Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: October 22, 2019

B. Bradley
Apple Inc.

April 20, 2019

Private Discovery
draft-bradley-dnssd-private-discovery-01

Abstract

This document specifies a mechanism for advertising and discovering in a private manner.

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 October 22, 2019.

Table of Contents

1. Introduction

Advertising and discovering with Bonjour can leak a lot of information about a device or person, such as their name, the types of services they provide or use, and persistent identifiers. This information can be used to identify and track a person's location and daily routine (e.g. buys coffee every morning at 8 AM at Starbucks on Main Street). It can also reveal intimate details about a person's behavior and medical conditions, such as discovery requests for a glucose monitor, possibly indicating diabetes.

This document specifies additions to Bonjour to retain the same level of advertising and discovery functionality while preserving privacy and confidentiality.

This document does not specify how keys are provisioned. Provisioning keys is complex enough to justify its own document(s). This document assumes each peer has a long-term asymmetric key pair (LTPK and LTSK) and communicating peers have each other's long-term asymmetric public key (LTPK).

2. Conventions and Terminology

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].

"Friend"
: A peer you have a cryptographic relationship with. Specifically, that you have the peer's LTPK.

"Probe"
: Unsolicited multicast message sent to find friends on the network.

"Announcement"
: Unsolicited multicast message sent to inform friends on the network that you have become available or have updated data.

"Response"
: Solicited unicast message sent in response to a probe or announcement.

"Query"
: Unsolicited unicast message sent to get specific info from a peer.

"Answer"
: Solicited unicast message sent in response to a query to provide info or indicate the lack of info.

"Multicast"
: This term is used in the generic sense of sending a message that targets 0 or more peers. It's not strictly required to be a UDP packet with a multicast destination address. It could be sent via TCP or some other transport to a router that repeats the message via unicast to each peer.

"Unicast"
: This term is used in the generic sense of sending a message that targets a single peer. It's not strictly required to be a UDP packet with a unicast destination address.

Multi-byte values are encoded from the most significant byte to the least significant byte (big endian).

When multiple items are concatenated together, the symbol "||" (without quotes) between each item is used to indicate this. For example, a combined item of A followed by B followed by C would be written as "A || B || C".

3. Protocol

There are two techniques used to preserve privacy and provide confidentiality in this document. The first is announcing, probing, and responding with only enough info to allow a peer with your public key to detect that it's you while hiding your identity from peers without your public key. This technique uses a fresh random signed with your private key using a signature algorithm that doesn't reveal your public key. The second technique is to query and answer in a way that only a specific friend can read the data. This uses ephemeral key exchange and symmetric encryption and authentication.

3.1. Probe

A probe is sent via multicast to discover friends on the network. A probe contains a fresh, ephemeral public key (EPK1), a timestamp (TS1), and a signature (SIG1). This provides enough for a friend to identify the source, but doesn't allow non-friends to identify it.

Probe Fields:

- EPK1 (Ephemeral Public Key 1).
- TS1 (Timestamp 1). See Timestamps Section 4.
- SIG1 (Signature of "Probe" || EPK1 || TS1 || "End").

When a peer receives a probe, it verifies TS1. If TS1 is outside the time window then it SHOULD be ignored. It then attempts to verify SIG1 with the public key of each of its friends. If verification fails for all public keys then it ignores the probe. If a verification succeeds for a public key then it knows which friend sent the probe. It SHOULD send a response to the friend.

3.2. Response

A response contains a fresh, ephemeral public key (EPK2) and a symmetrically encrypted signature (ESIG2). The encryption key is derived by first generating a fresh ephemeral public key (EPK2) and its corresponding secret key (ESK2) and performing Diffie-Hellman (DH) using EPK1 and ESK2 to compute a shared secret. The shared secret is used to derive a symmetric session key (SSK2). A signature of the payload is generated (SIG2) using the responder's long-term secret key (LTSK2). The signature is encrypted with SSK2 (ESIG2). The nonce for ESIG2 is 1 and is not included in the response. The response is sent via unicast to the sender of the probe.

When the friend that sent the probe receives the response, it performs DH, symmetrically verifies ESIG2 and, if successful, decrypts it to reveal SIG2. It then tries to verify SIG2 with the public keys of all of its friends. If a verification succeeds for a public key then it knows which friend sent the response. If any steps fail, the response is ignored. If all steps succeed, it derives a session key (SSK1). Both session keys (SSK1 and SSK2) are remembered for subsequent communication with the friend.

Response Fields:

- EPK2 (Ephemeral Public Key 2).
- ESIG2 (Encrypted Signature of "Response" || EPK2 || EPK1 || TS1 || "End").

Key Derivation values:

- SSK1: HKDF-SHA-512 with Salt = "SSK1-Salt", Info = "SSK1-Info", Output size = 32 bytes.
- SSK2: HKDF-SHA-512 with Salt = "SSK2-Salt", Info = "SSK2-Info", Output size = 32 bytes.

3.3. Announcement

An announcement indicates availability to friends on the network or if it has update(s). It is sent whenever a device joins a network (e.g. joins WiFi, plugged into Ethernet, etc.), its IP address changes, or when it has an update for one or more of its private Bonjour records (but not for public Bonjour records since those are handled using non-private Bonjour methods). Announcements are sent via multicast.

Announcement Fields:

- EPK1 (Ephemeral Public Key 1).
- TS1 (Timestamp 1). See Timestamps Section 4.
- SIG1 (Signature of "Announcement" || EPK1 || TS1 || "End").

When a peer receives an announcement, it verifies TS1. If TS1 is outside the time window then it SHOULD be ignored. It then attempts to verify SIG1 with the public key of each of its friends. If verification fails for all public keys then it ignores the probe. If a verification succeeds for a public key then it knows which friend sent the announcement.

3.4. Query

A query is sent via unicast to request specific info from a friend. The query data (MSG1) is encrypted with the symmetric session key (SSK1 for the original prober or SSK2 for the original responder) for the target friend previously generated via the probe/response exchange. This encrypted field is EMSG1. The nonce for EMSG1 is 1 larger than the last nonce used with this symmetric key and is not included in the query. For example, if this is the first message sent to this friend after the probe/response then the nonce would be 2. The query is sent via unicast to the friend.

When the friend receives a query, it symmetrically verifies EMSG1 against every active session's key and, if one is successful (which also identifies the friend), it decrypts the field. If verification fails, the query is ignored, If verification succeeds, the query is processed.

Query Fields:

- EMSG1 (Encrypted query data).

3.5. Answer

An answer is sent via unicast in response to a query from a friend. The answer data (MSG2) is encrypted with the symmetric session key of the destination friend (SSK1 it was the original prober or SSK2 if it was the original responder from the previous probe/response exchange). This encrypted field is EMSG2. The nonce for EMSG2 is 1 larger than the last nonce used with this symmetric key and is not included in the answer. For example, if this is the first message sent to this friend after the probe/response then the nonce would be 2. The answer is sent via unicast to the friend.

When the friend receives an answer, it symmetrically verifies EMSG2 against every active session's key and, if one is successful (which also identifies the friend), it decrypts the field. If verification fails, the answer is ignored, If verification succeeds, the answer is processed.

Answer Fields:

- EMSG2 (Encrypted answer data).

4. Timestamps

A timestamp in this document is the number of seconds since 2001-01-01 00:00:00 UTC. Timestamps sent in messages SHOULD be randomized by +/- 30 seconds to reduce the fingerprinting ability of observers. A timestamp of 0 means the sender doesn't know the current time (e.g. lacks a battery-backed RTC and access to an NTP server). Receivers MAY use a timestamp of 0 to decide whether to enforce time window restrictions. This can allow discovery in situations where one or more devices don't know the current time (e.g. location without Internet access).

A timestamp is considered valid if it's within N seconds of the current time of the receiver. The RECOMMENDED value of N is 900 seconds (15 minutes) to allow peers to remain discoverable even after a large amount of clock drift.

5. Implicit Nonces

The nonces in this document are integers that increment by 1 for each encryption. Nonces are never included in any message. Including nonces in messages would enable transactions to be easily tracked by following nonce 1, 2, 3, etc. This may seem futile if other layers of the system also leak trackable identifiers, such as IP addresses, but those problems can be solved by other documents. Random nonces could avoid tracking, but make replay protection difficult by requiring the receiver to remember previously received messages to detect a replay.

One issue with implicit nonces and replay protection in general is handling lost messages. Message loss and reordering is expected and shouldn't cause complete failure. Accepting nonces within N of the expected nonce enables recovery from some loss and reordering. When a message is received, the expected nonce is checked first and then nonce + 1, nonce - 1, up to nonce +/- N. The RECOMMENDED value of N is 8 as a balance between privacy, robustness, and performance.

6. Re-keying and Limits

Re-keying is a hedge against key compromise. The underlying algorithms have limits that far exceed reasonable usage (e.g. 96-bit nonces), but if a key was revealed then we want to reduce the damage by periodically re-keying.

Probes are periodically re-sent with a new ephemeral public key in case the previous key pair was compromised. The RECOMMENDED maximum probe ephemeral public key lifetime is 20 hours. This is close to 1 day since people often repeat actions on a daily basis, but with some

leeway for natural variations. If a probe ephemeral public key is re-generated for other reasons, such as joining a WiFi network, the refresh timer is reset.

Session keys are periodically re-key'd in case a symmetric key was compromised. The RECOMMENDED maximum session key lifetime is 20 hours or 1000 messages, whichever comes first. This uses the same close-to-a-day reasoning as probes, but adds a maximum number of messages to reduce the potential for exposure when many messages are being exchanged. Responses SHOULD be throttled if it appears that a peer is making an excessive number of requests since this may indicate the peer is probing for weaknesses (e.g. timing attacks, ChopChop-style attacks).

7. Message Formats

Messages defined by this document are use Type-Length-Value (TLV) payloads with an 8-bit type and a 16-bit length (TLV8x16). It has the following format.

7.1. TLV Structure

Field	Size (bytes)	Description
Type	1	Identifies a value type as defined in Section 7.2.
Length	1 or 2	Length of the value field in bytes.
Value	Variable	Value formatted based on the type field.

7.2. TLV Items

The following lists the TLV items defined by this document.

Type	Name	Description
0x00	Reserved	Reserved to protect against accidental zeroing.
0x01	Type	Type of message. See Section 7.3.
0x02	EPK	Ephemeral Public Key. 32-byte Curve25519 public key.
0x03	TS	Timestamp. 4-byte timestamp. See Timestamps Section 4.
0x04	SIG	Signature. 64-byte Ed25519 signature.
0x05	ESIG	Encrypted signature. Ed25519 signature encrypted with ChaCha20-Poly1305. Formatted as the 64-byte encrypted portion followed by a 16-byte MAC (96 bytes total).
0x06	EMSG	Encrypted message. Message encrypted with ChaCha20-Poly1305. Formatted as the N-byte encrypted portion followed by a 16-byte MAC (N + 16 bytes total).
0x07-0xFF		Reserved for future use. Types in this range MUST NOT be sent. If they are received, they MUST be ignored. This is to allow future versions of document or other documents to define new types without breaking parsers.

7.3. Message Types

Name	Type	Description
Invalid	0	Invalid message type. Avoid misinterpreting zeroed memory.
Probe	1	See Section 3.1.
Response	2	See Section 3.2.
Announcement	3	See Section 3.3.
Query	4	See Section 3.4.
Answer	5	See Section 3.5.

8. Security Considerations

- Privacy considerations are specified in draft-cheshire-dnssd-privacy-considerations.

- Ephemeral key exchange uses elliptic curve Diffie-Hellman (ECDH) with Curve25519 as specified in [RFC7748].
- Signing and verification uses Ed25519 as specified in [RFC8032].
- Symmetric encryption uses ChaCha20-Poly1305 as specified in [RFC7539].
- Key derivation uses HKDF as specified in [RFC5869] with SHA-512 as the hash function.
- Randoms and randomization MUST use cryptographic random numbers.

Information leaks may still be possible in some situations. For example, an attacker could capture probes from a peer they've identified and replay them elsewhere within the allowed timestamp window. This could be used to determine if a friend of that friend is present on that network.

The network infrastructure may leak identifiers in the form of persistent IP addresses and MAC addresses. Mitigating this requires changes outside of Bonjour, such as periodically changing IP addresses and MAC addresses.

9. IANA Considerations

The TLV and message types defined by this document are intended to be managed by IANA.

10. To Do

The following are some of the things that still need to be specified and decided:

- Figure out how sleep proxies might work with this protocol.
- Define probe and announcement random delays to reduce collisions.
- Describe when to use the same EPK2 in a response to reduce churn on probe/response collisions.
- Consider randomly answering probes for non-friends to mask real friends.

11. Normative References

[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>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC7539] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 7539, DOI 10.17487/RFC7539, May 2015, <https://www.rfc-editor.org/info/rfc7539>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

Author's Address

Bob Bradley
Apple Inc.
One Apple Park Way
Cupertino CA 95014
USA

Email: bradley@apple.com