Nesnelerin İnterneti için Hafif Kriptografi

Doç. Dr. **Cihangir Tezcan**

MIDDLE EAST TECHNICAL UNIVERSITY

Enformatik Enstitüsü, Siber Güvenlik Anabilim Dalı
ORTA DOĞU TEKNİK ÜNİVERSİTESİ, ANKARA

3. ULUSAL KRİPTO GÜNLERİ

5 Mayıs 2023

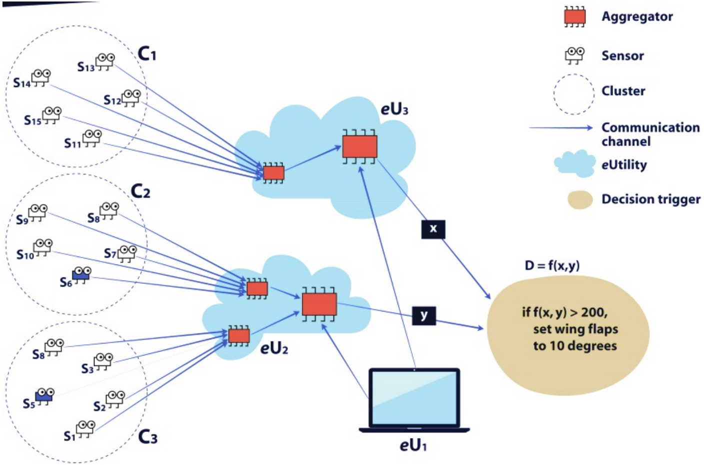
This work has been supported by TUBITAK 1001 Project under the grant number 121E228 and by Middle East Technical University Scientific Research Projects Coordination Unit under grant number AGEP-704-2023-11294.

Lightweight Cryptography

Need for Lightweight Cryptography

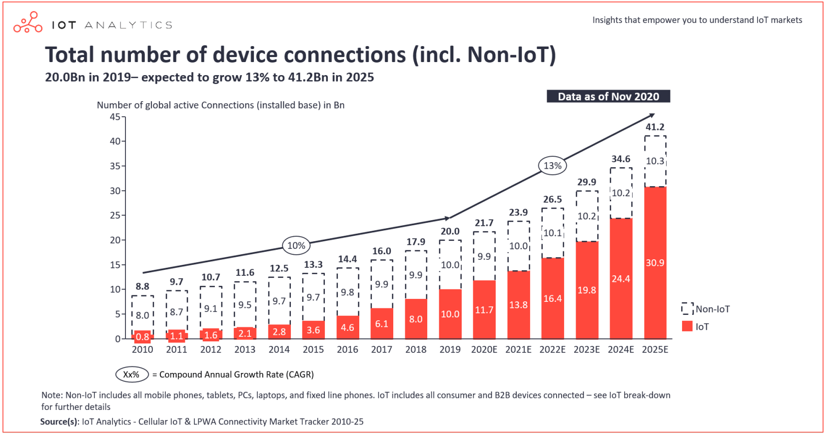
- Evolutionary change in computing and information technologies
    1. one computer - many users
    2. one computer - one user
    3. many computers - one user (IoT)
- Many IoT systems run on resource-constraint platforms (e.g. mobile tokens without battery, RFIDs or medical implants that do not allow change of batteries)
- Many IoT devices are extremely cost-sensitive because they are deployed in high volumes
- Industry needed for lightweight crypto for at least 30 years
- Became an active research area in academia over the last 15 years
- Industry deployed several proprietary, self-made algorithms and many disasters happened
    1. KeeLoq (for remote keyless entry)
    2. Mifare Classic (for contactless smartcards)
    3. ...

Internet/Network of Things



Taken from **NIST-SP 800-183 Network of Things**

Who is Responsible for IoT Security



IoT connections, surpassing non-IoT for the first time:
`https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time`
Instead of securing every device, current approach is the collect all data, process it and then detect problems

Internet/Network of Things

Need of Lightweight Cryptography

1. Encrypted communication / Security
2. Authenticated encryption
3. Battery life
4. Energy
5. Latency
6. Throughput
7. Side Channel Resistance

Data Encryption Standard (DES)

Data Encryption Standard (DES)

- Designed by IBM in 1970s, based on an earlier design by Feistel
- In 1976, NSA tweaked the algorithm by changing its S-boxes
  - **Block Size:** 64 bits
  - **Key Length:** 56 bits
  - **Rounds:** 16
- Currently known as Data Encryption Algorithm (DEA) since it is no longer a standard
- Became useless after 1990s since its short key is susceptible to brute force attacks

AES

Advanced Encryption Standard (AES)/Rijndael

- Designed by Joan Daemen and Vincent Rijmen
- Standardized in 2001 by NIST (winner of the AES competition)
- Other finalists were SERPENT, TWOFISH, RC6, MARS
  - **Block Size:** 128 bits
  - **Key Length:** 128, 192, 256 bits
  - **Rounds:** 10, 12, 14 *(depends on the key length)*
- Known attacks are *ineffective*

The Need for Lightweight Cryptography

Many modern cryptographic algorithms are designed for desktop computers or servers. Thus, they may not fit into many IoT device or when they did, their performance may not be as good as expected

How Hard Is It to Perform Brute Force Attack?

Lightweight cryptography does not mean short keys!!!!!

2^{56} = 72.057.594.037.927.936

2^{80} = 1.208.925.819.614.629.174.706.176

2^{128} = 340.282.366.920.938.463.463.374.607.431.768.211.456

2^{192} = 6.277.101.735.386.680.763.835.789.423.207.666.416.102.355.444.464.034.512.896

2^{256} = 115.792.089.237.316.195.423.570.985.008.687.907.853.269.984.665.640.564.039.457.584.007.913.129.639.936

Stream Ciphers

Generally was faster than block ciphers until lightweight designs came off

Some Stream Ciphers

- A5/1 (GSM)
- RC4 (WEP)
- E0 (Bluetooth)

Stream Ciphers

- eStream competition started in 2004 had 34 candidate algorithms and 7 of them are included in the final portfolio in 2008
- **Hardware:** Grain v1, MICKEY 2.0, Trivium
- **Software:** HC-128, Rabbit, Salsa20/12, SOSEMANUK
- Trivium became ISO/IEC standard (29192-3:2012)
- ChaCha20 was included in TLS 1.3

KeeLoq

KeeLoq (mid-1980s)

- KeeLoq is a proprietary hardware-dedicated block cipher that uses an NLFSR
- Used for keyless entry systems (cars, garages, buildings, etc.)
- Used by many companies like as Chrysler, Daewoo, Fiat, GM, Honda, Toyota, Volvo, Volkswagen Group, Clifford, Shurlok, and Jaguar

KeeLoq

Best Attacks [1]

1. Side-channel attacks can reveal both the secret key of a remote transmitter and the manufacturer key stored in a receiver. A remote control can be cloned from only ten power traces, allowing for a practical key recovery in few minutes.
2. Once knowing the manufacturer key, the secret key of a remote control can be obtained from a distance and replicated just by eavesdropping at most two messages (without physical access to the device)
3. Denial-of-service attack: The internal counter of the receiver (garage door, car door, etc.), which makes it impossible for a legitimate user to open the door

[1] Eisenbarth et al. Physical Cryptanalysis of KeeLoq Code Hopping Applications

TESLA

KU Leuven researchers demonstrate serious flaws in Tesla Model X keyless entry system

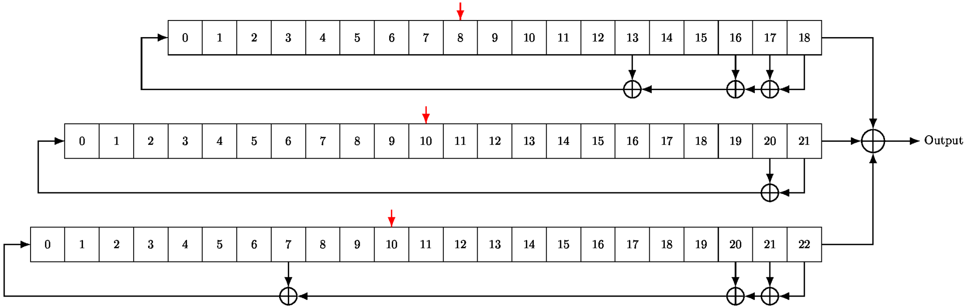
23 Nov 2020

Researchers at COSIC (KU Leuven/imec) have discovered major security flaws in the keyless entry system of the Tesla Model X. The same researchers previously hacked the Tesla Model S keyless entry system and now demonstrate how the more recent Tesla Model X can be stolen in a few minutes. Tesla has already released an over-the-air software update to mitigate these issues.

The Tesla Model X key fob allows the owner to automatically unlock their car by approaching the vehicle or by pressing a button. To facilitate the integration with phone-as-key solutions, which allow a smartphone APP to unlock the car, the use of Bluetooth Low Energy (BLE) is becoming more prevalent in key fobs. The Tesla Model X key fob is no different and uses BLE to communicate with the vehicle.

https://nieuws.kuleuven.be/en/content/2020/ku-leuven-researchers-demonstrate-serious-flaws-in-tesla-model-x-keyless-entry-system

A5/1 (2G Standard)



A5/1 (1987)

A5/1 Keystream Generation

1. All three registers are set to 0
2. For 64 cycles, key bits are fed to the LFSRs (without majority function)
3. For 22 additional cycles, a 22-bit frame number is fed to the LFSRs (without majority function)
4. 100 additional clocks with majority function is performed to obtain the *initial state*
5. 228 clocks are performed to produce 228 bits of keystream (which is XORed with the plaintext)

A5/1

Exhaustive Search on A5/1

- It is easy to guess a part of the GSM communication (i.e. we can guess some blocks of the plaintext)
- It is easy to capture ciphertext (since the communication is wireless, one can stay close to the target or to the base GSM station)
- So we can always capture some part of the key stream and perform exhaustive search on the key to check if it produces the same keystream
- We can perform 2^{32} encryptions using a single GPU
- With 32 GPUs, exhaustive search of 2^{64} encryptions would take a year

A5/1

Exhaustive Search on A5/1

- We cannot spend a year for every exhaustive search, we can do the search once and store the results
- Such a search would require $2^{64+22} = 2^{86}$ encyptions because we cannot know the frame number at the beginning
- Instead of performing exhaustive search on the key and the frame, we can do it on the initial state which can take at most 2^{64} different values (assume we have a way to go back from initial state to the key and the frame)
- Further analysis shows that not every 64-bit keystream can be achieved when every key and frame combination is tried: There are at most $2^{61.16}$ keystreams

A5/1

Time Memory Trade-off Attacks on A5/1

- We cannot store keystream for $2^{61.16}$ initial states
- We can reduce the memory cost by using rainbow tables
- In 2010, 1896 gigabytes of rainbow tables were constructed using GPUs for a M.Sc. Thesis at NTNU
- Given 114-bits of known plaintext, these tables can decrypt a GSM conversation with probability $> 19\%$
- Today we can create similar rainbow tables that would decrypt the conversation with probability $> 99\%$

A5/1 (2G Standard)

SECURITYRESEARCHLABS

Attack vector		Networks Avea	Turkcell	Vodafone
2G Over-the-air protection				
- Encryption algorithm	A5/1	**100%**	31%	**58%**
	A5/3	0%	**69%**	42%
- Require IMEI in CMC				
- Hopping entropy				
- Authenticate calls (MO)		27%	14%	11%
- Authenticate SMS (MO)		35%	29%	3%
- Authenticate paging (MT)		19%	57%	9%
- Authenticate LURs		91%	98%	89%
- Encrypt LURs		100%	100%	100%
- Update TMSI		30%	3%	4%
3G Over-the-air protection				
- Encryption				
- Update TMSI		9%	2%	10%

Data taken from GSM Map Project, June 2017 (Security Research Labs, Berlin)

STARLINK

MATT BURGESS SECURITY 10.08.2022 03:00 PM

The Hacking of Starlink Terminals Has Begun

It cost a researcher only $25 worth of parts to create a tool that allows custom code to run on the satellite dishes.

PHOTOGRAPH: NINA LYASHONOK/GETTY IMAGES

https://www.wired.co.uk/article/starlink-internet-dish-hack

Embedded Systems

Embedded Systems

- An embedded system is some combination of computer hardware and software that is specifically designed for a particular function
- Recently embedded systems have become more and more complex and are close to the functionality of a PC (e.g. smart phones)
- Many side-channel attacks focus on small embedded systems where all components (including RAM, Flash memory, etc.) that are critical for the security applications are integrated on a single piece of silicon
- Attacks on systems with multiple chips can be performed in similar ways and are typically simpler
- Integrated circuit (IC) Market: More than 5 billion devices per year!

Embedded System Security

Embedded System Security

- In general attack scenarios we only assume that the communication channel is insecure and the attacker can eavesdrop
- In case of embedded systems the attacker often has more *power*
  - **Pay TV:** The broadcasting company gives smart cards to customers, customers may try to duplicate the card
  - **Electronic purse:** The customers might want to add money themselves
  - **DRM:** The customers might want to copy the material for their friends
  - **Brand protection:** The customer wants to use cheaper ink, batteries,...
  - ....
- When designing an embedded system it is necessary to assume that the customer will try to break it

Attacks on Embedded Devices

Attacks on Embedded Devices

- Social Engineering
- Logical Attacks
  - Software vulnerabilities
  - API attacks
  - Cryptanalysis
- **Physical Attacks:** Observe and manipulate physical properties of the device or its environment

Examples of Physical Attacks

Examples of Physical Attacks

- You find a USB stick with secret data on it
- To access it you need to enter an 8-digit PIN
- Device has a delay response - so one trial takes 1 second
- Brute force requires 10^8 seconds $\approx$ 3 years
- Can we do better?

Examples of Physical Attacks

Examples of Physical Attacks

- Let's assume that password check is implemented on an embedded ARM processor in a straightforward way
- It checks correctness of the 1st digit, if it is correct then it checks the 2nd digit and so on
- Thus, execution time is directly proportional to the number of correct PIN digits!

A Timing Attack on the PIN Check

A Timing Attack on the PIN Check

1. Select a random PIN value
2. Fix all digits except the first one
3. Measure the timing for all 10 possible values of the first digit
4. The value that leads to the longest execution time is the correct one
5. Set the first digit to this value and proceed with the next digits in the same way
6. $10 \times 8 = 80$ trials are enough to access the USB stick
7. If the timing measurement is noisy, averaging over multiple measurements needs to be done

A Timing Attack on the PIN Check

A Timing Attack on the PIN Check

- Checking every digit of the entered PIN is not a solution because code requires more operations to be performed when a digit is correct
- Assume that the designer of the USB stick inserts NOP (no operation) instructions to balance the timing of both execution paths (i.e. password correct and password wrong)
- How can we attack such a fixed system?

Timing Attacks on Symmetric Cryptography

Timing Attacks on Symmetric Cryptography

- Timing attacks are in particular relevant for asymmetric cryptography
- However, attacks on symmetric cryptography is possible if either
  - the software implementation is not done carefully
  - the hardware adds data-dependent behavior
- Each intermediate data values of all cryptographic algorithm carries information about the key!!

Timing Attacks on Symmetric Cryptography

Timing Attacks on Symmetric Cryptography

- AES example: During the matrix multiplication, multiplication by 2 means
  - Shift left
  - If MSB was 1 before the shift, add $0x1B$ to the result
- because the multiplication is not an integer multiplication but it is a Galois Field multiplication

The Power Consumption of Digital Circuits

The Power Consumption of Digital Circuits

- The vast majority of digital circuits are implemented using CMOS logic
- The instantaneous power consumption of devices implemented in CMOS depends on
  - the instruction that is executed
  - the data that is being processed
- **Consequences for the PIN check implementation:** An attacker who measure the power consumption can easily detect whether NOP instructions are executed or not

A Power Analysis on the PIN Check

A Power Analysis on the PIN Check

1. Select a random PIN value
2. Fix all digits except the first one
3. Measure the power consumption for all 10 possible values of the first digit
4. The value that leads to a significantly different power consumption than the other nine is the correct one
5. Set the first digit to this value and proceed with the next digits in the same way
6. $10 \times 8 = 80$ power measurements are enough to access the USB stick
7. If the power measurement is noisy, averaging over multiple measurements needs to be done

Counteracting the Power Analysis Attack

Counteracting the Power Analysis Attack

- Counteracting power analysis attacks is a challenging task
- Many publications are available on this topic
- Assume that the designer of the USB stick is able to balance the timing and the power consumption of both execution paths (i.e. password correct and password wrong)
- How can we attack such a fixed system?

Reliability of Digital Circuits

Reliability of Digital Circuits

- Digital circuit require certain operating conditions to work properly
  - temperature range
  - operating frequency
  - supply voltage
  - ...
- Fault Attacks
  - By changing the operating conditions an attacker can bring the digital circuits into states with undefined or unpredictable behavior

DFA Playstation Vita (AES-256)

https://twitter.com/yifanlu/status/1091830262286020608

Mifare Classic

Mifare Classic 1k (1994)

- Produced by NXP Semiconductors used in contactless smart cards and proximity cards
- Produced more than 10 billion smart card chips and 150 million reader modules
- Uses CRYPTO1 stream cipher

Target Applications

- Public transportation
- Electronic toll collection
- Loyalty cards
- Event ticketing
- Car parking

Mifare Classic

ISO/IEC 14443-A

Identification, contactless integrated circuit, and proximity cards

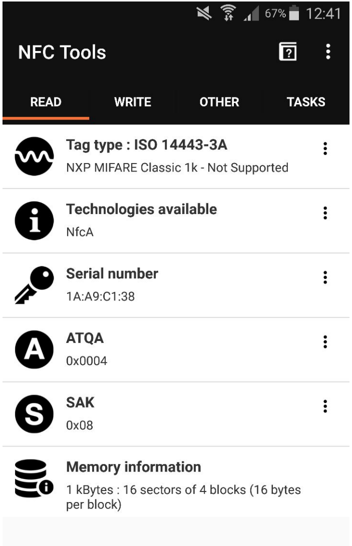
1. Physical characteristics
2. Radio frequency power and signal interface
3. Initialization and anticollision
4. Transmission protocol

Mifare Classic

ISO/IEC 14443-A (Mifare classic is not compatible with the 4th)

1. Physical characteristics
2. Radio frequency power and signal interface
3. Initialization and anticollision
4. Transmission protocol

Mifare Classic



CRYPTO1

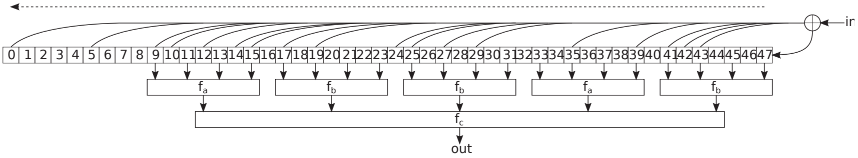


Figure: Structure of CRYPTO1 stream cipher

$$
\begin{aligned}
L := \;& x_0 \oplus x_5 \oplus x_9 \oplus x_{10} \oplus x_{12} \oplus x_{14} \oplus x_{15} \oplus x_{17} \oplus \\
& x_{19} \oplus x_{24} \oplus x_{25} \oplus x_{27} \oplus x_{29} \oplus x_{35} \oplus x_{39} \oplus x_{41} \oplus x_{42} \oplus x_{43}
\end{aligned}
$$

$$
\begin{aligned}
f_a(y_0, y_1, y_2, y_3) : &= ((y_0 \vee y_1) \oplus (y_0 \wedge y_3)) \oplus (y_2 \wedge ((y_0 \oplus y_1) \vee y_3)) \\
f_b(y_0, y_1, y_2, y_3) : &= ((y_0 \wedge y_1) \vee y_2) \oplus ((y_0 \oplus y_1) \wedge (y_2 \vee y_3)) \\
f_c(y_0, y_1, y_2, y_3, y_4) : &= (y_0 \vee ((y_1 \vee y_4) \wedge (y_3 \oplus y_4))) \oplus ((y_0 \oplus (y_1 \wedge y_3)) \wedge \\
& ((y_2 \oplus y_3) \vee (y_1 \wedge y_4)))
\end{aligned}
$$

$$
suc(x_0 x_1 \ldots x_{31}) := x_1 x_2 \ldots x_{31} || (x_{16} \oplus x_{18} \oplus x_{19} \oplus x_{21})
$$

Authentication Protocol

$$
\text{Tag} \left| \begin{array}{c} \xrightarrow{u} \\ \xrightarrow{n_T} \\ \xleftarrow{\{n_R\}\{a_R\}} \\ \xrightarrow{\{a_T\}} \end{array} \right| \text{Reader}
$$

Known Vulnerabilities

(Some) Known Vulnerabilities

1. **Weak PRNG:** Corrected in the hardened Mifare classic cards
2. **Short key length:** 48-bit key is too short!!! (Delay introduced by the communication and authentication procedure prevents online brute force attacks)
3. **Keystream leakage through error:** 4-bit leaks (Corrected in the hardened Mifare classic cards)
4. **Parity bits:** Parity of plaintext bytes are transmitted during communication
5. **Nested authentication:** When authentication for a sector for which the key is known is completed, the reader can request for authentication of another sector. This time tag sends encrypted n_T

Online Attacks

Table: Comparison of card only attacks.

Attack	Traces	Gather	Compute	a	b
(Garcia et al., 2009)	2	<1 sec	<1 sec	×	✓
(Courtois, 2009)	300	3 min	<1 sec	×	×
(Chiu et al., 2013)	∼ 100,000	10-20 hours	2-15 min	✓	×
(Meijer and Verdult, 2015)	∼ 10,000	6-12 min	5-10 min	✓	✓

[a] Does not require a weak PRNG
[b] Does not require the error code after a failed authentication

Offline Attacks

(Garcia et al., 2009)

- The attacker plays the role of a reader and tries to authenticate for a sector of her choice
- She answers the challenge of the tag with eight random bytes (and eight random parity bits) for n_R and a_R
- With probability 1/256, the parity bits are correct and the tag responds with the encrypted 4-bit error code
- A success leaks 12 bits of entropy (out of 48)
- In practice, 5 successful repetition of this attack is enough
- So we need 5×256 authentication attempts (takes less than a sec)
- 16 GTX280 GPUs can perform this attack in 14 hours

Table: Brute Force using Encrypted Error Code

Reference	GPU	Cores	Speed	Time
(Chih et al., 2010)	GTX 280	240	1296 MHz	9 days 8 hours

Our Bitsliced Brute Force using Encrypted Error Code

Table: Brute Force Attack on Mifare Classic 1k

	GTX 860M	GTX 970
Cores	640	1664
Clock	1020 MHz	1253 MHz
Keys per second	6,673 M	15,575 M
48-bit search	11.7 hours	5 hours

Offline Attacks on Hardened Mifare Classic

(Meijer and Verdult, 2015)

- Given an encrypted nonce obtained through a nested authentication, the adversary can attempt to decrypt it using the candidate key
- If the candidate is the correct key, the parity bits will be correct
- For a wrong key, a parity bit will be correct with probability $1/2$
- An encrypted nonce holds 4 bytes, thus 4 encrypted parity bits
- Therefore, on average, $48/4 = 12$ encrypted nonces are enough to get the key

Table: Brute Force on Hardened Mifare Classic

Reference	GPU	Cores	Speed	Time
(Meijer and Verdult, 2015)	GTX 460	336	1350 MHz	1 month

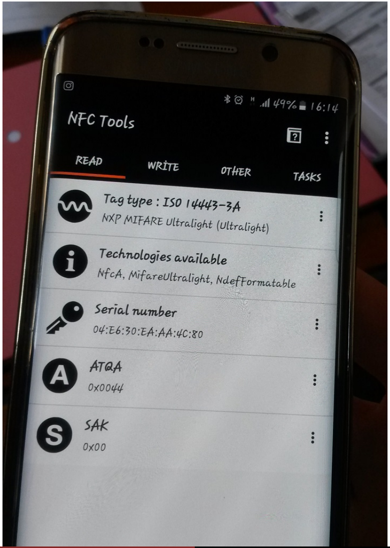
- 180 GTX460 GPUs can perform this attack in an hour and the system should cost around $12,600

Our Bitsliced Brute Force on Hardened Mifare Classic

Table: Brute Force Attack on Hardened Mifare Classic

	GTX 860M	GTX 970
Cores	640	1664
Clock	1020 MHz	1253 MHz
Keys per second	3,635 M	11,105 M
48-bit search	21 hours	7 hours

Mifare Ultralight



Lightweight Designs

Lightweight Designs

- Initial designs mainly focused on low hardware footprint but it is not realistic to have a single cipher to satisfy all needs
- To fit within constrained settings, lightweight ciphers rely on simpler round functions, or minimal key schedules
- The simpler structure of many of these ciphers may lend itself to new attacks

Lightweight Designs

Lightweight Designs

- Low hardware footprint
  1. PRESENT (ISO/IEC Lightweight Block Cipher Standard)
  2. HIGHT (ISO/IEC Block Cipher Standard)
  3. CLEFIA (ISO/IEC Lightweight Block Cipher Standard)
  4. LED
  5. KATAN
- Low memory consumption on small embedded processors
  1. ITUBee
  2. PRIDE
  3. SPECK
- Low latency
  1. PRINCE
- Ease of side-channel protection
  1. Zorro
  2. LS-Designs

HIGHT

HIGHT

- HIGHT is lightweight block cipher designed by Hong et al. (CHES 2006)
- It is shown to be highly convenient for extremely constrained devices (RFID tags and sensor networks)
- ISO/IEC Standard and standardized encryption algorithm in South Korea
  - Generalized Unbalanced Feistel Network
  - Block size: 64
  - Key size: 128
  - Number of Rounds: 32

Our Cryptanalysis of HIGHT and PRESENT

July 2008:

Discovered the related-key attack on HIGHT-31

24 November 2008:

Submitted to FSE 2009

20 January 2009:

Paper rejected from FSE 2009

- The ciphers PRESENT and HIGHT are in my opinion uninteresting targets for cryptanalysis. PRESENT has been designed with no actual diffusion layer completely ignoring the insights gained in the last 15 years of block cipher design and cryptanalysis. HIGHT has the problem that it alternates addition with XOR operation. This makes it very hard/expensive to protect it against DPA and as such HIGHT becomes unsuited for e.g. low-end smart cards and RFID, its supposed target platforms.

Lightweight Stream Cipher Standards

ISO/IEC 29192-3:2012 Lightweight Stream Cipher Standards

1. Enocoro (key sizes 80 or 128 bits)
2. Trivium (key size 80 bits)

ISO/IEC 29191-5 Lightweight Hash Functions

1. PHOTON
2. SPONGENT
3. Lesamnta-LW

ISO/IEC 29192-6:2019 Lightweight MAC Standards

1. LightMAC
2. Tsudik's keymode
3. Chaskey-12

Some Standardized Block Ciphers

Some Standardized Block Ciphers

ISO/IEC	**NIST**
TDEA	AES
MISTY1	TDEA (until 2024)
CAST-128	SKIPJACK (legacy use)
HIGHT (broken)	
AES	
Camellia	
SEED	
PRESENT (lightweight)	ASCON (since 2023)
CLEFIA (lightweight)	
LEA (lightweight)	

Licensing and standardized algorithms: Most of these ciphers (probably all) are free to use

NIST's Lightweight Cryptography Standardization Process

NISTIR 8114

Report on Lightweight Cryptography

Kerry A. McKay
Larry Bassham
Meltem Sönmez Turan
Nicky Mouha

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8114

NIST
National Institute of

NIST's Lightweight Cryptography Standardization Process

Evaluation criteria

Physical	**Performance**	**Security**
Area (GE or mm^2)	Latency	Minimum bit security
Memory (RAM/ROM)	Throughput	Attack Models
Implementation (hw/sfw)	Power (W)	Side channel resistance
Energy (J)		

NIST's Lightweight Cryptography Standardization Process

NIST's Lightweight Cryptography Standardization Process

- **July 2015** 1. Lightweight Cryptography Workshop
- **August 2016** (Draft) NISTIR 8114 Report
- **October 2016** 2. Lightweight Cryptography Workshop
- **March 2017** NISTIR 8114 Report
- **August 2018** Submission Requirements and Evaluation Criteria
- **February 2019** Submission Deadline
- **18 April 2019** Announcement of the 1. Round Candidates
- **30 August 2019** Announcement of the 2. Round Candidates
- **4-6 November 2019** 3. Lightweight Cryptography Workshop
- **19-21 October 2020** 4. Lightweight Cryptography Workshop
- **29 March 2020** Announcement of the Finalists
- **21 July 2021** NISTIR 8369 Status Report
- **9-11 May 2022** 5. Lightweight Cryptography Workshop
- **7 February 2023** ASCON Wins the Competition
- **21-22 June 2023** 6. Lightweight Cryptography Workshop

2. Round Candidates

ACE	ASCON	COMET	DryGASCON
Elephant	ESTATE	ForkAE	GIFT-COFB
Gimli	Grain-128AEAD	HYENA	ISAP
KNOT	LOTUS and LOCUS	mixFeed	ORANGE
Oribatida	PHOTON-Beetle	Pyjamask	Romulus
SAEAES	Saturnin	SKINNY	SPARKLE
SPIX	SpoC	Spook	Subterranean 2.0
SUNDAE-GIFT	TinyJambu	WAGE	Xoodyak

3. Round Candidates (FINALISTS)

1. ASCON
2. Elephant
3. GIFT-COFB
4. Grain128-AEAD
5. ISAP
6. Photon-Beetle
7. Romulus
8. Sparkle
9. TinyJambu
10. Xoodyak

ASCON

ASCON

- Designed by Christoph Dobraunig, Maria Eichlseder, Florian Mendel, Martin Schlaffer
- Type: Sponge construction
- Primitive: SPN
  - **Block size:** 64 or 128 bits
  - **State size:** 320 bits
  - **Key:** 128 bits (initial version supported 96 bits)
  - **Nonce:** 128 bits
  - **Tag:** 128 bits
  - **Rounds:** 12 or 6

ASCON

Properties

- Single-pass
- Online
- Inverse-free
- Security proof
- Lightweight
- Fast in software and hardware
- No table look-up (provides timing resistance against some side-channel attacks)

ASCON

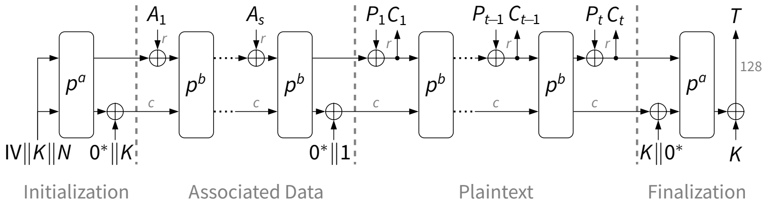


Figure: The encryption of ASCON. p^a means the permutation operation p is performed a times. We have $a = 12$ and $b = 6$.

ASCON

Figure: 320-bit state ASCON

ASCON

round	constant	round	constant
0	0x000000000000000000f0	6	0x00000000000000000096
1	0x000000000000000000e1	7	0x00000000000000000087
2	0x000000000000000000d2	8	0x00000000000000000078
3	0x000000000000000000c3	9	0x00000000000000000069
4	0x000000000000000000b4	10	0x0000000000000000005a
5	0x000000000000000000a5	11	0x0000000000000000004b

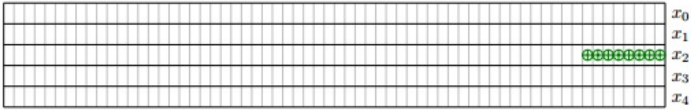
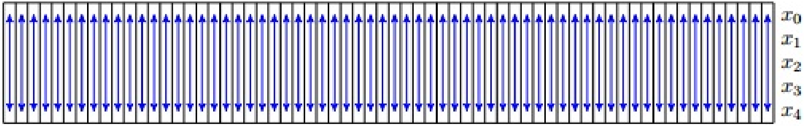


Figure: Adding constants.

ASCON



x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S(x)$	4	11	31	20	26	21	9	2	27	5	8	18	29	3	6	28
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$S(x)$	30	19	7	14	0	13	17	24	16	12	1	25	22	10	15	23

Figure: 5×5 S-box of Ascon

ASCON

$$\Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)$$
$$\Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)$$
$$\Sigma_2(x_2) = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6)$$
$$\Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)$$
$$\Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)$$

Figure: Linear Diffusion layer ASCON

ASCON Benchmarks

FPGA benchmarks

Xilinx Artix-7

	Throughput	Area	Throughput / Area
ASCON-128a	6297.6	2410	2.61
ASCON-128	3744.0	2126	1.76
AES128-GCM	2700.8	3270	0.83

Intel Cyclone 10 LP

	Throughput	Area	Throughput / Area
ASCON-128a	3031.0	4552	0.67
ASCON-128	2157.0	3215	0.67
AES128-GCM	1548.3	8754	0.18

Lattice ECP5

	Throughput	Area	Throughput / Area
ASCON-128a	2158.1	5909	0.37
ASCON-128	1427.5	3764	0.38
AES128-GCM	1384.4	6740	0.21

https://eprint.iacr.org/2020/1207

ASCON Benchmarks

ASIC benchmarks

	Throughput	Area	Throughput / Area
Ascon-128a	25.60	1.49	17.18
Ascon-128	16.00	1.56	10.25
AES128-GCM	11.63	2.75	4.22

https://eprint.iacr.org/2021/049

ASCON Benchmarks

Embedded SW implementations

Time to process NIST testvectors in [µs] on embedded devices

	Uno	F1	ESP	F7	R5
Ascon-128a	1981	66.4	18.4	11.8	7.3
Ascon-128	2337	76.7	22.3	13.8	8.5
AES128-GCM	-	332.8	67.2	35.8	23.7

https://lwc.las3.de/

Code size in [bytes] on embedded devices

	Uno	F1	ESP	F7	R5
Ascon-128a	2544	2252	1200	1240	1792
Ascon-128	2552	2157	1120	1180	1792
AES128-GCM	-	9908	14832	9836	14272

https://lwc.las3.de/

Summary

Summary

- IoT devices are very different than each other so it is hard to provide standards for all
- Current device production does not focus on security
- Producers should provide their own security solutions until IoT standards are available
- We may need different lightweight ciphers for different purposes
- Due to their simplicity, lightweight designs may be weak against attack types that are not discovered yet
- Lightweight does not mean shorter key. Using short keys provides almost no security. There is no need to use keys shorter than 128 bits

Teşekkürler

Teşekkürler

Mail: cihangir@metu.edu.tr
Udemy: cihangir-tezcan
CihangirTezcan
CihangirTezcan

This work has been supported by TUBITAK 1001 Project under the grant number 121E228 and by Middle East Technical University Scientific Research Projects Coordination Unit under grant number AGEP-704-2023-11294.