数理解析研究所講究録 1426 巻 2005 年 91-95

部分IDの一意性を考慮したID集合生成に関する問題

牧山 幸史 (Koji Makiyama)†, 納富貞嘉 (Sadayoshi Noutomi)‡, 安浦寛人 (Hiroto Yasuura) ‡‡
† 九州大学大学院 システム情報科学府 情報工学専攻
‡ 九州大学 システム LSI 研究センター
‡‡ 九州大学大学院 システム情報科学研究院 情報工学部門
†† Graduate School of Information Science and Electrical Engineering, Kyushu University
‡ System LSI Research Center Kyushu University

概 要

本論文では，任意の部分文字列がすべて異なるような文字列の集合を生成する二つのアルゴリズムを提示し，それに付随する問題について議論する．De Bruijn sequence を用いたアルゴリズムでは，その高速化および安全性向上が問題となる．また，単純なアルゴリズムでは，その停止性が問題となる．

1 はじめに

近年，急速に進む社会の電子化情報化に対して，電子サービスの安全性の確保や責任の所在の明確さについて見直された新しい社会基盤システムとして，PID システム (Personal ID system) が提案されている [1]. PID システムは，サービス受領者，サービス提供者，そして ID 発行者という 3 者から成り立ち[1]，サービス受領者とサービス提供者間で行われるサービスの授受は，すべて ID の認証を通してから行われる．PID システムでは，ID 発行者が個々のサービス受領者に一つの長い ID を割り当て，サービス提供者に対しては認証のためにその部分 ID を渡すという方法で，安全性を高めた取り引きを実現している．

この PID システムにおいて，ID 認証時に，サービス提供者が個々のサービス受領者を識別したい場合，ID 発行者から渡される部分 ID は，すべてのサービス受領者に対して異なっている必要がある．すなわち，ID 発行者は，それぞれのサービス受領者の ID から，すべて異なった部分 ID を抜き出さなければならない．しかし，ID 集合が無作為に作られていた場合，このような性質を満たす部分 ID 集合が抜き出せるかどうかは，まったく保証されていないことになる．

これを保証するために，我々は生成すべき ID 集合に次の制約を加えることにする．

- 各ユーザの ID から任意の部分 ID を取り出したとき，その部分 ID は他のサービス受領者のすべての部分 ID と異なる．

これにより，ID 発行者は，ランダムに部分 ID を抜き出すだけで，すべて異なった部分 ID の集合を得ることができる．

本稿では，上記性質を満たすように ID 集合を生成するアルゴリズムを二つ提案し，それぞれのアルゴリズムに付随して生じるいくつかの問題を示す．

本稿の構成は以下の通りである．第 2 章では，任意の部分 ID 集合が必ず一意性を持つために，ID 集合が満たすべき性質を定式化する．第 3 章では，この性質を満たすような ID 集合を生成する一つのアルゴリズムを提案し，このアルゴリズムの実用性について議論する．第 4 章では，もう一つのアルゴリズムを提示し，そこで問題となる停止性問題を定式化する．第 5 章で，まとめと今後の課題を述べる．

2 定式化

本章では，任意の部分 ID 集合が必ず一意性を持つために，ID 集合が満たすべき性質を定式化する．

まず，ID を文字列とみなし，部分 ID をその部分文字列として定義する．簡単のため，アルファベットは $\{0,1\}$ とし，ID 長は n ビット固定とする．

定義 2.1 n ビットの *ID* 全体の集合を $W^{(n)} = \{0,1\}^n$ で表す．

これにより，ID 集合は $W \subseteq W^{(n)}$ として表すことができる．この ID 集合 W が持つべき性質は，W

[1] 一つの PID システムにおいて，ID 発行者は一つであり，サービス提供者は複数存在してよい．すなわち，サービス受領者は，一つの ID で複数のサービスを受けることができる．

に属するすべてのIDから，m ビットの部分IDを任意にいくつか取り出したとき，それらの部分IDがすべて異なっていることである．これは，次のように表現できる．

定義 2.2 (m-uniqueness) 長さ n の *ID* 集合 $W \subseteq W^{(n)}$ が *m-uniqueness* であるとは，

$$\forall w_1, w_2 \in W, \forall k_1, k_2 [w_1 \neq w_2 \text{ ならば } \mathbf{sub}_{k_1}^{(m)}(w_1) \neq \mathbf{sub}_{k_2}^{(m)}(w_2)] \quad (1)$$

が成り立つときをいう．ただし，$1 \leq k_1, k_2 \leq n - m + 1$ であり，$\mathbf{sub}_k^{(m)}(w)$ は文字列 w の先頭から k 番目の文字から始まる長さ m の部分文字列を抜き出したものを表す．すなわち，$w = a_1 a_2 \ldots a_n$ とすると，$\mathbf{sub}_k^{(m)}(w) = a_k a_{k+1} \ldots a_{k+m-1}$ である．

以上により，我々の目的は，m-uniquenessを持つID集合 $W \subseteq W^{(n)}$ を生成することであることが明らかにされた．

3 De Bruijn Sequence を用いたアルゴリズム

本章では，m-uniqueness であるようなID集合 $W \subseteq W^{(m)}$ を生成するアルゴリズムを提示し，その実用性について考察する．このアルゴリズムは，まず de Bruijn sequence と呼ばれる，ある性質を満たす特殊な文字列を作り，その部分文字列をIDとして取り出すというものである．

3.1 アルゴリズム

定義 3.1 (m-de Bruijn sequence) 文字列 s が，ある正整数 m に対して次の性質を満たすとき，s を *m-de Bruijn sequence* と呼ぶ：

(i) $\forall w \in W^{(m)}, \exists k [\mathbf{sub}_k^{(m)}(s) = w]$,

(ii) $|s| = 2^m + m - 1$.

ただし，$1 \leq k \leq |s| - m + 1$ であり，$\mathbf{sub}_k^{(m)}(s)$ は文字列 s の先頭から k 番目の文字から始まる長さ m の部分文字列を抜き出したものを表す．

すなわち，m-de Bruijn sequence とは，長さ m のすべての文字列を部分文字列として持つような文字列の中で，最短の文字列のことである．

命題 3.2 *m-de Bruijn sequence* s は次の性質を満たす：

$$\forall k_1, k_2 [k_1 \neq k_2 \text{ ならば } \mathbf{sub}_{k_1}^{(m)}(s) \neq \mathbf{sub}_{k_2}^{(m)}(s)]. \quad (2)$$

ただし，$1 \leq k_1, k_2 \leq |s| - m + 1$ である．

したがって，m-de Bruijn sequence を一つ作成し，その中からお互いが重ならないように長さ n の部分文字列を抜き出すことによって，式 (1) を満たすようなID集合 W を得ることができる．

アルゴリズム 3.3 正整数 m, n, u が与えられたとき，*m-uniqueness* を満たすように，長さ n の *ID* を u 個生成するアルゴリズム．

1. *m-de Bruijn sequence* を作成する．
2. 作成された *m-de Bruijn sequence* から，長さ n の部分文字列を重複のないように取り出す．
3. 部分文字列の数が u 個になったら終了．そうでなければ *2* へ．

m-de Bruijn sequence の長さは定義より $2^m + m - 1$ であるので，このアルゴリズムが停止するためには，$u \leq (2^m + m - 1)/n$ を満たしていなければならないことに注意が必要である．

3.2 アルゴリズムの実用性

本節では，アルゴリズム 3.3 を用いてID集合を実際に生成し，本アルゴリズムの実用性を議論する．

ここでは，de Bruijn sequence を作成するアルゴリズムとして，Annexstein[2] のアルゴリズムを用いる．また，重複の無いように部分文字列を抜き出すアルゴリズムとしては，次のものを用いる．

アルゴリズム 3.4 正整数 n, u および $(m-1)$-循環文字列 s が与えられたとき[2]，s から長さ n の部分文字列を u 個抜き出すアルゴリズム．

1. 1 から $|s| - m + 1$ までの中から整数を一つランダムに選び k_0 に代入する．$k \leftarrow 1, j \leftarrow 0$.

[2] m-循環文字列とは，次を満たすものをいう：

$$\mathbf{sub}_1^{(m)}(s) = \mathbf{sub}_{|s|-m+1}^{(m)}.$$

ただし，m は正整数である．すなわち，m-循環文字列とは，最初の m 文字と最後の m 文字が同じ文字列のことである．

明らかに，m-de Bruijn sequence は $(m-1)$-循環文字列である．

m	所要時間
16	0.125 [seconds]
⋮	⋮
30	4 [minutes]
31	8 [minutes]
32	16 [minutes]
⋮	⋮
64	130,000 [**years**] (probably)

表 1: 実装結果 (UltraSPARC-III,750MHz)

2. $\mathbf{sub}_{k_0}^{(|s|-k_0-m+2)}(s)$ と $\mathbf{sub}_1^{(k_0+m-2)}(s)$ を連結した文字列を s' とおく.

3. $\mathbf{sub}_k^{(n)}(s')$ を部分文字列として抜き出す. $j \leftarrow j+1$.

4. $j = u$ ならば終了. 5へ.

5. $k+n$ から $|s|-(u-j)n$ までの中から整数を一つランダムに選び k に代入する. 3へ戻る.

アルゴリズム 3.3 と同様に, このアルゴリズムが停止するためには, $u \leq |s|/n$ を満たさなければならないことに注意が必要である.

PID システムにおいて, 部分 ID は認証用の鍵として用いられるため, 選出する際に画一的に選び出されてしまうと安全性は減少する. このため, アルゴリズム 3.4 は, 部分文字列をランダムに選び出すように設計してある.

これらのアルゴリズムを用いてアルゴリズム 3.3 を C 言語で実装した. m に対する ID 集合を生成するまでの所要時間を表 1 に示す.

表 1 より, $m = 32$ 付近までならば, 実用的な時間で ID 集合を生成できることが分かる. すなわち, アルゴリズム 3.3 が使用できるのは, 部分 ID の長さが 32 程度のときである. これは, 1000 ビットの ID を 400 万人のサービス受領者に割り当てることができるサイズである.

ただし, 部分 ID が認証鍵として使われることを考えたとき, 安全面からの実用性も考慮しなければならない. これは, ランダムに ID を作成したとき, それが本物の ID と一致する確率を見れば分かる.

もし, 1000 ビットの ID を 400 万人に配布したとすると, 本アルゴリズムを使用した場合, この確率は $(4\times10^6)/2^{32} = 9.3\times10^{-4}$ である. すなわち, 1000 回に一回は一致してしまう. これと比べて, 何の制約もない場合の確率を計算すると, $(4\times10^6)/2^{1000} = 3.7\times10^{-293}$ となり, 実に, 2.7×10^{294} 回に一回という割合でしか一致しない.

一般的に議論するために, ID の長さを n, 部分 ID の長さを m, サービス受領者数を u, 同時に受けられるサービスの数の最大値を p とおくと, ランダムに ID を作成したときそれが本物の ID と一致する確率は, 提案アルゴリズムを用いた場合は $u/2^m$, 制約の無い場合は $u/2^n$ となり, これらの比は, $n \simeq 2mp$ であるため,

$$\frac{u}{2^m}/\frac{u}{2^n} = \frac{2^n}{2^m} \simeq \frac{2^{2mp}}{2^m} = 2^{(2p-1)m}$$

となる. このことから, この確率比は, p または m の増加に従って, 指数関数的に大きくなっていくことが分かる.

以上により, 提案アルゴリズムが有効となるアプリケーションとして, 以下の場合を考えることができる.

1. 同時に受けられるサービスの数が少ない場合.
2. 部分 ID の長さが短い場合 (この場合, $un \leq 2^m+m-1$ より, サービス受領者数と ID の長さに制約を受ける)
3. サービス受領者数が少ない場合 ($u/2^m$ が下がるため).
4. 安全性に対する要求が低い場合.
5. 認証に, 部分 ID 以外の鍵システムを使う場合.
6. ID 生成アルゴリズムを公開しない場合.

本アルゴリズムは, $m = 32$ 程度までは実用的な時間で ID 集合を生成できるが, より大きな m に対しては実用的な時間で ID 集合を生成することはできない. したがって, より大きな m に対して ID 集合を生成したい場合, より高速な生成アルゴリズムが必要である. また, 安全性を考慮すると, 我々は本アルゴリズムの適用に制限を加えなければならない. これはランダムに de Bruijn sequence を生成するアルゴリズムを考案することによって解決できると考えられる. 今回 de Bruijn sequence を生成するために使用した Annexstein のアルゴリズムは, ある特定の de Bruijn sequence を生成するアルゴリズムであり, 安全性が低下するのはそのため

である．m-de Bruijn sequence の種類は $2^{2^{m-1}-m}$ 個であるため [3]，これらの中からランダムに選び出すアルゴリズムがあれば，安全性の低下を抑えることができると考えられる．

4 単純なアルゴリズムの停止性について

ID の長さを n，部分 ID の長さを m，サービス受領者数を u とすると，m-uniquness を持つ ID 集合を生成する単純なアルゴリズムとして次のものが考えられる．

アルゴリズム 4.1 正整数 n，m，u が与えられたとき，*m-uniqueness* を持つ長さ n の文字列 u 個からなる集合を生成するアルゴリズム．

1. ランダムに長さ n の文字列を作り，文字列集合 W とする *(初期化)*.
2. ランダムに長さ n の文字列を作り，W に付け加える．
3. W が *m-uniquness* を満たすかどうかを判定し，満たすならば *4* へ．満たさないならば，最後に付け加えられた文字列を破棄し，*2* へ戻る．
4. $|W| = u$ ならば終了．そうでないならば *2* へ戻る．

本章では，このアルゴリズムの停止性を保証するために解くべき問題を考え，定式化する．

4.1 アルゴリズムの停止性

第 2 章で示した，ID 集合が満たすべき性質を整理すると次のようになる：

- ID 集合の任意の ID は，長さ n の文字列である．
- ID 集合から任意に ID を取りだしたとき，その ID に含まれる長さ m の任意の部分文字列は，ID 集合の他のどの ID の長さ m の部分文字列とも一致しない．

このような性質を持つ文字列の集合は，m-de Bruijn Graph (正式な定義は次節) の中の，長さ $n-m+1$ の独立なパス[3] の集合として表すことができる．

m-de Bruijn Graph は，2^m 個の頂点を持ち，それぞれのパス (一つのパスが一つの ID に対応する) は $n-m+1$ 個の異なる頂点を持つ．したがって，上記の性質を満たすような ID 集合が持つことのできる ID の数の最大値は，$\lfloor 2^m/(n-m+1) \rfloor$ であることが分かる[4]．このことから，前節で示した ID 集合生成アルゴリズムは，u の値が $\lfloor 2^m/(n-m+1) \rfloor$ よりも大きいとき，絶対に停止しないことが分かる．

ところで，$\lfloor 2^m/(n-m+1) \rfloor$ という値は，ID 集合の持つことのできる ID の数の最大値であるが，これがアルゴリズムの停止条件を正確に表しているというわけではない．すなわち，u の値が $\lfloor 2^m/(n-m+1) \rfloor$ よりも大きいとき，アルゴリズムが絶対に停止しないことは確かであるが，u の値が $\lfloor 2^m/(n-m+1) \rfloor$ よりも小さいからといって，必ず停止するわけではない．

したがって，次の問題が ID 集合生成アルゴリズムの停止性問題として考えられる：

- $u \leq x$ ならばアルゴリズムが必ず停止すると保証されるような x のうち，最大のものを求めよ．

次節では，この問題を定式化する．

4.2 問題の定式化

定義 4.2 (m-de Bruijn Graph) $V_m = \{0,1\}^m$，$E_m = \{e \in V_m \times V_m \mid e = (xs, sy),\ x, y \in \{0,1\},\ s \in \{0,1\}^{m-1}\}$ とおき，$G_m = (V_m, E_m)$ を，*m-de Bruijn Graph* と呼ぶ．

定義 4.3 (長さ l のパス) $p = (v_1, v_2, \ldots, v_l) \in V_m^l$ が次の性質を満たすとき，長さ l のパスという：

$$\forall i, j (1 \leq i, j \leq l)[i \neq j \implies v_i \neq v_j],$$
$$\forall i (1 \leq i \leq l-1)[(v_i, v_{i+1}) \in E_m].$$

m-de Bruijn Graph における，長さ l のパス全体の集合を $P_{m,l}$ と書く．

定義 4.4 (独立) 二つの長さ l のパス $p_1 = (u_1, \ldots, u_l)$，$p_2 = (v_1, \ldots, v_l) \in P_{m,l}$ が独立であるとは，

$$\forall i, j (1 \leq i, j \leq l)[u_i \neq v_j]$$

[3] 他のパスと共有点を持たないパス
[4] $\lfloor x \rfloor$ は x を超えない最大の整数を表す．

が成り立つときをいう.

長さ l パスの集合 $P \subseteq P_{m,l}$ が独立であるとは,

$$\forall p_1, p_2 \in P[p_1 \neq p_2 \implies p_1 \text{と} p_2 \text{は独立}]$$

が成り立つときをいう.

m-de Bruijn Graph における, 独立な長さ l のパスの集合全体の集合を $\mathcal{P}_{m,l}$ と表す.

定義 4.5 (極大) 独立な長さ l のパスの集合 $P \in \mathcal{P}_{m,l}$ が極大であるとは,

$$\forall p \in P_{m,l} \setminus P[P \cup \{p\} \notin \mathcal{P}_{m,l}]$$

が成り立つときをいう. また,

$$\mathcal{Q}_{m,l} = \{P \in \mathcal{P}_{m,l} \mid P \text{は極大}\}$$

とおく.

問題 4.6 (停止性問題) 二つの自然数 m, l が与えられたとき,

$$\min_{P \in \mathcal{Q}_{m,l}} |P|$$

を求めよ. ただし, $1 \leq l \leq 2^m$.

5 おわりに

本稿では, 任意の部分 ID がすべて異なるような ID 集合を生成するアルゴリズムを二つ提示した. また, それらに付随して生じる問題を示した. 今後の課題は, これらの問題を解決することである.

参考文献

[1] Hiroto Yasuura, "Towords the Digitally Named World —Challenges for New Social Infrastructures based on Information Technologies", Proceedings of Euromicro Symposium on Digital System Design -Architectures, Methods and Tools-(DSD2003), pp.17-22, Sep. 2003.

[2] F. S. Annexstein, "Generating De Bruijn Sequences: An Efficient Implementation", IEEE Transaction on Computers, Vol. 46, No. 2, Feb. 1997.

[3] J. H. van Lint, R. M. Wilson, "A Course in Combinatorics", Cambridge University Press, 2001.