PRESENT runs fast: Efficient and Secure Implementation in Software

Tiago Reis, Diego Aranha, Julio López

Institute of Computing
University of Campinas

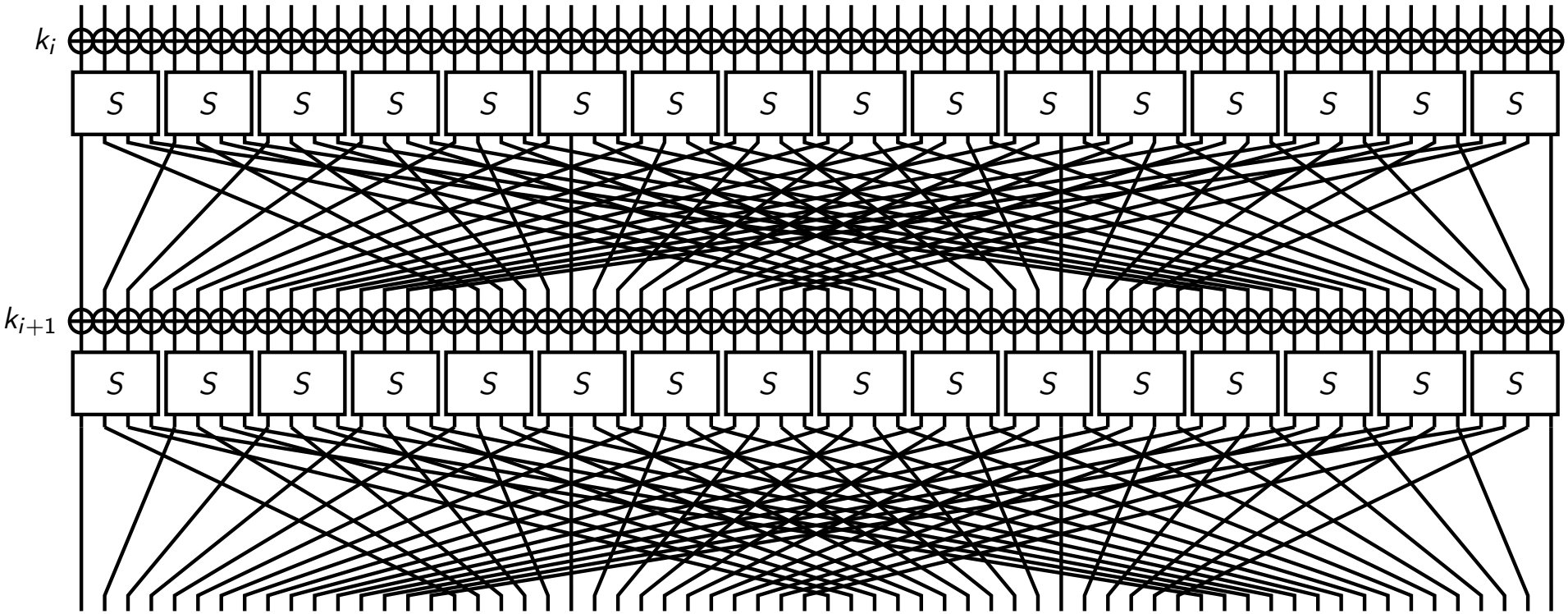
September 28, 2017

Introduction

- Lightweight crypto: focus not only on new designs, but maybe revisit old ones.
- PRESENT has received great deal of cryptanalytic attention.
- Efficiency is the goal, but we have to keep an eye on side-channel security.

The PRESENT block cipher

- Proposed by Bogdanov *et al.* during CHES 2007 as an ultra-lightweight block cipher, with 80-bit and 128-bit key versions, operating on a 64-bit block.
- Substitution-permutation network built over bit permutations: hardware-friendly design, not ideal for software.



The PRESENT block cipher

Input: A 64-bit block of plaintext B, a key K.
Output: A 64-bit block of ciphertext C.

1: $subkey = (subkey_1, subkey_2, ..., subkey_{32}) \leftarrow keySchedule(K)$
2: $C \leftarrow B$
3: **for** $i = 1$ **to** 31 **do**
4: $\quad C \leftarrow C \oplus subkey_i$
5: $\quad C \leftarrow S(C)$
6: $\quad C \leftarrow P(C)$
7: **end for**
8: $C \leftarrow C \oplus subkey_{32}$
9: **return** C

The PRESENT block cipher

The s-box, in hexadecimal notation:

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

The PRESENT block cipher

The s-box, in hexadecimal notation:

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

It can be interpreted as a boolean function from $\{0,1\}^4$ to $\{0,1\}^4$.

```
#define PRESENT_SBOX(x0,x1,x2,x3)          \
    T1 = x2 ^ x1;    T2 = x1 & T1;         \
    T3 = x0 ^ T2;    T5 = x3 ^ T3;         \
    T2 = T1 & T3;    T1 = T1 ^ T5;         \
    T2 = T2 ^ x1;    T4 = x3 | T2;         \
    x2 = T1 ^ T4;    x3 = ~x3;             \
    T2 = T2 ^ x3;    x0 = x2 ^ T2;         \
    T2 = T2 | T1;    x1 = T3 ^ T2;         \
    x3 = T5;
```

The PRESENT block cipher

Permutation P moves the i-th bit of the state to the position $P(i)$:

$$P(i) = \begin{cases} 16i \mod 63, & \text{if } i \neq 63, \\ 63, & \text{if } i = 63. \end{cases}$$

$$B = \begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 & 09 & 10 & 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 \\ 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 & 41 & 42 & 43 & 44 & 45 & 46 & 47 \\ 48 & 49 & 50 & 51 & 52 & 53 & 54 & 55 & 56 & 57 & 58 & 59 & 60 & 61 & 62 & 63 \end{bmatrix},$$

$$P(B) = \begin{bmatrix} 00 & 04 & 08 & 12 & 16 & 20 & 24 & 28 & 32 & 36 & 40 & 44 & 48 & 52 & 56 & 60 \\ 01 & 05 & 09 & 13 & 17 & 21 & 25 & 29 & 33 & 37 & 41 & 45 & 49 & 53 & 57 & 61 \\ 02 & 06 & 10 & 14 & 18 & 22 & 26 & 30 & 34 & 38 & 42 & 46 & 50 & 54 & 58 & 62 \\ 03 & 07 & 11 & 15 & 19 & 23 & 27 & 31 & 35 & 39 & 43 & 47 & 51 & 55 & 59 & 63 \end{bmatrix}.$$

The PRESENT block cipher

Permutation P moves the i-th bit of the state to the position $P(i)$:

$$P(i) = \begin{cases} 16i \mod 63, & \text{if } i \neq 63, \\ 63, & \text{if } i = 63. \end{cases}$$

$$B = \begin{bmatrix} 00 & 01 & 02 & 03 \\ & & & \end{bmatrix},$$

$$P(B) = \begin{bmatrix} 00 \\ 01 \\ 02 \\ 03 \end{bmatrix}.$$

The PRESENT block cipher

Permutation P moves the i-th bit of the state to the position $P(i)$:

$$P(i) = \begin{cases} 16i \mod 63, & \text{if } i \neq 63, \\ 63, & \text{if } i = 63. \end{cases}$$

$$B = \begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 & 06 & 07 \\ & & & & & & & \end{bmatrix},$$

$$P(B) = \begin{bmatrix} 00 & 04 \\ 01 & 05 \\ 02 & 06 \\ 03 & 07 \end{bmatrix}.$$

The PRESENT block cipher

Permutation P moves the i-th bit of the state to the position $P(i)$:

$$P(i) = \begin{cases} 16i \mod 63, & \text{if } i \neq 63, \\ 63, & \text{if } i = 63. \end{cases}$$

$$B = \begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 & 09 & 10 & 11 \\ & & & & & & & & & & & \\ & & & & & & & & & & & \end{bmatrix},$$

$$P(B) = \begin{bmatrix} 00 & 04 & 08 \\ 01 & 05 & 09 \\ 02 & 06 & 10 \\ 03 & 07 & 11 \end{bmatrix}.$$

The PRESENT block cipher

Permutation P moves the i-th bit of the state to the position $P(i)$:

$$P(i) = \begin{cases} 16i \mod 63, & \text{if } i \neq 63, \\ 63, & \text{if } i = 63. \end{cases}$$

$$B = \begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 & 09 & 10 & 11 & 12 & 13 & 14 & 15 \\ & & & & & & & & & & & & & & & \end{bmatrix},$$

$$P(B) = \begin{bmatrix} 00 & 04 & 08 & 12 \\ 01 & 05 & 09 & 13 \\ 02 & 06 & 10 & 14 \\ 03 & 07 & 11 & 15 \end{bmatrix}.$$

The PRESENT block cipher

Permutation P moves the i-th bit of the state to the position $P(i)$:

$$P(i) = \begin{cases} 16i \mod 63, & \text{if } i \neq 63, \\ 63, & \text{if } i = 63. \end{cases}$$

$$B = \begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 & 09 & 10 & 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 \\ 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 & 41 & 42 & 43 & 44 & 45 & 46 & 47 \\ 48 & 49 & 50 & 51 & 52 & 53 & 54 & 55 & 56 & 57 & 58 & 59 & 60 & 61 & 62 & 63 \end{bmatrix},$$

$$P(B) = \begin{bmatrix} 00 & 04 & 08 & 12 & 16 & 20 & 24 & 28 & 32 & 36 & 40 & 44 & 48 & 52 & 56 & 60 \\ 01 & 05 & 09 & 13 & 17 & 21 & 25 & 29 & 33 & 37 & 41 & 45 & 49 & 53 & 57 & 61 \\ 02 & 06 & 10 & 14 & 18 & 22 & 26 & 30 & 34 & 38 & 42 & 46 & 50 & 54 & 58 & 62 \\ 03 & 07 & 11 & 15 & 19 & 23 & 27 & 31 & 35 & 39 & 43 & 47 & 51 & 55 & 59 & 63 \end{bmatrix}.$$

Implementation strategies

- Two usual strategies: using large lookup tables to merge permutations and s-boxes; bitslicing.
- Large tables open vulnerabilities to exploits using side-channel leakage and lead to high memory usage.
- Bitsliced ciphers are hard to use in practice. Demand for specific situations and mode of operation.
- Our proposals: first, interchange permutations and s-boxes; second, decompose the permutations.

Our proposal for PRESENT encryption

Instead of applying S and then P:

$$S(B) = \left(\begin{array}{|cccc|cccc|cccc|cccc|} \hline 00 & 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 & 09 & 10 & 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 \\ 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 & 41 & 42 & 43 & 44 & 45 & 46 & 47 \\ 48 & 49 & 50 & 51 & 52 & 53 & 54 & 55 & 56 & 57 & 58 & 59 & 60 & 61 & 62 & 63 \\ \hline \end{array}\right)$$

Our proposal for PRESENT encryption

Instead of applying S and then P:

$$S(B) = \left(\begin{array}{|cccc|cccc|cccc|cccc|} \hline 00 & 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 & 09 & 10 & 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 \\ 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 & 41 & 42 & 43 & 44 & 45 & 46 & 47 \\ 48 & 49 & 50 & 51 & 52 & 53 & 54 & 55 & 56 & 57 & 58 & 59 & 60 & 61 & 62 & 63 \\ \hline \end{array}\right)$$

$$P(S(B)) = \left(\begin{array}{|cccccccccccccccc|} \hline 00 & 04 & 08 & 12 & 16 & 20 & 24 & 28 & 32 & 36 & 40 & 44 & 48 & 52 & 56 & 60 \\ 01 & 05 & 09 & 13 & 17 & 21 & 25 & 29 & 33 & 37 & 41 & 45 & 49 & 53 & 57 & 61 \\ 02 & 06 & 10 & 14 & 18 & 22 & 26 & 30 & 34 & 38 & 42 & 46 & 50 & 54 & 58 & 62 \\ 03 & 07 & 11 & 15 & 19 & 23 & 27 & 31 & 35 & 39 & 43 & 47 & 51 & 55 & 59 & 63 \\ \hline \end{array}\right)$$

Our proposal for PRESENT encryption

We apply P and then $\boldsymbol{S_{BS}}$:

$$P(B) = \begin{pmatrix} 00 & 04 & 08 & 12 & 16 & 20 & 24 & 28 & 32 & 36 & 40 & 44 & 48 & 52 & 56 & 60 \\ 01 & 05 & 09 & 13 & 17 & 21 & 25 & 29 & 33 & 37 & 41 & 45 & 49 & 53 & 57 & 61 \\ 02 & 06 & 10 & 14 & 18 & 22 & 26 & 30 & 34 & 38 & 42 & 46 & 50 & 54 & 58 & 62 \\ 03 & 07 & 11 & 15 & 19 & 23 & 27 & 31 & 35 & 39 & 43 & 47 & 51 & 55 & 59 & 63 \end{pmatrix}$$

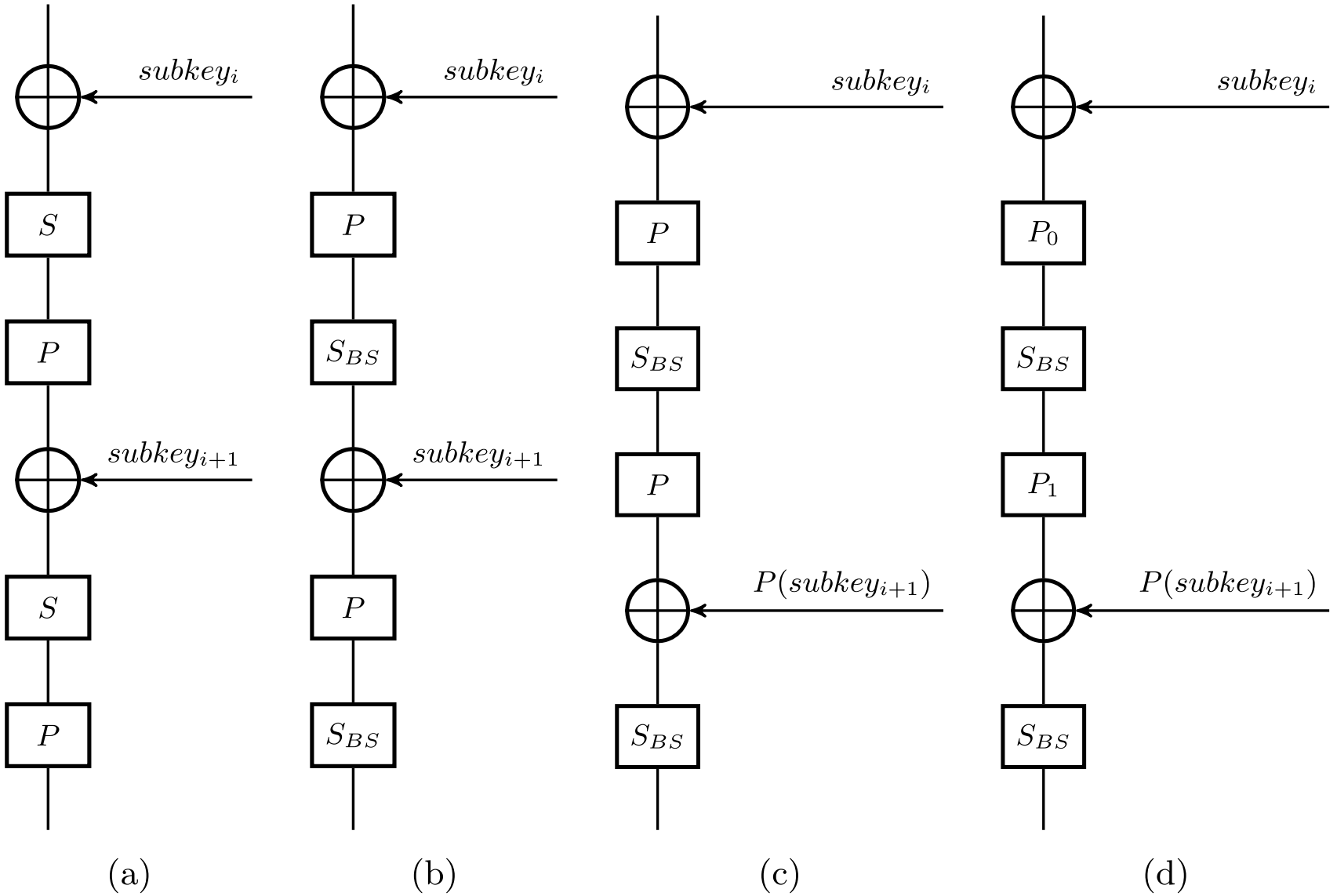
Our proposal for PRESENT encryption

We apply P and then $\boldsymbol{S_{BS}}$:

$$P(B) = \begin{pmatrix} 00 & 04 & 08 & 12 & 16 & 20 & 24 & 28 & 32 & 36 & 40 & 44 & 48 & 52 & 56 & 60 \\ 01 & 05 & 09 & 13 & 17 & 21 & 25 & 29 & 33 & 37 & 41 & 45 & 49 & 53 & 57 & 61 \\ 02 & 06 & 10 & 14 & 18 & 22 & 26 & 30 & 34 & 38 & 42 & 46 & 50 & 54 & 58 & 62 \\ 03 & 07 & 11 & 15 & 19 & 23 & 27 & 31 & 35 & 39 & 43 & 47 & 51 & 55 & 59 & 63 \end{pmatrix}$$

$$S_{BS}(P(B)) = \begin{pmatrix} 00 & 04 & 08 & 12 & 16 & 20 & 24 & 28 & 32 & 36 & 40 & 44 & 48 & 52 & 56 & 60 \\ 01 & 05 & 09 & 13 & 17 & 21 & 25 & 29 & 33 & 37 & 41 & 45 & 49 & 53 & 57 & 61 \\ 02 & 06 & 10 & 14 & 18 & 22 & 26 & 30 & 34 & 38 & 42 & 46 & 50 & 54 & 58 & 62 \\ 03 & 07 & 11 & 15 & 19 & 23 & 27 & 31 & 35 & 39 & 43 & 47 & 51 & 55 & 59 & 63 \end{pmatrix}$$

Our proposal for PRESENT encryption



Our proposal for PRESENT encryption

Input: A 64-bit block of plaintext B, a key K.
Output: A 64-bit block of ciphertext C.

1: $subkey = (subkey_1, subkey_2, ..., subkey_{32}) \leftarrow keySchedule(K)$
2: $C \leftarrow B$
3: **for** $i = 1$ **to** 15 **do**
4: $\quad C \leftarrow C \oplus subkey_{2i-1}$
5: $\quad C \leftarrow P_0(C)$
6: $\quad C \leftarrow S_{BS}(C)$
7: $\quad C \leftarrow P_1(C)$
8: $\quad C \leftarrow C \oplus P(subkey_{2i})$
9: $\quad C \leftarrow S_{BS}(C)$
10: **end for**
11: $C \leftarrow C \oplus subkey_{31}$
12: $C \leftarrow P(C)$
13: $C \leftarrow S_{BS}(C)$
14: $C \leftarrow C \oplus subkey_{32}$
15: **return** C

Our proposal for PRESENT encryption

$$B = \begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 & 09 & 10 & 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 \\ 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 & 41 & 42 & 43 & 44 & 45 & 46 & 47 \\ 48 & 49 & 50 & 51 & 52 & 53 & 54 & 55 & 56 & 57 & 58 & 59 & 60 & 61 & 62 & 63 \end{bmatrix},$$

$$P_0(B) = \begin{bmatrix} 00 & 16 & 32 & 48 & 04 & 20 & 36 & 52 & 08 & 24 & 40 & 56 & 12 & 28 & 44 & 60 \\ 01 & 17 & 33 & 49 & 05 & 21 & 37 & 53 & 09 & 25 & 41 & 57 & 13 & 29 & 45 & 61 \\ 02 & 18 & 34 & 50 & 06 & 22 & 38 & 54 & 10 & 26 & 42 & 58 & 14 & 30 & 46 & 62 \\ 03 & 19 & 35 & 51 & 07 & 23 & 39 & 55 & 11 & 27 & 43 & 59 & 15 & 31 & 47 & 63 \end{bmatrix},$$

$$P_1(B) = \begin{bmatrix} 00 & 01 & 02 & 03 & 16 & 17 & 18 & 19 & 32 & 33 & 34 & 35 & 48 & 49 & 50 & 51 \\ 04 & 05 & 06 & 07 & 20 & 21 & 22 & 23 & 36 & 37 & 38 & 39 & 52 & 53 & 54 & 55 \\ 08 & 09 & 10 & 11 & 24 & 25 & 26 & 27 & 40 & 41 & 42 & 43 & 56 & 57 & 58 & 59 \\ 12 & 13 & 14 & 15 & 28 & 29 & 30 & 31 & 44 & 45 & 46 & 47 & 60 & 61 & 62 & 63 \end{bmatrix}.$$

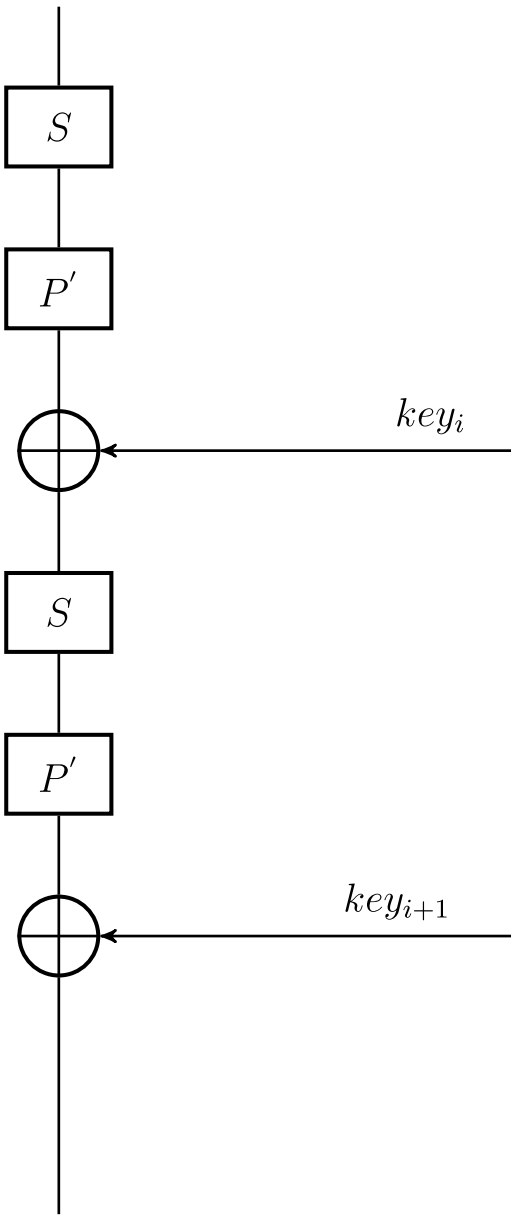
Our proposal for PRESENT encryption

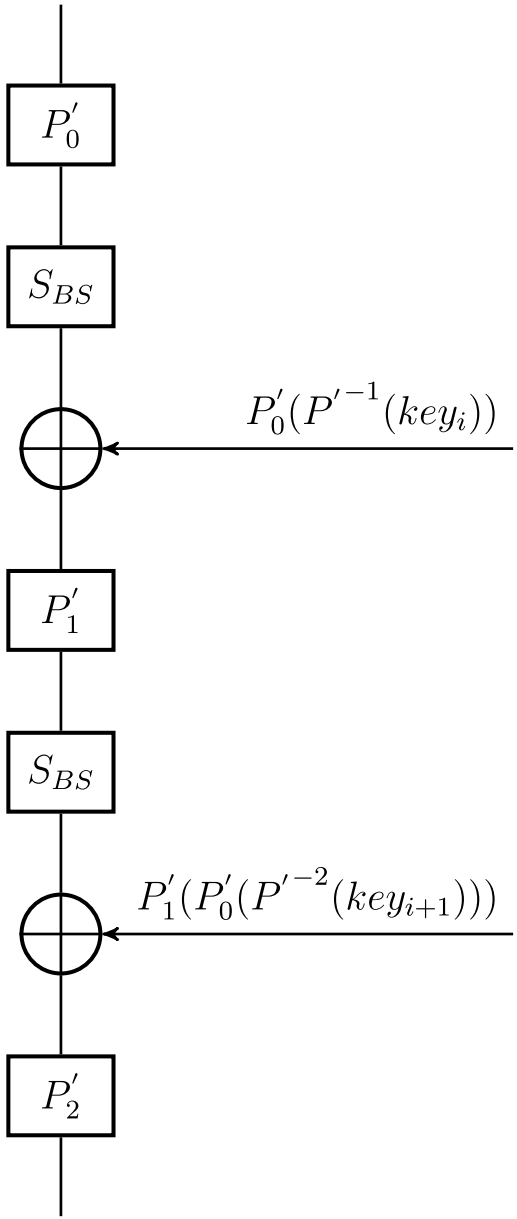
- S_{BS} is the s-box calculated in a bitsliced fashion.
- Both the s-boxes are calculated over a bitsliced state relatively to the original.
- Permutation P_0 and P_1 show these properties: $P_1 \circ P_0 = P^2$, $P_0 \circ P_1 = P^{-1}$, $P_0^{-1} = P_0$, $P_1^{-1} = P_1$.
- Both P_0 and P_1 can be implemented in 16 clock cycles using ARM instructions. P requires 28 cycles.
- Downside: the key-schedule undergoes one extra permutation every two rounds.

Generalizing the technique

- In an arbitrary algorithm, if we have interleaved permutations and s-boxes, the state size is a multiple of the s-box input size and this s-box is applied in parallel over all bits, we can generalize this strategy.
- Example: GIFT block cipher.
- It might not always render a performance improvement, since the decomposition of permutations can result in less software-friendly operations.
- The upside is that the s-box can always be computed efficiently and isochronously.

Generalizing the technique





Generalizing the technique

The permutations in the previous scheme are given by:

$$P_0^{'} := P_0;$$

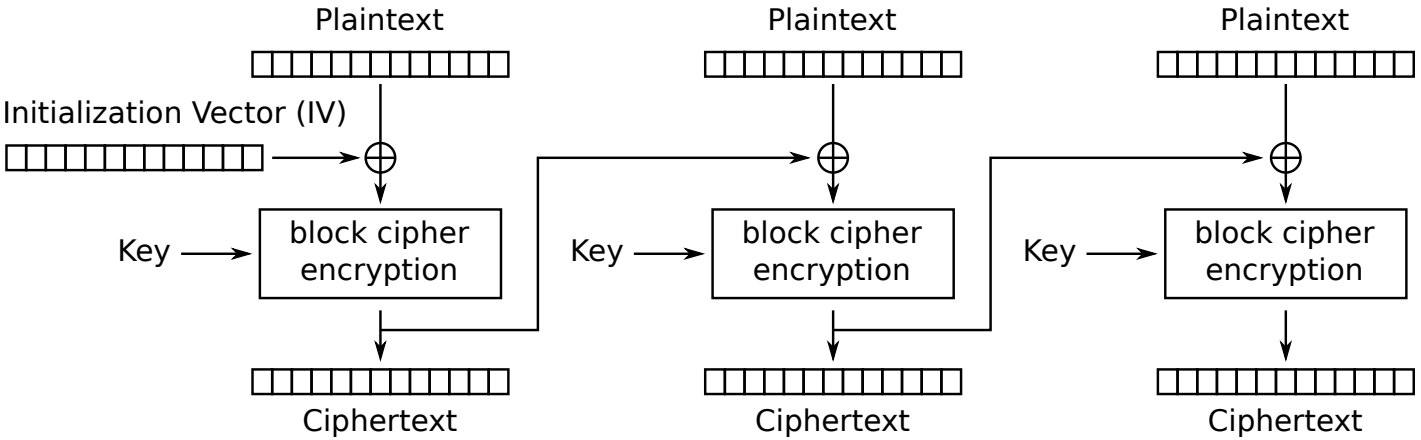
$$P_1^{'} := P \circ P^{'} \circ {P_0^{'}}^{-1};$$

$$P_2^{'} := {P^{'}}^2 \circ {P_0^{'}}^{-1} \circ {P_1^{'}}^{-1}.$$

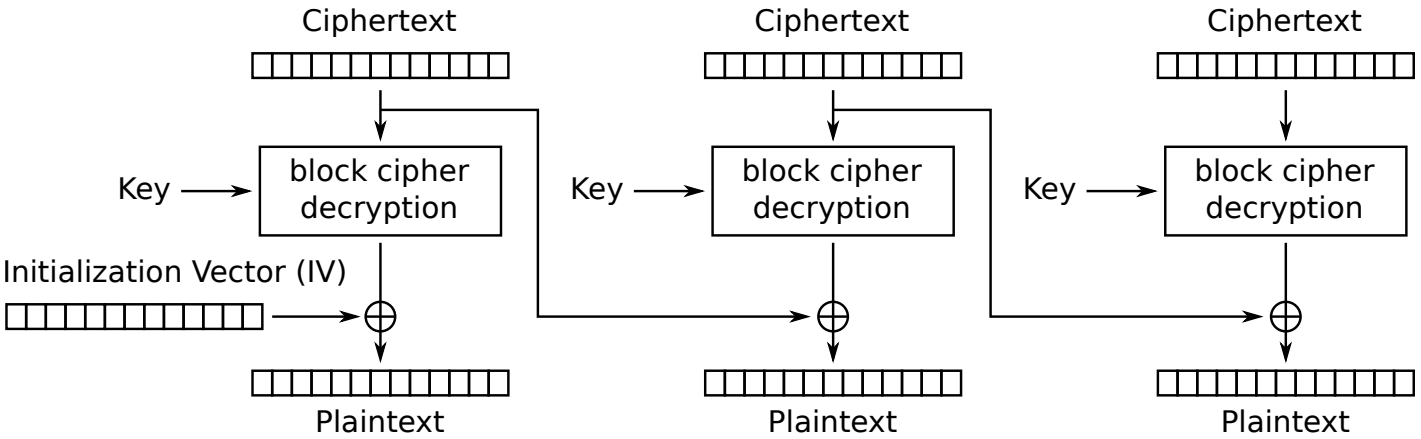
Implementation results

- We implemented this block cipher on platforms with ARM Cortex-M and Cortex-A processors, using CBC and CTR modes of operation.
- Depending on the mode used, we can encrypt up to two blocks in parallel, for 32-bit architecture. For 64-bit, four blocks. With ARM-NEON 128-bit registers, up to eight blocks in parallel.
- We achieved very relevant speedup over the state-of-the-art results for these microprocessors.

CBC mode of operation

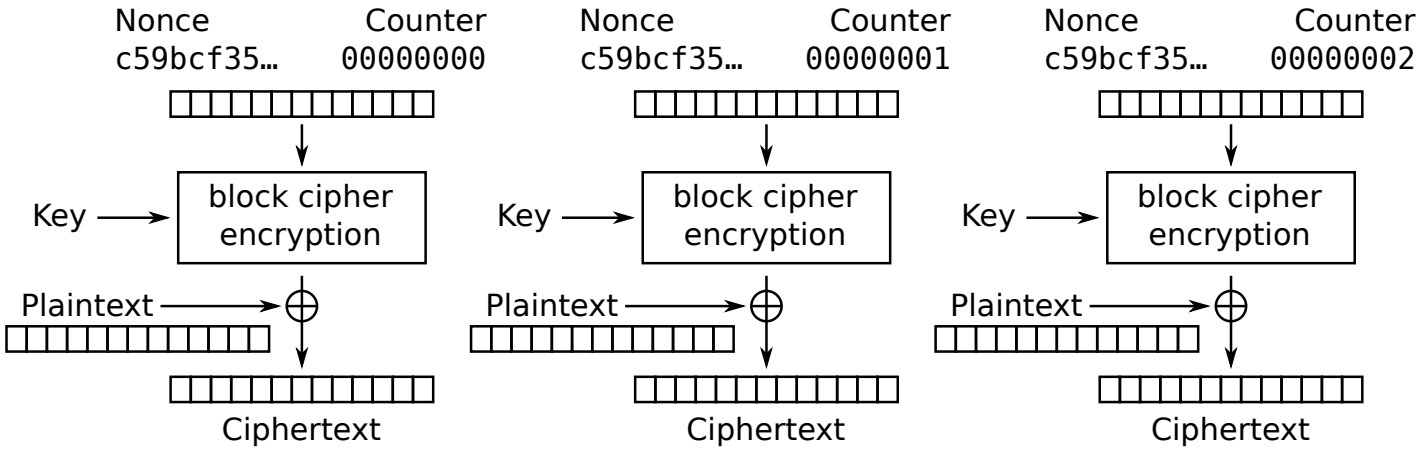


Cipher Block Chaining (CBC) mode encryption

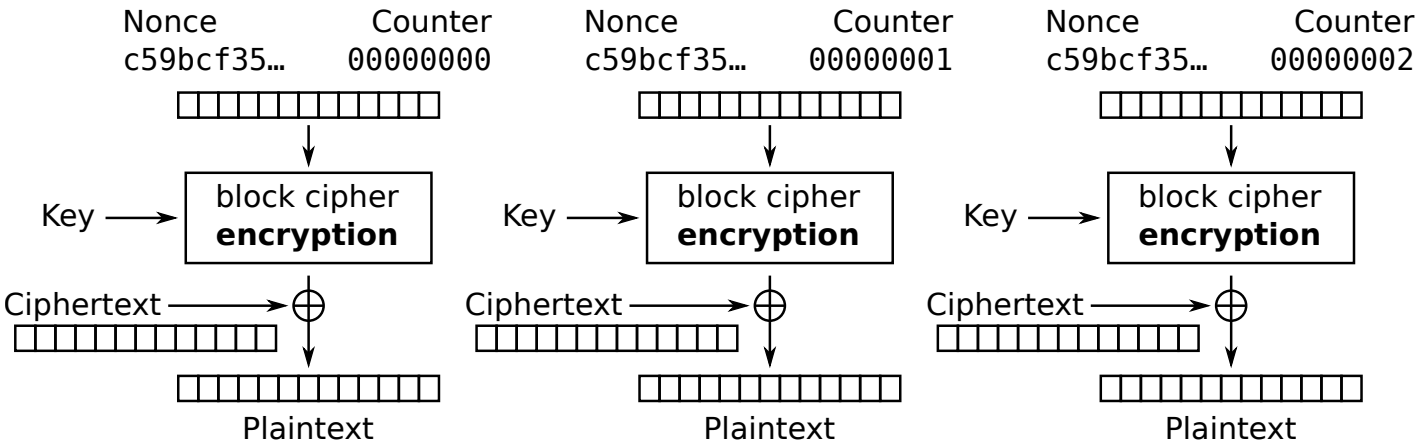


Cipher Block Chaining (CBC) mode decryption

CTR mode of operation



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Implementation results

Isochronous implementation of 128-bit encryption using CTR mode of operation, assuming precomputed subkeys:

Processor	Code size [bytes]	Execution time [cycles]
Cortex-M0+	2524	3183
Cortex-M3	2476	2116
Cortex-M4	2612	1599
Cortex-A7	2456	1708
Cortex-A15	2456	960

Implementation results

Comparing our implementation of PRESENT versus best previously published work, encrypting 128 bits on a Cortex-M3, assuming precomputed subkeys:

Implementation	Code size [bytes]	Execution time [cycles]
Dinu *et al.*'s work	3568	16786
Our work	2476	2116

Implementation results

Comparing PRESENT's performance versus P. Schwabe and K. Stofellen's implementation of AES, encrypting 128 bits in CTR mode:

Implementation	Code size [bytes]	Execution time [cycles]
AES on Cortex-M3	12120	1617
PRESENT on Cortex-M3	2476	2116
AES on Cortex-M4	12120	1618
PRESENT on Cortex-M4	2612	1599

Implementation results

- For our Cortex-A processors, NEON instructions such as `VTBL` suggest that the s-box could be realized as a lookup table, but our version still performs better.
- We also applied second-order masking to PRESENT, as W. de Groot *et al.* and compared with their work. Although the implementations are not fully compatible, for our scenario of encrypting 128 bits with precomputed keys, our version performs 15% faster.

Conclusion

- We devised a method to implement SPNs that follow a particular structure that might provide substantial performance improvements.
- For PRESENT, the case we worked on, we were able to achieve a speedup factor close to 8 comparing to state-of-the-art results.
- Using our technique, PRESENT becomes competitive in software compared to widely used block ciphers such as AES.

Acknowledgments

We would like to thank LG Electronics for funding the research.

Acknowledgments

We would like to thank LG Electronics for funding the research.

And thank you for your attention!