Secure Patterns for Internet CrEdentials
Internet-Draft
Intended status: Standards Track
Expires: 10 July 2026

D. Wang
F. Liu
L. Li
Y. Jiang
Huawei

6 January 2026

A Public Key Service Provider for Verification in Multiple Issuers and Verifiers
draft-wang-spice-public-key-service-provider-02

Abstract

SPICE provides a selective disclosure mechanism of credentials from issuer. However, future network services may be built on the trust between multiple entities. Obtaining the public key of multiple issuers for a verifer from potential multiple sources can be complex. In this contribution, an optional public key service is proposed in SPICE architecture for the issue of obtaining the public keys of the issuers from multiple trusted entities. The basic function of public key service is proposed including public key registration, token verification, and a potential implementation such as the distributed ledger. We hope that the proposed contribution can be used as infomative for SPICE regarding to the token validation procedure.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 10 July 2026.

Copyright Notice

Table of Contents

1. Introduction

Digital credential ecosystems require verifiers to obtain the correct issuer public key material to validate signatures. This becomes challenging in multi-stakeholder deployments across administrative domains, such as physical supply chains, IoT and critical infrastructure operations, and telecommunications service provisioning, where verifiers may encounter many issuers whose keys can rotate or be revoked.

Consider a cross-domain deployment in which manufacturer A produces device a, and operator X deploys device a alongside an existing device/system x. During onboarding and operation, a and x may need to validate signed statements issued by A or X regarding device identity, access control, compliance, or firmware/software integrity. In supply chain settings, such statements may traverse intermediaries and be validated asynchronously at scale. In telecommunications networks, providers may enable users or services to exchange digitally signed attributes (e.g., subscription status or account-related information) during interactions, requiring relying parties to validate credentials from multiple issuers.

In current PKI-based deployments, trust between a verifier and multiple issuers is often realized using one of the following approaches:

1. Manual configuration: verifiers maintain local trust stores and administrators provision issuer certificates/keys. For example, device a adds network operator X's credentials, and device x adds device manufacturer A's. This does not scale with many issuers, frequent key rotation, or constrained devices and operational environments.

2. Common trust anchor: a third party T acts as a trust anchor (directly or via hierarchical chains such as T->A->a for x and T->X->x for a.) for issuers. This requires global adoption of the same anchor and may not match real-world trust relationships across domains.

3. Cross-certification: issuers mutually certify each other' s keys. For example, A issues a credential for X's public key, and X does the same for A's, forming trust chains A->X->x for a and X->A->a for x. This becomes operationally complex in many-to-many relationships and is difficult to maintain under rotation and revocation.

These approaches become inadequate when (a) verifiers validate credentials from a large and changing issuer set, (b) verification is high-volume or time-sensitive, (c) trust relationships are dynamic and not well represented by a fixed hierarchy, or (d) operational constraints limit frequent updates to local trust stores.

This document proposes an optional Public Key Service Provider (PKSP) architecture to reduce this operational complexity. A PKSP provides a service for issuer public key registration and discovery, including key status information needed for verification (e.g., updates and revocation), enabling verifiers to obtain current verification key material on demand in multi-domain environments.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Architecture Overview

3.1. Multi-party trust

Figure 1 illustrates the traditional approach where issuers, i.e. CAs, must establish pairwise mutual trust. This process incurs high workload, and maintaining trust will require even more effort when public key updates and revocations occur.

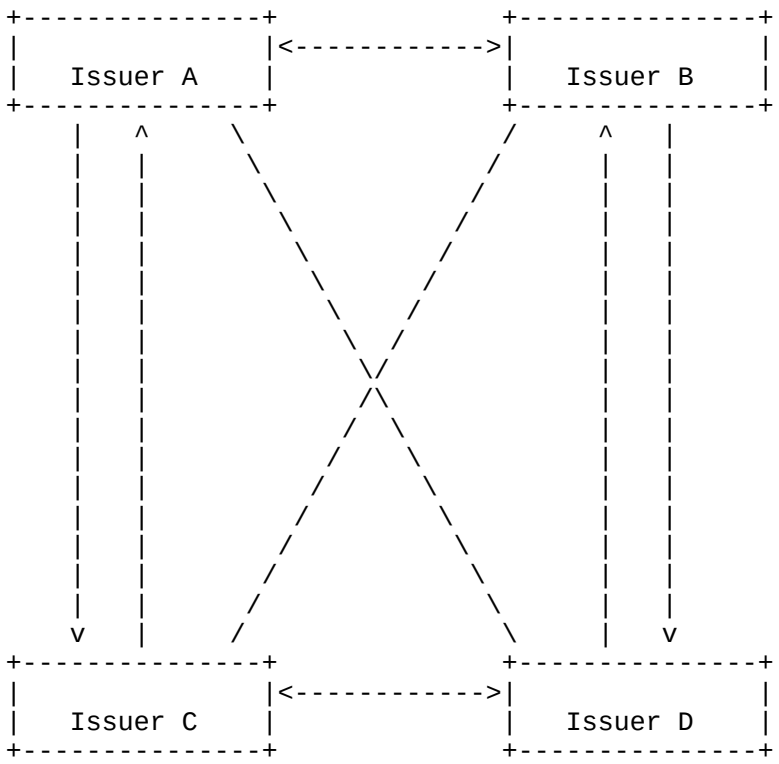

Figure 1 Establish mutual trust between every two issuers.

When we can establish mutual trust among multiple parties based on PKSP, as shown in Figure 2, the issuer only needs to publish registration and revocation messages of public keys/credentials to

the public-Key-Service-Provider(PKSP), which can then be accessed by other issuers or verifiers when they are performing verification, then, during the verification, the verifiers may verify the token such as RFC 9207[RFC9207] and RFC 9449 [RFC9449] , or the selective disclosed token such as SD-CWT([draft-ietf-spice-sd-cwt-02])

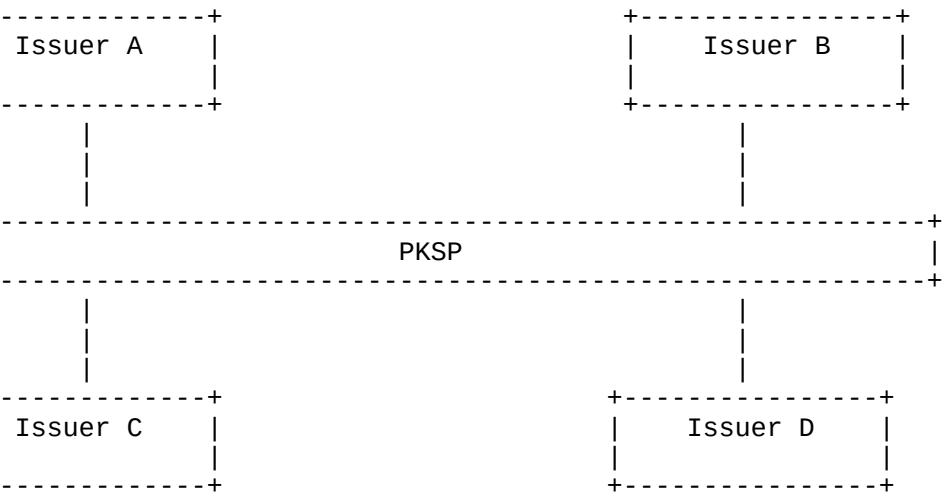

Figure 2 Multi-party Trust Based on PKSP

During the registration process, identity verification is usually carried out to ensure that only legitimate entities can add public keys. The service checks on the format and validity of the public key to ensure it meets the system requirements. The registration process between the issuer and the PKSP is as follows:

* 1. Submission by the Issuer. The issuer initiates registration by sending a request to the PKSP. This request contains the issuer’s public keys. Meanwhile, the issuer declares the purposes of these keys, such as for token issuance, revocation, and so on. Along with the public keys, detailed descriptive information about the issuer is provided. This includes the issuer’s name, identifier, the expiration time of the public keys, as well as a declaration of the claims of the tokens the issuer can provide.

* 2. Validation by the PKSP. Upon receiving the registration request, the PKSP first validates issuer’s public key. The descriptive information provided by the issuer is then examined for accuracy and completeness. The PKSP may cross-reference this information with existing databases or industry-recognized sources to verify the issuer’s legitimacy.

* 3. Response from the PKSP. If all the information is found to be valid and compliant, the PKSP sends a registration response to the issuer. This response includes a confirmation of successful registration. The PKSP may also provide additional information like the storage location of the issuer's public key within the PKSP. In case the registration request is not approved, the PKSP sends a rejection notice to the issuer, stating the reasons for rejection.

* 4. Completion of Registration. Once the issuer receives a positive response from the PKSP, the registration process is considered complete. The issuer can then start using the registered public key for its intended purposes and issue tokens according to the specifications provided during registration. The PKSP updates its records to reflect the newly registered issuer and is ready to support the verifiers in matters related to public key querying.

3.2. PKSP for verification

As a public public key service platform, PKSP accepts public key registration from issuers, stores the registered public keys securely and in a non-tamperable manner. When a verifier queries for a key, it provides the corresponding public key.

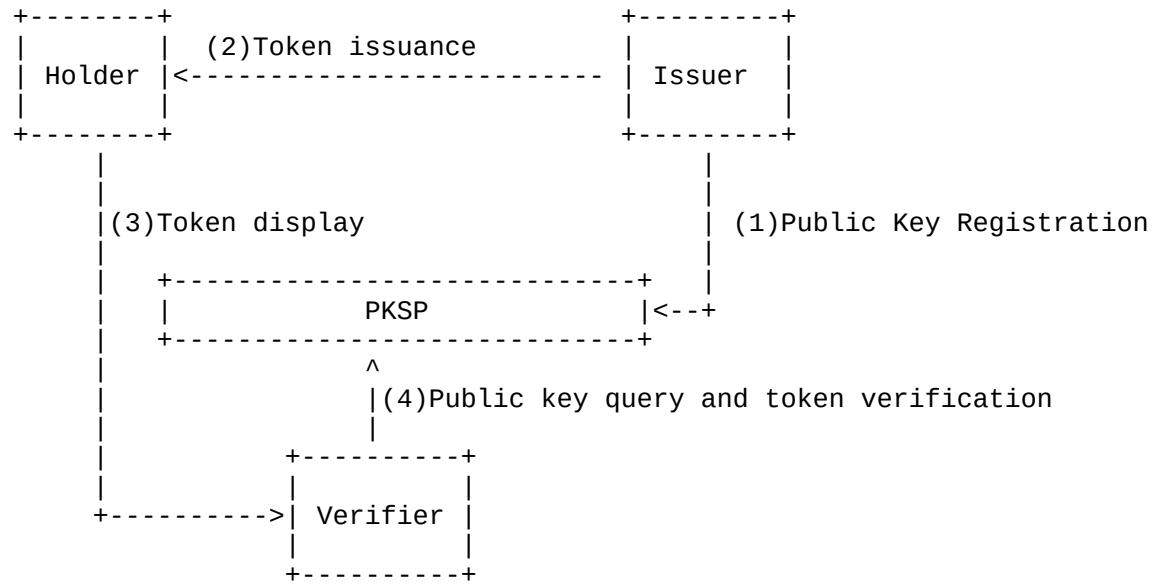

Figure 1 System architecture of a PKSP

When a verifier is performing SD-CWT verification, it can query the public key information of relevant issuers through the PKSP. It supports various query methods, such as precise queries based on the identity identifier of the entity, credential number, etc., to quickly obtain the required public key.

The following steps shall be performed between verifiers and PKSP:

* 1. Token Receipt. The Holder sends the token to the Verifier.

* 2. Request for public key from PKSP. After receiving the token, the verifier extracts the identification information of the issuer from the token. For example, the "iss" field in the header or payload may be found to identify the issuer. The verifier then uses this identification information to send a request to the PKSP, asking for the public key of this issuer.

* 3. PKSP Processes the Request and Returns the Public Key. When the PKSP receives the verifier’s request, it first authenticates the verifier’s identity. After successful authentication, based on the issuer identification provided in the request, it searches for the corresponding public key in its storage system. Then the PKSP returns this public key to the verifier as a response message. The response message may also contain some metadata, such as the expiration date of the public key and the type of the public key.

* 4. Token Verification. The Verifier uses the public key obtained from the PKSP to verify the token according to the signature algorithm specified in the token.

* 5. Processing of Verification Results If the verification passes, the verifier can perform subsequent business logic processing based on the information in the token payload, such as allowing the holder to access specific resources. If the verification fails, the verifier usually rejects the operations related to this token and may record relevant log information for subsequent security audits and problem-troubleshooting.

4. Enabling Technologies of PKSP

4.1. Permissioned distributed ledger

As for the implementation technology of PKSP, distributed ledger technology strongly enables PKSP to meet the following requirements. Decentralized storage spreads data across multiple nodes to prevent single-point failures. The tamper-proof data structure, formed by complex encryption, makes data modification extremely hard. Besides, the consistency maintenance based on multi - party consensus, such as the Practical Byzantine Fault Tolerance (PBFT) algorithm, ensures that multiple nodes conduct verification for public key operations in the PKSP. Only when most nodes approve are operations recorded, reducing the impact of malicious or wrong actions by individual nodes.

4.2. Operation of distributed ledger

TODO

5. Detailed protocol

TODO

6. IANA Considerations

This document has no IANA considerations.

7. Security Consideration

TODO

8. References

8.1. Normative Reference

[GS_PDL_024] PDL, E., "Architecture enhancements for PDL service provisioning in telecom networks", November 2024, <https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=68066>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC9207] Meyer zu Selhausen, K. and D. Fett, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9207, DOI 10.17487/9207, March 2022, <https://www.rfc-editor.org/info/rfc9207>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/9449, September 2023, <https://www.rfc-editor.org/info/rfc9449>.

8.2. Informative References

[draft-ietf-spice-sd-cwt-02] Prorock, M., Campbell, O., Steele, H., and R. Mahy, "SPICE SD-CWT", December 2024, <https://datatracker.ietf.org/doc/draft-ietf-spice-sd-cwt/>.

[draft-ietf-spice-use-cases-00] Prorock, M. and B. Zundel, "Use Cases for SPICE", September 2024, <https://datatracker.ietf.org/doc/draft-ietf-spice-use-cases/>.

Acknowledgments

TODO

Authors' Addresses

Donghui Wang
Huawei
Email: wangdonghui124@huawei.com

Faye Liu
Huawei
Email: liufei19@huawei.com

Lun Li
Huawei
Email: lilun20@huawei.com

Yuning Jiang
Huawei
Email: jiangyuning2@h-partners.com