

Performance Evaluation of Self-Sovereign Identity

Emin Adem Buran^{1,2}, Christian Prehofer^{2,1,*} and Srivatsav Chenna^{2,1}

¹Technical University of Munich, Arcisstraße 21, 80333 Munich, Germany

²DENSO AUTOMOTIVE Deutschland GmbH, Eching, Germany

Abstract

Self-Sovereign Identity (SSI) systems provide a decentralized approach to digital identity management, prioritizing privacy, security, and user control. This study evaluates the performance of the Walt.id SSI stack across multiple DID methods (CHEQD, WEB, KEY, JWK) on two hardware platforms: a resource-constrained Raspberry Pi and a standard PC. By systematically analyzing key metrics such as latency, CPU and memory usage, and scalability, this work provides insights into the capabilities of SSI systems across varying configurations. The study aims to contribute to the design and optimization of efficient SSI solutions for diverse real-world applications.

Keywords

Self-sovereign identity (SSI), decentralized identifier (DID), verifiable credential (VC), performance evaluation, scalability, CPU and memory usage, latency

1. Introduction

Self-Sovereign Identity (SSI) represents a transformative approach to digital identity management, addressing critical issues related to privacy, security, and user control [1]. Traditional identity systems often rely on centralized authorities, leaving users vulnerable to data breaches [2, 3]. In contrast, SSI enables individuals to own, manage, and selectively share their credentials, empowering them with autonomy over their personal information. This eliminates the need for intermediary authorities, reducing reliance on potentially insecure or opaque systems.

The motivation for this work stems from the increasing demand for SSI solutions that can operate effectively in diverse environments, including resource-constrained devices. Evaluating the performance of SSI systems in such scenarios is crucial to ensure their practical applicability and adoption. Systems, such as single-board computers like the Raspberry Pi [4], provide a compelling platform for testing the feasibility and performance of SSI technologies. The Raspberry Pi, widely used for prototyping and research, serves as an affordable yet powerful option to simulate resource-constrained environments.

This paper specifically focuses on the Walt.id SSI stack, an open-source identity and wallet infrastructure designed to provide a robust and flexible framework for implementing SSI solutions [5]. Walt.id supports the creation, issuance, and verification of Verifiable Credentials (VCs) [6] and incorporates a variety of Decentralized Identifier (DID) methods [7], including blockchain-based and key-based approaches. Its compliance with W3C standards ensures interoperability across platforms [8], while its support for multiple cryptographic key types [9] and APIs for issuers, verifiers, and wallets enables diverse use cases [6].

To evaluate the Walt.id SSI stack, this study examines its performance under different configurations, including varying DID methods (e.g., CHEQD, KEY). The experiments assess the impact of these configurations on key performance metrics such as latency, CPU and memory usage and scalability, providing valuable insights into how specific settings influence the system's behavior and resilience under load.

TDI 2025: 3rd International Workshop on Trends in Digital Identity, February 3, 2025, Bologna, Italy

*Corresponding author.

✉ adem.buran@tum.de (E. Buran); c.prehofer@eu.denso.com (C. Prehofer); s.chenna@eu.denso.com (S. Chenna)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

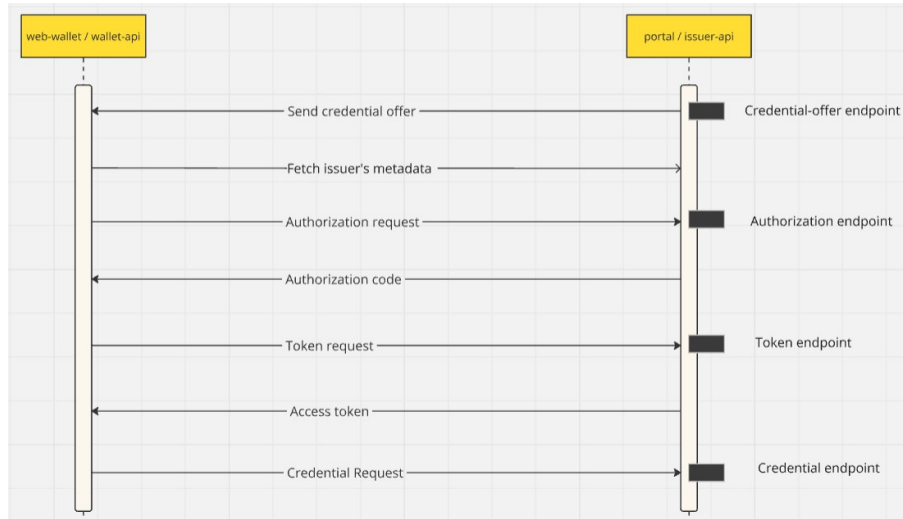


Figure 1: VC issuance flow in OID4VC.

2. Related Work

Two prior studies provide a foundation for evaluating SSI systems and help position our work in the broader context of performance analysis.

The first study [10], evaluates seven SSI frameworks, including four implementations of TCID developed by the authors themselves, as well as Hyperledger Indy [11], uPort [12], and Jolocom [13]. Performance comparisons are based on metrics such as latency, CPU usage, and network traffic, with separate measurements for credential issuance and verification. However, the study lacks analysis of scalability under concurrent user loads, and the specific DID methods or resolver configurations used are not clearly defined.

The second study [14], focuses on Hyperledger Indy, conducting scalability tests with concurrent users ranging from 20 to 120. It evaluates average response times, CPU and memory usage, and error rates during credential issuance and verification [14]. While the study offers detailed insights into system behavior under load, it does not explore the impact of using different DID methods or hardware configurations.

In comparison, we evaluate the Walt.id SSI stack using multiple DID methods (CHEQD, WEB, KEY, JWK) and consider both single-threaded and concurrent operations to analyze scalability. Additionally, our work includes performance measurements on both a PC and a Raspberry Pi.

3. Self-Sovereign Identity Experimental Setup

The experimental setup consists of a Raspberry Pi 4 and a x86 PC to evaluate the performance of the Walt.id identity stack under realistic conditions. The Raspberry Pi 4 is equipped with a quad-core CPU (1.8 GHz max) and 8GB of memory and operates as a more resource-constrained device running Ubuntu Core 24, simulating an embedded environment. In contrast, the PC is equipped with a quad-core CPU (3.4 GHz max) and 16GB of memory, runs Ubuntu 22.04.3 LTS as a control environment for performance comparison. Both devices were connected to the internet via a DSL connection from a home network, providing a practical setting that mirrors typical deployment scenarios outside controlled data center environments.

To create a Self-Sovereign Identity (SSI) environment, we deployed the Walt.id identity repository version 0.3.1 on the Raspberry Pi and the x86 PC. Walt.id offers three key services: Issuer, Verifier, and Wallet. The Issuer is used to create credentials, Verifier is used to validate credentials, and the Wallet is used to store and manage credentials [6].

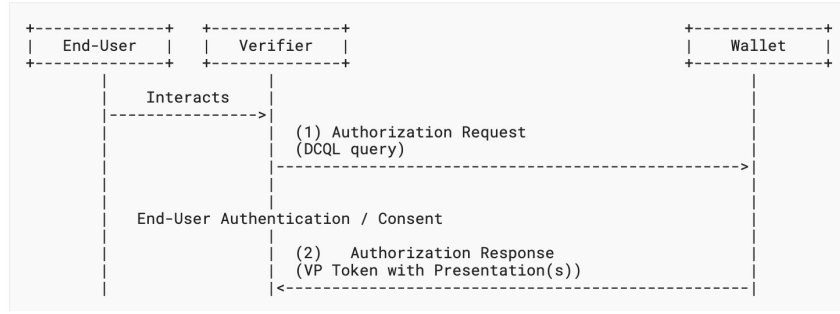


Figure 2: VC verification flow in OID4VC [15].

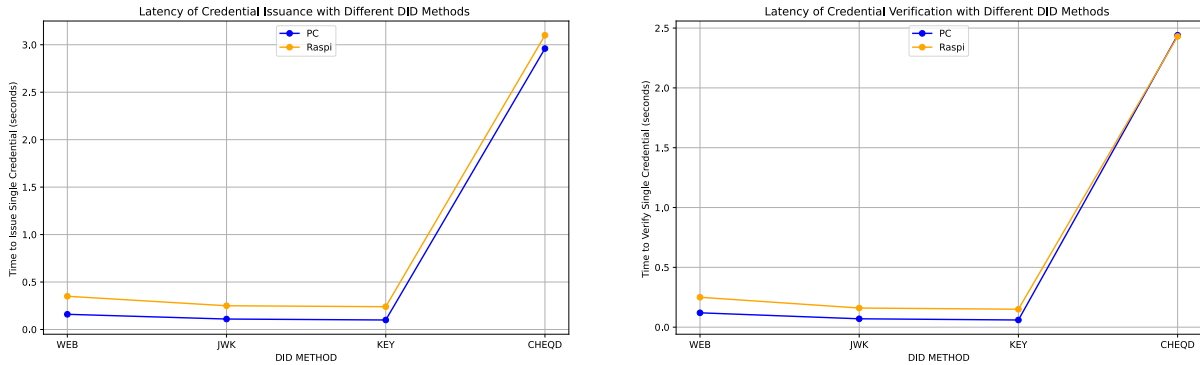


Figure 3: Latency of credential issuance and verification processes with different DID Methods.

Walt.id uses OIDC4VC (OpenID for Verifiable Credentials) as the data exchange protocol in the SSI stack [16]. OIDC4VC provides a standardized way to exchange Verifiable Credentials [17]. These message exchanges during credential issuance and verification introduce additional latency. Figure 1 and Figure 2 show the detailed interactions during the credential issuance and verification, respectively.

To evaluate the system’s performance, Python scripts were developed to automate interactions with the Issuer, Verifier, and the Wallet. The scripts facilitated the systematic measurement of key metrics such as latency, CPU usage, memory usage and scalability.

4. Performance Measurement Methodology and Results Analysis

In this section, the methodology and results of the performance measurements conducted on the Walt.id SSI stack are detailed. Each subsection below provides a detailed explanation of the measurement process for each metric, the results obtained, and factors influencing these results.

4.1. Latency Measurement

This subsection presents the latency measurements for two distinct operations: Verifiable Credential (VC) issuance and VC verification. These measurements were conducted separately on both the Raspberry Pi and the PC, allowing for a comparative analysis of latency across different devices.

To capture the impact of various DID methods on latency, four different DID methods were used during the measurements: CHEQD, WEB, KEY, JWK. We calculated the latency for both issuance and verification processes for all four DID methods and observed how the different DID methods affect the latency.

For accuracy, each VC issuance and verification operation was repeated 50 times. The average latency was calculated from these repeated operations to provide a reliable measure of latency for each DID method and device configuration. It is important to note that all measurements were conducted

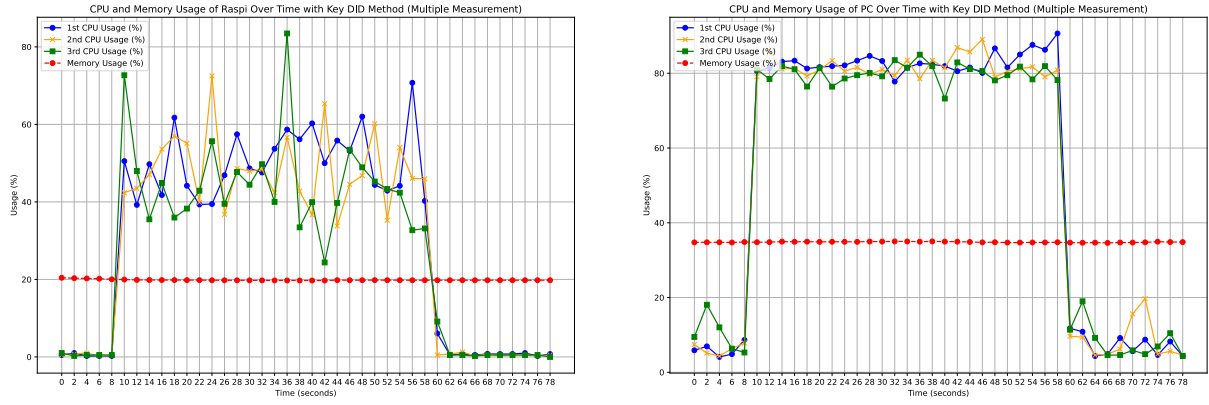


Figure 4: CPU and memory usage of Raspberry Pi 4 and PC over time with KEY DID method.

without concurrent users; each issuance or verification operation was performed sequentially to ensure consistent latency readings. Figure 3 shows the results of the latency measurements.

CHEQD DID Method Latency: Both issuance and verification times are significantly higher for the CHEQD DID method compared to other methods. The WEB, JWK, and KEY DID methods demonstrate consistently low latencies across both devices, with credential issuance and verification times below 0.5 seconds. These results contrast sharply with the CHEQD method, which exhibits significantly higher latencies. This discrepancy is likely due to CHEQD’s reliance on blockchain technology, where DID document resolution requires interaction with an external service, the Universal DID Resolver [18]. Universal DID Resolver is an independent service that, in this case, performs operations on the CHEQD blockchain network to resolve the DID document. For example, on the Raspberry Pi, CHEQD issuance latency is approximately 3.10 seconds, while verification latency is around 2.43 seconds—much higher than the latencies observed for other DID methods.

Raspberry Pi vs. PC: For most DID methods (WEB, JWK, and KEY), latency on the Raspberry Pi is approximately twice as high as on the PC. This difference is likely due to the CPU performance disparity. For instance, with the JWK method, the average issuance latency is around 0.25 seconds on the Raspberry Pi compared to 0.11 seconds on the PC. Similarly, verification latency is 0.16 seconds on the Raspberry Pi, roughly double the 0.07 seconds observed on the PC.

4.2. CPU and Memory Usage Measurement

This subsection evaluates the CPU and memory usage of the Walt.id during Verifiable Credential (VC) issuance and verification using two DID methods: KEY and CHEQD. The measurements were taken over an 80-second interval divided into three phases: an initial idle phase (10 seconds) to observe baseline resource usage, a load phase (60 seconds) where credentials were continuously issued and verified in sequence, and a final idle phase (10 seconds) to observe resource recovery after operations. The focus was on CPU and memory usage during sustained operation, rather than individual performance differences between issuance and verification.

In Figure 4, CPU and memory usage measurements conducted with the KEY DID method on both the Raspberry Pi and the PC are shown. These results are displayed in separate graphs, allowing for a clear comparison of CPU and memory usage between the two systems under the same operational conditions. It is worth noting that during the measurements with the KEY DID method, the experiments were repeated three times for each device under the same conditions. On the Raspberry Pi, a total of 115, 119, and 122 credentials were issued and verified across the three runs within the 60-second load phase. In comparison, on the PC, the system issued and verified 275, 269, and 264 credentials across the different runs. These repeated measurements ensure the reliability of the results and provide a robust

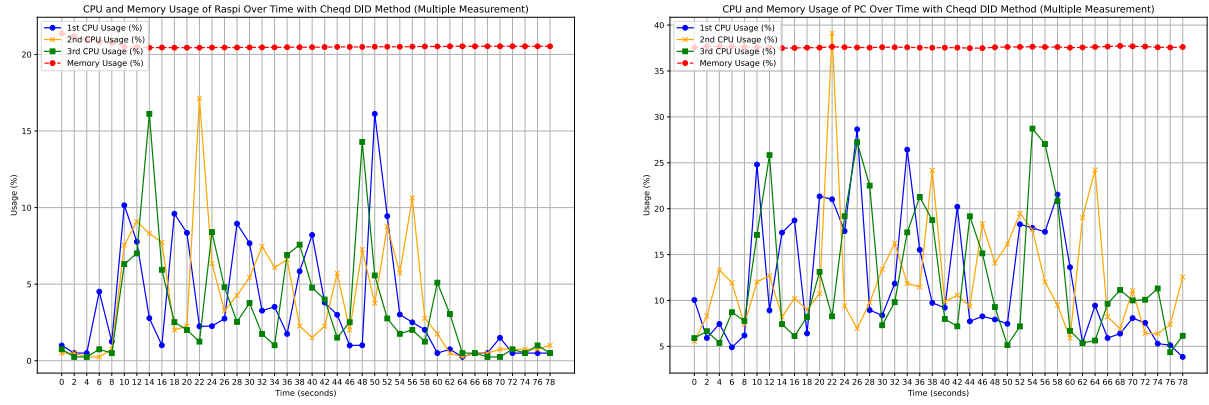


Figure 5: CPU and memory usage of Raspberry Pi 4 and PC over time with CHEQD DID method.

evaluation of system performance.

In the graphs, the results of the three measurements are shown separately: the blue line represents the first measurement, the orange line represents the second measurement, and the green line represents the third measurement. For memory usage, however, the differences between the three measurements were negligible. As such, the memory usage results are consolidated into a single red line, representing the average of all three measurements for simplicity.

To provide further insight into the performance differences specifically with the KEY DID method, three key metrics were calculated for the CPU usage data collected during the load phase: mean, range, standard deviation. Additionally, momentary outlier values were excluded to ensure a more accurate and representative analysis. It is important to note that these calculations are based on measurements from the load phase only.

The Raspberry Pi exhibited higher standard deviation and range in CPU usage compared to PC, as shown in the 3 measurements. The mean CPU usage on the Raspberry Pi was consistently lower than PC. The results of these measurements are shown in Figure 4.

For the CHEQD DID method during the 60-second load phase, both devices issued and verified approximately 10 VCs. All CPU usage metrics (mean, standard deviation and range) were lower on the Raspberry Pi compared to the PC, over the 3 measurements. The results of these measurements can be observed in Figure 5.

The CPU and memory usage results for the Walt.id system under the KEY and CHEQD DID methods offer a detailed view of resource consumption, illustrating how each system, Raspberry Pi and PC, handle continuous credential issuance and verification operations. These results highlight both the memory efficiency of the Walt.id and the distinct CPU utilization patterns across different device configurations. The following conclusions can be drawn based on the results:

Memory Usage: Across both systems (Raspberry Pi and PC), the Walt.id system shows no significant memory usage increase during the 60-second load phase compared to the idle phases. This observation is consistent across both DID methods (KEY and CHEQD), indicating that credential issuance and verification operations in Walt.id do not place a noticeable memory load on the system.

KEY DID Method: The PC maintained stable CPU usage with minimal fluctuations, indicated by a low standard deviation (2.65, 2.66, 2.48). In contrast, the Raspberry Pi exhibited greater variability, with higher standard deviation (8.27, 9.25, 13.7), reflecting its limited ability to maintain consistent processing under load.

The relatively narrow range of PC (12.88, 10.53, 11.75) compared to Raspberry Pi (31.51, 38.72, 39.99) further supports the observation that the PC has sufficient processing power to handle credential issuance and verification tasks smoothly, maintaining a consistent CPU load.

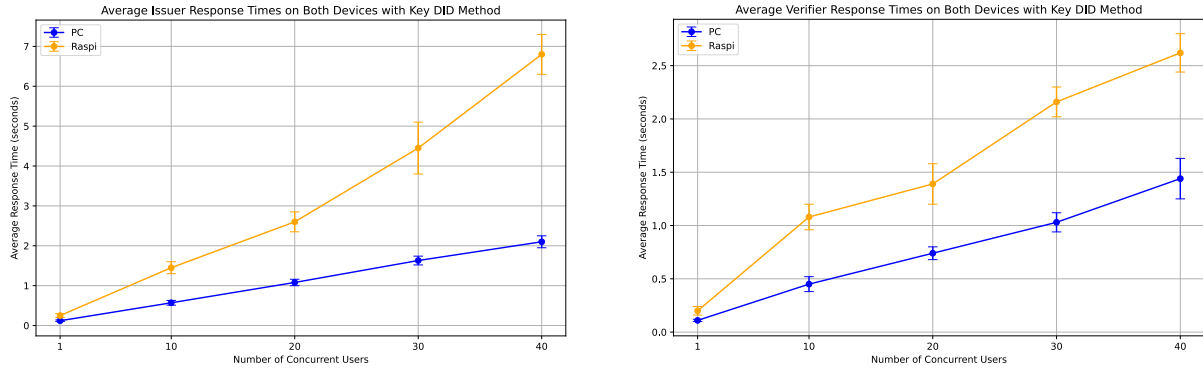


Figure 6: Average issuance and verification response times on PC and Raspberry Pi with KEY DID method.

CHEQD DID Method: Both devices showed noticeable CPU usage spikes, which align with the issuance and verification of 10 Verifiable Credentials. These spikes are linked to the system’s interaction with the external Universal Resolver for DID resolution. The PC, with higher standard deviation (6.56, 6.60, 7.53), demonstrated significant swings in CPU activity, scaling quickly to handle high-demand tasks and dropping during idle periods. The Raspberry Pi, with lower standard deviation (3.77, 3.44, 3.65), displayed more consistent CPU usage. These patterns emphasize the impact of external services like Universal Resolver on system performance.

4.3. Scalability Measurements

The next measurement focuses on scalability. In this measurement, issuance and verification operations were evaluated separately using two different DID methods: CHEQD and KEY. The scalability tests aimed to observe the impact of concurrent users on Verifiable Credential (VC) issuance and verification latency. To achieve this, 40 different Walt.id accounts were used, and the number of concurrent users was varied—starting from 1 and increasing to 10, then 20, 30, and finally 40 concurrent users.

To ensure the reliability of the results, measurements were repeated 10 times for each different number of concurrent users. The values shown in the graphs below represent the averages of these 10 measurements. Additionally, error bars are included in the graphs to illustrate the variability in average response times across different measurements. These error bars were calculated using the standard deviations for each concurrency level. To simulate concurrent users, threading was employed. For each level of concurrency, a corresponding number of threads was created. These threads sent a request to the Walt.id services.

In addition to measuring average response times, the maximum number of concurrent users that the Walt.id services could support before encountering a complete halt was also evaluated. This analysis aimed to determine the upper limit of requests the system could handle under increasing loads. At 110 concurrent issuance requests, the wallet service on both devices halted, requiring a restart. For verification, this limit increased to 120 requests. Under the CHEQD DID method, the PC supported up to 70 concurrent issuance requests, while the Raspberry Pi managed 65. For verification, both devices reached a critical threshold of 65 concurrent users before halting.

Figure 6 shows the average response times for VC issuance and verification measured with the KEY DID method on both devices under different levels of concurrent users.

Figure 7 shows the average response times for VC issuance and verification measured with the CHEQD DID method on both devices under different levels of concurrent users.

Following insights can be drawn based on the analysis of response times for VC issuance and verification across varying levels of concurrent users using the KEY and CHEQD DID methods:

KEY DID Method: With the KEY DID method, the average response time for both issuance and verification operations increases predictably with the number of concurrent users. This trend is evident

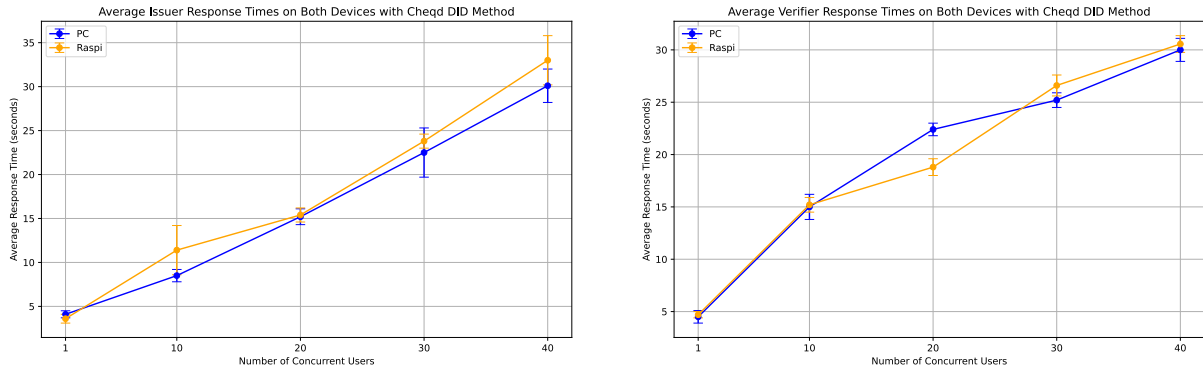


Figure 7: Average issuance and verification response times on PC and Raspberry Pi with CHEQD DID method.

on both the PC and Raspberry Pi devices, although the rate of increase and absolute response times differ between the two. The system exhibits consistent scaling behavior under this method, with response times gradually rising as user load increases.

Specifically, for issuance operations, the response time on the Raspberry Pi increases from an average of 0.22 seconds for 1 user to 6.73 seconds for 40 users, while the PC's response time increases from 0.10 seconds to 2.08 seconds for the same range of users. These results show that the Raspberry Pi experiences a larger proportional increase in issuer response time compared to the PC as concurrency levels grow, reflecting its limited processing power.

Specifically, for verification operations, the response time on the Raspberry Pi increases from an average of 0.20 seconds for 1 user to 2.62 seconds for 40 users, while the PC's response time increases from 0.11 seconds to 1.44 seconds for the same range of users. Unlike issuance operations, the proportional increase in verification response time is similar between the Raspberry Pi and the PC, indicating that both devices scale at a comparable rate under growing concurrency levels. This suggests that while the Raspberry Pi has higher absolute response times due to its limited processing power, the relative increase in verification latency remains consistent with that of the PC.

CHEQD DID Method: For the CHEQD DID method, the response times for both issuance and verification operations are significantly higher compared to the KEY DID method due to the additional overhead introduced by blockchain-based DID resolution. Specifically, for verification operations, the Raspberry Pi's response time increases from an average of 4.78 seconds for 1 user to 30.48 seconds for 40 users, while the PC's response time increases from 4.59 seconds to 30.02 seconds for the same range of users. Unlike the KEY DID method, where the PC consistently outperforms the Raspberry Pi in both absolute response times and proportional increases, the CHEQD DID method demonstrates more similar scaling behavior between the two devices. This suggests that the external DID resolution process imposes a dominant latency, diminishing the relative impact of hardware differences on performance.

For issuance operations with the CHEQD DID method, a similar trend is observed. The Raspberry Pi's response time increases from 3.45 seconds for 1 user to 33.10 seconds for 40 users, while the PC's response time increases from 3.98 seconds to 30.14 seconds for the same range. The results highlight that the CHEQD DID method introduces a substantial baseline latency for both devices, which grows predictably with increasing concurrency but is less influenced by the devices' internal processing capabilities.

Maximum Number of Concurrent Requests: Testing the maximum number of concurrent requests revealed consistent patterns across both devices. For the KEY DID method, the wallet service failed at 110 concurrent issuance requests on both the Raspberry Pi and the PC, requiring a restart to restore functionality. Interestingly, the wallet service was able to handle up to 120 concurrent verification requests on both devices, indicating that the service can support a higher number of concurrent users for verification than for issuance. Additionally, there were no significant differences between the two

devices in terms of the maximum number of concurrent users handled before the wallet service stopped.

For the CHEQD DID method, both devices demonstrated similar thresholds for stable operation. The PC supported up to 70 concurrent issuance requests before the wallet service stopped, while the Raspberry Pi managed 65 requests under similar conditions. During verification operations with the CHEQD DID method, both devices reached a maximum of 65 concurrent users before the wallet service stopped. These results highlight that, despite differences in processing power, the devices exhibited nearly identical maximum capacities when handling concurrent requests under both DID methods.

5. Conclusion

This study provides an evaluation of the Walt.id SSI stack's performance across multiple DID methods (CHEQD, WEB, KEY, JWK) in both a resource-constrained Raspberry Pi and a PC environment, using a residential DSL network. The results reveal notable differences across DID methods. Blockchain-based methods like CHEQD introduce significant delays (2-3 seconds per operation) due to reliance on external services such as Universal Resolver. In contrast, local DID methods such as KEY and JWK demonstrated much lower latencies (below 0.5 seconds) across both devices. These findings align with existing literature, with latency ranges of 2-3 seconds in [10], while Hyperledger Indy achieved latencies of up to 1.2 seconds [14].

Scalability testing for the KEY DID method showed response time increases with the number of concurrent for both devices. However, critical thresholds were observed at 110-120 concurrent requests, after which the wallet service of Walt.id required a restart. Comparatively, CHEQD exhibited greater challenges, with higher response times occurring at lower user loads due to the overhead of blockchain-based resolution. While prior studies on Hyperledger Indy in cloud environments showed scalability up to 120 concurrent users with error rate of approximately 35% [14], our results indicate that such scalability is harder to achieve in constrained environments like the Raspberry Pi and a PC.

Both devices demonstrated stability under local DID methods, such as KEY and JWK, with low variance in CPU and memory usage. However, the Raspberry Pi's constrained hardware led to longer response times and greater sensitivity to increasing loads compared to the PC. This highlights the importance of device capabilities when deploying SSI systems in embedded or resource constrained environments.

Acknowledgments

This paper is supported in part by the Federal Ministry for Economic Affairs and Climate Action (BMWK) as part of the MoveID project on the basis of a decision by the German Bundestag.

References

- [1] C. Allen, The Path to Self-Sovereign Identity, 2016. URL: <https://www.lifewithalacrity.com/article/the-path-to-self-sovereign-identity/>.
- [2] C-SPAN, Senate Banking Committee Hearing on Equifax Data Breach, 2017. URL: <https://www.c-span.org/program/senate-committee/senate-banking-committee-hearing-on-equifax-data-breach/487379>.
- [3] M. Isaac, S. Frenkel, Facebook Security Breach Exposes Accounts of 50 Million Users, The New York Times, 2018. URL: <https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html>.
- [4] Raspberry Pi Foundation, Raspberry Pi Products, 2024. URL: <https://www.raspberrypi.com/products/>.
- [5] Walt.id, Walt.id: Identity and Wallet Infrastructure, 2024. URL: <https://walt.id/>.
- [6] Walt.id, Walt.id Community Stack Documentation, 2024. URL: <https://docs.walt.id/community-stack/home>.

- [7] Walt.id, Walt.id Issuer API Documentation: Getting Started, 2024. URL: <https://docs.walt.id/community-stack/issuer/api/getting-started>.
- [8] Walt.id, Verifiable Credentials (W3C) Documentation, 2024. URL: <https://docs.walt.id/community-stack/concepts/digital-credentials/verifiable-credentials-w3c>.
- [9] Walt.id, Manage Keys SDK Overview, 2024. URL: <https://docs.walt.id/community-stack/issuer/sdks/manage-keys/overview>.
- [10] Q. Stokkink, G. Ishmaev, D. Epema, J. Pouwelse, Manage Keys SDK Overview, in: 2021 IEEE 46th Conference on Local Computer Networks (LCN), 2021, pp. 1–8. doi:10.1109/LCN52139.2021.9525011.
- [11] Hyperledger Indy, Hyperledger Indy: Decentralized Trust Framework, 2024. URL: <https://www.lfdecentralizedtrust.org/projects/hyperledger-indy>.
- [12] uPort, uPort: Decentralized Identity Management Platform, 2024. URL: <https://www.uport.me/>.
- [13] Jolocom, Decentralized Infrastructure for Self-Sovereign Identity, 2024. URL: <https://jolocom.io/>.
- [14] A. Siqueira, A. F. Da Conceição, V. Rocha, Performance Evaluation of Self-Sovereign Identity Use Cases, in: 2023 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), 2023, pp. 135–144. doi:10.1109/DAPPS57946.2023.00026.
- [15] OpenID Foundation, OpenID for Verifiable Presentations 1.0, 2024. URL: https://openid.net/specs/openid-4-verifiable-presentations-1_0.html.
- [16] Walt.id, The Community Stack, 2024. URL: <https://docs.walt.id/community-stack/home/the-community-stack>.
- [17] OpenID Foundation, OpenID for Verifiable Credentials, 2022. URL: https://openid.net/wordpress-content/uploads/2022/05/OIDF-Whitepaper_OpenID-for-Verifiable-Credentials_FINAL_2022-05-12.pdf.
- [18] M. Sabadello, A Universal Resolver for self-sovereign identifiers, Medium, 2017. URL: <https://medium.com/decentralized-identity/a-universal-resolver-for-self-sovereign-identifiers-48e6b4a5cc3c>.