cloud security alliance®

Stealth Mode SDP for Zero Trust Network Infrastructure

Introducing the Network-Infrastructure Hiding Protocol (NHP)

Authors:

- Log in to your Google account to access full editing permission.
- Change from Editing to Suggesting in the upper right of the Google doc to track each author's edits.

Reviewers/Visitors:

- If you have a Google Account, please login before commenting. Otherwise, please note your name and affiliation in the comment you leave.
- Use the Comments or Suggesting features on Google Docs to leave feedback on the document. Suggestions will be written in and identified by your Google Account. To use the comments feature, highlight the phrase you would like to comment on, right-click, and select "Comment" (or Ctrl+Alt+M). Or, highlight the phrase, select "Insert" from the top menu, and select "Comment." All suggestions and comments will be reviewed by the editing committee.

For more information about Google's Comments feature, please refer to http://support.google.com/docs/bin/answer.py?hl=en&answer=1216772&ctx=cb&src=cb&cbid=-rx63b0fx4x0v&cbrank=1

Acknowledgments

The CSA Zero Trust Research Working Group

The Zero Trust research and guidance framework encompasses cloud and on-premises environments, mobile endpoints, Internet of Things (IoT), and operational technology (OT). The Cloud Security Alliance (CSA) Zero Trust Working Group aims to:

- Develop and promote Zero Trust best practices as a modern, essential approach to information security that aligns with cloud environments
- Provide industry education and thought leadership on various Zero Trust approaches, highlighting their strengths and limitations to enable organizations to make informed decisions based on their specific requirements
- Maintain a vendor-neutral perspective when evaluating Zero Trust architectures and implementation methodologies for mature deployments
- Establish technically sound positions and recommendations on Zero Trust principles while preserving vendor neutrality
- The workstream(s) for this document is ZT5 - Pillar: Networks, led by Michael Roza and Vinotth Ramalingam

Lead Author(s)

Benfeng Chen
Justin Posey
Yuanyuan Liu

Contributors

Michael Roza
Leon Zeng
Jason Garbis

Reviewers

Justin Bowen
Philip Griffiths
Prashant Khanwale
Surendra Narang
Srinivasa Ravi Teja Peri
Jen Trahan
George Chi
Matias Katz
Vaibhav Malik
Dharnisha Narasappa
Venkataramana Ragothaman
Xinpi D
Washima Tuleun
Dr. Victor Monga
Chinaza Obidike
Shashank Shelat

CSA Staff

Erik Johnson

Table of Contents

Abstract

Our core TCP/IP networking systems and protocols have been with us since the 1970s and have in many ways served us well. Their inherent openness and interoperability have sparked incredible innovation and significantly changed our world. However, these systems were designed to facilitate easy connection, rather than to fend off malicious actors. As Vint Cerf, who personally designed many of these components, stated, “We didn’t focus on how you could wreck this system intentionally. You could argue with hindsight that we should have, but getting this thing to work at all was non-trivial.”[7] It should be clear that TCP/IP’s default network visibility has enabled much of today’s malicious activity. Given our current threat landscape and the widespread adoption of Zero Trust as a set of principles and best practices, we believe that we now have an imperative to pivot our core networking technologies to a default-deny stance—one that aligns with emerging concepts like Gartner’s preemptive cybersecurity[12], which emphasizes denying, deceiving, and disrupting threats before they can launch or succeed, shifting from reactive detection to proactive prevention.

The Network-infrastructure Hiding Protocol (NHP) introduces an innovative Zero Trust security approach that significantly reduces the attack surface and prevents unauthorized access before exploitation can occur. NHP builds upon and extends the Single-Packet Authorization (SPA) technology initially outlined in the Cloud Security Alliance (CSA) Software-Defined Perimeter (SDP) specification, representing the third generation of network hiding technology—evolving from first-generation port knocking (simple port sequences vulnerable to interception) to second-generation SPA (encrypted single-packet authorization) and now to NHP with advanced cryptography, mutual authentication, and scalability for modern threats. The protocol is developed as an open-source initiative under OpenNHP (https://github.com/OpenNHP/opennhp), fostering community collaboration and transparent development to accelerate adoption and ensure rigorous peer review of its security mechanisms. NHP is designed to enhance security across both cloud-native and on-premises environments, offering robust protection for hybrid and multi-cloud deployments as well. This whitepaper presents NHP as a strategic solution for protecting network infrastructures against reconnaissance, AI-driven scanning, DDoS attacks, and pre-authentication exploits, while significantly improving the reliability, security, and performance of Zero Trust networks through modern cryptographic constructs and serving as a key component in a defense-in-depth strategy, with comprehensive technical specifications to support its implementation.

Target Audience

This whitepaper is intended for cybersecurity professionals, network architects, system administrators, and decision-makers responsible for securing enterprise networks, cloud environments, and critical infrastructure. It provides valuable insights for organizations adopting Zero Trust security principles and looking to mitigate cyber threats. Additionally, it serves as a technical reference for developers and engineers implementing NHP in security frameworks.

Executive Summary

TCP/IP networking protocols, foundational since the 1970s, enabled massive innovation through their open, interoperable design, but they were never built to defend against malicious actors. Once a strength, this inherent visibility has become a liability in the modern era, where AI-powered tools are increasingly capable of autonomously discovering and exploiting vulnerabilities at machine speed. As cyber threats become more automated and sophisticated, shifting from a default-open to a default-deny networking model becomes urgent. The Network-infrastructure Hiding Protocol (NHP) addresses this challenge by building on technologies like Single-Packet Authorization (SPA) to conceal network resources and proactively block unauthorized access. This whitepaper introduces NHP as a next-generation targeted and strategic defense mechanism, Zero Trust-aligned protocol designed to safeguard network infrastructure against traditional and AI-driven threats, with full technical specifications for implementation.

NHP provides a simple and effective way to apply Zero Trust principles to core networking infrastructure. It builds on the Single-Packet Authorization (SPA) concepts within the Cloud Security Alliance (CSA) Software-Defined Perimeter (SDP) specification to deliver a more scalable, resilient, and cryptographically sophisticated solution. NHP renders network resources invisible to unauthorized users, dramatically reducing the attack surface (for example, hiding all ports and services compared to SPA's exposure of a single UDP port (typically 62201)) and significantly mitigating malicious reconnaissance and exploitation attempts. This ties to enhanced "bits of security" with the Elliptic Curve Diffie-Hellman (ECDH) algorithm, detailed later (see Cryptographic Algorithms and Frameworks), while eliminating man-in-the-middle threats from SPA's public key distribution, boosting performance and reliability through smaller, efficient keys. Combined with Noise and a large pre-shared key, NHP also supports quantum-resistant cryptography, as outlined in standards like RFC 8784, offering qualitative improvements in security, performance, and reliability alongside these quantitative gains. NHP advances beyond SPA by introducing:

- **Advanced Cryptographic Security:** This type of security utilizes modern cryptographic algorithms, such as Certificate-Less Public Key Cryptography (CL-PKC) and the Noise Protocol Framework, for mutual authentication and message encryption, minimizing the risk of man-in-the-middle attacks, though Identity-Based Cryptography (IBC) introduces key escrow risks due to the central Private Key Generator, which should be mitigated with safeguards like distributed key management or auditing
- **Enhanced Reliability:** Unlike SPA, NHP provides explicit authentication feedback, enabling clients to establish secure connections without multiple connection attempts
- **Scalability and Performance:** Separates authentication and access control functions, allowing horizontal scalability for enterprise-level deployments, though this decoupling introduces synchronization complexity that can be managed through protocols like keepalive messaging and distributed consensus mechanisms
- **Comprehensive Network Obfuscation:** This technique conceals server ports (like SPA), IP addresses, and domain names, creating a robust defense against reconnaissance, DDoS attacks, and DNS hijacking by tying DNS resolution directly to a

successful authentication handshake, embodying the “authenticate-before-connect” principle
- **Seamless Integration with Zero Trust Architectures:** Functions as an authenticate-before-connect mechanism that complements Software-Defined Perimeter (SDP), DNS security, and FIDO authentication to strengthen overall network security

Key Benefits of NHP

- Neutralizes cyber threats by preventing reconnaissance and automated exploitation attempts before they reach critical assets
- Minimizes the attack surface by making protected resources invisible to unauthorized users, while providing no feedback to enhance plausible deniability and reduce fingerprinting by potential attackers
- Strengthens Zero Trust implementation through rigorous authentication and authorization at the session layer
- Enhances compliance and auditing capabilities with comprehensive logging and seamless integration with SIEM systems
- Provides future-proof security through the planned adoption of post-quantum cryptography (PQC) to address emerging threats
- Delivers efficient performance on low-end devices and slow networks through the use of lightweight EC cryptographic algorithms, offering faster key exchange compared to RSA-based approaches in SPA
- Improves operational resilience and efficiency by silently discarding unauthorized traffic, reducing system load and bandwidth usage while ensuring stable performance for legitimate users, supporting business continuity and resource optimization

Introduction

Traditional network security models, based on perimeter defenses and implicit trust, have become liabilities in today's evolving threat landscape, as the openness and interoperability of TCP/IP—while catalyzing enormous innovation—fail to counter modern malicious actors. Unlike VPNs, which rely on trusted tunnels, or TLS/mTLS, which secure communication but expose endpoints, NHP adopts a default-deny approach to hide resources until authenticated. The emergence of AI-powered tools has further accelerated these risks, enabling adversaries to discover and exploit weaknesses autonomously at machine speed.

This convergence of threats demands a new paradigm—one that not only detects and responds to attacks but also prevents them by default. **Zero Trust Architecture (ZTA)** addresses this challenge by enforcing a "never trust, always verify" principle. It rejects implicit trust based on network location or prior authentication and instead mandates continuous validation of identity, device, and context.

The **Network-infrastructure Hiding Protocol (NHP)** builds on this Zero Trust foundation by addressing a critical gap: the visibility of network infrastructure before authentication. NHP ensures that protected systems remain completely hidden—undetectable to unauthorized users—until identity and access control policies are satisfied. Unlike legacy protocols that expose services by default, NHP enforces a **default-deny** approach at the network session layer, making exploitation impossible without prior trust establishment.

Evolving from technologies like **Single-Packet Authorization (SPA)** and defined within the Cloud Security Alliance (CSA) **Software-Defined Perimeter (SDP)** framework, NHP represents the third generation of network hiding protocols. It introduces stronger cryptographic mechanisms, real-time authentication feedback, and scalability for modern cloud-native and distributed environments.

In an era where both opportunistic and AI-driven attackers operate at scale, visibility is vulnerability. NHP offers a path forward—transforming the network itself into an active enforcer of Zero Trust principles by eliminating exposure until trust is verified.

The Growing Threat of AI-Powered Cyber Threats

The integration of artificial intelligence into cyber operations has fundamentally transformed the threat landscape. While AI enhances defensive capabilities, adversaries are increasingly leveraging AI-driven tools to automate and intensify attacks. This section examines the emerging categories of AI-powered threats and their implications for network security. Recent research has demonstrated AI systems' growing capability to discover and exploit zero-day vulnerabilities independently:

- OpenAI's o3 model has independently discovered a real zero-day vulnerability (CVE-2025-37899) in the Linux kernel [8].

- Google's Big Sleep AI agent autonomously discovered a previously unknown, exploitable zero-day memory-safety bug in a SQLite production release [2].
- A 2024 study by UIUC researchers found that GPT-4-powered agents could autonomously exploit 87% of publicly disclosed vulnerabilities [3].
- Artificial intelligence systems can automatically generate functional exploits for newly published Common Vulnerabilities and Exposures (CVEs) in just 10-15 minutes at approximately $1 per exploit [4].

The emergence of AI-powered threats has several critical security implications:

- **Speed of Execution:** The average time from vulnerability discovery to exploitation has decreased from days to seconds. According to a recent CrowdStrike report, the fastest eCrime breakout time has decreased by 88% over the past two years and continues to decline rapidly [9]. This acceleration renders traditional security models, which depend on human detection and response, fundamentally insufficient
- **Scale of Operations:** AI enables adversaries to conduct attacks at an unprecedented scale. A single AI-driven system can simultaneously scan and probe millions of potential targets, operating 24/7 continuously without interruption. Moreover, these automated attacks can dynamically tailor their tactics in real time, customizing payloads and exploitation methods for each target based on observed conditions
- **Adaptive Capabilities:** AI-powered threats exhibit a high degree of resilience, capable of dynamically adjusting their tactics in response to defensive measures. Many incidents demonstrate that, when initial attack methods fail, AI systems can autonomously learn and refine their strategy, often increasing the likelihood of success in subsequent attempts. This evolving behavior represents a significant shift from traditional, static tactics
- **Autonomous Exploitation:** Advanced AI agents can independently discover, prioritize, and exploit vulnerabilities without human direction. Demonstrated in multiple 2024–2025 cases (e.g., OpenAI o3, Google Big Sleep), this represents a qualitative shift in offensive capability
- **Deepfake Attacks and AI-Powered Social Engineering:** Attackers use AI deep learning models to generate highly convincing impersonations of employees or executives, bypassing human identification via voice/video cloning (requiring just 3-5 seconds of sample audio). According to the IRONSCALES Fall 2025 Threat Report, 85% of organizations experienced at least one deepfake-related incident in the past 12 months, with a ~10% year-over-year increase reflecting escalating financial losses—over $200 million in Q1 2025 alone.

Recent industry analysis reinforces the strategic urgency of adopting proactive Zero Trust measures such as network-infrastructure hiding. According to Gartner's 2025 prediction, the cybersecurity landscape is undergoing a fundamental shift driven by GenAI-enabled offensive automation [13]. Gartner states that "in the age of GenAI, preemptive capabilities — not detection and response — are the future of cybersecurity," emphasizing that organizations must assume threats will evolve faster than traditional monitoring and incident response can keep up. Instead of relying primarily on visibility, alerts, and post-compromise containment, Gartner highlights the need for architectural controls that eliminate exposure, reduce attack surfaces,

and prevent adversaries (including autonomous AI agents) from ever seeing or reaching protected assets. This aligns directly with the Zero Trust principle of “never trust, always verify” and underscores why techniques such as Software-Defined Perimeter (SDP) and NHP-based resource hiding are becoming foundational: they replace a reactive security posture with a prevent-first, visibility-minimized, cryptography-backed enforcement architecture designed to resist both human attackers and machine-speed exploitation.

Limitations of NHP Against AI Threats

While NHP effectively mitigates pre-authentication AI-driven threats like reconnaissance and scanning, it has limitations against post-authentication attacks, such as runtime exploits or behavioral anomalies. Complementary tools, including runtime exploit detection systems and behavioral analytics platforms, are essential to address these risks, ensuring a comprehensive defense against advanced AI-powered adversaries.

Assumptions and Limitations

- NHP assumes clients securely manage cryptographic keys (e.g., IBC private keys) to prevent unauthorized access
- NHP does not replace application-layer controls post-admission; additional security measures are required for runtime protection

Threat Surface Reduction by Need to Know

Prevention is the best medicine. In a landscape where external threats evolve faster than ever, reactive security is no longer sufficient. Effective defense begins by raising a digital drawbridge, ensuring attackers never gain visibility into network assets unless authorized on a need-to-know basis. This demands preventive controls that enforce Zero Trust principles at machine speed, resist AI-powered threats, and respond autonomously to emerging risks. NHP exemplifies this approach, concealing protected systems from unauthorized view and blocking reconnaissance efforts before they escalate into attacks. For instance, Microsoft’s Secure Future Initiative demonstrated how proactive Zero Trust measures, including automated patching and continuous verification, prevented over 91 million potential vulnerabilities in 2024, averting breaches through early threat disruption.

Threat Model

NHP is designed to address specific threat vectors within the cybersecurity landscape while requiring complementary solutions for comprehensive protection. Understanding this scope is critical for effective implementation.

Threats effectively mitigated by NHP:

- **Network Reconnaissance and Discovery**

    - NHP renders infrastructure components invisible to unauthorized entities, preventing attackers from identifying potential targets
    - Particularly effective against automated scanning and AI-powered asset discovery tools
    - Counters both passive and active reconnaissance techniques
- **Zero-Day Exploitation at the Network Layer**
    - By concealing network assets, NHP significantly reduces the chance of exploiting vulnerabilities or default/weak credentials in network services and exposed interfaces, though authorized endpoint communications may remain visible to attackers with network access (e.g., on a compromised wireless network) unless encrypted at higher layers
    - Especially valuable for protecting against vulnerabilities in common network protocols and services
- **DNS-based Attacks**
    - Mitigates DNS hijacking, cache poisoning, and related attacks through authenticated, encrypted DNS resolution
    - Prevents unauthorized DNS-based reconnaissance
- **Distributed Denial of Service (DDoS)**
    - Significantly reduces the DDoS attack surface by concealing IP addresses and requiring authentication before establishing a connection
    - Prevents volumetric attacks against hidden infrastructure components
- **Identity-Centric Session Attribution**
    - NHP authenticates before connecting, so trust is established earlier in the stack
    - Threat monitoring is extended deeper down the stack
- **Lateral Movement in Breached Networks**
    - NHP maintains infrastructure invisibility even inside corporate networks, preventing attackers from scanning and discovering additional systems after an initial compromise
    - NHP enforces the **least-privilege** principle of Zero Trust, limiting the blast radius of a single breached endpoint by eliminating lateral movement paths
- **Replay Attacks**
    - Protects against adversaries attempting to reuse captured authentication packets to gain unauthorized access
    - NHP uses session-specific cryptographic keys and counters, rendering every handshake unique and instantly invalidating replayed packets

The SDP Framework and SPA Technology: A Historical Basis for NHP

Overview of Software-Defined Perimeter (SDP)

Software-Defined Perimeter (SDP) [1] is a security framework developed by the Cloud Security Alliance (CSA) to provide dynamic, identity-centric access controls for network security. SDP

enforces Zero Trust principles by verifying identities before granting access to resources, making network assets invisible to unauthorized entities. This approach mitigates the risks associated with traditional perimeter-based security models, which rely on static defenses such as firewalls and VPNs.

Key characteristics of SDP include:

- **Pre-Authentication Access:** Only authenticated users can see and access network resources
- **Separation of Control and Data Planes:** Authentication and authorization occur independently from data traffic to enhance security
- **Granular Policy Enforcement:** Access policies are based on contextual factors, including identity, device posture, and risk assessment
- **Cloud- and Hybrid-Friendly Architecture:** SDP integrates seamlessly with both on-premises and cloud environments

Understanding Single-Packet Authorization (SPA)

Single-Packet Authorization (SPA) is a key mechanism within SDP, designed to prevent unauthorized network discovery and reconnaissance. It functions by requiring a cryptographically secure authentication packet before allowing any connection attempts. If a requestor fails to send a valid SPA packet, the system remains completely silent, denying visibility into network assets. The key benefits of SPA include:

- **Stealth Mode Operation:** Network resources remain hidden until authentication is completed
- **DDoS Resistance:** SPA mitigates volumetric attacks by filtering out unauthorized requests before engaging processing resources
- **Reduced Attack Surface:** Unlike traditional firewalls that expose open ports, SPA conceals all entry points from attackers

Despite its effectiveness, SPA has certain limitations:

- **Scalability Issues:** Managing a high volume of SPA requests in large-scale environments can introduce latency
- **Risk of Replay Attacks:** If not correctly implemented, attackers can intercept and replicate SPA packets; while timestamps and nonces in standard implementations (e.g., fwknop) mitigate this by ensuring packet freshness, they require loose time synchronization between client and server, introducing risks from clock drift that could reject legitimate requests or necessitate wider tolerance windows
- **Limited Flexibility:** SPA relies on predefined entry points, which can introduce complexity in dynamic cloud-based deployments
- **Lack of Cryptographic Agility:** Relying on static algorithms (e.g., fixed AES-128-CBC encryption and HMAC-SHA256 authentication in common implementations) complicates updates for emerging threats like quantum computing

These challenges led to the development of **NHP**, which builds upon SPA principles while addressing its limitations with enhanced scalability, cryptographic agility, and dynamic access control mechanisms.

NHP: Cryptographic Advancements Beyond SPA for Zero Trust Security

NHP is a lightweight, cryptography-driven Zero Trust networking protocol at the OSI 5th layer. It offers an enhanced alternative to SPA, featuring an improved architecture that boosts security, reliability, and scalability. In addition to its role as a component of SDP, NHP can also function as a versatile "authenticate-before-connect" secure handshake protocol for various user scenarios outside of SDP.

NHP builds upon earlier research in network hiding technology, utilizing a modern cryptographic framework and architecture to ensure security and high performance, thereby overcoming the limitations of previous technologies.

Table 1. Network Hiding Technology evolution

Network Hiding Technology	**1st Generation**	**2nd Generation**	**3rd Generation**
Protocol	Port knocking	SPA	NHP
Authentication	Port sequences	Shared secrets	Modern cryptography

The advantages of NHP are listed below:

- **Security:** As discussed, the SPA packet may be captured or altered during network transmission, which can lead to potential MITM risks. Some SPA implementations utilize PKI/RSA to mitigate this problem; however, the computational cost of RSA signature verification and the complexity of Public Key Infrastructure (PKI) key distribution render this approach too costly. In NHP, the modern Noise Protocol Framework is introduced for message encryption and mutual verification, providing a proven, secure, and fast solution. Instead of PKI, Identity-Based Cryptography (IBC) is introduced in NHP to simplify key management and reduce the cost of key distribution. However, organizations should consider IBC’s inherent risks, such as key escrow (where the central Private Key Generator can access private keys) and trust anchor centralization (increasing the impact of a single point of compromise), and mitigate them through appropriate governance and redundancy measures
- **Reliability:** The client sending the SPA packet is unaware of the outcome of SPA authentication. Consequently, it must repeatedly attempt to establish a connection to the protected server (Controller or Gateway) at arbitrary intervals after sending the SPA packet. This approach relies heavily on network speed and may fail under poor network

conditions, resulting in unreliable performance across different network environments. In NHP, if the authentication is successful, the client receives a confirmation message with the port status and opening time of the protected server, allowing the client to know precisely when to establish the connection. If the authentication fails, NHP maintains a "silent" mechanism, and the client receives no response

- **Scalability:** The SPA server is typically deployed on protected servers (Gateway or Controller) and provides two major features: 1) SPA packet decryption and user/device authentication, and 2) network access control. In scenarios with many concurrent clients, the SPA packet decryption and user/device authentication processes—often relying on computationally intensive RSA algorithms—can consume significant bandwidth and computing resources on the hosted servers, limiting the service's scalability. While SPA could theoretically leverage horizontal scaling with modifications (e.g., load balancing), its traditional design tightly couples these functions, making it challenging without deviating from standard implementations. In NHP, these two features are decoupled into separate components, which can be deployed independently in a cluster of servers, achieving high scalability
- **Lightweight Design:** NHP employs efficient cryptographic algorithms (e.g., ECC with 256-bit keys and the Noise Protocol Framework) that reduce computational overhead and key sizes compared to SPA's RSA-based approaches, enabling high performance on low-end devices and slow networks while maintaining strong security
- **Extensibility:** While the benefits of SPA apply to a broad range of network services, SPA was not designed to be an extensible protocol supporting various user scenarios beyond SDP. NHP, on the other hand, is designed as a general-purpose connection management protocol for the 5th layer of the network. In addition to hiding server ports, it can hide the IP address by providing private domain name resolution. It supports any scenario that requires "authenticate-before-connect" secure communications

The table below demonstrates that NHP enhances and extends SPA by integrating modern cryptographic frameworks, bidirectional communication, and more powerful network obfuscation capabilities. It also provides a detailed comparison between SPA and NHP, highlighting improvements in security, scalability, and interoperability.

Table 2. NHP vs. SPA

Aspect	**SPA**	**NHP**	**NHP Advantages**
Architecture	SPA packet decryption and user/device authentication are coupled with network access control on the same server	Authentication and access control are decoupled, allowing for separate deployment to enhance scalability	**Scalability and Security:** Separating authentication from access control enhances performance and makes it more difficult for attackers to locate protected servers

Communication Model	Single direction	Bi-directional	**Reliability:** NHP provides explicit authentication feedback, reducing unnecessary retries
Cryptographic Framework	Uses shared secrets or basic PKI	Uses modern cryptographic frameworks, such as the Noise Protocol Framework + IBC	**Security and Performance:** NHP enhances authentication security while reducing the computational burden
Infrastructure Hiding	Hides only server ports	Can be used to hide server IP addresses, domain names, and ports	**Stronger Defense:** Prevents reconnaissance attacks, DNS hijacking, and DDoS attacks
Replay Attack Resistance	Susceptible if not correctly implemented	Uses the Noise Protocol Framework for mutual verification to prevent replay attacks	**Enhanced Security:** Prevents replay attacks via dynamic session key generation
Authentication Feedback	No explicit feedback is provided; clients must retry until access is granted	Clients receive confirmation upon successful authentication, reducing unnecessary attempts	**Efficiency:** Reduces unnecessary network traffic and improves reliability
Scalability	Limited due to high computational cost on the protected server	Authentication and access control are decoupled, allowing distributed deployment and horizontal scaling	**Performance:** Scales efficiently across large enterprise deployments
Integration	Typically embedded within SDP implementations, although standalone libraries are available (SDP)	Designed with integration in mind; protocol-agnostic and works with SDP, DNS security, FIDO authentication, OAuth, and other security frameworks	**Versatility:** This can be integrated into diverse security architectures

Deployment Complexity	Requires static pre-defined access points, which may introduce challenges in cloud environments	Supports dynamic access control and adaptive authentication for cloud-native and hybrid environments	**Cloud-Readiness:** Better suited for dynamic and scalable enterprise infrastructures
Extensibility	Limited message types	Designed as a general-purpose “authenticate-before-connect” protocol with an extensible messaging framework	**Future-Proof:** Can be applied to various Zero Trust security models beyond SDP

Securing the Session Layer: Proactive Mitigation of AI-Driven Zero-Day Attacks

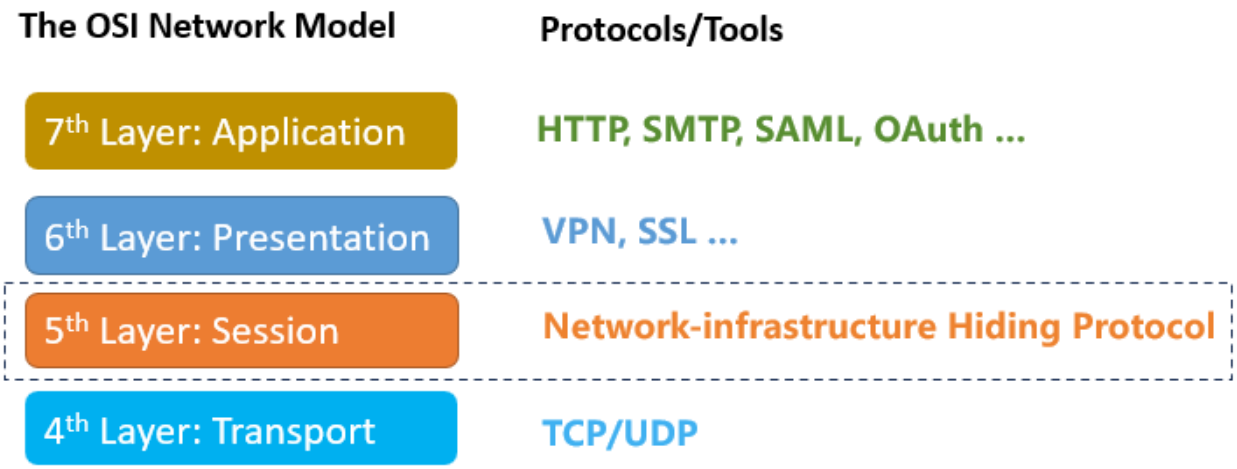

Figure 1. Position in OSI model

Modern Zero Trust solutions predominantly focus on securing the OSI model's 6th (presentation) and 7th (application) layers, such as VPN replacements for encrypted communication or web proxies for application-layer filtering. While these approaches mitigate specific risks, they fail to address the foundational vulnerability inherent in modern networks: the inherent trust granted to any connection attempt at the 5th layer (session). This oversight leaves organizations vulnerable to weaknesses in application-level protocols, as evidenced by high-profile breaches such as the MOVEit Transfer attack (2023) and the Log4j vulnerability (2021) where attackers exploited unauthenticated session-layer access to bypass traditional

defenses. By enforcing Zero Trust principles at Layer 5—authenticating and authorizing entities before a session is established—NHP eliminates the "trust-by-default" flaw in TCP/IP protocols.

- **Mitigate Vulnerability Risk:** The openness of TCP/IP protocols leads to a "trust by default" connection model, allowing anyone to establish a connection to a server port that provides services. Attackers exploit this openness to target server vulnerabilities. The NHP protocol implements the zero trust principle "never trust, always verify" by enforcing "deny-all" rules by default on the server side, only allowing authorized hosts to establish connections. This effectively mitigates vulnerability exploitation, particularly zero-day exploits
- **Mitigate Phishing Attacks:** DNS hijacking is a serious threat to internet security and is used for malicious purposes such as phishing, stealing sensitive information, or spreading malware. The NHP can function as an encrypted DNS resolution service to mitigate this problem. When the NHP-Agent on the client side sends a knock request to the controller component NHP-Server with the identifier (e.g., domain name) of the protected resource, the NHP-Server will return the IP address and port number of the protected resource if the NHP-Agent is successfully authenticated. Since NHP communication is encrypted and mutually verified, the risk of DNS hijacking is effectively mitigated
- **Mitigate DDoS Attacks:** A client cannot obtain the IP address and port number of protected resources without authentication. If the protected resources are distributed across multiple locations, the NHP-Server may return different IP addresses to different clients, making DDoS attacks significantly more difficult and expensive. Additionally, if a DDoS attack is launched using the same NHP-Agent identity, the NHP-Server can easily mitigate it by throttling requests from that agent; this forces attackers to use unique identities, which is not scalable and significantly increases the complexity of executing the attack
- **Reduces Lateral Movement:** Binds session identity to user and device, reducing lateral movement risk in Zero Trust by limiting unauthorized access within the network after initial authentication
- **Attack Attribution:** The TCP/IP protocol suite is IP-based. With NHP, however, the model becomes identity-based (ID-based). The connection initiator's identity must be authenticated before establishing the connection, making attacks much more auditable
    - **Example:** Oracle E-Business Suite Zero-Day Exploitation (2025). Attackers exploited CVE-2025-61882 as a zero-day starting in August 2025, targeting exposed Oracle EBS instances (e.g., at Harvard University) via reconnaissance and unauthenticated remote access to deploy ransomware and steal data. NHP would have hidden these endpoints entirely, enforcing identity-based authentication before any visibility or connection, preventing initial discovery, blocking exploitation, and enabling attribution through audited identity logs rather than anonymous IPs

NHP Architecture and Workflow

The NIST Zero Trust Architecture standard[5] inspires the NHP architecture. Its goal is to ensure that protected resources remain invisible to unauthorized entities and only accessible to authorized subjects through continuous verification. The architecture leverages modern cryptographic algorithms and frameworks to ensure security and trustworthiness against sophisticated attacks. Unlike SPA, the NHP architecture decouples the authentication and access control features, improving performance and scalability. The NHP architecture is an extensible framework that allows seamless integration with SDP for SPA replacement and interoperability with other protocols (e.g., DNS, FIDO, OAuth). The diagram below illustrates the core components of the NHP architecture, as well as other interacting components.

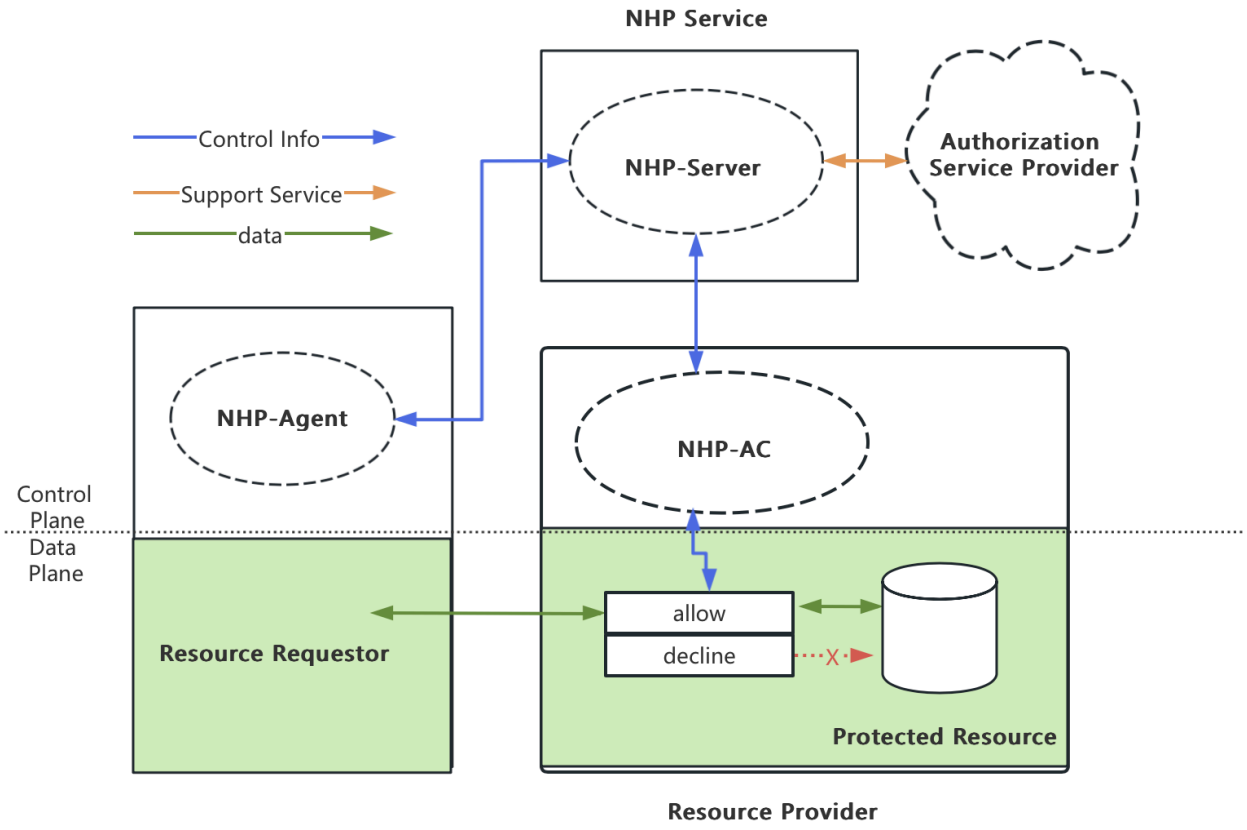

Figure 2. NHP architecture diagram

NHP Core Components

- **NHP-Agent:** A client-side component that initiates communication and requests access to protected resources by sending knock requests to the NHP-Server component. It can exist as a client, SDK, browser, application, or other similar entity
- **NHP-Server:** A server-side component that processes and validates knock requests, performs authentication and security policy validation, and manages the NHP-AC components to dynamically grant access to authorized NHP-Agents
- **NHP-Access Controller (NHP-AC):** This module enforces access policies and ensures the network invisibility of protected resources, such as servers or applications. It enforces the default deny-all security policy and ensures the protected resource's

network invisibility, potentially residing on the same host as the protected resource (referred to as “gateway” in SDP contexts). It is responsible for granting access to authorized NHP-Agents, denying access to unauthorized NHP-Agents, and performing actions based on commands returned by the NHP-Server

NHP’s core components align with the NIST Zero Trust Architecture (SP 800-207) model: the NHP-Server functions as the Policy Engine/Administrator, managing authentication and policy decisions, while the NHP-AC serves as the Policy Enforcement Point, enforcing access controls in real time. This mapping illustrates how NHP operationalizes NIST’s logical framework to enhance Zero Trust security.

Components That Interact with NHP

- **Protected Resources:** The resource provider is responsible for protecting these resources, including API interfaces, application servers, gateways, routers, and network devices. In the SDP scenario, the Protected Resources are the SDP Gateway and Controller
- **Resource Requestor:** The entity (e.g., user, application, device, server) seeking access to protected resources. It hosts the NHP-Agent to initiate secure, authenticated requests, embodying the “authenticate-before-connect” principle by sending knock requests only after local policy checks. This ensures requests align with Zero Trust by verifying context (e.g., device posture) before engaging the NHP-Server
- **Authorization Service Provider (ASP):** This provider validates access policies and provides the actual access addresses of Protected Resources. In the SDP scenario, the ASP may be the SDP Controller
- **Log Server:** An optional external system for centralized log collection, storage, and analysis. The NHP-Server uploads operational, security, and traffic logs (as detailed in the Logging section) to enable auditing, threat detection, and compliance integration with SIEM systems

NHP Workflow

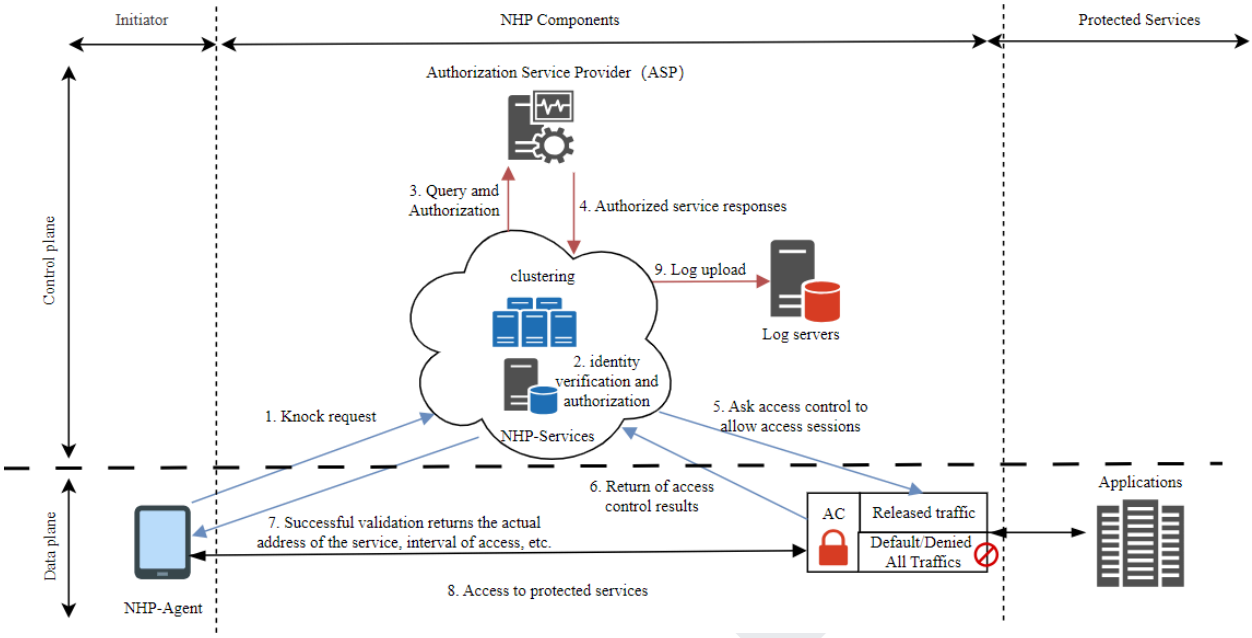

Figure 3. NHP workflow

1. Before the host (e.g., application, device, server) with the NHP-Agent accesses the protected resource, a knock request (NHP-KNK) is sent from the NHP-Agent to the NHP-Server.
2. The NHP-Server processes the request and retrieves the NHP-Agent information, which includes the agent's identity, device information, the authorized service provider (ASP) requesting access, and the target resource information.
3. The NHP-Server authenticates the NHP-Agent's identity using its public key, aggregates the agent information, locates the corresponding ASP server (typically the identity and access management (IAM) system) based on the ASP information, and sends a query.
4. The ASP server performs authorization based on agent information, authorizes the target resources that the NHP-Agent can access, and replies with the target resource's actual IP address and port, along with other authorization information (e.g., token). In Network Address Translation (NAT) environments (e.g., behind proxies or load balancers in cloud deployments), the returned address/port may point to an intermediary (e.g., SDP Gateway) configured for source IP preservation (via PROXY protocol or X-Forwarded-For headers) to maintain auditability and compatibility with identity-based policies, as supported by modern proxies like NGINX or HAProxy.
5. If the NHP-Server query in step 3 is successful, the NHP-Server will send an open-door request (NHP-AOP) to the target resource's corresponding NHP-AC.
6. Upon receiving the NHP-Server's request, the NHP-AC needs to verify whether the requested resource matches the protected resource. It then opens the connection channel from the NHP-Agent to the protected resource and replies (NHP-ART) with a validity period of the open-door session to the NHP-Server.

7. Once the NHP-Server confirms the success of the open-door session, it responds (NHP-ACK) to the NHP-Agent with the actual address of the protected resource and the duration of the open-door session.
8. From this point, the host with NHP-Agent can access the requested resource until the validity period has expired. Once the period expires, the NHP-Agent must negotiate a renewal of the open-door session by repeating steps 1–7.
9. (Optional) The NHP-Server collects all the logs generated during the above steps and uploads them to log servers for auditing and data analysis.

Cryptographic Algorithms and Frameworks

Cryptographic algorithms are fundamental to the NHP protocol, ensuring security and trustworthiness. NHP leverages modern cryptographic algorithms to boost security and performance, which include:

- **Elliptic Curve Cryptography (ECC):** ECC is a public-key cryptographic method recognized for its strong security and efficient performance, and it has been integrated into protocols such as TLS 1.2 [10] and Bitcoin [11]. This algorithm has been widely demonstrated to deliver both speed and robust security in public network environments
- **Noise Protocol Framework:** The Noise Protocol Framework [6] is a key exchange protocol that combines random ECC key pairs with the ECDH algorithm and has proven to be extraordinarily secure and fast in popular applications such as WhatsApp and WireGuard. In the NHP specification, the XX handshake pattern is generally recommended, since it supports decentralized key management, provides full forward secrecy, and offers stronger identity protection during the handshake. In implementations such as OpenNHP, different handshake patterns can be selected depending on operational needs. The OpenNHP reference implementation adopts the one-way K handshake pattern, utilizing the steps “e,” “es,” and “ss” for efficient key establishment. Static keys are distributed out-of-band and coordinated via sender IDs in messages, improving scalability and speed in resource-constrained environments. This choice avoids the multiple interactive rounds required by the XX pattern, though it comes at the cost of reduced forward secrecy and weaker identity hiding. Nonetheless, it still ensures mutual authentication and encryption, consistent with NHP’s default-deny security posture
- **Identity-Based Cryptography (IBC):** IBC represents a new paradigm in the field of cryptographic key distribution, offering significantly better scalability and lower costs

Compared to traditional digital signature schemes, NHP enhances performance by leveraging ECC, which provides the same level of security as RSA but with significantly smaller key sizes and far greater efficiency. ECC with a 256-bit key length is chosen in NHP because it offers an excellent balance between security strength and computational efficiency. For example, while RSA2048 requires a 2048-bit public key, NHP achieves equivalent classical security (approximately 128 bits) with only a 256-bit key, reducing both computational overhead and ciphertext size. However, both ECC and RSA are vulnerable to quantum attacks (e.g., Shor’s algorithm). To future-proof against this, NHP’s modular Noise Protocol Framework supports

migration to post-quantum key encapsulation mechanisms like ML-KEM (Kyber, NIST FIPS 203), potentially via hybrid modes combining classical and PQC algorithms during transition, ensuring compatibility in constrained environments like IoT while addressing emerging quantum threats. This choice mirrors Bitcoin's adoption of ECC 256 for the same reasons, a decision that has proven successful and secure in practice.

Moreover, NHP leverages the lightweight Noise Protocol Framework for efficient and secure mutual identity verification. This key negotiation mechanism integrates ephemeral key pairs with the ECDH algorithm, enabling the implicit authentication of both interacting parties while establishing a symmetric key for message encryption and decryption. The Noise framework's introduction of random key pairs at each session is particularly suitable for secure applications like NHP, which involves encryption and decryption key rotation for every session and minimizes susceptibility to brute-force attacks or replay attacks, aligning perfectly with the Zero Trust principles. Additionally, the modular nature of the Noise framework allows for flexible integration with various cryptographic algorithms and hash functions, supporting future protocol upgrades.

NHP supports both Public Key Infrastructure (PKI) and Identity-Based Cryptography (IBC) for key management and distribution. PKI key distribution systems have been well-known for decades and have been widely used. However, the increasing number of users and devices in Zero Trust security architecture scenarios makes the key distribution process for each user device and subsequent management processes increasingly complex and costly. In contrast, IBC can address this issue by simplifying the key distribution and verification process. The IBC algorithm uses a common public key matrix, allowing easily readable character combinations to serve as unique identifiers in place of public keys for transmission and computation. This significantly reduces the complexity of the key center's public key management and distribution processes. This significantly enhances security while reducing the overhead associated with key exchanges, making NHP a highly scalable and efficient Zero Trust solution for heavy workloads.

NHP Message Header

The NHP protocol header contains the data required for identity authentication, supporting mutual verification and data interaction between the communicating parties.

The NHP protocol data packet is transmitted using the UDP protocol as the standard process, but the TCP protocol (short connection) can be used when necessary. The NHP protocol header has a fixed length of 160 bytes (standard length, corresponding to international cryptographic algorithms) or 224 bytes (extended length, corresponding to domestic cryptographic algorithms). The message ciphertext immediately follows the header. To prevent plaintext exposure, leading obfuscation utilizes a 4-byte random number, along with the message type and length, for XOR operations. The message uses GCM AEAD encryption. To better support the TCP protocol, the size of the ciphertext message (plaintext size + 16-byte tag) must be included in the message length field of the header. The verification and processing of the header leverage the Noise Protocol Framework: during initiation, a fresh ephemeral key pair is generated and mixed; parsing decrypts via derived keys and verifies authenticity; confirmation/response follow the

pattern (e.g., XX for mutual auth). Concurrent sessions (e.g., overlapping knocks for different applications from one agent) are distinguished by fresh ephemerals per handshake and payload specifics (e.g., target resource ID), with independent states preventing interference.

Table 3. NHP header fields

Field	Size (Bytes)	Description
Leading Obfuscation	4	Random number generated by the sender for each message. Its primary purpose is to obscure the Message Type and Message Length fields by performing an XOR operation on them, preventing passive observers from easily identifying the message structure or type in transit (to avoid fingerprinting or traffic analysis attacks). Upon receipt, the receiver extracts this random value first and uses it to deobfuscate (reverse XOR) the subsequent type and length fields before further processing. This aligns with NHP's stealth mode by adding a layer of plausible deniability to packet headers. For example, if the random is 0xAABBCCDD, it would be XORed bitwise with the concatenated type and length bytes.
Message Type	2	Identifies the NHP message type (e.g., 0x01 for NHP-KNK knock request, 0x02 for NHP-ACK acknowledge). Obfuscated via XOR with part of the Leading Obfuscation random. Deobfuscated during parsing using the random value. This field ensures routing to the correct handler (e.g., authentication vs. keepalive) while maintaining deniability.
Message Length	2	Indicates the length of the encrypted message ciphertext that follows the header, including the plaintext payload size plus the 16-byte GCM authentication tag (in big-endian format). It is obfuscated via XOR with the Leading Obfuscation random to hide packet sizes from observers. The receiver deobfuscates it to know how much data to read and decrypt after the fixed header. The maximum supported length is 65,535 bytes (UDP limit), ensuring compatibility with UDP transport. For TCP, this field supports short connections by including the full ciphertext size for reliable parsing.

Protocol Major Version	1	Denotes the major version of the NHP protocol (e.g., 0x01 for version 1.x). It is sent in plaintext within the header and used by the receiver to ensure compatibility before processing. A mismatch triggers silent discard (default-deny), preventing version downgrade attacks. This supports backward-incompatible changes in future major releases while maintaining Zero Trust principles.
Protocol Minor Version	1	Specifies the minor version of the NHP protocol (e.g., 0x00 for 1.0). Sent in plaintext, it allows for backward-compatible enhancements (e.g., new flags, optional fields). The receiver checks this against supported versions; minor mismatches may allow graceful degradation, but critical features (e.g., specific Noise patterns) could lead to rejection. This enables iterative improvements without breaking deployments.
Protocol Flags	2	Contains bit flags for controlling protocol behavior, stored in big-endian format. Bits may indicate options like compression (e.g., bit 0: zlib enabled on payload), handshake pattern (e.g., bits 1-3: 0 for XX, 1 for K), encryption mode (e.g., bit 4: hybrid PQC), or transport hint (e.g., bit 5: TCP fallback). Reserved bits are set to 0. The receiver interprets these to adjust processing (e.g., decompress after decryption). This extensibility supports scalability and future-proofing, such as quantum-resistant modes, without expanding the header size.
Reserved	4	Reserved for future use and must be set to all zeros (0x00000000) by the sender. It provides padding and expansion space for protocol evolution (e.g., adding new identifiers or short extensions in minor versions). The receiver ignores it but may validate it as zero to detect tampering or non-compliant implementations. This ensures forward compatibility in enterprise deployments.
Counter	8	Holds a 64-bit counter (big-endian) acting as a unique transaction tracker and nonce for cryptographic operations, derived from the Noise Protocol Framework (e.g., the “n” nonce in CipherState). It starts at 0 for a new session and post-increments with each encryption/decryption operation, providing replay protection and sequencing. If exhausted (i.e., reaches 2^64-1) or mismatched, the

		session errors out. For concurrent knocks from the same NHP-Agent (e.g., to different applications), independent per-session counters and states prevent overlap. The receiver verifies it against expected values to mitigate replay attacks.
Ephemeral Public Key	32 or 64	Contains the sender's fresh ephemeral public key for the Noise handshake/ECDH key derivation (32 bytes for Curve25519/X25519 in standard/international mode; 64 bytes for SM2 (X+Y coordinates) in extended/domestic mode). Generated per message/transaction to introduce randomness, forward secrecy, and cryptographic strength, even for non-initial types (e.g., keepalives, logs)—treating each as a lightweight handshake. Not optional; zeroed or reused if irrelevant, but typically fresh to enhance security.
Local Public Key Ciphertext	48 or 80	Encrypted ciphertext of the sender's local static public key (ID-based or PKI-based), protected via AEAD (e.g., ChaCha20-Poly1305, AES-GCM) using keys derived from the ephemeral DH. Size includes the plaintext key size (e.g., 32 bytes) plus 16-byte tag, totaling 48 bytes standard (or 64+16=80 extended). Purpose: Secure transmission of the static key during handshake (e.g., "s" token in Noise XX: encrypted after initial "e"). The receiver decrypts it post-DH to verify identity, eliminating MITM risks from SPA. Fails silently on decryption error.
Timestamp Ciphertext	24	Contains the encrypted 8-byte UNIX timestamp (milliseconds since epoch, big-endian) plus 16-byte AEAD tag, using session-derived keys. It provides freshness to counter replay attacks (receiver checks against current time with a tolerance (e.g., ±5 minutes) to account for clock skew). Encrypted to prevent timing analysis. The sender sets it to current time; receiver decrypts and validates to ensure the message isn't stale. Fixed size across modes for consistency.
HMAC	32	HMAC (e.g., HMAC-SHA256) computed over the entire header (excluding this field) and payload ciphertext, using a key derived from the Noise chaining key (ck). It ensures integrity and authenticity of the message, detecting tampering. The sender computes it last; receiver verifies it before decryption. If invalid, the packet is silently dropped. This adds an extra layer beyond AEAD

		tags, especially for non-encrypted parts like ephemerals, aligning with NHP's defense-in-depth.

Here is an example of packet processing flow in a one-way transaction:

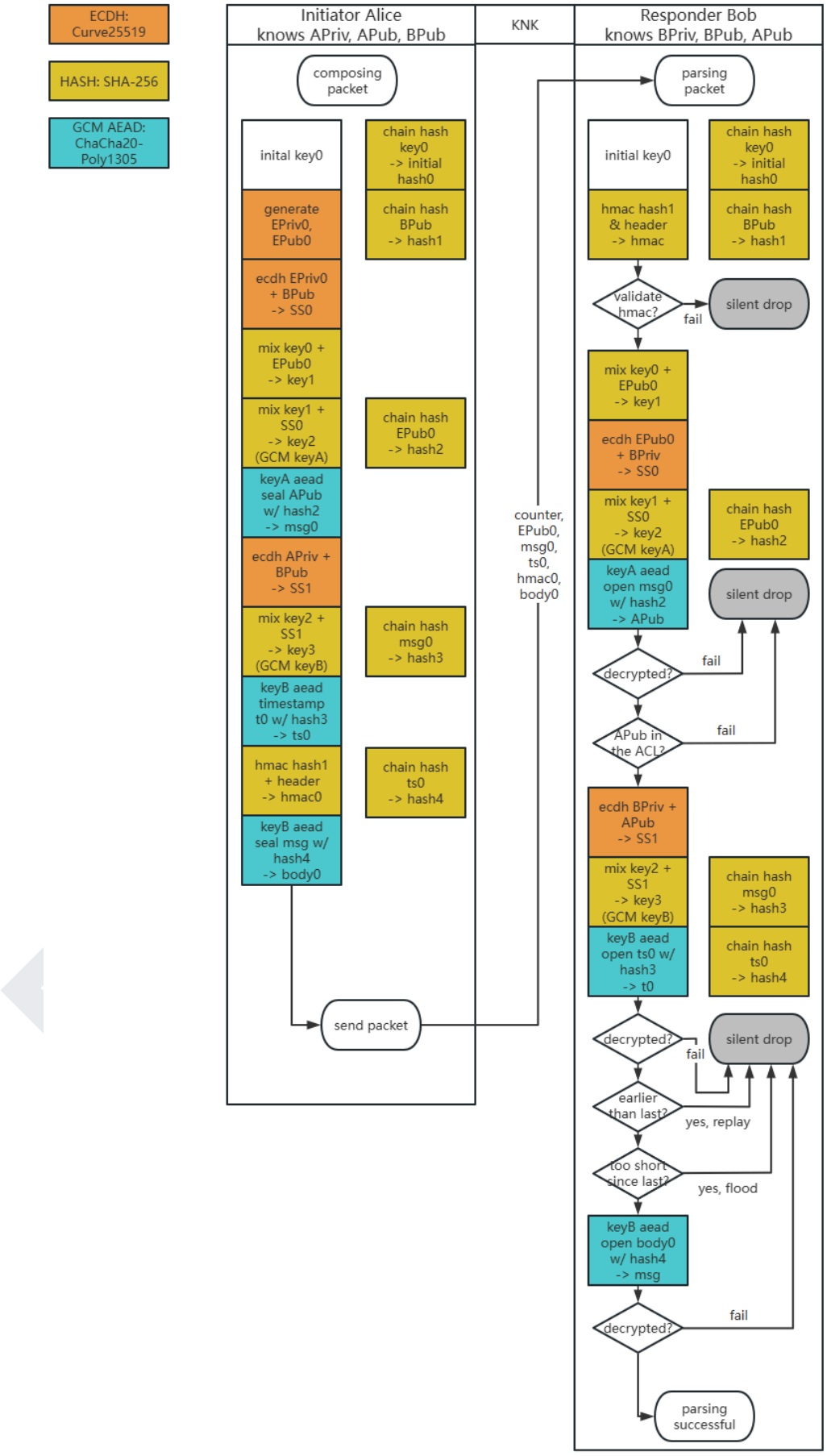

Figure 4. Example NHP packet processing flow

NHP Message Types

The NHP protocol message functions as the payload of the NHP header, collectively forming the NHP protocol packet. The total length of the header and payload must remain within the UDP packet size limit of 65,535 bytes. Each NHP protocol message is identified by its type, as specified in the protocol header, with distinct message types handled by separate logical components. Unless otherwise stated, the message body is formatted in JSON, compressed using zlib, and then encrypted and decrypted utilizing the key derived during the creation or parsing of the header.

The table below provides a detailed classification of NHP message types, along with their corresponding descriptions.

Table 4. NHP message types

Name	Type	Description
NHP-KPL	0	Protocol keep-alive mechanism.
NHP-KNK	1	Used by the NHP-Agent to initiate a server access request.
NHP-ACK	2	Used by the NHP-Server to respond to an access request from the NHP-Agent.
NHP-AOP	3	Used by the NHP-Server to instruct whether to allow or block access.
NHP-ART	4	Used by the NHP-AC to respond to an open-door request.
NHP-LST	5	Used by the NHP-Server to query the status of the NHP-Agent and validate access requests.
NHP-LRT	6	The NHP-Server or NHP-Agent is used to return the results of a query request.
NHP-COK	7	Used by the NHP-Server to return a cookie to the NHP-Agent.
NHP-RKN	8	Used by the NHP relay server to forward authentication requests.
NHP-RLY	9	Used in the NHP relay server to forward responses between devices.
NHP-AOL	10	Used by the NHP-AC to connect to and notify the NHP-Server of online status changes.
NHP-AAK	11	Used by the NHP-Server to respond to a connection request from the NHP-AC.

NHP-OTP	12	Used by the NHP-Agent to request and verify a one-time password (OTP).
NHP-REG	13	Used by the NHP-Agent to register for the authentication system.
NHP-RAK	14	Used by the NHP-Agent to revoke authentication keys.
NHP-ACC	15	Used by the NHP-AC to allow or deny access.
NHP-LOG	16	Used by the NHP-Agent to submit server access logs.
NHP-LAK	17	Used by the NHP-Server to respond to log submission results.
NHP-ARD	18	Used by the NHP-Server to instruct an AC to reconnect to a different NHP Server for registration or service continuity

Appendix 2 provides a detailed description of each message type. Please refer to it for specific explanations and message structures.

Logging

Logging and auditing are integral to the NHP protocol, providing transparency, continuous security monitoring, and forensic traceability. Both the NHP-Server and NHP-AC generate and maintain distinct types of logs that capture critical security events, system operations, and network interactions. This dual-perspective logging supports compliance with security policies, enables threat detection across the whole trust boundary, and ensures a robust audit trail for incident response and analysis.

Log Types

NHP defines three primary categories of logs:

- **Operational Logs:** Record events related to the operation and maintenance of NHP components, including system startup, shutdown, and inter-component communications
- **Security Logs:** Capture authentication attempts, access control decisions, and security-related events such as unauthorized access attempts and policy violations
- **Traffic Logs:** Provide insights into network activity, recording source and destination addresses, protocol types, and data flow volumes

Log Format

Each log entry includes the following fields:

Table 5. Log info fields

Field	Description
Timestamp	The date and time of the event
Log Level	The event's severity (info, debug, warning, error, critical)
Event Name	A descriptive identifier for the event
Source IP	The originating IP address of the packet
Destination IP	The IP address the packet is targeting, which may represent the resource/service endpoint the NHP is attempting to access or an intermediate server (e.g., proxy, NHP-AC), depending on the logging layer's position in the architecture
Session ID	A unique identifier for the session
Message Details	Additional contextual information about the event

Adhere to data minimization principles by logging only essential information required for security and operations. Explicitly avoid logging sensitive data such as passwords, biometric information (e.g., fingerprints from FIDO integration), full cryptographic keys, or personal health data.

Log Transmission and Storage

Logs are securely transmitted to a central logging server for analysis and storage. Encryption ensures the confidentiality and integrity of log data. Where feasible, hash or pseudonymize identifiers (e.g., user IDs, device IDs, IP addresses) via techniques like SHA-256 hashing before transport to reduce re-identification risks while preserving auditability. Logs are retained in accordance with compliance regulations and industry best practices.

Compliance and Auditing

- NHP logs align with security frameworks such as ISO 27001 and NIST 800-92, as well as privacy regulations like GDPR and CCPA through data minimization, anonymization, and consent management where personal data is involved
- Logs can be integrated with Security Information and Event Management (SIEM) systems for real-time monitoring and threat detection
- Periodic audits ensure adherence to security policies and facilitate forensic investigations
- NHP logs are primarily generated server-side (by NHP-Server and NHP-AC), facilitating secure storage, encryption, backup, and anonymization while maintaining compliance with standards like ISO 27001. Additionally, implementations can support Write-Once-Read-Many (WORM) storage for immutable logs to ensure tamper-proof auditing

IoT Device Logging

The integration of Internet of Things (IoT) devices into the NHP framework presents both unique challenges and opportunities for logging practices. As IoT devices become increasingly prevalent in modern network architectures, it's crucial to adapt logging strategies to accommodate their specific characteristics while maintaining the robust security posture demanded by NHP.

IoT devices, ranging from simple sensors to complex industrial control systems, often operate under significant constraints that impact their logging capabilities. These constraints include: limited storage capacity, restricted processing power, battery life considerations for wireless devices. Furthermore, the sheer diversity of IoT devices, with their varied operating systems, firmware, and security capabilities, adds layers of complexity to implementing a standardized logging approach within NHP.

Severely resource-constrained devices (for example, those with 8-bit microcontrollers, <32KB RAM, and low bus speeds) pose additional challenges for NHP deployments, as cryptographic operations (e.g., ECDH key exchanges in Noise), I/O for log generation, and upstream/downstream traffic can strain compute and power budgets, potentially leading to latency, battery drain, or instability. To mitigate, NHP prioritizes lightweight primitives: ECC-256 requires ~10-20x less computation than RSA-2048 equivalents, enabling viability on mid-range IoT hardware (e.g., ARM Cortex-M0+ with 64KB+ RAM).

Minimum and ideal requirements for NHP on IoT devices:

- **Minimum (Basic Functionality):** Devices with ≥32KB RAM, ≥100MHz CPU (e.g., ESP8266 or equivalent), and basic crypto acceleration (software ECC). Supports core NHP features like port hiding and simplified authentication via one-way Noise patterns (e.g., K), with minimal logging (e.g., prioritized security events only, no ML analysis). Suitable for sensors in low-threat environments; expect ~5-10% overhead on traffic/crypto
- **Ideal (Full Functionality):** Devices with ≥128KB RAM, ≥200MHz CPU (e.g., ESP32, ARM Cortex-M4), hardware crypto (e.g., AES accelerator), and Trusted Execution Environment (TEE) support. Enables advanced features like mutual authentication (XX pattern), comprehensive logging with edge aggregation/summarization, quantum-resistant hybrids (e.g., Kyber integration), and ML-based anomaly detection. Provides enhanced resilience (e.g., immutable logs via hash chains) with <5% overhead, ideal for critical OT/IoT in high-threat scenarios

Consider the following **tiered deployment guidance**. If devices meet minimum specs, achieve basic stealth and authentication for essential Zero Trust (e.g., hiding endpoints in smart grids). With ideal hardware, gain full defense-in-depth, including runtime behavioral analytics and SIEM integration, maximizing NHP's scalability in large-scale IoT networks.

The scale and volume of IoT deployments present another significant challenge. With potentially thousands or even millions of devices in a single network, each generating logs at high

frequency, there's a risk of overwhelming central log management systems. This is compounded by the often intermittent or low-bandwidth network connections characteristic of many IoT deployments, necessitating careful consideration of log transmission protocols and timing.

To address these challenges, several IoT-specific logging strategies can be implemented within the NHP framework:

- **Edge Logging:** Initial log aggregation and processing occur on local devices or edge computing nodes, helping manage data volume and reduce strain on central systems. To ensure the security of aggregated logs at the edge, apply cryptographic signatures using NHP's ID-based cryptography to verify integrity, signing each batch before transmission for an auditable chain of custody. Additionally, utilize TEE on the edge device to encrypt logs in a secure enclave, protecting confidentiality against unauthorized access and aligning with NHP's default-deny approach. For constrained IoT devices, implement key lifecycle management: periodic key rollover (e.g., automated rotation every 24-48 hours via Noise's ephemeral keys and out-of-band static updates) to limit exposure windows and revocation mechanisms (e.g., via IBC's central Private Key Generator or PKI CRLs) to invalidate compromised keys, ensuring rapid response to threats while maintaining minimal overhead. These measures enhance reliability, ensuring compliance and auditing in constrained IoT deployments
- **Log Summarization and Compression:** Techniques to reduce storage and transmission requirements, crucial for resource-constrained devices
- **Prioritized Logging:** Implementing a system that ensures critical security events are always captured and transmitted, even when resources are limited

In IoT environments, ML-based detection for log analysis is valuable but introduces challenges, such as false positives (incorrectly flagging legitimate activities) and false negatives (missing malicious events) due to limited training data or noisy conditions, impacting NHP's real-time enforcement and auditing accuracy. Mitigation involves integrating NHP's cryptographic logging with adaptive ML models, updating datasets regularly, and cross-validating anomalies with NHP-AC feedback to minimize errors while upholding Zero Trust principles, ensuring robust security in constrained IoT deployments.

Security measures for IoT logging within NHP must be robust yet lightweight. Key security considerations include:

- **Encryption:** Both at rest and in transit, using algorithms suitable for the limited computational power of IoT devices
- **Log Integrity Protection:** Through mechanisms such as hash chains or digital signatures
- **Blockchain-inspired Technologies:** For distributed log integrity in some implementations
- **Secure Boot Mechanisms:** To ensure the integrity of the logging software itself
- **Hardware-based Trust Anchors:** Where available, to enhance security

Compliance and privacy considerations add another layer of complexity to IoT logging in NHP. With regulations like GDPR and CCPA in force, it's crucial to implement:

- **Data Minimization Techniques:** Logging only necessary information
- **Anonymization:** Applying where appropriate to protect personal data
- **Consent Management Mechanisms:** To handle the collection of personal data, ensuring transparency and user control

The analysis of IoT log data within NHP presents both challenges and opportunities. Key aspects include:

- **Distributed Analysis Techniques:** Leveraging edge computing resources to manage data volume and reduce central processing load
- **Machine Learning Algorithms:** Employed for anomaly detection to identify potential security threats
- **Correlation of IoT Device Logs:** With broader network events to maintain a comprehensive security posture within the NHP framework

These strategies and considerations form the foundation of a practical IoT logging approach within the NHP framework, striking a balance between the need for comprehensive security monitoring and the practical limitations of IoT environments.

Integration with SDP

The NHP is an enhanced replacement for the SPA component in the SDP architecture. Network hiding was initially introduced in SDP 1.0 using SPA to protect infrastructure components. By replacing SPA with the NHP, organizations can significantly boost the security of their SDP deployments.

By incorporating the NHP, organizations can reduce their attack surface by rendering SDP components, such as Controllers and Gateways, invisible to unauthorized users. This integration ensures that even the SDP infrastructure remains concealed from potential attackers, enhancing security against network-based threats.

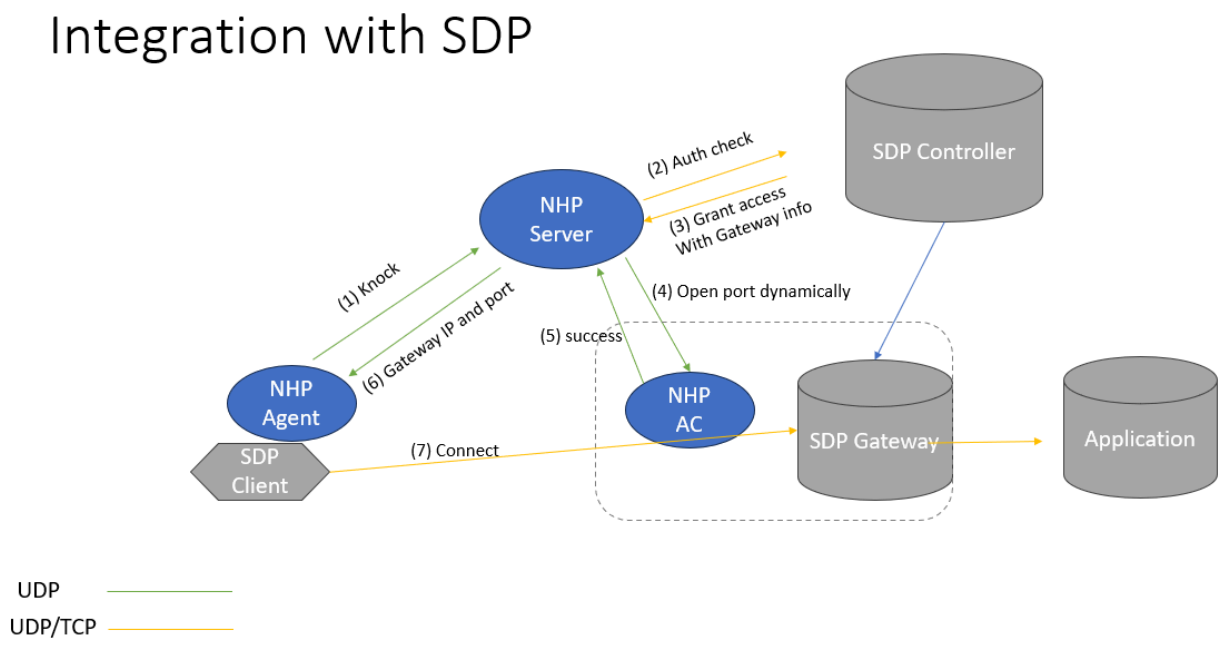

Figure 5. Integration with SDP

Integration Process

1. **Knock Request Initiation:** The NHP-Agent, integrated with the SDP Client on the Initiating Host (IH), prepares a secure "knock" request. This request includes the IH's identity information, device posture, and requested resource identifiers. Modern cryptographic methods, such as the Noise Protocol Framework, encrypt the request to ensure confidentiality and integrity. The knock request is sent to the NHP-Server over a secure channel. This initial communication does not expose any network infrastructure details, maintaining the invisibility of SDP components.
2. **Authentication Check:** Upon receiving the knock request, the NHP-Server decrypts and validates the message using Identity-Based Cryptography (IBC). The NHP-Server verifies the IH's identity and device posture against its records. It checks for compliance with security policies, such as device health, location, and time constraints. The NHP-Server collaborates with the SDP Controller (acting as the Authorization Service Provider (ASP)) to validate access policies. The SDP Controller determines if the IH can access the requested resources based on predefined policies.
3. **Access Grant:** The SDP Controller grants access if authentication and authorization are successful. The decision includes specifics about which resources and services the IH is allowed to access. The SDP Controller provides the NHP-Server with the necessary SDP Gateway information, such as IP addresses and port numbers.
4. **Port Opening Instructions:** The NHP-Server sends a secure instruction to the NHP-AC module on the SDP Gateway. This instruction includes details about the IH, such as its identity and the specific ports or services to be opened. The NHP-AC dynamically updates firewall rules or access control lists to allow the IH's IP address to connect to the

specified ports. This dynamic process occurs in real-time to minimize the open window during which ports are open.

5. **Confirmation:** The NHP-AC confirms to the NHP-Server that the necessary ports have been opened successfully. It may provide additional information, such as the validity period for the opened ports or any session-specific identifiers. The confirmation ensures that the IH can proceed to the next step, knowing that access has been provisioned.
6. **Gateway Information Relay:** The NHP-Server sends the SDP Gateway's IP address, port information, and any necessary session parameters back to the NHP-Agent on the IH. This message is securely transmitted, maintaining confidentiality and integrity. The NHP-Agent updates the SDP Client with the received information, preparing it to establish a connection with the SDP Gateway.
7. **SDP Connection Establishment:** The SDP Client on the IH initiates a connection with the SDP Gateway using the provided IP address and port. Mutual TLS (mTLS) authentication ensures that both the IH and SDP Gateway authenticate. Upon a successful mTLS handshake, the standard SDP workflow proceeds. The IH gains access to the authorized resources through a secure, encrypted tunnel. Throughout the session, access remains governed by policies defined by the SDP Controller. Any policy changes or detected anomalies can result in session termination.

By integrating the NHP with SDP, organizations enhance the invisibility of their network infrastructure, making it significantly more difficult for unauthorized users to discover or target SDP components.

Caveats and Risk Considerations

While the integration process enhances Zero Trust security, deployments should account for potential edge cases to ensure resilience:

- **NHP-Server Compromise:** If the NHP-Server is compromised (e.g., through insider threats, unpatched vulnerabilities), an attacker could potentially access SDP Gateway details or manipulate authentication decisions. To mitigate, deploy the NHP-Server in a hardened, isolated environment with multifactor authentication for administrative access. Leverage NHP's comprehensive logging for real-time monitoring and anomaly detection via SIEM integration. Regular key rotation using IBC and failover to redundant NHP-Servers can further limit impact, ensuring that core SDP components remain hidden by default
- **Explicit Feedback Mechanism:** Unlike SPA's complete silence (which maximizes plausible deniability but leads to unreliable connections in variable networks), NHP provides confirmation responses (e.g., NHP-ACK with resource details) upon successful authentication to improve client reliability and reduce retry overhead. However, this enhances usability at the potential cost of minor information leakage—attackers could infer endpoint existence from timed responses or patterns, though fully encrypted payloads and silent discards for failures minimize this risk. Mitigate by configuring short session windows and monitoring for probing attempts via integrated logging
- **Key Management Complexity:** IBC simplifies distribution compared to PKI but introduces centralization risks (e.g., key escrow), potentially complicating usability in

distributed teams. Balance by using hybrid modes (IBC for scalability, PKI for high-assurance scenarios) and automated tools for rollover/revocation, ensuring security without excessive administrative burden

- **High Latency Impact on Port-Opening Windows:** In high-latency networks (e.g., satellite, global, mobile deployments with variable cellular/Wi-Fi signals), delays in knock requests or confirmations could cause port-opening windows to expire before the IH connects, leading to failed sessions or increased retry overhead. This may degrade user experience in bandwidth-constrained scenarios, such as roaming mobile devices, without proper mitigation. NHP's explicit feedback mechanism (unlike SPA) allows retries, but adjustable timeouts can be configured in the NHP-AC (e.g., extending the real-time window in step 4 of the integration process based on observed round-trip time (RTT)). Monitor latency via performance logging and integrate with adaptive networking tools (e.g., QUIC for faster recovery) to dynamically adjust session parameters, maintaining usability without compromising the minimize-exposure principle
- **Performance in Constrained Environments:** Lightweight cryptography (e.g., ECC/Noise) aids deployment on IoT/low-end devices, but enabling advanced features like PQC hybrids or ML-based anomaly detection may increase latency or power draw, trading usability for future-proof security. Assess per use-case and prioritize basics for resource-limited setups, reserving full capabilities for critical infrastructure
- **Scope of Zero-Day Protection:** While NHP effectively minimizes exposure to network-layer zero-day vulnerabilities by concealing assets, it does not prevent exploitation of application-level zero-days once authenticated access is granted. Complement NHP with application-specific security controls, such as runtime protection and regular patching, for layered defense
- **Server Unavailability:** The unavailability of an NHP-Server poses a risk to service continuity. Mitigation options include implementing a high-availability design with redundant servers or cached authentication tokens and considering fallback mechanisms such as graceful degradation to a local SDP Controller to address failure scenarios in enterprise deployments

Benefits of Integrating NHP with SDP

- **Enhanced Security**
  - **Infrastructure Hiding:** The NHP employs advanced cryptographic techniques to conceal SDP components from unauthorized users, significantly reducing the risk of network reconnaissance
  - **Authenticate-Before-Connect:** Adds an extra layer of authentication before any connection is established
- **Improved Resilience to Attacks**
  - **DDoS Mitigation:** The NHP reduces the potential vectors for DDoS attacks against SDP components by ensuring that only authenticated clients can initiate communication
  - **Zero-Day Protection:** Hiding SDP components minimizes exposure to potential zero-day vulnerabilities
- **Scalability and Performance**

    - **Decoupled Architecture:** The NHP separates authentication and access control components, allowing for scalable deployment without significantly impacting SDP performance
    - **Efficient Key Management:** Modern cryptographic methods, such as IBC, simplify key distribution and management, enhancing security and performance. NHP's ECC-256 with the Noise Framework achieves equivalent classical security to SPA's RSA-2048 approaches but with significantly reduced overhead: key sizes of 256 bits (versus 2048 bits), 50-100x faster key exchanges/operations, and latencies as low as ~30ms (versus higher for RSA), boosting throughput in enterprise SDP deployments while enabling efficient handling of concurrent clients
- **Compliance and Monitoring**
    - **Enhanced Logging:** The NHP provides detailed logs that can be integrated with existing SIEM systems for comprehensive monitoring
    - **Simplified Compliance Reporting:** Improved audit trails facilitate adherence to security standards and regulations

Implementation Considerations

When integrating NHP into your SDP deployment, consider the following:

- **Component Placement:** Deploy the NHP-Server in a secure, centralized location to efficiently handle authentication requests before they reach SDP components
- **Synchronization:** Ensure the NHP's authentication processes are seamlessly integrated with SDP workflows to maintain a cohesive security posture
- **Key Management:** Automated key management solutions should synchronize cryptographic keys and certificates between the NHP and SDP components
- **Logging and Monitoring:** Integrate NHP logs with existing SDP and SIEM systems for comprehensive monitoring and incident response
- **Performance Impact:** Monitor the effect of NHP on SDP performance and adjust configurations as necessary to optimize performance
- **Decision and Planning Guidance:** Recognize that NHP integration introduces additional operational complexity. Conduct a balanced assessment of security benefits versus maintenance effort. Early architectural planning and phased rollout help manage adoption risk
- **Cost and Resource Planning:** Include total cost of ownership—covering infrastructure, automation, and training—in planning. Offset costs through reduced exposure, simplified audits, and decreased reliance on traditional perimeter defenses to ensure long-term sustainability
- **Resilient Design:** Implement redundant NHP-AC instances with automatic failover mechanisms (e.g., load balancers, cluster orchestration) to maintain access continuity and prevent single points of failure from misconfigurations or component failures, aligning with Zero Trust resilience principles

Applicability Across SDP Deployment Models

The NHP can enhance security in various SDP deployment models:

- **Client-to-Gateway Model:** The NHP ensures the SDP Gateway remains hidden until authentication is successful, adding an extra layer of security before the standard SDP connection is established
- **Client-to-Server Model:** Adds an extra layer of protection by authenticating access to individual servers
- **Server-to-Server Model:** Secures server-to-server communications by ensuring only authenticated servers can connect
- **Complex Topologies:** In models such as Client-to-Server and Client-to-Client, the NHP can authenticate multiple points along the communication chain. For instance, the NHP can authenticate multiple points in peer-to-peer applications or client-mediated interactions, ensuring secure communication channels throughout the network

Integration with DNS

NHP can be designed to resolve domain names securely by integrating DNS resolution with an authenticate-before-connect workflow, providing a substantial security improvement over traditional DNS. By combining identity verification, access policy enforcement, and encrypted session initiation into a single process, NHP keeps the network infrastructure fully concealed until the client is authenticated. Traditional DNS, in contrast, exposes domain metadata to the public, leaving systems vulnerable to hijacking, poisoning, and amplification attacks. As shown in the diagram, the NHP-Agent sends a request to the NHP-Server containing a private domain identifier (e.g., api.xyz.com), which serves as a reference token rather than a publicly resolvable name. If authentication succeeds, the NHP-Server queries a private DNS, accessible only by the NHP-Server itself, to retrieve the IP address and port of the protected service. Once resolved, the NHP-Server instructs the NHP-AC to open the port dynamically and returns the access information to the NHP-Agent. This architecture eliminates domain exposure, ties DNS resolution to verified identity, and upholds Zero Trust access control at the network layer.

It should be noted that NHP is not designed to replace DNSSEC but to offer an alternative mechanism for services that prefer not to be registered with a public domain name resolver. For example, an enterprise's internal application (e.g., internalapp.company.com) may remain unregistered in the public DNS while still being securely accessible within the organization.

DNS over NHP (DoN)

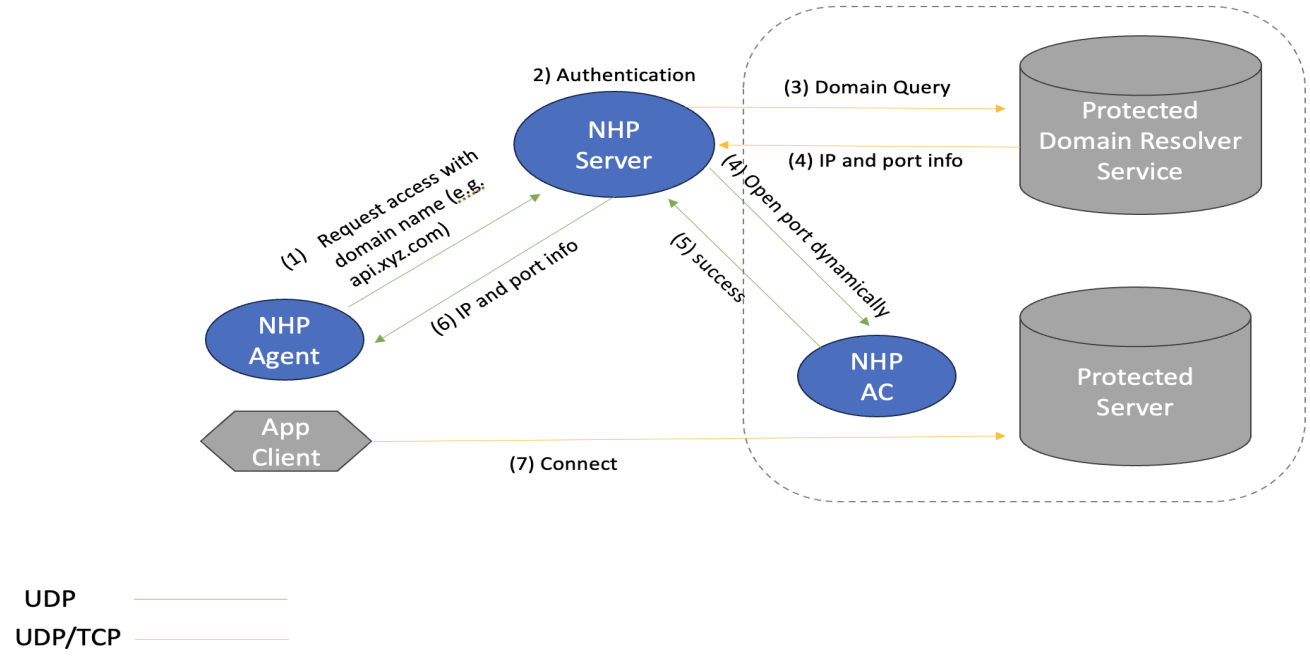

Figure 6. DNS over NHP

Integration Process

1. Before the host (e.g., application, device, server) with an NHP-Agent accesses the protected resource, a knock request is sent from the NHP-Agent to the NHP-Server containing the requested domain name (e.g., api.sdp.com).
2. The NHP-Server authenticates the NHP-Agent and validates the associated identity and policy before proceeding with any DNS resolution or connection setup.
3. Upon successful authentication, the NHP-Server issues a domain query to the Protected Domain Resolver Service, requesting the necessary connection details for the authorized domain.
4. The Protected Domain Resolver Service returns the corresponding IP address and port information to the NHP-Server, which securely relays this data to the NHP-AC.
5. The NHP-AC dynamically opens the required port for the specific session and confirms success to the NHP-Server.
6. The NHP-Server securely transmits the authorized IP address and port information to the NHP-Agent.
7. The NHP-Agent initiates a secure connection to the protected server using the authenticated, policy-approved, and time-limited session parameters.

Key benefits of integration with DNS:

- **Built-In Zero Trust Enforcement:** Enforces "never trust, always verify" before resolving or exposing any infrastructure. Session-level access is policy-controlled, ephemeral, and identity-driven
- **Elimination of DNS Exposure:** By making domain names not publicly resolvable, the approach prevents enumeration, scanning, and reconnaissance by external actors. OS-level resolution will not reveal protected records; unresolved queries will time out unless routed via NHP (e.g., through the authenticated NHP-Agent workflow)
- **Zero-Trust-based DNS Resolution:** DNS queries are only executed after successful authentication. It ties DNS resolution strictly to user or device identity and policy
- **Defense Against DNS-based Attacks:** The approach shields against DNS hijacking, cache poisoning, and amplification attacks
- **DNS Resolution Isolation:** The Protected Domain Resolver Service is only accessible by the NHP-Server and not discoverable or reachable on the public network
- **Port Access Control:** Ports are dynamically opened only after successful authentication and authorization, which reduces the attack surface by keeping ports closed by default
- **Seamless Integration with Encryption:** DNS resolution, access control, and session setup happen within the encrypted protocol flow, which prevents information leakage during the handshake
- **Resistance to DDoS and Scanning:** Since neither domain nor port is exposed before authentication, mass scanning and DDoS attacks are effectively neutralized
- **Fine-Grained Access Policy:** Each DNS query and port access is evaluated on a per-session and per-identity basis

Integration with FIDO

Fast Identity Online (FIDO) enables passwordless and phishing-resistant authentication through cryptographic credentials securely stored on user devices, such as biometric sensors or hardware security keys. NHP, as a cryptography-driven protocol built on an authenticate-before-connect model, verifies user identity before exposing any network or service details. This design aligns naturally with FIDO's architecture, making NHP well-suited for integrating FIDO authentication to strengthen identity assurance and enforce Zero Trust principles at the network access layer. Integrating FIDO into NHP results in a robust, user-friendly, and future-proof identity layer for secure, invisible-by-default network access.

Benefits of NHP and FIDO integration:

- **Passwordless Authentication:** Eliminates reliance on passwords, thereby reducing attack surfaces related to phishing, credential stuffing, and brute-force attacks
- **Phishing Resistance:** FIDO authentication is based on public-key cryptography and scoped to a specific domain, preventing credential reuse or interception
- **Strong Identity Assurance:** Verifies user identity through secure methods, such as biometrics or hardware tokens, ensuring that only authorized individuals can access protected services
- **Support for Modern Devices:** Compatible with a wide range of devices like smartphones, laptops with biometric sensors, and hardware tokens like YubiKeys

- **User Convenience with Passkeys:** Enables seamless, password-free access across devices through passkey synchronization (e.g., iCloud Keychain, Google Password Manager)
- **Lightweight Integration:** FIDO (through WebAuthn) can be integrated into existing NHP client-server exchanges with minimal overhead and no dependency on external Identity Providers (IdPs)
- **Hardware-Backed Security:** Credentials are generated and stored in secure elements (e.g., TPMs, Secure Enclaves, hardware keys), providing strong resistance against device compromise
- **Auditable, Policy-Driven Access:** FIDO authentication can be tied to user roles, device trust posture, and access context, enabling fine-grained Zero Trust policies

While FIDO integration enhances security by providing robust, phishing-resistant authentication, deployment must address practical challenges. These include ensuring device compatibility across diverse hardware platforms and establishing recovery mechanisms for lost hardware tokens. NHP's cryptographic flexibility (for example, IBC and the Noise Protocol Framework, as detailed earlier) can support adaptive solutions, such as secondary authentication methods or token re-provisioning, to maintain access continuity while upholding Zero Trust principles.

Recovery Models and Fallback Authentication

While FIDO provides strong, phishing-resistant authentication, deployments must incorporate robust recovery strategies to handle lost or compromised devices without compromising security.

Recommended approaches include:

- **Multi-Device Registration:** Encourage users to register multiple FIDO authenticators (e.g., a primary hardware key like YubiKey and a secondary biometric on a smartphone) during onboarding. NHP can leverage FIDO's multifactor capabilities to allow seamless switching, with the NHP-Server validating against registered credentials
- **Recovery Codes or Secondary Methods:** Generate one-time recovery codes (stored securely offline) or integrate fallback options like time-based OTP (TOTP) via authenticator apps, email/SMS (as a last resort with additional verification), or knowledge-based challenges. These should be policy-enforced through the ASP, triggering only after anomaly detection (e.g., unusual location) and logging for auditing
- **Administrative Re-Provisioning:** For enterprise scenarios, admins can re-provision credentials via out-of-band processes (e.g., in-person verification, integration with IAM systems like Okta), revoking lost keys via IBC/PKI revocation lists. Automate this with NHP's extensibility to minimize downtime while upholding "never trust" principles

Fallbacks must be temporary and risk-assessed (e.g., reduce privileges during recovery sessions) to prevent weakening the overall Zero Trust posture. Test these models in pilots to balance usability and security.

Conclusion and Future Outlook

Conclusion

Cyber threats necessitate a proactive and adaptive security approach. The Network-infrastructure Hiding Protocol (NHP) represents the next evolution of Zero Trust security, ensuring that organizations remain protected against autonomous vulnerability exploitation. NHP establishes a robust, scalable, and stealth-driven security framework inspired by SPA but enhanced for modern cyber threats. Importantly, NHP is not a standalone solution; it complements but does not replace other Zero Trust controls, such as IAM, continuous monitoring, and runtime protection, to form a comprehensive defense-in-depth strategy. By adopting NHP, enterprises can severely reduce their attack surface, prevent unauthorized access, and build a more resilient security posture in the high-risk AI era.

Future Outlook

As cyber threats evolve, NHP must adapt to ensure long-term resilience and security. Future development efforts will focus on integrating post-quantum cryptography (PQC) to safeguard against emerging threats posed by quantum computing. By leveraging PQC algorithms, NHP will maintain robust encryption and authentication mechanisms even in a post-quantum era, ensuring that network infrastructure remains protected against quantum-based attacks. Backward compatibility will be preserved through hybrid cryptographic modes (e.g., combining classical ECC with PQC like ML-KEM during transitions), version negotiation in the protocol header, and extensible structures, allowing gradual upgrades without breaking existing implementations. Additionally, NHP will be optimized for seamless integration with leading cloud platforms such as AWS, Azure, and Google Cloud to facilitate widespread adoption. This will include native support for cloud-native security controls (e.g., AWS VPC and Security Groups for dynamic access, Azure Network Security Groups and Azure AD for identity federation, Google Cloud Armor and Identity Platform for threat protection), key management services (e.g., AWS KMS, Azure Key Vault, Google Cloud KMS), and automated deployment mechanisms (e.g., APIs like AWS SDK, Azure Resource Manager, Google Cloud Deployment Manager), enabling enterprises to incorporate NHP into their existing cloud security architectures easily. Multi-Cloud Platforms (MCPs), such as unified management tools (e.g., Wiz, Mirantis), can play a key role by orchestrating NHP deployments across providers, ensuring consistent policy enforcement, centralized logging, and resilient multi-cloud Zero Trust implementations. By enhancing cryptographic agility through hybrid modes (classical + PQC) and improving cloud interoperability, NHP will continue to serve as a foundational Zero Trust security framework for modern, distributed IT environments. Future efforts may also explore deeper integration with AI-based defensive tools, such as automated threat response systems and ML-driven anomaly detection in logging/behavioral analytics, to proactively counter AI-powered adversaries.

Examples of Potential Seamless Integrations

To realize NHP's vision of pervasive Zero Trust enforcement, future developments could include seamless integrations that automate deployment, management, and policy enforcement across hybrid and cloud environments. Specific examples might encompass:

- **Integration with Cloud Identity and Access Management (IAM) Services:** Native hooks into IAM platforms from major cloud providers (e.g., AWS, Azure, Google Cloud) to enable centralized identity verification. This would allow NHP to leverage existing user directories for mutual authentication, ensuring that access requests are continuously validated against dynamic policies without requiring custom middleware
- **Automated Control of Cloud Network Security Constructs:** NHP-ACs could dynamically manage cloud-native security features, such as security groups, network ACLs, or virtual private cloud (VPC) configurations. For instance, a successful NHP knock could trigger just-in-time updates to these controls, granting temporary access to hidden resources while adhering to least-privilege principles
- **Deployment via Infrastructure as Code (IaC) Tools:** Support for IaC frameworks like Terraform, Ansible, or Pulumi to provision NHP components automatically. This would facilitate rapid deployment in containerized environments (e.g., Kubernetes clusters), where NHP could obscure pod-to-pod communications until authentication is verified, enhancing microsegmentation in cloud-native architectures
- **Integration with Centralized Policy Engines and Zero Trust Frameworks:** NHP could integrate with policy-as-code frameworks such as Open Policy Agent (OPA), Kubernetes Gatekeeper, or Zero Trust orchestration platforms like Zscaler, Illumio, or Elisity. This integration would unify access-control logic across infrastructure layers, enabling centralized, declarative policy enforcement and ensuring consistent application of least-privilege and context-aware access rules throughout distributed environments.
- **Orchestration with Security Operations Tools:** Extensions to integrate with SIEM systems or orchestration platforms (e.g., for automated threat response), where NHP logs (as described in the Logging section) feed into real-time analytics, enabling proactive adjustments to hiding policies based on emerging threats
- **Integration with Enterprise Key and Certificate Management Systems:** NHP could interface with existing KMS or HSM solutions, such as AWS KMS, Azure Key Vault, or HashiCorp Vault, to automate certificate issuance, renewal, and revocation. This integration would enable consistent key rotation, cryptoperiod enforcement, and lifecycle management across hybrid environments, maintaining strong cryptographic hygiene

These integrations would build on NHP's existing compatibility with SDP and FIDO, promoting scalable, automated Zero Trust implementations while maintaining cryptographic robustness.

Appendix 1: The OpenNHP Open-Source Project

NHP has been implemented as an open-source project named OpenNHP. At the time of publication of this whitepaper, the state of the OpenNHP project is version 0.6.0, released as an open-source reference implementation. The maturity of the project is assessed as “early stage / reference implementation” — functional for demonstrations and initial integrations, but still undergoing active development toward feature completeness, performance optimization, and production-ready stability.

The repository is accessible at https://github.com/OpenNHP/opennhp. This project aims to provide a reference implementation of NHP, allowing the community to:

- Understand the practical aspects of NHP implementation
- Contribute to the development and improvement of the protocol
- Test NHP in various network environments
- Integrate NHP into the existing security infrastructure
- Audit the NHP implementation

Appendix 2: The Details of NHP Message Types

NHP-KPL (Keepalive) Message

The keepalive message follows a standardized NHP protocol header with a fixed length of 12 bytes and all other fields set to zero. The message type is **0**, and the message body is empty. Upon receiving a keepalive message, the recipient performs no additional processing beyond acknowledging receipt. This message maintains network connections between the NHP-Server and the NHP-Agent, or between the NHP-Server and the NHP-AC. Intermediate NHP-Servers do not forward keepalive messages.

Table A2.1. NHP-KPL Message Fields

Field	Description
Message Type	Set to 0, indicating a keepalive message
Message Size	Always set to 0, with an empty message body

NHP-KNK (Knock) Message

This message is initiated by the NHP-Agent and received by the NHP-Server. It must use the NHP protocol header with the message type set to **1**. The message contains the necessary information for verification, including the user's details, device details, terminal operating environment parameters, and the target resource information that needs to be accessed (such as a custom domain name, ID, or other string). The user, device, and other information must interface with an external Authorized Service Provider (ASP).

Table A2.2. NHP-KNK Message Fields

Field	Description
User ID	A unique identifier is assigned to each user to distinguish different report-sending users
Device ID	A unique identifier is assigned to each device to distinguish different report-sending devices
Authorized Service Provider ID	A unique identifier is assigned to an external third-party authorization service provider (ASP), and this field is used to interact with a designated third-party service
Resource ID	A unique identifier is assigned to the protected resource being accessed, and this field specifies the target resource that requires access

Source IP Address	The source IP address of the current NHP-Agent (If required, the NHP-Agent must obtain the exit IP address, which the NHP-Server can optionally use)
Source Port	The source port number of the current NHP-Agent (If required, the NHP-Agent must obtain the source port of the exit, which the NHP-Server can optionally use)
Terminal Environment Parameters	The NHP-Agent returns terminal environment verification parameter validation results, formatted as ["checkID0": result, "checkID1": result, ...], and the validation parameters checkID must correspond to the verification mechanism provided by the NHP-Agent and third-party authorized service providers

NHP-ACK (Acknowledge) Message

This message is initiated by the NHP-Server and received by the NHP-Agent. It must use the NHP protocol header with the message type set to **2**. After authorization is completed, the NHP-Server responds with an NHP-ACK message that includes the authorization success code or the reason for failure. This message is linked with the previous NHP-KNK message to form a session. If the verification is successful, the message includes the actual service address of the protected resource. It also carries a session ID (8-byte uint64 type) for tracking, access authorization duration, and other related information. The NHP-Agent can use the information from the NHP-ACK message to establish a protected access connection.

Table A2.3. NHP-ACK Message Fields

Field	**Description**
User ID	Identifies the user initiating the request
Device ID	Identifies the device initiating the request
Service Provider ID	Identifies the responding service provider
Resource ID	Identifies the target resource being accessed
Session ID	A unique identifier is assigned to each session request
Error Code	Indicates failure reasons; 0 for successful authentication
Resource Address	The IP address or domain name of the target resource
Access Duration	The duration (in seconds) for which the session remains valid
Temporary Access Endpoint (Optional)	The temporary access endpoint provided for this session is combined with the NHP-ACC message

Temporary Access Key (Optional)	The temporary access key assigned for this session is combined with the NHP-ACC message

NHP-AOP (AC Operations) Message

This message is sent by the NHP-Server and received by the NHP-AC. It must use the NHP protocol header with a message type of **3**. The NHP-Server sends the NHP-AOP message to notify the NHP-AC to allow or block access. The message contains the NHP-Agent's source IP address, port number, and the protected resource's destination IP address and port.

NAT Considerations: In NAT environments, multiple NHP-Agents behind the same router may share a common public source IP address, making it difficult for the NHP-AC to distinguish individual endpoints and potentially allowing unintended group access. To mitigate this, implement an application-layer token or cookie exchanged between the authenticated NHP-Agent and the protected resource server for fine-grained verification. Additionally, adopting IPv6 can alleviate the issue by providing unique source addresses for each endpoint, enhancing precision in access control.

Table A2.4. NHP-AOP Message Fields

Field	Description
Session ID	A unique identifier for the session request matching the value in the NHP-Agent's response
Device ID	A unique identifier for the NHP-Agent's device
Public Key	The public key of the NHP-Agent
Source IP Address	The source IP address of the NHP-Agent
Source Port	The source port number of the NHP-Agent (optional)
Destination IP Address	The IP address of the protected resource
Destination Port	The port number of the protected resource
Access Duration	The duration for which access is granted (in seconds); 0 means access is denied

NHP-ART (AC Result) Message

This message is initiated by the NHP-AC and received by the NHP-Server. It must use the NHP protocol header with the message type set to **4**. After access authorization, the NHP-AC sends the NHP-ART message to respond to the NHP-Server. The NHP-ART message, together with

the NHP-AOP message, forms a complete session. The message includes the session ID, granted access duration, and other related information. The NHP-AC generates an authorization token using a unique random value and the agent's device ID. The access token can be used for additional authentication on the client side. If the token value is unnecessary, it can be returned as an empty response to the NHP-Agent. If the NHP-AC denies access, the granted duration is set to **0**.

Table A2.5. NHP-ART Message Fields

Field	Description
Session ID	Identifier for the corresponding session request
Access Duration	The actual duration for which access is granted (in seconds); 0 means denied
Temporary Access Endpoint (Optional)	The temporary access endpoint granted for this session is combined with the NHP-ACC message
Temporary Access Key (Optional)	The temporary access key assigned for this session is combined with the NHP-ACC message

NHP-LST (List) Message

This message is initiated by the NHP-Agent and received by the NHP-Server. The agent uses it to request application information from the server. The message must use the NHP protocol header, with the message type set to **5**. It carries the agent's user ID, device ID, and request ID to verify and match against previously known authorized service provider IDs and resource IDs.

Table A2.6. NHP-LST Message Fields

Field	Description
User ID	A unique identifier is assigned to each user and used to distinguish different report-sending users
Device ID	A unique identifier is assigned to each device to distinguish different report-sending devices

Request ID	A unique identifier for this request
Request Port Information	Indicates whether port information is required
Requested Application Information	Specifies the application information to be retrieved. A unique name (either UUID or customized string) to represent an application/service that the server can manage (to open access at its AC collections)

NHP-LRT (List Result) Message

This message is initiated by the NHP-Server and received by the NHP-Agent. It serves as the response from the NHP-Server to the NHP-Agent's list discovery request. The message must use the NHP protocol header, with the message type set to **6**. The message carries the source IP address and source port of the NHP-Agent's request, along with the NHP-Server's authorized service provider IDs and resource IDs.

Table A2.7. NHP-LRT Message Fields

Field	**Description**
List Request ID	The identifier corresponds to the received list request
Name	The name of the server
Type	Typically NHP-Server. Other types foreseen for future use.
Service and Application Info	Information about the protected resources managed by the server, including provider IDs and resource IDs

NHP-COK (Cookie) Message

This message is initiated by the NHP-Server and received by the NHP-Agent. When the NHP-Server works under high load, it uses this message to send a cookie to the NHP-Agent and validates the incoming message's HMAC field with the cookie. It plays as an early drop mechanism for the NHP-Server processing messages and allows the NHP-Server to still manage to respond to NHP-Agents that use the same cookie. The message must use the NHP protocol header, with the message type set to **7**. The message includes a 32-byte cookie value generated based on the session.

Table A2.8. NHP-COK Message Fields

Field	Description
Cookie	The NHP-Server generates the cookie value

NHP-RKN (Re-Knock) Message

This message is initiated by the NHP proxy and received by the NHP-Server. It serves as a second knock using a cookie. The NHP protocol header must be used, and the message type is **8**. The message carries authentication key information, which has the same fields as NHP-KNK; however, the HMAC calculation must also use the cookie value obtained from NHP-COK.

NHP-RLY (Relay) Message

This message is initiated by the NHP relay server and received by the NHP-Server. The NHP relay server uses it to forward messages that require the NHP proxy source address to be retained to the NHP-Server. Except in this case, other message types can generally be forwarded transparently. The NHP protocol header must be used, and the message type is **9**. The carried message is the original NHP protocol request packet from the NHP proxy, without additional encryption or compression. The protocol flag field carries a compression flag of 0.

NHP-AOL (AC Online) Message

This message is initiated by the NHP-AC and received by the NHP-Server. It is used by the NHP-AC to report its online status to the NHP-Server, allowing the server to maintain persistent communication. The message must use the NHP protocol header, with the message type set to **10**. The message carries the NHP-AC ID and information about the protected resources it manages, including service provider IDs, resource IDs, and traffic flow data.

Table A2.9. NHP-AOL Message Fields

Field	Description
AC ID	Identifier for the AC
Service and Application Info	Details about the protected resources the AC manages, including service provider IDs, resource IDs, and inbound/outbound traffic

NHP-AAK (AC Acknowledge) Message

This message is initiated by the NHP-Server and received by the NHP-AC. It confirms the connection status between the AC and the server. The message must use the NHP protocol

header, with the message type set to **11**. It carries the public exit IP address and port of the NHP-AC.

Table A2.10. NHP-AAK Message Fields

Field	Description
NHP-AC Address	The public IP address and port of the AC

NHP-OTP (One-time Password) Message

The NHP protocol provides an optional method for requesting a one-time password (OTP). This message is initiated by the NHP-Agent, NHP-Server, or Key Generation Center (KGC) and is used to pre-authenticate the NHP-Agent before registering with the server. The message must use the NHP protocol header, with the message type set to **12**. The message carries information about the user, device, and the NHP-Agent's account. Upon verification, the ASP triggers the issuance of an OTP, and the server does not perform any further processing on the NHP-OTP message.

Table A2.11. NHP-OTP Message Fields

Field	Description
User ID	A unique identifier is assigned to each user and used to distinguish different report-sending users
Device ID	A unique identifier is assigned to each device and is used to distinguish different report-sending devices

NHP-REG (Register) Message

This message is initiated by the NHP-Agent and received by the NHP-Server. The NHP-Agent uses it to register its public key with the NHP-Server. The message must use the NHP protocol header, with the message type set to **13**. The message carries the NHP-Agent's user ID, device ID, and a one-time authentication credential (which may be an SMS code, email token, or QR code). The NHP-Server verifies the enclosed public key and stores it. Upon successful registration, the NHP-Server responds with an NHP-RAK message to confirm registration completion.

Table A2.12. NHP-REG Message Fields

Field	Description
User ID	A unique identifier is assigned to each user to distinguish different report-sending users
Device ID	A unique identifier is assigned to each device to distinguish different report-sending devices
One-time Password (OTP)	The NHP-Agent obtains a one-time authentication code, which the NHP-Server validates, and the agent uses this code to prove its identity to the NHP-Server

NHP-RAK (Register Acknowledge) Message

This message is initiated by the NHP-Server or the Key Generation Center (KGC) and received by the NHP-Agent. It confirms that the public key registration has been completed. The message must use the NHP protocol header, with the message type set to **14**, and it does not contain a message body.

NHP-ACC (Access) Message

This message is initiated by the NHP-Agent and received by the NHP-AC. The NHP-Agent uses it to respond to the AC's temporary listening port. The message must use the NHP protocol header, with the message type set to **15**. It carries the NHP-Agent's user ID, device ID, and access token information. The NHP-AC verifies the access token before allowing traffic to the target service.

Table A2.13. NHP-ACC Message Fields

Field	Description
User ID	A unique identifier is assigned to each user to distinguish different report-sending users
Device ID	A unique identifier is assigned to each device to distinguish different report-sending devices
Temporary Access Token	The token is required to access the AC's temporary listening port

NHP-LOG (Log) Message

This message is initiated by the NHP-AC and received by the NHP-Server. It is used by the NHP-AC to report log content to the NHP-Server. The NHP protocol header must be used, and the message type is **16**. The message carries the AC ID, log ID, and log content.

Table A2.14. NHP-LOG Message Fields

Field	Description
AC ID	The unique identifier of each AC
Log ID	A unique identifier for each log report, usually a hash of the log content
Log Content	The AC generates the log content

NHP-LAK (Log Acknowledge) Message

This message is initiated by the NHP-Server and received by the NHP-AC. The NHP-Server uses it to confirm to the NHP-AC that the reported log content has been received. The NHP protocol header must be used, with a message type of **17**. The message carries the received log ID.

Table A2.15. NHP-LAK Message Fields

Field	Description
Log ID	The log ID received by the NHP-Server

NHP-ARD (AC Redispatch) Message

The NHP_ARD message is sent by an NHP Server to an AC to explicitly redirect the AC to another NHP Server endpoint. The message is used during NHP_AOL to enable load-balanced, policy-driven AC reassignment without exposing network topology.

Table A2.16. NHP-ARD Message Fields

Field	Description
Address	The newly assigned NHP-Server address (list).

Useful References

[1] Software-Defined Perimeter (SDP) Specification v2.0. Jason Garbis, Juanita Koilpillai, Junaid Islam, Bob Flores, Daniel Bailey, Benfeng Chen, Eitan Bremler, Michael Roza, and Ahmed Refaey Hussein. Cloud Security Alliance (CSA). Mar 2022

[2] From Naptime to Big Sleep: Using Large Language Models To Catch Vulnerabilities In Real-World Code, Google Big Sleep team. Google Project Zero. Oct 2024

[3] LLM Agents can Autonomously Exploit One-day Vulnerabilities. Richard Fang, Rohan Bindu, Akul Gupta, Daniel Kang. arxiv. April 2024

[4] AI Systems Can Generate Working Exploits for Published CVEs in 10-15 Minutes. Cybersecuritynews.com, Aug 2025

[5] NIST Special Publication 800-207, "Zero Trust Architecture," National Institute of Standards and Technology (NIST). Aug 2020

[6] Noise Protocol Framework

[7] The Washington Post, Net of insecurity, A flaw in the design, Craig Timberg. May 30, 2015

[8] OpenAI's o3 AI Found a Zero-Day Vulnerability in the Linux Kernel. May 24, 2025

[9] CrowdStrike 2025 GlobalThreat Report. Feb 27, 2025

[10] RFC8422. IETF. Aug 2018

[11] BitCoin Core Open Source Project

[12] Gartner Says That in the Age of GenAI, Preemptive Capabilities, Not Detection and Response, Are the Future of Cybersecurity, Gartner, Sept 2025