SNOW V: A new version of SNOW for 5G

Patrik Ekdahl[2], Thomas Johansson[1], Alexander Maximov[2], Jing Yang[1]

[1] Department of Electrical and Information Technology, Lund University

[2] Ericsson Research, Ericsson

Outline

- Motivation
  - Stream Ciphers
  - SNOW 3G
  - 5G Requirements
- SNOW V
  - Construction
  - Keystream Generation
  - AEAD Mode
- Performance Analysis
  - Hardware Implementation Aspects
  - Software Implementation Aspects
- Security Analysis
- Conclusion

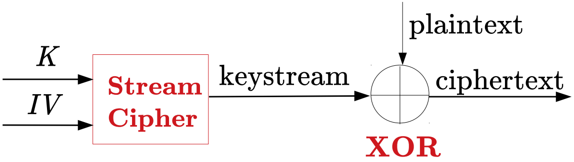
LUND UNIVERSITY

Outline

- Motivation
  - Stream Ciphers
  - SNOW 3G
  - 5G Requirements
- SNOW V
  - Construction
  - Keystream Generation
  - AEAD Mode
- Performance Analysis
  - Hardware Implementation Aspects
  - Software Implementation Aspects
- Security Analysis
- Conclusion

Stream Ciphers

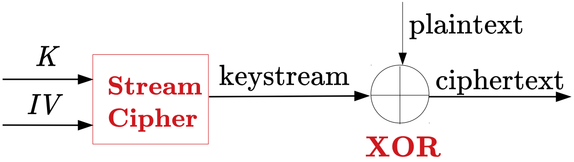
- Symmetric-key ciphers encrypt/decrypt data **digit by digit** through **XOR** operation



K : the secret key
IV : a public nonce

Stream Ciphers

- Symmetric-key ciphers encrypt/decrypt data **digit by digit** through **XOR** operation

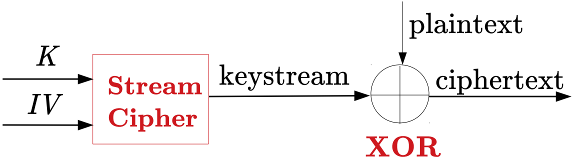


K : the secret key
IV : a public nonce

- Often constructed using linear-feedback shift registers (LFSRs) + a Non-Linear Part to disrupt the linearity of LFSR
  - **Easy implementation** and **very fast** in hardware environment

Stream Ciphers

- Symmetric-key ciphers encrypt/decrypt data **digit by digit** through **XOR** operation



K : the secret key
IV : a public nonce

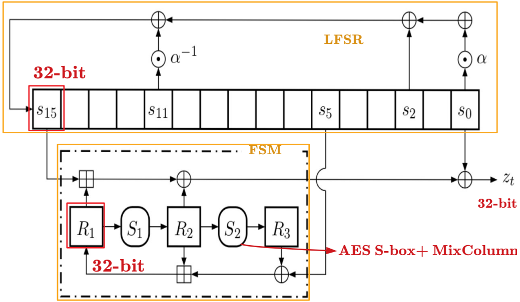
- Often constructed using linear-feedback shift registers (LFSRs) + a Non-Linear Part to disrupt the linearity of LFSR
  - **Easy implementation** and **very fast** in hardware environment
- Popular stream ciphers: Salsa20, Grain, SOBER, ***SNOW***, ZUC, etc.

SNOW 3G

- **SNOW 1.0**: Proposed by Thomas Johansson & Patrik Ekdahl in 2000, NESSIE candidate
- **SNOW 2.0**: Improved in 2003, included in ISO/IEC 18033-4 standard
- **SNOW 3G**: 2006, one of the three confidentiality/integrity algorithm standards for 3G/LTE

SNOW 3G

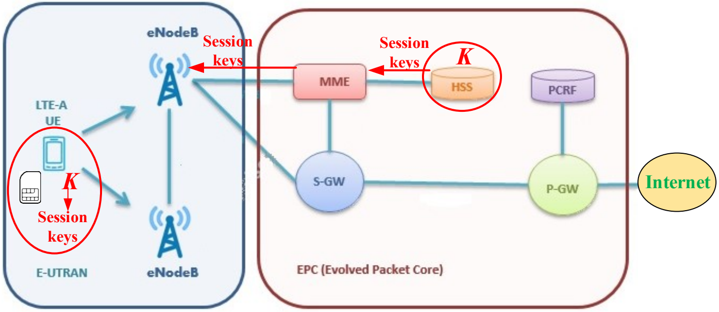
- **SNOW 1.0**: Proposed by Thomas Johansson & Patrik Ekdahl in 2000, NESSIE candidate
- **SNOW 2.0**: Improved in 2003, included in ISO/IEC 18033-4 standard
- **SNOW 3G**: 2006, one of the three confidentiality/integrity algorithm standards for 3G/LTE



- LFSR (512 bits in total) + Non-linear Part (FSM, finite state machine)
- Word-based, hardware-oriented, especially efficient in hardware environment

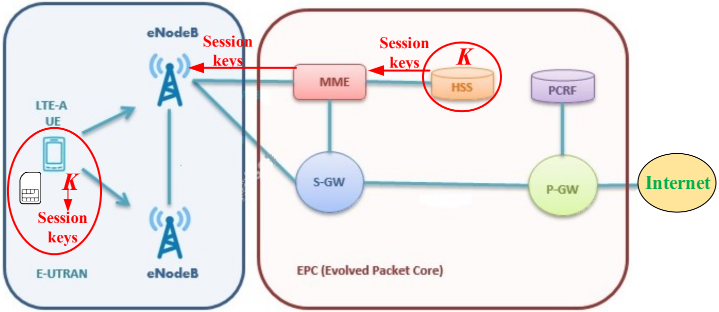
SNOW 3G Application

- Every user has a unique master key K embedded into the SIM card/ stored at HSS(Home Subscriber Server), to generate session keys and distribute to base stations (BSs) and Mobility Management Entity (MME)



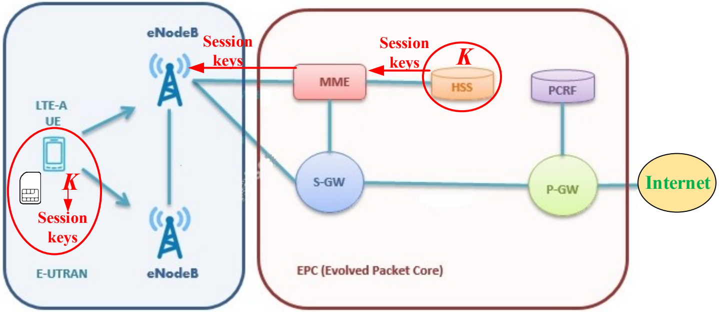
SNOW 3G Application

- Every user has a unique master key K embedded into the SIM card/ stored at HSS(Home Subscriber Server), to generate session keys and distribute to base stations (BSs) and Mobility Management Entity (MME)
- SNOW3G IP core is embeded into the physical boards of mobile phones / BS / MME
- User / BS / MME: keystream = SNOW3G($K_{session}$, IV)



SNOW 3G Application

- Every user has a unique master key K embedded into the SIM card/ stored at HSS(Home Subscriber Server), to generate session keys and distribute to base stations (BSs) and Mobility Management Entity (MME)
- SNOW3G IP core is embeded into the physical boards of mobile phones / BS / MME
- User / BS / MME: keystream = SNOW3G($K_{session}$, IV)
- Speed is lower than 20Gbps (the expected downlink speed in 5G)



5G

Challenges

- **Structure**: SDN-based, nodes are virtualized (No specific hardware cores)
- **Targeted data rate**: 20Gbps (downlink) 10Gbps (uplink)

LUND UNIVERSITY

LUND UNIVERSITY

Challenges

- **Structure**: SDN-based, nodes are virtualized (No specific hardware cores)
- **Targeted data rate**: 20Gbps (downlink) 10Gbps (uplink)

The speed of SNOW needs to be > 20 Gbps under software environment.

5G

Challenges

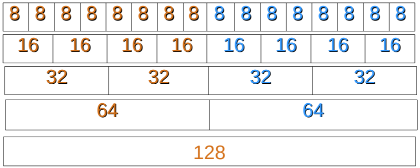
- **Structure**: SDN-based, nodes are virtualized (No specific hardware cores)
- **Targeted data rate**: 20Gbps (downlink) 10Gbps (uplink)

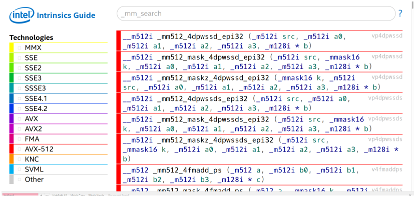
The speed of SNOW needs to be > 20 Gbps under software environment.

Opportunities

- **SIMD (Single Instruction Multiple Data) structure**: CPUs can handle large registers split into blocks of various sizes (8-, 16-, 32-, 64-, 128-, 256-, 512-bits)
- **Intrinsic instructions**: e.g., AES-NI set for AES, high speed in software

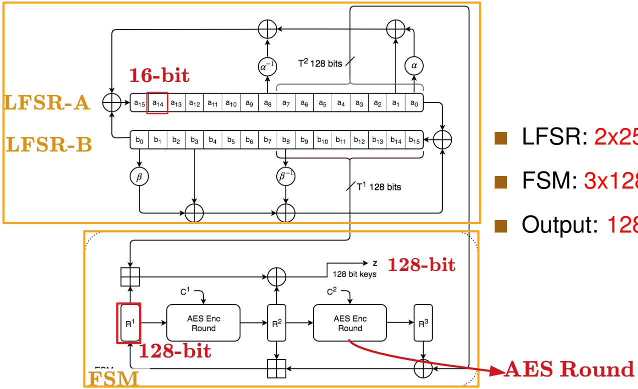
SIMD Structure





Outline

Construction



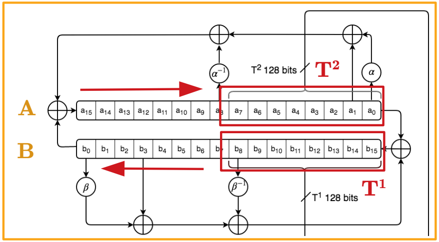
- LFSR: 2x256 bits
- FSM: 3x128-bit registers and 2 AES rounds
- Output: 128-bit keystream

	LFSRs	LFSR Stages	Stage Sizes	FSM Register Sizes	Output
SNOW 3G	1	16	32-bit	32-bit	32-bit
SNOW V	2	32	16 -bit	128-bit	128-bit

LUND UNIVERSITY

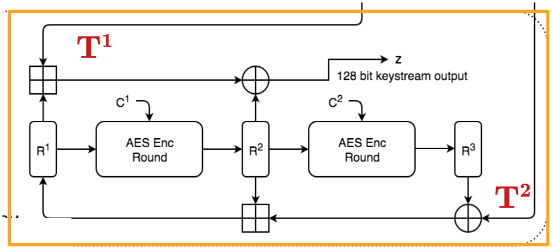
LFSR

- **Circular Construction**: Two LFSRs defined on two finite fields feeding to each other
  $g^A(x) = x^{16} + x^{15} + x^{12} + x^{11} + x^8 + x^3 + x^2 + x + 1 \in \mathbb{F}_2[x]$, with root α
  $g^B(x) = x^{16} + x^{15} + x^{14} + x^{11} + x^8 + x^6 + x^5 + x + 1 \in \mathbb{F}_2[x]$, with root β
- Proven to have a **maximum period** $2^{512} - 1$



procedure *LFSRupdate*()
  for $i = 0..7$ **do**
    $a_{16} \leftarrow b_0 + \alpha a_0 + a_1 + \alpha^{-1} a_8 \bmod g^A(\alpha)$
    $b_{16} \leftarrow a_0 + \beta b_0 + b_3 + \beta^{-1} b_8 \bmod g^B(\beta)$
    $A \leftarrow (a_{16}, a_{15}, \ldots, a_1)$
    $B \leftarrow (b_{16}, b_{15}, \ldots, b_1)$

FSM



procedure *FSMupdate*()

$T^2 \leftarrow (a_7, a_6, \ldots, a_0)$

$tmp \leftarrow R^2 \boxplus_{32} (R^3 \oplus T^2)$

$R^3 \leftarrow AES^R(R^2, C^2)$

$R^2 \leftarrow AES^R(R^1, C^1)$

$R^1 \leftarrow tmp$

Two round key constants C^1 and C^2 are set to zero.

Note: When used in AEAD mode, the value of C^1 is different (non-zero).

Keystream Generation

Algorithm 2 SNOW-V algorithm

procedure SNOW-V(k, iv)
  Declaration of internal parameters:
  $a = (a_{15}, a_{14}, \ldots, a_0)$
  $b = (b_{15}, b_{14}, \ldots, b_0)$
  R^1, R^2, R^3
  Initialization(k, iv)
  $i \leftarrow 0$
  while more keystream blocks needed **do**
    $T^1 \leftarrow (b_{15}, b_{14}, \ldots, b_8)$
    $z_i \leftarrow (R^1 \boxplus_{32} T^1) \oplus R^2$ → keystream
    $FSMupdate()$
    $LFSRupdate()$
    $i \leftarrow i + 1$

Algorithm 1 SNOW-V initialization

procedure INITIALIZATION(k, iv) — K/IV Setup
  $(a_{15}, a_{14}, \ldots, a_8) \leftarrow (k_7, k_6, \ldots, k_0)$
  $(a_7, a_6, \ldots, a_0) \leftarrow (iv_7, iv_6, \ldots, iv_0)$
  $(b_{15}, b_{14}, \ldots, b_8) \leftarrow (k_{15}, k_{14}, \ldots, k_8)$
  $(b_7, b_6, \ldots, b_0) \leftarrow (0, 0, \ldots, 0)$
  $R^1, R^2, R^3 \leftarrow 0, 0, 0$ — 16 rounds
  for $i = 0 \ldots 15$ **do**
    $z \leftarrow (R^1 \boxplus_{32} T^1) \oplus R^2$
    $FSMupdate()$ — keystream feeds
    $LFSRupdate()$ — back to LFSR
    $(a_{15}, a_{14}, \ldots, a_8) \leftarrow (a_{15}, a_{14}, \ldots, a_8) \oplus z$

Initialization is used to fully mix K and IV, after which the output should be random.

AEAD Mode

- **AEAD**: authenticated encryption with associated data, provides confidentiality, integrity, and authenticity assurances on the data

AEAD Mode

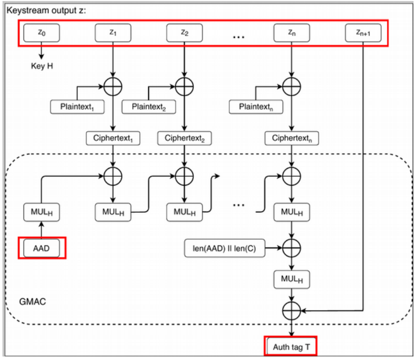
- **AEAD**: authenticated encryption with associated data, provides confidentiality, integrity, and authenticity assurances on the data
- GMAC (Galois Message Authentication Code) is used to generate authentication tag

AEAD Mode

- **AEAD**: authenticated encryption with associated data, provides confidentiality, integrity, and authenticity assurances on the data
- GMAC (Galois Message Authentication Code) is used to generate authentication tag
- Keystream generation process is the same as in the normal mode, except $C^1 = 0x0024406480A4C0E40420446084A0C4E0$

AEAD Mode

- **AEAD**: authenticated encryption with associated data, provides confidentiality, integrity, and authenticity assurances on the data
- GMAC (Galois Message Authentication Code) is used to generate authentication tag
- Keystream generation process is the same as in the normal mode, except $C^1 = 0x0024406480A4C0E40420446084A0C4E0$



Sender:

Ciphertext=keystream1 ⊕ Plaintext

T = GMAC (keystream2, AAD, Ciphertext)

Receiver:

T' =GMAC (keystream2, AAD, Ciphertext),

if T' = T

Plaintext=keystream1 ⊕ Ciphertext

else

Output *Fail* (data might be tampered)

Outline

Hardware Implementation Aspects

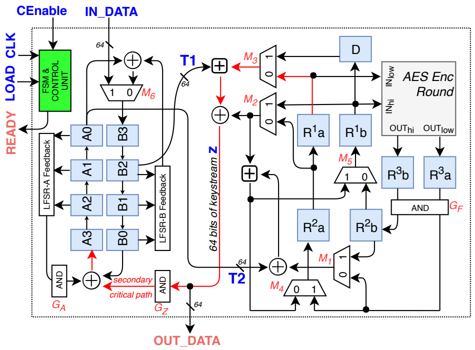
Four Hardware Implementations:

- SNOW V+1 external AES
- SNOW V+1 internal AES
- SNOW V+2 external AESs
- SNOW V+2 internal AESs

Hardware Implementation Aspects

Four Hardware Implementations:

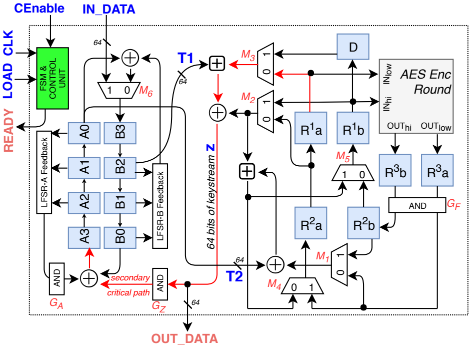
- SNOW V+1 external AES
- SNOW V+1 internal AES
- SNOW V+2 external AESs
- SNOW V+2 internal AESs



Hardware Implementation Aspects

Four Hardware Implementations:

- SNOW V+1 external AES
- SNOW V+1 internal AES
- SNOW V+2 external AESs
- SNOW V+2 internal AESs



Hardware design	AES256 from [1]	64-snow v external 1 AES core	64-snow v internal 1 AES Enc	128-snow v external 2 AES cores	128-snow v internal 2 AES Enc
Area(GE)	17232	8125	12099	10480	18428
Speed (Gbps)	50.85	358	358-500	712	712-1000

LUND UNIVERSITY

Software Implementation Aspects

Taking advantage of modern CPUs':

- SIMD structure:
  - Two LFSRs can fit into 2x 256-bit registers: __m256i
  - Registers in FSM can fit into 3x 128-bit registers: __m128i
- Intrinsic instructions, e.g.,
  - AES round: _mm_aesenc_si128(__m128i a, __m128i RoundKey)
  - Arithmetic additions: _mm_add_epi32(__m128i a, __m128i b)

Speed incl. initialization	Size of plaintext (bytes)				
	$2^{32}+$	2048	256	64	16
AES256	9.17 Gbps	8.48 Gbps	7.98 Gbps	6.75 Gbps	2.62 Gbps
SNOW V	61.18 Gbps	56.55 Gbps	27.55 Gbps	10.46 Gbps	3.04 Gbps

LUND UNIVERSITY

Outline

- Motivation
  - Stream Ciphers
  - SNOW 3G
  - 5G Requirements
- SNOW V
  - Construction
  - Keystream Generation
  - AEAD Mode
- Performance Analysis
  - Hardware Implementation Aspects
  - Software Implementation Aspects
- **Security Analysis**
- Conclusion

LUND UNIVERSITY

Security Analysis

Common Attacks on Stream Ciphers:

- **Attack on Initialization**
  - Chosen-IV attack: adversary attempts to build a distinguisher to introduce randomness failures in the ouput by setting arbitrary IV values, e.g., MDM attack
  - Differential Attacks: trace differences' transfer and discover where the cipher behaves non-random
- **Linear Distinguishing Attacks**

  Distinguish the cipher from random oracle
- **Time-Memory-Data Tradeoff Attacks**

  Balance/reduce one/two parameters in favor of the others
- **Slide Attacks**

  Analyze the key schedule and exploit weaknesses in it to break the cipher
- **Attacks on the Authentication Mode**

Security Analysis

Common Attacks on Stream Ciphers:

- **Attack on Initialization**
  - Chosen-IV attack: adversary attempts to build a distinguisher to introduce randomness failures in the ouput by setting arbitrary IV values, e.g., MDM attack
  - Differential Attacks: trace differences' transfer and discover where the cipher behaves non-random
- **Linear Distinguishing Attacks**

  Distinguish the cipher from random oracle
- **Time-Memory-Data Tradeoff Attacks**

  Balance/reduce one/two parameters in favor of the others
- **Slide Attacks**

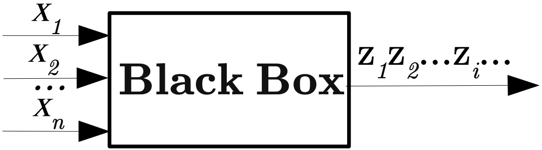
  Analyze the key schedule and exploit weaknesses in it to break the cipher
- **Attacks on the Authentication Mode**

LUND UNIVERSITY

MDM Attack

MDM: Maximum Degree Monomial

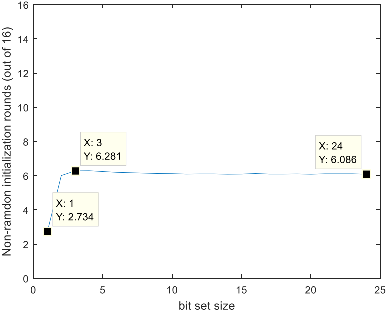
Rationale: Every cipher can be regarded as a black box with a series of Boolean functions (in SNOW V initialization, we have (128 x 16 =2048) Boolean functions)



$$z_i = f_i(x_1, x_2, \dots, x_n) = c_0 + c_1 x_1 + \dots + c_{12..n} x_1 x_2 \dots x_n$$

- $c_0, c_1, \dots, c_{12\dots n}$ should be 0 or 1 with probability of 0.5
- MDM : $c_{12\dots n} = \bigoplus_{x \in \{0,1\}^n} f_i(x)$
- Run through all possible input values, and xor the corresponding outputs to get MDM

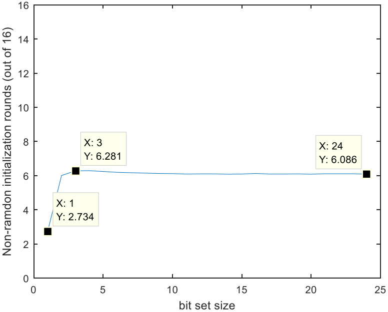
MDM Attack on SNOW V



- Select 1 to 24 bits from the (K,IV) space
- Run through all possible values, other bits are set 0
- Xor all the outputs to get the MDM
- The results have a long zeros before random-like, e.g., 000...00010110...

LUND UNIVERSITY

MDM Attack on SNOW V



- Select 1 to 24 bits from the (*K*,*IV*) space
- Run through all possible values, other bits are set 0
- Xor all the outputs to get the MDM
- The results have a long zeros before random-like, e.g., 000...00010110...

- The outputs of the first 7 rounds are not random, it would be not safe if we reduce the initialization rounds to 7 or fewer
- 16 rounds of initialization looks safe, it is not likely that an attacker would be able to build a distinguisher after 16 rounds

Outline

- Motivation
  - Stream Ciphers
  - SNOW 3G
  - 5G Requirements
- SNOW V
  - Construction
  - Keystream Generation
  - AEAD Mode
- Performance Analysis
  - Hardware Implementation Aspects
  - Software Implementation Aspects
- Security Analysis
- Conclusion

LUND UNIVERSITY

Conclusion

- We revised SNOW 3G to SNOW V to meet the 5G requirements on encryption speed under software environment, by taking advantage of modern CPUs':
  - SIMD structure to handle large registers and,
  - Intristic hardware-supported instructions
- In software, Snow V can perform up to ˜60Gbps on a user-grade laptop (single thread); it performs faster than AES256 utilizing AES-NI.
- In hardware, Snow V can reach up to ˜1Tbps.
- **Current status**: Security analysis is ongoing