

Deterministic Homomorphic Identity Tokens for Privacy-Preserving Attribute-based Authentication

Shashank Shekher Tripathi ^{1,2}, Kai Hendrik Wöhnert ¹, and Volker Skwarek ¹

Abstract: Traditional multi-factor authentication (MFA) schemes, despite their layered defences, have been compromised in several common vulnerabilities and exposures (CVEs) due to their sequential verification process, which allows adversaries to target individual identity factors in isolation. In this work, we propose a cryptographic framework that enhances the IdentiToken model by integrating deterministic homomorphic encryption inspired by the Paillier scheme. Our approach generates a structured token composed of encrypted sub-tokens, each representing either a stable or volatile element of a device's identity. These tokens can be recomputed in real-time and compared for authentication without exposing raw attribute values. By leveraging the additive homomorphism, we quantify the degree of change in core attributes across sessions, enabling similarity-based identity verification. Beyond authentication, this token architecture supports a wide range of security applications, including real-time fingerprinting, zero-trust access control, anomaly detection and behavioural malware analysis.

Keywords: identity, authentication, privacy

1 Introduction

The proliferation of network-connected devices has transformed the modern digital infrastructure, giving rise to smart homes, cyber-physical systems, and industrial IoT (Internet of Things) environments. However, this expansion has also led to a surge in identity-based cyberattacks, which now account for the majority of security breaches across sectors. According to IBM's 2024 X-Force Threat Intelligence Index, identity exploitation using valid but compromised credentials rose by 71% in just one year [IB24]. The vulnerability of authentication mechanisms is further reflected in Mitre's 2024 dataset, which lists 161 new CVE (common vulnerabilities and exposures) under CWE(common weakness enumeration)-287 (improper authentication) and 56 under CWE-306 (missing authentication for critical functions)[HLC21]. These findings highlight a systemic failure to authenticate users securely, especially in dynamic or constrained environments.

The situation is exacerbated by the rapid integration of IoT devices. With each node presenting a new attack surface, traditional authentication systems based on passwords, biometric templates, or stored tokens fall short. For instance, the 2024 IoT Security Landscape Report

¹ Hamburg University of Applied Sciences, RTC CyberSec, Ulmenliet 20, 21033 Hamburg, Germany,  <https://orcid.org/0000-0002-0208-5090>; firstname.lastname@haw-hamburg.de,  <https://orcid.org/0000-0002-1346-8126>,  <https://orcid.org/0000-0001-5065-1029>

² Leibniz Institute for Plasma Science and Technology (INP), Felix-Hausdorff-Str. 2, 17489, Greifswald, Germany,  <https://orcid.org/0000-0002-0208-5090>

shows that residential devices face an average of 10 attacks daily [Te24]. Existing multi-factor authentication (MFA) systems, while an improvement over single-factor credentials, are still fundamentally built upon sequential acceptance of one-dimensional authentication vectors. Their inability to reason over dynamic, behavioural and contextual factors limits their applicability in fast-changing environments [Gr17; Wa21; Zh19].

Attribute-based FaR-hash [TS24] addresses the concerns by introducing a framework, complementary to MFA, where entities are authenticated using different classes of attributes. These attributes are processed to compute a unique identity token that reflects the real-time state of an entity. Instead of evaluating isolated factors, the proposed system compares this n-dimensional vector with a previously observed or registered identity, allowing for drift-aware, multi-attribute verification.

However, in its current form, the FaR-Hash token is rigid. While it can indicate how many of a device's attributes have changed, it fails to reveal how much each attribute has changed. This limitation reduces its adaptability in scenarios where gradual or tolerable identity drift occurs over time. For instance, the rotation speed of an industrial motor or the battery capacity of an IoT device may degrade due to natural wear and operational usage. Capturing the magnitude of such change rather than merely detecting its presence enables more informed and flexible authentication decisions that can accommodate realistic, non-malicious variations in identity. Thus, we ask in the form of a research question:

How can a privacy-preserving authentication method be designed that integrates an entity's attributes into a unique identity token while supporting similarity comparison in identity-drift, preserving privacy and enabling secure third-party validation?

2 State of the Art

ISO [IS20] defines identity as a collection of attributes linked to an entity, and authentication as the process of verifying that identity by comparing presented attributes to previously registered ones. Despite their known weaknesses, static password systems remain prevalent and are frequently compromised in practice [KBM15]. Token-based approaches, such as JSON Web Tokens (JWTs) used in OAuth 2.0 and OpenID Connect [Ha12], support federated identity but are susceptible to attacks like token substitution and replay. WebAuthn [W319], a component of the FIDO2 standard, introduces passwordless authentication using asymmetric cryptographic key pairs. While it offers strong protection against phishing, practical challenges such as stale challenge reuse [Mi22], misconfigured verification servers and poor key lifecycle management continue to expose systems to vulnerabilities [Fe23].

WebAuthn, a component of the FIDO2 standard, employs public-key cryptography to enable passwordless authentication with strong phishing resistance. However, practical deployments still face issues such as stale challenge reuse [Mi22], misconfigured verifiers, and inadequate key management practices [Fe23].

Macaroons [Bi14] support flexible delegation by allowing the inclusion of context-specific constraints (e.g., expiration time, endpoint restriction, user roles). Yet, being bearer tokens, they are vulnerable to misuse if intercepted, as possession alone grants access.

3 Identity concept

This section introduces the foundational notion of identity underlying the FaR-hash algorithm and its modified variant proposed in this work. The model is designed to authenticate entities not through singular or static identifiers, but through the structured composition of multiple attributes. Identity, in this context, emerges from the collective expression of these attributes, capturing both stability and variability over time. In its original formulation, the system classified attributes into three categories based on their temporal stability:

- **Static attributes** ($\mathcal{A}_s$): Immutable properties that remain constant for the duration of an entity's identity and are used to derive a structural reference seed. *Examples include hardware fingerprint like physically unclonable function (PUF), MAC address, or device serial number.*
- **Dynamic attributes** ($\mathcal{A}_d$): Properties that vary over time but remain within a predictable range, and are therefore treated as stable components of an entity's identity. Examples include *IP address, screen size or geolocation during a connection window.*
- **Volatile attributes** ($\mathcal{A}_v$): Attributes whose values change unpredictably over time and therefore cannot be assumed to remain stable or within a bounded range. Some examples include *network latency, or DNS response time.*

Token Generation with FaR hash

Seed Construction. Each static attribute s_j is converted into binary form and concatenated:

$$\text{SEED} = \parallel_{j=1}^P \text{Bin}(s_j)$$

This seed acts as an intrinsic deterministic fingerprint of long-term identity traits. Since it's used in every downstream operation, any change in static attributes results in a structurally new token. Additionally, it acts as a virtual private key as it never becomes an explicit part of the token, but rather remains part of the token calculation.

Attribute-Specific Hashing. For every dynamic or volatile attribute x_i , we construct a seed-dependent digest:

$$x'_i := \text{Bin}(x_i) \parallel \text{SEED}, \quad h_i := H(x'_i)$$

This ties each observation to a fixed identity, ensuring that similar attributes from different devices produce dissimilar digests.

Truncation. To modulate the influence of each attribute:

$$\tilde{h}_i := \begin{cases} \text{Slice}_n(h_i), & \text{if } x_i \in \mathcal{D} \\ \text{Slice}_m(h_i), & \text{if } x_i \in \mathcal{V} \end{cases}$$

where $n > m$, $n \bmod m \neq 0$. This design ensures that dynamic attributes contribute more entropy than volatile ones, and non-divisible slice lengths promote bit mixing in later steps.

Stream Assembly. Truncated values are linearly ordered and concatenated into a stream:

$$\text{H_S} := \parallel_{x_i \in \pi(\mathcal{D} \cup \mathcal{V})} \tilde{h}_i$$

The ordering function π defines a fixed order in which attributes are processed.

Cross-Attribute Diffusion. Divide H_S into non-overlapping windows W_i of length ℓ (usually $\ell = m$), and compute:

$$\text{T}_i := \bigoplus_{j=0}^{\ell-1} W_i[j]$$

If the final window is shorter than ℓ , it is padded with zero bytes. This step achieves cross-attribute dependency, diffusing influence across bytes.

Final Token. The final token, thus produced, is called *IdentiToken*, which is derived by :

$$\text{T} := \parallel_i \text{T}_i$$

This token is structure-sensitive, drift-reactive, and privacy-preserving by design.

4 Homomorphic Identity token

This section discusses the method we propose in the paper by modifying the homomorphic encryption to make it deterministic and applying it to generate a modified *IdentiToken*. The modification enables the comparison of partial drift in behaviour and traits of an entity by comparing dynamic and volatile attributes while keeping its privacy properties.

In the base *IdentiToken*, each dynamic or volatile attribute is hashed after being concatenated with a static seed. Truncated hashes are then diffused to produce the final token as presented in section 3. The main element of the proposed modification is deterministic homomorphic encryption.

4.1 Deterministic homomorphic encryption

The homomorphic encryption used for the modification is inspired by Paillier homomorphic encryption [PP99]. The Paillier cryptosystem is a probabilistic public-key encryption scheme introduced by Pascal Paillier in 1999. It is notable for its support of additive homomorphism, meaning that arithmetic operations can be performed on ciphertexts without decrypting them. For any $m_1, m_2 \in \mathbb{Z}_n$,

$$\text{Enc}(m_1) \cdot \text{Enc}(m_2) = \text{Enc}(m_1 + m_2) \mod n^2$$

This property enables secure computations on encrypted data, which is particularly useful in applications like privacy-preserving authentication. However, the algorithm uses a random private key, whereas in our case, we need to make it deterministic. To meet the determinism and contextual binding requirements of the modified IdentiToken, we apply the following modifications:

- The encryption is made **deterministic** by replacing the random nonce r with a derived secret.
- This secret r is generated using a keyed hash:

$$r = \text{HMAC}(k, m \parallel \ell \parallel \text{SEED})$$

- The secret r must satisfy $\gcd(r, n) = 1$ and $r < n$ to remain valid for modular inversion.

Deterministic Homomorphic encryption Let p and q be two large primes such that $n = pq$ and n^2 is the modulus for encryption. The scheme defines a generator $g \in \mathbb{Z}_{n^2}^*$ satisfying $\gcd(L(g^\lambda \mod n^2), n) = 1$, where $\lambda = \text{lcm}(p-1, q-1)$ and $L(u) = \frac{u-1}{n}$, where the keys are:

- Public key: (n, g)
- Private key: (λ, μ) , where $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$

The encryption for a given message $m \in \mathbb{Z}_n$, rather than randomly choosing the blinding factor r , we compute r deterministically as:

$$r = \text{HMAC}(k, m \parallel \ell \parallel \text{SEED})$$

where $\gcd(r, n) = 1$ and label is the name of an attribute which remains standard across a given use case. After that, compute the ciphertext as: $\text{Enc}(m) = g^m \cdot r^n \mod n^2$

For decryption, given a ciphertext c , recover the message using: $\text{Dec}(c) = L(c^\lambda \mod n^2) \cdot \mu \mod n$

Contextual Binding via r By making r dependent on its attribute label and the static seed, the encryption is both:

- **Entity-bound:** different users with the same attribute value will yield different ciphertexts as long as they have at least one attribute different.
- **Attribute-specific:** swapping labels or modifying the seed will result in completely different encryptions.

This strengthens unlinkability and resists codebook attacks, while maintaining deterministic computation needed for token comparison.

4.2 Modified IdentiToken with Deterministic Homomorphic Encryption

In the original IdentiToken algorithm, each dynamic or volatile attribute is hashed after being concatenated with a static seed. Truncated hashes are then diffused to produce the final token as presented in section 3.

Algorithm 1 outlines the process of generating an IdentiToken composed of encrypted sub-tokens. The function `GENERATEIDENTITYKEYS` derives the Paillier parameters (n, g) from the static seed. Crucially, the system then performs a two-stage derivation for each dynamic attribute:

1. The $SEED_{str}$ is processed through the hashed message authentication code-based key derivation function (HKDF)-extract phase [Kr10], acting as the input keying material (IKM) to produce a high-entropy master PRK (Pseudorandom Key). This ensures that any non-uniformity in the raw hardware attributes is smoothed into a computationally independent cryptographic root.
2. For each attribute a_i with identifier id_i , the HKDF-expand phase derives a unique blinding factor r_i by binding the PSK to a specific context string. This string incorporates both the attribute's label and its current value, ensuring that identical data points across different attributes result in divergent, uncorrelated ciphertexts c_i .

4.3 Construction of the proposed IdentiToken

Step 1: Attribute Classification: All identity attributes are partitioned into three disjoint sets: static $S = \{s_1, \dots, s_p\}$, dynamic $D = \{d_1, \dots, d_k\}$, and volatile attributes $V = \{v_1, \dots, v_m\}$. These sets are mutually exclusive and chosen based on the stability profile of the target hardware.

Algorithm 1 IdentiToken generation algorithm

Require: Static attributes $\mathcal{S}$, dynamic $\mathcal{D}$, volatile $\mathcal{V}$

Ensure: Final token T as a collection of sub-tokens

```

1: function GENERATESEED( $\mathcal{S}$ )
2:    $SEED_{str} \leftarrow \text{sort\_and\_concat}((k, v) \in \mathcal{S})$ 
3:    $SEED_{hex} \leftarrow \text{SHA256}(SEED_{str})$ 
4:   return ( $SEED_{str}, SEED_{hex}$ )
5: end function
6: function DERIVESTATUSR( $a_i, id_i, SEED_{str}, n$ )
7:    $info \leftarrow attr: || id_i || : || Str(a_i)$  ▷ colon(:) is delimiter
8:    $PRK \leftarrow \text{HKDF-Extract}(\text{salt}=0, SEED_{str})$ 
9:    $r_{raw} \leftarrow \text{HKDF-Expand}(PRK, info, L = 256)$ 
10:   $r_i \leftarrow \text{BigInt}(r_{raw}) \bmod n$ 
11:  return  $r_i$ 
12: end function
13: function STATELESSIDENTITOKEN( $\mathcal{S}, \mathcal{D}, \mathcal{V}$ )
14:  ( $SEED_{str}, SEED_{hex}$ )  $\leftarrow$  GENERATESEED( $\mathcal{S}$ )
15:  ( $n, g$ )  $\leftarrow$  DERIVEIDENTITYKEYS( $SEED_{str}$ )
16:   $T \leftarrow \emptyset$ 
17:  for all ( $a_i, id_i$ )  $\in \mathcal{D}$  do
18:     $r_i \leftarrow \text{DERIVESTATUSR}(a_i, id_i, SEED_{str}, n)$ 
19:     $c_i \leftarrow (g^{a_i} \cdot r_i^n) \bmod n^2$ 
20:     $T \leftarrow T \cup \{(id_i, c_i, \text{dynamic})\}$ 
21:  end for
22:  for all ( $a_j, id_j$ )  $\in \mathcal{V}$  do
23:     $h_j \leftarrow \text{SHA256}(a_j || SEED_{hex})$ 
24:     $T \leftarrow T \cup \{(id_j, h_j, \text{volatile})\}$ 
25:  end for
26:  return  $T$ 
27: end function

```

Step 2: Stateless Parameter and Seed Generation In contrast to traditional PKI schemes, the system does not require a pre-distributed public key. Instead, all cryptographic parameters are derived as required during the calculation from the device’s hardware signature:

- **Identity Seed:** A deterministic string, $SEED_{str} = \text{sort_and_concat}(\mathcal{S})$, is computed from the static attributes. A hexadecimal salt, $SEED_{hex} = \text{SHA256}(SEED_{str})$, is generated for volatile attribute salting. This value serves as the primary *Input Keying Material* (IKM), anchoring all subsequent cryptographic operations to the physical hardware state.
- **Identity Keypair:** The Paillier modulus $n = pq$ and generator $g = n + 1$ are derived via a deterministic prime-finding function DERIVEPRIME using $SEED_{str}$ as the entropy source. This binds the very ability to decrypt to the persistence of the static hardware state.

Step 3: Hierarchical Entropy Expansion The system employs a hierarchical derivation architecture to generate isolated randomness for dynamic attributes. Utilising HKDF-SHA256, a unique blinding factor r_i is derived for each attribute a_i . This process utilises $SEED_{str}$ as the secret salt and incorporates the attribute's unique identifier id_i and its instantaneous value as the context-specific *info* parameter: $r_i = \text{HKDF-Expand}(\text{HKDF-Extract}(0, SEED_{str}), id_i \parallel a_i, L)$. This ensures domain separation, where the observation of a single sub-token provides no computational advantage in predicting the blinding factors of concurrent attributes.

Step 4: Orthogonal Attribute Encryption Each dynamic property is transformed into an orthogonal sub-token through modified homomorphic encryption. By utilising the context-bound blinding factor r_i , the resulting ciphertexts c_i are cryptographically decoupled. This design maintains bit-level reproducibility, that is, identical attribute values on the same hardware yield consistent ciphertexts while preventing cross-attribute correlation.

Step 5: Bound Volatile Hashing To maintain computational efficiency for high-frequency data, volatile attributes $\mathcal{V}$ are processed via salted commitment. The system utilizes $SEED_{hex}$ as a salt for SHA-256 hashing: $H_j = \text{SHA256}(v_j \parallel SEED_{hex})$. This mechanism cryptographically binds volatile observations to the specific device instance, theoretically mitigating cross-device replay attacks. The Paillier encryption is the most expensive part, but is only performed for the dynamic subset D , while volatile attributes V use $O(1)$ hashing.

Step 6: Structural Aggregation and Partial Authentication All sub-tokens are concatenated in a fixed order to produce the full *IdentiToken*. This modular architecture is designed to facilitate granular verification and partial authentication. Verifiers can selectively evaluate sub-tokens, discarding high-variance or noisy attributes without compromising the integrity of the overall similarity measurement or the validity of the underlying hardware claim. This fixed layout allows the verifier to isolate individual attribute sub-tokens for forensic drift analysis, as illustrated in the architecture (Fig. 1).

Token Comparison and Similarity When a new *IdentiToken* is received, the verifier reconstructs the identity parameters (n, g, λ, μ) using the claimant's current static attributes S' . If the static identity is stable ($S' = S$), the verifier can calculate the encrypted difference for any dynamic attribute:

$$\Delta_i = T_i^{(t+1)} \cdot (T_i^{(t)})^{-1} \pmod{n^2} \quad (1)$$

By decrypting Δ_i with the derived private key (λ, μ) , the verifier perfectly quantifies the numeric drift δ_i (e.g., a change in battery level or window resolution) without ever needing the absolute attribute values. A rejection occurs if the hardware state S diverges or if the drift δ_i breaches the defined policy boundaries.

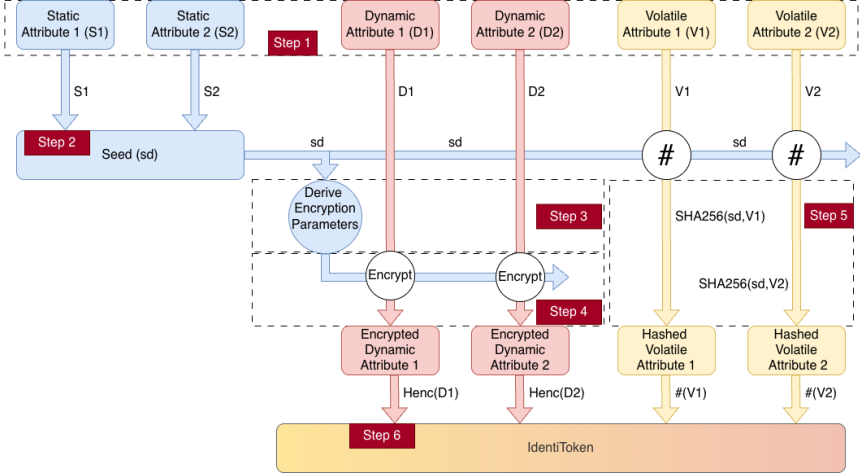


Fig. 1: Structural flow of the IdentiToken calculation using the three kinds of attributes

4.4 Security Analysis

Privacy preservation: The encryption scheme conceals the plaintext attribute value a_i by combining it with entity-specific seed $SEED$, attribute label $label_i$ and a secret key. Without access to the secret, it is computationally infeasible to derive a_i from the ciphertext due to the one-wayness of HMAC and the semantic security inherited from the composite residuosity assumption.

Replay resistance: While the token generation is deterministic for a fixed state, replay resistance is enforced through the integration of high-entropy volatile attributes $\mathcal{V}$ (e.g., session nonces or high-resolution timestamps).

$$H_{fresh} = \text{SHA256}(\text{timestamp} \parallel SEED_{hex}) \quad (2)$$

Furthermore, the system employs sequential drift analysis. Since physical telemetry (e.g., network latency or device memory usage (RAM)) is statistically expected to fluctuate, the repeated presentation of a zero-drift token across sessions serves as a heuristic to detect replayed or simulated identity claims.

Semantic unlinkability: Even if two entities report the same attribute value a_i , their respective ciphertexts remain distinct due to their unique seeds:

$$SEED_1 \neq SEED_2 \Rightarrow \text{Enc}_{SEED_1}(a_i) \neq \text{Enc}_{SEED_2}(a_i)$$

This provides resistance against correlation attacks and ensures user *unlinkability*.

Homomorphic drift evaluation: The additive homomorphic structure enables direct computation of encrypted drift:

$$\Delta = \frac{\text{Enc}(a_{\text{new}})}{\text{Enc}(a_{\text{old}})} \mod n^2 = \text{Enc}(a_{\text{new}} - a_{\text{old}})$$

This allows drift estimation without knowing the original values, enhancing privacy-aware decision-making.

Attribute-context binding: Attribute-specific context is enforced by embedding label_i into the encryption path. For the same value:

$$\text{Enc}(a_i, \text{label}_i) \neq \text{Enc}(a_i, \text{label}_j)$$

$$\text{if } \text{label}_i \neq \text{label}_j$$

ensuring that sub-token misuse across features is invalid.

Tokens are always computed from freshly observed attribute values. No persistent token is stored or reused. This design inherently prevents stale-token or replay-based impersonation.

The proposed tokenisation architecture preserves privacy, supports drift quantification, and resists both passive and active inference attacks. Its deterministic yet non-reversible encryption and contextual binding make it suitable for identity-sensitive IoT and access control systems.

5 Discussion

Our proposed enhancement to IdentiToken introduces a deterministic homomorphic encryption mechanism that maintains the privacy of individual attributes while enabling verifiable drift detection. One of the main benefits is the ability to recompute a sub-token on demand without storing the original value. This is especially useful in constrained or distributed systems where historical data may not be preserved.

Furthermore, by leveraging attribute-context binding through HMACs, we restrict token reuse and ensure that encrypted values remain tied to specific semantics. This results in improved resistance against replay and injection attacks.

The capability to calculate drift as an encrypted delta, and then decrypt only this difference, enables fine-grained monitoring of identity evolution. This is vital for environments such as industrial IoT, where slight changes in parameters can indicate emerging threats or behavioural anomalies.

Unlike earlier schemes that rely on static token comparisons (e.g., via Hamming distance), our construction allows quantitative analysis of the change magnitude. This opens the

door for threshold-based access control, where minor variations are tolerated while major deviations are flagged.

Compared to traditional Multi-Factor Authentication (MFA), which operates in a binary accept/reject manner based on independent checks, IdentiToken offers a vectorised, holistic view of identity. Each dynamic and volatile attribute contributes a verifiable sub-token, which, when compared across sessions, can serve as— authentication token via sub-token similarity scores; system fingerprint: unique per device or user, aiding in entity attestation; session integrity verification: detecting drift for session-based anomaly detection; access controller: enabling graded access based on partial similarity; malware detection: detecting atypical volatile behavior without revealing sensitive data.

A core contribution of this work is the ability to perform stateful authentication without state exposure. In traditional similarity matching approaches, the verifier must store or receive a plaintext baseline of the device’s telemetry to calculate a distance metric (e.g., Euclidean or Cosine distance). This creates a privacy leak, as absolute telemetry values can be used to fingerprint user behaviour or infer environmental conditions. By utilising the additive homomorphic property of the Paillier-based construction, the verifier performs a blind comparison. The verifier computes the encrypted difference Δ_i : $\Delta_i = T_i^{(t+1)} \cdot (T_i^{(t)})^{-1} \pmod{n^2}$.

Upon decryption, the verifier recovers only the relative shift δ_i . This allows the system to confirm that a device’s physical state has shifted by a permissible epsilon ($\delta_i < \tau$) without ever gaining access to the absolute telemetry values. Consequently, the IdentiToken provides a high-assurance identity claim while maintaining a zero-disclosure policy regarding the device’s internal physical state.

The use of HKDF-SHA256 for blinding factor derivation ensures that even if two attributes share the same numeric value (e.g., Temperature = 40, Volume = 40), their sub-tokens remain cryptographically independent. This domain separation is crucial for preventing cross-attribute correlation attacks, ensuring that a compromise of one sub-token does not degrade the security of the entire identity vector.

Despite its benefits, the proposed scheme has certain limitations. First, the resource overhead in the case of large token size in scenarios where there is a high number of attributes, especially due to the use of homomorphic encryption. While Paillier ciphertexts are larger than traditional hashes ($2 \cdot \log_2(n)$ bits), the IdentiToken mitigates transmission overhead through its hybrid structural design. Only the dynamic attributes $\mathcal{D}$ requiring drift analysis are encrypted. High-variance volatile attribute V is hashed, resulting in a typical total token size of approximately 1.5–2.5 KB for an average IoT profile. The increased packet size is offset by the elimination of server-side state. The verifier does not need to store baseline identity vectors; instead, the state is carried securely within the token. This shifts the burden from costly database lookups ($O(\log N)$ or $O(N)$) to constant-time ($O(1)$) cryptographic derivations, enhancing overall system scalability.

Second, the encryption cost, although minimised through determinism, still involves modular exponentiation and HMAC operations, which may not be lightweight for ultra-constrained devices. Third, the current scheme only supports additive comparisons, making it less suitable for attributes where relative ratios matter more than absolute differences. Lastly, we do not perform security analysis on the security of the modified homomorphic encryption, as for generating the token, our primary aim is anonymity and irreversibility rather than security.

Future work includes broader experimentation of the token across diverse device sets, a detailed security analysis following the integration of homomorphic encryption, and formal verification of the authentication protocol using BAN (Burrows–Abadi–Needham) logic [Si04].

6 Conclusion

In this paper, we presented a modified identity tokenisation scheme that extends the original *IdentiToken* algorithm with a deterministic homomorphic encryption layer. This enhancement enables encrypted dynamic and volatile attribute encoding that is both context-aware and privacy-preserving. By deriving a deterministic blinding factor from static seeds, attribute values and semantic labels, we ensure that each sub-token is recomputable, unlinkable across entities and resilient against replay and substitution attacks.

Our approach maintains the additive homomorphic property necessary for quantifying drift, allowing real-time comparison of updated tokens without revealing the underlying values. Compared to traditional MFA systems that perform isolated and binary factor verification, our method offers a unified and continuous authentication through structured identity vectors. This capability enables not only secure authentication but also supports advanced use cases such as session anomaly detection, graded access control and behaviour-based malware detection.

Future research will explore normalisation techniques for fair similarity computation across heterogeneous attribute types and further formalise the use of drift thresholds for adaptive security policies. The evolution of this work will involve moving toward a more formal verification of the underlying mechanisms. Defining these security bounds will provide a deeper validation of the system's robustness when subjected to various environmental and adversarial stressors.

References

- [Bi14] Birgisson, A. et al.: Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud. In: NDSS Symposium. 2014, 10.14722/ndss.2014.23212.

- [Fe23] Fereidooni, H. et al.: AuthentiSense: A Scalable Behavioral Biometrics Authentication Scheme using Few-Shot Learning for Mobile Platforms, 2023, arXiv: 2302.02740 [cs.CR], <https://arxiv.org/abs/2302.02740>.
- [Gr17] Grassi, P. A. et al.: Digital Identity Guidelines: Authentication and Lifecycle Management, tech. rep. NIST SP 800-63B, 10.6028/NIST.SP.800-63B-4, NIST, 2017.
- [Ha12] Hardt, D.: The OAuth 2.0 Authorization Framework, RFC 6749, 2012, datatracker.ietf.org/doc/html/rfc6749.
- [HLC21] Honkaranta, A.; Leppänen, T.; Costin, A.: Towards practical cybersecurity mapping of stride and cwe—a multi-perspective approach. In: 2021 29th Conference of Open Innovations Association (FRUCT). IEEE, pp. 150–159, 2021, 10.23919/FRUCT52173.2021.9435453.
- [IB24] IBM Newsroom: IBM Report: Identity Comes Under Attack, Straining Enterprises' Recovery Time from Breaches, <https://newsroom.ibm.com/2024-02-21-IBM-Report-Identity-Comes-Under-Attack,-Straining-Enterprises-Recovery-Time-from-Breaches>. [Accessed: Aug. 31, 2024], 2024.
- [IS20] ISO: iso/iec/ieee 8802-1ar:2020(en) telecommunications and exchange between information technology systems -2020, <https://www.iso.org/obp/ui#iso:std:iso-iec-ieee:8802-1AR:ed-2:v1:en:term:3.2>, 2020.
- [KBM15] Kiennert, C.; Bouzefrane, S.; Mostefa, A. F. B.: Digital Identity in Cloud Computing. In: Digital Identity Management. Elsevier, pp. 207–244, 2015, 10.1016/B978-1-78548-004-1.50005-5.
- [Kr10] Krawczyk, H.: Cryptographic extraction and key derivation: The HKDF scheme. In: Annual Cryptology Conference. Springer, pp. 631–648, 2010, <https://eprint.iacr.org/2010/264>.
- [Mi22] Mitre.org: CVE-2022-42731: Replay Attack in Django MFA2 WebAuthn Implementation, 2022, nvd.nist.gov/vuln/detail/CVE-2022-42731.
- [PP99] Paillier, P.; Pointcheval, D.: Efficient public-key cryptosystems provably secure against active adversaries. In: International conference on the theory and application of cryptology and information security. Springer, pp. 165–179, 1999, 10.1007/978-3-540-48000-6_14.
- [Si04] Sierra, J. M. et al.: Validating the Use of BAN LOGIC. In: International conference on computational science and its applications. Springer, pp. 851–858, 2004, 10.1007/978-3-540-24707-4_98.
- [Te24] Team, N. S.: The 2024 IoT Security Landscape Report, 2024, netgear.com/hub/network/2024-iot-threat-report.
- [TS24] Tripathi, S. S.; Skwarek, V.: Keystroke dynamics as challenge-response pair for attribute-based authentication using FaR Hash Token. In: 2024 4th Intelligent Cybersecurity Conference (ICSC). IEEE, IEEE, Spain, pp. 48–57, 2024, 10.1109%2FICSC63108.2024.10895813.
- [W319] W3C: Web Authentication: An API for accessing Public Key Credentials Level 1, <https://www.w3.org/TR/webauthn/>, 2019.
- [Wa21] Wang, X. et al.: Attacks and defenses in user authentication systems: A survey. Journal of Network and Computer Applications 188, p. 103080, 2021, 10.1016/j.jnca.2021.103080.
- [Zh19] Zhang, J. et al.: Physical layer security for the Internet of Things: Authentication and key generation. IEEE Wireless Communications 26(5), pp. 92–98, 2019, 10.1109/MWC.2019.1800455.