平成 20 年度

プロジェクト研究報告書

# パスワード認証に用いるハッシュ関数の比較

1090324　　越智 敦司

指導教員　　福本 昌弘

2009 年 3 月 5 日

高知工科大学 情報システム工学科

# 要 旨

## パスワード認証に用いるハッシュ関数の比較

越智 敦司

ユーザの認証を行う方法の一つとして，パスワードがある．一般にユーザ認証は，入力されたパスワードをハッシュ化し，あらかじめハッシュ化し保存していたハッシュ値と比較を行う．比較結果が同一であれば，正規のユーザであることになる．このハッシュ関数としてMD5 が広く使われていたが，MD5 には脆弱性が見つかっており他のハッシュ関数への移行が進んでいる [1]．本研究では，移行によりハッシュ関数の高速性，ランダム性について変化が見られるかを調べるため，現在使用されているハッシュ関数を，生成時間および特徴ある入力に対する出力の偏りに関して比較を行った．出力の偏については，本実験で付けた条件では，出力の偏りに大きな差は出ないという結果になった．

**キーワード**　ハッシュ関数，MD5，生成時間，出力の偏り

# 目次

# 図目次

図目次

# 表目次

# 第 1 章

# 序論

## 1.1 はじめに

認証を行う方法の一つとして，パスワードを用いるものがあり，よくハッシュ関数が使用されている．そのパスワードを用いた認証は，あらかじめパスワードをハッシュ関数に掛けハッシュ値を生成し，その値を保存しておく．そして認証する際には，パスワードを入力し同様にハッシュ関数に掛け，ハッシュ値を生成し，保存しておいたハッシュ値と比較を行う．その比較結果が同一であれば，正当であることになる．このパスワードを用いた認証に利用されるハッシュ関数として MD5 が広く使われていたが，MD5 には脆弱性が見つかっており他のハッシュ関数への移行が進んでいる [1]．また，ハッシュ関数には，計算が容易であることと出力値に偏りがないことが要求される．本研究では，移行によりこれらに変化が見られるかを調べるため，以前使用されていた MD4，MD5，現在利用されている SHA1，RIPEMD160 の四つのハッシュ関数について，生成時間および出力値の偏りに関して比較を行う．

## 1.2 本論文の構成

本論文の構成について述べる．

2 章では、パスワード認証に用いるハッシュ関数を分類や性質，代表的な構成ついて述べ，次に本研究で使用した，MD4，MD5，SHA1，RIPEMD160 について述べる．

3 章では、ハッシュ関数には計算が容易であることと，出力値の偏りがないことが要求さ

れるので，2 章で述べたハッシュ関数をそれらについて比較を行う．

4 章では，本研究の結論を述べ，今後の課題について述べる．

# 第 2 章

# ハッシュ関数の説明

## 2.1 はじめに

この章では，パスワード認証に用いるハッシュ関数を，性質と代表的な構成について述べる．次に、ハッシュ関数については本研究で使用した MD4，MD5，SHA1，RIPEMD160 について述べる．

## 2.2 ハッシュ関数

ハッシュ関数 $h$ は任意の長さのデータ $x$ を入力とし，固定長のハッシュ値 $h(x)$ を出力する関数である．そのハッシュ関数は以下のように定義される．

- ハッシュ値の計算は容易である
- ハッシュ値から入力値を求めるのは困難である (一方向性)
- 与えられた入力から得られる出力結果と同一する, 異なる入力を求めるのは困難である (衝突困難性)

### 2.2.1 分類

ハッシュ関数は鍵を使用するかによって MAC ハッシュ関数と MDC ハッシュ関数で分類される．MAC(Message Authentication Codes) ハッシュ関数は入力値 $m$ と秘密鍵 $k$ を利用してハッシュ値 $h(m, k)$ を生成する．一方 MDC(Modification Detection Codes)‘ ハッシュ関数は鍵を使用せず、入力値 $m$ だけを利用してハッシュ値 $h(m)$ を計算する．また鍵

を使用しないハッシュ関数はさらに暗号論的条件が課せられ，さらに二つに分類される．暗号論的条件は原像計算困難性と第二原像計算困難性と衝突困難性の三つであり，この三つのうち，原像計算困難性と第二原像計算困難性を満たすハッシュ関数を一方向性ハッシュ関数 (One-Way Hash Function) と呼び，さらに衝突発見困難性を満たすハッシュ関数を衝突困難ハッシュ関数 (Collision Free Hash Function) と呼ぶ．ハッシュ関数の分類を 2.1 に示す．

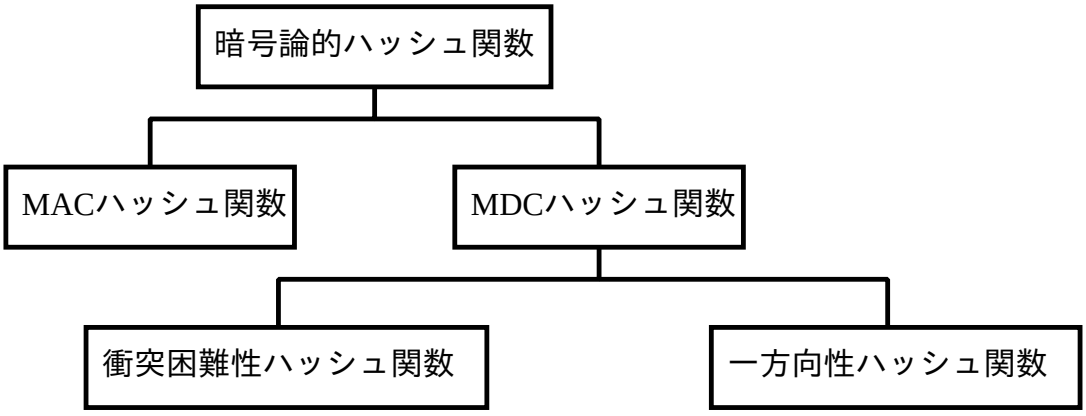



図 2.1　ハッシュ関数の分類

### 2.2.2　暗号論的条件

ハッシュ関数に課せられる，暗号論的条件は以下の通りである．

**原像計算困難性**　与えられたハッシュ値 $h(m)$ から，その入力 $m$ を計算することが困難である

**第二原像計算困難性**　与えられた入力 $x$ に対し，$h(x) = h(y)$ を満たす $x$ とは異なる $y$ を計算することが困難である

**衝突発見困難性**　$h(x) = h(y)$ を満たす $(x, y)$ を計算することが困難である

### 2.2.3　基本構造

ハッシュ関数の構成法は以下により分類される．

- 剰余算に基づく構成法
- ブロック暗号に基づく構成法
- 専用のアルゴリズムに基づく構成法

本研究では専用のアルゴリズムを用いた専用ハッシュ関数を使用する．専用ハッシュ関数はパディング，メッセージの分割，圧縮関数による演算，出力の順に実行される．

**1. パディング**　入力メッセージの長さが $m$ ビットの専用ハッシュ関数の場合入力メッセージ $M$ を $m$ ビットの倍数になるように $M$ の末尾に一つの ‘1’ と複数の ‘0’ が付加され，最終の 64 ビットに M のメッセージ長が付加される．このメッセージ長を付加する方法を MD-Dtrengthening と呼ぶ．

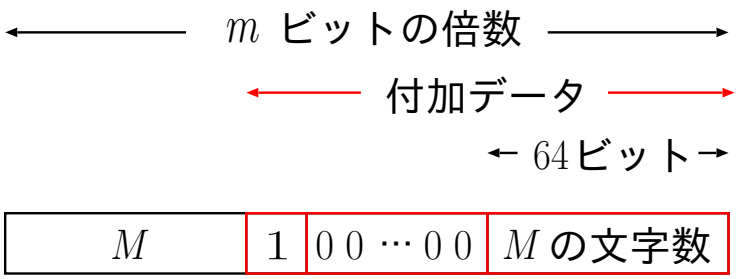



図 2.2　パディング

**2. メッセージの分割**　メッセージ $M$ を $m$ ビット毎の長さに分割する．そのときのメッセージ $M$ を $M = (M_{(1)}||M_{(2)}||\cdots||M_{(N)})$ とする．

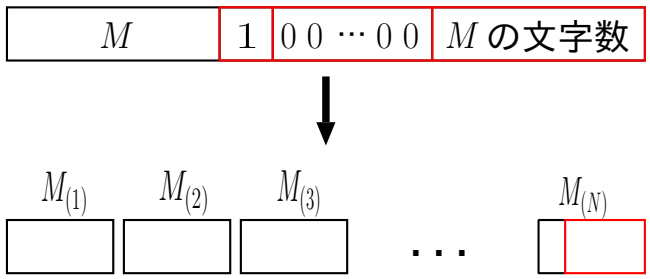



図 2.3　パディング

**3. 圧縮関数による演算**　$m$ ビットの固定長に分割されたメッセージ $M_{(i)}$ が順次，圧縮関数 $f$ へ入力される

**4. 出力**　出力関数 $g$ によりハッシュ値 $H$ が得られる

専用ハッシュ関数の基本構造を図 2.4 に示す.

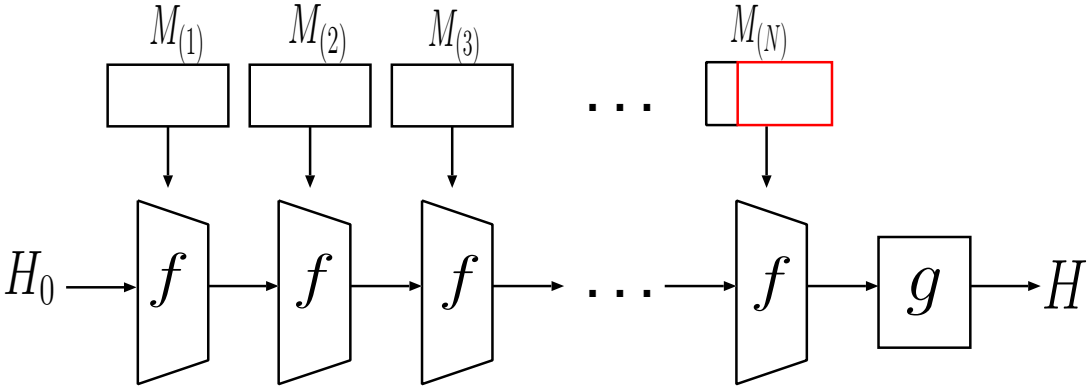



図 2.4　専用ハッシュ関数の基本構成

### 2.2.4　ハッシュ関数の種類

本実験で使用した専用ハッシュ関数 MD4，MD5，SHA1，RIPEMD160 について説明する.

**MD4**

MD4 は Ron Rivest によって設計されたハッシュ関数である．Rivest は圧縮関数を繰り返し適用した Message Digest(MD) という手法を提案した．その MD を用いたハッシュ関数には MD1，MD2，MD4，MD5 といったハッシュ関数がある．

MD4 は 128 ビットのハッシュ値を生成する専用ハッシュ関数である．入力メッセージは 512 ビット単位，演算は 32 ビット単位である．また，全体は 3 ラウンドで構成されている．1 ラウンドは 16 回の演算が行われるため，全体で 48 回演算が行われる．MD4 の構造を図 2.5 に示す.

MD4 では以下の圧縮関数 $f_0$，$f_1$，$\cdots f_{47}$ が用いられる．

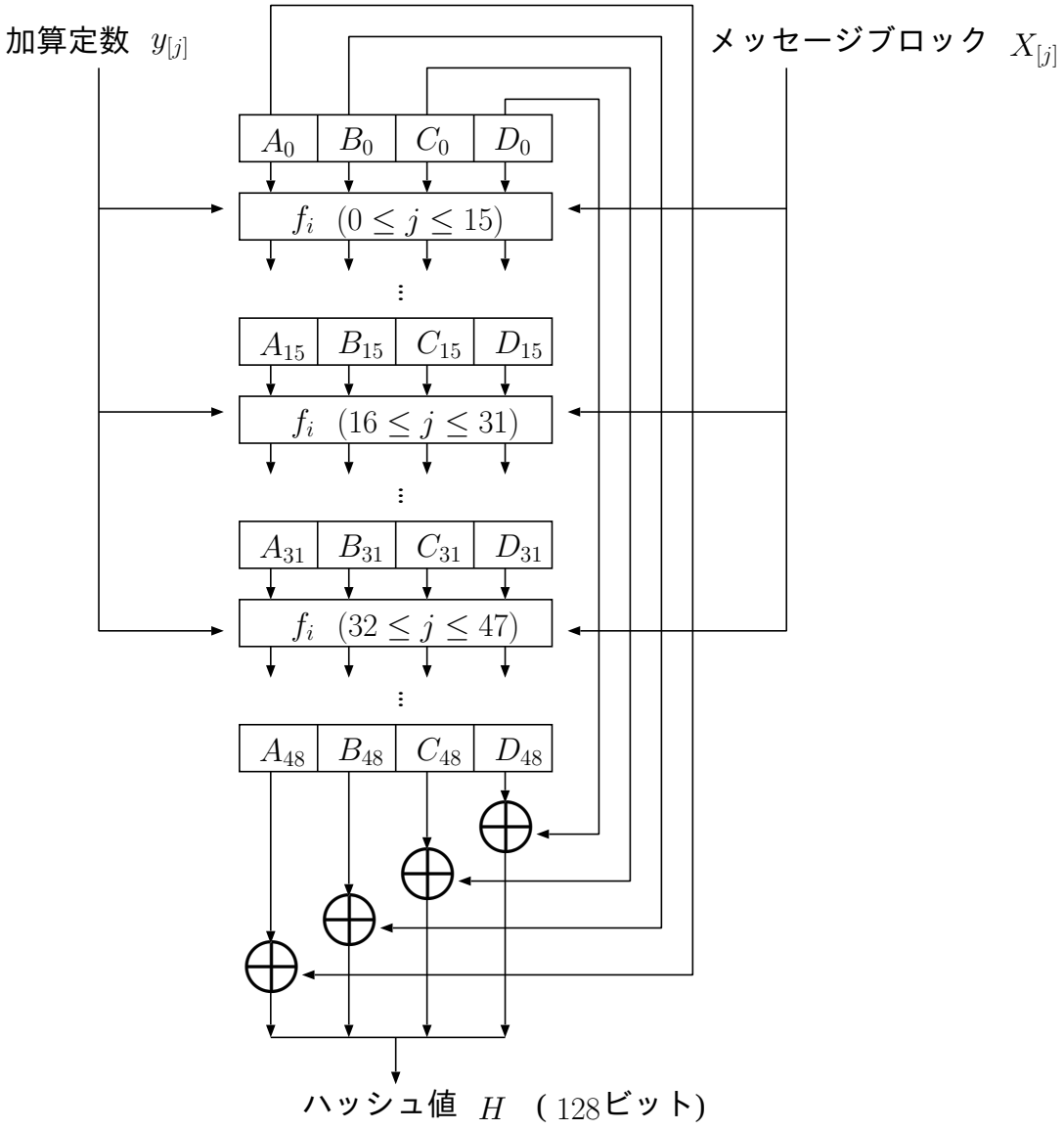
$$f_i(x,y,z) = \begin{cases} (x \wedge y) \vee (\neg x \wedge z) & (0 \leq j \leq 15) \\ (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) & (16 \leq j \leq 31) \\ x \oplus y \oplus z & (32 \leq j \leq 47) \end{cases} \tag{2.1}$$

初期ハッシュ値 $A_0$, $B_0$, $C_0$, $D_0$(各 32 ビット) は以下の通りである．

$$\begin{aligned} A_0 &= 67432301 \\ B_0 &= EFCDAB89 \\ C_0 &= 98BADCFE \\ D_0 &= 10325476 \end{aligned} \tag{2.2}$$

加算定数 $y_j$ は以下の通りである．

$$y_{[i]} = \begin{cases} 0x00000000 & (0 \leq j \leq 15) \\ 0x5A827999 & (16 \leq j \leq 31) \\ 0x6ED9EBA1 & (32 \leq j \leq 47) \end{cases} \tag{2.3}$$




$A_0$ , $B_0$ , $C_0$ , $D_0$ ： 初期ハッシュ値

$f_i$ $(0 \leq j \leq 48)$ ： 圧縮関数

図 2.5　MD4 の構造

### MD5

MD5 は MD4 を改善したものである．変更された内容は，ラウンド数が 1 ラウンド増加して 4 ラウンドになったこと，第 2 ラウンドで使用する圧縮関数の演算が変わったことなどがある [2]．また，他のハッシュ関数とは異なり，加算定数が演算毎に設定されている．出力は MD4 と同一の 128 ビットのハッシュ値を生成する．入力も同様に 512 ビット単位，演算も 32 ビット単位である．ラウンド数は 4 ラウンドであるため，全体で 64 回演算が行われる．MD5 の構造を図 2.6 に示す．

MD5 では以下の圧縮関数 $f_0$，$f_1$，$\cdots f_{63}$ が用いられる．

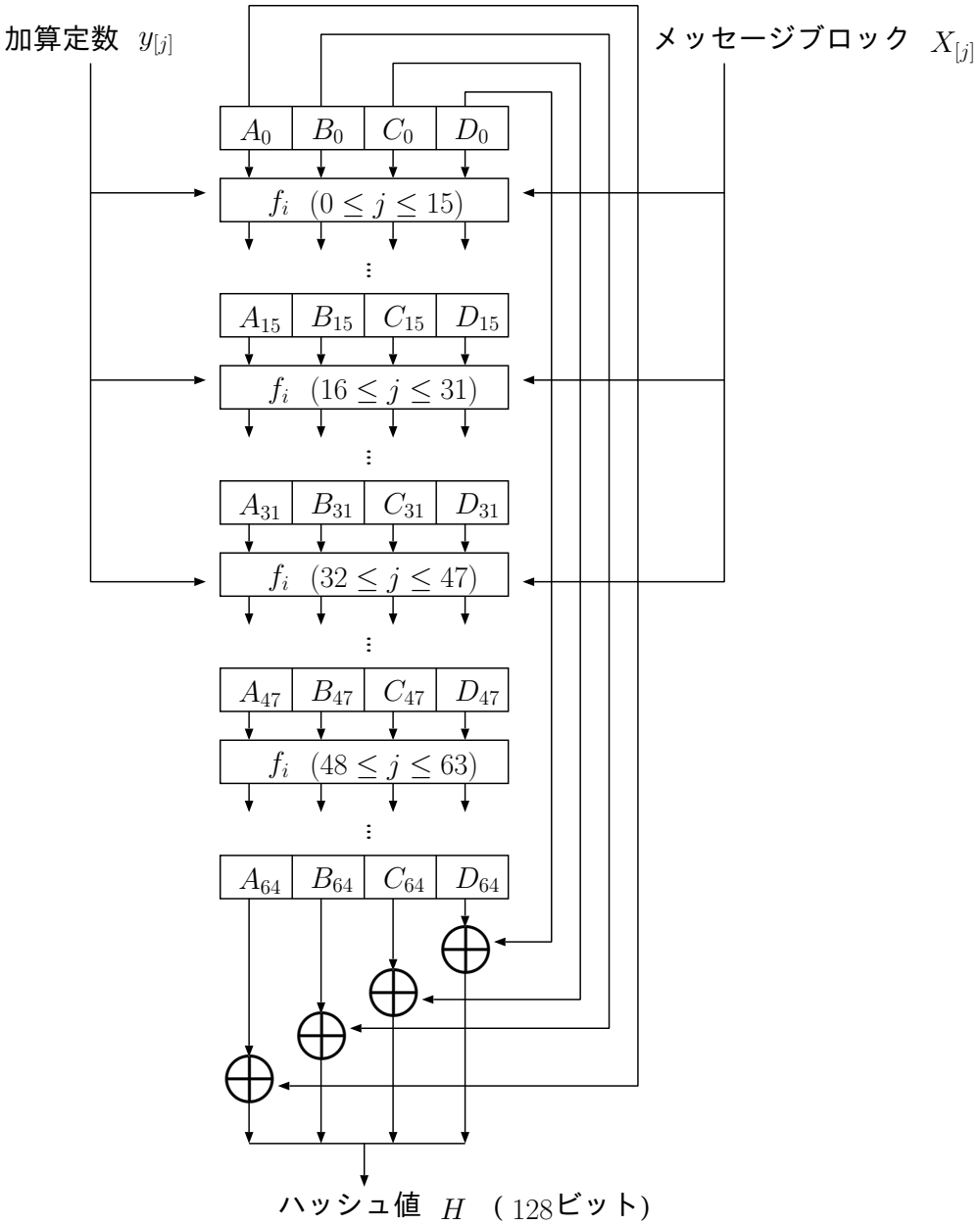
$$f_i(x,y,z) = \begin{cases} (x \wedge y) \vee (\neg x \wedge z) & (0 \leq t \leq 15) \\ (x \wedge z) \vee (y \wedge \neg z) & (16 \leq t \leq 31) \\ x \oplus y \oplus z & (32 \leq t \leq 47) \\ y \oplus (x \wedge \neg z) & (48 \leq t \leq 63) \end{cases} \tag{2.4}$$

初期ハッシュ値 $A_0$, $B_0$, $C_0$, $D_0$(各 32 ビット) は以下の通りである．

$$\begin{aligned} A_0 &= 01234567 \\ B_0 &= 89ABCDEF \\ C_0 &= FEDCBA98 \\ D_0 &= 76543210 \end{aligned} \tag{2.5}$$

加算定数 $y_j$ は以下の通りである．

0xD76AA478 0xE8C7B756 0x242070DB 0xC1BDCEEE 0xF57C0FAF 0x4787C62A 0xA8304613 0xFD469501 0x698098D8 0x8B44F7AF 0xFFFF5BB1 0x895CD7BE 0x6B901122 0xFD987193 0xA679438E 0x49B40821 0xF61E2562 0xC040B340 0x265E5A51 0xE9B6C7AA 0xD62F105D 0x24414530 0xD8A1E681 0xE7D3FBC8 0x21E1CDE6 0xC33707D6 0xF4D50D87 0x455A14ED 0xA9E3E905 0xFCEFA3F8 0x676F02D9 0x8D2A4C8A 0xFFFA3942 0x8771F681 0x6D9D6122 0xFDE5380C 0xA4BEEA44 0x4BDECFA9 0xF6BB4B60 0xBEBFBC70 0x289B7EC6 0xEAA127FA 0xD4EF3085 0x4881D050 0xD9D4D039 0xE6DB99E5 0x1FA27CF8 0xC4AC5665 0xF4292244 0x432AFF97 0xAB9423A7 0xFC93A039 0x655B59C3 0x8F0CCC92 0xFFEFF47D 0x85845DD1 0x6FA87E4F 0xFE2CE6E0 0xA3014314 0x4E0811A1 0xF7537E82 0xBD3AF235 0x2AD7D2BB 0xEB86D391




$A_0$ , $B_0$ , $C_0$ , $D_0$ ： 初期ハッシュ値

$f_i$ $(0 \leq j \leq 48)$ ： 圧縮関数

図 2.6　MD5 の構造

**SHA1**

SHA1 は米国商務省標準局 (National Institute of Standards and Technology) で開発されたハッシュ関数である．SHA には以前利用されていた，SHA0，SHA1 よりも出力が長い，SHA255，SHA512 などがある．SHA1 は MD4 を基板として設計されている．出力の長さは MD4 の 128 ビットより 32 ビット長い 160 ビットの出力値をもつ．入力は，MD4 と同一の 512 ビット単位で，演算の単位も 32 ビットである．ラウンド数は MD4 よりも多い 4 ラウンドとなっている．また，SHA1 は 1 ラウンド 20 回演算を行うため，全体で 80 回演算が行われる．SHA1 の構造を図 2.7 に示す．

SHA1 では以下の圧縮関数 $f_0$，$f_1$，$\cdots f_{79}$ が用いられる．

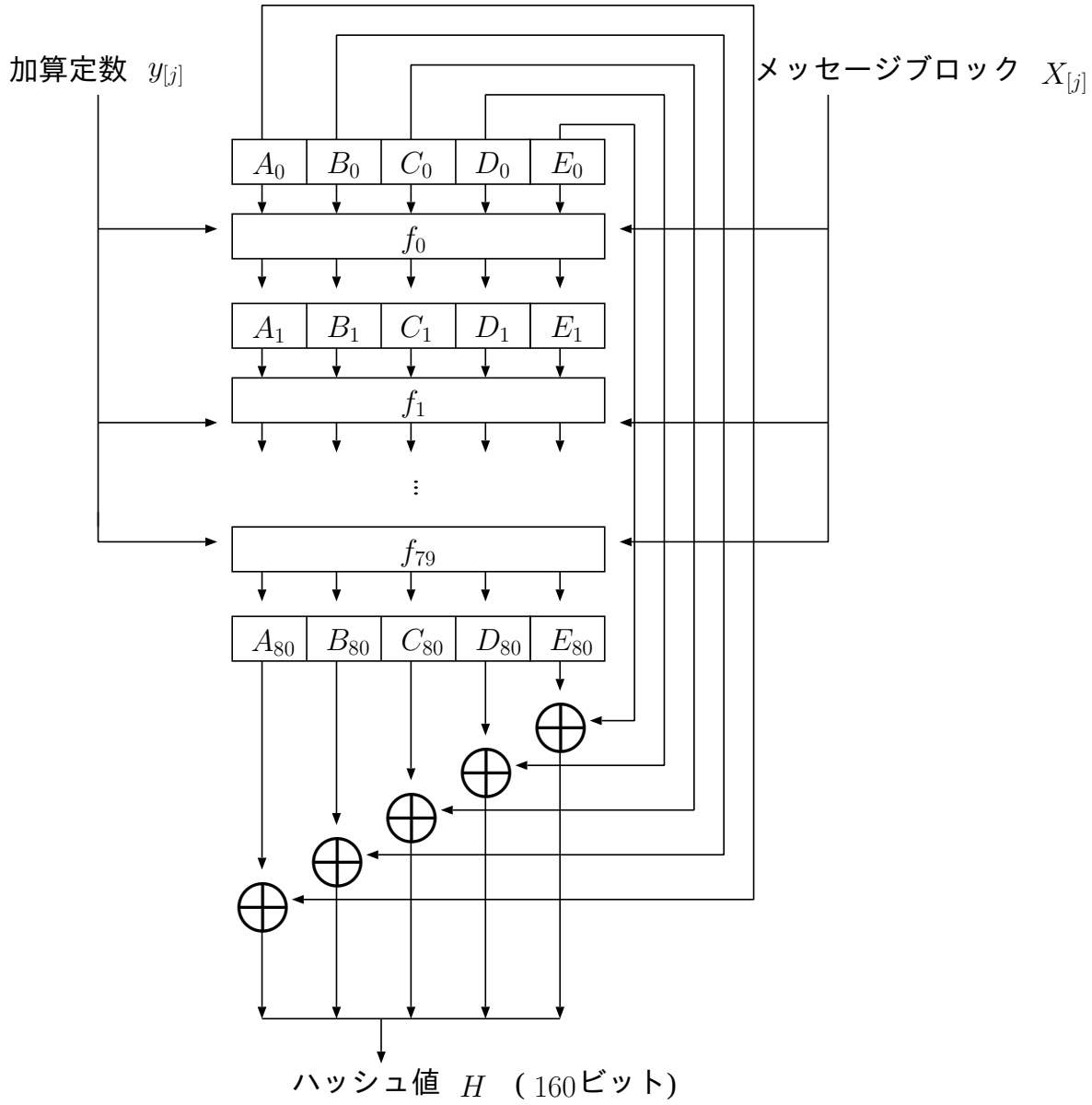
$$f_i(x,y,z) = \begin{cases} (x \wedge y) \vee (\neg x \wedge z) & (0 \leq t \leq 19) \\ x \oplus y \oplus z & (20 \leq t \leq 39) \\ (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) & (40 \leq t \leq 59) \\ x \oplus y \oplus z & (60 \leq t \leq 79) \end{cases} \tag{2.6}$$

初期ハッシュ値 $A_0$, $B_0$, $C_0$, $D_0$, $E_0$(各 32 ビット) は以下の通りである．

$$\begin{aligned} A_0 &= 67432301 \\ B_0 &= EFCDAB89 \\ C_0 &= 98BADCFE \\ D_0 &= 10325476 \\ E_0 &= C3D2E1F0 \end{aligned} \tag{2.7}$$

加算定数 $y_j$ は以下の通りである．

$$y_{[i]} = \begin{cases} 0x5A827999 & (0 \leq j \leq 19) \\ 0x6ED9EBA1 & (20 \leq j \leq 39) \\ 0x8F1BBCDC & (40 \leq j \leq 59) \\ 0xCA62C1D6 & (60 \leq j \leq 79) \end{cases} \tag{2.8}$$




$A_0$ , $B_0$ , $C_0$ , $D_0$ ： 初期ハッシュ値

$f_i$ $(0 \leq j \leq 48)$ ： 圧縮関数

図 2.7　SHA1 の構造

**RIPEMD160**

RIPEMD160 はヨーロッパ共同体の RIPE プロジェクトによって開発された RIPEMD128 を改良したハッシュ関数である．出力の長さは SHA1 と同一の 160 ビットである．入力も同一に 512 ビット単位で，演算の単位も 32 ビットである．また，RIPEMD は MD5 のような圧縮関数を使用するが，left-line と right-line の並列構造となっている．そのため，ラウンド数は left-line と right-line それぞれ 5 ラウンドで，RIPEMD は 1 ラウンド 16 回演算となっている．また，left-line と right-line は逆手順で処理を行う．ハッシュ値はそれぞれのラインの出力値と初期ハッシュ値を入力とした算術演算の処理結果として出力される．RIPEMD160 の構造を図 2.8 に示す.

RIPEMD160 では以下の圧縮関数 $f_1$，$f_2$，$f_3$，$f_4$，$f_5$ が用いられる．

$$f_i(x,y,z) = \begin{cases} x \oplus y \oplus z & (0 \leq t \leq 15) \\ (x \wedge y) \vee (\neg x \wedge z) & (16 \leq t \leq 31) \\ (x \vee \neg y) \oplus z & (32 \leq t \leq 47) \\ (x \wedge z) \vee (y \wedge \neg z) & (48 \leq t \leq 63) \\ x \oplus (y \vee \neg z) & (64 \leq t \leq 79) \end{cases} \tag{2.9}$$

初期ハッシュ値 $A_0$, $B_0$, $C_0$, $D_0$, $E_0$(各 32 ビット) は以下の通りである．

$$\begin{aligned} A_0 &= 67432301 \\ B_0 &= EFCDAB89 \\ C_0 &= 98BADCFE \\ D_0 &= 10325476 \\ E_0 &= C3D2E1F0 \end{aligned} \tag{2.10}$$

加算定数 $l_j$(Left-Line) と $l_j$(Right-Line) は以下の通りである．

$$l_{[i]} = \begin{cases} 0x00000000 & (0 \leq j \leq 15) \\ 0x5A827999 & (16 \leq j \leq 31) \\ 0x6ED9EBA1 & (32 \leq j \leq 47) \\ 0x8F1BBCDC & (48 \leq j \leq 63) \\ 0xA953FD4E & (64 \leq j \leq 79) \end{cases} \quad r_{[i]} = \begin{cases} 0x50A28BE6 & (0 \leq j \leq 15) \\ 0x5C4DD124 & (16 \leq j \leq 31) \\ 0x6D703EF3 & (32 \leq j \leq 47) \\ 0x7A6D76E9 & (48 \leq j \leq 63) \\ 0x00000000 & (64 \leq j \leq 79) \end{cases} \tag{2.11}$$

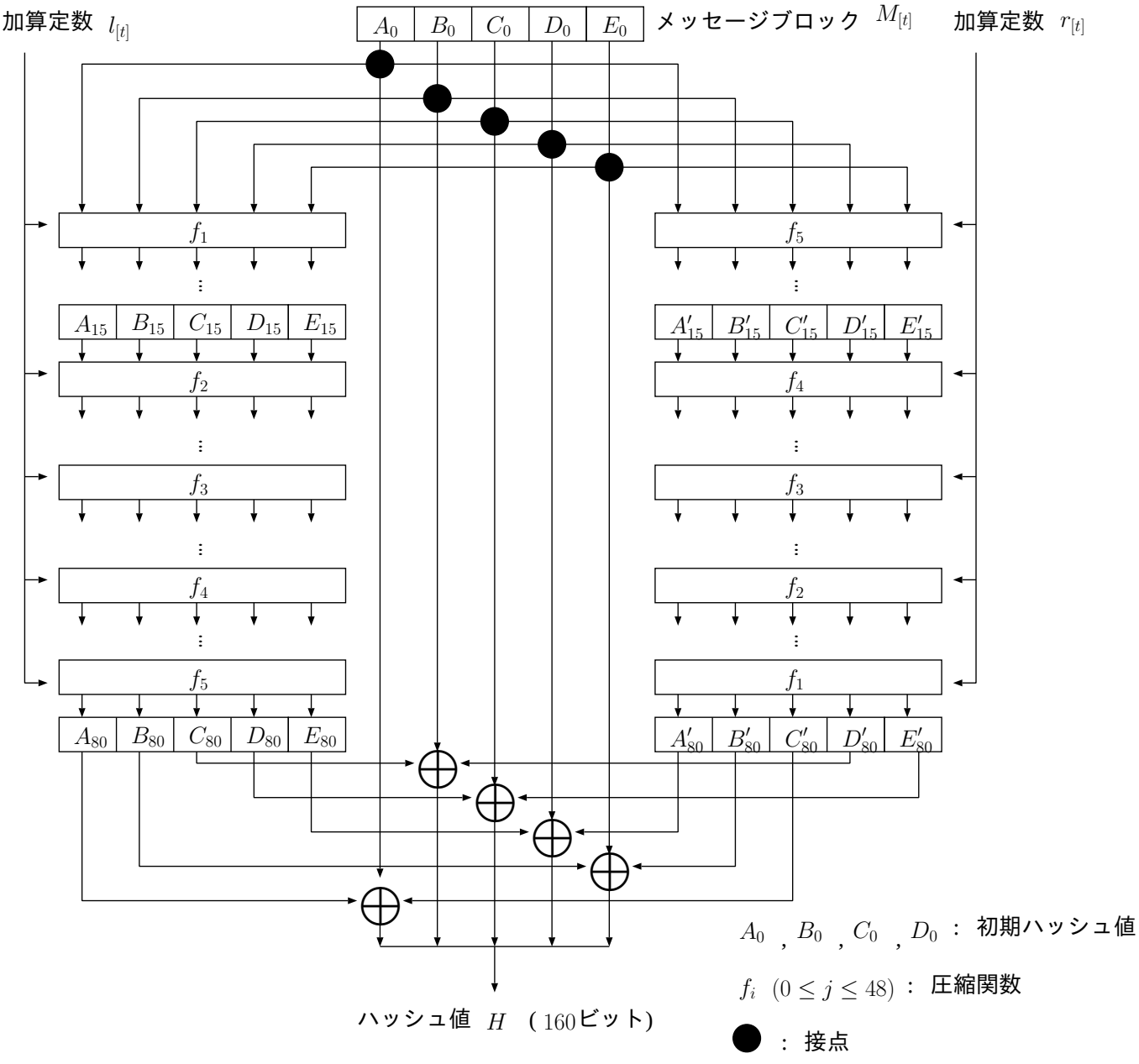



図 2.8　RIPEMD160 の構造

# 第 3 章

# 生成時間の比較と出力の偏りの比較

## 3.1　はじめに

パスワードを用いたハッシュ関数には計算が容易であることと，出力値に偏りがないことが望まれる．この章では，ハッシュ関数 MD4，MD5，SHA1，RIPEMD160 を生成時間および出力値の偏りに関して比較を行う．

## 3.2　生成時間の比較

パスワード認証に用いられるハッシュ関数は，認証を速く行うために計算が容易であることが望まれる．そこで，ハッシュ値の生成を繰り返し行い、その生成時間を測る．

### 3.2.1　手法

入力は “abcdefgh” とし，繰り返し回数は，10,000,000 回とした．そこから，1 回当たりの生成時間を平均値から算出した．

### 3.2.2　結果

計測結果を表 3.1 に示す．RIPEMD160 は他のハッシュ関数に比べ，時間を要した．

表 3.1　生成時間

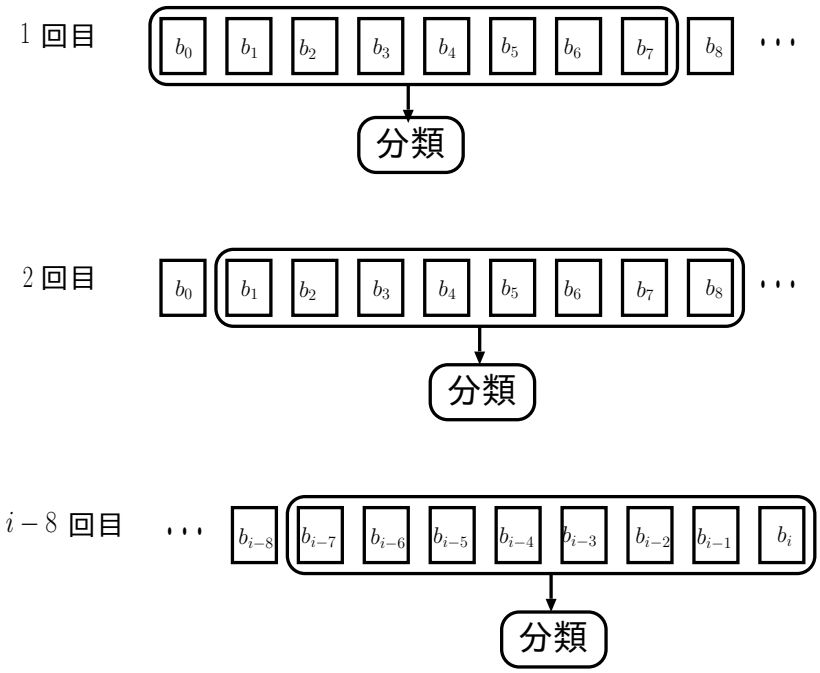
| ハッシュ関数 | 時間 ($\mu$sec) |
|---|---|
| MD5 | 1.7 |
| MD4 | 1.6 |
| SHA1 | 1.9 |
| RIPEMD160 | 6.7 |

## 3.3　出力の偏りの比較

ハッシュ関数の出力は，入力の値が少しでも変化した場合，全ての出力結果に変化が現れる．つまり似たような入力に対しても出力結果の偏りがないことが望まれる．偏りがあった場合，第三者から元の情報を推測される可能性がある．そこで，ハッシュ関数に任意の文字列の入力を行い，その出力の偏りを調べ比較を行う．

### 3.3.1　手法

本研究ではパスワード認証を対象としているため，ハッシュ関数への入力は，アルファベットと記号と数字の組合とする．安全性と言う観点では，パスワードはランダムな文字の組合せが理想ではあるが，実際には名前や英単語など，意味のある文字列や，覚えやすくするために文字数の少ないパスワードが利用されることが多く，入力には偏りが生じやすい．このことを考慮し，本実験では，入力の文字数を 1 文字から 6 文字とし小文字のアルファベット “a”から “z”までの組合せ全通りとした．ここで，6 文字まではハッシュ値の衝突が起こらないため，その入力から得られるハッシュ値を 8 ビットだけ切り出し，全ての場合に対して，256 通りに分類した．区間を小さくすることで衝突が起こりやすくなる．その分類した結果から出力の偏りを調べた．ハッシュ値 1 ビットを $b_i$，分類するビット列 (8 ビット) を $F_j$ としたときの分類の様子を図 3.1 に示す．このとき，平均値の差が大きいと出現頻度が偏っていることを示す．そこで，最も偏っている値の出現確率を求め，比較を行った．




$b_i$ : 出力値の1ビット部分

図 3.1　出力

### 3.3.2　結果

分類した結果を図 3.2 から図 3.25 に示す．図の横軸に分類した値を 0 から 255 にマッピングし，縦軸は分類した値の出現回数を表している．また，中央の横線は出現回数の平均を表している．この分類した結果から最も偏っている値の出現確率をそれぞれ求めたその結果を表 3.2 図 3.26 に示す.

### 3.3.3　まとめ

出力値を分類し出現確率を求めると，どのハッシュ関数も入力が多くなるに従い，出現確率が 0.003906 に近づいた．この値は全ての出現数が均等になったときの平均確率である．そのため，本実験で使用したハッシュ関数はどれも出力値の偏りに大きな差はない．

## 3.3　出力の偏りの比較

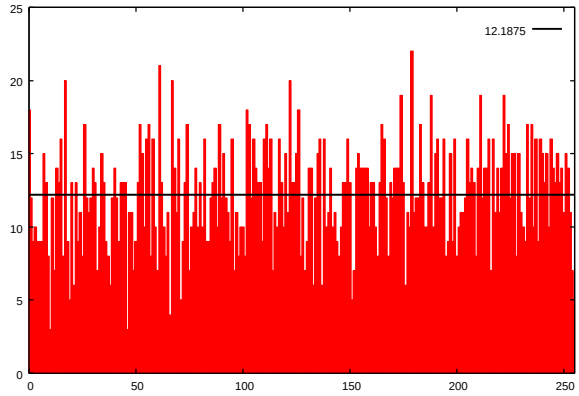



図 3.2　1 文字 MD4

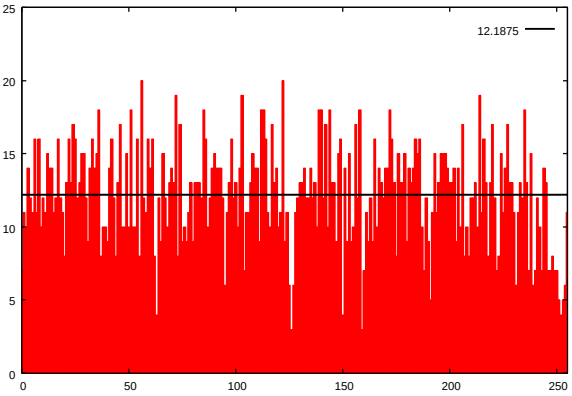



図 3.3　1 文字 MD5

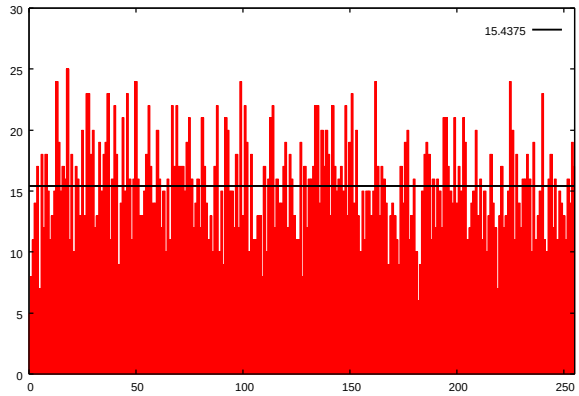



図 3.4　1 文字 SHA1

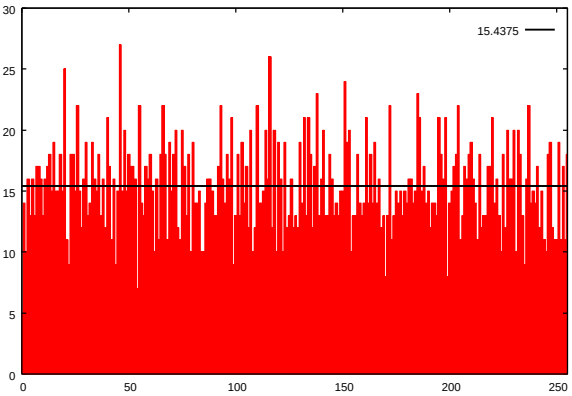



図 3.5　1 文字 RIPEMD160

## 3.3　出力の偏りの比較

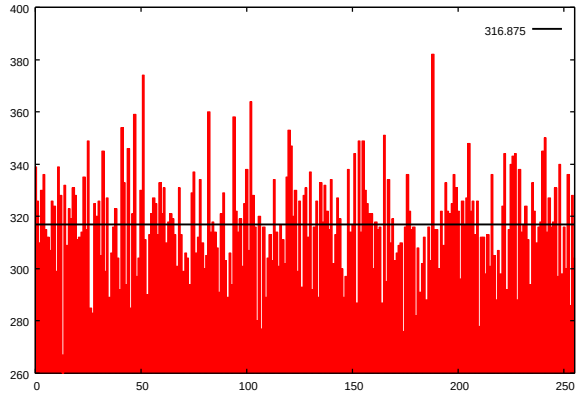



図 3.6　2 文字 MD4

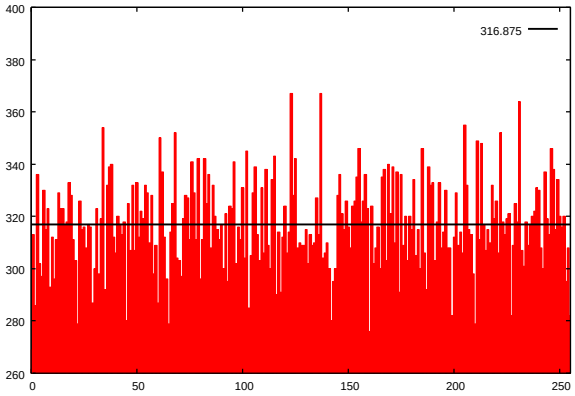



図 3.7　2 文字 MD5

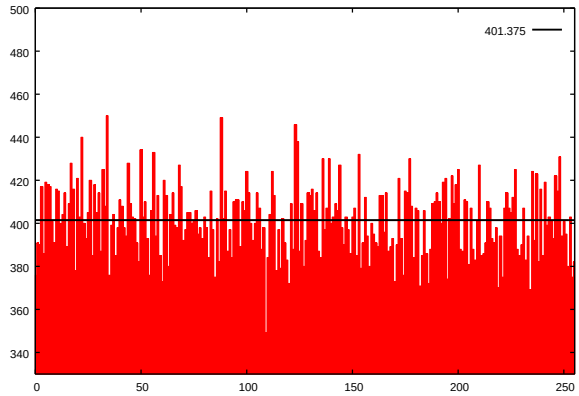



図 3.8　2 文字 SHA1

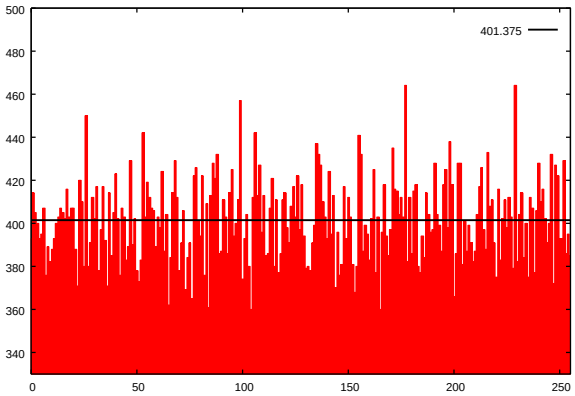



図 3.9　2 文字 RIPEMD160

## 3.3　出力の偏りの比較

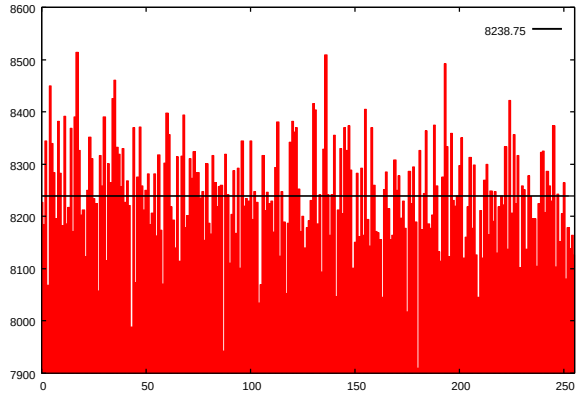



図 3.10　3 文字 MD4

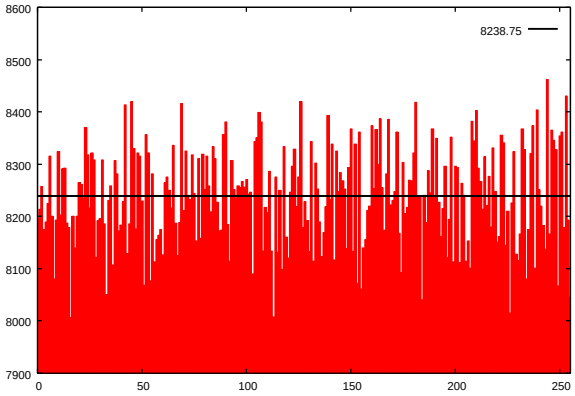



図 3.11　3 文字 MD5

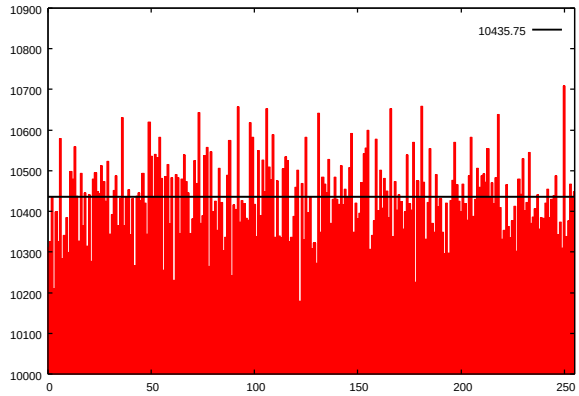



図 3.12　3 文字 SHA1

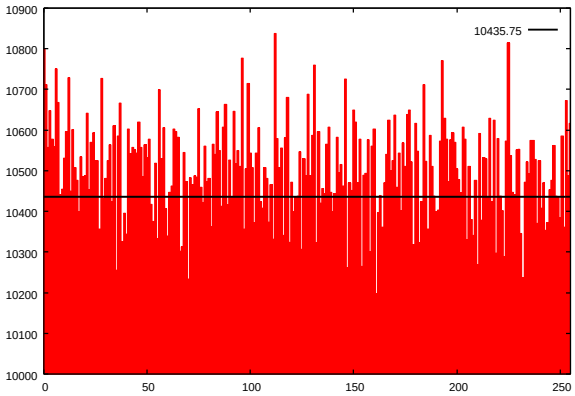



図 3.13　3 文字 RIPEMD160

## 3.3　出力の偏りの比較

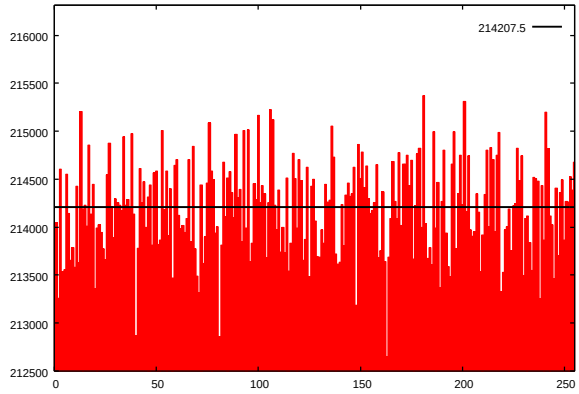



図 3.14　4 文字 MD4

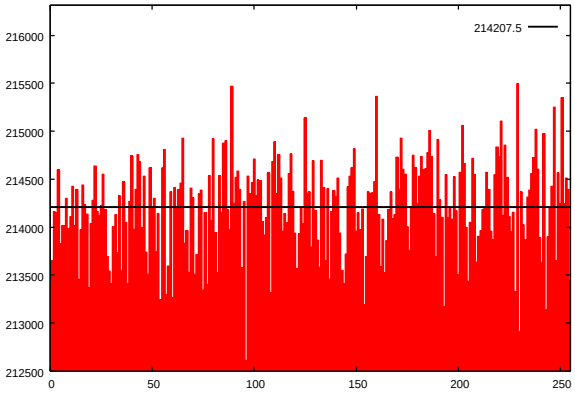



図 3.15　4 文字 MD5

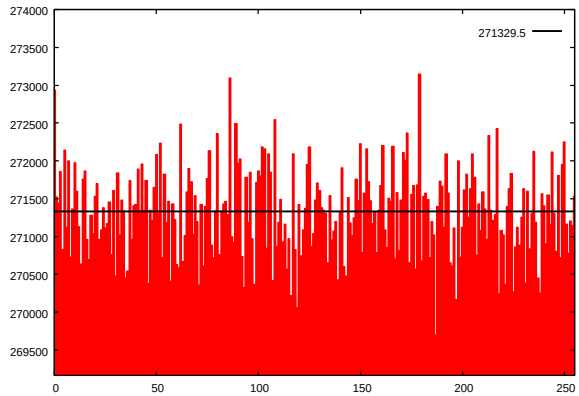



図 3.16　4 文字 SHA1

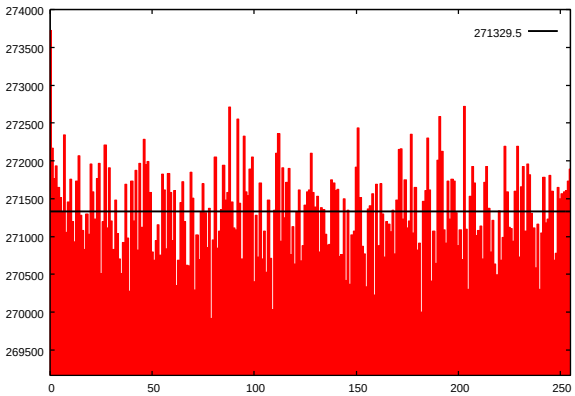



図 3.17　4 文字 RIPEMD160

## 3.3　出力の偏りの比較

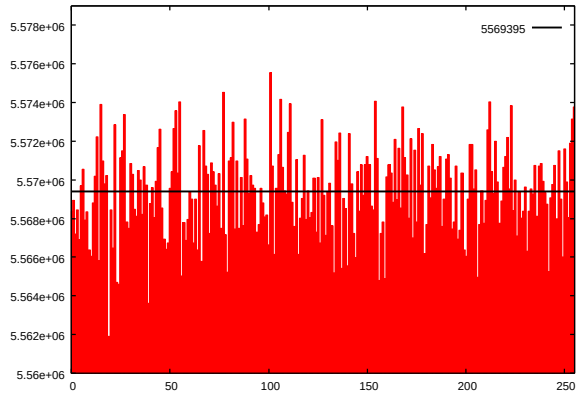



図 3.18　5 文字 MD4

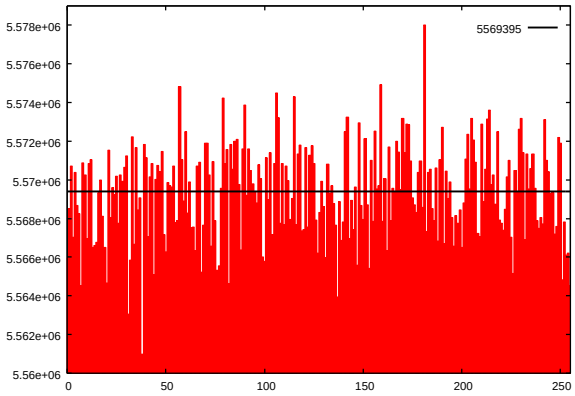



図 3.19　5 文字 MD5

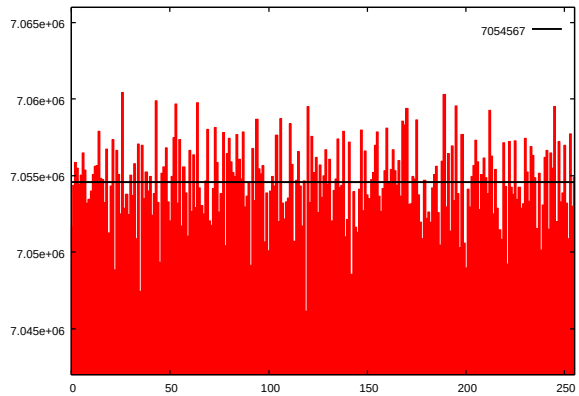



図 3.20　5 文字 SHA1

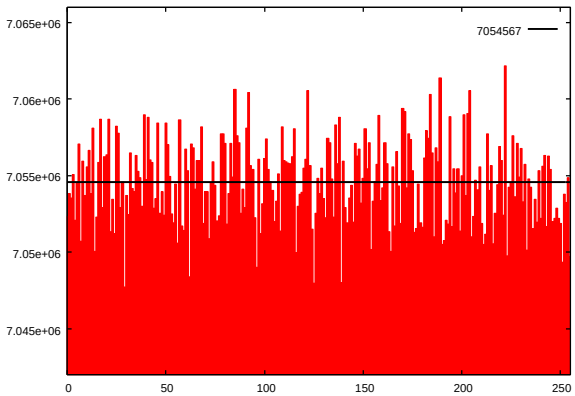



図 3.21　5 文字 RIPEMD160

## 3.3　出力の偏りの比較

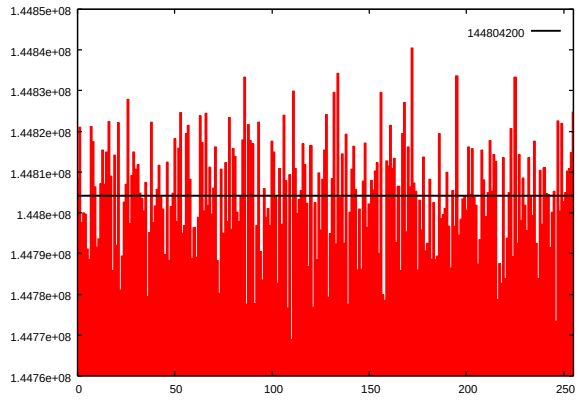



図 3.22　6 文字 MD4

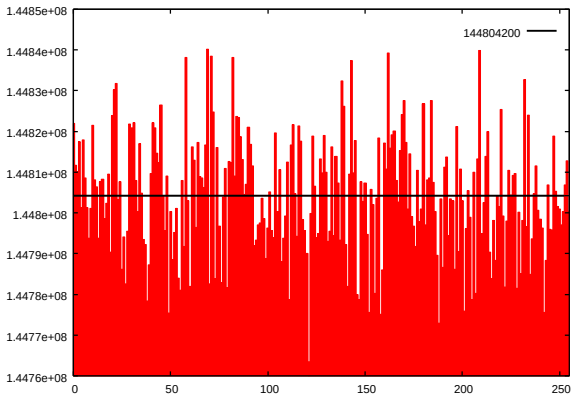



図 3.23　6 文字 MD5

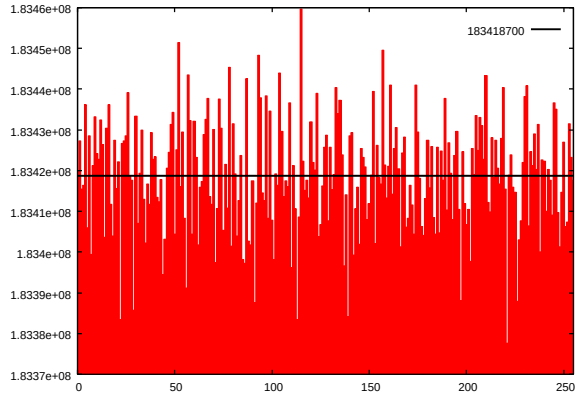



図 3.24　6 文字 SHA1

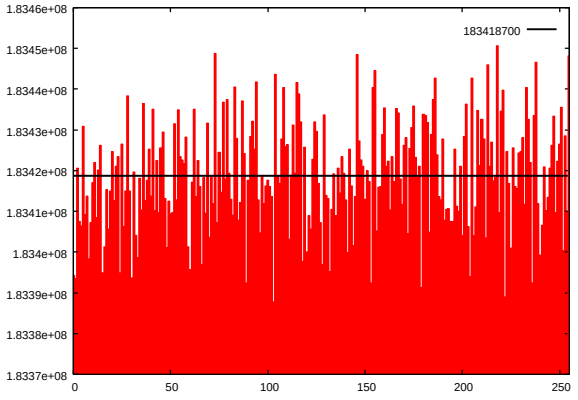



図 3.25　6 文字 RIPEMD160

## 3.3　出力の偏りの比較

表 3.2　出現確率

| ハッシュ関数 \ 文字数 | 1 文字 | 2 文字 | 3 文字 | 4 文字 | 5 文字 | 6 文字 |
|---|---|---|---|---|---|---|
| MD4 | 0.007051 | 0.004709 | 0.003750 | 0.003878 | 0.003901 | 0.003907 |
| MD5 | 0.000962 | 0.004524 | 0.003796 | 0.003877 | 0.003912 | 0.003905 |
| SHA1 | 0.001265 | 0.003397 | 0.004008 | 0.003932 | 0.003902 | 0.003905 |
| RIPEMD160 | 0.006832 | 0.004516 | 0.004034 | 0.003941 | 0.003910 | 0.003907 |

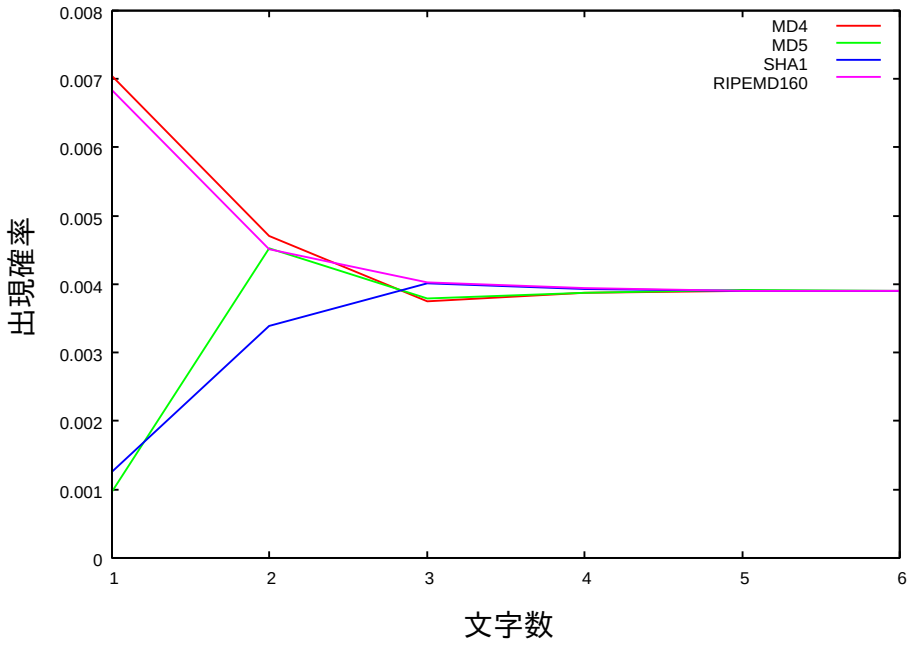



図 3.26　文字数による出現確率

# 第 4 章

# 結論

## 4.1 本研究のまとめ

パスワードを用いた認証に使用されているハッシュ関数に要求される，計算が容易であることと，出力値に偏りがないことについて，比較を行った．

計算が容易であることについては，ハッシュ値の生成に要する時間を測り比較を行った．生成時間は，RIPEMD160 は他のハッシュ関数に比べ時間を要したが，1 回の生成時間は 6.7$\mu sec$ なので，この生成時間で十分であり，いずれのハッシュ関数でも生成時間は問題はない．

出力値に偏りについては，任意の文字列の入力を行い出現確率を求め比較を行った．出力値の偏りは，入力数が多くなるにつれ，出現確率は平均に近づいていった．この結果から本実験で使用したハッシュ関数は出力値を 8 ビットで切り出した場合，どのハッシュ関数も偏りに大きな差がなかった．

## 4.2 今後の課題

出力値を 8 ビットで切り出した場合，どのハッシュ関数も偏りに大きな差がなかった．今後の課題として 8 ビット以上で切り出した場合では偏りがあるかを調べる必要がある．

# 謝辞

本研究を行うにあたり，福本昌弘准教授をはじめ，研究室の方々その他にも多くの人に大変お世話になり心から感謝致します．

福本昌弘准教授には，週次報告や研究室内での提出期限を過ぎてしまったり，研究が進まなかったりして，多大なる御迷惑を掛けしてしまい大変申し訳ありませんでした．そんな私に最後まで指導して頂きとても感謝致します．研究室で学んだこと経験したことは忘れません．また，合宿での変な方向へ飛んでいったロケット花火のこと，凄い匂いのお酒や赤ワインのことも忘れません．研究室での出来事をこれからの人生に役立てたいと思います．また，坂本明雄教授，吉田真一講師にも深くお礼を申し上げます．坂本明雄教授には本研究の審議を行って頂きました．吉田真一講師には発表の際の議長をして頂きました．吉田真一講師のおかげで，発表がやり易くなりました．

先生方だけでなく，研究や研究室内での生活において大変お世話になりました大学院生の福冨英次氏，佐伯幸郎氏，金井宏一朗氏にも心から感謝致します．福冨氏には，何も解らなかった私に，輪講から発表まで御指導して頂きました．佐伯氏は，研究が進まず締め切り間近になってしまった私を夜遅くまで残って面倒を見て頂きました．研究以外でも，食事や遊びに連れていってもらったりし幅広くお世話になりました．また，ミュージカルのお土産にアンケートボードを頂きました．そのアンケートボードを発表会で使用するつもりでしたが，当日は忘れてしまい使用することができませんでした．そこで来年は，今の 3 年生が発表する際に，是非このアンケートボードを使用してもらいたいと思います．金井氏には，プログラムを作成する際や梗概の書き方についてアドバイスして頂きました．また研究室では，よく外国の食べ物を頂きました．おこげ飴や現地のインスタントトムヤムクンの味は忘れられません．また刺激物の入ったお菓子や冷凍されたお弁当も頂きそこから冷凍保存の素晴らしさを学ぶことができました．私がこうして卒業できるのも福冨氏，佐伯氏，金井氏の御指導があったからです．心より感謝致します．

謝辞

また，研究を共にした福本研究室の一同にも感謝致します．3 年生のときの歓迎会で泥酔し，途中から記憶が無くなってしまいましたが，当時 3 年生は 6 人いたと思います．その後泥酔した私を介抱し，部屋まで運んで頂いたことを感謝と共にお詫び申し上げます．

研究を行う中で，何度も逃げ出したいと思うことがありました．しかし，皆様の支えがあったおかげで頑張ることができました．皆様と出会えたことを幸せに思います．

最後に今までに私と関わってくださった全ての方々に心より感謝致します．

# 参考文献

[1] 電子情報通信学会，“情報セキュリティハンドブック,” オーム社，2004.

[2] ブルース・シュナイアー，山形浩生，“暗号技術大全集,” ソフトバンクパブリッシング株式会社，2005.

[3] 岡本栄司，“暗号理論入門,” 共立出版株式会社，2002.