パケット往復時間の予測誤差を使った乱数生成法

白石善明[†] 孝富士武史[††] 森井昌克[††]

セキュリティプロトコルやセキュリティアプリケーションで使用されるセッション鍵を作成するためのマスタ鍵は予測不可能性，非再現性が必要となる．そのようなマスタ鍵を安全に生成する際には物理乱数を用いるのがよいとされているが，物理乱数を生成するためには一般に特別な装置を必要とするためにすべての端末にそのような装置を設置することは現実的ではない．本論文では一般的なコンピュータ環境においてネットワーク特性を利用することで物理乱数と同等の性質を有する乱数の生成方法を提案する．IP ネットワークのトラヒックをモデル化しようとしても急激なトラヒックの変化などのために厳密なモデル化は不可能である．そこで我々はクライアント/サーバ間のパケット往復時間をモデル化してパケットの往復時刻を予測しても，実際の往復時間と比較すると誤差が生じることに注目した．提案手法では，クライアント/サーバ間の往復時間を観測し，観測した過去の往復時間から次の時刻のパケット往復時間を AR モデルを使って予測し，その予測値と実際に計測した往復時間とを比較したときの誤差の一部を乱数として利用する．提案手法で生成した乱数を乱数検定ツールを使って検証したところ，物理乱数と同等の統計的性質を有することを確認した．

A Random Number Generation Using Prediction Error of Round Trip Time on IP Network

Yoshiaki Shiraishi,[†] Takeshi Kohfuji[††] and Masakatu Morii[††]

Master key used to generate session key in a certain security protocol and application needs not to be predictable and reproducible, and random number is generally used as the master key. It is said that the best way to generate such the master key securely is to use physical random number, but a certain special device is needed in the generation of physical random number. It is not, therefore, in practical to set such devices in all host. In this paper, we propose a random number generation using network property in general computer environment. The random number made by the proposed method has the property like physical random number. Even if the traffic of IP network is modeled in any way, it is impossible to be a strict model. Then, we note that the prediction error between actual round trip time and predicted one in a model of packet arrival process always occurs and the error is not predictable and reproducible. In our method, the round trip time between client and server is predicted by using AR model, then a part of the error between the predicted and actual round trip time is used as a random number. We evaluate the generated random number by using an examination tool of random number, and confirm that it has the most same statistical property as physical random number.

1. はじめに

保存している重要なデータや通信路を第三者から秘匿/保護する際には暗号アルゴリズムが使用される．その暗号アルゴリズムが使用される際に，暗号化鍵となる秘密鍵はランダムに作成し，予測されないようにする必要がある．ユーザを識別するためのパスワード認証の場合，ユーザ自身が覚えやすいように秘密鍵を作成するが，通常の暗号化アプリケーションにおける暗号化鍵や，SSL[4]，IPsec[6] のような暗号化通信用プロトコルにおけるセッション鍵のようにユーザがその生成に直接関与せずにソフトウェア側で生成する場合も多い．このような環境下では，乱数生成を行う端末がある一定の規則に基づいてセッション鍵のもととなるマスタ鍵を生成してしまうと，その規則を利用した攻撃者にセッションを乗っ取られたり，データを盗聴されたりしてしまう危険性がある．具体的には，なんらかのアルゴリズムに従ってソフトウェアでセッション鍵を生成する場合，マスタ鍵と呼ばれる入力ビット

† 近畿大学理工学部情報学科
Department of Informatics, Kinki University
†† 徳島大学工学部知能情報工学科
Department of Information Science and Intelligent Systems, The University of Tokushima

パターンを基にセッション鍵は計算されるため，そのマスタ鍵が予測されればセッション鍵も予測されてしまう．したがって，このようなセッション鍵を生成するためのマスタ鍵は予測不可能性と非再現性が必要であり，最も高い秘匿性を有する必要がある．

安全にマスタ鍵を作成するためには，物理乱数を使用するのがよいといわれている．しかしながら，物理乱数を生成するためには一般に特別な装置が必要であり，暗号化が必要な端末ごとにそのような装置を設置することは現実的に困難である．そこで，本論文では通常使用している端末上でソフトウェアを利用して真性乱数と同等の性質を有する乱数を生成する方法を提案する．

本論文で提案する乱数生成法は，IP ネットワークに接続された端末上でソフトウェアにより実現する．ソフトウェアだけで乱数を生成する方法はすでに阪口らによりパケットの到着時刻を用いた乱数生成法[10]として提案されている．彼らの方法はクライアント/サーバモデルにおける通信においてパケットの到着間隔がランダムであることを期待して，パケットの到着時刻を乱数源としている．しかしながら，現実には，たとえば文献 1) のように IP ネットワークのトラヒックのモデル化に関する報告も多くあることから前後のパケット間の到着時刻にはなんらかの相関があると考えられる．そこで，本論文では文献 10) における問題点を改良した，クライアント/サーバ間のパケットの往復時間（Round Trip Time: RTT）を AR（Auto Regressive，自己回帰）モデルでモデル化し，AR モデルによる RTT の予測値と実際の値との間の誤差を使って乱数生成する方法を提案し，生成された乱数の有効性を示す．

2. 乱数生成法

2.1 真性乱数と擬似乱数

乱数を大別すると真性乱数と擬似乱数がある．これらの違いは乱数の生成方法にある．前者はランダムに変化する物理現象を利用した物理的手段で発生させた乱数で，一般に物理乱数と呼ぶ．たとえば，抵抗体の熱雑音や放射性物質の崩壊，水晶発振器の不安定さを使ったものがある[2]．一方，後者の擬似乱数はなんらかの算術式によって発生させた乱数である算術乱数と，ランダム要素と呼ばれる発生数列の予測を困難にした乱数がある[3]．たとえば，算術乱数には線形合同法，暗号技術による乱数などがあり，またランダム要素にはキー入力の入力時間やマウス移動の軌跡，ハードディスクなどの媒体へのアクセス時間などのばらつきを使ったものがある[2]．

2.2 従来の問題点と要件

セキュリティプロトコルにおいて使用するマスタ鍵は，他の鍵と重複しないように，そして生成される鍵は予測不可能なように作成するのが望ましい．このようなマスタ鍵を生成するためには物理乱数を利用するのが望ましい．

物理乱数の生成法は古くから研究され，最近ではパソコンに内蔵できるように小型化を意図した研究がなされている[7),9]．そして，PCI カード上の乱数発生器を利用する方法[12] も実現されている．しかしながら，これらの方法で乱数を使用するためには新たなハードウェアを必要とすることが問題点である．その問題の解決を試みたものの 1 つに，物理乱数を生成するための装置をパソコンのマザーボード上に配置したものがある[5]．しかし，これも物理乱数発生器を搭載したマザーボードとそれを利用するためのファームウェアを必要とするために汎用性が劣ってしまうという問題がある．そこで特殊なハードウェアが不要で容易に生成できる擬似乱数の使用が望まれるが，暗号化アルゴリズムなどの算術処理を利用してマスタ鍵を生成する場合を考えると，その算術処理に入力するための秘密情報がさらに必要となり，算術乱数をマスタ鍵の生成に使うのは望ましくない．

また，SSL や IPsec などのセキュリティプロトコルは，パソコンなどのユーザが容易に構成を変更できる端末で利用されるばかりではなく，ルータやファイアウォールなどのネットワーク機器で使用されることも考慮しなければいけない．

以上の議論から，一般的なネットワーク環境においてマスタ鍵を生成する要件を整理すると次のようになる．

(1) 物理現象を発生させるための特殊なハードウェアを必要としない．
(2) 算術乱数を利用しない．
(3) パソコンやネットワーク機器の内部で乱数を発生することができる．

セキュリティプロトコルを使用するのはネットワークに接続された機器であるということを考えると，マスタ鍵の生成にはネットワークのなんらかの性質を利用することが可能である．そして，擬似乱数の中でもランダム要素をソフトウェアで生成し，マスタ鍵として利用することを考える．そこで本論文では，ネットワークの性質を使ってランダム要素を生成する方法を提案する．

2.3 パケット到着時刻を使った乱数生成[10)]

IP ネットワーク上でソフトウェアだけを使って乱数を生成する方法がすでに阪口らにより提案されている[10)]．その阪口らによるパケット到着時刻を使った乱数生成法は，クライアント/サーバモデルにおいて，クライアントからサーバにデータを要求し，クライアントがサーバから受信したデータの到着時刻を乱数源とする．到着時刻の取得は gettimeofday 関数を使ってマイクロ秒の精度で測定し，その得られたデータから乱数となるビット列を抽出する．文献 10) ではビット列の抽出に 3 つの方法が提案されているが，ここではその中で最も優良な乱数を取得することができるとされている手法について簡単に説明する．以下ではこの手法を従来手法と呼ぶことにする．

クライアントからサーバへパケットを送信し，サーバからの返信パケットの到着時刻で i 番目に計測したものを T_i としたときの，その乱数生成法は以下のようなものである．

(1) $i = 0$ とする．
(2) T_{i-1}，T_i を計測する．
(3) T_{i-1}，T_i を比較し，変化したビット位置の T_i のビットを取り出して連結したものを系列 B_i とし，これを乱数列として利用する．
(4) $i = i + 2$ として，必要なビット数が得られるまで (2) から繰り返し．

この方法は計測した到着時刻を連続して順次利用するのではなく，クライアント/サーバ間でパケットを 2 往復させるごとに 1 つの乱数列を採取するという方法である．また，手順 (4) の具体例をあげると次のようになる．

T_{i-1}	1	0	1	0	1	0
T_i	1	1	0	0	0	1
B_i		1	0		0	1

この例では B_i として 4 ビットが得られる．このようにして得られた B_i を連結し，乱数系列とする．ここで，ビット列 T_{i-1}，T_i におけるシンボル 0 の出現確率を p とすると，シンボル 1 の出現確率は $1-p$ となる．ある時点 i におけるビット列を $A_i = (A_i[0], A_i[1], \ldots)$，$A_i \in \{0, 1\}$ と定義する．従来手法では，$(T_{i-1}[j], T_i[j])$ が $(0, 0)$，$(1, 1)$ の場合にはビットを採取せず，$(0, 1)$，$(1, 0)$ の場合に採取するので，$B_i[j]$ に 0 が出現する確率は $p(1-p)/2p(1-p) = 1/2$ となる．同様に $B_i[j]$ に 1 が出現する確率は 1/2 となることから，従来手法では理論的には 0，1 の等頻度性は保証される．

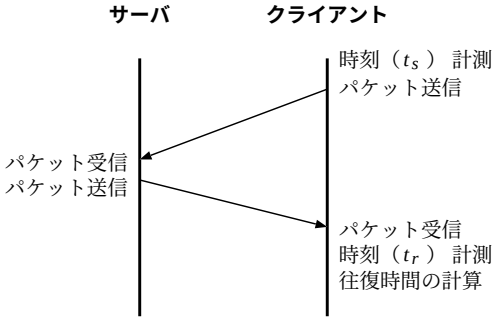

図 1 クライアントとサーバの処理
Fig. 1 Process in client and server.

3. 提案手法

従来手法では，ネットワークを介したあるトラヒックフローにおけるパケットの到着時間がランダムと仮定した場合に有効な手法であるといえる．しかしながら，IP ネットワークにおいても到着過程はモデル化できる[1)] ことから，前後のパケット間の到着時刻には時系列で見ると相関があると考えられ，必ずしも従来手法が有効ではない場合がある．そこで，本章ではモデル化を利用することで，前後のパケットとの到着時刻の相関を抑え，ネットワークトラヒックによる到着時刻のゆらぎを利用する改良手法を提案する．

クライアント/サーバモデルにおける通信において，クライアントから送信したパケットをサーバが受信し，それに対するサーバからの返答をクライアントが受信するまでの時間を RTT という．その RTT をモデル化し，次のパケットの RTT を予測してもネットワーク上のバースト的なトラヒックやトラヒックフローの増加などのノイズにより，その予測値と実測値には誤差が生じる．本論文で提案する乱数生成法は，そのようなネットワークトラヒックの予測不可能性を利用する．

以下では，RTT の測定方法，AR モデルによる RTT の予測，および乱数取得方法について述べる．

3.1 RTT の測定

提案する乱数生成法ではクライアント/サーバ間のパケットの RTT を利用する．クライアントとサーバで socket を使って TCP コネクションを張り，クライアントから IP データグラムを送信して，サーバから IP データグラムを受信するまでの時間を RTT とする．**図 1** に示すように，クライアント/サーバモデルにおけるクライアントとサーバの処理は以下のとおりである．

クライアント

(1) パケットの送信前に時刻（t_s）を測定．

(2) パケットをサーバに送信.
(3) パケットの受信後に時刻（t_r）を測定.
(4) RTT（$t_r - t_s$）を計算.

サーバ

(1) クライアントからのパケットを受信.
(2) パケットをクライアントへ送信.

本研究では 4.1 節に示すように，クライアントとサーバでは OS に UNIX を使用しており，パケットの送信および受信時刻の計測には gettimeofday システムコールを利用している．この関数では，秒の表現に long 型の変数（tv_sec），マイクロ秒の表現に long 型の変数（tv_usec）という構造体を利用してマイクロ秒の精度で時刻を計測することができる．

3.2 AR モデルによる RTT の予測

RTT の予測は過去の RTT の観測値 $(x_{t-1}, x_{t-2}, \cdots, x_{t-k})$ を用いてモデル化を行い予測する．

モデルの定式化には AR モデル（自己回帰モデル）[11] を利用する．AR モデルは離散系の時系列モデルとして時系列解析の代表的なモデルとされている．AR モデルは時系列の現時点における値が，自らの過去の時点での値の線形結合によって推定されるというモデルである．対象とする時系列を $x_t, t = 1, 2, \ldots, n$ とすると AR モデルは次の式 (1) で表される．

$$x_t + \sum_{i=1}^{k} a_i x_{t-i} = e_t. \tag{1}$$

式 (1) を k 次の AR モデルという．x_{t-i} は x_t より i 個過去の時点の測定値である．e_t は t の時点におけるモデル化の誤差であり，この値が 0 であれば誤差がないことになる．$a_i, i = 1, 2, \ldots, k$ は自己回帰係数である．

例として 1 次の AR モデルを**図 2** に示す．図中の z^{-1} は遅延演算子である．すなわち図 2 において，x_t は 1 次遅れを持つフィードバック制御系の出力ということになる．このことから，k 次の AR モデルは k 次までの遅れを持つフィードバック系でモデル化されるということになる．AR モデルを推定するためには，式 (1) の自己回帰係数 a_i を推定する必要がある．本研究では a_i を推定するために次のような Yule-Walker の推定[11] を用いている．まず，式 (1) の両辺に x_{t-l} をかけて平均をとると次のような関係式が成り立つことが知られている．

$$r_l + \sum_{i=1}^{k} a_i r_{l-i} = 0. \tag{2}$$

ここで，r_l は x_t の期待値の関数であり，x_t，x_{t-l} の積の平均で表される．この $(r_0, r_1, \ldots, r_k)$ はあらかじめ計算しておく．さらに，式 (2) において，$l = 1, 2, \ldots, k$ を考えると以下の方程式が導出されることが知られている．

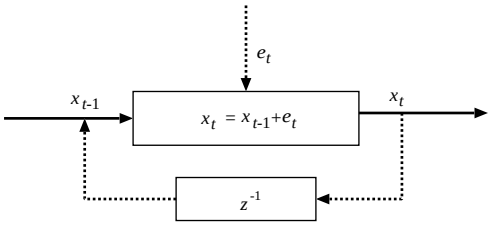

図 2 1 次の AR モデル
Fig. 2 AR model with first order.

$$\begin{cases} r_1 + a_1 r_0 + a_2 r_1 + \cdots + a_k r_{k-1} & = 0, \\ r_2 + a_0 r_1 + a_2 r_0 + \cdots + a_k r_{k-2} & = 0, \\ \vdots & \\ r_k + a_1 r_{k-1} + a_2 r_{k-2} + \cdots + a_k r_0 & = 0. \end{cases} \tag{3}$$

この k 次連立方程式を Yule-Walker の方程式という．自己回帰係数 a_i は式 (3) を解いて求めるのが一般的である．この a_i よりモデルの推定を行い，時点 i における RTT の予測を行う．

3.3 RTT の予測誤差を使った乱数生成

提案手法では RTT の予測値と観測値の誤差を乱数として利用する．実際に観測した RTT を x_i，k 次の AR モデルを使用した RTT の予測値を x'_i，予測誤差を d_i とすると d_i を次のような式 (4) で表す．

$$d_i = |x_i - x'_i|. \tag{4}$$

これらの x_i，x'_i，d_i を使った乱数は以下のようになる（**図 3**）．この手順は乱数を生成するクライアント側の処理である．

(1) $i = 0$ とする．
(2) RTT$(x_{i-k}, x_{i-k+1}, \ldots, x_{i-1})$ を用意する．
(3) 過去の RTT$(x_{i-k}, x_{i-k+1}, \ldots, x_{i-1})$ より，時点 i における k 次の AR モデルを推定する．
(4) 推定したモデルを使って時点 i における RTT の予測値 x'_i を計算する．
(5) 時点 i におけるパケット送信時刻 t_{s_i} を測定して，パケットを送信する．
(6) 時点 i における到着時刻 t_{r_i} を計測し，時点 i における RTT $x_i = t_{r_i} - t_{s_i}$ を計算する．
(7) 時点 i における予測誤差 $d_i = |x_i - x'_i|$ を計算し，その d_i を二進数表現した最下位 1 ビットを乱数として利用する．
(8) $i = i + 1$ として，必要なビット数が得られる

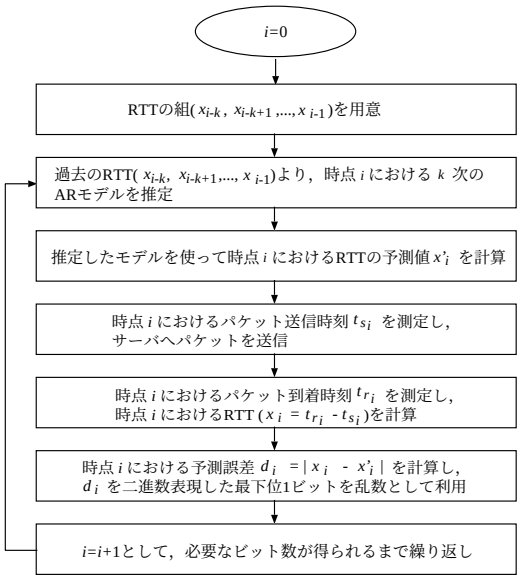

図 3 乱数生成の手順
Fig. 3 Flow of random number generation.

まで (3) から繰り返し．

4. 実験と考察

4.1 実験環境

実験は 100BASE-TX で接続された LAN 上の 2 台の PC 間で行った．それらの PC は以下のスペックである．

CPU	：PentiumIII 1 GHz
Memory	：256 MB
OS	：RedHat Linux 7.2

クライアント，サーバで使用したプログラムは C 言語（gcc）で作成した．

4.2 乱数の検定

3.3 節に示した乱数生成法では，予測誤差 d_i の最下位 1 ビットを利用して乱数を生成する．予測誤差は最下位から数ビット生じることが予想されるため，ここでは最下位 1 ビットを利用することの妥当性について検証する．また，AR モデルの次数 k を変更することで乱数の性質が変わることも予想されることから，利用する次数についても検討する．

一般に，不規則性，シンボル出現の等頻度性，前後のシンボル間の無相関性，非周期性などの真性乱数の条件を生成した乱数が満足するかどうかを検証するには統計的な性質を調べる．たとえば，文献 7)，9) で提案されている物理乱数生成法においても，生成した乱数についていくつかの統計的な性質を調べている．

表 1 乱数列として利用可能なビット数の検証
Table 1 Verification on the number of available bit as random sequence.

利用ビット数	1 ビット	2 ビット
合格した乱数系列数	20/20	18/20

これらの文献で示されている検定方法を含んだ一般的な物理乱数検定ツールとして NIST 検定ツール[8] が知られている．そこで，得られた予測誤差 d_i のうちで最下位から何ビットを乱数として利用できるかを検定ツールを使って検証する．

NIST 検定ツールには 16 種のテスト項目がある．この検定ツールに入力する乱数列の長さは 1 回の検定あたりそれぞれ約 10 M ビットとする．NIST 検定ツールの各テスト項目では P-value と呼ばれる値を使って，入力されたビット列が乱数であるかどうかを判断する．具体的にはそのビット列が乱数であるかどうかという仮説の有意水準を α とし，P-value が α よりも小さければ乱数ではないと判断する．このときの P-value は α を閾値とするようにツール内で正規化されている．以下では，$\alpha = 0.01$ を閾値とし，P-value < 0.01 となった場合にはそのテスト項目は不合格と見なすことにする．これは，テストに合格した場合の入力されたビット列は 99%の確率で乱数であるということである．そして，入力する乱数列を 20 通り用意し，それぞれの乱数列に対して NIST 検定ツールを実行する．各々の乱数列に対して NIST 検定ツールを実行し，16 種類のテストのうちで 1 つでも不合格となった項目があった場合には，その系列は統計的性質を満たさないと見なし不合格とする．

まず，予測誤差 d_i のうちで何ビットを乱数として利用できるかを検証するために，3.3 節で生成した各時刻における予測誤差のビット列で，最下位 1 ビットと最下位から 2 ビットを取り出して連結させた乱数列を各 20 通り用意し，NIST 検定ツールを実行した．このときの AR モデルの次数は 10 次である．実験結果を**表 1** に示す．最下位 1 ビットを利用した乱数列では，20 通りのすべての系列について合格となった．しかし，最下位から 2 ビットを利用した乱数列では 20 通りのうち 2 通りが不合格となった．不合格となった項目は，The Frequency（Monobit）Test，The Runs Test の 2 つであった．

乱数列の性質は予測誤差 d_i の利用ビット数により変化することが分かった．予備的な実験により，LAN 内のクライアント/サーバモデルにおいて採取した予測誤差の個数のうちで，予測誤差の有効ビット数が最下位から 1〜8 ビットのものが 90%以上になるという

表 2 AR モデルの次数の検証
Table 2 Verification on the degree of AR model.

AR モデルの次数 (k)	結果
5	不合格
10	合格
15	合格
20	合格
30	合格
40	合格
50	不合格

ことを確認している．先ほどの実験では最下位から 2 ビットまでを乱数列として使ったときの性質を検証したが，3 ビット以上を使うことが可能かを確認するために同様の実験を行った．まず，d_i の最下位 3 ビットを連結した乱数列を 5 通り用意し，NIST 検定ツールでテストしたが，すべての系列が不合格になった．このことから，上位のビットを使うほど乱数列の性質が悪くなると予想される．したがって，3.3 節の (7) で記述したように，最下位 1 ビットを乱数列として利用するのがよいと考えられる．

次に，乱数の性質に影響すると考えられるもう 1 つのパラメータである AR モデルの次数について実験を行い考察する．実験方法は AR モデルの次数を変えて予測誤差の最下位 1 ビットを連結した乱数列を採取し，各次数の 20 通りの系列について NIST 検定ツールでテストを行った．AR モデルの次数は $k = 5, 10, 15, 20, 30, 40, 50$ を使用した．検定結果を**表 2** に示す．次数が 5 のときには 20 通りの乱数列のうち 1 つがテストに合格しなかったために不合格となった．また，次数が 50 の場合にも不合格となった乱数列が 2 つあったために不合格とした．このことから，AR モデルに使用する次数が低いと RTT の予測値の精度が悪くなってしまうために統計的な性質が悪くなること，そして逆に次数を高くとると予測値の精度が良すぎるために統計的性質が悪くなると考えられる．

4.3 乱数の生成速度

提案手法では，乱数の 1 ビットを生成するために，毎回 k 次の AR モデルの係数 a_i を導出している．この a_i を求めるためには k 次連立方程式を解かなければならない．したがって，クライアント/サーバ間の RTT がほぼ一定という条件の下では，提案手法の中で最も乱数生成速度に影響するのは AR モデルの次数 k ということになる．そこで，次数 k を変化させたときの乱数生成速度を調べる．ここでは，$k = 5, 10, 15, 20, 30, 40, 50$ の各次数に対して 1 M ビットの乱数列を生成するのにかかる時間を計測し，1 秒あたりに何ビット生成できるかを計算した．これを各次数について 20 回試行したときの平均値が**図 4** である．次数 k が大きくなればなるほど生成速度も遅くなっていることが分かる．4.2 節の実験結果から AR モデルの次数 k により乱数の性質が変化することを考慮すると，$k = 5$ のときには約 1,225 bps で乱数を取得することができるが，その性質が悪いために乱数として使えないということになる．また，次数 k を大きくしすぎると，その乱数の性質が悪くなることと，生成速度が遅くなることから実用的ではないといえる．このことから，生成速度が 1,000 bps 程度で性質の良い実用的な乱数を生成できる次数は $10 < k < 20$ といえる．

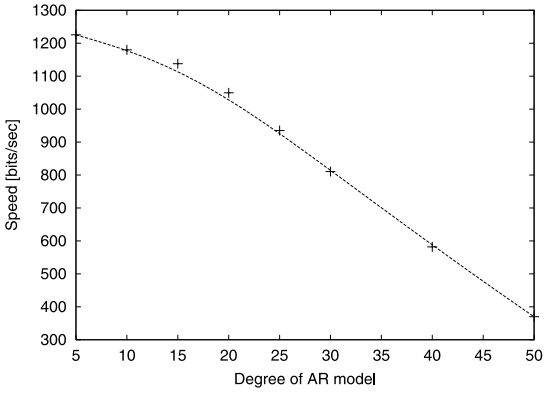

図 4 乱数の生成速度
Fig. 4 Speed of random number generation.

4.4 従来手法の追実験

2.3 節の従来手法で生成した乱数を，実験環境は 4.1 節と同じものを使用し，4.2 節と同様に 10 M ビットの系列を 20 通り用意して NIST 検定ツールを使ってテストを行った．その結果，20 通りすべての系列が不合格となった．B_i は i について平均して約 3 ビット程度採取できる[10]．我々の提案手法でも上位のビットを含めてしまうと検定に不合格になる傾向が見られたので，B_i の最下位 1 ビットだけを乱数として使用するテストを行ってみた．しかし，こちらも 20 通りすべての系列が不合格となった．到着過程がランダムになることが期待されるネットワークにおいては良好な乱数が得られるかもしれないが，実験環境は閉じたネットワークであり，到着過程がランダムにならなかったためにこのような結果になったものと考えられる．したがって，パケットの到着時刻というようなネットワーク上のある事象だけを使って乱数を生成するよりも，ある事象についてまずモデル化を行い，そのモデルを使って予想したものと実際の事象との誤差を利用するのが乱数の生成に有効であると考えられる．

5. まとめ

本論文ではネットワークトラヒックの予測不可能性を利用したランダム要素に分類される乱数を生成する方法を提案した．自律分散型ネットワークである IP ネットワークでは，トラヒックの変動を厳密に予測することは困難であるという点に着目し，クライアント/サーバモデルにおけるパケットの往復時間のモデル化を行い，次の時刻の往復時間の予測値を計算し，実際の往復時間との誤差を乱数として利用した．そして，提案手法で生成した乱数を乱数検定ツールを使って検証した結果，統計的に性質の良い乱数を生成できることを示した．ネットワークトラヒックの変動は物理現象ではないので，提案手法で生成した乱数は物理乱数ということはできない．しかし，様々な端末やアプリケーションが利用されたときのネットワークトラヒックの再現は現実的には不可能であるので，物理乱数と同等の非予測性，非再現性を有する乱数の生成を簡易なソフトウェアで実現できることを示したことは実用的にも意味があると考えられる．提案手法では IP ネットワークにおける往復時間の予測誤差を利用したが，マウスによる移動の軌跡やハードディスクなどの媒体へのアクセス時間のようにほぼ規則的な事象に適度なばらつきがあるような他の環境においてもモデル化を行ったうえで予測誤差を利用する手法が乱数生成に有効な手法であると期待される．

また，提案手法では AR モデルの次数と予測誤差の利用ビット数により乱数の統計的性質が変化することを乱数検定ツールを使って確認するとともに，LAN 内のクライアント/サーバモデルにおいてそれらの値について実験をしたところ，適切な値が存在することを示した．

今回の実験は LAN 上の遅延が比較的短い環境でしか行っていない．しかし，遅延が大きいネットワークなど環境の変化があったとしても AR モデルの次数を変更することで性質の良い乱数が生成できると予想している．そこで，今後の課題として，インターネットを介した外部ネットワークにサーバ位置を変更した場合の乱数の性質を調べることがあげられる．

参考文献

1) Corradi, M., Garroppo, R.G., Giordano, S. and Pagano, M.: Analysis of f-ARIMA processes in the modelling of broadband traffic, *IEEE ICC2001*, pp.964–968 (June 2001).
2) 電子商取引実証推進協議会：暗号利用技術ハンドブック (1998).
3) 電子商取引実証推進協議会：暗号利用技術ハンドブック（第 2 版）(2000).
4) Frier, A., Karlton, P. and Kocher, P.: *The SSL 3.0 Protocol*, Netscape Communications Corp., (Nov. 18, 1996).
5) Intel: Random Number Generator. http://developer.intel.com/design/security/rng/rng.htm
6) Kent, S. and Atkinson, R.: Security Architecture for the Internet Protocol, IETF, RFC2401 (Nov. 1998).
7) 岸本俊祐，福江万寿夫：ダイオードノイズを利用した物理乱数の発生とその評価，電子情報通信学会論文誌 A，Vol.J82-A, No.11, pp.1704–1709 (1999).
8) NIST: http://csrc.nist.gov/rng.
9) 斉藤義明，堀　潤一，西村浩志，木竜　徹：可変容量パラメトロンによる物理的乱数発生法，電子情報通信学会論文誌 A，Vol.J85-A, No.2, pp.181–188 (2002).
10) 阪口康弘，岡本栄司：IP パケットの到着時刻を用いた乱数生成，1999 年暗号と情報セキュリティシンポジウム講演論文集（SCIS'99），pp.491–496 (1999).
11) 杉原敏夫：適応的モデルによる経済時系列分析，工学図書 (1996).
12) 東芝：ランダムマスター. http://www.toshiba.co.jp/product/abwr/random/

付　録

A.1 乱数検定ツール

乱数検定ツールは乱数の安全性のテストを行う．代表的な乱数検定ツールとして NIST の検定ツール[8] が知られている．NIST 検定ツールは物理乱数および擬似乱数生成器からの出力データの乱数性のテストを行うツールであり，以下に示す 16 項目からのテストからなる統計パッケージである．

(1) The Frequency（Monobit）Test
(2) Frequency Test within Block
(3) The Runs Test
(4) Test for the Longest-Run-of-Ones in a Block
(5) Random Binary Matrix Rank Test
(6) Discrete Fourier Transform Test
(7) Non-overlapping Template Matching Test
(8) Overlapping Template Matching Test
(9) Maurer's Universal Statistical Test
(10) Lempel-Ziv Complexity Test
(11) Linear Complexity Test

表 3 NIST 検定ツールの実行パラメータ
Table 3 Parameter of our evaluation in NIST tool.

Data Length	1,000,000
Block Frequency Test block length	100
NonOverlapping Template Test Block Length	10
Overlapping Template Test Block Length	10
Universal Test Block Length	7
Universal Test Number Of Initialization Steps	1,280
Approximate Entropy Test Block Length	2
Serial Test Block Length	2
Linear Complexity Test Subsequence Length	500
bitstreams should be generated	10

(12) Serial Test
(13) Approximate Entropy Test
(14) Cumulative Sum Test
(15) Random Excursion Test
(16) Random Excursions Variant Test

このツールではバイナリのビット列を書き込んだ乱数のファイルに対して様々な統計量を計算する．そして，その操作を繰り返すことで，その乱数列の統計量の分布が理想的な分布に近いかどうかの検定を行うというものである．

本研究で検定する際に利用した NIST のパラメータを**表 3** に示す．各種テストを行うことによって出力された P-value が P-value > 0.01 を満たす場合に，そのテスト項目にパスしたと見なしている．

(平成 14 年 11 月 29 日受付)
(平成 15 年 6 月 3 日採録)

白石 善明（正会員）

1973 年生．1995 年愛媛大学工学部情報工学科卒業．2000 年徳島大学大学院工学研究科博士後期課程修了．博士（工学）．2002 年近畿大学理工学部講師，現在に至る．情報セキュリティ，コンピュータネットワーク等の研究，教育に従事．2002 年電子情報通信学会オフィスシステム研究賞受賞．電子情報通信学会，情報理論とその応用学会，IEEE，ACM 各会員．

孝富士武史

1979 年生．2001 年徳島大学工学部知能情報工学科卒業．2003 年徳島大学大学院工学研究科博士前期課程修了．

森井 昌克（正会員）

1958 年生．1989 年大阪大学大学院工学研究科通信工学専攻博士課程修了．工学博士．同年，京都工芸繊維大学工芸学部助手．1990 年愛媛大学工学部講師．1992 年同助教授．1995 年徳島大学工学部教授，現在に至る．符号理論，暗号理論，情報セキュリティ，コンピュータネットワーク等に関する研究，教育に従事．電子情報通信学会，情報理論とその応用学会，IEEE 各会員．