SCIS 2007 The 2007 Symposium on Cryptography and Information Security
Sasebo, Japan, Jan. 23-26, 2007
The Institute of Electronics, Information and Communication Engineers

RFID プライバシ保護方式に対する一考察 —OSK/AO 方式の評価—
A Study on a privacy protection method of RFID —Evaluation of the OSK/AO method—

阿部 正己* 尾形 わかは*
Masaki Abe Wakaha Ogata

あらまし 将来的に, 更なる RFID タグの普及への障害となりうる RFID プライバシ問題を解決する方法の一つとして，タグを内部更新し, 出力を毎回可変にする OSK 方式が大久保, 鈴木, 木下によって提案された [1]．しかし，この方式はデータベース側の負荷が高く, スケーラビリティに課題があった．そこで, OSK 方式よりスケーラビリティが高い OSK/AO 方式が Avoine, Oechslin によって提案された [5]．本稿では, この OSK/AO 方式を実行するための具体的なアルゴリズムを示し, それに基づいて効率の再評価を行い, 基本の OSK 方式と比較し, OSK/AO 方式の問題点を指摘する. 更に, その効率改善策を提案する.

キーワード RFID, OSK(Extended Hash-chain), Time-memory Trade-off , OSK/AO

1 はじめに

近年急速に RFID タグの普及が進み，オープンな環境におけるソリューションへの適用が期待されている．しかし，第三者が所有者に無断でタグの ID を読み取ることにより，所有者の履歴情報を収集したり，所有者の行動を追跡できてしまうというプライバシ問題があり, 将来的に, 更なる RFID タグの普及への障害となりうる．この問題を解決する方法の一つとして，タグをアクセスごとに内部更新し, 出力を毎回可変にする OSK 方式[1]が大久保, 鈴木, 木下によって提案された [1]．しかしながら，この方式はタグ数 I, 最大更新回数 K に対して，ID 認証毎にデータベース側で $O(IK)$ の計算を必要とする，もしくは, メモリを $O(IK)$ 用意して事前に $O(IK)$ の計算を必要とするため, データベース側の負荷が高く, スケーラビリティに課題があった. そこで, 暗号解読などで使われる Time-memory Trade-off 方式 [2, 3] をこの OSK 方式に適用させることで, メモリ量とデータベース側での計算量を調整できるスケーラビリティの高い OSK/AO 方式が Avoine, Oechslin らによって提案された [5].

本稿では, この OSK/AO 方式の事前計算と ID 認証の具体的なアルゴリズムを示し, それに基づいて効率の再評価を行い, 基本の OSK 方式と比較し, OSK/AO 方式の問題点を指摘する. 更に, その効率改善策を提案する.

* 東京工業大学, 〒 152-8552 東京都目黒区大岡山 2-12-1, Tokyo Institute of Technology, 2-12-1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan, amasakey@crypt.ss.titech.ac.jp, wakaha@mot.titech.ac.jp

[1] Extended Hash-chain 方式とも呼ばれる.

2 RFID システムと OSK 方式

2.1 RFID システム

RFID システムは, RFID タグ $Tag_i(0 \leq i < I)$, タグの情報を読み書きする装置 $Reader/Writer$, タグの認証をするデータベース B，からなるプロトコルである．このプロトコルは, Setup phase, Communication phase，Identification phase の 3 つの phase からなっている. Setup phase は B とタグの初期化段階であり，B は各タグ Tag_i にそれぞれ固有の ID である ID_i を割り当て, $Writer$ を通じて書き込む. また, B も ID_i を保存する. Communication phase では, $Reader$ がタグに ID 取得要求を出し, ID を得る．Identification phase では, B が $Reader$ から送られてきた ID に対して ID 認証を行う.

オープン環境では多数の RFID システムが混在し，$Reader$ が自システム以外のタグにアクセスしたり，タグが自システム以外の $Reader$ からアクセスを受けることもある. また，RFID タグは書き込み制御は可能だが, タグコストの面から, 耐タンパー性を保証することは難しく，ID の読み出し制御もない場合が多い. そのため, タグの ID 情報に関係するプライバシ問題が生じる．これは大きく以下の二つに分類される.

1. 所持品の漏洩
2. ID 追跡による行動追跡, 本人特定

この問題に対し，秘匿化 (暗号化など) した ID をタグに格納することで 1 を防ぎ, タグからの出力を可変にする

ことよって 2 を防ぐことが可能である．1, 2 を防ぎ, 更に, タグの物理的な解析に対しても, タグの送信履歴を推測できないというフォワードセキュア性も満たす安全性の高い RFID システムとして, OSK 方式が提案された [1].

2.2 OSK 方式

OSK 方式の大まかなプロトコルは次のようになる．Setup phase では, B は各タグに初期秘密情報を $Writer$ を通じて書き込む．また, B も初期秘密情報を保存する．Communication phase では $Reader$ がタグと通信し, 秘匿 ID を得る．タグは同時に秘密情報の更新をする．Identification phase では, B が $Reader$ から送られてきた秘匿 ID から ID 認証を行う．以下で，文中で用いる記号の定義を行う．

- I : B の管理するタグ数.
- K : タグの最大更新回数.
- s_i^0 : Tag_i と B に最初に保持される初期秘密情報.
- s_i^k : k 回更新された後に Tag_i が保持している秘密情報. $(0 \leq k < K)$
- r_i^k : 秘密情報 s_i^k を暗号化して生成した秘匿 ID.
- $|s|$: s_i^k と r_i^k のビット長.
  なお, $|s|$ は以下のように設定するのが望ましい.

$$|s| \geq 4 \log_2 IK \tag{1}$$

- G,H : $|s|$bit 出力のハッシュ関数
- T_{Pre} : Setup phase での事前計算量 (以後, 計算量はハッシュの計算回数で評価)
- T_{Worst}: Identification phase での最悪計算量
- M : OSK 方式のために, B で必要なメモリ量 (ID_i を保持するためのメモリ量は除く)

OSK 方式の基本方式は, ID 認証に完全探索を用いる．しかし, メモリ量に余裕がある場合 ID 認証に Lookup table を用いることで, ID 認証を高速に実行できる．以下に二通りの方式を示す. (以後, その他の方式については,OSK 方式 (完全探索) からの追加分, 変更点のみ示す.)

2.2.1 OSK 方式 (完全探索)

Setup phase

1. B は $ID_i (0 \leq i < I)$ とランダムに s_i^0 を生成する．
2. B は, s_i^0 を $Writer$ へ送信する.
3. $Writer$ は，s_i^0 を Tag_i へ書き込む.
4. B は，ID_i と s_i^0 を関連付けて保存する.

Communication phase

1. $Reader$ は, $Tag_{i'}$ に ID 取得要求を出す.
2. $Tag_{i'}$ は，現在保持している秘密情報 $s_{i'}^{k'} (0 \leq k' < K)$ から, 秘匿 ID $r_{i'}^{k'} = G(s_{i'}^{k'})$ を生成する．また秘密情報を $s_{i'}^{k'+1} = H(s_{i'}^{k'})$ のように更新する.
3. $Tag_{i'}$ は，$r_{i'}^{k'}$ を $Reader$ に送信する.
4. $Reader$ は，$r_{i'}^{k'}$ を B に送信する.

Identification phase

1. B は，全ての i, k に対して, s_i^0 を用いて $G(H^k(s_i^0))$ を算出し，受信した $r_{i'}^{k'}$ と比較する.
2. B は，$r_{i'}^{k'} = G(H^k(s_i^0))$ となる i に対する ID_i を $Reader$ に送信する．一致しなければ認証失敗とする.

この方式の必要計算量とメモリ量は以下のようになる.

$$\begin{aligned} T_{Pre} &= 0 \\ T_{Worst} &= 2IK \\ M &= I|s| \end{aligned}$$

2.2.2 OSK 方式 (Lookup table)

Setup phase

5. B は, s_i^0 からすべての秘匿 ID $r_i^k (0 \leq i < I, 0 \leq k < K)$ を求め, それらと ID_i を関連づけるため (r_i^k, i) のようなペアを生成する.
6. 上記のペアを r_i^k に関してソートして table として保存する.

Identification phase

1. B は，受信した $r_{i'}^{k'}$ と一致するものが, 事前に生成した table 内に存在するか二分探索を用いて比較する.
2. B は, $r_{i'}^{k'} = G(H^k(s_i^0))$ となる i に対する ID_i を $Reader$ に送信する．一致するものが table 内になければ認証失敗とする.

この方式の必要計算量とメモリ量は以下のようになる.

$$\begin{aligned} T_{Pre} &= 2IK \\ T_{Worst} &= \log_2 IK (\text{二分探索}) \\ M &\approx IK|s| + IK \log_2 I \end{aligned}$$

3 OSK/AO 方式

OSK 方式 (完全探索) では, 必要なメモリ量は小さいが ID 認証の際に時間がかかり，OSK 方式 (Lookup table) では, ID 認証は高速だが必要なメモリ量が大きくなるため, 利用者の環境による柔軟な適用ができず, タグ数の多い大規模システムでの運用に問題が生じる可能性がある.

そこで, 暗号解読などで使われる Time-memory Trade-off 方式 [2, 3] をこの ID 認証に適用させることで, メモリ量と ID 認証の時間を調整できる OSK/AO 方式が提案された [5].

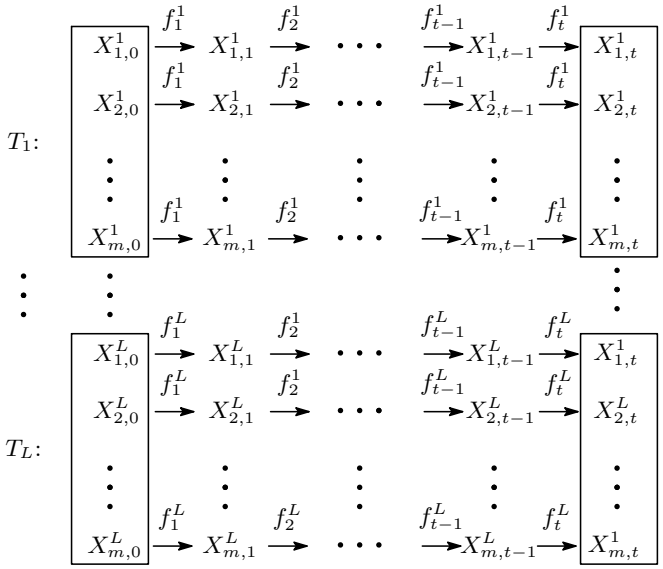

図 1: perfect rainbow table

3.1 Time-memory Trade-off

Time-memory Trade-off は暗号解読の一手法として, Hellman によって提案され [2], Oechslin によってその改良版が提案された [3].

この方式は, 完全探索, Lookup table といった従来の探索法と異なり, 使用できるメモリ量を利用者が決定でき, それに応じて, 計算量, 事前計算量が決定される. ただし, 確率的アルゴリズムであるため, 必ずしも成功しない. 成功確率は, メモリ量を増やすと増加する. 下に暗号解読を各探索法を適用し実行した場合の効率を示す. なお, Time-memory Trade-off の場合は, 典型的な値を示している.

	メモリ量	計算量	事前計算量	成功確率
完全探索	$O(1)$	$O(N)$	$O(1)$	1
Lookup table	$O(N)$	$O(1)$	$O(N)$	1
Time memory trade-off	$O(N^{\frac{2}{3}})$	$O(N^{\frac{2}{3}})$	$O(N)$	約 $\frac{1}{2}$

この表からわかるように, Time-memory Trade-off は, メモリ量, 計算量に関して完全探索と Lookup table の中間値を取ることに成功している. 更に, [3] では成功確率をより高くできる perfect rainbow table を用いた Time-memory Trade-off が提案された. この方式では, 事前計算において, 図 1 のような 長さ t の鍵の chain が m 本存在するような table を L 個生成し, 生成した table の最初と最後の列のみ保存することによりメモリ量を節約している. perfect rainbow table は, 各 table の各列の値はすべて異なるように構成されており, 成功確率が高くなる.

3.2 OSK/AO 方式

[5] では, 3.1 節の perfect rainbow table を用いた Time-memory Trade-off を OSK 方式の ID 認証に適用し, OSK/AO 方式を構成した. 以下で, OSK/AO 方式で用いる記号の定義を行う.

- P : 1 回の Communication phase と Identification phase での成功確率.
- x : perfect rainbow table 以外に B が保存する情報の量を示すパラメータ. ただし

$$1 \leq x \leq K \tag{2}$$

- m : Setup phase で生成される table 一つあたりの chain の本数を示すパラメータ.
- t : Setup phase で生成される chain の長さを示すパラメータ.
- L : Setup phase で生成される table の個数を示すパラメータ. P から以下のように定まる [4].

$$L = \left\lceil \frac{-\ln(1-P)}{2} \right\rceil$$

OSK/AO 方式を以下に示す. ただし, OSK(完全探索) からの追加分のみを示す.

Setup phase

4'. (**Preprocessing1**)s_i^0 から, $s_i^{x' \frac{K-1}{x}} (1 \leq x' < x)$ を作成する. $s_i^0, s_i^{\frac{K-1}{x}}, \ldots, s_i^{(x-1)\frac{K-1}{x}}$ を保存する.

5. (**Preprocessing2**)B は, $(i,k)(0 \leq i < I, 0 \leq k < K)$ の組を鍵とし, 関数 f による長さ t の chain が m 本存在するような table が L 個の perfect rainbow table を作成する. ただし, 適当な $R : \{0,1\}^{|s|} \mapsto (i',k')(0 \leq i' < I, 0 \leq k' < K)$ に対して

$$\begin{aligned} F &: (i,k) \mapsto r_i^k = G(H^k(s_i^0)) \\ f &: (i,k) \mapsto R(F(i,k)) \end{aligned}$$

とする.

Identification phase

1. (**Identification**)B は, 受信した $r_{i'}^{k'}$ に対し, Time-memory Trade-off を適用し, (i',k') のペアを求める. ペアを出力できないときは認証失敗とする.
2. B は, 求めた (i',k') から初期秘密情報 $s_{i'}^0$ を求め, $ID_{i'}$ を $Reader$ に送信する.

B が準備できるメモリ量が M_{AO} のとき, B における Setup phase の事前計算量 T_{Pre}^{AO} と Identification phase での平均計算量 T_{Ave}^{AO} は以下のようになる. ただし, γ は, 成功確率 P に依存する係数である. (例えば, $P = 0.999$ のとき $\gamma = 8$.)

$$\begin{aligned} T_{Ave}^{AO} &\approx \frac{3^3 I^3 K^3 (\log_2 IK)^2 |s| \gamma}{2M_{AO}^3} \\ T_{Pre}^{AO} &\approx \frac{3I^2K^2|s|}{2M_{AO}} \end{aligned} \tag{3}$$

4 OSK/AO 方式の計算量の再評価

本章では, 既存の OSK 方式と効率を比較するため, 式 (3) で示された事前計算量 T_{Pre}^{AO} の再評価を行う. また, これまで評価されていなかった ID 認証の最悪計算量 T_{Worst}^{AO} を新たに見積もる. T_{Worst}^{AO} は, Identification phase で認証プロセスが終了するまでにかかる最悪の計算量であり, 認証が失敗する場合にあたる. *Reader* が自システム以外のタグからの情報を B へ送信した場合, 必ず ID 認証が失敗するまで計算することになり, T_{Worst}^{AO} が大きい値をとる場合, システムが円滑に機能しなくなる危険性がある.

計算量の評価は, Preprocessing1, Preprocessing2, Identification として付録 A に示すものを用いるとする.

4.1 パラメータ最適化

方式の効率を評価するためには, 適用する RFID システムによって決定される値 $I, K, M_{AO}, P, |s|$ に対して, パラメータ x, m, t を決定する必要がある. 以下で, 最悪計算量 T_{Worst}^{AO} を最小にするようにパラメータを最適化する.

まず, Preprocessing1, 2 で必要なメモリ量, 成功確率 P は以下のようになる.

$$M_{AO} = I|s|x + 2(\log_2 IK)mL \quad (4)$$

$$P = 1 - (1 - \frac{m}{IK})^{tL} \approx 1 - e^{-tL\frac{m}{IK}} \quad (5)$$

式 (4),(5) より, 以下のように m, t を x で表せる.

$$m = \frac{I|s|(\frac{M_{AO}}{I|s|} - x)}{2L(\log_2 IK)} \quad (6)$$

$$t = \frac{2K(\log_2 IK)(-\ln(1-P))}{|s|(\frac{M_{AO}}{I|s|} - x)} \quad (7)$$

次に, T_{Worst}^{AO} を求める.

図 5 において, 最も計算時間が長くなるのは毎回 8 行目が成り立つが 15 行目が成り立たず (つまり false alarm[4] が毎回生じる), 最後まで認証が成功しないときである. 関数 f 1 回実行あたり, ハッシュ関数は最悪 $\frac{K}{x}$ 回実行されることに注意すると, T_{Worst}^{AO} は以下のように表せる.

$$T_{Worst}^{AO} = t(t-1)L\frac{K}{x} \approx t^2 L\frac{K}{x} = \frac{2^2K^3L(\log_2 IK)^2(-\ln(1-P))^2}{x|s|^2(\frac{M_{AO}}{I|s|} - x)^2} \quad (8)$$

したがって T_{Worst}^{AO} を最小にする x は,

$$x = \frac{M_{AO}}{3I|s|} \quad (9)$$

となる. 式 (6),(7),(9) より,

$$m = \frac{M_{AO}}{3L(\log_2 IK)} \quad (10)$$

$$t = \frac{-3\ln(1-P)IK(\log_2 IK)}{M_{AO}} \quad (11)$$

4.2 効率

4.1 節のパラメータを用いた場合の事前計算量 T_{Pre}^{AO} と最悪計算量 T_{Worst}^{AO} をアルゴリズムに基づいて示す.

式 (8),(9) より, 最小の T_{Worst}^{AO} は以下のように求められる.

$$T_{Worst}^{AO} \approx \frac{3^3I^3K^3(\log_2 IK)^2|s|L(-\ln(1-P))^2}{M_{AO}^3}$$

次に, $T_{Pre}^{AO}(= T_{Pre1}^{AO} + T_{Pre2}^{AO})$ を求める. まず T_{Pre1}^{AO} は,

$$T_{Pre1}^{AO} = I(K-1)\frac{x-1}{x} \approx IK - \frac{3I^2K|s|}{M_{AO}} \quad (12)$$

と表せる.

次に, 図 4 において, p 本目の chain を作成しているときに他の chain と衝突する確率は, 以下のように求められる.

$$1 - (1 - \frac{p-1}{IK})^t \approx 1 - e^{-t\frac{p-1}{IK}}$$

よって, T_{Pre2}^{AO} は, 以下のようになる.

$$T_{Pre2}^{AO} \approx \frac{K}{2x}L\sum_{p=1}^{m} te^{t\frac{p-1}{IK}} = \frac{K}{2x}Lt\frac{e^{\frac{mt}{IK}}-1}{e^{\frac{t}{IK}}-1} \approx \frac{K}{2x}IKL(e^{\frac{mt}{IK}} - 1) \quad (13)$$

式 (5), (9), (12), (13) より, T_{Pre}^{AO} は以下で表わされる.

$$T_{Pre}^{AO} \approx \frac{3I^2K^2L|s|}{2M_{AO}}((1-P)^{-\frac{1}{L}} - 1) + IK - \frac{3I^2K|s|}{M_{AO}} \quad (14)$$

式 (3) で示された事前計算量は, 成功確率に依存している部分等が無視されており, 式 (14) による本稿での見積もりよりかなり低いものとなっている. ($P = 0.999$ として比較した場合, 本稿の方が約 15 倍大きくなっている.)

5 OSK/AO 方式の考察

本章では, OSK/AO 方式の適用環境, 条件を考察し, 適用する RFID システムによって決定される値 I, K, $|s|$, M_{AO}, P について, 具体的な値に対する計算量を見積もることにより, OSK 方式 (完全探索/Lookup table) との性能を比較する.

5.1 OSK/AO 方式適用環境, 条件

タグ数 I の条件 I が小さい場合には, OSK 方式 (完全探索 or Lookup table) で十分であるため, I が非常に大きい大規模システムに適用する場合を想定する.

最大更新回数 K の条件 B が, 定期的に Setup phase を行うと仮定し, その期間内にタグが *Reader* によってアクセスされるであろう回数を設定する.

表 1: 各方式の性能評価

	メモリ量	最悪計算時間	事前計算時間	成功確率
完全探索	2.3GB	3.4hours	0	1
OSK/AO	6.9GB	107.4s	336.6days	0.999
	16GB	8.1s	142.0days	
	64GB	0.13s	35.5days	
	256GB	1.97ms	8.9days	
	1TB	30.8μs	2.2days	
Lookup table	1.3TB	fast	3.4hours	1

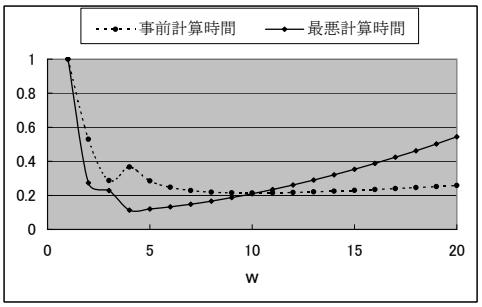

図 2: 繰り返しアクセスによる改善

秘密情報のビット長 $|s|$ の条件　$|s|$ は, 式 (1) より,

$$|s| = 64 \left\lceil \frac{4 \log_2 IK}{64} \right\rceil$$

と設定すると, タグ容量として区切りがよい.

メモリ量 M_{AO} の条件　$M_{AO} \geq IK|s| + IK \log_2 I$ のメモリ量を確保できるのであればOSK方式 (Lookup table) を用いることができるという事実と式 (2),(9) より, M_{AO} は次式を満たすとする.

$$3I|s| \leq M_{AO} \leq IK|s| + IK \log_2 I \qquad (15)$$

成功確率 P の条件　$P = 1$ とした場合, パラメータ t, L が無限大に大きくなってしまうため, ID 認証が失敗する確率を 0 とすることができない. そのため, P をできる限り 1 に近い値とすることで対応する.

5.2　評価

$I = 3 \cdot 2^{25}$(約一億個), $K = 2^9$(約 500 回), $|s| = 192bit, P = 0.999$ とし, 一秒当たり hash を 2^{23} 回計算できるとして, 式 (15) を満たすいくつかの M_{AO} に対して各方式で必要とされるメモリ量と計算時間を見積もると表 1 のようになる.

表 1 より, OSK/AO 方式のメモリ量, 最悪計算時間は, 完全探索と Lookup table 方式の中間的な値になっているが, 事前計算時間がかなり大きくなり, 大規模システムに OSK/AO 方式を用いるのは, このままでは困難であることがわかる.

6　OSK/AO 方式の効率の改善

これまで, B が期待するタグの ID 認証の成功確率 (P' とおく) は, 1 回の Communication phase と Identification phase での成功確率 P で評価してきた (つまり, $P' = P$ であった) . そのため, P には非常に高い成功確率が要求され, その結果, 5 章で示したように事前計算量が膨大となった.

改善方法として, Identification phase に失敗した場合には再びタグにアクセスし, 異なる秘匿 ID を受け取り, 再度 Identification phase を実行することを考える. アクセスの最大回数を $w(\geq 1)$ とした場合, w 回 Communication phase と Identification phase を行い 1 回でも Identification phase が成功すれば, ID 認証が成功となるので, P は P' から以下の式で決定される.

$$P = 1 - (1 - P')^{\frac{1}{w}}$$

従って, P の値は, これまでよりは低い値でよい.

ここで, 実際のタグの ID 認証の最大回数を K' とおく. これまでは, $K' = K$ であった. しかし, ID 認証が成功するまで最大 w 回タグへのアクセスが実行される場合, タグの更新回数 K は K' より大きく設定する必要がある. ID 認証が成功するまでのタグへのアクセス数の期待値 v は, 以下のようになる.

$$\begin{aligned} v &= P + 2P(1-P) + \cdots + wP(1-P)^{w-1} + w(1-P)^w \\ &= \frac{1-(1-P)^w}{P} = \frac{P'}{1-(1-P')^{\frac{1}{w}}} \end{aligned}$$

従って, $K = K'v$ とすればよい. このような条件のもと, 適用する RFID システムによって決定された P', I, K', M_{AO} に対して, システムの期待する計算時間になるよう w の値を決定すればよい. (ただし, 最悪計算量は Identification phase を w 回行ってすべて失敗する場合について評価することに注意.)

5 章の数値例の場合において, $1 < w \leq 20$ として事前計算時間と最悪計算時間を見積もった結果を図 2 に示す. なお, $w = 1$ が拡張前の方式を表わし, この場合の計算時間を基準にしてグラフ化した. 図 2 より, $w = 5 (K = 1.33 \cdot 2^9, |s| = 192bit, P = 0.75)$ のとき最悪計算時間が約 $\frac{1}{9}$, 事前計算時間が約 $\frac{2}{7}$ になっており, 効率が改善されているのがわかる.

7　まとめ

本稿では, [5] のアイデアを基に, OSK/AO 方式実行のための具体的なアルゴリズムに基づき事前計算量と最悪計算量を再評価した. その結果, 従来の OSK 方式の適用が困難であるような, 大規模システムに OSK/AO

方式を適用することを想定した場合, 事前計算時間が非常に大きくなってしまうという問題点が生じることがわかった. 最後にその改善策を提案した.

参考文献

[1] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita, “Cryptographic approach to“ privacy-friendly ” tags,” In RFID Privacy Workshop, MIT, Massachusetts, USA, November 2003.

[2] Martin Hellman, “A cryptanalytic time-memory trade off,” IEEE Transactions on Information Theory, IT-26(4):401–406, July 1980.

[3] Philippe Oechslin, “Making a faster cryptanalytic time-memory trade-off,” Advances in Cryptology, proceedings of Crypto 2003, Lecture Notes in Computer Science 2729, Springer-Verlag, pp. 617–630, 2003.

[4] Gildas Avoine, Pascal Junod, and Philippe Oechslin, “Time-memory trade-offs: False alarms detection using checkpoints (Extended Version),” Available online on http://lasecwww. epfl. ch/pub/lasec/doc/AJO05a. pdf, 2005.

[5] Gildas Avoine and Philippe Oechslin, “A scalable and provably secure hash based RFID protocol,” In International Workshop on Pervasive Computing and Communication Security . PerSec 2005, pages 110–114, Kauai Island, Hawaii, USA, March 2005. IEEE, IEEE Computer Society Press.

[6] Serge Vaudenay,「A Classical Introduction to Cryptography」, Springer, 2006.

A 評価に使用するアルゴリズムの詳細

計算量の評価に使用する具体的なアルゴリズムを図3,4,5 に示す.

Input:H, K, I, $s_i^0(0 \le i < I)$, x
Output:T_{info}^q

for i'=0 to $I-1$ **do**
  $s \leftarrow s_{i'}^0$
  insert(s) in table T^0
  for q=1 to $x-1$ **do**
    for z=1 to $\lfloor \frac{K-1}{x} \rfloor$ **do**
      $s \leftarrow H(s)$
    end for
    insert(s) in table T_{info}^q
  end for
end for

図 3: Preprocessing1

Input:G, H, P, M_{AO}, I, K, m, t, L, $T_{info}^q(0 \le q < x)$
Output:F, R_j^l, f_j^l, T_l

define function F
  $F : (i,k)(0 \le i < I, 0 \le k < K) \mapsto$
    $r_i^k = G(H^{k-\lfloor \frac{K-1}{x} \rfloor q}(s_i^{\lfloor \frac{K-1}{x} \rfloor q})$
    $q \leftarrow k/\lfloor \frac{K-1}{x} \rfloor$
    get the $(s_i^{\lfloor \frac{K-1}{x} \rfloor q})$ from T_{info}^q
for l=1 to L **do**
  for j=1 to t **do**
    pick a reduction function R_j^l
     at random and define
     $R_j^l : r_i^k (= F(i,k)) \mapsto (i', k')$
     $(0 \le i' < I, 0 \le k' < K)$
     $f_j^l : (i,k) \mapsto R_j^l(F(i,k))$
  end for
end for
for l=1 to L **do**
  for p=1 to m **do**
    repeat
      repeat
        pick(i', k')at random
        $(0 \le i' < I, 0 \le k' < K)$
      until T_l contains no (*,(i', k')) entry
      $(i,k) \leftarrow (i', k')$
      for j=1 to t **do**
        compute $(i,k) \leftarrow f_j^l(i,k)$
      end for
    until T_l contains no $((i,k)$,*)entry
    insert$((i,k)$, $(i', k'))$ in table T_l
  end for
end for

図 4: Preprocessing2

Input: $y = G(H^{k_0}(s_{i_0}^0))(= r_{i_0}^{k_0})$
Output: (i', j') or $\perp$

for l=1 to L **do**
  for j=t down to 1 **do**
    set u to 1
    set (i,k) to $R_j^l(y)$
    for p=$j+1$ to t **do**
      $(i,k) \leftarrow f_p^l(i,k)$
    end for
    if T_l contains $((i,k)$,*)entry **then**
      get $((i,k)$, $(i', k'))$entry from T_l
      while $r_{i'}^{k'} \neq y$ and $u < j$ **do**
        increment u
        $r_{i'}^{k'} \leftarrow F(i', k')$
        $(i', k') \leftarrow R_u^l(r_{i'}^{k'})$
      end while
      if $r_{i'}^{k'} = y$ **then**
        return (i', k') /* succeed */
      end if
    end if
  end for
end for
return $\perp$ /* fail */

図 5: Identification