Demonstrating Encrypted Client Hello (ECH) Privacy Benefits

Jasper Christian Stritzke, Tim Betzer*, Christian Dietze*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: jasper.stritzke@tum.de, betzer@net.in.tum.de, diec@net.in.tum.de

***Abstract*—Encrypted Client Hello (ECH) extends TLS 1.3 by encrypting the entire ClientHello, eliminating plaintext Server Name Indication (SNI) leakage. A reproducible container-based testbench, built from CoreDNS, an ECH-enabled Nginx instance, and an experimental curl client, allows direct, side-by-side observation of ECH-protected versus conventional TLS handshakes with the same endpoint. Packet captures produced by the framework show that ECH conceals the actual destination hostname, exposing only an innocuous outer SNI to passive observers. In contrast, plaintext TLS exposes the real domain in every trace. The work demonstrates that ECH eliminates the plaintext metadata leak in TLS 1.3 without altering the overall handshake semantics, compatibility, or application-layer behavior.**

***Index Terms*—TLS, Encrypted Client Hello, ECH, SNI, Privacy, Network Security, HPKE**

1. Introduction

1.1. Evolution of Internet Privacy and TLS Challenges

The increasing demand for internet privacy has driven significant advancements in cryptographic protocols, particularly within the Transport Layer Security (TLS) suite.

While TLS 1.3 encrypts most handshake data, a critical privacy vulnerability persists: the Server Name Indication (SNI) extension and other sensitive fields remain plaintext. The SNI, transmitted during the initial `ClientHello` message, reveals the target domain to network observers, enabling pervasive monitoring, censorship, and traffic analysis [1], [2], compromising user privacy despite encrypted communication.

Following post-Snowden surveillance concerns, the Internet Engineering Task Force (IETF) formally identified pervasive monitoring as an attack in RFC 7258 [3], transforming privacy from a technical feature into a foundational protocol requirement.

ECH represents a critical infrastructure upgrade for internet privacy, directly countering surveillance capabilities that undermine encrypted communication confidentiality.

1.2. ECH as a Comprehensive Privacy Solution

Encrypted Client Hello represents a significant advancement in addressing TLS privacy vulnerabilities through a comprehensive, protocol-level approach. Rather than addressing individual metadata leaks, ECH fundamentally restructures the TLS handshake to encrypt all sensitive client information from initial connection.

This enhancement eliminates piecemeal privacy solutions by providing complete ClientHello metadata protection, creating a unified defense against passive surveillance and traffic analysis across TLS 1.3 implementations. ECH prioritizes immediate privacy benefits and long-term protocol sustainability, ensuring infrastructure compatibility while establishing foundations for future privacy enhancements.

1.3. Purpose and Significance of the ECH Testbench Project

The ECH Testbench project provides a local, reproducible framework demonstrating ECH functionality and privacy benefits. By comparing ECH-enabled with non-ECH TLS traffic to the same server, the testbench shows how ECH obscures real target domains, challenging passive network observation. This setup provides an interactive playground for exploring ECH's operational flow and privacy implications.

2. Background on TLS and Encrypted Client Hello

This section provides essential background on TLS handshake mechanics, the privacy vulnerabilities in current implementations, and the evolution toward ECH as a comprehensive solution.

2.1. Fundamentals of TLS Handshake and SNI

TLS handshakes establish secure channels where client and server negotiate cryptographic parameters and authenticate identities. The Server Name Indication (SNI) extension allows clients to specify target hostnames in ClientHello messages. When multiple domains share IP addresses, servers use this field to present correct certificates [2].

Despite subsequent application data encryption, the ClientHello with SNI extension remains unencrypted in TLS 1.3, exposing target domains to on-path observers including ISPs, government agencies, and malicious actors [1]. This plaintext exposure undermines user privacy by revealing browsing habits and enabling censorship or traffic analysis [4]. Intermediaries can inspect the first TLS packet to determine specific targets, despite full communication encryption.

 doi: 10.2313/NET-2025-11-3_20

2.2. Evolution from ESNI to ECH

The privacy risks of plaintext SNI motivated the proposal of `Encrypted SNI (ESNI)`, an early attempt to hide the SNI field [2]. However, ESNI proved insufficient by protecting only SNI; remaining ClientHello extensions (e.g., ALPN list) stayed visible for client fingerprinting or service inference. This led to the comprehensive `Encrypted Client Hello (ECH)` protocol, encrypting the entire ClientHello. ECH distributes public keys and parameters via DNS `SVCB/HTTPS` resource records, more appropriate than ESNI's TXT records, bootstrappable over encrypted DNS transports, reducing sensitive metadata leakage. The unencrypted "outer" ClientHello carries only "innocuous" values [1].

2.3. ECH Technical Mechanisms

ECH operates by dividing the TLS ClientHello message into two parts: an "outer" ClientHello and an "inner" ClientHello [4]:

Outer ClientHello: This part is unencrypted and contains innocuous values for sensitive extensions, and a generic "outer SNI" (also known as a public name or fronting domain), extracted from the HTTPS SVCB record, visible to network observers, and used as a fallback if ECH fails.

Inner ClientHello: This part is encrypted using a public key retrieved by the client and contains the actual sensitive extensions, including the real target domain (the "inner SNI").

ECH uses the Hybrid Public Key Encryption (HPKE) standard for inner ClientHello encryption/decryption [5]. The server's ECH configuration, including HPKE public key, cipher suite, and public domain name, is conveyed via DNS. This DNS mechanism bootstraps ECH, allowing clients to discover encryption parameters before TLS handshake initiation.

This split ClientHello structure lets network observers only see the generic public name (e.g., `web.local` in the testbench, or `cloudflare-ech.com` in Cloudflare's deployment) in the outer SNI, while the actual target domain (e.g., `ech.test`) is hidden within the encrypted inner ClientHello [6]. This means multiple unrelated sites can appear indistinguishable to inspection tools if hosted by the same ECH-enabled provider, significantly enhancing privacy by obscuring the destination. This effect is especially relevant for companies like Cloudflare, where all ECH-enabled users could share the same `cloudflare-ech.com` domain. Sites that self-host their entire infrastructure still benefit from this technology, but the fronting domain would still maintain a 1:1 mapping. The privacy benefits in this scenario would not be as great as with shared ECH-compatible providers.

2.4. Challenges and Advanced ECH Mechanisms

Even with ECH, a potential challenge arises if only sensitive services adopt it, creating a "Do not stick out" problem where ECH usage itself becomes identifiable [1]. To address this, ECH-supporting clients always include the ECH extension: either a real ECH extension when server configuration is available, or a GREASE (Generate Random Extensions And Sustain Extensibility) ECH extension to mask which servers support ECH. Additionally, "Implicit ECH" allows clients to choose any outer SNI and config_id, further obfuscating ECH usage and preventing fingerprinting [7].

Implementing Implicit ECH requires servers to perform trial decryption across multiple potential keys, which adds computational complexity but significantly enhances the robustness of ECH against sophisticated traffic analysis and fingerprinting attempts. This demonstrates a nuanced understanding of privacy that extends beyond simple encryption, addressing the challenge of hiding the fact that privacy mechanisms are in use to avoid drawing unwanted attention and prevent network observers trying to filter or flag ECH traffic.

3. ECH Testbench Architecture and Experimental Setup

This section details the containerized testbench architecture designed to demonstrate ECH functionality through direct comparison of encrypted and plaintext TLS handshakes.

3.1. Overview of Testbench Design Principles

The ECH testbench is a self-contained environment demonstrating ECH privacy benefits through direct comparison of plaintext and ECH-TLS traffic, showcasing ECH's SNI and sensitive field protection. The architecture emphasizes simplicity and reproducibility, using containerization for deployment ease and consistent cross-environment setup.

The project supports both `podman compose` and `docker compose`.

3.2. Key Components and Their Functions

The testbench comprises three primary components, each with a distinct role in demonstrating ECH functionality:

3.2.1. DNS Server (CoreDNS). CoreDNS serves DNS records, including HTTPS SVCB records for `ech.test`. This HTTPS record contains ECH configuration (Base64-encoded ECHConfigList) for automatic client discovery of ECH-enabled connections [8]. HTTPS records standardize ECH configuration for service binding. The `coredns/zones/ech.test.zone` file defines the DNS zone with HTTPS record containing `ech` SvcParamKey for ECH configuration.

3.2.2. ECH Server (Nginx). A single Nginx server, specifically compiled with ECH support, handles both normal and ECH-enabled TLS requests to the same endpoint. The server decrypts the inner ClientHello for ECH requests to identify the real target domain (`ech.test`) while processing standard TLS requests normally [5].

ECH support implementation in Nginx requires utilizing an ECH feature branch of OpenSSL and Nginx sources, as native, out-of-the-box ECH support is still maturing. This highlights practical challenges in deploying bleeding-edge privacy protocols, often necessitating development branches or patched software [9].

 doi: 10.2313/NET-2025-11-3_20

3.2.3. ECH Client (curl). The curl client, specifically compiled with ECH support, generates test traffic for both ECH and plain TLS connections. It also facilitates packet capture with `tcpdump`. These `pcap` files can be inspected to see ECH and non-ECH traffic in action. The `clients/curl/scripts/benchmark.sh` script automates traffic generation for both ECH and plain TLS connections, ensuring a controlled and comparable setup.

3.3. Network Layout and Configuration

The testbench operates within a defined isolated network environment to simulate real-world interactions and ensure controlled experimentation. Table 1 summarizes the network configuration.

TABLE 1: Network Layout and IP Addresses

Component	IP Address	Role in Network
DNS Server	13.37.0.53	ECH config, HTTPS RR
ECH-enabled Nginx	13.37.0.10	Web target
Curl Client	13.37.0.50	Sends traffic

3.4. Operational Flow and Experimental Procedure

The ECH-enabled connection follows a distinct sequence of operations designed to ensure the encryption of sensitive ClientHello information:

DNS Query for ECH Configuration: The ECH-enabled client (curl [1]) initiates the process by querying the DNS server (CoreDNS) for the HTTPS record associated with `ech.test` [5]. The server's ECH public key from the ECH configuration is used to encrypt its subsequent ClientHello message.

ECH Config Retrieval: The DNS server responds with the HTTPS record, which includes the ECH configuration. This configuration contains `web.local` as the designated public fronting domain, along with the necessary Hybrid Public Key Encryption (HPKE) public key and cipher suite. The client parses this Base64-encoded configuration to prepare for the encrypted handshake.

ClientHello Construction: The client constructs a TLS handshake message with two distinct parts [4]:

- **Outer ClientHello:** The actual ClientHello that contains non-sensitive information, including `web.local` as the visible SNI and the `encrypted_client_hello` extension (type 65037)
- **Inner ClientHello:** Encrypted section containing the sensitive information like the real target `ech.test` as its SNI, embedded within the ECH extension

Server Processing: The ECH-enabled Nginx server receives the ClientHello, processes the Outer ClientHello, then attempts to decrypt the "inner" ClientHello using its corresponding private key, identified by the ECH config_id. Upon successful decryption, the server identifies the real target domain, presents the appropriate certificate and completes the handshake.

1. As curl currently does not support DoH with untrusted certs, the HTTPS record is manually fetched and passed to curl

3.4.1. Plain TLS Flow (Unencrypted Connection). In contrast, a plain TLS connection follows the traditional, unencrypted handshake process with a standard ClientHello containing the plaintext SNI directly specifying `ech.test`, allowing any on-path observer to extract the target domain.

3.4.2. Traffic Generation and Analysis. The testbench employs specific scripts for traffic generation and analysis. The `benchmark.sh` script automates the generation of both ECH and plain TLS traffic in the client using the ECH-enabled curl build, while the `analyze_traffic.sh` script processes the captured network traffic to extract and compare key fields such as SNI values and the presence or absence of the ECH extension.

4. Results and Discussion

This section presents the experimental findings from packet-level analysis of ECH versus plaintext TLS traffic, demonstrating ECH's effectiveness in concealing sensitive metadata.

4.1. Traffic Analysis: SNI Visibility and ECH Extension Presence

The experimental results from the testbench quantify ECH's privacy protection effectiveness. Running the benchmark will result in pcap files that contain all the network traffic between the client, the DNS, and the Nginx server.

Below are excerpts from the packet captures, which were opened in Wireshark for Mac.

Figure 1 shows the client requesting an `A` (and `AAAA`) record for `ech.test` and the DNS server responding. After the client receives the DNS response, a TCP connection is initiated. After the initial SYN, ACK procedure, the client sends the ClientHello, with the SNI visible (`ech.test`) to network observers.

13.37.0.50	13.37.0.53	DNS	68	Standard query 0x1797 AAAA ech.test
13.37.0.50	13.37.0.53	DNS	68	Standard query 0x2c90 A ech.test
13.37.0.53	13.37.0.50	DNS	138	Standard query response 0x1797 AAAA ech
13.37.0.53	13.37.0.50	DNS	126	Standard query response 0x2c90 A ech.te
13.37.0.50	13.37.0.10	TCP	74	47160 → 443 [SYN] Seq=0 Win=64240 Len=0
13.37.0.10	13.37.0.50	TCP	74	443 → 47160 [SYN, ACK] Seq=0 Ack=1 Win=
13.37.0.50	13.37.0.10	TCP	66	47160 → 443 [ACK] Seq=1 Ack=1 Win=64256
13.37.0.50	13.37.0.10	TLSv1…	583	Client Hello (SNI=ech.test)
13.37.0.10	13.37.0.50	TCP	66	443 → 47160 [ACK] Seq=1 Ack=518 Win=647
13.37.0.10	13.37.0.50	TLSv1…	1494	Server Hello, Change Cipher Spec, Appli

Figure 1: Plaintext TLS showing SNI exposure

Figure 2 shows the ECH-enabled client requesting the `A` (and `AAAA`) records but also the `HTTPS Resource Record`. After receiving the IPv4 and the `ECHConfig`, the client starts the TLS handshake with a ClientHello. This time, as the client used ECH, a network observer can only see the fronting domain `web.local` defined in the `ECHConfig`, hiding the real SNI encrypted in the "inner" ClientHello.

The Nginx server processes both (ECH and non-ECH) connections as requests to `ech.test`, while network observers perceive only the generic `web.local` domain for the ECH-enabled request.

ECH's privacy effectiveness depends on deployment scale: when `ech.test` remains the sole service behind

 doi: 10.2313/NET-2025-11-3_20

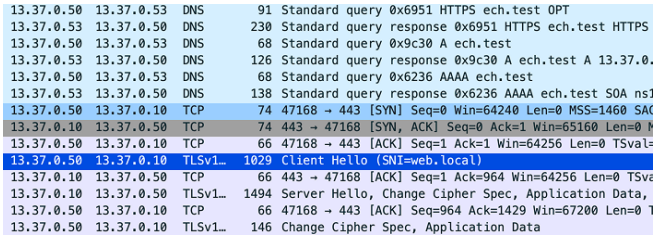

Figure 2: Demonstrating SNI concealment via ECH

a given outer SNI, the 1:1 mapping preserves some fingerprintability. However, when multiple unrelated services share the same fronting domain, such as `web.local` in this testbench or `cloudflare-ech.com` in Cloudflare's production deployment [4], all ECH-protected connections become indistinguishable to external observers, maximizing privacy through traffic aggregation.

4.2. Implications for Network Security

ECH effectively prevents several classes of passive attacks:

SNI-based censorship becomes infeasible as censors cannot identify target domains from ClientHello inspection. **Traffic correlation attacks** are significantly hindered when multiple services share the same outer SNI. **Pervasive monitoring** capabilities are reduced as ISPs and intermediaries lose visibility into user browsing patterns.

However, ECH does not protect against active attacks or endpoint-based monitoring. Network administrators must adapt security strategies toward behavioral analysis rather than content-based inspection of TLS metadata.

5. Related Work

This section positions our testbench contribution within the broader landscape of TLS privacy research and practical ECH evaluation approaches.

Privacy Analysis and Formal Verification. Hoang et al. [10] conducted the first comprehensive empirical study of domain name encryption privacy benefits, analyzing k-anonymity properties of 7.5M domains across nine global vantage points. Only 30% of domains achieve meaningful privacy (k>100), while 20% gain no benefit from one-to-one domain-IP mappings, establishing ESNI effectiveness limitations. Bhargavan et al. [11] provided the first mechanized formal analysis of TLS 1.3 privacy properties with ECH, using ProVerif to define client identity privacy, unlinkability, and extension privacy. They identified early ECH draft vulnerabilities and established theoretical foundations for encrypted handshake security.

Deployment Measurement Studies. Tsiatsikas et al. [12] measured ECH and ESNI adoption across top 1M domains, finding minimal ECH deployment despite theoretical benefits. Less than 19% of domains supported ESNI, with practically no ECH support as of 2023, highlighting deployment gaps.

Practical Implementation and Testing. While Cloudflare and other CDNs have enabled ECH in production environments [4], controlled evaluation frameworks remain limited. Hoang et al. [10] performed large-scale active DNS measurements to assess real-world co-hosting patterns, but focused on privacy quantification rather than demonstrating ECH functionality. Most evaluation approaches rely on theoretical analysis or internet-scale measurements rather than controlled, reproducible testbeds for direct ECH protocol comparison.

Censorship Circumvention Context. Recent work has examined ECH's role in circumventing SNI-based censorship [13], though ECH deployment remains insufficient to provide widespread censorship resistance. Research shows that authoritarian regimes have actively blocked ESNI traffic, emphasizing the need for robust ECH adoption and evaluation frameworks.

6. Conclusion and Future Work

This section synthesizes the key findings from our ECH testbench evaluation, assesses its broader implications for internet privacy, and outlines future research directions.

6.1. Summary of Findings

This paper analyzes Encrypted Client Hello (ECH) privacy benefits through a practical testbench framework demonstrating critical privacy enhancements in modern TLS communications. By comparing ECH-enabled and plain TLS connections, we prove ECH's effectiveness in hiding real target domains from network observers through encrypted inner ClientHello and generic "outer" SNI, cloaking real domain names.

The key findings include:

- ECH effectively conceals target domains (`ech.test`) behind generic public names (`web.local`), especially when multiple sites share the same outer SNI
- The same server infrastructure can seamlessly handle both ECH and plain TLS connections
- ECH presents significant challenges for traditional network security monitoring approaches

6.2. Significance for Internet Privacy

ECH eliminates the last major plaintext metadata leak in TLS 1.3, completing privacy protection by encrypting previously exposed ClientHello fields. The testbench demonstrates protection without functional compromises, maintaining TLS 1.3 compatibility while preventing passive domain monitoring. However, widespread adoption faces implementation complexity, evidenced by required experimental software builds.

Protocol effectiveness scales with adoption density as ECH maximizes privacy when multiple services share common outer SNIs, creating indistinguishable traffic for network observers.

ECH development reflects broader internet protocol design shifts toward privacy-by-design principles, where user privacy becomes fundamental rather than optional. Current deployment by major CDN providers like Cloudflare indicates ECH's transition from experimental to production-ready technology, with advancing browser support. This directly responds to the IETF's recognition of pervasive monitoring as an infrastructure attack.

 doi: 10.2313/NET-2025-11-3_20

6.3. Challenges and Limitations

While ECH provides substantial privacy benefits, challenges remain for successful deployment.

Implementation Complexity: ECH requires systematic DNS infrastructure, server software, and client application updates. Our testbench demonstrates experimental software needs, highlighting maturity challenges.

Adoption Incentives: The "Do not stick out" problem suggests ECH effectiveness depends on widespread adoption to avoid making ECH-enabled traffic a signal to adversaries.

Infrastructural Concentration: ECH offers greatest privacy gains when endpoints are shared, favoring large CDN deployment like Cloudflare, potentially incentivizing further internet infrastructure centralization.

6.4. Future Work and Extensions

Future research directions based on this work include:

Extended Testbench Scenarios: Expanding the testbench to include more complex scenarios such as testing ECH performance under varying network conditions, evaluating compatibility with different TLS proxies or middleboxes, and exploring the implications of Implicit ECH for further obfuscation [7].

Performance Analysis: Comprehensive latency and throughput measurements under varying network conditions, quantifying ECH's operational overhead in production-like environments.

Enterprise Security Solutions: Research into new methods for enterprise network security monitoring in an ECH-pervasive environment would be valuable, potentially exploring endpoint-based security solutions or privacy-preserving analytics techniques that can operate effectively without requiring plaintext SNI access.

Deployment Studies: Large-scale studies of ECH deployment challenges and adoption patterns in real-world environments.

Mitigating Infrastructural Concentration: Examine ECH deployment approaches that lessen reliance on dominant CDNs, thereby expanding the range of participating operators and enhancing end-user privacy.

References

[1] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, "TLS Encrypted Client Hello," Internet Engineering Task Force, Internet-Draft ietf-tls-esni-25, Jun. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-esni/25/

[2] NordVPN, "What is encrypted SNI, and how does it relate to censorship?" https://nordvpn.com/blog/encrypted-sni/, 2024, [Online; accessed June 21, 2025].

[3] S. Farrell and H. Tschofenig, "Pervasive Monitoring Is an Attack," RFC 7258, 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7258.html

[4] Cloudflare, "Encrypted Client Hello - the last puzzle piece to privacy," https://blog.cloudflare.com/announcing-encrypted-client-hello/, 2023, [Online; accessed June 21, 2025].

[5] T. Probst, "Decoding TLS Encrypted Client Hello extension," https://thibautprobst.fr/en/posts/ech/, 2025, [Online; accessed June 21, 2025].

[6] Security.com, "Navigating Encrypted Client Hello (ECH): Insights from RSAC 2025 Conference," https://www.security.com/expert-perspectives/navigating-encrypted-client-hello-ech-insights-rsac-2025, 2025, [Online; accessed June 21, 2025].

[7] N. Sullivan, "Implicit ECH Configuration for TLS 1.3," Internet Engineering Task Force, Internet-Draft draft-sullivan-tls-implicit-ech-00, Feb 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-sullivan-tls-implicit-ech/

[8] B. Schwartz, M. Bishop, and E. Nygren, "Bootstrapping TLS Encrypted ClientHello with DNS Service Bindings," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-svcb-ech-08, Feb 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-svcb-ech/

[9] Nginx, "Request for Encrypted Client Hello (ECH) Support in Nginx," https://github.com/nginx/nginx/issues/266, 2024, gitHub Issue #266, [Online; accessed June 21, 2025].

[10] N. P. Hoang, A. A. Niaki, N. Borisov, P. Gill, and M. Polychronakis, "Assessing the privacy benefits of domain name encryption," *CoRR*, vol. abs/1911.00563, 2019. [Online]. Available: http://arxiv.org/abs/1911.00563

[11] K. Bhargavan, V. Cheval, and C. A. Wood, "A symbolic analysis of privacy for TLS 1.3 with encrypted client hello," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: ACM, 2022, pp. 279–292. [Online]. Available: https://doi.org/10.1145/3548606.3559360

[12] Z. Tsiatsikas, G. Karopoulos, and G. Kambourakis, "Measuring the adoption of TLS encrypted client hello extension and its forebear in the wild," in *Computer Security. ESORICS 2022 International Workshops*, ser. Lecture Notes in Computer Science, vol. 13785. Cham: Springer, 2023, pp. 177–190. [Online]. Available: https://doi.org/10.1007/978-3-031-25460-4_10

[13] S. Wendzel, W. Mazurczyk, L. Caviglione, and A. Mileva, "A survey of internet censorship and its measurement: Methodology, trends, and challenges," 2025. [Online]. Available: https://arxiv.org/abs/2502.14945