社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－MPS－59 (8)
2006／5／25

ハッシュを用いたID照合における計算時間についての考察

中村徹[1], 馬場謙介[2], 安浦寛人[2]
[1] 九州大学大学院システム情報科学研究府
[2] 九州大学大学院システム情報科学研究院
〒816-8580 福岡県春日市春日公園 6-1

概要 近年，ICカードなどの小型デバイスを用いた電子的な認証が広く行われるようになり，暗号化として処理の軽い一方向ハッシュ関数が用いられることがある．しかし，ハッシュ化した電子的な情報による識別に必要な時間は，識別する対象の候補の数に比例し，大規模なシステムには適用できない．本稿では，識別とそれに対する攻撃の定式化を行い，ハッシュを用いた識別の計算時間について考察を行う．

On Computing Time of ID-Matching with Hush Function

Toru Nakamura[1], Kensuke Baba[2], and Hiroto Yasuura[2]
[1]Graduate School of Information Science and Electrical Engineering, Kyushu University
[2]Faculty of Information Science and Electrical Engineering, Kyushu University
Kasugakoen 6-1, Kasuga-City, Fukuoka 816-8580, Japan

Abstract Electronic authentication with a portable device such as a smart card has been receiving increasing attention. For portable devices with a poor computation resource, one-way hash functions are used as a cryptography technology for secure authentication. However the computing time of an identification with hashed data is proportional to the number of the users, and therefore it is not practical for large-scale systems. This paper introduces a formalization of identification and two kinds of attacks, and analyzes the computing time of an identification of a user.

1 はじめに

ICカードやRFIDタグの普及に伴い，電子的な情報を用いた人やモノを認証が社会基盤として普及している．この認証のための情報を不正に用いることにより，他人になりすましたり，利用者の行動履歴を追跡したりできることが重大な社会問題となっている．

これらの脅威は，適切な暗号化技術を用いることで防ぐことができるが，コスト制限の厳しい小型デバイスを用いる場合，公開鍵暗号などに比べ処理の軽い一方向ハッシュ関数を用いた方が現実的である [1]．しかし，一方向ハッシュ関数を用いた場合，個体の識別に必要な時間が識別する対象の候補の数 n に比例してしまい，大規模なシステムに適用することは困難であった．

この問題に対し，Wongら [5] はグラフを用いた $O(\log n)$ 時間の認証プロトコルを提案した．また，野原ら [4] は木構造を用いた $O(\log n)$ 時間プロトコルを提案した．我々は，これらの手法の本質的なアイデアは同じであると考え，これを文字列の照合問題の解法として一般的に表そうと考えた．また，現実には，事前に何らかの計算を行うことができる場合がある．これによって，計算時間を減らす可能性について考察を行い，ほぼ定数時間の識別プロトコルが，計算資源に応じて安全性を犠牲にすることで得られることがわかった．

本稿では，まず，識別およびそれに対する2つの攻撃について定式化を行う．次に，計算時間のモデル化を行い，既存のプロトコルについて評価を行う．最後に，事前に計算を行うプロトコルを提案し，計算資源と安全性の関係について考察を行う．

2 定式化

2.1 識別

識別の定式化を行う．誰かを識別しようとする人を**サーバ**と呼び，s で表す．また，識別されようとする人を**クライアント**と呼び，c で表す．クライアントは**ユーザ** $u_1, u_2, \ldots$ のうち誰かであるとし，ユーザ u_i がクライアント c であるとき $c \equiv u_i$ で表す．s による c の**識別**とは，あるプロトコルが与えられて，以下の条件の下で i を出力することである．

- s は，$c \equiv u_i$ である i を知らない．
- s は，できるだけ正しく $c \equiv u_i$ である i を出力したい．
- c は，$c \equiv u_i$ である i について，s に i と出力させたい．

今，サーバはひとりしかいない場合を考え，ユーザのうち誰もが本人と確認できるものとする．本研究の目的はICカードなどを用いた電子的な識別の計算時間を評価するこ

とであるから，本稿では，識別が文字列のやり取りによって行われるものとする．よって，識別を定義するための条件として以下を加える．

- s および c は，互いに文字列を提出することができる．
- s は，c が文字列 w_i を送るとき i を出力する．また，これは s と u_i の間の共有知識である．

s が，識別のための文字列についての共有知識を持っているユーザの集合を $U = \{u_i \mid 1 \leq i \leq n\}$ とする．また，各ユーザについての文字列の集合を $W = \{w_i \mid 1 \leq i \leq n\}$ とし，$w_i \in W$ を u_i の **ID** と呼ぶ．各 u_i は，別々の w_i を提出することができるものとする．

次に，識別のためのプロトコルを考える．プロトコルはサーバおよび全てのユーザ間の共有知識であるとする．

定義 1 あるプロトコルによって，$c \equiv u_i$ であるとき，かつそのときに限って，s が i を出力するとき，そのプロトコルは実行的であるという．

以下は識別の単純なプロトコルである．

単純プロトコル

1. c は，s に文字列 x を提出する．
2. s は，W 中に $w_i = x$ となる w_i を探し，存在すれば i を，そうでなければ 0 を出力する．

条件から，u_i は c として，s に i を出力させるために，x として ID である w_i を提出する．各ユーザが別々の ID を提出できるなら，以下は明らかである．

補題 1 単純プロトコルは実効的である．

2.2 ハッシュ化

識別に対する 2 種類の攻撃を定義する．s と c に加え，攻撃者 a を考える．a による u_i へのなりすましとは，a が，s による $c \equiv a$ の識別において i を出力させようとする攻撃である．また，a によるリンクとは，a が，2 つの異なる識別におけるクライアント c と c' について，$c \equiv c'$ かどうかを知ろうとする攻撃である．ただし，$c \equiv c'$ は，ある i について $c \equiv u_i$ かつ $c' \equiv u_i$ であることを表している．ここで，a についての条件を加える．

- a は，$c \equiv u_i$ である i を知らない．
- a は，任意の識別において，s と c の間で提出される任意の文字列を知ることができる．
- a は，できるだけ正しくなりすまし，もしくはリンクを行いたい．

また，a 以外について以下を加える．

- s および a でない c は，できるだけなりすまし，およびリンクを防ぎたい．

単純プロトコルについては，a がなりすましとリンクを確実に成功させる戦略が存在する．これは，a が，手順 1 において c から s へ提出される x と，x が出力する i との対応を十分に取得することで可能であることが容易にわかる．a が，x による出力 i を取得できないとしても，攻撃対象となる i を限定しなければ，なりすましおよびリンクとも成功する．

これらの攻撃は，乱数と暗号化によって識別のたびに x を変化させることで防ぐことができる．復号化できる暗号については，実際にはそれに十分な計算資源が必要である．IC カードなどの計算資源が限られた環境については，一方向のハッシュ関数が用いられる．以下，一方向ハッシュ関数を用いた識別のプロトコルを一般化する．

h を，文字列の対から文字列への関数とし，サーバおよび全てユーザ，攻撃者が共有するものとする．また，h は単射であり一方向（つまり，X から $h(x) = X$ となる x を計算することが現実的に困難である）とする．このような h をハッシュ関数と呼び，文字列の対から文字列を得る計算をハッシュ化と呼ぶ．以下はハッシュ関数を用いたプロトコルである．

ハッシュプロトコル

1. s は c に文字列 r を提出する．
2. c は $h(x, r)$ を計算する．
3. c は s に X として $h(x, r)$ を提出する．
4. s は，W 中に $h(w_i, r) = X$ となる w_i を探し，あれば i を，なければ 0 を出力する．

h は単射であるから，以下は補題 1 より明らかである．

補題 2 ハッシュプロトコルは実行的である．

a が，単純プロトコルの場合と同様の方法で攻撃する場合を考える．s は，a による攻撃を防ぐために，手順 1 で r として常に新しい文字列を提出すればよい．(s が r を決定する場合，なりすましを成功させることができる．)

3 計算時間

識別のための処理時間として，文字列の照合，文字列の通信，ハッシュ化に必要な時間を考える．文字列の通信については，提出される文字列の長さに，ハッシュ化については，ハッシュ化の回数に比例した時間が必要であるとする．文字列の照合については，以下に定義する問題の計算複雑さとして考える．

定義 2 *ID*照合問題とは，長さ m の文字列 n 個からなる集合 W と x が与えられ，$w_i = x$ となる $w_i \in W$ が存在すれば i を返し，そうでなければ 0 を返すものである．

単純なアルゴリズムによって，この問題は mn 回の文字の比較によって解くことができる．理想的な乱数とハッシュ関数が使われていれば，W に対して事前に照合のためのデータ構造を用意することはできないと考えるのが適当である．よって，この場合必要な計算時間は $O(mn)$ である．W についてのデータ構造を前もって用意できるならば，効率的なアルゴリズムが存在する [2]．例えば，葉が各 $1 \leq i \leq n$ に対応する木構造を準備すれば，ID 照合問題は $\Theta(m)$ で解決できる．この時間複雑さは文字列のアルファベットに依存するが，本稿では，以降文字列はビット列である（つまり，アルファベットサイズは 2 である）とする．よって，m は $\log n$ に依存するものとする．

以下，前章の 2 つのプロトコルについて計算時間の解析を行う．τ_m，τ_s，τ_h を，それぞれ，定数個の文字の対の比較，定数長の文字の送信，一回のハッシュ化についての係数とする．上の議論から，照合のための計算時間は，データ構造が前もって用意できる場合 $\tau_m \log n$ とする．また，用意できない場合，期待値を考えて，$(\tau_m n \log n)/2$ とする．

単純プロトコルについては，識別に必要な時間は，手順 1 における x の提出と，手順 2 における ID 照合の時間であるから，C_n を定数として，

$$\tau_s \log n + \tau_m \log n + C_n$$

である．

ハッシュプロトコルについては，識別に必要な処理時間は，手順 1 における r の提出，手順 2 におけるハッシュ化，手順 3 における X の提出，および手順 4 における ID 照合の時間である．ここで，r の長さは定数であり，$h(x,r)$ の長さは $\log n$ に比例すると仮定する．また，ハッシュ化の回数の期待値は $n/2$ であるから，C_h を定数として，

$$\tau_s \log n + \frac{1}{2}\tau_m n \log n + \frac{1}{2}\tau_h(n+2) + C_h$$

である．

4 高速化手法

4.1 ハッシュ計算回数の削減

ハッシュプロトコルは，なりすましとリンクを防ぐことができるという意味で安全だが，必要なハッシュ化の回数が n に比例するため，大きな n に対しては現実的な時間で実行できない．また，現実的には $\tau_h \gg \tau_m$ である [3] が，将来的に膨大な n を扱うことになった場合，$n \log n$ に比例する項も無視できなくなる．この問題を解決するために，ハッシュ化の回数を $O(\log n)$ に削減する手法がいくつか提案されている [5, 4]．これらの手法の識別の部分に注目すると，本質的なアイデアは，単純プロトコルのような操作を繰り返して ID 照合の候補を段階的に減らすことで，最悪の場合 n 回必要なハッシュ化を平均して $\log n$ 回にすることである．

上のアイデアを以下に一般化する．$V = \{v_1, v_2, \ldots, v_k\}$ を文字列の集合，$U_1, U_2, \ldots, U_k$ を $\bigcup_{i=1}^{k} U_i = U$ であるユーザの集合とする．今，U 中の任意のユーザは V 中のひとつの文字列を持つものとし，v_i を持つユーザは U_i に含まれるものとする．v_i を**部分 ID** と呼ぶ．部分 ID と対応するユーザの部分集合を，各 U_i について再帰的に定義することで，以下のプロトコルが得られる．

段階的プロトコル

1. s は c に r を提出する．
2. c は $h(x,r)$ を計算する．
3. c は s に X として $h(x,r)$ を提出する．
4. s は，V 中に $h(v_i,r) = X$ となる v_i を探し，あれば $U = U_i$ とし，なければ 0 を出力する．
5. s は $U = \{u_j\}$ ならば j を出力し，そうでなければ手順 1 に戻る．

補題 2 から，1 段階の操作により，任意のユーザ u_i について，$u_i \in U_i$ である部分集合 $U_i \subset U$ を決定することができる．また，ユーザが別々の部分 ID を提出するとき，1 段階の操作により出力の候補の数は明らかに減る．よって，各ユーザが別々の部分 ID の列を提出するならば，段階的プロトコルは実行的である．

前章のモデルに従うと，1 段階の操作において必要な計算時間は，C_p を定数として，$\tau_s \log k + (\tau_m k \log k)/2 + (\tau_h k)/2 + C_p$ である．k は各段階について異なる値をとり得るが，1 段階の操作により出力の候補が減少するという仮定から，十分大きな k' について $\log_{k'} n$ 回繰り返すことで上のプロトコルによる計算は停止する．例えば，二分木を用いた場合，この計算時間は

$$(\tau_s + \tau_m + \tau_h)\log n + C_p$$

である．

4.2 事前のハッシュ計算

ハッシュ計算をあらかじめ行うことによって，計算時間を見かけ上削減する手法を考える．以下のプロトコルはハッシュプロトコルの手順を変化させることで得られる．

事前計算プロトコル

1. s は，ある r と全ての $w_i \in W$ について，$h(w_i, r)$ を計算する．
2. s は c に文字列 r を提出する．
3. c は $h(x, r)$ を計算する．
4. c は s に X として $h(x, r)$ を提出する．
5. s は $h(w_i, r) = X$ となる $h(w_i, r)$ を探し，あれば i を，なければ 0 を出力する．

このプロトコルはハッシュプロトコルの順番を変えたものであるから，補題 2 より以下が明らかである．

定理 1 事前計算プロトコルは実行的である．

上のプロトコルの計算時間を，手順 1 とそれ以外について分けて考える．手順 1 については，n 回のハッシュ化が必要である．よって，前述のモデルに従うと，C_1 を定数として，

$$\tau_h n + C_1$$

である．それ以外について必要な計算時間は，手順 2 における r の提出，手順 3 におけるハッシュ化，手順 4 における X の提出，および手順 5 における ID 照合の時間である．ここで，手順 1 の後に，$h(w_i, r)$ の検索についてのデータ構造が作成されたとすると，手順 5 に必要な計算時間は $\Theta(\log n)$ であるから，C_2 を定数として，

$$\tau_s \log n + \tau_m \log n + \tau_h + C_2$$

である．

上の計算時間に加えて，データ構造を作成するための時間を考慮すると，事前計算プロトコル全体に必要な計算時間は，ハッシュプロトコルよりも長くなることがわかる．しかし，手順 1 とその後のデータ構造の作成が識別の前に行われていたと仮定すると，現実的には $\tau_h \gg \tau_m$ かつ $\tau_h \gg \tau_s$ である [3] ため，ほぼ定数時間の識別が可能である．

5 考察

事前計算プロトコルは，ある識別と次に行われる識別の時間が十分にあるならば適用できる．そうでない場合，安全性を落とすことで適用できる．

まず，安全性の簡単な定量化を行う．理想的なハッシュ関数を考えたとき，a は h の構造や乱数の出現順序などから X の解析を行うことはできないので，なりすましやリンクを成功させるためには，複数の識別において同じ r が用いられる可能性に依存する．r の長さを ℓ とすると，ある x についての $X = h(x, r)$ のとり得る値の数は 2^ℓ である．つまり，ある種の安全性を 2^ℓ に比例する値として定量化することができる．

識別の実行頻度によって十分な事前計算の時間が確保できない場合，2^ℓ の値を小さくする，つまり，安全性を犠牲にすることで，事前計算を実行可能な量に減らすことができる．あるいは，必ずしも新しい r を使わないことでも，事前計算に必要な時間を減らすことができる．

安全性と計算資源との詳しいトレードオフ関係は，安全性の具体的な定量化を行った上で与えられるが，大まかには，計算速度が速いほどある時間に計算できるハッシュ化の数が多くなり安全性が増す．また，メモリ領域が大きいほど保持しておけるハッシュ化された ID の数が多くなり安全性が増す．

6 おわりに

識別およびそれに対する 2 つの攻撃について定式化を行った．また，計算時間のモデル化を行い，既存のプロトコルについて評価を行った．さらに，事前に計算を行うプロトコルを提案し，計算資源と安全性の関係について考察を行った．

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．

参考文献

[1] Avoine, G. and Oechslin, P., "A Scalable and Provably Secure Hash-Based RFID Protocol", Proc. 2nd International Workshop on Pervasive Computing and Communications Security (PerSec2005), pp.110–114, 2005.

[2] Crochemore, M. and Rytter, W., "Text Algorithms", Oxford University Press, New York, 1994.

[3] 中村徹, 野原康伸, 馬場謙介, 井上創造, and 安浦寛人, "リンク不能性を持つ ID 照合システムの実装に向けて", 2006 年暗号と情報セキュリティシンポジウム予稿集 (概要集, pp.72), 2006.

[4] Nohara, Y., Inoue, S., Baba, K., and Yasuura, H., "Quantitative Evaluation of Unlinkable ID Matching Scheme", Proc. the 2005 ACM Workshop on Privacy in the Electronic Society (WPES'05), pp.55–60, 2005.

[5] Wong, C.K., Gouda, M., and Lam, S.S., "Secure Group Communications Using Key Graphs", IEEE/ACM Trans. Networking, vol.8, no.1, pp.16–30, 2000.