サーバなりすまし攻撃に対し安全なRFIDセキュリティ方式

丹羽 弘和　　高木 剛　　高橋 修

公立はこだて未来大学大学院
041-8655 函館市亀田中野町 116 番地 2

あらまし RFID システムにおいて, サーバはタグから受け取った識別子と一致する値を検索してタグを識別する. したがって, サーバがタグを識別するためには, 互いが同期していなければならない. しかし, SecureComm 2005 にて, 攻撃者がタグとサーバ間の送信データを改竄して同期を崩す攻撃が提案された. 更に UBICOMM 2008 にて, 攻撃者が正規のサーバになりすまし, タグの秘匿情報を更新することで同期を崩す攻撃が提案された. 非同期状態の時, サーバはタグを識別することが出来ないため, タグの認証に失敗する. 本稿では, 従来のセキュリティ要件 (識別不可能性, 追跡不可能性, なりすまし攻撃耐性) を満たしつつ, データ同期を保つ方式を提案する. 提案方式では, サーバに秘密鍵を格納し, タグに秘密鍵を一方向性ハッシュ関数で暗号化した値を格納する手法を用いる. 仮に攻撃者がタグの秘匿情報を入手しても, 一方向性ハッシュ関数で暗号化されているため, 入手したタグの秘匿情報からサーバの秘密鍵を計算出来ない. したがって, 提案方式は攻撃者のサーバなりすまし攻撃を防ぐことが出来る.

An RFID Protocol Secure against Server Impersonation Attack

Hirokazu Niwa　　Tsuyoshi Takagi　　Osamu Takahashi

Future University - Hakodate, School of System Information Science,
116-2, kamedanakano-cho, Hakodate, Hokkaido, 041-0806, Japan.

Abstract Previous RFID protocols provided security and privacy protection by updating the identifiers of servers and tags in every authentication. An attack that results in desynchronization between a tag and a server and an attack that impersonates a server by using the data stored in a tag have recently been discussed. In this paper, we present a security protocol focused on data synchronization. We first highlight the vulnerability to attacks that break synchronization in previous RFID security protocols. To address this vulnerability, we present a new method that resynchronizes tags and servers and propose a protocol using this method. Our protocol achieves all the important security requirements (indistinguishability, backward untraceability, and resistance against tag spoofing) for an RFID system and provides data synchronization.

1　はじめに

RFID は, 無線通信上での自動識別を可能にする技術であり, タグとバックエンドで管理するサーバで構成されている. RFID は, 従来のバーコード技術と比較して, 自動認証が可能, 複数のタグを同時に読み取ることが可能, 通信可能距離が長いなどの利点がある [8, 12]. しかし, タグは計算資源が乏しいため従来の暗号技術 (SSL など) を用いることが難しい. そのため, 攻撃者による盗聴, 改竄, なりすましなどの脅威が存在する. 近年, 安全な RFID システムを構成するために, 以下のセキュリティ要件が提案された.

2003 年, 大久保等がタグの出力から特定のタグを識別出来ない性質 (識別不可能性) と, 現在のタグの秘匿データが漏洩しても所有者の過去の行動を追跡出来ない性質 (追跡不可能性) を RFID Privacy Workshop 2003 にて提案した [6]. 更に 2005 年, Rhee 等が SPC 2005 にて, なりすまし攻撃に対して安全である性質 (なりすまし攻撃耐性) を提案した [7]. これらのセキュリティ要件を満たす方法の一つとして, 認証毎にタグとサーバが格納している識別子 (ID) や秘密鍵などを更新する方法が有効であると知られている [1, 2, 4, 5, 6, 7, 10, 11]. しかし, サーバがタ

グを識別するためには, 互いの秘匿情報 (*ID* 等) が同期していなければならない. 本稿では, 特にデータ同期に注目した方式を考察する.

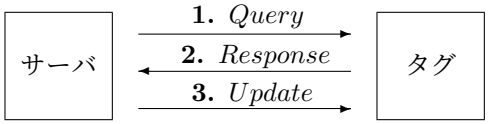
SecureComm 2005 にて, Dimitrios はタグとサーバ間の送信データを改竄して同期を崩す攻撃を提案した [2]. この攻撃に対応するために, 2008 年, Lim 等が ISPEC 2008 にて, タグとサーバに同期化された鍵を格納する方式を提案した [5]. 更に同じく 2008 年, Canard 等が RFIDsec 2008 にて, ハッシュチェイン方式を用いて同期を保つ方式を提案した [1]. しかし, UBICOMM 2008 にて Song がサーバなりすまし攻撃を提案した [9]. これは, 攻撃者がタグの格納している秘匿情報を入手し, 正規のサーバになりすまして特定の正規タグが格納する秘匿情報を更新する攻撃である. しかし, 攻撃者によって更新された正規タグに対し, 正規サーバの秘匿情報は更新されないため, 正規サーバと正規タグの同期が崩れてしまう. Lim や Canard らの方式は, サーバなりすまし攻撃に対して脆弱だった.

本稿では, 識別不可能性, 追跡不可能性, なりすまし攻撃耐性を満たし, かつ, サーバとタグのデータ同期を保つプロトコルを提案する. 提案方式では, サーバに秘密鍵を格納し, タグに秘密鍵を一方向性ハッシュ関数で暗号化した値を格納する. 仮に攻撃者がタグの秘匿情報を入手しても, タグの秘匿情報からサーバの秘密鍵を計算出来ないので, 攻撃者は正規サーバになりすますことが出来ない. また, タグとサーバが非同期状態ならば秘密鍵を用いてタグを識別し, 鍵と対応している識別子 (*ID*) を再同期化することで, サーバとタグの同期を保つことが出来る. 従来のセキュリティ要件に関しては, 認証毎に秘匿情報を一方向性ハッシュ関数を用いて更新することで識別不可能性と追跡不可能性を満たし, 乱数を暗号変換の計算に用いることでなりすまし攻撃耐性を満たす.

本論文の構成として, 二章では, 本稿で取り扱う RFID システムのモデル概要とセキュリティ要件について述べる. 三章では, 従来方式の評価を行い, サーバなりすまし攻撃に対して脆弱であることを示す. 四章では, 提案プロトコルの説明を行い, セキュリティ評価を行う. 五章では, 本論文のまとめを述べる.

2 RFID セキュリティシステム

本章では, 本稿で扱う RFID の認証プロセスについて述べる.



4. サーバの *ID* を更新　　5. タグの *ID* を更新

図 1: RFID システムの認証プロセス

2.1 RFID の構成

RFID システムは, タグとバックエンドで管理しているサーバから成る. タグは, 固有の *ID* をメモリに格納している. サーバは, 管理している全タグの *ID* とタグを取りつけている商品の情報 (値段, 商品名など) を対応付けて管理している. RFID システムにおける認証の流れは, 次の通りである; (1) サーバからの応答要求信号 (*Query*) に対し, (2) *ID* を暗号変換した値 (*Response*) をサーバへ返信する. サーバは *Response* を複合化し, 一致する *ID* を検索してタグを特定する. 特定に成功した場合, (3) タグの秘匿情報 (*ID* や共有鍵など) を更新する *Update* を計算してタグへ送信し, (4) 自身の秘匿情報を更新する. (5) タグは, 受信した *Update* が有効な値であるかを確認して, 秘匿情報を更新する. 図 1 にて認証プロセスの流れを示す. 本論文では, この認証が繰り返し行われると仮定する.

2.2 セキュリティ要件

本節では, 安全性を保つために必要なセキュリティ要件について述べる.

識別不可能性 (IND)[6] : *Response* から特定のタグを識別可能である場合, 攻撃者はタグの追跡が可能となる. したがって, タグの出力値は匿名でなければならない.

追跡不可能性 (BU)[6] : 耐タンパ性を備えていないタグは, 電気解析攻撃などの攻撃によって格納している秘匿データが攻撃者に漏洩する危険がある. もし, 漏洩した秘匿情報を用いて過去の秘匿情報を計算出来た場合, 攻撃者はタグ所有者の行動履歴を追跡可能となる. 追跡不可能性とは, 現在のタグの秘匿データが漏洩しても所有者の過去の行動を追跡出来ない性質である.

なりすまし攻撃耐性 (Spf)[7] : 攻撃者は正規タグと情報交換することで, タグの秘匿情報を知ることなくタグになりすますことが可能となる. なりすまし攻撃は, 以下の攻撃が存在する. RFID システムは, これらのなりすまし攻撃に対して安全でなければならない.

- Passive Attack: 攻撃者は, タグとサーバ間の通信を盗聴し, 入手した *Response* を再利用して正規タグになりすます.

- Active Attack: 攻撃者は, 正規タグに *Query* を送信し, *Response* を受け取る. そして, 受け取った *Response* を正規サーバに送信することで正規タグになりすます.

2.3 同期問題 (SP)

ID を用いた相互認証プロトコルにおいて, タグは *Response* をサーバの *Query* 毎に変えるために *ID* を認証毎に更新する. 通常, サーバは, タグの *ID* と同じ *ID* を検索してタグを識別する. したがって, サーバとタグの *ID* は認証時において常に同期状態でなければならない. 同期状態とは, タグの *ID* とサーバの *ID* が同じ値である状態を指す. この同期が崩れた場合, サーバは *ID* の検索に失敗してしまうため, タグの識別が不可能になる. 近年, サーバとタグの同期を崩す攻撃が提案されており, 本稿では以下の攻撃を考慮する.

Blocking Attack (BA)[2] : この攻撃は, サイドチャンネルアタックの一種であり, SecureComm 2005 にて, Dimitrios 等によって提案された. 攻撃者は, サーバからタグへ送信された $Update$ を改竄可能であるとする. 正規サーバから送信された $Update$ を攻撃者が改竄した場合, タグは受信した $Update$ が不正データであると判断し, 秘匿情報の更新を行わない. しかし, サーバ側では $Update$ の送信後に秘匿情報を更新しているため, 認証に用いる秘匿情報が非同期の状態となる.

Server Impersonation Attack (SIA)[9] : この攻撃は, UBICOMM 2008 にて, Song によって提案された. 攻撃者は, ある特定のタグ T_i の秘匿情報を入手可能であると仮定する. 攻撃者は入手した秘匿情報を格納した偽のサーバを作成し, T_i に $Query$ を送信する. $Query$ を受け取った T_i は, 格納している *ID* を用いて *Response* を計算して偽サーバに返信する. *Response* を受け取った偽サーバは, 攻撃者が入手した秘匿情報を用いて $Update$ を計算して T_i へ送信する. この時, 偽サーバが T_i から入手した秘匿情報を用いて有効な $Update$ を計算可能ならば, 偽サーバは正規のサーバになりすますことが出来る. 偽サーバが送信した $Update$ を受信した T_i は, 受信した $Update$ が有効な更新命令であると判断し, 秘匿情報の更新を行う. しかし, 正規サーバと T_i は認証を行っていないため, 正規サーバの秘匿情報は更新されない. したがって, タグとサーバの同期が崩れ, 以降の認証は全て失敗する. SIA の流れを図 2 にて示す.

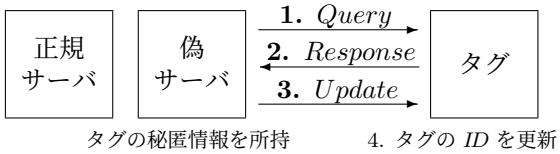


図 2: Server Impersonation Attack[9]

3 従来のセキュリティ方式

本章では, データ同期を考慮した従来のプロトコルに対する評価を行う.

LOK 方式 [5]

タグとサーバに同期化された共有鍵を格納し, *ID* の同期が崩れた場合は鍵を用いてタグを識別する方式である. しかし, この方式はタグとサーバで *ID* と鍵を共有しており, サーバがタグへ送信する $Update$ も共有鍵を用いて計算される. したがって, 攻撃者はタグの共有鍵を入手することで有効な $Update$ を計算出来, タグの共有鍵を更新することが可能である. SIA 後, *ID* と共有鍵の同期が共に崩れているためサーバはタグを識別出来ず, 再同期化も行われない.

CC 方式 [1]

ハッシュチェイン方式を用いて *ID* を更新することで, 同期が崩れてもサーバ側でタグの *ID* を計算出来る方式である. *ID* が非同期の場合, サーバは格納している *ID* のハッシュ値を計算することでタグを識別出来る. しかし, この方式は $Update$ を互いに共有している *ID* を用いて計算する. したがって, 攻撃者はタグの *ID* を入手することで有効な $Update$ を計算することが出来る. 攻撃者は, SIA を二回以上行うことでタグとサーバの同期を崩すことが出来る. この時, サーバはタグを識別出来ず, *ID* の再同期化も行われないので以降の認証は全て失敗する.

4 提案方式

本章では, ハッシュ関数, 乱数, 排他的論理和を用いて 2.2 章にて述べたセキュリティ要件を満たし, データ同期を保つプロトコルを提案する. 本章における表記法の定義を以下に示す.

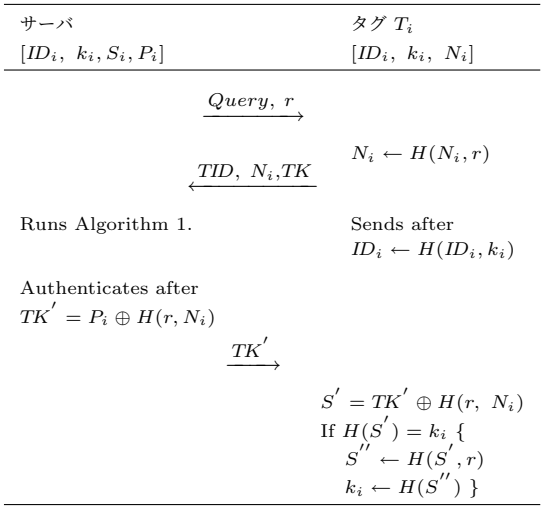


図 3: 提案方式の認証プロセス

Notation

k_i: タグ T_i と共有している鍵. $k_i \in (0,1)^m$.
r: 乱数. $r \in (0,1)^m$.
S_i: k_i を更新する際に用いる秘匿情報. $S_i \in (0,1)^m$.
P_i: 前回使用した秘匿情報. $P_i \in (0,1)^m$.
N_i: タグ T_i が格納している値. $N_i \in (0,1)^m$.
$H()$: 一方向性ハッシュ関数. $H\{(0,1)^*\} \rightarrow (0,1)^m$.
$\oplus$ 排他的論理和.
$\|$: 連結関数.
$\leftarrow$: 代入関数.

Algorithm 1 Authentication algorithm

INPUT: $TID,\ TK,\ N_i$
OUTPUT: $SUCCESS$ or $ERROR$

$ID^{'} \leftarrow TID \oplus H(r,\ N_i)$.
if $ID^{'} == ID_i$ **then**
  $ID_i \leftarrow H(ID_i, r)$.
  if $H(k_i, r \parallel N_i) == TK$ **then**
    $P_i \leftarrow S_i$.
    $S_i \leftarrow H(S_i,\ r)$.
    $k_i \leftarrow H(S_i)$.
  else
    $S_i \leftarrow H(P_i,\ r)$.
    $k_i \leftarrow H(S_i)$.
  end if
else if $H(k_i, r \parallel N_i) == TK$ **then**
  $ID_i \leftarrow H(ID^{'}, r)$.
  $P_i \leftarrow S_i$.
  $S_i \leftarrow H(S_i,\ r)$.
  $k_i \leftarrow H(S_i)$.
else if $H(H(P_i), r \parallel N_i) == TK$ **then**
  $ID_i \leftarrow H(ID^{'}, r)$.
  $S_i \leftarrow H(P_i,\ r)$.
  $k_i \leftarrow H(S_i)$.
else
  return $ERROR$.
end if
return $SUCCESS$.

4.1 提案方式

提案方式の認証プロセスを図 3 に示す. タグからの $Response$ を受信した時, サーバは Algorithm 1 を用いて認証を行う.

1. **サーバ**: 乱数 r を生成し, $Query$ と r をタグへ送信する.

2. **タグ**: 受け取った r を用いて $H(N_i, r)$ を計算して N_i を更新する. 次に, $TID = ID_i \oplus H(r, N_i)$, $TK = H(k_i, r \parallel N_i)$ を計算し, TID, TK, N_i を $Response$ としてサーバに送信する. 送信後, $H(ID_i, r)$ を計算し, 格納している ID_i を更新する.

3. **サーバ**: 受信した TID を用いて $ID^{'} = TID \oplus H(r,\ N_i)$ を計算し, $ID^{'}$ と一致する ID_i を検索する.

   検索に成功した時: 特定した ID_i を $H(ID_i,\ r)$ に更新する. 次に, ID_i と対応している k_i を用いて $H(k_i,\ r \parallel N_i)$ を計算して TK と比較する. 一致した時, P_i を現在の S_i に, S_i を $H(S_i, r)$ に更新する. 最後に, 更新した S_i を用いて k_i を $H(S_i)$ に更新する. 一致しなかった時, P_i の更新は行わず, S_i を $H(P,\ r)$ に更新し, 更新した S_i を用いて k_i を $H(S_i)$ に更新する.

   検索に失敗した時: TID を用いてタグを特定出来ないため, サーバに格納している全ての k_i 毎に $H(k_i,\ r \parallel N_i)$ を計算して TK と比較する. 一致する k_i を特定出来た時, k_i と対応している P_i を S_i に, S_i を $H(S_i,\ r)$ に更新し, k_i を更新した S_i を用いて $H(S_i)$ に更新する. 最後に, k_i と対応した ID を $H(ID^{'},\ r)$ に更新する. k_i を特定出来なかった時, 格納している P_i 毎に $H(H(P_i),\ r \parallel N_i)$ を計算して TK と比較する. 一致した時, S_i を $H(S_i, r)$ に更新し, S_i を用いて k_i を $H(S_i)$ に更新する. 最後に, P_i と対応している ID_i を更新する. P_i を用いても TK と一致する値を計算できない場合, 受信したデータは改竄された可能性が高

いと判断し, 破棄する. この時, これ以降の処理が行われない.

4. **サーバ**: 認証後, $TK^{'} = P_i \oplus H(r,\ N_i)$ を計算してタグへ送信する. この $TK^{'}$ が $Update$ に相当する.

5. **タグ**: $S^{'} = TK^{'} \oplus H(r,\ N_i)$ を計算する. もし, $H(S^{'}) = k_i$ ならば, $S^{''} = H(S^{'},\ r)$ を計算し, k_i を $H(S^{''})$ に更新する. 一致しなかった場合, k_i の更新は行われない.

4.2 セキュリティ評価

本節では, 提案方式の安全性について検証を行う.

識別不可能性 (IND) : 認証開始時, サーバは r を生成してタグへ送信する. この r は認証毎に生成される異なる乱数である. したがって, この r を用いて計算される TID, TK, N_i, $TK^{'}$ も認証毎に異なる値となる. したがって, 識別不可能性を満たす.

追跡不可能性 (BU) : タグが格納する秘匿情報は k_i, ID_i, N_i である. これらは, 一方向性ハッシュ関数を用いて認証毎に更新するため, 仮に秘匿情報が漏洩しても, 過去の秘匿情報を計算出来ないため, 過去の出力値を求められない. したがって, フォワードセキュア性を満たす.

なりすまし攻撃耐性 (Spf) : なりすまし攻撃である Passive Attack と Active Attack に対して, 提案方式は安全であることを示す.

- Passive Attack: サーバが生成する r は認証毎に異なる乱数である. よって, タグからサーバへ送信される TID, TK, N_i も認証毎に異なる値である. つまり, 攻撃者は TID, TK, N_i を入手しても次の認証セクションで再利用出来ない. したがって, 提案方式は Passive Attack に対して安全である.

- Active Attack: 攻撃者がタグへ送信する乱数を $r^{'}$ とし, 正規サーバが持つ乱数を r $(r \neq r^{'})$ とする. 攻撃者は $r^{'}$ と $Query$ を送信し, サーバは TID, TK, N_i を計算して攻撃者へ返信する. この時, $N_i = H(r^{'},\ N_i)$, $TID = ID_i \oplus H(r^{'},\ N_i)$, $TK = H(k_i,\ r^{'} \parallel N_i)$ である. 攻撃者は, 受け取った TID, TK, N_i を正規サーバへ送信する. 受け取ったサーバは TID から $ID^{'}$ を計算して一致する ID_i を検索する. しかし, サーバが持つ乱数 r と $r^{'}$ の値は異なるため, サーバは TID から ID_i と一致する $ID^{'}$ を計算出来ない. 同様に, TK と一致する P_i, k_i の検索も失敗する. この時, サーバは受信データを破棄して認証を行わない. したがって, 提案方式は Active Attack に対して安全である.

Status No	ID-Synch	Key-Synch
0	√	√
α	−	√
β	√	$H(pk_i)$
γ	−	$H(pk_i)$

√: 同期状態, −: 非同期状態

表 1: タグとサーバの同期状態

Action	Step 2	Step 4	Complement
A	√	√	正常
B	−	none	TID, TK, N_i を改竄
C	√	−	$TK^{'}$ を改竄

√: 正常に受信成功, −: 改竄の可能性あり

表 2: 攻撃者による妨害方法

4.3 同期問題

タグとサーバの同期を崩す攻撃として, 2.3 章で記述した BA と SIA に対して安全であることを示す.

Blocking Attack (BA) : タグとサーバの同期状態は, 表 1 に示した通り 4 つの状態に分類することが出来る (状態 0 は正常の状態を表す).

状態 α は, 攻撃者によって TID, TK, N_i を改竄された, またはサーバが正常に受信出来なかった場合である (Action B に相当). そのため, サーバ側の ID が更新されず ID が非同期の状態である. この時, k_i が同期しているため, サーバは TK と同じ値を計算出来る. したがって, サーバはタグの識別が可能である. また, $ID^{'}$ はタグが格納する ID_i と同じ値なので, 認証後, サーバが格納する ID_i をタグが更新した ID_i と同じ値に更新して再同期化を行う.

状態 β は, 攻撃者によって $TK^{'}$ を改竄された, またはタグが正常に受信出来なかった場合である (Action C に相当). そのため, タグ側の k_i が更新されず共有鍵が非同期の状態である. この時, タグとサーバが格納している ID は同期状態のため, サーバは TID を用いてタグを識別出来る. また, タグが受信した $TK^{'}$ を用いて, サーバが k_i の更新で用いた P_i を得ることが出来る. この P_i を用いてタグ側の k_i を更新して鍵の再同期化を行う.

状態 γ は, Action B と Action C が交互に連続して行われ, タグとサーバが共に格納している ID_i と k_i が非同期の状態である. この時, $H(P_i) = k_i$ が成

表 3: 他の方式との安全性における比較

セキュリティ方式	セキュリティ要件			
	IND	BU	Spf	SP
LOK Protocol[5]	√	√	−	−
CC Protocol[1]	−	−	√	−
Our Protocol	√	√	√	√

(√: 要件を満たす, −: 要件を満たさない)

り立つので, サーバは P_i を用いて TK と一致する値を計算出来る. したがって, サーバはタグの識別が可能である. また, ID と k_i は状態 α, 状態 β と同様の再同期化を行い, 正常の状態に戻る.

Server Impersonation Attack (SIA) : タグ T_i が格納する秘匿情報は ID_i, k_i, N_i であり, 正規サーバとは同期状態 (状態 0) であると仮定する. 攻撃者は, T_i が格納する秘匿情報を入手し, それらを格納した偽のサーバ F を作成する. F は T_i に乱数 r を生成して $Query$ と共に送信する. $Query$ を受け取った T_i は格納している秘匿情報を用いて TID, TK, N_i を計算して F へ返信する. T_i からの返信を受け取った F は, T_i の k_i を更新するための $TK^{'}$ を計算する. しかし, F は P_i を格納していないため, 有効な $TK^{'}$ を計算出来ない. T_i は, $TK^{'}$ から得られる $S^{'}$ を用いて k_i と比較するが一致しないため, タグの k_i を更新しない. 攻撃後, タグの ID_i が更新されてしまう為, タグとサーバの同期は状態 α となる. しかし, 提案方式は状態 α 時でもサーバはタグを識別可能である. したがって, 提案方式は SIA に対して安全であると言える.

4.4 従来方式との比較

本節では, 提案方式と 3 章で扱った従来方式との安全性の比較を行う. 表 3 では, 2.2 章で述べた四つのセキュリティ要件と 2.3 節で述べた同期問題に対する安全性を示す. LOK 方式は, タグが格納している ID を暗号変換せずに送信しているため, Active Attack に対して脆弱である. CC 方式は, タグの ID が更新されなかった場合, タグの出力値が前回の出力と同じ値になる場合が存在するため, 識別不可能性とフォワードセキュア性を満たさない. また, これらの方式は ID や秘密鍵をタグとサーバで共有する方式であるため, SIA に対して脆弱である. しかし, 提案方式は 4.3 章で述べた通り, BA や SIA に対して安全であり, 2.2 章で記述したセキュリティ要件も全て満たしている.

5 まとめ

本稿では, 従来のセキュリティ要件を満たし, かつ, サーバとタグのデータ同期を保つプロトコルを提案した. 提案方式では, サーバに秘密鍵を格納し, タグに秘密鍵を一方向性ハッシュ関数で暗号化した値を格納する手法を用いた. 仮に攻撃者がタグの秘匿情報を入手しても, 一方向性ハッシュ関数で暗号化されているため, 入手したタグの秘匿情報からサーバの秘密鍵を計算出来ない. 攻撃者は正規サーバのなりすましに失敗するので, 提案方式はサーバなりすまし攻撃に対して安全であると言える. また, タグとサーバに同期化した鍵を格納することでデータ同期を保ち, 一方向性ハッシュ関数と乱数を用いることで識別不可能性, 追跡不可能性, なりすまし攻撃耐性を満たすことが出来る.

参考文献

[1] S. Canard, and I. Coisel, “Data Synchronization in Privacy-Preserving RFID Authentication Schemes”, RFIDsec 2008. http://events.iaik.tugraz.at/RFIDSec08/Papers/

[2] T. Dimitrios, “A Lightweight RFID Protocol to protect against Traceability and Cloning attacks”, SecureComm 2005, IEEE Computer Society Press, pp.59-66, 2005.

[3] A. Juels, “RFID security and privacy: A research survey”, IEEE Journal on Volume 24, pp.381-394, 2006.

[4] J. Kang, and D. Nyang, “RFID Authentication Protocol with Strong Resistance against Traceability and Denial of Service Attacks”, ESAS 2005, LNCS 3813, pp.164-175, Springer-Verlag, 2005.

[5] J. Lim, H. Oh, and S. Kim, “A New Hash-Based RFID Mutual Authentication Protocol Providing Enhanced User Privacy Protection”, ISPEC 2008, LNCS 4991, pp.278-289, 2008.

[6] M. Ohkubo, K. Suzuki, and S. Kinoshita, “Criptographic Approach to *Privacy-Friendly* Tags”, RFID Privacy Workshop 2003, 2003. http://rfidprivacy.media.mit.edu/2003/papers/

[7] K. Rhee, J. Kwak, S. Kim, and D. Won, “Challenge-Response based RFID Authentication Protocol for Distributed Database Environment”, SPC 2005, LNCS 3450, pp.70-84, 2005.

[8] M. R. Rieback, B. Crispo, and A. S. Tanenbaum, “The evolution of RFID security”, PerCom’06, IEEE Pervasive Computing vol.5, pp. 62-69, 2005.

[9] B. Song, “Server Impersonation Attacks on RFID Protocols”, UBICOMM 2008, IEEE Computer Society, pp.50-55, 2008.

[10] B. Song, “RFID Tag Ownership Transfer”, RFIDSec 2008, 2008. http://events.iaik.tugraz.at/RFIDSec08/Papers/

[11] B. Song, and C. J. Mitchell, “RFID Authentication Protocol for Low-cost Tags”, WiSec 2008, Association for Computing Machinery, pp.140-147, 2008.

[12] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, “Security and Privacy Aspects of Low-Cost Radio-Frequency Identification Systems”, SPC 2003, LNCS 2802, pp.201-212, 2003.

[13] C. C. Tan, B. Sheng, and Q. Li, “Serverless Search and Authentication Protocols for RFID”, PerSec 2007, 5th Annual IEEE International Conference, pp.235-240, 2007.