Enhancing SPIFFE/SPIRE Environment with a Nested Security Token Model

Henrique Z. Cochak[1][a], Milton P. Pagliuso Neto[1][b], Charles C. Miers[1][c], Marco A. Marques[2][d] and Marcos A. Simplicio Jr.[2][e]

[1]*Graduate Program in Applied Computing (PPGCAP), Santa Catarina State University (UDESC), Brazil*

[2]*Laboratory of Computer Networks and Architecture (LARC), Universidade de São Paulo (USP), Brazil*

Keywords: SPIFFE, SPIRE, Nested Token, Token Chaining.

Abstract: Within the domains of authentication, authorization, and accounting, vulnerabilities often arise, posing significant challenges due to the inter-connectivity and communication among various system components. Addressing these threats, SPIFFE framework emerges as a robust solution tailored for workloads identity management. This work explores solutions for use cases not originally foreseen in the SPIFFE scope, focusing on enhancing security measures, particularly investigating a novel token model that introduces a nesting concept. This extended token model operates within a SPIRE environment, enabling token nesting with new features such as token tracing with both ephemeral and non-ephemeral keys and the possibility of delegated assertions.

1 INTRODUCTION

Cloud computing applications are usually composed of multiple workloads, requiring mutual authentication in authorization decisions based on the original caller, their context, and the actions of other workloads over the same transaction. This scenario has the insurance of the security through robust authentication and authorization processes. While authentication employs different methods (e.g., security tokens, certificated-based authentication) to verify the identity of users or systems accessing resources, authorization determines the permissions and actions allowed for authenticated entities, specifying who can access what and what operations they can perform.

One established authentication solution is Secure Production Identity Framework for Everyone (SPIFFE), which focuses on workload identity management (SPIFFE, 2023), ensuring (among other benefits) verifiable mutual Transport Layer Security (mTLS) connections between services (Feldman et al., 2020). However, the current scope of SPIFFE centers on workload identities without inherent support for end-user authentication, unlike protocols such as OAuth 2.0, SAML, or OpenID Connect. Also, its main security document, the SPIFFE Verifiable Identity Document (SVID), is signed by a SPIFFE trust domain authority without support for extension capabilities that could allow a new range of functionalities (e.g., delegation, attenuation, traceability). Delegation is a typical use case where a principal (i.e., an end-user or a workload) allows an authenticated third party to perform tasks on any required service on its behalf. Attenuation, is the ability to restrict the token permissions. This way, the token extension emerges as a practical mechanism to address these functionalities. Finally, traceability allows to track the path taken by the token throughout the request.

We propose a new token model with support for extension and alternative digital signature schemes, facilitating the implementation of new functionalities. This article is organized as follows. Section 2 introduces the basic concepts utilized around our security document. Section 3 delimits the problem to be addressed, as well as the functional and non-functional requirements assumed for the scenario. Section 4 presents the related work found in the literature and the corporate environment. Section 5 details the proposed solution, showing how it can address the problem of traceability in containerized environments. Section 6 describes a proof-of-concept implementa-

[a] https://orcid.org/0009-0007-3571-1709
[b] https://orcid.org/0009-0000-5206-3773
[c] https://orcid.org/0000-0002-1976-0478
[d] https://orcid.org/0000-0001-5800-8927
[e] https://orcid.org/0000-0001-5227-7165

Cochak, H., Neto, M., Miers, C., Marques, M. and Simplicio Jr., M.
Enhancing SPIFFE/SPIRE Environment with a Nested Security Token Model.
DOI: 10.5220/0012634400003711
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 14th International Conference on Cloud Computing and Services Science (CLOSER 2024)*, pages 184-191
ISBN: 978-989-758-701-6; ISSN: 2184-5042
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

tion of the solution. Section 7 presents the test performed and the results obtained.

2 BACKGROUND

Security mechanisms that define and coordinate identity and access management are critical to ensure that only authenticated principals (e.g., users, workloads) are authorized to access the target resource in the intended manner and with the least privilege. SPIFFE is a set of open-source specifications for bootstrapping and issuing short-lived, cryptographically verifiable identity documents to workloads operating across heterogeneous environments and organizational boundaries (SPIFFE, 2023). A workload is a running instance of software executing for a specific purpose. It can own an SVID that uniquely identifies it using a Uniform Resource Identifier (URI) containing two parts: a trust domain name and a unique workload identifier within that domain. Each trust domain contains a root of trust in a construction similar to a PKI, provisioning workloads with a verifiable identity document.

An SVID can be encoded as an X.509 certificate (Boeyen et al., 2008) or as a JSON Web Token (JWT) (Jones et al., 2015) and is digitally signed by an authority within the corresponding trust domain. Whichever the case, the SVID issuance procedure typically involves an attestation process, so only authorized workloads can obtain valid SVIDs via the specified API calls. Workloads can then use these identity documents when authenticating to each other, e.g., by establishing a authenticated mTLS connection with X.509 SVID or exchanging signed JWT-SVID within secure communication channels (SPIFFE, 2023). Even though X.509 SVID is preferred when mTLS connections can be established, JWT-SVID are important when identities need to go through multiple hops (SPIFFE, 2023).

The term identity management explores the notion of identity, and refers to a set of policies, tools, and mechanisms used to manage the life cycle of digital identities associated with participants in a system, which includes individuals, software or hardware components. Tokens, on the other hand, are signed documents carrying assertions used for various security purposes, including authentication, authorization, and access control (Aboba and Wood, 2003). As defined in (Campbell et al., 2015), an assertion is a statement or declaration made by an entity (the claimant), encoded and represented in a specific format. It could encompass various types of information, such as facts, or statements made by the issuer, and is typically related to an identified subject or entity.

3 PROBLEM DEFINITION

The crescent interest in distributed and signed assertions and new token improvements led to the creation of the “SPIFFE - Assertions and Tokens Workgroup”, a workgroup in the SPIFFE community (Slack, 2023) where the primary goal is to establish a comprehensive framework allowing identified entities to make authenticated statements in various scenarios. One of the points recognized as relevant in workgroup discussions deals with the limitation of SVID to workload identification only, without the ability to identify or integrate with an end-user identity. JWT-SVID is a promising alternative in cloud computing, but it still lacks the flexibility of extension with additional information, a relevant feature in distributed scenarios. The required solution must support various approaches to identity, since scenarios where the claimant’s identity doesn’t require validation from a trusted authority to instances involving anonymity or less structured identification methods, approaches included: X.509, JWT, JSON-based methods, and even situations in which the claimant’s identity remains unidentified.

Another crucial aspect revolves around methods for token extensions. There is a strong interest in empowering token bearers with the flexibility to extend tokens as needed. This capability enhances security in access management, enabling delegation, attenuation, or token sealing, ensuring a more adaptable and fine-granular access control system. The main challenges include handling multiple arbitrary claims within tokens and highlighting the importance of size sensitivity to streamline signing and verification processes.

These definitions led to the *Functional Requirements* (FRs) guiding the development of the proposed solution, setting it apart within the domain of similar research. There are four FRs: delegation of bearer token permissions through a SPIRE environment (*FR1*), possibility to create new valid tokens from previous one while restricting access rights with attenuation (*FR2*), existence of mechanisms to validate the token chain of custody (*FR3*), and possibility to add arbitrary information to a token, allowing for a more granular authentication/authorization mechanism (*FR4*).

4 RELATED WORK

Table 1 offers a succinct overview of connections among similar works, categorized according to the

Table 1: Related work.

	(Biscuits, 2023)	(Richer, 2023)	(SPIFFE, 2024)	(Peterson, 2021)
FR1	No	No	No	No
FR2	Yes	No	No	Yes
FR3	Yes	Yes	No	Yes
FR4	Yes	Yes	No	Yes

predefined FRs outlined in Section 3. Detailed discussions on each paper will follow below, providing a comprehensive examination of their respective contributions and alignment with the identified requirements. The term *Partial* signifies that, based on the documentation, the entity can fulfill some aspects of the FR, but falls short of meeting all the specified requirements.

Biscuits (Biscuits, 2023) employs public key cryptography to create extensible tokens. The foundation of Biscuit's token creation lies in the Ed25519 algorithm, where the private key is required for token generation and extension while the public key is used in token validation. The token structure is a sequence of signed blocks, where the signature generation of the first block B_0 uses a root private key. Before creating a new block B_i, it requires the generation of a new key pair (sk_{i+1}, pk_{i+1}) that must be used to sign the next block B_{i+1}. In this scenario, the public key pk_{i+1} should be part of the data, and the private key sk_{i+1} must be forwarded to the intended recipient through a secure channel.

Our nested token model takes inspiration from some concepts provided in this work, like decentralized verification and offline attenuation using Schnorr signatures. Their main structure is a block, while ours is a token with a payload and signature. To create valid new blocks based on an existing one (*FR2*), Biscuits copies all the blocks, gets the private key from the proof received in the secure channel, and generates a new random key pair. Then, it signs the data containing the new public key using the private key from the previous token. The nested token proposes two different alternatives to improve the scheme efficiency: (i) it leverages an existing Identity Provider (IdP) to avoid the key generation step, or (ii) it employs a signature aggregation mechanism that not only does not require the key generation but also reduces the token signature size.

(Richer, 2023) proposes a data structure tailored for managing transactional state information in complex API deployments involving interconnected services. This proposed data structure comprises single-element containers called a Bucket and a container to hold multiple such elements cohesively called a Crate. Single elements encapsulate tokens with specific properties. Although similar, it works with bearer tokens but solves different security demands. As this proposal works in a non-linear fashion and can reorder the dictionary of multiple single-element containers, it is impossible to trace the path of the bearer token more linearly.

(SPIFFE, 2024) JWT-SVID is one of the security tokens in the SPIFFE specification. As a modified form of standard JWT tokens, JWT-SVID implements restrictions to enhance security. The scope of this token is to solve difficulties associated with asserting identity across Layer 7 boundaries, making compatibility with existing applications and libraries a core requirement. Thus, it does not provide mechanics of end-user delegation, token extension, or tracing, lacking the functionalities desired by the community and listed in the *FRs*, which allows a more flexible token solution in the SPIFFE ecosystem.

(Peterson, 2021) explore the extension to the Personal Assertion Token (PASSporT) (Wendt and Peterson, 2018), a token format based on JWT for conveying cryptographically signed information about the people involved in personal communications. This scheme introduces a concept similar to nested tokens within the SIP (Session Initiation Protocol) communication framework. The PASSporT format, designed for conveying authenticated information in real-time communications, explicitly indicates call diversion, which is vital in scenarios involving changes to the original call destination, often seen in SIP re-targeting situations. Our work and PASSporT have many resemblances as both use notions of token nesting within a JWT domain, although each solves a different issue: while we focus on end-user and workload identity, the latter work is related to the SIP framework.

In this research about diverse token structures and methodologies, we've explored innovative approaches like Biscuits' attenuation mechanism, Richer-Wimse's transactional state management, OAuth identity continuity across trust domains, and the extension of PASSporT tokens for SIP communications. Each approach brings unique facets to tokenization, from creating new tokens through attenuation to managing transactional state seamlessly across distributed systems and maintaining identity context across trust boundaries. These advancements strengthen security measures and enhance the interoperability of interactions within complex networked environments, showcasing the evolving landscape of token-based mechanisms in current data handling and authentication realms.

5 PROPOSED SOLUTION

This work proposes the Nested Token model: a token construction that implements decentralized and local token creation and extension with support to authenticated statements. It focuses on lightweight, size-sensitive operations while ensuring robustness in authentication and validation processes. The fundamental innovation lies in using incremental signing to implement the extension mechanism, allowing the grouping of multiple signed sets of claims within a single token instance. This work also presents a proof-of-concept implementation, detailed in Section 6, that uses the nested token model to meet the functional requirements defined by the SPIFFE community and listed in Section 3.

The solution supports different signature schemes. Although not restricted to them, the proof-of-concept presents two alternatives: the ID-Mode and the Anonymous Mode. The ID-Mode leverages an existing IdP, using trusted identity documents and corresponding keys to implement authentication and authorization mechanisms, and the Anonymous Mode uses a Schnorr signature aggregation mechanism to offer an alternative to scenarios where an IdP is not available or where the token size is critical. Our design lies in supporting lightweight identity documents and pseudonyms while ensuring efficient handling of information. Throughout this document and our proof-of-concept implementation, the JavaScript Object Notation (JSON) format (Jones et al., 2015) is assumed as default, with the token adopting a JWT-based structure, being the payload and signature encoded using Base64 and a URL and filename safe alphabet (Josefsson, 2006). A high-level visual representation of the nested token architecture is in Figure 1.

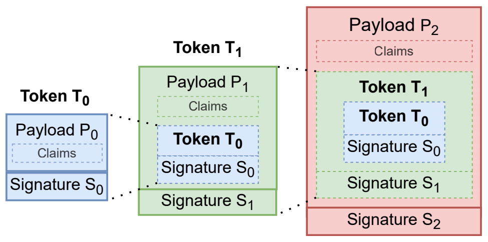

Figure 1: Nested token architecture.

In Figure 1, a workload W_0 creates a new token T_0, containing a payload P_0 and a corresponding signature S_0. It forwards the token to W_1, which extends it by creating a new token T_1, with a payload P_1 that must encompass T_0 and a new set of claims and the corresponding signature S_1. This recursive construction allows any workload to create or extend an existing token to include new signed claims. It enables several use cases where information integrity and rastreability are desirable (e.g., authentication, authorization, monitoring, auditing), detecting any potential tampering in its route, all related to the chained link between a group of specific claims, as performed by ID-Mode and depicted in Section 5.1 or directly through a signature concatenation scheme, as done in Anonymous mode and described in Section 5.2. Both use cases demonstrate a practical application and the effectiveness of the proposed solution in addressing the need for comprehensive token tracking and integrity validation in complex environments.

When considering the *FRs*, both scenarios allow token creation and extension with specific permissions based on authenticated statements within a SPIRE structure. Regarding *FR2*, the token scheme facilitates controlled privilege reduction or the addition of constraints through attenuation during token propagation across various workloads. The solution also meets the *FR3* by integrating mechanisms for validating the token chain of custody. The proposed solution enables comprehensive validation and tracking of the token's path using sequential signatures, ensuring integrity as it traverses different environments. Lastly, in meeting *FR4's*, the nested token scheme's flexibility enables the inclusion of context-specific data. This feature refines authentication and authorization decisions by accommodating diverse information within the token structure.

5.1 ID-Mode

The use case for this schema requires a trusted IdP that enables workloads to obtain a valid identity document associated with their signature keys (e.g., a regular X.509 certificate or X.509 SVID). Furthermore, each workload W_n along the path knows the identity of the next workload W_{n+1}, to which requests and tokens should be forwarded. In this scenario, the workload uses its private key to create the token signature, thus allowing its creation or extension in a non-repudiable manner, as illustrated in Fig. 2.

A straightforward approach would consist in W_n placing its whole verifiable identity document or its public key as the "issuer" claim in the payload before conveying it to W_{n+1}, that must be identified in the

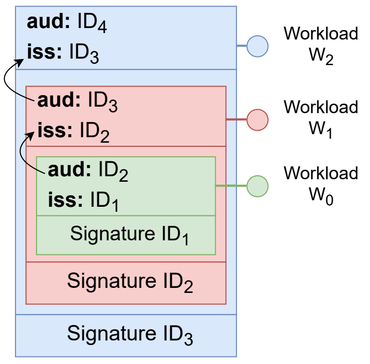

Figure 2: Claims chained link.

"audience" claim, therefore creating a chained link between workload hops. This approach would ensure a specific sequence of token signatures and identities that the receiver can easily verify.

5.2 Anonymous-Mode

This scheme adopts a signature aggregation scheme that focuses on granting the validation of a specific sequence of signatures rather than identifying the signers. Thus, it uses ephemeral keys and does not require an IdP. Also, it improves the efficiency by reducing the token size, and, different from Biscuits, it does not require a key pair generation when extending the token. This approach adopts the (Galindo and Garcia, 2009) Identity Based Signature (IBS) scheme to build a signature aggregation scheme using Edwards-curve Digital Signature Algorithm (EdDSA), adapting the Biscuits security model (Biscuits, 2023) to implement a token scheme with aggregated signatures, where the total signature size can be reduced up to 50%, facing the ID-mode. A high-level visual representation of this concept is shown in Fig. 3.

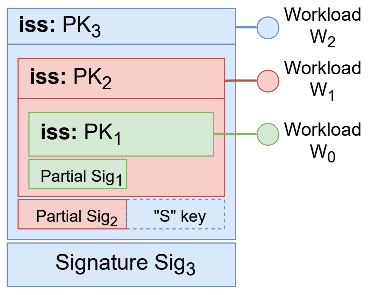

Figure 3: Anon-Mode signature chain link.

The process starts with the creation of a new token T_0 by an issuer (workload W_0), using the root private key sk_0. The payload of T_0 must, at least, contain the issuer claim (iss) filled with the root public key pk_0. It is signed using the Schnorr scheme and produces a signature $\sigma_0 = (R_0, s_0)$. The resulting token consists of a payload concatenated with a signature. Following the Galindo-Garcia scheme (Galindo and Garcia, 2009), when the workload W_1 wants to extend T_0 it needs to extract the aggregation key (s_0) to use it as the private key to sign the new token T_1, that will contain in its payload the token T_0, where the signature $\sigma_0 = (R_0)$ and $\sigma_1 = (R_1, s_1)$. This approach is repeatable as many times as needed, with multiple concatenations where only the last signature is complete (i.e., preserves both the "R" and "s" parts), while all previous are partial signatures containing just the "R" part.

6 IMPLEMENTATION AND TESTBED

As discussed in Section 3, the usage context of the token and its ability to support different digital signature schemes led to the development of two different models for the nesting and validation flow to test the token flexibility in different scenarios. For a SPIRE scenario under the ID-Mode use case, a previously SPIRE authenticated workload uses its private key to produce an Elliptic Curve Digital Signature Algorithm (ECDSA) digital signature with the corresponding public key available in its identity document (i.e., SVID), ensuring that the assertion token is verifiable during your workflow, as depicted on Fig. 4.

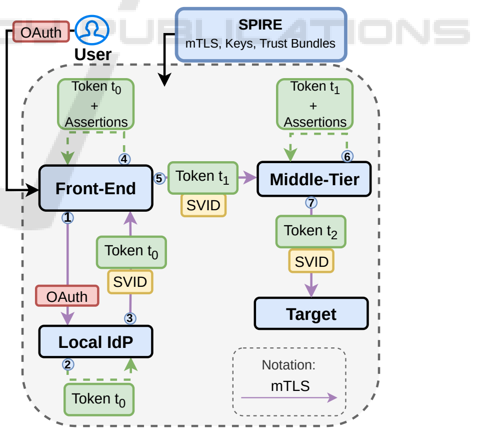

Figure 4: ID-Mode proof of concept.

The validation scheme for a token T_i using ID-mode has three steps executed recursively for every signer. The validation begins by checking if the identity document belongs to the token signer identified in the issuer ("iss") claim. Then, it checks if the token T_i issuer ("iss") value is the same as the audience ("aud") value in T_{i-1}, granting the link between signers. Finally, using the public key in the identity document, the corresponding signature is verified over the token payload.

The Anonymous-Mode use case is similar to Fig. 4. The main distinction lies in the signature concatenation scheme that reduces the token size while performing the key and signature generation in a single operation, not relying on any IdP. Consequently, the secret key required to extend the token does not have to be created and forwarded in a secure channel, as the token already contains it. The key is removed when used, avoiding the reuse by any subsequent signer and eliminating the dependency on external authentication entities. In the case of Fig. 4, there is no SVID to forward. In the proof-of-concept, we opted for executing the token validation process at the resource server, ensuring that the verification occurs after all necessary token transmissions. This final validation step offers an optimized approach, minimizing computational overhead and simplifying the authentication procedure across the application.

The experiment was conducted through on a server machine with the following specifications: GNU/Linux Ubuntu Server operating system (version 20.04.6), 192GB of RAM, 4TB of HDD storage, and CPU model Intel(R) Xeon(R) CPU E5-2620 2.5Ghz with 24 cores. Each containerized workload is limited to a CPU core and 128MB of RAM. Data scrapping from Prometheus occurs at intervals of 50 milliseconds. This interval maintains a balance, ensuring Prometheus collects updates frequently enough for a near-real-time system view without overloading the monitored systems with unnecessary data.

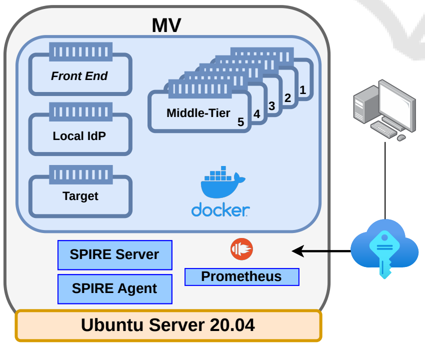

Figure 5: Testbed environment.

A set of metrics was chosen to evaluate the solution performance by measuring the effects of the nested token's extension, namely: (i) the token's size growth (through the nesting on different hops of a request flow inside the PoC environment), (ii) execution time of the creation, extension, and validation processes on all the components and (iii) the consumption of usage of computational resources (CPU and RAM).

7 RESULT ANALYSIS

The experiment individually collected the performance metrics for each proof-of-concept component. The resulting data was grouped by schematic (ID-Mode and Anon-Mode) and the metrics set (computational resources, token size growth, and execution time). Primarily, Fig. 6 explores the token growth when the concatenation happens on the nesting for each workload hop.

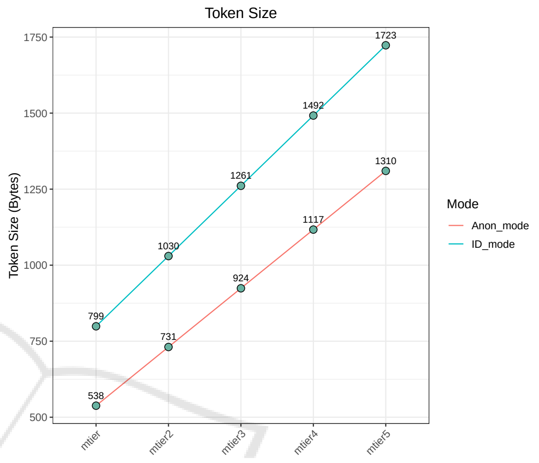

Figure 6: Token size growth.

For this benchmark, only the mandatory set of claims is included in the payload, illustrating the minimum token growth for each mode. This way, every extension results in a linear growth in the token, composed of the payload and signature sizes. The comparison between Anon-Mode and ID-Mode reveals a notably reduced token growth within the former. ID-Mode grows 231 bytes for each extension, while Anon-Mode only 193 bytes and it is related to the concatenation scheme in which the secret key is extracted from the last token signature, reducing its size, as explained in Section 5.2. Consequently, the final token size is bound to be smaller because it contains a smaller signature size. Am important note is Table 6 does not illustrate the certificate growth related to the SVID token on the ID-Mode. This means the actual communication cost for the ID-Mode is the token size plus the SVID document size, emphasizing a better choice overall for the Anon-Mode.

When observing the execution time on Table 2, the results exemplify similar behavior for all workloads. A higher standard deviation is observed on the Front-End workload, most likely due to the variable user load, depending on the caller.

Still on Table 2, the validation results show an increase in the execution time as the token is nested more times. In essence, more extensions result in

Table 2: Execution time cost for ID-Mode.

ID-Mode	
Components	**Token Minting cost (μs)**
Front-end	217,85 ± 347,56
Middle-Tier$_1$	187,26 ± 34,39
Middle-Tier$_2$	189,86 ± 41,57
Middle-Tier$_3$	189,69 ± 36,34
Middle-Tier$_4$	188,54 ± 36,60
Middle-Tier$_5$	196,26 ± 38,60
Components	**Validation cost (μs)**
Front-end	351,23 ± 84,56
Middle-Tier$_1$	708,39 ± 166,30
Middle-Tier$_2$	1067,33 ± 195,27
Middle-Tier$_3$	1482,79 ± 352,94
Middle-Tier$_4$	1800,60 ± 376,99
Middle-Tier$_5$	2209,10 ± 466,36
Target	2554,14 ± 472,53

a more complex validation process with more signatures and, consequently, a higher execution time and resource consumption.

The execution time for Anon-Mode shown in Table 3 illustrates similar results for the minting function in all workloads. The minting process has an increased time when compared to ID-Mode because of the key extraction process required before signing the new token.

Table 3: Execution time cost for Anon-Mode.

Anon-Mode	
Components	**Token Minting cost (μs)**
Front-end	1007,87 ± 249,24
Middle-Tier$_1$	992,67 ± 234,95
Middle-Tier$_2$	1015,01 ± 246,86
Middle-Tier$_3$	995,63 ± 234,02
Middle-Tier$_4$	1030,39 ± 325,15
Middle-Tier$_5$	1039,25 ± 302,71
Components	**Validation cost (μs)**
Target	4616,95 ± 1012,38

The validation process evaluation shows that using concatenated signatures is costly, especially compared to the usual approach used by ID-Mode. As the signature aggregation scheme adopted by Anonymous mode requires a recursive computation of the public key used in the signature validation, its execution time was higher than in ID-mode. The computational consumption (Tables 4 and 5) exhibits low usage of resources all around. The CPU usage (Table 4) was similar across all components, whether the operation was idling or under load (subjected to batches of requests). A slight increase can be perceived in the ID-Mode results when under load, and it is due to the validation scheme adopted that verifies the whole set of identity documents of all workloads that extended the token.

Table 4: CPU consumption comparison.

Workload	Idle (%)	Anon-Mode (%)	ID-Mode (%)
Front-End	14,3 ± 4,0	26,3 ± 5,4	28,7 ± 1,7
Local IdP/TTP	14,4 ± 4,0	26,3 ± 5,4	28,8 ± 4,0
Middle-Tier$_1$	14,3 ± 4,0	26,3 ± 5,3	28,8 ± 4,0
Middle-Tier$_2$	14,3 ± 4,0	26,2 ± 5,4	28,8 ± 4,0
Middle-Tier$_3$	14,3 ± 4,0	26,3 ± 5,3	28,8 ± 3,9
Middle-Tier$_4$	14,3 ± 4,0	26,3 ± 5,3	28,7 ± 4,0
Middle-Tier$_5$	14,3 ± 4,0	26,3 ± 5,4	28,8 ± 3,9
Target	14,7 ± 3,9	26,3 ± 5,3	28,8 ± 4,0

The results of memory consumption (Table 5) show a common trend of increased usage as the extended token is nested. The resource access layer is the component with the highest memory usage, as expected; its final validation purpose and value storage are reasons for this outcome. The overall increase in memory usage is anticipated, accompanying the growth in token size during each nesting and issuance process across the components.

Table 5: Memory consumption comparison.

Workload	Idle (MB)	Anon-Mode (MB)	ID-Mode (MB)
Front-end	13.4 ± 2.3	19.7 ± 3.7	31.0 ± 6.4
Local IdP/TTP	13.6 ± 2.3	18.3 ± 3.4	26.1 ± 5.0
Middle-Tier$_1$	13.6 ± 2.4	22.2 ± 4.7	28.6 ± 5.6
Middle-Tier$_2$	13.6 ± 2.4	22.3 ± 4.7	29.1 ± 5.8
Middle-Tier$_3$	13.4 ± 2.4	22.3 ± 4.7	29.0 ± 5.6
Middle-Tier$_4$	13.4 ± 2.4	22.3 ± 4.7	28.7 ± 5.8
Middle-Tier$_5$	13.4 ± 2.4	22.5 ± 4.8	29.2 ± 5.9
Target	13.9 ± 2.5	25.0 ± 5.7	34.9 ± 7.1

The values in Table 5 illustrate a higher cost on the ID-Mode when comparing Anon-Mode. This happen because to the scheme of storing and redirecting sets of SVID certificates and its validation being present on every component. The nested token exhibited linear growth in its payload, i.e., the effects of extension in its nesting are constrained for each hop. The execution time of processes using the nested token in both models can be considered efficient, with low variation for the issuance process. The use of identity documents as part of the validation scheme directly influences the consumption of computational resources, albeit with less time spent on token validation than the Anon-Mode. The concatenated signatures in the Anon-Mode offer lower resource consumption but a considerably higher validation and execution time when facing ID-mode.

8 CONSIDERATIONS

In this work, we've outlined a comprehensive framework that enhances the SPIFFE ecosystem. The primary focus was to address the current limitations of token usage between workloads. Our solution introduces a token nesting model with two signature schemes that serve as a robust mechanism for tracking the token path and a broader solution to distributed token signing and offline validation, ensuring authenticity and integrity within multi-cloud environments. The ID-Mode relies on established identity documents and signatures issued by a IdP, promoting a well-defined chain of trust between workloads. On the other hand, the Anon-Mode leverages ephemeral keys and IBS implementing a signature concatenation scheme that results in smaller tokens and a signature chain, avoiding dependencies on external authentication entities. These proposed models align with the diverse identity scenarios within the SPIFFE community, prioritizing lightweight and cost-effective approaches. By enabling the creation of authenticated statements and addressing the challenges related to assertions, token extension capabilities, and handling multiple arbitrary claims within tokens, our framework aims to be a valuable tool in secure access management.

For future work, we plan to implement a new scheme (Hybrid Mode) that aggregates ID and Anonymous private keys, resulting in a unique key that binds a workload identity to a specific token. We also foresee the possibility of employing the nested token to create an extensible and lightweight identity document suitable to resource-constrained environments.

ACKNOWLEDGMENTS

This work was supported by Hewlett Packard Enterprise (HPE), and in part by the Brazilian CNPq (grant PQ 304643/2020-3 and 311245/2021-8), FAPESP (grant 2020/09850-0), and CAPES (Finance Code 001). Special thanks for the discussions and contributions to the work: Adriane Cardozo (HPE), Andrew Harding (VMware), Caio Milfont (HPE), Eugene Weiss (Sentima), Evan Gilman (SPIRL), João Ambrosi (HPE), and Yogi Porla (Stealth Startup). This work was funding by FAPESC, UDESC, USP and developed at LabP2D/LARC.

REFERENCES

Aboba, D. B. D. and Wood, J. (2003). Authentication, Authorization and Accounting (AAA) Transport Profile. RFC 3539.

Biscuits (2023). Biscuits cryptography reference. doc.biscuitsec.org/getting-started/introduction. Access in: Nov 01. 2023.

Boeyen, S., Santesson, S., Polk, T., Housley, R., Farrell, S., and Cooper, D. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280.

Campbell, B., Mortimore, C., Jones, M. B., and Goland, Y. Y. (2015). Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants. RFC 7521.

Feldman, D., Fox, E., Gilman, E., Haken, I., Kautz, F., Khan, U., Lambrecht, M., Lum, B., Fayó, A. M., Nesterov, E., Vega, A., and Wardrop, M. (2020). Solving the bottom turtle — a SPIFFE way to establish trust in your infrastructure via universal identity.

Galindo, D. and Garcia, F. D. (2009). A schnorr-like lightweight identity-based signature scheme. In *Progress in Cryptology–AFRICACRYPT 2009*, pages 135–148. Springer.

Jones, M. B., Bradley, J., and Sakimura, N. (2015). JSON Web Token (JWT). RFC 7519.

Josefsson, S. (2006). The Base16, Base32, and Base64 Data Encodings. RFC 4648.

Peterson, J. (2021). Personal Assertion Token (PASSporT) Extension for Diverted Calls. RFC 8946.

Richer, J. (2023). Multi-token Container Data Structure. Internet-Draft draft-richer-wimse-token-container-00, IETF.

Slack (2023). Assertions and tokens workgroup. Online forum post. Accessed on December 26, 2023.

SPIFFE (2023). Spiffe overview. spiffe.io/docs/latest/spiffe-about/overview/. SPIFFE online documentation. Accessed on November 21, 2023.

SPIFFE (2024). The jwt spiffe verifiable identity document. Online forum post. Accessed on February 10, 2023.

Wendt, C. and Peterson, J. (2018). PASSporT: Personal Assertion Token. RFC 8225.