ is *uniquely* determined by t values. A linear polynomial, for example, is determined by two points. A quadratic polynomial by three points, and so on. We briefly revisit the scheme before using the $(2,n)$-SS for the rest of the paper.

The (t,n)-SS scheme has two main algorithms, namely the share generation algorithm *GenS* and the secret reconstruction algorithm *RecS*. Let p be a prime larger than n and let $\mathbb{F}_p$ be the field of p elements. Let the secret be represented by an element $a_0 \in \mathbb{F}_p$.

1. *GenS*: A trusted dealer selects coefficients a_i for $1 \leq i \leq t-1$ randomly from a uniform distribution over the integers in $\mathbb{F}_p$ with $a_{t-1} \neq 0$ to form

$$f(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1} \in \mathbb{F}_p[x].$$

   Next, for $1 \leq j \leq n$, the dealer computes distinct shares $s_j := f(x_j)$ from n nonzero pairwise distinct values x_j in $\mathbb{F}_p$ and distributes s_j to shareholder P_j secretly.

2. *RecS*: Given any t shares, the secrets a_0 can be reconstructed using the Lagrange interpolation formula since there is a unique polynomial $f(x)$ of degree $t-1$ such that $f(x_j) = s_j$ for $1 \leq j \leq n$. The formula, plus its generalization to the multivariable version, is given, *e.g.*, in [10, Thm. 1.71]. We reproduce it here for convenience.

   Theorem 1. *(Lagrange Interpolation Formula) Let $x_0, x_1, \ldots, x_t$ be $t+1$* **distinct** *elements of $\mathbb{F}_p$ and let $s_0, s_1, \ldots, s_t$ be $t+1$ arbitrary elements of $\mathbb{F}_p$. Then the unique polynomial $f(x) \in \mathbb{F}_p[x]$ of degree $\leq t$ such that $f(x_j) = s_j$ for $0 \leq j \leq t$ is given by*

$$f(x) = \sum_{j=0}^{t} s_j \prod_{k=0;k\neq j}^{t} (x_j - x_k)^{-1}(x - x_j).$$

   If an attacker can gather t shares, then it can recover the secret. If t participants become corrupted, then they can combine their shares to reconstruct the secret.

3. Related Works

Research in RFID protocols is very active. Various tools have been combined into protocols. Many with little variations from the ones rather immediately prior. Several leading researchers have even lamented that little, if any, real progress has been made in the last few years. The rather chaotic general scenery, however, seems unavoidable as the deployments of RFID systems widen and vary rapidly.

In this work we focus on ultralightweight authentication protocols. Before going there, we briefly mention several main strands which up to this point are considered not ultralightweight enough yet. Depending on advances in hardware engineering, they may well become viable options in the future.

Due to their attractive cryptographic strength, lightweight version of cryptographic primitives such as AES and hash function are natural candidates. There has been a lot of effort dedicated to making hash functions lightweight enough for on-tag implementation to make authentication anonymous. A prominent example is the PHOTON family of hash functions proposed in [11]. A remarkable recent contribution, in fact, managed to do away with such a requirement. Chen *et al.* in [12] listed down various protocols that claim to ensure anonymity in authentication. All but their own required on-tag hash functionalities.

Proposals based on the NTRU cryptosystem have been put forward. Already in 2008, Atici *et al.* discussed several low-cost implementations of NTRU for pervasive security, with deployments in RFID mentioned as a key application in [13]. While this appears to be likely the case, further analysis remains to be done to come up with more concrete protocols, especially for authentication purposes. Several works followed in this line of research. Very recently, Hwang and Lee proposed a lightweight NTRU-based mutual authentication scheme for RFID deployed in medical devices [14].

A related source of hard problems for cryptography primitives is error-correcting codes. Code-based systems provide fast and secure encryption and decryption schemes but suffer from large public key sizes. Chikouche *et al.* in [15] presented a survey on RFID authentication protocols that rely on variants of the McEliece system. The survey identified the protocols' common weaknesses and gave their performance evaluation.

Cryptography primitives based on elliptic curves have also been explored. Some works focused on their lightweight implementations while some others explored aspects of RFID deployments that can benefits from incorporating such primitives. Some recent works utilizing elliptic curves are [16] and [17]. The latter was subsequently shown to be insecure in [18].

Forming another direction are the human based protocols, originally introduced by Hopper and Blum in [19]. Many improved variants have been proposed since. A recent improvement with an LPN (Learning Parity with Noise) flavour is the Tree-LSHB+ protocol in [20].

Secret sharing schemes have been used rather extensively in RFID key management in supply chains to transfer control and or ownership of the devices and relevant records. Several recent works on this topic are [21, 22, 23]. Shared secret among tags, using a particular structure such as a tree or a grid had also been proposed. Most of them have traceability issues. See [24, Sect. 3] for a detailed treatment. Secret sharing schemes had been previously suggested for authentication protocol in [25]. The proposal, however, was unconvincing. First, the Lagrange interpolation to recover the secret is performed by the tag, which is not feasible since doing so involves multiplication by inverses in the underlying finite field. Second, the recovered exchanged random numbers between the reader and the tag are not used directly for authentication. The proposal's unfeasibility in ultralightweight setup is further shown in the generation of random number at the tag's end and in the usage of hash function on the payloads.

From hereon we limit our attention to ultralightweight protocols. A series of ultralightweight schemes had already been proposed for low-cost RFID systems since the late 1990s, initially with little attention paid to their security. The work by Vajda [26] was the first to propose the use of lightweight cryptography. Juels then introduced the notion of minimalist cryptography in [27]. Peris-Lopez *et al.* proposed the UMAP family consisting of LMAP in [28], EMAP in [29], and M^{2}AP in [30]. Their attractive ultralightweight design spurred further interest, directly inspiring several other protocols, *e.g.*, [31] to [32]. The family, unfortunately, was shown to be vulnerable against an active attacker.

Chien introduced SASI, a typical ultralightweight protocol that provided strong authentication and integrity, in [5]. It was later demonstrated in [33] that a complete secret data disclosure could be obtained. Despite its multiple vulnerabilities, SASI reflected a turning point in design, leading to Gossamer [34] and the scheme of Lee *et al.* in [35].

All of the above protocols assumed that *ultralightweight* tags could only compute simple bitwise operations XOR, AND, and OR. This posed a hefty penalty in terms of security due to their limited capabilities. AND, XOR, OR, and addition modulo 2^m are said to be *triangular functions* or *T-functions* [36]. They remained vulnerable against many types of attacks, *e.g.*, the tango attack since OR and AND produce imbalanced outputs.

Numerous improvements have since been proposed. Many claimed to have properly fixed general design concerns or to have completely addressed specific known shortfalls. Practically all of them have subsequently shown to be inadequate, either in general or in some specific aspects. One can refer to Piramuthu's survey [37, Table 1] for a list of vulnerabilities afflicting a number of prominent protocols up to 2010. A more detailed list can be found in [38, Table 1] for most of the protocols proposed up to early 2014.

Our aim in this work is to eliminate known vulnerabilities in UAPs, *e.g.*, by removing triangular functions. To further improve on the security we incorporate a secret sharing scheme for mutual authentication.

4. Our Protocol UMAPSS

Our design objectives are to enhance the robustness, improve the efficiency, and control the session-cycle timing while keeping the computation and storage costs feasible. We propose UMAPSS, an ultralightweight mutual-authentication protocol equipped with a $(2,n)$-SS. Its double verification mechanism reduces the success probability of malicious attacks without adding computational burden at both the reader's and the tags' ends. The protocol addresses the overtime-exit issue using a session control mechanism. It regulates the round-trip time of every challenge-response cycle and guarantees session unlinkability by adding forward security. It remains lightweight because the tags operations consist of addition modulo 2^{96}, denoted simply by $+$ since the risk of confusion is minimal, and the circular shift $\mathrm{Rot}(x,y)$. Table 1 summarizes the notations for entities and operations.

Here are the protocol's specific design considerations.

1. Each tag stores secret values *ID*, *IDS*, and $K = K_1|K_2|K_3$ in its memory. K is also stored by TD. The *IDS* serves as a search index to allocate all information related to a particular tag stored in TD. The key K encrypts the data being transferred. Since each tag has its own secret key, if the reader encrypts a message using K, then only the tag that has K can decrypt the message.

2. To prevent traceability, the local values *IDS* and K are updated after each successful authentication. This gives UMAPSS forward security and unlinkability. Additionally, the tags are required to store the old and (potential) new values of *IDS* and K to resist desynchronization.

Table 1: Notations used in UMAPSS

R, T, and TTP	Reader, Tag, and Trusted Third Party
TD	Trusted Database containing IDS, K, S, $f(x)$ between T and R
ID	Unique static identification information of T
IDS, IDS^{old}, IDS^{new}	Current, previous, and next index-pseudonym of T
K	T's secret key; shared between TD and T
K_1, K_2, K_3	Subkeys of K in current authentication session
$K_1^{old}, K_2^{old}, K_3^{old}$	Subkeys of K in previous authentication session
$K_1^{new}, K_2^{new}, K_3^{new}$	Subkeys of K in next authentication session
S	Secret based on $(2,n)$-SS for current authentication
m	Number of points on the curve defined by $f(x)$
(x_i, y_i)	A point on $f(x)$ based on $(2,n)$-SS for $1 \leq i \leq m$
$PRNG$	Pseudorandom number generator, *e.g.*, Warbler [39]
μ	An output of the $PRNG$
ΔT	Time of challenge-response session between R and T
P	Mutual-authentication session for R and T
$+$	Converts two binary strings into integers, add them modulo 2^{96}, then converts the resulting integer back to binary.
$\mathrm{Rot}(x,y)$	Left-shift on x by y (mod 96) positions

3. To avoid using triangular functions, we implement only $+$ and $\mathrm{Rot}(x,y)$ on tags. To improve robustness while maintaining optimal security, $\mathrm{Rot}(x,y)$ in our protocol performs a circular left shift on the value of x by y (mod 96) positions. This circumvents the known weakness of $\mathrm{Rot}(x, \mathrm{wt}(y))$ with $\mathrm{wt}(y)$ denoting the Hamming weight of y.

4. To reduce computational cost, the random number generation via $PRNG()$ is done at the reader's end. The tags only use the numbers and performs $+$ and $\mathrm{Rot}(x,y)$ to create fresh communication messages. Our protocol resists both passive and active attacks that include message modification, insertion, and or blocking.

5. The length of any message is 96 bits, compatible with commonly deployed encoding schemes, *e.g.*, GTIN and GRAI, in EPCGlobal.

In short, we want a protocol with adequate security level that can be realistically deployed in ultralightweight RFID tags. UMAPSS incorporates a $(2,n)$-SS to run a mutual-authentication session between R and T. Here R authenticates T by recovering the original secret S after T sends its legitimate, *i.e.*, not a forged, share. The reader imposes a time limit ΔT for each round of challenge-response session. UMAPSS halts if ΔT is exceeded.

4.1. Four Stages

The protocol runs in four stages, namely an initial setup, the tag identification, the mutual authentication, and the updating stages. Interactions between a particular reader R_j and a tag T_r are in chronological ordering. Figure 1 summarizes the stages.

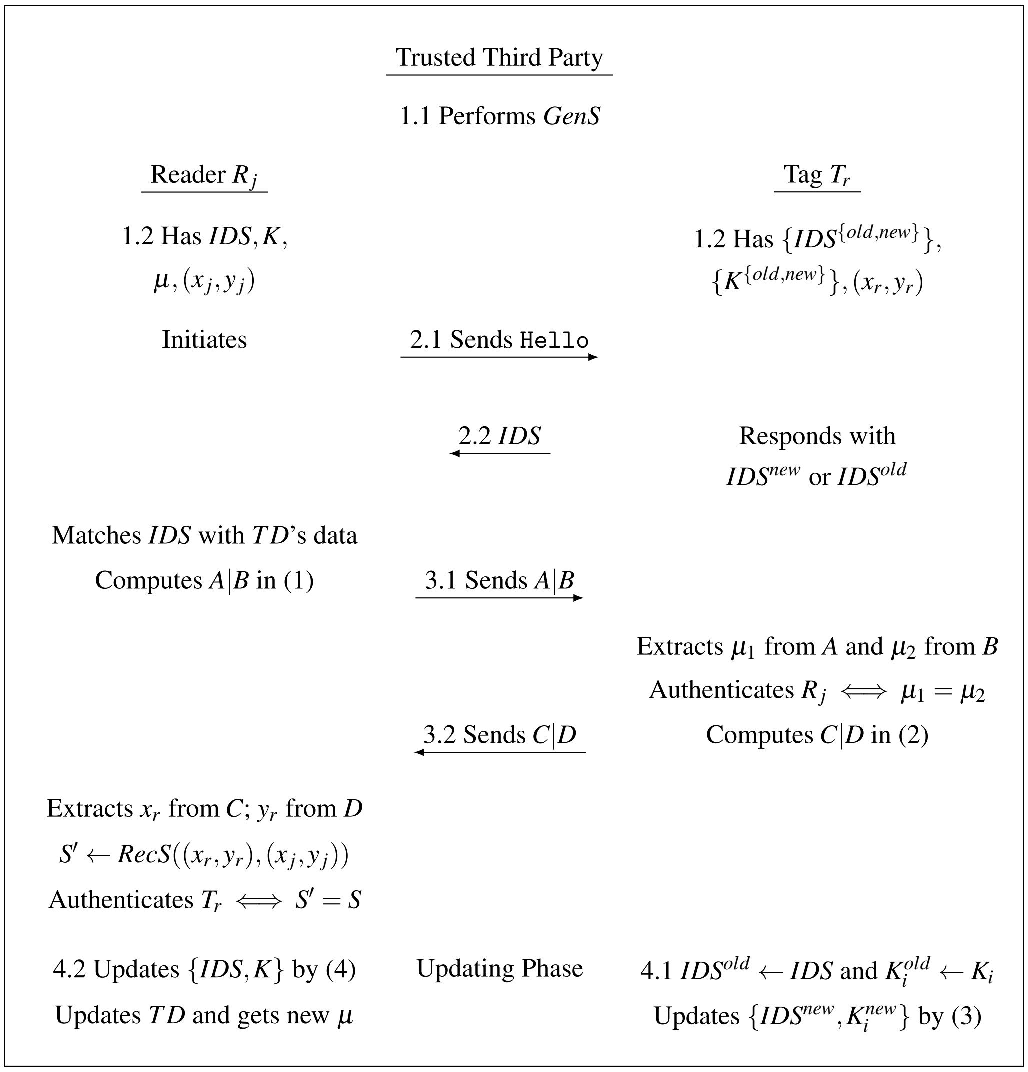
Trusted Third Party
1.1 Performs $GenS$
Reader R_j
Tag T_r
1.2 Has IDS, K, $\mu, (x_j, y_j)$
1.2 Has $\{IDS^{\{old,new\}}\}$, $\{K^{\{old,new\}}\}, (x_r, y_r)$
Initiates
2.1 Sends `Hello`
2.2 IDS
Responds with IDS^{new} or IDS^{old}
Matches IDS with TD's data
Computes $A|B$ in (1)
3.1 Sends $A|B$
Extracts μ_1 from A and μ_2 from B
Authenticates $R_j \iff \mu_1 = \mu_2$
3.2 Sends $C|D$
Computes $C|D$ in (2)
Extracts x_r from C; y_r from D
$S' \leftarrow RecS((x_r, y_r), (x_j, y_j))$
Authenticates $T_r \iff S' = S$
4.2 Updates $\{IDS, K\}$ by (4)
Updates TD and gets new μ
Updating Phase
4.1 $IDS^{old} \leftarrow IDS$ and $K_i^{old} \leftarrow K_i$
Updates $\{IDS^{new}, K_i^{new}\}$ by (3)

Figure 1: UMAPSS

The Initial Setup Phase

The TTP selects a $PRNG$, say $g : \{0,1\}^k \mapsto \{0,1\}^{2k}$ for some security parameter k. It then uses an output S of g as the secret in the current authentication session and constructs $f(x) = a_1 x + S \in \mathbb{F}_p[x]$, keeping $S = f(0)$ and a_1 secret. The TTP then generates m points $(x_i, y_i := f(x_i))$ for $1 \leq i \leq m$. It keeps ℓ of the points in ℓ valid readers R_j for $1 \leq j \leq \ell < m$ and keeps the remaining $m - \ell$ points in valid tags T_r for $\ell + 1 \leq r \leq m$. The reader R_j keeps (x_j, y_j) secret and, similarly, the tag T_r keeps (x_r, y_r) secret. Let $N := \{R_1, \ldots, R_\ell, T_1, \ldots, T_{m-\ell}\}$.

Each R_j stores ΔT to control a cycle of challenge-response session. We intend for Δ_T to be decided based on the implementation requirements. Based on the simulations done prior to deployment, keeping the usage and the assumptions on the attacker's capabilities, the system designer can set and then adjust this value to reach a desired trade-off between utility and security.

TD stores IDS, K, S, and $f(x)$. T_r stores IDS and K while R_j stores IDS, K, and S. Note that K, S, and $f(x)$ are always kept secret.

The Tag Identification Phase

To trigger mutual authentication, the reader has to identify the tag. R_j initiates a new session by sending a `hello` message to T_r. The tag transmits IDS^{new} in response. R_j uses IDS^{new} as an index to search for a matching entry in TD. Only an authorized reader is able to search TD and access T_r's secret key $K = K_1|K_2|K_3$, which is required in the next authentication stages.

If R_j finds a match, it proceeds to the mutual authentication phase using K_i^{new} for $1 \leq i \leq 3$. Otherwise, it notifies T_r. The tag subsequently replies by sending IDS^{old}. The identification is retried, using IDS^{old} instead of IDS^{new}. Note that T_r backscatters IDS^{old} upon request. If the identification is successfully done, then R_j uses K_i^{old} in the mutual authentication phase. If IDS is not in TD, the session is terminated.

The Mutual Authentication Phase

R_j uses IDS, K, and an output μ of g to compute

$$A := \mathrm{Rot}(\mathrm{Rot}(IDS + \mu, K_1), K_2) + K_3 \text{ and } B := \mathrm{Rot}(\mathrm{Rot}(IDS + K_1, \mu), K_3) + K_2. \tag{1}$$

Subsequently, A is used to send μ with a mask to T_r and B is used to authenticate R_j and the integrity of the messages. Effectively, R_j sends $A|B$ to T_r as a random challenge. T_r uses IDS and K to extract $\mu_1 \triangleq \mu$ obtained from A and $\mu_2 \triangleq \mu$ obtained from B. If $\mu_1 = \mu_2$, then R_j is authenticated. T_r's ability to retrieve the correct μ ensures R_j's legitimacy. Otherwise, T_r aborts the session since the received messages may have been modified by an attacker or sent by an unauthenticated reader.

Once R_j is authenticated, T_r constructs

$$C := \mathrm{Rot}(K_1 + K_2, K_3 + \mu) + x_r \text{ and } D := \mathrm{Rot}(K_2 + K_3, K_1 + \mu) + y_r \tag{2}$$

and sends $C|D$ to R_j for tag authentication.

If R_j receives $C|D$ in time ΔT, it uses K and μ to extract x_r and y_r. It determines S' using (x_r, y_r) of T_r and (x_j, y_j) of R_j. If $S' = S$, then T_r is successfully authenticated. Using IDS, R_j retrieves T_r's unique ID from the TD and considers T_r with this ID as detected. This concludes the mutual authentication session P. Note that the challenge-response between R_j and T_r does not transmit T_r's ID. If $S' \neq S$, then T_r is illegitimate and the session is aborted. If R_j does not receive the response from T_r within time ΔT, the session is similarly abandoned.

The Updating Phase

Upon completion of the mutual authentication phase, IDS and $K = K_1|K_2|K_3$ are updated and synchronized separately. T_r assigns $IDS^{old} \leftarrow IDS$ and $K_i^{old} \leftarrow K_i$, stores them to prevent desynchronization, and computes

$$\begin{aligned} IDS^{new} &\leftarrow \mathrm{Rot}(IDS^{old} + \mu + x_r, K_1^{old} + K_2^{old} + y_r) + K_3^{old}, \\ K_1^{new} &\leftarrow \mathrm{Rot}(K_2^{old} + \mu, K_3^{old} + x_r) + K_1^{old}, \quad K_2^{new} \leftarrow \mathrm{Rot}(K_3^{old} + \mu, K_1^{new} + y_r) + K_2^{old}, \\ K_3^{new} &\leftarrow \mathrm{Rot}(K_1^{new} + \mu, K_2^{new} + x_r) + K_3^{old}. \end{aligned} \tag{3}$$

R_j updates its local values

$$IDS \leftarrow \mathrm{Rot}(IDS + \mu + x_r, K_1 + K_2 + y_r) + K_3, \qquad K_1 \leftarrow \mathrm{Rot}(K_2 + \mu, K_3 + x_r) + K_1,$$
$$K_2 \leftarrow \mathrm{Rot}(K_3 + \mu, K_1 + y_r) + K_2, \qquad K_3 \leftarrow \mathrm{Rot}(K_1 + \mu, K_2 + x_r) + K_3. \tag{4}$$

and sends them to TD. Here R_j does not store the old values, unlike T_r's in (3). It keeps only the updated local values. The protocol has now run a complete round. The next authentication session starts with tag identification.

4.2. Four Mechanisms

There are four mechanisms in UMAPSS.

Mutual Authentication
To ensure access control, UMAPSS deploys a double-entity-round mutual authentication mode. Unlike in previous protocols, its mutual authentication covers not only a check of consistency between the local and the received values using the same algorithm but also a check of consistency between the secret S stored in a valid reader and the recovered secret S' based on a $(2,n)$-SS after (x_r, y_r) is extracted from $C|D$.

Double Verification
The double verification between the R_j and T_r is achieved by requiring both key verification and secret recovery verification. Previous protocols require only key verification. Here in UMAPSS the tag authenticates the reader by extracting μ from $A|B$ by using K. In contrast to previous protocols, the reader authenticates the tag not by comparing the received values with the local values, but by comparing the stored secret S with the recovered secret S'. The secret recovery can be done in $\mathcal{O}(n\ log^2 n)$. This mechanism makes UMAPSS robust against impersonation attack and reduces the success probability of the MITM or the counterfeiting attack.

Session Control
UMAPSS has an overtime-exit function. R_j sets a time ΔT for each challenge-response session cycle while authenticating T_r. If ΔT is exceeded, the session is aborted to avoid some security threats that exploit timeouts.

Dynamic Update
UMAPSS dynamically updates relevant values and keys in each authentication session to avoid desynchronization among the tags, the readers, and the TD. In the mutual authentication phase, the random number μ is used to generate $A|B$, which is then used in a new challenge-response cycle for dynamic updating. Once a reader is authenticated, the tag updates IDS and K. Similarly, when a tag is authenticated, the reader updates its local values. This mechanism randomizes every challenge-response between the reader and the tag to resist the tracking, the replay, and the desynchronization attacks and provide forward security and unlinkability.

5. Security Analysis

This section analyzes the security of UMAPSS based on commonly required criteria. Some basic security sanity-check criteria were suggested in [38]. While it remains very challenging to satisfy all the requirements given the constraints on tags, we first note that our protocol avoids common mistakes that have been highlighted from the cryptanalysis of past protocols.

While Rot has some linear structure, the operation $+$ add some nonlinearity. Recall that $+$ adds the *integer* representations of two binary strings of length 96 each, take modulo 2^{96}, and then converting the resulting integer back to binary string of length 96. Note in particular that we design A, B, C and D to be nonlinear, as can be directly verified by, *e.g.*, checking that $A(x+y) \neq A(x) + A(y)$. Similarly with the other relevant functions. Performing the operation $+$ requires less than $2 \times 96 \times 7^2$ steps in time complexity, which is calculated on $2\mathscr{O}(L\log^2(L))$.

It is typical to assume that in an RFID deployment, the content of the tag's memory is not accessible to the attacker. We weaken this assumption to allow for the possibility of key leaks. So long as the secret share of the tag remains uncompromised, a robust security protection persists. Even if an attacker can obtain K_is of T_r, that is, the attacker passes Steps 2.1 to 3.1, it will fail to pass Step 3.2, *i.e.*, recovering S remains hard.

Data Confidentiality

All messages must be securely transmitted. In both forward and backward links, the tag's *ID* is replaced by its current *IDS*. Data coming out of the tag and transmitted between the reader and the tag is protected using μ. The construction of the public $A|B$ and $C|D$ involves K, μ, (x_r, y_r), and *IDS*. It is difficult to recover μ and (x_r, y_r) without knowing K. An attacker cannot obtain any secret information about a tag from the intercepted messages due to the dynamic updating. Thus, the tag's *ID* and the secret values are well-protected, assuring data confidentiality.

Data Integrity

It is hard to infer the secret values from the messages transmitted through the wireless channel between the tag and the reader. Messages $A|B$ and $C|D$ not only provide the ingredients for mutual authentication, but also vouch for the integrity of the secret values. If an attacker tries to modify μ by flipping certain bits in A, then the tag finds B invalid. It is hard for the attacker to adjust B to the correct value without knowing the shared secret key. Even a little modification on μ leads to a very different output. Meanwhile, *IDS* and K are updated periodically. The updating mechanism requires valid K, μ, and (x_r, y_r). Only legitimate parties can calculate these values. If the attacker succeeds in modifying the exchanged messages from any reader, then the tag detects an anomaly and, thus, identifies an attack.

Tag Anonymity

The tag never reveals its *ID* but uses its current *IDS* as its identity in the protocol. The exchanged messages are random because μ hides *IDS*, anonymizing the challenge-response messages. It is easy to verify that $A|B$ and $C|D$ are independent of *ID*. Both the reader and the tag do not leak any information related to *ID* to any third-party. Even if an attacker intercepts and decodes *IDS*, $A|B$, and $C|D$, it obtains no relevant information about the tag's *ID*.

Mutual Authentication

Our protocol ensures that the reader and the tag authenticate each other and blocks any unauthorized access. Only a reader possessing K can generate a valid $A|B$. Similarly, only a tag that has K can compute μ from $A|B$ and then construct $C|D$. Only a legitimate reader can derive the tag's (x_r, y_r) from $C|D$ and successfully recovers S. Thus, only a valid party can generate valid messages and be authenticated by the other party.

Untraceability

Tracking and traceability are two potential tag's vulnerabilities. If a tag uses its *ID*, it is easy for an attacker to locate and trace where the tag has been. To prevent this leak, the tag uses its current *IDS*, instead of *ID*, and updates *IDS* and K after each successful mutual authentication. With this mechanism, tracing and tracking become more difficult since no attacker knows what the next *IDS* and K will be. A fresh *IDS* is backscattered when the tag is interrogated.

To protect the secret data in transmission over the R-T channel, $A|B$ and $C|D$ change dynamically at random from one authentication session to the next. Since a tag's successive *IDS*s and messages $A|B$s and $C|D$s look random, an attacker cannot retrieve the tag's *ID* and fails to obtain the same responses from a particular tag by interfering with two or more dependent challenge-response rounds. All communications between the tag and the reader remain unlinkable, thwarting the tracking attack. A legitimate tag's location privacy is not compromised.

Forward Security

Forward security guarantees the security of past communications even when a tag becomes compromised at a later stage. In UMAPSS, it is naturally embedded. Even if an attacker compromises a tag and acquires its current K, any information on the tag's previous interactions cannot be inferred by the attacker since μ is freshly generated and *IDS* and K are automatically updated.

It is clear that K and μ are random and periodic. An attacker cannot obtain the secret key $K_{j-\Delta T}$ of session $j-\Delta T$ based on K_j in session j due to the hardness of breaking the PRNG. Even if the tag becomes compromised in session j, the authentication sessions prior to session j remain valid. The attacker cannot compromise the tag's past communications from the historical transaction records.

6. Analysis on Attack Models

Due to the fact that most proposed protocols had been quickly shown to be vulnerable against at least one attack (see [38, Table 1] for a comprehensive list up to mid 2014), users need to be reasonable in managing their expectation. It obviously remains a challenge to satisfy all desired security aspects, especially against general attacks by a powerful adversary, while keeping the tags ultralightweight. The specifics of the deployment environment can help decide which protocols or aspects of certain protocols to retain and what price in terms of security one should prepare to pay.

In proposing UMAPSS we do not claim to have overcome all of the limitations of past protocols. Given known modes of attacks, we add the secret sharing ingredients and remove vulnerable operations to enhance the security while keeping the cost at least on par with prior ultralightweight protocols. We show that UMAPSS resists multiple modes of malicious attacks. They include the replay, the MITM, the counterfeit, the desynchronization, and the disclosure attacks. Against the tango attack, we remove known weak operations to reduce the attack's success probability.

Against the Replay Attack

Each session's μ and K are required to generate $A|B$ and $C|D$. This ensures that replay messages from either the tag or the reader will not be authenticated. In the replay attack, an attacker has access to and can store all exchanged messages. It can then proceed as follows.

1. It replays $C|D_{j-1}$ in session j.

First, suppose that authentication in session $j-1$ failed. Hence, $K_j = K_{j-1}$ and T_r uses IDS^{old} in session j. The reader constructs $A|B_j$ by using K_j and a freshly generated μ_j, and forwards it to T_r. After authenticating the reader, T_r computes $C|D_j$, by using K_j, μ_j, and (x_r, y_r), and sends it to the reader. The attacker intercepts $C|D_j$ from the tag and replays $C|D_{j-1}$ to T_r. Even if $K_{j-1} = K_j$, the (x'_r, y'_r) that the reader gets would not match (x_r, y_r) since $\mu_j \neq \mu_{j-1}$. The reader fails to recover S and the replayed $C|D_{j-1}$ is deemed invalid.

The second scenario supposes that authentication in session $j-1$ worked. Hence, $K_j \neq K_{j-1}$ and T_r uses IDS^{new} in session j. The reader constructs $A|B_j$ using K_j and the freshly supplied μ_j before forwarding $A|B_j$ to T_r. After authenticating the reader, T_r computes and sends $C|D_j$ to the reader. The attacker intercepts $C|D_j$ and replays $C|D_{j-1}$ to the reader, which then proceeds to extract $(x'_r, y'_r) \neq (x_r, y_r)$ since $K_j \neq K_{j-1}$ and $\mu_j \neq \mu_{j-1}$, making $S' \neq S$.

In another scenario, the attacker assumes that the reader and the tag have not done the updating phase in session $j-1$ and waits for IDS^{old} from the tag. A similar analysis to the first scenario shows that replaying $C|D_{j-1}$ leads to the reader's failure to authenticate the attacker.

2. It replays $A|B_{j-1}$ in session j.

Suppose that authentication in session $j-1$ failed. Hence, $K_{j-1} = K_j$ and T_r uses IDS^{old} in session j. The reader sends $A|B_j$ to T_r. The attacker intercepts $A|B_j$ but replays $A|B_{j-1}$ to T_r. Here T_r authenticates the attacker as a legitimate reader. T_r then uses K_j, μ_j, and (x_r, y_r) to construct $C|D_j$ and send it to the reader. The attacker can then intercept $C|D_j$. Note, however, that an attacker that had been successfully authenticated as a valid reader gains no secret information from $C|D_j$ since it knows none of K_j, μ_j and (x_r, y_r). In addition, even if the attacker lets $C|D_j$ reach the reader, the extracted (x'_r, y'_r) would not match (x_r, y_r), implying that the reader cannot recover S and the replayed $A|B_{j-1}$ is declared invalid.

Next, assume that authentication in session $j-1$ worked. Hence, $K_j \neq K_{j-1}$ and T_r uses IDS^{new}. The reader constructs a new $A|B_j$ and forwards it to the tag. The attacker intercepts $A|B_j$ but replays the challenge $A|B_{j-1}$. Since $K_j \neq K_{j-1}$ and $IDS^{new} \neq IDS^{old}$, the μ'_j extracted from $A|B_{j-1}$ matches μ_j with negligible probability. The tag declares the attacker an illegitimate reader.

Finally, the attacker may pretend that the reader and the tag have not accomplished the updating phase in session $j-1$. It waits for IDS^{old} from T_r. Similar to the above failed authentication in session $j-1$, the replayed $A|B_{j-1}$ would be found invalid.

Against the MITM Attack

UMAPSS provides a strong integrity and authentication protection on $A|B$ and $C|D$. Any modification on them leads to a failure to pass the validity verification. It is hard for the attacker to change the data without detection. In order to obtain the tag's valid messages in a successful authentication session, the attacker tries to modify A to A' and B to B' with either $A \neq A'$ or $B \neq B'$ and, similarly, change C to C' and D to D' with either $C \neq C'$ or $D \neq D'$ without getting noticed. An MITM attack may take several forms.

1. Modify $A|B$ to $A'|B'$.

Assume that the attacker sends $A'|B'$ to T_r. The tag then discovers that the extracted μ' from $A'|B'$ is not equal to μ from $A|B$. Both are computed by using the current K and IDS. Thus, $A'|B'$ is declared invalid, foiling the attack.

What if the attacker sends $A'|B'$ to an illegitimate tag? The tag, similarly, fails to verify and, hence, rejects $A'|B'$. It can use the modify-and-test method to guess the forged key $K' = K'_1|K'_2|K'_3$. Extracting the valid μ from $A'|B'$, however, is as hard as breaking *PRNG*. Even if this illegitimate tag pretends to have verified μ, it cannot construct valid $C|D$ since it does not know K, μ, and (x_r, y_r).

2. Modify $C|D$ to $C'|D'$.

   The attacker transmits $C'|D'$ to T_r, which then checks whether the recovered secret S' from $C'|D'$ is equal to S. It computes (x'_r, y'_r) from $C'|D'$ by using K. It is clear that $S' \neq S$ because $C'|D'$ is not the valid $C|D$ from the tag. The reader declares $C'|D'$ invalid.

   Assume now that the attacker transmits $C'|D'$ to an illegitimate reader, which then fails to match S' and S since only a legitimate reader stores S. Even if the illegitimate reader pretends to have validated $C'|D'$, the attacker cannot gain any secret information of a valid tag T_r from $C'|D'$ since it does not have K, μ, and (x_r, y_r).

In either case, an attacker cannot succesfully interfere in the challenge-response process.

Against Counterfeiting Attack

A timeout mechanism ensures that a legitimate tag responds within a time period ΔT. The reader refuses any response that exceeds the allotted time in one authentication session. We consider two cases, each having two possible scenarios.

1. Tag Impersonation.

   An attacker tries to impersonate a legitimate tag T_r within the broadcast range of some readers by forging the secret key $K' = K'_1|K'_2|K'_3$. Even if the attacker has the valid $A|B$, obtaining T_r's K by using the modify-and-test method to guess the correct value of μ means breaking PRNG.

   Suppose that the attacker impersonates T_r with a forged key K' upon receiving the valid $A|B$ from a legitimate reader. The forged tag will fail to guess μ correctly since $K \neq K'$. The forged tag may pretend to have gotten the correct μ by using K'. It will, however, fail to construct $C|D$ since it has no knowledge of K, the correct μ, and (x_r, y_r). The forged tag now has two possible response scenarios.

   First, it sends a forged response $C'|D'$ to a reader regardless of the latter's legitimacy. An argument similar to the one in the second case of the MITM attack above suffices to demonstrate that this tag will not be authenticated by any reader.

   Second, it replays $C|D_{j-1}$ in session j. A legitimate reader will fail to authenticate this tag, irrespective of whether the reader had performed the updating phase in session $j-1$. One can follow the reasoning given in the first case of the replay attack above to settle the matter.

2. Reader Impersonation.

An attacker tries to impersonate a legitimate reader in the current session and sends the challenge to the tag. Recall that the tag's *ID* plays no role in the transmitted challenge-response over the channel. Moreover, μ is freshly generated by a PRNG for each session. The forged reader cannot obtain any information about the tag's private information. If the forged reader eavesdrops and modifies the transmitted challenge-response, then the authentication session fails, foiling the attack. There are two scenarios to consider.

First, the forged reader sends a forged challenge $A'|B'$ constructed using a forged key K'. Any tag, legitimate or otherwise, is unable to authenticate the reader as already shown in the analysis for the first case of the MITM attack above.

Second, it replays $A|B_{j-1}$. Following the analysis for the second case in the replay attack shows why the tag, irrespective of whether it had done proper updating in session $j-1$, will fail to authenticate this reader.

Against Desynchronization

This attack tries to force the tag and the reader to use different random numbers to update their respective local data, causing authentication failure in all future transactions. An attacker can modify the transmitted messages to change the value of μ. UMAPSS guards the authenticity and the integrity of μ. Potential next secret key K is verified to ensure the correctness of $A|B$ and $C|D$. The process requires IDS, μ, and K. It is infeasible for the attacker to change the transmission without being noticed.

Old as well as potential new values of IDS and K are stored in the tag's memory as an extra precaution. Even if the attacker manages to make the tag update its local data while keeping the reader from doing so, *e.g.*, by intercepting $C|D$, the reader and the tag can still authenticate each other using the old values (IDS^{old}, K^{old}) in the next session. They will then be able to recover their synchronized state and recognize subsequent communication requests.

Against Disclosure Attack

In a disclosure attack (see, *e.g.*, [6], [31], and [40]) an attacker modifies the challenge from the reader slightly in the hope of gaining partial information from the tag's response. We have already shown earlier that our protocol detects any such modification.

Against the Tango Attack

The tango attack [41] mainly exploits the imbalance of the OR and AND bitwise operations, some improper message designs, and the fixed positions of the bits. Such an attack tries to obtain good approximations of the secrets. UMAPSS does not use OR and AND. It deploys the circular shift $\text{Rot}(x,y)$ to obfuscate the original positions of the bits, making it difficult to find proper approximations of the secrets without the secret values K, μ, and (x_r, y_r).

7. Formal Security Analysis

Sections 5 and 6 give some rather informal analysis of UMAPSS. We now present a formal analysis using Casper/FDR. Communication Sequential Process (CSP) is a language to specify a protocol's process. The generated CSP file is then analyzed using Failure-Divergence Refinement (FDR), which is a model checker that verifies the specifications of the protocol. The Casper/FDR tool [42] is a compiler to check the soundness of the protocol based on the specified security requirements. It takes a high-level description of the protocol and analyses the protocol description

Table 2: Casper Specifications: Free Variables and Protocol Description

Free Variables	
T	Agent
R	Server
μ	initial Seq
$IDS, IDS^{old}, IDS^{new}$	Session ID
$K_1^{old}, K_1^{new}, K_2^{old}, K_2^{new}, K_3^{old}, K_3^{new}, K_1, K_2, K_3, x_j, y_j, x_r, y_r$	Session Key
Rot	Circular Shift
$(K_1^{old}, K_1^{old}), (K_1^{new}, K_1^{new}), (K_2^{old}, K_2^{old}), (K_2^{new}, K_2^{new}), (K_3^{old}, K_3^{old}), (K_3^{new}, K_3^{new}), (K_1, K_1), (K_2, K_2), (K_3, K_3), (x_j, x_j), (y_j, y_j), (x_r, x_r), (y_r, y_r), (IDS^{old}, IDS^{old}), (IDS^{new}, IDS^{new}), (IDS, IDS).$	Inverse Keys
Protocol Description	
0. R	T
1. $R \rightarrow T$	`hello`
2. $T \rightarrow R$	IDS
$3a.$ $R \rightarrow T$	$Rot(Rot(IDS + \mu, K_1), K_2) + K3$
$3b.$ $R \rightarrow T$	$Rot(Rot(IDS + K_1, \mu), K_3) + K_2$
	$[(IDS \leftarrow IDS^{old}$ and $(K_i \leftarrow K_i^{old} : 1 \leq i \leq 3)$ or $(IDS \leftarrow IDS^{new}$ and $K_i \leftarrow K_i^{new} : 1 \leq i \leq 3)]$
$4a.$ $T \rightarrow R$	$Rot(K_1 + K_2, K_3 + \mu) + x_r$
$4b.$ $T \rightarrow R$	$Rot(K_2 + K_3, K_1 + \mu) + y_r$

against the stated specification. It has been used to model communication and security protocols and verify its authentication and security requirements. Its capability to find vulnerabilities has been demonstrated in many protocols, see, *e.g.*, [43], [44], and [44] for more details.

To verify UMAPSS formally, we specify the free variables and the protocol's steps in Table 2. Table 3 details the rest of the Casper code. Based on the **Specification** section in the script, Casper/FDR would not find any feasible attack from among those mentioned in Section 6.

We now formally prove our claim in Section 5 that UMAPSS can resist traceability attacks by using the Raphael's traceability model, also known as the Ouafi-Phan model from [46]. Consider an adversary A performing the following steps.

1. Learning phase: A eavesdrops a perfect session between a legitimate tag and a legitimate reader and obtains $A|B$ from the legitimate reader and $C|D$ from the legitimate tag.

2. Challenge phase: A chooses two fresh tags T_0 and T_1, having IDS_0 and IDS_1 respectively,

Table 3: Casper Specifications: The Remaining Details

Processes
RESPONDER$(T,R,\mu,x_r,y_r,K_1^{old},K_1^{new},K_2^{old},K_2^{new},K_3^{old},K_3^{new},IDS^{old},IDS^{new})$
SERVER $(R,T,\mu,K_1,K_2,K_3,IDS)$
Actual Variables
Tag and Mallory are the Agents
R and μ from Table 2
IDS entity R, IDS entity T : Session ID
Session Key and InverseKeys from Table 2
Specification
Aliveness(R,T)
Secret$(T,K_1,[R])$
Secret$(T,K_2,[R])$
Secret$(T,K_3,[R])$
Secret$(T,IDS,[R])$
Secret$(T,\mu,[R])$
Secret$(T,x_r,[T])$
Secret$(T,y_r,[T])$
Secret$(R,x_j,[R])$
Secret$(R,y_j,[R])$
Agreement$(R,T,[\mu,IDS,K_1,K_2,K_3])$
Agreement$(T,R,[\mu,K_1,K_2,K_3,x_r,y_r,x_j,y_j])$
System
RESPONDER$(T,R,K_1^{old},K_1^{new};K_2^{old},K_2^{new};K_3^{old},K_3^{new},x_r,y_r,IDS$ entity $T)$
SERVER$(R,T,\mu,K_1,K_2,K_3,IDS$ entity $R)$
Intruder Information
Intruder = Mallory
IntruderKnowledge= {Tag, Reader, Mallory}

to be tested and sends a `test` query. A performs the `execute` query by sending $A|B$ and receives $C'|D'$ in return.

3. Guess phase: If $C = C'$ and $D = D'$, then A outputs $b' = 0$, otherwise it outputs $b' = 1$. Let $Pr(E)$ be the probability that an event E occurs. We show that the advantage of A

$$Adv_A \triangleq |Pr(A \text{ wins}) - Pr(\text{random coin flip})| = |Pr([b' = b]) - 0.5)|$$

is negligible, *i.e.*, smaller than a chosen miniscule positive value ε. In UMAPSS, to protect the transmission of secret data, the public messages $A|B$ and $C|D$ over the *R*-*T* channel all depend on the secret values shared only between the legitimate reader and genuine tag, including the dynamic random number μ and the updated secret key $K = K_1|K_2|K_3$. The messages are randomized and change dynamically in different authentication sessions. Over different sessions, because of the successive *IDS*, the transmitted $A|B$ and $C|D$ from the same tag look random. The attacker cannot identify the identity of the tag and cannot obtain the same responses from the same tag by interfering with two or more dependent challenge-responses. The desired conclusion follows immediately and, consequently, all communications between the tag and the reader are unlinkable.

8. Performance Evaluation

We simulated the communication between the tag and the reader using a program written in `C`. The development tool was VS 2010 Integrated Development Environment (IDE) with client simulating a tag and the server simulating a reader. The network interaction was done by using sockets that abstracted a TCP client/server connection. The reader waited for the connection with the tag on a specified IP address and port. Once the tag successfully established a connection with the reader, the protocol then executed one mutual-authentication session. Both the server and the client machines ran Windows 10 on a hardware with Intel Core i5-3210 CPU at 2.50 GHz equipped with 8 GB of RAM. The secret sharing scheme used a quadratic polynomial in $\mathbb{F}_{101}[x]$. The average time required for a *complete* mutual authentication protocol executing the steps in Figure 1 was 52 miliseconds. The average execution times for *GenS* and *RecS* were, respectively, 1.2 and 1 miliseconds, showing that indeed the secret sharing part was feasible.

Table 4 summarizes a performance evaluation and some comparison between UMAPSS and several previous protocols. On the protocol's computational cost, storage requirement, and communication overhead, it suffices to evaluate the tag's performance since the hardware environments of the reader and the backend database are not as constrained. The variable i in UP^{2}RT [49] is a positive integer that depends on the amount of random numbers utilized. The reader controls this value.

All operations in UMAPSS are suitable for ultralightweight tags since their hardware implementation is very efficient. In terms of computation, we focus on the types and frequencies of operations for each tag. The storage requirement measures the memory to store the static values, the shared keys, and the random number used in one authentication session. The communication overhead calculates the transmitted messages over the channel in one authentication session.

Computational Cost

Costly operations, *e.g.*, multiplications and hash evaluations are not used in UMAPSS. Only simple control commands and four primitive arithmetic operations are required, namely a polynomial $f(x)$, a pseudo-random number generator $PRNG()$, bitwise addition modulo 2^{96}, and the circular shift $\mathrm{Rot}(x, y)$.

Table 4: Performance, Security, and Resistance Comparisons

No.		LMAP [28]	EMAP [29]	M^{2}AP [30]	SASI [5]	Gossamer [34]	UAPP [47]	LPCP [48]	UP^{2}RT [49]	UMAPSS
	Performance Comparison									
1	Types and frequencies of operations per tag	$+^2$, $\oplus^2$, OR^1	$\oplus^5$, AND^1, OR^1	$+^2$, $\oplus^2$, AND^1, OR^1	$+^1$, $\oplus^2$, OR^1, Rot^1	$+^6$, $\oplus^2$, Rot^2, MixBits^1	$\oplus^3$, Rot^1, Perm^2	$\oplus^5$, Perm^2 (CRC 16)^5	$\oplus^4$, Perm^1, Rot^1, Random tuple	$+^5$, Rot^1
2	Storage requirement on tag including temporal nonces	$6L$	$6L$	$6L$	$9L$	$12L$	$6L$	$6L$	$(8+2i)L$	$11L$
3	Random number requirement between R and T	$2L$	$2L$	$2L$	$2L$	$2L$	$2L$	$2L$	$(4+2i)L$	L
4	Memory size per tag on database	$6L$	$6L$	$6L$	$4L$	$4L$	$9L$	$9L$	$9L$	$4L$
5	Total communication messages for mutual authentication	$4L$	$5L$	$5L$	$4L$	$4L$	$5L$	$5L$	$(3+2i)L$	$4L$
	Security Properties									
1	Data confidentiality	No	No	No	No	No	No	No	No	Yes
2	Data integrity	No	No	No	No	No	No	No	No	Yes
3	Tag anonymity	No	No	No	No	Yes	No	Yes	Yes	Yes
4	Mutual authentication	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	Untraceability	No	No	No	No	Yes	Yes	Yes	Yes	Yes
6	Forward security	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Resistance Properties									
1	Replay attack	No	No	No	No	No	No	No	No	Yes
2	MITM attack	No	No	No	No	No	No	No	No	Yes
3	Counterfeiting attack	No	No	No	No	No	No	No	No	Yes
4	Desynchronization attack	No	No	No	No	No	No	No	No	Yes
5	Disclosure attack	No	No	No	No	Yes	Yes	Yes	Yes	Yes
6	Tango attack	No	No	No	No	Yes	Yes	Yes	Yes	Yes

The *GenS* algorithm is performed by the TD, the routine $PRNG()$ is implemented at the reader's end. Only two ultralightweight bitwise operations of addition modulo 2^{96} and $\mathrm{Rot}(x,y)$ are required at the tag's end. Their implementations are low-cost and highly efficient, requiring no additional hardware. The recovery *RecS* of a secret based on the $(2,n)$-SS is done at the reader's end. The required bitwise operations consume much less storage than cryptographic primitives such as hash function, cyclic redundancy code, and the sign-then-encrypt algorithms do.

Storage Requirement

Each tag stores *ID*, its share (x_r, y_r), and two records (the old and the potential new values) on *IDS* and K. Notes that in total only values relevant for two consecutive sessions are stored. A 96-bit length is assumed for all elements in accordance with the EPCGlobal Gen2 tag used in data deliveries. Since *ID* and (x_r, y_r) are static values, they are stored in ROM. The *IDS* and K values, both old and new, occupy $96 \times 8 = 768$ bits and are stored in a rewritable memory to be accessed during different authentication sessions.

In contrast to SASI and Gossamer that need, respectively, two and five temporal nonces for each authentication session, UMAPSS does not require any. Let L stands for 96 bits. Each tag in UMAPSS stores at most $11L$, less than the $12L$ in Gossamer but more than the $9L$ in SASI. Most other protocols link two random numbers to each session while UMAPSS uses only one, making it cheaper and easier to implement.

Communication Overhead

The communication overhead depends on the number of per-round exchanged messages between the readers and the tags. The total authentication process normally takes at least four rounds. Since the mutual authentication phase dominates the communication cost, it suffices to count this phase's number of messages. There are only four exchanged messages $A|B$ and $C|D$ for the individual challenge-and-response. Hence, the number of the transmitted messages is equal to that of SASI and Gossamer. The mutual authentication phase needs $4L = 384$ bits to be sent over the channel.

The comparison shows UMAPSS' superiority to the other protocols. In particular, both in types and frequency of per-tag computations and in the random number requirement, it takes less resources. Including the required temporal nonces in one authentication session, UMAPSS' storage requirement falls between that of SASI and Gossamer. The remaining aspects, including the memory size for each tag on database and the total communication messages, are similar to those of SASI and Gossamer. Our protocol remains ultralightweight, albeit with a storage requirement of $11L$ at the tag's end, which is considered almost borderline with being lightweight instead of ultralightweight.

The table also contains a comparison on the security and privacy aspects. Our protocol is superior since it supports data confidentiality and integrity, tag anonymity, mutual authentication, untraceability, and forward security. It protects against a variety of threats. In order to notably improve on the security level and reduce computational cost, it stores extra nonces, a very small price to pay. UMAPSS is implementable without obvious vulnerabilities, suitable for the resource-limited RFID applications fulfilling highly cost-effective requirements.

9. Conclusion

To overcome known security weaknesses and/or privacy omission in previous ultralightweight authentication protocols, we propose UMAPSS. It incorporates the $(2,n)$ Shamir's secret sharing and achieves significant security enhancement. It supplies a robust privacy protection through mechanisms for double verification and mutual authentication. It reduces the overtime security omission using a session control mechanism. It ensures unlinkability and randomization by applying a dynamic update mechanism. Based on typical security characteristics and the ability to resist malicious attacks, our protocol performs favourably.

It is ultralightweight, requiring only two simple bitwise operations on low-cost RFID tags without significantly increasing the burden at both the tag's and the server's ends. It removes triangular function operations, lessening exposure to security issues related to their biased outputs. UMAPSS balances superior security performance and practical competitiveness.

Acknowledgements

We thank Huang Keke who implemented our protocol and supplied us with the data. He is a Master student at the College of Information Engineering, Yangzhou University, Yangzhou, Jiangsu, 225127, China.

Y-L. Liu's work was supported by the National Natural Science Foundation of China Grant 61702237, the Natural Science Foundation of Jiangsu Province Grant BK20150241, the Natural Science Foundation of the Higher Education Institutions of Jiangsu Province Grant 14KJB520010, the Scientific Research Support Project for Teachers with Doctors Degree of Jiangsu Normal University Grant 14XLR035, and Jiangsu Provincial Government Scholarship for Overseas Studies. She was also partially supported by the National Natural Science Foundation of China Grants 61304174 and 61100167. Research Grants TL-9014101684-01 and MOE2013-T2-1-041 support the research carried out by M. F. Ezerman and H. Wang.

References

[1] S. Bono, M. Green, A. Stubblefield, A. Juels, A. D. Rubin, M. Szydlo, Security analysis of a cryptographically-enabled RFID device., in: Usenix Security, Vol. 5, 2005, pp. 1–16.

[2] S. A. Weis, Security and privacy in radio-frequency identification devices, Ph.D. Thesis, Massachusetts Institute of Technology (2003).

[3] A. Juels, RFID security and privacy: A research survey, IEEE Journal on Selected Areas in Communications 24 (2) (2006) 381–394.

[4] L. Ya-li, Q. Xiao-lin, L. Bo-han, L. Liang, A forward-secure grouping-proof protocol for multiple RFID tags, Int. J. of Computational Intelligence Systems 5 (5) (2012) 824–833.

[5] H.-Y. Chien, SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity, IEEE Trans. on Dependable and Secure Computing 4 (4) (2007) 337–340.

[6] H.-Y. Chien, C.-W. Huang, Security of ultra-lightweight RFID authentication protocols and its improvements, ACM SIGOPS Operating Systems Review 41 (4) (2007) 83–86.

[7] M. Burmester, J. Munilla, Lightweight RFID authentication with forward and backward security, ACM Trans. on Information and System Security (TISSEC) 14 (1) (2011) 11.

[8] G. R. Blakley, Safeguarding cryptographic keys, Proc. of the Nat. Computer Conf. 48 (1979) 313–317.

[9] A. Shamir, How to share a secret, Communications of the ACM 22 (11) (1979) 612–613.

[10] R. Lidl, H. Niederreiter, Finite Fields, Encyclopaedia of Mathematics and Its Applications, Cambridge Univ. Press, New York, 1997.

[11] J. Guo, T. Peyrin, A. Poschmann, The PHOTON family of lightweight hash functions, in: Proc. of the 31st Annual Conf. on Advances in Cryptology, CRYPTO'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 222–239.

[12] M. Chen, S. Chen, Y. Fang, Lightweight anonymous authentication protocols for RFID systems, IEEE/ACM Trans. on Networking 25 (3) (2017) 1475–1488.

[13] A. C. Atici, L. Batina, J. Fan, I. Verbauwhede, S. B. O. Yalcin, Low-cost implementations of NTRU for pervasive security, in: Proc. Int. Conf. on Application-Specific Systems, Architectures and Processors, 2008, pp. 79–84.

[14] Y-W. Hwang, I-Y. Lee, A study on lightweight mutual authentication for radio-frequency identification medical device, Int. J. of Engineering Business Management 10 (2018) 1–11.

[15] N. Chikouche, F. Cherif, P-L. Cayrel, M. Benmohammed, RFID authentication protocols based on error-correcting codes: a survey, Wireless Personal Communications 96 (1) (2017) 509–527.

[16] P. Urien, S. Piramuthu, Elliptic curve-based RFID/NFC authentication with temperature sensor input for relay attacks, Decision Support Systems 59 (2014) 28 – 36.

[17] Y.-P. Liao, C.-M. Hsiao, A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol, Ad Hoc Networks 18 (Supplement C) (2014) 133 – 146.

[18] R. Peeters, J. Hermans, Attack on Liao and Hsiao's secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol, Cryptology ePrint Archive, Report 2013/399, `http://eprint.iacr.org/2013/399` (2013).

[19] N. Hopper, M. Blum, A secure human-computer authentication scheme, Tech. rep., Computer Science Department, Carnegie Mellon University (2000). URL `http://repository.cmu.edu/compsci/2112/`

[20] G. Deng, H. Li, Y. Zhang, J. Wang, Tree-LSHB+: An LPN-based lightweight mutual authentication RFID protocol, Wireless Personal Communications 72 (1) (2013) 159–174.

[21] T. Li, Y. Li, G. Wang, Secure and Practical Key Distribution for RFID-Enabled Supply Chains, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 356–372.

[22] S. Abughazalah, K. Markantonakis, K. Mayes, Enhancing the key distribution model in the RFID-enabled supply chains, in: 28th Int. Conf. on Advanced Information Networking and Applications Workshops, 2014, pp. 871–878.

[23] K. Toyoda, I. Sasase, Secret sharing based unidirectional key distribution with dummy tags in Gen2v2 RFID-enabled supply chains, in: 2015 IEEE Int. Conf. on RFID (RFID), 2015, pp. 63–69.

[24] G. Avoine, M. A. Bingl, X. Carpent, S. B. O. Yalcin, Privacy-friendly authentication in RFID systems: On sublinear protocols based on symmetric-key cryptography, IEEE Trans. on Mobile Computing 12 (10) (2013) 2037–2049.

[25] H. Kapoor, D. Huang, Secret-sharing based secure communication protocols for passive RFIDs, in: GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conf., 2009, pp. 1–6.

[26] I. Vajda, L. Buttyán, Lightweight authentication protocols for low-cost RFID tags, in: Second Workshop on Security in Ubiquitous Computing UBICOM, 2003.

[27] A. Juels, Minimalist cryptography for low-cost RFID tags, in: Int. Conf. on Security in Communication Networks, Springer, 2004, pp. 149–164.

[28] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estévez-Tapiador, A. Ribagorda, LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags, in: Proc. of 2nd Workshop on RFID Security, 2006, p. 06.

[29] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, A. Ribagorda, EMAP: An efficient mutual-authentication protocol for low-cost RFID tags, in: On the move to meaningful internet systems: OTM 2006 Workshops, Springer, 2006, pp. 352–361.

[30] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, A. Ribagorda, M^2AP: a minimalist mutual-authentication protocol for low-cost RFID tags, Ubiquitous intelligence and computing (2006), pp. 912–923.

[31] T. Li, R. Deng, Vulnerability analysis of EMAP, an efficient RFID mutual authentication protocol, in: Availability, Reliability and Security, 2007. ARES 2007. The Second Int. Conf. on, IEEE, 2007, pp. 238–245.

[32] M. Bárász, B. Boros, P. Ligeti, K. Lója, D. Nagy, Passive attack against the M^2AP mutual authentication protocol for RFID tags, in: Proc. of First Int. EURASIP Workshop on RFID Technology, 2007, pp. 37–48.

[33] P. D'Arco, A. De Santis, On ultralightweight RFID authentication protocols, IEEE Trans. on Dependable and Secure Computing 8 (4) (2011) 548–563.

[34] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Tapiador, A. Ribagorda, Advances in ultralightweight cryptography for low-cost RFID tags: Gossamer protocol, in: Int. Workshop on Information Security Applications, Springer, 2008, pp. 56–68.

[35] Y.-C. Lee, Y.-C. Hsieh, P.-S. You, T.-C. Chen, A new ultralightweight RFID protocol with mutual authentication, in: Information Engineering, 2009. ICIE'09. WASE Int. Conf. on, Vol. 2, IEEE, 2009, pp. 58–61.

[36] A. Klimov, A. Shamir, New applications of t-functions in block ciphers and hash functions, in: Int. Workshop on Fast Software Encryption, Springer, 2005, pp. 18–31.

[37] S. Piramuthu, RFID mutual authentication protocols, Decision Support Systems 50 (2) (2011) 387 – 393.

[38] G. Avoine, X. Carpent, J. Hernandez-Castro, Pitfalls in ultralightweight authentication protocol designs, IEEE Trans. on Mobile Computing 15 (9) (2016) 2317–2332.

[39] K. Mandal, X. Fan, G. Gong, Design and implementation of Warbler family of lightweight pseudorandom number generators for smart devices, ACM Trans. Embed. Comput. Syst. 15 (1) (2016) 1:1–1:28.

[40] T. Li, G. Wang, Security analysis of two ultra-lightweight RFID authentication protocols, New approaches for security, privacy and trust in complex environments (2007) 109–120.

[41] J. C. Hernandez-Castro, P. Peris-Lopez, R. C.-W. Phan, J. M. Tapiador, Cryptanalysis of the David-Prasad RFID ultralightweight authentication protocol, in: Int. Workshop on Radio Frequency Identification: Security and Privacy Issues, Springer, 2010, pp. 22–34.

[42] G. Lowe, Casper: A compiler for the analysis of security protocols, in: Proc. 10th. IEEE Computer Security Foundations Workshop, 1997, pp. 18–30.

[43] A. Alshehri, J. A. Briffa, S. Schneider, S. Wesemeyer, Formal security analysis of NFC *m*-coupon protocols using Casper/FDR, in: Proc. 5th IEEE Int. Workshop on Near Field Communication (NFC), 2013, pp. 1–6.

[44] M. Aiash, G. Mapp, R. W. Phan, A. Lasebae, J. Loo, A formally verified device authentication protocol using Casper/FDR, in: Proc. 11th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications (Trust-Com), 2012, pp. 1293–1298.

[45] V. V. Kumari, K. K. Raju, Formal verification of IEEE 802.11w authentication protocol, in: Proc. 2nd Int. Conf. on Communication, Computing and Security (ICCCS), 2012, pp. 716–722.

[46] K. Ouafi, R. C. W. Phan, Privacy of recent RFID authentication protocols, in: Proc. Int. Conf. on Information Security Practice and Experience, 2008, pp. 263–277.

[47] Y. Tian, G. Chen, J. Li, A new ultralightweight RFID authentication protocol with permutation, IEEE Communications Letters 16 (5) (2012) 702–705.

[48] L. Gao, M. Ma, Y. Shu, Y. Wei, An ultralightweight RFID authentication protocol with CRC and permutation, J. of Network and Computer Applications 41 (2014) 37 – 46.

[49] L. Gao, M. Ma, Y. Shu, F. Lin, L. Zhang, Y. Wei, A low-cost RFID authentication protocol against desynchronization with a random tuple, Wireless Personal Communications 79 (3) (2014) 1941–1958.

Vitae

Ya-Li Liu received the B.Sc. degree in computer science and technology in 2002 from Jiangsu Normal University, China and the M.Sc. degree in computer science and technology in 2008 from Yangzhou University, China. She received the Ph.D. degree in 2014 from Nanjing University of Aeronautics and Astronautics, China. A member of ACM, CCF and YOCSEF, she is an associate professor at the Computer Science and Technology College, Jiangsu Normal University. Her main research interests include RFID authentication and privacy protection technology, cryptographic algorithms and protocols as well as their applications to computer and network security and mobile communications.

Martianus Frederic Ezerman grew up in East Java Indonesia. He received the B.A. degree in philosophy and the B.Sc. degree in mathematics in 2005 and the M.Sc. degree in mathematics in 2007, all from Ateneo de Manila University, Philippines. In 2011 he obtained the Ph.D. degree in mathematics from Nanyang Technological University, Singapore. After research fellowships at Université Libre de Bruxelles, Belgium and at the Centre for Quantum Technologies, National University of Singapore, he returned in March 2014 to NTU where he is currently a Senior Research Fellow. His main interests are coding theory, cryptography, and quantum information processing.

Huaxiong Wang received the Ph.D. degree in mathematics from University of Haifa, Israel, in 1996, and the Ph.D. degree in computer science from University of Wollongong, Australia, in 2001. Since 2006, he has been an associate professor with the Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. His research interests include cryptography, information Security, coding theory, and theoretical computer science.