MPI並列処理によるレインボーテーブル生成の高速化

安藤 公希† 桑原 寛明‡ 上原哲太郎‡ 國枝 義敏‡

†立命館大学大学院情報理工学研究科 ‡立命館大学情報理工学部
525-8577 滋賀県草津市野路東1丁目1-1
is0098ef@ed.ritsumei.ac.jp

あらまし 本稿では，PCクラスタとMPIを用いた並列処理によるレインボーテーブル生成の高速化手法を提案する．レインボーテーブルはパスワードクラックにおける辞書攻撃に使用する辞書を圧縮した特殊なテーブルであり，平文とハッシュ値のペアを複数接続して先頭の平文と末尾のハッシュ値を取り出すことで生成される．ペアを接続するためにはハッシュ値から平文を還元する必要があるが，異なるハッシュ値から同じ平文が還元される衝突の可能性があるため，衝突判定処理が必要である．この衝突判定処理に多くの計算時間を要する．衝突判定処理をMPI並列化によって負荷分散させることで高速化を図る．本稿では英小文字2，3，4文字のパスワードすべてを含むレインボーテーブルを実際に生成し，高速化の効果を示す．

Acceleration of Generating Rainbow Tables by MPI Parallelized Processing

Koki Ando† Hiroaki Kuwabara‡ Tetsutaro Uehara‡ Yoshitoshi Kunieda‡

†Graduate School of Information Science and Engineering, Ritsumeikan University
‡Department of Computer Science, Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu, Shiga 525-8577, JAPAN
is0098ef@ed.ritsumei.ac.jp

Abstract In this paper, we propose a method to parallelize rainbow table generation using MPI. Rainbow table is a special compressed dictionary to be used for password cracking. This table consists of connected pairs of password and its hash value. When connecting pairs, we have to detect collisions of reduced password because different hash values may be reduced to the same password. We aim to achieve load balancing and speed up this heavy collision detection process by parallelizing with MPI. We apply our proposed method to generate rainbow tables containing all passwords composed of 2, 3 or 4 lowercase letters for showing the effect of speeding up.

1 はじめに

情報システムにおける個人認証には，多くの場合ID(identifier:各個人で異なる識別子)とパスワードの組が用いられる．本人だけがこのパスワードを知っていることが前提であり，パスワードによってIDを使用する人物が本人であることを証明する．このパスワードを用いた認証技術を以下ではパスワードシステムと呼ぶ．パスワードシステムにおいては，パスワードクラックによって不正に取得した他人のパスワー

ドを用いて他人になりすまし，個人認証を成功させ個人に関する情報が盗まれるという危険性がある．

パスワードクラックに使用されるパスワード解析手法には，辞書攻撃，総当たり攻撃，レインボークラックなどがある．ここでいう攻撃とは，パスワードシステムに対し，ID とパスワードの組み合わせを変えながら認証を何度も試すことをいう．辞書攻撃はパスワードとして使用されるであろう単語をあらかじめ辞書に登録し，その辞書を用いて攻撃する手法である．総当たり攻撃は，パスワードとして使用されるであろうあらゆる単語の組合せを都度生成しながら攻撃する手法である．レインボークラック [1] は，辞書攻撃に使用される辞書を圧縮したレインボーテーブルを用いて攻撃する手法である．総当たり攻撃では膨大な解析処理時間を，辞書攻撃では使用する辞書の格納に膨大な記憶容量を必要とするため現実的でなく，パスワードシステムの脅威とはなりにくい．しかし，レインボークラックではレインボーテーブルにより辞書攻撃で使用される辞書の格納に必要な記憶容量を削減できる．このレインボーテーブルの存在により，レインボークラックは記憶容量的にも計算量的にも現実的なパスワード解析を可能にする．

本研究では，レインボークラックがパスワードシステムの脅威となりうること，および危険性がどの程度現実的となっているかを示すため，高速なレインボーテーブルの生成を目標とする．具体的には，並列処理によりレインボーテーブル生成を高速化する．レインボーテーブルは，平文とハッシュ値のペアを接続することで生成されるが，ペアとペアの接続のためにハッシュ値から平文を還元する必要がある．しかし，異なるハッシュ値から同一の平文が還元される衝突が高い確率で発生し，衝突判定の処理には多くの計算が必要である．したがって，逐次処理では膨大な生成時間が必要となるため，並列処理によって負荷を分散し，レインボーテーブル生成を高速化する．本研究では，並列化のために MPI を使用する．レインボーテーブルの生成例として，ネット掲示板などで利用されているパスワードシステムのパスワードを対象とするレインボーテーブルを生成する．このシステムに対し英小文字 2，3，4 文字のそれぞれで構成されるパスワードに対するレインボーテーブルを生成する．

本稿の構成は以下の通りである．2 章でレインボーテーブルについて説明し，3 章で MPI 並列環境，加えて MPI 並列化による提案手法と実装について述べる．4 章で提案手法の評価，および考察を述べ，5 章で関連研究について述べる．6 章で本稿のまとめとする．

2 レインボーテーブルによる解析

2.1 想定する攻撃状況

パスワードシステムは，パスワードをハッシュ関数によりハッシュ値に変換して保存している．個人がパスワードシステムを利用するとき，本人であることを証明するためにパスワードを入力する．入力されたパスワードのハッシュ値を保存してあるハッシュ値と比較して個人認証をする．

本研究では，攻撃者がすでに ID と解析対象のハッシュ値を取得している状況を想定する．攻撃者の目的は，取得したハッシュ値からもとのパスワードを導き出すこと，そして，ID とパスワードを不正に使用して個人になりすまし，システムに侵入して個人に関する秘密情報を盗み取ることである．

2.2 レインボーテーブル

レインボーテーブルは辞書攻撃に使用する辞書を圧縮した特殊なテーブルであり，チェインの集合である．チェインとは平文とハッシュ値のペアを複数接続したものであり，チェインを生成することをチェイン化という．

平文とハッシュ値のペアの生成にはハッシュ関数を使用し，ペアとペアをつなぐチェイン化には還元関数を使用する．あるペアに対し，そのハッシュ値に還元関数を適用して還元される平文を持つペアを接続する．

図 1 にレインボーテーブル基本形（以降，基本形と表記）を示す．平文とハッシュ値のペア

をつないでチェインを生成し，複数のチェインの集合体として一つの基本形を構成する．この時，基本形に含まれるすべての平文が異なっていることが望ましい．

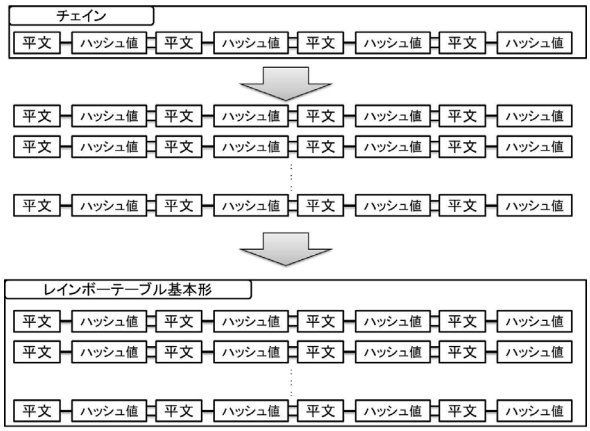

図 1: チェインとレインボーテーブル基本形

基本形には辞書中のすべての平文とハッシュ値が含まれており，辞書が大きくなると必要な記憶容量が膨大になるため，圧縮して容量を小さくする．基本形の圧縮には，チェインを圧縮したレインボーセットを用いる．レインボーテーブルは，基本形を構成する一つ一つのチェインが圧縮されたレインボーセットの集合である．

以下では，圧縮されたレインボーテーブル基本形を単にレインボーテーブルと呼ぶ．図 2 に，レインボーテーブル基本形と，基本形を圧縮したレインボーテーブルを示す．

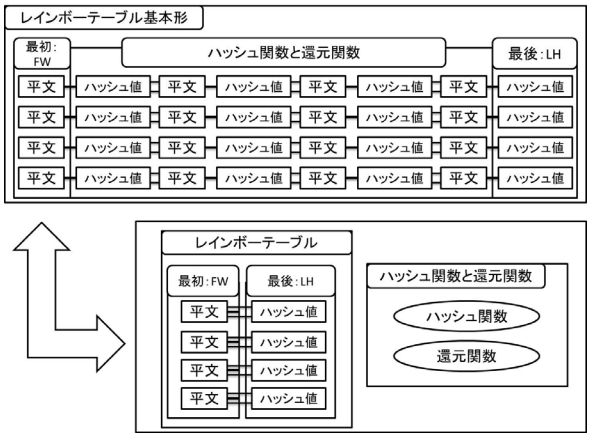

図 2: 基本形とレインボーテーブル

2.2.1 レインボーセット

チェインを圧縮する仕組みがレインボーセット（RainbowSet；以降 RS と表記）である．レインボーセットは，一つのチェインにおける最初の平文（FirstWord；以降 FW と表記）と最後のハッシュ値（LastHash；以降 LH と表記）の二つの要素で構成される．図 3 にチェインとレインボーセットの構成の違いを示す．図 3 にあるように，一つのチェインから FW と LH を取りだし，一つの RS を構成する．

RS は，チェイン生成に使用したハッシュ関数と還元関数を使用することで，もとのチェインと同様に扱うことができる．RS とチェインの違いは，FW と LH の間にある平文とハッシュ値を，必要となるたびに計算するか，一度生成したら値を保持するかである．

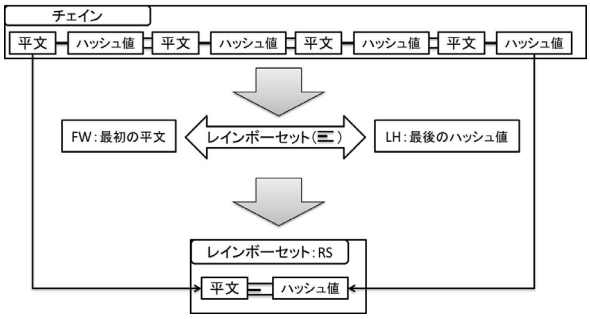

図 3: チェインとレインボーセット

2.2.2 ハッシュ関数

平文からハッシュ値を生成する関数を暗号学的ハッシュ関数（以降，ハッシュ関数と表記）と呼ぶ．ハッシュ関数は任意長の平文の入力に対し固定長ビット列のハッシュ値を出力する関数である．ハッシュ関数により平文がハッシュ化されハッシュ値となる．

2.2.3 還元関数

還元関数は，ハッシュ値から平文を還元する関数である．ハッシュ値から平文を還元するとき，異なるハッシュ値から同じ平文が還元される衝突が発生する可能性がある．

チェイン生成時にチェイン化済みの平文が還元されたときに衝突が発生したと判断する．平文が衝突した場合，その衝突した平文から生成されるチェインは常に同一であるため，異なるチェインの後半が重複することになり，生成時間と記憶容量の無駄となる．このため，衝突した場合はチェイン化されていない平文から新たにチェインを生成する．

2.3 レインボークラック

レインボーテーブルを用いて，あるハッシュ値に対応する平文を求めることをレインボークラックと呼ぶ．手順を次に示す．

1. 比較するハッシュ値を用意する．2. で一致しなかった場合，還元関数とハッシュ関数を順に適用して新しいハッシュ値を生成する．
2. ハッシュ値をレインボーテーブルの LH と比較する．ハッシュ値と LH が一致なかった場合は 1. へ一致した場合は 3. へ
3. LH に対応する FW からパスワードを発見するまで RS からチェインを復元しながら探索する．
4. 解析対象のハッシュ値とペアになっている平文がパスワードであり，解析成功となる．

レインボークラックのイメージを図 4 に示す．ここでは解析対象のハッシュ値を hoge’ とし，取得したいパスワードを hoge とする．① で新たなハッシュ値を順に生成し，② でハッシュ値と LH（pass’，ekus’，lamz’，true’，…，limi’）を比較している．ハッシュ値と LH 一致がしなかった場合は① の処理に移る．一致した場合（① の limi’ と② の limi’ が一致）は③ の処理に移る．③ では LH（② における limi’）に対応した FW（②，③ における fate）からパスワードを発見するまで探索し，④ でレインボークラックの成功となる．

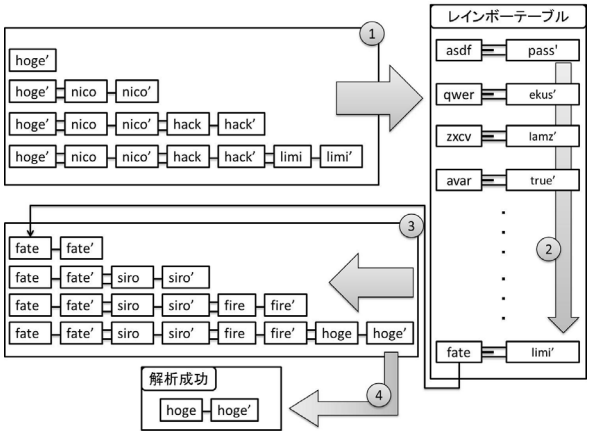

図 4: レインボークラック

2.4 レインボーテーブル生成手法

2.4.1 チェインの生成

以下のようにしてレインボーテーブルを構成する各チェインを生成する．

1. 平文とハッシュ値のペアからチェイン化を開始する．
2. チェイン化済みの平文と同一の平文が還元された場合，チェイン化中のチェインの生成を終了する．
3. まだチェイン化されていない平文から新たなチェインの生成を開始する．

長さや使用する文字の種類といった条件を満たすすべての平文とハッシュ値のペアが生成されるまでチェイン生成を続け，最終的に生成されたすべてのチェインをまとめて一つのレインボーテーブルとする．

生成し終えたチェインについて，チェインの FW と LH を RS として記憶する．RS を記憶した後，RS 生成に利用したチェインを破棄し，次の RS となるチェインを生成する．

2.4.2 LH 比較による衝突判定

還元された平文の衝突判定を効率的に行うために，LH 比較による衝突判定を行う．具体的には，チェイン化するペアのハッシュ値を生成

済みの RS の LH と比較することで衝突判定する．チェイン化するペアのハッシュ値と LH が衝突した場合，チェイン化中のチェインの FW から衝突したハッシュ値までの間で他に衝突がないか判定するために，LH に対応した FW から再び衝突判定を行う．これにより，生成済みのレインボーテーブル全体を探索しなくても衝突判定を実現できる．

図 5 にハッシュ値と LH の衝突時の処理を示す．初めに，ハッシュ値（図 5 の e’）と RS の LH（図 5 の e’）で衝突判定を行う．衝突しなかった場合は，チェイン生成を続ける．衝突した場合，LH に対応した FW（図 5 の a）とチェイン化中のチェインの FW（図 5 の q）から再び衝突判定を行う．還元された平文（図 5 の d）で衝突が起きているため，衝突前のハッシュ値（図 5 の r’）でチェイン化を終了しチェインを生成する．生成し終えたチェインの FW（図 5 の q）と LH（図 5 の r’）を RS として記憶する．

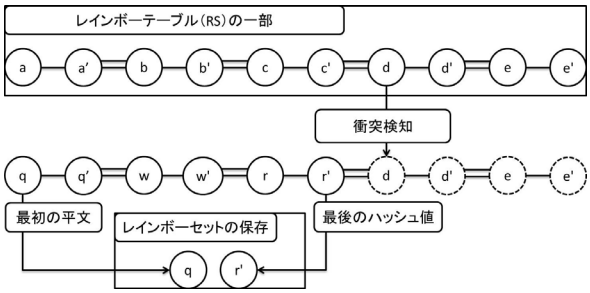

図 5: ハッシュ値と LH の衝突時の処理

LH 比較による生成手法の特徴として，生成処理におけるメモリ使用量を抑えられることが挙げられる．これはチェインではなく，RS を保存しているからである．生成するレインボーテーブルに格納すべき平文の数が膨大であるとき，で使用可能なメモリの容量を超えてしまうため，メモリ使用量を抑える必要がある．

3 MPI 並列化

3.1 MPI

MPI（Message Passing Interface）は，並列コンピューティングのための規格である．MPI は，C，C++，Fortran などの言語で使用できるライブラリとして実装されている [2]，[3]，[4]．MPI は，SIMD(Single Program, Multiple Data streams) モデルとして，並列処理を行える．このモデルでは，各プロセスは同じプログラムプを走らせて処理するが，各プロセスではプログラム内の分岐によって異なるステートメントを実行する．この分岐はプロセスのランク（rank）によって決める．

3.2 レインボーテーブル生成の MPI 並列化

3.2.1 衝突判定処理の並列化

本研究で提案するレインボーテーブル生成の並列化では，衝突判定処理を並列化する．チェイン生成において最も負荷がかかる処理が衝突判定処理である．したがって，衝突判定処理に集中する負荷を分散することがレインボーテーブル生成の並列化において重要である．

具体的には一つの親プロセスで生成した RS を複数の子プロセスに順番に割り当て，衝突判定処理を複数の子プロセスを用いて並列に実行する．

親プロセスでは，自身の衝突判定処理を含むチェインの生成と，子プロセスとのデータの送受信を行う．

複数の子プロセスでは，割り当てられた RS とハッシュ値との衝突判定処理，および親プロセスとのデータの送受信を行う．

3.2.2 親プロセスでの処理

親プロセスでは，チェインの生成を行う．まず，まだチェイン化されていない平文を見つけだし，その平文を FW とする．FW からハッシュ関数を使用してハッシュ値を生成し，各子プロセスに送信する．その後，子プロセスから衝突判定処理の結果を待つ．

もし衝突していた場合は，衝突前のハッシュ値が返される．衝突は一か所でしか発生しないため，返されるハッシュ値は一通りであり，そ

のハッシュ値を LH として RS を生成し，子プロセスに割り当てるために送信する．

もし衝突していなかった場合は，ハッシュ値から還元関数を用いて新たな平文を還元する．還元した平文がチェイン化済みでないかの衝突判定をするため，現在チェイン化中の FW から新たに還元した平文までを順番に衝突判定する．もし平文が衝突していた場合は，衝突前のハッシュ値を LH として RS を生成し，決められた子プロセスに RS を送信する．衝突していなかった場合は，ハッシュ関数によってハッシュ値を生成し各子プロセスに送信する．

この一連の処理をすべてのペアが生成されるまで続ける．すべてのペアが生成されたときがレインボーテーブル生成の完了である．

3.2.3 子プロセスでの処理

子プロセスでは，RS を受信するまで待ち状態となる．RS を受信した場合，その RS を記憶し，親プロセスから送信されるハッシュ値を受信する．受信したハッシュ値を記憶した RS の LH と衝突判定する．

もしハッシュ値と LH が衝突した場合は，LH に対応する FW から再度衝突判定していき，ほかに衝突部分がないか調べ，衝突前のハッシュ値を親プロセスに送信する．

もし衝突していなかった場合は，親プロセスに衝突していなかった旨を送信し，親プロセスからの応答を待つ．

この一連の処理をすべてのペアが生成されるまで繰り返す．

3.3 チェイン分割

衝突判定処理の並列化において，各子プロセスの負荷を効果的に分散させるために，生成したチェインを適当な長さに分割する．

チェイン分割は親プロセスで行う．生成した一つのチェインを，一定の長さで複数のチェインに分割する．分割した複数のチェインを各子プロセスへ順番に割り当てることで各子プロセスが持つチェインの長さの合計を均等化する．子プロセスが持つチェインを均等化することで，衝突判定処理が一つの子プロセスに集中せずに各子プロセスに分散され，チェイン生成における衝突を速く発見することができる．

本研究では，チェイン分割なしの並列化手法を LastHashUnlimited（以降，LHU と表記）と呼び，チェイン分割した並列化手法を LastHash-Bounded（以降，LHB と表記）と呼ぶ．表 1 に，それぞれの LH 生成手法の特徴を示す．図 6，図 7 に割り当てられる RS（チェイン）の長さをそれぞれ示す．

図 6 では，生成したチェインをそのまま子プロセスに割り当てているため，子プロセス:1 が記憶するチェインの長さが大きくなり，子プロセス:3 が記憶する長さが小さくなっている．このため，効果的な負荷分散が難しくなる．

図 7 では，生成したチェインを分割してから子プロセスに割り当てているため，各子プロセスが記憶するチェインの長さの均一化ができ，効果的な負荷分散が期待できる．

表 1: LHU と LHB の特徴

	LHU	LHB
生成 RS 数	LHB より少ない	制限値により変化
チェインの長さ	ランダム	制限値以下
生成時間	LHB より遅い	制限値により変化
生成時の並列性	LHB より劣る	制限値により変化
解析時の並列性	LHB より劣る	制限値により変化
負荷分散性	低い	高い

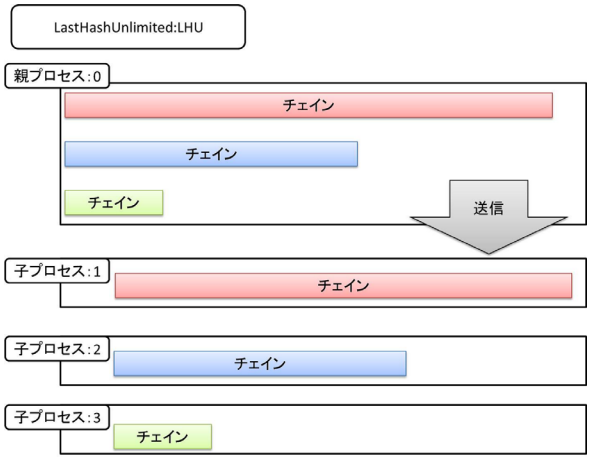

図 6: LHU におけるチェインの割り当て

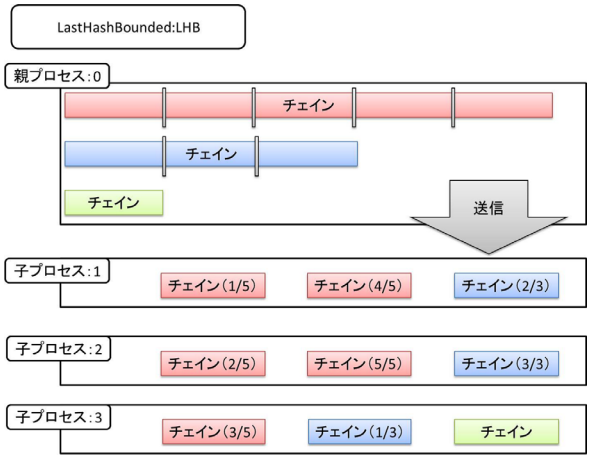

図 7: LHB におけるチェインの割り当て

4 評価と考察

逐次処理と並列処理のそれぞれで英小文字 2, 3, 4 文字のそれぞれで構成される平文のレインボーテーブルを生成し，生成にかかる処理時間を計測する．並列化において，LHU と LHB についても比較する．実行環境を表 2 に示す．

表 2: 評価環境

OS	CentOS 7.0
CPU	intel core i7 4790k 4.00GHz
MEM	32GB
使用言語	c/c++
コンパイラ	gcc/g++, mpic++
MPI	MPICH2
ノード数	8 ノード

4.1 ハッシュ関数

本研究では，解析対象のハッシュ値が crypt(3) によるハッシュ化を利用しているため，ハッシュ関数に Unix の crypt(3) を使用する．Unix の crypt(3) のアルゴリズムは，DES（Data Encryption Standard）を使用している．DES は，共通鍵暗号方式のアルゴリズムの一つであり，ブロック暗号の一種である [5]，[6].

4.2 レインボーテーブルの生成時間

4.2.1 逐次処理と並列処理による生成時間

逐次処理におよび並列処理（LHU）による計測結果を表 3 に示す．表 3 より，2 文字では並列処理よりも逐次処理の実行時間が速く，3，4 文字の場合は並列処理の実行時間が速いことがわかる．プロセス数を 8 から 32 に変えた場合，3 文字では 8 並列の方の実行時間が速いが，4 文字の場合は 32 並列の方の実行時間が速い．

結果から，MPI 並列処理によるレインボーテーブル生成の並列化の有用性が示せた．2 文字の並列生成処理と 3 文字の 32 並列生成処理が遅くなった理由に，単一マシンでの処理に負荷が集中していることと，通信処理のオーバヘッドがある．

表 3: 逐次処理と並列処理の実行時間

文字数	逐次	8 並列（LHU）	32 並列（LHU）
2 文字	0m0.302s	0m1.260s	0m2.509s
3 文字	0m49.476s	0m36.466s	1m10.608s
4 文字	1098m46.317s	184m31.947s	72m15.608s

4.2.2 チェイン分割並列処理による生成時間

並列化において，LHU と LHB の結果を表 4 に示す．ここで B:数字は，分割後のチェインの最大の長さである．また，生成処理の各段階に時間計測を追加したため，表 3 の結果とは異なる．

表 4 より，チェインを分割して各子プロセスに割り当てることで，レインボーテーブル生成にかかる処理時間を短縮でき，衝突判定処理の効果的な負荷分散ができる．しかし，3 文字 32 並列時の B:5 の場合をみると，B:10 や B:50 のときより遅い．チェインを細かく分割すると，衝突判定処理の負荷は効果的に分散されるが，通信処理の制御に時間がかかってしまい，生成時間が長くなった．

表 4: LHU と LHB の計測結果

文字数	並列数	LHU(s)	B:5(s)	B:10(s)	B:50(s)
2 文字	8	2.84	0.62		
	32	17.21	0.48		
3 文字	8	50.50	25.35	26.56	39.49
	32	169.81	62.23	25.54	34.52
4 文字	8	16420.70	4864.60	6038.10	8308.69
	32	16252.00	3394.15	5084.38	5219.66

5　関連研究

Sykes らは MPI，スレッド，GPGPU（CUDA）のそれぞれを用いてパスワードクラックの評価実験をおこなっている [7]．レインボークラックは，MPI を用いることがスレッド，GPGPU（CUDA）を用いる方法と比べ有利となっている．このことから，本研究においても MPI 並列処理を利用している．Gómez らは，MPI を用いたレインボークラックについて，並列の手法やレインボーテーブルの生成法を示している [8]．本研究では，8 並列と 32 並列における英小文字 2，3，4 文字それぞれのレインボーテーブル生成を試したが，Gómez らの研究では，16，32，64，90 並列における大文字，大文字と数字，大文字と数字と記号というように，文字の種類を変えて生成をしている．

6　おわりに

本研究では，MPI 並列処理によるレインボーテーブル生成の高速化手法を提案し，英小文字 2，3，4 文字のパスワードをすべて含むレインボーテーブルの生成に要する生成時間を調べた．

レインボーテーブルの生成において，RS の LH を用いて衝突判定を MPI を用いて並列化する手法を述べた．生成済みの RS を子プロセスに分割して持たせることで，衝突判定を並列に実行する．

レインボーテーブル生成の並列化の効果を示すため，英小文字 2，3，4 文字のそれぞれで構成される平文のレインボーテーブル生成処理の評価実験をおこなった．評価実験の結果から，文字数が多い方が並列処理による高速化の効果が大きくなったため，文字数を多くすることや文字の種類を増やすことよって生成する平文の数が多くなる場合のレインボーテーブル生成の高速化が期待できる．

今後の課題として MPI 並列処理における通信処理のオーバヘッドの削減がある．並列化するさい，各プロセス間でデータを送受信する必要があり，この送受信にかかる通信処理の時間が全体のレインボーテーブル生成の処理時間を長くしている．

謝辞

本研究の一部は JSPS 科研費 15K00112 の助成による．

参考文献

[1] 岩井博樹．標的型攻撃セキュリティガイド．ソフトバンククリエイティブ, 2013.

[2] P. パチェコ, 秋葉博 (訳)．MPI 並列プログラミング．培風館, 2001.

[3] ウイリアムグロップ，ラジーブタークル，ユーイングラスク, 畑崎隆雄 (訳)．実践 MPI - 2—メッセージパッシング・インタフェースの上級者向け機能．ピアソンエデュケーション, 2002.

[4] IBM Corporation. z/os v1r1.0-v1r12.0 unix system services parallel environment mpi programming and subroutine reference, 1990 ,2012. http://www-01.ibm.com/support/knowledgecenter/api/content/nl/ja-jp/SSLTBW_1.13.0/com.ibm.zos.r13.fomp200/ipezps00420.htm.

[5] 結城浩．新版暗号技術入門 秘密の国のアリス．ソフトバンククリエイティブ, 2008.

[6] 谷口功．よくわかる暗号化技術 (入門ビジュアルテクノロジー)．日本実業出版社, 2000.

[7] Edward R Sykes and Wesley Skoczen. An improved parallel implementation of rainbowcrack using mpi. *Journal of Computational Science*, Vol. 5, No. 3, pp. 536–541, 2014.

[8] Julio Gómez, Francisco G Montoya, R Benedicto, A Jimenez, Consolación Gil, and Alfredo Alcayde. Cryptanalysis of hash functions using advanced multiprocessing. In *Distributed computing and artificial intelligence*, pp. 221–228. Springer, 2010.