Network Working Group
Internet-Draft
Intended status: Informational
Expires: 25 October 2026

B. Chen
OpenNHP

23 April 2026

# Network-Infrastructure Hiding Protocol
# draft-opennhp-ztcpp-nhp-00

## Abstract

The Network-Infrastructure Hiding Protocol (NHP) is a cryptography-based session-layer protocol designed to operationalize Zero Trust principles by concealing protected network resources from unauthorized entities. NHP enforces authentication-before-connect access control, rendering IP addresses, ports, and domain names invisible to unauthorized users. This document defines the protocol architecture, cryptographic framework, message formats, and workflow to enable independent implementation of NHP. It represents the third generation of network hiding technology—evolving from first-generation port knocking to second-generation Single-Packet Authorization (SPA) and now to NHP with advanced asymmetric cryptography, mutual authentication, and scalability for modern threats. This specification also provides guidance for integration with Software-Defined Perimeter (SDP), DNS, FIDO, and Zero Trust policy engines.

## About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://OpenNHP.github.io/ietf-rfc-nhp/draft-opennhp-ztcpp-nhp.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-opennhp-ztcpp-nhp/.

Discussion of this document takes place on the ztcpp Independent Submission mailing list (mailto:ztcpp@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ztcpp/. Subscribe at https://www.ietf.org/mailman/listinfo/ztcpp/.

Source for this draft and an issue tracker can be found at https://github.com/OpenNHP/ietf-rfc-nhp.

## Status of This Memo

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

## Copyright Notice



## Table of Contents

1. Introduction . . . . . 4
2. Conventions and Definitions . . . . . 5
3. Design Objectives . . . . . 6
4. Relationship to TLS . . . . . 6
  4.1. OSI Layer Positioning . . . . . 6
  4.2. Key Differences . . . . . 7
  4.3. The Pre-Authentication Problem . . . . . 8
  4.4. Complementary Security Model . . . . . 8
  4.5. Vulnerabilities Addressed by NHP but Not TLS . . . . . 9
  4.6. Why Both Are Needed . . . . . 9
5. Threat Model . . . . . 9
  5.1. Reconnaissance and Scanning . . . . . 10
  5.2. Pre-Authentication Exploits . . . . . 10
  5.3. DDoS Attacks . . . . . 10
  5.4. Credential Theft and Replay . . . . . 10
  5.5. Man-in-the-Middle Attacks . . . . . 10
6. Architectural Overview . . . . . 10
  6.1. Core Components . . . . . 10
    6.1.1. NHP-Agent . . . . . 10
    6.1.2. NHP-Server . . . . . 11
    6.1.3. NHP-AC (Access Controller) . . . . . 11
    6.1.4. Authorization Service Provider (ASP) . . . . . 12
  6.2. Component Interactions . . . . . 12

6.3. Deployment Models . . . . . 12
6.3.1. Standalone Deployment . . . . . 12
6.3.2. Clustered Deployment . . . . . 13
6.3.3. Edge AC Deployment . . . . . 13
6.3.4. Multi-Tenant Deployment . . . . . 13
7. Protocol Workflow . . . . . 13
7.1. Control Plane vs Data Plane . . . . . 13
7.2. Workflow Steps . . . . . 13
7.3. Sequence Diagram . . . . . 14
8. Cryptographic Framework . . . . . 14
8.1. Cryptographic Primitives . . . . . 15
8.2. Noise Protocol Handshake Patterns . . . . . 15
8.2.1. XX Pattern (Default) . . . . . 15
8.2.2. IK Pattern (Performance Optimized) . . . . . 16
8.2.3. K Pattern (One-Way) . . . . . 16
8.3. Key Management . . . . . 16
8.3.1. Static Keys . . . . . 16
8.3.2. Ephemeral Keys . . . . . 16
8.3.3. Key Rotation . . . . . 16
9. Message Format . . . . . 17
9.1. Message Header . . . . . 17
9.1.1. Header Fields . . . . . 17
9.2. Message Types . . . . . 18
9.3. Message Definitions . . . . . 19
9.3.1. NHP-KPL (Keepalive) . . . . . 19
9.3.2. NHP-KNK (Knock) . . . . . 19
9.3.3. NHP-ACK (Acknowledge) . . . . . 19
9.3.4. NHP-AOP (AC Operations) . . . . . 20
9.3.5. NHP-ART (AC Result) . . . . . 20
9.3.6. NHP-ACC (Access) . . . . . 21
9.3.7. NHP-REG (Register) . . . . . 21
9.3.8. NHP-RAK (Register Acknowledge) . . . . . 22
9.3.9. NHP-LOG (Log) . . . . . 22
9.3.10. NHP-LAK (Log Acknowledge) . . . . . 23
10. Logging and Auditing . . . . . 23
10.1. Log Types . . . . . 23
10.2. Log Format . . . . . 23
10.3. Log Transmission . . . . . 24
10.4. Compliance Considerations . . . . . 24
11. Integration with SDP . . . . . 24
11.1. Integration Architecture . . . . . 24
11.2. Integration Process . . . . . 25
11.3. Benefits of NHP-SDP Integration . . . . . 25
12. Integration with DNS . . . . . 25
12.1. DNS Integration Architecture . . . . . 25
12.2. Integration Process . . . . . 26
13. Integration with FIDO . . . . . 26
13.1. FIDO Integration Flow . . . . . 26

13.2. Recovery and Fallback . . . . . 26
14. Security Considerations . . . . . 27
14.1. Infrastructure Invisibility . . . . . 27
14.2. Replay Attack Prevention . . . . . 27
14.3. Key Security . . . . . 27
14.4. Session Security . . . . . 27
14.5. Denial of Service Mitigation . . . . . 28
14.6. Limitations . . . . . 28
15. IANA Considerations . . . . . 28
16. Reference Implementation . . . . . 29
16.1. Implementation Characteristics . . . . . 30
16.1.1. Memory-Safe Language . . . . . 30
16.1.2. Cross-Platform Support . . . . . 30
16.1.3. Modular Architecture . . . . . 31
16.1.4. Cryptographic Implementation . . . . . 31
16.1.5. Performance Characteristics . . . . . 32
16.1.6. Open Source Governance . . . . . 32
16.2. Practical Use Case: StealthDNS . . . . . 32
17. References . . . . . 33
17.1. Normative References . . . . . 33
17.2. Informative References . . . . . 34
Acknowledgments . . . . . 34
Author’s Address . . . . . 34

## 1. Introduction

Since its inception in the 1970s, the TCP/IP networking model has prioritized openness and interoperability, laying the foundation for the modern Internet. However, this design philosophy also exposes systems to reconnaissance and attack. As Vint Cerf, who personally designed many of these components, stated, "We didn’t focus on how you could wreck this system intentionally."

Today, the cyber threat landscape has been dramatically reshaped by the rise of AI-driven attacks, which bring unprecedented speed and scale to vulnerability discovery and exploitation. Automated tools continuously scan the global network space, identifying weaknesses in real-time. Large Language Models (LLMs) can now autonomously exploit one-day vulnerabilities, and AI systems can generate working exploits for published CVEs in minutes. As a result, the Internet is evolving into a "Dark Forest," where *visibility equates to vulnerability*. In such an environment, any exposed service becomes an immediate target.

The Zero Trust model, which mandates continuous verification and eliminates implicit trust, has emerged as a modern approach to cybersecurity. Within this context, the Network-Infrastructure Hiding Protocol (NHP) offers a new architectural element: authenticated-before-connect access at the session layer.

NHP builds upon foundational work in the Cloud Security Alliance's Software-Defined Perimeter (SDP) and Single-Packet Authorization (SPA) frameworks, representing the third generation of network hiding technology:

* *First Generation - Port Knocking:* Simple port sequences vulnerable to interception and replay attacks.

* *Second Generation - SPA:* Encrypted single-packet authorization with improved security but limited scalability.

* *Third Generation - NHP:* Advanced asymmetric cryptography, mutual authentication, Noise Protocol-based key exchange, and enterprise-grade scalability.

This document outlines the motivations behind NHP, its design objectives, message structures, integration options, and security considerations for adoption within Zero Trust frameworks.

## 2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used throughout this document:

NHP Network-Infrastructure Hiding Protocol

NHP-Agent The client-side component that initiates NHP communication

NHP-Server The control-plane service that validates requests and makes access decisions

NHP-AC NHP Access Controller, the enforcement component near protected resources

SPA Single-Packet Authorization

SDP Software-Defined Perimeter

ZTA Zero Trust Architecture

ECC Elliptic Curve Cryptography

AEAD Authenticated Encryption with Associated Data

ASP Authorization Service Provider

PEP Policy Enforcement Point

KGC Key Generation Center

## 3. Design Objectives

The NHP protocol is designed to achieve the following objectives:

1. *Infrastructure Invisibility:* Eliminate unauthorized network visibility by enforcing authentication prior to session establishment. Protected resources remain invisible to unauthorized scanners and attackers.

2. *Session Layer Operation:* Operate at OSI Layer 5, complementing existing TCP, UDP, and QUIC transports without requiring changes to underlying network infrastructure.

3. *Decentralized Trust:* Support decentralized trust using asymmetric cryptography and ephemeral key exchange, eliminating single points of trust failure.

4. *Fine-Grained Access Control:* Enable context-based policy enforcement across heterogeneous environments, supporting least-privilege access.

5. *Integration Capability:* Integrate with existing Zero Trust controllers, SDP gateways, identity systems (IAM), DNS infrastructure, and FIDO authentication.

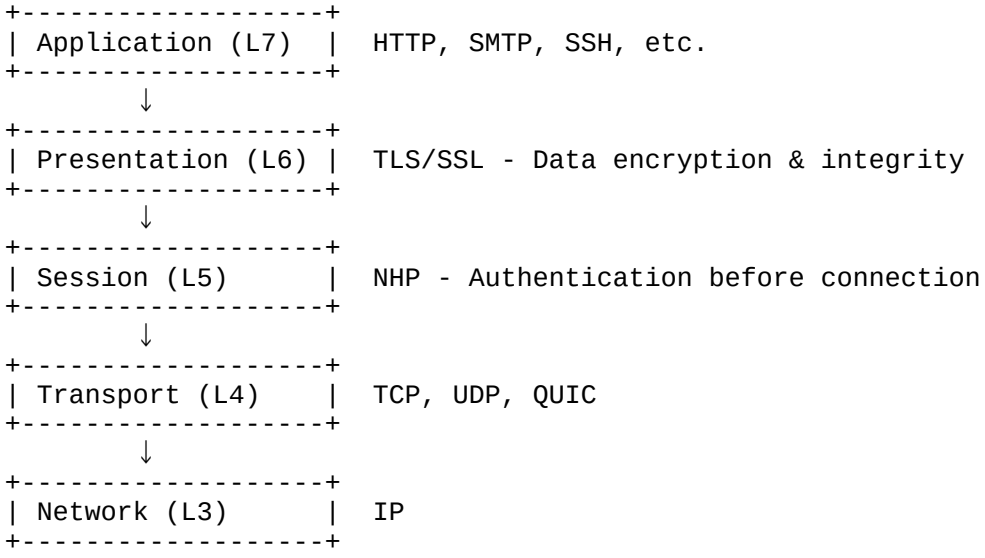
6. *Scalability:* Support enterprise-scale deployments with clustered servers, distributed access controllers, and multi-tenant isolation.

7. *AI Threat Mitigation:* Reduce the attack surface against AI-driven reconnaissance and exploitation by denying visibility before authentication.

## 4. Relationship to TLS

NHP and TLS (Transport Layer Security) are complementary protocols that operate at different OSI layers and serve distinct security purposes. This section clarifies their differences and how they work together.

### 4.1. OSI Layer Positioning




## 4.2. Key Differences

| Aspect | NHP (Layer 5) | TLS (Layer 6) |
|---|---|---|
| *Primary Purpose* | Infrastructure hiding and access control | Data encryption and integrity |
| *When Authentication Occurs* | BEFORE connection establishment | AFTER TCP connection established |
| *Service Visibility* | Services are INVISIBLE to unauthorized users | Services are VISIBLE, communication is encrypted |
| *Attack Surface* | Eliminates pre-authentication attack surface | Protects data in transit, but service ports remain exposed |
| *Port Exposure* | No ports exposed until authenticated | Ports must be open to initiate TLS handshake |
| *Vulnerability Window* | None—no connection without authentication | TLS handshake vulnerabilities can be exploited |

Table 1

## 4.3. The Pre-Authentication Problem

TLS provides excellent protection for data in transit, but it has a fundamental limitation: *the service must be reachable to initiate the TLS handshake*. This creates a pre-authentication attack window:

Traditional TLS Flow:

```
Attacker ———————— Open Port 443 ———————— TLS Handshake ———————— Authe
ntication
                    ↑
                Service is VISIBLE
                Port scan succeeds
                Pre-auth exploits possible
```

NHP + TLS Flow:

```
Attacker ———————— No Open Ports ———————— BLOCKED (Service Invisible)
                   ↑
                Cannot discover service
                Port scan fails

Authorized ———————— NHP Knock ———————— Port Opens ———————— TLS ———————
—— Application
User               ↑                    ↑
                Authenticated        Encrypted
                BEFORE connect       data transfer
```

## 4.4. Complementary Security Model

NHP and TLS are designed to work together, not replace each other:

1. *NHP provides:* Authentication-before-connect, infrastructure invisibility, access control

2. *TLS provides:* Data encryption, integrity verification, server authentication

A complete Zero Trust deployment SHOULD use both:

* *NHP* ensures only authorized users can discover and reach the service

* *TLS* encrypts all data exchanged after access is granted

### 4.5. Vulnerabilities Addressed by NHP but Not TLS

| Vulnerability Type | TLS Protection | NHP Protection |
|---|---|---|
| Port scanning and service discovery | None | Service invisible |
| Pre-authentication exploits (e.g., Heartbleed) | Vulnerable | No connection possible |
| TLS implementation bugs before handshake | Vulnerable | No handshake initiated |
| DDoS attacks on exposed services | Service reachable | Service hidden |
| Credential stuffing on login pages | Page accessible | Page invisible |
| Zero-day exploits before authentication | Service exposed | Service protected |

Table 2

### 4.6. Why Both Are Needed

NHP alone does not encrypt application data—it only controls access. TLS alone does not hide services—it only encrypts traffic. Together, they provide defense in depth:

* *Without NHP:* Attackers can scan, probe, and exploit services before any authentication occurs

* *Without TLS:* Authorized traffic would be transmitted in plaintext after NHP grants access

* *With Both:* Services are invisible to attackers, and all authorized traffic is encrypted

This layered approach aligns with Zero Trust principles: never trust, always verify, and minimize attack surface at every layer.

## 5. Threat Model

NHP is designed to mitigate the following threat categories:

### 5.1. Reconnaissance and Scanning

Automated scanning tools and AI-driven reconnaissance continuously probe Internet-facing services. NHP eliminates the ability to discover protected resources by requiring cryptographic authentication before any network visibility is granted.

### 5.2. Pre-Authentication Exploits

Many vulnerabilities can be exploited before authentication occurs. By enforcing authentication-before-connect, NHP prevents attackers from reaching vulnerable services.

### 5.3. DDoS Attacks

NHP reduces DDoS attack surface by hiding service endpoints. Attackers cannot target what they cannot discover.

### 5.4. Credential Theft and Replay

NHP uses ephemeral keys and timestamp-based nonces to prevent credential replay attacks. Each session requires fresh cryptographic material.

### 5.5. Man-in-the-Middle Attacks

Mutual authentication using asymmetric cryptography ensures both parties verify each other’s identity before establishing communication.

## 6. Architectural Overview

NHP operates as a distributed session-layer protocol that enforces authentication-before-connect access between clients and protected resources.

### 6.1. Core Components

#### 6.1.1. NHP-Agent

The NHP-Agent is a client-side process, SDK, or embedded module that initiates communication with the protected network. Its responsibilities include:

* Generating and sending NHP-KNK (Knock) messages to the NHP-Server

* Performing cryptographic key exchange using Noise Protocol handshakes

* Managing client identity credentials and device attestation

* Handling session lifecycle including keepalives and re-authentication

#### 6.1.2. NHP-Server

The NHP-Server is the core control-plane service responsible for:

* Receiving and validating NHP-KNK messages from NHP-Agents

* Authenticating the NHP-Agent identity and device posture

* Interfacing with external Authorization Service Providers (ASP) or IAM systems

* Evaluating access policies based on identity, context, and resource attributes

* Instructing NHP-AC components to open or close access paths

* Managing session state and expiration

Functionally, the NHP-Server maps to the *Policy Administrator* role defined in NIST SP 800-207 Zero Trust Architecture.

#### 6.1.3. NHP-AC (Access Controller)

The NHP-AC is the enforcement component residing logically or physically near protected resources. Its responsibilities include:

* Maintaining default-deny firewall rules for all protected resources

* Receiving NHP-AOP (AC Operations) commands from the NHP-Server

* Temporarily opening access paths for authorized NHP-Agents

* Automatically reverting to default-deny state when sessions expire

* Reporting access logs and status to the NHP-Server

The NHP-AC corresponds to the *Policy Enforcement Point (PEP)* in NIST SP 800-207 terminology.

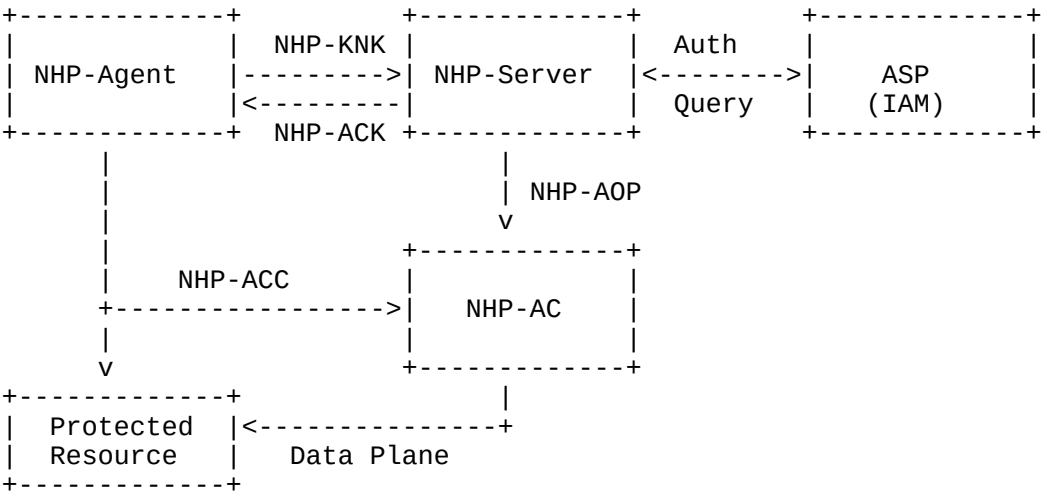
#### 6.1.4. Authorization Service Provider (ASP)

The ASP is an external identity and policy service that the NHP-Server queries for authorization decisions. This may include:

* Identity Providers (IdP) such as LDAP, Active Directory, or OIDC providers
* Policy Decision Points (PDP) implementing ABAC or RBAC policies
* Device posture assessment services
* Risk scoring engines

### 6.2. Component Interactions

The following diagram illustrates the relationship between NHP components:




### 6.3. Deployment Models

NHP components can be deployed in different configurations:

#### 6.3.1. Standalone Deployment

For small environments or testing scenarios, the NHP-Server and NHP-AC can coexist on the same host. This configuration simplifies setup while maintaining full protocol compliance.

#### 6.3.2. Clustered Deployment

In enterprise or cloud environments, multiple NHP-Servers can be deployed in a load-balanced cluster. Each server manages a pool of NHP-AC instances distributed across data centers or network segments. The NHP-Agent dynamically discovers the nearest NHP-Server through DNS or bootstrap configuration.

#### 6.3.3. Edge AC Deployment

Edge nodes (e.g., gateways, routers, or micro-segmentation agents) can host lightweight NHP-AC components. These edge ACs enforce fine-grained policies close to workloads, improving latency and fault isolation.

#### 6.3.4. Multi-Tenant Deployment

In service-provider or multi-cloud environments, each tenant can operate an independent NHP-Server while sharing an underlying AC infrastructure. The NHP protocol’s namespace isolation ensures complete tenant separation through identity-scoped keys and per-tenant policy databases.

## 7. Protocol Workflow

### 7.1. Control Plane vs Data Plane

The *Control Plane* carries cryptographic authentication and authorization information among NHP-Agent, NHP-Server, NHP-AC, and optional external ASP. Control plane messages are encrypted using Noise Protocol handshakes.

The *Data Plane* carries application data between the resource requester (NHP-Agent host) and the protected resource, but only after NHP-AC explicitly authorizes access.

This strict separation enforces the _authenticate-before-connect_ principle central to Zero Trust.

### 7.2. Workflow Steps

The complete NHP workflow consists of the following steps:

1. *Knock Request:* NHP-Agent sends NHP-KNK message to NHP-Server containing encrypted identity claims and access request.

2. *Authorization Query:* NHP-Server validates the cryptographic envelope and queries ASP for authorization decision.

3. *Authorization Response:* ASP returns authorization decision with granted permissions and session parameters.

4. *Door Opening:* NHP-Server sends NHP-AOP command to NHP-AC instructing it to open access for the specific NHP-Agent.

5. *AC Confirmation:* NHP-AC enforces the access rule and replies with NHP-ART confirming the operation.

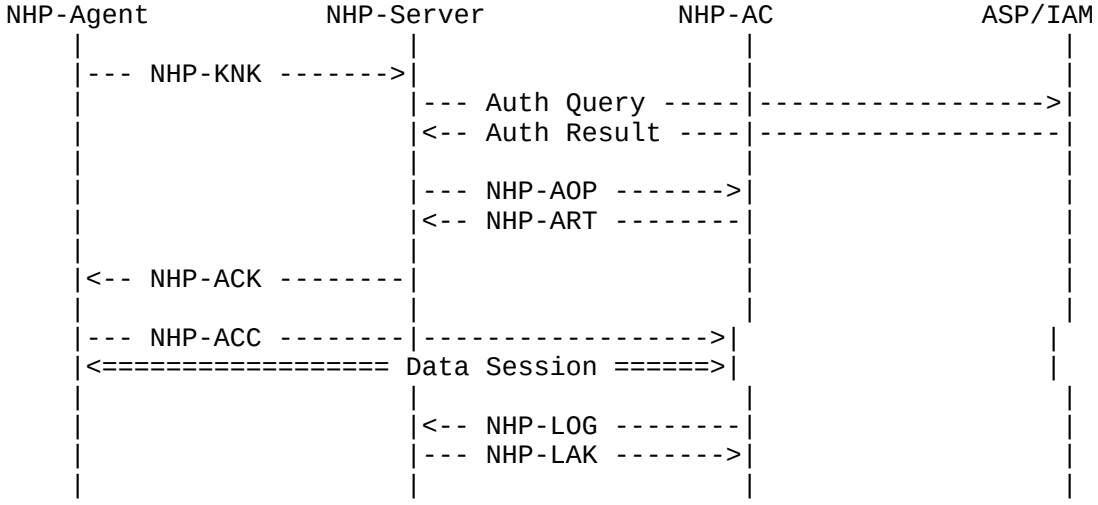
6. *Agent Notification:* NHP-Server sends NHP-ACK to NHP-Agent with access token and connection parameters.

7. *Resource Access:* NHP-Agent sends NHP-ACC to NHP-AC and establishes data plane connection to protected resource.

8. *Session Maintenance:* NHP-Server and NHP-AC maintain session state through NHP-KPL keepalive messages.

9. *Logging and Audit:* NHP-AC uploads access logs via NHP-LOG messages for compliance and auditing.

### 7.3. Sequence Diagram




## 8. Cryptographic Framework

NHP employs the Noise Protocol Framework [NoiseFramework] for all cryptographic operations. This section defines the required cryptographic primitives and handshake patterns.

### 8.1. Cryptographic Primitives

Implementations MUST support the following cryptographic primitives:

| Function | Algorithm | Reference |
|---|---|---|
| DH | Curve25519 | RFC 7748 |
| Cipher | ChaCha20-Poly1305 | RFC 8439 |
| Hash | SHA-256 | RFC 6234 |
| Key Derivation | HKDF | RFC 5869 |

Table 3

Implementations MAY additionally support:

| Function | Algorithm | Reference |
|---|---|---|
| DH | P-256 (secp256r1) | RFC 8422 |
| Cipher | AES-256-GCM | RFC 5116 |
| Hash | BLAKE2s | RFC 7693 |

Table 4

### 8.2. Noise Protocol Handshake Patterns

NHP supports the following Noise handshake patterns:

#### 8.2.1. XX Pattern (Default)

The XX pattern provides full forward secrecy and identity protection for both parties. It is the RECOMMENDED pattern for most deployments.

```
XX:
  -> e
  <- e, ee, s, es
  -> s, se
```

#### 8.2.2. IK Pattern (Performance Optimized)

The IK pattern is used when the NHP-Agent knows the NHP-Server's static public key in advance, reducing round trips.

```
IK:
  <- s
  ...
  -> e, es, s, ss
  <- e, ee, se
```

#### 8.2.3. K Pattern (One-Way)

The K pattern is used for one-way initiation where only the initiator needs to be authenticated by the responder.

```
K:
  <- s
  ...
  -> e, es, ss
```

### 8.3. Key Management

#### 8.3.1. Static Keys

Each NHP component maintains a static Curve25519 key pair:

* NHP-Agent: Used for client identity and authentication
* NHP-Server: Used for server identity and authentication
* NHP-AC: Used for secure communication with NHP-Server

Static public keys MUST be distributed through a secure out-of-band mechanism or registered through the NHP-REG message flow.

#### 8.3.2. Ephemeral Keys

Ephemeral keys are generated for each session to provide forward secrecy. Implementations MUST use cryptographically secure random number generators for ephemeral key generation.

#### 8.3.3. Key Rotation

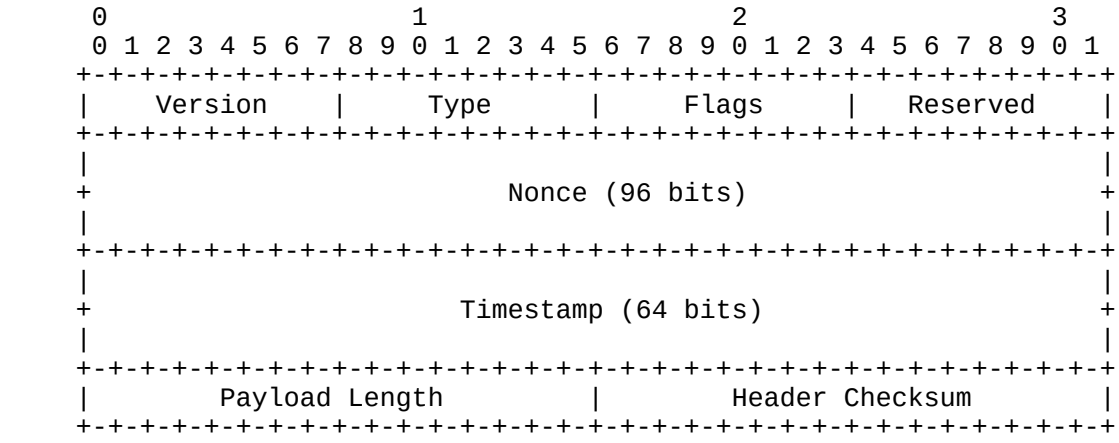
Static keys SHOULD be rotated periodically. The NHP-REG and NHP-RAK messages support key re-registration without service interruption.

# 9. Message Format

All NHP messages share a common header structure followed by an encrypted payload.

## 9.1. Message Header

The NHP message header is 32 bytes with the following structure:




### 9.1.1. Header Fields

Version (8 bits) Protocol version. Current version is 0x01.

Type (8 bits) Message type code. See Section 9.2.

Flags (8 bits) Control flags: * Bit 0: Compression enabled * Bit 1: Fragmentation flag * Bit 2: Priority message * Bits 3-7: Reserved

Reserved (8 bits) Reserved for future use. MUST be set to zero.

Nonce (96 bits) Unique nonce for AEAD encryption. MUST be unique per message within a session.

Timestamp (64 bits) UNIX epoch time in milliseconds. Used for replay protection.

Payload Length (16 bits) Length of the encrypted payload in bytes.

Header Checksum (16 bits) CRC-16 checksum of the header for integrity verification.

## 9.2. Message Types

| Type Code | Name | Direction | Description |
|---|---|---|---|
| 0x00 | NHP-KPL | Any | Keepalive message |
| 0x01 | NHP-KNK | Agent→Server | Knock request |
| 0x02 | NHP-ACK | Server→Agent | Knock acknowledgment |
| 0x03 | NHP-AOP | Server→AC | AC operation command |
| 0x04 | NHP-ART | AC→Server | AC operation result |
| 0x05 | NHP-LST | Agent→Server | Resource list request |
| 0x06 | NHP-LRT | Server→Agent | Resource list response |
| 0x07 | NHP-COK | Server→Agent | Cookie for session resumption |
| 0x08 | NHP-RKN | Agent→Server | Re-knock with cookie |
| 0x09 | NHP-RLY | Relay→Server | Relayed message |
| 0x0A | NHP-AOL | AC→Server | AC online notification |
| 0x0B | NHP-AAK | Server→AC | AC acknowledge |
| 0x0C | NHP-OTP | Any | One-time password request |
| 0x0D | NHP-REG | Agent→Server | Public key registration |
| 0x0E | NHP-RAK | Server→Agent | Registration acknowledgment |
| 0x0F | NHP-ACC | Agent→AC | Access request |
| 0x10 | NHP-LOG | AC→Server | Log upload |
| 0x11 | NHP-LAK | Server→AC | Log acknowledgment |

Table 5

### 9.3. Message Definitions

#### 9.3.1. NHP-KPL (Keepalive)

Keepalive messages maintain session state between components. The payload contains:

| Field | Size | Description |
|---|---|---|
| Session ID | 16 bytes | Current session identifier |
| Sequence | 4 bytes | Monotonic sequence number |

Table 6

#### 9.3.2. NHP-KNK (Knock)

The knock message initiates access request from NHP-Agent to NHP-Server. The encrypted payload contains:

| Field | Size | Description |
|---|---|---|
| User ID | Variable | Unique user identifier |
| Device ID | Variable | Unique device identifier |
| Device Fingerprint | 32 bytes | Device attestation hash |
| Requested Resources | Variable | List of resource identifiers |
| Context Data | Variable | Additional context (location, etc.) |

Table 7

#### 9.3.3. NHP-ACK (Acknowledge)

The acknowledge message confirms knock success and provides access parameters:

| Field | Size | Description |
|---|---|---|
| Status Code | 2 bytes | Result status |
| Session ID | 16 bytes | Assigned session identifier |
| Access Token | Variable | Token for NHP-AC access |
| AC Addresses | Variable | List of AC endpoints |
| Expiration | 8 bytes | Session expiration timestamp |
| Granted Resources | Variable | List of granted resource access |

Table 8

#### 9.3.4. NHP-AOP (AC Operations)

The AC operations message instructs NHP-AC to modify access rules:

| Field | Size | Description |
|---|---|---|
| Operation | 1 byte | OPEN (0x01) or CLOSE (0x02) |
| Agent Address | Variable | Source IP/port of authorized agent |
| Resource ID | Variable | Target resource identifier |
| Expiration | 8 bytes | Rule expiration timestamp |
| Access Token | Variable | Token for agent verification |

Table 9

#### 9.3.5. NHP-ART (AC Result)

The AC result message reports operation status:

| Field | Size | Description |
|---|---|---|
| Status Code | 2 bytes | Operation result |
| Operation ID | 16 bytes | Reference to NHP-AOP |
| Details | Variable | Additional status information |

Table 10

### 9.3.6. NHP-ACC (Access)

The access message is sent from NHP-Agent to NHP-AC to initiate data plane access:

| Field | Size | Description |
|---|---|---|
| User ID | Variable | User identifier |
| Device ID | Variable | Device identifier |
| Access Token | Variable | Token from NHP-ACK |
| Requested Service | Variable | Target service identifier |

Table 11

### 9.3.7. NHP-REG (Register)

The registration message registers NHP-Agent public key with NHP-Server:

| Field | Size | Description |
|---|---|---|
| User ID | Variable | User identifier |
| Device ID | Variable | Device identifier |
| Public Key | 32 bytes | Agent’s static public key |
| OTP | Variable | One-time password for verification |

Table 12

### 9.3.8. NHP-RAK (Register Acknowledge)

Confirms successful registration:

| Field | Size | Description |
|---|---|---|
| Status Code | 2 bytes | Registration result |
| Server Public Key | 32 bytes | Server’s static public key |
| Certificate | Variable | Optional server certificate |

Table 13

### 9.3.9. NHP-LOG (Log)

Log upload message from NHP-AC to NHP-Server:

| Field | Size | Description |
|---|---|---|
| AC ID | Variable | Access controller identifier |
| Log ID | 32 bytes | Unique log identifier (hash) |
| Log Content | Variable | Compressed log entries |

Table 14

#### 9.3.10. NHP-LAK (Log Acknowledge)

Confirms log receipt:

| Field | Size | Description |
|---|---|---|
| Log ID | 32 bytes | Received log identifier |

Table 15

## 10. Logging and Auditing

NHP provides comprehensive logging capabilities to support security monitoring, compliance, and forensic analysis.

### 10.1. Log Types

NHP defines the following log categories:

Access Logs Record all access attempts, including source identity, timestamp, requested resource, and decision outcome.

Authentication Logs Record authentication events including key exchanges, identity verification, and authentication failures.

Policy Logs Record policy evaluation decisions and the factors considered.

System Logs Record component health, configuration changes, and operational events.

### 10.2. Log Format

All NHP logs SHOULD use structured JSON format with the following mandatory fields:

```
{
  "timestamp": "2025-01-01T12:00:00.000Z",
  "log_type": "access",
  "component": "nhp-ac-01",
  "session_id": "abc123...",
  "user_id": "user@example.com",
  "device_id": "device-uuid",
  "source_ip": "192.0.2.1",
  "resource_id": "resource-001",
  "action": "access_granted",
  "details": {}
}
```

### 10.3. Log Transmission

NHP-AC components transmit logs to NHP-Server using NHP-LOG messages. Implementations MUST:

* Encrypt all log transmissions using the established Noise session
* Batch logs to reduce network overhead
* Implement retry logic for failed transmissions
* Store logs locally if transmission fails

### 10.4. Compliance Considerations

NHP logging supports compliance with:

* SOC 2 Type II audit requirements
* GDPR access logging requirements
* HIPAA audit trail requirements
* PCI-DSS logging requirements

## 11. Integration with SDP

NHP is designed to integrate seamlessly with existing Software-Defined Perimeter (SDP) deployments as defined in [CSA.SDP.Spec2.0].

### 11.1. Integration Architecture

In an SDP integration, NHP components map to SDP components as follows:

| NHP Component | SDP Component |
|---|---|
| NHP-Agent | SDP Initiating Host |
| NHP-Server | SDP Controller |
| NHP-AC | SDP Gateway |

Table 16

### 11.2. Integration Process

1. *Discovery:* SDP Controller advertises NHP-Server endpoint to SDP Initiating Hosts.

2. *Authentication:* SDP Initiating Host uses NHP-KNK to authenticate with NHP-Server instead of SPA.

3. *Authorization:* NHP-Server queries SDP Controller for policy decisions.

4. *Enforcement:* NHP-AC opens ports on SDP Gateway based on NHP-AOP commands.

### 11.3. Benefits of NHP-SDP Integration

* *Stronger Cryptography:* NHP’s Noise-based key exchange provides better forward secrecy than traditional SPA.

* *Mutual Authentication:* Both client and server authenticate each other.

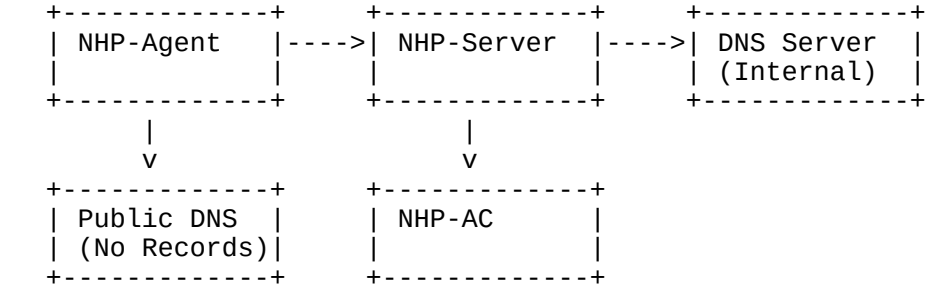
* *Scalability:* NHP’s architecture supports enterprise-scale deployments.

* *Extensibility:* NHP message types support richer interaction patterns.

## 12. Integration with DNS

NHP can integrate with DNS infrastructure to provide stealth resolution of protected resources.

### 12.1. DNS Integration Architecture




### 12.2. Integration Process

1. Protected resources have no public DNS records.

2. NHP-Agent authenticates with NHP-Server via NHP-KNK.

3. NHP-Server returns resource IP addresses in NHP-ACK only after successful authentication.

4. NHP-Agent can then connect to the resolved addresses.

This prevents DNS enumeration attacks and keeps resource addresses invisible to unauthorized users.

## 13. Integration with FIDO

NHP supports integration with FIDO2/WebAuthn for strong user authentication.

### 13.1. FIDO Integration Flow

1. User initiates NHP-KNK with FIDO assertion

2. NHP-Server validates FIDO assertion with FIDO server

3. Upon successful FIDO authentication, NHP-Server proceeds with access grant

### 13.2. Recovery and Fallback

For FIDO authentication failures, NHP supports fallback to:

* One-Time Password (OTP) via NHP-OTP message

* SMS/Email verification codes

* Recovery codes

# 14. Security Considerations

## 14.1. Infrastructure Invisibility

NHP ensures infrastructure invisibility by:

* Encrypting all control plane traffic using Noise Protocol
* Requiring mutual authentication before any resource visibility
* Maintaining default-deny firewall rules on all NHP-AC components
* Supporting ephemeral port allocation for data plane connections

## 14.2. Replay Attack Prevention

NHP prevents replay attacks through:

* Timestamp validation with configurable tolerance (RECOMMENDED: 60 seconds)
* Unique nonce per message
* Session-bound tokens that cannot be reused across sessions

## 14.3. Key Security

Implementations MUST:

* Use cryptographically secure random number generators for all key generation
* Store private keys in secure enclaves or HSMs where available
* Implement key rotation policies
* Securely erase key material when no longer needed

## 14.4. Session Security

* Sessions MUST have configurable expiration (RECOMMENDED default: 4 hours)
* Sessions MUST be revocable by NHP-Server
* Session tokens MUST be bound to client identity and IP address

### 14.5. Denial of Service Mitigation

NHP provides DoS resistance through:

* Cryptographic puzzles for computationally expensive operations
* Rate limiting on NHP-Server and NHP-AC
* Cookie-based session resumption to avoid repeated handshakes

### 14.6. Limitations

NHP does not protect against:

* Compromised endpoints with valid credentials
* Insider threats with legitimate access
* Attacks on the data plane after access is granted
* Social engineering attacks targeting user credentials

## 15. IANA Considerations

This document requests IANA to establish a new registry for NHP Message Types with the following initial values:

| Value | Name | Reference |
|---|---|---|
| 0x00 | NHP-KPL | This document |
| 0x01 | NHP-KNK | This document |
| 0x02 | NHP-ACK | This document |
| 0x03 | NHP-AOP | This document |
| 0x04 | NHP-ART | This document |
| 0x05 | NHP-LST | This document |
| 0x06 | NHP-LRT | This document |
| 0x07 | NHP-COK | This document |
| 0x08 | NHP-RKN | This document |
| 0x09 | NHP-RLY | This document |
| 0x0A | NHP-AOL | This document |
| 0x0B | NHP-AAK | This document |
| 0x0C | NHP-OTP | This document |
| 0x0D | NHP-REG | This document |
| 0x0E | NHP-RAK | This document |
| 0x0F | NHP-ACC | This document |
| 0x10 | NHP-LOG | This document |
| 0x11 | NHP-LAK | This document |

Table 17

Values 0x12-0xFF are reserved for future use.

## 16. Reference Implementation

An open-source reference implementation of NHP is available at:

https://github.com/OpenNHP/opennhp

### 16.1. Implementation Characteristics

The OpenNHP reference implementation is designed with the following characteristics:

#### 16.1.1. Memory-Safe Language

OpenNHP is implemented in *Go (Golang)*, a memory-safe programming language that eliminates entire classes of vulnerabilities common in C/C++ implementations:

* *No Buffer Overflows:* Go’s built-in bounds checking prevents buffer overflow attacks.
* *No Use-After-Free:* Automatic garbage collection eliminates dangling pointer vulnerabilities.
* *No Null Pointer Dereferences:* Go’s type system and nil handling prevent null pointer crashes.
* *Race Condition Detection:* Built-in race detector helps identify concurrency issues during development.

This choice aligns with recommendations from CISA, NSA, and other security agencies advocating for memory-safe languages in critical infrastructure software.

#### 16.1.2. Cross-Platform Support

OpenNHP provides native support across multiple platforms:

| Platform | Components | Description |
|---|---|---|
| Linux | Agent, Server, AC | Full production support for x86_64, ARM64 |
| Windows | Agent, Server, AC | Native Windows service integration |
| macOS | Agent | Desktop client with system integration |
| FreeBSD | Agent, Server, AC | BSD-family operating system support |
| Android | Agent (Library) | Mobile SDK for Android applications |
| iOS | Agent (Library) | Mobile SDK for iOS applications |

Table 18

#### 16.1.3. Modular Architecture

The implementation provides separate binaries for each NHP component:

* *nhp-agent:* Client-side agent for initiating NHP connections

* *nhp-server:* Control plane server for authentication and authorization

* *nhp-ac:* Access controller for policy enforcement

Each component can be deployed independently, enabling flexible deployment topologies from standalone to distributed enterprise configurations.

#### 16.1.4. Cryptographic Implementation

The reference implementation uses well-audited cryptographic libraries:

* *Noise Protocol:* flynn/noise library for Noise Framework handshakes

* *Curve25519:* golang.org/x/crypto for elliptic curve operations

* *ChaCha20-Poly1305:* Standard library crypto/cipher for AEAD encryption

* *HKDF:* golang.org/x/crypto/hkdf for key derivation

#### 16.1.5. Performance Characteristics

The Go implementation provides:

* *Low Latency:* Typical NHP handshake completes in under 10ms on local networks

* *High Throughput:* Single NHP-Server can handle thousands of concurrent sessions

* *Minimal Footprint:* Agent binary under 15MB, low memory consumption

* *Concurrent Design:* Goroutine-based concurrency for efficient resource utilization

#### 16.1.6. Open Source Governance

The OpenNHP project operates under the Apache 2.0 license, fostering community collaboration and transparent development to accelerate adoption and ensure rigorous peer review of its security mechanisms.

### 16.2. Practical Use Case: StealthDNS

StealthDNS is a Zero Trust DNS client powered by OpenNHP that demonstrates practical application of the NHP protocol for DNS-level infrastructure hiding. It is available at:

https://github.com/OpenNHP/StealthDNS

StealthDNS implements the NHP-DNS integration described in this specification, providing:

* *Invisible DNS Resolution:* Protected domains have no public DNS records. Only authenticated clients can resolve hidden service addresses.

* *NHP-Powered Authentication:* Uses the OpenNHP library to perform cryptographic NHP knocking before DNS resolution.

* *Transparent Local Resolver:* Runs as a local DNS resolver (127.0.0.1:53), requiring no application changes.

* *Cross-Platform Support:* Available on Windows, macOS, Linux, Android, and iOS.

The StealthDNS workflow demonstrates the authenticate-before-connect principle:

1. Application performs DNS lookup for a protected domain.

2. StealthDNS checks if the domain is NHP-protected.

3. If protected, StealthDNS performs NHP knock with identity and device context.

4. Upon successful authentication, the NHP Controller returns ephemeral address mappings.

5. StealthDNS returns valid DNS records only to authorized clients.

6. Unauthorized clients receive NXDOMAIN— the service remains invisible.

This enforces *identity before visibility* and *authorization before connectivity*, demonstrating real-world application of NHP principles.

# 17. References

## 17.1. Normative References

[NoiseFramework] Perrin, T., "The Noise Protocol Framework", 2018, <https://noiseprotocol.org/noise.html>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9180] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/rfc/rfc9180>.

### 17.2. Informative References

[CSA.NHP.Whitepaper]
Cloud Security Alliance, "Stealth Mode SDP for Zero Trust Network Infrastructure: Introducing the Network-Infrastructure Hiding Protocol (NHP)", 2026.

[CSA.SDP.Spec2.0]
Cloud Security Alliance, "Software Defined Perimeter Specification v2.0", 2022.

[NIST.SP.800-207]
Rose, S., Borchert, O., Mitchell, S., and S. Connelly, "Zero Trust Architecture", NIST Special Publication 800-207, 2020.

## Acknowledgments

This work builds upon foundational research from the Cloud Security Alliance (CSA) Zero Trust Working Group, particularly the "Stealth Mode SDP for Zero Trust Network Infrastructure" whitepaper [CSA.NHP.Whitepaper]. The authors acknowledge the contributions of the CSA Zero Trust Research Working Group.

The authors would also like to thank the China Computer Federation (CCF) for their collaborative support, and the OpenNHP open source community for their contributions, testing, and feedback on early implementations of the Network-Infrastructure Hiding Protocol.

## Author’s Address

Benfeng Chen
OpenNHP
Email: benfeng@gmail.com