2008 14th IEEE International Conference on Parallel and Distributed Systems

Secure RFID Identification and Authentication with Triggered Hash Chain Variants

Tong-Lee Lim, Tieyan Li, and Tao Gu
Institute for Infocomm Research
A*STAR Singapore
{tllim, litieyan, tgu}@i2r.a-star.edu.sg

***Abstract*—In this paper, we propose two RFID identification and authentication schemes based on the previously proposed Triggered Hash Chain scheme by Henrici and Muller [1]. The schemes are designed to mitigate the shortcomings observed in the Triggered Hash Chain scheme and to ensure privacy-preserving identification, tag-reader mutual authentication, as well as forward-privacy in the case of RFID tags that have been compromised. The first scheme uses a challenge-response mechanism to defend against an obvious weakness of the Triggered Hash Chain scheme. The second scheme uses an authenticated monotonic counter to defend against a session linking attack that the first scheme is vulnerable to. We compare the level of security offered by our proposed schemes against other previous schemes and find that the schemes perform well, while keeping within reasonable overheads in terms of computational, storage and communication requirements.**

I. Introduction

In recent years, we have witnessed the breakthrough deployment of Radio Frequency Identification (RFID) technology in various industries. As RFID tags (otherwise known as transponders) are being attached to the objects to be identified, an adversary is able to track the itineraries of the objects by eavesdropping and tracing their unique identifiers. This is the major privacy issue of lightweight and low-cost RFID tags that are passive in nature and emit their identifiers in clear each time they are interrogated. To counter such privacy-related threats, one possible solution is to hide or change the identifier of a tag on every read. At the other end of the picture, RFID readers (otherwise known as interrogators) need to ensure that the tags being queried are not compromised, cloned or spoofed. This is the RFID tag authentication issue. As one of our contributions in this paper, we identify and highlight a few distinct requirements for security and privacy in RFID identification and authentication, and elaborate on why we consider them to be important under the RFID context. We then propose two new schemes and show that they satisfy most of the security and privacy requirements under a multi-level adversarial model.

Considerable research efforts have been spent on designing privacy-enhanced RFID identification and authentication protocols. Of the protocols proposed so far, a number of them require either symmetric/asymmetric ciphers or lightweight cryptographic primitives to be implemented on the RFID tag and reader. Following the works in [1] and [3]–[9], we concentrate on a class of privacy-enhanced RFID authentication protocols that make use of secure one-way hash functions. We provide a background study on these works (see section II) and explain how some of them fail to ensure certain security and/or privacy properties while focusing on some other properties. In general, we find that it is challenging to design a protocol that can maintain most security features comprehensively while keeping to the constraints of RFID tags.

In this paper, we identify the shortcoming in the previously-proposed Triggered Hash Chain scheme [1] and propose two schemes that satisfy most security properties such as privacy-preserving identification, tag-reader mutual authentication, and forward privacy, etc. The first scheme uses a challenge-response mechanism to defend against the identified weakness of the Triggered Hash Chain scheme. However, this scheme is subject to a session linking attack that can be used by an adversary to track and trace the movement of a tag. To strengthen the scheme, we further propose a forward-rolling trigger hash scheme that can prevent such an attack. The security of both schemes are analyzed in a multi-level adversarial model. We claim that the proposed schemes provide stronger security and privacy than the previous schemes. The performance of the proposed schemes is also sound with respect to computational, storage and communication overhead.

The rest of the paper is organized as follows: in section II, we provide the background and related work. In section III, we describe the security requirements for RFID identification and authentication. In section IV, we describe the original Triggered Hash Chain scheme and expose a security weakness in the scheme before proposing extensions to the scheme to strengthen it. In section V, we provide a security analysis of the two proposed schemes and in section VI, we discuss some of the performance and implementation issues. Finally, we conclude the paper in section VII.

II. Background and Related Work

In [1], Henrici and Muller proposed the Triggered Hash Chain scheme to provide privacy-preserving identification of RFID tags. The scheme uses a constantly changing external identifier to identify a tag and the reader provides an authenticated message to trigger an update of the tag's internal identifier at the end of each successful identification session. To prevent the back-end server from being desynchronized with a tag due to a lost, corrupted or maliciously modified update message, the authors proposed that the server keep a copy of the previous state of the internal identifier for every tag. As we shall explain later, we find that such a mechanism used to provide desynchronization resilience can be easily exploited by an adversary to impersonate a legitimate tag. In this paper, we propose two variants of the Triggered Hash Chain scheme to mitigate the identified weakness in the scheme. The first variant uses a challenge-response mechanism. The second variant adopts a similar mechanism used in Conti *et al.*'s RIPP-FS protocol [3]. In the RIPP-FS protocol, the back-end server holds a Lamport hash chain that is used to authenticate the timer value released for each tag-reader interaction. However, as revealed by the authors, reader authentication is not guaranteed and it is possible for an adversary to impersonate as an authorized reader. In our proposed solution, this shortcoming is eliminated to provide privacy-preserving identification and tag-reader mutual authentication.

Previously, a number of RFID identification and/or authentication schemes based on secure one-way hash functions have been proposed. In one of the earlier works, Ohkubo, Suzuki and Kinoshita proposed the use of internal identifier update with a one-way hash function to ensure forward-privacy in RFID tags [4]. The scheme uses two different hash functions, one to update the internal tag identifier and another to compute the external identifier that is to be transmittted to the reader during tag identification. Such a scheme incurs a large overhead at the server due to the need to compute hash chains and perform exhaustive search in order to identify the tag. Avoine and Oechslin then came up with an optimization of the scheme using a time-memory trade-off for the computation of the hash chains [5]. However, as mentioned in [6] and [7], the scheme is vulnerable to tag impersonation and suffers from scalability problems in face of an attack, despite Avoine and Oechslin's optimizations.

In [6], Dimitriou proposed a challenge-response protocol for tag-reader authentication. While the scheme ensures privacy-preserving identification, forward-privacy and tag-reader mutual authentication, it is possible for an adversary to cause the server and a compromised tag to be in a desynchronized state, resulting in a denial of information attack. In a later work, Dimitriou proposed a tree-based privacy-preserving RFID identification scheme [7]. Under the scheme, each tag stores a set of secret keys that lie along the path of a key tree maintained by the back-end server. During RFID identification, a set of tag responses computed using this set of keys over a random challenge will be used by the server to identify the tag. The main drawback of this scheme is that it is difficult to implement key updating since some keys are shared across different tags. Without key-updating, forward-privacy cannot be ensured once the secret keys of a tag are compromised. In [10], Lu *et al.* looked to solve this problem by proposing a method for key updating in their Strong and Lightweight RFID Private Authentication (SPA) protocol. The method relied on the use of flags to indicate which keys in the key-tree should be updated and the update notification is sent to a tag in a $synchronization$ ($sync$) message. However, we note that the $sync$ message is not authenticated and an adversary can corrupt or spoof the message to cause the tag to perform wrong updates and become desynchronized from the server. Furthermore, the communication overhead required for the scheme is rather high.

Some other schemes that are based on secure one-way hash functions include Tsudik's YATRAP [8] and Chatmon, van Le and Burmester's YATRAP+ and OTRAP [9], which were proposed to mitigate the drawbacks in YATRAP. These schemes were essentially designed mainly with privacy-preserving identification in mind and provide tag authentication but not reader authentication. Furthermore, forward-privacy is not ensured and an adversary that has compromised the secrets stored in a tag can use the information to track and trace the history and past activities of the tag based on past interactions that have been eavesdropped and recorded. In this paper, we propose two schemes based on the Triggered Hash Chain scheme to provide privacy-preserving identification, tag-reader mutual authentication, as well as forward privacy.

III. Security Requirements for RFID Identification and Authentication

In this section, we list out and describe in detail the requirements for the design of a secure identification and authentication protocol in an RFID context. In our system model, we assume that an authorized reader maintains a secure communication channel with a back-end server, which stores the secrets shared between the reader and legitimate tags. Both the reader and the back-end server are secure against tampering. On the other hand, the wireless channel between the RFID reader and RFID tags is assumed to be insecure and the RFID tags could come under the possession of an adversary and be subject to various forms of probing, physical attacks or side channel attacks. The requirements for a secure RFID identification and authentication protocol would then be as follows:

Private Identification. In RFID identification, a reader and a tag (or a batch of tags) take part in some pre-defined protocol, which allows the reader to query the tag and identify it. To preserve the privacy of the queried tag, an adversary that eavesdrops over the protocol should not be able to make out the identity (in other words the tag identifier) of the tag with higher likelihood than a pure random guess. The same should also apply to an unauthorized interrogator that attempts to query the tag. In other words, the identification protocol should ensure tag anonymity against an adversary, i.e. the tag taking part in the protocol remains anonymous to an adversary.

Tag Authentication. Tag authentication is necessary to prevent an adversary from impersonating a legitimate tag. Without tag authentication, an adversary can easily claim to be a legitimate tag and lead a reader to forward false information to the back-end server, thereby corrupting the information held at the server. In general, tag authentication requires that a legitimate tag shares a secret with the back-end server and the tag needs to prove its knowledge of that secret when authenticating itself. We say that an RFID authentication protocol ensures tag authentication against a specific adversarial model if the protocol does not reveal any information that allows the adversary to impersonate a legitimate tag and an adversary taking part in the protocol cannot successfully claim to be a legitimate tag with greater success than a random guess to the shared secret.

Reader Authentication. Reader authentication is necessary if we wish to restrict access to information stored on a tag only to authorized readers. For example, if a tag stores some private information besides its own identifier, the reader would have to be authenticated before it can be allowed to read those information. Hence, reader authentication is used to provide access control over the private information. (Here, we note that in the absence of an encrypted channel, which is the case in most RFID deployments with low-cost tags, the authorized reader would have to ensure that reading of private data occurs only in an environment that is safe from eavesdropping.) Similar to tag authentication, reader authentication also requires the sharing of secrets between authorized readers and legitimate tags. We say that an RFID authentication protocol ensures reader authentication against a specific adversarial model if the protocol does not reveal any information that allows the adversary to impersonate as an authorized reader and an adversary taking part in the protocol cannot claim to be an authorized reader with greater success than a random guess to the shared secret.

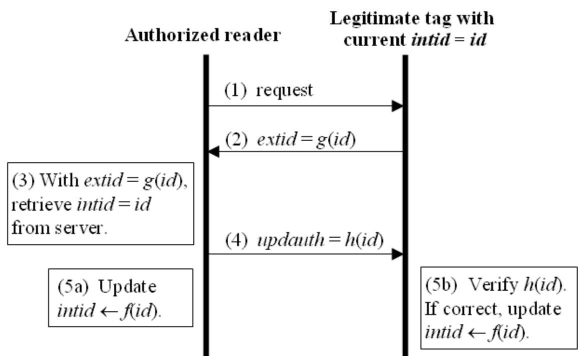
Session Unlinkability. Tag anonymity alone does not guarantee the privacy of tags. Session unlinkability must also be ensured. An adversary should not be able to link together two or more successful protocol sessions involving the same tag (regardless whether the identity of the tag is known or not) to track and trace the activities of the tag. To achieve this, any two protocol sessions involving the same tag must appear reasonably random such that the adversary cannot differentiate them (with non-negligible probability) from the protocol sessions undertaken by two separate tags.

Forward Privacy. When a shared secret is compromised (e.g. through capturing a tag and performing physical or side channel attacks), the adversary should not be able to use the compromised secret to obtain any information from previous protocol exchanges that took place between any authorized reader and the compromised tag. Hence, even when an adversary has successfully compromised a tag, the privacy of the tag in relation to past events is still preserved. This is important in the RFID context, especially in cases where the adversary can easily gain access to the RFID tags and capture them to launch physical or side channel attacks.

Desynchronization Resilience. An RFID protocol should be resilient to attacks that are targeted towards desynchronizing the tag and the back-end server. With the use of shared secrets and information, it is important that the copies of any shared secret or information stored at the tag and the back-end server must be consistent. It should not be possible for an adversary to induce changes to a tag that leads to an inconsistent (or desynchronized) state such that the tag would not be able to successfully take part in a protocol exchange with an authorized reader thereafter.

IV. The Enhanced Triggered Hash Schemes

In this section, we describe the Triggered Hash Chain scheme proposed by Henrici and Muller, and expose its shortcomings. We then present two variants of the scheme to mitigate the weaknesses in the scheme.



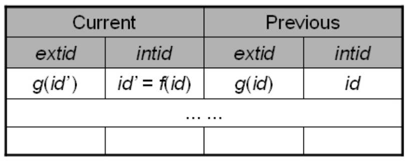
Authorized reader
Legitimate tag with current *intid* = *id*
(1) request
(2) *extid* = *g*(*id*)
(3) With *extid* = *g*(*id*), retrieve *intid* = *id* from server.
(4) *updauth* = *h*(*id*)
(5a) Update *intid* ← *f*(*id*).
(5b) Verify *h*(*id*). If correct, update *intid* ← *f*(*id*).

Fig. 1. The triggered hash chain protocol.

A. Review of the Triggered Hash Chain Scheme

In [1], Henrici and Muller proposed a triggered hash chain approach for a secure and privacy-preserving RFID identification scheme. The scheme uses three separate secure one-way hash functions (f, g and h). Each tag contains an internal identifier $intid = id$ that is kept secret (known only to itself and authorized readers) and uses its external identifier $extid = g(id)$ to identify and authenticate itself to a reader. A reader responds to a legitimate tag with an authenticated update trigger $updauth = h(id)$ that authenticates itself to the tag and triggers the identifier update on the tag. Upon verification of $updauth$, the tag would then update its tag by computing $intid \leftarrow f(id)$. Fig. 1 shows the protocol for the triggered hash scheme. A variant that replaces g with h^2 can be used to reduce the number of required one-way hash functions from three to two. Under the scheme, privacy-preserving identification, reader authentication, key secrecy, forward secrecy and desynchronization resilience are guaranteed as long as tag secrets are not exposed to the adversary. However, tag authentication is not guaranteed because of a possible attack that can allow an adversary to masquerade as a legitimate tag.

In order to ensure that the reader and the tag would not be desynchronized in the event of lost, intercepted or corrupted messages, Henrici and Muller proposed that the back-end server keeps a copy of each tag's previous identifier (see Fig. 2). During identification, if the message containing $updauth$ is not received by the tag, or is corrupted by an adversary in an attempt to desynchronize the tag, the tag would not update its identifier. Subsequently, for the next identification session, the tag would still identify itself based on the unmodified identifier while the copy of identifier held by the server has already been updated. Since the server holds the previous state for each of its copies of tag identifiers, it would be able to roll back to the previous consistent state and identify the tag based on that previous state. However, while this provides resilience to desynchronization, it gives rise to a subtle yet major problem – how do we distinguish between a valid $extid$ message that is transmitted by a desynchronized legitimate tag and one that is a copy of a previous message replayed by an adversary? The answer is that we simply cannot distinguish between the two. Under the proposed scheme, an adversary can simply



Current		Previous	
extid	*intid*	*extid*	*intid*
g(*id'*)	*id'* = *f*(*id*)	*g*(*id*)	*id*
……			

Fig. 2. The values stored at the back-end server for each tag.

replay the last valid $extid$ message sent by a legitimate tag to masquerade it and the reader would assume it to be transmitted by a legitimate tag that has been desynchronized. Hence, tag authentication is not ensured.

By recording the last valid $extid$ message transmitted by a legitimate tag and capturing the tag to prevent it from taking part in further identification sessions, an adversary can then claim the identity of the tag and authenticate successfully to authorized readers by replaying the recorded $extid$ message. Such an attack can be used to cause false and misleading information to be posted onto the server. As a countermeasure, steps can be taken to detect this attack by monitoring the tag messages and identifying the anomaly of the same $extid$ messages being transmitted over repeated times as a sign of an attack taking place. For example, the n previous valid $extid$ messages of each tag can be recorded and monitored. However, storing these messages would incur a large amount of memory. Alternatively, we can simply record the last valid $extid$ message for each tag and the number of times it has been retransmitted. An alarm would then be triggered if the number of repeated retransmissions exceeds a pre-defined threshold value. Even with such a measure, some damage would already have been incurred since the server would have captured some false information before the alarm is triggered. Clearly, the scheme needs to be improved to provide more secure identification and authentication.

B. Enhanced Scheme I - Challenge-Response Trigger

In our first scheme, we propose a challenge-response enhancement to the triggered hash scheme as a countermeasure to the replay attack described in section IV-A. In the scheme, each legitimate party to the protocol will present a random challenge. For example, the initial query request sent by the authorized reader will be accompanied by a random challenge R. The tag will then compute its external identifier as $extid = g(id, R)$ and send this, together with another random challenge R' to the reader. Upon receiving $extid$, the reader performs a search on the server to obtain id. Thereafter, the reader computes its authenticated update trigger as $updauth = h(id, R')$ and transmits this to the tag. If id cannot be found, the reader simply transmits some random value for $updauth$ and aborts the protocol. (This is necessary to prevent a covert channel, which would exist if the reader simply aborts the protocol without responding to the tag.) Upon receiving $updauth$, the tag will verify it and update its internal identifier by computing $intid \leftarrow f(id)$ if the verification is successful. Similar to the original triggered hash scheme, the server also keeps the previous state of its copy of internal identifier for

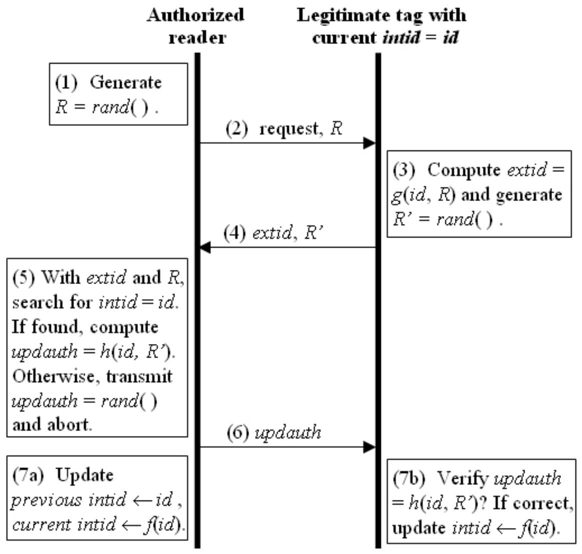

Authorized reader
Legitimate tag with current *intid* = *id*
(1) Generate *R* = *rand*( ) .
(2) request, *R*
(3) Compute *extid* = *g*(*id*, *R*) and generate *R'* = *rand*( ) .
(4) *extid*, *R'*
(5) With *extid* and *R*, search for *intid* = *id*. If found, compute *updauth* = *h*(*id*, *R'*). Otherwise, transmit *updauth* = *rand*( ) and abort.
(6) *updauth*
(7a) Update *previous intid* ← *id* , *current intid* ← *f*(*id*).
(7b) Verify *updauth* = *h*(*id*, *R'*)? If correct, update *intid* ← *f*(*id*).

Fig. 3. The protocol for the challenge-response triggered hash scheme.

each tag to provide desynchronization resilience in case of lost, intercepted or corrupted $updauth$ messages. In addition, the scheme also ensures privacy-preserving identification, reader authentication, and forward secrecy as properties inherited from the original triggered hash scheme. Fig. 3 shows the protocol for this enhanced scheme.

Due to the use of the random challenge R when computing $extid$, an adversary would not be able to replay the $extid$ message from the last valid protocol session unless if the same value of R was used. Hence, with a sufficiently large number of bits in R and a strong pseudo-random number generator with high entropy, the replay attack can be prevented. (We note that under the scheme, a successful replay can only occur with an $extid$ message from the last valid session. $extid$ messages from past valid sessions before the last valid session cannot be used even if R carries the same value due to irreversible updating of the internal identifier.)

An adversary that captures a legitimate tag can initiate the protocol with a specific value of R, record the $extid$ response from the tag, and then stop the protocol without completing it. In this case, the internal identifier of the tag remains unchanged. By iteratively repeating this process with different values of R, the adversary can collect a dictionary of valid (R, $extid$) pairs for different values of R. With a dictionary formed through brute-force querying of a captured tag, the adversary can then use the dictionary to masquerade as the tag and successfully authenticate itself to an authorized reader. Hence, it is important for $|R|$ (the bit-length of R) to be sufficiently large to make such an attack infeasible.

While private identification and tag-reader mutual authentication can be ensured, the scheme can still potentially suffer from session linking attacks. For example, an adversary can track a tag on the move in between two successive identifier updates. After a valid protocol session with an authorized reader and before the next valid session, the internal identifier of a tag remains the same. Hence, an adversary can query tags by issuing the same R value and expect to see the same $extid$ responses from the same tag since the internal identifier remains unchanged. This way, the adversary can track a tag as it is on the move by making queries and observing the $extid$ responses. Furthermore, an adversary can attempt to corrupt or block off the $updauth$ message to prevent the tag from receiving it correctly so that its internal identifier does not get updated. In this case, the adversary can go on to track the tag for as long as it is able to prevent an update to the tag's internal identifier. Thus, session unlinkability is not guaranteed.

C. Enhanced Scheme II - Forward-Rolling Trigger

In our second variant of the triggered hash scheme, we seek to completely prevent session linking attacks with the use of an authenticated reader challenge. In this case, the tag would only respond to a valid reader challenge and sends back some pseudo-random value otherwise. For this purpose, we make use of a method similar to Lamport's one-time password authentication scheme in [2]. The reader challenge is a counter value that always increments after every protocol session, i.e. a monotonically increasing counter, and it must be verified with a corresponding hash value. Obviously, such a counter will have a limited lifespan due to the limited number of bits in the counter value. Nonetheless, we contend that with careful design, it is possible to come up with a reasonable solution that matches the lifespan of an RFID tag under most practical usage scenarios.

Such a method was also used by Conti et al. in their RIPP-FS scheme [3]. In RIPP-FS, the reader stores a hash chain $h(w)$, $h^2(w) = h(h(w))$, ..., $h^{max}(w)$ (where h is a secure one-way hash function and w is a secret random seed) and uses a hash value from this chain to authenticate itself over time. Suppose the initial time at the reader is T_0 (we take this time to be discrete such that $T_n = T_{n-1} + 1$ for all $n > 0$). Every legitimate tag will store the time of the last valid protocol session carried out with the reader (T_{stored}) and this time is initialized to T_0 in the beginning. A corresponding hash value (L_{stored}), with an initial value $h^{max}(w)$, is also stored on each tag and is used to authenticate a reader. During time T_i (where $i > 0$), the reader would release the hash value $L_i = h^{max-i}(w)$ to authenticate itself. A tag first checks that $t = T_{stored} - T_i > 0$ and then, verifies that $L_{stored} = h^t(L_i)$ to authenticate the reader. The protocol exchange under the RIPP-FS scheme is shown in Fig. 4. As the authors pointed out, while RIPP-FS ensures privacy-preserving identification and tag authentication, reader authentication cannot be guaranteed due to a replay attack that is possible. Suppose there are two mutually exclusive sets of tags S_A and S_B, and all the tags in both sets last took part in a valid protocol session under RIPP-FS before time T_j, i.e. the value of T_{stored} in all those tags are less than T_j. Then, at time T_j, the authorized reader communicates with all the tags in S_A but not those tags in S_B (i.e. the tags in S_B may be away from the reader's RF field) by transmitting (T_j, L_j). An adversary that eavesadrops over the session can then replay (T_j, L_j) to any of the tags in S_B to impersonate as an authorized reader. Under our proposed scheme, such an attack would not be possible.

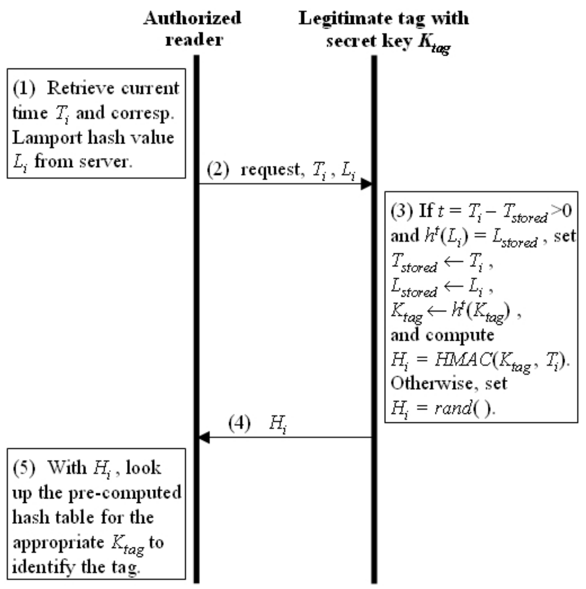

Authorized reader
Legitimate tag with secret key K_{tag}
(1) Retrieve current time T_i and corresp. Lamport hash value L_i from server.
(2) request, T_i, L_i
(3) If $t = T_i - T_{stored} > 0$ and $h^t(L_i) = L_{stored}$, set $T_{stored} \leftarrow T_i$, $L_{stored} \leftarrow L_i$, $K_{tag} \leftarrow h^t(K_{tag})$, and compute $H_i = HMAC(K_{tag}, T_i)$. Otherwise, set $H_i = rand()$.
(4) H_i
(5) With H_i, look up the pre-computed hash table for the appropriate K_{tag} to identify the tag.

Fig. 4. The protocol for Conti et al.'s RIPP-FS scheme.

In our proposed forward-rolling triggered hash scheme, an authorized reader uses the value of an incremental counter as a challenge to the tag and uses a Lamport hash value to authenticate the counter value. Both values are maintained by the back-end server and can be retrieved by the authorized reader through a secure channel. The counter value C is initialized to 0 in the beginning and is incremented with every protocol session that an authorized reader takes part in. Each legitimate tag stores the values C_{tag} and L_{tag} (the counter and Lamport hash values from the last valid protocol session), which are initialized to 0 and $h^{max}(w)$ respectively. During a protocol session where the counter value is $C = i$ (for $i > 0$), the reader will transmit $L = h^{max-i}(w)$ to authenticate the value of C. The tag verifies C by checking if $L_{tag} = h^{C-C_{tag}}(L)$. If C is verified, the tag will compute $extid = g(id, C)$ and send this, together with a random challenge R' to the reader. Otherwise, the tag simply transmits some random values for $extid$ and R', and aborts the protocol. Upon receiving $extid$, rhe reader contacts the back-end server to retrieve the tag internal identifier $intid = id$, and computes the authenticated update trigger $updauth = h(id, R')$. When the tag receives $updauth$, it verifies the value before updating its internal identifier as in the original triggered hash scheme and our challenge-response variant. The protocol exchange is shown in Fig. 5.

In order to have the tag transmit a valid $extid$, an adversary would need to have a valid (C, L) pair, where C is greater than C_{tag} and L is some pre-image of L_{tag} along the Lamport hash chain. Based on the properties of secure one-way hash functions, it is computationally infeasible for an adversary to compute such a valid L for any previously known value of L_{tag}. Moreover, the $extid$ value transmitted by a tag would either be a pseudo-random number for a non-valid (C, L) pair or a hash value computed from C for a valid (C, L)

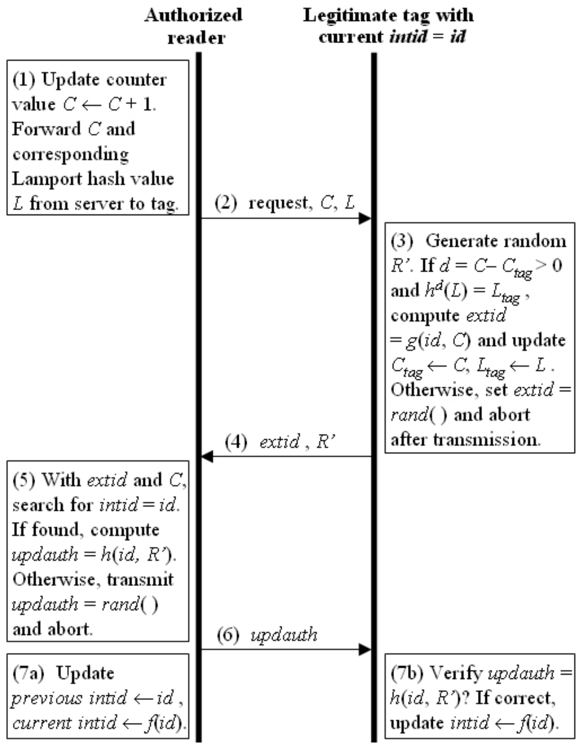

Authorized reader
Legitimate tag with current intid = id
(1) Update counter value C ← C + 1. Forward C and corresponding Lamport hash value L from server to tag.
(2) request, C, L
(3) Generate random R'. If d = C − C_tag > 0 and h^d(L) = L_tag, compute extid = g(id, C) and update C_tag ← C, L_tag ← L. Otherwise, set extid = rand( ) and abort after transmission.
(4) extid, R'
(5) With extid and C, search for intid = id. If found, compute updauth = h(id, R'). Otherwise, transmit updauth = rand( ) and abort.
(6) updauth
(7a) Update previous intid ← id, current intid ← f(id).
(7b) Verify updauth = h(id, R')? If correct, update intid ← f(id).

Fig. 5. The protocol for the forward-rolling triggered hash scheme.

pair, in which case C changes (increases) with each valid session. Hence, the adversary will not be able to link different $extid$ values transmitted by the same tag to track it. The session linking attack that the Challenge-Response Triggered Hash scheme is vulnerable to is therefore prevented under this scheme.

As in RIPP-FS, an adversary can eavesdrop a valid (C, L) pair elsewhere and replay it to a tag whose C_{tag} is less than C. However, to complete the protocol, the adversary would also have to compute a valid $updauth = h(id, R')$ in response to the $extid = g(id, C)$ message transmitted by the tag. Without knowledge of id and given a sufficiently random R', the adversary would not be able to compute a valid $updauth$ to impersonate as an authorized reader. The replay attack that works against RIPP-FS would not succeed against the forward-rolling triggered hash scheme. In this case, reader authentication can be guaranteed.

V. Security Analysis

In this section, we perform a security analysis of the proposed Challenge-Response Triggered Hash scheme and the Forward-Rolling Triggered Hash scheme. We examine the level of security offered by the schemes in terms of their ability to meet the requirements listed in section III (e.g. private identification, tag authentication, reader authentication, key secrecy, forward privacy and desyncronization resilience) against adversaries with different levels of power.

In our adversarial model, we consider adversaries with different levels of power as follows:

- **Level 1**: Ability to perform passive eavesdropping over legitimate protocol sessions and replay the eavesdropped messages.
- **Level 2**: Ability to actively take part in the protocol and query a legitimate tag/reader over a repeated number of tries (within a feasible duration) and replay the responses received.
- **Level 3**: Ability to actively corrupt/intercept/block/inject messages exchanged during legitimate protocol sessions.
- **Level 4**: Ability to capture a legitimate tag and extract its secrets through physical and side channel attacks.

We assume that a level N adversary also possesses the abilities of all levels preceding it, i.e. a level 3 adversary has the abilities of level 1 and 2 adversaries, together with its own additional abilities.

Following our analyses of the Challenge-Response Triggered Hash scheme in section IV-B and the Forward-Rolling Triggered Hash scheme in section IV-C, we find that both schemes guarantee key secrecy against level 1 to 3 adversaries since the secrets are never transmitted in clear and the transmitted $extid$ computed from the secret $intid$ involve the use of a one-way hash function such that an adversary cannot derive $intid$ from $extid$. Naturally, key secrecy cannot be guaranteed against a level 4 adversary since those tags that have been captured would have their secrets revealed to the adversary. Hence, private identification is ensured against level 1 to 3 adversaries but not against a level 4 adversary. Against a level 4 adversary, both schemes are able to ensure forward privacy since the secret internal identifiers are updated with the use of one-way hash functions and the adversary cannot derive past internal identifiers even if the current internal identifier has been compromised. The Challenge-Response Triggered Hash scheme does not ensure session unlinkability against a level 3 adversary due to the session linking attack described earlier, while the Forward-Rolling Triggered Hash scheme protects against the attack.

For both schemes, tag authentication is ensured as long as the secret $intid$ of a tag is not revealed to the adversary (i.e. a level 1, 2 or 3 adversary), and the random challenges and access passwords have adequate bit-length to prevent replay and dictionary attacks. Both schemes also ensure reader authentication against adversaries of all levels (1 to 4). In order to masquerade as an authorized reader, the adversary will need to compute a valid $updauth$ message, which is only possible if the adversary has knowledge of the tag's secret internal identifier. Since $updauth$ is computed based on the constantly updated internal tag identifier and a random nonce, it cannot be replayed from a previous valid session. We note that a level 4 adversary will be able to masquerade as an authorized reader to those tags that have already been captured and compromised. However, we consider such an attack to be irrelevant since the main objective of reader impersonation would be to access tag secrets and a level 4 adversary would have already achieved this purpose on those tags that it has captured and compromised. Rather, it is more crucial that the secrets obtained from those compromised tags do not allow the level 4 adversary to impersonate as an authorized reader to uncompromised tags. This requirement is met by both schemes

TABLE I
SECURITY COMPARISON AGAINST OTHER SCHEMES.

	Private Identification				**Tag Authentication**				**Reader Authentication**			
Adversarial Power	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Fwd-Roll Triggered Hash	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Chal-Resp Triggered Hash	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Triggered Hash Chain [1]	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes	Yes
RIPP-FS [3]	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	No	No	No
Dimitriou's Chal-Resp [6]	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
OTRAP [9]	Yes	Yes	Yes	No	Yes	Yes	Yes	No	No	No	No	No
Tree-based SPA [10]	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes

	Session Unlinkability				**Forward Privacy**				**Desync. Resilience**			
Adversarial Power	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Fwd-Roll Triggered Hash	Yes	Yes	Yes	No	N/A	N/A	N/A	Yes	Yes	Yes	Yes	No
Chal-Resp Triggered Hash	Yes	Yes	No	No	N/A	N/A	N/A	Yes	Yes	Yes	Yes	No
Triggered Hash Chain [1]	Yes	Yes	No	No	N/A	N/A	N/A	Yes	Yes	Yes	Yes	No
RIPP-FS [3]	Yes	Yes	Yes	No	N/A	N/A	N/A	Yes	Yes	Yes	Yes	No
Dimitriou's Chal-Resp [6]	Yes	Yes	No	No	N/A	N/A	N/A	Yes	Yes	Yes	No	No
OTRAP [9]	Yes	Yes	Yes	No	N/A	N/A	N/A	No	Yes	Yes*	Yes*	Yes*
Tree-based SPA [10]	Yes	Yes	No	No	N/A	N/A	N/A	Yes	Yes	Yes	No	No

* An attack to desynchronize a tag would force the authorized reader or back-end server to perform extra computations to recover the tag's internal identifier.

with the use of different secrets in separate tags.

Like the original Triggered Hash scheme, both schemes ensure desynchronization resilience against level 1 to 3 adversaries since a tag will only update its internal identifier upon receiving a valid $updauth$ message and the message cannot be replayed. A level 4 adversary would be able to compute a valid $updauth$ for a captured tag (assuming the adversary has compromised the secret $intid$) to trigger the update such that the tag becomes descynchronized with authorized readers. In addition, by keeping a copy of the previous state of $intid$, the scheme also protects against active attacks that modify or corrupt the $updauth$ message to prevent the tag from verifying it successfully or attacks that intercept and remove the $updauth$ message to prevent the tag from receiving it.

Table I gives a summary of the security comparison between the two proposed schemes and other previous schemes. In general, we find that the Forward-Rolling Triggered Hash scheme provides greater security and privacy than most other previously-proposed schemes. Like the other schemes it is only vulnerable to privacy violation, tag impersonation and desynchronization when the secret internal identifier of a captured tag has been compromised through physical means or side channel attacks. Even under such circumstances, reader authentication and forward privacy of the compromised tags can still be ensured. Besides the Challenge-Response Triggered Hash scheme, only Dimitriou's challenge-response protocol and Lu *et al.*'s tree-based SPA scheme can provide a level of protection that is close to that provided by the Forward-Rolling Triggered Hash scheme.

VI. PERFORMANCE AND IMPLEMENTATION ISSUES

In this section, we examine the performance and implementation issues for the proposed Challenge-Response and Forward-Rolling Triggered Hash schemes, and compare them against the other RFID identification and/or authentication protocols. In particular, we examine the computational, storage and communication overhead at the server and the RFID tag for each of the various schemes.

Under the Challenge-Response Triggered Hash scheme, the additional costs over other previously proposed schemes come in the form of exhaustive search at the server and storage of the previous state of $intid$ for the tags. Under the Forward-Rolling Triggered Hash scheme, the tag has to perform hash computations to verify the counter value and this comes as an additional cost over the other schemes. However, if the current counter value is close to the last counter value recorded on the tag, then the cost would be low. This is very much dependent on the frequency of tag-reader interactions. Over at the server side, exhaustive search would have to be performed to identify the tag based on the $extid$ received. This can be reduced to a simple table look-up by using a variant of the scheme that updates the $intid$ through hashing over a number of times as in the RIPP-FS scheme, instead of just once. In this case, both the server and the tag will update their copies of $intid$ with $f^d(id)$ instead of $f(id)$ (refer back to Fig. 5). This update for the tag would be performed in step (3) instead of step (7b), and step (7b) will only involve verifying $updauth$ to authenticate the reader. With this variant of the Forward-Rolling Triggered Hash scheme, the server can reduce the amount of required computations at the expense of storing a table of pre-computed hash values. On the other hand, the tag would need to perform more hash computations to update its $intid$.

A comparison of overhead between the different schemes is given in Table II. In general, we find that the Challenge-Response Triggered Hash scheme and the Forward-Rolling Triggered Hash scheme require more computational overhead and communication overhead than most of the other schemes. This is mainly due to the extra message that has to be computed and transmitted in order to ensure authentication of the reader. In fact, this security requirement is not met by most of the other schemes. For the two other schemes that ensure reader authentication, we find that the overhead under Dimitriou's Challenge-Response Authentication Protocol is comparable in terms of computations and communication overhead but lower in terms of server storage, while Lu *et al.*'s

TABLE II
OVERHEAD COMPARISON AGAINST OTHER SCHEMES.

	Computation		Storage		Communication	
	Server	Tag	Server	Tag	Reader-to-Tag	Tag-to-Reader
Fwd-Roll Triggered Hash	N + 2 Hashes or 1 T-Lookup, 2 Hashes	Δc + 3 Hashes or $2\Delta c$ + 2 Hashes	$(C_{max} + 4N)l_H$ or $(1 + 2N)C_{max}l_H$	l_H	$l_C + 2l_H$	$l_R + l_H$
Chal-Resp Triggered Hash	N + 2 Hashes	3 Hashes	$4Nl_H$	$l_R + l_H$	$l_R + l_H$	$l_R + l_H$
Triggered Hash Chain [1]	1 T-Lookup, 2 Hashes	3 Hashes	$2N(l_K + l_H)$	l_H	l_H	l_H
RIPP-FS [3]	1 T-Lookup	$2\Delta t$ + 1 Hashes	$(N + 1)T_{max}l_H$	l_H	$l_T + l_H$	l_H
Dimitriou's Chal-Resp [6]	1 T-Lookup, 2 Hashes	4 Hashes	$2Nl_H$	l_H	$l_R + l_H$	$l_R + 2l_H$
OTRAP [9]	1 T-Lookup	2 Hashes	$2Nl_H$	l_H	l_R	$2l_H$
Tree-based SPA [10]	$2log(N)$ Hashes	$log(N) + 1$ Hashes	$2Nl_H$	$log(N)l_H$	$l_R + l_H + l_{sync}$	$l_R + log(N)l_H$

• N denotes the total number of tags in the system.
• l_H denotes the length of the hash function output.
• The length of the secret internal tag identifier or key is also assumed to be l_H.
• l_R denotes the length of the pseudo-random number generator output.
• l_C and l_T denote the length of the counter value and timer value respectively.
• C_{max} and T_{max} denote the maximum counter and timer values respectively.
• Δc and Δt denote the difference between the received value and stored value for the counter and timer respectively.

SPA scheme has higher overhead in terms of tag computations and tag-reader communication. Hence, we contend that our proposed schemes satisfy the security requirements that are essential for a secure and privacy-preserving RFID identification and mutual authentication scheme at a reasonable cost.

VII. CONCLUSION

In this paper, we propose two RFID identification and authentication schemes, namely the Challenge-Response Triggered Hash scheme and the Forward-Rolling Triggered Hash scheme that are based on Henrici and Muller's Triggered Hash Chain scheme. The proposed schemes seek to mitigate the shortcomings and security weaknesses of the original scheme. We performed a security analysis of the proposed schemes by comparing its ability to meet security requirements against adversaries of different levels of power and find that the schemes perform well against other previously-proposed schemes. While the added security comes with some costs, we find that the costs are reasonable and can be comparable or even lower than previously proposed schemes.

REFERENCES

[1] D. Henrici, and P. Muller, "Providing Security and Privacy in RFID Systems Using Triggered Hash Chains", in *Proceedings of the IEEE Int'l Conference on Pervasive Computing and Communications (PerCom) 2008*, pp. 50-59, 2008.
[2] L. Lamport, "Password Authentication with Insecure Communication", in *Communications of the ACM*, vol. 24, no. 11, pp. 770-772, Nov 1981.
[3] M. Conti, R. Di Pietro, L. V. Mancini, and A. Spognardi, "RIPP-FS: An RFID Identification, Privacy Preserving Protocol with Forward Secrecy", in *Proceedings of the IEEE Int'l Conference on Pervasive Computing and Communications Workshops (PerComW) 2007*, pp. 229-234, 2007.
[4] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptogprahic Approach to 'Privacy-Friendly' Tags", in *RFID Privacy Workshop, MIT*, 2003.
[5] G. Avoine, and P. Oechslin, "A Scalable and Provably Secure Hash Based RFID Protocol", in *Proceedings of the Int'l Workshop on Pervasive Computing and Communication Security (PerSec) 2005*, pp. 110-114, 2005.
[6] T. Dimitriou, "A Lightweight RFID Protocol to Protect against Traceability and Cloning Attacks", in *Proceedings of the IEEE Int'l Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm) '05*, 2005.
[7] T. Dimitriou, "A Secure and Efficient RFID Protocol that could make Big Brother (partially) Obsolete", in *Proceedings of the 1st Int'l Conference on Pervasive Computing and Communications (PerCom) 2006*, pp. 269-275, 2006.
[8] T. Tsudik, "YA-TRAP: Yet Another Trivial RFID Authentication Protocol", in *Proceedings of the IEEE Int'l Conference on Pervasive Computing and Communications Workshops (PerComW) 2006*, pp. 640-643, 2006.
[9] C. Chatmon, T. van Le, and M. Burmester, "Secure Anonymous RFID Authentication Protocols", *Technical Report TR-060112*, Florida State University, Computer Science Dept, 2006.
[10] L. Lu, J. S. Han, L. Hu, Y. H. Liu and L. M. Ni, "Dynamic Key-Updating: Privacy-Preserving Authentication for RFID Systems", in *Proceedings of the IEEE Int'l Conference on Pervasive Computing and Communications (PerCom) 2007*, 2007.