DNSOP
Internet-Draft
Intended status: Informational
Expires: 3 September 2026

A. Duda
M. Korczynski
O. Hureau
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG
J. Zhang
H. Labiod
Huawei Technologies France S.A.S.U.

2 March 2026

A DNS-Based Framework for Privacy-Preserving Identity
draft-duda-dnsop-dns-did-00

Abstract

This document presents a framework for privacy-preserving identity management based on DNS, supporting large-scale management of users, IoT devices, and AI agents. It introduces Self-Certifying Identifiers (SIDs), User/Service Trustees as trusted proxies, and leverages DNSSEC-secured TXT records to bind public keys to identities. The framework enables privacy-by-design, where real identities are hidden behind trusted entities, through privacy-preserving intermediarie. Credentials bound to SIDs support role-based access control, while ephemeral tokens ensure short-lived authorization. Although initially DNS-dependent, the model can extend to other directories like DIDs or IPFS. This approach aligns with zero-trust architectures and supports automated, AI-driven interactions in future networks.

Table of Contents

1. Introduction

In this draft, we present a framework for Privacy-Preserving Identity Management based on DNS supporting large scale management of users, IoT devices, or AI agents. It defines new entities involved in Identity Management, specifies Self-Certifying Identifiers based on public keys, and offers public keys stored in DNS as trust anchors. This design draws inspiration from the architecture of Decentralized Identifiers (DIDs) [W3C.DID-Core], particularly in the use of cryptographic derivation of identifiers, though this document focuses on DNS as the initial resolution layer.

2. Framework for DNS-Based Identity

We can observe that the standard Self-Sovereign Identity frameworks place the User at the center and gives the user the control over unveiling its personal information. However, there is a tension between Privacy and Trust:

the User wants to remain anonymous and do not unveil its personal information, which may end up in the situation in which other entities in the system will not trust its - nobody wants to interact with an anonymous interlocutor;

if the User discloses its real identity, other entities in the system may trust him. The conclusion is that we need an entity that separates the User from other entities, hides its private attributes or personal information, and represents him with respect to other actors. Such an intermediary— referred to as a Trustee— acts as a privacy-preserving proxy. It manages identifiers and credentials on behalf of users or services, enabling authentication without exposing real identities. Unlike centralized identity providers, this framework relies on cryptographic identifiers and DNSSEC to ensure trust without depending on any single administrative authority.

The proposed architecture for management of identities includes the following entities:

Users Identity Owners correspond to persons or device owners who use and control their identifiers. They may also control the identifiers of devices/agents they own. The User entity may delegate the management of its identifiers to a trusted entity: a User Trustee entity.

User Trustee provides support for registering identifiers on behalf of Users. It takes care of the credentials and acts as a trusted proxy with respect to other entities. The advantage of introducing the User Trustee entity is double: i) this kind of organisms or companies may gain reputation and be recognized as a trust provider, so trusted by relying parties (for example, the User entity, the Service entity, and the Service Trustee), ii) it hides the User from all interactions and unveiling its personal information, thus providing privacy-by-design. The User Trustee operates based on technical transparency and auditable operations, rather than legal mandates. Its trustworthiness stems from cryptographic proofs and operational accountability, not state-granted privileges.

Service (or Server) the entity with respect to which persons and devices/agents want to authenticate and access authorized resources - the User entity may request access to a Service provided by the Service entity. It uses identifiers and public keys to authenticate and authorize an Identity Owner or her devices/agents. The Service needs to trust either a User, a User Trustee, or Credential Issuers (in the simplest case, the Service Trustee).

Service Trustee provides support for registering identifiers and public keys on behalf of Service entities. In a symmetric way to the User Trustee entity, the Service Trustee entity may help the Service entity to manage its identities and the relationship with User entities and other Trustee entities. The Service Trustee may create, update, and cancel Service identifiers stored in an appropriate form. The Service Trustee entity may also be configured to issue the Credential to the User entity. Its role to verify the Credential on behalf of the Service entity and the Service entity is configured to authorize the access further based on the verification result provided by the Service Trustee entity. Instead of presenting Credentials directly to the Service entity, the User entity may first authenticate to the User Trustee entity. Upon successful authentication, the User Trustee entity presents the Credential either to the Service entity or to the Service Trustee entity, facilitating secure and privacy-respecting interactions. The Service entity may verify the Credential and authorize access to the requested Service based on the role or attributes indicated in the Credential. In this process, the Service entity interacts with the User Trustee entity rather than the User entity directly, thus balancing privacy and trust of user identity. Entities store all public information on Identities (Identifiers and Public Keys) in DNS in the TXT record associated with the given Identifier.

3. Self-Certifying Identifiers, Credentials, and Ephemeral Tokens

A Self-Certifying Identifier (SID) is a concatenation of the Context and the Context-dependent Identifier. We assume that the User generates a pair of keys: public key P and secret key S. The Context-dependent Identifier (CI) is derived from a public key - it is the hash of public key P: SID = Context | CI, CI = base32(RIPEMD160(SHA256(P))), where SHA256 and RIPEMD160 are hash functions resulting in 256 and 160 bits, respectively, and base32 is a binary-to-text encoding scheme. The Context provides the information about the type of the cryptographic system used for generating the keys. The identifiers of all entities may be represented as follows:

User entity: sid_u, derived from the pair of keys (P_u and S_u),

Service entity: sid_s, derived from the pair of keys (P_s and S_s),

User Trustee entity: sid_tu, derived from the pair of keys (P_tu and S_tu),

Service Trustee entity: sid_ts, derived from the pair of keys (P_ts and S_ts).

The User Trustee keeps the following information on behalf of the User entity: sid_u, sid_c and C, where sid_u is the identifier of the user entity, sid_c is the identifier of the Credential, and C is the Credential. The user identifier is for a given connection to the Service entity with the Credential.

The Credential is defined as follows:

C =[sid_u, sid_c, role, SHA256(sidu, sidc, role)S_ts],

where sid_u is the user identifier, sid_c is the credential ID, role is an attribute of the user entity (e.g., role=admin), and the hash on this information is signed by the Service Trustee entity (in a case where the credential issuer is the Service Trustee entity) with its secret key S_ts. The Credential is kept secret by the User Trustee entity and presented to the Service entity in a form encrypted by the public key of the Service entity upon connection so only the Service entity may read it.

The Service entity is configured to authorize the access by generating an Ephemeral Token (ET) based on the Credential, and authorizing the access based on the ET.

ET is a short-lived token that enforces security by leaving attackers with a tiny window exploit, for example, stolen credentials. The idea is that SID identifiers are persistent (which may be stored in DNS) and only ET is used in the actual connection request to a service entity to authorize access by the user entity. ET is defined as follows:

ET = [SHA256(T, L, sid_u, sid_c, K_s)]S_ts,

where T is the timestamp, L is the token lifetime, sid_u is the user identifier, sid_c is the credential ID, K_s is a shared key, and the hash on this information is signed by the service trustee entity with its secret key S_ts.

4. DNS as a Public Directory of Identifiers

For the main entities of the framework: User, User Trustee, Server, and Server Trustee, their identifiers and the corresponding public keys are stored in DNS and are publicly available. The identifiers and the corresponding public keys stored in the TXT record are cryptographically signed in DNS and when an entity receives a public key associated with a given identifier, DNSSEC provides trust comparable to conventional Public Key Infrastructures (PKI) used in HTTPS.

Assume that the identifier of the user entity sid_u is registered under a user trustee-controlled domain, e.g., trustee.id. So, sid_u may exist as the following fully qualified domain name (FQDN): 1exq5yqbl6l48pf0fu7juhltjkrbu2rxf.trustee.id.

Other identifiers and keys may be stored in the TXT DNS record associated with the FDQN in the format based on request for comments (RFC) 6376 used for domain keys identified mail (DKIM), which uses DNS TXT records to store lists of tag/value pairs encoded in the form of tag=value separated by a semi-colon (;).

So, for a given SID user identifier:
1exq5yqbl6l48pf0fu7juhltjkrbu2rxf,

its associated identifiers and keys may be stored as the following TXT record:

```
1exq5yqbl6l48pf0fu7juhltjkrbu2rxf TXT
"sid=1exq5yqbl6l48pf0fu7juhltjkrbu2rxf;
key=1caac9c64711f66e6ed71b37dc...;
tid=16ed71b37dc5e69c5124fe93ee12446e1;
cid=1g43wxdm35yxtjyuje72j64esenyneplk"
```

Such publicly available and trusted information may allow other entities to authenticate the User entity, whose identifier being 1exq5yqbl6l48pf0fu7juhltjkrbu2rxf, having the public key 1caac9c64711f66e6ed71b37dc..., with respect to the Service entity represented by the Service Trustee entity 16ed71b37dc5e69c5124fe93ee12446e1, using the Credential with the credential ID 1g43wxdm35yxtjyuje72j64esenyneplk.

5. DNS-Based Identities for Agents

The identity framework can also support automated entities such as software agents or autonomous services. These entities can be assigned SIDs and managed through User or Service Trustees, enabling secure and auditable interactions without human intervention. Use cases include IoT automation, network management, and machine-to-machine (M2M) coordination, where accountability and role-based access control are required.

6. Conclusions

We have presented a framework for Privacy-Preserving Identity Management based on DNS supporting large scale management of users, IoT devices, or AI agents. Our approach to anchor trust is to rely on DNS as a directory of identities with DNSSEC guaranteeing the integrity of responses. Note that this document is intended as an informational overview of a possible architecture. It does not mandate any specific implementation, nor does it promote any commercial model. Alternative directories such as DIDs or IPFS may also be used to store identifiers and public keys, enabling deployment in decentralized environments. This framework is compatible with modern identity standards such as [W3C.DID-Core] and supports compact, secure token formats like [RFC8392], while leveraging DNSSEC in a manner consistent with [RFC8552].

7. Security Considerations

TODO

8. IANA Considerations

9. Acknowledgements

TODO

10. Normative References

1. *[W3C.DID-Core]* W3C, "Decentralized Identifiers (DIDs)", W3C Recommendation 10 July 2022, <https://www.w3.org/TR/did-core/ (https://www.w3.org/TR/did-core/)>.

2. *[RFC8392]* Jones, M., Lengyel, T., and S. Erdtman, "CBOR Web Token (CWT)", RFC 8392, DOI <https://doi.org/10.17487/RFC8392 (https://doi.org/10.17487/RFC8392)>, May 2018.

3. *[RFC8552]* Finch, T., "SMTP Security via DANE TLS", RFC 8552, DOI <https://doi.org/10.17487/RFC8552 (https://doi.org/10.17487/RFC8552)>, April 2019.

Authors' Addresses

Andrzej Duda
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG
Email: Andrzej.Duda@imag.fr

Maciej Korczynski
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG
Email: maciej.korczynski@grenoble-inp.fr

Olivier Hureau
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG
Email: olivier@hureau.com

Jun Zhang
Huawei Technologies France S.A.S.U.
Email: junzhang1@huawei.com

Houda Labiod
Huawei Technologies France S.A.S.U.
Email: houda.labiod@huawei.com