HARDWARE AND SOFTWARE IMPLEMENTATION OF LIGHTWEIGHT CRYPTOGRAPHY ALGORITHMS

Zarif Khudoykulov
PhD, Tashkent University of Information
Technologies named after Muhammad al-Khwarizmi,
zarif.khudoykulov@tuit.uz

Abstract
Lightweight Cryptography (LWC) algorithms are critical for securing resource-constrained devices in the Internet of Things (IoT), embedded systems, and cyber-physical applications. This paper provides a comprehensive analysis of hardware and software implementations of LWC algorithms, focusing on the sequence of implementation, target platforms, programming languages, tools, and environments. We present key performance metrics, including area, power, throughput, and memory usage, and conduct comparative analyses of tools and performance data for NIST LWC ciphers, such as Ascon, GIFT-COFB, and TinyJAMBU. Insights from the NIST LWC standardization process highlight the trade-offs between security and efficiency. Our findings offer guidance for practitioners and researchers deploying LWC in constrained environments.

Keywords: Lightweight Cryptography, Hardware Implementation, Software Implementation, NIST LWC, IoT Security, Performance Metrics.

1. Introduction

The rapid growth of the Internet of Things (IoT) and embedded systems has driven demand for cryptographic solutions that operate efficiently on resource-constrained devices, such as RFID tags, sensors, and microcontrollers. Lightweight Cryptography (LWC) algorithms address these needs by providing security with minimal computational, memory, and power requirements, unlike traditional algorithms like AES, which are resource-intensive. The National Institute of Standards and Technology (NIST) LWC standardization process (2018–2023) evaluated 57 candidates, selecting Ascon as the standard for its balance of security, performance, and versatility.

LWC implementations are realized in hardware (e.g., FPGAs, ASICs) for high efficiency and in software (e.g., microcontrollers) for flexibility. Hardware implementations prioritize small chip area and low power, while software implementations focus on low memory usage and portability. Key performance metrics guide the evaluation of these implementations, ensuring suitability for constrained environments.

1.1 Hardware and Software Implementations for LWC Algorithms

Hardware implementations leverage dedicated circuits to achieve high throughput and low power, critical for battery-powered IoT devices. Software implementations, executed on general-purpose processors or microcontrollers, offer ease of deployment but face challenges in meeting real-time constraints. Both approaches require careful optimization to meet LWC's stringent resource constraints.

1.2 Key Performance Metrics for Hardware and Software Implementation

Hardware Metrics:

- **Area:** Measured in gate equivalents (GE) for ASICs or Look-Up Tables (LUTs) for FPGAs, targeting <2000 GE for ultra-lightweight applications.
- **Power Consumption:** Measured in microwatts, critical for energy harvesting devices.
- **Throughput:** Measured in bits per second (bps), reflecting encryption speed.
- **Latency:** Time to process one block, crucial for real-time applications.

Software Metrics:

- **Memory Usage:** RAM and ROM requirements, often <512 bytes for constrained devices.
- **Throughput:** Measured in bytes per second (Bps), dependent on processor architecture.
- **Execution Time:** Clock cycles per operation, affecting responsiveness.
- **Code Size:** Flash memory footprint, critical for microcontrollers.

This paper explores the implementation sequences, platforms, tools, and performance of NIST LWC ciphers, providing comparative analyses to guide practitioners.

2. Hardware Implementation

2.1 Hardware Implementation Sequence

The hardware implementation of LWC algorithms follows a structured sequence [4]:

- **Algorithm Specification:** Define the algorithm's structure (e.g., block cipher, sponge construction) and parameters (key size, nonce, tag).
- **RTL Design:** Develop Register-Transfer Level (RTL) code in VHDL or Verilog, specifying data paths and control logic.
- **Simulation:** Use tools like ModelSim to verify functional correctness against test vectors.
- **Synthesis:** Map the design to target hardware (FPGA or ASIC) using synthesis tools like Xilinx Vivado or Synopsys Design Compiler.
- **Place-and-Route:** Optimize physical layout for area, power, and timing.
- **Verification:** Perform post-synthesis simulation and timing analysis.

- **Implementation:** Deploy the design on the target platform, testing for performance and power.

2.2 Target Platforms [5]

- **FPGAs:** Xilinx Artix-7 and Spartan-7 are widely used for prototyping due to their reconfigurability and low cost.
- **ASICs:** Custom chips for mass production, offering minimal area and power (e.g., 180nm or 65nm CMOS processes).
- **Microcontrollers:** Low-power MCUs like MSP430 for hybrid hardware-software designs.

2.3 Programming Languages [5]

- **VHDL/Verilog:** Standard hardware description languages for RTL design, offering precise control over logic.
- **SystemVerilog:** Used for advanced verification and constrained-random testing.
- **High-Level Synthesis (HLS):** C/C++-based HLS (e.g., Vivado HLS) accelerates design by abstracting low-level details.

2.4 Tools and Environments

- **Xilinx Vivado:** Supports FPGA synthesis, place-and-route, and power analysis. Used in NIST LWC for Artix-7 benchmarking.
- **Synopsys Design Compiler:** Optimizes ASIC designs for area and power, used for candidates like GIFT-COFB.
- **Cadence Genus:** Alternative ASIC synthesis tool, focusing on low-power designs.
- **ModelSim:** Simulates VHDL/Verilog designs, ensuring correctness.
- **GMU Hardware API:** Standardizes LWC hardware evaluation, providing AEAD interfaces.

2.5 Comparative Analysis Data of Performance of NIST LWC Ciphers

Table 1 and 2 summarizes the hardware performance of NIST LWC finalists on Xilinx Artix-7 FPGA and 22nm ASIC (and TSMC 65nm for one case) [3].

The ASIC implementations, fabricated using GF 22nm technology except for Romulus (TSMC 65nm), showcase diverse performance characteristics across the algorithms Ascon, GIFT-COFB, Grain-128AEAD, Elephant, ISAP, and Romulus. Throughput varies significantly, with Ascon achieving the highest at 531.9 Mbps, followed by ISAP at 195.03 Mbps and Elephant at 70.92 Mbps, while Grain-128AEAD offers the lowest at 0.01773 Mbps, indicating its suitability for ultra-low-bandwidth applications. Area efficiency, measured in kGE (kilo-gate equivalents), ranges from 4.3 kGE (Grain-128AEAD) to 17.3 kGE (Elephant), with Romulus standing out at 10,008 kGE due to its older 65nm technology, suggesting a less optimized design. Power consumption data is unavailable for most algorithms except Romulus (13.86 mWatt), which operates at 1 GHz, reflecting a moderate power profile for its throughput of

9.35 Mbps. Frequency data is absent for all but Romulus, highlighting a gap in performance optimization details. Ascon and ISAP demonstrate a balance of high throughput and low area, making them ideal for resource-constrained, high-speed environments, while Grain-128AEAD excels in minimal area usage, suited for IoT devices. The lack of frequency and power data for most algorithms limits a comprehensive power-efficiency analysis, but the trend suggests a trade-off between throughput and area, with Romulus's larger area reflecting its older technology.

Table 1

ASIC Implementation of NIST LWC finalist's ciphers

Algorithm	ASIC Implementation				
	Technology	Frequency (GHz)	Throughput (Mbps)	Area (kGE)	Power (mWatt)
Ascon	GF 22nm	N/A	531.9	11	-
GIFT-COFB	GF 22nm	N/A	159.57	8.1	-
Grain-128AEAD	GF 22nm	N/A	0.01773	4.3	-
Elephant	GF 22nm	N/A	70.92	17.3	-
ISAP	GF 22nm	N/A	195.03	15.4	-
Romulus	TSMC 65nm	1	9.35	10008	13.86

The FPGA implementations, primarily on Xilinx Artix 7 except for SPARKLE (Virtex 7), reveal a rich performance landscape across Ascon, GIFT-COFB, TinyJAMBU, Grain-128AEAD, Elephant, ISAP, Photon-Beetle, Romulus, and Xoodyak. GIFT-COFB leads in throughput at 2897.5 Mbps (249 MHz), followed by Xoodyak at 2960.4 Mbps (234 MHz) and ISAP at 2560 Mbps (280 MHz), showcasing their suitability for high-speed applications, while Grain-128AEAD offers the lowest at 3.96 Mbps (247.5 MHz), ideal for low-bandwidth scenarios. Area, measured in LUTs (look-up tables), ranges from 576 (TinyJAMBU) to 2065 (Photon-Beetle), with Romulus at 953 LUTs indicating efficient resource use. Power consumption, normalized at 75 MHz where available, varies from 76 mWatt (Elephant) to 242 mWatt (Photon-Beetle), with GIFT-COFB at 208 mWatt reflecting higher energy demands for its throughput. Frequency ranges from 178 MHz (Photon-Beetle) to 280 MHz (ISAP), with SPARKLE at 186.70 MHz on Virtex 7. GIFT-COFB and Xoodyak excel in throughput-per-area ratios, making them suitable for performance-critical systems, while TinyJAMBU and Elephant offer low area and power, ideal for resource-constrained environments. The variability in power data (e.g., missing for Grain-128AEAD) suggests further optimization

potential, with a clear trade-off between throughput and power/area efficiency across the algorithms.

Table 2

FPGA Implementation of NIST LWC finalist's ciphers

Algorithm	FPGA Implementation				
	Technology	Frequency (MHz)	Throughput (Mbps)	Area (LUTs)	Power (mWatt)
Ascon	Artix 7	233	1491.2	1913	100 (75 MHz)
GIFT-COFB	Artix 7	249	2897.5	1641	208 (75 MHz)
TinyJAMBU	Artix 7	240	1396.4	576	106 (75 MHz)
Grain-128AEAD	Artix 7	247.5	3.96	1730	-
SPARKLE	Virtex 7	186.70	232.09	1530	-
Elephant	Artix 7	229	282.9	1291	76 (75 MHz)
ISAP	Artix 7	280	2560	1674	94
Photon-Beetle	Artix 7	178	747	2065	242 (75 MHz)
Romulus	Artix 7	229	637.2	953	99 (75 MHz)
Xoodyak	Artix 7	234	2960.4	1355	130 (75 MHz)

3. Software Implementation

3.1 Software Implementation Sequence

The software implementation sequence includes:

- **Algorithm Specification:** Analyze the algorithm's operations (e.g., permutations, bitwise operations).
- **Code Development:** Write optimized code in C or assembly, targeting specific platforms.
- **Simulation:** Test code against NIST test vectors using IDEs or simulators.
- **Optimization:** Apply techniques like loop unrolling or table lookups to reduce cycles.
- **Profiling:** Measure memory usage, throughput, and execution time.
- **Deployment:** Integrate into the target application, ensuring compatibility.
- **Verification:** Validate functionality and performance in real-world conditions.

3.2 Target Platforms

- **Microcontrollers:** ARM Cortex-M0/M3 (32-bit, low-power), Renesas RL78 (16-bit), MSP430 (16-bit).
- **Embedded Processors:** RISC-V for open-source designs, used in IoT nodes.
- **General-Purpose CPUs:** For benchmarking, though less relevant for constrained devices.

3.3 Programming Languages

- **C:** Primary language for portability and optimization, used in NIST reference implementations.
- **Assembly:** For cycle-accurate optimizations on specific MCUs (e.g., Cortex-M0).
- **Python:** For prototyping and test vector generation, not deployment.

3.4 Tools and Environments

- **GCC/Clang:** Compilers for generating optimized code for ARM and RISC-V.
- **Keil uVision:** IDE for embedded development, supporting debugging and profiling.
- **PlatformIO:** Cross-platform IDE for IoT, used in NIST submissions.
- **IAR Embedded Workbench:** Optimizes code size and performance for RL78 and MSP430.
- **Crypto++:** Reference library for validation, though too heavy for deployment.

3.5 Comparative Analysis Data of Performance of NIST LWC Ciphers

Renner et al. [2] proposed a benchmarking framework to evaluate AEAD algorithm performance on microcontrollers, using execution time (average microseconds per test vector), compiled binary size, and RAM usage as key metrics. The study encompasses 295 implementations across five microcontroller platforms.

Selected results are summarized in Table 3. On the Arduino Uno, SPARKLE, GIFT-COFB, and Xoodyak demonstrated the fastest execution times, with ASCON, Tiny-JAMBU, and Romulus also showing competitive performance. On the ESP32, ASCON, Xoodyak, and Tiny-JAMBU ranked highest in speed. Regarding code size, ASCON and PHOTON-Beetle had the smallest binaries on the Arduino Uno, while ASCON, ISAP, Tiny-JAMBU, SPARKLE, and Xoodyak achieved the lowest code sizes on the Maixduino [1].

Table 3

Speed and code size measurements

Algorithm	Speed, microseconds		ROM size, b	
	Arduino Uno	ESP32	Arduino Uno	Maixduino
Ascon	2472.06	22.86	5388	1.24×10^5
GIFT-COFB	2250.02	55.61	5814	1.31×10^5
TinyJAMBU	2386.1	43.59	9956	1.24×10^5
Grain-128AEAD	5113.35	119.72	9394	1.29×10^5
SPARKLE	1999.74	62.2	6664	1.25×10^5
Elephant	12730.3	3986.3	7930	1.32×10^5
ISAP	22486	608.46	5486	1.25×10^5
Photon-Beetle	4821.26	185.76	4442	1.3×10^5
Romulus	2870.17	114.16	15166	1.29×10^5
Xoodyak	2371	39.18	5598	1.25×10^5

4. Summary

This paper provided a detailed examination of hardware and software implementations of LWC algorithms, focusing on the NIST LWC ciphers. Hardware implementations, leveraging FPGAs and ASICs, achieve small area (<2000 GE) and low power (<30 μW), with tools like Vivado and Design Compiler enabling optimization. Software implementations, targeting microcontrollers, prioritize low memory (60–120 bytes RAM) and portability, using GCC and Keil uVision. Comparative analyses reveal Ascon-128's balanced performance, TinyJAMBU's minimal footprint, and GIFT-COFB's efficiency. The NIST LWC process underscores the importance of standardized APIs and community benchmarking.

5. References

1. Turan M. S. et al. Status report on the final round of the NIST lightweight cryptography standardization process. – US Department of Commerce, National Institute of Standards and Technology, 2023.
2. Renner S, Pozzobon E, Mottok J LWC Benchmark, GitHub repository. Available at https://lab.las3.de/gitlab/lwc/compare.
3. Konstantopoulou E., Athanasiou G., Sklavos N. Review and Analysis of FPGA and ASIC Implementations of NIST Lightweight Cryptography Finalists //ACM Computing Surveys. – 2025.
4. Ovilla-Martínez B. et al. FPGA implementation of some second round NIST lightweight cryptography candidates //Electronics. – 2020. – T. 9. – №. 11. – C. 1940.
5. Rezvani B. et al. Hardware implementations of NIST lightweight cryptographic candidates: A first look //Cryptology ePrint Archive. – 2019.