

A. Cover Sheet

1. Name of submitted algorithm

- a) Hermes8-80 (Profile 1)
- b) Hermes8-128 (Profile 2)

2. Type of submitted algorithm

- Stream Cipher in $GF(2^8)$, i.e. Byte-Structure
- a) 23 byte state, 10 byte key (Hermes8-80)
 - b) 37 byte state, 16 byte key (Hermes8-128)

Proposed security level:

Short/Medium time data encryption;
but not government data for 30 years.

Proposed environment:

8 bit micro computers, ASICs
Hardware and Software

3. Principal submitter's name

Dr.-Ing. Ulrich Kaiser

Telephone

+49 8161 80 4109

Fax

+49 8161 80 4155

Organization

Texas Instruments Deutschland GmbH

Postal address

D 85350 Freising

Email address

d-kaiser@ti.com

4. Auxiliary submitters:

None.

5. Name of algorithm's inventor

Dr.-Ing. Ulrich Kaiser

6 Name of owner

Dr.-Ing. Ulrich Kaiser

7. Signature

B. Primitive specification and supporting documentation

1. Description

Hermes8 is based on the Substitution-Permutation-Network (SPN) principle [1,2,3,10]. The substitution (confusion) is performed by means of an S-BOX. The permutation and diffusion is performed by means of addressing the different state bytes, the different key bytes, and most importantly the chaining with help of the Accu (Figure 1).

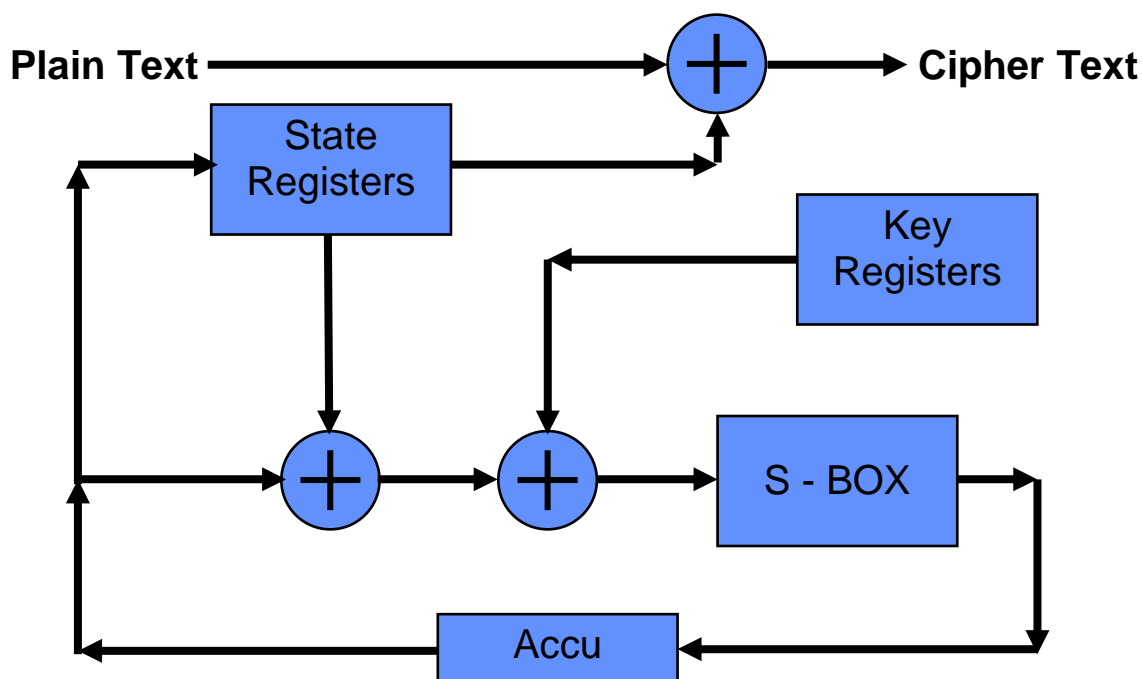


Figure 1 Principle of Hermes8

Hermes8-80 is based on 10 key bytes and 23 state bytes, whereas the larger Hermes8-128 contains 16 key bytes and 37 state bytes. There are two pointers involved: p1 addresses one of the state bytes, p2 addresses one of the key bytes (Figure 2). The pointers obey modulo addition operation in order to assure that they always address valid register space; such cyclic addressing is well known from DSPs [21]. The use of pointers is favorable over shift register designs when low-power requirements are dominating the design.

The core state operation (called sub-round) consists of

1. Select a certain state byte and EXOR it with Accu,
2. Select a certain key byte and EXOR it with previous result,
3. Take the previous result and apply the S-BOX function,
4. Store the previous result in Accu,
5. Copy Accu into the same state byte selected in step 1.

The S-BOX is 8-bit wide in order to provide a proper non-linear Boolean function needed for substitution, i.e. confusion [8,9]. First choice is the known S-BOX of AES [4,5] which is strong w.r.t. Differential Cryptanalysis. – But also random number based S-BOXes are suitable, if their differential distribution table (ddt) demonstrates good quality [15] with respect to DC attacks.

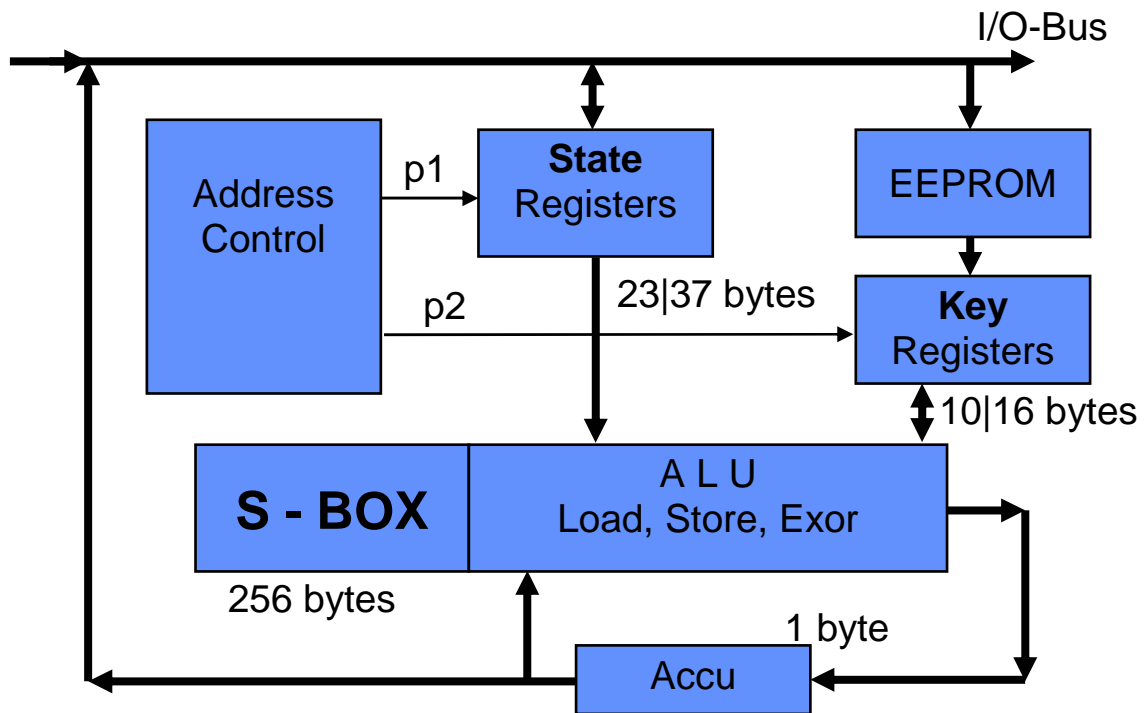


Figure 2 Byte-Architecture of Hermes8

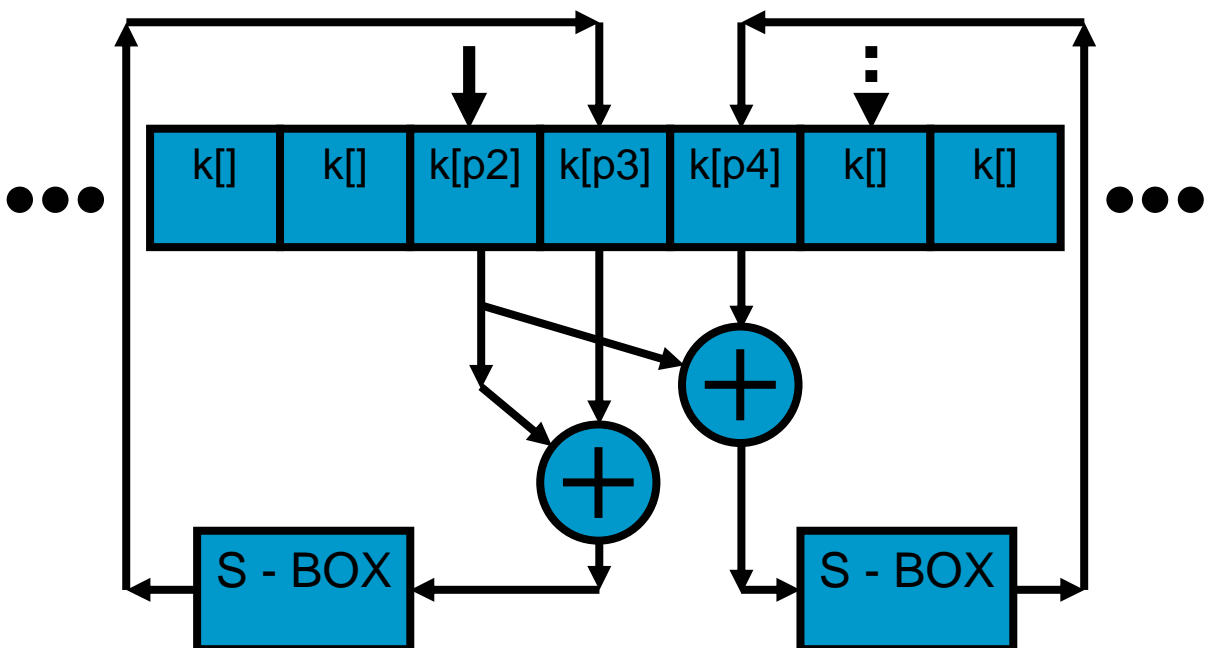


Figure 3 Key Modification and Scheduling Method

The key bytes are modified every KEY_STEP3, i.e. seven steps, during the sub-round loops depending on the position of p2. The details are shown in Figure 3 : Two temporary pointers p3 and p4 are addressing the key bytes following the byte addresses by p2. The byte k[p2] is not modified because it has to be used in the following sub-round. But the bytes k[p3] and k[p4] are 'rather old' and are therefore candidates for modification; they are replaced by $SBOX[k[p3] \text{ exor } k[p2]]$ and $SBOX[k[p4] \text{ exor } k[p2]]$ respectively. The exor'ing with

$k[p2]$ is advantageous over the direct application of the SBOX, because the inverse function of the SBOX does exist. Therefore, backtracking is hampered by means of this method.

The dashed pointer in Figure 3 represents the next $p2$ position (because $KEY_STEP1=3$) when addressing the next key byte needed for the next sub-round.

Figure 4 describes how the output bytes for the key stream $ks[]$ are derived from the state bytes $state[]$. Since the pointer $p1$ has been incremented after the last sub-round, it points to the 'oldest' available state byte. This is the first byte to be packed into the key stream block of e.g. eight bytes for Hermes8-80 or sixteen bytes for Hermes8-128. Then further bytes follow by means of output pointer po , that is incremented by two in order to separate consecutive sub-rounds from each other.

Since a new output block of key stream bytes follows not earlier than the next $STREAM_ROUNDS=3$ are completed, the state byte contents corresponding to the same address are separated by 3×23 sub-rounds respectively 3×37 sub-rounds.

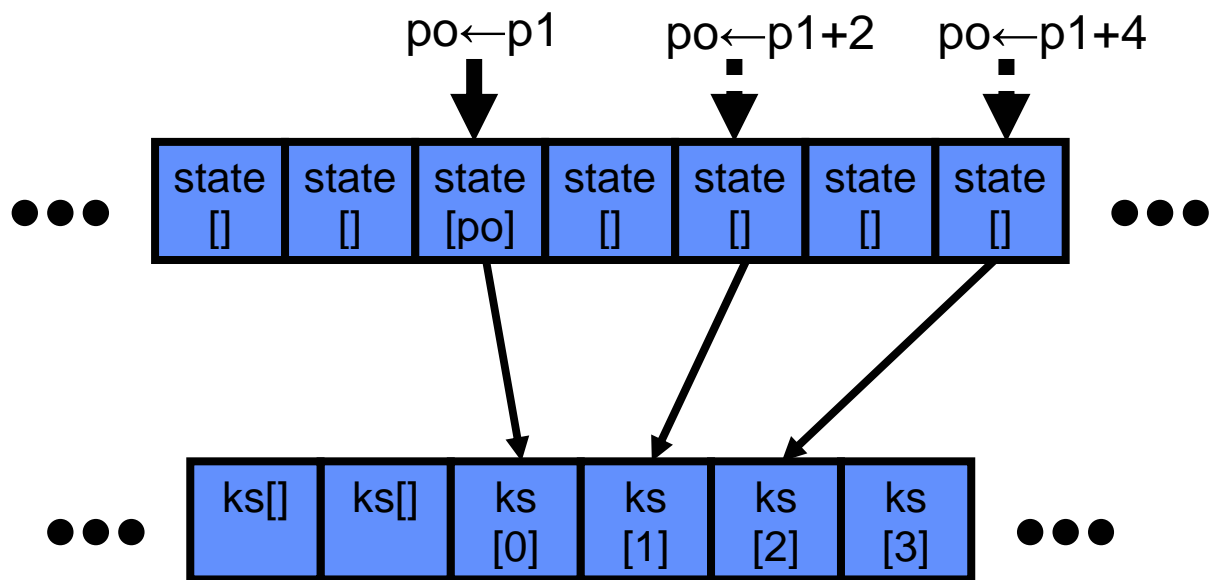


Figure 4 Output Function

During the 69 Hermes8-80 sub-rounds there are nearly ten occurrences of key modification, i.e. about 20 key bytes are modified per output block in relation to ten key byte registers.

During the 111 Hermes8-128 sub-rounds there are nearly 16 occurrences of key modification, i.e. about 32 key bytes are modified per output block in relation to 16 key byte registers.

A related mechanical model consists of two wheels. One has 23 teeth and needs 23 steps per round, the second one has only ten teeth, but rotates with a three-fold speed. When the first one has performed three rounds with 69 steps, the smaller one has rotated for 207 steps, i.e. nearly 21 turns.

Pseudo-Code of Hermes8-80

```

1   nx ← 23
2   nk ← 10
3   OUTPUTBYTES ← 8
4
5   INIT_ROUNDS ← 10
6   STREAM_ROUNDS ← 3
7   KEY_STEP1 ← 3
8   KEY_STEP2 ← 5
9   KEY_STEP3 ← 7
10
11  k[] ← load( nk key bytes)
12  state[] ← load( nx IV bytes )
13
14  p1 ← ( k[0] exor k[1] exor k[2] ) mod nx
15  p2 ← ( k[3] exor k[4] exor k[5] ) mod nk
16  accu ← k[6] exor k[7] exor k[8]
17  src ← ( k[9] exor k[0] exor k[3] ) mod KEY_STEP3
18  round ← 0
19
20  for INIT_ROUNDS do
21  begin
22      round ← round + 1
23      /* begin of core */
24      for nx subrounds do
25      begin
26          accu ← accu exor state[p1] exor k[p2]
27          accu ← SBOX[ accu ]
28          state[p1] ← accu
29          p1 ← ( p1 + 1 ) mod nx
30          p2 ← ( p2 + KEY_STEP1 ) mod nk
31          src ← src + 1
32          if( src ≥ KEY_STEP3 )
33          then
34              begin /* two key modifications */
35                  src ← src - KEY_STEP3
36                  p3 ← ( p2 + 1 ) mod nk
37                  p4 ← ( p3 + 1 ) mod nk
38                  k[p3] ← SBOX[ k[p3] exor k[p2] ]
39                  k[p4] ← SBOX[ k[p4] exor k[p2] ]
40              endif
41          endfor
42          if ( round mod KEY_STEP2 equal 0 ) then p2 ← ( p2 + 1 ) mod nk
43          /* end of core */
44
45
46  endfor
47  /* initialization completed */
48
49  pc ← 0
50  for MAX_ROUNDS do

```

```

51  begin
52      round  $\leftarrow$  round + 1
53      for STREAM_ROUNDS do
54          /* begin of core */
55          for nx subrounds do
56              begin
57                  accu  $\leftarrow$  accu exor state[p1] exor k[p2]
58                  accu  $\leftarrow$  SBOX[ accu ]
59                  state[p1]  $\leftarrow$  accu
60                  p1  $\leftarrow$  ( p1 + 1 ) mod nx
61                  p2  $\leftarrow$  ( p2 + KEY_STEP1 ) mod nk
62                  src  $\leftarrow$  src + 1
63                  if( src  $\geq$  KEY_STEP3 )
64                      then
65                          begin /* two key modifications */
66                              src  $\leftarrow$  src - KEY_STEP3
67                              p3  $\leftarrow$  ( p2 + 1 ) mod nk
68                              p4  $\leftarrow$  ( p3 + 1 ) mod nk
69                              k[p3]  $\leftarrow$  SBOX[ k[p3] exor k[p2] ]
70                              k[p4]  $\leftarrow$  SBOX[ k[p4] exor k[p2] ]
71                          endif
72                      endifor
73                  if ( round mod KEY_STEP2 equal 0 ) then p2  $\leftarrow$  ( p2 + 1 ) mod nk
74                  /* end of core */
75              endifor
76              /* key stream round completed */
77
78              po  $\leftarrow$  p1
79              for 1 to OUTPUTBYTES do
80                  begin
81                      ciphertext[pc]  $\leftarrow$  plaintext[pc] exor state[po] /* encrypt */
82                      pc  $\leftarrow$  pc + 1
83                      po  $\leftarrow$  ( po + 2 ) mod nx
84                  endifor
85              endifor

```

For Hermes8-128 only the three lines 1 - 3 are changed to $nx \leftarrow 37$, $nk \leftarrow 16$, and $OUTPUTBYTES \leftarrow 16$.

Lines 14 to 47 show the initialization phase assuming the IV has already been loaded into the state registers. The cyclic pointer p2 to the key registers is incremented in steps larger than 1 in order to assign a certain key byte to every state byte over time. Additionally, the pointer p2 is also incremented after every 5th round (line 42, KEY_STEP2); this shifts the key assignment pattern, too. After every 7 sub-rounds (KEY_STEP3) two key bytes are modified by means of the S-BOX (lines 31-40).

MAX_ROUNDS (line 50) specifies how many multiples of OUTPUTBYTES bytes shall be encrypted. It is assumed that the plaintext is also a multiple of OUTPUTBYTES bytes, i.e. has been padded accordingly.

The encryption by means of the key stream bytes in the state register is shown in lines 53-84.

During 'key streaming' the inner core of the algorithm (54-74) is the same as described for the initialization phase (23-43). The number of rounds between the output of two blocks of key-stream bytes is defined by `STREAM_ROUNDS`.

2. Hidden weaknesses

I have not implemented any hidden weakness.

3. Security properties, security levels, attacks

3.1 Strict Avalanche Criterion

The initialization phase has been evaluated with respect to the Strict Avalanche Criterion (SAC) [1,10]. This has been done not only for the key sensitivity but also for the IV sensitivity. Only two rounds are needed to get very close to the 50% goal (see appendix for SAC plots). If ten rounds are performed during the initialization, the security level is assumed to be so high, that only exhaustive search can find the correct key or IV value from known plaintext / cipher text pairs.

3.2 Differential and Linear Cryptanalysis

The algorithm has been tested for DC and LC weakness (sensitivity, affinity, correlation) with respect to the initialization phase of ten rounds. No problems were found.

Several parts of the output stream (e.g.192 bits) were applied to the Berlekamp-Massey algorithm. There was no exponential found below X^{93} .

3.3 Random Number Quality tests

The algorithm has been tested for FIPS 140-2; no problems were found.

The algorithm was also tested by means of the Diehard test suite; no problems could be discovered.

3.4 Some Attack Scenarios

In [22] some attacks on pseudorandom number generators (PRNG) are described: a) direct cryptanalytic attack, b) input-based attacks, c) state compromise extension attack. Since PRNGs are very similar to stream ciphers, the same attacks shall be considered here.

3.4.1 Direct Cryptanalytic Attack

Since the SAC is fulfilled quite well after only three rounds, a direct attack on ten rounds initialization seems to be unfeasible w.r.t. exhaustive search. - However the key stream generation is based on shorter rounds, i.e. only three. But only 8/23 respectively 16/37 state bytes can be directly seen here.

3.4.2 Input-Based Attacks

An adversary might use the initialization phase and the IV value for known-input, replayed-input or chosen-input attacks. However, there is a stream cipher application rule that the first IV has to be chosen as a good random number; sub-sequent IVs might be derived from that, and no (IV, key)-pair must be used twice. – In Hermes8 the IV is not used to derive any initial pointer value or similar variable. -- Since the SAC properties are strong, it is assumed that input-based attacks are not more efficient than exhaustive search.

3.4.3 State Compromise Extension Attacks

The key stream consists of consecutive blocks of 8 bytes (Hermes8-80) or 16 bytes (Hermes8-128). Two consecutive blocks are separated by 69 sub-rounds respectively 111 sub-rounds. And during these 69 (111) steps the key bytes are modified 20 (32) times. This leads to a certain number of unknown bits, i.e. a certain complexity.

Version	nx	nk	state output	state distance	state unknown	key unknown	bits unknown
Hermes8-80	23	10	8	69	61	20	648
Hermes8-128	37	16	16	111	95	32	1016

If the number of unknown bits is not enough, the algorithm can be made harder by extending the number of STREAM_ROUNDS to more than three.

3.5 Weak Keys

Due to the method of the key scheduling all keys with equal byte pattern are weaker than randomly generated keys.

Example: If the initial key is all zero we obtain for hermes8-80 after the 10 initial rounds:

Key: 0x
4b 4b b0 4d ba 44 2 a0 f3 25

and for hermes8-128 the related result is

Key: 0x
a3 c2 ee bf 3a a3 b2 45 e0 70 1b a3 c2 ee bf 3a

The repetition of bytes here is also caused by the application of KEY_STEP3 = 5, i.e. the pointer p2 is only one time during initialization increased additionally. – Of course, one could change KEY_STEP3 from 5 to 1 for the initialization phase only, but generally the key bytes have to be produced by means of a good random number generator.

4. Strength and Advantages

Strength:

- one 8x8 S-BOX (e.g. AES S-BOX)
- the S-BOX is used in every sub-round [10]
- the S-BOX is used for a specialized key scheduling
- every sub-round involves one state-byte and one key-byte
- no conditional branch is dependent directly on key content.
- learned from AES [4,5,11,12]

Advantages:

- number crunching of bytes (=> fast on 8-bit micros)
- no bit-shifting ! (=> high efficiency in software)
- low complexity [20]

5. Design Choices

The strength and advantages listed above are the result of the following design choices, options, and alternatives:

- The state size is more than twice as the key size, in order to prevent time-memory trade-off attacks [19].
- Substitution Permutation Network (SPN)
- Clarity of design, low complexity [20].
- Use of only registers, three pointers, EXORs, one large S-BOX [8,9], small control logic.
- Constants KEY_STEP1, 2, and 3 are chosen as primes not being factors of nx or nk.
- Prevention against related key attacks [4] due to key modification/scheduling.
- Prevention against backtracking attacks [22] due to special key modification/scheduling.
- No bit-shifting, no LFSRs in order to avoid slowdown of software implementations.
- No additions, subtractions, multiplications, divisions in the core data flow.
- No constraint on IV length, beside nx as maximum.
- Low-power architecture [16]
- Scalable architecture concept (StateSize > 37 bytes, KeySize > 16 bytes)

6. Computational efficiency

6.1 Computational efficiency in software

The following assumes an 8-bit microcomputer with two-operand instruction set and RISC architecture. The S-BOX access is assumed to be one cycle, i.e. table look-up. The **mod** operation is performed by means of conditional subtraction for software speed-up.

Key setup: 1 cycle per byte

Primitive setup:

1 cycle per byte	loading the IV, padding with constant
12 cycles	initialize pointers, counters, accu
1 cycle	reset round counter
2 cycles	loop control for INIT_ROUNDS
1 cycle	increment round counter
2 cycles	loop control for nx sub-rounds
2 cycles	2 times EXOR
1 cycle	S-BOX access
1 cycle	new state byte
3 cycles	update p1
3 cycles	update p2
1 cycle	increment src
1 cycle	conditional key modification
<i>1 cycle</i>	<i>decrement src</i>
<i>3 cycles</i>	<i>calculate p3</i>
<i>3 cycles</i>	<i>calculate p4</i>
<i>3 cycles</i>	<i>new k[p3]</i>
<i>3 cycles</i>	<i>new k[p4]</i>
2 cycles average	conditional increment p2

Sum: $nx + 13 + \text{INIT_ROUNDS} \cdot (3 + nx \cdot 14 + 1/7 \cdot nx \cdot 13 + 2)$

Streaming Part:

2 cycles	loop control for MAX_ROUNDS
1 cycle	increment round counter
2 cycles	loop control for STREAM_ROUNDS
2 cycles	loop control for nx sub-rounds
2 cycles	2 times EXOR
1 cycle	S-BOX access
1 cycle	new state byte
3 cycles	update p1
3 cycles	update p2
1 cycle	increment src
1 cycle	conditional key modification
<i>1 cycle</i>	<i>decrement src</i>
<i>3 cycles</i>	<i>calculate p3</i>
<i>3 cycles</i>	<i>calculate p4</i>
<i>3 cycles</i>	<i>new k[p3]</i>
<i>3 cycles</i>	<i>new k[p4]</i>
2 cycles average	conditional increment p2

<p>2 cycles 1 cycle 1 cycle 3 cycle</p>	<p>loop control for encryption EXOR operation on plaintext byte increment P/C pointer increment po pointer</p>
------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------

Sum for one block with OUTPUTBYTES bytes:

$$3 \cdot (3 + 2 + nx \cdot 14 + 1/7 \cdot nx \cdot 13 + 2) + OUTPUTBYTES \cdot 7$$

Both graphs below show the asymptotic efficiency curves (limes = 146 or 118 for $n \rightarrow \infty$); the efficiency for large amounts of data depends therefore as expected on the streaming loop performance. Some savings can be obtained by means of loop un-rolling, e.g. reducing the cycle count by $OUTPUTBYTES \cdot 2$ for the encryption loop.

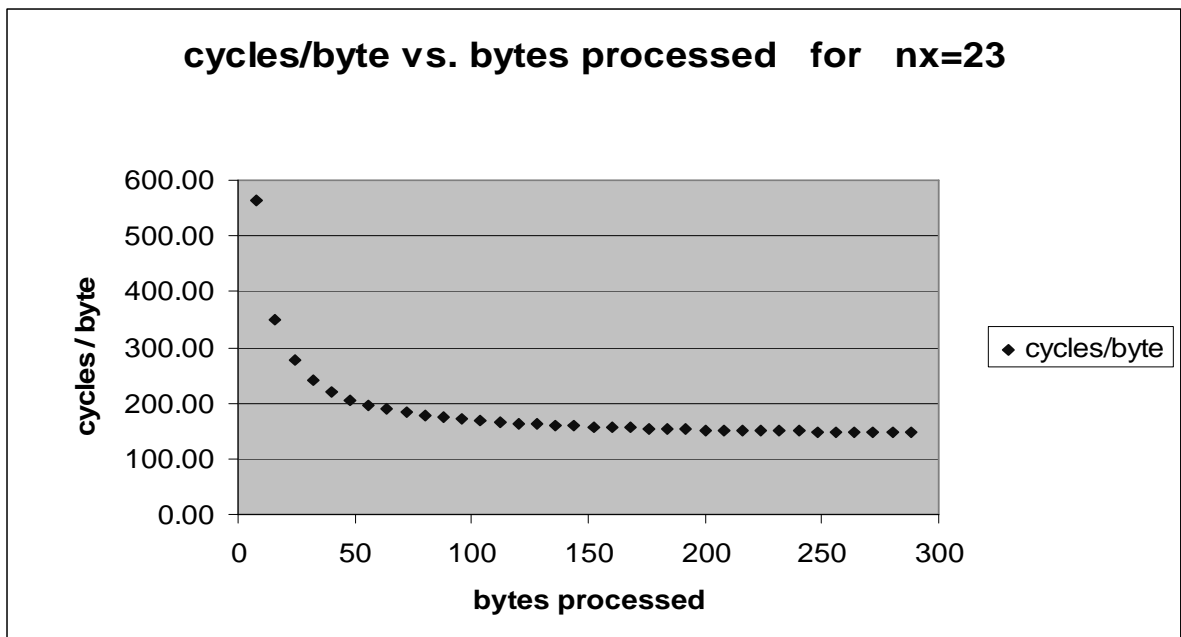


Figure 3 cycles/byte versus bytes processed for Hermes8-80

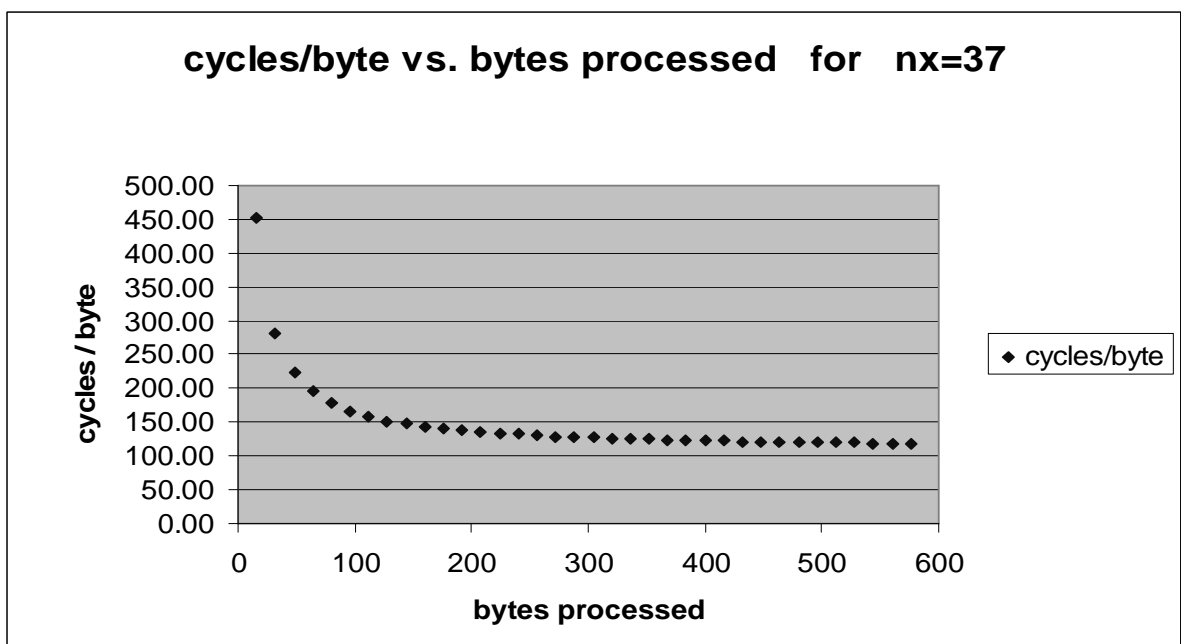


Figure 4 cycles/byte versus bytes processed for Hermes8-128

6.2 Computational efficiency in hardware

As described in the previous chapter, the key stream generation loop and the encryption loop are dominating the efficiency. In hardware, therefore, it is important to perform as many operations in parallel as possible. Since the ROM containing the S-BOX table is pre-charged with clock CLK=1 and read out with the falling edge of CLK, other operations are executed with the rising edge of CLK, e.g. update of registers and round counter. The related control logic (finite state machine, FSM) has the responsibility for the correct timing of the operations, especially the conditional modification of the key byte replacement by means of S-BOX application (line 63-71).

CLK rising edge operations:

```

52         round ← round + 1
           /* the following three lines, if output is required */
81         ciphertext[pc] ← plaintext[pc] exor state[po] /* encrypt */
82         pc ← pc + 1
83         po ← (po + 2) mod nx
58a        accu ← sbox_out
59         state[p1] ← sbox_out
57         address ← accu exor state[p1] exor k[p2]

```

CLK falling edge operations:

```

58b        sbox_out ← S-BOX-TABLE[ address ]
60         p1 ← ( p1 + 1 ) mod nx
61         p2 ← ( p2 + KEY_STEP1 ) mod nk
82         src ← src + 1
73         if ( round mod KEY_STEP2 equal 0 ) then p2 ← ( p2 + 1 ) mod nk

```

The operations above are executed 7 times (KEY_STEP3); then the following has to be inserted :

```

66         src ← src - KEY_STEP3
67         p3 ← ( p2 + 1 ) mod nk
68         p4 ← ( p3 + 1 ) mod nk
69         k[p3] ← SBOX[ k[p3] exor k[p2] ]
70         k[p4] ← SBOX[ k[p4] exor k[p2] ]

```

that means

CLK rising edge operations:

```

           /* p3 and p4 are always calculated in parallel to p2, line 61 */
69a        address ← k[p2] exor k[p3]

```

CLK falling edge operations:

```

69b        sbox_out ← S-BOX-TABLE[ address ]
50         src ← src - 7

```

CLK rising edge operations:

```

69c        k[p3] ← sbox_out
70a        address ← k[p2] exor k[p4]

```

CLK falling edge operations:

```

70b        sbox_out ← S-BOX-TABLE[ address ]

```

CLK rising edge operations:

```

70c          k[p4] ← sbox_out
57           address ← accu exor state[p1] exor k[p2]
a.s.o.

```

The resulting efficiency depends on the degree of parallelism reached and the amount of pipeline registers that are spent additionally.

7. Implementation items to avoid weaknesses

Compared to other ciphers, the literature about side-channel attacks on stream ciphers is rare; an overview is given in [18].

For Hermes8-80 and -128 the following countermeasures are proposed:

- a) When the key is loaded from non-volatile memory into the key byte array, the related bus should have bus-scrambling, 2x8 wire differential drivers, or similar DPA [13,14] protection.
- b) The S-BOX should be implemented as ROM with pre-charge technique. This is favorable over the algebraic S-BOX [11,12] with three internal multipliers that are sensitive to products of zero.
- c) The Accu should be built with 16 DFFs, so that the inverted output of the S-BOX is stored as well and DPA attacks are hampered.
- d) All DFFs in the registers and Accu should be built in CSEM style [16] in order to avoid hazards and minimize DPA susceptibility.
- e) The first IV must be generated by means of a TRNG, later IVs can be built by continuous incrementing the first IV [19].

8. Early Hardware Evaluations

An electrical Spice3 simulation was performed in an early design stage. The following hardware parts were connected:

- SBOX ROM 8 x 8 with pre-charged N-channel MOS transistor array
- Accu (8 DFFs)
- S-Register (8 DFFs)
- Eight capacitors (as replacement for the other nx-1 state registers)
- K-Register (instead of 8 multiplexers with nk inputs)
- 16 EXOR gates
- One clock driver

Based on the models of a 0.35 CMOS DLP TLM process, a current consumption of only 5uA was obtained when simulating with $f=500\text{kHz}$, $VCC=2\text{V}$, models=typical, temperature= 27°C . -- However, the technology allows decreasing the VCC to the sum of one N-channel transistor threshold voltage and one P-channel transistor threshold voltage. This is especially advantageous because the power dissipation is proportional to the supply voltage squared, but only proportional to the clock frequency.

The area estimation regarding the CMOS process mentioned above and the method of estimation in [17] is depicted below:

	0.35 CMOS	process in [17]
Hermes8-80	1711	4026
Hermes8-128	2400	5946

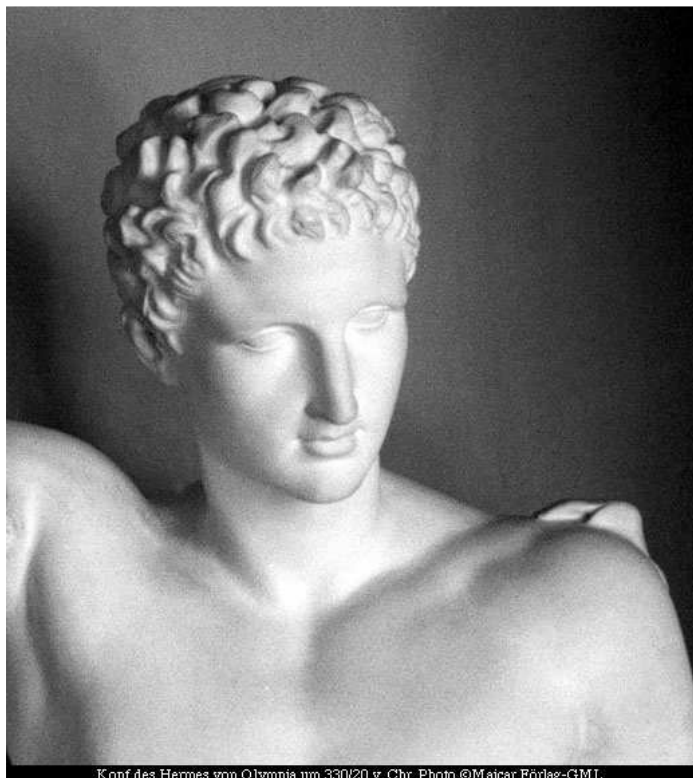
The higher numbers regarding [17] are caused by the much higher gate count for the DFF compared to the 0.35 μm CSEM DFF [16], i.e. 12 instead of 4.3 !

9. Outlook

The algorithm principle is not only extendable w.r.t. number of bytes for state and key, but also w.r.t. word length of the registers. For example, an architecture with 16 bit words and two S-BOXes (or S-BOX calls) could be build with the same property of low complexity [15]. Especially interesting is the low-power processor MSP430 in this case. - The same holds for an architecture with four S-BOXes (or S-BOX calls) on a 32-bit digital signal processor (DSP) such as TMS320C2xxx or TMS320C5xxx [21] where circular addressing is well supported. – A dedicated hardware can lead to a nearly four-fold throughput, then.

10. Acknowledgments

The author wants to thank for the kind support received from John Gordon, Vincent Rijmen, Christof Paar, Sean Murphy, and Matt Robshaw. Special thanks go to the three anonymous reviewers of the first Hermes8 version, the organizers of the SKEW 2005 workshop and Joan Daemen for the encouraging talk about Simplistic Stream Cipher Design [20].



Kopf des Hermes von Olympia um 330/20 v. Chr. Photo ©Malcar Förlag-GML

Hermes of Olympia 330 b.C.

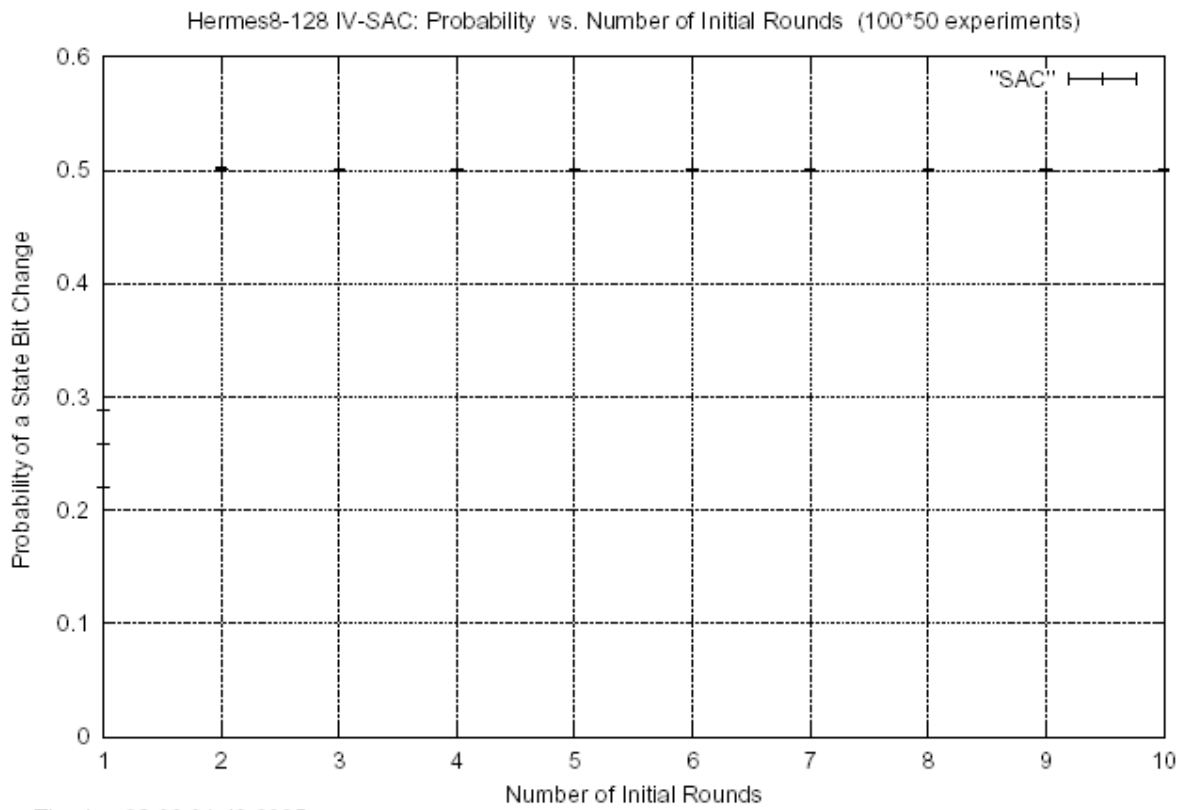
<http://en.wikipedia.org/wiki/Hermes>

11. References

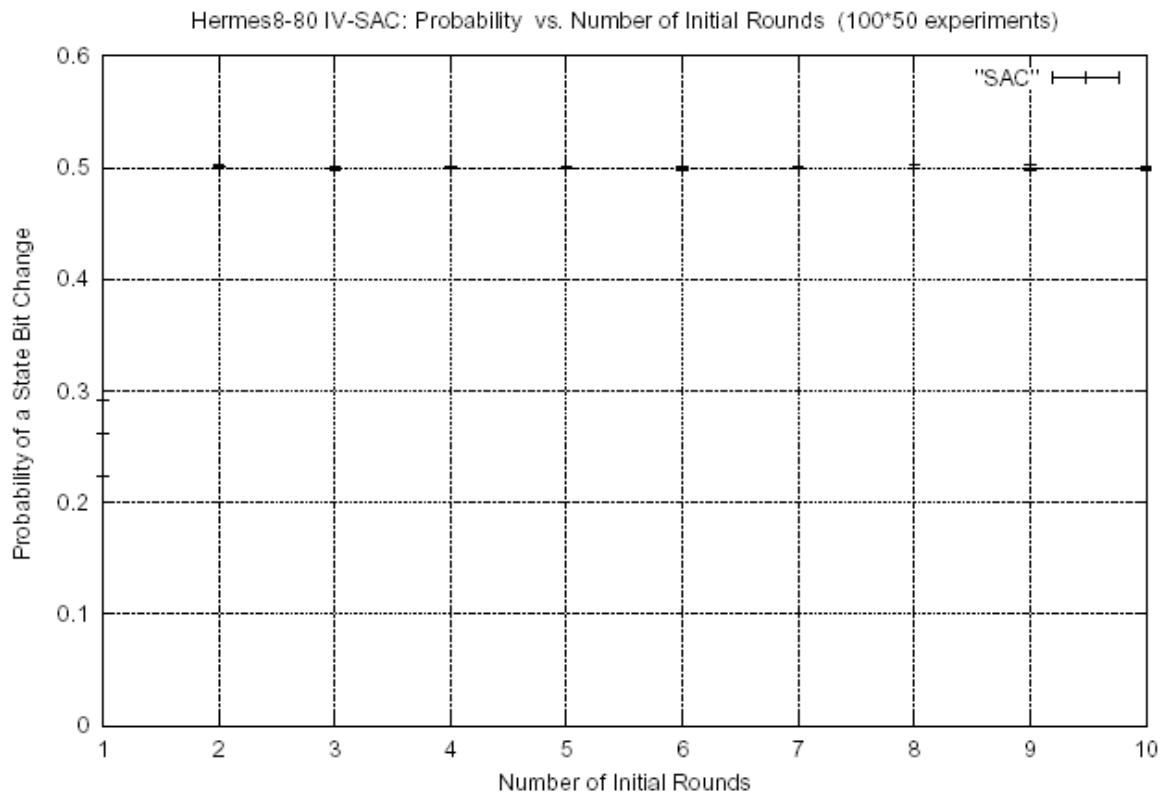
- [1] A. Menezes, P. van Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997
- [2] D. Stinson, Cryptography - Theory and Practice, CRC Press, 1995
- [3] B. Schneier, Applied Cryptography, Wiley, 1994
- [4] J. Daemen, V. Rijmen, AES Proposal: Rijndael, Version 2, 03/09/99, 45 pages and related Reference Code in C
<http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaelref.zip>
- [5] NIST FIPS 197, Advanced Encryption Standard (AES), Nov. 26, 2001, 47 pages
- [6] NIST FIPS 140-2, Security Requirements for Cryptographic Modules, May 25, 2001,
<http://csrc.nist.gov/cryptval> , <http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf>
- [7] National Institute of Standards and Technology, FIPS PUB 140-2 Annex A: Approved Security Functions, www.nist.gov/cmvp.
- [8] J. Seberry, X. Zhang, Y. Zheng, Pitfalls in Designing Substitution Boxes, Crypto'94, Aug. 1994, pp 383ff
- [9] J. Gordon, A. Retkin, Are Big S-Boxes Best ?, IEEE Workshop on Communication Security, Santa Barbara, Cal. 1981, pp. 1-6
- [10] H. Heys, S. Tavares, Substitution-Permutation Network Resistant to Differential and Linear Cryptanalysis, Journal of Cryptology, Vol. 9, No. 1, pp.1-19, 1996
- [11] J. Rejeb, V. Ramaswamy, K. Ghadiri, Hardware Implementation of the Rijndael Algorithm for High-Speed Networks, ISPC 2003, March 2003, Dallas, 6 pages
- [12] H. Kuo, I. Verbauwhede, Architectural Optimization for a 1.82Gbits/sec VLSI Implementation of the AES Rijndael Algorithm, CHES 2001, LNCS 2162, pp. 51-64, Springer 2001
- [13] Kocher, Jaffe, Jun, Differential Power Analysis, Advances in Cryptology, CRYPTO'99, LNCS 1666, Springer 1999, 10 pages
- [14] Kocher, Evaluating Cryptosystems, 31 slides, Cryptography Research 2002,
<http://www.cryptography.com/resources/whitepapers/HackingCryptosystems.pdf>
- [15] U. Kaiser, Universal Immobilizer Crypto Engine, "UICE, the little brother of AES",
http://www.aes4.org/english/events/aes4/downloads/AES_UICE_slides.pdf
- [16] C. Piguat, Design of Low-Power Libraries, ICECS 1998
- [17] L. Batina, J. Lano, N. Mentens, S. B. Oers, B. Preenel, I. Verbauwhede, Energy, performance, area versus security trade-offs for stream ciphers, ECRYPT workshop, SASC – The State of the Art of Stream Ciphers, Bruegge, 14.Oct.2004, 9 pages
- [18] S. Kumar, K. Lemke, C. Paar, Some Thoughts about Implementation Properties of Stream Ciphers, ECRYPT Workshop, SASC – The State of the Art of Stream Ciphers, Bruegge, 14.Oct.2004, 9 pages
- [19] C. DeCanniere, J. Lano, B. Preenel, Comments on the Rediscovery of the Time Memory Data Tradeoffs, KUL, April 2005, 5 pages, <http://www.ecrypt.eu.org/stream/TMD.pdf>
- [20] J. Daemen, Simplistic Stream Cipher Design, Workshop on Symmetric Key Encryption, SKEW 2005, 26.+27.May.2005, Aarhus, Denmark
- [21] TMS320C5x User's Guide, Digital Signal Processing Products, Texas Instruments, 1993
- [22] J. Kelsey et al., Cryptanalytic Attacks on Pseudorandom Number Generators, Fast Software Encryption, FSE 1998, March 1998, pp.168-188

12. Appendix

Strict Avalanche Criterion with IV stimulation

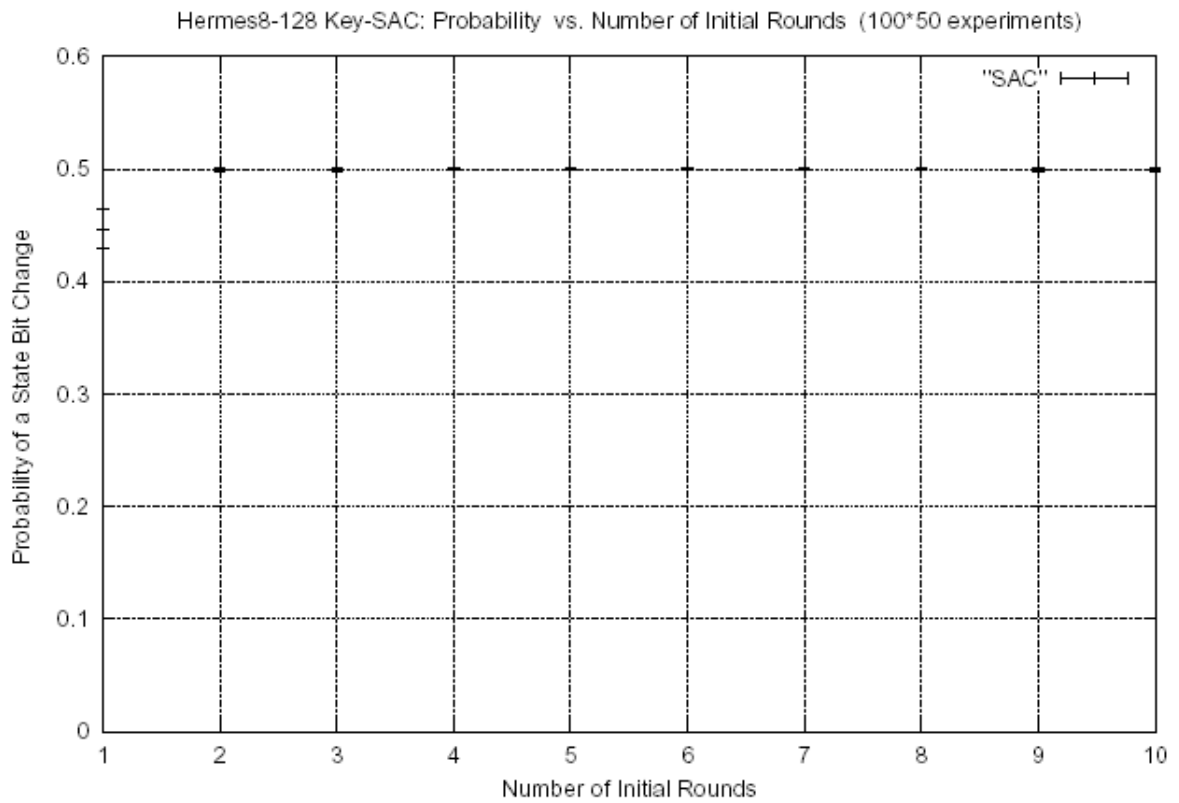


Thu Jun 02 23:31:48 2005

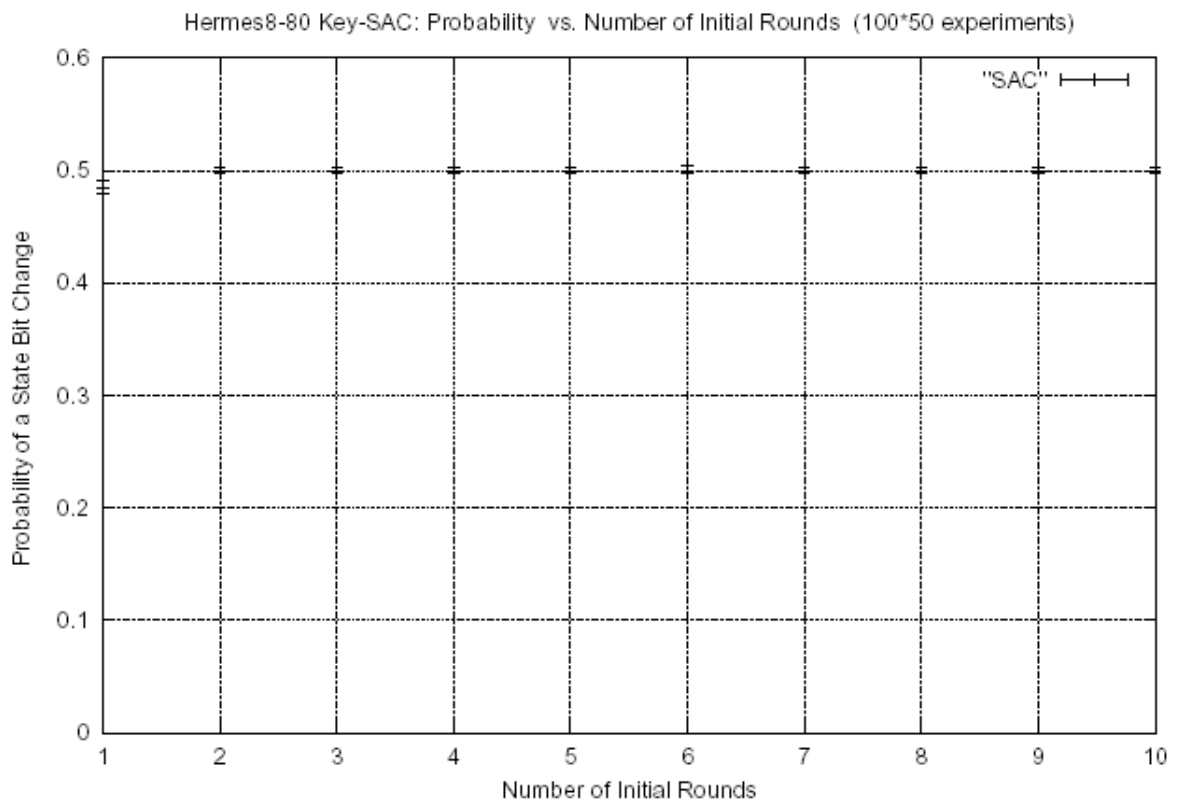


Thu Jun 02 22:42:29 2005

Strict Avalanche Criterion with KEY stimulation



Thu Jun 02 23:39:39 2005



Thu Jun 02 22:31:21 2005

D. Intellectual Property Statement

1. Intellectual property position, royalty policy

- I'm the sole owner of the Hermes8 IP. The development happened during my spare time. My employer was not involved.
- A copyright notice has to be copied together with any copy as usual
- There are no royalties involved.
- However, a later feedback from a 'customer' about the design that uses Hermes8 would be nice, e.g. a nice plate for my office wall {:-}).

2. Update the project

I promise to update this project if the cryptographic community is interested in.