A Privacy-preserving Lightweight Authentication Protocol for Low-Cost RFID Tags

Shucheng Yu[†], Kui Ren[‡], and Wenjing Lou[†]

[†] Department of ECE, Worcester Polytechnic Institute, MA 01609
{yscheng, wjlou}@wpi.edu
[‡] Department of ECE, Illinois Institute of Technology, IL 60616
kren@ece.iit.edu

ABSTRACT

Radio frequency identification (RFID) is an emerging technology for automatic object identification. For successful deployment of tags, a RFID system should provide effective protection over security and privacy. In particular, traceability is the main concern for user privacy. To address these issues, mutual authentication between the reader and tags is required when deploying a RFID system. Unfortunately, because low-cost RFID tags are highly resource constrained, they are not able to carry out expensive cryptographic primitives to achieve strong authentication. This paper introduces a lightweight authentication protocol for low-cost RFID tags. Compared with previous work, our solution provides better traceability protection while keeping the system efficient in terms of computation and communication. Our scheme also maintains comparable strength regarding other security aspects.

I. INTRODUCTION

Radio frequency identification (RFID), the technology for automatic object identification, is being increasingly deployed in a diverse range of applications such as inventory management, manufacturing and anti-counterfeiting. Compared with optical barcode, RFID has many obvious advantages such as unique identification and automation [1] and will hopefully take the place of the former.

However, consumer concerns on privacy may limit ubiquitous deployment of RFID tags [2], [3]. Among all privacy risks, information leakage and traceability are two most serious ones [4]. Information leakage means revealing data of an object to which a tag is attached, while traceability indicates that a tag is distinguishable and thus trackable. The former can be avoided if the back-end server allows access only to authenticated persons. The latter, however, is difficult to address because RFID tags are highly resource constrained.

The constraint on resource is caused by the acceptable price of tags in the market. Pervasive deployment of RFID requires tags to be low-cost and priced in the range of \$0.05 to \$0.10. With such a limitation on cost, a typical low-cost tag may only have few hundred bits of storage and no more than several thousand gates which can be used for security. Strong cryptographic primitives such as asymmetric encryption cannot be applied to low-cost tags.

Although the constraint on resource makes it a great challenge to protect privacy of RFID tags, several category of solutions have already been proposed in the literature. Physical approaches include tag "killing" adopted by EPC and "blocker tags" [5]. Tag "killing" protects privacy via deactivating tags. While "blocker tags" approach protects privacy by letting a tag disturb the tree-walking collision-avoidance protocol and block the tag-to-reader communication. Another category of approaches adopt authentication protocols. Most of them, e.g. [6], [7], take advantage of one-way property of the hash function to authenticate tags and/or readers. Re-encryption approaches [8]–[10] also belong to this category. These schemes are based on asymmetric cryptography but have this resource-consuming operation executed by the reader. Reference [11] achieves mutual authentication without using cryptographic primitives.

Each of these approaches solves some particular issues. However, all of them exhibit tradeoffs between efficiency and security and/or privacy. Traceability, in particular, is either addressed by sacrificing other aspects such as scalability and security or poorly defended.

To prevent traceability, the tag should respond differently upon each challenge. However, if the tag's response is totally random and unpredictable, the back-end server needs a brute-force search to find out a matching tag in its database. Actually, the number of tags in a RFID system could be on the order of millions. If every query needs a brute-force search, computation load on the back-end server would be extremely heavy.

In this paper, we propose a lightweight authentication protocol which adopts a challenge-response process to achieve mutual authentication. To make the system scalable, our scheme is designed in such a way that the response of a tag is somewhat predictable to the back-end server but appears random to outsiders. We address traceability by preventing an interrogator from querying the same tag using the same challenge number in different interrogations.

Our scheme can protect against tracking attack effectively. More important, this protocol is scalable and maintains comparable strength in other security aspects.

The rest of this paper is organized as follows. We analyze related work in Section II, and propose our protocol in Section III. We analyze our scheme in Section IV. Section V presents our conclusions.

II. Related Work

Many papers in the literature have addressed the security and privacy concerns on the use of RFID tags.

In [6], the authors devised a randomized hash key scheme. Upon each query, the tag generates a random number *r* and computes the signature *h(ID,r)*. Then the pair *(r,h(ID,r))* is sent to the reader as the response to the query. This scheme can protect against tracking attack effectively because the tag's response varies on each query. However, it requires to perform a brute-force search on the back-end server to verify the signature *h(ID,r)*. If the number of the tags is large, computation load on the back-end server would be extremely heavy.

Dimitriou proposed scheme [12] that intends to perform mutual authentication using a shared secret ID_i. In this scheme, the reader sends a random number N_R as the challenge. Upon receiving the challenge, the tag generates another random number N_T and computes the signature $h_{ID_i}(N_T,N_R)$ as the response to the challenge. To help the back-end server search the corresponding ID_i, the tag also sends a metaID $h(ID_i)$ to the reader. However, an adversary can trace the tag by metaID. To address this problem, the scheme updates ID_i after each successful interrogation. This enhancement can protect the tag from being traced for ever. But the tag is traceable between two successive successful interrogations because metaID remains unchanged.

In [11], Juels designed a challenge-response scheme which introduces no cryptographic primitives except for XOR operation. Each tag shares a list of items $(\alpha_i, \beta_i, \gamma_i)$, $1 \leq i \leq k$, with the reader. Upon query, the tag first sends a pseudonym α_i to the reader. A legitimate reader then authenticates itself to the tag by releasing the key β_i. If the tag verifies β_i, it sends key γ_i to the reader to authenticate itself. To protect against tracking attack, this scheme requires the tag releasing different pseudonym α upon each query. Pseudonyms are emitted at a low rate to prevent an adversary from harvesting all of them. However, due to the limitation on storage, a tag can store only a small list of $(\alpha_i, \beta_i, \gamma_i)$ items, say 4 or 5 for a real-world system as Juels mentioned. It is not difficult for an adversary to harvest all the pseudonyms. Frequent refreshing pseudonyms of the tag might enhance protection against tracking attack. However, it will introduce a heavy communication load to the system.

Tsudik proposed a scheme called YA-TRAP(Yet Another Trivial RFID Authentication Protocol) [13]. In YA-TRAP, tag T_i shares a unique key k_i with the reader. T_i also stores a timestamp t_i that records the last time at which it was interrogated. The reader needs to send current timestamp t_r to T_i to start the interrogation. T_i then compares t_r with its own timestamp t_i. If t_r is valid, saying $t_i \leq t_r \leq t_{MAX}$, tag T_i responds with $H_r = HMAC_{k_i}(t_r)$ and updates t_i with t_r. Otherwise, it responds with a random number. The reader can authenticate the tag by checking if there is a secret key k_j in the server that matches the equation $H_r = HMAC_{k_j}(t_r)$. This scheme is efficient in batch mode where a reader scans a lot of tags and then authenticates them in bulk. It is also not vulnerable to tracking attack. However, just as the author mentioned, this scheme is vulnerable to denial-of-service(DoS) attack. An adversary can incapacitate a tag by sending a wildly inaccurate timestamp. Besides, this scheme cannot guarantee forward security because the key k_i is never updated. Compromising the tag can disclose all its history data.

III. Our Protocol

A. Model and Requirements

System Model In our scheme, we assume the RFID system is composed of three components: tags T, readers R, and a trusted back-end server S.

Tags are all passive and each has limited resource which includes a hash function and few hundred bits of non-volatile memory. Each tag T_i is pre-configured with a secret key k_i which is *l* bits in length. To record information of previously used random numbers, our scheme requires each tag to have *m*-bit non-volatile memory to store it.

A reader is a device that queries the tag and gets its identification information. The back-end server stores all the information of tags and has all required functionality such as hash function, random number generator and so on.

Attack Model In our scheme, we assume that the back-end server is a trusted entity and well protected. Communication channel between readers and the back-end server is also assumed to be secure. Malicious readers can never get authenticated by the server.

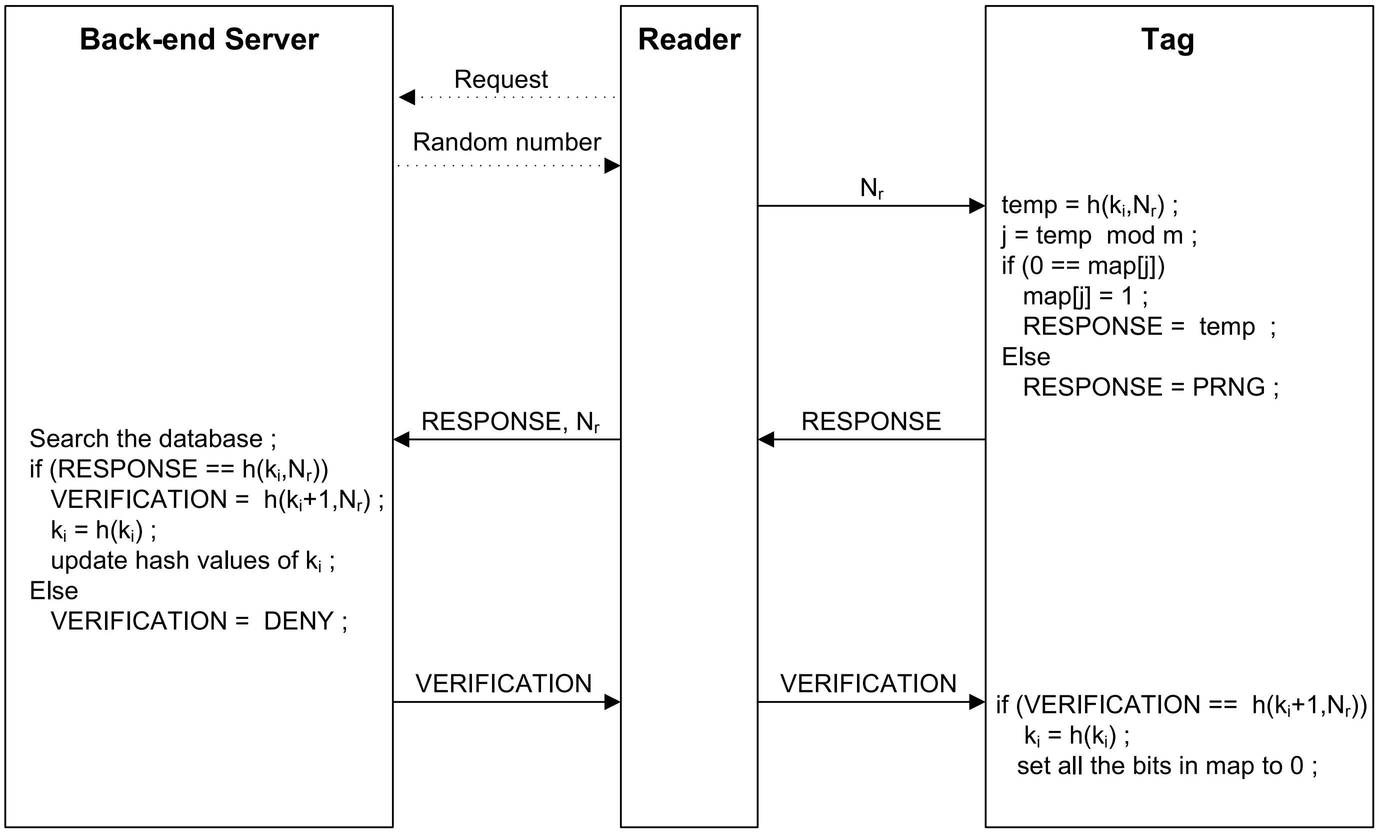

Fig. 1. protocol description

The adversary can eavesdrop the communication between tags and a reader as well as inject arbitrary messages into the communication channel. Therefore, the adversary can be either passive or active. It could be a malicious tag, a malicious reader or an eavesdropper.

In [14], Avoine classified untraceability as universal untraceability and existential untraceability. Existential untraceability means that the tag is not traceable for ever in theory. To achieve existential untraceability, computation load on the back-end server would be heavy. And more important, it is not necessary in most RFID systems. Actually, the adversary cannot execute around-the-clock attack as Juels mentioned in [1]. In [11], Juels also claimed that there is a cap on the number of times for the adversary to scan a tag or spoof an honest reader without being noticed. Our scheme follows this assumption.

Security Requirements To successfully deploy a RFID system, the following security and privacy requirements should be met.

Untraceability By analyzing a tag's response, an adversary cannot distinguish whether it is the target tag or not. Its response in history can not help identify the tag.

Confidentiality By overhearing messages between a tag and the reader, the adversary cannot learn the secret information of a tag.

Availability The RFID system is not vulnerable to Denial-of-Service(DoS) attack.

Forward Security Harvesting a tag's key can not disclose its history data.

B. Scheme Description

The protocol is illustrated as Fig. 1. N$_r$ is a random number generated by the back-end server. A tag has a *l*-bit secret key k_i and a *m*-bit map. The back-end server maintains a database which stores hash values $h(k_i, N)$ for all keys and random numbers.

The detail of our scheme is described in following steps. R represents a reader and T$_i$ represents a tag *i*.

step 1: The reader R sends a random number N_r to the tag T_i.

step 2: Upon receiving N_r, T_i first computes its position $j = h(k_i, N_r) \bmod m$ in the map, then checks bit *j* in the map (i.e. *map[j]*) to see if it has been set. If map[j] has not been set, it means that random number N_r has not been used before. T_i composes RESPONSE as $h(k_i, N_r)$ and sets map[j] to 1. Otherwise, it is very likely that random number N_r has already been used before. T_i assigns a random number to RESPONSE.

step 3: T_i sends RESPONSE to the reader.

step 4: On receiving RESPONSE from T_i, the reader queries the back-end server with (RESPONSE,N_r) . If the back-end server finds a matching item in the database, it computes VERIFICATION = $h(k_i+1, N_r)$ and updates the corresponding key k_i with $h(k_i)$ as well as the hash values $hash(k_i, N_j)$ for each random number N_j. Otherwise, the back-end server assigns DENY to VERIFICATION. Finally, the back-end server returns VERIFICATION to the reader.

step 5: The reader forwards VERIFICATION to T_i. Also, it checks VERIFICATION itself. If VERIFICATION equals DENY, the reader will query the tag with another random number.

step 6: To authenticate the reader, T_i compares VERIFICATION with $h(k_i+1, N_r)$. If only they are equal does T_i update its key with $h(k_i)$ and set all the bits in the map to 0.

The *m*-bit map is *m* bits of non-volatile memory. It is used to store information of previously used random numbers and protect against tracking attack between to successive successful interrogations. In the long run, tags are not traceable. Even if the adversary records the pair (N_r, RESPONSE) at some time point, it cannot make a link between this record with another response of the same tag after several successful interrogations. This is so because the secret key k_i is updated after each successful interrogation. However, the tag is traceable between two successful interrogations because the secret key is not changed during this period. To address this issue, we introduce the *m*-bit map to record previously received random numbers. The intuition here is to prevent a malicious reader from continuously interrogating the same tag with the same random number. If we can successfully stop a malicious reader from using the same random number in a reasonable long period, this type of tracking attack can be protected against practically.

Random number N_r is generated by the back-end server. One random number can be used to query a group of tags. Because it has already known keys of all tags, the back-end server can pre-compute $h(k_i, N_r)$ for each tag and store it in its database. During each interrogation, the back-end server simply searches its database to verify RESPONSE message from T_i. Searching complexity could be O(1) if appropriate searching algorithm, e.g. hash, is adopted. Therefore, even if the number of tags is large, realtime computation load on the back-end server is very low.

There is a tradeoff between efficiency and security. Because one random number is used to query a number of tags, an adversary may harvest the random number by eavesdropping. Then she queries a legitimate tag using this random number and stores its RESPONSE in a fake tag. Upon being queried, the fake tag can impersonate the legitimate tag by replaying the RESPONSE message. To avoid this type of cloning attack, we limit the use of our scheme to batch mode [13] where this type of cloning attack is infeasible or difficult.

If a tag is not illegally queried by the adversary, the reader can always successfully interrogate it at the first try. However, if the adversary has queried the tag, some bits of the map are set. As we will explain in the next section, the reader may need to query several times before it can receive a valid response from the tag. To make sure the reader can successfully query a legitimate tag, the back-end server should provide several random numbers for the reader. In our scheme, the back-end server generates a group of random numbers when the system is deployed. For each random number and each tag, the server pre-computes and stores the corresponding hash value $h(k_i, N_r)$. If the reader wants to query a batch of tags, it asks the server for one random number. After authentication, the server assigns one random number to the reader. Upon each successful interrogation, the back-end server should update key k_i as well as hash values $hash(k_i, N_i)$ for each random number N_i. The random number should also be updated after the back-end server verifying the batch of tags.

C. Scheme Parameters and Security Strength

Due to the limitation on memory, the tag cannot record the random numbers themselves. In this scheme, the tag records each random number using 1 bit by marking its corresponding position in the map. The position is computed by ($h(k_i, N_r)$ *mod m*). For each new random number, its position in the map is probabilistically uniformly distributed from 0 to *m-1* and not predictable to the interrogator. Apparently, collision may occur if some bits of the map are already set. If *n* bits of the map are already set, the probability of collision for the next random number is

$$Prob(Collision) = \frac{n}{m} \qquad (1)$$

An interrogator needs to retry $\frac{m}{n}$ times on average before it can get a valid response from the tag. If *n* equals *m*, the probability of collision is 1. To make itself available to a legitimate reader, the tag should clear all or part of the bits of the map. The adversary can take this chance and trace the tag using previously used random numbers. Therefore, the adversary can trace a tag by setting all the bits in the map and then querying the tag using previous random numbers. However, the number of retries to set all the bits of the map could be large due to collision. Probabilistically, it can be given by the following formula:

$$Num_{retries} = m \cdot \sum_{k=0}^{m-1} (\frac{1}{m-k}) \qquad (2)$$

Practically, a tag may have several hundred bits of memory and $Num_{retries}$ can be several thousand(see Fig.2). If the tag releases its RESPONSE at a certain (suitably slow) rate, it could take hours for an adversary to trace the tag. For low-cost RFID applications, tracking a tag in this way may be not profitable for an adversary considering the time cost. With this consideration, we believe that our scheme provides adequate protection against tracking attack. For applications where tracking between two successive successful interrogations is critical, resource abundant tags should be adopted.

While collision prolongs the time that is needed for an adversary to trace the tag, it also prevents a legitimate reader from interrogating the tag. Fortunately, the number of retries for a legitimate reader would not be large. More important, if a RFID system is not under an active attack, which is the most common case, a legitimate reader always harvests a valid response from the tag at the first try. In the worst case, *m-1* bits of the map are set by a smart adversary and the number of retries for a legitimate reader is *m* which is still much smaller than $Num_{retries}$. However, it is very hard for an adversary to intentionally set *m-1* bits of the map because the position in the map of each random number is not predictable. Generally, if the number of bits that an adversary has set is uniformly distributed from 0 to *m-1*, the average number of retries for a legitimate reader is

$$Num_{avg-tries} = \sum_{k=0}^{m-1} (\frac{1}{m-k}) \qquad (3)$$

which is not more than seven if *m* is on the order of hundreds.

Fig. 2 gives the comparison of retry number for an adversary and a legitimate reader.

On the server side, retries do not introduce extra computation load. In our scheme, the server generates *t* random numbers when the system is deployed. If *t* is carefully selected, e.g. a little bit larger than *m*, we can make sure a reader can successful harvest a valid response from a tag in most cases. The back-end server only needs to perform database searching for each retry. As we mentioned before, complexity for each search is O(1).

IV. Scheme Analysis

In this section, we will analyze security strength, system performance as well as tag functionality of our scheme. We also compare our scheme with previous work.

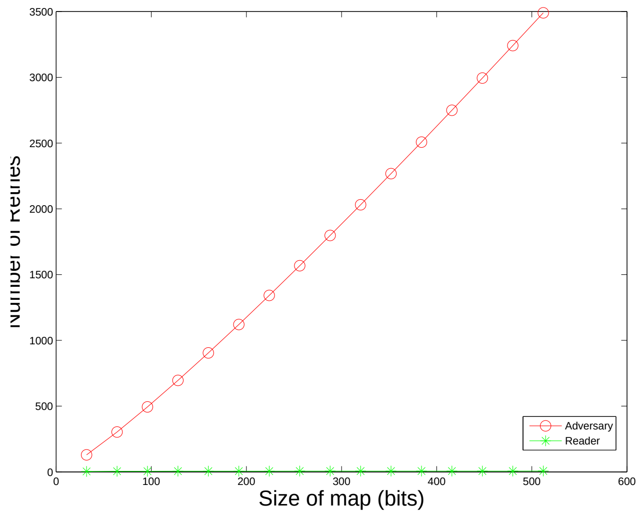

Fig. 2. Number of Retries Comaprison

A. Security Analysis

Though our scheme mainly focuses on traceability, other security issues are also important to successful deployment of RFID system. Now we examine these attacks one by one.

Tracking Attack Our scheme updates the secret key after each successful interrogation. Therefore, in the long run, an adversary cannot make a link between a tag and its RESPONSE. Between two successful interrogations, however, tracking a tag is also very hard. As we mentioned, it will take several thousand times of retry until an adversary can query the tag with previously used random numbers. If the tag releases its response at a suitable rate, e.g. slowing down when the number of 1s in the map is getting large, it will possibly take several hours for an adversary to track a tag between two successive successful interrogations. It is not profitable for an adversary in low-cost RFID applications.

Eavesdropping By eavesdropping, an adversary can harvest no secret of a tag. If a secure hash function is adopted in a tag, it is computationally infeasible for an adversary to recover the key from its hash value given the random number.

Denial-of-Service(DoS) Attack It's possible for an adversary to execute a DoS attack. By intercepting VERIFICATION message from the reader, an adversary can prevent a legitimate tag from updating its key and desynchronize it with the server. However, this issue is easy to solve. The server can backup hash value $h(k_{i-1},N)$ for the previous key (the most recently used one) when updating the key for a tag. If a tag responses with hash value on the previous key, the server can also find it in the database.

Compromising RFID tags are not resistant to physical compromising. If an adversary has compromised a tag, she can harvest all the information including the secret key. But we can still guarantee forward security for a tag. To update a key, the tag performs a one-way hash operation on the old key. The adversary cannot derive previously used keys even if she has harvested the current key.

The following table compares our scheme with previous work on security.

TABLE I
COMPARISON OF PROTOCOLS ON SECURITY

	Ours	[6]	[12]	[11]	[13]
Tracking	√	√			√
Fw Sec.	√		√	√	
DoS	√	√	√		

Since all these protocols are effective against eavesdropping, we do not include it in the table.

Based on above comparison, we can see that our scheme can protect against all these attacks. However, each of other protocols has at least one type of vulnerability.

B. *Efficiency Analysis*

Besides security, we also care about how efficient a RFID system operates. Here we will measure the efficiency of a RFID system by computation load on a tag, communication load, and computation load on the back-end server.

computation load on a tag We measure this by how many hash operations are needed on a tag for a complete interrogation. Our scheme involves two hash operations in total which are used for computing position of a random number (hash value $h(k_i,N_r)$ can be reused to compose a RESPONSE) and updating the secret key respectively. The modular operation can not be counted in the computation load because it is just taking the lower k bits of a random number if $m=2^k$.

communication load Three messages are needed for a complete interrogation. The first one is the random number, e.g. 80-bit in length, sent by the reader. The other two are RESPONSE and VERIFICATION. They are hash values of a (key, random number) pair and each has the length of 80 bits if 80-bit hash function is used. Therefore, in total we only need to transmit 240 bits for one complete interrogation if we are adopting an 80-bit hash function.

computation load on the server Our scheme can pre-compute the hash values before querying tags. During interrogation, the back-end server only needs to search the database. If appropriate searching algorithm is adopted, the server could find a matching value with complexity of O(1). In batch mode, the complexity is O(n).

The following table compares our scheme with previous work on efficiency. Computation load of the back-end sever is compared for batch mode.

TABLE II
COMPARISON OF PROTOCOLS ON EFFICIENCY

	Ours	[6]	[12]	[11]	[13]
Hash Op	2	1	3	0	1
Comm	3	3	5	4	2
Server	O(n)	O(n^2)	O(n^2)	O(n)	O(n)

According to TABLE II, we can see that the computation load of tags and communication load in our scheme is mediate. However, our computation load on the server is the lowest among all these protocols. Since the number of tags may be large, the computation load on the back-end server is critical to the practical deployment of a RFID system.

C. *Tag Functionality*

Functionality of a tag determines its cost. TABLE III compares tag's functionality of these protocols. The pseudo-random number generator(PRNG) in our scheme can be replaced with a keyed hash function as Tsudik mentioned in [13].

TABLE III
COMPARISON OF PROTOCOLS ON FUNCTIONALITY

	Ours	[6]	[12]	[11]	[13]
Hash	√	√	√		√
PRNG		√	√		
Memory	key,map	ID	ID	keys, pads	key

TABLE III shows that cost of a tag in our scheme is comparable to previous work.

V. CONCLUSION

In this paper, we study security and privacy issues in low-cost RFID systems as well as current RFID authentication protocols. We propose a lightweight authentication protocol which focuses on protecting against traceability of the tag. Compared with previous work, our scheme provides better traceability protection. More important, computation load on the back-end server is very low in our scheme. This feature makes the RFID system scalable and applicable to practical scenarios. When applied in batch mode, our scheme also maintains comparable strength regarding other security aspects.

ACKNOWLEDGMENT

This work was supported in part by the US National Science Foundation under grant CNS-0626601 and CNS-0716306.

References

[1] A. Juels, “Rfid security and privacy: A research survey,” *IEEE Journal on Selected Areas in Communication*, vol. 24, no. 2, pp. 381– 394, February 2006.

[2] “Rfid tags confirmed in australian mach3 razor packages,” 2003. [Online]. Available: http://www.boycottgillette.com/pressrelease12-22.html

[3] “Consumer group calls for immediate worldwide boycott of benetton,” 2003. [Online]. Available: http://www.boycottbenetton.com/PR_030313a.html

[4] S. E. Sarma, S. A. Weis, and D. W. Engels, “RFID Systems and Security and Privacy Implications,” in *Workshop on Cryptographic Hardware and Embedded Systems*, ser. Lecture Notes in Computer Science, vol. 2523, 2002, pp. 454–470.

[5] A. Juels, R. L. Rivest, and M. Szydlo, “The blocker tag: selective blocking of rfid tags for consumer privacy.” in *ACM Conference on Computer and Communications Security*, 2003, pp. 103–111.

[6] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, “Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems,” in *Security in Pervasive Computing*, ser. Lecture Notes in Computer Science, vol. 2802, 2004, pp. 201–212.

[7] J. Yang, J. Park, H. Lee, K. Ren, and K. Kim, “Mutual authentication protocol for low-cost RFID,” Handout of the Ecrypt Workshop on RFID and Lightweight Crypto, Ecrypt, Graz, Austria, July 2005.

[8] A. Juels and R. Pappu, “Squealing Euros: Privacy protection in RFID-enabled banknotes,” in *Financial Cryptography ’03*, ser. Lecture Notes in Computer Science, R. Wright, Ed., vol. 2742. Springer-Verlag, 2003, pp. 103–121. [Online]. Available: http://www.rsasecurity.com/rsalabs/staff/bios/ajuels/publications/euro/%Euro.pdf

[9] P. Golle, M. Jakobsson, A. Juels, and P. Syverson, “Universal re-encryption for mixnets,” in *The Cryptographers’ Track at the RSA Conference – CT-RSA*, ser. Lecture Notes in Computer Science, T. Okamoto, Ed., vol. 2964. San Francisco, California, USA: Springer-Verlag, February 2004, pp. 163–178.

[10] G. Ateniese, J. Camenisch, and B. de Medeiros, “Untraceable RFID tags via insubvertible encryption,” in *Conference on Computer and Communications Security – CCS’05*, ACM. Alexandria, Virginia, USA: ACM Press, November 2005.

[11] A. Juels, “Minimalist cryptography for low-cost RFID tags,” in *The Fourth International Conference on Security in Communication Networks – SCN 2004*, ser. Lecture Notes in Computer Science, C. Blundo and S. Cimato, Eds., vol. 3352. Springer-Verlag, 2004, pp. 149–164. [Online]. Available: http://www.rsasecurity.com/rsalabs/staff/bios/ajuels/publications/minim%alist/Minimalist.pdf

[12] T. Dimitriou, “A lightweight RFID protocol to protect against traceability and cloning attacks,” in *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm*. Athens, Greece: IEEE, September 2005.

[13] G. Tsudik, “YA-TRAP: Yet another trivial RFID authentication protocol,” in *International Conference on Pervasive Computing and Communications – PerCom 2006*, IEEE. Pisa, Italy: IEEE Computer Society Press, March 2006.

[14] G. Avoine, “Adversary model for radio frequency identification,” Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASEC), Lausanne, Switzerland, Technical Report LASEC-REPORT-2005-001, September 2005.