

Article

Secure PUF-ASCON-Based Gateway-Assisted D2D Authentication for Resource-Constrained Smart-Manufacturing IIoT Devices

Alanoud Subahi 

Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh 25732, Saudi Arabia; asubahi@kau.edu.sa

Abstract

Smart-manufacturing Industrial Internet of Things (IIoT) deployments increasingly depend on low-latency device-to-device (D2D) communication among resource-constrained, physically exposed field devices. This setting makes mutual authentication and session-key establishment difficult: public-key-intensive or cloud-dependent schemes add overhead, availability dependence, and single points of failure, while weak PUF-based designs may expose challenge-response pairs (CRPs) to replay, disclosure, and modeling attacks. This paper proposes PASMMap, a lightweight PUF-ASCON mutual authentication protocol for gateway-assisted D2D communication in smart-manufacturing IIoT. PASMMap combines SRAM-PUF key reconstruction, fuzzy-extractor helper data, hash- and XOR-based obfuscation, and ASCON authenticated encryption with associated data (AEAD) to protect hardware-rooted identities, hide raw PUF responses, and establish fresh session keys for post-authentication data exchange under an explicitly trusted local-gateway model. The protocol is evaluated against physical, protocol-level, and insider threats, including cloning, tampering, replay, man-in-the-middle, CRP disclosure, PUF modeling, stolen-verifier, and known-key attacks. A real-or-random (ROR) analysis bounds the adversary's session-key advantage using hash collisions, PUF-response prediction, online guessing, and ASCON AEAD security. A mixed-platform evaluation based on ESP32 primitive timings for the edge devices and desktop timings for the resource-rich gateway yields an estimated total computation cost of 4.762 ms. The initiator and responder require 2.006 ms/264.79 μ J and 2.679 ms/353.63 μ J of computational energy, respectively, while the five-message exchange carries 4704 bits. These results indicate low computational overhead under the stated benchmark and power-model assumptions. However, the protocol totals are operation-count-based estimates, the PUF and fuzzy-extractor operations are simulated, and the energy model excludes several platform- and communication-dependent costs. A complete embedded implementation is therefore required to validate end-to-end latency, memory use, energy consumption, communication-stack overhead, SRAM-PUF reliability, and fuzzy-extractor performance.



Academic Editor: Yujue Wang

Received: 17 June 2026

Revised: 30 July 2026

Accepted: 3 August 2026

Published: 4 August 2026

Copyright: © 2026 by the author.

Licensee MDPI, Basel, Switzerland.

This article is an open access article

distributed under the terms and

conditions of the [Creative Commons](#)[Attribution \(CC BY\)](#) license.

Keywords: ASCON; device-to-device (D2D) authentication; gateway-assisted authentication; Industrial Internet of Things (IIoT); lightweight cryptography; mutual authentication; physical unclonable function (PUF); smart manufacturing

MSC: 94A62; 94A60; 68M12

1. Introduction

The Industrial Internet of Things (IIoT) is a core enabler of smart manufacturing, where sensors, actuators, programmable logic controllers (PLCs), robotic units, embedded gateways, and edge controllers exchange operational data to support monitoring, predictive maintenance, closed-loop control, product traceability, and adaptive production [1,2]. Unlike conventional enterprise Internet-of-Things deployments, many factory-floor interactions must be executed locally and within tight timing constraints. Gateway-assisted device-to-device (D2D) communication is therefore important for reducing latency and maintaining production continuity [3,4]. In this setting, authentication directly affects process integrity, equipment behavior, worker safety, and production availability: impersonated devices, replayed commands, or injected control data can disrupt an entire production line [5].

Securing such communication is difficult because IIoT devices are often resource constrained, intermittently connected, and physically exposed. Limited computation, memory, energy, and secure storage make heavyweight mechanisms impractical, while physical accessibility increases the risk of capture, cloning, invasive inspection, and secret extraction [5–7]. At the protocol level, adversaries may eavesdrop, replay, modify transcripts, impersonate endpoints, or exploit desynchronization during pseudonym and counter updates [8,9]. For PUF-enabled devices, exposed or weakly masked challenge-response pairs (CRPs) can also enable disclosure and machine-learning-based modeling attacks [10].

Existing approaches address these issues only partially. Public-key-intensive schemes offer strong security but may impose excessive computation and energy costs on low-cost industrial nodes [6,7], while cloud-centric authentication adds delay, connectivity dependence, and possible single points of failure; edge- and gateway-assisted schemes reduce this dependence but must protect local verifier records [2,4,8]. Lightweight symmetric protocols reduce overhead, yet many still rely on long-term stored secrets or online servers and do not fully support local D2D authorization [3,5,9]. PUF-based schemes provide hardware-rooted identities, especially when SRAM startup behavior is used without additional hardware, but they require fuzzy extraction for noisy responses and careful CRP protection [10–12]. Similarly, ASCON authenticated encryption with associated data (AEAD) efficiently provides confidentiality, integrity, and message authentication for constrained platforms, but it does not by itself provide unclonable device identity or physical-capture resilience [13,14].

These limitations indicate the need for an integrated smart-manufacturing authentication scheme that combines hardware-rooted identity, reconstruction of noisy PUF-derived keys, lightweight obfuscation, gateway-assisted D2D authorization, authenticated encryption, and formal session-key validation while avoiding raw PUF-response transmission and excessive cloud dependence. Motivated by this need, this paper proposes PASMAL, a PUF-ASCON Secure Mutual Authentication Protocol for resource-constrained IIoT devices. PASMAL integrates SRAM-PUF key reconstruction, fuzzy-extractor helper data, hash- and XOR-based obfuscation, pseudonym management, gateway authorization, ASCON AEAD protection, freshness checks, session-key establishment, and secure post-authentication D2D communication over insecure channels.

The protocol treats the local gateway as an authorization coordinator rather than a cloud-dependent authenticator, allowing device pairs to establish trust close to the production process. SRAM-PUF reconstruction binds each endpoint to hardware-specific behavior, while fuzzy-extractor helper data stabilizes the derived secret without exposing the raw response. ASCON AEAD is used to protect both authentication material and post-authentication traffic whose confidentiality, integrity, and freshness must be verified. This combination aligns the protocol design with practical deployment concerns, includ-

ing verifier exposure, desynchronization, CRP modeling, and lightweight operation for constrained industrial nodes.

The main contributions of this paper are summarized as follows:

- PASMMap is designed as a lightweight gateway-assisted D2D mutual-authentication protocol that combines SRAM-PUF key reconstruction, fuzzy extractors, hash/XOR obfuscation, pseudonym management, gateway authorization, and ASCON AEAD to establish fresh session keys for smart-manufacturing IIoT devices.
- A threat-driven security analysis is developed under physical, Dolev–Yao, and CK-style adversarial capabilities, covering replay, man-in-the-middle, impersonation, desynchronization, CRP disclosure, PUF modeling, insider, stolen-verifier, known-key, and message-modification attacks.
- Formal security validation is provided through an ROR-based session-key proof that bounds the adversary’s advantage by hash-collision, PUF-prediction, online-guessing, and ASCON-AEAD terms.
- PASMMap is compared with recent authentication protocols using selected security and functionality criteria and mixed-platform performance estimates. Under the stated benchmark and power-model assumptions, PASMMap has an estimated total computation cost of 4.762 ms, with initiator and responder costs of 2.006 ms/264.79 μ J and 2.679 ms/353.63 μ J, respectively, and a communication overhead of 4704 bits over five messages. For the sixteen selected criteria, each PASMMap entry is identified as protocol-design support, informal analysis, formal proof, or an explicit assumption; the checklist provides a descriptive comparison rather than an overall protocol ranking.

2. Related Work

Lightweight authentication for IIoT has mainly relied on symmetric primitives, hash/XOR operations, pseudonyms, and authenticated encryption to reduce the cost of repeated authentication on constrained industrial devices. Esfahani et al. [15] proposed an M2M authentication mechanism for industrial IoT using symmetric operations, hash functions, and pre-shared secrets through an authentication server, while Lara et al. [16] presented a four-message IIoT protocol based on lightweight hash, XOR, addition, and subtraction operations, with BAN and AVISPA validation. More recent schemes strengthen privacy and formal validation: REAP-IIoT uses gateway support, hash operations, and AEGIS AEAD with ROR and Scyther verification [17]; CMAF-IIoT combines chaotic maps with ASCON-based authenticated encryption in a user–gateway–smart-device setting [18]; and the flexible-manufacturing key-agreement scheme of Hammad et al. uses AEGIS AEAD, hash operations, PUF-related material, and ROR analysis [19]. Other recent work also studies cross-layer key agreement for IIoT [20]. However, these protocols generally do not jointly provide hardware-rooted identity, explicit SRAM-PUF reconstruction, protected CRP handling, gateway-assisted D2D authorization, and ASCON-protected post-authentication data exchange.

PUF-based authentication is attractive because device-specific responses can be derived from manufacturing variations instead of relying only on permanently stored secrets. Mall et al. [21] surveyed PUF-based authentication and key-agreement protocols and identified noisy responses, CRP storage, helper data, replay, desynchronization, physical capture, and modeling attacks as central challenges. Several works address these risks directly: PUF-RAKE combines PUF authentication with server-side error correction, CRP obfuscation, and controls on useful CRP exposure [22]; Zheng et al. avoid server-side CRP storage in a peer-to-peer IoT mutual-authentication and key-exchange protocol [23]; Gu et al. use deception with fake PUF behavior and timing penalties against modeling attacks [24]; and Ebrahimabadi et al. manipulate challenge structures to reduce the usefulness of observed

CRPs [25]. Other PUF schemes study different overhead and deployment tradeoffs, including an SRAM-PUF framework without fuzzy extractors [26], PUF-based multi-factor authentication for IoV with biometrics, fuzzy extractors, and dynamic pseudonyms [27], dynamic PUF-based mutual device authentication for IIoT [28], and cross-layer mutual authentication for IIoT [20]. These works confirm the value of PUFs for lightweight identity and CRP protection, but they are not integrated with ASCON AEAD, gateway-side D2D authorization, fuzzy-extractor-based SRAM-PUF key reconstruction, and secure post-authentication data exchange in one smart-manufacturing protocol.

Gateway assistance and AEAD are also important for practical smart-manufacturing authentication. A local gateway can reduce dependence on remote cloud services and support low-latency authorization among nearby devices, but existing gateway-based IIoT protocols mostly focus on user–device or device–server authentication rather than explicit D2D authorization between industrial devices [17,18]. Similarly, helper-node and infrastructure-assisted PUF schemes improve authentication robustness but do not define a local industrial gateway that manages pseudonyms, authorization relations, and D2D session establishment [25,27]. AEAD-based protocols show that authenticated encryption is practical for constrained IIoT: REAP-IIoT and the flexible-manufacturing scheme use AEGIS AEAD [17,19], whereas CMAF-IIoT uses ASCON-based protection [18]. Nevertheless, AEAD alone does not provide unclonable device identity, and existing ASCON-based IIoT work does not combine ASCON with SRAM-PUF reconstruction, fuzzy-extractor helper data, and gateway-authorized D2D session establishment.

Formal validation is commonly used to support claims of session-key secrecy and mutual authentication. Recent protocols employ ROR or game-based proofs for session-key indistinguishability [17,19,23,29], BAN logic and AVISPA for authentication reasoning and symbolic verification [16,28,29], and ProVerif or Scyther for secrecy and authenticity checking [17,20,23,27,28]. These methods provide useful assurance, but many existing protocols give only partial treatment to gateway record exposure, stolen-verifier attacks, CRP disclosure, PUF modeling, desynchronization, physical capture, and post-authentication data confidentiality in a single integrated model.

The main limitation is not the absence of lightweight authentication, PUF support, AEAD protection, or formal analysis in isolation; rather, these elements are rarely combined in one smart-manufacturing protocol. The surveyed schemes do not simultaneously provide SRAM-PUF key reconstruction with fuzzy-extractor helper data, protection of raw PUF responses and CRPs, gateway-assisted D2D authorization with pseudonym management, ASCON-based AEAD for authentication and post-authentication data exchange, and an ROR-style proof covering hash collisions, PUF prediction, online guessing, and AEAD security. PASMMap addresses this gap through a unified lightweight design for resource-constrained, physically exposed, and latency-sensitive IIoT devices.

3. Background

This section presents the system model, adversary assumptions, design objectives, and cryptographic preliminaries of PASMMap. The key notations are summarized in Table 1.

Table 1. Key notations.

Symbol	Description
D_i, D_j, GW	Initiator, responder, and local gateway
ID_x, AID_x, PID_x	Real identity, authorization alias, and routing pseudonym of D_x
$s_{GW}, T_{reg,x}$	Gateway master secret and per-device registration salt/timestamp for D_x
$C_x, R_x, Helper_x$	PUF challenge, response, and helper data
K_x, K_x^{GW}	Device PUF key and gateway-recovered key

Table 1. Cont.

Symbol	Description
X_x	Masked gateway record for recovering K_x^{GW}
ctr_x, ctr_g	Device and gateway counters
r_i, r_j, r_g	Independent 128-bit nonces generated by the local CSPRNGs of D_i , D_j , and GW
$T_1, \dots, T_5, \Delta$	Timestamps and freshness window
$M_1, \dots, M_5$	Authentication messages
N_v, A_v	ASCON nonce and associated data, $v = 1, \dots, 5$
Q_v, Tag_v	ASCON ciphertext and tag, $v = 1, \dots, 5$
$E_K(\cdot), D_K(\cdot)$	ASCON encryption and decryption under key K
$\mathcal{L}_{ij}$	Gateway authorization relation for (D_i, D_j)
$H(\cdot), Tr_{128}(\cdot)$	Hash function and 128-bit truncation
SK_{ij}	Session key shared by D_i and D_j

3.1. System Model

The system model considered in this work, illustrated in Figure 1, represents a smart-manufacturing IIoT environment composed of resource-constrained end devices and a nearby industrial gateway. The end devices may include sensors, actuators, programmable logic controllers (PLCs), robotic units, and embedded monitoring nodes that cooperate through gateway-assisted device-to-device (D2D) exchanges to support real-time production tasks, such as sensing, coordination, and local control. These devices operate on factory floors, assembly lines, and other exposed industrial locations where low-latency local communication is required for timely actuation and coordination. The considered architecture supports local interactions among legitimate IIoT entities through the gateway without requiring continuous reliance on a remote cloud service or centralized backend during routine operation. A trusted administrative authority is assumed to perform deployment and provisioning, after which the devices communicate over wireless or industrial field communication links in the production environment. This setting reflects the practical requirements highlighted in the introduction, namely lightweight operation, low delay, scalability, and suitability for physically exposed and resource-constrained industrial nodes.

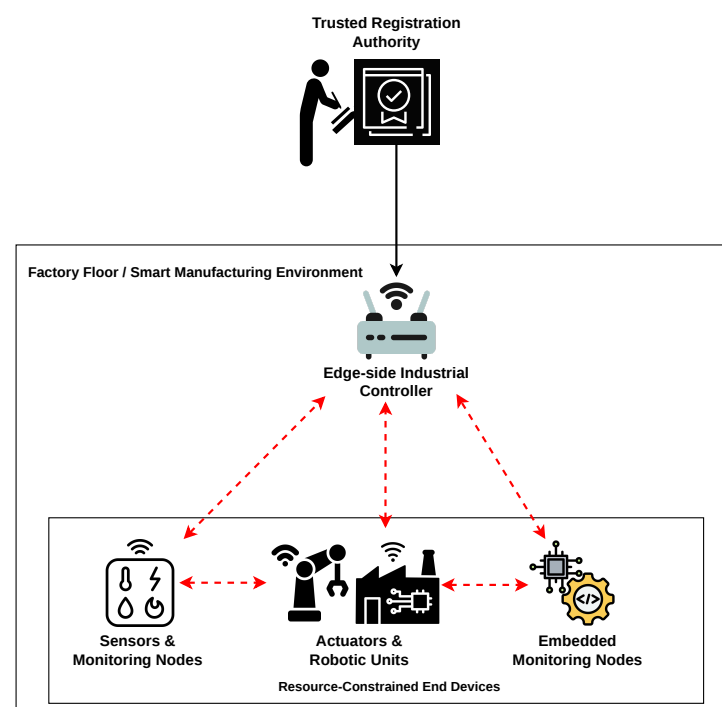


Figure 1. Smart-manufacturing industrial IoT system model.

3.2. Adversary Model

The security analysis adopts a Dolev–Yao and Canetti–Krawczyk (CK) style adversary model [30,31] in which a polynomial-time adversary $\mathcal{A}$ controls the public channels among devices and the gateway, and can eavesdrop, replay, delay, inject, modify, block, reorder messages, launch concurrent sessions, and issue session-oriented queries such as Send, Reveal, Corrupt, and Test subject to freshness restrictions. The adversary may attempt man-in-the-middle, impersonation, desynchronization, replay, and physical-capture-assisted attacks, and may read exposed endpoint state such as pseudonyms, counters, challenges, timestamps, and helper data; however, the trusted enrollment authority, the gateway protected secret region, ASCON, the hash function, and uncompromised SRAM-PUF behavior are assumed secure. The local gateway is a trusted protocol participant that generates and knows SK_{ij} ; therefore, confidentiality is claimed against external channel adversaries and unauthorized devices, not against a malicious, curious, or compromised gateway. If protected gateway storage or uncompromised SRAM-PUF behavior is lost, the corresponding session-key, verifier-protection, and physical-security claims no longer apply. Under these assumptions, PASMMap aims to prevent device impersonation, session-key disclosure, device tracing, and acceptance of stale or manipulated messages except with negligible probability.

Each endpoint and the gateway is assumed to implement a device-local cryptographically secure pseudorandom number generator (CSPRNG) using a deterministic random bit generator (DRBG) [32,33]. At secure startup, and whenever reseeding is required, the DRBG obtains sufficient entropy for 128-bit security from an onboard true random number generator (TRNG) or another validated hardware entropy source with health testing [34]. The internal DRBG state must be protected against disclosure and rollback. The values r_i , r_j , and r_g are independent 128-bit DRBG outputs; raw or reconstructed PUF responses are not used as a substitute for fresh randomness. If entropy-source testing, DRBG instantiation, reseeding, or random-bit generation fails, the entity aborts the protocol before transmitting a message or committing nonce, counter, pseudonym, or session state.

3.3. Design Objectives

PASMMap is designed with the following objectives for secure and efficient gateway-assisted D2D communication in smart-manufacturing IIoT environments:

- O1: Mutual Authentication: Ensure that only legitimate devices and the gateway can participate in an authenticated session.
- O2: Authorized Access: Permit secure communication only between device pairs allowed by the local access policy.
- O3: Session-Key Freshness: Establish a fresh session key for each protocol run to prevent reuse of stale session material.
- O4: Lightweight Efficiency: Minimize computation, memory, and communication overhead for resource-constrained industrial devices.
- O5: Hardware-Rooted Trust: Rely on device-unique hardware behavior to reduce dependence on permanently stored long-term secrets.
- O6: Privacy Preservation: Protect device identities and reduce linkability across different authentication sessions.
- O7: Attack Resilience: Resist common threats such as replay, impersonation, man-in-the-middle, desynchronization, physical-inspection, insider, stolen-verifier, and modeling attacks.
- O8: Secure Data Protection: Provide confidentiality, integrity, authenticity, and replay protection for post-authentication communication.

O9: Formal Security Support: Enable formal validation of secrecy, authentication, and session-key security properties.

3.4. Cryptographic Preliminaries

This subsection summarizes the lightweight cryptographic primitives used in PSMAP.

3.4.1. ASCON

ASCON [13] is a lightweight authenticated-encryption-with-associated-data (AEAD) scheme. It can be represented as

$$\Pi = (\mathcal{K}, \mathcal{N}, \mathcal{A}, \mathcal{M}, \mathcal{C}, \mathcal{T}, E, D), \quad (1)$$

where $\mathcal{K}$, $\mathcal{N}$, $\mathcal{A}$, $\mathcal{M}$, $\mathcal{C}$, and $\mathcal{T}$ denote the key, nonce, associated-data, message, ciphertext, and tag spaces, respectively. PSMAP specifically employs ASCON-128; therefore, the ASCON key, nonce, and authentication tag used in each AEAD call are 128 bits long.

ASCON must be used in a nonce-respecting manner: a nonce must not be repeated for encryption under the same key. This requirement is stated for the original ASCON v1.2 family and is also explicit in the standardized Ascon-AEAD128 security guidance [13,35].

For a key K , nonce N , associated data A , and plaintext M , ASCON encryption is written as

$$(Q, Tag) \leftarrow E_K(N, A, M), \quad (2)$$

where Q is the ciphertext and Tag is the authentication tag. The corresponding decryption and verification operation is written as

$$M \leftarrow D_K(N, A, Q, Tag), \quad (3)$$

which returns $\perp$ if tag verification fails. ASCON protects both the encrypted payload and the associated data, thereby providing confidentiality, integrity, and authenticity for protocol messages and post-authentication data in PSMAP.

3.4.2. Physical Unclonable Functions

A physical unclonable function (PUF) [11] is a hardware-rooted primitive that maps a challenge to a device-specific response according to manufacturing-induced physical variations. It is modeled as

$$PUF_x : \mathcal{C} \rightarrow \mathcal{R}, \quad (4)$$

where $\mathcal{C}$ is the challenge space and $\mathcal{R}$ is the response space of device D_x . For a challenge C_x , the response is

$$R_x \leftarrow PUF_x(C_x). \quad (5)$$

Since the response behavior is tied to the physical structure of the device, a genuine PUF is difficult to clone or predict with non-negligible advantage. PSMAP uses this property to bind authentication to the enrolled device hardware instead of relying only on secrets stored in non-volatile memory. For SRAM-PUF implementations, power-up states originate from manufacturing-dependent transistor mismatches and provide device-specific but noisy startup patterns; repeated measurements therefore require reliability characterization and error correction [36].

3.4.3. Fuzzy Extractor

Because repeated PUF evaluations may produce noisy responses, PASMMap uses a fuzzy extractor [12,37] to reconstruct a stable key. A fuzzy extractor consists of two algorithms,

$$\begin{aligned}\text{Gen} : \mathcal{R} &\rightarrow \{0,1\}^\ell \times \{0,1\}^*, \\ \text{Rep} : \mathcal{R} \times \{0,1\}^* &\rightarrow \{0,1\}^\ell \cup \{\perp\},\end{aligned}\quad (6)$$

where ℓ is the extracted key length. During enrollment, a reference PUF response R_x is processed as

$$(K_x, \text{Helper}_x) \leftarrow \text{Gen}(R_x), \quad (7)$$

where K_x is the stable PUF-derived key and Helper_x is public helper data. During authentication, a fresh noisy response R'_x reconstructs the key as

$$K_x \leftarrow \text{Rep}(R'_x, \text{Helper}_x). \quad (8)$$

Correctness requires reconstruction when R'_x is sufficiently close to R_x . Security is parameter dependent: public helper data may reduce the conditional min-entropy of the PUF response, so the extractor output length and error-correction parameters must account for this leakage. Helper-data exposure alone should not enable recovery of K_x only when the fuzzy extractor's entropy and leakage bounds are satisfied [37,38].

4. The Proposed Scheme

This section presents PASMMap, a lightweight PUF-ASCON scheme for gateway-assisted D2D authentication and secure communication in smart-manufacturing IIoT environments. PASMMap uses SRAM-PUF-based secrets, lightweight hashing, XOR masking, and ASCON, with provisioning handled by a trusted authority and routine authentication relayed locally by the gateway.

4.1. Design Overview

PASMMap separates enrollment-time credential generation from runtime authentication and message protection. During provisioning, each device derives a stable hardware-bound secret from its SRAM-PUF through a fuzzy extractor. This secret is never transmitted and is not stored persistently in plaintext; instead, it is reconstructed locally only when authentication is required. To avoid the scalability burden of storing pairwise secrets or peer pseudonyms, each endpoint keeps only its own PUF challenge, helper data, routing pseudonym, counter, and required authorization aliases. The gateway maintains the registration records, alias-to-pseudonym mapping, counters, and authorization relation; during runtime, it authenticates both endpoints independently, checks whether the requested pair is authorized, and delivers a fresh session key through ASCON-protected messages. Because GW generates and delivers this key, PASMMap provides gateway-assisted data protection rather than gateway-hidden end-to-end confidentiality.

Let $\text{Tr}_{128}(\cdot)$ denote truncation to 128 bits. For each device D_x , the trusted authority establishes one device-gateway credential rooted in the PUF-derived key K_x . The device reconstructs K_x from its local PUF, while the gateway recovers the corresponding value K_x^{GW} from the masked record X_x using its protected master secret. Consequently, endpoint storage remains $O(1)$ rather than $O(n-1)$ in a deployment with n devices, while the gateway enforces pair authorization and creates the session key used for protected data exchange.

All concatenations in hash inputs, ASCON-128 nonces, associated data, and encrypted plaintexts are implementation shorthand for a canonical byte encoding $\text{enc}(\lambda, f_1, \dots, f_m)$.

The domain label λ is a fixed encoded role string, e.g., req, gw, rsp, to- D_i , or to- D_j , and every field has a fixed width: identities, aliases, pseudonyms, nonces, keys, and tags are 128 bits, while timestamps and counters are 32-bit unsigned network-byte-order values. Thus $\parallel$ denotes unambiguous typed concatenation, not variable-length textual concatenation. Each ASCON-128 encryption must use a fresh key–nonce pair. PASMMap enforces this requirement through CSPRNG-generated 128-bit values r_i , r_j , and r_g , together with timestamps, monotonic counters, and replay rejection; an implementation must abort rather than reuse any pair (K_v, N_v) [13,35].

4.2. Provisioning Phase

The provisioning phase is executed once in a protected environment before devices are deployed on the factory floor.

- P1:** The trusted authority selects a master secret s_{GW} and stores it only in a protected provisioning or gateway region. It also fixes the hash function $H(\cdot)$, ASCON parameters, and freshness window Δ .
- P2:** For each device D_x , the trusted authority assigns a real identity ID_x , a gateway-resolved authorization alias AID_x , a PUF challenge C_x , an initial counter $ctr_x = 0$, and a unique, fixed-length per-device registration salt/timestamp $T_{reg,x}$.
- P3:** Device D_x evaluates its SRAM-PUF and derives a stable key using the fuzzy extractor:

$$R_x \leftarrow \text{PUF}_x(C_x), \quad (K_x, \text{Helper}_x) \leftarrow \text{Gen}(R_x). \quad (9)$$

- P4:** The trusted authority computes the initial routing pseudonym and masked gateway record:

$$\begin{aligned} PID_x &= \text{Tr}_{128}(H(ID_x \parallel s_{GW} \parallel T_{reg,x})), \\ X_x &= K_x \oplus \text{Tr}_{128}(H(s_{GW} \parallel ID_x \parallel T_{reg,x})). \end{aligned} \quad (10)$$

The gateway stores ID_x , AID_x , PID_x , $T_{reg,x}$, X_x , and ctr_x . The device stores only ID_x , AID_x , PID_x , C_x , Helper_x , and ctr_x ; neither the raw response R_x nor the key K_x is kept in plaintext after provisioning.

- P5:** The trusted authority installs access policies at the gateway. For an authorized pair (D_i, D_j) , the gateway stores $\mathcal{L}_{ij} = 1$ and maintains the mapping from authorization aliases to current and previous routing pseudonyms. No pairwise secret or global peer-pseudonym directory is installed on endpoints; the gateway alone maintains the per-device registration records and access-control relation $\mathcal{L}$.

4.3. Local Secret Reconstruction

Before starting an authentication run, each device reconstructs its own PUF-derived key. For device D_x , the reconstruction process is

$$R'_x \leftarrow \text{PUF}_x(C_x), \quad K_x \leftarrow \text{Rep}(R'_x, \text{Helper}_x). \quad (11)$$

If reconstruction returns $\perp$, the device aborts. Otherwise, D_x uses K_x only to authenticate itself to GW and to decrypt the session material returned by GW; it does not reconstruct or store any pairwise secret for another endpoint.

When needed, the gateway recovers the registered device key as

$$K_x^{GW} = X_x \oplus \text{Tr}_{128}(H(s_{GW} \parallel ID_x \parallel T_{reg,x})), \quad (12)$$

and uses K_x^{GW} to verify messages from D_x and protect key-delivery messages destined for D_x . This design binds authentication to the genuine PUF-enabled device while keeping endpoint storage constant as the deployment grows.

4.4. Authenticated Key Exchange Phase

Assume that D_i initiates a secure session with D_j through the local gateway GW. Device D_i does not need to know the current routing pseudonym of D_j ; it includes the requested peer alias AID_j inside an ASCON-protected payload sent to GW. The gateway resolves this alias, authenticates both endpoints, checks the authorization relation $\mathcal{L}_{ij}$, and delivers a fresh session key SK_{ij} to both devices.

As shown in Figure 2, messages M_1 – M_3 authenticate the participants. The gateway then derives SK_{ij} and delivers it through M_4 and M_5 . The equations below define these steps in detail.

A1: Initiation by D_i . Device D_i obtains an independent 128-bit nonce r_i from its hardware-seeded CSPRNG and prepares timestamp T_1 and monotonic counter value ctr_i . It computes

$$\begin{aligned} N_1 &= r_i, & A_1 &= PID_i \| ctr_i \| T_1 \| req, \\ K_1 &= \text{Tr}_{128}(H(K_i \| PID_i \| r_i \| ctr_i \| T_1)), \\ \text{plain}_1 &= ID_i \| AID_j \| r_i \| ctr_i. \end{aligned} \quad (13)$$

Then it computes $(Q_1, Tag_1) \leftarrow E_{K_1}(N_1, A_1, \text{plain}_1)$ and sends

$$M_1 = \langle PID_i, r_i, ctr_i, T_1, Q_1, Tag_1 \rangle \quad (14)$$

to GW.

A2: Authentication of D_i by GW. Upon receiving M_1 , GW verifies that PID_i is registered, checks $|T_{now} - T_1| \leq \Delta$, rejects stale counters, and recovers K_i^{GW} from X_i using its protected master secret. It reconstructs N_1 and A_1 , derives K_1 , and decrypts

$$\text{plain}_1 \leftarrow D_{K_1}(N_1, A_1, Q_1, Tag_1). \quad (15)$$

The gateway accepts D_i only if ASCON verification succeeds, the recovered ID_i , r_i , and ctr_i match the registered or session values, AID_j resolves to a registered responder, and $\mathcal{L}_{ij} = 1$.

A3: Gateway challenge to D_j . After authenticating D_i , GW resolves the current routing pseudonym of D_j , obtains an independent 128-bit nonce r_g from its hardware-seeded CSPRNG, and prepares timestamp T_2 and counter ctr_g . It computes

$$\begin{aligned} N_2 &= r_g, & A_2 &= PID_j \| ctr_g \| T_2 \| gw, \\ K_2 &= \text{Tr}_{128}(H(K_j^{GW} \| PID_j \| r_g \| ctr_g \| T_2)), \\ \text{plain}_2 &= AID_i \| r_i \| r_g \| ctr_i. \end{aligned} \quad (16)$$

It computes $(Q_2, Tag_2) \leftarrow E_{K_2}(N_2, A_2, \text{plain}_2)$ and sends

$$M_2 = \langle PID_j, r_g, ctr_g, T_2, Q_2, Tag_2 \rangle \quad (17)$$

to D_j .

A4: Authentication of GW by D_j . Device D_j checks that PID_j matches its current or previous routing pseudonym, verifies $|T_{now} - T_2| \leq \Delta$, and checks that ctr_g is fresh. It reconstructs K_j , reconstructs N_2 and A_2 , derives K_2 , and decrypts

$$\text{plain}_2 \leftarrow D_{K_2}(N_2, A_2, Q_2, Tag_2). \quad (18)$$

Device D_j authenticates GW only if ASCON verification succeeds and the recovered AID_i , r_i , r_g , and ctr_i are consistent with the gateway challenge and local request context.

A5: Response by D_j . After authenticating GW, D_j obtains an independent 128-bit nonce r_j from its hardware-seeded CSPRNG and prepares timestamp T_3 and monotonic counter value ctr_j . It computes

$$\begin{aligned} N_3 &= r_j, & A_3 &= PID_j \| ctr_j \| T_3 \| \text{rsp}, \\ K_3 &= \text{Tr}_{128}(H(K_j \| PID_j \| r_j \| r_g \| ctr_j \| T_3)), \\ \text{plain}_3 &= ID_j \| AID_i \| r_i \| r_j \| r_g \| ctr_j. \end{aligned} \quad (19)$$

It computes $(Q_3, Tag_3) \leftarrow E_{K_3}(N_3, A_3, \text{plain}_3)$ and sends

$$M_3 = \langle PID_j, r_j, ctr_j, T_3, Q_3, Tag_3 \rangle \quad (20)$$

to GW.

A6: Authentication of D_j by GW. Upon receiving M_3 , GW checks PID_j , verifies $|T_{now} - T_3| \leq \Delta$, and rejects stale counters. It reconstructs N_3 and A_3 , derives K_3 using K_j^{GW} , and decrypts

$$\text{plain}_3 \leftarrow D_{K_3}(N_3, A_3, Q_3, Tag_3). \quad (21)$$

The gateway accepts D_j only if ASCON verification succeeds and the recovered ID_j , AID_i , r_i , r_j , r_g , and ctr_j match the active run.

A7: Session-key generation and delivery by GW. After authenticating both endpoints, GW computes

$$SK_{ij} = \text{Tr}_{128}(H(K_i^{GW} \| K_j^{GW} \| r_i \| r_j \| r_g \| PID_i \| PID_j)). \quad (22)$$

This makes GW a trusted key-distribution party that knows SK_{ij} . It then generates delivery timestamps T_4 and T_5 and protects the key separately for D_i and D_j :

$$\begin{aligned} N_4 &= \text{Tr}_{128}(H(r_i \| T_4 \| \text{to-}D_i)), \\ A_4 &= PID_i \| T_4 \| ctr_i \| \text{to-}D_i, \\ K_4 &= \text{Tr}_{128}(H(K_i^{GW} \| PID_i \| r_i \| ctr_i \| T_4)), \\ \text{plain}_4 &= SK_{ij} \| AID_j \| r_i \| r_j \| r_g, \\ (Q_4, Tag_4) &\leftarrow E_{K_4}(N_4, A_4, \text{plain}_4), \\ N_5 &= \text{Tr}_{128}(H(r_j \| r_g \| T_5 \| \text{to-}D_j)), \\ A_5 &= PID_j \| T_5 \| ctr_j \| \text{to-}D_j, \\ K_5 &= \text{Tr}_{128}(H(K_j^{GW} \| PID_j \| r_j \| r_i \| r_g \| ctr_j \| T_5)), \\ \text{plain}_5 &= SK_{ij} \| AID_i \| r_j \| r_i \| r_g, \\ (Q_5, Tag_5) &\leftarrow E_{K_5}(N_5, A_5, \text{plain}_5). \end{aligned} \quad (23)$$

Finally, GW sends $M_4 = \langle PID_i, T_4, Q_4, Tag_4 \rangle$ to D_i and $M_5 = \langle PID_j, T_5, Q_5, Tag_5 \rangle$ to D_j .

A8: Session-key acceptance by D_i . Device D_i checks PID_i and $|T_{now} - T_4| \leq \Delta$, reconstructs N_4 and A_4 , derives K_4 from its reconstructed K_i , and decrypts

$$\text{plain}_4 \leftarrow D_{K_4}(N_4, A_4, Q_4, Tag_4). \quad (24)$$

It accepts SK_{ij} only if ASCON verification succeeds and the decrypted payload contains the expected AID_j , r_i , r_j , and r_g bound to this session.

A9: Session-key acceptance by D_j . Device D_j checks PID_j and $|T_{now} - T_5| \leq \Delta$, reconstructs N_5 and A_5 , derives K_5 from its reconstructed K_j , and decrypts

$$\text{plain}_5 \leftarrow D_{K_5}(N_5, A_5, Q_5, \text{Tag}_5). \quad (25)$$

It accepts SK_{ij} only if ASCON verification succeeds and the decrypted payload contains the expected AID_i , r_j , r_i , and r_g . After both endpoint checks succeed, D_i and D_j share SK_{ij} for subsequent protected communication.

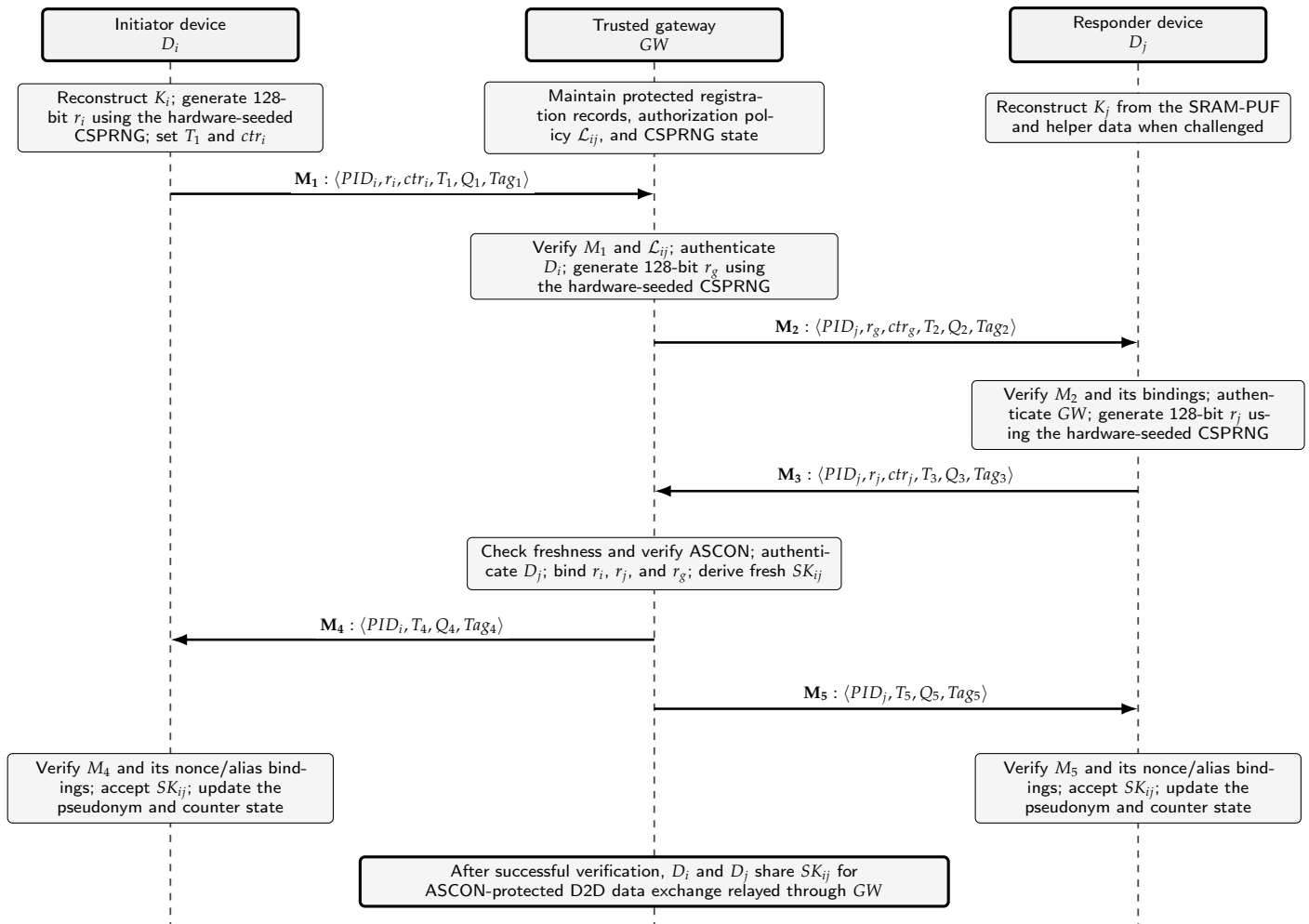


Figure 2. Sequence diagram of the PASMAP gateway-assisted authentication and session-key establishment exchange.

4.5. Pseudonym and Counter Update

After a successful authentication run, the endpoints update their routing pseudonyms and counters to reduce linkability across sessions:

$$\begin{aligned} PID_i^{new} &= \text{Tr}_{128}(H(PID_i \| K_i \| r_i \| r_j \| r_g \| ctr_i)), \\ PID_j^{new} &= \text{Tr}_{128}(H(PID_j \| K_j \| r_j \| r_i \| r_g \| ctr_j)). \end{aligned} \quad (26)$$

The gateway computes the corresponding updates using K_i^{GW} and K_j^{GW} . Each endpoint and the gateway retain current and previous pseudonym values to tolerate the loss of one final delivery message. The associated counters advance as $ctr_i^{new} = ctr_i + 1$ and

$ctr_j^{new} = ctr_j + 1$, and the update is maintained as a bounded pending/committed state rather than an immediate overwrite. An endpoint commits its new tuple only after verifying its final delivery message (M_4 for D_i or M_5 for D_j). The gateway records both the committed tuple (PID_x, ctr_x) and the pending tuple $(PID_x^{new}, ctr_x^{new})$ for each endpoint; in the next run it accepts only one fresh, ASCON-valid use of either tuple and then commits the tuple actually used while deleting stale pending state. This one-step recovery window handles loss of one final delivery message without accepting rollback to older counters. Other devices are not notified of PID_i^{new} or PID_j^{new} , and no network-wide pseudonym broadcast is performed. In later sessions, an initiator addresses the intended peer by the protected alias AID_j inside $plain_1$; only GW resolves this alias to the peer's current or previous routing pseudonym. Thus, dynamic pseudonyms remain known only to the owning device and GW, while unauthorized devices and passive observers cannot build a global pseudonym directory. Counters are increased monotonically in endpoint and gateway records, and messages carrying stale counters or timestamps outside Δ are rejected.

4.6. Secure Data Communication

Once SK_{ij} is established, subsequent process data are protected with ASCON and relayed through GW. Since GW generates SK_{ij} , this channel protects data from external channel adversaries but does not claim confidentiality from GW. For a payload M_d sent by either endpoint, the sender computes

$$(Q_d, Tag_d) \leftarrow E_{SK_{ij}}(N_d, A_d, M_d), \quad (27)$$

where N_d is derived from a session-local counter or sequence value, and A_d binds the session context, message type, direction, and sequence number without exposing long-term identities or updated routing pseudonyms. For ASCON-128 data protection, N_d is encoded as a 128-bit session nonce and must be unique for every encryption under SK_{ij} . The receiver accepts the data only if ASCON verifies Tag_d and the sequence number is fresh; otherwise, the message is discarded.

5. Security Analysis

This section analyzes PASMAP informally against known attacks and formally verifies session-key semantic security under the ROR model.

5.1. Informal Security Analysis

This subsection summarizes PASMAP's resilience to representative attacks and the functional properties that support secure gateway-assisted D2D authentication.

5.1.1. Mutual Authentication and Secure Session Key Agreement

PASMAP provides gateway-assisted mutual authentication among D_i , D_j , and GW through ASCON-protected challenge, nonce, timestamp, and counter checks. Upon receiving M_1 , GW validates PID_i , T_1 , and ctr_i , recovers K_i^{GW} from X_i , derives $K_1 = \text{Tr}_{128}(H(K_i^{GW} \| PID_i \| r_i \| ctr_i \| T_1))$, and verifies (Q_1, Tag_1) ; the decrypted payload must reveal matching ID_i , r_i , and authorized AID_j under $\mathcal{L}_{ij}$, so only a device reconstructing K_i can initiate successfully. D_j authenticates GW from M_2 by reconstructing K_j , deriving K_2 , verifying (Q_2, Tag_2) with $A_2 = PID_j \| ctr_g \| T_2 \| gw$, and confirming $plain_2 = AID_i \| r_i \| r_g \| ctr_i$. GW authenticates D_j from M_3 only when the ASCON payload contains the expected ID_j , AID_i , r_i , r_j , r_g , and ctr_j . After both endpoints are authenticated, GW derives $SK_{ij} = \text{Tr}_{128}(H(K_i^{GW} \| K_j^{GW} \| r_i \| r_j \| r_g \| PID_i \| PID_j))$ and delivers it in M_4 and M_5 ; D_i and D_j accept only after ASCON verification and confirmation of peer aliases and nonce bindings in $plain_4$ and $plain_5$. Thus, session-key acceptance requires endpoint authentication,

gateway validation, freshness, and implicit key confirmation. The unpredictability of SK_{ij} depends on the independent CSPRNG outputs r_i , r_j , and r_g ; failure of the entropy source or DRBG therefore causes the protocol run to abort.

5.1.2. Anonymity and Untraceability

PASMAP keeps ID_i , ID_j , AID_i , and AID_j inside ASCON-protected payloads, while only dynamic routing pseudonyms PID_i and PID_j appear in public messages M_1 – M_5 . Since GW resolves AID_j internally after decrypting M_1 , endpoints need not store, synchronize, or broadcast peers' current pseudonyms, and passive observers cannot link transcripts to long-term identities. After a successful run, endpoints and GW update $PID_i^{new} = \text{Tr}_{128}(H(PID_i \| K_i \| r_i \| r_j \| r_g \| ctr_i))$ and $PID_j^{new} = \text{Tr}_{128}(H(PID_j \| K_j \| r_j \| r_i \| r_g \| ctr_j))$; these values depend on PUF-derived secrets and fresh nonces and are therefore unlinkable to prior pseudonyms without K_i or K_j . Retaining current and previous pseudonyms tolerates one lost final message without network-wide PID broadcast. If an adversary blocks final key delivery before update completion, a routing pseudonym may be reused; consequently, strongest untraceability applies to completed updates, whereas aborted runs still rely on identity hiding, freshness checks, and replay-cache rejection.

5.1.3. PUF-Based Physical Security and Resistance to PI Attacks

Under the stated PUF and protected-storage assumptions, PASMAP provides hardware-rooted support for device authentication through SRAM-PUFs. Each endpoint stores only auxiliary state $(C_x, \text{Helper}_x, PID_x, AID_x, ctr_x)$, never persistent plaintext K_x or R_x , and reconstructs secrets on demand via $R'_x \leftarrow \text{PUF}_x(C_x)$ and $K_x \leftarrow \text{Rep}(R'_x, \text{Helper}_x)$; tampering that shifts PUF behavior beyond fuzzy-extractor tolerance causes reconstruction failure and aborts authentication. The gateway likewise stores no plaintext device keys, keeping only $X_x = K_x \oplus \text{Tr}_{128}(H(s_{GW} \| ID_x \| T_{reg,x}))$, while s_{GW} remains in a protected secure region. Physical inspection (PI) of ordinary gateway records therefore exposes only masked records, timestamps, aliases, counters, and pseudonyms, which are insufficient to recover K_x without s_{GW} . Non-invasive endpoint capture may reveal helper data and protocol state, but not a transferable PUF clone or live PUF oracle; helper data is public by design, and malicious helper modification prevents later key reconstruction or ASCON verification. Under this model, on-demand PUF reconstruction, masked gateway records, and protected storage reduce the feasibility of cloning and PI-assisted key recovery. These are conditional protections rather than “complete physical unclonability”: compromise of s_{GW} , access to a live PUF oracle, invasive extraction, or side-channel and fault attacks falls outside the stated guarantees.

5.1.4. Resistance Against Denial of Service (DoS) Attack

PASMAP mitigates DoS impact through early rejection and lightweight processing. Before accepting M_1 , GW verifies that PID_i is registered, checks $|T_{now} - T_1| \leq \Delta$, rejects replayed or non-monotonic counters, and accepts only ASCON-valid ciphertext-tag pairs, so malformed or modified requests do not alter accepted session state. Similarly, D_j validates its routing pseudonym, timestamp, and gateway counter in M_2 , while GW rejects M_3 messages that fail timestamp, counter, identity, nonce, or tag verification. Routine authentication relies on PUF reconstruction, hash/truncation, XOR masking, and ASCON rather than expensive public-key operations, keeping per-message cost low for constrained IIoT devices. Although volumetric flooding cannot be eliminated by the protocol alone, freshness checks, monotonic counters, replay-cache validation, AEAD rejection, and gateway-side rate limiting limit unnecessary protocol progression.

5.1.5. Resistance Against Replay Attack

PASMAP resists replay attacks through timestamps, nonces, monotonic counters, associated data, and session-state binding. Each message includes a timestamp $T_1, \dots, T_5$ within the freshness window Δ , and ctr_i , ctr_g , and ctr_j are checked against endpoint and gateway replay state; thus, an old M_1 is rejected once GW has recorded the accepted counter and nonce state for PID_i . Later messages are similarly bound to fresh context: M_2 includes PID_j , ctr_g , T_2 , r_i , and r_g ; M_3 includes PID_j , ctr_j , T_3 , r_i , r_j , and r_g ; and M_4 and M_5 bind SK_{ij} to endpoint aliases and nonces. Because ASCON authenticates both ciphertext and associated data, stale or transplanted messages fail timestamp/counter checks, active-session nonce checks, or tag verification. Post-authentication data exchange further uses session counters and sequence numbers in A_d , so replayed application payloads are rejected.

5.1.6. Resistance Against MITM Attack

PASMAP resists man-in-the-middle attacks because every security-critical payload is ASCON-protected under keys derived from PUF-rooted material and fresh protocol values. An adversary $\mathcal{A}$ controlling the public channel cannot compute valid tags for M_1 without K_i , for M_2 or M_3 without K_j , or for M_4 and M_5 without endpoint-specific delivery keys derived from K_i^{GW} and K_j^{GW} . Associated data binds each ciphertext to its routing pseudonym, counter, timestamp, and message role, so modifications to public fields such as PID_x , ctr_x , or T_x are detected. In addition, r_i , r_j , and r_g bind endpoint contributions, AID_i and AID_j remain inside authenticated plaintexts, and GW enforces $\mathcal{L}_{ij}$; therefore, a MITM adversary cannot replace the intended peer, redirect the session to an unauthorized device, splice concurrent sessions, impersonate an entity, or alter the delivered session key without detection.

5.1.7. Resistance Against Secret Leakage, CRP Disclosure, and Modeling Attacks

PASMAP limits secret-leakage impact because SK_{ij} is derived from fresh nonces and long-term PUF-rooted secrets: transcripts may reveal r_i , r_j , r_g , timestamps, counters, and public pseudonyms, but not K_i^{GW} or K_j^{GW} , so SK_{ij} remains computationally hidden. Exposure of endpoint public state (PID_x , C_x , $Helper_x$, AID_x , ctr_x) does not reveal K_x under the assumed fuzzy-extractor parameters only if the conditional min-entropy remaining after helper-data leakage is sufficient for the selected output length [37,38]. PASMAP avoids CRP disclosure because raw PUF responses R_x , reconstructed keys K_x , and plaintext CRP databases are never transmitted; GW stores only XOR-masked records X_x , and the unmasking secret s_{GW} remains in a secure region. Since the protocol neither exposes large labeled challenge-response observations nor provides useful oracle feedback to forged adaptive messages, machine-learning PUF modeling lacks effective training data. Under the assumed PUF and secure-region protections, PASMAP therefore resists ephemeral leakage, CRP disclosure, and modeling attempts.

5.1.8. Resistance Against Impersonation and Masquerade Attacks

Endpoint impersonation fails because valid M_1 and M_3 messages require ASCON tags under keys derived from the sender's PUF secret, current pseudonym, counter, timestamp, and nonce. A forged M_1 must decrypt to the registered ID_i and authorized AID_j , while a forged M_3 must contain the expected ID_j , AID_i , r_i , r_j , r_g , and ctr_j for the active session. Gateway masquerading is prevented because D_j accepts M_2 only after verifying K_2 and the protected binding to AID_i , r_i , r_g , and ctr_i , and both endpoints accept final key delivery only under their delivery keys K_4 and K_5 . Hence, neither an external attacker nor an unauthorized device can impersonate a legitimate endpoint or gateway without the corresponding PUF-rooted secret and authorization relation $\mathcal{L}_{ij}$.

5.1.9. Resistance Against Desynchronization Attacks

PASMAP reduces desynchronization by updating pseudonyms and counters only after authenticated session completion and by retaining both current and previous pseudonyms at endpoints and GW. If an adversary blocks M_4 or M_5 , the previous-pseudonym fallback allows the next valid run to resynchronize without broadcasting updated PIDs. More precisely, GW keeps a single pending tuple $(PID_x^{new}, ctr_x^{new})$ alongside the committed tuple (PID_x, ctr_x) ; the next fresh message that verifies under one of these tuples determines the committed state, and the unused pending state is deleted. Modified update-related messages fail ASCON verification, so no honest party accepts manipulated state, and replayed or rolled-back counters are rejected by the gateway and endpoints. Desynchronization attempts therefore degrade at most into denial of service within this bounded one-final-message-loss recovery model rather than unauthorized authentication or session-key compromise.

5.1.10. Resistance Against Insider and Stolen-Verifier Attacks

PASMAP limits insider misuse because each endpoint stores only its own PUF material, alias, pseudonym, and counter, not peer secrets or peers' current routing pseudonyms. A malicious registered device can authenticate only as itself and cannot derive another device's K_x from PID_x , $Helper_x$, or public transcripts. Disclosure of ordinary gateway records also does not reveal plaintext verifiers: $X_x = K_x \oplus \text{Tr}_{128}(H(s_{GW} \| ID_x \| T_{reg,x}))$ remains protected without the secure-region secret s_{GW} . Since GW enforces $\mathcal{L}_{ij}$ after decrypting M_1 , a legitimate but unauthorized device cannot force a protected session with an arbitrary peer.

5.1.11. Known-Key Security and Message-Modification Resistance

Disclosure of one accepted SK_{ij} does not reveal K_i , K_j , K_i^{GW} , or K_j^{GW} because the session key is a truncated hash over PUF-rooted secrets and fresh nonces. Other sessions use different r_i , r_j , r_g , counters, timestamps, and updated pseudonyms, so a known session key cannot be reused to compute past or future session keys under the one-wayness of $H(\cdot)$. Message modification is detected because ASCON authenticates both ciphertext and associated data; changes to identities, aliases, pseudonyms, counters, timestamps, nonces, sequence numbers, or encrypted payloads cause tag verification failure. This gives known-key resistance for independent sessions, while full forward secrecy after later exposure of both long-term PUF-derived keys remains outside the stated compromise model.

5.2. Formal Security Analysis

This subsection investigates the semantic security of the session key established by PASMAP under the ROR model [31,39], in which the adversary $\mathcal{A}$ is allowed to issue a range of queries to emulate passive, active, and physical-inspection attack scenarios. The formal ROR result is scoped to fresh-session key indistinguishability; the authentication, desynchronization, insider, stolen-verifier, physical-inspection, side-channel, gateway-compromise, and live-PUF-capture claims remain informal unless separately proven under additional games or symbolic verification. The main definitions and preliminaries are given as follows.

- *Participants*: In PASMAP, the participating entities are the initiator device D_i , the responder device D_j , and the local gateway GW. Their protocol instances are denoted by Π_i^a , Π_j^b , and Π_G^c , respectively. For convenience, Π^u denotes any of these instances.
- *Partnership*: Instances Π_i^a and Π_j^b are considered partnered when both endpoint devices have accepted the same session identifier $sid_{ij} = (PID_i, PID_j, r_i, r_j, r_g, ctr_i, ctr_j)$ and the same session key SK_{ij} , and when Π_G^c has authenticated both endpoint devices and delivered SK_{ij} through valid M_4 and M_5 messages.

- *Freshness*: An accepted instance is fresh if neither the instance nor its partner has been queried through Reveal, and if no Corrupt or PI query before acceptance exposes the PUF-derived keys K_i , K_j , or the secure-region gateway master secret s_{GW} used to recover them. Non-invasive disclosure of public endpoint state, helper data, masked database entries, counters, aliases, and pseudonyms does not by itself violate freshness because these values do not reveal K_i , K_j , or s_{GW} .
- *Adversary*: The adversary $\mathcal{A}$ controls the public channel and may issue the oracle queries summarized in Table 2. The goal of $\mathcal{A}$ is to distinguish the real session key of a fresh accepted PSMAP instance from a uniformly random key of the same length.

Table 2. Adversarial query oracles.

Query	Description
Execute	Returns the passive transcript M_1-M_5 among D_i , GW , and D_j .
Send	Sends a chosen message to Π^u and returns its response, if any.
Hash	Queries the random oracle $H(\cdot)$ for derivation, masking, or pseudonym updates.
PUF	Models PUF-response prediction without cloning an honest SRAM-PUF.
Corrupt/PI	Reveals non-secure state, but not s_{GW} or a live PUF; exposing K_i or K_j before acceptance breaks freshness.
Reveal	Returns Π^u 's session key unless it is the tested instance or its partner.
Test	On one fresh accepted endpoint, returns SK_{ij} or an equal-length random key according to hidden bit b .

Definition 1 (Semantic Security). *Compromising PSMAP's semantic security requires $\mathcal{A}$ to distinguish a real session key from an equally long random string for a fresh tested instance. Let $\text{Adv}_{\mathcal{A}}^P(\text{rt})$ denote the advantage of $\mathcal{A}$, running within polynomial time rt , in compromising the semantic security of the session key established by the proposed protocol P . Then $\text{Adv}_{\mathcal{A}}^P(\text{rt}) = |2 \cdot \Pr[b^* = b] - 1|$, where b is the hidden challenge bit generated by the Test oracle and b^* is the guess output by $\mathcal{A}$. If there exists a negligible function ϵ such that $\text{Adv}_{\mathcal{A}}^P(\text{rt}) \leq \epsilon$, then PSMAP is said to achieve semantic security in the ROR model.*

Definition 2. *Consider $\mathcal{A}$ running in probabilistic polynomial time rt against an AEAD scheme s . The adversary is restricted to Q_m encryption/decryption-oracle queries with total message length L . The advantage of $\mathcal{A}$ under an online chosen-ciphertext attack (OCCA3) is defined as:*

$$\begin{aligned} &\text{Adv}_s^{\text{OCCA3}}(Q_m, L, \text{rt}) \\ &\leq \text{Adv}_s^{\text{ORPR-CPA}}(Q_m, L, \text{rt}) \\ &\quad + \text{Adv}_s^{\text{INT-CT}}(Q_m, L, \text{rt}). \end{aligned} \quad (28)$$

Here, $\text{Adv}_{s,\mathcal{A}}^{\text{ORPR-CPA}}(Q_m, L, \text{rt})$ and $\text{Adv}_{s,\mathcal{A}}^{\text{INT-CT}}(Q_m, L, \text{rt})$ denote $\mathcal{A}$'s advantage in distinguishing the online pseudo-random permutation from random under chosen-plaintext attack and in violating ciphertext integrity, respectively. In PSMAP, s is instantiated by ASCON.

Theorem 1. *Let $\mathcal{A}$ run in polynomial time rt with the objective of challenging the semantic security of PSMAP against a fresh tested instance. Let $|\text{HS}|$ denote the output space of $H(\cdot)$, $|\text{PS}|$ denote the effective PUF response space, and $\ell = 128$ denote the length of PUF-derived keys, ASCON keys, and session keys. Let N_s , N_h , and N_p represent the number of Send, Hash, and PUF queries initiated by $\mathcal{A}$, respectively. Furthermore, let $\text{Adv}_{\text{ASCON}}^{\text{OCCA3}}(Q_m, L, \text{rt})$ be the advantage of $\mathcal{A}$ in*

breaking the ASCON AEAD primitive according to Definition 2. Then, the advantage of $\mathcal{A}$ in compromising the semantic security of PSMAP can be expressed as:

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^{\text{P}}(\text{rt}) &\leq \frac{N_h^2}{|\text{HS}|} + \frac{N_p^2}{|\text{PS}|} \\ &\quad + \frac{N_s}{2^{\ell-1}} \\ &\quad + 2 \cdot \text{Adv}_{\text{ASCON}}^{\text{OCCA3}}(Q_m, L, \text{rt}). \end{aligned} \quad (29)$$

The complete game-based proof of Theorem 1 is provided in Appendix A.

6. Performance Evaluation

This section evaluates PSMAP in terms of computational cost, energy consumption, communication overhead, and security/functionality coverage. The results are compared with recent IIoT authentication schemes to highlight the practicality of PSMAP for resource-constrained smart-manufacturing deployments.

6.1. Security and Functionality Features

Table 3 compares the protocols across sixteen selected criteria grouped as functional/architectural attributes, informally analyzed security properties, and formal session-key analysis. The † symbol identifies PSMAP-specific design criteria, while superscripts state the basis for each PSMAP entry. Coverage totals summarize the selected criteria only because the protocols address different architectures, threat models, and resource constraints. Alzahrani [29] does not report gateway authorization, SRAM-PUF trust, capture/CRP-modeling protection, desynchronization resilience, insider resistance, or ASCON-secured data exchange. Patel et al. [40] and Zhang et al. [6] report several authentication and attack-resistance properties but not the selected SRAM-PUF-ASCON combination. Tanveer et al. [17] and Hammad et al. [19] report gateway/controller-assisted authentication and ROR validation but not the selected combination of D2D mutual authentication, SRAM-PUF trust, CRP/modeling protection, and ASCON-secured post-authentication exchange.

Table 3. Comparison of security and functionality features.

Feature	Alzahrani [29]	Patel et al. [40]	Zhang et al. [6]	Tanveer et al. [17]	Hammad et al. [19]	PSMAP
Functional and architectural attributes						
$\mathcal{F}_1^\dagger$	✓	✓	✓	×	×	✓ ^D
$\mathcal{F}_2^\dagger$	×	✓	✓	✓	✓	✓ ^D
$\mathcal{F}_3$	✓	✓	✓	✓	✓	✓ ^{D/FP}
$\mathcal{F}_4^\dagger$	×	×	✓	×	✓	✓ ^D
$\mathcal{F}_5^\dagger$	×	×	×	×	×	✓ ^{D/A}
$\mathcal{F}_{15}^\dagger$	×	×	×	×	×	✓ ^D
Category coverage	2/6	3/6	4/6	2/6	3/6	6/6
Security properties supported by informal analysis and stated assumptions						
$\mathcal{F}_6$	✓	✓	✓	✓	✓	✓ ^{IA}
$\mathcal{F}_7$	✓	✓	✓	✓	✓	✓ ^{IA}
$\mathcal{F}_8$	✓	✓	✓	✓	✓	✓ ^{IA}
$\mathcal{F}_9$	✓	✓	✓	✓	✓	✓ ^{IA}
$\mathcal{F}_{10}$	×	✓	✓	✓	✓	✓ ^{IA/A}
$\mathcal{F}_{11}$	×	×	×	×	×	✓ ^{IA/A}
$\mathcal{F}_{12}$	✓	✓	×	✓	×	✓ ^{IA}
$\mathcal{F}_{13}$	×	×	×	✓	✓	✓ ^{IA}
$\mathcal{F}_{14}$	×	✓	✓	✓	✓	✓ ^{IA/A}
Category coverage	5/9	7/9	6/9	8/9	7/9	9/9

Table 3. Cont.

Feature	Alzahrani [29]	Patel et al. [40]	Zhang et al. [6]	Tanveer et al. [17]	Hammad et al. [19]	PASMAP
Formally analyzed session-key property						
$\mathcal{F}_{16}$	✓	✓	✓	✓	✓	✓ ^{FP}
Category coverage	1/1	1/1	1/1	1/1	1/1	1/1
Total selected-criteria coverage	8/16	11/16	11/16	11/16	11/16	16/16

Notes: ✓ and × indicate reported support and no reported support, respectively. PASMAP superscripts denote design support (D), informal analysis (IA), formal ROR proof of fresh-session key indistinguishability (FP), and an explicit assumption (A). Only FP denotes formal verification. The † symbol marks PASMAP-specific design criteria; totals summarize the selected criteria only. $\mathcal{F}_1$: D2D mutual authentication; $\mathcal{F}_2$: Gateway access authorization; $\mathcal{F}_3$: Fresh independent session keys; $\mathcal{F}_4$: Lightweight symmetric/PUF design; $\mathcal{F}_5$: SRAM-PUF hardware trust; $\mathcal{F}_6$: Anonymity and untraceability; $\mathcal{F}_7$: Replay resistance; $\mathcal{F}_8$: MITM/modification resistance; $\mathcal{F}_9$: Impersonation resistance; $\mathcal{F}_{10}$: Capture/cloning/PI resistance; $\mathcal{F}_{11}$: CRP/modeling protection; $\mathcal{F}_{12}$: DoS mitigation; $\mathcal{F}_{13}$: Desynchronization resilience; $\mathcal{F}_{14}$: Insider/stolen-verifier resistance; $\mathcal{F}_{15}$: ASCON-secured data exchange; and $\mathcal{F}_{16}$: ROR session-key security.

6.2. Computational Overhead

The execution time of each cryptographic primitive was measured on two platforms: an ESP32 representing the resource-constrained devices D_i and D_j , and an Intel x86_64 desktop representing the resource-rich gateway/server. Table 4 reports the mean execution times. Standard cryptographic functions were implemented using `MBEDTLS`, while ASCON and AEGIS used inline C implementations. The PUF and fuzzy-extractor operations were simulated on the target platforms; therefore, their timings do not validate physical PUF or fuzzy-extractor implementations. The protocol costs in Table 5 were estimated by combining the reported operation counts with the corresponding platform timings. ESP32 timings were used for D_i and D_j , whereas desktop timings were used for the gateway/server. The estimated totals are 2572.450 ms for Alzahrani [29], 59.348 ms for Patel et al. [40], 7.597 ms for Zhang et al. [6], 13.431 ms for Tanveer et al. [17], and 11.765 ms for Hammad et al. [19]. These differences mainly reflect the operation mix of each protocol and the assignment of operations to edge or gateway entities. The totals are mixed-platform estimates rather than end-to-end measurements of the complete protocols. Under the same assumptions, PASMAP has an estimated total computation time of 4.762 ms: 2.006 ms for the initiator on the ESP32, 0.077 ms for the gateway on the desktop, and 2.679 ms for the responder on the ESP32. This is the lowest estimated total among the selected protocols in Table 5 under the stated platform assumptions, but it does not establish overall protocol superiority. PASMAP combines PUF evaluation (T_P), fuzzy-extractor reconstruction (T_R), and ASCON AEAD (T_A), while assigning the gateway operations to the resource-rich platform. A complete device-level implementation is still required to measure physical PUF and fuzzy-extractor behavior, memory use, communication-stack overhead, and end-to-end latency.

Table 4. Basic operation times and implementations (Desktop Baseline vs. ESP32).

Device	ESP32 (Xtensa 32-bit)	Desktop PC (macOS x86_64)	
Processor	Xtensa Dual-Core 32-bit LX6 @ 240 MHz	Intel® Core i5 @ 3.4 GHz	
Memory	520 KB SRAM	High-Performance System RAM	
Sym.	Primitive	ESP32 (ms)	PC (ms)
T_H	Hash function (MBEDTLS SHA-256)	0.1450	0.0048
T_A	ASCON-128 AEAD operation	0.5280	0.0038
T_P	PUF evaluation/generation (simulated)	0.1650	0.0004
T_R	Fuzzy extractor reconstruction Rep(.)	0.2050	0.0005

Table 4. *Cont.*

Sym.	Primitive	ESP32 (ms)	PC (ms)
T_M	Modular Multiplication (mbedtls MPI)	0.0980	0.0014
T_S	ECC scalar mult. (NIST P-256)	642.5000	0.5017
T_{MSM}	Multi-scalar mult. (NIST P-256)	1585.3000	2.4424
T_{SYM}	Symmetric AEAD (AES-128-GCM)	0.4500	0.0041
T_{ASY}	Asymmetric encryption (RSA-2048)	14.5000	0.1030
T_{AEG}	AEGIS-128L AEAD operation	2.1000	0.0138

Note: The benchmark evaluations are performed on embedded and desktop platforms using mbedtls v2.28.10 and inline ASCON/AEGIS implementations. On the embedded platform, the firmware is compiled via the ESP-IDF with -O2 optimization. The desktop platform serves as a high-performance baseline, while the ESP32 measurements provide empirical validation on representative resource-constrained IIoT hardware. PUF and fuzzy extractor logic are evaluated directly via inline simulations on the target hardware.

Table 5. Mixed-platform comparison of computational overheads.

Protocol	Initiator D_i (ESP32)	Gateway/Server (PC)	Responder D_j (ESP32)	Total (ms)
Alzahrani [29]	$4T_H + 2T_M + 2T_S$	$7T_H + 4T_M + 2T_S$	$3T_H + 2T_M + 2T_S$	2572.450
Patel et al. [40]	$T_{SYM} + 2T_{ASY} + 3T_H$	$T_{SYM} + 5T_H$	$2T_{ASY} + 3T_H$	59.348
Zhang et al. [6]	$8T_H + T_P$	$13T_H + 2T_{MSM}$	$8T_H + T_P$	7.597
Tanveer et al. [17]	$3T_H + 4T_{AEG} + T_R$	$T_H + 3T_{AEG}$	$T_H + 2T_{AEG}$	13.431
Hammad et al. [19]	$T_P + 3T_{AEG} + 4T_H$	$T_P + 4T_{AEG} + 2T_H$	$T_P + 2T_{AEG} + 2T_H$	11.765
Proposed PSMAP	$T_P + T_R + 4T_H + 2T_A$	$12T_H + 5T_A$	$T_P + T_R + 5T_H + 3T_A$	4.762

6.3. Communication Overhead

Table 6 summarizes the communication overhead of PSMAP and the reference protocols. The calculation counts only the online PSMAP exchange M_1 – M_5 and excludes provisioning and application payloads. The calculation assumes 32-bit timestamps and counters, and 128-bit identities, aliases, pseudonyms, nonces, session keys, and ASCON-128 tags under the fixed-width encoding convention defined above. Since ASCON ciphertext length equals plaintext length, $|Q_1| = |Q_2| = 416$ bits, $|Q_3| = 672$ bits, and $|Q_4| = |Q_5| = 640$ bits. For example, $|M_1| = (128 + 128 + 32 + 32) + 416 + 128 = 864$ bits. The total overhead for PSMAP is 4704 bits over five messages. For comparison, Alzahrani [29] requires 2 messages and 1984 bits; Patel et al. [40] requires 4 messages and 3936 bits; Zhang et al. [6] requires 4 messages and 2432 bits; Tanveer et al. [17] requires 3 messages and 1440 bits; and Hammad et al. [19] requires 3 messages and 1632 bits. PSMAP has the highest communication cost in this comparison (4704 bits). This additional overhead supports its five-message protected exchange and the security and functionality features listed in Table 3. It therefore represents a trade-off between communication cost and protocol capabilities.

Table 6. Communication overheads comparison.

Protocol	No. of Messages	Total Overhead (bits)
Alzahrani [29]	2	1984
Patel et al. [40]	4	3936
Zhang et al. [6]	4	2432
Tanveer et al. [17]	3	1440
Hammad et al. [19]	3	1632
Proposed PSMAP	5	4704

6.4. Energy Consumption

Energy efficiency is a critical evaluation metric for IoT networks, where edge devices typically operate on limited battery capacities. The computational energy consumption of a cryptographic protocol is fundamentally tied to its execution time and the active power draw of the specific hardware architecture. For a rigorous mixed-architecture analysis, the ESP32 edge nodes (D_i, D_j) are modeled with an active computing current of $I = 40$ mA at $V = 3.3$ V (a standard Modem-sleep state where the CPU operates at 240 MHz), yielding an active power of $P_{edge} = 132$ mW. A standard active processing power of $P_{server} = 65$ W (65,000 mW) is assumed for the resource-rich gateway/server (desktop-class). The computational energy E (in microjoules, μJ) is computed as $E = P \times T$, where T is the execution time derived from Table 5. Figure 3 summarizes the estimated computational energy under the stated mixed-hardware power model. Patel et al. has low estimated edge-device energy but does not report the selected physical-layer mechanisms. Zhang et al. has low estimated edge-device energy and an estimated server-side cost of $\approx 321,568$ μJ because of its multi-scalar multiplication operations. PASMMap's estimated computational energy is 264.8 μJ for the initiator, 353.6 μJ for the responder, and 4979.0 μJ for the gateway. These values depend on the stated timing and power assumptions. Communication energy is also estimated using an ESP32 radio current of 120 mA at 3.3 V and a data rate of 1 Mbps. Based on the payload sizes in Section 6.3, the estimated communication energy is 570 μJ for Tanveer et al., 646 μJ for Hammad et al., 786 μJ for Alzahrani, 963 μJ for Zhang et al., 1558 μJ for Patel et al., and 1862 μJ for PASMMap. PASMMap therefore has the highest estimated communication energy in this comparison, while its combined computational and communication estimate remains below 2.5 mJ per edge node under the stated model. This estimate does not demonstrate device battery lifetime and requires confirmation through direct hardware measurements.

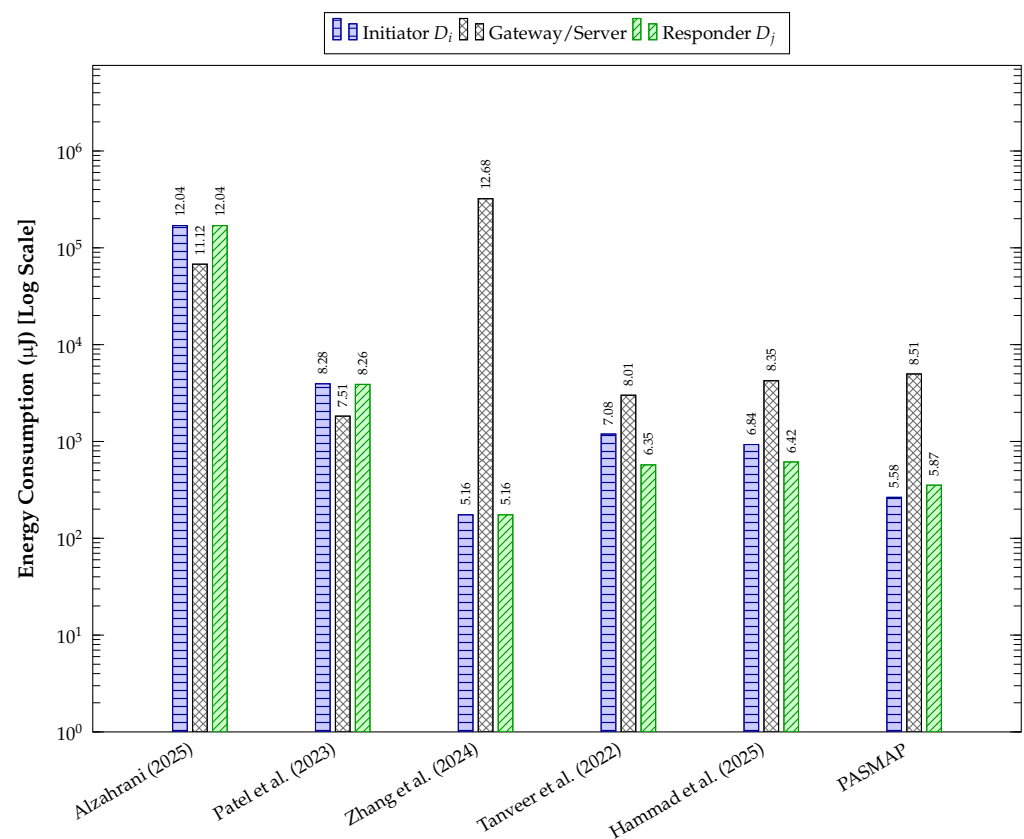


Figure 3. Comparison of energy consumption (μJ) across the initiator, gateway/server, and responder entities [6,17,19,29,40].

7. Conclusions

This paper addressed lightweight mutual authentication and session-key establishment for resource-constrained IIoT devices in smart-manufacturing environments, where physical exposure, low-latency D2D communication, and limited resources make cloud-dependent or public-key-intensive mechanisms difficult to deploy. PASMMap was proposed as a gateway-assisted protocol that integrates SRAM-PUF key reconstruction, fuzzy-extractor helper data, hash- and XOR-based obfuscation, dynamic pseudonyms, freshness checks, and ASCON AEAD protection. The design avoids transmitting raw PUF responses, reduces reliance on persistently stored endpoint secrets, and establishes an authenticated session key for secure D2D data exchange against external channel adversaries under the trusted-gateway model. These guarantees are conditional on a trusted gateway, protected gateway storage, and uncompromised PUF behavior. Because GW generates and knows SK_{ij} , PASMMap does not provide confidentiality from the gateway; gateway compromise, live-PUF capture, invasive extraction, and side-channel or fault attacks remain outside the current security claims. The security analysis showed that PASMMap resists replay, impersonation, man-in-the-middle modification, CRP disclosure, PUF modeling, stolen-verifier exposure, desynchronization, and known-key attacks under the stated assumptions. These attack-resistance claims are supported by the informal analysis and system assumptions, while the formal ROR theorem is limited to fresh-session key semantic security. The ROR proof bounded the adversary's session-key advantage through hash collisions, effective PUF-response prediction, online guessing, and ASCON AEAD security, supporting semantic security of fresh sessions. Because PASMMap uses PUF reconstruction, symmetric operations, counters, timestamps, and AEAD verification instead of expensive public-key operations, it fits constrained gateway-assisted deployments: the estimated mixed-platform computation cost is 4.762 ms; the initiator and responder costs are 2.006 ms/264.79 μ J and 2.679 ms/353.63 μ J, respectively; and the communication overhead is 4704 bits over five messages. Future work will validate the hardware entropy source, CSPRNG/DRBG behavior and cost, nonce-generation failure handling, and side-channel resilience on representative constrained devices. The study has four main limitations: no complete constrained-hardware prototype; simulated SRAM-PUF and fuzzy-extractor operations; model-based rather than measured computation and communication energy; and reliance on a trusted gateway, protected storage, and uncompromised PUF behavior. Because the gateway generates and knows SK_{ij} , confidentiality from it is not claimed. Future work will implement PASMMap on embedded devices, validate PUF and fuzzy-extractor behavior, measure end-to-end latency and energy, and examine alternative gateway-trust models.

Supplementary Materials: The following supporting information can be downloaded at <https://www.mdpi.com/article/10.3390/math14152800/s1>.

Funding: The project was funded by KAU Endowment (WAQF) at King Abdulaziz University, Jeddah, Saudi Arabia. The author, therefore, acknowledges with thanks WAQF and the Deanship of Scientific Research (DSR) for technical and financial support.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the Supplementary Materials. Further inquiries can be directed to the author.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AEAD	Authenticated encryption with associated data
ASCON	Authenticated cipher selected for lightweight cryptography
CSPRNG	Cryptographically secure pseudorandom number generator
CRP	Challenge-response pair
D2D	Device-to-device
DRBG	Deterministic random bit generator
DoS	Denial of service
IIoT	Industrial Internet of Things
MITM	Man-in-the-middle
PASMAP	PUF-ASCON Secure Mutual Authentication Protocol
PUF	Physical unclonable function
ROR	Real-or-random
TRNG	True random number generator

Appendix A. Proof of Theorem 1

Proof. Theorem 1 is derived through a stepwise analysis of a sequence of five games G_i ($i = 0, 1, 2, 3, 4$). These games evaluate whether $\mathcal{A}$ can distinguish the real session key of a fresh PASMAP instance from a random string. Let $\text{Prob}[\text{Win}_i]$ denote the probability that $\mathcal{A}$ successfully guesses the hidden challenge bit b by outputting $b^* = b$ in game G_i .

G_0 . G_0 models the real attack of $\mathcal{A}$ against PASMAP, with the objective of guessing the challenge bit b . Since b is randomly selected by the Test oracle, it follows that:

$$\text{Adv}_{\mathcal{A}}^{\text{P}}(\text{rt}) = |2 \cdot \text{Prob}[\text{Win}_0] - 1|. \quad (\text{A1})$$

G_1 . In G_1 , an eavesdropping attack is modeled by allowing $\mathcal{A}$ to issue the $\text{Execute}(\Pi_i^a, \Pi_G^c, \Pi_j^b)$ query, through which $\mathcal{A}$ obtains the full transcript M_1 – M_5 . The transcript contains public pseudonyms, timestamps, counters, nonces, ciphertexts, and ASCON tags, but it does not reveal K_i , K_j , K_i^{GW} , K_j^{GW} , or the plaintext SK_{ij} carried in M_4 and M_5 . The tested instance is fresh, so Reveal is not allowed on that instance or its partner. Therefore, passive observation alone does not change $\mathcal{A}$'s success probability, which yields:

$$|\text{Prob}[\text{Win}_1] - \text{Prob}[\text{Win}_0]| = 0. \quad (\text{A2})$$

G_2 . In G_2 , active hash and PUF-prediction attempts are modeled. Through Hash queries, $\mathcal{A}$ attempts to find collisions or useful preimages in $H(\cdot)$ for values used in K_1 – K_5 , SK_{ij} , X_x , and pseudonym updates. Through PUF queries, $\mathcal{A}$ attempts to predict an honest device's PUF-derived response or force a useful collision in the effective PUF response space. Under the random-oracle abstraction for $H(\cdot)$ and the unpredictability of the SRAM-PUF response space, the birthday bound gives:

$$|\text{Prob}[\text{Win}_2] - \text{Prob}[\text{Win}_1]| \leq \frac{N_h^2}{2|\text{HS}|} + \frac{N_p^2}{2|\text{PS}|}. \quad (\text{A3})$$

G_3 . In G_3 , $\mathcal{A}$ issues Corrupt/PI queries to obtain non-secure stored information from endpoints or gateway records. For an endpoint, this may reveal ID_x , AID_x , PID_x , C_x , $Helper_x$, and ctr_x ; for the gateway database, it may reveal ID_x , AID_x , PID_x , $T_{reg,x}$, X_x , ctr_x , and $\mathcal{L}_{ij}$. However, K_x is not persistently stored at endpoints, s_{GW} remains in the secure region, and X_x is computationally protected by $\text{Tr}_{128}(H(s_{GW} \| ID_x \| T_{reg,x}))$. The helper data does not reveal K_x without access to the genuine PUF behavior. Hence, the best remaining

strategy in a fresh session is to guess a required 128-bit PUF-derived key or equivalent derived secret during an online attempt, which is bounded by:

$$|\text{Prob}[\text{Win}_3] - \text{Prob}[\text{Win}_2]| \leq \frac{N_s}{2^\ell}. \quad (\text{A4})$$

G_4 . In G_4 , $\mathcal{A}$ attempts to exploit the ASCON-protected messages M_1 – M_5 or the subsequent ASCON-protected data channel to distinguish or recover the established session key. In particular, $\mathcal{A}$ may try to forge valid tags, alter associated data, or distinguish encrypted payloads carrying ID_x , AID_x , nonce bindings, or SK_{ij} . The AEAD security of ASCON under Definition 2 gives:

$$|\text{Prob}[\text{Win}_4] - \text{Prob}[\text{Win}_3]| \leq \text{Adv}_{\text{ASCON}}^{\text{OCCA3}}(Q_m, L, \text{rt}). \quad (\text{A5})$$

From G_0 to G_4 , $\mathcal{A}$ has executed all available oracle queries subject to the freshness restrictions, yet it has not obtained sufficient information to distinguish the tested session key from a random string. Therefore:

$$\text{Prob}[\text{Win}_4] = \frac{1}{2}. \quad (\text{A6})$$

Finally, based on Equations (A1)–(A6), Theorem 1 is formally proven as follows:

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^P(\text{rt}) &= 2 \cdot \left| \text{Prob}[\text{Win}_0] - \frac{1}{2} \right| \\ &= 2 \cdot |\text{Prob}[\text{Win}_0] - \text{Prob}[\text{Win}_4]| \\ &\leq 2 \cdot \sum_{k=1}^4 |\text{Prob}[\text{Win}_k] - \text{Prob}[\text{Win}_{k-1}]| \\ &\leq \frac{N_h^2}{|\text{HS}|} + \frac{N_p^2}{|\text{PS}|} + \frac{N_s}{2^{\ell-1}} + 2 \cdot \text{Adv}_{\text{ASCON}}^{\text{OCCA3}}(Q_m, L, \text{rt}). \end{aligned} \quad (\text{A7})$$

□

References

1. Koprov, P.; Fang, X.; Starly, B. Machine identity authentication via unobservable fingerprinting signature: A functional data analysis approach for MQTT 5.0 protocol. *J. Manuf. Syst.* **2024**, *76*, 59–74.
2. Cui, J.; Zhu, Y.; Zhong, H.; Zhang, Q.; Gu, C.; He, D. Efficient blockchain-based mutual authentication and session key agreement for cross-domain IIoT. *IEEE Internet Things J.* **2024**, *11*, 16325–16338.
3. He, K.; Ren, Z. A new three-factor authentication scheme using Chebyshev chaotic map for peer-to-peer Industrial Internet of Things. *Comput. Netw.* **2024**, *247*, 110450.
4. Zhu, W.; Chen, X.; Jiang, L. A secure and efficient authentication key agreement scheme for industrial internet of things based on edge computing. *Alex. Eng. J.* **2024**, *101*, 52–61.
5. Han, Y.; Guo, H.; Liu, J.; Ehui, B.B.; Wu, Y.; Li, S. An enhanced multifactor authentication and key agreement protocol in industrial Internet of Things. *IEEE Internet Things J.* **2024**, *11*, 16243–16254.
6. Zhang, Q.; Zhou, X.; Zhong, H.; Cui, J.; Li, J.; He, D. Device-side lightweight mutual authentication and key agreement scheme based on chameleon hashing for industrial internet of things. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 7895–7907.
7. Mahmood, K.; Fatima, M.N.; Shamshad, S.; Ghaffar, Z.; Das, A.K.; Alenazi, M.J. A cost-effective key agreement encryption protocol for securing IIoT-enabled WSN communication. *IEEE Internet Things J.* **2024**, *12*, 5185–5193.
8. Zhang, Z.; Huang, W.; Huang, Y.; Liao, Y.; Zhang, Z.; Zhou, S. A domain isolated tripartite authenticated key agreement protocol with dynamic revocation and online public identity updating for IIoT. *IEEE Internet Things J.* **2024**, *11*, 15616–15632.
9. Wu, Y.; Feng, T.; Su, C.; Liu, C. MSAUPL: A multi-server authentication and key agreement protocol for industrial IoT based on user privacy level. *J. Inf. Secur. Appl.* **2025**, *89*, 103991.
10. Fan, C.I.; Lai, C.I.; Medhane, D.V. Cake-puf: a collaborative authentication and key exchange protocol based on physically unclonable functions for industrial internet of things. *IEEE Internet Things J.* **2024**, *11*, 39709–39720.
11. Gassend, B.; Clarke, D.; Van Dijk, M.; Devadas, S. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*; Association for Computing Machinery: New York, NY, USA, 2002; pp. 148–160.

12. Dodis, Y.; Reyzin, L.; Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 523–540.
13. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schl  ffer, M. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.* **2021**, *34*, 33.
14. Kaur, J.; Cintas Canto, A.; Mozaffari Kermani, M.; Azarderakhsh, R. A survey on the implementations, attacks, and countermeasures of the NIST lightweight cryptography standard: Ascon. *ACM Comput. Surv.* **2025**, *58*, 6.
15. Esfahani, A.; Mantas, G.; Matischek, R.; Saghezchi, F.B.; Rodriguez, J.; Bicaku, A.; Maksuti, S.; Tauber, M.G.; Schmittner, C.; Bastos, J. A lightweight authentication mechanism for M2M communications in industrial IoT environment. *IEEE Internet Things J.* **2017**, *6*, 288–296.
16. Lara, E.; Aguilar, L.; Sanchez, M.A.; Garc  a, J.A. Lightweight authentication protocol for M2M communications of resource-constrained devices in industrial Internet of Things. *Sensors* **2020**, *20*, 501.
17. Tanveer, M.; Alkhayyat, A.; Khan, A.U.; Kumar, N.; Alharbi, A.G. REAP-IoT: Resource-Efficient Authentication Protocol for the Industrial Internet of Things. *IEEE Internet Things J.* **2022**, *9*, 24453–24465. <https://doi.org/10.1109/JIOT.2022.3188711>.
18. Tanveer, M.; Badshah, A.; Alasmari, H.; Khan, A.U.; Alasmari, H.; Chaudhry, S.A. CMAF-IoT: Chaotic map-based authentication framework for Industrial Internet of Things. *Internet Things* **2023**, *23*, 100902.
19. Hammad, M.; Badshah, A.; Almeer, M.A.; Waqas, M.; Herbert Song, H.; Chen, S.; Han, Z. Lightweight and Robust Key Agreement for Securing IIoT-Driven Flexible Manufacturing Systems. *IEEE Internet Things J.* **2025**, *12*, 17197–17209. <https://doi.org/10.1109/JIOT.2025.3535846>.
20. Yu, L.; Wu, W.; Mei, L. A lightweight cross-layer mutual authentication with key agreement protocol for iiot. *IEEE Internet Things J.* **2024**, *12*, 7051–7066.
21. Mall, P.; Amin, R.; Das, A.K.; Leung, M.T.; Choo, K.K.R. PUF-based authentication and key agreement protocols for IoT, WSNs, and smart grids: A comprehensive survey. *IEEE Internet of Things J.* **2022**, *9*, 8205–8228.
22. Qureshi, M.A.; Munir, A. PUF-RAKE: A PUF-based robust and lightweight authentication and key establishment protocol. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 2457–2475.
23. Zheng, Y.; Liu, W.; Gu, C.; Chang, C.H. PUF-based mutual authentication and key exchange protocol for peer-to-peer IoT applications. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 3299–3316.
24. Gu, C.; Chang, C.H.; Liu, W.; Yu, S.; Wang, Y.; O'Neill, M. A modeling attack resistant deception technique for securing lightweight-PUF-based authentication. *IEEE Trans. -Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *40*, 1183–1196.
25. Ebrahimbadi, M.; Younis, M.; Karimi, N. A PUF-based modeling-attack resilient authentication protocol for IoT devices. *IEEE Internet Things J.* **2021**, *9*, 3684–3703.
26. Cirillo, F.; Esposito, C. Efficient PUF-Based IoT Authentication Framework without Fuzzy Extractor. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*; Association for Computing Machinery: New York, NY, USA, 2025; pp. 695–704.
27. Yuan, M.; Xiao, Y. PMAKA-IoV: A Physical Unclonable Function (PUF)-Based Multi-Factor Authentication and Key Agreement Protocol for Internet of Vehicles. *Information* **2025**, *16*, 404.
28. Jain, U. Secure, light-weight and dynamic PUF-based mutual device authentication mechanism in industrial IoT networks. *Secur. Priv.* **2024**, *7*, e388.
29. Alzahrani, N. A Verifiably Secure and Lightweight Device-to-Device (D2D) Authentication Protocol for the Resource-Constrained IoT Networks. *IEEE Access* **2025**, *13*, 92982–92996. <https://doi.org/10.1109/ACCESS.2025.3568692>.
30. Dolev, D.; Yao, A.C. On the Security of Public Key Protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. <https://doi.org/10.1109/TIT.1983.1056650>.
31. Canetti, R.; Krawczyk, H. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2001*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001, Volume 2045, pp. 453–474. https://doi.org/10.1007/3-540-44987-6_28.
32. Barker, E.; Kelsey, J. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*; Technical Report NIST Special Publication 800-90A Revision 1; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015. <https://doi.org/10.6028/NIST.SP.800-90Ar1>.
33. Barker, E.; Kelsey, J.; McKay, K.; Roginsky, A.; Turan, M.S. *Recommendation for Random Bit Generator (RBG) Constructions*; Technical Report NIST Special Publication 800-90C; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2025. <https://doi.org/10.6028/NIST.SP.800-90C>.
34. Turan, M.S.; Barker, E.; Kelsey, J.; McKay, K.; Baish, M.; Boyle, M. *Recommendation for the Entropy Sources Used for Random Bit Generation*; Technical Report NIST Special Publication 800-90B; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2018. <https://doi.org/10.6028/NIST.SP.800-90B>.

35. Turan, M.S.; McKay, K.A.; Chang, D.; Kang, J.; Kelsey, J. *Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions*; Technical Report NIST Special Publication 800-232; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2025. <https://doi.org/10.6028/NIST.SP.800-232>.
36. Holcomb, D.E.; Burleson, W.P.; Fu, K. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Trans. Comput.* **2009**, *58*, 1198–1210. <https://doi.org/10.1109/TC.2008.212>.
37. Dodis, Y.; Ostrovsky, R.; Reyzin, L.; Smith, A. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *Soc. Ind. Appl. Math. J. Comput.* **2008**, *38*, 97–139. <https://doi.org/10.1137/060651380>.
38. Delvaux, J.; Gu, D.; Schellekens, D.; Verbauwhe, I. Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis. *IEEE Trans. -Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *34*, 889–902. <https://doi.org/10.1109/TCAD.2014.2370531>.
39. Bellare, M.; Pointcheval, D.; Rogaway, P. Authenticated Key Exchange Secure against Dictionary Attacks. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2000*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1807, pp. 139–155. https://doi.org/10.1007/3-540-45539-6_11.
40. Patel, C.; Bashir, A.K.; AlZubi, A.A.; Jhaveri, R. EBAKE-SE: A novel ECC-based authenticated key exchange between industrial IoT devices using secure element. *Digit. Commun. Netw.* **2023**, *9*, 358–366. <https://doi.org/10.1016/j.dcan.2022.11.001>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.