コンピュータセキュリティ 18−27
(2002. 7. 19)

共通鍵ブロック暗号 SC2000 の実装(III)

武仲 正彦† Helger Lipmaa‡ 鳥居 直哉†

†株式会社富士通研究所 〒674-8555 兵庫県明石市大久保町西脇 64
‡Department of Computer Science and Engineering Helsinki University of Technology
P.O.Box 5400, FIN-02015 HUT, Espoo, Finland
E-mail: †{takenaka, torii}@flab.fujitsu.co.jp, ‡helger@tcs.hut.fi

あらまし 共通鍵暗号 SC2000 の, IA-32 プロセッサ上のソフトウェア及びハードウェアにおける最新の高速実装結果について報告する．ソフトウェア実装では，SC2000 の特徴の一つである非均等 S-box の新たな結合方法を導入することで高速化を実現する．また Pentium III, Athlon 上で S-box の結合方法を変更した場合の処理速度遷移についての検討を行う．ハードウェア実装では，ASIC 実装での高速処理構成の検討を行い，その実装結果について述べる．特に本実装の結果，Pentium III 上の実装では 1.6 倍の高速化，ハードウェア実装では 26KGate で 1.4Gbps という高速処理を実現した．

キーワード SC2000，高速実装，IA-32 プロセッサ，ハードウェア

The Implementation of The Block Cipher SC2000 (III)

Masahiko TAKENAKA† Helger Lipmaa‡ and Naoya TORII†

†FUJITSU LABORATORIES LTD. 64 Nishiwaki, Ohkubo-cho, Akashi City Hyogo, 674-8555 Japan
‡Department of Computer Science and Engineering Helsinki University of Technology
P.O.Box 5400, FIN-02015 HUT, Espoo, Finland
E-mail: †{takenaka, torii}@flab.fujitsu.co.jp, ‡helger@tcs.hut.fi

Abstract We show the latest implementation results of the block cipher SC2000, which are a software implementation on the IA-32 processors and a hardware implementation. In the software implementation, we introduce a new combination strategy of non-homogeneous S-box that is one of the characteristics of SC2000, and achieve the fast implementation of SC2000. In addition, we discuss the transition of the speed by the combination strategies of S-box. In the hardware implementation, we introduce a new architecture for fast implementation of SC2000 and show the results of our implementation. In particular, we improve the software encryption speed by 1.6 times as compared to the previous best implementation, and we achieve the speed of 1.4Gbps with the hardware of 26KGates.

Keyword SC2000, Fast Implementation, IA-32 Processor, Hardware

1. はじめに

SC2000 は下山等によって提案された共通鍵暗号アルゴリズム[1]で，日本の電子政府向けの暗号技術評価(CRYPTREC)[2]における評価対象暗号アルゴリズムの 1 つである．SC2000 の特徴は，実装するプラットフォームに応じて実装アルゴリズムを変更することにより，高速実装から小規模実装まで効率よく実装可能なように設計されていることであり，この特徴を活用した，ソフトウェア，ハードウェアの高速，小規模実装が報告されている[3][4]．これらの結果から，IA-32 プロセッサ上でのソフトウェア性能が他のプロセッサ上の性能と比較して劣っていることがわかる．また，ハードウェア実装については，小規模実装では十分な速度での効率的な実装が示されているが，高速ハードウェア実装についてはオーソドックスなものしか示されていない．

本論文では，IA-32 プロセッサ上での高速ソフトウェア実装と ASIC を用いた高速ハードウェア実装について述べる．

ソフトウェア実装では，SC2000 の特徴の一つである非均等 S-box を（16,16）と結合する新たな方法を導入し，Pentium III 上での高速化を実現する．また Pentium III, Athlon 上で，S-box 結合方法を (6,5,5,5,5,6), (6,10,10,6), (11,10,11), (16,16)にした場合の実装結果を示し，結合方法により処理速度の遷移について検討する．

ハードウェア実装では，[4]で提案されている"RISC 型"の実装手法を応用し，処理遅延を保ったまま，処理サイクル数を削減することで，高い処理性能を小さい回路規模で実現する．

これらの検討の結果，Pentium III 上のソフトウェア実装では，従来の 1.6 倍の高速化を，ハードウェア実

装では26KGatesで1.6Gbpsの性能をそれぞれ実現している．

本論文では，2章でSC2000の概要，3章でIA-32プロセッサ上の高速ソフトウェア実装，そして4章でハードウェア実装について述べ，5章でまとめを行う．

2. SC2000の概要

本章ではSC2000アルゴリズムの構成について概要を述べる(詳細は[1]参照). SC2000は一般的な共通鍵アルゴリズムと同様に入力データを暗号化/復号するデータスクランブル部とユーザ鍵からデータスクランブル部で使用する拡大鍵を生成する鍵スケジュール部からなる．

2.1. データスクランブル部の構成

SC2000のデータスクランブル部は，32bit×4のデータ入出力の，I関数，B/B^{-1}関数，R関数から構成される．暗号化は，入力された平文に対して，鍵長が128bitの場合I-B-I処理を行ってから，R-R- I-B-I処理を6回，192, 256bitの場合はR-R- I-B-I処理を7回繰り返し処理し，暗号文として出力する．復号時は，B関数の代わりにB^{-1}関数を用い暗号時と同様の処理を行う．

I関数: I関数は32bit×4の入力データに4個の拡大鍵を各々XORし，出力する．拡大鍵を必要とするのはI関数のみである．

B/B^{-1}関数: B関数は，32bit×4の入力データを4bit×32に転置し，その32個の値それぞれを4bit S-Boxテーブル S4を用いて変換する．そして変換された4bit×32のデータを逆転置し，32bit×4として出力する．B^{-1}関数は，B関数の逆関数で，S4の代わりに4bit S-Boxテーブル S4iを使用する．なお，B/B^{-1}関数は，上述のような一般構成以外に，ビットスライス実装を行うことで高速化可能なように設計されている．

R関数: R関数はS, M, Lの3つの関数で構成され，入力32bit×4個データのうち2個をS, M, L関数で変換し，その出力32bit×2を残りの2個のデータにXORする．

S関数: S関数は，入力32bitを(6bit, 5bit, 5bit, 5bit, 5bit, 6bit)に分割し，それぞれ6bit S-Boxテーブル S_6 及び5bit S-Boxテーブル S_5 を索引する．そして索引結果の(6bit, 5bit, 5bit, 5bit, 5bit, 6bit)を再び32bitに結合して出力する．S関数は，上述のような(6,5,5,5,5,6)構成以外に，隣り合うS-Boxを結合して(6,10,10,6)や(11,10,11)等の結合方法(Strategy)を用いた実装を行うことで高速化可能なように設計されている．

M関数: M関数は，入力32bitと32×32のM行列をGF(2)上で乗算を行う関数である．これはM行列を32bitのテーブル32個 M_0~M_{31} として，入力値の各bitとの乗算を行い，その結果をXORするとみなすことができる．M関数は上述の行列演算構成以外に，S関数とM関数を結合したテーブルを用いる構成も可能である．

L関数: L関数は，32bit×2の入力に対してANDとXOR演算を行い出力する．

2.2. 鍵スケジュール部の構成

SC2000の鍵スケジュール部は，中間鍵生成部と拡大鍵生成部から構成される．

中間鍵生成部: 中間鍵生成部は，S関数とM関数で構成され，ユーザ鍵から32bit×12個の中間鍵を生成する．

拡大鍵生成部: 拡大鍵生成部は，12個の中間鍵から56/64個(鍵長128bit/鍵長192, 256bit)を生成する．1個の拡大鍵は12個の中間鍵から4個を取り出し，それに対して加減算とXOR，シフト演算で構成されるトーナメント型の処理により生成する．

3. IA-32プロセッサ上でのソフトウェア実装

3.1. 従来の実装結果

文献[1][3]では，IA-32プロセッサ上でのSC2000の高速実装結果が示されている．これらの結果をまとめたものを表1に示す．

文献[5]で青木等はAES最終候補をPentium II上で実装を行い，時間測定方法や，自己変更コードを使用しない等の実装基準について述べている．

本論文では，我々はこの指針に従って実装を行った．

3.2. 提案アルゴリズム

文献[3]で紹介されているSC2000の高速化手法のほとんどを今回提案する実装でも使用した．たとえば，B関数については[3]のPentium II/III用の論理表現を用いたビットスライス実装を行っており，M関数についてはテーブル化してS関数との結合を行っている．

提案する実装の特徴はS関数の結合である．文献[3]では，結合方法は(6,5,5,5,5,6), (6,10,10,6), (11,10,11)の3種類であった．これはS関数を結合することによって，テーブル索引回数を削減する手法である．各結合法における，R関数1回あたりのテーブル索引回数とテーブル量を表2に示す．

表2より，テーブル索引回数を削減すればするほど，テーブル容量が増加することがわかる．文献[3]では実験(表1参照)により，テーブル索引回数の削減が効果を示す上限を，テーブルサイズが1st-cache容量を超え

表1 従来のSC2000実装結果

Processor	Strategy	Encrypt	Decrypt	KeySchedule
Pentium III (Katmai)	(6,10,10,6)	383	403	427
	(11,10,11)	525	535	488
Athlon (K7)	(6,10,10,6)	392	402	426
	(11,10,11)	318	329	373

※ C+アセンブラ実装, VC++6.0, Windows上で測定

表2 S関数の結合方法とテーブル索引回数，サイズ

Strategy	#Table-look-up	Table size
(6,5,5,5,5,6)	6	1KB
(6,10,10,6)	4	8.5KB
(11,10,11)	3	20KB
(16,16)	2	512KB

るまでとしていた．これは，1st-cache 容量が 16KB の Pentium III の場合(6,10,10,6)実装が，1st-cache 容量が 64KB の Athlon の場合(11,10,11)実装がそれぞれ最高性能となることを表している．

本実装では S 関数を(16,16)と結合することを提案する．表 2 に示すように(16,16)実装の場合，R 関数 1 回あたりのテーブル索引回数は 2 回，テーブル容量は 512KB 必要である．Pentium III は 2nd-cache を 512KB 持っているので，テーブル索引が 2nd-cache にヒットすることを前提にプログラムを最適化することによって従来の結合方法を用いた場合より高速な実装が可能となる．一方 Athlon は 2nd-cache を 256KB しか持っていないため，(16,16)実装では(11,10,11)実装より低速になることが予想される．

3.3. 実装・評価環境

本論文で使用した開発・測定環境を以下に示す．

プロセッサ: 高速実装に用いられる IA-32 プロセッサとしては，Intel の P6 ファミリと P7 ファミリ，AMD の Athlon ファミリと Athlon4 ファミリがある．P6 ファミリのハイエンドは Pentium III, Pentium III-M (Tualatin-512K)で，P7 ファミリは Pentium4 (Northwood)である．同様に Athlon ファミリのハイエンドが Athlon (Thunderbird)で，Athlon4 ファミリは Athlon XP (Palomino)である．本実装では，広く一般に使用されていることや，従来結果との比較を考えて，Pentium III-M (Tualatin-512K)と Athlon (Thunderbird)を使用することとした．使用したプロセッサを以下に示す．

Processor	Frequency	1st-cache	2nd-cache
Pentium III-M (Tualatin-512K)	1.2GHz	16KByte	512KByte
Athlon (Thunderbird)	1.4GHz	64KByte	256KByte

コンパイラ: 本実装では，C 言語のみを使用している．コンパイルには 2 種類の異なるコンパイラ，Gnu C Compiler Version 3.0.4 (gcc 3.0.4)と Intel C Compiler Version 5.0 (icc 5.0)を使用する．これらのコンパイラでは Intel の MMX technology 命令や AMD の 3D now! 命令といった新しいマルチメディア命令は全く生成されることはない．つまり，本実装の処理速度はマルチメディア命令を使用していないといえる．gcc 3.0.4 で使用したコンパイルオプションを以下に示す．

```
-O4 -fomit-frame-pointer -mcpu=XXX
-march=XXX -D__OPTIMIZE__
-fexpensive-optimizations -funroll-loops
-mpreferred-stack-boundary=2
```

ここで，XXX はプロセッサタイプ(Pentium III-M の場合は"pentiumpro"，Athlon の場合は"athlon")と指定する．一方，icc 5.0 では特別な最適化オプションを使用しなかった．これは，高速化に貢献するオプションがなかったためである．

測定環境: 測定環境は，サーバでの使用を想定して OS に Linux を選択した．これはルータやファイアウォール等では Linux の使用頻度が高いためである．それらのマシンはワークステーションとして使用されるわけではなく，いくつかの特別な処理を行うだけなので，もしスループットが数割も向上するならば，500～600KB のメモリを暗号に使用することは妥当であると考えられる．測定には以下に示す 2 種類のマシンを使用した．

Processor	Memory	OS
Pentium III-M 1.2GHz	128MB	Linux 2.4.18
Athlon 1.4GHz	512MB	Linux 2.4.17

3.4. 実装結果

Pentium III, Athlon 上で(6,5,5,5,5,6), (6,10,10,6), (11,10,11), (16,16)の 4 種類の実装を行った結果を表 3 に，それをグラフ化したものを図 2 に示す．

表 3 より，Pentium III の場合は(16,16)実装で最高性能となった．この結果は従来の 1.6 倍高速である．一方 Athlon の場合，予想通り(11,10,11)実装が最高性能で(16,16)実装は低速となった．また図 2 より，Pentium III の場合，(11,10,11)実装の時にコンパイラによって性能が大きく違う．これは，gcc 3.0.4 を用いると 1st-cache を有効に利用するコードが生成されるのに対

表3 従来の SC2000 実装結果

Processor	Strategy	Compiler	Enc.	Dec.	Key
Pentium III-M (Tualatin-512K)	(6,5,5,5,5,6)	icc 5.0	521	527	519
		gcc 3.0.4	503	824	665
	(6,10,10,6)	icc 5.0	409	414	483
		gcc 3.0.4	388	609	511
	(11,10,11)	icc 5.0	349	356	427
		gcc 3.0.4	452	561	501
	(16,16)	icc 5.0	270	277	356
		gcc 3.0.4	269	489	359
Athlon (Thunderbird)	(6,5,5,5,5,6)	icc 5.0	471	478	427
		gcc 3.0.4	463	843	534
	(6,10,10,6)	icc 5.0	413	376	404
		gcc 3.0.4	366	606	438
	(11,10,11)	icc 5.0	319	327	361
		gcc 3.0.4	312	507	392
	(16,16)	icc 5.0	362	381	280
		gcc 3.0.4	374	543	309

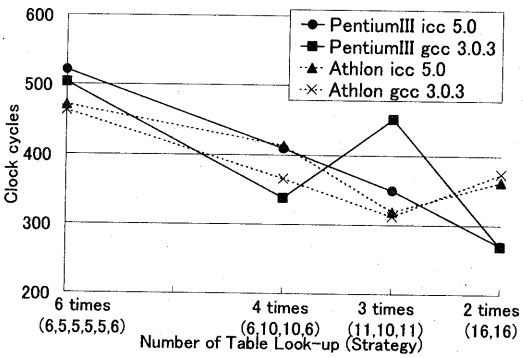

図 2 Pentium III, Athlon におけるテーブル索引回数と処理速度の遷移

表 4 SC2000 と他暗号との比較

Cipher	Compiler	Enc.	Dec.	Key Schedule Enc.	Key Schedule Dec.
AES	assembly	226	226	N/A	N/A
	icc 5.0	430	453	185	319
	gcc 3.0.4	346	371	171	280
RC6	assembly	222	205	N/A	N/A
	icc 5.0	257	289	1436	Enc
	gcc 3.0.4	234	265	1778	Enc
SC2000	icc 5.0	270	278	357	Enc
	gcc 3.0.4	269	376	357	Enc

し，icc 5.0 を用いると 2nd-cache を有効に利用するコードが生成されるのではないかと推測される．

次に，Pentium III 上で SC2000 と他との暗号の性能の比較を行う．比較対象は IA-32 プロセッサで高速といわれている AES[6]と RC6[7]とし，その性能は我々の知る限り最高速の値[8]を使用した．比較結果を表 4 に示す．

表 4 より Pentium III 上の C 言語実装においては，SC2000 は AES より高速であり RC6 と同程度の性能となった．またこれは，AES や RC6 のアセンブラ実装と比較しても遜色のない性能であるといえる．

4. 高速ハードウェア実装

4.1. 従来の実装結果

SC2000 のハードウェア実装結果が示されているものには，文献[1][3][4]がある(表 5 参照)．文献[1][3]のハードウェア構成はオーソドックスなもので，I 関数，B 関数，R 関数を直列に接続して，1 サイクルで可能な限りの処理を行うような構成である．この構成のことを文献[4]では"CISC 型"構成と呼んでいる．これに対して，文献[4]では"RISC 型"と呼ばれる構成を用いることで，SC2000 ハードウェアを回路規模 8.9KGate，スループット 200Mbps という，小規模で効率的な実装を実現している．

表 5 従来のハードウェア実装結果

テクノロジ	構成	回路規模 (KGate)	処理速度 (Mbps)
0.25μm	大規模(unroll)	226	1290
	中規模(loop)	63	964
	小規模(loop)	33	264
0.18μm	小規模(loop)	8.9	200

"RISC 型"構成の特徴は，I 関数，B 関数，R 関数を並列に接続することである．そして，各関数を 1 サイクルで処理するのではなく，分割して数サイクルで処理することで回路規模の削減を図る．さらに，同時処理可能な部分を並列実行するための結合と，1 サイクルあたりの遅延時間の削減で，処理効率を上げている．たとえば，文献[4]の回路では，SC2000 の R-R-I-B-I の部分を，(R/2+I/4)-(R/2+I/4)-(R/2+I/4)-(R/2+I/4)-(B/4)-(B/4)-(B/4)-(B/4)-(I/4)-(I/4)-(I/4)-(I/4)の 12 サイクルで処理している．この構成をとることで，各サイクルで必要な拡大鍵は最大 1 個となり，鍵スクランブル部の回路規模削減が同時に実現されている．

4.2. 提案する回路構成

文献[4]では"RISC 型"構成の手法を回路規模削減に応用した．それに対し，今回提案する構成では，同じ手法を高速化に応用する．

一般にハードウェア構成は「回路規模」，「処理サイクル数」，「1 サイクルの処理遅延」の 3 つのパラメータで評価でき，"CISC 型"構成は「1 サイクルの処理遅延」を増加させる方針で，逆に"RISC 型"構成は減少させる方針であると言える．文献[4]の回路は，残り 2 つのパラメータのうち「回路規模」を削減し，代わりに「処理サイクル数」を増加する構成である．これに対し，本提案では"RISC 型"構成を用いることで「1 サイクルの処理遅延」を低く保ったまま，「処理サイクル数」を削減し，代わりに「回路規模」を増加させる．本提案の具体的な構成を以下に示す．

R 関数: 文献[4]は R 関数を 2 分割した R/2 と I 関数

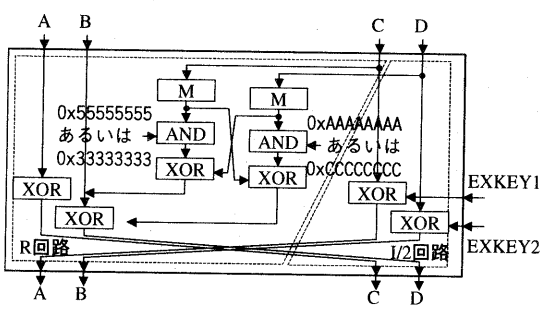

図 3 R 関数の(R+I/2)構成図

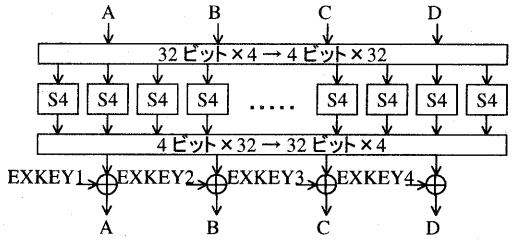

図 4　B 関数の(B+I)構成図

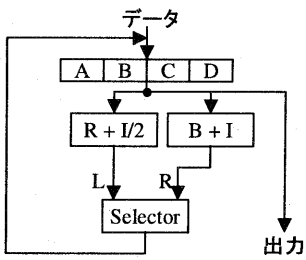

図 5　データスクランブル部の(R+I/2), (B+I)構成

を 4 分割した I/4 を結合した(R/2+I/4)の構成であった．この処理遅延を変えずに最大の処理を行う構成は(R+I/2)である．これは，(R/2+I/4)2 段分を結合したものといえる．(R+I/2)構成を図 3 に示す．

B/B-^{1}関数: 文献[4]は B 関数を 4(2)分割した B/4(B/2)で構成されている．B 関数の分割は処理遅延に影響しないので，本構成では分割しないオーソドックスな B の構成を採用する．さらに R 関数の処理遅延と比較すると B 関数の処理遅延は半分程度であるので，I 関数を直列に接続した(B+I)構成を用いて処理遅延の均一化を図る．(B+I)構成を図 4 に示す．なお B^{-1} 関数は B の部分を B^{-1} に入れ替えた構成である．

データスクランブル部: 本構成では，SC2000 の R-R-I-B-I の部分を，(R+I/2)-(R+I/2)-(B+I)の 3 サイクルで処理可能である．(R+I/2)と(B+I)とで構成したデータスクランブル部を図 5 に示す．

鍵スケジュール部: 文献[4]の構成では各サイクルで必要な拡大鍵は最大 1 個であったのに対し，本構成で必要な拡大鍵は(R+I/2)処理時に 2 個，(B+I)処理時に 4 個となる．拡大鍵の必要最大数は 4 個，1 サイクルあたりの平均必要拡大鍵数は 2.7 個である．同時に 4 個の拡大鍵生成を行うためには拡大鍵生成部が 4 個必要で，そのうち 2 個は 2/3 の確率で動作しないため，回路効率が悪い．そこで，本構成では拡大鍵生成部を 3 個とバッファを用意することでにより回路効率を向上する．(R+I/2)-(R+I/2)-(B+I)の各サイクルで 2 個-3 個- 3

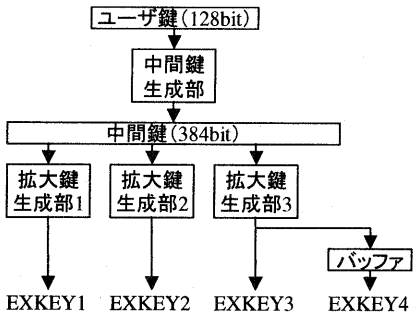

図 6　鍵スケジュール部の構成

個の拡大鍵生成を行い，第 2 サイクルで使用しない拡大鍵をバッファに保存，それを次サイクルで生成される 3 個の拡大鍵と同時に出力することで（B+I)サイクルで 4 個の拡大鍵使用に対応する．本構成で使用した鍵スケジュール部を図 6 に示す．

4.3. 実装・評価環境

本構成の実装環境には，富士通製 0.18μm テクノロジ 3 層 CMOS プロセス ASIC である CS81 シリーズ[9]を選択した．また，設計環境としては，SYNOPSYS 社製 Design Compiler[10] 2000.05-1 を使用した．シミュレーション条件はコマーシャルワーストを採用した．

暗号アルゴリズムの実装評価条件としては，鍵長は 128-bit 専用とし，モード処理等を含まない SC2000 コア関数（暗号化，復号，鍵スケジュール)のみとした．

4.4. 実装結果

以上の提案構成を用いた高速処理向け SC2000 の実装結果を表 6 に示す．評価項目は鍵セットアップ時間，処理サイクル数，動作周波数，ゲート数，スループット，効率(スループット／ゲート数)で，速度重視でシミュレートした結果と効率重視でシミュレートした結果を示した．なお動作周波数は，回路の最大遅延時間の逆数から換算したものである．

この結果 SC2000 は 1.4Gbps で処理可能であり，そのときの回路規模は 26KGate である．

文献[3]と比較すると，実装条件が違うものの，中規模実装と比較してスループット 1.5 倍，回路規模 1/2.4，大規模実装と比較してもスループット 1.2 倍，回路規模 1/10 となり高速で効率的な実装となった．

表 6　SC2000 小規模実装結果

評価項目	速度重視	効率重視
鍵セットアップ時間 (nsec)	69.5	76.5
処理サイクル数 (cycles)	22	22
動作周波数 (MHz)	244.5	222.2
ゲート数 (KGate)	26.4	21.6
スループット (Mbps)	1422.5	1292.8
効率 (Mbps / KGate)	53.8	59.8

5. まとめ

本論文では，共通鍵ブロック暗号 SC2000 の高速ソフトウェア・ハードウェア実装について，実装アルゴリズム及び回路構成を述べ，それらの実装結果を示した．

ソフトウェア実装においては，従来 SC2000 の性能が発揮されないと思われていた IA-32 プロセッサ上でも，高速な処理が可能であることが判明した．特に Pentium III プロセッサ上では C 言語実装で暗号化 1 回あたり 269 クロックサイクルで処理可能であり，他の暗号アルゴリズムと比較しても十分高速であることを示した．

ハードウェア実装においては，回路規模 26KGate でスループット 1.4Gbps と高速通信にも十分対応できる処理速度を効率よく実現した．

今後の課題としては，IA-32 プロセッサ上のアセンブラ実装を行うことで，ソフトウェア実装を更に高速化することである．また，2nd-cache まで利用する実装方法の更なる効率化も検討課題である．

参考文献

[1] 下山，屋並，武仲，伊藤，矢嶋，鳥居，田中，「共通鍵ブロック暗号 SC2000」,信学技報 ISEC2000-72, pp. 101 - 130, 2000.

[2] 情報処理振興協会，通信・放送機構，(http://www.ipa.go.jp/security/enc/CRYPTREC/index.html)

[3] 武仲，岡田，矢嶋，鳥居，「共通鍵ブロック暗号 SC2000 の実装」, 2001 年暗号と情報セキュリティシンポジゥム SCIS2001, 13A-4, pp743 - 748, 2001

[4] 武仲，岡田，矢嶋，鳥居，「共通鍵ブロック暗号 SC2000 の実装」, 2002 年暗号と情報セキュリティシンポジゥム SCIS2002, 9B-4, pp605 - 610, 2002

[5] K. Aoki and H. Lipmaa, "Fast Implementations of AES Candidates," In The Third Advanced Encryption Standard Candidate Conference, pp. 106 – 120, New York, NY, USA, April 2000. http://csrc.nist.gov/encryption/aes/round2/conf3/aes3conf.htm.

[6] J. Daemen and V. Rijmen, "The Design of Rijndael. AES - The Advanced Encryption Standard," Springer-Verlag, 2002.

[7] R. Rivest, M. Robshaw, R. Sidney, and Y. Yin, "The RC6 Block Cipher. Available from http://theory.lcs.mit.edu/~rivest/rc6.ps, 1998

[8] H. Lipmaa, "AES Candidates: A Survey of Implementations," (http://www.tcs.hut.fi/~helger/aes/)

[9] 富士通, "セミカスタム CMOS スタンダードセル CS81 シリーズ", 2002, (http://edevice.fujitsu.com/fj/DATASHEET/j-ds/j620206.pdf)

[10] Synopsys, "Design Compiler ファミリ",2002 , (http://www.synopsys.co.jp/products/designcompilerfamily/)