Performance Comparison of the Authentication Protocols in RFID System

S. Jantarapatin* & C. Mitrpant
National Electronics and Computer Technology Center (NECTEC)
Pathumthani, Thailand, 12120
supachok.jantarapatin@nectec.or.th

C.Tantibundhit & T. Nuamcherm
Electrical and Computer Engineering Department, Thammasat University
Bangkok, Thailand, 12120
tchatur@engr.tu.ac.th

P. Kovintavewat
Nakhon Pathom Rajabhat University
Nakhon Pathom, Thailand, 73000
piya@npru.ac.th

ABSTRACT
Radio frequency identification (RFID) is a technology for automated identification, which consists of a reading device, a server and RFID tags. We are interested in authentication protocols where two tags are simultaneously scanned by an RFID reader. Although there are several authentication protocols proposed in literature, some of them are insecure. This paper analyses the security of the authentication protocols based on two possible attacks: the brute-force attack against the proof and the forgery attack by using an authorised reader interacting with RFID tags. We also evaluate these protocols in terms of computation costs, communication costs and the resources used by adversary to run the attack (e.g., the size of memories and the number of queries). The results from our analysis indicate that our proposed protocol is more secure than the others and provides a countermeasure against forgery attack.

Categories and Subject Descriptors
C.2.2 [Network Protocols]: applications.

General Terms
Performance, Design, Experimentation, Security, Verification.

Keywords
RFID tags, proof, authentication protocol.

1. INTRODUCTION
An RFID system has been employed to identify the object of interest. The system consists of servers, readers and RFID tags. The readers and RFID tags communicate with each other via radio frequency waves. Since the RFID tags have data storage capability, and multiple RFID tags can be read without line-of-sight, they are widely used in various applications such as access control, transportation, ticketing and logistics [2]. In some applications, such as pharmaceutical distribution, some medications are required to be dispensed along with the leaflets describing their side-effects.

In such a case, two RFID tags are required for a medicine and a leaflet. In 2004, Juels proposed an authentication protocol for the case that two tags are required to be scanned simultaneously by a reading device and verified by a server. The protocol enables a pair of RFID tags to generate a proof (Yoking proof) that they have been scanned simultaneously by a reading device. However, the protocol has a problem that an unauthorised RFID reader can generate the proof even though two RFID tags are not presented at the same time [5]. Later several approaches [4, 5, 6] are proposed to resolve such a problem. In order to solve the problem in Yoking proof [4], Saito and Sakurai proposed "Grouping proof" [5]. However, if the adversary knows the time-stamp issued by a server, she can possibly generate a valid proof although two RFID tags are not concurrently presented. Later, Piramuthu proposed a "Modified proof" [6] but it is still vulnerable to the attack if the adversary knows the nonce sent by the server. Recently, we proposed a proof [1], which is more secure and can resolve this problem. In this consecutive paper, we analyse the efficiency and security of these authentication schemes in terms of communication bandwidth and the size of memories used for the scheme, computation cost and the number of queries that the adversary requires to perform the attack.

The rest of this paper is organised as follows. Section 2 briefly describes the existing protocols for scanning two tags simultaneously. We also examine the resources required to attack such protocols in this section. Section 3, the performance and security of the protocols mentioned in Section 2 are discussed and compared. Finally, Section 4 concludes this paper.

2. RELATED WORKS
This section briefly describes four protocols, namely, "Yoking Proof," "Grouping Proof," "Modified Proof," and "Proposed Proof," used to scan two tags simultaneously in a reader's field. We also show how the adversary forges the proof under some conditions. Before explaining these protocols, some notations need to be defined and used throughout this paper.

Notation:

- T_A, T_B : RFID tags (i.e., Tag A and Tag B)
- r, r_A, r_B, r_T : Nonces (random numbers)
- TS : a time stamp
- x_R, x_A, x_B : secret keys of the reader, T_A and T_B
- $MAC_x(m)$: MAC applying a secret key x on a message m

- P_{AB}: a proof that T_A and T_B are simultaneously presented

In our review, we also simulate forgery attack where the adversary can use an illegitimate reader to receive/send the messages from/to the server and the tags. The goal of the attack is to generate forged proofs of two tags being presented at the same time.

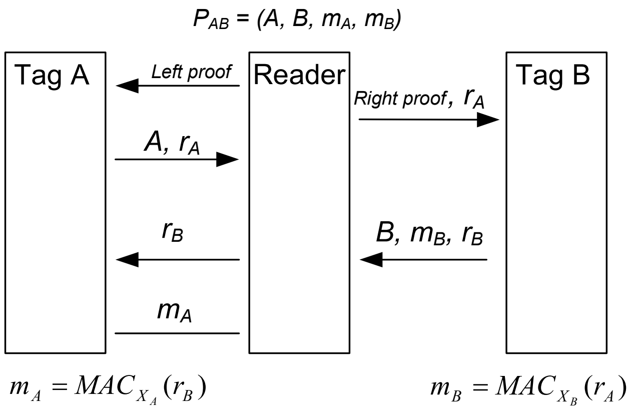

Figure 1. Yoking Proof

2.1 Yoking Proof

Yoking Proof [4] is proposed by Juels. The scheme (Figure 1) aims to provide a proof that is verifiable off-line by a trusted server, even when readers are untrusted. In order to generate a proof, the reader sends a "left proof" to T_A (tag A). T_A responds by sending a nonce (r_A) to the reader. Then the reader forwards r_A along with "right proof" to T_B (tag B). Upon receiving the message, T_B generates $m_B = MAC_{x_B}(r_A)$. After that, T_B sends m_B and a nonce (r_B) to the reader. The reader then forwards r_B to T_A. Upon receiving r_B, T_A generates $m_A = MAC_{x_A}(r_B)$ and sends it back to the reader. x_A and x_B are the secret keys shared by the server and both tags. The reader creates the proof P_{AB} by assembling A, B, m_A and m_B, and then sends the proof to the server for verification. With the stored values: r_A and r_B, the server uses x_A and x_B to verify the MACs: m_A and m_B.

However, the adversary can use an untrusted reader to forge the proof although two tags are not concurrently presented. Figure 2 shows the attack against a "Yoking proof," where the dashed line indicates the differences in time and place. In order to forge the proof, the attacker uses the illegitimate reader to acquire the values: r_A and m_A. By sending the "left proof" to T_A, the adversary receives A and r_A from T_A. Then, she generates and sends a nonce r' to T_A. In return, she receives the MAC m_A from T_A. The adversary sends the "right proof" and the nonce $r'$to T in order to obtain B, m_B and r_B at a different time. Finally she can forge the proof, $P_{AB} = (A, B, m_A, m_B)$, and uses it to convince the server that T_A and T_B are presented at the same time.

Suppose $r'$and r_A have i bits, and m_A and m_B have j bits. When the adversary sends the "*left proof*" to T_A, she needs i bits of memory to store r_A. Then, by sending r' to T_A, she needs j bits of memory for m_A.

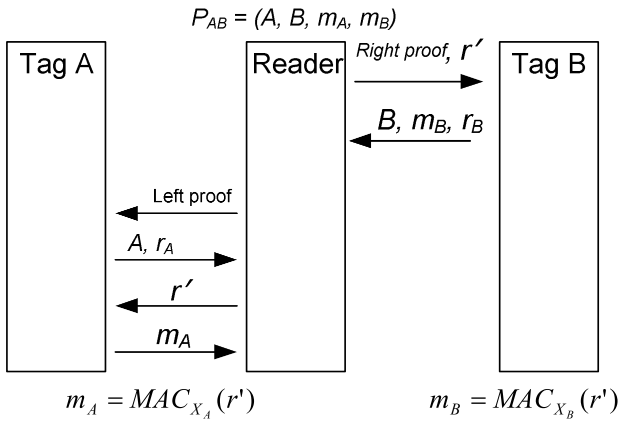

Figure 2. Forgery attack against Yoking proof

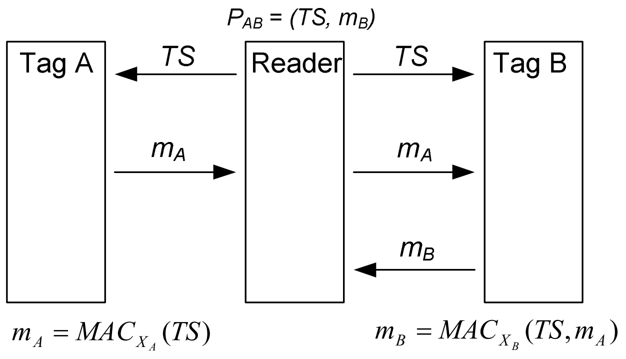

Figure 3. Grouping proof

Last, after she sends the "right proof" and $r'$to T_B, she needs $i+j$ bits of memory for m_B and r_B.The attack only requires three queries and she needs $2(i + j)$ bits of memory in total. The scheme is vulnerable because m_A and m_B are solely generated from r and r_A, in other words, m_B is independent of T_A , and m_A is independent of T_B.

2.2 Grouping Proof

In order to solve the problem of forgery, Saito and Sakurai proposed a "Grouping proof" or a "Yoking proof using time stamp" [5]. The scheme sets the condition that the protocol must be completed within a given time interval t by checking the time-stamp (*TS*) issued by the server. From the Figure 3, a reader receives the time-stamp (*TS*) from a server and sends it to T_A (tag A) and T_B (tag B) in order to create a proof. Upon the receipt of *TS*, T_A generates the MAC (m_A) from *TS* by using the secret key x_A. Then the reader sends m_A to T_B. By using m_A and *TS*, T_B generates the MAC (m_B) with the secret key x_B and sends it to the reader. The reader generates the proof: $P_{AB} = (TS, m_B)$. The server checks m_B to verify the proof.

However, the adversary can forge a proof in the scheme if she knows the value of *TS* [6]. She can obtain *TS* by either eavesdropping or guessing. If the communication channel between the server and the reader is secure, then she needs to guess *TS*.

$P_{AB} = (TS', m_B)$

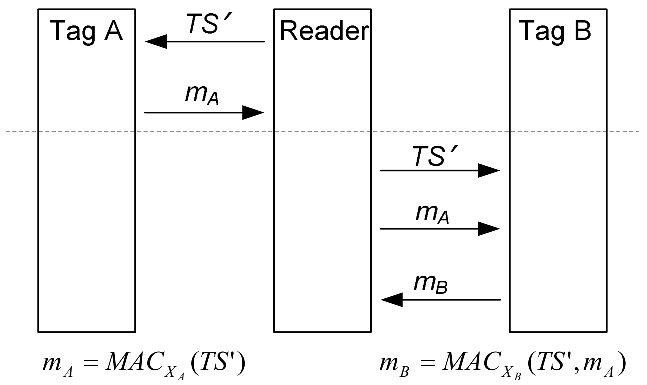

Figure 4. Forgery attack against Grouping proof

For the case that the adversary needs to guess TS in Figure 4, firstly, she generates a guess, TS', and sends it to T_A. Upon the receipt of TS', T_A will provide the adversary with the corresponding m_A. At a different time, the adversary sends TS' and m_A to T_B to obtain m_B. Finally, she can compute P_{AB}, and sends it to the server for verification. The proof is valid when TS is equal to TS'.

Assuming that TS has i bits and m_A has j bits, there are 2^i possible values of TS. Therefore, she needs $i \times 2^i$ bits to store all values of TS and $j \times 2^i$ bits to store m_B. We note that storage for m_A is temporary. Thus, a number of memory required are $(j+i) \times 2^i$ bits, and the attack totally requires 2^{i+1} queries. The scheme is still vulnerable for forgery attack since the MAC m_A is independent of T_B. Hence, she can create a valid proof for a specific time interval although both tags are not presented at the same time. The attack is shown in Figure 4.

2.3 Modified Proof

Piramuthu proposed a variation of "Yoking proof" called "Modified proof" [6]. The remarkable idea of the scheme is to create dependence of the tags on each other so that they cannot be read separately. The assumption of the scheme is that the reader is authenticated by the back-end verifier before beginning of the process of obtaining the nonce r from the server as well as when returning P_{AB} at the end of the process.

At the beginning of the protocol (Figure 5), the reader receives the nonce r from the server. Then, the reader forwards r along with the request to T_A (tag A). T_A responds with the identity A and the nonce r_A. Next, the reader sends the request and forwards r_A and the identity A to T_B (tag B). Upon the receipt of the message from T_A, T_B generates $m_B = MAC_{x_B}(r_A, r)$, and sends m_B, r_B and the identity B to the reader. After that, the reader forwards m_B to T_A. Upon the receiving of m_B, T_A computes $m_A = MAC_{x_A}(m_B, r_A)$ and sends it to the reader. Finally, the reader can compute the proof $P_{AB} = (r_A, r_B, r, m_A, m_B)$ and sends it to the server for verification.

$P_{AB} = (A, B, r_A, r_B, r, m_A, m_B)$

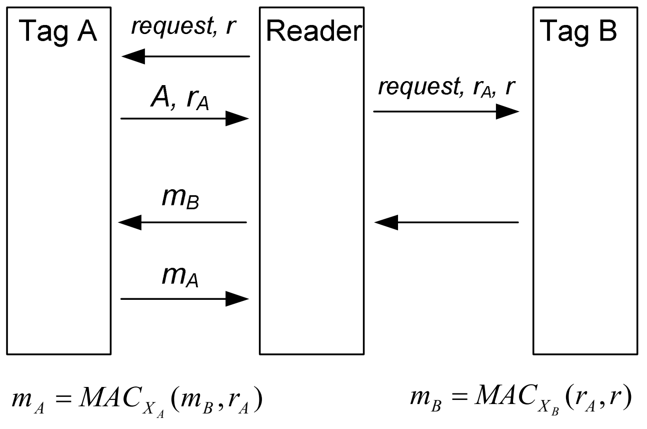

Figure 5. Modified proof

Although the reader is authenticated by the back-end verifier, the scheme did not define any mechanisms for a reader authentication. Therefore, it is possible that the adversary can use an illegitimate reader to intercept a nonce sent from the server. If she obtains a nonce r, she can forge the proof P_{AB} accepted by the server. If the unauthorised reader is not allowed to communicate with the server, then the adversary needs to guess the value of r. This attack can be performed as follows. First, the adversary generates all possible values of r and r_A along with a request to T_B. Apparently, one generated pair of r and r_A will correspond to the actual r and r_A, which then causes T_A to send m_B and r_B to the adversary. Next, the adversary sends m_B and r_B to T_B to obtain m_A and r_A. Finally, the adversary can now compute P_{AB} to be verified by the server.

Suppose r and r_A have i bits whilst m_A and m_B have j bits. There are 2^i possible values for each of r and r_A. In the attack, the attacker needs to send r for 2^i times and requires $i \times 2^i$ bits of memory to store r_A. Next, she needs to send "request", r and r_A for 2^i times and requires $(i+j) \times 2^i$ bits of memory to store r_B and m_B. Last, she needs to send m_B for 2^i times and requires $j \times 2^i$ bits of memory to store m_A. Therefore, the adversary needs $(2i+2j)$ bits of memory for storing all possible r, r_A, r_B, m_A and m_B. A number of queries required for the attack are 3×2^i queries in total. The scheme requires more memory than "Yoking proof" and "Grouping proof". However, the mechanism for reader authentication has not been defined in this protocol. Therefore, it is possible that the attacker may use an unauthorised reader to forge a proof.

2.4 Proposed Proof

Since the modified proof does not define any mechanisms for a reader authentication, the adversary possibly use an illegitimate reader to intercept a nonce that the server sends to the reader. Therefore, we proposed the secure authentication protocol in the previous paper, referred to as a "Proposed Proof" [1].

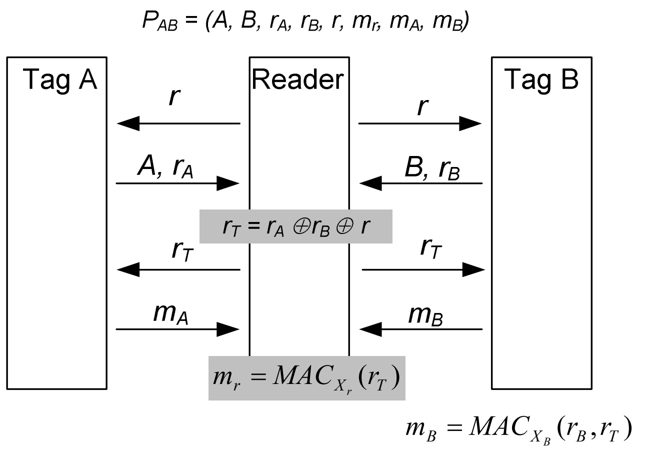
P_{AB} = (A, B, r_A, r_B, r, m_r, m_A, m_B)
Tag A
Reader
Tag B
r
r
A, r_A
B, r_B
$r_T = r_A \oplus r_B \oplus r$
r_T
r_T
m_A
m_B
$m_r = MAC_{X_r}(r_T)$
$m_B = MAC_{X_B}(r_B, r_T)$

Figure 6. Proposed proof

This scheme (Figure 6.) contains two mechanisms: a reader authentication and forgery resistance. For a server to authenticate a reader, both have shared secret keys that can be used for MAC computation and proof verification. A proof is created based on the MAC values from nonces collaboratively generated by all the tags and the server by using secret keys shared between each tag and the server.

Although the adversary uses an illegitimate reader to intercept the nonce r, she needs the value of m_R in order to forge a proof in this scheme. That means the adversary needs to know the value of the shared secret key x_r or obtains an authorised reader to generate m_R from the nonce r. There are two possible cases that the attacker can forge the proof.

First, in the case that the adversary uses the authorised reader to intercept r and to generate m_R. Then, she can obtain r_A and r_B from both tags by sending the nonce r (referred to Figure 6). After that, she generates r_T and sends it to obtain the MACs: m_A and m_B respectively. The adversary finally has sufficient information to compute P_{AB} verified by the server. For this case, we note that she can forge the proof by stealing an authorised reader.

Second, if the adversary has the authorised reader but she cannot obtain the nonce r from the server for some reasons. For this case, she can compute m_R from any trials of nonce r. Similar to the attack simulations in the previous schemes, she needs to guess r, compute m_R, and obtains the rest of the values used for generating a valid proof by sending queries to both tags. Assuming that r, r_A, r_B and r_T have i bits, and m_A, m_B and m_R have j bits. First, the attacker sends the nonce r to both tags (two queries) and requires $2{\times}i$ bits of memory for r_A and r_B. Then, she computes r_T and m_r from $2{\times}i$ possible r and requires $i{\times}2^i$ bits of memory for all r_T and $j{\times}2^i$ bits of memory for all m_r. Next, she sends r_T to T_A and T_B for $2{\times}2^i$ times in total and requires $2{\times}j{\times}2^i$ bits of memory for m_A and m_B. Therefore, the adversary totally needs $(i+3j){\times}2^i+2^i$ bits of memory and $2+2^{i+1}$ queries for the attack.

The adversary needs to obtain the secret key x_r in order to create the MAC m_R since only the authorised reader has the secret key x_r. It is impossible to create the valid proof without m_R although she can guess the nonce r or intercept it. The security of this scheme relies on the secret of the key x_r instead of a nonce or a time-stamp like the previous schemes. The attacker needs to use an exhaustive search to obtain the secret key or uses the authorised reader containing the secret key x_R to generate m_R.

3. PERFORMANCE COMPARISON

In this section, we discuss about the features and security of each protocol described in the previous section. Table 1 and Table 2 show various features, computation cost and communication cost for each of them. “Grouping proof” uses time-stamps for checking the freshness of the messages whereas the others use nonces. Therefore, “Grouping proof” has the least number of message exchanges comparing with others since it does not use challenge-response technique. “Yoking proof” and “Grouping proof” do not have any mechanisms to authenticate readers. Thus, it is very easy to forge a proof by using an illegitimate reader. Although “Modified proof” have mentioned about the back-end verifier used to authenticate the reader, it has not defined any mechanisms for the authentication. Hence, it is possible that the attacker may use any illegitimate readers to forge a proof and send it to the server. On the contrary, “Proposed proof” provides a mechanism for reader authentication. The server shares a secret key with the reader. The shared secret key is used to verify the proof whether it is sent from the authorised reader or not. This mechanism is a countermeasure against the forgery of proof. In addition, “Proposed proof” can be employed for scanning multiple tags (more than two tags) simultaneously while some of them are limited for scanning two tags only. In the case of multiple-tag authentication, the “Proposed proof” requires only $n+1$ hash operations comparing with “modified proof” which requires $2n$ hash operations.

Table 3 and Table 4 show the required resources (i.e., the size of memory and the number of queries) that the adversary needs for attacking the protocols. We examine the security of the protocols based on two possible attacks: the brute-force attack against the proof and the forgery attack by using a reader interacting with RFID tags.

Table 1. Features of different protocols

Protocols	**Time-varient parameter**	**Reader authentication mechanism**	**Multi-tags support**
Yoking proof	Nonce	Not available	No
Grouping proof	Time-stamp	Not available	No
Modified proof	Nonce	A reader is authenticated by back-end verifier	Yes
Proposed proof	Nonce	A reader is authenticated by using a shared key between reader and server	Yes

Table 2. Computation and Communication costs of different proofs

Protocols	**Number of message exchange between tags and a reader (passes)**	**Number of hash operations (H)**
Yoking proof	6	2
Grouping Proof	5	2
Modified proof	6	2 (for 2 tags) 2n (> 2tags)
Proposed proof	8	n+1 (≥ 2 tags)

If the adversary needs to forge a proof by using brute-force attack against "Yoking proof", a number of brute-force attempts are 2^{2j} times. For forging a valid proof, the adversary needs to send three queries to both tags and requires $2(i+j)$ bits of memory. For "Grouping proof", brute-force attempts are $2^{(i+j)}$ times in order to create a valid proof. If the adversary tries all possible TS in order to forge the proof, she needs to send 2^{i+1} queries to both tags and requires $(j+1)\times 2^i$ bits of memory. To launch a brute-force attack against "Modified proof", the adversary must try all $2^{(3i+2j)}$ possible proofs. For forging a proof by guessing all possible r, the adversary needs to send 3×2^i queries to both tags and requires $(3i+2j)\ \times 2^i$ bits of memory. Finally, Brute-force attempts against "Proposed proof" are $2^{(3i+3j)}$ times. We note that there are two scenarios for the forgery attack against our protocol as mention in the previous section. One of them is that the attacker obtains the authorised reader but somehow does not have any knowledge of r and x_r. Therefore, she needs to guess all possible r and use the reader to compute m_R. For such an attack, the adversary needs to send 4×2^i queries to both tags and requires $(3i+3j)\times 2^i$ bits of memory.

From Table 3 and Table 4, the adversary requires more resources than other protocols in order to launch the attacks against "Proposed proof".

Table 3. Memory consumption for the forgery attack

Protocols	Memory consumption (Bits)	Notes
Yoking proof	$2(i+j)$	The total memory includes i bits for r_A, j for m_B, i bits for r_B and j bits for m_A
Grouping proof	$(j+1)\ \times 2^i$	The total memory includes 2^i bits for TS and $j\times 2^i$ for m_B
Modified proof	$(3i+2j)\ \times 2^i$	The total memory includes $i\times 2^i$ for r_A, $j\times 2^i$ for m_A and $(i+j)\ \times 2^i$ for r_B and m_B
Proposed proof	$(3i+3j)\times 2^i$	The total memory includes as follows: $3i\times 2^i$ for r_A, r_B and r_T $3j\times 2^i$ for m_A, m_B and m_R

Although the memory consumption and the number of queries required for the attack between "Modified proof" and "Proposed proof" is not significantly different, the "Proposed proof" provides an essential security feature, the mechanism to authenticate an authorised RFID readers. Because of this mechanism, the attacker cannot use any illegitimate readers to forge the proofs. Only for the special case that the attacker obtains an authorised reader and uses it to forge the proof by guessing r.

Table 4. Number of queries and attempts required for the attacks

Protocols	**Number of queries required for the forgery attack**	**Number of brute-force attempts for forgery attack**
Yoking proof $P_{AB}=(A,B,m_A,m_B)$	3	2^{2j}
Grouping proof $P_{AB}=(A,B,TS,m_B)$	2^{i+1}	$2^{(i+j)}$
Modified proof $P_{AB}=(A,B,r_A,r_B,r,m_A,m_B)$	3×2^i	$2^{(3i+2j)}$
Proposed proof $P_{AB}=(A,B,r_A,r_B,r,m_A,m_B,m_R)$	4×2^i	$2^{(3i+3j)}$

4. CONCLUSION

In this paper, we examined and evaluated the performance and security of four protocols including our proposed protocol based on two types of the attack: the brute-force attack against the proof and the forgery attack by using a reader interacting with RFID tags. Furthermore, we compared the features of these schemes. The results are shown in Table 1, Table 2, Table 3 and Table 4. From the results, both attacks against "Proposed proof" require more resources and hassles than other protocols. Not only the "Proposed proof" is secure comparing with others, but it also can be extensively implemented for scanning multiple tags simultaneously while it requires less computation cost comparing with "modified proof".

5. ACKNOWLEDGMENTS

This work is supported by National Science and Technology Development Agency (NSTDA) and the RFID Program, National Electronics and Computer Technology Center (NECTEC), Thailand, under grant TG-44-21-50-097M. The authors would like to thank Assist. Prof. Dr. Chanathip Namprempre and Dr. Urachada Ketprom for insightful advice.

6. REFERENCES

[1] T. Nuamcherm, P. Kovintavewat, U. Ketprom, C. Tantibundhit and C. Mitrpant, "An Improved Proof for RFID Tags," in Proc. of ECTI-CON 2008, Krabi, Thailand, vol. II, pp. 737 – 740, May 14 – 16, 2008.

[2] K. Finkenzeller, RFID Handbook, 2nd ed., Wiley & Sons, 2002.

[3] A. Juels. “RFID Security and Privacy: A Research Survey,” IEEE Journal on Selected Areas in Communications, vol. 24, pp. 381-394, 2006.

[4] A. Juels. “Yoking Proofs” for RFID Tags,” in Proc. of the First International Workshop on Pervasive Computing and Communication Security. IEEE Press. 2004.

[5] J. Saito and K. Sakurai, “Grouping Proof for RFID Tags,” in Proc. of the 19th International Conference on Advanced Information Networking and Applications (AINA’05), pp. 621-624, 2005.

[6] S. Piramuthu, “On Existence Proofs for Multiple RFID Tags,” IEEE International Conference on Pervasive Services (ICPS’06), pp. 26-29, June 2006, Lyon, France.

[7] T. Dimitriou. “A Lightweight RFID Protocol to Protect Against Traceability and Cloning Attacks,” in Proc. of the IEEE International Conference on Security and Privacy for Emerging Areas in Communication Networks – SECURECOMM, 2005.

[8] A. Lenstra and E. Verheul. “Selecting Cryptographic Key Sizes,” Journal of Cryptography, vol. 14, no. 4, pp. 255 – 293, 2001.