POLITECNICO
MILANO 1863

PhD Course: Digital Design of Embedded Systems in the IoT and RISC-V Open Core Era

Hardware Design and Implementation of the Salsa20 Cryptosystem

Andrea Maioli (Politecnico di Milano, Italy)

Stream cipher

Input stream

plaintext
(ciphertext)

SALSA20

Encryption
Decryption

Key Expansion

Output stream

ciphertext
(plaintext)

Key

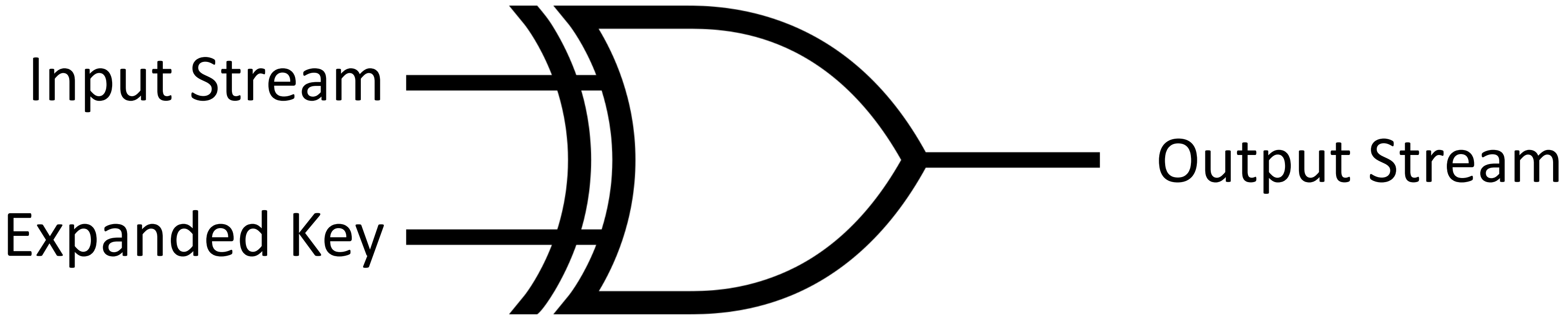
16 or 32 Bytes

Nonce

8 Bytes

Encryption / Decryption

- Salsa20 relies on a **One Time Pad** to encrypt/decrypt an input stream
  - **XOR** operator

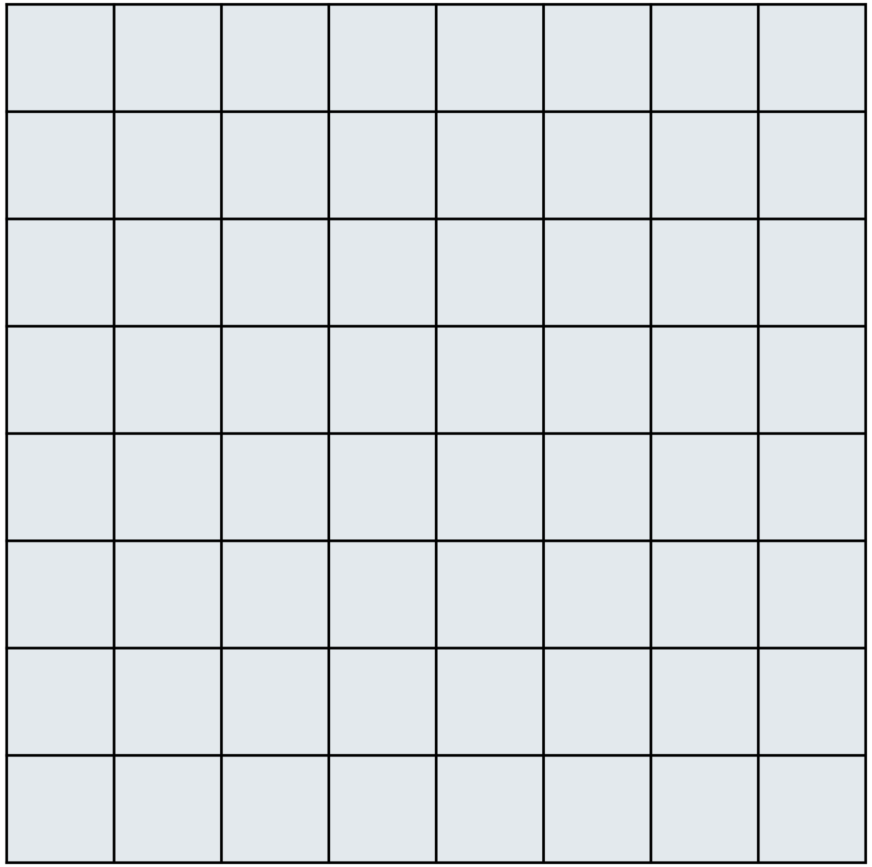


- XOR is an **involutory function**
  - Encryption and decryption functions are the **same**
  - **Low** hardware resource utilization

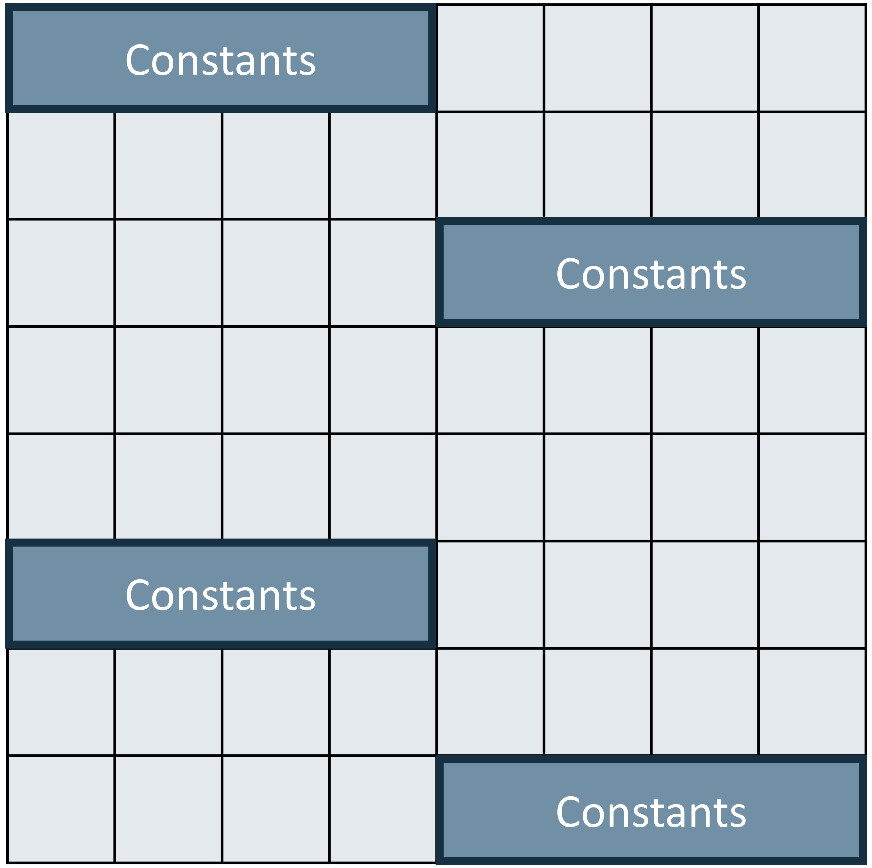
Key Expansion

1. Create a 64 Bytes matrix

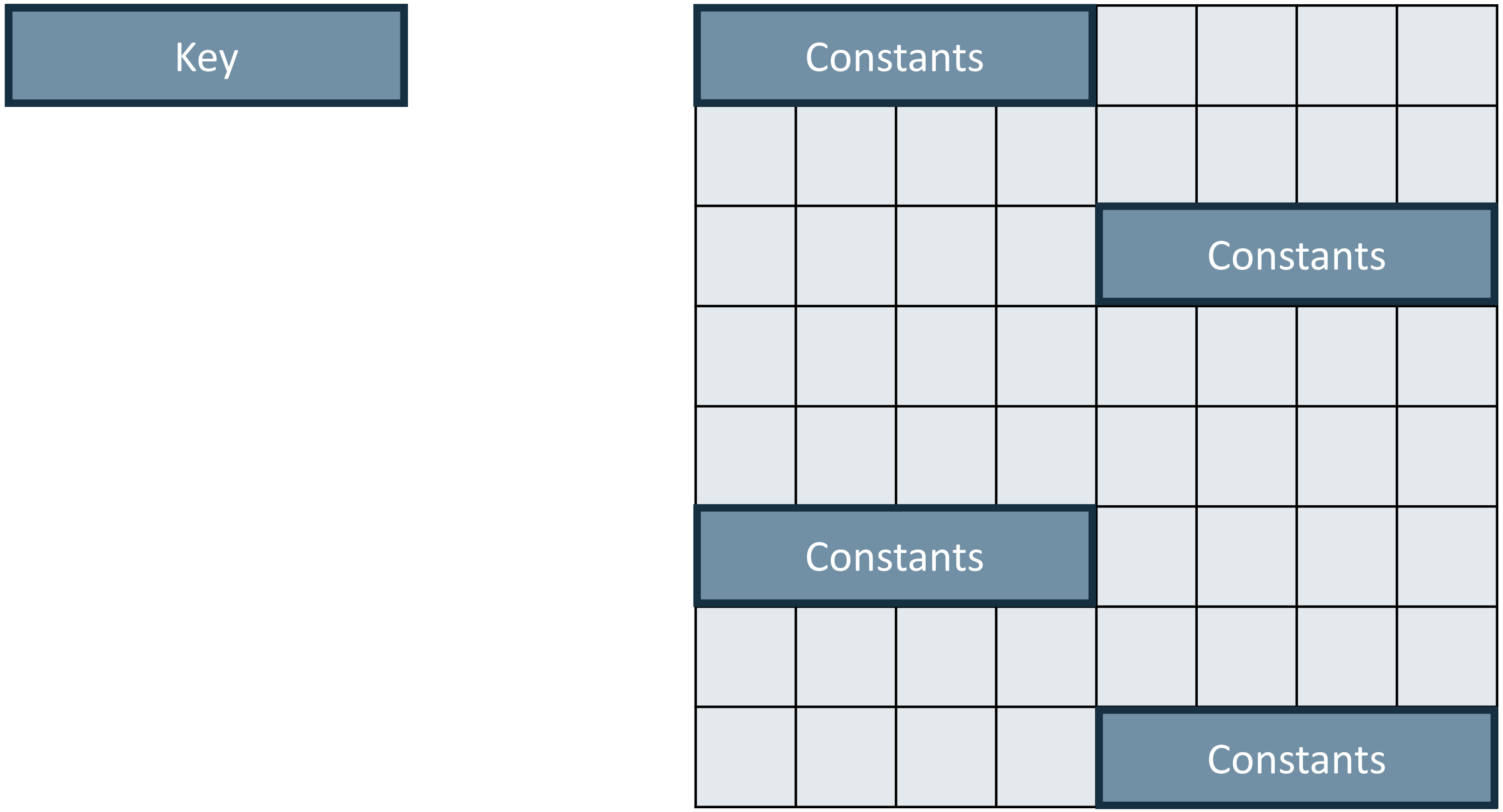
Constants



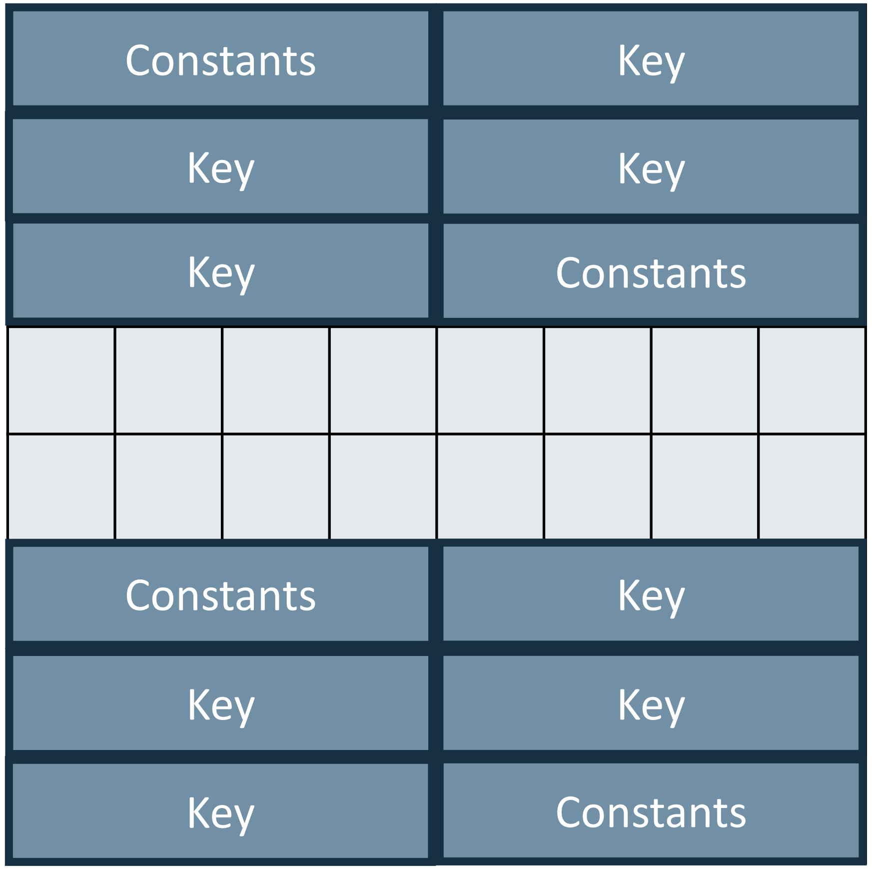
1. **Create a 64 Bytes matrix**



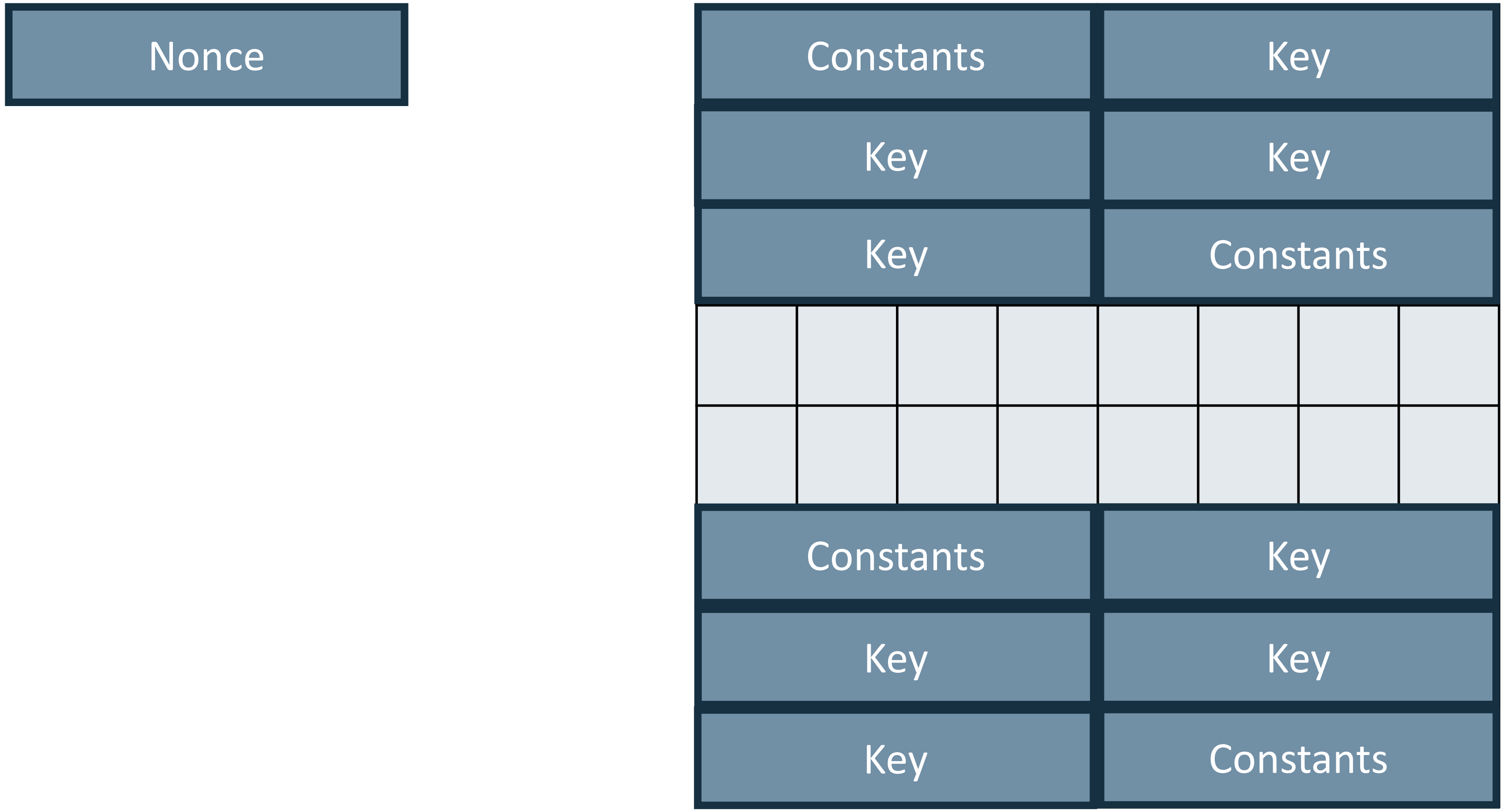
1. **Create a 64 Bytes matrix**



1. **Create a 64 Bytes matrix**



1. Create a 64 Bytes matrix



1. **Create a 64 Bytes matrix**

Constants	Key
Key	Key
Key	Constants
Nonce	Nonce
Constants	Key
Key	Key
Key	Constants

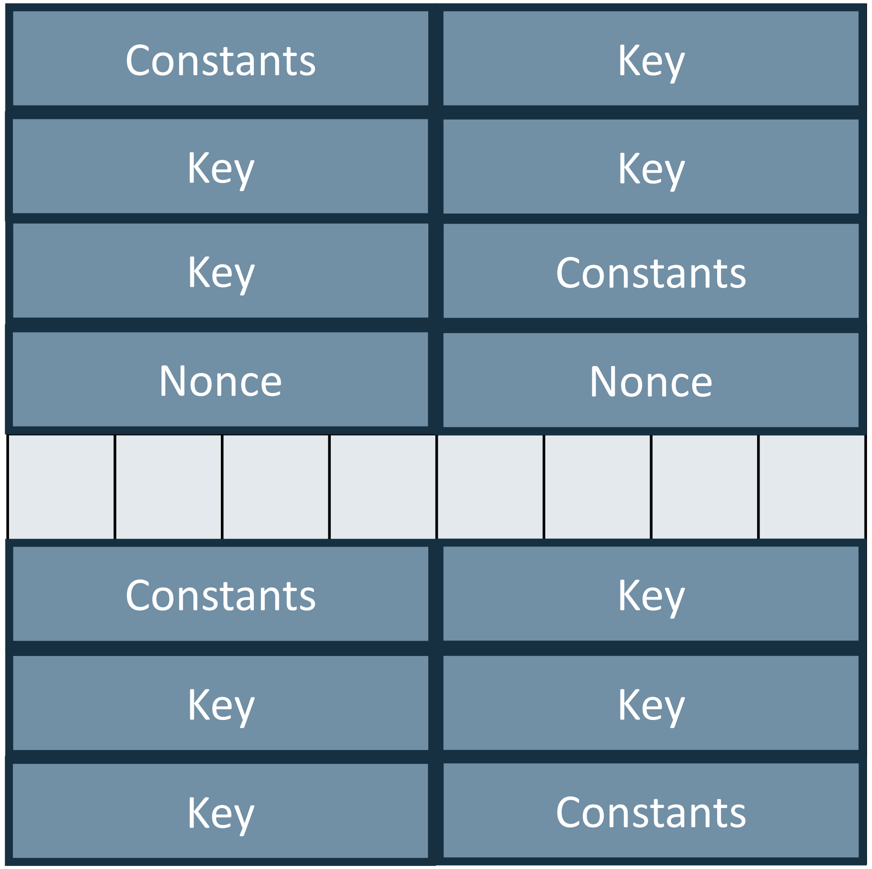
NETWORKED EMBEDDED SOFTWARE LAB

1. Create a 64 Bytes matrix

BLOCK ID

Block Identifier:
- 8 Bytes
- Allows to generate **unique keys**

- **Initialized** to **0**
- **Incremented** every 64 bytes of input



1. **Create a 64 Bytes matrix**

Constants	Key
Key	Key
Key	Constants
Nonce	Nonce
BLOCK ID	BLOCK ID
Constants	Key
Key	Key
Key	Constants

2. Transform the matrix

C_1	C_2	C_3	C_4	K_1	K_2	K_3	K_4
K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}	K_{12}
K_{13}	K_{14}	K_{15}	K_{16}	C_5	C_6	C_7	C_8
N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
C_9	C_{10}	C_{11}	C_{12}	K_{17}	K_{18}	K_{19}	K_{20}
K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	K_{27}	K_{28}
K_{29}	K_{30}	K_{31}	K_{32}	C_{13}	C_{14}	C_{15}	C_{16}

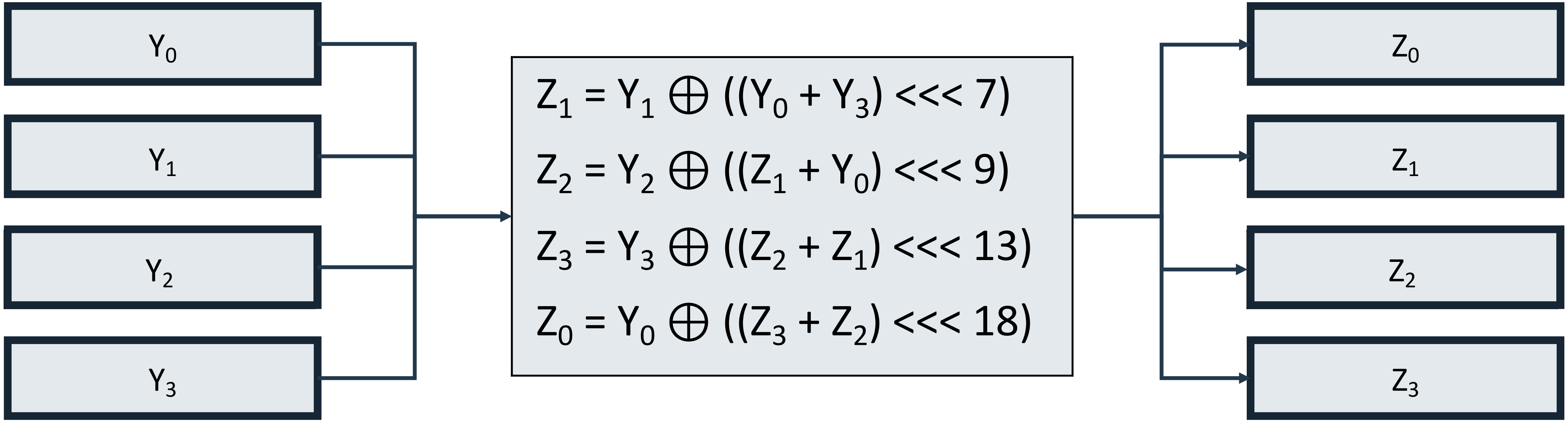
2. Transform the matrix – QuarterRound function

C_1	C_2	C_3	C_4	K_1	K_2	K_3	K_4
K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}	K_{12}
K_{13}	K_{14}	K_{15}	K_{16}	C_5	C_6	C_7	C_8
N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
C_9	C_{10}	C_{11}	C_{12}	K_{17}	K_{18}	K_{19}	K_{20}
K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	K_{27}	K_{28}
K_{29}	K_{30}	K_{31}	K_{32}	C_{13}	C_{14}	C_{15}	C_{16}

2. Transform the matrix – QuarterRound function

C_1	C_2	C_3	C_4
K_1	K_2	K_3	K_4
K_5	K_6	K_7	K_8
K_9	K_{10}	K_{11}	K_{12}

2. Transform the matrix – QuarterRound function



2. Transform the matrix – QuarterRound function

Z_0

Z_1

Z_2

Z_3

2. Transform the matrix – QuarterRound function

Z_0				Z_1			
Z_2				Z_3			
K_{13}	K_{14}	K_{15}	K_{16}	C_5	C_6	C_7	C_8
N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
C_9	C_{10}	C_{11}	C_{12}	K_{17}	K_{18}	K_{19}	K_{20}
K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	K_{27}	K_{28}
K_{29}	K_{30}	K_{31}	K_{32}	C_{13}	C_{14}	C_{15}	C_{16}

2. Transform the matrix – RowRound function

C_1	C_2	C_3	C_4	K_1	K_2	K_3	K_4
K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}	K_{12}
K_{13}	K_{14}	K_{15}	K_{16}	C_5	C_6	C_7	C_8
N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
C_9	C_{10}	C_{11}	C_{12}	K_{17}	K_{18}	K_{19}	K_{20}
K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	K_{27}	K_{28}
K_{29}	K_{30}	K_{31}	K_{32}	C_{13}	C_{14}	C_{15}	C_{16}

2. Transform the matrix – RowRound function

C_1	C_2	C_3	C_4	K_1	K_2	K_3	K_4
K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}	K_{12}
K_{13}	K_{14}	K_{15}	K_{16}	C_5	C_6	C_7	C_8
N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
C_9	C_{10}	C_{11}	C_{12}	K_{17}	K_{18}	K_{19}	K_{20}
K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	K_{27}	K_{28}
K_{29}	K_{30}	K_{31}	K_{32}	C_{13}	C_{14}	C_{15}	C_{16}

QuarterRound

QuarterRound

QuarterRound

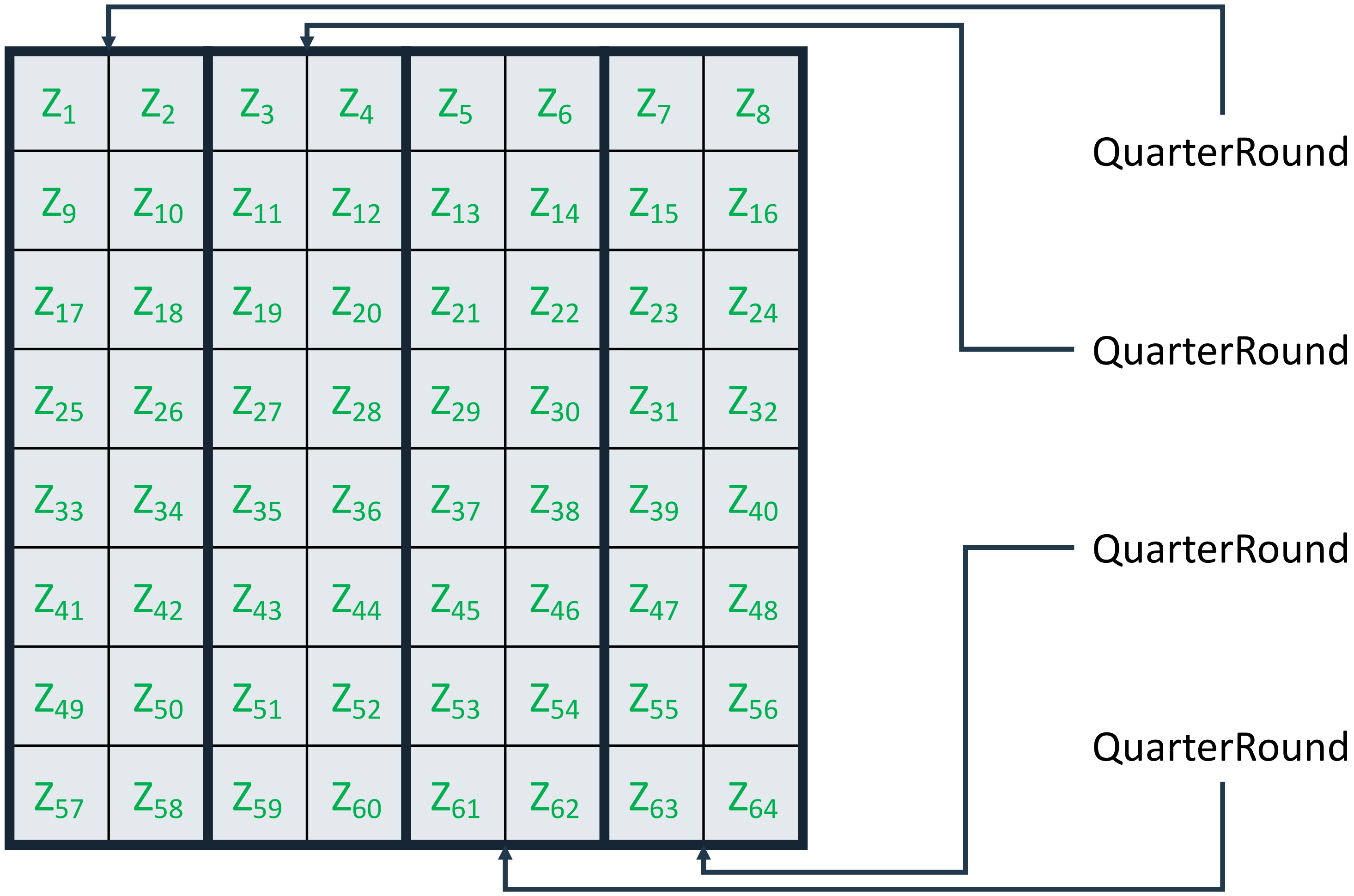
QuarterRound

Salsa20 Cryptosystem – Key Expansion

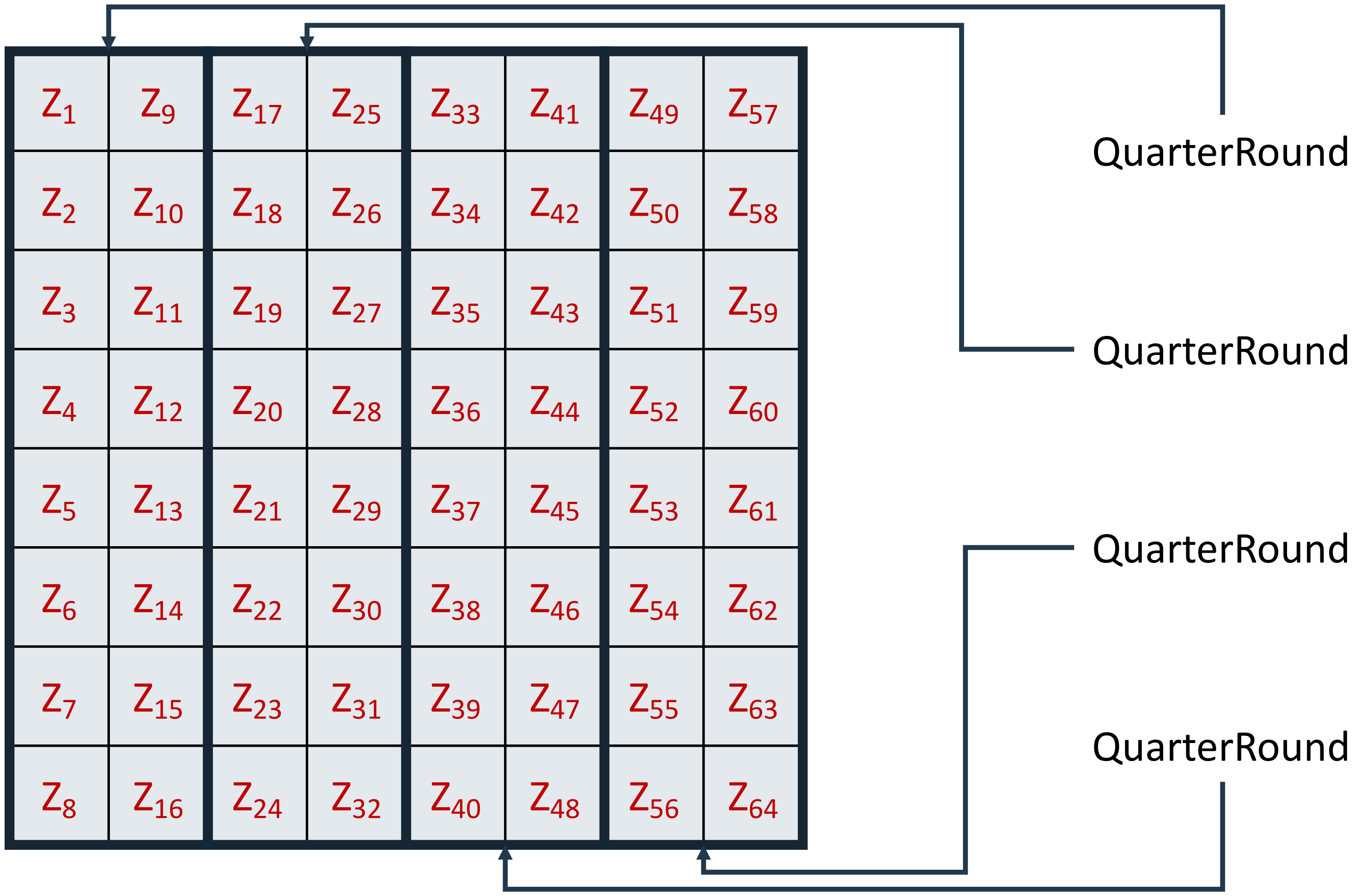
2. Transform the matrix – RowRound function

Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8	QuarterRound
Z_9	Z_{10}	Z_{11}	Z_{12}	Z_{13}	Z_{14}	Z_{15}	Z_{16}	
Z_{17}	Z_{18}	Z_{19}	Z_{20}	Z_{21}	Z_{22}	Z_{23}	Z_{24}	QuarterRound
Z_{25}	Z_{26}	Z_{27}	Z_{28}	Z_{29}	Z_{30}	Z_{31}	Z_{32}	
Z_{33}	Z_{34}	Z_{35}	Z_{36}	Z_{37}	Z_{38}	Z_{39}	Z_{40}	QuarterRound
Z_{41}	Z_{42}	Z_{43}	Z_{44}	Z_{45}	Z_{46}	Z_{47}	Z_{48}	
Z_{49}	Z_{50}	Z_{51}	Z_{52}	Z_{53}	Z_{54}	Z_{55}	Z_{56}	QuarterRound
Z_{57}	Z_{58}	Z_{59}	Z_{60}	Z_{61}	Z_{62}	Z_{63}	Z_{64}	

2. Transform the matrix – ColumnRound function



2. Transform the matrix – ColumnRound function



2. Transform the matrix – DoubleRound function

C_1	C_2	C_3	C_4	K_1	K_2	K_3	K_4
K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}	K_{12}
K_{13}	K_{14}	K_{15}	K_{16}	C_5	C_6	C_7	C_8
N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
C_9	C_{10}	C_{11}	C_{12}	K_{17}	K_{18}	K_{19}	K_{20}
K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	K_{27}	K_{28}
K_{29}	K_{30}	K_{31}	K_{32}	C_{13}	C_{14}	C_{15}	C_{16}

$\xrightarrow{\text{RowRound + ColumnRound}}$

Z_1	Z_9	Z_{17}	Z_{25}	Z_{33}	Z_{41}	Z_{49}	Z_{57}
Z_2	Z_{10}	Z_{18}	Z_{26}	Z_{34}	Z_{42}	Z_{50}	Z_{58}
Z_3	Z_{11}	Z_{19}	Z_{27}	Z_{35}	Z_{43}	Z_{51}	Z_{59}
Z_4	Z_{12}	Z_{20}	Z_{28}	Z_{36}	Z_{44}	Z_{52}	Z_{60}
Z_5	Z_{13}	Z_{21}	Z_{29}	Z_{37}	Z_{45}	Z_{53}	Z_{61}
Z_6	Z_{14}	Z_{22}	Z_{30}	Z_{38}	Z_{46}	Z_{54}	Z_{62}
Z_7	Z_{15}	Z_{23}	Z_{31}	Z_{39}	Z_{47}	Z_{55}	Z_{63}
Z_8	Z_{16}	Z_{24}	Z_{32}	Z_{40}	Z_{48}	Z_{56}	Z_{64}

2. Transform the matrix – DoubleRound function

C_1	C_2	C_3	C_4	K_1	K_2	K_3	K_4
K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}	K_{12}
K_{13}	K_{14}	K_{15}	K_{16}	C_5	C_6	C_7	C_8
N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
C_9	C_{10}	C_{11}	C_{12}	K_{17}	K_{18}	K_{19}	K_{20}
K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	K_{27}	K_{28}
K_{29}	K_{30}	K_{31}	K_{32}	C_{13}	C_{14}	C_{15}	C_{16}

$\xrightarrow{\text{DoubleRound}}$

Z_1	Z_9	Z_{17}	Z_{25}	Z_{33}	Z_{41}	Z_{49}	Z_{57}
Z_2	Z_{10}	Z_{18}	Z_{26}	Z_{34}	Z_{42}	Z_{50}	Z_{58}
Z_3	Z_{11}	Z_{19}	Z_{27}	Z_{35}	Z_{43}	Z_{51}	Z_{59}
Z_4	Z_{12}	Z_{20}	Z_{28}	Z_{36}	Z_{44}	Z_{52}	Z_{60}
Z_5	Z_{13}	Z_{21}	Z_{29}	Z_{37}	Z_{45}	Z_{53}	Z_{61}
Z_6	Z_{14}	Z_{22}	Z_{30}	Z_{38}	Z_{46}	Z_{54}	Z_{62}
Z_7	Z_{15}	Z_{23}	Z_{31}	Z_{39}	Z_{47}	Z_{55}	Z_{63}
Z_8	Z_{16}	Z_{24}	Z_{32}	Z_{40}	Z_{48}	Z_{56}	Z_{64}

2. Transform the matrix – DoubleRound function

- DoubleRound applied *n* times

- *n* = number of **rounds** in the Salsa **version**
  - Salsa20 -> *n = 10*
  - Salsa12 -> *n = 6*
  - Salsa8 -> *n = 4*

NETWORKED EMBEDDED SOFTWARE LAB

3. Compute the expanded key

C_1	C_2	C_3	C_4	K_1	K_2	K_3
K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}
K_{13}	K_{14}	K_{15}	K_{16}	C_5	C_6	C_7
N_1	N_2	N_3	N_4	N_5	N_6	N_7
B_1	B_2	B_3	B_4	B_5	B_6	B_7
C_9	C_{10}	C_{11}	C_{12}	K_{17}	K_{18}	K_{19}
K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	K_{27}
K_{29}	K_{30}	K_{31}	K_{32}	C_{13}	C_{14}	C_{15}

K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8
K_9	K_{10}	K_{11}	K_{12}	K_{13}	K_{14}	K_{15}	K_{16}
K_{17}	K_{18}	K_{19}	K_{20}	K_{21}	K_{22}	K_{23}	K_{24}
K_{25}	K_{26}	K_{27}	K_{28}	K_{29}	K_{30}	K_{31}	K_{32}
K_{33}	K_{34}	K_{35}	K_{36}	K_{37}	K_{38}	K_{39}	K_{40}
K_{41}	K_{42}	K_{43}	K_{44}	K_{45}	K_{46}	K_{47}	K_{48}
K_{49}	K_{50}	K_{51}	K_{52}	K_{53}	K_{54}	K_{55}	K_{56}
K_{57}	K_{58}	K_{59}	K_{60}	K_{61}	K_{62}	K_{63}	K_{64}

Z_9	Z_{17}	Z_{25}	Z_{33}	Z_{41}	Z_{49}	Z_{57}
Z_{10}	Z_{18}	Z_{26}	Z_{34}	Z_{42}	Z_{50}	Z_{58}
Z_{11}	Z_{19}	Z_{27}	Z_{35}	Z_{43}	Z_{51}	Z_{59}
Z_{12}	Z_{20}	Z_{28}	Z_{36}	Z_{44}	Z_{52}	Z_{60}
Z_{13}	Z_{21}	Z_{29}	Z_{37}	Z_{45}	Z_{53}	Z_{61}
Z_{14}	Z_{22}	Z_{30}	Z_{38}	Z_{46}	Z_{54}	Z_{62}
Z_{15}	Z_{23}	Z_{31}	Z_{39}	Z_{47}	Z_{55}	Z_{63}
Z_{16}	Z_{24}	Z_{32}	Z_{40}	Z_{48}	Z_{56}	Z_{64}

3. Compute the expanded key

K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8
K_9	K_{10}	K_{11}	K_{12}	K_{13}	K_{14}	K_{15}	K_{16}
K_{17}	K_{18}	K_{19}	K_{20}	K_{21}	K_{22}	K_{23}	K_{24}
K_{25}	K_{26}	K_{27}	K_{28}	K_{29}	K_{30}	K_{31}	K_{32}
K_{33}	K_{34}	K_{35}	K_{36}	K_{37}	K_{38}	K_{39}	K_{40}
K_{41}	K_{42}	K_{43}	K_{44}	K_{45}	K_{46}	K_{47}	K_{48}
K_{49}	K_{50}	K_{51}	K_{52}	K_{53}	K_{54}	K_{55}	K_{56}
K_{57}	K_{58}	K_{59}	K_{60}	K_{61}	K_{62}	K_{63}	K_{64}

The 64 bytes key is **valid** for 64 bytes of input

- **After** 64 bytes the block identifier is **incremented** and a **new key** is **computed**

3. Compute the expanded key

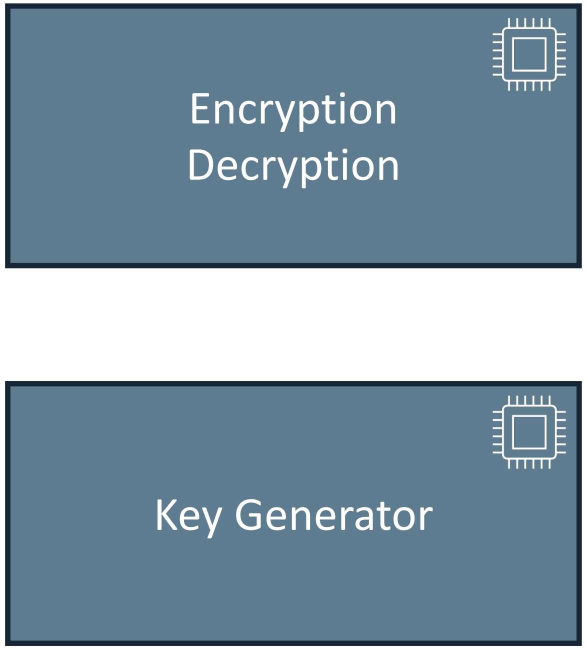
K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8
K_9	K_{10}	K_{11}	K_{12}	K_{13}	K_{14}	K_{15}	K_{16}
K_{17}	K_{18}	K_{19}	K_{20}	K_{21}	K_{22}	K_{23}	K_{24}
K_{25}	K_{26}	K_{27}	K_{28}	K_{29}	K_{30}	K_{31}	K_{32}
K_{33}	K_{34}	K_{35}	K_{36}	K_{37}	K_{38}	K_{39}	K_{40}
K_{41}	K_{42}	K_{43}	K_{44}	K_{45}	K_{46}	K_{47}	K_{48}
K_{49}	K_{50}	K_{51}	K_{52}	K_{53}	K_{54}	K_{55}	K_{56}
K_{57}	K_{58}	K_{59}	K_{60}	K_{61}	K_{62}	K_{63}	K_{64}

Salsa20 is a **stream cipher**

- Expanded key **does not rely** on **previous** encrypted or decrypted **data**
  - Key expansion requires **only** key, nonce, and block identifier
  - Main **difference** with respect to a **block cipher**
- **Advantages:**
  - The key can be **used** in **chunks**
    - **No** need to **wait** for 64 bytes of input data
  - **Multiple** keys can be **generated beforehand**

Salsa20 Hardware Design

- Same logical division of the Salsa20 cryptosystem

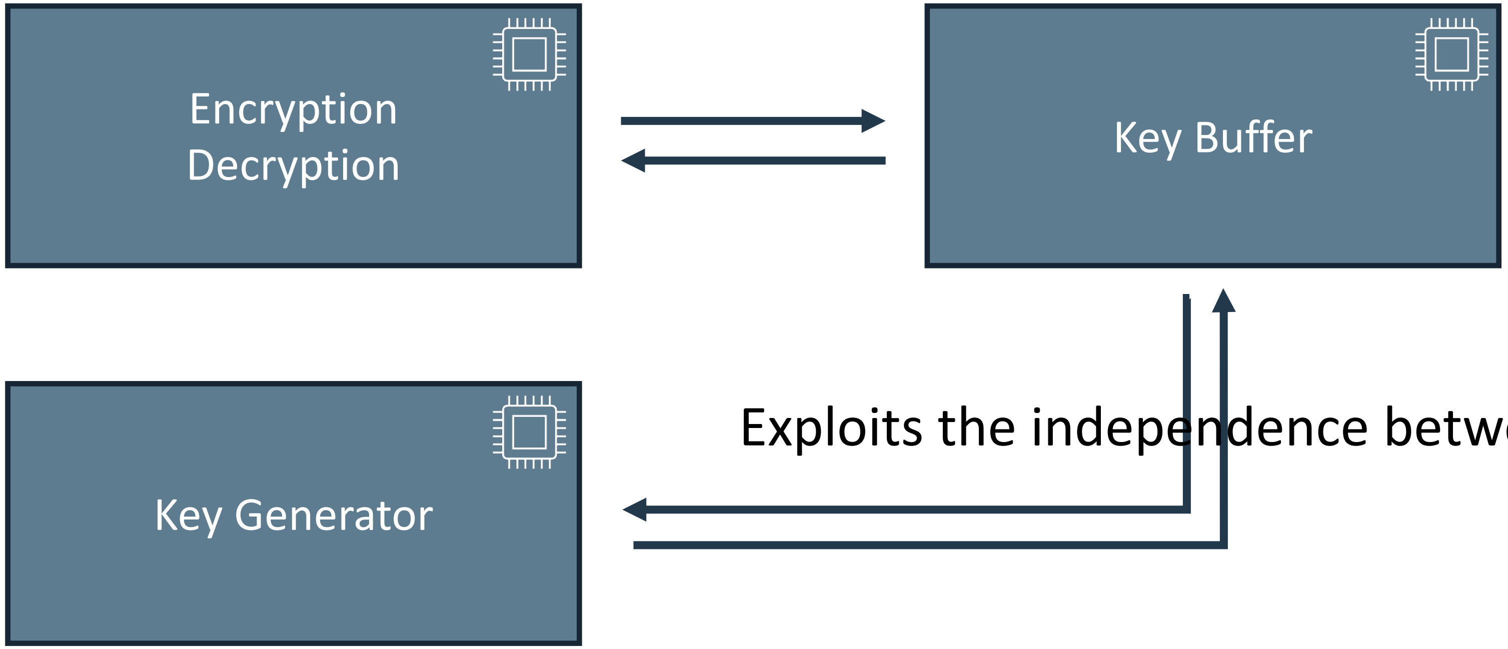


The design supports multiple Salsa20 versions

➢ Number of rounds specified during initialization
- **No** need to **resynthesize** the design

Salsa20 Cryptosystem – Design

- Same logical division of the Salsa20 cryptosystem



Circular buffer

- ➢ **Generates** multiple **keys** in **advance**
- ➢ **Reduces** key wait time
- ➢ **Increases** throughput

Exploits the independence between keys

Design Workflow

- Three different steps

1. **Cryptosystem initialization**
   - Initialize the cryptosystem (n, keylength, key, nonce)

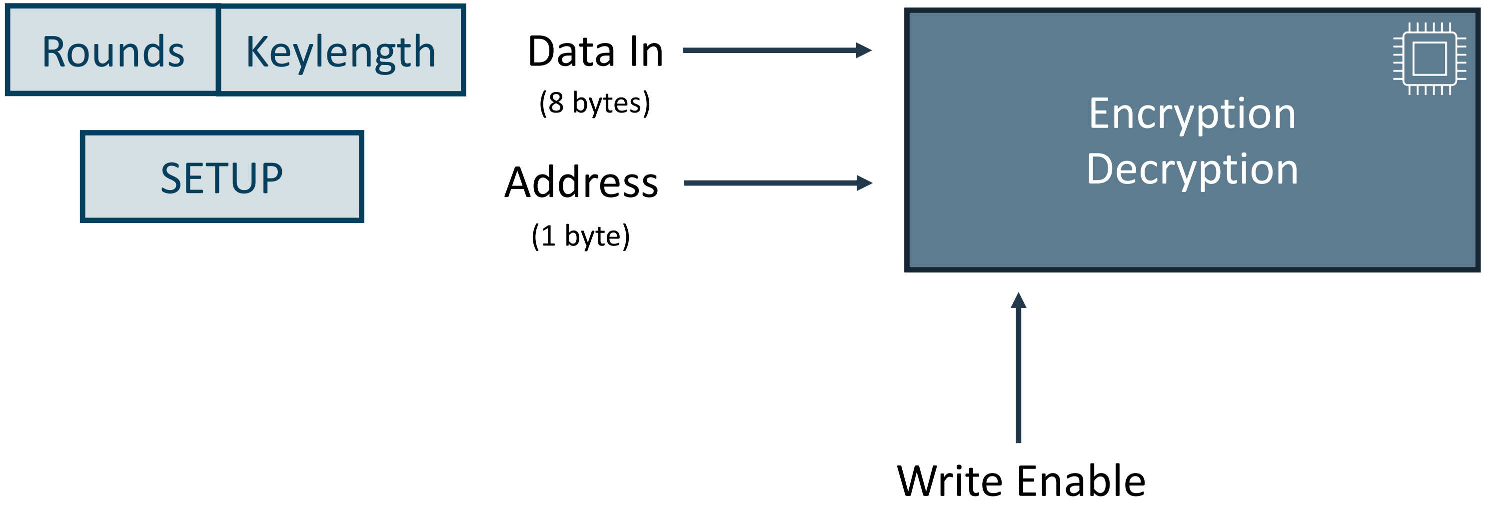
2. **Key generation**
   - Generate a new key

3. **Data encryption/decryption**
   - Encrypt/decrypt the input stream

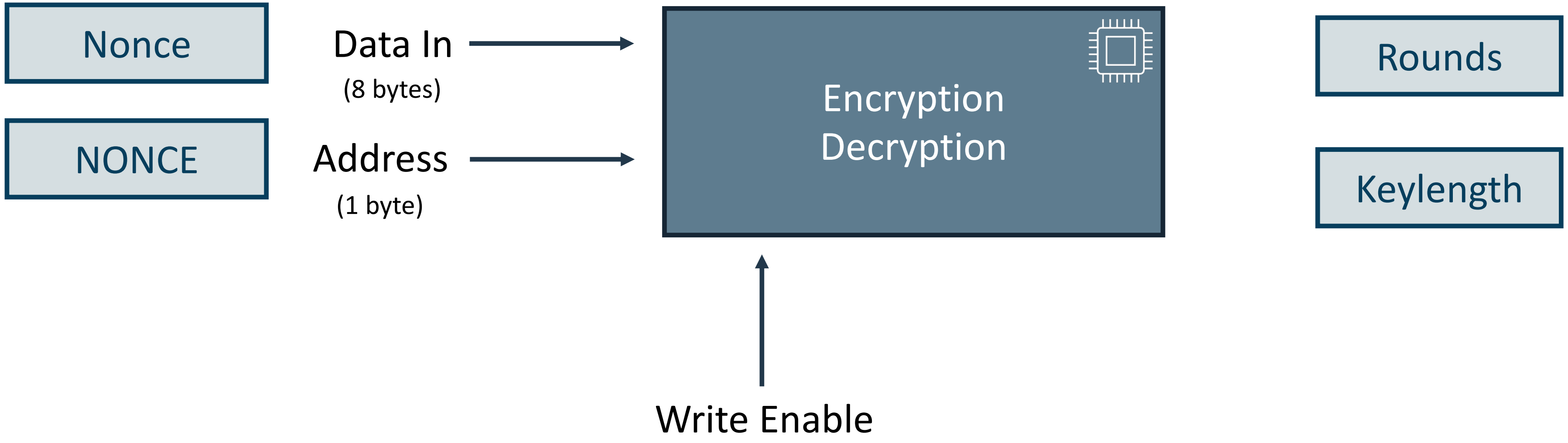
Cryptosystem Initialization

Workflow – Cryptosystem Initialization – Registers Initialization

- Constraint: low number of inputs/outputs

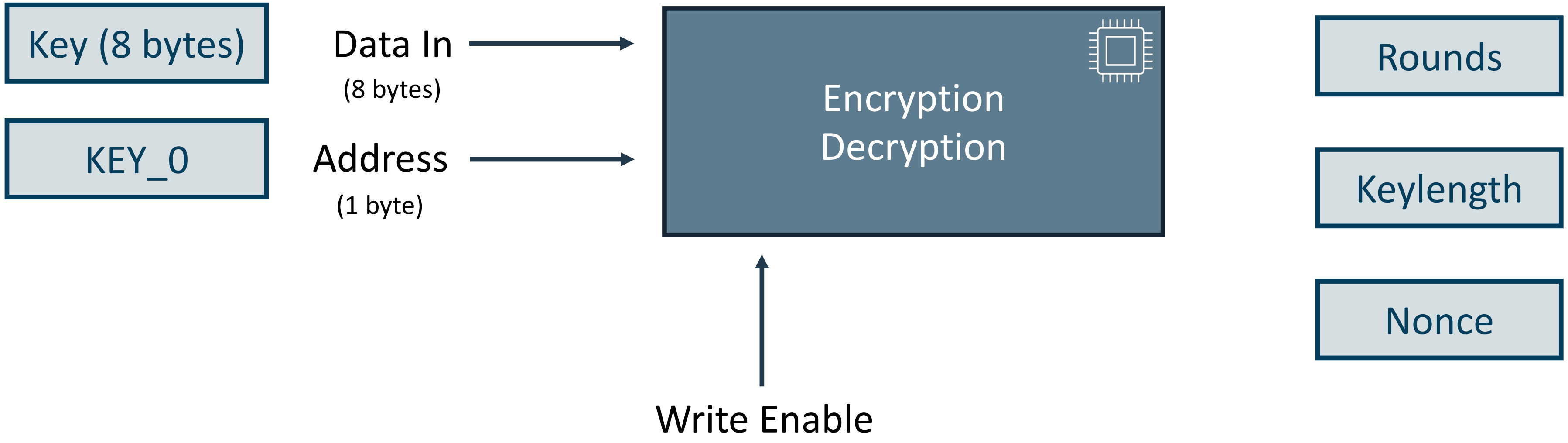


Workflow – Cryptosystem Initialization – Registers Initialization

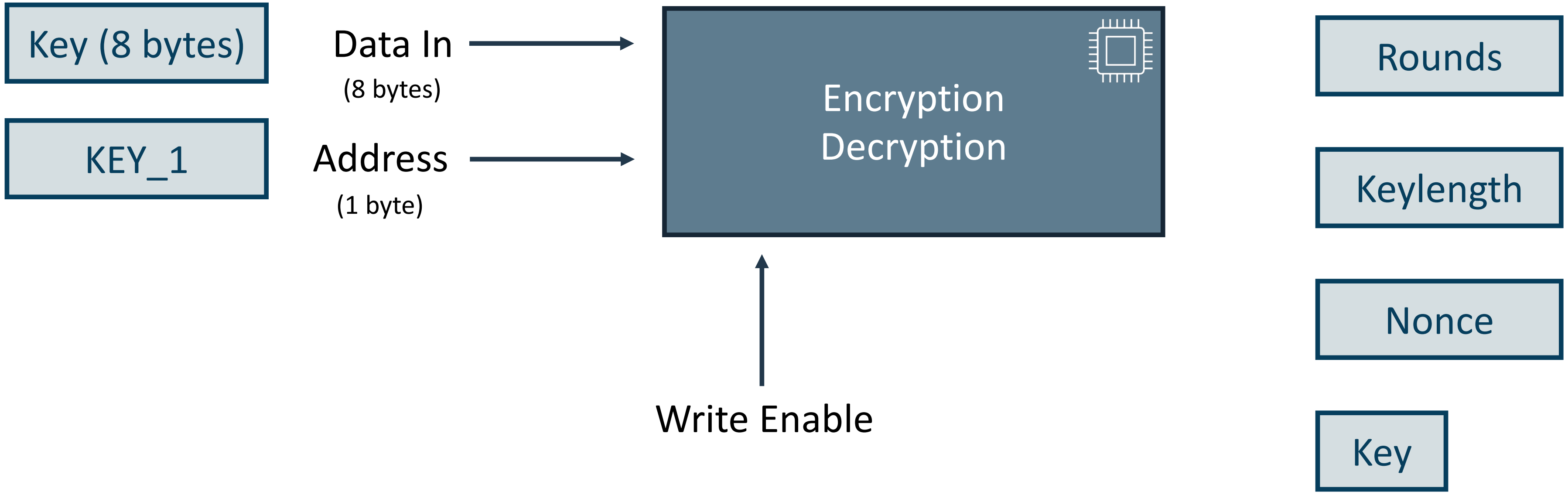


NETWORKED EMBEDDED SOFTWARE LAB

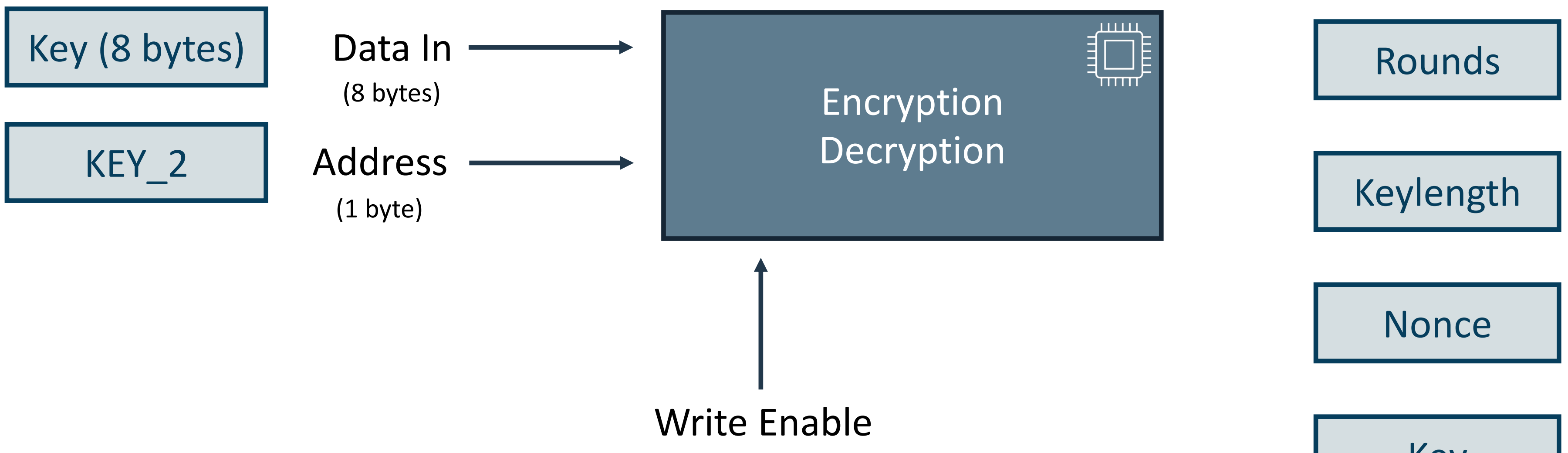
Workflow – Cryptosystem Initialization – Registers Initialization



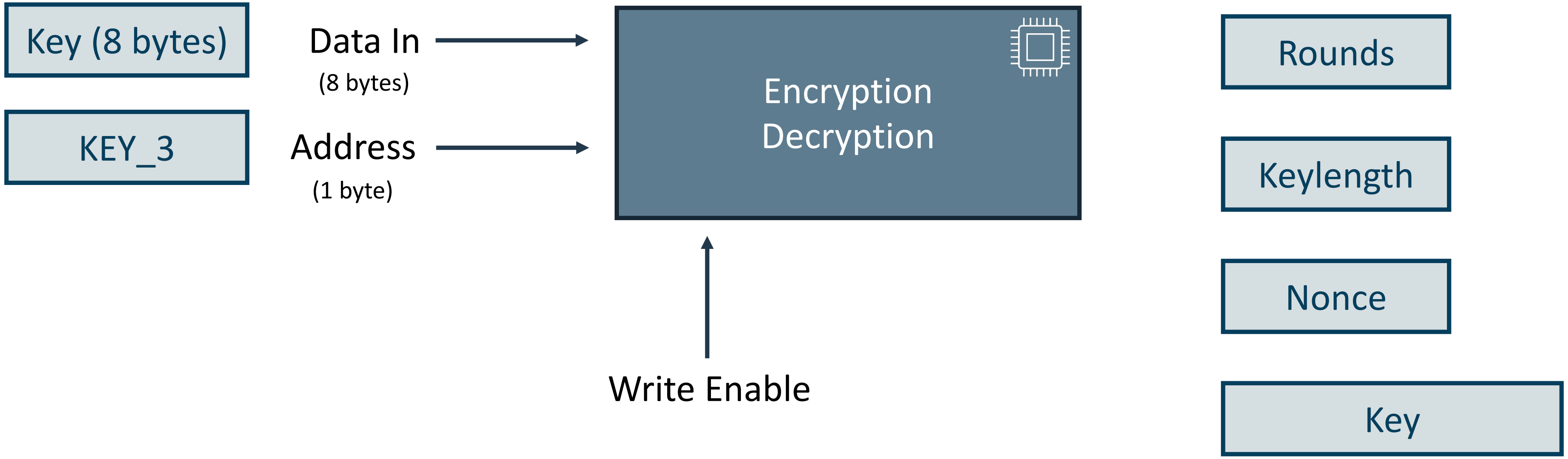
Workflow – Cryptosystem Initialization – Registers Initialization



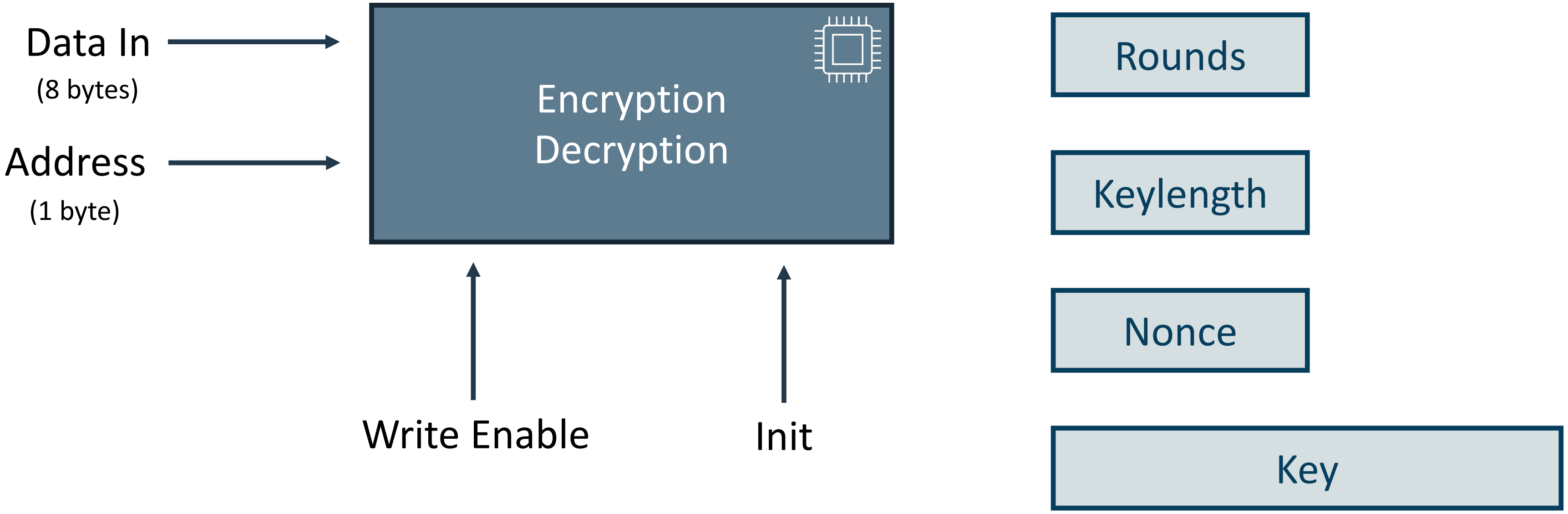
Workflow – Cryptosystem Initialization – Registers Initialization



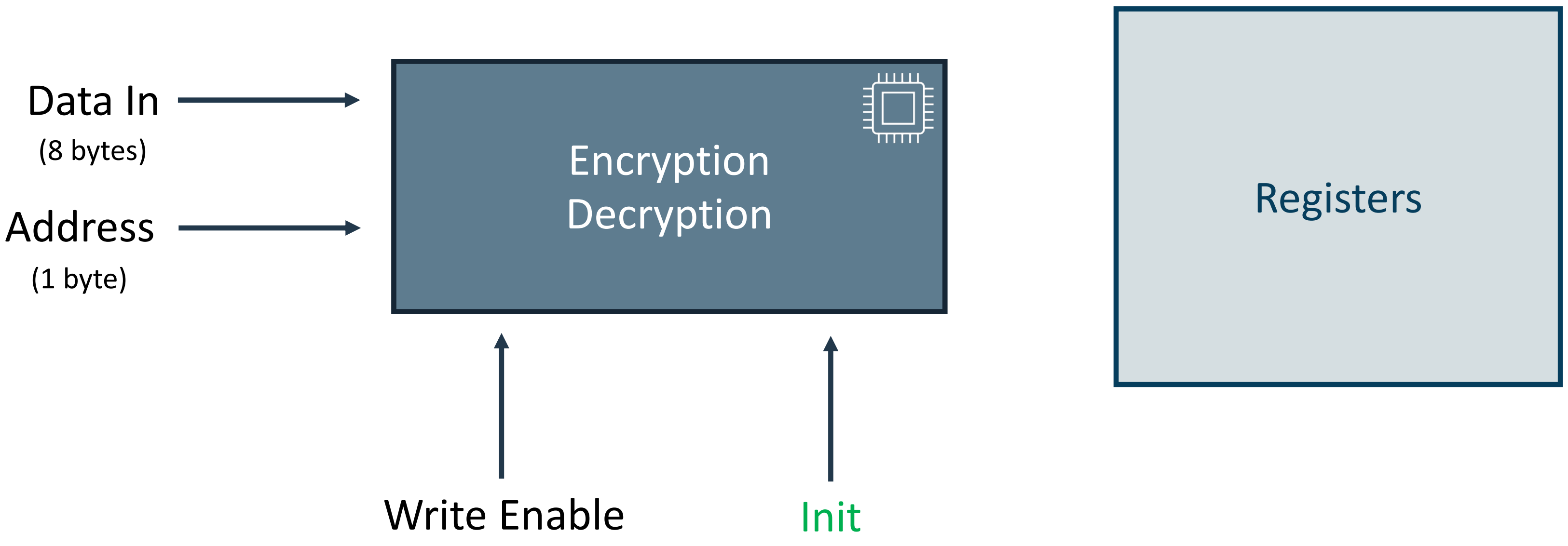
Workflow – Cryptosystem Initialization – Registers Initialization



Workflow – Cryptosystem Initialization

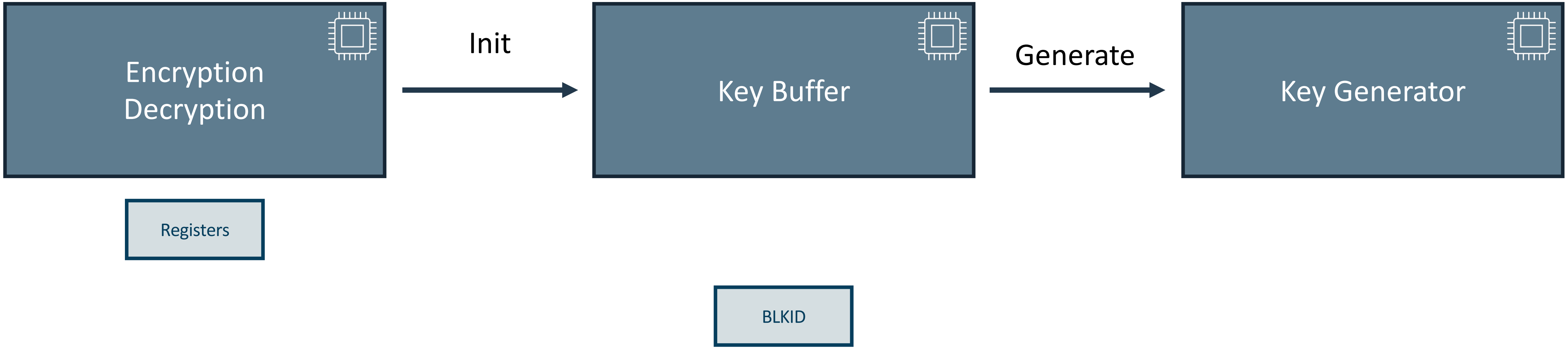


Workflow – Cryptosystem Initialization

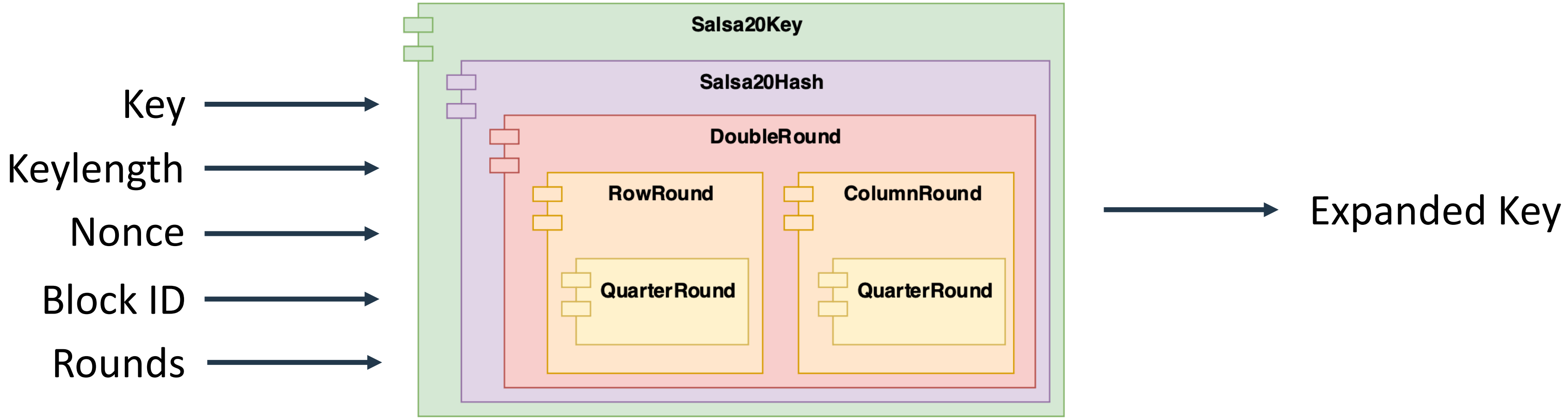


Key Generation

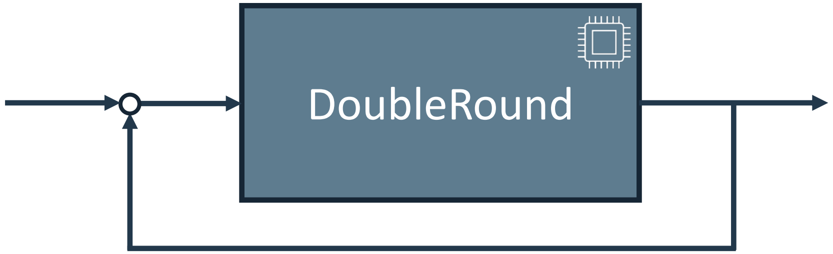
Workflow – Key Generation



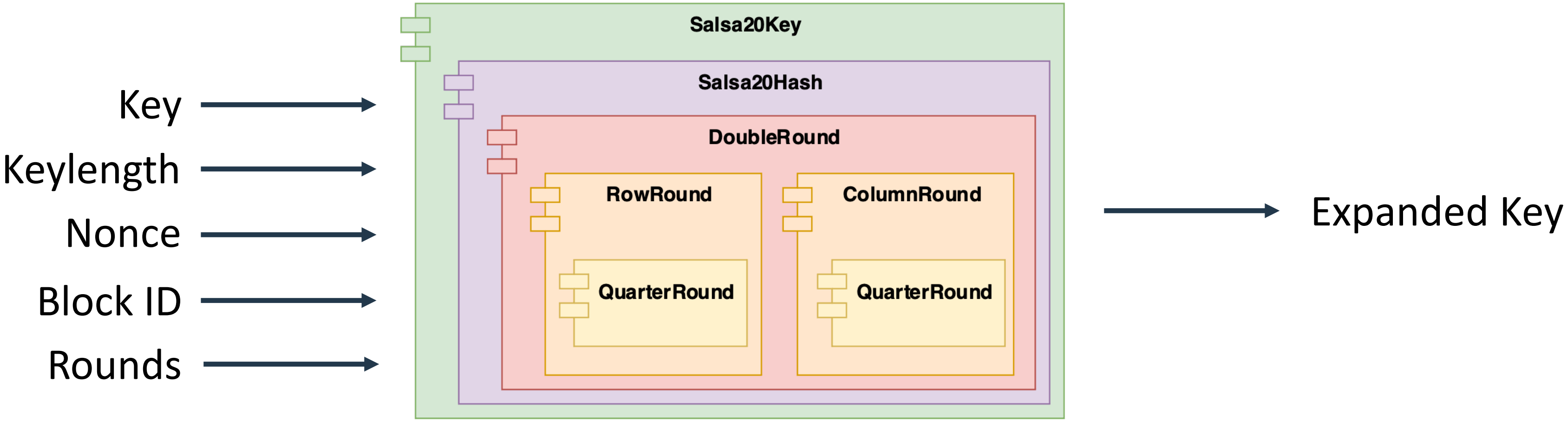
Design overview – Key Generation Modules



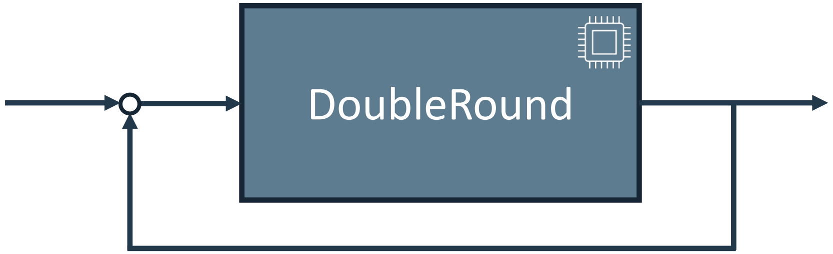
- Salsa20Key **initializes** the 64 bytes matrix
- Salsa20Hash uses a **feedback system** to transform the matrix *n* times



Design overview – Key Generation Modules

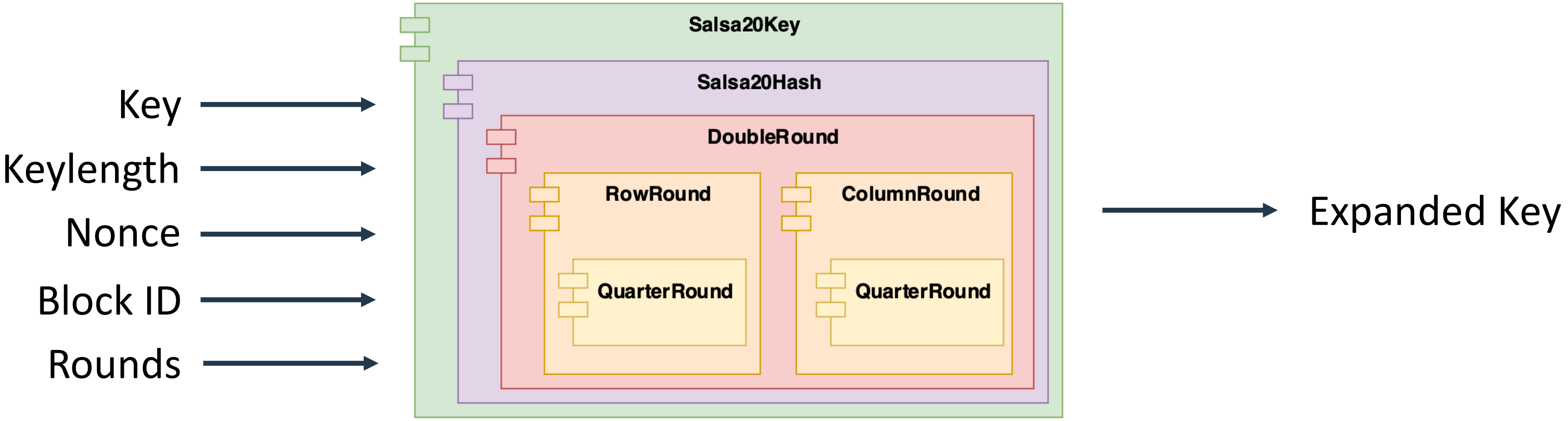


- Salsa20Key **initializes** the 64 bytes matrix
- Salsa20Hash uses a **feedback system** to transform the matrix *n* times



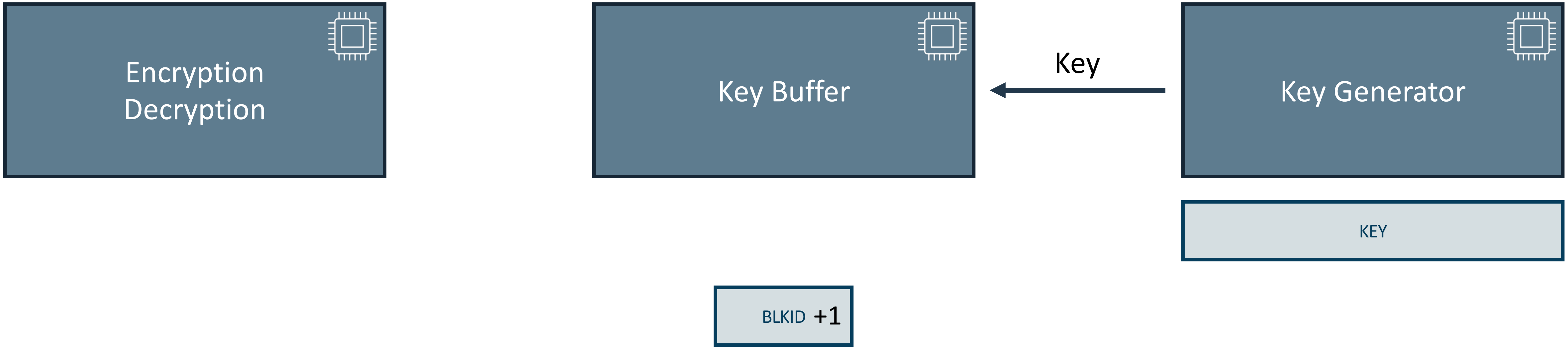
- **No** need to **re-synthesize** the design to change the number of rounds
- **Transformations** the matrix using only **one** DoubleRound **module**

Design overview – Key Generation Modules



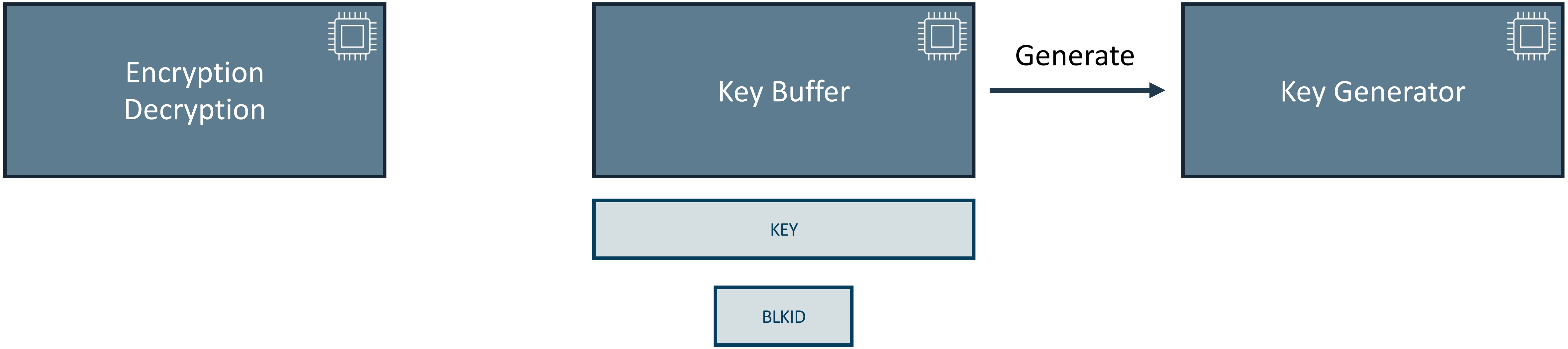
- The DoubleRound module **transforms** the **matrix**
  - Uses **combinatorial logic**
  - This is where **most** of the **LUTs** are **used**

Workflow – Key Generation



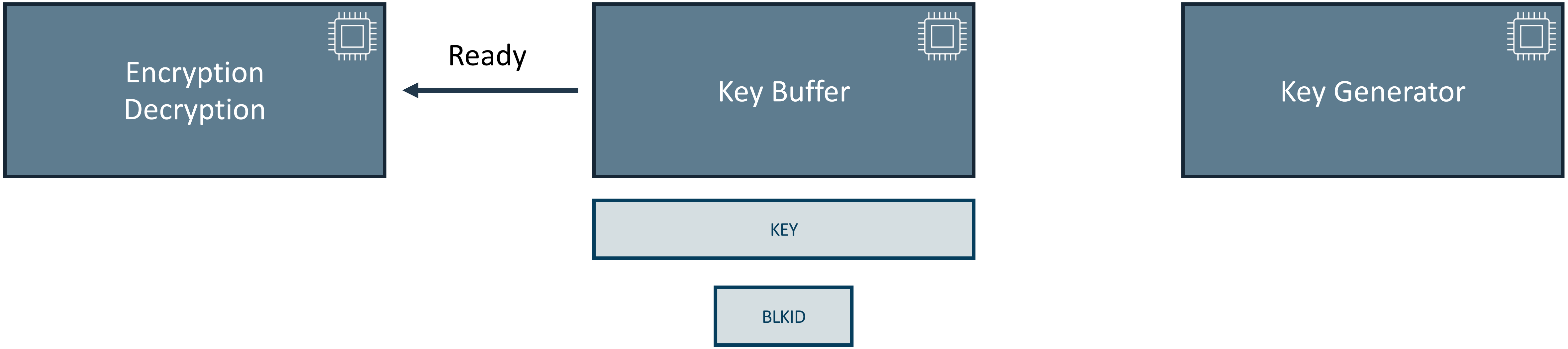
- The key is saved in a **circular buffer** of **two keys**
- The **block identifier** is **incremented**

Workflow – Key Generation

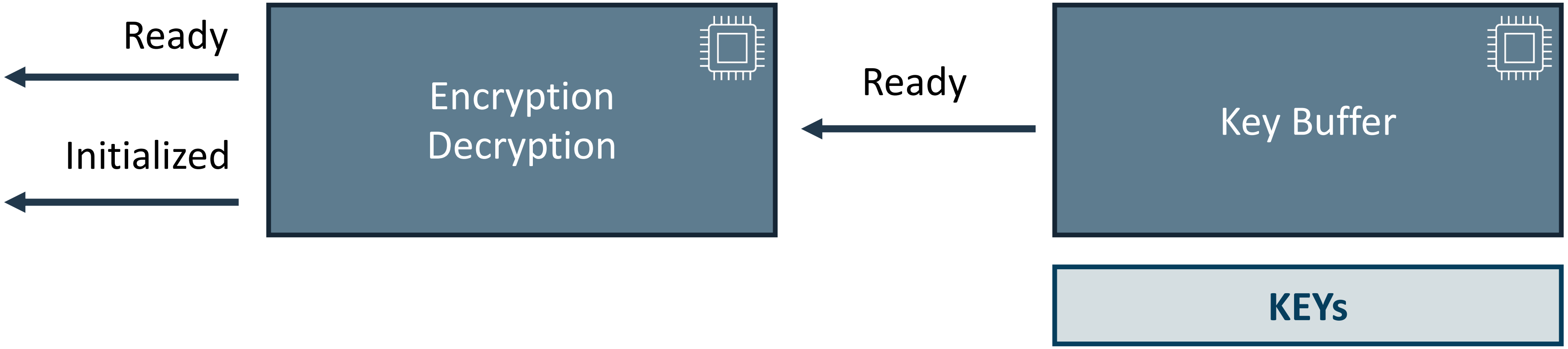


- A key **generation** requests is sent **whenever** the circular **buffer** is **not full**
- Every time a **key** is **used**, the **record** in the circular buffer is **invalidated**

Workflow – Key Generation

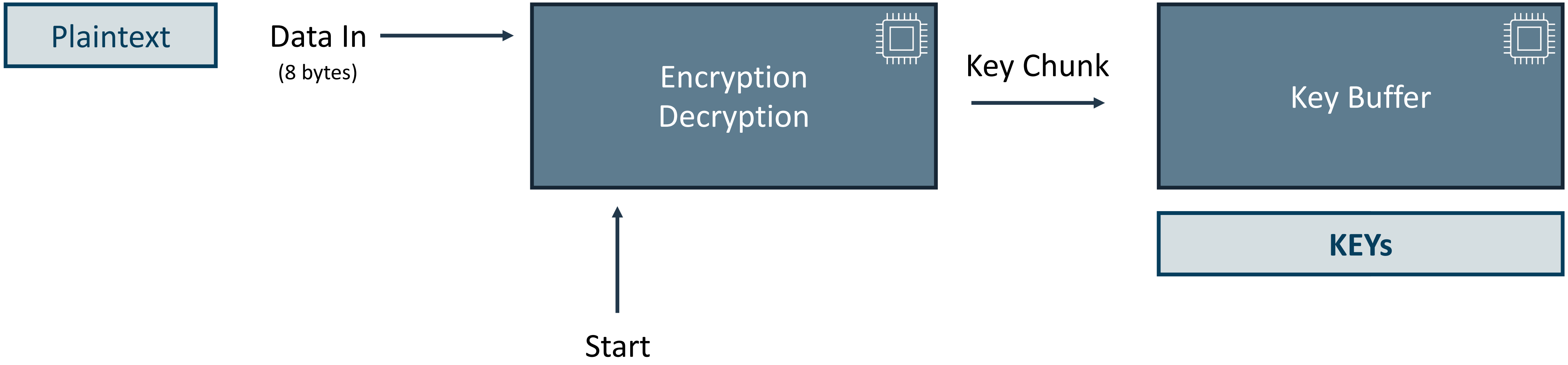


Workflow – Data Encryption / Decryption

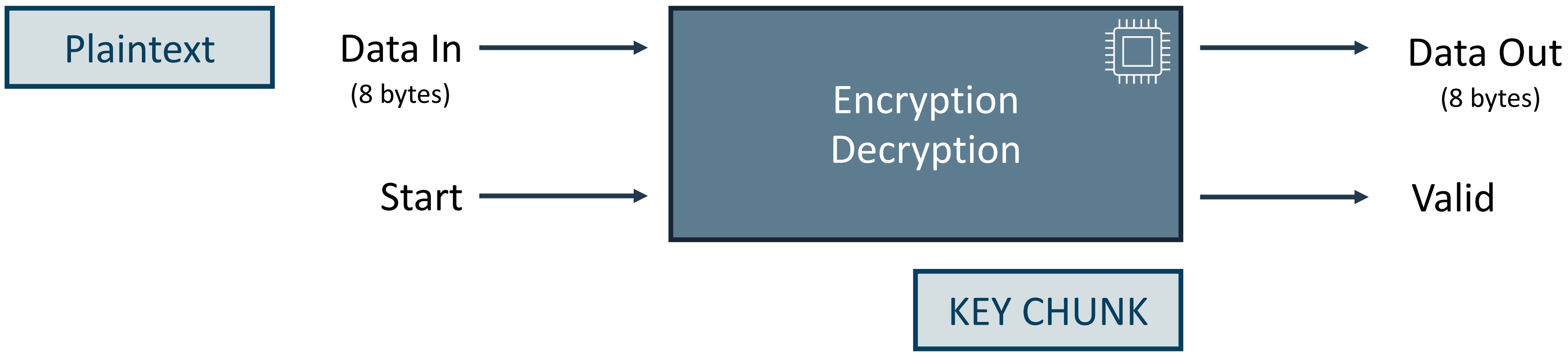


Encryption/Decryption

Workflow – Data Encryption / Decryption



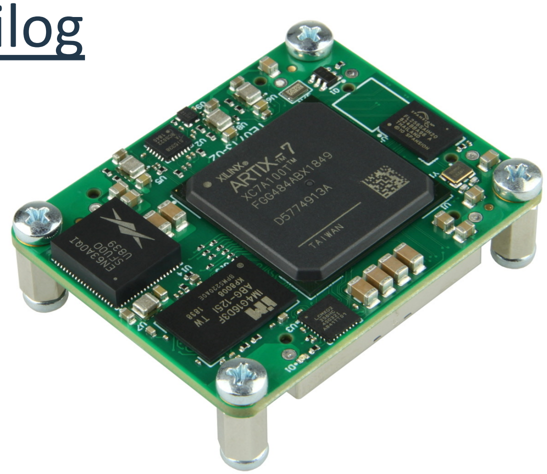
Workflow – Data Encryption / Decryption



- In the **same clock cycle** we can provide a **new input**
- The **same workflow** applies for data **decryption**

Design Evaluation

Design available at https://github.com/Maiux92/Salsa20SystemVerilog



✓ Technical report available in the repo

✓ **Synthesized** and **implemented**
- Target platform: **Xilinx Artix 7** model 7a100tcsg324-1

✓ **Validation**
- Data provided in Salsa20 **whitepaper** for Salsa20 **core functionalities**
- Python **implementation** of Salsa20 for **validating encryption** and **decryption**

Design Evaluation – Resources Utilization

	Synthesis	Implementation
LUTs	2586 (4.08%)	2582 (4.07%)
Slice Registers	2590 (2.04%)	2590 (2.04%)
F7 Muxes	128 (0.40%)	128 (0.40%)
F8 Muxes	64 (0.40%)	64 (0.40%)
IOB Ports	139 (66.19%)	139 (66.19%)

- The **DoubleRound** module uses **82%** of the **LUTs** (2136)
- Each expanded **key** requires **1000 registers**
  - **Key Buffer** module requires **85%** of the **registers** (2200)
- The **Key Buffer** module uses **all** the F7 and F8 **muxes**

Design Evaluation – Timing

System Timing

- The **DobuleRound** module is responsible for the **critical path**
  - **Only combinatorial logic**
- Minimum cock **period**: **23.33*ns***
- Maximum clock **frequency**: **43.33*MHz***

System Performance

- **Initialization**:
  - **4** clock cycles with 16 bytes key
  - **6** clock cycles with 32 bytes key

- **Key generator**: new key every *n + 2* clock cycles (n = number of rounds)
  - **Key Buffer – first key** wait cycles: *n + 2* clock cycles

  - **Key buffer – worst case**: key consumed after 8 clock cycles (64 bytes of input)
    - **Next keys wait** cycles: *(n + 2) - 8 clock cycles*
      - Salsa20: **4** clock cycles
      - Salsa12: **0** clock cycles
      - Salsa8: **0** clock cycles

Without key **buffer**: n + 2 clock cycles
- Salsa20: **12** clock cycles
- Salsa12: **8** clock cycles
- Salsa8: **6** clock cycles

- Encryption/decryption throughput: **8 bytes** of plaintext/ciphertext **per clock cycle**

Conclusion

NETWORKED EMBEDDED SOFTWARE LAB

Conclusion:

- **Design**, **synthesize**, and **implement** the Salsa20 cryptosystem
    - Support various Salsa20 versions, **without** re-synthesizing the design
    - Support 16 bytes and 32 bytes keys

    - A **feedback system** limits LUTs utilization
    - A **circular buffer** increases system throughput

- **General evaluation** of the implemented design

Questions?