PRESENT-80

An Ultra Lightweight Block Cipher

Lightweight Cryptography

- Provides security solutions for *resource-constrained* (resource-limited) IoT devices, with less memory, less computing resources and less power supply.
  - Smart meters,
  - vehicle security,
  - wireless patient monitoring systems,
  - Intelligent Transport Systems,
  - Internet of Things
  - RFID tags,
  - sensors,
  - Contactless smart card,
  - Health-care device,
  - etc.
- Is expected to be *simpler* and *faster* compared to conventional cryptography.

Confusion and Diffusion

- Are two methods for frustrating statistical cryptanalysis.
- **Confusion** *hides the relationship between the ciphertext and the key* and is used in block and stream cyphers.
  - A single bit change in a key should affect most or all bits in the ciphertext
- **Diffusion** *hides the relationship between the ciphertext and the plaintext* and is used in block cyphers only.
  - A single bit change in the plaintext, (statistically) half of the ciphertext bits should change.
- In Shannon's original definition,
  - **confusion** refers to making the relationship between the ciphertext and the symmetric key as complex and involved as possible.
  - **Diffusion** refers to dissipating the statistical structure of a plaintext over the bulk of ciphertext.
- For block cyphers or cryptographic hash functions, *the degree of confusion and diffusion is assessed through the* ***avalanche effect***.

The Avalanche Effect

- In an ideal encryption algorithm, a small change in the plaintext or the key significantly changes the ciphertext.

- The S-box in PRESENT was designed that
  - Improves the avalanche property
  - Resists various attacks, e.g.,
    - differential and linear cryptanalysis attacks
    - Structural attacks
    - Algebraic attacks
    - Key schedule attacks

Block size (B) considerations

- Typically, the block size B of a block cypher is **B ≥ 64**.
- Compact, lightweight cyphers, such as the PRESENT block cypher, have smaller block sizes of 64 bits.
  - Require high implementation efficiency
  - Can accommodate lower security levels
- Very small block sizes (such as 32 bits) would make a *dictionary attack* practical
  - The attacker may acquire some known plaintext/ciphertext pairs and build a table of plaintext-ciphertext mappings for a system with a given security key.

Key Size (K) Considerations

- *K* must be large enough to ensure a *brute force* or *exhaustive* key search attack is impossible.
  - In such an attack, the attacker can use knowledge of some plaintext/ciphertext pairs to encrypt the plaintext with all possible keys to determine which key results in the corresponding ciphertext.
  - Using only a modest number of plaintext/ciphertext pairs, the key found can be confirmed to be the correct one.
- To prevent *brute force* or *exhaustive key search attacks*, generally, K ≥ 80

Algorithm	Block size (B bits)	Key size (K bits)	Structure	No of rounds	Year
AES	128	128/192/256	SPN	10/12/14	1998
DES	64	56	Feistel	16	1977
PRESENT	64	80/128	SPN	31	2007

Lightweight Block Cipher Standards

- ISO/IEC 29192-2:2019

	Block size	Key size	Year standardised
PRESENT	64	80, 128	2012
CLEFIA	128	128, 192, 256	2012
LEA	128	128, 192, 256	2019

- ISO/IEC 29167-21:2018 (SIMON for RFID)
- ISO/IEC 29167-22:2018 (SPECK for RFID)

	Block size	Key size	Optimized for
SIMON	32, 48, 64, 96, 128	64, 72, 96, 128, 144, 192, 256	Hardware implementation
SPECK	32, 48, 64, 96, 128	64, 72, 96, 128, 144, 192, 256	Software implementation

- There are also many other lightweight block ciphers proposed from academia and industry, a good survey can be found in reference [3] at the end of the slides.

List of ciphers covered in the survey.

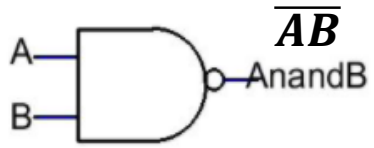
Algorithm	Key size	Block size	Structure	No. of rounds
AES (Daemen and Rijmen, 1998)	128/192/256	128	SPN	10/12/14
ARMADILLO (Badel et al., 2010)	128/160/184/192/256	64/128/192/240/288/384	SPN	Variable
Blowfish (Schneier, 1994)	32-448	64	Feistel	16
Camellia (Aoki et al., 2001)	128/192/256	128	Feistel	18/24
Cast5 (Adams, 1997)	40-128	64	Feistel	12/16
Cast6 (Adams, 1999)	128/160/192/224/256	128	Feistel	48
Clefia (Shirai et al., 2007)	128/192/256	128	Feistel	18/22/26
DES (Standard, 1977)	54	64	Feistel	16
3DES (Simpson and Baldwin, 1997)	56/112/168	64	Feistel	48
DESL (Leander et al., 2007)	54	64	Feistel	16
DESX (Rogaway, 1996; Leander et al., 2007)	184	64	Feistel	16
DESXL (Leander et al., 2007)	184	64	Feistel	16
GOST (Courtois, 2012)	256	64	Feistel	32
Hight (Hong et al., 2006)	128	64	Feistel	32
Hummingbird (Engels et al., 2009)	256	16	SPN	4
Hummingbird2 (Engels et al., 2012)	256	16	SPN	4
Iceberg (Standaert et al., 2004)	128	64	SPN	16
Idea (Lai and Massey, 2006)	128	64	Lai–Massey	8.5
Kasumi (3rd Generation Partnership Project, 2001)	128	64	Feistel	8
Katan (Canniere et al., 2009)	80	32/48/64	Stream	254
Ktantan (Canniere et al., 2009)	80	32/48/64	Stream	254
Khudra (Kolay and Mukhopadhyay, 2014)	80	64	Feistel	18
Klien (Gong et al., 2012)	64/80/96	64	SPN	12/16/20
Lblock (Wu and Zhang, 2011)	80	64	Feistel	32
LEA (Hong et al., 2013)	128,192,256	128	Feistel	24/28/32
Led (Guo et al., 2011)	64/128	64	SPN	32/48
mCrypton (Lim and Korkishko, 2006)	64/96/128	64	SPN	12
Mibs (Izadi et al., 2009)	64/80	64	Feistel	32
Misty1 (Matsui, 1997)	128	64	Feistel	8
Noekeon (Daemen et al., 2000)	128	128	SPN	16
Piccolo (Shibutani et al., 2011)	80/128	64	Feistel	25/31
Present (Bogdanov et al., 2007; Poschmann, 2009)	80/128	64	SPN	31
Prince (Borghoff et al., 2012)	128	64	SPN	12
PRINTcipher (Knudsen et al., 2010)	48/96	48/96	SPN	48/96
RC2 (Knudsen et al., 1998)	8-1024	64	Feistel	18
RC5 (Rivest, 1995)	0-2040	32/64/128	Feistel	1–255
RC6 (Rivest et al., 1998)	128/192/256	128	Feistel	20
Safer (Massey, 1994)	64	64	SPN	6
Sea (Standaert et al., 2006)	48,96,144,...	48,96,144,...	Feistel	Variable
Seed (Lee et al., 2005)	128	128	Feistel	16
Serpent (Anderson et al., 1998)	128/192/256	128	SPN	32
Skipjack (Biham et al., 1999)	80	64	Feistel	32
Tea (Wheeler and Needham, 1995)	128	64	Feistel	64
Twofish (Schneier et al., 1998)	128/192/256	128	Feistel	16
Xtea (Needham and Wheeler, 1997)	128	64	Feistel	64
Twine (Suzaki et al., 2013)	80/128	64	Feistel	32

The PRESENT Block Cipher

- AES, designed for **software efficiency** on 8-bit or 32-bit processors, is not suitable for extremely constrained environments such as RFID tags and sensor networks.
- PRESENT, the *hardware-optimized ultra-lightweight block cipher*, was designed to fill the gap.
- Security and **hardware efficiency** (*area* and *power* efficient) are equally important during the design of PRESENT.
- Serialized architecture in reference [2] requires only **1000 GE** (gate equivalent) hardware footprint, competitive with leading compact stream ciphers.
  - Stream ciphers are potentially more compact, however
  - Block cipher is a versatile primitive and can be used as a stream cipher
  - Block cipher design seems to be better understood than that of steam cipher

Gate Equivalent (GE)

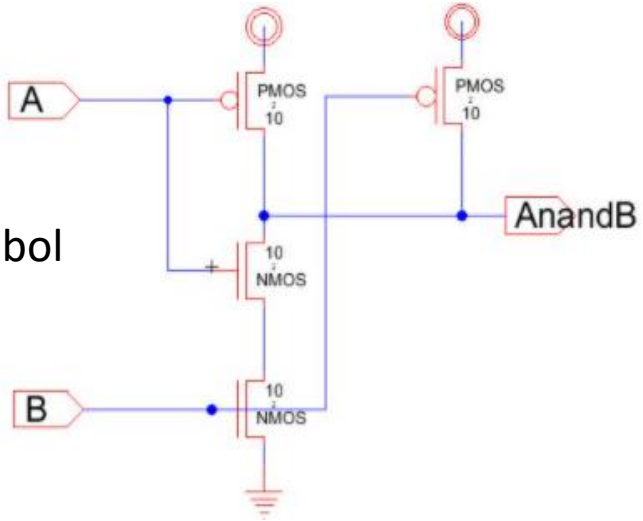
- A **gate equivalent** (**GE**) stands for a unit of measure which allows to specify *manufacturing-technology-independent complexity* of digital electronic circuits.
- A **gate equivalent (GE)** is the silicon area of a two-input NAND gate.
- A 2-input NAND gate in CMOS technology consists of four transistors



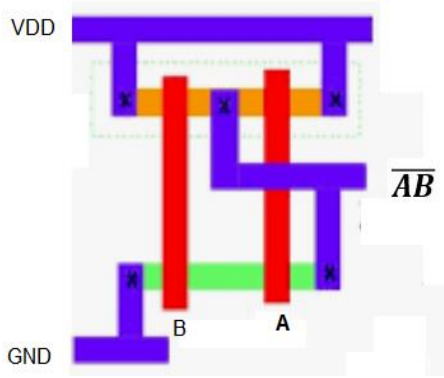
2-input NAND gate Logic symbol

Truth table

A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0



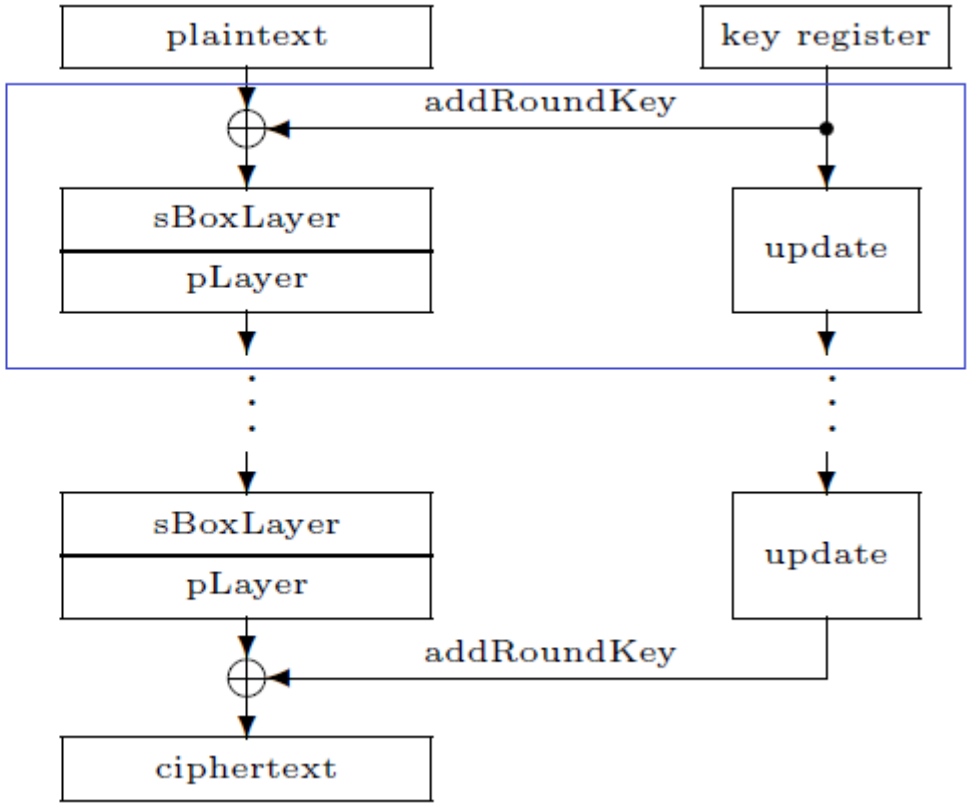
CMOS circuit diagram



Silicon mask layout diagram

Top-level algorithmic description of PRESENT

generateRoundKeys()
for $i = 1$ to 31 **do**
  addRoundKey(STATE,K_i)
  sBoxLayer(STATE)
  pLayer(STATE)
end for
addRoundKey(STATE,K_{32})



Three transformations in each round

- There are 3 transformations on the *state* in each of the 31 rounds
  - addRoundKey(STATE, K_i): XOR STATE with round key K_i, $1 \leq i \leq 31$
  - sBoxLayer(STATE): a non-linear substitution, a single *4-bit S-box* is used 16 times in parallel
  - pLayer(STATE): A linear bitwise permutation
- addRoundKey(STATE, K_{32})
  - K32 is used for post-whitening
  - In cryptography, key whitening is a technique *intended* to increase the security of an iterated block cypher. It consists of steps that combine the data with portions of the key.
  - The most common form of key whitening is *xor-encrypt-xor*, using a simple XOR before the first round and after the last round of encryption.[Wikipedia]

addRoundKey(STATE, K_i)

- Given *round key* $K_i = k_{63}^i \ldots k_0^i$, for 1 ≤ i ≤ 32,
- Current STATE $b_{63} \ldots b_0$
- addRoundKey consists of updating STATE as follows

$$b_j \rightarrow b_j \oplus \kappa_j^i$$

- Each b_j is updated with $b_j \oplus \kappa_j^i$, for 0 ≤ j ≤ 63.

sBoxLayer

- PRESENT uses a *4-bit to 4-bit S-box*, in *GF(2^4).*

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 Current STATE $b_{63} \ldots b_0$ is considered as sixteen 4-bit words $w_{15} \ldots w_0$, each represented using a hexadecimal digit, where $w_i = b_{4*i+3} \| b_{4*i+2} \| b_{4*i+1} \| b_{4*i}$, for 0 ≤ i ≤ 15.
- $S[w_i]$ provides the updated value for w_i
- The S-box is resistant to *differential and linear attacks*.
- The S-box is particularly well-suited to efficient hardware implementation
- The S-box is designed to mprove the avalanche property

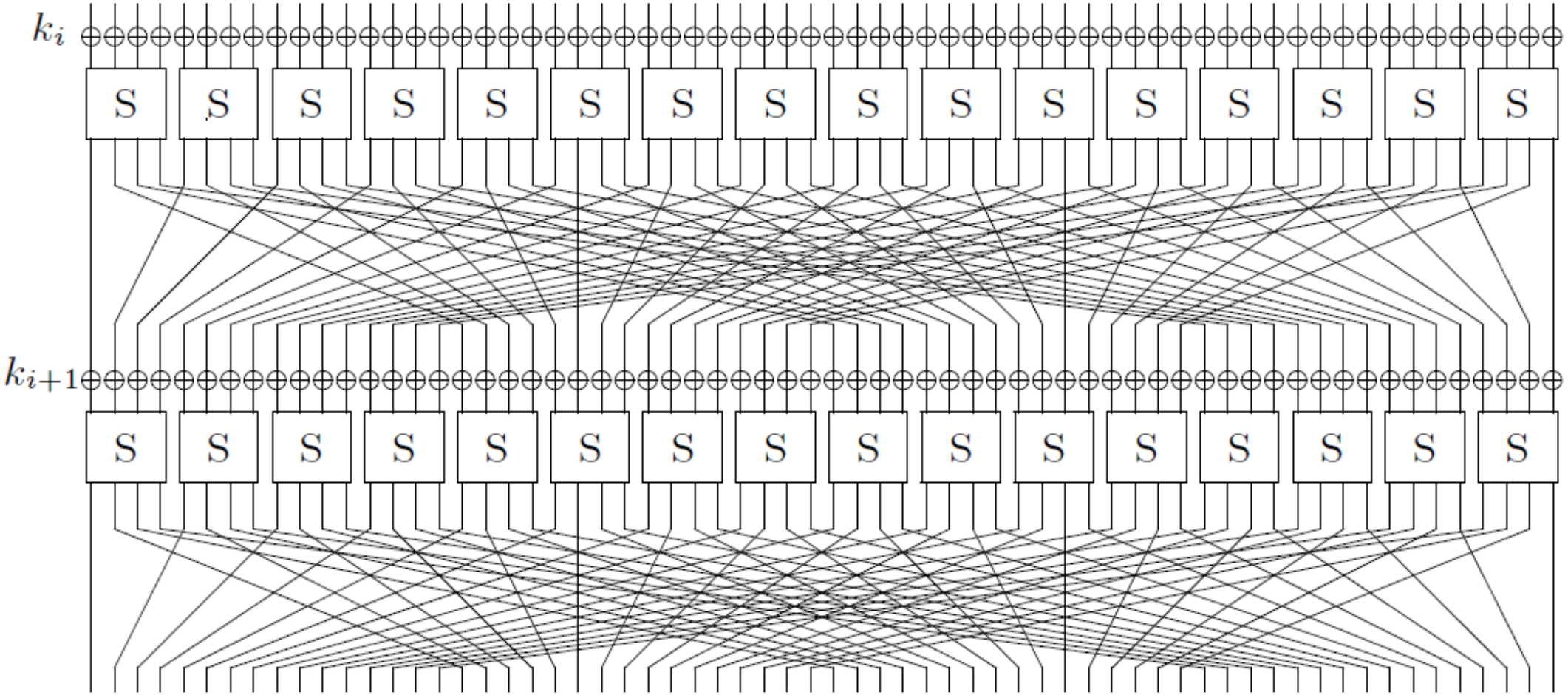
pLayer

- The bit permutation used in PRESENT is given by the following table:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

- Bit *i* of STATE is moved to bit position *P(i)*.

The Substitution/Permutation Network for PRESENT



PRESENT-80 Key Schedule

- The user-supplied 80-bit key, $k_{79}k_{78}\ldots k_0$, is stored in a *key register K*.

- At round *i*, the 64-bit *RoundKey*, $K_i = k_{63}k_{62}\ldots k_0$, is the *leftmost 64 bits of the current contents of the key register K*, i.e.

$$K_i = (k_{63}k_{62}\ldots k_0)_{\text{round\_key}} = (k_{79}k_{78}\ldots k_{16})_{\text{key\_register}}$$

- After extracting the *RoundKey* K_i, the key register $K = k_{79}k_{78}\ldots k_0$ is updated.

PRESENT-80 Key update procedure

1. The key register *K* is *left rotated 61-bit positions* (equivalent to *right rotated 80 – 61 (i.e. 19) bit positions*).

$$[k_{79}k_{78}\ldots k_1k_0] = [k_{18}k_{17}\ldots k_{20}k_{19}]$$

2. The *left-most four bits* are passed through the PRESENT S-box

$$[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$$

3. The 5 bits $k_{19}k_{18}k_{17}k_{16}k_{15}$ of the key register *K* are XORed with the *5-bit* round_counter value *i*

$$[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \texttt{round\_counter}$$

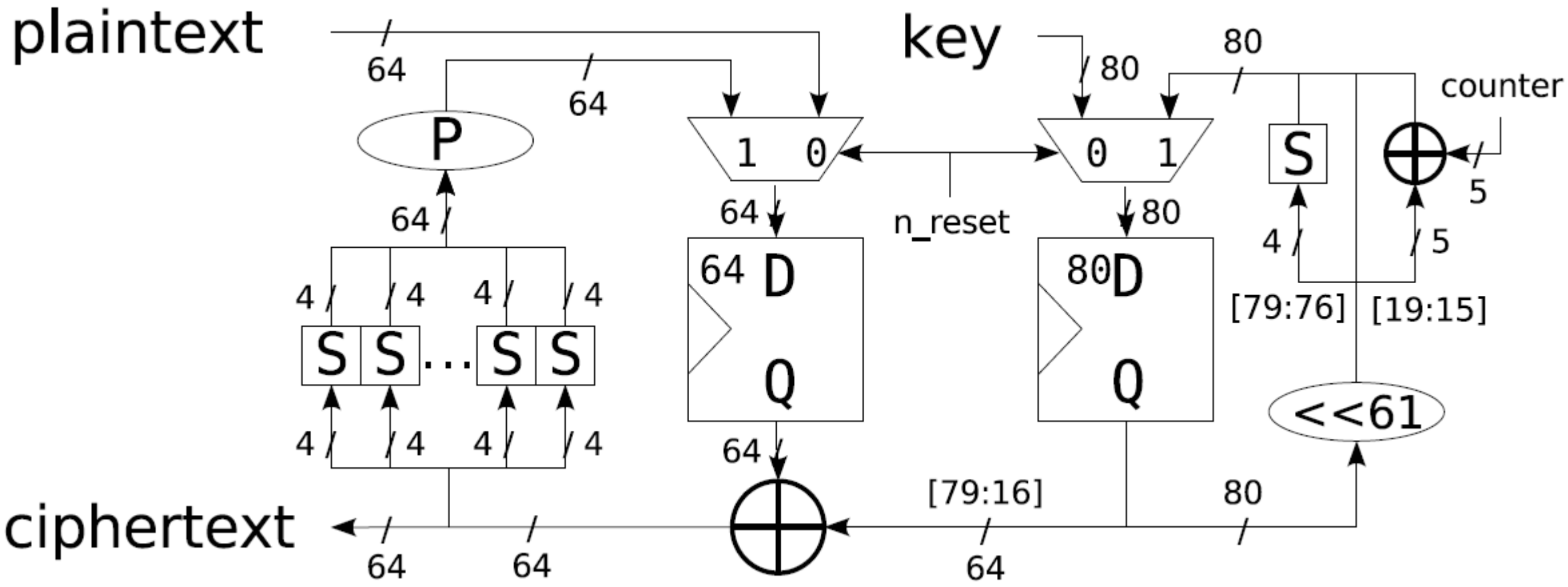
Design Issues for PRESENT

- Designed for *highly constrained environments.*
  - Not necessarily suitable for widespread use; we have AES for this.
  - Targeting some particular applications for which AES is unsuitable.
- Applications Characteristics
  - The cypher is optimised for hardware implementation.
  - Only requires moderate security levels; thus, 80-bit security is adequate.
  - Unlikely to require the encryption of large amounts of data. Thus, implementation is optimised for performance or space without too much practical impact.
  - In some applications, fixing the key during device manufacture is possible; there is no need to re-key a device, thus ruling out a range of key manipulation attacks.
  - Important metrics: security, space required for an implementation, peak and average power consumption.

PRESENT-80 needs less than half the size of AES in hardware

- Implementing both encryption and decryption of PRESENT is still smaller than an encryption-only AES.
- Implementing encryption-only PRESENT will give an ultra-lightweight solution.
- Encryption-only implementation can be used within challenge-response authentication protocols.
- In *counter* mode, the encryption engine is used for encryption and decryption.
- The encryption subkeys can be computed on the fly.

Area-optimized datapath of PRESENT-80



Round-based PRESENT-80 architecture

Attacks to PRESENT

- Attacks that manipulate *time-memory-data tradeoff* or the *birthday paradox when encrypting large amounts of data.*
  - Such attacks depend solely on the parameters of the block cypher and do not exploit the cypher's inner structure.
  - PRESENT is *unlikely* to be used in these scenarios.
- *Side-channel* and *invasive hardware attacks* are likely to be a threat to PRESENT, as they are to all cryptographic primitives
  - For the likely applications, where moderate security is adequate, any attacker would make minimal gains in practice.
  - In a risk assessment, such attacks are unlikely to be a significant factor.

Security Analysis of PRESENT-80

- Differential and linear cryptanalysis
- Structural attacks
- Algebraic attacks
- Key schedule attacks

Differential and linear cryptanalysis

- Theoretically proved that data required to exploit differential characteristic to launch differential cryptanalysis of PRESENT-80 exceeds the amount available.

- Under the assumption that a cryptanalyst need only approximate 28 of the 31 rounds in PRESENT to mount a key recovery attack, *linear cryptanalysis of PRESENT-80 would require of the order of* 2^{84} *known plaintext/ciphertext*.
  - Intended for applications not required to encryption large amounts of data.
  - Such a data requirement exceed the available text.

Structural attacks

- Structural attacks are well-suited to the analysis of SPN-based block ciphers (such as AES and PRESENT).
- These attacks are more effective on SPN-based ciphers with strong word-like structures, where the words are typically bytes.
- PRESENT is almost exclusively bitwise,
  - While the permutation operation is somewhat regular, the development and propagation of word-wise structure are disrupted by the bitwise operations in the cipher.
- So PRESENT is designed to be difficult for structural attacks.

Algebraic attacks

- Algebraic attacks had better success in attacking stream cyphers than block cyphers.
- The PRESENT S-box is described by 21 *quadratic equations* in the eight input/output-bit variables over $GF(2^4)$.[1]
- PRESENT-80 consists of 11,067 *quadratic equations* in 4,216 variables.
- The general problem of solving a system of multivariate quadratic equations is NP-hard.
- The time and memory complexity makes algebraic attacks on PRESENT-80 impractical.

Key schedule attacks

- The most effective key schedule attacks are *related-key attacks* and *slide attacks*
  - Both rely on the build-up of identifiable relationships between different sets of subkeys
- PRESENT uses a round-dependent counter so that subkey sets cannot easily be "slid",
- PRESENT uses a non-linear operation to mix the content of the key register *K*
  - All bits in the key register are a *non-linear function* of the 80-bit user-supplied key by round 21
  - That each bit in the key register after round 21 depends on at least four of the user-supplied key bits and
  - By the time we arrive at deriving K_{32},
    - six bits are degree two expressions of the 80 use-supplied key bits (involving quadratic terms or terms raised to the power of two)
    - 24 bits are of degree three,
    - the remaining bits are degree six or nine functions of the user-supplied key bits.
- The above properties are believed to be sufficient to resist key schedule-based attacks on PRESENT-80

Hardware Performance

- The designers implemented PRESENT-80 in VHDL and synthesized targeting Virtual Silicon (VST) standard cell library based on 0.18 µ logic process in 2007 [1].

module	GE	%	module	GE	%
data state	384.39	24.48	KS: key state	480.49	30.61
s-layer	448.45	28.57	KS: S-box	28.03	1.79
p-layer	0	0	KS: Rotation	0	0
counter: state	28.36	1.81	KS: counter-XOR	13.35	0.85
counter: combinatorial	12.35	0.79	key-XOR	170.84	10.88
other	3.67	0.23			
			sum	**1569.93**	**100**

Table 1. Area requirement of PRESENT-80

- area-optimized implementation
- Power-optimized implementation uses an additional 53 GE and attains a power consumption of only 3.3 µW.
- PRESENT-128 would occupy an estimated area of 1886 GE.

- GE numbers in the table are after synthesis (i.e., translated VHDL code to logic gates) but before placement and route.
- Bit permutations are simple wiring and increase the area only when the implementation is taken to the placement and route step.
- A serialised data path implementation of PRESENT-80 uses only 1000 GE in reference [3]

A comparison of lightweight cipher hardware implementations

	Key size	Block size	Cycles per block	Throughput at 100KHz (Kbps)	Logic process	Area	
						GE	rel.
Block ciphers							
PRESENT-80	80	64	32	200	0.18μm	1570	1
AES-128 [16]	128	128	1032	12.4	0.35μm	3400	2.17
HIGHT [22]	128	64	1	6400	0.25μm	3048	1.65
mCrypton [30]	96	64	13	492.3	0.13μm	2681	1.71
Camellia [1]	128	128	20	640	0.35μm	11350	7.23
DES [37]	56	64	144	44.4	0.18μm	2309	1.47
DESXL [37]	184	64	144	44.4	0.18μm	2168	1.38
Stream ciphers							
Trivium [18]	80	1	1	100	0.13μm	2599	1.66
Grain [18]	80	1	1	100	0.13μm	1294	0.82

Table 2. Comparison of lightweight cipher implementations

Table extracted from Reference [3]

Security level of AES and PRESENT

Table 1. Overview of the 19 lightweight block ciphers considered in this paper. Block, key and round key sizes are expressed in bits. The security level is the ratio of the number of rounds broken in a single key setting to the total number of rounds.

Cipher	Year	Block size	Key size	Round key size	Rounds	Security level	Type	Target
AES	1998	128	128	1408	10	0.70	SPN	SW, HW
Chaskey	2014	128	128	0	8/16	0.87/0.43	Feistel	SW
Fantomas	2014	128	128	0	12	NA	SPN	SW
HIGHT	2006	64	128	1088	32	0.81	Feistel	HW
LBlock	2011	64	80	1024	32	0.72	Feistel	HW, SW
LEA	2013	128	128	3072	24	0.63	Feistel	SW, HW
LED	2011	64	80	0	48	NA	SPN	HW, SW
Piccolo	2011	64	80	864	25	0.56	Feistel	HW
PRESENT	2007	64	80	2048	31	0.84	SPN	HW
PRIDE	2014	64	128	0	20	NA	SPN	SW
PRINCE	2012	64	128	192	12	0.83	SPN	HW
RC5*	1994	64	128	1344	20	0.80	Feistel	SW, HW
RECTANGLE	2015	64	80/128	1664	25	0.72	SPN	HW, SW
RoadRunneR	2015	64	80/128	0	10/12	0.5/0.58	Feistel	SW
Robin/Robin*	2014	128	128	0	16	1/NA	SPN	SW
Simon	2013	64	96/128	1344/1408	42/44	0.71/0.70	Feistel	HW, SW
SPARX	2016	64/128	128	1600/4224	24/32	0.62/0.68	Feistel	SW
Speck	2013	64	96/128	832/864	26/27	0.73/0.74	Feistel	SW, HW
TWINE	2011	64	80	1152	36	0.64	Feistel	HW, SW

Table extracted from reference [4]

PRESENT-80 Test Vectors

Sample test vectors are provided in Appendix 1 of the algorithm designers' paper

A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher", Cryptographic Hardware and Embedded Systems - CHES 2007. Proceedings 9th International Workshop. (Lecture Notes in Computer Science vol. 4727), p 450-66, 2007
https://iacr.org/archive/ches2007/47270450/47270450.pdf

Appendix I

Test vectors for PRESENT with an 80-bit key are shown in hexadecimal notation.

plaintext	*key*	*ciphertext*
00000000 00000000	00000000 00000000 0000	5579C138 7B228445
00000000 00000000	FFFFFFFF FFFFFFFF FFFF	E72C46C0 F5945049
FFFFFFFF FFFFFFFF	00000000 00000000 0000	A112FFC7 2F68417B
FFFFFFFF FFFFFFFF	FFFFFFFF FFFFFFFF FFFF	3333DCD3 213210D2

More test vectors can be found at:
https://github.com/cantora/avr-crypto-lib/blob/master/testvectors/present
https://github.com/cantora/avr-crypto-lib/blob/master/testvectors/present/nessie-present-80_le.txt
https://github.com/cantora/avr-crypto-lib/blob/master/testvectors/present/nessie-present-128_le.txt

References

1. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, C. Vikkelsoe, “PRESENT: An Ultra-Lightweight Block Cipher”, Cryptographic Hardware and Embedded Systems - CHES 2007. Proceedings 9th International Workshop. (Lecture Notes in Computer Science vol. 4727), p 450-66, 2007 https://iacr.org/archive/ches2007/47270450/47270450.pdf [Test Vectors included]
2. C. Rolfes, A. Poschmann, G. Leander, C. Parr, “Ultra-Lightweight Implementation for Smart Devices – Security for 1000 Gate Equivalents”, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v 5189, p 89-103, 2008, https://core.ac.uk/download/pdf/189198932.pdf
3. B. J. Mohd, T. Hayajneh, A. V. Vasilakos, “A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues”, Journal of Network and Computer Applications 58 (2015) 73-93 https://www.sciencedirect.com/science/article/abs/pii/S1084804515002076
4. D. Dinu, Y.L. Corre, D. Khovratovich, L. Perrin, J. Großschädl, “Triathlon of Lightweight Block Ciphers for the Internet of Things”, *Journal of Cryptographic Engineering*, v 9, n 3, p 283-302, September 1, 2019. https://eprint.iacr.org/2015/209.pdf

End of PRESENT Presentation

PRESENT-128 Test Vectors

Test vectors for PRESENT-128 can be found at:
https://github.com/cantora/avr-crypto-lib/blob/master/testvectors/present/nessie-present-128_le.txt

Set 1, vector# 0:
key=80000000000000000000000000000000
plain=0000000000000000
cipher=72FDB8013B1AB576

Set 2, vector# 0:
key=00000000000000000000000000000000
plain=8000000000000000
cipher=ECC531491456DFD2

Set 3, vector# 0:
key=00000000000000000000000000000000
plain=0000000000000000
cipher=96DB702A2E6900AF

Set 4, vector# 0:
key=000102030405060708090A0B0C0D0E0F
plain=0011223344556677
cipher=E6B982239DF3515D

Set 5, vector# 0:
key=80000000000000000000000000000000
plain=AF6327170DC4FB36
cipher=0000000000000000

Set 6, vector# 0:
key=00000000000000000000000000000000
plain=E7FB76C9174B3A19
cipher=8000000000000000

Set 7, vector# 0:
key=00000000000000000000000000000000
plain=F5C398817916CD42
cipher=0000000000000000

Set 7, vector#255:
key=FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
plain=FB6CD106CDCD8114
cipher=FFFFFFFFFFFFFFFF

Set 8, vector# 0:
key=000102030405060708090A0B0C0D0E0F
plain=01C6FA50BA66866C
cipher=0011223344556677

PRESENT Python Implementation

- http://www.lightweightcrypto.org/downloads/implementations/pypresent.py
- https://repo.or.cz/w/python-cryptoplus.git?a=blob;f=src/CryptoPlus/Cipher/pypresent.py;hb=HEAD