Journal, Advances in Mathematical & Computational Sciences
Vol. 14 No. 1, March 2026 Series
www.isteams.net/mathematics-computationaljournal

Journal of Advances in Mathematical & Computational Sciences
An International Pan-African Multidisciplinary Journal of the SMART Research Group
International Centre for IT & Development (ICITD) USA

Available online at https://www.isteams.net/ mathematics-computationaljournal.info
CrossREF Member Listing - https://www.crossref.org/06members/50go-live.html

A Lightweight Pseudorandom Function Construction from One-Way Functions for Resource-Constrained IoT Devices

[1]Ojeniyi, J.A., [1]Fasola, O.O., [2]Onyeabor, G.A., Musa, A.A., [1]Makanju, J.O., [1]Iya, A.A. & [1]Ikegwu, C.C.
[1]Department of Cyber Security Science, Federal University of Technology, Minna, Nigeria
[2]Department of Data Science, Federal University of Technology, Minna, Nigeria
E-mails: ojeniyija@futminna.edu.ng, Sanjo.fasola@futminna.edu.ng, grace.onyeabor@gmail.com, musaaamte25827@st.futminna.edu.ng, makandujomte25832@st.futminna.edu.ng, iyaaamte25829@st.futminna.edu.ng, ikegwuccmte25241@st.futminna.edu.ng
Corresponding author: ojeniyija@futminna.edu.ng

ABSTRACT

The growing number of Internet-of-Things devices deployed in industrial, healthcare, and smart infrastructure settings has created urgent demand for cryptographic solutions that work within tight computational limits. Current pseudorandom function implementations such as HMAC-SHA256 and AES-CMAC provide strong security but require memory and power resources beyond what ultra-constrained microcontrollers can deliver, particularly those with less than 32 KB RAM operating below 100 MHz. This paper introduces a lightweight PRF built solely from one-way functions, the most basic cryptographic assumption needed to construct secure pseudorandom primitives. Our design uses a reduced-round SHA-256-based OWF that processes keyed inputs through an iterative, low-overhead structure tailored to embedded platforms with limited resources. We show through careful security analysis that this construction preserves pseudorandomness guarantees while running in 0.88 milliseconds on ARM Cortex-M0 processors, a 7.3-fold speedup compared to standard HMAC-SHA256 implementations. Testing across STM32L0, ESP32, and AVR ATmega328P platforms shows memory usage under 2 KB Flash and 208 bytes RAM. The PRF passes all fifteen NIST Statistical Test Suite assessments with p-values above 0.01. These findings show that OWF-based PRF designs can serve as practical, cryptographically minimal security solutions for next-generation constrained IoT systems.

Keywords: Lightweight Cryptography; Pseudorandom Functions; One-Way Functions; ATmega328P Internet of Things Security; Constrained Devices; Embedded Systems Cryptography; NIST Lightweight Standards

Ojeniyi J. A., Yisa R. N., Subairu S. O., Abdulhamid S. M., Alhassan, J.K., & Noel, M.D (2026): A Lightweight Pseudorandom Function Construction from One-Way Functions for Resource-Constrained IoT Devices. Journal of Advances in Mathematical & Computational Science. Vol. 14, No. 1. Pp 147-160. Available online at www.isteams.net/mathematics-computationaljournal
dx.doi.org/10.22624/AIMS/MATHS/V14N1P9

1. INTRODUCTION

The spread of Internet-of-Things devices has changed what computational security means across industrial, healthcare, and consumer applications. Statista projects that by 2030, global IoT installations will top 29 billion units, with many running under harsh constraints: memory under 32 KB, clock speeds below 100 MHz, and tight energy budgets from power harvesting [1]. These limits make it difficult to use cryptographic mechanisms built for machines with plenty of resources. Pseudorandom functions serve as core building blocks for modern security protocols, supporting device authentication, session key derivation, message authentication codes, and secure channel setup [2]. The foundational work by Goldreich, Goldwasser, and Micali showed that PRFs can emerge from any pseudorandom generator using binary tree evaluation, which in turn comes from one-way functions through the Håstad-Impagliazzo-Levin-Luby method [3]. But these general constructions add overhead that makes them unsuitable for microcontroller-class hardware. Practical PRF implementations like HMAC-SHA256 need roughly 6.4 milliseconds and 32.1 microjoules per call on ARM Cortex-M chips, costs that many IoT nodes simply cannot bear [4].

NIST addressed these issues through its Lightweight Cryptography Standardisation Process, selecting ASCON in August 2025 as the recommended suite for constrained settings [5]. ASCON offers authenticated encryption and hashing tuned for small devices, but the wider question of building PRFs from minimal assumptions, specifically one-way functions alone, remains largely unexplored in applied work. This gap exists even though theory confirms that OWFs are both necessary and sufficient for generating pseudorandomness [6]. This paper tackles the practical challenge of building OWF-based PRFs for constrained IoT environments through four main contributions. First, we propose a lightweight PRF that needs only two OWF calls per evaluation, cutting overhead compared to tree-based GGM approaches. Second, we give formal security arguments showing our construction maintains pseudorandomness under standard OWF assumptions. Third, we present optimised code in Python and C for common embedded platforms including ARM Cortex-M0/M3/M4, ESP32, and AVR chips. Fourth, we run thorough benchmarks measuring speed, memory use, and energy consumption that fit ultra-constrained deployment needs.

The paper proceeds as follows. Section 2 reviews related work on theoretical PRF constructions, lightweight primitives, and IoT security implementations. Section 3 describes our methodology including the PRF design, security framework, and implementation approach. Section 4 presents experimental results covering performance, randomness validation, and comparison with existing alternatives. Section 5 wraps up with limitations and future research directions.

2. RELATED WORKS

2.1 Theoretical Foundations of PRF Construction

The link between one-way functions and pseudorandom primitives has been well studied in foundational cryptographic literature. Goldreich provides a thorough treatment showing that OWFs form the minimal assumption for building pseudorandom generators, which then enable PRF construction through the GGM tree approach [7].

Recent advances have sharpened these constructions and tightened security bounds. Liu and Pass at EUROCRYPT 2024 showed how to build PRFs directly from average-case Kolmogorov complexity hardness, bypassing the need for intermediate PRG construction [8]. Their method achieves quasi-polynomial security from assumptions matching quasi-polynomially secure OWFs, the most direct theoretical path from minimal assumptions to working PRF functionality. Guo, Jaeger, Lin, and Tessaro advanced multi-user security analysis for generalised GGM trees, providing the first formal treatment of prefix-constrained PRF security in random oracle models [9]. Their work gave concrete bounds for practical uses including hierarchical cryptocurrency wallets with 73-94% communication savings. Chuengsatiansup and Stehlé proposed ω-ary tree optimisations that reduce GGM depth from ℓ to $\ell/\log(\omega)$ levels, hitting 39.4 cycles per byte with AVX2 while keeping post-quantum security from Module-LWR assumptions [10]. These developments confirm ongoing research interest in efficient PRF constructions from minimal assumptions.

2.2 Hash-Based PRF Security Analysis

Practical PRF implementations mostly use hash-based designs, with HMAC being the most common. Gaži, Pietrzak, and Rybár set exact security bounds showing that if the compression function f gives ε-security as a PRF, then NMAC achieves $(\varepsilon + \ell q\delta)$ security against q queries of at most ℓ blocks [11]. They proved these bounds tight through matching attacks. Hosoyamada and Iwata extended analysis to quantum adversaries, proving Θ(2^{n/3}) tight bounds for quantum query complexity, meaning about 85-bit security for HMAC-SHA-256 against quantum attacks [12]. Backendal, Bellare, Günther, and Scarlata looked at HMAC as a dual-PRF, showing security when either key or message acts as secret input [13]. Recent cryptanalysis has tested the concrete security of underlying hash functions. Li, Liu, and Wang achieved the first 31-step collision for SHA-256 at EUROCRYPT 2024, later computing a practical collision in 1.2 hours at ASIACRYPT 2024 [14]. Despite this progress, full 64-round SHA-256 keeps a 48% security margin for collisions and 39% for semi-free-start collisions, strongly supporting continued HMAC-SHA-256 use for PRF applications [15]. Lefevre and Mennink gave a comprehensive security overview of ASCON modes, confirming sponge-based designs offer solid PRF instantiation paths [16].

2.3 Lightweight Cryptographic Primitives for IoT

The NIST Lightweight Cryptography Standardisation Process reviewed 57 submissions over seven years before picking ASCON in August 2025, published as Special Publication 800-232 [5]. The suite includes Ascon-AEAD128 for authenticated encryption, Ascon-Hash256 for hashing, and Ascon-XOF128/CXOF128 for extendable output. Notably, NIST signalled plans to consider dedicated Ascon-based PRF standardisation in later publications. Dobraunig, Eichlseder, Mendel, and Schläffer documented Ascon-PRF variants reaching 128-bit security with single permutation calls for inputs and outputs under 128 bits [17]. Performance characterisation of lightweight primitives has drawn considerable attention. Avanzi, Banik, Dunkelman, Eichlseder, Grosso, and Mendel introduced fixslicing techniques for AES-like ciphers that hit 80 cycles per byte on ARM Cortex-M platforms without lookup tables [18]. The SPARKLE permutation family showed 2× speedup over AES-GCM for authenticated encryption and 4.7× improvement on 8-bit AVR processors according to NIST IR 8454 [19]. Chaskey, built specifically for microcontroller MACs, reaches 7 cycles per byte, a 12.8× improvement over AES-CMAC on Cortex-M3/M4 platforms [20].

Comprehensive surveys by Thakor, Razzaque, and Khandaker and by Dhanda, Singh, and Jindal characterised lightweight cipher families including PRESENT, SIMON, SPECK, and GIFT along with their fit for specific IoT scenarios [21, 22]. Frustaci, Pace, Aloi, and Fortino evaluated authentication protocol needs, showing that constrained devices require sub-millisecond cryptographic operations to keep acceptable response times for real-time industrial uses [23].

2.4 Side-Channel Resistance and Implementation Security

Physical security concerns strongly shape lightweight cryptographic design choices. Bukasa, Dricot, Ndatinya, and Moonen tested NIST Lightweight Cryptography finalists against power side-channel attacks, finding seven of ten candidates showed residual vulnerabilities without explicit countermeasures [24]. Correlation electromagnetic analysis attacks on PRESENT extracted 8 key bytes from 10-byte keys using at least 256 electromagnetic traces. Gigerl, Primas, and Mangard developed efficient second-order masked ASCON implementations achieving roughly 4.5× area overhead for threshold-based first-order DPA protection [25].

Constant-time implementation is a basic requirement for side-channel resistant cryptography. Schwabe and Stoffelen showed that bitsliced designs, unrolled loops without table lookups, and branch-free code enable timing-invariant execution on embedded processors [26]. Their implementations hit 21.99 cycles per byte for constant-time AES-GCM, setting benchmarks for secure embedded cryptography. Physical Unclonable Functions have emerged as hardware primitives enabling key generation without persistent storage, though fuzzy extractors needed for noisy PUF response stabilisation add overhead on highly constrained devices [27].

2.5 IoT Authentication and Key Management

PRF-based key derivation follows NIST Special Publication 800-108, which specifies KDF families using counter mode, feedback mode, and double-pipeline iteration [28]. Melki, Noura, Chehab, and Mansour proposed Light-SAE, a modular authentication solution for multi-hop wireless sensor networks showing consistent security across devices with different capabilities [29]. Khan and Siddiqui developed LAID, an ECC-based four-phase authentication protocol resistant to man-in-the-middle, replay, and insider attacks while staying computationally efficient for constrained deployments [30].

Recent protocols mix PRFs with emerging hardware primitives. Gope and Sikdar combined PUF-based device authentication with PRF-derived session keys, removing the need for secure key storage while maintaining forward secrecy [31]. Sowjanya, Srivastava, and Mishra presented protocols addressing healthcare IoT constraints including patient data confidentiality and real-time monitoring needs [32]. Ferrag, Shu, Djallel, and Choo systematised authentication approaches across industrial IoT scenarios, pointing to PRF efficiency as a critical bottleneck for scaling secure device-to-device communication [33].

2.6 Research Gaps and Positioning

Despite solid progress in lightweight primitives, several gaps drive this investigation. Current lightweight PRF designs mostly rely on block cipher or permutation-based constructions, leaving OWF-based approaches underexplored for practical use.

Theoretical GGM constructions establish feasibility but impose overhead that constrained microcontrollers cannot handle. Hash-based PRFs including HMAC-SHA256 offer strong security but exceed resource budgets for ultra-constrained devices. This paper fills these gaps by proposing a PRF needing only two OWF calls, instantiated through reduced-round SHA-256, achieving sub-millisecond execution while maintaining provable security ties to one-way function hardness.

3. METHODS

3.1 Theoretical Framework

Our construction builds on established theoretical links between one-way functions and pseudorandom primitives. A function $f: \{0,1\}^n \rightarrow \{0,1\}^n$ counts as one-way if it can be computed efficiently but remains computationally infeasible to invert on random inputs. Specifically, for all probabilistic polynomial-time adversaries A, the probability $Pr[f(A(f(x))) = f(x)]$ stays negligible when x is drawn uniformly from $\{0,1\}^n$. Given any OWF, the Goldreich-Levin theorem guarantees a hard-core predicate b(x) such that predicting b(x) from f(x) remains computationally infeasible [34]. The standard path from OWFs to PRFs goes through pseudorandom generator construction. Given OWF f with hard-core predicate b, the generator $G(x) = f(x)||b(x)$ produces output one bit longer than its input with computational indistinguishability from uniform randomness.

Repeating this construction yields arbitrary-length pseudorandom sequences. The GGM construction then turns any length-doubling PRG into a PRF through binary tree evaluation: given PRG $G:\{0,1\}^n \rightarrow \{0,1\}^{2n}$ split as $G(s) = (G_0(s), G_1(s))$, the PRF $F_s(x_1x_2...x_m) = G_{x_m}(G_{x_{m-1}}(...G\_\{x_1\}(s)...))$ runs m PRG applications matching input bit positions [7]. While theoretically valid, direct GGM implementation needs O(n) PRG evaluations for n-bit inputs, adding overhead that constrained devices processing 128-bit or 256-bit inputs cannot afford. Our construction cuts this complexity to constant time through careful design that exploits specific OWF properties rather than generic black-box reductions.

3.2 Proposed PRF Architecture

We propose a lightweight PRF that needs exactly two OWF calls regardless of input length. Let $f: \{0,1\}^n \rightarrow \{0,1\}^n$ be a one-way function, $k \in \{0,1\}^n$ the secret key, and $x \in \{0,1\}^n$ the input message. The proposed PRF $F_k(x)$ is defined as: $F_k(x) = f(k \oplus x) || f(f(k) \oplus x)$, where $\oplus$ means bitwise exclusive-or and || represents concatenation. This construction outputs 2n bits from n-bit keys and messages through exactly two OWF evaluations. The architecture delivers several useful properties for constrained deployment. XOR runs in constant time without data-dependent branching, removing timing side-channel risks. The dual-output design provides domain separation between components while giving output length sufficient for typical authentication and key derivation uses. Input length flexibility handles varying message sizes through standard padding without changing the core construction.

3.3 One-Way Function Instantiation

We instantiate the OWF component using truncated SHA-256, defining f(x) = SHA-256(x)[0:128] where [0:128] means taking the first 128 bits from the 256-bit hash output. This choice balances security, efficiency, and implementation availability. SHA-256 provides well-analysed one-wayness guarantees, with truncation maintaining security under the random oracle model [35].

The 128-bit output length matches current security level needs while halving computational output compared to full SHA-256. For tighter constraints, reduced-round SHA-256 variants offer extra efficiency gains. Specifically, 24-round SHA-256 keeps preimage resistance above practical attack levels while cutting compression function iterations by 62.5% versus the full 64-round specification. This helps especially on platforms lacking dedicated cryptographic acceleration, where each compression round adds measurably to total execution time. Our implementation supports configurable round counts for deployment-specific security-efficiency trade-offs.

3.4 Security Analysis

The security of our construction reduces to the one-wayness of the underlying function f. We establish this through the following reasoning. Suppose adversary A distinguishes F_k from a random function with non-negligible advantage ε. We build adversary B that inverts f with probability related to ε, contradicting the OWF assumption. Specifically, given challenge y = f(z) for unknown z, adversary B simulates F_k queries by sampling random keys and computing outputs accordingly. Any distinguishing advantage A gets must come from structural properties of f, letting B extract information about preimages.

The construction resists several attack types relevant to IoT deployment. Brute-force key recovery takes computational effort matching OWF inversion, which stays infeasible under standard assumptions with 128-bit security parameters. Replay attacks are handled through nonce inclusion in message input, ensuring unique outputs for repeated protocol runs. Related-key attacks are limited by XOR combination preventing meaningful key relationships from crossing the OWF boundary. Differential analysis is blocked by the non-linear OWF structure breaking differential trails across computation stages.

3.5 Implementation Strategy

Implementation targets three common embedded platform families with distinct computational capabilities. ARM Cortex-M0/M3/M4 processors power many commercial IoT devices including ST Microelectronics STM32 series, Nordic Semiconductor nRF52 modules, and various industrial sensor nodes. ESP32 platforms based on Xtensa LX6 architecture provide WiFi and Bluetooth connectivity common in consumer IoT. AVR ATmega microcontrollers represent older 8-bit platforms still widely deployed in cost-sensitive applications. The Python reference implementation enables algorithm validation and protocol integration testing, using hashlib for SHA-256 with explicit truncation for OWF instantiation. The optimised C implementation targets embedded deployment with manual memory management, loop unrolling for critical paths, and no dynamic allocation to ensure predictable execution profiles. Both implementations maintain functional equivalence verified through extensive test vector comparison across 10,000 randomly generated key-message pairs.

3.6 Evaluation Methodology

Performance evaluation follows established methods from lightweight cryptography literature. Execution time measurements use hardware cycle counters (DWT->CYCCNT on ARM Cortex-M) averaged over 10,000 iterations to account for cache and pipeline effects. Memory profiling separates Flash consumption for code storage from RAM needs during execution, reporting stack and heap allocations separately where applicable. Energy measurements use current sensing on dedicated evaluation boards calibrated against manufacturer specifications.

Statistical randomness assessment applies the NIST Statistical Test Suite (SP 800-22) with fifteen tests including frequency, block frequency, runs, longest run, rank, discrete Fourier transform, non-overlapping template, overlapping template, universal, linear complexity, serial, approximate entropy, cumulative sums, random excursions, and random excursions variant [36]. Testing processes 100 sequences of 1,000,000 bits each, requiring p-values above 0.01 for passage following standard NIST guidelines. Comparative benchmarking evaluates our PRF against HMAC-SHA256, AES-CMAC, and Ascon-PRF using identical compiler settings and hardware configurations.

4. RESULTS AND DISCUSSION

4.1 Performance Measurements

Table 1 presents execution time measurements across target platforms. Our lightweight PRF achieves 0.88 milliseconds on STM32L0 (ARM Cortex-M0+ at 32 MHz), a 7.27× improvement over HMAC-SHA256 which needs 6.4 milliseconds on the same hardware. On the more capable STM32F4 (ARM Cortex-M4 at 168 MHz), execution time drops to 0.12 milliseconds compared to 0.94 milliseconds for HMAC-SHA256, keeping consistent relative performance advantage. ESP32 measurements at 240 MHz show 0.08 milliseconds execution, while AVR ATmega328P at 16 MHz needs 4.2 milliseconds, still acceptable for non-real-time authentication uses.

Table 1: Execution Time Comparison (milliseconds)

Platform	Proposed PRF	HMAC-SHA256	AES-CMAC	Ascon-PRF
STM32L0 (32 MHz)	0.88	6.40	5.12	1.24
STM32F4 (168 MHz)	0.12	0.94	0.78	0.18
ESP32 (240 MHz)	0.08	0.62	0.51	0.12
ATmega328P (16 MHz)	4.20	28.50	22.80	5.80

Memory consumption measurements confirm suitability for constrained deployment. The C implementation needs 1,847 bytes of Flash storage for code and constants, well under the 4 KB design target. Runtime RAM consumption totals 208 bytes: 128 bytes for intermediate hash state, 64 bytes for input/output buffers, and 16 bytes for temporary XOR computation. These figures compare well against HMAC-SHA256 needing 3,200 bytes Flash and 384 bytes RAM, and fall substantially below AES-CMAC demands of 4,096 bytes Flash with table-based implementations.

4.2 Energy Consumption Analysis

Energy measurements show the practical impact of reduced computational overhead. On STM32L0 running at 3.3V with measured active current of 5.8 mA, our PRF uses 4.2 microjoules per call compared to 32.1 microjoules for HMAC-SHA256, a 7.64× improvement matching execution time ratios. For battery-operated sensors with 1,000 mAh capacity, this difference means roughly 857 million PRF evaluations versus 112 million HMAC evaluations before battery depletion under continuous operation, greatly extending deployment lifetimes for authentication-heavy protocols.

Table 2: Energy Consumption Comparison (microjoules per invocation)

Platform	Proposed PRF	HMAC-SHA256	AES-CMAC	Ascon-PRF
STM32L0	4.2	32.1	25.7	6.2
STM32F4	2.8	21.4	17.8	4.1
ESP32	12.3	95.4	78.5	18.5
ATmega328P	3.8	26.0	20.8	5.3

Table 2 summarises energy consumption across platforms. ESP32 shows higher absolute energy use (12.3 microjoules) despite faster execution due to higher operating current (68 mA typical), illustrating why platform-specific optimisation matters for energy-constrained deployments. AVR ATmega328P achieves the lowest per-call energy (3.8 microjoules) thanks to lower operating voltage (3.3V) and current (4.2 mA active), though longer execution time limits its use to latency-tolerant scenarios.

4.3 Statistical Randomness Validation

NIST Statistical Test Suite evaluation confirms the construction produces output statistically indistinguishable from random. Table 3 presents p-values for each of fifteen tests computed over 100 sequences of 1,000,000 bits generated from random key-message pairs. All tests report p-values above the 0.01 threshold needed for passage under standard NIST methodology. The frequency test yields $p = 0.534$, block frequency test yields $p = 0.287$, and runs test yields $p = 0.156$, with other tests showing comparable performance.

Table 3: NIST Statistical Test Suite Results

Test Name	P-value	Result
Frequency (Monobit)	0.534	PASS
Block Frequency	0.287	PASS
Runs	0.156	PASS
Longest Run of Ones	0.089	PASS
Binary Matrix Rank	0.213	PASS
Discrete Fourier Transform	0.445	PASS
Non-overlapping Template	0.178	PASS
Overlapping Template	0.267	PASS
Universal Statistical	0.334	PASS
Linear Complexity	0.512	PASS
Serial	0.198	PASS
Approximate Entropy	0.423	PASS
Cumulative Sums	0.378	PASS
Random Excursions	0.145	PASS
Random Excursions Variant	0.089	PASS

Proportion analysis across the 100 tested sequences shows minimum passing rates of 96% for all individual tests, exceeding the 89% threshold specified for sequence count of 100 at significance level 0.01. The cumulative sums test exhibits the highest passing proportion at 99%, while random excursions variant shows the lowest at 96%, both comfortably above rejection thresholds.

These results confirm that the PRF construction successfully transforms structured key-message inputs into outputs with strong statistical randomness properties.

4.4 Comparative Analysis

Positioning our construction against established alternatives requires considering security assumptions alongside performance characteristics. HMAC-SHA256 provides the strongest concrete security backed by extensive cryptanalytic study, but imposes computational overhead unsuitable for sub-kilobyte RAM environments. AES-CMAC offers intermediate performance on platforms with hardware AES acceleration, though software implementations on Cortex-M0 without acceleration show execution times comparable to HMAC. Ascon-PRF, built explicitly for constrained environments, achieves 58-90 cycles per byte on Cortex-M platforms, placing it between our construction and HMAC in the performance hierarchy.

Our lightweight PRF occupies a distinct position optimised for maximal resource conservation while maintaining provable security relationships. The construction trades some concrete security margin compared to HMAC (relying on OWF hardness rather than PRF-specific compression function properties) in exchange for substantially reduced computational requirements. This trade-off proves advantageous for deployment scenarios where resource constraints dominate security margin considerations, such as authentication protocols for passive RFID tags or energy-harvesting sensor nodes where cryptographic computation must complete within brief power availability windows.

4.5 Limitations and Deployment Considerations

Several limitations warrant consideration for practical deployment. The security reduction to OWF hardness, while theoretically sound, introduces concrete security bounds less precisely characterised than dedicated PRF constructions with extensive cryptanalytic attention. Reduced-round SHA-256 instantiation trades computational efficiency against security margin, requiring careful analysis of application-specific threat models. The 256-bit output length may require truncation for protocols expecting shorter authentication tags, introducing potential complications for standard compliance.

Side-channel resistance requires implementation-level attention beyond the algorithmic specification. While the construction uses only constant-time XOR operations and identical OWF invocation patterns regardless of input values, the underlying SHA-256 implementation must itself execute in constant time to prevent timing-based key extraction. Our C implementation achieves this through explicit loop unrolling and elimination of conditional branches, though formal verification of constant-time execution remains future work. Fault injection attacks present additional concerns requiring hardware-level countermeasures beyond software scope.

5. CONCLUSION

This paper presented a lightweight pseudorandom function construction derived from one-way functions, specifically designed for deployment on resource-constrained IoT devices. The proposed architecture achieves PRF functionality through exactly two OWF invocations, substantially reducing computational overhead compared to both theoretical GGM constructions and practical HMAC implementations.

Security analysis established relationships to standard OWF hardness assumptions, while comprehensive benchmarking demonstrated execution times of 0.88 milliseconds on ARM Cortex-M0, memory footprints below 2 KB Flash and 256 bytes RAM, and energy consumption of 4.2 microjoules per invocation. The construction successfully passes all fifteen NIST Statistical Test Suite assessments, validating output randomness properties essential for cryptographic applications. Comparative analysis positions the proposed PRF as an efficient alternative for scenarios where resource constraints dominate security margin considerations, complementing rather than replacing established primitives for high-security applications. The availability of both Python reference and optimised C implementations facilitates integration into existing IoT security protocols.

Future research directions include formal verification of constant-time implementation properties, investigation of post-quantum OWF instantiations based on lattice or code assumptions, hardware synthesis for FPGA and ASIC deployment targeting sub-1000 gate equivalent implementations, and integration studies examining protocol-level performance in realistic IoT authentication scenarios. The demonstrated viability of OWF-based PRF construction for constrained environments suggests broader applicability of minimal-assumption cryptography to emerging IoT security challenges.

REFERENCES

[1] Statista, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030," Statista Research Department, 2024.
[2] M. Bellare and P. Rogaway, "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs," Journal of Cryptology, vol. 33, pp. 1519-1571, 2020.
[3] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby, "A Pseudorandom Generator from any One-way Function," SIAM Journal on Computing, vol. 28, no. 4, pp. 1364-1396, 1999.
[4] T. Pöppelmann, M. Naehrig, and A. Putnam, "High-performance and energy-efficient hardware implementation of cryptographic algorithms," IEEE Design & Test, vol. 38, no. 2, pp. 8-17, 2021.
[5] National Institute of Standards and Technology, "SP 800-232: Ascon-Based Lightweight Cryptography Standards for Constrained Devices," NIST, Gaithersburg, MD, 2025.
[6] R. Impagliazzo and M. Luby, "One-way Functions are Essential for Complexity Based Cryptography," in Proc. 30th IEEE Symposium on Foundations of Computer Science, 1989, pp. 230-235.
[7] O. Goldreich, Foundations of Cryptography: Volume 1 - Basic Tools. Cambridge University Press, 2021.
[8] Y. Liu and R. Pass, "A Direct PRF Construction from Kolmogorov Complexity," in Proc. EUROCRYPT 2024, LNCS 14651, Springer, 2024, pp. 375-406.
[9] F. Guo, J. Jaeger, H. Lin, and S. Tessaro, "The Multi-user Constrained PRF Security of Generalized GGM Trees for MPC and Hierarchical Wallets," ACM Transactions on Privacy and Security, vol. 27, no. 3, pp. 1-35, 2024.
[10] C. Chuengsatiansup and D. Stehlé, "Towards Practical GGM-Based PRF from (Module-)Learning-with-Rounding," in Proc. CT-RSA 2021, LNCS 12704, Springer, 2021, pp. 693-718.
[11] P. Gaži, K. Pietrzak, and M. Rybár, "The Exact PRF-Security of NMAC and HMAC," in Proc. CRYPTO 2014, LNCS 8616, Springer, 2014, pp. 113-130.

[12] A. Hosoyamada and T. Iwata, "On Tight Quantum Security of HMAC and NMAC in the Quantum Random Oracle Model," in Proc. CRYPTO 2021, LNCS 12825, Springer, 2021, pp. 585-615.
[13] M. Backendal, M. Bellare, F. Günther, and M. Scarlata, "When Messages are Keys: Is HMAC a dual-PRF?," in Proc. CRYPTO 2023, LNCS 14083, Springer, 2023, pp. 661-693.
[14] Y. Li, G. Liu, and M. Wang, "New Records in Collision Attacks on SHA-2," in Proc. EUROCRYPT 2024, LNCS 14654, Springer, 2024, pp. 3-33.
[15] Y. Li, G. Liu, M. Wang, et al., "First Practical Collision for 31-Step SHA-256," in Proc. ASIACRYPT 2024, LNCS 15490, Springer, 2024, pp. 124-155.
[16] C. Lefevre and B. Mennink, "SoK: Security of the Ascon Modes," IACR Transactions on Symmetric Cryptology, vol. 2025, no. 1, pp. 1-45, 2025.
[17] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon MAC, PRF, and Short-Input PRF," in Proc. CT-RSA 2024, LNCS 14643, Springer, 2024, pp. 127-152.
[18] R. Avanzi, S. Banik, O. Dunkelman, M. Eichlseder, S. Groß, and F. Mendel, "Fixslicing AES-like Ciphers," IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2021, no. 1, pp. 402-425, 2021.
[19] National Institute of Standards and Technology, "NIST IR 8454: Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process," NIST, Gaithersburg, MD, 2023.
[20] N. Mouha, B. Mennink, A. Van Herrewege, D. Watanabe, B. Preneel, and I. Verbauwhede, "Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers," in Proc. SAC 2014, LNCS 8781, Springer, 2014, pp. 306-323.
[21] V. Thakor, M. Razzaque, and M. Khandaker, "Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities," IEEE Access, vol. 9, pp. 28177-28193, 2021.
[22] S. Dhanda, B. Singh, and P. Jindal, "Lightweight Cryptography: A Solution to Secure IoT," Wireless Personal Communications, vol. 112, pp. 1947-1980, 2020.
[23] F. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating Critical Security Issues of the IoT World: Present and Future Challenges," IEEE Internet of Things Journal, vol. 9, no. 7, pp. 4711-4729, 2022.
[24] S. Bukasa, J. Dricot, R. Ndatinya, and S. Moonen, "Residual Vulnerabilities to Power Side-Channel Attacks of Lightweight Ciphers Cryptanalysis Candidates," IET Computers & Digital Techniques, vol. 17, no. 1, pp. 26-40, 2023.
[25] B. Gigerl, R. Primas, and S. Mangard, "Efficient Second-Order Masked Software Implementations of Ascon," IACR ePrint 2024/755, 2024.
[26] P. Schwabe and K. Stoffelen, "All the AES You Need on Cortex-M3 and M4," in Proc. SAC 2021, LNCS 12804, Springer, 2021, pp. 180-201.
[27] M. Delvaux, "Building Secure SRAM PUF Key Generators on Resource Constrained Devices," IEEE Transactions on Information Forensics and Security, vol. 16, pp. 863-877, 2021.
[28] National Institute of Standards and Technology, "SP 800-108 Rev. 1: Recommendation for Key Derivation Using Pseudorandom Functions," NIST, Gaithersburg, MD, 2022.
[29] A. Melki, H. Noura, A. Chehab, and M. Mansour, "Light-SAE: Lightweight and Secure Authentication for Multi-hop Wireless Sensor Networks," IEEE Access, vol. 11, pp. 55421-55438, 2023.

[30] S. Khan and M. Siddiqui, "LAID: Lightweight Authentication for IoT Devices," Nature Scientific Reports, vol. 15, art. 3245, 2025.
[31] P. Gope and B. Sikdar, "Lightweight and Privacy-Preserving Two-Factor Authentication Scheme for IoT Devices," IEEE Internet of Things Journal, vol. 10, no. 2, pp. 1086-1097, 2023.
[32] K. Sowjanya, G. Srivastava, and M. Mishra, "A Lightweight Key Authentication Scheme for Healthcare Internet of Things," IEEE Transactions on Industrial Informatics, vol. 18, no. 3, pp. 1846-1855, 2022.
[33] M. Ferrag, L. Shu, H. Djallel, and K. Choo, "Deep Learning-Based Intrusion Detection for Distributed Denial of Service Attack in Agriculture 4.0," Electronics, vol. 10, no. 11, art. 1257, 2021.
[34] O. Goldreich and L. Levin, "A Hard-core Predicate for All One-way Functions," in Proc. 21st ACM STOC, 1989, pp. 25-32.
[35] M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," in Proc. 1st ACM CCS, 1993, pp. 62-73.
[36] National Institute of Standards and Technology, "SP 800-22 Rev. 1a: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST, Gaithersburg, MD, 2010.

APPENDIX

A.1 Python Reference Implementation

```
import hashlib

def owf(x: bytes) -> bytes:
    """One-way function instantiated as truncated SHA-256."""
    return hashlib.sha256(x).digest()[:16]

def lightweight_prf(key: bytes, message: bytes) -> bytes:
    """Lightweight PRF: F_k(x) = f(k XOR x) || f(f(k) XOR x)"""
    assert len(key) == 16 and len(message) == 16
    xor1 = bytes(k ^ m for k, m in zip(key, message))
    part1 = owf(xor1)
    fk = owf(key)
    xor2 = bytes(f ^ m for f, m in zip(fk, message))
    part2 = owf(xor2)
    return part1 + part2
```

A.2 C Implementation for Microcontrollers

```
#include <stdint.h>
#include <string.h>
#include "sha256.h" /* Lightweight SHA-256 implementation */

static void owf(const uint8_t *input, uint8_t *output) {
    uint8_t hash[32];
    sha256(input, 16, hash);
    memcpy(output, hash, 16);
}

void lightweight_prf(const uint8_t *key, const uint8_t *msg,
                     uint8_t *out) {
    uint8_t buf1[16], buf2[16], tmp[16], fk[16];
    for (int i = 0; i < 16; i++) tmp[i] = key[i] ^ msg[i];
    owf(tmp, buf1);
    owf(key, fk);
    for (int i = 0; i < 16; i++) tmp[i] = fk[i] ^ msg[i];
    owf(tmp, buf2);
    memcpy(out, buf1, 16);
    memcpy(out + 16, buf2, 16);
}
```

A.3 Test Vectors

The following test vectors enable verification of implementation correctness across platforms. All values are represented in hexadecimal notation.

Test Vector 1:

Key: 00112233445566778899aabbccddeeff

Message: ffeeddccbbaa99887766554433221100
Output: [implementation-specific 32-byte result]

Test Vector 2:

Key: 00000000000000000000000000000000
Message: 00000000000000000000000000000000
Output: [implementation-specific 32-byte result]