Comparison of 256-bit stream ciphers at the beginning of 2006

Daniel J. Bernstein *

djb@cr.yp.to

Abstract. This paper evaluates and compares several stream ciphers that use 256-bit keys: counter-mode AES, CryptMT, DICING, Dragon, FUBUKI, HC-256, Phelix, Py, Py6, Salsa20, SOSEMANUK, VEST, and YAMB.

1 Introduction

ECRYPT, a consortium of European research organizations, issued a Call for Stream Cipher Primitives in November 2004. A remarkable variety of ciphers were proposed in response by a total of 97 authors spread among Australia, Belgium, Canada, China, Denmark, England, France, Germany, Greece, Israel, Japan, Korea, Macedonia, Norway, Russia, Singapore, Sweden, Switzerland, and the United States.

Evaluating a huge pool of stream ciphers, to understand the merits of each cipher, is not an easy task. This paper simplifies the task by focusing on the relatively small pool of ciphers that allow 256-bit keys. Ciphers limited to 128-bit keys (or 80-bit keys) are ignored. See Section 2 to understand my interest in 256-bit keys.

The ciphers allowing 256-bit keys are CryptMT, DICING, Dragon, FUBUKI, HC-256, Phelix, Py, Py6, Salsa20, SOSEMANUK, VEST, and YAMB. I included 256-bit AES in counter mode as a basis for comparison. Beware that there are unresolved claims of attacks against Py (see [4] and [3]), SOSEMANUK (see [1]), and YAMB (see [5]).

ECRYPT, using measurement tools written by Christophe De Cannière, has published timings for each cipher on several common general-purpose CPUs. The original tools and timings used reference implementations (from the cipher authors) but were subsequently updated for faster implementations (also from the cipher authors). I extended the list of CPUs and then wrote a few extra tools, now available from `http://cr.yp.to/streamciphers.html#timings`, to convert ECRYPT's timings into the tables and graphs shown in Section 3.

Section 4 discusses several other interesting cipher features. For example, some ciphers have "free" built-in message authentication, so users can avoid the cost of computing a separate authenticator. One can and should quantify this benefit by making a separate table of timings for authenticated encryption; I plan to do this in subsequent comparison papers.

* Permanent ID of this document: `eff0eb8eebacda58462948ab97ca48a0`. Date of this document: 2006.01.23. This document is final and may be freely cited.

2 Why use 256-bit keys?

Some readers may wonder why I am not satisfied with 128-bit keys. Haven't I heard that—without massive advances in computer technology—a brute-force attack will never find a 128-bit key? After all, if checking about 2^{20} keys per second requires a CPU costing about 2^6 dollars, then searching 2^{128} keys in a year will cost an inconceivable 2^{89} dollars.

Answer: Even without advances in computer technology, the attacker does not need to spend 2^{89} dollars. Here are three reasons that lower-cost attacks are a threat:

- The attacker can succeed in far fewer than 2^{128} computations. He reaches success probability p after just $2^{128}p$ computations.
- More importantly, each key-checking circuit costs far less than 2^6 dollars, at least in bulk: 2^{10} or more key-checking circuits can fit into a single chip, effectively reducing the attacker's costs by a factor of 2^{10}.
- Even more importantly, if the attacker simultaneously attacks (say) 2^{40} keys, he can effectively reduce his costs by a factor of 2^{40}.

One can counter the third reduction by putting extra randomness into nonces, but putting the same extra randomness into keys is less expensive.

See [2] for a much more detailed discussion of these issues.

3 Speed

Ciphers in the tables in this section are sorted by a low-level feature, namely the number of bytes of state recorded between blocks. At one extreme is HC-256, which expands a key and nonce into a pair of 4096-byte arrays, making several array modifications for each block. At the other extreme is Salsa20, which simply records a key, nonce, and block counter in a 64-byte array, performing computations anew for each block. Most ciphers lie somewhere in the middle.

This ordering is not meant to imply that one extreme is better than the other. A large state has both advantages and disadvantages: it is expensive to set up and maintain, but it is also expensive for the attacker to analyze.

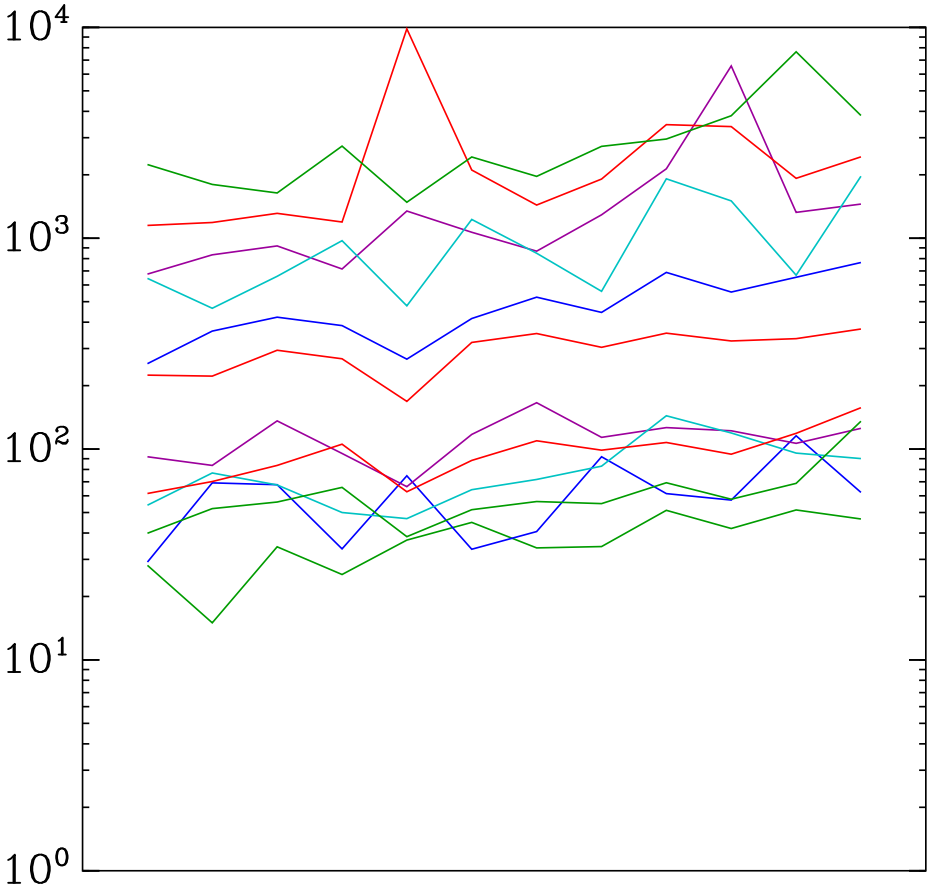
Table entries measure times for key setup, nonce setup, and encryption. All times are expressed as the number of cycles per encrypted byte. Smaller numbers are better here. Lines vary in how much setup they include, how many bytes are encrypted, and which CPU is measured. Bonus for readers using color displays: red means slower than AES; blue means faster than AES; lighter blue means twice as fast as AES; green means three times faster than AES.

FUBUKI has been omitted from the tables in this section. VEST has been omitted from the tables and graphs in this section. The cycle counts for FUBUKI and VEST are too large to be interesting.

	Salsa 20	Phelix	AES	Dragon	YAMB	SOSEMANUK	Py6	Crypt MT	Py	DICING	HC-256
Bytes	64	132	260	284	424	452	1124	3020	4196	4396	8396
Set up key, set up nonce, and encrypt 40-byte packet:											
A64	28.1	29.1	39.9	61.6	644.7	54.2	91.9	675.6	224.3	254.3	2236.5
PPC G4	15.0	69.1	52.2	70.2	465.1	77.0	83.7	834.4	221.9	362.6	1800.6
PM 695	34.4	67.8	56.1	83.7	659.4	67.6	136.0	919.0	294.1	422.1	1638.4
Athlon	25.4	33.6	65.8	105.5	974.3	50.0	95.3	714.5	268.1	385.0	2733.0
HP	37.0	74.7	38.4	62.7	478.0	46.8	66.3	1345.9	168.4	266.9	1481.0
P4 f41	44.9	33.5	51.6	88.4	1227.6	64.2	117.3	1066.9	320.6	416.2	2429.0
P3 68a	34.0	40.6	56.4	109.5	849.0	71.8	166.0	868.8	353.4	525.5	1964.3
SPARC	34.5	92.0	55.1	98.8	560.0	83.0	113.7	1292.1	303.7	444.7	2728.8
P4 f29	51.2	61.5	69.2	107.4	1914.6	143.9	126.4	2134.2	354.8	688.6	2953.2
P4 f12	42.0	57.3	57.8	94.5	1504.0	119.2	122.2	6560.6	325.6	555.3	3811.1
Alpha	51.4	115.7	68.8	118.7	667.8	95.7	106.5	1327.2	334.1		7660.2
P1 52c	46.6	62.3	135.5	157.2	1967.1	90.1	125.4	1452.1	371.0	766.7	3822.1
Set up nonce and encrypt 40-byte packet:											
A64	26.6	20.9	32.2	58.6	639.7	24.3	62.5	673.5	155.0	252.6	2234.7
PPC G4	13.6	52.3	44.6	66.9	459.8	31.9	62.4	832.3	169.2	361.2	1798.4
PM 695	32.5	53.8	47.3	80.1	656.0	36.0	94.0	917.2	168.8	420.0	1636.8
Athlon	23.4	24.9	56.7	99.9	970.2	27.9	65.7	712.2	196.6	382.6	2730.6
HP	35.0	59.9	31.9	58.1	473.5	29.7	42.4	1343.3	111.0	265.4	1478.8
P4 f41	42.1	23.6	43.6	83.0	1221.5	39.3	83.3	1064.7	230.3	414.7	2427.0
P3 68a	32.6	30.0	48.0	103.1	845.2	38.3	97.2	867.0	164.7	524.0	1963.1
SPARC	33.0	67.4	40.9	91.4	554.0	43.4	76.4	1288.7	227.4	442.9	2726.0
P4 f29	48.5	43.9	55.9	98.8	1902.8	51.2	84.3	2131.6	245.6	686.0	2950.6
P4 f12	39.6	39.6	46.0	86.3	1497.3	46.1	90.1	6556.6	256.4	552.6	3808.6
Alpha	49.7	83.6	57.7	109.6	661.7	50.7	70.3	1322.3	237.2		7647.3
P1 52c	42.5	46.0	113.2	148.6	1959.9	54.2	76.0	1449.3	252.8	763.6	3818.9
Set up nonce and encrypt 576-byte packet:											
A64	9.2	6.1	25.4	24.0	62.0	8.3	10.0	60.1	16.5	27.4	159.3
PPC G4	4.4	17.1	35.0	28.9	44.7	10.3	9.2	74.6	16.6	38.9	130.5
PM 695	12.1	14.9	35.1	27.8	64.9	9.6	9.1	74.8	14.1	41.5	117.5
Athlon	10.7	7.3	44.7	37.3	90.0	8.8	10.4	64.6	19.5	39.5	194.7
HP	11.6	16.4	22.5	26.0	47.5	8.8	6.6	113.7	11.3	28.4	107.2
P4 f41	14.3	7.0	33.5	32.6	106.7	12.4	9.3	94.6	19.0	42.1	171.6
P3 68a	14.5	9.0	37.7	35.4	81.7	12.0	9.8	73.4	14.5	50.7	142.0
SPARC	14.5	21.0	31.8	46.2	54.7	14.0	11.4	110.8	22.6	45.4	197.4
P4 f29	19.8	12.6	40.2	34.6	165.7	13.5	9.0	164.3	20.0	72.5	206.2
P4 f12	17.3	12.0	37.2	31.0	143.4	12.8	11.7	471.5	24.0	66.8	270.0
Alpha	22.6	28.3	43.2	52.3	64.4	16.9	11.0	128.0	23.2		549.5
P1 52c	19.8	14.2	85.7	60.3	181.5	17.3	17.4	136.2	28.3	82.4	275.4

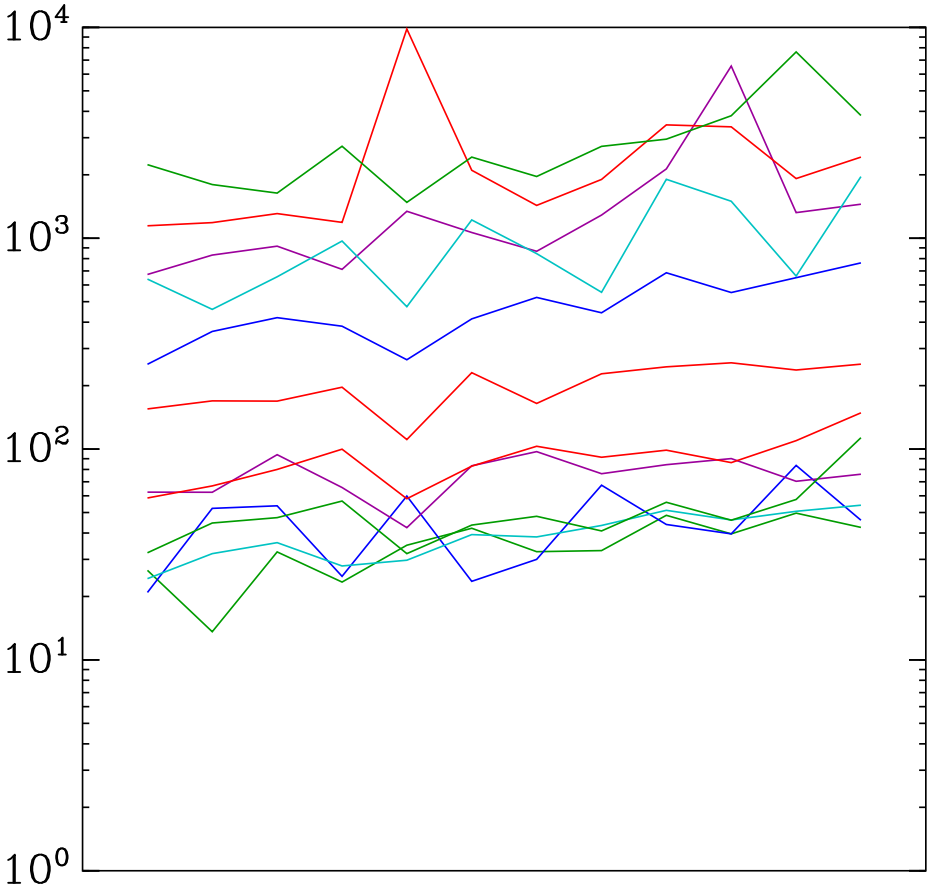
	Salsa 20	Phelix	AES	Dragon	YAMB	SOSEMANUK	Py6	Crypt MT	Py	DICING	HC-256
Bytes	64	132	260	284	424	452	1124	3020	4196	4396	8396
Set up nonce and encrypt 1500-byte packet:											
A64	9.4	5.4	25.4	22.3	35.5	7.3	7.7	28.0	10.1	17.2	64.2
PPC G4	4.5	15.5	35.0	27.1	25.6	8.9	6.8	38.4	9.7	24.4	54.2
PM 695	12.3	13.1	35.0	25.4	37.8	8.1	5.2	34.6	7.0	24.4	48.0
Athlon	10.9	6.5	44.7	34.2	49.5	7.5	7.8	32.1	11.2	24.1	78.5
HP	12.0	14.4	22.5	24.6	28.0	7.4	5.0	59.7	6.7	17.7	44.6
P4 f41	14.7	6.0	33.2	30.3	49.4	10.7	5.9	44.2	9.8	25.6	68.9
P3 68a	14.8	8.0	37.6	32.3	46.7	10.2	5.8	35.6	7.6	29.3	58.6
SPARC	14.9	18.9	31.8	44.0	31.8	12.2	8.5	54.7	13.2	27.4	81.6
P4 f29	20.0	11.0	39.5	32.1	79.2	10.8	5.5	72.4	10.3	41.7	82.7
P4 f12	20.1	10.9	37.2	28.9	80.8	10.6	8.6	200.5	13.6	36.8	106.7
Alpha	23.2	26.0	43.2	49.6	36.9	15.0	8.4	70.7	13.1		222.7
P1 52c	20.1	12.7	89.4	51.2	95.6	15.2	15.7	65.3	20.3	51.3	113.1
Encrypt one long stream:											
A64	8.9	4.9	25.2	8.1	18.9	4.4	3.9	9.3	4.0	10.8	4.4
PPC G4	4.2	9.6	34.8	8.4	13.7	6.2	5.3	16.4	5.4	15.2	6.2
PM 695	11.8	12.1	34.7	12.9	20.8	5.2	2.9	10.2	2.7	13.6	4.4
Athlon	10.5	6.0	44.4	13.4	24.3	5.6	4.4	13.1	5.0	14.3	5.7
HP	11.4	23.0	22.3	6.2	15.3	6.1	4.3	24.6	4.2	10.9	5.3
P4 f41	13.9	5.6	33.1	12.3	16.5	5.7	3.8	16.1	3.7	14.7	5.0
P3 68a	14.3	7.5	37.4	14.3	24.9	6.2	3.3	12.6	3.2	15.7	6.5
SPARC	14.3	16.9	31.6	8.8	17.6	8.3	6.5	20.7	6.7	16.2	9.0
P4 f29	17.0	10.1	39.3	12.9	29.2	6.5	3.5	15.3	3.8	23.5	4.8
P4 f12	17.0	10.1	36.8	12.9	37.9	6.2	4.5	16.1	4.8	21.7	5.0
Alpha	22.5	19.9	42.9	12.7	19.7	13.9	6.7	38.0	6.9		18.6
P1 52c	20.8	12.1	88.4	26.0	43.1	11.0	9.4	25.0	10.8	30.5	11.6
Encrypt many parallel streams in 256-byte blocks:											
A64	10.2	7.2	27.6	10.4	23.6	5.7	12.0	12.7	25.0	24.5	18.2
PPC G4	4.9	12.3	37.7	10.1	17.2	7.2	13.4	23.7	31.3	35.6	27.6
PM 695	12.8	14.5	37.7	15.1	25.5	6.3	10.7	17.1	26.7	31.1	21.3
Athlon	12.4	9.5	48.6	16.8	31.2	7.4	16.8	26.5	41.2	41.9	34.7
HP	12.1	24.7	24.9	8.1	18.4	7.3	8.3	28.8	14.7	23.1	17.8
P4 f41	16.4	9.3	37.1	16.2	24.0	7.5	12.8	23.8	26.4	38.4	28.6
P3 68a	15.8	11.6	43.3	19.9	37.6	7.8	25.3	41.1	77.3	58.4	55.2
SPARC	15.4	20.0	36.1	12.1	23.0	10.2	14.6	32.9	21.6	66.6	57.2
P4 f29	19.4	14.6	44.2	18.4	42.8	8.8	12.3	25.0	27.2	48.2	29.8
P4 f12	19.2	14.2	42.0	17.6	45.6	8.1	14.8	24.4	28.2	43.8	27.1
Alpha	23.4	22.4	49.2	15.5	24.9	15.0	15.1	38.4	36.0		50.0
P1 52c	21.3	14.7	85.8	27.2	47.1	12.1	18.0	29.3	39.5	50.3	33.5

Set up key, set up nonce, and encrypt 40-byte packet



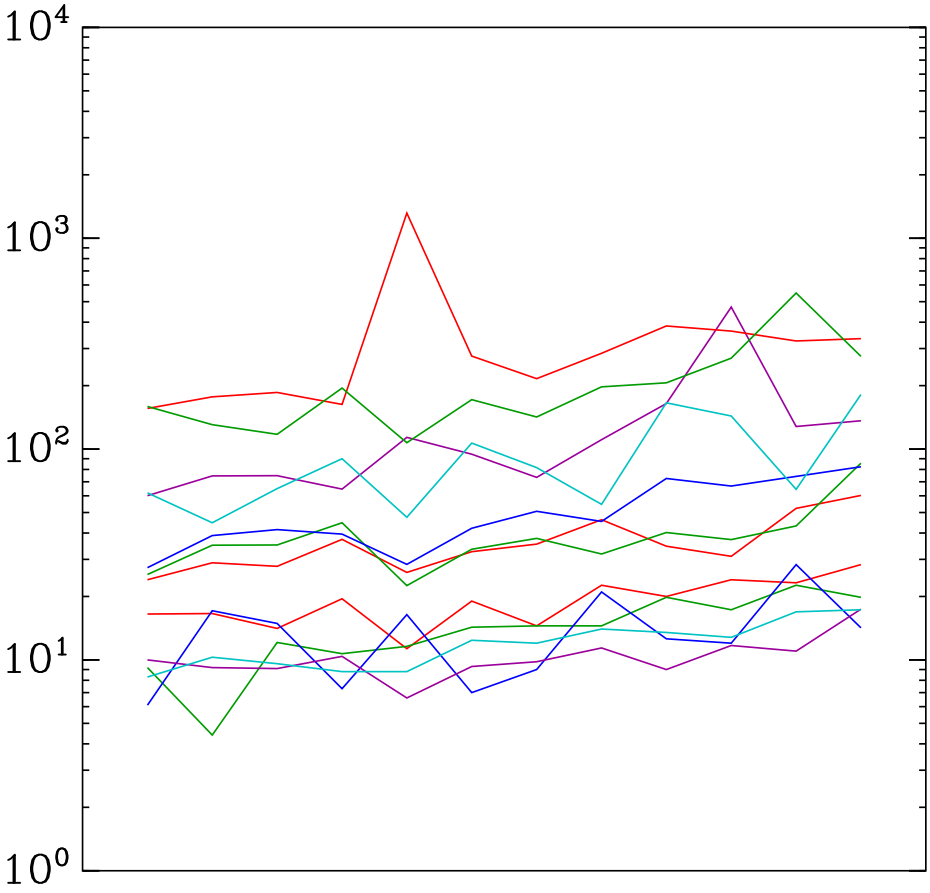
A64	PPC G4	PM 695	Athl	HP	P4 f41	P3 68a	SP	P4 f29	P4 f12	Alpha	P1 52c
HC	HC	HC	HC	FUB	HC	HC	HC	FUB	Cry		HC
FUB	FUB	FUB	FUB	HC	FUB	FUB	FUB	HC	HC	HC	FUB
Cry	Cry	Cry	YA	Cry	YA	Cry	Cry	Cry	FUB	FUB	YA
YA	YA	YA	Cry	YA	Cry	YA	YA	YA	YA	Cry	Cry
DIC	DIC	DIC	DIC	DIC	DIC	DIC	DIC	DIC	DIC	YA	DIC
Py	Py	Py	Py	Py	Py	Py	Py	Py	Py	Py	Py
Py6	Py6	Py6	Dra	Phe	Py6	Py6	Py6	SOS	Py6	Dra	Dra
Dra	SOS	Dra	Py6	Py6	Dra	Dra	Dra	Py6	SOS	Phe	AES
SOS	Dra	Phe	AES	Dra	SOS	SOS	Phe	Dra	Dra	Py6	Py6
AES	Phe	SOS	SOS	SOS	AES	AES	SOS	AES	AES	SOS	SOS
Phe	AES	AES	Phe	AES	Sal	Phe	AES	Phe	Phe	AES	Phe
Sal	Sal	Sal	Sal	Sal	Phe	Sal	Sal	Sal	Sal	Sal	Sal

Set up nonce and encrypt 40-byte packet



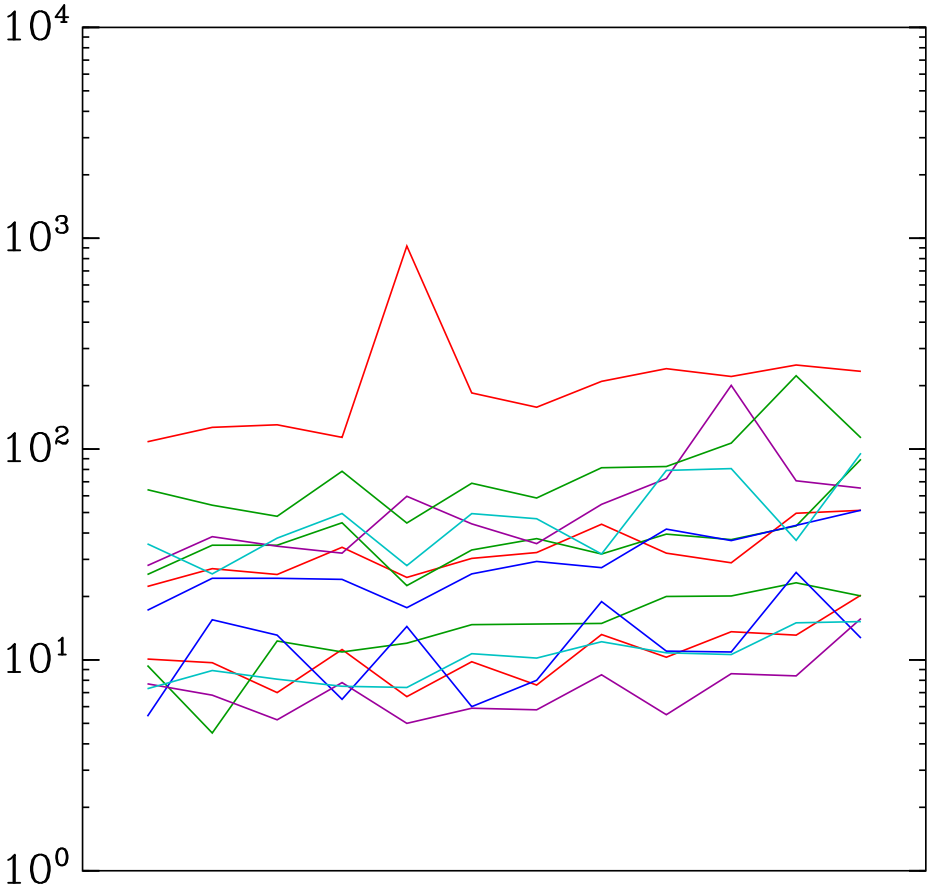
HC	HC	HC	HC	FUB	HC	HC	HC	FUB	Cry		HC
FUB	FUB	FUB	FUB	HC	FUB	FUB	FUB	HC	HC	HC	FUB
Cry	Cry	Cry	YA	Cry	YA	Cry	Cry	Cry	FUB	FUB	YA
YA	YA	YA	Cry	YA	Cry	YA	YA	YA	YA	Cry	Cry
DIC	DIC	DIC	DIC	DIC	DIC	DIC	DIC	DIC	DIC	YA	DIC
Py	Py	Py	Py	Py	Py	Py	Py	Py	Py	Py	Py
Py6	Dra	Py6	Dra	Phe	Py6	Dra	Dra	Dra	Py6	Dra	Dra
Dra	Py6	Dra	Py6	Dra	Dra	Py6	Py6	Py6	Dra	Phe	AES
AES	Phe	Phe	AES	Py6	AES	AES	Phe	AES	SOS	Py6	Py6
Sal	AES	AES	SOS	Sal	Sal	SOS	SOS	SOS	AES	AES	SOS
SOS	SOS	SOS	Phe	AES	SOS	Sal	AES	Sal	Sal	SOS	Phe
Phe	Sal	Sal	Sal	SOS	Phe	Phe	Sal	Phe	Phe	Sal	Sal
A64	PPC G4	PM 695	Athl	HP	P4 f41	P3 68a	SP	P4 f29	P4 f12	Alpha	P1 52c

Set up nonce and encrypt 576-byte packet



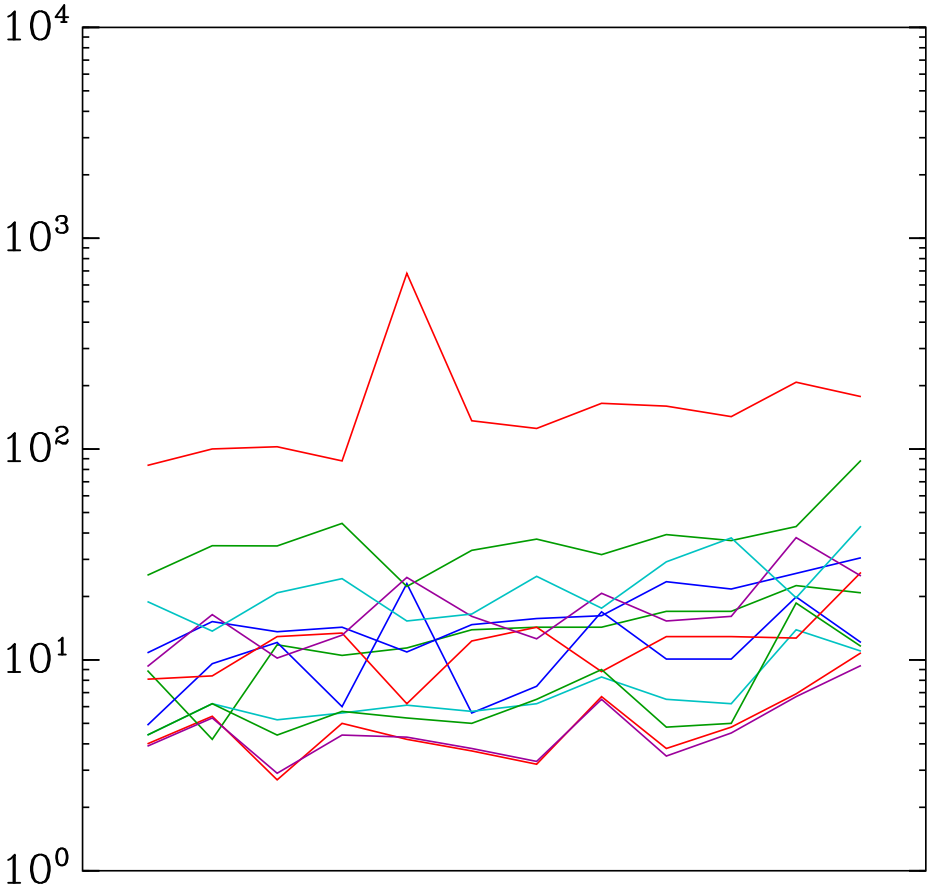
HC	FUB	FUB	HC	FUB	FUB	FUB	FUB	FUB	Cry		FUB
FUB	HC	HC	FUB	Cry	HC	HC	HC	HC	FUB	HC	HC
YA	Cry	Cry	YA	HC	YA	YA	Cry	YA	HC	FUB	YA
Cry	YA	YA	Cry	YA	Cry	Cry	YA	Cry	YA	Cry	Cry
DIC	DIC	DIC	AES	DIC	DIC	DIC	Dra	DIC	DIC	YA	AES
AES	AES	AES	DIC	Dra	AES	AES	DIC	AES	AES	Dra	DIC
Dra	Dra	Dra	Dra	AES	Dra	Dra	AES	Dra	Dra	AES	Dra
Py	Phe	Phe	Py	Phe	Py	Sal	Py	Py	Py	Phe	Py
Py6	Py	Py	Sal	Sal	Sal	Py	Phe	Sal	Sal	Py	Sal
Sal	SOS	Sal	Py6	Py	SOS	SOS	Sal	SOS	SOS	Sal	Py6
SOS	Py6	SOS	SOS	SOS	Py6	Py6	SOS	Phe	Phe	SOS	SOS
Phe	Sal	Py6	Phe	Py6	Phe	Phe	Py6	Py6	Py6	Py6	Phe
A64	PPC G4	PM 695	Athl	HP	P4 f41	P3 68a	SP	P4 f29	P4 f12	Alpha	P1 52c

Set up nonce and encrypt 1500-byte packet



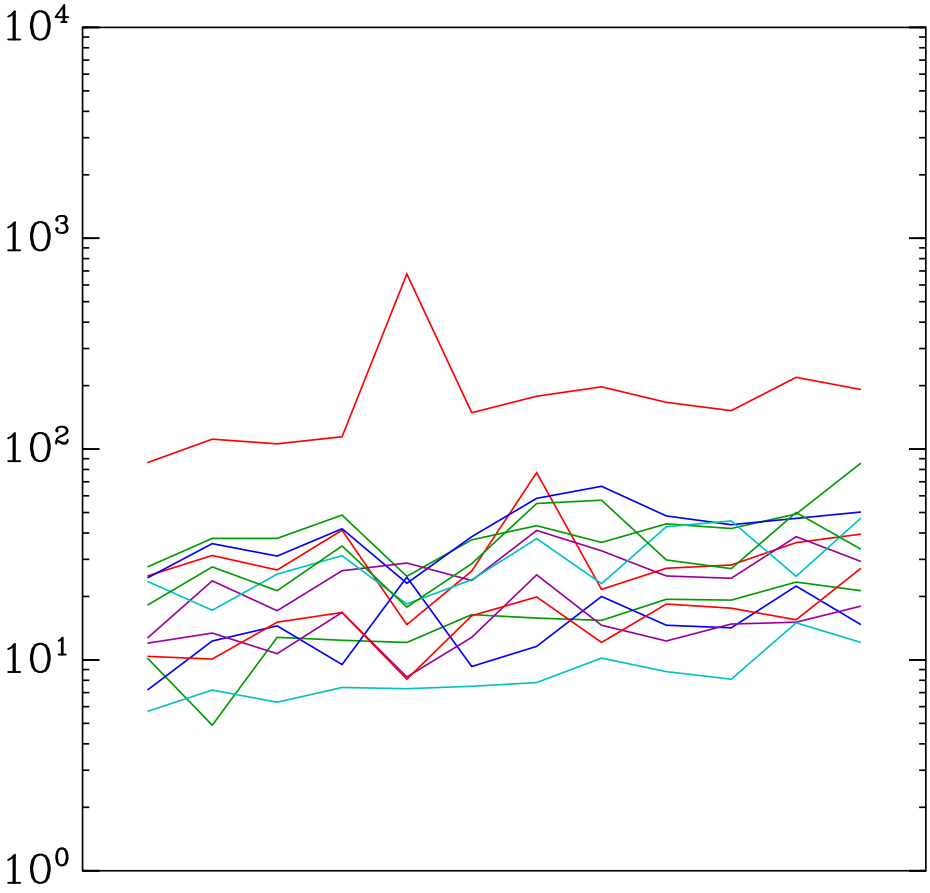
FUB	FUB	FUB	FUB	FUB	FUB	FUB	FUB	FUB	FUB		FUB
HC	HC	HC	HC	Cry	HC	HC	HC	HC	Cry	FUB	HC
YA	Cry	YA	YA	HC	YA	YA	Cry	YA	HC	HC	YA
Cry	AES	AES	AES	YA	Cry	AES	Dra	Cry	YA	Cry	AES
AES	Dra	Cry	Dra	Dra	AES	Cry	YA	DIC	AES	Dra	Cry
Dra	YA	Dra	Cry	AES	Dra	Dra	AES	AES	DIC	AES	DIC
DIC	DIC	DIC	DIC	DIC	DIC	DIC	DIC	Dra	Dra	YA	Dra
Py	Phe	Phe	Py	Phe	Sal	Sal	Phe	Sal	Sal	Phe	Py
Sal	Py	Sal	Sal	Sal	SOS	SOS	Sal	Phe	Py	Sal	Sal
Py6	SOS	SOS	Py6	SOS	Py	Phe	Py	SOS	Phe	SOS	Py6
SOS	Py6	Py	SOS	Py	Phe	Py	SOS	Py	SOS	Py	SOS
Phe	Sal	Py6	Phe	Py6	Py6	Py6	Py6	Py6	Py6	Py6	Phe
A64	PPC G4	PM 695	Athl	HP	P4 f41	P3 68a	SP	P4 f29	P4 f12	Alpha	P1 52c

Encrypt one long stream



FUB	FUB	FUB	FUB	FUB	FUB	FUB	FUB	FUB	FUB		FUB
AES	AES	AES	AES	Cry	AES	AES	AES	AES	YA	FUB	AES
YA	Cry	YA	YA	Phe	YA	YA	Cry	YA	AES	AES	YA
DIC	DIC	DIC	DIC	AES	Cry	DIC	YA	DIC	DIC	Cry	DIC
Cry	YA	Dra	Dra	YA	DIC	Sal	Phe	Sal	Sal	Sal	Dra
Sal	Phe	Phe	Cry	Sal	Sal	Dra	DIC	Cry	Cry	Phe	Cry
Dra	Dra	Sal	Sal	DIC	Dra	Cry	Sal	Dra	Dra	YA	Sal
Phe	SOS	Cry	Phe	Dra	SOS	Phe	HC	Phe	Phe	HC	Phe
SOS	HC	SOS	HC	SOS	Phe	HC	Dra	SOS	SOS	SOS	HC
HC	Py	HC	SOS	HC	HC	SOS	SOS	HC	HC	Dra	SOS
Py	Py6	Py6	Py	Py6	Py6	Py6	Py	Py	Py	Py	Py
Py6	Sal	Py	Py6	Py	Py	Py	Py6	Py6	Py6	Py6	Py6
A64	PPC G4	PM 695	Athl	HP	P4 f41	P3 68a	SP	P4 f29	P4 f12	Alpha	P1 52c

Encrypt many parallel streams in 256-byte blocks



A64	PPC G4	PM 695	Athl	HP	P4 f41	P3 68a	SP	P4 f29	P4 f12	Alpha	P1 52c
FUB	FUB	FUB	FUB	FUB	FUB	FUB	FUB	FUB	FUB		FUB
AES	AES	AES	AES	Cry	DIC	Py	DIC	DIC	YA	FUB	AES
Py	DIC	DIC	DIC	AES	AES	DIC	HC	AES	DIC	HC	DIC
DIC	Py	Py	Py	Phe	HC	HC	AES	YA	AES	AES	YA
YA	HC	YA	HC	DIC	Py	AES	Cry	HC	Py	Cry	Py
HC	Cry	HC	YA	YA	YA	Cry	YA	Py	HC	Py	HC
Cry	YA	Cry	Cry	HC	Cry	YA	Py	Cry	Cry	YA	Cry
Py6	Py6	Dra	Py6	Py	Sal	Py6	Phe	Sal	Sal	Sal	Dra
Dra	Phe	Phe	Dra	Sal	Dra	Dra	Sal	Dra	Dra	Phe	Sal
Sal	Dra	Sal	Sal	Py6	Py6	Sal	Py6	Phe	Py6	Dra	Py6
Phe	SOS	Py6	Phe	Dra	Phe	Phe	Dra	Py6	Phe	Py6	Phe
SOS	Sal	SOS	SOS	SOS	SOS	SOS	SOS	SOS	SOS	SOS	SOS

Notes on the timings

The tables and graphs use the following representative set of 12 machines, all with version 156 (2006.01.16) of ECRYPT's timing suite except where otherwise noted:

- A64: 2000MHz (one of two CPU cores) AMD Athlon 64 X2 (CPU identifier 15/43/1) named cph (gcc 4.0.2, Ubuntu 5.10).
- PPC G4: 533MHz (one of two CPUs) Motorola PowerPC G4 7410 named gggg (gcc 4.0.2, Ubuntu 5.10).
- PM 695: 1300MHz Intel Pentium M (695) named whisper (Fedora).
- Athlon: 900MHz AMD Athlon (622) named thoth (gcc 4.0.2, Ubuntu 5.10).
- HP PA: 440MHz (one of two CPUs) HP 9000/785 J5000 named hp400 (HP/UX).
- P4 f41: 3000MHz Intel Pentium 4 (f41) named pentium4b, timings collected by Christophe De Cannière.
- P3 68a: 1000MHz (one of two CPUs) Intel Pentium III (68a) named neumann (gcc 2.95.4 and gcc 3.0.4, Debian).
- SPARC: 900MHz Sun UltraSPARC III named wessel (SunOS 5.9).
- P4 f29: 2800MHz (one of two CPUs) Intel Pentium 4 (f29) named rzitsc (gcc 3.2.3, Red Hat).
- P4 f12: 1900MHz Intel Pentium 4 (f12) named fireball (gcc 4.0.2, Ubuntu 5.10).
- Alpha: 400MHz DEC Alpha EV5.6 21164A named alpha, using version 140 (2005.12.21), timings collected by Christophe De Cannière.
- P1 52c: 133MHz Intel Pentium (52c) named cruncher (gcc 4.0.2, Ubuntu 5.10).

The machines are sorted by the geometric average of all cipher cycle counts. This sorting accounts for the overall left-to-right upward trend in the graphs on previous pages.

See my web page `http://cr.yp.to/streamciphers.html#timings` for more comprehensive data. The web page includes speed reports for 24 machines; I'd also like to include timings for 8-bit CPUs and for ASICs. I will continue to update the web page as I receive newer information.

The graphs use cycles per byte, with a logarithmic scale, for the vertical axis. The labels below the graphs list ciphers in speed order. Consider, for example, the first graph: "Set up key, set up nonce, and encrypt 40-byte packet." The first column of the graph is labelled, from top to bottom, HC FUB Cry YA DIC Py Py6 Dra SOS AES Phe Sal A64. This column shows that, for setup and 40-byte encryption on an Athlon 64 (A64), HC-256 (HC) takes the most cycles per byte, and Salsa20 (Sal) takes the fewest cycles per byte. The graph shows that HC-256 takes about $2 \cdot 10^3$ cycles per byte while Salsa20 takes about $3 \cdot 10^1$ cycles per byte. The earlier table shows that HC-256 takes 2236.5 cycles per byte (i.e., 89460 cycles for 40 bytes) while Salsa20 takes 28.1 cycles per byte (i.e., 1124 cycles for 40 bytes).

4 Additional features

Bonus for readers using color displays: in this section, blue means an advantage compared to AES, and red means a disadvantage compared to AES.

AES in counter mode

Encryption. Unpatented. Variable time. 256-bit security conjecture. Security margin: has faster reduced-round versions; Ferguson et al. reported an attack on 7 out of 14 rounds; as far as I know, all claimed attacks on 8 rounds actually have worse price-performance ratio than brute-force search; there are no public claims of attacks on 9 rounds.

CryptMT

Encryption. Patented. Constant time. 256-bit security conjecture. No explicit security margin.

DICING

Encryption. Unpatented. Variable time. 256-bit security conjecture. No explicit security margin.

Dragon

Encryption. Unpatented. Variable time. 256-bit security conjecture. No explicit security margin.

FUBUKI

Encryption. Patented. Variable time. 256-bit security conjecture. No explicit security margin.

HC-256

Encryption. Unpatented. Variable time. 256-bit security conjecture. No explicit security margin.

Phelix

Authenticated encryption. Unpatented. Constant time. 128-bit security conjecture. No explicit security margin.

Py

Encryption. Unpatented. Variable time. 256-bit security conjecture. No explicit security margin. Attacks: Sekar, Paul, and Preneel in [4] reported an attack on Py using 2^{88} output bytes and comparable time. Crowley in [3] reduced 2^{88} to 2^{72}. The authors have not yet responded.

Py6

Encryption. Unpatented. Variable time. 256-bit security conjecture. No explicit security margin. Attacks: The attacks on Py by Sekar et al. can, presumably, be extended to Py6.

Salsa20

Encryption. Unpatented. Constant time. 256-bit security conjecture. Security margin: has faster reduced-round versions; Crowley reported an attack on 5 out of 20 rounds; there are no public claims of attacks on 6 rounds.

SOSEMANUK

Encryption. Unpatented. Variable time. 128-bit security conjecture. No explicit security margin. Attacks: Ahmadi, Eghlidos, and Khazaei in [1] reported an attack on SOSEMANUK using 2^{226} simple operations—but this doesn't disprove the original 128-bit security conjecture for SOSEMANUK. The authors have not yet responded.

VEST

Authenticated encryption. Patented. Variable time. 256-bit security conjecture. No explicit security margin.

YAMB

Encryption. Unpatented. Variable time. 256-bit security conjecture. No explicit security margin. Attacks: Wu and Preneel in [5] reported an attack on YAMB requiring 2^{58} output blocks and comparable time. There has been no response from the authors after six months.

5 Recommendations

Py, Py6, SOSEMANUK, and YAMB don't appear to provide 256-bit security. Unless there's a dispute regarding the attacks on these ciphers, they should be eliminated from consideration, at least as competition for 256-bit AES.

FUBUKI has no apparent advantages over AES and is several times slower. Unless there are dramatic speedups in the FUBUKI software, FUBUKI should be eliminated from consideration.

VEST is painfully slow in software but is claimed to provide considerably better performance in hardware. I haven't seen a careful evaluation of hardware performance, so I won't make any recommendations now regarding VEST.

The remaining 256-bit stream ciphers are CryptMT, DICING, Dragon, HC-256, Phelix, and Salsa20. Each of these ciphers provides better performance than AES for long streams, and some of them provide better performance than AES in other situations.

I recommend keeping all six ciphers—CryptMT, DICING, Dragon, HC-256, Phelix, and Salsa20—under consideration. One might be tempted to say, e.g., "CryptMT is practically always slower than Phelix and should be eliminated," but this will sound quite silly in retrospect if Phelix turns out to be breakable. The initial stream-cipher submission deadline was only eight months ago; the Py and SOSEMANUK attacks were published only a month ago; obviously we need more time for cryptanalysis.

References

1. Hadi Ahmadi, Taraneh Eghlidos, Shahram Khazaei, *Improved guess and determine attack on SOSEMANUK*, eSTREAM, ECRYPT Stream Cipher Project, Report 2005/085 (2005). URL: `http://www.ecrypt.eu.org/stream`. Citations in this paper: §1, §4.
2. Daniel J. Bernstein, *Understanding brute force* (2005). URL: `http://cr.yp.to/papers.html#bruteforce`. ID `73e92f5b71793b498288efe81fe55dee`. Citations in this paper: §2.
3. Paul Crowley, *Improved cryptanalysis of Py* (2006). URL: `http://www.ciphergoth.org/crypto/py/`. Citations in this paper: §1, §4.
4. Gautham Sekar, Souradyuti Paul, Bart Preneel, *Distinguishing attacks on the stream cipher Py*, eSTREAM, ECRYPT Stream Cipher Project, Report 2005/081 (2005). URL: `http://www.ecrypt.eu.org/stream`. Citations in this paper: §1, §4.
5. Hongjun Wu, Bart Preneel, *Distinguishing attack on stream cipher Yamb*, eSTREAM, ECRYPT Stream Cipher Project, Report 2005/043 (2005). URL: `http://www.ecrypt.eu.org/stream`. Citations in this paper: §1, §4.