Quaternion（四元数）を応用したハッシュ関数の乱数性評価

須藤 智寛　　長瀬 智行

弘前大学大学院理工学研究科
〒036-8224 青森県弘前市大字文京町 3
E-mail:　nagase@eit.hirosaki-u.ac.jp

あらまし　暗号化に用いるハッシュ関数として、QHF (Quaternion Hash Function)が当研究室にて考案されている。QHF では、ハッシュ値の衝突耐性の向上に、四元数の演算の特徴である乗算の非可換性を活かしている。しかし、QHF の安全性が未だ評価されていないという現状から、今回、QHF の評価を行った。評価方法として、NIST SP800-22 により安全性のベースとなるランダム性について統計的な評価を行った。結果として、安全な暗号学的ハッシュ関数として利用するためは、QHF の改良が必要であることが判明した。

Security Level Evaluation of Quaternion Based Hash Function

Tomohiro Suto　　Tomoyuki Nagase

Graduate School of Science and Technology, Hirosaki University
3 Bunkyo-cho, Hirosaki-shi, Aomori, 036-8224 Japan
E-mail:　nagase@eit.hirosaki-u.ac.jp

Abstract One of the security features for designing hash functions is collision resistance. It means that it is difficult to find a pair of messages producing the same hash value. Before analyzing a hash function through searching for collision resistances, we need to examine the randomness of hash values as a first step in determining whether a hash system is suitable for generating appropriate random hash values. This paper performs a statistical test for randomness of hash values that are produced by quaternion based hash function (QHF). The evolution test is based on NIST SP800-22 evaluated method for statistical randomness. The obtained results show that we need to reconsider the design of QHF to meet highest requirements for building a secure hash function.

1. はじめに

現在、ハッシュ関数は復号化を行わない暗号方式として、電子署名やメッセージ認証コード、擬似乱数生成器など、多岐に渡り利用されている。既存のハッシュ関数の代表されるものには、SHA-1[1]、MD5[2]といったものが挙げられる。しかし近年、米国商務省標準技術局（NIST: National Institute of Standard and Technology）が、2010 年に複数の暗号方式の保証期限を設定した。これを「暗号の 2010 年問題」[3]と呼び、SHA-1 や MD5 といった、現在多岐にわたって使用されているハッシュ関数に関しての脆弱性が数多く指摘されている[4][5]。こういった問題を受け、従来のハッシュ関数の衝突耐性より高い衝突耐性を持った新型ハッシュ関数の研究が日々行われている。NIST は SHA-2[6]への移行を推奨しているが、SHA-2 は SHA-1 と同じ設計思想に基づいている。そこで、2007 年に

NIST は米国標準の新型ハッシュ関数アルゴリズム SHA-3 を公募し、2011 年 8 月現在は最終選考に残った 5 つのハッシュ関数が発表されており、2012 年内には SHA-3 が発表される予定である。

上述の背景の中、従来の概念とは違う Quaternion（四元数）を応用した新型ハッシュ関数アルゴリズム QHF (Quaternion Hash Function)が当研究室にて考案された[7]。提案された四元数を応用した新型ハッシュ関数アルゴリズムでは、ハッシュ値の強い衝突耐性の確保のために、四元数の特徴である乗算の非可換性を利用している。しかし、考案された新型ハッシュ関数 QHF のハッシュ値の安全性評価についてこれまで議論がなされておらず、QHF の暗号としての性能が明らかになっていなかったという現状から、本研究では QHF の安全性評価を行った。また、ハッシュ関数は他の暗号アルゴリズムにおいて擬似乱数生成器として利用されることがあり、さらには SHA-3 選考時の重要な評価項目に、擬似乱数生成などのアプリケーション利用時の安全性が挙げられていることから、今回 QHF の評価方法として、安全性のベースとなる乱数性について統計的な評価を行った。

2. 新型ハッシュ関数アルゴリズム

四元数でハッシュ関数を定義するとき、その最大の利点は乗算の非可換性となる。乗算結果が順序の影響を受けるため、四元数を利用するハッシュ関数は、入れ替えによる改ざんで発生するハッシュ衝突を許さない高い安全性を持つように定義することが可能となっている。

QHF が出力する数値は四元数の各要素であり、まず、data-quaternion processing (DQP)と呼ばれる入力データから四元数を作り出す前処理を行う。前処理の最初の段階では、入力データをブロック列として分割する。このブロック列を四元数列として扱うため、各ブロックを a_n, b_n, c_n, d_n のように等しい長さでさらに 4 分割する。

QHFの基本処理はブロック毎の四元数の乗算であるため(1)式がベースである。

$$H = Q_1 Q_2, \cdots, Q_n \qquad (1)$$

ここで、Q_1, Q_2,…,Q_n は入力データから得られた各四元数としているが、(1)式のままではデータの前後関係を固めるような配慮がなく 0 となるような元に対する処理もない。したがって、ハッシュ衝突を避けるため、乗算によってハッシュ値を算出する前に、ブロックの順序を固定化するような何らかの処理が必要となる。図 2 では、あるブロックとその前後、計 3 つの連続するブロックを利用し、各ブロックの要素から a_n ,b_n, c_n, d_n を利用して四元数 Q_n を計算することを示している。

また、他のデータから同じ四元数が生成可能であった場合、ハッシュ衝突が発生しやすい。この対策のため、QHF のアルゴリズムではあるブロックとその前後のブロックの要素を利用し、ブロックの前後関係を強固なものにしている。1 つのブロック内でデータの攪拌を行うだけではハッシュ関数の堅牢性を高めるためには不十分と考え、計算中に異なるブロックから要素を利用している。

前処理ではただ四元数の演算を行うだけではなくビットシフトも行っている。このビットシフトは s_n を利用したリングシフトである。s_n での M には巨大素数、もしくはそれらの積を利用する。高い衝突耐性を得るために、M は 128 ビット以上の数である必要がある。前処理で、Q_n を生成しているが、計算中に要素の値がすべて 0 であるようなブロックが存在する可能性がある。

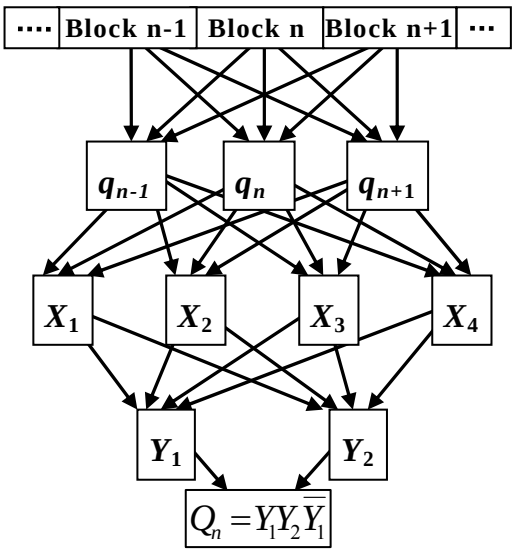

図1　QHFの前処理

入力データから得たブロックで値が 0 となるものが 1 つであれば演算 Q で対策されているが、それ以上の数の連続ブロックが 0 であるような

場合には、Q_nが 0 となり、 以降の計算もすべて 0 となってしまう。そこで、0 となるブロックの要素数 m を数え上げ、要素が 0 となるブロックの直前のブロック Q_n の値を利用した関数を利用する。この式で得られる四元数を 0 の連続部分のブロックの要素として利用する。
QHF の処理概要が図 3 である。この図で、図 2 で示した前処理の部分は *DQP* と略している。

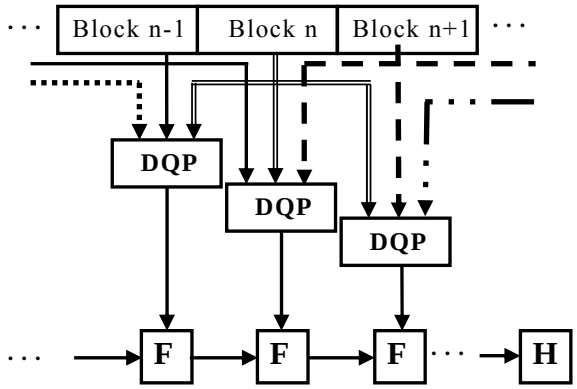

図 2: QHF 処理の概要

ハッシュ衝突の確率であるが、以下のように考えることができる。まず、2 つの四元数 p と q があり、そのベクトル部分がそれぞれ U、V であるとする。さらに、これらのベクトルは等しいとする。四元数の乗算は 2 つの四元数間でベクトル部分が等しい場合、(2)式のように乗算の非可換性が失われてしまう。

$$\begin{aligned} qp &= (q_0 p_0 - V \cdot U \ , \ p_0 V + q_0 U + V \times U) \\ &= (q_0 p_0 - \|U\|^2 \ , (p_0 + q_0) U) = pq \end{aligned} \qquad (2)$$

したがって、処理中にこのような場所があればそこが攻撃の対象となる。

そこで、ベクトル部分が等しい四元数が連続する確率を考える。一様乱数列の入力データから切り出したブロック列によって生成した四元数列で考えると、これは四元数の要素長 n によって決定し、一致する確率は $1/2^{3n}$ である。四元数が完全に一致して p=q となる確率は $1/2^{4n}$ である。*QHF* の強度は要素長に依存すると考えられるため、要素長 n=128 であれば、ベクトル部分の一致確率 $1/2^{384}$ で、四元数の完全一致確率は $1/2^{612}$ である。

3. QHF の評価

3.1. 乱数検定

本研究にて使用する、NIST(米国標準技術研究所) Special Publication 800-22[5](以下、NIST 乱数検定)は統計的な乱数検定法である。NIST 乱数検定は、表 1 に示す15種類の検定法で構成されており、各検定法は標準正規分布またはχ^2分布に基づいて行われ、p-valueと呼ばれる値が出力される。p- valueは、0から1の間の実数値で、実験で得た経験分布とその理論上の分布との適合度を表す。1ならば適合、0ならば非適合を意味する。つまり、p-valueが1に近づくほど、検定対象として入力された系列が真の乱数系列に近づくということである。

個々の検定に対しては、NISTは棄却率をα＝0.01(1%)に設定しており、p-value≧0.01のとき、良い乱数列であると判定され、そうでないときは良い乱数列ではないとされる。また、標準正規分布に基づいて検定が行われる場合、p-value は以下の関数 **erfc** (complementary error function)を用いて計算される。

$$\mathbf{erfc}\ (z) = \int_z^\infty \frac{2}{\sqrt{\pi}} e^{-x^2} dx \qquad (3)$$

χ2 分布に基づいて検定が行われる場合は、以下の関数 **igamc** (incomplete gamma function)を用いて p-value が計算される。

表 1 :NIST 乱数検定に含まれる検定法一覧

	検定名
1	一次元度数検定
2	ブロック単位の頻度検定
3	累積和検定
4	連の検定
5	ブロック単位の最長連検定
6	2 値行列ランク検定
7	離散フーリエ変換検定
8	重なりのないテンプレート適合検
9	重なりのあるテンプレート適合検
10	Maurer のユニバーサル統計検定
11	近似エントロピー検定
12	ランダム偏差検定
13	種々のランダム偏差検定
14	系列検定
15	線形複雑度検定

$$\mathbf{igamc}\ (a,z) = \frac{1}{\Gamma(a)} \int_z^{\infty} e^{-t} t^{a-1} dt \qquad (4)$$

$$\Gamma(a) = \int_0^{\infty} e^{-t} t^{a-1} dt \qquad (5)$$

擬似乱数生成器に関しては、得られる複数の乱数系列について検定を行い、

[i] p-valueが0.01以上になる比率(Proportion)

[ii] p-valueの一様性(Uniformity)

によって擬似乱数生成器の検定が行われる。

[i]については、表2における”Proportion(P:合格比率)”の評価で、p-valueが0.01以上になる系列数は正規分布$N(\mu,\sigma^2)$に従うと考え、$\mu \pm 3\sigma$以内に収まれば良いとする。

[ii]については、表 2 における”Uniformity(U:一様性)”の評価で、区間[0, 1)を 10 分割し、各区間に属する p-value の個数が均等であるかどうかを χ^2 分布によって検定する。具体的には、乱数列の個数が s 個の場合、$1 \leqq i \leqq 10$ について、F_i を区間[(i-1)/10,i/10)に属する p-value の個数とするとき、次式を計算し、

$$\chi^2 = \sum_{i=1}^{10} \frac{\left(F_i - s/10\right)^2}{s/10} \qquad (6)$$

さらに、p-value の p-value ともいえる **igamc**(9/2, χ^2/2)を計算し、**igamc**(9/2, χ^2/2)$\geqq$0.0001 となった場合、良い擬似乱数生成器であると判定される。

3.2. 計算機実験と考察

QHF のハッシュ値は 256 ビットとし、任意の入力値に対して複数回実行した後に、得られたハッシュ値をすべて連結した出力系列を NIST 乱数検定の検定対象とする。10 万ビット、100 万ビットの系列をそれぞれ、1000 本と 5000 本、500 本と 1000 本生成し、NIST 乱数検定でそれら各系列を検定した結果を表 2 に示す。なお、表の中で”x“は検定に合格しなかったことを表し、”-“は系列長が NIST 乱数検定の推奨値に満たないため、検定を行わなかったことを表している。

表 2 は今回の実験結果のまとめとなる。その表 2 から、QHF のハッシュ値を連結した乱数系列は、NIST 乱数検定において半数以上の検定法で「良い乱数性をもった乱数列ではない」と判定されることがわかる。10 万 bit 系列に関しては三分の二の割合、100 万ビット系列においては全 15 検定中 8 つの検定で不合格と判定された。NIST により推奨されている乱数系列 bit 数は 100 万 bit であり、乱数系列の本数は 1000 本となっている。そのことを考慮すると、表 2 内の D 列の検定結果の信頼が高いことになる。

NIST によって示される[i]Proportion の合否条件は、個々の検定項目が満たすべき条件であって、検定対象の乱数列がすべての検定項目で合格すべきかどうかは示されていない。

表 2：QHF 出力系列の検定結果

P：合格比率　U：一様性

A.10 万ビット×1000 本　　B.10 ビット×5000 本

C.100 万ビット×500 本　　D.100 万ビット×1000 本

○：PASSED, x：FAILURE, -：non-result

	A		B		C		D	
	P	U	P	U	P	U	P	U
1	x	x	x	x	x	x	x	x
2	x	x	x	x	x	x	x	x
3	x	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x	x
6	○	○	x	○	○	○	○	○
7	○	○	○	○	○	○	○	○
8	-	-	-	-	○	○	○	○
9	-	-	-	-	x	x	x	x
10	-	-	-	-	x	x	x	x
11	x	x	x	x	x	x	x	x
12	-	-	-	-	○	○	○	○
13	-	-	-	-	○	○	○	○
14	○	○	○	○	○	○	○	○
15	-	-	-	-	○	○	○	○

また、仮に真のランダム性を持った乱数列を NIST 乱数検定に通した場合でも、全ての検定項目において合格する確率は、二項分布を用いて概算すると約 54%、正規分布を用いて求めた場合(検定系列の本数が多い場合)の確率は約 78%となることが分かっている[6]。したがって、すべての検定項目が合格したときに限り、良い乱数性を持つというように判断することは正しいとは言えないのである。

この状況を踏まえ、全体的にではなく、ある一つの検定項目に絞って考察してみる。表 3 に、100 万 bit×1000 本の一次元度数検定(Frequency Monobit Test)の結果の詳細の一部を載せている。この検定は乱数性評価において最もベースとなるテストであり、このテストに合格しない場合、他の乱数検定においても合格しない可能性が高くなる重要な検定である。この検定は、’0’と’1’の個数をそれぞれ数え上げ、2 つの個数が近くなるほど乱数性が高いと評価する検定である。そして、表 3 から分かることは、’0’の割合が’1’の割合よりも必ず多くなっているということである。1000 本分の結果を全て確認したところ、1000 本全てにおいて’0’の割合が多くなっていた。

表 3：一次元度数検定の詳細結果
（一部抜粋）

BITSREAD = 1000000	
0s = 509770	1s = 490230
0s = 510032	1s = 489968
0s = 509741	1s = 490259
0s = 509503	1s = 490497
0s = 510248	1s = 489752
0s = 508882	1s = 492228
0s = 509096	1s = 490904
0s = 507750	1s = 492250
0s = 508813	1s = 491187
0s = 509894	1s = 490106

この結果は、QHF の出力が均等になっておらず、数値的に見ると値の小さい方にハッシュ値が偏っている可能性が高いということを示している。

表 4：一次元度検定の各出力成分の結果
（一部抜粋）

BITSREAD = 1000000			
W component		X component	
0s	1s	0s	1s
509122	490878	508821	491179
509178	490822	509282	490718
509086	490914	509287	490713
509147	490853	509206	490794
510242	489758	508604	491396
508806	491194	509685	490315
Y component		Z component	
0s	1s	0s	1s
509247	490753	509892	490108
509835	490165	510126	489874
509276	490724	508891	491109
509537	490463	510176	489824
59951	490049	510038	489962
509843	490157	508928	491072

表 5：256 ビットごとの一次元度検定の結果

assess 256bit x 1000000streams	
C1	100431
C2	101424
C3	99336
C4	128461
C5	74138
C6	80560
C7	86445
C8	90518
C9	94289
C10	144398
P-VALUE	0.000000
PROPORTION	0.987711

表 2、表 3 は、QHF のハッシュ値である四元数の全ての成分の出力値を NIST 乱数検定に通した全体像を捉えた検定結果であるので、次に各成分に分けた際の乱数性を調べた。その結果の一部が表 4 である。QHF の出力値は[*w*-*x*-*y*-*z*]のように演算後の四元数の各成分となっており、表 4 からは一次元度数検定による各成分の偏りはほとんどないことが判る。よって、成分の違いによる値の偏りはないといえる。表 3、表 4 から、必ず“0”の割合が“1”よりも多くなるっているということが分かるが、それらは QHF の約 3800 回分の出力値を連結し、100 万ビットにまで繋ぎ合わせた系列をひとつの乱数列としてみた場合の結果であり、QHF の試行一回ごとの乱数検定結果を示したものではない。そこで、256 ビットごとの乱数列の検定を行ってみると、必ずしも”0“の割合が”1“の割合より多くならないということが判明した。さらに、表 5 は 256 ビットの乱数列を 100 万本検定した結果であり、p-value はある程度ではあるが 0 から 1 の間で均等になっているという結果がわかる。

上記のことを総合的に考えると、QHF の各実行では“0”と”1“の数は偏りを見せないが、QHF を数多く実行すると明らかな偏り（”0“が”1“よりも多くなる）が浮き彫りとなり、その偏りは QHF の各成分[*w*-*x*-*y*-*z*]の出力値の違いが影響するものではなく、演算により全体的にある程度の偏りが存在しているということがわかる。そこから、繰り返し QHF で処理される F 関数などの処理系統にこのような偏りの原因があるのだと予測がつく。

4. まとめ

QHF (Quaternion Hash Function)について、安全性評価の第一歩となる暗号としての評価のベースとなるランダム性について研究を行った。それは、ハッシュ関数は単独ではなく、他暗号内で擬似乱数生成器として使用されることもあり、そのことからハッシュ関数の乱数性の評価は特に重要であると考えてのことである。

今回の評価の結果では、QHF のハッシュ値には偏りが存在することが判明した。一次元度数検定において明らかに、’0’が’1’よりも多い割合を占めるというものである。しかし、偏りが見えてくるのは複数回の QHF の出力結果を繋げ合わせたときで

あり、QHF の実行一回((256 ビット)において、その偏りは現れない。また、四元数の各成分の乱数性は同程度であること実験的に判明した。暗号が解読される際は偏りや規則性を見つけ、その部分が標的となり攻撃の糸口とされる。そのため、安全な暗号学的ハッシュ関数として利用するためは、QHF の改良が必要であることが判明したことが今回の研究報告である。

参考文献

[1] U.S Department of Commerce, National Institute of Standards and Technology, “Secure Hash Standards (Federal Information Processing Standards Publication 180-1),” 1995. http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[2] R. L. Rivest, “Request for comments 1321: The MD5 message digest algorithm,” The Internet Engineering Task Force, 1992.
http://www.ietf.org/rfc/rfc1321.txt.

[3] 宇根正志, 神田雅透, 暗号アルゴリズムにおける 2010 年問題について, 日本銀行金融研究所/金融研究, Aug 2006.

[4] X. Wang and H. Yu, “How to break MD5 and other hash functions,” in EUROCRYPT 2005, Lecture Notes in Computer Science, R.Cramer ed., vol.3494, pp.19-35, Springer-Verlag, 2005.

[5] X. Wang, Y. L. Yin, and H. Yu, “Finding collisions in the full SHA-1,” in CRYPTO 2005, Lecture Notes in Computer Science, V. Shoup ed.,vol.3621, pp.17-36, Springer-Verlag, 2005.

[6] U. S. Department of Commerce, National Institute of Standards and Technology, “Secure Hash Standards (Federal Information Processing Standards Publication 180-2),” Aug.2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[7] 門間朋美, 須藤智寛, 長瀬智行, 吉岡良雄, “Quaternion(四元数)を応用した暗号に関する研究,”IPSJ SIG Technical Report, CSEC-50, No.9, Vol.2010, pp.67-70, Jul.2010.

[8] Andrew Rukhin, et al., “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications” Special Publication 800-22 Revision 1a, Lawrence E. Bassham III .ed., April 2010.

[9] 廣瀬勝一, 疑似乱数生成系の検定方法に関する調査　調査報告書, 2004.