アルゴリズム 27－3
（1992．5．21）

制約付き疑似乱数を用いた認証方式

山上俊彦
ＮＴＴ通信網総合研究所

特定の制約条件を満たすような制約付き疑似乱数を用いた認証方式について述べる。特にｎビットパターンにおいて１のビット数がちょうどｍであるような疑似乱数を生成し、その排他的論理和を生成して送信し、ビットの一致数によって認証することによってターンアラウンドの短い高速な認証通信を行うことが可能である。２５６ビットの鍵のランダムに選択した２９ビットを反転することによって、１２８ビットの秘密鍵認証方式と同等のしらみ潰し攻撃に対する認証強度をもつことを示した。ネットワークを介した分散環境においてサーバの負荷を軽くすることができ、必要に応じて認証強度を調節できる認証方式として有効であることを論じた。

AN AUTHENTICATION METHOD USING CONSTRAINT PSEUDO-RANDOM NUMBERS

Toshihiko YAMAKAMI
NTT Telecommunication Networks Laboratories
1-2356-523A, Take, Yokosuka, Kanagawa 238-03, Japan
e-mail: yam@nttmhs.ntt.jp

An authentication method using constraint psuedo-random numbers is proposed. Particulaly, psudorandom numbers which have exact 'm' 1's in n-bit pattern are generated and used to mask n-bit secret keys. An authentication using this masked secret keys is useful for short turn-around time. 29 bit reversing out of 256 bit keys enables the same cryptic strength against exhaustive attacks. The proposed method is effective in network oriented distributed environment since it is asymmetric load balance with advantages to servers. In addtion, this method enables flexible strength tuning with scalability.

1．はじめに

ＬＡＮなどで結ばれた構内オフィスシステムにおいて、トランザクション毎に利用できるような高速な認証方式が必要とされる。コネクションレス型の通信ではアソシエーションという概念がなく、データを送信するたびにそれが正しい相手から送信されたものであるかどうかが確認できることが望ましい。筆者は、クライアント／サーバ方式の分散環境での通信（例えば、Distributed Office Application Model[ISO91, 春田92]による通信環境）において、認証に際して冗長な情報を送信することによって認証に利用する鍵を更新し、一度使った鍵は使わない方式を提案し、その処理性能を論ずる。ここで検討する方式は冗長な情報として ｎビットの中のあらかじめ通信するエンティティ間で共有したｍビットを反転するという方法を用いることにより、強度を適応的に変化させるとともに、認証を行なう側では単純な演算により高速な応答を可能とするなどの特徴を持つ。

2．アプローチ

2．1　提案方式の概要

認証には、秘密鍵方式、公開鍵方式などがある。従来の方式はその安全性に中心があり、一度認証が行なわれれば、秘密の通信路に基づいて通信が行なわれることを仮定している。ＬＡＮなどにおいては、コネクションレス型として、必要なトランザクション毎にオペレーションを単体で送信するデータ送信コマンドだけで、従来のようなアソシエーション制御を行なわない型の通信が利用されている。このような方式では、通信路の性質に応じた送信単位（たとえば、１０２４バイト）のデータを送るたびにそれが正しい相手から送信されたものであるかどうかを確認する必要がある。分散環境ではこのような認証を高速で行なう必要がある。

鍵は認証を行なう２つのプロセスの間において共有されるが、認証に際して冗長な情報を送信することによって認証に利用する鍵を更新し、一度使った鍵は使わないことによりネットワーク上での鍵の盗用を防ぐことができる。

筆者は、暗号を使わない使い捨て鍵による高速な認証方式を提案する。基本的なアイディアは、ｎビットの鍵とその鍵を更新する一方向性関数を共有することを仮定し、ｎビットのうち、ｍビットだけが一致するような認証情報を作成して送信することによって、暗号化しなくても秘密に認証を行なうことができる、というものである。ｍを適当な大きさにとれば、ネットワーク上でこの部分的に与えられた情報を盗んで、鍵を推察することは困難である。また、ｍを適当にとることによって十分な量の情報を冗長な情報として送ることによって、鍵の更新を複雑にし、この点からも鍵の安全性を高めることが可能である。

ｎビットのうちｍビットだけが一致するかどうかのチェックは排他論理和演算によってソフトウェアでも高速に実現でき、汎用性も高い。

以下、これをNPM (Nondeterministic Partial Matching authentication)方式と呼ぶ。

2．2　従来の方式

従来の方針では、図１のようになっていた。ネットワーク上での盗聴を避けるため、なんらかの方法での暗号化が必要である。もちろん、この暗号化を通信上でハードウェアで行なうことによって高速化することは可能である。今回の検討では、そのようなハードウェアによるネットワーク上での秘匿操作が行なわれていないようなものにおける高速認証を検討する。

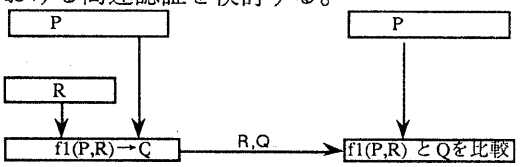


図１　従来の一方向性関数を用いた認証方式

この方式では、ネットワーク上で認証を要求した場合、その認証が終了して実際の処理が始まるまでに、認証要求側での乱数生成(Step P1)→認証要求側での f1関数(Step P2)→認証受付側でのf1関数(Step P3)、という３段階の操作が必要であった。

3．NPM 方式認証のアルゴリズム

分散アプリケーション上で、サーバに負荷をかけないで、ネットワーク上で起こる大量のトランザクションの１回１回に対して認証を行ない、セキュリティを強化したい、というのが本研究の目的である。分散環境では、サービスを提供するサーバ１つに対してサービスを受ける数多くのクライアントが存在する。このため、サーバとク

ライアントにおいて負荷が非対称であり、サーバ側の負荷が軽い方式があれば有効であると考えられる。また、全体のスループットが同じでもサーバのサービスを受けるまでのレスポンス時間を小さくすることができれば、分散環境では有効である。

この目的のために図２に示すような非対称な認証方式を提案する。

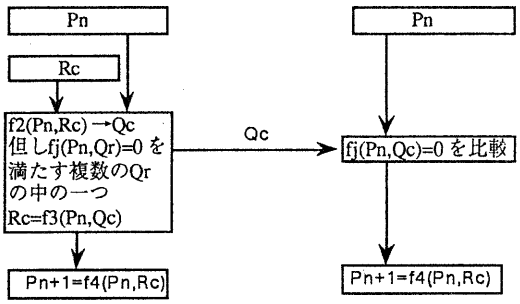


図２　提案する方法での認証方式

この方式では、認証までに認証要求側での乱数Rr生成(Step P1)→認証要求側での f2関数(Step P2)→認証受付側でのfj関数(Step P3)、という３段階の操作が必要になるが、f2, fj を大幅に高速な関数を利用することが可能になる。

筆者は、認証のfj関数において認証に成功する入力を複数認めるような関数を利用することにより、大幅に高速な関数を適用することを発見した。また、認証に成功する入力が十分にたくさんあれば、そのどれを使ったかの情報を利用してPn を更新することが可能である。

具体的には、高速な関数とは、例えば２つの数の異なるビットの数を数える演算である。nビットの鍵をmビットだけ変えて送信することが考えられる。この方式では正確にmビットだけが異なっていることを認証受付側で確認することはきわめて容易である。なぜならば、受け取った鍵と秘密に保持している鍵を排他的論理和し、そのビットの１が立っている数を数えればよいのである。排他的論理和 はほとんどの計算機でマシンワードサイズ毎に１命令で実現できる。マシンワードサイズが３２ビットでn=128 なら１２８／３２＝４回である。また、ビットを数える演算は、計算機で保持しえるテーブルのサイズを2^{TSize}とすると、TSizeビットについてそのビットパターンについてビットの１が立っている数を予め2^{TSize}個の要素を持つ表に設定しておいて参照すれば、TSize ビット毎に１回のローテーションとビット積と表アクセスと加算で実現できる。n ビットに対してはこれら４つの演算をそれぞれn / TSize回行なえばよい。TSize=16とすると表のサイズは65536となり、パソコンでも簡単に保持することができるサイズである。n=128, 256でTSize=16 を想定すると、ビット毎にチェックをして加算する場合では、ローテーションとビット積と加算がそれぞれ128, 256回必要なところで、表１のようになる。

表１　n=128, 256 におけるビット数え上げ演算のコストの比較

		ローテーション	論理積	加算	配列アクセス
n=128	ビット毎	128	128	128	0
	表	8	8	8	8
n=256	ビット毎	256	128	256	0
	表	16	16	16	16

４．暗号強度の検討

４．１　NPM における m の選択

最近の暗号研究では、１２８ビット暗号が検討されている。これは従来の６４ビット暗号が提供するしらみつぶし攻撃に対する2^{64}程度の場合の数の強度では十分でないと判断されているからである。NPM でも１２８ビット暗号と同程度の強度を持つ暗号方式を検討する。１２８ビットの中からnビットをたてるという演算では、しらみつぶし攻撃に対して2^{128}の強度を確保することはできない。

なぜなら

$$_{128}C_n < 2^{128}$$ が常になりたつからである。

よって、１２８ビットより大きな鍵を共有し、その中から何ビットかを一致させる方式を検討する必要がある。鍵をnビットとし、これをｍビット一致させるものとすると、その場合、認証情報として利用可能な場合の数は $_nC_m$ となる。すべてのパターン 2^nのうちからこれだけの数の認証情報がでたらめに行なっても破られ得るのであるから、これが１２８ビット暗号と同

適度の強度を持つためには、

$2^n / {}_nC_m > 2^{128}$ である必要がある。

これより、${}_nC_m < 2^{128}$ となることが必要である。

一方、mの値が小さければ小さいほど、鍵情報の露出する度合が大きくなり、これは暗号の強度を弱めることになる。このためmは上記の不等式を満たす範囲内で可能な限り大きいほうがよいと言える。

nは計算機のソフトウェアで実現した場合、32の倍数であることが高速に実現する上で望ましい。表2にnを160あるいは256を選択した、kを選択した場合の組み合わせの数と2^{128}との比較を示す。

表2　組み合わせの数と2^{128}との比較

組合せ	組合せの数	2^{128}との比較
${}_{160}C_{40}$	8.638e+37	0.2539×2^{128}
${}_{160}C_{41}$	2.528e+38	0.7430×2^{128}
${}_{160}C_{42}$	7.163e+38	2.1051×2^{128}
${}_{256}C_{28}$	1.910e+37	0.0561×2^{128}
${}_{256}C_{29}$	1.501e+38	0.4413×2^{128}
${}_{256}C_{30}$	1.136e+39	3.3392×2^{128}

これより、256ビットならば　$m<30$ でよいことがわかる。参考までに160ビットならば$m<42$ならばよい。計算機での取り扱いからは2の累乗であることが便利であるので、以下256ビットを中心に考察する。

4．2　暗号強度を高める方法

問題となるのは、このmの値によった場合、本当に、ネットワーク上で認証シーケンスを盗聴しているものが、不法な認証を行なうに足るだけの情報量を得ることを防ぐことができるかどうか、ということである。具体的には、いくつかのシーケンスからなる認証情報の交換を盗聴して秘密の鍵を解読することができるかどうかが問題となる。

暗号強度については、解読されない一方向性関数の選択が重要となる。どのような一方向性関数を選べばよいかについては、未解決の問題として将来の課題として残しておく。ここでは、一方向性関数が十分な複雑さをもっていないような環境において、その強度を補強する方法を2つ提案しておく。

ひとつは、nビットのうち、mビットだけでなく、m'ビット一致した場合にも認証を成功させる、ということである。このことによって解読は飛躍的に困難になる。送り側で疑似乱数を発生させて、mビットかm'ビットかを選択することにすれば、p個のシーケンスを解読に利用した場合のそれぞれのパターンにおけるビット差分のパターンは2^p通りとなる。より長いシーケンスを得れば、一方向性関数の性質を読み取るのには有利であると考えられるが、長いシーケンスになれば、飛躍的に調べる場合の数が拡大していき、解読を困難にすると考えられる。mビットかm'ビットかによって解読の困難さが異なれば、解読の容易な方がネックとなって破られる可能性がある。このため、m' としてはn-mを利用することができる。mビットが一致するのか、n-mビットが一致するのかは一致する鍵の数は全く同じであるので、しらみつぶしに対する強度は対称であるので、強度の弱い方に依存して全体の強度が弱まる度合が一番少なくて適当である。

もうひとつは、認証が成功したかどうかをネットワーク上でわからないようにすることである。認証が成功したことを示す情報を送られたビット差分を利用して表現することにすれば、ネットワーク上で盗聴してもそれが正しい認証情報を送ったのか、ダミーを送っただけなのかを判定することができない。このようにして、半分の情報がダミーになるように送信することによって、解読に対する強度を強めることができる。本認証方式は鍵の情報を部分的に公開する方式であるため、このような方式を併用して暗号強度を強化することが望ましい。前掲表2により、n-mビットとmビットの両方で認証を成功させる場合、256　ビットでは$m<30$であればよいことがわかる。

4．3　χ^2検定による評価

この暗号強度をシミュレーションで示すため、256ビットにおいてm=29の場合に、一方向性関数を使わず、鍵を固定した場合でも、ほとんどランダムな結果が出ることを実験した。

シミュレーションプログラムは　perlで記述し、乱数発生にはrand()を利用し、256

のそれぞれについて、χ^2検定を行なった。その結果を表３に示す。

表３　χ^2検定の結果
(256 ビットのうち29ビットをたてた場合)

試行回数	<1 %	1- 10%	10 - 90%	90 - 99%	99 %<
2 ** 12	5	0	221	27	3
2 ** 16	3	1	224	26	2
2 ** 20	4	7	226	18	1
2 ** 24	3	8	219	25	1

表３の結果より、ほぼ、ランダムに各ビットが設定されていると考えられる。なお、[山上92] に示したように、このｎビットのうちｍビットをたてたビットパターンは、ビットのかたまりとしては、相互関係がありランダムではない。多くのビットを選べば、ｎと　ｍの比によって設定されたビットとそうでないビットとの比率が与えられる。

２５６ビットのうち２９ビットだけを反転したデータを認証に利用するのであり、たとえその組み合わせの数が、2^{128}通りに近い場合の数があるにせよ、データの９０％が回線上などでオープンされることは利用している一方向性関数の性質によっては暗号強度を弱めることも考えられる。そこで、ｎビットとｎ－ｋビットが一致した場合のどちらにも認証が成功するようにした場合についてもχ^2検定を行った。表４に結果を示す。

表４　χ^2検定の結果
(256 ビットのうちランダムに
29ビットあるいは227ビットをたてた場合)

試行回数	<1 %	1-10%	10- 90%	90-99%	99 %<
2 ** 12	2	7	210	33	4
2 ** 16	4	8	235	9	0
2 ** 20	5	14	236	1	0
2 ** 24	2	10	237	7	0

このシミュレーションは　Ｃを用い、乱数にはrand() より遅いが乱数性に優れたrandom()システムコールを用いた。この結果は若干、単純なビット設定に比べてランダムに２９ビットの１を設定するほうを選ぶか、２９ビットの０を設定するかを選ぶかのばらつきによって、結果が影響されるようで、完全にランダムとはいえない。統計的にいえるものより１０－９０％の範囲にはいるものが１割位多くなっており、十分ランダムならば、２５６ビットのそれぞれのビットにおいて、試行回数のうちビット１がたつ回数が試行回数のちょうど半分に近かったり、半分から大きくずれたりする場合がもっと多いはずである。偏りが大きいよりは認証アプリケーション的には望ましいと考えられるが、この傾向の認証強度に与える影響は将来の課題である。

５．高速性の検討

ｎビットの中からｋビットを立てる演算、およびｎビットの２つのビットパターンのハミング距離を計算する演算は高速に実行することが可能である。

まず、単純にビットを立てる演算について考察してみる。ｎビットのうち、ごく小さいｍビットをたてる演算はただ、乱数を発生させてそのビットをたてるのがもっとも高速である。たっているビットが少ない間はほとんどふった乱数に基づいてビットをたてることが可能であるからである。リンクリストを作っておく方法もあるが、剰余をとったりすると遅くなるし、そうでないと偏る。

ｎビットのうち、ｘビットがたっている状態で、$\log_2 n$　ビットの乱数を発生させてまだビットが立っていないところを選択するために必要な回数Ｃは次式で与えられる。

$$C = 1 * ((n - x) / n) + 2*((x / n) * ((n - x) / n) + 3 *((x / n)2 * ((n - x) / n) + ...$$
$$= ((n - x) / n) * \Sigma k(x/n)^{(k - 1)}$$
$$= ((n - x) / n) / ((1 - (x / n))^2)$$
$$= n / (n - x)$$

となる。

これより、例えば、理想的な乱数を生成した場合には、２５６ビットのうちから２９ビットたてるためには、８ビットの乱数を次の数だけ生成する。

$$C = 1 + 256 / 255 + 256 / 254 + ... + 256 / 228 = 30.714$$

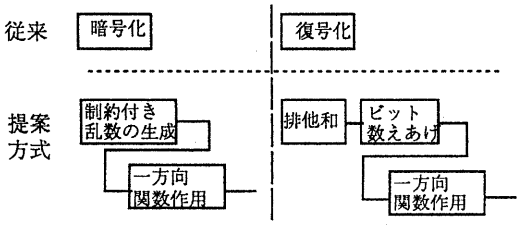
３２ビット乱数を８ビットずつ使うとすれば、３２ビット乱数を８回生成すればよい。なお、１９２ビットのうちから４０ビットなら同じく８ビットを44.781回必要となり、３２ビット乱数を１２回必要とする

とともに、余計な計算が必要になりあまり効率がよくない。

実際にシミュレーションした結果において乱数の平均生成回数を表５に示す。これはほぼ理論値に一致するものである。

表５　シミュレーションによる生成回数

	試行回数	乱数の平均生成回数
256ビット中29ビット	256	30.629
	1024	30.713
	4096	30.722
192ビット中40ビット	256	44.578
	1024	44.775
	4096	44.754



（A）クライアント側の操作　　（B）サーバ側の操作

図３　処理負荷の比較

図３に従来方式との処理負荷の比較を行う。ただし、この図では、分散環境におけるターンアラウンドタイムに影響する部分だけを記述している。すなわち、NPM では次の認証を行うまでの間に鍵を一方向性関数により更新しなくてはならない。このため、一方向性関数として従来の暗号関数を用いるとすれば、スループットとしては高速にすることはできない。ただし、一方向性関数は暗号よりは負荷の軽い演算を用いることも原理的には可能であると思われる。ただし、この部分に関する暗号的強度の検討はまだ行っていないので、ここでは通常の秘密鍵暗号を利用して更新することを前提とする。

乱数の生成はクライアント側でのみ必要であり、サーバ側では、排他論理和演算を行ってビットの数え上げを行うだけでよい。このため、サーバ側に負荷がかかるようなネットワークを介した分散処理環境に適している。排他論理和演算はほとんどのコンピュータにおいて１命令で実行でき、数えあげも１６ビット単位の数えあげ（シフトと論理積とテーブルアクセスが各１回、加算が１回）を１６回行えばよい。このような演算は従来の復号（これは秘密鍵方式では暗号化と同じコストがかかる）に比べて効率がよいと考えられる。

クライアント側では、２５６ビットに２９ビットの組み合わせを利用するとすると、制約付き乱数の生成では、３２ビット乱数の生成が８回、「ローテーションとビットマスクとシフトとビットマスクと比較と比較」（いずれも３２ビット）が３０回、加算が２９回必要となる。

サーバ側では、３２ビット排他和が８回、「ローテーションとビットマスクと配列アクセスと加算」が各１６回、比較が２回(mとn-mを認める場合）、初期設定が１回、必要となる。

筆者は［山上90］において１２８ビットの秘密鍵暗号演算の理論下限を示した。これは、１２８ビットの鍵と平文を３２ビットワードとして蓄積し、その全ての３２ビットワードとの間に演算による相関関係がなければどのビットもが結果に影響するような暗号にはならないという仮定によるものである。この仮定が正しいとするとソフトウェアで実現した場合の必要な３２ビット演算理論下限は３２回である。本提案方式では１２８ビットの理論下限に比べて約２倍の演算回数で実現できる。この理論下限が正しければ、いかなる１２８ビット暗号方式も３２ビット計算機で実現した場合、サーバ側の応答性能としては提案するNPMの２倍以上速くすることができない。

なお、本稿の考察では、３２ビット疑似乱数が、８ビット疑似乱数４個に利用できるという仮定を用いた。３１ビット疑似乱数random() を８ビットずつ３つの８ビット疑似乱数に用いる場合にこれが８ビットの疑似乱数として利用できることは実験的に示されている。しかし、この８ビット乱数をNPMのような方式に適用した場合、NPMが生成するビットパターンも十分ランダムなものになるかの検証は今後の課題である。

６．むすび

スケーラビリティにすぐれ、ビット長を自由に拡張することができ、分散処理環境においてサーバの負荷を軽くするような認

証方式を提案した。本稿では、主として現在、提案されている１２８ビット暗号による方式と比較するために２５６ビットを対象とした NPM を考察した。２５６ビットのうち２９ビットだけが一致する場合に認証を許可するような方式が１２８ビット暗号と同程度のしらみつぶし攻撃に対する強度を有することを示した。

提案方式の特徴を表６に示す。

表６　提案方式の特徴

項目	特性
スケーラビリティ	セキュリティ要求によって鍵の長さ、一致数を変えることが容易
負荷の非対称性	サーバ側の負荷が軽い
負荷のスケジューリング可能性	鍵の更新を遅延して実行することが可能

特にサーバ側においてはビットの数えあげはインプリメントも容易であり、必要な暗号強度に対して n やm を変更してもプログラムの変更はほとんど必要ない。また、認証が成功する毎に前回の認証から鍵のコピーによる不正アクセスがなかったことを証明することができる。このような暗号方式（鍵を毎回変える方式）については以前から提案されているが、本方式のように高速に比較を行うことが可能であれば、サーバ側の負荷を小さくすることができ、ネットワーク上で定期的に認証を行うことによって認証が安全であることを定期的に確認することことの実現性を高める利点がある。

NPM 方式自体はここで考察した制約付疑似乱数を利用した部分一致以外に、クライアント側でなんらかの予測不能で冗長性を持つ鍵を生成し、サーバ側がその予測不能な鍵を冗長性情報を利用して認証するようなアルゴリズム一般に利用可能である。ただし、計算機のソフトウェアで実現した場合、n ビットのうちm ビットだけ１がたっている制約付疑似乱数との排他的論理和を作る方法に比してより効率のいい方法を見いだすことは困難であると思われる。

謝辞

日頃、ご指導いただきますＮＴＴ通信網総合研究所ネットワークインテグレーション研究部木下研作部長、中田寿グループリーダ、ネットワークアーキテクチャ研究部清水明宏博士に厚く御礼申し上げます。

参考文献

[Knut81] Knuch, *The Art of Computer Programming, Vol.2 Seminumerical Algorithms*, Addison Wesley, 1981

[清水90] 清水、山上、: "A 32-bit microprocessor oriented fast enciphering algoritm', 電子情報通信学会論文誌 E73, (7), July 1990

[Sun] Sun Microsystems, UNIX Interface Reference Manual, 1986

[Wall90] Wall: *A Programming Language Perl*, O'Haily

[山上90] 山上、清水: "An Abstract Enciphering Machine Model", 電子情報通信学会論文誌　E73, (7), July 1990

[山上92] 山上「均衡制約付き疑似乱数の生成」情報処理学会アルゴリズム研究会、AL-25-4、January 1992