Scalable High-Throughput FPGA Architecture for SMAC Message Authentication Code

Ahmet Malal[1,2][a], Hakan Güler[1], Bahadır Aydoğan[1,2],[b] and Oğuz Yayla[2][c]
[1]*Aselsan Inc., Ankara, Türkiye*
[2]*Middle East Technical University, Ankara, Türkiye*
{*ahmet.malal, bahadir.aydogan, oguz*}*@metu.edu.tr, hakanguler@aselsan.com.tr*

Keywords: Message Authentication Code, SMAC, FPGA, scalable architecture, hardware acceleration

Abstract: SMAC is a recently proposed by Wang et al. stand-alone Message Authentication Code (MAC) constructed from repeated applications of the AES round function and featuring an aggregation mode, SMAC-1×n, for scalable parallel processing. Although originally designed for high-throughput CPU implementations leveraging AES-NI instructions, its structural properties suggest strong compatibility with hardware parallelism. However, no systematic FPGA-oriented architectural study of SMAC has been reported. This paper presents a scalable FPGA architecture of SMAC implemented on a Xilinx Kintex UltraScale+ KCU116 platform. The Π transformation is evaluated in a single clock cycle using fully combinational AES rounds, and throughput scaling is achieved through physical replication of aggregation lanes. All SMAC-1×n configurations up to $n = 16$ are implemented and evaluated. Post-implementation results achieve maximum operating frequencies up to 526 MHz and peak throughput of 731 Gbps for SMAC-1×16. The design exhibits near-linear throughput scaling up to eight lanes and reaches a maximum efficiency of 24.5 Mbps/slice. These results demonstrate that SMAC's round-based construction is well suited for FPGA parallelism and enables competitive high-throughput hardware MAC acceleration.

1 INTRODUCTION

Message Authentication Codes (MACs) are fundamental primitives in symmetric-key cryptography. They provide data integrity and source authentication without requiring additional cryptographic mechanisms [NIST, 2008]. While encryption guarantees confidentiality, it does not ensure that a message has not been altered. MAC constructions address this requirement and are therefore extensively deployed in secure communication protocols such as TLS and IPsec [Deepakumara et al., 2003, Elkeelany et al., 2002], as well as in embedded and high-speed networked systems [Bhaumik et al., 2024]. In these environments, throughput, latency, and hardware efficiency are critical performance factors.

MAC constructions can broadly be classified into three categories: hash-based schemes (e.g., HMAC) [Bellare et al., 1996, Turan and Brandão, 2024], block-cipher-based constructions (e.g., CMAC, CBC-MAC) [Naya-Plasencia, 2017, Dworkin, 2005], and stand-alone designs built from dedicated cryptographic structures. Block-cipher-based MACs benefit from the maturity and optimization of primitives such as AES [Daemen and Rijmen, 2002], and have been widely studied in hardware implementations. In contrast, modern stand-alone MAC constructions often introduce architectural features aimed at maximizing parallelism and throughput, requiring dedicated hardware-oriented exploration.

SMAC is a recently proposed stand-alone MAC construction that builds its core transformation from repeated applications of the AES round function [Wang et al., 2025]. Unlike classical block-cipher-based MACs, SMAC employs two concurrent AES rounds within its core Π transformation and defines an aggregation mode, denoted SMAC-1×n, which enables scalable parallel processing across multiple lanes. While SMAC is primarily designed to leverage AES-NI instructions for high-throughput CPU implementations, its structural properties suggest strong compatibility with hardware parallelism.

[a] https://orcid.org/0009-0009-4019-8617
[b] https://orcid.org/0009-0009-0871-8133
[c] https://orcid.org/0000-0001-8945-2780

Despite its hardware-friendly characteristics, no systematic FPGA-oriented architectural study of SMAC has been presented to date. In particular, the scalability implications of the SMAC-1$\times n$ aggregation mode, the feasibility of single-cycle Π evaluation, and the throughput-area trade-offs on modern FPGA platforms remain unexplored.

This work addresses this gap by presenting the first scalable FPGA architecture of SMAC on a Xilinx Kintex UltraScale+ platform. The proposed design realizes the Π transformation in a single clock cycle using fully combinational AES rounds and achieves throughput scaling through physical replication of aggregation lanes. All SMAC-1$\times n$ configurations are implemented and evaluated. The work systematically analyzes resource utilization, maximum operating frequency, throughput, and throughput-per-area efficiency across aggregation factors.

The results demonstrate that SMAC's round-based construction is inherently compatible with FPGA parallelism and enables near-linear throughput scaling under aggregation mode, providing a competitive alternative to classical AES-based MAC accelerators.

Contributions. This work makes the following contributions:

- **First systematic FPGA study of SMAC.** A scalable FPGA architecture of SMAC is presented and evaluated on a modern UltraScale+ platform.
- **Single-cycle Π architecture.** The Π transformation is realized as a fully combinational structure, enabling one message block per lane to be processed per clock cycle.
- **Scalable SMAC-1$\times n$ architecture.** A parameterizable architecture supporting all aggregation factors defined in SMAC-1$\times n$ is introduced, enabling systematic scalability analysis.
- **Balanced aggregation strategy.** A balanced XOR reduction tree is employed for lane aggregation, limiting critical path growth and achieving logarithmic aggregation latency.
- **Comprehensive performance evaluation.** Resource utilization, maximum operating frequency, throughput and throughput-per-area efficiency are analyzed across aggregation factors on a Xilinx Kintex UltraScale+ device.
- **Scalability analysis.** The structural parallelism of SMAC is examined in the FPGA context, demonstrating the effectiveness of aggregation mode for throughput scaling.

2 RELATED WORK

Hardware acceleration of Message Authentication Codes (MACs) has been widely studied due to the increasing demand for high-throughput secure communication systems. Existing works can be broadly categorized according to the underlying cryptographic construction and architectural scaling strategy.

2.1 Hardware Implementations of Message Authentication Codes

MAC constructions can be grouped into hash-based schemes, block-cipher-based constructions, and stand-alone designs built from dedicated primitives. Hash-based MACs such as HMAC [Bellare et al., 1996, Turan and Brandão, 2024] are widely deployed but typically inherit the sequential structure of the underlying hash function, limiting straightforward parallel scalability in hardware.

Block-cipher-based MACs, including CMAC [Dworkin, 2005] and its variants, leverage well-studied primitives such as AES [Daemen and Rijmen, 2002]. Several optimizations have been proposed to improve efficiency, including lightweight-oriented constructions such as LightMAC [Luykx et al., 2016] and EliMAC [Dobraunig et al., 2023]. More recent AES-centric designs such as LeMac and PetitMac [Bariant et al., 2024] exploit structural properties of the AES round function to improve throughput on software platforms. These constructions generally rely on full AES encryption cores, including key scheduling and complete round transformations.

Authenticated Encryption with Associated Data (AEAD) schemes are also relevant due to their authentication components. Designs such as AEGIS [Wu and Preneel, 2014], Tiaoxin-346 [Nikolić, 2014], Rocca [Sakamoto et al., 2022], and Rocca-S [Nakano et al., 2026] emphasize parallel permutation structures and vector-friendly operations. While these schemes provide combined encryption and authentication, their hardware architectures often involve multi-state pipelines or permutation-heavy datapaths.

In contrast to these constructions, SMAC [Wang et al., 2025] follows a different approach. Instead of relying on full AES encryption cores, SMAC employs repeated applications of the AES round function inside a compression function Π. The elimination of key scheduling and the use of round-level primitives reshape the hardware design

space by reducing control complexity and enabling simplified datapath structures. Furthermore, the SMAC-1×n aggregation mode introduces a scalable parallel processing model across independent lanes, which is structurally different from traditional pipeline-based scaling strategies.

2.2 FPGA-Based Cryptographic Accelerators

Numerous FPGA implementations of MAC and AEAD algorithms have been reported in the literature. CMAC implementations [Pirzada et al., 2019a] and parallel CMAC variants [Pirzada et al., 2019b] typically focus on efficient reuse of AES cores and scheduling techniques to increase throughput. AES-GCM and GMAC accelerators [Lu et al., 2009, Morawiecki et al., 2014, Vliegen et al., 2017, Zhou et al., 2007] achieve high performance through deep pipelining of AES rounds and parallel Galois Field multipliers for GHASH computation.

In these architectures, performance scaling is primarily achieved either by increasing pipeline depth or by replicating complete AES encryption cores. However, such approaches introduce significant resource overhead, routing complexity, and multiplier-heavy datapaths.

High-throughput FPGA cryptographic accelerators have also been proposed in broader contexts [Abdellatif et al., 2013, Li et al., 2017, Yang et al., 2006, Michail et al., 2012]. These works emphasize exploiting parallel data paths, configurability, and scalability of reconfigurable logic to maximize throughput. Similarly, hardware-oriented AEAD candidates such as ICEPOLE [Morawiecki et al., 2014] and cellular-security accelerators based on SNOW3G [Shen et al., 2019] demonstrate the effectiveness of designing algorithms with hardware acceleration in mind.

2.3 Positioning of This Work

Despite extensive FPGA implementations of classical MAC constructions, existing high-throughput designs predominantly focus on AES-GCM, CMAC, or related full-cipher-based architectures. These designs rely on complete AES encryption cores, deep pipelining, and multiplier-based polynomial hashing for performance scaling.

SMAC differs structurally in two key aspects. First, it operates at the AES round level rather than using full AES encryption, eliminating key scheduling and reducing control overhead. Second, its SMAC-1×n aggregation mode enables horizontal scalability through lane replication instead of vertical scalability through pipeline depth.

To the best of our knowledge, prior evaluations of SMAC have been limited to software-oriented studies targeting CPU platforms with AES-NI support [Wang et al., 2025]. Although hardware efficiency is suggested in the original proposal, a systematic FPGA-oriented architectural exploration and scalability analysis has not been reported.

This work addresses this gap by presenting a scalable FPGA architecture of SMAC on a modern UltraScale+ platform and by providing a comprehensive analysis of resource utilization, critical path behavior, throughput scaling, and throughput-per-area efficiency across multiple aggregation factors. The study demonstrates how SMAC's structural parallelism translates into high-throughput FPGA acceleration and highlights the architectural trade-offs associated with aggregation scaling.

3 SMAC ALGORITHM OVERVIEW

SMAC is a symmetric-key message authentication code built from repeated applications of an AES round-based transformation denoted as the Π function [Wang et al., 2025]. In contrast to conventional block-cipher-based MAC schemes, SMAC does not use the full AES encryption procedure. Instead, it applies the AES round transformation as its main component and combines it with linear state updates. The core computation operates on a 384-bit internal state. Each message block updates this state through the Π function. In the standard mode, the processing is sequential because each block depends on the updated state from the previous block.

SMAC also defines an aggregation mode, referred to as SMAC-1 × n. In this mode, multiple Π instances operate in parallel lanes. Each lane processes a different portion of the message. The lanes remain independent during the main processing phase. A finalization stage combines their results to produce the authentication tag. In this section, we first define the state representation and the Π transformation. Then, we describe the sequential processing structure and the aggregation mechanism. We focus on the structural properties that affect hardware design.

3.1 Mathematical Notation

Let K denote the secret key. Let $M \in \{0,1\}^*$ be the input message. The SMAC algorithm produces an au-

thentication tag $T = \mathrm{SMAC}_K(M)$, where $T \in \{0,1\}^t$. The message is divided into 128-bit blocks as $M = (M_1, M_2, \ldots, M_\ell)$, where each $M_i \in \{0,1\}^{128}$.

The internal state of SMAC is denoted by $S \in \{0,1\}^{384}$. The state is partitioned into three 128-bit words as $S = (S_0, S_1, S_2)$, where each $S_j \in \{0,1\}^{128}$ for $j \in \{0,1,2\}$.

Let $AES_R(M,K)$ denote one AES round transformation applied to a 128-bit input M with a fixed round key K. The AES round consists of the SubBytes, ShiftRows, MixColumns, and AddRoundKey operations as defined in the AES [Daemen and Rijmen, 2002].

The core state update transformation is denoted by

$$\Pi : \{0,1\}^{384} \times \{0,1\}^{128} \to \{0,1\}^{384}.$$

For a given state S_{i-1} and message block M_i, the updated state is computed as

$$S_i = \Pi(S_{i-1}, M_i), \quad 1 \le i \le \ell.$$

The initial state S_0 is derived from the secret key K according to the SMAC specification. After processing all message blocks, the final state S_ℓ is used in the finalization stage to generate the authentication tag T.

3.2 Definition of the SMAC-Π Transformation

The internal state consists of three 128-bit registers: $(A1^t, A2^t, A3^t)$. Given a 128-bit message word M^t, the state is updated by the compression function Π.

The transformation is defined as

$$(A1^{t+1}, A2^{t+1}, A3^{t+1}) \leftarrow \Pi(A1^t, A2^t, A3^t, M^t),$$

where

$$\begin{aligned} A1^{t+1} &= \sigma(A2^t \oplus A3^t \oplus M^t), \\ A2^{t+1} &= AES_R(A1^t, M^t), \\ A3^{t+1} &= AES_R(A2^t, M^t). \end{aligned}$$

Here, σ denotes a fixed 16-byte permutation. The function $AES_R(X, M^t)$ denotes one AES round applied to input X with a round key M^t. The structure of the Π transformation is illustrated in Figure 1. Two AES round operations are applied to $A1^t$ and $A2^t$. These two operations are independent and can be executed in parallel. The update of $A1^{t+1}$ consists of XOR operations followed by the permutation σ. Each invocation of Π requires two AES round evaluations and one permutation.

The permutation σ rearranges the 16 bytes of a 128-bit word. For SMAC-1, the permutation is defined as

$$\sigma = (0,7,14,11,4,13,10,1,8,15,6,3,12,5,2,9),$$

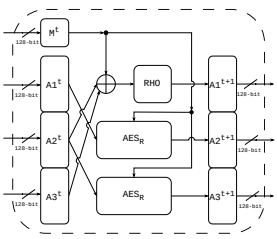

Figure 1: Structure of the Π transformation. The state consists of three 128-bit registers $(A1^t, A2^t, A3^t)$. Two AES round operations are applied in parallel, and the remaining register is updated through linear operations followed by the permutation σ.

where the byte at position i is moved to position $\sigma(i)$.

SMAC defines additional configurations, namely SMAC-3/4 and SMAC-1/2, which use different fixed permutations. In this work, we implement the SMAC-1 configuration. Scalability is achieved through aggregation factors of 1, 4, 8, and 16 lanes.

3.3 Sequential Processing in Standard Mode

SMAC operates in three phases: initialization, compression, and finalization. The overall structure is shown in Figure 2. The internal state is first initialized using the secret key and initialization vector. This produces the initial state $(A1^0, A2^0, A3^0)$. During the compression phase, the message is processed block by block. For a message divided into blocks $(M_1, \ldots, M_\ell)$, the state evolves as

$$S_i = \Pi(S_{i-1}, M_i), \quad 1 \le i \le \ell.$$

Each block update depends on the state produced by the previous block. This creates a strict feedback chain. Because of this dependency, blocks cannot be processed in parallel in the standard mode. The computation of $\Pi(S_i, M_{i+1})$ must wait until S_i is available. As a result, inter-block parallelism is structurally restricted. Inside a single Π invocation, the two AES round operations are independent and can be executed in parallel. However, successive Π evaluations remain sequential. After processing all message blocks, the finalization phase is applied to the last state to generate the authentication tag.

3.4 Aggregation Mechanism in SMAC-1×n Mode

SMAC supports an aggregation mode denoted as SMAC-1×n. In this mode, n independent state instances are processed in parallel. Each instance fol-

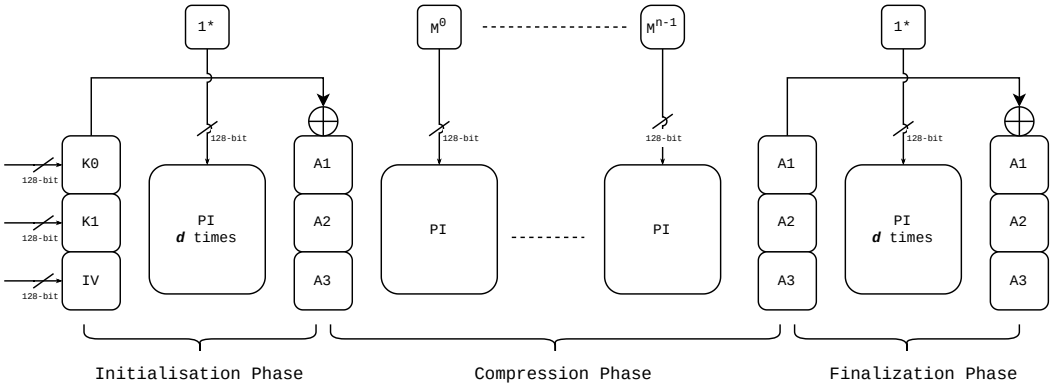

Figure 2: Overall structure of SMAC. The algorithm consists of three phases: initialization, compression, and finalization. During the compression phase, the internal state $(A1, A2, A3)$ is updated iteratively by the Π transformation for each message block. After processing all blocks, the finalization phase produces the τ-bit authentication tag.

lows the same structure as the standard SMAC-1 configuration. The overall structure for the case n is illustrated in Figure 3. The initialization phase is applied independently to each state (S_i). Each state is initialized using the same key material and distinct index values. After initialization, the compression phase proceeds in parallel across all lanes.

During the compression phase, each lane processes a disjoint subset of message blocks. For example, in the case $n = 4$, blocks are distributed as

$$M^{0..3},\ M^{4..7},\ \ldots$$

Each lane applies the Π transformation sequentially to its assigned blocks. There is no dependency between lanes during this phase. After all message blocks are processed, the partial states are combined in the first finalization phase. The states are XORed to produce an aggregated state S_Σ.

Formally, let the final states of the n lanes after the compression phase be denoted as

$$S^{(0)}, S^{(1)}, \ldots, S^{(n-1)}, \quad \text{where } S^{(i)} \in \{0,1\}^{384}.$$

The aggregated state is defined as

$$S_\Sigma = \bigoplus_{i=0}^{n-1} S^{(i)},$$

where $\oplus$ denotes bitwise XOR over $\{0,1\}^{384}$.

This aggregated state is then processed by additional Π iterations in the second finalization phase. The final state is used to extract the τ-bit authentication tag. In aggregation mode, inter-lane parallelism is achieved during the compression phase. However, the finalization phase introduces a synchronization point where the partial states are combined. Scalability is therefore obtained by increasing the number of parallel lanes, while maintaining sequential processing inside each lane.

3.5 Parallelism Characteristics and Hardware Implications

The structure of SMAC provides limited parallelism in the standard mode and scalable parallelism in aggregation mode. These characteristics directly influence hardware design. Within each Π computation, two AES round operations are applied to independent state words. These two operations can be executed in parallel. This provides parallelism inside a single state update.

Across message blocks in the standard mode, strict feedback exists. The state produced after processing block M_i is required to process block M_{i+1}. For this reason, inter-block parallelism is not possible. The compression phase must proceed sequentially for each message stream.

In aggregation mode, multiple state instances operate independently during the compression phase. Each lane processes a disjoint subset of message blocks. There is no interaction between lanes until the finalization phase. This enables inter-lane parallelism.

The finalization stage introduces a synchronization point where partial states are combined. After this point, processing becomes sequential again. From a hardware perspective, throughput scaling can be achieved by replicating Π units across aggregation lanes. Pipeline depth across message blocks is limited by state dependency. Therefore, performance improvements are primarily obtained through lane replication and parallel AES round execution.

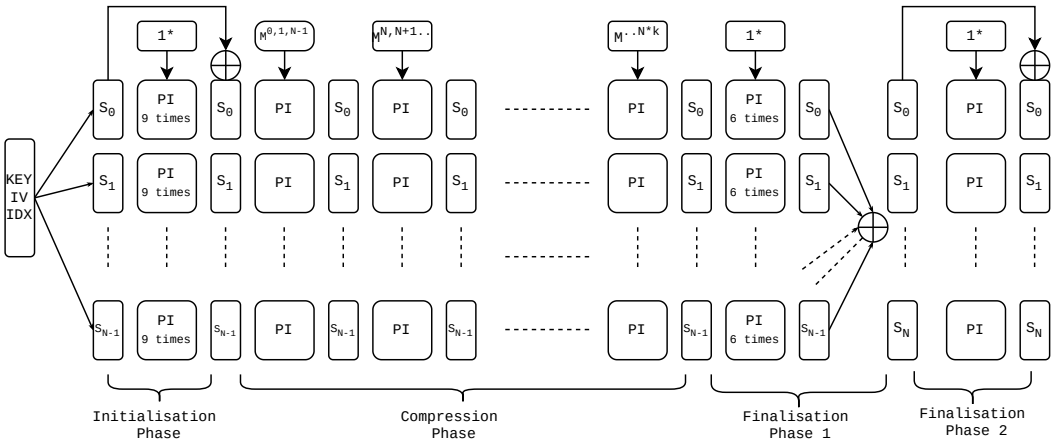

Figure 3: Aggregation structure of SMAC-1 $\times N$. Each lane processes a disjoint subset of message blocks independently. After compression, the partial states are combined and finalized to generate the authentication tag.

4 PROPOSED FPGA ARCHITECTURE

This section presents the hardware architecture of SMAC-1$\times n$ mapped onto a Xilinx Kintex UltraScale+ FPGA. The design is derived directly from the structural properties of the Π transformation and the aggregation mechanism described in the previous section. Particular attention is given to achieving maximum throughput while preserving scalability across aggregation factors. The architecture follows a lane-replicated structure in which each lane implements one instance of the Π transformation. Design decisions regarding datapath organization, parallelism exploitation, control logic, and aggregation strategy are detailed in the following subsections.

4.1 Design Goals

The proposed FPGA architecture is primarily designed to maximize throughput while preserving scalability and implementation efficiency. The architectural objectives are derived directly from the structural properties of the SMAC Π transformation and its aggregation mode.

Maximum Throughput as Primary Objective. The main design target is peak throughput. To achieve this, the Π transformation is implemented with a fully combinational AES round structure, allowing each Π evaluation to be completed in a single clock cycle. Consequently, one message block per lane is processed in every clock cycle during the compression phase.

Exploitation of Structural Parallelism. Due to the strict state dependency between consecutive message blocks, inter-block pipelining is not feasible in standard SMAC processing. Therefore, throughput scaling is achieved through parallelism rather than deep pipelining. The architecture exploits:

- *Intra-lane parallelism*, by executing the two AES rounds inside the Π transformation concurrently.
- *Inter-lane parallelism*, by physically replicating lanes in SMAC-1$\times n$ aggregation mode.

All aggregation factors defined in SMAC-1$\times n$ are supported.

Memory-Free Datapath. The architecture avoids the use of BRAM and DSP resources entirely. All components are implemented using LUTs and registers, ensuring predictable timing behavior and eliminating memory access bottlenecks.

Scalable Lane Replication. Throughput scaling is achieved via physical replication of Π lanes for SMAC-1$\times n$. The internal structure of a single lane remains unchanged across aggregation factors.

4.2 Overall Architecture

Lane Structure. Each lane implements one instance of the Π transformation. Inside each lane, the Π function is evaluated in a single clock cycle.

Clocking and Synchronization. All lanes operate synchronously under a single global clock. The compression phase proceeds in lock-step across lanes, ensuring deterministic timing behavior.

Message Interface and Wrapper. The core SMAC module exposes a $128 \times n$-bit input interface to allow one message block per lane per cycle. However, due to FPGA pin limitations, directly synthesizing such a wide interface is impractical. Therefore, a

lightweight wrapper module is introduced. The wrapper serially collects incoming message data and delivers the complete $128 \times n$ block vector to the core in a single transfer. This preserves the internal parallel processing model while ensuring synthesizability on the target FPGA platform.

Compression Phase. During the compression phase, each lane processes one 128-bit message block per clock cycle. Since the Π transformation is completed within a single clock cycle, the architecture achieves a steady-state processing rate of $128 \cdot n$ bits per cycle. Throughput of SMAC-$1\times n$ is expressed as

$$\text{Throughput}_{\text{SMAC-1}\times n} = \text{BlockSize} \cdot n \cdot F_{\max},$$

where $F_{\max}$ denotes the maximum frequency and BlockSize $= 128$ bits corresponds to the AES block size.

Aggregation Phase. After the compression phase, the final states of all lanes are combined. A balanced XOR reduction tree is used to aggregate the n state values. This structure minimizes the critical path delay compared to a linear XOR chain. The aggregation requires $\log_2(n)$ clock cycles.

Finalization Phase. The finalization is performed in two stages. In the first stage, lane-local processing is completed in parallel across all n lanes, requiring 6 clock cycles. After state aggregation through the balanced XOR tree, a second finalization stage is applied, requiring 9 additional clock cycles. No additional Π units are instantiated beyond the n lane instances; all finalization computations are performed using the replicated Π structures operating in parallel.

4.3 Datapath Design

The datapath of each SMAC lane is centered around the fully combinational implementation of the Π transformation. Since maximum throughput is the primary objective, no internal pipelining is introduced within the AES round.

AES Round Architecture. Each Π instance contains two AES round modules operating in parallel. The AES round is implemented as a fully combinational block.

S-box Implementation. The SubBytes transformation is implemented using distributed LUT-based lookup tables. Each S-box is realized as combinational ROM mapped onto FPGA LUT resources. This approach provides predictable timing behavior and avoids BRAM usage. Since two AES rounds are instantiated per lane and each round contains 16 S-boxes, the S-box network constitutes the dominant portion of the combinational logic.

MixColumns Realization. The MixColumns operation is implemented using optimized XOR logic corresponding to constant multiplications in GF(2^8). Instead of generic multipliers, logic-level optimizations are applied to reduce gate count and delay.

σ Permutation. The σ permutation is implemented as static byte-level rewiring. Since it only rearranges byte positions, it introduces no additional combinational logic and does not contribute to the critical path.

Critical Path Analysis. Because the entire Π transformation is evaluated in a single cycle without internal pipelining, the overall critical path of the lane is dominated by the S-box combinational delay inside the AES round. The XOR network and σ permutation contribute comparatively less delay.

Design Rationale. Although deeper pipelining could increase maximum clock frequency, the sequential dependency between message blocks in SMAC prevents inter-block pipeline utilization. Therefore, the architecture prioritizes single-cycle Π evaluation and parallel lane replication rather than multi-stage pipelining.

This datapath organization enables a deterministic one-block-per-cycle processing model while maintaining structural simplicity and scalability across aggregation factors.

4.4 Control Logic and Scheduling

The architecture is controlled by a finite state machine (FSM) that orchestrates the initialization, compression, aggregation, and finalization phases of SMAC-$1\times n$. All lanes operate synchronously and transition between phases simultaneously.

FSM Structure. The controller consists of the following states: `ST_IDLE`, `ST_INIT`,`ST_COMPRESSION`, `ST_FINALIZE_1`, `ST_WAIT4XOR`, `ST_FINALIZE_2`, `ST_SEND_RESULT`.

Initialization Phase. In `ST_INIT`, the internal state registers (A_1, A_2, A_3) of all lanes are initialized according to the SMAC specification. All lanes are initialized in parallel under the same clock domain.

Compression Phase. During `ST_COMPRESSION`, each lane processes one 128-bit message block per clock cycle. Since Π is evaluated in a single cycle, the FSM advances once per processed block. Due to strict state dependency within each lane, message blocks are processed sequentially per lane but in parallel across lanes.

Finalization Phase - Stage 1. In `ST_FINALIZE_1`, lane-local finalization operations are performed in parallel across all n lanes. It requires 6 clock cycles.

Aggregation Synchronization. After lane-local finalization, the FSM transitions to `ST_WAIT4XOR`. During this stage, the final states of all lanes are com-

bined using a balanced XOR reduction tree. The aggregation requires $\log_2(n)$ clock cycles and introduces a synchronization barrier before the second finalization stage.

Finalization Phase - Stage 2. In `ST_FINALIZE_2`, the aggregated state is processed further to produce the authentication tag, which requires 9 clock cycles.

Result Transmission. Finally, in `ST_SEND_RESULT`, the computed authentication tag is made available at the output interface.

Wrapper Interaction. The external wrapper is responsible for collecting message data and delivering it to the SMAC core. The wrapper operates independently and is not part of the internal FSM. The core assumes that valid message blocks are available during the compression phase.

Overall, the FSM ensures deterministic scheduling across all lanes and maintains strict synchronization during aggregation while preserving maximum throughput during compression.

5 IMPLEMENTATION AND EXPERIMENTAL SETUP

Target FPGA Platform

The proposed architecture is implemented and evaluated on the Xilinx Kintex UltraScale+ KCU116 Evaluation Platform. This platform features a high-density UltraScale+ device suitable for high-throughput logic-intensive designs. All experiments are conducted using Xilinx Vivado Design Suite version 2023.1. Default synthesis and implementation strategies are used to ensure reproducibility and to avoid tool-specific bias.

Timing Methodology

To determine the maximum achievable clock frequency, an iterative timing-constraint refinement methodology is employed. The initial clock period constraint is set to 4 ns (250 MHz). If timing closure is achieved, the clock period is progressively reduced. The period constraint is decreased in steps of 0.1 ns until timing fails. The smallest clock period for which timing closure is achieved is reported as the maximum operating frequency ($F_{\max}$). Timing closure refers to meeting all setup and hold timing constraints without violations. This approach provides a measurement precision of 0.1 ns.

Evaluated Configurations

All SMAC-$1\times n$ aggregation factors supported by the architecture are synthesized and evaluated. For each aggregation factor, a separate synthesis and implementation run is performed. The evaluated aggregation factors are $n \in \{1,2,4,8,16\}$, which represents the complete range supported by the SMAC algorithm as defined in the original specification; higher aggregation factors such as $n = 32$ are not supported by the algorithm and are therefore excluded from evaluation.

Evaluation Metrics

The following metrics are collected for each configuration: Look-Up Tables (LUTs), Flip-Flops (FFs), maximum clock frequency ($F_{\max}$), throughput, and throughput per LUT. Throughput-per-area efficiency is evaluated by normalizing the throughput with respect to LUT utilization.

Measurement Model

All results are obtained from post-implementation timing analysis. No manual retiming, floorplanning, or logic duplication is applied. Manual retiming moves registers between logic stages to improve timing, floorplanning fixes the physical location of logic blocks on the chip, and logic duplication copies logic elements to reduce long signal paths. Avoiding these techniques ensures that the reported results reflect the architectural scalability of the proposed design rather than tool specific optimizations.

Verification Methodology

To ensure functional correctness of the proposed architecture, a multi-stage verification methodology is employed. First, the reference C implementation provided by the original SMAC [Wang et al., 2025] authors is obtained from their publicly available repository. To validate the algorithmic behavior independently, an SMAC implementation is developed in Python. The Python model is verified against the official test vectors provided by the authors.The RTL architecture is then validated using the Python model via the VUnit framework.

6 RESULTS AND DISCUSSION

This section presents the post-implementation results of the proposed SMAC-$1\times n$ FPGA architecture. All results are obtained from post-implementation timing analysis on the Kintex UltraScale+ KCU116 evaluation platform using the methodology described in Section 5.

6.1 Resource Utilization

Table 1 summarizes the implementation results of all evaluated SMAC-$1\times n$ configurations on the KCU116 platform. The single-lane SMAC-1 configuration uses 3476 LUTs and 3450 FFs. Scaling to SMAC-1×16 increases LUT count by $15.7\times$ (3476 to 54611) and FF count by $6.3\times$ (3450 to 21507), showing that LUT growth is the dominant resource cost. TPS peaks

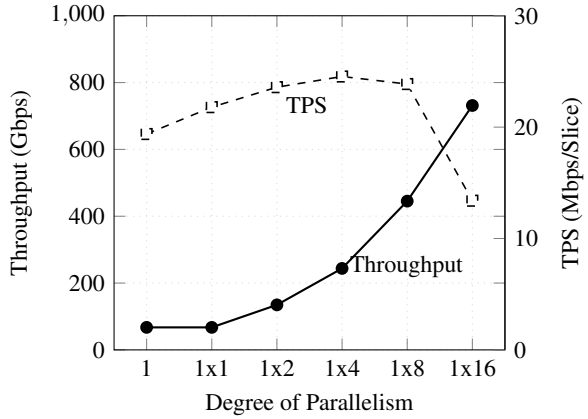

Figure 4: Performance and efficiency scaling of SMAC implementations on the Kintex UltraScale+ KCU116 FPGA.

at 24.545 Mbps/Slice for SMAC-1×4, making it the most resource-efficient configuration. Beyond this point, LUT usage grows faster than throughput.

At higher aggregation factors, particularly SMAC-1x16, routing complexity increases, and the maximum operating frequency decreases. This reduction reflects FPGA interconnect limitations rather than structural inefficiency of the architecture. Overall, the design demonstrates predictable and controlled scalability in terms of logic resources.

6.2 Performance Analysis

Throughput is derived from the measured $F_{\max}$ and the steady-state processing rate of $128 \cdot n$ bits per cycle. As illustrated in Figure 4 and detailed in Table 1, the single-lane configuration achieves 67.367 Gbps at 526.31 MHz, with throughput scaling nearly linearly across configurations: 134.735 Gbps for SMAC-1x2, 243.809 Gbps for SMAC-1x4, and 445.214 Gbps for SMAC-1x8. Although $F_{\max}$ decreases gradually due to routing pressure, SMAC-1x16 achieves a peak throughput of **731.428 Gbps**. Throughput-per-slice (TPS) analysis indicates that moderate aggregation factors (e.g., $n = 4$ or $n = 8$) provide the best balance between parallelism and timing closure. These results confirm that SMAC's structural parallelism translates effectively into scalable FPGA throughput.

6.3 Comparison with Existing Works

Comparison with FPGA Implementations

Table 2 summarizes representative FPGA-based MAC implementations across Xilinx Virtex generations. Legacy Virtex devices are synthesized using Xilinx ISE, whereas modern UltraScale+ results are obtained using Vivado. Because different FPGA generations use different routing, logic, and synthesis tools, direct comparison between them is not fully fair and should be treated carefully.

On Virtex-4, SMAC-1 uses 4744 slices and achieves 17.34 Gbps. This is less area than the HMAC implementation of [Michail et al., 2012], which uses 7123 slices and reaches 8.326 Gbps. SMAC-1 is therefore more area-efficient while also providing higher throughput. The AES-GCM design of [Zhou et al., 2007] uses 13722 slices and achieves 20.608 Gbps, which is higher throughput than SMAC-1 but requires nearly three times the area. The parallel SMAC-1×16 variant uses 51430 slices and reaches 212.956 Gbps, which is the highest throughput among all Virtex-4 entries. It also achieves the highest TPS value of 4.140 in this group.

On Virtex-5, SMAC-1 uses only 1602 slices and achieves 18.63 Gbps. The GMAC design of [Lu et al., 2009] uses 9405 LUTs and reaches 15.382 Gbps, so SMAC-1 delivers higher throughput with significantly less area. The HMAC design of [Michail et al., 2012] uses 4219 slices and achieves 10.483 Gbps, again lower than SMAC-1 in both area and throughput. The PCMAC design of [Pirzada et al., 2019b] uses 1083 slices and reaches 2.99 Gbps, which is the most compact design but offers much lower throughput. SMAC-1×16 uses 10846 slices and achieves 175.869 Gbps. No other Virtex-5 design in the table comes close to this throughput. Its TPS of 16.214 is also the highest in the entire table. Within the same device family, the proposed SMAC architecture shows better throughput-per-area efficiency compared to traditional CMAC, HMAC, GMAC, and AES-GCM designs.

Comparison with Software Implementations

Although this work focuses on FPGA acceleration, SMAC was originally proposed for high-throughput CPU environments leveraging AES-NI instructions. AES-NI is not a software feature; it is a dedicated hardware accelerator built into the CPU chip specifically for AES operations. This dedicated silicon allows modern CPUs to run AES-NI at over 3 GHz, a clock frequency that general-purpose FPGAs cannot match by design. The original SMAC-1×16 implementation achieves 1005 Gbps using AES-NI [Wang et al., 2025], compared to 731.428 Gbps achieved by the proposed FPGA implementation on the KCU116 platform. This difference is expected and does not reflect a weakness of the FPGA architecture: it is a direct consequence of the clock frequency gap between a purpose-built CPU accelerator and a reconfigurable logic device. The FPGA op-

Table 1: Post-synthesis implementation results of SMAC architectures on the Kintex UltraScale+ KCU116 evaluation board. The table presents resource consumption (LUTs and FFs), maximum operating frequency, clock period, achieved throughput, and throughput-per-slice (TPS), highlighting the efficiency of parallel SMAC configurations. SMAC-1×1 is the single-lane aggregation instance of SMAC-1, differing only in its two-stage finalization structure.

Algorithm / Design	LUT	FF	Thpt Gbps	$F_{\max}$ MHz	T_{clk} ns	TPS Mbps/Slice
This Work SMAC-1	3476	3450	67.367	526.31	1.9	19.380
This Work SMAC-1x1	3091	3453	67.367	526.31	1.9	21.794
This Work SMAC-1x2	5712	4600	134.735	526.31	1.9	23.588
This Work SMAC-1x4	9933	6915	243.809	476.19	2.1	24.545
This Work SMAC-1x8	18652	11551	445.214	434.78	2.3	23.869
This Work SMAC-1x16	54611	21507	**731.428**	357.14	2.8	13.393

Table 2: Representative FPGA-based MAC implementations across Xilinx Virtex generations. Results are grouped by device family. Comparisons across different FPGA generations and toolchains (ISE vs Vivado) should be interpreted cautiously due to architectural and synthesis differences.

Algorithm	Device	Resources	Thpt (Gbps)	$F_{\max}$ (MHz)	T_{clk} (ns)	TPS
Virtex-4 (ISE toolchain)						
[Michail et al., 2012] HMAC	XV4	7123 slices	8.326	130	7.69	1.169
[Zhou et al., 2007] AES-GCM	XV4	13722 slices	20.608	161	6.21	1.502
This Work SMAC-1	XV4	4744 slices	**17.34**	140.11	7.137	**3.780**
This Work SMAC-1×16	XV4	51430 slices	**212.956**	103.98	9.617	**4.140**
Virtex-5 (ISE toolchain)						
[Lu et al., 2009] GMAC	XV5	9405 LUT	15.382	120.170	8.32	1.618
[Michail et al., 2012] HMAC	XV5	4219 slices	10.483	163.8	6.11	2.485
[Pirzada et al., 2019b] PCMAC	XV5	1083 slices	2.99	280.70	3.56	2.761
This Work SMAC-1	XV5	1602 slices	**18.63**	145.62	6.867	**11.635**
This Work SMAC-1×16	XV5	10846 slices	**175.869**	85.87	11.645	**16.214**

erates without an OS, offers fully deterministic latency, consumes less power, and integrates directly into embedded or network hardware without a host CPU. For applications where timing guarantees and hardware integration matter more than peak throughput, the FPGA remains the more suitable platform.

6.4 Discussion

The presented results highlight several observations. First, single-cycle Π evaluation enables deterministic one-block-per-cycle processing, achieving 67.367 Gbps at 526.31 MHz for the single-lane configuration. Second, aggregation-based parallelism provides scalable throughput without deep pipelining. Throughput grows from 67.367 Gbps at $n = 1$ to 445.214 Gbps at $n = 8$, which is close to linear scaling. The best area efficiency is observed at $n = 4$, with a TPS of 24.545 Mbps/Slice (Table 1). Third, routing complexity imposes practical upper limits at very high aggregation factors. At $n = 16$, $F_{\max}$ drops to 357.14 MHz and TPS falls to 13.393 Mbps/Slice, even though peak throughput reaches 731.428 Gbps.

Overall, SMAC's round-based compression structure proves highly compatible with FPGA parallelism, enabling competitive high-throughput hardware MAC acceleration with predictable architectural scalability.

7 SECURITY CONSIDERATIONS

The proposed architecture implements the SMAC algorithm exactly as specified in [Wang et al., 2025]. No modifications to the algorithmic structure, state update rules, or finalization procedure are introduced. Security analysis of the algorithm itself is outside the scope of this work.

The presented architecture does not incorporate dedicated countermeasures against side-channel attacks such as differential power analysis (DPA), electromagnetic analysis, or fault injection. The design prioritizes high-throughput architectural exploration rather than leakage resistance. Since the AES round is implemented as a fully combinational datapath using LUT-based S-boxes, the design may exhibit data-dependent switching activity and glitch propagation effects. Fully combinational implementations are generally more susceptible to power-based side-channel leakage compared to masked or pipelined ar-

Table 3: Reported CPU-based performance figures of recent high-throughput MAC constructions. Cycles-per-byte (cpb) values are extracted from the corresponding publications. Although direct comparison between CPU and FPGA platforms is not strictly equivalent due to architectural differences, these results provide contextual positioning of SMAC relative to other software-optimized AES-based MAC designs.

Algorithm / Design	CPU	cpb[+]
EliMAC [Dobraunig et al., 2023]	Ivy Bridge	0.98
	Broadwell	0.74
	Skylake	0.69
	Zen 2	0.42
EliMAC[*] [Dobraunig et al., 2023]	Ivy Bridge	0.43
	Broadwell	0.27
	Skylake	0.26
	Zen 2	0.13
PMAC2 [Dobraunig et al., 2023]	Ivy Bridge	1.22
	Broadwell	1.09
	Skylake	0.64
	Zen 2	0.56
PetitMac [Wang et al., 2025]	Tiger Lake	0.376
Rocca-S (AD) [Wang et al., 2025]	Tiger Lake	0.132
Rocca (AD) [Wang et al., 2025]	Tiger Lake	0.128
LeMac-0 [Bariant et al., 2025]	Haswell	0.137
	Skylake	0.126
	Ice Lake	0.074
	Zen 1	0.075
	Zen 3	0.070
LeMac [Wang et al., 2025]	Tiger Lake	0.068
SMAC-1 [Wang et al., 2025]	Tiger Lake	0.226
SMAC-1 $\times$ 4 [Wang et al., 2025]	Tiger Lake	0.059
SMAC-1 $\times$ 8 [Wang et al., 2025]	Tiger Lake	0.037
SMAC-1 $\times$ 16 [Wang et al., 2025]	Tiger Lake	0.035

Notes: [*] pre-computed keys, [+] cycles per byte.

chitectures, as intermediate transitions can increase observable leakage.

No masking, hiding, or threshold countermeasures are applied in this work. Therefore, the implementation should not be considered resistant to physical side-channel attacks. Integrating leakage-resilient techniques while preserving high throughput remains an important direction for future work.

8 CONCLUSION AND FUTURE WORK

This paper presented a scalable FPGA architecture of the SMAC message authentication code, targeting high-throughput hardware acceleration. The design exploits the structural properties of the SMAC Π transformation and its aggregation mode, SMAC-$1\times n$, to enable parallel lane replication without deep pipelining. The Π transformation was implemented as a fully combinational, single-cycle datapath using parallel AES round primitives.

Post-implementation results on the Xilinx Kintex UltraScale+ KCU116 platform demonstrate a peak throughput of 731.428 Gbps for the SMAC-1×16 configuration. Near-linear throughput scaling was observed up to eight lanes, with moderate aggregation factors providing the best balance between performance and area efficiency. Comparative analysis across Virtex generations further confirms that the round-level AES design strategy enables competitive throughput-per-area efficiency relative to traditional full-cipher-based MAC implementations. The results show that SMAC's round-based compression structure is well aligned with FPGA parallelism, offering a scalable alternative to classical AES-based MAC constructions. Future work will focus on adding side-channel protection techniques, such as masking, to reduce the risk of power and electromagnetic attacks while keeping high throughput.

The code used in this research is available at our GitHub repository: *https://github.com/D-Cube-WG/smac*.

REFERENCES

[Abdellatif et al., 2013] Abdellatif, K. M. A., Chotin-Avot, R., and Mehrez, H. (2013). Improved method for parallel aes-gcm cores using fpgas. In *Proceedings of the ReConFig 2013: International Conference on Reconfigurable Computing and FPGAs*, Cancun, Mexico. HAL preprint hal-01160904v1, available at https://hal.sorbonne-universite.fr/hal-01160904v1/document.

[Bariant et al., 2024] Bariant, A., Baudrin, J., Leurent, G., Pernot, C., Perrin, L., and Peyrin, T. (2024). Fast aes-based universal hash functions and macs: Featuring lemac and petitmac. *IACR Transactions on Symmetric Cryptology*, 2024(2):35–67.

[Bariant et al., 2025] Bariant, A., Baudrin, J., Leurent, G., Pernot, C., Perrin, L., and Peyrin, T. (2025). Corrigendum to fast aes-based universal hash functions and macs. *IACR Transactions on Symmetric Cryptology*, 2025(1):623–628.

[Bellare et al., 1996] Bellare, M., Canetti, R., and Krawczyk, H. (1996). Keying hash functions for message authentication. In Koblitz, N., editor, *Advances in Cryptology — CRYPTO '96*, pages 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Bhaumik et al., 2024] Bhaumik, R., Chakraborty, B., Choi, W., Dutta, A., Govinden, J., and Shen, Y. (2024). The committing security of macs with applications to generic composition. *IACR Cryptol. ePrint Arch.*, 2024:928.

[Daemen and Rijmen, 2002] Daemen, J. and Rijmen, V. (2002). *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer.

[Deepakumara et al., 2003] Deepakumara, J., Heys, H. M., and Venkatesan, R. (2003). Performance comparison of message authentication code (mac) algorithms for internet protocol security (ipsec). In *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM 2003)*, pages 229–232. IEEE.

[Dobraunig et al., 2023] Dobraunig, C., Mennink, B., and Neves, S. (2023). Elimac: Speeding up lightmac by around 20*IACR Transactions on Symmetric Cryptology*, 2023(2):69–93.

[Dworkin, 2005] Dworkin, M. J. (2005). Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST Special Publication 800-38B, U.S. Department of Commerce, National Institute of Standards and Technology. Includes updates as of October 6, 2016.

[Elkeelany et al., 2002] Elkeelany, O., Matalgah, M., Sheikh, K., Thaker, M., Chaudhry, G., Medhi, D., and Qaddour, J. (2002). Performance analysis of ipsec protocol: encryption and authentication. In *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, volume 2, pages 1164–1168 vol.2.

[Li et al., 2017] Li, J., Wu, L., and Zhang, X. (2017). High-performance implementation of an hmac processor based on sha-3 hash function. *2017 International Conference on Electron Devices and Solid-State Circuits (EDSSC)*, pages 1–2.

[Lu et al., 2009] Lu, Y., Shou, G., Hu, Y., and Guo, Z. (2009). The research and efficient fpga implementation of ghash core for gmac. In *Proceedings of the 2009 International Conference on E-Business and Information System Security (EBISS 2009)*, pages 1–5. IEEE.

[Luykx et al., 2016] Luykx, A., Preneel, B., Tischhauser, E., and Yasuda, K. (2016). A mac mode for lightweight block ciphers. In Peyrin, T., editor, *Fast Software Encryption*, pages 43–59, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Michail et al., 2012] Michail, H. E., Athanasiou, G. S., Kelefouras, V., Theodoridis, G., and Goutis, C. E. (2012). On the exploitation of a high-throughput sha-256 fpga design for hmac. *ACM Transactions on Reconfigurable Technology and Systems*, 5(1):1–28.

[Morawiecki et al., 2014] Morawiecki, P., Gaj, K., Homsirikamol, E., Matusiewicz, K., Pieprzyk, J., Rogawski, M., Srebrny, M., and Wójcik, M. (2014). Icepole: High-speed, hardware-oriented authenticated encryption. In *Fast Software Encryption – FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 392–413. Springer.

[Nakano et al., 2026] Nakano, Y., Fukushima, K., and Isobe, T. (2026). Encryption algorithm Rocca-S. Internet-Draft draft-nakano-rocca-s-06, Internet Engineering Task Force. Work in Progress.

[Naya-Plasencia, 2017] Naya-Plasencia, M. (2017). Symmetric Cryptography for Long-Term Security. Habilitation à Diriger des Recherches thesis, Université Pierre et Marie Curie – Paris VI.

[Nikolić, 2014] Nikolić, I. (2014). Tiaoxin-346 Version 2.0. CAESAR Competition submission, Round 2. Available online.

[NIST, 2008] NIST (2008). FIPS PUB 198-1: The Keyed-Hash Message Authentication Code (HMAC). FIPS 198-1, NIST.

[Pirzada et al., 2019a] Pirzada, S. J. H., Murtaza, A., and Liu, J. (2019a). Implementation of cmac authentication algorithm on fpga for satellite communication. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 1358–1362.

[Pirzada et al., 2019b] Pirzada, S. J. H., Murtaza, A., Liu, J., and Xu, T. (2019b). The parallel cmac authentication algorithm. In *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, pages 800–804.

[Sakamoto et al., 2022] Sakamoto, K., Liu, F., Nakano, Y., Kiyomoto, S., and Isobe, T. (2022). Rocca: An efficient aes-based encryption scheme for beyond 5g (full version). *IACR Cryptol. ePrint Arch.*, page 116.

[Shen et al., 2019] Shen, C., Yan, X., Liu, Q., Kang, J., Liu, Y., Zheng, X., Zhang, X., and Liu, Y. (2019). An fpga design and implementation of 4g network security algorithm based on snow3g. In *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 387–392.

[Turan and Brandão, 2024] Turan, M. S. and Brandão, L. T. A. N. (2024). Keyed-Hash Message Authentication Code (HMAC): Specification of HMAC and Recommendations for Message Authentication. NIST Special Publication, Initial Public Draft 800-224, NIST.

[Vliegen et al., 2017] Vliegen, J., Reparaz, O., and Mentens, N. (2017). Maximizing the throughput of threshold-protected aes-gcm implementations on fpga. In *2017 IEEE 2nd International Verification and Security Workshop (IVSW)*, pages 140–145.

[Wang et al., 2025] Wang, D., Maximov, A., Ekdahl, P., and Johansson, T. (2025). A new stand-alone MAC construct called SMAC. *IACR Transactions on Symmetric Cryptology*, 2025(1):5–43.

[Wu and Preneel, 2014] Wu, H. and Preneel, B. (2014). Aegis: A fast authenticated encryption algorithm. In Lange, T., Lauter, K., and Lisoněk, P., editors, *Selected Areas in Cryptography – SAC 2013*, pages 185–201, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Yang et al., 2006] Yang, B., Karri, R., and Mcgrew, D. (2006). A high-speed hardware architecture for universal message authentication code. *IEEE Journal on Selected Areas in Communications*, 24(10):1831–1839.

[Zhou et al., 2007] Zhou, G., Michalik, H., and Hinsenkamp, L. (2007). Efficient and high-throughput implementations of aes-gcm on fpgas. *2007 International Conference on Field-Programmable Technology*, pages 185–192.