Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 12, 2019

N. Sullivan
Cloudflare
C. Wood
Apple Inc.

March 11, 2019

Anonymous Tickets for TLS 1.3
draft-sullivan-tls-anonymous-tickets-00

Abstract

This document describes a mechanism that enables unlinkable session resumption for TLS 1.3 without server-side state. In contrast to existing ticket-based resumption in TLS 1.3, wherein servers construct and issue tickets to clients, this document specifies a mechanism by which clients request "anonymous tickets" from servers. When a session is resumed using an anonymous ticket, a server only learns that it previously engaged in a session with some client, rather than linking it to a specific client. Anonymous tickets are only useful for clients with idempotent early data to send.

DISCLAIMER: This draft has not seen any significant security analysis and may contain major errors.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2019.

Table of Contents

1. Introduction

DISCLAIMER: This is a work-in-progress draft and as such has not seen any significant security analysis and may contain major errors.

TLS session tickets enable temporal cross-connection linkability. Per [RFC8446], session tickets are recommended to carry an encryption of the session resumption_master_secret, from which clients and servers derive keying material for subsequent upon resumption. With such tickets, servers can stitch together resumed sessions from the same client over time. This can be exploited to build a profile of client and application behavior.

In this document, we describe a mechanism for TLS 1.3 [RFC8446] by which sessions may be resumed in a way that does compromise client

privacy via cross-connection linkability. It reverses the process in which tickets are granted. Rather than the server construct and issue a ticket to the client, a client requests a blinded or anonymous ticket from the server. Upon resumption with an anonymous ticket, the server can only learn that the session is linked to some client with which a session previously existed. Absent per-client keys for deriving anonymous tickets, servers cannot link resumed sessions to any specific session in the past. Anonymous tickets are built on an oblivious pseudorandom function (OPRF) protocol, described in [I-D.sullivan-cfrg-voprf].

Anonymous tickets MUST only be used when clients have idempotent early data to send. Absent this data, clients SHOULD perform a fresh connection without resumption.

Anonymous tickets only address linkability within TLS. They do not address linkability based on network-layer information such as IP addresses or application-layer information such as HTTP cookies.

1.1. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Preliminaries

The proposal in this document is based on a two party OPRF protocol between a client C and server S. Specifically, we use the OPRF protocol of Jarecki et al. [Jarecki16], which is described in {{!I-D.sullivan-cfrg-voprf} and summarized here for completeness. Let G be a cyclic group of prime order q, and g be a generator of this group. Let k be a OPRF key chosen at random from Z_q, that is, the integers modulo q. Let H1 be a hash function that maps arbitrary input to G, and H2 be a hash function that maps arbitrary input to a fixed-length digest. The OPRF is built on blinded exponentiation, which dates back to Chaum's seminal work on blind signatures for untraceable payments [Chaum]. The actual PRF computation is as follows:

```
F(k, x) = y = H2(x, H1(x)^k)
```

The protocol for this PRF works as follows. Let x be the point of interest to be evaluated.

1. C generates a random element r (the blind) in Z_q, and computes p = (1/r) in Z_q.

2. C computes a = H1(x)^r.

3. C sends a to S.

4. S computes b = a^k = H1(x)^{rk}.

5. S sends b to C.

6. C removes the blind by computing c = b^p = H1(x)^k.

7. C finishes the PRF computation by computing y = H2(x, c).

This protocol is a secure PRF in the random oracle model.

3. Anonymous Ticket Protocol

The key idea for anonymous tickets is that, rather than servers issuing tickets to clients at will, clients request "anonymous tickets" from the server by executing the OPRF protocol outlined in Section 2. The server is free to provide such a ticket or reject the request with a suitable NACK. The protocol for fetching a ticket is outlined below. Let k be a suitable OPRF key owned by the server for constructing anonymous tickets. (Typically, such session ticket "encryption" keys are symmetric. Here, k is asymmetric.)

1. C generates random element x in Z_q as discussed in Section 4.

2. C computes a = H1(x)^r, where r is a random element in Z_q and H1 is the hash function as described in Section Section 2.

3. S computes b = a^k (=(H1(x)^r)^k = H1(x)^{rk}) and sends the result back to C.

4. C removes the blind r by computing b^{r^{-1}} = (a^k)^(r^{-1}) = H1(x)^k.

5. C finishes the PRF computation by computing y = H2(x, H1(x)^k).

At this point, the client has y = F(k, x), where F is the PRF output computed by the OPRF protocol. By the nature of the PRF, y is unpredictable given only x without access to k.

We now describe how clients resume sessions using this anonymous ticket. We will use PSK resumption from TLS 1.3 for illustration purposes.

1. C derives the PSK as sk = HKDF-Expand-Label(y, "anonyous-ticket-psk", "", Hash.length).

2. C builds the PreSharedKeyExtension with a PSKIdentity and PSKBinderEntry, where PSKIdentity = x and PSKBinderEntry = HMAC(sk, Transcript-Hash(ClientHello1[truncated]).

3. S parses the PreSharedKeyExtension and, from x, re-computes y and sk using k. (That is, it computes y = H2(x, H1(x)^k), and then derives sk.)

4. S verifies the PSK binder. If valid, S uses y as the PSK for the session as detailed in [RFC8446].

3.1. Anonymous Ticket Requests and Responses

Anonymous tickets may be requested via an AnonymousTicketRequest post-handshake message, anonymous_ticket_request(TBD). Its structure is shown below.

```
struct {
    opaque identifier<0..255>;
    opaque context<0..2^16-1>;
} AnonymousTicketRequest;
```

- identifier: A unique value for this anonymous ticket request. Clients SHOULD fill this in with a monotonically increasing counter.

- context: An opaque context to be used when generating the anonymous ticket request. Clients and servers may use this context to implement or exchange data to be included in the ticket computation. Clients SHOULD make this field empty if it is not needed.

When requesting an anonymous ticket, the contents of the AnonymousTicketRequest message are populated as follows:

- identifier: A value unique to the ticket request, i.e., a counter.

- context: The value a (=H1(x)^r).

Upon receipt of a TicketRequest message, servers MAY reply with a NewSessionTicket message, shown below as defined in [RFC8446].

```
 struct {
       uint32 ticket_lifetime;
       uint32 ticket_age_add;
       opaque ticket_nonce<1..255>;
       opaque ticket<1..2^16-1>;
       Extension extensions<0..2^16-2>;
   } NewSessionTicket;
```

The new ticket message MUST carry two extensions, ticket_identifer and ticket_context, defined below.

```
   enum {
       ...
       ticket_identifier(TBD),
       ticket_context(TBD+1),
       (65535)
   } ExtensionType;
```

The value of ticket_identifier MUST match that of the corresponding AnonymousTicketRequest identifier field. The value of ticket_context MAY be used by servers to convey ticket context to clients. Its value MUST be empty if the corresponding AnonymousTicketRequest context field is empty. NewSessionTicket.ticket MUST carry the value b (=H1(x)^{rk}). The remaining fields of the NewSessionTicket message are populated as specified in [RFC8446].

3.2. Anonymous Ticket Resumption

When resuming a TLS 1.3 connection using an anonymous ticket, clients do the following:

1. Derive the PSK as outlined in Section 3.

2. Construct the PSKIdentity as outlined in Section 3. Specifically, set PSKIdentity.identity to x, and PSKIdentity.obfuscated_ticket_age to (R + NewSessionTicket.ticket_age_add) mod 2^32, where R is a random value sampled from (now(), now() + NewSessionTicket.ticket_lifetime]. (This value MUST be sampled uniformly from this range.)

3. Use the PSK and PSKIdentity to construct a PreSharedKeyExtension extension for the ClientHello.

4. Include an empty anonymous_tickets extension in the ClientHello.

Upon receipt of such a ClientHello, servers SHOULD derive the PSK as outlined in Section 3, verify the PSK binder, and proceed with the connection according to [RFC8446].

3.3. Negotiation

Clients negotiate use of anonymous tickets via a new ExtensionType, anonymous_tickets(TBD). The extension_data for this extension MUST be empty, i.e., have length of 0. Servers that support ticket requests MAY echo this extension in the EncryptedExtensions, and SHOULD not send NewSessionTickets without receiving an AnonymousTicketRequest message from the client. Clients MUST NOT send anonymous ticket requests to servers that do not signal support for this message. If absent from a ClientHello, servers MUST NOT generate responses to AnonymousTicketRequests issued by the client.

4. Token Derivation

Anonymous ticket generation requires the offered value x to be unlinkable to any session state stored by client or server. Thus, one approach would be to randomly generate x independent of session state. While the PRF security properties ensure that F(k, x) is unpredictable given only x, a faulty or backdoored PRNG could be exploited to force clients to produce repeated values of x. This could be exploited to identify and link compromised clients, for example.

Thus, the token should be generated from a secret into which both the client and server have contributed randomness. This can be done by exporting a secret s from the exporter_master_secret, i.e., s = TLS-Exporter("anonymous-ticket", exporter_master_secret, Hash.length), where Hash is the negotiated hash function.

1. Generate a random bit string r of length equal to Hash.length.

2. Compute seed = HKDF-Expand(s XOR r, anonymous-ticket-input", Hash.length).

3. Compute x = H1(seed), i.e., hash seed into a the group G of order q.

This algorithm ensures that x is derived from randomness that is at least as good as the traffic secret. In the event that the client's PRNG fails or is subverted, x will still maintain entropy at least as high as exporter_master_secret.

5. 0-RTT Data Implications

Since clients obfuscate their view of the blind session ticket lifetime, servers MAY inadvertently determine that the resumption Client Hello is a replay or otherwise stale. This would regress the handshake to a full handshake, at the cost of privacy.

6. Linkability Beyond TLS

This section describes linkability concerns outside of TLS. Further discussion of this topic may be found in [I-D.wood-linkable-identifiers].

6.1. Network-Layer Linkability

Servers may link client TLS sessions together using information that exists outside of TLS. If a client uses a fixed and globally unique IPv6 address, for example, then a server may easily link a resumed session to past sessions. Thus, globally unique addresses render blind ticket resumption useless. Clients MUST use temporary IIDs for each resumed connection to the same host, as per RFC 7721. For similar reasons, IPv4-only clients SHOULD update their address or NAT binding for each resumed connection.

6.2. Application-Layer Linkability

Linkability above TLS is also a concern and prevalent problem, especially on the Web. Clients may be tracked via client-side identifiers such as cookies. See [TrackingOnTheWeb] and [ClientIdentification] for more details about these techniques. They may also be fingerprinted via mechanisms that do not rely on such identifiers [CookielessMonster13]. Clearly, these mechanisms may be used to link clients across session resumptions. However, in cases where these techniques are either ineffective or inaccurate, blind TLS session tickets allow clients to maintain some control over their privacy.

7. Open Issues

There are several open issues to consider for this proposal. In no particular order:

1. Why not use a simpler semi-static Diffie Hellman design? In particular, the semi-static Diffie Hellman design in [I-D.rescorla-tls-semistatic-dh] could be modified to support 0-RTT encryption using the server's semi-static key, thereby achieving similar anonymity properties. Such a design would be considerably simpler than that which is presented in this

document. One property that anonymous tickets provide which is not attainable by the semi-static approach is that tickets may only be used once per resumed connection. In contrast, clients may use a semi-static key share for "anonymous" connections with early data without limit.

2. Where is server-side state such as the negotiated ciphersuite and ALPN values stored? One possibility is to have servers store an OPRF key per (ciphersuite, ALPN token) tuple, and require clients to use the same values upon resumption. Assuming servers select the same values and identify the same OPRF key, the PSK binders will match.

8. IANA Considerations

This document makes no IANA requests (yet).

9. Security Considerations

This design has not yet received any security analysis. It may be completely broken.

10. Acknowledgments

The authors would like to thank Martin Thomson for helpful discussions that led to this design.

11. References

11.1. Normative References

[I-D.sullivan-cfrg-voprf]
Davidson, A., Sullivan, N., and C. Wood, "Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups", draft-sullivan-cfrg-voprf-03 (work in progress), March 2019.

[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/info/rfc2119>.

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

11.2. Informative References

[Chaum] "Blind Signatures for Untraceable Payments", n.d., <http://sceweb.sce.uhcl.edu/yang/teaching/csci5234WebSecurityFall2011/Chaum-blind-signatures.PDF>.

[ClientIdentification] "Technical analysis of client identification mechanisms", n.d., <https://www.chromium.org/Home/chromium-security/client-identification-mechanisms>.

[CloudflareResumption] "TLS Session Resumption - Full-speed and Secure", n.d., <https://blog.cloudflare.com/tls-session-resumption-full-speed-and-secure/>.

[CookielessMonster13] "Cookieless Monster - Exploring the Ecosystem of Web-based Device Fingerprinting", n.d., <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6547132>.

[I-D.rescorla-tls-semistatic-dh] Rescorla, E., Sullivan, N., and C. Wood, "Semi-Static Diffie-Hellman Key Establishment for TLS 1.3", draft-rescorla-tls-semistatic-dh-00 (work in progress), October 2018.

[I-D.wood-linkable-identifiers] Wood, C., "Linkable Identifiers", draft-wood-linkable-identifiers-00 (work in progress), October 2018.

[Jarecki16] "Highly-Efficient and Composable Password-Protected Secret Sharing (Or How to Protect Your Bitcoin Wallet Online)", n.d., <https://eprint.iacr.org/2016/144.pdf>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[Springall16] "Measuring the Security Harm of TLS Crypto Shortcuts", n.d., <https://aaspring.com/imc2016/crypto-shortcuts.pdf>.

[TrackingOnTheWeb] "Detecting and Defending Against Third-Party Tracking on the Web", n.d., <http://www.franziroesner.com/pdf/webtracking-NSDI2012.pdf>.

Authors' Addresses

Nick Sullivan
Cloudflare
101 Townsend St
San Francisco
United States of America

Email: nick@cloudflare.com

Christopher A. Wood
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: cawood@apple.com