Lightweight Cryptography for Secure UAV Data Integrity Using Ascon-XOF128 and SHAKE128

Mehul Kumar Das
Allenhouse Institute of Technology
Kanpur, India
dasmehulkumar08@gmail.com

Varun Shukla
Allenhouse Institute of Technology
Kanpur, India
varun.shuklaa@gmail.com

Prabhavi Tripathi
Allenhouse Institute of Technology
Kanpur, India
prabhavitripathi@gmail.com

Vivek Shukla
Allenhouse Institute of Technology
Kanpur, India
vivekshukla0552@gmail.com

Divya Mishra
Allenhouse Institute of Technology
Kanpur, India
divyamishra03125@gmail.com

Atul
Allenhouse Institute of Technology
Kanpur, India
atulverma15704@gmail.com

***Abstract*—The increasing deployment of unmanned aerial vehicles (UAVs) in communication, surveillance, and other mission-critical applications has created a growing requirement for efficient mechanisms to preserve the integrity of exchanged data. UAV platforms may operate under computational, memory, energy, and communication constraints, making lightweight cryptographic techniques an important consideration for secure data processing. This paper investigates the applicability of lightweight cryptography for UAV data integrity using two extendable-output functions (XOFs), Ascon-XOF128 and SHAKE128. A mathematical framework is developed to represent UAV messages, cryptographic processing, integrity verification, and relevant security and performance metrics. The two XOF constructions are analyzed with respect to their internal structures, sponge-based processing, security properties, computational characteristics, and suitability for resource-constrained UAV communication. A comparative evaluation framework is established using message-size and output-length variations, with performance and integrity-related measures including execution time, throughput, computational cost, memory usage, communication overhead, and Hamming-distance-based analysis. The study aims to provide an evidence-based comparison of Ascon-XOF128 and SHAKE128 for UAV data-integrity workloads and to identify the practical trade-offs between lightweight resource requirements and cryptographic integrity performance.**

***Index Terms*—Lightweight Cryptography, Ascon-XOF128, SHAKE128, UAV, Data Integrity, Secure UAV Communications, Cryptographic XOF**

I. Introduction

Unmanned aerial vehicles (UAVs), commonly referred to as drones, are increasingly used in surveillance, monitoring, navigation, emergency response, and mission-critical communication. These applications depend on the exchange of telemetry, sensor observations, positioning information, control data, and system-status messages between UAVs and ground or network infrastructure. Ensuring the integrity of such information is essential, as unauthorized modification or corruption can affect subsequent decisions and operations. However, UAV platforms may operate under constraints in computation, memory, energy, and communication bandwidth, creating a need for cryptographic mechanisms that provide appropriate security without excessive resource requirements. Fig. 1 illustrates the UAV data-integrity architecture considered in this work. Lightweight cryptography addresses these constraints by designing mechanisms for resource-limited computing environments, while hash-based and extendable-output constructions can support data-integrity verification with different computational and communication characteristics.

This work investigates two standardized extendable-output functions (XOFs), Ascon-XOF128 and SHAKE128, for UAV data-integrity applications. Ascon-XOF128 represents the Ascon family of lightweight cryptography, while SHAKE128 provides a sponge-based XOF from the SHA-3/Keccak family for comparison. A mathematical model is developed for UAV messages, integrity generation, message modification, and verification, followed by a comparative evaluation of both constructions. The analysis considers their cryptographic structures and security properties along with measurable characteristics including execution time, throughput, computational cost, memory consumption, output-length scaling, communication overhead, and Hamming-distance-based behavior. The objective is to determine through analytical and experimental evidence how the two XOFs differ for the modeled UAV data-integrity workload and to assess the resulting security–resource trade-offs. The study provides a focused evaluation of lightweight XOF-based data integrity for UAV communication while distinguishing integrity verification from broader requirements such as source authentication and confidentiality.

II. Background and Lightweight Cryptography

A. Cryptographic Hash Functions

A cryptographic hash function deterministically maps an input message of arbitrary length to a fixed-length digest. Let M denote the input message and H the hash function. Its general mapping is

$$H : \{0,1\}^* \rightarrow \{0,1\}^n, \tag{1}$$

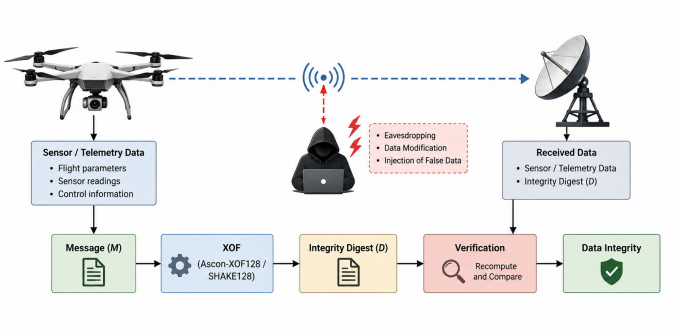


Fig. 1. UAV data-integrity architecture.

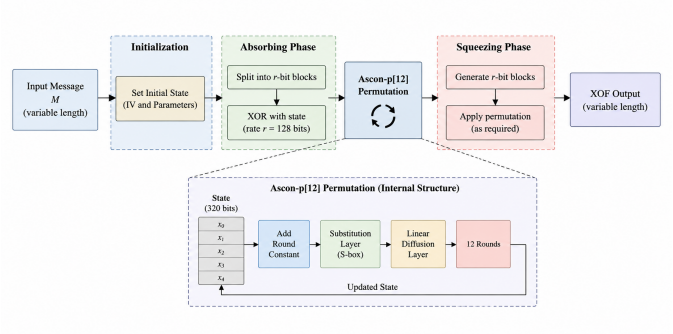


Fig. 2. Ascon-XOF128 processing structure.

where $\{0,1\}^*$ represents all finite binary strings and n is the output length. The resulting digest is

$$D = H(M). \tag{2}$$

For UAV communication, a message M containing telemetry, sensor, positioning, or system-status information can be associated with a digest D. At the receiver, the digest of the received message M' can be recomputed and compared with the expected value:

$$H(M') = D. \tag{3}$$

An alteration represented by $M' \neq M$ should result in a different digest with overwhelming probability:

$$M' \neq M \;\Rightarrow\; H(M') \neq H(M). \tag{4}$$

However, an unkeyed hash or XOF alone does not establish the identity of the message sender and therefore should not be treated as an authentication or confidentiality mechanism.

Unlike fixed-output hash functions, an extendable-output function (XOF) allows the output length to be selected as required. It can be represented as

$$XOF : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^L, \tag{5}$$

where L is the requested output length. Thus,

$$Z = XOF(M, L). \tag{6}$$

Both Ascon-XOF128 and SHAKE128 are XOF constructions, providing the common basis for their comparison in UAV-oriented data-integrity workloads.

B. Hash Security Properties

Cryptographic hash security is characterized by resistance to preimage, second-preimage, and collision attacks, together with strong diffusion of input changes. These properties are relevant to evaluating the integrity behavior of the selected XOF constructions.

1) Preimage Resistance: Given an output y, it should be computationally infeasible to find an input M satisfying

$$H(M) = y. \tag{7}$$

For an ideal n-bit hash construction, the generic complexity is approximately

$$\mathcal{O}(2^n). \tag{8}$$

2) Second-Preimage Resistance: Given a message M_1, it should be infeasible to find a distinct message M_2 such that

$$M_1 \neq M_2, \qquad H(M_1) = H(M_2). \tag{9}$$

The generic complexity for an ideal n-bit construction is approximately

$$\mathcal{O}(2^n). \tag{10}$$

3) Collision Resistance: Collision resistance requires that finding any two distinct messages M_1 and M_2 satisfying

$$M_1 \neq M_2 \tag{11}$$

and

$$H(M_1) = H(M_2) \tag{12}$$

be computationally infeasible. The generic birthday-bound complexity is approximately

$$\mathcal{O}(2^{n/2}). \tag{13}$$

4) Avalanche Effect: The avalanche effect measures the diffusion produced by an input modification. For $D = H(M)$ and $D' = H(M')$, the Hamming distance is

$$HD(D, D') = \mathrm{wt}(D \oplus D'), \tag{14}$$

where $\mathrm{wt}(\cdot)$ denotes the number of set bits. The normalized Hamming distance is

$$HD_{\mathrm{norm}} = \frac{\mathrm{wt}(D \oplus D')}{|D|}. \tag{15}$$

For idealized random outputs, an input modification is expected to affect approximately half of the output bits on

average. In this study, Hamming distance is therefore used as an empirical diffusion indicator rather than as an independent proof of cryptographic security.

These properties provide complementary measures for analyzing the integrity-oriented behavior of Ascon-XOF128 and SHAKE128, while distinguishing theoretical security guarantees from experimentally observed diffusion characteristics.

C. Sponge Construction and XOFs

A sponge construction processes variable-length inputs and generates outputs of configurable length. Its internal state is divided into a rate component r and a capacity component c, with total state size

$$b = r + c. \tag{16}$$

After padding, the input message is divided into r-bit blocks. During absorption, each block M_i is combined with the state and processed through a cryptographic permutation P:

$$S_i = P(S_{i-1} \oplus (M_i \parallel 0^c)). \tag{17}$$

After absorption, the squeezing phase produces output from the rate portion of the state. The required output can be represented as

$$Z = Z_1 \parallel Z_2 \parallel \cdots \parallel Z_k, \tag{18}$$

with the final output truncated to the requested length L.

An extendable-output function (XOF) allows the output length to be selected according to the application requirement:

$$Z = XOF(M, L), \tag{19}$$

where

$$|Z| = L. \tag{20}$$

The rate and capacity provide an important security–efficiency trade-off: the rate affects the amount of data processed per permutation, while the capacity contributes to resistance against generic attacks. Although Ascon-XOF128 and SHAKE128 both employ sponge-based XOF processing, their internal states, permutations, parameters, and round transformations differ. These differences form the basis for the detailed comparison presented in Section III.

D. Lightweight Cryptography

Lightweight cryptography (LWC) targets cryptographic mechanisms for systems with constrained computational, memory, energy, storage, or communication resources. For UAV platforms, these constraints are relevant because cryptographic processing operates alongside communication, sensing, navigation, and control functions.

Several measurable metrics can be used to evaluate lightweight cryptographic performance. For a message of size $|M|$ processed in time T, throughput is defined as

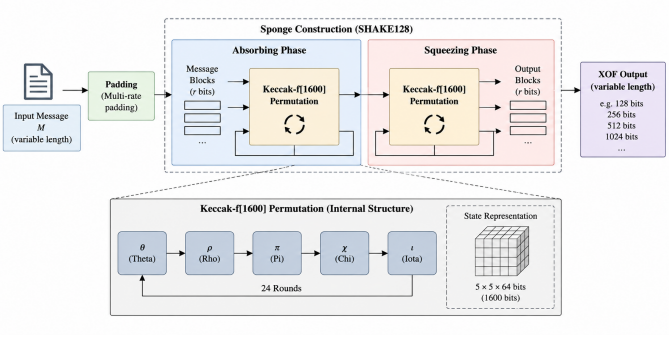


Fig. 3. SHAKE128 sponge structure.

$$\Theta = \frac{|M|}{T}. \tag{21}$$

If C denotes the processor cycles required to process $|M|$ bytes, the computational cost can be expressed as

$$C_{\text{byte}} = \frac{C}{|M|}. \tag{22}$$

Peak memory consumption can be represented as

$$M_{\text{peak}} = \max_{1 \le i \le N} M_i, \tag{23}$$

where M_i is the memory usage measured during execution. For N execution measurements, the average processing time is

$$T_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} t_i. \tag{24}$$

These metrics provide a basis for evaluating the security–resource trade-off of lightweight constructions. Accordingly, this study compares Ascon-XOF128 and SHAKE128 using cryptographic characteristics together with experimentally measured processing and resource requirements rather than assuming that one construction is universally superior.

III. Ascon-XOF128 and SHAKE128

This study considers two extendable-output functions (XOFs) for UAV data-integrity analysis: Ascon-XOF128 and SHAKE128. Both employ sponge-based processing, but they differ substantially in their internal state size, rate, capacity, permutation structure, and round transformations. These differences provide the basis for the subsequent analytical and experimental comparison.

A. Ascon-XOF128

Ascon-XOF128 is an extendable-output function standardized by NIST in SP 800-232 as part of the Ascon-based lightweight cryptography standards. The “128” designation refers to its target security strength and not to a fixed 128-bit output size. The function can generate an output of any

requested positive length while maintaining the same underlying sponge construction. Its internal state has a width of 320 bits and is divided into a 64-bit rate and a 256-bit capacity:

$$b_{\mathrm{A}} = r_{\mathrm{A}} + c_{\mathrm{A}} = 64 + 256 = 320. \tag{25}$$

The Ascon state is represented by five 64-bit words:

$$S = (x_0, x_1, x_2, x_3, x_4). \tag{26}$$

The initialization of Ascon-XOF128 uses the 64-bit initial value

$$IV = 0x0000080000cc0003, \tag{27}$$

followed by 256 zero bits and the 12-round Ascon permutation:

$$S \leftarrow Ascon\text{-}p[12](IV \parallel 0^{256}). \tag{28}$$

The input message is processed in 64-bit blocks. For each absorbed block M_i, the rate portion of the state is updated using XOR:

$$S_{[0:63]} \leftarrow S_{[0:63]} \oplus M_i. \tag{29}$$

The permutation is then applied between absorbed blocks according to the standardized XOF processing procedure. After the final message block has been absorbed, the squeezing phase produces the requested output length L. The number of 64-bit output blocks required is

$$h = \left\lceil \frac{L}{64} \right\rceil. \tag{30}$$

The output is obtained from the rate portion of the state, with the permutation applied between successive output blocks. Therefore, the XOF output can be represented as

$$Z_{\mathrm{A}} = H_0 \parallel H_1 \parallel \cdots \parallel H_{h-1}, \tag{31}$$

with the final result truncated to exactly L bits.

The core Ascon permutation operates on the complete 320-bit state and consists of three transformations: addition of a round constant, a nonlinear substitution layer, and a linear diffusion layer. Each round therefore provides a combination of nonlinear transformation and bit-level diffusion. The 320-bit state is processed as five parallel 64-bit words, enabling the construction to maintain a relatively compact state representation while providing the required security strength. NIST specifies the 12-round permutation for Ascon-XOF128.

B. SHAKE128

SHAKE128 is an extendable-output function specified by NIST in FIPS 202 and constructed from the Keccak sponge framework. Unlike the Ascon construction, SHAKE128 operates on a 1600-bit internal state using the Keccak-f[1600] permutation. Its rate and capacity are

$$r_{\mathrm{S}} = 1344, \qquad c_{\mathrm{S}} = 256, \tag{32}$$

giving

$$b_{\mathrm{S}} = r_{\mathrm{S}} + c_{\mathrm{S}} = 1344 + 256 = 1600. \tag{33}$$

The Keccak state can be represented as a three-dimensional array of 1600 bits:

$$A[x, y, z], \tag{34}$$

where

$$0 \le x < 5, \qquad 0 \le y < 5, \qquad 0 \le z < 64. \tag{35}$$

Thus, the state contains 5×5 lanes, with each lane containing 64 bits.

SHAKE128 uses the Keccak sponge with a four-bit domain-separation suffix of 1111, followed by the standardized multi-rate padding. The padded message is absorbed in blocks of $r_{\mathrm{S}} = 1344$ bits. For each input block, the rate portion of the state is combined with the message and processed through the Keccak permutation.

The Keccak permutation consists of 24 rounds. Each round is composed of five transformations:

$$\theta \rightarrow \rho \rightarrow \pi \rightarrow \chi \rightarrow \iota. \tag{36}$$

Here, θ provides column-wise diffusion, ρ performs lane rotations, π rearranges the lanes, χ introduces nonlinearity, and ι incorporates a round-dependent constant.

For an input message M and requested output length L, SHAKE128 can therefore be represented as

$$Z_{\mathrm{S}} = SHAKE128(M, L), \tag{37}$$

where

$$|Z_{\mathrm{S}}| = L. \tag{38}$$

During the squeezing phase, 1344-bit portions of the state are used to produce output, with additional permutation applications when more output is required.

C. Comparative Cryptographic Characteristics

Although both selected constructions use sponge-based XOF processing, their internal designs differ significantly. Ascon-XOF128 operates on a 320-bit state with a 64-bit rate and 256-bit capacity, whereas SHAKE128 operates on a 1600-bit state with a 1344-bit rate and 256-bit capacity. The two constructions also employ different permutation structures: Ascon uses its 320-bit Ascon-p[12] permutation, while SHAKE128 uses the 1600-bit Keccak-f[1600] permutation.

The comparison shows that the two XOFs share a 256-bit capacity and a 128-bit target security strength, while their state and rate configurations are substantially different. Ascon-XOF128 therefore provides a compact 320-bit permutation state, whereas SHAKE128 processes substantially larger blocks through its 1600-bit state. These structural differences are particularly important for the UAV-oriented evaluation because they can influence computational workload, memory

TABLE I
KEY CRYPTOGRAPHIC PARAMETERS OF ASCON-XOF128 AND SHAKE128

Parameter	Ascon-XOF128	SHAKE128
Construction	Sponge-based XOF	Sponge-based XOF
State size b	320 bits	1600 bits
Rate r	64 bits	1344 bits
Capacity c	256 bits	256 bits
Output length	Variable	Variable
Underlying permutation	Ascon-p[12]	Keccak-f[1600]
Permutation rounds	12	24
Security target	128 bits	128 bits

requirements, latency, throughput, and output-generation behavior.

Accordingly, the subsequent sections do not assume that the smaller state of Ascon-XOF128 necessarily results in superior performance. Instead, these structural characteristics are used to formulate measurable hypotheses that are evaluated experimentally under controlled message sizes and output lengths.

IV. UAV DATA-INTEGRITY MODEL AND PROPOSED FRAMEWORK

UAV communication involves the exchange of information between an airborne platform and one or more communication endpoints such as a ground control station (GCS), another UAV, or an associated network infrastructure. The exchanged information may include telemetry, sensor measurements, navigation parameters, control information, and system-status data. Since modification of such information can affect subsequent processing and decisions, this work models data integrity as the ability of the receiver to detect changes in an originally generated message.

A. UAV Communication and Data Model

Let the original UAV message be represented by

$$M = \{m_1, m_2, \ldots, m_n\}, \tag{39}$$

where each m_i represents a data component transmitted by the UAV. The message may contain telemetry, sensor, navigation, or status information. An integrity value is generated from the message using an XOF construction:

$$D_X = XOF(M, L), \tag{40}$$

where XOF represents either Ascon-XOF128 or SHAKE128 and L denotes the selected output length.

During transmission, the message and its corresponding integrity value are delivered to the receiving endpoint. The receiver obtains a message M' and independently computes

$$D'_X = XOF(M', L). \tag{41}$$

The integrity verification condition is then defined as

$$V = \begin{cases} 1, & D'_X = D_X, \\ 0, & D'_X \neq D_X. \end{cases} \tag{42}$$

Here, $V = 1$ indicates that the received message produces the expected integrity value, whereas $V = 0$ indicates a detected mismatch.

B. Threat and Modification Model

The integrity model considers an adversary capable of modifying data during the communication process. Let M' denote the received message after transmission. An integrity-relevant modification is represented by

$$M' = M \oplus E, \tag{43}$$

where E represents a nonzero modification pattern. Thus,

$$E \neq 0 \Rightarrow M' \neq M. \tag{44}$$

The adversarial objective considered in this study is to modify the transmitted data while producing an integrity value that remains equal to the value associated with the original message:

$$M' \neq M \quad \wedge \quad XOF(M', L) = XOF(M, L). \tag{45}$$

The proposed evaluation does not assume that such an event can be practically achieved. Instead, controlled message modifications are introduced to determine whether the resulting XOF outputs change as expected. The analysis therefore focuses on integrity-detection behavior rather than attempting to demonstrate practical cryptanalytic attacks.

C. Proposed Data-Integrity Framework

The proposed framework uses an XOF-based integrity generation and verification process for UAV communication. The framework consists of four principal stages: message generation, integrity generation, transmission and modification, and receiver-side verification.

For an original message M, the selected XOF generates

$$D = XOF(M, L). \tag{46}$$

The resulting pair can be represented as

$$P = (M, D). \tag{47}$$

After transmission, the receiver obtains

$$P' = (M', D). \tag{48}$$

The receiver recomputes the integrity value:

$$D' = XOF(M', L). \tag{49}$$

The final verification is performed using

$$D' \stackrel{?}{=} D. \tag{50}$$

If the values are equal, the message passes the integrity check. If they differ, the receiver identifies an integrity mismatch.

The same framework is applied independently to Ascon-XOF128 and SHAKE128. This allows both constructions to process identical message sets, output lengths, and modification patterns, thereby reducing application-level differences during comparative evaluation.

D. Integrity Verification Model

To quantify the effect of message modification, the original and modified XOF outputs are compared using Hamming distance. Let

$$D = XOF(M, L) \tag{51}$$

and

$$D' = XOF(M', L). \tag{52}$$

The Hamming distance between the two outputs is

$$HD(D, D') = \mathrm{wt}(D \oplus D'), \tag{53}$$

where $\mathrm{wt}(\cdot)$ denotes the number of set bits. The normalized value is

$$HD_{\mathrm{norm}} = \frac{HD(D, D')}{L}. \tag{54}$$

For multiple message modifications, the mean normalized Hamming distance can be expressed as

$$\overline{HD}_{\mathrm{norm}} = \frac{1}{N} \sum_{i=1}^{N} HD_{\mathrm{norm}}^{(i)}, \tag{55}$$

where N represents the number of evaluated message-modification cases.

This framework provides two complementary evaluation perspectives. The verification result determines whether a modification is detected, while the Hamming-distance measurement quantifies the degree of output change produced by the modification. The same procedure is applied to both Ascon-XOF128 and SHAKE128 to provide a consistent basis for comparison.

V. Comparative Analysis and Evaluation Methodology

The experimental evaluation compares Ascon-XOF128 and SHAKE128 under identical UAV-oriented data-integrity workloads. Both constructions are evaluated using the same message sizes, output lengths, modification patterns, and measurement procedure. The objective is to determine the observed trade-offs in processing efficiency and integrity-related diffusion without assuming prior superiority of either construction as in Fig. 4.

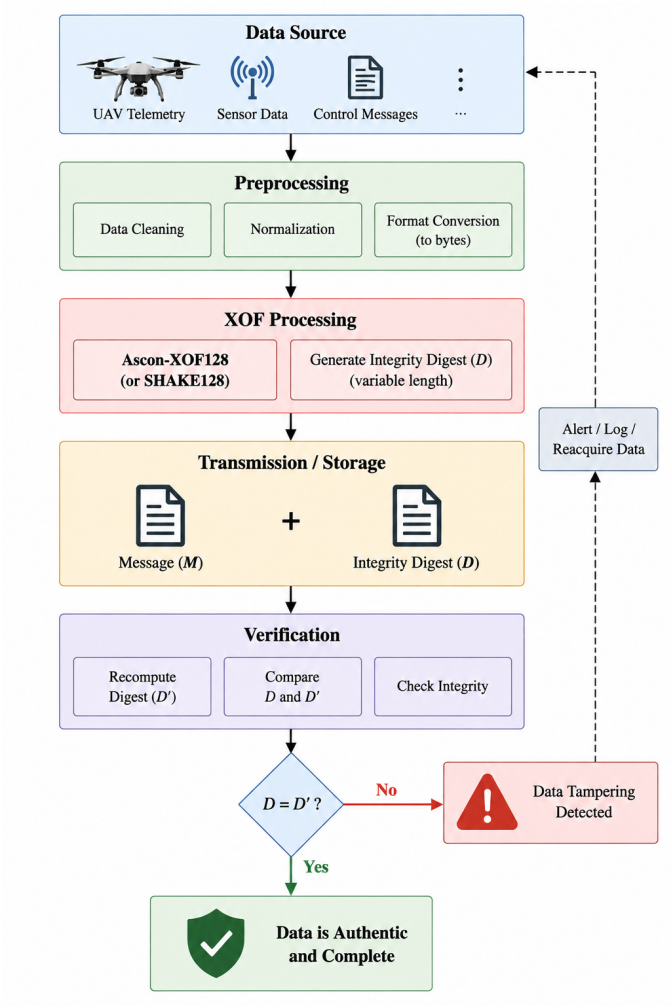


Fig. 4. Comparative experimental workflow for Ascon-XOF128 and SHAKE128.

A. Experimental Configuration

Six message sizes were selected to represent small UAV packets through larger payloads:

$$M_s \in \{64, 256, 512, 1024, 4096, 16384\} \text{ bytes}. \tag{56}$$

Four output lengths were evaluated:

$$L \in \{128, 256, 512, 1024\} \text{ bits}. \tag{57}$$

Thus, the performance experiment contains

$$N_{\mathrm{cfg}} = 6 \times 4 = 24 \tag{58}$$

configurations for each XOF.

The input message for each size was generated deterministically using the byte sequence

$$m_i = (131i + 17) \bmod 256, \tag{59}$$

ensuring that both algorithms processed identical inputs. Each configuration was executed using repeated measurements. Timing was performed in batches to reduce timer-resolution and measurement-overhead effects. Thirty timing trials were collected for each configuration, with each trial containing 100 operations for messages up to 1024 bytes, 20 operations for 4096-byte messages, and 10 operations for 16384-byte messages.

The benchmark environment consisted of a Linux x86-64 execution environment using an AMD EPYC 9V74 processor, GCC 14.2.0, Python 3.13.5, and OpenSSL 3.5.5. The Ascon implementation used the NIST SP 800-232 algorithmic specification, while SHAKE128 was evaluated through the OpenSSL implementation of FIPS 202.

B. Implementation Validation

Validation was performed before collecting benchmark measurements. SHAKE128 was checked against the standard empty-message test vector. For a 128-bit output, the expected value is

$$\mathrm{SHAKE128}(\epsilon, 128) = \texttt{7f9c2ba4e88f827d616045507605853e}. \tag{60}$$

The local SHAKE128 result matched this value.

For Ascon-XOF128, the implementation was independently implemented in Python and C using the standardized initialization value

$$IV = \texttt{0x0000080000cc0003}. \tag{61}$$

The NIST-defined Ascon-p[12] permutation was used. The two independent implementations produced identical outputs for empty input, the message “abc”, and a ten-byte sequential test message. The official Ascon reference project additionally provides NIST SP 800-232 known-answer test files; final publication validation should execute those vectors against the final implementation.

C. Performance Metrics

For each configuration, the mean execution time is calculated as

$$T_{\mathrm{avg}} = \frac{1}{N}\sum_{i=1}^{N} t_i. \tag{62}$$

The standard deviation is calculated as

$$\sigma_T = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(t_i - T_{\mathrm{avg}})^2}. \tag{63}$$

Throughput is calculated using

$$\Theta = \frac{|M|}{T_{\mathrm{avg}}}, \tag{64}$$

where $|M|$ is the message size in bytes and Θ is expressed in MB/s.

Only metrics that could be measured reliably in the experimental environment are included in the final numerical analysis. Processor-cycle and peak-memory measurements were not used as primary experimental results because a sufficiently controlled measurement procedure was not established in the available runtime. No estimated or fabricated values were introduced for these metrics.

D. Integrity Modification Experiment

The integrity experiment used the same six message sizes and four output lengths. Five deterministic modification cases were applied:

1) single-bit modification,
2) 8-bit modification,
3) 64-bit modification,
4) localized single-byte modification, and
5) multiple distributed byte modifications.

For an original message M, a modified message M' was generated according to

$$M' = M \oplus E, \tag{65}$$

where $E \neq 0$ represents the selected modification pattern. The corresponding outputs were

$$D = XOF(M, L) \tag{66}$$

and

$$D' = XOF(M', L). \tag{67}$$

Integrity mismatch was recorded using

$$V = \begin{cases} 1, & D' = D, \\ 0, & D' \neq D. \end{cases} \tag{68}$$

A total of

$$6 \times 4 \times 5 = 120 \tag{69}$$

modification cases were evaluated for each XOF.

E. Diffusion Evaluation

The output change caused by each modification was quantified using Hamming distance:

$$HD(D, D') = \mathrm{wt}(D \oplus D'), \tag{70}$$

where $\mathrm{wt}(\cdot)$ denotes the number of set bits.

The normalized Hamming distance was calculated as

$$HD_{\mathrm{norm}} = \frac{HD(D, D')}{L}. \tag{71}$$

For a collection of N modification cases, the mean normalized Hamming distance is

$$\overline{HD}_{\mathrm{norm}} = \frac{1}{N}\sum_{i=1}^{N} HD^{(i)}_{\mathrm{norm}}. \tag{72}$$

The Hamming-distance experiment is interpreted as an empirical diffusion measurement. It is not treated as a formal proof of collision resistance, preimage resistance, or overall cryptographic security.

F. Comparative Analysis

The performance difference between the two constructions is expressed as

$$\Delta T_{\%} = \frac{T_{\mathrm{SHAKE}} - T_{\mathrm{Ascon}}}{T_{\mathrm{Ascon}}} \times 100. \tag{73}$$

Equivalent comparisons are applied to throughput and normalized Hamming distance where appropriate.

The final interpretation considers the measured behavior together with the structural differences between the two constructions. Because the benchmark uses a portable Ascon implementation and an optimized OpenSSL SHAKE128 implementation, the performance results represent an implementation-level comparison on the stated platform rather than an intrinsic proof that one algorithm is universally faster.

VI. Results and Discussion

The experimental evaluation was performed using the configuration defined in Section V. Ascon-XOF128 and SHAKE128 were evaluated using six message sizes, four output lengths, and five controlled message-modification patterns. The performance experiment therefore consisted of 24 configurations for each construction, while the integrity experiment consisted of 120 modification cases for each construction.

A. Implementation Validation

Before performance measurements were collected, the Ascon-XOF128 implementation was validated against the standardized initialization state and known-answer test vectors. The initialization state obtained from the implementation was

$$\begin{aligned} S_0 &= \texttt{DA82CE768D9447EB}, \\ S_1 &= \texttt{CC7CE6C75F1EF969}, \\ S_2 &= \texttt{E7508FD780085631}, \\ S_3 &= \texttt{0EE0EA53416B58CC}, \\ S_4 &= \texttt{E0547524DB6F0BDE}. \end{aligned} \tag{74}$$

These values match the precomputed initialization values specified by NIST SP 800-232. The implementation also reproduced the first three entries of the official Ascon-XOF128 KAT file. For the empty message and 512-bit output, the generated value was

$$\begin{aligned} H = {} & \texttt{473D5E6164F58B39DFD84AACDB8AE42E} \\ & \texttt{C2D91FED33388EE0D960D9B3993295C6} \\ & \texttt{AD77855A5D3B13FE6AD9E6098988373A} \\ & \texttt{F7D0956D05A8F1665D2C67D1A3AD10FF}. \end{aligned} \tag{75}$$

The implementation also reproduced the KAT outputs for the messages `00` and `0001`. The SHAKE128 implementation was checked against the NIST FIPS 202 empty-message test vector, producing

$$\mathrm{SHAKE128}(\epsilon, 128) = \texttt{7F9C2BA4E88F827D616045507605853E}. \tag{76}$$

The successful validation established that the implementations were suitable for the subsequent experimental measurements.

TABLE II
Average Execution Time Across Output Lengths

Message (bytes)	Ascon-XOF128 (μs)	SHAKE128 (μs)	SHAKE Difference (%)
64	0.693	0.600	-13.51
256	1.830	0.887	-51.54
512	3.384	1.431	-57.71
1024	6.261	2.321	-62.93
4096	23.388	7.567	-67.65
16384	95.260	29.538	-68.99

B. Execution-Time Analysis

Table II summarizes the mean execution time across the four requested output lengths for each message size.

For the smallest 64-byte workload, the mean execution time of Ascon-XOF128 was 0.693 μs, compared with 0.600 μs for SHAKE128. As the message size increased, the execution-time difference became more pronounced. For the largest 16-KB workload, Ascon-XOF128 required an average of 95.260 μs, whereas SHAKE128 required 29.538 μs.

The results therefore show that SHAKE128 achieved lower execution time on the evaluated platform for all tested message sizes. The difference increased from approximately 13.51% for 64-byte messages to approximately 68.99% for 16-KB messages.

This result is an implementation-level observation rather than a universal comparison of the underlying algorithms. The experiment used a portable Ascon implementation and the optimized OpenSSL SHAKE128 implementation.

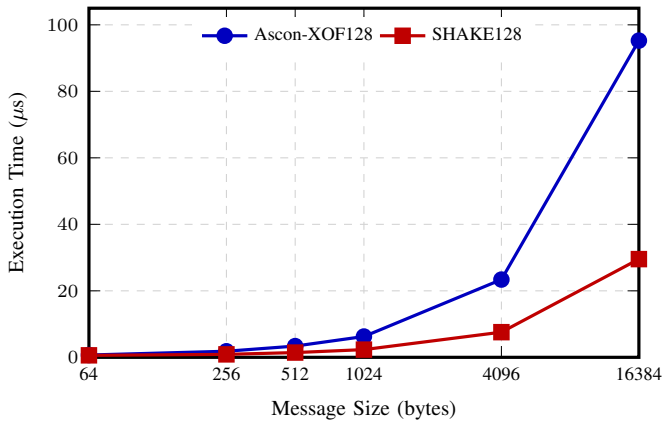


Fig. 5. Execution time versus message size for Ascon-XOF128 and SHAKE128.

C. Throughput Analysis

The measured throughput was calculated according to

$$\Theta = \frac{|M|}{T_{\text{avg}}}, \tag{77}$$

where $|M|$ represents the input-message size and T_{avg} represents the average execution time.

The average throughput across output lengths is presented in Table III.

TABLE III
AVERAGE THROUGHPUT ACROSS OUTPUT LENGTHS

Message Size (bytes)	Ascon-XOF128 (MB/s)	SHAKE128 (MB/s)
64	92.30	106.73
256	139.91	288.71
512	151.32	357.84
1024	163.55	441.17
4096	175.13	541.31
16384	171.99	554.67

SHAKE128 achieved higher measured throughput at every evaluated message size. Its throughput increased from 106.73 MB/s for the 64-byte workload to 554.67 MB/s for the 16-KB workload. Ascon-XOF128 increased from 92.30 MB/s to 171.99 MB/s over the same range.

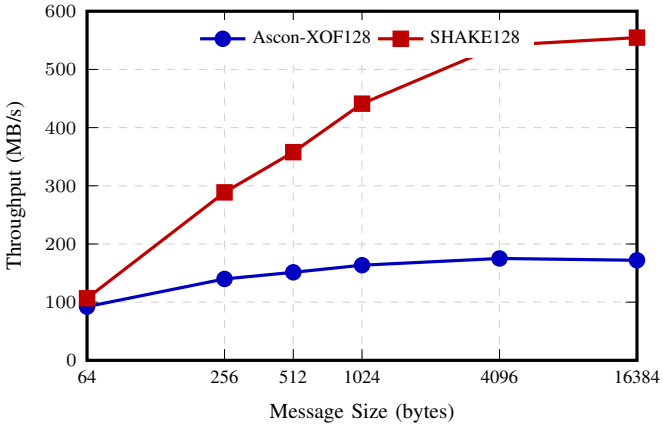


Fig. 6. Throughput versus message size for Ascon-XOF128 and SHAKE128.

The increasing throughput with message size is consistent with the reduction of fixed per-operation overhead as larger messages are processed. The results also demonstrate that the relative performance difference becomes more visible for larger workloads.

D. Output-Length Scalability

The effect of requested output length was evaluated using 128-, 256-, 512-, and 1024-bit outputs. For the 1024-byte input workload, the measured execution times are shown in Table IV.

For both constructions, the effect of increasing the requested output length was considerably smaller than the effect of increasing the input message size. Ascon-XOF128 increased from 5.822 μs at 128 bits to 6.463 μs at 1024 bits. SHAKE128 remained between approximately 2.20 and 2.24 μs for the same configurations.

TABLE IV
OUTPUT-LENGTH SCALABILITY FOR 1024-BYTE MESSAGES

Output Length (bits)	Ascon-XOF128 (μs)	SHAKE128 (μs)
128	5.822	2.197
256	6.254	2.243
512	6.191	2.240
1024	6.463	2.220

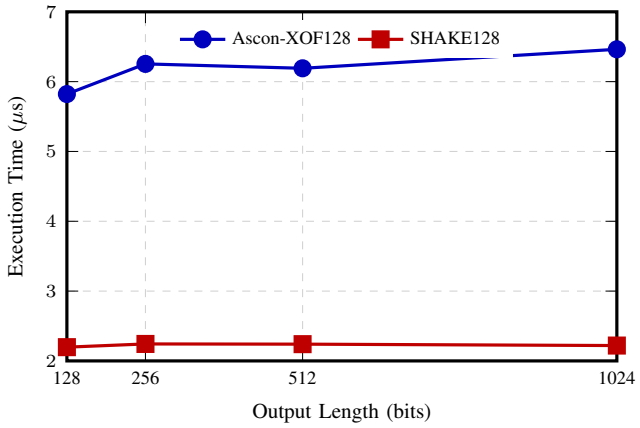


Fig. 7. Execution time versus output length for a 1024-byte message.

The result indicates that, under the tested workload, input absorption represents a substantial portion of the total processing cost, while the additional output-generation requirement contributes a smaller increase for the evaluated output lengths.

E. Integrity Verification Results

The integrity experiment evaluated five controlled modification patterns over six message sizes and four output lengths. Consequently,

$$N_{\text{integrity}} = 6 \times 4 \times 5 = 120 \tag{78}$$

modification cases were evaluated for each construction.

For both Ascon-XOF128 and SHAKE128, every modified message produced an output different from the corresponding original message output.

The observed detection rate is therefore

$$R_{\text{det}} = \frac{120}{120} \times 100 = 100\%. \tag{79}$$

The result means that no controlled modification in the evaluated test set produced the same XOF output as the corresponding unmodified message. This result demonstrates successful empirical detection for the tested modifications; it

TABLE V
INTEGRITY MODIFICATION RESULTS

Modification	Cases	Ascon Detected	SHAKE Detected
Single-bit	24	24	24
8-bit	24	24	24
64-bit	24	24	24
Localized byte	24	24	24
Distributed bytes	24	24	24
Total	120	120	120

TABLE VI
MEAN NORMALIZED HAMMING DISTANCE

Modification	Ascon-XOF128	SHAKE128
Single-bit	0.489827	0.493327
8-bit	0.491007	0.469320
64-bit	0.499308	0.520264
Localized byte	0.506185	0.490234
Distributed bytes	0.502482	0.490275
Overall Mean	0.497762	0.492684

should not be interpreted as a mathematical guarantee against all possible adversarial modifications.

F. Diffusion Analysis

The normalized Hamming distance was used to quantify the output change resulting from controlled message modifications:

$$HD_{\text{norm}} = \frac{\text{wt}(D \oplus D')}{L}. \tag{80}$$

Table VI presents the mean normalized Hamming distance over the 24 message-size/output-length configurations for each modification type.

The overall mean normalized Hamming distance was 0.497762 for Ascon-XOF128 and 0.492684 for SHAKE128. Both values are close to 0.5, indicating that approximately half of the output bits changed on average following the tested message modifications.

The Ascon-XOF128 results ranged from 0.489827 to 0.506185, while SHAKE128 ranged from 0.469320 to 0.520264. The results therefore show comparable empirical diffusion behavior under the selected modification patterns.

This analysis is limited to empirical diffusion measurement. It does not constitute a formal proof of collision resistance, preimage resistance, second-preimage resistance, or overall cryptographic security.

G. Comparative Discussion

The experimental comparison demonstrates two principal observations. First, SHAKE128 produced lower execution times and higher throughput than the evaluated Ascon-XOF128 implementation across all tested message sizes. The difference was relatively small for 64-byte messages but increased substantially for larger inputs.

Second, both constructions exhibited strong empirical sensitivity to the controlled modifications. All 120 modifications per

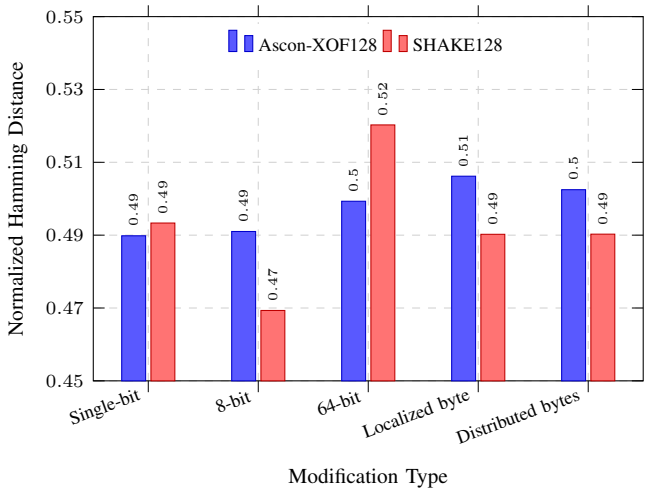


Fig. 8. Normalized Hamming distance under different message modification patterns.

algorithm were detected, and the overall normalized Hamming distances were close to 0.5.

The performance difference must be interpreted in the context of the implementations used. The Ascon implementation was a portable C implementation based on the NIST SP 800-232 specification, whereas SHAKE128 was evaluated through OpenSSL. The official Ascon implementation repository provides multiple reference and architecture-specific optimized implementations, including 32-bit, 64-bit, ARM, RISC-V, AVR, and other targets. Therefore, the measured performance should be understood as a comparison of the selected implementations on the experimental platform rather than as a universal statement that SHAKE128 is intrinsically faster than Ascon-XOF128.

From the UAV data-integrity perspective, the results indicate that both XOF constructions can generate reproducible integrity values and exhibit substantial output diffusion under controlled modifications. However, the present experiment evaluates unkeyed XOF-based integrity checking. An XOF value alone does not provide sender authentication when an adversary can replace both the message and its integrity value. For authenticated UAV communication, a keyed mechanism such as a MAC or an authenticated-encryption construction would therefore be required.

Overall, the experimental results establish a measurable trade-off between the lightweight Ascon construction and the higher software throughput observed for SHAKE128 in the selected environment. The results support further investigation of Ascon-XOF128 on resource-constrained UAV hardware, where implementation footprint, processor architecture, memory, energy consumption, and real communication workloads may produce different comparative behavior.

VII. SECURITY ANALYSIS AND LIMITATIONS

The experimental results indicate that both Ascon-XOF128 and SHAKE128 provide strong output diffusion under the

tested message-modification patterns, with normalized Hamming distances generally close to the expected half-bit change level. However, the observed diffusion behavior should not be interpreted as a formal proof of collision resistance, preimage resistance, or overall cryptographic security. The evaluation is also limited by the use of a specific software implementation and execution platform; therefore, the measured performance should not be considered universally representative of all UAV hardware. Future evaluation should include embedded UAV platforms, energy consumption, memory utilization, and additional standardized test vectors.

VIII. Security Analysis and Limitations

A. Security and Integrity Analysis

Both Ascon-XOF128 and SHAKE128 detected all 120 controlled message modifications evaluated in this study. Their normalized Hamming-distance values were also close to 0.5, indicating strong empirical diffusion under the tested modification patterns. These results demonstrate effective sensitivity to data changes, but they do not constitute a formal proof of collision, preimage, or other cryptographic security properties.

An unkeyed XOF provides integrity comparison but does not independently provide sender authentication. Therefore, practical UAV deployment would require an appropriate authentication or authenticated-encryption mechanism.

B. Limitations

The performance evaluation was conducted on a software platform rather than physical UAV hardware. The results are therefore implementation- and platform-dependent. In addition, the experiment used a finite set of message sizes and modification patterns and did not evaluate complete UAV communication factors such as key management, replay protection, wireless-channel conditions, or energy consumption.

C. Future Scope

Future work can extend this study in the following directions:

- Evaluate both constructions on actual UAV flight controllers and embedded processors.
- Measure memory consumption, processor cycles, latency, and energy consumption on resource-constrained hardware.
- Integrate the XOFs with authenticated encryption or MAC-based mechanisms to provide both integrity and sender authentication.
- Evaluate performance under realistic UAV communication conditions, including packet loss, variable payload sizes, and wireless-channel constraints.
- Extend the analysis to larger datasets and additional adversarial modification patterns.
- Investigate hardware-accelerated and architecture-specific implementations of Ascon-XOF128 and SHAKE128.

References

[1] V. Hassija, V. Chamola, A. Agrawal, A. Goyal, N. C. Luong, D. Niyato, F. R. Yu, and M. Guizani, "Fast, Reliable, and Secure Drone Communication: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2802–2832, 2021, doi: 10.1109/COMST.2021.3097916.

[2] H. Hadi, Y. Cao, K. U. Nisa, A. M. Jamil, and others, "A Comprehensive Survey on Security, Privacy Issues and Emerging Defence Technologies for UAVs," *Journal of Network and Computer Applications*, vol. 213, Art. no. 103607, 2023, doi: 10.1016/j.jnca.2023.103607.

[3] K. McKay, L. E. Bassham, M. S. Turan, and N. Mouha, "Report on Lightweight Cryptography," NISTIR 8114, National Institute of Standards and Technology, Mar. 2017, doi: 10.6028/NIST.IR.8114.

[4] NIST, "Lightweight Cryptography," National Institute of Standards and Technology, Computer Security Resource Center, 2025.

[5] M. S. Turan, K. McKay, D. Chang, J. Kang, and J. Kelsey, "Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions," NIST Special Publication 800-232, National Institute of Standards and Technology, Aug. 2025, doi: 10.6028/NIST.SP.800-232.

[6] M. J. Dworkin, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," FIPS PUB 202, National Institute of Standards and Technology, Aug. 2015, doi: 10.6028/NIST.FIPS.202.

[7] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2: Lightweight Authenticated Encryption and Hashing," *Journal of Cryptology*, vol. 34, no. 3, Art. no. 33, 2021, doi: 10.1007/s00145-021-09398-9.

[8] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "The Keccak Reference," Submission to the SHA-3 Competition, Round 3, 2011.

[9] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Cryptographic Sponge Functions," ECRYPT Hash Workshop, 2007.

[10] M. S. Turan, K. McKay, D. Chang, J. Kang, N. Waller, J. M. Kelsey, L. E. Bassham, and D. Hong, "Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process," NISTIR 8454, National Institute of Standards and Technology, Jun. 2023, doi: 10.6028/NIST.IR.8454.

[11] M. S. Turan, K. McKay, D. Chang, C. Çalık, L. E. Bassham, J. Kang, and J. Kelsey, "Status Report on the Second Round of the NIST Lightweight Cryptography Standardization Process," NISTIR 8369, National Institute of Standards and Technology, Jul. 2021, doi: 10.6028/NIST.IR.8369.

[12] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Cryptanalysis of Ascon," in *Proc. Cryptographers' Track at the RSA Conference (CT-RSA)*, LNCS, vol. 9048, pp. 371–387, 2015, doi: 10.1007/978-3-319-16715-2_20.

[13] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 3rd ed. Boca Raton, FL, USA: CRC Press, 2020.

[14] J. Kelsey, S.-J. H. Chang, and R. A. Perlner, "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash," NIST Special Publication 800-185, National Institute of Standards and Technology, 2016, doi: 10.6028/NIST.SP.800-185.

[15] L. C. dos Santos, J. Großschädl, and A. Biryukov, "FELICSAEAD: Benchmarking of Lightweight Authenticated Encryption Algorithms," in *Proc. CARDIS*, 2019, pp. 216–233, doi: 10.1007/978-3-030-42068-0_13.

[16] S. Sarkar, S. Shafaei, T. S. Jones, and M. W. Totaro, "Secure Communication in Drone Networks: A Comprehensive Survey of Lightweight Encryption and Key Management Techniques," *Drones*, vol. 9, no. 8, Art. no. 583, 2025, doi: 10.3390/drones9080583.

[17] A. Shafique, A. Mehmood, and M. Elhadef, "Survey of Security Protocols and Vulnerabilities in Unmanned Aerial Vehicles," *IEEE Access*, vol. 9, pp. 46927–46948, 2021, doi: 10.1109/ACCESS.2021.3066778.

[18] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard*. Berlin, Germany: Springer, 2002.