リンク不能性を持つID照合システムの実装に向けて

中村, 徹
九州大学

野原, 康伸
九州大学

馬場, 謙介
九州大学

井上, 創造
九州大学

他

SCIS 2006 The 2006 Symposium on Cryptography and Information Security
Hiroshima, Japan, Jan. 17-20, 2006
The Institute of Electronics, Information and Communication Engineers

リンク不能性を持つID照合システムの実装に向けて
Towards Implementation of Unlinkable Identification Scheme

中村 徹* 野原 康伸* 馬場 謙介* 井上 創造* 安浦 寛人*
Toru Nakamura Yasunobu Nohara Kensuke Baba Sozo Inoue Hiroto Yasuura

あらまし 近年 RFID タグ、非接触型 IC カードなどに代表される RFID デバイスが急速に普及している。しかし、第三者が様々な場所でユーザの ID を読み取り、ID を元にユーザの行動を追跡できる問題がある。この問題の解決策として Randomized Hash Lock 方式が提案されている。だがこの手法では、ID 照合の度にサーバにおいて $O(N)$ 回 (N はデバイスの数) ハッシュ計算を行う必要があった。K 段 ID 照合方式は、ID を木構造にすることでサーバにおけるハッシュ計算回数を $KN^{\frac{1}{K}}$ にする。K 段 ID 照合方式では、K を増加させることによりサーバの処理時間を短くすることができる。一方で K を増加させるとデバイスの処理時間は増加する。本稿では、実装結果を元に N 及びサーバの性能を変化させたときの総処理時間を最小にする K について導出を行い、大規模 RFID システムへの適用について考察する。

キーワード RFID セキュリティ、リンク不能性、ID 照合

1 はじめに

近年 RFID(Radio Frequency IDentification) と呼ばれる微小な無線チップを持ったデバイスを用いて ID 照合を行い、人やモノを識別、管理する技術が注目されている。ID 照合とは、RFID デバイスの出力とサーバの持つ ID リストを比較し、RFID デバイスを識別することである。RFID は社会の情報化・自動化を推進するための基盤技術として、非常にユーザ数の多いシステムで利用される可能性がある。

しかし一方で、第三者が様々な場所でユーザの ID を読み取り ID を元にユーザの行動を追跡できるというロケーションプライバシ問題がある。この問題を解決するために、ある 2 つのデバイスの出力を得た者が、その出力が同一のデバイスによるものであるかどうかを判別できないというリンク不能性という概念が重要になる。

リンク不能性を実現するハッシュ関数を用いた方法の一つとして Randomized Hash Lock 方式 [1] があるが、サーバが ID 照合の度に $O(N)$ 回 (N はデバイスの数) ハッシュ計算を行う必要があるため、N が大きくなると大規模システムに適用することは困難であった。K 段 ID 照合方式 [3, 4] ではサーバの計算量を $KN^{\frac{1}{K}}$ にする。さらに K を増加させることによりサーバの処理時間を短くすることができる。しかし K を増加させるとデバイスの処理時間は増加する。現実的にはサーバとデバイスの計算能力には大きな隔りがあるので、サーバとデバイスの負荷のバランスを考慮し適切な K を選択する必要がある。

本稿では、実装結果を元に N 及びサーバの性能を変化させたときの総処理時間を最小にする K について導出を行い、大規模 RFID システムへの適用について考察する。

本稿の構成は以下の通りである。2 章でハッシュ関数を用いた ID 照合について述べ、3 章で K 段 ID 照合方式について説明する。4 章で実装結果、考察を述べ、5 章で本稿をまとめる。

2 ハッシュ関数を用いた既存の ID 照合方式

本章では、ハッシュ関数を用いた既存の ID 照合方式について述べる。

ハッシュ関数を用いた ID 照合方式は、RFID デバイスに実装する回路が暗号回路よりも回路規模の小さなハッシュ関数ですむ。このため、デバイスコストに対する制約が大変厳しい RFID タグにも適した方式である。なお、本稿でいうハッシュ関数とは (1) 逆計算が困難である、(2) 対衝突耐性があるという性質を持つ関数である。

ハッシュ関数を用いた既存の ID 照合方式としては、Randomized Hash Lock 方式 [1] や Hash-Chain 方式 [2] がある。本稿では、Randomized Hash Lock 方式についてのみ説明する。

* 九州大学 〒 816-8580 福岡県春日市春日公園 6-1 Kyushu University 6-1, Kasuga-koen, Kasuga-shi, Fukuoka, Japan. {toru,nohara,baba,sozo,yasuura} @c.csce.kyushu-u.ac.jp

2.1 Randomized Hash Lock 方式

デバイス M_i は ROM 上に ID(id_i)、ハッシュ関数 H、擬似乱数生成器を持つ。サーバは、全デバイスの ID 集合 $id_i(1 \leq i \leq N)$ を保持する。デバイスからサーバへの ID 情報の送信は、以下のプロトコルにしたがって行われる (図 1 参照)。

STEP1: M_i は乱数 R を生成し、サーバに $X = H(id_i \parallel R)$[1] と R を送る。

STEP2: サーバはすべての ID について $H(id_i \parallel R)$ を計算する。デバイスから送られてきた X と一致する $H(id_i \parallel R)$ に対応する id_i がデバイスの ID となる。

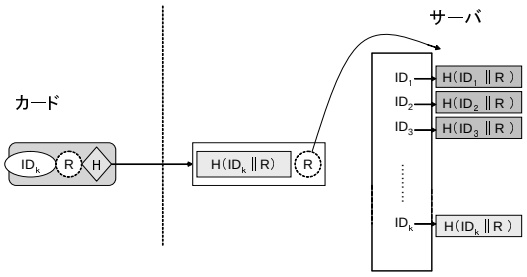

図 1: Randomized Hash Lock 方式による ID 照合

送信されるハッシュ値 X は、R が変化するため固定ではなく、出力値を追跡することは第三者にとって非常に困難である。ゆえにこの方式は第三者に対してリンク不能性を持つといえる。

2.2 問題点

Randomized Hash Lock 方式では、ユーザ数 N に対してハッシュ計算回数が $O(N)$ 必要である。これは、Hash-Chain 方式も同様である。ゆえに大規模システムに適用するのは困難である。

3 K 段 ID 照合方式

Randomized Hash Lock 方式では、サーバのハッシュ計算回数が $O(N)$ であるため大規模システムに適用するのは困難であった。本章ではサーバのハッシュ計算回数を $O(\log N)$ に削減する K 段 ID 照合方式 [3, 4] について述べる。

3.1 ID の構造

図 2 に示すような深さ K のラベル付き木を考える。木は N 個の葉を持ち、各葉が各 RFID デバイス $i(1 \leq i \leq N)$ に対応している。ある同じ親を持つ子達には、それぞれユニークなラベルが付けられる。根から葉に向かって進んで行った場合の各ラベルを集めたものが、その葉に対応するデバイスの持つ ID(id_i) となる。以下、RFID デバイス i の $k(1 \leq k \leq K)$ 番目のラベルを $id_i[k]$ と表記する。デバイス i は、$ID_i = (id_i[1], \cdots, id_i[K])$ を保存する。サーバはラベル付き木を木構造のまま保存しておく。

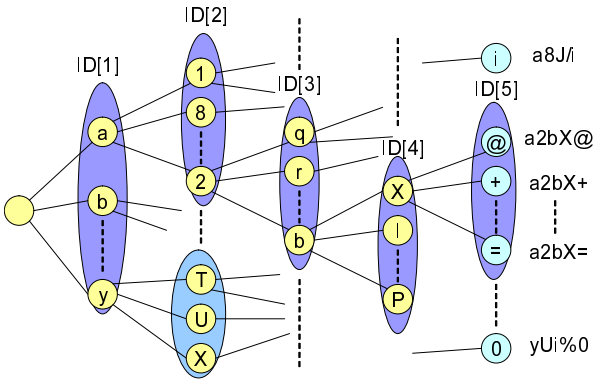

図 2: K 段 ID 照合方式における ID の構造

3.2 プロトコル

K 段 ID 照合方式におけるデバイスからサーバへの ID 情報の送信は、以下のプロトコルに従って行われる (図 3 参照)。

STEP1: M_i は乱数 R を生成し、$X_1, X_2, \cdots, X_k$ を計算する。ただし、$X_i = H(subID_i[i] \parallel R)$ である。M_i はサーバに $(R, X_1, X_2, \cdots, X_k)$ を送る。

STEP2: サーバは以下の処理を行う：

STEP2-1: Z を ID の根とし、$k \leftarrow 1$ とする。

STEP2-2: Z のすべての子 L_i について $H(L_i \parallel R)$ を計算する。デバイスから送られてきた X_k と一致する $H(L_i \parallel R)$ を検索し、対応する L_i を Z に代入する。

STEP2-3: Z が葉であれば、ID の根から Z までのラベルを集めたものがデバイスの ID となる。そうでなければ $k \leftarrow k+1$ として STEP2-2 に戻る。

$K = 1$ としたときの K 段 ID 照合方式は、Randomized Hash Lock 方式と同一のものとなる。

3.3 処理時間の見積もり

K 段 ID 照合方式において、1 回の ID 照合にサーバが要する処理時間について述べる。

K 段 ID 照合方式では、全てのノードが持つ子の数が等しく同じ場合に、サーバにおけるハッシュ関数の計算量が最小となる [3, 4]。このときのサーバにおけるハッシュ関数の計算量は、$KN^{\frac{1}{K}}$ である。なお、$K = \log N$ とす

[1] $\parallel$ はデータの連結を表す

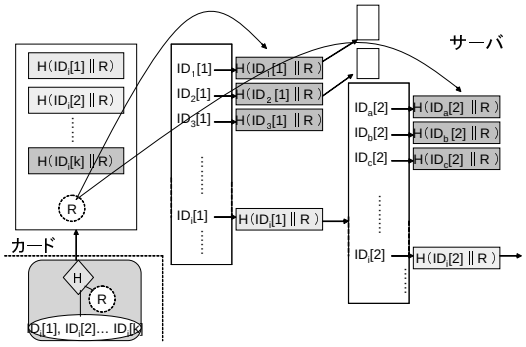

図 3: K 段 ID 照合方式による ID 照合

れば、サーバにおけるハッシュ関数の計算量は $O(\log N)$ となる。

サーバにおける処理時間では、ハッシュ関数の計算時間が支配的であると考えられる。ゆえに、サーバの 1 回のハッシュ処理時間を β_s とすると、サーバの処理時間 t_s は以下のように見積もることができる。

$$t_s = \beta_s K N^{\frac{1}{K}}$$

次にデバイスにおける処理時間について考える。K に比例してデバイスにおけるハッシュ関数の計算量とサーバ・デバイス間の通信量は増大するため、デバイスにおける計算時間と通信時間の和は K に比例して増加する (傾きを a とする)。一方、K に関わらず乱数生成等に要する時間は一定である (定数時間を b とする)。よってデバイスにおける処理時間は、以下のように見積もることができる。

$$t_d = aK + b$$

4 評価

今回の実験の目的を以下に示す。

- デバイスに K 段 ID 照合方式を実装し処理時間を計測し、a、b を求める。
- サーバのハッシュ計算時間を計測し、β_s を求める。

そこで、まずデバイスに K 段 ID 照合方式を実装し K を変化させて、サーバからコマンドを送りレスポンスを受け取るまでの時間を測定する。さらに、サーバ側で 1 回の ID 照合にかかる処理時間を計測する。

4.1 実験環境

実験環境を以下に示す。

- ハッシュ関数：SHA-1
- 乱数の長さ：4byte
- ID の長さ：$28 \times K$byte
- 実行時間は 10 回の測定の平均をとる。
- デバイス (Cyberflex Access e-gate 32K)
  - 接触式
  - OS:JavaCard
  - ROM:96Kbyte
  - RAM:4Kbyte
  - EEPROM:32Kbyte
  - 暗号化コプロセッサ
  - 乱数生成器 (乱数生成アルゴリズム:Cryptographically secure random number generation algorithms)
- サーバ
  - OS:Linux Fedora Core4(kernel-2.6.13)
  - コンパイラ:GCC 4.0.1(O3 option)
  - CPU:Pentium4 1.7GHz
  - メモリ:1Gbyte

4.2 実験結果

K を変化させたときのデバイスの処理時間を表 1 に示す。この結果から、最小二乗法よりカードの処理時間 t_d

表 1: K を変化させたときのデバイスの処理時間

K	1	2	3	4	5
t_d(sec)	0.6333	0.9085	1.1950	1.4835	1.7638

を一次関数で表すと、

$$t_d = 0.2836K + 0.3465$$

となる (相関係数 1.00)。

また $K = 1$、ID が 100 万件のときの照合時間は、目的の ID が最後に見つかったとすると、1.788 秒であった。処理時間の内訳を調べるために 100 万回のハッシュ計算を行ったところ、1.785 秒かかった。以上より ID 照合時間はハッシュ計算時間が支配的であるといえる。ゆえにサーバ側が 1 回のハッシュ計算に必要とする時間は 1.785×10^{-6} 秒といえる。そこで 4.3 章に示した式によりサーバの処理時間 t_s を以下のように表すことができる。

$$t_s = 1.785 \times 10^{-6} \cdot K N^{\frac{1}{K}}$$

以上の結果を表 2、図 4 にまとめる。

表 2: $N = 2^{20}$ のときの総処理時間の内訳

K	カード側 (sec)	サーバ側 (sec)	総処理時間 (sec)
1	0.6333	1.8717	2.5050
2	0.9085	0.00366	0.9122
3	1.1950	0.00054	1.1956
4	1.4835	0.00023	1.4837
5	1.7638	0.00014	1.7639

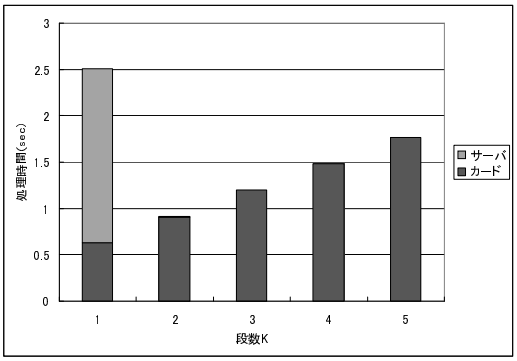

図 4: $N = 2^{20}$ のときの総処理時間の内訳

4.3 考察

K 段 ID 照合を用いたとき、K を増加させることによりサーバ側のハッシュ計算回数を削減することができるが、デバイス側のハッシュ計算回数は増加する。デバイス側のハッシュ計算時間はサーバ側のハッシュ計算時間に比べて非常に大きいので、K を増加させても必ずしも総処理時間が削減されるわけではない。今回の実験で想定したデバイスの数 $N = 2^{20}$ の場合、$K = 1$ のときサーバの処理時間が占める割合は全体の約 75 %であったが $K = 2$ のときサーバの処理時間が占める割合は全体の約 0.4 %となり、これ以上 K を増加させると処理時間は悪化する。

また先の実験結果により K 段 ID 照合に必要な総処理時間 T は以下のように表される。

$$T = 1.785 \times 10^{-6} \cdot KN^{\frac{1}{K}} + 0.2836K + 0.3465 \quad (1)$$

この T を最小にする K' が最適な K である。

また (1) 式を元に N、K を変化させたときのシミュレーションを行った。結果を表 3、図 5 に示す。現実的にはデバイスの数が非常に大きくなるとサーバ側のメモリアクセス等、他のパラメータによる処理時間の影響があると考えられるが、本稿では考慮しないものとする。

この結果によりデバイスの数が非常に大きくなった場合でも計算時間を抑えられていることがわかる。またデバイス数が増加するにつれ、処理時間が最小になる K の値は増加する。

次にサーバの演算速度を 10 倍にしたとき及び 1/10 にしたときの結果と比較する (表 4)。この結果はそれぞれハッシュ計算時間を 1/10、10 倍にしたときのシミュレーションの結果によるものである。この結果によりサーバの性能が低い場合、K を大きくすることにより処理時間を改善することができることがわかる。

さらに K 段 ID 照合を用いたときのリンク不能度 [4] の低下について述べる。定義より、あるユーザに対するリンク不能度 U_{user} は

$$\begin{aligned} U_{user} &= \log_2 N \\ &+ \frac{N-1}{N} \cdot \log_2(N^{\frac{1}{K}} - 1) \\ &- \frac{N-1}{N} \cdot \frac{N^{\frac{1}{K}}}{K(N^{\frac{1}{K}} - 1)} \log_2 N \end{aligned}$$

となる。この式を用いて処理時間が最小になる K を用いたときと $K = 1$、すなわち Randomized Hash Lock 方式とのリンク不能度の差を比較する (表 5)。この結果より Randomized Hash Lock 方式に対する K 段 ID 照合方式を用いた場合のリンク不能度の低下は非常に小さいといえる。

表 5: 処理時間が最小になる K を用いたときのリンク不能度の低下

N	$K = 1$	最適な K
2^{10}	9.9888	$9.9888(K = 1)$
2^{20}	19.9999	$19.9888(K = 2)$
2^{30}	30.0000	$29.9995(K = 2)$
2^{40}	40.0000	$39.9986(K = 3)$
2^{50}	50.0000	$49.9976(K = 4)$
2^{60}	60.0000	$59.9995(K = 4)$
2^{70}	70.0000	$69.9991(K = 5)$
2^{80}	80.0000	$79.9986(K = 6)$
2^{90}	90.0000	$89.9995(K = 6)$
2^{100}	100.0000	$99.9992(K = 7)$

5 終わりに

本稿では、実装結果を元に N 及びサーバの性能を変化させたときの総処理時間を最小にする K について導出を行い、大規模 RFID システムへの適用について考察した。

謝辞

本研究は平成 14-18 年度科学研究費補助金学術創成研究・課題番号 14GS0218 によるものである。

表 3: N、K を変化させたときの総処理時間 (太字は最小値)

	K							
N	1	2	3	4	5	6	7	8
2^{10}	**0.6319**	0.9138	1.197	1.481	1.765	2.048	2.332	2.615
2^{20}	2.502	**0.9174**	1.198	1.481	1.765	2.048	2.332	2.615
2^{30}	1.917e3	**1.031**	1.203	1.482	1.765	2.048	2.332	2.615
2^{40}	1.963e6	4.657	**1.253**	1.488	1.767	2.049	2.332	2.616
2^{50}	2.010e9	120.7	1.754	**1.522**	1.774	2.052	2.333	2.616
2^{60}	2.058e12	3834	6.812	**1.715**	1.801	2.059	2.336	2.618
2^{70}	2.107e15	1.227e5	57.80	2.804	**1.911**	2.083	2.344	2.621
2^{80}	2.158e18	3.925e6	571.7	8.968	2.349	**2.159**	2.366	2.630
2^{90}	2.210e21	1.256e8	5751	43.83	4.104	**2.399**	2.424	2.650
2^{100}	2.263e24	4.019e9	5.796e4	241.1	11.12	3.162	**2.581**	2.698

参考文献

[1] Stephan A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels, “Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems”, International Conference on Security in Pervasive Computing 2003,LNCS 2802,pp.201-212,2004.

[2] Miyako Ohkubo, Koutarou Suzuki and Shingo Kinoshita, “ Cryptographic Approach to a Privacy Friendly Tag”, RFID Privacy Workshop@MIT, 2003.

[3] 野原 康伸, 井上 創造, 馬場 謙介, and 安浦 寛人, “ リンク不能性を実現し大規模 RFID システムに適用可能な ID 照合プロトコル” , 2005 年 暗号と情報セキュリティシンポジウム (SCIS2005) 予稿集, Vol.3, pp.1567-1572, Jan. 2005.

[4] Yasunobu Nohara, Sozo Inoue, Kensuke Baba, and Hiroto Yasuura, “Quantitative Evaluation of Unlinkable ID Matching Schemes” , WPES(Workshop on Privacy in the Electronic Society) , pp.55-60, Nov. 2005.

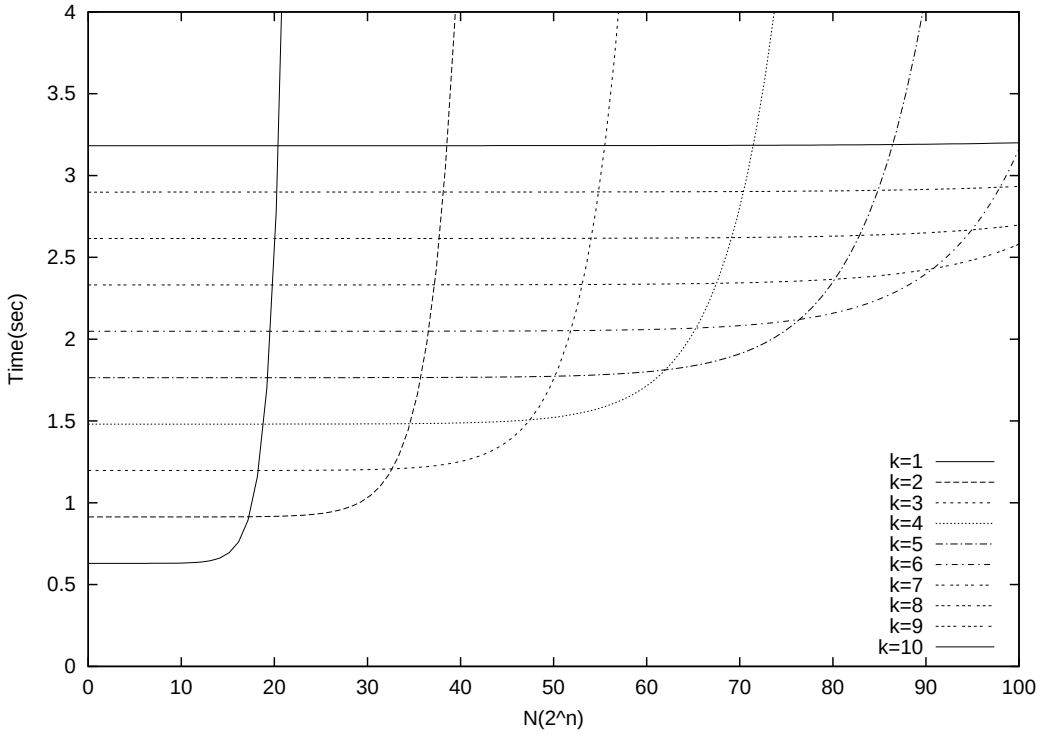

図 5: N、K を変化させたときの総処理時間

表 4: サーバの性能を変化させたときの最小の処理時間及び K の比較

N	10 倍の性能 (sec)	本実験で用いたサーバ (sec)	1/10 の性能 (sec)
2^{10}	0.6303(K=1)	0.6319(K=1)	0.6484(K=1)
2^{20}	0.8173(K=1)	0.9174(K=2)	0.9502(K=2)
2^{30}	0.9254(K=2)	1.0307(K=2)	1.2521(K=3)
2^{40}	1.2028(K=3)	1.2526(K=3)	1.5540(K=4)
2^{50}	1.2530(K=3)	1.5223(K=4)	1.8559(K=5)
2^{60}	1.5043(K=4)	1.7149(K=4)	2.1301(K=5)
2^{70}	1.6133(K=4)	1.9107(K=5)	2.3963(K=6)
2^{80}	1.8230(K=5)	2.1586(K=6)	2.6761(K=7)
2^{90}	1.9985(K=5)	2.3990(K=6)	2.9631(K=8)
2^{100}	2.1595(K=6)	2.5813(K=7)	3.2543(K=9)