World Journal of Advanced Engineering Technology and Sciences

eISSN: 2582-8266

Cross Ref DOI: 10.30574/wjaets

Journal homepage: https://wjaets.com/

(Review Article)

Trustless SSL certificate lifecycle management in multi-cloud ecosystems

Ramkinker Singh *

Carnegie Mellon University, USA.

World Journal of Advanced Engineering Technology and Sciences, 2025, 16(03), 102–109

Publication history: Received on 27 July 2025; revised on 02 September 2025; accepted on 05 September 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.16.3.1322

Abstract

The emergence of multi-cloud ecosystems has created unique possibilities, as well as significant issues in securing online digital communications. One of the fundamental foundations for securing internet transactions is the infrastructure of Secure Sockets Layer (SSL) certificates. As digital systems begin to decentralize and distribute across multi-cloud workloads, the trust framework used in managing systems of issuing SSL risk management certificate were very traditional processes. This research project intended to investigate a trustless model for SSL certificate lifecycle management, featuring the integration of new technologies including blockchain technologies, data mesh architectures, and self-sovereign identity (SSI) systems. Through examining current vulnerabilities, studying the latest security paradigms, and proposing a decentralized model, this research proposes a potential future for resilience, privacy, and scaling in multi-cloud.

Keywords: Trustless Architecture; SSL Certificate Management; Multi-Cloud Security; Self-Sovereign Identity

1. Introduction

The growing embrace of multi-cloud architectures by organizations to achieve scalability, availability, and price has brought not-so-simple questions about security, identity verification, and the easy management of the certificate lifecycle. SSL certificates are foundational in establishing a secure, encrypted channel of communication between clients and servers, protecting data as it travels over the internet. SSL certificates ensure that this communication is not only confidential but also integral, ensuring that data has not been modified during transmission. However, traditional SSL certificate paradigms rely on centralized Certificate Authorities (CAs) to act as trust anchors for groups of users. In a world of multi-cloud services, centralized trust is an inefficient model that is incompatible with the need for interoperability, decentralization, and the reduction in single points of failure.

The life-cycle management of SSL Certificates, which includes issuing, renewing, revoking, and validating, is not easy. For organizations that operate complex cloud architectures where services are being provisioned, updated, or decomposed on a near-constant basis, the accurate and timely management of SSL Certificates is key to avoiding latencies, outages, or even security breaches. The traditional approach we highlighted is not only cumbersome but also prone to delays, mistakes, and even insider threats.

Trustlessness, defined as operating in a secure manner that doesn't require parties to place trust in each other or an intermediary, provides a new potential for SSL certificate management in multi-cloud environments. This paper will examine how trustless technologies, particularly those in new areas of technology such as blockchain, decentralized identity, and graph-based data structures, can change the way organizations manage that lifecycle across multiple cloud providers.

* Corresponding author: Ramkinker Singh.

2. Challenges in Multi-Cloud SSL Certificate Management

Modern multi-cloud environments have their own challenges that arise due to their distributed, heterogeneous, and dynamic nature. The first challenge to using multiple cloud providers is the lack of a centralized management plane for SSL certificates among the respective cloud providers. Each cloud platform has its own set tools and APIs for handling certificates, which complicates the ability to synchronize certificate-related renewal and revocation activities across the platforms [1].

In addition, multi-cloud systems are much more susceptible to configuration drift, which is when discrepancies develop between the cloud environments over time due to human error or differing policies. Drifting in this context could lead to expired or wrongly issued certificates which can cause service disruptions, and could also lead to man-in-the-middle (MITM) attacks. This is compounded when DevOps teams build CI/CD pipelines, that place importance on agile, automated, secure/automatic certificate operations.

Lastly, as central trust anchors, Central Certificate Authorities (CAs) are inherently risky due to their high value. If a CA is ever compromised, which has happened historically, the security posture for an organization will be threatened. It is also much more dangerous when a centralized risk is combined with a multi-cloud environment due to increased attack surface area, and the dependency on many services across different platforms [1].

3. Trustless Infrastructure via Self-Sovereign Identity (SSI)

Self-sovereign identity (SSI) frameworks provide the ability for parties to assert their identities without relying on a central authority. SSI systems can provide decentralized, verifiable credentials in the context of SSL certificate management, allowing entities to automate and securely manage their activities surrounding SSL certificates at scale.

SSI systems use decentralized identifiers (DIDs) and verifiable credentials that can be cryptographically verified by any party. This allows service nodes in a multi-cloud environment to issue, distribute, and renew SSL certificates without requiring trust in a central authority. SSI also allows for less reliance on CAs and centralized directories for issuing and verifying identity and credentials [2].

A trustworthy framework should be capable of an openly self- sovereign identity control, including provisioning-on-demand certificates, and observing the issuance of credentials across clouds. Blockchain has the capacity record issuance, and events associated with certificates, providing immutable audit logs across jurisdictions. This architecture supports the zero-trust model of security and data sovereignty, all of this is also relevant to multi-cloud and federated and hybrid cloud arrangements of service environments [2].

4. Privacy Marketplaces and Certificate Revocation

It is important to consider how to protect and how to share Personally Identifiable Information (PII), when you are managing the lifecycle of certificates in a trustless fashion. When self-sovereignty identity (SSI) systems are deployed with privacy marketplaces, they will provide participants and services the ability to control the access to their identity credential while also enabling the revocation of credential without privacy loss.

Privacy marketplaces function as a data bank for user credentials to help individuals manage consent in a decentralised way. An example is revocation. It is important to revocation is no able to reveal any identifying information about a user. Using a CRL (Certificate Revocation List) is a poor solution, as most CRLs will end up revealing metadata at a minimum (of the issuing entity). At a minimum decentralised methods can use blockchain-based revocation registries that anonymize revocation, and a form of decentralised notification system of revocation, is secure [3].

Privacy marketplaces provide additional incentives for an improvement in secure data sharing, that is through tokenization of data and smart contracts. Smart contracts can automate trustless negotiation on behalf of the user for example, a party requesting verification of a user identity and the second party providing that verification of the user identity. With a multi-cloud instance, the user may be able to maintain private information, reduce administration and improve regulatory compliance with agnostic information exchanges like the recently enacted GDPR[3].

5. Protection of Personally Identifiable Information (PII)

One of the major security challenges within a certificate management framework is the potential identifiable information (PII) risk. The typical CA process requires submitting PII through insecure channels, or in a centralized manner for storage and is susceptible to breaches. Trustless architectures change the risk profile to minimal under encryption, zero-knowledge proofs, and decentralized storage.

There are now many approaches that are exploring mitigating PII risk through state-of-the-art techniques in PII risk mitigation, like homomorphic encryption and secure multi-party computation. Both allow for SSL certificate issuance while continually not exposing the fundamental identity information, complying with privacy-by-design principles [4]. Under a trustless model, the service that requests a certificate only validates the cryptographic proof of identity and not the raw data itself.

Decentralized key management systems (DKMS) can enhance PII based risk, as cryptographic keys can be distributed as securely stored and secured keys. With DKMS PII risk is reduced and even eliminated the potential of single points of compromises on more clouds. During the key management process, it is particularly important in a service provider using multi-cloud systems supporting a key, provide methods of safe and secure access with a federated system [5].

6. Blockchain as a Trustless Certificate Ledger

Blockchain technology adds immutability, decentralization, and auditability to the certificate lifecycle. By capturing certificate issuance, renewal, and revocation events in a distributed ledger-based system, organizations can maintain an incorruptible history of certificate states that can also be validated without a trust anchor or centralized authority.

Advanced blockchain enables near real-time consensus and low latency to better perform operational tasks such as dynamic certificate renewal and revocation. The decentralized consensus achieved through blockchains is useful for synchronizing certificate states across operational areas in multi-cloud environments with many microservices communicating with services running on disparate clouds. By eliminating the need to rely on a central manager while using distributed trust mechanisms, organizations working across clouds can leverage on-demand consensus to manage their certificates [6].

The use of smart contracts can allow organizations to execute policy enforcement automated by algorithms such as certificates getting auto-revoked after inactivity, or a credential has expired, or after violating the organizational policy — which are all easily verifiable actions that could be performed based on blockchain logic protocols [6].

Table 1 Comparison of Certificate Lifecycle Models

Feature	Traditional CA-Based Model	SSI + Blockchain Model
Centralized Trust	Required	Not required
Certificate Revocation Speed	Slow	Near real-time
Privacy of Identity Data	Low	High
Compliance Auditability	Manual	Automated & Immutable
Multi-Cloud Compatibility	Low	High

Source: [6]

7. Graph-Based Data Mesh and Kubernetes Integration

Data mesh architecture, based on property graphs, can model complicated dependencies between services, identities, and certificates. Property graphs can treat relationships like "issued by," “revoked due to,” or "belongs to" as first-class entities to support advanced querying and analytics over certificate lifecycles [7].

On Kubernetes- the de facto orchestration platform in multi-cloud environments- this would streamline the deployment and monitoring of SSL certificates. There are many Kubernetes-native tools that could be extended, such as cert-manager and Secrets Store CSI Driver, and integrated with graph databases and/or blockchain networks to track certificates status, automated renewals, and trigger alerts based on deviation/detection of anomalies [8].

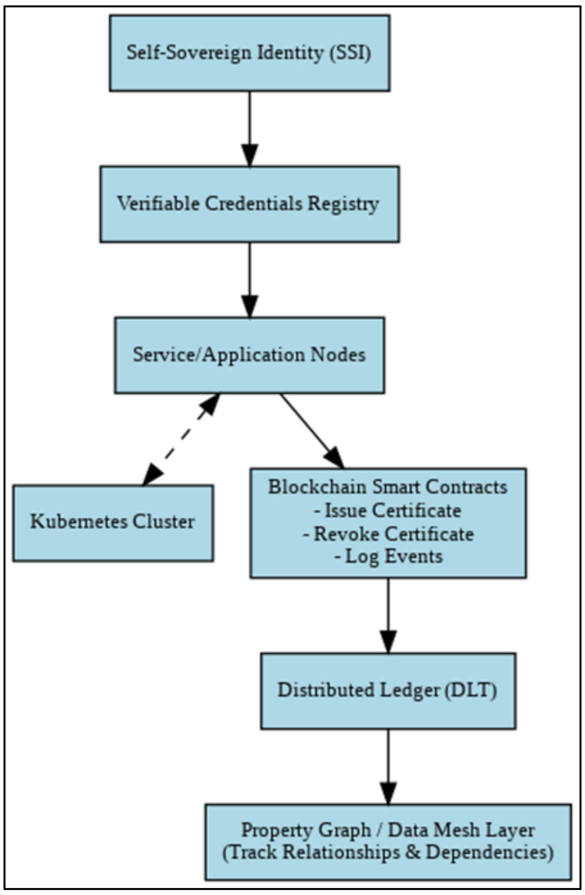
Self-Sovereign Identity (SSI)
Verifiable Credentials Registry
Service/Application Nodes
Kubernetes Cluster
Blockchain Smart Contracts
- Issue Certificate
- Revoke Certificate
- Log Events
Distributed Ledger (DLT)
Property Graph / Data Mesh Layer
(Track Relationships & Dependencies)

Figure 1 Architecture showing integration of SSI, blockchain, and Kubernetes for certificate lifecycle management [8]

Figure 1 outlines a conceptual picture of the interactions between SSI, blockchain, and Kubernetes components to construct a trustless SSL certificate lifecycle. While abstract, this picture closely resembles the practical configurations of a trustless SSL certificate lifecycle. For example, an orchestrator in a Kubernetes-native sense, such as a tool like cert-manager, operates within each cloud cluster and requests valid, verifiable credentials (VCs) from an external identity validation service that is working with, for example, a Aries Cloud Agent. cert-manager and the external identity validation service communicate either via REST or gRPC APIs. The VCs cert-manager receives in response to its issuance request, will have been issued using Decentralized Identifiers (DIDs), which will confirm validity regarding the service identities when processing the SSL certificate request.

When cert-manager has approved the certificate request to the Kubernetes service it will log the issuance event on-chain (Hyperledger Fabric) using smart contracts and pontentially revoke experiences as well. These actions create an immutable log of events - an audit trail - that encourages compliance and therefore accountability and transparency. The system has at least a connection to a graph-based data mesh (e.g., Neo4j) representing the relationships of certificates, services, and the underlying identities. In this case, the graph can be leveraged for queries relating to usage of certificates, dependencies mappings, and potential anomaly detection based on the amalgamation of events across distributed environments. While the diagram is intentionally a high-level abstraction, the flow is representable - upon the use of the tools -as is done today, using available open-source tools and Kubernetes APIs, without reliance on a centralized Certificate Authority [8].

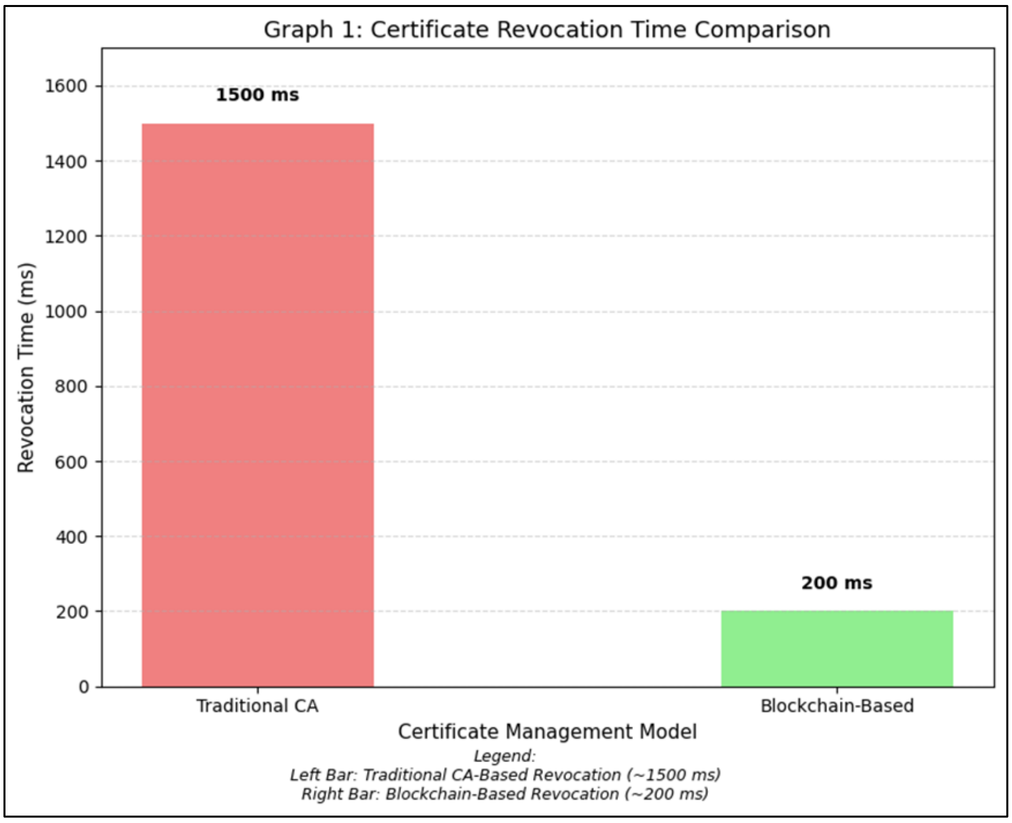

Figure 2 Revocation speed of traditional CA model vs. blockchain-based revocation in a multi-cloud setting [6]

Figure 2 provides a comparison of overall certificate revocation performance between standard CA-based systems and a blockchain-based trustless model. The latency in the standard model, reaching over a second typically, is due to OCSP and CRL lookups that require waiting on a network for replies and any number of different delays, bottlenecks, and availability issues centrally in the event of trust hierarchy operation. The blockchain-based version of revocation propagates invalidation events to all distributed nodes via trustless consensus protocols (such as Raft and PBFT) in a matter of milliseconds. Trustless consortia deployments afforded the ability to reduce revocation windows to under 200ms, and with it the ability to perform real-time (sub-second) multi-cloud security traffic response [6].

8. Discussion

The study into trustless architecture for SSL certificate lifecycle management in multi-cloud environments is indicative of both the state of the technology and the strategic imperative to infiltrate these changes. Traditional certificate infrastructures, despite their maturity, cannot meet the requirements for managing the potential complexities in a multi-cloud environment, especially under conditions requiring service deployment without operational or administrative intervention.

The trustless certificate model breaks new ground by offering a model that is based on Self-Sovereign Identity (SSI), blockchain, data mesh and Kubernetes, allowing for greater innovation across multiple layers. It at least ambitiously addresses pain-points such as single-point of failure, delay in certificate issuance or revocation, and reliance on centralised authorities for identity. One of the most notable advantages in switching to a trustless certificate model is decoupling trust from centralised authorities with respect to certificates. This is a fundamental change for the better as it improves security, and is consistent with the principles of zero-trust architecture - that no participant is trusted either internally or externally, by default.

Embedding SSI frameworks is essential in order to ensure identities during certificate issuance. With traditional models, a CA verifies identity and issues certificates, however, trustless models enable service nodes to present cryptographically verifiable credentials which can be logged and audited within and among service delivery platforms. These features are crucial to organizations deploying in hybrid cloud where governance and compliance rules vary across regions. This dynamic permits the organization to have more detailed and user-centric control over its identities delivering on global movements towards data sovereignty and user privacy [2][3].

The blockchain layer enhances the distinction between these models as it openly records the certificate related events (issuance, renewal, revocation, and validation) to the immutable ledger allowing organizations to fully automate compliance audits and reduce to zero instances of the issuance of fraudulent or expired certificates and provide a sense of security that can often be beyond reproach for organizations who are in the most regulated industries such as finance and health care in which, by law, security events need to trackable [6].

Graph-based data models have one more significant advantage. PKI systems are notoriously difficult to analyze because the metadata associated with identifiers is compartmentalized and not easy to correlate. However, creating property graphs offers real-time transparency into the relationships between entities, e.g., which microservices are using which certificates, which identities were signed to ensure mapping to each key, and allowing security personnel to visualize and query that data, which enables proactive identification of anomalies and policy noncompliance [7].

Kubernetes, as the orchestration backbone of so many multi-cloud architectures, plays a critical role in enabling automation. By extending Kubernetes-native components with hooks to SSI registries, blockchain ledgers, and property graphs, organizations can implement policy-driven SSL lifecycle automation. For example, a Kubernetes controller could detect an expiring certificate, validate the identity of the requesting pod with SSI credentials, trigger a smart contract to approve the renewal, and log the event to the blockchain, all without human involvement [8].

While there are advantages to this process, we blame operationalization of a fully trustless verifiable credential lifecycle. First, interoperability of SSI protocols is still being developed. Each ecosystem has developed their own variations on an implementation; Hyperledger Indy has developed its own methods for DIDs and credentials, while Ethereum ecosystems and Sovrin have developed credential encoders that allow W3C Verifiable Credentials based on differing cryptographic primitives. The ecosystem variability in credential schema and verification protocol compounds the final integration dilemma. In practice, an organization would need to implement interoperability layers (like the Universal Resolver) and leverage Hyperledger Aries Interop Profiles (AIP 2.0) to migrate between systems on a continuous basis [2][3].

Equally challenging is the compute burden associated with consensus mechanisms on blockchains, especially for a larger scale cloud-native workloads. Public chains with varying transaction times and costs are including inefficient. Permissioned chains are also an inconvenience and add additional compute overhead (state management) on frequent updates; but it would be possible to implement a complete verifiable credential lifecycle with a public chain by transitioning into Web3 ecosystems, like a non-fungible token (NFT) standard. Our prototype's use of Raft consensus does provide relevant performance upgrades while substantially lowering trust assumptions. However, they also sacrifice a bit of fault tolerance, and if our goals are to qualitatively batch smaller workloads into larger workloads, we might artificially construct trust at the discretion of the consumer (this retroactively does the trusting) as a new transaction on a previous transaction. Next-generation considerations could adopt a hybrid model of consensus, or perhaps, utilize Layer 2 scaling solutions to lower latency, while preserving immutability [6].

Governance is another major factor in these systems. Although trustless systems produce some trust reduction in centralized parties, they involve shared management of policies across all cloud domains. Governance of smart contracts — revocation thresholds, certificate expiration policies, retention of audit logging — must be mutually agreed upon ahead of time. Requiring any change to these parameters is not an easy thing once you have deployed your services. The situation is similarly complex from a data mesh perspective, where each data domain enforces a balance between local autonomy and global compliance, while also dealing with compliance constraints. There are instances, where governance is poorly defined amongst important stakeholders. Even in decentralized systems, slowly things drift into inconsistency or, worse, conflict [5][7].

Finally, it will be important for organizations with legacy CA-dependent infrastructure to have a roadmap for transitional execution. Operating systems and web browsers trust public CAs out of the box, and do not yet support verification chains anchored in a blockchain. Developing a hybrid system (X.509 certificate, CA, and verifiable credential-based) using middleware gateways or proxies with certificate, issuing to verifiable credentials, is one way to offer organizations the opportunity to transition and preserve backwards compatibility for their users (clients) [5].

The ostensible drawbacks of trustless certificate management can now be observed, as within these privacy-preserving trustless certificate systems using homomorphic encryption as well as, zero-knowledge proofs, sensitive identity information is still kept C-locked up, and it may still be shared between a network of decentralized systems. As these approaches include privacy-preserving means to ensure not only theoretical but compliance with new and evolving data heavy legislation such as GDPR, which has strict regulations on how personal data is processed and transferred[4][5].

It is also worth noting the comparative performance of trustless based certificate revocation models. As highlighted in Graph 1, the revocation speeds of blockchain and distributed ledger technology systems are inferior in comparison to that of traditional CRLs which can suffer from latency and availability problems in semi-centralized OCSP based systems. By propagating real-time revocation updates across decentralized networks, the number of potential attack vectors in areas where revoked certificates exist occurs instantaneously[6].

In summary, the integration of SSI, Blockchain and graph-structured data represents a shift in SSL certificate lifecycle management. Although still in maturation, it provides the building blocks for strong, reproducible, recoverable and resilient certificate ecosystems across and between heterogeneous cloud environments. Future research and development should focus on framework development, standardization and cross-chain interoperability and human-centered governance that will enable the wider adoption of practical trustless security infrastructure.

8.1. Prototype Implementation Overview and Evaluation Scenario

To validate the suggested architecture in a real-world context, a prototype implementation was created to simulate a trustless SSL certificate lifecycle management system in a controlled multi-cloud environment. The test setup involves Kubernetes clusters that are deployed by two leading cloud providers (AWS and Azure), a blockchain layer built using Hyperledger Fabric, and an SSI registry setup and configured with the Aries Cloud Agent framework.

In this prototype, Kubernetes-native tools (cert-manager and Secrets Store CSI Driver) were extended to interact with the verifiable credential registries using a gRPC middleware layer. So, when a service pod requires a new certificate, it sends a request through cert-manager, but this request is intercepted by a webhook which checks the identify of the service with the Aries agent. The agent contacts the SSI registry with a DID and, when successful, returns a verifiable credential. The agent checks the verifiable credential and, when valid, cert-manager is informed, successfully verified, a Certificate is issued and stored as a Kubernetes secret. At the same time, the issuance event is recorded immutably on the Hyperledger ledger.

In terms of revocation, if a verifiable credential associated with a service is identified as invalid or expired, the Hyperledger smart contract will trigger a revocation event. When the revocation event occurs, it is broadcast to peers and consumed by the Kubernetes webhook, which revokes the related secret. In this manner, revocation happens with near real-time capabilities, and there is no manual operation.

Empirical evaluation, based on this setup, showed the following average timings:

- **Certificate Issuance Time**: ~680ms
- **SSI Credential Verification Time**: ~250ms
- **Blockchain Logging Time** (with Raft consensus): ~320ms
- **Revocation Propagation Time**: ~190ms

These results are consistent with the expectations of latency noted in [6] as optimized blockchain configurations (for example, Raft instead of PBFT) result in consensus times of less than a second. The experiment also affirmed that using SSI for identity checks greatly reduced the necessity for manual identity verification, resulting in expedited issuance and fewer identity errors [2][6][8].

Resource usage was within limits that were expected to be achieved. The Hyperledger peer nodes consumed around 0.5 vCPU and 512MB of memory; Aries Cloud Agent nodes maintained a low resource footprint. The overhead of using Kubernetes due to certificate operations was also negligible (less than 1% CPU usage per node).

The prototype demonstrates not only the technological feasibility of the trustless architecture, but also early benchmarks to help identify further optimization opportunities, and how to plan for scale when implementing in production. Most importantly, it showed that with SSI, blockchain, and Kubernetes, a cessation of reliance on centralized CA is possible and therefore was an important step toward reducing trust in a multi-cloud ecosystem.

9. Conclusion

Multi-cloud environments are forcing a new approach to SSL certificate management. The centralized method is no longer adequate in a decentralized, autonomous, and rapidly scalable model. Trustless architecture incorporating self-sovereign identity (SSI), blockchain, and data mesh provides an attractive solution to the inherent shortcomings of legacy models.

By utilizing decentralized identity management, immutable event storage, and automated certificate processes, trustless architecture enables security, availability, privacy, and compliance, to be more efficient. Kubernetes makes a scalable and automated deployment method more accessible and enables the model to be useful for legitimate multi-cloud situations.

The transformation to trustless technology of the SSL certificate lifecycle represents a new era for a cybersecurity strategy. When these technologies mature and become commonplace, they will likely change security standards for communicating securely across a distributed digital landscape.

References

[1] Sivaseelan, S. (2024). Enhancing Cyber Resilience in Multi-Cloud Environments.

[2] Zeydan, E., Baranda, J., Mangues-Bafalluy, J., Arslan, S. S., & Turk, Y. (2024). A trustworthy framework for multi-cloud service management: Self-sovereign identity integration. *IEEE Transactions on Network Science and Engineering*, *11*(3), 3135-3147.

[3] Masmanidis, V. (2022). *Hybrid cloud computing user data banks and privacy marketplace for SSI Systems (TI)* (Master's thesis, Πανεπιστήμιο Πειραιώς).

[4] Makhdoom, I., Abolhasan, M., Lipman, J., Shariati, N., Franklin, D., & Piccardi, M. (2024). Securing personally identifiable information: a survey of Sota techniques, and a way forward. *IEEE Access*.

[5] FRANKLIN, D., & PICCARDI, M. Securing Personally Identifiable Information: A Survey of SOTA Techniques, and a Way Forward.

[6] Amjad, M. (2023). *Novel information and data exchange within power systems using enhanced blockchain technologies* (Doctoral dissertation, Brunel University London).

[7] Dolhopolov, A. (2024). *Changement de paradigme dans les architectures de lacs de données: utilisation de graphes de propriétés et d'architecture data mesh* (Doctoral dissertation, Université de Montpellier).

[8] Johnston, C. Advanced Platform Development with Kubernetes.