RefPassport

DNS-Anchored Reference Verification

The RefPassport Protocol

A DNS-Anchored Open Standard for Cryptographic Employment Reference Verification

Version:	v1.0 (Draft)
Date:	February 2026
Author:	John Parkinson
Standard:	refpassport1

Abstract

RefPassport defines an open protocol for issuing, delivering, and verifying employment references using Ed25519 digital signatures anchored to DNS TXT records. References are portable, tamper-evident, and independently verifiable by any third party without contacting the original issuer or creating an account. Inspired by DKIM, the protocol leverages well-understood standards (Ed25519, DNS TXT, JSON) to create a decentralised trust model for employment verification. This whitepaper describes the protocol specification, cryptographic primitives, security properties, and reference implementation.

Table of Contents

1. Introduction & Motivation

Employment references underpin hiring decisions worldwide, yet the process for verifying them has remained fundamentally unchanged for decades. A reference letter is typically a document on company letterhead, sometimes accompanied by a phone number for verbal confirmation. This system has three critical failures.

The Problem

- References are trivially faked. A letterhead, a PDF, and a fabricated phone number are all it takes to forge a convincing reference. Research suggests up to 30% of applicants embellish or fabricate their references entirely.
- Verification is painfully slow. Every new employer calls the old employer, waits days for a response, and typically receives a guarded two-sentence confirmation. The process does not scale and often fails to produce useful information.
- References cannot be revoked. Once a paper reference is issued, there is no mechanism to recall it if circumstances change -- for example, if misconduct is discovered after the employee has left.

The Solution

RefPassport addresses these failures with a cryptographic protocol inspired by DKIM (DomainKeys Identified Mail), the standard that secures billions of emails daily. The core insight is that DNS already provides a decentralised, globally accessible key-value store tied to verified domain ownership. By publishing an Ed25519 public key as a DNS TXT record, an employer can sign references that anyone can independently verify -- without trusting any central authority, creating an account, or making a phone call.

The result is a reference that is: (1) impossible to forge without access to the employer's private key; (2) instantly verifiable by scanning a QR code or visiting a URL; (3) portable, travelling with the candidate as a PDF or shareable link; and (4) revocable at any time by the issuing employer.

2. Protocol Overview

Actors

- Issuer: The employer who creates and signs references. Generates an Ed25519 key pair, publishes the public key in DNS, and signs references with their private key.
- Candidate: The employee who receives the signed reference. Carries it as a PDF or shareable link and presents it to future employers or recruiters.
- Verifier: Any third party (recruiter, HR team, hiring manager) who verifies a reference by checking the cryptographic signature against the issuer's public key published in DNS.

Protocol Flow

The protocol follows five steps:

- 1. Issuer generates an Ed25519 key pair. The private key is encrypted with the user's password (AES-256-GCM) and stored on the server. The plaintext key never leaves the client.
- 2. The public key is published as a DNS TXT record at {SELECTOR}._refpassport.{DOMAIN}.
- 3. To issue a reference, the issuer decrypts their private key client-side and signs the reference payload (Ed25519 detached signature).
- 4. The candidate receives the signed reference as a branded PDF with an embedded QR code, plus a shareable verification link.
- 5. Any verifier can check the signature against the public key from DNS. No account, login, or phone call required.

Design Goals

- Portability: References travel with the candidate, not locked to any platform or vendor.
- Tamper-evidence: Any modification to the reference content invalidates the signature.
- Decentralised trust: Verification relies on DNS, not a central certificate authority.
- Zero-knowledge: The server never has access to plaintext signing keys.
- Simplicity: The protocol uses well-understood, battle-tested standards (Ed25519, DNS TXT, JSON).

3. DNS Record Specification

The issuer's Ed25519 public key is published as a DNS TXT record under a well-known subdomain. This is the trust anchor that allows anyone to independently verify a reference without trusting RefPassport or any other intermediary.

Record Format

```
Name:  {SELECTOR}._refpassport.{DOMAIN}
Type:  TXT
Value: v=refpassport1; k=ed25519; p={BASE64_PUBLIC_KEY}
```

Field Definitions

Field	Type	Description
SELECTOR	String	Alphanumeric key selector (e.g. REF1). Max 63 chars. Allo...
_refpassport	Fixed	Protocol-reserved subdomain prefix. Must appear exactly a...
DOMAIN	String	The issuer's domain (e.g. example.com). Must be a domain ...
v	String	Protocol version. Must be "refpassport1" for this version.
k	String	Key algorithm. Must be "ed25519" in v1.
p	Base64	Ed25519 public key, base64-encoded. 32 bytes (44 characte...

Example

```
REF1._refpassport.acmecorp.com.  3600  IN  TXT
  "v=refpassport1; k=ed25519; p=Ky2hL9xR4bT...44chars...="
```

Key Rotation

To rotate keys, publish a new TXT record with a different selector (e.g. REF2) while keeping old records active. Previously-issued references continue to verify against the original selector. This is directly analogous to DKIM key rotation in email infrastructure.

4. Cryptographic Primitives

Ed25519 Digital Signatures

All references are signed using Ed25519 (Edwards-curve Digital Signature Algorithm on Curve25519, RFC 8032). Ed25519 provides 128-bit security with compact 32-byte public keys, 64-byte signatures, and fast sign/verify operations. It is the same algorithm used by Signal, SSH, and numerous blockchain protocols.

Component	Size	Description
Public Key	32 bytes	Base64-encoded (44 characters). Published in DNS.
Private Key	64 bytes	Base64-encoded (88 characters). Seed + public key. Never ...
Signature	64 bytes	Base64-encoded (88 characters). Detached signature over c...
Encoding	Base64	Standard base64 with padding.

AES-256-GCM (Private Key Encryption)

Private keys are encrypted at rest using AES-256-GCM, an authenticated encryption scheme that provides both confidentiality and integrity. The encryption key is derived from the user's password using PBKDF2 with SHA-256 and 100,000 iterations. A random 16-byte salt and 12-byte IV are generated for each encryption operation.

The server stores only the ciphertext, salt, and IV. Without the user's password, the server cannot decrypt the private key. This is the zero-knowledge property: the server facilitates reference storage and verification but never has the ability to sign references on behalf of the user.

SHA-256 Hashing

SHA-256 is used for two purposes: (1) as the hash function within PBKDF2 key derivation, and (2) for computing integrity hashes of reference payloads and PDF documents. Hashes are represented as lowercase hexadecimal strings (64 characters for 32 bytes).

5. Payload Format

The reference payload is a JSON object containing the reference data and a unique nonce. This is the exact data that gets signed and later verified.

JSON Structure

```
{
  "name":  "James Holloway",
  "role":  "Baggage Handler",
  "dates": "June 2021 - December 2024",
  "text":  "James was a reliable and hardworking...",
  "nonce": "550e8400-e29b-41d4-a716-446655440000"
}
```

Field Specifications

Field	Type	Description
name	String (max 200)	Full name of the candidate.
role	String (max 200)	Job title or position held.
dates	String (max 100)	Employment period in free-form text.
text	String (max 10,000)	Reference letter content. Newlines preserved.
nonce	UUID v4	Unique identifier. Prevents replay attacks.

Canonical Serialisation

To ensure that both the signer and verifier produce identical byte representations, the payload is serialised with alphabetically sorted keys using JSON.stringify(obj, Object.keys(obj).sort()). This is critical because JavaScript does not guarantee object key order. Without sorting, the same payload could produce different byte representations on different runtimes, causing signature verification to fail.

```
// Canonical form (keys sorted alphabetically):
// {"dates":"June 2021...","name":"James Holloway",
//  "nonce":"550e8400...","role":"Baggage...","text":"James..."}
```

6. Signing Process

Signing happens entirely on the client side. The private key is decrypted in the browser using the user's password, used to sign, and then discarded from memory. The plaintext private key is never sent to or stored on the server.

Steps

- 1. Compose reference data (name, role, dates, text).
- 2. Generate a UUID v4 nonce using crypto.randomUUID().
- 3. Create canonical JSON by sorting keys alphabetically and stringifying.
- 4. Encode to bytes using UTF-8 (TextEncoder).
- 5. Compute Ed25519 detached signature using nacl.sign.detached().
- 6. Base64-encode the signature (64 bytes -> 88 characters).

Code Example

```
import nacl from 'tweetnacl';
import { encodeBase64, decodeBase64 } from 'tweetnacl-util';

function signReference(payload, privateKeyBase64) {
  const canonical = JSON.stringify(
    payload, Object.keys(payload).sort()
  );
  const payloadBytes = new TextEncoder().encode(canonical);
  const keyBytes = decodeBase64(privateKeyBase64);
  const sigBytes = nacl.sign.detached(payloadBytes, keyBytes);
  return encodeBase64(sigBytes); // 88 characters
}
```

7. Verification Process

Verification is a multi-step process that checks the reference's integrity, authenticity, and current status. The result is one of three trust levels: Gold, Silver, or Invalid.

Algorithm

- 1. Fetch reference by ID from the issuing server or extract from PDF metadata.
- 2. Check revocation status. If revoked -> INVALID.
- 3. Check expiry date. If expired -> INVALID.
- 4. Reconstruct canonical JSON from the payload (sorted keys).
- 5. Verify Ed25519 signature using the stored public key. If invalid -> INVALID.
- 6. Perform DNS TXT lookup at {SELECTOR}._refpassport.{DOMAIN}.
- 7. If DNS record found and public key matches -> GOLD. Otherwise -> SILVER.

Signature Verification Code

```
function verifyReference(payload, signatureBase64, pubKeyBase64) {
  const canonical = JSON.stringify(
    payload, Object.keys(payload).sort()
  );
  const payloadBytes = new TextEncoder().encode(canonical);
  const sigBytes = decodeBase64(signatureBase64);
  const keyBytes = decodeBase64(pubKeyBase64);
  return nacl.sign.detached.verify(
    payloadBytes, sigBytes, keyBytes
  );
}
```

DNS Verification Code

```
async function verifyDns(domain, selector, expectedKey) {
  const name = `${selector}._refpassport.${domain}`;
  const expected =
    `v=refpassport1; k=ed25519; p=${expectedKey}`;
  const records = await dns.resolveTxt(name);
  const flat = records.map(r => r.join(''));
  return flat.some(r => r === expected);
}
```

8. Trust Levels

Verification produces one of three trust levels, determined by the combination of signature validity and DNS record presence. This tiered approach ensures the protocol degrades gracefully rather than failing completely when DNS is unavailable.

Condition	Level	Meaning
Signature valid + DNS record matches	Gold	Highest trust. Domain confirmed.
Signature valid + DNS not found	Silver	Partial trust. DNS may be pending.
Signature invalid	Invalid	Do not trust. Content altered.
Reference revoked by issuer	Invalid	Withdrawn by employer.
Reference past expiry date	Invalid	No longer valid.

The Gold/Silver distinction is significant: a Silver result does not mean the reference is fraudulent. It commonly occurs during DNS propagation (which can take up to 48 hours), after a domain transfer, or when the DNS record has been accidentally removed. The cryptographic signature still confirms that the reference data has not been altered since issuance.

9. PDF Proof Format

RefPassport generates a branded PDF document that serves as a portable proof of the reference. The PDF embeds all data needed for verification in its document metadata and includes a QR code linking to the online verification page.

Visual Structure

- Header bar with RefPassport branding (green banner).
- Reference content: candidate name, role, employment dates, and reference text.
- Verification footer: signature hash, verification URL, and reference ID.
- QR code: encodes the URL https://refpassport.com/verify/{REFERENCE_ID}.
- Footer bar: "Powered by RefPassport -- DNS-Anchored Cryptographic Verification".

Embedded Metadata

The following data is embedded in the PDF's document properties for machine-readable verification:

Field	Type	Contents

Title	String	"Reference Letter - {candidateName}"
Subject	JSON	Full canonical JSON payload.
Creator	String	"RefPassport"
Keywords[0]	JSON	Full canonical JSON payload (redundant copy).
Keywords[1]	String	"payload-hash:{SHA256_HEX}" -- hash of canonical payload.
Keywords[2]	String	"signature:{BASE64}" -- the Ed25519 signature.
Keywords[3]	String	"pdf-hash:{SHA256_HEX}" -- hash of final PDF bytes.

Tamper Detection

The SHA-256 hash of the PDF bytes is stored in the database at issuance. A verifier can upload the PDF to check whether SHA256(uploadedPdf) equals the stored hash. If the hashes differ, the PDF has been modified after issuance. This provides a second layer of integrity beyond the payload signature, detecting modifications to the visual presentation of the document.

10. Security Properties & Threat Model

Security Guarantees

- Tamper-Evidence: Any modification to the payload invalidates the Ed25519 signature. The verifier reconstructs the canonical JSON and checks the detached signature, ensuring byte-level integrity.
- Non-Repudiation: Only the holder of the private key can produce valid signatures for a given public key. The issuer cannot deny having created a reference that verifies against their published key.
- Domain Binding: DNS TXT records link public keys to verified domains. An attacker cannot claim signatures from a domain they do not control without compromising the domain's DNS.
- Zero-Knowledge: The server stores only the AES-256-GCM ciphertext of the private key. Without the user's password, the server cannot sign references. Even a complete database compromise does not expose signing capability.
- Replay Protection: Each reference includes a UUID v4 nonce, ensuring unique signatures even for identical reference content. This prevents an attacker from reusing a valid signature on different data.
- Revocability: Issuers can revoke any reference at any time via the API or dashboard. Revoked references always return Invalid status, regardless of signature validity.

Threat Model

Attack	Defence
Forged reference	Ed25519 verification fails without the correct private key.
Modified content	Any change to the payload invalidates the detached signat...
Stolen key from server	Private key is AES-256-GCM encrypted. Server cannot decrypt.
DNS hijacking	Degrades to Silver. DNSSEC recommended for additional pro...

Replay attack	UUID v4 nonce produces unique signatures for identical co...
PDF tampering	SHA-256 hash comparison detects any document modification.

Non-Goals

- Anonymity: References are explicitly linked to identifiable domains. The protocol is designed for attribution, not privacy.
- Confidentiality: Reference content is not encrypted in transit. Once shared, it is readable by any recipient. Encryption could be added as a protocol extension.
- Guaranteed availability: DNS lookup failures degrade to Silver, not Gold. Offline signature verification is possible using the stored public key from the reference database.

11. Implementation Notes

For Issuers

- Generate an Ed25519 key pair on the client using tweetnacl or an equivalent library.
- Encrypt the private key with a strong password (PBKDF2 with 100,000 iterations, AES-256-GCM).
- Publish the public key as a DNS TXT record at {SELECTOR}._refpassport.{DOMAIN}.
- Wait for DNS propagation (typically 15 minutes to 48 hours).
- Sign references client-side. Never send the plaintext private key to the server.
- Store the signature and payload, generate a branded PDF with QR code.
- Share the verification link and/or PDF with the candidate.

For Verifiers

- Receive a verification link or PDF from the candidate.
- Navigate to the verification page or perform programmatic verification via the API.
- Check the trust level: Gold (full trust), Silver (partial trust), Invalid (reject).
- Optionally upload the PDF to verify its SHA-256 hash against the stored value.

Best Practices

- Key Rotation: Use a new selector (e.g. REF2) when generating new keys. Keep old DNS records active.
- DNSSEC: Recommended but not required. Adds cryptographic verification to DNS responses.
- Password Strength: Use 12+ characters with mixed case, digits, and symbols for key encryption.
- DNS TTL: Set to 3600 seconds (1 hour) for a balance between caching and update speed.
- Nonce Generation: Always use crypto.randomUUID() for the nonce. Never reuse nonces.

Recommended Libraries

Library	Language	Purpose
tweetnacl	JavaScript	Ed25519 key generation, signing, and verification.
tweetnacl-util	JavaScript	Base64 encoding/decoding for keys and signatures.
pdf-lib	JavaScript	PDF generation with metadata embedding.
qrcode	JavaScript	QR code generation for verification URLs.
dns.promises	Node.js	DNS TXT record resolution for domain verification.
Web Crypto API	Browser/Node	AES-256-GCM encryption and PBKDF2 key derivation.

12. Comparison with Existing Approaches

Traditional Reference Checking

Traditional reference checking relies on phone calls and emails between the verifier and the previous employer. This process is slow (days to weeks), unreliable (employees may have left), and provides no cryptographic assurance of authenticity. A reference received by phone or email could have been fabricated by the candidate.

Survey-Based Tools

Survey-based reference tools (e.g. Xref, Referee) automate the collection process by sending questionnaires to referees. While faster than manual checking, they still rely on the referee's self-reported identity and do not provide cryptographic verification. The reference data is locked within the platform and is not portable.

Blockchain-Based Credentials

Some credential verification systems use blockchain for tamper-evidence. While cryptographically sound, these approaches typically require specialised wallets, incur transaction fees, and introduce complexity that is unnecessary for employment references. RefPassport achieves equivalent security guarantees using DNS as the trust anchor -- infrastructure that every organisation already manages and understands.

How RefPassport Differs

- No central authority: Verification uses DNS, the same decentralised infrastructure that underpins email (DKIM) and web certificates (CAA records).
- Truly portable: References are PDFs with embedded QR codes, shareable via any channel.
- Zero cost to verify: No account, subscription, or API key needed for verification.
- Instant: Verification takes less than one second, versus days for traditional checking.
- Revocable: Unlike paper references, issued references can be revoked at any time.

13. Future Work

- Formal standardisation: Submit the RefPassport protocol as an Internet-Draft to the IETF for review and potential adoption as an RFC.
- Multi-algorithm support: While v1 requires Ed25519, future versions may support additional signature algorithms (e.g. Ed448, ECDSA P-256) via the "k" field in the DNS record.
- Structured payload extensions: Support for additional fields (e.g. competency ratings, structured feedback) while maintaining backwards compatibility through the canonical serialisation model.
- HRIS integrations: Native connectors for major HR Information Systems (Workday, BambooHR, SAP SuccessFactors) to automate reference issuance as part of the employee offboarding workflow.
- Selective disclosure: Allow candidates to share only specific fields (e.g. employment dates and role without the full reference text) using zero-knowledge proof techniques.
- Cross-protocol interoperability: Alignment with W3C Verifiable Credentials and Decentralised Identifiers (DIDs) for broader credential ecosystem integration.

14. References

[1] D. J. Bernstein et al., "Ed25519: High-speed high-security signatures," 2012.

[2] S. Josefsson & I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," RFC 8032, 2017.

[3] M. Dworkin, "Recommendation for Block Cipher Modes: Galois/Counter Mode (GCM)," NIST SP 800-38D.

[4] E. Allman et al., "DomainKeys Identified Mail (DKIM) Signatures," RFC 6376, 2011.

[5] B. Kaliski, "PKCS #5: Password-Based Cryptography Specification v2.0," RFC 2898, 2000.

[6] P. Mockapetris, "Domain Names - Implementation and Specification," RFC 1035, 1987.

[7] ECMA International, "The JSON Data Interchange Syntax," ECMA-404, 2nd Edition, 2017.

www.refpassport.com