Work References

DNS-Anchored Reference Verification

The Work References Protocol

A DNS-Anchored Open Standard for Cryptographic Employment Reference Verification

Version:	v1.0 (Draft)
Date:	March 2026
Author:	John Parkinson
Standard:	workreferences1

Abstract

Every day, employers write references that can be trivially faked, and recruiters chase verifications that take days to complete. Work References is an open protocol that fixes both problems. It uses Ed25519 digital signatures anchored to DNS TXT records — the same infrastructure that secures email — to let employers issue references that anyone can verify instantly, without phone calls, accounts, or fees. This whitepaper describes how the protocol works, what it guarantees, and how to implement it.

Table of Contents

1. Introduction & Motivation
2. Protocol Overview
3. DNS Record Specification
4. Cryptographic Primitives
5. Payload Format
6. Signing Process
7. Verification Process
8. Trust Levels
9. PDF Proof Format
10. Security Properties & Threat Model
11. Implementation Notes
12. Comparison with Existing Approaches
13. References

1. Introduction & Motivation

The reference process has barely changed in decades. An employer writes a letter on company letterhead, perhaps includes a phone number, and the candidate carries that document from job to job. This system has three fundamental problems.

The Problem

- References are trivially faked. A letterhead, a PDF, and a fabricated phone number. That's all it takes to forge a convincing reference. It happens more often than most employers realise.
- Verification is painfully slow. The verifier emails or calls the previous employer, waits days for a response, and typically receives a guarded two-sentence confirmation. The process does not scale and frequently fails to produce useful information.
- References cannot be revoked. Once a paper reference is issued, there is no mechanism to recall it. If misconduct is discovered after an employee has left, the reference remains in circulation.

The Solution

Work References takes an idea from email security and applies it to employment references. DKIM (DomainKeys Identified Mail) secures billions of emails daily by publishing cryptographic keys in DNS. Work References applies the same principle. An employer publishes a public key as a DNS TXT record on their domain, then uses the corresponding private key to sign references. Any third party can verify that signature against the DNS record — without an account, a phone call, or a central authority.

The result is a reference that cannot be forged without the employer's private key, is verifiable by anyone with DNS access, and travels with the candidate as a self-contained PDF — no central server, account, or phone call required.

2. Protocol Overview

Actors

- Issuer: The employer. They generate a key pair, publish the public key in their DNS, and sign references with their private key.
- Candidate: The employee. They receive the signed reference as a self-contained PDF and carry it from job to job.
- Verifier: Anyone — recruiter, HR team, hiring manager — who checks the signature against the issuer's public key in DNS. No account needed.

Protocol Flow

The protocol operates in 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}._workreferences.{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 self-contained PDF. The payload, signature, domain, and selector are embedded in the PDF metadata.
- 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 public key is published as a DNS TXT record under a well-known subdomain. This is the trust anchor — it allows anyone to independently verify a reference without trusting Work References or any other intermediary.

Record Format

```
Name:  {SELECTOR}._workreferences.{DOMAIN}
Type:  TXT
Value: v=workreferences1; k=ed25519; p={BASE64_PUBLIC_KEY}
```

Field Definitions

Field	Type	Description
SELECTOR	String	Alphanumeric key selector (e.g. REF1). Max 63 chars. Allo...
_workreferences	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 "workreferences1" for this vers...
k	String	Key algorithm. Must be "ed25519" in v1.
p	Base64	Ed25519 public key, base64-encoded. 32 bytes (44 characte...

Example

```
REF1._workreferences.acmecorp.com.  3600  IN  TXT
  "v=workreferences1; 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, 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 most modern cryptographic infrastructure.

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. 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 operation.

The server stores only the ciphertext, salt, and IV. It can't decrypt the private key without the user's password. This is the zero-knowledge property: the server facilitates storage and verification but cannot sign references on behalf of the user, even if fully compromised.

SHA-256 Hashing

SHA-256 serves two purposes: as the hash function within PBKDF2 key derivation, and 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, a unique nonce, and the issuer's domain and key selector. This makes each reference self-contained — a verifier can extract the payload and know exactly where to look up the public key via DNS.

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",
  "domain":   "heathrow-gh.com",
  "selector": "REF1"
}
```

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.
domain	String	Issuer domain. Used for DNS public key lookup.
selector	String	DNS key selector (e.g. "REF1").

Canonical Serialisation

Both signer and verifier must produce identical byte representations from the same data. The payload is serialised with alphabetically sorted keys using JSON.stringify(obj, Object.keys(obj).sort()). This is necessary because JavaScript does not guarantee object key order — without sorting, the same payload can produce different bytes on different runtimes, causing verification to fail.

```
// Canonical form (keys sorted alphabetically):
// {"dates":"June 2021...","domain":"heathrow-gh.com",
//  "name":"James Holloway","nonce":"550e8400...",
//  "role":"Baggage...","selector":"REF1","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, domain, selector).
- 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 requires no central server or database. The reference document contains everything needed: the signed payload (including the issuer's domain and key selector), and the signature. A verifier extracts this data, looks up the public key via DNS, and checks the signature. The result is one of three trust levels: Gold, Silver, or Invalid.

Algorithm

- 1. Extract the payload and signature from the PDF metadata.
- 2. Read the domain and selector fields from the payload.
- 3. Perform DNS TXT lookup at {selector}._workreferences.{domain}.
- 4. If no DNS record found -> SILVER (signature unverifiable against domain).
- 5. Parse the public key from the DNS record.
- 6. Reconstruct canonical JSON from the payload (sorted keys).
- 7. Verify Ed25519 signature using the DNS public key.
- 8. If signature valid -> GOLD. If invalid -> INVALID.

Full Verification Code

```
async function verify(payload, signatureBase64) {
  // 1. Extract domain and selector from the payload
  const { domain, selector } = payload;
  // 2. Look up the public key via DNS
  const name = `${selector}._workreferences.${domain}`;
  const records = await dns.resolveTxt(name);
  const flat = records.map(r => r.join(''));
  const match = flat.find(r =>
    r.startsWith('v=workreferences1;')
  );
  if (!match) return 'silver'; // No DNS record
  const pubKey = match.match(/p=([A-Za-z0-9+/=]+)/)[1];
  // 3. Verify the signature
  const canonical = JSON.stringify(
    payload, Object.keys(payload).sort()
  );
  const bytes = new TextEncoder().encode(canonical);
  const valid = nacl.sign.detached.verify(
    bytes, decodeBase64(signatureBase64),
    decodeBase64(pubKey)
  );
  return valid ? 'gold' : 'invalid';
}
```

8. Trust Levels

Verification produces one of three trust levels, determined by the combination of signature validity and DNS record presence. The tiered approach ensures that verification degrades gracefully rather than failing outright 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.

A Silver result does not indicate fraud. It commonly occurs during DNS propagation (which can take up to 48 hours), after a domain transfer, or when a 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

References are delivered as branded PDF documents that are fully self-contained. The PDF metadata embeds the signed payload (including the issuer's domain and key selector) and the signature — everything a verifier needs to check authenticity using only DNS. No central server required.

Visual Structure

- Header bar with Work References branding (navy banner, #1e3a5f).
- Reference content: candidate name, role, employment dates, and reference text.

- Verification footer: signature hash, verification URL, and reference ID.
- QR code (optional): implementations may include a convenience link to an online verifier.
- Footer bar: "Powered by Work References -- 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 (includes domain, selector).
Creator	String	"Work References"
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.

Tamper Detection

The Ed25519 signature is the primary tamper-detection mechanism. Any modification to the payload invalidates the signature. The payload hash in the metadata provides a quick integrity check — a verifier can recompute SHA-256 from the extracted payload and compare it to the embedded value.

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 that domain's DNS.
- Zero-Knowledge: The server stores only AES-256-GCM ciphertext. 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 content. An attacker cannot reuse 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 verification to Silver, not Gold. Offline signature verification remains possible using the stored public key.

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}._workreferences.{DOMAIN}.
- Wait for DNS propagation (typically 15 minutes to 48 hours).
- Sign references client-side. Never send the plaintext private key to the server.
- Generate a PDF with the payload, signature, domain, and selector embedded in metadata.
- Share the PDF with the candidate. The document is self-contained proof.

For Verifiers

- Receive a PDF reference from the candidate.
- Extract the payload and signature from the PDF metadata.
- Look up the public key via DNS using the domain and selector from the payload.
- Verify the Ed25519 signature. Gold = valid, Silver = no DNS record, Invalid = signature failed.

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 (often days to weeks), unreliable (key contacts may have left the organisation), and provides no assurance of authenticity. A reference received by phone or email could have been fabricated by the candidate.

Survey-Based Tools

Survey-based tools automate collection by sending questionnaires to referees. While faster than manual checking, they still rely on the referee's self-reported identity — there is no cryptographic proof that the person responding is who they claim to be. The reference data also remains 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. Work References achieves equivalent security guarantees using DNS — infrastructure that every organisation already manages.

How Work References Differs

- No central authority: Verification uses DNS, the same decentralised infrastructure that underpins email security (DKIM) and web certificates (CAA records).
- Truly portable: References are self-contained PDFs with all verification data embedded. The candidate carries them from job to job — no dependency on the issuing platform.
- Zero cost to verify: No account, subscription, or API key is needed for verification.
- Instant: Verification completes in under one second, compared to days or weeks for traditional checking.
- Revocable: Unlike paper references, issued references can be revoked at any time by the issuing employer.

13. 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.
