暗号通信におけるリクエスト予測を用いたFPGA再構成オーバヘッドの低減手法

丹羽雄平[†] 前田敦司[†] 山口喜教[†]

クライアント・サーバ型の暗号通信において，暗号処理の部分を汎用の FPGA を動的に書き換えて実行するような，リコンフィギャラブルなシステムモデルを作成した．このようなシステムにおいて，性能の向上を妨げる要因となりうる FPGA 再構成オーバヘッドを低減する方法として，処理のオーバラップと過去の履歴からの近い将来のリクエストにおける暗号方式の予測を行い，その有効性を検証した．シミュレーション実験の結果，処理のオーバラップにより数十倍以上の，リクエストの予測によりさらに平均 70%，最大 250%以上の性能向上が可能であることが分かった．

A Method of Reducing Reconfiguration Overhead of an FPGA-based Encryption Communication System by Predicting the Use of Encryption Algorithm

Yuhei Niwa,[†] Atushi Maeda[†] and Yoshinori Yamaguchi[†]

In this paper, we proposed the reconfigurable system model which used FPGA dynamic reconfiguration to encrypt the data in the server-client encryption communication. In such a system, the factor which prevents an improvement of performance is FPGA reconfiguration overhead. Then, we take the way to reduce such overhead, predict the encryption algorithm used in the near future requests based on history of requests received so far and, overlap the software processing and FPGA reconfiguration. In this paper, we verify the effectiveness of these way to reduce FPGA reconfiguration overhead by simulation. Result, tens of performance times or more improvement by overlapping, in addition, performance improvement of average 70% and maximum 250% by request prediction.

1. はじめに

FPGA（Field Programmable Gate Array）は，ハードウェアの高速性とソフトウェアの柔軟性を兼ね備えたデバイスである．FPGA のように，回路を変更できるデバイスを用いることで，計算機で行われる様々な処理のうち，ある特定の計算処理を汎用プロセッサよりも高速に実行するシステムを，リコンフィギュラブルシステム（Reconfigurable System）[1] という．

リコンフィギュラブルシステムはハードウェアである FPGA を用いているため，暗号処理，パターンマッチング，GA，ニューラルネットワークなどといった処理を，汎用プロセッサよりも高速に実行できる可能性を持っている．

また，FPGA の動的再構成を行うことで，システムの停止・再起動を行わずに FPGA を必要に応じて別の回路に書き換えることが可能である．

このようなリコンフィギュラブルシステムを暗号通信に用いた場合，通信プロセスごとに異なる暗号処理を要求されることが考えられ，そのたびに FPGA の再構成が発生する可能性がある．このとき，ハードウェア化による高速化以上に再構成のオーバヘッドが大きければ，かえって処理性能が低下することになる．

本研究では，Web サーバへの暗号通信リクエストに対して，FPGA を搭載したリコンフィギャラブルシステムのモデル[2] を与え，このモデルにおける FPGA 再構成オーバヘッド低減の手法について述べ，その有効性を検証する．

本稿では，ソフトウェア処理とのオーバラップによる再構成オーバヘッドの隠蔽と，過去の通信記録を基に，近い将来の通信を予測し，FPGA 再構成のタイミングを決定することで再構成オーバヘッドの低減を目指すという手法をとった．

† 筑波大学大学院システム情報工学研究科
System and Information Engineering, University of Tsukuba

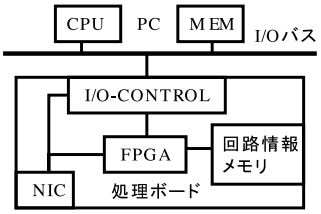

図 1 システムハードウェア構成
Fig. 1 Arcnitecture of this system.

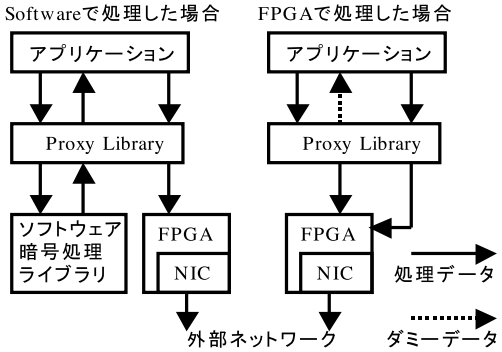

図 2 システムデータフロー
Fig. 2 Data flow of this system.

2. 暗号通信モデル

本研究では，クライアントからのリクエストに対し，サーバが持っているデータを暗号化してクライアントに返すという，クライアント・サーバ型の暗号通信を対象とする．Web サーバに送られて来るクライアントからのリクエストが，すべてセキュアな暗号通信リクエストであるような場合を考える．

クライアントからのリクエストを受けたサーバは，自身の持っているファイル（データ）を暗号化し，それをクライアントに送信する．このようなサーバの暗号処理の負荷を軽減し，それ以外の処理にその性能を生かすことができるようにするため，本システムモデルを適用することを目的とする．

2.1 システム構成

本モデルは，PC と，その PC に I/O バスを通じて接続された処理ボードから構成されており，処理ボードには I/O コントローラ，暗号処理用 FPGA，NIC，回路データメモリが搭載されている（**図 1**）．システムの起動時に，必要となる暗号処理回路データをあらかじめ回路データメモリに記憶させておくことで，I/O バスは処理対象データと制御信号の転送にのみ使用できるようになり，効率的な処理が可能になる．また，処理ボード上に NIC を搭載することで，暗号化の結果を再び PC に転送せずにそのまま NIC からネットワーク上に流せるような構成になっている．

2.2 データフロー

通常の暗号通信アプリケーションでは，リクエストされた暗号方式でデータを暗号化するために，対応するソフトウェア暗号処理モジュールを呼び出す．このモジュールを用いてリクエストされたデータを暗号化し NIC から送信する．

一方，本システムモデルでは，暗号通信アプリケーションにおけるソフトウェア暗号処理モジュール呼び出し部分と NIC にパケットを送信する部分とを，Proxy Library の呼び出しに変更する．呼び出された Proxy Library は，従来どおりのソフトウェア暗号処理モジュールを呼び出して処理を実行させるか，それともハードウェア暗号処理モジュールで処理を実行するかを決定し，さらには FPGA の再構成を行うかどうかの判断・指示も行う．

あるリクエストの処理をソフトウェアで行うと決定したならば，従来どおりのソフトウェア暗号処理モジュールを用いてデータを暗号化し，結果を返す．データを受け取ったアプリケーションは従来どおりそのデータをパケットにして，置き換えられた Proxy Library 経由で処理ボードに送り，そのまま NIC からネットワークに送信することになる．

Proxy Library が暗号化処理を FPGA で行うと決定したならば，アプリケーションに暗号化していないデータを返す．そのデータを受け取ったアプリケーションはそのままデータをパケットにして，Proxy Library 経由で処理ボードに送る．このとき Proxy Library は，このパケットは未暗号化データなので，NIC から送信する前に暗号化するという命令を処理ボードに送る．処理ボードではその命令に従って受け取ったパケットの中身を暗号化し，NIC からネットワークに送信する（**図 2**）．

2.3 FPGA の選択

本システムでは，処理を行うためのハードウェアデバイスとして FPGA を利用する．また，システム稼動中の暗号処理方式の変更に対応するため，FPGA の動的再構成を行う必要がある．

近年，動的再構成可能な特殊なデバイスが登場してきている．これらのデバイスは 1 クロック，あるいは数 μ 秒単位で動的に回路の再構成が可能であるが，汎用 FPGA に比べて小さな回路規模となっている．そのため，ある 1 つの大きな暗号処理回路の内部をさらに小さな部品に分けて，その部品を動的に再構成することで大きな暗号処理回路を少ないチップ面積で実現

することが可能となっている．

本システムでは，大きな暗号処理回路全体を動的に再構成する必要があるため，実装可能な回路規模の大きい汎用 FPGA（Xilinx 社製 xc2v8000 デバイス）を用いることとする．また，このデバイスのコンフィギュレーションは，コンフィギュレーションクロックと呼ばれる専用クロックに同期して，専用バスを通じて 8 bit ずつ回路構成データが回路構成情報メモリからデバイスに読み込まれることで完了する[3]．xc2v8000 の回路構成データのサイズは 29,063,072 bit であり，コンフィギュレーションクロックは最大 66 MHz での動作が可能である．このことより，xc2v8000 デバイスのコンフィギュレーションには，約 60 ms の時間が必要である．

3. 暗号処理性能

予備実験として，様々な暗号方式を FPGA に実装して実行した場合の性能と，ソフトウェアで実装した場合の性能の測定を行った．

測定の対象としたのは，AES 選考[4),5)] の対象となった暗号方式と CRYPTREC[6)] にあげられている暗号方式，さらに GNU の暗号ライブラリである libgcrypt[7)] に含まれる暗号方式などの中から 10 種類の暗号方式を選び出した．以下にその暗号方式の一覧を示す．
RC6-ECB，Camellia，DES-ECB，DES-CBC，
AES- ECB，AES- CBC，3DES-ECB，3DES-CBC，
SERPENT128-ECB，TWOFISH128-ECB

3.1 ソフトウェア処理性能

RC6-ECB のソフトウェア処理性能については，参考文献 8) より，Pentium III600 MHz，Borland C++ 5.01 で 1 block128 bit を平均 620 clocks で暗号化という報告から，Camellia[9)] についても同様に，参考文献 6) より，Pentium III650 MHz，VisualC++ Ver. 6.0 SP3 で 1 block128 bit を平均 487 clocks で暗号化との報告を参考に，**表 1** の環境下での処理性能を暗号化に要したクロックサイクル数より逆算して求めた．

また，それ以外の暗号方式については GNU の暗号ライブラリである libgcrypt のベンチマーク実行結果の数値を基に，その処理性能を決定した．ベンチマーク環境は表 1 のとおりである．

以上の文献，ベンチマーク結果から，上記 10 種類の暗号方式のソフトウェア処理性能は，表 1 の計算機環境において，**表 2** の値とした．

3.2 FPGA 処理性能

AES-ECB，3DES-ECB，3DES-CBC，DES-ECB，DES-CBC の FPGA 実装時の処理性能については，

表 1 ソフトウェアベンチマーク環境
Table 1 Parameter of software benchmark.

CPU	Pentium4 2.2 GHz
OS	Windows2000 SP4
Compilar	GCC-O3
言語	C 言語

表 2 各暗号方式の処理性能（Mbps）
Table 2 Throughput of each encryption (Mbps).

暗号方式	Software	FPGA	Slice 使用率
RC6-ECB	454.0	2,398.0	88/*
Camellia	478.0	6,750.0	*/50
DES-ECB	129.0	4,571.0	*/13
DES-CBC	104.6	333.3	*/1
AES-ECB	266.7	8,880/0	89.4/48
AES-CBC	170.2	300.1	43/*
3DES-ECB	75.5	4,000.0	*/14
3DES-CBC	64.0	95.2	*/1
SERPENT	842.8	4,860.0	73/*
TWOFISH	102.6	1,585.0	76/*

OPENCORES[10)] より提供されているフリーの暗号処理 IP コアの論理合成・配置配線を行い，その性能を測定した．論理合成配置・配線に用いた論理合成ツールは Xilinx 社の ISE6.2i，ターゲットデバイスは Xilinx 社の Vertex2-xc2v8000，スピードグレードは-4 とした．

Camelliaについては参考文献 6) より，1 block128 bit の暗号化に 20 clocks を要するとの報告を参考に，それ以外の暗号方式については参考文献 11)，12) において，xcv1000 デバイスでの評価結果より，RC6 が 1 block128 bit を 2 clocks で 37.5 MHz，AES-CBC が 1 block128 bit を 6 clocks で 14.1 MHz，SERPENT が 1 block128 bit を 1 clocks で 38.0 MHz，TWOFISH が 1 block128 bit を 2 clocks で 24.8 MHz との報告を参考に，それぞれの性能を計算した．その結果，上記 10 種類の暗号方式の FPGA 処理性能は，Vertex2-xc2v8000 デバイスをターゲットとして表 2 の値とした．

また，表 2 の回路規模に関する検証結果は，（xcv1000 の Slice 使用率/xcv8000 の Slice 使用率）という表記になっている．ただし，Camellia は参考文献 6) より，VertexE デバイスでの実装報告がなされており，ほぼ 2 倍の回路規模を持つ xcv8000 ならば使用率 50%に収まると判断した．なお，*印は未検証である．最も回路規模の大きい AES-ECB であっても，xcv8000 では 48%の使用率なので，1 枚の FPGA には最低 2 つ（2 種類）の暗号化回路が同時に搭載可能であると判断した．

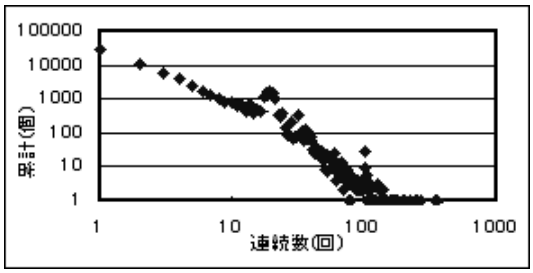

図 3 同一クライアントリクエスト連続回数累計
Fig. 3 Total of continuous frequency of the same client request.

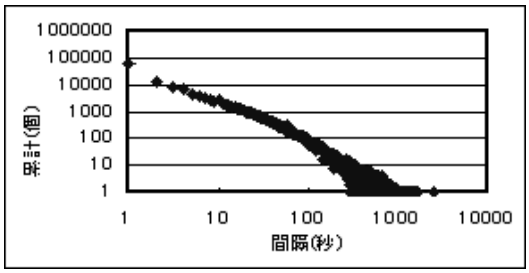

図 4 リクエスト到着間隔累計
Fig. 4 Total at request arrival intervals.

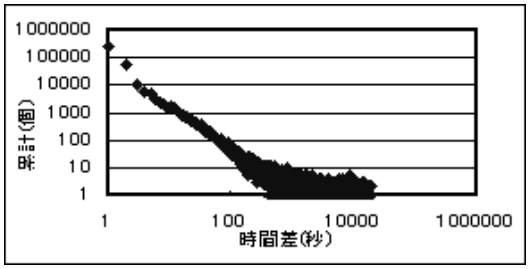

図 5 同一クライアントリクエスト到着時間差累計
Fig. 5 Total at same client request arrival intervals.

4. クライアントリクエストのモデル化

性能評価を行うにあたり，クライアントからのリクエストをシミュレートしたデータモデルを作成する．必要となる要素は，リクエストの到着頻度，リクエストを送ってきたクライアントの識別子，リクエストされたファイルのサイズである．

そこで，筑波大学学術情報メディアセンターの web サーバログ 1 カ月分のデータに対して以下の 3 つの統計データをとり，分析を行った．

(1) 同じ IP のクライアントからのリクエストが連続した個数 x と，その回数の累計 y の統計
(2) あるリクエストから次のリクエストが来るまでの間隔 x について，その個数の累計 y の統計
(3) あるクライアントからのリクエストを処理してから再び同じクライアントからのリクエストを処理するまでの時間差とその回数の累計

(1) をグラフ化したものを**図 3**，(2) を**図 4** に，(3) を**図 5** に示す．これらの図から，web サーバへのリクエストには以下のような偏りが存在すると考えられる．

- リクエストはバースト的に連続して到着することが多く，同じ時刻に複数のリクエストが来ることも多々ある．
- 同じクライアントからのリクエストは比較的短時間内に，連続して到着することが多い．

このような偏りを表現する入力データモデルを作成するため，図 3，図 4，図 5 のデータに回帰分析を行い，その累積分布関数の逆関数を導出した結果を式 (1)〜(3) にそれぞれ示す．

$$F^{-1}_{a(x)} = -0.72549/(x - 0.976927) \quad (1)$$
$$F^{-1}_{b(x)} = -0.475263/(x - 0.995667) \quad (2)$$
$$F^{-1}_{c(x)} = -0.444219/(x - 0.995429) \quad (3)$$

これらの式を用いて擬似リクエストを生成する．まず，式 (2) に対して（0-1）の一様乱数を与え，全リクエストの到着タイミングを決定する．次に式 (1) と (3) に対して一様乱数を与え，何件連続で同じクライアントからのリクエストが来たのか，どのリクエストとどのリクエストが同一クライアントからのリクエストなのかを決定する．最後に，一様乱数で同じクライアントには 0〜9 までの同じ ID を与える．この ID がクライアントの使用する暗号を表している．

こうして得られた実データとよく似た分布を示す擬似リクエストを，後の評価実験で入力とする．

また，要求されたファイルのサイズは，同様に筑波大学学術情報メディアセンターの Web サーバログ 1 カ月分のデータの統計をとってみると，ほぼ平均値 24 kbyte の指数分布に従うことが分かった．

5. リクエスト予測とオーバラップ

4 章の結果より，web サーバにおいては「きわめて短時間内に連続して同一クライアントからのリクエストが送信されて来る」という特徴が存在することが分かった．

このような特徴を利用して，クライアントリクエストの予測を行うことを考える．この予測により FPGA の再構成回数を減らすことで，そのオーバヘッドを低減させることを目的とする．

また，再構成オーバヘッドをソフトウェア処理とオーバラップさせることによる隠蔽を試みる．

表 3 2-gram 表（抜粋）
Table 3 A part of 2-gram table.

直前区間	直後区間	AES-ECB DES-CBC	0-9 0-9	AES-ECB DES-CBC	0-9 10-19	…	AES-ECB 3DES-ECB	0-9 30-39	…
AES-ECB 3DES-CBC	0-9 40-49	0.012		0.003		…	0.019		…
…		…		…		…	…		…
AES-ECB SERPENT	0-9 30-39	0.000		1.000		…	0.000		…
AES-ECB SERPENT	0-9 40-49	0.000		0.000		…	0.000		…
AES-ECB SERPENT	0-9 50-59	0.020		0.006		…	0.030		…
…		…		…		…	…		…

5.1 リクエスト予測

同じクライアントからの処理要求は非常に短い時間内に連続して来るという傾向にあると述べた．

ここで，単純のために同一クライアントからの処理要求はすべて同じ暗号化アルゴリズムを用いると仮定し，使用可能なアルゴリズムの種類は 3 章であげた 10 種類であるとすると，ある暗号アルゴリズムを用いるリクエストは一定時間内に連続して来る傾向にあるといえる．

つまり，近い将来にある特定の暗号方式を用いるリクエストが多数到着する，という偏りが起こりうると考えられる．この偏りの予測ができれば，それを基に FPGA の回路の再構成をすることで無駄な再構成の回数，すなわち FPGA 再構成のオーバヘッドを減らすことが可能であると考えられる．

5.2 2-gram 表

2-gram 表とは，ある値が出現した直後に，続けて別のある値が出現するという，2 つの値の共起関係が現れる頻度を表にまとめたものであり，主に自然言語処理における品詞解析などで用いられる．

本研究では，リクエストされたデータ 5 Mbyte 分の連続したリクエスト（おおよそリクエスト 200 件程度）を 1 区間とし，その区間の中で使用された暗号方式の，種類とその割合とを値とした 2-gram 表を作成する．

3.2 節で述べたように，1 つの FPGA には同時に 2 つの暗号化回路が搭載可能であるとした．そこで，サーバログの中のあるリクエストに注目し，その直前 1 区間と直後 1 区間の，それぞれのリクエストの中で使用頻度の高い暗号方式 2 種類とその頻度とを求める．この作業を全リクエスト全区間に対して行うことで，隣り合った前後の区間で使用頻度の高かった 2 種類の暗号方式とその割合の共起関係を測定し，この共起関係の出現確率をまとめることで 2-gram 表を作成する．

これにより，あるリクエストが到着したときに，その直前 1 区間分のリクエストで使用された暗号方式の種類と割合を基にこの 2-gram 表を引くことで，「次の 1 区間 200 件程度のリクエストで最も高い確率で来る暗号方式 2 種類の組合せとその割合」を確率的に求めることができる．

ここで，使用する暗号が 10 種類，FPGA 上に同時に 2 つの暗号処理回路を搭載可能とし，各暗号方式の使用割合を 0〜9%，10〜19% … 90〜99%の 10 段階と，ある 1 つの暗号方式のみが使用された 100%で表現すると，2-gram 表のサイズは，「(暗号 2 種類の組合せ 45）*（片方の暗号の使用頻度 10 段階）*（残りの片方 10 段階）＋（ある暗号方式 1 つのみ 100%使用 10 種類)」の行列で表現されるため，4510*4510 となる．実際に，筑波大学学術情報メディアセンターの web サーバログ 2003 年 12 月分から生成された 2-gram 表の一部を**表 3** に示す．

表 3 は，直前の区間で AES-ECB が 0〜9%で SERPENT が 30〜39%であった場合，次の直後の区間で AES-ECB が 0〜9%，DES-CBC が 10〜19%である確率が 1.000 であった，という意味になる．すなわち，必ずこのような暗号の組合せのリクエストが来た，という意味になる．

表 3 は，そのまま用いるには大きすぎるので，各行の要素の中で最も値の大きい要素，すなわち，ある区間の暗号方式の種類と割合に対して，その直後の区間で使用される確率の最も高い暗号方式の種類のみを集めた要素数 4,510 の行列（以後 2-gram 行列と呼ぶ）にまとめる（**表 4**）．これにより，ある区間の暗号方式の種類と割合が分かったときにこの行列をひくことで，直後の区間で使用される確率が最も高い暗号方式 2 種類の組合せを知ることができる．

また，2-gram 表を作成する際の基にした履歴情報

表 4 2-gram 行列（抜粋）
Table 4 A part of 2-gram matrix.

直前区間の暗号の	AES-ECB	0-9	AES-ECB	0-9	…	AES-ECB	0-9	AES-ECB	0-9
種類と割合	3DES-CBC	0-9	3DES-CBC	10-19	…	3DES-CBC	40-49	3DES-CBC	50-59
直後の区間で利用	AES-ECB		SERPENT		…	SERPENT		TWOFISH	
確率が最も高い暗号	3DES-ECB		RC6		…	AES-CBC		RC6（*）	

```
if(FPGA 状態==暗号処理中 or ソフトウェア暗号処理中) then
    暗号処理終了まで待つ
end if
if(FPGA 状態==再構成中) then
    ソフトウェア暗号処理
    FPGA 書き換えなし
else if(!(今来たアルゴリズム subseteq FPGA 搭載回路)) then /*&& FPGA 状態==idle*/
    if(手順 4 結果==FPGA 搭載回路)  then
        ソフトウェア暗号処理
        FPGA 書き換えなし
    else /*手順 4 結果!=FPGA 搭載回路*/
        ソフトウェア暗号処理
        FPGA 書き換え to 手順 4 結果
    end if
else /*今来たアルゴリズム ⊆ FPGA 搭載回路 && FPGA 状態==idle*/
    FPGA 暗号処理
    FPGA 書き換えなし
end if
```

図 6 手順 5 および 6 の詳細
Fig. 6 Details of procedure 5 and 6.

の中には現れなかった暗号方式の出現パターンというものが存在する．このとき，共起関係の測定は不可能であり，すべてのパターンの出現頻度が一律 0 となってしまう．このような場合については，履歴情報全体を通して見た中で，1 番目と 2 番目に頻度の高かった暗号方式を予測結果として用いる．これを 0 次予測と呼ぶ．表 4 において（*）の付いている項目は，今述べたような 2-gram 表に現れないパターンのリクエストであったため，0 次予測の結果を採用した部分である．

5.3 予　　測

前節で述べた 2-gram 表，2-gram 行列を基に実際に予測を行う大まかな手順は，以下 1〜8 のとおりである．

(1) 過去の履歴情報全体から 2-gram 表，2-gram 行列を作成する．
(2) リクエストが到着する．
(3) 2 のリクエストの直前 1 区間（200 件分程度のリクエスト）の中で，使用頻度の高かった暗号方式 2 種類とその頻度を調べる．
(4) (3) の情報を基に 2-gram 行列をひき，直後に使用される可能性の高い暗号方式 2 種類を求める．
(5) (2) と (4) と現在の FPGA の状態とを基に，暗号化処理をソフェアで実行するか FPGA で実行するかを決定する．
(6) (5) の結果と (2)，(4)，現在の FPGA の状態の情報を基に，FPGA を書き換えるのか，何に書き換えるのかを決定する．
(7) (6) を基に FPGA の状態を変更する．
(8) (2) に戻る．

ただし，現在のところ，一度作成した 2-gram 表および 2-gram 行列の更新は行わないこととしてある．

図 6 に，手順 (5) および (6) の詳細なアルゴリズムを示す．

手順 (5) において，FPGA を再構成中であった場合，または予測が外れるなどして今来たリクエストとは異なる暗号回路が載っていた場合には，ソフトウェアで暗号化処理を行うという決定がなされる．

そうでなければ，すなわち現在のリクエストを処理可能な暗号回路が FPGA 上に搭載されていた場合には，FPGA で暗号化処理を行うという決定がなされる．

また，手順 (6) において，手順 (4) で得られた予測結果と同じ暗号回路が FPGA 上に搭載されていた場合，および FPGA での暗号化を実行中であった場合には FPGA の再構成を行わないこととする．

5.4 オーバラップ

FPGA の再構成には，暗号化処理に比べて多大な時間が必要となる．通常のリコンフィギャラブルシス

テムでは，この再構成を行っているオーバヘッドの間は処理が完全に停止してしまうが，本システムではその再構成中には処理をソフトウェアで実行するという方法をとる．

このように，再構成オーバヘッド時間にソフトウェアでの処理をオーバラップさせることにより，その隠蔽をすることができる．こうすることで，全体的な処理時間の短縮が可能になり，各処理のレイテンシを低くすることが可能になる．

ただし，ソフトウェアでの暗号化処理と FPGA での暗号化処理のオーバラップ（並列暗号化処理）は，行わないこととする．暗号処理の並列化による高速化の影響を抑え，FPGA 再構成オーバヘッドの低減を図るためである．

6. 評価実験

このモデルの性能評価として，シミュレーション実験を行った．シミュレーションであるため，実際にデータの暗号化を行うわけではないが，リクエストデータサイズおよび表 2 の処理性能値を基に，処理にかかった時間などを計算し，比較評価を行う．

実験では，4 章で求めた累積分布関数の逆関数を基に作成された 3 つのパラメータ，「リクエストされたファイルサイズ」「リクエスト到着時刻」「リクエストしてきたクライアント ID」を持った擬似リクエストを生成する．

こうして生成されたリクエストを入力として，5 章で述べたオーバヘッド低減手法を使用した場合と使用しなかった場合など，以下にあげる様々なモデルでリクエストの頻度を 10 回/秒から 20,000 回/秒まで変化させて，**表 5** のパラメータで実験を行った．

- Software：FPGA を用いずにすべてソフトウェアのみで処理をする．
- Hardware：予測もオーバラップも用いずにすべて FPGA で処理を行う．FPGA 上に載っていない暗号回路を必要とするリクエストが来た場合には必ず FPGA を再構成する．
- Nopredict：オーバラップのみを用いて処理を行う．FPGA 再構成をする場合には，ランダムに 2 種類の暗号回路を搭載するようにする．再構成が終わる頃にはオーバラップしたソフトウェア処理でそのリクエストが終了してしまい，別の暗号方式のリクエストになっていることもありうる．
- Predict：予測とオーバラップを両方用いて処理を行う．
- Fix5：FPGA に，回路規模が小さい暗号方式 5 つを固定して搭載し，それら以外の暗号についてはソフトウェアで処理をする．
- Fix-ALL：すべての暗号処理回路をハードウェアとして保持しておく．回路規模が大きくなり，現実に実現するにはコストが非常に大きくなる．
- Oracle：将来のリクエストをすべて事前に知っているので必ず正しい予測をする．現実には実現不可能である．

表 5 シミュレーション実験環境
Table 5 Parameter of simulation experiment.

総処理データ数	61 万件
FPGA 再構成オーバヘッド	60 ms
パケット操作オーバヘッド	500 ns

予測を行う場合には，5.3 節に示した手順 (1)〜(8) を実行するため，まず，入力として筑波大学学術情報メディアセンターの web サーバログ 2003 年 12 月分を受け取り，5.2 節で示した 2-gram 表および 2-gram 行列を作成する．このとき，クライアント ID の 10 の剰余を暗号方式の識別番号とすることで，クライアント 1 つが 10 種類の暗号方式の 1 つに 1 対 1 で対応するようにした．

7. 考　　察

7.1 実験結果に関する考察

性能指標として，6 章で述べた各方式において各リクエストに対し，3 章で述べた暗号処理性能を基に「リクエスト到着時刻」から「リクエストの暗号化が終了した時刻」までのレイテンシを測定した．

このとき，ネットワークの速度，I/O 時間などは考慮せず，暗号化と FPGA の再構成に要した時間のみを測定した．**表 6** に結果を示す．

表 6 の一番右の列は Oracle のレイテンシ分布の 90%位の値を，それ以外の列は，Oracle の値を 1.0 としたときの相対値を表している．ここで 90%位の値というのは，全データの最悪値から 10%番目の値のことであり，すなわち，全データの 90%まではこの 90%位の値より小さな値を示すことを意味する．

表 6 より，Software と Hardware とを比較してみると，3 章の結果では，ソフトウェアでの処理性能よりも 1.5 倍から 15 倍程度 FPGA での処理性能のほうが高いにもかかわらず，Software の性能が数百倍以上良いことが分かる．これは，FPGA の書き換えのオーバヘッドによる性能の低下が，ソフトウェア処理と FPGA 処理の性能差を大きく上回っているためであると考えられる．

表 6 より Hardware と Nopredict とを比較すると，

表 6 レイテンシ分布の 90%位の値（Oracle 相対値）
Table 6 Latency 90% class value (relative for Oracle).

リクエスト頻度	Software	Hardware	Nopredict	Predict	Fix5	Fix-All	Oracle
10	3.70	118	1.17	1.00	1.50	0.78	0.000508
100	12.0	19700	1.29	1.03	1.90	0.59	0.000914
200	66.6	313000	2.02	1.16	3.67	0.33	0.00249
500	62.1	25900	3.00	1.37	10.3	0.20	0.0934
50	85.2	4900	2.74	1.31	7.95	0.33	0.569
1000	94.9	1570	3.14	1.24	6.63	0.30	1.89
2000	12.9	78.1	2.43	1.50	2.70	0.13	0.416
4000	3.30	19.0	1.34	1.12	1.40	0.34	0.0178
10000	2.57	13.3	1.23	1.08	1.20	0.55	0.0260
20000	2.07	12.2	1.21	1.08	1.20	0.59	0.0288

表 7 各暗号方式の使用頻度
Table 7 Use rate of each encryption.

暗号方式	AES-ECB	DES-EBC	DES-CBC	3DES-ECB	3DES-CBC
使用頻度	0.0911	0.0697	0.0886	0.0765	0.0907
暗号方式	SERPENT	TWOFISH	Camellia	RC6-ECB	AES-CBC
使用頻度	0.0786	0.2463	0.0706	0.0967	0.0848

FPGA 再構成のオーバヘッド時間をソフトウェアでの処理とオーバラップさせることによって，性能が数十倍から数万倍以上向上することが分かる．このような大きな性能向上が得られた原因は，以下のように考えられる．

1 回の FPGA 再構成に 60 ms のオーバヘッドが発生するが，これは 1 件のリクエストの暗号化に要する時間の数倍から数十倍の値となることがあり，このオーバヘッド中に次のリクエストが到着することがある．このような場合，現在のリクエストに発生したオーバヘッドが，次のリクエストのレイテンシに次々と積み重なっていってしまうことになる．しかし，再構成オーバヘッドをオーバラップすることで，このような事態の影響を抑えることが可能であるためと考えられる．

表 6 で Predict と Nopredict のレイテンシを比較する．Oracle に対して，Predict は平均 1.18 倍，Nopredict は平均 2.01 倍となった．このことから，予測を行うことによって性能を平均約 70%，最大約 250%向上させることが可能であることが分かった．

提案手法では，FPGA の再構成オーバヘッドを一律 60 ms とした．これはデバイスに固定の値であり，書き換え可能デバイスの中でも大きな部類に入る値であるにもかかわらず有効な性能向上結果が得られたと考えられる．デバイスのオーバヘッド自体が小さくなった場合，さらに性能向上が得られることが期待される．

表 6 より，リクエスト頻度が 400 回/秒から 1,000 回/秒あたりで，予測による性能向上率が 200%を超える結果となった．

表 8 対 software 暗号処理時間平均短縮率
Table 8 Average shortening rate at time of encryption.

暗号方式	Nopredict	Predict	Oracle
RC6-ECB	23.0	27.9	31.3
Camellia	58.8	61.6	64.9
DES-ECB	60.8	72.9	72.7
DES-CBC	46.0	53.3	55.0
AES-ECB	70.5	77.4	78.5
AES-CBC	40.3	42.1	42.9
3DES-ECB	49.6	68.8	72.0
3DES-CBC	21.9	25.5	26.3
SERPENT	19.4	28.2	36.2
TWOFISH	77.4	71.9	73.0
Total Average	47.6	53.4	55.1

しかし，それ以上，あるいはそれ以下の頻度ではすべての方式で性能差が縮まるという傾向が見られた．

次に，入力として与えた擬似リクエストの各暗号方式の使用頻度を**表 7** に示す．表 7 より，TWOFISH が他の暗号と比較してやや頻度が高いが，他の 9 種類の暗号に関しては，ほぼ一様な使用頻度となっていることが分かる．これは，4 章で述べたように同一クライアントからのリクエストが一定回数以上連続する，という特性により，TWOFISH の頻度がやや上がったためであると考えられる．今実験では，表 7 に示すようなほぼ一様な暗号使用頻度の場合にも有効な結果が得られたものと考えている．

Software と比較して，Nopredict，Predict，Oracle で，暗号方式ごとに暗号化のみに要した時間の短縮率の平均値を**表 8** に示す．これは暗号化に要した時間であり，表 6 で評価しているレイテンシとは異なる．表 8 より，使用頻度の高い TWOFISH と，ソフトウェ

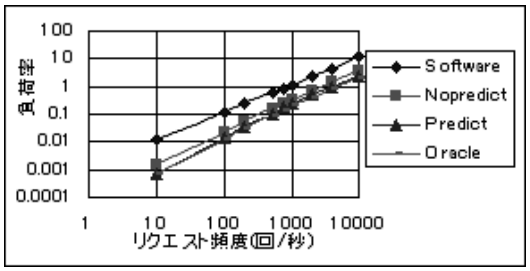

図 7 負荷率
Fig. 7 Load factor.

ア処理と FPGA 処理での性能差の大きい DES-ECB, AES-ECB で短縮率が高いことが分かる．

図 7 に，6 章で示した各方式における負荷率を示す．ここで負荷率というのは，最初のリクエストが到着してから，最後のリクエストの暗号化が終了するまでの時間に対して，データの暗号化に PC の処理能力を使用できる時間の割合である．この値が 1 を超えた場合，データの暗号化以外の処理が行えないことになる．つまり，ネットワーキングに必要な処理をいっさい行えないということを意味する．

ソフトウェアで暗号化処理を行わない Hardware, Fix-ALL，Fix5 などの負荷率は一律 0 である．

図 7 より，提案手法を採用した方式では，Software と比較して，数倍以上のリクエスト頻度まで，負荷率を 1 以下に抑えることが可能であると分かった．

本稿で提案した予測の方式は，クライアントリクエストの特徴を利用したものである．そのため，この特徴があてはまらないサーバに対して本システムを用いた場合，予測の正答率は極端に低くなることが考えられる．しかしながら，web サーバを対象とした場合は，クライアントリクエストには 5 章冒頭で述べた特徴が存在すると思われるので，予測の粒度やリクエストを区切る区間の大きさを変更することで，予測の正答率の大幅な低下は防ぐことができるのではないかと考えられる．

一方，処理のオーバラップに関しては，リクエストの特性にかかわらず実行可能である．そのため，サーバの種類にかかわらず，本システムを用いることによる性能向上は可能であると考えられる．

7.2 今後の展望

今実験では，暗号化処理はソフトウェア処理も FPGA 処理も並列には実行されない，というシステムモデルとした．ソフトウェア処理とのオーバラップを行ううえで，FPGA での暗号化処理とソフトウェアでの暗号化処理とを並列に行うことで，さらなる性能の向上が期待できる．

また，今回は予測の元となる履歴情報を一元的かつ静的に扱っている．今後は，履歴情報の鮮度による重み付けや，予測に用いる履歴情報を更新するなど，動的に扱った予測の方法についても考えていきたい．そのほかにも，クライアントを識別できるようなアルゴリズムを用いる，あるいは 3-gram 表を使用するなど，より精度の高い予測のアルゴリズムを考えていきたい．

8. 関 連 研 究

本研究と同様に，汎用 FPGA を利用したリコンフィギュラブルシステムを用いて処理の高速化を図るという研究が存在する．坂口らによる研究[13]，内田らによる研究[14] などでは，クラスタコンピューティングを行ううえで，各計算ノード上に FPGA を用いたリコンフィギュラブルシステムを搭載し，処理の一部を FPGA で実行することで高速化を図ることを目的としている．このとき，いつ，どのような回路に FPGA を書き換えるのかを，人間があらかじめ処理を開始する前にすべて決定しておくことを前提としている点で，本研究との相違が見られる．

また，Wirthlin らによる研究[15] では，アプリケーション固有の特殊な命令セットを直接実行するハードウェアモジュールを用意し，それを部分再構成可能 FPGA 上に実装することで実行するというものである．本研究とは，どのような命令を実行するかの予測を行わない点などで相違が見られる．

9. ま と め

クライアント・サーバ型の暗号通信において，暗号処理の部分を汎用 FPGA を動的に書き換えて実行するようなシステムのモデルを提案した．

このようなシステムで FPGA を書き換える際のオーバヘッドを低減するため，ソフトウェア処理とのオーバラップを行ってオーバヘッドの隠蔽を行い，また過去の通信履歴を基にした 2-gram 表をひくことで将来のリクエストを予測し，FPGA の無駄な書き換えを減らすという方法を提案した．

シミュレーション実験の結果，処理のオーバラップにより数十倍から数万倍，予測によりさらに平均で約 70%，最大で約 250%の性能向上が可能であるという結果が得られた．

謝辞 Web サーバのログの採取などに関して，筑波大学学術情報メディアセンター技術職員の佐藤守氏にご尽力いただきました．ここに感謝いたします．

また，本研究の一部は，日本学術振興会平成 15 年度科学研究費基盤研究（B）(2) 15300013「書き換え

可能デバイスによる高速パケット処理の研究」による．

参 考 文 献

1) 末吉，飯田：リコンフィギュラブルコンピューティング，情報処理，Vol.40, No.8, pp.778–782 (1999).
2) 丹羽，前田，山口：FPGA の動的書き換えシステムを用いた暗号通信のモデルとその性能，電子情報通信学会技術研究報告 CPSY，Vol.9, No.11, pp.61–66, 電子情報通信学会 (2004).
3) Xilinx: Virtex-II Platform FPGAs ハンドブック.
4) Nechvatal, et al.: Report on the Development of the Advanced Encryption Standard, Technical report, National Institute of Standards and Technology (2000).
5) http://www.nist.org/aes
6) http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/
7) http://www.gnupg.org/
8) Bassham, L., et al.: Efficiency Testing of ANSI C implementations of Round 2 Candidate Algorithms for the Advanced Encryption Standard, *3rd AES Candidate Conference*, pp.136–148, National Institute of Standards and Technology (2000).
9) Aoki, K., et al.: Camellia: A128-Bit Block Cipher Suitable for Multiple Platforms, *7th Annual Workshop on Selected Areas in Cryptography, SAC2000*, pp.41–54 (2000).
10) http://www.opencores.org/
11) Dandalis, A., et al.: A Comparative Study of Performance of AES Final Candidates Using FPGAs, *3rd AES Candidate Conference*, National Institute of Standards and Technology (2000).
12) Elbirt, A., et al.: An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists, *Third AES Candidate Conference*, pp.13–27, National Institute of Standards and Technology (2000).
13) 坂口ほか：リコンフィギャラブルハードウェアを用いたクラスタコンピューティングの高速化，リコンフィギャラブルシステム研究会論文集，Vol.4, No.9, pp.55–60, 電子情報通信学会 (2004).
14) 内田ほか：動的再構成システムのための機能ローディング機構の開発，リコンフィギャラブルシステム研究会論文集，Vol.4, No.25, pp.167–172, 電子情報通信学会 (2004).
15) Wirthlin, M. and Hutchings, B.: A Dynamic Instruction Set Computer, *FCCM'95*, IEEE Symposiumon Field-Programmable Custom Computing Machines (1995).

(平成 17 年 1 月 24 日受付)
(平成 17 年 5 月 6 日採録)

丹羽 雄平（学生会員）

2002 年筑波大学第三学群情報学類卒業．同年筑波大学大学院博士課程システム情報工学研究科コンピュータサイエンス専攻入学．現在同大学院在学中．FPGA による暗号処理，FPGA の利用法に興味を持つ．

前田 敦司（正会員）

1994 年慶應義塾大学大学院理工学研究科数理科学専攻単位取得退学．博士（工学）（慶應義塾大学 1997 年）．1997 年電気通信大学大学院情報システム学研究科助手．2000 年筑波大学電子・情報工学系講師．2004 年筑波大学大学院システム情報工学研究科助教授（現職）．並列/分散処理，コンピュータアーキテクチャ，プログラミング言語の実装，ガーベッジコレクション等に興味を持つ．日本ソフトウェア科学会，ACM 各会員．

山口 喜教（正会員）

1972 年東京大学工学部電子工学科卒業．同年通商産業省工業技術院電子技術総合研究所入所，計算機方式研究室長等を経て，1999 年筑波大学電子・情報工学系教授．博士（工学）（東京大学 1993 年）．現在，筑波大学システム情報工学科教授．高級言語計算機，並列計算機アーキテクチャ，並列実時間システム，ネットワーク侵入検知システム等の研究に従事．1991 年情報処理学会論文賞，1995 年市村学術賞受賞．著書『データ駆動型並列計算機』（共著）．IEEE Computer Society，ACM，電子情報通信学会各会員．