mathematics

MDPI

Article

Cryptanalysis of Two Recent Ultra-Lightweight Authentication Protocols

Mohammad Reza Servati [1], Masoumeh Safkhani [1,2], Saqib Ali [3], Mazhar Hussain Malik [4], Omed Hassan Ahmed [5], Mehdi Hosseinzadeh [6,*] and Amir H. Mosavi [7,*]

[1] Faculty of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran 16788-15811, Iran
[2] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), P.O. Box 19395-5746, Tehran 16788-15811, Iran
[3] Department of Information Systems, College of Economics and Political Science, Sultan Qaboos University, Al Khoudh, Muscat P.C. 123, Oman
[4] School of Computing and Creative Technologies, College of Arts, Technology and Environment (CATE), University of the West of England, Frenchay Campus, Coldharbour Lane, Bristol BS16 1QY, UK
[5] Department of Information Technology, University of Human Development, Sulaymaniyah 0778-6, Iraq
[6] Pattern Recognition and Machine Learning Lab, Gachon University, 1342 Seongnamdaero, Sujeonggu, Seongnam 13120, Republic of Korea
[7] Institute of Software Design and Development, Obuda University, 1034 Budapest, Hungary
* Correspondence: mehdi@gachon.ac.kr (M.H.); amirhosein.mosavi@stuba.sk (A.H.M.)

Abstract: Radio Frequency Identification (RFID) technology is a critical part of many Internet of Things (IoT) systems, including Medical IoT (MIoT) for instance. On the other hand, the IoT devices' numerous limitations (such as memory space, computing capability, and battery capacity) make it difficult to implement cost- and energy-efficient security solutions. As a result, several researchers attempted to address this problem, and several RFID-based security mechanisms for the MIoT and other constrained environments were proposed. In this vein, Wang et al. and Shariq et al. recently proposed CRUSAP and ESRAS ultra-lightweight authentication schemes. They demonstrated, both formally and informally, that their schemes meet the required security properties for RFID systems. In their proposed protocols, they have used a very lightweight operation called $Cro(\cdot)$ and $Rank(\cdot)$, respectively. However, in this paper, we show that those functions are not secure enough to provide the desired security. We show that $Cro(\cdot)$ is linear and reversible, and it is easy to obtain the secret values used in its calculation. Then, by exploiting the vulnerability of the $Cro(\cdot)$ function, we demonstrated that CRUSAP is vulnerable to secret disclosure attacks. The proposed attack has a success probability of "1" and is as simple as a CRUSAP protocol run. Other security attacks are obviously possible by obtaining the secret values of the tag and reader. In addition, we present a de-synchronization attack on the CRUSAP protocol. Furthermore, we provide a thorough examination of ESRAS and its $Rank(\cdot)$ function. We first present a de-synchronization attack that works for any desired $Rank(\cdot)$ function, including Shariq et al.'s proposed $Rank(\cdot)$ function. We also show that $Rank(\cdot)$ does not provide the desired confusion and diffusion that is claimed by the designers. Finally, we conduct a secret disclosure attack against ESRAS.

Keywords: medical wireless sensor network; ultra-lightweight; secret disclosure attack; $Cro(\cdot)$ function; $Rank(\cdot)$ function

MSC: 68P25; 94A62

check for updates

Citation: Servati, M.R.; Safkhani, M.; Ali, S.; Hussain Malik, M.; Hassan Ahmed, O.; Hosseinzadeh, M.; Mosavi, A.H.; Cryptanalysis of Two Recent Ultra-Lightweight Authentication Protocols. *Mathematics* **2022**, *10*, 4611. https://doi.org/10.3390/math10234611

Academic Editors: Xiaowei Huang, Wenjie Ruan and Xingyu Zhao

Received: 12 October 2022
Accepted: 18 November 2022
Published: 5 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The fundamental concept underlying the Internet of Things is that any device can be technologically enhanced to transform into a computing device and communicate with its surroundings autonomously and in real-time. This vision is becoming a reality due to the exponential growth in the number of IoT-connected devices around the world. In other words, IoT is a cutting-edge technology that aims to connect a large number of smart devices to the Internet. Because of the many IoT applications in areas such as smart things, commuting, and healthcare systems, IoT has evolved into an indispensable aspect of our daily lives. Each smart device has a sensor that allows it to sense, collect, and communicate data about its surroundings. Each device connects to the Internet, uses a unique identifier as the name, and sends data from one location to another. This interconnected system has many advantages to enhancing traditional ecosystems. The use of sensors in medical devices, for example, has a number of benefits, including remote and ongoing patient health surveillance and real-time illness treatment, which lowers healthcare costs and raises the standard of living for the elderly and children. The Medical Internet of Things (MIoT) employs an intelligent system that allows devices to collect patient data and send it to a secure cloud-based platform where it can be saved, processed, and evaluated. In addition to storing the information of numerous patients, these systems recommend a real-time evaluation of the patient's stored information in order to improve the effectiveness of healthcare systems. A significant number of corporations are making significant investments in the healthcare sector as a result of the incorporation of IoT in healthcare products. On the other hand, in this example, the patient's data are incredibly important and critical. Hence, using it improperly could put the patient in jeopardy or perhaps bring the entire system to a halt. Consequently, when these smart medical devices are detecting and transferring data, it is essential to use suitable security mechanisms to protect those data. As a result, the sender and receiver's identities must be verified using a security protocol known as the authentication protocol.

Chien et al. [1] categorized the four types of RFID authentication protocols as follows:

- **Full-fledged protocols**: These protocols should support common cryptographic components such as symmetric and asymmetric encryption functions, as well as other one-way cryptography functions. For example, they can support time-consuming primitives, such as Elliptic Curve Encryption (ECC) and RSA.
- **Simple protocols**: Support for one-way hash functions and symmetric encryption and the ability to generate pseudo-random numbers on tags are required for this category of protocols.
- **Lightweight protocols**: This category contains protocols that can generate pseudo-random numbers for tags and have simple operations, such as the Cyclic Redundancy Code (CRC) checksum, which is simpler than the one-way hash function.
- **Ultra-lightweight protocols**: This class of protocols allows only bit-wise logical operations on tags, e.g., bit-wise XOR, AND, OR, rotation, and so on. In addition, the tag's side could include a pseudo-random number generator.

Recently, Wang et al. [2] proposed a new ultra-lightweight RFID authentication protocol based on a new ultra-lightweight function called $Cro(\cdot)$ for MIoT devices. They also showed their scheme is appropriate for healthcare systems and meets the security criteria (such as consistency, synchronization, and tag anonymity) that are required for RFID systems. However, in this research, we present a de-synchronization attack against it that may be used against any chosen $Cro(\cdot)$ function in this scheme. It reveals a vulnerability in the way their protocol was designed. We also show that since the building block of their protocol, i.e., $Cro(\cdot)$, is linear and reversible, it has important security pitfalls. Then, we use the security vulnerability of $Cro(\cdot)$ to conduct a secret disclosure attack with the success probability of one and complexity of one protocol's execution and conducting some offline computations. In the same line, Shariq et al. [3] also proposed a new ultra-lightweight RFID authentication protocol called ESRAS based on a new ultra-lightweight operation called $Rank(\cdot)$ for low-cost tags. They also showed their scheme provides the expected security criteria for RFID systems. In this paper, we demonstrate an efficient de-synchronization attack against this protocol and show that the proposed $Rank(\cdot)$ function is not a good choice to provide the expected diffusion and confusion in a cryptographic protocol. We also presented a secret disclosure attack against Shariq et al.'s protocol.

1.1. Main Contribution

The following are this paper's contributions:

1. Presenting a de-synchronization attack against Wang et al.'s protocol called CRUSAP that may be used against any chosen $Cro(\cdot)$ function.
2. Presenting the security pitfall of $Cro(\cdot)$ operation, which was recently proposed by Wang et al. They used $Cro(\cdot)$ as a building operation of their proposed protocol, i.e., CRUSAP.
3. Implementing a secret disclosure attack against CRUSAP, both theoretically and practically.
4. Proposing an efficient de-synchronization attack against Shariq et al.'s scheme, ESRAS, which works for any desired $Rank(\cdot)$ function.
5. Analyzing the details of the proposed $Rank(\cdot)$ and showing that it is equivalent to the rotation function in reality and does not provide a high level of security.
6. We also propose a full secret disclosure attack against ESRAS.

1.2. Paper Organization

The remainder of the paper is structured as follows: The related works are reviewed in Section 2 and the explanation of the $Cro(\cdot)$ operation and Wang et al.'s protocol, i.e., CRUSAP, is included in Section 3. In Section 4.1, we explain the de-synchronization attack against CRUSAP. The security pitfall of $Cro(\cdot)$ is explained in Section 4.2, and then using that flaw, the Wang et al. protocol is subjected to our proposed secret disclosure attack in Section 4.3. We use the results of simulations in Section 4.4 to confirm that our proposed secret disclosure attack is practical and doable. The ESRAS protocol is described in Section 5 and its security pitfalls, including its vulnerability against de-synchronization and secret disclosure attacks and weakness of the cryptographic properties of $Rank(X,Y)$, are explained in detail in Section 6. We conclude the paper in Section 7 with concluding remarks.

2. Related Work

An authentication protocol is said to be rotation-based if a significant proportion of operations conducted on the parties are extremely lightweight functions, such as bitwise AND, OR, XOR, and rotations without the use of cryptographic primitives. The SASI protocol [1] is an RFID authentication system based on rotation functions, and several researchers have examined the security of these protocols. For instance, ref. [4] demonstrates that the protocols detailed in [1,5] are suspicious to de-synchronization and secret disclosure attacks. Tewari and Gupta [6] suggested another ultra-lightweight authentication scheme based on XOR and Rotation operations, but some research such as [7–9] showed that this scheme [6] is vulnerable to de-synchronization and secret disclosure attacks. Furthermore, ref. [7] introduced a modified version of Tewari and Gupta's [6] scheme, while [10] illustrated that their protocol is not secure enough against secret disclosure, de-synchronization, and man-in-the-middle attacks. In 2017, Fan et al. [11] proposed another ultra-lightweight authentication scheme named ULRAS, which uses the specific bit operation called the RR method. However, ref. [12] showed that their protocol is not secure against secret disclosure attacks and proposed a modified version of [11]. After that, ref. [13] proved that the proposed protocols in [11,12] suffer from de-synchronization attacks. In this line, another RFID authentication scheme using the hash function and bitwise operations was developed by [14]. However, [15] examined the security of this protocol and demonstrated that it contains security and privacy flaws. Thus, the researchers then made an effort to enhance the security of the scheme [14] by retaining a minimal level of computational cost in the database and offering a new secure lightweight protocol. Ref. [16] proposed a double authentication scheme via secret sharing for low-cost RFID tags, while [17] showed that their protocol does not withstand replay and de-synchronization attacks. Furthermore, ref. [18] proposed a lightweight authentication scheme for cloud environments, but [19] proved that their protocol is suspicious of anonymity and impersonation attacks. Ref. [20]

proposed an authentication scheme named URAP, and their protocol is secure against a wide range of attacks.

3. CRUSAP

In this section, first, the notations used in the paper are described. After that, the $Cro(\cdot)$ function is explained, and then the CRUSAP, in which the $Cro(\cdot)$ function is used as the main function, is described.

3.1. Notations

Throughout the paper, we used notations represented in Table 1.

Table 1. Notations used in this paper.

Symbol	Description
L	The Length of secret key and message
M_i	The i^{th} message was produced by one of three entities involved in the authentication phase
TID	The tag's distinct static identifying ID
$RIDS$	The reader index pseudo ID for the current session
$TIDS$	The tag index pseudo ID in the current session
$TIDS^{old}$	The tag index pseudo ID from the previous session
$TIDS^{new}$	The tag index pseudo ID in the next session
n_1, n_2, n_3	Random numbers
K_R	The shared secret key between the reader and the cloud server in the current session
K_T	A shared key between the tag and the cloud server
K_T^{old}	The previous shared key between the tag and the cloud server
K_T^{new}	The most recent shared key between the tag and the cloud server
K_1, K_2, K_3	Subkeys of K_T
$Rot(X,Y)$	The left rotation of X in the amount of Y used in the CRUSAP
$RoR(X,Y)$	The right rotation of X in the amount of Y used in the security analysis of CRUSAP; $RoR(Rot(X,Y),Y) = X$;
$Cro(X,Y)$	The proposed function in [2]
$\oplus$	The bitwise Exclusive-OR operation
$PRNG$	The generator of pseudo random numbers
IDT	The mapping table in the cloud server
$\|$	The strings concatenation operation
IDS^i	The tag's pseudonym in the i^{th} session
$rank(X)$	The number of bits of X that are 1
$nullity(X)$	The number of bits of X that are 0
$Rot(X,Y)$	The left rotation of string X in amount of $rank(Y)$ in the ESRAS
$Groupping(X)$	The string X is divided into $X_1\|X_2$ based on $rank(X)$, and this partitioning is continued as far as $\lvert X_i\rvert > T_h$, where T_h is a threshold value and suggested to be greater than 5 in the ESRAS
$Swapping(X')$	Assuming that the string X' has been partitioned based on $rank(X')$ and $nullity(X')$ into $X'_1\|X'_2$ then $Swapping(X'_1\|X'_2) = X'' = X'_2\|X'_1$
$Swapping^{-1}(\cdot)$	The inverse of $Swapping(\cdot)$
$Rank(X,Y)$	The special function used in the ESRAS
R_1^i	The random value generated in i^{th} session of ESRAS by the reader
B_L , B_R	The left and right halves of B, respectively
$\sim X$	The bitwise complement of the string X

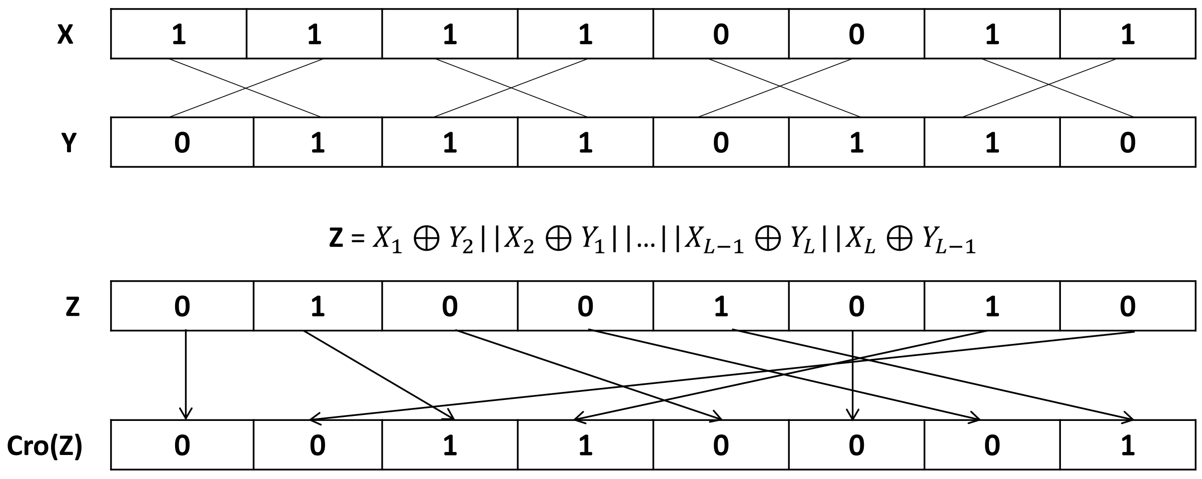
3.2. $Cro(\cdot)$ Function

Wang et al. have claimed that $Cro(\cdot)$ is a cryptographic operation that does not imposes a burden on tags and provides protocol security.In this paper, and in Section 4.3, we show that this function is very vulnerable against security attacks and the secret values used in its calculation can be easily obtained. Here, we describe the $Cro(\cdot)$ operation. To describe $Cro(\cdot)$, at first, a new bit operation called bit-wise crossover is declared by [2]. Given X, Y, and Z are three L bit binary strings (where L represents an even number) as follows:

$$X = x_1x_2x_3\ldots x_L; x \in \{0,1\}, i = 1,2,3,\ldots,L$$
$$Y = y_1y_2y_3\ldots y_L; y \in \{0,1\}, i = 1,2,3,\ldots,L$$
$$Z = z_1z_2z_3\ldots z_L; z \in \{0,1\}, i = 1,2,3,\ldots,L$$

The bit-wise crossover rearrangement operation is divided into two steps. The processes that perform adjacent parity XOR on the strings X and Y are executed first. In the second step, the output produced by the first step's operation is subjected to a bit-wise rearrangement operation. In the following, these two steps are described in more detail:

1. **XOR operation on a neighboring even–odd crossing XOR operation:** This particular operation process entails performing an XOR operation on the value of the odd bit of X and the value of the next even bit of Y, as well as an XOR operation on the value of the even bit of X and the value of the next odd bit of Y. If i is odd, Z_i is computed as $Z_i = X_i \oplus Y_{i+1}$, and if i is even, Z_i is computed as $Z_i = X_i \oplus Y_{i-1}$. After completing this phase, the binary string Z can be represented as $Z = Z_1\|Z_2\|\ldots\|Z_i\|\ldots\|Z_L$, which equals to $Z = X_1 \oplus Y_2\|X_2 \oplus Y_1\|\ldots\|X_{L-1} \oplus Y_L\|X_L \oplus Y_{L-1}$.
2. **Self-combination crossover rearrangement procedure:** At this stage, the bits resulting from XOR of X and Y based on the relations explained above, represented by Z, are rearranged according to the following pattern and form the final output of the operation, which is $Cro(X,Y)$. Given X and Y are 8-bit strings, $Cro(X,Y)$ is computed as $Cro(X,Y)_0 = Z_3$,$Cro(X,Y)_1 = Z_4$,$Cro(X,Y)_2 = Z_2$,$Cro(X,Y)_3 = Z_5$,$Cro(X,Y)_4 = Z_1$,$Cro(X,Y)_5 = Z_6$, and $Cro(X,Y)_7 = Z_7$. Figure 1 illustrates a recap of these procedures.



X
1 1 1 1 0 0 1 1
Y
0 1 1 1 0 1 1 0
Z = $X_1 \oplus Y_2||X_2 \oplus Y_1||...||X_{L-1} \oplus Y_L||X_L \oplus Y_{L-1}$
Z
0 1 0 0 1 0 1 0
Cro(Z)
0 0 1 1 0 0 0 1

Figure 1. $Cro(\cdot)$ computation procedure.

3.3. Protocol Description

CRUSAP includes three main entities, tags, (mobile) readers, and the cloud server. Following [2], Section 3.2, both channels between the reader and the tag and the reader and the server are wireless and insecure. It is worth noting the cloud server is connected to a database over the secure channel. This protocol is briefly reviewed in this section in two phases: Registration Phase and Authentication Phase.

Registration Phase: In this phase of CRUSAP, the cloud server stores the shared secret and information of tags and readers, including $TIDS^{old}$, $TIDS^{new}$, $RIDS$, and a mapping table IDT, which uses $TIDS\|RIDS$ as an index to find the related K_T and K_R necessary for verification. The tag also stores $TIDS$ and $K_T(K_1, K_2, K_3)$. Moreover, the reader stores $RIDS$ and K_R. It is worth noting that the mapping table IDT will keep the index $TIDS^{old}\|RIDS$ and its content during the previous round of verification.

Authentication Phase: To authenticate a legitimate tag in CRUSAP, the process is as follows:

1. The reader transmits a "Hello" to the tag and asks for verification.
2. As a response, the tag sends its $TIDS$ to the reader.
3. Following receipt of the tag response, the reader generates a random number n_1 using $PRNG$, then computes a message $M_1 = Rot(Cro(RIDS, K_R), n_1)$ and transmits it, along with the $TIDS$, n_1, and $RIDS$, to a cloud server over a public channel.
4. The $TIDS\|RIDS$ will be queried to the database by the cloud server. The corresponding shared key information K_T^{new} and K_R can be obtained if the match is successful. If not, the database uses the previous version of $TIDS^{old}\|RIDS$ and tries to match it. If it finds a matching, the server receives the crucial data K_T^{old} and K_R from the IDT, which corresponds to $TIDS^{old}$ and $RIDS$; otherwise, the cloud server will deem this tag to be invalid and will end the verification process. In the case of successful matching, a confirmation message $M_2 = Rot(Cro(RIDS \oplus n_1, K_R), K_R)$ is computed by the cloud server and sent to the reader over a public channel.
5. The reader verifies the received M_2 to successfully authenticate the cloud server and tag. In addition, the reader produces another random number n_2 using $PRNG$, which is used to compute $M_3 = Rot(Cro(TIDS, K_1), K_2) \oplus n_2$ and $M_4 = Cro(Rot(TIDS, K_2 \oplus n_2), K_1)$ values. After that, the reader sends M_3 and M_4 to the tag over a public channel.
6. The tag extracts the random number n_2' from M_3 as $n_2' = Rot(Cro(TIDS, K_1), K_2) \oplus M_3$ and verifies whether $M_4 \overset{?}{=} Cro(Rot(TIDS, K_2 \oplus n_2'), K_1)$ to authenticate the reader. Next, the tag calculates and transmits the message $M_5 = Cro(Rot(K_1 + n_2, TIDS), K_2)$ to the reader over a public channel. If authentication failed, the tag refuses to authenticate it, and the process will end.
7. Once M_5 is received, the reader verifies whether $M_5 \overset{?}{=} Cro(Rot(K_1 + n_2, TIDS), K_2)$ to authenticate the tag and update its local $RIDS$ and K_R. Then, using $PRNG$, the reader generates a random number n_3, calculates messages $M_6 = Rot(Cro(K_3, n_2), K_2) \oplus n_3$, $M_7 = Cro(Rot(K_3, K_1 \oplus n_3), n_2)$, and sends $M_6\|M_7$ and $M_3\|M_6\|M_7$ to the tag and cloud server, respectively, over a public channel. The reader also carries out an updating step as $RIDS = Rot(Cor(RIDS \oplus K_1, K_2 \oplus n_1), K_3)$ and $K_R = Rot(Cor(K_R, K_1), n_1 \oplus K_2)$.
8. The cloud server also extracts n_2' from M_3 and extracts n_3' from M_6, respectively, as $n_2' = Rot(Cro(TIDS, K_1), K_2) \oplus M_3$ and $n_3' = Rot(Cro(K_3, n_2'), K_2) \oplus M_6$ and verifies whether $M_7 \overset{?}{=} Cro(Rot(K_3, K_1 \oplus n_3'), n_2')$ to authenticate the reader and the tag and start the update phase if its computed M_7' equals the received M_7. The updating includes these computations: $TIDS^{old} = TIDS$, $TIDS^{new} = Rot(Cor(TIDS^{old} \oplus K_2, K_3 \oplus n_2'), K_1 \oplus n_3')$, $K_1^{new} = Rot(Cor(K_1, n_3'), n_2' \oplus K_2)$, $K_2^{new} = Rot(Cor(K_2, n_2'), n_3' \oplus K_3)$, $K_3^{new} = Rot(Cor(K_3, n_2'), n_3' \oplus K_1)$, $RIDS^{new} = Rot(Cor(RIDS \oplus K_1, K_2 \oplus n_1'), K_3)$, and $K_R^{new} = Rot(Cor(K_R, K_1), n_1' \oplus K_2)$.
9. Upon the receipt of $M_6\|M_7$, the tag extracts n_3' from M_6 as $n_3' = Rot(Cro(K_3, n_2'), K_2) \oplus M_6$ and verifies whether $M_7 \overset{?}{=} Cro(Rot(K_3, K_1 \oplus n_3'), n_2')$ to authenticate the reader and carry out the update phase as: $TIDS = Rot(Cor(TIDS^{old} \oplus K_2, K_3 \oplus n_2'), K_1 \oplus n_3')$, $K_1 = Rot(Cor(K_1, n_3'), n_2' \oplus K_2)$, $K_2 = Rot(Cor(K_2, n_2'), n_3' \oplus K_3)$, $K_3 = Rot(Cor(K_3, n_2'), n_3' \oplus K_1)$.

4. Security Analysis of CRUSAP

In this part, we first provide a de-synchronization attack that may be used against any chosen $Cro(.)$ function that reveals a vulnerability in the way CRUSAP was designed. The security flaw in the $Cro(\cdot)$ function will then be discussed. Then, a secret disclosure attack against CRUSAP is described to exploit that flaw.

4.1. De-Synchronization Attack

CRUSAP updates the tag's index after each successful session to avoid traceability and also provides forward secrecy. However, the reader is the only party that contributes to the protocol's exchanged messages' randomness. Hence, it is possible to apply the proposed attack by Safkhani et al. [17] on this protocol as follows, assuming that the current records of the tag is $(TIDS^i, K_1^i, K_2^i, K_3^i)$.

1. The adversary eavesdrops a session between the tag and the reader and stores $TIDS^i$, $M_3^i = Rot(Cro(TIDS^i, K_1^i), K_2^i) \oplus n_2^i$, $M_4^i = Cro(Rot(TIDS^i, K_2^i \oplus n_2^i), K_1^i)$, $M_5^i = Cro(Rot(K_1^i + n_2^i, TIDS^i), K_2^i)$, $M_6^i = Rot(Cro(K_3^i, n_2^i), K_2^i) \oplus n_3^i$ and $M_7^i = Cro(Rot(K_3^i, K_1^i \oplus n_3^i), n_2^i)$, where n_1^i, n_2^i, and n_3^i are random values generated in i^{th} session by the reader. However, the session is terminated by blocking M_6^i and M_7^i, which means that the tag will not update its records. Hence, the tag record is still $(TIDS^i, K_1^i, K_2^i, K_3^i)$, but the cloud server has $(TIDS^i, K_1^i, K_2^i, K_3^i)$ and $(TIDS^{i+1}, K_1^{i+1}, K_2^{i+1}, K_3^{i+1})$, where, for example, $TIDS^{i+1} = Rot(Cro(TIDS^i \oplus K_2^i, K_3^i \oplus n_2^i), K_1^i \oplus n_3^i)$.
2. The adversary allows another session between the tag and the reader, where the communicated messages are $TIDS^i$, M_3^{i+1}, M_4^{i+1}, M_5^{i+1}, M_6^{i+1}, and M_7^{i+1}, which are computed using n_1^{i+1}, n_2^{i+1}, and n_3^{i+1}, which are random values generated in $i+1^{th}$ session by the reader. However, the session is again terminated by blocking M_6^{i+1} and M_7^{i+1}. It means that the tag will not update its records. Hence, the tag record is still $(TIDS^i, K_1^i, K_2^i, K_3^i)$, but the cloud server has $(TIDS^i, K_1^i, K_2^i, K_3^i)$ and $(TIDS^{i+2}, K_1^{i+2}, K_2^{i+2}, K_3^{i+2})$, where, for example, $TIDS^{i+2} = Rot(Cro(TIDS^i \oplus K_2^i, K_3^i \oplus n_2^{i+1}), K_1^i \oplus n_3^{i+1})$.
3. The adversary impersonates the reader toward the tag based on the eavesdropped messages from Step 1 as follows:
   (a) The adversary transmits a message ("hello") to the tag and asks for another round of verification;
   (b) As a response, the tag sends its $TIDS^i$;
   (c) Following receipt of the tag response, the adversary sends the stored M_3^i and M_4^i to the tag;
   (d) The tag verifies the received M_3^i and M_4^i and computes $M_5^i = Cro(Rot(K_1^i + n_2^i, TIDS^i), K_2^i)$ to the reader/adversary;
   (e) The adversary returns the stored local M_6^i and M_7^i to the tag;
   (f) The tag authenticates the adversary as a legitimate reader and updates its records based on n_1^i, n_2^i, and n_3^i. Hence, the tag record is $(TIDS^{i+1}, K^{+1})$, but the cloud server has $(TIDS^i, K_1^i, K_2^i, K_3^i)$ and $(TIDS^{i+2}, K_1^{i+2}, K_2^{i+2}, K_3^{i+2})$, where, for instance, $TIDS^{i+1} = Rot(Cro(TIDS^i \oplus K_2^i, K_3^i \oplus n_2^i), K_1^i \oplus n_3^i)$, while $TIDS^{i+2} = Rot(Cro(TIDS^i \oplus K_2^i, K_3^i \oplus n_2^{i+1}), K_1^i \oplus n_3^{i+1})$.

At the end of the above attack, the cloud server's records for the tags $(TIDS, K)$ does not match the stored record in the tag's side with a high probability; hence, they have been de-synchronized. The attack complexity is just eavesdropping/impersonating three sessions of the protocol, which shows that the proposed attack does not just have a high chance of success but also a high efficiency.

4.2. $Cro(.)$ Security Analysis

In this section, we concentrate on $Cro(X, Y)$ and show how by having the output of this function and one of its inputs, e.g., X, the adversary can obtain the other input, i.e., Y. For this purpose, it is enough to carry out the steps shown below in order:

1. As previously stated, the $Cro(X, Y)$ relation can be used to calculate the value of Z. As mentioned before, Z is the result of applying XOR to bits of X and Y, which has

been converted to $Cro(X,Y)$ by an especial rearrangement. Therefore, we can easily achieve Z by changing the bit positions of the $Cro(X,Y)$ relation according to the definition stated in Section 3.2.

2. We also know that an XOR relationship can be retrieved if two specific values are known. In other words, if we have the Z and also one of the inputs of the $Cro(X,Y)$ function, e.g., X, one can obtain the other input of the $Cro(X,Y)$ function i.e., Y. In other words, in the $Cro(X,Y)$ function, Y different bits are calculated based on $Y_i = Z_{i+1} \oplus X_{i+1}$. As a result, Figure 2 shows how this process is completed.

In the next section of this paper, we will present a secret disclosure attack on CRUSAP based on this weakness of $Cro(\cdot)$.

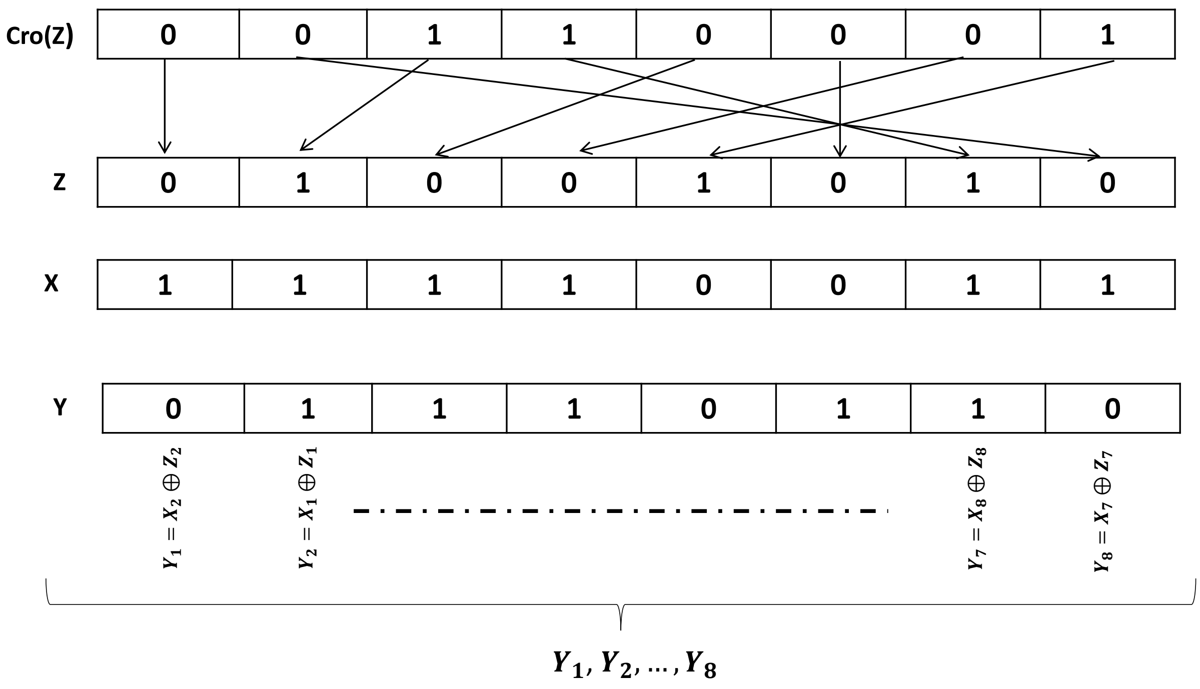


Figure 2. $Cro(.)$ security analysis.

4.3. Secret Disclosure Attack

Secret disclosure attack is a powerful security attack in which the adversary tries to discover one or more secret values used in the protocol. It is obvious that after applying the full secret disclosure attack, it is possible to perform many other attacks, such as impersonating one of the parties involved in the protocol and so on.

Secret Disclosure Attack on CRUSAP

In this section, we demonstrate how Wang et al.'s protocol is vulnerable to a secret disclosure attack. The proposed secret disclosure attack runs in two phases:

Learning phase: In this phase of the attack, the adversary eavesdrops the transferred messages over public channels in one run of CRUSAP and retrieved the exchanged messages in CRUSAP such as M_1, $RIDS$, n_1, and $M_2 \| K_T \oplus K_R$. In other words, all the required information for the attack is transferred over a public channel and can be captured by the adversary.

It is worth noting that the messages are transmitted in the insecure channel as they are, and everyone, including the adversary, can obtain those messages. On the other hand, a secure channel is a channel in which the adversary or any unauthorized person has no access to the messages exchanged in this channel and cannot obtain information about them. The channels that are used in the registration phase are usually of the secure type, and the channels that are usually used in the authentication phase are of the insecure type. In this section, we are facing an insecure channel used in the authentication phase of CRUSAP, in which the adversary can easily eavesdrop and obtain all the exchanged messages in this channel.

Secret disclosure phase: For this phase of the attack, it is enough if the adversary does as follows:

1. Given $M_1 = Rot(Cro(RIDS, K_R), n_1)$ and n_1, conduct $RoR(M_1, n_1)$, which equals to $Cro(RIDS, K_R)$. For the simplicity, the value of $Cro(RIDS, K_R)$ is called B, i.e., $B = Cro(RIDS, K_R)$.
2. Given $RIDS$ and $B = Cro(RIDS, K_R)$, it can be easily seen that the secret value of K_R can be calculated after rearranging B using the equation introduced in Section 3. Figure 3 shows an example of the implementation of our proposed secret disclosure attack against CRUSAP for $L = 8$ bit values.
3. Since the adversary retrieved $K_T \oplus K_R$ message in the learning phase of the attack, now with disclosing K_R, it is possible to retrieve K_T as $K_T = K_T \oplus K_R \oplus K_R$.
4. By obtaining K_R and K_T and also having $RIDS$ and $TIDS$ from the exchanged messages, other types of attacks can be applied to the protocol. The complexity of the attack described in this section is only one run of the protocol, and the adversary can perform this attack with a success probability of one.

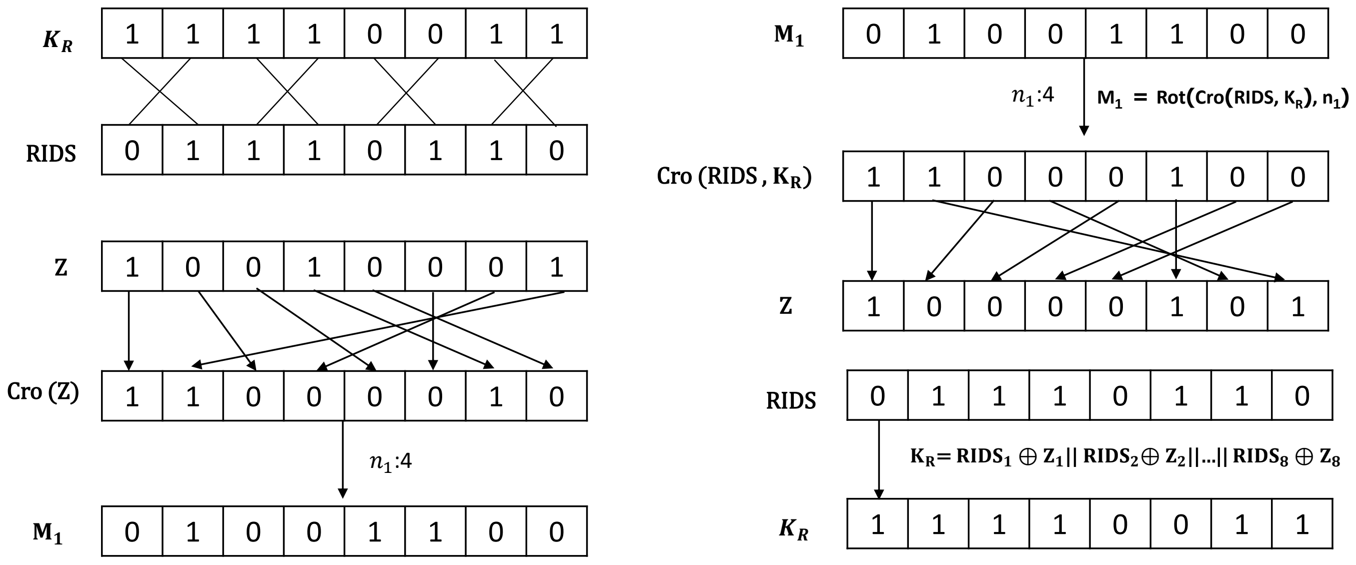


Figure 3. An example for proposed secret disclosure attack against CRUSAP.

4.4. Implementation of the Proposed Secret Disclosure Attack against CRUSAP with `C#`

(1) Calculation of CRUSAP main exchanged messages:

In this section, given $L = 32$ bits for $RIDS$ and K_R, we can compute CRUSAP exchanged messages, such as M_1, as follows:

1. $RIDS = 0 1 1 1 0 1 1 0, 0 1 1 1 0 1 1 0, 0 1 0 0 0 1 1 1, 1 1 0 0 0 1 1 0;$
2. $K_R = 1 0 1 0 0 1 0 0, 0 1 1 0 0 0 0 1, 1 0 1 0 0 1 0 0, 0 1 1 0 0 0 0 1;$
3. $K_T = 1 0 0 0 1 1 0 1, 0 1 1 1 1 1 1 0, 0 1 1 0 0 1 1 1, 1 1 0 0 1 0 0 1;$
4. $Z = Special \ \ XOR(RIDS, K_R) = 0 0 1 0 1 1 1 0, 1 1 1 0 0 1 0 0, 0 0 0 1 1 1 1 1, 0 1 0 1 0 1 0 0;$
5. $Cro(RIDS, K_R) = 0 1 1 1 1 1 0 0, 0 0 1 1 0 0 0 0, 0 0 1 1 1 1 1 1, 0 1 0 1 0 1 0 0\,;$
6. $n_1 = 4;$
7. $M_1 = Rot(Cro(RIDS, K_R), n_1) = 0 1 0 0 0 1 1 1, 1 1 0 0 0 0 1 1, 0 0 0 0 0 0 1 1, 1 1 1 1 0 1 0 1;$
8. $K_T \oplus K_R = 0 0 1 0 1 0 0 1, 0 0 0 1 1 1 1 1, 1 1 0 0 0 0 1 1, 1 0 1 0 1 0 0 0.$

In the following step, we demonstrate how to calculate the value of K_R and K_T following the secret disclosure attack explained in Section 4.3.

(2) Disclosure of K_R and K_T :

Given $L = 32$ bits values for M_1 and $RIDS$ and a known n_1, it can be easily seen that we can compute K_R and K_T following the steps described in Section 4.3. It worth noting that M_1, $RIDS$, $K_T \oplus K_R$, and n_1 are accessible from the exchanged messages in the insecure channel of the protocol, which we assume are as follows:

1. $M_1 =$ 0 1 0 0 0 1 1 1, 1 1 0 0 0 0 1 1, 0 0 0 0 0 0 1 1, 1 1 1 1 0 1 0 1;
2. $K_T \oplus K_R =$ 0 0 1 0 1 0 0 1, 0 0 0 1 1 1 1 1, 1 1 0 0 0 0 1 1, 1 0 1 0 1 0 0 0;
3. $RIDS =$ 0 1 1 1 0 1 1 0, 0 1 1 1 0 1 1 0, 0 1 0 0 0 1 1 1, 1 1 0 0 0 1 1 0;
4. $n_1 = 4$; then, we can carry out the following:
5. Computing $RoR(M_1, n_1) =$ 0 1 1 1 1 1 0 0, 0 0 1 1 0 0 0 0, 0 0 1 1 1 1 1 1, 0 1 0 1 0 1 0 0$= Cro(RIDS, K_R)$;
6. Obtaining Z from the rearrangement of $Cro(RIDS, K_R)$ as 0 0 1 0 1 1 1 0, 1 1 1 0 0 1 0 0, 0 0 0 1 1 1 1 1, 0 1 0 1 0 1 0 0 ;
7. Then, since $Z = Special\ \ XOR(RIDS, K_R)$, given Z and $RIDS$, we can compute K_R as 1 0 1 0 0 1 0 0, 0 1 1 0 0 0 0 1, 1 0 1 0 0 1 0 0, 0 1 1 0 0 0 0 1;
8. Then, K_T is calculated as $K_T = K_T \oplus K_R \oplus K_R =$ 1 0 0 0 1 1 0 1, 0 1 1 1 1 1 1 0, 0 1 1 0 0 1 1 1, 1 1 0 0 1 0 0 1.

It can be seen the retrieved values for K_R and K_T, respectively, equal our assumptions of K_R and K_T. Therefore, these implementations also showed that our proposed secret disclosure attack is practical and feasible.

5. ESRAS

ESRAS [3] is another ultra-lightweight authentication protocol that was recently proposed by Shariq et al. This scheme uses an ultra-lightweight operation called $Rank(X, Y)$ as the core of non-linearity to achieve desired diffusion and confusion, where X and Y are strings of bits as follows:

$$\begin{aligned} X =& x_1 \| x_2 \| \dots \| x_n \\ Y =& y_1 \| y_2 \| \dots \| y_n \end{aligned}$$

We describe this operation in this section first because it is crucial to understand the functionality of ESRAS. Through description, similar to the designers, we use the following strings for X and Y:

$$\begin{aligned} X =& 11000111101011101100011110011011 \\ Y =& 10111101110101100011110111000010 \end{aligned}$$

$Rank(X, Y)$ uses several other operations as follows:

- $rank(X)$: returns the number of bits of X that are 1, e.g., for the provided example $rank(X) = 20$ and $rank(Y) = 19$;
- $nullity(X)$: returns the number of bits of X that are 0, e.g., for the provided example $nullity(X) = 12$ and $nullity(Y) = 13$; it is obvious $rank(x) + nullity(x) = length(x)$;
- $Rot(X, Y)$: String X is left rotated by $rank(Y)$, for the given example $rank(Y) = 19$ and $Rot(X, Y) = 00111100110111100011110101110110$;
- $Groupping(X)$: The string X is divided into $X_1 \| X_2$ based on $rank(X)$, and this partitioning is continued as far as $|X_i| > T_h$, where T_h is a threshold value and suggested to be greater than 5. We will discuss $Groupping(X)$ in Section 6.2 in more detail;
- $Swapping(X')$: Assuming that the string X' has been partitioned based on $rank(X')$ and $nullity(X')$ into $X'_1 \| X'_2$ then $Swapping(X'_1 \| X'_2) = X'' = X'_2 \| X'_1$.

Based on these operations, $Rank(X, Y)$ is computed as

$$\begin{aligned} Groupping(X) &= X' \\ Groupping(Y) &= Y' \\ Swapping(X') = X'' &= x_{m+1} \| x_{m+2} \| \dots \| x_n \| x_1 \| x_2 \| \dots \| x_m;\ where\ m = rank(X') \\ Swapping(Y') = Y'' &= y_{m'+1} \| y_{m'+2} \| \dots \| y_y \| y_1 \| y_2 \| \dots \| y_{m'};\ where\ m' = rank(Y') \\ Rank(X, Y) &= \sim X'' \oplus \sim Y'' \end{aligned}$$

Protocol Description

The ESRAS protocol has two entities, i.e., RFID tag and the reader–server unit, and it includes two phases, i.e., initialization phase and authentication phase. It is worth noting that the channel between the tag and the reader–server is insecure, but the reader-to-sever communication considered over a secure channel.

In the initialization phase of ESRAS, for each tag $\mathcal{T}$, an identifier ID, a tag's index IDS, and two secret keys K_1 and K_2 are generated by the manufacturer and stored in the tag's internal memory and in the server side (BS). In addition, two records for IDS are considered in BS as $IDS_{old} = IDS$ and $IDS_{new} = Null$.

The authentication phase of ESRAS is as follows, in which all messages are transferred over a public channel:

1. The reader sends a Hello message to the tag.
2. The tag in response returns its IDS.
3. The reader generates a random number R_1 and computes $A = Rank(Rot(K_1, K_2), K_1) \oplus R_1$ and $B = Rank(Rot(K_1, R_1), K_1 \oplus K_2) \oplus Rot(Rank(K_2, R_1 \oplus K_2), K_1)$ and sends $\{A \| B_{R\ or\ L}\}$ to the tag. For sending messages, B_L and B_R, respectively, denote the left and the right halves of B. If $rank(B)$ is odd, then B_L is sent; otherwise, B_R is sent.
4. Once the message is received, the tag extracts R_1 from A and verifies $B_{R\ or\ L}$. Assuming that the verification was successful, the tag computes $C = Rank(Rank(K_1 \oplus K_2, R_1), Rank(R_1, K_2)) \oplus ID$ and sends it to the reader.
5. The reader verifies the received C to authenticate the tag and update $IDS_{old} = IDS$ and $IDS_{new} = Rank(Rot(Rank(IDS \oplus R_1, K_1), K_2))$. Next, it generates a random value R_2 and calculates $D = Rank(R_1, K_1 \oplus K_2) \oplus Rank(K_1, K_2) \oplus R_2$ and $E = Rank(Rot(Rot(R_2, R_2), K_2), IDS_{new}) \oplus Rot(Rank(R_1, R_1), R_2 \oplus K_2)$. Finally, the reader sends $\{D, E_{R\ or\ L}\}$ to the tag.
6. Once the message $\{D, E\}$ is received, the tag extracts R_2 from D, computes $IDS_{new} = Rank(Rot(Rank(IDS \oplus R_1, K_1), K_2))$, and verifies $E_{R\ or\ L}$. Assuming that the verification was successful, the reader is authenticated, and IDS is updated to IDS_{new}.

6. Security Analysis of ESRAS

In this section, we provide a more detailed security analysis of ESRAS. More precisely, while the authors claimed full diffusion and confusion by the introduced component $Rank(X, Y)$, and based on it, they claimed security against de-synchronization attacks and secret disclosure attacks, we show that the $Rank(X, Y)$ does not provide the expected diffusion and confusion. In addition, we show that despite of the used $Rank(X, Y)$, ESRAS suffers from the de-synchronization attack. Finally, we apply a successful secret disclosure attack on it.

6.1. De-Synchronization Attack

ESRAS updates the tag's index after each successful session to avoid traceability and also provides forward secrecy. However, the reader is the only party that contributes to the protocol's exchanged messages randomness. Hence, it is possible to apply the proposed attack by Safkhani et al. [17] on this protocol as follows, assuming that the current index of the tag is IDS^i:

1. The adversary eavesdrops a session between the tag and the reader and stores $IDS^i, \{A\|B_{L\ or\ R}\}^i, C^i, \{D\|E_{L\ or\ R}\}^i$ but blocks $\{D\|E_{L\ or\ R}\}^i$ and does not allow the tag to update its IDS^i. Hence, the tag's records of IDS is IDS^i, but the reader has $IDS_{old} = IDS^i$ and $IDS_{new} = Rank(Rot(Rank(IDS \oplus R_1^i, K_1), K_2))$, where R_1^i is a random value generated in i^{th} session by the reader.
2. The adversary allows another session between the tag and the reader, where the communicated messages are $IDS^i, \{A\|B_{L\ or\ R}\}^{i+1}, C^{i+1}, \{D\|E_{L\ or\ R}\}^{i+1}$; however, again, the adversary blocks $\{D\|E_{L\ or\ R}\}^{i+1}$ and does not allow the tag to update its IDS^i. Hence, the tag's records of IDS is IDS^i yet but the reader has $IDS_{old} = IDS^i$ and $IDS^i_{new} = Rank(Rot(Rank(IDS \oplus R_1^{i+1}, K_1), K_2))$, where R_1^{i+1} is a random value generated in $i+1^{th}$ session by the reader.
3. The adversary impersonates the reader toward the tag based on the eavesdropped messages from Step 1 as follows, i.e., $IDS^i, \{A\|B_{L\ or\ R}\}^i, C^i, \{D\|E_{L\ or\ R}\}^i$:
   (a) The adversary sends a Hello message to the tag;
   (b) The tag returns its IDS^i;
   (c) The adversary sends $\{A\|B_{L\ or\ R}\}^i$ to the tag;
   (d) Once the message is received, the tag extracts R_1^i from A^i and verifies $B^i_{R\ or\ L}$. The verification will be successful, and the tag computes $C^i = Rank(Rank(K_1 \oplus K_2, R_1^i), Rank(R_1^i, K_2)) \oplus ID$ and sends it to the reader/adversary;
   (e) The adversary sends $\{D, E_{R\ or\ L}\}^i$ to the tag;
   (f) Once the message $\{D, E\}^i$ is received, the tag extracts R_2^i from D^i, computes $IDS^{i+2}_{new} = Rank(Rot(Rank(IDS \oplus R_1^i, K_1), K_2))$, and verifies $E^i_{R\ or\ L}$. Assuming that the verification was successful, the reader is authenticated, and IDS is updated to IDS_{new}.

At the end of the above procedure, the reader's records for the tag's index are $IDS_{old} = IDS^i$ and $IDS^i_{new} = Rank(Rot(Rank(IDS \oplus R_1^{i+1}, K_1), K_2))$, while the tag's record is $IDS^{i+2}_{new} = Rank(Rot(Rank(IDS \oplus R_1^i, K_1), K_2))$, where R_1^i and R_1^{i+1} are two independent random values generated, respectively, by the reader in i^{th} and $i+1^{th}$ sessions. Hence, with the probability of $1 - 2^{-l}$, we have $R_1^i \neq R_1^{i+1}$ and with the same probability $IDS^{i+1}_{new} \neq IDS^{i+2}_{new}$. Since the probability for $IDS^{i+1}_{new} \neq IDS^i$ is also the same, the success probability of the proposed attack is $1 - 2^{-l+1}$. The attack complexity is just eavesdropping/impersonating three sessions of the protocol, which shows that the proposed attack not only has a high chance of success but also a high efficiency. This attack contradicts the designer's claim against the protocol's security against a de-synchronization attack in [3].

6.2. On the Cryptographic Properties of $Rank(X, Y)$

As it has been mentioned already, designers also used the operation $Groupping(X) = X'$ through the computation of $Rank(X, Y)$, in which the string X is divided into $X_1\|X_2$ based on $Rank(X)$, and this partitioning is continued as far as $|X_i| > T_h$, where T_h is a threshold value and suggested to be greater than 5. To understand $Groupping(X)$, the authors provided a numerical example [3] (Figure 1) based on $T_h = 6$. Following that example, it is clear $Groupping(X) = X' = X$. The provided example for $Groupping(X) = X' = X$ in [3] (Figure 2) also confirms that $Groupping(X) = X' = X$. Hence, we can omit the description of this operation. Following this, we can state that $Rank(X, Y) =\sim X'' \oplus \sim Y''$, where $\sim X$ is the bitwise complement of the string X. On the other hand, for any bit x_i, we can state that $\sim x_i = 1 \oplus x_i$ and $\sim x_i \oplus \sim y_i = 1 \oplus x_i \oplus 1 \oplus y_i$. Hence, $Rank(X, Y) = X'' \oplus Y''$.

To accomplish $Swapping(X')$, assuming that $X' = x'_1\|x'_2\| \ldots \|x'_n$ and $rank(X') = m$, it is computed as $Swapping(X') = X'' = x'_{m+1}\|x'_{m+2}\| \ldots \|x'_n\|x'_1\|x'_2\| \ldots \|x'_m$. Hence, given that

$X' = Groupping(X) = X$ and $Swapping(X') = X'' = x_{m+1}\|x_{m+2}\|\ldots\|x_n\|x_1\|x_2\|\ldots\|x_m$ and assuming that $rank(X) = m$ and $rank(Y) = m'$, then:

$$\begin{aligned} Swapping(X) = X'' =& x_{m+1}\|x_{m+2}\|\ldots\|x_n\|x_1\|x_2\|\ldots\|x_m \\ Swapping(Y) = Y'' =& y_{m'+1}\|y_{m'+2}\|\ldots\|y_y\|y_1\|y_2\|\ldots\|y_{m'} \\ Rank(X,Y) = X'' \oplus Y' =& (x_{m+1}\|x_{m+2}\|\ldots\|x_n\|x_1\|x_2\|\ldots\|x_m)\oplus \\ & (x_{m+1}\|x_{m+2}\|\ldots\|x_n\|x_1\|x_2\|\ldots\|x_m) \end{aligned}$$

Given the definition of $Rot(X,Y)$ and $Rank(X,Y)$, we can represent $Rank(X,Y)$ as follows:

$$Rank(X,Y) = Rot(X, \sim X) \oplus Rot(Y, \sim Y)$$

It shows that $Rank(X,Y)$ does not provide the desired diffusion and confusion. More precisely, for $Rank(X,Y)$, they claimed that it provides full diffusion, while if it provides full diffusion, then any modification in input should change any bit of the output with the probability of 0.5. However, for the given $Rank(X^1,Y)$, consider the case where for the string X^1, we have $x_1x_2 = 10$; thus, changing those bits into 01 to create X^2 does not affect $rank(X^2)$ compared to $rank(X^1)$. Additionally, $Rank(X^2,Y)$ is identical to $Rank(X^1,Y)$ in all $n-2$ bits that are not affected by those bits, and those bits of $Rank(X^2,Y)$ and $Rank(X^1,Y)$ complement each other exactly. It means that $rank(Rank(X^2,Y) \oplus Rank(X^1,Y)) = 2$ with the probability of 1. In addition, from $Rank(X^2,Y) \oplus Rank(X^1,Y)$, without the knowledge of X^1 or Y, the adversary can identify $rank(X)$, which contradicts another claim of the designers in which the claimed $Rank(X,Y)$ does not leak any information related to input values.

As another property of $Rank(X,Y)$, designers claimed that it is a one-way function, and given $Rank(X,Y)$ and Y for instance, it is not possible to determine X. However, given Y, the adversary easily computes:

$$Swapping(Y) = Y'' = y_{m'+1}\|y_{m'+2}\|\ldots\|y_n\|y_1\|y_2\|\ldots\|y_{m'}$$

Given Y'', X'' is computed as $X'' = Rank(X,Y) \oplus Y''$. On the other hand, $X'' = Swapping(X) = x_{m+1}\|x_{m+2}\|\ldots\|x_n\|x_1\|x_2\|\ldots\|x_m$ and $rank(X) = rank(X'') = m$. Hence, given $X'' = x''_1\|\ldots\|x''_n$, it is possible to invert the $Swapping(\cdot)$ and determine X as follows, where $Swapping^{-1}(\cdot)$ is used to denote the inverse of $Swapping(\cdot)$:

$$Swapping^{-1}(X'') = X = x''_{n-m}\|x''_{n-m+1}\|\ldots\|x''_n\|x''_1\|x''2\|\ldots\|x''_m$$

Hence, given $Rank(X,Y)$ and Y for instance, X is determined uniquely; vice versa, given $Rank(X,Y)$ and X for instance, Y is determined uniquely, which contradicts the designers' claim in [3], Section 3.2.

6.3. Secret Disclosure Attack

While the proposed attack in Section 6.1 works for any $Rank(\cdot)$ function and shows a structural flaw in the designed ESRAS, in Section 6.2, we described undesired properties of $Rank(\cdot)$, which are used in this section to mount a more dedicated attack. The proposed attack in this section is a secret disclosure attack that reveals confidential information of a given tag. During the proposed attack, we use the fact that $rank(X) = rank(X')$, then $Rot(Y,X) = Rot(Y,X')$.

The computed message through a session of the protocol are:

$$\begin{aligned}
A =& Rank(Rot(K_1, K_2), K_1) \oplus R_1 \\
B =& Rank(Rot(K_1, R_1), K_1 \oplus K_2) \oplus Rot(Rank(K_2, R_1 \oplus K_2), K_1) \\
C =& Rank(Rank(K_1 \oplus K_2, R_1), Rank(R_1, K_2)) \oplus ID \\
IDS_{new} =& Rank(Rot(Rank(IDS \oplus R_1, K_1), K_2)) \\
D =& Rank(R_1, K_1 \oplus K_2) \oplus Rank(K_1, K_2) \oplus R_2 \\
E =& Rank(Rot(Rot(R_2, R_2), K_2), IDS_{new}) \oplus Rot(Rank(R_1, R_1), R_2 \oplus K_2)
\end{aligned}$$

and the messages are transferred over a public channel. IDS, $\{A\|B_{R\ or\ L}\}$, C and $\{D, E_{R\ or\ L}\}$ are accessible by the adversary. In addition, IDS is updated at the end of each successful authentication.

Consider the transferred messages in the i^{th} and j^{th} sessions. Given that K_1 and K_2 are constant values for each tag and provided A^i and A^j, we can state that:

$$\begin{aligned}
A^i \oplus A^j =& [Rank(Rot(K_1, K_2), K_1) \oplus R_1^i] \oplus [Rank(Rot(K_1, K_2), K_1) \oplus R_1^j] \\
=& R_1^i \oplus R_1^j
\end{aligned}$$

On the other hand, assuming that $rank(R_1^i) = rank(R_1^j)$, then $Rank(Rot(K_1, R_1^i), K_1 \oplus K_2) = Rank(Rot(K_1, R_1^j), K_1 \oplus K_2)$. In addition, if $rank(R_1^i \oplus K_2) = rank(R_1^j \oplus K_2)$ and $rank(R_1^i) = rank(R_1^j)$, then $rank(A^i \oplus A^j) = rank(B^i \oplus B^j)$.

Let us assume the adversary has stored IDS^i, $\{A^i\|B^i_{R\ or\ L}\}$, C^i, and $\{D^i, E^i_{R\ or\ L}\}$ for a session and blocked the last message to the tag. Thus, the tag's record for its index is still IDS^i. Let us assume the adversary flips two bits of A^i to achieve A'^i, e.g., a_x^i and a_y^i, such that $|x - y| = \frac{l}{2}$, where l is the parameter length. In this case, assuming that $rank(R_1^i \oplus K_2) = rank(R_1^j \oplus K_2)$ and $rank(R_1^i) = rank(R_1^j)$, then considering B'^i corresponds to A'^i, we have $rank(B_L^i \oplus B_L'^i) = 1$ and $rank(B_R^i \oplus B_R'^i) = 1$, and the probability of switching from left to right or vice versa in the required part of B^i is $\frac{2}{l}$ and remains the same with the probability of $1 - \frac{2}{l}$. Hence, if the adversary sends $\{A'^i\|B'^i_{R\ or\ L}\}$ to the tag such that A'^i is computed by flipping two bits of A^i, as mentioned above, and $B'^i_{R\ or\ L}$ is computed by flipping a chosen bit of $B'^i_{R\ or\ L}$, with the probability of $(1 - \frac{2}{l})\frac{1}{2} \times \frac{1}{2} \times \frac{1}{l/2}$, the sent message will be accepted by the tag, and the tag will return a response for C. Assuming that the tag returned a response, it means that the provided conditions were held. Hence, we can conclude that $(Rank(Rot(K_1, K_2), K_1))_i \oplus (Rank(Rot(K_1, K_2), K_1))_j = 1$, where $(Rank(Rot(K_1, K_2), K_1))_i$ and $(Rank(Rot(K_1, K_2), K_1))_j$, respectively, denote i^{th} and j^{th} bits of $Rank(Rot(K_1, K_2), K_1)$. In this way, the adversary could achieve a single bit of information related to the secret parameters. The adversary can eavesdrop more IDS, $\{A\|B_{R\ or\ L}\}$, C, and $\{D, E_{R\ or\ L}\}$ and repeat the attack to extract whole $(Rank(Rot(K_1, K_2), K_1))_i$. The expected complexity (in the term of sessions) of the attack is as follows:

$$l \times \frac{1}{(1 - \frac{2}{l})\frac{1}{2} \times \frac{1}{2} \times \frac{1}{l/2} \times l} = \frac{1}{(1 - \frac{2}{l})\frac{1}{2} \times \frac{1}{2} \times \frac{1}{l/2}}$$

Given $(Rank(Rot(K_1, K_2), K_1))_i$ and eavesdropped A values, R_1 values are achievable. Then, from that information and the given values of $B_{R\ or\ L}$ on each session, the adversary can develop a linear equation system to extract K_1 and K_2. Following it and given C, it is possible to extract ID, which completes the secret disclosure attack on ESRAS.

7. Conclusions

Over the years, many ultra-authentication schemes have been proposed; however, unfortunately, all of those protocols are not secure. These protocols are commonly vulnerable to a variety of attacks, such as secret disclosure, de-synchronization, and impersonation attacks. This paper, similar to other papers in this field, once again showed that the ultra-lightweight operations, e.g., a limited number of bitwise operations, such as AND, OR, and XOR, are not enough to design a completely safe security protocol. This is why they are linear reversible operations.

In this paper, we proposed a de-synchronization attack and a secret disclosure attack against Wang et al.'s ultra-lightweight protocol called CRUSAP, with a success probability of one. We also show security vulnerabilities of ESRAS against de-synchronization and secret disclosure attacks.

Author Contributions: M.R.S.: Conceptualization, Methodology, Validation, Writing; M.S.: Experimentation, Validation, Writing—review and editing; M.H.M.: Experimentation, Validation, review and editing; S.A.: Conceptualization, Methodology, Experimentation, Validation, Writing—review and editing; O.H.A.: Experimentation, Validation, Writing—review and editing; M.H.: Methodology, Designing, Experimentation, Validation, Supervision, Review, Funding and editing; A.H.M.: Experimentation, Designing, Validation, Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: This manuscript does not contain any studies with human participants or animals performed by any of the authors.

Data Availability Statement: For any supplementary material, please contact the corresponding authors.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

RFID	Radio Frequency IDentification
IoT	Internet of Things
MIoT	Medical Internet of Things
ECC	Elliptic Curve Cryptography
RSA	Rivest-Shamir-Adleman Public-key Encryption Algorithm
ID	The tag unique identifier
IDS	The tag's pseudonym
$RIDS$	The reader's pseudonym
CRC	Cyclic Redundancy Code checksum
PRNG	Pseudo Random Number Generator

References

1. Chien, H.Y. SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity. *IEEE Trans. Dependable Secur. Comput.* **2007**, *4*, 337–340. [CrossRef]
2. Wang, X.; Fan, K.; Yang, K.; Cheng, X.; Dong, Q.; Li, H.; Yang, Y. A new RFID ultra-lightweight authentication protocol for medical privacy protection in smart living. *Comput. Commun.* **2022**, *186*, 121–132. [CrossRef]
3. Shariq, M.; Singh, K.; Lal, C.; Conti, M.; Khan, T. ESRAS: An efficient and secure ultra-lightweight RFID authentication scheme for low-cost tags. *Comput. Netw.* **2022**, *217*, 109360. [CrossRef]
4. Ain, Q.U.; Mahmood, Y.; Mujahid, U. Cryptanalysis of mutual ultralightweight authentication protocols: SASI & RAPP. In Proceedings of the 2014 International Conference on Open Source Systems & Technologies, Lahore, Pakistan, 18–20 December 2014 ; IEEE: Piscataway, NJ, USA, 2014; pp. 136–145.
5. Tian, Y.; Chen, G.; Li, J. A new ultralightweight RFID authentication protocol with permutation. *IEEE Commun. Lett.* **2012**, *16*, 702–705. [CrossRef]
6. Tewari, A.; Gupta, B.B. Cryptanalysis of a novel ultra-lightweight mutual authentication protocol for IoT devices using RFIDtags. *J. Supercomput.* **2017**, *73*, 1085–1102. [CrossRef]
7. Wang, K.H.; Chen, C.M.; Fang, W.; Wu, T.Y. On the security of a new ultra-lightweight authentication protocol in IoT environment for RFID tags. *J. Supercomput.* **2018**, *74*, 65–70. [CrossRef]

8. Khalid, M.; Mujahid, U.; Najam-ul Islam, M. Cryptanalysis of ultralightweight mutual authentication protocol for radio frequency identification enabled Internet of Things networks. *Int. J. Distrib. Sens. Netw.* **2018**, *14*, 1550147718795120. [CrossRef]
9. Huang, S.C.; Tsai, C.W.; Hwang, T. Comment on "Cryptanalysis of a novel ultralightweight mutual authentication protocol for IoT devices using RFID tags". In Proceedings of the 2018 International Conference on Data Science and Information Technology, Madrid, Spain, 1–2 October 2018; pp. 23–27.
10. Khor, J.H.; Sidorov, M. Weakness of ultra-lightweight mutual authentication protocol for IoT devices using RFlD tags. In Proceedings of the 2018 Eighth International Conference on Information Science and Technology (ICIST), Seville, Spain, 30 June–30 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 91–97.
11. Fan, K.; Ge, N.; Gong, Y.; Li, H.; Su, R.; Yang, Y. An ultra-lightweight RFID authentication scheme for mobile commerce. *Peer-to-Peer Netw. Appl.* **2017**, *10*, 368–376. [CrossRef]
12. Aghili, S.F.; Mala, H. Security analysis of an ultra-lightweight RFID authentication protocol for m-commerce. *Int. J. Commun. Syst.* **2019**, *32*, e3837. [CrossRef]
13. Safkhani, M.; Bagheri, N.; Shariat, M. On the security of rotation operation based ultra-lightweight authentication protocols for RFID systems. *Future Internet* **2018**, *10*, 82. [CrossRef]
14. Dass, P.; Om, H. A secure authentication scheme for RFID systems. *Procedia Comput. Sci.* **2016**, *78*, 100–106. [CrossRef]
15. Gholami, V.; Alagheband, M.R. Provably privacy analysis and improvements of the lightweight RFID authentication protocols. *Wirel. Networks* **2020**, *26*, 2153–2169. [CrossRef]
16. Liu, Y.; Ezerman, M.; Wang, H. Double verification protocol via secret sharing for low-cost RFID tags. *Future Gener. Comput. Syst.* **2019**, *90*, 118–128. [CrossRef]
17. Safkhani, M.; Rostampour, S.; Bendavid, Y.; Sadeghi, S.; Bagheri, N. Improving RFID/IoT-based generalized ultra-lightweight mutual authentication protocols. *J. Inf. Secur. Appl.* **2022**, *67*, 103194. [CrossRef]
18. Fan, K.; Zhu, S.; Zhang, K.; Li, H.; Yang, Y. A lightweight authentication scheme for cloud-based RFID healthcare systems. *IEEE Netw.* **2019**, *33*, 44–49. [CrossRef]
19. Nikkhah, F.; Safkhani, M. LAPCHS: A lightweight authentication protocol for cloud-based health-care systems. *Comput. Netw.* **2021**, *187*, 107833. [CrossRef]
20. Gao, M.; Lu, Y. URAP: A new ultra-lightweight RFID authentication protocol in passive RFID system. *J. Supercomput.* **2022**, *78*, 10893–10905. [CrossRef]