LUX Hash Function

Ivica Nikolić, Alex Biryukov, Dmitry Khovratovich

University of Luxembourg

Outline

Design

Design

General Design of LUX

- Stream based (RadioGatun like) hash function
- Big internal state - 3 $\times$ message digest
- Message is processed by small (32-bit or 64-bit) chunks
- Round function uses Rijndael-like transformation

The internal state of LUX

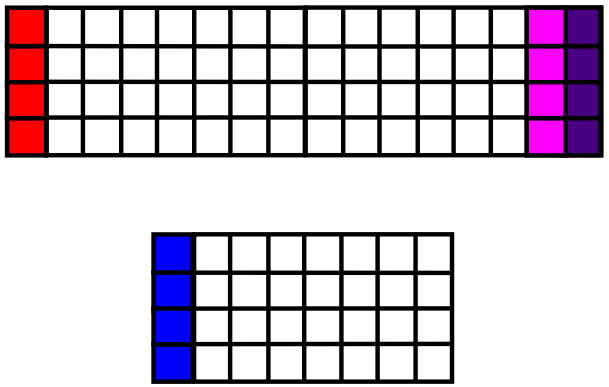
The state can be divided into two parts:

- Buffer - $m \times 16$ matrix of bytes (light transforms)
- Core - $m \times 8$ matrix of bytes (heavy transforms)

Output	m	Core	Buffer	Total
256	4	4×8	4×16	96
512	8	8×8	8×16	192

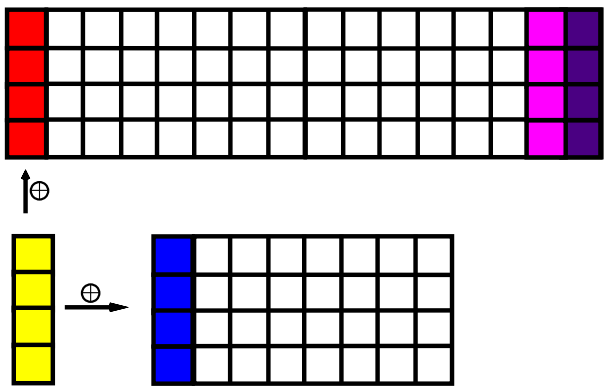
Feedforwards between the core and the buffer in each round

State update function (round transformation)



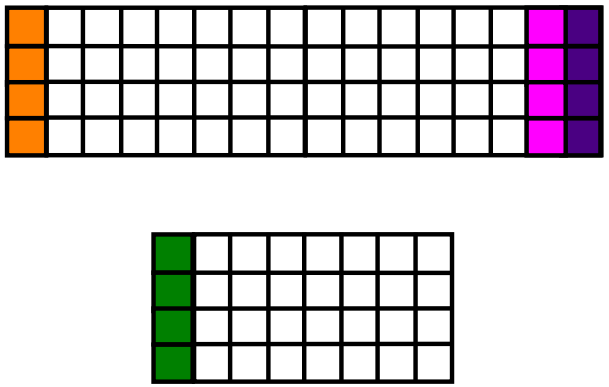
Message XOR to the core and the buffer
Update of the core and the buffer
XOR of the core to the buffer
Feedforward from the buffer to the core

State update function (round transformation)



Message XOR to the core and the buffer
Update of the core and the buffer
XOR of the core to the buffer
Feedforward from the buffer to the core

State update function (round transformation)



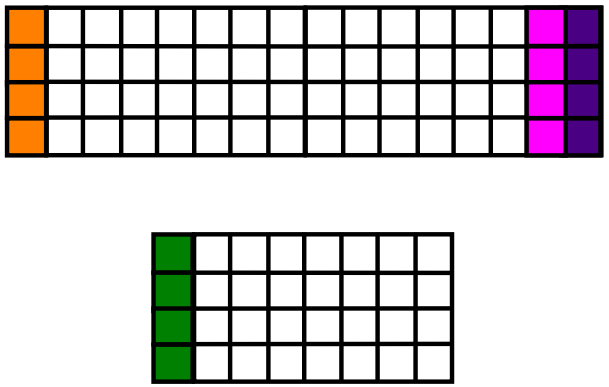
Message XOR to the core and the buffer

Update of the core and the buffer

XOR of the core to the buffer

Feedforward from the buffer to the core

State update function (round transformation)



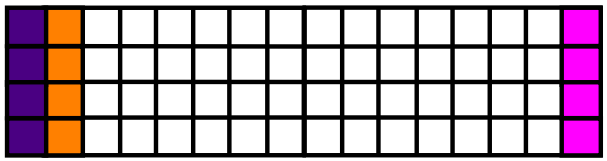
Message XOR to the core and the buffer

Update of the core and the buffer

XOR of the core to the buffer

Feedforward from the buffer to the core

State update function (round transformation)



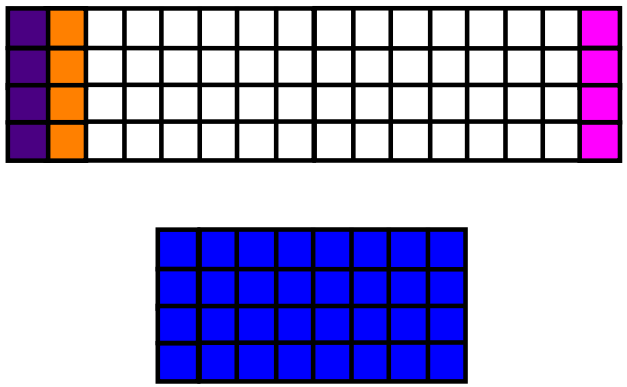
Message XOR to the core and the buffer

Update of the core and the buffer

XOR of the core to the buffer

Feedforward from the buffer to the core

State update function (round transformation)



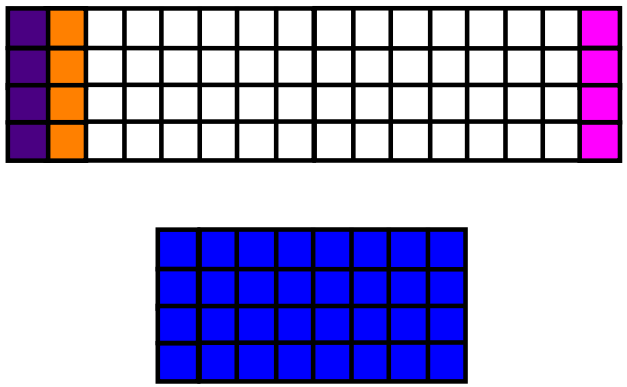
Message XOR to the core and the buffer

Update of the core and the buffer

XOR of the core to the buffer

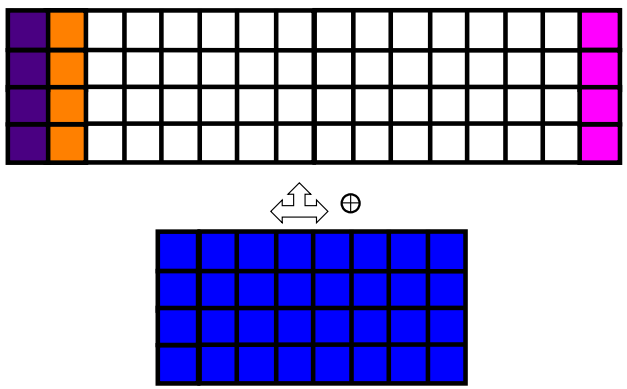
Feedforward from the buffer to the core

State update function (round transformation)



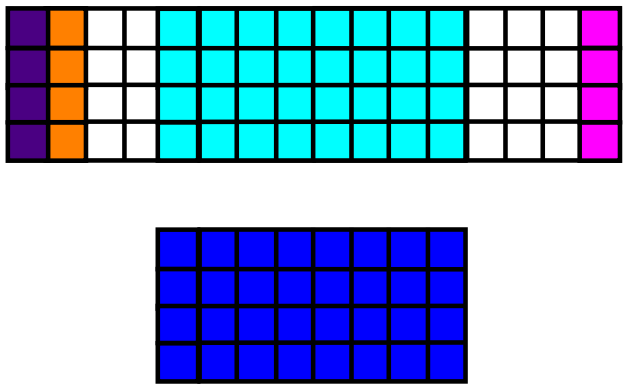
Message XOR to the core and the buffer
Update of the core and the buffer
XOR of the core to the buffer
Feedforward from the buffer to the core

State update function (round transformation)



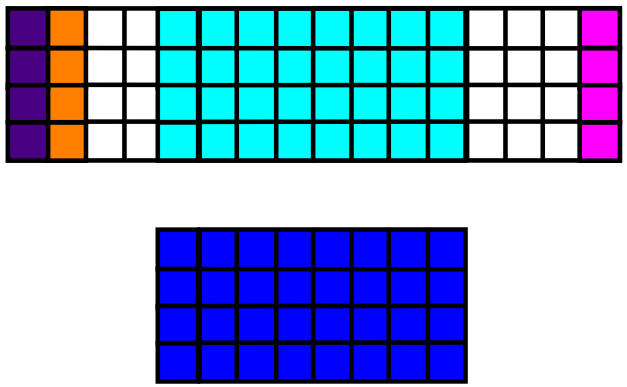
Message XOR to the core and the buffer
Update of the core and the buffer
XOR of the core to the buffer
Feedforward from the buffer to the core

State update function (round transformation)



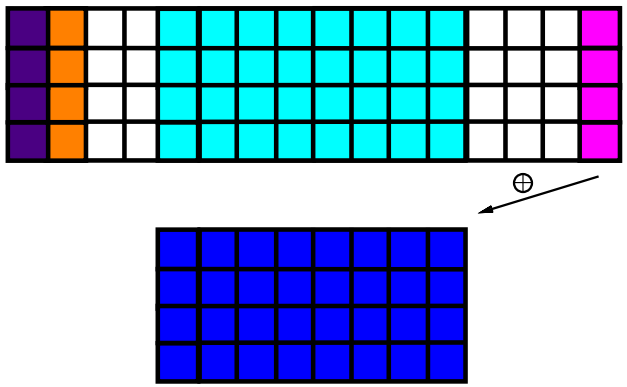
Message XOR to the core and the buffer
Update of the core and the buffer
XOR of the core to the buffer
Feedforward from the buffer to the core

State update function (round transformation)



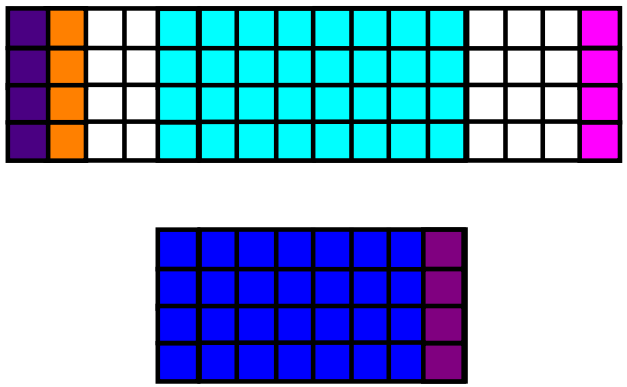
Message XOR to the core and the buffer
Update of the core and the buffer
XOR of the core to the buffer
Feedforward from the buffer to the core

State update function (round transformation)



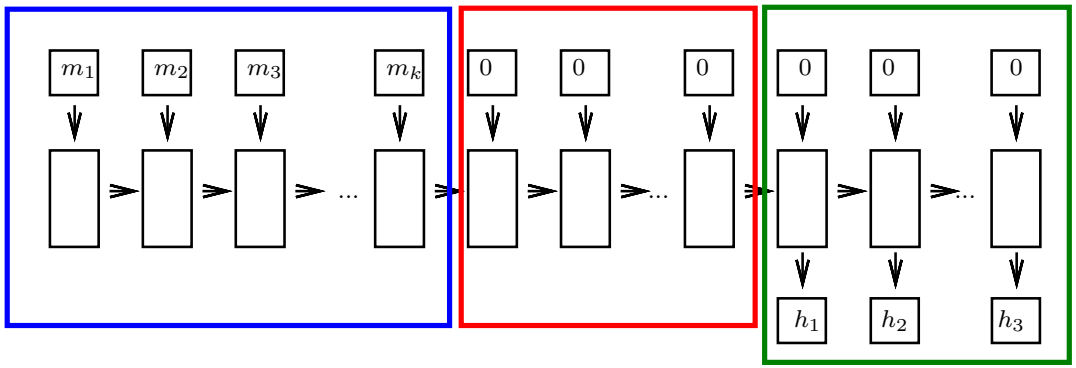
Message XOR to the core and the buffer
Update of the core and the buffer
XOR of the core to the buffer
Feedforward from the buffer to the core

State update function (round transformation)



Message XOR to the core and the buffer
Update of the core and the buffer
XOR of the core to the buffer
Feedforward from the buffer to the core

Hashing



Three phases of hashing:

- Input phase - absorb the whole message
- Blank rounds phase - increase diffusion of the last message blocks
- Output phase - produce the hash value from the state

Security

Security

Multicollisions, length-extension, herding, 2-nd

- Multicollision, length-extension and herding attacks require internal collisions
- 2-nd preimage attack (Dean, Kelsey-Schneier) requires finding at least one preimage for some intermediate state value

The big internal state of LUX-n has $3n$ bits $\Rightarrow$ internal collisions/preimages are expensive

Collisions

Truncated differentials (see Peyrin's attack on Grindahl)

- Build a trail of truncated differentials
- Complexity of the attack depends on the number of active S-Boxes
- Fix some values of the S-Boxes with the message input

The best truncated differential trail found for LUX-256 has 88 active S-Boxes where 38 can be fixed $\Rightarrow$ complexity 2^{300}

Preimages

- Whole execution of LUX is invertible $\Rightarrow$ try MITM attack for preimages
  Big internal state ($3n$) stops this attack
- Try to fix some intermediate values in the buffer.
  Due to the xor of the core to the buffer, only n bits can be fixed $\Rightarrow$ complexity of MITM on $2n$-bit state is 2^n

Recent cryptanalysis

- Free-start collisions/preimages and distinguishers (Wu et al.)
  Free start attacks on invertible functions are trivial. Outputting the whole state at once stops the distinguisher based on the properties of the output transform
- Length extension slide attack (Peyrin)
  Needs salt size to be equal to 31 (mod 32) bits. Salt size is fixed to 128-bits in LUX.

Implementation

Implementation

Implementation results

Primitive comparation to AES (counting the number of XORs and table look-ups) gives a speed-up of 1.2 in favor of LUX

	224/256	384/512
32-bit (C)	16.7	28.2
64-bit (asm)	10.2	9.5

- Speed on 32-bit can be improved with an assembler implementation
- The new Intel instruction set can improve the speed of LUX-256

Advantages

Advantages

Pros

Rijndael-based - well analyzed transformation

- Cryptanalysis can be focused only on the construction
- Implementation tricks of Rijndael can be used in LUX

Speed - one of the fastest on both 32 and 64-bit platforms

- Stable high speed on various processors (AMD, Intel)
- Overperforms all AES based functions

Check **LUX**embourg on **cryptolux.org/LUX**