Comparison of 256-bit stream ciphers

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. In response, a remarkable variety of stream ciphers were proposed by a total of 97 authors spread among Australia, Belgium, Canada, China, Denmark, England, France, Germany, Greece, Israel, Japan, Korea, Macedonia, Norway, Russia, Sweden, Singapore, 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 that allow 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 [3] and [2]) and YAMB (see [4]).

ECRYPT, using measurement tools written by Christophe De Cannière, has published timings for each cipher on several common general-purpose CPUs. Beware that the original tools and timings used the cipher authors' reference implementations; many of those implementations have not yet been replaced by faster implementations. I extended the list of CPUs and then wrote a few extra tools, to appear on `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

* Permanent ID of this document: `eff0eb8eebacda58462948ab97ca48a0`. Date of this document: 2005.12.23. This is a preliminary version meant to announce ideas; it will be replaced by a final version meant to record the ideas for posterity. There may be big changes before the final version. Future readers should not be forced to look at preliminary versions, unless they want to check historical credits; if you cite a preliminary version, please repeat all ideas that you are using from it, so that the reader can skip it.

the cost of computing a separate authenticator; in subsequent versions of this paper I plan to quantify this benefit by making a separate table of timings for authenticated encryption.

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 [1] 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.

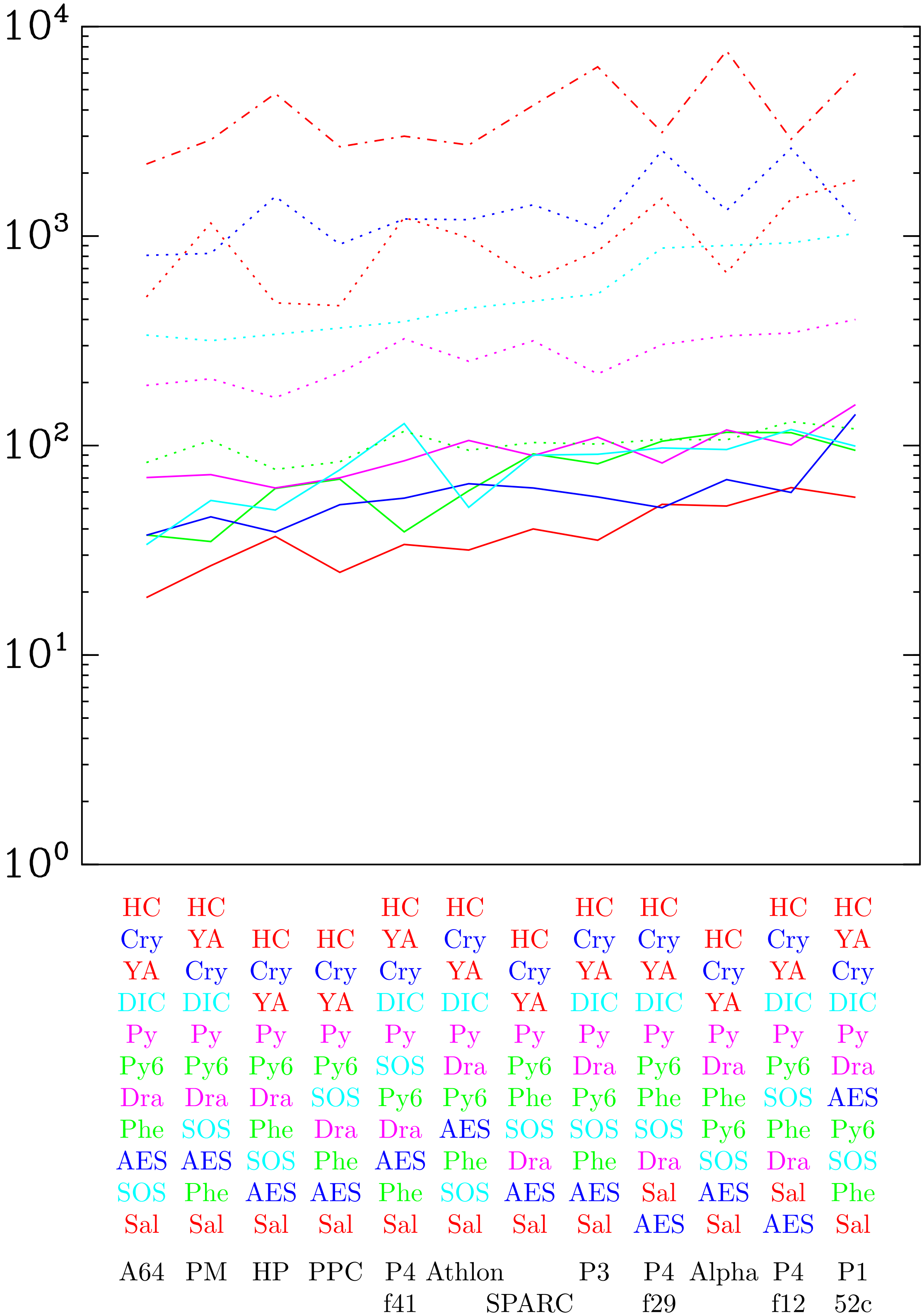
Table entries measure times for key setup, nonce setup, and encryption. All times are expressed as the number of cycles per encrypted byte. Smaller numbers are better here. Lines vary in how much setup they include, how many bytes are encrypted, and which CPU is measured. Red means slower than AES; blue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

Fubuki and VEST have been omitted from the tables and graphs in this section. All available timings for Fubuki and VEST are over 100 cycles per byte.

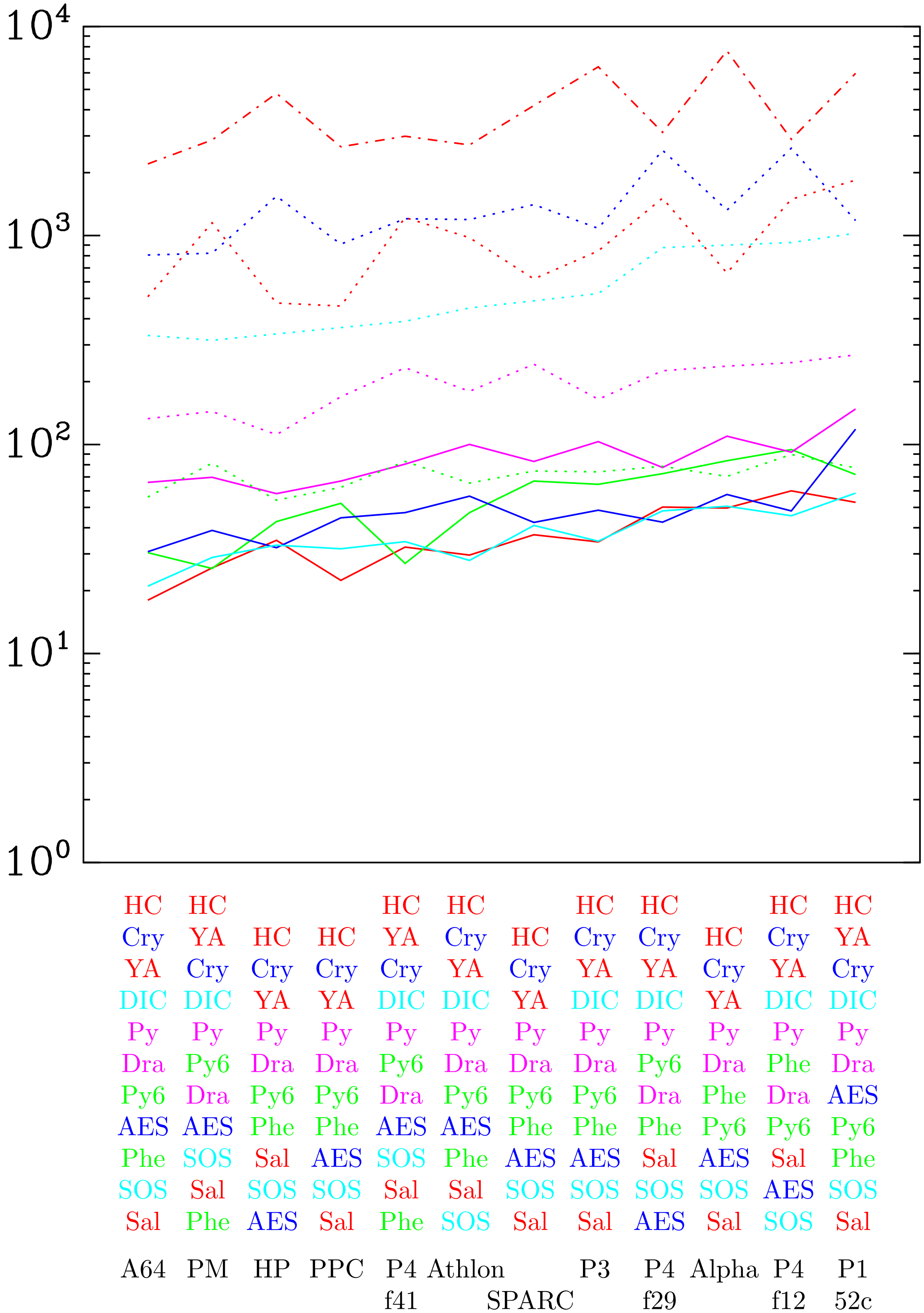
	Salsa20	Phelix	AES	Dragon	SOSEMANUK	YAMB	Py6	CryptMT	Py	DICING	HC-256
Bytes	64	132	260	284	452	612	1124	3024	4196	4396	8268
Set up key, set up nonce, and encrypt 40-byte packet:											
A64	18.8	37.4	37.3	70.3	33.6	512.6	83.0	809.2	193.6	336.8	2206.6
PM	26.7	34.8	45.7	72.6	54.7	1153.1	105.7	826.2	208.5	316.5	2881.8
HP	36.8	62.4	38.6	62.7	49.2	480.2	76.9	1546.1	169.2		4791.2
PPC	24.8	69.1	52.2	70.2	76.6	465.1	83.7	912.8	222.1		2670.8
P4 f41	33.7	38.7	56.1	84.6	127.3	1226.4	117.0	1206.3	323.3	390.5	2998.5
Athlon	31.7	60.8	65.7	105.8	50.7	981.0	94.9	1196.3	252.3	452.4	2721.1
SPARC	40.0	91.2	62.7	89.5	90.0	623.8	103.4	1410.3	316.1		4203.9
P3	35.3	81.8	56.8	109.6	90.9	848.1	101.6	1085.6	219.7	529.2	6429.1
P4 f29	52.4	104.9	50.5	82.5	97.4	1513.6	107.1	2563.5	303.4	876.9	3123.7
Alpha	51.4	115.7	68.8	118.7	95.7	667.8	106.5	1327.2	334.1		7660.2
P4 f12	63.0	115.1	59.7	100.6	119.3	1502.4	130.1	2623.0	344.6	928.1	2899.4
P1 52c	56.6	94.8	140.8	156.6	99.4	1847.6	120.1	1189.0	399.8	1029.4	5986.3
Set up nonce and encrypt 40-byte packet:											
A64	18.0	30.4	30.7	65.9	21.0	509.6	56.1	807.3	133.0	332.8	2201.0
PM	25.7	25.5	38.8	69.7	28.8	1149.8	81.5	823.7	143.8	315.5	2867.2
HP	34.8	42.8	32.1	58.2	33.0	475.6	54.2	1541.8	111.6		4783.0
PPC	22.4	52.3	44.6	66.9	31.7	459.9	62.4	909.9	169.2		2660.8
P4 f41	32.3	27.0	47.2	80.6	34.3	1220.2	83.0	1203.2	233.0	388.9	2987.5
Athlon	29.6	47.2	56.6	100.1	27.9	976.9	65.4	1192.0	179.8	450.3	2713.1
SPARC	37.0	66.9	42.4	83.0	41.0	620.0	74.7	1406.3	242.4		4191.0
P3	34.2	64.5	48.5	103.2	34.5	844.3	74.0	1082.1	164.5	527.8	6414.1
P4 f29	50.2	72.5	42.5	77.6	48.1	1508.7	78.7	2558.2	225.5	875.0	3114.7
Alpha	49.7	83.6	57.7	109.6	50.7	661.7	70.3	1322.3	237.2		7647.3
P4 f12	60.0	94.6	48.1	92.0	45.6	1492.0	89.6	2618.8	246.6	925.7	2887.5
P1 52c	52.9	71.9	118.4	148.0	58.5	1840.4	77.2	1180.2	268.4	1026.2	5973.1
Set up nonce and encrypt 576-byte packet:											
A64	11.2	9.5	24.8	29.4	7.1	48.8	7.8	71.4	12.8	35.2	160.6
PM	12.7	8.1	30.2	25.8	9.4	91.1	7.9	71.8	12.4	33.2	207.7
HP	11.5	13.2	22.4	26.0	12.2	47.7	7.6	131.3	11.3		345.3
PPC	13.7	17.1	35.0	28.9	11.5	44.7	9.2	85.1	16.7		194.3
P4 f41	15.8	7.2	33.7	29.3	12.5	106.9	9.3	107.0	19.1	40.1	216.7
Athlon	18.1	15.4	44.6	37.4	9.5	90.9	10.4	103.0	18.4	50.2	200.8
SPARC	22.7	21.1	31.9	34.4	15.4	59.0	10.9	124.7	22.6		302.6
P3	21.1	19.9	38.0	35.5	13.2	81.8	8.4	92.7	14.4	52.6	465.6
P4 f29	29.5	27.2	32.3	29.2	13.2	126.5	8.6	207.8	18.9	92.2	233.0
Alpha	22.6	28.3	43.2	52.3	16.9	64.4	11.0	128.0	23.2		549.5
P4 f12	37.1	36.4	38.2	37.6	12.2	143.1	13.6	220.2	24.6	96.9	217.0
P1 52c	33.9	22.4	88.8	59.8	20.4	172.3	12.4	126.5	28.7	102.9	440.4

	Salsa20	Phelix	AES	Dragon	SOSEMANUK	YAMB	Py6	CryptMT	Py	DICING	HC-256
Bytes	64	132	260	284	452	612	1124	3024	4196	4396	8268
Set up nonce and encrypt 1500-byte packet:											
A64	11.3	8.6	24.9	27.1	6.1	27.6	5.8	35.5	7.3	21.8	66.8
PM	12.9	7.2	30.3	23.7	8.2	41.7	4.7	35.7	6.4	20.4	85.4
HP	12.0	11.9	22.5	24.6	10.9	28.1	5.5	64.4	6.7		137.4
PPC	13.9	15.5	35.0	27.1	10.2	25.6	6.8	44.7	9.7		80.9
P4 f41	16.5	6.1	34.1	27.0	11.0	49.5	5.9	53.9	9.8	24.4	88.9
Athlon	18.4	14.0	44.6	34.5	8.1	50.1	7.8	50.8	10.8	32.4	85.2
SPARC	22.8	18.9	31.8	32.8	13.8	33.2	8.1	64.0	12.6		124.3
P3	21.4	17.8	37.9	32.3	11.8	46.7	5.4	44.7	7.6	30.8	189.1
P4 f29	29.9	25.6	31.9	26.5	11.2	56.9	9.7	99.6	9.7	56.3	95.0
Alpha	23.2	26.0	43.2	49.6	15.0	36.9	8.4	70.7	13.1		222.7
P4 f12	37.4	32.0	37.7	35.1	10.4	81.1	10.6	108.4	14.8	58.0	90.3
P1 52c	35.2	19.9	92.6	50.5	17.9	92.5	10.5	72.2	20.8	58.0	184.6
Encrypt one long stream:											
A64	11.1	5.8	24.7	7.8	5.6	14.4	3.9	15.4	3.9	13.4	8.3
PM	12.5	6.7	30.1	11.7	7.3	12.5	2.7	14.9	2.6	12.5	9.3
HP	11.3	10.5	22.3	6.2	10.1	15.3	4.4	24.6	4.3		9.9
PPC	13.6	9.6	34.8	8.4	9.4	13.7	5.3	22.7	5.4		10.4
P4 f41	15.0	5.5	33.4	12.3	8.7	16.6	3.8	23.5	3.7	14.5	9.6
Athlon	17.9	9.1	44.2	13.4	7.8	25.0	4.5	20.5	5.0	20.4	13.2
SPARC	22.3	16.9	31.6	7.9	12.2	17.3	6.1	28.3	6.1		12.9
P3	20.9	17.6	37.5	14.3	9.5	24.9	4.0	18.1	4.5	17.3	16.0
P4 f29	29.4	16.4	31.6	11.4	10.3	16.3	3.4	30.0	3.7	32.8	11.3
Alpha	22.5	19.9	42.9	12.7	13.9	19.7	6.7	38.0	6.9		18.6
P4 f12	37.0	18.2	36.9	13.1	9.4	37.8	4.4	40.5	4.8	33.9	14.0
P1 52c	34.8	17.9	91.2	25.8	17.4	42.7	9.4	43.0	10.4	29.7	26.7
Encrypt many parallel streams in 256-byte blocks:											
A64	12.0	7.5	26.5	9.3	6.3	17.8	9.2	11.0	17.5	23.8	19.2
PM	13.8	8.8	33.1	14.2	8.3	17.9	12.4	13.0	31.7	32.7	35.6
HP	12.2	14.6	24.4	8.2	11.2	18.9	10.4	26.5	20.0		28.4
PPC	14.5	12.2	38.6	10.1	10.3	17.3	13.4	21.0	31.5		31.1
P4 f41	17.6	9.7	37.3	16.2	10.7	24.0	12.7	29.3	26.4	38.5	35.3
Athlon	19.8	12.9	49.1	16.8	9.6	31.9	17.8	25.7	44.6	47.2	48.1
SPARC	23.5	19.9	35.3	10.5	13.5	20.8	15.6	30.9	31.8		41.8
P3	21.8	20.5	40.5	16.2	10.2	29.2	13.2	19.3	34.0	38.3	37.1
P4 f29	31.0	19.5	35.8	14.8	11.9	22.1	10.4	27.6	23.6	44.6	64.9
Alpha	23.4	22.4	49.2	15.5	15.0	24.9	15.1	38.4	36.0		50.0
P4 f12	39.5	23.0	41.8	17.5	11.5	45.8	17.1	41.1	30.9	50.5	38.0
P1 52c	35.0	20.0	89.0	26.8	17.5	47.2	17.1	35.4	44.1	50.6	44.8

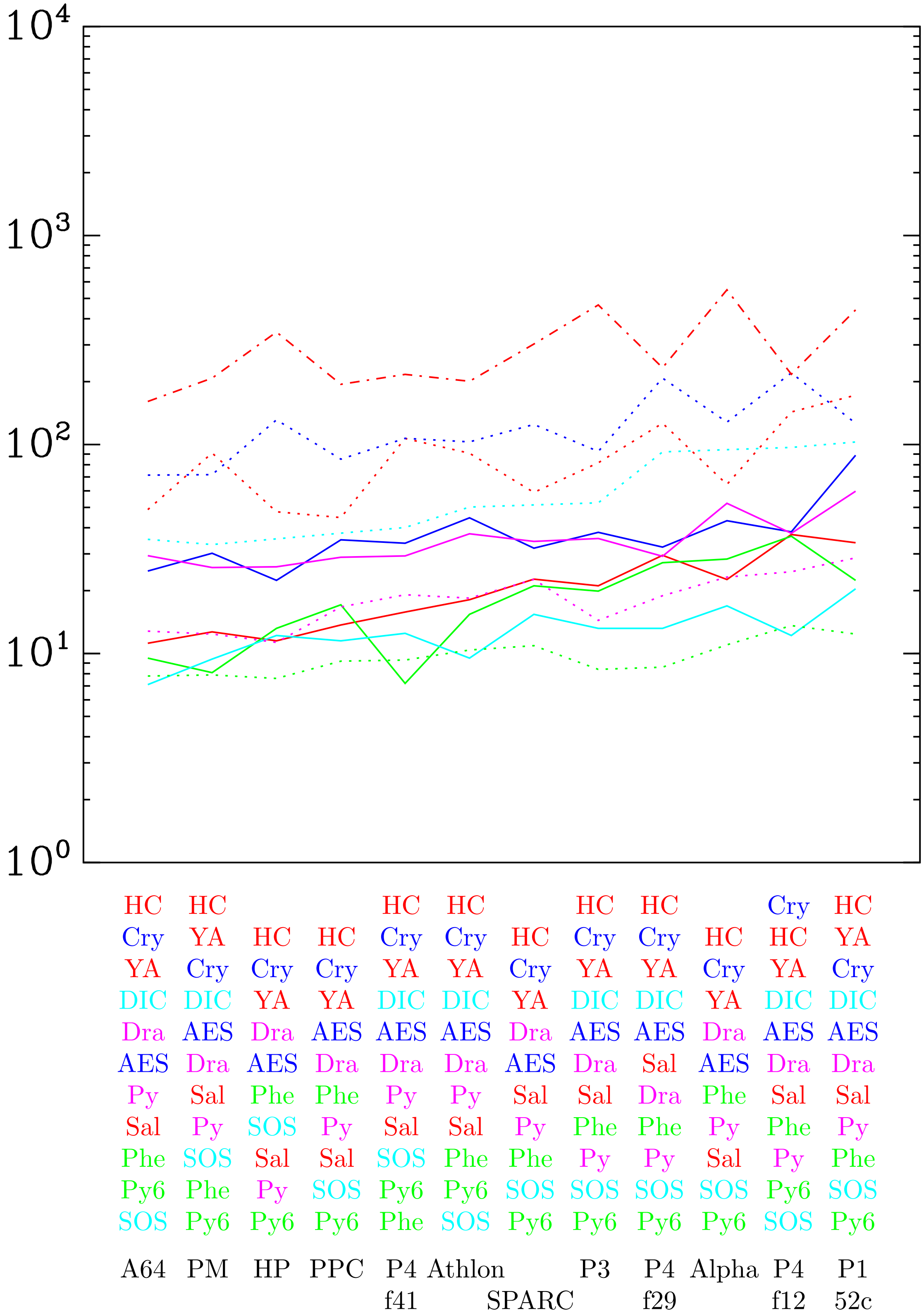
Set up key, set up nonce, and encrypt 40-byte packet



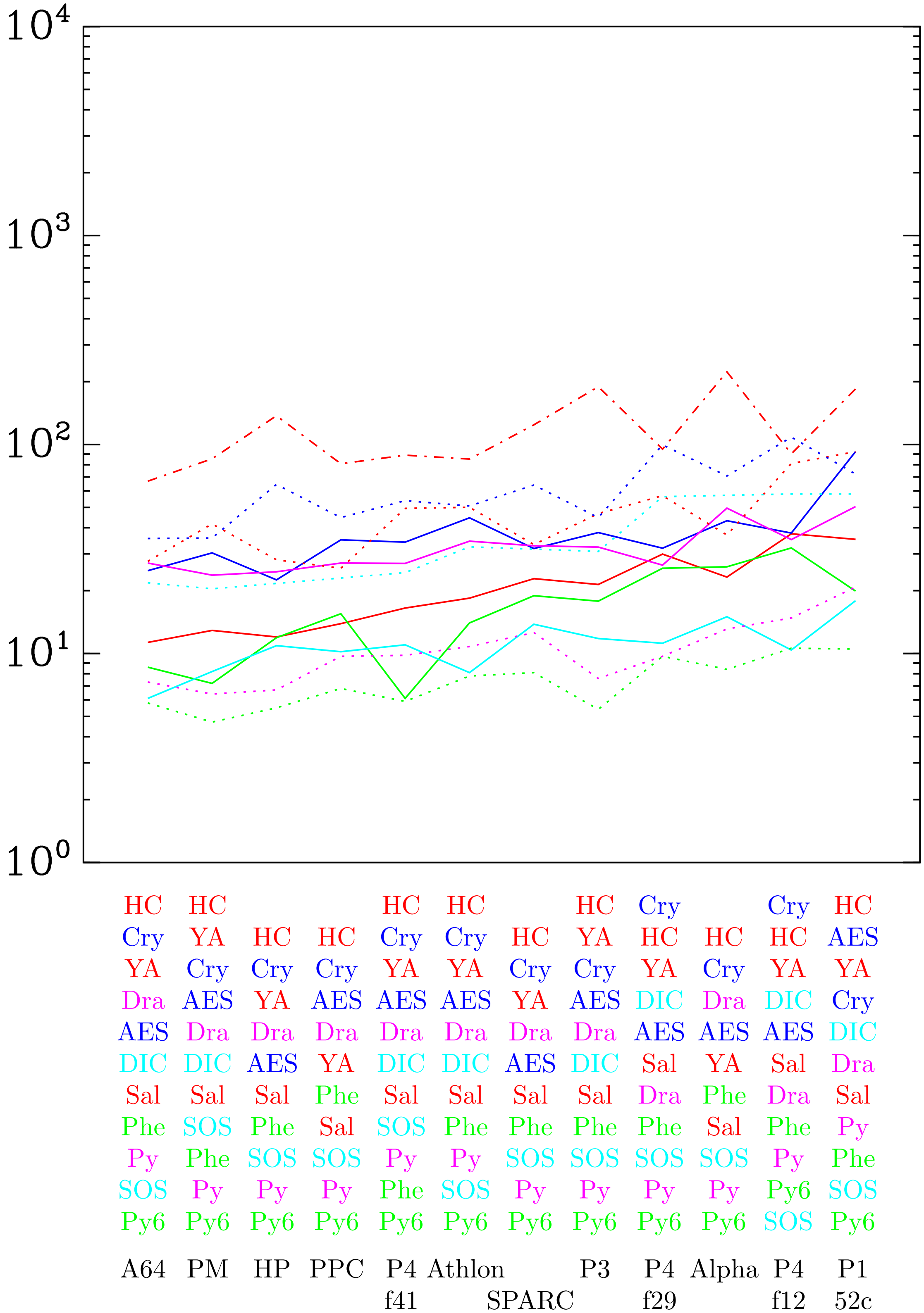
Set up nonce and encrypt 40-byte packet



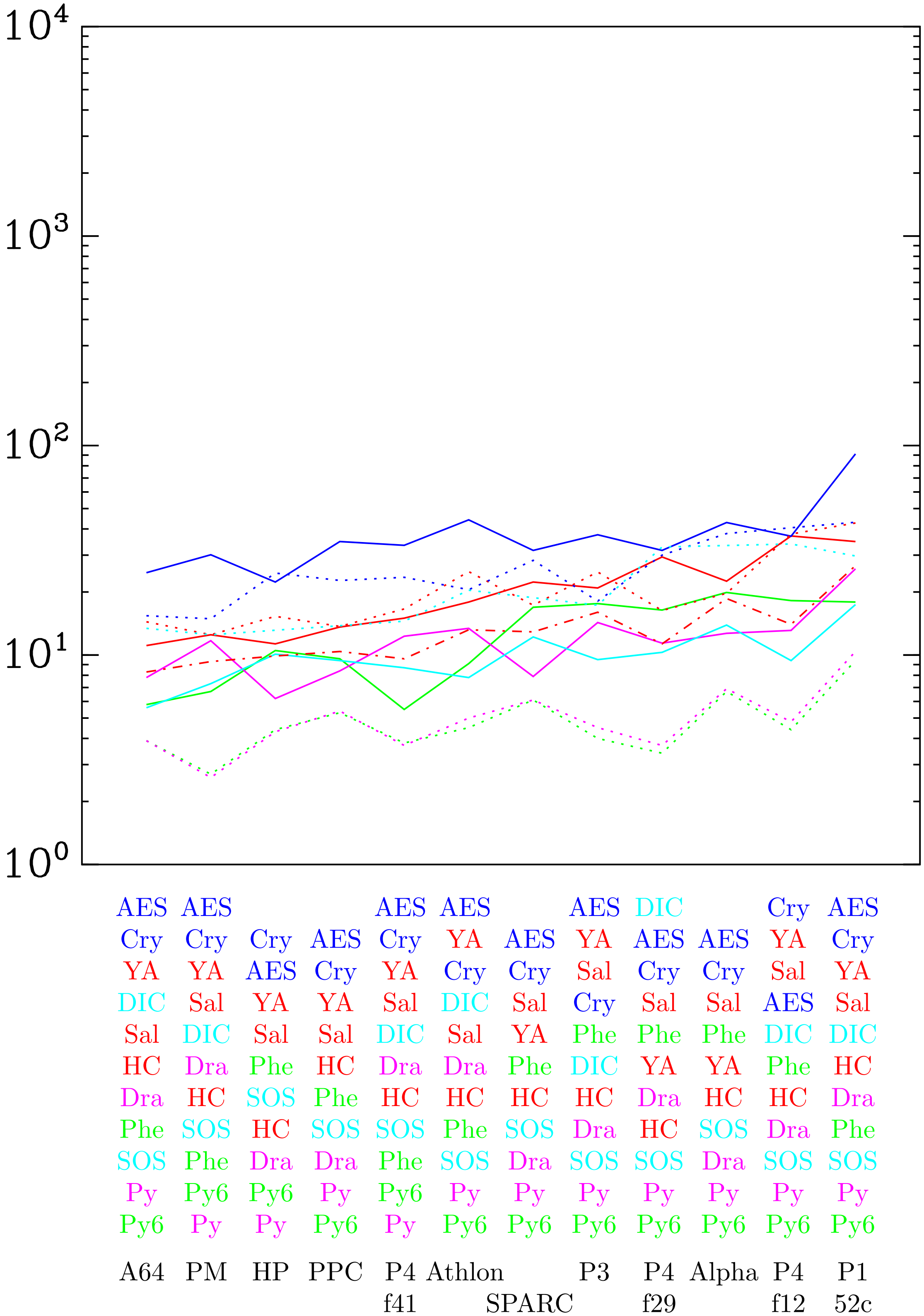
Set up nonce and encrypt 576-byte packet



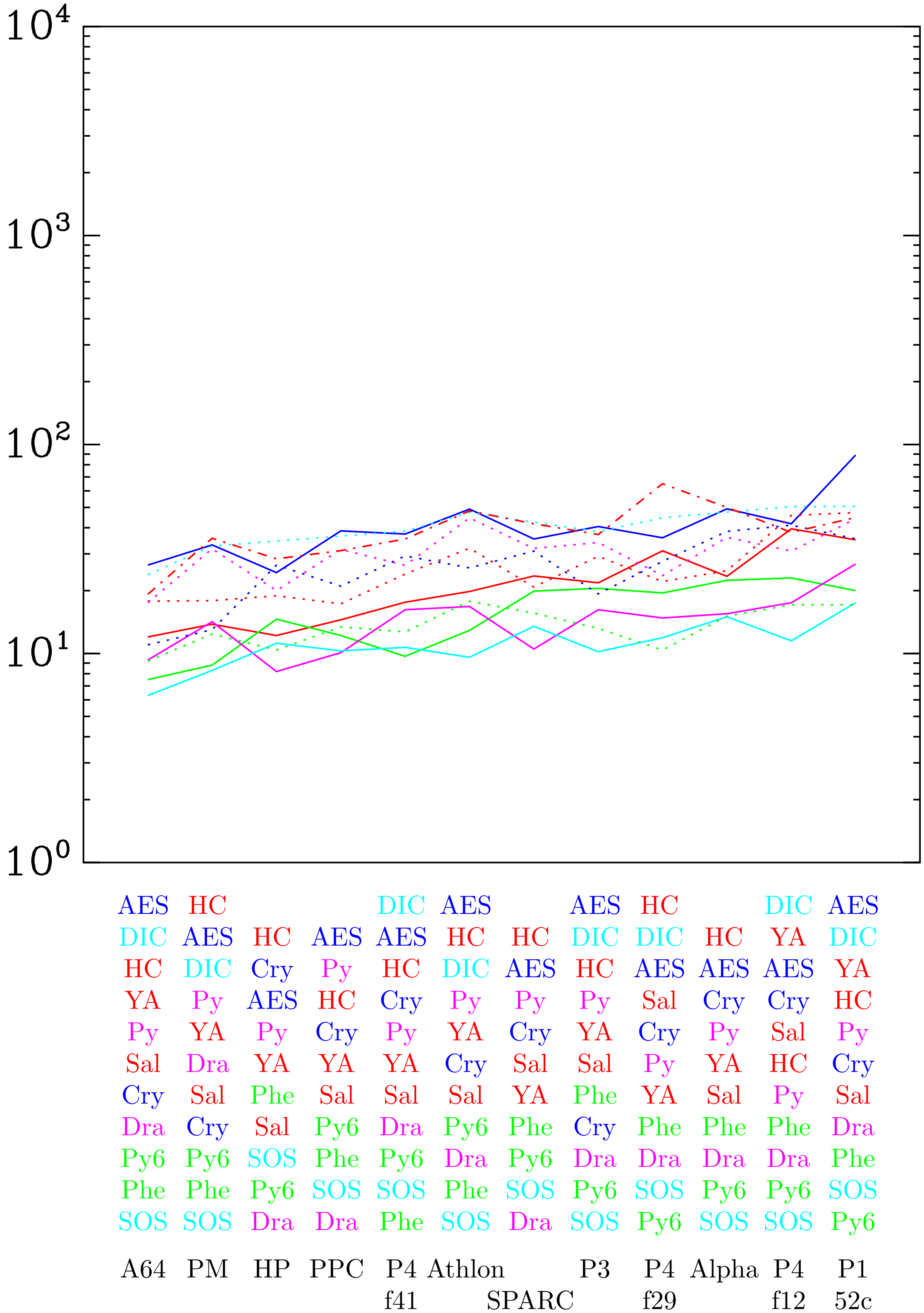
Set up nonce and encrypt 1500-byte packet



Encrypt one long stream



Encrypt many parallel streams in 256-byte blocks



4 Additional features

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

Dragon-256

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 [3] reported an attack on Py using 2^{84} output blocks and comparable time. Crowley in [2] reduced 2^{84} 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.

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 [4] reported an attack on YAMB requiring 2^{58} output blocks and comparable time. There has been no response from the authors after six months.

References

1. Daniel J. Bernstein, *Understanding brute force* (2005). URL: `http://cr.yp.to/papers.html#bruteforce`. ID `73e92f5b71793b498288efe81fe55dee`. Citations in this paper: §2.
2. Paul Crowley, *Improved cryptanalysis of Py* (2005). URL: `http://hacks.ciphergoth.org/py-cryptanalysis.pdf`. Citations in this paper: §1, §4.
3. 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.
4. 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.