Project Number	IST-1999-12324
Project Title	NESSIE
Deliverable Type	**Report**
Security Class	Public
Deliverable Number	D21
Title of Deliverable	**Performance of Optimized Implementations of the NESSIE Primitives**
Document Reference	NES/DOC/TEC/WP6/D21/2
Contractual Date of Delivery	Y3 M11
Actual Date of Delivery	Y3 M11
Revised Version	Y4 M2
Editors	Technion
Abstract	Performance of Optimized Implementations of the NESSIE Primitives
Keywords	NESSIE, Performance Evaluation.

Version 2.0

February 20, 2003

Performance of Optimized Implementations of the NESSIE Primitives [†]

B. Preneel[1], B. Van Rompay[1], S. B. Örs[1], A. Biryukov[1],
L. Granboulan[2], E. Dottax[2],
M. Dichtl[3], M. Schafheutle[3], P. Serf[3], S. Pyka[3],
E. Biham[4], E. Barkan[4], O. Dunkelman[4], J. Stolin[4],
M. Ciet[5], J-J. Quisquater[5], F. Sica[5]
H.Raddum[6], M. Parker[6]

February 20, 2003

[†]The work described in this report has been supported by the Commission of the European Communities through the IST program under contract IST-1999-12324. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

[1]Katholieke Universiteit Leuven, Dept. Elektrotechniek-ESAT/COSIC, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium

[2]École Normale Supérieure, Département d'Informatique, 45 Rue d'Ulm, Paris 75230 Cedex 05, France

[3]Siemens AG,Otto-Hahn-Ring 6, München 81732, Germany

[4]Technion, Computer Science Dept., Haifa 32000, Israel

[5]Université Catholique de Louvain, Dept. ELEC, Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium

[6]Universitetet i Bergen, Department of Informatics, Bergen, Norway

Executive Summary
Performance Evaluation II
(NESSIE Deliverable D21)

This document reports the results of NESSIE workpackage Performance Evaluation II (WP6).

The goal of this second phase of performance evaluation is to provide performance evaluation of the NESSIE submissions to support the choice of primitives. The requirement is to identify those submissions with such problems as to disqualify them from consideration as NESSIE recommendations.

The results given here have been thoroughly studied since the submission of candidates to Nessie to September 2000, and in particular since the last publication of the NESSIE performance results in NESSIE document D17 on March 2002.

Performance criteria are defined that provide discrimination between levels of performance assessment. A number of 'standard' algorithms are adopted as benchmarks to aid comparability, like DES, 3DES and Rijndael, along with various other algorithms to which we had software.

This document quotes various results from NESSIE documents D14 and D17 for which we have no updates at this stage, or which are inherently unmodified (e.g., claims of submitters). The results and the theoretical methods of measurement follow the approach of WP4, with various significant improvements to the measurement software.

The evaluation environment consists of:

- software measurement tools specially designed and optimized for this purpose, with a specially designed API for the implementation of the primitives, designed to provide uniformity in the results, and simplicity in developing and running the measurement software;

- standard instrumentation provided by platforms representing a wide range of computing platforms (Intel Pentium with MS windows and with Linux, Compaq Alpha, Sun Sparc, Apple PowerPC, and others);

- a standard template to identify and document critical characteristics and computational operations influencing performance.

The performance analysis comprises two components:

- short theoretical analysis of the primitives, based on the template data, mostly identical to the results in D14 and D17;

- practical measurement of actual implementations running on a wide range of computing platforms, in which speeds of all the types of primitives were much extended, optimized and improved.

Some very substantial discrepancies have been observed between the practical measurements of the various submissions. This is partly attributable to inherent performance characteristics, and partly to implementations ranging from near-optimal to fairly crude, despite attempts to normalize and to obtain or to generate good quality code wherever possible. It was our intention that these discrepancies should be reduced as much as possible in this document, a goal that we tried to reach by supplying new optimized implementations for Camellia, Idea, DES, Shacal, Whirlpool, and various other ciphers, some of which are targeted only to specific machines (e.g., DES for Alpha), while others work on a variety of machines.

The theoretical work is less susceptible to this variability, but nevertheless judgments must be made about the weightings given to computational operations on the various platforms. The results provide a meaningful comparison of underlying performance characteristics. The analysis also gives an indication of the degree of optimization in the implementations and of the applicability to the platforms under consideration.

Some of the primitives were especially designed for use on smartcards. Our current results for smartcard implementations are described at the end of this document along with our results on hardware implementation. We also list results of implementations external to NESSIE where we did not make similar implementations ourselves (e.g., assembly).

For performing the performance tests we have collected a large set of test vectors. We intend to publish the test vectors of all the asymmetric schemes in the NESSIE web site soon. We also applied the NESSIE statistical tests to all the codes of the block and stream ciphers mentioned in this report and verified that all the codes pass all the tests.

Contents

List of Tables

List of Figures

1 Introduction

NESSIE (New European Schemes for Signature, Integrity, and Encryption) is a research project within the Information Societies Technology (IST) Programme of the European Commission. The participants of the project are:

- Katholieke Universiteit Leuven (Belgium), coordinator;
- Ecole Normale Supérieure (France);
- Royal Holloway, University of London (U.K.);
- Siemens Aktiengesellschaft (Germany);
- Technion - Israel Institute of Technology (Israel);
- Université Catholique de Louvain (Belgium); and
- Universitetet i Bergen (Norway).

NESSIE is a 3-year project, which started on January 1st, 2000. The main objective of the project is to put forward a portfolio of strong cryptographic primitives obtained after an open call and evaluated using a transparent and open process. The evaluation concerns both the security and the performance aspects of the primitives. This report presents the state of Performance Evaluation work at the end of its second phase.

Detailed and up to date information on the NESSIE project is available at the project web site: `http://cryptonessie.org`.

Deliverable D21 is the final document of Performance Evaluation II. It provides a detailed review of all performance procedures and results undertaken up to January 2003. It provides new and improved speed estimates for the candidates submitted to NESSIE (along with standard primitives and other primitives to which we made implementations), with emphasis on the primitives accepted for the second phase of NESSIE. These estimates are the results of a long and thorough work, in which we created a test suite including codes for all the candidates (except for some of the asymmetric ones) following a special NESSIE API, along with a special software that make the actual measurement.

In total we have tested 285 different implementations for over 138 different variants of the measured primitives, with a total CPU time of several thousands hours for testing on all platforms.

We would like to thank Pnina Cohen, Dan Kenigsberg, François Koeune, Gaël Rouvroy and François-Xavier Standaert who spent a lot of time implementing NESSIE candidates.

2 The Submissions Received by NESSIE

- **Block ciphers**
  - **Anubis.** [6] (tweaked version is not considered),
  - **Camellia.** [5],
  - **CS-cipher.** [17]
  - **Grand Cru.** [11],
  - **Hierocrypt.** [46] Two distinct variants: Hierocrypt-L1 and Hierocrypt-3.
  - **IDEA.** [34]
  - **Khazad.** [7] (original+tweak)
  - **Misty1.** [43]
  - **Nimbus.** [40]
  - **Noekeon.** [14]
  - **Nush.** [36]
  - **Q.** [44],
  - **RC6.** [29],
  - **Safer++.** [41]
  - **SC2000.** [56]
  - **SHACAL.** [21] Two distinct variants: SHACAL-1 and SHACAL-2.
- **Stream ciphers and pseudo-random numbers generators**
  - **BMGL.** [22] (original+tweak),
  - **SNOW.** [16] (original+tweak)
  - **SOBER.** [23] Two distinct variants: SOBER-t16 and SOBER-t32.
  - **LEVIATHAN.** [45]
  - **LILI-128.** [15],
- **Hash functions**
  - **Whirlpool.** [8]
- **Message authentication codes**
  - **UMAC.** [32]
  - **Two-Track-MAC.** [67]
- **Asymmetric encryption**

  - **ACE-KEM.** [55] Upgrade of ACE-Encrypt.
  - **EPOC.** [19] Three distinct variants: EPOC-1, EPOC-2 and EPOC-3.
  - **ECIES.** [28]
  - **PSEC.** [18] Four distinct variants: PSEC-1, PSEC-2, PSEC-3 and PSEC-KEM.
  - **RSA-OAEP.** [30] Revised to RSA-KEM [58]

- **Digital signature schemes**
  - **ACE Sign.** [55]
  - **ECDSA.** [27]
  - **ESIGN.** [47] Tweaked to ESIGN-D.
  - **FLASH family.** [52] Three variants: FLASH, SFLASH and SFLASHv2.
  - **QUARTZ.** [12] QUARTZ was tweaked twice. In this document we do not consider the second tweak.
  - **RSA-PSS.** [31]

- **Digital identification schemes**
  - **GPS.** [53]

3 Performance Evaluation Methodology

Performance evaluation is an essential part in determining the practicality of a cryptographic algorithm. An algorithm that performs well is more likely to be adopted for practical applications.

NESSIE primitives will be used on a variety of platforms: PCs, smart cards, hardware, and in various other applications. Some applications areas impose very high performance requirements (such as hard disk encryption) and protection of high speed communications (Gigabit networks). For others, an acceptable performance is required in a low-end hardware platform and/or in compact hardware (cellular phone, smartcard). In general, we retain the candidates which are flexible on more than one platform and those which perform above average on a particular platform.

3.1 Performance Criteria

On PC (under Windows and Linux) and Unix machines speed is the main concern. We measure speed in the most uniform possible way: using clock counts as measured by the C `clock` function. Hardware and smartcards also influence the selection.

3.2 Comparison with Standards and Well-Known Algorithms

The NESSIE project also takes into account existing and emerging standards, even if these have not been formally submitted to the NESSIE project. Two recent examples in this context come from the standardization efforts run by NIST. The NESSIE project has contributed extensive comments to the AES process and Vincent Rijmen, one of the designers of the AES algorithm 'Rijndael', is a former member of the NESSIE project team. It is therefore decided that in the evaluation of block ciphers, the Rijndael algorithm should be used as a benchmark. While it is not possible to anticipate the NESSIE results, one can expect that the research by NESSIE on the security of block ciphers may well increase the confidence in and acceptability of the Rijndael algorithm as a standard. The NESSIE project also study the security and performance of SHA-2/256, SHA-2/384 and SHA-2/512, the new hash algorithms standardized recently by NIST to extend the result of SHA-1 to hash results between 256 and 512 bits.

The speed performance of candidates is compared with well-known algorithms:

- DES and triple-DES for 64-bit block ciphers,
- Rijndael, the FIPS standard for 128-bit block ciphers,
- Kasumi, the new 3GPP block cipher,
- Skipjack, the NSA's escrow encryption cipher,
- The AES finalists: Mars, Serpent, Twofish (RC6 is submitted to NESSIE, and for Rijndael see above),

- RC4, such as distributed in the OpenSSL package, for synchronous stream ciphers,
- SHA-1, SHA-2/256, SHA-2/384 and SHA-2/512 for hash functions,
- HMAC-SHA-1 and CBC-MAC with Rijndael and DES as underlying block ciphers for MACs,
- Other 'non-standard' primitives, such as RC5, Seal, Scream, various HMAC's and CBCMAC's, etc.

4 Theoretical Analysis

4.1 Template Results

4.1.1 Block Ciphers

Block ciphers are symmetric encryption primitives that typically encipher blocks of 64 or 128 bits. For most block ciphers, the ciphertext is obtained by repeatedly applying a relatively simple cryptographic function to the input. The simple cryptographic function is called a *round function* and the key-derived material used in the round function is called a *round key*. The round keys are computed from the key using a *key-schedule* algorithm. An *SP-network* is type of block cipher which has the effect of modifying the entire data block in each round. A *Feistel* cipher is a type of block cipher which modifies only half of the block in each round.

The NESSIE call for primitives specified the following security levels for block ciphers.
High: Key length of at least 256 bits. Block length at least 128 bits.
Normal: Key length of at least 128 bits. Block length at least 128 bits.
Normal-Legacy: Key length of at least 128 bits. Block length 64 bits.

Table 1 lists the block ciphers, the sizes of the blocks, and the security levels and key sizes. The top part lists the ciphers studied in the second phase of NESSIE, the middle part the other ciphers submitted to NESSIE, and the bottom part other ciphers to which we measured performance.

NOTES: NUSH was designed with two different block sizes: 64 bits and 128 bits. RC6 has a variable block length $4w$ bits, where $w \geq 32$ is recommended by the designers. There are two variants of SAFER++, one with 64-bit blocks and one with 128-bit blocks.

We can separate block ciphers in five different categories:

1. 64-bit block ciphers: CS-Cipher, Hierocrypt-L1, IDEA, Khazad (and the tweaked variant), Misty1, Nimbus, SAFER++, NUSH,
2. 128-bit block ciphers: Anubis, Camellia, NUSH, Grand Cru, Hierocrypt-3, Noekeon, Q, SC2000, SAFER++, NUSH, RC6,
3. 160-bit block ciphers: SHACAL-1,
4. 256-bit block ciphers: SHACAL-2, RC6.

A summary of the theoretical results is given in Table 2. More details about them can be found in D14 [51].

4.1.2 Synchronous Stream Ciphers

The stream ciphers submitted to NESSIE were as follows:

- LILI-128

Table 1: Block Ciphers Submitted to NESSIE, Along with Block Ciphers Whose Performance was Measured by NESSIE

Name of Cipher	Block Size	Submitted/Tested Keysizes		
		Normal-Legacy	Normal	High
IDEA	64	128		
Khazad	64	128		
Khazad - tweaked	64	128		
Misty1	64	128		
SAFER++	64,128	128	128	256
Camellia	128		128, 192	256
RC6	128,256		128	256
SHACAL-1	160			512
SHACAL-2	256			512
CS-Cipher	64	128		
Hierocrypt-L1	64	128		
Nimbus	64	128		
NUSH	64,128	128	128	256
Anubis	128		128, 160, 192, 224	256, 288, 320
Grand Cru	128		128	
Hierocrypt-3	128		128, 192	256
Noekeon Direct	128		128	
Noekeon Indirect	128		128	
Q	128		128	256
SC2000	128		128, 192	256
CAST-128	64		128	
DES	64	56		
Triple-DES	64	168		
Kasumi	64	128		
RC5	64	64		
Mars	128		128,192	256
Rijndael	128,256		128,192	256
Serpent	128		128,192	256
Skipjack	64	80		
Seed	128		128	
Twofish	128		128,192	256

Table 2: Block Ciphers Template Results

Theoretical Analysis	**Block size (bits)**	**Word size (bits)**	**Key size (bits)**	**Subkey size (bits)**	**Table lookups/Table size (bits×bits)**	**Shifts/Rotations + multiplications**	**XOR, ADD (bit size)**	**AND, OR, NOT**	**Total table lookups**	**Total logical operations (8-bit)**	**Total multiplications (8-bit)**	**Code size (8-bit) (kB)**	**Optimised in submission?**
Cs-cipher	64	8	128	576	192(8x8)	96(8bit)	488(8bit)	96(8bit)	192	584	0	7	yes
Hierocrypt-L1	64	8	128	896	8(8x8), 48(8x32), 40(8x64)	72(32bit)	60(32bit), 36(64bit)	96(32bit)	96	912	0	12	no
Idea	64	16	128	832		42+34 32-bit multiplications	388 (32bit)		0	1552	136	22.6	no
Khazad	64	8	128	576	64(8x64)	64	64(64bit)		64	512	0	58	no
Khazad - tweaked	64	8	128	576	64(8x64)	64	64(64bit)		64	512	0	58	no
Misty1	64	32	128	512	24(7x7), 48(9x9)	0	56(32bit), 72(16bit), 24(8bit)	10(32bit), 24(16bit), 10(32bit)	72	390	0	6.9	no
Nimbus6/1	64	64	128	704	80(4x64)	75+5 multiplications	86(64bit)	80(64bit)	80	1328	40		no
Nimbus6/2	64	64	128	704	40(8x64)	35+5 multiplications	46(64bit)	40(64bit)	40	688	40		no
Nimbus6/3	64	64	128	704	5(4x64), 25(12x64)	25+5 multiplications	36(64bit)	30(64bit)	30	528	40		no
Nush64 (legacy)	64	16	128, 192, 256	1280		36(16bit)	116(16bit)	36(16bit)	0	304	0	10	yes
Safer++ (legacy)	64	8	128	1152	64(8x8)	128	17(32bit), 404(8bit)		64	472	0	35	no
DES	64	32	56	768	128/8(6x4), 8/8(8x32), 64/11(8x48), 16/16(8x64), 0/8(8x56)		6(32bit), 64(48bit), 14(64bit)		216	520	0		n. a.
Triple DES	64	32	56		384/8(6x4), 8/8(8x32), 192/11(8x48), 16/16(8x64), 0/8(8x56)		6(32bit), 192(48bit), 14(64bit)		600	1288	0		n. a.
Anubis ($R = keysize/32 + 8$)	128	8	128-320	$128 * (R+1)$	$192+16*(R-12)$ (8x128)	$180+15*(R-12)$	$192+16*(R-12)$ (128bit)	$192 + 16 * (R - 12)$ (128bit)	$192 + 16 * (R - 12)$	$6144 + 512 * (R - 12)$	0	32	no
Camellia	128	8	128	1664	144(8x64)	130	2(128bit), 162(64bit), 8(32bit)	144(64bit), 4(32bit), 4(32bit)	144	2544	0	8.2	no
Camellia	128	8	192/256	2176	192(8x64)	174	2(128bit), 216(64bit), 12(32bit)	192(64bit), 6(32bit), 6(32bit)	192	3392	0	8.2	no
Grandcru	128	8	128	4224	50(5x8), 672(8x8)	144(8bit), 160(8bit)	1372(8bit), 32(8bit)	144(8bit)	722	1548	0	16	no
Hierocrypt-3	128	8	128	1792	16(8x8), 96(8x32), 80(8x128)	144(32bit)	120(32bit), 76(128bit)	192(32bit)	182	2464	0	15	no
Hierocrypt-3	128	8	192	2048	16(8x8), 112(8x32), 96(8x128)	168(32bit)	140(32bit), 91(128bit)	224(32bit)	224	2912	0	15	no
Hierocrypt-3	128	8	256	2304	16(8x8), 128(8x32), 112(8x128)	192(32bit)	160(32bit), 106(128bit)	256(32bit)	256	3360	0	15	no
Noekeon	128	32	128	2048		128	16(128bit), 304(32bit)	32(32bit), 32(32bit), 32(32bit)	0	1856	0	10.5	no
Nush128	128	32	128, 192, 256	4608		68(32bit)	212(32bit)	68(32bit)	0	1120	0	15	yes
Q (bit-sliced)	128	32		1280	128(8x8)	24	26(128bit), 144(32bit)	56(32bit) 16(32bit)	128	1280	0	11.3	no
RC6 (default)	128	32	128	1408		40(32bit), 80(32bit) +40(32bit) multiplications	40(32bit), 84(32bit)		0	496	160	8	no
Safer++/128	128	8	128	1920	112(8x8)		15(64bit), 456(8bit)		112	576	0	35	yes, for 8-bit and 32-bit
Safer++/256	128	8	256	2688	160(8x8)		21(64bit), 648(8bit)		160	816	0	35	yes, for 8-bit and 32-bit
SC2000 (128)	128	32	128	1792	24(10x10), 48(11x11)	48(32bit)	87(32bit), 12(64bit), 14(128bit)	145(32bit)	72	668	0	20	no
SC2000 (192-256)	128	32	192-256	2048	28(10x10), 56(11x11)	56(32bit)	98(32bit), 14(64bit), 16(128bit)	168(32bit)	84	760	0	20	no
Rijndael	128	8	128	1408	160(8x32)	30	11(128bit), 120(32bit)		160	656	0		n. a.
SHACAL-1	160	32	512	2560		224(32bit)	272(32bit), 320(32bit)	180(32bit)	0	3088	0	8	no

Table 3: Stream Ciphers Submitted to NESSIE, Along with Stream Ciphers Whose Performance Measured by NESSIE

Name of Cipher	Submitted Keysizes		IV size
	Normal	High	
BMGL	128		–
BMGL with IV	128		128
Snow	128	256	–
Snow with IV	128	256	64
Sober-t16	128	256	–
Sober-t16 with IV	128	256	128
Sober-t32	128	256	–
Sober-t32 with IV	128	256	128
Leviathan	128, 192	256	–
Lili	128		–
RC4	128		–
Scream	128		128
Seal	160		32

- LEVIATHAN
- SOBER-t16 and SOBER-t32
- SNOW
- BMGL

Table 3 lists the stream ciphers, the security levels and key sizes. The top part lists the ciphers studied in the second phase of NESSIE, the middle part the other ciphers submitted to NESSIE, and the bottom part other ciphers to which we measured performance.

A summary of the theoretical results is given in Tables 4 and 5.

More details about them can be found in D14 [51].

We note that the NESSIE call for primitives specified the following security levels for stream ciphers.

- *High.* Key length of at least 256 bits. Internal memory of at least 256 bits.
- *Normal.* Key length of at least 128 bits. Internal memory of at least 128 bits.

4.1.3 Collision Resistant and one-way Hash Functions

The candidates in this category:

Table 4: Stream Ciphers Template Results: Key Setup

Key Setup	**Table Lookups**	**Shifts**	**Rotations**	**And, Or, Xor**	**Add**
BMGL (8-bit word)	560	0	0	0	544
Snow128 (32-bit word)	256	268	64	736	158
Snow256 (32-bit word)	256	280	64	768	174
Sober-t16/128 (16-bit word)	371	70	0	189	212
Sober-t32/128 (32-bit word)	371	78	0	253	242
LEVIATHAN 128 or 256					
LILI-128					
RC4	256($\log_2$(bytekey-length) x8bit), 512(8x8bit), 512 table storages (8x8bit)	0	0	0	512

Table 5: Stream Ciphers Template Results: Key Stream Generation

Key stream Generation	**Table lookups**	**Shifts**	**Rotations**	**And, Or, Xor**	**Add**	**Bits output/LSFR cycle**
BMGL (32bit word)	160	80	0	224	20	32
Snow128 (32bit word)	64	112	16	224	77	32
Snow256 (32bit word)	64	112	16	224	77	32
Sober-t16/128 (16bit word)	138	39	0	156	63	16
Sober-t32/128 (32bit word)	288	111	0	262	119	32
LEVIATHAN 128 or 256						
LILI-128						
RC4	3(8x8bit), 2(8x8bit) table storages	0	0	0	3	8[1]

[1]This number measures bits/stream cipher cycle since there is no LFSR for RC4.

Table 6: Hash functions Submitted to NESSIE, Along with Hash Functions Whose Performance was Measured by NESSIE

Name of Primitive	Hash Sizes
Whirlpool	512
MD4	128
MD5	128
RIPEMD	160
SHA-0	160
SHA-1	160
SHA-2	256,384,512
Tiger	192
BCHASH-Rijndael	128

- Whirlpool
- SHA-1
- SHA-2/256
- SHA-2/384
- SHA-2/512

Table 6 lists the hash functions, the security levels and hash sizes. The top part lists the ciphers studied by NESSIE, and the bottom part other ciphers to which we measured performance. Note that BCHASH-Rijndael is the cipher that results by replacing the internals of the compression function of SHA-1 by Rijndael (leaving the feed-forward mixing untouched), i.e., by replacing the SHACAL-1 part of SHA-1 by Rijndael (or in other words BCHASH is defined such that SHA-1 = BCHASH-SHACAL-1).

A summary of the theoretical results is given in Table 8. More details can be found in D14 [51].

4.1.4 Message Authentication Codes

There are two candidates in this category:

- Two-Track-MAC
- UMAC

Table 7: MACs Submitted to NESSIE, Along with MACs Whose Performance was Measured by NESSIE

Name of Primitive	MAC Sizes	Keysizes
Ttmac	160	160
Umac	64	160
HMAC-Whirlpool	512	512
HMAC-MD4	128	512
HMAC-MD5	128	512
HMAC-RIPE-MD	160	512
HMAC-SHA-0	160	512
HMAC-SHA-1	160	512
HMAC-SHA-2	256, 384, 512	512
HMAC-Tiger	192	512
CBCMAC-Rijndael	128	128
CBCMAC-DES	64	56
CBCMAC-Shacal	512	160

Table 7 lists the message authentication codes (MACs), the security levels and key sizes. The top part lists the ciphers studied by NESSIE, and the bottom part other ciphers to which we measured performance. The implemented CBCMAC is as defined by Algorithm 2 of the ISO/IEC 9797-1 [25], i.e., the message is padded by one 1 bit, followed by as many 0's as required to fill the last block. Then, the result is CBC-encrypted under the CBCMAC key, and the last block of the ciphertext is encrypted again under the key reached by XORing all bytes of the CBCMAC key with $F0_x$. The MAC value is the result of this last encryption.

A summary of the theoretical results is given in Table 8. More details can be found in D14 [51].

4.1.5 Asymmetric Encryption Schemes

The primitives in this category are:

- ACE Encrypt – tweaked to ACE-KEM
- ECIES
- EPOC-1
- EPOC-2 (tweaked)
- EPOC-3

Table 8: Hash Functions and Message Authentication Codes Template Results

Theoretical Analysis	**Word size (bits)**	**Subkey size (bits)**	**Table lookups/ Table size (bits×bits)**	**Shifts/ Rotations + multiplications**	**XOR, ADD, MULT (bit size)**	**AND, OR, NOT**	**Total table lookups**	**Total logical operations (8-bit)**	**Total multiplications (8-bit)**	**Code size (kB)**	**Optimised in submission?**
Whirlpool	8	5120	1280(8x64)	1188	1175(64bit)	1268 (64bit)	1280	19544	0	65	no
SHA-1	32			0/224 (32bit)	312 (32bit), 325 (32bit)	100 (32bit)	0	2948	0		
SHA-2/256	32			96 (32bit)/ 576 (32bit)	640 (32bit), 600 (32bit)	384 (32bit)	0	6496	0		
SHA-2/384, SHA-2/512	64			128 (64bit)/ 736 (64bit)	816 (64bit), 760 (64bit)	480 (64bit)	0	16448	0		
Umac16	16	8192			2048(16bit), 1024(32bit), 1024(32bit mult.)		0	8192	4096	146	no
Umac32	32	8192			512(32bit), 256(64bit), 256(64bit mult.)		0	4096	2048	146	no
Two-Track-MAC	32	160		320	128(32bit), 613(32bit)	384 (32bit)	0	4500	0	13	no

- PSEC-1
- PSEC-2 – tweaked to PSEC-KEM
- PSEC-3
- RSA-OAEP
- RSA-KEM

More details can be found in D14 [51].

Table 9 contains the practical parameter length and the ciphertext length for each scheme. The parameters are chosen such that the security of all the schemes are considered equivalent. For the schemes based on the RSA problem, the modulus is a 128-byte integer, for schemes based on the Diffie-Hellman problem, the finite cyclic group is Z/PZ^* for P of length 128 bytes, for those based on the elliptic curve Diffie-Hellman problem, the field size and the size of the order of the base point are 20 bytes. For Diffie-Hellman based schemes, the decryption of the group and the base point are not considered to be part of the key. The private key also includes the public parameters needed to decrypt. Sometimes, the key size can be reduced, but the encryption/decryption time will be increased by some preprocessing. This reduced number is in parenthesis.

Table 9: Usual Parameter Lengths (in Bytes)

Scheme	Public Key Length	Private Key Length	Ciphertext Length for 16-Byte Messages
ACE-Encrypt	1348	160	432
ECIES	20	20	36
EPOC-1	432 (288)	144	160
EPOC-2	432 (288)	144	160
EPOC-2 - tweaked	432 (288)	144	160
EPOC-3	432 (288)	144	160
PSEC-1	20	20	60
PSEC-2	20	20	76
PSEC-3	20	20	96
RSA-OAEP	129	256 (129)	128
ACE-KEM over Z_p^*	512	80	400
ACE-KEM over EC	80	80	76
PSEC-KEM	20	20	52

* For elliptic-curve-based schemes we assume that a point-compression technique is used

The table also contains the ciphertext size corresponding to messages of size 16 bytes. We consider that the output of the hash functions is 20 bytes (160 bits), the output of the symmetric encryption scheme used in the EPOC and PSEC schemes is considered as 16 bytes (128 bits), the length of the random string (salt) is 16 bytes.

Table 10 describes the number of group operations required by each scheme.

4.1.6 Asymmetric Digital Signature Schemes

The primitives in this category are:

- ACE Sign
- ECDSA
- ESIGN – tweaked to ESIGN-D
- RSA-PSS
- FLASH
- SFLASH – tweaked to SFLASHv2
- QUARTZ – tweaked

Table 10: Number of Group Operations for Each Asymmetric Encryption Scheme

	RSA		EPOC-2		RSA-OAEP		ECIES		PSEC-KEM		ACE-KEM	
	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC
group exponentiations	1	1	2	3	1	1	2	1	2	2	5	3
group multiplications	–	–	1	2	–	–	–	–	–	–	1	–
random numbers	–	–	1	–	1	–	1	–	1	–	1	–
hash calls [1]	–	–	3	3	3	3	1	1	2	2	2	2
symmetric cipher calls	–	–	1	1	–	–	1	1	1	1	1	1
MAC calls	–	–	–	–	–	–	1	1	–	–	–	–

[1]This includes calls to functions that predominantly rely on hash functions, including key derivation functions (KDFs) and mask generating functions (MGFs).

More details about them can be found in D14 [51].

Table 11 and Figure 1 lists the sizes of the data handled by the algorithms. The most important considerations are the signature size, which can cause an overhead for a message transmission, and the public key size, which has to be handled by the public key infrastructure. When (part of) the public key is needed to sign, we include it in the private key length.

Most submitted signature schemes have their security and performance dependent upon the choice of some parameter. For this table, the choices have been 1024 bit moduli for two factors based schemes (ACE Sign and RSA-PSS), 1152 bit moduli for three factors based schemes (ESIGN), and 163 bit base field for discrete logarithm based schemes (ECDSA). The RSA public exponent is fixed to 3.

Most of the submitted schemes use a hash function which gives a limitation on the message length, e.g., 2^{61} bytes for SHA-1. Only ACE-Sign has signature length depending on the message length. We assume that the messages to be signed have length between 20 bytes and 1 Mbyte. The signature length of ACE-sign can be made constant using CRHF instead of using UOWHF.

4.1.7 Asymmetric Identification Schemes

The candidates in this category are:

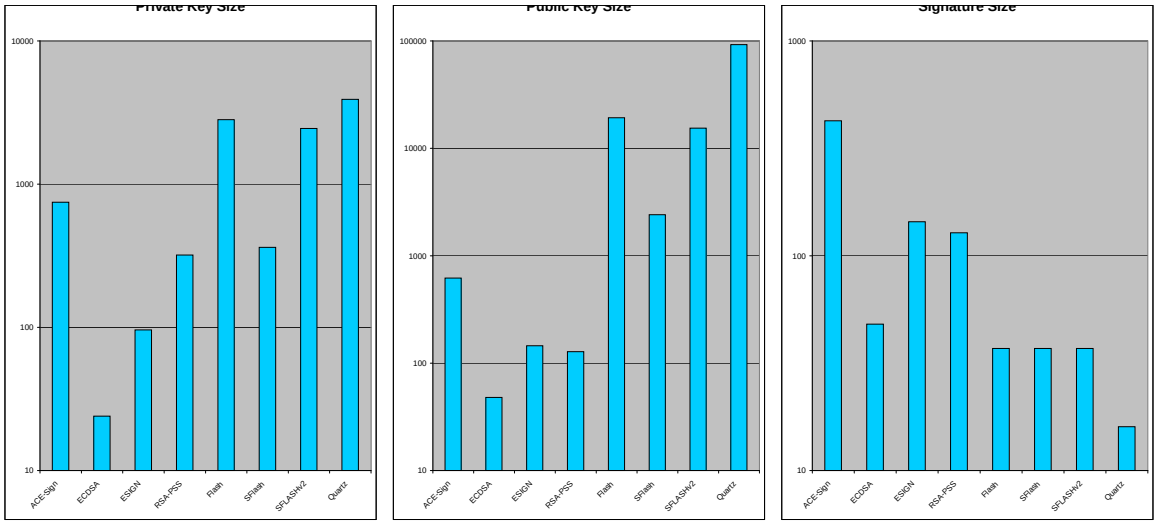
- GPS
- GPS – tweaked

More details can be found in D14 [51].

Table 11: Data Lengths

Scheme	Public Key	Private key	Signature
ACE-Sign	620 bytes	748 bytes	425 to 705 bytes
ECDSA	48 bytes	24 bytes	48 bytes (can be reduced to 326 bits)
ESIGN	145 bytes	96 bytes	144 bytes
RSA-PSS	128 bytes	320 bytes	128 bytes
Flash	19266 bytes	2822 bytes	37 bytes
SFlash	2409 bytes	362 bytes	37 bytes (can be reduced to 259 bits)
SFLASHv2	15400 bytes	2450 bytes	37 bytes (can be reduced to 259 bits)
Quartz	71 to 90 kB	3914 bytes	16 bytes
Quartz - tweaked	71 to 90 kB	3914 bytes	16 bytes
KCDSA	48 bytes	24 bytes	48 bytes (can be reduced to 326 bits)

Figure 1: Data Lengths



5 Claimed Performance

We present here performance results, concentrating mainly on the C code running on a Pentium III.

5.1 Block Ciphers - Legacy

5.1.1 CS-Cipher

Processor	Encryption (1) (cycles/block)	Encryption (cycles/byte)
Pentium	4053.12	506.64

(1) efficiency results for standard C implementation on a Pentium 133MHz.

5.1.2 Hierocrypt-L1

Processor	Encryption/(1) Decryption (cycles)	Encryption/ Decryption (cycles/byte)	Encryption/ Decryption (cycles/byte)	Encryption/ Decryption (cycles/byte)
Pentium	199	374	24.8	46.7

(1) best efficiency results in 10 trials to carry out 1000000 block encryptions in ECB mode for 128-bit key version on Pentium III 650 MHz, running MS Windows 98 CE, and MSVC++ 6.0. Data obtained from Proceedings of the 2nd NESSIE Workshop.

5.1.3 IDEA

We give here the performance claimed by the submitters for their candidate.

Processor	Encryption/ Decryption (Mbits/sec)	Encryption/ Decryption (cycles/byte)
90 MHz Pentium	4.2	171.4
366 MHz Pentium II	31	94.4
600 MHz Pentium III	61	78.7

In the following we present the results given by Lipmaa, who uses MMX and encrypts 4 blocks in parallel (i.e., CBC mode cannot be used with this speed).

Processor	Encryption/ Decryption Block size	Encryption/ Decryption (cycles)	Encryption/ Decryption (MBytes/sec)	Encryption/ Decryption (cycles/byte)
Pentium III (MMX)	4x64	440	55.5	13.75
Pentium MMX	4x64	543	45.0	16.9
Pentium MMX	64	358	17.0	44.7

5.1.4 Khazad and its Tweaked Variant

Khazad is an involutional cipher, so the effort required for encryption and decryption is the same. Because of this, decryption is not obtained by applying the encryption components in reverse order, and so the key schedules for encryption and decryption are not identical. The tweaked variant should have the same performance as the original variant.

Processor	Language	Key Setup (cycles/key)	Encryption/ Decryption (cycles/byte)	Encryption/ Decryption (Mbits/sec)
Pentium III (1)	ANSI C	717(encrypt) 1206(decrypt)	67	65.7

(1) IBM PC/ AT compatible PC, Intel Pentium III, 550 MHz.

5.1.5 MISTY1

Processor	Language	Key Setup (cycle/key)	Encryption (cycle/byte)	Encryption (Mbits/sec)
Pentium II (1)	ANSI C	170	56.25	71.1
Pentium II (2)	ANSI C	300	37.5	106.7

(1) IBM PC/ AT compatible PC, Intel Pentium II, 500 MHz, 128 MB memory, Windows 98, straightforward implementation.
(2) IBM PC/ AT compatible PC, Intel Pentium II, 500 MHz, 128 MB memory, Windows 98, optimized implementation requiring more memory.

5.1.6 NUSH

Processor	Encryption (1) (cycles/block)	Key Setup (cycles/key)	Encryption (cycles/byte)
Pentium	180	64	22.5

(1) efficiency results for C implementation.

5.1.7 SAFER++:

Processor	Key Setup(2) (cycles/key)	Encryption(1) (cycles/byte)	Decryption (cycles/byte)
Pentium	1333	169	160

(1) efficiency results for 128-bit key version on Pentium III 667 MHz, running MS Windows 2000, and MSVC++ 6.0 (optimized source code).
(2) efficiency of reference implementation (not optimized).

5.1.8 DES

The performance of DES has been well studied in the past, we give here some results to compare with the NESSIE candidates.

Origin	Processor	Encryption (cycles/byte)	(Mbits/sec)	(MBytes/sec)
Bosselaers	90 MHz Pentium	42.5	16.9	2.12

5.2 Block Ciphers - Normal and High (128 bit blocks)

5.2.1 Anubis:

Anubis is an involutional cipher, so the effort required for encryption and decryption is the same. Because of this, decryption is not obtained by applying the encryption components in reverse order, and so the key schedules for encryption and decryption are not identical.

Processor	Language	Key size	Key Setup (cycles/key)	Encryption/ Decryption (cycles/byte)	Encryption/ Decryption (Mbits/sec)
Pentium III (1)	ANSI C	128	3352(encrypt) 4527(decrypt)	36.8	119.5
		160	4445(encrypt) 5709(decrypt)	39.3	112.1
		192	6644(encrypt) 8008(decrypt)	41.6	105.9
		224	8129(encrypt) 9576(decrypt)	43.8	100.5
		256	9697(encrypt) 11264(decrypt)	46.3	95.1
		288	11385(encrypt) 11291(decrypt)	48.5	90.7
		320	13475(encrypt) 15169(decrypt)	50.8	86.6

(1) Intel Pentium III, 550 MHz.

5.2.2 Camellia:

Processor	Language	Key size	Key Setup (cycles/key)	Encryption/ (cycles/byte)	Encryption/ (Mbits/sec)
Pentium III (1)	Assembly	128	160	23.1	241.5
Pentium III (1)	Assembly	192	222	30.8	181
Pentium III (1)	Assembly	256	226	30.9	181
Pentium II (2)	ANSI C	128	263	36	66.6

(1) IBM PC/ AT compatible PC, Intel Pentium III (700MHz), 256KB on L2 cache, FreeBSD 4.0R, 128MB main memory.

(2) IBM PC/ AT compatible PC, Intel Pentium II (300MHz), 512KB L2 cache, Windows95, 160MB main memory.

5.2.3 Grand Cru

Processor	Encryption (1) (cycles/byte)	Decryption (cycles/byte)	Key Setup (cycles/key)
Pentium	2812.5	4062.5	200000
XEON	4062.5	5625	300000

(1) efficiency results of reference ANSI C implementation for 128-bit key version using gcc on an Intel Pentium 200 MHz, running Linux 2.0.38, and on an Intel XEON 550 MHz, running Linux 2.2.14.

5.2.4 Hierocrypt-3:

Processor	Key size	Encryption (1) (cycles/byte)	Decryption unpotimized	Key Setup (cycles/key)
Pentium	128	37.5	63.2	370
	192	44.4	78.1	386
	256	50.5	88.7	468

(1) best efficiency results in 10 trials to carry out 1000000 block encryptions in ECB mode for 128-bit key version on Pentium III 650 MHz, running MS Windows 98 CE, and MSVC++ 6.0. Data obtained from Proceedings of the Second NESSIE Workshop.

5.2.5 Noekeon

Note that Noekeon has two key schedules. In direct mode the user-selected key is used directly, and so requires no operations before encryption or decryption can start. In indirect mode the user-selected key is encrypted once using the all-zero key, and the ciphertext becomes the working key. This key schedule thus takes the same time as one encryption.

Processor	Language	Key Setup (cycles)	Encryption Decryption (cycles/byte)	Encryption Decryption (Mbits/sec)
Pentium II (1)	ANSI C	0(direct) 525(indirect)	32.8	48.8

(1) IBM PC/ AT compatible PC, Intel Pentium II, 200 MHz, Windows NT 4.0, Microsoft Visual C/C++ 6.0 compiler.

5.2.6 NUSH

Processor	Key Size (bits)	Encryption(1) (cycles/byte)	Key Setup (cycles/key)
Pentium	128	21.2	112

(1) efficiency results for C implementation.

5.2.7 Nimbus

Processor	Block Size	Encryption (cycles/byte)	Key Setup (cycles/key)
Pentium	64	66	24665

5.2.8 Q

Processor	Key Size	Encryption/Decryption (cycles/byte)	Key Setup (cycles/key)
Pentium	128	36.5	500

5.2.9 RC6

We present the performance claimed by the submitters of RC6, for the C code, in cycles/byte for 20-round and 128 bit-block version.

Processor	Compilator	Encryption/ Decryption (cycles/byte)
Pentium II	Borland	39
Pentium Pro	MSVC	30
Pentium Pro	MSVC + opt.	16
Pentium Pro	GCC	26
Pentium Pro	GCC + opt.	23

5.2.10 SAFER++:

Processor	Key size	Key Setup(2) (cycles/key)	Encryption(1) (cycles/byte)	Decryption (cycles/byte)
Pentium	128	2333	40	76
	256	3227	57	101

(1) efficiency results for 128-bit key version on Pentium III 667 MHz, running MS Windows 2000, and MSVC++ 6.0 (optimized source code).
(2) efficiency of reference implementation (not optimized).

5.2.11 SC2000: 128-bit Keys

Processor	Key size	Key Setup (cycles/key)	Encryption (1) (cycles/byte)	Decryption (cycles/block)
Pentium	128	488	23.93(assembly)	25.187(assembly)
	192	525	27.37(assembly)	28.75(assembly)
	256	526	27.37(assembly)	32.81(assembly)

(1) best efficiency results for 128-bit key version in MSVC++ 6.0 + Assembly (inline) on an Intel Pentium III 550 MHz, running Microsoft Windows NT 4.0; key schedule in C language.

5.2.12 Rijndael

Some performance results exist for this block cipher, which was well studied in the AES process. We give some significant results to compare with NESSIE candidates, in addition to our own tests.

Origin	Processor	Block size	Cycles/byte	MBytes/s
Lipmaa	Pentium II/III	128	28.6	53.3

5.2.13 SHACAL-1

Processor	Block size (bits)	Encryption (cycles/byte)	Decryption (cycles/byte)	Key setup (cycles/key)
Pentium II/III	160	140	116.5	3200
updated	160	124	116	2280

The SHACAL submitters estimate the computational efficiency at 2800 cycles per 20 bytes block encryption (block size), 2330 per 20 byte block decryption and 3200 per 64 byte key setup, on a PC platform with an AMD K6 processor running at 233 MHz.

5.2.14 SHACAL-2

Processor	Block size (bits)	Encryption (cycles/byte)	Decryption (cycles/byte)	Key setup (cycles/key)
Pentium II/III	160	112.5	115	2800

5.3 Stream Ciphers

5.3.1 BMGL

No claimed performance given.

5.3.2 Leviathan

No claimed performance given.

5.3.3 LILI

Processor	Key size (bits)	Key Setup (cycles/byte)	Key Setup (cycles/byte)
Pentium III 650MHz	128	–	1200

5.3.4 Snow

Processor	Key size (bits)	Key Setup (cycles/byte)	Key Setup (cycles/byte)
Pentium III 500MHz	128	2000	6.75
	256	2000	6.75

5.3.5 Snow with IV

Processor	Key size (bits)	Key Setup (cycles/byte)	IV Setup (cycles/byte)	Key Setup (cycles/byte)
Pentium III 500MHz	128	few	1000	6.75
	256	few	1000	6.75

5.3.6 Sober-t16

Processor	Key Size (bits)	Key Setup (cycles/key)	Key Setup (cycles/byte)
Sun Ultrasparc 248MHz	0	1084	42.6
	64	1346	42.6
	128	1581	42.6
	192	1812	42.6
	256	2062	42.6

For Sober-t16 with IV there are no claimed results for the IV setup

5.3.7 Sober-t32

Processor	Key size (bits)	Key Setup (cycles/key)	Key Setup (cycles/byte)
Sun Ultrasparc 248MHz	0	1110	24.51
	64	1208	24.51
	128	1348	24.51
	192	1480	24.51
	256	1564	24.51

For Sober-t32 with IV there are no claimed results for the IV setup

5.4 Hash Functions

5.4.1 Whirlpool

Processor	Hash (cycles/byte)
Pentium III 550MHz	133

Table 12: Claimed Performance of Asymmetric Encryption Schemes

Scheme	Key Setup		Encryption		Decryption		Architecture
	time	cycles	time	cycles	time	cycles	
ACE-Encrypt	14 sec	3724M	230 ms	61M	97 ms	26M	Pentium @ 266
PSEC-1			4.4 ms	3080K	4.4 ms	3080K	Pentium @ 700
PSEC-2			4.4 ms	3080K	4.4 ms	3080K	Pentium @ 700
PSEC-3			4.4 ms	3080K	2.4 ms	1680K	Pentium @ 700
ECIES			5.8 ms	1160K	4.4 ms	880K	Pentium @ 200
EPOC-1			13 ms	9100K	20 ms	14M	Pentium @ 700
EPOC-2			10 ms	7000K	17 ms	12M	Pentium @ 700
EPOC-3			10 ms	7000K	7 ms	4900K	Pentium @ 700
RSA-OAEP			1 ms	450K	27 ms	12M	Celeron @ 450
PSEC-KEM			4.4 ms	3080K	4.4 ms	3080K	Pentium @ 700

5.5 MACs

5.5.1 Two-Track MAC

Processor	MAC (1) (cycles/byte)	Key setup (cycles/key)
Pentium	95.8	10
assembly	16	10

(1) efficiency results for non-optimized reference C code.

5.5.2 UMAC

Processor	Message Size (bytes) (1)			
	43	256	1500	$256 \cdot 2^{10}$
Pentium	16.3	3.8	2.1	1.9

(1) efficiency results in cycles/byte on a 700 MHz Pentium III under gcc 2.95, mixed C/Assembly. Figures for UMAC32 submitted to NESSIE and called simply UMAC. Data obtained from Proceedings of First NESSIE Workshop.

5.6 Asymmetric Primitives

The claimed performance of the asymmetric encryption schemes is given in Table 12. Chinese remainders are used for RSA. No new claims have been made for the tweaks ACE-KEM, PSEC-KEM or EPOC-2. The changes for the latter two should not influence their performance. ACE-KEM is faster than ACE-Encrypt.

The claimed performance of the asymmetric digital signature schemes is given in Table 13.

The claimed performance of the asymmetric identification schemes is given in Table 14.

Table 13: Claimed Performance of Digital Signature Schemes

Scheme	Key Setup		Signature		Verification		Architecture
	time	cycles	time	cycles	time	cycles	
ACE-Sign	36 sec.	9576M	62 ms	16492K	73 ms	19418K	Pentium @ 266
ECDSA — GF(2^{163})	1.5 ms	600K	2.1 ms	840K	4.1 ms	1640K	Pentium @ 400
ECDSA — GF(p)	2.1 ms	840K	2.6 ms	1040K	6.5 ms	2600K	Pentium @ 400
ESIGN			3.4 ms	2980K	0.4 ms	280K	Pentium @ 700
RSA-PSS	1.9 sec	855M	27 ms	12.1M	3 ms	1350K	Celeron @ 450
Flash			49 ms	24.5M	50 ms	25M	Pentium @ 500
SFlash			44 ms	22M	50 ms	25M	Pentium @ 500
SFLASHv2*	1 sec	500M	2.7 ms	1350K	0.8 ms	400K	Pentium @ 500
Quartz			30 sec	15G	50 ms	25M	Pentium @ 500
Quartz-tweak*	4 sec	2000M	10 sec	5000M	0.9 ms	450K	Pentium @ 500

* The difference of times between SFLASH and SFLASHv2 is due to the use of a more efficient implementation of the submitters in their tweak claims than they had in their original submission. So is also the case for Quartz and it's tweak.

Table 14: Claimed Performance of Asymmetric Identification Schemes

Scheme	Key Setup		Commitment		Answer		Verification		Architecture
	time	cycles	time	cycles	time	cycles	time	cycles	
GPS	0.7 sec	315M	10 ms	4500K	1 μs	450	12 ms	5400K	Pentium @ 450
GPS-tweaked	0.7 sec	315M	10 ms	4500K	1 μs	450	12 ms	5400K	Pentium @ 450

6 Practical Software Implementation

We tested the performance of 285 implementations of 138 different variants of primitives. All the tests were performed on 11 different kinds of platforms (on over 20 computers), on various operating systems and compilers. On some processors (e.g, Pentiums) we made the tests on two operating systems with 4 compilers, and several different versions of some compilers, in order to achieve the best results. In total our performance tests run several thousands of computer hours.

6.1 Measurements for Symmetric Primitives

We tested all NESSIE symmetric candidates along with standard primitives and many 'non-standard' primitives. The tool measures the key setup time, times with their encryption time, decryption time, IV setup times, and hash and MAC initializations and finalizations. The tool checks the correctness of all codes by comparing encryption results to the supplied test vectors. We measured the time by:

- First, encrypting (decrypting, key setup, etc.) random plaintexts for about one second. Based on the number of plaintexts encrypted in one second we estimate how many encryptions (decryptions, key setups, etc.) are expected to run in 10 seconds.
- Then, running the estimated number of encryptions (decryptions, key setups, etc.) while measuring their run time.

The actual measurement is executed with many different keys on many different encryption/decryption blocks. By measuring the time we calculate the encryption, decryption, hash, MAC time in units of *cycles/byte* and the key setup, IV setup times and hash and MAC initializations and finalizations in *cycles/invocation*.

We compare the results on various machines, as listed in Table 29. The speed results are given in Tables 30, 31, 32, 33, 34 and 35 divided by the types of processors and operating systems, where Table 30 summarizes the results for legacy block ciphers, Table 31 for normal and high block ciphers, Table 32 for block ciphers with 160 and 256-bit blocks, Table 33 for stream ciphers, Table 34 for hash functions, and Table 35 for MACs. Tables 39–48 elaborate by presenting these results divided by (cores of) processors and compilers. Figures 2–23 show these results as bar histograms, and Figures 24–45 show these results in a sorted order. Each table is divided to three parts, the first part contains the candidates accepted to the second phase of NESSIE, the second part contains the other candidates submitted to NESSIE, and the third part contains primitives that we implemented and compared, but were not submitted to NESSIE; the (unsorted) figures are also shown in the same order. The only exception is Table 33 (stream ciphers) which has an additional division to ciphers which do not accept initial values, and ciphers that require initial values. Our results show very high consistency between different machines of the same type, especially between various PIII's (at different speeds and memory sizes). In many cases they gave clock counts with differences of less than one unit.

For the measurement of speed we compiled all ciphers under all the available compilers, with various optimization options (as adequate for the machine and compiler), and selected the best speed that resulted from all these options. In many cases "higher" optimizations (such as -O3) resulted in poorer speeds as opposed to lesser optimizations (such as -O1), and in many cases optimizations targeted to older processors (such as optimization targeted for 386 when running on PIII) gave better results than optimizations targeted to the newer (such as Pentium or Pentium-pro). For this reason, on most machines, our measurements consisted of more than a dozen compilations with different optimization options and target machine, to ensure that we do not miss the best code that the compiler can generate. In the case of PIII with Linux, we performed the measurement under three different versions of the gcc compiler, with over 40 different optimization options for the newer version.

We also ensured that the compiled code is correct by regenerating the test vectors in each run with each compilation option. In those rare cases that some compilation option generated wrong code on some machine, we ignored the speed results of the runs with the wrong results and did not use them for this comparison. For example, the compilations of Anubis on Alpha machines generated wrong codes when the optimization options were -O2 or higher. We ignored the resultant speeds of these wrong runs, although they were faster than the speeds we list in this document.

We also ensured that the main test program is compiled with the same optimization option in all cases, although the code of the primitives was compiled with different options, in order to make the overhead of the test program as fixed as possible. In order to measure this overhead, we measured the speed of dummy ciphers (that do nothing) and verified that their computation time is negligible. It should be noted that all codes (of each family of primitives) use the same API (which between other things ensures that the keys are set up into structures that are later passed as parameters to the encryption (decryption, etc.) function, and that no global nor static variables that depend on the key, state, or data are used), and thus the overhead of all codes of the same type and block size are expected to be similar.

It can be seen from the results that the codes for the primitives that we have is quite optimized. This is a result of several rounds of optimizations of the submitted codes by several people in the NESSIE project. In about half of the ciphers, our codes are even faster than the claims of the submitters, and in several others the results are only a few cycles slower than the claims. In particular, we wish to refer the reader to our optimizations of Camellia, which uncovers the design of Camellia, whose round function is designed for 32-bit processors, although the description in the submission describes only a byte-level implementation. We also wish to mention our novel optimization of Idea [9] which is to the best of our knowledge the fastest implementation of Idea in C.

In most cases the order of speeds of the primitives is similar on all machines. Exceptions are primitives that are optimized for 64-bit machines, which become the fastest on Alpha, although they are not so on other machines. Two examples are RC6 with 256-bit blocks (using 64-bit multiplications) which is even twice faster on Alpha than the standard RC6 with 128-bit blocks, although it is twice slower on all other machines, and Tiger which becomes even faster than MD4, although it has only a medium speed on all other machines.

Note that the same implementations of block ciphers and stream ciphers were also subjected to the NESSIE statistical tests, and all of them passed these tests.

We have also measured the amount of memory required for the various implementations, and verified that the speeds reported herein can be reached with a reasonable amount of memory.

We did not distinguish the cases where the key setup can be faster for encryption-only or decryption-only applications. In all cases we measured the time required for a full key setup. In the cases of Idea, Rijndael, and several others, the time required to setup encryption-only keys may be significantly faster than the full time of the key setup.

Finally, the test vectors that we used for verifying correctness of our codes were sent to the submitters for verification. In Table 15 we list the status of the test vectors at the time of writing.

Finally, a primitive is listed with two implementations in cases where there are inherent tradeoffs between the listed speeds, e.g., when encryption can be made faster with a slower key schedule, or when one implementation uses a feature that is usually unused in our implementations (e.g., MMX instructions).

6.1.1 Assembly Results External to NESSIE

The NESSIE project did not implement the primitives in assembly languages, except for a few MMX-optimized codes that were written in the C files and processed by the C compilers. For the selection of primitives it was not necessary to know the exact speedup gained by assembly code, and the average saving of a few cycles per byte did not worth directing our cryptanalytic efforts toward additional implementation work. For completeness, we bring in Table 16 the claimed speeds of implementations external to NESSIE.

6.2 Measurements for Asymmetric Primitives

We chose the smallest parameters that the submissions claimed to have a security of 2^{80}, i.e., 1024 or 1152 bits for factorization or discrete log based schemes, and 160 bits for elliptic curves. We use an RSA public exponent is 3, Type B parameters for (untweaked) EPOC, elliptic curves over the binary prime field GF(2^{163}) for ECIES and ECDSA and over a prime field for PSEC. Note that OEF (Optimal Extension Fields) are more efficient but are believed to weaken the discrete logarithm assumption. (For Epoc-2-tweaked a different set of parameters is required).

The measurement was performed on the same test suite and the same processors, operating systems and compilers as the symmetric but unlike in their case, not all the primitives were implemented, due to the inherent slow speeds of these primitives and the fact that it is easy to derive approximate times for some primitives from the times of others (such as when the key generation for several primitives are the same, or where the most time-consuming operation is common in several primitives, as in the case of modular exponentiation that dominates the computation time). However, note that our asymmetric implementations were not optimized, and should be treated only for order of magnitude.

Table 15: Status of the Test Vectors of the symmetric primitives

	Verified by Submitters	**Compared to Submited Vectors**	**Unknown**	**Non-Nessie Verified by Designers**	**Non-Nessie Verified by Us**	**Unverified**
Block Ciphers	Anubis Camellia Idea Khazad Hierocrypt-L1 Hierocrypt-3 Misty1 Noekeon Nush Safer++ Shacal RC6	Cs-cipher Grandcru Nimbus Q SC2000		Kasumi Serpent CAST-128 RC5	DES Triple-DES Rijndael Mars Twofish Skipjack	Seed
Stream Ciphers	BMGL Snow Sober		Leviathan Lili	Scream-S RC4	Scream-0/F Seal	
Hash Functions	Whirlpool			Tiger	SHA-0 SHA-1 SHA-2 MD4, MD5	RIPEMD BCHASH-Rijndael**
MACs	Ttmac UMAC			HMAC-Tiger	CBCMAC-DES	HMAC-Whirlpool** HMAC-SHA-0** HMAC-SHA-1* HMAC-SHA-2** HMAC-MD4* HMAC-MD5* HMAC-RIPE-MD** CBCMAC-Rijndael** CBCMAC-Shacal**

* Verified consistency of several implementations, and relative to the underlying hash function

*∗ Verified relative to the underlying primitive

Table 16: Assembly Performance Results

Primitive	**Machine**	**Speed (cycles/byte)**		**Reference**
		Encryption	Decryption	
Camellia	Sparc	22.2	22.2	[13]
	Alpha	17.6	17.6	[13]
4-Way IDEA	PIII	13.75		[38]
	PI/MMX	16.96		[38]
Misty1	PII	26.6	26	[13]
	Alpha	25.4	25.8	[13]
RC6	P.ProII	16		[54]
	Pentium	44		[54]
	PIII	22.18		[39]
Rijndael	PIII	14.13	14.93	[39]
	PIII	14.8		[13]
	P.ProII	18		[54]
	Pentium	20		[54]
Mars	P.ProIII	20		[54]
	Pentium	34		[54]
Twofish	P.ProII	16		[54]
	Pentium	19		[54]
Whirlpool	PIII/MMX	36.52		[42]
SHA-1	PIII/MMX	8.3		[42]
SHA-2/256	PIII/MMX	20.59		[42]
SHA-2/512	PIII/MMX	40.18		[42]
MD5	PIII/MMX	4.31		[42]
RIPEMD-160	PIII/MMX	11.34		[42]
UMAC	PII	1.93		[10]
	PPC	1.58		[10]
	Alpha	2.78		[10]

Table 17: Estimated Performance of Asymmetric Encryption Schemes

Scheme	Encryption		Decryption	
	ms	cycles	ms	cycles
ACE-Encrypt	50 ms	25M	45 ms	22.5M
PSEC-1	5 ms	2500K	5 ms	2500K
PSEC-2	5 ms	2500K	5 ms	2500K
PSEC-3	5 ms	2500K	2.5 ms	1250K
ECIES	5 ms	2500K	2.5 ms	1250K
EPOC-1	15 ms	7.5M	20 ms	10M
EPOC-2	10 ms	5M	15 ms	7.5M
EPOC-2 - tweaked	10 ms	5M	15 ms	7.5M
EPOC-3	10 ms	5M	7 ms	3.5M
RSA-OAEP (e=3)	1 ms	500K	11 ms	5.5M
ACE-KEM over $mZ_{P}*$	50 ms	25M	35 ms	17.5M
ACE-KEM over EC	12.5 ms	6250K	7.5 ms	3750K
PSEC-KEM	5 ms	2500K	5 ms	2500K
Rabin-SAEP	0.5 ms	250K	11 ms	5.5M

6.2.1 Asymmetric Encryption Schemes

The results of our performance test for asymmetric encryption schemes are given in Tables 36 and 49 where the top part lists results of (not necessarily optimized) codes of the NESSIE test suite, and the bottom part list results based on the submitters codes (without all the extra features and tests of the suite).

Table 17 shows estimations of performance for encryption and decryption on a Pentium III desktop, based on the theoretical analysis of Section 4. RSA-KEM has performance similar to RSA-OAEP.

6.2.2 Asymmetric Digital Signature Schemes

The results of our performance test for asymmetric digital signature schemes are given in Tables 37 and 50, where the top part lists results of (not necessarily optimized) codes of the NESSIE test suite, and the bottom part list results (mostly) based on the submitters codes (without all the extra features and tests of the suite).

We also tested the speed of invalid verifications, and found that in all the cases of Table 37 the times are about the same as of valid verifications.

	Machine I	Machine II
Parameters Generation	1.4 ms	1.5 ms
Commitment	9 ms	5 ms
Answer	0.018 ms	0.006 ms
Verification	11 ms	6.2 ms

Table 18: Performance of GPS in Java

6.2.3 Asymmetric Identification Schemes

The only indentification schemes submitted to NESSIE is GPS. GPS has five parts: the generation of the parameters, the generation of public and private keys, the commitment, the answer, and finally the verification. We used the parameters suggested in the submission which guarantee enough security, namely $|S| = 160$, $|B| = 35$, and $|A| = 275$.[1] We tested the performance of the submitters code, and got the results described in Tables 38 and 51 .

We also tested the performance of GPS in Java with the submitted Java code. We note that the code is really short with only about 2kB, including some short comment, but also uses external Java libraries. The results are summarized in Table 18. In this table, Machine I is a Pentium 550 MHz with 1GB of memory running under Windows 2000 and Machine II is a laptop Pentium 1GHz with 256 MB of memory and running under Windows 2000.

[1]With the notations of the submission, A/BS must be large enough to guarantee the statistical zero knowledge property, $|A| \geq |S| + |B| + 80$, which is the case with the chosen values.

7 Hardware and Smartcard Implementations

In addition to the implementations in software, the NESSIE project started to study implementation on hardware and smartcards. In this section we briefly describe the initial results of the NESSIE hardware and smartcard implementations.

7.1 Hardware Implementations

7.1.1 Previous Results

Table 19 lists known implementations external to the NESSIE project.

For block ciphers, the quoted results apply to encryption without key schedule. FPGA and ASIC implementations are shown together, despite their differences. This table is a collection of known results; primitives with no known results are marked by N/A. These results should not be considered as a fair comparison of the primitives, as each implementation was done with a different methodology, and the implementations were not evaluated by NESSIE.

7.1.2 Block Ciphers

Reprogrammable devices such as Field Programmable Gate Arrays (FPGA's) are highly attractive options for hardware implementations of encryption algorithms. Table 20 summarizes results of implementations for block ciphers MISTY1, KHAZAD, AES Rijndael and DES. These results were synthesized with FPGA Express (SYNOPSYS), implemented with XILINX ISE 4 within a VIRTEX1000 and they use no RAM blocks. Note that the pipeline version of Rijndael cannot fit into a VIRTEX1000. For more information see [60, 61, 62, 63, 64, 65]. Table 21 summarizes the results of our implementations of Rijndael on VIRTEX3200E.

7.1.3 Estimations for PSEC-KEM

In this section we give timing estimations for the PSEC-KEM [18] signature generation and verification algorithm on EC processor developed at KUL [50].

We describe below the 5 algorithms related to PSEC-KEM. The total latency to execute Algorithm 1 is basically the latency of the random number generation, which is $14.843ms$. The total latency to execute Algorithm 2 is $29.686ms$. The total latency to execute Algorithm 3 is $14.843ms$. Step 3. of Algorithm 4 takes $14.843ms$ and the latency of the other two steps are negligible when you compare with EC point multiplication.

7.1.4 Estimations for ECDSA

In this section we give timing estimations for the ECDSA [27] signature generation and verification algorithm on EC processor developed at KUL [50]. Most ECDSA computation time is devoted to 'modular reduction', 'modular multiplication', 'modular addition' and

Table 19: Previous Implementations in Hardware

Primitive	Type of Hardware	Speed	More Details
CS-cipher	ASIC 30MHz	73Mbps	estimation ; 1mm2
	ASIC 30MHz	2000Mbps	estimation ; 15mm2
Hierocrypt-L1	ASIC 0.14μ 128MHz	586Mbps	38.2Kgates
	FPGA 11MHz	44Mbps	11Klogic cells
IDEA	ASIC 0.25μ 100MHz	240Mbps	720 Mbps in ECB
Misty1	ASIC 0.35μ	72Mbps	7.6Kgates
	ASIC 0.35μ	800Mbps	50Kgates
Triple-DES	ASIC 0.35μ [4]	407Mbps	128Kgates
Camellia	ASIC 0.35μ [4]	1170Mbps	273Kgates
	ASIC 0.35μ [4]	220Mbps	11Kgates
	FPGA XC4000XLμ [4]	122Mbps	874CLBs
Hierocrypt-3	ASIC 0.14μ 126MHz	897Mbps	81.5Kgates
	ASIC 0.14μ 185MHz	85Mbps	26.7Kgates
	FPGA 7.4MHz	53Mbps	23Klogic cells
	FPGA 9MHz	4.1Mbps	6.3Klogic cells
RC6-128	DSP TMS320C6201	18cycles/byte	feedback
	DSP TMS320C6201	13cycles/byte	non-feedback
	DSP TMS320C64x	10cycles/byte	non-feedback
	FPGA XCV1000 14MHz	127Mbps	feedback
	FPGA XCV1000 38MHz	2400Mbps	non-feedback
	ASIC 0.5μ	104Mbps	iterative
	ASIC 0.5μ	2200Mbps	pipelined
	ASIC 0.35μ [4]	204Mbps	1.6Mgates
Rijndael-128	FPGA XCV1000 14MHz	300Mbps	feedback
	FPGA XCV1000 32MHz	1940Mbps	non-feedback
	ASIC 0.5μ	524Mbps	iterative ; 81mm2
	ASIC 0.5μ	5100Mbps	pipelined
	ASIC 0.35μ [4]	1950Mbps	613Kgates

Table 20: FPGA Implementations of Block Ciphers Within VIRTEX1000

Cipher	Nbr of slices	Output every (clk edges)	Estimated frequency(Mhz)	Throughput (Mbits/s)
MISTY1	6322	1	159	10176
Fast KHAZAD	8800	1	148	9472
Low area KHAZAD	7175	1	123	7872
Pipeline Rijndael*	17984	1	–	–
Sequential Rijndael	2257	5/52	127	1563
Pipelined DES	3681	1	175	11200
Sequential DES	189	1/18	176	626
Sequential 3-DES	604	1/18	165	587

* Does not fit into VIRTEX1000

Table 21: Rijndael Implementations on VIRTEX3200E

Type	Nbr of LUT	Nbr of reg.	Nbr of slices	RAM blocks	Latency (cycles)	Output every (cycles)	Freq. after Synt. (Mhz)	Freq. after Impl. (Mhz)
Pipeline Rijndael	4912	7792	5144	100	42	1	285	112
Pipeline Rijndael	4272	6832	4032	100	32	1	232	92
Pipeline Rijndael	3516	3840	2784	100	21	1	208	92
Sequential Rijndael	1036	1452	866	10	52	5/52	285	147
Sequential Rijndael	965	1372	739	10	42	4/42	232	135
Sequential Rijndael	877	932	550	10	31	3/31	208	117
Sequential Rijndael	877	668	542	10	21	2/21	208	119
Modified sequential	709	413	405	10	20	2/20	192	87

Algorithm 1 KGP-PSEC

Require: E: an elliptic curve parameter,
KDF: the choice from key derivation functions,
$hLen$: a nonnegative integer

Ensure: PK: PSEC public key,$(E, W, KDF, hLen)$,
s: PSEC private key, a nonnegative integer, $0 \leq s < p$

1: Generate a random integer $s \in 0, , p-1$.
2: Let $W := sP$.
3: $PK = (E, W, KDF, hLen)$ and s.

Algorithm 2 EP-PSEC

Require: PK: PSEC public key,
α: random value, a nonnegative integer, $0 \leq \alpha < p$
Ensure: Q: a point on E, C_1: a point on E
1: Let $Q := \alpha W$.
2: Let $C_1 := \alpha P$.
3: Output (Q, C_1).

Algorithm 3 DP-PSEC

Require: PK: PSEC public key,
C_1: a point on E,
s: PSEC private key, a nonnegative integer, $0 \leq s < p$
Ensure: Q: a point on E
1: Let $Q := sC_1$.
2: Q.

Algorithm 4 ES-PSEC-KEM-Encrypt

Require: PK: PSEC public key
Ensure: c_0 an octet string,
k an octet string
1: Let $(\alpha, k, r) := EME - PSEC - KEM - A(PK)$. (see [18] Section 7.1.1.)
2: Let $(Q, C_1) := EP - PSEC(PK, a)$. (see Algorithm 2
3: Let $c_0 := EME - PSEC - KEM - B(PK, Q, C_1, r)$. (see [18] Section 7.1.2.)
4: (c_0, k).

Algorithm 5 ES-PSEC-KEM-Decrypt

Require: PK: PSEC public key
s: PSEC private key, a nonnegative integer, $0 \leq s < p$
c_0: an octet string
Ensure: k': an octet string
1: Let $(C_1, c_2, g) := EME - PSEC - KEM - C(PK, c_0)$. (See [18] Section 7.1.3.) If the decoding operation returns "invalid", then assert "invalid" and stop.
2: Let $Q' := DP - PSEC(PK, C_1, s)$. (See [18] Section 5.3.)
3: Let $(\alpha', k') := EME - PSEC - KEM - D(PK, c_2, g, Q')$. (See [18] Section 7.1.4.)
4: Check $C_1 := DP - PSEC(PK, P, \alpha')$. (See [18] Section 5.3.) If it holds, output k'. Otherwise, assert "invalid" and stop.

'modular multiplicative inversion' operations. A generic arithmetic processor for these operations has been developed at K.U.Leuven, and the implementation of ECDSA uses this generic arithmetic processor.

If the elliptic curve is defined over $GF(p)$, we use elliptic curve processor (ECP) developed at KUL [50] for EC point multiplication and the other modular operations needed for signature generation and verification. If the elliptic curve is defined over $GF(2^m)$, we use Orlando and Paar's processor over $GF(2^m)$ [48] for EC point multiplication and ECP for modular operations. As a SHA-1 circuit, we use the data given by Helion Technology Lim. [24]. (For comparison of implementations of elliptic curve point multiplications see Table 22).

ECDSA Signature Generation: We give estimations on ECP for ECDSA signature generation algorithm starting from Step 2. To sign a message m, with domain parameters $D = (q, FR, a, b, G, n, h)$ and associated key pair (d, Q) the operations and the number of clock cycles to complete each operation on ECP are as follows:

1. Compute $kG = (x_1, y_1)$ and $r = x_1$: If q is a odd prime, $l_1(51l_2 + 66)$, $l_1 = \log_2 k$, $l_2 = \log_2 q$. If $q = 2^m$, m is odd prime, $10.5m^2 - 8.5m + 1.5\lfloor \log_2(m-1) \rfloor m$
2. Compute $k^{-1} \bmod n$: $9/2l_3^2 + 6l_3$, $l_3 = \log_2 n$.
3. Compute $e = SHA - 1(m)$: 889 clock cycles.
4. Compute $s = k^{-1}(e + dr) \bmod n$: $8l_3 + 9$.

If $l \approx l_1 \approx l_2 \approx l_3$ and q is a odd prime, total number of clock cycles for ECDSA signature generation is $55.5l^2 + 80l + 898$. If $m \approx l_2 \approx l_3$ and $q = 2^m$, m is odd prime, it is $15m^2 + 5.5m + 1.5\lfloor \log_2(m-1) \rfloor m + 898$

ECDSA Signature Verification: The operations used for verification of a signature and the number of clock cycles to complete each operation on ECP are as follows:

1. Compute $e = SHA - 1(m)$: 889 clock cycles.
2. Compute $w = s^{-1} \bmod n$: $9/2l_3^2 + 6l_3$.
3. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$: $6l_3 + 8$.
4. Compute $X = u_1G + u_2Q$: If q is a odd prime, $2l_3(51l_2 + 66) + 42l_2 + 56$. If $q = 2^m$, m is odd prime, $21m^2 - 6m + 3\lfloor \log_2(m-1) \rfloor m$.

If $l \approx l_2 \approx l_3$ and q is a odd prime, total number of clock cycles for ECDSA signature verification is $106.5l^2 + 186l + 953$.If $m \approx l_3$ and $q = 2^m$, m is odd prime, it is $25.5m^2 + 6m + 3\lfloor \log_2(m-1) \rfloor m + 897$.

The area and speed and FPGA resource values of the used circuits for estimations of ECDSA circuit are given in Table 23.

The times needed for signature generation and verification with $91.971MHz$ are given in Table 24.

Table 22: Area Utilization and Throughput of Known Hardware Implementations for Elliptic Curve Point Multiplication Over $GF(2^m)$

Multiplier	Device	Field	Digit Size	Frequency (MHz)	# Clk. Cyc. per Mult.	# Mult. per second	Throughput (bits/sec)	# gates	# CLBs	# FFs	# RAM
AMV93	VLSI	$GF(2^{310})$	32	40			444×10^3	11.000			
[3]		$GF(2^{155})$			275862	145	60×10^3				
AMV89	VLSI	$GF(2^{155})$	32	40	4444444	9		11.000			
[1]	M68008 controller										
AMV92 [2]	VLSI	$GF(2^{155})$	32	40	4310	580	200×10^3	11.000			6 Kbytes
SES98 [66]	XC4020XL	$GF(2^{155})$	32	10-15				17.000			
GSS99 [20]	XC4010XL	$GF(2^5)$			126	179856			272		
	XC4013XL	$GF(2^{11})$			825	19230			478		
	XC4028XL	$GF(2^{29})$			7158	1653			962		
	XC4044XL	$GF(2^{53})$			26753	417			1626		
Leung et al. [37]	XCV300-4	$GF(2^{113})$		45	166738	270			645		1808 bits
		$GF(2^{155})$		36	266443	147			784		2480 bits
		$GF(2^{281})$		33	474504	69			1311		4496 bits
OP00 [48]	XCV400E8BG432	$GF(2^{167})$	4	85.7	82212	1818			1627	1745	40 kbits
			8	74.5	45336	2857			2136	1753	40 kbits
			16	76.7	27776	4762			3002	1769	40 kbits
OP01 [49]	XCV1000E8BG680	$GF(2^{192} - 2^{64} - 1)$		40					11416	5735	140 kbits
S01 [59]	X4000XL	$GF(2^{191})$				85					
JT et al. [26]	Atmel FPSLIC	$GF(2^8)$		10	768	13020			668		
				200		260416					
		$GF(2^{16})$		10	3072	3255			852		
				200		65104					
		$GF(2^{72})$		10	62208	160			2189		
				200		3215					
		$GF(2^{192})$		10	442368	22			4097		
				200		452					

Table 23: The Area and Speed Values of the Used Circuits for Estimations of the ECDSA Circuit

Operation Block	LUTs	FFs	Block RAMs	Clock Frequency (MHz)
ECP over $GF(p)$	11,227	6,959		91.971
ECP over $GF(2^m)$	1627	1745	10	
SHA-1	722		2	95

Table 24: The Times Needed for Signature Generation and Verification with 91.971MHz

Operation	with curve over $GF\,(2^{192} - 2^{64} - 1)$	with curve over $GF\,(2^{167})$
Signature Generation	$22.42msec$	$4.59msec$
Signature Verification	$43.09msec$	$7.73msec$

7.2 Smartcard Implementations

7.2.1 Previous Results

Table 25 lists known implementations external to the NESSIE project. This table is a collection of known results; primitives with no known results are marked by N/A. These results should not be considered as a fair comparison of the primitives, as each implementation was done with a different methodology, and the implementations were not evaluated by NESSIE. In the following we describe our implementations on a 8051 Smartcard.

7.2.2 Implementation of Khazad, Misty1, and Safer++

Table 26 summarizes the results for our Khazad, Misty1 and Safer++ (legacy block size of 64 bits versions) implementations on a low cost 8-bit smart-card (8051). The RAM and ROM are expressed in bytes. The "(+16)" means that 16 bytes must be added if the key is to be kept. The given number of cycles is for the encryption of a 8-byte block *and* the key schedule.

As we can see, Safer++ has comparable performances with Khazad and Misty1 on a 8051-based CPU, whereas on desktop machines it is slower. This is due to the fact that most of operations in Safer++ are performed on 8-bit words, so it does not gain much performance on desktop machines, while Khazad and Misty1 can be optimized using 32-bit words. However, the implementation of Misty1 might be more optimized.

7.2.3 Implementation of (untweaked) ESIGN

SFLASH might have an advantage over other signature schemes only if it is faster on a low cost smart card without any coprocessor. Two algorithms are candidates to compete with SFLASH: ESIGN and ECDSA. These three algorithms have similar performances on desktop computers, and a rough evaluation shows that they may have similar performances

Table 25: Previous Implementations on Smartcards

Primitive	Smartcard	Speed	More Details
CS-cipher	6805	1600 cycles/byte	non optimised
Hierocrypt-L1	Z80/JT6N55	2425 states/byte	RAM/ROM:26/2447bytes
Misty1	Z80/JT6N55	3185 states/byte	RAM/ROM:44/1598bytes
Triple-DES	6805	6547 cycles/byte	RAM/ROM:213/379bytes
Camellia	8051	638.5 cycles/byte	submission document [4]
RC6-128	6805	2046 cycles/byte	RAM/ROM:200/639bytes
	8051	900 cycles/byte	RAM/ROM:221/596bytes
	ARM	49 cycles/byte	RAM/ROM:192/272bytes
Rijndael-128	6805	895 cycles/byte	RAM/ROM:50/540bytes
	8051	199 cycles/byte	RAM/ROM:49/1016bytes
	ARM	180 cycles/byte	RAM/ROM:16/3900bytes
SHACAL-1	6805	3362 cycles/byte	RAM/ROM:118/379bytes
SHA-1	6805	1051 cycles/byte	RAM/ROM:118/419bytes
GPS	6805	N/A	RAM/ROM:?/300bytes

Table 26: Results of our Khazad, Misty1, and Safer++ Implementations

	RAM	ROM (code + tables)	Cycles
Khazad	41(+16)	1227 (705 + 512)	4000
Misty1	31	2682 (1530 + 1152)	5280
Safer++	35(+16)	1345 (705 + 640)	3966

on low cost smart cards. ESIGN has been chosen for this evaluation because there are no IPR problems with optimized implementations of the algorithm.

Low cost smartcards usually have microprocessors either of the Intel 8051 family or the Motorola 6805 family. The 8051 was chosen as the platform for our comparison. All the evaluations are made on the simplest variant of the 8051.

Both signature schemes compute the hash of the message first. In both cases this step uses three SHA-1, and have comparable timings. Therefore, we can leave this and implement only the signature algorithm itself.

The implementation characteristics we are interested in are the memory requirements, the speed, and the code size. RAM is clearly the most sensitive resource on a smartcard. A basic 8051 processor has a bank of 128 bytes of internal memory, which is the fastest RAM available to programs. If we need more memory, we have to use external RAM, which is much slower and quite expensive. Therefore, we shall be especially attentive to RAM usage, and try to use as little RAM as possible. SFLASH parameters have fixed size, thus we have no decisions to make about their implementation. For ESIGN there is some flexibility available in the parameters size, but we do not use this feature as our implementation is designed to support a fixed key size. We chose the recommended size of 1152 bits for the modulus size and $e = 8$ for the exponent (this is the smallest exponent that fulfills the security requirements, as recommended by the designers in their original (untweaked) submission).

First thing to note when implementing ESIGN is that the parameters size forces us to use external RAM, for example the modulus will not fit in the internal RAM. The main steps of the ESIGN signature algorithm are two (very different) modular exponentiations. The first modular exponentiation is the computation of $r^8 \bmod n$, where n is a 144-byte modulus, and r is a 96-byte value. This step is clearly the most costly one: it's computation takes more than a third of the total time, due to the fact that we have to perform three squarings and three modular reductions on large numbers that have to be stored in external memory, as they cannot fit in the on-chip memory of the 8051. External RAM is more costly to handle, as there is only one pointer for accessing data located above the first 256 bytes of the external RAM. The second main step is the modular exponentiation $r^7 \bmod p$, where p is a 48-byte prime. Here we have to perform two squarings, two multiplications and four modular reduction. The main difference from the previous modular exponentiation is that, as the values we have to handle are not so big, we can use the internal RAM, and need only some smaller external RAM. Thus, this step is not as costly as the first. In total both modular exponentiations take about half of the total time.

In Table 27 we give the characteristics of our ESIGN implementation. The performance of this implementation could be improved following a comment of Chung-Huang Yang. The computation of $r^7 \bmod p$ can be replaced by one modular reduction $(r^8 \bmod n) \bmod p$ (where $r^8 \bmod n$ is already computed) and a multiplication by the modular inverse $r^{-1} \bmod p$.

An implementation of ESIGN on a 8-bit H8/300 based smartcard (Hitachi) has been done by Morita, Okamoto and Yang. The claimed results are that a signature generation takes 6.15 s, with a 1152-bit modulus and $e = 1024$, and a CPU running at 5MHZ. For

Table 27: Characteristics of the (Untweaked) ESIGN and SFLASHv2 Smartcard Implementations

	Key size (bytes)	**Code size** (bytes)	**Memory size** (bytes)	**Signature** (CPU cycles)
ESIGN	336	3000	800	5 100 000
SFLASHv2		2000 + 1369	127	7 200 000

comparison, our implementation would take 12 s to generate a signature with a CPU at the same speed. However, this should be not considered as a fair comparison, as the smartcards are different.

7.2.4 Implementation of SFLASHv2

We give the details of an implementation of the SFLASHv2 algorithm on a basic 8051 smartcard. We let K denote the field $\mathbb{F}_{128}$ and L denote the field $\mathbb{F}_{2^{259}}$ which is an extension of degree 37 of K. These fields are described in the submission respectively as quotients of the polynomial ring $\mathbb{F}_2[X]$ (resp. $K[X]$) by an ideal generated by an irreducible polynomial of degree 7 (resp. 37).

We implemented the steps 6–9 of the SFLASHv2 signature algorithm, as described in the submission. The initial stages consist of formatting a message into an element of L through a series of three applications of SHA-1. These steps (1–5) were neglected since the main goal of this implementation is to compare SFLASHv2 against ESIGN on smartcards and both have the same initial triple hashing.

The implementation was written in assembly. In the first attempt we focused on fitting the signature on a basic 8051 without external memory, that is with only the basic 128 bytes of RAM. We list below the significant data of this implementation.

- Code size: 2K + some extra tables (37×37 bytes)
- RAM: 127 bytes (contains registers, stack and 3 elements of L, namely 111 bytes)
- Run time for one signature: approximately 7.2 million cycles

In Table 27 we give these characteristics of our SFLASHv2 implementation. An independent implementation by Schlumberger-Sema gives around 223 000 CPU cycles (using external memory). The performance of the implementations by Schlumberger-Sema with and without optimization of the multiplications using Gray codes is given in Table 28.

We describe the technical part of our implementation. Steps 6–9 of the SFLASHv2 signature algorithm involve two affine maps of L viewed as a K-vector space of dimension 37. These maps are described in the `GEN_AFF` function, which inputs addresses of vectors `VAR_i` and `VAR_j` and computes `VAR_j` = `A` `VAR_i` + `B`, where `A` is a 37×37 matrix and `B` a 37×1 matrix (a column vector), both stored in ROM.

Table 28: Performance of Optimized Implementations of SFLASHv2 (as Updated by the Submitters)

CPU Type	**Using Gray Codes**	**CPU Cycles**	**Clock Cycles**	**Time on 3.57 MHz** (sec)	**Time on 10MHz** (sec)	**RAM** (bytes)	**ROM** (bytes)
Intel 8051	no	319 968	3 839 616	1.075	0.384	473	2.5K
	yes	223 118	2 677 416	0.750	0.268	473	3.1K
Infineon 66	no		821 135	0.230	0.082	473	2.5K
	yes		586 605	0.164	0.059	473	3.1K

The delicate part of the signature is the computation of $\mathtt{B} = \mathtt{A}^h$ (the modular exponentiation) in L. To do this we follow suggestion (b) of the SFLASHv2 description (whereas in the Schlumberger-Sema implementation they compute the minimal h, thus saving a lot of computation). To achieve this exponentiation we need a linear map, a multiplication map in L `MULTL` and a squaring map in L `SQUAREL`.

Note that we cannot use `MULTL` to square because we only use 128 of RAM: `MULTL` multiplies two elements and writes the result into a third different one, changing two of the three variables of L that are stored in the RAM, whereas `SQUAREL` squares a variable and writes it into a different one, changing only one of the three variables. In performing the modular exponentiation we often need to keep the values of two variables, hence the need for `SQUAREL`.

Since the linear map is the same as the affine map `GEN_AFF` with $B = 0$, we only introduced a switch (bit) to indicate before calling the map that the linear version will be called.

All these functions rely on a sub-function `MULTK` which performs the multiplication of two bytes (elements of K). This function was made SPA-resistant by using a table of logarithm and a table of exponentials to transform multiplications into sums. Thus we were able to lower the cycle count from 12.4 to 7.2 million with an additional cost of storing in ROM two 128-byte arrays.

The more dramatic improvements in the cycle count will come from a better implementation of the modular exponentiation. In particular, if we can efficiently compute inverses in L, we can speed up things quite a bit.

On the other hand, the Schlumberger-Sema implementation uses a smaller representative of the exponent h and exploits (as we did) a quasi-periodicity of some loops.

7.2.5 GPS

GPS submitters claim that GPS (and its tweak) were designed for smartcard applications, in view of speed measurements and code size (only 300 bytes). We obtained very high speed measurements: for smart card applications, the card only need compute the answer

which on our platforms takes between 1 and 3 ms. If we use a much larger number for $|A|$, the speed for the answer is about the same and only the commitment and the verification are slower (commitment may take about 100-300 ms).

References

[1] Gordon B. Agnew, Ronald C. Mullin, and S. A. Vanstone. A fast elliptic curve cryptosystem. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Proceedings of Eurocrypt'89*, number 434 in Lecture Notes in Computer Science, Advances in Cryptology, pages 706–708, Belgium, 1989. Springer-Verlag.

[2] Gordon B. Agnew, Ronald C. Mullin, and S. A. Vanstone. On the development of a fast elliptic curve cryptosystem. In Rainer A. Rueppel, editor, *Proceedings of Eurocrypt'92*, Lecture Notes in Computer Science, Advances in Cryptology, pages 482–487, Hungary, 1992. Springer-Verlag.

[3] Gordon B. Agnew, Ronald C. Mullin, and S. A. Vanstone. An implementation of elliptic curve cryptosystem over $f_{2^{155}}$. *IEEE Journal on Selected Areas in Communications*, 11(5):804–813, June 1993.

[4] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita. Camellia: A 128-bit block cipher suitable for multiple pltaforms. Technical report, submitted to Nessie, 2000.

[5] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms. Primitive submitted to NESSIE by NTT Corp., September 2000. See also `http://info.isl.ntt.co.jp/camellia/`.

[6] Paulo Sérgio L. M. Barreto and Vincent Rijmen. Anubis. Primitive submitted to NESSIE, September 2000.

[7] Paulo Sérgio L. M. Barreto and Vincent Rijmen. The KHAZAD legacy-level block cipher. Primitive submitted to NESSIE, September 2000.

[8] Paulo Sérgio L. M. Barreto and Vincent Rijmen. Whirlpool. Primitive submitted to NESSIE, September 2000.

[9] Eli Biham. Optimization of IDEA. Technical report, NESSIE document NES/DOC/TEC/WP6/026/1, 2002.

[10] J. Black, S. Halei, H. Krawczyk, T. Krovetz, and P. Rogaway. Umac: Fast and secure message authentication. Advances in Cryptology – proceedings of CRYPTO 1999, LNCS 1666, pages 216–233. Springer-Verlag, 1999.

[11] Johan Borst. GrandCru. Primitive submitted to NESSIE, September 2000.

[12] Nicolas T. Courtois, Louis Goubin, and Jacques Patarin. Quartz, an asymmetric signature scheme for short signatures on PC (second revised version). Primitive submitted to NESSIE, September 2001. See also `http://www.minrank.org/quartz/`.

[13] Cryptrec. Cryptrec liaison report to ISO/IEC 18033-2 and 18033-3. Technical report, Cryptography Research and Evaluation Committees, October 2002.

[14] Joan Daemen, Michael Peeters, Gilles Van Assche, and Vincent Rijmen. Noekeon. Primitive submitted to NESSIE, September 2000.

[15] Ed Dawson, W. Millan, L. Penna, L. Simpson, and Jovan Dj. Golic. LILI-128. Primitive submitted to NESSIE, September 2000.

[16] Patrick Ekdahl and Thomas Johansson. SNOW. Primitive submitted to NESSIE, September 2000.

[17] Pierre-Alain Fouque, Jacques Stern, and Serge Vaudenay. CS-cipher. Primitive submitted to NESSIE by CS Communication & Systmes, September 2000.

[18] Eiichiro Fujisaki, Tetsutaro Kobayashi, Hikaru Morita, Hiroaki Oguro, Tatsuaki Okamoto, Satomi Okazaki, and David Pointcheval. PSEC: Provably secure elliptic curve encryption scheme. Primitive submitted to NESSIE by NTT Corp., September 2000. See also `http://info.isl.ntt.co.jp/psec/`.

[19] Eiichiro Fujisaki, Tetsutaro Kobayashi, Hikaru Morita, Hiroaki Oguro, Tatsuaki Okamoto, Satomi Okazaki, David Pointcheval, and Shigenori Uchiyama. EPOC: Efficient probabilistic public-key encryption. Primitive submitted to NESSIE by NTT Corp., September 2000. See also `http://info.isl.ntt.co.jp/epoc/`.

[20] Lijun Gao, Sarvesh Shrivastava, and Gerald E. Sobelman. Elliptic curve scalar multiplier design using fpgas. In *Proceedings of the 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Lecture Notes in Computer Science, pages 257–305, Worcester, MA, USA, August 1999. Springer-Verlag.

[21] Helena Handschuh and David Naccache. Shacal. Primitive submitted to NESSIE by Gemplus, September 2000.

[22] Johan Håstad and Mats Nāslund. BMGL: Synchronous key-stream generator with provable security. Primitive submitted to NESSIE, September 2000.

[23] Philip Hawkes and Greg Rose. SOBER. Primitive submitted to NESSIE by Qualcomm International, September 2000.

[24] Helion Technology Limited, The Granary, Home End, Fulbourn, Cambridge CB1 5BS, UK. *DATASHEET - Compact Dual Hash Processor Core for Xilinx FPGA*. URL: www.heliontech.com.

[25] ISO/IEC. ISO/IEC 9797-1. International standard, 1999.

[26] S. Janssens, J. Thomas, W. Borremans, P. Gijsels, I. Verbauwhede, F. Vercauteren, B. Preneel, and J. Vandewalle. Hardware/software co-design of an elliptic curve public-key cryptosystem. In *Proceedings IEEE Workshop on of Signal Processing Systems*, pages 209–216, 2001.

[27] Don B. Johnson and Simon Blake-Wilson. ECDSA. Primitive submitted to NESSIE by Certicom, September 2000.

[28] Don B. Johnson and Simon Blake-Wilson. ECIES. Primitive submitted to NESSIE by Certicom, September 2000.

[29] Jakob Jonsson and Burt Kaliski. RC6 block cipher. Primitive submitted to NESSIE by RSA, September 2000.

[30] Jakob Jonsson and Burt Kaliski. RSA-OAEP. Primitive submitted to NESSIE by RSA, September 2000.

[31] Jakob Jonsson and Burt Kaliski. RSA-PSS. Primitive submitted to NESSIE by RSA, September 2000.

[32] Ted Krovetz, John Black, Shai Halevi, Alejandro Hevia, Hugo Krawczyk, and Phillip Rogaway. UMAC. Primitive submitted to NESSIE, September 2000.

[33] Xuejia Lai and James L. Massey. A proposal for a new block encryption standard. In Ivan B. Damgård, editor, *Proceedings of Eurocrypt'90*, number 473 in Lecture Notes in Computer Science, pages 389–404. Springer-Verlag, 1990.

[34] Xuejia Lai and James L. Massey. IDEA. Primitive submitted to NESSIE by R. Straub, MediaCrypt AG, September 2000. Based on [33] and [35].

[35] Xuejia Lai, James L. Massey, and Sean Murphy. Markov ciphers and differential cryptanalysis. In *Proceedings of Eurocrypt'91*, number 547 in Lecture Notes in Computer Science, pages 17–38. Springer-Verlag, 1991.

[36] Anatoly N. Lebedev and Alexey A. Volchkov. Nush. Primitive submitted to NESSIE by LAN Crypto, Int., September 2000.

[37] K. H. Leung, K. W. Ma, W. K. Wong, and Philip H. W. Leong. Fpga implementation of a microcoded elliptic curve cryptographic processor. In *Proceedings of Field-Programmable Custom Computing Machines (FCCM'00)*, pages 68–76, 2000.

[38] Helger Lipmaa. Idea a cipher for multimedia architectures?. Selected Areas in Cryptology'99, LNCS 1556, pages 256–268. Springer-Verlag, 1998.

[39] Helger Lipmaa. Fast software implementation of SC2000. Information Security Conference 2002, LNCS 2433, pages 63–74. Springer-Verlag, 2002.

[40] Alexis Warner Machado. Nimbus. Primitive submitted to NESSIE, September 2000.

[41] James L. Massey, Gurgen Khachatrian, and Melsik K. Kuregian. Nomination of SAFER++ as candidate algorithm for the New European Schemes for Signatures, Integrity, and Encryption (NESSIE). Primitive submitted to NESSIE by Cylink Corp., September 2000.

[42] M. Matsui. Performance analysis and parallel implementation of dedicated hash functions. Advances in Cryptology – proceedings of Eurocrypt 2002, LNCS 2332, pages 165–180. Springer-Verlag, 2002.

[43] Mitsuru Matsui. Specification of MISTY1 - a 64-bit block cipher. Primitive submitted to NESSIE by E. Takeda, Mitsubishi, September 2000.

[44] Leslie McBride. Q. Primitive submitted to NESSIE by Mack One Software, September 2000.

[45] David McGrew and Scott R. Fluhrer. LEVIATHAN. Primitive submitted to NESSIE by Cisco Systems, Inc., September 2000.

[46] Kenji Ohkuma, Fumihiko Sano, Hirofumi Muratani, Masahiko Motoyama, and Shinichi Kawamura. Hierocrypt. Primitive submitted to NESSIE by Toshiba Corp., September 2000.

[47] Tatsuaki Okamoto. ESIGN. Primitive submitted to NESSIE by NTT Corp., September 2000. See also `http://info.isl.ntt.co.jp/esign/`.

[48] Gerardo Orlando and Christof Paar. A high-performance reconfigurable elliptic curve processor for $gf(2^m)$. In *Proceedings of Workshop on Cryptograpic Hardware and Embedded Systems (CHES) 2000*, number 1965 in Lecture Notes in Computer Science. Springer-Verlag, 2000.

[49] Gerardo Orlando and Christof Paar. A scalable $GF(p)$ elliptic curve processor architecture for programmable hardware. In Cetin K. Koç, David Naccache, and Christof Paar, editors, *Proceedings of Workshop on Cryptograpic Hardware and Embedded Systems (CHES) 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 356–371, Paris, France, May 14-16 2001. Springer-Verlag.

[50] Sıddıka Berna Örs, Lejla Batina, and Bart Preneel. Hardware implementation of elliptic curve processor over GF(p). Technical report, NESSIE document NES/DOC/KUL/WP5/023, October 10, 2002.

[51] NESSIE partners. Report on the Performance Evaluation of NESSIE Candidates I. Technical report, NES/DOC/UCL/WP4/D14/1, 2001.

[52] Jacques Patarin et al. Flash and Sflash. Primitive submitted to NESSIE, September 2000.

[53] Guillaume Poupard et al. GPS. Primitive submitted to NESSIE, September 2000.

[54] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Performance comparision of the AES submissions. Proceedings of the Second AES Candidate Conference, NIST, pages 15–34. Springer-Verlag, March 1999.

[55] Thomas Schweinberger and Victor Shoup. ACE: The advanced cryptographic engine. Primitive submitted to NESSIE, September 2000.

[56] Takeshi Shimoyama, Hitoshi Yanami, Kazuhiro Yokoyama, Masahiko Takenaka, Kouichi Itoh, Jun Yajima, Naoya Torii, and Hidema Tanaka. SC2000. Primitive submitted to NESSIE by N. Torii, Fujitsu Laboratories LTD, September 2000. Based on [57].

[57] Takeshi Shimoyama, Hitoshi Yanami, Kazuhiro Yokoyama, Masahiko Takenaka, Kouichi Itoh, Jun Yajima, Naoya Torii, and Hidema Tanaka. The block cipher SC2000. In Mitsuru Matsui, editor, *Proceedings of Fast Software Encryption – FSE'01*, number 2355 in Lecture Notes in Computer Science, pages 312–327. Springer-Verlag, 2001.

[58] Victor Shoup. A proposal for an ISO standard for public key encryption (version 2.0). Available at `http://eprint.iacr.org/2001/112/`, 2001.

[59] Nigel P. Smart. The hessian form of an elliptic curve. In Cetin K. Koç, David Naccache, and Christof Paar, editors, *Proceedings of Workshop on Cryptograpic Hardware and Embedded Systems (CHES) 2001*, number 2162 in Lecture Notes in Computer Science, pages 121–128, Paris, France, May 14-16 2001. Springer-Verlag.

[60] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat. Design strategies and modified descriptions to optimize cipher FPGA implementations: Fast and compact results for DES and Triple-DES. Technical report, submitted to FPGA 2003, 2002.

[61] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat. A methodology to implement block ciphers in reconfigurable hardware and its application to fast and compact AES rijndael. Technical report, submitted to FPGA 2003, 2002.

[62] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat. Optimized FPGA implementation of block cipher MISTY1. Technical report, submitted to RAW 2003, 2002.

[63] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat. Efficient FPGA implementations of block ciphers KHAZAD and MISTY1. Technical report, in the proceedings of The Third NESSIE Workshop, November 6-7 2002.

[64] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat. Efficient FPGA implementation of block cipher MISTY1. Technical report, NESSIE report NES/DOC/UCL/WP6/005/1, September 2002.

[65] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat. Optimized FPGA implementation of block cipher KHAZAD. Technical report, NESSIE report NES/DOC/UCL/WP6/004/1, September 2002.

[66] Sarwono Sutikno, Ronny Effendi, and Andy Surya. Design and implementation of arithmetic processor $gf(2^{155})$ for elliptic curve cryptosystems. In *Proceedings of the 1998 IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS'98)*, pages 647–650, 1998.

[67] Bart Van Rompay and Bert Den Boer. TTMAC. Primitive submitted to NESSIE, September 2000.

Table 29: Host Computers

Machine	Location	Speed, Memory	Operating Systems	Compilers
PIII/Linux	TEC	450MHz, 256M	Linux 2.4.17	gcc 3.1.1, gcc3.3
	TEC	600MHz, 256M	Linux 2.4.17	gcc 2.95.2, gcc 3.0.3, gcc 3.1.1
	TEC	933MHz, 256M	Linux 2.4.17	gcc 2.95.2
	RHUL	665MHz	Linux 2.2.16	egcs 2.91.66
PIII/MS	TEC	450MHz	Win 2000	Visual C 6.0, gcc 2.95.3
	SAG	850MHz	Win 2000	gcc 2.95.3
	SAG	850MHz	Win 2000	Intel C++ 6.0, Visual C 6.0
	UCL	850MHz, 256M	Win 2000	gcc 2.95.3
PI/MMX	TEC	133MHz, 80M	Linux 2.2.9	gcc 2.7.2.3
Pentium4	TEC	1.8GHz	Linux 2.4.0	gcc 2.95.2
	TEC	1.7GHz	Linux 2.4.12	gcc 2.95.2, gcc 3.1.1
Pentium2	TEC	350MHz, 64M	Win 98	Visual C 6.0, gcc 2.95.3
Xeon	TEC	1680MHz	Linux 2.4.2	gcc 2.96, egcs 2.91.66
	KUL	1500MHz	Linux 2.4.7	gcc 2.96
486	TEC	33MHz	Linux 2.0.30	gcc 2.7.2.3
Alpha EV6.7 (21264A)	TEC	667MHz	OSF1 V4.0, V5.1	Dec C V6.4, gcc 2.97
Sun/Sparc V9	TEC	400MHz	Sun OS 5.8	SWC 5.1, gcc 3.0.4
	TEC	338MHz	Sun OS 5.8	SWC 5.1, gcc 3.0.4
	UIB	450MHz	Sun OS 5.8	SWC 5.2
	ENS	450MHz	Sun OS 5.7	SWC 5.0
	ENS	333MHz	Sun OS 5.8	gcc 3.2.1
	SAG	248MHz (Sparc)	Sun OS 5.6	gcc 2.95.3
Mac (PPC)	ENS	400MHz	Mac OS X	gcc 2.95.2
AMD	TEC	Duron 1200MHz,128M	WinXP	Visual C 7.0 (.NET), gcc 2.95.3
	ENS	Athlon 1700 (1467MHz)	Linux 2.4.18	gcc 2.96

Table 30: Legacy Block Ciphers Performance Results

Primitive Name	Key Size	Machine											
		PIII Claimed	**PIII/Linux**	**PIII/MS**	**PI/MMX**	**Pentium4**	**Pentium2**	**Xeon**	**486**	**Alpha**	**Sparc V9**	**Mac**	**AMD**
Idea	128	78.7/78.7/–	55/55/450	53/52/395	151/151/1738	100/100/472	59/59/802	100/100/448	133/133/941	55/56/399	88/88/599	53/53/494	55/55/434
Khazad	128	67/67/1206	39/40/765	38/38/823	72/73/2082	48/48/904	45/45/1023	45/45/898	87/87/2092	19/19/368	34/33/799	40/39/981	43/43/837
Khazad-tweak	128	67/67/1206	39/40/766	44/44/822	83/83/2084	45/45/916	45/45/1011	44/45/934	78/78/2182	19/19/373	33/33/808	41/41/996	43/43/861
Misty1	128	37.5/–/170(PII)	47/47/208	42/43/125	102/101/401	59/56/131	48/48/195	56/52/130	69/69/243	36/35/90	33/34/117	62/60/174	52/52/120
Safer++	128	169/160/1333	152/168/1504	143/161/1692	296/256/3570	154/171/3302	145/193/2199	156/171/1731	215/206/2678	85/87/1876	104/117/2173	148/179/2614	150/182/1344
Cs-cipher	128	506/506/–(PI)	156/140/2686	166/149/2920	268/329/4994	164/133/5227	166/150/2922	137/131/6513	209/219/4521	115/117/3047	124/156/3350	162/187/4108	196/185/3535
Hierocrypt-L1	128	24.8/24.8/–	43/46/34K	48/51/36K	79/90/47K	69/68/51K	56/58/36K	70/72/55K	118/121/37K	23/28/37K	61/66/33K	67/71/33K	39/42/32K
Nimbus	128	66/–/24665	34/34/12K	36/34/12K	62/58/24K	83/82/31K	36/34/12K	83/82/32K	74/73/30K	19/19/7219	64/68/22K	59/59/19K	39/42/16K
Nush	128	22.5/–/64*	44/38/1415	47/36/1402	111/111/2424	44/26/1103	55/67/1389	44/25/1094	45/50/1223	30/21/942	30/26/2137	36/36/1202	27/25/1659
CAST-128	128		30/30/916	34/34/774	73/73/1486	39/39/1024	38/37/1016	40/39/1007	47/47/2163	36/37/608	32/33/615	37/37/1107	30/31/1005
DES	56		59/59/883	62/62/929	88/88/1624	61/61/816	62/62/932	60/60/819	87/86/1234	37/37/596	54/53/798	60/59/764	54/54/773
Triple-DES	168		154/155/2636	160/160/2744	253/252/4868	158/158/2477	161/160/2755	157/157/2453	504/530/3737	99/99/1814	126/126/2336	156/155/2341	141/141/2312
Kasumi	128		73/74/297	91/92/267	146/146/738	157/148/253	151/150/439	152/161/257	126/126/370	88/88/180	96/97/278	98/99/310	88/85/198
RC5 (12 Rounds)	64		19/19/1202	24/24/1494	67/73/3302	30/30/1420	24/24/1504	30/30/1420	39/44/2673	28/25/1109	23/22/1555	19/25/1404	19/19/1236
Skipjack	80		114/120/16	133/150/11	165/166/38	108/107/21	150/152/10	110/107/20	100/100/28	74/78/24	53/54/32	172/133/10	149/144/8.4

The figures in the entries are of encryption/decryption/keysetup times, where the encryption and decryption times are measured in cycles/byte and the key setup time is measured in cycles/keysetup.

* The key schedule claims are for encryption only. Our figures also include decryption subkeys.

Table 31: Normal and High Block Ciphers Performance Results

Primitive Name	Key Size	Machine											
		PIII Claimed	**PIII/Linux**	**PIII/MS**	**PI/MMX**	**Pentium4**	**Pentium2**	**Xeon**	**486**	**Alpha**	**Sparc V9**	**Mac**	**AMD**
Camellia	128	36/36/263 (PII)	35/35/313	37/37/334	80/80/825	64/63/453	38/38/426	64/64/444	65/60/561	36/35/287	47/47/352	31/31/383	31/31/274
	192	30.8/30/222 (Asm)	45/45/420	49/48/434	105/105/1166	86/86/546	49/49/533	82/85/537	105/105/788	48/47/407	61/61/539	41/41/578	41/41/386
	256	30.9/30/266 (Asm)	45/45/429	49/49/447	105/105/1207	86/87/555	49/49/554	83/85/559	105/105/796	48/47/453	60/60/542	41/41/578	41/41/393
Camellia 2nd impl	128		35/35/285	37/37/265	82/82/676	72/73/415	37/37/342	72/72/408	62/65/412	44/44/276	50/46/293	32/32/336	32/32/235
	192		45/45/390	49/48/395	107/107/1036	96/96/516	49/49/514	95/95/513	96/101/698	58/58/416	65/60/508	41/41/541	40/41/346
	256		45/45/401	49/48/407	107/107/1075	96/96/539	49/49/526	95/95/533	96/101/701	58/58/441	65/60/514	41/41/541	42/42/361
RC6	128	16/16/– (P-pro)	17/17/1054	24/25/1040	84/85/3094	36/37/2056	24/25/1060	37/36/2050	86/86/2392	27/24/1509	69/68/1432	28/32/1106	17/17/947
(20 rounds)	192		18/17/1175	30/25/1258	84/85/3154	37/37/2141	32/25/1291	37/37/2162	86/86/2904	27/24/1725	117/117/1901	28/32/1115	17/18/1039
	256	16/16/– (P-pro)	18/17/1168	25/25/1262	84/85/3134	37/37/2153	25/25/1291	37/37/2061	86/86/2505	26/24/1727	69/69/1921	28/32/1149	17/18/1032
Safer++	128	40/76/2333	50/63/1333	46/60/1464	104/117/3154	47/58/2918	46/63/1480	47/53/1550	69/77/2035	30/44/1677	35/43/1939	46/74/2266	41/52/1192
	256	57/101/3227	68/88/1805	63/84/1865	144/165/4407	65/83/3998	64/89/1887	65/79/2085	98/109/2820	42/62/2257	49/73/2590	62/104/3226	57/73/1611
Anubis	128	36.8/36.8/4527	37/37/3550	37/37/3727	80/80/9189	35/35/3750	37/37/4389	35/35/3669	55/56/7895	25/25/2990	33/33/4885	36/36/5015	37/37/4687
	160	39.3/39.3/5709	40/40/4519	39/39/4927	85/85/11K	37/37/4845	39/39/5570	37/37/4751	61/64/10K	27/27/3950	35/35/6329	38/38/6540	37/37/6119
	192	41.6/41.6/8008	43/43/5857	42/42/6993	91/91/15K	40/39/6054	42/41/7692	40/40/5949	64/65/12K	29/29/4803	37/37/8088	40/40/8233	42/41/7601
	224	43.8/43.8/9576	45/45/7356	44/44/8546	97/97/18K	41/41/7338	44/44/9251	42/42/7330	67/68/15K	31/31/5715	40/40/9989	41/42/9940	43/44/9387
	256	46.3/46.3/11264	48/48/8876	47/47/10K	102/102/22K	44/44/8902	46/46/11K	44/45/8844	71/71/18K	33/33/8234	42/42/12K	43/43/11K	46/46/11K
	288	48.5/48.5/11291	50/50/10K	49/49/12K	108/108/26K	46/46/10K	49/49/12K	47/47/10K	75/76/21K	35/35/9954	45/45/14K	44/45/13K	48/48/13K
	320	50.8/50.8/15169	53/53/12K	51/51/13K	113/113/30K	48/48/11K	51/51/15K	50/49/11K	82/86/24K	36/37/10K	47/47/16K	47/47/16K	49/49/15K
Grandcru	128	2812/4062/20K*	1250/1518/89K	1294/1686/92K	3121/4177/214K	1634/2274/124K	1731/2024/120K	1675/2136/123K	2171/2646/161K	1028/1278/72K	1574/2166/117K	1798/2129/130K	1612/2106/114K
Hierocrypt-3	128	37/63/370*	51/64/125K	50/56/131K	97/134/169K	54/63/195K	49/56/131K	57/63/202K	135/174/137K	33/39/131K	59/78/118K	74/93/121K	52/61/100K
	192	44/781/386*	60/75/147K	59/67/154K	114/158/199K	63/75/229K	59/66/155K	67/74/239K	158/202/161K	38/46/153K	71/87/140K	85/107/142K	60/72/124K
	256	50/88/468*	69/86/170K	67/76/178K	130/182/230K	73/85/266K	67/76/178K	77/85/276K	183/235/185K	44/52/176K	80/103/161K	96/122/166K	69/83/147K
Noekeon-Dir	128	32/32/0	56/58/31	89/91/28	146/148/81	71/72/31	89/91/34	67/99/31	114/117/57	72/76/28	55/54/15	52/62/11	77/75/27
Noekeon-Ind	128	32/32/525	44/45/634	43/44/732	89/85/1546	64/66/948	52/57/1027	63/66/877	67/68/1049	41/42/635	37/36/546	31/32/515	36/36/620
Nush	128	21/21/112	23/20/2528	28/25/2579	34/30/4385	31/20/1982	28/25/2556	31/20/1978	30/30/2196	23/15/1305	24/21/3900	35/28/2100	22/20/3077
	256	21/21/–	23/20/2545	28/25/2595	34/30/4418	31/20/2000	28/25/2585	31/20/1999	30/30/2229	23/15/1322	24/21/3932	36/28/2110	22/19/3088
Q	128	36/36/500*	54/107/993	59/140/984	96/142/1923	51/231/986	59/144/1022	51/231/1032	80/131/1417	35/165/760	47/87/856	76/187/1319	48/167/954
	256	AMD-ATHLON	60/119/1154	64/156/1104	106/158/2184	58/256/1153	64/160/1193	56/257/1147	89/147/1605	39/184/852	54/98/1014	85/208/1583	54/185/1066
SC2000	128	23.9/25.18/488*	38/41/2424	70/42/2756	66/68/11K	60/60/4661	78/50/3031	64/64/4676	214/237/7713	32/31/3307	29/31/3866	32/32/3026	33/37/2718
	192	27.3/28.7/525*	43/46/2823	81/48/3073	75/77/12K	68/69/5856	90/57/3357	74/75/5798	310/325/8800	36/36/3701	34/35/4862	35/36/3407	45/41/3023
	256	27.5/32.8/526*	43/46/2860	81/48/3089	75/77/12K	68/69/5909	90/57/3386	73/74/5841	278/326/8932	37/37/3730	34/36/4865	35/36/3377	40/41/3044
Mars	128		31/30/2520	36/35/2354	116/112/2882	82/72/3492	43/37/2635	81/74/3306	133/119/2072	33/29/2924	48/49/2863	32/32/2269	29/32/2411
	192		31/30/2530	36/35/2337	117/116/2951	82/72/3480	43/37/2639	81/74/3294	124/119/2087	33/29/2929	48/49/2872	32/32/2290	29/32/2412
	256		31/30/2525	36/35/2336	116/112/2991	82/73/3599	43/37/2654	81/74/3337	138/113/2098	33/29/2934	49/49/2904	32/32/2311	29/32/2429
Rijndael	128		25/26/504	23/23/497	56/56/1461	24/25/689	22/23/612	24/25/711	48/48/1805	17/17/493	21/21/453	29/28/1011	30/31/500
	192		30/31/601	27/27/552	67/67/1713	28/29/820	27/27/714	27/28/839	56/58/2173	20/21/449	25/25/463	32/32/1178	35/36/511
	256		34/35/949	32/32/775	78/78/2159	32/35/1327	31/31/878	32/33/1363	64/64/2571	24/24/663	28/30/750	37/37/1405	39/41/708
Seed	128		45/45/409	51/50/421	78/78/949	63/63/401	51/51/447	69/70/385	76/77/758	40/40/340	34/34/353	36/36/403	41/41/393
Serpent	128		68/80/1577	59/57/1276	99/127/3072	154/172/2235	73/77/1872	153/162/2230	284/303/2952	54/41/1383	57/53/1563	50/51/1398	60/64/1675
	192		68/80/1578	59/57/1272	99/127/3053	155/171/2232	73/76/1874	152/161/2226	277/299/2952	54/41/1381	57/53/1577	50/49/1423	60/65/1681
	256		68/80/1566	59/57/1257	99/127/3041	154/171/2231	73/76/1884	153/161/2235	241/308/3003	54/41/1376	57/53/1575	50/51/1420	60/65/1686
Twofish	128		28/26/10K	28/29/15K	56/55/19K	51/49/8871	31/33/11K	51/48/12K	38/37/14K	20/20/8661	28/28/14K	35/35/14K	30/33/19K
	192		28/26/12K	30/30/17K	56/55/25K	52/49/11K	31/33/13K	52/48/14K	37/40/18K	20/20/12K	28/27/18K	35/35/19K	29/33/26K
	256		28/25/16K	28/29/20K	56/55/32K	52/49/13K	31/33/17K	51/48/16K	38/38/22K	20/20/17K	28/27/22K	35/35/23K	30/33/32K

The figures in the entries are of encryption/decryption/keysetup times, where the encryption and decryption times are measured in cycles/byte and the key setup time is measured in cycles/keysetup.

* The key schedule claims are for encryption only. Our figures also include decryption subkeys.

Table 32: Block Ciphers with 160, 192 and 256-Bit Blocks

Primitive Name	Key Size	Block Size	Machine											
			PIII Claimed	**PIII/Linux**	**PIII/MS**	**PI/MMX**	**Pentium4**	**Pentium2**	**Xeon**	**486**	**Alpha**	**Sparc V9**	**Mac**	**AMD**
SHACAL-1	512	160	140/116.5/3200	29/26/75	27/26/447	59/56/198	59/36/177	30/26/64	57/37/173	40/41/880	30/27/44	30/27/136	23/23/337	27/24/54
SHACAL-2	512	256	112.5/115/2800	44/43/737	44/46/748	82/82/1490	51/58/933	44/46/888	51/54/857	59/60/1677	30/31/671	37/37/1169	29/29/809	38/40/597
RC6 (20 rounds)	256	256		59/58/10K	85/83/12K	178/176/31K	132/123/21K	85/83/12K	133/120/21K*	153/154/20K	12/11/2716	98/100/15K	121/120/12K	58/60/12K
Rijndael	192	256		33/34/1785	33/32/1882	77/78/4828	35/35/2512	33/31/2145	35/35/2557	–/–/–	24/24/1818	30/29/2159	–/–/–	50/46/2291
	256	256		33/35/2298	34/33/2362	76/78/6126	35/36/3172	33/32/2705	34/36/3252	–/–/–	24/24/2326	31/32/2584	–/–/–	52/47/2985

The figures in the entries are of encryption/decryption/keysetup times, where the encryption and decryption times are measured in cycles/byte and the key setup time is measured in cycles/keysetup.

* On Alpha, RC6 with 256-bit blocks is more than twice faster than RC6 with 128-bit blocks, although on other processors the contrary is true.

Table 33: Stream Ciphers Performance Results

Primitive Name	Key Size	IV Size	Machine											
			PIII Claimed	**PIII/Linux**	**PIII/MS**	**PI/MMX**	**Pentium4**	**Pentium2**	**Xeon**	**486**	**Alpha**	**Sparc V9**	**Mac**	**AMD**
BMGL (m=16)	128			1475/2340	1484/2336	2869/5487	1992/2960	1624/2614	1917/2960	3753/3556	1126/1345	1583/2485	1701/2576	1867/2141
2nd impl	128			424/183K	416/184K	896/268K	706/345K	441/194K	668/332K	1734/211K	384/122K	570/224K	–/–	504/172K
Snow	128		6.76/2000	6.7/1179	6.5/1328	15/3101	8.6/1674	8.3/1349	7.7/1707	10/2284	5.6/997	6.7/1770	7.0/1490	8.1/1465
	256		6.75/2000	6.7/1206	6.4/1352	15/3216	8.7/1735	8.3/1382	7.8/1753	10/2302	5.6/1027	6.6/1791	7.0/1497	8.0/1491
Sober-t16	128		42.6/1581	40/1033	44/1188	77/2363	45/1164	62/1640	46/1146	50/1651	38/979	41/1404	42/1402	46/1207
	256		42.6/2062	40/1456	44/1604	77/3173	46/1610	55/2247	46/1526	50/2192	38/1307	41/1894	43/1900	46/1577
Sober-t32	128		24.5/1348	20/838	21/913	36/1617	27/944	26/1119	28/962	38/1749	18/759	22/1260	20/1187	18/846
	256		24.5/2564	20/1025	21/1119	37/1979	27/1140	26/1364	27/1146	38/2131	18/948	22/1508	20/1410	18/1021
Leviathan	128			10/77K	10/79K	23/126K	10/76K	13/80K	10/74K	17/70K	12/25K	14/130K	14/79K	10/98K
	192			10/77K	10/79K	23/127K	10/76K	13/80K	10/74K	17/70K	12/75K	14/129K	14/79K	10/98K
	256			10/77K	10/79K	23/127K	10/76K	13/80K	10/74K	17/70K	12/25K	14/129K	14/79K	10/98K
Lili	128		1200/–	827/46	921/43	1296/145	987/59	918/53	966/59	1040/78	566/34	665/70	717/65	841/44
RC4	128			7.3/2659	8.0/2704	17/6855	20/2568	7.8/2824	20/2879	15/5781	12/3010	13/2667	14/3024	11/2600
BMGL with IV	128	128		1474/2344/20	1504/2326/17	2869/5410/47	2002/2943/20	1618/2461/16	1899/2943/20	3675/3501/38	1124/1328/28	1571/2600/158	1615/2457/20	1861/2126/15
2nd impl	128	128		423/178K/20	414/187K/17	897/268K/47	710/335K/20	445/184K/16	675/333K/20	1690/204K/39	382/122K/29	579/220K/152	–/–/–	501/170K/15
Snow	128	64		6.7/15/612	6.2/12/690	15/41/1612	8.3/19/846	8.3/11/689	7.5/19/857	10/35/1186	5.6/14/520	6.7/51/880	7.1/11/763	8.0/14/638
with IV	256	64		6.7/44/612	6.2/16/689	15/82/1651	8.7/59/846	8.3/32/689	7.6/60/865	10/53/1229	5.6/18/541	6.6/72/889	7.1/20/774	8.0/38/680
Sober-t16	128	128		40/1158/1025	44/1253/1186	77/2636/2113	46/1287/1113	60/1714/1534	46/1299/1078	51/1747/1475	38/1031/911	41/1556/1326	43/1556/1380	46/1281/1127
with IV	256	128		40/1583/1025	44/1675/1186	77/3495/2113	46/1760/1092	55/2343/1530	46/1665/1061	51/2298/1474	38/1371/907	40/1985/1350	43/2031/1385	46/1668/1128
Sober-t32	128	128		20/995/995	21/1121/1176	36/2086/1724	27/1110/1126	26/1398/1200	27/1157/1166	38/1904/1344	18/850/843	23/1389/1400	20/1378/1361	18/925/1007
with IV	256	128		20/1176/994	21/1353/1180	36/2434/1728	27/1330/1129	26/1677/1205	27/1382/1175	39/2281/1333	19/1006/856	23/1651/1423	20/1598/1361	18/1106/1006
Scream-0	128	128		6.6/3603/1124	6.7/3651/1141	12/6917/2224	12/7196/2225	6.7/3684/1143	12/7150/2276	7.9/4653/1400	7.5/2944/991	10/5958/1606	10/6406/1910	6.8/4205/1281
Scream-F	128	128		7.5/3598/1125	7.6/3661/1141	13/6917/2224	12/7300/2221	7.6/3684/1143	13/7139/2266	8.9/4653/1404	10/2944/992	12/5995/1608	11/6419/1900	8.6/4204/1259
Scream-S	128	128		6.9/40K/1193	7.1/40K/1186	12/53K/2217	12/47K/2434	7.1/51K/1231	12/48K/2332	7.9/39K/1431	7.5/47K/995	9.9/28K/1678	8.7/48K/1481	7.2/49K/1383
Seal-3.0	160	32		5.9/158K/17	6.0/172K/12	15/304K/32	5.9/335K/16	6.0/178K/18	5.5/337K/15	7.7/208K/24	4.7/137K/10	3.8/190K/8.7	4.7/135K/6.2	5.8/141K/12

The figures in the entries are of key stream generation/keysetup times, where the key stream generation time are measured in cycles/byte and the key setup time is measured in cycles/keysetup.
For stream ciphers with initial value setup (IV) the figures in the entries are of key stream generation/keysetup/IVsetup times, where the IV setup time measured in cycles/IVsetup.

Table 34: Hash Functions Performance Results

Primitive Name	Hash Size	Machine											
		PIII Claimed	**PIII/Linux**	**PIII/MS**	**PI/MMX**	**Pentium4**	**Pentium2**	**Xeon**	**486**	**Alpha**	**Sparc V9**	**Mac**	**AMD**
Whirlpool	512	133/–/–	79/59/5534	82/47/5657	190/260/13K	108/130/8559	97/51/6804	112/125/8794	460/122/30K	35/14/2400	91/22/6266	133/82/8818	104/42/7012
2nd impl (MMX)	512		46/63/3199	73/50/5026	177/244/11K	60/132/4495	83/76/5992	60/125/4468	N/A	N/A	N/A	N/A	99/51/6573
MD4	128		4.7/15/440	4.5/12/411	6.9/44/639	6.4/34/600	4.7/13/444	6.4/29/586	4.2/36/601	6.3/11/467	6.8/10/515	7.0/14/518	4.7/8.3/419
MD5	128		7.2/15/602	6.8/12/558	8.9/44/768	9.4/34/799	7.2/12/611	9.4/29/785	5.8/36/704	10.0/11/691	10/10/746	10/14/752	7.5/10/599
RIPEMD	160		18/16/1339	16/14/1207	33/54/2363	26/36/1929	23/15/1651	26/36/1921	27/38/3741	24/11/1697	24/12/1751	20/17/1384	21/12/1493
SHA-0	160		15/16/1011	12/14/896	30/54/1769	23/36/1651	16/15/1008	23/36/1438	51/39/1933	9.8/11/653	12/12/853	12/16/829	12/12/796
SHA-1	160		15/16/1024	13/14/929	27/54/1830	25/20/1497	16/15/1086	25/20/1460	17/39/1500	11/10/745	13/12/955	9.7/16/872	12/12/825
SHA-2	256		39/44/2736	39/20/2643	71/82/4804	40/118/3159	40/34/2860	39/112/3042	60/50/4271	29/10/2021	34/34/2409	29/32/2052	34/39/2369
	384		83/157/11K	74/101/9907	187/432/25K	122/292/16K	84/244/11K	131/280/17K	176/207/24K	16/30/2333	65/118/9018	93/206/12K	71/105/9672
	512		83/156/11K	74/101/9940	187/433/25K	122/292/16K	84/244/11K	131/280/17K	176/207/24K	16/30/2348	65/117/9027	93/206/12K	71/106/9752
Tiger	192		21/16/1542	24/14/1744	41/60/2966	27/27/2015	24/13/1778	26/26/1895	73/39/5137	5.2/10/390	23/14/1551	28/19/1871	20/11/1449
BCHASH-Rijndael	128		49/15/1712	45/14/1588	90/42/3078	77/19/2579	45/13/1605	76/19/2673	214/32/6963	33/8.1/1179	54/82/1997	59/22/2043	47/11/1650
	256		112/44/3775	116/20/3913	241/93/8082	152/118/5254	123/35/4118	150/109/5288	–/–/–	97/10/3204	171/133/5583	–/–/–	149/40/4889

The figures in the entries are of hash/initialize/initialize+finalize times, where the hash time is measured in cycles/byte and the initialize and initialize+finalize times are measured in cycles.

Table 35: Message Authentication Codes Performance Results

Primitive Name	Key Size	MAC Size	Machine											
			PIII Claimed	PIII/Linux	PIII/MS	PI/MMX	Pentium4	Pentium2	Xeon	486	Alpha	Sparc V9	Mac	AMD
Ttmac	160	160	95.8/–/–	21/27/1564	19/21/1304	33/86/2479	40/50/2675	21/30/1582	37/47/2614	23/59/1906	25/16/1691	26/47/1769	21/55/1453	21/14/1526
Umac-16	128	64		6.0/52K/53K	5.8/52K/53K	22/109K/111K	6.2/76K/77K	6.1/53K/54K	6.1/76K/78K	21/72K/75K	4.5/48K/49K	20/72K/74K	12/71K/73K	6.2/74K/75K
Umac-32	128	64	≈ 2/–/–	2.9/54K/55K	2.7/53K/55K	6.3/150K/152K	6.7/76K/77K	2.5/54K/55K	6.6/78K/81K	8.1/94K/98K	1.1/46K/47K	10/74K/76K	6.6/72K/74K	1.9/70K/72K
HMAC-Whirlpool	512	512		72/5078/24K	73/5163/24K	178/12K/59K	98/7161/33K	86/6351/29K	103/7563/35K	462/31K/151K	36/2488/12K	97/6698/31K	136/9409/44K	100/6810/32K
HMAC-MD4	512	128		4.7/638/1961	4.5/640/1881	6.9/1059/3315	6.4/770/2396	4.7/682/2013	6.4/772/2385	4.2/850/2365	6.3/603/2186	6.9/934/2531	6.7/948/2682	4.7/609/1880
HMAC-MD5	512	128		7.3/804/2634	6.8/799/2470	8.9/1186/3822	9.4/963/3186	7.2/853/2705	9.4/968/3175	5.8/955/2766	10/845/3107	10/1150/3440	10/1204/3607	7.4/817/2709
HMAC-RIPE-MD	512	160		18/1571/5608	16/1440/5045	33/2969/10K	27/2083/7656	23/1952/7060	26/2095/7682	29/4318/14K	24/1793/7028	24/2102/10K	19/1872/6018	21/1793/6556
HMAC-SHA-0	512	160		15/1380/4619	12/1184/3905	30/2749/8858	23/1915/7065	16/1454/4955	23/1922/6723	54/6839/39K	9.8/908/3044	12/1267/3926	12/1395/4092	13/1196/3830
HMAC-SHA-1	512	160		15/1346/4697	13/1211/4029	27/2566/8550	25/1987/6702	16/1445/5255	24/1989/6722	17/2004/6872	11/934/3402	13/1361/4558	9.8/1249/3947	12/1162/3947
HMAC-SHA-2	512	256		39/2899/11K	40/2967/11K	71/5460/20K	40/3129/12K	40/3017/11K	39/3186/11K	60/4568/17K	29/2162/8364	34/2919/10K	29/2387/8478	33/2479/9469
	512	384		84/625/34K	75/599/30K	188/1410/75K	124/996/50K	84/660/34K	132/806/53K	178/871/74K	16/193/7261	65/686/28K	92/745/36K	72/488/29K
	512	512		84/614/34K	75/584/30K	188/1399/75K	124/993/51K	84/654/34K	132/801/53K	178/859/73K	16/195/7268	65/681/28K	92/722/37K	72/487/29K
HMAC-Tiger	512	192		21/1758/6526	24/1924/7117	41/3495/12K	28/2307/8489	24/1964/7182	26/2145/7836	73/6421/25K	5.2/500/1776	24/2130/7050	29/2513/8382	20/1654/6037
CBCMAC-Rijndael	128	128		26/616/2056	24/548/1919	58/1633/4833	26/958/2789	24/694/1661	27/960/3407	55/2220/11K	20/523/1636	22/649/2958	30/1135/3151	31/513/1124
CBCMAC-DES	64	64		61/973/2924	62/980/3017	91/1778/4848	72/1009/1932	62/971/1560	69/1036/3198	91/1393/8338	39/628/2153	56/902/2723	61/887/2713	54/747/1283
CBCMAC-Shacal	512	160		31/829/2886	28/846/2745	63/1870/5926	67/751/2473	31/877/1577	74/828/4725	43/1579/5472	37/984/3575	34/1017/3588	25/1008/2774	29/535/1237

The figures in the entries are of MAC/keysetup/keysetup+finalize times, where the MAC time is measured in cycles/byte and the keysetup and keysetup+finalize times are measured in cycles.

Table 36: Asymmetric Encryption Performance Results

Primitive Name	Key Size	Machine											
		PIII Claimed	PIII/Linux	PIII/MS	PI/MMX	Pentium4	Pentium2	Xeon	486	Alpha	Sparc V9	Mac	AMD
EPOC-2	1152		30M/7989K/366M	30M/8024K/569M	47M/10M/471M	83M/18M/1150M	25M/6235K/389M	23M/5866K/357M	–/–/–	–/–/–	–/–/–	–/–/–	18M/4654K/299M
RSA-OAEP	1024		2026K/42M/1654M	2583K/52M/2214M	4876K/97M/3916M	3028K/62M/2616M	2572K/53M/1935M	2981K/61M/2587M	4638K/90M/3776M	–/–/–	3872K/77M/3106M	2255K/46M/1956M	2289K/48M/2024M
ACE encrypt			57M/25M/3194M	–/–/–	52M/22M/2783M	63M/27M/3416M	–/–/–	51M/20M/3237M	90M/38M/4724M	–/–/–	268M/113M/13G	–/–/–	33M/14M/1748M
EPOC1			6312K/11M/46M	6433K/11M/47M	10M/17M/75M	18M/30M/123M	7599K/9853K/59M	18M/30M/122M	19M/32M/128M	1343K/2376K/10M	66M/106M/421M	–/–/–	4155K/6930K/31M
EPOC2			8254K/11M/46M	8477K/12M/48M	12M/19M/75M	24M/32M/123M	8538K/10M/47M	23M/32M/122M	23M/34M/128M	2112K/2461K/10M	75M/114M/418M	–/–/–	5823K/7270K/31M
EPOC3			8279K/4882K/46M	8554K/5158K/48M	12M/7689K/75M	23M/11M/123M	7981K/3895K/43M	23M/11M/121M	23M/12M/128M	2089K/1067K/10M	73M/40M/418M	–/–/–	5808K/2728K/31M
PSEC1			7952K/6322K/2306	9703K/7341K/–	9836K/8338K/2169	13M/10M/4776	19M/8373K/5295	12M/10M/4696	12M/10M/2268	7136K/6280K/3612	22M/20M/5017	–/–/–	7480K/6087K/2993
PSEC2			7703K/6176K/2380	–/–/–	9646K/8347K/2162	12M/10M/4710	10M/8197K/4534	12M/9959K/4538	12M/11M/2326	7040K/6197K/3742	22M/20M/5061	–/–/–	7301K/5757K/2992
PSEC3			7952K/76K/2370	–/–/–	9864K/84K/2154	13M/108K/4716	10M/72K/4546	13M/111K/4471	12M/121K/2228	7147K/40K/3838	22M/73K/4980	–/–/–	7495K/76K/2918

The figures in the entries are of encrypt/decrypt/key generation times. All of them are measured in cycles/invocation.

Table 37: Signature Performance Results

Primitive Name	Key Size	Machine											
		PIII Claimed	PIII/Linux	PIII/MS	PI/MMX	Pentium4	Pentium2	Xeon	486	Alpha	Sparc V9	Mac	AMD
ECDSA-GF(2^p)	160		4775K/6085K/4669K	4639K/5950K/4494K	7838K/9947K/7692K	6100K/7862K/5936K	4620K/5950K/4536K	6261K/8098K/6080K	8543K/11M/8481K	2523K/3212K/2466K	6745K/8680K/6656K	5480K/7000K/5600K	4025K/5251K/3909K
ECDSA-GF(2^{163})	163		5061K/6809K/4852K	5051K/6652K/4932K	8223K/10M/7966K	5922K/7981K/5669K	5162K/6912K/4970K	6161K/8217K/5840K	7507K/10M/7317K	3590K/4791K/3465K	4811K/6435K/4631K	4880K/6560K/4711K	4578K/6153K/4363K
KCDSA-GF(2^p)	160		4724K/6085K/4688K	4596K/5869K/4524K	7785K/9894K/7737K	6024K/7769K/5984K	4410K/5740K/4582K	6139K/7993K/6129K	8533K/10M/8448K	2512K/3184K/2482K	6770K/8630K/6744K	5640K/7280K/5651K	4013K/5149K/3933K
KCDSA-GF(2^{163})	163		5024K/6692K/4888K	5034K/6716K/4962K	8157K/10M/7999K	5939K/7854K/5707K	5141K/6912K/5020K	6094K/8068K/5892K	7431K/10M/7326K	3562K/4674K/3470K	4680K/6300K/4617K	4800K/6280K/4776K	4530K/6068K/4400K
Esign	1152		586K/167K/368M	604K/169K/460M	847K/319K/799M	996K/253K/545M	525K/190K/400M	527K/259K/483M	1221K/366K/863M	416K/85K/174M	1035K/300K/536M	890K/250K/688M	405K/148K/293M
RSA-PSS	1024		42M/2029K/1334M	52M/2562K/1443M	97M/4894K/3833M	62M/3019K/1937M	53M/2562K/1438M	62M/2979K/1929M	90M/4623K/–	–/–/–	77M/3520K/2994M	46M/2240K/1825M	48M/2288K/1353M
SFLASHv2			1385K/328K/731M	–/–/–	5106K/765K/2929M	–/–/–	1761K/344K/734M	1299K/269K/737M	–/–/–	1353K/143K/540M	1885K/312K/928M	1860K/289K/848M	2030K/385K/1020M
ACE sign			24M/19M/7313M	–/–/–	22M/16M/7314M	26M/20M/9645M	–/–/–	25M/19M/8790M	34M/28M/9710M	–/–/–	95M/84M/25G	–/–/–	14M/7113K/5485M
ECDSA			952K/2538K/805K	1211K/3207K/1023K	1667K/4589K/1491K	1971K/5415K/1758K	1236K/3251K/1042K	1463K/3870K/1241K	1446K/3758K/1194K	–/–/386K	809K/2083K/670K	1164K/3024K/948K	997K/2663K/841K
Esign (e=8)	1152		2046K/370K/104M	2099K/370K/113M	2362K/577K/169M	4434K/936K/269M	2074K/296K/95M	4107K/938K/265M	2861K/997K/279M	978K/113K/32M	5069K/2965K/872M	–/–/–	1818K/275K/74M
FLASH			2050K/291K/1092M	2721K/308K/1395M	6362K/973K/3220M	2406K/280K/1116M	7067K/–/2106M	2294K/244K/1129M	5425K/753K/3021M	1506K/197K/732M	3132K/394K/1058M	2416K/426K/854M	3296K/412K/1456M
QUARTZ			5500M/121K/1555M	6104M/115K/1826M	12G/315K/4444M	6261M/144K/3167M	12G/220K/4257M	7214M/136K/2823M	19G/347K/5926M	2855M/129K/1111M	5760M/191K/4703M	–/–/–	6406M/123K/2028M
SFLASH			1700K/195K/786M	2303K/259K/952M	5239K/640K/3676M	2387K/395K/1192M	2330K/269K/943M	2308K/337K/1132M	4183K/489K/2799M	1309K/164K/465M	2767K/567K/3753M	1800K/241K/735M	2910K/212K/1016M
RSA-PSS			56M/981K/2182M	62M/1208K/2388M	106M/2089K/4005M	82M/1587K/3206M	115M/1255K/3412M	76M/697K/2967M	95M/1924K/3623M	40M/779K/1558M	132M/2424K/4968M	48M/968K/1862M	53M/1171K/2078M

The figures in the entries are of sign/verify/key generation times. All of them are measured in cycles/invocation.

Table 38: Identification Performance Results

Primitive Name	Key Size	Machine											
		PIII Claimed	PIII/Linux	PIII/MS	PI/MMX	Pentium4	Pentium2	Xeon	486	Alpha	Sparc V9	Mac	AMD
GPS			312M/2864K/4691K/328/5475K	326M/2860K/4707K/297/5482K	439M/5333K/8541K/389/9699K	757M/9201K/14M/491/16M	249M/2899K/4571K/309/5194K	236M/2676K/4202K/343/4755K	–/–/–/–/–	71M/811K/1056K/144/1200K	–/–/–/–/–	–/–/–/–/–	183M/2138K/3340K/241/3796K

The figures in the entries are of parameter generation/key generation/commitment/answer/verification time. All of them are measured in cycles/invocation.

Table 39: Legacy Block Ciphers Performance Results by Processor and Compiler

Primitive Name	Key Size	Machine									
		PIII/Linux Coppermine gcc 2.95.2	PIII/Linux Coppermine gcc 3.1.1	PIII/Linux Coppermine gcc 3.0.3	PIII/Linux Tualatin gcc 2.96	PIII/Linux Katmai gcc 3.1.1	PIII/Linux Katmai gcc 3.3	PIII/Linux Coppermine gcc-egcs	PIII/Win Visual C 6.0	PIII/Win Intel C	PIII/Win gcc 2.95.3
Idea	128	61/61/878	57/56/450	58/58/492	60/60/1231	56/56/669	55/55/690	67/68/811	64/66/585	53/52/395	59/59/851
Khazad	128	45/46/834	39/40/765	40/40/807	43/44/813	40/41/814	40/41/829	46/46/884	38/38/823	41/41/853	45/45/823
Khazad-tweak	128	49/49/841	39/40/766	42/42/808	44/44/880	40/41/906	40/41/917	46/46/978	46/46/825	44/44/892	45/45/822
Misty1	128	47/48/257	47/47/209	47/47/210	50/49/296	48/48/208	48/48/243	49/48/340	61/62/195	42/43/125	47/47/243
Safer++	128	152/196/1835	178/169/1504	170/169/1505	279/316/1979	177/168/1526	181/169/1943	184/245/1952	143/161/2600	185/215/1692	158/193/2050
Cs-cipher	128	163/150/2851	184/191/2691	156/140/2873	392/434/12K	183/190/2686	294/193/2779	172/155/4837	414/432/5077	411/433/3208	166/149/2920
Hierocrypt-L1	128	54/57/36K	44/47/34K	48/51/36K	52/54/48K	44/47/34K	43/46/34K	53/54/66K	65/70/48K	48/51/52K	54/58/36K
Nimbus	128	34/34/12K	38/41/13K	38/42/13K	37/44/14K	38/41/13K	36/40/13K	37/36/13K	42/41/16K	41/42/17K	36/34/12K
Nush	128	95/66/1438	48/46/1422	44/38/1476	116/78/1497	48/46/1417	46/47/1415	99/73/1452	55/70/1402	47/36/1472	97/67/1489
CAST-128	128	37/38/1021	31/30/933	31/32/916	34/38/1085	30/30/938	31/30/943	52/58/992	48/48/1081	34/34/774	38/37/1009
DES	56	60/60/922	59/59/886	59/59/905	62/62/1024	59/59/883	59/60/910	61/61/969	72/72/952	68/68/960	62/62/929
Triple-DES	168	161/160/2753	155/156/2656	158/158/2726	162/162/3107	154/155/2636	157/156/2728	182/181/2899	186/187/2862	182/183/2858	160/160/2744
Kasumi	128	90/88/410	74/75/339	73/74/297	93/90/406	124/125/345	124/126/333	96/95/391	143/141/565	97/97/267	91/92/436
RC5 (12 Rounds)	64	24/24/1508	20/19/1235	20/19/1308	21/21/2604	19/19/1202	35/37/1394	35/38/1605	33/34/1711	35/31/1494	24/24/1500
Skipjack	80	160/149/18	115/121/16	121/120/16	166/183/22	114/120/16	115/121/17	191/197/18	256/409/11	133/150/13	150/152/17

		PII/Win Visual C 6.0	PII/Win gcc 2.95.3	PIV/Linux Northwood gcc 2.95.2	PIV/Linux Northwood gcc 3.1.1	Xeon Model 1 gcc 3.2	Xeon Model 0 egcs 2.91.66	Xeon Model 0 gcc 2.96	AMD Duron Win XP VC 7.0 (.NET)	AMD Duron Win XP gcc 2.95.3	AMD Athlon Linux gcc 2.96
Idea	128	68/67/802	59/59/1061	105/105/472	100/100/478	100/100/459	118/120/521	103/102/448	67/67/517	57/58/435	55/55/434
Khazad	128	47/47/1098	45/45/1023	52/52/1102	48/48/904	48/48/898	54/54/1175	45/45/963	48/46/837	45/45/866	43/43/904
Khazad-tweak	128	49/48/1105	45/45/1011	50/50/1089	45/45/916	46/46/934	50/50/1129	44/45/1026	51/51/889	45/45/861	43/43/948
Misty1	128	72/68/195	48/48/244	59/56/147	62/56/131	58/53/130	61/60/148	56/52/143	59/60/120	52/52/174	55/53/151
Safer++	128	145/203/2578	158/193/2199	198/278/3896	154/171/3302	156/171/1731	178/345/4598	168/187/3886	150/233/2322	187/208/1999	182/182/1344
Cs-cipher	128	642/448/5114	166/150/2922	323/133/5227	164/161/6312	143/169/7454	231/137/8629	137/131/6513	239/239/4522	248/189/3841	196/185/3535
Hierocrypt-L1	128	65/68/48K	56/58/36K	69/68/51K	69/69/57K	70/72/57K	76/72/61K	76/72/55K	39/42/44K	48/51/34K	43/47/32K
Nimbus	128	41/40/16K	36/34/12K	83/82/31K	93/99/34K	92/93/33K	83/82/32K	89/98/33K	39/42/20K	45/46/16K	45/51/16K
Nush	128	55/71/1389	97/67/1435	44/26/1113	44/26/1103	44/25/1094	48/29/1276	47/29/1118	34/27/1707	34/30/1716	27/25/1659
CAST-128	128	48/47/1107	38/37/1016	42/42/1054	39/39/1024	40/39/1007	88/93/1194	43/42/1093	36/35/1073	33/33/1192	30/31/1005
DES	56	73/71/943	62/62/932	73/73/879	61/61/816	60/60/819	73/73/954	61/61/829	68/69/856	54/54/782	55/54/773
Triple-DES	168	184/184/2822	161/160/2755	192/192/2631	158/158/2477	158/158/2453	387/382/2817	157/157/2517	189/181/2565	141/141/2312	143/144/2337
Kasumi	128	211/209/544	151/150/439	162/150/396	157/148/253	192/163/257	212/213/383	152/161/260	123/122/246	132/131/319	88/85/198
RC5 (12 Rounds)	64	32/33/1778	24/24/1504	32/31/2376	30/30/1420	30/30/1420	38/54/1863	31/32/3399	26/27/2203	26/24/1236	19/19/1477
Skipjack	80	354/480/10	150/152/19	243/327/23	108/107/21	110/107/20	240/267/22	159/176/25	560/864/8.4	182/173/15	149/144/14

		Alpha cc	Alpha gcc 2.97	Sun/Sparc V9 450MHz gcc 3.0.4+3.2.1	Sun/Sparc 248MHz gcc 2.95.3	Sun/Sparc V9 338MHz cc	Sun/Sparc V9 338MHz gcc 3.0.4	Sun/Sparc V9 400MHz cc	Sun/Sparc V9 400MHz gcc 3.0.4	Sun 450MHz cc	Sun 333MHz gcc 3.2.1
Idea	128	55/56/647	57/57/399	88/88/600	88/88/600	185/188/697	90/90/619	179/180/658	88/88/599	186/189/678	91/91/672
Khazad	128	25/25/384	19/19/368	35/36/971	39/39/1096	35/35/832	35/35/983	34/33/799	34/35/957	36/36/921	36/37/1002
Khazad-tweak	128	25/26/385	19/19/373	36/36/1019	39/39/1132	35/35/834	35/35/1034	33/33/808	34/35/991	36/36/921	39/40/1148
Misty1	128	47/49/135	36/35/90	51/52/159	56/56/150	35/36/125	54/55/151	33/34/117	53/53/146	44/47/121	56/56/158
Safer++	128	163/302/1968	85/87/1876	108/117/3013	116/142/3141	151/178/2334	111/120/3128	147/171/2253	104/117/3004	116/158/2173	106/120/3081
Cs-cipher	128	374/362/3887	115/117/3047	139/160/4054	124/156/4400	131/168/4456	145/166/4168	126/161/4275	141/162/4061	132/165/3350	136/157/4666
Hierocrypt-L1	128	23/28/39K	35/34/37K	63/68/33K	64/72/34K	64/73/42K	64/69/35K	62/70/41K	61/66/33K	62/66/38K	65/73/33K
Nimbus	128	19/19/7219	36/36/13K	77/76/25K	78/76/25K	66/72/24K	79/78/26K	64/68/22K	77/76/25K	65/71/23K	79/78/25K
Nush	128	31/27/942	30/21/1212	31/26/2177	30/27/2137	38/34/2865	31/27/2230	37/32/2744	31/26/2167	37/30/2813	33/29/2342
CAST-128	128	43/42/608	36/37/662	38/38/992	38/38/1067	34/35/658	40/41/1305	32/33/615	38/38/1217	34/34/674	39/40/1030
DES	56	37/37/603	39/39/596	68/68/888	63/63/899	55/56/842	65/65/913	54/54/819	63/63/897	56/53/798	64/65/904
Triple-DES	168	99/99/1873	101/101/1814	170/168/2641	171/171/2698	131/130/2527	182/181/2731	126/126/2455	172/171/2654	140/140/2336	176/175/2735
Kasumi	128	104/104/180	88/88/253	97/97/289	100/98/278	99/101/381	100/101/298	96/98/365	97/98/289	106/106/356	107/108/344
RC5 (12 Rounds)	64	28/27/1109	28/25/1392	23/23/1709	24/24/1555	28/31/1613	24/23/1772	27/30/1574	23/22/1726	30/30/1663	25/25/2078
Skipjack	80	396/421/32	74/78/24	55/55/32	60/58/32	90/92/117	55/56/33	88/90/105	53/54/32	127/128/119	55/57/32

The figures in the entries are of encryption/decryption/keysetup times, where the encryption and decryption times are measured in cycles/byte and the key setup time is measured in cycles/keysetup.

Table 40: Normal and High Block Ciphers Performance Results by Processor and Compiler

Primitive	Key	Machine									
Name	Size	PIII/Linux Coppermine gcc 2.95.2	PIII/Linux Coppermine gcc 3.1.1	PIII/Linux Coppermine gcc 3.0.3	PIII/Linux Tualatin gcc 2.96	PIII/Linux Katmai gcc 3.1.1	PIII/Linux Katmai gcc 3.3	PIII/Linux Coppermine gcc-egcs	PIII/Win Visual C 6.0	PIII/Win Intel C	PIII/Win gcc 2.95.3
Camellia	128	38/37/339	35/35/336	35/35/313	35/35/318	35/35/415	35/35/403	40/40/366	65/65/411	47/47/334	37/37/337
	192	49/48/489	45/45/462	47/47/447	46/46/420	45/45/556	45/45/558	51/51/487	84/84/605	59/59/434	49/48/445
	256	49/48/497	45/45/472	47/47/468	46/46/429	45/45/571	45/45/574	52/51/496	84/84/629	59/59/447	49/49/454
Camellia 2nd impl	128	38/38/327	35/35/304	35/35/285	36/36/297	35/35/342	35/36/343	40/40/337	64/65/387	45/45/265	37/37/298
	192	48/49/471	45/45/446	45/45/390	46/47/410	45/45/527	46/45/523	53/53/465	84/85/594	55/56/395	49/48/426
	256	49/49/479	45/46/454	46/46/401	47/47/415	45/45/538	45/45/542	53/52/473	84/85/616	55/56/407	49/48/435
RC6	128	31/25/1264	18/17/1054	18/17/1109	26/22/1162	17/17/1063	31/29/1524	33/34/2030	37/33/2073	29/29/1740	24/25/1040
(20 rounds)	192	37/25/1271	18/17/1176	18/17/1175	26/22/1333	18/17/1187	32/29/1537	33/34/2034	37/34/2076	30/30/1753	32/25/1258
	256	25/25/1270	18/17/1168	18/17/1184	25/22/1343	18/17/1180	32/29/1548	33/34/2059	36/34/2066	30/30/1758	25/25/1262
Safer++	128	52/65/1606	50/64/1341	51/63/1333	50/64/1750	50/64/1346	50/65/1717	51/66/1732	47/60/2258	46/78/1481	50/63/1464
	256	72/91/1823	69/91/1805	70/89/1815	68/88/2419	69/91/1808	69/90/2344	70/98/2241	65/84/3121	63/110/2047	68/89/1865
Anubis	128	38/38/4454	38/38/3611	40/40/3783	41/40/3845	37/37/3615	37/37/3550	40/40/4501	37/37/4325	43/43/3727	37/37/4527
	160	41/40/5775	41/40/4663	43/43/4801	43/43/5040	41/40/4668	40/40/4519	43/43/5821	40/40/5530	46/46/4927	39/39/5852
	192	43/43/7621	43/43/6043	45/45/6125	46/46/6955	43/43/6050	43/43/5857	46/46/7657	42/42/7843	49/49/6993	42/42/7630
	224	45/45/9631	46/46/7558	48/48/7873	49/49/8732	45/45/7524	45/45/7356	48/48/9445	44/44/9508	52/52/8546	44/44/9134
	256	48/48/11K	48/49/9076	51/51/9245	51/51/10K	48/48/9139	48/48/8876	51/51/11K	47/47/11K	55/55/10K	47/47/11K
	288	51/50/13K	52/52/10K	54/54/11K	54/54/12K	51/51/10K	50/50/10K	54/54/13K	49/49/13K	58/58/12K	49/49/12K
	320	53/53/15K	54/54/12K	57/57/12K	57/57/13K	53/53/12K	53/53/12K	57/56/15K	52/52/15K	61/61/13K	51/51/15K
Grandcru	128	1628/1999/113K	1255/1526/90K	1462/1721/105K	2234/2645/160K	1250/1518/89K	1365/1687/98K	2133/2743/150K	1856/2472/131K	1294/1686/92K	1732/2019/120K
Hierocrypt-3	128	58/78/130K	52/65/126K	51/64/130K	53/71/172K	51/65/125K	52/65/126K	57/75/240K	50/56/173K	52/61/191K	53/68/131K
	192	67/92/154K	60/76/149K	60/75/154K	64/84/204K	60/76/149K	61/76/147K	67/87/284K	59/67/204K	62/73/225K	62/80/154K
	256	77/106/178K	69/88/171K	69/86/178K	72/97/235K	69/87/170K	69/88/171K	77/102/328K	67/76/236K	70/83/260K	71/92/178K
Noekeon-Dir	128	88/91/31	60/59/34	92/95/34	116/115/35	59/59/34	56/58/34	91/92/31	137/137/58	116/117/28	89/91/34
Noekeon-Ind	128	53/56/1033	44/46/660	44/45/634	45/46/852	44/45/657	45/46/668	89/63/1172	75/75/1035	43/44/732	52/57/1016
Nush	128	28/25/2554	23/20/2559	23/21/2588	29/25/2606	23/20/2549	23/20/2568	28/25/2528	28/25/2579	30/25/2585	28/25/2584
	256	28/25/2584	23/20/2592	23/21/2634	29/25/2680	23/20/2573	23/20/2616	28/25/2545	28/25/2605	30/25/2595	28/25/2597
Q	128	59/142/993	54/112/1008	55/139/1022	69/157/1036	54/111/1010	54/107/1010	64/146/1071	61/146/1109	59/140/984	59/145/1026
	256	64/159/1174	61/124/1161	60/155/1155	77/181/1155	60/123/1158	61/119/1154	71/164/1229	67/163/1213	65/156/1104	64/162/1164
SC2000	128	72/42/2732	50/49/2424	38/41/2740	60/55/6738	59/58/2672	60/59/2534	77/50/3359	133/134/6988	138/138/3972	70/42/2756
	192	83/47/3047	56/57/3315	43/46/3055	70/62/7637	67/68/3582	68/68/2823	88/57/3766	152/153/7908	159/159/4443	81/48/3073
	256	83/47/3059	56/57/3327	43/46/3071	70/62/7660	67/67/3599	68/68/2860	88/57/3795	152/153/7955	159/159/4483	81/48/3089
Mars	128	44/37/2673	31/30/2536	31/34/2527	35/38/2571	31/30/2539	45/39/2547	51/47/2520	45/42/2902	36/35/2354	43/37/2628
	192	44/37/2658	31/30/2551	31/34/2530	36/38/2582	31/30/2550	45/39/2555	51/47/2530	45/42/2920	36/35/2337	43/37/2633
	256	44/37/2670	31/30/2549	31/34/2525	35/38/2571	31/30/2553	45/39/2548	52/47/2533	45/42/2929	36/35/2336	43/37/2644
Rijndael	128	28/29/902	26/26/504	25/26/523	27/27/792	26/26/621	26/26/630	28/30/986	23/23/497	24/29/546	28/28/910
	192	34/34/989	31/31/601	30/31/647	32/32/916	31/31/766	31/31/770	33/36/1168	27/27/552	29/34/587	32/33/1018
	256	38/39/1157	35/36/949	34/35/960	37/37/1153	35/35/1067	36/36/1077	39/41/1350	32/32/780	33/39/775	37/37/1199
Seed	128	50/50/441	45/45/409	48/48/417	50/50/470	45/45/423	45/46/422	55/55/492	57/57/446	51/50/421	51/51/465
Serpent	128	73/95/2027	70/81/1577	70/81/1925	84/82/2241	70/80/1878	70/80/1818	68/81/2307	84/77/1908	59/57/1276	73/86/1904
	192	73/95/2039	70/80/1578	70/81/1924	84/82/2223	70/80/1892	70/80/1820	68/81/2321	84/77/1931	59/57/1272	73/86/1919
	256	73/95/2032	70/80/1566	70/81/1917	84/82/2218	70/80/1880	70/81/1830	68/81/2306	84/77/1907	59/57/1257	73/86/1911
Twofish	128	36/36/17K	29/26/16K	28/29/17K	30/30/13K	29/26/11K	29/26/10K	35/36/23K	31/30/16K	28/29/15K	33/36/16K
	192	36/36/19K	29/26/18K	28/29/19K	30/30/18K	29/26/13K	29/26/12K	35/36/27K	31/30/17K	30/30/18K	33/36/19K
	256	36/36/22K	29/26/20K	28/29/22K	30/30/19K	29/25/17K	29/26/16K	35/36/33K	32/30/20K	28/29/20K	33/36/23K

The figures in the entries are of encryption/decryption/keysetup times, where the encryption and decryption times are measured in cycles/byte and the key setup time is measured in cycles/keysetup.

Table 41: Normal and High Block Ciphers Performance Results by Processor and Compiler

Primitive	Key	Machine									
Name	Size	PII/Win Visual C 6.0	PII/Win gcc 2.95.3	PIV/Linux Northwood gcc 2.95.2	PIV/Linux Northwood gcc 3.1.1	Xeon Model 1 gcc 3.2	Xeon Model 0 egcs 2.91.66	Xeon Model 0 gcc 2.96	AMD Duron Win XP VC 7.0 (.NET)	AMD Duron Win XP gcc 2.95.3	AMD Athlon Linux gcc 2.96
Camellia	128	64/64/479	38/38/426	71/72/535	64/63/453	64/64/444	71/71/626	69/68/580	68/68/330	34/35/274	31/31/290
	192	84/84/672	49/49/533	96/96/815	86/86/546	82/85/537	95/95/819	90/89/633	90/91/456	44/44/386	41/41/392
	256	84/84/697	49/49/554	96/96/830	86/87/555	83/85/559	95/95/837	89/89/673	90/90/463	44/45/398	41/41/393
Camellia 2nd impl	128	65/65/405	37/37/342	72/73/525	88/90/415	88/90/408	72/72/590	95/93/432	68/69/316	35/35/265	32/32/235
	192	84/84/646	49/49/514	96/96/732	116/118/516	117/115/513	95/95/796	123/128/612	90/92/448	45/45/371	40/41/346
	256	84/84/676	49/49/526	96/96/790	114/116/539	113/115/533	95/95/799	125/131/656	90/92/455	45/45/377	42/42/361
RC6	128	36/33/2070	24/25/1060	48/45/2136	36/37/2056	37/37/2050	101/85/3740	37/36/2053	32/31/1790	22/23/947	17/17/1034
(20 rounds)	192	37/34/2029	32/25/1291	103/66/2164	37/37/2141	37/37/2188	102/86/4071	37/37/2162	32/31/1792	28/23/1056	17/18/1039
	256	35/33/1984	25/25/1291	39/45/2153	37/37/2206	37/37/2157	92/85/3885	37/37/2061	32/31/1800	23/23/1057	17/18/1032
Safer++	128	46/77/2240	50/63/1480	54/59/3249	47/58/2918	47/58/1550	48/61/4083	52/53/3420	52/75/2043	45/52/1608	41/65/1192
	256	64/108/3115	68/89/1887	74/84/4154	65/83/3998	65/83/2085	66/94/5713	71/79/4773	68/104/2835	63/73/1686	57/89/1611
Anubis	128	37/37/4389	37/37/4575	35/35/4998	36/36/3750	37/37/3793	52/52/5176	35/35/3669	46/46/5576	37/37/5552	37/37/4687
	160	39/39/5570	39/39/5899	37/37/6282	38/39/4845	39/39/5364	56/56/6730	37/37/4751	52/51/7554	37/37/7095	38/39/6119
	192	42/41/7859	42/42/7692	40/39/7695	41/41/6054	41/41/6170	59/59/7787	40/40/5949	61/63/11K	42/41/9499	42/42/7601
	224	44/44/9556	44/44/9251	41/41/9627	43/44/7338	44/44/7469	61/62/10K	42/42/7330	66/63/12K	46/49/12K	43/44/9387
	256	46/46/11K	47/47/11K	44/44/11K	46/46/8902	46/47/8844	67/67/11K	44/45/8919	71/72/16K	48/47/15K	46/46/11K
	288	49/49/13K	49/49/12K	46/46/12K	49/48/10K	50/49/10K	69/70/13K	47/47/10K	72/75/19K	53/54/18K	48/48/13K
	320	51/51/15K	52/52/15K	48/48/15K	51/51/11K	53/53/11K	74/73/15K	50/49/12K	71/71/22K	49/49/18K	51/51/15K
Grandcru	128	4208/5651/305K	1731/2024/120K	1860/2401/132K	1634/2274/124K	1675/2192/123K	2626/3113/183K	1738/2136/124K	2752/4145/192K	1640/2109/114K	1612/2106/115K
Hierocrypt-3	128	49/56/172K	54/68/131K	54/79/195K	61/63/205K	60/63/204K	57/76/219K	60/64/202K	59/71/100K	56/72/125K	52/61/116K
	192	59/66/201K	63/81/155K	63/93/229K	69/75/242K	69/74/241K	67/90/265K	70/76/239K	70/88/124K	66/84/148K	60/72/138K
	256	67/76/232K	71/93/178K	73/107/266K	80/85/277K	79/85/278K	77/101/303K	81/86/276K	89/105/147K	99/134/173K	69/83/159K
Noekeon-Dir	128	138/138/61	89/91/34	97/100/31	71/72/32	67/116/31	100/99/31	107/107/32	114/102/27	79/75/35	77/78/35
Noekeon-Ind	128	74/75/1131	52/57/1027	97/100/1494	64/66/948	63/66/974	102/100/1559	64/67/877	57/58/853	42/49/800	36/36/620
Nush	128	28/25/2556	28/25/2581	37/21/2005	31/20/1982	31/20/1978	37/21/1999	32/20/2005	24/21/3077	25/22/3134	22/20/3079
	256	28/25/2585	28/25/2613	37/21/2019	31/20/2000	31/20/1999	37/21/2027	32/20/2015	24/21/3104	25/22/3156	22/19/3088
Q	128	60/144/1128	59/145/1022	53/254/986	51/231/1028	51/231/1032	55/256/1032	51/251/1079	69/185/1236	54/189/971	48/167/954
	256	67/160/1228	64/162/1193	58/283/1153	58/256/1161	57/257/1147	60/297/1251	56/279/1165	69/205/1413	58/199/1110	54/185/1066
SC2000	128	139/139/7106	78/50/3031	96/70/5188	60/60/4661	64/64/4676	123/79/6364	77/81/5146	33/55/10K	63/37/2718	40/43/3182
	192	159/159/8008	90/57/3357	111/80/5856	68/69/6612	74/75/6552	143/89/7149	90/91/5798	45/65/10K	72/41/3023	46/48/3566
	256	159/159/8100	90/57/3386	111/81/5909	68/69/6669	73/74/6624	141/89/7240	90/91/5841	40/65/9344	72/41/3044	46/48/3593
Mars	128	45/41/2871	43/37/2635	138/160/3628	82/72/3492	81/74/3469	148/161/3306	89/102/3494	39/38/2623	45/38/2435	29/32/2411
	192	45/42/2891	43/37/2639	138/161/3635	82/72/3480	81/74/3456	146/162/3294	88/102/3410	40/38/2627	45/38/2445	29/32/2412
	256	45/41/2893	43/37/2654	137/165/3667	82/73/3599	81/74/3458	148/164/3368	88/102/3337	40/39/2692	45/38/2440	29/32/2429
Rijndael	128	22/23/612	28/28/973	26/26/1329	24/25/689	24/25/711	33/33/1525	24/25/1228	32/35/500	31/32/852	30/31/748
	192	27/27/714	32/33/1208	31/30/1367	28/29/820	28/29/839	39/38/1831	27/28/1388	39/42/511	36/38/1107	35/36/833
	256	31/31/878	37/37/1412	35/35/1651	32/38/1327	32/33/1363	43/46/2209	32/33/1546	47/46/708	41/42/1431	39/41/1091
Seed	128	56/56/447	51/51/456	63/63/467	69/69/401	69/70/385	84/81/778	84/83/638	52/51/410	42/42/406	41/41/393
Serpent	128	83/77/1872	73/86/2139	168/199/2819	154/172/2235	153/170/2230	157/162/3078	153/174/2664	68/64/1731	62/80/1675	60/77/1691
	192	83/76/1874	73/86/2148	167/199/2826	155/171/2232	152/169/2226	158/161/3111	153/174/2706	68/65/1731	62/79/1681	60/77/1704
	256	83/76/1884	73/86/2139	167/199/2828	154/171/2231	153/171/2235	157/161/3106	153/173/2650	68/65/1707	63/79/1703	60/77/1686
Twofish	128	31/33/11K	34/36/12K	66/57/12K	51/49/8871	52/49/12K	57/51/20K	51/48/15K	37/33/24K	41/47/24K	30/34/19K
	192	31/33/13K	34/36/15K	66/56/15K	52/49/11K	52/48/14K	56/51/23K	52/48/17K	37/33/27K	41/47/30K	29/34/26K
	256	31/33/17K	34/36/20K	65/57/21K	52/49/13K	52/48/16K	57/51/28K	51/48/19K	37/33/32K	41/46/37K	30/34/32K

The figures in the entries are of encryption/decryption/keysetup times, where the encryption and decryption times are measured in cycles/byte and the key setup time is measured in cycles/keysetup.

Table 42: Normal and High Block Ciphers Performance Results by Processor and Compiler

Primitive	Key	Machine									
		Alpha	Alpha	Sun/Sparc V9	Sun/Sparc	Sun/Sparc V9	Sun/Sparc V9	Sun/Sparc V9	Sun/Sparc V9	Sun	Sun
Name	Size			450MHz	248MHz	338MHz	338MHz	400MHz	400MHz	450MHz	333MHz
		cc	gcc 2.97	gcc 3.0.4+3.2.1	gcc 2.95.3	cc	gcc 3.0.4	cc	gcc 3.0.4	cc	gcc 3.2.1
Camellia	128	48/48/287	36/35/320	49/49/375	55/55/352	57/57/759	54/54/405	47/47/557	50/50/376	53/53/712	51/51/381
	192	64/64/512	48/47/407	72/72/539	74/74/556	71/73/958	70/69/585	61/61/808	66/66/546	64/64/828	75/75/540
	256	64/64/525	48/47/453	72/72/544	74/74/563	72/73/978	70/71/569	60/60/849	66/65/542	65/64/817	74/74/564
Camellia 2nd impl	128	51/51/276	44/44/311	50/46/327	55/56/293	60/59/664	53/51/335	52/52/535	50/48/306	53/52/568	51/47/323
	192	66/64/504	58/58/416	72/72/529	74/74/508	78/77/891	70/63/559	65/67/772	66/60/518	68/71/829	68/64/521
	256	66/64/521	58/58/441	72/72/521	74/73/528	79/78/910	72/65/549	65/67/799	65/60/514	68/72/807	75/75/574
RC6	128	27/24/1509	27/27/1664	69/68/1465	70/68/1581	120/123/1599	71/69/1508	116/115/1537	70/68/1432	120/121/1725	69/68/1582
(20 rounds)	192	27/24/1725	27/27/1905	174/167/6946	176/171/7267	121/124/2062	178/175/7274	117/117/1901	170/169/6856	119/121/2023	190/183/7651
	256	26/24/1727	27/27/1906	70/70/2317	69/69/2207	121/123/2070	72/72/2378	95/95/1921	70/70/2315	99/100/2047	71/70/2609
Safer++	128	33/93/1723	30/44/1677	47/70/2665	48/65/2817	36/45/2076	50/74/2800	35/43/2002	48/71/2695	37/49/1939	47/66/2758
	256	43/129/2425	42/62/2257	67/99/3656	68/91/3842	52/75/2839	69/102/3790	49/73/2720	68/100/3588	52/74/2590	67/93/3685
Anubis	128	29/29/2990	25/25/3854	33/33/7064	37/37/6250	38/38/5054	34/34/6611	37/37/4885	33/33/6341	35/34/5011	36/36/6188
	160	31/30/3950	27/27/4993	35/35/8874	38/38/7525	40/40/6614	36/37/8510	38/38/6329	36/36/8275	37/37/6573	38/39/7659
	192	32/32/4803	29/29/6274	37/37/11K	40/41/9652	43/43/8424	39/40/10K	42/41/8088	38/38/10K	39/39/8337	41/41/10K
	224	34/34/5715	31/31/7755	40/40/12K	43/43/11K	45/45/10K	41/42/12K	43/43/9989	40/40/12K	42/42/11K	43/43/12K
	256	36/36/8234	33/33/9237	42/42/16K	45/45/14K	48/48/12K	44/44/15K	46/46/12K	42/43/14K	44/44/14K	46/46/14K
	288	37/37/9954	35/35/10K	51/51/18K	50/49/16K	49/49/14K	46/46/18K	47/48/14K	45/45/17K	47/47/16K	49/49/17K
	320	39/39/10K	36/37/12K	53/53/21K	51/51/19K	54/53/16K	49/49/21K	52/52/16K	47/47/20K	50/49/19K	50/50/20K
Grandcru	128	1326/1754/93K	1028/1278/72K	1760/2247/125K	1867/2312/139K	1895/2719/133K	1809/2321/129K	1815/2638/130K	1762/2250/125K	2029/2896/139K	1574/2166/117K
Hierocrypt-3	128	33/39/131K	34/39/136K	81/116/128K	80/99/123K	62/81/162K	90/117/126K	59/78/153K	87/113/122K	62/82/138K	82/118/118K
	192	38/46/153K	39/46/161K	103/132/147K	93/111/145K	75/90/192K	96/140/149K	71/87/183K	94/134/140K	80/88/163K	101/130/141K
	256	44/52/176K	44/52/185K	109/162/170K	109/130/168K	83/106/218K	120/156/171K	80/103/208K	117/153/169K	84/108/217K	109/145/161K
Noekeon-Dir	128	108/110/40	72/76/28	77/76/15	74/76/25	73/62/15	79/78/15	70/60/15	76/75/15	55/54/16	85/87/16
Noekeon-Ind	128	46/46/647	41/42/635	37/36/547	40/39/648	49/49/802	38/37/559	47/47/780	38/36/546	40/40/703	39/39/598
Nush	128	23/15/1305	27/16/2145	31/26/3985	35/26/3900	28/28/5277	31/27/4081	27/27/5127	31/26/3975	24/21/4915	29/29/4272
	256	23/15/1322	27/16/2161	30/26/4001	35/26/3932	28/28/5138	32/27/4080	27/27/5002	31/26/4000	24/21/4985	29/29/4285
Q	128	35/174/764	42/165/760	63/180/895	55/206/880	52/150/888	65/190/896	51/146/856	64/181/886	47/87/930	63/204/980
	256	39/195/861	47/184/852	70/201/1021	61/230/1014	56/167/1058	73/212/1039	54/162/1017	70/202/1021	56/98/1150	69/229/1132
SC2000	128	59/63/4053	32/31/3307	35/32/4955	31/32/4885	31/33/8004	37/33/5101	29/31/7733	36/32/4977	31/31/7587	48/52/3866
	192	66/71/4491	36/36/3701	41/37/5483	37/37/5461	35/37/8948	42/38/5676	34/36/8721	41/37/5532	35/35/8823	55/60/4862
	256	66/71/4515	37/37/3730	41/37/5523	37/37/5574	35/38/9013	42/38/5757	34/36/8720	41/37/5549	35/36/8638	55/61/4865
Mars	128	33/29/2924	37/33/3189	48/49/2865	51/52/2863	77/76/3152	50/51/2987	74/74/2975	49/50/2884	74/73/2925	55/55/3127
	192	33/29/2929	37/33/3185	48/49/2872	51/52/2884	76/76/3102	50/51/2984	75/74/3004	49/50/2891	73/73/2947	55/56/3153
	256	33/29/2934	37/33/3191	49/49/2904	51/52/2931	77/76/3140	50/51/3054	75/73/3040	49/50/2928	73/72/2998	55/56/3179
Rijndael	128	17/17/493	19/19/756	31/29/684	31/31/705	24/21/557	31/32/673	22/21/532	30/31/650	21/22/453	29/30/655
	192	20/21/449	22/22/847	36/35/783	38/36/724	26/26/568	37/37/770	25/25/542	37/36/734	25/28/463	35/36/796
	256	24/24/663	25/25/1044	42/41/1122	43/43/1192	29/31/779	44/43/1231	28/30/750	43/42/1186	29/32/753	43/42/1245
Seed	128	44/44/358	40/40/340	35/35/381	36/36/447	39/38/399	39/39/483	36/36/368	36/36/453	34/34/353	39/37/396
Serpent	128	54/41/1383	67/54/1904	67/61/1563	69/65/1626	66/55/1716	70/63/1635	63/53/1645	68/61/1586	57/53/1708	66/61/1728
	192	54/41/1381	67/54/2082	67/61/1577	69/65/1639	65/55/1720	70/63/1658	63/53/1665	68/61/1606	57/54/1716	66/61/1737
	256	54/41/1376	67/54/1900	67/61/1575	69/65/1645	65/55/1719	70/63/1638	64/53/1654	69/61/1601	57/53/1710	66/61/1739
Twofish	128	24/24/8661	20/20/10K	40/41/15K	39/40/14K	29/28/15K	43/44/16K	28/28/14K	40/42/15K	28/28/15K	38/39/15K
	192	24/24/12K	20/20/13K	40/42/19K	41/40/18K	29/29/19K	42/44/20K	29/28/18K	41/42/19K	28/27/18K	38/39/20K
	256	24/24/23K	20/20/17K	40/41/24K	40/40/22K	30/29/23K	42/42/26K	28/28/22K	43/42/24K	28/27/23K	38/39/24K

The figures in the entries are of encryption/decryption/keysetup times, where the encryption and decryption times are measured in cycles/byte and the key setup time is measured in cycles/keysetup.

Table 43: Block Ciphers with 160, 192 and 256-Bit Blocks by Processor and Compiler

Primitive Name	Key Size	Block Size	Machine									
			PIII/Linux Coppermine gcc 2.95.2	**PIII/Linux Coppermine gcc 3.1.1**	**PIII/Linux Coppermine gcc 3.0.3**	**PIII/Linux Tualatin gcc 2.96**	**PIII/Linux Katmai gcc 3.1.1**	**PIII/Linux Katmai gcc 3.3**	**PIII/Linux Coppermine gcc-egcs**	**PIII/Win Visual C 6.0**	**PIII/Win Intel C**	**PIII/Win gcc 2.95.3**
SHACAL-1	512	160	29/26/78	34/27/445	29/27/78	31/34/515	34/27/75	34/27/498	30/31/439	44/65/839	27/31/715	30/26/447
SHACAL-2	512	256	45/46/745	46/45/737	45/47/765	46/48/891	46/45/871	46/45/885	44/43/793	60/59/1187	46/47/748	44/46/753
RC6 (20 rounds)	256	256	85/89/13K	61/58/10K	60/62/11K	86/85/17K	61/58/10K	59/59/11K	96/88/15K	120/120/17K	125/118/12K	85/83/12K
Rijndael	192	256	34/39/2426	36/35/1785	35/35/2609	37/38/2811	36/34/2001	36/35/1999	33/39/2476	33/32/1882	40/39/2068	34/39/2454
	256	256	33/37/3053	36/36/2298	35/35/3255	36/35/3575	36/36/2538	36/36/2560	33/37/3146	34/33/2362	35/36/2798	34/37/3105

			PII/Win Visual C 6.0	**PII/Win gcc 2.95.3**	**PIV/Linux Northwood gcc 2.95.2**	**PIV/Linux Northwood gcc 3.1.1**	**Xeon Model 1 gcc 3.2**	**Xeon Model 0 egcs 2.91.66**	**Xeon Model 0 gcc 2.96**	**AMD Duron Win XP VC 7.0 (.NET)**	**AMD Duron Win XP gcc 2.95.3**	**AMD Athlon Linux gcc 2.96**
SHACAL-1	512	160	45/56/64	30/26/80	63/36/177	59/37/457	57/37/423	81/62/402	62/37/173	37/37/54	27/25/58	28/24/318
SHACAL-2	512	256	60/58/1277	44/46/888	55/60/933	51/58/1015	52/59/1075	79/64/930	51/54/857	56/52/861	38/40/597	40/40/633
RC6 (20 rounds)	256	256	125/125/18K	85/83/12K	190/171/23K	132/123/21K	133/120/21K	173/174/25K	133/133/23K	101/101/18K	78/79/15K	58/60/12K
Rijndael	192	256	33/31/2145	34/39/2558	42/48/4735	35/35/2512	35/35/2557	42/47/3778	35/37/3841	50/46/2291	50/49/2722	50/50/2430
	256	256	33/32/2705	34/38/3254	35/44/5982	38/36/3172	34/36/3252	35/43/4797	58/36/4834	53/47/2985	52/56/3561	53/53/3089

			Alpha cc	**Alpha gcc 2.97**	**Sun/Sparc V9 450MHz gcc 3.0.4+3.2.1**	**Sun/Sparc 248MHz gcc 2.95.3**	**Sun/Sparc V9 338MHz cc**	**Sun/Sparc V9 338MHz gcc 3.0.4**	**Sun/Sparc V9 400MHz cc**	**Sun/Sparc V9 400MHz gcc 3.0.4**	**Sun 450MHz cc**	**Sun 333MHz gcc 3.2.1**
SHACAL-1	512	160	35/27/60	30/27/44	34/32/288	33/31/259	37/27/152	35/33/146	36/27/141	34/32/136	30/27/257	34/32/289
SHACAL-2	512	256	37/33/671	30/31/706	37/41/1169	37/42/1191	39/38/1267	39/43/1212	38/37/1237	37/41/1176	38/38/1251	37/42/1184
RC6 (20 rounds)	256	256	12/11/2716	12/11/5511	100/100/19K	98/100/18K	–/–/–	103/103/20K	–/–/–	101/100/19K	–/–/–	110/110/15K
Rijndael	192	256	24/24/1818	–/–/–	46/47/3001	44/45/3052	32/30/2273	47/50/3086	31/29/2159	46/49/2957	30/30/2301	47/55/2759
	256	256	24/24/2326	–/–/–	45/56/3629	47/45/3801	33/33/2698	47/51/3763	31/32/2584	46/49/3610	33/32/2815	48/54/3441

The figures in the entries are of encryption/decryption/keysetup times, where the encryption and decryption times are measured in cycles/byte and the key setup time is measured in cycles/keysetup.

Table 44: Stream Ciphers Performance Results by Processor and Compiler

Primitive Name	Key Size	IV Size	Machine									
			PIII/Linux Coppermine gcc 2.95.2	PIII/Linux Coppermine gcc 3.1.1	PIII/Linux Coppermine gcc 3.0.3	PIII/Linux Tualatin gcc 2.96	PIII/Linux Katmai gcc 3.1.1	PIII/Linux Katmai gcc 3.3	PIII/Linux Coppermine gcc-egcs	PIII/Win Visual C 6.0	PIII/Win Intel C	PIII/Win gcc 2.95.3
BMGL (m=16)	128		1674/2455	1501/2355	1483/2460	1854/7234	1492/2541	1475/2541	2834/2340	1641/3258	1484/2762	1631/2336
2nd impl	128		540/193K	426/190K	463/214K	533/203K	437/188K	424/183K	510/205K	438/184K	416/189K	506/195K
Snow	128		8.2/1319	6.8/1199	6.7/1179	11/1981	6.8/1207	6.8/1224	7.8/1451	7.5/1757	6.5/1424	8.3/1328
	256		8.2/1333	6.8/1243	6.7/1206	11/2059	6.8/1264	6.8/1285	7.8/1461	7.5/1845	6.4/1444	8.3/1352
Sober-t16	128		61/1753	40/1074	40/1119	68/1635	40/1070	42/1033	67/1720	49/1414	44/1188	62/1635
	256		61/2379	40/1546	40/1604	67/2236	40/1546	42/1456	67/2263	49/1993	44/1604	62/2234
Sober-t32	128		21/923	22/842	20/838	25/900	26/1061	25/1068	21/949	24/1105	23/1057	21/913
	256		21/1138	22/1025	20/1032	26/1118	25/1296	25/1267	21/1152	24/1351	23/1293	21/1119
Leviathan	128		13/78K	10/79K	12/78K	12/113K	10/79K	10/79K	16/77K	15/112K	10/108K	13/79K
	192		12/78K	10/78K	12/78K	12/113K	10/78K	10/79K	16/77K	15/111K	10/108K	13/79K
	256		13/78K	10/79K	12/78K	12/113K	10/78K	10/79K	16/77K	15/112K	10/108K	13/79K
Lili	128		914/60	935/46	964/51	1103/72	921/46	936/47	827/65	1056/43	959/43	921/58
RC4	128		7.4/2659	7.3/3015	7.3/2885	8.1/3698	7.3/2879	7.3/3195	7.6/2963	8.2/3340	10/3469	8.0/2704
BMGL with IV	128	128	1679/2437/20	1506/2344/20	1485/2457/20	1872/7087/57	1494/2421/20	1474/2436/20	2848/2393/20	1779/3237/17	1504/2753/18	1620/2326/21
2nd impl	128	128	538/193K/20	427/183K/20	460/209K/20	526/203K/23	438/181K/20	423/178K/20	510/204K/20	439/187K/17	414/191K/18	507/195K/21
Snow	128	64	8.2/17/680	6.9/15/620	6.7/17/612	11/46/916	6.9/15/620	6.9/15/624	7.6/17/730	7.5/12/782	6.2/12/705	8.3/17/690
with IV	256	64	8.2/51/680	6.9/44/621	6.7/50/612	11/56/947	6.9/44/621	6.9/45/625	7.6/50/729	7.5/34/831	6.2/16/719	8.2/49/689
Sober-t16	128	128	61/2005/1759	40/1167/1033	40/1240/1052	67/1950/1649	40/1178/1028	42/1158/1025	67/1908/1740	49/1498/1332	44/1253/1186	61/1772/1575
with IV	256	128	61/2688/1759	40/1627/1034	40/1712/1052	68/2572/1642	40/1659/1029	42/1583/1025	67/2500/1740	49/2061/1332	44/1675/1186	61/2353/1577
Sober-t32	128	128	21/1126/1202	21/995/995	20/1005/995	26/1142/1151	26/1279/1002	25/1253/1012	21/1149/1158	24/1253/1282	23/1198/1285	21/1121/1176
with IV	256	128	21/1355/1200	22/1176/994	20/1189/994	25/1380/1153	26/1499/995	25/1481/1011	21/1379/1157	24/1504/1285	23/1439/1290	21/1353/1180
Scream-0	128	128	6.6/3603/1124	10/4059/1671	6.8/3732/1169	6.7/3845/1233	9.9/3983/1652	10/4071/1683	6.9/3603/1126	10.0/5072/2167	7.5/4002/1163	6.7/3651/1141
Scream-F	128	128	7.5/3604/1125	11/4022/1659	7.9/3732/1169	10/3854/1234	10/3990/1645	10/4018/1681	10/3598/1126	12/5077/2169	8.0/4008/1161	7.6/3661/1141
Scream-S	128	128	7.0/51K/1236	9.8/40K/1723	6.9/49K/1193	7.3/61K/1253	9.7/40K/1713	10/41K/1740	7.1/56K/1224	10/48K/2285	7.5/40K/1186	7.1/52K/1230
Seal-3.0	160	32	5.9/176K/18	6.2/161K/18	6.7/166K/18	7.9/180K/20	6.2/158K/18	6.2/161K/18	8.7/187K/17	6.9/227K/21	7.1/172K/12	6.0/183K/18

The figures in the entries are of key stream generation/keysetup times, where the key stream generation time are measured in cycles/byte and the key setup time is measured in cycles/keysetup.
For stream ciphers with initial value setup (IV) the figures in the entries are of key stream generation/keysetup/IVsetup times, where the IV setup time measured in cycles/IVsetup.

Table 45: Stream Ciphers Performance Results by Processor and Compiler

Primitive	**Key**	**IV**	**Machine**									
Name	**Size**	**Size**	**PII/Win Visual C 6.0**	**PII/Win gcc 2.95.3**	**PIV/Linux Northwood gcc 2.95.2**	**PIV/Linux Northwood gcc 3.1.1**	**Xeon Model 1 gcc 3.2**	**Xeon Model 0 egcs 2.91.66**	**Xeon Model 0 gcc 2.96**	**AMD Duron Win XP VC 7.0 (.NET)**	**AMD Duron Win XP gcc 2.95.3**	**AMD Athlon Linux gcc 2.96**
BMGL (m=16)	128		1644/3310	1624/2614	2140/4596	1992/2960	2009/2960	2501/4011	1917/4042	1945/2777	2128/2141	1867/2160
2nd impl	128		441/211K	519/194K	895/350K	706/345K	707/332K	904/356K	668/362K	504/541K	592/185K	530/172K
Snow	128		9.1/1823	8.3/1349	9.2/2725	8.6/1674	8.7/1707	9.3/2707	7.7/2601	8.3/1465	8.1/1530	12/1554
	256		8.9/1908	8.3/1382	9.2/2920	8.7/1735	8.7/1753	9.4/2777	7.8/2664	8.3/1491	8.0/1597	12/1622
Sober-t16	128		64/1672	62/1640	55/1891	45/1164	46/1146	56/1933	52/1346	46/1249	52/1264	46/1207
	256		55/2259	62/2247	55/2778	46/1610	46/1526	56/2856	52/1726	46/1646	52/1682	46/1577
Sober-t32	128		27/1247	26/1119	27/1098	28/944	28/962	29/1119	28/1112	21/1009	19/846	18/862
	256		27/1539	26/1364	27/1290	28/1140	28/1146	29/1307	27/1294	21/1226	21/1116	18/1021
Leviathan	128		15/119K	13/80K	16/76K	10/79K	10/80K	18/74K	13/98K	14/103K	11/98K	10/99K
	192		15/119K	13/80K	15/76K	10/79K	10/80K	18/74K	12/98K	14/103K	11/98K	10/99K
	256		15/120K	13/80K	16/76K	10/80K	10/80K	18/74K	12/98K	14/103K	11/98K	10/99K
Lili	128		1104/53	918/59	1023/76	987/59	966/59	1361/78	1087/61	869/44	855/53	841/50
RC4	128		7.8/3484	8.1/2824	20/2568	20/4680	20/4553	20/4675	20/2879	11/3211	11/2600	12/2675
BMGL with IV	128	128	1822/3263/16	1618/2461/21	2136/4791/20	2002/2943/21	2020/2943/20	2505/4136/20	1899/3993/20	2103/2913/15	2059/2127/20	1861/2126/19
2nd impl	128	128	445/184K/16	520/195K/21	918/353K/20	710/335K/21	706/333K/20	890/344K/20	675/368K/21	501/566K/15	593/170K/20	530/174K/18
Snow	128	64	9.0/11/772	8.3/17/689	9.2/27/1270	8.3/19/846	8.5/19/857	9.5/24/1168	7.5/23/933	8.5/14/638	8.0/15/679	13/15/725
with IV	256	64	9.0/32/819	8.3/50/689	9.2/59/1327	8.7/73/846	8.8/76/865	9.4/60/1176	7.6/82/913	8.5/38/809	8.0/45/680	13/44/758
Sober-t16	128	128	60/1714/1534	61/1808/1583	55/1700/1630	46/1287/1113	46/1299/1078	56/1908/1739	52/1502/1325	46/1309/1202	51/1411/1236	46/1281/1127
with IV	256	128	55/2343/1530	61/2425/1583	55/2497/1619	46/1760/1092	46/1665/1061	57/2736/1765	52/1891/1280	46/1684/1199	51/1831/1238	46/1668/1128
Sober-t32	128	128	27/1410/1302	26/1398/1200	27/1242/1271	28/1110/1126	28/1157/1166	28/1338/1270	27/1243/1258	21/1063/1196	20/1163/1190	18/925/1007
with IV	256	128	27/1696/1307	26/1677/1205	27/1446/1274	27/1330/1129	28/1382/1175	29/1566/1264	27/1422/1241	21/1280/1196	20/1315/1187	18/1106/1006
Scream-0	128	128	9.0/5453/2124	6.7/3684/1143	12/7700/2225	14/7196/2544	14/7206/2531	12/7736/2276	12/7150/2379	10/5295/1651	6.8/4403/1281	7.1/4205/1330
Scream-F	128	128	12/5391/2124	7.6/3684/1143	12/7700/2221	16/7300/2541	16/7260/2531	13/7735/2266	13/7139/2381	11/5295/1649	8.6/4403/1259	9.6/4204/1330
Scream-S	128	128	10/74K/2150	7.1/51K/1231	12/70K/2434	14/47K/2792	14/48K/2809	12/73K/2553	12/57K/2332	10/49K/1855	7.2/54K/1383	7.2/56K/1414
Seal-3.0	160	32	6.8/220K/22	6.0/178K/18	5.9/376K/18	6.9/335K/16	7.0/339K/15	13/374K/18	5.5/337K/16	5.9/300K/12	5.8/159K/16	5.9/141K/17

The figures in the entries are of key stream generation/keysetup times, where the key stream generation time are measured in cycles/byte and the key setup time is measured in cycles/keysetup.
For stream ciphers with initial value setup (IV) the figures in the entries are of key stream generation/keysetup/IVsetup times, where the IV setup time measured in cycles/IVsetup.

Table 46: Stream Ciphers Performance Results by Processor and Compiler

Primitive	Key	IV	Machine									
			Alpha	**Alpha**	**Sun/Sparc V9 450MHz**	**Sun/Sparc 248MHz**	**Sun/Sparc V9 338MHz**	**Sun/Sparc V9 338MHz**	**Sun/Sparc V9 400MHz**	**Sun/Sparc V9 400MHz**	**Sun 450MHz**	**Sun 333MHz**
Name	**Size**	**Size**	**cc**	**gcc 2.97**	**gcc 3.0.4+3.2.1**	**gcc 2.95.3**	**cc**	**gcc 3.0.4**	**cc**	**gcc 3.0.4**	**cc**	**gcc 3.2.1**
BMGL (m=16)	128		1126/1345	1382/1653	1693/2713	1861/2485	1643/2648	1731/2812	1583/2557	1665/2717	1739/2913	1691/2694
2nd impl	128		384/126K	429/122K	641/224K	676/235K	647/297K	664/244K	630/282K	640/236K	570/278K	699/230K
Snow	128		10/997	5.6/1242	7.2/1787	7.5/1826	8.2/1857	7.4/1951	7.7/1770	7.0/1838	6.7/1876	7.5/1823
	256		10/1027	5.6/1285	7.1/1801	7.5/1944	8.1/1905	7.6/2032	7.8/1791	7.1/1916	6.6/1882	7.4/1856
Sober-t16	128		79/1551	38/979	42/1538	47/1599	43/1565	44/1543	41/1495	42/1503	47/1656	45/1404
	256		79/2033	38/1307	43/2046	47/2090	42/2089	44/2067	41/2022	42/2004	46/2200	46/1894
Sober-t32	128		34/795	18/759	24/1501	25/1594	23/1303	24/1560	22/1260	24/1512	25/1355	24/1539
	256		34/957	18/948	24/1787	25/1898	23/1572	25/1859	22/1508	24/1795	25/1625	24/1794
Leviathan	128		12/25K	13/71K	15/135K	15/130K	16/165K	15/140K	16/156K	14/135K	15/156K	15/134K
	192		12/75K	13/83K	14/134K	15/129K	17/165K	15/140K	16/158K	15/134K	16/156K	15/133K
	256		12/25K	13/70K	14/133K	15/129K	17/163K	15/139K	16/158K	14/134K	16/156K	15/137K
Lili	128		581/47	566/34	757/71	672/70	691/73	798/73	665/71	760/71	757/75	720/71
RC4	128		12/3010	12/3243	14/3045	14/3070	13/3359	14/2714	13/3212	14/2667	13/3342	14/3539
BMGL with IV	128	128	1124/1328/44	1382/1687/28	1688/2770/160	1846/2600/158	1571/3092/192	1744/2861/186	1582/2979/170	1687/2752/169	1729/2938/196	1668/2691/174
2nd impl	128	128	382/126K/44	439/122K/29	679/225K/152	662/233K/160	657/304K/192	679/250K/185	623/275K/180	639/220K/162	579/280K/185	704/239K/182
Snow	128	64	10/28/520	5.6/14/579	7.2/52/918	7.6/56/903	8.0/55/940	7.4/56/995	7.6/52/880	7.0/52/947	6.7/51/900	7.2/51/963
with IV	256	64	10/30/541	5.6/18/610	7.2/74/890	7.6/80/973	8.2/78/954	7.4/86/1001	7.7/72/889	7.0/79/935	6.6/77/899	7.3/74/974
Sober-t16	128	128	79/1603/1489	38/1031/911	43/1672/1434	47/1746/1531	42/1703/1456	45/1714/1496	41/1643/1415	42/1657/1438	42/1798/1580	45/1556/1326
with IV	256	128	79/2059/1498	38/1371/907	43/2185/1459	47/2264/1505	42/2240/1464	45/2251/1485	40/2171/1410	42/2161/1434	47/2332/1548	45/1985/1350
Sober-t32	128	128	34/856/944	18/850/843	23/1609/1528	25/1741/1617	23/1449/1445	24/1673/1551	23/1389/1400	25/1596/1518	24/1482/1488	25/1734/1502
with IV	256	128	34/1034/948	19/1006/856	23/1899/1508	25/2037/1619	24/1712/1480	24/1965/1560	23/1651/1423	24/1901/1508	24/1758/1500	24/1973/1520
Scream-0	128	128	9.5/5780/1597	7.5/2944/991	10/5958/1606	10/6038/1672	11/6422/1725	10/6415/1704	11/6249/1660	10/6106/1634	11/7417/1932	13/6639/2299
Scream-F	128	128	10/5786/1586	12/2944/992	12/5995/1608	12/6045/1699	13/6514/1834	12/6396/1680	13/6250/1752	12/6104/1627	13/7470/1710	15/6693/2301
Scream-S	128	128	9.6/75K/1481	7.5/47K/995	9.9/44K/1712	10/42K/1678	12/30K/2007	10/48K/1752	11/28K/1892	10/46K/1691	13/29K/2166	13/41K/2255
Seal-3.0	160	32	10/137K/11	4.7/153K/10	3.8/206K/8.7	4.5/215K/10	5.2/199K/13	4.0/214K/9.6	5.0/193K/12	3.8/206K/8.8	5.2/190K/13	3.9/209K/9.4

The figures in the entries are of key stream generation/keysetup times, where the key stream generation time are measured in cycles/byte and the key setup time is measured in cycles/keysetup.
For stream ciphers with initial value setup (IV) the figures in the entries are of key stream generation/keysetup/IVsetup times, where the IV setup time measured in cycles/IVsetup.

Table 47: Hash Functions Performance Results by Processor and Compiler

Primitive Name	Hash Size	Machine									
		PIII/Linux Coppermine gcc 2.95.2	**PIII/Linux Coppermine gcc 3.1.1**	**PIII/Linux Coppermine gcc 3.0.3**	**PIII/Linux Tualatin gcc 2.96**	**PIII/Linux Katmai gcc 3.1.1**	**PIII/Linux Katmai gcc 3.3**	**PIII/Linux Coppermine gcc-egcs**	**PIII/Win Visual C 6.0**	**PIII/Win Intel C**	**PIII/Win gcc 2.95.3**
Whirlpool	512	83/64/5859	79/60/5557	79/59/5534	82/82/5771	89/62/6098	89/60/6122	86/65/6109	82/54/5657	90/47/6323	84/65/5946
2nd impl (MMX)	512	72/63/4869	46/64/3199	72/65/4884	78/83/5273	54/63/3762	54/63/3764	77/70/5201	130/83/8590	95/50/6298	73/71/5026
MD4	128	4.7/16/481	4.8/15/442	4.8/15/449	4.7/18/493	4.8/15/440	4.7/15/447	4.7/15/441	4.8/14/467	4.5/12/411	4.7/15/444
MD5	128	7.2/16/645	7.3/15/611	7.5/15/602	7.5/18/665	7.3/15/608	7.5/15/626	7.3/15/610	7.2/14/612	6.8/12/558	7.3/15/611
RIPEMD	160	23/16/1716	18/16/1345	22/16/1568	24/19/1748	18/16/1339	18/16/1347	22/16/1580	23/16/1675	16/14/1207	24/17/1704
SHA-0	160	20/16/1148	15/16/1025	15/16/1011	16/19/1097	15/16/1020	16/16/1024	16/16/1121	19/16/1100	12/14/896	16/17/1003
SHA-1	160	15/16/1195	15/16/1029	16/16/1053	16/19/1149	15/16/1024	15/16/1049	15/16/1187	19/16/1152	13/14/929	16/17/1083
SHA-2	256	40/45/2829	40/44/2755	40/47/2803	44/49/3068	40/44/2747	39/44/2736	41/45/2891	56/36/3816	39/20/2643	40/45/2850
	384	83/168/11K	87/157/11K	87/158/11K	89/195/12K	86/252/11K	104/254/13K	110/157/14K	80/144/10K	74/101/9907	84/162/11K
	512	83/168/11K	87/156/11K	87/158/11K	89/196/12K	86/252/11K	104/253/13K	110/157/14K	80/144/10K	74/101/9940	84/162/11K
Tiger	192	24/18/1722	22/16/1548	21/16/1542	21/19/1624	21/16/1547	21/16/1589	24/16/1761	25/15/1822	24/14/1745	24/18/1744
BCHASH-Rijndael	128	63/15/2188	49/15/1712	50/15/1721	59/18/2072	49/15/1723	50/15/1757	64/15/2182	45/14/1588	50/15/1686	58/15/2029
	256	143/45/4778	112/45/3775	146/47/4854	164/49/5514	112/45/3882	112/44/3878	155/45/5231	116/35/3913	128/20/4159	143/45/4785

		PII/Win Visual C 6.0	**PII/Win gcc 2.95.3**	**PIV/Linux Northwood gcc 2.95.2**	**PIV/Linux Northwood gcc 3.1.1**	**Xeon Model 1 gcc 3.2**	**Xeon Model 0 egcs 2.91.66**	**Xeon Model 0 gcc 2.96**	**AMD Duron Win XP VC 7.0 (.NET)**	**AMD Duron Win XP gcc 2.95.3**	**AMD Athlon Linux gcc 2.96**
Whirlpool	512	115/51/7870	97/68/6804	113/130/8905	108/132/8559	112/125/8794	123/127/9665	114/127/8851	104/42/7012	106/51/7328	109/49/7341
2nd impl (MMX)	512	146/83/9666	83/76/5992	97/132/7350	60/132/4495	60/125/4468	103/132/7652	99/129/7380	127/73/8335	102/54/6707	99/51/6573
MD4	128	4.8/13/462	4.7/15/444	6.4/34/639	6.5/34/600	6.4/34/598	6.4/29/611	6.5/33/586	4.8/8.3/419	4.7/10/443	4.8/11/423
MD5	128	7.2/12/611	7.3/15/613	9.4/34/828	9.5/34/799	9.4/34/794	9.4/29/786	9.5/33/785	7.5/11/599	8.0/10/653	8.0/11/631
RIPEMD	160	23/15/1651	24/17/1706	46/36/3203	26/36/1929	26/36/1921	35/36/2507	40/36/2866	21/12/1493	22/13/1628	23/12/1623
SHA-0	160	19/15/1089	16/17/1008	30/36/1828	23/36/1651	23/36/1619	27/36/1813	25/36/1438	12/13/1383	13/13/870	13/12/796
SHA-1	160	19/15/1136	16/17/1086	30/20/1903	25/20/1497	25/20/1487	26/30/2125	26/20/1460	14/13/1361	13/13/910	12/12/825
SHA-2	256	56/34/3799	40/46/2860	51/118/3817	40/120/3159	39/112/3042	53/114/4105	48/116/3510	46/39/3135	34/42/2425	34/41/2369
	384	118/244/15K	84/247/11K	122/294/16K	130/292/17K	135/280/18K	144/285/19K	131/282/17K	80/105/10K	71/112/9672	74/106/10K
	512	118/244/16K	84/247/11K	122/294/16K	130/292/17K	135/280/18K	145/284/19K	131/282/17K	88/106/17K	71/112/9752	74/106/10K
Tiger	192	27/13/1959	24/18/1778	27/27/2046	28/27/2015	29/27/2025	34/27/2449	26/26/1895	24/11/1912	21/14/1608	20/11/1449
BCHASH-Rijndael	128	45/13/1605	59/16/2085	131/19/4385	77/19/2579	76/20/2673	99/19/3445	87/19/2868	47/11/1650	70/14/2384	60/13/2008
	256	123/35/4118	146/45/4914	260/118/8815	152/120/5254	150/109/5288	276/114/9268	204/116/7000	149/40/4889	163/43/5432	156/41/5151

		Alpha cc	**Alpha gcc 2.97**	**Sun/Sparc V9 450MHz gcc 3.0.4+3.2.1**	**Sun/Sparc 248MHz gcc 2.95.3**	**Sun/Sparc V9 338MHz cc**	**Sun/Sparc V9 338MHz gcc 3.0.4**	**Sun/Sparc V9 400MHz cc**	**Sun/Sparc V9 400MHz gcc 3.0.4**	**Sun 450MHz cc**	**Sun 333MHz gcc 3.2.1**
Whirlpool	512	36/14/2485	37/15/2567	171/22/11K	184/31/12K	114/87/7281	183/32/12K	109/82/7000	173/30/11K	124/89/8371	176/28/11K
2nd impl (MMX)	512	84/15/5476	35/15/2400	143/24/9450	146/33/9711	102/88/6552	149/33/9842	97/82/6266	142/32/9551	91/91/6541	149/32/9808
MD4	128	7.1/12/585	6.3/11/467	7.1/11/526	7.6/11/529	7.1/10/526	7.3/11/542	6.8/10/515	7.1/11/528	7.2/16/531	7.1/11/536
MD5	128	10/12/812	10.0/11/691	10/11/795	10/11/754	10/10/767	10/11/787	10/10/746	10/11/750	10/17/782	10/11/763
RIPEMD	160	25/12/1817	24/11/1697	24/15/1751	25/15/1776	27/12/1868	25/13/1798	26/12/1811	24/13/1758	27/17/1839	24/13/1797
SHA-0	160	11/12/859	9.8/11/653	13/16/952	14/15/915	12/12/886	13/13/979	12/12/859	13/13/952	13/17/853	15/13/1015
SHA-1	160	11/10/883	11/11/745	13/13/1061	15/15/1211	14/12/1021	13/13/1119	14/12/955	13/13/1096	16/14/1095	13/13/1103
SHA-2	256	31/10/2117	29/19/2021	36/46/2411	37/49/2520	35/149/2834	37/47/2492	34/137/2725	36/45/2409	34/153/2745	37/34/2467
	384	17/30/2357	16/46/2333	100/118/13K	98/135/12K	72/283/9842	102/219/13K	69/268/9541	100/213/13K	65/287/9018	107/212/14K
	512	17/30/2374	16/46/2348	100/117/13K	98/137/12K	71/284/9863	102/220/13K	69/273/9560	100/213/13K	65/288/9027	160/324/21K
Tiger	192	5.4/10/407	5.2/10/390	28/17/1682	23/17/1551	28/14/1787	26/16/1696	26/14/1691	25/16/1643	32/18/1990	46/21/2871
BCHASH-Rijndael	128	33/8.1/1179	49/13/1676	65/82/2289	73/84/2596	63/98/2063	78/100/2701	61/83/1997	66/88/2383	54/100/2007	65/94/2340
	256	97/10/3204	–/–/–	221/133/7417	224/133/7688	182/150/6011	229/150/7635	171/134/5583	223/133/7456	187/152/6248	237/149/7935

The figures in the entries are of hash/initialize/initialize+finalize times, where the hash time is measured in cycles/byte and the initialize and initialize+finalize times are measured in cycles.

Table 48: Message Authentication Codes Performance Results by Processor and Compiler

Primitive Name	Key Size	MAC Size	Machine									
			PIII/Linux Coppermine gcc 2.95.2	PIII/Linux Coppermine gcc 3.1.1	PIII/Linux Coppermine gcc 3.0.3	PIII/Linux Tualatin gcc 2.96	PIII/Linux Katmai gcc 3.1.1	PIII/Linux Katmai gcc 3.3	PIII/Linux Coppermine gcc-egcs	PIII/Win Visual C 6.0	PIII/Win Intel C	PIII/Win gcc 2.95.3
Ttmac	160	160	21/30/1564	22/29/1577	23/29/1660	26/40/1842	22/27/1573	22/30/1585	21/32/1599	29/34/1976	19/21/1304	21/30/1576
Umac-16	128	64	6.2/52K/53K	6.0/56K/57K	6.2/55K/56K	6.4/53K/55K	–/–/–	6.0/57K/58K	6.6/57K/58K	5.8/71K/72K	6.2/52K/53K	6.1/52K/53K
Umac-32	128	64	3.1/55K/56K	3.0/59K/60K	3.1/59K/60K	3.2/54K/55K	–/–/–	2.9/59K/60K	3.3/60K/61K	2.7/64K/69K	3.2/53K/55K	3.2/54K/55K
HMAC-Whirlpool	512	512	72/5078/24K	74/5195/24K	72/5129/24K	78/6544/27K	83/5935/28K	86/6076/29K	77/5381/25K	81/5875/27K	90/6407/30K	73/5163/24K
HMAC-MD4	512	128	4.7/638/1991	4.8/663/1976	4.7/676/2020	4.8/1753/3175	4.7/693/2024	4.7/684/2028	4.7/642/1961	4.8/867/2247	4.5/640/1881	4.7/656/1985
HMAC-MD5	512	128	7.3/804/2677	7.3/838/2653	7.5/853/2699	7.5/1940/3873	7.3/866/2685	7.5/868/2750	7.3/808/2634	7.2/1018/2833	6.8/799/2470	7.3/828/2655
HMAC-RIPE-MD	512	160	23/1863/6910	18/1571/5608	22/1781/6451	24/2992/8141	18/1594/5623	18/1593/5646	22/1782/6526	23/2069/7100	16/1440/5045	24/1907/6987
HMAC-SHA-0	512	160	20/1695/5925	16/1380/4697	15/1398/4619	16/2530/5960	15/1413/4936	16/1428/4971	16/1393/5159	19/1788/5394	12/1184/3905	16/1414/4828
HMAC-SHA-1	512	160	15/1390/5237	15/1346/4697	16/1422/4786	17/2550/6145	15/1367/4825	15/1362/4825	15/1376/5200	19/1829/5597	13/1211/4029	16/1437/5207
HMAC-SHA-2	512	256	42/3087/12K	40/2958/11K	40/3018/11K	44/4304/13K	43/3378/12K	39/2899/11K	41/3106/11K	56/4188/15K	40/3020/11K	40/2967/11K
	512	384	84/626/34K	87/638/35K	87/641/35K	89/1706/37K	87/702/35K	104/694/42K	111/625/44K	80/745/32K	75/730/30K	84/599/34K
	512	512	84/615/34K	88/629/35K	87/638/35K	89/1699/37K	87/700/35K	104/694/42K	111/614/44K	80/744/32K	75/727/30K	84/584/34K
HMAC-Tiger	512	192	24/1867/7116	21/1766/6655	21/1758/6729	21/2881/7700	21/1787/6664	21/1789/6526	25/1965/7251	25/2214/7803	25/1974/7210	24/1924/7117
CBCMAC-Rijndael	128	128	29/1024/3151	28/616/2084	26/623/2056	28/880/2654	28/744/2308	28/752/2347	29/1082/3174	24/548/1919	33/595/2425	29/1007/2946
CBCMAC-DES	64	64	61/1000/3012	62/973/2924	61/973/2956	64/1131/3330	61/981/2931	62/1009/2998	62/1047/3099	74/980/3178	70/988/3099	62/1005/3017
CBCMAC-Shacal	512	160	31/829/2886	36/840/3043	32/856/2936	34/1090/3531	35/1027/3150	36/1017/3060	33/957/3143	46/1097/3978	28/856/2745	31/846/2916

			PII/Win Visual C 6.0	PII/Win gcc 2.95.3	PIV/Linux Northwood gcc 2.95.2	PIV/Linux Northwood gcc 3.1.1	Xeon Model 1 gcc 3.2	Xeon Model 0 egcs 2.91.66	Xeon Model 0 gcc 2.96	AMD Duron Win XP VC 7.0 (.NET)	AMD Duron Win XP gcc 2.95.3	AMD Athlon Linux gcc 2.96
Ttmac	160	160	28/32/1959	21/30/1582	41/57/2894	40/50/2675	39/47/2671	47/55/3254	37/53/2614	25/14/1754	21/21/1526	22/20/1596
Umac-16	128	64	6.4/73K/69K	6.1/53K/54K	6.4/76K/77K	6.2/79K/81K	6.1/76K/78K	6.5/80K/82K	7.2/85K/88K	7.7/74K/75K	6.2/76K/77K	6.9/74K/75K
Umac-32	128	64	2.5/69K/70K	3.1/54K/55K	6.7/76K/77K	6.7/79K/81K	6.6/78K/81K	7.0/81K/82K	7.4/86K/89K	2.2/70K/72K	1.9/77K/78K	1.9/75K/76K
HMAC-Whirlpool	512	512	116/8120/39K	86/6351/29K	103/7466/34K	98/7161/33K	106/7592/36K	103/7563/35K	104/7654/35K	104/7232/34K	102/6908/33K	100/6810/32K
HMAC-MD4	512	128	4.8/885/2295	4.7/682/2013	6.4/807/2529	6.5/770/2396	6.4/786/2385	6.4/836/2490	6.5/772/2397	4.8/821/2056	4.7/637/1944	4.7/609/1880
HMAC-MD5	512	128	7.2/1032/2889	7.4/853/2705	9.4/998/3305	9.5/963/3186	9.4/987/3175	9.5/1027/3288	9.5/968/3187	7.4/985/2736	8.0/848/2782	8.0/817/2709
HMAC-RIPE-MD	512	160	23/2054/7060	24/1952/7071	47/3385/12K	27/2083/7656	26/2095/7682	35/2753/10K	40/3036/11K	21/1928/6556	22/1812/6673	23/1793/6657
HMAC-SHA-0	512	160	19/1798/5442	16/1454/4955	30/2413/8320	23/1915/7065	23/1922/7099	27/2230/7969	25/2041/6723	16/2206/6517	13/1254/4033	13/1196/3830
HMAC-SHA-1	512	160	20/1849/5661	16/1445/5255	30/2497/8701	25/1987/6702	24/1989/6722	26/2157/8421	26/2106/6883	14/1687/5695	13/1204/4141	12/1162/3947
HMAC-SHA-2	512	256	56/4172/15K	40/3017/11K	51/3927/15K	40/3129/12K	39/3186/11K	53/4132/16K	48/3634/14K	47/3558/13K	33/2479/9469	34/2564/9602
	512	384	118/815/47K	84/660/34K	124/1041/50K	131/996/53K	136/829/55K	145/842/59K	132/806/53K	80/688/32K	72/512/29K	74/488/30K
	512	512	118/804/48K	84/654/34K	124/1041/51K	131/993/53K	136/827/55K	145/820/59K	132/801/53K	80/689/32K	72/504/29K	74/487/30K
HMAC-Tiger	512	192	27/2332/8184	24/1964/7182	28/2307/8489	29/2325/8515	28/2336/8540	34/2750/10K	26/2145/7836	24/2062/7188	22/1739/6523	20/1654/6037
CBCMAC-Rijndael	128	128	24/694/2102	29/1078/1661	40/1483/2789	26/958/3318	27/960/3407	35/1720/5347	27/1489/4432	33/513/1124	32/861/1512	31/819/2727
CBCMAC-DES	64	64	74/971/3175	62/973/1560	82/1009/1932	72/1061/3218	72/1036/3198	82/1175/3652	69/1041/3238	61/747/1301	54/766/1283	56/831/2581
CBCMAC-Shacal	512	160	46/1235/4065	31/877/1577	67/751/2473	86/864/5068	80/854/5037	90/1107/5932	74/828/4725	38/594/1438	31/535/1237	29/605/2372

			Alpha cc	Alpha gcc 2.97	Sun/Sparc V9 450MHz gcc 3.0.4+3.2.1	Sun/Sparc 248MHz gcc 2.95.3	Sun/Sparc V9 338MHz cc	Sun/Sparc V9 338MHz gcc 3.0.4	Sun/Sparc V9 400MHz cc	Sun/Sparc V9 400MHz gcc 3.0.4	Sun 450MHz cc	Sun 333MHz gcc 3.2.1
Ttmac	160	160	25/16/1691	28/35/1952	26/57/1796	27/52/1790	30/57/2067	27/58/1851	29/53/1993	26/57/1769	28/47/1903	26/61/1849
Umac-16	128	64	7.8/105K/107K	4.5/48K/49K	28/76K/79K	20/74K/76K	51/79K/83K	22/85K/89K	45/72K/74K	21/78K/81K	46/82K/86K	29/78K/80K
Umac-32	128	64	1.2/104K/106K	1.1/46K/47K	10/75K/78K	10/74K/76K	–/–/–	11/83K/85K	–/–/–	10/78K/80K	–/–/–	10/79K/80K
HMAC-Whirlpool	512	512	37/2488/13K	36/2710/12K	145/9890/48K	147/9982/49K	104/7148/34K	154/10K/51K	97/6698/31K	146/10K/48K	98/7279/34K	150/10K/50K
HMAC-MD4	512	128	7.1/603/2332	6.3/759/2186	7.1/934/2678	7.7/953/2657	7.0/972/2615	7.3/974/2763	6.9/936/2531	7.1/947/2714	7.3/979/2654	7.1/968/2768
HMAC-MD5	512	128	10/845/3258	10/990/3107	10/1197/3564	10/1159/3543	10/1189/3541	10/1247/3725	10/1150/3440	10/1215/3636	10/1195/3574	10/1224/3674
HMAC-RIPE-MD	512	160	25/1793/7189	24/1950/7028	24/2102/11K	25/2114/11K	27/2284/10K	25/2162/12K	26/2200/10K	24/2103/11K	27/2316/11K	24/2118/14K
HMAC-SHA-0	512	160	11/908/3581	9.8/981/3044	13/1439/4830	14/1458/4468	12/1297/3986	14/1480/5033	12/1267/3926	13/1432/4870	13/1280/4054	15/1601/5117
HMAC-SHA-1	512	160	11/934/3670	11/1073/3402	13/1448/4798	15/1500/5283	14/1408/4698	13/1495/4979	14/1361/4558	13/1453/4829	16/1443/5260	13/1482/4895
HMAC-SHA-2	512	256	31/2162/8462	29/2292/8364	36/2919/10K	37/2951/10K	35/3029/10K	37/3005/10K	34/2943/10K	36/2947/10K	34/2938/10K	37/2952/10K
	512	384	17/193/7280	16/412/7261	100/695/40K	98/686/39K	71/896/31K	103/804/41K	70/863/30K	100/785/40K	65/893/28K	105/788/42K
	512	512	17/195/7285	16/408/7268	100/693/40K	98/681/39K	71/880/31K	103/788/41K	69/835/30K	101/770/40K	65/873/28K	105/765/42K
HMAC-Tiger	512	192	5.4/500/1776	5.2/682/1933	28/2381/8132	24/2130/7050	28/2407/7963	26/2324/7628	27/2323/7749	25/2270/7407	33/2659/9073	31/2624/9143
CBCMAC-Rijndael	128	128	20/523/1636	21/812/2321	34/691/3592	34/839/3606	23/731/3151	36/699/3748	22/671/2979	32/676/3596	24/649/2958	33/727/3881
CBCMAC-DES	64	64	39/628/2154	46/638/2153	65/924/2991	66/969/3098	59/940/2843	70/971/3164	56/912/2723	68/934/3038	61/902/2946	69/937/3081
CBCMAC-Shacal	512	160	42/984/3575	37/1278/3953	36/1017/3629	36/1017/3588	40/1259/4027	37/1056/3837	39/1229/3904	36/1035/3633	34/1284/3762	37/1074/3851

The figures in the entries are of MAC/keysetup/keysetup+finalize times, where the MAC time is measured in cycles/byte and the keysetup and keysetup+finalize times are measured in cycles.

Table 49: Asymmetric Encryption Performance Results by Processor and Compiler

Primitive Name	Key Size	Machine									
		PIII/Linux Coppermine gcc 2.95.2	**PIII/Linux Coppermine gcc 3.1.1**	**PIII/Linux Coppermine gcc 3.0.3**	**PIII/Linux Tualatin gcc 2.96**	**PIII/Linux Katmai gcc 3.1.1**	**PIII/Linux Katmai gcc 3.3**	**PIII/Linux Coppermine gcc-egcs**	**PIII/Win Visual C 6.0**	**PIII/Win Intel C**	**PIII/Win gcc 2.95.3**
EPOC-2	1152	31M/8224K/553M	31M/8262K/532M	31M/8255K/532M	31M/8076K/577M	31M/8246K/366M	31M/8306K/534M	30M/7989K/457M	–/–/–	–/–/–	30M/8024K/569M
RSA-OAEP	1024	2642K/57M/2415M	2058K/42M/1806M	2336K/48M/1654M	2964K/60M/2522M	2026K/42M/1736M	2087K/43M/1828M	2768K/57M/2460M	2588K/52M/2214M	2621K/56M/2272M	2583K/53M/2234M
ACE encrypt		62M/26M/3306M	63M/26M/3321M	60M/25M/3195M	57M/25M/3194M	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–
EPOC1		6803K/11M/48M	6914K/11M/49M	6547K/11M/47M	6312K/11M/46M	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	6433K/11M/47M
EPOC2		8782K/12M/48M	8815K/12M/48M	8779K/12M/47M	8254K/11M/46M	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	8477K/12M/48M
EPOC3		8763K/5230K/48M	8936K/5403K/49M	8815K/5250K/48M	8279K/4882K/46M	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	8554K/5158K/48M
PSEC1		8876K/7240K/2747	8321K/6839K/2472	8387K/6693K/2306	7952K/6322K/2422	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	9703K/7341K/–
PSEC2		8654K/6897K/2724	8174K/6557K/2480	8194K/6417K/2380	7703K/6176K/2424	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–
PSEC3		8838K/81K/2686	8291K/78K/2534	8339K/79K/2370	7952K/76K/2483	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–

		PII/Win Visual C 6.0	**PII/Win gcc 2.95.3**	**PIV/Linux Northwood gcc 2.95.2**	**PIV/Linux Northwood gcc 3.1.1**	**Xeon Model 1 gcc 3.2**	**Xeon Model 0 egcs 2.91.66**	**Xeon Model 0 gcc 2.96**	**AMD Duron Win XP VC 7.0 (.NET)**	**AMD Duron Win XP gcc 2.95.3**	**AMD Athlon Linux gcc 2.96**
EPOC-2	1152	–/–/–	25M/6235K/389M	84M/18M/1152M	83M/18M/1150M	23M/5866K/357M	83M/18M/1141M	83M/18M/1211M	–/–/–	18M/4654K/299M	22M/6066K/402M
RSA-OAEP	1024	2576K/53M/2003M	2572K/53M/1935M	3521K/75M/3141M	3028K/62M/2616M	2981K/61M/2587M	3835K/79M/3336M	3246K/67M/2792M	2468K/49M/2073M	2383K/48M/2024M	2289K/48M/2027M
ACE encrypt		–/–/–	–/–/–	63M/27M/3426M	64M/27M/3416M	61M/26M/3237M	51M/20M/3402M	62M/26M/3314M	–/–/–	–/–/–	33M/14M/1748M
EPOC1		–/–/–	7599K/9853K/59M	18M/30M/124M	19M/30M/123M	19M/30M/122M	19M/30M/123M	18M/30M/122M	–/–/–	–/–/–	4155K/6930K/31M
EPOC2		–/–/–	8538K/10M/47M	24M/33M/123M	24M/32M/123M	23M/32M/122M	24M/33M/124M	24M/32M/122M	–/–/–	–/–/–	5823K/7270K/31M
EPOC3		–/–/–	7981K/3895K/43M	24M/11M/123M	23M/11M/123M	23M/11M/122M	24M/11M/123M	23M/11M/121M	–/–/–	–/–/–	5808K/2728K/31M
PSEC1		–/–/–	19M/8373K/5295	14M/11M/5729	13M/10M/4776	12M/10M/4696	14M/11M/5021	13M/10M/4842	–/–/–	–/–/–	7480K/6087K/2993
PSEC2		–/–/–	10M/8197K/4534	14M/10M/5727	12M/10M/4710	12M/10M/4780	14M/10M/4840	13M/9959K/4538	–/–/–	–/–/–	7301K/5757K/2992
PSEC3		–/–/–	10M/72K/4546	14M/128K/5784	13M/108K/4716	13M/111K/4561	14M/120K/7003	13M/112K/4471	–/–/–	–/–/–	7495K/76K/2918

		Alpha cc	**Alpha gcc 2.97**	**Sun/Sparc V9 450MHz gcc 3.0.4+3.2.1**	**Sun/Sparc 248MHz gcc 2.95.3**	**Sun/Sparc V9 338MHz cc**	**Sun/Sparc V9 338MHz gcc 3.0.4**	**Sun/Sparc V9 400MHz cc**	**Sun/Sparc V9 400MHz gcc 3.0.4**	**Sun 450MHz cc**	**Sun 333MHz gcc 3.2.1**
EPOC-2	1152	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–
RSA-OAEP	1024	–/–/–	–/–/–	3892K/77M/3224M	3927K/77M/3225M	7131K/138M/5634M	4005K/80M/3281M	6853K/133M/5431M	3872K/77M/3106M	6198K/120M/5012M	3932K/78M/3280M
ACE encrypt		–/–/–	–/–/–	272M/114M/14G	–/–/–	304M/128M/16G	277M/117M/14G	283M/119M/14G	268M/113M/13G	–/–/–	275M/115M/14G
EPOC1		1584K/2834K/13M	1343K/2376K/10M	–/–/–	–/–/–	70M/114M/455M	–/–/–	66M/106M/421M	–/–/–	–/–/–	–/–/–
EPOC2		2467K/3001K/13M	2112K/2461K/10M	–/–/–	–/–/–	81M/123M/455M	–/–/–	75M/114M/418M	–/–/–	–/–/–	–/–/–
EPOC3		2456K/1282K/12M	2089K/1067K/10M	–/–/–	–/–/–	79M/43M/453M	–/–/–	73M/40M/418M	–/–/–	–/–/–	–/–/–
PSEC1		8303K/7438K/4116	7136K/6280K/3612	–/–/–	–/–/–	25M/23M/5756	–/–/–	22M/20M/5017	–/–/–	–/–/–	–/–/–
PSEC2		8238K/7383K/3857	7040K/6197K/3742	–/–/–	–/–/–	24M/22M/5746	–/–/–	22M/20M/5061	–/–/–	–/–/–	–/–/–
PSEC3		8348K/40K/4094	7147K/49K/3838	–/–/–	–/–/–	25M/86K/5537	–/–/–	22M/73K/4980	–/–/–	–/–/–	–/–/–

The figures in the entries are of encrypt/decrypt/key generation times. All of them are measured in cycles/invocation.

Table 50: Signature Performance Results by Processor and Compiler

Primitive Name	Key Size	Machine									
		PIII/Linux Coppermine gcc 2.95.2	**PIII/Linux Coppermine gcc 3.1.1**	**PIII/Linux Coppermine gcc 3.0.3**	**PIII/Linux Tualatin gcc 2.96**	**PIII/Linux Katmai gcc 3.1.1**	**PIII/Linux Katmai gcc 3.3**	**PIII/Linux Coppermine gcc-egcs**	**PIII/Win Visual C 6.0**	**PIII/Win Intel C**	**PIII/Win gcc 2.95.3**
ECDSA-GF(2^p)	160	5326K/6808K/5215K	4968K/6387K/4860K	4775K/6085K/4669K	6344K/8047K/6243K	4871K/6359K/4779K	5277K/6810K/5181K	5947K/7575K/5832K	4639K/5950K/4494K	4689K/6005K/4662K	5413K/6984K/5330K
ECDSA-GF(2^{163})	163	5061K/6809K/4852K	5504K/7471K/5320K	5411K/7351K/5190K	6052K/8166K/5842K	5502K/7396K/5276K	5502K/7396K/5294K	5405K/7243K/5221K	5315K/7031K/5096K	5051K/6652K/4938K	5124K/6831K/4932K
KCDSA-GF(2^p)	160	5314K/6869K/5275K	4935K/6326K/4889K	4724K/6085K/4688K	6319K/8116K/6275K	4871K/6359K/4843K	5277K/6855K/5239K	5907K/7575K/5867K	4596K/5869K/4524K	4656K/5971K/4701K	5405K/6950K/5367K
KCDSA-GF(2^{163})	163	5024K/6692K/4888K	5462K/7291K/5342K	5387K/7291K/5248K	5953K/7902K/5774K	5457K/7261K/5304K	5502K/7216K/5331K	5362K/7176K/5252K	5281K/7005K/5127K	5034K/6716K/4977K	5090K/6831K/4962K
Esign	1152	671K/185K/458M	647K/178K/401M	648K/167K/368M	644K/246K/519M	682K/177K/452M	658K/221K/406M	586K/198K/497M	806K/209K/582M	705K/169K/568M	604K/187K/460M
RSA-PSS	1024	57M/2663K/1798M	43M/2060K/1343M	48M/2359K/1506M	60M/2961K/1874M	42M/2029K/1334M	43M/2090K/1354M	57M/2761K/1825M	52M/2581K/1443M	54M/2585K/1483M	53M/2562K/1982M
SFLASHv2		1555K/384K/1192M	1403K/348K/845M	–/–/–	1753K/328K/1060M	–/–/–	1576K/399K/916M	1385K/395K/731M	–/–/–	–/–/–	–/–/–
ACE sign		27M/19M/8201M	24M/19M/7313M	24M/19M/7617M	25M/20M/7436M	25M/19M/9800M	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–
ECDSA		1320K/3456K/1116K	952K/2540K/805K	1008K/2701K/857K	983K/2617K/824K	958K/2538K/807K	–/–/–	–/–/–	–/–/–	–/–/–	1211K/3207K/1023K
Esign (e=8)	1152	2340K/405K/112M	2104K/395K/107M	2285K/393K/105M	2046K/370K/104M	2112K/401K/106M	–/–/–	–/–/–	–/–/–	–/–/–	2099K/370K/113M
FLASH		2251K/505K/2335M	2050K/443K/1869M	2411K/529K/2181M	2110K/291K/1092M	2423K/665K/1960M	2429K/668K/1967M	2844K/296K/1211M	–/–/–	–/–/–	2721K/308K/1395M
QUARTZ		6881M/129K/1788M	6072M/121K/1622M	6690M/126K/1652M	5500M/122K/1555M	7271M/154K/1781M	6954M/152K/1769M	7426M/142K/1901M	–/–/–	–/–/–	6104M/115K/1826M
SFLASH		1924K/274K/1153M	1724K/195K/793M	2084K/216K/1056M	1700K/207K/889M	1743K/207K/834M	1747K/208K/816M	2365K/284K/786M	–/–/–	–/–/–	2303K/259K/952M
RSA-PSS		64M/1252K/2440M	57M/1090K/2188M	61M/1180K/2357M	60M/981K/2317M	57M/1098K/2183M	56M/1101K/2182M	69M/1366K/2629M	–/–/–	–/–/–	62M/1208K/2388M

		PII/Win Visual C 6.0	**PII/Win gcc 2.95.3**	**PIV/Linux Northwood gcc 2.95.2**	**PIV/Linux Northwood gcc 3.1.1**	**Xeon Model 1 gcc 3.2**	**Xeon Model 0 egcs 2.91.66**	**Xeon Model 0 gcc 2.96**	**AMD Duron Win XP VC 7.0 (.NET)**	**AMD Duron Win XP gcc 2.95.3**	**AMD Athlon Linux gcc 2.96**
ECDSA-GF(2^p)	160	4620K/5950K/4536K	5530K/7087K/5411K	6332K/8190K/6172K	6100K/7862K/5936K	6281K/8098K/6080K	7513K/9656K/7452K	6261K/8261K/6150K	4025K/5251K/3909K	4464K/5762K/4354K	4612K/5940K/4505K
ECDSA-GF(2^{163})	163	5390K/7105K/5316K	5162K/6912K/4970K	5922K/7981K/5669K	6218K/8379K/5936K	6161K/8217K/5840K	6437K/8673K/6181K	6538K/8732K/6227K	4578K/6153K/4363K	4602K/6159K/4395K	5272K/7106K/5027K
KCDSA-GF(2^p)	160	4410K/5740K/4582K	5477K/7052K/5444K	6359K/8262K/6245K	6024K/7769K/5984K	6139K/7993K/6129K	7412K/9538K/7533K	6219K/8093K/6188K	4013K/5149K/3933K	4440K/5720K/4406K	4557K/5984K/4533K
KCDSA-GF(2^{163})	163	5355K/7105K/5348K	5141K/6912K/5020K	5939K/7854K/5707K	6108K/8116K/5973K	6094K/8068K/5892K	6345K/8505K/6230K	6454K/8572K/6290K	4530K/6068K/4400K	4572K/6135K/4448K	5236K/6967K/5087K
Esign	1152	807K/208K/584M	525K/190K/400M	1049K/281K/579M	996K/253K/545M	527K/259K/483M	1166K/312K/694M	957K/259K/618M	767K/206K/504M	405K/148K/293M	519K/154K/401M
RSA-PSS	1024	53M/2584K/1438M	53M/2562K/1983M	74M/3524K/2337M	62M/3019K/1937M	62M/2979K/1929M	78M/3815K/2462M	67M/3274K/2103M	49M/2454K/1353M	48M/2351K/1793M	48M/2288K/1502M
SFLASHv2		1837K/363K/734M	1761K/344K/1237M	–/–/–	–/–/–	1299K/269K/802M	1455K/299K/737M	–/–/–	2030K/406K/1020M	2121K/387K/1498M	2214K/385K/1449M
ACE sign		–/–/–	–/–/–	30M/20M/10G	26M/20M/9645M	25M/19M/8790M	28M/19M/10G	26M/19M/9822M	–/–/–	–/–/–	14M/7113K/5485M
ECDSA		–/–/–	1236K/3251K/1042K	2077K/5780K/1833K	1971K/5415K/1758K	1463K/3870K/1241K	2117K/5923K/1895K	1753K/4807K/1565K	–/–/–	1205K/3263K/1030K	997K/2663K/841K
Esign (e=8)	1152	–/–/–	2074K/296K/95M	4726K/954K/269M	4434K/936K/270M	4107K/938K/265M	4846K/951K/268M	4487K/939K/266M	–/–/–	–/–/–	1818K/275K/74M
FLASH		–/–/–	7067K/–/2106M	2882K/288K/1584M	2406K/280K/1116M	2336K/244K/1239M	3000K/293K/1137M	2294K/295K/1129M	–/–/–	3758K/412K/1612M	3296K/443K/1456M
QUARTZ		–/–/–	12G/220K/4257M	7378M/152K/3248M	6261M/144K/3167M	7214M/136K/2823M	10G/161K/3553M	7856M/147K/3281M	–/–/–	12G/123K/2133M	6406M/128K/2028M
SFLASH		–/–/–	2330K/269K/943M	2824K/431K/1499M	2387K/395K/1192M	2336K/337K/1202M	2842K/388K/1395M	2308K/363K/1132M	–/–/–	3332K/310K/1165M	2910K/212K/1016M
RSA-PSS		–/–/–	115M/1255K/3412M	84M/1615K/3243M	82M/1587K/3206M	76M/1473K/2967M	84M/697K/3670M	83M/1487K/3183M	–/–/–	57M/1171K/2202M	53M/1381K/2078M

		Alpha cc	**Alpha gcc 2.97**	**Sun/Sparc V9 450MHz gcc 3.0.4+3.2.1**	**Sun/Sparc 248MHz gcc 2.95.3**	**Sun/Sparc V9 338MHz cc**	**Sun/Sparc V9 338MHz gcc 3.0.4**	**Sun/Sparc V9 400MHz cc**	**Sun/Sparc V9 400MHz gcc 3.0.4**	**Sun 450MHz cc**	**Sun 333MHz gcc 3.2.1**
ECDSA-GF(2^p)	160	2523K/3212K/2466K	3646K/4702K/3574K	7515K/9540K/7417K	6745K/8680K/6656K	7706K/9734K/7597K	7571K/9497K/7436K	7480K/9520K/7362K	7280K/9320K/7207K	8100K/9990K/7805K	7592K/9657K/7522K
ECDSA-GF(2^{163})	163	3590K/4791K/3465K	4629K/6225K/4446K	7020K/9450K/6790K	4811K/6522K/4640K	5002K/6760K/4806K	7334K/9768K/7068K	4920K/6560K/4690K	7040K/9400K/6848K	4815K/6435K/4631K	7459K/10M/7289K
KCDSA-GF(2^p)	160	2512K/3184K/2482K	3646K/4679K/3615K	7515K/9540K/7462K	6770K/8630K/6744K	7807K/10M/7830K	–/–/–	7600K/9640K/7584K	7360K/9400K/7348K	7920K/10M/7907K	–/–/–
KCDSA-GF(2^{163})	163	3562K/4674K/3470K	4585K/6041K/4484K	7335K/9675K/7192K	4786K/6324K/4673K	5577K/7300K/5249K	–/–/–	4920K/6600K/4814K	7080K/9360K/6937K	4680K/6300K/4617K	–/–/–
Esign	1152	416K/85K/174M	700K/156K/381M	1053K/322K/562M	1035K/300K/536M	1270K/359K/605M	1090K/338K/575M	1246K/347K/591M	1050K/325K/560M	1257K/368K/623M	1073K/328K/578M
RSA-PSS	1024	–/–/–	–/–/–	77M/3858K/2994M	77M/3943K/3049M	136M/7098K/5351M	80M/3988K/–	132M/6840K/5166M	77M/3520K/3036M	120M/6120K/4707M	79M/3962K/3095M
SFLASHv2		1353K/246K/773M	1628K/143K/540M	1885K/312K/972M	2026K/357K/958M	2332K/413K/1042M	2009K/344K/1053M	2100K/364K/928M	1926K/324K/1001M	2139K/383K/942M	2034K/373K/1019M
ACE sign		–/–/–	–/–/–	95M/85M/25G	–/–/–	110M/95M/32G	97M/87M/26G	112M/90M/31G	102M/84M/26G	–/–/–	96M/85M/26G
ECDSA		–/–/386K	–/–/533K	–/–/–	–/–/–	912K/2347K/749K	1423K/3934K/1272K	809K/2083K/670K	1372K/3785K/1232K	–/–/–	–/–/–
Esign (e=8)	1152	1044K/128K/41M	978K/113K/32M	–/–/–	–/–/–	5578K/3218K/945M	–/–/–	5069K/2965K/872M	–/–/–	–/–/–	–/–/–
FLASH		1506K/242K/769M	1531K/197K/732M	3134K/394K/1089M	–/–/–	3358K/436K/1140M	3236K/409K/1370M	3135K/410K/1058M	3132K/394K/1330M	–/–/–	3374K/443K/1188M
QUARTZ		2855M/180K/1111M	2930M/129K/1131M	13G/191K/5958M	–/–/–	6246M/237K/5132M	14G/236K/6086M	5760M/224K/4703M	14G/227K/5893M	–/–/–	15G/225K/5926M
SFLASH		1309K/206K/558M	1324K/164K/465M	2767K/574K/3753M	–/–/–	3207K/671K/4388M	2894K/640K/4190M	2885K/627K/4085M	2784K/626K/3960M	–/–/–	2824K/567K/3930M
RSA-PSS		40M/794K/1558M	40M/779K/1560M	132M/2591K/5075M	–/–/–	147M/2424K/5609M	145M/2682K/5618M	134M/2619K/4968M	142M/2757K/5438M	–/–/–	133M/2642K/5110M

The figures in the entries are of sign/verify/key generation times. All of them are measured in cycles/invocation.

Table 51: Identification Performance Results by Processor and Compiler

Primitive Name	Key Size	Machine									
		PIII/Linux Coppermine gcc 2.95.2	PIII/Linux Coppermine gcc 3.1.1	PIII/Linux Coppermine gcc 3.0.3	PIII/Linux Tualatin gcc 2.96	PIII/Linux Katmai gcc 3.1.1	PIII/Linux Katmai gcc 3.3	PIII/Linux Coppermine gcc-egcs	PIII/Win Visual C 6.0	PIII/Win Intel C	PIII/Win gcc 2.95.3
GPS		320M/3580K/5689K/328/6467K	322M/3597K/5730K/337/6495K	321M/3597K/5730K/337/6495K	312M/3510K/5574K/339/6367K	320M/3590K/5714K/333/6472K	322M/3603K/5737K/338/6512K	321M/2864K/4691K/338/5475K	–/–/–/–	–/–/–/–	326M/2860K/4707K/297/5482K
		PII/Win Visual C 6.0	PII/Win gcc 2.95.3	PIV/Linux Northwood gcc 2.95.2	PIV/Linux Northwood gcc 3.1.1	Xeon Model 1 gcc 3.2	Xeon Model 0 egcs 2.91.66	Xeon Model 0 gcc 2.96	AMD Duron Win XP VC 7.0 (.NET)	AMD Duron Win XP gcc 2.95.3	AMD Athlon Linux gcc 2.96
GPS		–/–/–/–	249M/2899K/4571K/309/5194K	760M/9235K/14M/508/16M	757M/9201K/14M/491/16M	236M/2676K/4202K/343/4755K	751M/9177K/14M/470/16M	749M/9143K/14M/437/16M	–/–/–/–	183M/2138K/3340K/241/3796K	237M/2596K/4092K/322/4634K
		Alpha cc	Alpha gcc 2.97	Sun/Sparc V9 450MHz gcc 3.0.4+3.2.1	Sun/Sparc 248MHz gcc 2.95.3	Sun/Sparc V9 338MHz cc	Sun/Sparc V9 338MHz gcc 3.0.4	Sun/Sparc V9 400MHz cc	Sun/Sparc V9 400MHz gcc 3.0.4	Sun 450MHz cc	Sun 333MHz gcc 3.2.1
GPS		–/–/–/–	71M/811K/1056K/144/1200K	–/–/–/–	–/–/–/–	–/–/–/–	–/–/–/–	–/–/–/–	–/–/–/–	–/–/–/–	–/–/–/–

[1]The figures in the entries are of parameter generation/key generation/commitment/answer/verification time. All of them are measured in cycles/invocation.

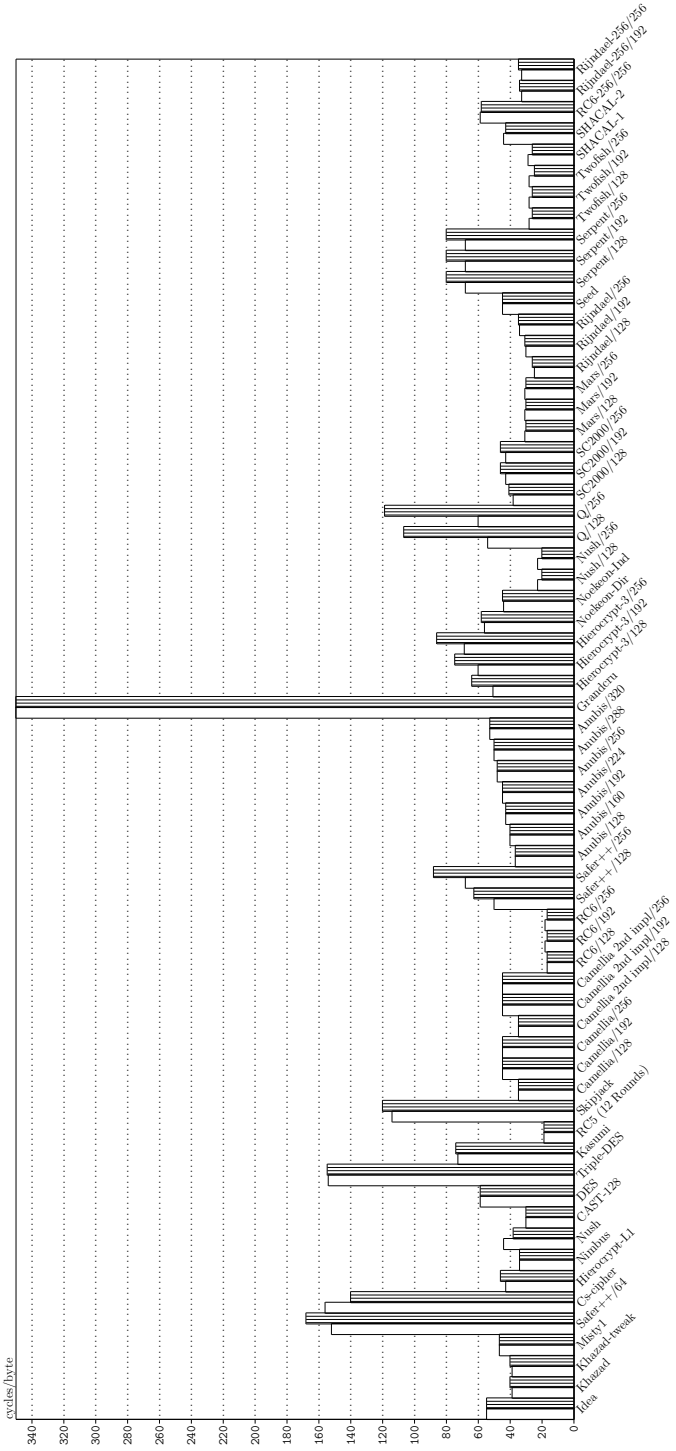


Figure 2: Block Ciphers on PIII/Linux

Legend:

encryption, decryption,

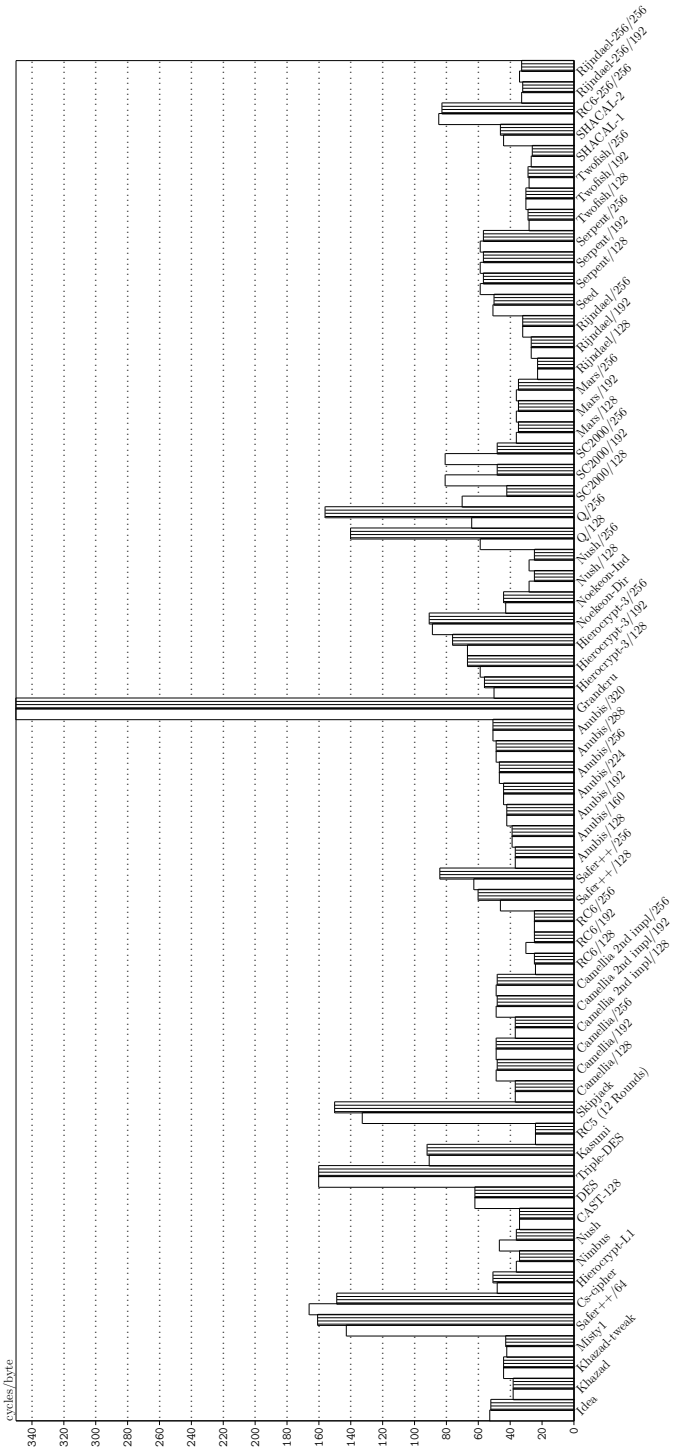


Figure 3: Block Ciphers on PIII/MS

Legend:

encryption, decryption,

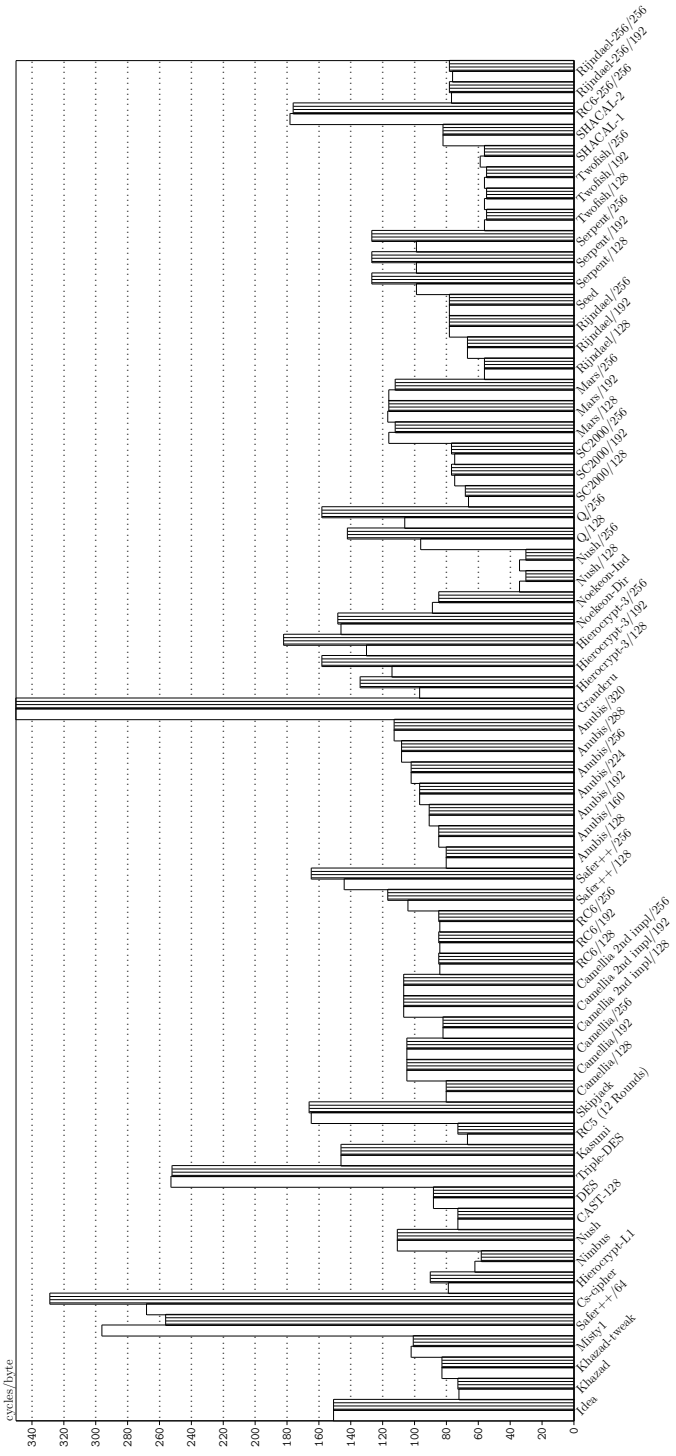


Figure 4: Block Ciphers on PI/MMX

Legend:

encryption, decryption,

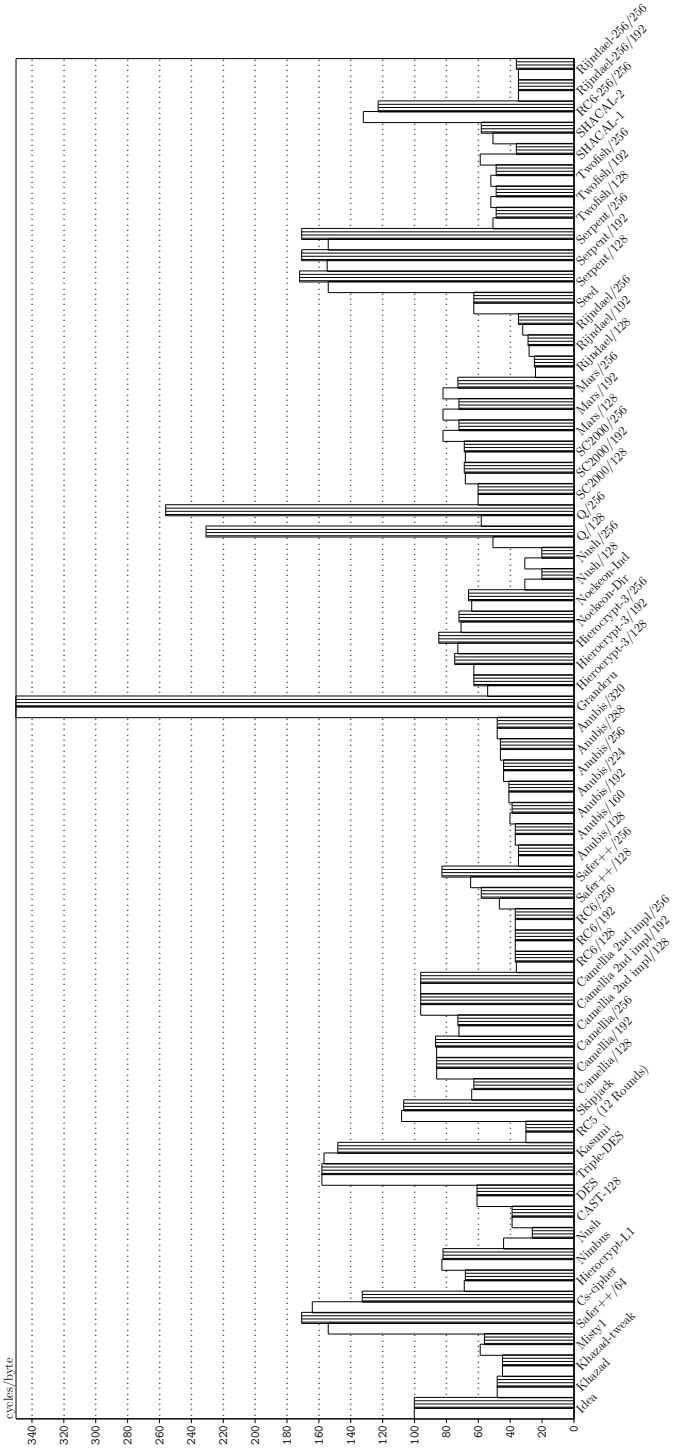


Figure 5: Block Ciphers on Pentium4

Legend:

encryption, decryption,

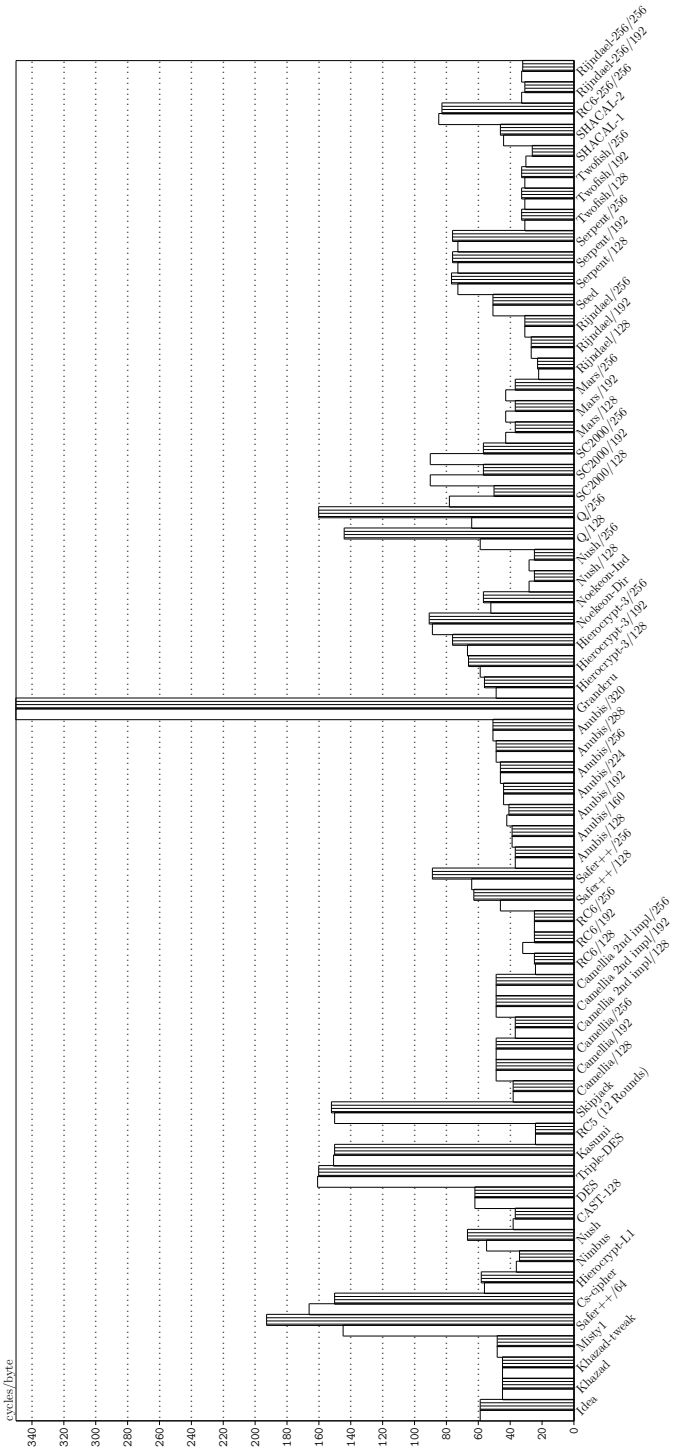


Figure 6: Block Ciphers on Pentium2

Legend:

encryption, decryption,

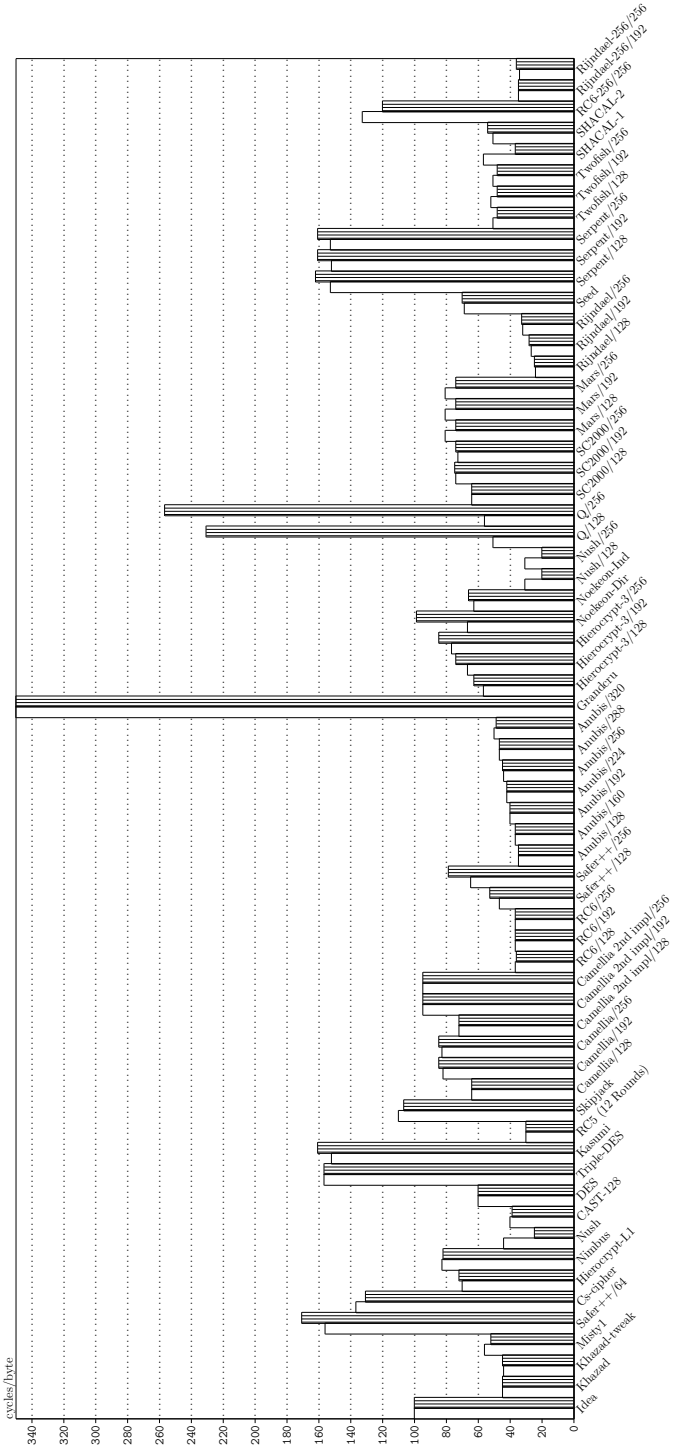


Figure 7: Block Ciphers on Xeon

Legend:

encryption, decryption,

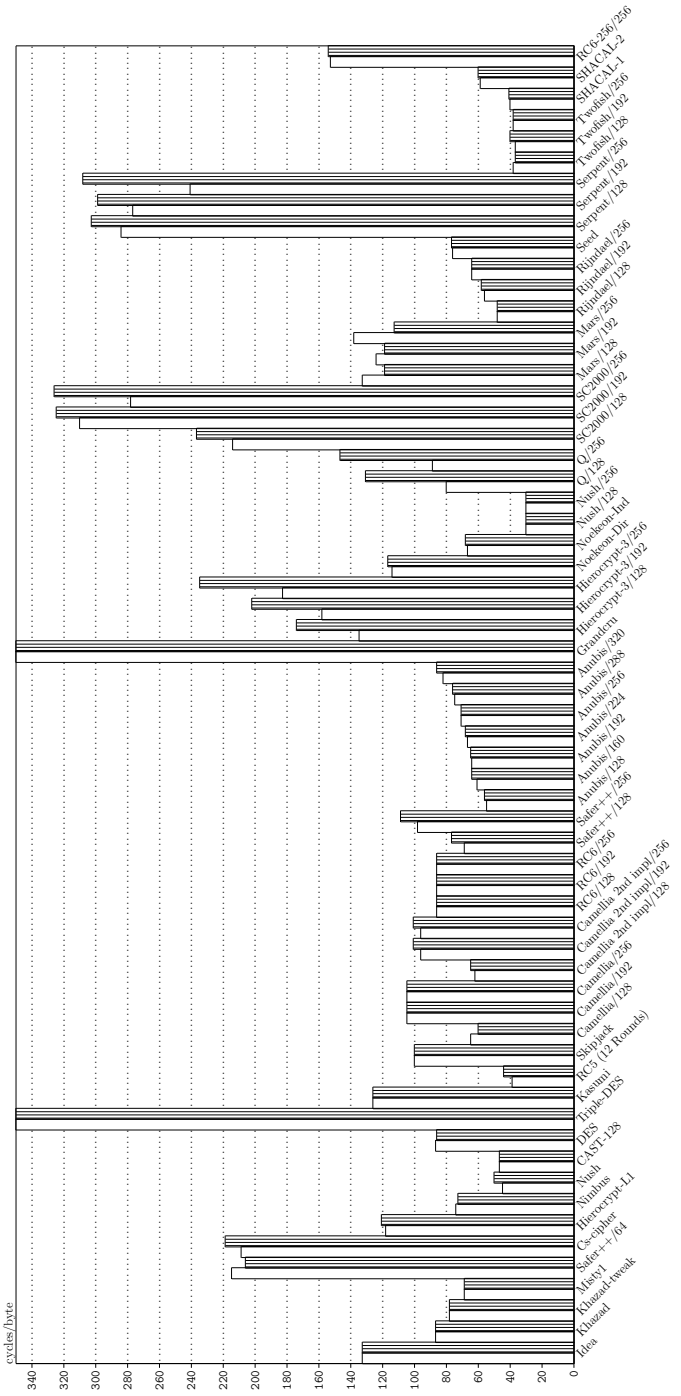


Figure 8: Block Ciphers on 486

Legend:

encryption, decryption,

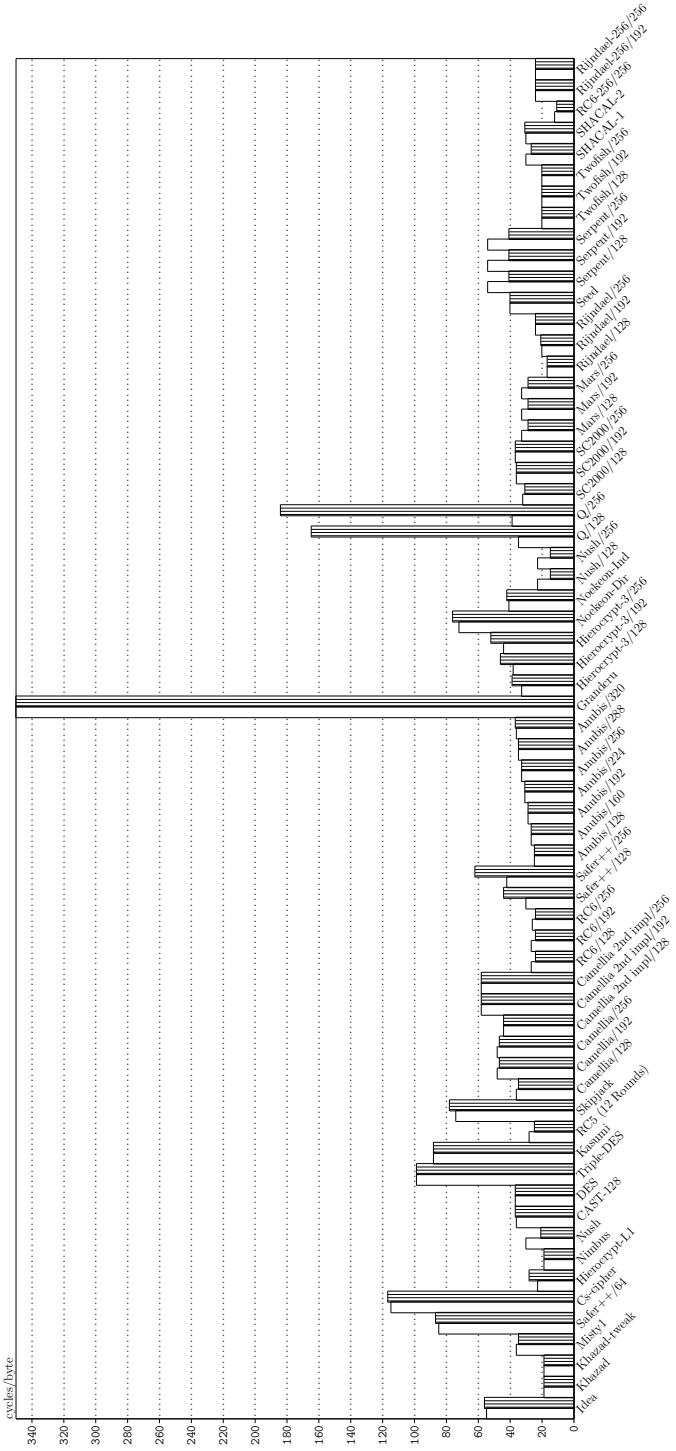


Figure 9: Block Ciphers on Alpha

Legend:

encryption, decryption,

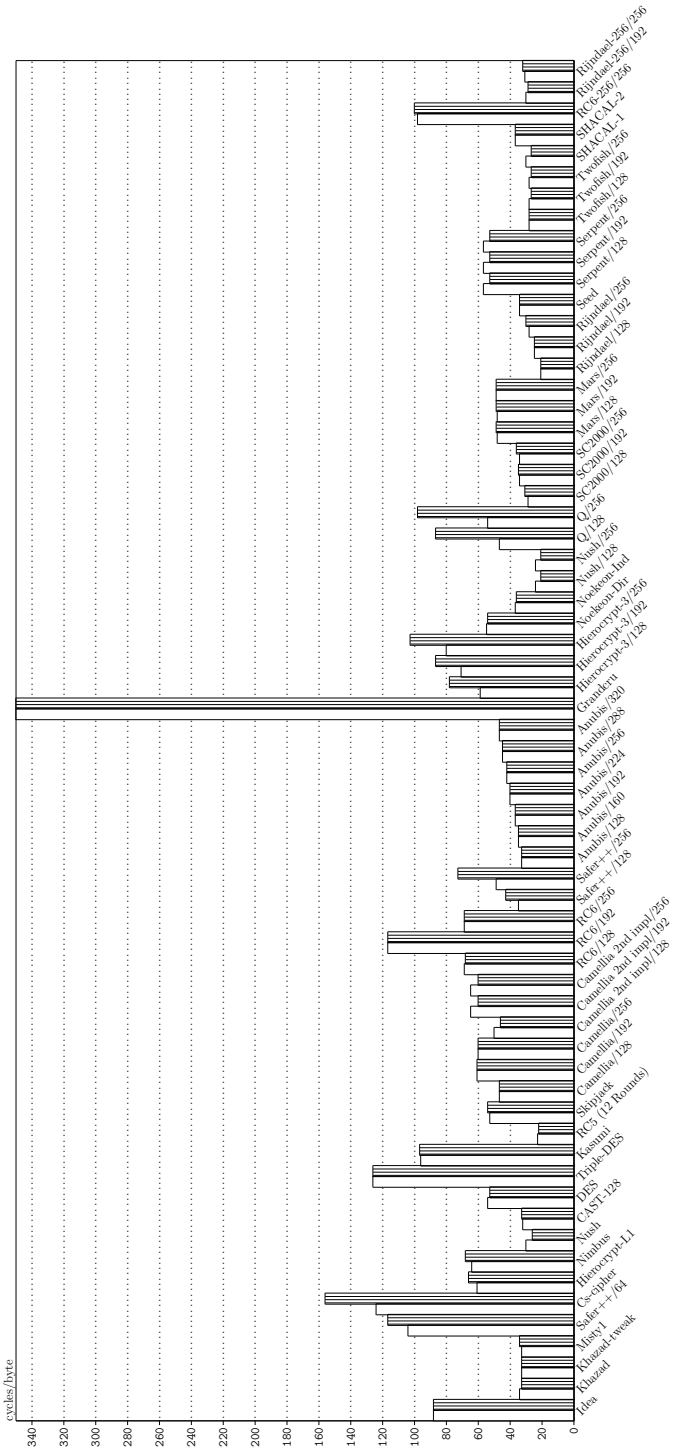


Figure 10: Block Ciphers on Sparc V9

Legend:

encryption, decryption,

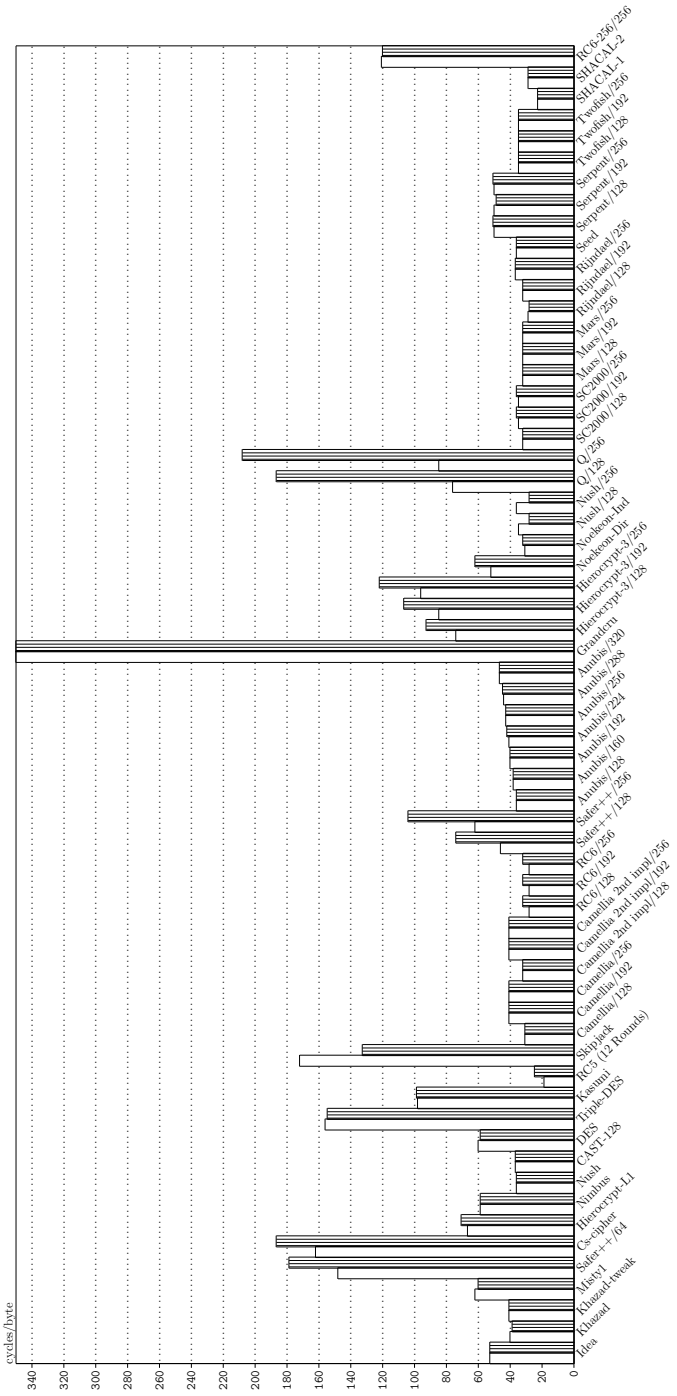


Figure 11: Block Ciphers on Macintosh

Legend:

encryption, decryption,

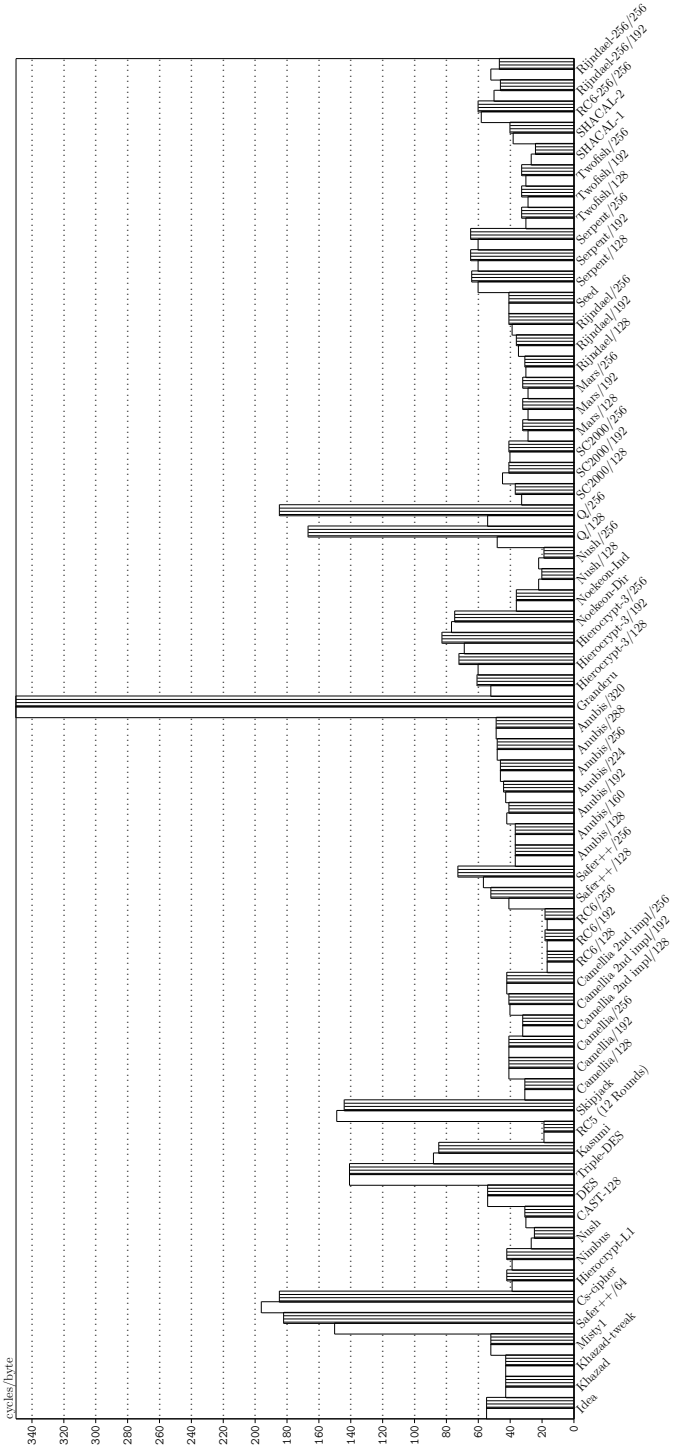


Figure 12: Block Ciphers on AMD

Legend:

encryption, decryption,

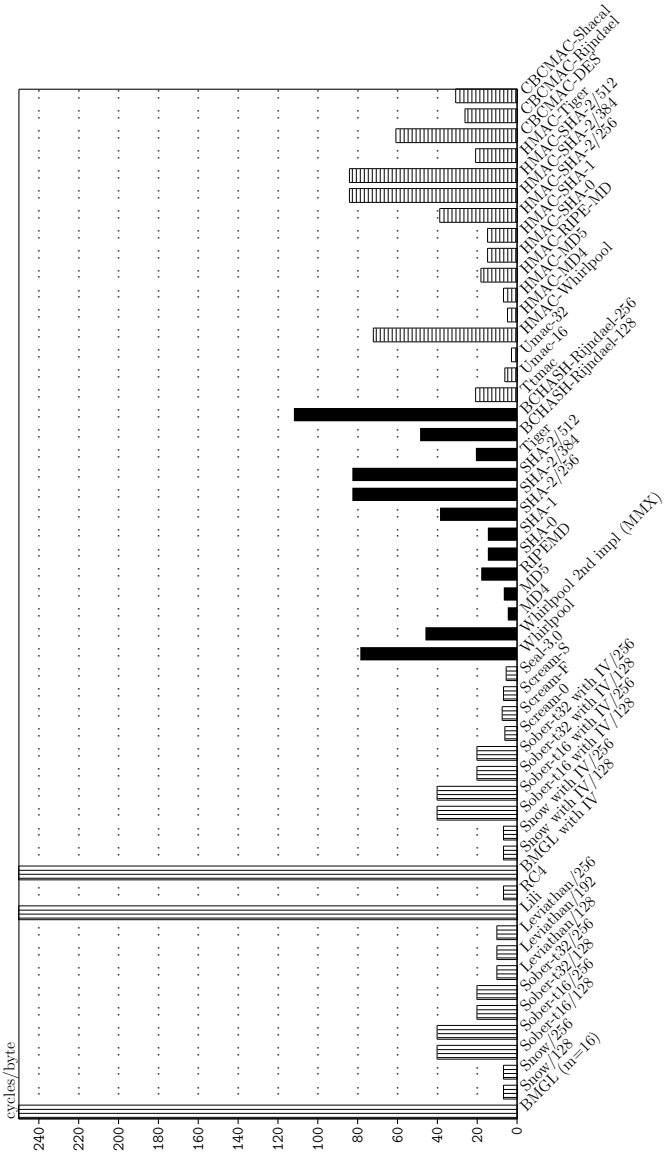


Figure 13: Stream Ciphers, Hash Functions and MACs on PIII/Linux

Legend:

stream ciphers, hash function, MAC,

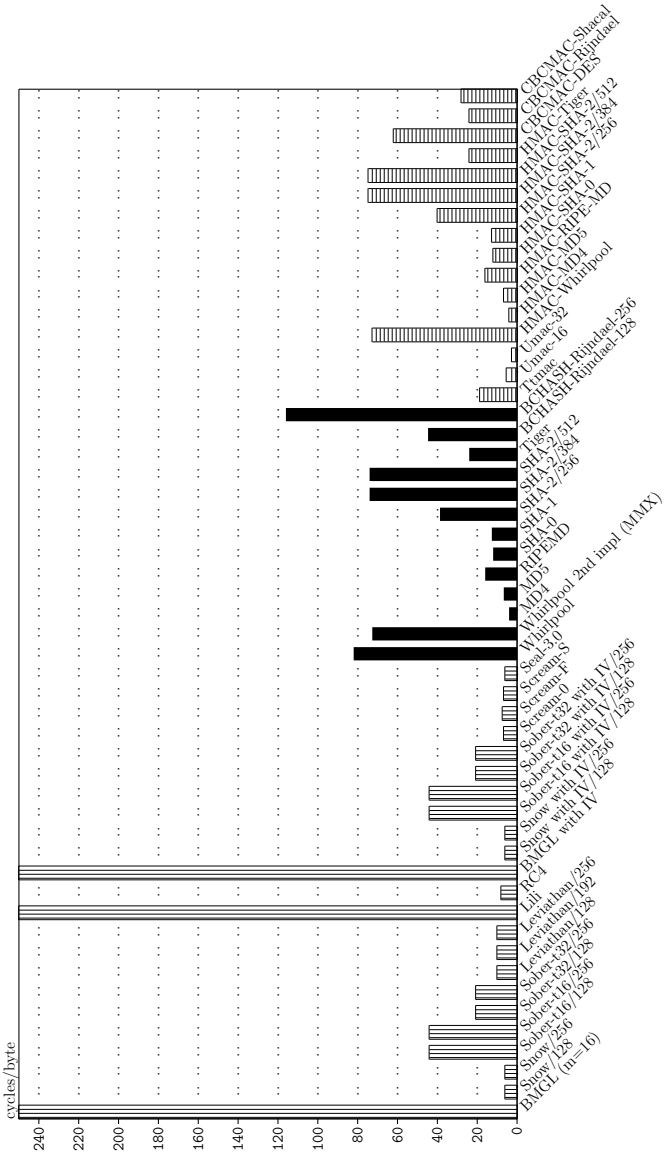


Figure 14: Stream Ciphers, Hash Functions and MACs on PIII/MS

Legend:

stream ciphers, hash function, MAC,

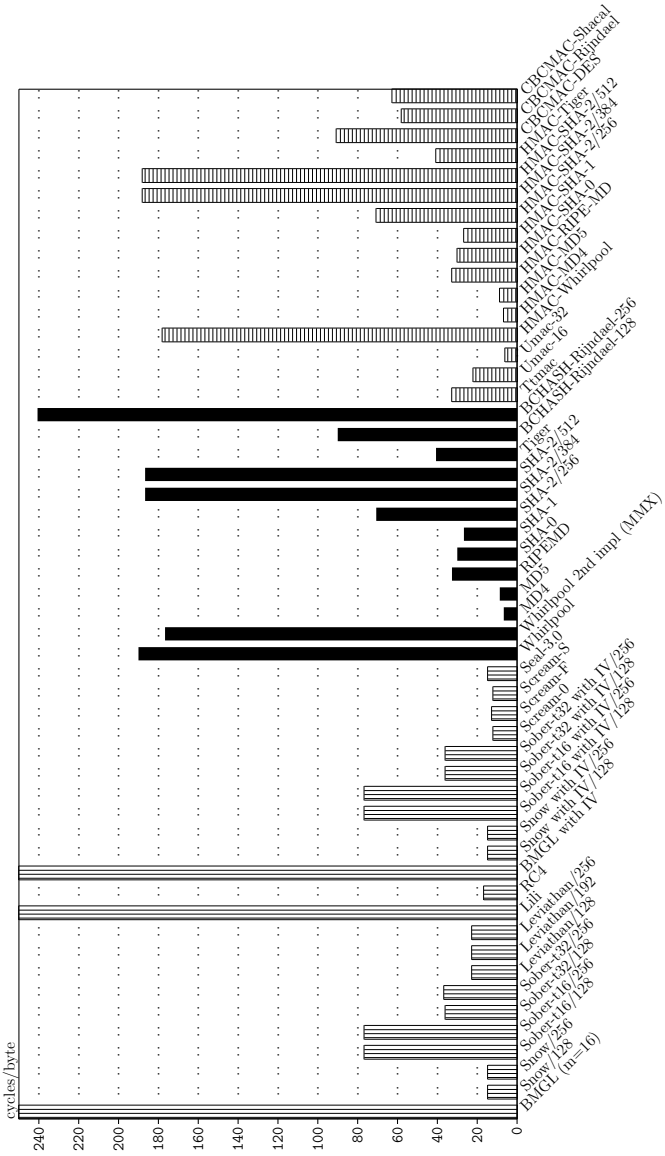


Figure 15: Stream Ciphers, Hash Functions and MACs on PI/MMX

Legend:

stream ciphers, hash function, MAC,

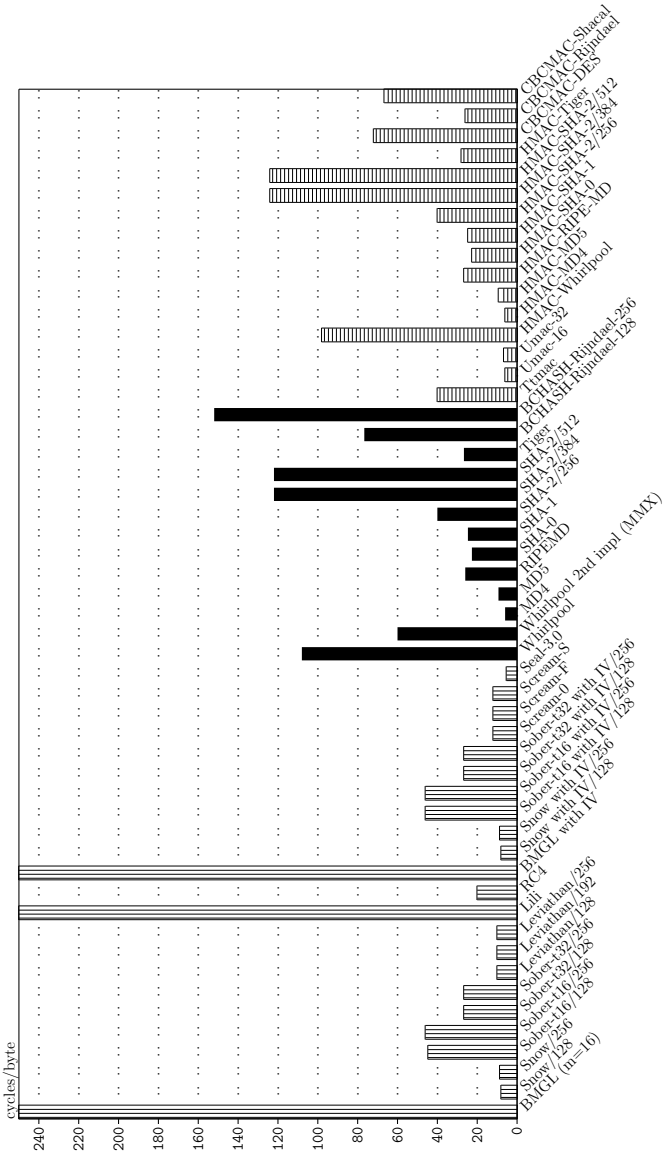


Figure 16: Stream Ciphers, Hash Functions and MACs on Pentium4

Legend:

stream ciphers, hash function, MAC,

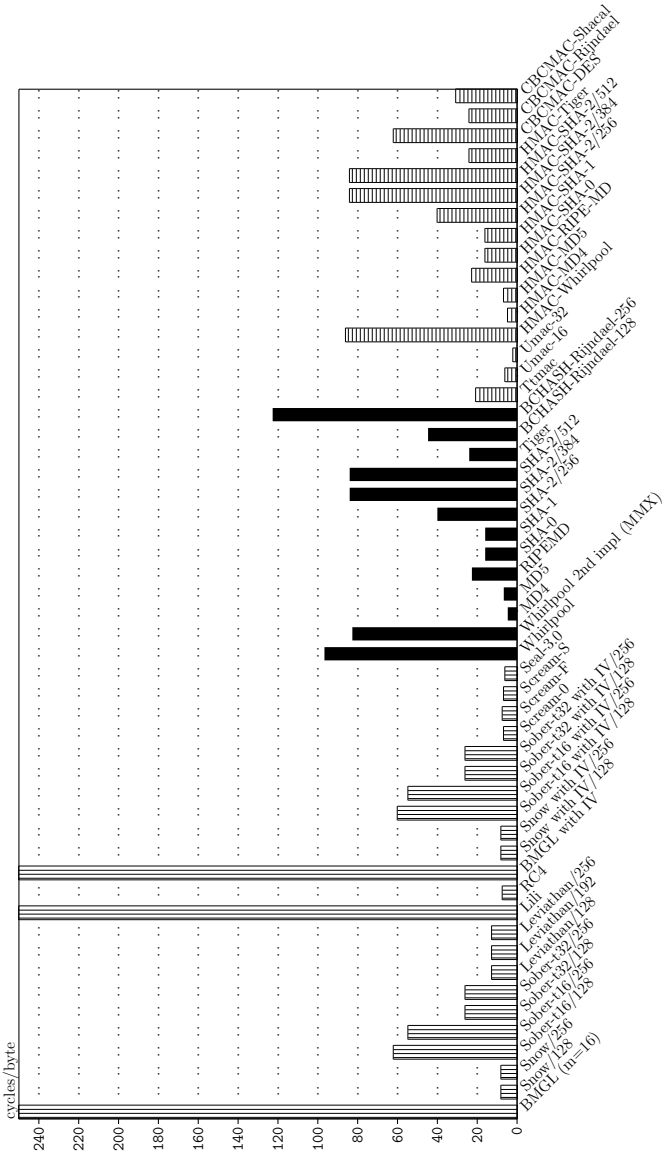


Figure 17: Stream Ciphers, Hash Functions and MACs on Pentium2

Legend:

stream ciphers, hash function, MAC,

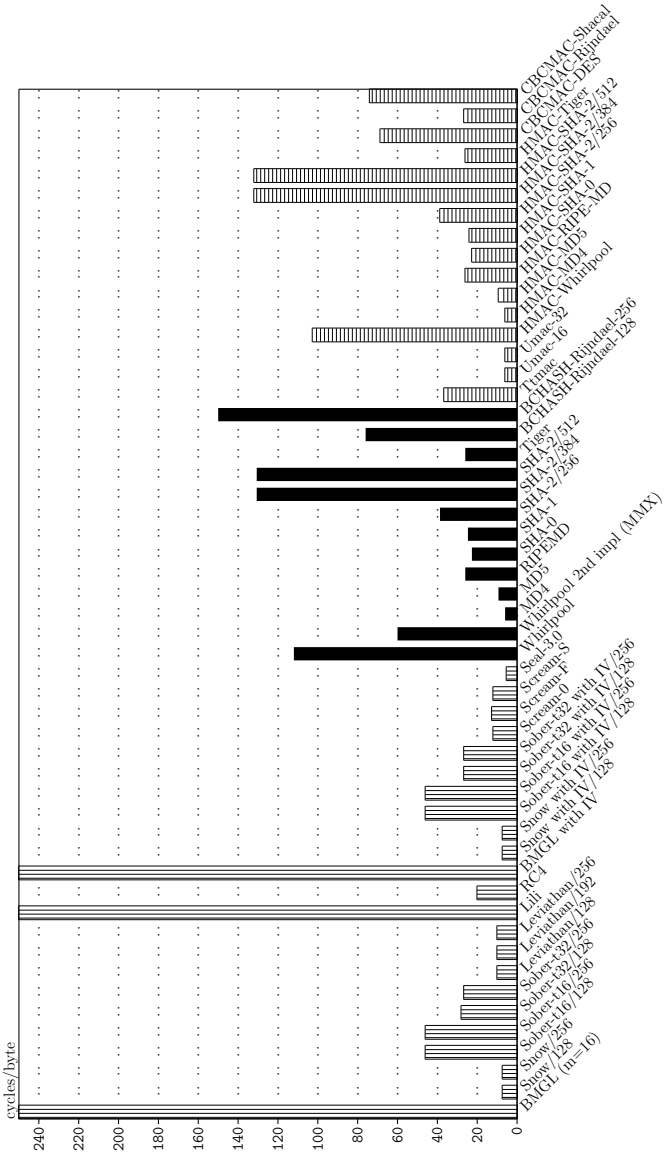


Figure 18: Stream Ciphers, Hash Functions and MACs on XEON

Legend:

stream ciphers, hash function, MAC,

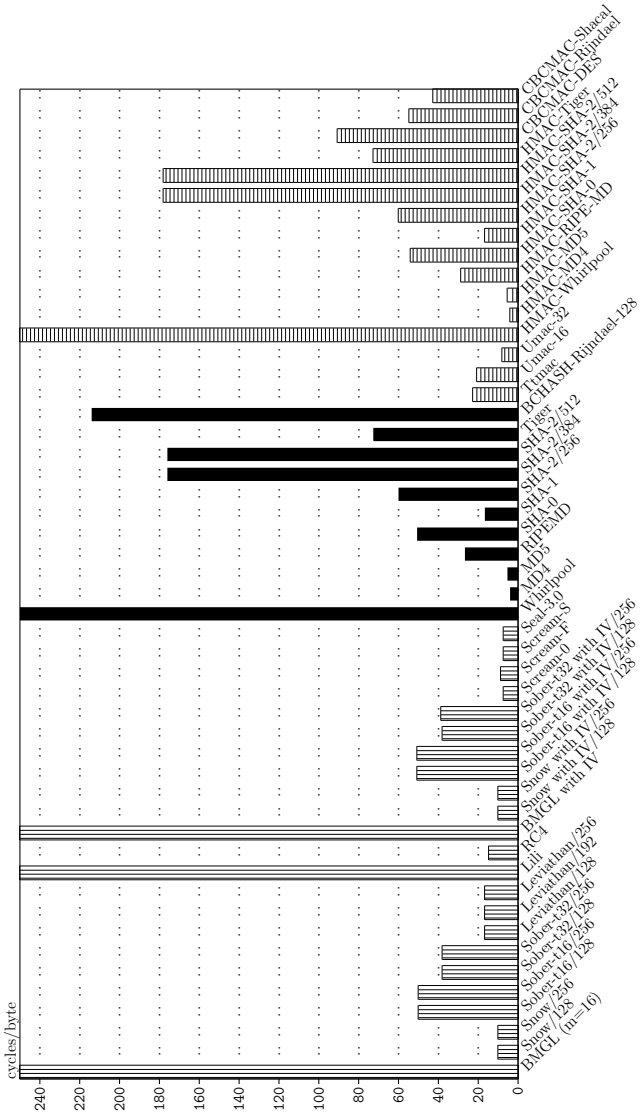


Figure 19: Stream Ciphers, Hash Functions and MACs on 486

Legend:

stream ciphers, hash function, MAC,

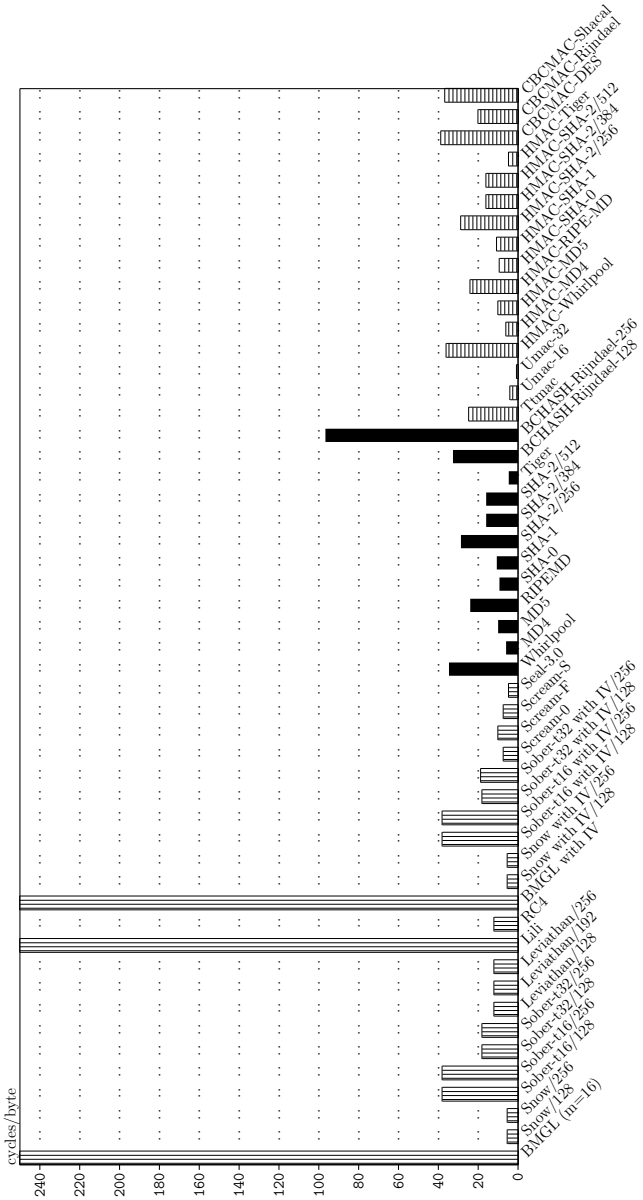


Figure 20: Stream Ciphers, Hash Functions and MACs on Alpha

Legend:

stream ciphers, hash function, MAC,

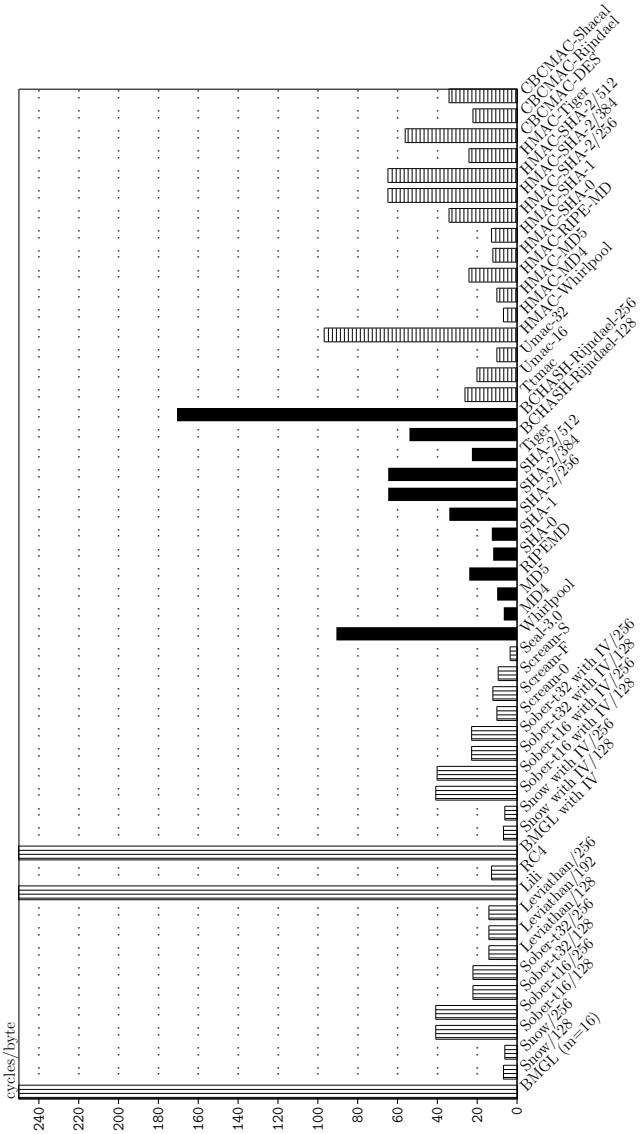


Figure 21: Stream Ciphers, Hash Functions and MACs on Sparc V9

Legend:

stream ciphers, hash function, MAC,

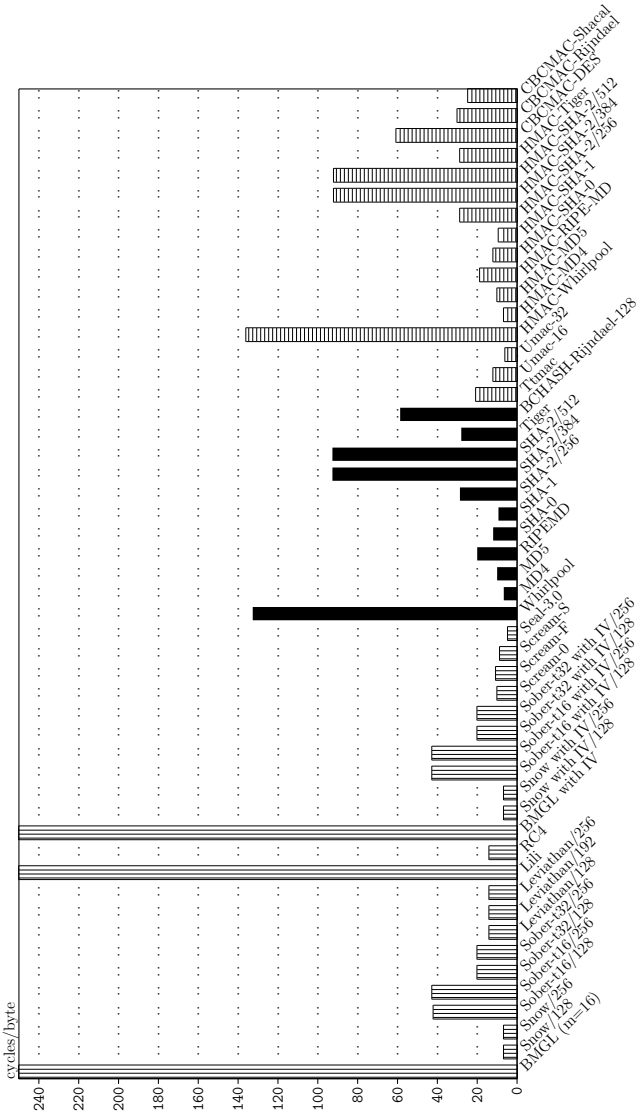


Figure 22: Stream Ciphers, Hash Functions and MACs on Macintosh

Legend:

stream ciphers, hash function, MAC,

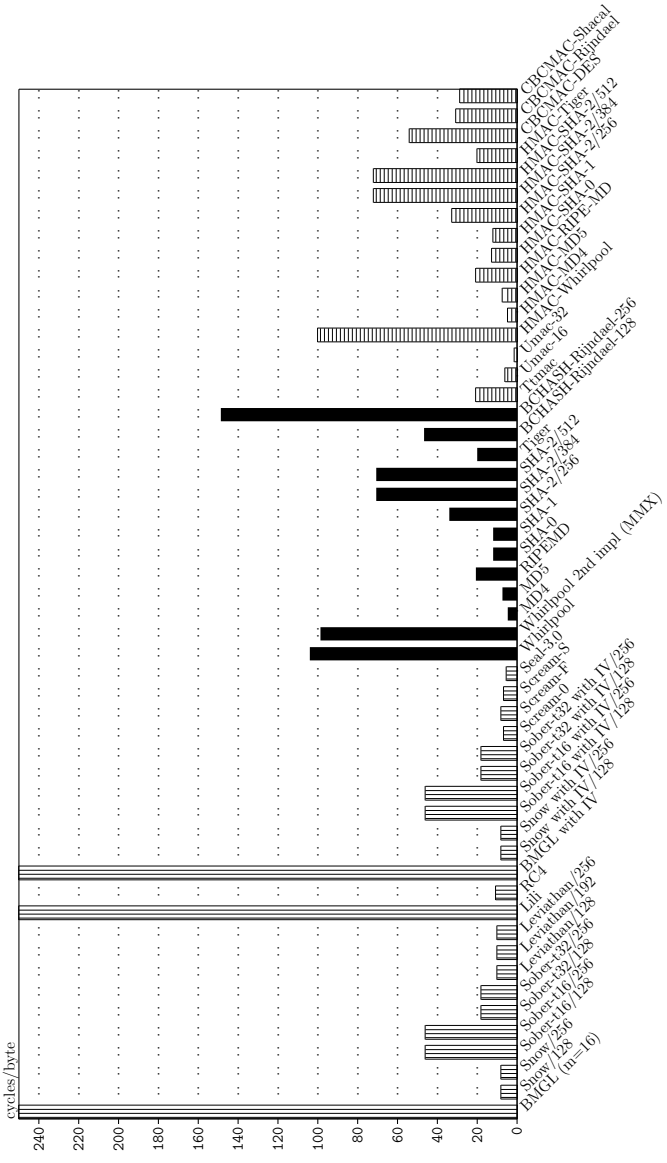


Figure 23: Stream Ciphers, Hash Functions and MACs on AMD

Legend:

stream ciphers, hash function, MAC,

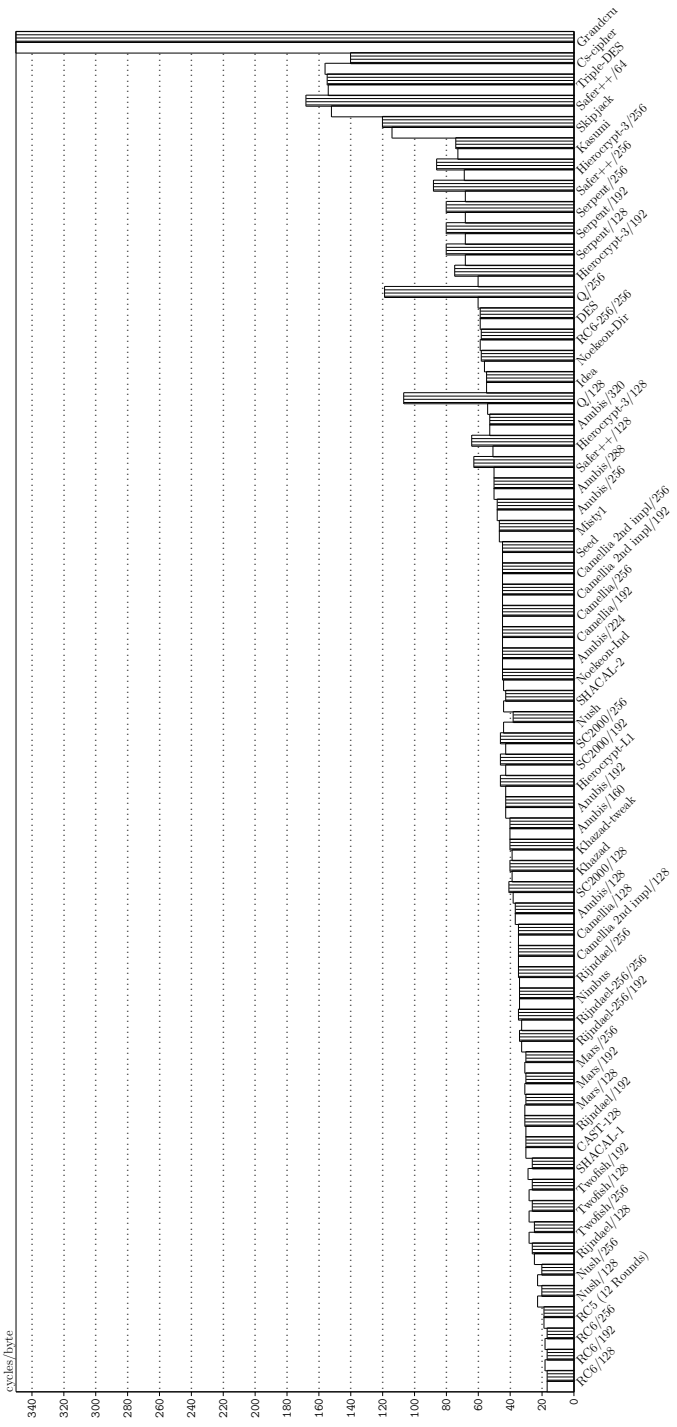


Figure 24: Block Ciphers on PIII/Linux (Sorted)

Legend:

encryption, decryption,

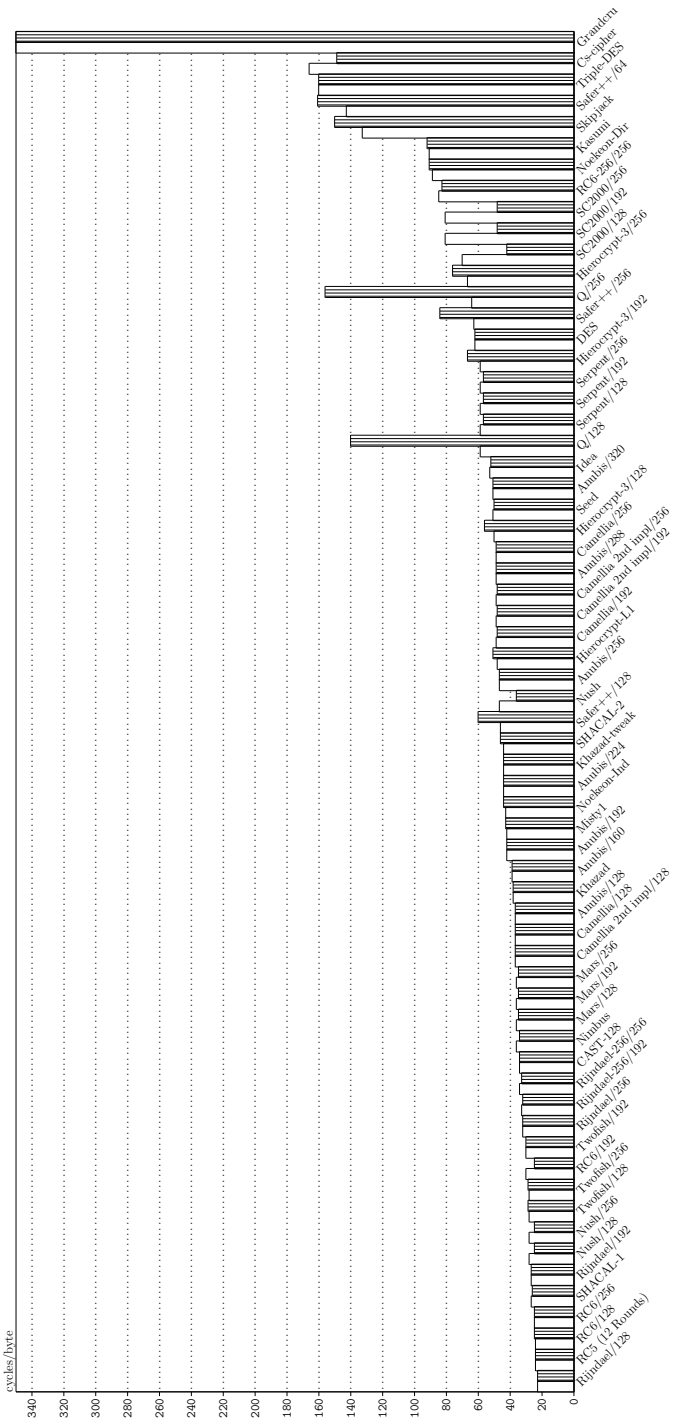


Figure 25: Block Ciphers on PIII/MS (Sorted)

Legend:

encryption, decryption,

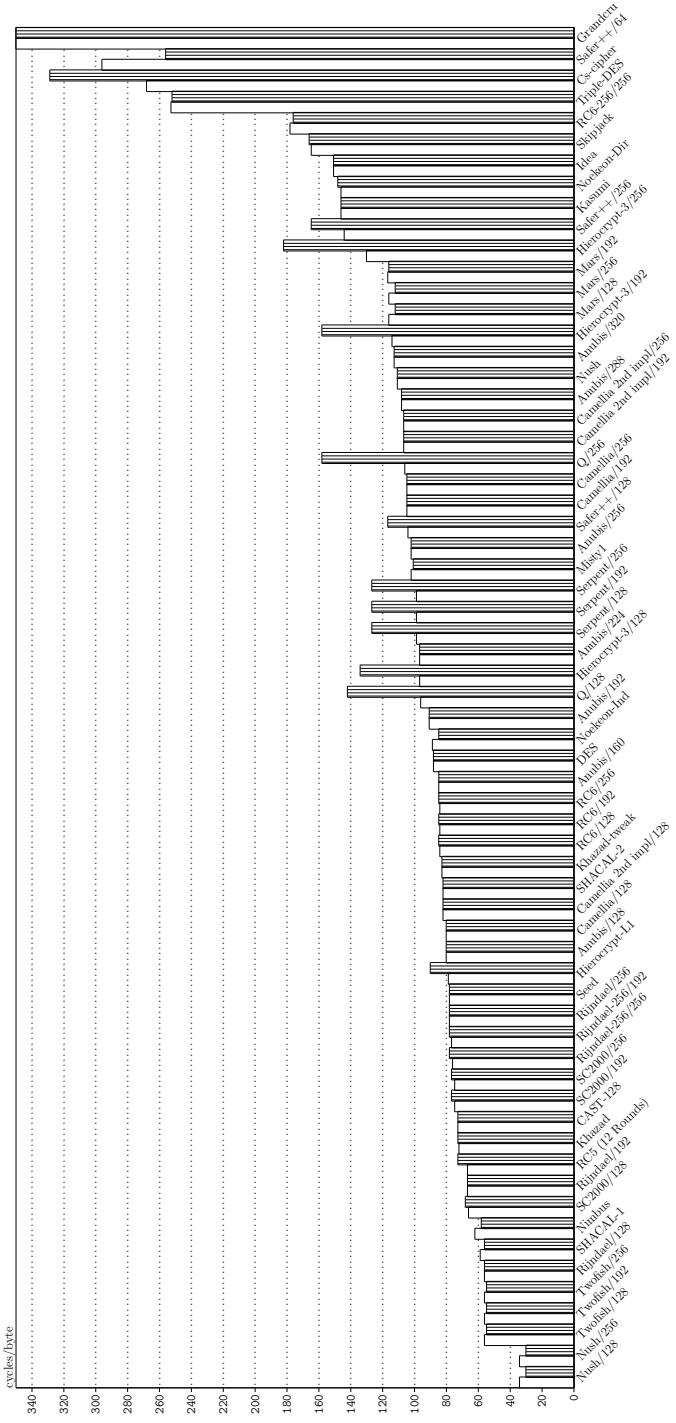


Figure 26: Block Ciphers on PI/MMX (Sorted)

Legend:

encryption, decryption,

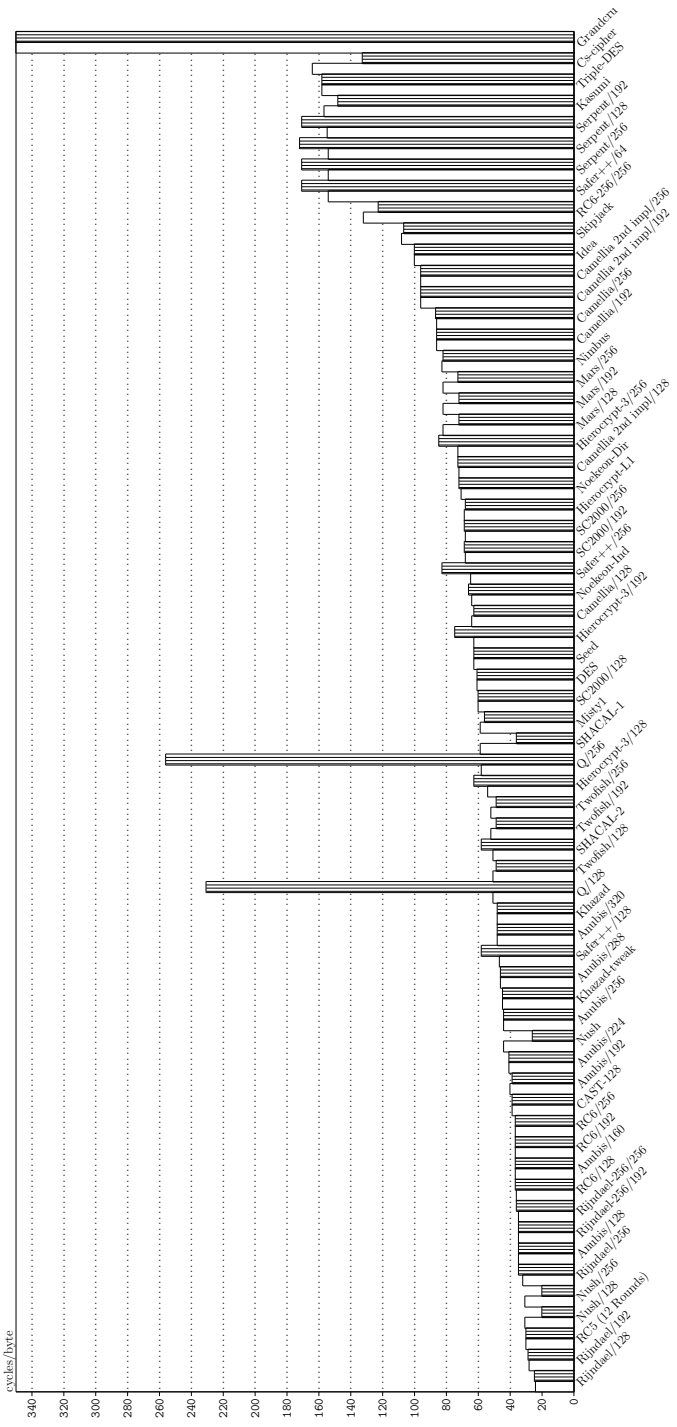


Figure 27: Block Ciphers on Pentium4 (Sorted)

Legend:

encryption, decryption,

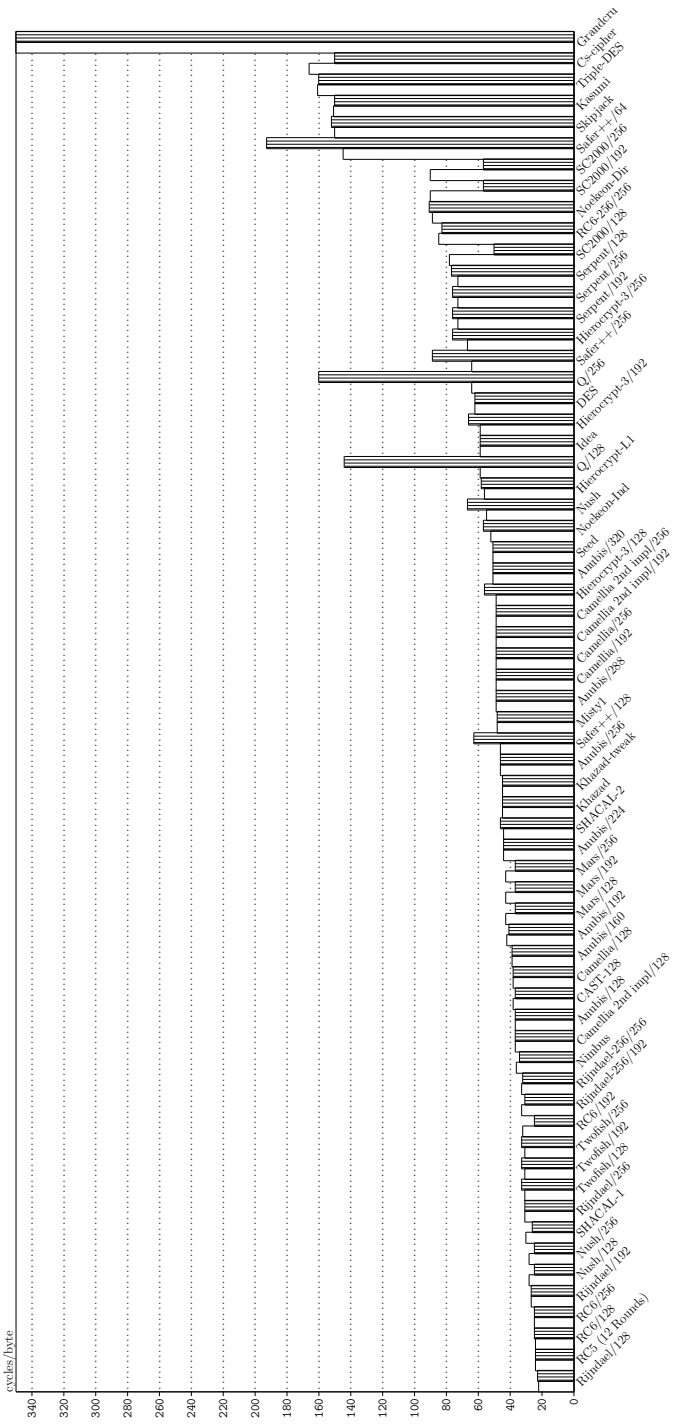


Figure 28: Block Ciphers on Pentium2 (Sorted)

Legend:

encryption, decryption,

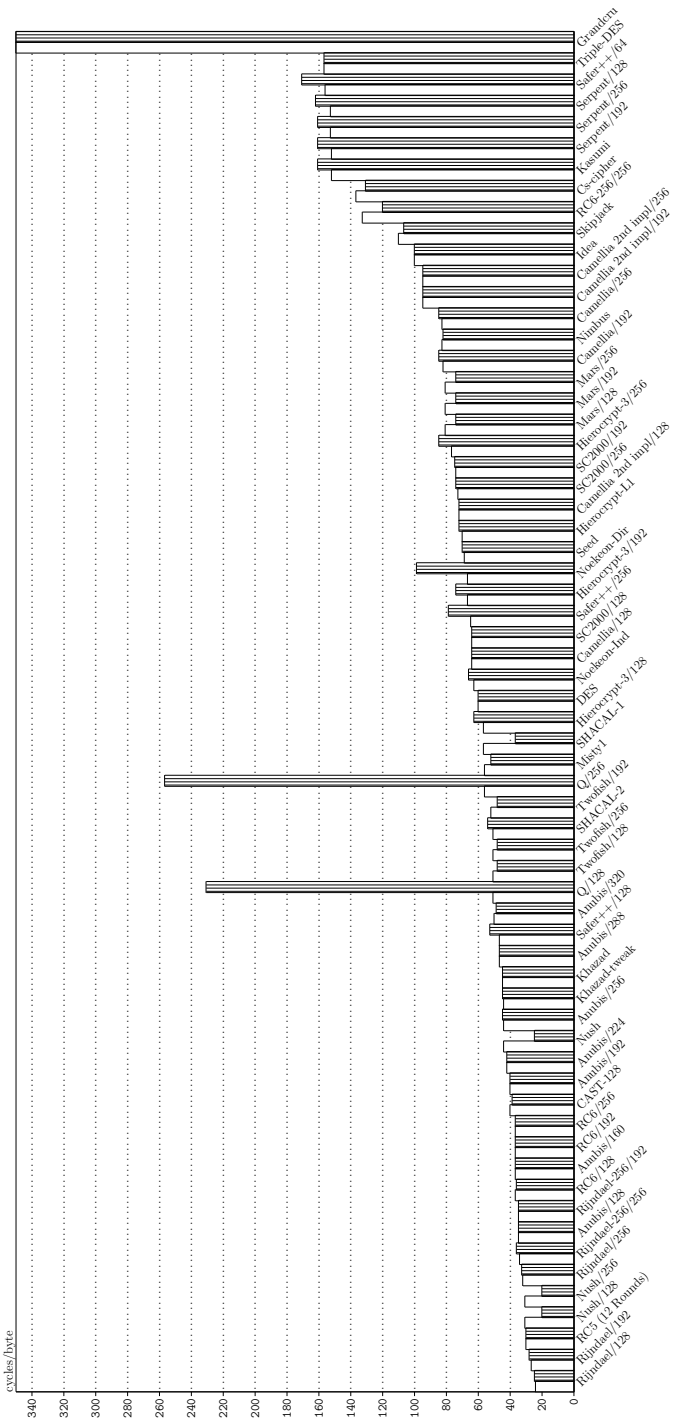


Figure 29: Block Ciphers on Xeon (Sorted)

Legend:

encryption, decryption,

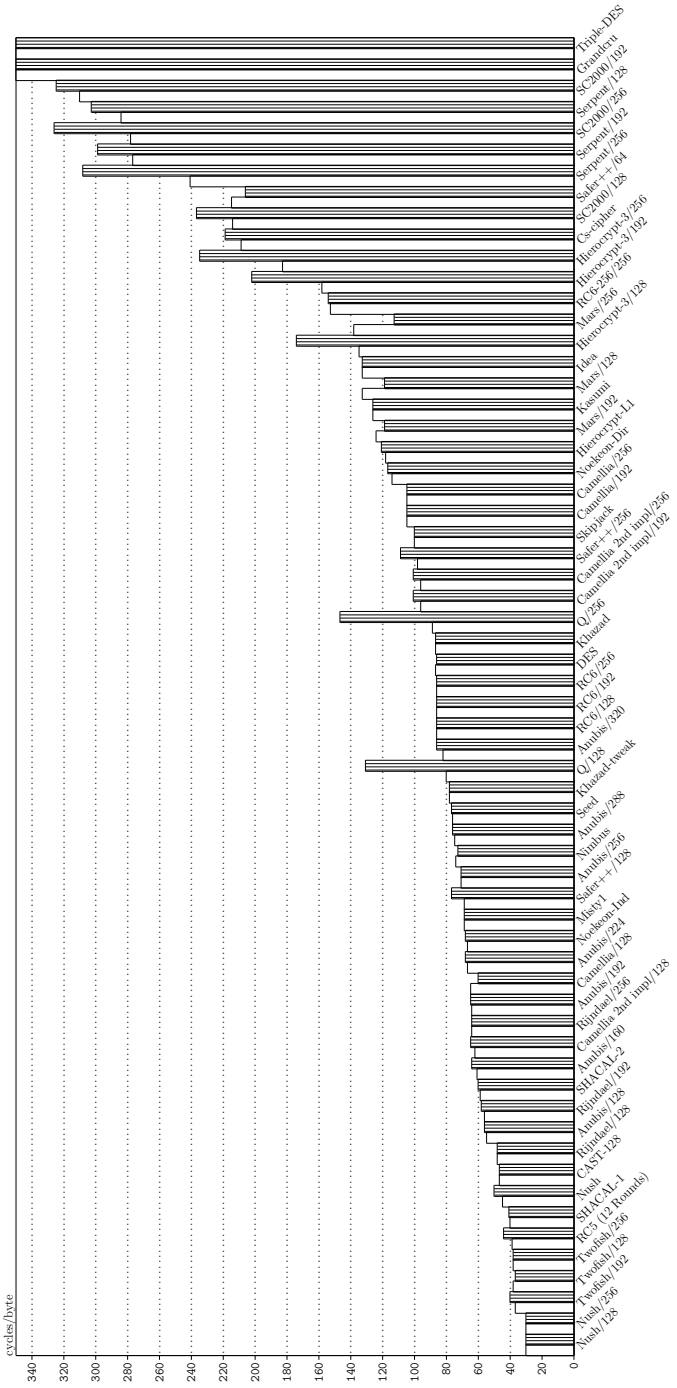


Figure 30: Block Ciphers on 486 (Sorted)

Legend:

encryption, decryption,

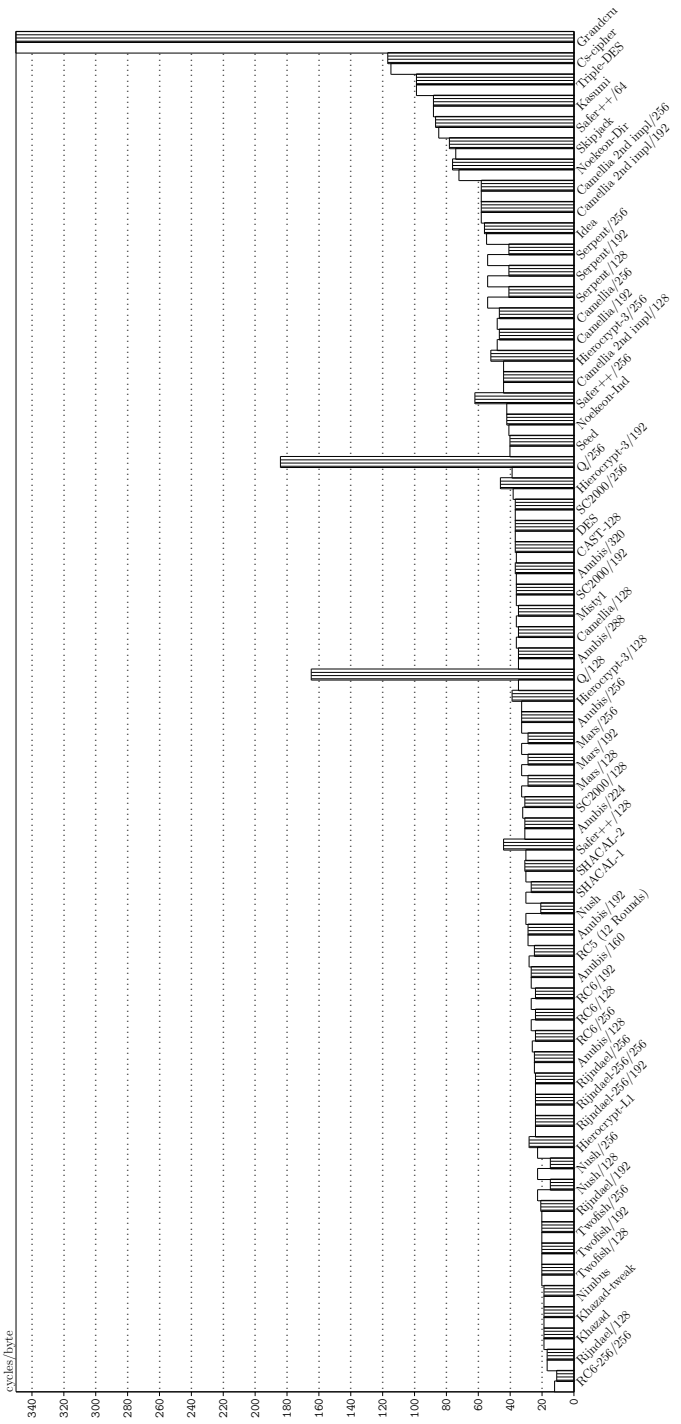


Figure 31: Block Ciphers on Alpha (Sorted)

Legend:

encryption, decryption,

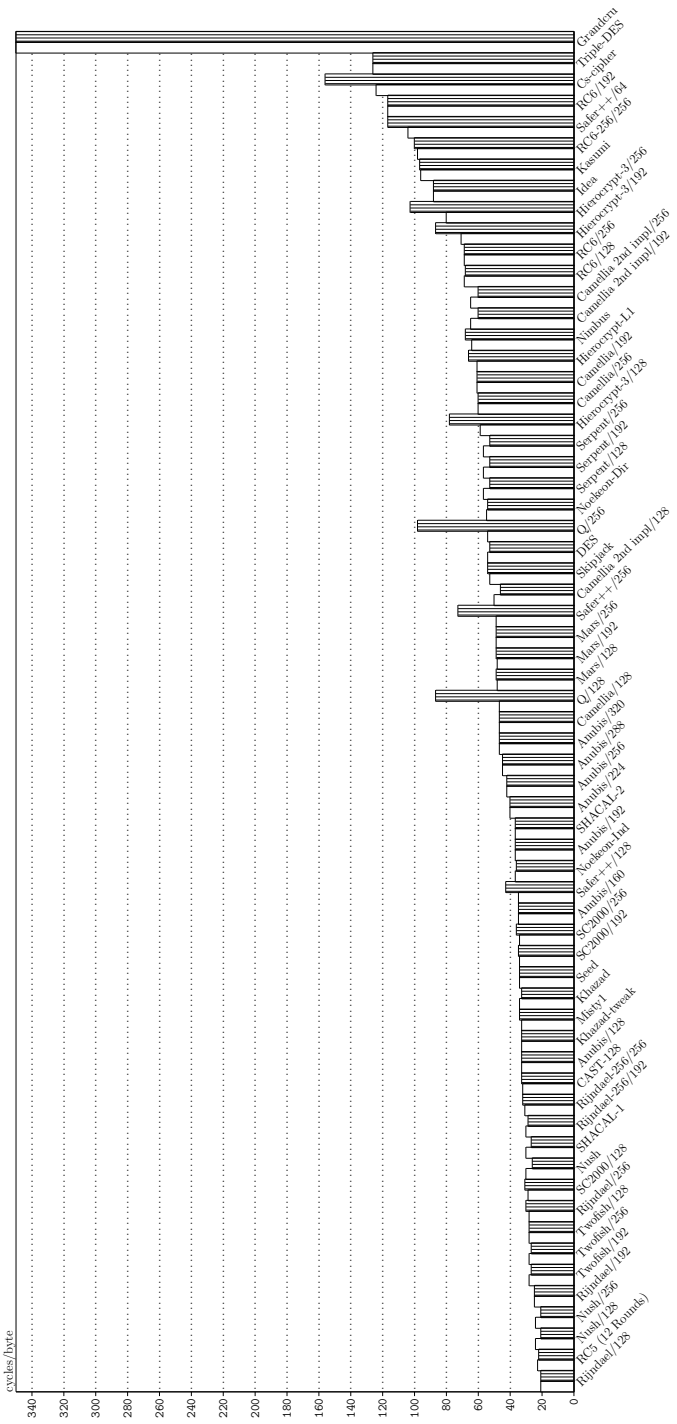


Figure 32: Block Ciphers on Sparc V9 (Sorted)

Legend:

encryption, decryption,

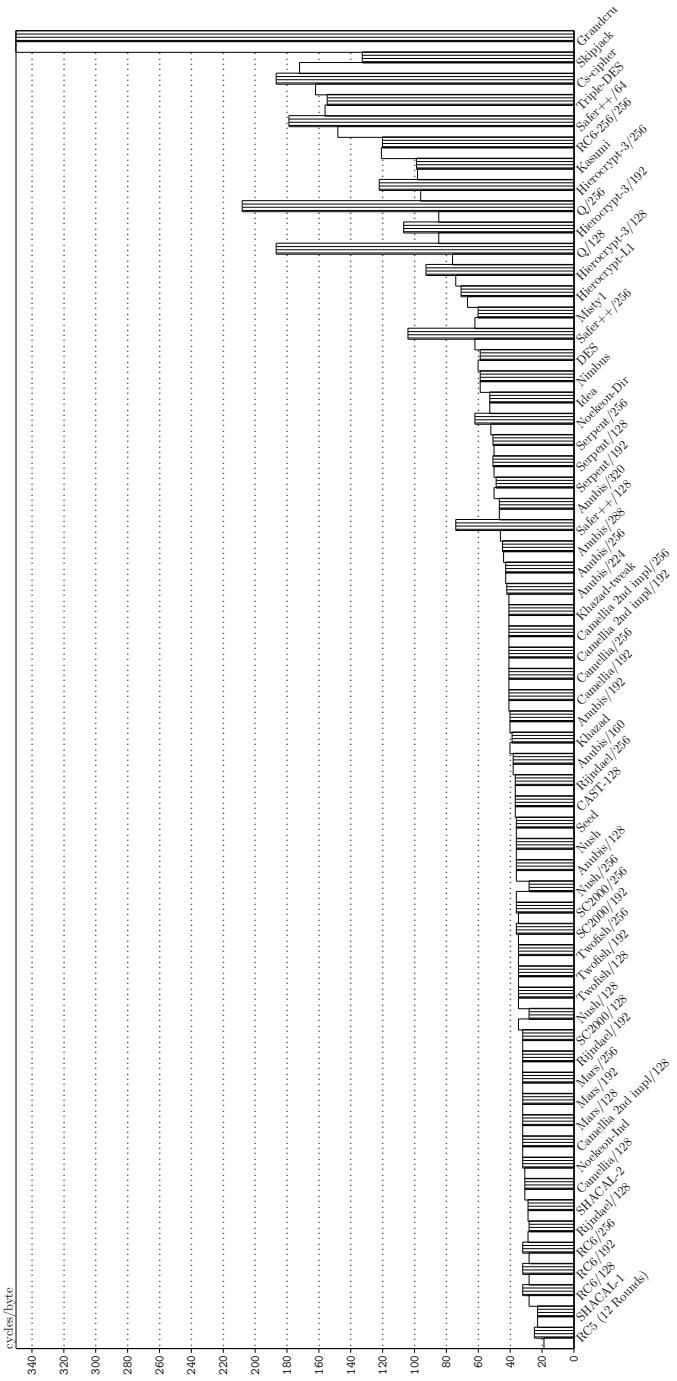


Figure 33: Block Ciphers on Macintosh (Sorted)

Legend:

encryption, decryption,

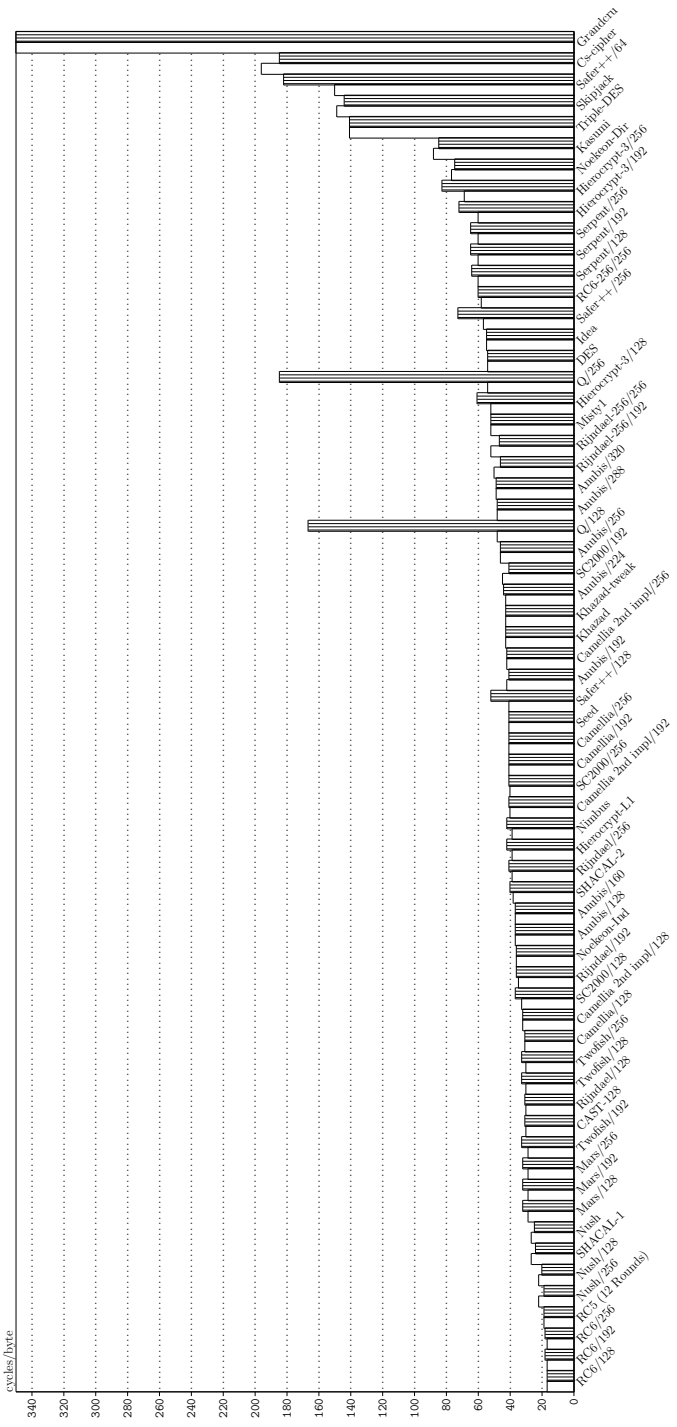


Figure 34: Block Ciphers on AMD (Sorted)

Legend:

encryption, decryption,

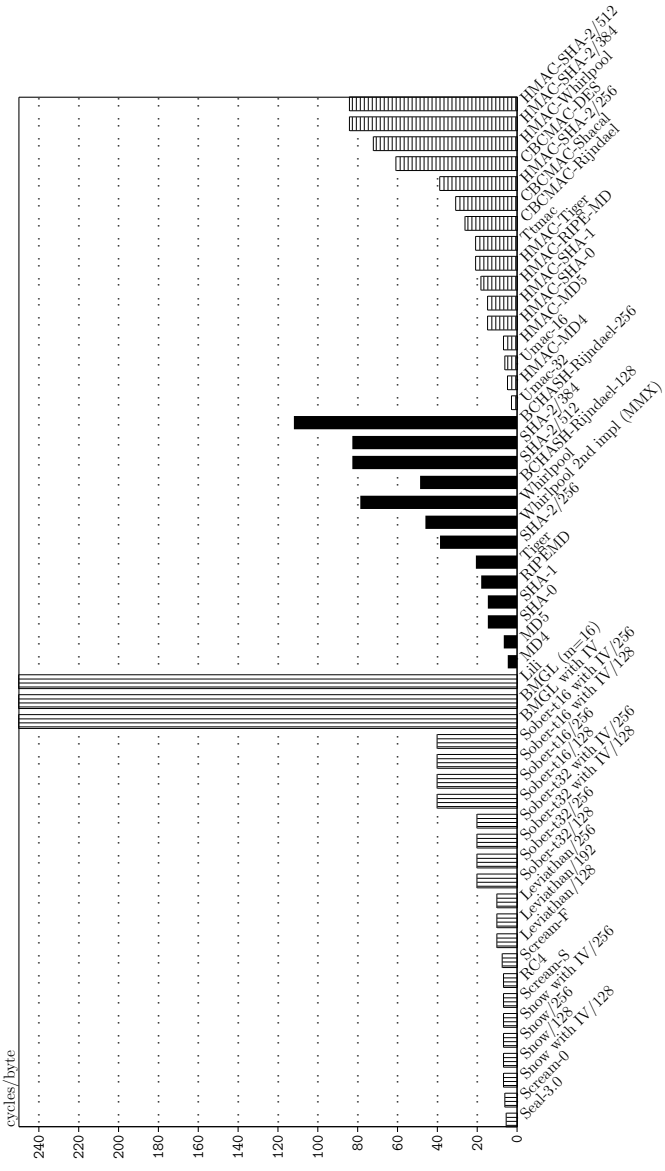


Figure 35: Stream Ciphers, Hash Functions and MACs on PIII/Linux (Sorted)
Legend:

stream ciphers, hash function, MAC,

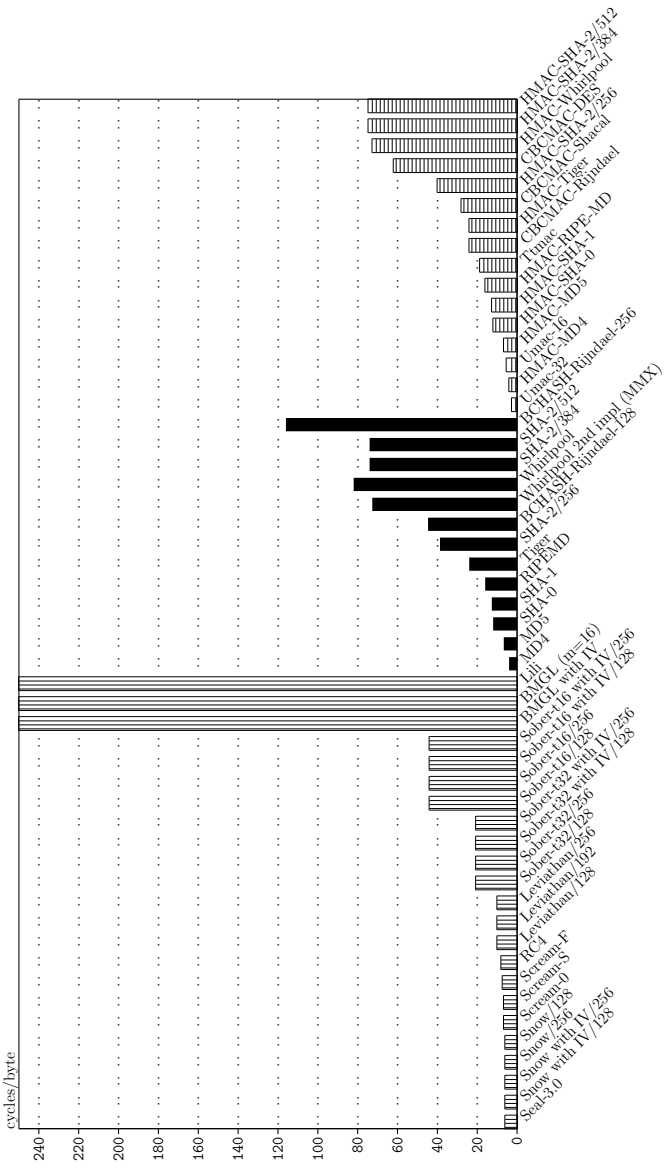


Figure 36: Stream Ciphers, Hash Functions and MACs on PIII/MS (Sorted)
Legend:

stream ciphers, hash function, MAC,

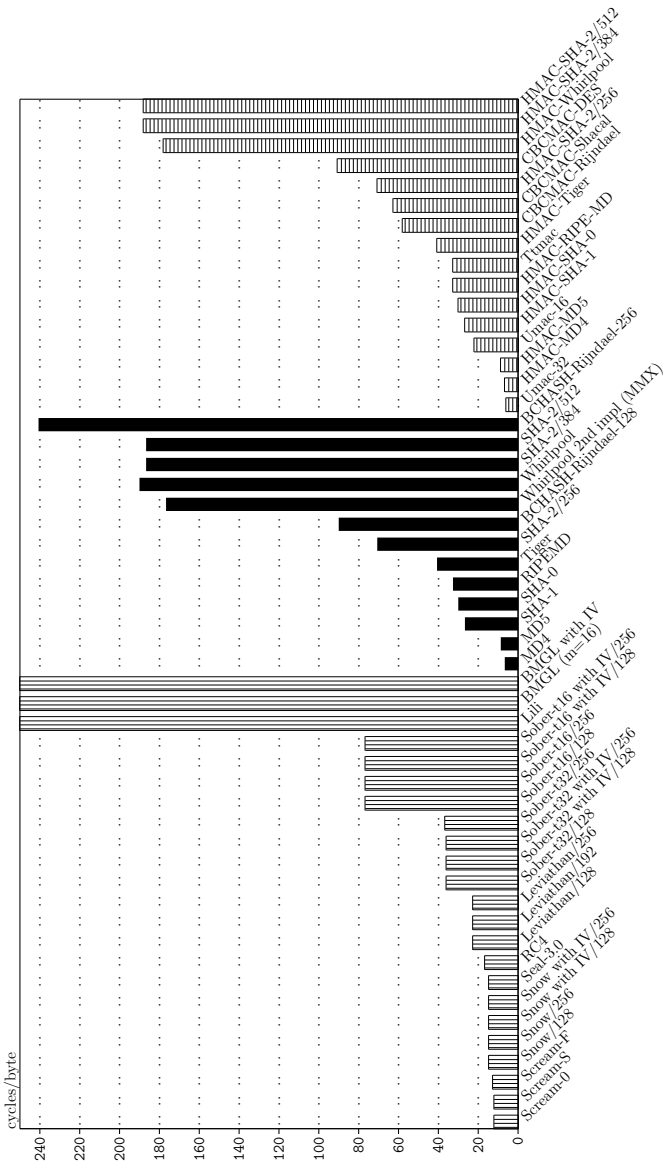


Figure 37: Stream Ciphers, Hash Functions and MACs on PI/MMX (Sorted)
Legend:

stream ciphers, hash function, MAC,

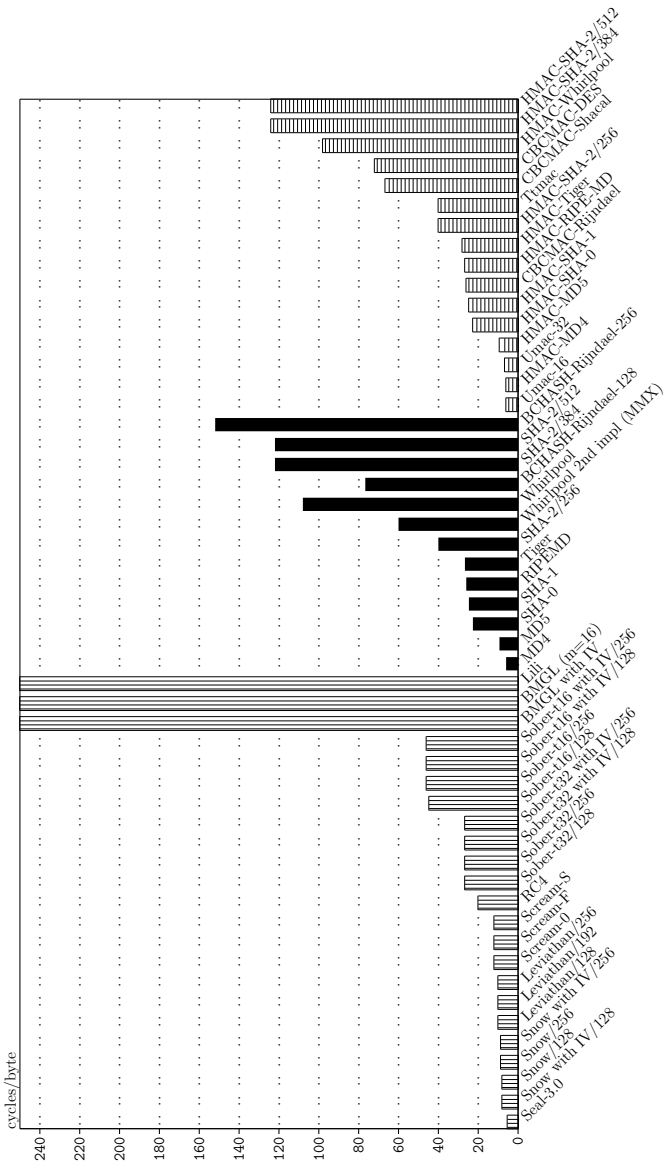


Figure 38: Stream Ciphers, Hash Functions and MACs on Pentium4 (Sorted)
Legend:

stream ciphers, hash function, MAC,

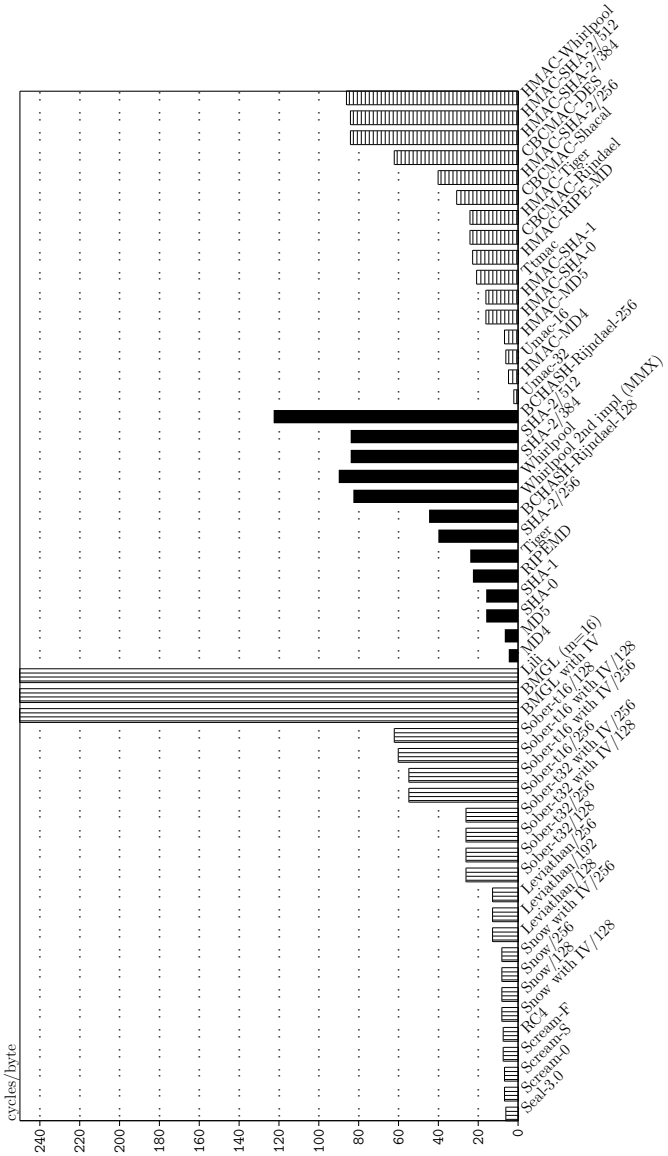


Figure 39: Stream Ciphers, Hash Functions and MACs on Pentium2 (Sorted)
Legend:

stream ciphers, hash function, MAC,

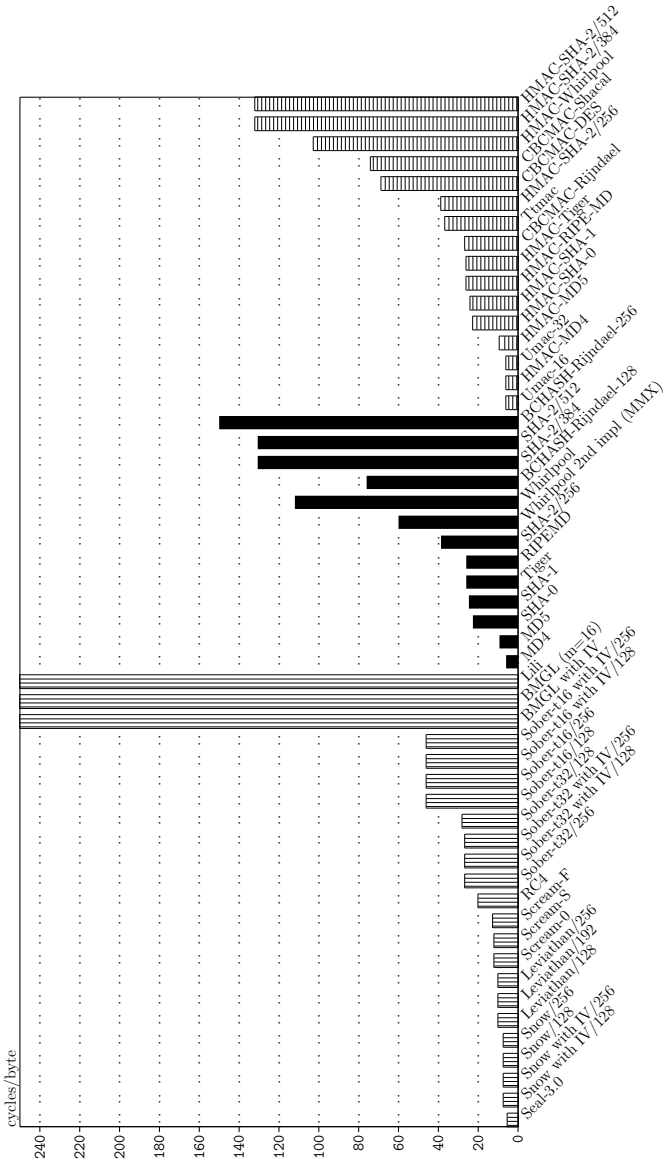


Figure 40: Stream Ciphers, Hash Functions and MACs on XEON (Sorted)
Legend:

stream ciphers, hash function, MAC,

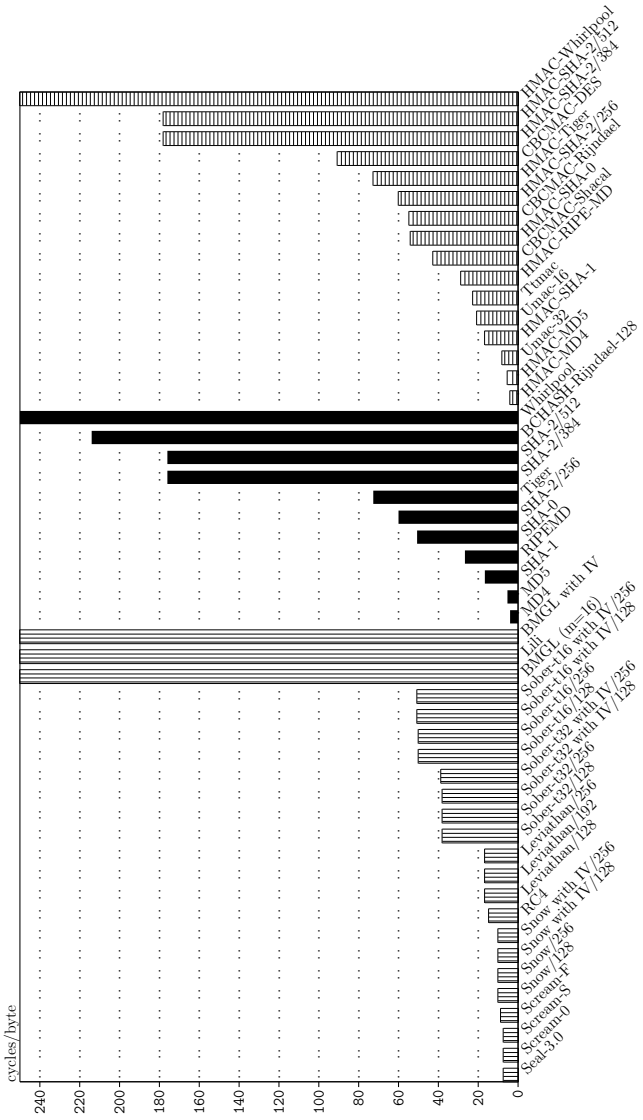


Figure 41: Stream Ciphers, Hash Functions and MACs on 486 (Sorted)
Legend:

stream ciphers, hash function, MAC,

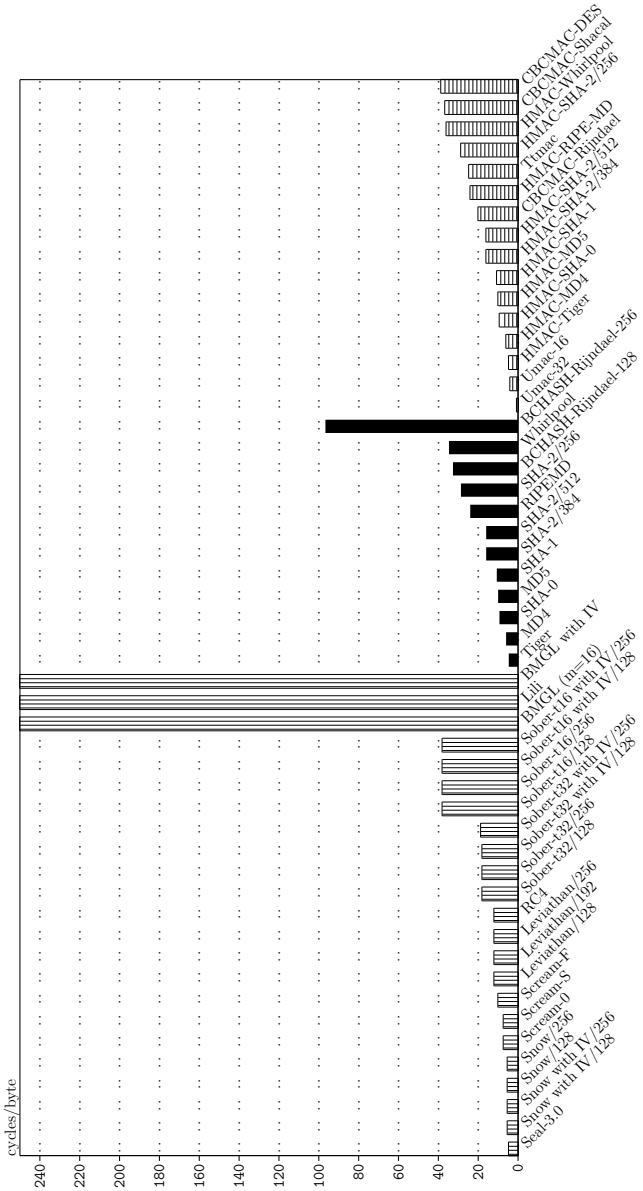


Figure 42: Stream Ciphers, Hash Functions and MACs on Alpha (Sorted)
Legend:

stream ciphers, hash function, MAC,

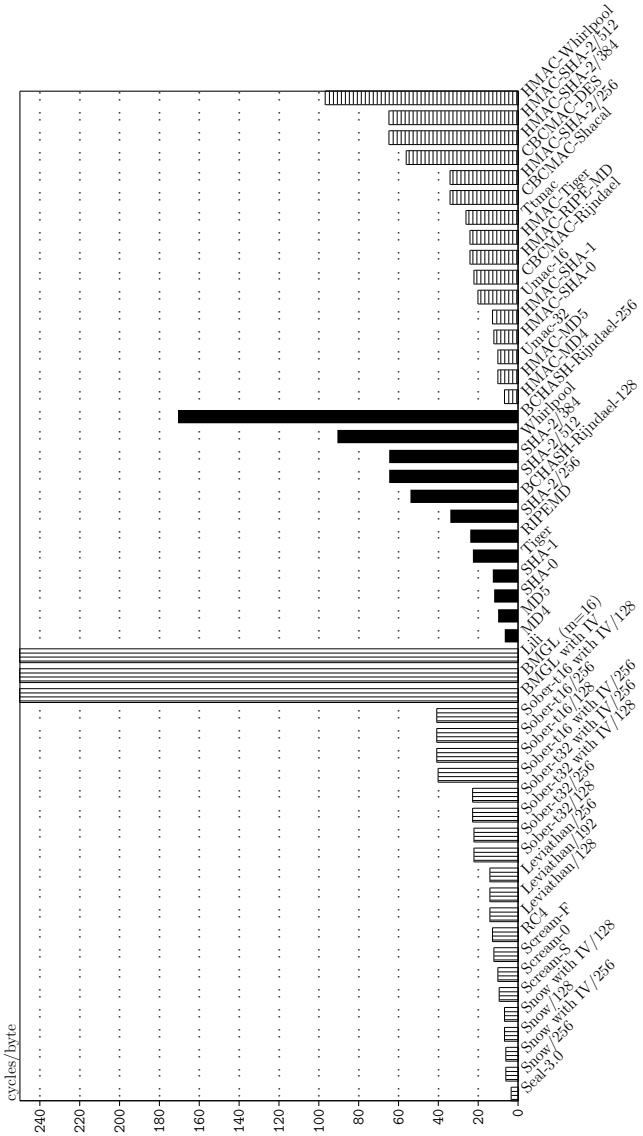


Figure 43: Stream Ciphers, Hash Functions and MACs on Sparc V9 (Sorted)
Legend:

stream ciphers, hash function, MAC,

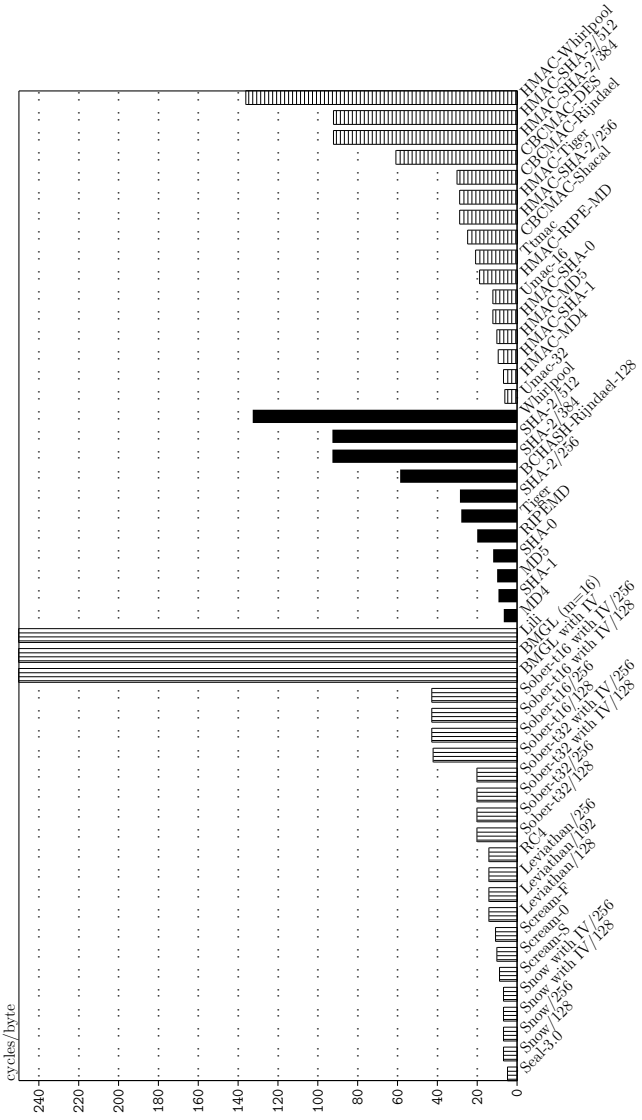


Figure 44: Stream Ciphers, Hash Functions and MACs on Macintosh (Sorted)
Legend:

stream ciphers, hash function, MAC,

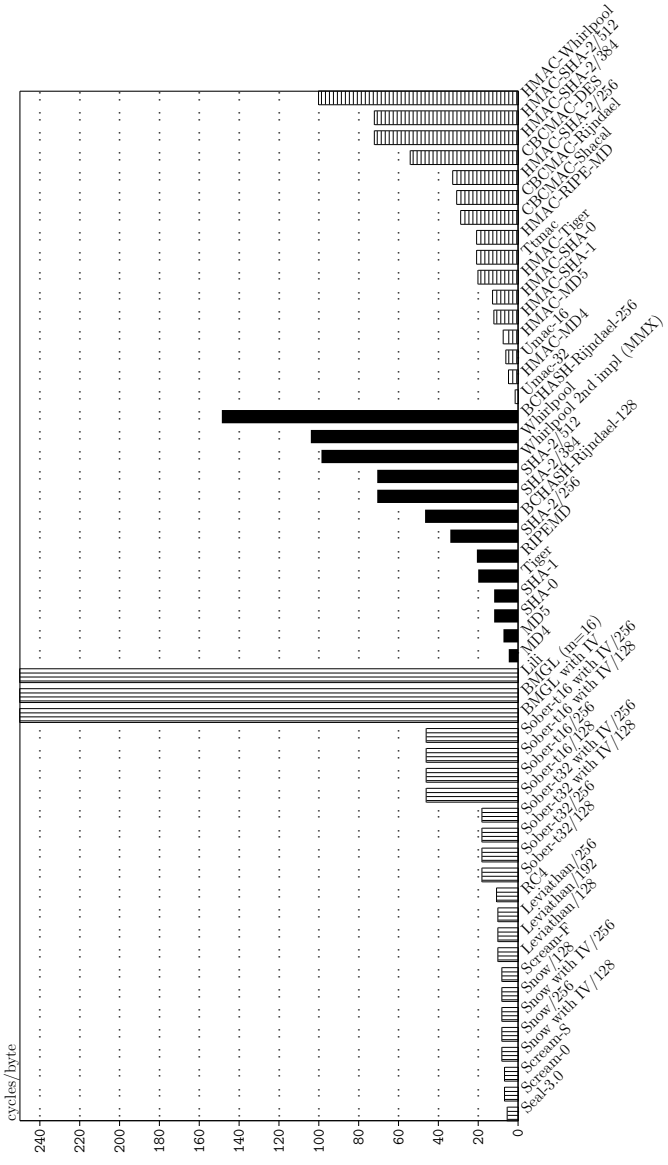


Figure 45: Stream Ciphers, Hash Functions and MACs on AMD (Sorted)
Legend:

stream ciphers, hash function, MAC,