Lightweight Block Cipher Circuits for Automotive and IoT Sensor Devices

Santosh Ghosh, Rafael Misoczki, Li Zhao and Manoj R Sastry
SPR/Intel Labs, Intel Corporation
Hillsboro, OR, 97124 USA
{firstname.middlename.lastname}[at]intel.com

HASP 2017 : Hardware and Architectural Support for
Security and Privacy 2017
Toronto, Canada
June 25, 2017

Security in Automotive and IoT Sensor Devices

- IoT devices such as sensors typically have die area and power constraints
  - Attack against *integrity*, *authentication* and *confidentiality* are the major concerns [3]
  - This talk focuses on Automotive Security vertical
- Electronic Control Units (ECUs) control critical functionality in a car such as braking, acceleration etc
  - Connected to Controller Area Network (CAN) [1] in a car
- Lack of security in CAN has been exploited by hackers [2]
  - Security (*authenticity* of the sender, *integrity* of the messages and *replay* protections) is challenging because of very restrictive CAN packet format and safety critical applications such as braking and acceleration have a *low latency* requirement
  - Question: Whether cryptographic security is feasible?
    Which crypto algorithm would be best suited?

[1] BOSCH, 1991. CAN Specification Version 2.0
[2] Miller, C. and Valasek, C. 2016. Advanced CAN injection techniques for vehicle networks
[3] Dhanjani, N. 2013. Hacking lightbulbs: Security evaluation of the Philips hue personal wireless lighting system

Agenda

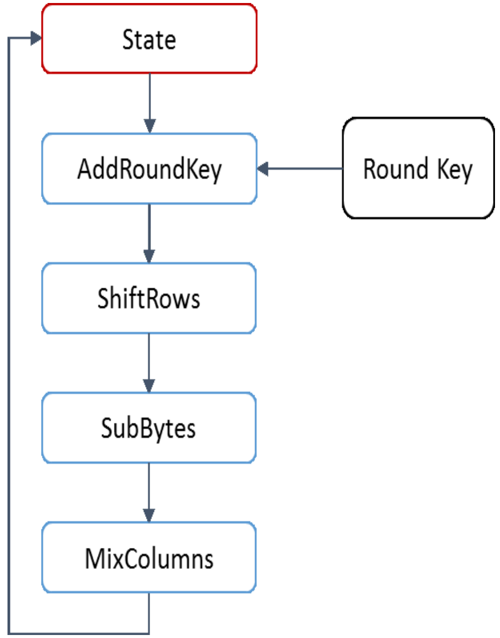
- Standard cipher algorithm and overhead
- New lightweight block ciphers and their SW overhead
- How fast they could be on HW
    - Design and implementations of PRINCE, SIMON, SPECK and PRESENT
    - Results & comparison
- Conclusion

Traditional cipher algorithm and overhead

Advanced Encryption Standard (AES)

- Block cipher w/ block size 128-bit and key size 128-bit/256-bit
- Round functions w/ four major operations – AddRoundKey, ShiftRows, SubBytes and MixColumns
- 10/14 rounds for 128-bit/256-bit keys
- SW overhead (32-bit MCU, 128-bit key) [4]:
  - Object code + constant footprint: 1.4kB
  - Latency: 12,300 clock cycles/block
- HW overhead (area optimized, 128-bit key) [5]:
  - Area footprint: 3400 gates
  - Latency: 1032 clock cycles/block

Unacceptable Latency for safety critical operations



AES round function

[4] Texas Instruments. C Implementation of Cryptographic Algorithms. http://www.ti.com/lit/an/slaa547a/slaa547a.pdf
[5] Feldhofer, M., Wolkerstorfer, J. and Rijmen. V. 2005. AES Implementation on a Grain of Sand

Lightweight block ciphers and SW overhead

Lightweight: Small code/area footprint, minimum latency and low power

- Block cipher w/ 64-bit block and 128-bit key
- PRESENT, PRINCE, SIMON, SPECK, …
- Software overhead:

Block Cipher	Object code + constant size	Latency [clock cycles/block]
PRINCE [6]	-	-
SIMON [7]	282	1988
SPECK [7]	186	1197
PRESENT [8]	487	10666

Footprint and latency of lightweight block ciphers in 8-bit software

[6] Borghof et al. 2012. PRINCE – A low-latency block cipher for pervasive computing applications. IACR eprint archive, report 529
[7] Beaulieu et al. 2013. The SIMON and SPECK families of lightweight block ciphers. IACR eprint archive, report 404
[8] Bogdanov et al. 2007. PRESENT: An Ultra-Lightweight Block Cipher

Lightweight block ciphers and existing HW overhead

Block Cipher	Area [GE] per Round Path	Latency [clock cycles/block]
PRINCE [6]	689	12
SIMON [7]	1000	44
SPECK [7]	1127	27
PRESENT [8]	1339	31
MCRYPTON [9]	2949	12
LED [9]	2265	48
PICCOLO [7]	1334	31

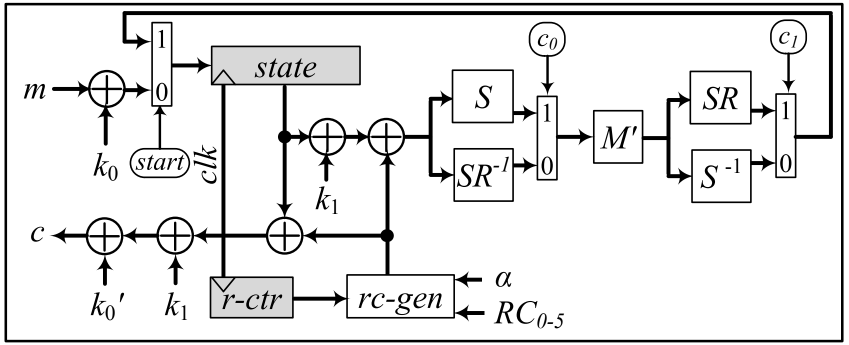
Area and latency of the existing hardware designs

[6] Borghof et al. 2012. PRINCE – A low-latency block cipher for pervasive computing applications. IACR eprint archive, report 529
[7] Beaulieu et al. 2013. The SIMON and SPECK families of lightweight block ciphers. IACR eprint archive, report 404
[8] Bogdanov et al. 2007. PRESENT: An Ultra-Lightweight Block Cipher
[9] Miroslav Kneževic et al. 2012. Low-Latency Encryption - Is "Lightweight = Light + Wait"? CHES 2012

HW Design and implementations of PRINCE

11 Round, 64-bit block, 128-bit key

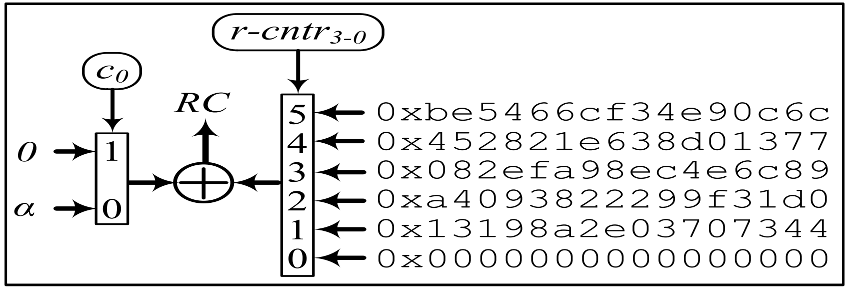
- Rounds: 5 forward, 1 middle, 5 reverse
- Ki-add: state is XORed w/ 64-bit sub-keys (k_0, k_0', k_1)
- S-Layer: 4-bit Sbox/Inverse-Sbox operations
- M/M'-Layer: state is multiplied w/ a 64 x 64 matrix M
  - $M = SR \circ M'$ and $M^{-1} = M' \circ SR^{-1}$
- RCi-add: state is XORed w/ 64-bit round constant



PRINCE round computation block

Key Expansion

- 128-bit → 192-bit
- $(k_0 \,||\, k_1) \rightarrow (k_0 \,||\, k_0' \,||\, k_1)$,
  where $k_0' = (k_0 >>> 1) \oplus (k_0 >> 63)$



RC generation

Implemented w/ 1 round operation/clock

- Optimized Boolean mapping for S and M/M' layers

[6] Borghof et al. 2012. PRINCE – A low-latency block cipher for pervasive computing applications. IACR eprint archive, report 529

HW Design and implementations of PRESENT

31 round, 64-bit block, 128-bit key

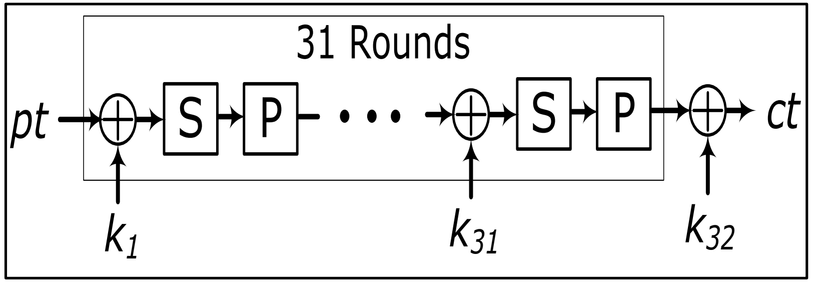
- AddRoundKey: Round key is XORed with 64-bit state
- pLayer (P): Permutation of the state
- Sbox (S): A 4x4 non-linear mapping

Key Schedule

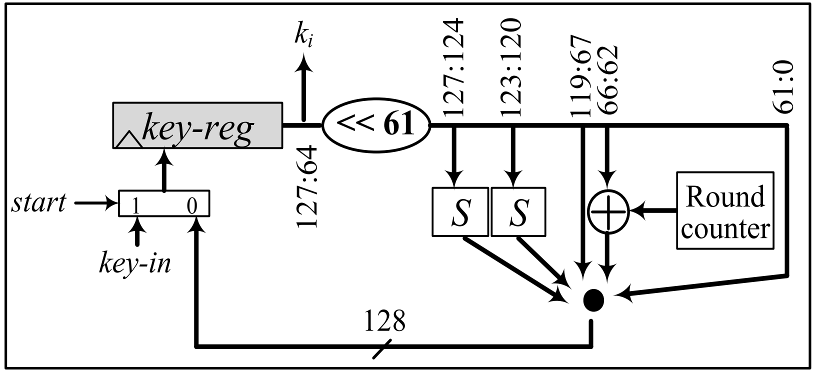
- kj consists of 64 most significant bits
- Key register is updated at every clock
  - 61-bit left shift
  - 2 Sbox operations
  - XORed 5-bit round number with 5 intermediate bits

Implemented w/ 1 round operation/clock

- Optimized Boolean mappings for S
- Simple rewiring w/o any logic gates for P



PRESENT cipher computation



PRESENT key schedule

[8] Bogdanov et al. 2007. PRESENT: An Ultra-Lightweight Block Cipher

HW Design and implementations of SIMON

44 round, 64-bit block, 128-bit key

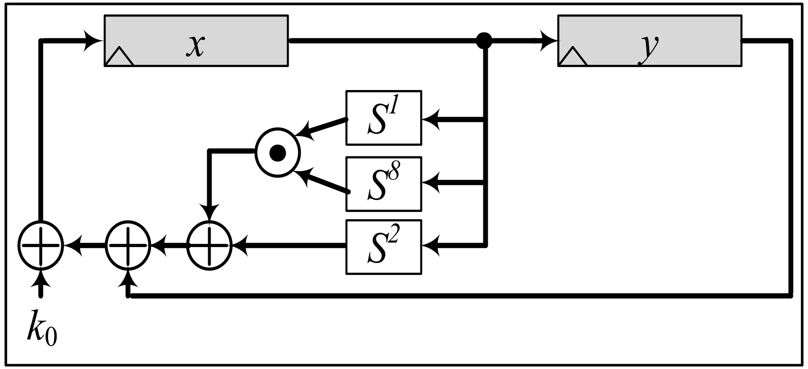
- AddRoundKey: Round key (k) is XORed with 64-bit state
- Rotation (S^j/S^{-j}): j-th bit clockwise and anti-clockwise
- Feistel Structure: Second half (y) is replaced with first half (x); whereas x is updated w/ a function F(x, y, k)

Key Expansion

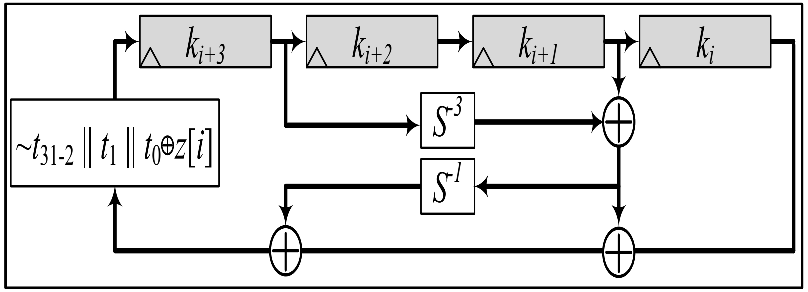
- 128-bit key is divided into 4 words (k_3, k_2, k_1, k_0)
- Word k_0 is considered as the current round key
- 40-bits round constant z, absorbed in rounds 5 to 40 @1-bit/round
- Key words are updated as: $k_0 \leftarrow k_1$, $k_1 \leftarrow k_2$, $k_2 \leftarrow k_3$, and $k_3 \leftarrow F_2(k_0, k_1, k_3, z)$

Implemented w/ 1 round operation/clock

- Simple rewiring w/o any logic gates for S^j/S^{-j}



SIMON64/128 round computation



SIMON64/128 key expansion

[7] Beaulieu et al. 2013. The SIMON and SPECK families of lightweight block ciphers. IACR eprint archive, report 404

HW Design and implementations of SPECK

27 round, 64-bit block, 128-bit key

- AddRoundKey: round key (k) is XORed with 64-bit state
- Rotation (S^j/S^{-j}): j-th bit clockwise and anti-clockwise
- Double Feistel Structure: both halves are updated w/ functions $F_1(x, y, k)$ and $F_2(x, y, k)$

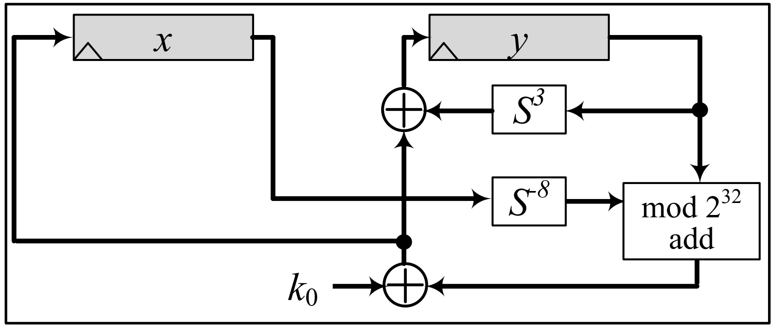
Key Expansion

- 128-bit key is divided into 4 words (k_3, k_2, k_1, k_0)
- Word k_0 is considered as the current round key
- Key words are updated as: $k_0 \leftarrow F_3(k_0, k_1, r)$, $k_1 \leftarrow k_2$, $k_2 \leftarrow k_3$, and $k_3 \leftarrow F_4(k_0, k_1, r)$,

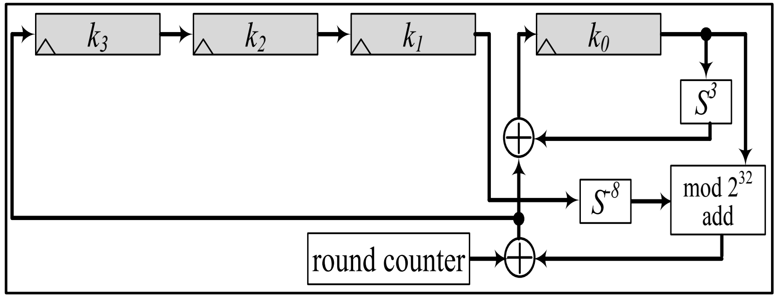
  where r is the round number

Implemented w/ 1 round operation/clock

- Simple rewiring w/o any logic gates for S^j/S^{-j}
- Round counter w/ only 6-bit vs 32-bit registers



SPECK64/128 round computation



SPECK64/128 key expansion

[7] Beaulieu et al. 2013. The SIMON and SPECK families of lightweight block ciphers. IACR eprint archive, report 404

Results & Comparison

RTL in Verilog, Synopsys Design Compiler G-2012.06-SP3, Intel's 14nm high-K/metal-gate FinFET CMOS @200MHz, 0.75V [10]

Block Cipher	block, key	Area [μm^2]			Gates	Latency [CC]	Latency x Gates x 10^3
		Comb	Seq	Total			
PRINCE	64, 128	236	45	281	1258	12	15.10
PRESENT	64, 128	99	111	210	934	31	29.89
SPECK	64, 128	146	132	278	1244	27	33.59
SIMON	64, 128	133	175	308	1378	44	60.63

Area and latency results

Block Cipher	block, key	Power [μW]				Energy [pJ] /bit
		Internal	Switch	Leak	Total	
PRINCE	64, 128	64	35	7	116	0.11
PRESENT	64,128	86	15	7	108	0.23
SPECK	64, 128	94	17	7	118	0.25
SIMON	64, 128	120	15	8	143	0.49

Power and energy consumption

[10] Natarajan et al. 2014. A 14nm logic technology featuring 2nd-generation FinFET, air-gapped interconnects, self-aligned double patterning and a 0.0588 μm2 SRAM cell size

Conclusions

Whether cryptographic security is feasible for CAN messages?

- Payload size of a CAN packet [1]: 64 bits
- CAN operating speed [1]: 125 Kbits/s
- Latency of one CAN packet transmission: 0.8 ms
- Payload latency overhead: ~0.1% (compared to transmission latency)

Block Cipher	Payload encryption latency @40MHz ECU [11]	Payload latency overhead
PRINCE	300 ns	0.04%
PRESENT	775 ns	0.10%
SPECK	675 ns	0.08%
SIMON	1100 ns	0.14%

Payload encryption latency

Which crypto algorithms would be best suited for CAN and IoT sensor devices?

- PRINCE, SPECK, PRESENT, SIMON

[1] BOSCH, 1991. CAN Specification Version 2.0
[11] Discovery Plus Kit for SPC56 L line - with SPC56EL70L5 MCU. http://www.st.com/en/evaluation-tools/spc56l-discovery.html

Thanks!