Fast Software Implementations of SC2000

Helger Lipmaa
Helsinki University of Technology
{helger}@tcs.hut.fi
http://www.tcs.hut.fi/~helger

Short Overview

- Flexible design
- $\geq$ four different implementation strategies, incl. large S-boxes
- Our results: up to 1.6x faster by using large S-boxes
- SC2000 is surprisingly fast: in C faster than Twofish in assembly
- Cache size matters

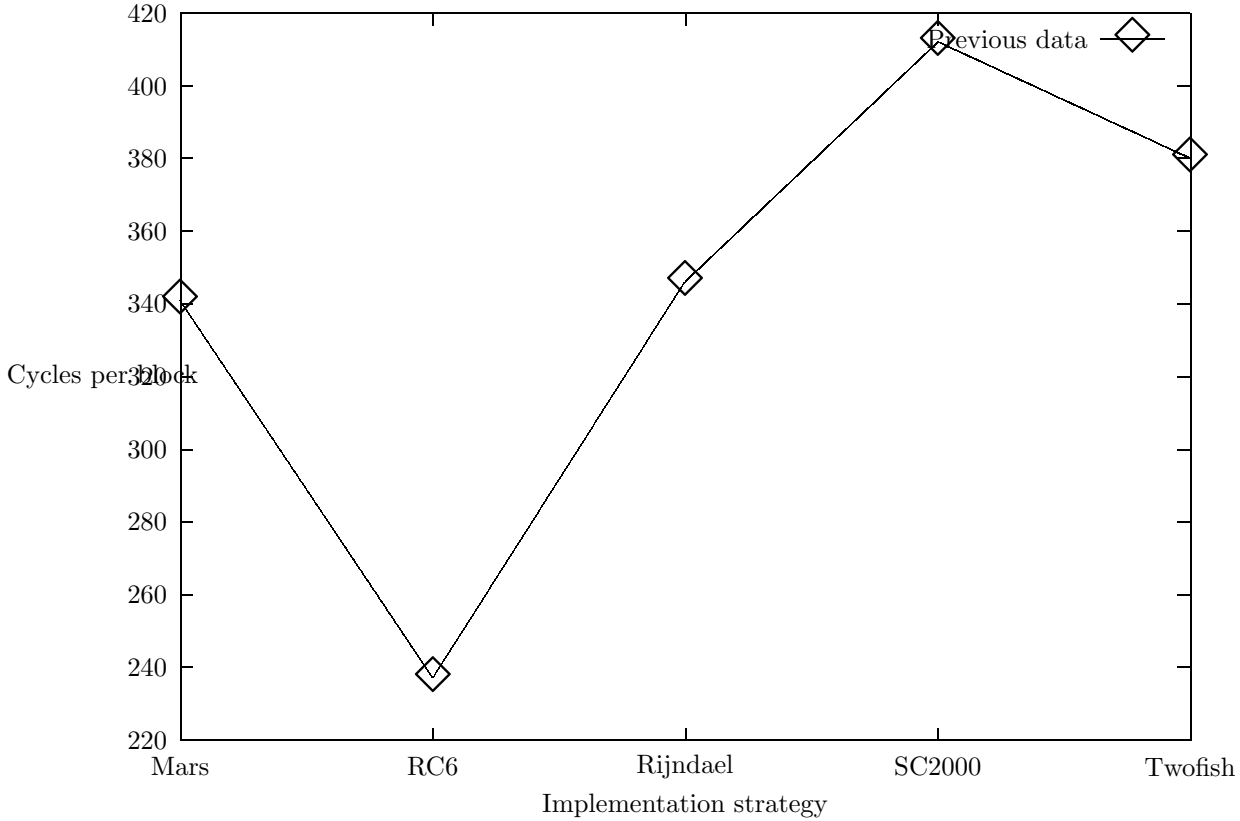
Brief intro to SC2000 (I)

- A block cipher design by Fujitsu
- Designed in 2000, first published in 2001
- Slightly late for the AES process
- Uses the new “hindsight” knowledge, obtain from this process

Brief intro to SC2000 (II)

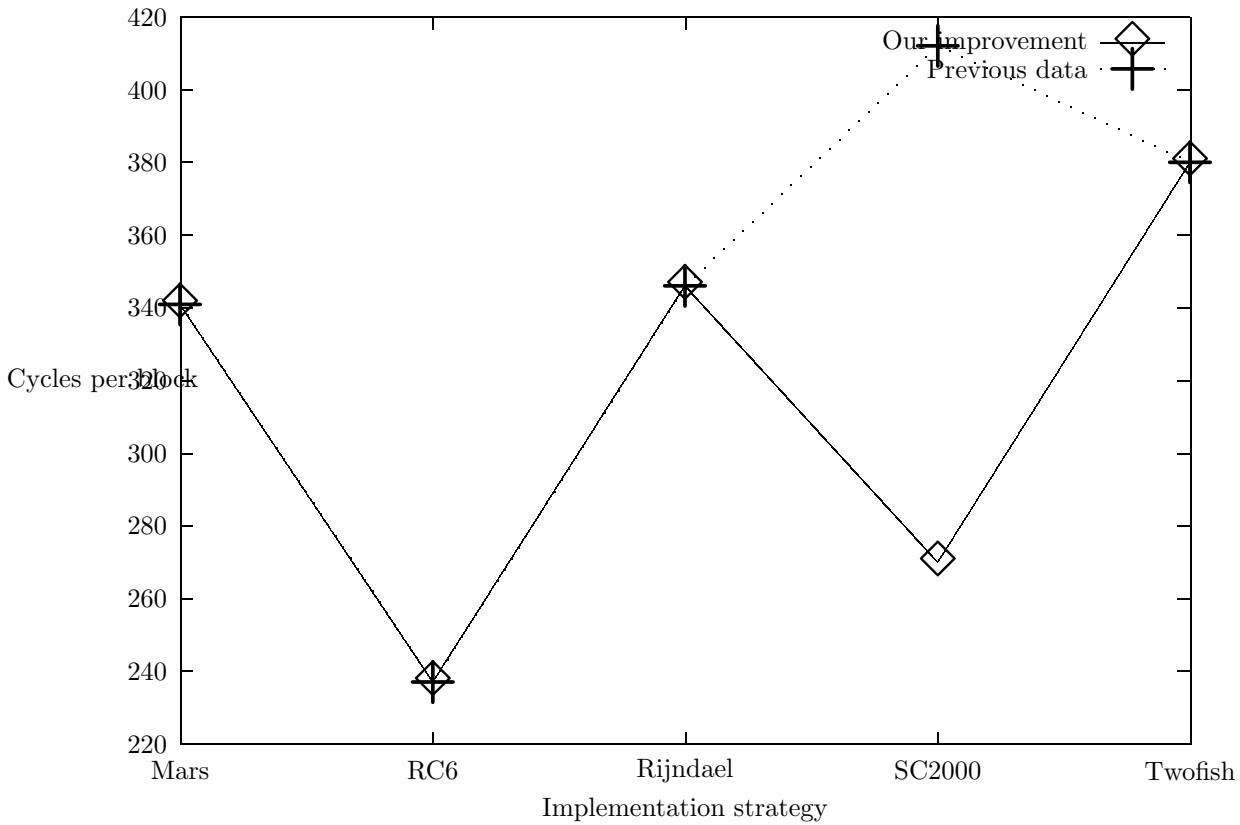
- Submitted to NESSIE, CRYPTREC for evaluation
- Very flexible, many implementation strategies
- Claimed to be very fast — yet no citable numbers about that
- This together with the relative complexity of SC2000 is probably the main reason why SC2000 did not make it to the second round of NESSIE

Previous best implementations



On Pentium III, C implementations

Our main result



On Pentium III, C implementations

Short description of SC2000

- Encryption: Apply the meta-round function $\phi_i = R_i \circ R_i \circ I \circ B \circ I$ six times, $i = 1 \ldots 6$, and then apply $I \circ B \circ I$ to the result.

- Decryption: Apply $\psi_i = R_{i+1} \circ R_{i+1} \circ I \circ B^{(i)} \circ I$ six times, then apply $I \circ B^{(i)} \circ I$ to the result.

- Key schedule: create intermediate keys and then the final key.

- Basic functions: R, I, B, $B^{(i)}$. $R_{2i} = R(0x3333333)$ and $R_{2i+1} = R(0x55555555)$.

- Functions S, M: used internally by R and by the key schedule.

Basic functions

- S: S-box layer (next slide)
- M: MDS matrix
- R: two calls to $SM := S \circ M$ and some Boolean operations
- I: XOR-ing in the key schedule
- B: bitsliced nonlinear function
- $B^{(i)}$: the inverse of B

Overall implementation notes

- Implementation of B and $B^{(i)}$ needs efficient bitslicing

$\star\star$ ≈ 20 Boolean operations

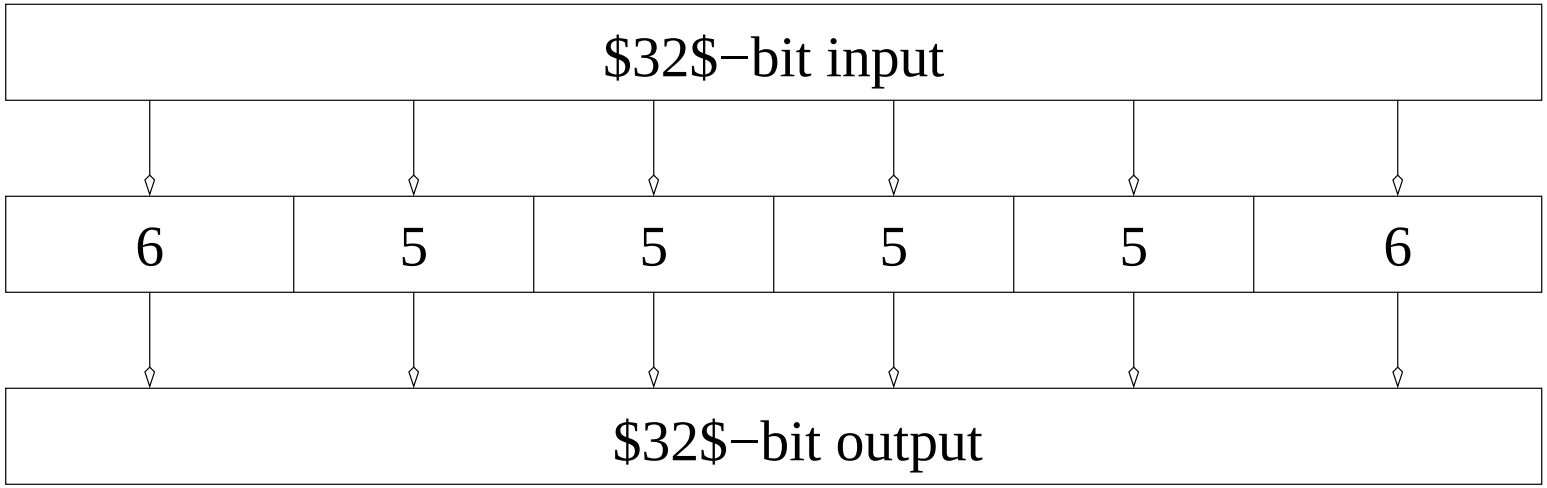
- S and M can combined to a single function $SM = S \circ M$

$\star\star$ SM uses table-lookups to precomputed tables

- Implementation of I: straightforward

- Implementation of R: straightforward using of SM

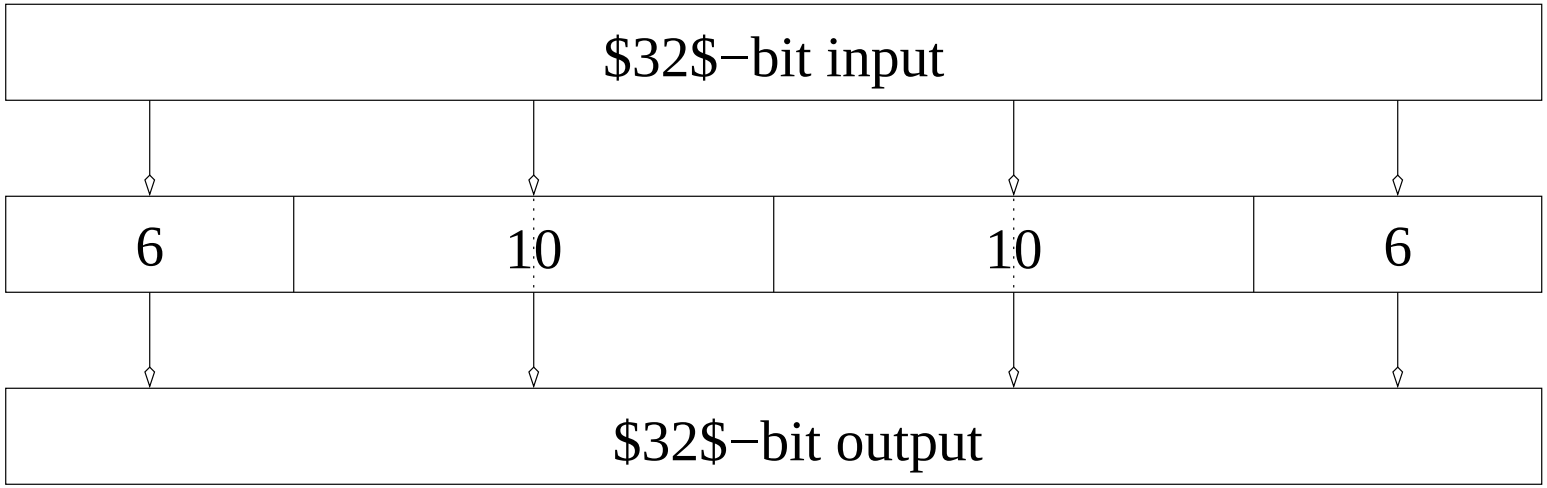
SC2000: S-box strategy (6-5-5-5-5-6)



- Six S-boxes: two 6-bit S-boxes and four 5-bit S-boxes

- 6 table-lookups, memory: $2 \cdot 2^6 + 4 \cdot 2^5 = 256$ cells or 1 KB

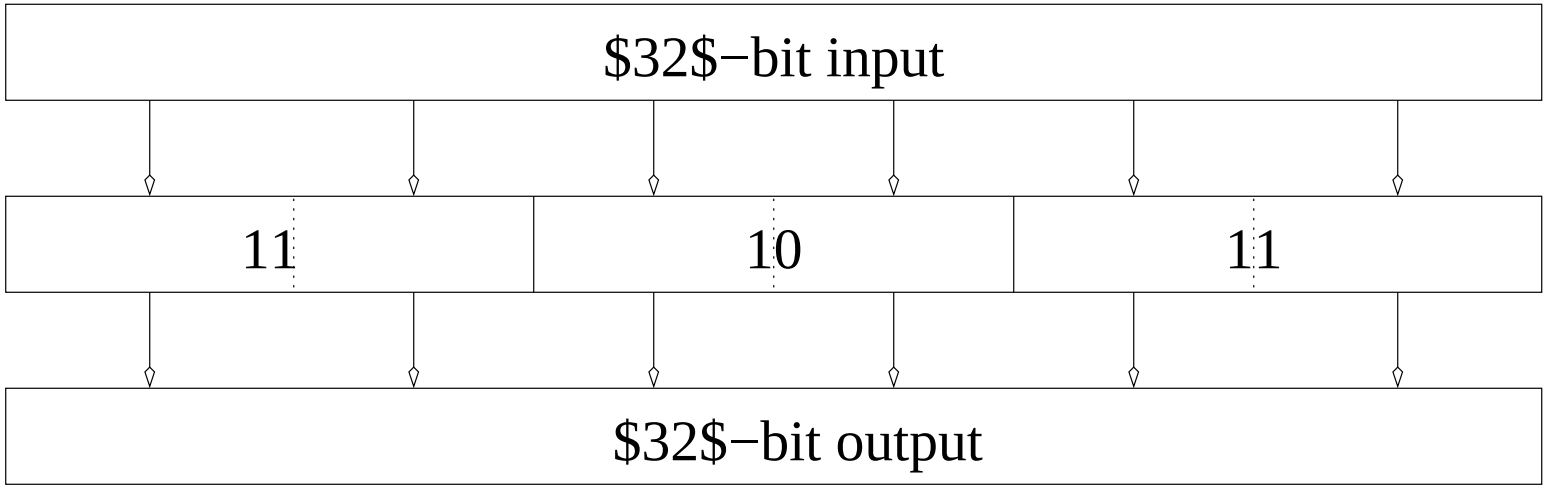
SC2000: S-box strategy (6-10-10-6)



- Four S-boxes: two 6-bit S-boxes and two 10-bit S-boxes

- 4 table-lookups, memory: $2 \cdot 2^6 + 2 \cdot 2^{10} = 2176$ cells or 8.5 KB

SC2000: S-box strategy (11-10-11)



- Three S-boxes: two 11-bit S-boxes and one 10-bit S-boxes

- 3 table-lookups, memory: $2 \cdot 2^{11} + 2^{10} = 5120$ cells or 20 KB

SC2000: S-box strategies

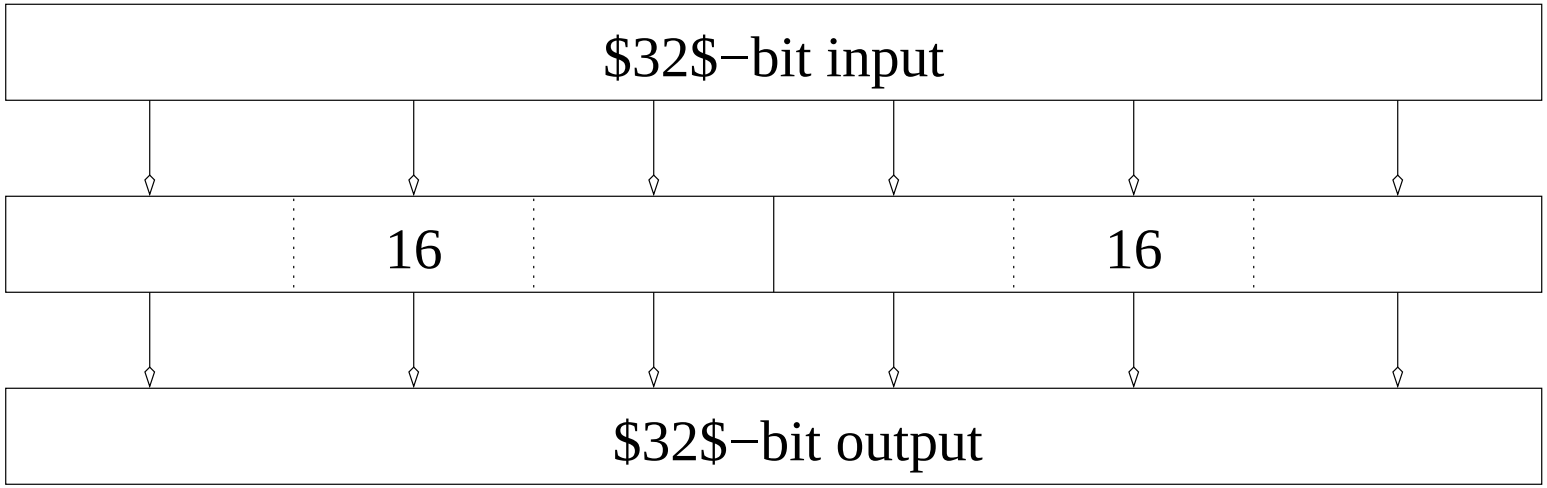
- Best previous result: `412` cycles (C, Pentium III) using the 6-10-10-6 strategy

- Best previous result: `554` cycles (C, Pentium III) using the 11-10-11 strategy

- Small number of table-lookups, reasonable number of used memory

Can you do better?

SC2000: New S-box strategy (16-16)



- Two 16-bit S-boxes

- 2 table-lookups, memory: $2 \cdot 2^{16} = 131072$ cells or 512 KB

SC2000: Why 16-16 matters

- 512 KB is relatively small compared to mainstream RAM amount

⋆⋆ My desktop has 1 GB

- Target processors: Pentium III, Athlon

- Pentium III has 512 KB cache, Athlon has 256 KB cache

- We can expect 16-16 to be faster on P3 than on Athlon

Used environment

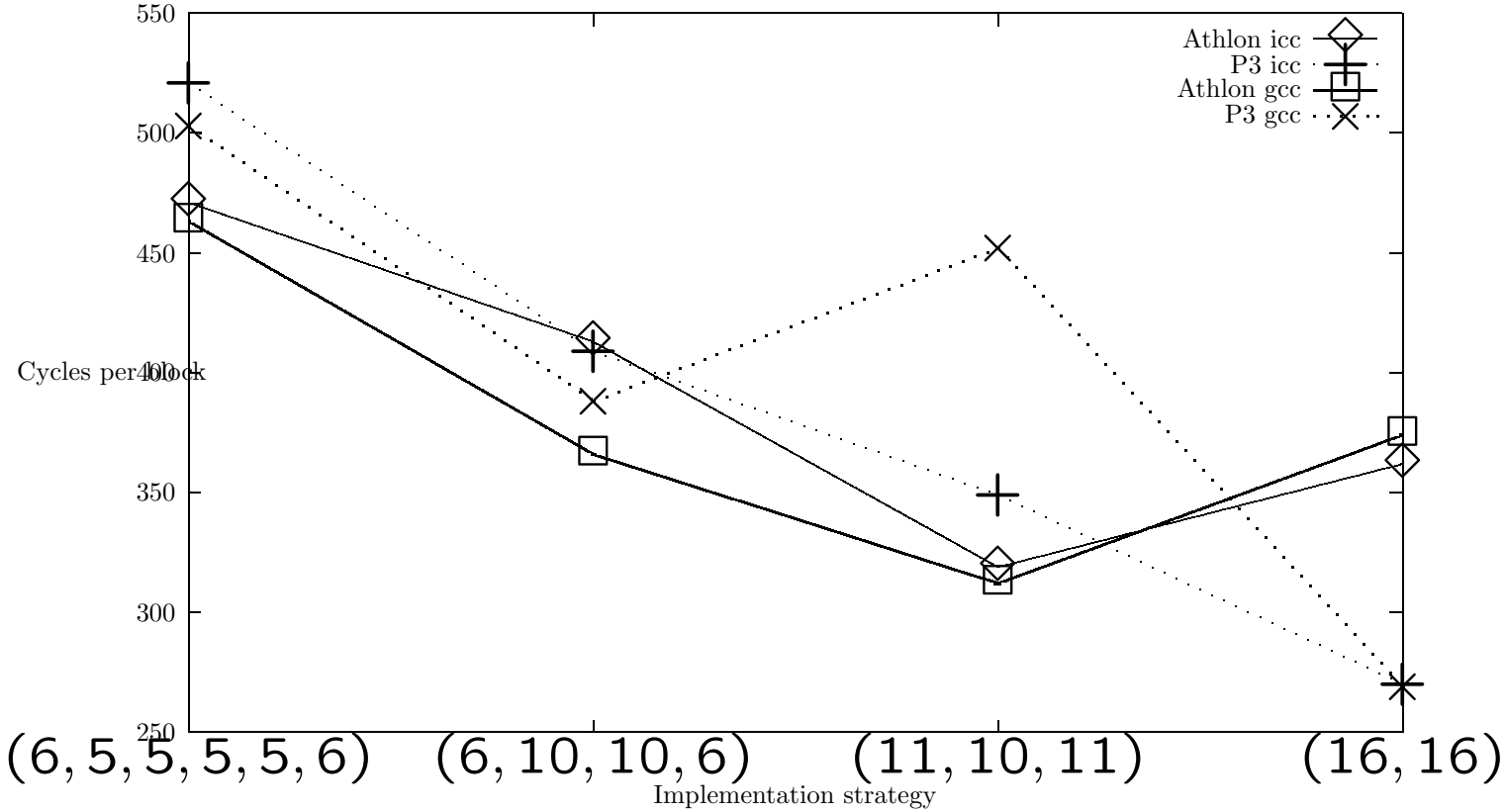
- Processors Pentium III and Athlon

⋆⋆ Compatible, but different cores (optimizations different)

- Compilers gcc 3.0.4 and icc 5.0

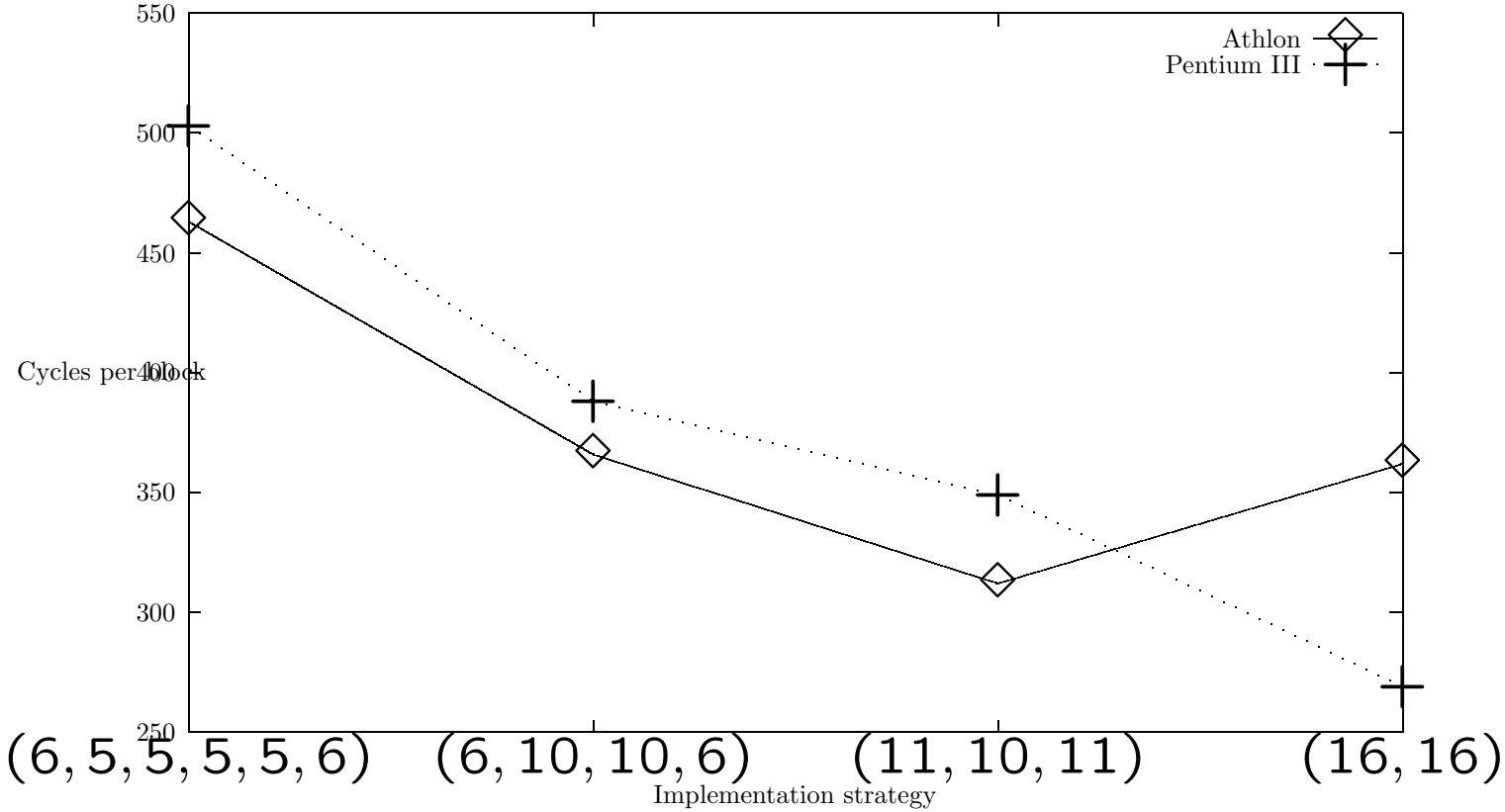
⋆⋆ Both relatively good compilers

Results

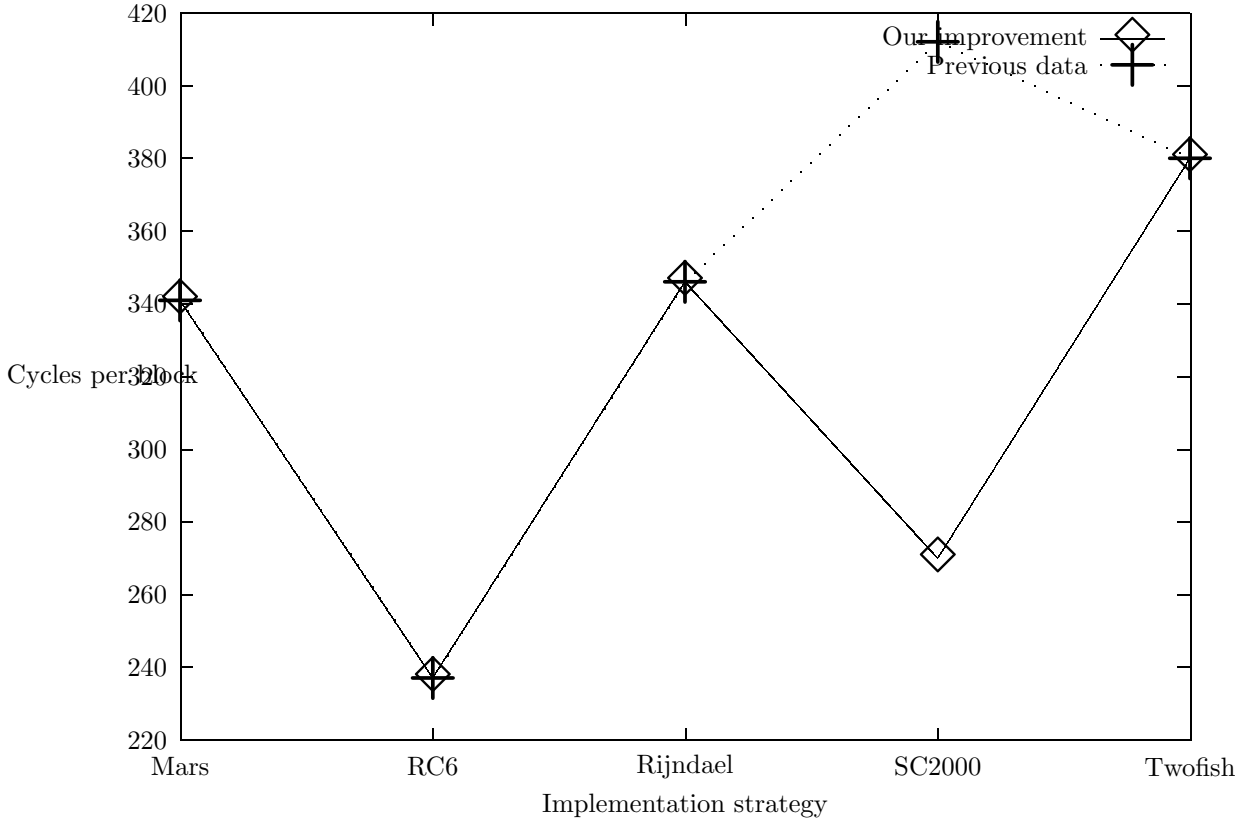


Precise numbers: see proceedings.

Results: P3 versus Athlon

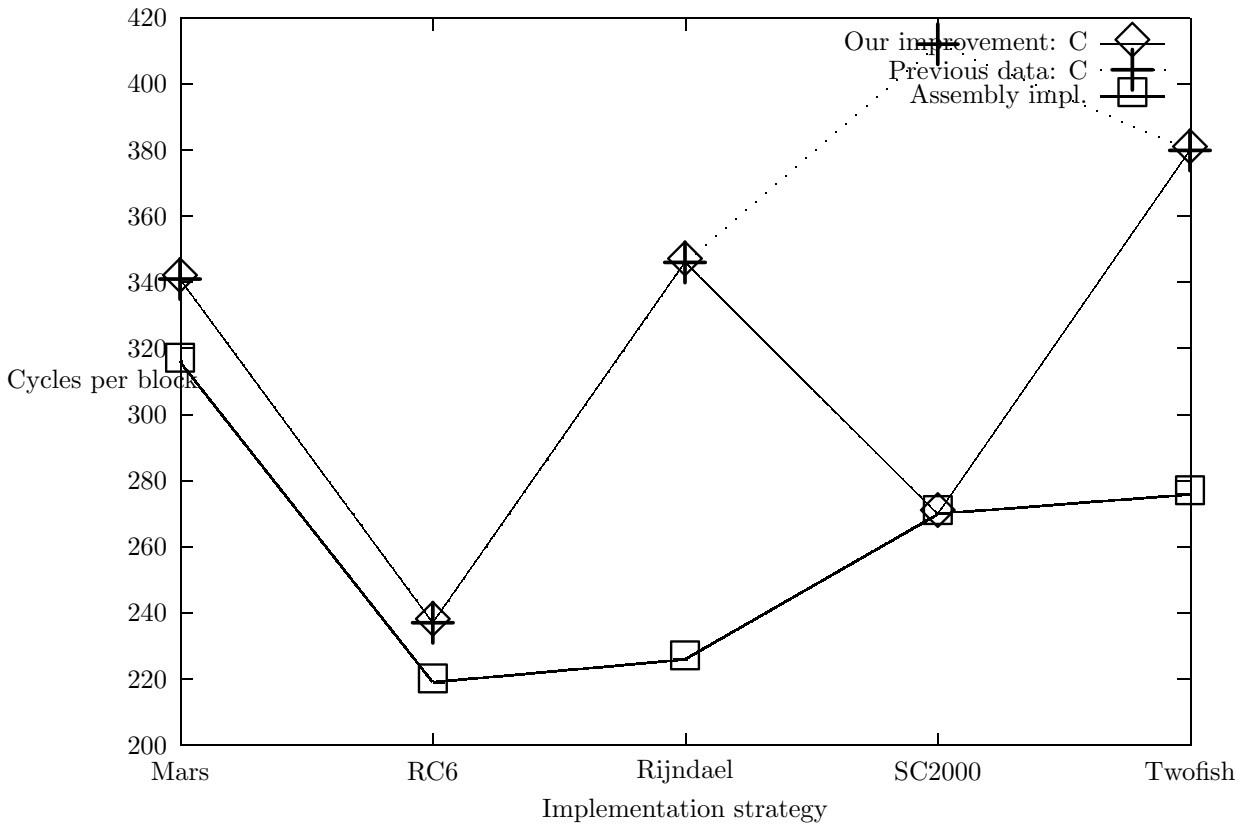


Results: SC2000 vs other ciphers



On Pentium III, C implementations

Results: SC2000 vs other ciphers



On Pentium III, C *and* assembly implementations

Conclusions: SC2000 implementation

- Pentium III: Our 6-10-10-6 is 1.06x faster than best previous, 11-10-11 is 1.59x faster, and 16-16 is 1.3x better than 11-10-11
- Athlon: Our 6-10-10-6 is 1.05x faster than best previous, 11-10-11 is 1.08x faster, and 16-16 is 1.16x *worse* than 11-10-11
- If bulk encryption speed is primary: use P3, icc and 16-16
- If memory is constrained: use Athlon, gcc and 11-10-11
- SC2000 C implementation is faster than the assembly implementation of Twofish!

Conclusions: general

- Use of 6-5-5-5-5-6 S-boxes was an interesting idea that results in many possible strategies. Many tradeoffs between speed and memory
- Cache size matters!
- After the AES process, it is easy to design *fast* ciphers
- We did not analyze the security