Journal of Information Hiding and Multimedia Signal Processing
Ubiquitous International

Volume **7**, Number **3**, May **2016**

An Efficient Improvement on Safkhani et al.'s Hash-Based Mutual Authentication Protocol for RFID Systems

Wei-Liang Tai

Department of Information Communications
Chinese Culture University
55, Hwa-Kang Road, Yang-Ming-Shan, Taipei, Taiwan
dwl@ulive.pccu.edu.tw

Ya-Fen Chang* and Ti-Sheng Kwang

Department of Computer Science and Information Engineering
National Taichung University of Science and Technology
No.129, Sec.3, Sanmin Rd,North Dist.,Taichung, Taiwan
*Corresponding author: cyf@nutc.edu.tw; guandskun@gmail.com

Received December, 2015; revised March, 2016

Abstract. *Nowadays, plenty of radio frequency identification (RFID) systems have been proposed and used in the real world for convenience. An RFID system requires advanced properties of confidentiality, indistinguishability, forward security and mutual authentication to provide a higher security level. Safkhani et al. proposed a hash-based authentication protocol for RFID systems. Though their protocol possesses the above properties, the server's computational load is heavy. In this paper, we propose an authentication protocol for RFID systems to improve the computational efficiency with a long-term secure value and provide the same security level.*
Keywords: RFID, Authentication

1. **Introduction.** RFID is common and practical because it utilizes radio waves to transmit data and one category can work without a battery embedded. An RFID system has three main parts: RFID tag, reader and back-end server. RFID tags are used as a label for item identication or verification, readers are used to send and receive data between a tag and the back-end server, and the server maintains a database to keep tags' data or some secure value to authenticate a tag. The first RFID system for identification is used in World War II. Up to now, RFID systems are widely used in people's life such as electronic wallet, intruder alarms of buildings, and motor vehicles because an RFID system provides not only celerity but also convenience. However, for a higher security level, the above applications require advanced properties, confidentiality, indistinguishability, forward security, and mutual authentication [1, 2, 3]. Definitions of these properties are listed as follows.

Confidentiality: Data should be transmitted securely via authentication or encryption. If a tag transmits data without authentication or encryption, an attacker can analyze its owner's private information.

Indistinguishability: An attacker cannot trace a specific tag by the transmitted data. Therefore, the transmitted data should not be the same as the one transmitted previously.

Forward security: If the secret of one session is revealed, an adversary cannot trace the past location of the specific tag.

Mutual authentication: A tag, reader and the back-end server can authenticate each other for secure communication.

To ensure security and efficiency in RFID systems, many mechanisms have been proposed. In 2003, Vajda and Buttyn proposed a set of extremely lightweight tag authentication protocols for RFID systems and also provided analyses of these protocols [4]. Subsequently, Peris-Lopez et al. proposed an efficient mutual authentication protocol for RFID systems, EMAP [5]. They claimed that EMAP could offer an adequate security level for certain applications and also could be implemented by limited low-cost RFID tags. However, Li and Deng presented two effective attacks against EMAP [6]. Via de-synchronization attack, an attacker can permanently disable the authentication capability of an RFID tag. Via full disclosure attack, all secret information stored in a tag will be completely compromised. Some authentication protocols for low-cost RFID are listed in [7, 8]. In 2011, Cho et al. proposed a hash-based authentication protocol for RFID systems to overcome the privacy and forgery problems [1]. In Cho et al.'s protocol, instead of the secret value, random numbers generated by a tag and the reader are transmitted. This prevents an attacker from mounting replay attack. However, Safkhani et al. pointed out that Cho et al.'s protocol suffers from de-synchronization attack, tag impersonation attack and reader impersonation attack [9]. This results from that the updated secret value is not verified, and an attacker can tamper the transmitted data easily. Subsequently, Khedr proposed a new protocol for a low-cost RFID system [2] that provides mutual authentication between the back-end server and the tag without a secure channel. In Khedr's protocol, a tag's identity is updated after every read operation for security and synchronization, and the cost is also reasonable.

On the other hand, Cho et al. proposed a hash-based mutual authentication protocol for RFID systems [3]. Cho et al. claimed that their protocol has properties of confidentiality, indistinguishability, forward security, and mutual authentication. However, Safkhani et al. [10] showed that Cho et al.'s protocol suffers from de-synchronization attack, tag impersonation attack, and reader impersonation attack. Safkhani et al. also proposed an improvement to overcome these found weaknesses. After analyzing Safkhani et al.'s protocol, we find that the server's computational load is heavy because the server cannot identify the tag. That is, the server needs to check all entries stored in the database to find the matched one. This approach places a burden on the back-end server.

In this paper, we will propose a mutual authentication protocol based on hash function for RFID systems by improving Safkhani et al.'s protocol with a long-term secure value to have the sever identify the tag easily. The rest of this paper is organized as follows. We briefly review Safkhani et al.'s mutual authentication protocol in Section 2. Our improvement is shown in Section 3, and security and efficiency analyses are given in Section 4. Finally, some conclusions are drawn in Section 5.

2. Review of Safkhani et al.'s Authentication Protocol.

Safkhani et al. proposed an improved protocol to overcome the weaknesses of Cho et al.'s. In Safkhani et al.'s protocol, the server first chooses a one-way hash function $h(.)$. To initialize the kth tag, the server chooses two secret values s^t and s^r shared between the server and the kth tag. The kth tag is initialized with $(ID^k, h(.), s^t, s^r)$, and the server stores the entry $(ID^k, s_1^t, s_0^t, s_1^r, s_0^r, DATA^k)$ for the kth tag, where ID^k is the identifier of the kth tag, $DATA^k$ is the kth tag's related information, $s_1^t = s_0^t = s^t$, and $s_1^r = s_0^r = s^r$. After $i-1$ successful authentication rounds, the entry stored by the server is $(ID^k, s_i^t, s_{i-1}^t, s_i^r, s_{i-1}^r, DATA^k)$ and the tag contains $(ID^k, h(.), s_i^t, s_i^r)$ or $(ID^k, h(.), s_{i-1}^t, s_{i-1}^r)$.

To simplify the illustration, we assume that tag contains $(ID^k, h(.), s_i^t, s_i^r)$. As to the ith round, the details are as follows:

Step 1: The reader randomly generates a number R^r and sends a request to the tag with R^r.

Step 2: After receiving the request, the tag randomly generates a number R^t, computes $RID_i^t = h(R^t \oplus s_i^t)$ and $\alpha = h(ID^k \oplus R^t \oplus R^r \oplus RID_i^t)$, and sends α and R^t to the reader.

Step 3: Upon receiving α and R^t, the reader passes them with R^r to the back-end server.

Step 4: When receiving α, R^t, and R^r from the reader, the server computes $RID_i^{t\prime}$ and α' for the kth tag with tuples (ID^k, s_i^t, s_i^r) and $(ID^k, s_{i-1}^t, s_{i-1}^r)$, where $k = 1, 2, ..., n$ and n is the number of entries stored by the server. If the received α and the retrieved α' are equal, the server utilizes the corresponding (ID^k, s_i^t, s_i^r) or $(ID^k, s_{i-1}^t, s_{i-1}^r)$ to authenticate the tag. Assume that (ID_i^k, s_i^t, s_i^r) is the tuple used to authenticate the kth tag. The server updates the entry by choosing new secrets s_{i+1}^t and s_{i+1}^r, setting $(s_i^t, s_{i-1}^t, s_i^r, s_{i-1}^r)$ to $(s_{i+1}^t, s_i^t, s_{i+1}^r, s_i^r)$, and storing $(ID^k, s_{i+1}^t, s_i^t, s_{i+1}^r, s_i^r, DATA^k)$. The back-end server generates $DATA^k \parallel h(RID_i^r \oplus RID_i^t) \parallel RID_i^r \oplus s_{i+1}^t \parallel RID_i^t \oplus s_{i+1}^r \parallel h(s_{i+1}^t \parallel s_{i+1}^r)$ and sends it to the reader, where $RID_i^r = h(R^r \oplus s_i^r)$.

Step 5: After getting the reply $DATA^k \parallel h(RID_i^r \oplus RID_i^t) \parallel RID_i^r \oplus s_{i+1}^t \parallel RID_i^t \oplus s_{i+1}^r \parallel h(s_{i+1}^t \parallel s_{i+1}^r)$, the reader acquires the information of the tagged object and passes $h(RID_i^r \oplus RID_i^t) \parallel RID_i^r \oplus s_{i+1}^t \parallel RID_i^t \oplus s_{i+1}^r \parallel h(s_{i+1}^t \parallel s_{i+1}^r)$ to the tag.

Step 6: Upon receiving the reader's response, the tag computes $RID_i^r = h(R_i^r \oplus s_i^r)$ and $h(RID_i^r \oplus RID_i^t)$. Then the tag checks whether the computed $h(RID_i^r \oplus RID_i^t)$ and the received one are equal. If it holds, the back-end server is authenticated successfully by the tag; otherwise, this protocol terminates.

Step 7: The tag extracts s_{i+1}^t and s_{i+1}^r from $RID_i^r \oplus s_{i+1}^t$ and $RID_i^t \oplus s_{i+1}^r$, respectively. Consequently, the tag verifies the integrity through the received $h(s_{i+1}^t \oplus s_{i+1}^r)$. If s_{i+1}^t and s_{i+1}^r are valid, the tag updates s_i^t to s_{i+1}^t and s_i^r to s_{i+1}^r.

3. **Our Proposed Authentication Protocol.** Safkhani et al.'s protocol overcomes the weakness of Cho et al.'s. However, its computational load is heavy because the back-end server needs to compute $RID_i^{t\prime}$ and α' to authenticate a tag with all (ID^k, s_i^t, s_i^r)'s and $(ID^k, s_{i-1}^t, s_{i-1}^r)$'s in Sept 4. When the number of entries stored in the database is large, the back-end server needs to execute hash function plenty of times. To improve computational efficiency and ensure security, we propose an improvement.

In our protocol, the server first chooses a long-term secure value l and a one-way hash function $h(.)$. To initialize the kth tag, the server chooses two secret values s^t and s^r shared between the server and the kth tag. The kth tag is initialized with $(ID^k, h(.), s^t, s^r, l)$, and the server stores the entry $(ID^k, s_1^t, s_0^t, s_1^r, s_0^r, DATA^k)$ for the kth tag, where ID^k is the identifier of the kth tag, $DATA^k$ is the kth tag's related information, $s_1^t = s_0^t = s^t$, and $s_1^r = s_0^r = s^r$. The server also keeps l in its database. After $i-1$ successful authentication rounds, the entry stored by the server is $(ID^k, s_i^t, s_{i-1}^t, s_i^r, s_{i-1}^r, DATA^k)$ and the tag contains $(ID^k, h(.), s_i^t, s_i^r, l)$ or $(ID^k, h(.), s_{i-1}^t, s_{i-1}^r, l)$. For simplicity, we assume that tag contains $(ID^k, h(.), s_i^t, s_i^r)$. As to the ith round, the details are as follows:

Step 1: The reader randomly generates a number R^r and sends a request to the tag with R^r.

Step 2: After receiving the request, the tag randomly generates a number R^t and computes $RID_i^t = h(R^t \oplus s_i^t)$, $\alpha = h(ID^k \oplus R^t \oplus R^r \oplus RID_i^t)$, and $L = h(R^t \parallel R^r \parallel l) \oplus ID^k$. The tag then sends α, L, and R^t to the reader.

Step 3: Upon receiving α, L, and R^t, the reader passes them with R^r to the back-end server.

Step 4: When receiving α, L, R^t, and R^r from the reader, the server computes $ID^{k\prime} = h(R^t \parallel R^r \parallel l) \oplus L$. Then the server uses $ID^{k\prime}$ as the index to get the corresponding entry $(ID^k, s_i^t, s_{i-1}^t, s_i^r, s_{i-1}^r, DATA^k)$. If no such entry exists, the server terminates the protocol immediately; otherwise, the server computes $RID_i^{t\prime} = h(R^t \oplus s_i^t)$ and $\alpha' = h(ID^{k\prime} \oplus R^t \oplus R^r \oplus RID_i^{t\prime})$. If $\alpha' = \alpha$, the tag is authenticated successfully. The server updates the entry by choosing new secrets s_{i+1}^t and s_{i+1}^r, updating $(s_i^t, s_{i-1}^t, s_i^r, s_{i-1}^r)$ to $(s_{i+1}^t, s_i^t, s_{i+1}^r, s_i^r)$, and storing $(ID^k, s_{i+1}^t, s_i^t, s_{i+1}^r, s_i^r, DATA^k)$. The back-end server computes $RID_i^r = h(R^r \oplus s_i^r)$, generates $DATA^k \parallel (RID_i^r \oplus RID_i^t) \parallel RID_i^r \oplus s_{i+1}^t \parallel RID_i^t \oplus s_{i+1}^r \parallel h(s_{i+1}^t \parallel s_{i+1}^r)$, and sends it to the reader.

Step 5: After getting the reply $DATA^k \parallel h(RID_i^r \oplus RID_i^t) \parallel RID_i^r \oplus s_{i+1}^t \parallel RID_i^t \oplus s_{i+1}^r \parallel h(s_{i+1}^t \parallel s_{i+1}^r)$, the reader acquires the information of the tagged object and passes $h(RID_i^r \oplus RID_i^t) \parallel RID_i^r \oplus s_{i+1}^t \parallel RID_i^t \oplus s_{i+1}^r \parallel h(s_{i+1}^t \parallel s_{i+1}^r)$ to the tag.

Step 6: Upon receiving the reader's response, the tag computes $RID_i^r = h(R_i^r \oplus s_i^r)$ and $h(RID_i^r \oplus RID_i^t)$. Then the tag checks whether the computed $h(RID_i^r \oplus RID_i^t)$ and the received one are equal. If it holds, the back-end server is authenticated successfully by the tag; otherwise, this protocol terminates.

Step 7: The tag extracts s_{i+1}^t and s_{i+1}^r from $RID_i^r \oplus s_{i+1}^t$ and $RID_i^t \oplus s_{i+1}^r$, respectively. Consequently, the tag verifies the integrity through the received $h(s_{i+1}^t \parallel s_{i+1}^r)$. If s_{i+1}^t and s_{i+1}^r are valid, the tag updates s_i^t to s_{i+1}^t and s_i^r to s_{i+1}^r.

Note that if the tag contains $(ID^k, h(.), s_{i-1}^t, s_{i-1}^r, l)$, the sever will authenticate the tag with $(ID^k, s_{i-1}^t, s_{i-1}^r)$ instead of (ID^k, s_i^t, s_i^r). In Step 4, the server will update the entry by choosing new secrets s_{i+1}^t and s_{i+1}^r, updating $(s_i^t, s_{i-1}^t, s_i^r, s_{i-1}^r)$ to $(s_{i+1}^t, s_{i-1}^t, s_{i+1}^r, s_{i-1}^t)$, and storing $(ID^k, s_{i+1}^t, s_{i-1}^t, s_{i+1}^r, s_{i-1}^t, DATA^k)$.

4. Security and Efficiency Analyses.

In this section, we make discussions on security levels and computational loads of our protocol and Safkhani et al.'s. By the following, it is shown that our protocol provides an equivalent security level and possesses better computational efficiency.

4.1. Security analyses.

Safkhani et al.'s protocol overcomes the weaknesses, tag impersonation, reader impersonation and de-synchronization attack, which Cho et al.'s protocol suffers from. Because the difference between our protocol and Safkhani et al.'s is that ours utilizes a long-term secure value to have the server identify the tag easily, we first make discussions on tag anonymity. To ensure security, why our protocol can resist common attacks are also shown as follows.

4.1.1. *Tag anonymity.* In our protocol, the tag computes $L = h(R^t \parallel R^r \parallel l) \oplus ID^k$ in Step 2. Later, the back-end server can obtain the corresponding entry with the retrieved identity ID^k. An attacker cannot obtain the information embedded in the tag, but the transmission media is public but insecure. That is, an attacker can intercept random numbers R^t and R^r. However, he still cannot obtain the tag's identity ID^k without the knowledge of l. As a result, tag anonymity is ensured.

4.1.2. *Tag impersonation.* In Step 2 of the proposed protocol, the tag generates a number R^t, computes $RID_i^t = h(R^t \oplus s_i^t)$, $\alpha = h(ID^k \oplus R^t \oplus R^r \oplus RID_i^t)$, and $L = h(R^t \parallel R^r \parallel l) \oplus ID^k$, and sends α, L, and R^t to the reader, where R^r is the random number chosen by the reader. In Step 4, the server first computes $ID^{k\prime} = h(R^t \parallel R^r \parallel l) \oplus L$ to get the corresponding entry $(ID^k, s_i^t, s_{i-1}^t, s_i^r, s_{i-1}^r, DATA^k)$, computes $RID_i^{t\prime} = h(R^t \oplus s_i^t)$ and $\alpha' = h(ID^{k\prime} \oplus R^t \oplus R^r \oplus RID_i^{t\prime})$, and checks if $\alpha' = \alpha$ to authenticate the tag. Only the legal tag possesses s_i^t to compute RID_i^t and α with random numbers R^t and R^r to have itself authenticated. Because α is computed by $h(ID^k \oplus R^t \oplus R^r \oplus RID_i^t)$, an attacker may

attend to impersonate the tag by retransmitting α, L, and R^t_{new} to the reader, where R^r_{new} and R^t_{new} are random numbers for the present session, R^r_{old} and R^t_{old} are random numbers for the past intercepted session, and $R^r_{new} \oplus R^t_{new} = R^r_{old} \oplus R^t_{old}$. However, this attack will not threaten our protocol as well. It is because the retransmitted RID^t_i equals $h(R^t_{old} \oplus s^t_i)$ instead of $h(R^t_{new} \oplus s^t_i)$, and the server computes $ID^{k\prime}$ by $h(R^t_{new} \parallel R^r_{new} \parallel l) \oplus L$ instead of $h(R^t_{old} \parallel R^r_{old} \parallel l) \oplus L$. According to the above analyses, the proposed protocol can resist tag impersonation.

4.1.3. *Reader impersonation.* In the proposed protocol, the reader performs as a relay and possesses no secret. In the proposed protocol, readers are connected to the back-end server, and the tag only can determine whether the reader is legal or not by the data sent by the server. In Step 4, the server stores $(ID^k, s^t_{i+1}, s^t_i, s^r_{i+1}, s^r_i, DATA^k)$, computes $RID^r_i = h(R^r \oplus s^r_i)$, generates $DATA^k \parallel h(RID^r_i \oplus RID^t_i) \parallel RID^r_i \oplus s^t_{i+1} \parallel RID^t_i \oplus s^r_{i+1} \parallel h(s^t_{i+1} \parallel s^r_{i+1})$, and sends it to the reader. Then the reader passes $h(RID^r_i \oplus RID^t_i) \parallel RID^r_i \oplus s^t_{i+1} \parallel RID^t_i \oplus s^r_{i+1} \parallel h(s^t_{i+1} \parallel s^r_{i+1})$ to the tag. In Step 6, the tag computes $RID^r_i = h(R^r_i \oplus s^r_i)$ and $h(RID^r_i \oplus RID^t_i)$ after receiving the reader's response. Then the tag checks whether the computed $h(RID^r_i \oplus RID^t_i)$ and the received one are equal. If it holds, the back-end server is authenticated successfully by the tag. Meanwhile, the reader is also authenticated because the reader is incapable of computing these parameters, and only the legal reader which is connecting to the server can pass them. According to the above analyses, the proposed protocol can resist reader impersonation.

4.1.4. *De-synchronization attack.* In Step 4, the server updates the entry by choosing new secrets s^t_{i+1} and s^r_{i+1}, updating $(s^t_i, s^t_{i-1}, s^r_i, s^r_{i-1})$ to $(s^t_{i+1}, s^t_i, s^r_{i+1}, s^r_i)$, and storing $(ID^k, s^t_{i+1}, s^t_i, s^r_{i+1}, s^r_i, DATA^k)$ after authenticating the tag successfully. In Step 6, the tag computes $RID^r_i = h(R^r_i \oplus s^r_i)$ and $h(RID^r_i \oplus RID^t_i)$ and checks whether the computed $h(RID^r_i \oplus RID^t_i)$ and the received one are equal to authenticate the back-end server. In Step 7, the tag extracts s^t_{i+1} and s^r_{i+1} from $RID^r_i \oplus s^t_{i+1}$ and $RID^t_i \oplus s^r_{i+1}$, respectively. Consequently, the tag verifies the integrity through the received $h(s^t_{i+1} \parallel s^r_{i+1})$. If s^t_{i+1} and s^r_{i+1} are valid, the tag updates s^t_i to s^t_{i+1} and s^r_i to s^r_{i+1}. If the data passed by the reader is modified by an attacker, the back-end server cannot be authenticated by the tag. That is, the tag will keep $(ID^k, h(.), s^t_i, s^r_i)$ instead of $(ID^k, h(.), s^t_{i+1}, s^r_{i+1})$. Although the entry stored by the server has been updated to $(ID^k, s^t_{i+1}, s^t_i, s^r_{i+1}, s^r_i, DATA^k)$, the tag still can be authenticated by the back-end server with $(ID^k, h(.), s^t_i, s^r_i)$ in the proposed protocol. Consequently, our protocol can resist de-synchronization attack.

4.2. **Efficiency analyses.** In Step 2, the tag computes $L = h(R^t \parallel R^r \parallel l) \oplus ID^k$ and sends it to the reader. Later, the back-end server can reveal ID^k by computing $ID^k = h(R^t \parallel R^r \parallel l) \oplus L$. Therefore, the server does not need to compute $RID^{t\prime}_i$ and α' for all $(ID^k_i, s^t_{i-1}, s^r_{i-1})$'s and (ID^k_i, s^t_i, s^r_i)'s.

In our protocol and Safkhani et al.'s, only simple operations are used, and one-way hash function is the most time-consuming and complex one. Thus, only the time to execute a one-way hash function is taken into consideration in efficiency analyses. The efficiency comparisons between our protocol and Safkhani et al.'s are shown in Table 1, where T(h) denotes the time needed to compute a one-way hash function and n is the number of entries stored by the server.

According to Table 1, although one more hash function operation is executed by the tag in our protocol, the server's performance is greatly improved because the server does not need to compute $RID^t_i{}'$ and α' for all $(ID^k_i, s^t_{i-1}, s^r_{i-1})$'s and (ID^k_i, s^t_i, s^r_i)'s. When the server keeps a large amount of entries, our protocol must be superior to Safkhani et al.'s.

TABLE 1. Performance comparisons

	Server	Tag
Safkhania et al.'s	$4n$T(h)+ 3T(h)	4T(h)
Our proposed	8T(h)	5T(h)

5. **Conclusions.** Safkhania et al. proposed a hash-based authentication protocol for RFID systems to in compliance with essential security properties. After analyzing their protocol, we find that the server requires excessive computation of the hash function such that the server's computational load is heavy. In this paper, we propose an improvement to make the server identify the tag easily. According to security and efficiency analyses, it is ensured that our protocol provides an equivalent security level and possesses better computational efficiency. This makes our protocol superior to Safkhania et al.'s and practical to be implemented in the real world.

Acknowledgment. This work was supported in part by Ministry of Science and Technology under the Grants MOST 104-2622-E-034-004 -CC3, MOST 104-2221-E-034-004-, and MOST 104-2221-E-025-006-.

REFERENCES

[1] J. S. Cho, S. S. Yeo, and S. K. Kim, Securing against brute-force attack: A hash-based RFID mutual authentication protocol using a secret value, *Computer Communications*, vol. 34, no. 3, pp. 391–397, Mar. 2011.

[2] W. I. Khedr, SRFID: A hash-based security scheme for low cost RFID systems, *Egyptian Informatics Journal*, Vol. 14, no. 1, pp. 89–98, Mar. 2013.

[3] J. S. Cho, Y. S. Jeong, and S. O. Park, Consideration on the brute-force attack cost and retrieval cost: A hash-based radio-frequency identification (RFID) tag mutual authentication protocol, *Computers & Mathematics with Applications*, vol. 69, no. 1, pp. 58–65, Jan. 2015.

[4] I. Vajda and L. Buttyn, Lightweight authentication protocols for low-cost RFID tags, *Proc. of the 2nd Workshop on Security in Ubiquitous Computing*, Seattle, WA, Aug. 2003.

[5] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, EMAP: An efficient mutual-authentication protocol for low-cost RFID tags, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, LNCS 4277, pp. 352–361, Nov. 2006.

[6] T. Li and R. Deng, Vulnerability analysis of EMAP-an efficient RFID mutual authentication protocol, *Proc. of the Second International Conference on Availability, Reliability and Security*, pp. 238–245, Apr. 2007.

[7] C. M. Chen, S. M. Chen, X. Zheng, L. Yan, H. Wang, and H. M. Sun, Pitfalls in an ECC-based Lightweight Authentication Protocol for Low-Cost RFID, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 5, no. 4, pp. 642–648, Oct. 2014.

[8] C. M. Chen, S. M. Chen, X. Zheng, P. Y. Chen, and H. M. Sun, A Secure RFID Authentication Protocol Adopting Error Correction Code, *The Scientific World Journal*, vol. 2014, ID 704623, 12 pages, 2014.

[9] M. Safkhani, P. Peris-Lopez, J. C. Hernandez-Castro, N. Bagheri, and M. Naderi, Cryptanalysis of Cho et al.'s protocol, a hash-based mutual authentication protocol for RFID systems, *IACR Cryptology ePrint Archive*, Report 331, Jun 2011.

[10] M. Safkhani, P. Peris-Lopez, J. C. Hernandez-Castro, and N. Bagheri, Cryptanalysis of the Cho et al. protocol: A hash-based RFID tag mutual authentication protocol, *Journal of Computational and Applied Mathematics*, vol. 259, Part B, no. 15, pp. 571–577, Mar. 2014.