Transport Layer Security
Internet-Draft
Intended status: Standards Track
Expires: 4 September 2025

M. Thomson
Mozilla
M. Fayed
Cloudflare

3 March 2025

Public Name Masquerade for TLS Encrypted Client Hello
draft-thomson-tls-ech-pnmasq-00

Abstract

An alternative method for authenticating Encrypted Client Hello (ECH) retry configurations is described. This method enables the use of multiple public names for the same server anonymity set.

About This Document

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

The latest revision of this draft can be found at https://martinthomson.github.io/ech-pnmasq/draft-thomson-tls-ech-pnmasq.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-thomson-tls-ech-pnmasq/.

Discussion of this document takes place on the Transport Layer Security Working Group mailing list (mailto:tls@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tls/. Subscribe at https://www.ietf.org/mailman/listinfo/tls/.

Source for this draft and an issue tracker can be found at https://github.com/martinthomson/ech-pnmasq.

Status of This Memo

This Internet-Draft will expire on 4 September 2025.

Table of Contents

1. Introduction

The TLS Encrypted Client Hello (ECH) [ECH] defines a 'retry' fallback mechanism that is used when a client attempts to use an outdated or incorrect configuration. The retry requires that the server can authenticate its identity to the client. This recovery is an essential feature of ECH, but it can contribute to a reduction in the size of the anonymity set of server identities.

1.1. Privacy in ECH

In a TLS Encrypted Client Hello (ECH) [ECH], the level of privacy is dependent on its deployment. Privacy in ECH is proportional to the number of possible names that could be encrypted in the ClientHelloInner relative to the name(s) in the ClientHelloOuter. This means that privacy is defined by the 'herd' of names, not of users, which contrasts with longer-standing schemes for secure or private communication such as VPNs, Tor, and oblivious proxies, to name a few.

Privacy in ECH is conferred by either or both of the ClientHelloInner and the ClientHelloOuter. For example, 100 names that share the same ECH configuration establish an anonymity set of 100, irrespective of the number of users, i.e., an observer's ability to know the contents of the ClientHelloInner is neither advantaged nor disadvantaged by the number of users connecting to the ClientHelloOuter. However, this setup is dependent on the operator, and confers no additional privacy to one-server one-name deployments.

Alternatively, an ECH deployment may vary the number of names in the ClientHelloOuter. If the set size is finite consisting of public names, then every query for the HTTPS RR may be populated with any of the names selected at random. In this setup an adversary must engage in a "Coupon Collector" exercise to identify all names in the set. Names could also be randomized, which would force every server to decrypt all connections, including for names it knows it does not own.

A natural way to improve privacy maximizes the 'herd' quality by creating an anonymity set consisting of all names from all servers--- effectively rendering every connection indistinguishable to an adversary from every other connection. This would mean sharing a single consistent ECH configuration (Section 4 of [ECH]) across all clients or, alternatively, across all providers and servers.

However, a consistent configuration negates any single server's ability to authenticate itself on the SNI in the ClientHelloOuter. Authentication against a public name is needed so the server can safely invoke a retry mechanism, for example, when a client attempts to use outdated or incorrect configuration. This recovery is an essential feature of ECH that also ensures a server attempts to decrypt only those ECH connections it expects, for example, so that a server for example.com does not attempt to decrypt ECH connections for not-example.com.

The need to authenticate a public name also limits the size of the anonymity set to the number of names available at the server, thereby upper-bounding ECH privacy to its server's deployment.

1.2. Unique Public Names

This document describes an approach that seeks to improve privacy by increasing the number of public names. Rather than having as few public names as possible, it increases the size of the anonymity set for public names by using as many public names as possible.

In the ideal form of this approach, a unique public name is used for each client. In practice, caching of HTTPS records [RFC9460] will ensure that the same public name is likely to be used by some number of clients. A client cannot be sure that a configuration is not available to others, unless it is provided directly through a mechanism like a ECH Retry, as defined in Section 6.1.6 of [ECH].

Any reuse of a name will cluster clients into relatively small anonymity sets. Any clustering will be based on attributes that already leak to a passive observer. This includes the time, the network that a client uses, or the choice of DNS resolver.

The net effect is that the public name is either unique (used for a single connection) or forms a small anonymity set (used for a small number of connections). Assembling observed connection attempts into groups that represent the true anonymity set requires that an adversary obtain mappings for all of the public names that correspond to the same hidden names. If public name is a sufficient-strong encryption of the hidden name, an adversary either needs to receive that mapping or they are only able to use inference (such as website fingerprinting).

An increased diversity of public names leads to a much larger effective anonymity set. An adversary that is ignorant of mappings for each public name effectively observes a single anonymity set that corresponds to every name that it does not know. Both the public name and other ECH configuration values, such as HPKE [RFC9180] parameters, are obscured.

The use of multiple public names can undermine the effectiveness of ECH if poorly implemented; Section 6 includes a discussion of these considerations. This approach also cannot hide the use of IP addresses that correspond to a set of hidden names.

1.3. Limitations

Though diversity in public names could create a larger anonymity set for hidden names, that could be partitioned by information that leaks as part of connection establishment and use. In particular, server IP addresses are not hidden and can be used to distinguish services. Website fingerprinting [WFP] might also be used to recover the identity of hidden sites.

This protection is limited to those public names for which an adversary is unable to recover the mapping to the corresponding hidden names. As mappings could be retained in shared caches at DNS resolvers, this creates a significant risk; see Section 6.2 for details.

The use of multiple public names can undermine the effectiveness of ECH if incautiously implemented. Section 6 includes a discussion of these considerations.

1.4. Alternative Authentication for Public Names

This document defines a new method for validating the certificate that a server proffers during an ECH retry. This enables a retry process that does not depend on the public key infrastructure that is used for server authentication, making it far easier to create new public names.

These names can even overlap with parts of the domain name system. The client determines whether a server is authorized to provide an ECH retry configuration based on the choice of name alone. This removes any need to rely on anything other than the public key that is bound to the name. This obviates any need for revocation checks,

This change to public name authentication is the only aspect of this document that requires client changes. This document describes a server architecture that helps demonstrate the feasibility of this approach, including an analysis of drawbacks; see Section 3.

2. Conventions and Definitions

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

3. Alternative Retry Authentication

ECH defines a retry process for when the ECH configuration that a client uses is rejected by a server. The server completes the handshake using the outer (or unprotected) ClientHello.

By default, the server offers a certificate that is valid for the outer "server_name" that is used. The client then authenticates the handshake using that certificate and the process by which a client would ordinarily validate a server identity.

That ordinary process relies on the client having previously established which name it wishes to authenticate. The name that is used during an ECH retry comes from the ECH configuration. The ECH configuration is typically obtained from the SVCB record [SVCB], which ECH treats as unauthenticated; see Section 10.2 of [ECH].

The client therefore relies on the ECH configuration to choose the public name that it authenticates. This opens the possibility that the ECH configuration can also specify alternative means of authentication.

The public name therefore does not need to be valid according to ordinary client expectations. The public name exists solely to carry information to the server about the anonymity set into which the connection attempt falls. The public name value could even be encrypted; see Section 6.

This document defines an ECH configuration extension, "public_name_authn", that specifies an alternative name and the means by which that alternative name is authenticated.

4. ECH Public Name Masquerade Extension

An extension is defined for ECH that includes a randomized name and information necessary to authenticate the TLS handshake that supplies an ECH retry configuration.

```
struct {
  SignatureScheme scheme;
  opaque spki_hash[32];
} PublicNameAuthentication;
```

Figure 1: public_name_authn Extension Structure

This extension is defined as mandatory, because a server that uses this approach relies on clients applying the alternative authentication method to validate the public name.

Clients MUST NOT use an ECH configuration with this extension unless the connection they establish includes the indicated signature scheme in a "signature_algorithms" extension.

An ECH configuration with this extension refers to a public name that the operator of the client-facing server might not have a valid certificate for; see Section 6.1. Clients that use an ECH configuration with this extension MUST follow the process for authenticating an ECH retry described in Section 5.

> This might be marginally more compatible if the extension were optional. In that case, the extension would have to include the bogus public name as well, which would be less efficient in the longer term.
>
> In the short term, this approach is less efficient as it forces a server that wishes to support clients that do not support this extension to provide additional configurations.

5. Retry Configuration Authentication

A server that rejects an ECH configuration can use a certificate or raw public key [RAW]. Clients extract the subjectPublicKeyInfo, either from the certificate or, for raw public keys, the Certificate message content.

The resulting subjectPublicKeyInfo structure is hashed using SHA-256 and compared to the spki_hash value from the "public_name_authn" extension in the ECH configuration. If the value matches, the retry configuration is accepted. Otherwise, the connection attempt MUST be aborted and any retry configuration that is provided is discarded.

This procedure largely replaces the procedure in Section 6.1.7 of [ECH]. This does not change the requirement that the client not provide a certificate if requested or regard the connection as authenticated for the origin.

6. Deployment

Client-facing server deployments can use dynamically-generated public names using techniques similar to those described in this section.

The use of these techniques requires careful consideration of the privacy risks. Two basic outcomes are worth considering:

* ECH configurations that are never seen by an adversary. If this is achievable, unique public names can be provided to clients, which greatly improves privacy. The net effect can be that only the server IP address leaks information to observers about the extent of the anonymity set.

* ECH configurations are seen by adversaries. This is a more likely situation, because the use of DNS for distribution means that any given ECH configuration is likely to be served to many clients.

Section 6.2 discusses how the choice of public name is important for maintaining the privacy of hidden names in the second case. Section 7 discusses options for constructing public names to minimize the potential risks that come from adversary access to the ECH configuration.

This section addresses other deployment considerations.

6.1. Public Name Selection

The public name used in the ECH configuration does not need to be valid according to any grammar. Any sequence of octets up to the limit for public names (255 bytes) is possible.

Names that appear to be domain names are most likely to be widely compatible. That is, names formed from a sequence LDH labels, as defined in Section 2.3.1 of [IDNA], each joined with periods ('.').

6.2. One-to-One Name Mappings

A simple method for generating public names is to generate a fresh name for every ECH configuration. If every public name is different, then each public name also corresponds to a single hidden name.

Use of such a public name reveals the hidden name to any entity that knows of the relation. Using a unique public name for every hidden name is therely only possible if the ECH configuration can be kept secret.

Unique names are incompatible with distribution using DNS, which involves shared caches at recursive resolvers. An adversary that is able to obtain the ECH configuration from a shared DNS cache would be able to learn the hidden name that corresponds to the included public name.

It is therefore important that any given public name be plausibly associated with multiple hidden names. This can be achieved by increasing the size of the anonymity set, together with the inclusion of information that diversifies the public name.

7. Candidate Processes for Name Selection

A deployment can generate a secret and distribute that secret to all client-facing servers and all authoritative name servers. A corresponding short key identifier might be generated using a counter.

The secret can be split into two parts for use:

* A key for enciphering public names. Section 7.3 describes a sample process.

* A secret for generating authentication keys. Section 7.4 addresses this process.

These secrets need to be distributed to all authoritative DNS resolvers for hidden names and all client-facing servers. There also needs to be a consistent view across these servers of how hidden names correspond to ECH profiles (a concept that this document defines in Section 7.2).

7.1. Key Lifetime

If public names are -- in effect -- encrypted, the lifetime of any keys that are used needs to exceed the lifetime of ECH keys. Otherwise, servers will be unable to recover when clients use old ECH configurations.

The keys used to protect public names only exist to protect the extent of the anonymity set. These keys can be rotated less often than the keys that are used to protect hidden names.

Because these keys are only used if a retry is necessary, it might be possible to ensure that they are only valid during a periodic transition of ECH keys. This depends on being able to propagate ECH configurations to all clients between the time that these keys are emplaced and when the ECH keys are changed out.

7.2. ECH Profiles

These procedures assume that a client-facing server maintains multiple active ECH profiles.

ECH profile includes a config identifier, an HPKE KEM identifier, a HPKE key pair, a set of HPKE symmetric cipher suites, any other extensions, and (optionally) a public name for use with clients that do not support this extension.

Each profile can be allocated an identifier. This identifier needs to be unique within the set of profiles that might be concurrently active.

A client-facing server can limit the number of such profiles it supports at the one time. An identifier that is unique over a larger scope or longer span of time is inadvisable as that makes it more difficult to avoid creating unique name mappings (see Section 6.2).

For a single ECH configuration, each hidden name needs to be mapped to a single profile. This ensures that a client-facing server is able to successfully answer connection attempts by decrypting the inner ClientHello or presenting a retry configuration with acceptable authentication.

A hidden name might also be associated with different profiles and ECH configurations according to deployment needs. In particular, hidden names might be mapped to different profiles, as key pairs are rotated or changes are made to account for different deployment strategies.

With an unmodified ECH deployment, a client-facing server uses the combination of the config_id from the outer "encrypted_client_hello" extension and the public name from the "server_name" extension to recover a profile.

In this design, a client-facing server uses the same information, except that the true profile identifier is encrypted and encoded into these two fields.

7.3. Sample Public Name Generation Method

Public names might be generated by enciphering the profile identifier. The encipher values is then encoded into a public name and, optionally, the configuration identifier field.

If this were to only contain the profile identifier, each profile identifier might produce just one public name. To diversify the set of public names, additional information is encoded in each name.

The amount of entropy included for diversification is limited so that there is a significant chance that different hidden names produce the same public name. This is achieved by hashing the inputs used for diversifying names and taking a limited number of bits from the output.

Suggested inputs that can be used for diversification include:

* The IP address of the DNS client, or the DNS Client Subnet option, if present.

* The current time, rounded to the expected DNS record TTL or a small integer fraction of that time. For instance, with a TTL of 30 seconds the time might be rounded to multiples of 5 seconds.

* A small amount of randomness.

In the following pseudocode '^' is an exclusive OR, '||' is concatenation, and '[a..b]' takes a range of bits from bit 'a' (inclusive, default 0) to bit 'b' (exclusive). This code also uses functions for randomness ('random()'), a collision-resistant hash ('H()'), a pseudorandom function ('prf()'), and encoding a byte sequence as a DNS name ('encode_dns()').

```
k1 = prf(secret, "k1" || key_id)
r = random()[..RANDOM_BITS]
diversity = H(client_ip || time || r)[..DIVERSITY_BITS]
k2 = prf(secret, "k2" || diversity)

d = key_id || k1 ^ (diversity || k2 ^ profile_id)

config_id = d[0..8]
public_name = encode_dns(d[8..])
```

> | Note that the PRF function only needs to produce enough bits to mask the value it obscures using XOR. Any additional bits can be discarded.

| ISSUE: This approach reveals to an observer that two values
| share a diversity value.
|
| Something like format-preserving encryption might be necessary.
| For instance, a Feistel network might ensure that the entire
| value depends on all inputs, including the profile identifier.
|
| The challenge being that the two layers of protection here, to
| protect the profile identifier and the diversity value, would
| each require an application of the network.

7.3.1. Authenticated Encryption

The use of authenticated encryption [AEAD] is not necessary to achieve privacy goals. Authentication identifies any name that is generated without access to the key.

Avoiding authentication ensures that an adversary is unable to use side channels to determine whether any given public name is valid as all public names receive the same treatment.

Authenticated encryption could be used if a deployment is concerned about the cost of responding to connection attempts. This approach could lead to a significant amount of additional load on servers due to the need to generate authentication keys and certificates for every unique public name.

Using authenticated encryption only increases the length of public names; it does not increase the diversity of names. Authentication of names therefore does not affect the potential privacy of public names.

7.3.2. Parameter Selection

An adversary that is able to make a request at about the same time and from the same client IP or subnet will learn the mapping from hidden name to public name. Including more randomness will reduce the odds that the same public name is used for a different hidden name.

The optimal number of random bits that are added (RANDOM_BITS) depends on the number of hidden names that correspond to the same profile identifier, that is, the size of the anonymity set. Larger anonymity sets allow for more diversity in names as there are more public names generated and a higher chance of collision.

For RANDOM_BITS, generating less than twice the number of bits as the base 2 logarithm of the anonymity set size ensures that a collision across the names in the set is highly likely. This value might be set significantly less than this limit to account for the risk that not all hidden names will be in active use.

Combining this information using a hash function, then taking a limited number of bits ensures that the number of public names is limited. This improves the chances that the same public name is generated for different hidden names. The number of bits that are retained (DIVERSITY_BITS) can exceed the number of random bits, but only based on the expected number of different ECH configurations that might be concurrently active for the active names in the anonymity set.

Setting DIVERSITY_BITS to less than twice the base 2 logarithm of the total number of ECH configurations ensures that a collision across the names in the set is highly likely.

The total number of ECH configurations can be determined most reliably using an empirical measure. DNS resolvers might count the number of queries that produce different answers for hidden names with the same profile within the TTL of the resource record.

Alternatively, the number of possible active ECH configurations is the anonymity set size times an estimate of the number of different resolvers. The alternative approach likely produces a larger estimate, so it might be adjusted downward to improve the odds of collisions.

In both cases, a minimum amount random entropy ensures that even small anonymity set has some diversity. A minimum of 5 bits ensures that every hidden name produces public names from 32 different options.

7.3.3. Retry Configuration Generation

The process described here can be greatly simplified for ECH configurations that are generated by a TLS server and sent in the "retry_configs" field.

Every ECH configuration that is provided using "retry_configs" can be unique. This can be achieved by setting DIVERSITY_BITS and RANDOM_BITS to values that are large enough to ensure that the collision risk is negligible. The only relevant consideration is the length of the resulting public name.

If a different scheme is used for generating public names based on how the ECH configurations are delivered, it is necessary to distinguish between the forms. This could use the unprotected key identifier or just the length of the name (using a larger value of DIVERSITY_BITS likely results in a longer name).

7.4. Sample Authentication Key Generation Method

An authentication key might be generated as a function of the public name, and optionally the config_id, using a pseudorandom function (PRF) that is keyed with a secret.

```
authn_secret = prf(secret, public_name)
authn_public_key = pk(authn_secret)
```

8. Security Considerations

Use of this mechanism inherits many of the security considerations from Section 10 of [ECH]. Depending on how it is deployed, it can alter the privacy characteristics, as it obscures the extent of an anonymity set presented by a client-facing server; see Section 10.1 of [ECH].

This makes it far more difficult to get a precise enumeration of the names that correspond to any given anonymity set.

8.1. Recovery of Anonymity Sets

This design only permits linking public names based on what an adversary can observe about the relation between each hidden name and all of the public names that are used for that each hidden name.

Obviously, the reuse of a public name reveals that the connection attempts are accessing the same ECH configuration. The use of ECH still means that uses of those public names could correspond to different hidden names.

To provide a comprehensible view of the true anonymity set, an adversary would need to obtain all public names that are in use across all hidden names in the set. If different names are provided in response to every DNS query from an authoritative resolver, an adversary would -- at best -- need to query every DNS resolver cache that queries that authoritative. Diversifying the choice of name based on client IP or the Client Subnet option ensures that an adversary needs to make multiple queries to obtain all of the mappings.

This makes it far more difficult to get a precise enumeration of the names that correspond to any given anonymity set.

8.2. Configuration Availability

An adversary that is able to obtain every ECH configuration that is ever produced can recover the anonymity set perfectly. This design therefore depends on adversaries being unable to access all ECH configurations.

Clients that use shared DNS caching resolvers are less able to benefit from these protections. Caching resolvers can improve privacy protections by including the Client Subnet option [RFC7871] in DNS queries. Authoritative resolvers are then able to change the public name based on the Client Subnet prefix. An adversary would then need to present an IP address with the same prefix as a target to learn the ECH configuration that the target was presented with.

A shorter TTL on resource records increases the work required by an adversary.

An authoritative DNS resolver that generates public names based on the Client Subnet option makes it easier for an adversary to enumerate all the available public names for a given hidden name. The adversary can easily vary the Client Subnet option that it includes in queries. Whether the space is easily enumerable depends on the minimum of how many different public names might be generated and how many Client Subnet values are in use.

8.3. Unique Mapping To Hidden Names

The use of a unique public name could identify the hidden name. If each hidden name corresponds to a different public name, an adversary that is able to obtain that mapping might reverse the mapping to recover the hidden name from the unprotected "server_name" extension. This attack is described in Section 6.2; potential mitigations are described in Section 7.3.

Providing a unique name mapping in a retry configuration carries no such risk; see Section 7.3.3.

8.4. Client Privacy

The method in Section 6.2 recommends the inclusion of a client IP address or the identity of its DNS recursive resolver in the derivation of a public name. This introduces a potential privacy leak.

Any entity that can observe the TLS handshake and is also able to obtain the same ECH configuration might be able to learn something about the client IP address or DNS resolver from the public name that is used. The client-facing server also obtains the same information with higher certainty.

This risk is partly counteracted by the limited number of bits that are retained when generating public names in Section 7.3. That increases the chances that different inputs result in the same public name, which might help if the adversary has no prior knowledge of the client. However, if an adversary only seeks to improve their confidence in an existing hypothesis of client identity, this is unlikely to be sufficient protection.

A client that uses a tunnel, such as a VPN or proxy, for privacy purposes can avoid leaking unwanted information by accessing a DNS resolver using the tunnel. This is also good practice for the use of this sort of privacy mechanism for reasons other than privacy, such as ensuring that services are selected for proximity to the tunnel egress point rather than proximity to the client.

9. IANA Considerations

This document registers an extension in the TLS "ECH Configuration Extension Registry" established by [ECH].

Value: 0xTBD (value has to be 0x8000 or greater)
Extension Name: public_name_authn
Recommended: Y
Reference: This document
Notes: (none)

10. References

10.1. Normative References

[ECH] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-23, 19 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-23>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

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

10.2. Informative References

[AEAD] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/rfc/rfc5116>.

[IDNA] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/rfc/rfc5890>.

[RAW] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/rfc/rfc7250>.

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/rfc/rfc7871>.

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

[RFC9460] Schwartz, B., Bishop, M., and E. Nygren, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)", RFC 9460, DOI 10.17487/RFC9460, November 2023, <https://www.rfc-editor.org/rfc/rfc9460>.

[SVCB] Schwartz, B., Bishop, M., and E. Nygren, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)", RFC 9460, DOI 10.17487/RFC9460, November 2023, <https://www.rfc-editor.org/rfc/rfc9460>.

[WFP] Goldberg, I., Wang, T., and C. A. Wood, "Network-Based Website Fingerprinting", Work in Progress, Internet-Draft, draft-irtf-pearg-website-fingerprinting-01, 8 September 2020, <https://datatracker.ietf.org/doc/html/draft-irtf-pearg-website-fingerprinting-01>.

Acknowledgments

TODO

Authors' Addresses

Martin Thomson
Mozilla
Email: mt@lowentropy.net

Marwan Fayed
Cloudflare
Email: marwan@cloudflare.com